计 算 机 科 学 丛 书

CAMBRIDGE

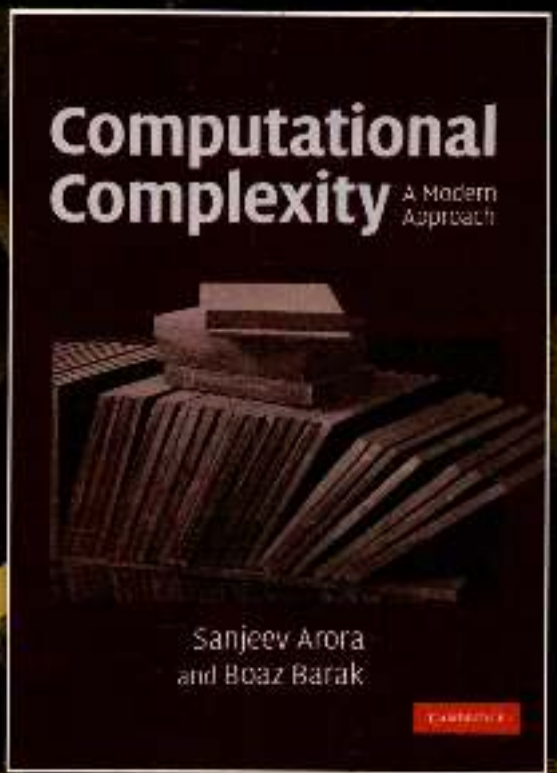
计算复杂性

现代方法

[美] 桑杰夫·阿罗拉（Sanjeev Arora）
博阿兹·巴拉克（Boaz Barak） 著
骆吉洲 译

Computational Complexity

A Modern Approach



机械工业出版社
China Machine Press

计算复杂性 现代方法

Computational Complexity A Modern Approach

计算复杂性理论是理论计算机科学研究的核心。本书是关于包含了计算复杂性领域近30年来所有令人兴奋的成果，是对此领域感兴趣的读者的必读书籍。

阿维·维德森（Avi Wigderson），普林斯顿大学数学学院高级研究所教授

本书综述了复杂性理论的所有重大成果，对学生和研究者来说是非常有用的资源。

——迈克尔·西普塞（Michael Sipser），麻省理工学院数学系教授

本书既描述了计算复杂性理论最近取得的成果，也描述了其经典结果。具体内容包括：图灵机的定义和基本的时间、空间复杂性类，概率型算法，交互式证明，密码学，量子计算，具体的计算模型的下界（判定树、通信复杂度、恒定的深度、代数和单调线路、证明复杂度），平均复杂度和难度放大，去随机化和伪随机数产生器，以及PCP定理。

本书仅要求读者有完备的数学知识，可以作为任何对计算复杂性感兴趣的读者的自学参考书，包括物理学家、数学家和其他科学家，也可以作为各种课程和研讨会的教科书。

作者简介



桑杰夫·阿罗拉（Sanjeev Arora） 普林斯顿大学计算机科学系教授，在概率可验证和NP-难问题的可近似性方面取得了基础性的研究成果。他发起创办了“计算难解性问题中心”，该项目由国家科学基金资助。



博阿兹·巴拉克（Boaz Barak） 现为哈佛大学计算机科学系教授，哈佛大学工学院计算理论研究组成员，同时还是微软新英格兰研究院首席研究员，之前是普林斯顿大学计算机科学系副教授。他在计算复杂性和密码学方面，特别是“非黑盒”技术方面，取得了基础性的研究成果。

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org

上架指导：计算机/计算复杂性

ISBN 978-7-111-51899-0

9 787111 518990 >

定价：129.00元

投稿热线：(010) 88379604
读者热线：(010) 88379711 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

计 算 机 科 学 丛 书

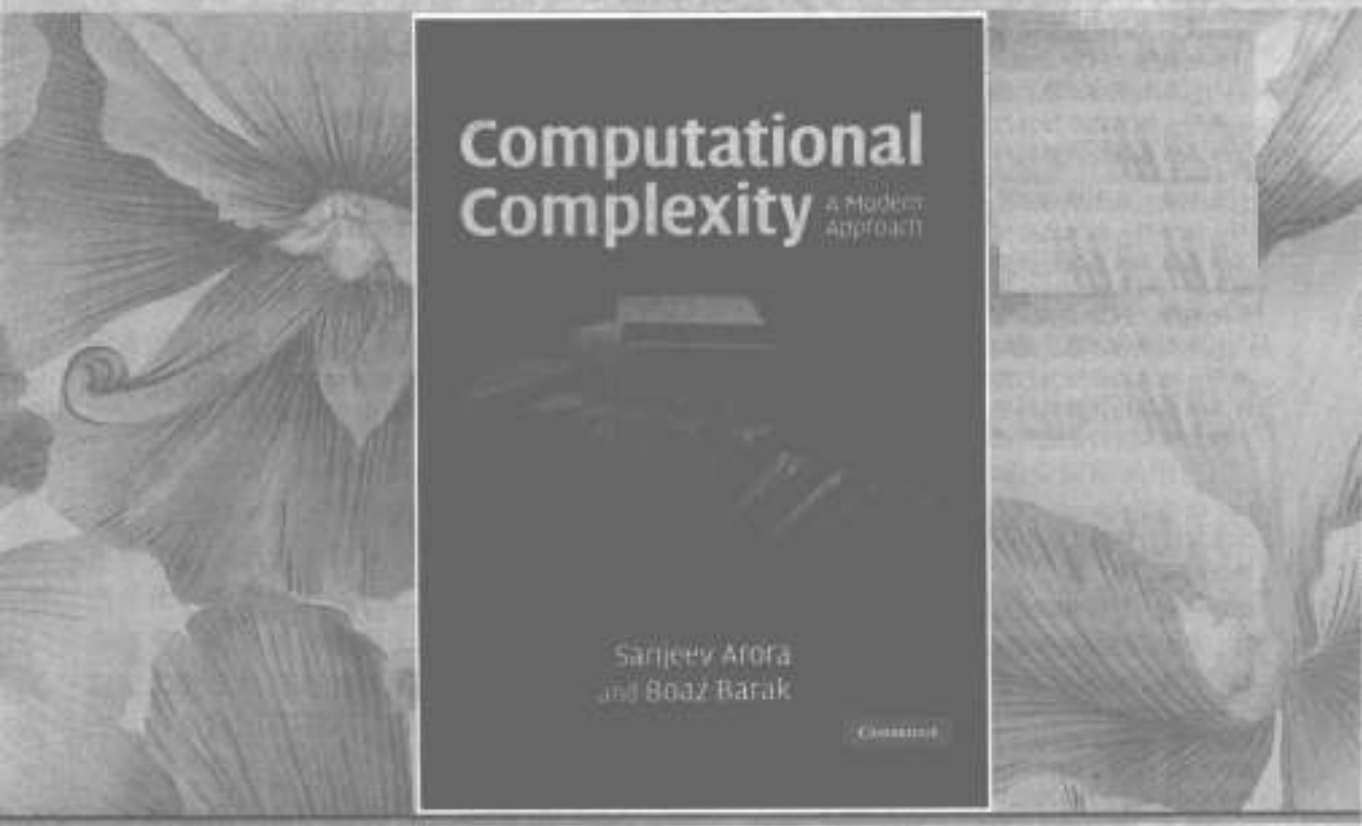
计算复杂性

现代方法

[美] 桑杰夫·阿罗拉（Sanjeev Arora）
博阿兹·巴拉克（Boaz Barak） 著
骆吉洲 译

Computational Complexity

A Modern Approach



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

计算复杂性：现代方法 /（美）阿罗拉（Arora, S.），（美）巴拉克（Barak, B.）著；骆吉洲译．
—北京：机械工业出版社，2015.11
（计算机科学丛书）
书名原文：Computational Complexity：A Modern Approach

ISBN 978-7-111-51899-0

I. 计… II. ①阿… ②巴… ③骆… III. 计算复杂性 IV. TP301.5

中国版本图书馆 CIP 数据核字（2015）第 253023 号

本书版权登记号：图字：01-2012-3791

Sanjeev Arora and Boaz Barak: Computational Complexity, A Modern Approach(ISBN 978-0-521-42426-4).

© Sanjeev Arora and Boaz Barak 2009.

This simplified Chinese for the People's Republic of China (excluding Hong Kong, Macau and Taiwan) is published by arrangement with the Press Syndicate of the University of Cambridge, Cambridge, United Kingdom.

© Cambridge University Press and China Machine Press in 2016.

This simplified Chinese is authorized for sale in the People's Republic of China (excluding Hong Kong, Macau and Taiwan) only. Unauthorized export of this simplified Chinese is a violation of the Copyright Act. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of Cambridge University Press and China Machine Press.

本书原版由剑桥大学出版社出版。

本书简体字中文版由剑桥大学出版社与机械工业出版社合作出版。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

此版本仅限在中华人民共和国境内（不包括香港、澳门特别行政区及台湾地区）销售。

本书系统地介绍计算复杂性理论的经典结果和近 30 年来取得的新成果，旨在帮助读者了解和掌握复杂性理论中的基本结果、思维方法、主要工具、研究前沿和待决问题。本书分三部分。第一部分（第 1 ~ 11 章）较宽泛地介绍了复杂性理论，包括复杂性理论的经典结果和一些现代专题。第二部分（第 12 ~ 16 章）讨论了各种具体计算模型上的计算复杂性下界。第三部分（第 17 ~ 23 章）主要是 1980 年以后人们在复杂性理论方面获得的进展，内容包括计数复杂性、平均复杂性、难度放大、去随机化和伪随机性、PCP 定理的证明以及自然证明。

本书内容丰富，结构灵活，语言流畅，是从事计算复杂性理论及相关领域的研究人员必不可少的参考书，非常适合作为打算进入该研究领域的研究生、博士生快速接触研究前沿的参考资料，还非常适合作为普通高校计算机科学与技术、数学专业本科生、研究生相关课程的教材，其中的高级专题还可以作为博士生相关讨论班的素材。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）

责任编辑：和　静	责任校对：殷　虹
印　刷：北京市荣盛彩色印刷有限公司	版　次：2016 年 1 月第 1 版第 1 次印刷
开　本：185mm × 260mm　1/16	印　张：31.25
书　号：ISBN 978-7-111-51899-0	定　价：129.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

版权所有・侵权必究
封底无防伪标均为盗版
本书法律顾问：北京大成律师事务所　韩光 / 邹晓东

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：(010) 88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章教育
华章科技图书出版中心

译者序

Computational Complexity, A Modern Approach

阿兰·图灵（Alan Turing）等人对计算的精确定义导致了现代电子计算机的诞生。如今，计算机早已融入社会管理、经济活动、工程实践、军事活动、休闲娱乐等现代生活的方方面面，各种计算机软件精彩纷呈、层出不穷。可以说，计算无时无刻地发生在我们的周围。对各种计算问题的计算过程所消耗的时间/空间等资源数量的多少进行量化，进而对各种计算问题进行分类，并研究各类计算问题之间的相互联系，研究近似求解无法精确求解的问题的难度，力争最终解决计算中最核心的问题，围绕这些任务逐渐发展和形成的理论、技术和方法，形成了理论计算机科学中的一门基础性学科——计算复杂性理论。它是为各种计算问题合理地选择算法、配置资源并进行软件开发活动的基础。

计算复杂性理论形成于20世纪五六十年代。1960年，哈特马尼斯（Hartmanis）和斯特恩斯（Stearns）在他们的开创性论文“On the computational complexity of algorithms”中引入了时间复杂性类并利用对角线方法证明了时间分层定理，由此奠定了计算复杂性理论的基础。在其后的三十年中，人们逐渐得到了各种基本复杂性类、**NP**完全理论等经典结论，并提出了计算复杂性理论中最核心的问题**P**≠**NP**。在过去三十多年中，计算复杂性理论发展迅速。自1990年以来，人们取得了大量出人意料的结果和基础性的结果，这些结果涉及的领域非常广泛，包括：经典复杂性类的概率型新定义（**IP**＝**PSPACE**和各种**PCP**定理）以及它们在近似算法中的应用，肖尔（Shor）为量子计算机设计的整数因数分解算法，对人们目前处理著名的**P**≠**NP**问题的各种方法为什么未能获得成功的理解，去随机化理论和基于计算难度的伪随机性，以及随机性提取器和扩张图等伪随机对象的优美构造。

作为计算机科学与技术相关专业的学生，全面系统地学习计算复杂性中的概念、基本结果、思维方法和重要工具并了解一些悬而未决的问题是十分必要的。本书正是适合于上述目标的一部优秀教科书。

本书作者桑杰夫·阿罗拉（Sanjeev Arora）和博阿兹·巴拉克（Boaz Barak）都是在普林斯顿大学计算机科学系一直从事复杂性理论研究的著名教授。桑杰夫·阿罗拉在概率可验证明**NP**-难问题的可近似性方面取得了基础性的研究成果。博阿兹·巴拉克在计算复杂性理论和密码学方面，特别是“非黑盒”技术方面，也取得了基础性的研究成果。本书已经逐步成为国内外计算复杂性理论课程的标准教材，其翻译和出版对国内读者学习和应用复杂性理论具有重要的意义。有幸承担该书的翻译工作，我们感到十分荣幸。

本书旨在介绍计算复杂性理论的基本概念、经典结果、近年来取得的有用的结果，帮助读者理解和掌握计算复杂性理论中的思维方法、主要工具和研究前沿。基础概念和经典结果可以帮助读者建立计算复杂性理论的知识框架，掌握复杂性理论的思维方法和证明技巧。高级专题是经典结果的有益补充和延伸，而最新的研究成果和悬而未决的问题则可以帮助读者接触计算复杂性理论研究的最前沿。此外，本书还涉及了一些学术界尚存的争论，这些深入分析和深刻见解也是本书的精髓所在。全书特别强调计算复杂性理论的各种概念的直观含义，阐述它们在何种场合下是有用的，以及为什么这些概念要这样定义。全书围绕两条主要线索进行组织：其一是人们所尝试的用于处理**P**≠**NP**问题的各种方法以

及对这些方法的局限性的阐释；其二是逐步准备证明 **PCP** 定理所需的各种素材，最终完成 **PCP** 定理的证明。这种组织使得本书内容丰富，结构灵活，可供不同层次的读者使用。

译者翻译时在深刻理解全书内容的基础上力求准确，对于发现的多处笔误和印刷错误进行了更正。在本书的翻译过程中，译者得到了多位同事、朋友和家人的支持和帮助，他们对译稿提出了很多中肯的意见和建议，使译者受益匪浅。在此一并向他们表示感谢！

限于译者水平，译文中疏漏和错误难免，敬请读者批评指正。如有任何建议，请发送邮件至 luojizhou@hit.edu.cn。

译者

2015 年 10 月

译者简介

Computational Complexity: A Modern Approach

骆吉洲，男，1975 年生，博士，副教授。2006 年 5 月毕业于哈尔滨工业大学计算机科学与技术学院软件与理论专业，获工学博士学位。1999 年、2001 年在哈尔滨工业大学数学系基础数学专业分别获得理学学士学位和理学硕士学位。现就职于哈尔滨工业大学计算机科学与技术学院海量数据计算研究中心，讲授“算法设计与分析”“数学建模”“编译原理”等课程。出版教材《算法设计与分析》一部，出版译著《图论导引》一部。近年来一直从事生物信息学、压缩数据库技术、传感器网络、算法理论等领域的研究。主持和参加多项国家自然基金、863 计划、973 项目、国防预研等项目等多项；2001 年 9 月至 2003 年 5 月参加“计算机机群并行数据库系统”的研制，该项目获得了 2004 年度国家科学技术进步二等奖。近年来发表 30 余篇论文。

计算复杂性理论在过去三十多年中发展迅速。自 1990 年以来取得的出人意料的结果和基础性的结果本身就可以写出一部书。这些结果涉及的领域非常广泛，包括：经典复杂性类的概率型新定义（**IP**＝**PSPACE** 和各种 **PCP** 定理）以及它们在近似算法中的应用，肖尔（Shor）为量子计算机设计的整数因数分解算法，对人们目前处理著名的 **P** ≠ **NP** 问题[⊖]的各种方法为什么未能获得成功的理解，去随机化理论和基于计算难度的伪随机性，以及随机性提取器和扩张图等伪随机对象的优美构造。

本书的目标就是为了在介绍复杂性理论经典结果的同时阐述近年来取得的新成果。写作本书的出发点是让它既可以作为教科书使用，也可以作为自学的参考书使用。这意味着我们在写作本书时必须兼顾广泛的读者。为实现这一目标，我们对全书进行了精心的设计。我们实际上还假设读者不具备关于计算的任何背景而且只具备附录 A 中概述的最少数学背景。我们为本书提供了一个网站 http://www.cs.princeton.edu/theory/complexity。网站上列出了相关的辅助材料，包括用本书作为教材时的详细教学计划、全书各个章节的草稿，以及涵盖相关主题的其他资源的超链接。全书始终强调各个概念在何种场合下是有用的，以及为什么这些概念要这样定义。在一些关键的定义上，我们还用一些例子进行了阐释。为了使行文流畅，我们力争尽可能少地引用参考文献。参考文献的引用有两种情况，其一是当前的结果用到了文献中的标准术语，其二是我们觉得为特定的结果提供一些历史信息将有助于阐明其动机和适用的场合。每章末尾有一个单独的注记小节，它简明扼要地讨论了更多的相关工作。当一个概念有多种定义时，我们会选择相对简单的定义；当一个结果有多种证明时，我们会选择能证得更具一般性的结论或者最优结论的证明。

全书分为三个部分。

- *第一部分：基本复杂性类。*这个部分是对复杂性理论的广泛介绍。从图灵机的定义和可计算理论的基本概念开始，这个部分涵盖了各种基本的时间复杂性类和空间复杂性类，还包含了更现代的一些专题，包括概率算法、交互式证明、密码学、量子计算机和 **PCP** 定理及其应用。
- *第二部分：具体计算模型的下界。*这个部分讨论在线路和判定树等具体计算模型上用算法求解各种计算任务所需的计算资源的下界。这些计算模型初看起来与图灵机有很大的区别，但更深入研究将得到它们与图灵机之间的有趣的相互联系。
- *第三部分：高级专题。*这个部分主要是 1980 年以后人们在复杂性理论方面获得的进展。内容包括计数复杂性、平均复杂性、难度放大、去随机化和伪随机性、**PCP** 定理的证明以及自然证明。

本书的每一章几乎都可以单独进行阅读，但是第 1 章、第 2 章和第 7 章不能跳过。正是这种设计，使得本书可以适用于下面各种不同的读者。

- *物理学家、数学家和其他科学家。*这个读者群对计算复杂性理论越来越感兴趣，他们特别感兴趣的是那些高调的研究结果，例如肖尔算法（Shor algorithm）和最

⊖ 译文用“**P** ≠ **NP**”来表示原文中的“**P** versus **NP**”。——译者注

近取得的确定型素性测试算法。这个读者群的知识储备丰富，他们可以快速通读第一部分，然后迅速进入第二部分和第三部分，也可以单独阅读各个章节并找到理解当前研究结果所需的每个知识点。

- 本身不从事计算复杂性理论研究的计算机科学家。他们既可以用本书来自学，也可以将本书作为参考书，还可以用本书来讲授本科生或研究生的计算理论或计算复杂性理论课程。
- 从事计算复杂性理论研究或者打算从事这种研究的任何人，包括教授和学生。本书讲解最新研究结果和高级专题的详细程度可以让打算从事复杂性理论和相关领域研究的读者具有充足的知识储备。

本书可以作为如下几类课程的教科书。

- 本科生的计算理论课程。很多计算机科学系都用西普赛尔（Sipser）的书［Sip96］来为本科生开设计算复杂性理论这门课。本书可以用作对西普赛尔的教材在一些更现代的专题上的补充，这些专题包括概率算法、密码学和量子计算。相比于自动机理论和可计算理论的精细划分，本科生可能会发现这些专题更能令人耳目一新。所需的数学背景是能够比较自然地阅读数学证明以及离散数学知识，这些知识通常涵盖于“离散数学”或“计算机数学”等课程中，而目前多数计算机系都已经开设了这样的课程。
- 为高年级本科生和新入学的研究生开设的计算复杂性导论课程。本书还可以用来为计算机科学专业的高年级本科生和新入学的研究生开设计算复杂性导论课程，以替代 1994 年帕帕迪米特里奥（Papadimitriou）撰写的教材［Pap94］（该书未包含很多最近的研究成果）。这门课程可以讲授第一部分的多数专题，再零星地讲授第二部分和第三部分的内容，并且假设学生具备了一定的算法知识和计算理论的知识。
- 研究生的计算复杂性课程。本书也可以作为研究生的计算复杂性课程的教材，以培养学生在复杂性理论或者算法和机器学习等相关领域开展研究的能力。这门课程可以用第一部分来复习基本知识，然后进入第二部分和第三部分的高级专题中。本书的内容多于一个学期的教学内容，网站上提供了这门课程的其他几种教学大纲。
- 研究生讨论班或高级课程。第二部分和第三部分中的各个独立章节都可以用于复杂性理论的讨论班和高级课程，比如关于去随机化、**PCP** 定理和下界的讨论班或高级课程。

本书网站为这些课程提供了几种教学计划和素材。如果你在课程中采用了本书，我们乐意了解情况并得到你的反馈。我们要求你不要在网上发布本书习题的答案，这样其他人才可以用这些习题给学生留作业或出考题。

在写作本书的过程中，我们清醒地意识到我们不得不舍弃对一些重要结果的讲述。我们希望本书对其他教材的大量引用有助于读者的进一步阅读。同时，我们还计划对本书的网站进行周期性的更新，以帮助读者了解和浏览他们感兴趣的新结果。

最重要的是，我们希望通过本书将计算复杂性中激动人心的研究结果以及它们对其他学科的深刻影响传递给读者。

让我们一起为彻底解决 **P** 与 **NP** 问题而努力吧！

我们对复杂性理论的理解是在与同行交流的过程中逐步形成的。我们从他们身上学到了太多的东西，要感谢的人也远不止下面提到的这些人。Boaz 首先要感谢两位导师 Oded Goldreich 和 Avi Wigderson，是他们把他引入了理论计算机科学的世界并不断影响他在这一领域的思想。

我们感谢 Luca Trevisan，他（从 8 年前开始!）对本书的写作提供了持续不断的鼓励，并为第一版草稿中若干章节的撰写提供了大量帮助。一些同行毅然地同意并审阅了本书部分章节的早期草稿，他们是：Scott Aaronson，Noga Alon，Paul Beame，Irit Dinur，Venkatesan Guruswami，Jonathan Katz，Valentine Kavanets，Subhash Khot，Jiří Matoušek，Klaus Meer，Or Meir，Moni Naor，Alexandre Pinto，Alexander Razborov，Oded Regev，Omer Reingold，Rosen Shaltiel，Madhu Sudan，Amnon Ta-Shma，Iannis Tourlakis，Chris Umans，Salil Vadhan，Dieter van Melkebeek，Umesh Vazirani 和 Joachim von zur Gathen. 特别感谢 Jiří、Or、Alexandre、Dieter 和 Iannis，他们对本书的很多章节给出了具体而有用的评述。

很多人指出了本书的拼写错误和缺陷，他们给出的评述帮助我们改进了文字质量，还有一些人回答了我们在特定证明中提出的问题或者给出了相应的参考文献。在此，一并对他们表示感谢，他们是：Emre Akbas，Eric Allender，Djangir Babayev，Miroslav Balaz，Arnold Beckmann，Ido Ben-Eliezer，Siddharth Bhaskar，Goutam Biswas，Shreeshankar Bodas，Josh Bronson，Arkadev Chattopadhyay，Bernard Chazelle，Maurice Cochand，Nathan Collins，Tim Crabtree，Morten Dahl，Ronald de Wolf，Scott Diehl，Dave Dody，Alex Fabrikant，Michael Fairbank，Joan Feigenbaum，Lance Fortnow，Matthew Franklin，Rong Ge，Ali Ghodsi，Parikshit Gopanlan，Vipul Goyal，Stephen Harris，Johan Håstad，Andre Hernich，Yaron Hirsch，Thomas Holenstein，Xiu Huichao，Moukarram Kabbash，Bart Kastermans，Joe Kilian，Tomer Kotek，Michal Koucy，Sebastian Kuhnert，Katrina LaCurts，Chang-Wook Lee，James Lee，John Lenz，Meena Mahajan，Mohammad Mahmoody-Ghidary，Shohei Matsuura，Mauro Mazzieri，John McCullough，Eric Miles，Shira Mitchell，Mohsen Momeni，Kamesh Munagala，Rolf Neidermeier，Robert Nowotniak，Taktin Oey，Toni Pitassi，Emily Pitler，Aaron Potechin，Manoj Prabhakaran，Yuri Pritykin，Anup Rao，Saikiran Rapaka，Nicola Rebagliati，Johan Richter，Ron Rivest，Sushant Sachdeva，Mohammad Sadeq Dousti，Rahul Santhanam，Cem Say，Robert Schweiker，Thomas Schwentick，Joel Seiferas，Jonah Sherman，Amir Shpilka，Yael Snir，Nikhil Srivastava，Thomas Starbird，Jukka Suomela，Elad Tsur，Leslie Valiant，Vijay Vazirani，Suresh Venkatasubramanisn，Justin Vincent-Foglesong，Jirka Vomlel，Daniel Wichs，Avi Wigderson，Maier Willard，Roger Wolff，Jureg Wullschleger，Rui Xue，Jon Yard，Henry Yuen，Wu Zhanbin 和 Yi Zhang. 谢谢你们!

毫无疑问，上述列表仍有可能遗漏了这些年向我们的创作提供过帮助的人，我们对那

些被遗漏的人表示感谢和歉意。

本书用 LATEX 进行排版，为此特别感谢 Donald Knuth 和 Leslie Lamport。Stephen Boyd 和 Lieven Vanderberghe 爽快地向我们提供了他们写作《凸优化》一书时使用的 LATEX宏定义。

最重要的是，感谢我们的家人——Silvia、Nia、Rohan Arora 和 Ravit、Alma Barak。

Sanjeev 要感谢父亲，父亲是他创作本书的源泉。

如果一个科学分支还能找到大量的研究问题，则这个分支还活着；研究问题的缺失则预示着这个科学分支的消亡或者独立发展的终结。

——戴维·希尔伯特（David Hilbert），1900

我演讲的主题或许可以直接用两个简单的问题来揭示。第一个问题是，乘法比加法更难吗？而第二个问题则是，为什么？……我（想）要证明不存在乘法的算法在计算上同加法的算法一样简单，这就证明了乘法计算中确实存在某种绊脚石。

——阿兰·科巴姆（Alan Cobham），1964

几千年来，人们在账目管理和天文学中不断地进行着各种计算，因此计算的概念以这种形式一直存在着。例如，利用计算可以求解下面的任务。

- 给定两个整数，计算它们的乘积。
- 给定 n 个变量上 n 个方程构成的方程组，找出它的一个解（如果解存在的话）。
- 给定一个熟人列表和这些人中彼此不能和睦相处的一些人员对，找出你在宴会中打算邀请的最大熟人子集使得他们彼此都能够和睦相处。

在历史长河中，人们总结得出：在概念上，计算就是在给定的输入上用有限个步骤得到输出的过程。他们认为，“计算”就是“在演草纸上根据一定规则写出一些数字的人”。

20 世纪中叶，科学上的一项重要突破就是对“计算”的精确定义。根据这个定义，人们马上清楚了计算其实就存在于各种物理系统和数学系统中——图灵机（Turing machine），λ 演算（Lambda calculus），细胞自动机（cellular automata），指针机（pointer machine），弹跳桌球（bouncing billiards ball），康韦生命游戏（Conway's game of life），等等。出人意料的是，这些形式的计算都是等价的，也就是说，其中一个模型能够实现的所有计算也能在其他模型上完成（参见第 1 章）。在这种认识的基础上，人们马上发明了能够执行所有程序的硬件，这就是标准的通用电子计算机。在接下来的几十年中，计算机迅速被社会接纳，这使得计算融入了现代生活的方方面面，也使得计算问题在设计、计划、工程、科学发现等人类活动中变得越来越重要，于是计算机算法（亦即，求解计算问题的各种方法）变得无处不在。

然而，计算并不“仅仅”是一种用于实践的工具，它也是一个主要的科学概念。事实上，科学家现在将许多自然现象都视为一种计算过程，这些计算过程实际上是细胞自动机的推广。例如，对生物繁衍过程的理解曾经导致了自体复制计算机的发现。再比如，物理学家薛定谔（Schroedinger）曾在他的书［Sch44］中预言细胞中肯定存在类似于 DNA 的物质，后来沃森（Watson）和克里克（Crick）发现了这种物质，克里克坦承他们的研究正是受到了薛定谔的工作的启发。如今，各种计算模型已经成为生物学和神经科学中许多领域研究的基础。电子电动力学（QED）等几种物理理论的本征刻画非常类似于计算的定义，这种现象甚至还促使一些科学家相信整个宇宙就是一台超级计算机（参见 Lloyd［Llo06］）。更有趣的是，这样的物理理论在过去的十年中已经被用于设计量子计算机（参见第 10 章）。

可计算性与计算复杂性

研究者在成功地给出计算的概念之后，开始研究什么样的问题是可计算的。他们证明了几个有趣的问题本质上都不是可计算的，也就是说，没有计算机能够求解这些问题而不在任何输入上陷入无限循环（即不停机）。可计算性理论是一个优美的专题，但本书不会用太多篇幅讨论它。我们仅在第 1 章中简要地讨论可计算性理论，感兴趣的读者可以参阅标准的教科书［Sip96，HMU01，Koz97，Rog87］。事实上，本书的焦点是计算复杂性理论，它关注计算的效率，也就是量化地研究求解给定计算任务所需的计算资源的数量。下面将会对计算效率非形式化地进行量化。然后，再讨论复杂性研究中有关计算效率的几个问题。

计算效率的量化

我们用前面提到的三个计算任务来阐释计算效率的含义。首先，考虑两个整数的乘法。我们可以用两种不同的方法（或算法）来完成这项任务。第一种方法是累加算法；也就是说，为了计算 $a \cdot b$，只需用 $b-1$ 次加法将 b 个 a 进行累加。第二种方法是图 I-1 所示的小学列式算法。虽然累加算法比小学列式算法简单，但我们总觉得后者比前者更好。事实上，小学列式算法的效率更高。例如，计算 577 乘以 423 时，累加算法需要计算 422 次加法；但小学列式算法却只需将其中一个数分别与 3 个一位数相乘，再计算 3 次加法。

$$
\begin{array}{cccccc}
 & & & 5 & 7 & 7 \\
 & & & 4 & 2 & 3 \\
\hline
 & & 1 & 7 & 3 & 1 \\
 & 1 & 1 & 5 & 4 & \\
2 & 3 & 0 & 8 & & \\
\hline
2 & 4 & 4 & 0 & 7 & 1
\end{array}
$$

图 I-1　用 577 · 423 展示乘法的小学列式算法

算法效率的量化就是研究算法所执行的基本操作个数随着输入规模的增长是如何变化的。在整数乘法中，单个数位相加或相乘就是基本操作（在其他场合中，除法也可以是基本操作），而两个因数中的数位个数就是输入规模。在计算两个 n 位整数的乘法时（也就是说，两个因数都大于 10^{n-1} 且小于 10^n），小学列式算法执行的基本操作数不超过 $2n^2$，而累加算法执行的基本操作数至少是 $n10^{n-1}$。经过这样的分析，两种算法之间的巨大差异就显而易见了。即使计算两个 11 位整数的乘法，执行小学列式算法的便携式计算器也会快于执行累加算法的超级计算机。对于稍大一些的整数，小学五年级学生用笔和纸执行列式算法也会胜于执行累加算法的超级计算机。由此可见，算法的效率明显比运行算法所用的技术要重要得多。

更令人意外的是，借助傅里叶变换可以设计出更快的乘法算法。这个算法 40 年前才被人们发现，在这个算法中，两个 n 位整数的乘法仅用 $cn\log n\log\log n$ 个操作就可以完成，其中 c 是独立于 n 的绝对常数，参见第 16 章。我们将这样的算法称为 $O(n\log n\log\log n)$ 步的算法，参见后面的记号约定。随着 n 的增大，该算法执行的基本操作数将远小于 n^2。

对于线性方程组的求解，经典的高斯消元法用 $O(n^3)$ 个基本的算术操作就可以求解 n 个变量上 n 个方程构成的方程组（虽然这一算法用高斯的名字命名，但早在公元 1 世纪中国数学家就已经掌握了这种算法的某种形式）。20 世纪 60 年代末，斯特拉森（Strassen）找到了更加高效的算法，该算法大约只需要执行 $O(n^{2.81})$ 个操作就能求解这个问题。目

前，求解这一问题的最佳算法需要执行 $O(n^{2.376})$ 个操作，参见第 16 章。

宴会问题也有一段有趣的历史。同乘法的情况一样，宴会问题也存在显而易见的简单低效算法——从大到小地依次枚举 n 个人的每个子集，直到找到一个子集使得其中不含任何两个无法和睦相处的人。这个算法需要执行的计算步骤数可能与 n 个人的所有子集数一样多，亦即 2^n。这使得该算法根本无法用于实践，因为如果某人用这个算法来安排一个 70 人参加的宴会，即使她用超级计算机来进行处理，也需要提前一千年开始筹备。但出乎意料的是，人们至今仍没有为宴会问题找到效率显著更优的算法。事实上，正如第 2 章中我们将会看到的那样，我们有理由怀疑宴会问题不存在高效的算法，因为我们可以证明它等价于独立集这个计算问题，而独立集问题以及其他成千上万个计算问题都是 **NP**-完全问题。著名的 **P** ≠ **NP** 问题（参见第 2 章）是问：有没有哪个 **NP**-完全问题存在高效的算法？

证明高效算法的不存在性

我们已经看到，某些计算任务存在非平凡的算法使得其效率比几千年来人们一直使用的算法更高。一件特别有意义的事情是：证明某些组合任务的当前算法是最佳的。也就是说，这些计算任务不存在更有效的算法。例如，我们可以证明整数乘法的 $O(n\log n\log\log n)$ 步算法无法进一步改进，这就说明乘法在本质上确实比加法更难，因为加法存在 $O(n)$ 步算法。再比如，我们还可以证明，没有算法能用少于 $2^{n/10}$ 个计算步骤来求解宴会问题。证明这样的结论是计算复杂性的一个核心任务。

如何才能证明这种不存在性呢？求解计算任务的算法可能有无穷个！因此，我们只能用数学手段证明其中的每个算法都比已知的算法更低效。这种方法之所以可行，是因为计算本身也是一个精确的数学概念。事实上，一旦这样一个结果被证明，则它必然吻合于数学上的某个不可能性结果。例如，几何中其他公理无法推导出欧几里得平行公理、尺规作图无法三等分一个角等。这些结论都是数学上最有价值、最可靠和最出人意料的结果。

在复杂性理论中，我们很少能证明这种不存在性结果。但是，在能力弱于一般计算机的计算模型上，我们确实已经证得了一些重要的不存在性结果。本书第二部分将讨论这些结果。由于在一般的计算机上我们还缺少这样的好结果，因此复杂性理论在一般计算机上获得的重要结果指的是在不同复杂性问题之间建立的相互联系。人们在这方面获得了很多漂亮的结果，本书介绍了大量这样的结果。

关于计算效率的几个有趣问题

现在，我们概述关于计算复杂性的几个重要问题，本书后续章节将详细地讨论这些问题。附录 A 给出了相关的数学背景。

1. 生命科学、社会科学和运筹学等学科中的很多计算任务都通过搜索海量的解空间来找出问题的解。比如，前面已经提到的线性方程组的求解和宴会问题中找出最大的受邀者集合都属于这种情况。这种搜索通常称为穷举搜索，因为搜索过程穷举了所有的可能。这种穷举搜索能替换为更有效的算法吗？

正如我们将在第 2 章所见，这本质上就是著名的 **P** ≠ **NP** 问题，它是计算复杂性理论的核心问题。很多有趣的搜索问题都是 **NP**-完全问题，这意味着，如果 **P**≠**NP** 这个著名的猜想是正确的，则这样的搜索问题将不存在高效的算法；亦即，这种搜索问题本质上是难解的。

2. 用随机性（即硬币投掷）能加快算法的计算速度吗？

第 7 章介绍了随机计算，并为一些特定的计算任务给出了高效的概率型算法。但是，第 19 章和第 20 章给出了人们最近得到的一个出人意料的结果，该结果提供了很强的证据来表明随机性对提高计算效率而言作用非常有限，因为任意概率型算法都可以替换为一个效率相当的确定型算法（即不用投掷硬币的算法）。

3. 如果允许算法在小部分输入上出错，或者只要求算法求得问题的近似解，那么难解的问题会变得容易一些吗？

平均复杂性和近似算法的研究出现在本书的第 11 章、第 18 章、第 19 章和第 22 章。这几章还建立了上述问题、随机性的效能、数学证明的不同概念和纠错码理论之间的优美的相互联系。

4. 计算上的难解问题对实践有什么帮助呢？例如，我们能借助这些难解问题构造出牢不可破的密码协议吗（至少相对于大家认可的敌手而言）？

正如第 9 章所讲，安全的数字密码与 **P** ≠ **NP** 问题（参见第 2 章）和平均复杂性（参见第 18 章）密不可分。

5. 我们能利用物质的违背直觉的量子力学性质建造出更快的计算机吗？

第 10 章将介绍量子计算机这一备受瞩目的概念，它利用量子力学加速某些计算。彼得·肖尔（Peter Shor）已经证明，只要量子计算机被建造出来，则它能够高效地完成整数的因数分解（继而，现今的很多密码都将被攻破）。但是，要建造出量子计算机，目前还存在着很多令人生畏的障碍。

6. 只有人才能证明数学定理吗？换句话说，数学证明能被自动生成吗？能在不完整阅读数学证明的情况下验证数学证明的真伪吗？证明者和验证者通过对话来完成的交互式证明比标准的“静态”数学证明更有效力吗？

证明是数学上的核心概念。事实证明，它也是计算复杂性的核心概念。而且，计算复杂性已经对数学证明的含义赋予了新的解释。数学证明能否自动生成将取决于 **P** ≠ **NP** 问题的答案（参见第 2 章）。第 11 章研究概率可验证明（Probabilistic Checkable Proof）。概率可验证明是一种健壮的数学证明。要查验这种证明的真伪，只需概率地选取证明中的少数几个位置进行查验即可。相比之下，传统的证明则需要逐行阅读才能查验其真伪。类似地，第 8 章介绍了交互式证明的概念，并用它得出了一些出人意料的结果。最后，第 15 章研究了证明复杂性。它是复杂性的一个子领域，研究各种命题的最小证明长度。

历经近 40 年的发展，复杂性理论仍是一门年轻的科学，许多重要结果的发现还不到 20 年。上述这些问题还没有被完全解决。一个令人意外的转折是，复杂性理论被用于某些数学定理的证明中；它们提供的证据表明计算复杂性中某些问题是难解的，参见第 23 章。

我们引用希尔伯特 1900 年演讲中的另一些话作为结束[⊖]。

> （这种）不可能性的证明古人早已实现……［而］在后来的数学中，关于某些解的不可能性的问题发挥了重要作用……
>
> 在其他科学中，人们也经常遇到一些老问题，通过不可能性的证明，这些问

⊖ 引文的第二段和第三段与希尔伯特演讲中的顺序是颠倒的。——译者注

题被一种对科学来说是最满意、最有用的方式解决了……在构造永动机的努力失败以后，科学家在这种机器不可能存在的情况下研究了自然力之间必须遵循的关系；而这个反问题导致了能量守恒定律的发现……

也许正是这一值得注意的事实，加上其他的哲学因素，给人们以这样的信念……每个明确的数学问题都必然能被毋庸置疑地精确求解，或者是成功地对所给问题做出了回答，或者是证明了该问题解的不可能性，从而表明解答所给问题的一切努力都肯定要归于失败……这种信念……对于数学工作者是一种巨大的鼓舞。在我们中间，常常听到这样的呼唤：这里有一个数学问题，快找出它的答案！你能通过纯粹的数学推理找到答案，因为数学中不存在不可知的东西。

目 录

Computational Complexity: A Modern Approach

出版者的话
译者序
译者简介
前言
致谢
引言

第0章 记号约定 …… 1

0.1 对象的字符串表示 …… 1
0.2 判定问题/语言 …… 2
0.3 大O记号 …… 2
习题 …… 3

第一部分 基本复杂性类

第1章 计算模型——为什么模型选择无关紧要 …… 6

1.1 计算的建模：你真正需要了解的内容 …… 6
1.2 图灵机 …… 7
1.2.1 图灵机的表达能力 …… 10
1.3 效率和运行时间 …… 11
1.3.1 定义的健壮性 …… 11
1.4 机器的位串表示和通用图灵机 …… 14
1.4.1 通用图灵机 …… 14
1.5 不可计算性简介 …… 15
1.5.1 停机问题 …… 16
1.5.2 哥德尔定理 …… 17
1.6 类P …… 18
1.6.1 为什么模型选择无关紧要 …… 19
1.6.2 P的哲学意义 …… 19
1.6.3 P的争议和解决争议的一些努力 …… 20
1.6.4 埃德蒙兹的引言 …… 21
1.7 定理1.9的证明：$O(T\log T)$时间的通用模拟 …… 21
本章学习内容 …… 24
本章注记和历史 …… 24
习题 …… 26

第2章 NP和NP完全性 …… 29

2.1 类NP …… 29
2.1.1 P和NP的关系 …… 31
2.1.2 非确定型图灵机 …… 31
2.2 归约和NP完全性 …… 32
2.3 库克-勒维定理：计算的局部性 …… 34
2.3.1 布尔公式、合取范式和SAT问题 …… 34
2.3.2 库克-勒维定理 …… 34
2.3.3 准备工作：布尔公式的表达能力 …… 35
2.3.4 引理2.11的证明 …… 35
2.3.5 将SAT归约到3SAT …… 38
2.3.6 深入理解库克-勒维定理 …… 38
2.4 归约网络 …… 39
2.5 判定与搜索 …… 42
2.6 coNP、EXP和NEXP …… 43
2.6.1 coNP …… 43
2.6.2 EXP和NEXP …… 44
2.7 深入理解P、NP及其他复杂性类 …… 45
2.7.1 NP的哲学意义 …… 45
2.7.2 NP与数学证明 …… 45
2.7.3 如果P=NP会怎样 …… 45
2.7.4 如果NP=coNP会怎样 …… 46
2.7.5 NP和NP完全之间存在其他复杂性类吗 …… 47
2.7.6 NP难的处理 …… 47
2.7.7 更精细的时间复杂性 …… 48
本章学习内容 …… 48
本章注记和历史 …… 48

习题 …… 49

第 3 章 对角线方法 …… 53

3.1 时间分层定理 …… 53
3.2 非确定型时间分层定理 …… 54
3.3 拉德纳尔定理：NP 非完全问题的存在性 …… 55
3.4 神喻机器和对角线方法的局限性 …… 57
3.4.1 逻辑独立与相对 …… 59
本章学习内容 …… 59
本章注记和历史 …… 59
习题 …… 60

第 4 章 空间复杂性 …… 61

4.1 空间受限计算的定义 …… 61
4.1.1 格局图 …… 62
4.1.2 一些空间复杂性类 …… 63
4.1.3 空间分层定理 …… 64
4.2 PSPACE 完全性 …… 64
4.2.1 塞维奇定理 …… 67
4.2.2 PSPACE 的本质：最佳博弈策略 …… 67
4.3 NL 完全性 …… 68
4.3.1 基于证明的 NL 定义：仅能读一次的证明 …… 70
4.3.2 NL=coNL …… 71
本章学习内容 …… 72
本章注记和历史 …… 73
习题 …… 73

第 5 章 多项式分层和交错 …… 75

5.1 类 Σ_2^p …… 75
5.2 多项式分层 …… 76
5.2.1 多项式分层的性质 …… 76
5.2.2 PH 各层的完全问题 …… 77
5.3 交错图灵机 …… 78
5.3.1 无限次交错 …… 79
5.4 时间与交错：SAT 的时空平衡 …… 79
5.5 用神喻图灵机定义多项式分层 …… 80
本章学习内容 …… 81
本章注记和历史 …… 81
习题 …… 82

第 6 章 布尔线路 …… 83

6.1 布尔线路和 $\mathbf{P}_{/poly}$ …… 83
6.1.1 $\mathbf{P}_{/poly}$ 和 P 之间的关系 …… 85
6.1.2 线路的可满足性和库克-勒维定理的另一种证明 …… 86
6.2 一致线路 …… 87
6.2.1 对数空间一致线路族 …… 87
6.3 纳言图灵机 …… 88
6.4 $\mathbf{P}_{/poly}$ 和 NP …… 88
6.5 线路下界 …… 89
6.6 非一致分层定理 …… 90
6.7 线路复杂性类的精细分层 …… 91
6.7.1 类 NC 和类 AC …… 92
6.7.2 P 完全性 …… 92
6.8 指数规模的线路 …… 93
本章学习内容 …… 93
本章注记和历史 …… 94
习题 …… 94

第 7 章 随机计算 …… 96

7.1 概率型图灵机 …… 97
7.2 概率型图灵机示例 …… 98
7.2.1 寻找中位数 …… 99
7.2.2 概率型素性测试 …… 100
7.2.3 多项式恒等测试 …… 101
7.2.4 二分图的完美匹配测试 …… 102
7.3 单面错误和“零面”错误：RP、coRP、ZPP …… 103
7.4 定义的健壮性 …… 103
7.4.1 准确度常数的作用：错率归约 …… 104
7.4.2 期望运行时间与最坏运行时间 …… 105
7.4.3 使用比均匀硬币投掷更具一般性的随机选择 …… 106

7.5 BPP 同其他复杂性类之间的关系 …… 106
7.5.1 BPP⊆$P_{/poly}$ …… 107
7.5.2 BPP⊆PH …… 107
7.5.3 分层定理与完全问题 …… 108
7.6 随机归约 …… 109
7.7 空间受限的随机计算 …… 109
本章学习内容 …… 110
本章注记和历史 …… 110
习题 …… 111

第 8 章 交互式证明 …… 113

8.1 交互式证明及其变形 …… 113
8.1.1 准备工作：验证者和证明者均为确定型的交互式证明 …… 113
8.1.2 类 IP：概率型验证者 …… 115
8.1.3 图不同构的交互式证明 …… 116
8.2 公用随机源和类 AM …… 118
8.2.1 私有随机源的模拟 …… 119
8.2.2 集合下界协议 …… 120
8.2.3 定理 8.12 的证明概要 …… 123
8.2.4 GI 能是 NP-完全的吗 …… 123
8.3 IP=PSPACE …… 124
8.3.1 算术化 …… 125
8.3.2 #SAT_0的交互式协议 …… 125
8.3.3 TQBF 的协议：定理 8.19 的证明 …… 127
8.4 证明者的能力 …… 128
8.5 多证明者交互式证明 …… 129
8.6 程序检验 …… 130
8.6.1 具有验证程序的语言 …… 131
8.6.2 随机自归约与积和式 …… 131
8.7 积和式的交互式证明 …… 132
8.7.1 协议 …… 133
本章学习内容 …… 134
本章注记和历史 …… 134
习题 …… 135

第 9 章 密码学 …… 137

9.1 完全保密及其局限性 …… 138
9.2 计算安全、单向函数和伪随机数产生器 …… 139
9.2.1 单向函数：定义和实例 …… 141
9.2.2 用单向函数实现加密 …… 142
9.2.3 伪随机数产生器 …… 143
9.3 用单向置换构造伪随机数产生器 …… 144
9.3.1 不可预测性蕴含伪随机性 …… 144
9.3.2 引理 9.10 的证明：戈德赖希-勒维定理 …… 145
9.4 零知识 …… 149
9.5 应用 …… 151
9.5.1 伪随机函数及其应用 …… 151
9.5.2 去随机化 …… 153
9.5.3 电话投币和比特承诺 …… 154
9.5.4 安全的多方计算 …… 154
9.5.5 机器学习的下界 …… 155
本章学习内容 …… 155
本章注记和历史 …… 155
习题 …… 158

第 10 章 量子计算 …… 161

10.1 量子坐相：双缝实验 …… 162
10.2 量子叠加和量子位 …… 163
10.2.1 EPR 悖论 …… 165
10.3 量子计算的定义和 BQP …… 168
10.3.1 线性代数预备知识 …… 168
10.3.2 量子寄存器及其状态向量 …… 168
10.3.3 量子操作 …… 169
10.3.4 量子操作实例 …… 169
10.3.5 量子计算与 BQP …… 171
10.3.6 量子线路 …… 172
10.3.7 传统计算是量子计算的特例 …… 173
10.3.8 通用操作 …… 173
10.4 格罗弗搜索算法 …… 174
10.5 西蒙算法 …… 177
10.5.1 定理 10.14 的证明 …… 177
10.6 肖尔算法：用量子计算机实现整数分解 …… 178
10.6.1 Z_M上的傅里叶变换 …… 179

10.6.2 Z_M上的量子傅里叶变换 …… 180
10.6.3 肖尔的阶发现算法 …… 181
10.6.4 因数分解归约为阶发现 …… 184
10.6.5 实数的有理数近似 …… 185
10.7 BQP和经典复杂性类 …… 186
10.7.1 量子计算中类似于NP和AM的复杂性类 …… 187
本章学习内容 …… 187
本章注记和历史 …… 188
习题 …… 190

第11章 PCP定理和近似难度简介 …… 192

11.1 动机：近似求解NP难的优化问题 …… 193
11.2 用两种观点理解PCP定理 …… 194
11.2.1 PCP定理与局部可验证明 …… 194
11.2.2 PCP定理与近似难度 …… 197
11.3 两种观点的等价性 …… 197
11.3.1 定理11.5与定理11.9的等价性 …… 198
11.3.2 重新审视PCP的两种理解 …… 199
11.4 顶点覆盖问题和独立集问题的近似难度 …… 200
11.5 NP⊆PCP(poly(n), 1)：由沃尔什-哈达玛编码得到的PCP … 202
11.5.1 线性测试与沃尔什-哈达玛编码 …… 202
11.5.2 定理11.19的证明 …… 203
本章学习内容 …… 206
本章注记和历史 …… 206
习题 …… 207

第二部分 具体计算模型的下界

第12章 判定树 …… 210

12.1 判定树和判定树复杂性 …… 210
12.2 证明复杂性 …… 212
12.3 随机判定树 …… 213
12.4 证明判定树下界的一些技术 …… 214
12.4.1 随机复杂性的下界 …… 214
12.4.2 敏感性 …… 215
12.4.3 次数方法 …… 216
本章学习内容 …… 217
本章注记和历史 …… 217
习题 …… 218

第13章 通信复杂性 …… 219

13.1 双方通信复杂性的定义 …… 219
13.2 下界方法 …… 220
13.2.1 诈集方法 …… 220
13.2.2 铺砌方法 …… 221
13.2.3 秩方法 …… 222
13.2.4 差异方法 …… 223
13.2.5 证明差异上界的一种技术 …… 223
13.2.6 各种下界方法的比较 …… 224
13.3 多方通信复杂性 …… 225
13.4 其他通信复杂性模型概述 …… 227
本章学习内容 …… 228
本章注记和历史 …… 228
习题 …… 229

第14章 线路下界：复杂性理论的滑铁卢 …… 232

14.1 AC^0和哈斯塔德开关引理 …… 232
14.1.1 哈斯塔德开关引理 …… 233
14.1.2 开关引理的证明 …… 234
14.2 带“计数器”的线路：ACC … 236
14.3 单调线路的下界 …… 239
14.3.1 定理14.7的证明 …… 239
14.4 线路复杂性的前沿 …… 242
14.4.1 用对角线方法证明线路下界 … 242
14.4.2 ACC Vs P的研究现状 …… 243
14.4.3 具有对数深度的线性线路 …… 244
14.4.4 线路图 …… 244
14.5 通信复杂性方法 …… 245
14.5.1 与ACC^0线路之间的联系 …… 245
14.5.2 与线性规模对数深度的线路之间的联系 …… 246
14.5.3 与线路图之间的联系 …… 246

14.5.4 卡奇梅尔-维格德尔森通信游戏与深度下界 …… 246
本章学习内容 …… 248
本章注记和历史 …… 249
习题 …… 249

第 15 章 证明复杂性 …… 251

15.1 几个例子 …… 251
15.2 命题演算与归结 …… 252
15.2.1 用瓶颈法证明下界 …… 253
15.2.2 插值定理和归结的指数下界 … 254
15.3 其他证明系统概述 …… 256
15.4 元数学的思考 …… 258
本章学习内容 …… 258
本章注记和历史 …… 258
习题 …… 259

第 16 章 代数计算模型 …… 260

16.1 代数直线程序和代数线路 …… 261
16.1.1 代数直线程序 …… 261
16.1.2 例子 …… 262
16.1.3 代数线路 …… 263
16.1.4 代数线路中类似于 P、NP 的复杂性类 …… 264
16.2 代数计算树 …… 266
16.2.1 下界的拓扑方法 …… 268
16.3 布卢姆-舒市-斯梅尔模型 …… 270
16.3.1 复数上的复杂性类 …… 271
16.3.2 完全问题和希尔伯特零点定理 …… 271
16.3.3 判定性问题——曼德勃罗集 … 272
本章学习内容 …… 272
本章注记和历史 …… 273
习题 …… 274

第三部分 高级专题

第 17 章 计数复杂性 …… 278

17.1 计数问题举例 …… 278
17.1.1 计数问题与概率估计 …… 279
17.1.2 计数可能难于判定 …… 279
17.2 复杂性类 #P …… 280
17.2.1 复杂性类 PP：类似于 #P 的判定问题 …… 281
17.3 #P 完全性 …… 281
17.3.1 积和式和瓦利安特定理 …… 282
17.3.2 #P 问题的近似解 …… 286
17.4 卢田定理：$PH \subseteq P^{\#SAT}$ …… 287
17.4.1 过渡：具有唯一解的布尔满足性问题 …… 288
17.4.2 ⊕的性质和对 NP、coNP 证明引理 17.17 …… 289
17.4.3 引理 17.17 的证明：一般情形 …… 290
17.4.4 第二步：转换为确定型归约 … 291
17.5 待决问题 …… 292
本章学习内容 …… 293
本章注记和历史 …… 293
习题 …… 293

第 18 章 平均复杂性：勒维定理 …… 295

18.1 分布问题与 distP …… 296
18.2 “实际分布”的形式化定义 …… 298
18.3 distNP 及其完全问题 …… 298
18.3.1 distNP 的一个完全问题 …… 300
18.3.2 P-可抽样的分布 …… 301
18.4 哲学意义和实践意义 …… 301
本章学习内容 …… 303
本章注记和历史 …… 303
习题 …… 303

第 19 章 难度放大和纠错码 …… 305

19.1 从温和难度到强难度：姚期智 XOR 引理 …… 306
19.1.1 用因帕利亚佐难度核引理证明姚期智 XOR 引理 …… 307
19.1.2 因帕利亚佐难度核引理的证明 …… 309
19.2 工具：纠错码 …… 310

19.2.1 显式纠错码 …… 312
19.2.2 沃尔什-哈达玛纠错码 …… 312
19.2.3 里德-所罗门纠错码 …… 313
19.2.4 里德-穆勒纠错码 …… 313
19.2.5 拼接纠错码 …… 314
19.3 高效解码 …… 315
19.3.1 里德-所罗门解码 …… 315
19.3.2 拼接解码 …… 316
19.4 局部解码与难度放大 …… 316
19.4.1 沃尔什-哈达玛纠错码的局部解码算法 …… 318
19.4.2 里德-穆勒纠错码的局部解码算法 …… 318
19.4.3 拼接纠错码的局部解码算法 … 319
19.4.4 局部解码算法综合运用于难度放大 …… 320
19.5 列表解码 …… 321
19.5.1 里德-所罗门纠错码的列表解码 …… 322
19.6 局部列表解码：接近 BPP=P …… 323
19.6.1 沃尔什-哈达玛纠错码的局部列表解码 …… 323
19.6.2 里德-穆勒纠错码的局部列表解码 …… 323
19.6.3 拼接纠错码的局部列表解码 … 325
19.6.4 局部列表解码算法综合运用于难度放大 …… 325
本章学习内容 …… 326
本章注记和历史 …… 327
习题 …… 328

第 20 章 去随机化 …… 330

20.1 伪随机数产生器和去随机化 … 331
20.1.1 用伪随机数产生器实现去随机化 …… 331
20.1.2 难度与去随机化 …… 333
20.2 定理 20.6 的证明：尼散-维格德尔森构造 …… 334
20.2.1 两个示意性例子 …… 334
20.2.2 尼散-维格德尔森构造 …… 336
20.3 一致假设下的去随机化 …… 339
20.4 去随机化需要线路下界 …… 340
本章学习内容 …… 343
本章注记和历史 …… 343
习题 …… 344

第 21 章 伪随机构造：扩张图和提取器 …… 345

21.1 随机游走和特征值 …… 346
21.1.1 分布向量和参数 $\lambda(G)$ …… 346
21.1.2 无向连通性问题的随机算法的分析 …… 349
21.2 扩张图 …… 349
21.2.1 代数定义 …… 350
21.2.2 组合扩张和扩张图的存在性 … 350
21.2.3 代数扩张图蕴含组合扩张图 … 351
21.2.4 组合扩张图蕴含代数扩张图 … 352
21.2.5 用扩张图设计纠错码 …… 353
21.3 扩张图的显式构造 …… 355
21.3.1 旋转映射 …… 356
21.3.2 矩阵乘积和路径乘积 …… 356
21.3.3 张量积 …… 356
21.3.4 替换乘积 …… 357
21.3.5 显式构造 …… 359
21.4 无向连通性问题的确定型对数空间算法 …… 361
21.4.1 连通性问题的对数空间算法（定理 21.21 的证明） …… 361
21.5 弱随机源和提取器 …… 362
21.5.1 最小熵 …… 363
21.5.2 统计距离 …… 364
21.5.3 随机性提取器的定义 …… 364
21.5.4 提取器的存在性证明 …… 364
21.5.5 基于哈希函数构造提取器 …… 365
21.5.6 基于扩张图的随机游走构造提取器 …… 366
21.5.7 由伪随机数产生器构造提取器 …… 366
21.6 空间受限计算的伪随机数产生器 …… 368

本章学习内容 …… 372
本章注记和历史 …… 372
习题 …… 374

第 22 章 PCP 定理的证明和傅里叶变换技术 …… 378

22.1 非二进制字母表上的约束满足问题 …… 378
22.2 PCP 定理的证明 …… 379
22.2.1 PCP 定理的证明思路 …… 379
22.2.2 迪纳尔鸿沟放大：引理 22.5 的证明 …… 380
22.2.3 扩张图、随机游走和 INDSET 的近似难度 …… 381
22.2.4 迪纳尔鸿沟放大 …… 382
22.2.5 字母表削减：引理 22.6 的证明 …… 387
22.3 $2CSP_W$ 的难度：鸿沟和字母表大小之间的平衡 …… 389
22.3.1 莱斯的证明思想：并行重复 … 389
22.4 哈斯塔德 3 位 PCP 定理和 MAX-3SAT 的难度 …… 390
22.4.1 MAX-3SAT 的近似难度 …… 390
22.5 工具：傅里叶变换 …… 391
22.5.1 GF (2)^n 上的傅里叶变换 …… 391
22.5.2 从较高层面看傅里叶变换和 PCP 之间的联系 …… 393
22.5.3 GF (2) 上线性测试的分析 …… 393
22.6 坐标函数、长编码及其测试 …… 395
22.7 定理 22.16 的证明 …… 396
22.8 SET-COVER 的近似难度 …… 400
22.9 其他 PCP 定理概述 …… 402
22.9.1 具有亚常数可靠性参数的 PCP 定理 …… 402
22.9.2 平摊的查验复杂度 …… 402
22.9.3 2 位测试和高效傅里叶分析 …… 403
22.9.4 唯一性游戏和阈值结果 …… 404
22.9.5 与等周问题和度量空间嵌入之间的联系 …… 404
22.A 将 qCSP 实例转换成“精细”实例 …… 405
本章学习内容 …… 406
本章注记和历史 …… 407
习题 …… 408

第 23 章 为什么线路下界如此困难 …… 411

23.1 自然证明的定义 …… 411
23.2 为什么自然证明是自然的 …… 412
23.2.1 为什么要求可构造性 …… 413
23.2.2 为什么要求广泛性 …… 413
23.2.3 用复杂性测度看自然证明 …… 414
23.3 定理 23.1 的证明 …… 415
23.4 一个“不自然的”下界 …… 416
23.5 哲学观点 …… 417
本章注记和历史 …… 417
习题 …… 418

附录 A 数学基础 …… 419

部分习题的提示 …… 438

参考文献 …… 447

术语索引 …… 472

复杂性类索引 …… 478

记号约定

我们先给出全书使用的记号和一些约定。我们使用离散数学中的一些概念，如字符串、集合、函数、元组和图。所有这些概念均在附录 A 中进行了概述。

标准记号

我们用 $\mathbf{Z}=\{0, \pm 1, \pm 2, \cdots\}$ 表示整数集，用 $\mathbf{N}$ 表示自然数集(即非负整数集)。由字母 i，j，k，l，m，n 表示的数总是整数。如果 $n \geqslant 1$，则 $[n]$ 表示集合 $\{1, 2, \cdots, n\}$。对于实数 x，$\lceil x \rceil$ 表示满足 $n \geqslant x$ 的最小整数 $n \in \mathbf{Z}$，而 $\lfloor x \rfloor$ 表示满足 $n \leqslant x$ 的最大整数 $n \in \mathbf{Z}$。在需要使用整数的任何地方均隐式地使用操作符 $\lceil \rceil$。$\log x$ 表示 x 的以 2 为底的对数。如果存在某个数 N 使得条件 $P(n)$ 对任意满足 $n > N$ 的数均成立，则称条件 $P(n)$ 对充分大的 n 成立(例如，$2^n > 100n^2$ 对充分大的 n 成立)。在上下文明确了 i 的取值范围的情况下，我们使用形如 $\sum_i f(i)$ 的表达式；否则使用形如 $\sum_{i=1}^{n} f(i)$ 的表达式。如果 u 是字符串或向量，则 u_i 表示 u 的第 i 个符号或第 i 个坐标的值。

字符串

如果 S 是有限集合，则字母表 S 上的字符串是由 S 中的元素构成的长度有限的有序元组。在本书中，最典型的字符串是二元集合 $\{0, 1\}$ 上的字符串。对于任意整数 $n \geqslant 0$，我们用 S^n 表示 S 上长度为 n 的所有字符串构成的集合(S^0 表示由空元组构成的单元素集合)。S^* 表示所有字符串的集合(即 $S^* = \bigcup_{n \geqslant 0} S^n$)。如果 x，y 均是字符串，则用 $x \circ y$ 或直接用 xy 表示 x 和 y 的拼接，亦即先顺序列出 x 的元素再顺序列出 y 的元素。如果 x 是字符串且 $k \geqslant 1$ 是自然数，则 x^k 表示 k 个 x 的拼接。例如，1^k 表示由 k 个 1 构成的字符串。字符串 x 的长度表示为 $|x|$。

其他记号

如果 S 是一个分布，则用 $x \in_R S$ 表示服从分布 S 的一个随机变量；如果 S 是一个集合，则该记号表示 x 均匀分布于 S 的所有成员中。我们用 U_n 表示 $\{0, 1\}^n$ 上的均匀分布。对于两个长度为 n 的字符串 x，$y \in \{0, 1\}^n$，用 $x \odot y$ 表示它们的点积对 2 取模，亦即 $x \odot y = \sum_i x_i y_i \pmod 2$。而 n 维实数向量或复数向量 $\boldsymbol{u}$，$\boldsymbol{v}$ 的点积则用 $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ 表示(参见 A.5.1 节)。对于任意对象 x，我们用 $\underline{x}$(切勿与向下取整运算符 $\lfloor x \rfloor$ 混淆)表示 x 的字符串表示形式(参见 0.1 节)。

0.1 对象的字符串表示

本书讨论的基本计算任务是对函数的计算。但是，我们在多数时候仅讨论输入和输出均是长度为有穷值的位串(即 $\{0, 1\}^*$ 中的成员)的函数。

表示

仅考虑位串上的操作并未真正限制讨论的范围，因为只需通过编码就可以将整数、整

数对、图、向量、矩阵等一般对象表示为位串。例如，整数可以表示为二进制形式(比如，34可以表示为100010)，图可以表示为邻接矩阵(亦即，一个n顶点图G可以表示为一个$n\times n$的0/1矩阵A，其中$A_{ij}=1$当且仅当边$\overline{ij}$出现在G中)。我们将尽力避免像这样明确地处理底层的表示过程，而是直接用$\langle x\rangle$表示对象x在(未明确指定的)某种规范方法下的二进制表示。甚至，我们常省略符号$\langle\ \rangle$，直接用x表示对象x及其位串表示。

序对和元组的表示

我们用记号$\langle x, y\rangle$表示由x和y构成的有序对。由x和y的位串表示可以非常容易地得到$\langle x, y\rangle$的规范位串表示。例如，可以首先将$\langle x, y\rangle$编码为$\{0, 1, \#\}$上的字符串$\langle x\rangle\#\langle y\rangle$，然后通过映射$0\mapsto 00$，$1\mapsto 11$，$\#\mapsto 01$将$\langle x\rangle\#\langle y\rangle$表示为位串。为了避免杂乱，我们不使用记号$\langle\langle x, y\rangle\rangle$，而是直接用$\langle x, y\rangle$表示序对$\langle x, y\rangle$及其位串表示。类似地，用$\langle x, y, z\rangle$表示由$x$，$y$，$z$构成的有序三元组及其位串表示。对于4元组和5元组等，我们采用类似的记号。

计算输入和输出不是位串的函数

位串表示方法使我们可以讨论如何计算输入和输出不是位串的函数(如以自然数为输入的函数)。此时，我们隐式地将定义域和值域不是位串的函数f等同于函数$g:\{0, 1\}^*\to\{0, 1\}^*$，它以对象$x$的位串表示为输入，以$f(x)$的位串表示为输出。并且，利用序对和元组的位串表示，我们还可以讨论输入或输出是多个元素的计算函数。

0.2　判定问题/语言

在所有将位串映射为位串的函数中，布尔函数这种特例非常重要，该函数的输出只有一个位。我们将一个具体的布尔函数f表示为$\{0, 1\}^*$的子集$L_f=\{x:\ f(x)=1\}$，并将该集合称为语言或判定问题(这两个术语交替使用)[⊖]。这样，函数f的计算问题(即给定x计算$f(x)$)就等同于语言L_f的判定问题(即给定x判定$x\in L_f$是否成立)。

例0.1　将受邀参加晚宴的人表示为图的顶点，不能和睦相处的任意两人之间有一条边。引言中给出的晚宴计算问题就变成了在给定的图中找出大小最大的独立集(即没有边相连的一些顶点)。该问题的语言是

$$\mathrm{INDSET}=\{\langle G,k\rangle:\exists S\subseteq V(G)\text{ s.t. }|S|\geqslant k\text{ 且 }\forall u,v\in S,\overline{uv}\notin E(G)\}$$

求解该语言的算法将以一个图G和一个整数k为输入，看输出是否存在至少有k个无冲突受邀者的集合(即独立集)。目前，你可能还不清楚这种算法是否可以用来计算独立集，但我们将在第2章看到，这种算法确实可以用于计算独立集。现在，我们只需坚信上述语言是该问题的有益的形式化表示。◀

0.3　大O记号

算法的计算效率一般通过将该算法执行的基本操作的个数表达为算法输入的长度的函数来表示。这就是说，算法的效率用从自然数集$\mathbf{N}$到其自身的函数T来刻画，$T(n)$是算法在所有长度为n的输入上执行的基本操作的最大个数。然而，函数T的形式有时严重地

⊖　或许，选用"语言"这个词来表示$\{0, 1\}^*$的子集并不太合适。然而，由于历史原因，它却是目前采用的标准术语。

依赖于基本操作的具体定义。例如，在整数的加法中，基本操作既可以按照十进制(以 10 为基数)也可以按照二进制(以 2 为基数)来定义，但后者执行的基本操作个数是前者执行的基本操作个数的 3 倍还多。为了避免在基本操作的定义上纠缠不清而仅关注算法的宏观行为，采用下面的记号十分有益。

定义 0.2（大 O 记号）　设 f，g 是从 $\mathbf{N}$ 到 $\mathbf{N}$ 的两个函数。(1)如果存在常数 c 使得 $f(n)\leqslant c\cdot g(n)$ 对充分大的 n 恒成立，则称 $f=O(g)$；(2)如果 $g=O(f)$，则也称 $f=\Omega(g)$；(3)如果 $f=O(g)$ 且 $g=O(f)$，则称 $f=\Theta(g)$；(4)如果对任意 $\varepsilon>0$ 均有 $f(n)\leqslant\varepsilon\cdot g(n)$ 对充分大的 n 恒成立，则称 $f=o(g)$；(5)如果 $g=o(f)$，则也称 $f=\omega(g)$。

当需要强调函数的输入参数时，常用 $f(n)=O(g(n))$ 代替 $f=O(g)$。类似的记号也适用于记号 o，Ω，ω，Θ。

例 0.3　下面给出大 O 记号的几个用例。

1. 如果 $f(n)=100n\log n$ 而 $g(n)=n^2$，则有关系 $f=O(g)$，$g=\Omega(f)$，$f=o(g)$，$g=\omega(f)$。

2. 如果 $f(n)=100n^2+24n+2\log n$ 而 $g(n)=n^2$，则 $f=O(g)$。我们通常将这种关系记为 $f(n)=O(n^2)$。注意，我们还有关系 $g=O(f)$，因而有 $f=\Theta(g)$ 和 $g=\Theta(f)$。

3. 如果 $f(n)=\min\{n,10^8\}$ 而 $g(n)=1$ 对任意 n 成立，则 $f=O(g)$。我们通常将这种关系记为 $f=O(1)$。类似地，如果函数 h 的值随 n 增大而趋于无穷大(对于任意 c，$h(n)>c$ 对充分大的 n 恒成立)，则记为 $h=\omega(1)$。

4. 设 $f(n)=2^n$ 而 $g(n)=n^c$，其中 $c\in\mathbf{N}$ 是任意的自然数，则 $g=o(f)$。通常，我们将这种关系记为 $2^n=n^{-\omega(1)}$。类似地，我们用 $h(n)=n^{O(1)}$ 表示 h 以多项式为上界，亦即存在常数 $c>0$ 使得 $h(n)\leqslant n^c$ 对充分大的 n 恒成立。对于这种情况，我们还常用 $h(n)=\mathrm{poly}(n)$ 来表示。◀

对于更多的例子和相关论述，请参阅本科生算法教材[DPV06，KT06，CLR01]或西普赛尔(Sipser)的书[Sip96]的第 7.1 节。

习题

0.1　对下列的每对函数 f，g，确定 $f=o(g)$，$g=o(f)$ 和 $f=\Theta(g)$ 中哪个关系成立。如果 $f=o(g)$，请找出满足 $f(n)<g(n)$ 的最小 n 值。

(a) $f(n)=n^2$，$g(n)=2n^2+100\sqrt{n}$

(b) $f(n)=n^{100}$，$g(n)=2^{n/100}$

(c) $f(n)=n^{100}$，$g(n)=2^{n^{1/100}}$

(d) $f(n)=\sqrt{n}$，$g(n)=2^{\sqrt{\log n}}$

(e) $f(n)=n^{100}$，$g(n)=2^{(\log n)^2}$

(f) $f(n)=1000n$，$g(n)=n\log n$

0.2　对下面每个递归函数 f，找出一个(非递归)的封闭表达式 g 使得 $f(n)=\Theta(g(n))$，并证明之。(注记：下面仅给出了递归式，你可以假定 $f(1)=f(2)=\cdots=f(10)=1$ 而将递归式作用于 $n>10$ 的自变量。在各个小题中，无论初始值等于多少均不会影响答案的形式。你能说明为什么吗?)

(a) $f(n)=f(n-1)+10$

(b) $f(n)=f(n-1)+n$

4

(c) $f(n)=2f(n-1)$

(d) $f(n)=f(n/2)+10$

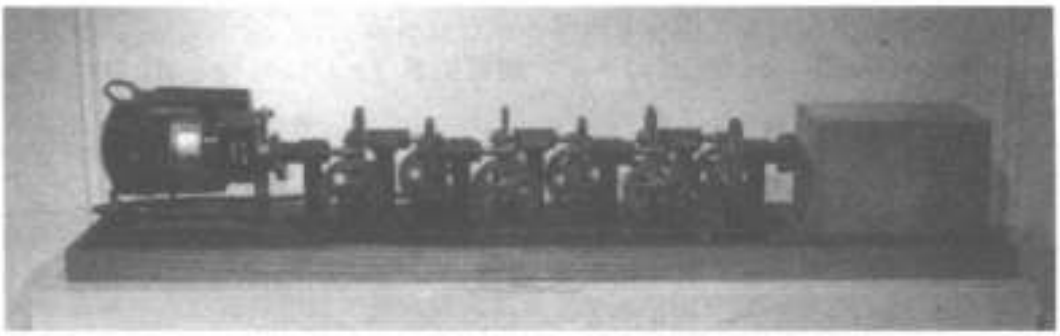
(e) $f(n)=f(n/2)+n$

(f) $f(n)=2f(n/2)+n$

(g) $f(n)=3f(n/2)$

(h) $f(n)=2f(n/2)+\Theta(n^2)$

0.3　麻省理工学院的博物馆中收藏了一件由亚瑟·冈松(Arthur Ganson)制作的一架称为混凝土机器的传动结构。该机器由13个齿轮连接而成，每个齿轮比前一个齿轮慢50倍。最快的齿轮由引擎匀速驱动，每分钟转212圈。最慢的齿轮固定在一块混凝土上，因此它根本不能转动。请解释为什么该机器不会破碎。



5

图0-1　由亚瑟·冈松制作的混凝土机器，本照片经艺术家授权使用

第一部分

Computational Complexity, A Modern Approach

基本复杂性类

第 1 章

Computational Complexity: A Modern Approach

计算模型——为什么模型选择无关紧要

数字计算机背后的思想可以阐述为：这种机器旨在模拟执行人所能施行的所有计算操作。假定人只能按指定的规则进行计算，且不能丝毫偏离计算规则。可以这样认为，计算手册提供计算规则，但只有在执行新的计算任务时才更换计算手册。人在演草纸上完成计算，并且演草纸是用之不竭的。

——阿兰·图灵

[图灵]首次成功地给出了一个认识论概念的有意义的不依赖于其所用形式的完整定义。

——库尔特·哥德尔

初看起来，为计算建立数学模型可谓难上加难。这是由于，历史上人类在解决各种计算任务的过程中用尽了各种各样的方法——从直觉和灵感到算盘或计算尺，再到现代的计算机。此外，自然界中其他生物或系统也时刻需要处理各种计算任务，而它们的解决之道也是纷乱繁杂。怎样才能找出一个能抓住这些计算方法共性的简洁的数学模型呢？如果再考虑到本书要关注的计算效率问题，则建模问题就更加无从下手了。考虑计算效率问题似乎必须小心地选择计算模型，因为即便是孩童也知道一款新的视频游戏是否能"高效运行"将依赖于他的计算机硬件。

令人惊讶的是，一个简洁的计算模型——图灵机足以研究关于计算及其效率的所有问题。将讨论范围仅限于这种计算模型的理由是充分的，因为它能够模拟所有能被物理实现的计算方法而仅在计算效率上略有损失。因此，图灵机"能高效解决"的计算任务至少与其他计算方法能高效解决的计算任务一样多(一个可能的例外是第 10 章讨论的量子计算机模型，但目前还不清楚它能否被物理实现)。

本章将给出图灵机的形式定义，并研究它的一些基本性质。1.1 节概述了图灵机模型及其基本性质。该小节还为普通读者概述了 1.2 节至 1.5 节的结论，以帮助这些读者跳过图灵机模型杂乱的细节而直接从 1.6 节开始进入复杂性理论。

由于复杂性理论[⊖]关注的是计算效率，因此 1.6 节给出了本书最重要的几个定义之一，亦即复杂性类 **P** 的定义，它从数学上刻画了由所有能被高效求解的判定问题构成的集合。1.6 节还就下面的问题展开了一些讨论：类 **P** 是否真的刻画了"高效计算"这一非正式概念的本质。该小节还表明了图灵机的定义是如何贯穿全书的；同时还指出，类 **P** 是定义很多其他模型的出发点，这些模型包括非确定型图灵机、概率型图灵机、量子图灵机、布尔线路、并行计算机、判定树和通信游戏，其中有些模型用于研究物理计算的有争议的实现模式，而另一些则主要用于深入理解图灵机计算的本质。

1.1 计算的建模：你真正需要了解的内容

形式地讨论图灵机将不可避免地遇到一些单调乏味的概念。我们先概述这些直觉概念，

⊖ "complexity"一词译做"复杂性"或"复杂度"。当比较抽象地论及"complexity"时译作"复杂性"，当用具体函数论及"complexity"的高低时，译作"复杂度"。

以便普通读者可以跳过正式的讨论而直接进入从 1.6 节开始的复杂性理论。当他们在阅读后续章节的过程中偶尔遇到确实需要图灵机模型的细节时，可以随时回头阅读跳过的小节。

千百年来，“计算”这个词曾一直被理解为将机械的规则作用于操作数上，其中完成操作的人或机器允许使用演草纸来记录中间结果。图灵机是这种直觉概念的具体实现。1.2.1 节表明，图灵机等价于任何一种现代程序设计语言——除了对内存大小没有限制之外[⊖]。

下面，我们按照本章开头所引用的图灵的话非正式地描述图灵机模型。令 f 是一个以位串（即$\{0,1\}^*$中的成员）为输入而以 0 或 1 为输出的函数。计算 f 的一个算法是一组机械的规则，根据这组规则我们可以为任意输入 $x\in\{0,1\}^*$ 计算函数值 $f(x)$。所采用的计算规则是固定不变的（即同一组规则必须适用于所有输入），尽管其中每条规则被使用的次数是任意的。每条规则将用到如下定义的一个或多个“基本”操作：

1. 从输入中读取一个位；

2. 从算法使用的演草纸或工作空间中读取一个位（也可能是某个更大的字母表中的一个符号，例如集合$\{0,\cdots,9\}$中的一个十进制位）。

根据读取的值，

1. 在草稿纸上写出一个位或符号；

2. 要么停机并输出 0 或 1，要么重新选择下一步要执行的计算规则。

10

最后，图灵机的运行时间指的是上述基本操作被执行的次数。我们采用渐进术语描述运行时间。因此，如果图灵机在长度为 n 的输入上至多执行 $T(n)$ 个基本操作，则称该图灵机的运行时间为 $T(n)$。

下面列举图灵机模型的一些简单事实。

1. 该模型对定义中的几乎所有微调均是健壮的，例如将字母表由$\{0,1,\cdots,9\}$改为$\{0,1\}$，或者允许使用多张演草纸，等等。该模型的最基本的形式能够模拟其最复杂的形式执行计算，只是运行时间要慢多项式倍（其实是二次多项式倍）。因此，在复杂的图灵机模型上执行的 t 个步骤需要用相对简单的图灵机模型的$O(t^c)$个步骤来模拟完成，其中 c 是仅依赖于被模拟者和模拟者的常数；见 1.3 节。

2. 采用规范编码，每个算法（即图灵机）可以表示为一个位串。因此，一个算法或图灵机可能是另一个算法的输入——这将使得输入、软件和硬件之间的界限非常柔性。（注意，这种柔性的界限恰好是许多计算机技术的基础。）我们用 M_α 表示位串表示为 α 的图灵机。

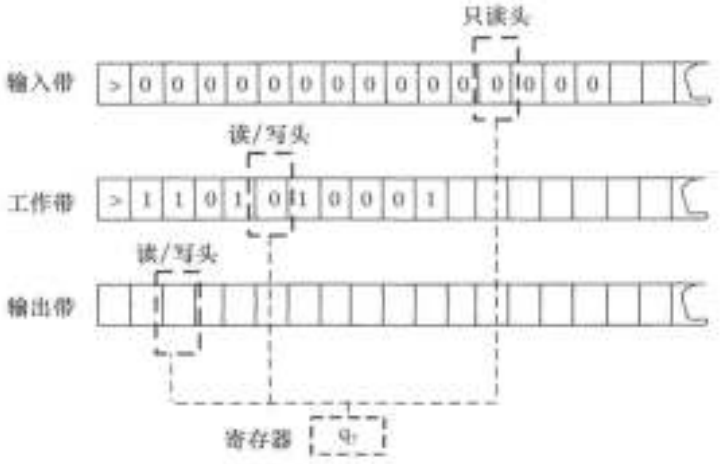
3. 存在通用图灵机$\mathcal{U}$使得任意给定其他图灵机的位串表示之后通用图灵机能够模拟执行计算操作。亦即，给定位串对$\langle x,\alpha\rangle$，通用图灵机能够模拟 M_α 在 x 输入上的操作过程，并且模拟过程非常高效：如果 M_α 的运行时间为 $T(|x|)$，则$\mathcal{U}$的运行时间为$O(T(|x|)\log T(|x|))$，见 1.4 节。

4. 简单地利用前面两个事实可以证明，存在不能被任何图灵机计算的函数，见 1.5 节。不可计算性与著名的哥德尔不完全定理关系密切，见 1.5.2 节。

1.2　图灵机

上述非正式的概念由 k-带图灵机按下面的方式具体实现，参见图 1-1。

⊖ 虽然假设具有无限内存初看起来似乎不太现实，然而就复杂性的讨论而言这不会造成任何影响。这是由于，我们仅讨论在任何输入上至多使用有限个计算步骤和内存单元的机器，尽管实际使用的计算步骤和内存单元的数量可能与输入有关。



11

图 1-1　具有输入带、工作带和输出带的一个 3-带图灵机的一个运行快照

演草纸

演草纸包含 k 条带，每条带均由单向无限延伸的很多单元构成，每个单元能够存储称为机器的字母表的有限集 Γ 中的一个符号。每条带均有一个带头，它具有在带上每次读或写一个符号的能力。机器的计算划分为一系列离散的时间步骤，带头在每个步骤中能够向左或向右移动一个单元。

机器的第一条带是输入带，其带头只能从该带上读取符号而不能写符号，因此这是一条只读带。另外的 $k-1$ 条读写带称为工作带，其中最后一条带是输出带，机器在计算终止前把最终计算结果写在输出带上。

图灵机还允许使用随机访问存储器[⊖]。然而，已经证明，图灵机的这种变形与标准的图灵机具有相同的计算能力(参见习题 1.9)。

有限操作/规则集

图灵机有有限种状态，表示为 Q。机器有一个"寄存器"，它能够在任何时刻记录机器处于 Q 中的何种"状态"。状态确定了机器在下一个计算步骤要采取的动作，包括：(1)直接读取 k 个带头所在存储单元的符号；(2)在 $k-1$ 条读写带上，将带头所在存储单元的符号替换为新的符号(也可以通过再次写下原来的符号而不改变带)；(3)修改寄存器使其记录来自有限集 Q 中的另一种状态(状态也可以保持不变，这只需选择与之前相同的状态)；(4)将每个带头向左或向右移动一个单元(或保持不动)。

图灵机可以看成是现代计算机的简化，它的带是计算机的内存，而转移函数[⊖]和寄存器则是计算机的中央处理器(CPU)。虽然如此，但最好还是将图灵机视为描述算法的一种简单的形式化方法。尽管描述算法的最佳方法是用白话，但这种形式化的描述方法将有助于对算法的数学分析(与此类似，用程序设计语言描述算法则是为了在计算机上执行算法)。

形式定义

形式上，一个图灵机 M 是一个三元组(Γ，Q，δ)，其中

⊖ 随机访问指的是能够在一个步骤内访问存储器的任意第 i 个存储单元，而不必通过一系列步骤将读写头移动到第 i 个单元。"随机访问"这个名称其实名不副实，因为其概念与随机性毫无关系。采用"索引访问"或许更贴切一些，然而，由于"随机访问"已被广泛采用，因此本书也采用这种叫法。

⊖ 转移函数的概念还未给出，请读者参阅下面的形式定义。——译者注

- 有限集 Γ，包含 M 的存储带上允许出现的所有符号。同时，假设 Γ 中还包含：一个特定的“空白符”，记为□；一个特定的“开始符号”，记为▷；以及 0 和 1。Γ 称为 M 的字母表。
- 有限集 Q，包含 M 的寄存器中可能出现的所有状态。此外，假设 Q 还包含一个特定的开始状态 q_{start} 和一个特定的终止状态 q_{halt}。
- 函数 δ：$Q\times\Gamma^k\to Q\times\Gamma^{k-1}\times\{L, S, R\}^k$，刻画了 M 在各个步骤中使用的规则，其中 $k\geqslant 2$。该函数称为 M 的转移函数（参见图 1-2）。

如果			则			
输入带上读取的符号	工作带/输出带上读取的符号	当前状态	移动输入带带头	工作带/输出带上写下的新符号	移动工作带/输出带带头	新状态
⋮	⋮	⋮	⋮	⋮	⋮	⋮
a	b	q	右移	b'	左移	q'
⋮	⋮	⋮	⋮	⋮	⋮	⋮

图 1-2　一个 2-带图灵机（即只有一条输入带和一条工作/输出带）的转移函数

如果图灵机的状态为 $q\in Q$，k 条带上当前读到的符号是 $(\sigma_1, \sigma_2, \cdots, \sigma_k)$，并且 $\delta(q, (\sigma_1, \sigma_2, \cdots, \sigma_k))=(q', (\sigma_2', \cdots, \sigma_k'), z)$，其中 $z\in\{L, S, R\}^k$，则在下一个步骤中后 $k-1$ 条带上的各个符号 σ 将被相应地替换为 σ'，图灵机将进入状态 q'，且 k 个带头将根据 z 相应地向左(L)、向右(R)移动或保持不动(S)（如果带头要在位于带的最左端的位置向左移动，则保持不动）。

除输入带之外，所有带的第一个单元初始化为开始符号▷而其余单元则初始化为空白符□。输入带的第一个单元初始化为开始符号▷，后续单元存储长度有限的非空符号串 x（即“输入”），其余所有单元是空白符□。所有带头位于带的左端，而机器处于特殊开始状态 q_{start}。这就是图灵机在输入 x 上的*初始格局*。计算过程的每个步骤就是如上自然段所述那样应用一次转移函数 δ。机器一旦处于特定的终止状态 q_{halt} 下，转移函数 δ 将不再允许机器修改带上的内容或改变机器的状态。显然，机器进入状态 q_{halt} 就已经停机。在复杂性理论中，我们通常只关注在任意输入上经过有限个操作步骤必然停机的图灵机。

例 1.1　布尔函数 PAL 如下定义：对于任意 $x\in\{0, 1\}^*$，如果 x 是回文则 $PAL(x)$ 等于 1，否则等于 0。亦即，$PAL(x)=1$ 当且仅当 x 从左向右读和从右向左读是一样的（即 $x_1x_2\cdots x_n=x_nx_{n-1}\cdots x_1$）。下面给出一个计算 PAL 的图灵机 M，它将用少于 $3n$ 个步骤完成计算。

图灵机 M 有 3 条带（即输入带、工作带和输出带）和字母表{▷, □, 0, 1}，它如下进行操作：

1. 将输入复制到读写工作带上；
2. 将输入带带头移动到输入的开始位置；

3. 输入带带头向右移动，而工作带带头向左移动。如果机器在带头移动过程的任何时刻发现了两个不同的值，则停机并输出 0。

4. 停机并输出 1。

下面再用更形式化的方法描述该图灵机。图灵机有 5 种状态$\{q_{start}, q_{copy}, q_{left}, q_{test}, q_{halt}\}$，其转移函数的定义如下：

1. 在开始状态 q_{start} 上：输入带带头向右移动，在工作带上写下开始符号 ▷ 之后工作带带头向右移动，机器状态变成 q_{copy}。（除非明确说明，机器不会修改输出带的内容或移动输出带带头。）

2. 在状态 q_{copy} 上：

- 如果从输入带上读到的符号不是空白符□，则输入带带头和工作带带头均向右移动，将输入带上读到的符号写在工作带上，机器停留于状态 q_{copy} 上。
- 如果从输入带上读到的符号是空白符□，则输入带带头向左移动，工作带带头停留于当前位置（不写任何符号），机器进入状态 q_{left}。

3. 在状态 q_{left} 上：

- 如果从输入带上读到的符号不是开始符号 ▷，则输入带带头向左移动，工作带带头停留于当前位置（不写任何符号），机器停留于状态 q_{left} 上。
- 如果从输入带上读到的符号是开始符号 ▷，则输入带带头向右移动，工作带带头向左移动（不写任何符号），机器进入状态 q_{test}。

4. 在状态 q_{test} 上：

- 如果从输入带上读到的符号不是空白符□，而从工作带上读到的符号是开始符号 ▷，则在输出带上写下 1，机器进入状态 q_{halt}。
- 否则，如果从输入带和工作带上读到的符号不同，则在输出带上写下 0，机器进入状态 q_{halt}。
- 否则，如果从输入带和工作带上读到的符号相同，则输入带带头向右移动，工作带带头向左移动，机器停留在状态 q_{test} 上。

显然，完整地描述一个图灵机非常繁琐，但却并不是总能提供更多的信息。尽管如此，你自己构造一两个完整的图灵机也是十分有益的（见习题 1.1）。本书其余部分将不再像上例一样给出每个图灵机的每个细节，而将在更高的层次上描述图灵机。对于具备了部分编程经验的读者，例 1.2 将使他们（至少在原理上）了解到如何构造图灵机来求解通过编程能解决的各种计算任务。

1.2.1　图灵机的表达能力

我们的第一印象是，图灵机是否概括了关于计算的直觉概念还不甚明了。读者最好能自己做一些简单的练习，比如将加法和乘法的标准算法用图灵机表示出来以完成相应的函数计算（参见习题 1.1），这必将大有裨益。做完这样的练习，就可以考虑下面的例子，把你用熟悉的编程语言编写的程序转换为图灵机。（反之，大多数编程语言也能模拟图灵机。）

14

例 1.2（用图灵机模拟一般编程语言）　本例需要关于计算的一些背景知识。我们概略地说明如何把用熟悉的编程语言（如 C 和 Java）编写的程序转换为等价的图灵机。首先，用编程语言编写的程序可以（用编译技术）翻译成用机器语言表示的程序，也就是一个指令序列，每个指令均是如下的几种简单操作之一：(a)从内存读取数据放入一个寄存器中，寄存器的个数是有限的；(b)把某个寄存器中的数据写入内存；(c)将某两个寄存器的值相

加后将结果存入第三个寄存器；(d)将某两个寄存器的值相乘后将结果存入第三个寄存器。所有这些操作均可以很容易地被图灵机模拟。内存和寄存器可以实现为图灵机的带，而指令则实现为图灵机的转移函数。例如，不难实现将两数相加或相乘的图灵机。2-带图灵机可以用来模拟计算机内存，它用一条带表示被模拟的内存，另一条带则用来实现二进制到一进制的转换。这样，对每个用二进制表示的数 i，图灵机可以借助第二条带将 i 转换成一进制，再据此读取或修改第一条带上的第 i 个位置的值。我们将上述过程的细节留作习题 1.8。

练习 1.10 要求为定制的程序设计语言严格地证明上述模拟。

1.3　效率和运行时间

现在，我们将运行时间的概念形式化。由于即便是最平凡的计算任务也需要读取输入，因此把执行的基本操作的个数表达为输入长度的函数，该函数可以定义为运行时间。

定义 1.3（函数的计算及运行时间）　令 $f:\{0,1\}^*\to\{0,1\}^*$ 和 $T:\mathbf{N}\to\mathbf{N}$ 是两个函数，M 是一个图灵机。如果将 M 初始化为任意输入 $x\in\{0,1\}^*$ 上的初始格局，则 M 停机时将 $f(x)$ 写在它的输出带上，我们就称 M 计算 f。如果 M 在任意输入 x 上计算 $f(x)$ 时最多只需要 $T(|x|)$ 个步骤，则称 M 在 $T(n)$ 时间[⊖]内计算 f。 15

例 1.4　不难验证，例 1.1 中识别回文的图灵机的运行时间是 $3n$。

时间可构造函数

函数 $T:\mathbf{N}\to\mathbf{N}$ 称为时间可构造的，如果 $T(n)\geqslant n$ 且存在一个在 $T(n)$ 时间内计算函数 $x\mapsto\underline{T(|x|)}$ 的图灵机 M（正如约定的那样，$\underline{T|x|}$ 是数 $T(|x|)$ 的二进制表示）。例如，n，$n\log n$，n^2，2^n 均是时间可构造函数的实例。本书用到的函数几乎均是时间可构造的，也只用时间可构造函数作为时间界限。（允许使用非时间可构造函数作为时间界限将得出反常的结论。）条件 $T(n)\geqslant n$ 确保算法有足够时间来读取其输入。

1.3.1　定义的健壮性

前面给出的图灵机的定义在许多细节方面还存在随意性。经过简单的论证可以看到，对该定义的大多数修改不会产生本质上不同的计算模型，因为我们定义的图灵机能够模拟这些新的模型执行计算。然而，从计算复杂性角度看，不仅需要论证我们的模型能够模拟其他模型，还需要论证这种模拟的高效性。现在，我们给出几个这样的结果。这些结果表明，如果我们愿意忽略运行时间上的多项式因子，则确切地选用哪个模型无关紧要。我们研究的模型的变形包括：将字母表 Γ 限定为 $\{0,1,\square,\triangleright\}$，限定机器只有一条工作带，允许带是双向无限的。本节对所有结果的证明均是概要式的——完善这些证明概要得出完整的证明过程是获得对图灵机的直观认识的好办法，参见习题 1.2，1.3 和 1.4。

论断 1.5　对任意 $f:\{0,1\}^*\to\{0,1\}$ 和时间可构造的 $T:\mathbf{N}\to\mathbf{N}$，如果 f 是字母表 Γ 上的图灵机 M 在 $T(n)$ 时间内可计算的函数，则 f 也能够被字母表 $\{0,1,\square,\triangleright\}$ 上的一个图灵机 $\widetilde{M}$ 在 $4\log|\Gamma|\,T(n)$ 时间内计算。

证明概要　设 M 是字母表为 Γ、状态集为 Q 的一个 k 带等价图灵机，它在 $T(n)$ 时间

⊖ 严格地讲，应该写成"T 时间"而非"$T(n)$ 时间"，但我们仍依据惯例写成 $T(n)$，以强调 T 是作用于输入长度上的函数。

内计算函数 f。我们给出计算 f 的另一个字母表为{0，1，□，▷}，状态集为 Q' 的 k 带等价图灵机 $\widetilde{M}$。由 M 构造 $\widetilde{M}$ 的基本思想很简单，即对 Γ 中的每个符号用 $\log|\Gamma|$[⊖]个位进行编码。于是，$\widetilde{M}$ 的每条工作带编码 M 的一条带，但是 M 的带上的每个单元对应于 $\widetilde{M}$ 上相应带上的 $\log|\Gamma|$ 个单元(参见图 1-3)。



图 1-3　字母表{□，▷，a，b，…z}上的图灵机 M 可以被字母表{□，▷，0，1}上的图灵机 $\widetilde{M}$ 模拟，其中 M 的每个存储单元被 $\widetilde{M}$ 的 5 个存储单元编码

为了模拟 M 的一个步骤，机器 $\widetilde{M}$ 需要完成(1)用 $\log|\Gamma|$ 个步骤从每条带上读取 $\log|\Gamma|$ 个位，从每条带上读取的位串是 Γ 中的一个符号的编码；(2)利用其状态寄存器存储读取的符号；(3)利用 M 的转移函数计算 M 在这些信息要写出的符号和 M 要进入的状态；(4)将
16 上述信息存储在 $\widetilde{M}$ 的状态寄存器中；(5)用 $\log|\Gamma|$ 个步骤将待写符号的编码写到 $\widetilde{M}$ 的带上。

可以证明，只要能够在 $\widetilde{M}$ 的寄存器状态上同时表示 M 的状态，Γ 中的 k 个符号和介于 1 到 $\log|\Gamma|$ 之间的计数器，则上面由 M 构造 $\widetilde{M}$ 的想法将是可行的。由此，可以构造出一个状态数量不超过 $c|Q|\,|\Gamma|^{k+1}$ 的图灵机 $\widetilde{M}$，其中 c 是一个绝对常数。(一般而言，我们可以用状态空间较大的一个寄存器来模拟状态空间较小的多个寄存器。例如，状态分别取自 A，B，C 的三个寄存器可以用状态取自 $A\times B\times C$ 的一个寄存器来模拟，其中模拟者的状态空间的大小为 $|A|\,|B|\,|C|$。)

不难证明，对于任意输入 $x\in\{0,1\}^*$，如果图灵机 M 在 $T(n)$ 个步骤内为 x 输出 $f(x)$，则 $\widetilde{M}$ 将在 $4\log|\Gamma|\,T(n)$ 个步骤内输出同样的值。■

现在，考虑限定图灵机只用一条带会造成什么影响。这意味着，图灵机只用一条读写带，它既是输入带又是工作带和输出带(很多本科生教材(如[Sip96])将这种图灵机作为标准计算模型)。我们将证明，多带图灵机的运行时间至多是单带图灵机运行时间的平方。运行时间的平方增加对某些语言来讲是固有的，比如例 1.1 所考虑的回文识别；参见本章注记。

论断 1.6　定义单带图灵机是仅使用一条读写带的图灵机，它将这条带既用作输入带又用作工作带和输出带。对任意 $f:\{0,1\}^*\to\{0,1\}$ 和时间可构造的 $T:\mathbf{N}\to\mathbf{N}$，如果 f 是 k 带图灵机 M 在 $T(n)$ 时间内可计算的，则 f 也能够被一个单带图灵机 $\widetilde{M}$ 在 $5kT(n)^2$ 时间内计算。◀

证明概要　证明的思想同样很简单。$\widetilde{M}$ 用它仅有的一条带对机器 M 的 k 条带进行编码，其中位置 1，$k+1$，$2k+1$，…用于对 M 的第一条带编码，位置 2，$k+2$，$2k+2$，…用于对 M 的第二条带编码，以此类推(参见图 1-4)。对于 M 的字母表中的每个符号 a，$\widetilde{M}$ 的字母表中将包含两个字符 a 和 $\dot{a}$。编码 M 的每条带时，恰有一个"·型"的符号出现，该符号用以表明这条带的带头所处的位置(参见图 1-4)。$\widetilde{M}$ 将输入存放在带的前 $n+1$ 个位置上，因此除了开始阶段为实现上述编码需要用 $O(n^2)$ 个步骤将输入逐位复制到带的其余位置之外，$\widetilde{M}$ 不会访问带上的前 $n+1$ 个位置。

⊖ 注意，根据我们的约定，对数以 2 为底。而且，在必要时，非整数值向上近似取整。

要模拟 M 的一个步骤，机器 $\widetilde{M}$ 需要扫描工作带两趟。第一趟从左向右扫描，将标有"*"的 k 个符号记录到寄存器中。然后，$\widetilde{M}$ 用 M 的转移函数确定新状态、要写出的符号以及带头移动方向。最后，$\widetilde{M}$ 通过第二趟自右向左地扫描工作带，更新 M 的每条带的编码。显然，$\widetilde{M}$ 的输出与 M 的相同。并且，由于 M 在长度为 n 的输入上绝不会访问带上位于位置 $T(n)$ 之外的存储单元，因此 $\widetilde{M}$ 也不会访问工作带上位于位置 $2n+kT(n)\leqslant(k+2)T(n)$ 之外的单元。这意味着，对于 M 的至多 $T(n)$ 个步骤中的每个步骤，$\widetilde{M}$ 至多执行 $5\cdot k\cdot T(n)$ 个步骤（正向和逆向两趟扫描需要大约 $4\cdot k\cdot T(n)$ 步，另外还需要一些步骤来更新带头移动并相应地进行标记）。■

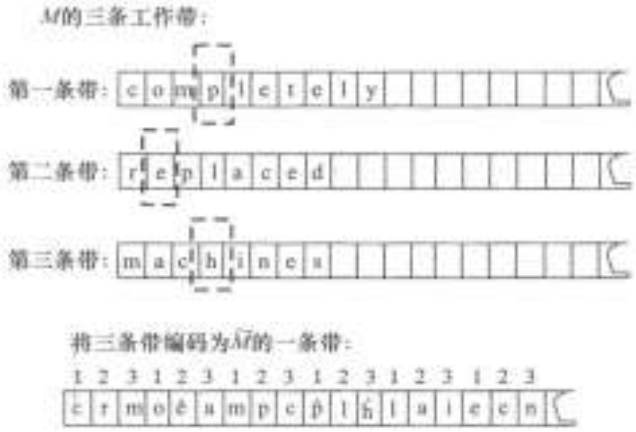


图 1-4　单带图灵机 $\widetilde{M}$ 模拟 3 带图灵机 M

评注 1.7（敏感图灵机）　稍加小心，你将注意到，论断 1.6 中的证明过程构造的图灵机 $\widetilde{M}$ 具有如下性质：其带头的移动不依赖于输入而仅依赖于输入的长度。这就是说，对于任意输入 $x\in\{0,1\}^*$ 和 $i\in\mathbf{N}$，在图灵机 M 的第 i 个步骤时，各条带的带头所处的位置是 $|x|$ 和 i 的函数。具有这种性质的图灵机称为敏感的。任意图灵机均可以被一个敏感图灵机模拟，后面我们将利用这一事实简化一些证明过程（参见习题 1.5，1.6 和定理 2.10 的证明）。

论断 1.8　双向图灵机定义为所有带均双向无限延伸的图灵机。对任意 $f:\{0,1\}^*\to\{0,1\}$ 和时间可构造的 $T:\mathbf{N}\to\mathbf{N}$，如果 f 是双向图灵机 M 在 $T(n)$ 时间内可计算的函数，则 f 也能够被一个标准的（单向）图灵机 $\widetilde{M}$ 在 $4T(n)$ 时间内计算。

证明概要　证明的思想如图 1-5 所示。如果 M 的字母表为 Γ，则 $\widetilde{M}$ 的字母表为 Γ^2（亦即，$\widetilde{M}$ 的每个符号是 M 字母表中的一对符号）。我们将 M 的每条双向无限延伸的带在任意位置"对折"就产生 $\widetilde{M}$ 的一条（单向无限延伸的）标准带。这样，$\widetilde{M}$ 带的每个单元对应 M 带上的两个单元。开始时，$\widetilde{M}$ 将忽略所读单元的第二个符号，同时根据 M 的转移函数采取动作。然而，一旦转移函数命令 $\widetilde{M}$ 跨过带的"边界"，则 $\widetilde{M}$ 将忽略所读单元的第一个符号而采用第二个符号。在跨越边界时，需要将带头左移翻译为右移，反之亦然。如果再次跨越边界，则重新使用存储单元的第一个符号并正常地移动带头。■



图 1-5　我们用字母表为 Γ^2 的图灵机 $\widetilde{M}$ 来模拟字母表为 Γ 的具有双向无限带的图灵机 M，其中 $\widetilde{M}$ 的每条带编码 M 的一条"对折"后的带

对图灵机其他修改均不会造成重要影响，这些修改包括使用二维或三维的带，允许机器对带进行随机访问，限定输出带为只写带(参见习题 1.7 和 1.9，教材[Sip96，HMU01]还讨论了图灵机的更多示例)。特别指出，图灵机的这些修改均不会改变将在 1.6 节定义的由多项式时间内可计算的判定问题构成的类 **P**。

1.4　机器的位串表示和通用图灵机

几乎显而易见的是，图灵机可以表示为位串：这只需将图灵机描写在纸上，再用 0，1 序列将描写过程编码即可。编码图灵机得到的位串可以作为另一个图灵机的输入。这个简单的观察结果有着十分深刻的内涵，因为它使得软件、硬件和数据之间的区别变得模糊。历史上，该结果曾直接推动人们发明了通用电子计算机。计算机也是一个图灵机，通过在其上加载恰当的程序(或软件)，它就能用来自动地求解指定的任意计算任务。

由于本书将大量使用图灵机的位串表示的概念，因此稍微更详细地讨论这种表示是值得的。由于图灵机的行为取决于它的转移函数，因此将转移函数的所有输入和输出(很容易将它们编码为$\{0,1\}^*$中的串)列出来就可以得到图灵机的编码[⊖]。为了方便后面的讨论，我们还假定图灵机的位串表示满足下列性质：

1. $\{0,1\}^*$中每个串均表示一个图灵机。

19 这条假设很容易保证其合理性，只需认为无效的位串编码为一个规范的平凡图灵机，例如它在任意输入上立刻停机并输出 0。

2. 每个图灵机存在无穷个位串表示。

为保证这条假设的合理性，可以认为图灵机的位串表示后面紧跟任意个可以忽略的 1。这类似于很多程序设计语言的注释机制(如 C，C++和 Java 用/*…*/)，它允许在程序后面添加任意个多余的符号。尽管这似乎是一个挑剔的假设，但它有助于一些证明过程的简化。

我们用$\langle M\rangle$表示图灵机 M 的二进制位串表示。如果 α 是一个位串，则 M_α 是它所表示的图灵机。根据约定，我们也经常用 M 表示图灵机及其位串表示。习题 1.11 要求读者给出图灵机的一个表示方案，使其满足上面的两条性质。

1.4.1　通用图灵机

第一个注意到通用图灵机可能存在的人是图灵(Turing)，他证明了给定任意图灵机 M 的位串表示作为输入，通用图灵机可以模拟 M 的运行。对当代的人而言，台式或便携式的通用计算机已经是司空见惯的事情，因此人们已经理所当然地接受了通用图灵机的存在性。然而，留意当初的这个违背直觉的结论仍是有益的。通用图灵机的各种参数均是固定的，包括字母表的大小、状态的数量和带的数量。被模拟的图灵机的各项参数均可能比通用图灵机的大得多。这之所以不是障碍，得益于编码的能力。即使通用图灵机的字母表很简单，其他图灵机的状态和转移函数也可以编码后放于通用图灵机的带上，然后由通用图灵机一步一步地模拟执行。

现在，我们给出一个计算高效的图灵构造，该构造是亨尼(Hennie)和斯特恩斯(Ste-

⊖ 注意，字母表的大小、带的数量和状态空间的大小均可以从转移函数表推知。我们也可以调整转移函数表的顺序使得 q_{start} 和 q_{halt} 是图灵机的第一个和最后一个状态。类似地，我们还假定 $\triangleright$，$\square$，0，1 是图灵机字母表的前四个符号。

arms)[HS66]给出的。在介绍其核心思想之前，我们先证明其宽松的形式，亦即将下面结论中的 $T\log T$ 换成 T^2。此外，由于本书将多次用到高效通用图灵机，因此本章结束前我们将给出它的完整证明(参见 1.7 节)。

定理 1.9（高效通用图灵机）　存在图灵机$\mathcal{U}$使得$\mathcal{U}(x, a)=M_a(x)$对于任意 $x, a\in\{0,1\}^*$ 成立，其中 M_a 是 a 表示的图灵机。并且，如果 M_a 在输入 x 上至多运行 T 步之后就停机，则$\mathcal{U}(x, a)$将在 $CT\log T$ 步内停机，其中 C 是一个独立于 $|x|$ 而仅依赖于 M_a 的字母表的大小、带的数量和状态的数量的常数。

程序设计课程的一个常见练习是，用特定编程语言为同一语言编写一个解释程序(解释程序以一个程序 P 为输入并输出程序 P 的执行结果)。定理 1.9 可视为这个习题的一个变形。

定理 1.9 的宽松形式的证明　通用图灵机$\mathcal{U}$的输入是 x, a，其中 a 表示某个图灵机 M，输出是 $M(x)$。关键在于，我们可以假设：(1)M 除了输入带和输出带之外只有一条工作带；(2)M 的字母表为{▷，□，0，1}。这是由于，$\mathcal{U}$可以像论断 1.5 和 1.6 的证明过程那样将任意图灵机 M 的位串表示转换成一个等价的满足上述假设的图灵机$\widetilde{M}$的表示。注意，这种转换可能使得运行时间平方增加(亦即一个运行时间为 T 的图灵机转换后它的运行时间将变成 $C'T^2$，其中 C' 只依赖于 M 的字母表大小和带的数量)。

20

通用图灵机$\mathcal{U}$使用字母表{▷，□，0，1}。除了输入带和输出带之外，它还使用三条工作带(参见图 1-6)。$\mathcal{U}$按照与 M 相同的方式使用其输入带、输出带和一条工作带。此外，$\mathcal{U}$用第二条工作带存储 M 的转移函数值的表(先将转移函数按前面注记中提到的方法进行转换)，用第三条工作带存储 M 的当前状态。为了模拟 M 的一个计算步骤，$\mathcal{U}$先通过扫描 M 的转移函数表和当前状态来确定 M 的下一个状态和要写出的符号以及带头移动方向，然后执行相应的动作。可以看到，M 的每个计算步骤需要通过$\mathcal{U}$的 C 个计算步骤来模拟，其中 C 依赖于 M 的转移函数表的规模。



图　1-6

上面给出了图灵机$\mathcal{U}$在较高层次上的描述，这种描述可以进一步转换成$\mathcal{U}$的精确描述，但是，我们将这项工作留给读者。为此，有益的做法是，先考虑如何用熟悉的编程语言实现上面的各个步骤，然后再把编写的程序转换成对一个图灵机的描述。■

具有时间界限的通用图灵机

通用图灵机$\mathcal{U}$的一种变形在某些场合也很有用。在这种变形中，通用图灵机的输入除了 x, a 之外还包含一个数 T，它输出 $M_a(x)$当且仅当 M_a 在 x 上至多计算 T 步就停机(否则，就输出其他符号以表示失败)。在定理 1.9 的证明过程中，在$\mathcal{U}$上添加一个时间计数器，可以证明仍有一个具有相同时间界限的通用图灵机满足该定理。时间计数器用于记录目前已被模拟执行的计算步骤的个数。

1.5　不可计算性简介

下面的事情看起来似乎很"显然"：只要时间充分，任何函数均是可被计算的。然而，

21 这已被证明是错误的。因为，存在这样的函数，它们在无穷步骤内不能被计算！本节简要介绍这一事实和由此引出的学科分支。尽管严格地讲不可计算性对复杂性而言不是必需的，但它却构成了复杂性的知识背景。

下面的定理证明了不可计算函数的存在性。(事实上，该定理证明了值域为{0，1}的不可计算函数(即语言)的存在性。值域为{0，1}的不可计算函数也就是众所周知的不可判定语言。)定理的证明用到了对角线方法，这种方法在复杂性理论中也很有用：参见第3章。

定理 1.10 存在不能被任意图灵机计算的函数 $UC: \{0,1\}^* \to \{0,1\}$。

证明 函数 UC 如下定义。对于任意 $\alpha \in \{0,1\}^*$，如果 $M_\alpha(\alpha)=1$，则 $UC(\alpha)=0$；否则(若 $M_\alpha(\alpha)$输出其他值或进入无限循环)$UC(\alpha)=1$。

为导出矛盾，我们假设函数 UC 是可计算的，因此，存在图灵机 M 使得 $M(\alpha)=UC(\alpha)$对任意 $\alpha \in \{0,1\}^*$ 成立。于是，特别地，有 $M(\langle M\rangle)=UC(\langle M\rangle)$。但这不可能，因为由函数 UC 的定义可知，

$$UC(\langle M\rangle)=1 \quad \Leftrightarrow \quad M(\langle M\rangle)\neq 1$$

■

图 1-7 展示了上面的证明技术被称为对角方法的原因。

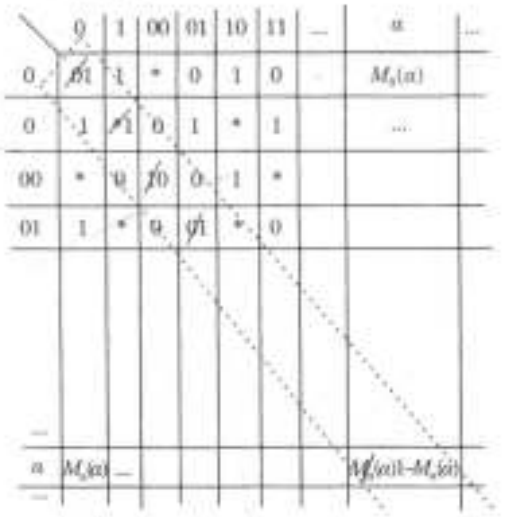


图 1-7　假设所有位串按字典序排列，表格对所有 α，x 记录 $M_\alpha(x)$的值，其中 M_α 是 α 表示的图灵机，并用 * 表示 $M_\alpha(x)$的值不在{0，1}中或 M_α 在输入 x 上不停机的情况。于是，函数 UC 的定义恰好是上述表格对角线元素的"否定"。由于表格的行表示了所有图灵机，所以可以得出结论：UC 不能被任何图灵机计算

1.5.1　停机问题

读者可能不禁要问，为什么我们要研究上面定义的函数 UC 是不是可计算的呢？什么样的人会关心到底如何计算这种刻意构造的
22 函数呢？

现在我们展示一个更自然的不可计算函数。函数 HALT 以序对$\langle \alpha, x\rangle$为输入，它输出 1 当且仅当 α 表示的图灵机 M_α 在输入 x 上会在有限步骤内停机。这肯定是我们需要计算的函数。因为，给定一个计算机程序和输入之后，我们肯定希望知道该程序在该输入上是否会陷入无限循环。如果计算机能够计算函数 HALT，则设计无 bug 的计算机软件和硬件将变得容易得多。遗憾的是，现在我们可以证明计算机计算不了这个函数，即使运行时间允许任意的长。

定理 1.11 函数 HALT 不能被任何图灵机计算。

证明 为了导出矛盾，假设存在计算函数 HALT 的图灵机 M_{HALT}。我们用 M_{HALT}构造一个计算函数 UC 的图灵机 M_{UC}，这与定理 1.10 矛盾。

图灵机 M_{UC}的构造很简单。对于输入 α，M_{UC}运行 $M_{HALT}(\alpha,\alpha)$。如果结果是 0(这意味着 M_α 在 α 上不停机)，则 M_{UC}输出 1。否则，M_{UC}用通用图灵机 U 计算 $b=M_\alpha(\alpha)$。如果 $b=1$，则 M_{UC}输出 0；否则，M_{UC}输出 1。

在 $M_{HALT}(\alpha,\alpha)$会在有限步骤内输出 $HALT(\alpha,\alpha)$的假设下，图灵机 M_{UC}将输出 $UC(\alpha)$。

■

定理 1.11 证明过程中的技术称为归约。定理的证明已经表明，如果假设存在计算 HALT 的算法，则必存在计算 UC 的算法；这就将对 UC 的计算归约到了对 HALT 的计算。在本书中，我们将遇到许多归约。通常，归约技术用于证明问题 B 至少同问题 A 一样难，这正是通过证明"如果给定求解问题 B 的算法，则存在求解问题 A 的算法"来完成的。

还有许多有趣的不可计算(也称不可判定)函数，参见习题 1.12。甚至，还有一些不可计算函数，它们看起来似乎与图灵机或算法毫无关系。例如，下面的问题不能被任何图灵机在有限时间内求解：给定一族整系数的多项式方程，问这些方程是否存在整数解(亦即，是否存在变量的整数赋值满足所有方程)。这就是著名的丢番图问题。1900 年，希尔伯特曾将寻找丢番图方程的求解算法这一问题作为最难的 23 个开放数学问题之一提出。本章的注记提到了更多关于可计算理论的资源。

1.5.2　哥德尔定理

1900 年，当年最卓越的数学家戴维·希尔伯特提出了一项雄心勃勃的研究计划。该计划旨在为所有数学建立严格的公理系统，以最终确保所有为真的结论均能被严格地证明。罗素(Russell)、怀特黑德(Whitehead)、策梅洛(Zermelo)、弗兰克尔(Fraenkel)等数学家在接下来的数十年中各自提出了自己的公理系统，但他们谁也不能证明他们的公理系统既是完备的(亦即，能证明所有正确的数学结论)又是可靠的(亦即，不会证明出错误的结论)。1931 年，库尔特·哥德尔证明了所有这些努力都注定要失败，这震惊了整个数学界。因为他证明了：对于任意可靠的公理和推理规则的系统S，必存在正确的数论结论不能在S中被证明。下面，我们给出这个结果的证明概要。注意，我们给出的讨论并不针对哥德尔的更强结论：亦即，数学上任何一个足够强的公理系统均不能证明其自身的一致性。采用下面的思想，要证明这个更强的结论也不是很难。

哥德尔的工作直接推动了图灵和邱奇在可计算性方面的工作。我们下面颠倒这两项工作之间的关系；亦即，用不可计算性概略地证明哥德尔定理。我们观察到的主要结果如下所述。在任意一个足够强的公理系统中，我们能为任意输入(a, x)构造一个数学结论$\phi_{a,x}$使得$\phi_{a,x}$为真当且仅当 HALT$(a, x)=1$。(下面是这个构造过程的梗概。)如果公理系统是完备的，则它必然能够证明$\phi_{a,x}$和$\neg\phi_{a,x}$之一。如果公理系统是可靠的，则它不能同时证明$\phi_{a,x}$和$\neg\phi_{a,x}$。因此，如果公理系统既是完备的又是可靠的，则停机问题的如下算法对任意输入必然在有限步骤内终止。"给定输入(a, x)，开始逐一枚举长度有限的每个位串；对于枚举的每个串，检查它是否表示公理系统中对$\phi_{a,x}$或$\neg\phi_{a,x}$的证明；在等待充分长的时间之后，$\phi_{a,x}$和$\neg\phi_{a,x}$之一的证明必然出现在枚举的位串中；此时，HALT(a, x)是等于 1 还是等于 0 也就揭晓了；只需相应地进行输出就可以了。"(注意，上述过程隐式地使用了一个简单的事实：即，用图灵机很容易验证基于公理系统的证明的正确性，这是因为，证明的每个步骤都是在之前的证明步骤上机械地应用公理。)

现在，我们再概要地给出目标结论$\phi_{a,x}$的构造过程。假定公理系统能够使用下述的操作符表达关于自然数的结论，这些操作符包括加法(+)和乘法(×)运算符、比较操作符(=，<，>)以及与($\wedge$)、或($\vee$)、非($\neg$)等逻辑操作符；所采用的语言还允许使用全称量词($\forall$)、存在量词($\exists$)和常数 1(另外的常数 c 可通过将 1 相加 c 次来获得)。例如，"x 整除 y"的形式表达式是 DIVIDES$(x, y)=\exists k\,[y=k\times x]$，而"$y$ 是素数"的形式表达式为 PRIME$(y)=\forall_x[(x=1)\vee(x=y)\vee\neg$DIVIDES$(x, y)]$，其中 DIVIDES$(x, y)$是相应形式表达式的缩写。

我们可以用整数表示字符串，因此图灵机及其输入和带也可以表示成整数。再注意到，图灵机的一个基本操作仅影响其带上一系列位中的一个(或几个，如果图灵机有多条带)，这可以看成在表示带上内容的整数或字符串上执行一个简单算术操作。再花些工作将这些操作形式地表示出来，最终将得到表达式 $\varphi_{\langle M,x\rangle}(t)$，它为真当且仅当图灵机 M 在输入 x 上的计算会在 t 步内停机。于是，M 在输入 x 上停机当且仅当 $\exists_t \varphi_{\langle M,x\rangle}(t)$，它即是我们想要的数学结论。我们将细节留作习题 1.13。

注意，上述构造最早是由图灵给出的。该构造过程还表明，所有正确数学结论构成的集合是不可判定的。这同时还表明，希尔伯特著名的判定问题无解。(希尔伯特曾提出用"机械过程"(现译作"算法过程")来判定数学结论的真伪。)

1.6　类 P

复杂性类是在给定资源界限下能被计算的所有函数构成的集合。现在，我们引入第一
24 个复杂性类。为技术方便计，本书大多数时候将只讨论布尔函数；亦即，只输出一个二进制位的函数。将这类函数定义为判定问题或判定语言。我们称一个图灵机判定一个语言 $L\subseteq\{0,1\}^*$，如果该图灵机计算函数 $f_L:\{0,1\}^*\to\{0,1\}$，其中 $f_L(x)=1\Leftrightarrow x\in L$。

定义 1.12（类 DTIME）　设 $T:\mathbf{N}\to\mathbf{N}$ 是一个函数。称语言 $L\in\mathbf{DTIME}(T(n))$ 当且仅当存在运行时间为 $c\cdot T(n)$ 的图灵机判定语言 L，其中 $c>0$ 是常数。

记号 DTIME 中的 D 指的是"确定型的"。更确切地讲，本章引入的图灵机应称为确定型图灵机。因为对于任意输入 x，图灵机执行计算的方式恰有一种。在后续章节，我们将看到其他类型的图灵机，包括非确定型图灵机和概率型图灵机。

现在，我们将"高效计算"的概念精确化。将高效计算等同于多项式运行时间，即对某个常数 $c>0$ 而言至多为 n^c 的运行时间。这一概念由下面的复杂性类刻画，其中 **P** 表示"多项式的"。

定义 1.13（类 P）　$\mathbf{P}=\bigcup_{c\geq 1}\mathbf{DTIME}(n^c)$。

这样，引言中提出的关于"晚宴问题是否存在高效算法?"这一问题可以重述为"INDSET 属于 **P** 吗?"，其中 INDSET 是例 0.1 定义的语言。

例 1.14（图连通性问题）　在图连通性问题中，我们给定图 G 和它的两个顶点 s，t，要求判定 s 在 G 上是否可以连通到 t。该问题属于 **P**。证明这一论断可以采用本科阶段学过的深度优先搜索算法。算法从 s 出发，逐一检查 G 的边，并对访问过的边进行标记。后续步骤访问的边是与之前访问过的边邻接的仍未被访问的边。至多经过个 $\binom{n}{2}$ 步骤之后，所有的边要么已被访问要么永远无法访问。　◀

习题 1.14 讨论了属于 **P** 的其他语言。

例 1.15　本例通过一些例子来强调类 **P** 的定义中的两个方面。首先，该类仅含判定问题。于是，我们不能有"整数乘法属于 **P**"这种说法，而应该称该问题的判定形式属于 **P**，其判定形式是如下的语言：

$$\{\langle x,y,i\rangle : xy \text{ 的第 } i \text{ 个位等于 } 1\}$$

25 其次，运行时间是输入中位的个数的函数。考虑有理数线性方程组的求解问题：也就是，给定序对 $(A,\boldsymbol{b})$，其中 A 是 $m\times n$ 的有理数矩阵而 $\boldsymbol{b}$ 是一个 m 维有理数向量，要求判定是否存在 n 维向量 $\boldsymbol{x}$ 使得 $A\boldsymbol{x}=\boldsymbol{b}$。标准的高斯消元法用 $O(n^3)$ 个算术操作求解该问题。但

在图灵机上，每个算术操作均需要按部就班地按照小学算术方法逐位地进行。因此，要证明该判定问题属于 **P**，我们必须证明高斯消元法(或求解问题的其他算法)在图灵机上的运行时间是表示 a_1、a_2、…、a_n所需的二进制位的个数的多项式。这就是说，以高斯消元法为例，需要证明计算过程涉及的所有中间数据可以用多项式个二进制位来表示。幸运的是，事实恰恰如此(相关结论参见习题 2.3)。◀

1.6.1　为什么模型选择无关紧要

我们用图灵机定义"可计算"语言的各种复杂性类及类 **P**。如果选用其他计算模型，这些类会有区别吗？在发现了计算但采用其他模型而不是图灵机作为计算模型的高等外星人眼里，这些类会有区别吗？

我们已经遇到图灵机模型的各种变形，用其中最弱的模型来模拟最强的模型将导致运行时间平方增加。因此，作为可计算问题的集合，多项式时间在这些变形上是一样的。

在图灵和邱奇的工作发表之后的最初几十年中，很多其他的计算模型被发现，其中有些模型十分怪异。当时人们很容易地就证明了图灵机能够模拟其他计算模型，模拟过程至多使得运行时间多项式地变高。因此，在这些模型上类似地定义的类 **P** 不会比图灵机定义的 **P** 更大。

绝大多数科学家均相信**邱奇-图灵(CT)命题**，它断言任何可被物理实现的计算装置均可以被图灵机模拟，不管这种装置是基于芯片、DNA、中子的还是基于外星人技术的。这就是说，任何其他模型上的可计算问题的集合不会比图灵机上可计算问题的集合更大。(CT 命题不是定理，正如我们目前的理解，它仅仅是对世界的本质的一种信念。)

然而，高效可计算问题方面的情形就不那么明了。**强 CT 命题**断言，任何可物理实现的计算模型均可以被图灵机以多项式代价模拟(换句话说，模型的 t 个计算步骤可以用图灵机的 t^c 个步骤来模拟，其中 c 是依赖于模型的常数)。如果强 CT 命题成立，则连外星人定义的类 **P** 也同本节定义的 **P** 别无二致。然而，强 CT 命题是存在争议的，特别是在量子计算机模型(参见第 10 章)的出现之后。量子计算机似乎不能在图灵机上高效模拟。但另一方面，目前也不清楚量子计算机能否被物理实现。

1.6.2　P 的哲学意义

感觉上，类 **P** 是用"可行的"判定过程来刻画判定问题。然而颇受争议的是，$\mathbf{DTIME}(n^{100})$是否真是"可行的"计算，这是因为即便 n 只有中等大小，n^{100}也是天文数字。但是，在实际当中，证明一个问题属于 **P** 时所用算法的时间通常是 n^3或 n^5(外加一些合理的常数)而不是 n^{100}。(某个计算问题的第一个多项式算法也曾出现过高复杂性，如 n^{25}，但它很快就被其他人简化成 n^6时间的算法。)

注意，只有在特定的场合下类 **P** 才有意义。如果要设计一个在最新电脑上花不超过 1 秒的时间运行得出结果的算法程序，则图灵机将是粗糙的模型，因为编程的过程必须还要考虑硬件等因素。然而，如果要解决的问题是问是否存在亚指数时间算法计算某个语言，比如例 0.1 定义的语言 INDSET，则复杂性为 n^{25}的算法也将是一个巨大突破。

从程序员的角度看，**P** 是一个顺其自然的类。假设要求程序员给出"高效"计算的定义。首先，她会大概地认为线性时间或平方时间内完成的计算是高效的。接下来，由于程序员编写的程序通常会调用其他程序(或子例程)，所以她会自然而然地认为，高效计算所

执行的计算和所调用的例程均应该是高效的。由此得出的关于"高效计算"的概念恰好是类**P**[Cob64]。

1.6.3　P的争议和解决争议的一些努力

现在，我们给出关于类**P**的定义的一些争议和解决这些争议的过程中出现的以下相关复杂性类。

*最坏情况精确计算过于严格。*类**P**的定义考虑的仅仅是在所有可能的输入上计算函数的算法。一个争议之处是，所有输入并不都出现在实际场合中，而算法只需对实际场合下出现的输入上有效即可。平均复杂性部分地解决了这一争议，并由此产生了平均复杂性意义下类似**P**的复杂性类，参见第18章。我们的观点是，限定在"实际场合"讨论**P**仅有纯技术意义。

类似地，在遇到像图的最大独立集这样大小的计算函数时，人们通常更愿意讨论问题的近似解。第11章和第22章对近似复杂性进行了严格处理。

*其他可物理实现的模型。*前面已经提到了强邱奇-图灵命题，它断言在任意可物理实现的计算模型上类**P**不会更大，但其中还有一些微妙之处有待讨论。

(a) *精度问题。*图灵机用离散符号进行计算，但现实生活中某些物理量可以取**R**中的实数。因此，人们也可以基于能操作实数的某些物理现象来构想执行实数计算的模型。由于存在精度问题，图灵机只能近似地模拟实数计算。即便这样，精度问题也不是图灵机遇到的本质性障碍(尽管有些研究者不认同这种观点)。毕竟，由于现实生活中设备存在噪音，因此物理量的测量只能精确到有限精度。因此，物理过程不可能涉及任意精度，继而图灵机当然可以用有限精度来完成模拟。即便这样，我们仍将在第16章中考虑图灵机的一种修改，使其能够将实数运算当作基本操作。由此产生的复杂性类跟标准复杂性类之间的联系十分密切。

27

(b) *随机性的采用。*前面定义的图灵机是确定型的。如果计算的领域中存在随机性，则人们也可以构想利用随机位源(如投掷硬币)的计算模型。第7章讨论了允许投掷硬币的图灵机并研究了复杂性类**BPP**，它是这种计算模型上与**P**类似的复杂性类。然而，我们在第19章和第20章将会看到，随机型计算模型极可能并不比确定型计算模型更强。

(c) *量子力学的采用。*一个更聪明的计算模型使用了量子力学中违背直觉的一些性质。我们在第10章定义了复杂性类**BQP**，它是类**P**在量子计算模型中的推广。我们将看到**BQP**中的一些问题，目前还不知道它们是否属于**P**(尽管目前仍未证明**BQP**≠**P**)。然而，量子计算模型是否能被物理实现还是未知的。并且，量子计算机目前似乎也只能高效地求解为数不多的几个不属于**P**的问题。因此，研究**P**时获得的知识仍可能适用于量子计算机。

(d) *其他怪诞物理知识(如弦论)的采用。*目前，许多物理理论似乎都得到同样的复杂性类**BQP**，尽管许多理论还有待深入理解。

*判定问题的限制太强。*有些计算问题不易表达成判定问题。事实上，本书将引入几个复杂性类来处理某些任务的计算复杂性，这些任务包括非布尔函数的计算、搜索问题的求解、优化问题的近似求解、交互式任务，等等。然而，判定问题的框架已被证明具有强大的表达能力，本书也经常使用这种框架。

1.6.4　埃德蒙兹的引言

我们用埃德蒙兹的引言[Edm65]结束本节。埃德蒙兹曾在其著名的论文中给出了求解最大匹配问题的多项式时间算法，他在论文中这样解释其研究结果的意义：

> 就实践目的而言，计算的细节至关重要。然而，我的目的仅仅在于竭尽所能地将高效算法的存在性证明得更有魅力。字典对"高效"的解释是"操作或性能上是足够的"。这基本上就是我需要的含义——从人们普遍相信最大匹配没有高效算法的角度看。
>
> （最大匹配问题）有一个显而易见的有穷算法，但该算法的难度[⊖]随图的大小指数地增长。该问题是否存在一个难度随图的大小代数增长[⊜]的算法绝不是显而易见的。
>
> 如果问题规模的测度是合理的且假设规模取任意大的值时，渐进地估计……算法难度的阶在理论上是重要的。不可能通过使算法在较小的规模上具有人为的难度来操纵算法难度的阶。
>
> 除了最大匹配及其推广之外，还有很多类问题，它们有指数阶的算法且不太可能被改进……就实践目的而言，代数阶与指数阶之间的区别比有穷与无穷之间的区别更加关键。
>
> 标准有时是生硬的，然而由此阻止开发未知或已知不完全符合这些标准的算法却是令人遗憾的。当前已知的很多最佳的算法思想也墨守这种理论成规……然而，如果仅为推动寻找优秀实用的算法，则先问问这种算法是否存在从数学上看是明智的，认识到这一点很重要。这样一来，寻找算法这项任务就可以描述为具体的猜想。

1.7　定理 1.9 的证明：$O(T\log T)$时间的通用模拟

现在，我们证明定理 1.9，亦即构造一个通用图灵机$\mathcal{U}$，使得给定输入 x 和一个在 x 上运行至多 T 步后停机的图灵机 M，$\mathcal{U}$将在 $O(T\log T)$时间内输出 $M(x)$（其中隐藏在 O 记号中的常数依赖于被模拟图灵机 M 的参数）。

$\mathcal{U}$的一般结构如 1.4.1 节所述。$\mathcal{U}$以与 M 一样的方式使用输入带和输出带。$\mathcal{U}$用一条工作带对 M 的工作带上的内容进行编码，另用两条工作带分别存储 M 的转移函数和当前状态。需要解决的困难是，不能用论断 1.6 将 M 的工作带减为一条，因为论断 1.6 为模拟过程引入了过高的开销。因此，我们将给出另一种方法来将 M 的工作带上的内容编码后存储在$\mathcal{U}$的一条工作带上。我们称$\mathcal{U}$的这条工作带为主工作带。

令 k 是 M 所用的（除输入带和输出带之外的）带的条数，Γ 是它的字母表。根据论断 1.5 的证明过程，我们假定$\mathcal{U}$的字母表为 Γ^k（因为这使得模拟的代价只依赖于 k 和 $|\Gamma|$）。于是，$\mathcal{U}$的主工作带上的每个单元可以编码 Γ 中的 k 个符号，其中每个符号来自 M 的一条带。这意味着，$\mathcal{U}$的主工作带不再是一条简单的工作带，而是 k 条并行的带。换句话说，可以认为$\mathcal{U}$具有这样 k 条带，在每个步骤中这 k 条的带头作为整体一起向左、向右移动或保持不动。虽然我们可以很容易地将 M 的 k 条工作带上的内容编码存储在$\mathcal{U}$的 k 条并行带

⊖ 这里，"难度"指的是复杂性。——译者注

⊜ 这里，"代数增长"即多项式增长。——译者注

上，但仍需解决如下困难：M 的 k 个读写头独立地向左或向右移动而U的并行带只能一起移动。套用一句名言，我们采取的策略将是“如果带头不能将就带上的位置，就让带上的位置来将就带头”。

策略的含义是，既然我们不能让U的读写头朝不同方向移动，那么就让并行带在带头下面移动。为了模拟 M 的单个步骤，我们将所有并行带上的非空白符号向左、右移动，直到并行带上位于带头下的符号恰好对应 M 的各个带头处的符号。例如，如果 $k=3$ 并且在某个特定的步骤中 M 的转移函数要求各个带头分别向左、向右、向右移动，则U将第一条并行带上的所有非空白符全部向右移动一个单元，并将第二条和第三条并行带上的所有非空白符全部向左移动一个单元(参见图 1-8)。再利用一条额外的“草稿”工作带，U可很容易地实现上述移动。

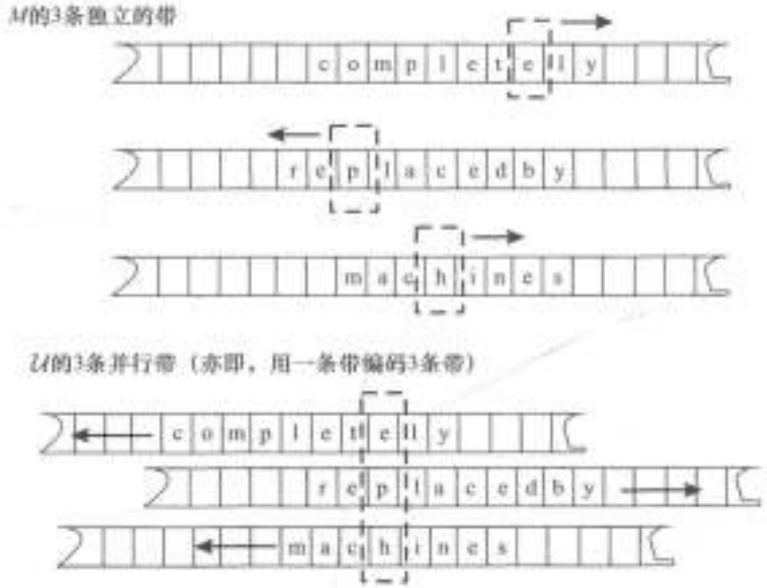


图 1-8　M 的 k 条带编码为U的一条带。我们将U的工作带看成由 k 条并行带构成。其带头整体移动，因此通过移动带上内容来模拟带头的独立移动

上述的方法仍不足以实现 $O(T\log T)$时间的通用模拟。这是由于，每条并行带上的非空白符有 T 个之多，因此模拟 M 的每个步骤时移动符号的操作就可能花费U的 T 个步骤，
29 这将使得整个模拟过程移动符号的总开销达到 $O(T^2)$时间。为了克服这个问题，我们将把各条带上的信息进行编码使得符号移动具有较低的平摊代价。我们将确保每次符号移动操作无需移动每条带上的所有非空白符。具体而言，我们对信息的编码将确保一半的符号移动操作需要用 $2c$ 个步骤，四分之一的符号移动操作需要用 $4c$ 个步骤；更一般地，2^{-i}的符号移动操作需要用 $2^i c$ 个步骤，其中 c 是一个常数。这样，整个模拟过程的开销大约为 $cT\log T$ 时间(参见后面的论述)。(上面这种分析称为平摊分析，它被广泛用于算法设计。)

将 M 的多条带编码为U的一条带

我们对信息编码的过程中将预留“缓冲区”以提高符号移动操作的效率。这就是说，不让U的每条并行带恰好对应 M 的一条带，而是要在U的每条并行带的字母表中添加一个特殊的空白符⊠，它的语义是要求模拟过程忽略它。例如，如果 M 的一条带上存储 010，则它编码在对应的U的并行带上可能不再是 010 本身，而是 0⊠10 或 0⊠⊠1⊠0，或者类似的其他串。

为方便计，假设U的并行带均向左和向右无限延伸(这可以用最小的代价进行模拟，参见论断 1.8)。因此，我们用 0，±1，±2…表示并行带上存储单元的位置。正常情况下，将U的带头固定在所有并行带的位置 0，只有在移动符号时才临时移动带头。正如方法概述中所说，我们会通过向右移动并行带来模拟带头左移，反之亦然；操作完成后，带头将重置到位置 0。

我们将U的所有并行带划分为一系列区域，依次记为 R_0，L_0，R_1，L_1，…(至多使用到区域 R_{i_0+1}，L_{i_0+1})位于位置 0 的存储单元不属于任何区域。区域 R_0包含位置 0 右侧 30
的两个存储单元(即位于位置+1，+2 的存储单元)，而区域 R_1包含位于位置+3，+4，+5，+6 的四个存储单元。一般而言，对任意 $i\geqslant 1$，区域 R_i包含区域 R_{i-1}右侧紧邻的 $2\cdot 2^i$ 个存储单元(即位置依次为 $2^{i+1}-1$，…，$2^{i+2}-2$ 的存储单元)。类似地，区域 L_0包含位置为−1，−2 的两个存储单元；且一般地，区域 L_i包含位置依次为 $-2^{i+2}+2$，…，$-2^{i+1}+1$ 的存储单元。我们还维持下列三个不变量：

- 每个区域中非⊠的符号要么是空的，要么是满的，要么是半满的。这就是说，区域 R_i中不等于⊠的符号的个数等于 0，2^i或 $2\cdot 2^i$ 个，同样的结论对 L_i也成立。(符号□当成正常符号处理。特别地，如果一个区域内全是符号□，则该区域是满的。)
  假定初始时所有区域均是半满的，通过在首次遇到一个区域时将其一半存储单元用⊠符号填充可以确保假设是合理的。
- $R_i\cup L_i$中非⊠的符号的总数是 $2\cdot 2^i$。这就是说，要么 R_i是空的而 L_i是满的，要么 R_i是满的而 L_i是空的，要么二者均是半满的。
- 位置 0 的元素始终不等于⊠。

符号移动

按照上述方式设置区域的优点是便于符号移动。事实上，我们不必移动整条并行带上的所有符号，而仅需移动部分区域中的符号。为证明这一点，考虑U在第一条并行带上如何向左移动符号(参见图 1-9)。

1. U找到最小的 i_0使得 A_{i_0}不是空的。注意，i_0同时也是使得 L_{i_0}不为满的最小整数。我们称数 i_0是本次符号移动的索引数。

2. 将区域 R_{i_0}最左边的非⊠的符号放到位置 0，将 R_{i_0}最左边的其余 2^{i_0-1} 个非⊠的符号左移到区域 R_0，R_1，…，R_{i_0}中使得每个区域恰好被填充一半。注意，R_0，R_1，…，R_{i_0-1} 31
中恰有这些空间来完成上述操作，因为这些区域原来是空的而且 $2^{i_0}-1=\sum_{j=0}^{i_0-1}2^j$。

3. U在位置 0 的左侧执行对称操作。亦即，令变量 j 从 i_0-1 递减到 0，U递归地从区域 L_j中移出 $2\cdot 2^j$ 个符号来填充区域 L_{j+1}的一半空间。最后，U将原来位于位置 0 的符号(根据 M 的转移函数恰当修改后)移到区域 L_0中。

4. 移动操作完成后，区域 R_0，L_0，R_1，L_1，…R_{i_0-1}，L_{i_0-1}均是半满的，R_{i_0}中非⊠的符号减少了 2^{i_0} 个，L_{i_0}中非⊠的符号增加了 2^{i_0} 个。因此，不变量仍然维持。

5. 执行符号移动操作的总代价与区域 R_0，L_0，…R_{i_0}，L_{i_0}的总大小成正比。亦即总共执行了 $O(\sum_{j=0}^{i_0}2\cdot 2^j)=O(2^{i_0})$ 个操作。

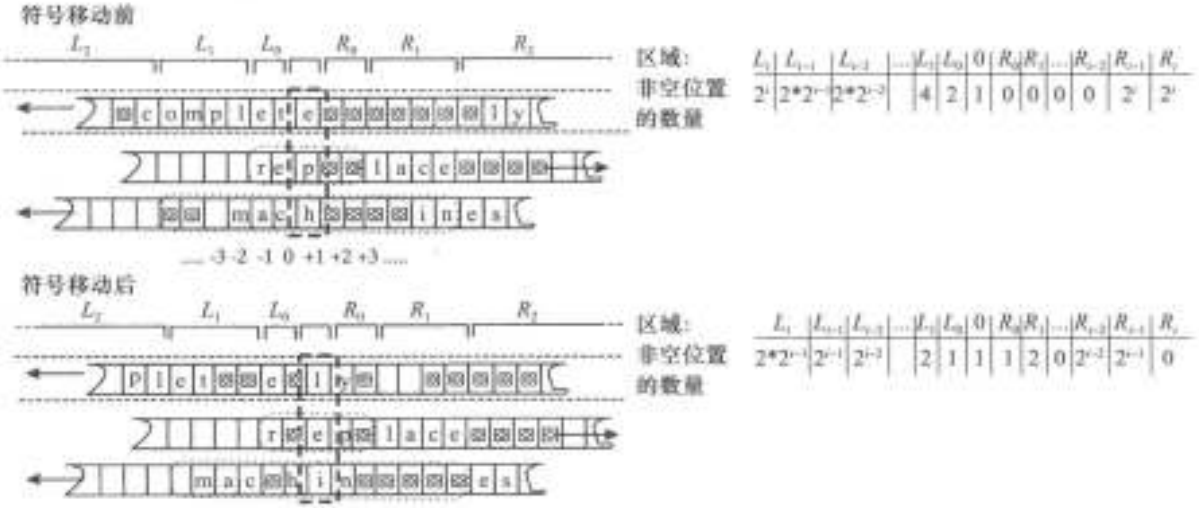


图 1-9　执行并行带移动。第一条带的左移涉及区域 R_0，L_0，R_1，L_1，R_2，L_2；第二条带的右移仅涉及区域 R_0，L_0；第三条带的左移涉及区域 R_0，L_0，R_1，L_1。我们维持不变量：每个区域要么是空的，要么是满的，要么是半满的；并且 $R_i \cup L_i$ 中非空单元的总数等于 $2 \cdot 2^i$。如果左移之前区域 L_0，…，L_{i-1} 原来是满的且 L_i 原来是半满的（而区域 R_0，…，R_{i-1} 原来是空的且 R_i 原来是半满的），则左移操作完成后 R_0，L_0，…，R_{i-1}，L_{i-1} 将是半满的，L_i 将是满的而 R_i 将是空的

在索引数为 i 时执行符号移动之后，R_0，L_0，…，R_{i-1}，L_{i-1} 均是半满的，这意味着在所有区域 L_0，…，L_{i-1} 全部变空之前至少还要执行 2^i-1 次符号左移操作，在所有 R_0，…，R_{i-1} 全部变空之前也至少要执行 2^i-1 次符号右移操作。无论何种情况，一旦在索引数 i 处执行了符号移动操作，则接下来在同一条并行带上执行的 2^i-1 次符号移动操作的索引数必然均小于 i。这说明，在每条并行带上，符号移动操作的总次数中至多 $1/2^i$ 的索引数等于 i。由于我们至多执行 T 次符号移动操作且可能的索引数最大为 $\log T$，因此 U 在模拟 M 的 T 个步骤的过程中移动并行带上的符号的总开销为

$$O\left(k \cdot \sum_{i=1}^{\log T} \frac{T}{2^{i-1}} 2^i\right) = O(T \log T) \qquad \blacksquare$$

本章学习内容

- 存在许多等价方法对计算过程进行数学建模。我们采用标准的图灵机定义。
- 图灵机可以表示为位串。存在通用图灵机，它可以（代价很小地）模拟任意给定的用位串表示的图灵机。
- 存在函数不能被任意图灵机计算，不管计算时间多长。停机问题是不可计算的。
- 类 **P** 是由可以被图灵机在多项式时间内求解的所有判定问题构成的。我们称 **P** 中的问题是可以被高效求解的。
- 图灵机定义中（带的数量、字母表大小等）底层结构的选取不是本质的，因为它们不会改变 **P** 的定义。

本章注记和历史

尽管算法的研究已持续千百年，并且一些计算装置也早在二十世纪以前就被发明出来（最值得关注的是十九世纪中叶由查尔斯·巴贝奇（Charles Babbage）发明的差分机和分析

机)。但确切地讲，现代计算机科学的基础直到二十世纪三十年代才真正奠定。

1931 年，库尔特·哥德尔证明了在自然数上肯定成立的命题是固有的不可证明的。这一结果震惊了整个数学界，同时粉碎了希尔伯特在 1900 年提出的旨在建立完备算术公理的雄心勃勃的研究计划。1936 年，阿隆佐·邱奇(Alonzo Church)定义了一种称为 λ-演算的计算模型(多年后由此产生了 LISP 这种程序设计语言)，并证明了该计算模型上存在固有的不可计算函数[Chu36]。几个月之后，阿兰·图灵独立地引入了他的图灵机，并证明了图灵机上也存在固有的不可计算函数[Tur36]。图灵还介绍了能加载任意程序的通用图灵机的思想。前述的两种计算模型已被证明是等价的。然而，正如邱奇自己所述，图灵机的优点在于"根据普通意义(而非显式的定义)就能立即明确地认定高效性"。文集[Dav65]收录了可计算理论方面的许多影响深远的开创性论文。西普赛尔(Sipser)的书[Sip96]的第二部分很好地对可计算理论作了一般性介绍，而[Rog87，HMU01，Koz97]等书则相对深入地对此进行了讨论。这些书也涵盖了自动机理论，它是计算理论的另一个领域，本书目前未做相关讨论。本书网站提供了可计算理论和自动机理论相关信息的链接。

二战期间，图灵设计了机械密码破译装置，它在破译德军的"谜"密码的过程中发挥了关键作用，同时该项成就也改变了战争的进程(参见传记[Hod83，Lea05])[⊖]。二战后，建造通用电子计算机的任务在英国和美国同时进行，建造过程的一个关键人物是约翰·冯·诺依曼。这位极其多产的科学家除参加了曼哈顿项目的整个过程之外，还发现了经济博弈论。截止到目前，所有的数字计算机本质上均采用了"冯·诺依曼体系结构"，这是由他在设计最早的数字计算机 EDVAC 的过程中提出的[vN45]。

随着计算机的日益盛行，计算效率问题成为中心问题。科巴姆(Cobham)[Cob64]定义了类 **P**，并指出这可能是高效计算的不错的形式刻画。埃德蒙兹(Edmonds)也曾在给出高效非平凡多项式时间算法来求解一般图上的最大匹配的字里行间中流露过类似的提法([Edm65]，见之前的引用)。哈特马尼斯(Hartmanis)和斯特恩斯(Stearns)[HS65]为任意函数 T 定义了类 **DTIME**($T(n)$)，并证明了本章所证的定理 1.9 的稍宽泛的形式(我们给出的定理是由亨尼(Hennie)和斯特恩斯给出的[HS66])。他们还创造了"计算复杂性"这个术语，并证明了一个有用的"加速定理"：如果 f 是图灵机 M 在 $T(n)$时间内计算的函数，则对于任意常数 $c \geqslant 1$，f 能够被另一个图灵机$\overline{M}$在$T(n)/c$时间内计算，其中$\overline{M}$的状态可能比 M 更多且字母表也可能比 M 更大。该加速定理也再次解释了 **DTIME**($T(n)$)的定义中常数因子可以被忽略的原因。布卢姆(Blum)[Blu67]给出了复杂性理论的另一套公理，其中没有明确提到图灵机。

我们略去了对一些"怪异条件"的讨论，这些条件只有在考虑"非时间可构造"这种时间界限时才可能遇到，特别是"超大的"时间界限(如 $T(n)$远高于 n 的指数时间)。例如，存在非时间可构造的函数 T：**N**→**N**，使得任意 $T(n)$时间内可计算函数均可以在远小于 $\log T(n)$的时间内被计算。尽管如此，本书不涉及任何非时间可构造函数。

PAL 问题在仅有一条读/写带的图灵机上需要 $\Omega(n^2)$个计算步骤才能求解，这一结果来自[Maa84]，也可参见习题 13.3。正如我们所讲，计算步骤少于 n 的算法是无意义的，因为这种算法甚至连输入都无法读入。上述结论对图灵机模型成立。然而，如果允许对输

⊖ 不幸的是，图灵活着的时候，他在战时获得的所有成就被严格保密。因此，他难逃英国法院的判决，必须通过注射荷尔蒙来"治疗"同性恋，这最终导致了他在 1954 年自杀。

入进行随机访问并采取随机操作步骤，则许多有意义的计算均可以在亚线性时间内完成。关于这方面的研究，请参阅[Fis04]。

习题

1.1　令 f 是加法函数，它将数对 x，y 的位串表示映射为数 $x+y$ 的位串表示。令 g 是乘法函数，它将 $\langle x, y\rangle$ 映射为 $\langle x \cdot y\rangle$。通过构造图灵机证明 f，g 均是可计算函数(要求给出图灵机的状态、字母表和转移函数)。

1.2　通过明确描述图灵机 M 完成论断 1.5 的证明。

1.3　完成论断 1.6 的证明。

1.4　完成论断 1.8 的证明。

1.5　如果图灵机 M 带头的移动不依赖于其输入而只依赖于输入的长度，则定义该图灵机是散漫的。亦即，对于任意输入 $x\in\{0,1\}^*$ 和 $i\in\mathbf{N}$，散漫图灵机 M 在输入 x 上执行第 i 步时它的每个带头的位置均仅是 $|x|$ 和 i 的函数。证明：对于任意时间可构造的 T：$\mathbf{N}\to\mathbf{N}$，如果 $L\in\mathbf{DTIME}(T(n))$，则存在散漫图灵机在 $O(T(n)^2)$ 时间内判定 L。并且，存在 2 带散漫图灵机满足上述要求：它的一条带是输入带，另一条带是工作带/输出带。

1.6　证明：对于任意时间可构造的 T：$\mathbf{N}\to\mathbf{N}$，如果 $L\in\mathbf{DTIME}(T(n))$，则存在散漫图灵机在 $O(T(n)\log T(n))$ 时间内判定 L。

1.7　将每条带均是无穷网格且每个带头不仅能左、右移动而且还能上、下移动的图灵机定义为 2 维图灵机。证明：对于任意(时间可构造)的 T：$\mathbf{N}\to\mathbf{N}$ 和布尔函数 f，如果 f 能被 2 维图灵机在 $T(n)$ 时间内计算，则 $f\in\mathbf{DTIME}(T(n)^2)$。

1.8　设 LOOKUP 如下定义：在输入对 $\langle x, i\rangle$ 上，其中 x 是一个二进制串而 i 是一个自然数，LOOKUP 输出 x 的第 i 个位，在 $|x|<i$ 时输出 0。证明：LOOKUP$\in\mathbf{P}$。

34

1.9　定义具有随机访问存储的图灵机为 RAM 图灵机，它可以进一步形式化如下：机器有一个全初始化为空白符的无穷数组 A。机器这样访问数组 A：机器的一条工作带被用作地址带，同时机器有两个特殊符号 R，W 以及一个额外的状态 $q_{\text{.....}}$。一旦机器进入状态 $q_{\text{.....}}$，如果地址带上的符号是 $\langle i\rangle$ R(其中 $\langle i\rangle$ 是 i 的二进制表示)，则值 $A[i]$ 被写到地址带上符号 R 后面的单元中。如果地址带上的符号是 $\langle i\rangle$ Wσ(其中 σ 是机器字母表中的符号)，则将 $A[i]$ 置为值 σ。

证明：如果布尔函数 f 可以被一个 RAM 图灵机在 $T(n)$ 时间内计算，其中 T 是一个时间可构造函数，则 f 属于 $\mathbf{DTIME}(T(n)^2)$。

1.10　考虑如下简单的编程语言。它有一个无穷数组 A 和一个整数变量 i，其中数组 A 只能存储符号 $\{0, 1, \square\}$ 且初始值均为 $\square$。该语言的每个程序由如下的一系列行构成：

$$label:\text{If } A[i] \text{ 等于 } \sigma \quad \text{then cmds}$$

其中 $\sigma\in\{0, 1, \square\}$，而 cmds 是下列命令中的一个或多个：(1)将 $A[i]$ 置为 $\tau\in\{0, 1, \square\}$；(2)转到 *label*；(3)将 i 增大 1；(4)将 i 减小 1；(5)输出 b 后停机，其中 $b\in\{0, 1, \square\}$。程序在输入 $x\in\{0,1\}^*$ 上运行时，先将 $A[i]$ 置为 x 的第 i 个位，然后根据显而易见的语义运行。

证明：对于任意函数 f：$\{0,1\}^*\to\{0,1\}$ 和(时间可构造的)T：$\mathbf{N}\to\mathbf{N}$，如果 f 可

以用上述编程语言在 $T(n)$时间内计算，则 $f\in\mathbf{DTIME}(T(n))$。

1.11　明确给出一个完整方案，使得它能将图灵机表示成二进制串。亦即，给出一个能将任意图灵机 M(例如，例 1.1 中描述的计算函数 PAL 的图灵机)转换成二进制串 $\langle M\rangle$ 的过程，它要保证能够从 $\langle M\rangle$ 还原图灵机 M，或者说至少能够还原得出一个功能等价的图灵机 $\tilde{M}$(即 $\tilde{M}$能够在相同运行时间内计算 M 能计算的函数)。

1.12　从$\{0,1\}^*$到$\{0,1\}$的一个部分函数是不必在所有输入上均有定义的一个函数。图灵机 M 计算一个部分函数 f 指的是，图灵机在任意 x 上运行时，如果 f 在 x 上有定义则输出 $f(x)$，如果 f 在 x 上没有定义则图灵机进入无限循环。如果$\mathcal{S}$是一个由部分函数构成的集合，则定义布尔函数 $f_{\mathcal{S}}$使得 $f_{\mathcal{S}}(\alpha)=1$ 当且仅当 M_α 计算$\mathcal{S}$中的一个部分函数。莱斯定理(Rice Theorem)定理断言，对于任意非平凡$\mathcal{S}$(既不是空集也不是由某个图灵机能够计算的所有部分函数构成的集合)，则 $f_{\mathcal{S}}$是不可计算的。

(a) 用莱斯定理证明 HALT 是不可计算的，由此得到定理 1.11 的另一个证明。

(b) 证明莱斯定理。

1.13　已知存在常数 C 使得对于任意 $i>C$，存在大于 i^3 小于 $(i+1)^3$ 的素数[Hoh30, Ing37]。对于任意 $i\in\mathbf{N}$，令 p_i 是介于$(i+C)^3$和$(i+C+1)^3$之间的最小素数。我们 35
称数 n 对位串 $x\in\{0,1\}^*$ 进行了编码，如果 p_i 整除 n 当且仅当 $x_i=1$ 对任意 $i\in\{1,\cdots,|x|\}$成立[⊖]。

(a) 利用 1.5.2 节引入的操作符，给出一个逻辑表达式 $\mathrm{BIT}(n,i)$使得 $\mathrm{BIT}(n,i)$为真当且仅当 p_i整除 n。

(b) 给出一个逻辑表达式 $\mathrm{COMPARE}(n,m,i,j)$使得该表达式为真当且仅当 n 编码的位串的第 i 个位置和 m 编码的位串的第 j 个位置取值相同。

(c) 图灵机的一个*格局*由所有带上的符号、带头位置和寄存器状态构成。亦即，图灵机的格局是它在执行输入的某个时刻上机器的所有信息。证明：图灵机的格局可以表示为二进制串(同论断 1.6 一样，可以假设图灵机只有一条带)。

(d) 对于图灵机 M 和输入 $x\in\{0,1\}^*$，给出一个表达式 $\mathrm{INIT}_{M,x}(n)$使其为真当且仅当 n 是 M 在 x 上的初始格局的编码。

(e) 为图灵机 M 给出一个表达式 $\mathrm{HALT}_M(n)$使其为真当且仅当 M 在 n 为其格局编码之后停机。

(f) 为图灵机 M 给出一个表达式 $\mathrm{NEXT}(n,m)$使其为真当且仅当 n，m 分别是 M 的两个格局 x，y 的编码且 y 是在格局 x 上执行 M 的一个计算步骤后所得的格局。

(g) 为图灵机 M 给出一个表达式 $\mathrm{VALID}_M(m,t)$使其为真当且仅 m 是 M 的 t 个格局 $x_1,\cdots,x_t$构成的元组且 x_{i+1}是在格局 x_i上执行 M 的一个计算步骤后所得的格局。

(h) 对于图灵机 M 和输入 $x\in\{0,1\}^*$，给出一个表达式 $\mathrm{HALT}_{M,x}(t)$使其为真当且仅当 M 在 x 上的计算在 t 个步骤内停机。

(i) 设 TRUE-EXP 是一个函数，它的输入是一个(前缀形式的)数论结论 φ(的位串表示)，如果 φ 为真则它输出 1，否则输出 0。证明：TRUE-EXP 是不可计

⊖ 从技术上讲，在此定义下一个数可以编码多个位串，这不是问题。尽管可以通过如下技术避免这种情况：先用映射 $0\mapsto 00$，$1\mapsto 11$ 将串 x 编码为长度等于 $2|x|$ 的串 y，再在 y 的末尾添加位序列 01。

算的。

1.14　证明图上的下列语言/判定问题属于 **P**。（你既可以用邻接矩阵表示图，也可以用邻接列表表示图。表示形式不影响结论，请说明为什么？）

(a) CONNECTED——所有连通图的集合。亦即，如果图 G 的任意一对顶点 u，v 均通过一条路径连通，则 $G\in$ CONNECTED。

(b) TRIANGLEFREE——不含三角形的图的集合。（三角形指的是相互连通的三个不同顶点。）

(c) BIPARTITE——所有二分图的集合。亦即，如果图 G 的顶点可以划分为两个集合 A，B 使得 G 的所有边均是从 A 中的一个顶点连接到 B 中的一个顶点（A 中任意两个顶点之间或 B 中任意两个顶点之间不存在边），则 $G\in$ BIPARTITE。

(d) TREE——所有树的集合。如果一个图是连通的并且没有环，则称该图是一棵树。等价地，如果图 G 中任意一对顶点 u，v 恰通过一条简单路径连通，则称 G 是一棵树，其中简单路径是指无重复顶点的路径。

1.15　注意，通常情况下将数表示为串时都以 2 为基数。这就是说，数 n 表示为满足 $n=\sum_{i=0}^{k} x_i 2^i$ 的序列 x_0，x_1，…$x_{\log n}$。然而，我们也可以用其他表示方法。如果 $n\in$ **N** 且 $b\geqslant 2$，则 n *以 b 为基数的表示*（记为 $\langle n\rangle_b$）可以如下获得：先将 n 表示为 $\{0,\cdots,b-1\}$ 中的一系列数位，然后将每个数位 $d\in\{0,\cdots,b-1\}$ 替换为它的二进制表示。n *的一元表示*（记为 $\langle n\rangle_{\text{unary}}$）指的是串 1^n（即重复的 n 个 1）。

(a) 证明：选用不同基数进行表示不会影响类 **P**。亦即，证明，对自然数的任意子集 S，如果定义 $L_S^b=\{\langle n\rangle_b: n\in S\}$，则对任意 $b\geqslant 2$ 均有 $L_S^b\in$ **P** 当且仅当 $L_S^2\in$ **P**。

(b) 通过证明下面的语言属于 **P** 来证明选用一元表示会影响类 **P**：

$$\text{UNARYFACTORING}=\{\langle\langle n\rangle_{\text{unary}},\langle \ell\rangle_{\text{unary}},\langle k\rangle_{\text{unary}}\rangle: \text{存在素数 } j\in(\ell,k) \text{ 整除 } n\}$$

如果采用二进制表示，则该语言是否属于 **P** 还是未知的（参见第 9 章和第 10 章）。在第 3 章中将看到一个问题，如果采用一元表示则可以证明该问题属于 **P**，如果采用二元表示则可以证明它不属于 **P**。

NP 和 NP 完全性

[如果 $b(n) \approx Kn^2$][⊖]，则将造成极其重要的影响。也就是说，这将清楚地表明，尽管[希尔伯特]判定问题无解，但数学家为判定问题所伤的脑筋将完全被机器取代……然而，在我看来，这似乎完全是可能的。

——库尔特·哥德尔给约翰·冯·诺依曼的信，1956

我猜想，旅行售货商问题不存在高效算法。我的理由同所有数学猜想一样：(1)它完全有可能成立；(2)我不知道它是否成立。

——杰克·埃德蒙兹，1966

本文将给出几个定理，明确指出，不是暗示，这些问题同许多其他问题一样，将永远是难解的。

——理查德·卡普，1972

如果你曾玩过填字游戏，那你一定知道从头开始游戏远比验证其他人给出的答案难得多。同样，自己动手做数学家庭作业远比阅读和理解老师给的答案难得多。区别在于，寻找填字游戏的答案和解数学题是创造性劳动。验证答案相对容易的原因是其他人已经完成了创造性劳动。

本章研究计算过程中类似的现象。在 2.1 节中，我们定义类 **NP**，它旨在刻画解可以被高效地验证的所有问题。相比之下，前一章定义的类 **P** 则包含了所有可以被高效求解的判定问题。著名的 **P** ≠ **NP** 问题是问这两个复杂性类是否相同。

2.2 节将引入 **NP**-完全问题这一重要概念。**NP**-完全问题确切地说就是 **NP** 中最难的问题。目前，实践中已发现的 **NP**-完全问题已多达数千个。这些问题中的任何一个存在多项式算法当且仅当 **P**=**NP**。**NP** 完全性的研究要用到归约这一基础概念，它用以将不同的计算复杂性问题关联在一起。归约概念和它的各种姊妹概念在后续章节(比如第 7 章、第 17 章和第 18 章)中经常用到。本章提出的概念将用于后续许多其他章节。

P=**NP** 蕴含着难以置信的结果。如前所述，**NP** 问题似乎刻画了问题求解时的“创造性”过程，如果 **P**=**NP**，则这样的创造性过程可以由计算机完成。例如，在这种情况下，计算机将能够找出任意一个数学真结论的证明，只要这样的证明存在。我们将在 2.7.3 节纵览上述“**P**=**NP** 的乌托邦”。解决 **P** ≠ **NP** 问题确实具有重要的实践意义、科学意义和哲学意义。

2.1 类 NP

现在，我们定义复杂性类 **NP**，由此将“可被高效验证的解”这一直觉概念形式化。在第 1 章，我们曾说一个问题是“可被高效求解的”，如果它可以被图灵机在多项式时间内求解。于是，问题的解是“可被高效验证的”，指的是该问题的解可以在多项式时间内得到验

⊖ 用现代术语说，如果 SAT 存在二次时间算法。

证。由于图灵机每步只能读一个位，因此这意味着给定的解不能太长——至多是输入长度的多项式。

定义 2.1（类 NP）　语言 $L\subseteq\{0,1\}^*$ 属于 **NP**，如果存在多项式 $p:\mathbf{N}\to\mathbf{N}$ 和一个多项式时间图灵机 M（称为 L 的验证器），使得对于任意 $x\in\{0,1\}^*$ 有

$$x\in L\Leftrightarrow\exists u\in\{0,1\}^{p(|x|)}\text{ 满足 }M(x,u)=1$$

如果 $x\in L$ 和 $u\in\{0,1\}^{p(|x|)}$ 满足 $M(x,u)=1$，则称 u 是 x（关于语言 L 和图灵机 M）的证明。

有些教科书使用“见证”（witness）这一术语而不用术语“证明”（certificate）。显然，$\mathbf{P}\subseteq\mathbf{NP}$，因为多项式 $p(x)$ 可以等于 0（换句话说，u 可以是空串）。

例 2.2（INDSET$\in$**NP**）　为理解定义，我们证明例 0.1 中定义的（关于“你能举行的最大宴会的”）语言 INDSET 属于 **NP**。回顾该语言包含所有序对 $\langle G,k\rangle$，其含义是图 G 含有至少 k 个顶点的子图使得该子图中所有顶点之间没有边（这样的子图称为独立集）。考虑下面的多项式时间算法 M。给定序对 $\langle G,k\rangle$ 和位串 $u\in\{0,1\}^*$，算法输出 1 当且仅当 u 是 G 的任意两个顶点间没有边的 k 个顶点的编码。显然，$\langle G,k\rangle\in$INDSET 当且仅当存在位串 u 使得 $M(\langle G,k\rangle,u)=1$，进而 INDSET$\in$**NP**。$u$ 中 k 个顶点构成的 G 的独立集是 $\langle G,k\rangle$ 属于 INDSET 的证明。注意，如果 n 是 G 中顶点的个数，则 k 个顶点可以用 $O(k\log n)$ 个位进行编码。因此，位串 u 的长度至多为 $O(n\log n)$，它是 G 的二进制表示的长度的多项式。◀

39

例 2.3　下面的几个判定问题也属于 **NP**（参见习题 2.2）。

旅行售货商问题：给定 n 个结点的集合，$\binom{n}{2}$ 个任意结点对之间的距离 $d_{i,j}$ 以及一个数 k，判定是否存在一个封闭回路（即“商人的售货路线”）使得访问每个结点恰好一次且代价不超过 k。证明是这种回路中所有顶点的序列。

子集和问题：给定数集 $\{A_1,\cdots,A_n\}$ 和数 T，判定是否存在一个和等于 T 的子集。证明是这种子集中所有数的列举。

线性规划问题：给定 n 个变量 $u_1,\cdots,u_n$ 上的 m 个有理系数线性不等式（每个线性不等式形如 $a_1u_1+\cdots+a_nu_n\leqslant b$，其中 $a_1,\cdots,a_n,b$ 是系数），判定是否存在变量 $u_1,\cdots,u_n$ 的有理数赋值满足所有不等式。证明是这样一个赋值（参见习题 2.4）。

0/1 整数规划问题：给定 n 个变量 $u_1,\cdots,u_n$ 上的 m 个有理系数线性不等式，判定能否将变量 $u_1,\cdots,u_n$ 赋值为 0 或 1 使其满足所有不等式。证明是这样一个赋值。

图同构问题：给定两个 $n\times n$ 的邻接矩阵 M_1，M_2，判定在顶点重新命名的意义下 M_1 和 M_2 是否定义了同一个图。证明是如下的一个置换 $\pi:[n]\to[n]$，它使得将 M_1 的行和列按 π 进行调整后 M_1 等于 M_2。

合数问题：给定一个整数 N，判定 N 是否是一个合数（不是素数的数）。证明是 N 的一个因数分解。

因数问题：给定三个整数 N、L、U，判定 N 是否有一个素因数 p 属于区间 $[L,U]$。证明是这样的素因数 p[⊖]。

图连通性问题：给定图 G 和 G 中的两个顶点 s、t，判定 s 和 t 在 G 中是否连通。证明

⊖　存在验证素性的多项式时间算法[AKS04]。我们也可以采用习题 2.5 的素性证明来说明**因数分解**问题属于 **NP**。

是 G 中从 s 到 t 的一条路径。

在上面列举的问题中，**图连通性**问题、**合数**问题和**线性规划**问题均属于 **P**。图连通问题属于 **P**，这是由于简单而著名的广度优先搜索算法（参见任何一本算法教材，例如[KT06，CLRS01]）。合数问题属于 **P** 是最近才知道的([AKS04]给出了一个漂亮的算法)。线性规划问题属于 **P** 也是一个不平凡的结果，它是由哈奇扬（Khachiyan）的椭球算法[Kha79]得出的结论。

还没有结果表明列表中其他问题属于 **P**，尽管也无法证明它们不属于 **P**。但是，正如我们将在 2.2 节中看到的那样，**独立集**问题(INDSET)、**旅行售货商**问题、**子集和**问题和 **0/1 整数规划**问题均被证明是 **NP**-完全的，这意味着它们不属于 **P**，除非 **P**=**NP**。对于**图同构**问题和**因数**问题，既没有结果表明它们属于 **P**，也没有结果表明它们是 **NP**-完全的。 ◀ 40

2.1.1　P 和 NP 的关系

类 **NP** 与第 1 章引入的类 **P** 和类 **DTIME**($T(n)$)(见定义 1.12 和 1.13)有如下平凡的关系。

论断 2.4　设 $\mathbf{EXP}=\bigcup_{c>1}\mathbf{DTIME}(2^{n^c})$，则 $\mathbf{P}\subseteq\mathbf{NP}\subseteq\mathbf{EXP}$。

证明　(**P**⊆**NP**)设语言 $L\in\mathbf{P}$ 被多项式时间图灵机 N 判定，则可以将 N 视为定义 2.1 中的机器 M，将零多项式视为 $p(x)$(换句话说，u 是空串)，于是 $L\in\mathbf{NP}$。

(**NP**⊆**EXP**)：如果 $L\in\mathbf{NP}$，且 M，$p()$分别是定义 2.1 中的机器和多项式，为了在 $2^{O(p(n))}$ 时间内判定 L，可以依次枚举每个可能的位串 u，再用 M 验证 u 是不是输入 x 的一个有效证明；机器 M 接受当且仅当 u 是 x 的证明。由于 $p(n)=O(n^c)$对某个常数 $c>1$ 成立，因此 u 的取法共有 $2^{O(n^c)}$ 种，因此机器 M 的运行时间也是类似的指数形式。 ■

除了论断 2.4 中平凡的关系之外，人们目前还不了解类 **NP** 与类 **DTIME**($T(n)$)之间是否存在其他更强的关系。**P**=**NP** 是否成立这一问题是复杂性理论有待解决的核心问题，同时也是一个重要的数学问题和具体的科学问题(见 2.7 节)。绝大多数研究者相信 **P**≠**NP**，这是由于经过多年尝试至今仍未找到任何 **NP**-完全问题的有效算法。

2.1.2　非确定型图灵机

非确定型图灵机有时缩写为 NDTM，它是图灵机的一种变形，类 **NP** 也可以用非确定型图灵机来定义。事实上，这种定义才是 **NP** 的原始定义，也是该类名字的来源，亦即 **NP** 表示非确定型多项式时间。非确定型图灵机与标准的图灵机(见定义 1.2)之间的区别仅在于，非确定型图灵机有两个转移函数 δ_0，δ_1和一个特殊状态 q_{accept}。用非确定型图灵机 M 计算函数时，M 在每个步骤中均可以任意选用两个转移函数之一加以应用。对于任意输入 x，如果存在转移函数的一个选用序列使得 M 在输入 x 上进入状态 q_{accept}，则$M(x)=1$；(相应的转移函数选用序列称为 M 的非确定型选择序列。)否则，(M 在转移函数的任意选用序列上均不进入状态 q_{accept})$M(x)=0$。对于任意输入 $x\in\{0,1\}^*$ 和任意非确定型选择序列，如果 M 在 $T(|x|)$ 个步骤内要么停机要么进入状态 q_{accept}，则称 M 的运行时间为 $T(n)$。

定义 2.5　对任意函数 T：**N**→**N** 和 $L\subseteq\{0,1\}^*$，如果存在常数 $c>0$ 和一个运行时间为 $c\cdot T(n)$的非确定型图灵机 M 使得 $x\in L\Leftrightarrow M(x)=1$ 对任意 $x\in\{0,1\}^*$ 成立，则称 $L\in\mathbf{NTIME}(T(n))$。

下面的定理表明，**NP** 是非确定型图灵机在多项式时间内计算的所有语言的集合，这

就得到了 NP 的另一个定义。

定理 2.6 $\mathbf{NP}=\bigcup_{c\in\mathbf{N}}\mathbf{NTIME}(n^c)$。

41 **证明**　基本思想如下。非确定型图灵机在接受计算前所选择的非确定型序列可以看作是该输入属于一个语言的证明；反之亦然。

设 p：$\mathbf{N}\to\mathbf{N}$ 是多项式，L 是 $p(n)$时间的非确定型图灵机 N 判定的语言。对任意 $x\in L$，存在 N 的一个非确定型选择序列使得 N 在 x 上进入状态 q_{accept}。该选择序列可以用作 x 的一个证明。该证明的长度为 $p(|x|)$，并且可以用一个确定型图灵机在多项式时间内验证，而该确定型图灵机模拟 N 的非确定选择序列并验证 N 在这个非确定型选择序列上是否进入状态 q_{accept}。根据定义 2.1，$L\in\mathbf{NP}$。

反之，根据定义 2.1 设 $L\in\mathbf{NP}$，于是我们构造一个多项式时间的非确定型图灵机 N 来判定 L。在输入 x 上，N 利用它能进行非确定型选择的能力写出一个长为 $p(|x|)$的位串 u(具体地说，这一个过程可如下完成。如果机器选择转移函数 δ_0则在带上写 0，如果选择 δ_1则在带上写 1)。然后，N 运行定义 2.1 中的确定型验证器图灵机，验证 u 是否为 x 的有效证明。如果是，则 N 进入状态 q_{accept}。显然，N 在 x 上进入状态 q_{accept} 当且仅当存在 x 的有效证明。由于 $p(n)=O(n^c)$对某个 c 成立，故 $L\in\mathbf{NTIME}(n^c)$。 ■

与确定型图灵机一样，非确定型图灵机也很容易表示成位串，且存在通用非确定型图灵机(见习题 2.6)。事实上，非确定性还可以使通用图灵机模拟输入的效率稍有提高。

需要注意的是，与标准图灵机不同，非确定型图灵机并不能作为任何可物理实现的计算设备的模型。

2.2　归约和 NP 完全性

人们已经证明，独立集问题至少与 NP 中其他任意一个问题一样难。也就是说，如果独立集问题存在多项式时间算法，则 NP 中所有问题均存在多项式时间算法。这种别具魅力的性质称为 NP 难性质。由于绝大多数科学家认为 $\mathbf{NP}\neq\mathbf{P}$，因此一个语言是 NP-难的将被视为该语言不能在多项式时间内判定的有力证据。

我们怎样才能证明一个语言 C 至少与另外一个语言 B 一样难呢？归约的概念是我们完成这种任务的关键工具。

定义 2.7　(归约，NP-难和 NP-完全性)语言 $L\subseteq\{0,1\}^*$ 多项式时间卡普归约到语言 $L'\subseteq\{0,1\}^*$，记为 $L\leqslant_p L'$，如果存在多项式时间可计算的函数 $f:\{0,1\}^*\to\{0,1\}^*$ 使得对于任意 $x\in\{0,1\}^*$ 均有 $x\in L$ 当且仅当 $f(x)\in L'$。多项式时间卡普归约也可以简称为多项式时间归约。

如果 $L\leqslant_p L'$对任意 $L\in\mathbf{NP}$ 成立，则称 L'是 NP-难的。如果 L'是 NP-难的且 $L'\in\mathbf{NP}$，则称 L'是 NP-完全的。

"多项式时间卡普归约"在有的教科书中也称作"多到一归约"或"多项式时间映射
42 归约"。

容易验证，多项式时间归约有一个重要性质，亦即，如果 $L\leqslant_p L'$且 $L'\in\mathbf{P}$，则 $L\in\mathbf{P}$(参见图 2-1)。这就是为什么在定义 L'至少与 L 一样难时只考虑多项式时间算法的原因。注意，$\leqslant_p$定义了语言间的关系。定理 2.8 的第一部分表明，该关系是传递的。后续章节将定义其他归约概念，其中很多也满足传递性。定理 2.8 的第二部分揭示了采用"NP-难"这个术语的原因，即 NP-难的语言至少与 NP 中其他任意一个语言一样难。类似地，第三部分揭示了采

用"NP-完全"这个术语的原因：为了研究 **P** ≠ **NP** 问题，只需研究任意一个 **NP**-完全问题是否可以在多项式时间内判定。

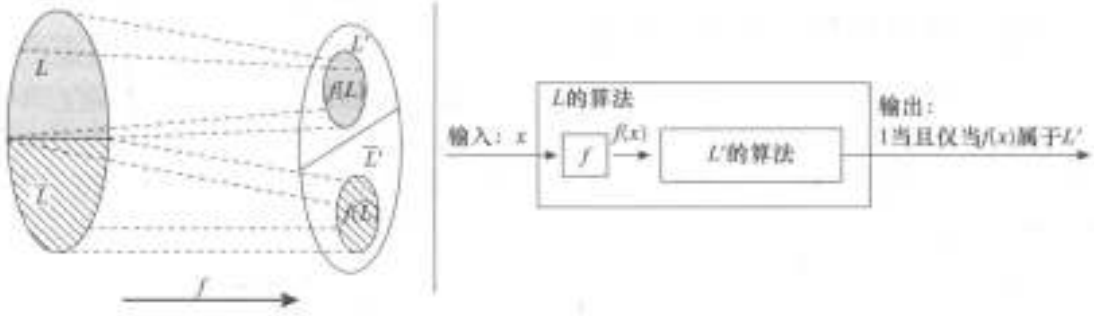


图 2-1　从 L 到 L' 的卡普归约是一个多项式时间函数 f，它将 L 中的串映射为 L' 中的串，并将 $\overline{L}=\{0,1\}^*$ 中的串映射为 $\overline{L'}$ 中的串。卡普归约 f 可以用来将判定 L' 的图灵机 M' 在多项式时间内转换成判定 L 的图灵机 M，使得 $M(x)=M'(f(x))$

定理 2.8

1. (传递性)如果 $L\leqslant_p L'$ 且 $L'\leqslant_p L''$，则 $L\leqslant_p L''$。
2. 如果语言 L 是 **NP**-难的且 $L\in\mathbf{P}$，则 $\mathbf{P}=\mathbf{NP}$。
3. 如果语言 L 是 **NP**-完全的，则 $L\in\mathbf{P}$ 当且仅当 $\mathbf{P}=\mathbf{NP}$。

证明　定理的三个部分均依赖于如下简单的观察结果：如果 p、q 分别是至多按 n^c 和 n^d 增长的多项式，则二者的复合 $p(q(n))$ 也是至多按 n^{cd} 增长的多项式。现在，我们证明 1，把其他两个部分的证明留作简单习题。

如果 f_1 是从 L 到 L' 的归约而 f_2 是从 L' 到 L'' 的归约，则映射 $x\mapsto f_2(f_1(x))$ 是从 L 到 L'' 的多项式时间归约，这是由于在给定的 x 上 $f_2(f_1(x))$ 在多项式时间内完成计算。最后，$f_2(f_1(x))\in L''$ 当且仅当 $f_1(x)\in L'$，它成立当且仅当 $x\in L$。 ■

NP-完全语言真的存在吗？换句话说，类 **NP** 中真有一个语言与该类中其他任意语言一样难吗？下面给出了这种语言的一个简单例子。

定理 2.9　下面的语言是 **NP**-完全的：

$$\mathrm{TMSAT}=\{\langle\alpha,x,1^n,1^t\rangle:\exists u\in\{0,1\}^*\text{ 使得 }M_\alpha\text{ 以}\langle x,u\rangle\text{ 为输入时将在 }t\text{ 步内输出 }1\},$$

其中 M_α 是位串 α 表示的(确定型)图灵机[⊖]。

证明　一旦理解了 **NP** 的内涵，便可直接得出定理 2.9 的证明。设 L 是一个 **NP** 语言。根据定义 2.1，存在多项式 p 和一个验证器图灵机 M 使得 $x\in L$ 当且仅当存在位串 $u\in\{0,1\}^{p(|x|)}$ 满足 $M(x,u)=1$ 且 M 的运行时间是多项式时间 $q(n)$。为将 L 归约到 TMSAT，只需将任意位串 $x\in\{0,1\}^*$ 映射到元组 $\langle\lfloor M\rfloor,x,1^{p(|x|)},1^{q(m)}\rangle$，其中 $m=|x|+p(|x|)$，$\lfloor M\rfloor$ 是图灵机 M 的位串表示。根据 TMSAT 的定义和 M 的选择，该映射显然可以在多项式时间内完成，并且

$$\langle\lfloor M\rfloor,x,1^{p(|x|)},1^{q(m)}\rangle\in\mathrm{TMSAT}\Leftrightarrow$$

$$\exists_{u\in\{0,1\}^{p(|x|)}}\text{ 使得 }M(x,u)\text{ 在 }q(m)\text{ 步内输出 }1\Leftrightarrow x\in L$$ ■

TMSAT 不是一个有用的 **NP**-完全问题，因为该语言的定义与图灵机概念的联系过于

43

⊖ 记住，1^k 表示一个长度为 k 的位串，其中每个位均是 1。在复杂性理论中，我们通常将 1^k 作为输入的一部分，以确保多项式图灵机的运行时间是 k 的多项式。

密切。因此，TMSAT 是 NP-完全问题的事实不能加深我们对 NP 的理解。在 2.3 节中，我们将给出更"自然"的一些 NP-完全问题的例子。

2.3　库克-勒维定理：计算的局部性

1971 年前后，库克(Cook)和勒维(Levin)各自独立地发现了 NP 完全性的概念并给出了一些 NP-完全的组合问题的例子。这些 NP-完全问题的定义看上去与图灵机似乎毫无关系。很快，卡普证明了 NP-完全性是广泛存在的，并且许多有实践意义的问题都是 NP-完全的。如今，各个分支学科中数以千计的计算问题已经被证明是 NP-完全的。

2.3.1　布尔公式、合取范式和 SAT 问题

最简单的 NP-完全问题来自命题逻辑领域。变量 u_1，…，u_n 上的一个布尔公式由变量和逻辑运算符与(AND)($\wedge$)、或(OR)($\vee$)、非(NOT)($\neg$)构成。例如，$(u_1 \wedge u_2) \vee (u_2 \wedge u_3) \vee (u_3 \wedge u_1)$ 是一个布尔公式。如果 φ 是变量 u_1，…，u_n 上的一个布尔公式且 $z \in \{0,1\}^n$，则 $\varphi(z)$ 表示将 φ 的变量依次赋予 z 中各个值之后 φ 的取值，z 中的 1 表示真而 0 表示假。如果存在赋值 z 使得 $\varphi(z)$ 为真，则称 φ 是*可满足的*，否则称 φ 是*不可满足的*。

前面给出的公式 $(u_1 \wedge u_2) \vee (u_2 \wedge u_3) \vee (u_3 \wedge u_1)$ 是可满足的，因为赋值 $u_1=1$，$u_2=0$，$u_3=1$ 满足该公式。一般而言，赋值 $u_1=z_1$，$u_2=z_2$，$u_3=z_3$ 满足该公式当且仅当至少有两个 z_i 是 1。

如果变量 u_1，…，u_n 上的布尔公式是在变量(或变量的否定)上 OR 操作所得的若干个公式上的 AND 操作，则称该公式是*合取范式*(Conjunctive Normal Form，CNF)。例如，下面的公式是一个 3CNF(这里 $\bar{u}_i$ 表示 $\neg u_i$，其他地方含义相同)

$$(u_1 \vee \bar{u}_2 \vee u_3) \wedge (u_2 \vee \bar{u}_3 \vee u_4) \wedge (\bar{u}_1 \vee u_3 \vee \bar{u}_4)$$

44

更一般地，合取范式形如

$$\bigwedge_i \left(\bigvee_j v_{ij} \right)$$

其中每个 v_{ij} 要么是一个变量 u_k，要么是变量的否定 $\bar{u}_k$，项 v_{ij} 称为公式的*文字*而项 $\bigvee_j v_{ij}$ 称为公式的*子句*。如果一个公式中每个子句至多包含 k 个文字，则称该公式为一个 *k 合取范式*或 *kCNF*。所有可满足的 CNF 公式构成的语言记为 SAT，所有可满足的 3CNF 公式构成的语言记为 3SAT。[⊖]

2.3.2　库克-勒维定理

下面的定理给出了第一个自然的 NP-完全问题。

定理 2.10　(*库克-勒维定理*[Coo71，Lev73])

1. SAT 问题是 NP-完全问题；

2. 3SAT 问题是 NP-完全问题。

我们现在证明定理 2.10(另一个用到布尔线路的证明将在 6.1 节中给出)。SAT 问题和 3SAT 问题显然属于 NP，因为满足布尔公式的任意一个赋值均是该公式可被满足的证

⊖　严格地说，表示布尔公式的字符串必须具有良好的结构。例如，字符串 $u_1 \wedge \wedge u_2$ 不表示任何有效的公式。通常，我们忽略这样的问题，因为字符串是否具有良好的结构很容易识别，而这种不具有良好结构的所有字符串可以视为同一个固定的恒假公式。

明。因此，仅需证明这两个问题均是 **NP**-难的。为此，我们(a)先证明 SAT 问题是 **NP**-难的；(b)然后将 SAT 问题多项式时间卡普归约到 3SAT 问题。由多项式时间归约过程的传递性可知，3SAT 问题是 **NP**-难的。(a)可以由下面的引理得到。

引理 2.11 SAT 问题是 **NP** 难的。

为证明引理 2.11，我们要给出任意 **NP** 语言 L 到 SAT 的归约过程。换句话说，需要一个多项式时间变换将任意 $x\in\{0,1\}^*$ 转换成合取范式 φ_x 使得 $x\in L$ 当且仅当 φ_x 是可满足的。由于除了 L 属于 **NP** 之外，我们对 L 一无所知，因此，归约过程只能依赖于计算的定义并将它表示成布尔公式。

2.3.3　准备工作：布尔公式的表达能力

作为证明定理 2.11 的准备工作，我们展示如何将各种条件表示成合取范式。

例 2.12（表达位串相等）　公式 $(x_i\vee\overline{y_i})\wedge(\overline{x_i}\vee y_i)$ 是合取范式，它只能被 x_i，y_i 取值相等的赋值满足。于是，公式

$$(x_1\vee\overline{y_1})\wedge(\overline{x_1}\vee y_1)\wedge\cdots\wedge(x_n\vee\overline{y_n})\wedge(\overline{x_n}\vee y_n)$$

被一个赋值满足当且仅当在该赋值中所有 x_i 和 y_i 取值相同。

因此，尽管＝不是像∧或∨这样的标准布尔操作符，但是我们却可以将它方便地用作一种缩写方法。这是由于，公式 $\phi_1=\phi_2$ 等价于公式 $(\phi_1\vee\overline{\phi_2})\wedge(\overline{\phi_1}\vee\phi_2)$，其中等价的意思是在相同的赋值上得到满足。◀

事实上，具有指数大小的 CNF 公式可以表达任意布尔函数，下面的简单论断证明了上述事实。

论断 2.13（与、或、非的通用性）　对任意布尔函数 $f:\{0,1\}^\ell\to\{0,1\}$，存在 ℓ 个变量上的大小为 $\ell 2^\ell$ 的 CNF 公式 φ 使得 $\varphi(u)=f(u)$ 对于任意 $u\in\{0,1\}^\ell$ 成立，其中 CNF 公式的规模定义为其中包含的符号∧和∨的个数。

证明概要　对于任意 $v\in\{0,1\}^\ell$，不难发现，存在 ℓ 个变量上的子句 $C_v(z_1,z_2,\cdots,z_\ell)$ 使得 $C_v(v)=0$ 并且 $C_v(u)=1$ 对任意 $v\neq u$ 成立。例如，如果 $v=(1,1,0,1)$，则相应的子句为 $\overline{z_1}\vee\overline{z_2}\vee z_3\vee\overline{z_4}$。

令 φ 是使 $f(v)=0$ 的所有 v 对应的子句 C_v 的与(AND)。亦即，

$$\varphi=\bigwedge_{v:f(v)=0}C_v(z_1,z_2,\cdots,z_\ell)$$

注意，φ 的大小至多为 $\ell 2^\ell$。对于满足 $f(u)=0$ 的任意 u 而言，有 $C_u(u)=0$，进而 $\varphi(u)$ 也等于 0。另一方面，如果 $f(u)=1$，则 $C_v(u)=1$ 对满足 $f(v)=0$ 的任意 v 成立，进而 $\varphi(u)=1$。于是，$\varphi(u)=f(u)$ 对于任意 u 成立。■

在本章中，论断 2.13 被使用时变量个数是固定不变的。

2.3.4　引理 2.11 的证明

设 L 是一个 **NP** 语言。由定义可知，存在多项式时间图灵机 M 使得对于任意 $x\in\{0,1\}^*$，$x\in L\Leftrightarrow M(x,u)=1$ 对某个 $u\in\{0,1\}^{p(|x|)}$ 成立，其中 $p:\mathbf{N}\to\mathbf{N}$ 是一个多项式。下面证明 L 可以多项式时间卡普归约到 SAT，具体的办法是构造多项式时间变换 $x\to\varphi_x$ 将位串 x 映射为一个 CNF 公式 φ_x 使得 $x\in L$ 当且仅当 φ_x 是可满足的。等价地，

$$\varphi_x\in\mathrm{SAT}\quad\text{当且仅当}\quad\exists u\in\{0,1\}^{p(|x|)}\text{ 满足 }M(x,u)=1\tag{2.1}$$

46 其中 · 表示字符串的拼接。[⊖]

公式 ϕ_x 如何构造呢？一个平凡的想法是，将论断 2.13 的应用到将 $u\in\{0,1\}^{p(|x|)}$ 映射为 $M(x,u)$ 的这个布尔函数上。这将得到 CNF 公式 ϕ_x 使得 $\phi_x(u)=M(x,u)$ 对任意 $u\in\{0,1\}^{p(|x|)}$ 成立。于是，满足 $M(x,u)=1$ 的位串 u 存在当且仅当 ϕ_x 是可满足的。但是，这个平凡的想法不行，因为由断言 2.13 所得公式 ϕ_x 的大小可能达到 $p(|x|)2^{p(|x|)}$。为得到更小的公式，我们要用到如下两个事实。其一，M 的运行时间是多项式。其二，图灵机的每个基本步骤具有高度的局部性(亦即图灵机只检查和修改它的带上的很少几个位)。更小的布尔公式可以用来描述这些具有局部性的操作步骤。

证明过程中，我们对图灵机 M 作如下几个简单的假设：(i)M 只有两条带，一条是输入带，另一条是工作/输出带；(ii)M 是散漫图灵机，亦即其带头的移动不依赖于带上内容。这就是说，M 在长度为 n 的所有输入上的计算时间是相同的，并且对于每个 i 值，M 在第 i 个步骤时带头的位置仅依赖于 i 和输入的长度。

上述假设不失一般性，这是由于，对于任意一个 $T(n)$ 时间图灵机 M，存在 $O(T(n)^2)$ 时间内计算同一函数的 2 带散漫图灵机 $\bar{M}$(参见评注 1.7 和习题 1.5)。[⊜]因此，特别地，如果 L 属于 **NP**，则存在 2 带的多项式时间散漫图灵机 M 和多项式 p 使得

$$x\in L\Leftrightarrow\exists u\in\{0,1\}^{p(|x|)}\text{ 满足 }M(x\cdot u)=1 \tag{2.2}$$

注意，由于 M 是散漫图灵机，我们可以让 M 在平凡输入 $(x,0^{p(|x|)})$ 上运行，以确定 M 在具有相同长度的其他输入上进行计算时带头的确切位置。我们后面将用到这一事实。

将 M 所有可能的状态构成的集合记为 Q，字母表记为 Γ。M 在输入 y 上运行到第 i 个步骤的快照是一个三元组 $\langle a,b,q\rangle\in\Gamma\times\Gamma\times Q$，其中 a、b 是 M 在第 i 步时带头在两条带上分别读到的符号，而 q 是 M 在第 i 步时所处的状态(参见图 2-2)。显然，M 的快照可以编码为二进制位串。令 c 是该位串的长度，它是依赖于 $|Q|$ 和 $|\Gamma|$ 的一个常数。

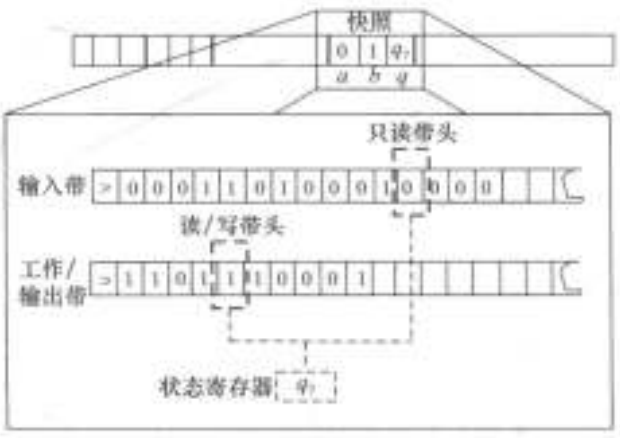


图 2-2　图灵机在特定步骤上的快照包含当前状态和图灵机读到的符号。如果机器在第 i 步时读到的符号分别是 0，1 且处于状态 q_7，则第 i 步的快照是元组 $\langle 0,1,q_7\rangle$

对于任意 $y\in\{0,1\}^*$，M 在输入 y 上运行到第 i 步时的快照依赖于(a)机器在第 $i-1$ 步时的状态；(b)输入带和工作带上当前存储单元内的符号。

证明的核心思路如下。假设某人声称存在 u 使得 $M(x\cdot u)=1$，并将 M 在 $x\cdot u$ 上的运行过程中出现的快照序列提供给你作为证明。你如何判断该快照序列确实构成了一个由 M 完成的有效的计算呢？

⊖ 由于输入的第二项 u 的长度 $p(|x|)$ 很容易计算，因而可以将序对 (x,u) 简单地表示为 $x\cdot u$，而无需在 x 和 u 之间使用"分隔符"。

⊜ 事实上，如果小心地加以设计，则每个 $T(n)$ 时间非散漫的图灵机可以被一个散漫图灵机在 $O(T(n)\log T(n))$ 时间内模拟，参见习题 1.6。散漫图灵机可以用两条以上的带，但接下来的证明很容易推广以处理这种情况。

显然，只需对每个 $i \leqslant T(n)$，在给定前 $i-1$ 步的快照后验证第 i 个快照 z_i 是正确的。然而，由于图灵机在每一次只能读/写一个位，因此，为了验证 z_i 的正确性，只需检查之前的两个快照。具体地说，为了验证 z_i 的正确性，只需查看 z_{i-1}，$y_{\text{inputpos}(i)}$，$z_{\text{prev}(i)}$（参见图 2-3），其中 y 是 $x \cdot a$ 的简写；inputpos(i)表示 M 的输入带带头在第 i 步时的位置（记住，输入带是只读的，因此计算过程中输入带始终存储 $x \cdot a$）；prev(i)是 M 的工作带带头在第 i 步之前最后位于第 i 步时带头所处位置的那个步骤[⊖]。这么少的信息之所以足以验证 z_i 的正确性，是因为带上当前单元存储的符号从第 prev(i)个步骤到第 i 个步骤未被修改过。

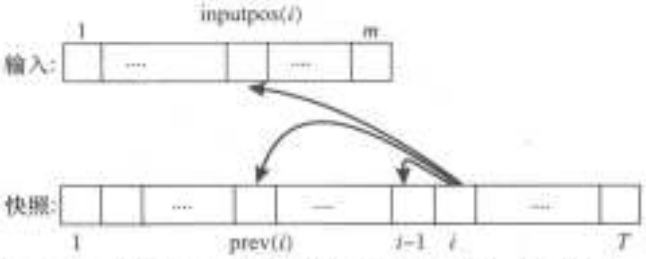


图 2-3　图灵机在第 i 步骤时的快照依赖于前一步的状态（含于第 $i-1$ 步时的快照中），从输入带上读到的符号（出现在输入带的位置 inputpos(i)上）和从工作带上读到的符号（该符号是第 prev(i)步时最后写到工作带上的）

事实上，由于 M 是确定型图灵机，因此在元组 z_{i-1}，$y_{\text{inputpos}(i)}$，$z_{\text{prev}(i)}$ 的所有可能的取值中，至多存在一种取值使得 z_i 是正确的。于是，存在从 $\{0,1\}^{2c+1}$ 到 $\{0,1\}^c$ 的一个函数 F（由 M 的转移函数构造而来）使得正确的 z_i 满足

$$z_i = F(z_{i-1}, y_{\text{inputpos}(i)}, z_{\text{prev}(i)}) \tag{2.3}$$

由于 M 是散漫的，因此 inputpos(i)和 prev(i)的值不依赖于特定的输入 y。同时，正如前面所提到的，这两个下标可以在多项式时间内通过在平凡输入上模拟 M 来获得。

现在，我们将上述思考过程转化为一个归约。注意，由(2.2)式知，输入 $x \in \{0,1\}^*$ 属于 L 当且仅当 $M(x \cdot u)=1$ 对某个 $u \in \{0,1\}^{p(n)}$ 成立。前面的讨论表明，后一个条件成立当且仅当存在位串 $y \in \{0,1\}^{n+p(n)}$ 和一系列位串 $z_1, \cdots, z_{T(n)} \in \{0,1\}^c$（其中 $T(n)$ 是 M 在长度为 $n+p(n)$ 的输入上的计算步数）满足如下 4 个条件：

1. y 的前 n 个位等于 x。

2. 位串 z_1 是 M 的初始快照的编码。亦即，z_1 是元组 $\langle \triangleright, \square, q_{\text{start}} \rangle$ 的编码，其中 $\triangleright$ 是输入带上的开始符号，$\square$ 是空白符号，q_{start} 是图灵机 M 的初始状态。

3. 对任意 $i \in \{2, \cdots, T(n)\}$，$z_i = F(z_{i-1}, y_{\text{inputpos}(i)}, z_{\text{prev}(i)})$。

4. 最后一个位串 $z_{T(n)}$ 编码的快照对应机器停机并输出 1。

公式 φ_x 以 $y \in \{0,1\}^{n+p(n)}$ 和 $z \in \{0,1\}^{cT(n)}$ 为变量，并用以验证 y，z 满足上述四个条件的与。因而，$x \in L \Leftrightarrow \varphi_x \in \text{SAT}$。于是，唯一剩下的事情是证明 φ_x 可以表示为具有多项式大小的 CNF。

条件 1 可以表示为大小为 $4n$ 的 CNF 公式（参见例 2.12）。条件 2 和条件 4 均依赖于 c 个变量，由论断 2.13 可知，这两个条件可以分别表示为大小为 $c2^c$ 的 CNF 公式。条件 3 是 $T(n)$ 个条件的与，其中每个条件至多依赖于 $3c+1$ 个变量，因此，条件 3 可以最终表示为大小至多为 $T(n)(3c+1)2^{3c+1}$ 的 CNF 公式。于是，所有条件的与可以表示为大小为 $d(n+T(n))$ 的 CNF 公式，其中 d 是仅依赖于 M 的常数。而且，该 CNF 公式可以在 M 的运行时间的多项式时间之内被构造出来。

⊖ 如果步骤 i 首次访问一个位置，则定义 prev(i)=1。

2.3.5　将 SAT 归约到 3SAT

要完成定理 2.10 的证明，还需证明下面的引理。

引理 2.14 $SAT \leqslant_p 3SAT$。

证明　我们构造一个变换将任意 CNF 公式 φ 映射为一个 3CNF 公式 ψ，使得 ψ 是可满足的当且仅当 φ 是可满足的。首先，我们考虑 φ 是 4CNF 公式的情形。令 C 是 φ 的一个子句，不妨设 $C=u_1 \vee \overline{u_2} \vee \overline{u_3} \vee u_4$。$\varphi$ 添加一个新变量 z 并将 C 替换为两个子句 $C_1=u_1 \vee \overline{u_2} \vee z$ 和 $C_2=\overline{u_3} \vee u_4 \vee \overline{z}$。显然，如果 $u_1 \vee \overline{u_2} \vee \overline{u_3} \vee u_4$ 为真，则存在 z 的赋值使得 $u_1 \vee \overline{u_2} \vee z$ 和 $\overline{u_3} \vee u_4 \vee \overline{z}$ 均为真；反之亦然，即如果 C 为假，则无论 z 取什么值，C_1，C_2 均为假。同样的思想可用于任意大小为 4 的子句。更进一步，该思想还可以用来将任意大小为 $k(k>3)$ 的子句等价地转换成两个子句 C_1，C_2，其中 C_1 的大小为 $k-1$ 而 C_2 的大小为 3，并且 C_1，C_2 均仅使用 C 中原有的变量和一个新变量 z。不断应用上述变换过程，得到一个多项式时间变换将 CNF 公式 φ 等价地变换为 3CNF 公式 ψ。 ■

49

2.3.6　深入理解库克-勒维定理

库克-勒维定理展示了抽象的作用。无论计算模型是 C 语言还是图灵机，库克-勒维定理均成立。然而，将 C 语言视为计算模型来证明库克-勒维定理可能要困难得多。

库克-勒维定理的证明过程实际上得到了比定理所述内容稍强一些的结果。

1. 如果对用散漫图灵机模拟标准图灵机这一过程采用更高效的方式(参见习题 1.6)，其模拟代价将是对数形式，则库克-勒维定理的证明过程得到的公式 φ_x 的大小将变得更小。则对于任意 $x \in (0,1)^*$，所得公式 φ_x(及计算它的时间)均将变成 $O(T\log T)$，其中 T 是 M 在输入 x 上运行的步数(参见习题 2.12)。

2. 引理 2.11 证明过程中从 **NP** 语言 L 到 SAT 问题的归约 f 不仅满足 $x \in L \Leftrightarrow f(x) \in SAT$，证明过程还表明，它将 $x \in L$ 的证明转换成了满足布尔公式 $f(x)$ 的一个赋值；反过来，满足布尔公式 $f(x)$ 的一个赋值也被转换成 $x \in L$ 的证明。我们称具有这种性质的归约为勒维归约。对证明稍加修改(见习题 2.13)，可以确保证明过程得到一个从 x 的证明集合到 $f(x)$ 的满足性赋值集合之间的一对一映射和满的映射。这意味着，x 的证明集合和 $f(x)$ 的满足性赋值集合具有相同大小。具有这种性质的归约称为简约归约。绝大多数已知的 **NP**-完全问题(包括本章提到的所有 **NP**-完全问题)均存在从所有 **NP** 语言到该问题的简约勒维归约。在第 17 章我们将看到，上述事实在研究 **NP** 问题实例的证明计数的复杂性时非常有用。

为什么是 3SAT 问题?

读者可能会疑惑，为什么 3SAT 问题的 **NP** 完全性比定理 2.9 中语言 TMSAT 的 **NP** 完全性要有意义得多呢？第一个原因是，3SAT 问题在证明其他问题的 **NP** 完全性时非常有用；3SAT 问题具有极其简单的组合结构，便于归约过程采用。第二个原因是，命题逻辑在数理逻辑中具有中心地位，这正是库克和勒维首先研究 3SAT 问题的原因所在。第三个原因是，3SAT 具有重要的实践意义：实际上，3SAT 问题是约束满足问题的一个简单特例，而约束满足问题广泛存在于包含人工智能在内的许多领域中。

2.4　归约网络

库克和勒维必须表明每个 **NP** 语言如何归约到 SAT 问题。然而，要证明其他语言 L 的 **NP** 完全性，我们却不必这么劳力费神了：由定理 2.8 可知，只需将 SAT 问题或 3SAT 问题归约到 L。一旦知道了 L 是 **NP**-完全的，就可以通过将 L 归约到 **NP** 语言 L' 来证明 L' 的 **NP** 完全性。这种方法建立了"归约网络"，并由此证明了数以千计的有趣语言实际上都是 **NP**-完全的。现在，我们证明几个问题的 **NP** 完全性，更多的例子出现在习题中（参见图 2-4）。

50



图 2-4　本章及习题中出现的 **NP**-完全问题的归约网络。数以千计的问题被证明是 **NP**-完全的

回顾一下第 0 章中的如何安排晚宴以确保任意一对宾客之间均能和睦相处的问题，例 0.1 将该问题形式化为如下的语言：

$$\text{INDSET} = \{\langle G, k\rangle : G \text{ 含有大小为 } k \text{ 的独立集}\}$$

定理 2.15　INDSET 是 **NP**-完全的。

证明　如例 2.2 所述，INDSET 属于 **NP**，因此只需证明它是 **NP**-难的。我们将 3SAT 归约到 INDSET。具体地说，我们证明如何在多形式时间内将含有 m 个子句的 3CNF 公式 φ 转换成含有 $7m$ 个顶点的图 G，使得 φ 是可满足的当且仅当 G 含有大小至少为 m 的独立集。

图 G 如下定义（参见图 2-5）：将 φ 中的每个子句关联到由 7 个顶点构成的一个顶点族。与子句 C 关联的顶点族中，7 个顶点分别对应 C 中三个变量的满足 C 的 7 种部分赋值（之所以称为部分赋值，是因为所有变量中仅有部分变量被赋值）。例如，如果 C 是 $\overline{u_2} \vee \overline{u_5} \vee u_7$，则与 C 关联的顶点族中的 7 个顶点分别对应于二进制向量 $\langle a, b, c\rangle \neq \langle 1, 1, 0\rangle$ 表示的部分赋值 $u_2 = a$，$u_5 = b$，$u_7 = c$。（如果 C 中的变量少于 3 个，则重复书写部分赋值中的一个项，这样 7 个顶点中某些顶点可能对应相同的部分赋值。）如果两个部分赋值对其公共变量赋予相同的值，则称它们是一致的。例如，部分赋值 $u_2 = 0$，$u_{11} = 1$，$u_{20} = 1$ 与部分赋值 $u_2 = 1$，$u_5 = 0$，$u_7 = 1$ 不一致，因为它们的公共变量 u_2 被赋予不同的值。如果 G 的两

个顶点表示的部分赋值不一致，则在这两个顶点之间添加一条边。此外，在同一顶点族内
51 部的任意两个顶点之间也添加一条边。

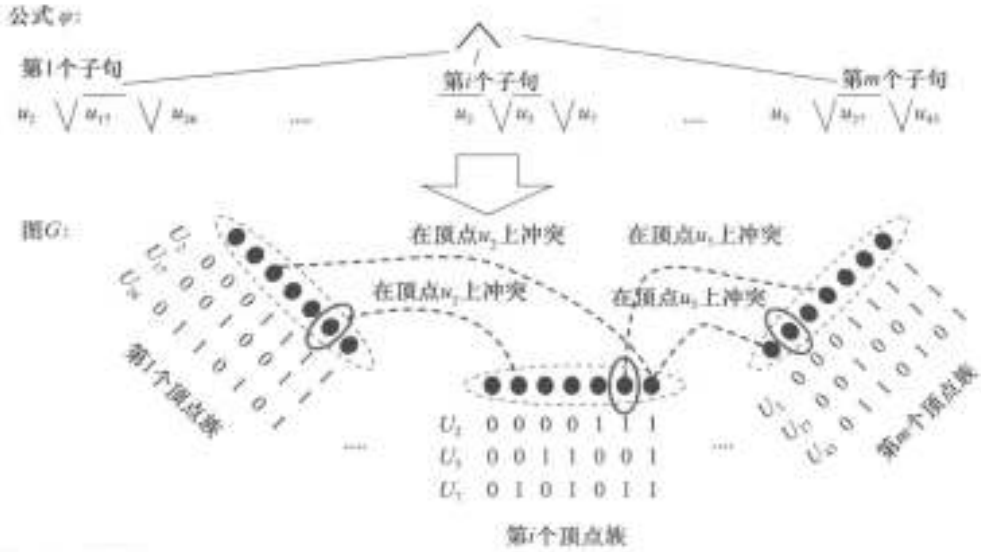


图 2-5　如下将含有 m 个子句的 3CNF 公式 φ 转换成含有 $7m$ 个顶点的图 G：每个子句 C 关联到由 7 个顶点构成的一个顶点族，这 7 个顶点分别对应 C 中三个变量的满足性赋值。在属于同一族的顶点之间和表示不一致的部分赋值的顶点之间添加边。图 G 含有大小为 m 的独立集当且仅当 φ 是可满足的。上图只是边的一个例子。位于圆圈内的 3 个顶点构成一个独立集

显然，将 φ 转换成图 G 可以在多项式时间内完成，因此，还需证明 φ 是可满足的当且仅当 G 含有大小至少为 m 的独立集。

- 假设 φ 有一个满足性赋值 u，如下定义 G 的 m 个顶点构成的集合 S：对于 φ 的每个子句 C，将 C 关联的顶点族中与 u 限制在 C 中变量得到的部分赋值对应的顶点添加到 S 中。由于我们选择的顶点均对应于赋值 u 的限制，因此 S 中没有顶点对应于不一致的部分赋值。故 S 是一个大小为 m 的独立集。
- 假定 G 有一个大小为 m 的独立集 S。我们用 S 来构造 φ 的一个满足性赋值 u。如下定义 u：对于任意 $i\in[n]$，如果 S 中的一个顶点对应的部分赋值给定了 u_i 的值 a，则令 $u_i=a$；否则令 $u_i=0$。上述定义是良定义的，因为 S 是独立集，因此每个变量 u_i 在 S 的所有顶点对应的部分赋值中至多有一个值。另一方面，由于在顶点族内的每对顶点间均添加了边，因而 S 至多包含每个顶点族中的一个顶点，进而 m 个顶点族中每个顶点族均有一个顶点位于 S 中。根据我们对 u 的定义，它满足 φ 的所有子句。 ■

令 0/1 IPROG 是所有可满足的 0/1 整数规划构成的集合，如例 2.3 所述，一个 0/1 整数规划是变量 u_1，…，u_n 上的一组有理系数线性不等式，如果存在变量 u_1，…，u_n 在 $\{0,1\}$ 上的赋值使得给定的所有不等式均被满足，则该 0/1 整数规划属于 0/1 IPROG。

定理 2.16　0/1 IPROG 是 **NP**-完全的。

证明　0/1 IPROG 显然属于 **NP**，因为满足不等式的赋值就是证明。为了将 SAT 归约到
52 0/1 IPROG，只需注意到每个合取范式公式均很容易表示成一个 0/1 整数规划，其中每个子句表示为一个不等式。例如，子句 $u_1\vee\overline{u_2}\vee\overline{u_3}$ 可以表示为 $u_1+(1-u_2)+(1-u_3)\geqslant 1$。 ■

有向图的哈密顿路径是访问该图中所有顶点恰好一次的路径。令 dHAMPATH 表示

由所有含有哈密顿路径的有向图构成的集合。

定理 2.17 dHAMPATH 是 **NP**-完全的。

证明　dHAMPATH 属于 **NP**，因为哈密顿路径上所有顶点构成的有序序列就是证明。为了证明 dHAMPATH 是 **NP**-难的，我们给出一个方法将任意合取范式公式 φ 映射为一个图 G，使得 φ 是可满足的当且仅当 G 含有哈密顿路径(访问图 G 的所有顶点恰好一次的路径)。

归约过程表示为图 2-6。图 G 有(1)m 个顶点，它们分别对应于 φ 的 m 个子句 C_1，…，C_m；(2)一个特定的出发顶点 v_{start} 和一个特定的终止顶点 v_{end}；(3)对应于 φ 中 n 个变量的 n 条"链"，每条链含有 $4m$ 个顶点 v_1，…，v_{4m}，其中 $i\in[4m-1]$，v_i 和 v_{i+1} 用两条方向不同的有向边连接。

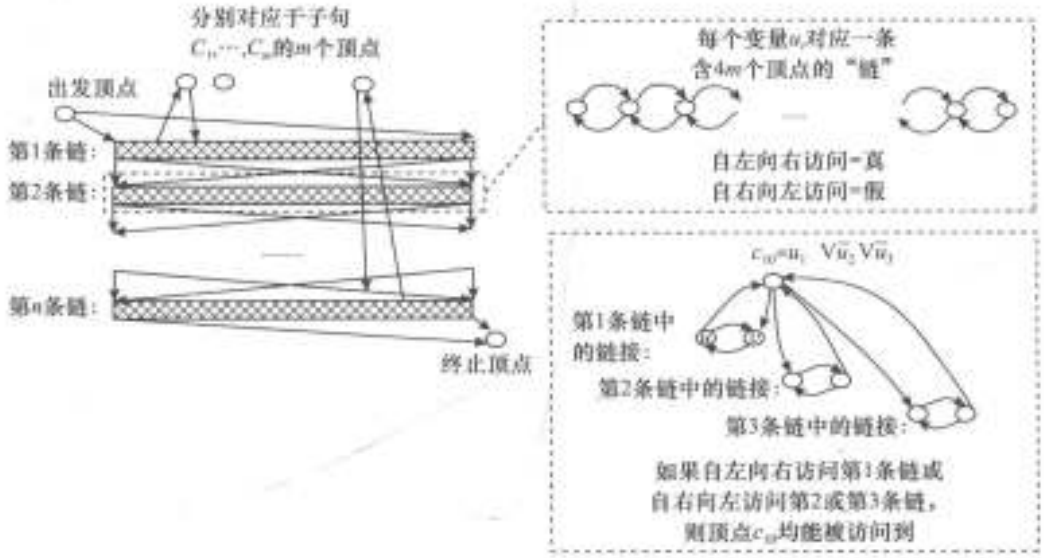


图 2-6　从 SAT 到 dHAMPATH 的归约。具有 n 个变量和 m 个子句的公式 φ 被映射到一个图 G，它含有对应于子句的 m 个顶点和对应于变量的 n 个双向链，每个链含有 $4m$ 个顶点。自左向右访问一个链表示变量取真，而自右向左访问一个链表示变量取假。注意，图中使用从 u 到 c_{ij} 的边的任意哈密顿路径必须马上使用从 c_{ij} 到 v 的边，否则下次访问 v 时路径将"阻塞"

我们从出发顶点 v_{start} 到第 1 条链的两个端点分别添加有向边。对任意 $j\in[n-1]$，还从第 j 条链的两个端点到第 $j+1$ 条链的两个端点分别添加有向边。最后，从第 n 条链的两个端点到终止顶点 v_{end} 分别添加有向边。

除上述这些边之外，对于 φ 的每个子句 C，我们按下面的方式在子句 C 的变量对应的链和子句 C 对应的顶点 v_C 之间添加边：如果 C 包含文字 u_j，则在第 j 条链中取两个相邻顶点 v_i，v_{i+1} 并添加从 v_i 到 v_C 和从 v_C 到 v_{i+1} 的有向边。如果 C 包含文字 $\overline{u_j}$，则用反向的边连接上述顶点(亦即从 v_C 到 v_i 和从 v_{i+1} 到 v_C 添加有向边)。在添加边的过程中，我们在每条链上不会"重用"任何链接 v_i，v_{i+1} 并且在用过的两个链接之间保留一个不用的链接。这可以做到，因为每条链有 $4m$ 个顶点，它们多于需要的顶点个数。现在，我们证明 $\varphi\in$ SAT$\Leftrightarrow$ $G\in$ dHAMPATH。 53

($\varphi\in$ SAT$\Rightarrow G\in$ dHAMPATH)　假设 φ 存在一个满足性赋值 u_1，…，u_n。我们给出访问 G 的所有顶点的一条路径。路径从 v_{start} 出发，依次访问所有的链，最后结束于 v_{end}。

首先，如果 $u_j=1$，则路径从左向右地访问第 j 条链；如果 $u_j=0$，则路径从右向左地访问第 j 条链。这时，除了子句对应的顶点之外，所得路径访问了 G 的其他所有顶点。然而，如果 u 是一个满足性赋值，则上述路径很容易如下修改，使得它访问所有子句对应的顶点：对于每个子句 C，其中至少有一个文字取真值，于是路径可以借助该文字对应的链上的链接"跳"到顶点 v_C，然后再回到以前的路径上继续访问其他顶点。

($G\in$ dHAMPATH$\leqslant_\varphi\in$ SAT)　假设 G 有一条哈密顿路径 P。首先，我们注意到，P 必然起始于顶点 v_{out}（因为该顶点没有入边）而终止于顶点 v_{in}（因为该顶点没有出边）。进一步，我们断言 P 必然顺序访问所有的链，并且访问每条链时要么自左向右进行要么自右向左进行。如果路径没有使用从链到子句对应的顶点的边，则上述断言显然成立。断言在下述情况下也成立：如果哈密顿路径使用了边 $u\to w$，其中 u 是链上顶点而 w 是某个子句对应的顶点，则该路径接下来必然使用边 $w\to v$，其中 v 是相应的链上与 v 相邻的顶点。否则，上述情况不发生，意味着路径必然在下次访问 v 时被阻塞，因为 v 的所有出边均已经被路径使用过（参见图 2-6）。现在，我们如下定义 φ 的一个赋值 u_1，…，u_n：如果 P 自左向右地访问第 j 条链，则定义 $u_j=1$；否则，定义 $u_j=0$。不难看到，由于 P 访问了所有子句对应的顶点，所以 u_1，…，u_n 满足 φ。 ■

归约的赞誉

虽然多项式时间归约概念（以及它的第一个孪生概念——随机多项式时间归约，将在 7.6 节给出定义）的提出源于 **NP** 完全性理论，然而由此引出的对复杂性理论的理解远远超出了 **NP** 完全性本身。如今，复杂性理论和密码学（因而本书的许多章节）的相当一部分工作是利用归约为不同的复杂性理论猜想建立联系。为什么复杂性理论学家均擅长于使用归约，而不擅长于踏踏实实地证明图灵机的下界呢？或许这是由于，他们的创造性更适于创造精巧的构件和设计算法（毕竟，归约只是将一个问题转换为另一个问题的算法），而不是证明图灵机的下界。

2.5　判定与搜索

我们用判定问题（"给定的公式是否是可满足的？"）而不是搜索问题（"如果给定的公式是满足的，则找出一个满足性赋值。"）来定义 **NP** 这一概念。显然，搜索问题比相应的判定问题更难。因此，如果 **P**≠**NP**，则二者均不能在就 **NP**-完全问题求解。然而，对于 **NP**-完全问题而言，如果它的判定形式可以多项式时间内求解（因而 **P**=**NP**），则相应的搜索形式也可以在多项式时间内求解。从这个意义上说，**NP**-完全问题的判定形式和搜索形式是等价的。

54

定理 2.18　*假定* **P**=**NP**。*那么，对于任意* **NP** *语言* L *和* L *的一个验证器图灵机* M（*见定义 2.1*），*存在一个多项式时间图灵机* B *为任意输入* $x\in L$ *计算它*（*相对于语言* L *和图灵机* M）*的证明。*

证明　我们需要证明，如果 **P**=**NP**，则对于任意多项式时间图灵机 M 和多项式 $p(n)$，存在具有下述性质的多项式时间图灵机 B：对于任意 $x\in\{0,1\}^*$，如果 $u\in\{0,1\}^{p(n)}$ 使得 $M(x,u)=1$（亦即 u 是 x 属于 M 所验证的语言的证明），则 $|B(x)|=p(n)$ 且 $M(x,B(x))=1$。

从考虑 SAT 问题入手证明定理。特别地，我们证明，给定判定 SAT 问题的算法 A，可以构造一个算法 B，它以 n 个变量上的可满足的 CNF 公式 φ 为输入，通过 $2n+1$ 次调用

算法 A 和其他多项式时间的计算，输出 φ 的一个满足性赋值。

算法 B 按下述方式操作：首先，用算法 A 验证输入公式 φ 是不是可满足的。如果是，则将 $x_1=0$ 和 $x_1=1$ 依次代入 φ（这种变换略为简化和缩短了公式，使得公式中仅剩下 $n-1$ 个变量，并且该变换肯定可以在多项式时间完成），然后用算法 A 判断哪种代入所得到的公式仍然是可满足的（必有一个是可满足的）。不妨设第一种代入是可满足的。然后，我们固定 $x_1=0$ 在后续过程中保持不变，并用简化后的公式继续。重复上述过程，确保使用的每个中间公式均是可满足的，直到 n 个变量的值全部固定下来。于是，最终得到的赋值满足 φ。

对于任意 **NP** 语言 L，为求解其搜索问题，我们使用如下事实：定理 2.10 证明过程中从 L 到 SAT 的归约实际上是一个parsimonious归约。这意味着，我们有一个多项式时间可计算函数 f，它不仅满足 $x\in L\Leftrightarrow f(x)\in\mathrm{SAT}$，而且还将 $f(x)$ 的一个满足性赋值映射为 x 的一个证明。因此，可以先用上面的算法为 $f(x)$ 计算一个满足性赋值，再将 x 映射回去得到 x 的一个证明。■

定理 2.18 的证明过程表明，SAT 问题是向下自归约的，亦即给定输入长度小于 n 的 SAT 问题求解算法，可以求解输入长度为 n 的 SAT 问题。SAT 问题的向下自归约性十分有用，本书后续章节将几次用到它。利用库克-勒维归约可以证明，所有 **NP**-完全问题均有类似的性质。

2.6　coNP、EXP 和 NEXP

现在，我们定义与 **P** 和 **NP** 相关的其他一些复杂性类。

2.6.1　coNP

如果 $L\subseteq\{0,1\}^*$ 是一个语言，则用 $\overline{L}$ 表示 L 的补集，亦即 $\overline{L}=\{0,1\}^*\setminus L$。由此有如下定义。

定义 2.19　$\mathbf{coNP}=\{L:\overline{L}\in\mathbf{NP}\}$。

coNP 不是类 **NP** 的补集。事实上，**coNP** 与 **NP** 的交集是非空集合，因为 **P** 中每个语言均属于 **coNP**∩**NP**（见习题 2.23）。**coNP** 中一个语言的例子是 $\overline{\mathrm{SAT}}=\{\varphi:\varphi$ 不是可满足的$\}$。学生们有时会错误地认为 $\overline{\mathrm{SAT}}\in\mathbf{NP}$。他们认为的一个多项式时间非确定型图灵机这样工作：在输入 φ 上，机器猜测一个赋值：如果猜测的赋值不满足 φ，则接受输入（即进入状态 q_{accept} 并停机）；如果猜测的赋值满足 φ，则停机并且不接受输入。上述非确定型图灵机不能胜任。事实上，它接受所有不能满足的公式 φ，但是它还额外地接受很多可满足的公式（例如只有一种赋值使公式不能被满足的公式）。因此，从教学方法的角度看，我们更愿意采用下面的方式来定义 **coNP**（很容易证明它与前面的定义等价，参见习题 2.24）。

定义 2.20　（**coNP** 的另一种定义）　对于任意 $L\subseteq\{0,1\}^*$，如果存在多项式 $p:\mathbf{N}\to\mathbf{N}$ 和一个多项式时间图灵机 M 使得对任意 $x\in\{0,1\}^*$ 有

$$x\in L\Leftrightarrow\forall u\in\{0,1\}^{p(|x|)},M(x,u)=1$$

则称 $L\in\mathbf{coNP}$。

注意，本定义使用全称量词 $\forall$ 而定义 2.1 使用存在量词 $\exists$。

类似于 **NP** 完全性，我们也可以定义 **coNP** 完全性。一个语言 L 称为 **coNP** 完全的，如果它属于 **coNP** 并且 **coNP** 中的任意语言均可以多项式时间卡普归约到它。

55

例 2.21 下面的语言是 coNP-完全的。

TAUTOLOGY = {φ | φ 是一个永真公式 —— 能够被任意赋值满足的布尔公式}

由定义 2.20 可知，TAUTOLOGY 属于 **coNP**。因此，我们需要证明，对于任意 $L\in$ **coNP**，均有 $L\leqslant_p$TAUTOLOGY。但这很容易证明，只需修改从 $\overline{L}$(属于 **NP**)到 SAT 的库克-勒维归约即可。对任意输入 $x\in\{0,1\}^*$，库克-勒维归约产生一个公式 φ_x 使得 φ_x 是可满足的当且仅当 $x\in\overline{L}$。现在，考虑公式 $\neg\varphi_x$。它属于 TAUTOLOGY 当且仅当 $x\in L$。这样就得到所需的归约。◀

不难看到，如果 **P**=**NP**，则 **NP**=**coNP**=**P**(习题 2.25)。观察其逆否命题可知，如果能够证明 **NP**≠**coNP**，则就能证明 **P**≠**NP**。绝大部分研究者认为 **NP**≠**coNP**，因为很难相信，"给定的公式属于 TAUTOLOGY"会存在一个简短的证明。换句话说，"所有赋值满足给定公式"很难存在简短证明。显然，这种直觉与 **P** ≠ **NP** 问题的直觉一样强。

2.6.2　EXP 和 NEXP

在论断 2.4 中，我们曾遇到类 $\mathbf{EXP}=\bigcup_{c\geqslant 1}\mathbf{DTIME}(2^{n^c})$，其定义类似于 **P** 的定义，但采用了指数时间。类似地，在 **NP** 的定义中采用指数时间将得到类 **NEXP**，其定义是 $\mathbf{NEXP}=\bigcup_{c\geqslant 1}\mathbf{NTIME}(2^{n^c})$。

56

正如前面看到的那样，由于 **NP** 中的每个问题均可以在指数时间内通过蛮力搜索寻找证明来求解，故 **P**⊆**NP**⊆**EXP**⊆**NEXP**。有必要研究指数时间复杂性类吗？关于这个问题，下面简单的结论给出部分答案。该结论仅仅是我们将要建立的各种不同复杂性问题之间的复杂关系网的冰山一角。

定理 2.22 如果 **EXP**≠**NEXP**，则 **P**≠**NP**。

证明　我们证明其逆否命题。假设 **P**=**NP**，往证 **EXP**=**NEXP**。设 $L\in\mathbf{NTIME}(2^{n^c})$ 并且非确定型图灵机 M 判定该语言。我们断言，语言

$$L_{pad}=\{\langle x,1^{2^{|x|^c}}\rangle : x\in L\} \tag{2.4}$$

属于 **NP**。判定 L_{pad} 的非确定型图灵机如下：在输入 y 上，机器首先检查是否存在位串 z 使得 $y=\langle z,1^{2^{|z|^c}}\rangle$。如果否，则输出 0(即停机但不进入状态 q_{accept})。如果 y 是这种形式，则用 $2^{|z|^c}$ 个步骤模拟 M 在 z 上的计算过程并输出其结果。显然，机器的运行时间是 $|y|$ 的多项式，因此 $L_{pad}\in\mathbf{NP}$。于是，如果 **P**=**NP**，则 $L_{pad}\in\mathbf{P}$。但是，如果 $L_{pad}\in\mathbf{P}$，则 $L\in$ **EXP**。这是由于，为了确定 x 是否属于 L，我们可以先对输入进行填充，然后利用 L_{pad} 的多项式时间图灵机来判定填充后的输入是否属于 L_{pad}。■

上述证明中的填充技术用于将语言进行变形，它将语言中的每个串用一些(无用)符号构成的串进行"填充"。这种技术出现在复杂性理论的几个结果中(例如，参见 14.4.1 节)。在许多场合下，填充技术可用于证明复杂性"增高"后复杂性类之间的相等关系。换句话说，如果用两种不同的计算资源在界限 $T(n)$ 内求解了相同的一些问题，则用这两种不同的计算资源在界限 T' 内也将求解相同的一些问题，其中 T' 大于 T。反过来看，填充技术也可以用来证明资源界限从 $T'(n)$"降低"为资源界限 $T(n)$ 时复杂性类之间的不等关系。

同 **P** 和 **NP** 一样，本书研究的许多复杂性类均同时含于 **EXP** 和 **NEXP** 中。

2.7　深入理解 P、NP 及其他复杂性类

2.7.1　NP 的哲学意义

在纯抽象层面，**P** ≠ **NP** 问题是在查问图灵机模型中非确定性的作用。在有穷自动机等简单模型中，类似的问题已经完全解决。

然而，**NP** 的基于证明的定义还表明，**P** ≠ **NP** 问题刻画了具有一定哲学意义(和挫折源头)的普遍现象：识别答案的正确性比构造答案容易得多；欣赏贝多芬奏鸣曲比谱写奏鸣曲要容易得多；验证吊桥设计方案的稳固性比拿出一个优秀的设计方案容易得多；验证定理证明的正确性比完成证明本身要容易得多(本章开始部分引用的哥德尔的信中指出了这个事实)，诸如此类。在此类情况下，构造正确的答案似乎都需要穷举搜索指数大小的空间。**P** ≠ **NP** 问题是问，一般情况下能否避免这种穷举搜索。对大多数人而言，穷举搜索不能被避免似乎是显而易见的事。因此，一些业余爱好者给出了该问题的许多错误证明。不幸的是，事实已经表明，要将人们的直觉认识变成一个证明过程是异常艰难的。

57

2.7.2　NP 与数学证明

根据定义，**NP** 是成员资格具有短证明的所有语言构成的集合。这不禁让人联想到一个类似概念，即数学证明。正如上个世纪人们注意到的那样，原则上所有数学都可以公理化，因而证明就是公理的简单形式化操作。于是，证明的正确性很容易验证——只须用公理逐行地检查证明的每一行是否均可以由上一行推导出来。事实上，对大多数著名的公理系统(如皮亚诺算术公理和策梅洛-弗兰克尔集合论公理)而言，上述验证过程均能在证明长度的多项式时间内完成。因此，对于通常的任意公理系统 A，下述问题属于 **NP**：

$$\text{THEOREMS} = \{(\varphi, 1^n) : \varphi \text{ 在公理系统 } A \text{ 中存在长度} \leqslant n \text{ 的证明}\}$$

本章开始部分引用的哥德尔 1956 年的信提出的问题是，THEOREMS 问题能否在二次时间内求解。他注意到，这是希尔伯特判定问题的有穷形式，其中希尔伯特判定问题是问：用于判定数学结论是否存在证明(无长度要求)的算法过程是否存在。哥德尔指出，如果 THEOREMS 问题能够在二次时间内求解，则希尔伯特判定问题的不可判定性就不那么令人沮丧了，这是由于人们通常只对不太长(不妨假设可以写成几部书)的证明感兴趣。

习题 2.11 要求证明 THEOREMS 问题是 **NP**-完全的。因此，**P** ≠ **NP** 问题是哥德尔问题的重述。哥德尔问题是问，是否存在一个算法能够在证明长度的多项式时间内找到数学证明。

当然，你也会勇于承认，找到数学证明比验证证明的正确性要难得多。因此不难猜想，你也在直觉上相信 **P**≠**NP**。

2.7.3　如果 P＝NP 会怎样

如果 **P**＝**NP**——具体地讲，如果像 3SAT 这样的一个 **NP**-完全问题存在二次时间的高效算法——则世界很可能会变成一个计算乌托邦。数学家将会被高效的证明发现程序取代(该事实在哥德尔 1956 年的信中就曾被指出，并且 20 年后又重新被指出)。一般而言，对于答案能够被高效地验证(或正确性具有短证明)的任何搜索问题，我们将能够在多项式时间内高效地找到答案或短证明。人工智能(AI)软件将变得非常完美，因为对大型概率树的

穷举搜索将很容易完成。发明家和工程师将可以借助各种软件包来为当前任务设计出完美
58 的零部件。超大规模集成(VLSI)工程师将能够采用耗电量最小的最佳电路。一旦科学家获得了一些实验数据，她将能够自动地找出解释这些实验数据的最简理论(对最简性需要选用合理的度量)；根据奥卡姆剃刀(Occam's Razor)原理[⊖]，最简单的解释很可能是正确的。然而，在某些情况下，科学家花费了几百年才找到解释已知实验数据的最简理论。这种方法也可以用来处理非科学问题。比如，可以从纽约《时代》杂志的最佳销售商列表中找出一个能够解释该列表的最简理论。(当然，找到"最简"的正确定义本身甚至就需要在人工智能和自然语言理解方面有所突破，以便 **NP** 算法能够在这两个领域中使用。)所有这些应用均是第 5 章将要研究的多项式分层的结果(找出最简"理论"的问题与第 5 章研究的 MIN-EQ-DNF 问题密切相关)。

有趣的是，计算乌托邦不需要考虑随机性。正如我们将看到的那样，如果 **P**=**NP**，则在确定型算法中引入随机性并不能获得效率的提高，参见第 7 章。(这一点值得哲学家们深思。)

计算乌托邦需要付出的代价是：数字领域中不再存在隐私。任何加密方案均存在平凡的解码算法。数字货币、安全套接层(SSL)、公钥密码体系(RSA)和完美隐私(PGP)都将不存在(参见第 9 章)。这样，大家就必须学会在没有这些隐私保护机制下如何与人相处。

计算乌托邦显得很荒谬，然而我们却无法排除它存在的可能性。这一事实表明人类对计算的认识还少得可怜。从半全杯(half-full cup)的观点看，还存在大量精彩的事情有待发现。

2.7.4　如果 NP=coNP 会怎样

如果 **NP**=**coNP**，结果同样受人关注。主要结果是，看上去不存在的短证明将会存在。例如，如下的 **NP**-完全问题就是一个例子：判定给定的多变量多项式是否存在公共根(参见习题 2.20)。也就是说，判定下面的方程组是否有解是一个 **NP**-完全问题：

$$\begin{aligned} f_1(x_1,\cdots,x_n) &= 0 \\ f_2(x_1,\cdots,x_n) &= 0 \\ &\vdots \\ f_m(x_1,\cdots,x_n) &= 0 \end{aligned}$$

其中每个 f_i 均是次数不超过 2 的多项式。

如果方程组有解，则这个解就是方程组有解的证明(当然，我们需要证明这个解可以
59 表示为具有多项式长度的位串，此处从略)。判定方程组是否无解的问题显然属于 **coNP**。我们能给出方程组无解的证明吗？希尔伯特零点定理似乎可以胜任这项工作。该定理是说，方程组无解当且仅当存在多项式序列 g_1，…，g_m 使得 $\sum_i f_i g_i = 1$，其中等式右端的 1 表示常数多项式 1。

这是什么意思？难道零点定理证明了 **NP**=**coNP**？不是，因为多项式 g_i 的次数可能是 n、m 的指数形式，这意味着表示这些多项式可能会花费 n、m 的指数长度。很容易构造

⊖ 奥卡姆剃刀原理是一个著名的哲学原理，该原理在机器学习这一计算机科学分支学科中已经得到应用。关于可学习性的瓦利安特(Valiant)理论[Val84]给出了奥卡姆剃刀的数学基础。瓦利安特理论深受计算复杂性理论的影响，有关于此的精彩论述参见卡恩斯(Kearns)和瓦利安特(Valiant)的著作[KV94]。如果 **P**=**NP**，则机器学习中许多有趣的问题都将存在多项式时间算法。

出这样的 f_i，使得 g_i 的次数是 n、m 的指数。

然而，如果 **NP**＝**coNP**，则必将产生其他概念来刻画方程组无解的短证明。这种概念对数学的影响可能比希尔伯特零点定理更加深远。（希尔伯特零点定理在第 15 章和第 16 章还会出现。）

2.7.5　NP 和 NP 完全之间存在其他复杂性类吗

NP 完全性理论极其有用和重要，因为数千个有用的问题已被证明是 **NP**-完全的（继而可以假定它们不属于 **P**）。然而，还有一些有趣的 **NP** 问题既未被证明属于 **P** 也未被证明是 **NP**-完全的。对于这些问题，最好能采用某种方法来表明它们尽管很难求解但其难度却很难量化。有时，研究者们将一些更著名的问题用问题自身的名字来单独命名复杂性类；例如，因素分解问题（第 9 章会用到）和所谓的唯一性游戏猜想问题（第 22 章）。这些问题的复杂性与其他问题的复杂性密切相关。类似地，帕帕迪米特里奥（Papadimitriou）[Pap90] 定义了大量 **P** 和 **NP** 之间有趣的复杂性类来刻画各种有趣问题的复杂性。其中最重要的类是 **PPAD**，它刻画了为两方博弈寻找纳什均衡这个问题的复杂性。

有时，可以如下证明这些问题不太可能是 **NP**-完全的。我们证明，如果某个问题是 **NP**-完全的，则某个其他的猜想将不成立（这里，采用的猜想至少与 **P**≠**NP** 具有同样的可信度）。在 8.1.3 节中，我们将在图同构问题上看到这样一个结果。

我们在 3.3 节将看到另一个结果——拉德纳尔定理（Ladner's Theorem），它断言，如果 **P**≠**NP**，则存在既不属于 **P** 也不是 **NP**-完全的问题。

2.7.6　NP 难的处理

NP-完全问题在大量应用中涌现，这些应用涉及从航班调度到基因序列化的各个领域。如果你手中待解的问题被证明是一个 **NP**-完全问题，你怎么办？初看起来，形势似乎不妙：如果 **P**≠**NP**，则不存在求解这种问题的高效算法[⊖]。然而，也不是毫无求解的希望：**NP** 完全性仅仅意味着（假设 **P**≠**NP**）不存在能够在任意输入上精确求解问题的高效算法。 60
但是，对许多应用而言，能给出某些输入上的近似解就足矣。

旅行售货商问题（TSP 问题）就是如此，它要求在任意两个城市之间的距离已知的 n 个城市中安排一条最短的旅行线路来访问所有的城市。假设你负责为商人们安排旅行路线以便他们能访问你们国家的一些城市。难道说，仅仅由于 TSP 问题是一个 **NP**-完全问题，你就给他们胡乱安排一些不靠谱的旅行路线吗？情况显然不是这样的。

首先，注意到，你需要的算法并不是要在所有可能的距离上求解问题。我们可以对上述情况中实际出现的输入这样建模：将 n 个城市表示成平面上的点，两个城市间的距离对应相应两点之间的距离（这里忽略了旅行距离和有向/无向距离之间的区别）。不难证明，并不是所有可能的距离均可以用这种方式表示。我们将可以用这种方式表示的距离成为欧几里得距离。另外，我们还注意到，准确地计算最优旅行路线并没有那么重要。事实上，如果你总能构造出代价不超过最优旅行路线代价 1% 的旅行路线，则这就已经足够好了。

事实证明，上面注意到的两个事实无论哪一个也不足以使得问题的求解更容易。在欧几里得距离下定义的 TSP 问题仍是 **NP**-完全的；并且，如果 **P**≠**NP**，则 TSP 问题也不能

⊖ 排除如下情形：对问题描述稍作修改将导致问题的复杂性发生巨大变化。因此，用抽象问题对实际问题建模时必须格外小心，不要将简单问题建模成一个 **NP** 完全问题。

近似到常数比范围内。但是，把这两个事实放在一起却确实使得问题更容易求解：存在一个时间复杂度为 $\mathrm{poly}(n(\log n)^{O(1/\varepsilon)})$ 的近似算法，它以 n 个城市之间的欧几里得距离和任意 ε 为输入，输出一个代价不超过最优解代价 $(1+\varepsilon)$ 倍的近似旅行路线[Aro96]。

因此，发现待求解的问题是 **NP**-完全问题不能成为你束手无策的理由。相反，你应将它视为一种提示你更仔细地对问题进行建模的信号，让你从文献中学到的复杂性和算法知识帮助你认识清楚：问题的哪些特征会使得问题更易于处理。另一种方法是考虑最坏情况下的精确计算。本书第 11 章和第 18 章将讨论相关内容，其中第 11 章讨论近似算法而第 18 章讨论平均复杂性。

2.7.7　更精细的时间复杂性

本章旨在集中讨论多项式时间和非多项式时间的区别。研究者们还探讨了关于时间复杂性的更精细的一些问题。例如，对于一个计算问题，不妨设是 INDSET 问题，我们相信它不能在多项式时间内求解。那么，它确切的时间复杂性到底是什么呢？是 $n^{(1/2)\log n}$，是 $2^{n^{1/2}}$，还是 $2^{n/10}$ 呢？多数的研究者相信它其实是 $2^{\Theta(n)}$，其背后的直觉是：穷举所有子集的这个平凡算法接近于最优算法。

在最大独立集的大小至多为 k 的情况下，验证上述直觉十分有益。平凡算法枚举大小为 k 的所有可能的顶点子集，因此它花费的时间为 $\binom{n}{k}\approx n^k$，其中 $k\ll n$（此时，k 可以视为一个任意大的常数）。我们能做得更好吗？比如，能否在 $2^k\mathrm{poly}(n)$ 时间内完成？或者，更一般地，能否在 $f(k)\mathrm{poly}(n)$ 时间内完成，其中 f 是某个函数？固定参数难解性理论研究这样的问题。很多 **NP** 问题（其中包括 INDSET 问题）对固定参数难解性而言是完全的；也就是说，其中一个问题存在 $f(k)\mathrm{poly}(n)$ 时间算法当且仅当所有这些问题均存在这种算法。毫无疑问，这种“完全性”也是相对于一种特殊的归约而言的。关于这个专题，一本优秀的参考书是[FG06]。

61

类似地，我们或许还会问，是否存在 **NP** 完全性的某种扩展切实地表达了“INDSET 问题和其他许多 **NP** 完全问题的时间复杂性是 $2^{\Theta(n)}$ 而不仅仅是非多项式”这种直觉认识。因帕利亚佐(Impagliazzo)、帕图里(Paturi)和赞恩(Zane)[IPZ98]中给出了这样的一种理论，它用到了一种为研究这类问题而量身定制的归约概念。

本章学习内容

- 类 **NP** 由成员资格存在多项式时间验证算法的所有语言构成。它包含了许多已知不属于 **P** 的重要问题。**NP** 也可以用非确定型图灵机来定义。
- **NP**-完全问题是 **NP** 中最难的问题，其确切含义是，**NP**-完全问题存在多项式时间算法当且仅当 **P**=**NP**。与图灵机似乎毫无关系的许多自然的问题已被证明是 **NP**-完全的。3SAT 语言就是这样的例子，它由可满足的所有 3CNF 布尔公式构成。
- 如果 **P**=**NP**，则解能够被高效验证的任何搜索问题也能够被高效地求解。
- 类 **coNP** 是由所有 **NP** 语言的补集构成的集合。人们相信 **coNP**≠**NP**，它是一个比 **P**≠**NP** 还强的假设。

本章注记和历史

从 20 世纪 50 年代起，前苏联科学家就已经意识到一个事与心违的事实：求解组合问

题要用穷举搜索或蛮力搜索(他们称穷举搜索为"最热已")。同时，他们还提出一个问题：有些问题在本质上是不是必须用穷举搜索才能求解(有关历史请参阅[Tra84])。在西方，这个问题的首次公开表述是埃德蒙兹[Edm65]在本章开头引用的信中给出的。然而，东方和西方在相互隔绝的情况下均花费了很长的时间才找到正确的方法将这个问题形式化，并最终演变成现代对类 **P** 和 **NP** 的定义。令人惊讶的是，在本章开头引用的哥德尔在 1956 年写给冯·诺依曼的信中，哥德尔本质上已经提出了 **P** 与 **NP** 问题，虽然没有证据表明他已经意识到他提到的一个问题是 **NP**-完全的。不幸的是，冯·诺依曼当时已经病入膏肓。并且，据我们现在所知，哥德尔和冯·诺依曼均没有在该问题上作进一步的研究，而那封信在 20 世纪 80 年代才被人们发现。

1971 年，库克发表了他的开创性论文[Coo71]。他在这篇论文中定义了 **NP** 完全性的概念，并证明了 SAT 问题是 **NP**-完全的。随后，卡普[Kar72]证明了 21 个重要的问题实际均是 **NP**-完全的，由此掀起了 **NP** 完全性概念的热潮。同时，在前苏联，勒维独立地定义了 **NP** 完全性(尽管其研究的焦点是搜索问题)，并证明了 SAT 问题的一种变形是 **NP**-完全的。勒维的论文[Lev73]发表于 1973 年，但他从 1971 年开始就在演讲中阐述其研究结果。在那个年代，东方和西方的科学家本质上没有任何交流。特拉克特恩布诺特(Trakk- [62]
tenbrot)的综述[Tra84]介绍了勒维的发现，并准确地翻译了勒维的论文。要了解更多关于 **P** 和 **NP** 的历史和哥德尔的著名信件的完整翻译，请参阅西普赛尔(Sipser)的综述[Sip92]。

加里(Garey)和约翰逊(Johnson)的书[GJ79]以及网站[CK00]中均包含了大量 **NP**-完全问题的例子。有些 **NP**-完全问题在计算机被发明之前就得到了深入的研究。例如，旅行售货商问题的研究始于 19 世纪(参见[LLKS85])。再比如，最近发现的一封高斯(Gauss)写给舒马赫(Schumacher)的信表明，早在 19 世纪初高斯就一直在思考如何求解著名的欧几里得斯坦纳树(Euclidean Steiner tree)问题，人们现在已经知道这个问题是 **NP**-完全的。想更多地了解关于 **NP** 和数学的关系，请参阅维格德尔森(Wigderson)的综述[Wig06]。

阿伦森(Aaronson)[Aar05]综述了利用各种"非传统"计算设备求解 **NP**-完全问题的各种尝试。

即使 **NP**≠**P**，这也不意味着我们在 2.7.3 节中提到的各种乌托邦式的应用就必定不能实现。或许，比方说 3SAT 问题，最坏情况下它虽然在任意输入上难以求解，但可能在平均情况下却很容易求解。关于平均复杂性的详细研究请参见第 18 章。关于这个专题，因帕利亚佐也做了精彩的综述[Imp95b]。

一种耐人寻味的可能是，用数学上现有的公理根本不可能解决 **P** ≠ **NP** 问题。这种可能性已经在康托尔(Cantor)的"连续统假设"等其他问题上得到了印证。阿伦森的综述[Aar03]探讨了这种可能性。

阿龙(Alon)和吉莉安(Kilian)(私下交流)曾证明，在例 2.3 给出的因数问题的定义中，条件"p 是素数"对定义因数问题是必需的，因为缺少该条件之后语言将是 **NP**-完全的(这也正是它与整数的因数分解毫无关系的原因)。

习题

2.1　证明：在定义 2.1 中允许证明的长度至多为 $p(|x|)$(而不是等于 $p(|x|)$)不会造成任何区别。即证明：对于任意多项式时间图灵机 M 和多项式 p：**N**→**N**，语言

$$\{x:\ \exists u \text{ 满足 } |u|\leqslant p(|x|) \text{ 且 } M(x,\ u)=1\}$$

属于 **NP**。

2.2　证明下列语言属于 **NP**。

2 着色：2COL＝{G：图 G 可以用两种颜色的着色}，其中用 c 种颜色对 G 着色是为 G 的每个顶点赋予[c]中的一个数使得没有相邻顶点上的数是相等的。

3 着色：3COL＝{G：G 可以用三种颜色的着色}

连通性：CONNECTED＝{G：G 是连通图}

63　哪个语言属于 **P**？

2.3　令 LINEQ 表示由可满足的所有有理线性方程组构成的集合；亦即，LINEQ 是所有序对$\langle A, b\rangle$构成的集合，其中 A 是一个 $m\times n$ 的有理数矩阵，b 是 m 维有理数向量，并且 $Ax=b$ 对某个 n 维向量 x 成立。证明：LINEQ 属于 **NP**（关键是证明，如果这样的 x 存在，则必存在一个 x 使得它的各个系数表示成位串之后其长度是 A，b 的位串表示的长度的多项式）。（注意，LINEQ 实际上属于 **P**。你能证明该结论吗？）

2.4　证明：例 2.3 中的线性规划问题属于 **NP**。（同样，该问题也属于 **P**，其证明是一个极不平凡的算法[Kha79]）

2.5　[Pra75]令 PRIMES＝{$\langle n\rangle$：n 是素数}。证明：PRIMES$\in$**NP**。你可以借助下述事实：数 n 是素数当且仅当对于 $n-1$ 的任意素因子 q，存在数 $a\in\{2,\cdots,n-1\}$ 满足 $a^{n-1}=1(\mathrm{mod}\ n)$但 $a^{(n-1)/q}\neq 1(\mathrm{mod}\ n)$。

2.6　证明：存在非确定型通用图灵机（类似于定理 1.9 中的确定型通用图灵机）。即证明：存在非确定型图灵机的表示方法和一个非确定型图灵机$\mathcal{NU}$使得对任意位串 α 和输入 x，均有$\mathcal{NU}(x,\alpha)=M_\alpha(x)$成立。

（**a**）证明：存在一个非确定型通用图灵机$\mathcal{NU}$使得，如果 M_α 以 x 为输入时在 T 步内停机，则$\mathcal{NU}$以 x，α 为输入时在 $CT\log T$ 步内停机，其中 C 是依赖于 α 所表示的机器的常数。

（**b**）证明：存在一个非确定型通用图灵机使得它在上述输入上在 CT 步内停机。

2.7　证明定理 2.8 的第 2 部分和第 3 部分。

2.8　令 HALT 是定理 1.11 中定义的停机语言。证明：HALT 是 **NP**-难的。它是 **NP**-完全的吗？

2.9　我们已经在所有语言中定义了关系$\leqslant_p$。注意，它是自反的（即，$L\leqslant_p L$ 对任意语言 L 成立）和传递的（即，如果 $L\leqslant_p L'$且 $L'\leqslant_p L''$，则 $L\leqslant_p L''$）。证明：该关系不是对称的，即 $L\leqslant_p L'$不必蕴含 $L'\leqslant_p L$。

2.10　假设 L_1，$L_2\in$**NP**。那么，$L_1\cup L_2$属于 **NP** 吗？$L_1\cap L_2$呢？

2.11　数学可以用策梅洛-弗兰克尔公理系统这类的具有有穷描述的公理系统来公理化。在较高的层次上（不必去了解策梅洛-弗兰克尔公理系统的任何细节）证明下面的语言是 **NP**-完全的。

$\{\langle\varphi,1^n\rangle$：数学结论 φ 在策梅洛-弗兰克尔公理系统中有长度不超过 n 的证明$\}$

上述语言是否属于 **P**，这个问题本质上就是在本章前面引用的信中哥德尔提出的问题。

2.12　证明：对于任意的时间可构造的 T：**N**$\to$**N**，如果 $L\in$**NTIME**$(T(n))$，则可以给出一个从 L 到 3SAT 的多项式时间卡普归约使得它将大小为 n 的实例转换成大小为 $O(T(n)\log T(n))$的 3CNF 公式。你能让该归约的运行时间也是 $O(T(n)\mathrm{polylog}T(n))$吗？

64

2.13　回顾一下，**NP** 语言 L 到 **NP** 语言 L'的归约 f 称为简约的，如果 x 的证明的个数等于 $f(x)$的证明的个数。

(a) 证明：引理 2.11 的证明过程中给出的从任意 **NP** 语言 L 到 SAT 的归约均可以改造成简约归约。

(b) 给出从 SAT 到 3SAT 的一个简约归约。

2.14　库克[Coo71]使用了不同的归约概念：语言 L 被多项式时间库克归约到语言 L'，如果存在多项式时间图灵机 M 使得它能够在给定的判定 L' 的神喻下判定 L。L' 的神喻是额外给 M 的一条魔法带，任何时候只要 M 在该带上写下一个串并进入一个特殊的"调用"状态，则机器将在一个步骤内根据该带上的串是否属于 L' 分别将这个串改写成 1 或 0；更精确的定义参见 3.4 节。

证明：库克归约是传递的，并且 3SAT 可以库克归约到 TAUTOLOGY。

2.15　在 CLIQUE 问题中，我们给定一个无向图 G 和一个整数 K，要求判定是否存在大小至少为 K 的顶点子集 S 使得任意两个不同顶点 u，$v \in S$ 之间均存在边(这样的顶点子集称为 G 的团)。在 VERTEX-COVER 问题中，我们给定一个无向图 G 和一个整数 K，要求判定是否存在大小至多为 K 的顶点子集 S 使得 G 的任意边 (i,j) 均至少有一个端点(i 或 j)属于 S(这样的顶点子集称为 G 的顶点覆盖)。证明：上述两个问题均是 **NP**-完全的。

2.16　在 MAX CUT 问题中，我们给定一个无向图 G 和一个整数 K，要求判定是否存在顶点子集 S 使得至少有 K 条边的一个端点属于 S 而另一个端点属于 $\overline{S}$。证明：上述问题是 **NP**-完全的。

2.17　在 EXACTLY ONE 3SAT 问题中，我们给定一个 3CNF 公式 φ，要求判定是否存在一个满足 φ 的赋值 u 使得 φ 中每个子句恰有一个文字为真。在 SUBSET SUM 问题中，我们给定 n 个数的序列 A_1，…，A_n 和一个数 T，要求判定是否存在子集 $S \in [n]$ 使得 $\sum_{i \in S} A_i = T$(问题的大小指的是将所有数表示为位串时所用二进制位的个数和)。证明：EXACTLY ONE 3SAT 问题和 SUBSET SUM 问题均是 **NP**-完全的。

2.18　证明由包含哈密顿路径的所有无向图构成的语言 HAMPATH 是 **NP**-完全的。证明例 2.3 给出的 TSP 语言是 **NP**-完全的。证明由包含哈密顿环(即包含所有顶点的简单环)的所有无向图构成的语言 HAMCYCLE 是 **NP**-完全的。

2.19　令 QUADEQ 是 0/1 变量上所有可满足的二次方程构成的集合(变量 u_1，…，u_n 上的二次方程形如 $\sum_{i,j \in [n]} a_{i,j} u_i u_j = b$，其中加法对 2 取模)。证明：QUADEQ 是 **NP**-完全的。 65

2.20　令 REALQUADEQ 是由实数变量上所有可满足的二次方程构成的语言。证明：REALQUADEQ 是 **NP**-完全的。

2.21　证明：3COL(参见习题 2.2)是 **NP**-完全的。

2.22　在有 n 项拍卖品的一个典型的拍卖会上，拍卖师将会把第 i 项拍卖品卖给出价最高的出价人。然而，有时一些拍卖品与另一项拍卖品是关联的(例如，待售的很多块地皮与另一块地皮邻接)；因此，出价人只有在能够将拍卖品{2，5，17}一起购得的情况下，才愿意出更高的价格购买它们。在这种情况下，决定将哪件拍卖品出售给哪位出价人并不是一件容易的事情。COMBINATORIAL AUCTION 问题是：给定 n，k 和一系列序对 $(S_i, x_i)_{i=1}^{m}$，其中 S_i 是 $[n]$ 的子集，x_i 是一个整数，要求判定是否存在不相交的集合 S_{i_1}，…，S_{i_l} 使得 $\sum_{j=1}^{l} S_{i_j} \geq k$。亦即，如果 x_i 是拍卖人对

S_i的出价，则问题是问拍卖师能够卖完所有拍卖品并使收入至少为 k，显然应遵守的条件是不能把同一件拍卖品出售两次。证明：COMBINATORIAL AUCTION 是 NP-完全的。

2.23　证明：$\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$。

2.24　证明：定义 2.19 和定义 2.20 确实定义了同一个类 **coNP**。

2.25　证明：如果 $\mathbf{P}=\mathbf{NP}$，则 $\mathbf{NP}=\mathbf{coNP}$。

2.26　证明：$\mathbf{NP}=\mathbf{coNP}$ 当且仅当 3SAT 和 TAUTOLOGY 可以在多项式时间从一个归约到另一个。

2.27　不用非确定型图灵机，给出 **NEXP** 的一个定义(类似于定义 2.1 中的类 **NP** 的定义)，再证明它的两个定义等价。

2.28　我们称一个语言是 **NEXP** 完全的，如果它属于 **NEXP**，并且 **NEXP** 中的任意语言均可以多项式时间归约到它。给出一个 **NEXP** 完全语言 L，并证明：如果 $L \in \mathbf{EXP}$，则 $\mathbf{NEXP}=\mathbf{EXP}$。

2.29　假设 L_1，$L_2 \in \mathbf{NP} \cap \mathbf{coNP}$。证明 $L_1 \oplus L_2$ 属于 $\mathbf{NP} \cap \mathbf{coNP}$，其中 $L_1 \oplus L_2 = \{x : x$ 恰出现在 L_1，L_2 之一中$\}$。

2.30　(贝尔曼(Berman)定理 1978)一个语言称为一元的，如果其中的每个串均形如 1^i(i 个 1 形成的串)对某个 $i>0$ 成立。证明：如果存在 **NP** 完全的一元语言，则 $\mathbf{P}=\mathbf{NP}$。(习题 6.9 加强了这个结论。)

2.31　定义语言 UNARY SUBSET SUM 是习题 2.17 中语言 SUBSET SUM 问题的变形，这里要求所有的数均采用一元表示(即，数 k 表示为 1^k)。证明：UNARY SUBSET SUM 属于 **P**。

2.32　证明：如果任意的一元 **NP**-完全问题均属于 **P**，则 $\mathbf{EXP}=\mathbf{NEXP}$。(语言 L 是一元的当且仅当它是$\{1\}^*$的子集，参见习题 2.30。)

66

2.33　令 $\boldsymbol{\Sigma}_2$SAT 表示下述判定问题：给定一个具有如下形式的量化公式 ψ，

$$\psi = \exists_{x \in \{0,1\}^*} \forall_{y \in \{0,1\}^m} \text{ 满足 } \varphi(x,y) = 1$$

其中 φ 是一个合取范式公式，判定 ψ 是否为真。亦即，判定是否存在 x 使得对于任意 y 均有 $\varphi(x, y)$为真。证明：如果 $\mathbf{P}=\mathbf{NP}$，则 $\boldsymbol{\Sigma}_2$SAT 属于 **P**。

2.34　假设给你一个图 G 和一个数 K，并告诉你 G 的最小顶点覆盖(参见习题 2.15)的大小(i)要么至多为 K；(ii)要么至少为 $3K$。给出一个多项式时间算法来区别上述的两种情况。你能为小于 3 的常数设计出同样的算法吗？既然 VERTEX COVER 是 **NP** 难的，为什么该算法不能证明 $\mathbf{P}=\mathbf{NP}$？

67

对角线方法

P 与 **NP** 这一比较性问题在某些神喻上的答案是肯定的，而在另一些神喻上的答案又是否定的。我们认为，这是 **P** 与 **NP** 问题难以处理的新证据。

——贝克(Baker)，吉尔(Gill)，索洛韦(Solovay)[BGS75]

复杂性理论的基本目标是区分一些特定的复杂性类(如 **P** 和 **NP**)。为此，需要从一个类中找出一个机器使得它不同于另一个类中的任意一个机器，其中两个机器不相同的含义是它们至少在一个输入上具有不同的输出。本章讨论对角线方法，它是用于构造上述机器的唯一的一般性方法。

1.5 节已经用对角线方法证明了不可计算函数的存在性。本章将更巧妙地使用对角线方法。首先，在 3.1 和 3.2 节中，我们将用对角线方法证明分层定理。该定理断言，如果给图灵机更多的计算资源，则它必然能求解更多的问题。然后，在 3.3 节，我们将用对角线方法证明拉德纳尔的一个引人注目的定理。该定理断言，如果 **P**≠**NP**，则存在既不是 **NP**-完全的也不属于 **P** 的问题。

尽管对角线方法在复杂性理论诞生早期获得了这些成果，但是研究者们在 20 世纪 70 年代已经做出了如下结论：仅用对角线方法不足以解决 **P** 与 **NP** 问题以及其他一些有意义的问题。3.4 节阐释了其中的原因。有趣的是，对角线方法的局限性是用对角线方法本身证得的。

对角线方法的局限性使得该方法多年不受青睐，却使得线路下界等其他方法逐渐流行起来(第 14 章对此进行了介绍)。然而，现在其他方法也遇到了障碍，而在最近的一些研究成果中对角线方法又重见天日成为了关键技术(20.4 节给出了一个例子)。因此，未来的复杂性理论家在接触研究前沿之前应该先掌握这种简单的思想。

机器的位串表示和通用图灵机

对角线证明方法的唯一公用工具就是图灵机的位串表示，1.4 节讨论了这种表示，我们回顾一下这种表示的一些显著特点。首先，位串表示是高效的。确切地说，存在一个通用图灵机，它能够用较小的开销(即至多下降一个对数因子)模拟任意位串 x 表示的图灵机。其次，任意位串 $x\in\{0,1\}^*$ 均表示一个图灵机(记为 M_x)，并且每个图灵机可以表示为无穷多个位串。(虽然这显得有些挑剔，但却有助于简化证明。)最后，本章将始终使用记号 M_i(其中 $i\in\mathbf{N}$)来表示整数 i 的二进制形式(去掉第一个 1 之后)的位串对应的图灵机。

3.1 时间分层定理

时间分层定理表明，如果允许图灵机使用更长的计算时间，则图灵机判定的语言集将会严格增大。回想一下，对任意函数 $f:\mathbf{N}\to\mathbf{N}$，**DTIME**($f(n)$)是由所有能够被图灵机在 $O(f(n))$时间内判定的语言构成的集合。通常，我们只讨论时间可构造函数 f：亦即，映射 $x\mapsto f(x)$可以在 $O(f(n))$时间内被计算(参见 1.3 节和 1.6 节)。

定理 3.1（时间分层定理[HS65]）　如果 f，g 是满足 $f(n)\log f(n)=o(g(n))$ 的时间可构造函数，则

$$\mathbf{DTIME}(f(n)) \subset \mathbf{DTIME}(g(n)) \tag{3.1}$$

证明　为了用最少记号展示定理 3.1 证明过程的本质思想，我们证明稍简单的结论 $\mathbf{DTIME}(n)\subset\mathbf{DTIME}(n^{1.5})$。

考虑下面的图灵机 D："在输入 x 上，运行定理 1.9 中的通用图灵机 $\mathcal{U}$ 模拟 M_x 在 x 上执行的 $|x|^{1.4}$ 个步骤。此时，如果 $\mathcal{U}$ 输出 $\{0,1\}$ 中的一个位 b，则 D 输出相反的答案 $1-b$；否则，D 输出 0。"这里，M_x 是用串 x 表示的图灵机。

根据定义，D 在 $n^{1.4}$ 个步骤内停机，因此由 D 判定的语言 L 属于 $\mathbf{DTIME}(n^{1.5})$。我们断言 $L\notin\mathbf{DTIME}(n)$。若不然，假设存在图灵机 M 和常数 c，使得给定任意输入 $x\in\{0,1\}^*$，M 在 $c|x|$ 个步骤内停机并输出 $D(x)$。

通用图灵机 $\mathcal{U}$ 模拟 M 在任意输入 x 上运行时，至多在 $c'c|x|\log|x|$ 个步骤内必然停机，其中 c' 是依赖于 M 的字母表大小、带的条数和状态个数而独立于 $|x|$ 的常数。存在数 n_0 使得 $n^{1.4}>c'cn\log n$ 对任意 $n\geqslant n_0$ 恒成立。令 x 是表示 M 的长度至少为 n_0 的一个位串（这样的串存在，因为表示 M 的位串有无穷个）。这样，$D(x)$ 将在 $|x|^{1.4}$ 个步骤内得到输出 $b=M(x)$；但根据 D 的定义，我们又有 $D(x)=1-b\neq M(x)$。由此得出矛盾。

对于定理 3.1 在一般 f，g 上的情形，利用通用图灵机模拟其他机器时效率至多下降一个对数因子这一事实，可以得到类似的证明过程。■

3.2　非确定型时间分层定理

下面是非确定型图灵机的分层定理。

定理 3.2（非确定型时间分层定理[Coo72]）　如果 f，g 是满足 $f(n+1)=o(g(n))$ 的时间可构造函数，则

69

$$\mathbf{NTIME}(f(n)) \subset \mathbf{NTIME}(g(n)) \tag{3.2}$$

证明　同样，我们仅通过证明 $\mathbf{NTIME}(n)\subset\mathbf{NTIME}(n^{1.5})$ 来展示定理证明的主要思想。直觉上看，只需将定理 3.1 的证明照搬过来即可，因为在非确定型计算上也存在通用图灵机（参见习题 2.6）。但是，仅仅这样还不能证明定理，因为新定义的机器 D 需要具备"翻转答案"的能力；亦即，给定任意非确定型图灵机 M 和输入 x，D 需要高效地计算 $1-M(x)$。在非确定型通用图灵机上如何达成上述目标并不是显然的；这是由于，正如我们先前（在 2.6.1 节中）讨论猜想 $\mathbf{NP}\neq\mathbf{coNP}$ 时所见的那样，非确定型图灵机如何"翻转答案"并不十分清楚。具体地讲，不能指望一个 $\mathbf{NTIME}(n)$ 语言的补集是一个 $\mathbf{NTIME}(n^{1.5})$ 语言。然而，任意 $\mathbf{NTIME}(n)$ 语言的补集却显然可以在指数时间内平凡判定（甚至可以用确定型图灵机来完成判定），这只需逐个查验非确定型图灵机所有可能的非确定型选择，但这似乎与证明 $\mathbf{NTIME}(n)\subset\mathbf{NTIME}(n^{1.5})$ 毫无关系。令人惊讶的是，非确定型图灵机的这种平凡的模拟确实可以用来建立分层定理。

证明的关键思想是采用惰性对角线方法。之所以这样称呼它是因为，新构建的机器 D 不急于对角化，而仅仅确保它能将任意线性时间非确定型图灵机 M_x 在众多（指数个）输入串中的一个输入串上的答案翻转。

定义函数 $h:\mathbf{N}\to\mathbf{N}$ 为：$h(1)=2$，$h(i+1)=2^{h(i)^{1.5}}$。给定 n，不难在 $O(n^{1.5})$ 时间内找出一个整数 i 使得 n 介于 $h(i)$ 和 $h(i+1)$ 之间。我们的对角化机器 D 会将图灵机 M_x 在 $\{1^n:h(i)<n\leqslant h(i+1)\}$ 中的一个输入上的答案翻转。D 的定义如下：

"在输入 x 上，若 $x\notin\{1\}^*$，则拒绝。如果 $x=1^n$，则计算 i 使得 $h(i)<n\leqslant h(i+1)$ 并且

1. 如果 $h(i)<n<h(i+1)$，则利用非确定性在 $n^{1.1}$ 时间内模拟 M_i 在输入 1^{n+1} 上的运行，并输出其答案。(如果 M_i 在指定时间内未停机，则停机并接受输入。)

2. 如果 $n=h(i+1)$，则接受 1^n 当且仅当 M_i 在 $(h(i)+1)^{1.1}$ 时间内拒绝 $1^{h(i)+1}$。"

第 2 种情形需要遍历 M_i 在输入 $1^{h(i)+1}$ 上的所有可能的 $2^{(h(i)+1)^{1.1}}$ 个分支。这没有问题，因为输入的大小 $h(i+1)$ 等于 $2^{h(i)^{1.2}}$。因此，非确定型图灵机 D 在 $O(n^{1.2})$ 时间内运行。

令 L 是 D 判定的语言。我们断言 $L\notin\mathbf{NTIME}(n)$。若不然，假设 L 是非确定型图灵机 M 在 cn 步骤内判定的语言，其中 c 是一个常数。由于每一个非确定型图灵机可以表示为无穷个位串，因此可以找到充分大的 i 使得 $M=M_i$，并且在任意长度 $n\geqslant h(i)$ 的输入上，M_i 可以在小于 $n^{1.1}$ 步内被模拟。这意味着，D 的定义中的两个条件确保了

$$\text{如果 } h(i)<n<h(i+1),\text{则 } D(1^n)=M_i(1^{n+1}) \tag{3.3}$$

$$\text{但是} \qquad D(1^{h(i+1)})\neq M_i(1^{h(i+1)}) \tag{3.4}$$

根据我们的假设，M_i 和 D 在 n 位于半开半闭区间 $(h(i),h(i+1)]$ 的任意输入 1^n 上应该相等。再由(3.3)式，这意味着 $D(1^{h(i+1)})=M_i(1^{h(i+1)})$，与(3.4)式矛盾。(参见图 3-1。) ■ 70

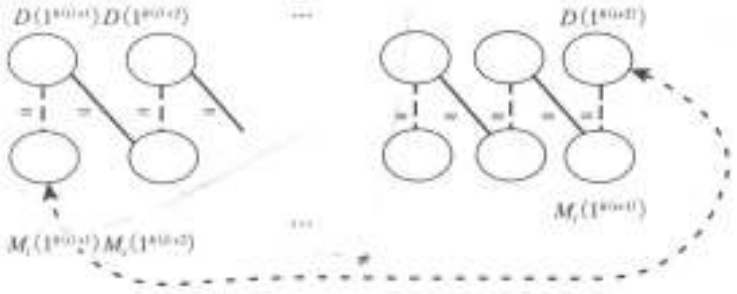


图 3-1　D 和 M_i 在输入 1^n 上的取值，其中 $n\in(h(i),h(i+1)]$。实线表示由 D 的定义导出的相等关系，虚线表示由"$D(x)=M_i(x)$ 对任意 x 成立"导出的相等关系，虚线箭头表示由 D 的定义导出的不等关系。联立所有这些关系就得出矛盾

3.3　拉德纳尔定理：NP 非完全问题的存在性

NP 完全性引人注目的一个原因是，大量 **NP** 问题均被证明是 **NP**-完全的，其中不乏曾被研究多年的问题，这完全出乎人们意料。由此产生了一个大胆的猜想：**NP** 中的任意问题要么属于 **P**，要么是 **NP**-完全的。如果 **P**=**NP**，则该猜想是平凡为真而毫无意义的。本节证明，如果(正如被广泛认为的)**P**≠**NP**，则上述猜想是错误的，因为存在语言 $L\in\mathbf{NP}\setminus\mathbf{P}$ 不是 **NP**-完全的。证明过程将会哥德尔式地定义一个语言 SAT_H，使得它"编码"了该语言自身的求解难度，这种有趣的定义方式使得本证明体现出对角线方法的特征。

定理 3.3　("**NP** 中间"语言[Lad75])　如果 **P**≠**NP**，则存在语言 $L\in\mathbf{NP}\setminus\mathbf{P}$ 不是 **NP**-完全的。

证明　对任意函数 $H:\mathbf{N}\to\mathbf{N}$，定义语言 SAT_H 由长度为 n 的可满足公式用 $n^{H(n)}$ 个 1 填充后得到的所有串构成，即 $\mathrm{SAT}_H=\{\varphi 01^{n^{H(n)}}:\varphi\in\mathrm{SAT}\text{ 且 }|\varphi|=n\}$。

定义一个函数 $H:\mathbf{N}\to\mathbf{N}$ 如下：

$H(n)$是满足下列两个条件的最小i值：(1)$i<\log\log n$；(2)对于满足$|x|\leqslant\log n$的任意$x\in\{0,1\}^*$，M_i在$i|x|^i$个步骤内输出$\mathrm{SAT}_H(x)$。[⊖]如果上述i不存在，则$H(n)=\log\log n$。

H是良定义的，因为$H(n)$恰好确定长度大于n的串在SAT_H中的成员资格，并且H的定义仅依赖于长度不超过$\log n$的串的具体情况。事实上，H的定义蕴含着一个由n计算$H(n)$的$O(n^3)$时间的递归算法(参见习题3.6)。[⊖]H的这种定义保证了下面的论断成立。

论断：$\mathrm{SAT}_H\in\mathbf{P}$当且仅当$H(n)=O(1)$(即存在常数$C$使得$H(n)\leqslant C$对任意$n$成
71 立)。而且，如果$\mathrm{SAT}_H\notin\mathbf{P}$，则$H(n)$随着$n$增大而趋于无穷。

论断的证明：

($\mathrm{SAT}_H\in\mathbf{P}\Rightarrow H(n)=O(1)$)　假设存在图灵机$M$在至多$cn^c$步骤内求解$\mathrm{SAT}_H$。由于表示$M$的位串有无穷个，故存在数$i>c$使得$M=M_i$。$H(n)$的定义表明，$H(n)\leqslant i$对$n>2^{2^i}$恒成立。因此，$H(n)=O(1)$。

($H(n)=O(1)\Rightarrow\mathrm{SAT}_H\in\mathbf{P}$)　如果$H(n)=O(1)$，则$H$的函数值只取有穷个值。因此，存在$i$使得$H$在无穷个$n$上均取值为$i$。这意味着图灵机$M_i$在$in^i$时间内求解了$\mathrm{SAT}_H$；否则，存在输入$x$使得$M_i$未在$in^i$时间内输出正确答案，则对任意$n>2^{|x|}$均有$H(n)\neq i$。注意，即使只假设只有某个常数$C$使得$H(n)\leqslant C$对有限多的$n$成立，该结论也成立，因此使论断得到完整的证明。

利用上述论断可以证明，如果$\mathbf{P}\neq\mathbf{NP}$，则$\mathrm{SAT}_H$既不属于$\mathbf{P}$也不是$\mathbf{NP}$-完全的。

- 假设$\mathrm{SAT}_H\in\mathbf{P}$。则由上述论断可知$H(n)\leqslant C$对某个常数$C$成立，这意味着$\mathrm{SAT}_H$就是SAT填充至多多项式(即$n^C$)个1得到的语言。这样，$\mathrm{SAT}_H$的多项式时间算法也能在多项式时间内求解SAT，这意味着$\mathbf{P}\neq\mathbf{NP}$。
- 假设SAT_H是$\mathbf{NP}$-完全的。这意味着存在从SAT到SAT_H的一个运行时间为$O(n^i)$的归约f，其中i是一个常数。由于SAT_H不属于$\mathbf{P}$，故上面的论断表明$H(n)$将随着n增大而趋于无穷。再由归约的运行时间为$O(n^i)$可知，对于充分大的n，该归约必然将SAT中长度为n的实例映射为SAT_H中长度小于$n^{H(n)}$的实例。因此，对于充分长的公式φ，归约f必然会将它映射为形如$\psi 01^{H(|\varphi|)}$的串，其中ψ的长度小于某个多项式因子，不妨假设小于$\sqrt[3]{n}$。但是，该归约将得到求解SAT问题的一个多项式时间递归算法，这与$\mathbf{P}\neq\mathbf{NP}$矛盾！(细节的完善留给读者，见习题3.6。) ■

虽然定理表明，如果$\mathbf{P}\neq\mathbf{NP}$，则$\mathbf{NP}\setminus\mathbf{P}$中存在一个非$\mathbf{NP}$-完全语言，但是这个语言似乎多少有些矫揉造作。定理的证明过程一直未被加强，以得到一个更自然的非$\mathbf{NP}$-完全语言。事实上，这种语言的候选者屈指可数，因为绝大多数比较自然的语言都已经借助巧妙的算法或归约得到了解决。两个例外的语言是因数语言和图同构语言(参见例2.3)。这两个语言目前均未找到多项式时间算法，同时也有很强的证据表明它们不是$\mathbf{NP}$-完全的(见第8章)。

⊖ 注意，M_i是二进制串i表示的图灵机，而$\mathrm{SAT}_H(x)$等于1当且仅当$x\in\mathrm{SAT}_H$。

⊖ 术语"递归算法"借用于标准的编程实践。在那里，如果一个程序在其他输入上调用自身，则称该程序被"递归"调用。

3.4　神喻机器和对角线方法的局限性

将对角线方法的局限性进行量化并不容易。你肯定已经看到，相比于 3.1 节中对角线方法的使用或 1.5 节中证明停机问题的非判定性时对对角线方法的使用，3.2 节和 3.3 节中对对角线方法的使用似乎要精妙得多。 72

为了使论述更加准确，我们把仅使用图灵机的如下性质的任何技术均称为对角线方法：

Ⅰ. 图灵机可以高效地表示成位串。

Ⅱ. 任意一个图灵机可以用较低的时间、空间开销模拟另一个任意的图灵机。

仅用到上述两个事实的任何论述均把图灵机视为黑盒，即不关心图灵机的内部如何工作。下面给出一个一般性技术来定义图灵机的两种变形，这两种变形均被称为神喻图灵机，它们满足上面的两条性质。但是，其中一种神喻图灵机使得 $\mathbf{P}=\mathbf{NP}$，而另一种神喻图灵机却使得 $\mathbf{P}\neq\mathbf{NP}$。这样就得到如下结论：解决 $\mathbf{P}\neq\mathbf{NP}$ 问题必须要用到性质Ⅰ和Ⅱ之外的其他性质。

神喻图灵机也是一种图灵机，它可以访问一个黑盒或"神喻"，其中神喻具有判定某个语言 $O\subseteq\{0,1\}^*$ 的魔力。具体地讲，神喻图灵机有一条称作神喻带的特殊工作带，它可以将位串 $q\in\{0,1\}^*$ 写在神喻带上，并且能够在一个步骤内获得"q 属于 Q 吗?"这种问题的答案。这种魔法般回答问题的过程可以对不同的问题重复任意次。如果 O 是一个难以求解的语言(比如说，多项式时间内无法判定的语言，甚至不可判定语言)，则神喻将会给图灵机赋予更强的计算能力。

定义 3.4（神喻图灵机）　神喻图灵机是一个图灵机 M，它有一条称作神喻带的特殊读/写带和三个特殊状态 q_{query}，q_{yes}，q_{no}。要运行图灵机 M，除了为 M 指定输入之外，还需为 M 指定一个用作神喻的语言 $O\subseteq\{0,1\}^*$。运行过程中，一旦 M 进入状态 q_{query}，机器便立刻根据神喻带上的内容 q 决定下一个状态：如果 $q\in O$ 则机器进入 q_{yes}；如果 $q\notin O$ 则机器进入 q_{no}。注意，无论 O 是何种语言，判定 q 是否属于 O 均仅用一个计算步骤。如果 M 是一个神喻图灵机，$O\subseteq\{0,1\}^*$ 是一个语言，$x\in\{0,1\}^*$，则将 M 以 O 为神喻在输入 x 上得到的输出记为 $M^O(x)$。

类似地，可以定义非确定型神喻图灵机。

定义 3.5　对任意 $O\subseteq\{0,1\}^*$，$\mathbf{P}^O$ 表示以 O 为神喻的确定型图灵机在多项式时间内判定的所有语言构成的集合。$\mathbf{NP}^O$ 表示以 O 为神喻的非确定型图灵机在多项式时间内判定的所有语言构成的集合。

例 3.6　为理解神喻图灵机的定义，我们证明下面的三个简单事实。

1. 令 $\overline{\mathrm{SAT}}$ 表示由所有不可满足的布尔公式构成的语言。则 $\overline{\mathrm{SAT}}\in\mathbf{P}^{\mathrm{SAT}}$。

事实上，给定神喻 SAT，要判定公式 φ 是否属于 $\overline{\mathrm{SAT}}$，多项式时间神喻图灵机可以向神喻查验"$\varphi\in\mathrm{SAT}$ 是否成立"，然后输出相反的答案。

2. 如果 $O\in\mathbf{P}$，则 $\mathbf{P}^O=\mathbf{P}$。

事实上，允许图灵机使用神喻将会使该图灵机能够计算更多的语言，因此 $\mathbf{P}\subseteq\mathbf{P}^O$。如果 $O\in\mathbf{P}$，则以 O 为神喻是多余的，因为任意以 O 为神喻的多项式时间图灵机均可以转换为一个(没有神喻的)标准图灵机，这只需将神喻的每次使用均替换为 O 语言的计算过程即可，因此 $\mathbf{P}^O\subseteq\mathbf{P}$。 73

3. 令 EXPCOM 是如下语言

$$\{\langle M,x,1^n\rangle : M \text{ 以 } x \text{ 为输入时在 } 2^n \text{ 步内输出 } 1\}$$

则 $\mathbf{P}^{\mathrm{EXPCOM}}=\mathbf{NP}^{\mathrm{EXPCOM}}=\mathbf{EXP}$。（注意，$\mathbf{EXP}=\bigcup \mathbf{DTIME}(2^{n^k})$。）

显然，以 EXPCOM 为神喻，可以在一个步骤内实现指数时间计算，因此 $\mathbf{EXP}\subseteq \mathbf{P}^{\mathrm{EXPCOM}}$。另一方面，如果 M 是一个多项式时间的非确定型神喻图灵机，则它以 EXPCOM 为神喻时执行的计算可以在指数时间内被模拟，因为指数时间允许我们枚举 M 的所有非确定型选择并回答 M 向神喻查询的所有问题。因此，$\mathbf{EXP}\subseteq \mathbf{P}^{\mathrm{EXPCOM}}\subseteq \mathbf{NP}^{\mathrm{EXPCOM}}\subseteq \mathbf{EXP}$。◀

神喻图灵机上的一个关键事实是：*无论 O 是何种语言*，以 O 为神喻的所有图灵机组成的集合均满足性质 **I** 和 **II**。这是由于，以 O 为神喻的图灵机可以表示成位串，并且通用图灵机（也以 O 为神喻）仍可以在这种位串上模拟神喻图灵机的运行。因此，图灵机上或复杂性类中任何仅用到性质 **I** 和 **II** 的结论在以 O 为神喻的图灵机上也都成立。这样的结论称为相对结论。本书中许多结论均是相对结论，特别地，定理 3.1、定理 3.2 和定理 3.3 均是相对结论。

下面的定理表明，无论 $\mathbf{P}=\mathbf{NP}$ 和 $\mathbf{P}\neq\mathbf{NP}$ 中哪个结论成立，它都不是相对结论。

定理 3.7（贝克(Baker)，吉尔(Gill)，索洛韦(Solovay)[BGS75]）*存在神喻 A 和 B 使得 $\mathbf{P}^A=\mathbf{NP}^A$ 和 $\mathbf{P}^B\neq\mathbf{NP}^B$。*

证明　令 A 是例 3.6 中的语言 EXPCOM。在例 3.6 中，已经证明了 $\mathbf{P}^A=\mathbf{NP}^A=\mathbf{EXP}$。

对任何语言 B，我们用 U_B 表示一元语言

$$U_B=\{1^n : B \text{ 中存在长度为 } n \text{ 的串}\}$$

显然，U_B 属于 $\mathbf{NP}^B$ 对任意神喻 B 成立，因为非确定型图灵机可以通过非确定型猜测从 $\{0,1\}^n$ 找出 x 使得 $x\in B$。下面构造一个神喻 B 使得 $U_B\notin\mathbf{P}^B$，这表明 $\mathbf{P}^B\neq\mathbf{NP}^B$。

构造 B

对任意 i，令 M_i 表示 i 的二进制形式表示的神喻图灵机。我们通过一系列的阶段来构造 B，其中第 i 个阶段确保 M_i 在 $2^n/10$ 的时间内无法判定 U_B。初始时令 B 是空集，然后逐渐向其中添加串。每个阶段确定有穷个串是否最终属于 B。

第 i 个阶段：目前，已经确定了有穷个串是否属于 B。选择充分大的 n 使得它大于任何已经考虑过的串的长度，然后让图灵机 M_i 在输入 1^n 上运行 $2^n/10$ 步。当图灵机 M_i 通过神喻确定已经考虑过的串是否属于 B 时，M_i 相应地进入状态 q_{yes} 或 q_{no}。当图灵机 M_i 通过
74 神喻确定未考虑过的串是否属于 B 时，M_i 进入状态 q_{no}（即，串不属于 B）。在 M_i 完成 1^n 上的计算后，无论 M_i 是否接受输入 1^n，我们让 M_i 在 1^n 上的输出是错误的。这可以做到，因为 $\{0,1\}^n$ 中的所有串原来均不在 B 中，而现在仅考虑了其中的至多 $2^n/10$ 个串是否属于 B。因此，如果 M_i 接受 1^n，则我们认为 $\{0,1\}^n$ 中长度为 n 的串均不在 B 中，这就确保了 $1^n\notin U_B$。反之，如果 M_i 拒绝 1^n，则选择一个 M_i 未通过神喻确定它是否属于 B 的长度为 n 的串 x（这样的串存在，因为 M_i 在 $2^n/10$ 个步骤内至多考虑过 $2^n/10$ 个串），并申明它属于 B，这样就确保了 $1^n\in U_B$。无论哪种情况，M_i 的输出均是错误的。由于在充分大的 n 上任意多项式 $p(n)$ 均小于 $2^n/10$ 并且每个图灵机 M 均可以表示为无穷个位串，因此上述构造使得 M 无法判定 U_B。于是，我们已经证明了 U_B 不属于 $\mathbf{P}^B$（事实上，上述过程同样表明 U_B 不属于 $\mathbf{DTIME}(f(n))$，其中 $f(n)=o(2^n)$ 是任意函数）。■

现在，我们来回答早先提出的问题：对角线方法或任意的模拟技术能解决 $\mathbf{P}$ 与 $\mathbf{NP}$ 问题吗？答案是，有可能，但它必须利用某种非相对事实（即图灵机的某种性质，它在神喻图灵机上不成立）。虽然复杂性理论中很多结果都是相对的，但同时也存在一些显著结果

不是相对的，例如 **IP**＝**PSPACE**(见第 8 章)和 **PCP** 定理(见第 11 章)。当然，目前仍不清楚如何用这些非相对技术来解决 **P** ≠ **NP** 问题！

评注 3.8　神喻图灵机在复杂性理论的很多地方都十分有用。例如，它们将在定理 5.12 和第 17 章中再次实几地出现。在第 17 章中，神喻将不再是语言(即，布尔函数)，而是一般的函数 $f:\{0,1\}^*\to\{0,1\}^*$。一般而言，神喻图灵机是如下这类算法的抽象：允许将其他函数作为黑盒例程来调用而不用关心函数的具体实现。

3.4.1　逻辑独立与相对

相对的概念是由数理逻辑中独立的概念引出的，独立的含义指的是某个自然的数学结论不能在某个特定的公理系统中得到证明或否证。两个著名的例子是，欧几里得第五公理独立于前四条公理(由此导致了非欧几何的发现)，连续统假设独立于策梅洛-弗兰克尔集合论。

由于所有的相对结论都能表明 **P**＝**NP** 不能用"已知的技术"证明或否证，因此这些相对结论也可以视为独立结论。然而，由于"已知的技术"这一提法太模糊，我们总感觉这种看法没有连续统假设的独立性那么准确。

阿罗拉(Arora)、因帕利亚佐(Impagliazzo)和瓦兹拉尼(Vazirani)[AIV93]曾尝试厘清上述问题。他们在论文中给出了一个公理系统，它类似于科巴姆在 1964 年对 **P** 的公理化刻画。该公理系统被证明恰好蕴含了关于 **P** 的相对结论，进而非相对结论就是那些不能由该公理系统证明的结论。

新的问题紧随而至：如何扩展该公理系统才能使它能够证明非相对结论呢？一种思想是，每发现一个非相对结论就将它作为新的公理添加到系统中。另一种保守一些的方法 75
是，找出能够蕴含其他已知非相对结论的一个非相对结论添加到系统中。出人意料的是，这样一个非相对结论确实存在，它本质上就是第 2 章讲过的库克-勒维定理。库克-勒维定理的证明过程用到了计算的局部性，亦即图灵机的每个基本操作仅读取和修改存储带上常数个位置上的符号。论文[AIV93]给出了几种方法来形式化地刻画计算的局部性，这些方法全都：(a)不是相对的(参见习题 3.7)；(b)通过公理系统，蕴含所有已知的非相对结论；(c)使得公理系统强到如下程度：如果 **P** ≠ **NP** 问题的任何一种情况能被证明，则类似有意义的结论也都可以用该公理系统证明。

当然，知道了计算的局部可验证性是解决 **P** ≠ **NP** 问题的关键，但并不意味着就知道了如何用它来解决 **P** ≠ **NP** 问题。这就好比，知道算术公理未必就知道如何证明费尔马大定理。

本章学习内容

- 对角线方法利用图灵机的位串表示来分离各种复杂性类。
- 利用对角线方法可以证明，如果允许图灵机更多地使用(时间、非确定性、空间中的)任何一种计算资源，则它必然能够求解更多的计算问题。同样，利用对角线方法还可以证明，如果 **NP**≠**P**，则 **NP** 中存在既不属于 **P** 又不是 **NP**-完全的问题。
- 仅依赖对角线方法证得的结论称为相对的。相对结论的含义是指，这种结论在以任意 $O\subseteq\{0,1\}^*$ 为神喻的图灵机上也成立。相对化方法可以用来证明对角线方法的局限性。特别地，单独使用相对化方法无法解决 **P** ≠ **NP** 问题。

本章注记和历史

19 世纪，格奥尔格·康托尔(Georg Cantor)发明了对角线方法，并用它证明了实数集

是不可数的。库尔特·哥德尔证明不完全定理时采用了类似的技术。计算机科学的本科学生在学习停机问题时往往才首次遇到对角线方法。

时间分层定理源自哈特马尼斯(Hartmanis)和斯特恩斯(Stearns)的开创性论文[HS65]。非确定型时间分层定理源于库克[Coo72]，虽然本章给出的简单证明实际上出自[Zak83]。类似的证明对其他复杂性类也成立，如下一章将要讨论的多项式分层。拉德纳尔定理源自[Lad75]，但本章采用的证明出自因帕利亚佐(Impagliazzo)未发表的手稿。我们仅证明了拉德纳尔定理的简化形式，完整的定理在$\mathbf{P}\neq\mathbf{NP}$的假设下给出了**P**和**NP**之间复杂性类的无穷分层，其中每个类均含于一个更大的类中，同时每个类中的问题均无法多项式时间归约到较小类中的任何问题上。**P**≠**NP**问题的相对性概念源自贝克(Baker)、吉尔(Gill)和索洛韦(Solovay)[BGS75]。

神喻图灵机的概念可以用来研究复杂性类之间的相互关系。事实上，库克[Coo71]用
76 神喻图灵机定义了**NP**完全性。结构复杂性是复杂性理论的子领域，其中详细研究了神喻图灵机和用神喻图灵机定义的各种复杂性类；关于这一专题的更多情况，参阅赫马斯潘德拉(Hemaspaandra)和获原(Ogihara)[HO02]。

贝克、吉尔和索洛韦的相对结论集中讨论线路的下界，他们旨在用这种方法来分离复杂性类，这种尝试进行了10年之后就陷入了困境。第23章形式化地论述了为什么已知的证明技术("自然的证明")很难用于证明有意义的线路下界。

习题中引入的高层性(superiority)术语本质上未在文献中出现过，但它与文献中研究过的免疫复杂性和几乎处处复杂性等概念密切相关。

习题

3.1　证明：下面的语言是不可判定的。

$$\{\langle M\rangle : M\text{是运行时间为}\ 100n^2+200\ \text{的图灵机}\}$$

3.2　证明：$\mathbf{SPACE}(n)\neq\mathbf{NP}$。(注意，人们目前并不清楚其中一个类是否包含了另一个类。)

3.3　证明：存在语言$B\in\mathbf{EXP}$使得$\mathbf{NP}^B\neq\mathbf{P}^B$。

3.4　称类C_1比类C_2高层，如果存在类C_1中的机器M_1使得对于C_2中的每个机器M_2和足够大的n，有一个长度介于n和n^2之间的输入使得M_1和M_2在该输入上的输出不同。

(a) $\mathbf{DTIME}(n^{1.1})$比$\mathbf{DTIME}(n)$高层吗？

(b) 为什么非确定型时间分层定理的证明过程不能证明$\mathbf{DTIME}(n^{1.1})$比$\mathbf{DTIME}(n)$高层？

3.5　证明：存在不是时间可构造的函数。

3.6　(a) 证明：定理3.3证明过程中构造的函数H是多项式时间可计算的。

(b) 令$H:\mathbf{N}\to\mathbf{N}$是满足$\lim_{n\to\infty}H(n)=\infty$的多项式时间可计算函数，$\mathrm{SAT}_H$是定理3.3证明过程中定义的语言。证明：如果$\mathrm{SAT}_H$是**NP**-完全的，则SAT属于**P**。

3.7　证明：存在神喻A和语言$L\in\mathbf{NP}^A$使得L不能多项式时间归约到3SAT，即使允许完成归约函数计算的图灵机使用神喻A。

3.8　假设我们用下述方式随机选择一个一元语言B：对于任意n，B不含长度为n的串的概率为1/2，B仅含长度为n的一个随机串的概率为1/2。证明：$\mathbf{P}^B\neq\mathbf{NP}^B$高概率地成立。(要给出结论的准确形式，需要用到基础测度论。)

77 3.9　假设任意串独立地以概率1/2出现在随机语言C中。证明：$\mathbf{P}^C\neq\mathbf{NP}^C$以高概率成立。

空间复杂性

我们的构造……还表明，"博弈"比"难题"(即，NP-完全问题)更难处理的原因在于博弈的每一步均在两名博弈者之间交替进行。

——西蒙·伊雯(Shimon Even)，罗伯特·塔吉安(Robert Tarjan)，1976

本章研究各种计算任务对存储空间的需求。为此，我们对图灵机执行计算的过程中使用的存储带单元的个数进行限制，由此定义空间受限计算的概念。我们定义空间受限的确定型图灵机和非确定型图灵机，并研究这两种图灵机能够求解的所有问题构成的复杂性类。4.2.1 和 4.3.2 节给出了空间受限的确定型图灵机和非确定型图灵机之间出人意料的关系。

同研究类 **NP** 一样，我们在各种空间复杂性类上定义完全问题，并找出这些类中具体而有意义的完全问题。我们将证明，多项式空间界限复杂性类的完全问题是象棋和围棋这样的全信息双人博弈中必胜策略的确定问题(参见 4.2.2 节)。正如开头伊雯和塔吉安的引言所述，确定这类博弈的必胜策略本质上与求解 SAT 这样的 **NP** 问题不同(甚至可能更难)。

我们还研究在亚线性空间中的计算。在这种计算中，算法所需的空间远小于问题的输入大小。亚线性空间计算十分有用，也很有趣。因此，4.1.2 节定义类 **L** 来表示用对数空间进行的亚线性空间计算，4.3 节研究它的非确定型形式。

4.1 空间受限计算的定义

首先，我们给出确定型空间受限计算和非确定型空间受限计算的形式定义(参见图 4-1)。

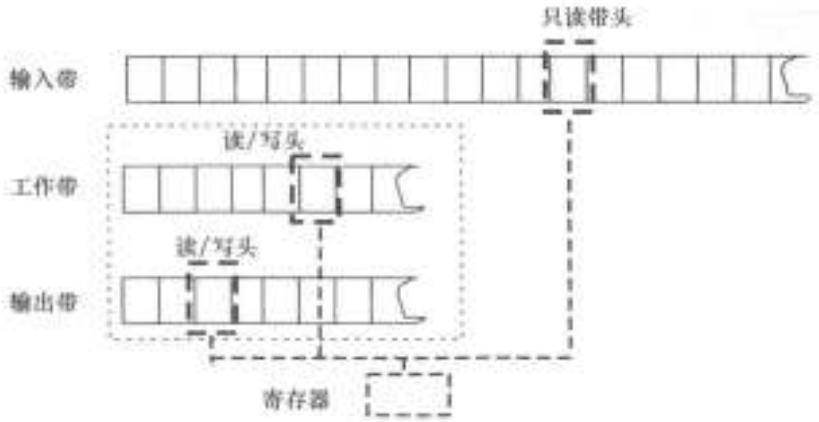


图 4-1 空间受限计算。只有读写带上使用的单元才计入空间界限中

定义 4.1 (空间受限计算) 设 $S: \mathbf{N} \to \mathbf{N}$ 且 $L \subseteq \{0,1\}^*$。如果存在常数 c 和判定 L 的图灵机 M，使得对任意长度为 n 的输入，M 完成计算的过程中 M 的带头至多只访问除输入带之外的各条工作带上的 $c \cdot S(n)$ 个存储单元，则称 $L \in \mathbf{SPACE}(S(n))$。

类似地，如果存在判定 L 的非确定型图灵机 M 使得对任意长度为 n 的输入，M 完成计算的过程中在任意非确定型选择上至多只使用 $c \cdot S(n)$ 个非空白存储单元，则称

$L \in \mathbf{NSPACE}(S(n))$。

事实上，本章讨论的所有图灵机均不使用输出带，因此，空间界限只作用在工作带上。类似于时间复杂性，空间界限仅使用空间可构造函数 $S: \mathbf{N} \to \mathbf{N}$，其中空间可构造指的是存在图灵机使得它在输入 x 上仅用 $O(S(|x|))$ 空间即可完成 $S(|x|)$ 的计算。从直觉上看，如果 S 是空间可构造的，则计算 S 的图灵机"知道"S 表示的空间界限。我们要研究的所有函数(包括 $\log n$、n 和 2^n)均是空间可构造函数。因此，定义中要求 S 为空间可构造函数是一个较宽松的条件。

由于图灵机的工作带不同于它的输入带，因此讨论所用空间小于输入长度(即 $S(n)<n$)的空间受限图灵机是有意义的。相比之下，在时间受限计算中，对 $T(n)<n$ 讨论 $\mathbf{DTIME}(T(n))$ 毫无意义，因为在这样的时间界限下任何图灵机连完整的输入都无法读完。我们要求 $S(n)>\log n$，这是由于工作带的长度为 n，图灵机的工作带至少应该有足够空间来表示输入带上的位置，以便它能够在工作带上"记住"图灵机正在读取输入带上的哪个存储单元。

$\mathbf{DTIME}(S(n)) \subseteq \mathbf{SPACE}(S(n))$，因为图灵机在每个步骤中只能访问带上一个存储单元。然而，一个 $\mathbf{SPACE}(S(n))$ 图灵机的运行步骤却可能远大于 $S(n)$，这是由于工作带上的每个存储单元均可以多次重写，亦即空间是可复用的。事实上，$S(n)$ 空间图灵机很容易运行 $2^{O(S(n))}$ 个步骤。例如，利用大小为 $S(n)$ 的工作带实现从 1 到 $2^{S(n)}-1$ 的计数的图灵机就是这样一个机器。上述界限是紧的，因为下面的定理表明，$\mathbf{SPACE}(S(n))$(甚至 $\mathbf{NSPACE}(S(n))$)中的任何语言均属于 $\mathbf{DTIME}(2^{O(S(n))})$，定理的证明见 4.1.1 节。奇怪的是，如果忽略对数因子的差异，则该定理是唯一已知的刻画空间受限计算的能力和时间受限计算的
79 能力之间关系的结论。该结论的改进将会成为重大的研究成果。

定理 4.2　对于任意空间可构造函数 $S: \mathbf{N} \to \mathbf{N}$ 有

$$\mathbf{DTIME}(S(n)) \subseteq \mathbf{SPACE}(S(n)) \subseteq \mathbf{NSPACE}(S(n)) \subseteq \mathbf{DTIME}(2^{O(S(n))}).$$

评注 4.3　对于非确定型空间受限图灵机的定义，有些教科书还要求非确定型图灵机无论在何种非确定型选择下均必须停机并输出一个答案。然而，在讨论 $\mathbf{NSPACE}(S(n))$ 时，只要 $S(n)$ 是空间可构造的，则上述要求是多余的，因为很容易修改所有非确定型图灵机使其在任意输入上均停机：只需让图灵机再使用一个计数器，并在执行 $2^{cS(n)}$ 个计算步骤之后立刻停机，其中 c 是一个适当大小的常数。

4.1.1　格局图

为了证明定理 4.2，我们在图灵机上定义格局图的概念。该概念对本章后面的讨论和本书后续部分均十分有用。令 M 是一个确定型或非确定型图灵机。图灵机 M 在其执行过程中的某个特定时刻上的格局由此时所有带上的非空白符号、机器的状态和所有带头的位置构成。对于任意 $S(n)$ 空间图灵机 M 和输入 $x \in \{0,1\}^*$，M 在输入 x 上的格局图是一个有向图，记为 $G_{M,x}$，其顶点对应在输入为 x 且所有工作带至多包含 $S(n)$ 个非空白存储单元的条件下 M 的所有可能的格局。如果从格局 C 开始，根据 M 的转移函数，通过一个计算步骤可以到达格局 C'，则从 C 到 C' 有一条有向边(参见图 4-2)。如果 M 是确定型图灵机，则图 G 的出度为 1。如果 M 是非确定型图灵机，则 G 的出度至多为 2。如果修改图灵机 M 使其停机之前擦除所有工作带上的内容，则可以假定图灵机有唯一格局 C_{accept} 使得 M 在该格局上停机并输出 1。这意味着，图灵机 M 接受输入 x 当且仅当 $G_{M,x}$ 中存在从 C_{start} 到 C_{accept}

的有向路径。关于格局图，下面的论断成立，其中第 2 个部分仅在下一个小节会用到。

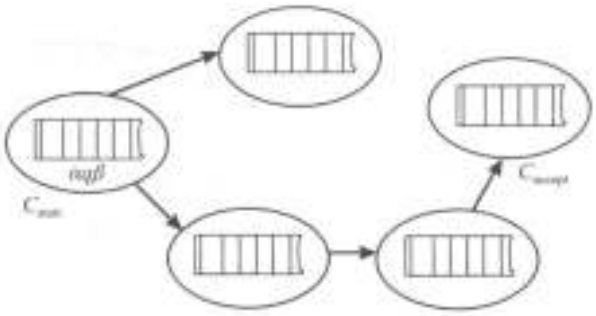


图 4-2　格局图 $G_{M,x}$ 是 M 在 x 上运行时所有可能格局上的一个图。如果在格局 C 上执行一步计算可以得到格局 C'，则从格局 C 到格局 C' 有一条有向边。如果 M 是确定型图灵机，则 G 的出度为 1。如果 M 是非确定型图灵机，则 G 的出度至多为 2

论断 4.4　设 $G_{M,x}$ 是 $S(n)$ 空间图灵机 M 在长度为 n 的输入 x 上的格局图，则　80

1. $G_{M,x}$ 的任意顶点均可以表示为长度是 $cS(n)$ 的位串，进而 $G_{M,x}$ 至多有 $2^{cS(n)}$ 个顶点，其中 c 是一个仅依赖于 M 的字母表大小和带的条数的常数。

2. 存在大小为 $O(S(n))$ 的合取范式公式 $\varphi_{M,x}$，使得对任意两个位串 C，C'，有 $\varphi_{M,x}(C, C')=1$ 当且仅当 C 和 C' 表示 $G_{M,x}$ 中两个相邻的格局。

证明概要　第 1 部分基于如下的观察结果。每个格局完全取决于所有工作带上的存储符号、每条带的带头所在位置和机器所处的状态(参见 1.2 节)。因此，每个格局可以如下编码：先编码快照(即当前的状态和所有带上当前读取的符号)，然后依次编码每条工作带上的所有非空白符并在恰当位置插入特殊的"标记符"以表明该带上带头所在的位置。

第 2 部分的证明采用库克-勒维定理(定理 2.10)证明中类似的思想。当时，我们证明了判定两个格局是否相邻可以表达成许多条件的 AND，每个条件均只依赖于常数个变量，并且由论断 2.13 可知每个条件均可以表达成具有常数大小的合取范式公式。变量的总数与所用空间的大小成正比。■

下面证明定理 4.2。

定理 4.2 的证明　显然，$\mathbf{DTIME}(S(n))\subseteq\mathbf{SPACE}(S(n))\subseteq\mathbf{NSPACE}(S(n))$，因此，只需证明 $\mathbf{NSPACE}(S(n))\subseteq\mathbf{DTIME}(2^{O(S(n))})$。通过枚举所有可能的格局，我们可以先在 $2^{O(S(n))}$ 时间内构造图 $G_{M,x}$，然后，用标准的广度优先搜索算法判断在 $G_{M,x}$ 中从 C_{start} 到 C_{accept} 是否连通，其中广度优先搜索算法的运行时间是图的大小的线性时间(参见[CLRS01])。■

4.1.2　一些空间复杂性类

我们对下面的复杂性类特别感兴趣。

定义 4.5

$$\mathbf{PSPACE}=\bigcup_{c>0}\mathbf{SPACE}(n^c)$$

$$\mathbf{NPSPACE}=\bigcup_{c>0}\mathbf{NSPACE}(n^c)$$

$$\mathbf{L}=\mathbf{SPACE}(\log n)$$

$$\mathbf{NL}=\mathbf{NSPACE}(\log n)$$

可以将空间复杂性类 **PSPACE** 和 **NPSPACE** 分别视为与时间复杂性类 **P** 和 **NP** 对应的

类。然而，由于时间界限小于输入长度时毫无意义，因此 **L** 和 **NL** 在时间复杂性类中没有与之对应的类。

例 4.6 本例证明 3SAT∈**PSPACE**。我们给出一个图灵机使得它在线性空间内判定 3SAT 问题，亦即它用 $O(n)$空间判定任意规模为 n 的布尔公式是否属于 3SAT。该图灵机
81 只用线性空间依次枚举 k 个变量的 2^k 个赋值，验证枚举的赋值是否满足输入的布尔公式。注意，一旦一个赋值被验证过，就可以将它从工作带上擦除；这样，工作带又可以用于验证枚举的下一个赋值。事实上，类似的思想可以用于枚举任意 **NP** 语言的所有可能的证明，因此 **NP**⊆**PSPACE**。◀

例 4.7 利用小学算术知识和对数空间图灵机有足够空间实现从 1 到 n 的计数这一事实，容易证明下面的两个语言属于 **L**：

$$\mathrm{EVEN}=\{x: x \text{ 的二进制表示有偶数个 } 1\}$$

$$\mathrm{MULT}=\{\langle n\rangle,\langle m\rangle,\langle mn\rangle : n\in \mathbf{N}\}$$

除了基本算术操作之外，似乎很难相信任何复杂的计算仅使用 $O(\log n)$空间。然而，人们目前仍不能否定 3SAT∈**L**。换句话说，“**NP**≠**L** 是否成立”仍是个未解决的问题(参见习题 4.6)。在空间 $S(n)\ll n$ 的条件下，空间受限计算似乎与网页爬取等计算问题有些相关。万维网可以大致看作一个有向图，图的顶点是所有网页而边是网页间的超链接。网页爬虫旨在探查整个图以获取各种信息。这种应用场景可以自然地刻画为如下的 PATH 问题：

$$\mathrm{PATH}=\{\langle G,s,t\rangle : \text{有向图 } G \text{ 中存在从 } s \text{ 到 } t \text{ 的路径}\} \tag{4.1}$$

我们证明，PATH∈**NL**。注意，如果 G 中存在从 s 到 t 的路径，则必然存在一条长度至多为 n 的路径，其中 n 是 G 的顶点个数。因此，非确定型图灵机可以采用始于 s 的“非确定型游走”，它总维护目前所处顶点的编号，并用非确定型性来选择目前所处顶点的一个相邻顶点作为“非确定型游走”下一步前进的方向。机器接受输入当且仅当非确定型游走在至多 n 个步骤内到达 t。如果非确定型游走已经前进了 n 步但仍未到达 t，则机器拒绝输入。在任何时刻，工作带上仅需记录非确定型游走前进的步数和该游走目前所处顶点的编号。因此，机器完成计算的过程中，仅使用工作带上的 $O(\log n)$个单元。

PATH 也属于 **L** 吗？这个问题仍未解决。我们很快将看到，该问题等价于问 **L**=**NL** 是否成立。也就是说，PATH 刻画了 **NL** 的“本质”，这就如同 3SAT 刻画了 **NP** 的“本质”一样。形式地说，我们将证明 PATH 是 **NL**-完全的。最近，一个出人意料的结果表明，PATH 问题限制在无向图上属于 **L**，参见第 7 章和第 20 章。◀

4.1.3　空间分层定理

类似于时间受限复杂性类，空间受限计算也有分层定理。

定理 4.8（空间分层定理[SHL65]）　*如果 f，g 是满足 $f(n)=o(g(n))$的空间可构造函数，则*

82
$$\mathbf{SPACE}(f(n))\subset \mathbf{SPACE}(g(n)) \tag{4.2}$$

证明过程完全类似于时间分层定理(定理 3.1)的证明。不同的是，我们可以使用使得空间效率下降常数因子的通用图灵机，因此不需要定理 3.1 中的对数因子。定理 4.8 的证明留作习题 4.1。

4.2　PSPACE 完全性

如前所见，目前仍不清楚 **P**=**PSPACE** 是否成立，但人们强烈地相信它不成立。这是

因为，由于 NP⊆PSPACE，故 P=PSPACE 蕴含 P=NP。注意，如果 L 可以多项式时间归约到 L'，则记 $L\leqslant_p L'$（参见定义 2.7）。下面，我们给出多项式空间归约的一些完全问题（你能注意到多项式空间归约毫无意义吗？）。

定义 4.9　如果对任意 $L\in$ PSPACE 均有 $L\leqslant_p L'$，则称语言 L' 是 PSPACE-难的。此外，如果还有 $L'\in$ PSPACE，则称 L' 是 PSPACE-完全的。

由第 2 章中给出的多项式时间归约的一些性质，我们可以看到，如果任意一个 PSPACE-完全语言属于 P，则 PSPACE 中所有其他语言也将属于 P。反之，如果 PSPACE≠P，则任意 PSPACE-完全语言不属于 P。因此，直觉上讲，PSPACE-完全语言是 PSPACE 中"最难"的问题。容易证明，下面的语言是 PSPACE-完全问题（参见习题 4.2）。

$$\text{SPACE TMSAT}=\{\langle M,w,1^n\rangle:\text{确定型图灵机 } M \text{ 在空间 } n \text{ 内接受输入 } w\} \tag{4.3}$$

下面，我们证明另一个更有意义的 PSPACE-完全问题，它要用到如下概念。

定义 4.10（量化布尔公式）　量化布尔公式（QBF）是形如 $Q_1x_1Q_2x_2\cdots Q_nx_n\varphi(x_1,x_2,\cdots,x_n)$ 的布尔公式，其中任意 Q_i 是量词∀或∃，变量 $x_1,\cdots,x_n$ 的值域为 $\{0,1\}$，φ 是一个纯布尔公式（即不含量词的布尔公式）。量词∀称为全称量词，表示"对所有的"，而量词∃称为存在量词，表示"存在"。

定义 4.10 只考虑前缀范式的量化布尔公式，亦即所有量词出现在纯布尔公式左侧的布尔公式。当然，也可以考虑将量词放置在公式其余位置的其他量化布尔公式。但是，利用恒等式 $\neg\forall x\phi(x)=\exists x\neg\phi(x)$ 和 $\phi\vee\exists x\varphi(x)=\exists x\phi\vee\varphi(x)$，其中 x 不在 ϕ 中出现，可以在多项式时间内将任意量化布尔公式转换成等价的前缀范式的量化布尔公式。与 SAT 问题和 3SAT 问题不同的是，我们不再要求纯布尔公式 φ 是 CNF 公式或 3CNF 公式。这种区别并不重要，因为采用与库克-勒维定理证明过程中类似的方法引入新的布尔变量，可以在多项式时间内将一般的量化布尔公式转换成纯布尔公式为 3CNF 的等价形式。

由于 QBF 中每个变量均被一个量词约束，因此 QBF 要么为真要么为假。通过下面的例子，你可以清楚地认识到这一点。

例 4.11　考虑公式 $\forall x\exists y(x\wedge y)\vee(\bar{x}\wedge\bar{y})$，其中∀和∃是全域 $\{0,1\}$ 上的量词。该公式可以用自然语言描述为"对于任意 $x\in\{0,1\}$，存在 $y\in\{0,1\}$ 等于 x"，它还可以非正式地表示为 $\forall x\exists y(x=y)$。该公式为真。但是，把其中第二个量词改为∀，将得到公式 $\forall x\forall y(x\wedge y)\vee(\bar{x}\wedge\bar{y})$，这个公式为假。注意，符号=和≠本身并不是标准的逻辑操作符，但在非正式场合可以用作公式的简写方法，以使公式更容易阅读，见例 2.12。◀

例 4.12　回顾一下，给定 n 个自由变量 $x_1,\cdots,x_n$ 上的一个布尔公式 φ，SAT 问题要求判定是否存在满足性赋值 $x_1,\cdots,x_n\in\{0,1\}^n$ 使得 $\varphi(x_1,\cdots,x_n)$ 为真。这可以等价地重述为，判定量化布尔公式 $\phi=\exists x_1,\cdots,x_n\varphi(x_1,\cdots,x_n)$ 是否为真。同时，还可以证明，公式 $Q_1x_1,\cdots,Q_nx_n\varphi(x_1,\cdots,x_n)$ 的否定形如 $Q'_1x_1,\cdots,Q'_nx_n\neg\varphi(x_1,\cdots,x_n)$，其中若 Q_i 是∀则 Q'_i 为∃，反之亦然。将语言 SAT 中的存在量词∃改为全称量词∀，则得到第 2 章引入的 coNP-完全语言 TAUTOLOGY。◀

我们定义 TQBF 是由所有为真的量化布尔公式构成的语言。

定理 4.13（SM[73]）　TQBF 是 PSPACE-完全的。

证明　首先证明 TQBF∈PSPACE。令

$$\phi=Q_1x_1\cdots Q_nx_n\varphi(x_1,\cdots,x_n) \tag{4.4}$$

是有 n 个变量的一个量化布尔公式，并将 φ 的规模记为 m。我们给出一个简单的递归算法

A 使其在 $O(m+n)$ 空间内判定 φ 的真伪。算法 A 能够处理更一般的情形，除了变量和变量的否定之外，它还允许常量 0（即"假"）和 1（即"真"）出现在 φ 中。如果 $n=0$，则 φ 中只有常量而没有变量，进而 φ 的真假可以在 $O(m)$ 空间和 $O(m)$ 时间内判定。因此，我们假设 $n>0$。对于 $b\in\{0,1\}$，用 $\varphi_{|x_1=b}$ 表示去掉 φ 中的第一个量词 Q_1 并将其中所有 x_1 替换为常数 b 之后得到的公式。算法 A 如下工作：如果 $Q_1=\exists$，则算法 A 输出 1 当且仅当 $A(\varphi_{|x_1=0})$ 和 $A(\varphi_{|x_1=1})$ 中至少有一个输出 1。如果 $Q_1=\forall$，则算法 A 输出 1 当且仅当 $A(\varphi_{|x_1=0})$ 和 $A(\varphi_{|x_1=1})$ 均输出 1。根据存在量词 $\exists$ 和全程量词 $\forall$ 的含义易知，算法 A 确实能对任意公式 φ 输出正确答案。

令 $S_{n,m}$ 是算法 A 输出参数为 n 和 m 的量化布尔公式的正确答案时所用空间的大小，其中 n 是布尔公式中变量的个数，m 是纯布尔公式的大小。关键在于运用了空间可以复用这个事实，即 $A(\varphi_{|x_1=0})$ 和 $A(\varphi_{|x_1=1})$ 这两个递归计算可以使用相同的空间。具体地说，
84 计算 $A(\varphi_{|x_1=0})$ 之后，算法 A 只需用 1 个位保存其输出，而其余空间可以被计算 $A(\varphi_{|x_1=1})$ 复用。因此，假设 A 递归调用时用 $O(m)$ 空间记录公式 $\varphi_{|x_1=b}$，则有 $S_{n,m}=S_{n-1,m}+O(m)$。由此可得 $S_{n,m}=O(n\cdot m)$。⊖

现在，我们证明 $L\leqslant_p$TQBF 对任意 $L\in$ **PSPACE** 成立。令 M 是在 $S(n)$ 空间内判定 L 的图灵机且 $x\in\{0,1\}^n$。下面，我们构造一个大小为 $O(S(n)^2)$ 的量化布尔公式使得它为真当且仅当 M 接受 x。令 $m=O(S(n))$ 是编码 M 在长度为 n 的输入上的格局所需的二进制位的个数。由论断 4.4 可知，存在布尔公式 $\varphi_{M,n}$ 使得对于任意两个串 $C,C'\in\{0,1\}^m$，$\varphi_M(C,C')=1$ 当且仅当 C 和 C' 编码图 $G_{M,n}$ 中相邻的两个格局。我们将用 $\varphi_{M,n}$ 来构造具有多项式规模的量化布尔公式 ψ，使得：(1)ψ 有多项式个受量词约束的变量和两个自由变量；(2)对任意 $C,C'\in\{0,1\}^m$，$\psi(C,C')$ 为真当且仅当格局图 $G_{M,n}$ 中存在从格局 C 到格局 C' 的有向路径。然后，将 C_{start} 和 C_{accept} 代入 ψ 得到一个量化布尔公式，该公式为真当且仅当 M 接受 x。

我们归纳地定义公式 ψ。令 $\psi_i(C,C')$ 为真当且仅当格局图 $G_{M,n}$ 中存在从格局 C 到格局 C' 的长度不超过 2^i 的路径。注意，$\psi_0=\varphi_{M,n}$ 且 $\psi_m=\psi$。构造过程的关键在于，从 C 到 C' 存在长度不超过 2^i 的路径当且仅当存在格局 C'' 使得从 C 到 C'' 存在长度不超过 2^{i-1} 的路径且从 C'' 到 C' 也存在长度不超过 2^{i-1} 的路径。这表明，ψ_i 可以定义为：

$$\psi_i(C,C')=\exists C''\psi_{i-1}(C,C'')\wedge\psi_{i-1}(C'',C')$$

然而，上述定义仍不够好，因为 ψ_i 的规模至少是 ψ_{i-1} 的大小的 2 倍，通过简单的归纳可知，ψ_m 的大小约为 2^m，它太大了。为了克服这一困难，我们引入新的量化变量来缩减公式的规模，将 $\psi(C,C')$ 更紧凑地定义为

$$\exists C''\forall D^1\forall D^2((D^1=C\wedge D^2=C'')\vee(D^1=C''\wedge D^2=C'))\Rightarrow\psi_{i-1}(D^1,D^2)$$

其中，同例 4.11 一样，为方便计，采用了 $=$ 和 $\Rightarrow$ 等简写方法，它们可以替换为标准布尔操作符的恰当组合。注意，$\mathrm{size}(\psi_i)=\mathrm{size}(\psi_{i-1})+O(m)$，进而 $\mathrm{size}(\psi_m)\leqslant O(m^2)$。可以证明，上述两种方式定义的 ψ_i 逻辑上是等价的，我们将这项工作留给读者。正如前面所说，

⊖ 上述分析能够证明 TQBF 属于 **PSPACE**，而且 A 实际上可以使用线性空间，即 $O(n+m)$ 空间。这是因为，A 的每次递归调用时其输入都是同一个公式 φ 的各种限制。因此，A 可以用一个全局数组来记录每个变量 x_i 的部分赋值，数组中对应于变量 x_i 的位置等于 0、1 或 q，其中 q 表示 x_i 仍是取值待定的量化变量。借助这个全局空间，算法 A 如下完成所有操作：在每次递归调用中，A 先根据全局空间确定第一个量化变量，将它赋值为 0 之后进行相应的递归调用，再将它赋值为 1 之后再进行相应的递归调用，最后再将该变量重新设置为量化变量 q。可以看到，A 使用的空间大小满足方程 $S_{n,m}=S_{n-1,m}+O(1)$，由此解得 $S_{n,m}=O(n+m)$。

可以在多项式时间内将最后得到的公式等价地转换为前缀形式。 ■

4.2.1 塞维奇定理

敏锐的读者或许已经发现，定理 4.13 的证明用到了格局图，但并不要求格局图的出度为 1。因此，证明过程实际上已经得到了更强的结论：TQBF 不仅是 **PSPACE**-难的，而且是 **NPSPACE**-难的。由于 TQBF $\in$ **PSPACE**，因此上述结论又意味着 **PSPACE** = **NPSPACE**，这与我们的直观认识大相径庭，因为在时间复杂性类上仍不能证明 **P** $\neq$ **NP**。 85
事实上，利用同样的思想，我们还可以证明下面的定理。

定理 4.14（塞维奇定理[Sav70]） 对任意满足 $S(n) \geqslant \log n$ 的空间可构造函数 S：**N**$\rightarrow$**N**，均有

$$\mathbf{NPSPACE}(S(n)) \subseteq \mathbf{SPACE}(S(n)^2)$$

证明 证明过程与定理 4.13 的很相似。设 $L \in \mathbf{NPSPACE}(S(n))$ 是由图灵机 M 判定的语言，满足：(1)对任意 $x \in \{0,1\}^*$，格局图 $G = G_{M,x}$ 至多有 $m = 2^{O(S(n))}$ 个顶点；(2)判定 $x \in L$ 是否成立等价于判定在格局图中从 C_{start} 到 C_{accept} 是否可达。下面，构造一个递归算法 REACH?(u, v, i) 使得它输出"YES"如果格局图中存在从 u 到 v 的一条长度至多为 2^i 的有向路径；否则，它输出"NO"。同样，构造过程的关键在于，从 u 到 v 存在长度不超过 2^i 的有向路径当且仅当存在 z 使得从 u 到 z 存在长度不超过 2^{i-1} 的有向路径，且从 z 到 v 也存在长度不超过 2^{i-1} 的有向路径。因此，在输入 u，v，i 上，算法 REACH? 使用 $O(\log m)$ 空间枚举所有可能的顶点 z。如果在某个 z 上，REACH?$(u, z, i-1)$="YES" 且 REACH?$(z, v, i-1)$="YES"，则输出"YES"。同样，尽管算法递归调用了 n 次，它可以在每一次递归调用中复用存储空间。因此，如果令 $S_{m,i}$ 是算法 REACH?(u, v, i) 在含有 m 个顶点的图上运行时的空间复杂性，则 $S_{m,i} = S_{m,i-1} + O(\log m)$，进而，$S_{m,\log m} = O(\log^2 m) = O(S(n)^2)$。由于从 C_{start} 到 C_{accept} 可达当且仅当从 C_{start} 到 C_{accept} 可以通过一条长度不超过 m 的路径可达，这就证得定理。 ■

特别指出，对于定理 4.14 证明过程得到的算法，它的运行时间高达 $2^{O(S(n)^2)}$。相比之下，定理 4.12 的证明过程中得到的算法则完全不同，它的时间复杂度上界为 $2^{O(S(n))}$。

4.2.2 PSPACE 的本质：最佳博弈策略

回顾一下，**NP**-完全问题的核心特征是问题的 yes 答案存在短证明(参见定义 2.1)。类似地，**PSPACE**-完全问题也有核心特征，它似乎对应于全信息双人博弈中必胜策略的特征。例如，象棋是一种典型的全信息双人博弈，该博弈中博弈双方交替在棋盘上移动棋子，棋子的每次移动对博弈双方均是可见的，这就是所谓的全信息。博弈的一方存在"必胜策略"是什么意思呢？博弈中(执黑先行的)黑方存在必胜策略当且仅当存在黑方第一步棋的一种下法使得对于白方第一步棋的任意下法均存在黑方第二步棋的一种下法使得……(依次类推)使得最后黑方获胜。要判定黑方是否存在"必胜策略"，似乎需要搜索由所有可能博弈策略构成的整棵博弈树。这使我们联想到 **NP**，它好像也必须完成指数搜索。但是，关键的区别是"黑方存在必胜策略"这种结论好像没有短"证明"，因为最好的"证明"莫过于必胜策略本身；然而正如所见，描述这种必胜策略却需要指数个二进制位。因此，**PSPACE**-完全问题的核心特征与 **NP**-完全问题的核心特征好像存在本质区别。

存在量词和全称量词在必胜策略的描述过程中交替出现，这种现象促使我们发明下面

86 的博弈。

例 4.15（QBF 博弈）　QBF 博弈的"棋盘"是一个以 x_1，x_2，…，x_{2n} 为自由变量的布尔公式 φ。博弈双方交替进行步骤，每个步骤依次为 x_1，x_2，…，中的一个变量赋值。因此，先手方依次为变量 x_1，x_3，x_5，…赋值，后手方依次为变量 x_2，x_4，x_6，…赋值。先手方获胜当且仅当 $\varphi(x_1, x_2, \cdots, x_{2n})$ 为真。

为使先手方存在必胜策略，先手方选择的方法必须能够战胜后手方后续的所有可能方法，也就是

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdots \forall x_{2n} \varphi(x_1, x_2, \cdots, x_{2n})$$

所表示的量化布尔公式取值为真。

因此，判定 QBF 博弈中给定棋盘上先手方是否存在必胜策略是一个 **PSPACE**-完全问题。◀

读到这里，读者很可能会联想到象棋、围棋和西洋跳棋。那么，复杂性理论能否帮助人们区分这三种博弈的难度呢？例如，人们普遍认为围棋比象棋难得多，复杂性理论能否解释这种直观感受呢？不幸的是，用渐进复杂性刻画这类问题却很棘手，因为渐进复杂性适于刻画无穷语言而这类博弈却是有穷对象，并且博弈中的必胜策略的存在性不外乎三种情况：先手方存在必胜策略，后手方存在必胜策略或二者均不存在必胜策略（平局）。然而，我们可以将博弈推广到 $n \times n$ 的棋盘上，其中 n 是任意大小的整数，当然这可能还需要扩展博弈规则，毕竟象棋博弈规则仅适用于 8×8 的棋盘。经过这种推广，我们就得到无穷的博弈场景，然后就可以证明：对于大多数常见的博弈（包括象棋），判定它在 $n \times n$ 棋盘上是否存在必胜策略是 **PSPACE**-完全问题（参见[Pap94]或[GJ79]）。因此，如果 **NP**≠**PSPACE**，则无论先手方还是后手方有必胜策略均不存在短证明。

证明博弈的 **PSPACE** 完全性本身似乎无足挂齿，然而类似的思想却证明了许多实践问题的 **PSPACE** 完全性。通常，在对手具备无限计算能力的情况下，各种代理终端的往复运动问题均是 **PSPACE**-完全问题。例如，机器人的许多计算问题均涉及机器人在变换场景下的游弋活动。在这样的应用中，可以将场景视为移动的，这样机器人的游弋就可以视为敌手采取的行动。在这种假设下，机器人的许多问题的求解均是 **PSPACE**-完全的。尽管有些研究者认为，将机器人所处的场景视为敌手过分消极、很不合适，然而，即使假设场景比较"温和"或"中立"而不会像敌手一样刻意采取行动，这类问题仍然是 **PSPACE**-完全的；参见章节注记中讨论与自然博弈的文献。

4.3　NL 完全性

现在，对于非确定型对数空间计算，我们给出其中的"本质"问题，即 **NL**-完全问题。

87 我们应该使用什么样的归约呢？在需要恰当选择归约类型来定义某个复杂性类的完全性时，必须深刻理解待刻画的复杂性现象。在这里，待刻画的复杂性现象是"**NL**=**L** 是否成立"。我们不能再使用多项式时间归约，因为 **L**⊆**NL**⊆**P**（参见习题 4.3）。我们选择的归约不应该比最弱的复杂性类 **L** 的描述能力还强。因此，我们选用对数空间归约。正如其称呼所表明的含义，对数空间归约指的是能够被确定型图灵机在对数空间内完成的归约。为给出确切定义，需要恰当地处理一个棘手的问题，即对数空间图灵机连写下输出的空间都不够。解决这个问题的办法是，只要求归约能够在对数空间内计算得到输出的任何一个位即可。换句话说，归约 f 在对数空间内是隐式可计算的：亦即，存在一个 $O(\log n)$ 空间图灵机能够在输入 $\langle x, i \rangle$ 上计算得到 $f(x)_i$，其中 $i \leqslant |f(x)|$。

定义 4.16（对数空间归约和 NL 完全性）　函数 $f:\{0,1\}^*\to\{0,1\}^*$ 称为隐式对数空间可计算的，如果 f 是多项式有界的（即 $|f(x)|\leqslant|x|^c$ 对某个常数 c 和任意 $x\in\{0,1\}^*$ 成立）并且语言 $L_f=\{\langle x,i\rangle\mid f(x)_i=1\}$ 和语言 $L'_f=\{\langle x,i\rangle\mid i\leqslant|f(x)|\}$ 均属于 **L**。

称语言 B 对数空间可归约到语言 C，记为 $B\leqslant_L C$，如果存在隐式对数空间可计算函数 $f:\{0,1\}^*\to\{0,1\}^*$ 使得 $x\in B$ 当且仅当 $f(x)\in C$ 对任意 $x\in\{0,1\}^*$ 成立。

如果语言 C 属于 **NL** 且 $B\leqslant_L C$ 对任意 $B\in$ **NL** 成立，则称 C 是 **NL**-完全的。

定义对数空间归约的另一种观点（有几本教材采用这种观点）是，归约图灵机有一条“一次性书写”输出带，机器每步要么在输出带上写下一个位要么将输出带带头右移，但不允许输出带带头左移或读取之前写下的任何一个位。容易证明，上述两种定义是等价的（参见习题 4.8）。

下面的引理表明，对数空间归约满足一般归约满足的性质。引理同时还表明，**NL**-完全语言属于 **L** 当且仅当 **NL**=**L**。

引理 4.17

1. 如果 $B\leqslant_L C$ 且 $C\leqslant_L D$，则 $B\leqslant_L D$。
2. 如果 $B\leqslant_L C$ 且 $C\in$ **L**，则 $B\in$ **L**。

证明　注证，如果 f，g 是两个隐式对数空间可计算函数，则它们的复合函数 $h(x)=g(f(x))$ 也是隐式对数空间可计算函数。这样，为证明引理第 1 部分，只需令 f 是从 B 到 C 的归约而 g 是从 C 到 D 的归约。同样，为证明引理第 2 部分，只需令 f 是从 B 到 C 的归约而 g 是 C 的特征函数（即 $g(y)=1$ 当且仅当 $y\in C$）。

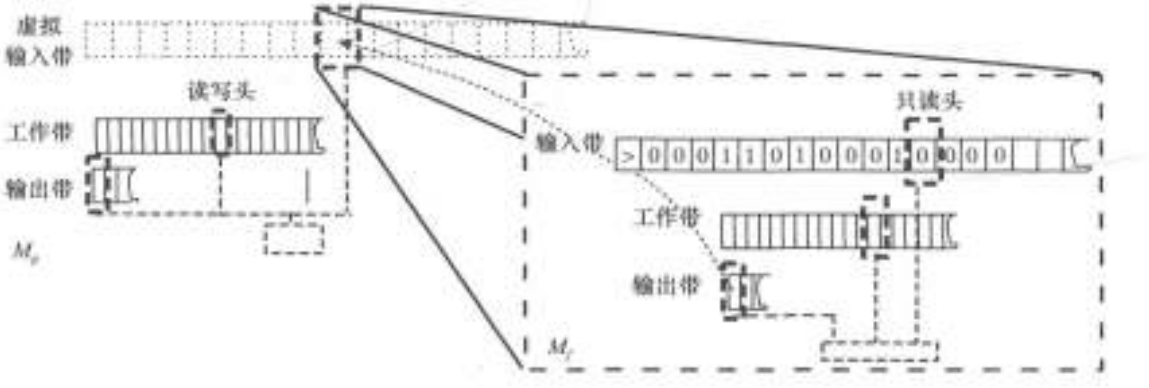


图 4-3　隐式对数空间可计算函数 f，g 的复合。M_g 通过调用 M_f 实现一条“虚拟输入带”。总共的空间开销是 M_f 的空间 $+M_g$ 的空间 $+O(\log|f(x)|)=O(\log|x|)$

令 M_f，M_g 分别是计算映射 $x,i\mapsto f(x)_i$ 和 $y,j\mapsto g(y)_j$ 的对数空间图灵机。我们构造一个图灵机 M_h 使得它在满足 $j\leqslant|g(f(x))|$ 的输入 x，j 上输出 $g(f(x))_j$。假装 M_h 有一条虚构的用于记录 $f(x)$ 的输入带，M_h 用这条带来模拟 M_g（参见图 4-3）。当然，真正的输入带用于记录 x，j。为确保用于记录 $f(x)$ 的输入带是虚构的，M_h 花费 $\log|f(x)|$ 的空间在工作带上记录整数 i，以表明 M_g 目前正在访问虚拟带上的哪个存储单元。为了完成一个计算步骤，M_g 必须知道该存储单元的内容，即 $f(x)_i$。此刻，M_h 暂停对 M_g 的模拟（将 M_g 工作带上的内容拷贝到 M_h 工作带上的安全位置），转而调用 M_f 在输入 x，i 上计算得到 $f(x)_i$，之后再利用 $f(x)_i$ 恢复对 M_g 的模拟。M_h 的总空间开销为 $O(\log|g(f(x))|)+$

$\log|x|+\log|f(x)|$[⊖]。由于 $f(x)\leqslant \mathrm{Poly}(x)$ 且 $g(y)\leqslant \mathrm{Poly}(y)$，因此上述表达式即为 $O(\log|x|)$。■

下面给出一个 **NL**-完全语言。回顾一下，4.1.2 节定义了语言 PATH，其中每个三元组 $\langle G, s, t\rangle$ 的含义是：有向图 G 中从 s 到 t 是可达的。

定理 4.18 PATH 是 **NL** 完全的。

证明 前面已经证明，PATH 属于 **NL**。令 L 是 **NL** 中的任意语言，M 是在 $O(\log n)$ 空间内判定 L 的非确定型图灵机。我们构造一个隐式对数空间可计算函数 f 将 L 归约到 PATH。对任意长度为 n 的输入 x，$f(x)$ 是格局图 $G_{M,x}$，它的顶点是图灵机 M 在输入 x 上所有可能的 $2^{O(\log n)}$ 个格局，其中开始格局记为 C_{start}，接受格局记为 C_{acc}。格局图 $G_{M,x}$ 存在从 C_{start} 到 C_{acc} 的有向路径当且仅当 M 接受 x。如果将每个格局依次用介于 0 到 $2^{O(\log n)}$ 的数字标识，则格局图 $f(x)$ 可以表示成邻接矩阵，其中第 C 行第 C' 列的位置 $\langle C, C'\rangle$ 等于 1 当且仅当 $G_{M,x}$ 中存在从 C 到 C' 的有向边。为完成定理证明，只需证明该邻接矩阵可以被一个对数空间归约计算。换句话说，仅需给出一个能够计算邻接矩阵的每个位的对数空间图灵机。这样的图灵机很容易构造，这是因为给定 $\langle C, C'\rangle$ 之后，确定型图灵机可以根据 M 的转移函数在 $O(|C|+|C'|)=O(\log|x|)$ 空间内检验 C 和 C' 是否表示 M 的两个格局以及从格局 C 出发仅用一个计算步骤能否到达格局 C'。■

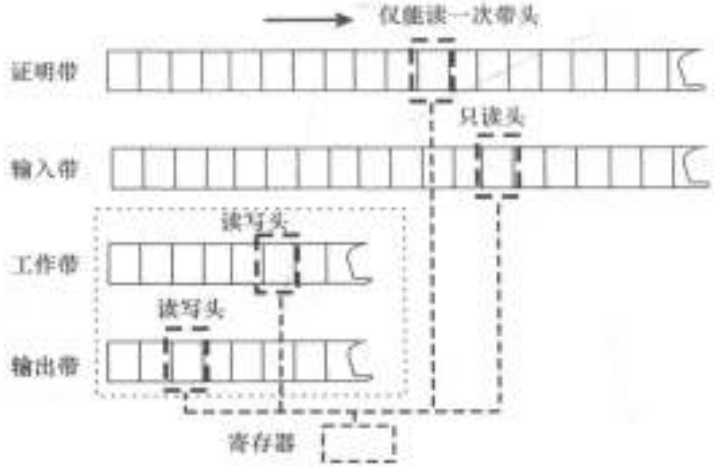


图 4-4　基于证明的 NL 定义。输入 x 的证明位于图灵机的一条特殊的"仅能读一次"的带上，该带的带头从不向左移动

4.3.1　基于证明的 NL 定义：仅能读一次的证明

89

第 2 章曾用成员资格的证明代替非确定性给出了 **NP** 的另一种定义。现在我们也想尝试用证明代替非确定型图灵机，给出 **NL** 的另一种定义。我们需要解决一个棘手的问题：证明具有多项式长度，因而对数空间图灵机没有足够的空间来存储这样的证明。因此，基于证明的 **NL** 定义需要假设：证明通过单独的"仅能读一次"的带提供给对数空间图灵机。

⊖ 原文是"$O(\log|g(f(x))|)+s(|x|)+s'(|f(x)|)$"，其中 s、s' 含义不清楚。译者联系上下文进行了修正，$\log|x|$ 是记录输入中当前访问位置所需的空间，$\log|f(x)|$ 是记录虚拟带上当前访问位置所需的空间。——译者注

这就是说，在图灵机的这条带上，带头只能从左向右移动，并且不能读取这条带上存储的证明中任何一个位两次。更明确地说，在图灵机的这条带上，带头在每个计算步骤中要么保持不动要么向右移动。不难看到，由于"证明的每个位仅能读一次"是计算过程中"非确定型选择"的另一种表述方式，因而 **NL** 还可以如下定义(参见图 4-4)。

定义 4.19 (**NL** 的另一种定义) 称语言 L 属于 **NL**，如果存在一个具有一条仅能读一次的特殊输入带的确定型图灵机 M(称为 L 的验证图灵机)和一个多项式 p：$\mathbf{N}\to\mathbf{N}$，使得对任意 $x\in\{0,1\}^*$ 均有

$$x\in L\Leftrightarrow\exists u\in\{0,1\}^{p(|x|)}\text{ 满足 }M(x,u)=1$$

其中 $M(x,u)$ 是 M 在下述条件下得到的输出：(1)x 放在 M 的输入带上；(2)u 放在 M 的仅能读一次的特殊输入带上；(3)M 在任意输入 y 上至多使用读写带上的 $O(\log|y|)$ 空间。

在定义 4.19 中，如果取消"仅能读一次"的限制条件，允许图灵机的带头在证明上往复移动，进而证明中的每个位可以被多次读取，则被定义的复杂性类是什么呢？出人意料的是，这种改动将由 **NL** 的定义得到 **NP** 的定义(参见习题 4.7)。

4.3.2 NL=coNL

与 **coNP** 的定义类似，我们将 **coNL** 定义为 **NL** 中各个语言的补集构成的集合。例如，PATH 的补集$\overline{\text{PATH}}$属于 **coNL**。判定语言$\overline{\text{PATH}}$的算法接受三元组(G,s,t)，如果图 G 中不存在从 s 到 t 的路径。不难看到，$\overline{\text{PATH}}$不仅属于 **coNL**，而且还是 **coNL**-完全的，也就是说，**coNL** 中的任意语言均可以对数空间归约到$\overline{\text{PATH}}$。与 PATH 不同的是，没有比较自然的证明能表明从 s 到 t 不存在路径。因此研究人员似乎显然可以得出$\overline{\text{PATH}}\notin\mathbf{NL}$。然而，这是错误的，因为 20 世纪 80 年代人们发现了下面的定理。 90

定理 4.20 (伊默曼-史泽勒普克森奈定理(Immerman-Szelepcsényi Theorem)[Imm88, Sze87]) $\overline{\text{PATH}}\in\mathbf{NL}$。

证明 由 4.3.1 节中基于证明的 **NL** 定义可知，只需构造一个 $O(\log n)$ 空间验证算法 A(即验证图灵机)使得对于任意 n 顶点图 G 和顶点 s，t，存在多项式证明 u 使得 $A(\langle G,s,t\rangle,u)=1$ 当且仅当 G 中从 s 到 t 不可达；其中 u 仅能被 A 读一次。为使论述简单计，下面假设 G 的所有顶点构成集合$\{1,\cdots,n\}$。

为完成定理的证明，我们站在证明构造者的立场。一旦证明 u 被构造出来，不难发现它能够被仅读一次的对数空间图灵机验证。

令 C_i 是由 G 中从 s 出发至多经 i 步可达的所有顶点构成的集合。C_i 的成员资格很容易验证，这个事实将在下面的构造中用到。对于任意 $i\in[n]$ 和顶点 v，v 属于 C_i 的证明为：一条从 s 到 v 的路径依次经过顶点序列 v_0，v_1，$\cdots$，v_k，其中 $k\leqslant i$。注意，该证明的长度确实至多为 n 的多项式。算法可以仅读一次来如下验证该证明：(1)验证 $v_0=s$；(2)对任意 $j>0$，验证从 v_{j-1} 到 v_j 有一条边；(3)$v_k=v$；(4)通过简单计数验证路径长度至多为 i。

下面，利用"C_i 成员资格的证明易于验证"这一事实，构造两种复杂的证明。

1. 假设验证算法已知(即已确知)C_i 的大小，构造顶点 v 不属于 C_i 的证明。

2. 假设验证算法已确知 C_{i-1} 的大小，对某个 c 构造 $|C_i|=c$ 的证明。

由于 $C_0=\{s\}$ 且验证算法也确知这一点，我们可以将第 2 类证明逐步提供给验证算法，使得它确知集合 C_1，$\cdots$，C_n 的大小。最后，由于 C_n 包含从 s 可达的所有顶点，并且验证算法也确知了它的大小，因此可以向验证算法提供第 1 类证明，使其确知 $t\notin C_n$。

给定$|C_i|$后验证v不属于C_i。我们构造的证明为：对于满足$u\in C_i$的所有顶点u，按u递增的顺序(回顾顶点是[n]中的数)依次列出$u\in C_i$的证明。验证算法如下工作：(1)验证每个证明均是有效的；(2)对于带有证明的顶点u，验证它确实大于前一个带有证明的顶点；(3)验证顶点v没有证明；(3)验证带有证明的顶点恰有$|C_i|$个。如果$v\notin C_i$，则验证算法将接受所构造的证明。另一方面，如果$v\in C_i$，则C_i将不会存在$|C_i|$个带有证明的顶点$u_1<u_2<\cdots<u_{|C_i|}$使得$u_j\neq v$对任意j成立。

给定$|C_{i-1}|$后验证v不属于C_i。在说明给定$|C_{i-1}|$后如何验证$|C_i|=c$之前，先说明在
91 给定$|C_{i-1}|$后如何验证v不属于C_i。这一验证过程与前一验证过程非常相似，我们构造的证明为：对于满足$u\in C_{i-1}$的$|C_{i-1}|$个顶点，按顶点u递增顺序依次列出$u\in C_{i-1}$的证明。验证过程执行上一验证过程中除第(3)项之外的所有验证工作，并将第3项验证工作改为：(3)验证顶点v及其相邻顶点均没有证明。由于$v\in C_i$当且仅当存在顶点$u\in C_{i-1}$使得$u=v$或u是v在图G中的相邻顶点，因此同前一验证过程一样，本验证过程也不会接受错误证明。

给定$|C_{i-1}|$后验证$|C_i|=c$。我们已经看到，对于任意顶点v，无论$v\in C_i$和$v\notin C_i$哪一个成立，我们均能给出相应的证明。因此，$|C_i|=c$的证明由n个顶点的证明按顶点递增顺序(从1到n)排列而成，其中顶点v的证明的具体形式依赖于$u\in C_i$还是$u\notin C_i$。验证算法需要验证所有证明，并对$u\in C_i$成立的所有顶点u计数。如果计数结果等于c，则验证算法接受证明。 ■

利用格局图的概念修改定理4.20的证明，可以证明如下推论(参见习题4.11)。

推论4.21 对任意空间可构造函数$S(n)\geq\log n$，有$\mathbf{NSPACE}(S(n))=\mathbf{coNSPACE}(S(n))$。

对空间受限复杂性的理解

下式列举了人们目前对各种空间受限复杂性类和各种时间受限复杂性类之间的关系的理解。

$$\mathbf{L}\subseteq\mathbf{NL}\subseteq\mathbf{P}\subseteq\mathbf{NP}\subseteq\mathbf{PSPACE}\subseteq\mathbf{EXP}$$

由于分层定理表明$\mathbf{L}\subset\mathbf{PSPACE}$和$\mathbf{P}\subset\mathbf{EXP}$，故上式中某些包含关系必然是严格的。然而，人们还不知道哪些包含关系是严格的。事实上，多数研究人员认为，所有的包含关系均是严格的。

本章学习内容

- 图灵机除了可用来对时间开销建模之外，也可以用来对空间使用情况建模。
- **PSPACE**是由所有可以在多项式空间内计算的语言构成的复杂性类，它包含**P**和**NP**，并且人们相信这两个包含关系均是严格的。类**PSPACE**中有一个完全问题TQBF，它是SAT问题的自然推广，刻画了象棋等博弈游戏中寻找最佳策略这类问题的复杂性。
- 存储空间通常是比时间更宝贵的计算资源，因此**L**和**NL**等低空间复杂性类具有重要意义。
- 时间复杂性上的某些结论(如分层定理)对空间复杂性类也成立。但是，空间复杂性有时也表现出完全不同的性质：非确定型空间在求补操作下封闭，并且用确定型图灵机模拟非确定型图灵机仅导致空间开销平方地增加。人们相信类似的结论对时间
92 复杂性不成立，因为他们相信$\mathbf{NP}\neq\mathbf{coNP}$并且$\mathbf{NP}\neq\mathbf{P}$。

本章注记和历史

空间复杂性的概念最早起源于 20 世纪 60 年代。特别地，斯特恩斯(Stearns)、哈特马尼斯(Hartnmanis)和刘易斯(Lewis)的空间分层定理[SHL65]和塞维奇定理[Sav70]均早于库克-勒维定理。斯托克迈耶(Stockmeyer)和迈耶(Meyer)[SM73]证明了 TQBF 问题的 **PSPACE** 完全性，紧随其后库克发表了他的论文。几年后，伊雯(Even)和塔吉安(Tarjan)建立了博弈和 **PSPACE** 完全性之间的联系，并证明了一款称为广义六角游戏的博弈游戏是 **PSPACE**-完全的。帕帕迪米特里奥(Papadimitriou)在[Pap94]这本书中详细列举了各种 **PSPACE**-完全问题。他还证明了几个与自然博弈的问题是 **PSPACE**-完全问题，其中与自然博弈的最初定义出现在[Pap85]中。不同于 TQBF 博弈中先手方使用存在量词而后手方使用全程量词的情况，在与自然博弈中后手方将随机地采取行动。与自然博弈旨在对对抗自然的博弈过程进行建模，其中"自然"不仅仅可以指天气等自然现象，它还可以指股票市场等个人命运大致表现出"中立"立场的大型系统。帕帕迪米特里奥利用这种博弈给出了 **PSPACE** 的其他特征。基于交互式证明的 **PSPACE** 的一个更强的特征将在本书第 8 章中进行讨论。

定理 4.2 中的平凡界限 **DTIME**($S(n)$)$\subseteq$**SPACE**($S(n)$)被霍普克罗夫特(Hopcroft)、保罗(Paul)和瓦里安特(Valiant)[HPV75]改进为 **DTIME**($S(n)$)$\subseteq$**SPACE**($S(n)/\log(S(n))$)。

伊默曼(Immerman)对定理 4.20 的证明构成了描述复杂性理论的一部分。描述复杂性是复杂性理论的一个子领域，它用数理逻辑的语言给出各种复杂性类的新特征以及与图灵机无关的特征。关于这一领域的综述，请参阅伊默曼的书[Imm99]。

习题

4.1　证明：空间受限计算存在通用图灵机(类似于定理 1.9 给出的确定型通用图灵机)。即，证明存在图灵机$\mathcal{U}$使得对于任意串 α 和输入 x，如果由 α 表示的图灵机 M_α 以 x 为输入时在使用工作带上第 t 个存储单元之前停机，则$\mathcal{U}(\alpha, t, x)=M_\alpha(x)$，并且$\mathcal{U}$至多使用所有工作带上的 Ct 个存储单元，其中 C 是仅依赖于 M_α 的常数。(尽管这里给出的界限优于定理 1.9 中的界限，但本结论的证明实际上比定理 1.9 的证明简单。)利用通用图灵机证明定理 4.8。

4.2　证明：(4.3)式定义的语言 SPACETM 是 **PSPACE**-完全的。

4.3　证明：在多项式时间卡普归约下，非空且非$\{0,1\}^*$的任意语言 L 均是 **NL**-完全的。

4.4　证明：语言$\{\langle G\rangle : G$ 是强连通图$\}$是 **NL**-完全的。

4.5　证明：2SAT 属于 **NL**。

4.6　假设我们用对数空间归约代替多项式时间归约来定义 **NP** 完全性。利用库克-勒维定理的证明过程，证明 SAT 和 3SAT 在新定义下仍是 **NP**-完全的。由此得出结论：SAT$\in$**L** 当且仅当 **NP**=**L**。 93

4.7　证明：在基于证明的 **NL** 定义中(4.3.1 节)，如果允许验证图灵机的带头在证明上往复移动，则定义的类将变成 **NP**。

4.8　定义函数 $f:\{0,1\}^*\to\{0,1\}^*$ 是一次写入对数空间可计算的，如果它可以被输出带"只能一次写入"的 $O(\log n)$空间图灵机 M 计算，其中输出带只能一次写入的含义

是 M 在每个计算步骤中要么将输出带带头保持不动，要么在输出带上写出一个符号后将带头右移。输出带上使用的存储单元不计入 M 的空间开销中。

证明：函数 f 是一次写入对数空间可计算的当且仅当它在定义 4.16 的意义下是隐式对数空间可计算的。

4.9　证明：TQBF 在对数空间归约下仍是 **PSPACE**-完全的。

4.10　证明：在任意有穷全信息双人博弈中，先手方和后手方必有一方存在必胜策略。有穷指的是，有一个事先已知的游戏回合数的上界 n，游戏进行 n 个回合之后必然终止且一方胜出，不会出现平局。

4.11　证明推论 4.21。

4.12　定义 **polyL** 为 $\bigcup_{c>0}$ **SPACE**$(\log^c n)$。史蒂夫类 SC（为纪念史蒂夫·库克(Steve Cook)而得名）定义为由可以用确定型图灵机在多项式时间和 $\log^c n$ 空间（c 是大于 0 的常数）内计算的所有语言构成的集合。

PATH$\in$SC 是否成立仍是一个未解决的问题。为什么塞维奇定理未解决该问题？

94　SC 与 **polyL**$\cap$**P** 相同吗？

多项式分层和交错

线路设计本身极其困难。更难的是，证明所设计的线路是实现函数最经济的方式。这种困难源于线路设计时有大量本质不同的线路网络可供选用。

——克劳德·香农(Claude Shannon)，1949

我们已经采用了几种方法，通过证明某些计算问题是复杂性类的完全问题来"刻画"各种自然的复杂性类的本质。本章将继续采用这种方法来研究另一类自然的计算问题，这些计算问题包括在香农的上述引言中提到的计算问题，并且这些问题的本质不能单独用非确定性来刻画。确切地说，刻画这些计算问题的复杂性类称为*多项式分层*，记为 **PH**，它是 **P**、**NP** 及 **coNP** 等类的推广。**PH** 包含无穷个子类，每个子类也称为一个层，每个层本身也是重要的复杂性类。人们猜想 **PH** 的各个子类各不相同。该猜想是 **NP**≠**P** 这一猜想的更强形式，它不时地出现在复杂性理论的许多研究中，包括本书第 6 章、第 7 章和第 17 章。

本章用三种等价的方式定义多项式分层。

1. 在 5.2 节中，我们将多项式分层定义为常数个全称量词(∀)和存在量词(∃)交错出现的多项式时间谓词(predicate)构成的语言集合。这种定义是第 2 章 **NP** 和 **coNP** 定义的推广。

2. 在 5.3 节中，我们用交错图灵机等价地刻画多项式分层。交错图灵机是 2.1.2 节定义的非确定型图灵机的推广。

3. 在 5.5 节中，我们证明多项式分层还可以由 3.4 节给出的*神谕图灵机*来定义。

多项式分层的第四种刻画将用到布尔线路的概念。布尔线路是大家不太熟悉的内容，因此我们将这部分内容放到第 6 章。在 5.4 节中，我们利用多项式分层的不同特征证明一个有意义的结论：SAT 问题不能同时在线性时间和对数空间内求解。这一结论代表了 **P** ≠ **NP**问题目前研究的前沿方法。

5.1 类 Σ_2^p

为给出研究 **PH** 的动机，先讨论一些不能用 **NP** 完全性来刻画的计算问题。

作为准备工作，回顾一下 **NP** 问题 INDSET(参见例 2.2)，该问题确实存在验证成员资格的短证明。

$$\text{INDSET} = \{\langle G,k\rangle : \text{图 } G \text{ 存在大小} \geqslant k \text{ 的独立集}\}$$

对 INDSET 问题稍加修改，即要求确定图中的最大独立集。将修改后的问题重述为判定问题的形式，得到

$$\text{EXACT INDSET} = \{\langle G,k\rangle : G \text{ 的最大独立集的大小恰好为 } k\}$$

现在，EXACT INDSET 问题似乎不再存在判定成员资格的短证明。这是由于，$\langle G, k\rangle \in$ EXACT INDSET 当且仅当 G 的最大独立集有 k 个顶点并且 G 的其他任意独立集至多有 k 个顶点。

类似地，考虑香农引言中提到的问题：亦即，确定与给定的布尔公式等价的最小布尔公式。为方便计，我们将该问题表述为判定问题。

MIN-EQ-DNF = {⟨φ,k⟩:存在规模 ≤ k 的析取范式公式 ψ 等价于析取范式公式 φ}

其中析取范式公式是将或操作(OR)应用到若干个仅含与操作(AND)的公式上得到的布尔公式。如果两个布尔公式在变量的所有可能的赋值上取相同的值，则称这两个布尔公式等价。语言 MIN-EQ-DNF 的补集即是香农引言中提到的问题。不同之处在于，香农考虑与一般布尔公式等价的最小布尔公式，而不仅仅考虑与析取范式公式等价的最小公式。

$$\overline{\text{MIN-EQ-DNF}} = \{\langle\varphi,k\rangle:\forall \text{ 规模} \leqslant k \text{ 的析取范式公式 } \psi,\ \exists \text{ 赋值 } u \text{ 使得 } \varphi(u) \neq \psi(u)\}$$

同样，MIN-EQ-DNF 也不存在明显的成员资格证明。因此，为了刻画 EXACT INDSET 和 MIN-EQ-DNF 等语言，我们不仅(像定义 2.1 中定义 **NP** 那样)需要单个的"存在"量词(∃)或(像定义 2.20 中定义 **coNP** 那样)需要单个的"全程"量词(∀)，而且还需要将这两种量词组合使用。由此得到如下的定义。

定义 5.1　对于语言 L，如果存在多项式时间图灵机 M 和多项式 q 使得对任意 $x\in\{0,1\}^*$ 均有

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)} M(x,u,v) = 1$$

则称 $L\in\boldsymbol{\Sigma}_2^{\mathrm{p}}$；亦即，类 $\boldsymbol{\Sigma}_2^{\mathrm{p}}$ 是由符合上述条件的所有语言 L 构成的集合。

96　注意，$\boldsymbol{\Sigma}_2^{\mathrm{p}}$ 包含了 **NP** 和 **coNP** 两个类。

例 5.2　语言 EXACT INDSET$\in\boldsymbol{\Sigma}_2^{\mathrm{p}}$，这是因为，如前所述，序对⟨$G$，$k$⟩属于 EXACT INDSET 当且仅当存在 G 的大小为 k 的顶点子集 S，使得对于任意 G 的大小为 $k+1$ 的顶点子集 S' 均有：S 是 G 的独立集但 S' 不是 G 的独立集。(习题 5.9 将 EXACT INDSET 放入了更精细的复杂性类。)

语言 MIN-EQ-DNF 也属于 $\boldsymbol{\Sigma}_2^{\mathrm{p}}$，因为序对⟨$\varphi$，$k$⟩属于 MIN-EQ-DNF 当且仅当存在析取范式公式 ψ 使得对于任意赋值 u，有 $\varphi(u)=\psi(u)$。已经证明，MIN-EQ-DNF 是 $\boldsymbol{\Sigma}_2^{\mathrm{p}}$-完全的[Uma98]。◀

5.2　多项式分层

多项式分层的定义是 **NP**，**coNP** 和 $\boldsymbol{\Sigma}_2^{\mathrm{p}}$ 的定义的推广。它包含如下定义的所有语言：每个语言均是由多项式时间可计算的谓词通过常数个量词∀/∃组合在一起得到的串构成的。

定义 5.3　(多项式分层)　对 $i>1$，称语言 L 属于 $\boldsymbol{\Sigma}_i^{\mathrm{p}}$，如果存在多项式时间图灵机 M 和多项式 q，使得对任意 $x\in\{0,1\}^*$ 均有

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)}\ \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} M(x,u_1,u_2,\cdots,u_i) = 1$$

其中 Q_i 在 i 分别为偶数和奇数的情况下相应地取∀和∃。

多项式分层指的是集合 $\mathbf{PH}=\bigcup_i \boldsymbol{\Sigma}_i^{\mathrm{p}}$。

注意，$\boldsymbol{\Sigma}_1^{\mathrm{p}}=\mathbf{NP}$。对任意 i，定义 $\boldsymbol{\Pi}_i^{\mathrm{p}}=\mathbf{co}\boldsymbol{\Sigma}_i^{\mathrm{p}}=\{\overline{L}:\ L\in\boldsymbol{\Sigma}_i^{\mathrm{p}}\}$。因此，$\boldsymbol{\Pi}_1^{\mathrm{p}}=\mathbf{coNP}$。同时还可以注意到，对任意 i 均有 $\boldsymbol{\Sigma}_i^{\mathrm{p}}\subseteq\boldsymbol{\Pi}_{i+1}^{\mathrm{p}}\subseteq\boldsymbol{\Sigma}_{i+2}^{\mathrm{p}}$，进而 $\mathbf{PH}=\bigcup_{i>0}\boldsymbol{\Pi}_i^{\mathrm{p}}$。

5.2.1　多项式分层的性质

我们相信 $\mathbf{P}\neq\mathbf{NP}$ 和 $\mathbf{NP}\neq\mathbf{coNP}$。这两个猜想可以推广为另一个更吸引人的猜想：对于任意 i，$\boldsymbol{\Sigma}_i^{\mathrm{p}}$ 均是 $\boldsymbol{\Sigma}_{i+1}^{\mathrm{p}}$ 的严格子集。该猜想在复杂性理论中经常使用，它常被表述为"多项

式分层不坍塌”，其中多项式分层坍塌指的是存在某个 i 使得 $\mathbf{\Sigma}_i^p=\mathbf{\Sigma}_{i+1}^p$。

下面将会看到，$\mathbf{\Sigma}_i^p=\mathbf{\Sigma}_{i+1}^p$ 意味着 $\mathbf{\Sigma}_i^p=\bigcup_{j\geqslant 1}\mathbf{\Sigma}_j^p=\mathbf{PH}$。此时，我们称多项式分层坍塌到第 i 层。i 越小，猜想“PH 不坍塌到第 i 层”就越弱，因而该猜想就越可信。

定理 5.4

1. 对任意 $i\geqslant 1$，如果 $\mathbf{\Sigma}_i^p=\mathbf{\Pi}_i^p$，则 $\mathbf{PH}=\mathbf{\Sigma}_i^p$；亦即，多项式分层坍塌到第 i 层。

2. 如果 $\mathbf{P}=\mathbf{NP}$，则 $\mathbf{PH}=\mathbf{P}$；亦即，多项式分层坍塌到 P。

97

证明　我们仅证明第 2 部分，第 1 部分的证明与此类似，将它留作习题 5.12。假设 $\mathbf{P}=\mathbf{NP}$，我们对 i 用归纳法证明 $\mathbf{\Sigma}_i^p$，$\mathbf{\Pi}_i^p\subseteq\mathbf{P}$。显然，$i=1$ 时结论成立，因为 $\mathbf{\Sigma}_1^p=\mathbf{NP}$ 且 $\mathbf{\Pi}_1^p=\mathbf{coNP}$。设结论对 $i-1$ 成立，只需证 $\mathbf{\Sigma}_i^p\subseteq\mathbf{P}$。这是由于，$\mathbf{\Pi}_i^p$ 是由 $\mathbf{\Sigma}_i^p$ 中各个语言的补集构成的集合，且 P 在求补操作下是封闭的，因此由 $\mathbf{\Sigma}_i^p\subseteq\mathbf{P}$ 即得 $\mathbf{\Pi}_i^p\subseteq\mathbf{P}$。

令 $L\in\mathbf{\Sigma}_i^p$。根据定义 5.3，存在多项式时间图灵机 M 和多项式 q 使得对任意 $x\in\{0,1\}^*$ 均有

$$x\in L\Leftrightarrow\exists u_1\in\{0,1\}^{q(|x|)}\,\forall u_2\in\{0,1\}^{q(|x|)}\cdots Q_i u_i\in\{0,1\}^{q(|x|)}M(x,u_1,u_2,\cdots,u_i)=1 \tag{5.1}$$

其中 Q_i 在 i 分别为偶数和奇数的情况下相应地取 $\forall$ 和 $\exists$。如下定义语言 L'：

$$\langle x,u_1\rangle\in L'\Leftrightarrow\forall u_2\in\{0,1\}^{q(|x|)}\cdots Q_i u_i\in\{0,1\}^{q(|x|)}M(x,u_1,u_2,\cdots,u_i)=1$$

显然，$L'\in\mathbf{\Pi}_{i-1}^p$。因此，由归纳假设可知，$L'$ 属于 P。这意味着，存在多项式时间图灵机 M' 计算语言 L'。将 M' 代入(5.1)式，得到

$$x\in L\Leftrightarrow\exists u_1\in\{0,1\}^{q(|x|)}M'(x,u_1)=1$$

这意味着 $L\in\mathbf{NP}$。因而根据假设 $\mathbf{P}=\mathbf{NP}$ 可得，$L\in\mathbf{P}$。 ■

5.2.2　PH 各层的完全问题

回顾一下，称语言 B 可以多项式时间卡普归约到语言 C，记为 $B\leqslant_p C$，如果存在一个多项式时间可计算函数 f：$\{0,1\}^*\to\{0,1\}^*$ 使得 $x\in B\Leftrightarrow f(x)\in C$ 对任意 $x\in\{0,1\}^*$ 成立(参见定义 2.7)。如果语言 $L\in\mathbf{\Sigma}_i^p$ 且 $L'\leqslant_p L$ 对任意 $L'\in\mathbf{\Sigma}_i^p$ 成立，则称 L 是 $\mathbf{\Sigma}_i^p$-完全的。用同样的方法，我们还可以定义 $\mathbf{\Pi}_i^p$-完全性和 PH-完全性。本小节证明，对于任意 $i\in\mathbf{N}$，$\mathbf{\Sigma}_i^p$ 和 $\mathbf{\Pi}_i^p$ 均存在完全问题。相反，人们相信多项式分层本身却不存在完全问题，正如下面简单的论断所述。

论断 5.5　*如果存在语言 L 是 PH-完全的，则存在 i 使得 $\mathbf{PH}=\mathbf{\Sigma}_i^p$(进而多项式分层坍塌到第 i 层)。*

证明概要　由于 $L\in\mathbf{PH}=\bigcup\mathbf{\Sigma}_i^p$，因此存在 i 使得 $L\in\mathbf{\Sigma}_i^p$。由于 L 是 PH-完全的，因而 PH 中任意语言均可以归约到 L。然而，任何可以多项式时间归约到 $\mathbf{\Sigma}_i^p$ 中某个语言的语言它本身也属于 $\mathbf{\Sigma}_i^p$，因此，$\mathbf{PH}\subseteq\mathbf{\Sigma}_i^p$。 ■

不难看到，同 NP 和 coNP 一样，PH 也含于 PSPACE 中。论断 5.5 的一个简单推论是：$\mathbf{PH}\neq\mathbf{PSPACE}$，除非多项式分层坍塌。事实上，否则的话，4.2 节定义的 PSPACE-完全问题 TQBF 将是 PH-完全的。

98

例 5.6　(PH 各层的完全问题)对任意 $i\geqslant 1$，类 $\mathbf{\Sigma}_i^p$ 的一个完全语言是由如下限定交错次数的量化布尔表达式构成的语言：

$$\mathbf{\Sigma}_i\mathrm{SAT}=\exists u_1\,\forall u_2\,\exists\cdots Q_i u_i\varphi(u_1,u_2,\cdots,u_i)=1 \tag{5.2}$$

其中 φ 是一个布尔公式(它不必是合取范式，尽管布尔公式的形式不影响语言的定义)，每个 u_i 均是布尔变量构成的向量，Q 在 i 分别为偶数和奇数的情况下相应地取 $\forall$ 和 $\exists$。注意，对每个 i，$\mathbf{\Sigma}_i$SAT 均是 4.2 节给出的 TQBF 问题的特例。习题 5.1 要求证明，$\mathbf{\Sigma}_i$SAT 确实是 $\mathbf{\Sigma}_i^p$-完全的。我们可以类似地定义问题 $\mathbf{\Pi}_i$SAT，它是 $\mathbf{\Pi}_i^p$-完全的。

在 SUCCINCT SET COVER 问题中，给定由 n 个布尔变量的 3DNF 公式构成的集合 $S=\{\varphi_1, \varphi_2, \cdots, \varphi_m\}$ 和整数 k，要求判定是否存在大小至多为 k 的子集 $S'\subseteq\{1, 2, \cdots, m\}$ 使得 $\vee_{i\in S'}\varphi_i$ 是一个永真公式(在变量的任意赋值上均取真)。根据定义，SUCCINCT SET COVER 显然属于 $\mathbf{\Sigma}_2^p$。乌曼斯(Umans)证明了该语言是 $\mathbf{\Sigma}_2^p$-完全的[Uma98]。 ◀

5.3　交错图灵机

交错图灵机(ATM)是非确定型图灵机的推广。回顾一下，即使非确定型图灵机不是一个现实的计算模型，研究非确定型图灵机也仍有助于理解一个自然的计算现象，即猜测答案和验证答案之间的明显区别。然而，单用非确定性不足以刻画明显不存在短证明的那些语言，此时交错图灵机替代了非确定型图灵机，发挥着类似的作用。

同非确定型图灵机类似，交错图灵机也有两个状态转移函数，它在每个计算步骤中均可以任意选用其中一个。不同的是，交错图灵机除 q_{accept} 和 q_{rah} 之外的每个内部状态要么被标记为 $\exists$，要么被标记为 $\forall$。回顾一下，非确定型图灵机接受输入，如果存在一个非确定型选择序列使得机器进入状态 q_{accept}。对于交错图灵机而言，每个非确定型选择的存在量词需要根据机器的当前状态，替换成该状态下恰当的量词。于是，交错图灵机能够在标记为 $\exists$ 的状态和标记为 $\forall$ 的状态之间相互切换，名字中的"交错"指的就是上述事实。

定义 5.7（交错时间）　对任意函数 $T: \mathbf{N}\to\mathbf{N}$，如果在任意输入 $x\in\{0, 1\}^*$ 和任意可能的转移函数选用序列上，交错图灵机 M 至多运行 $T(|x|)$ 个步骤后停机，则称 M 在 $T(n)$ 时间内运行。

称语言 L 属于 **ATIME**$(T(n))$，如果存在常数 c 和一个 $c\cdot T(n)$ 时间的交错图灵机 M，使得对任意 $x\in\{0, 1\}^*$，M 接受 x 当且仅当 $x\in L$。M 接受输入 x 定义如下：

回顾一下，$G_{M,x}$ 表示 M 在输入 x 上的(无环有向的)格局图，其中从格局 C 到格局 C' 有一条边当且仅当在格局 C 上运用一步 M 的转移函数可以得到格局 C'(参见 4.1.1 节)。
99
我们反复运用下列规则，将 $G_{M,x}$ 的一些顶点标记为"ACCEPT"，直到不能再运用任何规则为止：

- 表示机器处于状态 q_{accept} 的格局 C_{accept} 被标记为"ACCEPT"。
- 如果格局 C 表示机器处于标记为 $\exists$ 的状态，并且从 C 出发有一条边到达被标记为"ACCEPT"的格局 C'，则将 C 标记为"ACCEPT"。
- 如果格局 C 表示机器处于标记为 $\forall$ 的状态，并且从 C 出发在一步内到达被标记为"ACCEPT"的格局 C' 和 C''，则将 C 标记为"ACCEPT"。

如果上述标记过程结束后，格局 C_{start} 被标记为"ACCEPT"，则称 M 接受 x。

我们也对交错图灵机的交错次数做出限制。

定义 5.8　对任意 $i\in\mathbf{N}$，定义 $\mathbf{\Sigma}_i$**TIME**$(T(n))$ 和 $\mathbf{\Pi}_i$**TIME**$(T(n))$ 分别是由初始状态标记为 $\exists$ 和初始状态标记为 $\forall$ 的如下交错图灵机 M 在 $T(n)$ 时间内接受的语言构成的集合：在任意输入和格局图中任意从初始状态出发的(有向)路径上，M 在标记为 $\exists$ 的状态和标记为 $\forall$ 的状态之间至多交错 i 次。

下述论断的证明留作习题 5.2。

论断 5.9　对任意 $i\in\mathbf{N}$，$\Sigma_i^p=\bigcup\Sigma\mathbf{TIME}(n^i)$ 且 $\Pi_i^p=\bigcup\Pi\mathbf{TIME}(n^i)$。

5.3.1　无限次交错

定义 5.8 用交错次数受限的交错图灵机定义 **PH**，它要求图灵机的交错次数是独立于输入大小的常数。现在，我们再回头讨论交错次数无限制的多项式时间交错图灵机。令 $\mathbf{AP}=\bigcup\mathbf{ATIME}(n^i)$，则下面的定理成立。

定理 5.10　$\mathbf{AP}=\mathbf{PSPACE}$。

证明概要　$\mathbf{PSPACE}\subseteq\mathbf{AP}$，这是因为 TQBF 显然属于 **AP**（用标记为 $\exists$ 的状态来“猜测”受存在量词限定的变量的值，用标记为 $\forall$ 的状态来“猜测”受全称量词限定的变量的值，然后再用确定型多项式计算来验证猜测），并且 **PSPACE** 中的任意语言均可以归约到 TQBF。为了证明 $\mathbf{AP}\subseteq\mathbf{PSPACE}$，可以采用一个递归算法，类似于证明 $\mathrm{TQBF}\in\mathbf{PSPACE}$ 时采用的算法（见习题 5.5）。■

类似地，我们还可以考虑在多项式空间内运行的交错图灵机，这种图灵机接受的语言构成的集合称为 **APSPACE**。习题 5.7 要求证明，$\mathbf{APSPACE}=\mathbf{EXP}$。类似地，对数空间交错图灵机接受的语言构成的集合等于 **P**。

100

5.4　时间与交错：SAT 的时空平衡

尽管人们普遍认为，SAT 问题必须用指数（或者超多项式）时间才能求解，或者必须用线性（或者超对数）空间才能求解，但是，我们目前无法证明这两个猜想。从目前来看，SAT 问题确实有可能存在对数空间算法和线性时间的算法，但绝不可能存在既是对数空间又是线性时间的算法。事实上，我们可以证明下面更强的定理。

定理 5.11　（SAT 的时空平衡[For97a，FLvMV00]）　对于任意两个函数 S，T：$\mathbf{N}\to\mathbf{N}$，令 $\mathbf{TISP}(T(n),S(n))$ 是图灵机 M 在任意输入 x 上至多使用 $O(T(|x|))$ 个步骤和读写带上的至多 $O(S(|x|))$ 个存储单元所能判定的语言构成的集合，则 $\mathrm{SAT}\notin\mathbf{TISP}(n^{1.1},n^{0.1})$。

绝大多数教材用具有随机访问存储器（RAM）的图灵机定义类 $\mathbf{TISP}(T(n),S(n))$，其中具有随机访问存储器的图灵机指的是可以随机访问存储带的图灵机（这种图灵机可以定义为类似于 3.4 节中的神喻图灵机）。定理 5.11 对具有随机访问存储器的图灵机也成立。在此，需要说明的是，在这两种计算模型上均存在更强的结论（参见习题 5.6 和本章注记）。

证明　我们先证明

$$\mathbf{NTIME}(n)\subsetneq\mathbf{TISP}(n^{1.2},n^{0.2}) \tag{5.3}$$

在此基础上，沿用证明库克-勒维定理（定理 2.10）的思想，可以得出本定理的结论。这是由于，仔细分析库克-勒维定理的证明过程可以得到一个归约，它将判定某个 $\mathbf{NTIME}(T(n))$ 语言的成员资格归约到判定大小为 $O(T(n)\log T(n))$ 的布尔公式是否可满足，并且该归约输出的每个位均可以在多项式时间和多项式空间内被计算出来。（参见习题 4.6 和本书定理 6.15 的证明中类似的分析。）进而，如果 $\mathrm{SAT}\in\mathbf{TISP}(n^{1.1},n^{0.1})$，则 $\mathbf{NTIME}(n)\subseteq\mathbf{TISP}(n^{1.1}\mathrm{polylog}(n),n^{0.1}\mathrm{polylog}(n))$。下面证明（5.3）式，证明的主要过程是如下的论断，它表明了如何将时间替换为交错。

论断 5.11.1　$\mathbf{TISP}(n^{12}, n^2) \subseteq \mathbf{\Sigma}_2(n^6)$。

论断 5.11.1 的证明　证明过程类似于塞维奇定理的证明或 TQBF 的 **PSPACE** 完全性的证明(参见定理 4.14 和定理 4.13)。假设 L 是图灵机 M 在 n^{12} 时间和 n^2 空间内判定的语言。对任意 $x \in \{0,1\}^*$，考虑 M 在输入 x 上的格局图 $G_{M,x}$，其中每个格局均可以表示为长为 $O(n^2)$ 的位串；并且，$x \in L$ 当且仅当 $G_{M,x}$ 中存在从初始格局 C_{start} 到一个接受格局的长度为 n^{12} 的路径。这样一条路径存在当且仅当存在 n^6 个格局 $C_1, \cdots, C_{n^6}$(需要用 $O(n^6)$ 个位来表示)，使得如果 $C_0 = C_{start}$ 且 C_{n^6} 是一个接受格局，则对任意 $i \in [n^6]$ 而言从格局 C_{i-1} 到格局 C_i 需经过 n^6 个计算步骤。由于上述条件可以在 $O(n^7)$ 时间内验证，因而得到判
101 定语言 L 的成员资格的一个 $O(n^6)$ 时间 $\mathbf{\Sigma}_2$ 图灵机。■

接下来证明，假设(5.3)式不成立(因而 $\mathbf{NIMTE}(n) \subseteq \mathbf{TISP}(n^{1.2}, n^{0.2}) \subseteq \mathbf{DIMTE}(n^{1.2})$，则交错也可以替换为时间。

论断 5.11.2　假设 $\mathbf{NIMTE}(n) \subseteq \mathbf{DIMTE}(n^{1.2})$，则 $\mathbf{\Sigma}_2\mathbf{TIME}(n^6) \subseteq \mathbf{NIMTE}(n^{5.4})$。

论断 5.11.2 的证明　利用交错时间与多项式分层之间的等价性(参见论断 5.9)可知，L 属于 $\mathbf{\Sigma}_2\mathbf{TIME}(n^6)$ 当且仅当存在图灵机 M 使得

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{c|x|^6} \forall v \in \{0,1\}^{d|x|^6} M(x,u,v) = 1$$

对某两个常数 c，d 成立，其中 M 的运行时间为 $O(|x|^6)$。然而，如果 $\mathbf{NIMTE}(n) \subseteq \mathbf{DIMTE}(n^{1.2})$，则(同定理 2.22 的证明一样)用填充技术可以得到一个确定型算法 D，它在满足 $|x| = n$ 和 $|u| = cn^6$ 的输入 x，u 上的运行时间为 $O((n^6)^{1.2}) = O(n^{7.2})$，并且其输出为 1 当且仅当存在 $v \in \{0,1\}^{dn^6}$ 使得 $M(x,u,v) = 0$。于是，

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{c|x|^6} D(x,u) = 0$$

这意味着 $L \in \mathbf{NIMTE}(n^{7.2})$。■

论断 5.11.1 和论断 5.11.2 表明，在假设 $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^{1.2}, n^{0.2})$ 下将产生矛盾。事实上，该假设和填充技术一起将得到 $\mathbf{NTIME}(n^{10}) \subseteq \mathbf{TISP}(n^{12}, n^2)$。根据论断 5.11.1，这表明 $\mathbf{NTIME}(n^{10}) \subseteq \mathbf{\Sigma}_2\mathbf{TIME}(n^6)$。再由论断 5.11.2 可知，这意味着 $\mathbf{NTIME}(n^{10}) \subseteq \mathbf{NTIME}(n^{8.4})$。这与非确定型时间分层定理(定理 3.2)矛盾。■

5.5　用神喻图灵机定义多项式分层

回顾 3.4 节定义的神喻图灵机，它可以通过访问一条特殊带来获得"x 属于 Q 吗?"这种问题的答案，其中 Q 是某个语言。对于任意 $O \subseteq \{0,1\}^*$，神喻图灵机 M 和输入 x，用 $M^O(x)$ 表示 M 以 O 为神喻时在输入 x 上的输出。多项式分层具有如下性质。

定理 5.12　对任意 $i \geqslant 2$，$\mathbf{\Sigma}_i^p = \mathbf{NP}^{\mathbf{\Sigma}_{i-1}^{SAT}}$，其中 $\mathbf{NP}^{\mathbf{\Sigma}_{i-1}^{SAT}}$ 是以 $\mathbf{\Sigma}_{i-1}$SAT 为神喻的非确定型图灵机在多项式时间内判定的语言构成的集合。

证明　我们仅通过证明 $\mathbf{\Sigma}_2^p = \mathbf{NP}^{SAT}$ 来展示定理证明的基本思想。假设 $L \in \mathbf{\Sigma}_2^p$，则存在多项式时间图灵机 M 和多项式 q，使得

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x,u_1,u_2) = 1$$

然而，对每个固定的 u_1 和 x，断言"对任意 u_2，$M(x, u_1, u_2) = 1$"是对一个 **NP** 断言的否
102 定，因此该断言可以用 SAT 神喻来判定。于是，存在以 SAT 为神喻的非确定型图灵机 N 来判定语言 L：在输入 x 上，N 非确定地猜想 u_1，然后用神喻判定 $\forall u_2 M(x, u_1, u_2) = 1$ 是否成立。容易看到，$x \in L$ 当且仅当存在 u_1 使得 N 接受 x。

另一方面，假设 L 是一个以 SAT 为神喻的多项式时间非确定型图灵机 N 判定的语言。N 使用 SAT 神喻的次数可以是多项式次，并且当前向神喻提出的问题可以基于之前提过的所有问题。初看起来，这似乎使得 N 的计算能力强于 Σ_2^p 图灵机，因为如前面所讲，Σ_2^p 仅能够以非确定的方式向 **coNP** 语言提出一个问题。要将 N 替换为一个等价的 Σ_2^p 图灵机，我们的主要思想是：先猜测所有需要向 SAT 神喻查验的问题及其答案，然后向一个 **coNP** 语言神喻问一个问题使得该问题的答案能够验证所有猜测的正确性。

更确切地说，x 属于 L 当且仅当存在一系列的非确定型选择和一系列正确的神喻答案使得 N 接受 x。也就是说，存在一系列形如 $c_1, c_2, \cdots, c_m \in \{0, 1\}$ 的非确定型选择和神喻对问题的答案 $a_1, \cdots, a_k \in \{0, 1\}$，使得如果 N 在输入 x 上运行时的非确定型猜测是 $c_1, c_2, \cdots, c_m$，并且向 SAT 神喻提出的第 i 个问题的答案是 a_i，则(1)N 最后进入状态 q_{accept}；(2)所有的答案均是正确的。令 φ_i 表示 N(在输入 x 上采用非确定型猜测 $c_1, c_2, \cdots, c_m$ 且收到答案 $a_1, \cdots, a_k$ 的运行过程中)向 SAT 神喻提出的第 i 个问题。于是，上面的条件(2)可以重述为，且如果 $a_i=1$，则对存在赋值 u_i 有 $\varphi_i(u_i)=1$；如果 $a_i=0$，则任意赋值 v_i 使得 $\varphi_i(v_i)=0$。由此，我们得到

$$x \in L \Leftrightarrow \exists c_1, c_2, \cdots, c_m, a_1, \cdots, a_k, u_1, \cdots, u_k \forall v_1, \cdots, v_k \text{ 使得}$$
$$N \text{ 采用猜测 } c_1, c_2, \cdots, c_m \text{ 和答案 } a_1, \cdots, a_k \text{ 时接受 } x, \text{ 并且}$$
$$\forall i \in [k], \text{如果 } a_i = 1 \text{ 则 } \varphi_i(u_i) = 1, \text{并且}$$
$$\forall i \in [k], \text{如果 } a_i = 0 \text{ 则 } \varphi_i(v_i) = 0$$

这就表明 $L \in \Sigma_2^p$。■

由于以复杂性类中的完全语言为神喻的图灵机可以判定该复杂性类中的所有语言，因此，在神喻记号中，有些教科书采用类的名字来代替完全语言。例如，$\Sigma_2^p = \mathbf{NP}^{SAT}$ 被表示为 $\mathbf{NP}^{NP}$，而 Σ_3^p 则被表示为 $\mathbf{NP}^{NP^{NP}}$，等等。

本章学习内容

- 多项式分层是语言构成的一个集合，其中的语言既可以用存在量词和全称量词交错的常数次数来定义，也可以等价地用交错图灵机和神喻图灵机来定义。多项式分层中，有一些自然的问题还不清楚是否属于 **NP**。
- 人们猜想：多项式分层不坍塌；亦即，多项式分层中每个层均不同于它的前一个层。
- 利用交错的概念，我们证明了 SAT 不能同时在线性时间和对数空间内求解。 103

本章注记和历史

卡普在其开创性论文[Kar72]中提到"如果 **P**=**NP**，则克林(Kleene)算术分层的多项式有界形式是平凡的"，这一结论似乎早就预示了定理 5.4。对多项式分层的定义和仔细研究始于迈耶(Meyer)和斯托克迈耶(Stockmeyer)[MS72]，他们不仅证明了，若 $\Sigma_i^p = \Pi_i^p$，则 $\mathbf{PH} = \Sigma_i^p$，还证明了 Σ_iSAT 在多项式分层第 i 层中的完全性。

习题 5.9 中提到的类 **DP** 是帕帕迪米特里奥(Papadimitriou)和杨纳卡卡斯(Yannakakis)[PY82]定义的，他们用它来刻画识别多面体各个面的复杂性。

PH 各层的完全问题目前远不如 **NP**-完全问题那样丰富。谢弗(Schaefer)和乌曼斯(Umans)的综述[SU2a, SU02b]给出了一系列有意义的完全问题。SUCCINCT SET

COVER 问题出自乌曼斯[Uma98]。

SAT 问题的时空下界最初由福特劳(Fortnow)证明[For97a]，后来又被逐步改进[FLvMV00，Wil05，DvM05]，目前最新的结论属于威廉姆斯(Williams)[Wil07]，他证明了 SAT$\notin$**TISP**(n^c，$n^{o(1)}$)，其中 $c<2\cos(\pi/7)=1.801$ 是任意常数。所有这些工作均受到 1983 年坎纳安(Kannan)的论文[Kan83]中的证明技术的启发。

习题

5.1　证明：在多项式时间归约下，(5.2)式定义的语言 Σ_iSAT 是 Σ_i^p-完全的。

5.2　证明论断 5.9。

5.3　证明：如果 3SAT 可以多项式时间归约到$\overline{\text{3SAT}}$，则 **PH**=**NP**。

5.4　证明：用交错图灵机给出的 **PH** 定义与它的其他定义等价。

5.5　证明定理 5.10。

5.6　修改定理 5.11 的证明过程，证明：SAT$\notin$**TISP**(n^c，n^d)对任意满足 $c(c+d)<2$ 的常数 c，d 成立。

5.7　证明：**APSPACE**=**EXP**。

5.8　根据定理 5.12 的证明概要，完成定理证明。

5.9　定义类 **DP** 由如下的所有语言 L 构成：存在 $L_1\in$**NP**，$L_2\in$**coNP**，使得 $L=L_1\cap L_2$。(注意，不要将 **DP** 和 **NP**$\cap$**coNP** 混淆，尽管它们表面上相似。)证明：

(a) EXACT INDSET$\in\Pi_1^p$。

(b) EXACT INDSET$\in$**DP**。

(c) **DP** 中的任意语言均可以多项式时间归约到 EXACT INDSET。

104　5.10　假设语言 A 满足 $\mathbf{P}^A=\mathbf{NP}^A$。证明：$\mathbf{PH}^A\subseteq\mathbf{P}^A$(这说明定理 5.4 的证明是相对的)。

5.11　证明：SUCCINCT SET COVER$\in\Pi_2^p$。

5.12　证明定理 5.4 的第 1 部分：对任意 $i\geqslant 1$，如果 $\Sigma_i^p=\Pi_i^p$，则多项式分层坍塌到第 i 层。

5.13　[Sch96]本题研究机器学习理论中的一个重要概念——VC 维(Vapnik-Chervonenkis dimension)。如果$\mathcal{S}=\{S_1,S_2,\cdots,S_m\}$是集合 U 的一个子集族，则$\mathcal{S}$的 VC 维(记为 VC($\mathcal{S}$))是满足如下条件的最大子集 $X\subseteq U$ 的大小：对于任意 $X'\subseteq X$，均存在 i 使得 $S_i\cap X=X'$。(称 X 被$\mathcal{S}$打散。)

对于布尔线路 C，如果每个 S_i恰由满足 $C(i,x)=1$ 的所有元素 $x\in U$ 构成，则称 C 紧凑地表示了集族$\mathcal{S}$。最后，

$$\text{VC-DIMENSION}=\{(C,k):C\text{ 紧凑地表示了一个满足 VC}(\mathcal{S})\geqslant k\text{ 的集族}\mathcal{S}\}$$

(a) 证明：VC-DIMENSION$\in\Sigma_3^p$。

105　(b) 证明：VC-DIMENSION 是 Σ_2^p-完全的。

布尔线路

人们可以认为 **P**≠**NP**。但是，在下述意义下，SAT 是易处理的：对于任意 ℓ，存在运行时间为 ℓ^2 的短程序正确地处理所有规模为 ℓ 的实例。

——卡普，利普顿(Lipton)，1982

本章研究一种称为布尔线路的计算模型，它是布尔公式的推广，也是现代计算机中硅芯片的简化模型。布尔线路作为处理非一致计算的自然模型，经常出现在计算复杂性理论中(例如，参见第 19 章和第 20 章)。在标准(或一致)的图灵机模型中，同一图灵机适用于所有无穷种输入规模。相比之下，非一致模型允许为每个输入规模采用不同的算法。因此，卡普和利普顿的上述引言指出了如下的可能性：人们有可能量身定制出一款体积小、效率高的硅芯片用于求解(不妨说)100 000 个变量上的任意 3SAT 问题。这种硅芯片的存在性不能排除，即使 **P**≠**NP**。读者可能已经猜到，本章给出的证据将表明：3SAT 问题的这种高效的硅芯片不可能存在，至少当 3CNF 公式中变量的个数开始变大时这种硅芯片是不可能存在的。

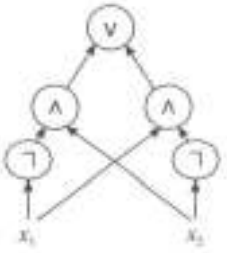
研究布尔线路的另一个动机是，从数学上看布尔线路比图灵机简单一些。因此，为布尔线路证明下界可能比为图灵机证明下界要容易一些。事实上，正如 6.1 节所述，布尔线路下界在原理上有望证明 **P**≠**NP**。从 20 世纪 70 年代晚期开始，人们一直致力于证明布尔线路下界。第 14 章将给出这一过程中得到的一些成果，第 23 章将说明这一个过程在何处陷入了困境并阐释其中的原因。

6.1 节定义了布尔线路和类 $\mathbf{P}_{/poly}$，其中 $\mathbf{P}_{/poly}$ 是多项式规模的线路所计算的所有语言构成的类。我们将证明，$\mathbf{P}_{/poly}$ 包含了由多项式时间图灵机计算的语言构成的类 **P**。同时，我们还利用布尔线路给出了库克-勒维定理(定理 2.10)的另一种证明。6.2 节将研究一致生成布尔线路族，并证明这种布尔线路族刻画了 **P** 的另一种性质。在 6.3 节中，我们从另一个方向展开讨论，用"纳言"图灵机来刻画 $\mathbf{P}_{/poly}$ 的性质。6.4 节研究卡普和利普顿的一个结果，也就是第一自然段指出的结论，亦即，如果多项式分层 **PH**(在第 5 章中定义)不坍塌，则 $\mathbf{NP}\not\subseteq\mathbf{P}_{/poly}$。当然，(不依赖于其他条件)证明 $\mathbf{NP}\not\subseteq\mathbf{P}_{/poly}$ 非常难，因为它意味着 **P**≠**NP**。不要说证明 $\mathbf{NP}\not\subseteq\mathbf{P}_{/poly}$，就连从 **NEXP** 中找出一个不属于 $\mathbf{P}_{/poly}$ 的语言目前仍是一个未解决的问题！我们已知的结果是，几乎所有的布尔函数都需要指数规模的线路才能被计算：参见 6.5 节。6.6 节采用指数规模和常数深度的一致布尔线路族，给出了多项式分层的另一个性质。最后，6.7.1 节研究了 $\mathbf{P}_{/poly}$ 的一些有意义的子类，如 **NC** 和 **AC**，并阐述它们与并行计算的关系。我们引入 **P** 完全性，并以它为工具研究计算问题是否存在高效并行算法。

6.1 布尔线路和 $\mathbf{P}_{/poly}$

布尔线路是表示如何将一系列标准布尔操作与(∧)、或(∨)、非(¬)作用到输入二进制串上产生输出结果的一种程序图。例如，图 6-1 给出了在两个位上计算异或(XOR)函数的布尔线路。现在，我们给出布尔线路的形式定义。为方便计，假设布尔线路仅输出一个位。我们的定

义可以直接推广，以定义"输出不止一个位"的布尔线路。然而，我们通常不需要这种推广。



定义 6.1（布尔线路）　对任意 $n\in\mathbf{N}$，一个 n 输入单输出的布尔线路是具有 n 个源顶点和 1 个汇顶点的有向图。源顶点也称输入顶点，指的是入度为 0 的顶点。汇顶点也称输出顶点，指的是出度为 0 的顶点。每个非源顶点称为一个逻辑门，并用逻辑操作 $\wedge$（与）、$\vee$（或）、$\neg$（非）中的一个操作进行标记。顶点的扇入度指的是进入该顶点的边的条数。标记为 $\vee$ 和 $\wedge$ 的顶点的扇入度等于 2，而标记为 $\neg$ 的顶点的扇入度等于 1。布尔线路 C 的规模（或大小）是 C 中顶点的个数，记为 $|C|$。

如果 C 是一个布尔线路而 $x\in\{0,1\}^n$ 是它的一个输入，则将 C 在 x 上的输出记为 $C(x)$，$C(x)$ 可以自然地定义如下。形式上，我们为 C 中的每个顶点 v 定义输出值 $\mathrm{val}(v)$：如果 v 是第 i 个源顶点，则定义 $\mathrm{val}(v)=x_i$；否则，$\mathrm{val}(v)$ 递归地定义为将顶点 v 上的逻辑操作作用于 v 的所有入边关联的顶点的输出值上得到的结果。$C(x)$ 是 C 的输出顶点的值。

尽管限定了布尔线路的扇入度小于等于 2，但上述定义仍不失一般性，因为每个 $\wedge$ 逻辑门或 $\vee$ 逻辑门均可以替换为由 $f-1$ 个扇入度为 2 的逻辑门构成的子线路。在 6.7.1 节中，布尔线路的扇入度非常重要，因为在那里将研究深度为限定常数的布尔线路。同时，注意到，前面各章遇到的布尔公式也都是布尔线路，其中每个顶点的扇出度为 1。顶点的扇出度指的是从该顶点出发的边的条数。相对于扇出度为 1 的布尔线路，扇出度为 2 的布尔线路允许将顶点的输出值作为中间值多次使用。注意，利用扇出度为 2 的布尔线路可以直接实现具有任意扇出度的布尔线路。

107

作为定义布尔线路的动机之一，布尔线路是现代计算机中硅芯片的模型[⊖]。因此，如果证明了某个任务可以由一个小规模布尔线路求解，则该任务可以高效地实现到一个硅芯片上。

同往常一样，我们用渐进分析来刻画布尔线路所判定的语言的复杂性。

定义 6.2（线路族和语言识别）　设 $T:\mathbf{N}\to\mathbf{N}$ 是一个函数，$T(n)$ 规模的线路族是一系列布尔线路 $\{C_n\}_{n\in\mathbf{N}}$，其中 C_n 有 n 个输入位和一个输出位，并且其规模 $|C_n|\leqslant T(n)$ 对任意 n 成立。

对于语言 L，如果存在 $T(n)$ 规模的线路族 $\{C_n\}_{n\in\mathbf{N}}$ 使得 $x\in L\Leftrightarrow C_n(x)=1$ 对任意 $x\in\{0,1\}^n$ 成立，则称 $L\in\mathbf{SIZE}(T(n))$。

例 6.3　语言 $\{1^n:n\in\mathbf{Z}\}$ 可以用一个线性规模线路族判定。布尔线路是一棵由逻辑与门构成的树，它计算所有输入位的逻辑与。语言 $\{\langle m,n,m+n\rangle:m,n\in\mathbf{Z}\}$ 也可以用一个线性规模线路族来判定。布尔线路实现小学生算术加法。记住，这里加法指的是将两个数按二进制位逐位相加。两个位相加可以用一个 $O(1)$ 规模的布尔线路实现，并且该操作可能向前进位，所进的位将作为下一个位置上按位相加操作的输入。◀

注记 6.4（直线程序）　布尔线路除了可以建模成带标签的有向图，还可以建模成直线程序。直线程序指的是不含（像 if 或 Goto 这样的）分支语句和循环语句的程序，因而直线程序的运行时间可以由程序包含的指令个数来界定。

⊖ 实际上，硅芯片的线路都不含环路，而是将环路用于实现内存。但是，具有 C 个逻辑门的硅芯片在时间 T 内能够完成的任何计算也能用一个规模为 $O(C\cdot T)$ 的布尔线路计算。

布尔线路和直线程序的等价性颇具一般性，也就是说这种等价性本质上对任意合理的编程语言均成立。当然，同一个布尔线路用不同语言编程时运行时间可能会相差一个多项式因子。上述等价关系可以用带布尔操作的直线程序来很好地进行展示。以 $x_1, x_2, \cdots, x_n \in \{0,1\}$ 为输入变量的长度为 T 的布尔直线程序是形如 $y_i = z_{j_1} OP z_{j_2}$ 的 T 条语句的序列，其中 $i=1,2,\cdots,T$，而 OP 等于 $\vee$ 或 $\wedge$，z_{j_1} 和 z_{j_2} 要么是输入变量，要么是输入变量的否定，要么是某个 y_j 且 $j<i$。在输入变量的每种赋值上，直线程序的一次计算是其中每个语句的一次顺序执行，进而确定了 $y_1, y_2, \cdots, y_T$ 的值。这次计算的输出指的是值 y_T。

可以直接证明，n 个比特的一个布尔函数 f 可以用上述形式的 S-行直线程序计算当且仅当 f 可以用一个规模为 S 的布尔线路计算(参见习题 6.2)。例如，图 6-1 所示的布尔线路可以如下地写成以变量 x_1, x_2 为输入的直线程序：

$$y_3 = \neg x_1;$$
$$y_2 = \neg x_2;$$
$$y_3 = y_1 \wedge x_2;$$
$$y_4 = x_1 \wedge y_2;$$
$$y_5 = y_3 \vee y_4$$

由于合取范式公式是特殊类型的布尔线路，因此论断 2.13 表明，从 $\{0,1\}^n$ 到 $\{0,1\}$ 的任意函数 f 均可以用一个规模为 $n2^n$ 的布尔线路来计算。事实上，习题 6.1 表明，这样的函数用规模为 $O(2^n/n)$ 的布尔线路就可以计算。因此，当我们需要考虑"小规模"布尔线路时，就会出现各种有趣的复杂性类。例如，下面定义的复杂性类非常有用。

定义 6.5 (类 $\mathbf{P}_{/poly}$) $\mathbf{P}_{/poly}$ 是由多项式规模的线路族能够判定的所有语言构成的类，亦即 $\mathbf{P}_{/poly} = \bigcup \mathbf{SIZE}(n^c)$。

当然，同 $\mathbf{P}$ 一样，人们也可以质疑 $\mathbf{P}_{/poly}$ 中语言是否真的就是能够被高效判定的。例如，尽管规模为 n^{100} 的线路族具有多项式规模，但它是高效的吗？1.6.2 节在一定程度上回答了这个问题。复杂性理论学家对这个问题给出的另一个答案是，他们希望最终能够证明 SAT 这样的语言不属于 $\mathbf{P}_{/poly}$。因此，如果允许 $\mathbf{P}_{/poly}$ 的定义中使用大规模的布尔线路，则复杂性理论学家得到的结论将会变得更强。 108

6.1.1 $\mathbf{P}_{/poly}$ 和 $\mathbf{P}$ 之间的关系

$\mathbf{P}_{/poly}$ 和 $\mathbf{P}$ 之间有什么关系？首先，我们证明 $\mathbf{P} \subseteq \mathbf{P}_{/poly}$。

定理 6.6 $\mathbf{P} \subseteq \mathbf{P}_{/poly}$。

证明 证明过程类似于库克-勒维定理(定理 2.10)的证明。事实上，用定理 6.6 可以得到库克-勒维定理的另一个证明。

回顾一下评注 1.7，每个运行时间为 $O(T(n))$ 的图灵机 M 均可以用一个运行时间为 $O(T(n)^2)$ 的散漫图灵机 $\tilde{M}$ 模拟(其带头移动独立于输入)；并且如果采用更细致的模拟，则模拟时间甚至可以降为 $O(T(n)\log T(n))$。因此，只需证明：对每个 $T(n)$ 时间的散漫图灵机 M，存在一个 $O(T(n))$ 规模的线路族 $\{C_n\}_{n\in N}$，使得 $C_n(x) = M(x)$ 对任意 $x \in \{0,1\}^n$ 成立。 109

令 M 是一个散漫图灵机，$x \in \{0,1\}^*$ 是 M 的某个输入，定义 M 在 x 上的运行副本是运行过程中各个步骤的快照构成的序列 $z_1, \cdots, z_{T(n)}$。快照 z_i 由第 i 步时机器的状态和各个带头读取的符号构成，它可以用具有常数长度的二进制串编码，而且 z_i 可以基于一些信息被计

算出来，这些信息包括输入 x、前一个步骤的快照 z_{i-1} 和快照 z_{i_1}，…，z_{i_k}，其中 i_j 表示 M 的第 j 条带的带头在第 i_j 步时停留在与第 i 步相同的位置上[⊖]。由于上述信息仅有常数个位串且每个串的长度也是常数，因而可以用一个常数规模的布尔线路根据这些快照计算得到 z_i。

所有这种具有常数规模的布尔线路复合在一起就得到一个布尔线路，它以 x 为输入，计算得到图灵机 $\tilde{M}$ 在 x 上执行的最后一个步骤所对应的快照 $z_{T(n)}$。再注意到，存在常数规模的布尔线路在 $z_{T(n)}$ 上输出 1 当且仅当 $z_{T(n)}$ 是一个接受快照（此时，M 输出 1 并停机）。因此，存在一个 $O(T(n))$ 规模的线路 C_n 使得 $C_n(x)=M(x)$ 对任意 $x\in\{0,1\}^n$ 成立。 ■

评注 6.7 定理 6.6 的证明实际上得到了比定理本身更强的结论：定理证明中得到的线路不仅具有多项式规模，而且还可以在多项式时间内计算得到，甚至还可以在对数空间内计算得到。为看清这一点，只需注意到，任意图灵机 M 可以被一个数漫图灵机 $\tilde{M}$ 模拟并确保如下函数 f 是对数空间可计算的：f 将 n，i 映射为 $\tilde{M}$ 在长度为 n 的输入上运行到第 i 步时带头所处的位置。

$\mathbf{P}\subseteq\mathbf{P}_{/poly}$ 是真包含关系。事实上，存在不可判定的一元语言不属于 $\mathbf{P}$（甚至不属于 $\mathbf{EXP}$）；然而所有的一元语言均属于 $\mathbf{P}_{/poly}$。

论断 6.8 令 $L\subseteq\{0,1\}^*$ 是任意一元语言（即 $L\subseteq\{1^n:n\in\mathbf{N}\}$），则 $L\in\mathbf{P}_{/poly}$。

证明 我们构造一个判定 L 的布尔线路族 $\{C_n\}_{n\in\mathbf{N}}$，使其具有线性规模。如果 $1^n\in L$，则取 C_n 为例 6.3 中给出的规模为 n 的线路。否则，取 C_n 为总输出 0 的线路。 ■

下面的一元语言是不可判定的，该语言是停机问题（参见 1.5.1 节）的一元形式。

$$\mathrm{UHALT}=\{1^n:n\text{ 是序对}\langle M,x\rangle\text{ 的二进制编码，}M\text{ 在输入 }x\text{ 上停机}\}$$

6.1.2　线路的可满足性和库克-勒维定理的另一种证明

布尔线路可以用来给出库克-勒维定理（定理 2.10）的另一种证明，这需要借助下面的语言。

定义 6.9（线路可满足性或 CKT-SAT）　语言 CKT-SAT 由仅输出 1 个位且存在满足性赋值的所有布尔线路（的位串表示）构成。亦即，一个 n 输入线路 C 的位串表示属于 CKT-SAT 当且仅当存在 $u\in\{0,1\}^n$ 使得 $C(u)=1$。

110

显然，CKT-SAT 属于 $\mathbf{NP}$，因为满足性赋值 u 就是 CKT-SAT$\in\mathbf{NP}$ 的证明。由下面的两个引理即可得出库克-勒维定理。

引理 6.10 CKT-SAT 是 $\mathbf{NP}$-难的。

证明 如果 $L\in\mathbf{NP}$，则存在多项式时间图灵机 M 和多项式 p 使得 $x\in L$ 当且仅当 $M(x,u)=1$ 对某个 $u\in\{0,1\}^{p(|x|)}$ 成立。注意到，定理 6.6 的证明得到了一个多项式时间变换，它将 M，x 变换为一个线路 C 使得 $M(x,u)=C(u)$ 对任意 $u\in\{0,1\}^{p(|x|)}$ 成立。因此，$x\in L$ 当且仅当 $x\in$ CKT-SAT。 ■

引理 6.11 CKT-SAT$\leqslant_p$3SAT。

证明 对于任意线路 C，我们将它映射为一个 3CNF 公式 φ：对于 C 的每个顶点 v_i，我们为 φ 相应地引入一个变量 z_i。如果顶点 v_i 是顶点 v_j 和 v_k 的逻辑与，则在 φ 中引入等价地表达条件“$z_i=(z_j\wedge z_k)$”的几个子句，亦即引入

⊖ “由于 M 是数漫图灵机，故 i_1，…，i_k 仅依赖于 i，而不依赖于 x。

$$(\overline{z_i} \vee \overline{z_j} \vee z_k) \wedge (\overline{z_i} \vee z_j \vee \overline{z_k}) \wedge (z_i \vee \overline{z_j} \vee z_k) \wedge (z_i \vee \overline{z_j} \vee \overline{z_k})$$

类似地，如果 v_i 是 v_j 和 v_k 的逻辑或，则在 φ 中引入等价地表达条件"$z_i=(z_j \vee z_k)$"的几个子句；如果 v_i 是 v_j 的逻辑非，则在 φ 中引入子句 $(z_i \vee z_j) \wedge (\overline{z_i} \vee \overline{z_j})$。最后，如果 v_i 是 C 的输出顶点，则在 φ 中引入子句 (z_i)，该子句为真当且仅当 z_i 为真。不难证明，φ 是可满足的当且仅当 C 是可满足的。显然，上述归约在输入规模的多项式时间内完成。 ■

6.2　一致线路

在复杂性理论中，$\mathbf{P}_{poly}$ 是一个特别别扭的复杂性类，因为它甚至包含了 6.1.1 节定义的不可判定语言 UHALT。出现这种现象的根本原因在于，语言 L 属于 $\mathbf{P}_{poly}$ 仅要求存在判定 L 的线路族，而不管这样的线路族实际是否能够被构造出来。因此，如果仅限于考虑能够被具有一定效率的图灵机构造出来的线路，则可能会得到新的结果。

定义 6.12（**P 一致线路族**）　对于线路族 $\{C_n\}$，如果存在多项式时间图灵机 M 在输入 1^n 上输出线路 C_n 的位串表示，则称线路族 $\{C_n\}$ 是 **P** 一致的。

如果将线路限定为 **P** 一致线路，则类 $\mathbf{P}_{poly}$ 将坍塌到类 **P**。

定理 6.13　语言 L 可以被一个 **P** 一致线路族计算当且仅当 $L \in \mathbf{P}$。

证明概要　如果 L 可被多项式时间图灵机 M 产生的线路族 $\{C_n\}$ 计算，则我们可以如下构造一个计算 L 的多项式时间图灵机 $\widetilde{M}$：在输入 x 上，图灵机 $\widetilde{M}$ 先运行 $M(1^{|x|})$ 得到布尔线路 $C_{|x|}$，然后 $\widetilde{M}$ 再计算 C 在 x 上的输出。 111

反过来，如果 $L \in \mathbf{P}$，则严格地沿用定理 6.6 的证明过程，将得到一个 **P** 一致线路族，它恰好计算语言 L。 ■

6.2.1　对数空间一致线路族

我们还可以引入一种更强的一致线路——由对数空间图灵机产生的一致线路。回顾一下定义 4.16，函数 $f:\{0,1\}^* \to \{0,1\}^*$ 是隐式对数空间可计算的指的是映射 $x, i \mapsto f(x)_i$ 是对数空间可计算的。

定义 6.14（**对数空间一致线路族**）　对于线路族 $\{C_n\}$，若存在隐式对数空间可计算函数将 1^n 映射为线路 C_n 的位串表示，则称线路族 $\{C_n\}$ 是对数空间一致的。

由于对数空间计算可以在多项式时间内完成，因此对数空间一致线路也是 **P** 一致线路。注意，定义 6.14 是健壮的，它所定义的对数空间一致线路可以有多种变形，因为定义中没有要求线路的串表示具体是什么形式。具体地讲，规模为 S 的线路表示为其底层有向图的 $S \times S$ 的邻接矩阵和一个规模为 S 的数组，其中数组用于存储每个顶点的标记（或逻辑门的类型）。将所有顶点视为 $[S]$ 中的数，我们让最前面的 n 个顶点表示输入顶点，而最后一个顶点表示输出顶点。也就是说，线路族 $\{C_n\}$ 是对数空间一致的当且仅当下列函数是 $O(\log n)$ 空间可计算的：

- SIZE(n) 返回线路 C_n 的规模 S（二进制形式）；
- TYPE(n, i) 返回 C_n 的第 i 个顶点的标记，其中 $i \in [m]$，$m=\text{size}(n)$，并且每个顶点的标记均属于 $\{\vee, \wedge, \neg, \text{NONE}\}$。
- EDGE(n, i, j) 输出 1，表明线路 C_n 中从第 i 个顶点到第 j 个顶点有一条有向边。

注意，上述三个函数的输入和输出均可以表示为 $O(\log|C_n|)$ 个位。习题 6.10 要求证

明，如果用邻接列表表示线路，则(相对于用邻接矩阵表示线路而言)线路判定的语言构成的类不会改变。周密地修改定理 6.6 的证明过程，可以证明下面的定理(参见习题 6.4)。

定理 6.15 存在判定语言 L 的多项式规模的对数空间一致线路族当且仅当 L 属于 **P**。

6.3 纳言图灵机

利用"纳言"图灵机可以等价地定义 $\mathbf{P}_{/poly}$。"纳言"图灵机在每个 n 值上有一个建言串 a_n，该建言可以被图灵机在输入长度为 n 的任意输入上计算时使用。

定义 6.16 设 T，a：$\mathbf{N}\to\mathbf{N}$ 是两个函数。由带有 $a(n)$ 位建言的 $T(n)$ 时间图灵机判定
112 的所有语言构成一个复杂性类，记为 $\mathbf{DTIME}(T(n))/a(n)$，语言 $L\in\mathbf{DTIME}(T(n))/a(n)$ 使得存在满足 $a_n\in\{0,1\}^{a(n)}$ 的一系列位串 $\{a_n\}_{n\in\mathbf{N}}$ 和一个图灵机 M 满足对于任意 $x\in\{0,1\}^n$ 均有

$$M(x,a_n)=1\Leftrightarrow x\in L$$

其中 M 在输入 (x,a_n) 上至多运行 $O(T(n))$ 个步骤。

例 6.17 任意一元语言均可以被带 1 位建言的多项式时间图灵机判定。在长度为 n 的输入上，仅含 1 个位的建言串表明 1^n 是否属于该语言。特别地，上述结论对 6.1.1 节中定义的语言 UHALT 成立。

用带有建言的图灵机可以如下刻画 $\mathbf{P}_{/poly}$ 的性质。

定理 6.18 (带建言的多项式时间图灵机判定 $\mathbf{P}_{/poly}$)　$\mathbf{P}_{/poly}=\bigcup_{c,d}\mathbf{DTIME}(n^c)/n^d$。

证明　如果 $L\in\mathbf{P}_{/poly}$，则 L 可以被一个多项式规模线路族 $\{C_n\}$ 计算。在长度为 n 的输入上，我们可以将 C_n 的位串表示作为如下简单的多项式时间图灵机 M 的建言串：M 的输入是串 x 和一个 n 输入线路 C，其输出是 $C(x)$。

反过来，如果 L 是由多项式时间图灵机 M 判定的语言，M 采纳规模为 $a(n)$ 的建言族 $\{a_n\}_{n\in\mathbf{N}}$，其中 a 是一个多项式，则可以采用定理 6.6 证明过程中的方法，为任意 n 构造一个多项式规模的线路 D_n，使得 $D_n(x,a)=M(x,a)$ 对任意 $x\in\{0,1\}^n$ 和 $a\in\{0,1\}^{a(n)}$ 成立。令 C_n 是以 x 为输入以 $D_n(x,a_n)$ 为输出的多项式规模的线路，亦即将线路 D_n 的后半部分输入"取定"为串 a_n 即得到线路 C_n。(取定线路的一部分输入指的是，将线路 C 的输入视为两个部分 $x\in\{0,1\}^n$，$y\in\{0,1\}^m$ 并将 y 对应的输入固定下来；这样得到线路 C_y，它在任意输入 x 上输出 $C(x,y)$。完成上述操作的同时，可以很容易确保 C_y 的规模不超过 C 的规模。这种简单的思想在复杂性理论中很常见。)　■

6.4 $\mathbf{P}_{/poly}$ 和 NP

卡普和利普顿将"SAT 是否存在小规模线路"这一问题形式化为"SAT 是否属于 $\mathbf{P}_{/poly}$"。他们就此问题给出的答案是，如果多项式分层不坍塌，则 SAT 不属于 $\mathbf{P}_{/poly}$。

定理 6.19 (卡普-利普顿定理[KL80])　如果 $\mathbf{NP}\subseteq\mathbf{P}_{/poly}$，则 $\mathbf{PH}=\mathbf{\Sigma}_2^p$。

证明　根据定理 5.4，要证 $\mathbf{PH}=\mathbf{\Sigma}_2^p$，只需证明 $\mathbf{\Pi}_2^p\subseteq\mathbf{\Sigma}_2^p$；特别地，只需证明 $\mathbf{\Sigma}_2^p$ 中包
113 含了 $\mathbf{\Pi}_2^p$-完全语言 $\mathbf{\Pi}_2$SAT，其中 $\mathbf{\Pi}_2$SAT 是由具有如下形式的取值为真的所有布尔公式构成的语言：

$$\forall u\in\{0,1\}^n\exists v\in\{0,1\}^n\varphi(u,v)=1 \tag{6.1}$$

其中 φ 是不含量词的布尔公式。

如果 $\mathbf{NP}\subseteq\mathbf{P}_{/poly}$，则存在多项式 p 和一个规模为 $p(n)$ 的线路族 $\{C_n\}_{n\in\mathbb{N}}$ 使得对任意布尔公式 φ 和 $u\in\{0,1\}^*$ 均有：$C_n(\varphi,u)=1$ 当且仅当存在 $v\in\{0,1\}^*$ 满足 $\varphi(u,v)=1$。这样就得到求解判定问题 SAT 的布尔线路族。然而，定理 2.18 给出了一个算法，它可以将任意一个 SAT 的判定算法转换成一个输出满足性赋值的算法。将定理 2.18 给出的算法也视为一个线路，它与 $\{C_n\}_{n\in\mathbb{N}}$ 中的线路复合，就将线路族 $\{C_n\}_{n\in\mathbb{N}}$ 转换成一个 $q(n)$ 规模的线路族 $\{C'_n\}_{n\in\mathbb{N}}$（其中 $q(\cdot)$ 是一个多项式），使得对于任意布尔公式 φ 和 $u\in\{0,1\}^*$，如果存在串 $v\in\{0,1\}^*$ 满足 $\varphi(u,v)=1$，则 $C'_n(\varphi,u)$ 将输出满足 $\varphi(u,v)=1$ 的一个串 v。（注：虽然我们没有形式上定义输出不止一个位的线路，但推广定义 6.1 来定义这种线路是显然的。）

当然，假设条件 $\mathbf{NP}\subseteq\mathbf{P}_{/poly}$ 仅隐含了这种布尔线路族的存在性。卡普和利普顿的主要思想是，这种布尔线路族的成员可以通过存在量词（$\exists$）来"猜测"。这是因为，布尔线路 C'_n 输出了一个满足性赋值（如果这种赋值存在），并且很容易直接验证输出的赋值是否是满足性赋值。形式地说，由于 C'_n 可以用 $10q(n)^2$ 个位来表示，因此如果(6.1)式成立，则下述的量化公式为真：

$$\exists w\in\{0,1\}^{10q(n)^2}\ \forall u\in\{0,1\}^n \text{ 满足 } w \text{ 表示布尔线路 } C' \text{ 且 } \varphi(u,C'(\varphi,u))=1 \qquad (6.2)$$

而且，如果(6.1)式不成立，则对某个 u，不存在 v 使得 $\varphi(u,v)=1$，进而(6.2)式也不成立。因此，(6.2)式成立当且仅当(6.1)式成立！最后，由于线路在指定输入上的输出结果可以用确定的方式在多项式时间内计算，因而(6.2)式可以在 $\mathbf{\Sigma}_2^p$ 内被验证。 ■

类似地，下面的定理表明，$\mathbf{P}_{/poly}$ 不可能包含 **EXP**。

定理 6.20（迈耶定理[KL80]） 如果 $\mathbf{EXP}\subseteq\mathbf{P}_{/poly}$，则 $\mathbf{EXP}=\mathbf{\Sigma}_2^p$。

证明概要 令 $L\in\mathbf{EXP}$。则 L 可以被一个 $2^{p(n)}$ 时间的散漫图灵机 M 计算，其中 p 是一个多项式。令 $x\in\{0,1\}^*$ 是一个输入串。对于任意 $i\in[2^{p(n)}]$，用 z_i 表示 M 在输入 x 上运行的过程中第 i 步的快照（参见定理 6.6 的证明）。如果 M 有 k 条工作带，则 $x\in L$ 当且仅当对于任意由索引 i，i_1，…，i_k 给定的 $k+1$ 个快照 z_i，z_{i_1}，…，z_{i_k} 满足下列易于验证的准则：(1)如果 z_i 是最后一个快照，则它应表明 M 的输出是 1；(2)如果 i_1，…，i_k 标识的步骤分别是 M 的各条带的带头最后停留在与第 i 步相同的位置上的步骤，则快照 z_i 从各条带上读取的符号应分别与输入和第 i_1，…，i_k 步时写到各条工作带上的符号一致。（注意，索引 i，i_1，…，i_k 可以在多项式时间内表示为二进制形式）然而，如果 $\mathbf{EXP}\subseteq\mathbf{P}_{/poly}$，则存在一个 $q(n)$ 规模的线路 C 能够根据 i 计算 z_i，其中 q 是一个多项式。因此，可以认为线路 C 隐式地计算了运行副本 $C(1)$，$C(2)$，…，$C(2^{p(n)})$。现在，关键在于，将运行副本表示成 **coNP** 谓词，也就是验证运行副本满足所有局部性准则的谓词。因此，$x\in L$ 当且仅当如下条件为真：

$$\exists C\in\{0,1\}^{q(n)}\ \forall i,i_1,\cdots,i_k\in\{0,1\}^{p(n)}\ T(x,C(i),C(i_1),\cdots,C(i_k))=1 \qquad 114$$

其中 T 是用于验证前述条件的多项式时间图灵机。这表明 $L\in\mathbf{\Sigma}_2^p$。 ■

定理 6.20 表明，如果 $\mathbf{P}=\mathbf{NP}$，则 $\mathbf{EXP}\subseteq\mathbf{P}_{/poly}$。事实上，由定理 5.4 可知，如果 $\mathbf{P}=\mathbf{NP}$，则 $\mathbf{P}=\mathbf{\Sigma}_2^p$。因此，如果 $\mathbf{EXP}\subseteq\mathbf{P}_{/poly}$，则 $\mathbf{P}=\mathbf{EXP}$，这与时间分层定理（定理 3.1）矛盾。因此，各种上界（此时指的是 $\mathbf{NP}\subseteq\mathbf{P}$）可以用于证明线路下界。

6.5 线路下界

由于 $\mathbf{P}\subseteq\mathbf{P}_{/poly}$，如果还能证明 $\mathbf{NP}\subseteq\mathbf{P}_{/poly}$，则我们就证明了 $\mathbf{P}\neq\mathbf{NP}$。卡普-利普顿定理提供了 $\mathbf{NP}\subseteq\mathbf{P}_{/poly}$ 的证据。**P** 与 **NP** 问题能通过证明 $\mathbf{NP}\subseteq\mathbf{P}_{/poly}$ 来加以解决吗？人们有理由将希望寄托于这种方法，而不是直接去证明图灵机的下界。通过用线路来表示计算，我们

似乎能够透视图灵机的内核，而不是仅仅将它视为一个黑盒。因而，第3章中阐述的相对化方法的局限性有望被规避。事实上，不难证明，某些函数确实需要大规模线路才能被计算。

定理6.21（硬函数[⊖]的存在性[Sha49a]）　对于任意 $n>1$，存在函数 $f:\{0,1\}^n\to\{0,1\}$ 不能被规模为 $2^n/(10n)$ 的线路 C 计算。

证明　证明过程通过简单的计数方法完成：

- 从 $\{0,1\}^n$ 到 $\{0,1\}$ 的函数共有 2^{2^n} 个；
- 由于规模不超过 S 的任意线路均可以表示为 $9\cdot S\log S$ 个二进制位（比如，表示成邻接表），因此规模不超过 S 的线路至多有 $2^{9S\log S}$ 个。

取定 $S=2^n/(10n)$，则规模为 S 的线路至多有 $2^{9S\log S}\leqslant 2^{2^n n/10n}<2^{2^n}$ 个。因此，这种线路所计算的函数少于 2^{2^n} 个。这意味着，存在函数不能被规模为 $2^n/(10n)$ 的线路计算。要细致的计算将表明，布尔线路的规模界限可以修改为 $(1-\varepsilon)2^n/n$ 甚至 $2^n(1+\log n/n-O(1/n))$，其中 $\varepsilon>0$，此时定理仍然成立（参见[FM05]）。 ■

用另一种方法也可以得出证明。考虑函数 $f:\{0,1\}^n\to\{0,1\}$ 的随机选择，即对于定义域中的每个输入 $x\in\{0,1\}^n$，均匀独立地选择 $\{0,1\}$ 中的一个值作为 $f(x)$。于是，对于任意固定的线路 C 和输入 x，事件 $C(x)=f(x)$ 的概率等于1/2。由于 f 在每个输入上的函数值均是独立选取的，因此 C 恰好计算函数 f（亦即，$C(x)=f(x)$ 对任意 $x\in\{0,1\}^n$ 成立）的概率等于 2^{-2^n}。由于规模至多为 $2^n/(10n)$ 的线路不超过 $2^{n\cdot 2^n}$ 个，应用合并界限（参见A.2节）可知，存在计算 f 的规模不超过 $2^n/(10n)$ 的线路的概率至多为

$$\frac{2^{n\cdot 2^n}}{2^{2^n}}=2^{-(1-n)\cdot 2^n}$$

115 这个数随着 n 的增大快速趋于0。特别地，由 $2^{-(1-n)\cdot 2^n}<1$ 表明，存在一个函数 f 不能被任意规模不超过 $2^n/(10n)$ 的线路计算。上述证明技术称为**概率方法**，它通过证明“满足某种性质的对象存在的概率大于0”来证明“满足该性质的对象确实存在”。概率方法广泛地用于计算机科学和组合学（例如，参见本书第13章，第19章和第21章）。注意，概率方法得到的结论实际上强于定理6.21的结论。事实上，上述概率方法证明了不仅存在硬函数（即，不能被规模为 $2^n/(10n)$ 的线路计算的函数），而且绝大多数从 $\{0,1\}^n$ 到 $\{0,1\}$ 的函数均是硬函数。因此，我们有希望找出一个不属于 **NP** 的硬函数，进而证明 $\mathbf{NP}\not\subseteq\mathbf{P}_{/poly}$。遗憾的是，这种愿望至今仍未变成现实。二十年过去了，**NP** 语言的线路规模的最佳下界目前是 $(5-o(1))n$[ILMR05]。然而，习题6.5表明，**PH** 语言的线路规模下界优于上述下界。值得肯定的是，在限制得更强的线路模型上，线路下界的证明已经获得了显著的成果。有关这一点，我们将在第14章中讨论。

6.6　非一致分层定理

同时间受限的（确定型或非确定型）图灵机和空间受限的图灵机一样，布尔线路也有分层定理。也就是说，大规模线路计算的函数严格地多于小规模线路计算的函数。

定理6.22（非一致分层定理）　对于任意满足 $2^n/n>T'(n)>10T(n)>n$ 的两个函数

⊖ 参照人工智能中“hard function”的中文术语。——译者注

T，T'：$\mathbf{N}\to\mathbf{N}$均有

$$\mathbf{SIZE}(T(n))\subset\mathbf{SIZE}(T'(n))$$

证明 有意思的是，虽然第3章的对角线方法似乎无能为力了，但定理6.21中的计数方法却能用于证明定理6.22。我们仅证明$\mathbf{SIZE}(n)\subset\mathbf{SIZE}(n^2)$，以此展示证明过程的主要思想。

由定理6.21可知，对于任意ℓ，存在函数$f:\{0,1\}^\ell\to\{0,1\}$不能被规模为$2^\ell/(10\ell)$的线路计算。另一方面，由论断2.13可知，从$\{0,1\}^\ell$到$\{0,1\}$的任意函数均可以被规模为$2^\ell 10\ell$的线路计算(习题6.1给出了更紧的界)。

于是，如果令$\ell=1.1\log n$，$g:\{0,1\}^*\to\{0,1\}$是将f限制在输入的前ℓ个位上得到的函数，则

$$g\in\mathbf{SIZE}(2^\ell 10\ell)=\mathbf{SIZE}(11n^{1.1}\log n)\subseteq\mathbf{SIZE}(n^2)$$
$$g\notin\mathbf{SIZE}(2^\ell/(10\ell))=\mathbf{SIZE}(n^{1.1}/(11\log n))\supseteq\mathbf{SIZE}(n)$$ ■

6.7 线路复杂性类的精细分层

本节引入$\mathbf{P}_{poly}$的一些子类。这样做的原因有两个。第一，区分**NP**和这些子类将有助于了解如何区分**NP**和$\mathbf{P}_{poly}$。第二，这些子类对应于一些计算模型，而这些模型本身也很有意义。

116

也许最重要的莫过于*大规模并行机*与线路之间的联系。这里简要介绍一下大规模并行机。读者无需详细地了解它，因为定理6.27的正确性不依赖于这些详细信息。在并行计算机中，若干个微处理器成品被简单地通过*互联网络*链接在一起，各个微处理器之间可以互相发送信息。连接微处理器常用的网络结构(如*超方体*)通常均能确保n个微处理之间的通信能够——假设网络总负载存在某个上界——在$O(\log n)$个步骤内完成。微处理器的计算以锁步为单位，每个锁步可能是若干个全局时钟周期。微处理器在每个锁步内完成少量计算，比如$O(\log n)$个位上的一个操作。于是，每个处理器均需要足够的内存来记录它在整个网络中的地址以及任意其他处理器的地址。这样，处理器才能与其他处理器进行通信。

如果存在具有$n^{O(1)}$个微处理器的并行机在$\log^{O(1)}n$的时间内求解一个规模为n的计算问题的任意实例，则称该计算问题*存在高效并行算法*。

例6.23 给定两个n位整数x，y，我们希望快速并行地计算$x+y$。小学生加法算法从最低位开始，逐位地执行进位加法。

该算法在n个步骤之后才能计算出最高位，并且它也不是并行算法。*先行进位法*是更好的算法，它让每个处理器处理一个位上的加法，然后通过处理器之间的通信实现后一位向前一位的进位。该算法使用$O(n)$个处理器，计算时间为$O(\log n)$。

同样，整数乘法和除法也存在高效并行算法，其中整数除法的并行算法并不像小学生除法算法那样直观！

许多矩阵运算(包括矩阵乘法、秩、行列式和逆)也可以通过并行来高效地完成。这些操作的并行算法请参见习题和本章注记。

一些图论算法(如最短路径算法和最小生成树算法)也有高效的并行实现。

但是，一些著名的多项式时间问题(如最大流问题和线性规划问题)却仍未找到高效的并行算法。而且，人们猜想，这些问题不存在高效的并行算法。有关于此，参见6.7.2节对**P**完全性的讨论。 ◀

6.7.1　类 NC 和类 AC

现在，我们将并行计算与线路关联起来。线路的深度指的是从输入顶点到输出顶点的所有有向路径中最长路径的长度。

定义 6.24（类 NC）　对于任意 d，如果存在判定语言 L 的线路族 $\{C_n\}$ 使得 C_n 的规模为 $\text{poly}(n)$ 且深度为 $O(\log^d n)$，则称 L 属于类 $\mathbf{NC}^d$。类 NC 指的是 $\bigcup_{d\geq 1}\mathbf{NC}^d$。

117

我们还可以要求定义中的线路是对数空间一致的，由此得到一致 NC 的定义。此外，一个相关的类定义如下。

定义 6.25（类 AC）　类 $\mathbf{AC}^d$ 的定义与类 $\mathbf{NC}^d$ 的定义类似，只是线路中逻辑门的扇入度是没有限制的；也就是说，∨ 逻辑门和 ∧ 逻辑门可以作用到多于两个的位上。类 AC 指的是 $\bigcup_{d\geq 0}\mathbf{AC}^d$。

由于无限制的扇入度 $\text{poly}(n)$ 可以用深度为 $O(\log n)$ 的一棵 ∨ 树或 ∧ 树来模拟，因而 $\mathbf{NC}^d\subseteq\mathbf{AC}^d\subseteq\mathbf{NC}^{d+1}$。在第 14 章中我们将会看到，当 $i=0$ 时，上述包含关系是严格的。注意，$\mathbf{NC}^0$ 受到的限制极其苛刻，因为线路的输出仅依赖于常数个输入位，但 $\mathbf{AC}^0$ 却不受这种限制。

例 6.26　语言 PARITY $=\{x:x$ 含有奇数个 $1\}$ 属于 $\mathbf{NC}^1$。计算该语言的线路是二叉树，左子树计算前 $\lfloor|x|/2$ 个位的奇偶性，右子树计算后 $\lfloor|x|/2$ 个位的奇偶性，根结点综合左、右子树的输出得出答案。显然，消去上述线路中的递归，将得到一个深度为 $O(\log n)$ 的线路。该线路也是对数空间一致的。

第 14 章将证明，PARITY 不属于 $\mathbf{NC}^0$。　◀

人们已经证明，NC 刻画了存在高效并行算法的语言。

定理 6.27（NC 与并行算法）　一个语言有高效并行算法当且仅当该语言属于 NC。

证明概要　假设语言 $L\in\mathbf{NC}$ 且判定 L 的线路族为 $\{C_n\}$，其中 C_n 的规模为 $N=O(n^c)$ 而深度为 $D=O(\log^d n)$。选用具有 N 个处理器的通用并行机，并进行如下配置以判定语言 L。先找到 C_n 的明确定义，并让 C_n 的每个顶点对应并行机的一个单独的处理器。每个处理器计算得到相应顶点的输出之后，将表示结果的位发送到需要该结果的其他（线路顶点对应的）处理器。假设连接处理器的网络发送所有消息需要 $O(\log N)$ 时间，则并行算法的总时间开销为 $O(\log^{d+1}N)$。（注意，如果线路是非一致的，则并行算法也是非一致的。另一方面，如果线路是对数空间一致的，则并行算法也是对数空间一致的。）

反方向的证明可以类似地进行。让线路中包含 $N\cdot D$ 个顶点，并将这些顶点放入 D 个层中，其中第 t 层的第 i 个顶点完成第 i 个处理器在 t 时刻所做的计算。连接处理器的网络则通过线路中的线来实现。　■

6.7.2　P 完全性

一个仍未解决的主要问题是，是否所有多项式时间算法都存在高效的并行实现；换句话说，P=NC 是否成立。尽管目前人们甚至还不能区分 PH 和 $\mathbf{NC}^1$，但他们仍相信问题的答案是否定的。由此，引发了对 P 完全性理论的研究。P 完全性理论主要用于区分哪些问题可能是高效可并行化的（即，属于 NC），哪些问题不是高效可并行化的。

118

定义 6.28　如果一个语言属于 **P** 并且 **P** 中的每个语言均可以对数空间归约到它(参见定义 4.16)，则称该语言是 **P** 完全的。

下面定理的证明作为习题 6.15 留给读者。

定理 6.29　如果语言 L 是 **P** 完全的，则

1. $L\in$ **NC** 当且仅当 **P**=**NC**；

2. $L\in$ **L** 当且仅当 **P**=**L**。(其中，**L** 是所有对数空间可判定语言构成的集合，参见定义 4.5。)

下面给出了一个很自然的 **P** 完全语言。

定理 6.30　令 CIRCUIT-EVAL 是具有如下形式的所有序对$\langle C, x\rangle$构成的语言，C 是一个 n 输入单输出的线路，而 $x\in\{0,1\}^n$ 是满足 $C(x)=1$ 的一个输入。则 CIRCUIT-EVAL 是 **P** 完全的。

证明概要　显然，语言 CIRCUIT-EVAL 属于 **P**。从 **P** 中任意其他语言到 CIRCUIT-EVAL 的对数空间归约隐含于定理 6.15 的证明中。 ■

6.8　指数规模的线路

前面已经提到，任意语言均有规模为 $O(2^n/n)$的线路。然而，实际地构造出这样的线路却很难，甚至可能是不可判定的。如果我们要求线路必须是一致的，也就是要求线路必须是能够被有效计算的，则某些语言的线路复杂度可能会超过 2^n。事实上，用指数规模的线路可以给出一些熟悉的复杂性类的其他定义，这类似于定理 6.15 中给出的 **P** 的定义。

定义 6.31　(DC 一致)　设$\{C_n\}_{n\geq 1}$是一个线路族。如果存在一个多项式时间算法以$\langle n, i\rangle$为输入能够在多项式时间内计算得到 C_n的(邻接矩阵表示的)第 i 个位，则称线路族$\{C_n\}_{n\geq 1}$是有向连通一致的，简称为 DC 一致的。更确切地说，线路族$\{C_n\}_{n\geq 1}$是 DC 一致的当且仅当 6.2.1 节中为线路族$\{C_n\}_{n\geq 1}$定义的三个函数 SIZE，TYPE，EDGE 均是多项式时间可计算的。

注意，DC 一致线路族可以具有指数规模，但它们可以紧凑地表示为一个图灵机，并且该图灵机可以系统地根据需要在多项式时间内产生出线路的任意顶点。下面给出类 **PH** 的另一个特性。这里，我们将它定义为可以由限定深度的一致线路族计算的语言族。

定理 6.32　$L\in$ **PH** 当且仅当 L 可以被满足如下条件的一个 DC 一致线路族$\{C_n\}$计算：

1. 线路族仅使用“与”逻辑门、“或”逻辑门和“非”逻辑门；

2. 线路族的规模为 $2^{n^{O(1)}}$ 且深度为常数；

3. 线路族的扇入度无界(即可以是指数)；

4. 线路族的“非”逻辑门仅出现在输入层；亦即，“非”逻辑门仅直接作用于输入上而不能作用于其他逻辑门的输出值上。

我们将定理 6.32 的证明留作习题 6.17。如果在定理中去掉“线路族的深度为常数”这一限制，则得到的复杂性类恰好是 **EXP**(参见习题 6.18)。

本章学习内容

- 布尔线路可以用作图灵机的另一个模型。能够被多项式规模的布尔线路判定的所有语言构成类 $\mathbf{P}_{poly}$，它是 **P** 的严格超集且不包含 **NP**，除非多项式分层坍塌。

- 几乎所有从$\{0,1\}^*$到$\{0,1\}$的函数均需要指数规模的线路。在**NP**中找出这样一个函数就可以证明**P**≠**NP**。
- 类**NC**是由具有对数多项式深度和多项式规模的(一致)线路族判定的所有语言构成的复杂性类。**NC**中的计算任务对应于能够被高效并行化的计算任务。

本章注记和历史

从20世纪40年代开始，也就是从逻辑门刚用真空管实现的那个年代开始，线路就一直受到电子工程领域的研究。香农(Shannon)的开创性论文[Sha49a]提出了用最小线路实现布尔函数这一问题，并证明了n个位上最难的布尔函数的线路复杂度为$\Theta(2^n/n)$。研究这些专题的领域称为"开关理论"或"逻辑设计"。赛韦吉(Savage)[Sav72]初步建立了线路和图灵机之间的一些联系，并给出了线路和直线程序之间的紧密关系。

卡普(Karp)和利普顿(Lipton)[KL80]定义了类$\mathbf{P}_{poly}$并将它刻画为多项式时间纳言图灵机采用多项式建言时能够计算的语言族(定理6.18)。更一般地，他们还为任意复杂性类$\mathcal{C}$和函数$a:\mathbf{N}\to\mathbf{N}$定义了复杂性类$\mathcal{C}/a(n)$。但是，本书未采用这个定义，因为这种定义似乎没能刻画某些语言的建言的直观概念，这些语言包括**NP**∩**coNP**，**BPP**以及其他一些语言。

卡普和利普顿[KL80]最先给出了定理6.19的证明，但他们当时证明的结论是**PH**=$\mathbf{\Sigma}_2^p$，该结论稍弱于定理6.19的结论。他们将定理6.19的结论归功于西普赛尔(Sipser)。他们还陈述了定理6.20，并将定理证明归功于迈耶。

NC代表"Nick Class"，这个类最初由尼克·皮朋吉尔(Nick Pippenger)定义，后来由史蒂夫·库克(Steve Cook)根据他的名字命名。但是，**AC**中的"*A*"不代表某个人的名字，而是表示"Alternation"。雷顿(Leighton)的教科书[Lei91]讨论了**NC**算法类和许多与并行计算相关的问题。

大约在1989年，波普潘纳(Boppana)和西普赛尔(Sipcer)很好地综述了线路下界的相关知识[BS90]。幸运的是(或者说不幸的是)，该综述目前仍很好地反映了该领域的研究现
120 状。第14章将继续讨论线路下界。

习题

6.1　本题要求证明香农的一个结论，即任意函数$f:\{0,1\}^*\to\{0,1\}$均可以被规模为$O(2^n/n)$的线路计算。该界限被卢帕诺夫(Lupanov)[Lup58]改进到$\frac{2^n}{n}(1+o(1))$，其中$o(1)$是随n的增大而趋于0的项。上述改进还可以参见[Weg87，FM05]。

(**a**) 证明：任意函数f均可以被规模不超过$10\cdot 2^n$的线路计算。

(**b**) 改进(a)中的界限，证明：任意函数f均可以被规模不超过$1000\cdot 2^n/n$的线路计算。

6.2　证明：对于任意函数$f:\{0,1\}^*\to\{0,1\}$和$S\in\mathbf{N}$，函数f可以被规模为S的布尔线路计算当且仅当f可以被一个S行的直线程序计算，其中直线程序具有注记6.4给出的形式。

6.3　构造一个属于$\mathbf{P}_{poly}$但不属于**P**的可判定语言。

6.4　证明定理6.15。

6.5　(坎纳安[Kan81])证明：对于任意k，**PH**包含线路复杂度为$\Omega(n^k)$的所有语言。

6.6　将习题6.5中的**PH**替换为$\mathbf{\Sigma}_2^p$，如果你的证明过程未能证明修改后的结论，请给出新的证明过程。

6.7　(上界蕴含下界)证明：如果 **P**=**NP**，则 **EXP** 中存在需要规模为 $2^n/n$ 的线路才能计算的语言。

6.8　对于语言 $L\subseteq\{0,1\}^*$，如果存在多项式 p 使得 $|L\cap\{0,1\}^n|\leqslant p(n)$ 对任意 $n\in\mathbf{N}$ 成立，则称 L 是稀疏语言。证明：任意稀疏语言均属于 $\mathbf{P}_{poly}$。

6.9　(马哈尼定理(Mahaney's Theorem)[Mah80])证明：如果一个稀疏语言是 **NP** 完全的，则 **P**=**NP**。(这一结论加强了第 2 章习题 2.30 的结论。)

6.10　给出一个隐式对数空间可计算函数 f，该函数将任意 n 顶点图的邻接矩阵表示映射为该图的邻接表表示。你可以认为 n 顶点图的邻接表表示由 n 个位串构成，每个位串含 $O(n\log n)$ 个位，并且第 i 个串由图中第 i 个顶点的所有相邻顶点构成(在必要时可能需要填充一些 0 位)。

6.11　(一个仍未解决的问题)在如下比 $\mathbf{NP}\subseteq\mathbf{P}_{poly}$ 还强的假设条件下：**NP** 中的每个语言均存在线性规模的线路，能证明比 $\mathbf{PH}=\Sigma_2^p$ 更强的某种结论吗？

6.12　(a) 构造一个 **NC** 线路来计算给定的两个 $n\times n$ 矩阵 A，B 的乘积，其中 A，B 定义在有限域 **F** 上且 **F** 的大小至多是 n 的多项式。

(b) 构造一个 **NC** 线路，为给定的 $n\times n$ 矩阵 A 计算 A^n，其中 A 定义在有限域 **F** 上且 **F** 的大小至多是 n 的多项式。

(c) 得出结论：PATH 问题属于 **NC**(进而 **NL** 中的任意语言也都属于 **NC**)。 121

6.13　规则线路指的是任意非输入顶点的出度都等于 1 的线路。证明：一个语言可以被多项式规模的规则线路计算当且仅当该语言属于非一致 $\mathbf{NC}^1$。非一致 $\mathbf{NC}^1$ 是 $\mathbf{NC}^2$ 的变形，它的定义中不要求线路是对数空间算法产生的。

6.14　证明：$\mathbf{NC}^1\subseteq\mathbf{L}$。由此得出 $\mathbf{PSPACE}\neq\mathbf{NC}^1$。

6.15　证明定理 6.29。亦即，证明：如果 L 是 **P** 完全语言，则 $L\in\mathbf{NC}$(相应地 **L**)当且仅当 **P**=**NC**(相应地 **L**)。

6.16　(乔恩斯基算法(Csansky's Algorithm)[Csa76]：需要一些线性代数的知识)证明如下问题属于 **NC**：

$$\{\langle M,k\rangle: M \text{ 是行列式为 } k \text{ 的矩阵}\}$$

其中，M 的元素是整数，你也可以更一般地假设 M 是复数域 **C** 上的矩阵。

6.17　证明定理 6.32。亦即，**PH** 是所有具有常数深度的 DC 一致线路计算的语言构成的集合。

6.18　证明：**EXP** 恰好是规模为 2^{n^c} 的 DC 一致线路计算的所有语言构成的集合，其中 c 是依赖于具体语言的常数。

6.19　证明：如果线性规划存在高效平行算法，则 **P**=**NC**。 122

第7章

Computational Complexity, A Modern Approach

随机计算

机会主宰一切，未来的机会又不甚明了，我们为什么要恐惧呢？那最好就随随便便地活着吧，因为只能如此。

——索福克勒斯(Sophocles)，俄狄浦斯王(Oedipus Rex)[⊖]

这里，我们将定义并刻画一种方法，它可以用来处理数学物理中的一大类问题。本质上，该方法就是用统计方法来研究微分方程。

——N·梅特罗波利斯(N. Metropolis)，S·乌拉姆(S. Ulam)《蒙特卡罗方法》，1949

对于待求解的问题，我们不对它的实例服从何种分布做任何假设，而是将随机性融入算法本身之中……首先，出人意料的是，这样运用随机性也能得到高效的算法。这一点可以在两个例子上得到证实。第一个例子是从 $\mathbf{R}^k$ 的 n 个点中找出最邻近的点对。第二个例子是判定素数的一个极高效的算法。

——米歇尔·拉宾(Michael Rabin)，1976

迄今为止，我们将第1章中定义的图灵机当做计算的标准模型。但是，该模型却不允许在计算过程中做出随机选择，然而这正是计算中的客观需求。比如，多数程序设计语言均通过内置的随机数生成器在计算过程中实现随机选择。尽管科学家和哲学家仍就现实世界中是否存在真正的随机性争论不休，但是投掷一枚硬币(或者观察其他物理实验)的结果却确实表现出足够的随机性，这些实验结果相对于任何实践目的而言也确实是无法预测的。因此，考虑能够投掷硬币的算法(或图灵机)是有意义的。"算法(或图灵机)能够投掷硬币"指的是它能够使用产生二进制位的某种随机源。稍加思索就会发现，这种算法的研究其实一直在隐式地进行着。例如，经典统计学中的民意测验等基本方法就是这样的算法，其中民意测验旨在通过抽取人群的小规模随机样本来估计整个人群中的某种事实。同样，随机方法也可以作为一种自然的工具，用来模拟某些具有概率特性的现实系统，这样的系统包括核裂变或股票市场等。统计思想也一直被用于微分方程的研究，参见本章开头梅特罗波利斯和乌拉姆的引言。他们根据欧洲著名的赌博胜地的地名将这种算法命名为蒙特卡罗方法。

123

在过去的几十年中，随机方法被成功地应用到许多问题中，由此得到了这些问题的更简单或更高效的算法。这些问题涉及从数论到网络路由等众多的领域，并且从表面上看似乎都与概率毫无关系。本章将论及其中一些问题。本章不会论及随机数产生器的质量，这部分内容被推迟安排到第9章、第20章和第21章中。

站在复杂性理论家的角度看，本章的主要目的是理解能够投掷硬币的图灵机的能力。在7.1节中，我们先给出概率型计算的数学模型，然后定义复杂性类 **BPP**，它刻画了概率型算法能够高效求解的判定问题构成的集合[⊖]。7.2节将通过几个非平凡的概率型算法来

⊖ 索福克勒斯是古希腊三大悲剧家之一，俄狄浦斯王是其代表作《俄狄浦斯王》中的主人翁，他是一个盲目无知的国王。作者借用俄狄浦斯王的话来说明计算中可能存在随机性。——译者注

⊖ **BPP** 是"bounded-error probabilistic polynomial time"的缩写，参见本章注记。

说明随机方法有可能赋予算法设计者更强的能力。事实上，由于随机数产生器广泛存在（先忽略"它们产生的随机数到底有多好"这个问题），因此在刻画"有效计算"方面，类 **BPP** 及其姊妹类 **RP**、**coRP**、**ZPP** 与 **P** 有异曲同工之妙，尽管就此问题还存在一些争议。前面提到的那些算法将表明，**P** 是 **BPP** 的真子集，然而出人意料的是，也有一些证据令人相信 **BPP** 和 **P** 是同一个集合；参见第 20 章。

概率型算法的定义将允许它以小概率输出错误答案。初一看，读者可能会担心输出错误答案的算法不太实用。然而，7.4 节将阐述如何把错误概率减小到很小的数值。

本章还将研究 **BPP** 与前几章研究过的类 $\mathbf{P}_{poly}$ 和类 **PH** 之间的关系。

前几章研究过的许多概念也可以扩展到概率型算法上来。例如，7.6 节和 7.7 节将研究随机归约和概率型对数空间算法。它们分别类似于第 2 章中研究的归约和第 4 章中研究的对数空间算法。

随机性在复杂性理论中的作用远不止用于研究随机算法和 **BPP** 这样的复杂性类。事实上，整个复杂性理论体系中，随机性发挥关键作用的领域还包括密码学（见第 9 章）、交互式证明（第 8 章）和概率可验证明（第 11 章）。有时，随机性还用来证明一些表述形式似乎与随机性毫无关系的结论。因此，本章将为本书中的许多后续章节奠定基础。

本章及本书的后续章节将不断用到有穷样本空间上基本的概率概念。附录 A 快速地复习了这些概念。

7.1　概率型图灵机

随机算法能以某种方式进行随机选择，比如随机地将变量初始化为某个范围的一个整数。在实践中，随机算法通过随机数产生器来实现。容易证明（习题 7.1），随机数产生器只需能够产生随机二进制位即可，其中产生 0 的概率为 1/2，产生 1 的概率也为 1/2。这样的随机数产生器通常称为均匀硬币投掷。

124

同第 1 章用标准图灵机为确定型算法（即非概率型算法）建模一样，我们用概率型图灵机为随机算法建模。

定义 7.1　概率型图灵机（简称 PTM）是有两个转移函数 δ_0，δ_1 的图灵机。概率型图灵机 M 在输入 x 上运行时，每个步骤以 1/2 的概率选用转移函数 δ_0，以 1/2 的概率选用转移函数 δ_1。每一步的选择均独立于之前所做的所有选择。

概率型图灵机只能输出 1（接受）或 0（拒绝）。概率型图灵机在输入 x 上运行结束时的输出结果是一个随机变量，记为 $M(x)$。对于函数 $T: \mathbf{N} \to \mathbf{N}$，如果概率型图灵机 M 在任意输入 x 上运行时无论 M 做何种随机选择，它都在 $T(|x|)$ 个步骤内停机，则称 M 的运行时间为 $T(n)$。

回顾一下，2.1.2 节给出的非确定型图灵机也有两个转移函数。因此，它与概率型图灵机的结构十分相似。二者的区别在于工作方式。在概率型图灵机中，每个转移函数被选用的概率均为 1/2；因此，由时间 t 内构成的计算将存在 2^t 种可能的计算过程，每种计算过程发生的概率为 $1/2^t$。因此，$\Pr[M(x)=1]$ 就是结束时输出为 1 的计算过程占所有可能的计算过程的比例。概率型图灵机与非确定型图灵的区别主要在于如何解释所有可能的计算过程：如果存在一个输出为 1 的计算过程，则非确定型图灵机接受输入；而概率型图灵机考虑的却是输出为 1 的计算过程占所有可能的计算过程的比例。从概念的层面上看，概率型图灵机与非确定型图灵机有很大不同：概率型图灵机旨在对现实的计算装置进行建

模，这与图灵机相似，而不同于非确定型图灵机。

如下定义的类 **BPP** 用来刻画高效的概率型计算。下面，对语言 $L\subseteq\{0,1\}^*$ 和 $x\in\{0,1\}^*$，如果 $x\in L$ 则定义 $L(x)=1$；否则定义 $L(x)=0$。

定义 7.2（类 **BPTIME** 和类 **BPP**）　对于 $T:\mathbf{N}\to\mathbf{N}$ 和 $L\subseteq\{0,1\}^*$，如果对于任意 $x\in\{0,1\}^*$，概率型图灵机 M 在 x 上运行时无论 M 做何种随机选择，它都在 $T(|x|)$ 个步骤内停机且 $\Pr[M(x)=L(x)]\geqslant 2/3$，则称 M 在时间 $T(n)$ 内判定语言 L。

我们定义 **BPTIME**$(T(n))$ 是概率型图灵机在 $O(T(n))$ 时间内判定的所有语言构成的类，并定义 $\mathbf{BPP}=\bigcup_{c}\mathbf{BPTIME}(n^c)$。

注意，定义 7.2 中的概率型图灵机满足非常强的“排中”性。也就是说，在任意输入上，概率型图灵机要么至少以 2/3 的概率接受该输入，要么至少以 2/3 的概率拒绝该输入。这种性质使得定义 7.2 非常健壮，7.4 节将就此进行阐述。比如，我们将看到常数2/3可以替换为大于 1/2 的任意常数而不改变所定义的类 **BPTIME**$(T(n))$和 **BPP**，从这个意义上说，定义中的常数可以是任意的。我们还可以按其他方式修改定义中的概率型图灵机。比如，使用“不均匀”硬币，也就是头面朝上的概率不等于 1/2 的硬币；再比如要求概率型图灵机的期望运行时间是多项式时间。

然而，定义 7.2 允许概率型图灵机在输入 x 上的输出不等于 $L(x)$（亦即，输出错误答
125 案）的概率大于 0，该概率是相对于 M 在计算过程中所做的随机选择而言的。特别地，在任意输入 x 上，输出结果 $M(x)$ 等于正确值 $L(x)$ 的概率至少为 2/3。因此，同 **P** 一样，**BPP** 所刻画的仍然是最坏复杂性。

由于确定型图灵机可以视为一个特殊的概率型图灵机（即，两个转移函数相同的概率型图灵机），因此 **BPP** 显然包含 **P**。为了研究 **BPP** 与其他复杂性类之间的关系，采用如下的另一种定义将大有裨益。

正如对 **NP** 下的另一种定义，我们也可以利用确定型图灵机定义 **BPP**，其中每一步所需的掷币序列可作为图灵机的额外输入。

定义 7.3（**BPP** 的另一种定义）　对于语言 L，如果存在多项式时间图灵机 M 和多项式 $p:\mathbf{N}\to\mathbf{N}$ 使得 $\Pr_{r\in_R\{0,1\}^{p(|x|)}}[M(x,r)=L(x)]\geqslant 2/3$ 对任意 $x\in\{0,1\}^*$ 成立，则称 L 属于 **BPP**。

由上述定义易知 $\mathbf{BPP}\subseteq\mathbf{EXP}$，因为我们可以在 $2^{p(|x|)}$ 时间内枚举一个多项式时间概率型图灵机所有可能的随机选择。目前，研究者们仅证明了 **BPP** 介于 **P** 和 **EXP** 之间，还未能证明 **BPP** 是 **NEXP** 的真子集。

复杂性理论中一个仍未解决的核心问题是 $\mathbf{BPP}=\mathbf{P}$ 是否成立。读者根据前几章的惯例可能会猜测“复杂性理论家相信 $\mathbf{BPP}\neq\mathbf{P}$”。然而不是这样的！多数复杂性理论家实际上相信 $\mathbf{BPP}=\mathbf{P}$，也就是说，他们相信存在一种方法可以将任意概率型算法转换成（不投掷硬币的）确定型算法，并且算法的性能仅下降一个多项式因子。导致这种信念的原因将在第 19 章和第 20 章给出。

7.2　概率型图灵机示例

下面的几个例子表明，随机性可以在计算中发挥重要作用。本书其余章节还有大量这样的例子。

7.2.1 寻找中位数

数集$\{a_1, \cdots, a_n\}$的中位数是该集合中的一个元素x，它使得集合中至少有$\left\lfloor \frac{n}{2} \right\rfloor$个$a_i$小于等于$x$并且集合中也至少有$\left\lfloor \frac{n}{2} \right\rfloor$个$a_i$大于等于$x$。在许多计算中，为给定的数集找出中位数非常有用。一种寻找中位数的简单方法是，先将数集中的所有数排序，然后输出排序后的第$\left\lfloor \frac{n}{2} \right\rfloor$小的数；这种方法的时间开销是$O(n\log n)$[⊖]。下面，我们给出寻找中位数的一个简单的概率型算法，它的时间开销仅为$O(n)$。虽然寻找中位数存在线性时间的确定型算法，但下面给出的概率型算法仍然是目前最简单和最实用的算法。

我们的算法实际上求解了更一般的问题：对于任意k，找出给定数集中第k小的数。算法如下。

126

算法$\mathrm{F_{IND}K_{TH}E_{LEMENT}}(k, a_1, \cdots, a_n)$：

1. 随机选择$i \in [n]$并令$x = a_i$。
2. 扫描列表$\{a_1, \cdots, a_n\}$，统计满足$a_i \leqslant x$的a_i的个数m。
3. 如果$m = k$则输出x。
4. 否则，如果$m > k$，则将满足$a_i \leqslant x$的所有a_i拷贝到新列表L中，再执行$\mathrm{F_{IND}K_{TH}E_{LEMENT}}(k, L)$。
5. 否则（如果$m < k$），将满足$a_i > x$的所有a_i拷贝到新列表H中，再执行$\mathrm{F_{IND}K_{TH}E_{LEMENT}}(k-m, H)$。

显然，$\mathrm{F_{IND}K_{TH}E_{LEMENT}}(k, a_1, \cdots, a_n)$输出第$k$小的数，因此，剩下的问题就是分析算法的运行时间。直观上，我们希望每次递归调用时数的个数至少能缩减$n/10$个（因为在最坏情况下$k = n/2$，我们希望此时新列表的大小约为$\frac{3}{4}n$）。因此，如果用$T(n)$表示算法的运行时间，则它满足公式$T(n) = O(n) + T\left(\frac{9}{10}n\right)$，由此可得$T(n) = O(n)$。下面，我们形式地完成证明。

论断 7.4 对于$\mathrm{F_{IND}K_{TH}E_{LEMENT}}$的任意输入$k, a_1, \cdots, a_n$，令$T(k, a_1, \cdots, a_n)$表示算法在该输入上完成计算时的期望步骤数，令$T(n)$表示在所有长度为$n$的输入上$T(k, a_1, \cdots, a_n)$达到的最大值。则$T(n) = O(n)$。

证明 算法中除递归调用之外的其他操作可以在线性步数内完成，不妨将这些步数记为cn，其中c是常数。我们用归纳法证明$T(n) \leqslant 10cn$。事实上，取定一个输入$k, a_1, \cdots, a_n$。对于任意$j \in [n]$，我们选择$a_1, \cdots, a_n$中第j小的数作为x的概率为$\frac{1}{n}$。这样，在$j > k$的情况下将至多执行$T(j)$步操作，而在$j < k$的情况下将至多执行$T(n-j)$步操作。由此，可以看到

$$T(k, a_1, \cdots, a_n) \leqslant cn + \frac{1}{n}\left(\sum_{j>k} T(j) + \sum_{j<k} T(n-j)\right)$$

⊖ 这里，假设可以用单位代价完成每个数上的基本操作。要得到基本操作的个数，此处和后面的界限均需要再乘一个因子h，其中h是a_i的二进制形式的长度。

由于归纳假设 $T(j)\leqslant 10cj$ 对 $j<n$ 成立，代入上式得到

$$T(k,a_1,\cdots,a_n)\leqslant cn+\frac{10c}{n}\Big(\sum_{j<k}j+\sum_{j<k}(n-j)\Big)\leqslant cn+\frac{10c}{n}\Big(\sum_{j<k}j+kn-\sum_{j<k}j\Big)$$

利用 $\sum_{j<k}j\leqslant\frac{n(n-k)}{2}$ 和 $\sum_{j<k}j\geqslant\frac{k^2}{2}(1-o(1))\geqslant\frac{k^2}{2.5}$（$k$ 充分大之后）这两个事实，我们得到

$$T(k,a_1,\cdots,a_n)\leqslant cn+\frac{10c}{n}\Big(\frac{n(n-k)}{2}+kn-\frac{k^2}{2.5}\Big)=cn+\frac{10c}{n}\Big(\frac{n^2}{2}+\frac{kn}{2}-\frac{k^2}{2.5}\Big)$$

$$\leqslant cn+\frac{10c}{n}\cdot\frac{9n^2}{10}=10cn$$

[127] 其中最后一个不等式可以对 $k<n/2$ 和 $k>n/2$ 两种情况分别证得。 ■

7.2.2　概率型素性测试

素性测试问题要求判定给定的整数 N 是否是素数。在计算机诞生之前，人们早就开始寻找素性测试算法，这是由于数学家需要用素数来验证各种猜想[⊖]。最理想的素性测试算法的运行时间应该是 N 的二进制长度 $\log N$ 的某个多项式时间 $\mathrm{ploy}(\log N)$。几个世纪以来，人们一直未能找到这样的算法[⊜]。后来，在 20 世纪 70 年代，素性测试问题的高效概率型算法被发现，它成为了展示概率型算法效能的一个实例。注意，素性测试算法的研究最近已经获得突破，阿格拉沃尔(Agrawal)、卡亚尔(Kayal)和萨克塞纳(Saxena)[AKS04]给出了素性测试的一个多项式时间的确定型算法。

形式地，素性测试问题就是判定如下语言的成员资格：

$$\mathrm{PRIMES}=\{\langle N\rangle:N\text{ 是素数}\}$$

下面，我们概要地给出一个算法，说明 PRIMES 属于 **BPP**（事实上，它还属于 **coRP**，见 7.3 节）。对于任意自然数 N 和 $A\in[N-1]$，定义

$$QR_N(A)=\begin{cases}0 & \gcd(A,N)\neq 1\\ +1 & A\text{ 是模 }N\text{ 的二次剩余}\\ & \text{亦即，}A=B^2(\mathrm{mod}N)\text{，其中 }B\text{ 满足 }\gcd(B,N)=1\\ -1 & \text{其他}\end{cases}$$

我们将用到下面三个事实，它们均可以用初等数论加以证明（如，参见[Sho05]）：

- 对任意奇素数 N 和 $A\in[N-1]$，均有 $QR_N(A)=A^{(N-1)/2}(\mathrm{mod}N)$。
- 对任意奇素数 N 和任意 A，雅各比记号 $\left(\frac{N}{A}\right)$ 定义为 $\prod_{i=1}^{k}QR_{P_i}(A)$，其中 $P_1,\cdots,P_k$ 是 N 的所有（不必不同）素因子，亦即 $N=\prod_{i=1}^{k}P_i$。则雅各比记号可以在 $O(\log A\cdot\log N)$ 时间内被计算。
- 对任意奇合数 N，在所有满足 $\gcd(N,A)=1$ 的 $A\in[N-1]$ 中，至多有一半的 A 满足 $\left(\frac{N}{A}\right)=A^{(N-1)/2}(\mathrm{mod}N)$。

⊖ 一件有趣的轶事是，尽管高斯他自己就是一台高速计算机，但他在进行素性测试的时候还借助了另一台超级人力计算机——一个患自闭症却精于速算的人。

⊜ 事实上，在第 2 章引用的哥德尔写给冯·诺依曼的那封信中，他曾明确指出素性测试问题是一个值得研究的 **NP** 中仍未找到高效算法的问题。

这些事实放在一起，就得到一个简单的判定 N 的素性的算法。（不失一般性，假设 N 是奇数。）随机选择 $1 \leqslant A < N$。如果 $\gcd(N, A) > 1$ 或 $\left(\frac{N}{A}\right) \neq A^{(N-1)/2} (\mathrm{mod} N)$，则算法输出“合数”；否则，算法输出“素数”。如果 N 是素数，则该算法总是输出“素数”；但是，如果 N 是合数，则该算法将仅以大于等于 1/2 的概率输出“合数”。当然，可以重复常数次测试，将上述概率放大。 128

奇怪的是，素性测试问题的搜索形式（亦即找出给定合数 N 的质因数）其难度却高出许多。人们在因数分解问题上猜想的难度是许多密码系统的基础。尽管如此，我们将在第 10 章中看到，因数分解问题在量子计算机模型下可以被高效求解。

7.2.3　多项式恒等测试

现在，我们给出一个多项式时间概率型算法来求解一个至今未找到高效确定型算法的计算问题。问题表述如下。隐式地给定一个整系数多项式，要求判定该多项式是否恒等于 0。假设多项式是以代数线路的形式给定的。代数线路类似于布尔线路，只是将逻辑操作 $\wedge$，$\vee$，$\neg$ 替换为代数运算 $+$，$-$，$\times$（参见 16.1.3 节）。形式地，一个 n 变量代数线路是一个无环有向图，其中每个源顶点被集合 $\{x_1, \cdots, x_n\}$ 中的一个变量名字标记，而入度为 2 的非源顶点被集合 $\{+, -, \times\}$ 中的一个运算符标记。图中仅有一个汇顶点，它也被称为输出顶点。将输入整数置于代数线路的源顶点上，依次在每个顶点上执行标记的运算[⊖]，最终在汇顶点输出一个整数。可见，代数线路是从 $\mathbf{Z}^n$ 到 $\mathbf{Z}$ 的映射。通过简单的归纳，可以证明代数线路计算的函数 $f(x_1, \cdots, x_n)$ 可以表示成变量 $x_1, \cdots, x_n$ 上的多项式。语言 ZEROP定义为由计算零多项式的所有代数线路构成的集合。ZEROP 语言的成员资格判定问题也称为多项式恒等测试问题。因为判定两个线路 C，C' 是否计算了同一个多项式可以归约为判定由 $D(x_1, \cdots, x_n) = C(x_1, \cdots, x_n) - C'(x_1, \cdots, x_n)$ 定义的线路 D 是否属于 ZEROP。多项式恒等测试问题在复杂性理论中具有重要作用，参见第 8 章、第 11 章和第 20 章。

ZEROP 问题并不是一个平凡的计算问题，因为紧凑的线路可以表示含有很多单项式的多项式。例如，多项式 $\prod (1 + x_i)$ 完全展开之后有 2^n 个项，但却可以用规模为 $2n$ 的线路来计算。出人意料的是，存在一个简单而高效的概率型算法来判定 ZEROP 的成员资格。算法本质上依赖于如下的事实，该事实也被大家称为西瓦兹-齐佩尔引理（Schwartz-Zippel Lemma），其证明在附录 A 中（参见引理 A.36）。

引理 7.5　令 $p(x_1, x_2, \cdots, x_n)$ 是总次数[⊜]至多为 d 的非零多项式，S 是一些整数构成的有穷集合。如果 $a_1, a_2, \cdots, a_n$ 是可放回式地从 S 中随机选择的整数，则

$$\Pr[p(a_1, a_2, \cdots, a_n) \neq 0] \geqslant 1 - \frac{d}{|S|}$$ 129

规模为 m 的线路 C 至多包含 m 个乘法运算，因此 C 定义的多项式的次数至多为 2^m。由此得到如下简单的概率型算法：从 1 到 $10 \cdot 2^m$ 中随机选择 n 个数 $x_1, x_2, \cdots, x_n$（这需要使用 $O(n \cdot m)$ 个随机位），计算线路 C 在输入 $x_1, x_2, \cdots, x_n$ 上的值 y。如果 $y = 0$，则接受 C；否则，拒绝 C。显然，如果 $C \in$ ZEROP，则算法总接受 C。根据引理 7.5，如果

⊖ 线路也可以使用常数，如 0，1 以及其他数均可以作为常数。但不会影响此处的讨论。

⊜ 单项式 $x_1^{e_1} \cdot x_2^{e_2} \cdot \cdots \cdot x_n^{e_n}$ 的总次数是 $e_1 + e_2 + \cdots + e_n$。多项式的总次数是其中单项式总次数的最大值。

$C \notin \mathrm{ZEROP}$，则算法拒绝 C 的概率至少为 9/10。

然而，上述算法还存在一个问题。由于线路 C 表示的多项式的次数可能高达 2^n，因此输出值 y 以及计算过程中出现的中间值可能达到 $(10 \cdot 2^n)^{2^n}$，表示和记录这些值可能需要指数个位！

这个问题可以用指纹技术来克服。其思想是，随机从 $[2^{2n}]$ 中选择一个数 k 并计算 C 在输入 $x_1, x_2, \cdots, x_n$ 上的值模 k 的余数。也就是说，不直接计算 $y=C(x_1, x_2, \cdots, x_n)$，而是计算 $y(\mathrm{mod}k)$。显然，如果 $y=0$，则 $y(\mathrm{mod}k)$ 也等于 0。另一方面，我们断言：如果 $y \neq 0$，则 k 不能整除 y 的概率至少为 $\delta=\frac{1}{4m}$。这足以解决问题，因为可以重复运行上述算法 $O(1/\delta)$ 次，并且在所有测试中 $y=0(\mathrm{mod}k)$ 均成立时才接受 C。事实上，假设 $y \neq 0$，用 $B=\{p_1, \cdots, p_l\}$ 表示 y 的所有相异素因数构成的集合。只需证明，算法随机选择的 k 是不属于 B 的素数的概率至少为 δ。为此，由素数定理可知，当 m 充分大时，$[2^{2n}]$ 中至少有 $\frac{2^{2n}}{2m}$ 个素数。由于 y 至多有 $\log y \leqslant 5m2^n = o\left(\frac{2^{2n}}{2m}\right)$ 个素因数，因此，当 m 充分大时，$[2^{2n}]$ 中不属于 B 的素数 k 至少有 $\frac{2^{2n}}{4m}$ 个。这意味着，算法随机选择的 k 是不属于 B 的素数的概率至少为 $\frac{1}{4m}=\delta$。

7.2.4　二分图的完美匹配测试

设 $G=(V, E)$ 是部集大小相等的二分图。也就是说，$V=V_1 \cup V_2$ 且 $E \subseteq V_1 \times V_2$，其中 V_1，V_2 是两个大小相同的不相交的集合。G 的一个完美匹配是一个边子集 $E' \subseteq E$，它使得 G 的任意顶点恰好是 E' 中一条边的端点。换一种说法，令 $n=|V_1|=|V_2|$ 并将两个部集都视为集合 $[n]$，则完美匹配 E' 是一个置换 $\sigma: [n] \to [n]$，它将 $i \in [n]$ 映射到 $j \in [n]$ 使得 $\overline{ij} \in E'$。人们已经设计了几个确定型算法来判定一个图是否存在完美匹配。这里，我们给出一个利用西瓦兹-齐佩尔引理设计的概率型算法（该算法源自洛瓦兹(Lovász)）。

设 $G=(V, E)$ 是如上所述的 $2n$ 顶点二分图，令 X 是 $n \times n$ 的实数矩阵，其中第 (i, j) 个位置的元素 $X_{i,j}$ 等于变量 $x_{i,j}$，如果 $\overline{ij}$ 是 E 中的一条边；否则，$X_{i,j}=0$。回顾一下，矩阵 A 的行列式的定义为

$$\det(A)=\sum_{\sigma \in S_n}(-1)^{\mathrm{sgn}(\sigma)}\prod_{i=1}^{n}A_{i,\sigma(i)}$$

其中 S_n 是 $[n]$ 的所有置换组成的集合，置换 σ 中的逆序 (i, j) 满足 $i<j$ 且 $\sigma(i)>\sigma(j)$，
130 $\mathrm{sgn}(\sigma)$ 表示置换 σ 中逆序个数的奇偶性；参见 A.5 节。因此，$\det(X)$ 是变量 $\{x_{i,j}\}_{\overline{ij} \in E}$ 的 n 次多项式，其中每个单项式对应 G 的一个完美匹配。换句话说，G 中存在完美匹配当且仅当 $\det(X)$ 不是恒等于 0 的多项式。现在，虽然 $\det(X)$ 可能包含指数个单项式，但是当 $x_{i,j}$ 的值任意取定之后却可以用任意一个已知的行列式算法来计算 $\det(X)$。

再联立引理 7.5，就得到洛瓦兹随机算法：从 $[2n]$ 中随机选择每个 $x_{i,j}$ 的值代入 X，计算 X 的行列式 $\det(X)$。如果 $\det(X) \neq 0$，则接受 G；否则，拒绝 G。除了简洁性之外，该算法的另一个优点是它可以高效地并行化，这只需利用计算行列式的 **NC**-算法，参见习题 6.16。

7.3 单面错误和"零面"错误：RP、coRP、ZPP

类 **BPP** 刻画的概率型算法可能会发生双面错误，也就是说，在判定语言 L 的成员资格时，概率型算法有可能在 $x\in L$ 时输出 0，也可能在 $x\notin L$ 时输出 1。因此，这种概率型算法被称为是带双面错误的概率型算法。但是，许多概率型算法只犯单面错误。比如，某些概率型算法绝不会在 $x\notin L$ 时输出 1，但有可能在 $x\in L$ 时输出 0。概率型算法的这种性质由类 **RP** 来刻画。下面是该类的定义。

定义 7.6 **RTIME**($T(n)$)包含满足如下条件的所有语言 L：存在 $T(n)$时间的概率型图灵机 M 使得

$$x\in L\Rightarrow \Pr[M(x)=1]\geqslant 2/3$$
$$x\notin L\Rightarrow \Pr[M(x)=1]=0$$

定义 $\mathbf{RP}=\bigcup_{c>0}\mathbf{RTIME}(n^c)$。

注意，**RP**$\subseteq$**NP**，因为概率型图灵机 M 在输入 x 上使 $M(x)=1$ 的每个计算过程均提供了一个 $x\in L$ 的证明。相比之下，我们还不能确定 **BPP**$\subseteq$**NP** 是否成立。类 **coRP**$=\{L\mid \overline{L}\in \mathbf{RP}\}$刻画了犯"另一面错误"的单面错误概率型算法：也就是说，**coRP** 中的概率型算法可能在$x\notin L$时输出 1，但绝不会在 $x\in L$ 时输出 0。

"零面"错误。对于概率型图灵机 M 和输入 x，定义随机变量 $T_{M,x}$ 表示 M 在输入 x 上的运行时间，即 $\Pr[T_{M,x}=T]=p$ 表示的含义是，在 M 以 x 为输入时所做的所有随机选择中，M 在 T 个步骤内停机的概率为 p。对于任意 $x\in\{0,1\}^*$，如果数学期望 $\mathrm{E}(T_{M,x})$至多为$T(|x|)$，则称 M 的期望运行时间为 $T(n)$。现在，我们定义不犯错误的概率型图灵机，这种概率型图灵机也称为"零错误"概率型图灵机。

定义 7.7 类 **ZTIME**($T(n)$)包含满足如下条件的所有语言 L：存在期望运行时间为 $O(T(n))$的概率型图灵机 M，使得在任意输入 x 上，无论 M 在 x 上何时停机，M 的输出结果 $M(x)$都恰为 $L(x)$。

定义 $\mathbf{ZPP}=\bigcup_{c>0}\mathbf{ZTIME}(n^c)$。 131

下述定理的结论稍微有些出人意料，因为非确定型图灵机上相应的问题(即，**P**$=$**NP**$\cap$**coNP** 是否成立)仍未解决。

定理 7.8 **ZPP**$=$**RP**$\cap$**coRP**。

我们将定理 7.8 的证明留给读者(见习题 7.6)。综上所述，各种概率型复杂性类之间满足下列关系：

$$\mathbf{ZPP}=\mathbf{RP}\cap\mathbf{coRP}$$
$$\mathbf{RP}\subseteq\mathbf{BPP}$$
$$\mathbf{coRP}\subseteq\mathbf{BPP}$$

7.4 定义的健壮性

在定义 **P** 和 **NP** 时，我们阐述了定义的健壮性，表明了即便是银河系之外的外星人研究这些概念时也会采用同样的定义。本节为概率型计算阐释类似的问题。

7.4.1　准确度常数的作用：错率归约[⊖]

常数 2/3 的选取颇为随意。下面，我们证明常数 2/3 可以替换为大于 1/2 的任意常数，甚至还可以替换为 $1/2+n^{-c}$，其中 c 是一个大于 0 的常数。

引理 7.9　对 $c>0$，如果 $\mathbf{BPP}_{1/2+n^{-c}}$ 是由满足如下条件的所有语言 L 构成的类：存在多项式时间的概率型图灵机 M 使得 $\Pr[M(x)=L(x)]\geqslant 1/2+|x|^{-c}$ 对任意 $x\in\{0,1\}^*$ 成立，则 $\mathbf{BPP}_{1/2+n^{-c}}=\mathbf{BPP}$。

由于 $\mathbf{BPP}\subseteq\mathbf{BPP}_{1/2+n^{-c}}$ 显然成立；因此，要证明引理，只需证明成功概率等于 $1/2+n^{-c}$ 的概率型图灵机可以转换成一个成功概率等于 2/3 的概率型图灵机。为此，我们证明一个更强的结论：任意一个成功概率等于 $1/2+n^{-c}$ 的概率型图灵机可以转换成一个成功概率接近于 1 的概率型图灵机。

定理 7.10（**BPP 的错率归约**）　设 L 是一个语言。假设存在多项式时间的概率型图灵机 M 使得 $\Pr[M(x)=L(x)]\geqslant 1/2+|x|^{-c}$ 对任意 $x\in\{0,1\}^*$ 成立。那么，对于任意 $d>0$，存在多项式时间的概率型图灵机 M' 使得 $\Pr[M(x)=L(x)]\geqslant 1-2^{-|x|^d}$ 对任意 $x\in\{0,1\}^*$ 成立。

证明　概率型图灵机 M' 如下工作：在任意输入 $x\in\{0,1\}^*$ 上，先模拟 $M(x)$ 运行 $k=8|x|^{2c+d}$ 次，得到 k 个输出 $y_1,\cdots,y_k\in\{0,1\}$。如果 M 的多数输出是 1，则 M' 输出 1；否则，M' 输出 0。

为了分析概率型图灵机 M'，对任意 $i\in[k]$ 定义一个随机变量 X_i。如果 $y_i=L(x)$，则 X_i 等于 1；否则，$X_i=0$。注意，$X_1,\cdots,X_k$ 是独立的布尔随机变量，且 $\mathrm{E}[X_i]=\Pr[X_i=1]\geqslant p$，其
132 中 $p=1/2+|x|^{-c}$。切尔诺夫界（参见推论 A.15）表明，对于充分小的 δ 有

$$\Pr\Big[\Big|\sum_{i=1}^{k}X_i-pk\Big|>\delta pk\Big]<\mathrm{e}^{-\frac{\delta^2}{3}pk}$$

这里 $p=1/2+|x|^{-c}$，并且令 $\delta=|x|^{-c}/2$ 就可以保证：如果 $\sum_{i=1}^{k}X_i\geqslant pk-\delta pk$，则 M' 输出的答案就是正确的。因此，M' 输出错误答案的概率不超过切尔诺夫界给出的概率，亦即不超过

$$\mathrm{e}^{-\frac{1}{3}\left(\frac{1}{2|x|^c}\right)^2 8|x|^{2c+d}}\leqslant 2^{-|x|^d}$$

■

类似的结论对单面错误类 **RP** 和 **coRP** 也成立，并且证明过程也更简单（参见习题 7.4）。此时，常数 2/3 甚至可以替换为比 1/2 小的常数。

这些错率归约结论表明，我们可以采用具有中等大小成功概率的概率型算法，并且这种概率型算法还可以转换成具有压倒性成功概率的概率型算法。事实上，即使对中等大小的 n 值，阶为 2^{-n} 的错误概率也非常小，这种概率型算法适用于任何实际应用，同确定型算法一样有效。

随机性的高效重复

定理 7.10 的证明利用 $O(k)$ 次独立的重复，将成功概率为 2/3 的概率型算法转换为成

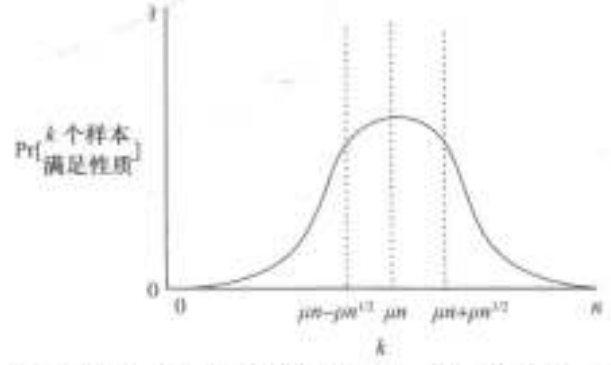
⊖ 原文是"error reduction"，直译为"错误归约"易引起误解，也不利于反映该概念与概率计算的关系，故采用"错率归约"，反映了归约过程减小"错误概率"之意。——译者注

功概率为 $1-2^{-k}$的概率型算法。因此，如果原概率型算法使用 m 个随机硬币，则新的概率型算法将使用 $O(km)$个随机硬币。出人意料的是，我们可以节省一些随机硬币：存在一种用 $O(m+k)$个随机硬币的转换方法达到相同的错率归约。这种转换方法将在第 21 章中介绍(参见 21.2.5 节)。

注记 7.11（切尔诺夫界和统计估计）　切尔诺夫界有时也被称为其他名字，它被广泛地应用于计算机科学的各个领域和其他科学领域。典型的应用场景如下。在对象全域 U 中，满足某种性质的对象所占比例为 μ，而 μ 是待估计的量。例如，在定理 7.10 的证明中，全域是某个概率型算法可能的 2^m次硬币投掷，我们想知道的是其中有多少次硬币投掷使得算法接受其输入。再如，全域 U 也可以是所有美国公民，而我们想知道的是养狗的公民有多少。

估算比例 μ 的一个自然的方法是，先从全域中随机、独立地抽取 n 个样本，然后找出其中满足性质的样本的个数 k，最后输出 k/n 作为 μ 的猜测值。当然，在小样本上获得的猜测值不太可能等于准确答案。比如，真正的养狗人可能占 10%，但在 1000 个样本中可能仅发现了 99 个养狗人(即 9.9%)。因此，我们仅将目标定为为某个 $\varepsilon>0$ 在 $\pm\varepsilon$ 的误差范围内估计实际比例 μ。尽管允许一定的误差范围，但仍可能不走运，亦即我们的抽样可能不具有代表性(例如，有可能抽取的 1000 个样本均是养狗人)。因此，我们允许有较小的失败概率 δ，也就是估计值不在区间$[\mu-\varepsilon,\ \mu+\varepsilon]$内的概率为 δ。我们自然要问，为了以至少 $1-\delta$的概率在 $\pm\varepsilon$ 的误差范围内估计 μ，要使用多少的样本呢？切尔诺夫界告诉我们，(将 μ 视为常数)样本数量为 $O(\log(1/\delta)/\varepsilon^2)$。

取 $\rho=\log(1/\delta)$，切尔诺夫界意味着，k 偏离 μn 超过 $\rho\sqrt{n}$的概率随着 ρ 的增大而指数地下降。也就是说，上述概率表现出著名的“钟形曲线”。



概率的这种指数下降现象将在本书中多次使用；定理 7.14 证明 $\mathbf{BPP}\subseteq\mathbf{P}_{poly}$的过程就是一个例子。

7.4.2　期望运行时间与最坏运行时间

在定义 **RTIME**($T(n)$)和 **BPTIME**($T(n)$)时，我们曾要求概率型图灵机无论在何种随机选择下均在 $T(n)$时间内停机。然而，在定义 **ZPP** 时(定义 7.7)，我们又转而使用期望运行时间。可以证明，采用期望运行时间和最坏运行时间将等价地定义相同的类。事实上，期望运行时间为 $T(n)$的概率型图灵机 M 可以转换成最坏运行时间为 $100T(n)$的概率型图灵机 M'。这只需增加一个计数器，当机器运行时间超出 $100T(n)$步之后让机器停机，并输出一个任意的结果。根据马尔可夫不等式(Markov's Inequality，参见引理 A.7)，M 的运行时间超过 $100T(n)$的概率至多为 1/100，这种转换至多只能使接受概率改变 1/100。

7.4.3　使用比均匀硬币投掷更具一般性的随机选择

可以设想，现实的计算机也可以使用头面朝上的概率等于 ρ 而不等于 1/2 的"硬币"。我们将这样的硬币称为 ρ-硬币。事实上，还可以构想基于量子计算机的随机源，此时 ρ 是一个无理数，比如 $1/e$。这样的硬币能赋予概率型算法新的能力吗？下面的引理表明，不能，至少在 ρ 可以被高效计算时不能。习题的结论表明，如果 ρ 不能被高效计算，则 ρ-硬币确实会赋予概率型算法更多的能力。

133

引理 7.12　只要 ρ 的第 i 个位能够在 $\mathrm{poly}(i)$ 时间内计算，则满足 $\Pr[Heads]=\rho$ 的硬币可以被期望时间为 $O(1)$ 的概率型计算机计算。

证明　令 ρ 的二进制表达式为 $0.p_1p_2p_3\cdots$ 概率型图灵机 M 逐位地产生随机位序列 b_1，

134　b_2，…，其中 b_i 是第 i 步产生的随机位。如果 $b_i<p_i$，则 M 输出"头面"，然后停机；如果 $b_i>p_i$，则 M 输出"背面"，然后停机；否则，M 进入第 $i+1$ 步。显然，M 进入 $i+1$ 步当且仅当 $b_j=p_j$ 对任意 $j\leqslant i$ 成立；该事件发生的概率为 $1/2^i$。因此，"头面"的概率为 $\sum_i p_i\frac{1}{2^i}$，它恰好等于 ρ。而且，M 的期望运行时间为 $\sum_i i^c\cdot\frac{1}{2^i}$。无论常数 c 等于什么，上述无穷和将以另一个常数为上界(参见习题 7.2)。■

反过来，仅用 ρ-硬币的概率型算法也不比标准概率型算法能力弱。

引理 7.13　(冯·诺依曼[vN51])　满足 $\Pr[Heads]=1/2$ 的硬币可以被使用多个 ρ-硬币的概率型图灵机在 $O\left(\frac{1}{\rho(1-\rho)}\right)$ 的期望时间内模拟。

证明　我们构造一个图灵机 M，让它投掷 ρ-硬币，并输出一个 1/2-硬币。图灵机 M 投掷一对 ρ-硬币，直到两枚硬币朝向不同面才停止(亦即，直到出现"头面-背面"或者出现"背面-头面"才停止)。此时，如果第一枚硬币是"头面"，则 M 输出"头面"；否则，M 输出"背面"。

一对硬币出现"头面-背面"的概率是 $\rho(1-\rho)$，而出现"背面-头面"的概率是 $(1-\rho)\rho=\rho(1-\rho)$。因此，在每一步中，$M$ 停机的概率是 $2\rho(1-\rho)$；在 M 已停机的条件下，M 等概率地输出"头面"和"背面"，这就是说 M 输出了一枚均匀硬币。注意，在模拟过程中，无需具体地知道 ρ 的值。■

弱随机源

物理学家(和哲学家)仍不十分肯定世界上是否真存在随机性；即便世界上真存在随机性，他们也不清楚计算机是否能无限地随机独立地投掷硬币。可以想象，我们或许只能让计算机使用非完美的随机源，也就是说，这种随机源产生的随机位虽然在一定程度上难以预测，但这些随机位不是独立的。在第 21 章，我们将展示如何用非完美随机源来运行在完美随机源的假设下设计的概率型算法。

7.5　BPP 同其他复杂性类之间的关系

本节将证明 $\mathbf{BPP}\subseteq\mathbf{P}_{/\mathrm{poly}}$，进而得到 $\mathbf{P}\subseteq\mathbf{BPP}\subseteq\mathbf{P}_{/\mathrm{poly}}$。而且，我们还将证明 $\mathbf{BPP}\subseteq\Sigma_2^p\cap\Pi_2^p$；因而，如果 $\mathbf{NP}=\mathbf{P}$，则有 $\mathbf{BPP}=\mathbf{P}$。当然，由于我们不相信 $\mathbf{P}=\mathbf{NP}$，因此 $\mathbf{BPP}\neq\mathbf{P}$ 是否有可能成立仍有待研究。然而，正如上面所述(第 19 章和第 20 章还将细致地讨论)，如果一些貌似可信的复杂性理论的猜想成立，则 $\mathbf{BPP}=\mathbf{P}$。因此，我们怀疑 **BPP** 和 **P** 是同一个类；

进而，由时间分层定理可知，**BPP** 是 $\mathbf{DTIME}(n^{\log n})$ 的真子集。然而，研究者们目前还未能证明 **BPP** 是 **NEXP** 的真子集。

7.5.1　$\mathbf{BPP} \subseteq \mathbf{P}_{/poly}$

现在，我们证明 **BPP** 语言均存在多项式规模的线路。与定理 6.19 一起，这说明 3SAT 不能在概率型多项式时间内求解，除非多项式分层坍塌。

定理 7.14　([Adl78])$\mathbf{BPP} \subseteq \mathbf{P}_{/poly}$。

证明　假设 $L \in \mathbf{BPP}$，则由 **BPP** 的第二种定义和定理 7.10 中的错率归约可知，存在在长度为 n 的输入上使用长度为 m 的随机位串 r 的图灵机 M 使得 $\Pr[M(x, r) \neq L(x)] \leqslant 2^{-n-1}$ 对任意 $x \in \{0, 1\}^n$ 成立。对于 $r \in \{0, 1\}^m$，如果 $M(x, r) \neq L(x)$，则称 r 是输入 x 的劣性随机串；否则，称 r 是 x 的良性随机串。对于任意 x，至多存在 x 的 $\frac{2^m}{2^{n+1}}$ 个劣性随机串。对所有 $x \in \{0, 1\}^n$ 做和，至多存在 $2^n \cdot \frac{2^m}{2^{n+1}} = 2^m/2$ 个随机串 r 对某个 x 是劣性的。特别地，存在一个随机串 $r_0 \in \{0, 1\}^m$ 对任意 $x \in \{0, 1\}^n$ 均是良性随机串。在 r_0 上添加连线，可以构造一个线路 C，其规模至多是 M 运行时间的平方，它在输入 x 上输出 $M(x, r_0)$。线路 C 使得 $C(x) = L(x)$ 对任意 $x \in \{0, 1\}^n$ 成立。 ■

135

7.5.2　$\mathbf{BPP} \subseteq \mathbf{PH}$

初看起来，**BPP** 似乎与多项式分层毫无关系，因此下述定理多少有些出人意料。

定理 7.15　(西普塞尔-高奇定理(Sipser-Gács Theorem))　$\mathbf{BPP} \subseteq \boldsymbol{\Sigma}_2^p \cap \boldsymbol{\Pi}_2^p$。

证明　只需证明 $\mathbf{BPP} \subseteq \boldsymbol{\Sigma}_2^p$，因为 **BPP** 在补集运算下封闭，亦即 $\mathbf{BPP} = \mathbf{coBPP}$。

假设 $L \in \mathbf{BPP}$，则由 **BPP** 的第二种定义和定理 7.10 中的错率归约可知，存在判定语言 L 的多项式时间的确定型图灵机 M，它在长度为 n 的输入上使用 $m = \mathrm{poly}(n)$ 个随机位，并满足

$$x \in L \Rightarrow \Pr[M(x, r) \text{ 接受}] \geqslant 1 - 2^{-n}$$

$$x \notin L \Rightarrow \Pr[M(x, r) \text{ 接受}] \leqslant 2^{-n}$$

对 $x \in \{0, 1\}^n$，令 S_x 是由使得 M 接受输入序对 (x, r) 的所有 r 构成的集合。于是，要么 $|S_x| \geqslant (1-2^{-n})2^m$，要么 $|S_x| \leqslant 2^{-n}2^m$，具体是哪种情况取决于 $x \in L$ 是否成立。下面，我们说明如何借助两个量词来判定具体是哪种情况发生。

136

对集合 $S \subseteq \{0, 1\}^m$ 和位串 $u \in \{0, 1\}^m$，我们用 $S+u$ 表示 S 在 u 上的“平移”，亦即 $S+u = \{x+u : x \in S\}$，其中 $+$ 是模 2 的向量加法(也就是按位异或操作)。令 $k = \left\lfloor \frac{m}{n} \right\rfloor + 1$。定理 7.15 可由下列两个论断得出(参见图 7-1)。

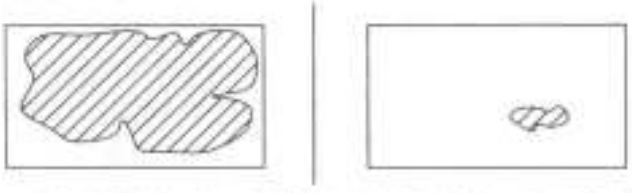


图 7-1　满足“$M(x, r)$=接受”的所有 r 构成的集合只有两种大小，要么该集合包含 $\{0, 1\}^m$ 中几乎所有的串，要么该集合仅含 $\{0, 1\}^m$ 中很小比例的串。在前一种情况下，集合的少数几个随机“平移”很可能覆盖整个集合 $\{0, 1\}^m$，在后一种情况下，少数几个平移不能覆盖 $\{0, 1\}^m$

论断 1：对于任意满足 $|S|\leqslant 2^{n-n}$ 的集合 $S\subseteq\{0,1\}^n$ 和任意 k 个向量 $u_1,\cdots,u_k$，有

$$\bigcup_{i=1}^{k}(S+u_i)\neq\{0,1\}^n$$

证明　由于 $|S+u_i|=|S|$，故由合并界限可得 $\left|\bigcup_{i=1}^{k}(S+u_i)\right|\leqslant k|S|<2^n$ 对充分大的 n 成立。■

论断 2：对于任意满足 $|S|\geqslant(1-2^{-n})2^n$ 的集合 $S\subseteq\{0,1\}^n$，存在 k 个向量 $u_1,\cdots,u_k$ 使得

$$\bigcup_{i=1}^{k}(S+u_i)=\{0,1\}^n$$

证明　该论断可由概率方法如下证得。只需证明，如果 $u_1,\cdots,u_k$ 是随机独立地选取的，则 $\Pr[\bigcup_{i=1}^{k}(S+u_i)=\{0,1\}^n]>0$。事实上，对于 $r\in\{0,1\}^n$，令 B_r 表示随机事件 $r\notin\bigcup_{i=1}^{k}(S+u_i)$，这是我们不希望发生的"坏事件"。于是，只需证明，$\Pr[\exists_{r\in\{0,1\}^n}B_r]<1$。该式可由合并界限得到，只要能证得 $\Pr[B_r]<2^{-n}$ 对任意 r 成立。但是，$B_r=\bigcap_{i\in[k]}B_i$，其中 B_i 表示事件 $r\notin(S+u_i)$，或等价地，$r+u_i\notin S$（注意，在模 2 的加法运算下恒有 $a+b=c\Leftrightarrow a=c+b$）。此时，$r+u_i$ 可视为均匀分布于 $\{0,1\}^n$ 中的随机位串，因而 $r+u_i$ 属于 S 的概率至少为 $1-2^{-n}$。并且，所有事件 B_i 在 i 为不同值时是独立的，这意味着 $\Pr[B_r]=\Pr[B_i]^k\leqslant 2^{-nk}<2^{-n}$。■

论断 1 和论断 2 表明，$x\in L$ 当且仅当下述断言为真：

$$\exists u_1,\cdots,u_k\in\{0,1\}^n\ \forall r\in\{0,1\}^n r\in\bigcup_{i=1}^{k}(S+u_i)$$

或等价地，

$$\exists u_1,\cdots,u_k\in\{0,1\}^n\ \forall r\in\{0,1\}^n\ \bigvee_{i=1}^{k}M(x,r+u_i)\text{ 接受}$$

由于 k 是 $\mathrm{poly}(n)$，故上式是一个 $\boldsymbol{\Sigma}_2^p$ 计算。因此，我们证得 $L\in\boldsymbol{\Sigma}_2^p$。■

7.5.3　分层定理与完全问题

读者可能会问，**BPP** 存在完全问题吗？或者，概率型计算有分层定理吗？下面，我们讨论这两个问题。

BPP 有完全问题吗

虽然 **BPP** 是一个很自然的类，但它与之前见过的其他类却有所不同。比如，据我们所知，**BPP** 还没有完全语言。造成这种窘境的原因是，**BPTIME** 的定义性质是基于机器的语义的，即这种机器要么以至少 2/3 的概率接受一个输入，要么以至多 1/3 的概率拒绝该输入。测试图灵机是否具有这种性质是不可判定的。相比之下，非确定型图灵机的定义性质是基于语法的，也就是说，给定一个位串，我们可以轻易地判断它是否是某个非确定型图灵机的有效编码。为基于语法定义的复杂性类寻找完全问题明显地易于为基于语义定义的复杂性类寻找完全问题。例如，在寻找 **BPP**-完全语言时，很自然地会进行如下尝试：定义语言 L 是所有如下三元组 $\langle M,x,1^t\rangle$ 构成的集合：M 以 x 为输入时将在 t 个步骤内 137

以至少 2/3 的概率输出 1。语言 L 确实是 **BPP**-难的，但难以证明它属于 **BPP** 的，因为对于$\langle M, x, 1^n\rangle \notin L$，我们有可能证得(比方说)$\Pr[M(x)=1]=1/2$，该概率大于 1/3。事实上，我们将在第 17 章中看到，该语言实际上是 $\#\mathbf{P}$-完全的，因而它不可能属于多项式分层的任何一层，除非多项式分层坍塌。然而，正如我们相信的那样，如果 **BPP**$=$**P**，则 **BPP** 确实会存在完全问题(因为 **P** 存在完全问题)。

BPTIME 有分层定理吗

BPTIME(n^2)中的每个问题也属于 **BPTIME**(n)吗？应该不是，这一结论用第 3 章中的对角线方法似乎就能证明。然而，目前人们连 **BPTIME**$(n)\neq$**BPTIME**$(n^{\log^* n})$也无法证得。标准的对角线方法失效了，同样这也是由于 **BPTIME** 的定义性质是语义的。尽管如此，最近人们在一些密切相关的复杂性类上证明分层定理已经有了一些进展(参见本章注记)。

7.6　随机归约

既然已经定义了随机算法，我们就可以定义语言之间的随机归约。随机归约的概念在研究某些复杂性时非常有用(参见第 8 章和第 17 章)。

定义 7.16　*对于语言 B 和语言 C，如果存在多项式时间的概率型图灵机 M 使得 $\Pr[B(M(x))=C(x)]\geqslant 2/3$对任意 $x\in\{0,1\}^*$ 成立，则称语言 B 被多项式时间随机归约到语言 C，记为 $B\leqslant_r C$。*

尽管随机归约不满足传递性，但是从下述意义上来说该概念仍是有用的：如果 $C\in$ **BPP** 且 $B\leqslant_r C$，则 $B\in$ **BPP**。这一观察也使我们意识到，既然 **BPP** 和 **P** 在刻画有效计算的概念时差不多，那么在 **NP** 完全性的定义中就可以将确定归约替换为随机归约。回顾一下，库克-勒维定理表明，**NP** 可以定义为集合$\{L:\ L\leqslant_p 3\mathrm{SAT}\}$。将这一定义中的"确定型多项式时间归约"替换为"随机归约"，则将得到一个稍有区别的类。

定义 7.17　**BP · NP**$=\{L:\ L\leqslant_r 3\mathrm{SAT}\}$。

我们在习题中讨论 **BP · NP** 的性质，包括$\overline{3\mathrm{SAT}}\in$ **BP · NP** 是否有可能成立。

第 17 章将给出随机归约的一个意义明确的应用。在那里，我们要使用随机归约的一种变形，将 3SAT 随机归约到 3SAT 的一种特殊形式，这种特殊 3SAT 要求每个布尔公式要么是不可满足的，要么存在唯一的满足性赋值。

138

7.7　空间受限的随机计算

第 4 章定义的空间受限计算也可以扩展到概率型计算上来。在长度为 n 的输入上，如果概率型图灵机工作带上的非空白符在任何计算分支上都不超过 $O(S(n))$个，则称该概率型图灵机的空间开销为 $S(n)$。最有意义的是空间开销为 $O(\log n)$的情况。类 **BPL** 和类 **RL** 分别是与第 4 章中定义的类 **L** 相对应的双面错误概率型复杂性类和单面错误概率型复杂性类。

定义 7.18　*(类 **BPL** 和类 **RL**)　称语言 L 属于 **BPL**，如果存在 $O(\log n)$空间的概率型图灵机 M 使得 $\Pr[M(x)=L(x)]\geqslant 2/3$。*

*称语言 L 属于 **RL**，如果存在 $O(\log n)$空间的概率型图灵机 M 满足：(1)如果 $x\in L$，则 $\Pr[M(x)=1]\geqslant 2/3$；(2)如果 $x\notin L$，则 $\Pr[M(x)=1]=0$。*

读者可以证明，7.4.1 节中给出的错率归约可以仅用对数空间开销来实现。因而，在

BPL 和 **RL** 的定义中，概率常数的选取并不重要。还可以注意到 **RL**⊆**NL**，进而 **RL**⊆**P**。习题 7.9 还要求证明 **BPL**⊆**P**。

一个著名的 **RL** 算法是求解 UPATH 的算法。UPATH 是 **NL**-完全问题 PATH（参见第 4 章）在无向图中的推广，亦即，给定 n 顶点无向图 G 和顶点 s，t，要求判定 s 和 t 在 G 中是否连通。

定理 7.19 ([AKL+79])　UPATH∈**RL**。

求解 UPATH 的算法实际上很简单：只需从 s 出发进行长度为 $\ell=100n^3$ 的随机游走。也就是说，将变量 v 初始化为顶点 s。此后，在每个步骤中，随机选取 v 的一个相邻顶点 u，并令 $v\leftarrow u$。算法接受输入当且仅当随机游走在 ℓ 步内到达顶点 t。由于算法运行过程中只需要保存一个计数器、当前顶点的索引和一些空白空间来计算随机游走的下一个相邻顶点，因此算法的空间开销是对数空间。显然，如果 s 和 t 不连通，则算法永远不会接受输入。可以证明，如果 s 和 t 是连通的，则在从 s 出发抵达 t 时所用随机游走的期望步数不超过 $10n^3$，因而算法接受输入的概率至少为 3/4。具体的分析过程留作习题 7.11。第 21 章将介绍一些用于分析图上随机游走的一般工具，用这些工具可以迅速得出上述期望步数的界限（甚至更好的界限）。21.4 节还将给出最近得到的一个求解 UPATH 的确定型对数空间算法。

更一般地，关于概率型对数空间计算和确定型对数空间计算之间的关系，人们还得到了一些非平凡的结果。已经证明，**BPL**（继而 **RL** 也）含于 **SPACE**($\log^{3/2}n$)。关于这一专题 139
的更多信息，参见 21.6 节以及第 21 章的章节注记。

本章学习内容

- 类 **BPP** 包含多项式时间概率型算法能够求解的所有语言，其中概率是相对于算法随机选择的操作而言的，而不是相对于输入的分布而言的。它比 **P** 更能刻画高效计算，尽管在这一点上人们看法还不尽相同。
- **RP**，**coRP** 和 **ZPP** 都是 **BPP** 的子类，前两个子类对应于单面错误概率型算法，而第三个子类对应于零面错误概率型算法。
- 通过重复调用，概率型算法的成功概率可以被显著地放大。
- 人们仅知道 **P**⊆**BPP**⊆**EXP**，但人们怀疑可能有 **BPP**=**P**。
- **BPP** 是 $\mathbf{P}_{/poly}$ 和 **PH** 的子集。特别地，后一个包含关系意味着：如果 **NP**=**P**，则 **BPP**=**P**。
- 除 **BPP** 之外，随机方法还可以用于复杂性理论的其他专题。两个典型的专题是随机归约和随机对数空间算法。后续章节将见到更多这样的专题。

本章注记和历史

研究者们最早意识到随机方法的效能是由于他们在从事计算（如设计原子弹）时用到了蒙特卡罗模拟等概率工具。概率型图灵机由德·莱乌（De Leeuw）等人定义[dLMSS56]。**BPP**（Bounded-error Probabilistic Polynomial-time），**RP**（Randomized Polynomial-time）和 **ZPP**（Zero-error Probabilistic Polynomial-time）等类的定义源自吉尔（Gill）[Gill77]。**BPP**

㊀ 关于 n 顶点图中从 s 出发通过随机游走访问所有与 s 连通的顶点，随机游走期望步数的最佳上界目前是 $\frac{4}{27}n^3+o(n^3)$[Fei95]。该上界是紧的，因为它可以在随机游走"棒棒糖图"时达到。所谓棒棒糖图，指的是将一条长为 $n/3$ 的路径连接到一个长为 $2n/3$ 的团上得到的图。

没有采用更简单的名字 **PP**(即 Probabilistic Polynomial-time)是因为吉尔在[Gil74]中已经将 **PP** 这个名字用于另一个更强大的类(参见第 17 章)。

证明 PRIMES 属于 **coRP** 时所采用的算法源自索洛维(Solovay)和施特拉森(Strassen)[SS77]。同时代提出的另一个素性测试算法则源自拉宾(Rabin)[Rab80]。此后多年,更好的素性测试算法不断被提出。直到最近终获突破,阿格拉沃尔(Agrawal)、卡亚尔(Kayal)和萨克塞纳(Saxena)三人最终证明了 PRIMES∈**P**[AKS04]。这些概率型素性测试算法和确定型素性测试算法都收录在舒普(Shoup)的书[Sho05]中。多项式恒等测试算法中使用的指纹技术源自卡普和拉宾[KR81]。洛瓦兹提出的用于判定完美匹配存在性的随机 **NC** 算法[Lov79]不尽令人满意,因为算法接受输入时不能给出任何线索来帮助人们找出完美匹配!后来的概率型 **NC** 算法克服了上述缺陷,参见[KUW85, MVV87]。对随机算法感兴趣的读者可以参考如下两本书。一本是密特森迈克尔(Mitzenmacher)和尤普夫(Upfal)的[MU05],另一本是莫特瓦尼(Motwani)和拉加万(Raghavan)的[MR95]。

$\mathbf{BPP}\subseteq\mathbf{P}_{/\mathrm{poly}}$(定理 7.14)源自阿德尔曼(Adelman)[Adl78]。$\mathbf{BPP}\subseteq\mathbf{PH}$ 属于西普赛尔[Sip83],更强的形式 $\mathbf{BPP}\subseteq\Sigma_2^p\cap\Pi_2^p$(定理 7.15)则源自 P·高奇(P. Gács)。我们给出的证明属于劳特曼(Lautemann)[Lau83]。最新成果表明,**BPP** 含于某些可能弱于 $\Sigma_2^p\cap\Pi_2^p$ 的复杂性类中[Can96, RS95]。

尽管证明 **BPP** 的分层定理似乎还力不能及,但是,在密切相关的类 **BPP**/1 上,证明分层定理已经取得了一些进展[Bar02, FS04, GST04]。其中 **BPP**/1 是以一个二进制位作为非一致建言的 **BPP** 类。我们注意到,如果将 **BPP** 推广到诺言问题(或者等价地说,定义于$\{0,1\}^*$的某一个子集上的所有布尔函数),则完全问题的存在性和分层定理的存在性等问题都不存在了。 140

7.6 节定义的 **BP·NP** 也可以推广到 **NP** 之外的任意复杂性类上,参见[Koz06]第 G 课。在这种推广之下,**BP·P**=**BPP**。

第 21 章将更深入地讨论随机游走,不但详细讨论了 7.4.1 节中提到的随机高效的错率归约,而且还分析了 7.7 节概述的对数空间连通算法。

习题

7.1 证明:投掷硬币可以高效地模拟从 1 到 N 中选取随机数。也就是证明,对于任意 N 和 $\delta>0$,存在运行时间为 $\mathrm{poly}(\log N\log(1/\delta))$ 且输出属于$\{1,\cdots,N,?\}$的概率型随机算法 A,使得:
(1) 只要输出不等于"?",则 A 的输出均匀分布于$[N]$中;并且
(2) A 输出"?"的概率不超过 δ。

7.2 证明:对于任意 $c>0$,无穷和 $\sum_{i=1}^{\infty}\frac{i^c}{2^i}$ 以依赖于 c 的某个常数为上界。亦即,证明:对于任意 $c>0$,存在常数 D 使得 $\sum_{i=1}^{n}\frac{i^c}{2^i}\leqslant D$ 对任意 $n\geqslant 1$ 成立。

7.3 给定正整数 a、n、p 的二进制表示,说明如何在多项式时间内计算 $a^n \pmod p$。

7.4 (**RP** 的错率归约)设 $L\subseteq\{0,1\}^*$ 是一个语言,且存在多项式时间的概率型图灵机 M 使得对任意 $x\in\{0,1\}^*$ 都有:(1)如果 $x\in L$,则 $\Pr[M(x)=1]\geqslant n^{-1}$;(2)如果 $x\notin L$,则 $\Pr[M(x)=1]=0$。
证明:对于任意 $d>0$,存在多项式时间的概率型图灵机 M' 使得对任意 $x\in\{0,1\}^*$ 都有:(1)如果 $x\in L$,则 $\Pr[M'(x)=1]\geqslant 1-2^{-n^d}$;且(2)如果 $x\notin L$,则

$\Pr[M(x)=1]=0$。

7.5　我们研究在何种程度上引理 7.12 真正需要假设 p 是高效可计算的。构造一个实数 p 使得，如果给定一枚"头面朝上"的概率等于 p 的硬币，则图灵机能够在多项式时间内判定一个不可判定语言。

7.6　(a) 证明：语言 L 属于 **ZPP** 当且仅当存在一个输出位于 $\{0,1,?\}$ 的多项式时间概率型图灵机 M 使得对任意 $x\in\{0,1\}^*$ 都有 $\Pr[M(x)\in\{L(x),?\}]=1$ 和 $\Pr[M(x)=?]\leqslant 1/2$。

(b) 证明定理 7.8，亦即证明：**ZPP**$=$**RP**$\cap$**coRP**。

7.7　非确定型线路 C 有两个输入 x，y。称 C 接受 x 当且仅当存在 y 使得 $C(x,y)=1$。线路的规模用 $|x|$ 的一个函数来度量。令 **NP**/poly 是由多项式规模的非确定型线路
141 能够判定的所有语言构成的集合。证明：**BP**$\cdot$**NP**$\subseteq$**NP**/poly。

7.8　证明：如果 $\overline{3SAT}\in$**BP**$\cdot$**NP**，则 **PH** 坍塌到 $\boldsymbol{\Sigma}_2^p$。(因此，$\overline{3SAT}\leqslant 3SAT$ 不太可能成立。)

7.9　证明：**BLP**$\subseteq$**P**。

7.10　证明：用随机游走求解连通性的思想不能用于有向图。换句话说，请你构造一个 n 顶点有向图和顶点 s，t，使得：从 s 到 t 存在有向路径，但是从 s 出发通过随机游走到达 t 需要的期望步数为 $\Omega(2^n)$。

7.11　(完成 UPATH$\in$**RL** 的证明)　令 G 是所有顶点具有相同度数的一个 n 顶点图。

(a) 对 G 的顶点上的一个分布 $\mathbf{p}$，其中 $\mathbf{p}_i$ 表示根据 $\mathbf{p}$ 随机选中第 i 个顶点的概率。如果根据 $\mathbf{p}$ 随机选中顶点 i，从 i 出发进行一步随机游走(到达 i 的一个相邻顶点 j 或停留在顶点 i)之后，所得的分布仍然是 $\mathbf{p}$，则称 $\mathbf{p}$ 是一个静态分布。证明：G 的顶点上的均匀分布是一个静态分布。

(b) 对 G 的顶点上的一个分布 $\mathbf{p}$，令 $\Delta(\mathbf{p})=\max_i\{\mathbf{p}_i-1/n\}$。对任意 k，$\mathbf{p}'$ 表示根据 $\mathbf{p}$ 随机选择一个顶点 i 再在 G 上进行 k 步随机游走之后得到的分布。证明：如果 G 是连通的非二分图，则存在 k 使得 $\Delta(\mathbf{p}')\leqslant(1-n^{-100})\Delta(\mathbf{p})$。由此得出结论：

1) 均匀分布是 G 的唯一静态分布。

2) 对 G 的任意两个顶点 u，v，如果从 u 出发进行足够长的随机游走，则访问顶点 v 的次数占随机游走总步数的期望比例约为 $1/n$。亦即，对于充分大的 N，从 u 出发的 N 步随机游走访问顶点 v 的期望次数至少为 $N/(2n)$。

(c) 对于 G 的顶点 u，把从 u 出发的随机游走回到顶点 u 所需步数的数学期望记为 E_u。证明：$E_u\leqslant 10n$。

(d) 对 G 的任意两个顶点 u，v，把从 u 出发的随机游走到达顶点 v 所需步数的数学期望记为 $E_{u,v}$。证明：如果 u 和 v 通过长度不超过 k 的路径连通，则 $E_{u,v}\leqslant 100kn^2$。由此得出结论：如果在图 G 中从 s 到 t 是连通的，则从 s 出发随机游走 $1000n^3$ 步仍不能达到 t 的概率不超过 $1/10$。

(e) 令 G 是一个 n 顶点图，它不一定是正则图，也就是说顶点的度可能不全相同。在 G 的每个顶点上添加一些自环可以将 G 变成一个正则图 G'。证明：如果从 s 出发在 G 上随机游走 k 步达到顶点 t 的概率至少为 0.9，则从 s 出发在 G' 上随
142 机游走 $10n^2k$ 步到达顶点 t 的概率至少为 $1/2$。

交互式证明

定理的证明过程到底需要满足哪些直观性质？首先，它必须能够"证明"正确的定理。其次，它不能"证明"错误的定理。第三，证明必须能够被高效地交流；也就是说，无论证明者通过多么复杂的计算才得到了证明过程，证明的验证者本质上仅需简单的计算就可以验证证明过程的真伪。

——戈德瓦瑟(Goldwasser)，米卡利(Micali)，拉科夫(Rackoff)，1985

数学证明这一标准概念与 **NP** 的证明式定义密切相关。人们将数学结论的正确性证明用一系列符号写在纸上，验证者查验证明者写出的证明是否有效。当然，只有正确的结论才存在有效证明。但是，更一般地，人们还通常通过交流来让大家相信结论的正确性。也就是说，查验证明的人(此后称验证者)要求提供证明的人(此后称证明者)先给出一系列解释，然后才相信结论是正确的。

自然地，人们希望从复杂性理论的角度研究这种交互式证明的能力。例如，可以简短地证明给定的公式是不可满足的吗？这个问题是 **coNP**-完全问题。因此，人们相信，该问题不存在传统意义上的具有多项式长度的证明。出人意料的是，如果允许验证者与证明者进行交互，则该问题将存在简短的证明(8.3 节)。同样，TQBF 以及 **PSPACE** 中的任意其他问题也存在简短证明。(注意，这种简短的证明要求验证者具备随机性，且随机性至关重要，参见 8.1 节。)上述事实本身就使得交互式证明的研究具有重要意义。此外，研究交互式证明还有助于对一系列问题的理解。这些问题包括密码协议(评注 8.8 和 9.4 节)、近似算法能力的局限性(8.5 节)、程序检验(8.6 节)、图同构等著名问题不属于 **P** 的证据(8.1.3 节)以及最短格向量的近似问题不是 **NP**-完全问题，等等(参见本章注记)。

8.1 交互式证明及其变形

如前所述，交互式证明将交互过程引入了基本复杂性类 **NP**。在 **NP** 的定义中，证明者不是将写好的证明交给验证者，而是验证者向证明者不断地询问一系列问题并听取证明者对这些问题的解答。最后，验证者决定是否接受输入。当然，在交互过程的每个时刻上，验证者提出问题和证明者给出解答都可以依赖于他们之前交流过的信息。证明者被假定为一台全能型机器(参见定义 8.3 之后的注记)，但是，后面将看到，假设证明者是一台 **PSPACE** 机器即可；参见定义 8.6 后面的评注。

在完整定义交互式证明的过程中，我们还可以对几个其他的因素进行选择。其一，证明者是确定型的还是概率型的；其二，验证者是确定型的还是概率型的；其三，如果选用概率型机器，如何定义"接受"和"拒绝"。有关于此，我们在第 7 章曾看到几种选择，这些选择均依赖于允许机器所犯错误是单面的还是双面的。

我们将研究这些因素的某些选择对定义的影响。

8.1.1 准备工作：验证者和证明者均为确定型的交互式证明

首先，考虑验证者和证明者均为确定型的交互式证明。

例 8.1 我们考虑一个平凡的交互式证明，它用于判定 3SAT 的成员资格。验证者逐一地考察公式中的每个子句，向证明者询问每个子句中的文字的具体取值。验证者跟踪记录证明者对每个子句给出的答案。最后，如果所有子句确实被满足且证明者没有为同一个变量给出不同的取值，则验证者接受公式。

因此，验证者和证明者均是确定型的。

当然，在此种情况下，我们可能会问：在上述交互过程中，证明者能否仅在第一个回合中给出所有布尔变量的取值，然后在后续回合中进入休眠呢？下面马上就会看到，这实际上将作为一种特殊情况包含于如下更具一般性的现象中：在验证者为确定型时，交互式证明仅需一个回合而不必持续多个回合。◀

首先，让我们澄清例 8.1 中“交互”一词的含义。它的意思是指，验证者和证明者是两个确定型函数，在交互式计算的每个回合中，这两个函数给出的问题/答案既依赖于输入也依赖于之前各个步骤的问题和答案。

定义 8.2（确定型函数的交互） 设 $f, g: \{0,1\}^* \to \{0,1\}^*$ 是两个函数，k 是一个整数（其取值可以依赖于输入规模）。f 和 g 在输入 $x \in \{0,1\}^*$ 上的 k 回合交互记为 $\langle f, g\rangle(x)$，指的是如下定义的位串序列 $a_1, a_2, \cdots, a_k \in \{0,1\}^*$：

144

$$\begin{aligned} a_1 &= f(x) \\ a_2 &= g(x, a_1) \\ &\cdots \\ a_{2i+1} &= f(x, a_1, \cdots, a_{2i}) \qquad 2i \leqslant k \\ a_{2i+2} &= g(x, a_1, \cdots, a_{2i+1}) \qquad 2i+1 \leqslant k \end{aligned} \tag{8.1}$$

交互结束时 f 的输出定义为 $f(x, a_1, \cdots, a_k)$，记为 $\mathrm{out}_f\langle f, g\rangle(x)$；我们假设 $\mathrm{out}_f\langle f, g\rangle(x)$ 属于 $\{0,1\}$[⊖]。

定义 8.3（确定型证明系统） 称语言 L 存在确定型交互证明系统，如果存在确定型图灵机 V 在输入 $x, a_1, a_2, \cdots, a_k$ 上的运行时间是 $|x|$ 的多项式，且 V 可以与任意函数 P 交互 k 个回合，使得：

（完备性） $x \in L \Rightarrow \exists P: \{0,1\}^* \to \{0,1\}^*$ 满足 $\mathrm{out}_V\langle V, P\rangle(x) = 1$

（可靠性） $x \notin L \Rightarrow \forall P: \{0,1\}^* \to \{0,1\}^*$ 满足 $\mathrm{out}_V\langle V, P\rangle(x) = 0$

类 **dIP** 包含所有存在 $k(n)$ 回合确定型交互证明系统的语言，其中 $k(n)$ 是 n 的多项式。

注意，上述定义未限制证明者 P 的计算能力，这恰好实现了交互式系统的直观意义，因为错误的断言不应该被证明，无论证明者多么聪明。此外还注意到，由于没有限制证明者的计算能力，因此，在完备性和可靠性条件中，证明者是否依赖于输入 x 无关紧要（参见习题 8.2）。

正如例 8.1 暗示的那样，**dIP** 实际上是我们熟悉的类。

引理 8.4 **dIP**=**NP**。

证明 直接可以看到，任意 **NP** 语言均存在 1-回合确定型证明系统，因此任意 **NP** 语言属于 **dIP**。往证，如果 $L \in$ **dIP** 则 $L \in$ **NP**。如果 V 是 L 的验证者，则任意 $x \in L$ 的证明就是使得 V 接受 x 的交互笔录 $(a_1, a_2, \cdots, a_k)$。为了验证该笔录，只需查验 $V(x)=a_1$，$V(x, a_1, a_2)=a_3$，$\cdots$，$V(x, a_1, a_2, \cdots, a_k)=1$ 确实成立。如果 $x \in L$，则上述笔录存在。反过来，如果上述笔录存在，则可以定义一个证明者函数 P 使得 $P(x, a_1)=a_2$，$P(x, a_1, a_2, a_3)=a_4$，等等。

⊖ $a_1, a_2, \cdots, a_k$ 称为交互过程的交互笔录，原文未给出该概念，但后文将多次用到。——译者注

该确定型证明者函数满足 $\text{out}_V\langle V, P\rangle(x)=1$，这表明 $x\in L$。 ■

8.1.2　类 IP：概率型验证者

8.1.1 节表明，为使交互过程发挥更大作用，我们需要让验证者是概率型的。这意味着，验证者向证明者咨询的每个问题都是用概率型算法计算得到的，而且验证者甚至可以 145
小概率地作出错误结论(比如，错误结论的某个证明可能被验证者接受)。同概率型算法一样，上述概率是相对于验证者的所有随机选择而言的，而且我们还要求验证者在证明者的任意应对策略下都能够高概率地拒绝错误结论的证明。交互过程和随机性的这种组合对交互式证明系统具有重大的影响。事实上，正如 8.3 节将要看到的那样，这种交互式证明系统判定的语言构成的集合将由 **NP** 跃变成 **PSPACE**。

例 8.5　下面给出一个直观例子，展示交互过程与随机性组合在一起之后的能力。考虑如下情形。玛丽有一只红袜子和一只黄袜子，她的一个色盲朋友亚瑟不相信她的袜子颜色不同，玛丽如何才能让亚瑟相信这是真的呢?

一个简单的办法如下。玛丽把两只袜子都交给亚瑟，并告诉他哪只是红的哪只是黄的；亚瑟将红袜子放在右手，将黄袜子放在左手；然后，玛丽转过身背对着亚瑟。亚瑟投掷一枚硬币：如果头面朝上则保持袜子在手中的位置不动；否则，他交换左、右手上的袜子；然后，他让玛丽猜他是否交换过手中的袜子。当然，玛丽通过观察亚瑟左手上是否拿着红袜子，可以很容易给出答案。但是，如果两只袜子是同色的，则玛丽猜中答案的概率不会大于 1/2。因此，如果玛丽在(不妨说)100 次重复的实验中都给出了正确答案，则亚瑟的确可以相信两只袜子颜色不同。

上述"交互式证明系统"背后的原理也是我们后面在 8.1.3 节为图不同构和例 8.9 中为二次非剩余构造交互式证明系统的基础。在袜子这个例子中，验证者是色盲，其能力弱于(或者说技能少于)证明者。在一般的交互式证明中，验证者有多项式运行时间，其计算能力也弱于证明者。 ◀

下面，我们精确定义具有概率型验证者的交互式证明。为了扩展定义 8.2 来建模 f 是概率型函数时 f，g 之间的交互，我们在(8.1)式的函数 f 中增加长为 m 位的一个输入 r，亦即 $a_1=f(x, r)$，$a_3=f(x, r, a_1, a_2)$，等等。但是，函数 g 的计算仍只依赖于这些 a_i，而不将 r 作为输入。这种模型实际上就表示了如下事实：证明者无法"看见"验证者所做的随机选择，而只能"看见"验证者给出的信息。正是由于这个原因，上述模型也称为交互式证明的私有随机源模型，与之相对的是 8.2 节中的公用随机源模型。因此，在私有随机源模型下，交互$\langle f, g\rangle(x)$是 $r\in_u\{0, 1\}^*$ 上的一个随机变量。类似地，输出 $\text{out}_f\langle f, g\rangle(x)$也是一个随机变量。

定义 8.6　(概率型验证者和类 **IP**)　对于整数 $k\geqslant 1$(k 可以依赖于输入的长度)，我们称语言 L 属于 **IP**$[k]$，如果存在一个概率型多项式时间图灵机 V，它可以与任意函数 $P: \{0, 1\}^*\to\{0, 1\}^*$ 进行 k 个回合的交互，使得 146

$$\text{(完备性)}\quad x\in L\Rightarrow\exists P \text{ 满足 } \Pr[\text{out}_V\langle V, P\rangle(x)=1]\geqslant 2/3 \tag{8.2}$$

$$\text{(可靠性)}\quad x\notin L\Rightarrow\forall P \text{ 满足 } \Pr[\text{out}_V\langle V, P\rangle(x)=1]\leqslant 1/3 \tag{8.3}$$

其中概率是相对于 V 的任意随机选择 r 而言的。

我们定义 $\mathbf{IP}=\bigcup_{c>1}\mathbf{IP}(n^c)$。

下面，我们研究定义的健壮性。首先，我们证明，定义 8.6 中的概率 2/3 和 1/3 可以

分别任意地接近于 1 和 0，这只需借助 7.4.1 节处理 **BPP** 时用过的概率放大技术。

引理 8.7 对于任意固定常数 s，如果将定义 8.6 中的完备性参数 2/3 替换为 $1-2^{-s}$ 并将可靠性参数 1/3 替换为 2^{-s}，则所定义的类 **IP** 将保持不变。

证明 验证者一遍又一遍地重复执行整个协议，不妨设执行 m 遍；最后，验证者接受输入当且仅当验证者在超过半数的协议执行过程中接受输入。如果 $x\in L$，则证明者每次让验证者接受输入的概率为 2/3，由切尔诺夫界(定理 A.14)可知，验证者最后接受的概率为 $1-2^{-\Omega(m)}$。如果 $x\notin L$，我们必须证明任意证明者的策略都将高概率地导致验证者拒绝输入。我们断言，在协议的每一遍执行过程中，无论之前的各遍执行过程中发生什么情况，证明者仅以 1/3 的概率使得验证者接受输入。原因在于，尽管证明者在本次协议执行时给出的答案可能以任意方式依赖于前面各次协议执行时给出的答案，但是由于(8.3)式对任意证明者均成立，故(8.3)式对于了解前面各次答案的证明者也成立。

因此，仍由切尔诺夫界可知，证明者在超过半数的协议执行过程中接受输入的概率仅为 $2^{-\Omega(m)}$。取定 $m=O(n)$ 即可证得引理。 ■

现在，我们给出 **IP** 类上的几个论断，习题 8.1 要求读者证明其中的部分论断。

1. 如果证明者是概率型的，也就是说允许每次给出的答案 a_i 依赖于证明者使用(验证者不知道)的一个随机串，则类 **IP** 不会改变。这是由于，对于任意语言 L，如果概率型证明者 P 能够让验证者 V 以某个概率接受，则对证明者的所有随机选择取平均将得到一个确定型证明者，他使得验证者以同样的概率接受。

2. 由于证明者可以使用任意的函数，因此，原则上讲证明者可以使用计算能力不受限的函数，甚至可以使用不可判定函数。尽管如此，可以证明：给定任意验证者，我们能够找到仅用 $\mathrm{poly}(|x|)$ 空间(继而也仅用 $O(2^{\mathrm{poly}(|x|)})$ 时间)的最佳证明者，它在给定的 x 上使验证者的接受概率达到最大值。因此 **IP**$\subseteq$**PSPACE**。

3. 将(8.2)式完备性条件中的 2/3 替换为 1 不会改变类 **IP**。这一事实是非平凡的，其最初的证明曾采用了复杂的方法，如今采用 8.3 节给出的 **IP** 的性质就可以证明。

4. 相比之下，如果将(8.3)式可靠性条件中的 1/3 替换为 0，则等价于采用确定型验
147 证者，因此这种修改将 **IP** 变成了 **NP**。

5. *私有随机源*：目前，证明者函数仅依赖于验证者发送的消息/问题，但不依赖于验证者的随机选择位串。换句话说，验证者的随机选择位串是私有的。通常，这种交互式证明称为私有随机源上的交互式证明。8.2 节将考虑公用随机源上的交互式证明(也就是著名的亚瑟-梅林证明)，其中验证者的所有咨询问题都通过投掷硬币产生，并且咨询的问题和相应的硬币投掷结果将一起发送给证明者。

6. 引理 8.7 的证明将基本协议重复执行 m 次并采纳所有执行过程得到的主流答案。事实上，采用稍复杂一些的过程还可以证明：采用并行重复(即所有 m 个咨询问题一次性全部提出)，则无需增加交互回合数 m(证明者和验证者并行执行 m 次协议)也能使概率减小到同样的值。在公用随机源模型下，上述事实的证明要容易一些。

8.1.3　图不同构的交互式证明

我们展示 **IP** 中的另一个语言，该语言目前还不知道是否属于 **NP**。在将图表示为邻接表或邻接矩阵等形式时，通常需要将图的每个顶点标记为一个唯一的整数。如果图 G_1 和 G_2 可以通过对顶点进行恰当标记来将它们转换成同一个图，则称 G_1 和 G_2 同构。换句话说，

如果 G_1 和 G_2 同构，则在 G_1 的所有顶点标签上存在一个置换 π 使得 $\pi(G_1)=G_2$，其中 $\pi(G_1)$ 是将 π 作用到 G_1 的各个顶点上之后得到的图。例如，图 8-1 中的两个图在置换 $\pi=(12)(3654)$ 下是同构的，其中 π 将 1 和 2 相互映射到对方，3 映射到 6，6 映射到 5，5 映射到 4，4 映射到 3。如果图 G_1 和 G_2 同构，则记为 $G_1\cong G_2$。GI 问题定义为：给定两个图 G_1 和 G_2，判定 G_1，G_2 是否同构。

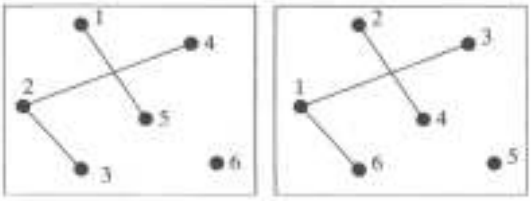


图 8-1　两个同构的图

显然，GI∈**NP**，因为两个图同构的一个证明可以是置换 π 的明确描述。图同构问题在各种领域中非常重要，并且对它的研究由来已久（参见[Hof82]）。"GI 是否是 **NP**-完全问题"仍是一个未解决的问题，它与因数分解问题是两个最著名的 **NP** 问题，人们既未能证明它们属于 **P** 也未能证明它们是 **NP**-完全的。8.2.4 节将证明，除非多项式分层坍塌，否则 GI 将不是 **NP**-完全问题。证明过程的第一个步骤是 GI 的补集 GNI 上的一个交互式证明；GNI 中的一个问题是判定给定的两个图是否不同构。

协议：私有随机源图不同构协议

V：均匀随机地选取 $i\in\{1,2\}$，将 G_i 的所有顶点进行随机置换，得到新图 H。将 H 发送给 P。

P：确定 H 是由 G_1，G_2 中哪一个图产生的。如果 G_j 产生了 H，则将 j 发送给 V。

V：如果 $i=j$，则接受输入；否则，拒绝输入。

148

为了看清上述协议符合定义 8.6，注意到：如果 $G_1\ncong G_2$，则存在证明者使得 Pr[V 接受]=1，这是因为，如果两个图是不同构的，则全能型证明者将确切地指明哪个图同构于 H。另一方面，如果 $G_1\cong G_2$，则最厉害的证明者也只能靠猜，因为 G_i 的随机置换看上去与 G_i 的随机置换别无二致；因此，此时对任意证明者而言均有 Pr[V 接受]$\leqslant 1/2$。这一概率可以通过顺序重复或并行重复减小为 1/3。

评注 8.8（零知识证明）　现在，我们简要地讨论一下零知识证明，它是密码学中大量研究工作的基础。粗略地讲，零知识证明系统也是一个交互式证明协议，它用于证明某个语言的成员资格，而且在该证明系统中验证者除了能够最终相信输入 x 属于该语言之外不能了解到任何其他信息。我们如何才能将"验证者不能了解到任何其他信息"这一条件量化呢？我们的做法是证明验证者无需证明者的帮助就可以在多项式时间内给出协议的交互笔录。在 9.4 节将会看到，上述的图不同构协议是零知识的。

不难理解零知识证明的概念为什么会用于密码学，因为正是零知识证明使得"参与密码的各方在不泄露本方秘密的条件下向他方提供证据"成为可能。例如，你可以在无需泄露密码本身的前提下证明你确实持有密钥。这也正是发明交互式证明系统的原始动机之一。9.4 节将给出零知识证明的形式定义，并给出几个实例。该小节将不依赖于第 9 章的其他部分，因此现在就可以独立地进行阅读。

例 8.9（二次非剩余）　我们再给出一个交互式证明，它所判定的语言也不知道是否属于 **NP**。如果存在整数 b 使得 $a\equiv b^2 \pmod p$，则称整数 a 是模 p 的二次剩余，并称 b 是 a 模 p 的平方根。显然，$-b$ 是 a 模 p 的另一个平方根。而且，a 模 p 不存在其他平方根，因为 $x^2-a=0$ 在域 GF(p) 上至多有两个解。

任意素数 p 和模 p 的二次剩余 a 形成的所有序对 $\langle a, p\rangle$ 构成的语言属于 **NP**，因为 a 模 p 的平方根就是 $\langle a, p\rangle$ 属于该语言的证明。当然，p 的素性也存在短证明。事实上，素性测试存在多项式时间算法，参见第 2 章。

任意素数 p 和模 p 的二次非剩余 a 形成的所有序对 $\langle a, p\rangle$ 构成一个语言，我们将该语言记为 QNR。与二次剩余语言相反，QNR 的成员资格不存在自然的短证明，目前也不知道 QNR 是否属于 **NP**。尽管如此，如果允许验证者是概率型的，则 QNR 确实存在一个简单的交互式证明。

验证者取定模 p 的随机数 r 和一个随机二进制位 $b\in\{0, 1\}$，并且将随机位 b 对证明者保密。如果 $b=0$，则验证者将 $r^2 \bmod p$ 发送给证明者；如果 $b=1$，验证者将 $ar^2 \bmod p$ 发送给证明者。然后，验证者要求证明者猜 b 的值，她接受输入当且仅当证明者猜中 b 的值。

如果 a 是二次剩余，则 ar^2 和 r^2 均是取自模 p 的二次剩余群中的元素并且服从相同的概率分布。（为看清这一点，注意任意二次剩余 a' 均可以写成 as^2 的形式，其中 s 是 a/a' 的
149 平方根。）因此，证明者猜中 b 值的概率至多为 1/2。

另一方面，如果 a 是二次非剩余，则 ar^2 和 r^2 服从完全不同的概率分布：前者是模 p 的一个随机二次非剩余，而后者是模 p 的一个随机二次剩余。全能型证明者能够区分二次非剩余和二次剩余，因此他猜中 b 值的概率为 1。因而，证明者能够让验证者以概率 1 接受输入。◀

8.2　公用随机源和类 AM

图不同构和二次非剩余的交互式证明系统的关键在于，验证者使用了其私有的随机源，而证明者无法使用验证者的私有随机源。如果允许证明者使用验证者的所有随机串，则得到下面基于公用随机源的交互式证明。

定义 8.10（**AM，MA**）　在 **IP**$[k]$ 的定义中（参见定义 8.6），如果限定验证者发送的消息是一系列随机二进制位，并且除了发送给证明者的消息之外，验证者不使用其他随机位，则得到 **IP**$[k]$ 的一个子集，将该子集定义为复杂性类 **AM**$[k]$。

如果验证者具有上面的形式，则将相应的交互式证明称为公用随机源证明，也就是著名的亚瑟-梅林证明[⊖]。

我们将类 **AM**$[2]$ 记为 **AM**[⊜]。也就是说，**AM** 是存在如下交互式证明的所有语言构成的类：在这种交互式证明中，验证者发送一个随机位串给证明者，证明者再发送一个响应消息给验证者，然后验证者将一个确定型多项式时间函数作用到交互笔录上来决定是否接受输入。类 **MA** 表示由证明者首先发送消息的两回合公用随机源交互式证明能够判定的所有语言构成的类。也就是说，如果 $L\in$ **MA**，则语言 L 存在一个交互式证明系统，该系统首先由证明者发送消息，然后验证者投掷硬币并在输入、证明者的消息和投掷硬币的结果上通过确定型多项式时间计算来做出拒绝或接受的决定。

评注 8.11　我们给出类 **AM**$[k]$ 的一些性质：

⊖ 据一个古老的传说讲，亚瑟是中世纪时期英格兰的一位伟大的国王，而梅林是他的一位宫廷魔法师。Babai [Bab85]用"亚瑟-梅林"来命名交互式证明系统恰恰是为了表达证明者的无限能力和梅林的魔法无所不能之间的相似性。虽然梅林无法预测亚瑟今后投掷硬币的结果是什么，但是亚瑟投掷硬币已经得出的结果却无法瞒过梅林的魔法。

⊜ 注意，**AM**=**AM**$[2]$ 而 **IP**=**IP**[poly]。尽管这确实有些不一致，但它是文献中采用的标准记号。有些文献还将类 **AM**$[3]$ 记为 **AMA**，将 **AM**$[4]$ 记为 **AMAM**，等等。

1. 注意，即使在公用随机源证明中，证明者也不会立刻知晓验证者投掷硬币得到的所有随机结果；相反，这些结果是通过一条一条的消息逐步发送给证明者的。也就是说，一个 **AM**[k]-证明是一个 **IP**[k]-证明，其中验证者的随机存储带上存储$\lceil k/2 \rceil$个随机串 $r_1, \cdots, r_{\lceil k/2 \rceil}$，他的第 i 个消息恰好为 r_i。验证者需要将多项式时间可计算的函数作用到交互笔录上，才能决定是接受输入还是拒绝输入。 150

2. **AM**[2]=**BP**·**NP**，其中 **BP**·**NP** 见定义 7.17。特别地，由此可得 **AM**[2]$\subseteq\Sigma_2^p$（参见习题 8.3）。

3. 对于常数 $k\geqslant 2$，我们有 **AM**[k]=**AM**[2]（习题 8.7）。这种"坍塌"有些出人意料。这是因为，初看起来 **AM**[k]类似于 **PH**，只是全称量词 $\forall$ 被修改成了"概率型全称量词 $\forall$"，该量词的含义是绝大多数的计算分支将导致验证者接受输入，参见图 8-2。

4. 一个未解决的问题是问 **AM**[$\sigma(n)$]是否存在良好的性质，其中 $\sigma(n)$是 n 的适度缓慢增长的函数，如 $\log\log n$。

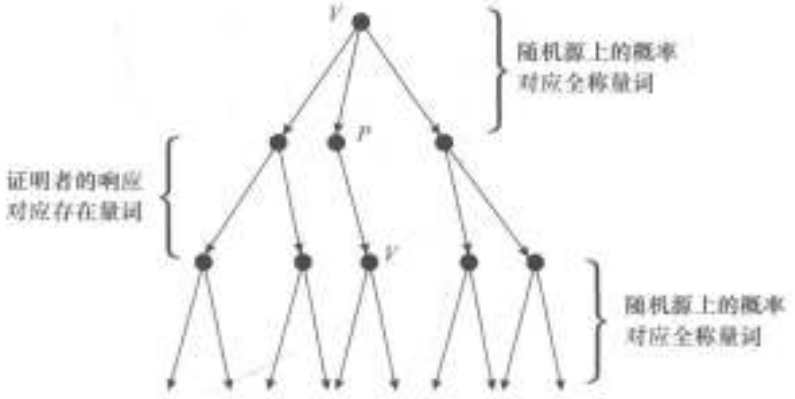


图 8-2　**AM**[k]看上去像 Π_2^p，只是将其中的全称量词替换为概率型选择

8.2.1　私有随机源的模拟

显然，对任意 k 均有 **AM**[k]$\subseteq$**IP**[k]。GNI 的交互式证明系统的关键似乎在于，该证明依赖于"P 不能知晓 V 的所有随机位"这一事实。如果 P 知晓这些随机位，则 P 也将知晓 i，进而可以直接准确地猜到答案。因此，允许验证者拥有私有随机源似乎极大地提升了交互式证明的能力。故下面的结论颇为出人意料。

定理 8.12（戈德瓦塞-西普塞尔[GS87]）　对任意函数 k：**N**→**N**，如果 $k(n)$可以在 poly(n)时间内被计算，则 **IP**[k]$\subseteq$**AM**[$k+2$]。

定理 8.12 的证明概要放到定理 8.13 的证明之后，该定理考虑 GNI 的子情况。

定理 8.13　GNI$\in$**AM**[2]。

定理 8.13 的证明是一个很好的例子，它展示了如何通过重塑问题来得出非平凡的交互式证明。证明过程的关键思想是用不同的方法来审视图不同构：确切地说，是用更加量化的方法来审视图不同构。考虑下面的标记图集 $S=\{H: H\cong G_1 \text{ 或 } H\cong G_2\}$。注意，要验证 H 是 S 的成员，只需给出一个置换将 G_1 或 G_2 映射为 H。任意 n 顶点图至多有 $n!$ 个同构图。为简洁计，我们假设 G_1 和 G_2 都恰有 $n!$ 个同构图。因此，G_1 和 G_2 同构和不同构将导致 $|S|$ 相差因子 2，亦即 151

$$\text{如果 } G_1\ncong G_2, \text{则 } |S|=2n! \tag{8.4}$$

$$\text{如果 } G_1 \cong G_2, \text{则 } |S| = n! \tag{8.5}$$

在一般情况下，G_1或G_2的同构图可能少于$n!$个。n顶点图G的同构图少于$n!$个当且仅当该图存在非平凡的自同构π(亦即，π不是恒等置换但满足$\pi(G)=G$)。令$\mathrm{aut}(G)$表示图G的所有自同构构成的集合。我们将S重新定义为

$$S = \{(H,\pi): H \cong G_1 \text{ 或 } H \cong G_2, \text{且 } \pi \in \mathrm{aut}(H)\}$$

利用$\mathrm{aut}(H)$是子群这一事实，可以验证S满足(8.4)式和(8.5)式。同样，该集合的成员资格也很容易验证。

因此，为了使验证者相信$G_1 \ncong G_2$，证明者需要让验证者相信(8.4)式成立，而(8.5)式不成立。这可以用集合下界协议来完成。

8.2.2　集合下界协议

假设存在一个证明者和验证者都知道的集合S，S的成员资格在下述意义下容易验证：如果某个串x碰巧属于S，则证明者能够(利用其超级计算能力)向验证者提供$x \in S$的证明。更正式地，假设S属于**BP · NP**。集合下界协议是一个公用随机源协议，证明者能够证实S的近似大小。注意，证明者能够(利用其超级计算能力)确切地计算$|S|$并将它通报给验证者。关键是如何让验证者相信所通报的答案是正确的或近似正确的。假设证明者宣称$|S|$等于K，下面将定义一个满足如下性质的协议：如果$|S|$的真实值实际上大于等于K，则证明者将让验证者高概率地接受；如果$|S|$的真实值实际上至多为$K/2$(证明者宣称的答案位于高半区)，则证明者将让验证者高概率地拒绝。该协议称为集合下界协议。显然，集合下界协议可以用来证明定理8.13。

工具：两两独立的哈希函数

集合下界协议要用到的主要工具是两两独立的哈希函数族，它在复杂性理论和计算机科学中有很多其他应用(参见注记8.16)。

定义8.14　(两两独立的哈希函数)　设$\mathcal{H}_{n,k}$是将$\{0,1\}^n$映射到$\{0,1\}^k$的一个函数族。如果对于任意满足$x \neq x'$的$x, x' \in \{0,1\}^n$和任意$y, y' \in \{0,1\}^k$均有$\Pr_{h \in_R \mathcal{H}_{n,k}}[h(x)=y \wedge h(x')=y'] = 2^{-2k}$，则称$\mathcal{H}_{n,k}$是两两独立的。

152

等价地，两两独立的哈希函数族还可以如下定义。对于固定的任意两个不同的串$x, x' \in \{0,1\}^n$，如果从$\mathcal{H}_{n,k}$中随机选取h，则随机变量$(h(x), h(x'))$均匀分布于$\{0,1\}^k \times \{0,1\}^k$。

我们可以将$\{0,1\}^n$的元素等同于有限域$\mathrm{GF}(2^n)$的元素，$\mathrm{GF}(2^n)$的定义参见A.4节。回顾一下，该有限域的加法(+)和乘法(·)可以高效计算，并且满足普通意义下的交换律和分配律；任意元素x均存在加法逆元素(记为$-x$)；如果x不是0元素，则它还存在乘法逆元素(记为x^{-1})。下面的定理给出了构造高效可计算的两两独立的哈希函数族的方法，习题8.4给出了另一种不同的构造方法。

定理8.15　(高效的两两独立哈希函数)　对任意n，定义函数族$\mathcal{H}_{n,n}$为$(h_{a,b})_{a,b \in \mathrm{GF}(2^n)}$，如果对于任意$a, b \in \mathrm{GF}(2^n)$，函数$h_{a,b}: \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$将$x$映射到$ax+b$，则$\mathcal{H}_{n,n}$是两两独立的哈希函数族。

定理8.15表明，对于任意n，k，均存在高效可计算的两两独立的哈希函数族$\mathcal{H}_{n,k}$。事实上，如果$k>n$，则这个函数族可以是$\mathcal{H}_{k,k}$，将长度为n的输入用0填充为长度为k的输入即得到$\mathcal{H}_{n,k}$。如果$k<n$，则这个函数族可以是$\mathcal{H}_{n,n}$，将长度为n的输入去掉最末$n-k$位缩减为长度是k的输入即得到$\mathcal{H}_{n,k}$。

证明　对于任意 $x\neq x'\in \mathrm{GF}(2^n)$ 和 $y,\ y'\in \mathrm{GF}(2^n)$，$h_{a,b}(x)=y$ 且 $h_{a,b}(x')=y'$ 当且仅当 $a,\ b$ 满足方程组

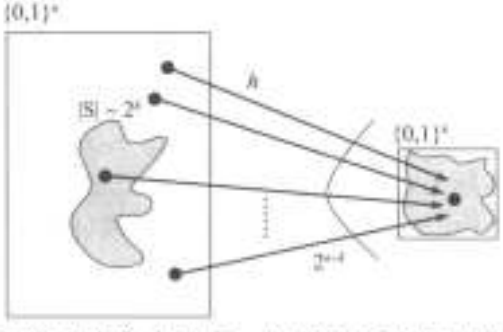
$$a\cdot x+b=y$$
$$a\cdot x'+b=y'$$

该方程组意味着 $a=(y-y')(x-x')^{-1}$；由于 $x-x'\neq 0$，因此 a 是良定义的。又由于 $b=y-a\cdot x$，故序对 $(a,\ b)$ 由上述方程组唯一确定。因此，在所有可能的组合中随机选择 $a,\ b$ 时，取中满足方程的序对 $(a,\ b)$ 的概率恰为 $\frac{1}{2^{2n}}$。 ■

注记 8.16（哈希函数示例[CW77]）　在很多计算机程序中，哈希函数被用于创建哈希表。哈希表的目的是存储集合 $S\subseteq\{0,\ 1\}^n$，以使高效地响应成员资格查询；亦即，快速地判定 x 是否属于 S。其中，集合 S 可以动态地插入或删除元素；尽管如此，S 的大小小于所有可能的元素个数 2^n。

要创建规模为 2^k 的哈希表，需要先选择一个从 $\{0,\ 1\}^n$ 到 $\{0,\ 1\}^k$ 的哈希函数 h，然后将 $x\in S$ 存储在 $h(x)$ 表示的存储位置上。在需要判定 x 是否属于 S 的时候，只需计算 $h(x)$ 的值并到哈希表中 $h(x)$ 表示的存储位置上去查找 x。注意，如果 $h(x)=h(y)$，则 $x,\ y$ 将被存储在相同的位置上，这种情况称为一个冲突。我们希望尽量避免这样的冲突进而使得成员资格查询被高效处理，因此需要通过选择足够"随机"的哈希函数来使得冲突的个数最少。

为此，人们通常不是使用一个固定的哈希函数，而是从一个哈希函数族中随机选用一个哈希函数，其中哈希函数族可以类似于定理 8.15 中给出的函数族。这样，$\{0,\ 1\}^n$ 中的大多数元素都能在 S 中找到大约 $|S|2^{-k}$ 个原像；事实上，如果哈希函数 h 是完全随机的，则 $|S|2^{-k}$ 是 $\{0,\ 1\}^k$ 中元素在 S 中的原像个数的数学期望。特别地，如果 S 的大小约为 2^k，则随机哈希函数给出的映射差不多是一个一一映射，因此冲突的期望个数很少。也就是说，在哈希函数 h 下，S 的像如下图所示。



在待处理集合是动态变化的许多应用中，人们往往偏向于使用哈希表。只要 S 的大小小于 2^k 且哈希函数又是通过独立于集合 S 的随机二进制位来随机选取的，则上一自然段所分析的期望冲突个数就是有效的。

在理论计算机科学中，哈希函数也有大量应用。例如，引理 17.19 将证明，如果哈希函数族是两两独立的且 $S\subseteq\{0,\ 1\}^n$ 的大小约为 2^k，则 0^n 在 S 中恰有一个原像的概率很大。再如，残余哈希引理（引理 21.26）将证明，如果 S 的大小大于 2^k，则 S 中的一个随机元素将几乎被完美地映射为 $\{0,\ 1\}^k$ 中的一个随机元素。

两两独立的哈希函数族仅仅是一类哈希函数族。通过不同性质来刻画哈希函数的高效可计算性和输出值的均匀性之间的关系，人们还研究了很多其他的哈希函数族，包括几乎两两独立的哈希函数族，k-独立哈希函数族，ε-偏斜的哈希函数族，等等。感兴趣的读者

可以参阅卢比(Luby)和维格德尔森(Wigderson)的综述[LW06]。

集合下界协议

集合下界协议如下所示。

> **协议：戈德瓦瑟-西普赛尔集合下界协议**
>
> 条件：$S\subseteq\{0,1\}^n$是一个集合，其成员资格可以被验证。协议双方均知晓整数K。证明者旨在让验证者相信$|S|\geqslant K$，而且，如果$|S|\leqslant\frac{K}{2}$，则验证者将高概率地拒绝。令k是满足$2^{k-2}<K\leqslant 2^{k-1}$的一个整数。
>
> V：从两两独立的哈希函数族$\mathcal{H}_{n,k}$中随机选取一个函数$h:\{0,1\}^n\to\{0,1\}^k$。取$y\in_R\{0,1\}^k$，将h和y发送给证明者。
>
> P：尝试找出$x\in S$使得$h(x)=y$。将这个x发送给验证者V，同时将$x\in S$的证明也发送给验证者。
>
> V的输出：如果$h(x)=y$且收到的证明确实表明$x\in S$，则接受；否则，拒绝。

153 / 154

显然，(全能型)证明者能够让验证者接受输入当且仅当恰好存在$x\in S$使得h，y满足$h(x)=y$。下面的论断表明，对于$p'=K/2^k$，我们感兴趣的两种情况($|S|\geqslant K$和$|S|<K/2$)发生的概率分别是$\frac{3}{4}p'$和$p'/2$，这两个概率之间存在明显的鸿沟。论断仅讨论了$|S|\leqslant K$的情况；尽管如此，请注意，显然集合越大，证明者让验证者接受的概率就越高。

论断 8.16.1　设$S\subseteq\{0,1\}^n$满足$|S|\leqslant\frac{2^k}{2}$，则对于$p=|S|/2^k$有

$$p\geqslant\Pr_{h\in_R\mathcal{H}_{n,k},y\in_R\{0,1\}^k}[\exists_{x\in S}:h(x)=y]\geqslant\frac{3p}{4}-\frac{p}{2^k}$$

证明　要得到概率的上界，只需注意到：原像位于S中的所有y构成的集合$h(S)$其大小至多为$|S|$。下面证明下界。事实上，我们将证明更强的结论：

$$\Pr_{h\in_R\mathcal{H}_{n,k}}[\exists_{x\in S}h(x)=y]\geqslant\frac{3}{4}p$$

对任意$y\in\{0,1\}^k$成立。实际上，对任意$x\in S$定义E_x表示$h(x)=y$这一随机事件，则有$\Pr[\exists x\in S:\ h(x)=y]=\Pr[\bigvee_{x\in S}E_x]$。根据容斥原理(推论 A.2)，该概率至少为

$$\sum_{x\in S}\Pr[E_x]-\frac{1}{2}\sum_{x\neq x'\in S}\Pr[E_x\cap E_{x'}]$$

但是，由哈希函数的两两独立性可知，如果$x\neq x'$，则$\Pr[E_x]=2^{-k}$且$\Pr[E_x\cap E_{x'}]=2^{-2k}$。因此，上述概率至少为(注意，$p=|S|/2^k$)

$$\frac{|S|}{2^k}-\frac{1}{2}\frac{|S|^2}{2^{2k}}=\frac{|S|}{2^k}\left(1-\frac{|S|}{2^{k+1}}\right)\geqslant\frac{3p}{4}$$

■

定理 8.13 的证明

在GNI的公用随机源交互式证明系统中，证明者和验证者在前面定义的集合S上运行集合下界协议若干次之后，验证者最后接受当且仅当在集合下界协议的所有运行中验证者接受的比例至少为$5p'/8$；注意，$p'=K/2^k$很容易被验证者计算。利用切尔诺夫界(定理 A.14)可以看到，运行集合下界协议常数次即可保证完备性概率至少为2/3而可靠性概率至多为1/3。

最后，由于验证者可以一次性选择多个 h，y 并将它们一起发送给证明者，也就是说，集合下界协议的多次运行可以通过一次并行执行来模拟，因此，交互回合数可以固定为 2。不难验证，即使并行地向证明者提问，上面分析得出的"证明者成功使验证者接受"的概率也不受影响。

评注 8.17 注意，与 GNI 的私有随机源协议不同，定理 8.13 中的公用随机源协议并不具有完美的完备性(即完备性概率不等于 1)，这是由于集合下界协议没有完美的完备性。但是，我们可以构造一个具有完备性参数 1 的集合下界协议(参见习题 8.5)，这意味着可以构造 GNI 的一个具有完美完备性的公用随机源证明。推广上述过程可以证明，任意私有随机源证明系统(即使它不具有完美的完备性)均可以转换为一个回合数大致相当的具有完美完备性的公用随机源证明系统。

155

8.2.3　定理 8.12 的证明概要

将 GNI 的私有随机源协议转换成公用随机源协议的过程也表明了如何将其他私有随机源协议转换成相应的公用随机源协议。此时，仍需关注原协议中能让私有随机源验证者进入接受状态的所有随机串构成的集合，基本思想仍是让公用随机源证明者向公用随机源验证者证明上述集合大小的近似下界。

思考一下，GNI 的公用随机源协议与 8.1.3 节中的私有随机源协议之间有何联系。粗略地讲，集合 S 对应于协议中验证者可能发送的所有消息，而验证者每次发送的消息是 S 中的一个随机元素。如果两个图是同构的，则验证者发送的消息完全隐藏了随机选自 $\{1, 2\}$ 的 i；如果两个图是不同构的，则验证者发送的消息将暴露 i 是 $\{1, 2\}$ 的哪个值(至少，在计算时间不受限时，i 的值将暴露给证明者)。因此，如果考虑从验证者的随机源到所发送消息之间的映射，则前一种情况下该映射是二对一的，而后一种情况下该映射是一对一的。这就是说，后一种情况下集合 S 的大小是前一种情况下集合 S 的大小的 2 倍。于是，私有随机源的验证者将高概率地接受"两个图不同构"；而事实上，公用随机源证明者恰好让验证者相信这一点。类似的思想也能用来证明 $\mathbf{IP}[k] \subseteq \mathbf{AM}[k+2]$。但是，证明过程必须利用上述思想考虑交互过程的每个回合中。在每个回合中，证明者必须向验证者证明"有些消息极可能被验证者发送"，换句话说，私有随机源协议中能让验证者发出这些消息的随机位串构成的集合是很大的。

8.2.4　GI 能是 NP-完全的吗

前面已经提到，GI 是否是 **NP**-完全问题仍然是一个待解决问题。下面证明，如果 GI 是 **NP**-完全问题，则多项式分层将坍塌。

定理 8.18 ([BHZ87])　如果 GI 是 **NP**-完全的，则 $\mathbf{\Sigma}_2^p = \mathbf{\Pi}_2^p$。

证明　我们证明，在假设条件下，$\mathbf{\Sigma}_2^p \subseteq \mathbf{\Pi}_2^p$；这意味着 $\mathbf{\Sigma}_2^p = \mathbf{\Pi}_2^p$，因为 $\mathbf{\Sigma}_2^p = \mathbf{co\Pi}_2^p$。

如果 GI 是 **NP**-完全的，则 GNI 是 **coNP**-完全的，这意味着，存在函数 f 使得对于任意 n 变量公式 φ 均有：$\forall_y \varphi(y)$ 成立当且仅当 $f(\varphi) \in \mathrm{GNI}$。考虑 $\mathbf{\Sigma}_2^p$SAT 的任意公式

$$\psi = \exists_{x \in \{0,1\}^n} \forall_{y \in \{0,1\}^n} \varphi(x, y)$$

公式 ψ 等价于

$$\exists_{x \in \{0,1\}^n} g(x) \in \mathrm{GNI}$$

其中 $g(x) = f(\varphi_{x,\cdot})$，而 $\varphi_{x,\cdot}$ 是 $\varphi(x, y)$ 固定 x 之后得到的公式。

156

由评注 8.17 和评注 8.11 可知，GNI 有一个完美完备性[⊖]的两回合 **AM** 交互式证明系统，并且(经过恰当的缩放)它的可靠性概率小于 2^{-n}。设 V 是该证明系统的验证者算法，用 m 表示验证者的随机带的长度，而 m' 表示发送给证明者的消息的长度。我们断言，ϕ 为真当且仅当

$$\forall_{r\in\{0,1\}^m}\exists_{z\in\{0,1\}^{m'}}\exists_{a\in\{0,1\}^m}(V(g(z),r,a)=1) \tag{8.6}$$

事实上，如果 ϕ 为真，则完美完备性显然蕴含(8.6)式。另一方面，如果 ϕ 为假，则意味着

$$\forall_{z\in\{0,1\}^m} g(z)\notin \mathrm{GNI}$$

现在，由于交互式证明的可靠性概率小于 2^{-n} 而 x 共有 2^n 种不同的取法，因此(根据概率方法基本原理)可以得出如下结论：存在一个串 $r\in\{0,1\}^m$ 使得对于任意 $x\in\{0,1\}^n$ 均有"如果 GNI 的 **AM** 交互式证明中验证者先发送消息 r 给证明者，则证明者不存在响应消息 a 来让验证者接受 $g(x)$"。换句话说，

$$\exists_{r\in\{0,1\}^m}\forall_{x\in\{0,1\}^n}\forall_{a\in\{0,1\}^m}(V(g(x),r,a)=0)$$

这正是(8.6)式的否定。由于"判定(8.6)式是否成立"是一个 Π_2^p 问题(因为该问题形如 $\forall x\exists y P(x,y)$，其中 P 是多项式时间可计算的谓词)，这就证明了 $\Sigma_2^p\subseteq\Pi_2^p$。 ■

8.3　IP = PSPACE

IP 是存在交互式证明的所有语言构成的类，刻画 **IP** 的性质曾经是一个悬而未决的问题。当时，人们已经知道 **NP**⊆**IP**⊆**PSPACE**，并且(二次非剩余语言和 GNI 语言存在交互式证明等)证据表明第一个包含关系是真包含。大多数研究人员觉得第二个包含关系也是真包含。他们的推理过程如下：由 8.1.1 节可知，交互过程本身不会产生任何不属于 **NP** 的语言。第 7 章也表明，随机性本身也不会赋予计算过程非常强大的计算能力，研究者甚至怀疑 **BPP**=**P**(有关这种怀疑的理由，将在第 20 章给出)。因此，交互过程与随机性的结合能增加多少计算能力呢？1990 以前能追溯到的所有证据均表明"不会增加太多"。对于任意固定的 k 值，**IP**[k]坍塌到类 **AM**=**AM**[2]。根据评注 8.11 可知，**AM**=**BP**·**NP**；并且 **BP**·**NP** 与 **NP** 看上去没"多大差别"[⊜]。最后，由于现有的交互式协议均不要求 k 是超常数，因此 **IP**=**IP**[poly(n)]看上去不比 **IP**[$O(1)$]大太多。上述直观认识是错误的，因为
157 1990 年人们证明了如下结论，它给出了 **IP** 的一个出人意料的性质。

定理 8.19 ([LFKN90, Sha90])　**IP**=**PSPACE**。

由前面的评注可知，我们只需证明非平凡的关系 **PSPACE**⊆**IP**。为此，只需证明 TQBF∈**IP**[poly(n)]，这是由于任意 $L\in$**PSPACE** 都可以多项式时间归约到 TQBF。为了完成证明，我们给出 TQBF 的一个公用随机源协议，使得如果输入属于 TQBF，则存在证明者让验证者概率为 1 地接受该输入。

我们先不直接为 TQBF 设计交互式协议，而是先考虑如何为 $\overline{\mathrm{3SAT}}$ 设计交互式协议。证明者如何才能让验证者相信给定的 3CNF 公式不存在满足性赋值呢？更一般地，我们将说明如何让证明者向验证者证明给定的 3CNF 公式恰有 K 个满足性赋值，其中 K 是某个常数。也就是说，我们先给出一个交互式证明来判定如下语言的成员资格。

定义 8.20 ($\#\mathrm{SAT}_p$)　$\#\mathrm{SAT}_p=\{\langle\phi,K\rangle:\phi$ 是一个 3CNF 公式且恰存在 K 个满足

⊖ 证明过程的后面部分也可以不依赖于完美完备性，具体细节留给感兴趣的读者。

⊜ 事实上，根据令人信服的复杂性猜想可以证明 **AM**=**NP**，参见习题 20.7。

性赋值）。

显然，SAT是$\#SAT_D$语言在$K=0$时的特例。第 17 章还将证明，$\#SAT_D$是一个更强大的类$\#\mathbf{P}$的完全问题。

注意，8.2.2 节中的集合下界协议可以近似地处理$\#SAT_D$，也就是说，在 2 倍因子（或任意常数倍因子）范围内证明 K 值。该协议只有两个回合。相比之下，我们为$\#SAT_D$设计的协议有 n 个回合。这个协议引入了算术化的思想，这种思想在为 *TQBF* 设计协议时也非常有用。

8.3.1　算术化

关键的思想是将布尔公式表示成多项式，进而用代数观点来研究布尔公式。注意，0，1 既可以视为真值，也可以视为某个有限域 $\mathbf{F}$ 中的元素。因此，$x_i \wedge x_j$取值为真当且仅当$x_i \cdot x_j = 1$在有限域中成立，而$\neg x_i$取值为真当且仅当$1 - x_i = 1$。

算术化指的是如下的技巧。给定一个有 m 个子句和 n 个变量的 3CNF 公式 $\varphi(x_1, x_2, \cdots, x_n)$，我们引入有限域中的变量 $X_1, X_2, \cdots, X_n$，并为每个规模为 3 的子句引入一个等价的 3 次多项式，例如

$$x_1 \vee \overline{x_2} \vee x_3 \leftrightarrow X_1(1 - X_2)X_3$$

令第 j 个子句对应的多项式为 $p_j(X_1, X_2, \cdots, X_n)$。尽管它在形式上依赖于 n 个变量，但前面的例子清楚地表明 p_j 至多依赖于三个变量。对于 $X_1, X_2, \cdots, X_n$的每个 0，1 赋值，如果相应的布尔变量赋值满足第 j 个子句，则 $p_j(X_1, X_2, \cdots, X_n) = 1$；否则，$p_j(X_1, X_2, \cdots, X_n) = 0$。

158

将所有多项式相乘得到多变量多项式 $P_\phi(X_1, X_2, \cdots, X_n) = \prod_{j=1}^{m} p_j(X_1, X_2, \cdots, X_n)$。它在布尔公式的满足性赋值上取值为 1，在非满足性赋值上取值为 0。该多项式的次数至多为 $3m$。我们将这个多项式表示为 3 次多项式的乘积，而不展开括号，这样 $P_\phi(X_1, X_2, \cdots, X_n)$具有规模为 $O(m)$的表示形式。上述将 φ 转换为 P_ϕ 的过程称为算术化。一旦得到多项式 P_ϕ，就可以将 0，1 替换成域 $\mathbf{F}$ 中的任意值，然后代入 P_ϕ 以计算多项式的值。下面将看到，这将使得交互式系统中的验证者拥有证明者意想不到的能力。

8.3.2　$\#SAT_D$的交互式协议

现在，我们证明如下结论。

定理 8.21　$\#SAT_D \in \mathbf{IP}$。

证明　给定输入$\langle \phi, K \rangle$，其中 ϕ 是一个有 m 个子句和 n 个变量的 3CNF 公式，我们用 8.3.1 节中的算术化技巧构造多项式 P_ϕ。ϕ 的满足性赋值的个数$\#\phi$满足

$$\#\phi = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\phi(b_1, \cdots, b_n) \tag{8.7}$$

证明者需要向验证者证明断言“上述和等于 K”。从现在起，我们抛开布尔公式，而只关注关于多项式 P_ϕ 的上述断言。

开始时，证明者将区间$(2^n, 2^{2n}]$中的一个素数 p 发送给验证者。验证者可以用概率型素性测试算法或确定型素性测试算法来验证 p 是素数。这里给出的所有计算均是在整数模 p 产生的有限域 $\mathbf{F} = \mathbf{F}_p$中进行的。注意，由于(8.7)式介于 0 到 2^n之间，因此该等式在整数上成立当且仅当它是模 p 成立的。因此，我们现在将(8.7)式视为 $\mathbf{F}_p$上的等式。为了完成

定理证明，我们给出一个一般的工具来验证形如(8.7)式的等式，这个工具称为和校验(sumcheck)。

和校验协议

给定 d 次多项式 $g(X_1, \cdots, X_n)$，整数 K 和一个素数 p，我们说明证明者如何通过交互式协议向验证者证明论断

$$K = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, \cdots, X_n) \tag{8.8}$$

其中，所有计算均是模 p 进行的。为确保协议能够被执行，验证者要求 g 具有如下性质：g 具有多项式规模(poly(n))的表示形式。这样，对于变量在域 $\mathbf{F}_p$ 上的每种赋值 $X_1 = b_1$，$X_2 = b_2$，$\cdots$，$X_n = b_n$，验证者能够在多项式时间内计算 $g(b_1, b_2, \cdots, b_n)$。如前所述，$g = P_g$ 满足该性质。

注意，对于 X_2，$\cdots$，X_n 的每种赋值 b_2，$\cdots$，b_n，$g(X_1, b_2, \cdots, b_n)$ 是变量 X_1 的 d 次一元多项式。因此，下式也是一个 d 次一元多项式：

159

$$h(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, b_2, \cdots, b_n) \tag{8.9}$$

如果论断(8.8)成立，则必有 $h(0) + h(1) = K$。

考虑如下协议。

> **协议：验证论断(8.8)的和校验协议**
>
> V：如果 $n = 1$，验证 $g(1) + g(0) = K$ 是否成立。如果是，则接受；否则拒绝。如果 $n \geqslant 2$，则要求证明者 P 发送(8.9)式定义的 $h(X_1)$。
>
> P：向验证者 V 发送一个多项式 $s(X_1)$(如果证明者不会"欺骗"验证者，则有 $s(X_1) = h(X_1)$)。
>
> V：如果 $s(0) + s(1) \neq K$，则拒绝。否则，从 $\mathbf{F}_p$ 中选择一个随机数 a。递归地利用本协议验证
>
> $$s(a) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_2, \cdots, b_n)$$

论断：如果(8.8)式不成立，则验证者将至少以概率 $\left(1 - \frac{d}{p}\right)^n$ 拒绝。

该论断蕴含定理。这是因为，如果(8.8)式成立，则证明者将让验证者以概率 1 接受；此外，根据我们选择的 p 可知，$\left(1 - \frac{d}{p}\right)^n$ 约等于 $1 - dn/p$，这个值非常接近于 1。

论断的证明　假设(8.8)式不成立。我们对 n 做归纳来证明论断。当 $n = 1$ 时，验证者 V 简单计算 $g(0)$，$g(1)$，如果二者的和不等于 K，则验证者将以概率 1 拒绝。假设论断对 $n-1$ 个变量上的 d 次多项式成立。

在第一个回合中，验证者 P 要求证明者发回了多项式 h。如果证明者确实返回 h，则由于论断假设 $h(0) + h(1) \neq K$，故验证者 V 立刻拒绝(亦即，以概率 1 拒绝)。因而，可以假设证明者返回的 $s(X_1)$ 不同于 $h(X_1)$。由于 d 次非零多项式 $s(X_1) - h(X_1)$ 至多存在 d 个根，故至多存在 d 个 a 满足 $s(a) = h(a)$。因而，当 V 随机地选取 a 时，

$$\Pr_a[s(a) \neq h(a)] \geqslant 1 - \frac{d}{p} \tag{8.10}$$

如果 $s(a)\neq h(a)$，则证明者在接下来的递归步骤中要证明的也是一个错误的论断。根据归纳假设，证明者证明"剩下的待证论断是错误的"失败的概率至少为 $\left(1-\frac{d}{p}\right)^{m-1}$。因而，我们有

$$\Pr[\text{V 拒绝}]\geqslant\left(1-\frac{d}{p}\right)\cdot\left(1-\frac{d}{p}\right)^{m-1}=\left(1-\frac{d}{p}\right)^{m} \tag{8.11}$$

这就完成了论断的证明，同时也完成了定理 8.21 的证明。 ■ 160

8.3.3　TQBF 的协议：定理 8.19 的证明

我们用类似的思想来为 TQBF 构造交互式协议：给定量化的布尔公式 $\Psi=\forall x_1\,\exists x_2\,\forall x_3\cdots\exists x_n\phi(x_1,\cdots,x_n)$，我们用算术化技巧构造多项式 P_ϕ。于是，$\Psi\in$ TQBF 当且仅当

$$\prod_{b_1\in\{0,1\}}\sum_{b_2\in\{0,1\}}\prod_{b_3\in\{0,1\}}\cdots\sum_{b_n\in\{0,1\}}P_\phi(b_1,\cdots,b_n)\neq 0 \tag{8.12}$$

首先，我们可能会想到采用同 $\neq\mathrm{SAT}_0$ 一样的协议来验证(8.12)式，只是由于(8.12)式中第一个变量 x_1 受全称量词 $\forall$ 约束，因而改为验证 $s(0)\cdot s(1)=k$ 是否成立；此后，对受全称量词 $\forall$ 约束的其他变量类似地进行处理。该协议基本上是正确的，只是它的运行时间不是多项式。事实上，与多项式相加不同，多项式相乘将使得多项式的次数增加。如果采用类似于(8.9)式的方法，通过在(8.12)中设置自由变量 X_1 来定义 $h(X_1)$，则其次数可能高达 2^n。该多项式的系数可能多达 2^n 个，因此它不能被转化为一个多项式时间验证器。

正确的方法需要注意到，(8.12)式断言的结论仅使用到{0，1}中的值并且 $x^k=x$ 对任意 $x\in\{0,1\}$ 和任意 $k\geqslant 1$ 成立。因此，原则上讲，我们可以将任意多项式 $p(X_1,\cdots,X_n)$转换成一个多变量线性多项式 $q(X_1,\cdots,X_n)$(亦即，$q(\cdot)$视为 X_i的多项式时，其次数至多为 1)，使得 $p(\cdot)$和 $q(\cdot)$在 $X_1,\cdots,X_n\in\{0,1\}$上取相同的值。具体地讲，我们引入多项式的线性化操作。对于任意多项式 $p(\cdot)$，令 $L_{X_i}(p)$(或简写为 $L_i(p)$)是如下多项式：

$$L_{X_i}(p)(X_1,\cdots,X_n)=X_i\cdot p(X_1,\cdots,X_{i-1},1,X_{i+1},\cdots,X_n)+(1-X_i)\cdot p(X_1,\cdots,X_{i-1},0,X_{i+1},\cdots,X_n) \tag{8.13}$$

于是，$L_i(p)$是 X_i的线性多项式，并且在 $X_1,\cdots,X_n\in\{0,1\}$上的取值与 $p(\cdot)$完全相同。因此，$L_1(L_2(\cdots(L_n(p))\cdots))$是一个多变量线性多项式，它在{0，1}上的取值与 $p(\cdot)$完全相同。

我们还需要将 $\forall X_i$和 $\exists X_i$也视为多项式操作，其中

$$\forall X_i p(X_1,\cdots,X_n)=p(X_1,\cdots,X_{i-1},0,X_{i+1},\cdots,X_n)\cdot p(X_1,\cdots,X_{i-1},1,X_{i+1},\cdots,X_n) \tag{8.14}$$

$$\exists X_i p(X_1,\cdots,X_n)=p(X_1,\cdots,X_{i-1},0,X_{i+1},\cdots,X_n)+p(X_1,\cdots,X_{i-1},1,X_{i+1},\cdots,X_n) \tag{8.15}$$

这样，(8.12)式的论断可以重述如下：如果在多项式 $P_\phi(X_1,X_2,\cdots,X_n)$上依次应用操作序列 $\forall X_1\,\exists X_2\,\forall X_3\cdots\exists X_n$(首先应用操作 $\exists X_n$，最后应用操作 $\forall X_1$)，则必将得到非零值 K。

如前所述，(8.12)式的断言仅考虑变量在{0，1}中的取值，因此，如果在该断言的首尾之间插入任意线性化操作序列，则该断言的正确性不受影响。于是，我们可以插入一些线性化操作使得和校验协议中的每个中间多项式都具有较低的次数。具体地讲，我们使用

如下的表达式：

$$\forall X_1 L_1 \exists X_2 L_1 L_2 \forall X_3 L_1 L_2 L_3 \cdots \exists X_n L_1 L_2 L_3 \cdots L_n P_\phi(X_1, \cdots, X_n)$$

[161] 该表达式的规模为 $O(1+2+3+\cdots+n)=O(n^2)$。

下面，我们归纳地定义交互式协议。假设有一个多项式 $g(X_1, \cdots, X_k)$，对于任意 $a_1, a_2, \cdots, a_k, C$，如果 $g(a_1, a_2, \cdots, a_k)=C$，则证明者能让验证者概率为 1 地相信 $g(a_1, a_2, \cdots, a_k)=C$；如果 $g(a_1, a_2, \cdots, a_k)\neq C$，则证明者能让验证者相信 $g(a_1, a_2, \cdots, a_k)=C$ 的概率小于 ε。令 $U(X_1, \cdots, X_l)$ 是如下定义的 l 个变量的任意多项式：

$$U(X_1, \cdots, X_l) = \mathcal{O}\, g(X_1, \cdots, X_k)$$

其中 $\mathcal{O}$ 是某个变量 X_i 上的 $\exists X_i$ 操作或 $\forall X_i$ 操作或 L_{X_i} 操作。因此，对前两类操作，l 等于 $k-1$；而对第三类操作，l 等于 k。令 d 是 U 中变量 x_i 的次数的一个上界，并假设验证者知晓这个上界；在我们的讨论中，$d\leqslant 3m$。下面将说明，对于任意 $a_1, a_2, \cdots, a_l, C'$，证明者在 $U(a_1, a_2, \cdots, a_l)=C'$ 时如何能让验证者概率为 1 地相信 $U(a_1, a_2, \cdots, a_l)=C'$，而在 $U(a_1, a_2, \cdots, a_l)\neq C'$ 时如何能使验证者相信 $U(a_1, a_2, \cdots, a_l)=C'$ 的概率小于 $\varepsilon+d/p$。

必要时对变量重命名，可以假设 $i=1$。验证者的校验过程如下进行：

情形 1：$\mathcal{O}=\exists X_1$。证明者将一个 d 次多项式 $s(X_1)$ 当作 $g(X_1, a_2, \cdots, a_k)$ 发送给验证者。

验证者验证 $s(0)+s(1)=C'$ 是否成立。如果不成立，则拒绝；如果成立，则随机选取 $a\in\mathbf{F}_p$ 并要求证明者证明 $s(a)=g(a, a_2, \cdots, a_k)$。

情形 2：$\mathcal{O}=\forall X_1$。过程与前一种情形一样，只需将 $s(0)+s(1)$ 替换为 $s(0)\cdot s(1)$。

情形 3：$\mathcal{O}=L_{X_1}$。证明者希望向验证者证明 $U(a_1, a_2, \cdots, a_k)=C'$。证明者将一个 d 次多项式 $s(X_1)$ 当作 $g(X_1, a_2, \cdots, a_k)$ 发送给验证者。

验证者验证 $a_1 s(0)+(1-a_1)s(1)=C'$ 是否成立。如果不成立，则拒绝；否则，随机选取 $a\in\mathbf{F}_p$ 并要求证明者证明 $s(a)=g(a, a_2, \cdots, a_k)$。

上述交互式证明系统是正确的，证明过程类似于 $\#\mathrm{SAT}_D$ 的情况。只需注意，如果 $s(X_1)$ 不是 $g(X_1, a_2, \cdots, a_k)$，则在上述三种情形中验证者未拒绝的概率为 $1-d/p$ 并转而在下一个回合中继续证明错误的结论。■

习题 8.8 勾勒了另一种证明定理 8.19 的思路。

8.4　证明者的能力

为了证明语言 L 的成员资格，许多交互式证明系统都要求证明者具有比"判定 L 的成员资格"强大得多的计算能力，这也是这些交互式证明系统共同的重要特征。下面列举两个这样的交互式证明系统。

1. 在定理 8.13 中，图不同构的公用随机源系统要求：证明者能够为随机选取的哈希函数 h 及其值域范围内随机选定的任意 y 产生一个图 H 使得 $h(H)$ 同构于 G_1 或 G_2 并且 $h(x)=y$。这显然强于证明图不同构（即使我们还无法给出图不同构的任何证明）。

2. **coNP** 语言 $\overline{\mathrm{3SAT}}$ 的交互式证明系统要求证明者能够计算 $\#\mathrm{SAT}_D$。但是，人们却无法确定 $\#\mathrm{SAT}_D$ 能否在多项式时间内被计算，即使以 3SAT 为神喻也如此。事实上，在第 17 章我们将看到 $\#\mathrm{SAT}_D$ 是 $\#\mathbf{P}$-完全的，因此，能够在多项式时间内计算 $\#\mathrm{SAT}_D$ 将意味着 $\mathbf{PH}\subseteq\mathbf{P}^{\#\mathrm{SAT}_D}$。

上述两个交互式系统能否使用弱一些的证明者呢？这仍是悬而未决的问题。相比之

下，TQBF 协议的不同之处在于，该协议只要求证明者具备判定 TQBF 语言的计算能力；这背后的原因如前所述：证明者给出的响应可以在 **PSPACE** 内被计算，继而这些计算都可以归约到 TQBF。这一观察结果奠定了如下结论的基础。该结论本质上与第 6 章中的卡普-利普顿定理一样，只是该结论要强一些，这是由于 **MA** 含于 Σ_2^p 中(事实上，语言 L 的 **MA**-证明系统及其完美完备性直接表明了 $L\in\Sigma_2^p$)。第 20 章中引理 20.18 给出了一个相关的结论。 162

定理 8.22　如果 $\mathbf{PSPACE}\subset\mathbf{P}_{poly}$，则 **PSPACE**＝**MA**。

证明　如果 $\mathbf{PSPACE}\subset\mathbf{P}_{poly}$，则 TQBF 交互式协议中的证明者可以替换为一个多项式规模的线路。梅林(亦即证明者)可以在第 1 个回合中将该线路交给亚瑟(亦即验证者)，然后验证者就可以直接在交互式证明中使用该线路，而不再需要其他交互行为。注意，亚瑟无需盲目地信任梅林交给的线路，因为 TQBF 协议的正确性证明已经表明：如果输入的公式非真，则没有证明者能够让亚瑟高概率地接受该公式。 ■

8.5　多证明者交互式证明

交互式证明系统也可以有多个证明者。这种系统的一个重要的假设是：*交互式协议执行期间，证明者之间不能互相交流。*但是，所有证明者可以在交互式协议开始执行前进行交流，以便在回答问题的策略上达成一致。这种交互式协议通常可以类比为警察在不同房间同时讯问两名犯罪嫌疑人。犯罪嫌疑人在讯问开始前可能已经对"向警察讲述哪种情节"达成了一致，但是自从对他们的隔离讯问开始后，他们却可能会不经意地暴露出不一致性的情节。

多证明者交互式证明系统能够判定的所有语言构成的集合称为 **MIP**，其形式定义类似于定义 8.6。我们假定证明系统中有两个证明者(注意，即使允许使用多项式个证明者，所定义的复杂性类也不会变化，参见习题 8.12)，并且，在交互过程的每个回合中验证者向每个证明者询问一个问题，两名证明者被咨询的问题不一定相同。每名证明者每个回合给出所问问题的一个答案。

显然，**IP**⊂**MIP**，这是因为验证者可以忽略一个证明者。此外，还可以证明，**MIP** 严格大于 **IP**(除非 **PSPACE**＝**NEXP**)。这就得到如下的定理。

定理 8.23　([BFL90])　**NEXP**＝**MIP**。

第 11 章将进一步讨论定理 8.23，同时还将讨论另一个类 **PCP**。直观上，两个证明者比一个证明者计算能力更强大的原因在于可以用第二位证明者来处理非适应性。也就是说，将交互式证明视为一个"讯问过程"，其中验证者向每名证明者提问并要求证明者对所问的问题给出答案。如果验证者希望证明者对问题 q 给出的答案仅是 q 的函数而不依赖于证明者之前所听到的其他问题，则验证者可以向第二位证明者讯问问题 q。只有两位证明者对问题 q 给出的答案是一致的，验证者才接受该答案。这种技术曾被用来证明多证明者交互式证明可以用作"概率可验证明"概念模型的一种实现(事实上，二者是等价的)。在概率可验证明模型中，(类似于 **NP** 问题的证明一样)证明将被视为一个静态的位串，由于该串可能非常长，因而最好将它视为一个大表，其中包含了证明者对所有可能被验证者咨询的问题的答案。为了验证证明的正确性，验证者仅需根据某种分布随机选定表格中的少数几个位置并验证表格中的选定位置上的串是否正确。如果用类 **PCP**$[r, q]$ 表示"查验大小为 2^r 的表格中 q 个位置上的串是否正确即可判定其成员资格的"所有语言构成的集合，则 163

定理 8.23 可以重述为：

定理 8.24（定理 8.23 的重述）　$\mathbf{NEXP}=\mathbf{PCP}[\mathrm{poly},\ \mathrm{poly}]=\bigcup \mathbf{PCP}[n^c,\ n^c]$。

人们已经证明，定理 8.23 可以缩减到更小的复杂性类上以得到 $\mathbf{NP}=\mathbf{PCP}[\mathrm{polylog},\ \mathrm{polylog}]$。事实上，经过繁琐的推导，人们证明了下面的定理；参见第 11 章。

定理 8.25（**PCP** 定理，[AS92，ALM+92]）　$\mathbf{NP}=\mathbf{PCP}[O(\log n),\ O(1)]$。

上述定理将出现在第 11 章，其证明将出现在第 22 章。该定理在复杂性理论中有许多应用；特别地，它可以用来证明，对于许多 **NP**-完全的优化问题，获取其近似最优解同获取其最优解一样难。由此可见，在复杂性理论中交互式证明系统构建了如下的完全环：从 **NP** 开始，逐渐向其中添加交互过程、随机化方法和多个证明者，将得到复杂性很高的类 **NEXP**，最后又得到了关于类 **NP** 的全新的基础性解释。

8.6　程序检验

有一个称为程序检验的研究领域，有时也称为实例检验，它直接推动了 $\#\mathrm{SAT}_D$ 的交互式协议的发现。布卢姆(Blum)和坎纳安(Kannan)奠定了程序检验领域的基础。他们最早注意到，尽管程序验证是不可判定的(也就是说，无法判定一个程序是否在所有输入上均正确地求解了某个计算任务)，但在许多情况下却只需逐个输入地检验程序的正确性就足够。程序验证程序这一概念恰好概括了上述含义。程序 P 的验证程序 C 可以将 P 当作子程序来调用。当 P 在任意一个输入上运行时，C 的任务就是验证 P 在该输入上得到的答案是否正确(或是否存在 bug)。为此，允许验证程序计算 P 在其他输入上的答案。形式地讲，验证程序 C 是将其他程序代码视为一个黑盒作为输入的图灵机。将验证程序 C 调用子程序 P 时得到的输出记为 C^P。

定义 8.26　设 T 是一个计算任务。T 的一个验证程序是一个概率型多项式时间图灵机 C，它对于声称"在任意输入 x 上求解了计算任务 T"的每个程序 P 均有：

1. 如果 P 是计算任务 T 的正确程序(亦即 $P(y)=T(y)$ 对任意 y 成立)，则 $\Pr[C^P\text{接受}P(x)]\geqslant\frac{2}{3}$。

164

2. 如果 $P(x)\neq T(x)$，则 $\Pr[C^P\text{接受}P(x)]<\frac{1}{3}$。

注意，验证程序并不是证明程序的正确性。而且，即使 P 在输入 x 上正确(即 $P(x)=T(x)$)而在除 x 之外的其他输入上不正确，验证程序的输出也可能是任意的。

出人意料的是，对于许多问题，对求解问题的程序进行验证比实际计算问题的答案本身要容易得多。因此，根据布卢姆和坎纳安的建议，验证程序应该嵌入求解这类问题的软件系统中，这样，软件系统的程序能够自动验证其计算工作但验证程序造成的开销却可以忽略。

例 8.27（图不同构的验证程序）　图不同构问题的输入是标签图构成的序对 $(G_1,\ G_2)$，问题要求判定 $G_1\ncong G_2$ 是否成立。如前所述，目前人们还没有高效的算法来求解这个问题。但是，该问题却有高效的验证程序。

根据程序声称 $G_1\ncong G_2$ 是否成立，可以将程序的输入分为两类。如果程序声称 $G_1\ncong G_2$，则通过对图进行细微调整并利用程序，可以实际地构造出一个将 G_1 映射为 G_2 的置换 π；如果构造不出这样的置换，则可以找出程序的 bug(参见习题 8.11)。下面，我们说明如何

利用前面给出的图不同构的交互式证明来检验程序声称的 $G_1 \ncong G_2$。

回顾一下，在图不同构的交互式证明中：

- 如果证明断言 $G_1 \ncong G_2$，则重复下列步骤 k 次：
- 随机选择 $i \in_R \{1, 2\}$，然后将 G_i 的顶点进行随机置换得到 H；
- 询问证明者 G_i 和 H 是否同构并检验此次得到的答案与之前的答案是否一致。

如果计算机程序 P 声称能够判定图是否同构，如何验证它是否正确呢？程序检验方法告诉我们，将计算机程序 P 视为 IP 中的证明者即可。令 C 是验证者为 P 且执行 IP 协议的验证程序。

定理 8.28 如果 P 是正确求解图不同构问题的程序，则 C 总输出"正确"。否则，如果 $P(G_1, G_2)$ 不正确，则 $\Pr[C \text{ 输出"正确"}] \leqslant 2^{-k}$。并且，$C$ 的运行时间是多项式时间。

8.6.1　具有验证程序的语言

只要语言 L 存在一个交互式证明系统且它的证明者仅以 L 为神喻，则语言 L 有验证程序。因此，根据我们现在已经看到的交互式证明系统可以直接得到下面的定理。

定理 8.29 图同构 GI，$\#\mathrm{SAT}_0$ 和真量化布尔公式问题 TQBF 均有验证程序。

类似地，可以证明[Rub90]，随机自归约问题和向下自归约问题也都存在验证程序。向下自归约性的定义参见习题 8.9。

基于 **P**-完全语言在 **NC** 归约下(其实，在稍弱的对数空间归约上也如此)可以互相归约这一事实，为了证明如下有意义的结论，只需为某一个 **P**-完全语言给出一个在 **NC** 中的验证程序，这项工作最先由布卢姆和坎纳安完成。 165

定理 8.30 任意 **P**-完全语言均存在一个属于 **NC** 的验证程序。

由于人们相信 **P**-完全语言不能在 **NC** 内被计算，因此，上述结论再次提供了证明"验证比实际计算容易"的证据。

对于那些验证程序用到交互式证明系统的语言，布卢姆和坎纳安实际上还刻画了它们的性质。此处从略，因为这个结论的技术性太强了。

8.6.2　随机自归约与积和式

大多数验证程序的设计都基于如下的观察：一个程序在 x 上的输出往往与它在其他输入上的输出有关联。这些关联关系中，最简单的一种称为随机自归约。许多问题都具有随机自归约性质。

粗略地讲，一个计算问题具有随机自归约性指的是，求解该问题在任意输入 x 上的解可以归约为求解该问题在均匀随机抽取的一系列输入 y_1，y_2…上的解。完整精确的定义更具一般性，也更具技术性，此处仅需上述模糊定义即可。随机自归约性对于计算问题的平均复杂性的理解非常重要；有关于此，定理 8.33 和 19.4 节进行了进一步的阐述。

例 8.31 假设 $f: \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ 是一个线性函数；亦即，存在系数 $a_1, a_2, \cdots, a_n$ 使得 $f(x_1, x_2, \cdots, x_n) = \sum_i a_i x_i$。

那么，对于任意 $x, y \in \mathrm{GF}(2)^n$，我们有 $f(x)+f(y)=f(x+y)$。这一事实可以用来证明 f 的计算满足随机自归约性。如果需要对任意 x 计算 $f(x)$，只需任取 y 并计算 $f(y)$ 和 $f(x+y)$，此时 y 和 $x+y$ 均是 $\mathrm{GF}(2)^n$ 中的随机向量(尽管它们不是独立分布的)。此

外，提请大家注意：第 11 章将在不同的场合下利用上述观察结果来测试线性性质。 ◀

或许，例 8.31 显得过于平凡，但事实上，有些非常不平凡的问题也是随机自归约的。矩阵的积和式与行列式非常相像，其定义如下。

定义 8.32 设 $A\in F^{n\times n}$ 是域 F 上的矩阵。A 的积和式指的是

$$\mathrm{perm}(A)=\sum_{\sigma\in S_n}\prod_{i=1}^{n}a_{i,\sigma(i)} \tag{8.16}$$

积和式的计算问题显然属于 **PSPACE**。第 17 章将证明，积和式计算问题是 $\#\mathbf{P}$-完全
166 的，这将表明，积和式计算问题本质上与 $\#\mathrm{SAT}_0$ 等价。特别地，如果积和式能够在多项式时间内计算，则 $\mathbf{P}=\mathbf{NP}$。这里，我们仅证明积和式计算问题是随机自归约的。证明过程要用到的主要观察事实是，如果将积和式视为（表示矩阵 A 的所有元素的）n^2 个变量的函数，则(8.16)式表明该函数是一个 n 次多项式。

定理 8.33（利普顿[Lip91]） 存在一个随机算法使得只要给定能够在 $\mathbf{F}^{n\times n}$ 中比例为 $1-\frac{1}{3n}$ 的输入上正确计算积和式的神喻（其中域 $\mathbf{F}$ 的大小 $>3n$），则该随机算法能够高概率地为所有输入正确计算积和式。

证明 令 A 是某个输入矩阵。随机选取矩阵 $R\in_R\mathbf{F}^{n\times n}$，并令 $B(x)=A+x\cdot R$，其中 x 是一个变量。注意，

- $\mathrm{perm}(B(x))$ 是一个一元 n 次多项式。
- 对于任意固定的 $a\neq 0$，$B(a)$ 是一个随机矩阵，因此，神喻能正确计算 $\mathrm{perm}(B(a))$ 的概率至少为 $1-\frac{1}{3n}$。

现在，可以直接得到计算 A 的积和式的随机算法如下。取定域中的 $n+1$ 个不同的值 $a_1,a_2,\cdots,a_{n+1}$，然后向神喻咨询所有矩阵 $\{B(a_i)\mid 1\leqslant i\leqslant n+1\}$ 的积和式。根据合并界限，神喻为所有矩阵正确计算积和式的概率至少为 $1-\frac{n+1}{3n}\approx\frac{2}{3}$。

回顾一下如下事实（参见定理 A.35），给定 $n+1$ 个（点或值的）有序对 $\{(a_i,b_i)\mid i\in[n+1]\}$，则存在唯一的 n 次多项式 p 使得 $p(a_i)=b_i$ 对任意 i 成立。于是，只要值 $B(a_i)$ 是正确的，则随机算法便可以得到插值多项式 $B(x)$，进而计算得到 $B(0)=\mathrm{perm}(A)$。 ■

定理 8.33 的假设可以放宽到只要求神喻能够在 $\frac{1}{2}+\varepsilon$ 比例的输入上正确计算积和式，其中 $\varepsilon>0$ 是任意常数。证明放宽后的结论要用到更强的插值定理，参见 19.6 节。

8.7　积和式的交互式证明

由 $\#\mathrm{SAT}_0$ 和 TQBF 存在交互式证明协议可知，积和式计算问题也存在交互式证明协议。尽管如此，本节将为积和式计算问题设计专门的交互式证明协议。这样做的原因有两个。一方面，从历史上看，积和式的交互式证明协议早于另两个问题的交互式证明协议。另一方面，积和式的交互式证明协议在后续章节的研究中十分有用。

积和式的交互式证明协议要用到积和式的随机自归约性和向下自归约性，其中向下自归约性是我们在第 2 章处理 SAT 问题时遇到的性质（也可以参见习题 8.9）。对于积和式计算问题，向下自归约性指的是如下观察事实：

$$\mathrm{perm}(A)=\sum_{i=1}^{n}a_{1,i}\mathrm{perm}(A_{1,i})$$

其中 $A_{i,j}$ 是从 A 中删除第 1 行和第 i 列之后得到的 $(n-1)\times(n-1)$ 的矩阵(回顾一下，行列式也有类似的公式，只是其中要用到交错的正、负号)。因此，计算 $n\times n$ 的积和式可以归约到计算 n 个 $(n-1)\times(n-1)$ 矩阵的积和式。 167

在记号方面再做一些约定。我们假设 $\mathbf{F}$ 等于域 $\mathrm{GF}(p)$，其中素数 $p>n$；因而，$1, 2, \cdots, n\in\mathbf{F}$，而且，$a_{ij}$ 表示矩阵第 i 行第 j 列的元素。对于任意 $n\times n$ 的矩阵 A 和 $i\in[n]$，我们定义 $D_A(i)$ 表示 $(n-1)\times(n-1)$ 的矩阵 $A_{i,*}$。如果 $x\in\mathbf{F}\setminus[n]$，则用唯一的方式定义 $D_A(x)$ 使得对于任意 $j, k\in[n-1]$，函数 $(D_A(x))_{j,k}$ 为一个次数不超过 n 的一元多项式。注意，由于 $(n-1)\times(n-1)$ 矩阵的积和式是矩阵元素的 $n-1$ 次多项式，因此 $\mathrm{perm}(D_A(x))$ 是次数不超过 $(n-1)n<n^2$ 的一元多项式。

8.7.1　协议

下面给出积和式的交互式证明协议。具体地讲，定义 L_{perm} 是所有如下的三元组 (A, p, k) 构成的语言，其中 A 是域 $\mathrm{GF}(p)$ 上的一个 $n\times n$ 矩阵，$p>n^2$ 是素数，且 $\mathrm{perm}(A)=k$。我们往证下面的定理。

定理 8.34　$L_{\mathrm{perm}}\in\mathbf{IP}$。

证明　我们用归纳法证明定理。假设维数小于等于 $n-1$ 的矩阵存在交互式证明协议，下面为 $n\times n$ 的矩阵构建交互式证明协议。也就是说，我们假设对于 $(n-1)\times(n-1)$ 的任意矩阵 B，如果 $\mathrm{perm}(B)=k'$，则证明者能够让验证者概率为 1 地接受 $\mathrm{perm}(B)=k'$；而如果 $\mathrm{perm}(B)\neq k'$，则证明者能够让验证者接受 $\mathrm{perm}(B)=k'$ 的概率至多为 ε(在基本情况 $n-1=1$ 时，验证者可以直接计算积和式，因而此时 $\varepsilon=0$)。在此假设下往证，对 $n\times n$ 的任意矩阵 A，如果 $\mathrm{perm}(A)=k$，则证明者能够让验证者概率为 1 地接受 $\mathrm{perm}(A)=k$；而如果 $\mathrm{perm}(A)\neq k$，则证明者能够让验证者接受 $\mathrm{perm}(A)=k$ 的概率至多为 $\varepsilon+(n-1)^2/p$。下面的信息交换过程能实现上述目标：

- 第 1 回合：证明者将一个次数不超过 $(n-1)^2$ 的多项式 $g(x)$ 当作 $\mathrm{perm}(D_A(x))$ 发送给验证者。
- 第 2 回合：验证者验证 $k=\sum_{i=1}^{n}a_{1,i}g(i))$ 是否成立。如果不成立，验证者立刻拒绝输入。否则，验证者随机选择一个元素 $b\in_R\mathbf{F}$，并要求证明者证明 $g(b)=\mathrm{perm}(D_A(b))$。注意，$D_A(b)$ 是 $\mathbf{F}$ 上的一个 $(n-2)\times(n-2)$ 矩阵，因此，由归纳假设，可以设计一个验证该结论的交互式协议。

下面分析上述协议。如果 $\mathrm{perm}(A)=k$，则全能型证明者必然能给出 $\mathrm{perm}(D_A(x))$；于是，由归纳假设可知，证明者必能让验证者概率为 1 地接受 $\mathrm{perm}(A)=k$。

另一方面，假设 $\mathrm{perm}(A)\neq k$。如果第 1 回合中发送的多项式 $g(x)$ 就是正确的多项式 $\mathrm{perm}(D_A(x))$，则

$$\sum_{i=1}^{n}a_{1,i}g(i)=\mathrm{perm}(A)\neq k$$

此时，验证者将立刻拒绝。因此，仅需考虑证明者发送的多项式 $g(x)\neq\mathrm{perm}(D_A(x))$ 的情况。由于次数为 $(n-1)^2$ 的两个多项式至多在 $(n-1)^2$ 个 x 上取相同的值，于是，随机选取的 $b\in\mathbf{F}$ 恰好满足 $g(b)=\mathrm{perm}(D_A(b))$ 的概率至多为 $(n-1)^2/p$。如果 b 不满足 $g(b)=\mathrm{perm}(D_A(b))$，则证明者将陷入“试图证明错误结论”的困境。由归纳假设可知，在此条件下证明者能让验证者接受错误结论的概率至多为 ε。这就得到了交互式证明的正确性。 168

将上述归纳过程展开可以看到，证明者能让验证者接受 $n\times n$ 矩阵的错误积和式的概率不超过

$$\frac{(n-1)^2}{p}+\frac{(n-2)^2}{p}+\cdots+\frac{1}{p}\leqslant\frac{n^2}{p}$$

由 p 的取法可知，上述概率远远小于 1/3。 ■

本章学习内容

- 交互式证明是数学证明的一种推广，其中证明者与概率型多项式时间验证者进行交互。
- 允许使用随机方法和交互过程将显著地提高证明系统的能力：如果 **IP** 是能被多项式时间的交互式证明证明的所有语言构成的类，则 **IP** 等于 **PSPACE**。
- 所有能够被常数回合的交互式证明系统判定的语言构成类 **AM**；也就是说，这些语言有一个交互式证明系统，其中验证者仅发送一个随机串给证明者而证明者也仅回复一个消息。
- 交互式证明与密码学、近似算法(特别是近似算法的存在性)以及程序检验之间存在非凡的联系。

本章注记和历史

1985 年，戈德瓦瑟(Goldwasser)、米卡利(Micali)和拉科夫(Rackoff)给出了交互式证明的定义并将它应用于密码学；同时，巴拜(Babai)独立地定义了公用随机源交互式证明[Bab85](也可参见巴拜和莫兰(Moran)[BM88])。图不同构的私有随机源交互式证明源自戈德赖希(Goldreich)、米卡利和维格德尔森[GKW87]。用公用随机源模拟私有随机源的方法源自戈德瓦瑟和西普赛尔[GS87]。[Gol08，附录 A]对此给出了精彩的完整证明。这一结论深受早期的一些结果(如 **BPP**$\subseteq$**PH**)的影响(7.5.2 节)，同时也受到"$\#\mathrm{SAT}_D$可以在 $\mathbf{P}^{\mathbf{NP}}$ 内近似求解"这一事实的影响。本·欧尔(Ben-Or)等人定义了多证明者交互式证明，他们的目的是用这个概念在无密码学假设的前提下来获得 **NP** 的零知识证明系统(参见 9.4 节)。

当时，研究者们普遍认为交互式证明仅"稍微地"扩展了 **NP**，而且连$\overline{\mathrm{3SAT}}$也不一定存在交互式证明。例如，福特劳(Fortnow)和西普赛尔(Sipcer)[FS88]曾猜想了上述结论，并给出了一个相对于 $\mathbf{coNP}^{\mathbf{NP}}\subseteq\mathbf{IP}^{O}$ 的神喻 O；也就是说(用 3.4 节的术语来说)，**IP**$=$**PSPACE** 不是一个相对性定理。

IP$=$**PSPACE** 这一结果大大地出人意料，该结论被发现的故事非常有趣。在 20 世纪 80 年代晚期，布卢姆(Blum)和坎纳安(Kannan)[BK95]引入了程序检验的概念。大约同一时期，毕威尔(Beaver)和费根鲍姆(Feigenbaum)[BF90]以及利普顿[Lip91]分别发表了论
169 文，它们丰富了随机自归约的概念与程序检验之间的联系。受到这些进展的推动，尼散(Nisan)在 1989 年 11 月证明了积和式问题(进而$\#\mathrm{SAT}_D$问题)存在多证明者交互式证明。他通过电子邮件将证明过程通知了他的几位同行，然后就去南美度假去了。这封邮件在全世界研究界掀起了一阵研究热潮。伦德(Lund)、福特劳和卡尔洛夫(Karloff)证明了$\#\mathrm{SAT}_D$属于 **IP**，他们将尼散加为合著者，最后发表了论文[LFKN90]。此后，萨米尔(Shamir)证明了 **IP**$=$**PSPACE**[Sha90]，巴拜、福特劳和伦德[BFL90]证明了 **MIP**$=$

NEXP。巴拜在其幽默风趣的综述[Bab90，Bab94]中讲述了上述故事，并介绍了后来的一系列研究进展，包括 **PCP** 定理；也可以参见第 11 章的章节注记。

借助线性化操作证明 **IP**＝**PSPACE** 源自 Shen[She92]。证明者的能力问题与"判定问题与搜索问题"的复杂性相关，贝拉雷(Bellare)和戈德瓦瑟[BG94]对此进行了研究，也可以参见瓦德翰(Vadhan)[Vad00]。戈德瓦瑟等人[GGH+07]已经将定理 8.30 推广到 **NC** 中的语言。

正如本章的引言所述，最短向量的 $\sqrt{n/\log n}$ 近似问题属于 **AM**[2]，因而不太可能是 **NP**-难的；上述结论源自戈德赖希和戈德瓦瑟[GG98]。阿哈诺夫(Aharonov)和雷格夫(Regev)[AR04]证明了最短向量的 $\sqrt{n}$ 近似问题属于 **NP**∩**coNP**。

习题

8.1　证明 8.1 节对 **IP** 的所有断言。亦即：

(a) 令 **IP**′是允许定义 8.6 使用概率型证明者时得到的复杂性类，其中概率型证明者所使用的策略可以是从服从某种分布的函数中随机选取的。证明：**IP**′＝**IP**。

(b) 证明：**IP**⊆**PSPACE**。

(c) 令 **IP**′是将(8.2)式中的 2/3 改为 1 之后得到的复杂性类。证明：**IP**′＝**IP**。

(d) 令 **IP**′是将(8.3)式中的 1/3 改为 0 之后得到的复杂性类。证明：**IP**′＝**IP**。

8.2　在(8.2)式的完备性条件中，如果要求对每个 $x\in L$ 存在一个单独的证明者 P(而不是要求对任意 $x\in L$ 存在一个证明者)，将新定义的复杂性类记为 **IP**′。证明：**IP**′＝**IP**。

8.3　证明：**AM**[2]＝**BP**·**NP**。

8.4　令 $k\leqslant n$。证明：如下定义的 $\mathcal{H}_{n,k}$ 是从 $\{0,1\}^n$ 到 $\{0,1\}^k$ 的一个两两独立函数族。将 $\{0,1\}$ 等同于域 GF(2)，对 GF(2) 上的每个 $k\times n$ 的矩阵 A 和 $\mathbf{b}\in \mathrm{GF}(2)^k$，$\mathcal{H}_{n,k}$ 包含函数 $h_{A,\mathbf{b}}$：$\mathrm{GF}(2)^n\to\mathrm{GF}(2)^k$，其中 $h_{A,\mathbf{b}}(x)=Ax+\mathbf{b}$。

8.5　证明：存在用于证明集合大小的下界的完美完备 **AM**[$O(1)$]协议。　170

8.6　证明：对于语言 L 的任意 **AM**[2]协议，如果证明者和验证者并行地执行该协议 k 次(亦即，验证者发送给证明者的消息包含 k 个独立的随机串)并且仅在协议的 k 次执行中验证者均接受之后他才最终接受，则验证者接受 $x\notin L$ 的概率至多为 $(1/3)^k$。(注意，你不能假定证明者在各次协议执行时相互独立。)你能推广证明使它对任意 k 成立吗？

8.7　(巴拜-莫兰[BM88])证明：**AM**[$k+1$]⊆**AM**[k]对任意 $k\geqslant 2$ 成立。

8.8　本题推广 **coNP**⊆**IP** 的证明，并由此给出 **IP**＝**PSPACE** 的另一种证明。

(a) 假设 φ 是一个 QBF 公式，它不必具有前缀形式，但必须满足如下性质：如果 φ 的所有变量依先后出现的次序列出来是 x_1，…，x_n。则对于每个 x_i，在 φ 的最后一个包含 x_i 的子句之前至多存在一个全称量词作用于满足 $j>i$ 的变量 x_j 上。证明：如果将 8.3.2 中的和校验协议修改为 $s(0)\cdot s(1)=K$ 以适于乘积运算，则对于具备上述性质的公式 φ，证明者只需发送一个 $O(n)$ 次多项式给验证者。

(b) 证明：规模为 n 的任意 QBF 公式 φ 均可以逻辑等价地转换为一个满足上述性质的规模为 $O(n^2)$ 的公式 φ。

8.9　对于语言 L，如果存在一个多项式时间算法 R 使得 $R^{L'}(x)=L(x)$ 对任意 n 和任意

$x \in \{0,1\}^n$成立，其中L_n表示能在规模至多为k的输入上判定L的神喻，则称语言L是向下自归约的。证明：如果L是向下自归约的，则$L \in \mathbf{PSPACE}$。

8.10　完成例8.27的证明，并证明图同构也存在验证程序。具体地讲，你需要证明，如果程序声称$G_1 \cong G_2$，则我们可以做进一步验证(包括在其他输入上调用程序)并高概率地得到如下两个结论之一：(a)程序在该输入是正确的；(b)该程序在某个输入上是错误的，因而它不是一个正确的图同构程序。

8.11　证明：$\mathbf{MIP} \subseteq \mathbf{NEXP}$。

8.12　证明：如果重新定义多证明者交互式证明，允许在规模为n的输入上使用$m(n)=\mathrm{poly}(n)$个证明者，而不是仅用两个证明者，则类**MIP**不会改变。

171

第9章

Computational Complexity: A Modern Approach

密 码 学

人类的聪明才智无法编制出人类智慧无法破译的密码。

——埃德加·阿兰·伯奕(E. A. Poe), 1841

为了设计出色的密码，仅仅确信"标准的密码分析方法无法破译"是不够的……我们必须确保任何方法都无法轻易破解密码系统。而这事实上恰好是许多密码系统的弱点所在……研制出色的密码本质上是在某些约束条件下寻找难解问题。这其实是一项非同寻常的任务，因为人们通常仅擅长在某个领域内寻找简单而易于求解的问题。

——C. 香农[Sha49b]

NP 完全问题有望用于密码研制，但是人们目前对这些问题难度的理解都仅限于平均复杂度。要将它们用于密码系统，必须分析它们在典型情况下的复杂性。

——W. 迪菲(W. Diffie), M. 赫尔曼(M. Hellman)[DH76]

密码学远比计算复杂性古老得多。自从人类学会书写起，人们就发明了"密写术"以防止他人解密书写的内容。但是，千百年来人们所设计的加密方法或"密写术"均遭遇了相同的命运——使用不久之后就被破解了。20 世纪 70 年代之后，情况从根本上得到了改变。当时，几名研究者的工作催生了现代密码学，它用计算复杂性来阐述加密方案的安全性。现在看来，密码学和计算复杂性之间的这种联系是非常自然的，因为密码破译者只有有限的计算资源(尽管她或许会争辩说她有计算机)。因而，要保证密码系统的安全性，我们应该确保密码的破译问题是难计算的。

现代密码学与传统密码学之间的另一个显著区别是，加密的安全性不再依赖于对加密技术本身的保密。在现代密码学中，加密技术虽然是众所周知的，但它仍难以被攻破。并且，现代密码学除了加密之外，它还涉及在各种场合下设计安全可靠的计算方案。这些计算方案的安全性也是通过归约技术来证明的，但这里的归约技术类似于但不同于 **NP** 完全性理论中的归约技术。

现代密码学的中心任务就是将归约过程作用到一些基本问题上来构建密码系统，这一任务使得现代密码学能够同时达成两个看似自相矛盾的目标。一方面，现代密码学方法获得的加密方案将更加安全可靠。例如，公钥密码体系(RSA)诞生后[RSA78]，卓越的数学家借助最尖端的计算机对它持续地实施了前所未有的攻击。另一方面，现代密码学对新密码系统的安全性要求更加严格：有的密码系统(如公钥密码)要求，即使攻击者知晓密钥，加密方案仍是安全的；有的密码系统要求，即使攻击者可以有选择地获得密文及其解密文本(亦即在选择明文攻击或者在选择密文攻击下)，加密方案仍是安全的。此外，现代密码学的任务远不止简单地提供加密方案，它还要提供各种其他工具的方案，这些工具包括数字签名、零知识证明、电子投票与审计，等等。已经证明，所有这些工具的方案在任意多项式时间的攻击下都是安全的，这些攻击不仅仅局限于现在能想到的攻击方式，只要这些方案背后的计算问题仍是难解的，则它们将永远是安全的。

现代密码学的研究产生了一些重要的概念，这些概念对复杂性理论和其他领域的研究

具有重要影响和应用。第一个重要的概念是伪随机数。哲学家和科学家对如何量化一个位串"是否足够随机"而争论了多年。密码学家对此的解答是，只要一个串的分布对高效的观察者(如多项式时间观察者)而言"看起来是随机的"，则可以认为该串是"足够随机的"(参见9.2.3节)。这一概念不仅在构建许多加密方案时至关重要，同时，它对于需要随机位的其他领域也极其重要。例如，密码学中的伪随机数产生器可以用来降低概率算法(如第7章中遇到的算法)的随机性；也可参见第20章。另一个重要的新概念是模拟。密码学中的一个自然的问题是，如何说明"通过观察密文消息的各个参与方的行为，攻击者无法了解该密文消息的任何信息"。密码学家给出的答案是，证明"攻击者的任意观察结果都可以在不接触密文消息的条件下被模拟出来"。该方法可以概括为9.4节的零知识证明，并且被许多其他密码学应用所采用。

作为本章的开始，9.1节给出香农定义的完全保密机制，并指出其局限性。这些局限性引出了新的加密机制，即在多项式时间攻击下仍然安全的加密机制。9.2节将利用伪随机数产生器来构造这样的密码机制；9.2.3节还将说明在表面上很弱的假设条件下如何构造伪随机数产生器。9.4节介绍零知识证明这一精妙绝伦的概念，它在密码学和复杂性等方面有许多深入的应用。最后，9.5节将勾勒出如何用这些概念在各种场合下确保安全。密码学是一个巨大的领域，本章涉及的内容仅是它的一个很小的侧面；章节注记中给出了一些优秀的参考书供大家进一步阅读时选择。密码学还与本书研究的其他概念紧密相关，这些概念包括平均复杂性、难度放大和去随机化，参见第18章、第19章和第20章。

9.1　完全保密及其局限性

173　密码学的基本任务是加密。本节从较高的层面刻画这一任务并讨论"安全加密"的含义。我们介绍被称作一次一密的加密方法。一次一密加密方法已被证明是安全加密，同时它也被证明具有严重的局限性。

基本的加密过程如图9-1所示。爱丽丝想将明文 x 用秘密消息的形式发送给波比。但二人之间的通信信道正在被敌方伊蔚窃听。因此，爱丽丝将明文 x 用加密算法 E 加密成密文 y 之后，再将 y 发送给波比。这样，伊蔚很难或者根本不可能将密文 y 解码成明文 x，但波比却可以用解密算法 D 轻易地将密文 y 解码成明文 x。

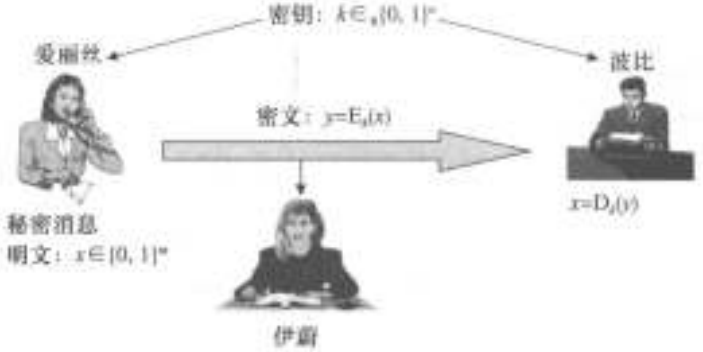


图9-1　在私钥密码中，爱丽丝和波比拥有随机选定的公共密钥 k。爱丽丝要将明文 x 发送给波比，先将明文 x 用密钥 k 和加密函数 $E(\cdot)$ 加密成密文 $y=E_k(x)$，再将密文 y 发送给波比。波比将密钥 k 和密文 y 作为解密算法 D 的输入来解密 x

当然，波比收到的消息和伊蔚窃听到的消息是一样的。因此，为了使波比完成伊蔚无法完成的解码任务，波比必须知道一些伊蔚不知道的信息。在私钥密码体系中，假设爱丽

丝和波比知晓随机选定的一个秘密的字符串 k(称为密钥)。当然，这要求爱丽丝和波比在通信前经过协商，对选用的密钥达成一致意见。

因此，加密方案由两个算法 E，D 构成，每个算法的输入都包含一个密钥和一个消息，并且对于任意密钥 k 和明文 x 总有

$$D_k(E_k(x)) = x \tag{9.1}$$

其中输入的密钥记为算法的下标。

条件(9.1)并未提及加密方案的安全性，并且"直接输出明文"这种平凡的"加密"方案也满足该条件。事实证明，定义加密方案的"安全性"是非常微妙的。首先，可以尝试将加密方案的安全性定义为"伊蔚无法用 $E_k(x)$ 来计算 x"。这种定义不行，因为它无法排除"伊蔚可以设法用 $E_k(x)$ 来计算得到 x 的部分信息"这种可能性。例如，若伊蔚知道明文不是"buy"就是"sell"，那么即使她无法完全恢复原消息也没有关系，她仅需知道消息的第 1 个字母就行了。为了使伊蔚无法根据密文来了解明文的任何信息，香农如下定义了私钥密码体系的安全性。

定义 9.1（完全保密） 设(E，D)是一个满足条件(9.1)的加密方案，它用于加密长度为 m 的消息并且密钥长度为 n。如果对于任意两个消息 x，$x' \in \{0,1\}^m$，$E_{U_n}(x)$ 和 174
$E_{U_n}(x')$ 都服从相同的分布[⊖]，则称(E，D)是完全保密的。

在完全保密的加密方案中，无论明文是什么，伊蔚窃听到的密文都将服从相同的分布，因此她绝对无法从密文中了解到明文的任何信息。这么强的条件似乎不可能被满足，然而，事实上存在一个非常简单的完全保密的加密方案。在一次一密方案中，要加密消息 $x \in \{0,1\}^n$，先随机选定密钥 $k \in_R \{0,1\}^n$，然后通过发送 $x \oplus k$ 来实现对 x 的加密，其中⊕表示按位异或(也就是模 2 的向量加法)。消息接收者将 $y = x \oplus k$ 与 k 再次按位异或就可以恢复出原消息 x。不难看到，无论被加密的明文是什么，密文都服从均匀分布。因此，一次一密方案是完全保密的方案(习题 9.1)。

当然，正如其名字所表明的含义，"一次一密"的方案的一次性密钥绝不能用于加密第二个消息。否则的话，如果消息 x 和 x' 用同一个密钥 k 加密，则伊蔚将窃听到 $k \oplus x$ 和 $k \oplus x'$，这样，她就能够计算 $(k \oplus x) \oplus (k \oplus x') = x \oplus x'$，这将暴露出原消息的一些非平凡的信息。事实上，还可以证明：在完全保密的任意加密方案中，密钥的长度不能短于消息的长度(参见习题 9.2)。

9.2 计算安全、单向函数和伪随机数产生器

尽管一次一密加密方案确实是完全保密的，但它完全不适用于现在的各种应用，这是因为，这些应用通常要交换上兆或上 G 字节的信息量；而前面的讨论表明，完全保密性要求私钥和消息本身一样长，用户之间交换这么长的密钥显然是不安全的。理想情况下，我们希望用户间的共享密钥应该适当地短，比如仅有几百个比特。显然，要实现这一目标，我们必须以某种方式放宽完全保密这一条件。正如引言中所指出的那样，我们的主要想法是：设计加密方案使得它仅相对于高效的窃听者(如多项式时间的窃听者)而言是安全的。然而，下面的引理表明，如果 **P=NP**，则即使对窃听者施加了上述限制条件，要确保完全保密的加密方案具有短密钥也是不可能的。因此，想要沿着我们的思路来设计完全加密方

⊖ 注意，U_n 表示 $\{0,1\}^n$ 上的均匀分布。

案，必须假定 **P≠NP**。事实上，我们的假设比 **P≠NP** 还强，具体地讲，我们将假设存在单向函数。将假设条件削弱到恰能证明加密方案的安全性(最理想的情况是削弱为**P≠NP**)仍是一个重要的研究问题。

引理 9.2 假设 **P=NP**。如果(E，D)是一个多项式时间可计算的加密方案，它满足(9.1)式并且密钥比消息短，则存在一个多项式时间算法 A 使得：对于任意的输入长度 m，均存在两个消息 x_0，$x_1\in\{0,1\}^m$ 满足

$$\Pr_{\substack{k\in_R\{0,1\}^n\\ k\in_R\{0,1\}^n}}[A(\mathrm{E}_k(x_b))=b]\geqslant 3/4 \tag{9.2}$$

175

其中 $n<m$ 是长为 m 的消息的密钥长度。

这样的算法将使得加密方案不安全。考虑 9.1 节中给出的"非'buy'即'sell'"的例子。安全的加密方案最起码应该确保伊蔚从密文中辨别出二选一的任意随机消息的概率不能超过 1/2。

引理 9.2 的证明：设(E，D)是满足引理要求的一个加密方案，它的消息长度为 m 并且密钥长度为 $n<m$。令 $S\subseteq\{0,1\}^*$ 表示 $\mathrm{E}_{U_n}(0^m)$ 的支持域，亦即 $y\in S$ 当且仅当 $y=\mathrm{E}_k(0^m)$ 对某个密钥 k 成立。因此，如果 **P=NP**，则 S 的成员资格能够被高效地验证。算法 A 可以简单地如下操作：在输入 y 上，如果 $y\in S$，则输出 0，否则输出 1。我们断言，对于 $x_0=0^m$，必存在 $x_1\in\{0,1\}^m$ 使得(9.2)式成立。

事实上，对于任意消息 x，令 D_x 表示分布 $\mathrm{E}_{U_n}(x)$。根据算法 A 的定义和 $x_0=0^m$ 这一事实，我们有 $\Pr[A(D_{x_0})=0]=1$。又由于

$$\Pr_{\substack{k\in_R\{0,1\}^n\\ b\in_R\{0,1\}}}[A(\mathrm{E}_k(x_b))=b]=\frac{1}{2}\Pr[A(Dx_0)=0]+\frac{1}{2}\Pr[A(Dx_1)=1]$$

$$=\frac{1}{2}+\frac{1}{2}\Pr[A(Dx_1)=1]$$

故只需证明，存在 $x_1\in\{0,1\}^m$ 使得 $\Pr[A(D_{x_1})=1]\geqslant 1/2$。换句话说，仅需证明 $\Pr[D_{x_1}\in S]\leqslant 1/2$ 对某个 $x_1\in\{0,1\}^m$ 成立。

若不然，假设 $\Pr[D_x\in S]>1/2$ 对任意 $x\in\{0,1\}^m$ 成立。如下定义 $S(x,k)$：如果 $\mathrm{E}_k(x)\in S$，则定义 $S(x,k)=1$；否则，定义 $S(x,k)=0$。将 $S(x,k)$ 的数学期望记为 $T=\mathrm{E}_{x\in_R\{0,1\}^m,k\in_R\{0,1\}^n}[S(x,k)]$。于是，在假设条件下有 $T>1/2$。但另一方面，还可以注意到，S 的大小 $\leqslant 2^n$，并且在任意固定的 k 上 $x\mapsto\mathrm{E}_k(x)$ 都是一个一一映射，进而 $x\mapsto\mathrm{E}_k(x)$ 至多将 $2^n\leqslant 2^m/2$ 个 x 映射到 S 中。基于上述事实，如果顺序地计算数学期望，我们得到

$$T=\mathrm{E}_{k\in_R\{0,1\}^n}[\mathrm{E}_{x\in_R\{0,1\}^m}[S(x,k)]]\leqslant 1/2$$

由此得到矛盾。 ■

在继续讨论之前，我们先引入一个简单的定义，它可以极大地简化本章中后续的记号。

定义 9.3（可忽略函数） 如果函数 $\varepsilon:\mathbf{N}\to[0,1]$ 满足 $\varepsilon(n)=n^{-\omega(1)}$，亦即 $\varepsilon(n)<n^{-c}$ 对任意 c 和充分大的 n 均成立，则称 ε 是可忽略函数。

由于可忽略函数在自变量增大时迅速趋于 0，因此如果随机事件发生的概率可以表示为一个可忽略函数，则这种随机事件可以在大多数实际应用和理论研究中被忽略。

9.2.1 单向函数：定义和实例

前面的讨论表明，证明加密方案的安全性必须在复杂性理论的猜想下进行。下面要引入的对象不仅对加密方案的安全性非常有用，而且对密码学的其他方面也非常有用。这个对象就是单向函数，亦即函数本身易于计算但其逆函数在多项式时间内却难以计算的函数。 176

定义 9.4（单向函数） 多项式时间可计算的函数 $f:\{0,1\}^*\to\{0,1\}^*$ 称为单向函数，如果对任意多项式时间的概率算法 A 均存在可忽略函数 $\varepsilon:\mathbf{N}\to[0,1]$ 使得对任意 n 均有

$$\Pr_{\substack{x\in_R\{0,1\}^n\\ y=f(x)}}[A(y)=x'\text{ s.t. }f(x')=y]<\varepsilon(n)$$

猜想 9.5 存在单向函数。

习题 9.5 要求证明猜想 9.5 蕴含 $\mathbf{P}\neq\mathbf{NP}$。大多数研究者都相信猜想 9.5 是正确的，这是由于有几个函数迄今为止仍未找到逆函数。下面介绍几个这样的函数。

乘法。简单的乘法很难求逆。也就是说，乘法函数将其输入 $x\in\{0,1\}^*$ 视为两个 $n/2$ 位的整数 A，B，并输出 $A\cdot B$；人们相信乘法函数是单向函数。乘法函数的逆函数也称为整数分解问题。当然，整数 N 的分解可以简单地通过至多 N 次除法（甚至 $\sqrt{N}$ 次除法）来实现。但是 N 的二进制表示只有 $\log N$ 个位，因此上述平凡算法实际上是一个指数时间算法。迄今为止，整数分解问题仍未找到多项式时间算法（亦即 $\mathrm{poly}(\log N)$ 时间的算法），最快的整数分解算法的运行时间为 $2^{(c\log N)^{1/3}(\log\log N)^{2/3}}$ [LLM90]。㊀

基于整数分解来实现单向函数，更标准的做法是将输入 $x\in\{0,1\}^n$ 视为产生 $n^{1/3}$ 位的两个随机素数 P，Q 的随机速度。也就是说，产生随机数 P，Q 并用第 7 章的素性测试算法测试它们是否为素数。如果 P，Q 是素数，则输出 $P\cdot Q$。

整数分解问题一直被数学家关注。事实上，在计算机被发明之前的 2000 年里，数学家一直致力于寻找高效的整数分解算法。这些努力的失败促使人们猜想"这种算法根本不存在"，继而整数乘法是单向函数。显然，上述猜想强于 $\mathbf{P}\neq\mathbf{NP}$ 这一猜想，也强于单向函数的存在性猜想。

RSA 和拉宾函数。（本例的讨论需要一些数论知识，请读者快速浏览 A.3 节。）RSA 函数㊁是单向函数的另一个公认的候选函数。假设在任意输入长度 n 上，能以某种方式产生一个 n 位复合整数 N，并且存在整数 e 与 $\varphi(N)=|\mathbf{Z}_N^*|$ 互素，其中 $\mathbf{Z}_N^*$ 是与 N 互素的所有整数构成的乘法群。典型地，N 通过两个 $n/2$ 位的素数的乘积来产生，而 e 通常简单地设置为 3。函数 $\mathrm{RSA}_{N,e}$ 将输入视为 $\mathbf{Z}_N^*$ 中的整数 X 并输出 $X^e \bmod(N)$。㊂可以证明，由于 e 与 $\varphi(N)$ 互素，因此函数 $\mathrm{RSA}_{N,e}$ 是 $\mathbf{Z}_N^*$ 上的一一映射。 177

拉宾函数是单向函数的另一个候选者，它也与 $\mathbf{Z}_N^*$ 相关。给定整数 N，它是满足 P，

㊀ 如果 A，B 是随机选取的，则不难找出 $A\cdot B$ 的一个素因数，这是因为 $A\cdot B$ 存在小素因数的概率很高。但是，找出 $A\cdot B$ 的所有素因数或者将 $A\cdot B$ 表示为两个不超过 $2^{n/2}$ 的整数的乘积等价于（时间复杂度相差一个多项式因子）分解两个随机素数的乘积。

㊁ RSA 是该函数的三位发明人（Rivest，Shamir，Adleman）名字的首字母，参见本章注记。

㊂ 将输入对 N 取模就可以将输入整数映射到 $\mathbf{Z}_N^*$。相对于输入的随机选取而言，转换后的值与 N 不互素的概率可以忽略。

$Q=1 \pmod 4$的两个奇素数 P，Q 的乘积，拉宾函数将 $X \in QR_N$ 映射为 $X^2 \pmod N$，其中 QR_N 是模 N 的所有平方剩余构成的集合(对于元素 $X \in \mathbf{Z}_N^*$，如果 $X=W^2 \pmod N$ 对某个 $W \in \mathbf{Z}_N^*$ 成立，则称 X 是模 N 的平方剩余)。同样，可以证明，拉宾函数是 QR_N 上的一一映射。

尽管人们相信 RSA 函数和拉宾函数均难以求逆，但是，如果已知 N 的因数分解，则这两个函数却很容易求逆。对于 RSA 函数，因数分解可以用来计算 $\varphi(N)$ 和 $d=e^{-1} \pmod{\varphi(N)}$。不难证明，函数 $Y^d \pmod N$ 是函数 $X^e \pmod N$ 的逆函数。对于拉宾函数，如果已知 N 的素因数分解，则可以用中国剩余定理将求模 N 的平方根转换为求 N 的素因数模的平方根，后者可以在多项式时间内完成。由于人们相信 RSA 函数和拉宾函数都是很难求逆的函数，但在了解一定信息(N 的因数分解)的条件下它们却又是很容易求逆的，因此，二者也被称为陷门单向函数，它们对公钥密码而言是至关重要的。已经证明，求拉宾函数的逆函数在计算上等价于求 N 的因数分解(习题 9.7)。对于 RSA 函数，这样的等价关系仍未建立。

勒维通用单向函数。存在具有如下奇特性质的函数 f_U：如果单向函数存在，且 f 是一个单向函数，则 f_U 也是一个单向函数。由于上述原因，函数 f_U 也称为通用单向函数。其定义如下：将输入视为一个位串序列 x_1，…，$x_{\log n}$，其中每个串的长度等于 $n/\log n$；输出 $M_1^t(x_1)$，…$M_{\log n}^t(x_{\log n})$，其中 M_i 指的是所有图灵机的某种规范化表示中的第 i 个图灵机。如果图灵机 M 在输入 x 上至多执行 t 个计算步骤，则 $M^t(x)$ 表示 M 在输入 x 上的输出；如果 M 在输入 x 上执行的计算步骤多于 t 个，则 $M^t(x)$ 表示全为 0 的位串 $0^{|x|}$。习题 9.6 要求读者证明 f_U 的通用性。

也存在与数论无关的候选单向函数。例如，在设计类似于 AES[DR02]的分组密码过程中(分组密码通过反复地把输入消息的各个位按某种机制进行混合来实现加密)，所使用的函数就与数论无关。

9.2.2　用单向函数实现加密

下面证明，单向函数可以用来设计安全的加密方案，使得密钥的长度远短于消息的长度。

定理 9.6　(用单向函数实现加密)　*假设单向函数存在。则对于任意 $c \in \mathbf{N}$，存在计算安全的加密方案在长度为 n^c 的消息上仅使用长度为 n 的密钥。*

当然，为使定理 9.6 的含义确切，需要定义"计算安全"这一术语。我们的思想仍然是，安全的加密方案不能向多项式时间窃听者泄露明文的任意的部分信息。由于该定义的微妙性，它的确切定义有些笨拙。因此，为了便于表述，我们此处只给出一个相对宽泛的
178 定义。在一个加密方案中，如果明文中任意随机选定的位不能被窃听者以显著高于 1/2 的概率猜出，则称该加密方案是"计算安全"的。也就是说，对于在长度为 m 的消息上使用长度为 n 的密钥的一个加密方案(E，D)是计算安全的，如果对于任意多项式时间的概率型算法 A，均存在一个可忽略函数 ε：$\mathbf{N} \to [0,1]$，使得

$$\Pr_{\substack{k \in_R \{0,1\}^n \\ x \in_R \{0,1\}^m}} [A(\mathrm{E}_k(x)) = (i,b) \text{ s.t. } x_i = b] \leqslant 1/2 + \varepsilon(n) \tag{9.3}$$

计算安全性的标准术语是语义安全性，我们将它的完整定义放到习题 9.9 中。完整定义比上述的定义强。习题 9.9 还证明了定理 9.6 在语义安全性下是成立的。

9.2.3 伪随机数产生器

回想一下9.1节中一次一密方案的思想，其唯一的局限性在于：密钥的长度必须等于所有待发送消息拼接在一起之后的总长度。注意，密钥是收、发双方共享的随机串。证明定理9.6所需要的主要思想是说明如何将长度为n的短密钥大幅度地拉伸成一个长度为m的串，并确保拉伸之后的串仍然是"足够随机的"，这样，将拉伸后的串用作一次性密钥，就可以确保加密过程对多项式时间窃听者是安全的。拉伸随机串采用的工具称为伪随机数产生器，它在密码学之外的其他领域也存在很多应用。

例9.7 什么样的串才是足够随机的呢？对于这个问题，科学家们早就进行过争论。柯尔莫戈洛夫(Kolmogorov)给出了这样的定义：长度为n的串是随机的，如果编码长度小于(不妨设)$0.99n$的任意图灵机在空白带上运行时无法输出该串。

从哲学和技术的意义上讲，上述定义是"正确的"(我们并不对此展开讨论)；但是，它对复杂性的研究却毫无用处，这是由于"用该定义来判断一个串是不是随机串"是不可判定的。

统计学家也曾尝试给出随机串的定义。在他们看来，判断一个串是不是随机串应该检查其中的节略性子串模式出现的次数是否"恰好"可以由统计规律计算得到(比如查看11100作为子串出现的次数是否可以由统计规律计算)。(完整的讨论请参阅[Knu73]。)事实证明，这样定义的随机串非常弱，根本不适用于密码学。这是由于可能存在能够通过所有统计检验的分布，但用它构造出来的一次一密方案的密码本仍然毫无安全性可言。 ◀

对于例9.7中提出的问题，密码学家给出的答案简单而令人满意。首先，他们不是试图去解答什么样的单个串"看上去是随机的"，而是关注所有串的分布。其次，他们不是像统计学家那样关注个别的测试算法，而是强调对任意多项式时间算法而言所有串的分布必须"显示出"均匀分布的特性。这种分布称为伪随机分布。随机串判定算法的输入是一个样本串，它既可以是基于均匀分布抽取的，也可以是根据其他未知的分布抽取的。判定算法根据样本串是否"表现"出随机性，相应地输出"是"或者"否"。例如，例9.7中基于统计方法的测试程序就是一个随机串判定算法。一个分布称为伪随机的，如果不管选用什么样的 179
多项式时间算法，算法在相关的两个分布上输出1的概率本质上是一样的。⊖

定义9.8 (安全的伪随机数产生器) 设$G:\{0,1\}^*\to\{0,1\}^*$是一个多项式时间可计算的函数，$\ell:\mathbf{N}\to\mathbf{N}$也是一个多项式时间可计算的函数，并且$\ell(n)>n$对任意$n$成立。我们称$G$是一个拉伸度为$\ell(n)$的伪随机数产生器，如果$|G(x)|=\ell(|x|)$对任意$x\in\{0,1\}^*$成立，并且对于任意多项式时间的概率型算法$A$，均存在一个可忽略函数$\varepsilon:\mathbf{N}\to[0,1]$使得

$$|\Pr[A(G(U_n))=1]-\Pr[A(U_{\ell(n)})]=1|<\varepsilon(n)$$

对任意$n\in\mathbf{N}$成立。

定理9.9 (用单向函数构造伪随机数产生器[HILL99]) 如果单向函数存在，则对任意$c\in\mathbf{N}$均存在拉伸度为$\ell(n)=n^c$的安全伪随机数产生器。

定理9.9的证明概要 定义9.8是说，多项式时间的任意敌手不可能区别长度为$\ell(n)$

⊖ 注意，这种定义不禁让人联想到"蒙眼测试"。例如，如果普通的顾客通过"蒙眼品尝"无法区分某种甜味剂和糖，则可以说这种甜味剂"像糖"。但是，伪随机分布的定义更严格一些，因为这种分布必须输过所有的随机串判定算法。如果将类似概念用到甜味剂的例子上，则要求每个人品尝甜味剂时都要像糖。

的真随机串和将 G 作用于长度为 n 的短随机串上产生的伪随机串。这意味着，定理 9.9 蕴含了定理 9.6。事实上，如果将一次一密方案调整为将拉伸度为 n' 的安全伪随机数产生器作用到长度为 n 的短密钥上来产生一个长度为 n' 的伪随机密钥，则多项式时间窃听者将无法分辨该密钥是真随机密钥还是伪随机密钥。为了看清这一点，假设有一个敌手 A 能够以显著高于 1/2 的概率预测明文中的一个位(这显然违背了(9.3)式定义的计算安全性)，那么，由于在真随机密钥下这种预测是不可能的(参见习题 9.3)，因此 A 可以用来区分真随机密钥和伪随机密钥，进而这与定义 9.8 刻画的安全伪随机数产生器矛盾。 ■

9.3　用单向置换构造伪随机数产生器

对于定理 9.9，我们仅证明单向函数是置换函数这种特殊情况。

引理 9.10　*假设存在单向函数 $f:\{0,1\}^*\to\{0,1\}^*$ 使得 f 是一一映射且 $|f(x)|=|x|$ 对任意 $x\in\{0,1\}^*$ 成立，则对任意 $c\in\mathbf{N}$ 均存在拉伸度为 $\ell(n)=n^c$ 的安全的伪随机数产生器。*

定理 9.9 比引理 9.10 更具一般性，但引理 9.10 的证明过程确实会反映出证明定理 9.9 的一些思想。并且，在这些思想中，混合参数和戈德赖希-勒维定理还具有独立的研究价值，它们在计算机科学的其他领域也有一些应用。

180

9.3.1　不可预测性蕴含伪随机性

要证明引理 9.10，用下面的性质来刻画伪随机数产生器将十分有益。在历史上，这才是伪随机数产生器的原始定义。它刻画的性质相对而言要弱一些，因此更容易通过显式构造来实现。姚期智证明了该性质与定义 9.8 之间的等价性，这在当时是一个重要的发现。

设 $G:\{0,1\}^*\to\{0,1\}^*$ 是一个拉伸度为 $\ell(n)$ 的多项式时间可计算的函数(亦即，$|G(x)|=\ell(|x|)$ 对任意 $x\in\{0,1\}^*$ 成立)。我们称 G 是不可预测的，如果对任意多项式时间概率型算法 B，均存在一个可忽略函数 $\varepsilon:\mathbf{N}\to[0,1]$ 使得

$$\Pr_{\substack{x\in_R\{0,1\}^n\\ y=G(x)\\ i\in_R[\ell(n)]}}[B(1^n,y_1,\cdots,y_{i-1})=y_i]\leqslant 1/2+\varepsilon(n) \tag{9.4}$$

也就是说，给定前 $i-1$ 个位之后(其中 i 是随机选定的索引编码)，任意多项式时间算法仍难以预测第 i 个位。

显然，如果 G 是伪随机数产生器，则它也是不可预测的。事实上，如果 $y_1,\cdots,y_{\ell(n)}$ 是均匀地随机选取的，则给定 $y_1,\cdots,y_{i-1}$ 之后仍然难以预测 y_i；进而，如果存在 G 的预测算法，则对于随机串 x 上的函数 $y=G(x)$，该预测算法也能区分 $U_{\ell(n)}$ 和 $G(U_n)$。出人意料的是，上述结论的逆命题也成立。

定理 9.11　*(不可预测性蕴含伪随机性[Yao82a])　设 $\ell:\mathbf{N}\to\mathbf{N}$ 是一个多项式时间可计算的函数，$G:\{0,1\}^*\to\{0,1\}^*$ 也是一个多项式时间可计算的函数，并且 $|G(x)|=\ell(|x|)$ 对任意 $x\in\{0,1\}^*$ 成立。如果 G 是不可预测的，则 G 也是一个安全的伪随机数产生器。而且，对于任意多项式时间的概率型算法 A，存在多项式时间概率型算法 B，使得对任意 $n\in\mathbf{N}$ 和 $\varepsilon>0$ 均有：如果 $\Pr[A(G(U_n))=1]-\Pr[A(U_{\ell(n)})=1]\geqslant\varepsilon$，则*

$$\Pr_{\substack{x\in_R\{0,1\}^n\\ y=G(x)\\ i\in_R[\ell(n)]}}[B(1^n,y_1,\cdots,y_{i-1})=y_i]\geqslant 1/2+\varepsilon/\ell(n)$$

证明 首先注意到，定理的主要结论蕴含在"而且"部分中。否则，假设 G 不是一个安全的伪随机数产生器，则存在算法 A 和常数 c 使得

$$|\Pr[A(G(U_n))=1]-\Pr[A(U_{\ell(n)})=1]|\geqslant n^{-c} \tag{9.5}$$

在无穷个 n 上成立。这样，必要时将算法 A（只含一个二进制位）的输出结果取反（亦即，用算法 $1-A$ 替换 A），就可以确保(9.5)式去掉绝对值符号之后仍然在无穷多个 n 上成立。在这些 n 上，我们就得到概率为 $1/2+n^{-c}/\ell(n)$ 的一个预测算法 B。这与 G 的不可预测性矛盾。 181

下面往证"而且"部分。令 A 是一个多项式时间的概率型算法，假设 A 在随机地取自分布 $G(U_n)$ 的输入上输出 1 的概率大于它在随机地取自分布 $U_{\ell(n)}$ 的输入上输出 1 的概率。我们的预测算法 B 简单地如下工作：在输入 1^n，$i\in[\ell(n)]$，y_1，…，y_{i-1} 上，算法 B 随机独立地选取 z_i，…，$z_{\ell(n)}$ 并计算 $a=A(y_1,\cdots,y_{i-1},z_i,\cdots,z_{\ell(n)})$。如果 $a=1$，则 B 推测第 i 个位应该取 z_i，于是输出 z_i；否则，算法 B 输出 $1-z_i$。

令 $n\in\mathbf{N}$ 且 $\ell=\ell(n)$。假设 $\Pr[A(G(U_n))=1]-\Pr[A(U_{\ell(n)})=1]\geqslant\varepsilon$，我们往证

$$\Pr_{\substack{x\in_R\{0,1\}^n\\ y=G(x)\\ i\in_R[\ell]}}[B(1^n,y_1,\cdots,y_{i-1})=y_i]\geqslant 1/2+\varepsilon/\ell \tag{9.6}$$

为了分析预测算法 B 的性能，我们定义 $\{0,1\}^\ell$ 上的 ℓ 个分布 $\mathcal{D}_0$，…，$\mathcal{D}_\ell$。（这种技术称为混合参数。）对每个 i，分布 $\mathcal{D}_i$ 如下获得：取 $x\in_R\{0,1\}^n$，记 $y=G(x)$，输出 y_1，…，y_i，z_{i+1}，…，z_ℓ，其中 z_{i+1}…，z_ℓ 随机独立地取自 $\{0,1\}$。注意，$\mathcal{D}_0=U_\ell$ 且 $\mathcal{D}_\ell=G(U_n)$。对任意 $i\in\{0,\cdots,\ell\}$，定义 $p_i=\Pr[A(\mathcal{D}_i)=1]$。于是，$p_\ell-p_0\geqslant\varepsilon$。因而，记

$$p_\ell-p_0=(p_\ell-p_{\ell-1})+(p_{\ell-1}-p_{\ell-2})+\cdots+(p_1-p_0)$$

我们得到 $\sum_{i=1}^{\ell}(p_i-p_{i-1})\geqslant\varepsilon$。换句话说，我们有 $\mathrm{E}_{i\in[\ell]}(p_i-p_{i-1})\geqslant\varepsilon/\ell$。为了证明(9.6)式，我们往证在任意 i 上均有

$$\Pr_{\substack{x\in_R\{0,1\}^n\\ y=G(x)}}[B(1^n,y_1,\cdots,y_{i-1})=y_i]\geqslant 1/2+(p_i-p_{i-1})$$

注意算法 B 的运行过程。它先为 y_i 猜测一个 z_i，然后调用算法 A 计算 a；若 $a=1$ 则输出 z_i，否则输出 $1-z_i$。因此，当"$a=1$ 且 $y_i=z_i$"或者"$a\neq1$ 且 $y_i=1-z_i$"时，算法 B 对 y_i 的预测均是正确的。这意味着，算法 B 正确预测 y_i 这一随机事件的概率为

$$\frac{1}{2}\Pr[a=1\,|\,z_i=y_i]+\frac{1}{2}(1-\Pr[a=1\,|\,z_i=1-y_i] \tag{9.7}$$

此外，不难验证，在 $z_i=y_i$ 的条件下，算法 B 调用算法 A 产生的分布恰好是 $\mathcal{D}_i$，这意味着 $\Pr[a=1\,|\,z_i=y_i]=p_i$。另一方面，在不考虑 z_i 的条件下，算法 B 调用算法 A 产生的分布恰好是 $\mathcal{D}_{i-1}$。因此，

$$\begin{aligned}p_{i-1}&=\Pr[a=1]=\frac{1}{2}\Pr[a=1\,|\,z_i=y_i]+\frac{1}{2}\Pr[a=1\,|\,z_i=1-y_i]\\&=\frac{1}{2}p_i+\frac{1}{2}\Pr[a=1\,|\,z_i=1-y_i]\end{aligned}$$

用 p_i 和 p_{i-1} 替换(9.7)式中相应的表达式，我们得到算法 B 成功预测 y_i 的概率为 $1/2+p_i-p_{i-1}$。 ■

9.3.2 引理 9.10 的证明：戈德赖希-勒维定理

设 f 是一个单向置换。为了证明引理 9.10，需要用 f 构造出拉伸度 $\ell(n)$ 为任意多项

式的伪随机数产生器。事实证明，构造过程的关键步骤是构造一个拉伸度为 $\ell(n)=n+1$ 的伪随机数产生器(亦即，将长度为 n 的随机串拉伸一个位的伪随机数产生器)。该步骤可
182 以用下面的定理来完成。

定理 9.12 (戈德赖希-勒维定理[GL89])　假设函数 $f:\{0,1\}^*\to\{0,1\}^*$ 是一个单向函数，它是一个一一映射且 $|f(x)|=|x|$ 对任意 $x\in\{0,1\}^*$ 成立。则对于任意多项式时间的概率型算法 A，均存在一个可忽略函数 $\varepsilon:\mathbf{N}\to[0,1]$ 使得

$$\Pr_{x,r\in_R\{0,1\}^n}[A(f(x),r)=x\odot r]\leqslant 1/2+\varepsilon(n)$$

其中 $x\odot r$ 定义为 $\sum_{i=1}^{n}x_ir_i(\mathrm{mod}\ 2)$。

定理 9.12 表明，函数 $G(x,r)=f(x),r,x\odot r$ 是一个拉伸一位的安全伪随机数产生器(它将长度为 $2n$ 的串拉伸为长度为 $2n+1$ 的串)。事实上，若不然，则由定理 9.11 可知，函数 $G(x,r)$ 将存在预测算法 B。由于 f 是 $\{0,1\}^*$ 上的一个置换，因此 $G(U_{2n})$ 的前 $2n$ 个位完全是随机独立的，因此每个位均不能由它前面的所有位以高于 $1/2$ 的概率来预测。这意味着，$G(x,r)$ 的预测算法 B 用前 $2n$ 个位来预测第 $2n+1$ 个位的概率必然显著地大于 $1/2$，而这恰好违背了定理 9.12。

定理 9.12 的证明　假设定理不成立，则存在一个多项式时间的概率型算法 A 违背了定理 9.12 的结论。我们构造一个多项式时间的概率型算法 B 使得它恰好计算置换 f 的逆函数，这将与 f 是单向函数矛盾。具体地讲，我们往证：若对某个 n 有

$$\Pr_{x,r\in_R\{0,1\}^n}[A(f(x),r)=x\odot r]\geqslant 1/2+\varepsilon \tag{9.8}$$

则 B 将在 $O(n^3/\varepsilon^2)$ 时间内以至少为 $\Omega(\varepsilon)$ 的概率求得单向函数 f 在所有长度为 n 的输入上的逆。这就是说，如果 A 的成功概率大于 $1/2+n^{-c}$ 对常数 c 和无穷多个 n 成立，则算法 B 的运行时间是多项式时间并且在无穷多个 n 上至少以概率 $\Omega(n^{-c})$ 计算单向函数 f 的逆。

令 n,ε 是满足(9.8)式的参数。则简单地运用均值论证法可以证明：在至少 $\varepsilon/2$ 比例的 x 上，相对于 r 的随机选择而言，$A(f(x),r)=x\odot r$ 的概率至少为 $1/2+\varepsilon/2$。我们称这样的 x 为良性的并构造算法 B 使得它在每个良性 x 上均能高概率地求得 $f(x)$ 的逆。

我们的任务可以重述为：给定一个"黑盒"，它至少能在 $1/2+\varepsilon/2$ 比例的 r 上计算未知线性函数 $x\mapsto x\odot r$，我们要给出一个算法使得它能根据函数值来重构 x 并且其时间复杂度是 $|x|$ 和 $1/\varepsilon$ 的多项式。(特别指出，这种重述形式与第 8 章引入的程序验证的思想存在联系，后面章节注记中将讨论这种联系。)

作为预备工作，注意，如果相对于 r 的所有随机选择而言 $\Pr_r[A(f(x),r)=x\odot r]=1$，则很容易根据 $f(x)$ 来重构 x。事实上，只需运行 $A(f(x),e^1),\cdots,A(f(x),e^n)$，其中 e^i 是第 i 个坐标等于 1 而其他坐标等于 0 的位串。由于 $x\odot e^i$ 显然等于 x 的第 i 个坐标，因而上述 n 次调用恰好完整地恢复了 x。当然，如果 $\Pr_r[A(f(x),r)=x\odot r]$ 小于 1，则上
183 述重构过程行不通。下面，我们先给出一个稍简单的重构算法使得它在概率为 0.9 时能够重构 x；然后，再扩展这个稍简单的算法得到更具一般性的算法。

在成功概率为 0.9 的情况下实现重构

现在，假设 $\Pr_r[A(f(x),r)=x\odot r]\geqslant 0.9$ 对 $\Omega(\varepsilon)$ 比例的 x 成立。对于这样的 x，我们不能指望 $A(f(x),e^i)=x\odot e^i$，因为 $e^1,\cdots,e^n$ 可能恰好出现在 A 给出错误答案的那 $2^n/10$ 个 r 中。同样，存在一个简单的绕过上述问题的重构方案：如果取 $r\in_R\{0,1\}^n$，则 $r\oplus e^i$ 也

服从均匀分布。因此，由合并界限可知，算法 A 在 r 或 $r\oplus e^i$ 上给出错误答案的概率是

$$\Pr[A(f(x),r)\neq x\odot r \text{ 或 } A(f(x),r\oplus e^i)\neq x\odot(r\oplus e^i)]\leqslant 0.2$$

但是 $x\odot(r\oplus e^i)=(x\odot r)\oplus(x\odot e^i)$。这意味着，如果我们随机选取 r，并计算 $z=A(f(x),r)$ 和 $z'=A(f(x),r\oplus e^i)$，则 $z\oplus z'$ 等于 x 的第 i 个位的概率至少等于 0.8。多次重复上述操作，取出现次数最多的值，可以将恢复 x 的第 i 个位的概率放大到 $1-1/(10n)$。对每个 i 重复上述过程即可重构得到 x。具体地讲，我们有如下算法。

算法 *B*

1. 随机独立地从 $\{0,1\}^n$ 中选取 $r^1,\cdots,r^m$（后面将指出 m 的值）。
2. 对任意 $i\in[n]$：
   - 计算 $z_1=A(f(x),r^1)$，$z_1'=A(f(x),r^1\oplus e^i)$，$\cdots$，$z_m=A(f(x),r^m)$，$z_m'=A(f(x),r^m\oplus e^i)$；
   - 输出 $\{z_j\oplus z_j'\}_{j\in[m]}$ 中出现频数最高的值作为 x_i 的猜测值。

我们断言，如果 $m=200n$，则对任意 $i\in[n]$，频数最高的值是 x_i 的正确值的概率至少为 $1-1/(10n)$（进而，算法 B 成功地重构 x 的每个位的概率至少为 0.9）。为了证明该断言，我们引入随机变量 Z_j。如果 $A(f(x),r^j)=x\odot r^j$ 和 $A(f(x),r^j\oplus e^i)=x\odot(r^j\oplus e^i)$ 都成立，则定义 $Z_j=1$；否则，定义 $Z_j=0$。注意，随机变量 $Z_1,\cdots,Z_m$ 是相互独立的，并且由前面的讨论可知 $\mathrm{E}[Z_j]\geqslant 0.8$ 对任意 j 成立。只需证明"有超过 $m/2$ 个 Z_j 等于 1"的概率至少为 $1-1/(10n)$。换句话说，令 $Z=Z_1+\cdots+Z_m$，则仅需证明 $\Pr[Z\leqslant m/2]\leqslant 1/(10n)$。但是，由于 $\mathrm{E}[Z]=\sum_j \mathrm{E}(Z_j)\geqslant 0.8m$，所以仅需给出 $\Pr[|Z-\mathrm{E}(Z)|\geqslant 0.3m]$ 的上界。根据切比雪夫不等式（引理 A.12）可知[⊖]，

$$\Pr[|Z-\mathrm{E}[Z]|\geqslant k\sqrt{\mathrm{Var}(Z)}]\leqslant 1/k^2$$

由于 Z_j 是相互独立的 0/1 随机变量，因此 $\mathrm{Var}(Z)=\sum_{j=1}^{m}\mathrm{Var}(Z_j)$ 并且 $\mathrm{Var}(Z_j)\leqslant 1$ 对任意 j 成立，这意味着

$$\Pr[|Z-\mathrm{E}[Z]|\geqslant 0.3m]\leqslant \frac{1}{(0.3\sqrt{m})^2}$$

由于我们选择 $m=200n$，故上式中的概率小于 $1/(10n)$。 184

在成功概率为 $1/2+\varepsilon/2$ 的情况下实现重构

前面的分析用到了一个不太现实的假设，亦即，在很多 x 上，相对于 r 的随机选择而言 $A(f(x),r)$ 是正确的概率至少为 0.9。不难看到，一旦上述概率降到 0.75 以下，则前面的分析将会失效，因为将合并界限应用在随机事件 $A(f(x),r)=x\odot r$ 和 $A(f(x),r\oplus e^i)=x\odot(r\oplus e^i)$ 上之后得不到任何有意义的信息。不幸的是，我们唯一可以保证的甚至仅为：如果 x 是良性的，则 $A(f(x),r)$ 是正确的概率至少为 $1/2+\varepsilon/2$（该概率可能会远小于 0.75）。将前面的证明过程扩展到一般情况的关键在于，如果两两独立地随机选取 $r^1,\cdots,r^m$ 而不是完全随机独立地选取，则前面的分析过程将仍然是可行的。事实上，在前面

⊖ 我们可以用切尔诺夫界给出更好的上界。但是，由切比雪夫不等式给出的界限更容易推广到一般情况，以便在成功概率更小的情况下实现重构。

的分析中，随机独立性仅仅用于论证随机变量 Z_1，…，Z_m 满足 $\mathrm{Var}\left(\sum_i Z_i\right) = \sum_i \mathrm{Var}(Z_i)$，但这一条件对两两独立的随机变量也成立（参见论断 A.13）。

下面，我们说明如何在"已知"每个 $x\odot r^j$ 的前提下两两独立地选取 r^1，…，r^m。虽然这似乎有些缘木求鱼的味道，因为 x 是未知的，但是，这一过程却仍然可以通过穷举性猜测来操作，下面很快将阐明这一点。令 k 是满足 $m\leqslant 2^k-1$ 的一个整数，并如下地进行操作：

1. 从 $\{0,1\}^n$ 中随机独立地选取 k 个串 s^1，…，s^k；

2. 对任意 $j\in[m]$，我们用某种规范的方式将 j 关联到唯一的非空集合 $T_j\subseteq[k]$ 并定义 $r^j=\sum_{i\in T_j} s^i \pmod 2$；也就是说，$r^j$ 是 s^1，…，s^k 中属于第 j 个集合的所有串按位异或之后得到的结果。

可以证明，串 r^1，…，r^m 是两两独立的（参见习题 8.4）；而且，$x\odot r^j=\sum_{i\in T_j} x\odot s^i$ 对任意 $x\in\{0,1\}^n$ 成立。这意味着，如果我们知道 $x\odot s^1$，…，$x\odot s^k$ 这 k 个值，则可以推算出 $x\odot r^1$，…，$x\odot r^m$ 这 m 个值。这就是穷举性猜测的由来。由于 $2^k=O(m)$，我们可以在多项式时间内猜测 $x\odot s^1$，…，$x\odot s^k$ 的所有可能的取值。这样就得到计算 $f(\cdot)$ 的逆函数的如下算法 B'。

算法 B'

输入： $y\in\{0,1\}^n$，其中 $y=f(x)$ 对某个未知的 x 成立。

假设 x 是"良性的"，继而 $\Pr_r[A(f(x),r)=x\odot r]\geqslant 1/2+\varepsilon/2$。我们并不关心 B' 在非良性的 x 上完成哪些操作。

操作： 令 $m=200n/\varepsilon^2$ 且 k 是满足 $m\leqslant 2^k-1$ 的最小整数。从 $\{0,1\}^n$ 中随机独立地选取 s^1，…，s^k，并像前文所述那样定义 r^1，…，r^m。然后，对每个 $w\in\{0,1\}^k$，完成如下操作：

- 在"$x\odot s^i=w_i$ 对任意 $i\in[k]$ 成立"的假设条件下调用算法 B；也就是说，对于每个 $i\in[n]$ 令 $z_i=\sum_{i\in T_j} w_i \pmod 2$，然后计算我们对 $x\odot r^j$ 的猜测值 z_j。类似地，令 $z_j'=A(y,r^j\oplus e^i)$，然后调用算法 B 计算我们对 $x\odot(r\oplus e^i)$ 的猜测值 z_j'。
- 同算法 B 中一样，对任意 $i\in[n]$，将 $\{z_j\oplus z_j'\}_{j\in[m]}$ 中出现频数最高的值作为 x_i 的猜测值。
- 测试我们猜测的 $x=x_1$，…，x_n 是否满足 $f(x)=y$。如果满足，则算法终止并输出 x。

185

对算法 B' 的分析过程几乎与对算法 B 的分析过程一样。在 2^k 次循环中，我们必然在某一次中会猜测出 $x\odot s^1$，…，$x\odot s^k$ 的正确值 w_1，…，w_k。注意，在这一次循环体执行过程中，对于任意 $i\in[n]$，算法 B' 正确地猜出 x_i 的值的概率至少为 $1-1/(10n)$。事实上，固定某个 $i\in[n]$，像分析算法 B 时一样引入随机变量 Z_1，…，Z_m；亦即，Z_j 是 0/1 随机变量，如果 $z_j=x\odot r^j$ 和 $z_j'=x\odot(r^j\oplus e^i)$ 都成立，则 $Z_j=1$。如果循环体执行过程中正确地猜测到 w_1，…，w_k，则 $z_j=x\odot r^j$ 肯定成立；于是，Z_j 的取值取决于第二个随机事件是否发生；注意，该事件发生的概率至少为 $1/2+\varepsilon/2$。因此，仅需证明：对 $m=100n/\varepsilon^2$，

如果 Z_1，…，Z_n 是两两独立的 0/1 随机变量且 $E[Z_i]\geqslant 1/2+\varepsilon/2$ 对任意 i 成立，则 $\Pr[\sum_i Z\leqslant n/2]\leqslant 1/(10n)$。该条件立刻可以由切比雪夫不等式得到。 ■

获得任意大的拉伸度

定理 9.12 得到了一个拉伸度为 $\ell(n)=n+1$ 的安全伪随机数产生器。但是，要完成定理 9.10 的证明(亦即，为了得到具有较短密钥的安全加密方案)，我们需要构造一个拉伸度为任意大的多项式的伪随机数产生器。这种伪随机数产生器由下述定理得到。

定理 9.13 *若 f 是一个单向置换且 $c\in\mathbf{N}$，则将 x，$r\in\{0,1\}^n$ 映射到 r，$f^1(x)\odot r$，$f^{-1}(x)\odot r$，…，$f^1(x)\odot r$ 的函数 G 是一个拉伸度为 $\ell(2n)=n+n^c$ 的安全伪随机数产生器，其中 $\ell=n^c$ 且 f^i 表示将函数 f 连续 i 次作用在输入上所得到的函数。*

证明：由姚期智定理(定理 9.11)，只需证明 G 输出的每个位均难以预测。若不然，假设存在多项式时间的概率型多项式时间图灵机 A 使得，当 x，$r\in\{0,1\}^n$ 和 $i\in\{1,2,\cdots,N\}$ 都是随机选取时(其中 $N=\ell=n^c$)有

$$\Pr[\text{在给定 } r,\ f^N(x)\odot r,\ f^{N-1}(x)\odot r,\ \cdots,\ f^{i+1}(x)\odot r \text{ 的条件下 } A \text{ 正确地预测 } f^i(x)\odot r]\geqslant\frac{1}{2}+\varepsilon$$

下面，我们来构造一个多项式时间的概率型算法 B 使得它在这种 n 上能由 $f(x)$，r 预测出 $x\odot r$ 的概率也至少为 $1/2+\varepsilon$。这就是说，如果 A 的成功概率是不可忽略的，则 B 将会违反定理 9.12。

算法 B 的输入是 r 和 y，其中 $y=f(x)$ 对某个 x 成立。算法 B 随机选取 $i\in\{1,2,\cdots,N\}$，然后计算 $f^{N-i}(y)$，…，$f(y)$，并输出 $a=A(r,f^{N-i-1}(y)\odot r,\cdots,f(y)\odot r,y\odot r)$。由于 f 是一个置换，因此“算法 B 输出的分布”恰好等同于“随机选取 $x'\in_R\{0,1\}^n$ 且令 $x=f^i(x')$ 后，A 预测 $f^i(x')\odot r$ 的分布”。又由于 A 以 $1/2+\varepsilon$ 的概率预测出 $f^i(x')\odot r$，故算法 B 也以 $1/2+\varepsilon$ 的概率预测出 $x\odot r$。 ■

9.4 零知识

我们通常认为，一个证明也就是提供证据来说明某个结论的正确性。因此，通过仔细阅读和验证某个结论的证明，我们除了相信结论正确之外，还将了解到更多的其他信息。这种现象是一定的吗？例如，假设你有某种办法来调度某航空公司的所有航班使得该公司能节省数百万美元的资金。你想向这家航空公司证明你确实有这样的调度方法，却不想将调度方法真正透露给他们(至少在你没有得到丰厚的报酬之前，你不想这么做)。这有可能吗？ 186

类似的情况出现在认证过程中。假设某家公司有一座涉密的建筑，只有特定的员工才能进入该建筑。严格的门禁可以如下实施。随机选定两个素数 P 和 Q，并将它们告知允许进入该建筑的员工，同时将 $N=P\cdot Q$ 告知在建筑外执勤的门卫。门卫只允许知道如何分解 N 的员工进入该建筑。员工们能否在不泄露因数分解的前提下向门卫证明他们确实知道因数分解呢？

事实证明，利用零知识证明这是可以做到的。零知识证明是一种概率型交互式证明系统，它与我们在第 8 章中遇到的交互式系统类似。但是，零知识证明系统除了要求具有完备性(证明者能让验证者高概率地接受正确结论)和可靠性(验证者高概率地拒绝错误结论)之外，还要求具有零知识性。粗略地讲，零知识性要求验证者除了从交互过程中了解结论的正确性之外，无法了解更多的其他信息。也就是说，零知识性要求：验证者在参与结论 x 的交互证明过程之后，她所了解的其他信息只能是她在交互过程之外自己通过计算得到

的信息。下面，我们为 **NP** 语言形式地定义零知识证明(当然，也可以为 **NP** 之外的语言定义零知识证明；但是，限制在 **NP** 语言上将零知识和零知识证明系统讨论清楚，就已经非常了不起了，而且也够用了。)

定义 9.14　(零知识证明)　设 L 是一个 **NP** 语言，M 是一个多项式时间图灵机，并且 $x\in L\Leftrightarrow\exists_{u\in\{0,1\}^{p(|x|)}}$ s.t. $M(x,u)=1$，其中 $p(\cdot)$ 是多项式。

两个多项式时间的交互式概率型算法 P，V 称为语言 L 的零知识证明，如果下列三个条件成立：

完备性：对于任意 $x\in L$ 和该事实的一个证明 u(亦即 $M(x,u)=1$)都有 $\Pr[\mathrm{out}_V\langle P(x,u),V(x)\rangle]\geqslant 2/3$，其中 $\langle P(x,u),V(x)\rangle$ 表示 P 和 V 之间的交互，P 以 x，u 为输入而 V 以 x 为输入，并且 $\mathrm{out}_V l$ 表示交互过程 l 结束之后 V 的输出。

可靠性：如果 $x\notin L$，则对任意策略 P^* 和输入 u 都有 $\Pr[\mathrm{out}_V\langle P^*(x,u),V(x)\rangle]\leqslant 1/3$。注意，策略 P^* 的运行时间不一定是多项式。

完全零知识性：对于多项式时间的任意交互式概率型策略 V^*，存在另一个(单独的)期望运行时间为多项式时间的概率型算法 S^*，使得对于任意 $x\in L$ 和该事实的一个证明 u 均有

$$\mathrm{out}_{V^*}\langle P(x,u),V^*(x)\rangle = S^*(x) \tag{9.9}$$

(这就是说，即使 S^* 不使用 x 的任何证明，上述等式两端表示的两个随机变量也将服从相同的分布。)算法 S^* 称为 V^* 的模仿者，因为它模拟了 V^* 与证明者交互过程产生的输出。

187

零知识性这一条件意味着，即使验证者不遵守协议而采用其他的任何策略 V^*，她也不能从交互过程了解到任何新信息。这是由于，她在交互过程中所能了解的信息与在公众知晓的输入 x 上单独运行算法 S^* 所能了解的信息相同。完全零知识性这一条件可以适当地放宽，比如仅要求(9.9)式的两个分布具有较小的统计距离(参见 A.26)或者仅要求这两个分布是计算不可分的(参见习题 9.17)。上述修改产生的概念分别是统计零知识和计算零知识，二者也都是密码学和复杂性理论中的核心概念。具有统计零知识证明系统的所有语言构成的类就是著名的 **SZK** 类，它有一些精巧绝伦的性质，人们相信它严格介于 **P** 和 **NP** 之间([Vad99]对此进行了精彩的综述)。人们已经证明([GMW86]，也可以参见[BCC86])，如果单向函数存在，则任意 **NP** 语言都将存在一个计算零知识证明；这一结论在密码协议的设计中发挥着重要作用(参见本章注记)。

利用模拟来展示安全性，这种思想也在密码学的许多方面发挥了关键作用。除零知识之外，模拟还用于定义加密方案的语义安全性(参见习题 9.9)、多方计算的安全性(9.5.4 节)以及很多其他的概念。在所有这些概念中，安全性均被定义为如下的条件：攻击者无法了解理想情况下她无法了解的任何信息，也无法完成理想情况下她无法完成的任何计算(例如，在加密方案中理想情况是攻击者甚至不知道密文；在零知识中理想情况是无需与证明者进行交互)。

例 9.15　我们给出图同构语言 GI 的一个零知识证明。语言 GI 属于 **NP**，并且"证明者将同构映射发送给验证者"这个平凡的证明满足完备性和可靠性。但是，该证明未必是一个零知识证明，因为我们没有多项式时间算法来计算给定的两个图之间的同构。

图同构的零知识证明

公共输入：两个 n 顶点图 G_0，G_1；具体地，假设两个图均被表示为邻接矩阵。

证明者的私有输入：一个置换 $\pi:[n]\to[n]$，它满足 $G_1=\pi(G_0)$，其中 $\pi(G)$ 表示将顶点 i 置换为 $\pi(i)$ 之后得到的图(或者等价地说，将 π 作用到 G 的邻接矩阵的所有行和所有

列上之后得到的图)。

证明者的第一个消息：证明者随机选择一个置换 $\pi_1:[n]\to[n]$，并将 $\pi_1(G_i)$ 的邻接矩阵发送给验证者。

验证者的消息：验证者随机选择 $b\in_R\{0,1\}$，并将 b 发送给证明者。

证明者的最后一个消息：如果 $b=1$，则证明者将 π_1 发送给验证者。如果 $b=0$，则证明者将 $\pi_1\circ\pi$(亦即将 i 映射为 $\pi_1(\pi(i))$的置换)发送给验证者。

验证者的检验：设 H 是验证者收到的第一个消息所表示的图，而 π 是验证者收到的最后一个消息表示的置换，则验证者接受"G_0同构于G_1"当且仅当 $H=\pi(G_b)$。 188

显然，如果证明者和验证者均遵守上述协议，则验证者接受的概率为1。下面考虑可靠性。我们断言，如果 G_0 和 G_1 不同构，则验证者至少以 $1/2$ 的概率拒绝(通过重复执行协议，该概率还可以进一步减小)。事实上，此时无论证明者采用何种策略，他在第一个消息中发送给验证者的图 H 不可能同时与 G_0 和 G_1 同构。因此，存在 $b\in\{0,1\}$ 使得 H 不同构于 G_b；而验证者选中这个 b 值的概率等于 $1/2$。这样，证明者找不到满足 $H=\pi(G_b)$ 的置换 π，进而导致验证者拒绝。

令 V^* 是验证者的一个策略。我们用如下的模仿者 S^* 来证明零知识性。在输入的两个图 G_0，G_1 上模仿者 S^* 随机选取 $b'\in_R\{0,1\}$ 和 $[n]$ 上的置换 π，并计算 $H=\pi(G_{b'})$；然后，模仿者将 H 交给验证者 V^*，并得到验证者的响应消息 $b\in\{0,1\}$。如果 $b=b'$，则 S^* 将 π 发送给验证者 V^*，然后输出 V^* 的输出结果。否则(若 $b\neq b'$)，则 S^* 从头再开始运行。

关键在于注意到，S^* 的第一个消息和证明者的第一个消息都随机地发送了一个同构于 G_0 或 G_1 的图，这个随机的图服从同一个分布。这也意味着，H 没有泄露关于 b' 的选择的任何信息，因此 $b=b'$ 的概率为 $1/2$。如果 $b=b'$ 确实发生，则 V^* 见到的消息 H 和 π 与他真正同证明者进行交互时见到的消息也服从相同的分布。由于 S^* 成功使得 $b=b'$ 成立的概率为 $1/2$，因此模仿者循环 k 次的概率为 2^{-k}，这意味着，模仿者的期望运行时间为 $T(n)\sum_{i=1}^{\infty}2^{-k}=O(T(n))$，其中 $T(n)$ 表示 V^* 的运行时间。因此，S^* 是一个期望时间为多项式的概率算法。㊀

9.5 应用

下面我们给出本章介绍的思想的一些应用。

9.5.1 伪随机函数及其应用

伪随机函数是伪随机数产生器的自然推广。当然，这不禁让我们联想到伪随机数产生器的定义，亦即在"蒙眼测试"中伪随机数看上去必须跟真随机数一样。不同之处在于，伪随机函数是真值表具有指数大小的函数。伪随机函数的(多项式时间的)识别算法只能在选定的一些输入上查验函数的取值。

也就是说，伪随机函数族中的每个伪随机函数都能够被高效地计算并且其表示形式具有多项式大小(因此肯定不是随机函数)。然而，相对于计算能力受限的观察者(亦即，他只能观察到函数在他任意选定的一些输入上的取值)而言，典型的伪随机函数与随机函数却是无法区分的。

㊀ 在第15章，我们将看到更严格的"扩型概率多项式时间"的概念(参见定义18.4)。证明过程中的模仿者满足这个更严格的概念。

定义 9.16 设$\{f_s\}_{s\in\{0,1\}^*}$是满足$f_s:\{0,1\}^{|s|}\to\{0,1\}^{|s|}$对任意$s\in\{0,1\}^*$成立的一族函数，而且存在多项式时间算法在给定$s\in\{0,1\}^*$和$x\in\{0,1\}^{|s|}$上计算$f_s(x)$。
189 我们称该函数族是伪随机的，如果对于多项式时间的任意概率型神谕图灵机A均存在可忽略函数$\varepsilon:\mathbf{N}\to[0,1]$使得

$$\left|\Pr_{s\in_R\{0,1\}^n}[A^{f_s(\cdot)}(1^n)=1]-\Pr_{F\in_R\mathcal{F}_n}[A^F(1^n)=1]\right|<\varepsilon(n)$$

对任意n成立，其中$\mathcal{F}_n$表示从$\{0,1\}^n$到$\{0,1\}^n$的所有函数构成的集合。

可以验证，如果$\{f_s\}$是一个伪随机函数族，则对于任意多项式$\ell(n)$，将$k\in\{0,1\}^n$映射为$f_k(1)$，…，$f_k(\ell(n))$的函数G是一个安全伪随机数产生器(当然，这要求将1，…，$\ell(n)$规范地表示为$\{0,1\}^n$中的串)。于是，伪随机函数的存在性蕴含了具有任意多项式拉伸度的安全伪随机数产生器的存在性。这一结论的逆命题也成立。

定理 9.17 ([GGM84]) 如果存在一个拉伸度为$\ell(n)=2n$的安全伪随机数产生器，则存在伪随机函数族。

证明 设G是定理条件中的安全伪随机数产生器，它将长度为n的随机串映射为长度为$2n$的伪随机串。对任意$x\in\{0,1\}^n$，令$G_0(x)$表示$G(x)$的前n个位而$G_1(x)$表示$G(x)$的后n个位。对任意$k\in\{0,1\}^n$，函数$f_k(x)$定义为

$$f_k(x)=G_{x_n}(G_{x_{n-1}}(\cdots(G_{x_1}(x))\cdots)) \tag{9.10}$$

其中自变量x取遍$\{0,1\}^n$中的每个元素。注意，$f_k(x)$的计算可以通过函数G的n次调用来实现，故$f_k(x)$的计算可以在多项式时间内完成。还可以用另一种视角来观察f_k。如图9-2所示，考察一棵深度为n的完全二叉树，将树根标记为k，将标签为y的结点的左、
190 右孩子分别标记为$G_0(y)$和$G_1(y)$，这样，$f_k(x)$表示该树中的第x个叶子节点上的标签。虽然将这棵树实际地画出来显然需要指数时间和指数空间，但正如(9.10)式所示，我们可以从树根出发沿一条长度为n的路径访问树的每个叶子结点，进而该叶子结点上的标签可以在多项式时间内被计算出来。

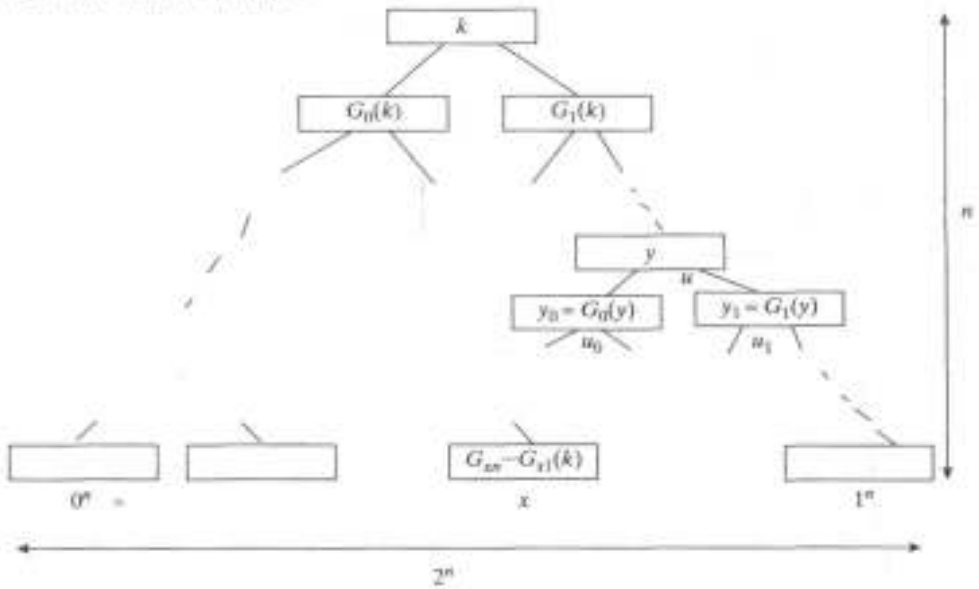


图 9-2　伪随机函数$f_k(x)$输出一棵深度为n的完全二叉树中第x个叶子结点上的标签，该树的树根被标记为k，每个标记为y的结点u的左孩子u_0和右孩子u_1被分别标记为$G_0(y)$和$G_1(y)$

为什么这个函数族是伪随机的呢？我们如下证明这一结论。我们证明，能在 ε 误差范围内区分 f_{U_n} 和真随机函数的 T 时间算法 A 可以转换成一个能在 $\varepsilon/(nT)$ 误差范围内区分 U_{2n} 和 $G(U_n)$ 的 $\mathrm{poly}(n)T$ 算法 B。

不失一般性，假设 A 恰好向神喻咨询 T 个问题（为确保这一假设，可以添加一些多余的问题）。下面，我们实现一个适用于 f_{U_n} 的神喻 $\mathcal{O}$，让它根据实际需要对深度为 n 的完全二叉树的顶点添加标签。初始时，只有树根被标记为随机串 k。每当算法 A 让神喻对标签为 y 的结点 u 的两个孩子 u_0，u_1 进行标记时，神喻就以 y 为输入调用 G 得到 $y_0=G_0(y)$ 和 $y_1=G_1(y)$，然后它将结点 u_0 和 u_1 分别标记为 y_0 和 y_1，最后删除结点 u 的标签 y。值得注意的是，一旦 u_0，u_1 被标记，结点 u 的标签就没有用了。根据 f_k 的定义可知，当算法向神喻咨询问题 x 时，神喻 $\mathcal{O}$ 将以第 x 个叶子结点的标签来作答。注意，G 至多被神喻 $\mathcal{O}$ 调用 Tn 次。必要时添加一些多余的调用，我们假设 G 恰好被神喻 $\mathcal{O}$ 调用 Tn 次。

现在，我们对任意 $i\in\{0,\cdots,Tn\}$ 运用混合参数方法，如下定义 $\mathcal{O}_i$：神喻 $\mathcal{O}_i$ 与 $\mathcal{O}$ 的工作过程相同，只是在前 i 次调用 G 确定标签为 y 的结点 u 的两个孩子 u_0 和 u_1 的标签 y_0 和 y_1 时，不再令 $y_0=G_0(y)$ 和 $y_1=G_1(y)$，而是随机独立地从 $\{0,1\}^n$ 中选取 y_0 和 y_1。注意，$\mathcal{O}_0$ 就是 f_{U_n} 的神喻 $\mathcal{O}$，而 $\mathcal{O}_{Tn}$ 是真随机函数的神喻。令 $p_i=\Pr[A^{\mathcal{O}_i}(1^n)=1]$。进而，同定理 9.11 的证明过程一样，我们可以假设 $p_{i+1}-p_i\geqslant\varepsilon$，进而可以证明 $\mathrm{E}_{i\in_R[Tn]}[p_i-p_{i-1}]\geqslant\varepsilon/(nT)$。我们定义区分 U_{2n} 和 $G(U_n)$ 的算法 B 如下工作：在输入 $y\in\{0,1\}^{2n}$ 上，随机选取 $i\in_R[Tn]$，然后用 $\mathcal{O}_{i-1}$ 作为神喻运行算法 A；亦即，在 G 的前 i 次调用时用随机选择的串作为结点的标签，而在其余调用中正常运行 G；最后算法 B 输出算法 A 得到的结果。不难验证，对于任意随机选取的 i，如果 y 服从分布 U_{2n}，则算法 B 输出结果的分布将与 $A^{\mathcal{O}_i}(1^n)$ 的分布相同；如果 y 服从分布 $G(U_n)$，则算法 B 输出结果的分布将与 $A^{\mathcal{O}_{i-1}}(1^n)$ 的分布相同。 ■

伪随机函数族可以视为一种变换方法，它将随机串 $k\in\{0,1\}^n$ 变换为一个具有指数大小的“显得很随机”的串，并给出这个串的一种隐式表示形式（亦即，f_k 的函数值列表）。伪随机函数是密码学中能力强大的基本方法。例如，在考虑单个消息的加密方案时，在实践中我们通常还希望用同一个密钥去加密许多其他的消息。随机函数将使得爱丽丝和波比能够共享一个“具有指数大小的一次性密码本”。也就是说，爱丽丝和波比能够将伪随机函数的下标 k 作为密钥进行共享；每当爱丽丝希望将消息 $x\in\{0,1\}^n$ 加密后发送给波比时，她可以先随机选取 $r\in_R\{0,1\}^n$，并将 $(r,f_k(r)\oplus x)$ 发送给波比。由于波比知晓密钥 k，故他可以解密得到 x。但是，敌方由于不知晓密钥，因此在他看来爱丽丝似乎发送了两个随机串（当然，这要求爱丽丝不能用同一个随机串 r 加密两个不同的消息，但这种情况发生的概率仅为指数大小）。伪随机函数还可以用于消息认证码。如果爱丽丝和波比 191
将伪随机函数的下标 k 作为密钥进行共享，则爱丽丝在将消息 x 发送给波比时可以将 $f_k(x)$ 添加在 x 之后。这样，波比可以验证所接收到的合法消息 (x,y) 满足 $y=f_k(x)$。敌方伊蔚虽然可以监听爱丽丝和波比之间的通信信道，但她无法将 x 修改为 x' 而不被波比发现；这是由于伊蔚能够正确预测 $f_k(x')$ 的概率是可忽略的（毕竟随机函数是不可预测的）。此外，伪随机函数还深刻地阐释了“为什么目前的下界技术不能够区分 **P** 和 **NP**”；参见第 23 章。

9.5.2 去随机化

伪随机数产生器的存在性将得到 **BPP** 问题的亚指数时间的确定型算法。这一过程也通常称为 **BPP** 的去随机化。也就是说，如果 $L\in\mathbf{BPP}$，则对任意 $\varepsilon>0$ 均存在时间复杂度

为 2^{n^2} 的确定型算法 A 使得 $\Pr[A(X_n)=L(X_n)]>0.99$ 对 $X_n\in\{0,1\}^n$ 的任意可抽样分布 $\{X_n\}$ 均成立。注意，上述概率是相对于输入的随机选取而言的，而算法 A 是确定型的。算法 A 可以简单地如下工作：先利用伪随机数产生器将 L 的概率型算法需要做的随机选择次数降低到 n^2，然后再枚举伪随机数产生器的可能的所有输入。在第 20 章中，我们将给出 **BPP** 问题的更强的去随机化。

9.5.3　电话投币和比特承诺

A，B 双方如何通过电话产生投掷均匀硬币的结果呢？（许多密码协议都以此为基础。）如果仅由一方产生投掷硬币的结果，则无法防止他谎报投掷硬币的结果。下面的方法克服了这一问题：双方各投掷一枚硬币，然后在双方的投掷结果上执行异或操作，将操作结果作为双方公认的投掷结果。即使 B 不相信 A 在投掷硬币时使用的是均匀硬币，B 也知道只要自己随机地投掷了硬币，则异或操作的结果仍将是随机的。不幸的是，上述思想仍然不行，这是由于首先亮出投掷结果的一方将很不利，因为另一方可以据此"调整"己方的投掷结果使得异或操作的结果有利于自己。

上述问题可以由下面的方案来解决：假设 A，B 双方均是多项式时间图灵机，并且他们都无法求出单向置换的逆函数。首先，A 选择长度为 n 的两个串 x_A 和 r_A 并发送消息 $(f_n(x_A),r_A)$，其中 f_n 是一个单向置换。接下来，B 选择一个随机比特 b 并将它发送给 A。然后，A 亮出 x_A。双方一致同意将异或操作作用于 b 和 $x_A\odot r_A$ 上得到的结果作为双方公认的投掷结果。注意，B 应用函数 f_n 即可验证 x_A 是否是第一个消息中使用的串；因此，A 在知晓 B 的投掷结果之后也无法更改 x_A。（正是由于这一原因，人们将 A 的第一个消息称为 $x_A\odot r_A$ 这一比特的密码承诺。）另一方面，由定理 9.12 可知，B 无法根据 A 的第一个消息来预测 $x_A\odot r_A$，因此，他也无法根据 $x_A\odot r_A$ 来调整己方的投掷结果。

9.5.4　安全的多方计算

安全的多方计算极大地推广了 9.5.3 节处理的情况，它共涉及 k 方，第 i 方持有串 $x_i\in\{0,1\}^*$。参与计算的各方均希望计算函数值 $f(x_1,x_2,\cdots,x_k)$，其中 192
$f:\{0,1\}^*\to\{0,1\}$ 是各方均知晓的多项式时间可计算函数。（9.5.3 节所讨论的是安全多方计算的特殊情况，其中 x_i 仅由随机产生的一个位构成而 f 是异或操作。）显然，参与计算的各方可以交换各自的输入（如有必要，交换过程需要对输入进行加密以确保窃听者无法了解任何信息），然后各方自己计算 f 的函数值。但是，这样的话参与计算的各方均能了解其他各方的输入，这在许多应用中是不允许的。例如，在几家医院各自持有的医疗数据库上联合计算统计值（如均值）时，严格的《隐私保护与保密法》将不允许各家医院共享病人的信息。姚期智定义安全多方计算时给出的原始例子是，k 个人在各自不向他人泄露自己的工资的条件下计算出他们的平均工资。

我们称计算 f 的一个多方协议是安全的，如果参与计算的各方最终除了计算得出 $f(x_1,x_2,\cdots,x_k)$ 之外不能了解更多的其他信息。安全多方计算协议的形式定义参照了零知识证明的定义，它要求参与计算的各方在协议执行过程中能够了解的信息是他们在理想情况下能够从模拟计算中了解到的信息；其中理想情况下，参与计算的各方均信任一个权威方，他们在模拟计算中都将自己的输入发送给权威方，而权威方在收到的所有输入上计算函数 f 的值，并将结果发送给各方。令人惊讶的是，对于任意数量的参与方和任意的

多项式时间可计算函数 f，均存在满足上述要求的安全多方计算协议；参见本章注记。[⊖]

9.5.5 机器学习的下界

机器学习的目的是从形如$(x_1, f(x_1))$，$(x_2, f(x_2))$，…的序列中学习得到一个紧凑的函数 $f: \{0,1\}^n \to \{0,1\}$，其中 x_i 是随机选取的输入。显然，在一般情况下这是不可能的，因为随机函数不存在紧凑的表示形式。但是，如果假设 f 存在(如小的布尔线路的)紧凑表示形式，我们能学习得到 f 的紧凑形式吗?

伪随机函数的存在性表明，即使一个函数是多项式时间可计算的，也不可能在实例上通过多项式时间的学习来得到这个函数。不可能用机器学习来获得的函数还包括一些限制性更强的函数族，例如 $\mathbf{NC}^1$(参见卡恩斯(Kearns)和瓦利安特[KV89])。

本章学习内容

- 密码学是一个古老的学科。人们曾发明了许多密写方法，但这些方法最终都被破解了。20 世纪 70 年代中期，基于某些计算问题目前仍难以求解的事实，诞生了现代密码系统。许多现代密码系统从未被破解过。
- 从信息论的观点看，安全加密方案要求窃听者即使在具有无限计算能力的条件下也无法根据密文了解明文的相关信息。这种安全加密方案可以由一次一密加密方案实现，但一次一密方案的致命缺陷在于它使用的密钥太长。
- 更切合实际的一次一密方案要用到伪随机数产生器，它能够将长度较短的真随机串拉伸成长度大得多的伪随机串，并且拉伸后的串在敌手"看来像随机串"。
- 单向函数是这样一类函数，它将串映射为易于计算的串，但在一般的输入上却很难计算函数的逆。人们提出了很多候选的单向函数，但是由于相关计算问题的复杂性下界仍未被证明，这些候选单向函数还不能最终被证明确实是单向函数。
- 单向函数存在当且仅当伪随机数产生器存在。上述结论的必要性很容易证明，充分性的证明要复杂一些。在本章中，我们看到了如何由单向置换这一类特殊的单向函数来构造伪随机数产生器。
- 现代密码学远不限于仅研究安全加密方案。许多协议被设计用于完成更复杂的计算任务，包括随机数的产生、投票、审计，等等。在设计这种协议时，零知识证明具有重要作用。
- 伪随机函数可以视为具有"指数拉伸度"的伪随机数产生器。伪随机函数在密码学中有许多应用，它的另一个精彩绝伦的应用将出现在第 23 章中。

193

本章注记和历史

在本章讲述过程中，我们将潜在的窃听者和可能存在的单向函数的求逆算法都视为多项式时间的概率型图灵机。另一种合理的选择是，用多项式规模的线路作为窃听者和单向函数求逆算法的模型；或者等价地，也可以用一种特殊的多项式时间概率型图灵机作为二者的模型，这种图灵机可以将依赖于输入长度的多项式规模的常数"硬件写入"到图灵机上

⊖ 回到医疗数据库的例子中，各家医院可以通过安全多方计算协议在联合数据库上计算统计值而不互相泄露自己持有的信息，至少那些可以直接从数据库抽取到的信息不会被泄露，还不清楚的是，现行的《隐私保护法》是否允许医院在安全多方计算中使用病例数据，这也是法律滞后于科学进展的一个例子。

作为图灵机的建言。本章给出的所有结论对这种模型也成立，并且采用这种模型之后某些定义和证明将变得稍简单一些。为了避免本章内容依赖于第 6 章，我们选用一致图灵机作为模型。

戈德赖希的书[Gol04]很好地阐述了本章的大部分内容(还讨论了本章未涉及的一些内容)。本科生教科书[KL07]宽泛地介绍了一些基本的事实。关于最近出现的一些主题，特别是应用密码学中最近出现的一些主题，请参阅伯纳(Boneh)和舒普(Shoup)即将出版的新书[BS08]。关于计算数论的更多内容，请参阅舒普的书[Sho05]或巴赫(Bach)和沙利特(Shallit)的书[BS96]。

卡恩(Kahn)的书[Kahn96]精彩地讲述了密码学在各个时期多姿多彩的历史。截止到 20 世纪 50 年代，密码学的发展历史一直遵循本章开头引用的埃德加·阿兰·伯奕(Edgar Alan Poe)的断言：也就是说，人们设计并广泛使用的每种加密方案最终都被破解了。香农[Sha49b]首次严格地研究了加密方案的安全性：他证明了 9.1 节的结论，给出了安全性的首个形式定义，并证明了加密方案满足安全性定义的充分必要条件是密钥与消息一样长。香农认识到解决这一局限性的方法要用到计算难解性，但是他未能给出具体的方法。这一点不足为怪，因为用数学方法研究高效计算(如算法设计和复杂性理论)始于 20 世纪
194 60 年代；而正是由于这些研究，人们才最终明确地区分了多项式时间和指数时间。

1974 年前后，迪菲和赫尔曼开始研究安全通信这一古老概念是否真的需要事先共享密钥；同时独立开展这项研究的还有梅克莱(Merkle)。这项研究让迪菲和赫尔曼提出了公钥密码学的概念并发表了这一领域的开山之作[DH76]。这篇论文还给出了公钥密码学的第一个实现方案，也就是大家现在熟知的迪菲-赫尔曼密钥交换协议；该协议立刻得到一个公钥密码方案，也就是大家现在熟知的 El-Gamal 密码。但是，要在保密和带认证的通信中完全实现迪菲和赫尔曼提出的无需共享密钥的加密方案，还需要用到陷门置换。迪菲和赫尔曼曾猜想陷门置换是存在的，但未能给出具体的实现。[⊖]陷门置换首先由李维斯特(Rivest)、萨米尔(Shamir)和阿德尔曼(Adleman)构造出来[RSA78]；由此得到的加密方案和签名方案非常高效，并且它们目前仍然是被最广泛地采用的方案。李维斯特等人曾猜想，他们构造的陷门置换等价于因数分解问题，但他们未能给出证明(直到现在也没有找到证明)。后来，拉宾[Rab79]给出了一个等价于因数分解问题的陷门置换。

有意思的是，类似的研究在封闭的情报界也得到了开展；事实上，情报界开展的工作可能比[DH76, RSA78]还早，尽管这些工作在 20 年之后才曝光[Ell99]。在 1970 年，在英国情报局下属的国家通信总局(GCHQ)工作的詹姆士·埃利斯(James Ellis)也曾意识到，无需共享密钥也有可能实现安全通信。在 1973 年之前，情报局中没有人能够具体地实现他的想法。1973 年，克利福德·柯克斯(Clifford Cocks)建议使用一种与 RSA 陷门置换非常接近的陷门置换；几个月后，马尔科姆·威廉姆森(Malcolm Williamson)发现了一个密钥交换协议，也就是如今的迪菲-赫尔曼密钥交换协议。(但情报界并未预知到其他相关的概念，如数字签名、拉宾陷门置换和基于编码或格的公钥密码方案等。)或许，密钥交换协议的相关概念首先由国家通信总局提出而不是首先见诸于公开的文献也是合情合理的，这是因为，从香农论文发表到[DH76]发表这段时间，情报界之外的其他地方很难开

⊖ 迪菲和赫尔曼实际上将如今称为陷门置换的概念称为"公钥密码"。事实上，陷门置换可以视为公钥密码的一种变形，这种变形使用了确定型加密函数而不使用概率型加密函数。但是，由[GM82]中的工作可知，概率型加密函数不仅对获得更强的安全性是必需的，而且对获得不使用陷门函数的加密方案(如迪菲-赫尔曼加密方案和 El-Gamal 加密方案)也非常有益。

展密码学相关的研究。

尽管密码交换协议是一个巨大的成就，但 RSA 和迪菲-赫尔曼协议所获得的安全性仍不能完全令人满意，它们实现的安全性与香农用一次一密方案展示的安全性不匹配，因为香农的安全性要求消息的任何部分的信息均不能被泄露。戈德瓦瑟和米卡利[GM82]在论文中说明了如何才能获得这样强的安全性；这篇论文为后来得到的许多具有更强的安全性的加密方案或其他计算方案奠定了基础，同时也促进了这些方案的设计。另一个里程碑由戈德瓦瑟、米卡利和李维斯特[GMR84]共同建立，他们给出了数字签名的强安全性的定义，同时还说明了在因数分解问题难以求解的假设条件下如何实现数字签名的强安全性。

在早期的计算中，人们就已经实际地使用了伪随机数产生器。萨米尔[Sha81]创先建立了难解性和伪随机性之间的联系，他证明了：如果 RSA 函数是单向函数，则存在满足 195
某种弱随机性(即分组难预测性(block unpredictability))的产生器。布卢姆和米卡利[BM82]定义了稍强的随机性，也就是后位难预测性(next-bit-unpredictability)，并证明了基于整数分解的产生器满足后位难预测性。姚期智[Yao82a]定义了更强的随机性，这种随机性要求能够骗过所有的多项式时间检测(定义 9.8)，并证明了这种随机性等价于后位难预测性(定理 9.11)。在这篇论文中，姚期智还证明了引理 9.10(他的证明不同于本章给出的证明)，亦即用单向置换构造了一个伪随机数产生器。戈德赖希-勒维定理的证明出现在[GL89]中，但是本章采用了拉科夫(Rackoff)未发表的证明。(用单向函数构造伪随机数产生器的)定理 9.9 的颇具技术深度的证明源自哈斯塔德(Håstad)、因帕利亚佐、卢比(Luby)和勒维[HILL99](该论文的会议版本比论文早发表 10 年)。9.5.1 节中伪随机函数的构造应归功于戈德赖希、戈德瓦瑟和米卡利[GGM84]。

零知识证明的发明源自戈德瓦瑟、米卡利和拉科夫[GMR85]，他们给出了二次剩余问题的一个零知识证明(参见例 8.9)。戈德赖希、米卡利和维格德尔森[GMW86]证明了如果单向函数存在，则任意 **NP** 语言均存在可计算的零知识证明系统。例 9.15 中给出的图同构的零知识协议也源自同一篇论文。布拉萨尔(Brassard)、乔姆(Chaum)和克雷皮奥(Crépeau)独立地在一个具体的难解性假设下给出了 **NP** 语言的完美零知识证明；在完美零知识证明中，可靠性指的是可计算性，零知识性条件要相对于计算能力不受限的敌手。

姚期智[Yao82b]首先提出了两方计算的安全协议(参见 9.5.4 节)，但他的协议仅适用于消极敌手(有时也称为窃听者或"诚实而好奇"的敌手)。戈德赖希、米卡利和维格德尔森[GMW87]将上述协议扩展到任意数量的参与方，并说明了如何用零知识证明来获得相对于"主动"攻击的安全性。自此以后，安全多方计算才被广泛使用。

一些早期的密码系统建立在子集和问题 SUBSET SUM 的基础上；20 世纪 80 年代以前，许多这样的密码系统都被破解了。近年来，人们又重新开始研究这些问题以及相关的最短格向量问题和最近格向量问题；这些研究都源于 Ajtai[Ajt96]给出的一个单向函数以及 Ajtai 和 Dwork[AD97]给出的一个公钥密码系统(其他研究者后来不断改进了该密码系统)。这些构造对大多数实例是安全的当且仅当它们在最坏情况实例上是安全的。(所用的思想是随机自归约的一种变形。)奥代德·雷格夫(Oded Regev)的综述[Reg06]和他的讲义(从他的主页可以下载)提供了这一迷人的研究领域的更多素材(也可以参阅更早一些的书[MG02])。人们希望借助这些思想能够将密码学建立在类似于 $\mathbf{P}\neq\mathbf{NP}$ 或 $\mathbf{NP}\cap\mathbf{coNP}\not\subseteq\mathbf{BPP}$ 这样的关于最坏情况的猜想上，但是达到上述目的还存在一些重大的障碍。

许多研究工作致力于探求各种密码学任务所需的安全性的确切概念。例如，语义安全

性的概念(参见 9.2.2 节和习题 9.9)看起来似乎够强了，但事实证明语义安全性对大多数
196 应用而言是不够的，我们需要更强的选择密文安全性[RS91，DDN91]。要了解这一专题的更多内容，请参阅伯纳和舒普的书[BS08]。在获得选择密文安全性的过程中，零知识证明也发挥了关键作用。

习题

9.1　证明：一次一密加密方案满足定义 9.1 中给出的完全安全性。

9.2　证明：如果(E，D)是满足(9.1)式的一个加密方案，它的消息长度为 m 并且密钥长度为 $n<m$，则存在 x，$x'\in\{0,1\}^m$ 使得 $\mathrm{E}_{U_n}(x)$ 和 $\mathrm{E}_{U_n}(x')$ 不服从相同的分布。

9.3　证明：在一次一密加密方案中，任意窃听者无法以大于 1/2 的概率猜出明文的任意一个位。也就是证明，对于任意函数 A，如果(E，D)表示一个一次一密加密方案，则

$$\Pr_{\substack{k\in_R\{0,1\}^n\\ x\in_R\{0,1\}^m}}[A(\mathrm{E}_k(x))=(i,b)\text{ s.t. }x_i=b]\leqslant 1/2$$

可见，一次一密加密方案以更强的形式满足(9.3)式定义的计算安全性。

9.4　习题 9.2 和引理 9.2 表明，为确保加密方案相对于计算时间不受限的窃听者(如果 **P**=**NP**则包括高效的窃听者)是安全的，密钥的长度需要等于消息的长度。这样的密码方案还隐含地用到了一个微妙的假设条件：加密过程是确定型的。在概率型加密方案中，加密函数 E 可以是概率算法；也就是说，给定消息 x 和密钥 k，密文 $\mathrm{E}_k(x)$ 不再是固定的值，而是服从某个分布 $Y_{x,k}$。当然，由于解密函数 D 只知道选定的密钥 k，而不知道 E 在加密过程中所做的随机选择，因此，(9.1)式需要调整为要求“$\mathrm{D}_k(y)=x$属于 $\mathrm{E}_k(x)$的支持域对任意 y 成立”。证明：即使对于概率型加密方案，密钥也不能显著地比消息短。也就是证明，对于密钥长度为 n 而消息长度为 $n+10$ 的任意概率型加密方案(D，E)，必存在两个消息 x_0，$x_1\in\{0,1\}^{n+10}$ 和一个函数 A 使得

$$\Pr_{\substack{b\in_R\{0,1\}\\ k\in_R\{0,1\}^n}}[A(\mathrm{E}_k(x_b))=b]\geqslant 9/10 \tag{9.11}$$

进一步证明：如果 **P**=**NP**，则函数 A 可以在多项式时间内计算。

9.5　证明：如果 **P**=**NP**，则不存在单向函数。

9.6　(a) 证明：如果存在单向函数 f，则也存在 n^2 时间内可计算的单向函数 g。

197 (b) 证明：如果存在单向函数 f，则 9.2.1 节中给出的函数 f_M 也是单向的。

9.7　证明：对于 9.2.1 节中的拉宾函数 $f_M(X)=X^2\pmod M$，如果存在时间复杂度为 $\mathrm{poly}(\log M)$的算法能够为 $1/\mathrm{poly}(\log M)$比例的输入求逆，则可以在 $\mathrm{poly}(\log M)$时间内得到 M 的因数分解。

9.8　令$\{(p_n,g_n)\}_{n\in\mathbf{N}}$是由 n 位整数对构成的一个序列，其中 p_n是素数，g_n是群 $\mathbf{Z}_{p_n}^*$的生成元素，并且存在一个确定型多项式时间算法 S 使得 $S(1^n)=(p_n,g_n)$对任意 $n\in\mathbf{N}$ 成立。

假设 A 是一个运行时间为 $t(n)$的算法，对 $\delta(n)$比例的 $x\in\{0,1,\cdots,p_n-1\}$，该算法能以 $g_n^x\pmod{p_n}$为输入计算出 x。证明：对于任意 $\varepsilon>0$，存在运行时间为 $O\left(\frac{1}{\delta\log 1/\varepsilon}(t(n)+\mathrm{poly}(n))\right)$的概率算法 A'使得 $\Pr[A'(g_n^x\pmod{p_n})=x]\geqslant 1-\varepsilon$对任

意 $x\in\{0,1,\cdots,p_n-1\}$ 成立。上述性质就是大家熟知的离散对数问题的自归约性。

9.9 称满足 $X_n\in\{0,1\}^{m(n)}$ 对某个多项式 $m(\cdot)$ 成立的随机变量序列 $\{X_n\}_{n\in\mathbf{N}}$ 是可抽样的，如果存在概率多项式时间算法 D 使得 X_n 的分布等于 $D(1^n)$ 的分布在任意 n 上成立。设(E, D)是一个加密方案，它用长度为 n 的密钥来加密长度为 $m(n)$ 的消息，其中 $m(\cdot)$ 是一个多项式。加密方案(E, D)称为语义安全的，如果对任意的可抽样序列 $\{X_n\}$(其中 $X_n\in\{0,1\}^{m(n)}$)，任意多项式时间可计算函数 $f:\{0,1\}^*\to\{0,1\}$ 和任意的概率多项式时间算法 A 而言，均存在一个可忽略函数 $\varepsilon:\mathbf{N}\to[0,1]$ 和一个概率多项式时间算法 B 使得

$$\Pr_{\substack{k\in_R\{0,1\}^n\\ x\in_R X_n}}[A(\mathrm{E}_k(x))=f(x)]\leqslant\Pr_{x\in_R X_n}[B(1^n)=f(x)]+\varepsilon(n)$$

也就是说，给定 x 的密文，A 计算出 $f(x)$ 的概率不会大于根据 X_n 的分布直接猜出 $f(x)$ 的概率。

(a) 证明：如果(E, D)是语义安全的，则它也必然满足 9.2.2 节中"计算安全性"的各项条件。

(b) 证明：如果 G 是将 $\{0,1\}^n$ 映射到 $\{0,1\}^{n'}$ 的伪随机数产生器，则加密方案 $\mathrm{E}_k(x)=x\oplus G(k)$，$D_k(y)=y\oplus G(k)$ 是语义安全的。

(c) 证明：语义安全等价于它的如下特殊情况：对于任意 n，X_n 是串对 x_n^0，x_n^1 上的均匀分布，且 f 在任意 n 上都将 x_n^0 映射为 0 而将 x_n^1 映射为 1。

9.10 证明：如果存在拉伸度为 $\ell(n)=n+1$ 的安全伪随机数产生器，则对任意 c 均存在拉伸度为 $\ell(n)=n^c$ 的伪随机数产生器。

9.11 证明：如果 f 是一个单向置换，则 f^k 也是一个单向置换，其中 $k=n^c$ 对某个固定的 $c>0$ 成立，并且 $f^k(x)=f(f(f(\cdots f(x)\cdots)))$(亦即将 f 在 x 上重复作用 k 次)。

9.12 假设单向函数存在，证明习题 9.11 的结论对单向函数不成立。也就是说，请设计一个单向函数 f 使得对任意 $c>0$ 而言 f^{n^c} 都不是单向函数。 198

9.13 假设 $x\in\{0,1\}^n$ 是一个未知向量。设 $r^1,\cdots,r^m\in\{0,1\}^n$ 是随机选取的，并且我们知道 $x\odot r^i$，$i=1,2,\cdots,m$。给出一个确定型算法根据已知信息来重构 x，并证明：相对于 r^i 的随机选择而言，重构算法得到正确向量的概率至少为 1/4。这就表明，如果 $r^1,\cdots,r^m$ 是完全独立的，则我们不可能以大于 2^{-n} 的概率猜测 $x\odot r^1,\cdots,x\odot r^m$；进而，在定理 9.12 的证明中要求"向量集合仅两两独立"是很关键的。

9.14 假设某人持有一个未知的 n-位向量 a。每当你给他一个随机选取的下标子集 $S\subseteq\{1,\cdots,n\}$，他至少以概率 $1/2+\varepsilon$ 告诉你"a 中下标位于 S 的等于 1 的位的个数的奇偶性"。给出一个猜测策略使得它能够让你至少以概率 $\left(\frac{\varepsilon}{n}\right)^c$ 得到 a(一个长度为 n 的位串)，其中 $c>$ 是一个常数。

9.15 称满足 $X_n,Y_n\in\{0,1\}^{m(n)}$ 对某个多项式 $m(n)$ 成立的两个随机变量序列 $\{X_n\}$，$\{Y_n\}$ 是计算不可分的，如果对于任意概率多项式时间算法 A 均存在一个可忽略函数 $\varepsilon:\mathbf{N}\to[0,1]$ 使得

$$|\Pr[A(X_n)=1]-\Pr[A(Y_n)=1]|<\varepsilon(n)$$

对任意 n 成立。证明：

(a) 如果 $f:\{0,1\}^*\to\{0,1\}^*$ 是一个多项式时间可计算的函数并且 $\{X_n\}$，$\{Y_n\}$ 是计算不可分的，则 $\{f(X_n)\}$，$\{f(Y_n)\}$ 也是计算不可分的。

(b) 拉伸度为 $\ell(n)$ 的多项式时间可计算的函数 G 是一个安全的伪随机数产生器当且仅当序列 $\{U_{\ell(n)}\}$ 和 $\{G(U_n)\}$ 是计算不可分的。

(c) 密钥长度为 n 而消息长度为 $\ell(n)$ 的加密方案(E，D)是语义安全的当且仅当对于满足 $|X_0(1^n)|=|X_1(1^n)|=\ell(n)$ 的任意两个概率多项式时间算法 X_0，X_1 而言，序列 $\{\mathrm{E}_{U_n}(X_0(1^n))\}$ 和 $\{\mathrm{E}_{U_n}(X_1(1^n))\}$ 是概率不可分的。

9.16　假设单向置换存在。证明：在 n 顶点图上存在多项式时间内可抽样的计算不可分的两个分布 $\{G_n\}$ 和 $\{H_n\}$ 使得：对于任意 n，G_n 和 H_n 是 n 顶点图，并且 $\Pr[G_n\text{是3-可着色的}]=1$ 但 $\Pr[H_n\text{是3-可着色的}]=0$。(图 G 是3-可着色的，如果它的每个顶点可以用3种颜色之一进行着色，使得任意相邻顶点的颜色不同。参见习题 2.2。)

9.17　称语言 L 存在一个计算零知识证明，如果该语言满足定义 9.14 的宽松形式。亦即将条件(9.9)替换为条件 $\{\mathrm{out}_V\langle P(X_n,U_n),V^*(X_n)\rangle\}=S^*(X_n)\}$，其中 $S^*(X_n)$ 与满足 $|X_n|=n$ 和 $\Pr[M(X_n,U_n)=1]=1$ 的任意可抽样分布 (X_n,U_n) 之间是计算不可分的。

(a) 证明：如果勒维归约(参见 2.3.6 节)下的一个 **NP**-完全语言 L 存在计算零知识证明，则任意 $L\in\mathbf{NP}$ 均存在计算零知识证明。

199

(b) 证明：下面的协议(源自布卢姆[Blu87])是哈密顿环语言的一个完备性为1且可靠性错误为1/2的计算零知识证明：[⊖]

公共输入：n 个顶点上的图 G。

证明者私有的输入：图中的一个哈密顿环。

证明者的第一个消息：随机选择 G 的所有顶点上的一个置换 π，令 M 是将 G 的邻接矩阵的所有行和所有列分别用 π 进行置换后得到的矩阵。对任意 $i,j\in[n]$，随机选取 $x^{ij},r^{ij}\in_R\{0,1\}^n$，然后将 $\pi(x^{ij})$，r^{ij}，$(x^{ij}\odot r^{ij})\oplus M_{ij}$ 发送给验证者。

验证者的消息：随机选取 $b\in_R\{0,1\}$ 并将 b 发送给证明者。

证明者的最后一个消息：如果 $b=0$，证明者将第一个消息中使用的所有随机数发送给验证者；亦即，证明者将置换 π、邻接矩阵 M 和 x^{ij}(其中 i，j 取遍 $[n]$)告诉验证者。如果 $b=1$，则证明者先计算 C 经置换 π 变换后得到的环 C'；也就是说，对于任意 $\overline{ij}\in C$，C' 包含边 $(\pi(i),\pi(j))$。然后，证明者将 C' 和其中边对应的随机数发送给验证者；也就是说，仅将任意边 $(i,j)\in C'$ 对应的随机数 x^{ij} 发送给验证者。

验证者检验：如果 $b=0$，则验证者查验证明者给出的信息是否与第一个消息一致；亦即，查验 G 的邻接矩阵在 π 的变换下能否得到 M，还需查验 x^{ij} 是否与第一个消息发送过来的 y^{ij} 一致。如果 $b=1$，则验证者查验 C' 是否是一个哈密顿环，同时查验验证者发送的值是否与第一个消息发送的一致，亦即 $M_{ij}=1$ 对任意 $(i,j)\in C'$ 是否成立。验证者接受当且仅当上述查验过程成功。

200

⊖ 可靠性错误可以通过重复执行协议来降低。

量子计算

谈到量子力学……秘密，秘密，快关上门！在理解量子力学表示的世界时，我们总会遇到大量的困难……就我而言，并不是说就显然不存在任何真正的问题了。我无法定义真正的问题，因而我怀疑是否存在真正的问题，但我又无法肯定不存在真正的问题。这正是我愿意对事情展开研究的原因。

——理查德·费曼(Richard Feynman)，《用计算机模拟物理》，1982

概率型经典世界和量子世界的方程之间唯一的区别在于，不知道为什么在后者中概率必须要取负值。

——理查德·费曼，《用计算机模拟物理》，1982

我们的第一个结果是德鲁奇模型(Deutsch's Model)下高效的通用量子图灵机的存在性……我们给出第一个形式化的证据，以表明量子图灵机违背了现代复杂性理论的基本猜想——邱奇-图灵命题。我们证明，存在相对于某个神喻的一个计算问题，它在量子图灵机上可以用多项式时间求解，而在错误概率受限的概率型图灵机上却需要超多项式时间才能求解。

——E·伯恩斯坦(E. Bernstein)，U·瓦兹拉尼(U. Vazirani)，《量子复杂性理论》，1997

量子计算是一种可能被物理实现的新的计算模型，其计算速度指数倍地快于经典的概率型图灵机和确定型图灵机。本章综述量子计算的基本原理和量子计算模型上的重要算法。

研究量子计算机的一个重要原因是，它构成了强邱奇-图灵命题(参见 1.6.3 节)的严重挑战：该猜想断言，任意可物理实现的计算装置都可以被图灵机模拟而计算速度至多下降一个多项式因子。在 10.6 节我们将看到，在量子计算机上存在整数分解问题的多项式时间算法；然而，在概率型图灵机和确定型图灵机上，人们经过长期努力仍未为整数分解问题找到多项式时间算法。如果整数分解问题根本不存在高效的经典算法(目前，现实社会严重依赖于这一猜想，因为诸如 RSA 等密码方案的安全性全部建立在该猜想的基础之上)，又如果量子计算机是可物理实现的，则强邱奇-图灵命题就是错误的。此外，物理学家也对量子计算机感兴趣，因为这有助于他们理解量子力学理论。目前，尽管量子力学理论在预测实验中取得了很大成功，但人们尚未完全理解该理论。

理解量子计算领域取得的核心成果仅需很少的物理知识。首先，电子等基本粒子的(能量、动量和自旋等)物理参数都是量子化的并且只能取值于一个离散集合。其次，与我们的基本直觉不同的是，粒子的(位置和能量等)物理参数在任何时间点上的取值都不是单个数值，而是与粒子相关联的一种概率波，它将相应参数的所有可能取值进行"模糊"或"叠加"。只有当观察者对参数进行测量时，该参数才会取得明确的数值，此时我们称相应的概率波塌陷到单个数值。

参数值被观测之前的模糊效应看上去颇具如下的哲学韵味：如果森林中的一棵大树倒

下来时没有人听到声响，那它真的弄出动静来了吗？但粒子参数的概率波却是确实存在的，这些概率波的相互作用和交互干涉产生了一些实验可测的效应。而且，由量子力学可知，概率波不仅仅与单个粒子相关联，而且还与粒子簇相关联(比如，人也可以视为一簇粒子!)。不同粒子簇关联的概率波之间的相互作用正是量子计算的计算能力的关键，正是这种交互作用使得在量子计算模型上某些问题的求解可以被加速指数倍。另一方面，量子计算机可以极其方便地视为一个"大规模并行"的计算机，很多广受欢迎的科普著作的作者都采用了这种观点。这种"大规模并行"也严格地遵循量子力学的规律，这使得量子计算目前仅能将少数几个具有良好结构的问题的求解速度提高指数倍。

本章组织结构如下。10.1 节介绍双缝实验，它是能够展示量子力学中模糊效应和干涉效应的诸多实验之一。10.2 节形式地给出一个称为量子位(简记为 qubit)的小型量子系统，它是量子计算的基本单元。我们给出可以在一个或几个量子位构成的量子系统上执行的操作，10.2.1 节利用 EPR 悖论展示这些操作，其中 EPR 悖论是一个旨在展示和验证"量子力学有悖于直观直觉"的实验。10.3 节定义 n-qubit 的量子寄存器和可以在这种寄存器上执行的操作(也包括计算)。我们定义量子线路和复杂性类 **BQP**，它是量子计算模型下类似于 **BPP** 的复杂性类。接下来的三个小节依次介绍量子计算机上的三个著名的基本算法，它们的发明者分别是格罗弗(Grover)、西蒙(Simon)和肖尔(Shor)。本章末论及量子计算的几个重要主题，包括量子计算的计算能力的下界、量子密码学和量子纠错；本章注记提供了可供进一步阅读的材料。

本章要用到线性代数和空间 $\mathbf{C}^n$ 的一些基本事实。附录 A 概述了这些事实，也可以参见 10.3.1 节。

10.1　量子怪相：双缝实验

202

现在，我们描述双缝实验来说明如下事实：基本粒子的基本物理性质是"模糊的"。

如图 10-1 所示，在一面有两条细缝的墙前放置一个光源。假设光源一个一个地发射光子(比方说，每秒发射一个光子)。墙后放置一组感应器；每当光子击中感应器时，感应器都会亮灯。我们统计每个感应器在一个小时内亮灯的总次数。当遮挡住其中一条细缝时，我们希望观察到位于打开的细缝后侧的感应器被光子击中的次数最高；图 10-1 表明，实际情况确实如此。如果两条细缝都打开，我们希望观察到每个感应器被光子击中的次数等于第 1 条细缝单独打开时该感应器被光子击中的次数与第 2 条细缝单独打开时该感应器被光子击中的次数之和。特别地，在两条细缝同时打开时，各个感应器被光子击中的次数应该增加。

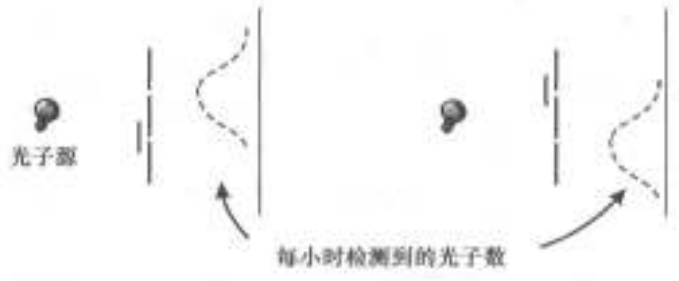


图 10-1　双缝实验中，光子源放在一面有两条细缝的墙前，墙后放置一组感应器。当一条缝被遮挡后，正如期望的观察结果一样，检测到的光子数量在打开的细缝后侧达到最大

出人意料的是，真正发生的现象却并非如此。如图 10-2 所示，各个感应器被光子击中的次数呈现出“干涉”现象。特别地，在几个感应器上，当两条缝同时打开时光子击中该感应器的总次数低于仅有一条缝单独打开时光子击中该感应器的总次数。这说明，认为光子仅具有粒子性或仅具有“小球”的性质是站不住脚的。



图 10-2　如果双缝实验中将两条细缝都打开，每个位置检测到的光子数量不等于两条细缝分别单独打开时该位置检测到的光子数量之和。甚至在某些位置上，在两条细缝分别单独打开时能检测到光子，但在两条细缝同时打开时却检测不到光子

由量子力学可知，“认为光子是只能通过第 1 条细缝或第 2 条细缝的小球”是错误的；或者说，“认为光子仅明确地具有粒子性”是错误的。相反，光子能够通过某种未知的方式在瞬间探寻到通过两条细缝击中感应器的所有可能的路线，其中某些路线具有正“振幅”，而另一些路线具有负“振幅”（参见本章开头费曼的引言）；两条具有相反符号的路线抵达感应器时将会相互抵消。最终造成的结果是，各个位置上感应器被击中的次数的分布依赖于处于打开状态的细缝的数量，这是由于光子能够通过探测所有可能路线来“发现”有几条细缝处于打开状态。

你可能会怀疑这种“路线探测”。为了检测光子是否真的进行了路线探测，你在每条细缝中放置一个感应器，使得每当一个光子通过细缝时，处于该细缝的感应器就会亮灯。因此，如果一个光子真的同时通过两条细缝，那么在两条细缝处都会检测到它。但是，当你用这种方式来让光子显露其量子本性时，量子本性（即干涉现象）却消失了。墙后的每个感应器被击中的次数恰好等于将两条细缝单独打开时该感应器被击中次数之和；这就是说，此时光子仿佛又变成了小球。这种现象的原因是，如前面所述，观测粒子会使其概率分布塌陷为一个数值继而改变了实验结果。[⊖]这背后的寓意是，量子计算机如果真被物理实现，它也必须从外部影响和噪音中隔离出来，因为噪音也可以视为“环境对系统实施的检测”。当然，我们无法完全地隔离系统，这意味着必须让量子计算能够容忍一点噪音。在某些噪音模型下，这是有可能实现的；参见本章注记。

10.2　量子叠加和量子位

下面，我们用一种简单的量子系统来刻画量子叠加，这种量子系统称为量子位，它是下一节形式定义量子计算的基础。我们还为那些不熟悉量子力学的读者给出一个有助于理解量子力学的例子——EPR 悖论；EPR 悖论有时也称 EPR 佯谬。严格地讲，EPR 悖论对理解本章其余内容不是必需的。

传统计算需要对存储在内存中的元素进行操作，而内存具有有限个存储单元。这些存储单元既可以是图灵机的存储带上的一个单元，也可以是布尔线路中的一个二进制位。量子计算机上类似的存储单元称为量子位。我们可以将量子位视为一个具有两种基态的基本

⊖　当然，目前仍不清楚为什么人或人放置的感应器会使概率波塌陷但像细缝这样无生命的对象却不会使概率波塌陷。这正是困扰量子力学的难题之一，参见本章注记。

粒子：这两种基态对应于粒子在能量、自旋或其他物理参数上的取值，将它们分别表示为1和0。但是，不同于传统的二进制位，粒子可以同时处于两种基本态。因此，量子位在任意时刻的状态称为两个基本态的叠加。正式地，我们将粒子的两种基本态分别记为$|0\rangle$和$|1\rangle$；一般情况下，量子位处于形如$\alpha_0|0\rangle+\alpha_1|1\rangle$的任意状态，其中$\alpha_0$，$\alpha_1$称为振幅并且是满足$|\alpha_0|^2+|\alpha_1|^2=1$的复数。[⊖] 如果量子位不受外界影响，则它将始终处于叠加态，直到观测者对它进行观测。如果量子位被观测，则观测到状态$|0\rangle$的概率为$|\alpha_0|^2$而观测到状态$|1\rangle$的概率为$|\alpha_1|^2$。观测之后，量子位的振幅塌陷了，振幅的取值将会不可逆转地永
204 远消失了。

本节仅限于讨论振幅是实数(但可以取负值)的情况，用它足以展示量子计算的能力和量子计算中出现的"怪相"了(也请参见习题10.5)。

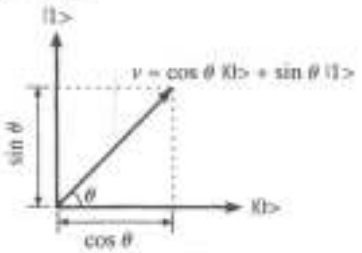
类似地，由两个量子位构成的量子系统存在四种基本态$|00\rangle$，$|01\rangle$，$|10\rangle$，$|11\rangle$，并且2-量子位系统在任何时刻的状态都可以表述为如下的叠加态：

$$\alpha_{00}|00\rangle+\alpha_{01}|01\rangle+\alpha_{10}|10\rangle+\alpha_{11}|11\rangle$$

其中$\sum_{b_1,b_2}|\alpha_{b_1b_2}|^2=1$。当这种量子系统被观测时，观测到状态$|b_1b_2\rangle$的概率为$|\alpha_{b_1b_2}|^2$。

有时，状态$|xy\rangle$也表示成$|x\rangle|y\rangle$。记号$|\cdot\rangle$在我们的讨论中是不可避免的，因为长时间以来人们已经形成了采用该记号的传统。对记号$|\cdot\rangle$心存疑惑的读者可以阅读注记10.1，它从几何角度解释了记号$|\cdot\rangle$。

注记10.1（量子态的几何表示）　有时，将量子态视为几何向量十分有益。例如，对于(具有实数振幅的)单个量子位构成的系统，两个基本态可以可视化地表示为$\mathbf{R}^2$中的两个互相垂直的单位向量$|0\rangle$，$|1\rangle$(不妨设$|0\rangle=(1,0)$，$|1\rangle=(0,1)$)。这样，系统的叠加态可以表示为$\alpha_0|0\rangle+\alpha_1|1\rangle$，它可以解释为第一个向量的$\alpha_0$倍与第二个向量的$\alpha_1$倍之和。由于$\alpha_0$，$\alpha_1$是满足$\alpha_0^2+\alpha_1^2=1$的实数，故存在唯一角度$\theta\in[0,2\pi)$使得$\alpha_0=\cos\theta$且$\alpha_1=\sin\theta$。因此，系统的叠加态可以视为$\cos\theta|0\rangle+\sin\theta|1\rangle$；亦即，叠加态是一个单位向量，它与$|0\rangle$的夹角为$\theta$而与$|1\rangle$的夹角为$\pi/2-\theta$。当系统被测量时，测得状态$|0\rangle$的概率为$\cos^2\theta$而测得状态$|1\rangle$的概率为$\sin^2\theta$。



虽然在系数为复数或量子位个数大于1时难以将叠加态可视化，但几何直观含义仍有助于量子态的推导。

例10.2　对于单量子位系统，向量$\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$和$\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle$都是合法的量子叠加态。当系统被测量时，尽管上述叠加态塌陷为$|0\rangle$和$|1\rangle$的概率都等于1/2，我们仍将它
205 们视为不同的叠加态。后面将会看到，量子操作将可以区分这两种状态。

⊖ 注意，量子力学中$\alpha_0|0\rangle+\alpha_1|1\rangle$被称为纯量子态，参见定义10.9后面的评注。

由于量子态总取单位向量，因此我们可以略去归一化因子。例如，$|0\rangle-|1\rangle$即表示量子态$\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle$。

系数相等的量子态称为均匀态。例如，2-量子位系统的均匀态是

$$|00\rangle+|01\rangle+|10\rangle+|11\rangle$$

(其中省略了归一化因子$\frac{1}{2}$)。由于前面已经假定$|x\rangle|y\rangle$表示状态$|xy\rangle$(容易验证，这种表示方法满足分配率)，因此 2-量子位系统的均匀态也可以表示为

$$(|0\rangle+|1\rangle)(|0\rangle+|1\rangle)$$

这就表明，2-量子位系统的均匀态恰由两个处于均匀态的 1-量子位系统构成。 ◀

操作量子位的状态要用量子操作来实现。量子操作是将一个量子态映射为另一个量子态的函数。本节将仅用到一个量子位上的量子操作。量子力学仅允许使用一元操作，它是保持$a_0^2+a_1^2=1$不变的线性操作。在操作具有实数系数的单个量子位时，量子操作仅仅允许如下两种情况：要么将叠加态向量相对于$\mathbf{R}^2$中的一个固定向量做反射，要么将叠加态向量旋转一个角度$\theta\in[0,2\pi)$。

10.2.1　EPR 悖论

EPR 悖论是爱因斯坦(Einstein)、波多尔斯基(Podosky)和罗森(Rosen)提出的一个佯谬[EPR35]，EPR 是三位提出者名字的缩写。EPR 悖论是一个思想实验，它证明了量子力学可以使分别处于宇宙中两个遥远角落的系统在瞬间协同它们的行为，然而爱因斯坦的狭义相对论却断言宇宙中任何东西的传播速度都无法超越光速，因此二者看上去似乎是矛盾的。爱因斯坦，这位量子力学的奠基人(他在 1905 年发表的论文中提出了光电效应)，对此感到很不满意，因此他认为必须对量子力学做适当修正以消除上述悖论。

1964 年，约翰·贝尔(John Bell)指出了如何将 EPR 思想实验变成实际实验。宇宙中相距遥远的两个系统(实际指的是 2-量子位系统)具有相同的量子态。相同的量子态使得这两个系统的行为是协同的，而这种协同在"传统的"系统上是不可能的。

此后，贝尔实验在不同场合下反复重现，总得到相同的结果：量子力学的预测是正确的，与爱因斯坦的直觉是矛盾的。现在，EPR 悖论已经不再被认为是一个悖论，因为实验涉及的两个系统并没有超光速地传递任何信息，而仅仅是在两个系统已有的公共信息上表现出相同的行为，只是共享的信息是量子态而已。下面，我们给出贝尔实验的一种形式(源自克罗斯(Clauser)等[CHSH69])。

奇偶性游戏

我们先给出一个看似与量子力学毫不相干的游戏。两名游戏参与者爱丽丝和波比被相互隔离。实验者要求他们参与如下的竞猜游戏。 206

1. 实验者随机选择两个二进制位x，$y\in_R\{0,1\}$；
2. 实验者将x交给爱丽丝，将y交给波比；
3. 爱丽丝和波比分别猜测二进制位a和b；
4. 爱丽丝和波比获胜当且仅当$a\oplus b=x\wedge y$，其中$\oplus$是异或操作(即模 2 的加法)。

注意，将两名游戏参与者相互隔离可以用狭义相对论来实现。让两名参与者相距

(比方说)1 光年，再为每名参与者配一名助理实验员。在指定的时间上，助理实验员分别随机独立地投掷硬币产生 x 和 y，并将这两个二进制位分别交给爱丽丝和波比，然后收取两名游戏参与者给出的答案，最后将所有信息发送到处于中心位置的实验者。由于爱丽丝和波比相隔 1 光年，他们无法在从接收 x，y 到给出答案这段时间内交换任何信息。

对于爱丽丝和波比而言，很容易确保以至少为 3/4 的概率获胜(比如，一直以 $a=b=0$ 作为答案)。下面证明，3/4 是游戏参与者的最佳胜率。直觉告诉我们，这是正确的，因为游戏中设置的场景使参与者根本无法协商答案；因而，两名参与者的策略是如下这样一对函数 f，g：$\{0,1\}\to\{0,1\}$；参与者给出的答案 a，b 只能是他们所看到的信息的函数；也就是说，$a=f(x)$ 而 $b=g(y)$。随机策略是所有策略上的一个分布。

定理 10.3 ([Bel64]，CHSH69)　*在奇偶性游戏中，爱丽丝和波比使用的任何策略(无论是确定策略还是随机策略)均不能使他们获胜的概率大于 3/4。*

证明　若不然，我们假设存在一个策略(可能是随机策略)使得参与者获胜的概率大于 3/4。根据标准的平均值论证法，参与者有一种固定的作出随机选择的方法使得他们以相同概率获胜。进而，不失一般性，我们假定参与者的策略是确定型的。

爱丽丝使用的函数 f：$\{0,1\}\to\{0,1\}$ 只有四种可能的情况：恒等于 0 或恒等于 1 的函数，或函数 $f(x)=x$，或函数 $f(x)=1-x$。我们分析 $f(x)=x$ 的情况，其他情况类似。于是，爱丽丝的答案 a 也就是 x；因此，参与者获胜当且仅当 $b=(x\wedge y)\oplus x$。输入为 y 时，波比需要选择一个能使他们获胜的 b。如果 $y=1$，则 $x\wedge y=x$，因而波比选择 $b=0$ 将确保爱丽丝和波比以概率 1 获胜。但是，如果 $y=0$，则 $(x\wedge y)\oplus x=x$；而波比不知道 x，因此他给出的答案 b 至多以 1/2 的概率等于 x。因此，参与者获胜的总概率至多为 3/4。■

共享量子信息的奇偶性游戏

下面证明，如果爱丽丝和波比能够共享一个 2-量子位系统(两位参与者创建处于某种状态的量子系统，在他们被带到距离 1 光年远的位置之前，他们将 2-量子位系统分裂为两个 1-量子位系统)，则他们将能够绕开定理 10.3，以大于 3/4 的概率赢得奇偶性游戏。他们的策略如下：

1. 游戏开始前，爱丽丝和波比准备一个处于叠加态 $|00\rangle+|11\rangle$ 的 2-量子位系统，这 207 种叠加态也称为 EPR 态。

2. 爱丽丝和波比分裂量子位：爱丽丝获取第一个量子位，波比获取第二个量子位。注意，量子力学并未要求 2-量子位系统的两个量子位彼此互相靠近。重要的是，爱丽丝和波比仍未测量过这两个量子位。

3. 爱丽丝收到来自实验者的随机二进制位 x。如果 $x=1$，则她将她的量子位旋转 $\pi/8$(即 22.5 度)。由于爱丽丝的操作仅涉及她自己的量子位，因此她即使在不知道波比的输入的前提下也能完成旋转操作(操作多量子位系统中的单个量子位遵循直观的直觉概念，其形式化描述参见 10.3.3 节)。

4. 波比收到来自实验者的随机二进制位 y。如果 $y=1$，则他将他的量子位旋转 $-\pi/8$(即 -22.5 度)。

5. 爱丽丝和波比都测量自己的量子位，将测得的结果分别作为答案 a，b。

注意，爱丽丝和波比执行旋转操作和测量操作的顺序无关紧要。可以证明，所有的顺序都得到相同的分布(参见习题 10.6)。倒是"分裂 2-量子位系统并在分裂后的每个量子位上执行一元操作"似乎有些天方夜谭，但是该实验已经多次被实践验证，从而验证了下面的量子力学预言。

定理 10.4 采用上述策略，爱丽丝和波比赢得奇偶性游戏的概率至少为 0.8。

证明 注意，爱丽丝和波比要想赢得游戏，他们需要在 $x=y=1$ 的情况下给出不同的答案，而在其他情况下需要给出相同的答案。证明过程的直觉是：除非 $x=y=1$，否则两个量子位的状态应该彼此非常"近"(亦即，二者之间的夹角应小于 $\pi/8$ 或 22.5 度)；而在其他情况下，两个量子位的状态则非常"远"(夹角为 $\pi/4$ 或 45 度)。具体地讲，我们将证明(记爱丽丝的答案为 a，波比的答案为 b)：

1. 如果 $x=y=0$，则 $a=b$ 的概率为 1；
2. 如果 $x\neq y$，则 $a=b$ 的概率为 $\cos^2(\pi/8)\geqslant 0.85$；
3. 如果 $x=y=1$，则 $a=b$ 的概率为 1/2。

这意味着，爱丽丝和波比获胜的总概率至少为 $\frac{1}{4}\cdot 1+\frac{1}{2}\cdot 0.85+\frac{1}{4}\cdot\frac{1}{2}=0.8$。

对于情形(1)，爱丽丝和波比都不在各自的量子位上执行操作，因此测量结果要么是 $|00\rangle$ 要么是 $|11\rangle$。两种情况下，爱丽丝和波比都将给出相同的答案。对于情形(2)，只需考虑 $x=0$，$y=1$ 的情况(另一种情况是对称的)。此时，爱丽丝不在她的量子位上执行操作，而波比则将他的量子位旋转 $-\pi/8$ 的角度。不妨设由爱丽丝先测量她的量子位，然后波比旋转量子位再进行测量(这种假设没有关系，因为操作顺序不影响他们给出的答案)。爱丽丝给出答案 0 的概率为 1/2；波比的量子位旋转 $-\pi/8$ 之后将得到状态 $|0\rangle$，这意味着波比测量量子位时得到答案 0 的概率为 $\cos^2(\pi/8)$。类似地，如果爱丽丝给出的答案是 1，则波比得到答案 1 的概率仍为 $\cos^2(\pi/8)$。 208

为了分析情形(3)，我们直接进行计算。此时，当两位参与者的旋转操作都完成之后，2-量子位系统进入状态

$$\begin{aligned}&(\cos(\pi/8)|0\rangle+\sin(\pi/8)|1\rangle)(\cos(\pi/8)|0\rangle-\sin(\pi/8)|1\rangle)\\&+(-\sin(\pi/8)|0\rangle+\cos(\pi/8)|1\rangle)(\sin(\pi/8)|0\rangle+\cos(\pi/8)|1\rangle)\\=\;&(\cos^2(\pi/8)-\sin^2(\pi/8))|00\rangle-2\sin(\pi/8)\cos(\pi/8)|01\rangle\\&+2\sin(\pi/8)\cos(\pi/8)|10\rangle+(\cos^2(\pi/8)-\sin^2(\pi/8))|11\rangle\end{aligned}$$

但由于

$$\cos^2(\pi/8)-\sin^2(\pi/8)=\cos(\pi/4)=\frac{1}{\sqrt{2}}=\sin(\pi/4)=2\sin(\pi/8)\cos(\pi/8)$$

故该量子态的所有系数具有相同的绝对值；进而测量这个 2-量子位系统时，测得 00，01，10，11 中的任何一个状态的概率都等于 1/4。 ■

常数 0.8 可以用某种方法进一步提高，参见习题 10.1。同时，在另一些已知的游戏上，参与者获胜的概率在传统场景中和在量子场景中的差别甚至更大。一个有意义的转折是，EPR 实验和贝尔实验背后的思想近年来已经在实践中被用于产生量子加密方案，其安全性仅依赖于量子力学的原理，而不再依赖于 **P≠NP** 这种未经证明的猜想(参见本章注记)。

10.3　量子计算的定义和 BQP

本节介绍量子操作，由此定义量子门、量子计算和 **BQP**——由存在高效量子判定算法的语言构成的复杂性类。

10.3.1　线性代数预备知识

本章将用到空间 $\mathbf{C}^M$ 的几个基本事实和概念。这些知识在 A.5 节进行了概述，这里仅做简单回顾。

- 如果 $z=a+ib$ 是复数(其中 $i=\sqrt{-1}$)，则 $\bar{z}=a-ib$ 是 z 的共轭复数。注意，$z\bar{z}=a^2+b^2=|z|^2$。
- 向量 u，$v\in\mathbf{C}^M$ 的内积，记为 $\langle u,v\rangle$，等于 $\sum_{x\in[M]}u_x\bar{v}_x$。[⊖]
- 向量 u 的范数，记为 $\|u\|_2$，等于 $\sqrt{\langle u,u\rangle}=\sqrt{\sum_{x\in[M]}|u_x|^2}$。
- 如果 $\langle u,v\rangle=0$，则称 u 与 v 正交。
- $\mathbf{C}^M$ 的向量集 $\{v^i\}_{i\in[M]}$ 称为 $\mathbf{C}^M$ 的正交基，如果对于任意 $i,j\in[M]$ 有：若 $i=j$ 则 $\langle u^i,v^j\rangle$ 等于 1，若 $i\neq j$ 则 $\langle u^i,v^j\rangle$ 等于 0。

209

- 如果 A 是一个 $M\times M$ 的矩阵，则 A^* 表示 A 的共轭转置：亦即，$A^*_{x,y}=\overline{A_{y,x}}$ 对任意 $x,y\in[M]$ 成立。
- $M\times M$ 的矩阵 A 称为酉阵，如果 $AA^*=I$，其中 I 是 $M\times M$ 的单位矩阵。

注意，如果 z 是实数(即 z 没有虚部)，则 $\bar{z}=z$。因此，如果所有向量和矩阵都只使用实数，则内积就是实数空间 $\boldsymbol{R}^n$ 上的标准内积，而共轭转置就是标准转置。此外，对于实数向量 u，v，有 $\langle u,v\rangle=\cos\theta\|u\|_2\|v\|_2$，其中 θ 是 u 和 v 之间的夹角。

下面的论断(留作习题 10.2)总结了酉阵的性质。

论断 10.5　对于 $M\times M$ 的任意复数矩阵 A，下列条件等价：

1. A 是酉阵(亦即 $AA^*=I$)；
2. $\|Av\|_2=\|v\|_2$ 对任意向量 $v\in\mathbf{C}^M$ 成立；
3. 对于 $\mathbf{C}^M$ 的任意正交基 $\{v^i\}_{i\in[M]}$，$\{Av^i\}_{i\in[M]}$ 也是 $\mathbf{C}^M$ 的正交基；
4. A 的所有列构成 $\mathbf{C}^M$ 的一个正交基；
5. A 的所有行构成 $\mathbf{C}^M$ 的一个正交基。

10.3.2　量子寄存器及其状态向量

在标准的数字计算机中，内存中的一个二进制位被实现为具有“开”和“关”这两种状态的物理元件，“开”状态表示为 1 而“关”状态表示为 0。将 m 个这样的元件整合在一起就得到 m-位的寄存器，寄存器的每种状态可以表示为 $\{0,1\}^m$ 中的一个字符串。量子寄存器由 m 个量子位构成，它的状态是 2^m 个基本态的叠加，其中每个基本态指的就是 10.1 节的“概

[⊖] 某些量子计算的教科书也用 $\sum_{x\in[M]}\bar{u}_xv_x$。

率波"；因此，量子寄存器的每个状态可以表示为一个向量 $v=(v_{0^n}, v_{0^{n-1}1}, \cdots, v_{1^n}) \in \mathbf{C}^{2^n}$，其中 $\sum_x |v_x|^2 = 1$。由量子力学可知，每当测量寄存器(亦即读取寄存器的值)时，得到值 x 的概率等于 $|v_x|^2$，同时寄存器的状态将塌陷为状态 $|x\rangle$；也就是说，满足 $y \neq x$ 的状态 $|y\rangle$ 的系数将变成 0。从原理上讲，这种量子寄存器可以用任意 m 个具有"开"和"关"状态的物理元件实现，但在实践上这种实现却面临着巨大的困难。

10.3.3　量子操作

下面，我们定义符合量子力学原理的操作。

定义 10.6（量子操作）　m-量子位寄存器上的一个量子操作是一个函数 $F: \mathbf{C}^{2^m} \to \mathbf{C}^{2^m}$，它能将寄存器的前一种状态映射为一种新状态，并满足下列条件：

线性：F 是一个线性函数。亦即，对任意 $v \in \mathbf{C}^{2^m}$ 有 $F(v) = \sum_x v_x F(|x\rangle)$。

保范数性：F 将单位向量映射为单位向量。亦即，对任意满足 $\|v\|_2 = 1$ 的向量均有 $\|F(v)\|_2 = 1$。

第二个条件(保范数性)是顺理成章的，因为只有单位向量才能表示量子态。线性条件是量子力学理论的要求。两个条件放在一起，意味着每个量子操作 F 都可以表示为一个 $2^m \times 2^m$ 的酉阵。由此即得如下引理。

引理 10.7（量子操作的复合）　如果矩阵 A_1，A_2 表示两个量子操作，则两个量子操作的复合(即执行 A_1 之后再执行 A_2)也是一个量子操作，其矩阵表示为 A_2A_1。

特别地，由于 $A_1A_1^\dagger = I$，因此每个量子操作都存在逆操作来消除原量子操作的影响；亦即，量子操作是"可逆的"。

由于量子操作具有线性性质，因此只需描述量子操作在线性空间 $\mathbf{C}^{2^m}$ 的任意基底上的行为。于是，在我们描述量子操作时，通常只给出它在标准基底上的操作结果。但是，每个传统的线性变换并不一定是酉变换，因此在设计量子操作时需要格外小心。

10.3.4　量子操作实例

下面给出量子操作的一些实例。

翻转量子位。如果我们希望将 m-量子位寄存器的第一个量子位"翻转"，亦即在第一个量子位上执行 NOT 操作，则可以使用如下的量子操作：将基态 $|b, x\rangle$(其中 $b \in \{0, 1\}$，$x \in \{0, 1\}^{m-1}$)变成基态 $|1-b, x\rangle$。该量子操作的矩阵是标准基的置换矩阵，而置换矩阵都是酉阵。关于量子操作的重要注记：本例涉及的翻转操作仅操作第一个量子位，因此 x 中的其他量子位不受影响，进而不必在讨论翻转操作时提及它们。此后，每当我们讨论仅涉及部分量子位的操作时，将略去操作中不受影响的量子位。继而，NOT 操作可以描述为 $|0\rangle \mapsto |1\rangle$ 和 $|1\rangle \mapsto |0\rangle$。

重排量子位。如果想交换两个量子位的值，则可以用以下量子操作：$|10\rangle \mapsto |01\rangle$，$|01\rangle \mapsto |10\rangle$，而 $|00\rangle$ 和 $|11\rangle$ 映射到它们自身。同样，该量子操作也是标准态的置换，其矩阵是如下 $2^2 \times 2^2$ 的置换酉阵(矩阵所有行和所有列的编号顺序是 $|00\rangle$，$|01\rangle$，$|10\rangle$，$|11\rangle$ 的字典顺序)：

210

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

注意，恰当地组合运用上述操作，可以将 m-量子位寄存器的所有量子位按任意顺序重排。

复制量子位。现在，假设我们希望将第一个量子位的值复制到第二个量子位上。首先考虑如下简单的想法：将 $|10\rangle$ 和 $|11\rangle$ 都映射为 $|11\rangle$，而将 $|00\rangle$ 和 $|01\rangle$ 都映射为 $|00\rangle$。但是，该操作不是可逆的，进而它也不是一个酉操作！事实上，不可克隆定理(no cloning theorem)排除了存在复制量子位这种量子操作的可能性；参见本章注记。然而，对量子算法设计而言，通常仅需以"一次地写出"的方式复制量子位，这可以通过如下方式实现：假设有一些处于特定状态(不妨设是状态 $|0\rangle$)的量子位，则源量子位的状态可以"一次地写出"到这样的量子位中。事实上，只要量子算法设计者能够确保操作之前第二个量子位处于状态 $|0\rangle$ 而从未被其他操作使用过，则量子操作 $|xy\rangle \mapsto |x(x \oplus y)\rangle$ 的结果绝不会等于 $|01\rangle$ 或 $|10\rangle$，而只能是 $|00\rangle$ 或 $|11\rangle$。注意，从另一个角度看，上述操作相当于"当且仅当第一个量子位处于状态 $|1\rangle$ 时，对第二个量子位执行 NOT 操作"。正是由于这个原因，文献中将该量子操作称为受控 NOT 操作，也简称 CNOT 操作。

211

旋转单个量子位。正如注记 10.1 所述，量子位的状态可以视为一个二维向量，于是我们可以将状态向量旋转角度 θ。这就是如下的旋转操作：$|0\rangle \mapsto \cos\theta|0\rangle + \sin\theta|1\rangle$，且 $|1\rangle \mapsto -\sin\theta|0\rangle + \cos\theta|1\rangle$；旋转操作可以表示为酉阵 $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$。注意，当 $\theta = \pi$(即 180 度)时，旋转操作相当于改变状态向量的符号(亦即，映射 $v \mapsto -v$)。

两个量子位的 AND 操作。下面，考虑传统的 AND 操作，也就是将寄存器的第一个量子位替换为前两个量子位执行 AND 操作后的结果。你或许会认为该操作可以通过线性变换 $|b_1 b_2\rangle \mapsto |b_1 \wedge b_2\rangle|b_2\rangle$ 来完成，其中 $b_1, b_2 \in \{0, 1\}$；然而，上述变换不是可逆变换，因而也不是酉变换。

尽管如此，AND 操作仍可以通过其他方法实现。该方法得到一个"可逆的 AND 操作"，它通过使用一个额外的未使用过的量子位 b_3 将 AND 操作实现为变换 $|b_1\rangle|b_2\rangle|b_3\rangle \mapsto |b_1\rangle|b_2\rangle|b_3 \oplus (b_1 \wedge b_2)\rangle$，其中 $b_1, b_2, b_3 \in \{0, 1\}$。该变换是一个酉变换，因而是一个合法的量子操作，它的矩阵是如下的酉置换矩阵：

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

同前面的情况一样，算法设计者在使用 AND 操作时必须确保 b_3 是处于状态 $|0\rangle$ 的未使用过的量子位。该操作在量子计算中也就是著名的 Toffoli 量子门。我们也可以类似地得到"可逆的 OR 操作"这种量子操作。在证明量子计算机可以模拟普通图灵机时(参见第

10.3.7 节），可逆 OR 门和可逆 AND 门将发挥关键作用。

哈达玛操作（Hadamard Operation）。哈达玛门是作用在单个量子位上的操作，（在归一化操作下）它将 $|0\rangle$ 映射到 $|0\rangle+|1\rangle$，而将 $|1\rangle$ 映射到 $|0\rangle-|1\rangle$；将它表示为更紧凑的形式，也就是将状态 $|b\rangle$ 映射到 $|0\rangle+(-1)^b|1\rangle$。该操作对应于矩阵 $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$。

注意，如果将哈达玛门应用到 m-量子位寄存器的每个量子位上，则对于任意 $x\in\{0,1\}^m$，量子态 $|x\rangle$ 将被变换为

$$\begin{aligned}&(|0\rangle+(-1)^{x_1}|1\rangle)(|0\rangle+(-1)^{x_2}|1\rangle)\cdots(|0\rangle+(-1)^{x_m}|1\rangle)\\&=\sum_{y\in\{0,1\}^m}\Big(\prod_{i,y_i=1}(-1)^{x_i}\Big)|y\rangle=\sum_{y\in\{0,1\}^m}(-1)^{x\odot y}|y\rangle\end{aligned}\tag{10.1}$$

212

其中 $x\odot y$ 表示 x 和 y 对模 2 的点积。这个操作对应一个 $2^m\times 2^m$ 的酉阵，其中 (x,y) 位置的元素为 $\frac{-1^{x\odot y}}{\sqrt{2^m}}$（将 $[2^m]$ 等同于 $\{0,1\}^m$）。该操作在量子算法中起着重要作用。㊀

10.3.5　量子计算与 BQP

虽然量子力学原理允许在量子寄存器的当前状态上执行任意一个酉阵表示的量子操作，但并非所有这些操作都可以被物理实现。事实上，只有“局部化”程度较高的操作（亦即仅操作有限个量子位的操作）才有望被物理实现。因此，这样的操作被定义为量子计算的基本操作。

定义 10.8（基本量子操作和量子门）　如果一个量子操作在寄存器上操作的量子位不超过 3 个，则称该操作为基本量子操作，有时也称为量子门。㊁

注意，m 量子位寄存器上的基本操作可以用三个取自 $[m]$ 的标号和一个 8×8 的酉阵来描述。例如，如果 U 是一个 8×8 的酉阵并且它只能作用于第 2，3，4 个量子位上，则 U 可以视为一个基本量子操作 $F:\mathbf{C}^{2^m}\to\mathbf{C}^{2^m}$，它将基态 $|x_1x_2\cdots x_m\rangle$ 变换为量子态 $|x_1\rangle(U|x_2x_3x_4\rangle)|x_5\cdots x_m\rangle$，其中 $x_1,x_2,\cdots,x_m\in\{0,1\}$。

现在，我们可以将量子计算定义为量子寄存器上执行的一系列基本操作。

定义 10.9（量子计算和类 BQP）　设 $f:\{0,1\}^*\to\{0,1\}$ 和 $T:\mathbf{N}\to\mathbf{N}$ 是两个函数。我们称 f 是量子 $T(n)$ 时间可计算的，如果存在一个多项式时间的传统图灵机，对任意 $n\in\mathbf{N}$，该图灵机能够在输入 $(1^n,1^{T(n)})$ 上输出量子门 $F_1,\cdots,F_{T(n)}$ 的酉阵使得下面的计算过程在任意 $x\in\{0,1\}^n$ 上将至少以 2/3 的概率计算得到 $f(x)$：

1. 将 m 量子位的量子寄存器初始化为状态 $|x0^{m-n}\rangle$（亦即 x 填充若干个 0），其中 $m\leqslant T(n)$。

2. 将 $T(n)$ 个基本量子操作 $F_1,\cdots,F_{T(n)}$ 依次作用到寄存器上。

3. 测量量子寄存器得到一个值 Y（也就是说，如果 v 是量子寄存器的最终状态，则 Y 是一个随机变量，它取任意 $y\in\{0,1\}^m$ 的概率等于 $|v_y|^2$。）

4. 输出 Y_1。

㊀ 第 11 章和第 19 章描述 Walsh-Hadamard 纠错码时还会遇到上面的矩阵。（不过，那时它将变成 GF(2) 上的 0/1 矩阵，而不再是 $\mathbf{C}$ 上的 ±1 矩阵。）

㊁ 常数 3 是任意的。事实上，把它修改为任意大于等于 2 的常数将得到计算能力等价的模型。

布尔函数 $f:\{0,1\}^* \to \{0,1\}$ 属于 **BQP**，如果存在多项式 $p:\mathbf{N}\to\mathbf{N}$ 使得 f 是量子
213 $p(n)$时间可计算的。

关于量子计算的定义，下面再给出一些评注。

1. 上述定义很容易推广，以计算那些输出不止一个比特位的函数。

2. 所有基本操作都被表示成 8×8 的复数矩阵，传统的图灵机无法直接输出这种矩阵本身。然而，图灵机只需输出每个复数的前 $O(\log T(n))$ 个比特位就足矣，参见习题 10.8。

3. 虽然基本量子操作或量子门有无穷个，但可以证明这些操作中仅需保留两个通用操作而不影响操作的一般性，参见 10.3.8 节。

4. 熟悉量子力学或量子计算的读者可能已经注意到，我们给出的定义不允许使用量子力学中允许使用的几个性质。例如，不允许混态，它既会涉及量子叠加，也会涉及在标准基之外的其他基上的测量和概率。但是，这些性质都不会给量子计算机增加新的计算能力。此外，我们不允许的另一个性质是，完成计算的过程中不允许对一部分量子位进行测量(也称为部分测量)。虽然部分测量可以为算法描述带来一些方便，但习题 10.7 将证明消除算法中的部分测量不会损失太大的计算效率。

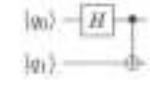
量子计算 Vs. 概率计算

至此，读者或许会认为，量子计算模型"显然"将计算速度提升了指数倍，因为其状态寄存器是 2^m 维的向量，并且操作也是 $2^m\times 2^m$ 的矩阵。但是，这并不是计算能力增加的真正原因。事实上，普通的概率计算也可以用这种方式来描述：传统的 m 位寄存器可以视为 2^m 维的向量，它的第 x 维坐标等于寄存器中恰好出现串 x 的概率；这样，概率型操作就可以视为从 $\mathbf{R}^{2^m}$ 到 $\mathbf{R}^{2^m}$ 的随机线性映射(参见习题 10.4)。量子计算的计算能力增加的真正原因或许在于，我们允许向量取负系数(参见本章开头费曼的引言)并且每个计算步骤中保持的范数是欧几里得范数(即 ℓ_2 范数)而不是和范数(即 ℓ_1 范数)；参见习题 10.5。注意，在 10.3.7 节中我们将看到，传统的计算(不管是概率型计算还是确定型计算)是量子计算的特例。

10.3.6　量子线路

定义 10.9 让我们联想到传统的直线程序，第 6 章已经看到直线程序是布尔线路的等价模型(参见注记 6.4)。类似地，我们也可以用量子线路来定义量子计算和 **BQP**：事实上，大多数教科书都采用了这种定义。类似于布尔线路，量子线路是无环有向图，其中源顶点(即入度为 0 的顶点)表示输入，汇顶点(出度为 0 的顶点)表示输出，中间顶点表示量子门。区别之一是，量子门不再标记为 AND，OR 或 NOT 操作，而是标记为 2×2，4×4
214 或 8×8 的酉阵。另一个区别是，(由于量子位不存在复制操作)量子门的出度和输入顶点的出度都不能再取任意大的值；相反，输入顶点的出度只能等于 1，量子门的入度和出度必须相等(且至多为 3)。此外，量子线路还允许使用初始状态为 $|0\rangle$ 的特殊"工作空间"或"演算空间"输入。

量子线路在各种文献中通常被描述为如下的流图：在输入 $|q_0\rangle$ 和 $|q_1\rangle$ 上，下图给出的量子线路首先在 $|q_0\rangle$ 上运用哈达玛操作，再运用变换 $|q_0q_1\rangle\mapsto|q_0(q_0\oplus q_1)\rangle$。



10.3.7　传统计算是量子计算的特例

在 10.3.4 节中，我们看到了传统的 NOT 操作和 AND 操作的量子实现。更一般地，任意传统操作均可以用量子操作来高效地模拟。

引理 10.10（布尔线路是量子线路的特例）　如果 $f:\{0,1\}^n\to\{0,1\}^m$ 可以由一个规模为 S 的布尔线路计算，则映射 $|x\rangle|0^{s+m+n}\rangle\mapsto|x\rangle|f(x)\rangle|0^{s+n}\rangle$ 可以由一个包含 $2S+m+n$ 个基本量子操作的操作序列来计算。

证明　将布尔线路中的每个布尔门(AND，OR，NOT)替换为 10.3.4 节介绍的相应的量子门。所得到的量子线路在输入 $|x\rangle|0^m\rangle|0^s\rangle$ 上计算得到 $|x\rangle|f(x)0^m\rangle|z\rangle$，其中 z 是量子线路的所有中间顶点的取值构成的串(这些中间顶点对应于在各个量子门上完成量子计算时使用的用于演算的内存)，而 0^m 是目前仍未使用的 m 个量子位。现在，利用形如 $|bc\rangle\to|b(b\oplus c)\rangle$ 的 m 个操作将 $f(x)$ 复制到串 0^m 上。接下来，执行所有量子操作的逆操作。这样就可以擦除原有的 $f(x)$ 和 z 而使相应的量子位处于状态 $|0\rangle$，最后仅保留了一个 $f(x)$。 ■

由于传统图灵机在 $T(n)$ 个步骤内完成的计算也可以用规模为 $O(T(n)\log T(n))$ 的布尔线路完成，因此 **P**⊑**BQP**。利用哈达玛操作将 $|0\rangle$ 映射为 $|0\rangle+|1\rangle$，我们可以得到一个量子位，其测量值为 $|0\rangle$ 的概率等于 1/2 且测量值为 $|1\rangle$ 的概率也等于 1/2，因此量子位可以用来模拟硬币投掷。于是，我们立刻得到下面的推论。

推论 10.11　**BPP**⊑**BQP**。

10.3.8　通用操作

由于在 3 个量子位上存在无穷多个量子操作，因此将它们都视为基本量子操作似乎会引起一些问题。相比之下，传统布尔线路仅使用三种逻辑门(AND，OR，NOT)。值得庆幸的是，类似的结果对量子计算也成立。下面的定理表明(证明从略)，仅用少数几个量子操作就可以构造出任意的量子操作。 215

定理 10.12（量子操作的通用基[Deu89，Kit97]）　对任意 $D\geqslant 3$ 和 $\varepsilon>0$，存在 $\ell\leqslant 100(D\log 1/\varepsilon)^3$ 使得如下结论成立。任意 $D\times D$ 的酉阵 U 都可以近似地表示成酉阵 $U_1,\cdots,U_\ell$ 的乘积，其中近似的含义指的是：对任意 $i,j\leqslant D$ 而言，矩阵中 (i,j) 位置上的元素都满足

$$|U_{ij}-(U_\ell U_{\ell-1}\cdots U_1)_{ij}|<\varepsilon$$

并且每个 U_i 都是三个量子位上的哈达玛门 $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$，Toffoli 门 $|abc\rangle\mapsto|ab(c\oplus a\wedge b)\rangle$ 或相位偏移门 $\begin{pmatrix}1 & 0\\ 0 & \mathrm{i}\end{pmatrix}$ 之一。

可以证明，令 $\varepsilon<\frac{1}{10T}$，则 ε-近似足以模拟任意 T 时间量子计算(参见习题 10.8)；也就是说，由 T 个基本量子操作完成的任意计算也可以用上面提到的三种量子门来完成。人们还得到了其他一些通用量子门；特别地，施尧耘[Shi03]证明了哈达玛门和 Toffoli 门这两种量子门足以实现量子计算，其证明过程用到了如下事实：量子计算中复数不是必需的(参见习题 10.5)。

定理 10.12 的一个推论是，如果 $k>3$ 是任意的常数，则 3 量子位的量子门可以用来模拟 k 量子位的量子门，只是模拟过程的时间开销是 k 的指数，因为模拟过程要使用 $2^k\times$

2^k的矩阵。这意味着，当我们设计量子算法时，只要k小于一个绝对常数，则k量子位的量子门可以看做一个基本操作。基于这一事实，我们可以在量子计算中得到类似于传统编程语言的"If cond then"结构的操作。也就是说，如果n量子位上量子操作U用T步量子线路来实现，则我们可以在$O(T)$个步骤内实现一个受控的U量子操作。在$x_{n+1}=1$的条件下，受控的U量子操作将向量$|x_1\cdots x_n x_{n+1}\rangle$映射为$|U(x_1\cdots x_n)x_{n+1}\rangle$；在$x_{n+1}=0$时，该操作将向量$|x_1\cdots x_n x_{n+1}\rangle$映射为向量自身。受控的$U$量子操作能被实现，这是因为完成$U$操作的过程中用到的每个基本操作$F$都可以类似地实现为"受控的$F$操作"。由于每个受控的$F$操作都仅依赖于4个量子位，因此它也可以被视为基本操作。

10.4　格罗弗搜索算法

现在，我们给出格罗弗算法，它是量子计算机上的基本算法，也是很有用的算法。本节独立于10.5节和10.6节，其中10.5节给出西蒙算法而10.6节给出肖尔算法。因此，对于急于了解整数分解算法的读者，可以跳过本节直接进入10.5节。

考虑**NP**完全问题SAT，其输入是n个变量的布尔公式φ，要求判定是否存在赋值
216 $a\in\{0,1\}^n$满足$\varphi(a)=1$。在"传统的"确定型图灵机或概率型图灵机上，人们迄今为止为SAT问题找到的最好算法就是时间复杂度为$\mathrm{poly}(n)2^n$的平凡算法。[⊖]格罗弗给出了量子计算机上求解SAT问题的时间复杂度为$\mathrm{poly}(n)2^{n/2}$的算法。相对于传统算法而言，这是一个重大改进，同时也简洁地证明了**NP**$\subseteq$**BQP**。事实上，格罗弗算法求解了更具一般性的问题：亦即，判定具有n个输入的线路的可满足性。

定理10.13（格罗弗算法[Gro96]）　存在一个量子算法，在输入的任意多项式时间可计算函数$f:\{0,1\}^n\to\{0,1\}$（亦即表示为计算f的一个线路）上，该算法能够在$\mathrm{poly}(n)2^{n/2}$时间内输出一个满足$f(a)=1$的串a（如果这样的串存在的话）。

证明　描述格罗弗算法的最佳方式是使用几何术语。我们假设函数f存在唯一的满足性赋值a（17.4.1节介绍的技术可以将一般的问题转换成这种形式）。考虑n量子位寄存器，令$\boldsymbol{u}$表示该寄存器的均匀状态向量，亦即$\boldsymbol{u}=\frac{1}{2^{n/2}}\sum_{x\in\{0,1\}^n}|x\rangle$。向量$\boldsymbol{u}$与基态$|a\rangle$的夹角等于二者内积$\langle\boldsymbol{u},|a\rangle\rangle=\frac{1}{2^{n/2}}$的反余弦。由于$\frac{1}{2^{n/2}}$是正数，因此$\boldsymbol{u}$与$|a\rangle$的夹角小于$\pi/2$（即90度），进而我们可以将这个夹角记为$\pi/2-\theta$，其中$\sin\theta=\frac{1}{2^{n/2}}$。于是，由于不等式$\theta\geqslant\sin\theta$对任意$\theta>0$成立，故$\theta>2^{-n/2}$。

格罗弗算法从状态$\boldsymbol{u}$开始，通过每个操作步骤逐渐靠近状态$|a\rangle$。如果当前状态与$|a\rangle$的夹角为$\pi/2-\alpha$，则经过一个操作步骤之后状态与$|a\rangle$的夹角将变为$\pi/2-\alpha-2\theta$。因此，经过$O(1/\theta)=O(2^{n/2})$个步骤之后，算法得到的状态$\boldsymbol{v}$与$|a\rangle$的内积大于1/2，这意味着，此时测量寄存器得到$|a\rangle$的概率至少为1/4。

实现每个步骤的主要思想是，旋转当前状态向量$\boldsymbol{w}$使得它与未知向量$|a\rangle$的夹角比旋转之前它与$|a\rangle$的夹角小2θ；该操作仅需通过将$\boldsymbol{w}$连续地绕向量$\boldsymbol{u}$和向量$\boldsymbol{e}=\sum_{x\neq a}|x\rangle$做两次反射来实现。注意，$\boldsymbol{e}=\sum_{x\neq a}|x\rangle$是$\boldsymbol{u}$和$|a\rangle$张成的平面中垂直于$|a\rangle$的向量；参见图10-3

⊖ 问题的特殊形式3SAT存在复杂性略低的算法。

给出的“证明示意图”。

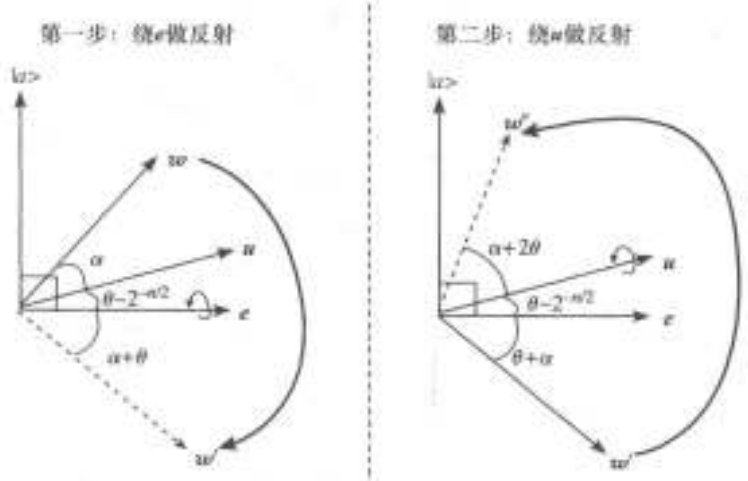


图 10-3　在 $|a\rangle$ 和 u 张成的平面中通过两次反射将向量 w 变换为向量 w' 使得 w' 与 $|a\rangle$ 的夹角比 w 与 $|a\rangle$ 的夹角小 2θ。首先，将 w 绕 $e=\sum_{x\neq a}|x\rangle$ 做反射（注意 e 在该平面上垂直于 $|a\rangle$），再绕 u 做反射。如果 w 原来与 $|a\rangle$ 的夹角为 $\pi/2-\theta-\alpha$，则 w' 与 $|a\rangle$ 的夹角为 $\pi/2-\theta-\alpha-2\theta$。可以将关注的焦点定位在 $|a\rangle$ 和 u 张成的平面上，这是因为反射操作使得正交于该平面的所有向量保持不动

为完整地给出算法，只需说明如何将 w 连续地绕向量 u 和向量 e 做反射。也就是说，我们需要说明如何将寄存器的当前状态 w 变换成另一个状态，使得新状态是 w 绕向量 u（或 e）做反射得到的向量。然而，我们并不直接在 n 量子位寄存器上完成上述操作，而是在 m 量子位寄存器上完成上述操作，其中 m 是 n 的多项式。这些额外的量子位仅用作“演算空间”，它们将一直保持状态 $|0\rangle$，除非计算过程中需要使用它们（借助引理 10.10 中“清零”的思想，使用过的演算空间可以还原为状态 $|0\rangle$），因此这些额外的量子位可以忽略。

绕 e 做反射

回顾一下向量 w 绕向量 v 的反射，如果将 w 分解为 $\alpha v+v^{\perp}$（其中 $v^{\perp}$ 与 v 正交），则反射操作将输出 $\alpha v-v^{\perp}$。因此，w 绕 e 的反射等于向量 $\sum_{x\neq a}w_x|x\rangle-w_a|a\rangle$。这种变换很容易如下实现。

1. 由于 f 是多项式时间可计算的，我们可以在多项式时间内计算变换 $|x\sigma\rangle\mapsto|x(\sigma\oplus f(x))\rangle$（上述记号忽略了计算过程需要“演算”的额外量子位，我们不列出这些量子位）。在 $x\neq a$ 时，上述变换将 $|x0\rangle$ 变换为 $|x0\rangle$，并将 $|a1\rangle$ 变换为 $|a0\rangle$。

2. 然后，在量子位 σ 上运用量子基本变换（即著名的 Z 门）$|0\rangle\mapsto|0\rangle$，$|1\rangle\mapsto-|1\rangle$。这样，当 $x\neq a$ 时，$|x0\rangle$ 被映射为 $|x0\rangle$，并且 $|a1\rangle$ 被映射为 $-|a1\rangle$。

3. 最后，再次运用变换 $|x\sigma\rangle\mapsto|x(\sigma\oplus f(x))\rangle$。在 $x\neq a$ 时，$|x0\rangle$ 被映射为 $|x0\rangle$，并且 $|a1\rangle$ 被映射为 $|a0\rangle$。

上述 3 个步骤的最终效果是，当 $x\neq a$ 时，$|x0\rangle$ 被映射为 $|x0\rangle$，并且 $|a0\rangle$ 被映射为 $-|a0\rangle$。忽略上述变换中的第二个量子位，这恰好实现了绕 e 的反射。

绕 u 做反射

为了绕 u 做反射，我们首先将哈达玛操作运用到每个量子位上，将 u 映射为 $|0\rangle$。然

217

后，绕$|0\rangle$做反射(这可以用类似于绕$|a\rangle$做反射的方法来完成；为此，只需将f替换为函数$g:\{0,1\}^n\to\{0,1\}$，它输出1当且仅当其输入全部为0)。最后，再次运用哈达玛操作将$|0\rangle$变回u。

将上述操作放在一起，则我们能够在$|a\rangle$和u张成的平面上将向量旋转2θ弧度到达距离$|a\rangle$更近的位置。因此，如果从向量u开始，则仅需重复$O(1/\theta)=O(2^{n/2})$次旋转操作就可以获得一个向量，使得测量该向量得到$|a\rangle$的概率为常数。

这就完成了定理10.13的证明。为使论述更加完备，图10-4完整地给出了格罗弗
218 算法。■

目标：给定多项式时间可计算的函数$f:\{0,1\}^n\to\{0,1\}$并且存在唯一的$a\in\{0,1\}^n$满足$f(a)=1$，目标是找出a

量子寄存器：使用一个$n+1+m$量子位的寄存器，其中m充分大。它使得寄存器足以计算变换$|xz0^m\rangle\mapsto|x(z\oplus f(x))0^m\rangle$

操作	状态(忽略归一化因子)
	初始化状态：$\|0^{n+m+1}\rangle$
将哈达玛操作运用到前n个量子位	$u\|0^{n+1}\rangle$(其中u表示$\sum_{x\in\{0,1\}^n}\|x\rangle$)
	$v^1\|0^{n+1}\rangle$
For $i=1,\cdots,2^{n/2}$ do	令$v^i=u$并维护不变量$\langle v^i,\|a\rangle\rangle=\sin((i\theta)$，其中$\theta\sim 2^{-n/2}$是满足$\langle u,\|a\rangle\rangle=\sin(\theta)$的角
第1步：绕$e=\sum_{x\neq a}\|x\rangle$做反射：	
1.1 计算$\|xz0^m\rangle\mapsto\|x(z\oplus f(x))0^m\rangle$	$\sum_{x\neq a}v^i_x\|x\rangle\|0^{n+1}\rangle+v^i_a\|a\rangle\|1^m\rangle$
1.2 若$z=1$，则将向量乘以-1； 否则，不做任何操作	$\sum_{x\neq a}v^i_x\|x\rangle\|0^{n+1}\rangle-v^i_a\|a\rangle\|1^m\rangle$
1.3 计算$\|xz0^m\rangle\to\|x(z\oplus f(x))0^m\rangle$	$w^i\|0^{m+1}\rangle-\sum_{x\neq a}v^i_x\|x\rangle\|0^{n+1}\rangle-v^i_a\|a\rangle\|00^m\rangle$ (w^i是v^i绕$\sum_{x\neq a}\|x\rangle$的反射)
第2步：绕u做反射：	
2.1 将哈达玛操作运用到前n个量子位	$\langle w^i,u\rangle\|0^n\rangle\|0^{m+1}\rangle+\sum_{x\neq 0^n}a_x\|x\rangle\|0^{m+1}\rangle$，其中系数$a_x=\sum_z(-1)^{x\cdot z}w^i_z\|z\rangle$。
2.2 绕$\|0\rangle$做反射	
2.2.1 如果前n个量子位全部等于0，则将第$n+1$个量子位翻转	$\langle w^i,u\rangle\|0^n\rangle\|1^m\rangle+\sum_{x\neq 0^n}a_x\|x\rangle\|0^{m+1}\rangle$
2.2.2 如果第$n+1$个量子位是1，则乘-1	$-\langle w^i,u\rangle\|0^n\rangle\|1^m\rangle+\sum_{x\neq 0^n}a_x\|x\rangle\|0^{m+1}\rangle$
2.2.3 如果前n个量子位全部等于0，则将第$n+1$个量子位翻转	$-\langle w^i,u\rangle\|0^n\rangle\|0^{m+1}\rangle+\sum_{x\neq 0^n}a_x\|x\rangle\|0^{m+1}\rangle$
2.3 将哈达玛操作运用到前n个量子位	$v^{i+1}\|0^{m+1}\rangle$(v^{i+1}是w^i绕u的反射)
测量寄存器，令a'是前n个量子位上测得的值。如果$f(a')=1$，则输出a'；否则，重复执行算法。	

图10-4　格罗弗算法

10.5　西蒙算法

尽管格罗弗算法很漂亮，但它有一个严重的缺陷：它的运行时间仅仅是求解同一问题的最佳传统算法的运行时间的平方根。相比之下，本节介绍的西蒙算法是一个多项式时间量子算法，而求解同一问题的最佳传统算法却具有指数时间的复杂度。 219

西蒙问题

输入：函数 $f:\{0,1\}^n\to\{0,1\}^n$ 的一个多项式规模的传统线路，其中 f 满足如下条件：存在 $a\in\{0,1\}^n$ 使得“$f(x)=f(y)$ 当且仅当 $x=y\oplus a$ 对任意 $x,y\in\{0,1\}^n$ 成立”。

求解目标：找出串 a（它也被称作 f 的“周期”）

定理 10.14（西蒙算法[Sim94]）　存在求解西蒙问题的多项式时间量子算法。

人们自然要问如下两个问题。(1)研究西蒙问题有何意义？(2)为什么西蒙问题在传统计算机上难以求解？问题(1)的最佳答案是，西蒙问题的推广形式在为著名的整数分解问题设计多项式时间量子算法时将发挥重要作用，我们将在 10.6 节看到这一点。关于问题(2)，当然我们仍不十分确定西蒙问题不存在传统的多项式时间算法（特别地，如果 **P=NP**，则显然存在这样的算法），但是，从直觉上看西蒙问题的难解性可以用如下的黑箱模型来简单地理解。假设你用黑箱（或神喻）来获取输入 $x\in\{0,1\}^n$ 上的函数值 $f(x)$。那么，如果仅向黑箱咨询亚指数次函数值，你能找到 a 吗？不难看到，如果 a 是从 $\{0,1\}^n$ 中随机选取的并且 f 也是在满足“$f(x)=f(y)$ 当且仅当 $x=y\oplus a$”的条件下随机选取的，则在仅向黑箱咨询少于 $2^{n/2}$ 次函数值的情况下任意传统的算法都无法以恰当的概率找到 a。事实上，如果算法向黑箱咨询函数值的次数再少一些，则算法甚至极可能无法发现两个具有相同函数值的输入，进而它也就无法获得 a 的任何信息。

10.5.1　定理 10.14 的证明

西蒙算法实际上非常简单。它使用具有 $2n+m$ 个量子位的寄存器，其中 m 个量子位是在计算 f 时用作演算的量子位。在下面的描述中，我们将忽略用于演算的 m 个量子位，因为这些量子位除了在中间计算 f 时之外将全部被置为 0。西蒙算法首先通过 n 个哈达玛操作将寄存器的前 n 个量子位置为均匀态，然后在寄存器上运用操作 $|xz\rangle\mapsto|x(z\oplus f(x))\rangle$ 得到如下状态（忽略归一化因子）：

$$\sum_{x\in\{0,1\}^n}|x\rangle|f(x)\rangle=\sum_{x\in\{0,1\}^n}(|x\rangle+|x\oplus a\rangle)|f(x)\rangle \tag{10.2}$$

然后，算法测量寄存器的 $2n$ 个量子位，寄存器的状态将塌陷到

$$|xf(x)\rangle+|(x\oplus a)f(x)\rangle \tag{10.3}$$

其中 x 是均匀随机地取自 $\{0,1\}^n$ 的一个串。你也许会认为上述步骤即为西蒙算法，因为(10.3)式中已经包含了 a，但是，从(10.3)式表示的状态中，我们还无法直接得到 a，这是由于，如果测量寄存器的前 n 个量子位，则我们得到 x 的概率是 1/2，并且得到 $x\oplus a$ 的概率也是 1/2。虽然将 x 和 $x\oplus a$ 两个值放在一起可以得出 a，但仅由其中一个值却无法得出 a。（停下来做上述思考是值得的，因为“由(10.3)式计算 a”是量子计算中的标准子程序，而且它同时还解释了为什么量子算法在一般情况下不等价于指数个传统计算的并行执行。） 220

下面考虑如果在寄存器的前 n 个量子位上再执行 n 个哈达玛操作会发生什么情况。由

于这些操作将 x 映射到向量 $\sum_y (-1)^{x\odot y}|y\rangle$，因此前 n 个量子位将进入新状态

$$\sum_y ((-1)^{x\odot y}+(-1)^{(x\oplus a)\odot y})|y\rangle = \sum_y ((-1)^{x\odot y}+(-1)^{x\odot y}(-1)^{a\odot y})|y\rangle \quad (10.4)$$

对于任意 $y\in\{0,1\}^n$，状态(10.4)的第 y 个系数不是 0 当且仅当 $a\odot y=0$；并且，对状态(10.4)进行测量时将得到服从均匀分布的满足 $a\odot y=0$ 的每个 $y\in\{0,1\}^n$。

重复整个过程 k 次，得到满足 $a\odot y=0$ 的服从均匀分布的 k 个串 y_1，…，y_k；换句话说，我们得到(有限域 GF(2)上)变量 a_1，…，a_n 的 k 个方程。容易证明，如果 $k\geqslant 2n$，则这 k 个方程中存在 $n-1$ 个线性独立方程的概率很高(参见习题 10.9)；因而，用高斯消去法可以从这些方程中解得 a。这就完成了定理 10.14 的证明。为了使论述更完备，图 10-5 给出了完整的西蒙算法。■

求解目标：给定一个多项式时间可计算函数 $f:\{0,1\}^n\mapsto\{0,1\}^n$，满足如下条件：存在 $a\in\{0,1\}^n$ 使得 $f(x)=f(y)$ 当且仅当 $y=x\oplus a$ 对任意 x，$y\in\{0,1\}^n$ 成立。求解目标是找出 a。

量子寄存器：使用具有 $2n+m$ 量子位的寄存器，其中 m 充分大，它使得寄存器足以计算变换 $|x0^m\rangle\mapsto|x(x\oplus f(x)0^m)\rangle$。下面将忽略寄存器末尾的 m 个量子位，因为这些量子位除了在中间计算 f 时之外将会存储 0^m。

操作	状态(忽略归一化因子)
	初始状态：$|0^{2n}\rangle$
在前 n 个量子位上运用哈达玛操作	$\sum_x |x0^n\rangle$
计算 $|x\rangle\mapsto|x(x\oplus f(x))\rangle$	$\sum_x |xf(x)\rangle = \sum_x (|x\rangle+|x\oplus a\rangle)|f(x)\rangle$
测量寄存器的 $2n$ 个量子位	$(|x\rangle+|x\oplus a\rangle)|f(x)\rangle$
在前 n 个量子位上运用哈达玛操作	$\left(\sum_y (-1)^{x\odot y}(1+(-1)^{a\odot y})|y\rangle\right)|f(x)\rangle$ $= 2\sum_{y:a\odot y=0}(-1)^{x\odot y}|y\rangle|f(x)\rangle$
测量寄存器的前 n 个量子位得到满足 $y\odot a=0$ 的 y 值。重复上述过程直到得到 a 上足够多的线性独立方程	

图 10-5　西蒙算法

10.6　肖尔算法：用量子计算机实现整数分解

整数分解问题要求为给定整数 N 找出所有的素因数(也就是能够整除 N 的素数)。整数分解问题的多项式时间算法指的是运行时间为 N 的二进制表示的长度 $\log N$ 的多项式的算法。尽管人们已经对整数分解问题冥思苦想了至少 2000 年，却仍未找到求解该问题的多项式时间算法。目前，分解 N 的最佳传统算法大约需要 $2^{(\log N)^{1/3}}$ 个计算步骤[LLMP90]。事实上，对整数分解问题难度的猜测是许多广泛使用的密码方案的基础，其中包括 9.2.1 节中遇到的公钥密码系统 RSA。因此，1994 年彼得·肖尔证明下面的结论时，确实让人们大感意外。肖尔算法目前是量子计算机上最著名的算法，同时也为"**BQP** 可能包含了 **BPP** 之外的问题"提供了最强有力的证据。

221

定理 10.15（肖尔算法：**BQP** 中的因数分解[Sho97]）　存在一个量子算法，它在给定的 N 上运行 $\mathrm{poly}(\log N)$ 时间，并输出 N 的素因数分解。

肖尔的简要想法

算法将用到如下的观察结果。首先，由于 N 至多有 $\log N$ 个因数，因此仅需说明如何在 poly(logN)时间内求得 N 的一个因数，这是由于我们可以在 N 除以该因数的商上重复调用算法进而找出所有的因数。其次，众所周知，为了找到 N 的一个因数，仅需对任意随机整数 A 找出模 N 的阶，也就是满足 $A^r \equiv 1 \pmod N$ 的最小正整数 r。阶的发现将在 10.6.4 节中详细讨论，但其主要思想是：在很大概率上，A 的阶 r 是偶数，并且 $A^{r/2}-1$ 和 N 存在非平凡的最大公因数，而最大公因数可以用 GCD("最大公因数"的首字母)算法找到。最后，映射 $A \mapsto A^r \pmod N$ 可以用快速乘幂法在 poly(logN)时间内计算，即使在传统图灵机上也是如此(因而用量子计算机也行)，参见习题 10.10。

利用上述观察结果可以构造一个简单的 poly(logN)时间的量子算法，它将初始状态全为 0 的量子寄存器转换到所有形如 $|x\rangle$ 的状态的均匀叠加态，其中 $x \leqslant N$ 且 $A^x \equiv y_0 \pmod N$ 对某个随机取定的 $y_0 \leqslant N-1$ 成立。由初等数论可知，所有这样的 x 构成算术级数[⊖] $x_0 + ri$，其中 $i=1, 2, \cdots, A^r \equiv 1 \pmod N$ 且 r 是 A 的阶。

现在，我们要解决的问题看起来有点像西蒙问题了，因为我们得到的量子态具有很强的周期性(即算术级数)而我们恰好需要求出它的周期。在工程中和数学上，用于计算周期的传统工具就是傅里叶变换(参见 10.6.1 节)。在紧接下来的一节中，我们将引入量子傅里叶变换(QFT)，它将用于计算量子态的周期。QFT 是一个量子算法，它将处于任意状态 $f \in \mathbf{C}^M$ 的寄存器变换到一个新的状态使得新状态的向量 $\hat{f}$ 恰好是 f 的傅里叶变换。QFT 仅执行 $O(\log^2 M)$ 个基本操作，因此它非常高效。注意，我们不能说 QFT "计算"了傅里叶变换，因为变换的结果存储在状态的振幅中，而前面已经指出，量子力学原理无法"读出"这些振幅本身。从量子态获取信息的唯一手段就是对它进行测量，它将以与振幅相关的概率测得一个基态。所得的基态难以全面反映整个傅里叶变换向量，但有时(比如肖尔算法中)它也提供了足够的在传统(非量子)计算机上无法获得的非平凡信息。

222

10.6.1　$\mathbf{Z}_M$ 上的傅里叶变换

我们下面定义 $\mathbf{Z}_M$ 上的傅里叶变换，其中 $\mathbf{Z}_M$ 是 $\{0, \cdots, M-1\}$ 中的整数在模 M 加法下构成的加法群。这里给出的定义是为讨论肖尔算法量身定制的。傅里叶变换在复杂性理论有很多应用，关于它的更多讨论，参见第 22 章。

定义 10.16　对任意向量 $f \in \mathbf{C}^M$，f 的傅里叶变换是另一个向量 $\hat{f}$，其中 $\hat{f}$ 的第 x 坐标是[⊖]

$$\hat{f}(x) = \frac{1}{\sqrt{M}} \sum_{y \in \mathbf{Z}_M} f(y) \omega^{xy}$$

其中 $\omega = e^{2\pi i/M}$ 且 i 是虚数单位。

傅里叶变换就是 f 在傅里叶基 $\{\chi_y\}_{y \in \mathbf{Z}_M}$ 下的简单表示，其中 χ_y 是第 y 个坐标取值为 $\frac{1}{\sqrt{M}\omega^{xy}}$ 的向量/函数。傅里叶基中的两个向量 χ_x 和 χ_y 的内积等于

$$\langle \chi_x, \chi_y \rangle = \frac{1}{M} \sum_{z \in \mathbf{Z}_M} \omega^{xz} \overline{\omega^{yz}} = \frac{1}{M} \sum_{z \in \mathbf{Z}_M} \omega^{(x-y)z}$$

⊖　即等差数列。——译者注

⊖　讨论傅里叶变换时，人们习惯于将向量 f 的第 x 个坐标记为 $f(x)$ 而不是 f_x。这种表示法也会给我们带来方便。

一方面，如果 $x=z$，则 $\omega^{(x-z)y}=1$，进而上面的和也等于 1。另一方面，如果 $x\neq z$，则用几何级数求和公式可得，上面的和等于 $\frac{1}{M}\frac{1-\omega^{(x-z)M}}{1-\omega^{x-z}}=\frac{1}{M}\frac{1-1}{1-\omega^{x-z}}=0$。换句话说，傅里叶基是标准正交基。这就表明，傅里叶变换 $f\to\tilde{f}$ 是一个酉操作。

什么原因造成了傅里叶基的上述特性呢？原因在于，如果将 $\mathbf{C}^M$ 中的向量视为从 $\mathbf{Z}_M$ 到 $\mathbf{C}$ 的函数，则不难看出，傅里叶基中的每个函数 χ 是从 $\mathbf{Z}_M$ 到 $\mathbf{C}$ 的同态映射，亦即 $\chi(y+z)=\chi(y)\chi(z)$ 对任意 $y,z\in\mathbf{Z}_M$ 成立。同时，χ 还是一个周期函数，亦即存在 $r\in\mathbf{Z}_M$ 使得 $\chi(y+r)=\chi(y)$ 对任意 $y\in\mathbf{Z}_M$ 成立。事实上，如果 $\chi(y)=\omega^{xy}$，则可以将 r 取为 ℓ/x，其中 ℓ 是 x 和 M 的最小公倍数。因此，直观上看，如果函数 $f:\mathbf{Z}_M\to\mathbf{C}$ 本身是周期的（或者大概上是周期的），则将 f 用傅里叶基表示出来之后，基向量的系数的周期很可能与函数 f 本身的周期是一致的。于是，我们可以从傅里叶变换的结果中找出函数 f 的周期。事实证明，情况确实如此，并且这也正是肖尔算法的关键之处。

快速傅里叶变换

将傅里叶变换表示为一个操作 FT_M，它将任意 $f\in\mathbf{C}^M$ 映射到傅里叶变换 $\tilde{f}$。操作 FT_M 可以表示为一个 $M\times M$ 的矩阵，其中 (x,y) 位置上的元素等于 ω^{xy}。该矩阵如果用平凡的算法来计算，则需要 M^2 个操作。著名的快速傅里叶变换算法（FFT）在计算傅里叶变换时
223 则仅用 $O(M\log M)$ 个操作。下面，我们梗概地介绍一下 FFT 这一传统算法的思想，10.6.2 节将用这些思想来描述量子傅里叶变换算法。

注意

$$\tilde{f}(x)=\frac{1}{\sqrt{M}}\sum_{y\in\mathbf{Z}_M}f(y)\omega^{xy}$$

$$=\frac{1}{\sqrt{M}}\sum_{y\in\mathbf{Z}_{M/2}\text{是偶数}}f(y)\omega^{x(y/2)}+\omega^{x}\frac{1}{\sqrt{M}}\sum_{y\in\mathbf{Z}_{M/2}\text{是奇数}}f(y)\omega^{x(y-1)/2}$$

现在，由于 ω^2 是 1 的 M 次方根中的第 $M/2$ 个，因此 $\omega^{M/2}=-1$。令 W 是对角线元素为 $\omega^0,\cdots,\omega^{M/2-1}$ 的 $M/2\times M/2$ 的对角矩阵，则得到

$$FT_M(f)_{\text{low}}=FT_{M/2}(f_{\text{even}})+W\cdot FT_{M/2}(f_{\text{odd}}) \tag{10.5}$$

$$FT_M(f)_{\text{high}}=FT_{M/2}(f_{\text{even}})-W\cdot FT_{M/2}(f_{\text{odd}}) \tag{10.6}$$

这里，对于 M 维向量 v，我们用 v_{even}（和 v_{odd}）表示 v 的偶数（和奇数）坐标分量构成的 $M/2$ 维向量，亦即，坐标的下标的二进制表示中末位等于 0（和 1）的坐标分量构成的 $M/2$ 维向量。类似地，我们用 v_{low}（和 v_{high}）表示 v 的前（和后）$M/2$ 个坐标分量构成的 $M/2$ 维向量，亦即坐标的下标的二进制表示中首位等于 0（和 1）的坐标分量构成的 $M/2$ 维向量。

等式(10.5)和(10.6)是基于分治思想设计 FFT 算法的关键，因为这两个等式告诉我们如何将规模为 M 的计算问题转换为两个规模均为 $M/2$ 的子问题。由此得到求解问题的时间复杂度的递归方程 $T(M)=2T(M/2)+O(M)$，进而解得 $T(M)=O(M\log M)$。

10.6.2　$\mathbf{Z}_M$ 上的量子傅里叶变换

量子傅里叶变换是一个量子算法，它将量子寄存器的状态 $f\in\mathbf{C}^M$ 转变成该向量的傅里叶变换 $\tilde{f}$。

引理 10.17（量子傅里叶变换[BV93]）　对任意 m，令 $M=2^m$，则存在使用 $O(m^2)$ 个

基本量子操作的一个量子算法将量子寄存器从状态 $f=\sum_{x\in\mathbb{Z}_M} f(x)|x\rangle$ 转换到状态 $\hat{f}=\sum_{x\in\mathbb{Z}_M}\hat{f}(x)|x\rangle$，其中 $\hat{f}(x)=\frac{1}{\sqrt{M}}\sum_{y\in\mathbb{Z}_M}\omega^{xy}f(y)$。

证明　算法的关键仍是等式(10.5)和(10.6)，它们使得在规模 M 下计算 FT_M 的问题可以被分裂成两个相同的在规模 $M/2$ 下计算 $FT_{M/2}$ 的问题。通过递归调用，$FT_{M/2}$ 可以使用相同的基本量子操作。(特别提醒，并不是每个传统的分治算法都能实现为快速的量子算法。这里，我们仅仅使用了 FT_M 特有的结构。)

矩阵 W 表示的线性变换作用在 $m-1$ 个量子位上，它可以定义为 $|x\rangle\mapsto\omega^{x'}=\omega^{\sum_{i=0}^{m-2}2^i x_i}$，其中 x_i 是 x 的第 i 个坐标。不难发现，对所有 $i\in\{0,\cdots,m-2\}$，在第 i 个量子位上运用基本量子操作"$|0\rangle\mapsto|0\rangle$ 且 $|1\rangle\mapsto\omega^{2^i}|1\rangle$"之后，根据等式(10.5)和(10.6)，量子寄存器最终的状态等于 $\hat{f}$。我们将上述结论的验证和时间复杂度分析留作习题 10.14。 ■ 224

量子傅里叶变换 FT_M 初始状态：$f=\sum_{x\in\mathbb{Z}_M} f(x)\|x\rangle$ 最终状态：$\hat{f}=\sum_{x\in\mathbb{Z}_M}\hat{f}(x)\|x\rangle$	
操作	**状态(忽略归一化因子)**
	$f=\sum_{x\in\mathbb{Z}_M} f(x)\|x\rangle$
在前 $m-1$ 个量子位上递归计算 $FT_{M/2}$	$(FT_{M/2}f_{even})\|0\rangle+(FT_{M/2}f_{odd})\|1\rangle$
如果最末一个量子位等于 1，则在前 $m-1$ 个量子位上计算 W	$(FT_{M/2}f_{even})\|0\rangle+(W\cdot FT_{M/2}f_{odd})\|1\rangle$
在最末一个量子位上运用哈达玛门 H	$(FT_{M/2}f_{even})(\|0\rangle+\|1\rangle)+(W\cdot FT_{M/2}f_{odd})(\|0\rangle-\|1\rangle)$ $=(FT_{M/2}f_{even}+W\cdot FT_{M/2}f_{odd})\|0\rangle$ $+(FT_{M/2}f_{even}-W\cdot FT_{M/2}f_{odd})\|1\rangle$
将最末一个量子位前置作为寄存器的首个量子位	$\|0\rangle(FT_{M/2}f_{even}+W\cdot FT_{M/2}f_{odd})+$ $\|1\rangle(FT_{M/2}f_{even}-W\cdot FT_{M/2}f_{odd})=\hat{f}$

10.6.3　肖尔的阶发现算法

现在，我们给出肖尔因数分解算法的核心步骤——找出整数 A 模整数 N 的阶的量子多项式时间算法。

引理 10.18　存在多项式时间的量子算法在输入 A，N(表示为二进制形式)上找到满足 $A^r=1(\mathrm{mod}\ N)$ 的最小 r。

证明　令 $m=\lceil 5\log N\rceil$ 和 $M=2^m$。量子寄存器将使用 $m+\mathrm{polylog}(N)$ 个量子位。注意，函数 $x\mapsto A^x(\mathrm{mod}\ N)$ 可以在 $\mathrm{polylog}(N)$ 时间内被计算(参见习题 10.10)。因此，我们假设可以计算映射 $|x\rangle|y\rangle\mapsto|x\rangle|y\oplus\lfloor A^x(\mathrm{mod}\ N)\rfloor\rangle$，其中 $\lfloor X\rfloor$ 是整数 $X\in\{0,\cdots,N-1\}$ 的长度为 $\log N$ 的二进制表示形式。[⊖] 下面，我们给出阶的发现算法。该算法要用到初等数论中

⊖ 为了计算这个映射，量子寄存器需要使用另外 $\mathrm{polylog}N$ 个量子位，但这些扩展的量子位可以被忽略，因为出来在中间计算中使用它们的时候之外它们将始终等于 0。

的连分数作为工具。连分数确保，我们可以通过传统算法把任意实数 a 近似为一个有理数 p/q，其中 q 有一个事先给定的上界(参见 10.6.5 节)。在分析过程中，仅需证明算法将至少概率 $\Omega(1/\log N)$输出 A 的阶 r。(可以通过重复运行算法，然后取各次运行过程输出的最小 r 值，最终将算法成功的概率放大。)

阶发现算法

求解目标：给定整数 N 和 $A<N$，其中 $gcd(A, N)=1$，找出最小的 r 使得 $A^r=1(\mathrm{mod}\ N)$。

量子寄存器：我们使用具有 $m+\mathrm{polylog}N$ 量子位的寄存器，其中 $m=\lceil 3\log N\rceil$ 且 $2^m=M$。下面，我们用前 m 个量子位来编码 $\mathbf{Z}_M$中的整数。

操作	状态(忽略归一化因子)
在前 m 个量子位上执行傅里叶变换	$\frac{1}{\sqrt{M}}\sum_{x\in\mathbf{Z}_M}\lvert x\rangle\lvert 0^n\rangle$
计算变换 $\lvert x\rangle\lvert y\rangle\mapsto\lvert x\rangle\lvert y\oplus(A^x(\mathrm{mod}\ N))\rangle$	$\frac{1}{\sqrt{M}}\sum_{x\in\mathbf{Z}_M}\lvert x\rangle\lvert(A^x(\mathrm{mod}\ N))\rangle$
测量寄存器的后 n 个量子位得到值 y_0	$\frac{1}{\sqrt{K}}\sum_{t=0}^{K-1}\lvert x_0+tr\rangle\lvert y_0\rangle$，其中 x_0 是满足 $A^{x_0}=y_0(\mathrm{mod}\ N)$ 的最小整数，$K=\lfloor(M-x_0-1)/r\rfloor$
在前 m 个量子位上执行傅里叶变换	$\frac{1}{\sqrt{M}\sqrt{K}}\left(\sum_{x\in\mathbf{Z}_M}\sum_{t=0}^{K-1}\omega^{(x_0+tr)x}\lvert x\rangle\right)\lvert y_0\rangle$
测量寄存器的前 m 个量子位得到整数 $x\in\mathbf{Z}_M$。求得 $\frac{x}{M}$ 在误差范围 $1/(10M)$内的一个近似有理数 a/b，其中 a 和 b 互素，并且 $b<N$(参见 10.6.5 节)。如果用于近似的有理数满足 $A^b=1(\mathrm{mod}\ N)$，则输出 b。	

分析 $r\mid M$ 的情形

我们从分析"$M=rc$ 对某个 c 成立"的情况开始来分析算法。虽然这样的情况很多(注意 M 是 2 的幂)，因而无法一一列举，但是对这类情况的分析却直观地说明了为什么傅里叶变换可用来计算周期。

论断：此时，x 的测量值等于 ac，其中 $a\in\{0, \cdots, r-1\}$是一个随机值。

由上述断言即可完成证明，这是由于该断言表明 $x/M=a/r$ 对小于 r 的随机整数 a 成
225 立。并且，对任意 r，$[r-1]$中至少有 $\Omega(r/\log r)$个数与 r 互素。事实上，素数定理(参见 A.3 节)断言，$[r-1]$中至少有 $\Omega(r/\log r)$个素数，又由于 r 至多有 $\log r$ 个素因数，这 $\log r$ 个素因数之外的其他素数都与 r 互素。因此，当肖尔算法计算得到 x/M 的近似有理数后，有理数的分母实际上就是 r。

往证上述断言。在对寄存器进行测量之前，我们先对任意 $x\in\mathbf{Z}_M$计算 $\lvert x\rangle$的系数的绝对值。忽略归一化因子后，该系数等于

$$\left|\sum_{t=0}^{c-1}\omega^{(x_0+tr)x}\right|=\left|\omega^{x_0x}\right|\left|\sum_{t=0}^{c-1}\omega^{trx}\right|=1\cdot\left|\sum_{t=0}^{c-1}\omega^{trx}\right| \tag{10.7}$$

如果 c 不能整除 x，则 ω^{rx} 是 1 的 c 次方根，于是由几何级数求和公式可得 $\sum_{t=0}^{c-1}\omega^{trx}=0$。因此，测量得到这种 x 的概率等于 0。但是，如果 $x=cj$，则 $\omega^{trx}=\omega^{trcj}=\omega^{Mtj}$，进而在所有 $j\in\{0, 2, \cdots, r-1\}$上这种 x 的振幅都相等。

一般情形

在一般情形下，r 不一定整除 M，因而，我们无法证明测量值 x 满足 $M\mid xr$。但是，我们将证明，下列两个条件以至少为 $\Omega(1/\log r)$ 的概率成立：(1)xr“几乎总能”被 M 整除，亦即 $0\leqslant xr(\bmod M)<r/10$；并且(2)$\lfloor xr/M\rfloor$ 与 r 互素。条件(1)表明，$|xr-cM|<r/10$ 对 $c=\lfloor xr/M\rfloor$ 成立。不等式两端同时除以 rM 则得到 $\left|\frac{x}{M}-\frac{c}{r}\right|<\frac{1}{10M^2}$。于是，$\frac{c}{r}$ 这个分母至多为 N 的有理数以 $1/(10M)<(1/4N^2)$ 的误差近似表示了 $\frac{x}{M}$。不难看到，这种近似是唯一的(习题 10.11)。于是，此时算法将得到 c/r，并输出其分母 r(参见 10.6.5 节)。 226

剩下的工作就是证明如下两个引理。第一个引理表明，满足上述两个条件的 x 有 $\Omega(r/\log r)$ 个，第二个引理表明，每个这样的 x 被算法测量得到的概率为 $\Omega((1/\sqrt{r})^2)=\Omega(1/r)$。

引理 10.19　存在 $\Omega(r/\log r)$ 个 $x\in\mathbf{Z}_M$ 满足

1. $0\leqslant xr(\bmod M)<r/10$。
2. $\lfloor xr/M\rfloor$ 与 r 互素。

引理 10.20　如果 x 满足 $0\leqslant xr(\bmod M)<r/10$，则阶发现算法在执行最后的测量步骤之前，$|x\rangle$ 的系数至少为 $\Omega\left(\frac{1}{\sqrt{r}}\right)$。

引理 10.19 的证明　我们仅证明 r 与 M 互素的情况，一般情况的证明留作习题 10.15。在这种情况下，映射 $x\mapsto rx(\bmod M)$ 是 $\mathbf{Z}_M^*$ 上的排列。于是，$\{1,\cdots,r/10\}$ 中至少存在 $\Omega(r/\log r)$ 个与 r 互素的数(在上述范围的所有素数中 r 的素因数仅占 $\log r$ 个)。因此，$\{1,\cdots,r/10\}$ 中存在 $\Omega(r/\log r)$ 个与 r 互素的 x 满足 $rx(\bmod M)=xr-\lfloor xr/M\rfloor M$。但这又意味着 $\lfloor xr/M\rfloor$ 和 r 没有非平凡的公因数。否则，这样的公因数也必然是 $\lfloor xr/M\rfloor$ 和 $rx(\bmod M)$ 的公因数。 ■

引理 10.20 的证明　设 x 满足 $0\leqslant xr(\bmod M)<r/10$。阶发现算法在执行最后的测量步骤之前，$|x\rangle$ 的系数的绝对值是

$$\frac{1}{\sqrt{K}\sqrt{M}}\left|\sum_{\ell=0}^{K-1}\omega^{\ell x}\right| \tag{10.8}$$

其中，$K=\lfloor(M-x_0-1)/r\rfloor$。注意，由于 $x_0<N\ll M$，故 $\frac{M}{2r}<K<\frac{M}{r}$。

令 $\beta=\omega^x$(注意，由于 $M\nmid rx$，故 $\beta\neq1$)，再根据几何级数的求和公式可知，$|x\rangle$ 的系数的绝对值至少为

$$\frac{\sqrt{r}}{2M}\left|\frac{1-\beta^{\lceil M/r\rceil}}{1-\beta}\right|=\frac{\sqrt{r}}{2M}\frac{\sin(\theta\lceil M/r\rceil/2)}{\sin(\theta/2)} \tag{10.9}$$

其中角度 $\theta=\frac{rx(\bmod M)}{M}$ 满足 $\beta=e^{i\theta}$(图 10-6 图示了(10.9)式的证明)。在我们的假设条件下，有 $\lceil M/r\rceil\theta<1/10$，进而(根据“$\sin\alpha\sim\alpha$”在 α 很小时成立)得到，$|x\rangle$ 的系数至少为 $\frac{\sqrt{r}}{4M}\lceil M/r\rceil\geqslant\frac{1}{8\sqrt{r}}$。 ■

这就完成了引理 10.18 的证明。 ■

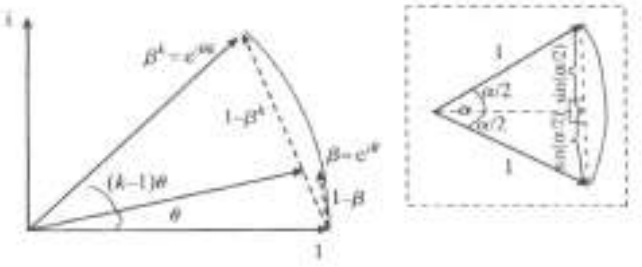


图 10-6　复数 $z=a+ib$ 可以视为长度为 $|z|=\sqrt{a^2+b^2}$ 的二维向量 (a, b)。复数 $\beta=e^{i\theta}$ 是与 x 轴夹角为 θ 的单位向量。对于任意这样的 β，只要 k 不太大（不妨设 $k<1/\theta$），则由初等几何可得 $\frac{|1-\beta^k|}{1-\beta}=\frac{2\sin(\theta/2)}{2\sin(k\theta/2)}$。这里，用到了如下事实（其证明过程如虚线框所示）：α 弧度的角在单位圆圆周上张成的弦的长度等于 $2\sin(\alpha/2)$

227

10.6.4　因数分解归约为阶发现

因数分解问题可以通过传统算法归约为阶发现问题，特别地，这种归约也可以在量子计算中实现。具体的归约过程可以表述为下面的两个引理。

引理 10.21　如果合数 N 不是素数的幂，则"$\mathbf{Z}_N^*=\{X\in[N-1]:\ gcd(X, N)=1\}$ 中的随机元素 X 具有偶数阶 r 且 $X^{r/2}\not\equiv -1(\mathrm{mod}\ N)$"的概率至少为 1/4。

引理 10.22　对于任意 N 和 Y，如果 $Y^2\equiv 1(\mathrm{mod}\ N)$ 但 $Y(\mathrm{mod}\ N)\notin\{+1, -1\}$，则 $gcd(Y-1, N)\notin\{1, N\}$。

引理 10.21 和引理 10.22 放在一起将表明：给定一个不是素数幂的合数 N，如果我们从 $[N-1]$ 中随机选取 A，则"$gcd(A, N)$ 和 $gcd(A^{r/2}-1, N)$ 之一是 N 的一个非平凡因数 F"的概率将很高。进而，可以对 F 和 N/F 递归地进行因数分解，由此得到时间复杂度为 polylog(N) 的因数分解算法。（注意，如果 N 是素数的幂，则直接对每个 $\ell\in[\log N]$ 查验 N 的 ℓ 次根是否是 N 的因数，即可实现对 N 的因数分解。）因此，要完成定理 10.15 的证明，仅需证明引理 10.21 和引理 10.22 即可。两个引理的证明都将用到数论中的一些基本事实，A.3 节对这些基本事实进行了概述。

引理 10.22 的证明　在引理的假设条件下，N 整除 $Y^2-1=(Y+1)(Y-1)$ 但它不能整除 $Y+1$ 和 $Y-1$。而这意味着 $gcd(Y-1, N)>1$。这是由于，如果 $Y-1$ 和 N 互素，则由 N 整除 $(Y+1)(Y-1)$ 可知 N 必然整除 $Y+1$（习题 10.12）。由 $Y-1<N$ 可知，$gcd(Y-1, N)<N$ 显然成立，因此引理成立。 ■

引理 10.21 的证明　我们仅证明"$N=PQ$ 对素数 P，Q 成立"的情形，证明过程可以恰当地推广使得它对任意 N 成立。现在，由中国剩余定理可知，任意 $X\in\mathbf{Z}_N^*$ 都同构于序对 $\langle X(\mathrm{mod}\ P), X(\mathrm{mod}\ Q)\rangle$。特别地，"从 $\mathbf{Z}_N^*$ 中随机选取 X"等价于"从 $\mathbf{Z}_P^*$ 和 $\mathbf{Z}_Q^*$ 中分别随机地选取 Y 和 Z，并令 X 是上述同构映射下唯一与 $\langle Y, Z\rangle$ 对应的数"。于是，对于任意 k 而言，$X^k(\mathrm{mod}\ N)$ 同构于 $\langle Y^k(\mathrm{mod}\ P), Z^k(\mathrm{mod}\ Q)\rangle$。因此 X 的阶是 Y 模 P 的阶和 Z 模 Q 的阶的最小公倍数。为了完成引理的证明，我们往证：Y 的阶是偶数的概率至少为 1/2。具体而言，我们将证明 Y 的阶形如 $2^k c$（其中 $k\geqslant 1$ 而 c 是奇数）的概率至少为 1/2。然后，再证明 Z 的阶形如 $2^\ell d$（其中 $\ell\neq k$ 而 d 是奇数）的概率至少为 1/2。这就表明，X 的阶为 $r=2^{\max\{k,\ell\}}\mathrm{lcm}(c, d)$，其中 lcm 表示求最小公倍数。这又意味着，$X^{r/2}$ 在同构映射下至少有一个坐标分量等于 1。这是由于 $-1(\mathrm{mod}\ N)$ 同构于元组 $(-1, -1)$，亦即，$X^{r/2}\neq -1$

(mod P)或 $X^{r'}\neq -1 \pmod{Q}$。

因此，我们仅需证明：

- Y 的阶是偶数的概率至少为 1/2。

228

事实上，$\mathbf{Z}_P^*$ 中阶为奇数的所有数构成 $\mathbf{Z}_P^*$ 的一个子群。如果 Y，Y' 的阶分别是奇数 r，r'，则 $(YY')^{rr'}=1 \pmod{P}$，这意味着 YY' 的阶整除 rr' 这个奇数。另一方面，-1 的阶是偶数，这意味着上述子群是 $\mathbf{Z}_P^*$ 的真子群，进而其中至多包含 $\mathbf{Z}_P^*$ 的一半元素。

- 存在整数 ℓ_0 使得："随机取自 $\mathbf{Z}_Q^*$ 的数 X 的阶形如 $2^\ell c$（其中 $\ell\leqslant\ell_0$）"的概率恰为 1/2。（这表明，对于任意固定的 k，X 的阶形如 $2^k d$ 的概率至多为 1/2。）

对任意 ℓ，定义 G_ℓ 是 $\mathbf{Z}_Q^*$ 中模 Q 的阶形如 $2^j c$（其中 $j\leqslant\ell$ 且 c 是奇数）的所有数构成的子集。不难验证，对任意 ℓ 而言，G_ℓ 均是 $G_{\ell+1}$ 的子群。并且，由于对素数 P 取模，映射 $x\mapsto x^2 \pmod{P}$ 是一个二对一的映射并且将 $G_{\ell+1}$ 映射到 G_ℓ 中（习题 10.13）。由此可知，$|G_\ell|\geqslant |G_{\ell+1}|/2$。据此，如果取 ℓ_0 为保证"G_{ℓ_0} 是 $\mathbf{Z}_P^*$ 的真子群"的最大整数，则 $|G_{\ell_0}|=|\mathbf{Z}_P^*|/2$。■

10.6.5　实数的有理数近似

在许多场合（包括肖尔算法）中，我们经常以程序的形式给定一个实数 x，该程序能够在 poly(t)的时间复杂度计算 x 的前 t 个二进制位。我们感兴趣的是，为 x 找到一个形如 a/b 的非常接近于真实值的近似，其中 b 有一个事先给定的上界。数论中的连分数是找出上述近似的一个有用的工具。

连分数是具有如下形式的数

$$a_0+\cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cdots}}}$$

其中 a_0 是非负整数而 a_1，a_2，…是正整数。

给定实数 $\alpha>0$，我们可以如下地将它表示成一个无限分数。先将 α 分裂成整数部分 $\lfloor\alpha\rfloor$ 和小数部分 $\alpha-\lfloor\alpha\rfloor$，然后递归地找出 $1/(\alpha-\lfloor\alpha\rfloor)$ 的无限分数形式 R。最后，将结果写成

$$\alpha=\lfloor\alpha\rfloor+\frac{1}{R}$$

如果上述递归过程仅进行 n 步，则我们得到一个有理数，将它记为 $[a_0, a_1, \cdots, a_n]$。这个有理数可以表示成 $\frac{p_n}{q_n}$ 的形式，其中 p_n 与 q_n 互素。利用归纳法，可以证明下列事实：

- $p_0=a_0$，$q_0=1$；并且对于任意 $n>1$，$p_n=a_n p_{n-1}+p_{n-2}$，$q_n=a_n q_{n-1}+q_{n-2}$
- $\dfrac{p_n}{q_n}-\dfrac{p_{n-1}}{q_{n-1}}=\dfrac{(-1)^{n-1}}{q_n q_{n-1}}$

并且，人们还证明了

$$\left|\frac{p_n}{q_n}-\alpha\right|<\frac{1}{q_n q_{n+1}} \tag{10.10}$$

这表明，在分母至多为 q_n 的所有有理数中，$\frac{p_n}{q_n}$ 是最接近于 α 的分数。(10.10)式还表明，如果 α 极度接近于某个有理数，不妨设 $\left|\alpha-\frac{a}{b}\right|<\frac{1}{4b^2}$ 对互素的 a，b 成立，则连分数算法

至多迭代 polylog$b$ 次就可以找出 a 和 b。事实上，设(连分数算法迭代过程中)q_n 是首个满足 $q_{n+1}\geqslant b$ 的分母。如果 $q_{n+1}>2b^2$，则(10.10)式将变为 $\left|\frac{p_n}{q_n}-a\right|<\frac{1}{2b^2}$，这就意味着 $\frac{p_n}{q_n}=\frac{a}{b}$，这是由于在分母至多为 b 的所有有理数中至多只有一个可以如此接近于 a。另一方面，如果 $q_{n+1}\leqslant 2b^2$，则 $\frac{p_{n+1}}{q_{n+1}}$ 比 $\frac{a}{b}$ 更接近于 a 且 $\left|\frac{p_{n+1}}{q_{n+1}}-a\right|<\frac{1}{4b^2}$，这又意味着 $\frac{p_{n+1}}{q_{n+1}}=\frac{a}{b}$。不难验证，$q_n\geqslant 2^{n/2}$。这意味着 p_n 和 q_n 可以在 polylog(q_n)时间内被计算出来。

229

10.7　BQP 和经典复杂性类

BQP 与我们前面遇到的 **P**，**BPP** 和 **NP** 等传统复杂性类之间有什么联系吗？这一问题的讨论还远未深入，有待进一步研究。不难证明，相对于传统计算机而言，量子计算机的能力也并不是无所不能的。

定理 10.23　**BQP**$\subseteq$**PSPACE**。

证明概要　为了模拟 m 量子位寄存器上执行的 T 步量子计算，需要构造一个过程 Coeff使得它能够对任意 $i\in[T]$和 $x\in\{0,1\}^m$(在一定精度范围内)计算在执行第 i 遍时量子寄存器的第 x 个系数。并且，在输入 x，i 上，Coeff 完成计算的过程中至多 8 次递归调用 Coeff 在 x'，$i-1$ 上执行的计算(8 次递归调用时要用到 8 个串 x'。每个 x'恰有 3 个位是由 x 计算系数时用过的位)。由于递归调用时可以重用以前用过的空间，如果将计算 Coeff(x,i)时需要的空间大小记为 $S(i)$，则 $S(i)\leqslant S(i-1)+O(\ell)$(其中 ℓ 是存储每个系数需要的比特数)。

为了计算得到"量子计算执行最后一个操作步骤之后，寄存器中(不妨说)第 1 个量子位上测得值 1"的概率，只需在任意 $x\in\{0,1\}^m$ 上对 Coeff(x,T)求和。同样，由于计算每个 Coeff(x,T)时使用的空间是可以复用的，因此上述概率的计算也可以在多项式空间内完成。■

本书后面习题 17.7 中，要求改进定理 10.23，证明 **BQP**$\subseteq$**P**$^{\#\mathbf{P}}$(其中 #**P** 将在第 17 章中定义，它是 **NP** 的计数形式)。还可以证明 **BQP**$\subseteq$**PP**[ADH97](参见定义 17.6)。本质上，这些结果就是所有已被证明的 **BQP** 的最佳上界。

BQP=**BPP** 成立吗？人们相信上述结论不成立，其主要原因就是整数分解问题存在多项式时间的量子算法，但是人们还相信概率型计算中任何算法都无法在多项式时间内求解整数分解问题。虽然上述证据没有表明 **NP**$\not\subseteq$**BPP** 的证据那么强(毕竟 **NP** 中数以千计的计算问题可能不存在高效的算法)，但整数因数分解问题是最古老的和研究最充分的计算问题之一，人们至今未能找到高效求解它的算法，这也促使人们猜想：整数的因数分解问题不存在高效的算法。同时，还需要注意到，不同于那些最终找到高效算法的著名计算问题(如线性规划问题[Kha79]，素性测试问题[ASK04])，我们就连对两个大素数乘积的分解，甚至也未找到任何疑似行之有效的或实验可行的启发式算法(更不用说证明)。

BQP 与 **NP** 之间有什么关系？通过格罗弗算法，量子计算机似乎仅能将 **NP** 完全问题的求解速度提高为传统算法复杂度的平方根。有的神喻结果还表明，某些 **NP** 问题在量子计算机上也需要指数时间[BBBV97]。因此，大多数研究者相信 **NP**$\not\subseteq$**BQP**。另一方面，**BQP** 中也存在一个问题(递归傅里叶抽样问题或简称 RFS 问题[BV93])仍未被证明是否属于多项式分层，更不用说判定它是否属于 **NP**。因此，**BQP** 和 **NP** 似乎是不可比较的两个类。

230

10.7.1　量子计算中类似于 NP 和 AM 的复杂性类

我们可以在量子计算中定义类似于 **NP** 的复杂性类吗？类 **NP** 是用证明这一概念来定义的，并要求证明必须能够被多项式时间的确定型(传统)图灵机验证。但是，量子计算把概率型传统计算作为子程序进行了调用。于是，在我们需要关注的计算模型中，证明必须能够被多项式时间随机算法查验，也就是 **MA** 模型(参见定义 8.10)。因而，我们将量子计算中类似于 **NP** 的复杂性类记为 **QMA**。更一般地，我们可以定义量子交互式证明，它可以推广 **AM**[k]的定义。事实证明，这将极大地提高计算能力。事实上，3 个回合的量子交互式证明系统刻画的复杂性类就是 **PSPACE**，这一结论是瓦特罗斯(Watrous)证明的[Wat03]。如果同样的结论对传统的交互式证明也成立，则 **PH** 就会坍塌。

"量子库克-勒维定理"被基塔耶夫(Kitaev)证明(这一结果并未发表，参见乌迈什·瓦兹拉尼(Umesh Vazirani)的讲义，在本书的网站上可以找到该讲义的链接)。上述结果证明了 3SAT 在量子计算中的类似问题(称为 Q5SAT)是 **QMA** 完全问题。在 Q5SAT 问题中，我们给定 n 位量子寄存器上的 m 个基本量子操作 H_1，H_2，…，H_m，每个操作仅操作寄存器的 5 个量子位(因此，每个操作都可以表示成一个 $2^n\times 2^n$ 的矩阵，该矩阵隐式地表示了 $2^n\times 2^n$ 的一个矩阵)。令 H 是 $2^n\times 2^n$ 的矩阵 $\sum_j H_j$。我们已知要么 H 的所有特征值都$\geqslant b$要么 H 有一个特征值$\leqslant a$，其中 $0\leqslant a\leqslant b\leqslant 1$ 并且 $b-a$ 至少为 $1/n^c$(c 是一个常数)。我们需要判定上述两种情况中到底哪种情况成立。

读者可以将证明 Q5SAT 问题的完全性作为一个习题。作为预备工作，你需要先将 3SAT 归约到 Q5SAT。

本章学习内容

- 量子力学有悖于直觉——n 个粒子构成的系统可以同时处于 2^n 种状态的叠加态。
- 量子计算机由具有 n 个量子位的寄存器构成，寄存器上可以执行量子操作，而量子操作可以表示为 $2^n\times 2^n$ 的酉阵。每个基本量子操作能操作 3 个量子位，并且可以表示为 8×8 的矩阵。
- 格罗弗算法能在 $2^{n/2}$ 时间内求解 SAT 的含 n 个变量的实例。格罗弗算法还可以扩展用于找出含 n 个输入位的(传统)线路的一个满足性赋值。
- 西蒙算法和肖尔算法都用"量子计算机能够高效地计算状态向量的傅里叶变换"这一事实来实现对传统计算机(即图灵机)上最佳算法的指数级加速。傅里叶变换可以用来找出状态向量中的周期。
- 肖尔算法在量子计算机上用 poly(n)时间完成 n 位整数的因数分解。这构成了对强邱奇-图灵命题的严重挑战，因为许多人都相信因数分解问题在传统计算机上不存在多项式时间算法。
- 人们可以研究判定树、通信协议和交互式证明等很多传统计算模型在量子计算中的类似模型。
- 人们仍无法确定量子计算机能够真正被物理地实现。当前，这是非常活跃的研究领域。理论研究表明，建造量子计算机不存在本质性的困难。

本章注记和历史

量子计算机是可逆的(引理 10.7)。在开始着手研究量子计算之前，人们曾对可逆计算这一领域开展了研究[Ben87]。可逆计算旨在从热力学角度找出传统计算机计算速度的局限性。Toffoli 门正是在这样的背景下被提出的。

1982 年，费曼[Fey82]指出，在传统的图灵机上似乎无法高效地模拟量子力学，并建议建立量子计算机来执行这种模拟。(事实上，如果量子计算机真的被建造出来，则它的主要应用也就是模拟传统计算机。)同时，费曼还指出，量子图灵机的计算能力可能会强于传统图灵机。1985 年，德吾奇(Deutsch)首先形式地定义了量子图灵机，虽然现在看来他的定义不尽令人满意。后来，更完善的定义出现在德吾奇，约萨(Josza)的论文[DJ92]和伯恩斯坦(Bernstein)，瓦兹拉尼(Vazirani)的论文[BV93]中。后一篇论文首次证明了通用量子图灵机的存在性，并证明了用通用量子图灵机模拟其他量子图灵机时仅导致计算效率下降一个多项式因子。姚期智[Yao93]将这些结果推广到了量子线路上，本章给出的量子计算的定义源自姚期智。(与量子线路相比，伯恩斯坦-瓦兹拉尼定义的量子图灵机容忍噪音的能力较差，因此被认为是不太可能被实现的。)德吾奇[Deu89]证明了某个 3 量子门在量子线路中是通用量子门，而索洛维(Solovay)(1995 年未发表的手稿)和基塔耶夫[Kitaev]各自独立地证明了，通用量子门可以将任意酉阵近似到量子门个数的指数精度，由此得到定理 10.12(尽管本章表述该定理时使用了书[NC00]中提到的特殊通用基)。

伯恩斯坦和瓦兹拉尼还引入了计算傅里叶变换的量子算法，并给出证据来说明该量子算法实现了对传统算法的超多项式时间加速。西蒙和肖尔的各自发表的论文沿着这种研究路线给出了更多的证据。特别地，肖尔的论文触及了科学界设想的基石，同时也是统治现实世界的基石(人们现在开始怀疑密码系统的安全性了)。

量子计算与密码学之间有着深刻的联系。一方面，一旦量子计算机被实现，则肖尔算法以及由此产生的各种推广可以用来彻底地破解各种密码系统，包括公钥密码系统 RSA 和其他基于因数分解或离散对数的密码系统。另一方面，人们已经证明，运用量子力学和 EPR/贝尔"悖论"背后的思想可以构造出具有无条件安全性的公钥密码系统，这种概念也就是著名的量子密钥分发[BB84]，更一般的概念是量子密码学。也就是说，这些密码系统在抵御计算能力无限的敌方时仍然是安全的。事实上，构建这样的密码系统不必等到量子计算机羽翼丰满之后，人们已经实现了这种密码系统的原型系统。当然，要使这些系统可用于真实世界，还有大量工程技术问题需要解决。但是，我们也要注意到，即使量子计算机被建造出来，某些传统的计算密码用多项式时间量子算法仍然很可能无法破解。例如，据我们所知，用量子计算机求单向函数(定义 9.4)的逆函数的复杂度至多是传统算法求逆函数复杂度的平方根(使用格罗弗算法)。因此，大多数研究者相信，私钥密码(甚至包括数字签名)也能抵御量子计算机的攻击，就如同它们能够抵御"传统"图灵机的攻击一样。而且，基于一些未找到高效量子算法的计算问题，人们甚至也设计出了(为数不多的)几个公钥密码系统。或许，在设计最不可能被量子计算机破解的密码系统时，人们应该考虑整数格上的某些计算问题(关于整数格，请参见第 9 章的注记)。

232

格罗弗算法和西蒙算法实际上可以运行在更一般的计算模型上，这种模型也称为量子黑箱模型，其上运行的算法将执行如下计算的神喻当做黑箱进行访问：神喻能够对于某个函数 f 计算酉变换 $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$ 并找出 f 的性质。这种算法的计算能力具有有意义的上界和下界。特别地，人们已经知道，格罗弗算法在这种模型下是最优的

[BBBV97]。在这种模型下，人们还找到了其他的几个"类格罗弗算法"的算法，参见综述[Amb04]。格罗弗算法可以视为在 $N=2^n$ 个变量上计算 OR 操作的结果。因此，人们自然要问，格罗弗算法能否推广到更一般的公式上？特别有意义的是，由于 AND-OR 树(亦即，求 OR 之后再求 AND 再求 OR……)经常出现在博弈策略等很多应用中，那么格罗弗算法能否推广到 AND-OR 树上呢？人们曾一度未能回答上述问题。特别地，人们当时不知道：在平衡的完全二叉 AND-OR 树上，量子算法是否能够快于最佳随机算法，其中最佳随机算法需要使用 $O(N^{\log_4(\frac{1+\sqrt{33}}{4})})=O(N^{0.753\cdots})$ 个变量[Sni81，SW86]。最近，该问题的研究获得了突破，法尔希(Farhi)，戈德斯斗恩(GoldStone)和古特曼(Gutmann)[FGG07]给出了该问题的一个 $O(N^{1/2+o(1)})$ 时间量子算法，该算法又被安柏剌斯(Ambainis)等人[ACR+07]改进到对所有 AND-OR 树成立。

量子计算方面的研究不仅加深了人们对传统的计算复杂性的理解，而且也加深了人们对"非计算"物理学的理解。在第一个方面上，一个很好的例子是阿哈诺夫(Aharono)和雷格夫(Regev)的论文[AR04]，它用量子观点证明了传统计算复杂性中的一个结果(格最短向量问题的 $\sqrt{n}$ 近似属于 **coNP**)。第二个方面的例子包含了量子纠错方面的研究(参见下面的讨论)和绝热计算方面的结果[AvDK+04，AGIK07，vDMV01]，这些研究结果厘清了绝热计算模型的概念并驳斥了物理学家最初对该模型的直观认识。

本章并未研究量子纠错问题。量子纠错处理如下的重要问题：在每个计算步骤都可能受到噪音干扰的条件下，我们如何才能正确运行量子算法呢？毫无疑问，上述问题至关重要。这是由于，当肖尔算法被实现并在我们感兴趣的 N 上运行时，必须确保成千上万个粒子长时间处于量子叠加态。迄今为止，人们还不清楚这一目标能否在实践中被实现。物理学家最初的直觉是，噪音和脱散使得量子计算是不切实际的，一个广为人知的障碍是不可复制定理[WZ82]，它似乎排除了在量子计算中运用传统纠错思想的可能性。但是，肖尔后来发表的关于量子纠错的论文[Sho95]得到了违背上述直觉的结论，由此引发了量子纠错方面的更多研究工作。阿哈诺夫和贝诺尔(Benor)证明了，在合理的噪音模型下，只要每个计算步骤中噪音概率小于某个常数，则可以执行任意长度的量子计算并高概率地获得正确的计算结果，参见普雷士格尔(Preskill)的文章[Pre97，Pre98]。不幸的是，物理系统上目前能够估计到的噪音率仍然高于该常数。

233

事实上，人们还不清楚什么才是正确的噪音模型，这一问题与如下的基本问题密切相关：用量子来描述世界的现实基础是什么？尽管量子力学理论在预测实验结果方面获得了巨大的成功(或许实验结果正是评判一个物理理论是否正确的唯一标准)，然而一些物理学家对"当前的量子力学理论将现实世界描述为一个规模巨大的状态数组，但是当人们观测这些状态时它们却会发生变化"感到相当不满意，这是可以理解的。因此，物理学家和哲学家试图为量子力学找到一种称心如意的能够解释实验结果的描述方法，流传甚广的科普书[Bru04]对这些尝试进行了综述(甚至还包括了作者的一些偏见)。

在更加技术的层面上，任何人都不怀疑量子效应存在于微观世界中。科学家们的问题是，为什么量子效应没有显现在宏观层面上来呢？或者说，为什么量子效应在宏观层面的表现人们无法注意到呢？物理学家彭罗斯(Penrose)[Pen90]甚至给出了一个具有高度争议性的建议：将"宏观量子效应无法引起人们的注意"关联到"概率波的塌陷"上。《科学美国人》上由 Yam 撰写的一篇文章[Yam97]描述了当年盛行的其他解释，包括脱敷(它用量子理论来解释宏观世界中量子效应的缺失现象)和隐变量理论(它将世界还原为确定的秩序)。还没有哪一种单独的解释能让所有物理学家感到满意。

最后，我们注意到，由于量子位是一个极其简单的量子系统，因此，在介绍量子力学时，越来越多的人倾向于使用量子位和量子计算，而不再使用标准的氢原子模型或箱面电子(Electron-in-a-box)。这是一个极好的例子，展示了基于计算的世界观(相对于整数分解这样的计算过程)是如何渗透到科学领域中的。

要了解量子计算和量子信息学在上述专题上的更多细节，请参阅基塔耶夫、Shen 和维亚尔伊(Vyalyi)的书[KVS02]，以及尼尔森(Nielsen)和 Chuang 的书[NC00]。一些优秀的讲义和综述也可以在乌迈什·瓦兹拉尼(Umesh Vazirani)和斯科特·阿伦森(Scott Aaronson)的主页中找到。阿伦森在《科学美国人》上发表的文章[Aar08]是量子计算领域优秀的科普读物。

习题

10.1　给出一个量子策略，使得爱丽丝和波比能以 0.85 的概率赢得定理 10.3 和定理 10.4 中的奇偶性游戏。

10.2　证明论断 10.5。

10.3　对于哈达玛操作、NOT 操作、受控的 NOT 操作、旋转 $\pi/4$ 操作、Toffoli 操作中的每个操作，写出一个 8×8 的矩阵来实现在 3 量子位寄存器的第一个量子位上执行该操作。

10.4　定义线性函数 $F:\mathbf{R}^{2^m}\to\mathbf{R}^{2^m}$ 为一个概率型基本操作，如果该函数满足下列条件：

- F 是概率型的：亦即，$\sum_x (F(x))_x = 1$ 对满足 $\sum_i v_i = 1$ 的任意向量 $v\in\mathbf{R}^{2^m}$ 成立。
- F 至多依赖于 3 个位。也就是说，存在一个线性函数 $G:\mathbf{R}^{2^3}\to\mathbf{R}^{2^3}$ 和 3 个坐标标号 $i<j<k\in[m]$ 使得，对于任意形如 $|x_1x_2\cdots x_m\rangle$ 的向量有

234

$$F|x_1x_2\cdots x_m\rangle = \sum_{a,b,c\in\{0,1\}} (G|x_ix_jx_k\rangle)_{abc}\,|x_1\cdots x_{i-1}ax_{i+1}\cdots x_{j-1}bx_{j+1}\cdots x_{k-1}cx_{k+1}\cdots x_m\rangle$$

设 $f:\{0,1\}^*\to\{0,1\}$ 和 $T:\mathbf{N}\to\mathbf{N}$ 是两个函数。我们称 f 是概率 $T(n)$ 时间可计算的，如果对于任意 $n\in\mathbf{N}$ 和任意 $x\in\{0,1\}^n$，$f(x)$ 都可以由如下过程计算：

(a) 将 m 位寄存器初始化为状态 $|x0^{m-n}\rangle$(亦即，x 后面填充一些 0)，其中 $m\leqslant T(n)$。

(b) 在 m 位寄存器上依次执行 $T(n)$ 个基本操作 $F_1,\cdots,F_{T(n)}$(这里，要求存在一个多项式时间的图灵机 TM 使得它在输入 1^n，$1^{T(n)}$ 上恰好输出基本操作 $F_1,\cdots,F_{T(n)}$ 的某种表示)。

(c) 测量寄存器得到值 Y。也就是说，如果 v 是寄存器的最终状态，则 Y 是一个随机变量，它取任意 $y\in\{0,1\}^m$ 的概率等于 v_y。

我们要求 Y 的第一个位等于 $f(x)$ 的概率至少为 2/3。

证明：在上述定义下，函数 $f(x)$ 是概率 $p(n)$ 时间可计算的(其中 p 是一个多项式)，当且仅当 $f\in\mathbf{BPP}$。

10.5　证明：如果 $f\in\mathbf{BQP}$，则 f 存在一个量子多项式时间算法使得其中的所有矩阵都是实数矩阵，亦即，不含形如 $a+ib$ 且 $b\neq 0$ 的数。(可以认为本习题证明了，量子计算的能力强于传统概率计算的能力是由于机器的状态表示中允许使用负数，而不是由于使用了复数。)

10.6　假设一个 2 量子位寄存器处于任意状态 v。证明：下述三个实验得到的输出服从相

同的概率分布。

(a) 测量寄存器，并输出测得的值。

(b) 先测量第 1 个量子位，输出测得的值；然后再测量第 2 个量子位，输出测得的值。

(c) 先测量第 2 个量子位，输出测得的值；然后再测量第 1 个量子位，输出测得的值。

10.7　假设 f 可以由一个量子算法在 T 时间内计算，但该量子算法执行过程中每个步骤都采用了部分测量并根据部分测量的不同结果相应地执行计算步骤。证明：f 可以由 $O(T)$ 个基本操作计算。

10.8　证明：对于一个包含 T 步的量子计算，将它的每个量子门都替换为任意量子门(即 8×8 的矩阵)，只要替换前、后两个矩阵中每个实数的前 $10\log T$ 个位相同，则量子计算得到的最终状态的振幅在前 T 个位上相同。

10.9　证明：如果对于某个 $a\in\{0,1\}^n$，一致随机地从 $\{0,1\}^n$ 中选取位串 $y_1,\cdots,y_{n-1}$ 使得 $y_i\odot a=0$ 对任意 $i\in[n-1]$ 成立，则"存在非零位串 $a'\neq a$ 使得 $y_i\odot a'=0$ 对任意 $i\in[n-1]$ 成立"的概率至少为 $1/10$。(换句话说，$y_1,\cdots,y_{n-1}$ 线性独立的概率至少为 $1/10$。)

235

10.10　证明：给定 A 和 $x\in\{0,\cdots,M-1\}$，可以(用传统算法!)在 $\log M$ 时间内计算 $A^x \pmod M$。

10.11　证明：对于任意 $a<1$，至多存在一个有理数 a/b 使得 $b<N$ 且 $|a-a/b|<1/(2N^2)$。

10.12　证明：如果 A，B 是整数，N 和 A 互素并且 N 整除 AB，则 N 整除 B。

10.13　完成引理 10.21 的证明：

(a) 证明：对于任意素数 P，映射 $x\mapsto x^2 \pmod P$ 是 $\mathbf{Z}_P^*$ 上的二对一的映射。

(b) 证明：如果 X 模 P 的阶形如 $2^j c$，其中 $j\geqslant 1$ 而 c 是奇数，则 X^2 模 P 的阶形如 $2^{j-1}c'$，其中 c' 是奇数。

(c) 对于不是素数幂的任意合数 N，证明引理 10.21。

10.14　证明引理 10.17。

10.15　证明引理 10.19 在 r 和 M 的情形下仍成立。亦即，证明此时仍存在至少 $\Omega(r/\log r)$ 个 $x\in\mathbf{Z}_M$ 使得(1) $0\leqslant xr \pmod M<r/10$；并且(2) $\lfloor xr/M\rfloor$ 与 r 互素。

10.16　(连分数知识的应用)假设 j，$r\leqslant N$ 互素，但二者均是未知的。证明：如果已知 j/r 的前 $2\log N$ 个位，则可以在多项式时间内恢复 j，r。

236

第 11 章

Computational Complexity, A Modern Approach

PCP 定理和近似难度简介

大多数归约都不会产生成保持这种鸿沟……要在一般的归约中产生这种鸿沟(参见库克定理)……似乎也不太可能。从直观上看,其中的原因在于:计算在本质上是不稳定的和非健壮的数学对象,因为对计算过程稍作修改,即使这种修改用任何合理的测度来度量都是非显著的,计算结果也可能从接受状态变为非接受状态。

——帕帕迪米特里奥,杨纳卡卡斯[PY88]

本文的贡献有两点。首先,它证明了图中最大团的大小的近似计算和多证明者交互式证明之间的联系。其次,构造了 **NP** 语言的一个高效的多证明者交互式证明,其中验证者仅使用了非常少的随机位和通信位。

——费格(Feige),戈德瓦瑟,洛瓦兹,萨弗拉(Safra)和塞盖迪(Szegedy)[FGL+91]

我们给出了 **NP** 的新性质,亦即,它恰好包含如下的所有语言:语言的成员资格证明在概率多项式时间内仅用对数个随机位就可以被验证,并且验证过程仅需读取证明中的亚对数个位。

——阿罗拉,萨弗拉[AS92]

本章介绍 **PCP** 定理,它是复杂性理论中的一个出人意料的结果,并且在算法设计中有很多应用。自从 1972 年 **NP** 完全性被发现之后,人们就曾经一直在思考如下的问题——是否能够为 **NP** 难的优化问题高效地计算出近似解。人们在尝试为许多 **NP** 难的优化问题设计这种近似算法时都失败了(11.1 节介绍了近似算法)。继而,人们转而尝试证明:近似求解 **NP** 难的优化问题也很难。除了在为数不多的几个独立的问题上获得了成功之外,其他尝试也都陷入了困境。研究者们逐渐认识到,库克-勒维-卡普式的归约不足以证明近似算法能力上的局限性(参见本章开头的引言,它源自帕帕迪米特里奥和杨纳卡卡斯的一篇颇具影响的论文,该论文的发表时间比本章要讨论的结果早几年)。1992 年发现的 **PCP** 定理给出了 **NP** 的新定义,并且提供了新的归约概念。**PCP** 定理当时曾一度被认为是非常出人意料的结论(参见 11.2.2 节末尾的注记)。

11.2 节将指出,可以用两种观点来理解 **PCP** 定理。一种观点是,**PCP** 定理建立了局部可验证的证明系统。**PCP** 定理给出了对任意数学证明进行变换的方法,变换后的证明仅需查验其中少数几个(随机选择的)位就可以验证证明的真伪。"PCP"是"概率可验证明(Probabilistically Checkable Proofs)"的缩写。另一种观点将 **PCP** 定理视为近似难度方面的一个结果。亦即,**PCP** 定理证明了,在许多 **NP** 难的优化问题上,求问题的近似解和求问题的精确解一样难。因此,这些问题无法高效地近似求解,除非 **P**=**NP**。在 11.3 节,我们将证明上述两种理解的等价性。

11.4 节将展示 **PCP** 定理的用处。在独立集问题 INDSET 和最小顶点覆盖问题 MIN-VERTEX-COVER 上,我们将用 **PCP** 定理导出近似求解它们的强难度。

尽管人们熟知的 **PCP** 定理特指一个具体的结果(即定理 11.5),但实际上人们已经找到好几个"**PCP** 定理",这些定理之间的区别在于参数设置各不相同。本章将证明一个这样

的 **PCP** 定理(11.5 节的定理 11.19)，它的结论比完整的 **PCP** 定理稍弱，但它仍非常有用。证明定理 11.19 的另一个动机是，它将在证明 **PCP** 定理时发挥重要作用。**PCP** 定理的完整证明将出现在第 22 章中。

各种 **PCP** 定理革命性地改变了人们对 **NP** 难问题的可近似性的理解。第 22 章将概览其中的几个 **PCP** 定理。

11.1　动机：近似求解 NP 难的优化问题

我们在第 2 章曾指出，研究 **NP** 完全性理论的主要动机之一就是为了理解求解 TSP 或 INDSET 这类组合优化问题的计算复杂度。由于 **P**≠**NP** 意味着数以千计的 **NP** 难的优化问题不存在高效的求解算法，因此人们转而研究能否为这类问题寻找高效的近似算法。在许多实际应用中，获得问题的近似解几乎与获得该问题的精确解一样好，而获取近似解可能要容易得多。研究者们感兴趣的是为 **NP** 难的优化问题找出可能的最佳近似解。例如，人们自然要问：能否在任意精度内近似求解感兴趣的 **NP** 问题。如果能够，则 **P**≠**NP** 不会对实际应用造成重大的影响。但是，很多研究者怀疑近似计算存在固有的局限性，而证明这种局限性正是导致 **PCP** 定理被发现的原动力。

本节将通过实例来展示近似算法的概念。设 MAX-3SAT 是如下定义的计算问题：给定 3CNF 布尔公式 φ 作为输入，找出所有布尔变量的一个赋值使得 φ 中被满足的子句个数达到最大值。该问题显然是 **NP** 难的，因为相应的判定问题 **3SAT** 是一个 **NP** 完全问题。我们如下定义求解 MAX-3SAT 的近似算法。

定义 11.1（MAX-3SAT 的近似）　对任意 3CNF 公式 φ，φ 的值指的是任意赋值能够使 φ 中被满足的所有子句的最大个数占 φ 的子句总数的比例。我们将 φ 的值记为 $\mathrm{val}(\varphi)$。特别地，φ 是可满足的当且仅当 $\mathrm{val}(\varphi)=1$。

238

对任意 $\rho\leqslant 1$，称算法 A 是 MAX-3SAT 的一个 ρ-近似算法，如果对于任意具有 m 个子句的 3CNF 公式 φ，$A(\varphi)$ 输出的赋值至少满足 φ 的 $\rho\cdot\mathrm{val}(\varphi)m$ 个子句。

下面，我们给出近似算法的两个例子。要了解更多非平凡的近似算法，请参阅本章注记中给出的参考书。

例 11.2（MAX-3SAT 的 1/2-近似）　我们用一个多项式时间算法来计算 MAX-3SAT 的 1/2-近似解。算法以贪心策略逐一地对每个布尔变量赋值。在对第 i 个变量赋值时需要确保"包含第 i 个变量的所有子句中至少有一半的子句被满足"。当一个子句被满足后，该子句被删除，亦即对其余变量赋值时将不再考虑该子句。显然，算法最终输出的赋值至少会使所有子句中的一半得到满足；进而，被近似解满足的子句个数将至少是被优化解满足的子句个数的一半。

对任意 $\varepsilon>0$，利用半正定规划还可以为 MAX-3SAT 设计一个多项式时间的 $(7/8-\varepsilon)$-近似算法(参见本章注记)。如果仅限于讨论"每个子句均包含 3 个不同的布尔变量"这种情况，则设计近似算法达到上述近似比是很容易的。事实上，所有变量的随机赋值满足每个子句的概率为 7/8。由数学期望的线性性质(论断 A.3)可知，被随机赋值满足的子句个数占所有子句个数的比例的数学期望为 7/8。根据这一观察结果，很容易设计一个简单的概率型(甚至是确定型)的 7/8-近似算法(参见习题 11.3)。

在有些问题上，对于任意 $\varepsilon>0$，都能够设计出 $(1-\varepsilon)$-近似算法。习题 11.2 要求读者对背包问题这一 **NP** 完全问题设计这种近似算法。 ◀

例 11.3（MIN-VERTEX-COVER 的 1/2-近似）　第 2 章例 2.15 曾介绍了顶点覆盖问题 VERTEX-COVER 的判定形式，它的优化形式记为 MIN-VERTEX-COVER。该问题要求我们在给定的图上确定最小顶点覆盖的大小（注意，顶点覆盖是一些顶点构成的集合，它使得图的每条边均至少有一个端点位于该集合中）。对于 $\rho \leqslant 1$，MIN-VERTEX-COVER 的一个 ρ-近似算法在输入图 G 上输出一个顶点覆盖，其大小至多是该图上最小顶点覆盖的大小的 $1/\rho$ 倍[⊖]。下面，我们给出 MIN-VERTEX-COVER 的一个 1/2-近似算法。

算法开始时 $S=\varnothing$。从图中任意选取一条边 e_1，将它的两个端点添加到 S 中。从图中删除这两个端点以及与这两个端点关联的所有边。迭代上述过程，选择边 e_2，e_3，…，并将它们的端点添加到 S 中，直到图变成空图。

显然，图的任意一条边都有一个端点位于最终输出的集合 S 中。因此，S 是一个顶点覆盖。而且，构建 S 时使用的所有边 e_1，e_2，…两两之间没有公共端点。换句话说，这些边构成一个匹配。S 的大小等于该匹配中边数的两倍。并且，根据定义，最小顶点覆盖也至少包含匹配中每条边的一个端点。因此，S 的大小至多是最小顶点覆盖的大小的两倍。 ◀

239

11.2　用两种观点理解 PCP 定理

PCP 定理可以用两种观点来理解，深刻地理解这两种观点是理解 **PCP** 定理及其证明的关键。一种观点是，**PCP** 定理讨论的是一种新的极其健壮的证明系统。另一种观点是，**PCP** 定理讨论的是组合优化问题的近似求解。

11.2.1　PCP 定理与局部可验证明

在第一种观点下（定理的名字源自这种观点），**PCP** 定理提供了一种新型的证明系统。假设某人希望让你相信某个布尔公式是可满足的。他向你提供了常规的证明，也就是一个满足性赋值。你可以将这个赋值代回到布尔公式中，以检验它是否是一个满足性赋值。然而，这样做的过程中，你必须读取整个证明。**PCP** 定理给出了另一种有趣的查验证明方法：提供证明的人很容易将证明改写使得你在验证证明的真伪时仅需查验其中随机选取的常数个位置（比方说 3 个位置）。并且，这种概率查验方法还具有下列性质：(1)正确的证明绝不会让你怀疑其正确性，也就是说，无论你投掷硬币后做出的随机选择是什么，你都不会拒绝证明的正确性；(2)如果公式本身是不可满足的，则在任何证明上你都将以很高的概率拒绝它。

当然，由于布尔公式的满足性是 **NP** 完全的，故任意其他的 **NP** 语言都可以用确定型算法高效地归约为布尔公式的可满足性语言。因此，**PCP** 定理可以应用于任意的 **NP** 语言。我们给出一个（由 **PCP** 定理得出的）有悖于直觉的后果。令 $\mathcal{A}$ 是任意一个常见的数学公理系统，其上的每个证明都可以用确定型图灵机在证明长度的多项式时间内验证。回顾一下，下面的语言属于 **NP**：

$$L=\{\langle \varphi, 1^n\rangle : \varphi \text{ 在 } \mathcal{A} \text{ 中存在长度} \leqslant n \text{ 的证明}\}$$

PCP 定理断言，L 存在概率可验证明。这种概率可验证明可以视为数学结论在通常意义下的有效证明经过改写后得到的另一种“证明”。与“标准数学证明需要逐行地查验才能验证其有效性”不同的是，在这种新的概念下，证明的有效性仅需查验证明中的常数个二

⊖ 许多教科书将这种近似算法称为 $1/\rho$-近似算法。

进制位就可高概率地得到验证。[⊖]

下面，我们更形式化地给出定义。回顾一下定义 2.1，语言 L 属于 **NP**，如果存在多项式时间图灵机 V（"验证者(Verifier)"的缩写）来查验输入 x"满足 $x\in L$"的证明（亦即，成员资格的证明）。换句话说，

$$x\in L \Rightarrow \exists \pi \quad \text{s.t.}\ V^{\pi}(x)=1$$
$$x\notin L \Rightarrow \forall \pi \quad V^{\pi}(x)=0$$

其中 V^{π} 表示证者可以访问证明串 π。

复杂性类 **PCP** 是上述概念的推广，推广过程包括如下的改变。第一，验证者是概率型的。第二，验证者可以对证明串 π 进行随机访问。也就是说，验证者可以使用一条特殊的地址带来随机独立地查询证明串中的每个二进制位。如果验证者要获取证明串中的第 i 个二进制位，则它先将 i 写在地址带上，然后获取二进制位 $\pi[i]$。[⊖] 在复杂性类 **PCP** 的定义中，"对证明的查询"将被视为一种珍贵的资源，要求它被执行的次数尽可能少。此外，还要求地址的长度是证明长度的对数。在这个模型下，多项式时间的验证者原则上可以查验具有指数长度的成员资格证明。

验证者既可以是自适应的，也可以是非自适应的。非自适应的验证者仅仅基于输入和随机带来选择要查询的二进制位。也就是说，任何查询都不能依赖于在这之前进行的其他查询反馈的结果。相反，自适应的验证者可以根据它已经获得的 π 上的一些位来选择下一个查询。我们只讨论非自适应的验证者，因为绝大多数 **PCP** 定理的证明都采用了这种验证者。习题 11.2 研究了自适应查询的效能。

定义 11.4（**PCP** 验证者）　令 L 是一个语言而 q、r：$\mathbf{N}\to\mathbf{N}$ 是两个函数。我们称 L 有一个 $(r(n), q(n))$-**PCP** 验证者，如果存在一个多项式时间的概率算法 V 满足下列条件：

高效性：在输入 $x\in\{0,1\}^*$ 上，允许验证者对一个长度不超过 $q(n)2^{r(n)}$ 的串 $\pi\in\{0,1\}^*$ 进行随机访问（我们称 π 是 x 的证明）。V 至多使用 $r(n)$ 个随机二进制位，并且非自适应地查询 π 中至多 $q(n)$ 个位置上的二进制位（参见图 11-1）。然后，V 输出 1（表示"接受"）或 0（表示"拒绝"）。V 在 x 上根据随机访问 π 的结果所得到的输出是一个随机变量，我们将这个随机变量记为 $V^{\pi}(x)$。

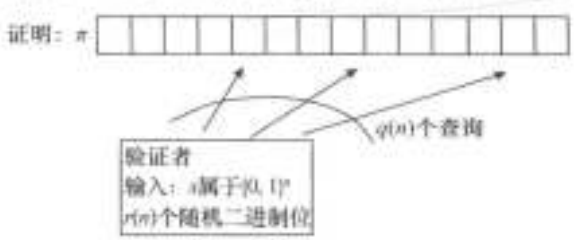


图 11-1　语言 L 的 **PCP** 验证者在输入 x 上允许随机访问位串 π。如果 $x\in L$，则存在位串 π 使得验证者接受；但如果 $x\notin L$，则验证者至少以 1/2 的概率拒绝任意一个证明 π

完备性：如果 $x\in L$，则存在一个证明 $\pi\in\{0,1\}^*$ 使得 $\Pr[V^{\pi}(x)=1]=1$。这种位串 π 称为 x 的正确证明。

可靠性：如果 $x\notin L$，则对于任意 $\pi\in\{0,1\}^*$ 均有 $\Pr[V^{\pi}(x)=1]\leqslant 1/2$。

我们称语言 L 属于复杂性类 **PCP**$(r(n), q(n))$，如果有常数 $c, d>0$ 使得 L 存在一个 $(c\cdot r(n), d\cdot q(n))$-**PCP** 验证者。

PCP 定理断言，任意 **NP** 语言都存在非常高效的 **PCP** 验证者。

定理 11.5（**PCP** 定理[AS92, ALM+92]）　**NP**=**PCP**$(\log n, 1)$。

⊖ **PCP** 定理被发现后，报纸上相关新闻报道采用了"长数学证明找到了新捷径(New shortcut found for long math proofs)"作为标题。

⊖ 更精确的形式化定义，请参见习题 1.9 讨论的随机访存图灵机，或者参见 5.5 节讨论的神喻图灵机。

评注 11.6 下面依次给出几点注记。

1. 可靠性条件要求，如果 $x\notin L$，则验证者必须至少以 1/2 的概率拒绝任意一个证明。建立可靠性条件是定理证明中最难的部分。

2. "可被(r,q)-验证者查验的证明的长度至多为 $q2^r$"这一限制虽然看似不合情理，但是验证者根据 2^r 个具有非零概率的随机串之一来查看证明中相应的二进制位，于是她将至多只能涉及证明串中的 $q2^r$ 个位置上的二进制位。

3. 注意，$\mathbf{PCP}(r(n),q(n))\subseteq\mathbf{NTIME}(2^{O(r(n))}q(n))$。这是由于，非确定型图灵机可以在 $2^{O(r(n))}q(n)$ 时间内猜测证明并通过"穷举验证者的 $2^{O(r(n))}$ 种可能的随机选择"以确定型方式运行验证者来验证猜测的证明是否正确。如果验证者在所有可能的随机选择上都接受，则非确定型图灵机接受。

特别地，$\mathbf{PCP}(\log n,1)\subseteq\mathbf{NTIME}(2^{O(\log n)})=\mathbf{NP}$。这就证明了 **PCP** 定理中平凡的包含关系。

4. 在 **PCP** 定理的表述中，不同 **NP** 语言的验证者可以查询证明中不同数量的二进制位（只要查询的二进制位的数量是常数即可）。然而，由于任意 **NP** 语言都可以在多项式时间内归约到 SAT，因此所有 **NP** 语言的验证者查询的证明中二进制位的数量存在一个通用的上界，这个上界就是 SAT 语言的验证者查询的证明中二进制位的数量。

5. 在定义 11.4 的可靠性条件中，常数 1/2 可以是任意的。也就是说，将它修改为其他小于 1 的任意正常数之后，所定义的语言构成的复杂性类不会改变。事实上，对于可靠性参数为 1/2 且使用 r 个随机二进制位和 q 个查询的 **PCP** 验证者，只需将它重复运行 c 次，则可以得到一个可靠性参数为 2^{-c} 且使用 cr 个随机二进制位和 cq 个查询的 **PCP** 验证者（参见习题 11.1）。

例 11.7 为了让读者更清晰地看到 **PCP** 证明系统的样子，我们给出两个非平凡的 **PCP** 证明系统。

1. 所有不同构的两个图形成的序对构成语言 GNI，它属于 $\mathbf{PCP}(\mathrm{poly}(n),1)$。不妨设 GNI 成员资格判定问题的输入是$\langle G_0,G_1\rangle$，其中 G_0，G_1 都有 n 个顶点。验证者希望 π 包含如下信息。对于任意具有 n 个顶点的标号图 H，二进制位 $\pi[H]$ 表明 $H\equiv G_0$ 或 $H\equiv G_1$（如果二者都不成立，则 $\pi[H]$ 可以是任意的值）。也就是说，π 是二进制位构成的一个指数长的数组，其中的每个二进制位都由一个可能的 n 顶点图（的邻接矩阵）来标识。

242

验证者随机选择 $b\in\{0,1\}$ 和一个随机排列。她先将排列作用到 G_b 的所有顶点上，得到一个同构于 G_b 的图 H。然后，她查询 π 中由 H 标识的二进制位。最后，她接受当且仅当她查询得到的二进制位恰好是 b。

显然，如果 $G_0\not\equiv G_1$，则证明 π 能让验证者以概率 1 接受；如果 $G_0\equiv G_1$，则证明 π 能让验证者接受的概率至多为 1/2。

2. 第 8 章中介绍的交互式证明协议可以用来证明，积和式存在具有多项式规模的随机串和查询个数的 **PCP** 证明系统。同样，该系统中，证明也具有指数长度。 ◀

事实上，上述两个 **PCP** 证明系统都是如下定理的特殊情况。

定理 11.8（**PCP** 定理的放大[BFLS91，ALM+92，AS92]） $\mathbf{PCP}(\mathrm{poly}(n),1)=\mathbf{NEXP}$。

上面的论述中，$\mathbf{PCP}(\mathrm{poly}(n),1)$ 表示类 $\bigcup_{c>0}\mathbf{PCP}(n^c,1)$。定理 11.8 可以视为"放大"后的 **PCP** 定理。我们略去定理 11.8 的证明，因为它的证明过程仅采用了证明原始

PCP 定理时用到的类似技术和定理 8.19(IP=PSPACE)。

11.2.2　PCP 定理与近似难度

用另一种观点看，PCP 定理证明了：对很多 NP 优化问题，计算它们的近似解并不比计算它们的精确解更容易。

为使论述更具体，下面将讨论的焦点放在 MAX-3SAT 上。1992 年之前，人们一直无法回答如下问题：对于任意 $\rho<1$，能否为 MAX-3SAT 问题找到多项式时间的 ρ-近似算法？可以证明，PCP 定理意味着上述问题的答案是"否"(除非 P=NP)。这是由于，PCP 定理可以等价地表示为如下定理。

定理 11.9　(PCP 定理：近似的难度)　*存在 $\rho<1$，使得对任意 $L\in$ NP 均存在多项式时间函数 f 将所有位串映射为 3CNF 公式(的恰当表示形式)使得*

$$x\in L \Rightarrow \mathrm{val}(f(x))=1 \tag{11.1}$$

$$x\notin L \Rightarrow \mathrm{val}(f(x))<\rho \tag{11.2}$$

由上述定理，立刻可以得到下面的推论。

推论 11.10　*存在常数 $\rho<1$，使得如果 MAX-3SAT 存在多项式时间的 ρ-近似算法，则 P=NP。*

事实上，定理 11.9 给出了一种方法，对于任意 $L\in$ NP，该方法能将 MAX-3SAT 的 ρ-近似算法 A 转变成一个判定 L 的算法。这是因为(11.1)式和(11.2)式放在一起意味着，$x\in L$ 当且仅当 $A(f(x))$ 得到的赋值至少能满足 $f(x)$ 的所有子句中 ρ 比例的子句。

我们将在后面的第 22 章中证明一个更强的 PCP 定理。这个源自哈斯塔德的定理表明，对于任意 $\epsilon>0$，如果 MAX-3SAT 存在多项式时间的 $(7/8+\epsilon)$-近似算法，则 P=NP。因此，例 11.2 中介绍的 MAX-3SAT 的近似算法很可能是最优的。PCP 定理(以及由它推 243
导产生的其他 PCP 定理)为许多重要的问题建立了近似难度方面的一大批结果。此外，它还经常用于证明"已知的近似算法是最优的，除非 P=NP"这样的结论。

为什么库克-勒维定理不足以证明定理 11.9

证明定理 11.9 时，首先就会想到利用库克-勒维定理(定理 2.10)中的归约，它曾帮助我们将任意 NP 问题归约到 3SAT。遗憾的是，这种归约无法得到定理 11.9 要求的函数 f，这主要是由于，由归约得到的函数不满足性质(11.2)。事实上，习题 11.11 要求读者证明，对于归约产生的布尔公式，总存在一个能够满足几乎所有子句的赋值。(这正是本章开头帕帕迪米特里奥和杨纳卡卡斯的引言所表达的含义。)因此，$\mathrm{val}(\cdot)$ 在归约产生的布尔公式上几乎总接近于 1，但定理 11.9 却要求 $\mathrm{val}(\cdot)<\rho$ 对其中的一种情况总成立。

11.3　两种观点的等价性

现在，我们证明 PCP 定理的"交互式证明的观点"和"近似难度的观点"之间的等价性。亦即，我们证明定理 11.5 和定理 11.9 是等价的。为此，我们引入*约束满足问题*(Constraint Satisfaction Problem 或简写为 CSP)的概念。该问题是 3SAT 问题的推广，它出现在大量的应用中，它还将在 PCP 定理的证明中发挥重要作用。CSP 问题以如下形式推广了 3SAT 问题：它允许布尔公式中的每个子句具有任意的形式(而不仅仅是将 OR 操作作用在文字上)，并且每个子句也不仅仅限于使用 3 个变量。

定义 11.11（约束满足问题(CSP)）　设 q 是自然数。qCSP 的实例 φ 是一族从 $\{0,1\}^n$ 到 $\{0,1\}$ 的函数 $\varphi_1,\cdots,\varphi_m$（这族函数也被称为约束），其中每个 φ_i 都依赖于输入中的至多 q 个位置。也就是说，对任意 $i\in[m]$，存在 $j_1,\cdots,j_q\in[n]$ 和 $f:\{0,1\}^q\to\{0,1\}$ 使得 $\varphi_i(\boldsymbol{u})=f(u_{j_1},\cdots,u_{j_q})$ 对任意 $\boldsymbol{u}\in\{0,1\}^n$ 成立。

如果赋值 $\boldsymbol{u}\in\{0,1\}^n$ 使得 $\varphi_i(\boldsymbol{u})=1$，则称 $\boldsymbol{u}$ 满足约束 φ_i。$\boldsymbol{u}$ 满足的约束占所有约束的比例是 $\dfrac{\sum_{i=1}^{m}\varphi_i(\boldsymbol{u})}{m}$，我们令 $\mathrm{val}(\varphi)$ 表示上述比例在 $\boldsymbol{u}\in\{0,1\}^n$ 上取到的最大值。如果 $\mathrm{val}(\varphi)=1$，则称 φ 是可满足的。q 称为 φ 的约束参数。

例 11.12　3SAT 是 qCSP 在 $q=3$ 时的特例，它的每个约束都是将 OR 操作作用到该约束的所有文字上。◀

注记

1. 我们将 qCSP 的实例 φ 的大小定义为 φ 中约束的个数 m。由于不出现在任何约束中的变量是冗余的，故我们总假定 $n\leqslant qm$。此外，还需要注意，在 n 个变量上具有 m 个约束的任意 qCSP 实例可以用 $O(mq\log n2^q)$ 个二进制位来描述。在我们感兴趣的所有情况中，q 将是独立于 n，m 的常数。

244

2. qCSP 的优化形式 MAXqCSP 要求计算给定实例中能够被同时满足的约束的最大个数。3SAT 问题的简单的基于贪心思想的近似算法可以推广用于近似求解 MAXqCSP。对于 qCSP 的包含 m 个约束的任意实例 φ，推广后的算法将输出满足 $\dfrac{\mathrm{val}(\varphi)}{2q}m$ 个约束的一个赋值。

11.3.1　定理 11.5 与定理 11.9 的等价性

现在，我们证明 **PCP** 定理的两种形式(定理 11.5 和定理 11.9)是等价的。具体地，我们将证明这两个定理都等价于 qCSP 的某种鸿沟形式的 **NP** 完全性。

定义 11.13（鸿沟 CSP）　对任意 $q\in\mathbf{N}$ 和 $\rho\leqslant 1$，定义问题 ρ-GAPqCSP 是判定给定的 qCSP 实例 φ 是否满足：(1) $\mathrm{val}(\varphi)=1$（此时称 φ 是 ρ-GAPqCSP 的"YES"实例）或 (2) $\mathrm{val}(\varphi)<\rho$（此时称 φ 是 ρ-GAPqCSP 的"NO"实例）。

我们称 ρ-GAPqCSP 是 **NP** 难的，如果对任意 $L\in\mathbf{NP}$ 均存在一个多项式时间函数 f 将任意串映射为 qCSP 的实例(的任意表示)使得

$$\text{完备性：}x\in L\Rightarrow\mathrm{val}(f(x))=1$$
$$\text{可靠性：}x\notin L\Rightarrow\mathrm{val}(f(x))<\rho$$

定理 11.14　存在常数 $q\in\mathbf{N}$ 和 $\rho\in(0,1)$ 使得 ρ-GAPqCSP 是 **NP** 难的。

下面我们证明定理 11.5、定理 11.9 和定理 11.14 相互等价。

定理 11.5 蕴含定理 11.14

假设 $\mathbf{NP}\subseteq\mathbf{PCP}(\log n,1)$，往证 1/2-GAP$q$CSP 在某个 q 上是 **NP** 难的。这仅需在某个 q 上将一个 **NP** 完全语言(如 3SAT)归约到 1/2-GAPqCSP 上。根据我们的假设，3SAT 存在 **PCP** 系统，它的验证者仅做常数次查询，我们将查询的次数记为 q。并且，验证者也仅用 $c\log n$ 个随机二进制位，其中 c 是常数。给定任意输入 x 和 $r\in\{0,1\}^{c\log n}$，如果验证者

在输入 x 和随机串 r 上接受证明 π，则我们定义 $V_{x,r}(\pi)=1$；否则，定义 $V_{x,r}(\pi)=0$。这样就得到函数 $V_{x,r}(\cdot)$，它将证明 π 映射到 1 或 0。注意，$V_{x,r}$ 仅依赖于 π 中的至多 q 个位置。于是，对于任意 $x\in\{0,1\}^*$，函数族 $\varphi=\{V_{x,r}\}_{r\in\{0,1\}^{m'}}$ 是一个多项式规模的 qCSP 实例。而且，由于 V 的运行时间是多项式，因此将 x 转换成 φ 也仅需多项式时间。根据 **PCP** 系统的完备性和可靠性，如果 $x\in$3SAT，则 φ 将满足 $\mathrm{val}(\varphi)=1$；但如果 $x\notin$3SAT，则 φ 将满足 $\mathrm{val}(\varphi)\leqslant 1/2$。■

定理 11.4 蕴含定理 11.5

假设在常数 q 和 $\rho<1$ 上，ρ-GAPqCSP 是 **NP** 难的。那么，对任意 $L\in$**NP**，我们很容易由此构造一个 **PCP** 系统使得它恰好使用 q 个查询，具有 ρ-可靠性，并且使用对数空间的随机二进制位。事实上，对于任意输入 x 上，验证者可以运行归约映射 $f(x)$ 得到 qCSP 的一个实例 $\varphi=\{\varphi_i\}_{i=1}^m$。验证者希望"$x$ 的证明 π"恰好是 φ 中所有变量的赋值。为了验证 π 是否满足 φ，验证者随机选择 $i\in[m]$ 并(用通过 q 次查询获取 π 中的 q 个二进制位来)验证 φ_i 是否被满足。显然，如果 $x\in L$，则验证者接受的概率为 1；但如果 $x\notin L$，则验证者接受的概率至多为 ρ。将随机二进制位的个数和查询次数乘以某个相同的倍数，则可靠性可以被放大为 1/2(参见习题 11.1)。■

245

定理 11.9 等价于定理 11.14

由于 3CNF 公式是 3CSP 的特殊实例，因此定理 11.9 蕴含定理 11.4。下面，我们证明逆命题。

设 $\varepsilon>0$ 和 $q\in$**N** 使得 $(1-\varepsilon)$-GAPqCSP 是 **NP** 难的，亦即定理 11.4 成立。设 φ 是 qCSP 在 n 个变量上具有 m 个约束的实例。φ 的每个约束 φ_i 都可以表示为至多 2^q 个子句的 AND，其中的每个子句又至多是 q 个变量(或它们的否定)的 OR。φ 的所有约束在上述表示下对应的子句的总个数不超过 $m2^q$，将所有这些子句构成的集合记为 φ'。如果 φ 是 $(1-\varepsilon)$-GAPqCSP 的"YES"实例(亦即，φ 是可满足的)，则存在满足 φ' 中所有子句的一个赋值。如果 φ 是 $(1-\varepsilon)$-GAPqCSP 的"NO"实例，则任意赋值都至少使得 φ 中 ε 比例的约束不被满足，进而 φ' 中至少 $\frac{\varepsilon}{2^q}$ 比例的子句不被满足。利用第 2 章中库克-勒维定理(定理 2.10)所采用的技术，定义在 q 个变量 u_1，…，u_q 上的每个子句 C 可以转变成定义于变量 u_1，…，u_q 和辅助变量 y_1，…，y_q 上的一组子句 C_1，…，C_q 使得：(1)每个 C_i 都是至多 3 个变量(或它们的否定)的 OR；(2)如果 u_1，…，u_q 满足 C，则存在 y_1，…，y_q 的一个赋值使得 y_1，…，y_q，u_1，…，u_q 同时满足 C_1，…，C_q；(3)如果 u_1，…，u_q 不满足 C，则对于 y_1，…，y_q 的任意赋值均使得某个 C_i 不被 y_1，…，y_q，u_1，…，u_q 满足。

将 φ' 中的至多 $m2^q$ 个子句都用上述方式变形，得到定义于 $n+qm2^q$ 个变量上的至多 $qm2^q$ 个子句，将这些子句构成的集合记为 φ''。注意，φ'' 是 3SAT 的实例。因此，我们得到一个归约，它将 φ 变换为 φ''。该归约具备完备性，这是由于，如果 φ 是可满足的，则 φ' 也是可满足的，进而 φ'' 也是。该归约具有定理希望的可靠性，这是由于，如果任意赋值都至少使得 φ 中 ε 比例的约束不被满足，则任意赋值至少使 φ' 中 $\frac{\varepsilon}{2^q}$ 比例的子句不被满足，进而任意赋值至少使 φ'' 中 $\frac{\varepsilon}{q2^q}$ 比例的子句不被满足。■

11.3.2　重新审视 PCP 的两种理解

前面证明了 **PCP** 定理的"交互式证明的观点"和"近似难度的观点"之间的等价性，这

种等价性非常有用。因此，根据表 11-1 列出的线索，重新审视这两种观点之间的等价性将大有裨益。

表 11-1　理解 PCP 定理的两种观点

交互式证明观点		近似难度的观点
PCP 验证者(V)	↔	CSP 实例 φ
PCP 证明(π)	↔	所有变量的一个赋值(w)
证明的长度	↔	变量的个数(n)
查询的个数(q)	↔	约束的个数(m)
随机二进制位的个数(r)	↔	约束的个数的对数($\log m$)
可靠性参数(一般取 1/2)	↔	NO 实例上 $\mathrm{val}(\varphi)$ 达到的最大值
定理 11.5(**NP**⊆**PCP**($\log n$, 1))	↔	定理 11.14(ρ-GAPφCSP 是 **NP** 难的) 定理 11.9(**MAX-3SAT** 的 ρ 近似是 **NP** 难的)

246

11.4　顶点覆盖问题和独立集问题的近似难度

除了 3SAT 问题和 CSP 问题之外，**PCP** 定理还可以得出许多其他问题的近似难度。作为示例，本节将证明最大独立集问题(MAX-INDSET)和最小顶点覆盖问题(MIN-VERTEX-COVER)的近似难度，其中前一个问题曾在第 2 章遇到而后一个问题曾在本章例 11.3 中遇到。注意，MAX-INDSET 的不可近似性强于 MIN-VERTEX-COVER 的不可近似性，因为它排除了 MAX-INDSET 在任意 $\rho<1$ 上存在 ρ-近似算法的可能性。

定理 11.15　*存在 $\gamma<1$ 使得为 MIN-VERTEX-COVER 计算 γ-近似是* **NP** *难的。对任意 $\rho<1$，为 MAX-INDSET 计算 ρ-近似解是* **NP** *难的。*

由于顶点覆盖是一个顶点集合，它包含了图中所有边的一个顶点，故顶点覆盖的补集是一个独立集。因此，最大独立集是最小顶点覆盖的补集。可见，这两个问题在精确求解时是等价的。但是，这并不意味着近似求解这两个问题也是等价的。将最小顶点覆盖的大小记为 VC，将最大独立集的大小记为 IS，我们已经看到 $\mathrm{VC}=n-\mathrm{IS}$。于是，最大独立集问题的一个 ρ-近似将得到一个大小为 $\rho\cdot\mathrm{IS}$ 的独立集，如果我们想用这个独立集来得到最小顶点覆盖问题的一个近似解，则得到的顶点覆盖的大小等于 $n-\rho\cdot\mathrm{IS}$。这样得出的 MIN-VERTEX-COVER 的近似解的近似比等于 $\frac{n-\mathrm{IS}}{n-\rho\cdot\mathrm{IS}}$。只要 IS 接近于 n，则这个近似比可以任意小。事实上，定理 11.15 表明，如果 **P**≠**NP**，则最大独立集问题和最小顶点覆盖问题的可近似性存在本质区别：我们在例 2.2 中已经看到 MIN-VERTEX-COVER 的一个多项式时间的 1/2-近似算法；但是，只要 **P**≠**NP**，则 MAX-INDSET 对任意 $\rho<1$ 都不存在 ρ-近似算法。

首先，我们用 **PCP** 定理证明：存在某个常数 ρ 使得最大独立集问题和最小顶点覆盖问题均不能在多项式时间内被 ρ-近似(除非 **P**=**NP**)。然后，我们再将近似鸿沟"放大"使得 ρ 小到足以使定理 11.15 中 MAX-INDSET 的不可近似性成立。

引理 11.16　*存在多项式时间内可计算的变换 f 将每个 3CNF 公式变换为一个图使得：(1)对于每个 3CNF 公式 φ，$f(\varphi)$ 都是一个 n-顶点图；(2)$f(\varphi)$ 的最大独立集的大小等于 $\mathrm{val}(\varphi)\frac{n}{7}$。*

证明概要　在每个 3CNF 公式上运用证明 INDSET 的 **NP** 完全性时采用的"规范"归约（参见定理 2.15 的证明过程）。注意，所得到的图满足引理要求的性质。我们将细节的验证留作习题 11.5。■

由此立刻得到下面的推论。

推论 11.17　如果 **P**≠**NP**，则存在常数 $\rho<1$，$\rho'<1$ 使得：MAX-INDSET 不能在多项式时间内被 ρ-近似，且 MIN-VERTEX-COVER 不能在多项式时间内被 ρ'-近似。 247

证明　令 L 是任意的 **NP** 语言。定理 11.9 表明，L 的成员资格判定问题可以归约到 MAX-3SAT。具体地讲，归约过程得到的 3CNF 公式 φ 要么是可满足的，要么 $\mathrm{val}(\varphi)<\rho$，其中 $\rho<1$ 是一个常数。然后，我们将引理 11.16 中的归约应用到 3CNF 公式 φ 上，所得 MAX-INDSET 实例的 ρ-近似解可以转换为 MAX-3SAT 实例 φ 的 ρ-近似解。由此可知，MAX-INDSET 的 ρ-近似是 **NP** 难的。

要得到 MIN-VERTEX-COVER 的不可近似性，需要注意到：对于上一自然段的归约过程得到的图，它的最小顶点覆盖大小等于 $n-\mathrm{val}(\varphi)\frac{n}{7}$。由此可知，对于 $\rho'=6/(7-\rho)$，如果 MIN-VERTEX-COVER 存在 ρ'-近似，则我们在 $\mathrm{val}(\varphi)=1$ 的情况下将为归约得到的图找出一个大小为 $\frac{1}{\rho'}\left(n-\frac{n}{7}\right)$ 的顶点覆盖，其大小不超过 $n-\rho n/7$。由此可知，最小顶点覆盖问题的 ρ'-近似算法将使得我们能够区分 $\mathrm{val}(\varphi)=1$ 和 $\mathrm{val}(\varphi)<\rho$，但定理 11.9 表明区分这两种情况是 **NP** 难的。■

为了完成定理 11.15 的证明，我们需要为 MAX-INDSET 将近似鸿沟放大。类似的放大过程可以应用于许多组合优化问题，只要它们具有某种"自我改进"性。对于 MAX-INDSET，一种简单的自我改进性可以用图的乘积来获得。

证明　（定理 11.15）对于任意 n-顶点图 G，将 G^t 定义为 $\binom{n}{k}$ 个顶点上的一个图，其中每个顶点表示 G 的 k 个顶点构成的一个子集。如果两个子集 S_1，S_2 的并集 $S_1\cup S_2$ 是 G 的一个独立集，则 S_1，S_2 相邻。容易验证，G^t 的最大独立集对应于 G 的最大独立集的所有 k 元子集。因而，G^t 的最大独立集的大小等于 $\binom{\mathrm{IS}}{k}$，其中 IS 是 G 的最大独立集的大小。于是，对于推论 11.17 中归约产生的图，如果取它 k 次幂，则由 G 的最大独立集及其 ρ-近似独立集可以分别构造乘幂图上的最大独立集和近似最大独立集，这两个独立集的大小的比值为 $\binom{\rho^*\mathrm{IS}}{k}/\binom{\mathrm{IS}}{k}$，该比值约等于 ρ^k。取充分大的 k 值可以使得 ρ^k 小于任意给定的常数。此时，归约的时间复杂度将变为 n^k，它在任意固定的 k 上仍是一个多项式。■

评注 11.18（勒维归约（Levin Reduction））　在第 2 章中，我们曾定义：如果任意 $L\in$ **NP** 都可以归约到 L'，则称 L' 是 **NP** 难的。所采用的归约是一个满足"$x\in L\Leftrightarrow f(x)\in L'$"的多项式时间函数 f。在我们应用这种归约的所有情况中，为了证明"$x\in L\Rightarrow f(x)\in L'$"，我们的做法都是给出某种方法将"$x\in L$ 的证明"转换成"$x'\in L'$ 的证明"。虽然卡普归约的定义中并未要求证明之间转换的映射是高效的，但是在我们见过的所有卡普归约中这种映射都是高效的。类似地，为了证明"$f(x)\in L'\Rightarrow x\in L$"，我们也会给出某种方法将"$x'\in L'$ 的证明"转换成"$x\in L$ 的证明"。同样，我们的所有证明过程往往也会得到转换映射的高效计算方法。我们将具有上述性质的归约称为勒维归约（参见定理 2.18 的证明）。

值得注意的是，本章讨论的 PCP 归约也满足上述性质。例如，对于定理 11.16 证明过程中得到的归约，它实际上不仅仅将 CNF 公式 φ 映射为一个图 G 使得 φ 是可满足的当且仅当 G 中存在一个"大"独立集，而且还表明了如何高效地将 φ 的一个满足性赋值转换成 G 的一个大独立集和如何将 G 的一个不太小的独立集转换为 φ 的一个满足性赋值。有关这一
248 点，第 22 章证明 PCP 定理时将更加清晰地进行阐述。

11.5　NP⊆PCP(poly(n)，1)：由沃尔什-哈达玛编码得到的 PCP

本节证明 PCP 定理的较弱形式，亦即，我们将证明：任意 NP 结论都存在一个具有指数长度的证明，使得它的真伪仅需查验其中的常数个二进制位即可得到验证。此外，本节给出的各种技术还展示了各种 PCP 定理的证明技巧，这些技术在第 22 章中用于证明完整的 PCP 定理。

定理 11.19（NP 的指数规模 PCP 系统[ALM+92]）　NP⊆PCP(poly(n)，1)。

我们通过为一个 NP 完全语言设计一个恰当的验证者来证明上述定理。验证者希望拿到的"证明"是"普通证明"的某种编码，以确保验证者可以对这种编码后的证明进行概率查验。

11.5.1　线性测试与沃尔什-哈达玛编码

我们采用沃尔什-哈达玛(Walsh-Hadamard)编码(参见 19.2.2 节，但此处的处理仍是自包含的)，这种编码方法用 GF(2) 上的 n 个变量构成的线性函数来编码长度为 n 的位串。编码函数 $\mathrm{WH}:\{0,1\}^*\to\{0,1\}^*$ 将串 $u\in\{0,1\}^n$ 映射为函数 $x\mapsto u\odot x$ 的真值表，其中对任意 $x,y\in\{0,1\}^n$ 定义 $x\odot y=\sum_{i=1}^{n}x_iy_i \pmod 2$。注意，这是一个十分低效的编码方法，因为每个含有 n 个位的串 $u\in\{0,1\}^n$ 被编码为 $|\mathrm{WH}(u)|=2^n$ 个二进制位。如果 $f\in\{0,1\}^{2^n}$ 等于 $\mathrm{WH}(u)$ 对某个 u 成立，则称 f 是沃尔什-哈达玛编码的一个码字。这样的一个串 $f\in\{0,1\}^{2^n}$ 也可以视为从 $\{0,1\}^n$ 到 $\{0,1\}$ 的一个函数。

在后面的讨论中，我们将反复利用如下事实(参见论断 A.31)。

随机子和原理：如果 $u\neq v$，则有 1/2 的 x 满足 $u\odot x\neq v\odot x$。

随机子和原理表明，沃尔什-哈达玛编码是一种最小距离为 1/2 的纠错码。也就是说，对于任意 $u\neq v\in\{0,1\}^n$，$\mathrm{WH}(u)$ 和 $\mathrm{WH}(v)$ 至少在 1/2 的二进制位上是相异的。下面，我们讨论沃尔什-哈达玛编码的局部检测(亦即，仅使用 $O(1)$ 个查询的检测)。

沃尔什-哈达玛编码的局部检测

假设我们可以获取函数 $f:\{0,1\}^*\to\{0,1\}$ 在任意输入上的函数值，并且希望检测 f 是否为沃尔什-哈达玛编码的一个码字。由于沃尔什-哈达玛所有码字恰好是从 $\{0,1\}^n$ 到 $\{0,1\}$ 的所有线性函数，因此为了检测 f 我们可以在 2^{2n} 对 $x,y\in\{0,1\}^n$ 上分别计算

$$f(x+y)=f(x)+f(y) \tag{11.3}$$

其中(11.3)式左端的"+"是 FG(2)^n 上的加法，而右端的"+"则是 GF(2) 上的加法。上述检测在定义上可行，但是实际检测过程需要获取 f 的 2^n 个函数值。

仅获取 f 的常数个函数值，能否实现对 f 的检测呢？一种自然的想法是在随机选取的
249 x,y 上验证(11.3)式是否成立。显然，在这种局部测试下，线性函数通过检测的概率等于 1。但是，现在我们却无法高概率地确保任意的非线性函数均无法通过测试！例如，如

果 f 非常接近于一个线性函数，比方说 f 是将线性函数在一小部分输入上的函数值修改之后得到的函数，则上述的局部检测遇到函数的非线性部分的概率非常小，因此这种局部检测方法无法将 f 从线性函数中区分出来。于是，我们略微地降低目标：我们的检测方法一方面应该让所有线性函数通过检测，另一方面它应该以较高的概率确保距离线性函数非常远的函数无法通过检测。这样，上述自然的检测方法就足以胜任了。

定义 11.20　令 $\rho\in[0,1]$，如果函数 $f,g:\{0,1\}^n\to\{0,1\}$ 满足 $\Pr_{x\in_R\{0,1\}^n}[f(x)=g(x)]\geqslant\rho$，则称它们是 ρ-接近的。对于函数 f，如果存在一个线性函数 g 使得 f 和 g 是 ρ-接近的，则称 f 是 ρ-接近于线性函数的。

定理 11.21　（线性检测[BLR90]）　如果 $f:\{0,1\}^n\to\{0,1\}$ 在某个 $\rho>1/2$ 上满足

$$\Pr_{x,y\in_R\{0,1\}^n}[f(x+y)=f(x)+f(y)]\geqslant\rho$$

则 f 是 ρ-接近于线性函数的。

我们将定理 11.21 的证明推迟到第 22 章的第 22.5 节。对于任意 $\delta\in(0,1/2)$，随机独立地对(11.3)式进行 $O(1/\delta)$ 次测试，则我们得到一个线性测试，它至少以 1/2 的概率拒绝非$(1-\delta)$-接近于线性函数的任意函数。我们将这种测试称为$(1-\delta)$-线性测试。

沃尔什-哈达玛编码的局部解码

假设对于 $\delta<\frac{1}{4}$，函数 $f:\{0,1\}^n\to\{0,1\}$ 与某个线性函数 $\overline{f}$ 是$(1-\delta)$-接近的。由于任意两个线性函数至少在一半的输入上取相异的函数值，函数 $\overline{f}$ 由函数 f 唯一地确定。假设给定 $x\in\{0,1\}^n$ 并且可以查询函数 f 在任意输入上的函数值，仅查询 f 在常数个输入上的函数值，我们能否计算得出 $\overline{f}(x)$ 呢？一个直白的答案是，由于多数的 x 均满足 $f(x)=\overline{f}(x)$，我们仅需查询 f 在 x 上的函数值应该就可以高概率地得到 $\overline{f}(x)$。问题是，x 也很可能是使得 f 和 $\overline{f}$ 取不同的值的自变量！幸运的是，我们仍然有一种简单的方法来得到 $\overline{f}(x)$，它仅需查询 f 在两个输入上的函数值：

1. 随机选择 $x'\in_R\{0,1\}^n$；
2. 令 $x''=x+x'$；
3. 取 $y'=f(x')$ 且 $y''=f(x'')$；
4. 输出 $y'+y''$。

由于 x' 和 x'' 各自都服从均匀分布(尽管它们不是独立的)，因此由合并界限可知，"$y'=\overline{f}(x')$ 且 $y''=\overline{f}(x'')$"的概率至少为 $1-2\delta$。再由 $\overline{f}$ 的线性可知，$\overline{f}(x)=\overline{f}(x'+x'')=\overline{f}(x')+\overline{f}(x'')$。于是，$\overline{f}(x)=y'+y''$ 的概率至少为 $1-2\delta$。(上述推导中用到了 $a+b=a-b$ 在 GF(2)中恒成立这一事实。)上述技术称为沃尔什-哈达玛编码的局部解码，因为这种技术使我们仅需查询常数次函数的值就能够从损坏的码字(函数 f)中恢复得到正确的码字(线性函数 $\overline{f}$)。沃尔什-哈达玛编码是一种著名的自纠错编码。 250

11.5.2　定理 11.19 的证明

我们将为一个特定的 **NP** 完全语言 L 构造一个$(\mathrm{poly}(n),1)$-验证者证明系统。由此可得 **NP**$\subseteq$**PCP**$(\mathrm{poly}(n),1)$，因为任意的 **NP** 语言都可以归约到 L。我们要使用的 **NP** 完全语言是 QUADEQ，它是由 GF(2)$=\{0,1\}$ 上可被满足的所有二次方程组构成的语言。

例 11.22　下面给出了定义在变量 $u_1,\cdots,u_6$ 上的一个二次方程组，它是 QUADEQ

的一个实例。

$$\begin{aligned} u_1u_2 + u_3u_4 + u_1u_3 &= 1 \\ u_2u_3 + u_1u_4 &= 0 \\ u_1u_4 + u_2u_3 + u_3u_4 &= 1 \end{aligned}$$

上述实例是可满足的，因为所有变量全部赋 1 则可满足所有的方程。 ◀

QUADEQ 是 **NP** 完全的，因为它可以由 **NP** 完全语言 CKT-SAT 归约得到，其中 CKT-SAT 是所有可被满足的布尔线路构成的语言(参见 6.1.2 节)。归约的思想是，用一个变量表示布尔线路中每条线(wire)上的值(包括表示输入的线)，并将 AND 和 OR 表示为等价的二次多项式：$x \vee y = 1$ 当且仅当 $(1-x)(1-y)=0$，如此等等。归约的细节留作习题 11.15。

由于 $u_i = (u_i)^2$ 在 GF(2)上恒成立，因此我们假设所有方程不含形如 u_i 的单项式。亦即，方程中所有的单项式都恰好是二次的。于是，变量 u_1，…，u_n 上的 m 个二次方程可以表示为一个 $m \times n^2$ 的矩阵 A 和一个 m 维向量 $\boldsymbol{b}$(矩阵和向量都定义在 **GF**(2)上)。这样，QUADEQ 的成员资格判定问题可以重述为如下的计算任务：给定 A，$\boldsymbol{b}$，找出一个 n^2 维的向量 U 使它满足(1)$AU=\boldsymbol{b}$；并且(2)U 是某个 n 维向量 $\boldsymbol{u}$ 的张量积 $\boldsymbol{u}\otimes\boldsymbol{u}$。[⊖]



图 11-2　**PCP** 证明系统包括 WH($\boldsymbol{u}$)和 WH($\boldsymbol{u}\otimes\boldsymbol{u}$)($\boldsymbol{u}$ 是某个向量)，其中二次方程组的集合是可满足的。验证者先查验证明接近这个形式，再用沃尔什-哈达玛局部解码来确保 $\boldsymbol{u}$ 是二次方程实例的一个解。图中点域代表损坏的坐标

PCP 验证者

现在，我们给出 QUADEQ 的 **PCP** 系统(参见图 11-2)。设 A，$\boldsymbol{b}$ 是 QUADEQ 的一个实例，并假设 A，$\boldsymbol{b}$ 可以被赋值 $\boldsymbol{u} \in \{0,1\}^n$ 满足。验证者 V 可以访问证明 $\pi \in \{0,1\}^{2^n+2^{n^2}}$，其中证明 π 被我们翻译成两个函数 f：$\{0,1\}^n \to \{0,1\}$ 和 g：$\{0,1\}^{n^2} \to \{0,1\}$。在实例 A，$\boldsymbol{b}$ 的正确 PCP 证明 π 中，函数 f 是 $\boldsymbol{u}$ 的沃尔什-哈达玛编码，而函数 g 是 $\boldsymbol{u}\otimes\boldsymbol{u}$ 的沃尔什-哈达玛编码。这就是说，我们要设计的验证者 V 必须以概率 1 接受这种形式的证明，这样验证者 V 才满足完备性条件。分析过程将会反复用到随机子和原理。

251

第 1 步：验证 f，g 是线性函数。前面已经注意到，验证者仅用局部测试本身将无法达成验证目标。于是，我们让验证者对 f，g 都执行 0.999-线性测试。如果其中任何一个线性测试失败，则验证者立刻拒绝。

因此，如果 f 或 g 中任何一个不是 0.999-接近于线性函数，则验证者将高概率地拒绝。因此，在验证过程的后续部分中，我们假设存在两个函数 $\widetilde{f}$：$\{0,1\}^n \to \{0,1\}$ 和 $\widetilde{g}$：$\{0,1\}^{n^2} \to \{0,1\}$ 使得 $\widetilde{f}$ 是 0.999-接近于 f 的，而 $\widetilde{g}$ 是 0.999-接近于 g 的。(注意，对于正确的证明，线性测试成功的概率为 1，并且 $f=\widetilde{f}$ 而 $g=\widetilde{g}$。)

事实上，在第 2 步和第 3 步中，我们将假设验证者能够在任何必要的时刻查询 $\widetilde{f}$ 和 $\widetilde{g}$ 的函数值。这种假设的合理性在于，局部解码使得验证者高概率地恢复出所需的 $\widetilde{f}$ 和 $\widetilde{g}$ 的函数值。在第 2 步和第 3 步中，验证者仅对 $\widetilde{f}$ 和 $\widetilde{g}$ 的函数值进行少量次数(小于 20 次)的查

⊖ 如果 x，y 是两个 n 维向量，则它们的张量积，记为 $x\otimes y$，是一个 n^2 维的向量(或者说 $n\times n$ 的矩阵)，它的第 (i,j) 个元素是 x_iy_j(通过某种标准的方法将 $[n^2]$ 等同于 $[n]\times[n]$)。也请参见 21.3.3 节。

询。因此，局部解码以很高的概率(>0.9)为所有查询恢复出正确函数值。

关于记号的约定：为了简化记号，在后续部分中我们用 f，g 分别表示 $\overline{f}$，$\overline{g}$。(这是可以的，因为前面已经阐明：验证者可以根据需要来查询 $\overline{f}$，$\overline{g}$ 的函数值。)特别地，我们假设 f，g 都是线性函数，继而它们必然分别是某两个串 $u\in\{0,1\}^n$ 和 $w\in\{0,1\}^{n^2}$ 的编码。换句话说，f，g 分别是由 $f(r)=u\odot r$ 和 $g(z)=w\odot z$ 给定的函数。

第 2 步：验证 g 是 $u\otimes u$ 的编码，其中 $u\in\{0,1\}^n$ 是编码 f 得到的串。验证者用随机独立的二进制串将下面的测试重复执行 10 次："随机独立地从 $\{0,1\}^n$ 选取 r，r'，如果 $f(r)f(r')\neq g(r\otimes r')$，则终止并拒绝。"

在正确的证明中，$w=u\otimes u$，因此

$$f(r)f(r')=\Big(\sum_{i\in[n]}u_ir_i\Big)\Big(\sum_{j\in[n]}u_jr'_j\Big)=\sum_{i,j\in[n]}u_ir_iu_jr'_j=(u\otimes u)\odot(r\otimes r')$$

上式在正确证明中等于 $g(r\otimes r')$。故第 2 步不会拒绝正确证明。

下面假设，与正确证明的情况不同，$w\neq u\otimes u$。我们断言，在连续 10 次测试中，验证者 V 每次测试时"终止并拒绝"的概率至少为 $\frac{1}{4}$(进而，验证者至少在一次测试中"终止并拒绝"的概率的至少为 $1-(3/4)^{10}>0.9$)。事实上，令 W 是一个 $n\times n$ 的矩阵，其中的元素与 w 中的元素相同；令 U 是一个 $n\times n$ 的矩阵，其中 $U_{i,j}=u_iu_j$；将 r 视为行向量，将 r' 视为列向量。在这些记号下，

$$g(r\otimes r')=w\odot(r\otimes r')=\sum_{i,j\in[n]}w_{i,j}r_ir'_j=rWr'$$

$$f(r)f(r')=(u\odot r)(u\odot r')=\Big(\sum_{i=1}^{n}u_ir_i\Big)\Big(\sum_{j=1}^{n}u_jr'_j\Big)=\sum_{i,j\in[n]}u_ir_iu_jr'_j=rUr'$$

于是，V 拒绝当且仅当 $rWr'\neq rUr'$。随机子和原理意味着，如果 $W\neq U$，则至少有 1/2的 r 满足 $rW\neq rU$。对每个满足上述条件的 r 再用随机子和原理，可以知道，至少有 1/2的 r' 满足 $rWr'\neq rUr'$。由此可得，测试过程至少在 1/4 的 r，r' 序对上拒绝。 252

第 3 步：验证 g 是满足性赋值的编码。利用前两步中对 f，g 所做的所有验证，不难验证某个特定的方程(不妨设第 k 个方程)能够被 u 满足，亦即

$$\sum_{i,j}A_{k,(i,j)}u_iu_j=b_k \tag{11.4}$$

用 z 表示 n^2 维向量 $(A_{k,(i,j)})$，其中 i，j 取遍 $1,\cdots,n$。可以看到，(11.4)式左端恰好是 $g(z)$。由于验证者知道 $(A_{k,(i,j)})$ 和 b_k，因此验证者可以查询 g 在 z 上的函数值，并验证 $g(z)=b_k$。

上述验证思想的缺点在于，为了验证 u 满足整个方程，验证者需要对 $k=1,2,\cdots,m$ 依次查询函数 g 的取值，但是 **PCP** 系统却要求查询次数必须独立于 m。幸运的是，我们可以再次使用随机子和原理来做到这一点！验证者随机选取一些方程，并在模 2 操作下计算这些方程的和(换句话说，对于 $k=1,2,\cdots,m$，将(11.4)式表示的方程两端同时乘以一个随机位，再对两端分别求和。)求和的结果是一个新的二次方程。随机子和原理意味着，如果 u 不满足原方程组中的某一个方程，则它至少以 1/2 的概率不满足求解得到的新方程。于是，验证者仅验证 u 是否满足这个新方程。

综上所述，我们得到了一个验证者 V，它使得：(1)如果 A，b 是可满足的，则 V 接受正确证明的概率为 1；(2)如果 A，b 是不可满足的，则 V 接受任意证明的概率至多为 0.8。为了在 A，b 是不可满足的情况下将接受任意证明的概率降低到 1/2，仅需简单地重

复执行上述的验证过程，这就完成了定理 11.19 的证明。 ■

本章学习内容

- 计算 **NP** 难问题的近似解是一个重要的研究方向。在许多有意义的 **NP** 难问题上，传统的库克-勒维定理未能排除近似算法的存在性。
- 许多 **NP** 难问题都能设计出非平凡的近似算法。
- **PCP** 定理给出了 **NP** 的一种新的概率性质，同时还证明了：如果 **P**≠**NP**，则 MAX-3SAT 不能被近似到任意的精度。事实上，上述两个结果互相等价。
- 选择不同的参数，还可以得到许多其他的 **PCP** 定理。在本章给出的这种例子中，验证者使用 poly(n)个随机二进制位，并且仅查验证明中的 $O(1)$个位。
- 各种 **PCP** 定理的证明都需要用某种有意义的方法来编码布尔公式的满足性赋值，同时还需要一种检验过程来查验任意串是否是一个编码串。本章给出的证明中采用的
253 是哈达玛(Hadamard)编码(它是仅用 GF(2)上的线性函数实现的一种编码)。

本章注记和历史

近似算法概念的提出早于 **NP** 完全性的发现。事实上，1966 年格莱翰(Graham)发表的论文[Gra66]已经给出了一种调度问题的近似算法，该问题后来才被证明是 **NP** 完全的。**NP** 完全性被发现后，约翰逊[Joh74]很快就将"计算近似解"这一问题形式化，给出了许多问题的一些简单的近似算法(比如 MAX-SAT 的 1/2-近似近似算法)，并提出了"是否能够找到更好的近似算法?"这样的问题。在接下来的 20 年内，人们虽然在证明近似求解问题的难度方面取得的结果屈指可数(比如萨尼(Sahni)和冈萨雷斯(Gonzalez)[SG76]证明了一般的 TSP 的近似难度)，并且设计出来的近似算法也为数不多，但是却逐渐认识到在证明近似难度时还缺乏严谨的技术。人们遇到的主要困难似乎是，没有明显的归约方法能够将计算问题相互归约并且保持计算问题的可近似性。帕帕迪米特里奥和杨纳卡卡斯的论文[PY88]在一大族计算问题之间实现了这种归约(他们将这族问题称为 MAX-SNP)并证明了 MAX-3SAT 是这族问题中的完全问题。这使得无论从算法设计的角度看，还是从近似难度证明的角度看，MAX-3SAT 都是一个很有吸引力的问题。

在这项工作之后，交互式证明领域的研究很快取得了一些进展(本书第 8 章涉及了其中一部分内容)。在当时看来，这些进展似乎与近似难度证明毫不相干。与本章内容最密切相关的结果源自巴拜、福特劳和伦德[BFL90]，他们证明了 **MIP**=**NEXP**。巴拜、福特劳、勒维和塞盖迪[BFLS91]很快又将这一结果改造到 **NP** 上。此后，人们迅速地得到了一系列结果。1991 年，费格、戈德瓦瑟、洛瓦兹、萨弗拉和塞盖迪等人给出了一个令人瞠目结舌的结果[FGL+91]。他们证明了，如果 SAT 不存在亚指数时间的算法，则对任意 $\epsilon>0$，MAX-INDSET 不可能在 $2^{\log^{1-\epsilon} n}$ 因子范围内被近似求解。这一结论首次建立了不可近似性与类似于 **PCP** 定理的结论之间的联系。然而，当时许多研究者都感觉(特别是，由于结论并未证明得出 **NP** 完全性本身)所采用的是一种"错误的方法"并且他们的结论在不使用交互式证明的条件下最终也能被证明。(更有趣的是，迪纳尔(Dinur)的鸿沟放大引理(参见第 22 章)推进了实现上述想法的步伐。)但是，一年之后，阿罗拉和萨弗拉[AS92]进一步细化了[BFL90]中的思想(并提出了单独构造验证者的想法)，证明了 MAX-INDSET 的近似求解问题实际上是 **NP** 完全的。同时，他们还用 **PCP** 系统证明了 **NP** 的一种新的出

人意料的性质，亦即，**NP**=**PCP**($\log n$, $\sqrt{\log n}$)。此时，他们已经逐渐清晰地认识到，如果查询次数能达到亚对数，则它也极有可能达到常数！在发表的下一篇论文中，阿罗拉、伦德、牟特瓦尼(Motwani)、苏丹(Sudan)和塞盖迪[ALM+92]完成了这一步骤(在此过程中引入了 11.5 节中具有常数查询次数的验证者以及其他的一些思想)，证明了 **NP**=**PCP**($\log n$, 1)，同时这一个结论也表明了 MAX-3SAT 的近似求解是 **NP** 完全的。自此以后，人们逐渐证明了许多其他的 **PCP** 定理，第 22 章将综述这些定理。注意，本章用 MAX-3SAT 的近似难度推导得出了 MAX-INDSET 的近似难度，但是历史上人们却先得出后者。

PCP 定理的 AS-ALMSS[⊖]证明的整体思想(事实上还包括证明 **MIP**=**NEXP** 的思想)类似于定理 11.19 的证明。事实上，写这本书时，原始证明的各个部分中，我们只保留了定理 11.19，第 22 章中给出的证明的其余部分源自后来由迪纳尔(Dinur)给出的证明。但是，AS-ALMSS 证明除了使用沃尔什-哈达玛编码之外，还用到了基于次数较低的多变量多项式的编码。这些编码方法也有类似于线性测试和局部解码的一些子程序，但是证明的正确性要比定理 11.19 要难一些。所有证明也都用到了源自自测试(Self-testing)和自纠错(Self-correcting)编程等相关专题的一些直觉思想[BLR90, RS92]。 254

PCP 定理得出了近似难度方面的一大批结果。关于这些结果，特雷韦桑(Trevisan)[Tre05]最近进行了综述，阿罗拉和伦德[AL95]早期也曾做过综述。

PCP 定理及其远亲结论 **MIP**=**NEXP** 都不是相对性结论[FRS88]。

本章仅论及了一些平凡的近似算法，这些算法并不能代表近似算法的研究现状。要了解目前已经获得的一些精巧的近似算法，请参阅霍赫鲍姆(Hochbaum)的专著[Hoc97]和瓦兹拉尼的专著[Vaz01]。

习题

11.1　证明：对于任意两个函数 r, q: **N**→**N** 和常数 $s<1$，将定义 11.4 中可靠性条件的 1/2 修改为 s 不会改变所定义的复杂性类 **PCP**(r, q)。

11.2　证明：对于任意语言 L，如果 L 有一个使用 r 个随机二进制位和 q 个自适应查询的 **PCP** 验证者，则 L 也存在一个使用 r 个随机二进制位和 2^q 个查询的标准验证者(亦即，非自适应型验证者)。

11.3　给出一个概率多项式时间算法使得，在每个子句恰含 3 个不同变量的任意 3CNF 公式 φ 上，算法输出的赋值至少能够满足 φ 中 7/8 比例的子句。

11.4　给出一个多项式时间的确定型算法使得它达到习题 11.3 中的近似要求。

11.5　证明引理 11.6。

11.6　证明：**PCP**(0, $\log n$)=**P** 且 **PCP**(0, poly(n))=**NP**。

11.7　令 L 是所有如下的序对$\langle A, k\rangle$构成的语言，其中 A 是 0/1 矩阵而 $k\in\mathbf{Z}$ 使得 perm(A)=k(参见 8.6.2 节)。证明：L 属于 **PCP**(poly(n), poly(n))。

11.8　([AS92])证明：如果 SAT∈**PCP**($r(n)$, 1)在 $r(n)=o(\log n)$时成立，则 **P**=**NP**。这就表明，在忽略常数的情况下，**PCP** 定理可能是最优的。

11.9　(具有对数空间验证者的一个简单的 **PCP** 定理)利用"正确的表格(tabelau)可以在对数空间内被验证"这一事实，我们可以将 **NP** 的性质精确地刻画为：

⊖ 前面提到的两篇论文[AS92]和[ALM+92]的所有作者名字的首字母。——译者注

NP=｛L：存在对数空间图灵机 M 使得 $x\in L$ 当且仅当 $\exists y$，M 接受(x, y)｝

注意，M 可以对 y 进行双向访问[⊖]。令 **L-PCP**$(r(n))$是如下的所有语言构成的复杂性类：语言的成员资格证明可以用对数空间图灵机概率地验证，验证过程中图灵机只使用 $O(r(n))$个随机二进制位并且仅对证明扫描一遍(用上面的术语讲，图灵机
255 可以对 x 进行双向访问，但对 y 却只能进行单向访问)。同 **PCP** 定理中一样，“成员资格证明的概率验证”指的是：如果 $x\in L$，则存在一个证明 y 使得图灵机以概率 1 接受；否则，图灵机将至少以概率 1/2 拒绝所有证明。证明：**NP**=**L-PCP**$(\log n)$。注意，不要假设 **PCP** 定理成立！

(上述简单的 **PCP** 定理隐含于利普顿的论文[Lip90]，习题中要求的证明源自梅克尔比克(Melkebeek)。)

11.10　设 **J-PCP**$(r(n))$的定义类似于 **L-PCP**$(r(n))$的定义，但只允许验证者读取成员资格证明中的某 $O(r(n))$个前后相继的二进制位(验证者自己能决定读取哪些二进制位)。证明：**J-PCP**$(\log n)\subseteq$**L**。

11.11　本题研究为什么库克-勒维定理的证明过程(2.3 节)使用的归约不足以证明 MAX-3SAT 的近似难度。回顾一下，对任意 **NP** 语言 L，我们都定义了一个归约 f 使得：如果 $x\in L$，则 $f(x)\in$3SAT。证明：存在 $x\notin L$ 使得 $f(x)\in$3SAT 是一个具有 m 个子句的公式并且 $f(x)$有一个赋值能同时满足多于 $m(1-o(1))$个子句，其中$o(1)$表示一个随$|x|$增大而趋于 0 的函数。

11.12　为背包问题给出一个 poly$(n, 1/\varepsilon)$时间的$(1+\varepsilon)$-近似算法。也就是说，给出一个算法使得，在给定的 $n+1$ 个数 a_1，…，$a_n\in\mathbf{N}$，$k\in[n]$上，算法输出一个子集 $S\subseteq[n]$使得$|S|\leqslant k$ 且 $\sum_{i\in S}a_i\geqslant\frac{\mathrm{opt}}{1+\varepsilon}$，其中 $\mathrm{opt}=\max_{S\subseteq[n],\,|S|\leqslant k}\sum_{i\in S}a_i$。

11.13　给出一个多项式时间算法使得，在给定的可满足的 2CSP 实例 φ(的位串表示)上，算法输出 φ 的一个满足性赋值。

11.14　给出一个 poly$(n, 2^q)$时间的确定型算法使得，在给定的具有 m 个约束的 qCSP 实例 φ(的位串表示)上，算法输出的赋值能够满足其中 $m/2^q$个约束。

11.15　证明：QUADEQ 是 **NP** 完全的。

11.16　考虑如下问题：给定 n 个有理系数线性方程构成的方程组，找出可被同时满足的方程子集使得子集中方程的个数达到最大值。证明：存在常数 $\rho<1$ 使得求上述问题的 ρ 近似子集是 **NP** 难的。

11.17　证明习题 11.16 中的论断在下述情况下仍成立：所有方程定义在 GF(2)上并且每
256 个方程仅含 3 个变量。

⊖ 亦即，既能读又能写。下文的“单向访问”值得是只能读。——译者注

第二部分

Computational Complexity: A Modern Approach

具体计算模型的下界

第 12 章

Computational Complexity: A Modern Approach

判 定 树

让每个人都不再说采取行动是艰难的……世界上最难的事情是做决定。

——弗朗茨·格里尔帕策(Franz Grillparzer)(1791—1872)

目前，图灵机计算能力相关的一些基本问题仍然远未解决。于是，为了用其他方式深刻地理解"高效计算"这一难以捉摸的概念，人们转而研究更简单的限制性更强的计算模型。而且，这些限制性更强的计算模型也很自然地出现在大量的应用中，甚至还出现在计算机科学之外的某些应用中。因此，研究这些模型的性质也有其固有的价值。

或许这些模型中最简单的就是判定树[⊖]。此时，布尔函数 f 的"复杂度"度量的是，在输入 x 上计算 $f(x)$ 时 x 中被查验的二进制位的个数。本章概述判定树方面的基本结果和一些未解决的问题。12.1 节定义判定树和判定树复杂性。同研究图灵机时一样，我们还会定义非确定型判定树和概率型判定树；12.2 节和 12.3 节将分别介绍它们。12.4 节给出了一些用于证明判定树下界的技术。我们还给出了姚期智的最小最大引理(Min Max Lemma，参见注记 12.8)。该引理在证明随机判定树的下界时非常有用；更一般地，该引理在证明其他计算模型的随机复杂度下界时也很有用。

12.1 判定树和判定树复杂性

设 $f:\{0,1\}^n \to \{0,1\}$ 是一个函数。f 的判定树是这样一棵树，每个内结点均标记为某个 x_i，并且它恰有两条分别被标记为 0 和 1 的出边，每个叶结点被标记为输出 0 或 1。判定树的每个顶点在输入 $x=x_1x_2\cdots x_n$ 上进行计算时，先根据结点自身的标记来检查输入中相应的二进制位 x_i。如果 $x_i=1$，则让计算从该顶点出发沿 1 边继续在相应的子树中进行；如果 $x_i=0$，则让计算从该点出发沿 0 边继续在相应的子树中进行。于是，每个输入对应判定树中的一条路径。叶结点处的输出值就是 $f(x)$。例如，图 12-1 中给出了多数函数(majority function)的判定树。

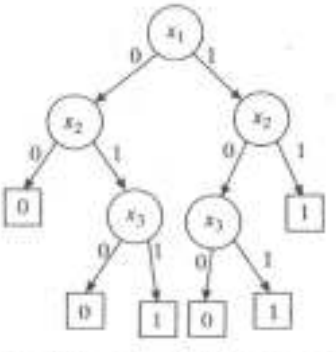


图 12-1 计算 3 个二进制位上多数函数 $Maj(x_1, x_2, x_3)$ 的一棵决策树。也就是说，如果输入的 3 个二进制位中至少有两个 1，则判定树的输出是 1；否则，判定树的输出是 0

判定树经常出现在医学诊断过程中。它可以紧凑地表示"由病症和检测数据得出诊断结论"这一过程。判定树也常出现在运筹学(用于描述"商业决策"算法)和机器学习(其目的是从实例数据中发现判定树)中。本章将判定树用作一种简单的计算模型，并证明一些非平凡的下界。

函数的判定树复杂性指的是用最有效的判定树来计算在最坏情况下输入中需要被查验

⊖ 判定树也称决策树。在复杂性理论中，通常称为判定树。——译者注

的二进制位个数。也就是说，我们有如下的定义。

定义 12.1 （判定树复杂度） 判定树 t 在输入 x 上的代价，记为 $\mathrm{cost}(t, x)$，指的是 x 中被树 t 查验的二进制位个数。

函数 f 的**判定树复杂度**定义为

$$D(f) = \min_{t \in \mathcal{T}_f} \max_{x \in \{0,1\}^n} \mathrm{cost}(t, x)$$

其中 $\mathcal{T}_f$ 表示计算函数 f 的所有判定树构成的集合。

由于 $\{0, 1\}^n$ 上的每个布尔函数都可以用一棵深度为 n 的完全二叉树（具有 2^n 个叶结点）来计算，因此 $D(f) \leqslant n$ 对任意 $f: \{0, 1\}^n \to \{0, 1\}$ 成立。我们感兴趣的事情是确定对各种有意义的函数，上述平凡的上界是最佳的，还是该函数存在更高效的判定树。

例 12.2 下面的三个例子给出了三个特定函数的判定树复杂度。

OR 函数。令 $f(x_1, x_2, \cdots, x_n) = \bigvee_{i=1}^{n} x_i$。此时，我们可以证明，任意判定树的深度都不可能小于平凡上界 n。为此，我们采用敌手论证法。设 t 是一棵计算 f 的判定树。我们设想，在 t 的计算过程中存在一个敌手，t 获得的每个输入二进制位都是由敌手告诉 t 的。于是，当 t 需要获得前 $n-1$ 个位中的每个位时，敌手都以 0 来作答。这样，判定树的输出结果总是“悬而未决”，直到第 n 个二进制位被告知，第 n 个位才最后确定了 OR 作用 260
在所有输入位上的结果是 1 还是 0。因此，$D(f) = n$。（将上面的论证过程表述为如下的另一种形式。我们证明了如果判定树在每个分支上至多处理 $n-1$ 个位，则必然存在 x，x' 使得它们在判定树中对应相同的路径，但是 $f(x) = 0$ 而 $f(x') = 1$。因此，判定树在 x 和 x' 之一上必然会给出错误的答案。）

图的连通性：假设给定一个 m-顶点图作为输入，它被表示成一个长度为 $n = \binom{m}{2}$ 的位串，其中如果 e 是 G 的一条边，则第 e 个二进制位等于 1；否则，这个位等于 0。我们希望知道，为了确定 G 是否连通（亦即，任意两个顶点均通过一条路径相连），判定树算法至少要查验邻接矩阵中的多少个二进制位。同样，我们证明任何判定树的深度不可能小于平凡上界 $\binom{m}{2}$。

我们再次运用敌手论证法。通过逐条地向图中添加边，敌手构造一个图来应对判定树算法的查询。在每个步骤上，敌手对之前的所有查询给出的答案定义了一个部分图，只要恰当地对后续查询作答，这个部分图就既可以被扩展成一个连通图，也可以扩展成一个非连通图。这样，判定树算法的输出结果总是“悬而未决”的，直到所有可能的边都被算法查询过为止。

当判定树每次对一条边 e 的查询时，敌手都以 0 作答（亦即，这条边不存在），除非这个答案使得当前的部分图变为非连通图，此时敌手以 1 作答。简单地运用数学归纳法，可以证明：敌手的上述策略确保当前的部分图是一片“森林”（亦即，该图恰由无公共顶点的若干棵树构成），并且直到我们希望的最后一条边被查询之后这片森林才会变成一个连通图。因此，只要判定树算法没有查询完所有可能的 $\binom{m}{2}$ 条边，则算法的输出就是“悬而未决”的。

AND-OR 函数。对任意 k，我们将 f_k 定义为以长度为 $n = 2^k$ 的位串为输入的如下函数：

$$f_k(x_1, \cdots, x_n) = \begin{cases} f_{k-1}(x_1, \cdots, x_{2^{k-1}}) \wedge f_{k-1}(x_{2^{k-1}+1}, \cdots, x_n) & \text{如果 } k \text{ 是偶数} \\ f_{k-1}(x_1, \cdots, x_{2^{k-1}}) \vee f_{k-1}(x_{2^{k-1}+1}, \cdots, x_n) & \text{如果 } k > 1 \text{ 是奇数} \\ x_n & \text{如果 } k = 1 \end{cases}$$

AND-OR 函数 f_n 可以用深度为 k 的布尔线路来计算(参见图 12-2)；但是，它的判定树复杂度为 2^k(参见习题 12.2)。

地址函数。假设 $n=k+2^k$，且函数 f 将 x_1，…，x_k，y_1，…，y_{2^k} 映射为 y_i(其中 i 是 $x_1\cdots x_k$ 表示的整数)；也就是说，函数 f 将输入的前 k(其中 $k\sim\log n$)个位作为索引来访问后 $n-k$ 位中的一个位。显然，这个函数有一棵深度为 $k+1$ 的判定树(它先查验输入的前 k 个位并根据查验结果来访问其余 $n-k$ 个位中的一个位)。因而，$D(f)\leqslant\log n+1$。另一方面，习题 12.1 表明 $D(f)\geqslant\Omega(\log n)$。

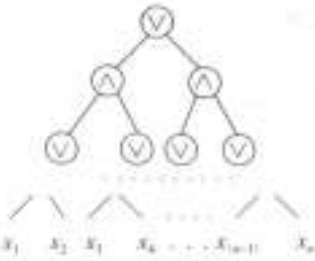


图 12-2　用布尔线路计算 AND-OR 函数，布尔线路有 k 层交替门，其中 $n=2^k$

◀

12.2　证明复杂性

下面我们引入证明复杂度(certificate complexity)[⊖]的概念。我们可以将它视为确定型判定树复杂性，这类似于区分确定型图灵机和非确定型图灵机时所采用的概念(参见第 2 章)。

定义 12.3　(证明复杂度)　设 $f:\{0,1\}^n\to\{0,1\}$ 且 $x\in\{0,1\}^n$。x 的 **0-证明**指的是子集 $S\subseteq[n]$，它使得 $f(x')=0$ 在满足 $x'|_S=x|_S$ 的任意 x' 上成立(其中 $x|_S$ 表示 x 中坐标编号属于 S 的所有二进制位构成的串)。类似地，x 的 **1-证明**指的是使得 $f(x')=1$ 在满足 $x'|_S=x|_S$ 的任意 x' 上成立的子集 $S\subseteq[n]$。

函数 f 的**证明复杂性**，记为 $C(f)$，定义为使得任意 x 均存在大小为 k 的 $f(x)$-证明的最小 k 值(注意，一个串不可能既存在 0-证明又存在 1-证明)。

如果 f 有一棵深度为 k 的判定树 t，则 $C(f)\leqslant k$；这是由于，t 在 x 上计算 $f(x)$ 时所查验的所有二进制位的位置编号恰好构成 x 的一个 $f(x)$-证明。因此，$C(f)\leqslant D(f)$。而且，在某些情况下，$C(f)$ 还会严格小于 $D(f)$。

例 12.4　确定例 12.2 中给出的某些函数的证明复杂度。

图的连通性：令 f 表示图连通性的判定函数。前面已经得到，对于 m-顶点图，f 的判定树复杂度是 $D(f)=\binom{m}{2}=\frac{m(m-1)}{2}$。图 G 的 1-证明是 G 中能表明"G 是连通图"的一组边构成的子集。因此，任意连通的 m-顶点图 G 都存在一个大小为 $m-1$ 的 1-证明(亦即，G 的任意生成树)。图 G 的 0-证明是这样一组边，这组边的缺失将表明 G 是非连通图；亦即，0-证明是图的割集。由于割集中边数在两边的顶点子集的大小相等时达到最大，故任意非连通的 m-顶点图都存在大小不超过 $(m/2)^2=m^2/4$ 的 0-证明。另一方面，某些图(例如，由两个分别包含 $m/2$ 个顶点的团构成的图)不存在更小的 0-证明。因此，$C(f)=m^2/4$。

AND-OR 函数。令 f_n 是输入长度 $n=2^k$ 的 AND-OR 函数。前面已经得到，$D(f)=2^k$。下面，我们证明 $C(f)\leqslant 2^{\lceil k/2\rceil}$，它约等于 $\sqrt{n}$。图 12-2 表明，f_n 可以用 k 层布尔线路来定

⊖ "Certificate Complexity"和第 15 章中的"Proof Complexity"都译为"证明复杂度"，它们在概念上是不同的，读者应根据上下文进行区分。之所以这样处理，是由于全书将"Certificate"和"Proof"都译为"证明"而未加区分。——译者注

义。每个层要么全部包含 OR 门，要么全部包含 AND 门，并且各层之间门的类别交替出现。最底层以 x 中的各个二进制位为输入，最顶层的单个逻辑门恰好输出答案 $f_n(x)$。如果 $f(x)=1$，则可以如下构造 x 的 1-证明。对于布尔线路中的每个 AND 门，我们需要证明它的两个孩子逻辑门都输出 1，而对于每个 OR 门仅需证明它有一个孩子逻辑门输出 1。因此，x 的 1-证明是布尔线路中的一棵子树，其中每个 AND 逻辑门都有两个孩子，而每个 OR 逻辑门却只有一个孩子。这意味着，x 的 1-证明仅需涉及 x 中的 $2^{\lceil n/2 \rceil}$ 个二进制位。 262
在 $f(x)=0$ 的情况下，只需将 OR 门和 AND 门的角色对调，将 1 和 0 的角色对调，类似的论证同样有效。 ◀

回顾一下，第 2 章(定义 2.1)曾将 **NP** 定义为"高效图灵机通过短证明能够确信 $f(x)=1$"的所有语言 f 构成的复杂性类。类似地，我们也可以将 1-证明视为判定树能够确信 $f(x)=1$ 的证明，进而得到如下的类比关系：

较低的判定树复杂度 $\leftrightarrow$**P**
较低的 1-证明复杂度 $\leftrightarrow$**NP**
较低的 0-证明复杂度 $\leftrightarrow$**coNP**

在这种类比关系下，下面的定理颇出人意料，因为它表明在判定树模型下"**P=NP∩coNP**"，这不同于图灵机模型下人们猜想的结论。

定理 12.5　对任意函数 f，均有 $D(f)\leqslant C(f)^2$。

证明　设 $f:\{0,1\}^n\to\{0,1\}$ 是满足 $C(f)=k$ 的函数。对任意 $x\in\{0,1\}^n$，用 S_x 表示 x 的 $f(x)$-证明，它是 $[n]$ 的一个 k 元子集。我们的证明将依赖于如下的观察结果：每个 1-证明必然与任意 0-证明相交。若不然，将存在这样一个串，它既存在 1-证明又存在 0-证明，但这是不可能的。

如下的判定树算法至多用 k^2 次查询就能得到 f 的函数值。算法维护一个集合 χ 来记录与所有已经进行的查询一致的所有输入。

> 初始时，$\chi=\{0,1\}^n$。如果存在某个 $b\in\{0,1\}$ 使得 $f(x)=b$ 对任意 $x\in\chi$ 成立，则算法终止并输出 b。否则，算法任取 $x_*\in\chi$ 使得 $f(x_*)=0$，然后查询 x_* 中由 S_{x_*} 标识的那些位置上所有目前仍未查询过的二进制位，从 χ 中删除在查询位置上与查询结果不一致的所有串 $x'\in\{0,1\}^n$。重复上述过程，直到 f 在 χ 中剩下的所有串上具有相同的函数值。

由于任意输入 x 均存在 1-证明或 0-证明来证明正确答案 $f(x)$，上述算法能够在每个输入上都最终输出正确答案。并且，当算法每次查询 0-证明中的二进制位时，所有 1-证明的大小至少减小 1。这是由于，正如前面的观察结果所述，每个 1-证明都与任意的 0-证明相交。(当然，如果某次查询的结果与一个 1-证明标识的所有位置都不一致，则这个 1-证明的大小将一次性减小到 0。同样的削减过程对 0-证明也成立。)因此，k 次迭代之后，所有 1-证明的大小将削减到 0。这意味着 χ 中剩下的所有串都有 0-证明，于是，算法能够正确地回答 0。由于每次迭代至多查询 k 个二进制位，故算法在至多进行 k^2 次查询之后终止。 ■

12.3　随机判定树

类似于图灵机模型的概率型图灵机，我们也可以定义随机判定树。在随机判定树中，输入中被查验的二进制位的位置每次都是随机选取的。更为方便的一种等价的刻画方法是 263
将随机判定树定义为所有确定型判定树上的一个概率分布。我们只考虑输出正确答案的随

机判定树，随机性的使用仅可能使它的期望代价降低(这恰好类似于复杂性类 **ZPP** 的情况，参见 7.3 节)。

定义 12.6（随机判定树）　对于任意函数 f，设$\mathcal{P}_f$表示计算 f 的所有判定树上的所有概率分布构成的集合。f 的**随机判定树复杂度**定义为

$$R(f)=\min_{\mathcal{R}\in\mathcal{P}_f}\max_{x\in\{0,1\}^n}E_{t\in_{\mathcal{R}}\mathcal{P}}[\mathrm{cost}(t,x)] \tag{12.1}$$

随机判定树复杂性刻画的是在所有判定树的所有可能的概率分布中，最佳概率分布在可能出现的最坏输入上的复杂性。显然，$R(f)\leqslant D(f)$，因为任意一棵确定型判定树都是一个特殊的概率分布。此外，不难证明 $R(f)\geqslant C(f)$，这是由于，在任意输入 $x\in\{0,1\}^n$ 和计算 f 的每棵判定树 t 上，x 都有一个大小为 $\mathrm{cost}(t,x)$ 的 $f(x)$-证明。因此，(12.1)式中的数学期望必然大于等于 x 的最小 $f(x)$-证明的大小(这就好比 **ZPP**$\subseteq$**NP**$\cap$**coNP**)。

例 12.7　考虑多数函数 $f=\mathrm{Maj}(x_1,x_2,x_3)$。可以直接得到 $D(f)=3$。下面，我们证明 $R(f)\leqslant 8/3$。我们采用如下的随机判定树，它取 x_1，x_2，x_3的一个随机排列，然后根据排列依次查验每个二进制位。如果查验过程得到两个相同的答案，则终止。如果输入串 x 中所有的二进制位都相同，则上述随机判定树在两次查验之后必然终止计算过程。如果有两个位相同而另一个位不同，不妨设 $x_1=1$ 而 $x_2=x_3=0$。于是，如果 x_1 在随机排列中位于最后，则随机判定树算法进行两次查验后就终止，而这种情况发生的概率等于 1/3；否则，随机判定算法必须查验所有的 3 个位。于是，随机判定树算法的期望代价为 $2\cdot(1/3)+3\cdot(2/3)=8/3$。在后面的例 12.9 中将会看到，我们实际上有 $R(f)=8/3$。◀

12.4　证明判定树下界的一些技术

我们在前面已经看到，敌手论证法可以用来证明确定型判定树复杂性的下界。然而，这种方法并不总是有效的，特别是在处理证明复杂性的下界和随机判定树复杂性的下界时，要证明这种下界，需要用到下面讨论的更复杂的技术。这些技术在复杂性理论中有很多其他的应用，在复杂性领域之外也有一些应用。

12.4.1　随机复杂性的下界

随机判定树是所有判定树上的概率分布，这是一种复杂的结构。因此，对随机判定树的
264 论述要比确定型判定树难得多。幸运的是，姚期智已经证明了：随机判定树复杂性的下界可以在确定型判定树的平均复杂性上进行推理获得。具体地讲，姚期智的最小最大引理(参见注记 12.8)表明，对于任意函数 f，如果我们能够找到输入集$\{0,1\}^n$上的一个概率分布$\mathcal{D}$使得 $E_{t\in_{\mathcal{R}}\mathcal{D}}[\mathrm{cost}(t,x)]\geqslant k$ 对计算 f 的任意确定型判定树 t 成立，则我们可以用 k 作为 $R(f)$的下界。换句话说，如果任意判定树 t 在服从$\mathcal{D}$的输入上的平均代价大于等于 k，则 $R(f)$以 k 为下界。也就是说，为了获得随机判定树复杂性的下界，我们不需要对判定树的概率分布和具体的输入进行论证，而只需对输入的概率分布和具体的判定树进行论证。

注记 12.8（姚期智最小最大引理）　姚期智最小最大引理广泛地用于证明随机算法复杂性下界。设 χ 是一个有限输入集合，而$\mathcal{A}$是求解该输入集合上某个计算问题 f 的有限个确定型算法构成的集合。对于 $x\in\chi$，$A\in\mathcal{A}$，我们用 $\mathrm{cost}(A,x)$表示算法 A 在输入 x 上的代价(该代价可以是运行时间、判定树复杂性等)。每个**随机算法**都可以视为$\mathcal{A}$上的一个概率分布$\mathcal{R}$。$\mathcal{R}$在输入 x 上的代价，记为 $\mathrm{cost}(\mathcal{R},x)$，是 $E_{A\in_{\mathcal{R}}\mathcal{R}}[\mathrm{cost}(A,x)]$。问题的**随**

机复杂性指的是

$$\min_{\mathcal{R}} \max_{x \in I} \mathrm{cost}(\mathcal{R}, x) \tag{12.2}$$

设$\mathcal{D}$是所有输入上的一个分布。对于任意确定型算法 A，A 在$\mathcal{D}$上的代价，记为 $\mathrm{cost}(A, \mathcal{D})$，是 $E_{x \in_R \mathcal{D}}[\mathrm{cost}(A, x)]$。问题的**分布复杂性**指的是

$$\max_{\mathcal{D}} \min_{A \in \mathcal{A}} \mathrm{cost}(A, \mathcal{D}) \tag{12.3}$$

姚期智引理断言(12.2)式和(12.3)式相等。根据零和博弈过程的冯·诺依曼最小最大引理(参见注记 19.1.2)，可以很容易地得出姚期智引理。该引理刻画了 min 操作和 max 操作在一定约束下可以交换顺序，这一特征使得它常常用于分析随机算法的复杂性下界。为此，我们需要在函数 f 的所有输入上定义一个恰当的分布$\mathcal{D}$(而"定义是否合理"依赖于引理使用者对问题的理解是否深入，有时也有运气成分)，然后再证明 f 的每个确定型算法在输入的上述分布上都有较大的代价 C。最后，根据姚期智引理可知，随机算法的复杂性至少为 C。

例 12.9 我们回过头来考虑 3 个二进制位上的多数函数 f，目的是找出 $R(f)$的一个下界。考虑输入串的如下概率分布：三个位相同的输入串(即 000 和 111)出现的概率等于 0，而其余每个输入串出现的概率都等于 1/6。对于任意的判定树(亦即，以任意顺序查验三个输入位)，查验到前两个位相等的概率为 1/3，因此，判定树的代价等于 2 的概率为 1/3。类似地，判定树的代价等于 3 的概率为 2/3。因此，对于计算多数函数的每棵判定树而言，在上述分布上完成计算时总代价的数学期望为 8/3。姚期智最小最大引理表明，$R(f) \geqslant 8/3$。结合例 12.7，我们得到 $R(f)=8/3$。 ◀ 265

12.4.2 敏感性

函数的敏感度是证明判定树复杂性下界的另一种方法。

定义 12.10 (敏感性和区组敏感性) 如果 $f: \{0,1\}^n \to \{0,1\}$是一个函数且 $x \in \{0,1\}^n$，则 f 在 x 上的**敏感度**，记为 $s_x(f)$，指的是满足 $f(x) \neq f(x^i)$的二进制位位置 i 的个数，其中 x^i 是将 x 的第 i 位翻转之后得到的位串。f 的**敏感度**，记为 $s(f)$，指的是$\max_x \{s_x(f)\}$。

f 在 x 上的**区组敏感度**，记为 $bs_x(f)$，指的是满足如下条件的最大 b：存在所有二进制位位置的不相交分组 $B_1, \cdots, B_b$使得 $f(x) \neq f(x^{B_i})$对所有 B_i成立，其中 x^{B_i}是将 x 中位置属于 B_i的所有二进制位翻转之后得到的位串。f 的**区组敏感度**，记为 $bs(f)$，指的是 $\max_x \{bs_x(f)\}$。

显然，$s(f) \leqslant bs(f)$。人们猜想，存在常数 c 使得 $bs(f)=O(s(f)^c)$对任意 f 成立。这一猜想的研究工作还未取得任何实质性进展(但是，要使猜想成立，必须要求 $c \geqslant 2$)。不难证明，函数 f 的敏感度和区组敏感度都是 f 的确定型判定树复杂性的下界。

引理 12.11 $s(f) \leqslant bs(f) \leqslant D(f)$对任意函数成立。

证明 设 x 满足 $bs_x(f)=bs(f)=s$(其中 s 是某个具体的数值)而 $B_1, \cdots, B_s$是二进制位置的相应分组。对于 f 的任意判定树 t，在给定的输入 x 上，t 要想将 x 从所有 x^{B_i}(其中 $i \in [m]$)中区分出来，它至少需要查验每个位置分区 B_i中的一个位置所标识的二进制位。 ■

另一方面，敏感度的平方是 f 的证明复杂性的上界。

定理 12.12 $C(f) \leqslant s(f)bs(f)$

证明 对任意 $x \in \{0,1\}^n$，我们给出 x 的一个大小为$s(f)bs(f)$的证明。输入 x 的证

明这样获得：取达到最大值 $b=bs_x(f)\leqslant bs(f)$ 的不相交位置分区 B_1，$B_2\cdots$，B_b 的并集，其中每个 B_i 的大小都达到最小。也就是说，如果 B_i 有一个真子集 B_i' 使得 $f(x^{B_i})\neq f(x^{B_i'})$，则用 B_i' 替代 B_i。这意味着，令 $x'=x^{B_i}$，则 $f(x')\neq f(x^{[j]})$ 对任意 $j\in B_i$ 均成立。这又意味着，$|B_i|\leqslant s(f)$ 对任意 $i\in[b]$ 成立。进而，x 的证明的大小至多为 $s(f)bs(f)$。

下面证明，按上述方法得到的 B_1，$B_2\cdots$，B_b 的并集确实是 x 的一个 $f(x)$-证明。若不然，假设输入 x' 满足 $f(x')\neq f(x)$ 但是 x' 与 x 在 $B_1\cup B_2\cup\cdots\cup B_b$ 中的每个位置上都取相同的二进制位。令 B_{b+1} 是将 x 转换成 x' 需要翻转的二进制位的所有位置构成的集合，则
266 显然 B_{b+1} 与 B_1，$B_2\cdots$，B_b 都不相交。这与 $b=bs_x(f)$ 矛盾。 ■

12.4.3　次数方法

在判定树下界最近的研究中，研究者们使用了布尔函数的**多项式表示法**。回顾一下，多变量线性多项式是每个变量的次数均不超过 1 的多变量多项式。

定义 12.13　对于实数域上的 n 变量多项式 $p(x_1,x_2,\cdots,x_n)$ 和函数 $f:\{0,1\}^n\to\{0,1\}$，如果 $p(x)=f(x)$ 对所有 $x\in\{0,1\}^n$ 成立，则称 $p(x_1,x_2,\cdots,x_n)$ 表示了函数 f。

f 的次数，记为 $deg(f)$，指的是表示 f 的多变量线性多项式的次数。习题 12.7 要求读者证明，表示 f 的多变量线性多项式是唯一的。因此，f 的次数是良定义的。

例 12.14　n 个变量 x_1，x_2，$\cdots$，x_n 上的 AND 函数可以表示为多变量线性多项式 $\prod_{i=1}^{n}x_i$，而这 n 个变量上的 OR 函数则可以表示为 $1-\prod_{i=1}^{n}(1-x_i)$。 ◀

AND 函数和 OR 函数的次数都是 n，因此它们的判定树复杂度也是 n。事实上，$deg(f)\leqslant D(f)$ 对任意函数 f 均成立(参见习题 12.7)。反之，另一个方向的不等式也大致成立，我们略去其证明。

定理 12.15

1. $bs(f)\leqslant 2\cdot deg(f)$；
2. $D(f)\leqslant deg(f)^2\cdot bs(f)$。

267 表 12-1 总结了本章引入的各种复杂性测度。

表 12-1　本章记号及其相互关系的总结。我们仅证明了 $D(f)\leqslant bs(f)^4$，更强的关系 $D(f)\leqslant bs(f)^3$ 请参阅[BBC+98]

$D(f)$	确定型判定树复杂性(对应于 P)
$R(f)$	随机判定树复杂性(对应于 ZPP)
$C(f)$	证明复杂性(对应于 NP∩coNP)
$s(f)$	f 的敏感性(翻转函数值 $f(x)$ 需要修改的二进制的最大个数)
$bs(f)$	f 的区组敏感性(翻转函数值 $f(x)$ 需要修改的区组的最大个数)
$deg(f)$	f 的多变量线性多项式的次数

$$C(f)\leqslant R(f)\leqslant D(f)\leqslant C(f)^2$$
$$s(f)\leqslant bs(f)\leqslant D(f)\leqslant bs(f)^3$$
$$C(f)\leqslant s(f)bs(f)$$
$$bs(f)\leqslant 2deg(f)$$
$$D(f)\leqslant deg(f)^2bs(f)\leqslant 2deg(f)^4$$

本章学习内容

- 函数 f 的判定树复杂性是计算 f 值时输入上需要被查验的二进制位的个数。存在随机判定树复杂性和非确定型判定树复杂性的概念。
- 不同于图灵机模型中的各种猜想，各种判定树复杂性(包括确定型、随机型的和非确定型的)是多项式地相互关联的。
- 利用姚期智最小最大引理，可以将证明随机算法的最坏复杂性转换为证明确定型算法的平均复杂性。
- 证明判定树复杂性下界的其他方法还包括敌手论证法、敏感性和区组敏感性，以及次数方法。

本章注记和历史

在早期的计算中，判定树已经被用于医疗诊断的表示和运筹学中。博拉克(Pollack)[Pol65]实现了一个将判定树转换为计算机程序的算法，转换过程既可以选择最小化计算机程序的规模，也可以选择最短化计算机程序的运行时间(即判定树复杂性)。加里(Garey)[Gar72]形式地定义了判定树并给出了一些判定树计算算法。亚菲(Hyafil)和李维斯特(Rivest)则证明了"为某个分类问题寻找最优判定树"是 **NP** 完全的[HR76]。

布尔曼(Burhman)和德沃夫(de Wolf)[BdW02]对判定树复杂性进行了很好的综述。"连通性及其他一些问题的判定树复杂度是 $\binom{n}{2}$"这一结论受到如下猜想的启发：m-顶点图的任意非常数的单调图性质 f 都满足 $D(f)=\binom{n}{2}$。(上述猜想究竟应该归功于安德尔拉(Anderaa)，卡普(Karp)和罗森博格(Rosenberg)中的哪一位，大家还没有统一的意见。)该猜想中，"单调性"指的是，往图中添加边不会使得原本具有的性质消失(比如，连通性是单调的)；"图性质"指的是，性质不依赖于图中顶点的编号(例如，连通性，含有 k-团等性质是图性质，而顶点 1 和顶点 2 之间存在一条边则不是图性质)。当 m 是素数幂时，李维斯特和维耶米恩(Vuillemin)[RV97]已经证明了猜想是正确的。在一般情况下，猜想在忽略常数因子的条件下成立[KSS83]。证明过程用到了拓扑学，堵丁柱和葛可一[DK00]对此进行了精彩的讲解。另一个猜想是，尽管单调图性质的随机判定树复杂度是 $\Omega(n^2)$，但它们的最佳判定树下界却可能接近于 $n^{4/3}$[Yao87，Kin88，Haj90]。关于这些猜想及其研究现状，参见综述[LY02]。

敏感性的概念是库克(Cook)、迪沃克(Dwork)和雷斯洽克(Reischuk)[CDR86]等人定义的；而区组敏感性则是由尼散(Nisan)定义的，他同时还证明了定理 12.12[Nis89]。定义这两个概念的动机都是为了证明并行随机访存机器的计算能力的下界。

定理 12.15 的第 1 个部分源自尼散(Nisan)和塞盖迪(Szegedy)[NS92]，第 2 部分源自尼散(Nisan)和斯莫伦斯基(Smolensky)(未发表)，其证明过程参见[BdW02]。 268

证明判定树复杂性下界的多项式方法参见综述[BdW02]。该方法也可以用来证明随机判定树复杂性的下界，最近还被用来证明量子判定树复杂性的下界。但是，此时需要考虑函数的多项式近似表示。

习题

12.1　假设 f 是依赖于所有输入二进制位的一个函数。换句话说，对每个位置 i，存在一个满足 $f(x)\neq f(x')$ 的输入 x，其中 x' 是将 x 的第 i 位翻转之后得到的位串。证明：$s(f)=\Omega(\log n)$。

12.2　对于任意 $k\in\mathbf{N}$，令 f_k 是长度为 2^k 的输入上的 AND-OR 函数(参见例 2.2)。证明：$D(f_k)=2^k$。

12.3　令 f 是如下定义在 $n=k^2$ 个变量上函数，它将 AND 操作作用到 k 个 OR 子句上，每个 OR 子句将 OR 操作作用到 k 个变量上，各个 OR 子句使用变量均各不相同。证明：$s(f)=bs(f)=C(f)=\sqrt{n}$，并且 $deg(f)=D(f)\geqslant\Omega(n)$。

12.4　令 f 是如下定义在 $n=k^2$ 个变量上函数，它将 OR 操作作用到函数 $g:\{0,1\}^k\to\{0,1\}$ 的 k 次调用上，每次调用使用 k 个变量，各次调用使用的变量互不相同。而且，$g(x_1,\cdots,x_k)=1$，如果存在 $i\in[k-1]$ 使得 $x_i=x_{i+1}=1$ 且 $x_j=0$(其中 $j\neq i,i+1$)。证明：$s(f)=\sqrt{n}$，且 $bs(f)=n/2$。

12.5　证明：对于任意 $f:\{0,1\}^n\to\{0,1\}$，存在唯一的多变量线性多项式来表示它。

12.6　将 n 个二进制位上的奇偶性函数表示为多变量线性多项式。

269　12.7　证明：$deg(f)\leqslant D(f)$。

第 13 章

Computational Complexity, A Modern Approach

通信复杂性

本文研究了两个处理器协同计算布尔值函数时需要交换的信息。

——姚期智，1979

通信复杂性考虑如下情形。两个计算能力不受限制的参与方协同地计算双方都知道的函数 $f:\{0,1\}^n\times\{0,1\}^n\to\{0,1\}$，每个参与方各自持有一个长度为 n 的输入串 x，y 且双方都不知道另一方的输入。参与双方希望协同地计算得到 $f(x,y)$。当然，参与计算的双方在获知 x，y 之前早就预见到，双方可能相距非常遥远(比如，一方在宇宙飞船上而另一方则在地球上的基站内)。因此，他们达成了双方都认可的通信协议[⊖]。通信协议的代价指的是，在 x，y 的最坏选择上，双方计算 $f(x,y)$时需要通讯的二进制的个数。

研究者还研究了上述基本模型的变形，包括随机通信协议，非确定型通信协议和平均通信协议。并且，通信复杂性的下界还被广泛应用到各个领域，包括并行计算的下界、超大规模集成电路(VLSI)计算的下界、线路下界、多项式分层理论、数据结构下界，以及其他领域。在复杂性理论研究所获得的众多模型中，通信复杂性模型是最成功的模型之一，这是由于该模型在简单性和一般性上达到了某种难以置信的平衡。它非常简单，这使得在该模型上能够证明得到一些较强的下界；同时，它又具有很强的一般性，这使得所证得的下界具有一些重要的应用。

本章初步地介绍通信复杂性这一研究领域。13.1 节给出确定型双方通信复杂性的基本概念。13.2 节概要地介绍"证明各种函数的通信复杂性下界"所采用的技术，介绍过程中将始终以相等函数(亦即，$f(x,y)=1$ 当且仅当 $x=y$)为例。13.3 节将定义多方通信复杂性，并证明广义内积函数的多方通信复杂性下界。13.4 节简要概述了其他通信复杂性模型，包括概率型通信复杂性模型和非确定型通信复杂性模型。通信复杂性有许多应用，本章注记会提到其中的一些应用。

13.1 双方通信复杂性的定义

现在，我们形式地定义上面提及的通信复杂度。

定义 13.1 (双方通信复杂性) 设 $f:\{0,1\}^{2n}\to\{0,1\}$ 是一个函数。计算 f 的 t-回合双方通信协议 Π 指的是由 t 个函数构成的序列 P_1，…，P_t，$\{0,1\}^*\to\{0,1\}$。协议 Π 在输入 x，y 上的一次执行指的是如下过程：参与方 1 计算 $p_1=P_1(x)$并将 p_1 发送给参与方 2，然后参与方 2 计算 $p_2=P_2(y,p_1)$并将 p_2 发送给参与方 1，以此类推。一般地，在第 i 个回合中，如果 i 是奇数，则参与方 1 计算 $p_i=P_i(x,p_1,\cdots,p_{i-1})$并将 p_i 发送给参与方 2。类似地，如果 i 是偶数，参与方 2 计算 $p_i=P_i(y,p_1,\cdots,p_{i-1})$并将 p_i 发送给参

⊖ 不要将本章的讨论同信息论混淆。信息论中的算法需要在噪声信道上传递给定的信息。其目的是在实现稳健的信息传递的条件下将通信量最小化。在通信复杂性中，信道是无噪声的，参与通信的各方需要确定传递哪些信息。

与方 1。

如果在任意输入 x，y 上通信协议 Π 发送的最后一个消息(即消息 p_t)是 $f(x, y)$，则称通信协议 Π 是有效的。Π 的**通信复杂度**指的是在任意 x，$y\in\{0,1\}^n$ 上执行协议时需要通讯的二进制位的最大个数(亦即，$|p_1|+\cdots+|p_t|$ 的最大值)。函数 f 的**通信复杂度**，记为 $C(f)$，指的是在所有计算 f 的有效通信协议 Π 中的最小通信复杂度。

$C(f)\leqslant n+1$ 对任意函数成立。因为在平凡的通信协议中，第一参与方可以将他的整个输入传递给第二参与方，然后第二参与方计算 $f(x, y)$并将计算结果传递给第一参与方。通信双方能够以更少的通信量解决问题吗?

例 13.2（奇偶性） 假设函数 $f(x, y)$要求统计 x，y 的所有位中(等于 1 的二进制位的个数)的奇偶性，则 $C(f)=2$。显然，$C(f)\geqslant 2$；这是由于函数值非平凡地依赖于 x 和 y，因此参与双方各自至少需要传送 1 个位。另一方面，$C(f)\leqslant 2$；因为如下通信协议可以达到这一通信复杂度：参与方 1 将 x 的奇偶性 a 发送给参与方 2，参与方 2 将 y 的奇偶性 b 与 a 进行异或(XOR)操作并将操作结果发送给参与方 1。◀

例 13.3（停机问题） 考虑如下定义的函数 H：$\{0,1\}^*\times\{0,1\}^*\to\{0,1\}$。如果 $x=1^n$而 $y=\mathrm{code}(M)$，其中 M 是在 x 上能停机的某个图灵机，则 $H(x, y)=1$；否则，$H(x, y)=0$。该函数的通信复杂度至多为 2。事实上，第一参与方首先给第二参与方发送一个二进制位，以表明第一参与方的输入是否为 1^n；然后，第二参与方确定答案并将答案传回给第一参与方。本例强调的是，参与通信的双方都具有无限的计算能力，甚至可以具有求解停机问题的能力。◀

有的同学会问：通信的参与方可以不发送任何信息吗?(毕竟，通信双方在通信的每
271 个回合中都有三种选择，即发送 0，发送 1 或者不发送任何信息。)我们认为，这样的通信协议也交换了一个二进制位，并同其他通信协议一起类似地进行分析。

13.2　下界方法

下面，我们讨论证明通信复杂性下界的各种方法。我们将使用如下定义的相等函数(equality function)贯穿本章的讨论。

$$\mathrm{EQ}(x,y)=\begin{cases}1 & \text{如果 } x=y\\ 0 & \text{否则}\end{cases}$$

可以证明，该函数的通信复杂度几乎不可能在 $n+1$ 的基础上再有任何改进了，其中 $n+1$是计算该函数的平凡通信协议的复杂度。

定理 13.4 相等函数具有线性的通信复杂度 $C(\mathrm{EQ})\geqslant n$。

下面，我们用几种不同的方法来证明定理 13.4。

13.2.1　诈集方法

证明定理 13.4 的第一种思想称为诈集(fooling sets)[⊖]。对于任意函数的任意通信协议，假设 x，x'是两个长度为 n 的不同位串。如果通信协议在输入(x, x)和(x', x')上具有相同的通信模式(亦即通信过程传递相同的位序列)，则我们断言通信双方在四对输入

⊖ "诈"的含义同"东虞我诈"的"诈"。"诈集"的含义是，为了证明通信复杂度的某个下界，只需"骗出"一个集合并证明这个集合确实可以表明给定的下界。——译者注

(x, x)，(x, x')，(x', x)和(x', x')上将得到相同的最终答案。为了证明该论断，可以使用数学归纳法。如果参与方 1 在通信的第一个回合中发送一个二进制位，根据归纳假设，无论他的输入是 x 还是 x'，他将发送同一个二进制位。如果参与方 2 在第二个回合中发送一个二进制位，则无论他的输入是 x 还是 x'他也将发送同一个二进制位，因为他收到的源自参与方 1 的二进制位是相同的。以此类推，最终将得到结论：通信双发在(x, x)上得到的答案必然同他们在(x, x')上得到的答案一样。

为了证明 $C(\mathrm{EQ})\geqslant n$，只需注意到：如果存在复杂度至多为 $n-1$ 的通信协议，则所有可能的通信模式至多只有 2^{n-1} 种。但是，形如(x, x)的输入序对却有 2^n 个；由鸽巢原理可知，至少存在两个不同的输入序对(x, x)和(x', x')具有相同的通信模式。然而，$\mathrm{EQ}(x, x')=0\neq\mathrm{EQ}(x, x')$却又表明，通信协议是无效的。这就完成了证明。这种论证方法可以推广为如下的形式(参见习题 13.1)。

引理 13.5　我们称函数 $f: \{0, 1\}^n\times\{0, 1\}^n\to\{0, 1\}$有一个大小为 M 的许集，如果存在一个大小为 M 的子集 $S\subseteq\{0, 1\}^n\times\{0, 1\}^n$和值$b\in\{0, 1\}$使得：(1)$f(x, y)=b$ 对任意$(x, y)\in S$成立；(2)对 S 中任意不同的序对(x, y)和(x', y')，要么 $f(x, y')\neq b$ 要么 $f(x', y)\neq b$。

如果 f 存在大小为 M 的许集，则 $C(f)\geqslant\log M$。

272

例 13.6　(不相交性)　将输入 x 和 y 分别视为集合$\{1, 2, \cdots, n\}$的两个子集的特征向量。如果相应的两个子集不相交，则令 $\mathrm{DISJ}(x, y)=1$；否则，令 $\mathrm{DISJ}(x, y)=0$。作为引理 13.5 的推论，我们得到 $C(\mathrm{DISJ})\geqslant n$；这是因为如下的 2^n 个序对构成一个许集

$$S=\{(A, \overline{A}): A\subseteq\{1, 2, \cdots, n\}\}$$

◀

13.2.2　铺砌方法

证明通信复杂性下界的第二种方法称为铺砌方法，它从更加全局的视角处理函数 f。f 的矩阵 $M(f)$是一个 $2^n\times 2^n$的矩阵，其中(x, y)位置存储元素 $f(x, y)$，参见图 13-1。我们用矩阵 $M(f)$将函数 f 的通信协议可视化。

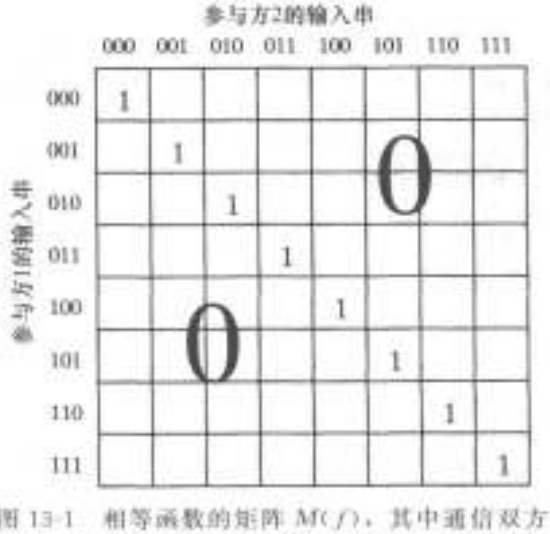


图 13-1　相等函数的矩阵 $M(f)$，其中通信双方的输入都只含 3 个二进制位，矩阵中的数即为 f 的值

矩阵 M 的一个组合矩形(有时也简称矩形)是抽取 M 中位置属于 $A\times B$ 的所有元素得到的一个子矩阵，其中 $A\subseteq\{0, 1\}^n$，$B\subseteq\{0, 1\}^n$。如果在所有 $x\in A$，$y\in B$ 上 $M_{x,y}$都具有相同的值，则称 $A\times B$ 是单色的。如果通信协议以第一参与方发送一个二进制位开始，则矩阵 $M(f)$可以划分为两个类型为 $A_0\times\{0, 1\}^n$，$A_1\times\{0, 1\}^n$的矩形，其中 A_0包含所有使得第一参与方发送二进制位 b 的输入。注意，$A_0\cup A_1=\{0, 1\}^n$。如果接下来第二参与方发送一个二进制位，则根据他发送的是 1 还是 0，上述的两个矩形将被进一步划分成更小的矩形。最后，如果通信过程总共交换了 k 个二进制位，则函数的矩阵将被划分为 2^k 个小矩形。注意，在划分得到的每个小矩形中，所有输入序对都具有相

同的通信模式(参见图 13-2 给出的例子)。当通信协议执行结束时，通讯双方在通讯过程中交换的所有二进制位就确定 f 的值；因此，同一个小矩形中的所有 x，y 都具有相同的 f 值。这就是说，所有通信模式构成的集合必然将函数的矩阵划分成一些单色矩形。

273

定义 13.7 $M(f)$的一个单色铺砌指的是将 $M(f)$ 划分为一些不相交的单色矩形。在 $M(f)$的所有单色铺砌中，将单色矩形的最小个数记为 $\chi(f)$。

铺砌与通信复杂性的联系如下所述。

定理 13.8 (铺砌与通信复杂度[AUY 83]) $\log_2\chi(f)\leqslant C(f)\leqslant 16(\log_2\chi(f))^2$。

证明 第一个不等式由前面的讨论可得：亦即，如果 f 的通信复杂度为 k，则 $M(f)$存在一个至多由 2^k 个矩形构成的单色铺砌。第二个不等式留作习题 13.5。 ■

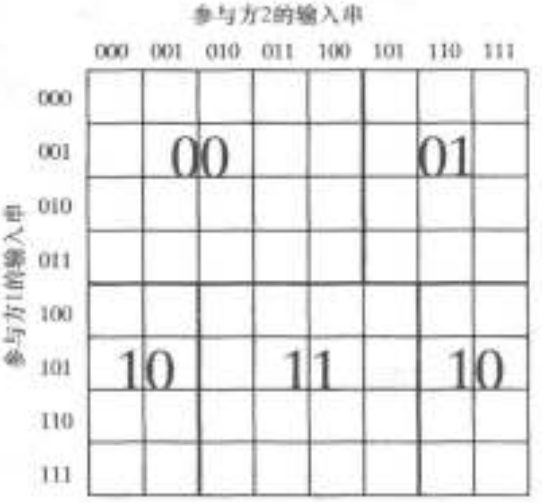


图 13-2　通信协议执行两步之后的双向通信矩阵。矩阵中，用于标记的大号数字的第 1 个位是第一参与方发送的二进制位，第 2 个位是第二参与方发送的二进制位

下面的观察结果表明，对于任意函数 f，如果诈集方法能够给出通信复杂度的下界，则铺砌方法也能给出同样的下界。因此，铺砌方法涵盖了诈集方法。

引理 13.9 如果 f 存在一个由 m 个序对构成的诈集，则 $\chi(f)\geqslant m$。

证明 如果(x_1, y_1)和(x_2, y_2)是诈集中的两个序对，则它们不可能位于同一个单色矩形中，因为在(x_1, y_1)，(x_2, y_2)，(x_1, y_2)，(x_2, y_1)上函数 f 的值并不完全相同。 ■

13.2.3　秩方法

下面，我们引入一种代数方法来获得 $\chi(f)$的下界，进而也能得到 f 的通信复杂度下界。回顾一下，方阵的秩指的是矩阵中线性无关的行的最大行数。下面的引理给出了秩的

274

等价描述，我们将其证明留作习题 13.6。

引理 13.10 域 F 上的一个 $n\times n$ 矩阵 M 的秩，记为 rank(M)，是使得 M 可以表示为如下形式的最小 ℓ 值

$$M=\sum_{i=1}^{\ell}B_i$$

其中每个 B_i 都是秩为 1 的 $n\times n$ 矩阵。

注意，0，1 是任意域中的元素。因此，我们可以对任意域上的矩阵计算秩。但是，域的选择会影响秩的大小，参见习题 13.9。

对于任意的单色矩形，我们可以将矩形之外的其他元素用 0 填充，这样单色矩形就可以视为秩至多为 1 的矩阵。于是，我们有如下的定理。

定理 13.11 对任意函数 f，$\chi(f)\geqslant\text{rank}(M(f))$。

例 13.12 相等函数的矩阵就是单位矩阵，因此 rank($M(EQ)$)$=2^n$。于是，$C(EQ)\geqslant\log_2\chi(EQ)\geqslant n$，这就得到定理 13.14 的另一个证明。 ◀

13.2.4　差异方法

在这种方法中，将函数 f 视为取值为 ± 1 的函数更为方面，而这只需通过映射 $b \mapsto (-1)^b$（亦即，$0 \mapsto +1$，$1 \mapsto -1$）就可以实现。这样，$M(f)$ 就是一个 ± 1 矩阵。在 $2^n \times 2^n$ 矩阵 M 中，我们将矩形 $A \times B$ 的差异(discrepancy)定义为

$$\frac{1}{2^{2n}}\left|\sum_{x \in A, y \in B} M_{x,y}\right|$$

矩阵 $M(f)$ 的差异，记为 $\mathrm{Disc}(f)$，指的是所有矩形中差异的最大值。下面的简单引理建立了差异与 $\chi(f)$ 的联系。

引理 13.13　$\chi(f) \geqslant \dfrac{1}{\mathrm{Disc}(f)}$。

证明　如果 $\chi(f) \leqslant K$，则存在一个至少包含 $2^{2n}/K$ 个元素的单色矩形。这个矩形的差异至少为 $1/K$。 ■

引理 13.13 可能很宽松。在相等函数的矩形上（即整个矩阵上），其差异至少为 $1-2^{-n}$。这只能说明 $\chi(f)$ 以 2 为下界。但是，正如前面所见，$\chi(f)$ 实际上至少为 2^n。

下面，我们用矩阵的特征值来刻画一种方法，它能给出差异的上界。 275

引理 13.14（特征值界）　对于任意对称实数矩阵 M，矩形 $A \times B$ 的差异至多为 $\lambda_{\max}(M)\sqrt{|A||B|}/2^{2n}$，其中 $\lambda_{\max}(M)$ 是矩阵 M 的最大特征值的大小。

证明　证明过程将基于如下事实：$x^{\mathrm{T}} M y \leqslant \lambda_{\max}(M)|(x, y)|$ 对任意的单位向量 x，y 成立。令 $\mathbf{1}_S \in \mathbf{R}^{2^n}$ 是子集 $S \subseteq \{0,1\}^n$ 的特征向量；也就是说，如果 $x \in S$，则 $\mathbf{1}_S$ 的第 x 个分量等于 1，否则该分量取 0。注意，$\|\mathbf{1}_S\|_2 = \sqrt{\sum_{x \in S} 1^2} = \sqrt{|S|}$。再注意到，对任意 A，$B \subseteq \{0,1\}^n$，有 $\sum_{x \in A, y \in B} M_{x,y} = \mathbf{1}_A^{\mathrm{T}} M \mathbf{1}_B$。

矩形 $A \times B$ 的差异为

$$\frac{1}{2^{2n}}\mathbf{1}_A^{\mathrm{T}} M \mathbf{1}_B \leqslant \frac{1}{2^{2n}}\lambda_{\max}(M)\,|\mathbf{1}_A^{\mathrm{T}} M \mathbf{1}_B| \leqslant \frac{1}{2^{2n}}\lambda_{\max}(M)\sqrt{|A||B|}$$

最后一个不等式使用了柯西-西瓦兹(Cauchy-Schwarz)不等式。 ■

例 13.15　模 2 的内积函数定义为 $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \odot \boldsymbol{y} = \sum x_i y_i \pmod 2$，该函数已经在本书中出现了多次。为了给出其差异的上界，令 N 是 f 的 ± 1 矩阵，亦即 $N_{x,y} = (-1)^{x \odot y}$。不难验证，$N$ 的任意两行是正交的，每行的 ℓ_2-范数等于 $2^{n/2}$，且 $N^{\mathrm{T}} = N$。因此，我们得到 $N^2 = 2^n I$，其中 I 是单位矩阵。于是，矩阵的每个特征值要么为 $+2^{n/2}$ 要么为 $-2^{n/2}$。进而，引理 13.14 表明，任意矩形 $A \times B$ 的差异至多为 $2^{-3n/2}\sqrt{|A||B|}$。又由于 $|A||B| \leqslant 2^n$，因此 f 的差异至多为 $2^{-n/2}$。 ◀

13.2.5　证明差异上界的一种技术

下面，我们给出证明差异上界的一种技术，这种技术在后面 13.3 节中讨论多方通信复杂性时也很有用。同 13.2.4 节一样，假设 f 是取值为 ± 1 的函数。我们定义下面的数量。

定义 13.16　$\mathcal{E}(f) = \mathrm{E}_{a_1, a_2, b_1, b_2}\left[\prod_{i=1,2}\prod_{j=1,2} f(a_i, b_j)\right]$。

注意，类似于矩阵的秩，$\mathcal{E}(f)$也可以在矩阵$M(f)$的大小的多项式时间内计算得到。相比之下，由于差异的定义涉及在所有子集A，B上取最大值，因此计算差异需要花费$M(f)$的大小的指数时间。事实上，精确地计算差异是一个**NP**难问题，但是它可以被高效地近似计算(参见本章注记)。下面的引理建立了$\mathcal{E}(f)$和$\mathrm{Disc}(f)$之间的联系。

276

引理 13.17 $\mathrm{Disc}(f)\leqslant\mathcal{E}(f)^{1/4}$。

证明　我们需要证明，对任意矩形$A\times B$，有

$$\mathcal{E}(f)\geqslant\left(\frac{1}{2^{2n}}\left|\sum_{a\in A,b\in B}f(a,b)\right|\right)^4$$

令g，h分别是A，B的特征函数(亦即，若$a\in A$则$g(a)=1$，否则$g(a)=0$；若$b\in B$则$h(b)=1$，否则$h(b)=0$)。这样，上式右端可以简单地改写为$(E_{a,b\in\{0,1\}^n}[f(a,b)g(a)h(b)])^4$。但是，

$$\begin{aligned}
\mathcal{E}(f)&=\mathop{\mathrm{E}}_{a_1,a_2}\left[\mathop{\mathrm{E}}_{b_1,b_2}\left[\prod_{i=1,2}\prod_{j=1,2}f(a_i,b_j)\right]\right]\\
&=\mathop{\mathrm{E}}_{a_1,a_2}\left[\left(\mathop{\mathrm{E}}_{b}[f(a_1,b)f(a_2,b)]\right)^2\right]\\
&\geqslant\mathop{\mathrm{E}}_{a_1,a_2}\left[g(a_1)^2g(a_2)^2\left(\mathop{\mathrm{E}}_{b}[f(a_1,b)f(a_2,b)]\right)^2\right]\qquad(\text{由 } g(a)\leqslant 1)\\
&=\mathop{\mathrm{E}}_{a_1,a_2}\left[\left(\mathop{\mathrm{E}}_{b}[f(a_1,b)g(a_1)f(a_2,b)g(a_2)]\right)^2\right]\\
&\geqslant\left(\mathop{\mathrm{E}}_{a_1,a_2}\left[\left(\mathop{\mathrm{E}}_{b}[f(a_1,b)g(a_1)f(a_2,b)g(a_2)]\right)^2\right]\right)^2\qquad(\text{由 } \mathrm{E}[X^2]\geqslant\mathrm{E}[X]^2)\\
&=\left(\mathop{\mathrm{E}}_{b}\left[\left(\mathop{\mathrm{E}}_{a}[f(a,b)g(a)]\right)^2\right]\right)^2\\
&\geqslant\left(\mathop{\mathrm{E}}_{a,b}[f(a,b)g(a)h(b)]\right)^4\qquad(\text{重复上述推导步骤})\\
&\geqslant(\mathrm{Disc}(f))^4
\end{aligned}$$

■

习题13.12要求利用上述技术给出内积函数的通信复杂度下界。我们将在13.3中看到上述技术应用的另一个例子。

13.2.6　各种下界方法的比较

辅助论证法是证明下界的所有方法中最强的方法，因为秩的下界，差异的下界和诈集都蕴含着$\chi(f)$的下界；因此，同辅助方法得到的下界相比，这些方法都不会得出更优的下界。同时，正如定理13.8所述，$\log_2\chi(f)$在多项式因子误差范围内刻画了函数f的通信复杂度。秩方法和诈集方法没有可比性，也就是说两者中的每种方法都可能在某些函数上强于另一种方法。但是，如果忽略常数因子，则秩方法将几乎至少与诈集方法一样强(参见习题13.8)。同时，各种下界论证方法的能力也是有区别的。比如，人们已经找到了一些函数使得$\log_2\chi(f)$和$\log_2\mathrm{rank}(M(f))$之间存在多项式鸿沟(polynomial gap)。但是，下面的猜想(我们仅给出了它的一种形式)断言：在忽略多项式因子的前提下，秩实际上是最优的。

277

猜想 13.18　(对数秩猜想(log rank conjecture))　存在常数$c>1$使得$C(f)=O(\log(\mathrm{rank}(M(f)))^c)$对所有函数$f$和所有输入规模$n$成立。其中秩是实数域上求得的。

当然，证明上述猜想时，困难在于证明"较小的秩"蕴含了"f具有较低复杂度的通信

协议”。尽管，证明上述猜想目前仍有很长的路要走，但尼散(Nisan)和维格德尔森(Wigderson)已经证明：“较小的秩”至少能表明“1/Disc(f)也比较小”。

定理 13.19 ([NW94])$1/\mathrm{Disc}(f)=O(\mathrm{rank}(f)^{1/2})$。

13.3　多方通信复杂性

将通信复杂性推广到具有多个参与方的方法不止一种，其中最有意义的一种称为“脑门记数”模型。在这种模型中，每个参与方脑门上都有一个二进制位串，各个参与方能看到其他参与方脑门上的位串但无法看见自己脑门上的位串。也就是说，有 k 个参与方和 k 个二进制位串 x_1，…，x_k，第 i 个参与方获得了除 x_i 之外的其他所有位串。所有参与方感兴趣的是如何计算函数 $f:((0,1)^n)^k\to(0,1)$ 在 x_1，…，x_k 上的函数值 $f(x_1,\cdots,x_k)$。同双方通信模型一样，k 个参与方在获得输入数据之前就已经达成了一致的通信协议；并且，所有参与方之间的通信都通过一个“公用黑板”进行，因此各方都能了解到每次通信的信息；此外，通信协议还规定了各个参与方在黑板上书写信息的顺序；最后通讯的信息恰好是函数在各方输入数据上的函数值 $f(x_1,\cdots,x_k)$。类似于双方通信模型，我们用 $C_*(f)$ 表示计算函数的最佳通信协议需要交换的二进制位的个数。注意，$C_*(f)$ 至多为 $n+1$；因为只要满足 $j\neq i$ 的任意一个参与方 j 将 x_i 写在黑板上，则参与方 i 就知道了所有的输入串，进而他可以计算得到 $f(x_1,\cdots,x_k)$ 并将计算结果发布在黑板上。

例 13.20　考虑在三方模型中计算定义在长度为 n 的输入 x_1，x_2，x_3 上的如下函数：

$$f(x_1,x_2,x_3)=\bigoplus_{i=1}^{n}\mathrm{maj}(x_{1i},x_{2i},x_{3i})$$

该函数的通信复杂度是 3。事实上，每个参与方根据看到的其他两方的输入，能够确定多数函数在某些位置上 x_{1i}，x_{2i}，x_{3i} 的结果，他将这种位置的个数的奇偶性写在黑板上；三方结果的奇偶性就是函数的最终取值。上述通信协议是正确的，因为多数函数在每个行 x_{1i}，x_{2i}，x_{3i} 上的取值要么能被 1 个参与方获悉，要么能被 3 个参与方获悉，但无论哪种结果它都是一个奇数。◀

例 13.21　(广义内积)　广义内积函数 $\mathrm{GIP}_{k,n}$ 如下地将 nk 个二进制位映射为一个位

$$f(x_1,\cdots,x_k)=\bigoplus_{i=1}^{n}\bigwedge_{j=1}^{k}x_{ji}\tag{13.1}$$

注意，但 $k=2$ 时，广义内积函数就变成了例 13.15 中的模 2 的内积函数。◀ 278

在双方通信模型中，我们引入了单色矩形来证明复杂性的下界。具体地说，通信协议被看做对矩阵 $M(f)$ 的一种划分方法：如果通信协议执行过程中交换了 c 个二进制位，则矩阵被划分为 2^c 个矩形。如果通信协议是有效的，则这 2^c 个矩形都是单色的。

在 k 方通信协议中，相应的概念称为柱体交集。第 i 维柱体指的是满足如下条件的输入子集 S：如果 $(x_1,\cdots,x_k)\in S$，则 $(x_1,\cdots,x_{i-1},x_i',x_{i+1},\cdots,x_k)\in S$ 对任意 x_i' 成立。柱体交集指的是 $\bigcap_{i=1}^{k}T_i$，其中 T_i 是第 i 维柱体。由于参与方 i 在通信中发送的二进制位不依赖于 x_i；因此根据各维柱体形成的柱体交集可以认为是对输入集合的一个划分。由此可知，k 方通信协议执行完成后，超方体 $(0,1)^n$ 被划分为一些柱体交集；而且，如果通信过程交换了 c 个二进制位，则划分中将包含至多 2^c 个单色的柱体交集。由此可以证明下面的引理。

引理 13.22　如果将 $M(f)$ 划分成单色柱体交集时每种划分方案都至少使用 R 个柱体交集，则 f 的 k 方通信复杂度至少为 $\lceil \log_2 R \rceil$，其中 $M(f)$ 是 k 维的表，它的 $(x_1, \cdots, x_k)$ 位置存储的元素是 $f(x_1, \cdots, x_k)$。

基于差异的下界

下面，同前面讨论差异时一样，我们假设 f 的值域为 $\{-1, 1\}$。类似于双方通信协议的情况，我们定义 f 的 k 方差异为

$$\mathrm{Disc}(f) = \frac{1}{2^n} \max_{T} \left| \sum_{(a_1, \cdots, a_k) \in T} f(a_1, \cdots, a_k) \right|$$

其中 T 取遍所有的柱体交集。

为了给出差异的上界，我们在 k 方通信协议中类似地引入 $\mathcal{E}(f)$ 的概念。令 (k, n)-超方体表示 $\{0, 1\}^n$ 中形如 $\{a_1, a_1'\} \times \{a_2, a_2'\} \times \cdots \times \{a_k, a_k'\}$ 的 2^k 个点构成的（多重）子集 D，其中每个 $a_i, a_i' \in \{0, 1\}^n$。我们定义

$$\mathcal{E}(f) = \mathop{\mathrm{E}}_{\substack{D \\ (k,n)\text{-超方体}}} \left[\prod_{\boldsymbol{a} \in D} f(\boldsymbol{a}) \right]$$

注意，在上述定义中令 $k=2$ 就得到双方通信协议中的 $\mathcal{E}(f)$。下面的引理同样可以通过简单的推广来得到。

引理 13.23　$\mathrm{Disc}(f) \leqslant (\mathcal{E}(f))^{1/2^k}$。

证明过程类似于引理 13.17 的证明，我们将它留作习题 13.14。唯一的区别是，我们要让基本推导步骤重复 k 遍而不再是重复两遍。

现在，我们可以证明广义内积函数（GIP）的多方通信复杂度下界了。注意，在值域为 $\{-1, 1\}$ 的假设条件下，广义内积函数可以重新定义为

$$\mathrm{GIP}_{k,n}(x_1, x_2, \cdots, x_k) = (-1)^{\sum_{i=1}^{n} \prod_{j=1}^{k} x_{ji}} \tag{13.2}$$

279　（该定义省去了对 2 取模的操作，因为 $(-1)^m = (-1)^{m \bmod 2}$ 对任意 m 成立）。

定理 13.24　（广义内积函数的多方通信复杂度下界）　函数 $\mathrm{GIP}_{k,n}$ 的 k 方通信复杂度下界为 $\Omega(n/4^k)$。

证明　根据引理 13.23，只需给出 $\mathcal{E}(\mathrm{GIP}_{k,n})$ 的上界。从 (13.2) 式中可以看到，对任意的 k、n 均有

$$\mathrm{GIP}_{k,n}(x_1, x_2, \cdots, x_k) = \prod_{i=1}^{n} \mathrm{GIP}_{k,1}(x_{1,i}, \cdots, x_{k,i})$$

其中 $\mathrm{GIP}_{k,1}(x_1, \cdots, x_k) = (-1)^{\prod_{j=1}^{k} x_j}$。因此，

$$\mathcal{E}(\mathrm{GIP}_{k,n}) = \mathop{\mathrm{E}}_{\substack{D \\ (k,n)\text{-超方体}}} \prod_{\boldsymbol{a} \in D} \prod_{i=1}^{n} [\mathrm{GIP}_{k,1}(\boldsymbol{a}_i)]$$

这里，对于向量 $\boldsymbol{a} = (a_1, \cdots, a_k) \in (\{0, 1\}^n)^k$ 而言，$\boldsymbol{a}_i$ 表示所有 a_j 中的第 i 个二进制位拼接得到的位串 $a_{1,i}, a_{2,i}, \cdots, a_{k,i}$。但是，由于向量中的每个坐标都是独立选取的，上式右端等于

$$\prod_{i=1}^{n} \mathop{\mathrm{E}}_{\substack{C \\ (k,1)\text{-超方体}}} \left[\prod_{\boldsymbol{a} \in C} \mathrm{GIP}_{k,1}(\boldsymbol{a}) \right] = \mathcal{E}(\mathrm{GIP}_{k,1})^n$$

但是，$\mathcal{E}(\mathrm{GIP}_{k,1}) \leqslant 1 - 2^{-k}$。事实上，在随机的 $(k, 1)$-超方体 $C = \{a_1, a_1'\} \times \{a_2, a_2'\} \times \cdots \times \{a_k, a_k'\}$ 上，随机事件 E^n 对所有的 i，(a_i, a_i') 不是 $(1, 0)$ 就是 $(0, 1)^n$ 发生的概率等于 2^{-k}。

然而，由于 $\mathrm{GIP}_{k,1}(a)=-1$ 当且仅当 a 是全等于 1 的位向量；所以，如果 E 确实发生，则 C 中恰有一个 k-位串向量 a 满足 $\mathrm{GIP}_{k,1}(a)=-1$ 而对于其他向量 $b\in C$ 则有 $\mathrm{GIP}_{k,1}(b)=1$。因此，如果 E 发生，则 $\prod_{a\in C}\mathrm{GIP}_{k,1}(a)=-1$。由此可知，$\prod_{a\in C}\mathrm{GIP}_{k,1}(a)=1$ 几乎总成立，

$$\mathcal{E}(\mathrm{GIP}_{k,1})=\underset{\substack{C\\ |C|=\text{规定值}}}{E}\left[\prod_{a\in C}\mathrm{GIP}_{k,1}(a)\right]\leqslant 2^{-k}\cdot(-1)+(1-2^{-k})\cdot 1\leqslant 1-2^{-k}$$

因此，$\mathcal{E}(\mathrm{GIP}_{k,n})\leqslant(1-2^{-k})^n\leqslant e^{-n/2^k}=2^{-\Omega(n/2^k)}$。进而，$\mathrm{Disc}(\mathrm{GIP}_{k,n})\leqslant 2^{-\Omega(n/2^k)}$。这意味着，函数 $\mathrm{GIP}_{k,n}$ 的 k 方通信复杂度是 $\Omega(n/4^k)$。 ■

目前，人们未发现满足 $C_k(f)\geqslant n2^{-o(k)}$ 的任何明确的函数 f。特别地，对于 $k\geqslant\log n$，任意明确的函数 $f:(\{0,1\}^n)^k\to\{0,1\}$ 都不存在非平凡的多方通信复杂性下界。这种结论可以用来证明线路的新下界，参见 14.5.1 节。

13.4　其他通信复杂性模型概述

人们还研究了许多其他的通信复杂性模型，下面概述其中几种模型。

随机通信协议：首先考虑如何在随机通信模型上计算函数的值。在这种模型下，参与通信的各方可以访问一个共享的随机串 r，各参与方都用这个随机串来确定通信行为。我们定义 $R(f)$ 是所有参与方交换的二进制位的个数的数学期望。人们已经证明，随机性在某些场合中可能对通信复杂性造成重大影响。例如，相等函数有一个复杂度为 $O(\log n)$ 的随机通信协议（参见习题 13.15）。当然，证明随机通信协议复杂性下界也存在一些专门的技术。 280

非确定型通信协议：采用定义 **NP** 时所用的类似方法，我们也可以定义非确定型通信协议。在这种模型中，通信双方都有另一个长度为 m 的输入 z（亦即，“非确定型猜测”），其中 z 可能依赖于 x，y。除了这个猜测之外，通信协议的其余部分都是确定型的。通信协议要求，$f(x,y)=1$ 当且仅当存在 z 使得通信双方都输出 1。通信协议的复杂度定义为 m 与通信过程中交换的二进制位的个数之和。同本节介绍的其他模型一样，非确定性也可能对通信复杂性造成重大影响。例如，如果将不等函数和相交函数分别定义为例 13.6 中的相等函数 EQ 和不相交函数 DISJ 的否定，则可以证明这两个函数都具有对数的非确定型通信复杂度。同理，类似于 **coNP** 的定义，我们还可以将 f 的“补非确定型通信复杂度”定义为函数 $g(x,y)=1-f(x,y)$ 的非确定型通信复杂度。有趣的是，可以证明：如果 f 的非确定型通信复杂度是 k 并且 f 的补非确定型通信复杂度是 ℓ，则 $C(f)\leqslant 10k\ell$。这意味着，在通信复杂性领域中，**NP** 对应的复杂性类和 **coNP** 对应的复杂性类的交集就等于 **P** 对应的复杂性类。相比之下，在图灵机模型中，人们却相信 $\mathbf{P}\neq\mathbf{NP}\cap\mathbf{coNP}$。

平均通信协议：就像我们可以在图灵机模型中研究平均复杂性一样，我们也可以研究输入服从分布 $\mathcal{D}$ 时的通信复杂性。平均通信复杂度的定义是

$$C_{\mathcal{D}}(f)=\min_{\substack{\text{协议}\mathcal{P}\\ \text{计算}f}}\underset{(x,y)\in_R\mathcal{D}}{\mathrm{E}}[\text{协议}\mathcal{P}\text{在}x,y\text{上交换的二进制位的个数}]$$

非布尔函数的计算：此时，函数的输入不再属于 $\{0,1\}$，而是一个 m 位的数（其中 m 是某个整数）。我们将在习题中给出一个例子。

非对称通信：在这种通信模型下，通信“代价”是非对称的。也就是说，存在一个 B 使得第一参与方发送每个二进制位的代价是第二参与方的 B 倍。通信协议的目标是最小化总的代价。

关系的计算：通信协议除了可以用来计算函数之外，还可以考虑用它来计算关系。也

就是说，我们有一个关系 $R\subseteq\{0,1\}^n\times\{0,1\}^n\times\{1,2,\cdots,m\}$，给定 $x,y\in\{0,1\}^n$ 之后，通信参与双方需要找出一个 $b\in\{1,2,\cdots,m\}$ 使得 $(x,y,b)\in R$。参见习题 13.16。

关于这些模型以及更多其他模型的讨论，请参见[KN97]。

本章学习内容

- 对于具有两个输入的函数 f，它的通信复杂度指的是在一个参与方持有 x 而另一个参与方持有 y 的情况下双方协同计算 $f(x,y)$ 的过程中需要交换的二进制位的个数。
- 为各种具体的函数证明通信复杂性下界的方法包括：诈集方法、铺砌方法、秩方法和差异方法。利用这些方法，我们在几个具体的以两个 n 位串为输入的函数上证明了它们的通信复杂度至少为 n。
- 对于具有 k 个输入的函数 f，它的多方通信复杂度指的是在每个参与方 i 知道除 x_i 之外的其他输入的前提下 k 个参与方协同计算 f 值的过程中需要交换的二进制位的个数。人们已经证明，对于任何明确的函数，它的 k 方通信复杂度的最佳下界都形如 $n/2^{o(k)}$。
- 人们还研究了通信复杂性的其他模型，包括随机通信复杂性、非确定型通信复杂性、平均通信复杂性，以及关系计算的通信复杂性。

本章注记和历史

本章仅蜻蜓点水式地介绍了复杂性理论中的一个自包含的迷你领域——通信复杂性；更精彩、更详细的论述请参阅库思勒维兹(Kushilevitz)和尼散(Nisan)的书[KN97](但该书缺少最新的研究结果)。

通信复杂性的定义源自姚期智[Yao79]。为这一领域奠定基础的其他早期论文还包括帕帕迪米特里奥(Papadimitriou)和西普赛尔(Sipser)[PS82]，梅尔伯恩(Mehlborn)和施密特(Schmidt)[MS82](他们引入了秩下界)，以及阿霍(Aho)，阿曼(Ullman)和杨纳卡卡斯(Yannakakis)[AUY83]。

我们曾在第 6 章简要地讨论了并行计算。姚期智[Yao79]发明通信复杂性，并用它来分析某些计算任务的平行算法的运行时间下界。其思想如下，由于输入分布于许多处理器上，如果将这些处理器划分为两个大小相当的部分，则并行计算的时间复杂度下界可以通过考虑两部分处理器之间通信的二进制位个数的下界来给出。类似的思想可以用来证明超大规模集成电路(VLSI)的时间-空间下界。例如，在布线为 $m\times m$ 网格的 VLSI 芯片中，如果函数 f 的通信复杂度为 c，则计算该函数的时间将至少是 c/m。

通信复杂性还可以用来证明图灵机的时间-空间复杂性下界(参见习题 13.4)和线路下界(参见第 14 章)。

堆、有序数组和链表是算法设计中常用的基本数据结构。通常，算法设计者需要确定他们设计的数据结构是不是最优的。通信复杂性可以用来建立这样的结论，参见[KN97]。流算法只能对大规模输入扫描一遍。在这一领域中，通信复杂性的界限既可以用来表明流算法的最优性，也可以用来表明流算法的某种不可能性。阿龙(Alon)，马蒂亚斯(Matias)和塞盖迪(Szegedy)[AMS96]率先以通信复杂性为工具来证明流算法的复杂性下界。以后，人们开展了大量的研究工作；一方面，人们将通信复杂性应用到流算法的各种下界问题

上；另一方面，受到"研究如何在频率估算之类的流问题中进一步减小现存的鸿沟"这种需求的促进，人们还提出了证明通信复杂性下界的新工具（可以参见如[CSWY01，BYJKS02]）。

杨纳卡卡斯(Yannakakis)[Yan88]说明了如何用通信复杂性下界来证明"表示 **NP** 完全问题所需的多胞形的规模"的下界。求解了习题 13.13 中提到的待决问题也就给出了"表示顶点覆盖问题"所需多胞形的下界。

定理 13.24 源自巴拜(Babai)，尼散(Nisan)和塞盖迪(Szegedy)，但是我们给出的证明是莱斯(Raz)化简[Raz00]Chung 的证明[Chu90]之后得到的。

实数矩阵的差异也称为斜范数。差异的计算是 **NP** 难问题，甚至将差异近似到任意小的常数精度也是 **NP** 难问题。但是，用半正定规划可以将差异近似到常数因子 K_G 范围之内，参见阿龙(Alon)和纳尔(Naor)[AN04]。K_G 也称为格罗坦狄克常数(Grothendieck Constant)，它介于 1.5 到 1.8 之间，但确定其精确值目前仍是一个主要的待决问题。差异概念在著名的塞梅尔雷迪正则性引理(Szemerédi regularity lemma)[Sze76]中指的就是正则性。在该引理中，参数$\mathcal{E}(f)$类似于在给定的二分图中四个顶点构成的环的个数占所有四个顶点的组合总数的比例，因此它跟图的正则性直接相关。多方差异与超图的正则性引理相关，高尔斯(Gowers)[Gow07]在证明该引理的过程中用到了参数$\mathcal{E}(f)$。群论中与$\mathcal{E}(f)$密切相关的一个参数称为高尔斯范数或高尔斯一致性；高尔斯[Gow01]曾用这个参数改进了塞梅尔雷迪定理(Szemerédi Theorem)中的数量，而该定理断言稠密集合中必然存在较大的算术级数。陶哲轩(Terence Tao)和 Vu Ha Wan[TV06]对这些专题进行了深入的研究。

洛瓦兹(Lovasz)和赛克斯(Saks)已经注意到对数秩猜想(log rank conjecture)与离散数学中关于"邻接矩阵和色数之间的关联关系"的一个猜想之间的联系。对数秩猜想首先由莱斯(Raz)和斯皮克尔(Spieker)发现[RS93]，其原始形式是 $C(f)=O(\log \mathrm{rank}(M(f)))$。秩方法和许集方法的比较最早出现在迪茨费尔宾格(Dietzfelbinger)，赫罗姆科维克(Hromkovic)和施尼特格尔(Schnitger)的论文[DHS94]中。

一般地，$C(f)$和$C_\varepsilon(f)$的计算复杂性仍未深入地得到理解，这或许就是人们在实践中难以给出 $C(f)$和$C_\varepsilon(f)$的下界的原因。此外，让人感到困惑的是已经证得的通信复杂性下界都与某个多项式时间可计算的数量有关。比如，给定矩阵 $M(f)$之后，它的秩可以在多项式时间内被计算出来。(第 23 章还会遇到更广泛的"自然证明"现象。)就本章的知识点而言，还需要注意到：差异这一参数的计算是 **NP** 难的，但是在双方通信中差异可以用多项式时间算法[AN04]近似到一个常数因子范围内。相比之下，三方通信中差异的计算非常难(尽管还没有关于其难度的任何结果发表)，这或许可以解释为什么要获得多方通信复杂性的下界如此困难。

本章未涉及的一个较新的研究领域是量子通信复杂性，它讨论粒子与其他粒子之间的量子态交换，参见[Bra04]。有意思的是，量子通信复杂性领域中的一些技术[She07]可以用来得到不相交函数(DISJ)的新的 k 方通信复杂度下界 $\Omega(n^{1/(4^k-1)}/2^{2^k})$[LS07，CA08]，这一结果将非确定型 k 方通信复杂性与确定型 k 方通信复杂性明确地区分开来。

习题

13.1　证明引理 13.5。

13.2　证明：对任意集合 $S\subseteq\{(x,x):x\in\{0,1\}^n\}$和任意能够在 n 位串输入上正确地计

算相等函数的通信协议Π，S 中必存在一个输入序对使得Π在该序对上执行时至少
283 交换 $\log|S|$ 个二进制位。

13.3　证明：单带图灵机(其输入带也是它的读写工作带)至少需要用 $O(n^2)$ 时间才能判定例 1.1 给出的回文语言 $\mathrm{PAL}=\{x_1\cdots x_1x_1\cdots x_{n+1}\ x_1,\ \cdots,\ x_n\in\{0,\ 1\}^*,\ n\in\mathbf{N}\}$。

13.4　设 $S(n)\leqslant n$，证明：$S(n)$空间图灵机至少需要用 $\Omega(n^2/S(n))$ 个计算步骤才能判定语言 $\{x\#x:\ x\in\{0,\ 1\}^*\}$。

13.5　证明定理 13.8 的第二个不等式，亦即，证明：$C(f)=O(\log^2\chi(f))$ 对任意函数 $f:\ \{0,\ 1\}^n\times\{0,\ 1\}^n\to\{0,\ 1\}$ 成立。

13.6　证明引理 13.10。

13.7　证明：在几乎所有的函数 $f:\ \{0,\ 1\}^n\times\{0,\ 1\}^n\to\{0,\ 1\}$ 上，矩阵 $M(f)$ 在 $GF(2)$ 上的秩是 n，且最大诈集的大小小于 $3\log n$。这就表明，用秩方法得出的通信复杂性下界指数地优于用诈集方法得出的通信复杂性下界。

13.8　对于两个 $n\times n$ 的矩阵 A，B，它们的张量积 $A\otimes B$ 定义为一个 $n^2\times n^2$ 的矩阵，其中的每个位置用 $[n]$ 上的四元组表示[⊖]。证明：在任何域上，$A\otimes B$ 的秩都等于 A 的秩乘以 B 的秩。

利用上述事实，证明：如果函数 f 有一个大小为 S 的诈集，则秩方法可以给出通信复杂性的一个至少为 $(1/2)\lceil\log S\rceil$ 的下界。这就表明，秩方法不会远远劣于诈集方法。

13.9　证明：如果 M 是 0/1 实数矩阵，M' 是在 M 上应用变换 $a\mapsto(-1)^a$ 后得到的 ±1 矩阵，则 $\mathrm{rank}(M)-1\leqslant\mathrm{rank}(M')\leqslant\mathrm{rank}(M)+1$。

13.10　将 $GF(2)^n$ 中的元素 x，y 视为向量，令 $f(x,\ y)$ 是 x，y 的内积(mod 2)。利用秩方法，证明：f 的通信复杂度是 n。

13.11　设在 $f:\ \{0,\ 1\}^n\times\{0,\ 1\}^n\to\{0,\ 1\}$ 的矩阵 $M(f)$ 中各行互不相同，证明：$C(f)\geqslant\log n$。

13.12　证明：$\varepsilon(IP)\leqslant2^{-n}$，其中 IP 是内积函数。由此得出 IP 的通信复杂度的一个下界。

13.13　对任意具有 n 个顶点的图 G，考虑如下的通信问题。参与方 1 获得的输入是 G 的一个团 C，参与方 2 获得的输入是 G 的一个独立集 I。参与双方要通过通信计算 $|C\cap I|$。注意，$|C\cap I|$ 要么等于 0，要么等于 1。证明：上述计算问题的通信复杂度的上界为 $O(\log^2n)$。

你能改进上述上界，或者证明一个优于 $\Omega(\log n)$ 的下界吗？(该问题目前仍未解决。)

284　13.14　证明引理 13.23。

13.15　证明：相等函数的随机通信复杂度 $R(EQ)$ 至多为 $O(\log n)$。(注意：随机通信协议输出错误答案的概率至多为 1/3。)

13.16　(Karchmer-Wigderson 游戏[KW88])考虑如下的关系计算问题。将下面的通信问题关联到任意函数 $f:\ \{0,\ 1\}^n\to\{0,\ 1\}$ 上。参与方 1 持有满足 $f(x)=0$ 的输入 x，而参与方 2 持有满足 $f(y)=1$ 的输入 y。参与双方需要通过通信确定一个位置 i 使得 $x_i\neq y_i$。证明：上述问题的通信复杂度恰好等于计算 f 的所有线路的最小深度。(每个逻辑门的最大扇入度是 2。)

⊖　此处对张量积的定义不够清晰，请读者参见 11.5.2 节中的脚注。——译者注

13.17　利用习题 13.16，证明：计算 n 个二进制位的奇偶性需要深度至少为 $2\log n$ 的线路。

13.18　证明如下的计算问题属于 **EXP**：给定布尔函数的矩阵 $M(f)$ 和数 K，判定 $C(f)\leqslant K$ 是否成立。

你能证明上述问题是某个复杂性类的完全问题吗？（自从姚期智[Yao 79]提出这个问题后，直到现在它仍未能被解决。）

13.19　([AMS96])一个 $S(n)$空间的流算法是一个仅对输入进行一遍扫描的空间复杂度为 $S(n)$的图灵机。这种情形非常自然地出现在许多应用中。证明：不存在 $o(n)$空间的流算法来求解如下的计算问题：给定[n]中的元素构成的一个序列 x_1，…，x_n，计算序列中元素的最大频率 $\max_{x\in[n]}|\{i:x_i=x\}|$。你能证明上述问题甚至不可能在 $o(n)$空间内求得它的 3/4-近似解吗？

285

线路下界：复杂性理论的滑铁卢

我们在第 2 章中已经看到，如果存在一个 **NP** 语言不能被多项式规模的线路计算，则 **NP**≠**P**。因此，证明线路下界是证明 **NP**≠**P** 的一种方法。而且，这是一种有望获得成功的方法，因为卡普-利普顿定理(定理 6.19)已经证明了：如果多项式分层(**PH**)不坍塌，则存在 **NP** 语言不能被任何多项式规模的线路计算。

二十世纪 70 年代和 80 年代，许多研究者确信证明线路下界是解决 **P** 与 **NP** 这一问题的最佳路线，因为用线路进行推理易于用图灵机进行推理。沿着这一路线开展的研究工作有其成功的一面，也有失败的一面。

关于一般线路的下界，研究工作几乎毫无进展。任何函数，只要函数值依赖于输入中的所有二进制位，则人们只知道它的复杂度的平凡下界是 n。人们甚至无法给出一个超线性的复杂性下界使得它对于任何 **NP** 问题都成立。经过多年的努力，人们只证明了 $5n-o(n)$ 是任何 **NP** 问题的复杂性下界。

为使得复杂性下界的证明变得相对容易一些，研究者们将注意力放到受限的线路族上，并证明了一些良好的下界。本章证明这一领域获得的主要结果中的一部分。具体地讲，我们只讨论深度受限的线路(14.1 节)，具有"计数器"门的深度受限线路(14.2 节)和单调线路(14.3 节)。在所有研究结果中，我们将用到计算的"进度"这一概念。我们将证明，小规模线路无法获得由输入得到输出所必需的计算进度。

14.4 节指出线路下界研究的前沿中目前面临的主要问题，指明目前的研究在何处陷入了困境。任何想在该领域从事研究工作的新手都可以着手研究我们指出的任何一个待决问题。后面的第 23 章将指出，要想使这方面的研究获得突破需要克服的一些固有的困难。

14.1 AC^0 和哈斯塔德开关引理

正如我们在第 6 章中所见，复杂性类 $\mathbf{AC}^0$ 中的每个语言都可以用具有常数深度的多项式规模的逻辑门扇入度都不受限的布尔线路来计算。之所以要求扇入度不受限，是因为否则的话逻辑门的输出无法从所有输入二进制位中获取信息。

286

我们曾在第 2 章(论断 2.13)中看到，任意布尔函数都可以用一个深度为 2 的规模为指数的线路来计算；也就是说，任意布尔函数都可以用一个 CNF(或 DNF)范式来计算。学生们在学习数字逻辑设计时，学会了如何用卡诺图(Karnaugh map)来找出"最小线路"，其中的线路指的就是深度为 2 的线路。事实上，用卡诺图技术容易证明，即使将非常简单的函数(例如下面即将介绍的奇偶性函数)表示为 CNF 范式或 DNF 范式时，表示结果也具有指数规模。但是，卡诺图这样的技术甚至难以推广到深度为 3 的布尔线路上，更不用说用来处理类 $\mathbf{AC}^0$(参见 6.7.1 节)中具有任意常数深度的多项式规模线路了。

二十世纪 70 年代，最热的研究焦点就是考虑团和 TSP 之类的问题是否存在 $\mathbf{AC}^0$ 线路。1981 年，弗斯特(Furst)，萨克斯(Saxe)和西普塞尔(Sipser)证明了这类问题不存在 $\mathbf{AC}^0$ 线路，奥伊陶伊(Ajtai)也独立地证明了这一结果。事实上，他们给出的下界在更简单的函数上也成立。

定理 14.1（[FSS81，Ajt83]）　设⊕是奇偶性函数，亦即，对任意 $x \in \{0,1\}^n$，$\oplus(x_1,\cdots,x_n)=\sum_{i=1}^{n} x_i \pmod 2$。则⊕∉$\mathbf{AC}^0$。

证明定理 14.1 的主要工具是随机限制的概念。设 f 可以被一个深度为 d 的多项式规模的线路计算。假设随机地选择绝大部分输入变量（亦即选择 $n-n^{\varepsilon}$ 个变量，其中 $\varepsilon>0$ 是依赖于 d 的常数），并将它们随机地固定为 0 或 1。我们可以证明 f 在上述限制条件下将以很高的概率取常数值（亦即要么总等于 0，要么总等于 1）。对于奇偶性函数，由于在输入变量的任意子集上固定取值之后都无法保证函数取常数值，因此奇偶性函数不能用深度为常数的多项式规模线路来计算。

14.1.1　哈斯塔德开关引理

同 2.3 节一样，我们定义 k-CNF 范式是将 AND 作用在若干个 OR 子句上得到的布尔表达式，其中每个 OR 子句至多涉及 k 个变量。类似地，k-DNF 公式是将 OR 作用在若干个 AND 子句上得到的布尔公式，其中每个 AND 子句至多涉及 k 个变量。如果 f 是定义在 n 个变量上的函数而 ρ 是 f 中所有变量的一个部分赋值（ρ 也称为一个限制），则用 $f|_\rho$ 表示 f 在 ρ 下的限制。也就是说，$f|_\rho$ 的输入是一个赋值 τ，其中由 τ 赋值的变量均未在 ρ 中被赋值，$f|_\rho$ 的输出等于将 f 作用到 τ 和 ρ 上得到的输出。下面，我们证明如下的主要引理，以此表明线路在随机赋值下是如何被简化的。

引理 14.2（哈斯塔德开关引理[Hås86]）　假设 f 可以表示为一个 k-DNF 范式，并且随机限制 ρ 对 t 个随机选择的输入变量进行了随机赋值。那么，对于任意 $s\geqslant 2$，均有

$$\Pr_{\rho}[f|_\rho \text{不能表示为} s\text{-CNF 范式}] \leqslant \left(\frac{(n-t)k^{10}}{n}\right)^{s/2} \tag{14.1}$$

其中 $f|_\rho$ 是函数 f 在部分赋值 ρ 下的限制。

287

我们将引理 14.2 的证明延后到 14.1.2 节给出。在使用引理 14.2 时，典型的设置是用 k，s 表示常数且 $t\approx n-\sqrt{n}$。这样，引理给出的概率上界将形如 n^{-c}，其中 c 是一个常数。注意，引理 14.2 应用到函数$\neg f$上将得到类似的结论，只需把引理中 DNF 和 CNF 的位置调换。

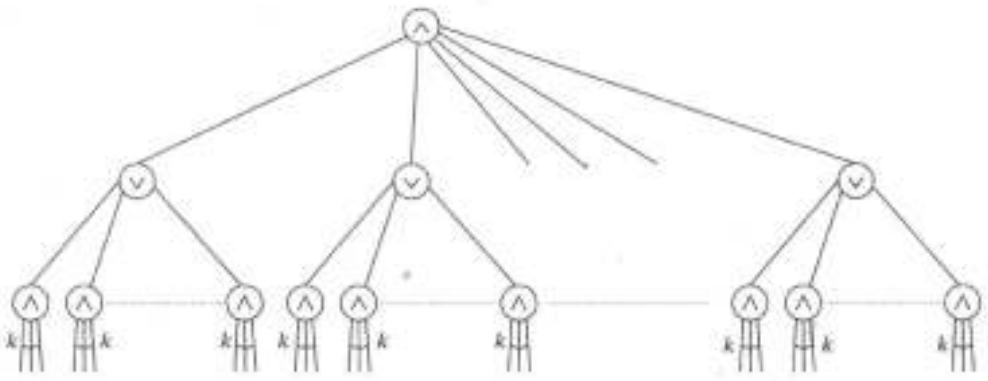


图 14-1　应用哈斯塔德变换之前的线路

用引理 14.2 证明定理 14.1

现在，我们说明为什么哈斯塔德引理蕴含着奇偶性函数不属于 $\mathbf{AC}^0$。我们从任意 $\mathbf{AC}^0$ 线路开始，先将它简化为如下的形式（简化过程非常直接，我们将其留作习题 14.1 和习

题 14.2)：(a)所有扇出度都等于 1，线路是一棵树；(b)所有 NOT 门全部位于输入层，也就是说，线路的输入是 $2n$ 个二进制位，后 n 个位分别是前 n 个位的否定；(c)∧门和∨门在树中交替出现，也就是说，树的每一层要么全是∧门要么全是∨门；(d)树的最底层是扇入度为 1 的∧门。

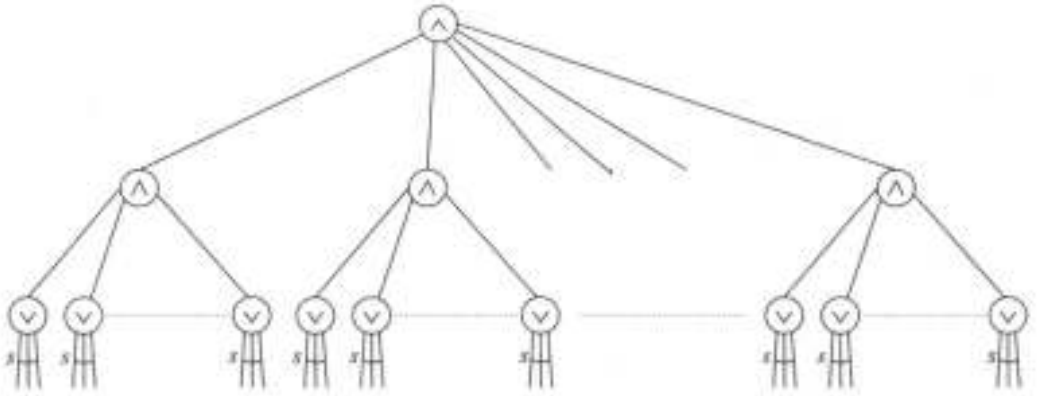


图 14-2　哈斯塔德变换之后的线路。注意，新层中的所有∧门可以坍塌到它们的父∧门上，这样线路中的层数就减小了 1

将满足上述性质的线路中逻辑门个数的上界记为 n^k。我们逐步地随机地限制一些变量的取值：在每一个步骤中，我们将线路的深度减小 1，并确保底层逻辑门的扇入度是一个常数。具体地讲，假设第 i 步随机限制之后未被限制的变量仍有 n_i 个，则我们在第 $i+1$ 步中将限制 $n_i-\sqrt{n_i}$ 个变量的取值。由于 $n_0=n$，故 $n_i=n^{1/2^i}$。令 $k_i=10b2^i$。我们将证明，在第 i 步随机限制之后将会高概率地得到一个深度为 $d-i$ 且底层扇入度至多为 k_i 的线路。事实上，假设底层逻辑门全部是∧门，而其上一层的逻辑门全部是∨门。于是，这些∨门所计算的函数全部是 k_i-DNF 范式；根据引理 14.2 可知，每个∨门所计算的 k_i-DNF 范式在第 $i+1$ 步随机限制之后可以改写为 k_{i+1}-CNF 范式的概率为 $1-\left(\frac{k_i^{10}}{n^{1/2^{i+1}}}\right)^{k_{i+1}^2}$（当 n 充分大之后，该概率至少为 $1-1/(10n^k)$）。由于 CNF 范式的根结点是∧门，我们可以把这些∧门和在其上的∧门混合，进而把线路深度减少 1(参见图 14-1 和图 14-2)。对称地，可以将推论应用于底层∨门上，运用引理将这层上的 k_i-CNF 转变为 k_{i+1}-DNF。注意，我们在原始线路中的每 n^k 个逻辑门上至多应用引理 14.2 一次。根据合并界限(即 A.1 式)，随机限制进行 $d-2$ 步之后，我们将以 9/10 的概率得到一个深度为 2 且底层扇入度为 $k=k_{d-2}$ 的线路。这种线路要么是一个 k-CNF 范式，要么是一个 k-DNF 范式。在这种布尔公式中，最多只需要固定 k 个变量的取值，就可以保证整个公式取常数值(比如，对于 DNF 的情形，只需将它的第一个子句中的所有变量都固定为 1，则整个公式的值就等于 1)。在奇偶性函数中，由于任意固定少于 n 个变量的取值都无法确保整个函数取常数值，这就表明定理 14.1 成立。■

288

14.1.2　开关引理的证明

下面，我们证明引理 14.2。我们给出的证明源于莱兹波佛夫(Razborov)，原始证明要复杂一些。

我们先引入几个新的定义。函数 f 的最小项(min-term)是 f 的一个部分赋值，它使得

无论 f 的其余变量如何赋值，f 都输出 1。因此，f 的 k-DNF 范式中的每个项都可以得到 f 的一个大小为 k 的最小项。函数 f 的最大项(max-term)是 f 的一个部分赋值，它使得无论 f 的其余变量如何赋值，f 都输出 0。因此，f 的 k-CNF 范式中的每个项都可以得到 f 的一个大小为 k 的最大项。我们将始终假设最小项(或最大项)都是极小的，也就是说，最小项(或最大项)中所有变量的真子集的任何赋值都不会使得 f 仅输出 1(或 0)。因此，如果一个函数不能被表示为 s-CNF 范式，则它至少存在一个大小为 $s+1$ 的最大项(参见习题 14.3)。

如果 π 和 ρ 是对两组无公共元素的变量上的限制，则我们用 $\pi\rho$ 表示这两个限制的合并。也就是说，如果 π 对 S 中的变量进行赋值而 ρ 对 T 中的变量进行赋值，则 $\pi\rho$ 将分别根据 π 和 ρ 对 $S\cup T$ 中的变量进行赋值。

令 R_t 是 t 个变量上的所有可能的限制构成的集合，其中 $t\geqslant n/2$。注意，$|R_t|=\binom{n}{t}2^t$。如果 $\rho\in R_t$ 使得 $f|_\rho$ 不能表示成一个 s-CNF 范式，则称 ρ 是一个劣性限制。令 B 是所有劣性限制构成的集合。要证明引理，需要证明 B 很小。为此，我们给出一个一对一的映射[⊖]将 B 映射到 R_{s+t} 和 $\{0,1\}^\ell$ 的笛卡尔积中，其中 R_{s+t} 是 $s+t$ 个变量上的所有限制构成的集合而 $\ell=O(s\log k)$。该笛卡尔积的大小为 $\binom{n}{t+s}2^{s+t}2^{O(s\log k)}=\binom{n}{t+s}2^t k^{O(s)}$，因此劣性限制发生的概率 $|B|/|R_t|$ 的上界为

$$\frac{\binom{n}{t+s}2^t k^{O(s)}}{\binom{n}{t}2^t}=\frac{\binom{n}{t+s}k^{O(s)}}{\binom{n}{t}} \tag{14.2}$$

289

直观上看，上述概率非常小，因为 k，s 被视为常数。因此对于非常接近于 n 的 t 值，我们有 $\binom{n}{t+s}\approx\binom{n}{t}/n^s$ 和 $n^s\gg k^{O(s)}$，这意味着(14.2)式的上界为 $n^{-\Omega(s)}$。正式地，利用近似公式 $\binom{n}{t}\approx(ne/a)^a$ 可以证明(14.1)式给出的上界，我们将证明过程留作习题 14.4。

因此，要证明引理，只需给出前面提到的一对一的映射。我们对 k-DNF 范式 f 的劣性限制 ρ 进行推理。在劣性限制 ρ 下，f 的任何一个项的取值都不等于 1；否则，$f|_\rho$ 将是一个常值函数。因此，f 的某些项取值为 0，但并不是所有的项都取值为 0；否则 $f|_\rho$ 也将是一个常数值。事实上，由于 $f|_\rho$ 不能被表示为一个 s-CNF 范式，它必然存在一个大小至少为 s 的最大项，我们将这个最大项记为 π。也就是说，π 是 ρ 未赋值的某些变量上的一个限制，它使得 $f|_{\rho\pi}$ 恒等于 0 但 $f|_{\rho\pi'}$ 不恒等于 0，其中 π' 是 π 的任意一个真子限制。我们的大致思想是，一对一的映射将把 ρ 映射为 $\rho\sigma$，其中 σ 是 π 中变量的一个恰当的赋值，这样 $\rho\sigma$ 就至少限制了 $t+s$ 个变量的取值。

将 f 的所有项以任意顺序排列为 t_1，t_2，…并在每个项的内部将所有变量任意地排定一个顺序。根据前面的定义，在限制 $\rho\pi$ 下函数 f 取值为 0，因此 f 的每个子句均取值为 0。我们如下地将 π 分裂成 $m\leqslant s$ 个子限制 π_1，…，π_m。假设我们已经找出了 π_1，…，π_{i-1} 且 $\pi_1\pi_2\cdots\pi_{i-1}\neq\pi$。用 t_{j_i} 表示 f 的所有项中在限制 $\rho\pi_1\cdots\pi_{i-1}$ 下不等于 0 的第一个项；这样的项必然存在，因为 π 是一个最大项而 π 的真子集 $\pi_1\cdots\pi_{i-1}$ 不是最大项。令 Y_i 是 t_{j_i} 中的被 π 赋值但未被 $\rho\pi_1\cdots\pi_{i-1}$ 赋值的所有变量构成的集合。由于 π 使得 t_{j_i} 取值为 0，故 Y_i 不是空

⊖ 即单射。——译者注

集。我们定义 π_i 是将 Y_i 中的所有变量按 π 的方式进行赋值时得到的限制；于是 π_i 使得 t_i 取值为 0。定义 σ_i 是 Y_i 的使得 t_i 取值不等于 0 的一个限制；这样的限制必然存在，否则 t_i 将恒等于 0。重复上述过程，直到首次找到一个限制 π_m 使得 π_1，…，π_m 至少对 s 个变量进行了赋值；此时，σ_1，…，σ_m 也至少对 s 个变量进行了赋值。必要时需要对 π_m 进行裁剪（亦即让它少赋值几个变量）使得 π_1，…，π_m 恰好对 s 个变量进行赋值。

我们的一对一的映射将把 ρ 映射为 $(\rho\sigma_1\cdots\sigma_m, c)$，其中 c 将在下一自然段中定义。它是一个长度为 $O(s\log k)$ 的二进制位串。为了证明这个映射是一对一的，我们需要说明如何唯一地将映射过程逆转。相比映射过程而言，逆转映射的过程更难一些，因为不存在任何先验知识帮助我们从 $\rho\sigma_1\cdots\sigma_m$ 中找出 ρ。事实上，辅助信息 c 恰好是为了完成上述任务而设置的，其定义如下。

假设给定赋值 $\rho\sigma_1\cdots\sigma_m$。我们将 $\rho\sigma_1\cdots\sigma_m$ 表示的赋值逐渐地代入 f 就可以找到哪一项是 t_{i_1}，亦即 f 的第一个取值固定但不等于 0 的项。（它是在 ρ 的限制下第一个不等于 0 的项，这一性质由 σ_1 维护，而 $\sigma_2\cdots\sigma_m$ 不再对项 t_{i_1} 中的变量进行赋值。）令 s_1 是 π_1 中的变量个数。c 将包含 s_1，π_1 中各个变量在 t_{i_1} 中的索引号，以及 π_1 对各个变量的具体赋值。注意，一旦知道了 t_{i_1}，则 c 中维护的相应信息足以用来重构 π_1。而且，由于 f 的每个项至多包含 k 个变量，在 c 中维护 π_1 的相关信息仅需 $O(s_1\log k)$ 个二进制位。重构 π_1 之后，我们可以将部分赋值 $\rho\sigma_1\cdots\sigma_m$ 修改为 $\rho\pi_1\cdots\sigma_m$，然后再去找出哪个项是 t_{i_2}。同样，它恰好是在新的部分赋值 $\rho\pi_1\cdots\sigma_m$ 下第一个取值固定但不等于 0 的项，因为 π_1 使得 t_{i_1} 取值为 0。因此，c 中接下来的 $O(s_2\log k)$ 个二进制位表示了赋值 π_2（其中 s_2 是 π_2 中变量的个数）。重复上述过程直到找出
290 m 个项并重构得出 π_3，…，π_m。现在，我们“取消”π_1，…，π_m 对各个变量的赋值就可以重构得出 ρ。这就说明，我们给出的映射是一对一的。存储辅助信息的总长度为 $O((s_1+s_2+\cdots+s_m)\log k)=O(s\log k)$。■

14.2　带“计数器”的线路：ACC

在上一节中的 $\mathbf{AC}^0$ 下界被证明之后，研究者们曾经致力于将该下界扩展到更具一般性的线路族上。最简单的扩展是，允许线路中使用 $\vee$ 门和 $\wedge$ 门之外的其他逻辑门，同时仍保证线路的深度为 $O(1)$。例如，人们考虑过的一个简单的逻辑门是奇偶门，它计算输入串中等于 1 的二进制位的个数的奇偶性。显然，在 $\mathbf{AC}^0$ 线路上即使简单地添加一个奇偶门，所得到的线路就可以用来计算奇偶函数了。但是，是否存在明确的函数不能被这种线路计算呢？莱兹波佛夫（Razborov）利用近似方法率先证明这种线路的一个下界。斯莫伦斯基（Smolensky）后来扩展了这一结果，并将该方法应用到本节研究的线路族上。

定义 14.3（$\mathbf{ACC}^0$）　对任意整数 m，如下定义 MOD_m 门。如果输入中所有二进制位之和模 m 等于 0，则 MOD_m 输出 0；否则，MOD_m 输出 1。

对于整数 m_1，m_2，…，$m_k>1$，我们称语言 L 属于 $\mathbf{ACC}^0(m_1, m_2, \cdots, m_k)$，如果存在一个由 $\wedge$ 门、$\vee$ 门、$\neg$ 门和 MOD_{m_1}，…，MOD_{m_k} 门构成的深度为常数的多项式规模的扇入度不受限制的线路族 $\{C_n\}$ 接受 L。

复杂性类 $\mathbf{ACC}^0$ 包含任意 $\mathbf{ACC}^0(m_1, m_2, \cdots, m_k)$ 的所有语言，其中 $k\geqslant 0$ 且 m_1，m_2，…，$m_k>1$ 是整数。

人们仅在只含一类特殊模门的线路上证得了良好的下界。

定理 14.4（莱兹波佛夫-斯莫伦斯基[Raz87, Smo87]）　对于不同的素数 p 和 q，函

数 MOD_p 不属于 **ACC**0(q)。

我们仅通过证明"奇偶性函数不能被任何 **ACC**0(3)计算"来展示定理证明的主要思想。

证明　证明过程分为两个步骤。

第 1 步：本步骤将通过对 h 用数学归纳法来证明：对于 n 位输入上的深度为任意 h 且规模为 S 的每个 MOD_2 线路，均存在一个次数为 $(2l)^h$ 的多项式使得它与线路在比例为 $1-S/2^l$ 的输入上具有一致的输出。如果线路 C 的深度为 d，则令 $2l=n^{1/2d}$ 就得到一个次数为 $\sqrt{n}$ 的多项式，它与 C 在比例为 $1-S/2^{n^{1/2d}/2}$ 的输入上具有一致的输出。

第 2 步：本步骤将证明不存在次数为 $\sqrt{n}$ 的多项式与 MOD_3 在比例为 49/50 的输入上具有一致的输出。

将两个步骤的结论放在一起则表明，计算 MOD_3 的深度为 d 的任意线路至少具有 $S>2^{n^{1/2d}/2}/50$ 的规模，这就完成了定理的证明。下面详细地讨论两个步骤。 291

第 1 步：考虑线路中深度为 h 的一个结点 v（如果 v 是输入顶点，则定义其深度为 0）。如果 $g(x_1, \cdots, x_n)$ 是结点 v 计算的函数，则我们构造 GF(3) 上的一个 $(2l)^h$ 次多项式 $\widetilde{g}(x_1, \cdots, x_n)$ 使得 $g(x_1, \cdots, x_n)=\widetilde{g}(x_1, \cdots, x_n)$ 对"绝大部分" $x_1, \cdots, x_n \in \{0, 1\}$ 成立。我们还会确保 $\widetilde{g}$ 的输出属于 $\{0, 1\}$ 对 $\{0, 1\}^n \subseteq$ GF(3) 中的任意输入均成立。这不会丧失一般性，因为这可以通过取多项式的平方来实现（回顾一下，GF(3) 中的元素只有 0，-1，1，并且 $0^2=0$，$1^2=1$ 而 $(-1)^2=1$）。

我们归纳地构造近似多项式。如果 $h=0$，则相应的"门"是一个输入位 x_i，它恰好可以表示为 1 次多项式 x_i。现在，假设我们已经为深度小于等于 $h-1$ 的结点构造了相应的近似多项式，而 g 是深度为 h 的一个门。

1. 如果 g 是 NOT 门，则 $g=\neg f_1$ 对某个深度小于等于 $h-1$ 的门成立。归纳假设给出了 f_1 的一个近似多项式 $\widetilde{f}_1$。于是，我们用 $\widetilde{g}=1-\widetilde{f}_1$ 作为 g 的近似多项式，该多项式的次数与 $\widetilde{f}_1$ 的次数相同。注意，只要 $f_1(x)=\widetilde{f}_1(x)$，则有 $g(x)=\widetilde{g}(x)$，因此我们并未引入新的错误。

2. 如果 g 是 MOD_3 门，不妨设它的输入是 f_1，…，f_k。此时，用 $\widetilde{g}=\left(\sum_{i=1}^{k}\widetilde{f}_i\right)^2$ 作为 g 的近似多项式。该多项式的次数至多变为 $2\times(2l)^{h-1}<(2l)^h$。由于 $0^2=0$，$(-1)^2=1$，因此我们并未引入新的错误。

3. 如果 g 是 AND 门或 OR 门，则我们需要小心地处理。我们只给出 OR 门的近似多项式；由德摩根定律，AND 门可以类似地进行处理。不妨设 $g=\bigvee_{i=1}^{k} f_i$。一种简单直接的方法是，用 $OR(\widetilde{f}_1, \cdots, \widetilde{f}_k)=1-\prod_{i=1}^{k}(1-\widetilde{f}_i)$ 作为 g 的近似多项式。不幸的是，这样做将导致近似多项式的次数等于 $\widetilde{f}_i$ 的次数乘以 k（g 门的扇入度），但 k 远远大于 $2l$。正确做法将允许引入新的错误。具体地讲，若 $g=\bigvee_{i=1}^{k} f_i$，则在输入 x 上 $g(x)=1$ 当且仅当至少有一个 f_i 在输入 x 上输出 1。而且，根据随机子和原理（参见论断 A.31）可知，如果存在 i 使得 $f_i(x)=1$，则"$\{f_i(x)\}$ 的随机子集在 GF(3) 上的和是非 0 值的概率"将至少为 1/2。

随机选取 $\{1, \cdots, k\}$ 的 l 个子集 T_1，…，T_l，然后计算 l 个多项式 $\left(\sum_{i\in T_1}\widetilde{f}_i\right)^2, \cdots, \left(\sum_{i\in T_l}\widetilde{f}_i\right)^2$，其中每个多项式的次数至多是次数最高的输入多项式的次数的 2 倍，将上一

自然段中介绍的简单直接方法应用到这些多项式上，将得到一个次数至多为 $2l\times(2l)^{k-1}=(2l)^k$ 的多项式。相对于随机子集的选取，所得多项式与 $OR(\overline{f}_1,\cdots,\overline{f}_k)$ 不一致的概率至多为 $\frac{1}{2^l}$。根据概率论证法，必然存在 l 个子集的一种取法使得相对于 x 的随机选取而言所得多项式与 $OR(\overline{f}_1,\cdots,\overline{f}_k)$ 不一致的概率至多为 $\frac{1}{2^l}$。我们用这些子集上构造得到的多项式作为 g 的近似多项式 $\tilde{g}$。

在深度为 d 的线路中，为每个门依次运用上述方法，最终为整个线路的输出门构造出一个 $(2l)^d$ 次的近似多项式。将每个门替换为它的近似多项式，至多在比例为 $1/2^l$ 的输入上引入错误。因此，最终的近似多项式至多在 $S/2^l$ 个输入上出现错误，其中 S 是布尔线路的规模。（注意，各个门上的错误可能会相互影响。某个门上引入的错误可能被在更高层次上的另一个门上引入的错误抵消。因此，将合并界限应用到各个门的近似多项式出错概率上，所得的最终出错概率是保守的。）

第 2 步：假设多项式 f 与 MOD_3 在 $G'\subseteq\{0,1\}^n$ 中的所有输入上是一致的。如果 f 的
202 次数不超过 $\sqrt{n}$，则我们往证 $|G'|<\left(\frac{49}{50}\right)2^n$。

考虑变量代换 $y_i=1+x_i(\mathrm{mod}\ 3)$，于是，$0\to1$ 且 $1\to-1$。这一代换过程将多项式的定义域由 $\{0,1\}^n$ 变换为 $\{\pm1\}^n$。因此，在该变换下，f 变成了另一个多项式 $g(y_1,\cdots,y_n)$，它的次数仍为 $\sqrt{n}$。集合 G' 被变换成 $\{\pm1\}^n$ 的一个同样大小的子集 G，并且 g 和 MOD_3（的变换形式）在 G 上是一致的。

但是，不难发现，MOD_3 在变量代换之后变成了 $\prod_{i=1}^{n}y_i$。也就是说，$\sqrt{n}$ 次多项式 $g(y_1,\cdots,y_n)$ 与 $\prod_{i=1}^{n}y_i$ 在 G 上是一致的。这就蕴含着一些古怪，下面我们证明这样的 G 肯定很小。具体地，令 F_G 是所有函数 $S:G\to\{0,1,-1\}$ 构成的集合。显然，$|F_G|=3^{|G|}$。我们下面证明 $|F_G|\leqslant3^{\left(\frac{49}{50}\right)2^n}$，由此完成第 2 步。

引理 14.5 对于任意 $S\in F_G$，存在由形如 $G_I\prod_{i\in I}y_i$（其中 $|I|\leqslant\frac{n}{2}+\sqrt{n}$）的单项式之和构成的多项式 g_S 使得 $g_S(x)=S(x)$ 对所有 $x\in G$ 成立。

证明　令 $\hat{S}:\mathrm{GF}(3)^n\to\mathrm{GF}(3)$ 是任意一个与 S 在 G 上具有一致取值的函数。则 $\hat{S}$ 可以表示成所有变量 y_i 上的多项式。但是，我们感兴趣的仅仅是函数在输入 $(y_1,\cdots,y_n)\in\{-1,1\}^n$ 上的取值，其中 $y_i^2=1$。因此，我们可以要求多项式中的每个单项式 $\prod_{i\in I}y_i^{r_i}$ 均满足 $r_i\leqslant1$ 而不失一般性。这样，$\hat{S}$ 就是一个次数至多为 n 的多项式。现在，我们考虑其中次数 $|I|>n/2$ 的任意一个单项式 $\prod_{i\in I}y_i$，它可以被改写成

$$\prod_{i\in I}y_i=\prod_{i=1}^{n}y_i\prod_{i\notin I}y_i \tag{14.3}$$

也就是说，单项式 $\prod_{i\in I}y_i$ 与 $g(y_1,\cdots,y_n)\prod_{i\notin I}y_i$ 在 G 上的取值一致。因此，$\hat{S}$ 中的每个多项式都可以替换为一个次数不超过 $n/2+\sqrt{n}$ 的单项式。■

为了得出结论，我们给出引理 14.5 中所有可能的多项式 g_S 的个数的上界。这种多项式的个数是 3 的幂，而幂次数恰好是符合条件的单项式的总个数。但是，符合条件的所有

单项式的总个数是

$$|\{I\subseteq[n]:|I|\leqslant n/2+\sqrt{n}\}|=\sum_{i=0}^{n/2+\sqrt{n}}\binom{n}{i}$$

利用二项分布的尾界限(或直接计算)可以证明，上式小于$\left(\frac{49}{50}\right)2^n$。 ■

14.3　单调线路的下界

如果一个布尔线路仅含 AND 门和 OR 门而不含 NOT 门，则称该线路是单调的。单调线路只能计算如下定义的单调函数。

定义 14.6 对于 $x,y\in\{0,1\}^n$，如果在每个位置上 x 的二进制位等于 1 均蕴含 y 的二进制位也等于 1，则记 $x\leqslant y$。如果函数 $f:\{0,1\}^n\to\{0,1\}$ 使得 $f(x)\leqslant f(y)$ 对任意 $x\leqslant y$ 成立，则称 f 是单调函数。

单调函数 f 的另一个性质是，将输入中的一个二进制位由 0 改为 1 不会使得函数的值从 1 变为 0。

293

不难验证，单调线路计算的函数是单调函数，并且每个单调函数也可以用一个(充分大的)单调线路来计算。CLIUQE 是一个单调函数，因为在图中添加任意一条边不会使得现有的团消失。人们自然想到去证明 CLIQUE 不能被多项式规模的单调线路计算。莱兹波借夫率先证明了这一结论。安德列夫(Andreev)立刻得到了改进的结论，在此基础上阿龙(Alon)和波帕拉(Boppana)做出了进一步改进，最后形成了如下定理。

定理 14.7（CLIQUE 的单调线路下界[Razb85a, And85, AB87]）　设函数 $\mathrm{CLIQUE}_{k,n}:\{0,1\}^{\binom{n}{2}}\to\{0,1\}$ 在 n 顶点图 G 的邻接矩阵上输出 1 当且仅当 G 含有 k 顶点团。

存在 $\varepsilon>0$ 使得，对任意 $k\leqslant n^{1/4}$ 均不存在规模不超过 $2^{\varepsilon\sqrt{k}}$ 的单调线路来计算函数 $\mathrm{CLIQUE}_{k,n}$。

当然，人们还相信定理 14.7 对非单调线路也(粗略地)成立，亦即 $\mathbf{NP}\not\subset\mathbf{P}_{/\mathrm{poly}}$。事实上，在考虑单调线路时，人们最初相信单调线路复杂性与非单调线路复杂性之间存在某种联系。人们广泛认同的一个猜想曾经是：对于任意单调函数 f，它的单调线路复杂度与它的一般线路(非单调线路)复杂度多项式地相关。然而，这一猜想后来被莱兹波借夫[Razb85b]否证了。事实上，人们现在已经证明这两个复杂度之间的鸿沟是指数级的[Tar88]。

14.3.1　定理 14.7 的证明

团示性函数

为了让大家直观地理解定理 14.7 的正确性，我们先证明函数 $\mathrm{CLIQUE}_{k,n}$ 不能被具有指数规模的特殊单调线路计算(甚至近似计算)。对于任意 $S\subseteq[n]$，令 $\mathbf{C}_S$ 是 $\{0,1\}^{\binom{n}{2}}$ 上如下定义的函数，它在图 G 上输出 1 当且仅当 S 是 G 的团。我们称 $\mathbf{C}_S$ 是 S 的团示性函数。注意，$\mathrm{CLIQUE}_{k,n}=\bigvee_{S\subseteq[n],|S|=k}\mathbf{C}_S$。现在，我们证明：将 OR 操作应用到少于 $n^{\Omega(k)}$ 个团示性函数上不可能得出 $\mathrm{CLIQUE}_{k,n}$。

令 $\mathcal{Y}$ 是 n 顶点图 G 的如下概率分布：随机选取一个集合 $K\subseteq[n]$ 使得 $|K|=k$，输出一

个图 G 使得它在 K 上形成一个团并且不包含其他边。令 $\mathcal{N}$ 是 n 顶点图 G 的如下概率分布：随机选取函数 $c:[n]\to[k-1]$，在顶点 u 和 v 之间添加一条边当且仅当 $c(u)\neq c(v)$。显然，“$\mathrm{CLIQUE}_{k,n}(\mathcal{Y})=1$ 且 $\mathrm{CLIQUE}_{k,n}(\mathcal{N})=0$”的概率为 1。“表示 $\mathrm{CLIQUE}_{k,n}$ 时至少需要 $n^{\sqrt{k}/20}$ 个团示性函数”这一事实可以立刻从下面的引理得到。

引理 14.8 设 n 充分大，$k\leqslant n^{1/3}$，且 $S\subseteq[n]$。则要么 $\Pr[\mathbf{C}_S(\mathcal{N})=1]\geqslant 0.99$ 要么 $\Pr[\mathbf{C}_S(\mathcal{Y})=1]\leqslant n^{-\sqrt{k}/20}$。

证明　令 $\ell=\sqrt{k-1}/10$。如果 $|S|\leqslant\ell$，则由生日界限（参见例 A.4）可知，随机函数 $f:S\to[k-1]$ 产生的冲突次数的数学期望为 0.01。因此，由马尔可夫不等式可知，f 是单射函数的概率至少为 0.99。这就意味着，$\Pr[\mathbf{C}_S(\mathcal{N})=1]\geqslant 0.99$。

294

如果 $|S|>\ell$，则 $\Pr[\mathbf{C}_S(\mathcal{Y})=1]$ 等于“$S\subseteq K$ 对随机选取的大小为 k 的集合 K 成立”的概率。这一概率等于 $\binom{n-\ell}{k-\ell}/\binom{n}{k}$；由二项式系数公式可知，它小于 $\left(\frac{2k}{n}\right)^{\ell}\leqslant n^{-\ell/2}<n^{-\sqrt{k}/20}$（对充分大的 n）。 ■

用团示性函数实现近似

下面的引理与引理 14.8 一起就表明了定理 14.7 的正确性。

引理 14.9 设 C 是规模 $s<2^{\sqrt{k}/2}$ 的一个单调线路。那么，存在集合 $S_1,\ldots,S_m$（其中 $m<n^{\sqrt{k}/20}$）使得

$$\Pr_{G\in_R\mathcal{Y}}\left[\bigvee_i \mathbf{C}_{S_i}(G)\geqslant C(G)\right]>0.9 \tag{14.4}$$

$$\Pr_{G\in_R\mathcal{N}}\left[\bigvee_i \mathbf{C}_{S_i}(G)\leqslant C(G)\right]>0.9 \tag{14.5}$$

证明　令 $\ell=\sqrt{k}/10$，$p=10\sqrt{k}\log n$ 且 $m=(p-1)^{\ell}\ell!$。注意，$m\ll n^{\sqrt{k}/20}$。我们可以将线路 C 视为从 $\{0,1\}^{\binom{n}{2}}$ 到 $\{0,1\}$ 的一系列函数 $f_1,\ldots,f_s$，其中每个函数 f_u 既可能是两个函数 $f_{u'}$，$f_{u''}$（其中 $u',u''<k$）的 AND 或 OR，也可能是某个输入变量 $x_{\{v,w\}}$（其中 $v,w\in[n]$）的值（亦即 $f_u=C_{\{v,w\}}$）。C 所计算的函数是 f_s。我们将给出一系列函数 $\overline{f}_1,\ldots,\overline{f}_s$ 使得：(1)每个 $\overline{f}_u$ 都是 m 个满足 $|S_i|\leqslant\ell$ 的团示性函数 $\mathbf{C}_{S_i},\ldots,\mathbf{C}_{S_m}$ 的 OR；(2)每个 $\overline{f}_u$ 都是 f_u 的近似，也就是说，二者在服从概率分布 $\mathcal{Y}$ 和 $\mathcal{N}$ 的输入上将以很高的概率取相同的值。（这样，(14.4)式和(14.5)式就可以视为特殊的近似形式。）我们将满足条件(1)的任意函数 $\overline{f}_u$ 称为一个 (ℓ,m)-函数。

我们归纳地构造函数 $\overline{f}_1,\ldots,\overline{f}_s$。对于 $1\leqslant k\leqslant s$，如果 f_u 是一个输入变量，则我们令 $\overline{f}_u=f_u$。如果 $f_u=f_{u'}\vee f_{u''}$，则令 $\overline{f}_u=\overline{f}_{u'}\sqcup\overline{f}_{u''}$；如果 $f_u=f_{u'}\wedge f_{u''}$，则令 $\overline{f}_u=\overline{f}_{u'}\sqcap\overline{f}_{u''}$，其中 $\sqcup$ 和 $\sqcap$ 将在下面定义。我们将证明，对于任意 $f,g:\{0,1\}^{\binom{n}{2}}\to\{0,1\}$，(a)如果 f,g 是 (ℓ,m)-函数，则 $f\sqcup g$ 和 $f\sqcap g$ 也都是 (ℓ,m)-函数；(b)$\Pr_{G\in_R\mathcal{Y}}[f\sqcup g(G)<f\vee g(G)]<1/(10s)$ 且 $\Pr_{G\in_R\mathcal{N}}[f\sqcup g(G)>f\vee g(G)]<1/(10s)$（相应地，$\Pr_{G\in_R\mathcal{Y}}[f\sqcap g(G)<f\wedge g(G)]<1/(10s)$ 且 $\Pr_{G\in_R\mathcal{N}}[f\sqcap g(G)<f\wedge g(G)]<1/(10s)$）。利用合并界限可知，(b)中的概率不等式对 $\overline{f}_1,\ldots,\overline{f}_s$ 同时成立的概率为 0.9，进而证得引理。下面我们定义操作 $\sqcup$ 和 $\sqcap$，根据操作的定义即可得到条件(a)，而条件(b)则需要进行证明。

操作 $f\sqcup g$

设 f,g 是两个 (ℓ,m)-函数；也就是说，$f=\bigvee_{i=1}^{m}\mathbf{C}_{S_i}$ 而 $g=\bigvee_{i=1}^{m}\mathbf{C}_{T_i}$（如果 f,g 是少于 m

个的团示性函数的 OR，则可以在表达式中重复若干个集合使得团示性函数的个数恰好等于 m）。考虑函数 $h=\mathbf{C}_{Z_1}\vee\cdots\vee\mathbf{C}_{Z_m}$，其中 $Z_i=S_i$，$Z_{n+j}=T_j$ 对任意 $1\leqslant i,j\leqslant m$ 成立。函数 h 不是一个 (ℓ,m)-函数，因为它是 $2m$ 个团示性函数的 OR。我们如下地将它改造成一个 (ℓ,m)-函数：只要函数中的集合多于 m 个，则找出其中具有向阳花形状的 p 个集合 295
$Z_{i_1},\cdots,Z_{i_p}$；亦即，存在集合 $Z\subseteq[n]$ 使得 $Z_{i_j}\cap Z_{i_{j'}}=Z$ 对任意 $1\leqslant j,j'\leqslant p$ 成立。（将集合 Z 视为向阳花的中心，将集合 $Z_{i_1}\setminus Z,\cdots,Z_{i_p}\setminus Z$ 视为向阳花的花瓣，由形状的类比得到上述命名）。然后，将 h 中的函数 $\mathbf{C}_{Z_{i_1}},\cdots,\mathbf{C}_{Z_{i_p}}$ 替换为函数 $\mathbf{C}_Z$。重复上述过程，一旦替换过程得到一个 (ℓ,m)-函数 h'，则我们定义 $f\sqcup g$ 为 h'。定义过程不会遇到困难，因为下面的引理成立（其证明过程将延后给出）。

引理 14.10（向阳花引理[ER60]）　令 $\mathcal{Z}$ 是由不同集合构成的集族，其中每个集合的大小均不超过 ℓ。如果 $|\mathcal{Z}|>(p-1)^\ell\ell!$，则存在 p 个集合 $Z_1,\cdots,Z_p\in\mathcal{Z}$ 和一个集合 Z 使得 $Z_i\cap Z_j=Z$ 对任意 $1\leqslant i,j\leqslant p$ 成立。

操作 $f\sqcap g$

设 f,g 是两个 (ℓ,m)-函数；亦即，$f=\bigvee_{i=1}^{m}\mathbf{C}_{S_i}$ 而 $g=\bigvee_{j=1}^{m}\mathbf{C}_{T_j}$。令 h 是函数 $\bigvee_{1\leqslant i,j\leqslant m}\mathbf{C}_{S_i\cup T_j}$。我们在 h 上执行如下操作步骤：(1)丢弃满足 $|Z|>\ell$ 的 $\mathbf{C}_Z$；(2)利用上面的向阳花引理将 h 中的函数个数减小到 m。

证明条件(b)

为完成引理的证明，我们证明如下的四个不等式。

- $\Pr_{G\in_R\mathcal{Y}}[f\sqcup g(G)<f\vee g(G)]<1/(10s)$。

如果 $Z\subseteq Z_1,\cdots,Z_p$，则对于任意 i，$\mathbf{C}_{Z_i}(G)$ 均蕴含 $\mathbf{C}_Z(G)$。因此，操作 $f\sqcup g$ 不会引入任何"假阴性(false negative)"错误。

- $\Pr_{G\in_R\mathcal{N}}[f\sqcup g(G)>f\vee g(G)]<1/(10s)$。

只有在将向阳花 $Z_1,\cdots,Z_p$ 的团示性函数替换为太阳花中心 Z 的团示性函数时，我们才有可能为图 G 引入假阳性(false positive)错误。此时，$\mathbf{C}_{Z_i}(G)$ 为假对每个 i 均成立，但 $\mathbf{C}_Z(G)$ 为真。记住，随机选取 $G\in_R\mathcal{N}$ 时，我们是先选取随机函数 $c:[n]\to[k-1]$，再在满足 $c(u)\neq c(v)$ 的任意 u,v 之间添加一条边。于是，之所以会引入假阳性错误，是因为 c 限制在 Z 上构成单射（将这一随机事件记为 B），但限制在任意 Z_i 上却不是单射（将这些随机事件分别记为 $A_1,\cdots,A_p$）。我们下面将证明 B 与 $A_1,\cdots,A_p$ 同时发生的概率至多为 2^{-p}。根据 p 的选择可知，这一概率小于 $1/(10m^2s)$。由于利用向阳花引理削减函数的操作至多执行 m 次，故待证的不等式成立。

对任意 i，$\Pr[A_i|B]<1/2$。事实上，由于 $|Z_i|=\ell<\sqrt{k-1}/10$，生日界限（参见例 A.4）断言，$\Pr(A_i)<1/2$；在已知函数 c 未在 Z 上引起冲突的条件下，c 在 Z_i 上出现冲突的概率会更小，亦即 $\Pr[A_i|B]\leqslant\Pr(A_i)<1/2$。以 B 为条件时，$A_1,\cdots,A_p$ 是独立的随机事件，因为此时这些事件依赖于 c 在一些不相交集合上的取值。因此，$\Pr(A_1\wedge\cdots\wedge A_p\wedge B)\leqslant\Pr(A_1\wedge\cdots\wedge A_p|B)=\prod_{i=1}^{p}\Pr(A_i|B)\leqslant 2^{-p}$。

- $\Pr_{G\in_R\mathcal{Y}}[f\sqcap g\ (G)<f\wedge g(G)]<1/(10s)$。

根据分配律可知，$f\wedge g=\bigvee_{i,j}\mathbf{C}_{S_i}\wedge\mathbf{C}_{T_j}$。服从 $\mathcal{Y}$ 分布的图 G 在某个集合 K 上形成一个团。对于这种图，$\mathbf{C}_{S_i}\wedge\mathbf{C}_{T_j}$ 成立当且仅当 $S_i,T_j\subseteq K$；进而，$\mathbf{C}_{S_i}\wedge\mathbf{C}_{T_j}$ 成立当且仅当 $\mathbf{C}_{S_i\cup T_j}$

成立。我们丢弃满足$|Z|>\ell$的函数$\mathbf{C}_Z$时，可能会引入假阴性错误；但是，根据引理14.8可知，$\Pr[\mathbf{C}_Z(\mathcal{Y})=1]<n^{-\sqrt{n}/10}<1/(10sm^2)$对每个被丢弃$Z$均成立。由于我们至多丢弃了$m^2$个集合，故待证的不等式成立。

- $\Pr_{G\in_R\mathcal{N}}[f\sqcap g(G)>f\wedge g(G)]<1/(10s)$。

由于$\mathbf{C}_{S\cup T}$同时蕴含了$\mathbf{C}_S$和$\mathbf{C}_T$，从$f\wedge g$变形到$\bigvee_{i,j}\mathbf{C}_{S_i\cup T_j}$不会引入假阳性错误。从
296 OR操作中移除函数也不会引入假阳性错误。因此，唯一可能引入假阳性的是"将向阳花中的团示性函数替换为太阳花中心的团示性函数"这一过程。采用同$\sqcup$操作一样的方法，我们可以给出假阳性概率的上界。■

向阳花引理的证明

证明过程是对ℓ用归纳法。$\ell=1$的情形是平凡的，因为大小为1的不同集合必然互不相交(因而形成一个以$Z=\varnothing$为中心的向阳花)。对于$\ell>1$的情形，令$\mathcal{M}$是Z中互不相交的集合构成的极大子集族。我们可以假设$|\mathcal{M}|<p$；否则，$\mathcal{M}$本身就形成了一朵满足引理要求的向阳花。由$\mathcal{M}$的极大性可知，对任意$Z\in\mathcal{Z}$，均存在$x\in\bigcup\mathcal{M}=\bigcup_{M\in\mathcal{M}}M$使得$x\in Z$。由于$|\bigcup\mathcal{M}|\leqslant(p-1)\ell$，由平均值方法可知，存在$x\in\bigcup\mathcal{M}$使得$x$至少出现在$\mathcal{Z}$中$\frac{1}{\ell(p-1)}$比例的集合内。令$Z_1,\cdots,Z_t$是$\mathcal{Z}$中包含$x$的所有集合，注意$t>(p-1)^{\ell-1}(\ell-1)!$。于是，由归纳假设可知，在大小为$\ell-1$的集合$Z_1\setminus\{x\},\cdots,Z_t\setminus\{x\}$中存在$p$个集合形成向阳花。将$x$放回这$p$个集合中就得到原集族中的一朵向阳花。注意，引理的叙述和证明都没有提及$\mathcal{Z}$中集合的全域。■

14.4　线路复杂性的前沿

本节概述线路复杂性的"前沿"，亦即，指明什么样的线路下界是可以被证明的，什么样的线路下界是还不能被证明的。在此过程中，我们将再次遇到多方通信，因为在证明某些新的线路下界时多方通信可能会十分有用。

14.4.1　用对角线方法证明线路下界

我们在前面曾指出，目前**NP**问题的线路规模的最佳下界是$5n-o(n)$。对于**PH**，更佳的下界已经被证明。事实上，习题6.5和习题6.6曾要求用对角线方法证明，对任意$k>0$，计算**PH**中(事实上是$\boldsymbol{\Sigma}_2$中)的某些语言需要用规模为$\Omega(n^k)$的线路。我们可以想象的是，**PH**"之上"[⊖]的其他复杂性类应该会包含更难的语言。于是，一个自然的待决问题是：

前沿1：NEXP中含有需要用超多项式规模的线路才能计算的语言吗？

如果考虑**NEXP**之上稍微更靠上层的复杂性类，则我们确实可以证明得出超多项式下界。事实上，我们知道$\mathbf{MA}_{\mathbf{EXP}}\subseteq\mathbf{P}_{/\mathrm{poly}}$，其中$\mathbf{MA}_{\mathbf{EXP}}$是具有全能型证明者和指数时间的概率型验证者的一回合交互式证明系统所接受的所有语言构成的复杂性类。(它是8.2节所定义的**MA**的指数时间形式)。这是由于，如果$\mathbf{MA}_{\mathbf{EXP}}\subseteq\mathbf{P}_{/\mathrm{poly}}$，则特别地将会有$\mathbf{PSPACE}\subseteq\mathbf{P}_{/\mathrm{poly}}$。然而，根据$\mathbf{IP}=\mathbf{PSPACE}$(定理8.19)可知，此时将有$\mathbf{PSPACE}=\mathbf{MA}$(交互式证明系统中的证明者可以在1个回合的通信中将计算证明者策略的线路发送给验证者)。再简单

⊖ 所有复杂性类可以视为一个层状结构，多项式分层**PH**位于靠近底层的位置。——译者注

地应用填充技术就可以证明，这意味着 $\mathbf{MA}_{\mathbf{EXP}}$ 等于所有指数空间语言构成的类。运用对角线方法可以直接证明，指数空间语言类不含于 $\mathbf{P}_{/poly}$，由此得出矛盾。有趣的是，$\mathbf{MA}_{\mathbf{EXP}} \nsubseteq \mathbf{P}_{/poly}$ 表明的线路下界不是相对性的，因为存在一个神喻使得 $\mathbf{MA}_{\mathbf{NEXP}} \subseteq \mathbf{P}_{/poly}$[BFT98]。当然，我们在上述证明过程中采用的结论 **IP**＝**PSPACE** 也不是相对性的。 297

14.4.2　ACC Vs P 的研究现状

PARITY 不属于 $\mathbf{AC}^0$，这一结论区分了复杂性类 **NC**1 和 $\mathbf{AC}^0$。因此，一个符合逻辑的想法是，将 **ACC**0 与 **NC**1 也区分开。稍退一步讲，我们希望找出 **P** 或 **NP** 中的某个函数使得它不属于 **ACC**0。

即使允许线路中使用两种类型的模门（不妨设 MOD_2 和 MOD_3），则莱兹波偌夫-斯莫伦斯基方法也将失效。事实上，如果我们允许深度受限的线路使用算术模门 MOD_q，其中 q 不是素数（更确切地说，q 是素数的方幂），则我们也将达到我们知识的极限。（习题 14.6 要求读者解释，当模门中的模是合数时，为什么定理 14.4 的证明将不再有效。）例如，根据我们现有的知识，CLIQUE 可以用仅含 MOD_6 门的具有常数深度和线性规模的线路来计算。这种现象的症结可能是，只要 m 是合数，则模 m 的次数较低的多项式也具有很强的表达能力[BBR92]。

前沿 2：证明 CLIQUE 不属于 **ACC**0(6)。

或者稍退一步：

前沿 2.1：从 **NEXP** 中找出一个不属于 **ACC**0(6)的语言。

值得注意的是，本章迄今为止都只讨论了非一致线路（定理 14.4 所适用的场合就是非一致线路）。在一致线路上，人们证明了更佳的下界。艾伦德（Allender）和戈尔（Gore）[AG94]证明了，积和式的判定形式（进而积和式本身也）需要用指数规模的“确定型对数时间一致的”**ACC**0 线路才能被计算。（线路族 $\{C_n\}$ 称为是确定型对数时间一致的（Dlogtime uniform），如果存在确定型图灵机 M 使得，给定整数 n 和两个门 g，h 之后，M 能够在 $O(\log n)$ 时间内判定门 g 和 h 的类型以及在 C_n 中 g 是否为 h 的父门。）

回到非一致线路族 **ACC**0 上来，我们给出 **ACC**0 线路的另一种表示方法，它可能对进一步研究线路下界十分有用。对称门是一种特殊的门，它的输出仅依赖于输入中等于 1 的二进制位的个数。例如，多数门（majority gate）和模门（mod gate）都是对称门。姚期智证明了，所有 **ACC**0 线路都可以等价地简化为深度为 2 且输出门是对称门的线路（参见图 14-3）。贝格尔（Beigel）和 Tarui 后来改进了姚期智的结果。



图 14-3　定理 14.11 中深度为 2 且具有对称输出门的线路

298

定理 14.11（[Yao90，BT91]）　如果 $f \in \mathbf{ACC}0$，则 f 可以用深度为 2 的如下线路 C 来计算，C 的输出层是具有伪多项式扇入度（亦即 $2^{\log^k n}$）的对称门，输入层是具有对数多项式扇入度的 ∧ 门。

我们将在 14.5.1 节中再研究这个定理。

14.4.3　具有对数深度的线性线路

当线路的扇入度受限后，我们需要允许线路具有非常数的深度(即 $\Omega(\log n)$)，这样线路的输出才有可能依赖于输入中的所有二进制位。认识到这一点之后，最简单的有意义的扇入度受限的线路族就应该是规模为 $O(n)$ 且深度为 $O(\log n)$ 的线路。

前沿 3：明确给出一个 n 位布尔函数使得它不能被规模为 $O(n)$ 且深度为 $O(\log n)$ 的线路计算。

次前沿：找出从 $\{0,1\}^n \to \{0,1\}^n$ 的一个映射(而不是布尔函数)使得它不能被规模为 $O(n)$ 且深度为 $O(\log n)$ 的线路计算。

(注意，用计数论证法容易证明，存在需要超多项式规模的线路才能计算的 n 位函数，进而，要计算这种函数，当线路的扇入度受限后，线路的深度必然是超对数的，参见第 6 章的习题。因此，人们希望用一个明确的函数(如 CLIQUE)来证明这一结论。)

瓦利安特(Valiant)曾在二十世纪 70 年代考虑过这个问题。他从众多的候选函数出发，经过筛选最终提出去证明：一种称为超级集中器(superconcentrator)的图需要用超线性规模的线路才能被计算。最终，他只证明了超级集中器的存在性，却未能证明这种图需要超线性规模线路。但是，瓦利安特开展的研究得到了一个副产品，这就是下面用于削减线路深度的重要引理。

引理 14.12 ([Val75a])　在具有 m 条边且深度为 d 的任意无环有向图中，存在 $km/\lceil \log d \rceil$ 条边构成的集合 S 使得删除 S 后图的深度将至多为 $d/2^{k-1}$。

证明概要　将图排成 d 层，使得：如果 $\overrightarrow{uv}$ 是一条边，则 u 所在的层比 v 所在的层更低。令 $\ell=\lceil \log d \rceil$，我们将每个层都表示为一个长度为 ℓ 的位串。然后用 $i\in[\ell]$ 来标记每条边 $\overrightarrow{uv}$，其中 i 是 u 所在层的位串与 v 所在层的位串中首个不同二进制位出现的位置。令 I 是使用频率最低的 k 个标签构成的集合，S 是由 I 中标签标记的所有边构成的集合。显然，$|S|\leqslant km/\ell$。而且，还可以证明，每条长度大于 $2^{\ell-k}\leqslant d/2^{k-1}$ 的路径所使用的标签将多于 $\ell-k$ 个(进而必然包含 S 中的一条边)。习题 14.10 要求读者完成引理的证明。 ■

引理 14.12 可以如下应用。假设我们有一个规模为 $O(n)$ 且深度为 $c\log n$ 的线路 C，它的 n 个输入是 $\langle x_1, \cdots, x_n\rangle$ 并且它的 n 个输出是 $\langle y_1, \cdots, y_n\rangle$，假设 $2^c=c/\varepsilon$，其中 $\varepsilon>0$ 是一个任意小的数。根据引理，我们可以在 C 中找出 $O(n/\log\log n)$ 条边，删除这些边之后，我们将得到一个深度至多为 $\varepsilon\log n$ 的线路。但另一方面，由于 C 的扇入度是受限的，我们知道每个输出 y_i 必然至多与 $2^{\varepsilon\log n}=n^{\varepsilon}$ 个输入相关联，因此，C 的每个输出 y_i 必然由 n^{ε} 个输入以及那些被删除的线上的值完全确定。也就是说，被删除的线将通过某种方式使得 $y_1, y_2, \cdots, y_n$ 的真值表可以被"压缩"为某种特殊的形式。人们并不指望这种压缩对某种合理的复杂函数也成立。但出人意料的是，没有人能够给出一个明确的函数使得这种压缩不可行。

299

14.4.4　线路图

正如线路可以用于研究图灵机的时间复杂性一样，线路图(branching program)是用于研究空间复杂性的组合学工具。n 个输入变量 $x_1, \cdots, x_n$ 上的线路图是一个无环有向图，它的每个出度不为 0 的顶点都由某个变量 x_i 标记；它有两个出度为 0 的顶点，这两个顶点分别用输出值 ACCEPT 或 REJECT 来标记；图中的每条边用 0 或 1 来标记；图中的一个

顶点被指定为开始顶点。给定输入变量 x_1，…，x_n 的一组取值，则在线路图中对应一条从开始顶点出发到达某个输出顶点的路径。在每个步骤中，如果当前顶点被标记为 x_i，则从该顶点出发沿着 x_i 的取值所标记的边前进。如果线路图中每个非输出顶点都恰有一条出边被标记为 0 并且恰有一条出边标记为 1，则称该线路图是确定型的；否则，称该线路图是非确定型的。线路图的规模是图中顶点的个数。语言的线路图复杂度的定义与线路复杂度的定义类似。有时，人们还要求线路图是分层的，也就是说，线路图的所有顶点被排列成一系列不同的层，而线路图的边只能从一个层指向另一个层。线路图的宽度指的是最大层中顶点的个数。

定理 14.13　如果 $S(n)\geqslant\log n$ 且 $L\in\mathbf{SPACE}(s(n))$，则 L 存在一个规模至多为 $c^{S(n)}$ 的线路图，其中 $c>1$ 是一个常数。

证明　证明过程本质上是模仿定理 4.2 证明 $\mathbf{SPACE}(S(n))\subseteq\mathbf{DTIME}(2^{O(S(n))})$ 的过程。线路图中的每个顶点对应于空间受限图灵机的一个格局，如果顶点对应的格局表明图灵机在相应的计算步骤中读取了第 i 个输入位，则将该顶点标记为 x_i。■

对于非确定型图灵机和非确定型线路图复杂性，类似结论也成立。

前沿 4：找出 **P**（甚至 **NP**）中的一个语言，使得它需要用规模大于 $n^{1+\varepsilon}$ 的线路图才能被计算，其中 $\varepsilon>0$ 是某个常数。

有证据表明，线路图的能力超乎人们的想象。例如，具有常数宽度的线路图不禁让人联想到具有 $O(1)$ 个存储单元的图灵机，它似乎很弱。因此，如下结论颇出人意料。

定理 14.14　（巴林顿（Barrington）[Bar86]）　一个语言存在宽度为 5 的多项式规模线路图当且仅当它属于 **NC1**。

14.5　通信复杂性方法

本节勾勒出一种具体方法（或者说一种机制）。这种方法是对前一章介绍的通信复杂性的推广，利用该方法得出的更佳下界将有望解决前面提到的某些前沿问题。我们将主要运用（13.3 节中定义的"脑门记数"模型下的）多方通信复杂性，但 14.5.4 节将用到计算关系所需的通信复杂性。

300

14.5.1　与 ACC0 线路之间的联系

假设 $f(x_1,\cdots,x_n)$ 有一个深度为 2 的如下线路，线路的输出门是一个扇入度为 N 的对称门，并且线路的输入层是扇入度为 $k-1$ 的 ∧ 门（参见图 14-4）。莱兹波诺夫（Razborov）和维格德尔森（Wigderson）[RW93] 观察到，此时 f 的 k 方通信复杂度至多为 $k\log N$。为看清这一点，首先，将每个 ∧ 门指派给 k 个参与方中的一方。具体指派过程如下：将每个输入位告诉除一个参与方之外的其他所有参与方，因此，对每个 ∧ 门，它的所有输入位至少被一个参与方全部知晓，将每个 ∧ 门指派给知晓其全部输入位的编号最小的参与方。然后，所有参与方广播指派给自己的所有 ∧ 门中输出等于 1 的 ∧ 门的个数。由于表示这一数量至多需要 $\log N$ 个位，因此



图 14-4　如果 f 可由本图中的线路来计算，则 f 存在一个复杂度为 $k\log N$ 的 k 方通信协议

f 的 k 方通信复杂度至多为 $k\log N$。

我们希望利用与通信复杂性之间的上述联系和定理 14.11 来找出 **ACC0** 线路的复杂性下界。例如，由定理 13.24 可知"有一个明确的 n 位函数，其 k 方通信复杂度为 $\Omega(n/4^k)$"，因此，任何具有上述形式的底层扇入度为 $k-1<\log n/4$ 的深度为 2 的多项式规模线路都无法计算该函数。但是，这还不足以得出 **ACC0** 线路的下界。这是由于，要应用定理 14.11，我们还必须说明 k 不是(n 的)多项式。尽管如此，CLIQUE 函数的 k 方通信复杂度下界 $\Omega(n/\mathrm{poly}(k))$ 将有望用于解决前面指出的前沿 2。

14.5.2　与线性规模对数深度的线路之间的联系

假设函数 $f:\{0,1\}^*\times\{0,1\}^{\log n}\to\{0,1\}^n$ 可以用一个扇入度受限的具有对数深度的线性规模线路来计算。如果 $f(x,j,i)$ 表示 $f(x,j)$ 的第 i 个二进制位，则瓦利安特引理(Valiant Lemma，即引理 14.12)表明，$f(x,j,i)$ 有一个同时三方协议(也就是说，三方都只发送一条信息并且各方(非自适应地)同时将自己要发送的信息写在黑板上)使得

301

- 参与方 (x,j) 发送 $O(n/\log\log n)$ 个位；
- 参与方 (x,i) 发送 n^{ε} 个位；
- 参与方 (i,j) 发送 $O(\log n)$ 个位。

因此，如果我们能证明某个函数不存在这种协议，则我们就得到了该函数在"扇入度受限的具有对数深度的线性规模线路"上的复杂性下界。例如，即使对于简单函数 $f(x,j,i)=x_{j\oplus i}$(其中 $j\oplus i$ 是按位异或操作)，人们也还未能找到前面所说的这种协议。因此，这个函数可能不能用扇入度受限的具有对数深度的线性规模线路来计算。

14.5.3　与线路图之间的联系

钱德拉(Chandra)，弗斯特(Furst)和利普顿(Lipton)[CFL83]发明多方通信复杂性(至少这里所讨论的"脑门记数"模型)就是为了证明线路图下界，特别是为了证明 14.4.4 节讨论的具有常数宽度的线路图的下界。

14.5.4　卡奇梅尔-维格德尔森通信游戏与深度下界

PARITY 不属于 $\mathbf{AC}^0$，这一结论区分了复杂性类 **NC1** 和 $\mathbf{AC}^0$。因此，接下来人们希望再区分 **NC2** 和 **NC1**(当然，这要先搁置区分 **ACC0** 与 **NC1** 的任务)。卡奇梅尔(Karchmer)和维格德尔森(Wigderson)说明了如何用通信复杂性来证明计算某个函数的线路的最小深度的下界。他们证明了下面关于单调线路的结果，我们略去其证明。

定理 14.15　([KW88])　*用深度为 $O(\log n)$ 的扇入度受限的单调线路来判定一个图是否存在完美匹配是不可能的。*

但是，本小节给出证明上述结果时采用的一种通信游戏，亦即基本的 Karchmer-Wigderson 游戏。我们之所有这么做，是由于这种游戏也与非单调线路相关。对于函数 $f:\{0,1\}^n\to\{0,1\}$，Karchmer-Wigderson 游戏的定义如下。

游戏有两个参与方，分别记为 **ZERO** *和* **ONE**。**ZERO** *持有的输入 x 满足 $f(x)=0$，而* **ONE** *持有的输入 y 满足 $f(y)=1$。双方对输入中的位进行通信，直到双方找到 $i\in\{1,2,\cdots,n\}$ 使得 $x_i\neq y_i$。*

游戏所用的通信机制类似于 13 章中的定义。参与双方在获得输入之前已经达成了一

致的通信协议。注意，这里的通信协议和第 13 章中的通信协议之间的关键区别是：通信协议执行的结果不再是单个的二进制位，并且最终结果也不唯一(答案的个数等于 x，y 在对应位置上不相同的二进制位的个数)。有时，这种通信协议也称为计算关系的通信协议。在 Karchmer-Wigderson 游戏中，我们考虑的是由满足 $f(x)=0$，$f(y)=1$ 且 $x_i \neq y_i$ 的所有三元组(x, y, i)构成的关系。

我们将函数 f 在 Karchmer-Wigderson 游戏中的通信复杂度记为 $C_{KW}(f)$。也就是说，$C_{KW}(f)$是在所有 $x \in f^{-1}(0)$，$y \in f^{-1}(1)$上计算游戏的答案时所要交换的二进制位的最大个数。下面的定理表明，$C_{KW}(f)$具有另一种出人意料的性质。该定理假设线路都不包含 NOT 门，之所以可以做到这一点，是因为线路中的 NOT 门都可以通过德摩根定律下推到 302
输入中。换句话说，线路的输入可以视为$(x_1, x_2, \cdots, x_n, \overline{x_1}, \overline{x_2}, \cdots, \overline{x_n})$。定理还假设，所有 AND 门和 OR 门的扇入度都等于 2。所有这些假设都不重要，它们不影响定理的正确性。

定理 14.16 ([KW88])　$C_{KW}(f)$等于计算 f 的所有线路的最小深度。

证明　首先，我们证明：如果 f 可以用一个深度为 K 的线路 C 来计算，则 $C_{KW}(f) \leqslant K$。游戏中，每方都有线路 C 的一个拷贝，并用它来计算函数 f 在自己持有的输入上的取值。当然，参与方 **ZERO** 在输入上计算的结果是 0，而参与方 **ONE** 在输入上计算的结果则是 1。假设线路最顶层的输出门是 OR 门。于是，在输出门的两条输入线中，参与方 **ONE** 至少在一条线上计算得到 1。因此，在第一回合中，参与方 **ONE** 发送一个二进制位，以表明他在输出门的哪条线上计算得到了 1。注意，参与方 **ZERO** 在这条线上得到的计算结果是 0。在接下来的通信回合中，参与游戏的双方都关注这条线上另一端的门。(如果线路最顶层的输出门是 AND 门，则在第一个回合中先由参与方 **ZERO** 发送信息，以表明他在哪条线上计算得到 0。此时，参与方 **ONE** 在这条线上计算得到 1。)重复上述过程，则游戏双方将深入线路的底层。在此过程中，始终维护如下的循环不变量：参与方 **ONE** 在当前的门上计算得到 1 而参与方 **ZERO** 则在当前的门上计算得到 0。最后，至多经过 K 步之后，双方将到达输入中的一个二进制位。根据维护的循环不变量，参与方 **ONE** 所到达的二进制位必然等于 1，而参与方 **ZERO** 所到达的二进制位必然等于 0。由此得到的二进制位在输入中的位置 i 必然是游戏的有效答案。

对于逆命题，我们需要证明：如果 $C_{KW}(f)=K$，则必然存在一个深度为 K 的线路来计算 f。我们证明一个更一般的结论。对于任意两个非空集合 $A \subseteq f^{-1}(0)$，$B \subseteq f^{-1}(1)$，令 $C_{KW}(A, B)$是在 $x \in A$ 且 $y \in B$ 的条件下 Karchmer-Wigderson 游戏的通信复杂度。我们往证，存在一个深度为 $C_{KW}(A, B)$的线路使得，它在属于 A 的每个输入上输出 0 而在属于 B 的每个输入上输出 1。这种线路称为集合 A，B 的区分器(distinguisher)。证明过程是对 $K=C_{KW}(A, B)$做数学归纳。$K=0$ 时的基本情况是平凡的。原因在于，由于参与游戏的双方不需要通信就一致地得到了游戏的答案 i。因此 $x_i \neq y_i$ 对任意 $x \in A$，$y \in B$ 都成立，这意味着，要么(a)$x_i=0$ 对任意 $x \in A$ 成立且 $y_i=1$ 对任意 $y \in B$ 成立；要么(b)$x_i=1$ 对任意 $x \in A$ 成立且 $y_i=0$ 对任意 $y \in B$ 成立。在情况(a)下，我们用深度为 0 的线路 x_i，而在情况(b)下，则用线路 $\overline{x_i}$，这样就得到了 A，B 的区分器。

在归纳步骤中，假设 $C_{KW}(A, B)=K$，并且通信协议中参与方 **ZERO** 首先发送消息。这样，A 是两个不相交的子集 A_0，A_1 的并集，其中 A_b 中的每个元素会使参与方 **ZERO** 发送二进制位 b。于是，$C_{KW}(A_b, B) \leqslant K-1$ 对每个 b 成立。由归纳假设可知，集合 A_b，B 存在一个深度至多为 $K-1$ 的区分器 C_b。我们断言，线路 $C_0 \wedge C_1$(注意，该线路的深度至

多为 K)可以用来区分 A，B。原因如下：一方面，$C_0(y)=C_1(y)=1$ 对任意 $y\in B$ 都成立；另一方面，由于 $C_0(x)=0$ 对任意 $x\in A_0$ 成立，故 $C_0(x)\wedge C_1(x)=0$ 对任意 $x\in A$ 都成立。■

由此得出下面的研究前沿。

前沿 5：给出 **P**(甚至 **NEXP**!)中的一个函数 f 使得 $C_{KW}(f)=\Omega(\log n\log\log n)$。

卡奇梅尔(Karchmer)，莱斯(Raz)和维格德尔森(Wigderson)[KRW95]给出了一个可能满足要求的候选函数。他们的结果用到如下两个事实：其一，k 位函数 f 的真值表大小为 2^k；其二，大多数 k 位函数都是难计算的(亦即，计算函数的线路的规模为 $\Omega(2^k/k)$，或者深度为 $\Omega(k)$，等等)。他们在定义函数时，假设 n 位输入中的一部分二进制位是一个难以计算的函数的编码，而这个函数将以"柯"形方式作用到输入的其余部分上。

303

对于任意函数 $g:\{0,1\}^k\to\{0,1\}$ 和 $s\geqslant 1$，如下地定义函数 $g^{\oplus s}:\{0,1\}^{k^s}\to\{0,1\}$。如果 $s=1$，则 $g^{\oplus s}=g$。否则，将输入 $x\in\{0,1\}^{k^s}$ 表示为 $x_1x_2\cdots x_k$，其中 $x_i\in\{0,1\}^{k^{s-1}}$，并定义

$$g^{\oplus s}(x_1x_2\cdots x_k)=g(g^{\oplus(s-1)}(x_1)g^{\oplus(s-1)}(x_2)\cdots g^{\oplus(s-1)}(x_k))$$

显然，如果 g 可以用深度为 d 的线路来计算，则 $g^{\oplus s}$ 可以用深度为 sd 的线路来计算。然而，对于任意选取的函数 g，似乎很难再减小计算 $g^{\oplus s}$ 的线路的深度。

现在，我们描述卡奇梅尔，莱斯和维格德尔森给出的候选函数 $f:\{0,1\}^n\to\{0,1\}$。令 $k=\left\lfloor\log\frac{n}{2}\right\rfloor$，且 s 是满足 $k^s\leqslant n/2$ 的最大整数，因此 $s=\Theta\left(\frac{\log n}{\log\log n}\right)$。对于任意的 n 位输入 x，令 g_x 是"真值表恰为 x 的前 2^k 个位"的函数，$x|_z$ 是 x 的后 k^s 个位构成的位串，则

$$f(x)=g_x^{\oplus s}(x|_z)$$

根据前面的直观解释，如果 x 的前 2^k 个位确实表示一个难以计算的函数(在许多输入上情况也确实如此)，则 $g_x^{\oplus s}(x|_z)$ 应该需要深度为 $\Omega(sk)=\Omega\left(\frac{\log^2 n}{\log\log n}\right)$ 的线路才能被计算。当然，想要准确地证明这一结论却似乎非常困难。

上述构造过程本质上提出了如下问题：问题的 s 个实例的计算复杂度是否等于问题的一个实例的计算复杂度的 s 倍？这类计算复杂性问题称为**直和问题**。人们在许多计算模型上研究了类似的直和问题，并且在某些情况下还证明了一些有违直觉的结论。例如，利用计数论证法可以证明，在 $\{0,1\}$ 上存在一个 $n\times n$ 的矩阵 A 使得，计算线性函数 $v\mapsto Av$(其中 $v\in\{0,1\}^n$)的最小线路的深度为 $\Omega(n^2)$。但是，利用快速矩阵乘法算法[Str69](目前最好的时间复杂度为 $O(n^{2.38})$[CW90])，在 n 个独立实例 v_1，…，v_n 上计算这个线性函数所需的总的计算步骤却远小于 n^3。

本章学习内容

- 尽管人们相信许多明确的函数都具有指数级的线路复杂度，但目前对任意这样的函数都未证得它的线路复杂性具有超线性的下界。
- 相比之下，在限定的线路族(如深度为常数的线路和单调线路)上，人们得到了一些非常不平凡的下界。
- 线路复杂性当前的研究中有几个前沿问题，对其中任何一个问题的研究获得突破都将对深入理解计算的能力产生重要影响。

本章注记和历史

1949 年，香农(Shannon)定义了线路复杂性，包括单调线路复杂性。从二十世纪 50 年代开始，线路复杂性在俄罗斯得到了研究(Trakhtenbrot[Tra84]引用了当时的部分文献)。赛韦吉(Savage)[Sav72]率先注意到"图灵机判定语言所需的时间复杂性"与线路复杂性之间的紧密联系，并建议用线路下界来区分不同的复杂性类。随后，在二十世纪 80 年代产生一批这样的研究结果。例如，**P** 和 $\mathbf{AC}^0$ 可以用线路复杂性区分[FSS81，Ajt83]；莱兹波偌夫(Razborov)用线路复杂性区分了单调 **NP** 和单调 $\mathbf{P}_{poly}$[Razb85a]，这一结果曾让人们憧憬 **P** 与 **NP** 这一问题的彻底解决已经指日可待。这一憧憬被莱兹波偌夫[Razb89]自己粉碎了，因为他证明了近似方法不可能应用到非单调线路上。后来，莱兹波偌夫和卢吉奇(Rudich)[RR94]又形式地定义了自然证明(Natural Proof)，并以此证明了当时所做的所有努力都不能解决 **P** 与 **NP** 问题(参见第 23 章)。

本章 14.2 节和 14.3 节改写自波普潘纳(Boppana)和西普赛尔(Sipser)对线路复杂性的精彩综述[BS90]。在 15 年之后的今天，该综述仍未过时。但它未讨论代数线路的下界，关于这方面的最新研究结果请参见[Raz04]。

哈斯塔德(Håstad)开关引理[Hås86]是论文[FSS81，Ajt83，Yao85]中所述结果的加强形式。莱兹波偌夫和斯莫伦斯基(Smolensky)的多项式近似方法源自[Razb87]并在[Smo87]中得到加强。瓦利安特(Valiant)在超线性线路下界上得到的观察结果源自 1975 年的一篇论文[Val75]和一篇未发表的手稿——当代的研究者们应该就这么长时间以来仍未对这一基本问题取得任何研究进展感到羞愧。

一般线路的下界 $5n-o(n)$ 是岩间(Iwama)和守角(Morizumi)[⊖]证明的，莱基(Lachish)和莱斯(Raz)后来将结果显著地改进为 $4.5n-o(n)$。关于这两个结果的完整论述，请参阅[ILMR05]。

研究者们对线路的一些直观认识可能是完全错误的，巴林顿定理(Barrington Theorem，定理 14.14)就是一个很好的例子。巴林顿定理提供了计算 **NC1** 函数的一种超级简单的方法。事实证明，巴林顿定理对密码学研究产生了重要影响(例如，参见[GMW87，Kil88，AIK04])。

习题

14.1　假设 f 可以用一个深度为 d 且规模为 S 的 $\mathbf{AC}^0$ 线路 C 来计算。证明：f 也可以用一个深度为 d 且规模小于 $10S$ 的如下 $\mathbf{AC}^0$ 线路 C' 来计算：C' 不包含 NOT 门，但它的输入多出 n 个二进制位且这 n 个位分别是原始输入中 n 个位的否定。

14.2　假设 f 可以用一个深度为 d 且规模为 S 的 $\mathbf{AC}^0$ 线路 C 来计算。证明：f 也可以用一个每个逻辑门扇出度均为 1 的深度为 d 且规模小于 $(10S)^d$ 的 $\mathbf{AC}^0$ 线路 C' 来计算。

14.3　证明：如果布尔函数 f 的所有最大项的大小都至多为 s，则 f 可以表示成一个 s-CNF范式。

14.4　证明：$\binom{n}{t+k} \leqslant \binom{n}{t}\left(\frac{e(n-t)}{n}\right)^k$ 对任意 $t>n/2$ 成立。利用这一结论完成 14.1.2 节中引理 14.2 的证明。

⊖ "岩间"和"守角"都是日本姓氏。——译者注

14.5　证明：$\mathbf{ACC}^0 \subseteq \mathbf{NC}^1$。

14.6　找出原因来解释，为什么 m 是合数时莱兹彼佐夫-斯莫伦斯基引理不适用于含 MOD_m 门的线路。

305

14.7　证明：如果 q 是素数，则将 n 个变量 $x_1, \cdots, x_n$ 的 OR 表示为 $GF(q)$ 上的多项式时恰好需要 n 次多项式。

14.8　Karchmer-Wigderson 游戏可以用来证明上界，而不是只能用于证明下界。利用该游戏证明 PARITY 和 MAJORITY 属于 $\mathbf{NC}^1$。

14.9　证明：对于任意常数 c，如果一个语言可以用宽度为 c 的多项式规模线路图来计算，则该语言属于 $\mathbf{NC}^1$。

306

14.10　给出瓦利安特引理(引理 14.12)的完整证明。

证明复杂性

出人意料的是，鸽巢原理的证明复杂性本质上依赖于鸽子的数量。

——亚历山大·莱兹波佐夫(Alexander Razborov)，2001

NP 的定义旨在刻画如下的现象：如果一个论断(如"给定的布尔公式是可满足的")是真的，则存在这一论断的短证明。根据一般情况下究竟何种形式的论断(再如"给定的布尔公式是不可满足的")不存在短证明，我们还引出了 **NP**≠**coNP** 这一猜想。本章将更加仔细地研究上述现象，特别地，我们主要研究"短证明的存在性不显然"的各种情形。

15.1 节先用几个例子来说明我们的研究动机。15.2 节通过命题证明这个简单的例子形式地给出证明系统的概念。我们将会用两种方法证明分解证明系统的指数下界：这两种方法是两个简单的例子，很好地展示了关于证明复杂性的两种重要技术。15.3 节概述了人们研究过的其他证明系统，这些系统的复杂性下界都已经得到了证明。15.4 节站在数学的角度对"证明复杂性能否解释 **P** vs **NP** 这一问题的难度"进行思考。与证明复杂性相关的一个有意义的问题是，在短证明存在的假设条件下如何才能找出短证明。本章将忽略对该问题的讨论，本章注记会简要地提及它。

15.1 几个例子

我们先给出几个例子，其中多数例子在计算复杂性这一概念诞生之前就被人们研究过。考虑如下几个计算任务。

1. **线性不等式组的无解性**。给定不等式组

$$\begin{aligned}\langle \boldsymbol{a}_1, \boldsymbol{x}\rangle &\leqslant b_1\\ \langle \boldsymbol{a}_2, \boldsymbol{x}\rangle &\leqslant b_2\\ &\vdots\\ \langle \boldsymbol{a}_m, \boldsymbol{x}\rangle &\leqslant b_m\end{aligned}$$

其中 $\boldsymbol{a}_i \in \mathbf{R}^n$ 且 $b_i \in \mathbf{R}$ 对每个 i 都成立。验证不存在非负向量 $\boldsymbol{x} \in \mathbf{R}^n$ 满足上面的不等式组。

2. **整系数线性不等式组的无解性**。问题的定义与上面一样，但要求 $\boldsymbol{a}_i \in \mathbf{Z}^n$ 且 $b_i \in \mathbf{Z}$，而且解 $\boldsymbol{x}$ 也属于 $\mathbf{Z}^n$。

3. **多项式方程组的无解性**。给定一组实系数多项式 $g_1(x_1, x_2, \cdots, x_n)$，$g_2(x_1, x_2, \cdots, x_n)$，$\cdots$，$g_m(x_1, x_2, \cdots, x_n)$，验证方程组 $g_i(x_1, x_2, \cdots, x_n)=0$(任意 $i=1, 2, \cdots, m$)不存在公共解。

4. **矛盾式**。给定 n 个变量上的布尔公式 φ，验证 φ 不存在满足性赋值。

5. **有限表现群中的非平凡词**。给定有限集 S 上的一个群，也就是说，群中的每个元素都是形如 $s_1^{p_1} s_2^{p_2} \cdots s_n^{p_n}$ 的词，其中 n 是正整数，p_i 是允许取负值的整数。群不是显式地给出的，而是隐式地描述为有穷个形如 $s_1^{p_1} s_2^{p_2} \cdots s_n^{p_n} = e$ 的关系，其中 $s_i \in S$，$p_i \in \mathbf{Z}$，并且 $e \in S$ 是某个指定的单位元素。在群的这种有限关系的表现形式下，给定一个词 w，如下的计算任务是非平凡的：判定能否重复利用给定的关系将 w 化简为 e。如果一个词可以化简为 e，

则这一事实将存在有限长度的证明(亦即，化简过程中依次应用的各个关系)。本例中，我们感兴趣的问题是：给定一个词 w，验证 w 不等于 e，也就是说，需要验证 w 的非平凡性。

上面的每个例子似乎都不存在显而易见的短证明。但是，有时这种直觉会让我们误入歧途。例如，注记 19.1.2 中给出的一个称为法卡斯引理(Farkas's Lemma)的结果表明，第一个问题实际上存在短证明：不等式组无解当且仅当不等式的某种组合会得到矛盾。也就是说，存在向量 $\boldsymbol{y}\in\mathbf{R}^m$ 使得 $\sum_{i=1}^{n} y_i a_i$ 是非负的但 $\sum_{i=1}^{n} y_i b_i < 0$。注意，该证明 $\boldsymbol{y}$ 的"规模"很小，因为表示 $\boldsymbol{y}$ 时所用二进制位的个数是表示输入中所有 a_i，b_i 时所用的二进制位的个数的多项式。

接下来的三个问题是 **coNP** 难的。非平凡词问题在一般情况下是不可判定的，而对于具体的群而言，该问题仍是 **coNP** 难的。于是，如果 **NP**≠**coNP**，则后几个问题不可能存在短证明。尽管如此，对每个问题的具体实例研究最短证明的长度仍然具有重要意义。验证不可满足性(在计算上，等价的说法是验证永真性，参见例 2.21)这个很自然的问题经常出现在人工智能或计算机系统(或线路)的形式验证中。这里，人们感兴趣的是某一个精心构造的问题实例的永真性。事实上，在 15.4 节讨论元数学的思考时，我们将看到与 **P** vs **NP** 相关的一个公式(或一族公式)。复杂性科学家们怀疑该公式是永真的，但是其永真性却难以被证明。类似地，在代数几何中，人们感兴趣的可能仅仅是某一个系统的特性。

注意，人们可以无条件地证明：某些语言或判定问题不存在短证明。也就是说，在 **coNP** 之外还存在其他语言(这种语言的存在性可以用第 3 章介绍的对角线方法来证明)。此外，有一个非常著名的语言——自然数上用一阶逻辑表示的所有真命题构成的语言——
308 也不存在短证明(这正是哥德尔不完全定理得出的结论，参见 1.5.2 节)。

15.2　命题演算与归结

命题逻辑是对两千年来人们常用的推理模式的形式化描述。命题逻辑讨论的基本对象是布尔公式，它的一个重要任务是验证给定的布尔公式是否为永真式(亦即，在所有赋值上均取值为"真"的公式)。为方便计，我们讨论该基本任务的反面问题：也就是说，我们考虑验证给定的布尔公式是否为矛盾式：亦即，验证给定的布尔公式是否不存在满足性赋值。而且，由于任意布尔公式都可以改写为 CNF 范式，因此我们只讨论 CNF 范式公式。具体地讲，为了验证一个一般的布尔公式 Ψ 是否为永真式，只需用第 2 章介绍的归约将 $\neg\Psi$ 变换为 CNF 范式公式(可能需要引入新的变量)，然后验证所得的公式是否为矛盾式。

下面，我们给出一个称为*归结*的简单过程，它可以用来证明给定的公式是矛盾式。令 φ 是变量 x_1，x_2，…，x_n 上的一个 CNF 范式公式，φ 的子句分别是 C_1，…，C_m。对于 $j=m+1$，$m+2$，…，归结过程将在之前得到的子句 C_1，…，C_{j-1} 上利用如下的规则导出子句 C_j：假设存在变量 x_i 和子句 C，D 使得 $x_i\vee C$ 和 $\neg x_i\vee D$ 都是之前导出过的子句(也就是说，$x_i\vee C$ 和 $\neg x_i\vee D$ 都属于 $\{C_1,\cdots,C_{j-1}\}$)，则令 $C_j=C\vee D$。注意，C_j 可能存在多种选择，因此归结得到的证明其实是一系列的选择。重复上述过程，直到得到某个变量 x_i 上的两个子句 x_i 和 $\neg x_i$。这就产生了明显的矛盾，此时，归结过程结束。φ 的*归结证明*是存在上述矛盾的子句系列 C_1，…，C_T，其中 C_1，…，C_m 是 φ 的所有子句而 C_j(其中 $j>i$)

则是在 C_1，…，C_{i-1} 上利用前面的规则导出的子句。显然，我们导出的每个子句都蕴含在之前得出的所有子句中，因此归结是一个可靠的证明系统。也就是说，φ 存在归结辩驳当且仅当 $\neg\varphi$ 是一个永真式。不难证明，归结也是完备的；也就是说，如果 $\neg\varphi$ 是永真式，则 φ 存在一个长度为 $2^{O(n)}$ 的归结辩驳(参见习题 15.1)。我们的问题是：是否真的存在归结辩驳的长度为指数的布尔公式，或者说是不是每个不可满足的布尔公式都存在多项式长度的归结辩驳呢？由于判定布尔公式的不可满足性是 **coNP** 完全问题，并且人们相信 **NP**≠**coNP**，因此我们认为上述问题的答案是否。下面，我们无条件地证明该问题的答案确实是否。

15.2.1　用瓶颈法证明下界

下面，我们介绍瓶颈法。哈肯(Haker)[Hak85]用这种方法证明得到了归结的下界。我们还将用到限制的一种特殊形式，限制曾在第 6 章中用于讨论线路的下界。

这里，我们讨论的永真式是数学中的一个基本原理——鸽巢原理。通俗地讲，鸽巢原理是说：如果有 m 只鸽子和 n 个鸽巢，其中 $m>n$，则至少有两只鸽子会飞入同一个鸽巢。用数学的话说，鸽巢原理断言：不存在一对一的满射将大小为 m 的集合映射到大小为 n 的集合。尽管鸽巢原理非常简单，但数学中很多非平凡的结果却以它为基础。比如，明科夫斯基凸体定理(Minkowski Convex Body Theorem)就建立在鸽巢原理的基础之上，参见本章注记。因此，要说明我们前面提出的问题的答案是否，一种令人信服的做法是证明"用归结这种简单的证明系统来紧凑地证明鸽巢原理是不可能的"。下面，我们来证明这件事情。

将鸽巢原理用命题逻辑的语言表述出来，得到一族永真式 $\{\neg\mathrm{PHP}_n^m : m>n\}$，其中 $\neg\mathrm{PHP}_n^m$ 是如下的 CNF 范式公式。对于整数 $i\leqslant m$，$j\leqslant n$，公式中含有一个变量 $P_{i,j}$。如果第 i 只鸽子飞入第 j 个鸽巢，则变量 $P_{i,j}$ 取值为"真"。公式中含有 $m+\binom{m}{2}n\leqslant m^3$ 个子句，各个子句的定义是：(1)为每个 $i\leqslant m$ 定义一个子句 $P_{i,1}\vee P_{i,2}\vee\cdots\vee P_{i,n}$，该子句是说第 i 只鸽子要飞入某一个鸽巢中；(2)对每组 i，$j\leqslant m$ 和 $k\leqslant n$ 定义一个子句 $\neg P_{i,k}\vee\neg P_{j,k}$，该子句是说第 i 只鸽子和第 j 只鸽子不能同时飞入第 k 个鸽巢。所有这样的子句组合在一起就表明：每个鸽巢容纳的鸽子都不超过 1 只。

定理 15.1　对任意 $n\geqslant 2$，$\neg\mathrm{PHP}_{n-1}^n$ 的任何归结辩驳都至少具有长度 $2^{n/20}$。

我们可以通过对变量进行赋值来"测试"归结辩驳是否正确。一个正确的归结辩驳证明了所有变量的任何赋值都不会满足给定的子句族。我们允许归结辩驳只证明了"在变量的某些赋值构成的子集上，给定的子句族不会被同时满足"。也就是说，将赋值子集中的每种赋值代入所有子句，归结辩驳将会正确地得出矛盾。但是，在赋值子集之外的其他赋值上，归结辩驳可能不能正确地得出矛盾。因此，这实际上放宽了对归结辩驳的要求，得到了一种宽松的归结辩驳。然而，在宽松的归结辩驳上得出的下界必然对一般的归结辩驳也成立。

为了验证鸽巢原理的归结辩驳的正确性，我们使用如下的赋值子集。子集中的每个赋值将 $n-1$ 只鸽子按一对一的方式放入 $n-1$ 个鸽巢，而暂时不让剩下的那只鸽子入巢。换句话说，取值为"真"的所有变量 $P_{i,j}$ 构成一个大小为 $n-1$ 的匹配。注意，这样的赋值恰有 $n!$ 个。如果这样一个赋值使得第 k 只鸽子未入巢，则称这个赋值是 k-临界的。

限制在上述赋值子集上讨论将减少讨论过程使用的记号，因为它可以使归结辩驳变成

309

单调的(亦即，不使用否定变量)。事实上，对于归结证明中得出的每个子句 C，我们将其中的否定变量 $\neg P_{i,j}$ 替换为 $\bigvee_{i'} P_{i',j}$ 就得到一个单调子句。容易验证，上述变换得到的单调子句和原始子句在待验证的赋值集族的同一个子集族上被满足。下面的引理表明，单调归结辩驳总会使用规模较大的子句。引理的证明稍微推迟。

引理 15.2 $\neg\mathrm{PHP}^{n}_{n-1}$ 的任何单调的归结辩驳必然会使用至少包含 $2n^2/9$ 个变量的子句。

证明(定理 15.1) 在引理 15.2 成立的条件下，定理 15.1 可以如下证明。在单调的归结辩驳中，如果一个子句至少使用了 $n^2/10$ 个变量，则称该子句是大子句。令 L 是大子句的个数，引理 15.2 断言 $L\geqslant 1$。我们在某些变量上定义一个限制使得大子句的个数大幅减少。平均值论证法表明，存在一个变量 $P_{i,j}$ 出现在 1/10 比例的大子句中。如下定义一个限制：$P_{i,j}=1$，$P_{i',j}=0$(其中 $i'\neq i$)且 $P_{i,j'}=0$(其中 $j'\neq j$)。该限制使得包含变量 $P_{i,j}$ 的所有单调子句都为"真"，因此这些子句可以从归结证明中删除，使得剩下的大子句至多还剩 $\frac{9}{10}L$ 个。而且，该限制使得证明过程需要考虑的鸽子和鸽巢都减少了 1 个，因此剩下的子句实际上是对 $\neg\mathrm{PHP}^{n-1}_{n-2}$ 的单调归结辩驳。重复上述过程 $t=\log_{10/9}L$ 遍，剩下的子句是对 $\neg\mathrm{PHP}^{n-t}_{n-t-1}$ 的不含大子句的单调归结辩驳。在此基础上，再如下运用反证法即可证得定理 15.1。注意，如果 $L<2^{n/3}$，则 $t<n/3$，进而，剩下的子句是对 $\neg\mathrm{PHP}^{n-t}_{n-t-1}$ 的单调归结辩驳，并且每个子句至多包含 $n^2/10$ 个变量。但是，$n^2/10<2(n-t)^2/9$，这与引理 15.2 矛盾。■

310

证明(引理 15.2) 对于单调归结辩驳中的每个子句 C，令

$$witness(C)=\{i:\text{存在一个 } i\text{-临界的赋值 } a \text{ 使得 } C \text{ 为"假"}\}$$

将子句 C 的复杂度 $\mathrm{comp}(C)$ 定义为 $|witness(C)|$。归结辩驳在之前得到的子句 C'，C'' 上导出子句 C 时，必然有 $\mathrm{comp}(C)\leqslant\mathrm{comp}(C')+\mathrm{comp}(C'')$，这是由于使得 C 为"假"的任意赋值都至少使得 C'，C'' 之一也为"假"。于是，如果 C 是归结辩驳中出现的第一个复杂度 $>n/3$ 的子句，则有 $n/3\leqslant\mathrm{comp}(C)\leqslant 2n/3$。我们往证：这样的子句 C 是大子句。

具体地讲，我们往证：如果 $\mathrm{comp}(C)=t$，则它至少包含 $t(n-t)$ 个变量。这就完成了引理的证明，因为 $t(n-t)>2n^2/9$。

任意取定 $i\in witness(C)$，再任意取定一个 i-临界的赋值 a 使得 C 为"假"。对任意 $j\notin witness(C)$，考虑将 i 替换为 j 之后得到的 j-临界的赋值 a'。也就是说，如果 a 让第 j 只鸽子飞入第 l 个鸽巢，则 a' 让第 i 只鸽子飞入第 l 个鸽巢而不让第 j 只鸽子入巢。由于 $j\notin witness(C)$，所得的 j-临界的赋值必然满足子句 C。因此，我们断言 C 必然包含变量 $P_{i,l}$。在取定的赋值 a 上，让 $j\notin witness(C)$ 取遍所有可能的 $n-t$ 个值，则可以看到子句 C 必然包含形如 $P_{i,l}$ 的 $n-t$ 个变量。再让 $i\in witness(C)$ 取遍所有可能的值，则可以看到 C 至少包含 $t(n-t)$ 个变量。■

15.2.2　插值定理和归结的指数下界

本节介绍下界证明的另一种不同的方法。该方法用插值定理来证得了归结的下界。插值定理这种思想很有用，复杂性理论中的几个结果都用到了这一定理。我们证得的下界本身也很有趣，因为它用到了第 14 章中给出的单调线路的下界。

我们先分别介绍插值定理的规范形式和易用形式。

定理 15.3(规范的插值定理)　设 φ 是变量 $x_1,\cdots,x_n,z_1,\cdots,z_k$ 上的一个布尔公式，而 ψ 是变量 $y_1,\cdots,y_m,z_1,\cdots,z_k$ 上的一个布尔公式(亦即，只有变量 $z_1,\cdots,z_k$

是它们公共的变量)。那么，$\varphi(x, z) \vee \Psi(y, z)$是永真式当且仅当存在布尔函数 $I: \{0, 1\}^s \to \{0, 1\}$使得

$$(\varphi(x, z) \vee I(z)) \wedge (\Psi(y, z) \vee \neg I(z)) \tag{15.1}$$

是永真式。

证明：容易看到，(15.1)式是永真式当且仅当将 z 任意固定为常值向量 c 之后要么 $\varphi(x, c)$是永真式要么 $\Psi(y, c)$是永真式。因此，如果(15.1)式是永真式，则在 x，y，z 的任意赋值上$\varphi(x, z) \vee \Psi(y, z)$都为真。另一方面，如果存在 c 使得$\varphi(x, c)$和$\Psi(y, c)$ 311
都不是永真式，则这意味着：存在 x 的赋值 a 和 y 的赋值 b 使得 $\varphi(a, c)$和 $\Psi(b, c)$都为假。 ■

我们感兴趣的是插值定理的量化形式，它将 $I(\cdot)$的计算复杂性的上界表示为最短归结辩驳的长度的函数。

定理 15.4（易用的插值定理）　在定理 15.3 的假设条件下，如果$\neg(\varphi(x, z) \vee \Psi(y, z))$有一个长度为 S 的归结辩驳，则满足定理 15.3 的函数 I 可以用规模为 $O(S^2)$的线路来计算。

而且，如果 z 中的变量在 Ψ 中都以肯定形式出现，则计算 I 的线路是单调线路(亦即，不含否定门)。类似地，如果 z 中的变量在 φ 中都以否定形式出现，则计算 I 的线路也是单调线路。

证明　要证明定理 15.4，我们需要说明：给定$\neg(\varphi(x, z) \vee \Psi(y, z))$的一个长度为 S 的归结辩驳和 z 中所有变量的一个赋值 c，我们如何才能在 $O(S^2)$时间内找出一个值 $I(c) \in \{0, 1\}$使得(a)如果 $I(c)=0$ 则 $\varphi(x, c)$是永真式；(b)如果 $I(c)=1$ 则 $\Psi(y, z)$是永真式。注意，定理 15.3 已经表明了这种 $I(c)$的存在性。

我们说明：给定 c 之后，我们如何在 $O(S^2)$时间内将$\neg(\varphi(x, z) \vee \Psi(y, z))$的长度为 S 的归结辩驳转换成$\neg\varphi(x, c)$的辩驳或者$\neg\Psi(y, c)$的辩驳。为此，我们将所有子句中的 z 变量全部"消除"。也就是说，我们将$\neg(\varphi(x, z) \vee \Psi(y, z))$的归结辩驳 C_1，…，C_S转换为$\neg(\varphi(x, c) \vee \Psi(y, c))$的归结辩驳$\overline{C}_1$，…，$\overline{C}_S$，其中每个$\overline{C}_i$要么只含 x 中的变量(称这种子句为 x-子句)要么只含 y 中的变量(称这种子句为 y-子句)。由于归结辩驳最后得出的矛盾形如"x_i且$\neg x_i$"或形如"y_i且$\neg y_i$"，因此我们最终能够证明$\overline{C}_1$，…，$\overline{C}_S$中的某个公式是一个矛盾式。

我们一步一步地完成上述变换。假设子句 C_1，…，C_{j-1}消除 z 中的变量后已经得到了子句$\overline{C}_1$，…，$\overline{C}_{j-1}$(其中每个子句要么只含 x 中的变量要么只含 y 中的变量)，我们考虑如何在 C_j 上消除 z 中的变量。注意，C_j等于 $C \vee D$，并且 $C'=w \vee C$ 和 $D'=w \vee D$(其中 w 是某个变量)都是之前已经由归结过程导出的子句。由归纳假设知道，消除 C'，D' 中的 z 变量后，我们已经得到了$\overline{C}$，$\overline{D}$。如果$\overline{C}$和$\overline{D}$都是 x 子句，则 w 必然同时出现在$\overline{C}$和$\overline{D}$中[⊖]。这样，把归结规则直接作用在$\overline{C}$和$\overline{D}$上得到$\overline{C}_j$。$\overline{C}$和$\overline{D}$都是 y 子句的情况类似地处理。如果$\overline{C}$是 x 子句而$\overline{D}$都是 y 子句，则 w 必然是一个 z 变量；此时，根据 c 很容易得到 w 的值。如果 $w=0$ 则令$\overline{C}_j=\overline{C}$；如果 $w=1$ 则令$\overline{C}_j=\overline{D}$。我们将归结辩驳的最后一个子句视为空子句，这个空子句可以视为将归结规则作用到子句 w 和$\neg w$(其中 w 是一个变量)得到的结果。这样，由于$\overline{C}_j$蕴含在 C_j中(j 是任意的)，因此在消除 z 变量后得到的归结辩驳中，最后一个子句也是一个空子句，这表明所得的归结辩驳仍结束于一个显而易见的矛盾式。 312

⊖ 我们维护一个循环不变量：从不消除 x 子句中的 x 变量和 y 子句中的 y 变量。

我们将定理中"而且"部分留作习题 15.2。这里仅指出"而且"部分之所以成立的直观原因。事实上，如果 z 变量在 Ψ 中都以肯定形式出现，则将这些变量的值从 0 改为 1 只会使得 Ψ 变为永真式的可能性更大，进而 $I(c)$ 从 0 变成 1 的可能也很大。对于 z 变量在 φ 中都以否定形式出现的情况，推理过程与此类似。 ■

现在，我们可以证明归结的下界了。

定理 15.5（指数归结下界）　对任意 $n\in\mathbf{N}$，如果 φ_n，Ψ_n：$\{0,1\}^{\binom{n}{2}}\to\{0,1\}$ 是如下定义的布尔函数：

- $\varphi_n(x,z)$=true 当且仅当在 z 表示的图中 x 表示一个大小为 $n^{1/4}$ 的团
- $\Psi_n(y,z)$=true 当且仅当在 z 表示的图中 y 表示一个 $(n^{1/4}-1)$-真着色

则存在常数 ε 使得 $\varphi_n(x,z)\wedge\Psi_n(y,z)$ 的最短归结辩驳的长度至少为 $2^{n^{\varepsilon}}$。

证明　注意，在任意的图中，如果存在 k-团，则该图将不存在 $(k-1)$-真着色。因此，在我们的定义下，$\varphi_n(x,z)\wedge\Psi_n(y,z)$ 实际上是不可满足的。此外，不难将 φ_n 和 Ψ_n 都表示为规模为 $O(n^2)$ 的 CNF 范式公式使得 z 中的变量都以肯定形式出现在 φ_n 并且都以否定形式出现在 Ψ_n 中（习题 15.3）。

定理 15.5 可以立刻由定理 15.4 和定理 14.7 得出，其中后者的证明过程得出了团函数的单调线路复杂度的指数下界。这是由于，定理 14.7 的证明过程实际上证明了：在 $k<n^{1/4}$ 时，不存在规模为 $2^{k\sqrt{k}}$ 的单调线路来区分具有 k-团的图和色数至多为 $k-1$ 的图。 ■

15.3　其他证明系统概述

下面，我们简要地介绍一下人们曾研究过的其他证明系统。有几个证明系统与 15.1 节给出的计算问题相关。

分割平面证明系统

这种证明系统主要用来验证整数变量整系数不等式组的无解性。15.1 节已经指出，该问题是一个 **coNP** 完全问题。例如，给定一个 3CNF 范式公式 φ，我们可以将它表示成一个整变量整系数不等式组使得：φ 是可满足的当且仅当不等式组有解。为此，对 φ 中的每个布尔变量 x_i，我们引入一个整数变量 X_i 使得 $0\leqslant X_i\leqslant 1$（亦即，$X_i\in\{0,1\}$）。对于 φ 的每个子句 $x_i\vee x_j\vee x_k$，我们引入不等式 $X_i+X_j+X_k\geqslant 1$（如果任何变量 x_i 的否定形式出现在子句中，则在不等式中用 $1-X_i$ 代替 X_i）。

给定一个整变量整系数的不等式组，分割平面证明系统通过在有限步骤内导出不等式 $0\geqslant 1$，由此证明不等式组的无解性。该系统产生一个不等式序列 $l_1\geqslant 0$，$l_2\geqslant 0$，…，$l_T\geqslant 0$，其中任意的第 r 个不等式是如下三种情形之一：(a)它是原不等式组中的一个不等式；(b)它是不等式 $\alpha l_u+\beta l_v\geqslant 0$，其中 α，β 是非负整数且 u，$v<r$；(c)它是在某个 l_u（其中 $u<r$）上利用如下规则得到的不等式。如果 l_u 形如

313

$$\sum_{i=1}^{n}a_ix_i-b\geqslant 0$$

其中 a_1，a_2…，a_n 的最大公因数 D 大于等于 2（亦即存在非平凡的公因数），则新的不等式为

$$\sum_{i=1}^{n}\frac{a_i}{D}x_i-\left\lceil\frac{b}{D}\right\rceil\geqslant 0$$

（值得注意的是，当 D 不能整除 b 时，$\lceil b/D\rceil$ 不等于 b/D。）分割平面证明系统也存在易用的插值定理，利用该定理可以得到分割平面证明系统的指数下界[BPR97，Pud97]。

零点系统与多项式演算系统

这里要考虑的是多项式方程组的无解性。注意，3CNF 范式公式的不可满足性也可以表示为这种方程组的无解性。事实上，对于 3CNF 范式公式中的每个变量 x_i，我们引入一个变量 X_i 和一个方程 $X_i^2-X_i=0$，这些方程将确保方程组的解满足 $X_i\in\{0,1\}$。公式中的每个子句也可以转换为一个 3 次方程，比如子句 $x_i\vee x_j\vee\overline{x_k}$ 可以转换为方程 $(1-X_i)(1-X_j)X_k=0$。

希尔伯特零点定理是代数中的基本结论，它给出判定多项式方程组的无解性的标准。域 $\mathbf{F}$ 上的方程组 $p_1(X_1,\cdots,X_n)=0$，$p_2(X_1,\cdots,X_n)=0$，$\cdots$，$p_m(X_1,\cdots,X_n)=0$ 无解当且仅当存在多项式 g_1，g_2，$\cdots$，g_m 使得

$$\sum_i g_i(X_1,\cdots,X_n)p_i(X_1,\cdots,X_n)=1 \tag{15.2}$$

注意，如果 g_1，g_2，$\cdots$，g_m 存在，则表明方程组无解；否则，将方程的解代入(15.2)式将得到 $0=1$。因此，希尔伯特零点定理的非平凡部分是证明了在每个无解的方程组上都存在 g_1，g_2，$\cdots$，g_m。(值得注意的是，所有 g_i 的系数在一般情况下将属于 $\mathbf{F}$ 的某个扩张域。但是，当方程组中包含了方程 $X_i(X_i-1)=0$(对所有 i)时，方程组的解必然满足 $X_i\in\{0,1\}$，在这种特殊情况下，所有 g_i 的系数必然属于域 $\mathbf{F}$。)

现在，我们给出零点证明系统的定义。在该系统中，p_i 是公理而无解性的一个证明是满足(15.2)式的多项式 g_i。希尔伯特零点定理表明，这个系统既是可靠的，又是完备的。我们假设多项式可以显式地表示为它的系数，而证明的规模则是表示所有 g_i 的系数所需的二进制位的个数。

多项式演算系统与零点证明系统类似。区别在于，在多项式演算系统中，g_i 可以用直线计算(straight-line computation)得到，而无需显式地表示为它们的系数。具体地讲，在多项式演算系统中，辩驳是一个有限的多项式序列 f_1，$\cdots$，f_T，其中任意的第 r 个多项式 f_r 是如下三种情形之一：(a)f_r 是输入中的某个多项式 p_i；(b)f_r 是多项式 $af_u+\beta f_v$，其中 a，β 是常数且 u，$v<r$；(c)f_r 是 x_if_u，其中 x_i 是一个变量而 $u<r$。辩驳的规模指的是 T，而辩驳的次数指的是所有多项式 f_r 的最大次数。 314

零点系统和多项式演算系统都具有指数下界，这可以通过证明辩驳次数的下界是 $n^{\Omega(1)}$ 来得到。这些下界的证明最早出现在[BCE+95]中。

弗雷格(Frege)系统与扩展的弗雷格系统

弗雷格证明系统是谓词演算中利用有限公理和推理规则进行推理的一个一般系统。归结系统是弗雷格系统的特例，它使用的推理规则都是子句(亦即，或言式)。深度受限的弗雷格系统介于归结系统和弗雷格系统之间，它要求证明过程中采用的公式都具有有限的深度。奥伊陶伊(Ajtai)[Aj88]率先给出了深度受限的弗雷格系统的下界，他的方法受启发于第 14 章中讨论 $\mathbf{AC}^0$ 下界时所采用的限制，但他所用的限制更精巧一些。

扩展的弗雷格系统是弗雷格系统的变形，它允许在证明过程中引入新的变量 y_1，y_2，$\cdots$，而且还允许在证明步骤中令 $y_i=\Psi$(其中 Ψ 是某个公式)。这种做法的优点是，我们可以在推理规则(如归结规则)中将 y_i 作为名副其实的变量来使用，这样就可能会为证明过程节省大量的推理步骤。(一般情况下，允许证明系统引入新的变量将极大地增强系统的能力。)

人们还没有得到弗雷格系统或扩展的弗雷格系统的任何下界。但是，人们已知的是，类似于插值定理这类现有的技术可能无法证得它们的任何下界了(即使我们以某些合理的复杂性假设(如"RSA 加密系统是安全的")为前提)。

15.4　元数学的思考

有几位研究者怀疑，**P** vs **NP** 这一问题可能独立于数学公理。即使不独立于数学公理，该问题也很难证明。具体的证明系统中难以证明的某些永真式有可能源自这一问题吗？

例如，考虑验证永真式的归结系统或类弗雷格系统。我们可以试着考虑某个具体的命题公式的最小证明规模，不妨设所考虑的命题公式是"规模为 n 但不能用规模为 n^{100} 的线路来计算的 SAT 实例"。这个命题公式的定义最初出现在[Rabz98]中，它有 $O(n^{100})$ 个变量（将所有变量记为 **Z**），其具体形式如下

$$\mathbf{Z} \text{ 是某个规模为 } n^{100} \text{ 的 } n\text{- 输入线路的编码} \Rightarrow \text{线路 } \mathbf{Z} \text{ 无法计算 SAT} \qquad (15.3)$$

注意，(15.3)式的结论部分是将 OR 操作作用在 2^n 个规模为 n 的输入公式上，其含义是：线路 **Z** 在这些输入公式中的某一个公式上计算得出的值不是 SAT 的真实值。因此，(15.3)式表示的命题公式的规模为 $2^{O(n)}$。当 n 充分大之后，我们可以认为它是一个永真式。但是，永真性的证明的规模为 $2^{O(n^{100})}$，这是命题公式规模的超多项式。我们能够证明这个命题公式的归结复杂度和弗雷格系统复杂度也具有超多项式的下界吗？莱兹波诺夫(Razborov)[Razb98]证明了这个命题公式在多项式演算系统中具有超多项式的下界。他还给出了这个命题公式的另一种不同的表示方法，在这种表示方法下，要获得该命题公式的
315 归结复杂度下界似乎非常困难。

莱斯(Raz)证明了上述命题公式既不是永真式，也无需超多项式的归结证明规模[Razb01，Razb03a，Razb04a]。但是，在更强的证明系统中(比如说，在弗雷格系统中)，人们还未能得出类似的下界。

独立于弱算术理论

绝大多数数学结论都可以用人们广泛接受的公理系统推导得出。这样的公理系统包括策梅洛(Zermelo)-弗兰克尔(Fraenkel)集合公理(连同选择公理)或皮亚诺(Peano)算术公理。但是，组合数学中的许多结论，由于它们刻画的是数学对象的有穷特性，故而无需这些公理系统的全部能力。因此，人们转而使用一些弱公理系统，比如库克(Cook)的 PV 系统[Coo75]或者巴斯(Buss)的"有界算术"分层 S_2^i[Bus90]。如果研究者们要试图证明 **P** vs **NP** 这一问题独立于皮亚诺公理之类的公理系统，他们或许应该先证明该问题独立于前面提到的弱理论。这些弱理论与扩展的弗雷格系统之间具有深刻的联系，"线路下界公式(Circuit Lower Bound Formulae)"在扩展的弗雷格系统上的下界可以得出这样的独立性(参见综述[Razb04b])。

本章学习内容

- 证明复杂性旨在证明永真公式在各种证明系统上的证明规模的下界。
- 如果 **NP**≠**coNP**，则在任何完备的和高效可验证的证明系统上应该都会存在某些永真公式，它们没有多项式规模的证明。
- 在某些证明系统中(如归结系统，多项式演算系统，分割平面系统等等)，人们证明了不同的永真式在这些系统上都具有指数规模下界。但是，人们还未能得到弗雷格系统和扩展的弗雷格系统的超多项式下界。

本章注记和历史

由于证明系统都是"非确定型的"，因此不存在显而易见的算法来找出短证明(如果短

证明存在)。尽管如此，许多证明系统上都可以设计出寻找短证明的启发式算法，这样的启发式算法在实践中极其有用。事实上，在绝大多数情况下，恰恰是这些启发式算法促成了证明系统的形式定义。于是，在这些系统上证明规模下界就变成了证明这些启发式算法的运行时间的下界。

例如，归结的定义受启发于戴维斯(Davis)和帕特南(Putnam)的启发式算法[DP60]，同时它还促成人们发现了其他启发式算法，如"优先将归结结果最小的两个子句进行归结"。哈肯(Haken)[Hak85]率先证明了这种启发式算法的运行时间的超多项式下界。关于这项工作的进一步扩展，请参阅[Urq87，CS88]。

类似地，契瓦塔尔(Chvatal)定义分割平面证明系统也是受启发于戈莫里(Gomory)的分割平面法[Gom63]，它是商业软件中求解整数规划问题的一种重要的启发式算法。 316

库克(Cook)和雷克霍(Reckhow)率先指出了证明规模在复杂性理论中的重要性，并开始从事证明复杂性方面的研究。有限表现群中的词问题(word problem)是数学家德恩(Dehn)在二十世纪初明确地提出的，他还在许多有意义的群上给出了求解这一问题的算法。1955 年，诺维科夫(Novikov)证明了上述问题在一般情况下是不可判定的。最近的一些研究工作表明，词问题在有些群上是 **NP** 完全问题[SBR02]，这说明非平凡词的判定问题是 **coNP** 完全问题，书[BMMS00]对此进行了综述。

易用的插值定理建立以及它在下界证明中的应用是在[Kra94，Razb95，BPR97，Kra97，Pud97]几篇论文中逐渐得出的。

多项式演算与通过计算格鲁布纳基(Groebner bases)来求解多项式方程组的算法相关。

用鸽巢原理可以为几个弱证明系统(包括归结和多项式演算)构造出难证明的永真式。但是，鸽巢原理在弗雷格系统中存在多项式规模的证明。

克拉伊切克(Krajicek)的书[Kra95]介绍了证明复杂性和有界算术。

习题

15.1　证明：如果 φ 是 n 个变量上的一个不可满足的 CNF 范式公式，φ 存在一个长度为 $2^{\Omega(n)}$ 的归结辩驳。

15.2　如下完成定理 15.4 的证明：

(a) 如果 Ψ 中的 z 变量只有肯定形式(不含否定形式)，则计算 $I(c)$ 的算法可以实现为一个规模为 $O(S^2)$ 的单调线路。

(b) 如果 φ 中的 z 变量只有否定形式(全部是否定形式)，则计算 $I(c)$ 的算法可以实现为一个规模为 $O(S^2)$ 的单调线路。

15.3　证明：定理 15.5 中的 φ_n 和 Ψ_n 都可以表示为规模为 $O(n^2)$ 的 CNF 范式公式。而且，证明：φ_n 的 CNF 范式公式可以只使用 z 变量的肯定形式，而 Ψ_n 的 CNF 范式公式可以只使用 z 变量的否定形式。

15.4　证明：分割平面系统是可靠的和完备的。

15.5　将(15.3)式用自然语言表述的公式表示为永真式。

15.6　用永真式表示本章注记中提到的鸽巢原理。 317

第 16 章

Computational Complexity, A Modern Approach

代数计算模型

用霍纳法则(Horner Rule)来计算多项式的值是最优方法吗?

——奥斯特洛夫斯基(Ostrowski)

图灵机模型刻画了二进制位串(等价于整数)上的计算。然而，许多自然而有用的算法更主要的是用来刻画不可数集合上的运算，如实数集 $\mathbf{R}$ 或复数集 $\mathbf{C}$ 上的操作。一个简单的例子是，牛顿法用于找出实值函数 f 的根，其计算过程是通过迭代找出候选实数序列 x_0，x_1，x_2，…$\in \mathbf{R}$ 使得 $x_{i+1}=x_i-f(x_i)/f'(x_i)$。在恰当的条件下，可以证明上述序列收敛于 f 的根。类似地，在数值分析、信号处理、计算几何、机器人和符号代数等领域中，许多算法所执行的基本步骤通常都会涉及域 $\mathbf{F}$ 上的(+，×，/)等基本运算(域 $\mathbf{F}$ 是任意的)。研究这种算法的领域称为计算机代数(computer algebra)[vzGG99]。

你或许会争辩说，允许算法执行任意域上的操作不现实(至少执行实数域 $\mathbf{R}$ 上的操作不现实)，因为我们的计算机实际上只能执行有限精度的计算。事实上，牛顿法之类的算法在实践中必须精心地进行实现，才能通过有限精度的算术计算使它满足给定的约束条件。本章始终采用另一种不同的方法。我们研究的计算模型允许在实数(或 $\mathbf{R}$ 之外的其他域上的数)上执行任何算术运算。这种理想的计算模型可能无法被直接实现，但是，它却可以近似地表示计算机在计算精度逐渐提高时的渐进行为，而这种渐进行为的近似表示无疑是非常有用的。此外，从复杂性下界的角度看，人们希望成熟的数学分支(如代数几何和拓扑学)中得到的现成技术可以用来证明某些复杂性下界。正如我们在第 14 章中所见，目前人们还未能证明布尔线路的任何较强的下界。

例 16.1 (代数计算模型的设计者面临的困难) 要设计出一个有意义的、具有优良性质的代数计算模型并不是一项简单的任务。事实上，允许计算模型将实数上(任意精度)的算术运算作为基本操作将马上得到一个非常强的不现实的模型。例如，在这种模型上将操作 $x \mapsto x^2$ 重复执行 n 遍就可以计算得到 2^n，其计算结果包含 2^n 个位。事实上，萨米尔(Shamir)已经证明了，在允许执行任意精度的算术运算(包括 mod 操作)的任意计算模型上可以在 poly(logN)时间内分解任意整数 N(参见习题 16.10)。然而，在经典的图灵机模型上该问题却是一个臭名昭著的难解问题。

318

此外，一个实数可以实现对无穷信息量的编码。例如，一个单独的实数足以用来编码 SAT(更一般地，任意其他语言)的所有实例的答案。因此，定义模型时必须格外小心，即使所定义的模型只允许在其程序中使用一个单独的实数，相比之下，人们却很容易让规范的图灵机在其程序中使用任意常数个整数。 ◀

克服上述困难的常规做法是限制算法独立地访问各个二进制位的能力。另一方面，如果建立模型的目的是证明复杂性下界，则考虑能力超强的不现实的模型也未尝不可。这是因为，在超强的不现实的模型上得到的下界也必然对较弱的模型成立。

本章概要的介绍代数复杂性。我们将介绍 3 个代数计算模型——代数线路、代数计算树和代数图灵机。代数图灵机模型与标准图灵机模型非常相似。在代数图灵机上，我们可

以在取自任意域的输入上研究可判定性和复杂性等问题，就像我们之前用标准图灵机在取自$\{0,1\}^*$的输入上研究类似问题一样。我们还将引入代数图灵机模型上的一个不可判定问题(曼德勃罗集(Mandelbrot Set)的成员资格判定问题)和一个 **NP** 完全问题(希尔伯特零点定理的判定形式)。一般情况下，代数复杂性和图灵机复杂性之间似乎存在密切的联系，参见 16.1.4 节。

本章始终有如下约定。算法的输入是取自某个域或环 **F**(通常是 **R** 或 **C**)上的数构成的元组。我们称输入$(x_1,\cdots,x_n)\in\mathbf{F}^n$的规模为$n$。域/环 **F** 上的一个语言指的是$\bigcup_{n\geqslant 1}\mathbf{F}^n$的一个子集合。

16.1　代数直线程序和代数线路

本节定义两个代数计算模型，它们是等价的。不同作者通常根据风格的需要和符号的简约性来选用其中一个模型。在这两个模型中，我们也定义与 **P**，**NP** 类似的复杂性类，并介绍已知的结果，包括归约概念和这些复杂性类的完全性。

16.1.1　代数直线程序

域 **F**(一般情况下，**F** 也可以是一个环)上的一个代数直线程序类似于布尔直线程序(参见注记 6.4)。代数直线程序是 C 或 C++ 等标准程序设计语言的片段，但它只允许使用"赋值"语句，而不包含循环语句或(像 IF-THEN-ELSE 这样的)条件语句。代数直线程序的形式定义如下。

定义 16.2　(**F** 上的代数直线程序)　输入变量$x_1,x_2,\cdots,x_n\in\mathbf{F}$和常数$c_1,c_2,\cdots,c_r\in\mathbf{F}$上的一个长度为 T 的代数直线程序是由形如$y_i=z_{i_1}\ \mathrm{OP}\ z_{i_2}$的 T 个语句构成的一个语句序列，其中对每个$i=1,2,\cdots,T$而言 OP 都是域操作$+$或$\times$，且每个z_{i_1}，z_{i_2}都要么是一个输入变量，要么是一个给定的常量，要么是某个$y_j(j<i)$。给定输入变量的任何一组取值，代数直线程序的一次直线计算是对所有简单语句的顺序执行，计算得到$y_1,\cdots,y_T$的值。直线计算的输出指的是y_T。具有多个输出值的代数直线程序可以类似地定义。

例 16.3　(计算多项式的值)　对于任意$a\in\mathbf{F}$，函数$\sum_{i=1}^{n}a^ix_i$可以用长度至多为$3n-2$的代数直线程序来计算。我们给出这样一个代数直线程序，它只使用一个常数(即a)。输入是$x_1,x_2,\cdots,x_n$。(这些输入可以视为一个$n-1$次多项式的系数，而我们要计算的正是这个多项式在a上的值。)直线程序先通过$n-1$个步骤计算得到$a,a^2,\cdots,a^n$，然后再通过n个步骤依次在$i=1,2,\cdots,n$上将x_i与a^i相乘，最后通过$n-1$个步骤计算累加和$\sum_{i=1}^{n}a^ix_i$。 ◀

显然，上面定义的模型是非一致的，因为在不同长度的输入上可能需要用不同的代数直线程序。通常，我们只对渐进复杂性感兴趣；也就是说，我们希望用最短代数直线程序构成的程序族来计算函数族$\{f_n\}$，其中f_n表示输入长度为n的函数。习题 16.1 要求读者证明，GF(2)上的代数直线程序本质上等价于布尔线路，进而同样的结论在任意有限域上也成立。因此，**F** 是无穷域的情况才是我们最感兴趣的。

回顾一下，多变量多项式$p(x_1,\cdots,x_n)$的次数定义为其中所有单项式的最大次数，而单项式$c\prod_i x_i^{d_i}$的次数是$\sum_i d_i$。下面的引理表明，每个代数直线程序都计算了某个多

319

项式的值，其中多项式的次数与直线程序的长度有关。

引理 16.4 变量 x_1，x_2，…，x_n 上长度不超过 T 的代数直线程序的输出是一个次数至多为 2^T 的多项式 $p(x_1, x_2, \cdots, x_n)$。

证明 简单地运用归纳法即可证得引理。由于每个输入变量 x_i 都是一个次数为 1 的多项式，而直线程序的每个步骤执行的操作是将之前得到的两个多项式相加或相乘。两个多项式的乘积的次数至多是这两个多项式次数之和。因此，每个步骤都至多使多项式的次数增大为前一步的 2 倍，故第 T 步得到的多项式的次数至多为 2^T。 ■

如果允许代数直线程序将÷作为标准操作会发生什么情况呢？由于程序无法测试一个数是否为 0，故它可能将 0 作为除数，因此程序的输出可能是没有定义的。此外，即使除式是非零多项式 $p(x)$，程序的输出仍然可能是无定义的，因为 x 可能是 $p(x)$ 的根。尽管如此，如果直线程序所计算的是形式对象，则其输出是有定义的。下面的引理表明，直线程序计算的形式对象是有理函数，即形如 $f(x_1, x_2, \cdots, x_n)/g(x_1, x_2, \cdots, x_n)$ 的函数，其中 f，g 都是多项式。有理函数 f/g 的次数是 f 和 g 的次数之和。我们略去引理的
320 (简单)证明。

引理 16.5 如果÷(除式只能是非零多项式及其倍数)可以作为直线程序中的基本操作，则对于长度为 T 的任意代数直线程序 Π 都存在一个次数至多为 2^T 的有理函数 r 使得：在 Π 有定义的所有输入值上，r 的取值和 Π 的输出均是一致的。

施特拉森(Strassen)[Str73]给出了一种一般的方法，它能将使用除法的直线程序转换为另一个不使用除法的直线程序并使得两个直线程序具有大致相当的长度，参见评注 16.8。

16.1.2 例子

作为例子，我们给出几个有意义的函数，它们都可以用多项式长度的代数直线程序来计算。

多项式乘法。给定 $(a_0, a_1, \cdots, a_n)$ 和 $(b_0, b_1, \cdots, b_n)$，计算多项式 $\sum_i a_i x^i$ 和 $\sum_j b_j x^j$ 的乘积，也就是计算向量 $(c_0, c_1, \cdots, c_{2n})$，其中 $c_k = \sum_{i+j=k} a_i b_j$。利用例 16.3 的思想，我们可以得到一个直线复杂度为 $O(n^2)$ 的算法。利用快速傅里叶变换(下一个例子)，在包含 m 次单位原根[⊖] 的域上(其中 m 是使得 $2^m > 2n$ 的最小整数)该算法的直线复杂度可以改进到 $O(n\log n)$。改进过程使用如下的思想：先用 FFT 计算两个多项式在 m 个点上的值，将相应的值分别相乘，最后利用插值(FFT 的逆变换)恢复出 c_i。类似的方法在所有域上也能得到正确的输出，但具有稍高一些的运行时间 $O(n\log n\log\log n)$(参见斯古恩黑格(Schoenhage)和施特拉森(Strassen)[SS71])。

快速傅里叶变换(FFT)。向量 $\boldsymbol{x} = (x_0, x_1, \cdots, x_{n-1}) \in \mathbf{C}^n$ 的离散傅里叶变换指的是一个向量 $M \cdot \boldsymbol{x}$，其中 M 是一个 $n \times n$ 的矩阵，它在 (i, j) 位置上的元素是 ω^{ij} 且 ω 是 1 的 n 次原根(亦即，使得 $\omega^n = 1$ 且 $\omega^r \neq 1$ 对非零 $r < n$ 成立的复数)。参见 10.6.1 节。用代数直线程序来计算离散傅里叶变换，可以简单直接地将 ω 视为常数，然后对 M 的每一行依次使用例 16.3 中的代数直线程序，这可以得到一个长度为 $O(n^2)$ 的代数直线程序。但是，

⊖ 使得 $\omega^m = 1$ 和 $\omega^i \neq 1$ 对任意 $0 < i < m$ 成立的复数称为 m 次单位原根。——译者注

我们实际上还可以找到更短的代数直线程序。事实上，利用库利(Cooley)和图基(Tukey)[CT65]给出的著名的快速傅里叶变换(参见 10.6.1 节)可以得到一个长度为 $O(n\log n)$ 的代数直线程序来计算离散傅里叶变换。人们还未能证明这个代数直线程序的最优性，但摩根斯坦(Morgenstern)[Mor73]证明了：当上述直线程序只将 0，1 作为常数使用时它是最优的。人们还对这一结果进行了一些扩展，参见[Cha94]。

矩阵乘法。在矩阵乘法问题中，给定两个 $n\times n$ 的矩阵 $X=(X_{i,j})$ 和 $Y=(Y_{i,j})$，要求计算它们的乘积 Z，其中 Z 也是一个 $n\times n$ 的矩阵且

$$Z_{i,j}=\sum_{k=1}^{n}X_{i,k}Y_{k,j} \tag{16.1}$$

由(16.1)式即可得到矩阵乘法问题的一个长度为 $O(n^3)$ 的代数直线程序。(正如前面指出的那样，代数直线程序的定义可以很容易扩展使得它输出多个值。)该代数直线程序的最优性似乎"显而易见"，因为乘积矩阵包含 n^2 个元素而每个元素需要用 $O(n)$ 次操作。但是，在 1969 年施特拉森(Strassen)[Str69]的工作基础上，人们找到了求解矩阵乘法的一系列复杂性形如 $O(n^w)$ 的算法，其中 $w<3$(参见习题 16.4)。在目前人们找到的最佳算法[CW90]中 $w\sim2.376$。人们已经证明，矩阵乘法的复杂度等价于其他几个线性代数问题的复杂度，参见综述[vzG88]。莱斯(Raz)[Raz02]已经证明：当直线程序只将 0，1 作为常数使用时，矩阵乘法的代数直线程序至少具有长度 $\Omega(n^2\log n)$。 321

行列式。$n\times n$ 矩阵 $X=(X_{i,j})$ 的行列式定义为

$$\det(X)=\sum_{\sigma\in S_n}(-1)^{\mathrm{sgn}(\sigma)}\prod_{i=1}^{n}X_{i,\sigma(i)}$$

其中，S_n 是(1，2，…，n)的所有 $n!$ 个排列构成的集合，$\mathrm{sgn}(\sigma)$ 是 σ 中反序对的个数的奇偶性，而反序对(i,j)指的是满足 $i>j$ 且 $\sigma(i)<\sigma(j)$ 的序对。行列式可以用大家熟悉的高斯消去法来计算。但事实上，行列式还可以用其他改进的算法来计算(参见习题 16.6)，而且这种改进的算法还具有很小的深度。(深度的概念将在下面的 16.1.3 节中定义。)

行列式函数可以作为一个很好的例子来说明：虽然某些函数的定义中包含了指数个项(行列式函数包含 $n!$ 个项)，而这些函数却可以用多项式长度的代数直线程序来计算。正如读者可能已经预计到的那样，代数直线程序复杂性下界的研究现状糟糕透顶。人们只证明了计算中间对称多项式的值需要用 $\Omega(n\log n)$ 个操作，却仍未在任何明确的多项式上得到任何更优的下界[BCS97]。

16.1.3　代数线路

域 $\mathbf{F}$ 上的代数线路可以参照第 6 章中的布尔线路类似地进行定义。代数线路是一个无环有向图，它的叶子称为输入结点并用 x_1，…，x_n 来标记(当然，x_1，…，x_n 不再是布尔变量，而是取值于 $\mathbf{F}$ 的变量)。我们还允许代数线路包含 k 个特殊的输入结点，这些结点用取自 $\mathbf{F}$ 的任意常数 c_1，…，c_k 来标记。代数线路的每个内结点也称为一个门，每个门上的标记是一个算术操作$\{+,\times\}$(而不再是布尔线路中采用的布尔操作 $\wedge$，$\vee$，$\neg$)。我们只考虑仅含一个输出门且所有门的入度均为 2 的代数线路。代数线路的规模指的是其中所含的门的个数。代数线路的深度指的是其中从输入结点到输出结点的最长路径的长度。我们也可以在布尔线路的各个门上使用除法操作$(\div)$。代数公式指的是所有门的出度均为 1 的代数线路。

为了计算代数线路的值，我们依次将线路中每个门上的操作应用到出现在这个门的两

条入边(或者线)上的数上，然后再将计算结果放到这个门的出边上。代数线路的输出是上述过程结束后出现在输出门的出边上的数。下面的引理表明，代数线路和代数直线程序这两个模型是等价的(引理的证明留作习题 16.7)。

引理 16.6 设 $f: \mathbf{F}^n \to \mathbf{F}$ 是一个函数。如果 f 有一个长度为 S 的代数直线程序，则 f 也有一个规模为 $3S$ 的代数线路。如果 f 可以用规模为 S 的代数线路来计算，则 f 也可以用一个长度为 S 的代数直线程序来计算。并且，如果代数线路是一个代数公式，则等价的
322 代数直线程序是一次性的(亦即，任意非输入变量 y 只在赋值操作的右端出现一次)。

注意，代数线路和代数直线程序之间的等价性涉及一个较小的常数因子 3，这是由于代数线路不允许使用并行边，因此要完成操作 $x \to x^2$，代数线路需要先通过把 x 与 0 相加来实现对 x 的复制操作。

16.1.4　代数线路中类似于 P、NP 的复杂性类

人们猜想，有些函数可能需要用超多项式规模甚至指数规模的代数线路才能计算。积和式(参见 8.6.2 节和 17.3.1 节)就是这样一个函数。$n \times n$ 矩阵 X 的积和式定义为

$$\mathrm{perm}(X) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} X_{i,\sigma(i)}$$

初看起来，积和式似乎很像行列式。但是，行列式的计算存在多项式时间算法(继而也存在多项式长度的代数直线程序)。与此不同的是，(据人们猜测)积和式的计算可能不存在多项式时间的算法。在第 17 章中，我们将证明积和式是一个 $\#\mathbf{P}$ 完全问题，也就是说，积和式的计算不存在多项式时间算法，除非 $\mathbf{P}=\mathbf{NP}$。

瓦利安特(Valiant)[Val79a]在代数线路上定义了类似于 **P**，**NP** 的复杂性类，还引入了归约的概念。行列式函数和积和式函数在这一理论中发挥了重要的作用，因为它们是下列重要复杂性类的完全问题。

定义 16.7 ($\mathbf{AlgP}_{poly}$，$\mathbf{AlgNP}_{poly}$)设 $\mathbf{F}$ 是一个域，对于多项式族 $\{p_n\}_{n\in\mathbb{N}}$，其中 p_n 含 n 个取值于 $\mathbf{F}$ 的变量，如果存在常数 c 使得在每个 n 上 p_n 的次数都至多为 cn^c，则称多项式族 $\{p_n\}$ 的次数是多项式有界的。

复杂性类 $\mathbf{AlgP}_{poly}$(需要强调基础域 $\mathbf{F}$ 时记为 $\mathbf{AlgP}^{\mathbf{F}}_{poly}$)表示可以用多项式规模和多项式次数的代数线路(不允许使用÷)来计算的具有多项式有界次数的多项式族构成的集合。

复杂性类 $\mathbf{AlgNP}_{poly}$ 是如下定义的具有多项式有界次数的多项式族 $\{p_n\}$ 构成的集合，

$$p_n(x_1, x_2, \cdots, x_n) = \sum_{(e_{n+1}, \cdots, e_m) \in \{0,1\}^{m-n}} g_m(x_1, x_2, \cdots, x_n, e_{n+1}, \cdots, e_m)$$

其中 $g_m \in \mathbf{AlgP}_{poly}$ 且 m 是 n 的多项式。

许多教科书分别用 **VP** 和 **VNP** 来表示复杂性类 $\mathbf{AlgP}_{poly}$ 和 $\mathbf{AlgNP}_{poly}$，其中 V 是瓦里安特(Valiant)名字的首字母。这是因为，正是瓦利安特定义了这两个复杂性类并证明了这两个类的性质的基础性结果。我们用 $\mathbf{AlgP}_{poly}$ 和 $\mathbf{AlgNP}_{poly}$ 来表示这两个复杂性类是为了强调它们的非一致性。

评注 16.8 在 $\mathbf{AlgP}_{poly}$ 的定义中不允许使用÷操作，这似乎具有很强的限制性。但斯
323 特拉森(Strassen)[Str73]已经证明，在有限域上，无论定义中是否允许使用÷操作，所定义的 $\mathbf{AlgP}_{poly}$ 不会改变。类似地，在 $\mathbf{AlgNP}_{poly}$ 的定义中，如果除了要求 g_m 属于 $\mathbf{AlgP}_{poly}$ 之外我们还要求 g_m 具有多项式大小的代数公式规模，则所定义的复杂性类同样不会改变

[Val79a]。

例 16.9 为了用例子来说 $\mathbf{AlgNP}_{poly}$ 的定义，我们下面证明积和式属于 $\mathbf{AlgNP}_{poly}$。为方便计，将$[n]$上的每个排列表示为一个 $n\times n$ 的排列矩阵(它的每个元素都取 0 或 1，并且每行或每列都恰有一个 1)。证明过程的关键在于，将"n^2 个变量的取值形成了一个排列"这一条件表示为多项式。

对于任意的 n 个变量 c_1，c_2，…，c_n，我们如下定义一个多项式 Exactly-one 使得：对于变量 c_1，c_2，…，c_n 的任意 0/1 赋值，如果赋值中恰有一个变量取 1，则该多项式的取值为 1；否则，该多项式取值为 0。

$$\text{Exactly-one}(c_1,c_2,\cdots,c_n)=\sum_{i\le n}c_i\prod_{j\neq i}(1-c_j)$$

现在，我们在 n^2 个变量 σ_{ij}(其中 $1\leqslant i,\ j\leqslant n$)上如下定义一个多项式 ls-permutation 来验证矩阵的每行、每列是否恰有一个 1。

$$\text{ls-permutation}(\sigma)=\prod_i\text{Exactly-one}(\sigma_{i1},\sigma_{i2},\cdots,\sigma_{in})\text{Exactly-one}(\sigma_{1i},\sigma_{2i},\cdots,\sigma_{ni})$$

最后，我们在 n^2 个变量 σ_{ij}(其中 $1\leqslant i,\ j\leqslant n$)和另外 n^2 个变量 X_{ij}(其中 $1\leqslant i,\ j\leqslant n$)上如下定义多项式 Permpoly：

$$\text{Permpoly}(\sigma,\mathbf{X})=\text{ls-permutation}(\sigma)\prod_i\left(\sum_j X_{ij}\sigma_{ij}\right)$$

显然，$\text{Permpoly}\in\mathbf{AlgP}_{poly}$。最后，$\mathbf{X}$ 的积和式可以改写为

$$\sum_{\sigma\in\{0,1\}^{n^2}}\text{Permpoly}(\sigma,\mathbf{X})$$

这就证明了积和式属于 $\mathbf{AlgNP}_{poly}$。◀

$\mathbf{AlgNP}_{poly}$ 的定义与我们期望的形式有些出入，因此多花些笔墨来讨论它是值得的。瓦利安特注意到代数运算＋类似于布尔运算 OR，这才促使他给出了 $\mathbf{AlgNP}_{poly}$ 的定义。回顾一下，语言 A 属于 **NP**，仅当存在语言 $B\in\mathbf{P}$ 使得 $x\in A\Leftrightarrow\exists e$ 使得$(x,e)\in B$。可见，**NP** 的定义涉及操作 $\exists_{e\in\{0,1\}^{n^c}}$，用 OR 操作表述出来也就是 $\vee_{e\in\{0,1\}^{n^c}}$。代数运算中类似的操作是 $\sum_{e\in\{0,1\}^{n^c}}$，这恰好就是用于定义 $\mathbf{AlgNP}_{poly}$ 的性质。注意，这使得 $\mathbf{AlgNP}_{poly}$ 比 **NP** 本质上更像 $\#\mathbf{P}$。

下面，我们介绍一个关键的概念，亦即代数问题之间能保持代数线路复杂性的归约。同普通的归约一样，我们希望问题 A 到问题 B 的归约 f 能满足如下性质：问题 B 的一个高效算法(即具有多项式长度的代数直线程序或具有多项式规模的代数线路)能够得出问题 A 的一个高效算法。曾有的一些想法指出，为了实现上述目标，所使用的归约函数只要是 [324] 可以用多项式规模的代数直线程序来计算的具有多项式有界次数的多项式族即可。瓦利安特对归约函数提出了更严格的要求：归约函数必须是极其平凡的"变量重命名"。显然，归约函数的要求越严格，在这种归约下证明完全性就越困难。因此，这种简单的归约函数能满足我们对归约的要求，这颇有些出人意料。当然，如果我们回顾自二十世纪 70 年代以来所证明的经典 **NP** 完全性结果，则不难发现，所有这些证明也都是用精巧的局部构造来对问题进行变形，而非使用任意的多项式时间算法来对问题进行变形。

定义 16.10（投影归约）　如果存在从$\{y_1,y_2,\cdots,y_m\}$到$\{0,1,x_1,x_2,\cdots,x_n\}$的映射 σ 使得 $f(x_1,x_2,\cdots,x_n)=g(\sigma(y_1),\sigma(y_2),\cdots,\sigma(y_m))$，则称函数 $f(x_1,x_2,\cdots,x_n)$ 是函数 $g(y_1,y_2,\cdots,y_m)$ 的一个投影。

如果 f 是 g 的投影，则称 f 可以投影归约到 g。

例 16.11 函数 $f(x_1, x_2)=x_1+x_2$ 可以投影归约到 $g(y_1, y_2, y_3)=y_1y_3+y_2$，因为 $f(x_1, x_2)=g(1, x_1, x_2)$。◀

可以这样来看投影归约，如果我们有一个计算函数 g 的芯片，则将该芯片的每个输入恰当地调整为 0 或 1 或 x_i，就可以得到一个计算函数 f 的芯片。下面的定理表明，行列式函数或积和式函数的芯片具有广泛的"通用性"，也就是说，适当调整它就可以用来计算一大族函数。定理的证明需要设计精巧的局部结构，此处略去其证明。

定理 16.12（行列式、积和式的完全性[Val79a]）　在任意域 $\mathbf{F}$ 上，n 个变量上可以用规模为 n 的代数公式来计算的任何多项式函数族都可以投影归约为（同一个域上）定义在 n^2+2 个变量上的行列式函数。

在特征不等于 2 的任意域上，$\mathbf{AlgNP}_{poly}$ 中的任意多项式族都可以投影归约到（同一个域上的）一个积和式函数，并且该积和式函数中变量的个数仅比多项式族中的变量个数多出多项式个。

此外，瓦利安特等人[VSBR81]还证明了，$\mathbf{AlgP}_{poly}$ 中的每个函数都存在规模为 $2^{(\log n)^2}$ 的代数公式（也参见习题 16.6）。于是，为了区分 $\mathbf{AlgP}_{poly}$ 和 $\mathbf{AlgNP}_{poly}$，只须证明如下的与计算无关的纯数学猜想。

猜想 16.13（瓦利安特）　在特征不等于 2 的任意域上，$n\times n$ 矩阵的积和式不是任何 $m\times m$ 矩阵的行列式的投影，其中 $m=2^{(\log n)^2}$。

作为一个很好的例子，猜想 16.13 展示了计算复杂性与纯数学中一些意义明确的问题之间的密切联系。另一个引人注目的事实是，为了证明 $\mathbf{P}\neq\mathbf{NP}$，我们必须先证明 $\mathbf{AlgP}_{poly}\neq\mathbf{AlgNP}_{poly}$（参见本章注记）。

325

16.2　代数计算树

本节讨论一个能力更强的代数计算模型——代数计算树。代数计算树可以在任意的环上执行计算（参见定义 16.15 后面的评述），但是为了简洁计，我们只讨论在实数值上执行计算的代数计算树。代数计算树模型是（含÷操作的）代数线路模型的增强形式，它允许计算过程根据变量 y_i 是否大于 0 来执行相应的程序分支。也就是说，代数计算树可以根据比较结果，从两个分支中选取一个分支作为下一个计算步骤。因此，代数计算树的整体结构名副其实地表现为一棵二叉树，而不再是一条直线。依据变量的值来执行相应的程序分支，这种能力类似于第 12 章中介绍的布尔判定树，只是代数计算树中的变量和输入都取实数值而不再是二进制位。

正式地讲，代数计算树可以用来求解实数输入数据上的判定问题，也就是说，代数计算树计算形如 $f: \mathbf{R}^n\to\{0, 1\}$ 的布尔函数（或语言）。

例 16.14（两个判定问题）下面的两个例子展示了我们要研究的几个实数判定问题。

元素相异性问题。给定 n 个实数 $x_1, x_2, \cdots, x_n$，判定这些元素是否各不相同。这一问题等于判定 $\prod_{i<j}(x_i-x_j)\neq 0$ 是否成立。

实值子集和问题。给定 n 个实数 $x_1, x_2, \cdots, x_n$，判定是否存在子集 $S\subseteq[n]$ 使得 $\sum_{i\in S}x_i=1$ 成立。◀

为了说明定义代数计算树的动机，我们考虑元素相异问题的平凡算法：先用$O(n\log n)$个步骤将所有输入实数排序，然后再通过$O(n)$个步骤扫描排序后的序列是否含有两个相邻的相同元素。这个平凡算法是求解该问题的最佳算法吗？或者说，能否在$o(n\log n)$个步骤内求解元素相异问题呢？答案显然依赖于所采用的计算模型。本节讨论的代数计算树是以用来实现元素相异问题的所有已知算法。正如我们将在后面 16.2.1 节中看到的那样，在代数计算树模型下，上面给出的平凡算法在忽略常数因子的情况下确实是求解元素相异问题的最佳算法。

回顾一下，在基于比较的排序算法中，每个基本操作只问"$x_i>x_j$"是否成立，而这等价于问"$x_i-x_j>0$"是否成立。注意，不等式$x_i-x_j>0$的左端是一个线性函数。我们可以设想，其他算法可能会用到其他更复杂一些的操作。代数计算树允许(a)使用任意有理函数的能力和(b)引入新的变量并在新变量上执行算术运算以及在新变量上进行条件判断。代数计算树的代价指的是最坏情况下所执行的算术运算和条件判断的总次数。 326

定义 16.15（$\mathbf{R}$ 上的代数计算树）　代数计算树是函数 $f:\mathbf{R}^n\to\{0,1\}$的一种表示方法，它表明了在输入向量$(x_1,x_2,\cdots,x_n)$上如何计算函数值 $f(x_1,x_2,\cdots,x_n)$。代数计算树是一棵二叉树，其中的每个结点具有如下形式：

- 每个叶子结点都标记为"接受"或"拒绝"；
- 计算结点 v 标记为 $y_v=y_u\ \mathrm{OP}\ y_w$，而 y_u，y_w要么是$\{x_1,\cdots,x_n\}$的元素之一要么是 v 的祖先结点上的标记，运算符 OP 取自$\{+,-,\times,\div,\sqrt{}\}$。
- 分支结点 v 的出度是 2。计算过程在结点 v 上要执行的分支依赖于形如 $y_u=0$，$y_u\geqslant 0$或 $y_u\leqslant 0$ 的条件的判断结果，其中 y_u要么是$\{x_1,\cdots,x_n\}$的元素之一要么是 v 的祖先结点上的标记。

代数计算树在任意输入$(x_1,x_2,\cdots,x_n)$上的一个计算过程对应树根结点到某个叶子结点的路径。在这条路径上，计算过程在每个内结点(包括分支结点)以不言自明的方式进行计算，直到到达叶结点。该叶结点标记为"接受"当且仅当 $f(x_1,x_2,\cdots,x_n)=1$。这条路径对应的计算复杂度用如下的代价函数来度量(该度量函数在一定程度上反映了计算的实际代价)：

- $+$，$-$操作无需任何代价
- $\times$，$\div$，$\sqrt{}$和分支结点的代价定义为 1

代数计算树的深度定义为树中所有路径的最大代价。

代数计算树的定义允许将$\sqrt{}$作为基本操作，但是在许多域(如有理数域)上这种操作可能没有定义。如果忽略$\sqrt{}$操作，则上述定义可以推广到任意有序域上。要将上述定义推广到无序域上，分支结点上执行的判断条件只能是形如"某个变量是否等于 0"的形式。

图 16-1 给出了代数计算树中的一个片段。

定义 16.16（代数计算树复杂性）　设 $f:\mathbf{R}^n\to\{0,1\}$，f 的代数计算树复杂度定义为

$$AC(f)=\min_{f\text{的代数计算树}}\{T\text{的深度}\}$$

代数计算树模型比一般编程语言的能力强大得多。这是由于深度为 d 的树可以包含 2^d 个结点，因此一棵深度为 d 的代数计算树(在最坏情况下)可以得到一个规模为 2^d 的经典算法。正因

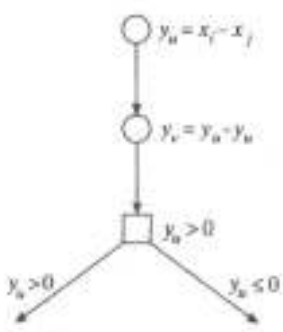


图 16-1　一棵代数计算树

327

为如此，下面的定理(证明从略)不能得到子集和问题这一 NP 完全问题的高效算法。

定理 16.17（梅尔·沃夫·德·海德(Meyer auf der Heide)[adH88]）　实值子集和问题可以用深度为 $O(n^2)$ 的代数计算树来求解。

定理 16.17 表明，代数计算树最好只用于证明某个问题具有形如 $n\log n$ 或 n^2 的下界，而不要用它来证明类似于"实数值子集和问题具有超多项式的下界"这样的宏伟目标。

16.2.1　下界的拓扑方法

要证明函数 f 的代数计算树算法的最小代价的下界，我们将用到 $f^{-1}(1)$ 和 $f^{-1}(0)$ 的拓扑结构。具体地讲，我们要用到它们的连通分支数。

定义 16.18（连通分支）　集合 $S\subseteq\mathbf{R}^n$ 称为连通的，如果任意 x，$y\in S$ 均有一条从 x 到 y 的路径位于 S 中(也就是说，存在从 $[0,1]\subseteq\mathbf{R}$ 到 $\mathbf{R}^n$ 的一个连续函数 f 使得 $f(0)=x$，$f(1)=y$ 且 $f(t)\in S$ 对任意 $t\in[0,1]$ 成立)。$W\subseteq\mathbf{R}^n$ 的一个连通分支是 W 的一个连通子集且该连通子集不是 W 的其他连通子集的真子集。我们用 $\#(W)$ 表示 W 中连通分支的个数。

下面的定理建立了连通分支数与代数计算树复杂性之间的联系。

定理 16.19（代数计算树的拓扑下界[B0083]）　对任意函数 $f:\mathbf{R}^n\to\{0,1\}$ 具有

$$AC(f)=\Omega(\log(\max\{\#(f^{-1}(1)),\#(\mathbf{R}^n\setminus f^{-1}(1))\})-n)$$

在证明定理之前，我们先用它得出元素相异问题的下界。前面已经指出这个下界，它可以用下面的定理立刻得到，因为 $\log n!=\Omega(n\log n)$。

定理 16.20　令 $W=\{(x_1,x_2,\cdots,x_n)\mid\prod_{i\neq j}(x_i-x_j)\neq 0\}$，则 $\#(W)\geqslant n!$。

证明　对每个排列 σ，令

328

$$W_\sigma=\{(x_1,x_2,\cdots,x_n)\mid x_{\sigma(1)}<x_{\sigma(2)}<\cdots<x_{\sigma(n)}\}$$

也就是说，W_σ 是严格遵循由排列 σ 所给顺序的所有 n 元组 $(x_1,x_2,\cdots,x_n)$ 构成的集合。只需证明，如果 $\sigma\neq\sigma'$，则 W_σ 和 $W_{\sigma'}$ 是不连通的。

对任意两个不同的排列 σ 和 σ'，存在不同的 i，j(其中 $1\leqslant i,j\leqslant n$)使得 $\sigma^{-1}(i)<\sigma^{-1}(j)$ 但 $\sigma'^{-1}(i)>\sigma'^{-1}(j)$。因此，在 W_σ 中有 $X_i-X_j>0$ 而在 $W_{\sigma'}$ 中却有 $X_i-X_j>0$。考虑从 W_σ 到 $W_{\sigma'}$ 的任意一条路径。由于在路径的两个端点上，X_i-X_j 具有不同的符号，故由中值定理可知在路径中间的某个点上必然存在 $X_i-X_j=0$ 的点。这个点既不属于 W_σ，也不属于 $W_{\sigma'}$。由定义 16.18 可知，W_σ 和 $W_{\sigma'}$ 不连通。■

现在，我们回头证明定理 16.19。证明过程分为两步。首先，我们找出代数计算树复杂度为 d 的函数的性质：这种函数都可以定义为"为数不多"的一些方程组的解集。

引理 16.21　如果 $f:\mathbf{R}^n\to\{0,1\}$ 有一个深度为 d 的代数计算树，则 $f^{-1}(1)$(和 $f^{-1}(0)$)可以表示为至多 2^d 个集合 C_1，C_2，…，$\subseteq\mathbf{R}^n$ 的并集，其中 C_i 是如下定义的集合。存在一个至多含 d 个方程的方程组，它的每个方程形如

$$p_i(y_1,\cdots,y_d,x_1,\cdots,x_n)\bowtie 0$$

其中 $p_i(r\leqslant d)$ 是一个二次多项式，$\bowtie$ 属于 $\{\leqslant,\geqslant,=,\neq\}$，$y_1$，…，$y_d$ 是新变量。C_i 是使得"存在 y_1，…，y_d 使 $(y_1,\cdots,y_d,x_1,\cdots,x_n)$ 是上述方程组的解"的所有 $(x_1,\cdots,x_n)$ 构成的集合。此外，不失一般性，我们还可以假设方程组不使用 $\neq$ 作为约束条件(此时，方程中变量 y_i 的个数需要翻倍)。

证明　由于 f 的代数计算树至多有 2^d 个叶结点，因此只需为树的每个叶结点关联一个集合 C_ℓ，它由计算过程终止于该叶结点的所有 $(x_1, \cdots, x_n)$ 构成。在从树根到该叶结点的路径上，为每个计算结点或分支结点引入一个变量 y_i，这样，总共至多引入 d 个变量 $y_1, y_2, \cdots y_d$。对每个计算结点，我们用不言指明的方式为它关联一个方程(如图 16-2)。例如，如果该计算结点执行计算 $y_i = y_u \div y_w$，则意味着该结点满足约束 $y_w y_i - y_u = 0$。对每个分支结点，我们为它关联一个显而易见的不等式。因此，对于任何导致计算终止于该叶结点的 $(x_1, \cdots, x_n)$，都存在值 $y_1, y_2, \cdots$，使得二者的组合同时满足上述的至多 d 个方程或不等式。

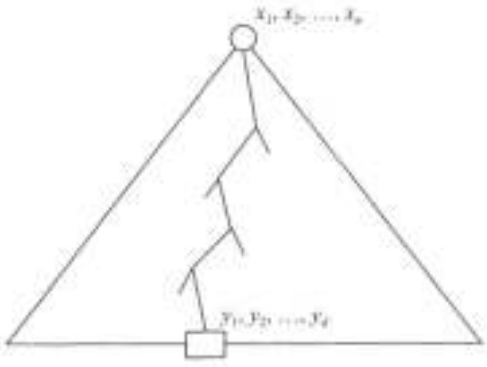


图 16-2　长度为 d 的计算路径 p 定义了对应于 p 上结点的在 n 个输入变量 x_i 和 d 个引入变量 y_i 上的约束集

为了将"$\neq$"约束替换为"$=$"约束，我们为每个形如

$$p_i(y_1, \cdots, y_m) \neq 0$$

的约束引入一个变量 z_i，并将它替换为约束

$$q_i(y_1, \cdots, y_m, z_i) = 1 - z_i p_i(y_1, \cdots, y_m) = 0$$

(这种变换称为拉比诺维茨(Rabinovitch)技巧，它在所有域上均成立。)注意，该约束的次数仍然等于 2，因为上述变换技巧仅把分支结点上的约束 $y_w \neq 0$ 转变为约束 $1 - z_w y_w = 0$。

类似地，对于形如 $p_i(y_1, \cdots, y_m) > 0$ 的约束，我们可以引入变量 z_i 并将它转换为约束 $p_i(y_1, \cdots, y_m) = z_i^2$。 ■

在 16.3.2 节证明希尔伯特零点定理的判定形式的完全性时，我们还会遇到拉比诺维茨技巧这种有用的变换。

现在，为了用拓扑方法证明 $AC(f)$ 的下界，我们需要一个关于"代数系统的解集中所含连通分支的个数"的一个结论。下面的定理是数学中的一个非常重要的结论。

定理 16.22　(由来勒-托姆定理(Milnor-Thom Theorm)得出的结论)　如果 $S \subseteq \mathbf{R}^n$ 是由次数为 d 的 m 个等式和 h 个不等式所定义的集合，则 $\#(S) \leqslant d(2d-1)^{n+k-1}$。

注意，定理 16.22 给出的上界独立于 m。由此，我们可以证明定理 16.19(也称本·欧尔定理(Ben-Or's Theorem))。

定理 16.19 的证明　假设 f 的代数计算树的深度是 d，于是树中至多有 2^d 个叶结点。我们将使用如下的事实：如果 $S \subseteq \mathbf{R}^n$ 且 $S|_k$ 表示将 S 中所有点的后 $n-k$ 个坐标去除之后得到的点集(亦即，在前 k 个坐标上投影得到的点集)，则 $\#(S|_k) \leqslant \#(S)$(参见图 16-3)。

树中的每个叶结点有一族次数为 2 的约束(引理 16.21)。因此，考虑叶结点 ℓ 对应的约束族 C_ℓ，把其中用到的所有变量记为 $y_1, \cdots, y_d, x_1, \cdots, x_n$。令 $W_\ell \subseteq \mathbf{R}^n$ 是计算过程终止于叶结点 ℓ 的所有输入构成的子集，而 $S_\ell \subseteq \mathbf{R}^{n+d}$ 是满足约束族 C_ℓ 的所有点构成的集合。注意，$W_\ell = S_\ell|_n$，也就是说，W_ℓ 是 S_ℓ 在前 n 个坐标上投影得到的点集。因此，W_ℓ 中的连通分支数以 $\#(S_\ell)$ 为上界。由定理 16.22 可知，$\#(S_\ell) \leqslant 2 \cdot 3^{n+d-1} \leqslant 3^{n+d}$。于是，$f^{-1}(1)$ 中连通分支的总数不超过 $2^d 3^{n+d}$。于是，$d \geqslant \Omega(\log(\#(f^{-1}(1)))) - O(n)$。同理，$f^{-1}(0) = \mathbf{R}^n \setminus f^{-1}(1)$ 中的连通分支的总数也不超过 $2^d 3^{n+d}$；故 $d \geqslant \Omega(\log(\#(\mathbf{R}^n \setminus f^{-1}(1)))) - O(n)$。 ■

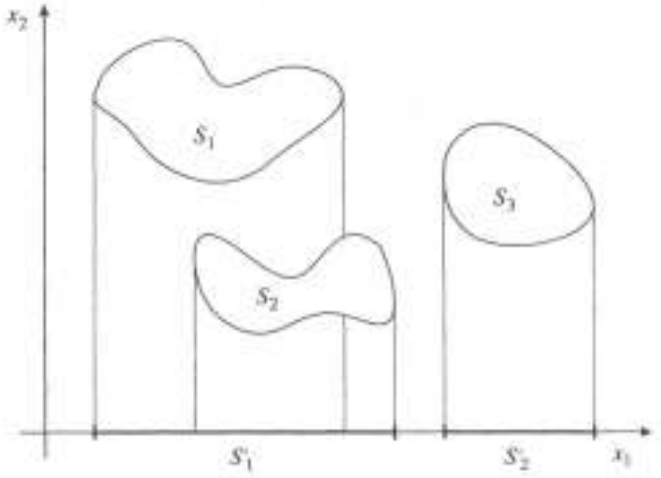


图 16-3　投影的合并不会导致连通分支的增加

16.3　布卢姆-舒布-斯梅尔模型

前面两种代数复杂性模型都是非一致的。本节介绍布卢姆(Blum)、舒布(Shub)和斯梅尔(Smale)[BSS89]引入的一种一致的代数复杂性模型(简称 BSS 模型，也称为代数图灵机)。该模型中的图灵机可以在任意域或环 $\mathbf{F}$(例如 $\mathbf{F}=\mathbf{R}$，$\mathbf{C}$，GF(2)等)上执行计算。模型的输入是 $\mathbf{F}^n$(其中 $n\geqslant 1$)中的字符串，模型的输出是"接受"或"拒绝"。模型的每个存储单元可以容纳 $\mathbf{F}$ 中的一个元素。初始时，模型中除输入之外的其他无穷个存储单元都存储"空白符"。可见，这个模型是标准图灵机模型的推广。注意，标准图灵机模型只能执行位操作，这些操作也可以视为 GF(2)上的操作，参见习题 16.9。模型中每个机器都有有穷多种内部状态，每个状态属于如下三类状态之一：

- 读写头移动状态：将读写头从当前位置向左或向右移动一个存储单元；
- 分支状态：如果读写头指向的当前存储单元存放了 a，则机器状态转换到 q_1，否则机器状态转换到 q_2；
- 计算状态：这种状态有一个"用硬件实现的"的函数 f 与之关联。如果机器处于这种状态，并且读写头从当前存储单元读取的值是 $a\in\mathbf{F}\cup\{$空白符$\}$，则将它替换为函数值 $f(a)$。如果 $\mathbf{F}$ 是一个环，则 f 是一个多项式；而如果 $\mathbf{F}$ 是一个域，则 f 可以是形如 g/h 的任意有理函数，其中 g，h 是多项式且 h 不等于 0。无论哪种情况，f 都用 $\mathbf{F}$ 中的常数个元素来表示。这些表示 f 的元素可以视为机器中"用硬件实现的"常数。

注意，在标准图灵机模型中，计算操作和分支操作可以在同一个计算步骤中完成。然而，在 BSS 模型中，这两类操作必须分开执行。这样做纯粹是为了记号上的方便。但如此一来，机器为了执行分支操作必须"记住"前一个操作步骤读到的数值。正因为如此，机器有一个单独的"寄存器"用来拷贝读写头在当前存储单元读到的值，以便机器在下一个操作中使用这个值。

同前面研究过的两种代数复杂性模型一样，BSS 模型的能力似乎也强于真正的计算机。例如，通过不断重复操作 $x\leftarrow x^2$，BSS 模型能够用 n 个计算步骤在一个存储单元内完成对 x^{2^n} 的计算(而不引起溢出)。

331

但是，由于机器只能用类似"存储单元的内容是否等于 a"这样的测试来执行分支操作，因此这种将计算操作和分支操作合二为一的办法只能有限地增加机器的计算能力。如果允

许执行稍加变形的测试——"存储单元的内容是否大于 a"，则机器的计算能力将大幅度地增强。事实上，如此一来，机器将能够在多项式时间内判定 $\mathbf{P}_{/poly}$ 中的任何语言(特别地，机器将能够在多项式时间内判定不可判定语言)。原因在于，$\mathbf{P}_{/poly}$ 中语言的线路族可以表示为一个单独的实数，并且在 BSS 图灵机上该实数可以"用硬件实现为一个常数"。具体地讲，这个实数可以实现为某个状态所关联的多项式 $p(x)$的一个系数。要单独地访问系数中一组二进制位，只需连续若干次将系数除以 2 然后用分支测试来查看所得的值是否大于 0(具体细节留作习题 16.12)。于是，机器能够从系数中抽取得到每个线路的多项式长度的编码。

类似地，如果允许机器将舍入操作($\lfloor x \rfloor$的计算)作为基本操作，则在 BSS 模型上有望用多项式时间实现整数的因数分解，当然这仍需借助萨米尔(Shamir)早先的思想(参见习题 16.10)。

注意，与 BSS 模型密切相关的还有一个经典模型：具有"分支"门和"一致"条件的代数线路(这样，输入规模不同的各个线路可以用同一个传统图灵机来构造)。这种联系类似于第 6 章中介绍的"传统图灵机与一致布尔线路"之间的密切联系。

16.3.1　复数上的复杂性类

本节定义 BSS 模型相关的一些复杂性类。为简单计，我们只讨论 $\mathbf{C}$ 上的代数图灵机。同以往一样，代数图灵机的复杂性也表示成输入规模的函数，而输入规模指的是输入数据占用的存储单元的个数。下面定义两个复杂性类，它们是 $\mathbf{C}$ 上类似于 **P** 和 **NP** 的复杂性类。

定义 16.23　($\mathbf{P}_{\mathbf{C}}$，$\mathbf{NP}_{\mathbf{C}}$)　$\mathbf{P}_{\mathbf{C}}$ 包含 $\mathbf{C}$ 上"可以用 $\mathbf{C}$ 上的多项式时间的 BSS 图灵机来判定"的所有语言。语言 L 称为属于 $\mathbf{NP}_{\mathbf{C}}$，如果存在语言 $L_0 \in \mathbf{P}_{\mathbf{C}}$ 和数 $d>0$ 使得：任意输入 x 属于 L 当且仅当存在 $\mathbf{C}^{d'}$ 中的字符串 $(y_1, \cdots, y_{d'})$ 使得 (x, y) 属于 L_0。

人们感兴趣的另一件事情是，用代数图灵机模型来研究标准语言(其输入是二进制串)的复杂性。因此，我们有如下的定义。

$$0\text{-}1\text{-}\mathbf{NP}_{\mathbf{C}} = \{L \cap \{0,1\}^* \mid L \in \mathbf{NP}_{\mathbf{C}}\}$$

注意，0-1-$\mathbf{NP}_{\mathbf{C}}$ 机器的输入是二进制串，但是其非确定型"证明"却可以使用复数。显然，**NP** 是 0-1-$\mathbf{NP}_{\mathbf{C}}$ 的子集。原因在于，虽然 BSS 机器的"证明"可能是一串复数，但是机器可以先用相等测试来检验这些复数是否都等于 0 或 1。一旦验证了"证明"实际上是一个二进制串，则机器可以像正常的图灵机一样来验证"证明"的真伪。

0-1-$\mathbf{NP}_{\mathbf{C}}$ 真的比 **NP** 大吗？人们已经证明 0-1-$\mathbf{NP}_{\mathbf{C}} \subseteq$ **PSPACE**。1997 年，柯依兰(Koiran)[Koi97]证明了：如果假设黎曼猜想(Riemann Hypothesis)成立，则 0-1-$\mathbf{NP}_{\mathbf{C}} \subseteq$ **AM**。第 20 章中将证明(参见习题 20.7)，在合理的假设下有 **AM** = **NP**。因此，柯依兰的结论表明，0-1-$\mathbf{NP}_{\mathbf{C}}$ = **NP** 有可能成立。

332

16.3.2　完全问题和希尔伯特零点定理

语言 $\mathbf{HN}_{\mathbf{C}}$ 定义为希尔伯特零点定理在 $\mathbf{C}$ 上的判定形式(我们曾在 2.7 节和 15.3 节中遇到过希尔伯特零点定理)。问题的输入是 $x_1, \cdots, x_n$ 上的 m 个多项式 $p_1, \cdots, p_m$。问题的输出是"YES"当且仅当这 m 个多项式存在公共根 $a_1, \cdots, a_n$。注意，这个问题是一个非常一般的问题，它表示了 SAT 问题，因为 SAT 实例中的每个子句都可以表示为一个 3 次多项式：

$$x \vee y \vee z \mapsto (1-x)(1-y)(1-z) = 0$$

下面利用上述事实来证明 0-1-$\mathbf{HN}_{\mathbb{C}}$(所有多项式的系数都等于 0 或 1)在 0-1-$\mathbf{NP}_{\mathbb{C}}$ 中是完全的。

定理 16.24 ([BSS89]) 0-1-$\mathbf{HN}_{\mathbb{C}}$ 在 0-1-$\mathbf{NP}_{\mathbb{C}}$ 中是完全的。

证明概要 直接验证可知，0-1-$\mathbf{HN}_{\mathbb{C}}$ 属于 0-1-$\mathbf{NP}_{\mathbb{C}}$。我们模仿库克-勒维定理(定理 2.10)来证明 0-1-$\mathbf{HN}_{\mathbb{C}}$ 是 0-1-$\mathbf{NP}_{\mathbb{C}}$ 难的。回顾一下，每个 **NP**-计算都可以归约为一些局部测试的 AND，而每个局部测试都只依赖于常数个变量。这里，我们进行同样的归约。同代数计算树上的推理过程(引理 16.21)一样，我们将局部测试表示为次数有界的一些多项式约束。计算状态 $c \leftarrow q(a, b)/r(a, b)$ 很容易处理，只需令 $p(c)=q(a, b)-cr(a, b)$ 即可。对于分支状态 $p(a, b)\neq 0$，我们用拉比诺维茨技巧(Rabinovitch's Trick)将它转换为相等测试 $q(a, b, z)=0$。因此，所得约束的次数依赖于机器上"用硬件实现的"多项式的次数。并且，这些约束多项式的系数都是实数(这些实数都是机器上"用硬件实现"的实数)。将这些多项式约束转换为仅以 0 和 1 为系数的约束，这是一个复杂的工作。该过程的思想是证明：机器上"用硬件实现的"的实数不会影响所得约束的系数是否为 0/1，因为问题的输入是二进制串。这里略去对这一过程的论述。 ■

16.3.3 判定性问题——曼德勃罗集

由于 BSS 模型的计算能力远远强于普通的图灵机，因此研究 BSS 模型上的不可判定问题也是有意义的。本小节给出 BSS 模型上的一个不可判定问题——曼德勃罗集(Mandelbrot set)的成员资格判定问题。曼德勃罗集是一个著名的分形。本章注记中提到了做这种研究的一个动机，它与罗杰·彭罗斯(Roger Penrose)的断言"人工智能不可能实现"密切相关。

定义 16.25 (曼德勃罗集判定问题) 令 $P_C(Z)=Z^2+C$。曼德勃罗集定义为

333

$$\mathcal{M}=\{C\in\mathbf{C}\mid \text{序列 } P_C(0),\ P_C(P_C(0)),\ P_C(P_C(P_C(0))),\ \cdots \text{是有界的}\}$$

注意，如果我们允许使用不等式约束，则 $\mathcal{M}$ 的补问题是可以判定的。这是由于序列是无界的当且仅当复数 $P_C^k(0)$ 的模大于 2 对某个 k 成立(请读者自己证明!)。而上述条件可以通过有限次测试来验证。但是，验证 $P_C^k(0)$ 有界对任意 k 成立却困难得多。事实上，我们有如下的定理。

定理 16.26 用 **C** 上的代数图灵机无法判定 $\mathcal{M}$ 的成员资格。

证明概要 定理的证明过程要用到一些超出本书范围的数学工具，因此我们只给出一个粗略的证明概要。证明过程要用到曼德勃罗集的拓扑性质和豪斯多夫维(Hausdorff Dimension)的概念。测度空间中的一个半径为 r 的球是形如 $B(x_0, r)=\{y:\ \mathrm{dist}(x_0, y)\leqslant r\}$ 的集合。粗略地讲，测度空间中一个集合的豪斯多夫维指的是如下的数 d：当 $r\to 0$ 时，覆盖该集合所需的半径为 r 的球的个数趋于 $1/r^d$。

设复数域上的代数图灵机 $\mathcal{N}$ 可以判定曼德勃罗集的成员资格。考虑该图灵机在 T 个步骤内所做的计算。同定理 16.24 和定理 16.21 的推理过程一样，我们可以得到如下结论(参见习题 16.11)：$\mathcal{N}$ 在 T 步之内能接受的所有输入是有限个半代数集的并集(其中，半代数集指的是可以用多项式方程组的解集来定义的集合)。这样，$\mathcal{N}$ 接受的语言是可数个半代数集的并集。由人们已知的结论可知，这意味着 $\mathcal{N}$ 接受的语言的豪斯多夫维等于 1。但是，人们还证明了曼德勃罗集的豪斯多夫维等于 2。因此，$\mathcal{M}$ 的成员资格不能由 $\mathcal{N}$ 来判定。 ■

本章学习内容

- 可以在更代数化的场景下来研究计算复杂性，此时，基本操作可以执行域或环上的

代数操作。我们在这种情形下研究了布尔线路、直线程序、判定树和图灵机的类似模型。

- 在某些代数复杂性类中，我们可以定义完全问题，甚至还可以研究不可判定问题。
- 证明代数计算树的复杂性下界时，我们使用了一种有用的拓扑学方法。该方法考虑的是多项式方程组解集中的连通分支数。
- 代数复杂性与本书其余章节研究的各种复杂性之间存在着明确的有意义的联系。两个典型的例子是：(a)瓦利安特(Valiant)的结果表明，积和式属于 $\mathbf{AlgNP}_{poly}$；(b)BSS模型中图灵机所定义的以复数为输入的两个复杂性类与标准复杂性类之间存在着密切联系。

本章注记和历史

人们自然希望研究："在输入上得到想要的输出"所需要执行的代数操作的最少个数。肖尔茨(A. Scholz)[Sch37]在 1937 年率先给出了这一问题的形式化描述，这与图灵在不可判定性方面所开展的工作大致属于同一时期。直线程序这一概念的提出要追溯到奥斯特洛夫斯基(Ostrowski)[Ost54]在"用霍恩法则来计算多项式的值是否为最佳方法"方面开展的研究工作。直线程序的正式定义和代数计算树的正式定义分别出现在斯特拉森(Strassen)[Str72]和拉宾(Rabin)[Rab72]，但是，拉宾只讨论了线性函数而没有讨论一般的多项式。斯特拉森在上世纪六、七十年代为代数复杂性理论的建立开展了大量的研究工作。克鲁斯(Knuth)自 1969 年开始陆续发表的三卷书[Knu69]中的第二卷对当时代数复杂性理论的研究进展进行了很好的综述。代数计算树在计算几何中也得到了研究。此时，人们将基于线性函数进行的 3-分支判断解释为询问输入点 $x \in \mathbf{R}^n$ 是位于线性函数描述的平面上，还是在平面左侧或在平面右侧。

334

类 $\mathbf{AlgP}_{poly}$ 和 $\mathbf{AlgNP}_{poly}$ 是瓦利安特(Valiant)[Val79a]定义的，他当时用"**P**-可定义的"来表示 $\mathbf{AlgNP}_{poly}$ 而用 **AlgP** 来表示 $\mathbf{AlgP}_{poly}$。后续的一些工作用 **VNP** 和 **VP** 来分别表示这两个类。相关理论在斯基兀姆(Skyum)和瓦利安特发表[SV85]这篇论文之后变得逐渐成熟起来，他们还通过在 **NP** 类上的投影来扩展了瓦利安特的完全性理论。这项扩展工作依赖于如下的观察结果：库克-勒维归约本身就是一种投影归约。这项扩展工作的一个重要结论是：在人们解决 $\mathbf{P} \neq \mathbf{NP}$ 这一问题之前，必须先解决 $\mathbf{AlgP}_{poly} \neq \mathbf{AlgNP}_{poly}$ 这一问题。

16.1.4 节中提到的计算行列式的 **NC** 算法源自钱基(Csanky)[Csa76]。该算法在特征为 0 的所有域上都是正确的。人们还提出了该算法的许多推广形式。行列式存在规模为 $2^{poly(\log n)}$ 的代数公式，这一事实是瓦利安特等人[VSBR81]证明的。事实上，他们给出了具有一般性的变换过程，它可以将计算多项式 f 的规模为 S 的任意代数线路转换成一个计算 f 的深度为 $O(\log S \log deg(f))$ 的代数线路。(同往常一样，线路的深度指的是线路图中从输入结点到输出结点的最长路径的长度。)

"证明代数计算树的下界"这一问题具有悠久的历史，本·欧尔定理(定理 16.19)出现在这段历史的中期。最近由博约纳(Bjorner)[BLY92]和姚期智[Yao94]开展的一些工作表明了在 $\#(W)$ 比较小的情况下如何证明代数计算树的下界。他们给出的方法用到了与集合相关的其他拓扑参数，如贝蒂数(Betti Numbers)。

1997 年出版的 Bürgisser 等人的书[BCS97]是讨论代数复杂性的一般性著作，它既讨论了算法，也讨论复杂性下界。计算代数的现代综述可以参阅冯·楚·加滕(von zur Gathen)和格哈德(Gerhad)的书[vzGG99]。

本章未涉及的一个重要主题是斯特拉森的证明代数线路下界的技术。这种技术以代数族的次数为基础，用它可以证明一些问题的 $\Omega(n\log n)$ 下界是最优的。一个相关的主题是著名的鲍尔-斯特拉森引理(Baur-Strassen Lemma)。该引理表明，计算 f 的偏导数所需的计算资源恰好是计算 f 本身所需的计算资源。上述两个主题的细节都可以参见[BCS97]。

BSS 模型相关结论的最佳综述是布卢姆(Blum)等人的书[BCSS97]。曼德勃罗分形集合的成员资格判定问题源自罗杰·彭罗斯(Roger Penrose)对人工智能的争议性批评[Pen89]。这场争论由来已久，但梗概地看，罗杰·彭罗斯提出的一个论题是：人类可以通过直觉来理解和掌握许多图灵机模型无法刻画的东西。他以曼德勃罗集的定义作为具体例子，讨论了实数域 $\mathbf{R}$ 上的计算来支持他的论点。他指出，曼德勃罗集这样的数学对象超出了计算机科学的范畴。他还提出建议，让大家都不要讨论这种集合的判定性问题。布卢
335 姆(Blum)，舒布(Shub)和斯梅尔(Smale)的工作却表明，对图灵机稍加变形就可以讨论曼德勃罗集的判定性问题。布雷弗曼(Braverman)和库克(Cook)最近的综述[BC06]对 BSS 模型进行了谨慎的评价，他们指出了代数图灵机在概念上的一些局限性，并建议用基于二进制位的模型(与标准图灵机很接近)来作为实数计算的模型。

习题

16.1　证明：对于任意的有限域 $\mathbf{F}$，存在一个常数 c 使得：对于布尔线路复杂度为 S 的任意布尔函数 $f:\{0,1\}^n\to\{0,1\}$，域 $\mathbf{F}$ 上计算 f 的最小代数线路的规模介于 S/c 和 $c\cdot S$ 之间。

16.2　梗概地给出快速傅里叶变换的一个长度为 $O(n\log n)$ 的代数直线程序。

16.3　梗概地给出一个算法，用 $O(n\log n)$ 个复数操作(+和×)将两个 n 次一元变量复数系数的多项式相乘。

16.4　([Str69])

(a) 证明：对于任意 $\omega>2$，如果存在 $k\in\mathbf{N}$ 和一个至多含 k^ω 个乘法门的代数直线程序 $\prod_k$ 来计算 $k\times k$ 矩阵的乘积[⊖]，则对任意 $n\in\mathbf{N}$ 均存在一个长度为 $O(n^\omega)$ 的代数直线程序来计算 $n\times n$ 矩阵的乘积。

(b) 证明：存在一个仅含 7 个乘法门的代数直线程序来计算 2×2 矩阵的乘法。由此得出结论，存在长度为 $O(n^{2.81})$ 的代数直线程序来计算 $n\times n$ 矩阵的乘法。

16.5　证明：可以用深度为 d 的代数线路来计算的任意函数也可以用一个规模为 $O(2^d)$ 的代数公式来计算。

16.6　([Berch])本习题为行列式函数给出一个具有较小深度的多项式规模的代数线路。这种线路也可以用高斯消去法和斯特拉森(Strassen)[Str73]的除法消去技术来得到。

(a) 证明：存在深度为 $O(\log n)$ 且规模为 $O(n^3)$ 的代数线路来计算 $n\times n$ 矩阵的乘法。

(b) 证明：对任意 $i\in[n]$，均存在深度为 $O(\log^2 n)$ 且规模为 $O(n^3)$ 的代数线路来计算任意 $n\times n$ 矩阵 M 的 i 次幂 M^i。

(c) 回顾一下，矩阵 A 的特征多项式 p_A 定义为 $p_A(x)=\det(A-xI)$。证明：如果 $A=\begin{pmatrix}A_{1,1} & \boldsymbol{r}\\ \boldsymbol{c} & M\end{pmatrix}$，其中 M 是 $(n-1)\times(n-1)$ 的矩阵，$\boldsymbol{r}$ 是 $n-1$ 维行向量而 $\boldsymbol{c}$ 是

⊖ 此处，作者叙述过程中利用了代数线路和代数直线程序之间的等价性。——译者注

$n-1$ 维列向量，则 $p_A = C \cdot p_M$（将 p_M 视为一个向量，它的第 i 个分量对应 x^{n-i} 的系数，$1 \leqslant i \leqslant n+1$），其中 C 是如下定义的 $(n-1) \times n$ 矩阵：

336

$$C_{i,j} = \begin{cases} 0 & i < j \\ 1 & i = j \\ -A_{1,1} & i = j+1 \\ -\mathbf{r}M^{i-j-2}\mathbf{c} & i \geqslant j+2 \end{cases}$$

(**d**) 证明：行列式可以用深度为 $O(\log^2 n)$ 且规模为 poly(n) 的代数线路来计算。（经过各种优化，代数线路的规模可以减小到 $O(n^{\omega+\varepsilon+1})$，其中 ε 是任意常数，ω 是“计算矩阵乘法的深度为 $O(\log n)$ 且规模为 $O(n^\omega)$ 的代数线路”中的常数 ω。）

16.7　证明引理 16.6。

16.8　假设我们期待算图中哈密顿环（Hamilton cycle）的个数。将这个问题表示为一个代数计算问题，并证明这个问题属于 $\mathbf{AlgNP}_{/poly}$。

16.9　证明：域 GF(2) 上的 BSS 模型等价于标准图灵机模型。

16.10　（萨米尔（Shamir）[Sha79]）证明：允许在任意大的自然数上执行算术操作（包括“mod”和整数除法）的任意计算模型都可以在 poly(log$n$) 时间内分解任意给定的整数 n。

16.11　证明：如果函数 $f: \mathbf{R}^* \to \{0, 1\}$ 可以用代数图灵机在 T 时间内被计算，则 f 存在一棵深度为 $O(T)$ 的代数计算树。

16.12　证明：如果允许 BSS 模型（在 $\mathbf{R}$ 上）以任意精度测试“$a>0$”是否成立，则 $\mathbf{P}_{/poly}$ 中的任意语言都可以在多项式时间内被判定。

337

第三部分

Computational Complexity: A Modern Approach

高级专题

第 17 章

Computational Complexity, A Modern Approach

计数复杂性

经验告诉我们，许多组合问题的解的存在性是容易获知的，但却不存在计算高效的方法来统计解的个数……这种现象在许多问题上是可以解释清楚的。

——瓦利安特(L. Valiant)，1979

复杂性类 **NP** 刻画了找出证明的难度。然而，在许多情形中，人们感兴趣的不再是单个的证明，而是证明的个数。本章研究复杂性类 #**P**(发音同"sharp P")，它刻画了上面的概念。

当人们需要在统计估计、统计物理学、网络设计和经济等各种领域中估计概率时，经常就会涉及计数问题。计数问题还出现于一个数学分支——计数组合学，这个数学分支旨在获取计数问题的封闭数学表达式。例如，1847 年，基希奥夫(Kirchoff)说明了网络阻抗是如何由网络中生成树的棵数来确定的，为此，他给出了一个计算生成树棵数的封闭数学公式，该公式只涉及一个行列式的计算。本章的结果表明，这种高效可计算的封闭数学公式在许多其他自然的计数问题上不太可能存在。

17.1 节非正式地介绍一些计数问题，并指出这些问题是如何出现在统计估计中的。我们将看到一种有趣的现象——有些时候，计数问题可能很难，但相应的判定问题却非常简单。

17.2 节定义复杂性类 #**P**，由此开始正式讨论计数问题。反映 #**P** 的精髓的计数问题是 #SAT，它要求在给定的布尔公式上统计满足性赋值的个数。然后，17.3 节介绍 #**P** 完全性，并证明"计算 0，1 矩阵的积和式"这一重要问题的 #**P** 完全性。

然后，我们研究复杂性类 #**P** 是否与本书前面介绍的各种概念之间有联系。17.4 节介绍户田[⊖](Toda)的一个出人意料的结果：#SAT 的任意神喻可以帮助我们在多项式时间内解决 **PH** 中的任何问题。有意思的是，尽管该结果的表述不涉及概率，但其证明过程却用到了概率论证法。

341

17.1 计数问题举例

计数问题的输出是一个数，而不再像判定问题一样输出 0 或 1。普通判定问题的计数形式在复杂性理论中也具有重要意义。下面给出了两个计数问题。

- #CYCLE 问题要求在给定的有向图 G 上计算 G 中简单环的个数。(注意，在简单环中，任意顶点不会出现两次。)该问题的判定形式是问给定的有向图 G 中是否存在简单环。这个判定问题是平凡的，它可以在线性时间内被求解。
- #SAT 问题要求在给定的布尔公式 φ 上计算 φ 的满足性赋值的个数。该问题的判定形式是问给定的布尔公式 φ 是否存在满足性赋值。当然，这个判定问题是 **NP** 完全问题。因此，计数问题通常比判定问题更难。

⊖ 日本姓氏。——译者注

17.1.1　计数问题与概率估计

计数问题通常出现在各种需要估算概率的场合中。

例 17.1　图可靠性问题的输入是一个含有 n 个顶点的有向图，且已知每个顶点失效的概率为 1/2，问题的输出是图中存在从顶点 1 到顶点 n 路径的概率。

经过简单的思考不难发现，在这种简单的顶点失效模型下，剩下的有效的图均匀地分布于原图的所有诱导子图中。因此，问题需要计算的答案实际上是

$$\frac{1}{2^n}\cdot(\text{其中存在从顶点 1 到顶点 } n \text{ 的路径的所有子图的个数})$$

因此，原始问题归约为一个子图计数问题，并且判定子图中是否存在从顶点 1 到顶点 n 的路径是一个平凡的问题。◀

例 17.2（贝叶斯网络中的极大似然估计）　假设某些数据是一个随机过程产生的，但其中部分数据丢失了。这种情形通常会出现在很多领域中，如机器学习和经济等领域。极大似然估计可以用来恢复缺失数据最可能的值。

最简单的数据生成模型是贝叶斯网络。我们这里只讨论贝叶斯网络的一个特别简单的例子。假设 $x_1,\cdots,x_n\in\{0,1\}$ 是 n 个隐变量，它们的取值通过独立地投掷 n 枚均匀硬币来产生，隐变量的取值我们是不知道的。我们能够知道的仅仅是 m 个可见的随机变量 $y_1,\cdots,y_m$ 的取值，其中每个随机变量是将 OR 操作作用到至多 3 个隐变量或隐变量的否定上得到的。假设我们观察到 $y_1,\cdots,y_m$ 的值都等于 1，我们现在希望估计 x_1 等于 1 的后验概率。

342

当然，复杂性理论学家立刻可以认识到，3 个文字的 OR 操作实际上是一个 3CNF 子句，因此待求解的问题可以重新表述为：给定 n 个变量上含有 m 个子句的 3CNF 布尔公式，问所有满足性赋值中使 $x_1=1$ 的占多大比例？可以证明，这个问题等价于 #SAT 问题（参见习题 17.1 和本章注记）。◀

例 17.3（统计物理学中的估计问题）　统计物理学中研究最广泛的模型是伊辛模型（Ising Model），该模型的引入源自二十世纪 20 年代楞次（Lenz）和伊辛（Ising）对铁磁性的研究。在伊辛模型的一个实例中，已知的参数包括：给定的 n 个阵点，任意无序阵点对 i，j 之间的相互作用能 V_{ij}，磁场强度 B，以及逆温 β。这些参数所定义的系统的一个格局是 n 个阵点各自的自旋状态，它可以表示为 n 个变量在 $\{+1,-1\}$ 上的一个赋值 σ。因此，系统可能的格局共有 2^n 种。格局 σ 的能量由如下的哈密顿（Hamilton）函数 $H(\sigma)$ 来定义：

$$H(\sigma)=-\sum_{\{i,j\}}V_{ij}\sigma_i\sigma_j-B\sum_i\sigma_i \tag{17.1}$$

我们感兴趣的是(17.1)式中的第一个求和项，每个阵点对都出现在该求和项中。这个求和项的重要性源自吉布斯分布（Gibbs Distribution）。由吉布斯分布可知，系统处于格局 σ 的概率正比于 $\exp(-\beta H(\sigma))$。也就是说，系统处于格局 σ 的概率等于 $1/Z\times\exp(-\beta H(\sigma))$，其中 Z 是归一化参数（也称为系统的划分函数）

$$Z=\sum_{\sigma\in\{1,-1\}^n}\exp(-\beta H(\sigma))$$

人们已经证明，划分函数的计算问题恰好等价于 #SAT 问题（参见本章注记）。◀

17.1.2　计数可能难于判定

#SAT 和 #CYCLE 的复杂性如何？显然，如果 #SAT 存在多项式时间算法，则

SAT∈**P**，进而 **P**=**NP**。那#CYCLE 呢？其相应的判定问题（亦即，在输入图上判定简单环的存在性）可以用广度优先搜索算法在线性时间内求解。下面的定理表明，问题的计数形式可能要难得多。

定理 17.4　*如果#CYCLE 存在多项式时间算法，则* **P**=**NP**。

证明　我们往证，如果#CYCLE 存在多项式时间算法，则 Ham∈**P**，其中 Ham 是一个 **NP** 完全问题——判定给定的有向图中是否存在哈密顿环（Hamilton Cycle）（哈密顿环是包含图的所有顶点的简单环）。给定含 n 个顶点的图 G，我们构造一个图 G 使得 G 存在哈密顿环当且仅当 G' 至少含有 n^2 个环。

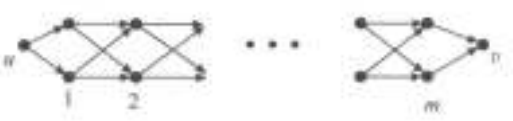


图 17-1　将 Ham 归约到#CYCLE。把 G 中的每条边 (u, v) 分别替换为图中的结构，由此得到 G'。G 中长度为 l 的每个环变成了 G' 中的 $(2^m)^l$ 个环

为了得到 G'，我们将 G 中的每条边 (u, v) 都分别替换为图 17-1 所示的结构。该结构是由 $m=n\log n$ 个层构成的一个无环有向图，因此，G' 中的环对应于 G 中的环。而且，在这个结构中存在 2^m 条从 u 到 v 的有向路径；因此，G 中长度为 l 的一个环可以得到 G' 中的 $(2^m)^l$ 个环。

343

注意，如果 G 中存在哈密顿环，则 G' 至少存在 $(2^m)^n > n^2$ 个环。另一方面，如果 G 中没有哈密顿环，则 G 中环的最大长度至多为 $n-1$，并且 G 中环的个数以 n^{n-1} 为上界。因此，G' 中环的个数至多为 $(2^m)^{n-1} \times n^{n-1} < n^2$。■

17.2　复杂性类#P

下面，我们尝试用复杂性类#**P** 来刻画前面提到的各种计数问题。注意，#**P** 中的函数的输出是自然数，而非 0/1。

定义 17.5　（#**P**）　*称函数 $f: \{0,1\}^* \to \mathbf{N}$ 属于#**P**，如果存在一个多项式 $p: \mathbf{N} \to \mathbf{N}$ 和一个多项式时间图灵机 M 使得在任意 $x \in \{0,1\}^*$ 上均有*

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\}|$$

当然，定义 17.5 还表明 $f(x)$ 可以表示为长为 $\mathrm{poly}(|x|)$ 的位串。

同 **NP** 的定义一样，我们也可以用非确定型图灵机来定义复杂性类#**P**。也就是说，#**P** 由如下的所有函数 f 构成：存在一个多项式时间的非确定型图灵机 M 使得 $f(x)$ 恰好等于 M 在输入 x 上的格局图 $G_{M,x}$ 中从初始格局到接受格局的路径的条数（每条这样的路径都简称为一条接受路径）。（格局图的定义见 4.1.1 节。）显然，17.1 节中所有的计数问题都属于这个复杂性类。

关于#**P** 的一个重大的待决问题是：这个复杂性类中的所有问题是否都是可以高效求解的。我们定义 **FP** 是从 $\{0,1\}^*$ 到 $\{0,1\}^*$（或者从 $\{0,1\}^*$ 到 **N**）的所有函数中可以用多项式时间的确定型图灵机计算的所有函数构成的类。显然，**FP** 中的函数类似于可高效计算的函数。也就是说，**FP** 类似于 **P**，不同之处在于 **FP** 中函数的输出不再是单个二进制位。于是，前面提出的问题也就是问#**P**=**FP** 是否成立。由于“计算证明的个数”至少难于“判定证明是否存在”，故有：如果#**P**=**FP**，则 **NP**=**P**。人们还不知道上述结论的逆命题（即，如果 **NP**=**P**，则#**P**=**FP**）是否成立。但我们现在可以知道的是：如果 **PSPACE**=**P**，则#**P**=**FP**。这是由于，计算证明的个数可以在多项式空间内完成。

17.2.1　复杂性类 PP：类似于 #P 的判定问题

类似于搜索问题，在研究计数复杂性时也可以只关注判定问题。计数判定问题的定义如下。

344

定义 17.6（PP）　称语言 L 属于 **PP**，如果存在一个多项式时间图灵机 M 和一个多项式 p：$\mathbf{N}\to\mathbf{N}$ 使得在任意 $x\in\{0,1\}^*$ 上均有

$$x\in L\Leftrightarrow|\{u\in\{0,1\}^{p(|x|)}:M(x,u)=1\}|\geqslant\frac{1}{2}\cdot 2^{p(|x|)}$$

也就是说，要使得 x 属于 L，不仅仅需要一个证明（但这正是 **NP** 语言的要求，参见定义 2.1），而是要求多数字符串都能提供证明。

引理 17.7　$\mathbf{PP}=\mathbf{P}\Leftrightarrow\#\mathbf{P}=\mathbf{FP}$。

证明　引理给出的非平凡蕴含关系是 $\mathbf{PP}=\mathbf{P}\Rightarrow\#\mathbf{P}=\mathbf{FP}$。令 f 是 #**P** 中的一个函数。那么，存在一个多项式时间的图灵机 M 使得：在任意输入 x 上，$f(x)$ 都等于满足 $M(x,u)=1$ 的字符串 $u\in\{0,1\}^*$ 的个数 $\#_M(x)$，其中 m 是某个多项式在 $|x|$ 上的取值，它刻画了 M 所采用的证明的长度。

对于采用 n-位证明的任意两个图灵机 M_0，M_1，下面的论述中将用 M_0+M_1 来表示采用 $(n+1)$-位证明的图灵机 M'，其中 $M'(x,bu)=M_b(x,u)$。于是，$\#_{M_0+M_1}(x)=\#_{M_0}(x)+\#_{M_1}(x)$。此外，对于 $N\in\{0,\cdots,2^n\}$，我们用 M_N 来表示“在输入 x，u 上输出 1 当且仅当字符串 u（视为自然数时）小于 N”的图灵机。显然，$\#_{M_N}(x)=N$。如果 **PP**=**P**，则我们可以在多项式时间内判定(17.2)式是否成立：

$$\#_{M_N+M}(x)=N+\#_M(x)\geqslant 2^n \tag{17.2}$$

因此，为了计算 $\#_M(x)$，只需用二分搜索找出满足(17.2)式的最小 N 值，这个 N 值必然等于 $2^n-\#_M(x)$。■

直观上讲，**PP** 对应于“计算 #**P** 函数中第一个不等于 0 的位”。也就是说，如果 $f(x)$ 的取值范围是 $[0,N-1]$，则 **PP** 问题需要判定 $f(x)\geqslant N/2$ 是否成立。我们也可以定义判定问题来计算 #**P** 函数中最后一个不等于 0 的位，这种判定问题称为 $\oplus$ **P**（参见定义 17.15）。

另一个相关的复杂性类是 **BPP**（参见第 7 章）。在 **BPP** 中，我们需要确保在每个输入上非确定型图灵机的接受路径的比例要么 $\geqslant 2/3$ 要么小于 1/3，并且要求在具体的输入上判定究竟是哪种情况发生。就这一点而言，**PP** 与 **BPP** 的差别很大，因为“输入属于 **PP** 语言”要求非确定图灵机的接受路径的比例 $\geqslant 1/2$，但“输入不属于 **PP** 语言”时非确定图灵机的接受路径的比例却可能 $\leqslant 1/2-\exp(-n)$。这两个比例之间不存在“鸿沟”[⊖]，这意味着：如果用随机抽样来判定输入是否属于 **PP** 语言，则需要做 $\exp(n)$ 次随机试验。相比之下，**BPP** 语言的抽样验证却很简单，我们甚至可以考虑将抽样算法替换为确定型算法，参见第 20 章。

17.3　#P 完全性

下面定义 #**P** 完全性。粗略地讲，如果函数 f 属于 #**P** 并且计算 f 的多项式时间算法

⊖ 原文是“gap”，意为“较大的间隙”，我们将它译为“鸿沟”，这同第 11 章的用法一样。——译者注

将得出 #P=FP，则称 f 是 #P 完全的。在正式定义 #P 完全性时，我们采用神喻图灵机（其定义见 3.4 节）。回顾一下，图灵机 M 以语言 $O\subseteq\{0,1\}^*$ 为神喻（或者说 M 可以通过神喻访问语言 O）指的是 M 可以在一个计算步骤内获得问题“$q\in O$ 吗？”的答案。我们将上
345 述概念推广到非布尔函数上。我们称图灵机 M 以函数 $f:\{0,1\}^*\to\{0,1\}^*$ 为神喻，如果 M 能够通过神喻访问语言 $O=\{\langle x,i\rangle: f(x)_i=1\}$。我们将同样的概念用于函数 $f:\{0,1\}^*\to\mathbf{N}$，这只需将函数值视为用二进制表示的位串。对于函数 $f:\{0,1\}^*\to\{0,1\}^*$，我们用 $\mathbf{FP}^f$ 表示以 f 为神喻的图灵机在多项式时间内能够计算的所有函数构成的集合。

定义 17.8　如果 f 属于 #P 并且任意 $g\in$ #P 都属于 $\mathbf{FP}^f$，则称 f 是 #P 完全的。

如果 $f\in\mathbf{FP}$，则 $\mathbf{FP}^f=\mathbf{FP}$。由此可以立刻得到下面的命题。

命题 17.9　如果 f 是 #P 完全的并且 $f\in\mathbf{FP}$，则 $\mathbf{FP}=$ #P。

3SAT，Ham，CLIQUE 等许多 NP 完全语言的计数形式自然都是 #P 完全语言。为了展示这一点，我们证明 #SAT 的 #P 完全性。

定理 17.10　#SAT 是 #P 完全的。

证明　考虑 2.3 节中的库克-勒维归约，它将任意 NP 语言 L 归约到 SAT。该归约是一个多项式时间可计算的函数 $f:\{0,1\}^*\to\{0,1\}^*$，它使得 $x\in L\Leftrightarrow f(x)\in$SAT 对任意 $x\in\{0,1\}^*$ 成立。归约的正确性的证明过程得到了更多的信息。在 2.3.6 节中，我们已经看到，证明过程实际上得到了一个勒维归约。也就是说，归约过程实际上将 $x\in L$ 的证明转换为了 $f(x)\in$SAT 的证明（亦即，一个满足性赋值），并且反之亦然（$f(x)\in$SAT 的证明可以转换为 $x\in L$ 的证明）。

事实上，为了得到证明本定理所需的归约，只需注意到从 x 的证明到 $f(x)$ 的满足性赋值的映射既是一对一的也是满的（亦即，它是一个双射）。因此，$f(x)$ 的满足性赋值的个数等于 x 的证明的个数。这种归约称为简约的。（更一般地，简约归约的定义允许证明之间的映射是 k 对 1 的或者 1 对 k 的。因此，在相差常数倍的意义下两个问题的证明的个数可以视为相同的。）■

下面的结果表明，某些 #P 完全问题的判定形式实际上属于 P。

17.3.1　积和式和瓦利安特定理

现在，我们研究另一个问题。$n\times n$ 矩阵 A 的积和式定义为

$$\mathrm{perm}(A)=\sum_{\sigma\in S_n}\prod_{i=1}^{n}A_{i,\sigma(i)} \tag{17.3}$$

其中 S_n 表示 n 个元素上所有置换构成的集合。回顾一下，行列式具有类似的表达式

346
$$\det(A)=\sum_{\sigma\in S_n}(-1)^{\mathrm{sign}(\sigma)}\prod_{i=1}^{n}A_{i,\sigma(i)}$$

相比之下，行列式的公式中多了一个“符号”项（参见 A.5）。计算公式相似并不意味着计算效率也等价。事实上，行列式可以在多项式时间内被计算，但积和式的计算要难得多，这一点将在下面证明。（第 16 章用另一种观点讨论了积和式的难计算性。）

积和式函数也可以用组合数学来解释。首先，假设矩阵 A 的所有元素都取自 $\{0,1\}$，则 A 可以视为二分图 $G(X,Y,E)$ 的邻接矩阵，其中 $X=\{x_1,\cdots,x_n\}$，$Y=\{y_1,\cdots,$

$y_n\}$，并且$\{x_i, y_j\} \in E$当且仅当$A_{i,j}=1$。对于任意置换σ，项$\prod_{i=1}^{n} A_{i,\sigma(i)}$等于 1 当且仅当$\sigma$是一个完美匹配(由 n 条边构成的集合，图的每个顶点恰好出现在其中的一条边上)。于是，如果 A 是 0，1 矩阵，则 perm(A)就是相应二分图 G 中的完美匹配的个数。注意，二分图中完美匹配的存在与否可以在多项式时间内判定。这说明，perm(A)计算问题属于#**P**。其次，如果 A 是{−1，0，1}矩阵，则 $\mathrm{perm}(A) = \left|\left\{\sigma: \prod_{i=1}^{n} A_{i,\sigma(i)} = 1\right\}\right| - \left|\left\{\sigma: \prod_{i=1}^{n} A_{i,\sigma(i)} = -1\right\}\right|$。因此，我们调用#SAT 神喻两次就可以计算出 perm(A)。最后，如果 A 中的元素是一般的整数(可能取负值)，则 perm(A)的组合学解释如下。将矩阵 A 视为含 n 个顶点的加权完全有向图(允许出现自环和权值 0)的邻接矩阵。每个置换σ对应图的一个环覆盖(亦即，原图的所有顶点和原图的一些边构成的一个子图，其中每个顶点的出度和入度都等于 1)，它必然可以分解为一组无公共顶点的环。环覆盖的权值定义为其中所有边的权值之积。这样，perm(A)就等于相应图上所有环覆盖的权值之和。基于这一观察，我们可以证明积和式计算问题属于 $\mathbf{FP}^{\#\mathrm{SAT}}$(参见习题 17.2)。

二十世纪 70 年代得到的如下定理曾让研究者们大感意外，因为该定理表明：如果 perm$\in$**FP**，则 **P**＝**NP**。因此，除非 **P**＝**NP**，积和式的计算将远远难于行列式的计算。

定理 17.11　(瓦利安特定理(Valiant Theorem)[Val79b])　0，1 矩阵的积和式计算是#**P**完全的。

定理 17.11 的证明需要构造一种非常精巧的结构。作为预备，我们先介绍一种简单的思想。

约定：本章其余部分在绘制图结构时，均假设底图(underlying graph)是完全有向图，绘制过程略去所有权值为 0 的边，因为在考虑环覆盖时这些边可以忽略。未标记的边的权值为+1。有时，我们还允许从一个顶点到另一个顶点之间使用平行边(可能具有不同的权值)，但平行边不会出现在积和式的讨论中。此时，从顶点 i 到顶点 j 只有一条权值为 $A_{i,j}$的边。在这种约定下，由于我们绘图的目的是描述归约，因此，为了得到最终的归约，只需在绘制的结构上将权值为 w 的平行边替换成一条长度为 2 的路径(路径的两条边的权值分别为 1 和 w)。而这只需在每条平行边中部添加一个顶点(不与其他顶点连接)即可。

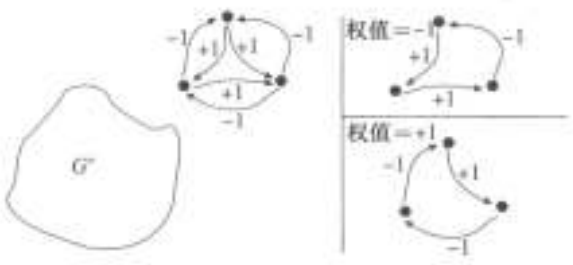


图 17-2　上面的图由两个无公共顶点的子图构成，一个子图是 G'而另一个子图是如图所示的"结构"。该结构只有两个非零的环覆盖，这两个环覆盖的权值分别为−1和 1。由于对 G'的每个权值为 w 的环覆盖，整个图 G 中将存在两个权值分别为$+w$和$-w$的环覆盖，所以整个图的所有环覆盖的权值之和等于 0

例 17.12　考虑图 17-2 中的图。尽管我们不知道子图 G'是什么样子，我们却能够证明整个图的积和式等于 0。事实上，对于 G'的每个权值为 w 的环覆盖，覆盖另外三个顶点的环只有两个，这两个环的权值分别为+1和−1。由于整个图的任意非零环覆盖都由 G'的一个环覆盖和覆盖另三个顶点的环覆盖之一来构成。因此，G 的所有环覆盖的权

值之和等于 0。 ◀

瓦利安特定理(定理 17.11)的证明　我们将 #P 完全问题 #3SAT 归约到 perm。给定 n 个变量上含有 m 个子句的 3CNF 布尔公式 ϕ，我们先说明如何构造一个含有负数的整数矩阵(或者等价地说，一个加权有向图 G')使得 $\mathrm{perm}(G')=4^m\cdot(\#\phi)$，其中 $\#\phi$ 表示 ϕ 的满足性赋值的个数。然后，再说明如何由 G' 来构造权值为 0，1 的有向图 G 使得积和式 $\mathrm{perm}(G)$ 可以用来计算 $\mathrm{perm}(G')$。

主要思想是让构造过程在 G' 中产生两类环覆盖，第一类环覆盖对应于 ϕ 的满足性赋值，第二类环覆盖则不对应于 ϕ 的满足性赋值(下面的论述将准确地实现这种想法)。采用例 17.12 中类似的推理方法，我们将通过负权值使得第二类环覆盖对积和式的贡献相互抵消。另一方面，我们还将证明，第一类环覆盖中的每一个环覆盖对积和式的值贡献 4^m，进而 $\mathrm{perm}(G')=4^m\cdot(\#\phi)$。

为了从 ϕ 构造 G'，我们对图 17-3 所示的三种结构进行组合。每个变量都有一个变量结构，每个子句有一个子句结构，另外还有一种称为 XOR 结构的构件用于连接变量结构和子句结构。这三种结构如图 17-3 所示。

XOR 结构。假设我们有一个加权有向图 H，并希望 H 的某两条边 $\overrightarrow{uu'}$，$\overrightarrow{vv'}$ 中恰有一条边能够出现在对积和式有贡献的任意环覆盖中。为此，我们将 H 中的边 $\overrightarrow{uu'}$，$\overrightarrow{vv'}$ 替换为图 17-3 给出的 XOR 结构，由此构造出一个新图 H'。

对于 H 的每个权值为 w 的环覆盖，如果 $\overrightarrow{uu'}$，$\overrightarrow{vv'}$ 之一恰好出现于在这个环覆盖中，则这个环覆盖对应于 H' 的一族环覆盖(其中每个环覆盖要么从顶点 u 进入 XOR 结构并从 u' 离开这个结构，要么从 v 进入 XOR 结构并从 v' 离开这个结构)，这族环覆盖的权值之和为 $4w$。利用例 17.12 中类似的推理方法可知，H' 的其他环覆盖的权值之和等于 0(参见习
348 题 17.3)。

变量结构。变量结构包含内部边(不与图的其余部分相连的边)和外部边(通过 XOR 结构与图的其余部分相连的边)。外部边又分为"真"边和"假"边。变量结构只有两种环覆盖，它们分别对应于变量在 0 和 1 上的两种赋值。变量取 1 对应于"让变量结构中的所有真外
349 部边参与环覆盖并让变量结构中的其余顶点用自环来覆盖"。类似地，变量取 0 则对应于"让变量结构中的所有假外部边参与环覆盖并让变量结构中的其余顶点用自环来覆盖"。变量结构的每条外部边都关联到该变量出现的一个子句上——在"真"外部边关联的子句上，该变量以肯定形式出现(且不含否定形式)；在"假"外部边关联的子句上，该变量以否定形式出现(且不含肯定形式)。

子句结构。子句结构仅有 4 个顶点，它的"外部边"都连接到子句结构之外的其他顶点上。具体地讲，子句结构的每条外部边都通过 XOR 结构与一个变量结构的外部边连接，且这个变量结构对应的变量是出现于该子句中的变量之一。覆盖子句结构中 4 个顶点的任意环覆盖至少需要忽略子句结构的一条外部边。而且，给定三条外部边的任意真子集，存在唯一权值为 1 的环覆盖来包含真子集中的所有边。

G' 的整体构造如图 17-3 的下部所示。如果子句 C 包含变量 x，则将"C 的子句结构中 x 对应的外部边"与"x 的变量结构中 C 对应的(真)外部边"通过 XOR 结构连接起来。如果 C 包含变量 x 的否定形式，则在上述连接过程中与 C 对应的 x 中的边为假外部边。

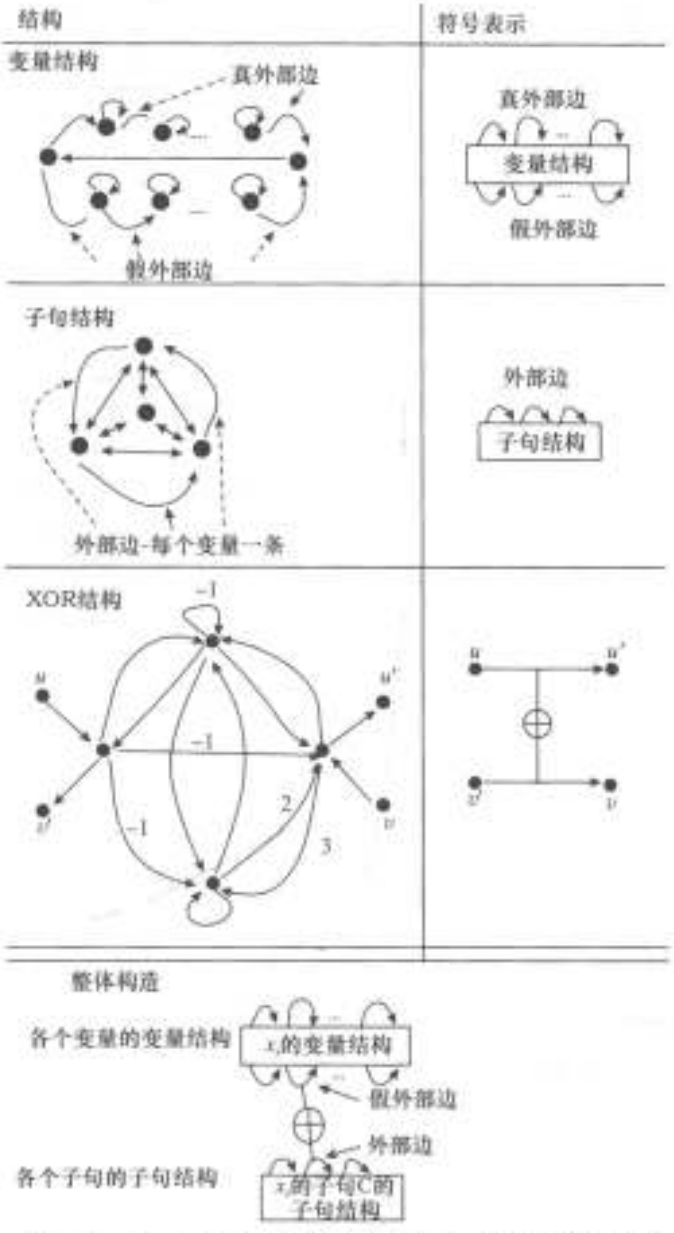


图 17-3　证明瓦里安特定理(定理 17.11)所需的各种结构

下面对G'的构造进行分析。注意，因为每个变量结构都只有两个环覆盖，它们分别对应于将变量赋值为 0 和 1。因此，所有变量 x_1，…，x_n的所有赋值可以一对一地映射为覆盖G'中所有变量结构的环覆盖上。给定赋值 $\boldsymbol{x}$，令$\mathcal{C}_x$是G'的所有如下环覆盖构成的集合：环覆盖根据 $\boldsymbol{x}$ 覆盖了G'的所有变量结构。也就是说，如果 $x_i=1$ 则环覆盖使用了 x_i的变量结构中的“真”外部边，如果 $x_i=0$ 则环覆盖使用了 x_i的变量结构中的“假”外部边。令 w_x是$\mathcal{C}_x$中所有环覆盖的总权值。因此，我们只需证明：如果 $\boldsymbol{x}$ 是一个满足性赋值，则 $w_x=4^{3m}$；否则，$w_x=0$。

事实上，根据 XOR 结构的性质和子句结构与变量结构之间的连接方式可知：如果 $x_i=1$，则$\mathcal{C}_x$的任意环覆盖必然弁用“x_i以肯定形式出现的子句”所对应的子句结构中的外部边，而必然采用“x_i以否定形式出现的子句”所对应的子句结构中的外部边。类似地，如果

$x_i=0$，则C_i的任意环覆盖也必然弃用和必须采用相应的外部边。（准确地说，不具备上述性质的环覆盖不会对积和式的取值造成影响。）又由于子句结构的任意环覆盖至少弃用该子句结构中的一条外部边，因此，除非每个子句都有一个文字在 x 中取值为"真"（亦即，除非 x 满足 ϕ），C_i中所有环覆盖的总权值将为 0。如果 x 确实满足 ϕ，则C_i中所有环覆盖的总权值为 4^{3m}。这是由于，x 唯一地确定了各个子句结构的覆盖，而这些覆盖恰好又通过 $3m$ 个 XOR 结构连接在一起。因此，$\mathrm{perm}(G')=4^{3m}(\neq\phi)$。

归约到 0，1 矩阵的积和式

接下来的变形分两个步骤完成。第一步，我们先构造一个权值属于{−1，0，1}的图 G''使得它的积和式等于 G'的积和式。第二步，我们删除负权值，构造一个图 G 使得它含有充分的信息来计算 $\mathrm{perm}(G')=\mathrm{perm}(G'')$。上述变形使得图的顶点个数增大 $O(nL^2\log n)$ 倍，其中 L 是描述 G'中所有权值所需的位串总长度。

注意，权值为 2 的幂（不妨设 2^k）的边可以替换为长度为 k 的路径，路径中的每条边的权值等于 2（路径中新引入的顶点不与图的其余部分连接）。类似地，权值为 2^k+2^l 的边可
350 以替换为权值分别为 2^k 和 2^l 的两条平行路径。结合上述两个观察结果，权值不为 2 的幂的每条边都可以根据权值的二进制形式将它替换为一组平行路径，这些平行路径上新引入的顶点的个数不超过原权值的二进制位串长度的平方。上述变换得到了一个权值属于{−1，0，1}的图 G''，新引入的顶点至多为 $O(L^2)$个。

我们用模算术来删除负权值。对于边权值等于±1 的 n-顶点图，其积和式 x 必然属于$[-n!,+n!]$。因此，只需计算积和式对 2^m+1 的余数，其中 $m=n^2$。又由于 $-1\equiv 2^m(\mathrm{mod}\ 2^m+1)$，因此将所有权值为 −1 的边重新赋予权值 2^m，则图的积和式模 2^m+1 的余数不会改变。用前一个自然段给出的方法，这种权值为 2^m的边也可以转换为一组平行路径，其中每条边的权值都等于 1，并且新引入的顶点个数为 $O(m)=O(n\log n)$。这样，我们就得到了一个权值为 0，1 的图 G，它的积和式可以用来计算原积和式（亦即，取 G 的积和式模 2^m+1 的余数）。整个变换过程至多新增 $O(nL^2\log n)$个顶点。 ■

17.3.2　#P 问题的近似解

由于很难计算 #P 完全问题的准确解，故一个自然的问题是能否在下面定义的意义下近似地计算证明的个数。

定义 17.13　设 $f:\{0,1\}^*\to\mathbf{N}$ 且 $\alpha<1$，如果算法 A 使得 $\alpha f(x)\leqslant A(x)\leqslant f(x)/\alpha$，则称 A 是 f 的一个 α-近似算法。

近似求解各种 #P 问题的难度不尽相同。将某些问题近似到任意常数因子 $\alpha>0$ 是 NP 难的（参见习题 17.4）。另有一些问题，如 0/1 积和式，存在完全多项式随机近似模式（FPRAS），这种近似算法在任意给定的 ε，$\delta>0$ 的条件下都能在 $\mathrm{poly}(n,\log 1/\delta,\log 1/\varepsilon)$ 时间内以 $1-\delta$ 概率计算出问题的一个$(1-\varepsilon)$-近似解。也就是说，允许算法得到错误解的概率为 δ。计数问题的这种近似解能够满足大多数应用的需要，尤其当计数的目的是为了计算某种随机事件发生的概率时（例如，例 17.1 讨论的有向图可靠性问题就属于这种情形）。有意思的是，如果 P＝NP，则任意 #P 问题均存在完全多项式随机近似模式（FPRAS），参见习题 17.5。不仅如此，事实上，此时任意 #P 问题均存在确定型多项式时间近似模式。

现在，我们解释积和式近似算法背后的基本思想，当然这种思想也适用于大量其他

#P完全问题的近似算法。这里，我们只梗概地进行讨论，本章注记给出了更多的参考文献。

为#P 近似算法奠定基础的一个重要结果源自杰鲁姆(Jerrum)，瓦里安特(Valiant)和瓦兹拉尼(Vazirani)[JVV86]。该结果表明，在相当广泛的条件下，下述两个计算任务之间存在着密切联系(指的是它们可以在多项式时间内互相归约)

1. 找出集合 S 的大小的近似公式；

2. 找出一个高效算法来均匀(或近似均匀)随机地产生 S 中的元素；

建立这种联系的基本思想是利用我们曾在第 2 章见过的向下自归约的计数形式。 351

假设我们用随机抽样来实现近似计数，而 S 是 $\{0,1\}^*$ 的一个子集。令 $S=S_0 \cup S_1$，其中 S_b 是由 S 中第一个位等于 b 的所有字符串构成的子集。随机抽取 S 中的少量元素，我们就能将 $p_1=|S_1|/|S|$ 估计到一个合理的精度。然后，我们固定元素的第一个位，递归应用上述算法估计 $|S_1|$ 的大小，并将估计结果与 $1/p_1$ 相乘就可以估计 $|S|$。

如果要用近似计数来产生 S 的一个随机样本，我们同样可以逐位地进行处理，这只需将前一自然段的过程逆转即可。首先，我们估算 $|S_1|$，$|S|$，然后用它们的比值来估计 p_1。投掷一枚偏斜度为 p_1 的硬币就可以产生一个位 b(亦即，如果硬币头面朝上，则 $b=1$；否则，$b=0$)。然后，用 b 作为随机样本的第一个位，再递归调用同样的算法从 S_b 抽取随机样本。

要点在于"随机从 S 抽样"足以实现"近似计数"。对 S 进行随机抽样的所有算法都使用了马尔可夫链-蒙特卡罗方法。该方法先在 S 上定义一个 d-正则有向图，然后在图上进行随机游走。由于图对某个 d 而言是 d-正则的，随机游走的静态分布是 S 上的均匀分布。在恰当的条件下，在图的某个扩张(建立这个扩张正是这种论证方法要完成的工作)上随机游走将变成"平稳的(mixed)"。此时，随机抽样趋于均匀分布。

17.4　户田定理：$\mathbf{PH} \subseteq \mathbf{P}^{\#\mathrm{SAT}}$

在 20 世纪 80 年代，一个重要的待决问题是比较多项式分层 **PH** 和计数问题构成的复杂性类#**P** 之间的相对能力。二者都是 **NP** 的自然推广，前者用交错量词这一特征来定义，后者用计算证明的个数这一特征来定义。这两个特征似乎很难直接比较。因此，当户田 1989 年证明下面的定理后，人们着实大感意外。

定理 17.14　(户田定理(Toda Theorem)[Tod91])　$\mathbf{PH} \subseteq \mathbf{P}^{\#\mathrm{SAT}}$。

也就是说，用特定的#**P** 完全问题作为神喻，我们将能够高效地求解多项式分层中的任意问题。

注意，即使没有户田定理，我们仍然知道：如果#**P**=**FP**，则 **NP**=**P**，进而 **PH**=**P**。但是，这并不意味着：用#SAT 作为神喻时我们将能够在多项式时间内求解 **PH** 中的任意问题。例如，户田定理蕴含的一个结论是，#SAT 的亚指数时间(即 $2^{o(n)}$ 时间)算法将得到 **PH** 中任意问题的亚指数时间算法；但是，SAT 的亚指数时间算法却无法得出 **PH** 中任意问题的亚指数时间算法。

为了证明户田定理，我们先考虑具有奇数个满足性赋值的布尔公式。我们依赖如下定义的复杂性类来刻画这种布尔公式。

定义 17.15　语言 L 属于$\oplus$**P**(读作"parity P")当且仅当存在一个多项式时间的非确定型图灵机 M 使得：$x \in L$ 当且仅当 M 在 x 上的接受路径的条数是奇数。 352

正如定理 17.10 的证明过程所说，**NP** 完全性的标准归约是简约归约，这就意味着如下定义的⊕ SAT(在多对一卡普(Karp)归约下)是⊕ **P** 完全的。

定义 17.16（⊕量词和⊕ SAT）　量词⊕如下定义：在 n 个变量的布尔公式 φ 上，如果使得 $\varphi(x)$ 为真的变量赋值 x 的个数是奇数，则称 $\oplus_{x\in\{0,1\}^n}\varphi(x)$ 为真。语言⊕ SAT 是使得 $\oplus_{x\in\{0,1\}^n}\varphi(x)$ 为真的所有量化布尔公式构成的集合，其中 φ 是未量化的布尔公式(不一定是 CNF 范式)。

*注意，如果用 1 表示真，用 0 表示假，则 $\oplus_{x\in\{0,1\}^n}\varphi(x)=\sum_{x\in\{0,1\}^n}\varphi(x)\pmod 2$。此外，还可以注意到 $\oplus_{x\in\{0,1\}^n}\varphi(x)=\oplus_{x_1\in\{0,1\}}\cdots\oplus_{x_n\in\{0,1\}}\varphi(x_1,\cdots,x_n)$。

⊕ **P** 中的判定问题对应于判定 #**P** 函数的解中最后一个位是否等于 0。不难想象，这个类的刻画能力不会太强。例如，人们还不清楚能否将 **NP** 归约到⊕ **P**。但出人意料的是，户田定理证明过程的第一部分首先将 **PH** 随机归约到⊕ SAT。然后，第二部分再巧妙地“消除归约过程中的随机因素”(参见 17.4.4 节)。

引理 17.17（从 **PH** 到⊕ SAT 的随机归约）　设 $c\in\mathbf{N}$ 是一个常数。存在如下的概率型多项式时间算法 A，它的输入是参数 m 和规模为 n 的量词交错次数为 c 的任意量化布尔公式 Ψ，算法的运行时间为 $\mathrm{poly}(n,m)$ 并且满足

$$\Psi\text{ 是真}\Rightarrow\Pr[A(\Psi)\in\oplus\mathrm{SAT}]\geqslant 1-2^{-m}$$

$$\Psi\text{ 是假}\Rightarrow\Pr[A(\Psi)\in\oplus\mathrm{SAT}]\leqslant 2^{-m}$$

当然，为了将 **PH** 归约到⊕ SAT，我们最好先大致勾勒出将 **NP** 归约到⊕ SAT 的办法(虽然人们还不清楚这种办法是否可行)。为此，我们需要借助具有唯一满足性赋值的布尔公式。

17.4.1　过渡：具有唯一解的布尔满足性问题

假设某人给我们一个布尔公式并承诺：该公式要么不存在满足性赋值，要么存在唯一的满足性赋值。如果把一些经典的数学问题用满足性表达出来，则会得到这种具有唯一解的布尔公式。(例如，第 9 章讨论了数论中的离散对数问题，它是某些加密方案的基础，用满足性来表示离散对数问题就会得到这种布尔公式。)令 USAT 是具有唯一满足性赋值的所有布尔公式构成的语言。求解这个特殊问题仍然很难吗？也就是说，如果给定的布尔公式属于 USAT 则回答“是”，如果它属于 $\overline{\mathrm{SAT}}$ 则回答“否”，如果是其他情况则任意给出一个答案，这仍然很难吗？瓦利安特(Valiant)和瓦兹拉尼(Vazirani)给出的如下结论表明：如果 USAT 存在如上所述的多项式时间算法，则 **NP**=**RP**。在上世纪 80 年代，这个结果曾让多数研究者大感意外。

353

定理 17.18（瓦利安特-瓦兹拉尼定理(Valiant-Vazirani Theorem)[VV86]）　存在概率型多项式时间算法 f 使得对于任意 n 变量布尔公式 φ 有

$$\varphi\in\mathrm{SAT}\Rightarrow\Pr[f(\varphi)\in\mathrm{USAT}]\geqslant\frac{1}{8n}$$

$$\varphi\notin\mathrm{SAT}\Rightarrow\Pr[f(\varphi)\in\mathrm{SAT}]=0$$

值得强调的是，定理第二个结论得出的不是 $f(\varphi)\notin\mathrm{USAT}$，而是 $f(\varphi)\notin\mathrm{SAT}$。

定理 17.18 的证明将用到下面关于两两独立哈希函数(参见 8.2.2 节)的引理。

引理 17.19（瓦利安特-瓦兹拉尼引理(Valiant-Vazirani Lemma)）　如果 $\mathcal{H}_{n,k}$ 是从 $\{0,1\}^n$ 到 $\{0,1\}^k$ 的一族两两独立的哈希函数，并且 $S\subseteq\{0,1\}^n$ 满足 $2^{k-2}\leqslant|S|\leqslant 2^{k-1}$，那么，

$$\Pr_{h\in_R\mathcal{H}_{n,k}}[\text{存在唯一 } x\in S \text{ 满足 } h(x)=0^k]\geqslant\frac{1}{8}$$

证明　对任意 $x\in S$，令 $p=2^{-k}$ 表示随机选取 $h\in_R\mathcal{H}_{n,k}$ 时 $h(x)=0^k$ 的概率。注意，对于任意 $x\neq x'$，$\Pr[h(x)=0^k\wedge h(x')=0^k]=p^2$。令随机变量 N 表示满足 $h(x)=0^k$ 的 $x\in S$ 的个数。注意，$E[N]=|S|p\in\left[\frac{1}{4},\frac{1}{2}\right]$。由容斥原理可知，

$$\Pr[N\geqslant 1]\geqslant\sum_{x\in S}\Pr[h(x)=0^k]-\sum_{x\neq x'\in S}\Pr[h(x)=0^k\wedge h(x')=0^k]=|S|p-\binom{|S|}{2}p^2$$

再由合并界限，我们得到 $\Pr[N\geqslant 2]\leqslant\binom{|S|}{2}p^2$，因此

$$\Pr[N=1]=\Pr[N\geqslant 1]-\Pr[N\geqslant 2]\geqslant|S|p-2\binom{|S|}{2}p^2\geqslant|S|p-|S|^2p^2\geqslant\frac{1}{8}$$

其中最后一个不等式是利用 $\frac{1}{4}\leqslant|S|p\leqslant\frac{1}{2}$ 得到的。 ■

现在，我们证明定理 17.18。

定理 17.18 的证明　给定 n 个变量上的布尔公式 φ，从 $\{2,\cdots,n+1\}$ 中随机选取一个 k，再随机选取哈希函数 $h\in_R\mathcal{H}_{n,k}$。考虑论断

$$\exists_{x\in\{0,1\}^n}\varphi(x)\wedge(h(x)=0^k)\tag{17.4}$$

如果 φ 是不可满足的，则(17.4)式是假，因为没有 x 满足 $\varphi(x)$。如果 φ 是可满足的，则存在唯一 x 满足(17.4)式的概率至少为 $1/8n$。这是由于，如果 S 是 φ 的所有满足性赋值构成的集合，则 k 满足 $2^{k-2}\leqslant|S|\leqslant 2^{k-1}$ 的概率为 $1/n$。在 k 满足 $2^{k-2}\leqslant|S|\leqslant 2^{k-1}$ 的条件下，存在唯一 x 满足 $\varphi(x)\wedge(h(x)=0^k)$ 的概率至少为 $1/8$。

上一个自然段描述的基本思想可以如下实现。先调用库克-勒维归约中的变换过程将(17.4)式中的 $\exists_{x\in\{0,1\}^n}(h(x)=0^k)$ 改写为一个布尔公式 $\tau(x,y)$ 使得 $h(x)=0^k$ 当且仅当存在唯一的 y 使得 $\tau(x,y)=1$，其中 $x\in\{0,1\}^n$，$y\in\{0,1\}^m$，$m=\mathrm{poly}(n)$。注意，y 是库克-勒维归约中表示图灵计算过程需要的变量。[⊖] 最后，输出布尔公式 354

$$\Psi=\varphi(x)\wedge\tau(x,y)$$

其中 x，y 是变量。 ■

17.4.2　⊕的性质和对 NP、coNP 证明引理 17.17

引理 17.17 的归约只允许以极小的概率 2^{-m} 失效，其中 m 是任意的。如果我们允许较高的失效概率，则瓦利安特-瓦兹拉尼定理本身就给出了从 **NP** 到⊕SAT 的平凡归约。具体而言，定理 17.18 的第一部分结论表明，归约所得的布尔公式具有唯一满足性赋值，而 1 是奇数；第二部分结论则表明，归约所得的布尔公式不存在满足性赋值，而 0 是偶数。由此，立刻得到定理 17.18 的如下推论。

推论 17.20（瓦利安特-瓦兹拉尼推论）　存在概率型多项式时间算法 A 使得对于任意 n 变量布尔公式 φ 有

$$\varphi\in\mathrm{SAT}\Rightarrow\Pr[A(\varphi)\in\oplus\mathrm{SAT}]\geqslant\frac{1}{8n}$$

[⊖] 在哈希函数的某些实现方式上，如习题 8.4 给出的实现方式，我们可以直接构造出所需的布尔公式，既不需引入变量 y，也无需调用库克-勒维归约。

$$\varphi \notin \mathrm{SAT} \Rightarrow \Pr[A(\varphi) \in \oplus\,\mathrm{SAT}] = 0$$

能否将瓦利安特定理到 USAT 的归约概率从 $1/8n$(定理 17.18)提升至某个常数(比方说 1/2)，目前这仍是一个待决问题。然而，如果归约的目标语言是⊕ SAT，上述归约概率的提升是有望实现的。事实上，⊕ SAT 的表达能力确实比 USAT 强。下面，我们讨论⊕量词的几个事实。

对于 n 个变量上的布尔公式 φ，令 $\#(\varphi)$ 表示 φ 的满足性赋值的个数。分别给定变量 $x \in \{0,1\}^n$ 和 $y \in \{0,1\}^m$ 上的布尔公式 φ 和 Ψ，我们可以在多项式时间内构造出 $n+m$ 个变量上的一个布尔公式 $\varphi \cdot \Psi$ 和 $\max(n,m)+1$ 个变量上的一个布尔公式 $\varphi+\Psi$ 使得 $\#(\varphi \cdot \Psi) = \#(\varphi) \cdot \#(\Psi)$ 且 $\#(\varphi+\Psi) = \#(\varphi) + \#(\Psi)$。事实上，取 $(\varphi \cdot \Psi)(x,y) = \varphi(x) \wedge \Psi(y)$ 和 $(\varphi+\Psi)(z) = ((z_0=0) \wedge \varphi(z_1,\cdots,z_n)) \vee ((z_0=1) \wedge (z_{n+1}=0) \wedge \cdots \wedge (z_m=0) \wedge \Psi(z_1,\cdots,z_m))$，其中 $m<n$。对于布尔公式 φ，我们引入记号 $\varphi+1$ 来表示布尔公式 $\varphi+\Psi$，其中 Ψ 是某个仅有一个满足性赋值的布尔公式。

由于乘积是偶数当且仅当一个因数是偶数，并且任意数加 1 之后其奇偶性必然发生变化，因此对于上述的布尔公式 φ 和 Ψ 有

$$(\underset{x}{\oplus}\varphi(x)) \wedge (\underset{y}{\oplus}\Psi(y)) \Leftrightarrow \underset{x,y}{\oplus}(\varphi \cdot \Psi)(x,y) \tag{17.5}$$

355

$$\neg \underset{x}{\oplus}\varphi(x) \Leftrightarrow \underset{x,z}{\oplus}(\varphi+1)(x,z) \tag{17.6}$$

$$(\underset{x}{\oplus}\varphi(x)) \vee (\underset{y}{\oplus}\Psi(y)) \Leftrightarrow \underset{x,y,z}{\oplus}((\varphi+1) \cdot (\Psi+1)+1)(x,y,z) \tag{17.7}$$

观察结果(17.6)式的含义是，⊕ **P** 在补操作下是封闭的。也就是说，可以为任意布尔公式 φ 在多项式时间构造另一个布尔公式 Ψ 使得 $\neg \underset{x}{\oplus}\varphi(x)$ 等价于 $\underset{y}{\oplus}\Psi(y)$。观察结果(17.5)式和(17.7)式的含义是，分别将 AND 操作和 OR 操作作用到多项式个⊕ SAT 实例上，所得布尔公式也可以在多项式时间内转换为一个等价的⊕ SAT 实例。

现在，我们对 **NP** 和 **coNP** 证明引理 17.17，亦即，我们证明在布尔公式 φ 仅含一个量词∀(或∃)时引理 17.17 成立。事实上，只需给出一个归约过程将 **NP** 归约到⊕ **P**。因为，同一个归约过程也将 **coNP** 归约到 $\overline{\oplus\,\mathbf{P}}$，又由⊕ **P** 在补操作下的封闭性可知⊕ **P**=$\overline{\oplus\,\mathbf{P}}$。因此，该归约过程也将 **coNP** 归约到⊕ **P**。

证明　(引理 17.17，φ 只含一个∃量词的情形)假设 φ 是一个用单个存在量词∃量化的布尔公式。将 φ 归约为⊕ SAT 实例的思想很简单：执行推论 17.20 给出的归约 $R=O(mn)$ 遍，每遍执行都会得到一个⊕公式。然后将 OR 操作作用到所得的所有⊕公式上。如果原公式 φ 是可满足的，则所得公式为真的概率至少为 $1-(1-(1/8n))^R = 1-2^{-n}$；如果原公式 φ 是不可满足的，则所得公式绝不可能为真。最后，运用观察结果(17.7)式 R 次，将所得公式转换为一个⊕公式，但这可能会导致公式的大小扩大一个多项式因子。 ■

17.4.3　引理 17.17 的证明：一般情形

一般情况的证明通过对 c 做数学归纳法来完成，其中 c 是量词在 φ 中交错的次数。基本情况 $c=1$(即 **NP** 和 **coNP**)已经在 17.4.2 节被证明。为了证明一般情况，我们需要瓦利安特-瓦兹拉尼引理的更抽象的形式。我们发现，在这种抽象形式中，归约过程将不再查看它所要处理的布尔公式，因此它对任意的布尔公式都成立。(用第 1 章的术语来说，这

种瓦利安特-瓦兹拉尼引理是散漫的。）

引理 17.21（瓦利安特-瓦兹拉尼引理的散漫形式）　存在概率型多项式时间算法在给定的输入 1^n 上产生一个布尔公式 $\tau(x, y)$（其中 x 是 n 个布尔变量构成的向量，y 也是布尔变量向量）使得任意布尔函数 $\beta: \{0,1\}^n \to \{0,1\}$ 均满足

$$\exists_{x_1}\beta(x_1) \Rightarrow \Pr\left[\bigoplus_{x_1,y}\tau(x_1,y) \wedge (\beta(x_1)=1)\right] \geqslant \frac{1}{8n} \tag{17.8}$$

$$\neg\exists_{x_1}\beta(x_1) \Rightarrow \Pr\left[\bigoplus_{x_1,y}\tau(x_1,y) \wedge (\beta(x_1)=1)\right] = 0 \tag{17.9}$$

现在，我们就能够证明引理 17.17 了。

证明　（引理 17.17）设 φ 中量词交错出现了 c 次。我们在 17.4.2 节中已经知道，$\oplus$SAT在补操作下是封闭的。因此，不失一般性，我们可以假设 φ 的第一个量词是 $\exists$。 356
于是，

$$\varphi = \exists_{x_1}\Psi(x_1)$$

其中 $\Psi(x_1)$ 是量词至多交错出现 $c-1$ 次的一个量化布尔公式，且 x_1 中的所有变量在 $\Psi(x_1)$ 中都是自由变量。假设 x_1 包含 n 个布尔变量。根据归纳假设，存在一个随机归约使得：对于 x_1 的每种取值，随机归约都产生一个$\oplus$ SAT 公式 $\beta(x_1)=\bigoplus_z \rho(z, x_1)$，并且 $\beta(x_1)$ 等价于 $\Psi(x_1)$ 的概率至少为 $1-2^{-(n+2)}$。现在，用独立的随机位串执行引理 17.21 中的归约 $K=O(mn)$ 遍，将每遍产生的布尔公式依次记为 $\tau_1(x_1, y)$，$\tau_2(x_1, y)$，…，$\tau_K(x_1, y)$。考虑公式

$$\alpha = \bigvee_{j=1}^{K}\left(\bigoplus_{x_1,y}\tau_j(x_1,y) \wedge \beta(x_1)\right)$$

如果 $\exists_{x_1}\beta(x_1)$ 为真，则根据引理 17.21 可知 $\Pr[\alpha \text{ 为真}] \geqslant 1-(1-1/8n)^K = 1-2^{-(8n)}$。反之，如果 $\exists_{x_1}\beta(x_1)$ 为假，则 $\Pr[\alpha \text{ 为真}]=0$。

最后，注意，由归纳假设可知 $\beta(x_1)$ 是一个$\oplus$ SAT 实例，故 α 也可以用 17.4.2 节中的方法转换为一个$\oplus$ SAT 实例（规模扩大多项式倍）。整个归约中可能会引入错误的两个地方是：(a)将 $\Psi(x_1)$ 转换为等价的$\oplus$ SAT 实例，由归纳假设可知，这个过程失效的概率是 $2^{-(n+2)}$；(b)当我们换掉 $\exists_{x_1}$ 来构造 α 时也会引起错误，此时的失效概率仍然是 $2^{-(n+2)}$。因此，整个归约过程的失效概率为 $2\times 2^{-(n+2)}$，该概率小于 2^{-n}。 ■

17.4.4　第二步：转换为确定型归约

现在，对于引理 17.17 中的随机归约，我们将消除它的随机性，由此完成卢田定理（定理 17.14）的证明。下面的确定型归约是我们的关键工具。

引理 17.22　存在（确定型）多项式时间变换 T 使得：任意布尔公式 α 和布尔公式 $\beta=T(\alpha, 1^l)$ 均满足

$$\alpha \in \oplus\text{SAT} \Rightarrow \#(\beta) = -1 \pmod{2^{l+1}}$$

$$\alpha \notin \oplus\text{SAT} \Rightarrow \#(\beta) = 0 \pmod{2^{l+1}}$$

证明　回顾一下，对任意两个布尔公式 φ，τ，前面定义的 $\varphi+\tau$ 和 $\varphi\cdot\tau$ 满足 $\#(\varphi+\tau)=\#(\varphi)+\#(\tau)$ 且 $\#(\varphi\cdot\tau)=\#(\varphi)\cdot\#(\tau)$。而且，$\varphi+\tau$ 和 $\varphi\cdot\tau$ 的规模至多比 φ，τ 的规模大常数倍。考虑公式 $4\tau^3+3\tau^4$（其中 τ^3 表示 $\tau\cdot(\tau\cdot\tau)$，τ^4 类似）。容易验证，

$$\#(\tau) = -1 \pmod{2^l} \Rightarrow \#(4\tau^3+3\tau^4) = -1 \pmod{2^{l+1}} \tag{17.10}$$

357

$$\#(\tau)=0 \pmod{2^{r}} \Rightarrow \#(4\tau^{3}+3\tau^{2})=0 \pmod{2^{r+1}} \tag{17.11}$$

令 $\Psi_0=a$，$\Psi_{i+1}=4\Psi_i^3+3\Psi_i^2$。令 $\beta=\Psi_{\lceil \log(l+1)\rceil}$。反复运用等式(17.10)和(17.11)，可以证明，如果 $\#(a)$ 是奇数，则 $\#(\beta)=-1 \pmod{2^{l+1}}$；如果 $\#(a)$ 是偶数，则 $\#(\beta)=0 \pmod{2^{l+1}}$。此外，$\beta$ 的规模比 a 的规模至多大 $\exp(O(\log l))$ 倍，因此归约过程的运行时间是输入长度的多项式。 ■

用引理 17.17 和引理 17.22 证明定理 17.14：设 f 是令参数 $m=2$ 时引理 17.17 得到的归约。由于它是一个随机归约，我们将它视为具有两个输入的确定归约，第一个输入是量化的布尔公式 φ，第二个输入是随机位串 r。令 R 是随机位串的长度。再设 T 是令 $l=R+2$ 时引理 17.22 得到的归约，当然该归约的运行时间是 $\mathrm{poly}(R, |f(\Psi)|)$。

考虑将复合归约 $T\cdot f$（亦即，使用 f 之后再使用 T）以确定的方式应用到输入 φ，r 上，关注下面的累加和模 2^{l+1} 的结果：

$$\sum_{r\in\{0,1\}^{R}} \#(T\cdot f(\varphi,r)) \tag{17.12}$$

如果 φ 为真，则(17.12)式中比例至少为 3/4 的项等于 $-1 \pmod{2^{l+1}}$，其余的项等于 $0 \pmod{2^{l+1}}$。因此，此时(17.12)式模 2^{l+1} 的结果将介于 -2^R 和 $-\lceil 3/4\times 2^R\rceil$ 之间。

另一方面，如果 φ 为假，则(17.12)式中比例至少 3/4 的项等于 $0 \pmod{2^{l+1}}$，其余的项等于 $-1 \pmod{2^{l+1}}$。因此，此时(17.12)式模 2^{l+1} 的结果将介于 $-\left\lceil \frac{1}{4}\times 2^R\right\rceil$ 和 0 之间。

由于 $2^{l+1}>2^{R+2}$，上面得出的两个区间是不相交的。因此，如果我们用某个问题向 #SAT 神喻进行咨询，进而计算得到(17.12)式的值，则可以判定这个值属于哪个区间，进而判定出 φ 是否为真。

但为了得到对 #SAT 神喻的咨询问题，我们可以直接利用库克-勒维构造[⊖]来表达 $T\cdot f$ 所执行的确定型计算。具体地讲，把布尔公式 $T\cdot f(\varphi, r)$ 的变量向量记为 y。我们构造一个布尔公式 $\Gamma(r, y, z)$ 使得：$\Gamma(r, y, z)$ 等于 1 在某个赋值 (r, y, z) 上成立当且仅当 y 是 $T\cdot f(\varphi, r)$ 的满足性赋值。公式 $\Gamma(r, y, z)$ 可以通过将库克-勒维构造用于如下线路 C 来获得：线路 C 完成两项计算任务，它首先计算得到 $T\cdot f(\varphi, r)$（其中 φ 被"硬件实现"在线路中），然后它再将 y 代入 $T\cdot f(\varphi, r)$。公式 $\Gamma(r, y, z)$ 中的变量 z 对应于线路 C 的内部连线，由于 C 是确定型线路，故 z 的值由 y，r 的值唯一确定。

于是，$\#(\Gamma(r, y, z)) \bmod 2^{l+1}$ 恰好是(17.12)式。因此，向 #SAT 神喻提出的咨询问题是要求获得 $\#(\Gamma)$ 的答案。 ■

17.5　待决问题

- ⊕SAT 和 #SAT 究竟有什么样的确切能力？
- $n\times n$ 积和式模较小的素数（如 3 或 5）的平均复杂性是怎样的？注意，对于素数 $p>n$，积和式的随机自归约意味着：如果用随机算法计算积和式的最坏时间复杂性很高，则在比例为 $1-O(n/p)$ 的输入上精确计算积和式的复杂性也很高（亦即，

358

计算积和式的平均复杂性很高，参见定理 8.33。）

⊖ 指的是引理 6.11 中的归约方法。——译者注

本章学习内容

- 复杂性类 #P 中的每个函数要求在给定的实例上计算证明的个数。如果 P≠NP，则 #P 中的问题并不都是多项式时间可解的。
- 对于许多自然的 NP 完全问题，它们的计数形式都是 #P 完全问题。但是，却存在一些 #P 完全问题，它们的判定形式属于 P。例如，矩阵积和式的计算问题 perm（它等价于在图上计算完美匹配的个数）是 #P 完全问题，但是图上的完美匹配判定问题却属于 P。
- 出人意料的是，计数的能力强于量词交错的能力：如果用 #P 完全问题作为神喻，则我们可以高效地求解多项式分层中的每个问题。
- 复杂性类 PP 和 $\oplus$P 中的判定问题分别对应于计算 #P 函数中第一个和最后一个不等于 0 的位。在"如果 PP=P 则 #P=FP"所表明的意义下，复杂性类 PP 的刻画能力至少与 #P 本身的刻画能力一样强。虽然还不知道同样的结论对 $\oplus$P 是否也成立，但已经知道的是：PH 中的每个语言都可以随机归约到 $\oplus$P。

本章注记和历史

#P 的定义（连同 #P 的几个有趣的例子）源自瓦利安特（Valiant）的开创性论文[Val79c]。积和式的 #P 完全性源自他的另一篇论文[Val79b]。伊辛模型的划分函数的 #P 完全性源自杰鲁姆（Jerrum）和辛克莱（Sinclair）[JS90]，这篇论文给出了该问题的完全多项式随机近似模式。贝叶斯网络中极大似然估计的 #P 完全性（例 17.2）最早出现在罗斯（Roth）[Rot93]。戴冈（Dagum）和卢比（Luby）[DL93]证明了，极大似然估计的近似计算仍是 NP 难的。威尔士（Welsh）的书[Wel93]给出了复杂性类 #P 丰富的数学结构和该类刻画的数学问题（这些问题涉及纽结理论、图着色和铺砌）。

要初步了解近似求解各种计数问题的完全多项式随机近似模式，请参阅瓦兹拉尼（Vazirani）[Vaz01]的相关章节（这本书是关于一般性近似算法的优秀著作）和杰鲁姆及辛克莱（Sinclair）的综述文章[JS97]。积和式的完全多项式随机近似模式源自杰鲁姆，辛克莱和维戈达（Vigoda）[JSV01]。

户田定理（Toda Theorem）在[Tod91]中被证明。这一定理对复杂性理论造成了积极、有益的影响，因为它展示了算术论证法在复杂性类之间进行推理的效能（第 8 章和第 11 章曾对这一主题进行过深入讨论）。

除了本章涉及的 #P 和 $\oplus$P 等复杂性类，还有许多复杂性类也涉及一些计数概念，参见福特劳（Fortnow）的综述[For97b]。

习题

17.1　证明：例 17.2 中的问题实际上等价于 #SAT，进而是 #P 完全的。

17.2　证明：整数矩阵的积和式计算属于 $\mathbf{FP}^{\#\mathrm{SAT}}$。

17.3　完成定理 17.11 的证明过程中对 XOR 结构的分析。设 G 包含边 $\overrightarrow{uu'}$，$\overrightarrow{vv'}$ 的任意加权图，将这两条边替换为 XOR 结构之后得到的图记为 G'。证明：对于 G 的权值为 w 的环覆盖，如果 $\overrightarrow{uu'}$，$\overrightarrow{vv'}$ 之一恰好出现于在这个环覆盖中，则这个环覆盖对应于

359

G'的一族总权值为 $4w$ 的环覆盖，并且，G'的其他环覆盖的总权值为 0。

17.4　证明：如果存在多项式时间算法将 #CYCLE 近似到 1/2 因子范围内，则 **P**=**NP**。

17.5　证明：如果 **NP**=**P**，则任意 $f \in \#\mathbf{P}$ 均存在一个随机多项式时间算法将 f 近似到 1/2因子范围内。你能证明同样的结论对因子 $1-\epsilon$（其中 ϵ 是任意小的常数）也成立吗？你能使得这些算法变成确定型算法吗？

注意，人们目前仍不清楚：**P**=**NP** 是否意味着 #**P** 函数的准确计算可以在多项式时间完成。

17.6　证明：对于 $\mathbf{AC}^0$ 中的每个语言，存在一个深度为 3 且规模为 $n^{\mathrm{poly}(\log n)}$ 的线路能够在比例为 $1-1/\mathrm{poly}(n)$ 的输入上判定语言的成员资格，并且这个线路具有如下形式：它的输出门是⊕门而其余的门都是扇入度至多为 $\mathrm{poly}(\log n)$ 的∨门和∧门。

360　17.7　改进定理 10.23，证明：$\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$。

第 18 章

Computational Complexity: A Modern Approach

平均复杂性：勒维定理

问题不易求解和问题难于求解之间有很大差别。

——拉赛尔·因帕利亚佐(Russell Impagliazzo)，1995

截止到目前，我们只研究了在所有输入上求解给定计算任务的算法的复杂性，亦即，算法的最坏复杂性。除少数例外(如第 9 章)，大多数的复杂性类考虑的都是最坏复杂性。**NP** 完全性是研究最坏复杂性的经典范例。

在考虑复杂性时，分析者通常只对"实践中"出现的问题实例感兴趣，而算法的最坏行为可能永远不会出现在实践中。当然，要确切地知道哪些问题实例会出现在实践中，这并不总是一件容易的事。算法设计者们尝试了各种办法来刻画这种问题实例并针对"许多"或"大多数"这样的实例设计高效的算法。这类工作被不同的人称作不同的名字，有人称之为平均复杂性分析，也有人称之为算法分析。人们已经发现，事实上有几个 **NP** 完全问题在"平均"图上很容易求解，当然这也依赖于如何定义"平均"的含义。定义平均图的一种方式是将它视为随机产生的图。产生 n-顶点图的最简单的概率模型是，为所有可能的 $\binom{n}{2}$ 条边中每一条边投掷一枚无偏硬币来决定它是否真正出现在图中。这种方法最终以概率 $2^{-\binom{n}{2}}$ 产生每个 n-顶点图。(如果每条边出现在图中的概率是 p 而不是 1/2，则所得分布称为 $G(n, p)$，这种模型也得到了深入研究。)在这种随机图上，许多 **NP** 完全问题都很容易求解。3-COLOR 以很高概率可以在多项式时间内求解。CLIQUE 和 INDSET 可以在 $n^{2\log n}$ 时间内求解，这一时间略高于多项式时间但远低于 2^n。注意，2^n 是这两个问题的最佳算法在最坏实例上的运行时间。但另一方面，我们在第 9 章中对单向函数的研究表明，并非所有 **NP** 问题的随机实例都是易于求解的。

我们的问题是：能否为平均复杂性建立类似于 **NP** 完全性的理论，并利用恰当的归约概念找出平均复杂性下的"难解问题"或"完全问题"。本章概述勒维(L. Levin)——也就是发现库克-勒维定理的那个人——给出的这样一个理论。为简单计，我们只讨论判定问题。 361

该理论的首要目标是厘清问题的"平均"实例的准确含义。这一目标通过假设问题实例服从某个具体的概率分布来达成。但接下来的问题是，哪种概率分布才是符合"实践"的分布呢？勒维给出的大胆的建议是：我们采用可在多项式时间内实现抽样的任何概率分布(亦即，**P**-可抽样分布)。勒维对这种分布的解释如下。"实际的"实例可以通过我们周围的计算工具来产生；因此，如果我们相信邱奇-图灵命题的强形式(16.1 节)是对的，则所有计算都可以用图灵机来模拟，进而我们可以合理地假设产生实例的"计算"不会非常复杂(也就是说，是高效的)。因此我们假设问题实例的产生时间是实例大小的多项式，细节在 18.2 节中给出。

这样，"平均问题"包含两个部分，一是判定问题，二是产生问题输入的多项式可抽样的分布。接下来的问题是如何定义这种平均问题的"高效算法"，也就是如何定义类似于 **P**

的复杂性类。这个问题有点难，我们在 18.1 节准确定义复杂性类 distP。在 18.3 节我们试图在平均复杂性中定义类似于 NP 完全性的概念。这也有一些难度，特别是当定义我们所需的"归约"时。我们定义复杂性类 distNP(在平均复杂性中，它类似于 NP)，然后相应地定义 distNP 完全性，并证明有些问题确实是 distNP 完全的。但是，与 NP 完全性不同的是，我们并不能证明大量的自然问题是 distNP 完全问题。

确定各种 NP 问题的平均复杂性是复杂性理论最重要的目标之一。18.4 节介绍这一领域的现状，并指出它与复杂性的其他研究之间的联系。

18.1　分布问题与 distP

问题的平均复杂性只有相对于问题输入的一个特定的分布才能明确定义。我们将这种认识精确化。

定义 18.1（分布问题）　一个分布问题是一个序对$(L, \mathcal{D})$，其中$L \subseteq \{0,1\}^*$是一个语言，而$\mathcal{D}=\{\mathcal{D}_n\}$是一系列分布，$\mathcal{D}_n$是$\{0,1\}^n$上的一个分布。

例 18.2　下面给出一些分布问题作为例子。

植入团问题(Planted Clique)。令$G_{n,p}$是n-顶点图上的如下分布：每条边以概率p独立地出现在图中。这个分布显然是 P-可计算的。最常见的情况是$p=1/2$，此时，$G_{n,p}$中的每个图都等概率地出现，我们把取自这个分布的图称为"随机图"。

设$k: \mathbf{N} \to \mathbf{N}$是满足$k(n) \leqslant n$的一个函数。在平均复杂性下，一个类似于 CLIQUE 的
362 简单直接的问题是问随机图中是否存大小为$k(n)$的团。但是，事实证明这个问题并不难求解，因为随机图中团的大小高概率地等于一个易于计算的值(它大约等于$2\log n$)[BE76, Mat76]。[⊖]

因此，在平均复杂性下"正确的"$k(n)$-团问题采用如下分布$\mathcal{D}_n$来定义。该分布以 1/2 的概率输出一个n-顶点随机图，以 1/2 的概率随机选择一个大小为$k(n)$的子集S使得S在输出图中是一个团。$k(n)$-团问题要求判定给定的图是否有一个大小至少为$k(n)$的团。注意，$k(n) \gg 2\log n$，因而随机图具有大小为$k(n)$的团的概率非常小。利用谱方法(spectral methods)，人们已经知道了在$k(n) \sim \sqrt{n}$时如何高效求解$k(n)$-团问题[Kuc95, AKS98]。但是，当$k(n)=n^{0.01}$时，人们对如何求解$k(n)$-团问题还知之甚少。

随机 3SAT。n个变量上含有m个子句的随机 3CNF 范式公式可以如下选取：每个子句定义为将 OR 操作作用到随机选取的 3 个文字上。显然，子句个数m越大，随机布尔公式被满足的可能性就越小。不难证明，存在两个常数$c_1 < c_2$使得：如果$m < c_1 n$，则随机布尔公式将高概率地是可满足的公式；如果$m > c_2 n$，则随机布尔公式将高概率地是不可满足的公式(比如，$c_1=1$，$c_2=8$就满足这种性质)。事实上，弗里德古特(Friedgut)[Fri99]已经证明，存在函数$f(n)$(其中$c_1 n \leqslant f(n) \leqslant c_2 n$对任意$n$成立)使得：对任意$\varepsilon > 0$，如果子句个数$m$小于$(1-\varepsilon)f(n)$，则随机布尔公式将高概率地是可满足的公式；如果子句个数m大于$(1+\varepsilon)f(n)$，则随机布尔公式将高概率地是不可满足的公式。人们相信$f(n)=c^* n$，其中常数$c^* \sim 4.26$。如果子句个数m非常接近于$4.26n$时，判定n个变量上含m

⊖ 在无穷多个n上，n-顶点随机图上团的大小以概率$1-o(1)$等于$g(n)=\lfloor 2(\log n - \log\log n + \log(e)+1)\rfloor$。对于任意$n$，$n$-顶点随机图上团的大小以概率$1-o(1)$属于集合$\{g(n)-1, g(n), g(n)+1\}$，也可以参见习题 18.2。

个子句的随机布尔公式是否是可满足的，这似乎非常困难。事实上，由于本章要求平均算法总输出问题的正确解（见定义 18.4），因此当 m 远远大于 $4.26n$ 时判定随机布尔公式的满足性仍然十分困难。特别地，当 $m=n^{1.1}$ 时，尽管此时随机布尔公式以压倒性概率是不可满足的，但人们还没有找到任何多项式时间算法来判定随机布尔公式的满足性（局部性进展工作包括[GK01，FO04，FKO06]）。

随机线性码的解码问题。设 A 是 GF(2) 上的一个 $m\times n$ 矩阵，其中 $m>n$（比如，$m=10n$）。A 的解码问题要求为给定的向量 $z\in \mathrm{GF}(2)^m$ 找出距离 z 最近的向量 y 使得 y 是 A 的像（亦即，$y=Ax$ 对某个 $x\in \mathrm{GF}(2)^n$ 成立）。（研究这一问题的动机是将 A 视为纠错码的生成矩阵，19.2 节将研究纠错码。）当 A 具有各种特殊形状时，线性码的解码问题存在高效算法。但是，当 A 是一个随机矩阵时，该问题还没有找到高效算法。随机线性码的解码问题还有一个广为人知的名字——带噪声的奇偶性学习问题（Learning Parity with Noise）。

随机线性码的解码问题显然是一个搜索问题。固定某个常数 $\varepsilon>0$。如下定义的判定问题 $(L,\mathcal{D}_*)$ 还不知道是否属于 distP。我们令 (a) L 包含所有如下的序对 (A,y)：y 与 A 的某个像之间的汉明距离（Hamming Distance）至多为 εm；(b) 分布 $\mathcal{D}_*$ 以 1/2 的概率输出一个 $m\times n$ 的随机矩阵 A 和随机选取的一个向量 $y\in_R \mathrm{GF}(2)^m$，以 1/2 的概率输出一个 $m\times n$ 的 363
随机矩阵 A 和向量 $y=Ax+e$，其中 x 随机地取自 $\mathrm{GF}(2)^n$ 而 e 是取自 $\mathrm{GF}(2)^m$ 的一个恰有 $\lfloor \varepsilon m \rfloor$ 个分量等于 1 的随机向量。

上述两个问题都密切相关于子集和问题和 $\mathbf{R}^n$ 中离散格上的一些问题。事实证明，离散格上的这些问题在密码学中大有用武之地，参见第 9 章的章节注记。◀

接下来，我们定义复杂性类 distP。在平均复杂性中，它类似于 P。distP 旨在刻画可以高效求解的分布问题 (L,D) 构成的集合[⊖]。对任意算法 A 和输入 x，我们用 $\mathrm{time}_A(x)$ 表示算法 A 在输入 x 上的计算步骤的个数。distP 的一种可能的定义是：我们称 $\langle L,\mathcal{D}\rangle$ 是平均多项式可解的，如果存在算法 A 和一个多项式 p 使得 $A(x)=L(x)$ 对任意 x 成立且 $E_{x\in_R\mathcal{D}_n}[\mathrm{time}_A(x)]\leqslant p(n)$ 对任意 n 成立。

遗憾的是，上述定义不够健壮。事实上，当我们将计算模型替换成计算速度平方下降的其他模型（例如，将多带图灵机替换成单带图灵机）时，上面定义的多项式时间算法可能突变成指数时间算法。这正是如下的简单论断表明的事实。

论断 18.3　*存在一个算法 A 使得在任意 n 上均有：$E_{x\in_R\{0,1\}^n}[\mathrm{time}_A(x)]\leqslant n+1$ 但是 $E_{x\in_R\{0,1\}^n}[\mathrm{time}_A^2(x)]\geqslant 2^n$。*

证明：考虑这样一个算法 A，它在全为 0 的输入上运行 2^n 步，在其他输入上则在 n 步内停机。因此，A 的期望运行时间是 $(1-2^{-n})n+2^{-n}2^n\leqslant n+1$。另一方面，如果运行时间平方增大，则期望运行时间变为 $(1-2^{-n})n^2+2^{-n}2^{2n}\geqslant 2^n$。■

这促使我们采用下面的定义。

定义 18.4（平均多项式和 distP）　*分布问题 $\langle L,\mathcal{D}\rangle$ 属于 distP，如果存在 L 的算法 A 和常数 C，以及 $\varepsilon>0$ 使得在任意 n 上均有*

$$E_{x\in_R\mathcal{D}_n}\left[\frac{\mathrm{time}_A(x)^\varepsilon}{n}\right]\leqslant C \tag{18.1}$$

⊖ 本章只讨论确定型算法。但是所给的理论可以很自然地推广到概率型算法上得到 BPP，RP，coRP，ZPP 等类在平均复杂性下的类似复杂性类。

有几点值得注意。首先，$\mathbf{P}\subseteq\mathrm{dist}\mathbf{P}$。原因在于，如果一个语言可以用确定型算法 A 在 $O(|x|^c)$时间内判定，则(18.1)式给出的数学期望无论在何种分布下都以一个常数为上界。其次，定义 18.4 在更换模型的意义下是健壮的。事实上，如果更换模型使得算法的计算时间平方增加，则只需将 ϵ 乘以 1/2 就可以保证(18.1)式仍然有界。

定义 18.4 的另一个特点是，算法的运行时间是多项式时间的概率很高。事实上，由马尔科夫不等式可知，(18.1)式意味着“在任意 $K>1$ 上，$\Pr\left[\frac{\mathrm{time}_A(x)^\epsilon}{n}\geqslant KC\right]=$

364 $\Pr[\mathrm{time}_A(x)\geqslant(KCn)^{1/\epsilon}]$均至多为 $1/K$”。

最后注意，定义 18.4 在小改动下也是健壮的。例如，对任意 $d>0$，(18.1)式给出的条件等价于存在 ϵ，C 使得

$$\underset{x\in_R\mathcal{D}_n}{E}\left[\frac{\mathrm{time}_A(x)^\epsilon}{n^d}\right]\leqslant C \tag{18.2}$$

参见习题 18.6。

18.2 “实际分布”的形式化定义

实际的问题实例源自我们周围的世界(例如，需要进一步理解的图片、机器人要巡游的建筑，等等)，而世界却不会花时间将这些实例修剪成算法难以处理的实例——辩证地看，世界对算法而言是中立的。如果假设实际的问题实例是由一个高效算法产生的(请参见 1.6.3 节的讨论)，则我们可以用计算术语来刻画世界的中立性。我们给出以下两种刻画方式。

多项式时间可计算的(或称 P-可计算的)分布。这种分布与一个确定型多项式时间图灵机相关联。在任意输入 $x\in\{0,1\}^*$上，该图灵机能够计算累计概率 $\mu_{\mathcal{D}_n}(x)$，其中

$$\mu_{\mathcal{D}_n}(x)=\sum_{y\in\{0,1\}^n\,|\,y\leqslant x}\Pr_{\mathcal{D}_n}[y]$$

这里，$\Pr_{\mathcal{D}_n}[y]$表示字符串 y 在分布$\mathcal{D}_n$中的概率而 $y\leqslant x$ 则表示在字典序中 y 比 x 靠前或等于 x。

将 x 在字典序中的前驱记为 $x-1$，我们有

$$\Pr_{\mathcal{D}_n}[x]=\mu_{\mathcal{D}_n}(x)-\mu_{\mathcal{D}_n}(x-1)$$

上式意味着，如果 $\mu_{\mathcal{D}_n}$是多项式时间可计算的，则$\Pr_{\mathcal{D}_n}[x]$也是多项式时间可计算的。但是，如果 $\mathbf{P}\neq\mathbf{NP}$，则该结论的逆命题不成立(参见习题 18.3)。均匀分布是 P-可计算的，同样，许多其他用显式公式定义的分布也是 P-可计算的。

多项式时间可抽样的(或称 P-可抽样的)分布。这种分布与一个概率型多项式时间图灵机关联，该图灵机可以产生服从这种分布的样本。具体地讲，我们称$\mathcal{D}=\{\mathcal{D}_n\}$是 P-可抽样的，如果存在一个多项式 p 和概率型 $p(n)$时间算法 S 使得随机变量 $S(1^n)$和$\mathcal{D}_n$服从同一个分布。

如果一个分布是 P-可计算的，则它也是 P-可抽样的，但是如果 $\mathbf{P}\neq\mathbf{P}^{\#\mathbf{P}}$则上述结论的逆命题不成立(参见习题 18.4 和 18.5)。本章主要限定在 P-可计算的分布上进行讨论，但我们给出的理论很容易推广到 P-可抽样的分布上(参见 18.3.2 节)。

18.3 distNP 及其完全问题

365 下面的复杂性类是研究平均复杂性的核心，它是 NP 在平均复杂性下的类似描述。

定义 18.5（复杂性类 distNP）　如果 $L\in\mathbf{NP}$ 且$\mathcal{D}$是 P-可计算的分布，则称分布问题$(L, \mathcal{D})$属于复杂性类 distNP。

下面，定义分布问题之间的归约。

定义 18.6（平均归约）　称分布问题$(L, \mathcal{D})$平均归约(Average-case Reduction)到分布问题$(L', \mathcal{D}')$，记为$(L, \mathcal{D})\leqslant_p(L', \mathcal{D}')$，如果存在多项式时间可计算的映射 f 和两个多项式 p，q：$\mathbf{N}\to\mathbf{N}$ 满足

1.(正确性)在任意 $x\in\{0, 1\}^*$ 上，$x\in L\Leftrightarrow f(x)\in L'$；

2.(长度匀整性)在任意 $x\in\{0, 1\}^*$ 上，$|f(x)|=p(|x|)$；

3.(支配性)在任意 $n\in\mathbf{N}$ 和 $y\in\{0, 1\}^{p(n)}$ 上，$\Pr[y=f(\mathcal{D}_n)]\leqslant q(n)\Pr[y=\mathcal{D}'_{p(n)}]$。

定义 18.6 的第一个条件是标准归约要满足的条件，它确保了 L'的判定算法可以容易地转换为 L 的判定算法。第二个条件是技术性要求，它主要用来简化定义并确保归约关系满足传递性(参见习题 18.7)。第三个条件要求$\mathcal{D}'$(在忽略多项式因子的意义下)支配"f 作用在$\mathcal{D}$上所得的分布 $f(\mathcal{D})$"，下面解释要求该条件成立的动机。注意，定义的目标是为了确保"如果$(L, \mathcal{D})$是难解的，则$(L', \mathcal{D}')$也是难解的"，因此，如果算法 A'可以高效求解$(L', \mathcal{D}')$，则如果如下"显而易见"的算法 A 能够高效求解$(L, \mathcal{D})$就好了。算法 A 如下工作：在服从分布$\mathcal{D}$的输入 x 上，它先计算 $y=f(x)$，然后在 y 上调用算法 A'。然而，简单推理不难发现，该算法 A 不符合要求，因为我们无法排除如下的可能性：尽管 A'在服从分布$\mathcal{D}'$的输入上复杂度不高，但是在某些服从$\mathcal{D}$的 x 上，A'在 $f(x)$上的复杂度却很可能非常高。支配性条件恰好排除了这种可能性。

定理 18.7　如果$(L, \mathcal{D})\leqslant_p(L', \mathcal{D}')$且$(L', \mathcal{D}')\in\mathrm{distP}$，则$(L, \mathcal{D})\in\mathrm{distP}$。

证明　假设 A'是求解$(L', \mathcal{D}')$的多项式时间算法：也就是说，存在常数 C，$\epsilon>0$ 使得在任意 m 上均有

$$E\left[\frac{\mathrm{time}_{A'}(\mathcal{D}'_m)^\epsilon}{m}\right]\leqslant C \tag{18.3}$$

设 f 是$(L, \mathcal{D})$到$(L', \mathcal{D}')$的归约映射而 A 是用于判定 L 的如下"显而易见"的算法：在输入 x 上，先计算 $f(x)$，然后输出 $A'(f(x))$。由于算法 A 能够判定 L，故只需证明算法 A 在服从分布$\mathcal{D}$的所有输入上的平均复杂度是多项式时间。

为简单计，我们假设在任意 x 上，$|f(x)|=|x|^d$，并且在长度为 n 的输入上计算 f 远快于在长度为n^d的输入上运行 A'，进而 $\mathrm{time}_A(x)\leqslant 2\mathrm{time}_{A'}(f(x))$。(下面的证明很容易推广到忽略这些假设条件的情况上。)为了完成定理的证明，只需证明

$$E\left[\frac{\frac{1}{2}\mathrm{time}_A(\mathcal{D}_n)^\epsilon}{q(n)n^d}\right]\leqslant C$$

366

其中 q 是支配性条件中出现的多项式。根据习题 18.6，上式足以表明$(L, \mathcal{D})\in\mathrm{distP}$。

事实上，根据 A 的定义和我们的假设条件可知

$$\begin{aligned}E\left[\frac{\frac{1}{2}\mathrm{time}_A(\mathcal{D}_n)^\epsilon}{q(n)n^d}\right]&\leqslant\sum_{y\in\{0,1\}^{n^d}}\Pr[y=f(\mathcal{D}_n)]\frac{\mathrm{time}_{A'}(y)^\epsilon}{q(n)n^d}\\&\leqslant\sum_{y\in\{0,1\}^{n^d}}\Pr[y=f(\mathcal{D}'_{n^d})]\frac{\mathrm{time}_{A'}(y)^\epsilon}{n^d}\qquad\text{(由支配性)}\end{aligned}$$

$$= E\left[\frac{\text{time}_A(\mathcal{D}'_{\varepsilon'})^{\varepsilon'}}{n^d}\right] \leqslant C \qquad \text{(由(18.3)式)} \qquad \blacksquare$$

18.3.1　distNP 的一个完全问题

当然，只有找到相关问题之间的归约之后，才能运用定理 18.7。为展示这一点，我们给出一个 distNP 完全问题(尽管这个问题是我们人为构造的)。如果$(L', \mathcal{D}') \in \text{distNP}$且$(L, \mathcal{D}) \leqslant_p (L', \mathcal{D}')$对任意$(L, \mathcal{D}) \in \text{distNP}$成立，则称$(L', \mathcal{D}')$是 distNP 完全的。我们有如下定理。

定理 18.8　(distNP 完全问题的存在性[Lev86])　令 U 是如下的所有三元组$\langle M, x, 1^t\rangle$构成的语言：存在位串 $y \in \{0,1\}^*$使得非确定型图灵机 M 以 x 为输入时能够在 t 步之内输出 1。

对于任意 n，用$\mathcal{U}_n$表示长度为 n 的所有三元组$\langle M, x, 1^t\rangle$上的如下概率分布：随机选择长度至多为 $\log n$ 的位串来表示 M，再从集合$\{0, \cdots, n-|M|\}$中随机选取 t，最后从$\{0,1\}^{n-t-|M|}$中随机选取 x。该分布是多项式时间可计算的(习题 18.8)。[⊖]

那么，$\langle U, \mathcal{U}\rangle$是 distNP 完全的。

问题 U 当然是一个 **NP** 完全问题，这只需注意到如下的平凡归约：给定 $p(n)$时间非确定型图灵机 M 判定的语言 L，把每个 x 映射为三元组$\langle M, x, 1^{p(|x|)}\rangle$，则 L 就被归约到 U。但是，这个归约不是平均归约，因为它可能不满足支配性条件。之所以出现这种问题，是由于我们需要将每个分布问题$(L, \mathcal{D})$归约到$(U, \mathcal{U})$，但是如果$\mathcal{D}$存在"峰值"(也就是说，某个长度为 n 的输入 x 在$\mathcal{D}$中出现的概率显著地大于 2^{-n})，则归约过程得到的确定的三元组$\langle M, x, 1^t\rangle$在$\mathcal{U}_n$中的概率却有可能不超过 2^{-n}。

这种障碍可以用下面的引理来克服。该引理表明，在任意多项式可计算的分布上，我们可以在输入上运用一种简单的变换使得所得分布不再具有"峰值"。

367

引理 18.9　(峰值消除)　如果$\mathcal{D}=\{\mathcal{D}_n\}$是 **P**-可计算分布，则存在一个多项式时间可计算的函数 $g: \{0,1\}^* \to \{0,1\}^*$满足

1. g 是一对一的函数：$g(x)=g(z)$当且仅当 $x=z$；
2. 对任意 $x \in \{0,1\}^*$，$|g(x)| \leqslant |x|+1$；
3. 对任意位串 $y \in \{0,1\}^n$，$\Pr[y=g(\mathcal{D}_n)] \leqslant 2^{-n+1}$。

证明　对于任意位串 $x \in \{0,1\}^*$，定义 $h(x)$是 $\mu_{\mathcal{D}_n}(x)$和 $\mu_{\mathcal{D}_n}(x-1)$的二进制表示的最长公共前缀。注意，如果 $\Pr_{\mathcal{D}_n}[x] \geqslant 2^{-k}$，则由 $\mu_{\mathcal{D}_n}(x)-\mu_{\mathcal{D}_n}(x-1)=\Pr_{\mathcal{D}_n}[x]$可知，值 $\mu_{\mathcal{D}_n}(x)$和 $\mu_{\mathcal{D}_n}(x-1)$必然在前 k 个位中的某些位置上取不同的二进制位，进而 $|h(x)| \leqslant k$。再注意到，由于$\mathcal{D}$是 **P**-可抽样的，故函数 h 是多项式时间可计算的。而且，h 是一对一的函数，因为一旦两个串 s_1 和 s_2的最长公共前缀为 z，则另一个具有前缀 z 的串 s_3必然与 s_1，s_2之一有更长的公共前缀。

现在，我们为任意 $x \in \{0,1\}^*$定义

$$g(x) = \begin{cases} 0x & \text{如果 } \Pr_{\mathcal{D}_n}[x] \leqslant 2^{-n} \\ 1h(x) & \text{否则} \end{cases}$$

⊖　严格地讲，如果将分隔符等因素考虑进来，则输入的长度肯定不止 n 个位。但是，这些细节很容易处理，因而我们后续的讨论将忽略它们。

显然，g 是满足 $|g(x)|\leqslant|x|+1$ 的一对一的函数。下面证明，$\Pr[y=g(\mathcal{D}_n)]\leqslant 2^{-n+1}$ 对任意 $y\in\{0,1\}^{n+1}$ 成立。如果 y 不是任意 x 的函数值 $g(x)$，则结论是平凡的，因为 $\Pr_{g,\mathcal{D}}[y]=0$。

如果 $y=0x$，其中 $\Pr_{\mathcal{D}}[x]\leqslant 2^{-|x|}$，则 $\Pr_{g,\mathcal{D}}[y]\leqslant 2^{-|x|+1}$，因此也无需证明什么。最后，如果 $y=g(x)=1h(x)$，其中 $\Pr_{\mathcal{D}}[x]>2^{-|x|}$，则如前所述，我们有 $|h(x)|\leqslant\log(1/\Pr_{\mathcal{D}}[x])$，进而 $\Pr_{g,\mathcal{D}}[y]=\Pr_{\mathcal{D}}[x]\leqslant 2^{-|x|+1}$。■

现在，我们可以证明定理 18.8 了。

定理 18.8 的证明　设$(L,\mathcal{D})$属于 distNP，且 M 是接受语言 L 的多项式时间非确定型图灵机。我们新定义一个非确定型图灵机 M'：在输入 y 上，猜测满足 $y=g(x)$ 的 x（其中 g 是引理 18.9 得出的函数），然后执行 $M(x)$。令多项式 p 是 M' 的运行时间。

为了将$(L,\mathcal{D})$归约到$(U,\mathcal{U})$，我们简单地将 x 映射为三元组$(M',g(x),1^k)$，其中 $k=p(n)+\log n+n-|M'|-|g(x)|$（我们可以假设当 n 充分大之后，M' 的位串表示的长度 $|M'|$ 至多为 $\log n$）。这种归约显然满足长度匀整性条件。同时，由于 g 是一对一的函数，这种归约也满足正确性条件。因此，剩下的只需证明这种归约满足支配性条件。

事实上，由引理 18.9，这种归约得到的长度为 m 的三元组$(M',y,1^k)$的概率至多为 $2^{-|x|+1}$，而$\mathcal{U}_m$产生该元组的概率至少为 $2^{-\log m}2^{-|x|}\dfrac{1}{m}$，进而支配性条件满足。■

上述证明依赖于如下事实：每个图灵机都可以用长度为常数的位串来表示（也就是说，图灵机的位串表示的长度独立于输入的长度）。事实上，证明过程导致图灵机的计算速度下降该常数的指数因子。由于这个常数对典型 **NP** 语言而言通常很大，因此，实践过程中必须仔细考虑它。

368

18.3.2　P-可抽样的分布

辩证地看，某些分布实质上即使不是多项式可计算的，也是多项式可抽样的。如果 $L\in\mathbf{NP}$ 且$\mathcal{D}$是 **P**-可抽样的，则称分布问题$(L,\mathcal{D})$属于复杂性类 sampNP。如果$(L',\mathcal{D}')\in$ sampNP 且$(L,\mathcal{D})\leqslant_p(L',\mathcal{D}')$，则称$(L',\mathcal{D}')$是 sampNP 完全的。幸运的是，借助下面的结论，定理 18.8 等相关结论都可以转换到 sampNP 上。

定理 18.10　([IL90])　如果$(L,\mathcal{D})$是 distNP 完全的，则它也是 sampNP 完全的。

我们略去定理 18.10 的证明，它将用到一种去随机化技术——残余哈希引理（见第 21 章的引理 21.26）。

18.4　哲学意义和实践意义

截止到目前，读者已经学习和了解了许多复杂性类以及相关的一些猜想，因此从全局范围内考量复杂性世界（world of complexity）或许大有裨益。因帕利亚佐（Impagliazzo）[Imp95b]很好地将复杂性世界划分为五个界（five world）[⊖]，并用非常容易记忆的名字来表示它们。目前，我们仍不清楚它们中的哪一界才是真实的（也就是说，现实世界到底属于哪个界仍不清楚）。

⊖　本节中的"界"不同于"上、下界"中的"界"，它是英文单词"world"的意译，是因帕利亚佐对复杂性的一种整体认识。

算法界(Algorithmica)：算法界描述是 **P**=**NP**(或本质上相同，如 **NP**⊆**BPP**)的世界，具体地讲，算法界也可以定义为"SAT 问题在其中存在简单而神奇的线性时间算法"的世界。正如 2.7.3 节所述，这种世界是一个计算乌托邦。在这种世界中，工程设计、程序编写、数学，甚至写作、作曲和绘画等目前需要极大创造力才能实现的任务，都可以被自动化。另一方面，SAT 问题的线性时间算法还可以用来破译各种加密方案，因而目前被人们广泛使用的各种密码方案都将从这个世界上消失。

启发式界(Heuristica)：启发式界描述的是 **P**≠**NP** 但 dist**NP**，samp**NP**⊆dist**P** 的世界。也就是说，这个世界存在一个高效而神奇的算法，它"几乎"能求解每个 **NP** 问题。虽然算法可能在某些输入上失效或者运行时间很长，但是这种输入很难找到，因为它们基本不会出现在实际应用中。在某些方面，启发式界和算法界非常相似。原因在于，如果我们无法找出算法失效的实例，则这两个世界将难以区分！事实上，求解 **NP** 优化问题、找到较短的数学证明、破译密码方案等等大多数在算法界中成立的应用，在启发式界中仍然成立。但是，也有某些应用在算法界成立但在启发式界中却不成立。特别地，尽管我们已经知道"如果 **P**=**NP**，则多项式分层 **PH** 将坍塌到 **P**"(见定理 5-4)，但是在平均复杂性上我们却没有得到类似的结论。

悲观界(Pessiland)：在悲观界描述的世界中，不仅 dist**NP** 和 samp**NP** 都不含于 dist**P** 中，而且单向函数(参见第 9 章)也不存在。因帕利亚佐之所以将这个世界称为悲观界，是因为在某种程度上讲我们的真实世界最不可能是这种世界。在这个世界中，一方面，我们将不可能
369 再拥有算法界和启发式界中令人心潮澎湃的神奇算法，另一方面，我们也不能再使用绝大多数的加密方案了。(记住，第 9 章中讨论的单向函数本质上是绝大多数加密算法的基础。)

迷你密码界(Minicrypt)：在迷你密码界中，虽然单向函数存在(进而 dist**NP**⊄dist**P**，参见习题 18.10)，但是高度结构化的 **NP** 问题(如，因数分解问题)却可以在多项式时间内求解。更正式的说法是，迷你密码界是存在单向函数但不存在公钥密码方案和密钥交换协议的世界。尽管私密密码方案、伪随机数产生器、伪随机函数、数字签名等许多密码应用都只依赖于单向函数，但是人们目前还不清楚只用这些单向函数能否构造出公钥密码、安全多方计算等重要而振奋人心的加密方案。

密码界(Cryptomania)：在密码界描述的世界中，大整数的因数分解问题具有指数级的平均复杂度(也可以是，离散对数问题或最短格向量问题等其他高度结构化的 **NP** 问题具有指数级的平均复杂度)。很多研究者相信，我们所处的真实世界就是密码界。在密码界中，我们没有通用的算法，因而我们必须借助启发式规则、近似、创造力和辛苦工作来为很多重要的计算问题设计算法。这些重要的计算问题包括了，在各参与方不事先共享密钥的条件下如何才能获得安全通信的能力(亦即，大多数电子商务广泛采用的公钥密码方案)，以及如何获得安全的在线审计和投票方案等更复杂的密码方案。

严格地讲，因帕利亚佐还忽略了一些中间情况，我们将这些情况统统归入"怪异界"。比方说，怪异界包含了如下情况：SAT 的复杂度既不是线性的，也不是二次的，而是类似于 n^{100} 的多项式或者类似于 $n^{\log n}$ 这样的缓慢增长的超多项式函数。再比方说，怪异界也包含了下述情况：SAT 等计算问题的复杂度随着输入规模的变化而诡异地变化。也就是说，任何复杂度表达式都只对某些输入规模有效而对另一些输入规模却无效。但是，不可否认，因帕利亚佐给出的五界已经较好地刻画了 **NP** 的平均难度的几种可能的情形。事实上，在一定程度上讲，"从五界中排除与我们现实世界矛盾的一些界"是计算复杂性理论的中心任务。

本章学习内容

- 平均复杂性是相对于输入的特定分布来定义的。因此，同一个计算问题很可能在一个分布上具有较低的平均复杂度，而在另一个分布上却有较高的平均复杂度。
- distP 是平均复杂性中类似于 P 的复杂性类，它刻画了在平均复杂性下存在高效算法的分布问题。
- 在平均复杂性中，与 NP 类似的复杂性类是 distNP 或 sampNP，具体依赖于我们用 P-可计算分布还是用 P-可抽样分布来作为"实际分布"的模型。定理 18.8 中的分布问题$\langle U, U\rangle$在 distNP 和 sampNP 中都是完全的。
- 同 P≠NP 一样，NP 问题的平均难度目前也是待决问题。目前，人们仍未找到这两个复杂性类之间的任何非平凡的关系。例如，人们还不清楚 NP$\subseteq$P 是否蕴含着 distNP$\subseteq$distP。

370

本章注记和历史

随机图分布是输入上最自然的分布之一，对它的研究始于厄尔多斯(Erdos)和任殿(Renyi)在 1959 年发表的一篇论文[ER59]，波罗巴斯(Bollobas)的书对这一宏大的领域精彩地进行了综述[Bol01]。算法的平均复杂性分析也称为算法概率分析(Probabilistic Analysis of Algorithm)，参见里德(Reed)的综述[RF98]。思皮尔曼(Spielman)和腾尚华(Shanghua Teng)引入了算法平滑分析(Smoothed Analysis of Algorithm)——一种基于介于最坏复杂性分析和概率分析之间的分析概念。在算法平滑分析的概念下也存在类似于 NP 完全性的理论。

勒维(Levin)在[Lev86]中给出了他的理论和定理 18.8。勒维对该理论的形式化论述比本章给出的理论更具一般性。例如，勒维给出的理论允许所有算法(亦即，计算 P-可计算分布的算法和归约算法)都是随机算法。

勒维理论在 P-可抽样分布上的扩展源自因帕利亚佐(Impagliazzo)和勒维[IL90]。本·戴维(Ben-David)等人[BDCGL89]讨论了勒维理论的许多基本事实，例如修改假设条件对理论产生的影响、P-可抽样性和 P-可计算性之间的关系等等，参见戈德赖希(Goldreich)的综述[Gol97]。约翰逊(Johnson)对平均复杂性的综述[Joh84]有些过时了(它几乎与勒维的原始论文同时发表)，但仍具有很高的可读性。平均复杂性领域的一个延续至今的目标是证明各种自然的 NP 问题的平均复杂性完全性。利夫勒(Livne)最近发表的一篇论文[Liv06]给出这方面研究取得的最强结果(其中的问题是"自然的"，但是概率分布不是自然的)。

习题

18.1　给出一个平均复杂度为线性时间的算法来判定均匀分布图(每条边在图中出现的概率为 1/2)的 3-可着色性。

18.2　给出一个平均复杂度为 $n^{\log n}$ 的算法来求解分布$\langle G, k\rangle$上的 CLIQUE 问题，其中 G 是服从均匀分布的 n-顶点图，k 是随机取自$[n]$的整数。

18.3　证明：如果 P≠NP，则存在长度为 n 的位串上的一族分布$\mathcal{D}=\{\mathcal{D}_n\}$使得，对于任意

$x\in\{0,1\}^n$存在算法计算 $\Pr[\mathcal{D}_n=x]$，但是$\mathcal{D}$不是 **P**-可计算的。

18.4　证明：如果一个分布是 **P**-可计算的，则它也是 **P**-可抽样的。

18.5　证明：$\mathbf{P}^{\#\mathbf{P}}\neq\mathbf{P}$，则存在一个多项式时间可抽样的但不是多项式时间可计算的分布。

18.6　证明：如果一个算法满足(18.2)式，则它也必然满足(18.1)式，只是常数 ε，C 可
371 能需要重新选取。

18.7　证明：本章定义的平均归约满足传递性：也就是说，如果$\langle L_1,\mathcal{D}_1\rangle\leqslant_p\langle L_2,\mathcal{D}_2\rangle$并且$\langle L_2,\mathcal{D}_2\rangle\leqslant_p\langle L_3,\mathcal{D}_3\rangle$，则$\langle L_1,\mathcal{D}_1\rangle\leqslant_p\langle L_3,\mathcal{D}_3\rangle$。

18.8　证明：定理 18.8 中的分布$\mathcal{U}$是 **P**-可计算的。

18.9　证明：引理 18.9(峰值消除引理)中定义的函数在下述意义下可以高效地求得逆函数：如果 $y=g(x)$，则给定 y 之后我们可以在$|x|^{O(1)}$时间内重构出 x。

372 18.10　证明：如果单向函数是存在的，则 dist**NP**$\subseteq$dist**P**。

第 19 章

Computational Complexity: A Modern Approach

难度放大和纠错码

core(核)：事物的中心，字面意义或比喻意义的中心。

——《哥伦比亚美国标准英语指导》，1993

复杂性理论研究各种函数的计算难度。本章继续研究第 9 章和第 18 章中曾研究过的重要主题——平均复杂性下的各种难解函数。后面，第 20 章还将继续研究它。本章将专门研究难度放大技术，这种技术在许多场合都大有用武之地。在密码学(第 9 章)中，难解函数对"设计具有非平凡密钥长度的安全加密方案"而言是必需的。人们猜想的难解函数(如，因数分解)都仅在少量实例上难解，而并非在所有实例上都难解。因此，这种函数对某些密码方案而言还不能胜任，但是利用难度放大技术可以将这种函数改造成能够胜任这些密码方案的函数。第 20 章还将介绍难度放大技术的另一种强大的应用——在最坏复杂性的理论假设下去除复杂性类 **BPP** 的随机性。图 19-1 概要地介绍了本章的各个小节，以及各个小节的结论与第 20 章的结论之间的联系。本章介绍的各种思想除了在复杂性理论中大有用武之地外，它还可以用于其他领域，比如构造新的纠错码和设计新的机器学习算法，等等。

为简单计，我们只在布尔函数上研究难度放大技术，但这些技术也可以应用到非布尔函数上。19.1 节介绍第一种难度放大技术——姚期智 XOR 引理(Yao's XOR Lemma)。这种技术可以让我们将弱难解函数转变为强难解函数。粗略地讲，如果任意中等规模的线路在"某个不可忽略的比例(如 0.01)的"输入上无法计算 f，则称 f 是弱难解的；如果任意中等规模的线路在将近一半(如，$1/2-\varepsilon$)的输入上无法计算 f，则称 f 是强难解的。(注意，每个布尔函数都存在一个平凡的线路，它至少能够在一半输入上正确地计算函数，这个平凡的布尔线路就是在所有输入上全部输出 1 或全部输出 0 的线路。)19.1 节给出了一种变换方法，它能利用一种简单的 XOR 构造来变换任意的布尔函数，变换后的函数的计算复杂性不会大幅增加；但是这种方法具有如下性质：如果变换前的函数是弱难解的，则变换后的函数将变成强难解的。这种 XOR 构造在密码学应用中也非常有用，正如第 9 章已经指出的那样。这种 XOR 构造所完成的工作就是识别函数的"难度核"(hard core)，这是一个引人注目的概念。

然后，我们再转而研究各种不同的难度放大技术。即使给定的函数仅仅在最坏情况下是难解的，应用这些难度放大技术所得到的函数也将是强难解的。显然，这些变换技术具有高度的非平凡性，因为计算问题的最坏复杂性和它们的平均复杂性之间差别非常大。(例如，尽管为给定的整数找出最小因数在一般情况下是难解的，但是至少在一半的输入上(即，偶数)解决该问题却是平凡的。)研究这些难度放大技术时，我们的主要工具是纠错码(Error-correcting Code)。19.2 节和 19.3 节先复习纠错码的基本定义和基本构造方法；而 19.4 节则讨论局部解码，它是确保纠错码能运用于难度放大技术所必需的主要概念。最终，我们得到一种函数变换方法，它能将最坏复杂性下的难解函数 f 变换为平均复杂性下的难解函数 $\hat{f}$。

将 19.4 节介绍的变换方法和 19.1 节介绍的姚期智 XOR 引理组合在一起，我们可以从最坏复杂性下的难解函数得到平均复杂性下极其难解的函数。但是，从量化的角度看上述变换不是最优的：也就是说，即使变换前的函数对指数规模($2^{\Omega(n)}$)的线路而言在最坏复杂性下是难解的，我们也只能保证变换后的函数仅对亚指数规模($2^{n^{\Omega(1)}}$)的线路而言在平均复杂性下是难解的。19.5 节和 19.6 节将证明一个更强的结论，该结论中的变换方法能够一下子直接将最坏复杂性下的难解函数 f 变换为平均复杂性下极其难解的函数 $\hat{f}$。这种变换以更加复杂的方式来应用纠错码，其中涉及的技术称为列表解码(List Decoding)，它本身也具有独立的研究价值。19.5 节讨论列表解码；19.6 节介绍难度放大变换中要用到
374 的纠错码技术——局部列表解码(Local List Decoding)，它是列表解码的一种扩展。

熟悉纠错码理论的读者在第一遍阅读时可以跳过 19.2 节和 19.3 节(只需停下来通过定义 19.10 和定义 19.12 回顾一下里德-所罗门码(Reed-Solomon code)和里德-穆勒码(Reed-Muller code)以及相应的解码算法即可)直接进入 19.4 节。

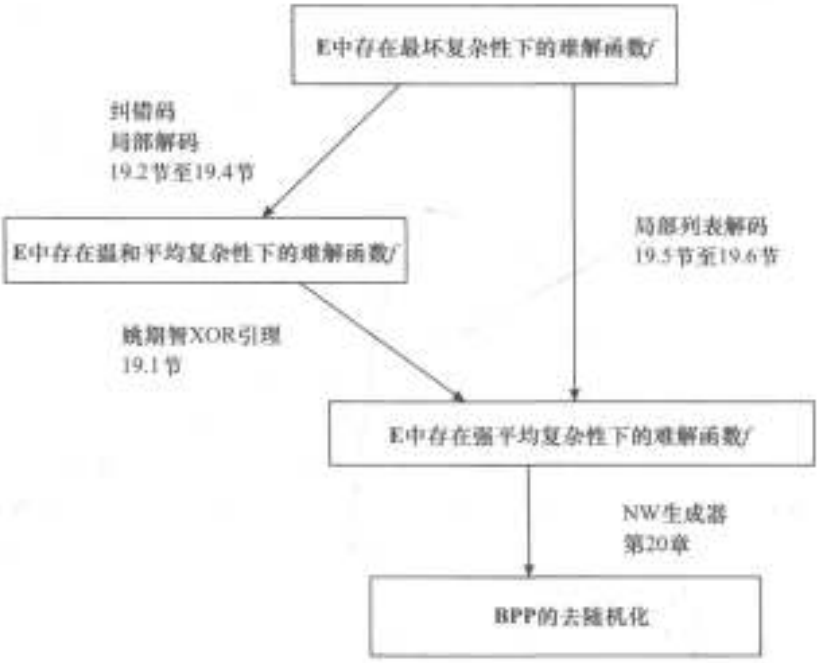


图 19-1　第 19 章的组织结构[⊖]

19.1　从温和难度到强难度：姚期智 XOR 引理

姚期智 XOR 引理将具有"温和"平均难度的函数变换为具有强平均难度的函数。变换过程实际上简单而自然，但其分析却颇为复杂(在我们看来它同时也很优美)。要表述引理，我们需要先精确地定义函数的最坏难度和函数的平均难度。

定义 19.1　(平均难度和最坏难度)　对 $f:\{0,1\}^n\to\{0,1\}$ 和 $\rho\in(0,1]$，函数 f 的 ρ-平均难度 $\mathrm{H}^{\mathrm{avg}}_{\rho}(f)$ 定义为使得如下条件成立的最大 S：规模不超过 S 的任意线路 C 都满足 $\Pr_{x\in_R\{0,1\}^n}[C(x)=f(x)]<\rho$。对于定义在无穷集合上的函数 $f:\{0,1\}^*\to\{0,1\}$，我

⊖ 原文未给出图 19-1 中记号"E"的含义。事实上，$\mathbf{E}=\mathbf{DTIME}(2^{2^{O(n)}})$。——译者注

们令 $H^{p}_{avg}(f)(n)$表示 $H^{p}_{avg}(f_n)$，其中 f_n是 f 在$\{0,1\}^n$上的限制。

我们定义函数 f 的最坏难度 $H_{wrs}(f)$等于 $H^{1}_{avg}(f)$，并且定义 f 的平均难度 $H_{avg}(f)$等于 $\max\{S: H^{1/2+1/S}_{avg}(f)\geqslant S\}$。也就是说，$H_{avg}(f)$是使得"规模不超过 S 的任意线路 C 在长度为 n 的输入上都使 $\Pr_{x\in_R\{0,1\}^n}[C(x)=f(x)]<1/2+1/S$"成立的最大数 S。

注意，任意函数 $f:\{0,1\}^*\to\{0,1\}$都满足 $H_{avg}(f)\leqslant H_{wrs}(f)\leqslant O(2^n/n)$(参见习题 6.1)。定义 19.1 中定义的平均难度也能适用于第 20 章将要讨论的去随机化，并且这种定义只考虑函数输入的均匀分布。对平均复杂性更一般的处理，请参见第 18 章。现在，我们已经完成讨论姚期智 XOR 引理的所有准备工作。

定理 19.2（姚期智 XOR 引理[Yao82a]）　对任意 $f:\{0,1\}^n\to\{0,1\}$，$\delta>0$ 和 $k\in\mathbf{N}$，如果 $\varepsilon>2(1-\delta)^k$，则

$$H^{1/2+\varepsilon}_{avg}(f^{\oplus k})\geqslant\frac{\varepsilon^2}{400n}H^{1-\delta}_{avg}(f)$$

其中 $f^{\oplus k}:\{0,1\}^{nk}\to\{0,1\}$定义为 $f^{\oplus k}(x_1,\cdots,x_k)=\sum_{i=1}^{k}f(x_i)\pmod 2$。

姚期智引理是说，如果较小的线路不能以大于 $1-\delta$ 的概率计算函数 f，则某种较小的线路也不能以大于 $1/2+2(1-\delta)^k$的概率来计算函数 $f^{\oplus k}$。直观上看，如果你只能在比例为 $1-\delta$ 的输入上计算 f，那么给定 k 个随机输入 $x_1,\cdots,x_k$，除非在这些输入上你都恰好能够计算 f(这个随机事件发生的概率为$(1-\delta)^k$)，否则你只能随机猜测 $\sum_{i=1}^{k}f(x_i)\pmod 2$ 的答案，而猜到正确答案的概率至多为 1/2(参见习题 19.1)。然而，要将上述直观理解转换 [375]
为严格的证明还需要花些功夫。证明过程的主要步骤是下面由因帕利亚佐(Impagliazzo)得出的优美结论。

引理 19.3（因帕利亚佐难度核引理(Impagliazzo's Hardcore Lemma)[Imp95a]）　对于$\{0,1\}^n$上的分布 H，如果任意 $x\in\{0,1\}^n$ 均有 $\Pr[H=x]\leqslant 1/(\delta 2^n)$，则称 H 的密度为 δ。对于任意 $\delta>0$，$f:\{0,1\}^*\to\{0,1\}$和 $\varepsilon>0$，如果 $H^{1-\delta}_{avg}(f)\geqslant S$，则存在密度为 δ 的分布 H 使得任意规模不超过$\frac{\varepsilon^2 S}{100n}$的线路 C 都满足

$$\Pr_{x\in_R H}[C(x)=f(x)]\leqslant 1/2+\varepsilon$$

从经验上看，我们可以认为小规模线路仅能以概率 $1-\delta$ 计算的难解函数存在两种可能的形式：(a)难度"分散"于所有输入上，不同的线路在不同的输入上出错，进而任意规模显著的输入子集都大致具有 $1-\delta$ 的计算难度；(b)由比例大致为 δ 的输入构成一个输入子集 H，函数在 H 上是极其难解的(规模较小的任何线路都无法在 H 上以大于$\frac{1}{2}+\varepsilon$ 的概率计算 f，其中 ε 很小)，但函数在 H 之外的其他输入上却易于计算。可以认为，这种 H 集合构成了 f 的难度核(hard core)；因此，有时称之为"难度核集"(Hardcore Set)。因帕利亚佐引理表明，每个难解函数事实上都具有(b)所述的形式。虽然引理用分布来描述而没有使用难度核集，但很容易将它转换为难解核集的形式，参见习题 19.2。

19.1.1　用因帕利亚佐难度核引理证明姚期智 XOR 引理

现在，我们说明如何用因帕利亚佐难度核引理(引理 19.3)来证明姚期智 XOR 引理(定理 19.2)。设 $f:\{0,1\}^n\to\{0,1\}$是满足 $H^{1-\delta}_{avg}(f)\geqslant S$ 的函数，令 $k\in\mathbf{N}$，为了利用反

证法，我们假设线路 C 的规模为 $S'=\frac{\varepsilon^2}{400n}S$ 并且

$$\Pr_{(x_1,\cdots,x_k)\in_R\{0,1\}^{nk}}\left[C(x_1,\cdots,x_k)=\sum_{i=1}^{k}f(x_i)\pmod 2\right]\geqslant 1/2+\varepsilon \tag{19.1}$$

其中 $\varepsilon>2(1-\delta)^k$。我们先证明引理在 $k=2$ 的情况下成立，然后说明如何将证明过程推广到任意 k 上。

令 H 是引理 19.3 得到的密度为 δ 的难度核分布。注意，规模为 S' 的任意线路在 H 上以大于 $1/2+\varepsilon/2$ 的概率无法计算 f。我们用如下的过程来均匀随机地从 $\{0,1\}^n$ 中抽取一个元素。首先，我们投掷一枚头面朝上的概率等于 δ 的偏斜硬币。如果投掷结果是头面朝上，则根据分布 H 随机抽取一个元素；如果投掷结果是背面朝上，则根据分布 G 抽取一个元素，这里 G 是 H 的补分布。也就是说，分布 G 的定义是 $\Pr[G=x]=(2^{-n}-\delta\Pr[H=x])/(1-\delta)$。（习题 19.3 要求读者证明 G 确实是一个分布，并且上述的抽样过程确实均匀地产生了 $\{0,1\}^n$ 中的元素。）我们将上述抽样过程简记为

376

$$U_n=(1-\delta)G+\delta H \tag{19.2}$$

如果我们用 $(U_n)^2$ 表示 $\{0,1\}^{2n}$ 中两个随机独立元素的拼接所服从的分布，则由(19.2)式可知 $(U_n)^2$ 可以记为

$$(U_n)^2=(1-\delta)^2G^2+(1-\delta)\delta GH+\delta(1-\delta)HG+\delta^2H^2 \tag{19.3}$$

其中 G^2 表示服从分布 G 的两个独立元素的拼接，GH 表示服从分布 G 的一个元素与服从分布 H 的另一个独立元素的拼接，其他记号的含义类似。

现在，对于 $\{0,1\}^{2n}$ 上的任意分布 $\mathcal{D}$，令 $P_{\mathcal{D}}$ 表示(19.1)式左端表示的随机事件的概率；也就是说，$P_{\mathcal{D}}$ 表示 $C(x_1,x_2)=f(x_1)+f(x_2)\pmod 2$（其中 x_1，x_2 是服从分布 $\mathcal{D}$ 的随机元素）的概率。联立(19.1)式和(19.3)式，我们得到

$$1/2+\varepsilon\leqslant P_{(U_n)^2}=(1-\delta)^2P_{G^2}+(1-\delta)\delta P_{GH}+\delta(1-\delta)P_{HG}+\delta^2P_{H^2} \tag{19.4}$$

但是，由于 $\varepsilon>2(1-\delta)^2$ 且 $P_{G^2}\leqslant 1$，故(19.4)式意味着

$$1/2+\frac{\varepsilon}{2}\leqslant(1-\delta)\delta P_{GH}+\delta(1-\delta)P_{HG}+\delta^2P_{H^2} \tag{19.5}$$

由于(19.5)式右端的所有系数之和小于 1，故由平均论证法可知(19.5)式右端的三个概率中至少有一个大于(19.5)式的左端。例如，不妨假设 $P_{GH}>1/2+\varepsilon/2$（其他情况是对称的）。这意味着，

$$\Pr_{x_1\in_RH,x_2\in_RG}[C(x_1,x_2)=f(x_1)+f(x_2)\pmod 2]>1/2+\varepsilon/2$$

于是，再由均值论证法可知，存在一个固定的 x_2 使得

$$\Pr_{x_1\in_RH}[C(x_1,x_2)=f(x_1)+f(x_2)\pmod 2]>1/2+\varepsilon/2$$

这等价于

$$\Pr_{x_1\in_RH}[C(x_1,x_2)+f(x_2)\pmod 2=f(x_1)]>1/2+\varepsilon/2$$

但这意味着存在规模为 S' 的线路 D（它计算了映射 $x_1\to C(x_1,x_2)+f(x_2)\pmod 2$）能够以大于 $1/2+\varepsilon/2$ 的概率在服从分布 H 的输入上计算 f，这与 H 是难度核的假设相矛盾！

这就证明了定理在 $k=2$ 时成立。用同样的思路可以证明定理对一般的 k 也成立，此时只需将(19.3)式替换为如下等式

$$(U_n)^k=(1-\delta)^kG^k+(1-\delta)^{k-1}\delta G^{k-1}H+\cdots+\delta^kH^k$$

我们将细节的验证作为习题 19.4 留给读者。 ■

19.1.2　因帕利亚佐难度核引理的证明

下面证明因帕利亚佐难度核引理(引理 19.3)。设函数 f 满足 $H^{1-\varepsilon}_{size^t}(f)\geqslant S$ 且 $\varepsilon>0$。为了证明引理，我们需要找出一个密度为 δ 的分布 H 使得规模为 $S'=\frac{\varepsilon^2 S}{100n}$ 的任意线路 C 都 377
无法在 H 上以大于 $1/2+\varepsilon$ 的概率计算 f。

我们将寻找 H 的过程视为拉塞尔(Russell)和诺姆(Noam)之间的一场博弈。诺姆想要计算函数 f，拉塞尔想让诺姆的计算出现错误。双方的博弈过程如下：拉塞尔先选择一个密度为 δ 的分布 H，然后诺姆再选择一个规模为 S' 的线路。博弈的结果是拉塞尔向诺姆支付 v 美元，其中 $v=\Pr_{x\in_R H}[C(x)=f(x)]$。为了导出矛盾，我们假设引理不成立。于是，对于拉塞尔选择的每个密度为 δ 的分布 H，诺姆总能找到一个大小为 S' 的线路使得 $\Pr_{x\in_R H}[C(x)=f(x)]\geqslant 1/2+\varepsilon$。

注意，上述博弈过程是一个零和博弈，因此我们可以运用冯·诺依曼(von Neumann)的最小-最大定理(参见注记 19.4)。该定理说，如果博弈双方都采用随机策略(也称为混合策略)，则即使让诺姆先选择线路，他仍可以获得同样的收益。这里，随机策略指的是，拉塞尔和诺姆都可以在各自可用的所有策略上任意选择一个概率分布。对拉塞尔而言，这与他原来做法别无二致，因为所有密度为 δ 的分布上的一个分布仍然是一个密度为 δ 的分布[⊖]。但是，对于诺姆而言，我们需要让他在规模为 S' 的所有线路上选择一个分布 $\mathcal{C}$。把我们的假设条件和最小-最大定理结合在一起将意味着：在规模为 S' 的所有线路上*存在*一个分布 $\mathcal{C}$ 使得下面的(19.6)式对任意密度为 δ 的分布 H 均成立。

$$\Pr_{C\in_R\mathcal{C},x\in_R H}[C(x)=f(x)]\geqslant 1/2+\varepsilon \tag{19.6}$$

如果 $x\in\{0,1\}^n$ 使得 $\Pr_{C\in_R\mathcal{C}}[C(x)=f(x)]<1/2+\varepsilon$，则称 x 是"劣性的"；否则，称 x 是"良性的"。劣性 x 的个数小于 $\delta 2^n$，否则，令 H 是所有劣性 x 上的均匀分布，则 H 不满足(19.6)式。现在，我们按如下方式选择一个线路 C：令 $t=50n/\varepsilon^2$，根据分布 $\mathcal{C}$ 随机独立地选择 t 个线路 $C_1,\cdots,C_t$，对于任意 $x\in\{0,1\}^n$，我们定义 $C(x)$ 是 $C_1(x),\cdots,C_t(x)$ 中占多数的那个值。注意，C 的规模是 $tS'<S$。我们断言，如果 C 是按上述方式选取的线路，则在每个良性 $x\in\{0,1\}^n$ 上都有 $\Pr[C(x)\neq f(x)]\leqslant 2^{-n}$。事实上，上述断言可以用切尔诺夫界(Chernoff Bound)(参见推论 A.15)来证明。由于良性 x 的个数至多为 2^n，利用合并界限可以证明：*存在*规模为 S 的线路 C 使得 $C(x)=f(x)$ 对任意良性 x 都成立。又由于劣性 x 的个数小于 $\delta 2^n$，这意味着 $\Pr_{x\in_R U_n}[C(x)=f(x)]>1-\delta$，这与假设条件 $H^{1-\varepsilon}_{size^t}(f)\geqslant S$ 矛盾。■

引理 19.3 的逆否命题表明，如果对于规模显著的任意输入子集，均存在某个线路能够在该子集上以大于 1/2 的概率计算 f，则必然存在一个线路能够在所有输入上以较高概率计算 f。在机器学习中，类似的结果(将函数的弱预测形式转变为该函数的强预测形式)称为学习方法的提升(Boosting)。虽然我们给出的证明不是构造性的，但因帕利亚佐给出的原始证明本身是构造性的，由它构造的提升算法得到了机器学习中的一些新结果，参见[KS99]。 378

⊖ 事实上，密度为 δ 的所有分布构成的集合也可以视为所有 $\delta 2^n$-扁平分布上的所有概率分布构成的集合，其中 K 扁平分布指的是大小为 K 的集合上的均匀分布(参见习题 19.7)。这一事实意味着，我们可以将拉塞尔和诺姆之间的博弈看作是有限的，进而可以将注记 19.4 节给出的最小-最大定理应用到该博弈上。

注记 19.4（最小-最大定理(Min-Max Theorem)）　零和博弈(也称为零和游戏)是由两个参与方共同参与的博弈过程，其中一方输掉的损失恰好等于另一方赢得的收益。零和博弈可以表示为一个 $m \times n$ 的实数矩阵 $A=(a_{i,j})$。博弈过程只有两步，一方(称为最小化方或列参与者)在一个步骤中选择编号 $j \in [n]$；另一方(称为最大化方或者行参与者)在另一个步骤中选择一个编号 $i \in [m]$。博弈的结果是，列参与者必须支付金额为 $a_{i,j}$ 的现金给行参与方(如果 $a_{i,j}$ 是负数，则行参与方要支付金额为 $|a_{i,j}|$ 的现金给列参与方)。显然，博弈双方的操作顺序很重要。但出人意料的是，最小-最大定理断言，如果允许博弈双方采取随机策略，则双方的操作顺序不再重要。

随机策略也称为混合策略，指的是列参与者选择所有列上的一个分布(亦即，一个满足 $\sum_{i=1}^{n} p_i = 1$ 的一个向量 $\boldsymbol{p} \in [0,1]^n$)。类似地，行参与者也选择所有行上的一个分布 $\boldsymbol{q}$。博弈结果的数学期望是 $a_{i,j}$ 的数学期望，其中 j 服从分布 $\boldsymbol{p}$ 且 i 服从分布 $\boldsymbol{q}$。如果我们将 $\boldsymbol{p}$ 视为一个列向量且将 $\boldsymbol{q}$ 视为一个行向量，则博弈结果的数学期望就是 $\boldsymbol{qAp}$。最小-最大定理断言，

$$\min_{\substack{\boldsymbol{p}\in[0,1]^n \\ \sum_i p_i = 1}} \max_{\substack{\boldsymbol{q}\in[0,1]^m \\ \sum_i q_i = 1}} \boldsymbol{qAp} = \max_{\substack{\boldsymbol{q}\in[0,1]^m \\ \sum_i q_i = 1}} \min_{\substack{\boldsymbol{p}\in[0,1]^n \\ \sum_i p_i = 1}} \boldsymbol{qAp} \tag{19.7}$$

正如习题 19.6 中所讨论的那样，最小-最大定理可以用如下的超平面分离定理来证明：如果 C 和 D 是 $\mathbf{R}^n$ 中的两个不相交的凸集，则存在分离这两个集合的超平面。(子集 $C \subseteq \mathbf{R}^n$ 称为凸集，如果 C 包含任意 $\boldsymbol{x}$，$\boldsymbol{y}$ 的同时也必然包含了以 $\boldsymbol{x}$，$\boldsymbol{y}$ 为端点的线段 $\{\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}: 0 \leqslant \alpha \leqslant 1\}$。)习题 19.5 要求读者先证明超平面分离定理(的一种宽松形式)，下面给出了二维空间中超平面分离定理的"图示证明"。



19.2　工具：纠错码

我们的下一个目标是用最坏复杂性下的难解函数直接构造平均复杂性下的难解函数。我们的主要工具是纠错码。纠错码通过将位串映射为更长的位串来实现"差别放大"：也就是说，两个不同的串(即使它们只到某一个位上不同)经过纠错编码之后可能变为"差别很大"的串。纠错码的正式定义如下。

379

定义 19.5（纠错码）　对于 x，$y \in \{0,1\}^n$，定义 x 和 y 之间的分数汉明距离 $\Delta(x,y)$ 等于 $\frac{1}{n}|\{i: x_i \neq y_i\}|$。

对任意 $\delta \in [0,1]$，如果函数 $E: \{0,1\}^n \to \{0,1\}^n$ 使得 $\Delta(E(x), E(y)) \geqslant \delta$ 对任意 $x \neq y \in \{0,1\}^n$ 成立，则称 E 是距离为 δ 的纠错码(Error-Correcting Code，简称 ECC)。集合 $\mathrm{Im}(E)=\{E(x): x \in \{0,1\}^n\}$ 称为 E 的码字集。

注意，有些教科书不是将纠错码定义为函数 $E: \{0,1\}^n \to \{0,1\}^n$，而是将它定义为从 $\{0,1\}^n$ 到 $\{0,1\}^n$ 的一个 2^n 子集(对应于定义 19.5 中的 $\mathrm{Im}(E)$)的函数。纠错码在计算机科学和工程中具有重要的理论和实践意义，这种编码的定义源自如下的简单应用。假设

爱丽丝(Alice)要将位串 $x\in\{0,1\}^n$ 传送给波比(Bob)，但是她与波比之间的通信信道上的噪音使得她发送的任意位串 y 在至多 10% 的二进制位上出现错误。也就是说，她唯一能够确保的是波比收到的位串 y' 满足 $\Delta(y,y')\leqslant 0.1$。为了通过这种信道完成信息传输任务，爱丽丝可以使用距离 $\delta>0.2$ 的纠错码 $E:\{0,1\}^n\to\{0,1\}^m$。她的做法是将 $y=E(x)$ 发送给波比，而后波比收到的位串 y' 将会满足 $\Delta(y,y')\leqslant 0.1$。由于 $\Delta(y,E(w))>0.2$ 对任意 $w\neq x$ 成立，因此在 E 的所有码字中 y 是唯一满足 $\Delta(y,y')\leqslant 0.1$ 的码字。于是，波比可以从 $\mathrm{Im}(E)$ 中找出 y，进而找出满足 $E(x)=y$ 的 x(参见图 19-2)。从这个例子中不难发现，编码距离 δ 应该越大越好，输出长度 m 应该越小越好，并且需要确保通信双方(爱丽丝和波比)各自独立而高效地进行编码和解码。下面的引理表明，如果不考虑计算效率，则存在良好的纠错码。

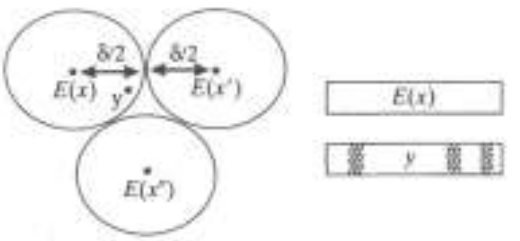


图 19-2：在距离为 δ 的纠错码中，$\Delta(E(x),E(x'))\geqslant\delta$ 对任意 $x\neq x'$ 成立。从错误率低于 $\delta/2$ 的任意 y 上(亦即，$\Delta(y,E(x))<\delta/2$)，我们都可以恢复出 x：因为以各个码字为中心以 $\delta/2$ 为半径的球互不相交。阴影区域表示出错的坐标

引理 19.6　(吉尔伯特-瓦尔沙莫夫界(Gilbert-Varshamov Bound))　*对任意 $\delta<1/2$ 和充分大的 n，存在距离为 δ 的纠错码 $E:\{0,1\}^n\to\{0,1\}^{n/(1-H(\delta))}$，其中 $H(\delta)=\delta\log(1/\delta)+(1-\delta)\log(1/(1-\delta))$。*[⊖]

证明　我们证明引理的稍弱的形式：存在距离为 δ 的纠错码 $E:\{0,1\}^n\to\{0,1\}^m$，其中 $m=2n/(1-H(\delta))$ 而不是 $m=n/(1-H(\delta))$。为此，只需随机选择 E 即可。也就是说，我们随机选取 2^n 个位串 $y_1,y_2,\cdots y_{2^n}\in\{0,1\}^m$，并让 E 将输入 $x\in\{0,1\}^n$ 映射到 y_x(此时，将 x 等同于 $[2^n]$ 中的一个整数)。

只需证明：对任意 $i,j\in[2^n]$，只要 $i\neq j$，则 $\Delta(y_i,y_j)<\delta$ 的概率小于 1。事实上，对于任意 y_i，与 y_i 距离至多为 δ 的位串的个数是 $\binom{m}{\lceil\delta m\rceil}$，当 m 充分大时这个值小于 $0.99\cdot 2^{H(\delta)m}$(参见附录 A)，因此对于任意 $j\neq i$，位串 y_j 与 y_i 距离至多为 δ 的概率不超过 $0.99\cdot 2^{H(\delta)m}/2^m$。由于至多有 2^{2n} 个数对 i,j 满足 $i\neq j$，因此 $\Delta(y_i,y_j)<\delta$ 对所有 $i\neq j$ 成立的概率至多为 $0.99\cdot 2^{2n}\cdot\frac{2^{H(\delta)m}}{2^m}$。因此，我们只需证明 $0.99\cdot 2^{2n}\cdot\frac{2^{H(\delta)m}}{2^m}<1$，根据我们对 m 的选择，该结论成立。通过稍微巧妙一些的论证，我们可以证明引理也成立(参见习题 19.9)。还可以证明，当 δ 趋于 0 时，m 还可以是小于 $n/(1-H(\delta))$ 的值。目前，人们还不清楚的是当 δ 趋于 1/2 时引理 19.6 是不是最优的。■

380

为什么只要求距离是 1/2?

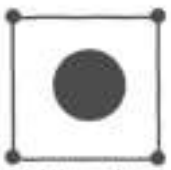
引理 19.6 仅保证了 $\delta<1/2$ 时存在距离为 δ 的纠错码，我们或许会问：这是由于只能得到这种纠错码，还是存在距离更大的纠错码呢？可以证明，距离为 1/2 的纠错码确实存在，但此时要求 m 至少是 n 的指数(亦即，$m\geqslant 2^{n-1}$)。在 $\delta>1/2$ 时，则当 n 充分大时不存在距离为 δ 的纠错码 $E:\{0,1\}^n\to\{0,1\}^m$，无论 m 取多大。习题 19.10 讨论了上述两个结果。

⊖ $H(\delta)$ 称为香农熵函数。不难看到，$H(1/2)=1$，$H(0)=0$，并且 $H(\delta)\in(0,1)$ 对任意 $\delta\in(0,1/2)$ 成立。

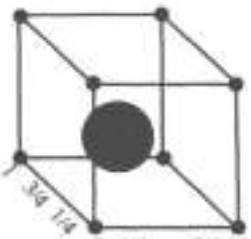
注记 19.7（高维几何）　虽然我们通常只研究平面几何或立体几何，但是我们却能从高维几何中比较直观地认识纠错码。或许，要了解维度对几何造成的最重要的影响，我们可以计算各边长度均为 1 的超方体体积与半径为 1/4 的球体体积之间的比值。在 1 维情况下，该比值等于 $1/(1/2)=2$；在 2 维情况下，该比值等于 $1/(\pi/4^2)=16/\pi$；在 3 维情况下，该比值等于 $1/\left(\frac{4}{3}\pi\frac{1}{4^3}\right)=48/\pi$。随着维数增大，该比值将会以维数的指数形式增长。（在 m 维几何空间中，半径为 r 的球体体积约为 $\frac{\pi^{m/2}}{\lfloor m/2\rfloor!}r^m$。）类似地，在 m 维几何空间中给定两个半径 $r_1>r_2$，则半径为 r_1 的球体体积也将是半径为 r_2 的球体体积的指数倍。



球体体积=1/2



球体体积=π(1/4)²≈3.14/16



球体体积=4/3π(1/4)³≈3.14/48

因此，从直观上看，距离为 1/4 的纠错码需要将长度为 n 的位串映射为长度为 $m=5n$ 的位串。因为只有这样才能确保 $2^{m/5}$ 个码字彼此之间的距离至少是 1/4。这是由于，即使在离散空间中，半径为 1/4 的球体体积（亦即，其中包含的点的个数）也将会指数地小于超方体 $(0,1)^m$ 的体积。这样，该超方体中才能“装下”$2^{m/5}$ 个半径为 1/4 的独立球体。

381

19.2.1　显式纠错码

只证明纠错码的存在性并不能满足大多数应用的要求，我们还需要能够实际地计算出纠错码。为此，我们需要显式地构造出满足如下性质的纠错码 $E:\{0,1\}^n\to\{0,1\}^m$：

编码高效性：存在时间复杂度为 $\mathrm{poly}(m)$ 的算法由 x 计算得到 $E(x)$。

解码高效性：存在一个多项式时间算法由满足 $\Delta(y,E(x))<\rho$（ρ 是一个具体的数）的任意 y 计算得到 x。为了保证这一点，数 ρ 必须小于 $\delta/2$，其中 δ 是纠错码 E 的距离，参见习题 19.11。

下面，我们陆续给出一些显式的纠错码。

19.2.2　沃尔什-哈达玛纠错码

对任意两个位串 $x,y\in\{0,1\}^n$，我们定义 $x\odot y=\sum_{i=1}^{n}x_iy_i\pmod 2$。沃尔什-哈达玛纠错码（Walsh-Hadamard Code）指的是函数 $\mathrm{WH}:\{0,1\}^n\to\{0,1\}^{2^n}$，它将任意位串 $x\in\{0,1\}^n$ 映射为位串 $z\in\{0,1\}^{2^n}$ 使得 $z_y=x\odot y$ 对任意 $y\in\{0,1\}^n$ 成立，其中 z_y 是 z 的第 y 个分量，并且通过某种标准方法将 $\{0,1\}^n$ 等同于 $[2^n]$。

论断 19.8　函数 WH 是距离为 1/2 的纠错码。

证明　首先，注意到 WH 是一个线性映射。也就是说，$\mathrm{WH}(x+y)=\mathrm{WH}(x)+\mathrm{WH}(y)$，其中 $x+y$ 表示将 x 和 y 的对应分量相加然后对 2 取模（亦即，在对应分量上执行 XOR 操作）。因此，对于任意 $x\neq y\in\{0,1\}^n$，位串 $\mathrm{WH}(x)+\mathrm{WH}(y)=\mathrm{WH}(x+y)$ 中

1 的个数等于 WH(x)和 WH(y)的对应分量取不同值的坐标个数。因此，只需证明：对于任意 $w\neq 0^n$，WH(w)中至少有一半的坐标等于 1。但这可以由随机子和原理(论断 A.31)保证，因为该原理说：当 $y\in_R\{0,1\}^n$ 时，$w\odot y=1$ 的概率恰好是 1/2。 ■

19.2.3　里德-所罗门纠错码

沃尔什-哈达玛纠错码存在严重的缺陷：其输出的长度是输入长度的指数。由引理 19.6可知，我们可以找到更好的纠错码(至少能找到距离稍小于 1/2 的纠错码)。为了得到更好的编码，我们利用非二进制字母表上的纠错码作为跳板。

定义 19.9　对于任意有限集 $\sum$ 和 $x,y\in\sum^n$，定义 $\Delta(x,y)=\frac{1}{n}|\{i:x_i\neq y_i\}|$。如果函数 $E:\sum^n\to\sum^m$ 使得 $\Delta(E(x),E(y))\geqslant\delta$ 对任意 $x\neq y\in\sum^n$ 成立，则称 E 是字母表 $\sum$ 上距离为 δ 的纠错码。

较大的字母表将使得纠错码的构造变得更容易一些。例如，二进制字母表{0，1}上的距离为 δ 的任意纠错码可以自动地产生字母表{0，1，2，3}上具有相同距离的一个纠错码。这只需将{0，1，2，3}中的每个字符串以显而易见的方法编码为{0，1}上的位串即可。但是，反过来却不行。事实上，如果将{0，1，2，3}上的纠错码以自然的方式转换成 382
{0，1}上的一个纠错码，则编码距离可能从 δ 变为 2δ(参见习题 19.12)。里德-所罗门纠错码(Reed-Solomon Code)允许在充分大的任意域 $\mathbf{F}$ 上构造纠错码。

定义 19.10　(里德-所罗门纠错码)　设 $\mathbf{F}$ 是一个域且整数 n，m 满足 $n\leqslant m\leqslant|\mathbf{F}|$。里德-所罗门纠错码(Reed-Solomon Code)指的是函数 RS：$\mathbf{F}^n\to\mathbf{F}^m$，它在输入 $a_0,\cdots,a_{n-1}\in\mathbf{F}$ 上输出字符串 $z_0,\cdots,z_{m-1}\in\mathbf{F}^m$，其中 $z_i=\sum_{j=0}^{n-1}a_jf_i^j$，而 f_j 是 $\mathbf{F}$ 的第 j 个元素($\mathbf{F}$ 的元素以某种方式排定顺序)。

注意，里德-所罗门纠错码的另一种等价的定义是：它的输入是 $n-1$ 次多项式 $A(x)=\sum_{i=0}^{n-1}a_ix^i$ 的某种表示，其输出是多项式 A 分别在点 $f_0,\cdots,f_{m-1}$ 上的取值。

引理 19.11　里德-所罗门纠错码 RS：$\mathbf{F}^n\to\mathbf{F}^m$ 的编码距离为 $1-\frac{n}{m}$。

证明　同沃尔什-哈达玛纠错码一样，里德-所罗门纠错码也是线性的。也就是说，RS($a+b$)=RS(a)+RS(b)，其中加法指的是在对应分量上执行 $\mathbf{F}$ 的加法。因此，同论断 19.8 的证明一样，我们只需证明：对任意 $a\neq 0^n$，RS(a)至多有 n 个分量等于 0。但这是显然的，因为 $n-1$ 次非零多项式至多存在 n 个根(参见附录 A)。 ■

19.2.4　里德-穆勒纠错码

下面介绍一族称为里德-穆勒码(Reed-Muller Code)的纠错码。前面的沃尔什-哈达玛纠错码和里德-所罗门纠错码都是这族编码的特例。

定义 19.12　(里德-穆勒纠错码)　设 $\mathbf{F}$ 是一个有限域，ℓ，d 是两个整数且 $d<|\mathbf{F}|$。参数为 $\mathbf{F}$，ℓ，d 的里德-穆勒纠错码指的是函数 RM：$\mathbf{F}^{\binom{\ell+d}{d}}\to\mathbf{F}^{|\mathbf{F}|^\ell}$，它将 $\mathbf{F}$ 上 ℓ 个变量的总次数为 d 的多项式 P 映射为 P 在 $\mathbf{F}^\ell$ 的所有点上的取值。

也就是说，里德-穆勒纠错码的输入是如下形式的多项式

$$P(x_1,\cdots,x_t)=\sum_{i_1+\cdots+i_t\leqslant d} c_{i_1,\cdots,i_t}x_1^{i_1}\cdots x_t^{i_t}$$

的$\binom{\ell+d}{d}$个系数$(c_{i_1,\cdots,i_t})$，而输出则是该多项式在所有$x_1,\cdots,x_t\in\mathbf{F}$上的函数值构成的集合$\{P(x_1,\cdots,x_t)\}$。

在里德-穆勒纠错码中，令$\ell=1$则得到里德-所罗门纠错码($m=|\mathbf{F}|$的情形)；而令$d=1$且$\mathbf{F}=\mathrm{GF}(2)$则得到沃尔什-哈达玛纠错码的一种变形，此时任意$x\in\{0,1\}^n$被映射为长度为$2\cdot 2^n$的位串$z$，其中$z_{y,a}=x\odot y+a(\mathrm{mod}\ 2)$对任意$y\in\{0,1\}^n$和$a\in\{0,1\}$成立。施瓦兹-兹佩尔引理(Schwartz-Zippel Lemma，即引理A.36)表明，里德-穆勒纠错码是编码距离为$1-d/|\mathbf{F}|$的纠错码。注意，这同时也表明前面所得的沃尔什-哈达玛纠错码和里德-所罗门纠错码的编码距离都是正确的。

383

19.2.5　拼接纠错码

沃尔什-哈达玛纠错码的缺陷在于它的输出具有指数长度，而里德-所罗门纠错码的缺陷在于它的字母表不是二进制字母表。下面说明，二者的组合可以同时克服它们各自的缺陷。

定义19.13　如果里德-所罗门纠错码RS将$\mathbf{F}^n$映射到$\mathbf{F}^m$(对某个n，m，$\mathbf{F}$成立)，并且沃尔什-哈达玛纠错码WH将$\{0,1\}^{\log|\mathbf{F}|}$映射到$\{0,1\}^{2^{\log|\mathbf{F}|}}=\{0,1\}^{|\mathbf{F}|}$，则定义WH∘RS按如下方式将$\{0,1\}^{n\log|\mathbf{F}|}$映射到$\{0,1\}^{m|\mathbf{F}|}$：

1. 通过某种规范映射将$\mathbf{F}$中的所有元素分别表示为$\{0,1\}^{\log|\mathbf{F}|}$上的一个位串，这样RS就可以视为从$\{0,1\}^{n\log|\mathbf{F}|}$到$\mathbf{F}^m$的纠错码，而WH则可以视为从$\mathbf{F}$到$\{0,1\}^{|\mathbf{F}|}$的纠错码。

2. 在任意输入$x\in\{0,1\}^{n\log|\mathbf{F}|}$上，WH∘RS$(x)$等于$\mathrm{WH}(\mathrm{RS}(x)_1),\cdots,\mathrm{WH}(\mathrm{RS}(x)_m)$，其中$\mathrm{RS}(x)_i$表示$\mathrm{RS}(x)$的第$i$个符号。

注意，纠错码WH∘RS的计算时间是n，m和$|\mathbf{F}|$的多项式。下面，我们分析它的编码距离。

论断19.14　设$\delta_1=1-n/m$表示RS的编码距离并且$\delta_2=1/2$表示WH的编码距离，则WH∘RS是编码距离为$\delta_1\delta_2$的纠错码。

证明　设x，y是$\{0,1\}^{n\log|\mathbf{F}|}$中的两个不同的位串。如果令$x'=\mathrm{RS}(x)$，$y'=\mathrm{RS}(y)$，则$\Delta(x',y')\geqslant\delta_1$。如果再令$x''$(相应地，$y''$)表示将WH依次作用到$x'$(相应地，$y'$)的各个符号上之后再拼接在一起得到的位串，只要两个区组不相同，则对应区组经WH编码之后的距离将等于1/2，因此$\Delta(x'',y'')\geqslant\delta_1\delta_2$。■

由于对任意$k\in\mathbf{N}$均存在一个大小属于$[k,2k]$的有限域$\mathbf{F}$(例如，取$[k,2k]$中的一个素数或者2的幂次)，因此，利用上述事实，我们可以为任意n构造一个多项式时间可计算的纠错码$E:\{0,1\}^n\to\{0,1\}^{2m^2}$使得$E$的编码距离等于0.4。

定义19.13和论断19.14不仅仅对里德-所罗门纠错码和沃尔什-哈达玛纠错码成立，它们都可以推广到其他纠错码上。也就是说，对于任意两个纠错码$E_1:\{0,1\}^n\to\Sigma^m$和$E_2:\Sigma\to\{0,1\}^s$，则二者的拼接$E_2\circ E_1$是从$\{0,1\}^n$到$\{0,1\}^{ms}$的编码距离为$\delta_1\delta_2$的纠错码，其中δ_1和δ_2分别是E_1和E_2的编码距离，参见图19-3。特别地，将定义19.13中的WH换成二进制字母表上的其他纠错码，则可以构造出一个多项式时间可计算的纠错码$E:\{0,1\}^n\to\{0,1\}^m$使得它具有常数编码距离$\delta>0$并且$m=O(n)$，参见习题19.18。

384

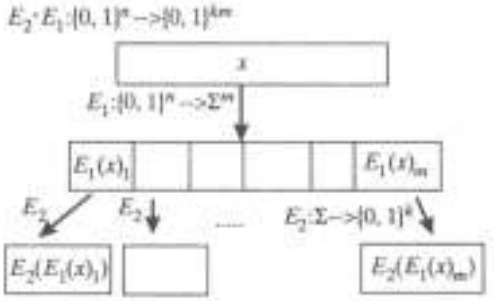


图 19-3　如果 E_1，E_2 分别是形如 $E_1 : \{0,1\}^n \to \Sigma^m$ 和 $E_2 : \Sigma \to \{0,1\}^k$ 的纠错码，则它们的拼接编码 $E : \{0,1\}^n \to \{0,1\}^{km}$ 将每个 x 映射为一个区组序列 $E_2(E_1(x)_1)$，…，$E_2(E_1(x)_m)$

19.3　高效解码

要将纠错码切实地用于信息存储和传输，我们需要有高效的方法来将编码结果 $E(x)$ 解码为 x，而且还要求即使编码 $E(x)$ 在比例为 ρ 的位置上出现错误的情况下解码过程仍能正确进行。下面，我们说明如何在里德-所罗门纠错码和拼接纠错码上实现高效解码。

19.3.1　里德-所罗门解码

注意，里德-所罗门编码以一个多项式作为输入，以这个多项式在 m 个点上的取值作为输出。由定理 A.35 可知，一元 d 次多项式可以通过在 $d+1$ 个点上的插值得到。这里，我们将使用插值过程的一种健壮形式：亦即，当多项式在 m 个点上的所有取值中有 ρm 个值是"错的"或"带噪音的"，我们仍然希望通过插值能够恢复出多项式。

定理 19.15（里德-所罗门唯一解码[BW86]）　*存在一个多项式时间算法使得：它以有限域 $\mathbf{F}$ 上的元素对 (a_1, b_1)，…，(a_m, b_m) 为输入，其中存在 $t > \frac{m}{2} + \frac{d}{2}$ 个 $i \in [m]$ 使得相应的序对在某个 d 次多项式 $G : \mathbf{F} \to \mathbf{F}$ 上满足 $G(a_i) = b_i$，算法的输出是 G。*

由于里德-所罗门纠错码的编码距离是 $1 - \frac{d}{m}$，故定理 19.15 意味着：我们可以高效地从编码中恢复得到正确的多项式，即使这些编码在比例为 ρ 的位置上出现了错误。只要 ρ 小于编码距离的 1/2。一旦错误率超过编码距离的 1/2，则我们不能再确保解的唯一性。从这个意义上讲，里德-所罗门解码是最优的。

定理 19.15 的证明　作为预备工作，我们先考虑错误率非常小的情况。（在这种情况下得到的解码过程已经足以支持很多应用。）

*随机插值：$t \geqslant \left(1 - \frac{1}{2(d+1)}\right)m$ 的情形。假设 t 非常大，$t > \left(1 - \frac{1}{2(d+1)}\right)m$。此时，*我们可以从集合 $\{(a_i, b_i)\}$ 中随机挑选 $d+1$ 个序对 (x_1, y_1)，…，(x_{d+1}, y_{d+1})，用标准的多项式插值方法计算唯一的一个多项式 P 使得 $P(x_j) = y_j$ 对所有 $j \in [d+1]$ 都成立。然后，我们再验证 P 是否使得 $\{(a_i, b_i)\}$ 中的至少 t 个序对满足 $P(a_i) = b_i$。如果确实如此，则输出 P（否则，在新的随机选择上重新计算 P）。由合并界限可知，$G(x_j) \neq y_j$ 在随机选择的所有 $d+1$ 个序对中的某一个上不成立的概率至多为 $(d+1)\frac{m-t}{t} \leqslant 1/2$。因此解码过程得到 $P = G$ 的概率至少为 1/2。

波利坎普-韦尔奇过程(Berlekamp-Welch Procedure)：$t \geqslant \frac{m}{2}+\frac{d}{2}+1$ 的情形。现在，我们用广为人知的波利坎普-韦尔奇解码来证明定理 19.15。为使记号简单，我们假设 $m=4d$ 且 $t=3d$。但是，证明过程可以推广到满足 $t>\frac{m}{2}+\frac{d}{2}$ 的任意参数 m，d，t 上，参见
385 习题 19.13。于是，定理的题设为：有一个 d 次多项式 G 使得

$$G(a_i)=b_i \text{ 在 } [m]=[4d] \text{ 中的至少 } 3d \text{ 个 } i \text{ 上成立} \tag{19.8}$$

我们使用如下的解码过程：

1. 找出一个 $2d$ 次多项式 $C(x)$ 和一个 d 次非零多项式 $E(x)$ 使得

$$C(a_i)=b_i E(a_i) \text{ 对任意 } i \in [m] \text{ 都成立} \tag{19.9}$$

这可以如下完成：将 $C(x)$ 的 $2d+1$ 个系数和 $E(x)$ 的 d 个系数都视为未知数，根据(19.9)式建立含有 $4d$ 个线性方程的方程组。该方程组存在一个解使得 $E(x)$ 是非零多项式，因为我们可以通过在 $G(a_i)\neq b_i$ 的 a_i 上要求 $E(a_i)=0$ 来定义一个非零多项式 $E(x)$(根据我们的假设，这样的 a_i 至多有 d 个)。[⊖]

2. 用 E 除 C 得到一个多项式 P 使得 $C(x)=P(x)E(x)$(下面会证明除法不会产生余式)，输出 P。

根据(19.8)式和(19.9)式可知，$C(x)=G(x)E(x)$ 至少在 $3d$ 个 a_i 上成立。这意味着 $2d$ 次多项式 $C(x)-G(x)E(x)$ 至少有 $3d$ 个根，进而 $C(x)-G(x)E(x)$ 只能是零多项式(亦即，$C(x)=G(x)E(x)$ 对任意 $x\in\mathbb{F}$ 成立)。因此，$G=C/E$ 确实成立。 ■

19.3.2　拼接解码

拼接纠错码可以用比较自然的算法来解码。注意，如果 $E_1:\{0,1\}^{\ell}\to\Sigma^n$ 和 $E_2:\Sigma\to\{0,1\}^{\ell}$ 是两个纠错码，则 $E_2\circ E_1$ 将任意 $x\in\{0,1\}^{\ell n}$ 映射为 $E_2(E_1(x)_1)$，…，$E_2(E_1(x)_n)$。假设 E_1 和 E_2 各自的解码算法能够分别处理错误率 ρ_1 和 ρ_2，则 $E_2\circ E_1$ 存在一个能够处理错误率 $\rho_2\rho_1$ 的解码算法。该解码算法如下工作。给定 $y\in\{0,1\}^{n\ell}$，记 y 的 n 个区组分别是 y_1，…，$y_n\in\{0,1\}^{\ell}$，先用 E_2 的解码算法依次将每个区组 y_i 解码为另一个符号 z_i；然后，再用 E_1 的解码算法将 z_1，…，z_n 解码。$E_2\circ E_1$ 的这个解码算法确实能够处理错误率 $\rho_2\rho_1$。这是由于，如果 $\Delta(y,E_2\circ E_1(x))\leqslant\rho_2\rho_1$，则 y 中至多有 ρ_1 比例的区组与 $E_2\circ E_1(x)$ 中相应区组之间的距离达到 ρ_2 以上。 ■

19.4　局部解码与难度放大

现在，我们说明纠错码与难度放大之间的联系。思路其实很简单(参见图 19-4)。函数 $f:\{0,1\}^n\to\{0,1\}$ 可以视为一个长度为 $N=2^n$ 的位串。假设我们用从 $\{0,1\}^N$ 到 $\{0,1\}^M$ 的一个纠错码将函数 f 编码成一个位串 $\hat{f}\in\{0,1\}^M$，不妨设纠错码的编码距离大于 0.2。则 $\hat{f}$ 可以视为从 $\{0,1\}^{\log M}$ 到 $\{0,1\}$ 上的一个函数。原则上讲，我们至少要能够从存在一些
386 错误(不妨设错误率为 10%)的 $\hat{f}$ 恢复出原来的函数 f。换句话说，如果我们能在 90% 的输入上正确计算 $\hat{f}$，则我们应该可以在所有输入上正确地计算 f。其逆否命题意味着，如果 f 在最坏复杂性下是难计算的，则 $\hat{f}$ 在平均复杂性下也是难计算的。

⊖ 为了从方程组中高效地解得所需的多项式，我们可以尝试向方程组添加一个方程 $E_i=\epsilon_i$，其中 E_i 是 $E(x)$ 中 x^i 的系数而 ϵ_i 是 $\mathbb{F}$ 中的一个非零元素。形如 $E_i=\epsilon_i$ 的方程共有多项式个，其中之一必然能够帮助我们从方程组中解得所需的多项式。

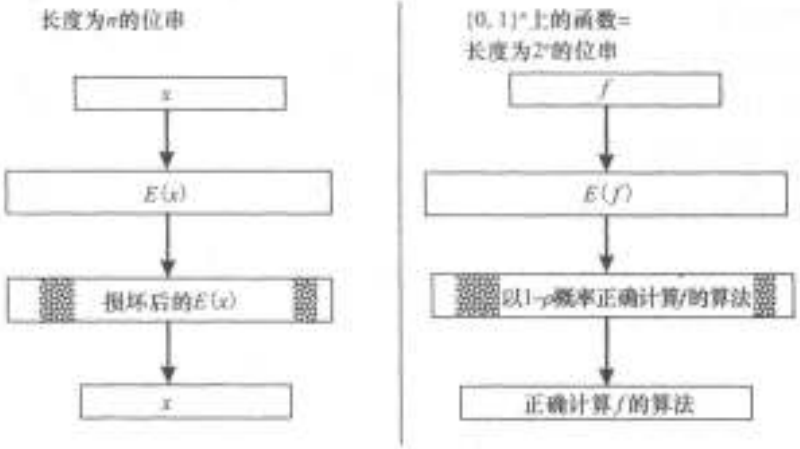


图 19-4　纠错码允许我们将位串 x 映射为 $E(x)$，并且我们还可以从损坏后的 $E(x)$ 中恢复出 x。要用这种思路来处理函数，先将函数 $f:\{0,1\}^n\to\{0,1\}$ 视为 $\{0,1\}^{2^n}$ 上的一个位串，用纠错码将它编码为另一个函数 $\hat{f}$。直观上，如果 f 在最坏复杂性下是难解的，则 $\hat{f}$ 在平均复杂性下也是难解的。因为能在 $1-\rho$ 比例的输入上正确计算 $\hat{f}$ 的任意算法都可以转换为能在所有输入上正确计算 f 的算法

要使上述思路切实可行，我们需要说明：能够将“在大多数输入上能够正确计算 $\hat{f}$ 的任何线路”转换为“在所有输入上能够正确计算 f 的线路”。这种转换过程可以形式地描述为*局部解码算法*（参见图 19-5）。局部解码算法是纠错码的一种解码算法，它可以通过对（可能已经损坏的）码字 y' 进行随机访问（其中 y 与 $E(x)$ 的非常接近）来计算原始输入 x 中的任何指定的二进制位。由于我们感兴趣的仅仅是规模为 $\mathrm{poly}(n)$ 的线路（换句话说，线路的规模是 $N=2^n$ 的对数多项式），因此，局部解码算法的运行时间也必须是 $\mathrm{poly}(n)$ 或 N 的对数多项式。

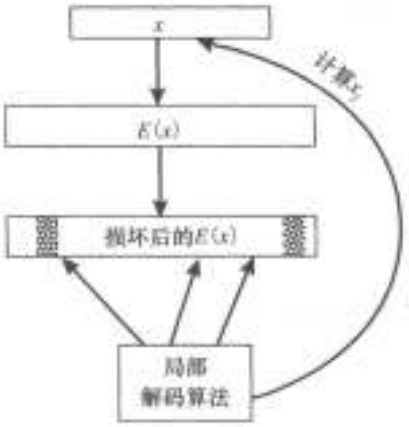


图 19-5　局部解码算法利用损坏后的 $E(x)$ 和下标 j 高概率地计算得到 x_j

387

定义 19.16（局部解码算法）　设 $E:\{0,1\}^n\to\{0,1\}^m$ 是一个纠错码，并且 ρ 和 q 是两个任意数。E 的处理错误率 ρ 的局部解码算法是这样的，它的输入是可以随机访问的任意 y（其中 $\Delta(y,E(x))<\rho$ 对某个未知的 $x\in[2^n]$ 成立）和一个下标 $j\in[n]$，算法能够在 $\mathrm{poly}(m)$ 时间内至少以概率 2/3 输出 x_j。

常数 2/3 是任意的，它可以替换为大于 1/2 的任意常数，这是由于，正确计算函数的概率可以通过重复执行计算来提高。同时，我们还注意到，定义 19.16 可以很容易推广到具有更大字母表（例如，非二进制字母表）的纠错码上。纠错码的局部解码在许多与难度放大毫无关系的应用中也十分有用（例如，如果用纠错码来编码一个超大的文件，则我们可以高效地解码文件的一部分而不需要将整个文件整体解码出来）。下面的定理强调了局部解码与难度放大之间的联系。

定理 19.17（由局部解码得到难度放大）　假设存在一个纠错码，它有多项式时间的编码算法和一个处理错误率 ρ 的局部解码算法。还假设 $f\in\mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 满足

$H_{wc}(f)(n) \geqslant S(n)$对某个满足 $S(n) \geqslant n$ 的函数 $S: \mathbf{N} \to \mathbf{N}$ 成立。那么，存在 $\varepsilon > 0$ 和 $\tilde{f} \in \mathbf{E}$ 使得 $H_{wc}^{\tilde{f}}(\tilde{f})(n) \geqslant S(\varepsilon n)$。

定理 19.17 的证明可以根据前面的基本思路得到，我们将其留作习题 19.14。下面，我们给出几个显式纠错码的局部解码算法。

19.4.1　沃尔什-哈达玛纠错码的局部解码算法

下面是沃尔什-哈达玛纠错码的一个局部解码算法，它通过两次查询实现局部解码，并且能够处理任意错误率 ρ（其中 $\rho < 1/4$）。这是我们能够处理的最佳错误率。事实上，不难证明：任意二进制纠错码不存在能够处理错误率 $\rho \geqslant 1/4$ 的（局部）解码算法。

定理 19.18　*对任意 $\rho < 1/4$，沃尔什-哈达玛纠错码存在能够处理错误率 ρ 的局部解码算法。*

证明　下面的算法证明了定理 19.18 的正确性。

沃尔什-哈达玛局部解码算法：$\rho < 1/4$

输入：$j \in [n]$，对函数 $f: \{0,1\}^n \to \{0,1\}$ 的随机访问能力，其中 $\Pr_y[f(y) \neq x \odot y] \leqslant \rho$ 对某个 $\rho < 1/4$ 和某个 $x \in \{0,1\}^n$ 成立。

输出：二进制位 $b \in \{0,1\}$（我们的目标是 $x_j = b$）。

操作：令 e^j 是 $\{0,1\}^n$ 中的一个向量，它的第 j 个坐标分量等于 1 而其余坐标分量全部为 0。算法随机选择 $y \in_R \{0,1\}^n$，然后输出 $f(y) + f(y + e^j) \pmod 2$（其中 $y + e^j$ 表示分别将 y 和 e^j 的对应坐标相加之后对 2 取模，等价的说法是，将 y 的第 j 个坐标翻转）。

分析：由于 y 和 $y + e^j$ 都服从均匀分布（尽管它们是相互依赖的），合并界限表明 $f(y) = x \odot y$ 和 $f(y + e^j) = x \odot (y + e^j)$ 同时成立的概率至少为 $1 - 2\rho$。再由 $\odot$ 操作的双线性可知，$f(y) + f(y + e^j) = x \odot y + x \odot (y + e^j) = 2(x \odot y) + x \odot e^j = x \odot e^j \pmod 2$。但另一方面，$x \odot e^j = x_j$。因此，算法输出正确解的概率至少为 $1 - 2\rho$。（该概率可以通过重复运行算法来提高）。

388

沃尔什-哈达玛解码算法经过修改之后，不仅仅可以用于计算 $x_j = x \odot e^j$，而且还可以用来为任意 $z \in \{0,1\}^n$ 计算 $x \odot z$ 的值。因此，该算法不仅可以用来计算原始信息 x 中的每个二进制位，而且还可以用来计算未损坏的码字 $\mathrm{WH}(x)$。这种性质有时也称为沃尔什-哈达玛纠错码的自纠错性。 ■

19.4.2　里德-穆勒纠错码的局部解码算法

本小节给出里德-穆勒纠错码的局部解码算法，其运行时间是 ℓ 和 d 的多项式。因此，恰当设置参数之后，里德-穆勒局部解码算法的运行时间是编码输出长度的多项式。

定理 19.19　*在任意域 $\mathbf{F}$ 和任意整数 d，ℓ 上，参数为 $\mathbf{F}$，d，ℓ 的里德-穆勒纠错码存在时间复杂度为 $\mathrm{poly}(|\mathbf{F}|, \ell, d)$ 的处理错误率 $\left(1 - \frac{d}{|\mathbf{F}|}\right)/6$ 的局部解码算法。*

也就是说，存在一个时间复杂度为 $\mathrm{poly}(|\mathbf{F}|, \ell, d)$ 的算法 D，它的输入包括：(a) 可以随机访问的函数 $f: \mathbf{F}^\ell \to \mathbf{F}$，其中 f 与某个 d 次多项式 P 在 $1 - \left(1 - \frac{d}{|\mathbf{F}|}\right)/6$ 比例的输入上是一致的；(b) $x \in \mathbf{F}^\ell$，算法至少以 2/3 的概率输出 $P(x)$。

证明　注意，里德-穆勒纠错码的输入是 $\mathbf{F}$ 上 ℓ 个变量的 d 次多项式 P。在前面的讨

论中，我们曾假设 P 表示为它的系数序列。但是，在下面的讨论中，为方便计，我们假设所有输入点按某种标准方法排序，而多项式 P 用它在前 $\binom{d+\ell}{\ell}$ 个点上的取值来描述。利用标准的插值方法，即使在这种表示方法下，我们仍能够得到里德-穆勒纠错码的多项式时间的编码算法。因此，为了证明定理 19.19，只需给出一个算法使得它能够通过对损坏的 P 的随机访问来计算 P 在任意 $x\in\mathbf{F}^\ell$ 上的值。现在，我们给出这样一个算法。

里德-穆勒局部解码算法：$\rho\leqslant\left(1-\frac{d}{|\mathbf{F}|}\right)/6$

输入：字符串 $x\in\mathbf{F}^\ell$，对函数 $f:\{0,1\}^*\to\{0,1\}$ 的随机访问，其中 $\Pr_{z\in\mathbf{F}^\ell}[f(x)\neq P(x)]<\rho$ 对 ℓ 个变量上的某个 d 次多项式 $P:\mathbf{F}^\ell\to\mathbf{F}$ 成立。

输出：$y\in\mathbf{F}$（我们的目标是 $y=P(x)$）

操作：

1. 令 L_z 是通过 x 的一条随机的直线。亦即，$L_z=\{x+tz:\ t\in\mathbf{F}\}$ 对某个随机选取的 $z\in\mathbf{F}^\ell$ 成立。

2. 在 L_z 的所有 $|\mathbf{F}|$ 个点上分别查询 f 的函数值，得到点集 $\{(t,\ f(x+tz))\}_{t\in\mathbf{F}}$；

3. 调用里德-所罗门解码算法得到一元多项式 $Q:\mathbf{F}\to\mathbf{F}$ 使得 $Q(t)=f(x+tz)$ 对尽量多的 t 成立（参见图 19-6）。[⊖]

4. 输出 $Q(0)$。

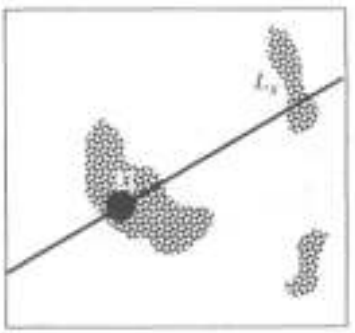


图 19-6　给定对多项式 $P:\mathbf{F}^\ell\to\mathbf{F}$ 的损坏形式的随机访问能力，为了计算 $P(x)$，我们利用里德-所罗门解码算法恢复 $P(x)$ 在一条通过 x 的直线 L_z 上的限制

分析：对于 ℓ 个变量上的任意 d 次多项式 P，满足 $Q(t)=P(x+tz)$ 的一元多项式至多为 d 次多项式。因此，证明里德-穆勒局部解码算法的正确性，只需证明“L_z 上满足 $f(w)\neq P(w)$ 的点 $w\in L_z$ 的个数小于 $(1-d/|\mathbf{F}|)/2$”的概率至少为 2/3。事实上，对于任意 $t\neq0$，当 z 是从 $\mathbf{F}^\ell$ 中均匀随机地选取时，点 $x+tz$ 也服从 $\mathbf{F}^\ell$ 上的均匀分布并且独立于 x，因此 L_z 上满足 $f(w)\neq P(w)$ 的点 $w\in L_z$ 的个数的数学期望不超过 $\rho|\mathbf{F}|$。由马尔科夫不等式可知，“L_z 上满足 $f(w)\neq P(w)$ 的点 $w\in L_z$ 的个数超过 $3\rho|\mathbf{F}|<(1-d/|\mathbf{F}|)|\mathbf{F}|/2$”的概率至多为 2/3。于是，里德-所罗门解码算法正确的概率至少为 2/3。当里德-所罗门解码算法正确时，里德-穆勒局部解码算法获得了 Q 限制在直线 L_z 上产生的多项式 q，因此 $q(0)=P(x)$。■

389

19.4.3　拼接纠错码的局部解码算法

下面的引理表明，给定两个可以局部解码的纠错码 E_1 和 E_2，则二者的拼接纠错码 $E_1\circ E_2$ 也存在局部解码算法。

引理 19.20　设 $E_1:\{0,1\}^*\to\Sigma^*$ 和 $E_2:\Sigma\to\{0,1\}^*$ 是两个纠错码，它们的局部解码算法分别对函数进行了 q_1 和 q_2 次随机访问并且能够处理的错误率分别是 ρ_1 和 ρ_2。则纠错码 $E=E_2\circ E_1:\{0,1\}^*\to\{0,1\}^*$ 也存在局部解码算法使得该解码算法仅对函数进行 $O(q_1q_2\log q_1\log|\Sigma|)$ 次随机访问并且能够处理错误率 $\rho_1\rho_2$。

⊖　如果 ρ 足够小（比如，$\rho<1/(10d)$），则我们可以用 19.3 节中更简单的随机的里德-所罗门解码算法。

证明　我们用自然的算法来证明引理。也就是说，$E_2 \circ E_1$的解码算法调用E_1的解码算法，而在对E_1的随机访问过程中再调用E_2的解码算法(参见图 19-7)。

拼接纠错码的局部解码算法：$\rho \leqslant \rho_1\rho_2$

输入：标号$i \in [n]$，对位串$y \in \{0,1\}^m$的随机访问能力，其中$\Delta(y, E_2 \circ E_1(x)) < \rho_1\rho_2$对某个$x \in \{0,1\}^n$成立。

输出：$b \in \{0,1\}$(我们的目标是$b = x_i$)

操作：模拟E_1的解码算法的运行过程，一旦解码算法需要随机访问$E_1(x)$的第j个区组时，再调用E_2的解码算法$O(q_2 \log q_1 \log|\Sigma|)$次以不小于$1-1/(10q_1)$的概率恢复得到第$j$个区组的每个二进制位。

390

分析：一个重要事实是，y的每个区组的长度为k，且y中至多仅有ρ_1比例的区组与$E_2 \circ E_1(x)$的相应区组之间的距离大于ρ_2。因此，E_1的解码算法进行q_1次随机访问时所获得的答案与它直接随机访问"距离E_1的码字不超过ρ_1的任何位串"时所得的答案一致的概率至少为 0.9。

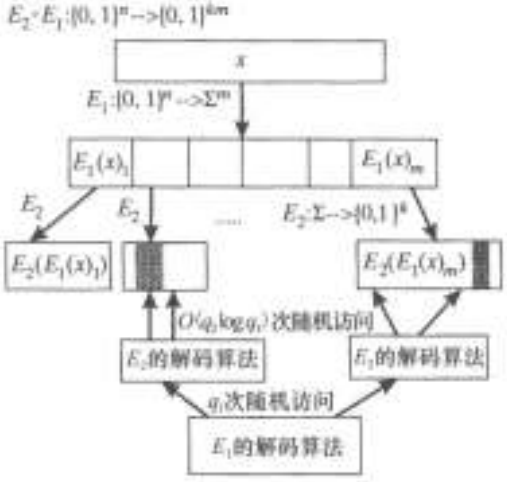


图 19-7　为了对$E_2 \circ E_1$进行局部解码，我们调用E_1的解码算法，并且在E_1的解码算法中再调用E_2的解码算法。需要注意如下的重要事实，如果y与$E_2 \circ E_1(x)$的距离小于$\rho_1\rho_2$，则y中至多仅有ρ_1比例的区组与$E_2 \circ E_1(x)$的相应区组之间的距离大于ρ_2

■

19.4.4　局部解码算法综合运用于难度放大

做好了所有准备工作，现在我们证明本章的第二个主要结果：最坏复杂性下的难解函数可以转换为平均复杂性下具有"温和难度"的难解函数。

定理 19.2　(最坏难度转换为温和难度)　若$S_1: \mathbf{N} \to \mathbf{N}$和$f \in \mathbf{E} = \mathbf{DTIME}(2^{(n+1)})$使得$H_{wc}(f)(n) \geqslant S(n)$对任意$n$成立，则存在函数$g \in \mathbf{E}$和常数$c > 0$使得$H^{1-1/n}_{wc}(g)(n) \geqslant S(n/c)/c$对充分大的$n$成立。

证明　对任意的n，我们将函数f在$\{0,1\}^n$上的限制视为一个位串$f' \in \{0,1\}^N$，其中$N = 2^n$。这样，我们就可以用一个恰当的纠错码$E: \{0,1\}^N \to \{0,1\}^{N^C}$来编码$f'$，其中$C > 1$是一个常数。我们定义函数$g$在任意$x \in \{0,1\}^{C'}$上的输出为$E(f')$的第$x$个二进制位。[⊖]为了证明$g$确实满足定理的要求，我们只需证明纠错码满足下列性质：

391

1. 在任意$x \in \{0,1\}^N$上，$E(x)$可以在 poly(N)时间内被计算出来；

2. E存在运行时间为 polylog(N)的局部解码算法，它在运行过程中只随机访问函数 polylog(N)次，并且能够处理错误率 0.01。

上述两个性质可以用沃尔什-哈达玛纠错码与具有恰当参数的里德-穆勒纠错码的拼接来实现：

1. 令 RM 表示具有下列参数的里德-穆勒纠错码：

- 域$\mathbf{F}$的大小为$\log^2 N$。

⊖　如果必要的话用一些 0 进行填充，则我们可以假设函数g的输入的长度是C的倍数。

- 变量个数 ℓ 为 $\log N/\log\log N$。
- 多项式的次数 d 等于 $\log^2 N$。

RM 的输入的长度至少是 $\left(\frac{d}{\ell}\right)^{\ell}<N$（因此，经过填充后，我们可以认为 RM 的输入是 $\{0,1\}^n$），其输出的长度是 $|\mathbf{F}|^{\ell}\leqslant\mathrm{poly}(n)$，其距离至少是 $1-1/\log N$。

2. 令 WH 是从 $\{0,1\}^{\log|\mathbf{F}|}=\{0,1\}^{2\log\log N}$ 到 $\{0,1\}^{|\mathbf{F}|}=\{0,1\}^{\log^2 N}$ 的沃尔什-哈达玛纠错码。

我们最终的编码是 WH $\circ$ RM，将沃尔什-哈达玛纠错码的局部解码算法和里德-穆勒纠错码的局部解码算法组合在一起就可以得出定理结论。 ■

结合定理 19.21 和姚期智 XOR 引理（引理 19.2），我们得到下面的推论。

推论 19.22　设 $S:\mathbf{N}\to\mathbf{N}$ 是一个单调的时间可构造函数。则存在 $\varepsilon>0$ 使得：如果存在 $f\in\mathbf{E}-\mathbf{DTIME}(2^{O(n)})$ 使得 $\mathrm{H}_{wc}(f)(n)\geqslant S(n)$ 对任意 n 成立，则存在 $\hat{f}\in\mathbf{E}$ 使得 $\mathrm{H}_{wa}(\hat{f})(n)\geqslant S(\sqrt{n})^{\varepsilon}$ 对任意 n 成立。

证明　由定理 19.21 可知，在推论的假设条件下，存在函数 $g\in\mathbf{E}$ 使得 $\mathrm{H}^{1/2}_{wc}(g)(n)\geqslant S'(n)=S(n)/\mathrm{poly}(n)$ 成立。这里，我们假设 $S'(n)\geqslant\sqrt{S(n)}$ 对充分大的 n 成立（若不然，$S(n)$ 是多项式，此时推论是平凡的）。考虑函数 $g^{\oplus k}$，其中 $k=c\log S'(n)$ 并且 c 是一个充分小的常数。由姚期智 XOR 引理可知，在长度为 kn 的输入上，用规模为 $S'(n)$ 的线路不可能以大于 $1/2+2^{-\varepsilon' cn/1000}$ 的概率计算函数 $g^{\oplus k}$。由于 $S(n)\leqslant 2^n$，故 $kn<\sqrt{n}$，进而我们得到 $H_{wa}(g^{\oplus k})(n)\geqslant S(\sqrt{n})^{\varepsilon/1000}$。 ■

19.5　列表解码

推论 19.22 能将最坏复杂性下的难度转变为平均复杂性下的难度，这种能力让人非常惊讶。但是，这种能力还不能令人非常满意，因为它在实现这种转换的过程中使得线路的规模损失很大。具体地讲，如果转换前的函数 f 在最坏复杂性意义下在规模为 $2^{O(n)}$ 的线路上是难解的，则转换后的函数 f' 在平均复杂性意义下仅在规模为 $2^{O(\sqrt{n})}$ 的线路上是难解的。这种难度变换或许也能满足某些应用的需要，但是，就第 20 章将要进行的在最坏复杂性假设下去除 **BPP** 的随机性而言，这种难度变换还不够。

我们将用更强的归约来实现从最坏复杂性到平均复杂性的转换，这需要抛弃 XOR 引理，直接用纠错码将最坏复杂性下的难解函数转换成另一个函数使得它在平均复杂性下以略大于 1/2 的概率是难解的。但是，这个想法面临着基础性的困难：如果 f 在最坏复杂性 392
意义下是难解的，则用任意纠错码 E 似乎都很难说明 $E(f)$ 在平均复杂性意义下以概率 0.6 是难解的。这是由于，二进制字母表上的任意纠错码的编码距离至多是 1/2；但是，任意解码算法能够处理的错误率却至多是编码距离的一半；因此，如果 $E(f)$ 在超过 1/4 比例的位置上出现错误，则损坏的 $E(f)$ 与原始 $E(f)$ 至多在 0.75 比例的位置上是一致的，进而解码算法很难从损坏的 $E(f)$ 中恢复出。

上一自然段所述的情况似乎是一个切实的障碍。人们以前在纠错码的各种应用中曾经一直将它视为不可逾越的障碍，直到深刻领悟了下述事实的重要性："设 E 是一个编码距离足够好的纠错码。如果 $E(x)$ 在（不妨设）0.4 比例的位置出现了错误，将这个带错误的位串记为 y，那么，与 y 的距离不超过 0.4 的码字可能不止一个，但这样的码字也不会太多。"形式地讲，我们有如下的定理。

定理 19.23（约翰逊界(Johnson Bound)[Joh62]）　如果 $E:(0,1)^n\to(0,1)^m$ 是编码距离至少为 $1/2-\varepsilon$ 的一个纠错码，则对于任意 $x\in(0,1)^n$ 和 $\delta\geqslant\sqrt{\varepsilon}$ 而言，至多存在 $1/(2\delta^2)$ 个向量 $y_1,\cdots,y_\ell$ 使得 $\Delta(x,y_i)\leqslant 1/2-\delta$ 对任意 $i\in[\ell]$ 成立。

证明　假设 $x,y_1,\cdots,y_\ell$ 满足条件。我们在 $\mathbf{R}^m$ 中如下定义 ℓ 个向量 $z_1,\cdots,z_\ell$。对于任意 $i\in[\ell]$ 和 $k\in[m]$，如果 $y_{i,k}=x_k$，则令 $z_{i,k}$ 等于 $+1$；否则，令 $z_{i,k}$ 等于 -1。在我们的假设条件下，对任意 $i\in[\ell]$ 均有

$$\sum_{k=1}^{m} z_{i,k}\geqslant 2\delta m \tag{19.10}$$

这是因为 z_i 与 x 在比例为 $1/2+\delta$ 的坐标位上是一致的。同时，对任意 $i\neq j\in[\ell]$ 均有

$$\langle z_i,z_j\rangle=\sum_{k=1}^{m} z_{i,k}z_{j,k}\leqslant 2\varepsilon m\leqslant 2\delta^2 m \tag{19.11}$$

这是因为 E 的编码距离至少为 $1/2-\varepsilon$。

往证(19.10)式和(19.11)式联合在一起意味着 $\ell\leqslant 1/(2\delta^2)$。事实上，令 $w=\sum_{i=1}^{\ell} z_i$。一方面，由(19.11)式可得，

$$\langle w,w\rangle=\sum_{i=1}^{\ell}\langle z_i,z_i\rangle+\sum_{i\neq j}\langle z_i,z_j\rangle\leqslant \ell m+\ell^2 2\delta^2 m$$

另一方面，由(19.10)式可得，$\sum_k w_k=\sum_{i,j} z_{i,j}\geqslant 2\delta m\ell$，进而 $\langle w,w\rangle\geqslant|\sum_k w_k|^2/m\geqslant 4\delta^2 m\ell^2$。这是由于对任意 c，满足 $\sum_k w_k=c$ 的具有最小 2-范数的向量 $w\in\mathbf{R}^m$ 是均匀向量 $(c/m,c/m,\cdots,c/m)$。于是，$4\delta^2 m\ell^2\leqslant\ell m+\ell^2 2\delta^2 m$，这表明 $\ell\leqslant 1/(2\delta^2)$。 ■

19.5.1　里德-所罗门纠错码的列表解码

在许多场合下，从损坏的码字中恢复出消息的一个候选列表与将它唯一地解码一样有用。例如，我们有时还拥有一些外部信息，将这些信息与消息的候选列表关联起来，就有可能恢复出正确的消息。为了实现这种想法，我们需要一个算法来计算消息的候选列表。1996 年，苏丹(Sudan)在广泛使用并且十分重要的里德-所罗门纠错码上得到了这种算法。该算法可以在错误率高达 $1-2\sqrt{\frac{d}{m}}$ 的受损的长度为 m 的里德-所罗门码字上恢复出多项式个候选码字。注意，当 m/d 增大时，该算法处理的错误率逐渐趋向于 1，而 19.3 节讨论的波利坎普-韦尔奇算法(Berlekamp-Welch Algorithm)不能处理大于 1/2 距离的错误率(其他实现唯一解码的算法也如此)。

393

定理 19.24（里德-所罗门纠错码的列表解码[Sud96]）　存在一个多项式时间算法以 $\mathbf{F}^2$ 上的序对集合 $\{(a_i,b_i)\}_{i=1}^m$ 为输入并输出一系列 d 次多项式 G 使得满足 $G(a_i)=b_i$ 的 i 的个数大于 $2\sqrt{dm}$。

证明　下面的算法证明了定理 19.24 的正确性。

里德-所罗门列表解码算法： $t>2\sqrt{dm}$

1. 找出一个非零的二元多项式 $Q(x,y)$ 使得其中 x 的次数至多为 $\sqrt{dm}$，y 的次数至多为 $\sqrt{m/d}$，并且 $Q(b_i,a_i)=0$ 对任意 $i\in[m]$ 成立。

上述条件可以表示成 m 个线性方程构成的方程组，其中的未知数是 Q 的 $(\sqrt{dm}+1)(\sqrt{m/d}+1)>m$ 个系数。由于这些方程都是齐次方程(亦即，方程右端都是 0)并且未知数的个数大于方程的个数，因此，可以用高斯消去法求得方程组的一个非零解。

2. 利用高效的多项式分解算法(参见[VG99])对 $Q(x, y)$ 进行因式分解。对 $Q(x, y)$ 的每个形如 $(P(x)-y)$ 的因式，验证其次数是否不超过 d 并且是否存在 $\{(a_i, b_i)\}_{i=1}^{n}$ 中的至少 t 个序对使得它取值为 0。如果两个条件都满足，则输出 P。

事实上，如果 $\{(a_i, b_i)\}_{i=1}^{n}$ 中的至少 t 个序对使得 $G(x)$ 满足 $G(a_i)=b_i$，则 $G(x)-y$ 是 $Q(x, y)$ 的因式。为了看清这一点，请注意 $Q(G(x), x)$ 是一个次数不超过 $\sqrt{dm}+d\sqrt{m/d}=2\sqrt{dm}<t$ 的一元多项式。由于它至少在 t 个点上取值为 0，故它是零多项式。这就说明 $G(x)-y$ 整除 $Q(x, y)$(参见习题 19.16)。 ■

19.6　局部列表解码：接近 BPP＝P

同 19.4 节一样，要将列表解码切实地用于难度放大，我们需要为所选用的纠错码提供局部列表解码算法。幸运的是，沃尔什-哈达玛纠错码，里德-穆勒纠错码，以及二者的拼接纠错码都存在局部列表解码算法。下面，我们给出局部列表解码的定义，它稍微有点晦涩，需要读者仔细地研读。

定义 19.25　(局部列表解码算法)　设 $E: \{0, 1\}^n \to \{0, 1\}^m$ 是一个纠错码，$\rho=1-\varepsilon, \varepsilon>0$。称算法 D 是 E 的处理错误率 ρ 的局部列表解码算法，如果对于满足 $\Delta(E(x), y)\leqslant\rho$ 的任意 $x\in\{0, 1\}^n$ 和任意 $y\in\{0, 1\}^m$ 均存在整数 $i_0\in[\mathrm{poly}(n/\varepsilon)]$ 使得：对于任意 $j\in[m]$，算法 D 以 i_0，j 和对 y 的随机访问能力为输入，在 $\mathrm{poly}(\log(m)/\varepsilon)$ 时间内至少以概率 2/3 输出 x_j。

394

定义 19.25 中的 i_0 可以视为 x 在"列表解码算法 L 输出的 $\mathrm{poly}(n/\varepsilon)$ 个消息构成的候选列表"中的一个位置标号。同定义 19.16 一样，定义 19.25 也很容易推广到非二进制字母表的纠错码上。

19.6.1　沃尔什-哈达玛纠错码的局部列表解码

事实上，本书前面已经在戈德赖希-勒维定理(定理 9.12)的证明中介绍了沃尔什-哈达玛纠错码的局部列表解码算法。该定理的证明给出了一个算法，它通过访问一个"能以概率 $1/2+\varepsilon$ 计算函数 $y\mapsto x\odot y$(其中 $x, y\in\{0, 1\}^n$)"的黑盒，最终计算出一个列表 $x_1, \cdots, x_{\mathrm{poly}(n/\varepsilon)}$ 使得 $x_{i_0}=x$ 对某个 i_0 成立。第 9 章运用该算法来找出正确的 x 时需要查验列表中的每个串是否能得到 $f(x)$(其中 f 是一个单向函数)。这个例子很好地说明了如何运用外部信息来缩短列表解码算法得到的候选码字列表。

19.6.2　里德-穆勒纠错码的局部列表解码

本小节给出里德-穆勒纠错码的一个局部列表解码算法。注意，该纠错码的码字是 ℓ 个变量上的某个 d 次多项式 $P: \mathbf{F}^\ell\to\mathbf{F}$ 的函数值列表，局部解码的任务就是在给定的 $x\in\mathbf{F}^\ell$ 上计算 $P(x)$。

定理 19.26　(里德-穆勒局部列表解码算法[BF90, Lip91, BFNW93, STV99])　里德-穆勒纠错码存在处理错误率为 $1-10\sqrt{d/|\mathbf{F}|}$ 的局部列表解码算法。

也就是说，对任意的 $\mathbf{F}$，d，ℓ，存在时间复杂度为 $\mathrm{poly}(|\mathbf{F}|, d, \ell)$ 的如下算法 D。算法 D 的输入包括：(a)对函数 $f: \mathbf{F}\to\mathbf{F}$ 的随机访问能力；(b)编号 $i\in\mathbf{F}^{\ell+1}$；(c)一个输入点 $x\in\mathbf{F}$。算法 D 的输出 $D^f(i, x)$ 满足：如果 f 与某个 ℓ 元 d 次多项式 $P: \mathbf{F}\to\mathbf{F}$ 在 $10\sqrt{d/|\mathbf{F}|}$ 比例的输入上取相同的值，则存在 $i_0\in\mathbf{F}^{\ell+1}$ 使得 $\Pr[D^f(i_0, x)=P(x)]\geqslant 2/3$ 在任意 x 上成立。

证明　定理要求的局部列表解码算法在给定 i_0 后必须在任意 $x\in\mathbf{F}$ 上高概率地输出 $P(x)$。下面给出的算法是这种解码算法的宽松形式，它只能保证给定 i_0 后仅在绝大多数(亦即，0.9 比例的)x 上高概率地输出正确的 $P(x)$。将这个算法与 19.4.2 节给出的里德-穆勒局部解码算法组合在一起，就能得出定理要求的局部列表解码算法。这样，下面的算法证明了定理 19.26 的正确性。

里德-穆勒局部列表解码算法： $\rho\leqslant 1-10\sqrt{d/|\mathbf{F}|}$

输入：

- 对函数 $f: \mathbf{F}\to\mathbf{F}$ 的随机访问能力，其中 $\Pr_{x\in\mathbf{F}}[P(x)=f(x)]>10\sqrt{d/|\mathbf{F}|}$ 对某个 ℓ 元 d 次多项式 $P: \mathbf{F}\to\mathbf{F}$ 成立。我们假设 $|\mathbf{F}|>d^3$ 并且 d 足够大(比如，$d>1000$ 就可以了)。这些条件在我们的应用中总是成立的。
- 编号 $i_0\in\mathbf{F}^{\ell+1}$，我们将它视为序对 (x_0, y_0)，其中 $x_0\in\mathbf{F}$，$y_0\in\mathbf{F}$；
- 位串 $x\in\mathbf{F}$；

输出：$y\in\mathbf{F}$(在某个序对 (x_0, y_0) 上计算得到的 y 应该会使 $P(x)=y$ 至少以概率 0.9
395 成立，其中概率是相对于算法的随机选择和从 $\mathbf{F}$ 中随机选择 x 而言的)。

操作：

1. 令 L_{x, x_0} 是通过 x，x_0 的随机 3 次曲线。也就是说，我们随机选取一元 3 次多项式 $q: \mathbf{F}\to\mathbf{F}$ 使得 $q(0)=x$，$q(r)=x_0$ 对某个随机的 $r\in\mathbf{F}$ 成立，然后令 $L_{x, x_0}=\{q(t): t\in\mathbf{F}\}$，参见图 19-8。

2. 在 L_{x, x_0} 的 $|\mathbf{F}|$ 个点上获得函数 f 的值，得到由 $|\mathbf{F}|$ 个序对构成的集合 $S=\{t, f(q(t)): t\in\mathbf{F}\}$。

3. 运行苏丹(Sudan)给出的里德-所罗门列表解码算法，得出至少通过 S 中 $8\sqrt{d|\mathbf{F}|}$ 个点的次数为 $3d$ 的所有多项式 $g_1, g_2, \cdots, g_k$。

4. 如果存在唯一的 $i\in[k]$ 使得 $g_i(r)=y_0$，则输出 $g_i(0)$。否则，算法停止但不输出任何值。

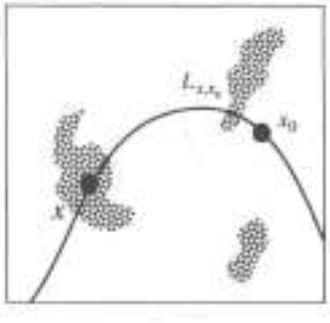


图 19-8　给定损坏的多项式 $P: \mathbf{F}\to\mathbf{F}$ 和一个编号 (x_0, y_0) 之后，为了计算 $P(x)$，我们在一条通过 x，x_0 的随机 3 次曲线 L_{x, x_0} 上利用里德-所罗门局部解码算法恢复出 P 在 L_{x, x_0} 上限制形式的一个候选列表。如果该列表中只有一个多项式在 x_0 上取值为 y_0，则利用该多项式来计算 $P(x)$

我们证明：对于任意函数 $f: \mathbf{F}\to\mathbf{F}$ 和 $x\in\mathbf{F}$，只要 f 与某个 ℓ 元 d 次多项式 P 在比例为 $10\sqrt{d/|\mathbf{F}|}$ 的输入点上取相同的值，如果 x_0 随机取自 $\mathbf{F}$ 而 $y_0=P(x_0)$，则里德-穆勒局部列表解码算法输出 $P(x)$ 的概率至少是 0.9(其中概率是相对于算法的随机选择和从 $\mathbf{F}$ 中随机选择 x 而言的)。根据标准的均值论证法，上述结论意味着：存在序对 (x_0, y_0) 使得算法以它为输入时将在 $\mathbf{F}$ 中比例为 0.9 的 x 上输出 $P(x)$。

考虑下面的虚拟算法，不难发现：它在任意 $x\in\mathbf{F}$ 上得到的输出与里德-穆勒局部列表解码算法以 x，随机的 $x_0\in\mathbf{F}$ 和 $y_0=P(x_0)$ 为输入时得到的输出是一样的。

1. 随机选择一条通过 x 的 3 次曲线 L。也就是说，$L=\{q(t):t\in\mathbf{F}\}$，其中 $q:\mathbf{F}\to\mathbf{F}$ 是满足 $q(0)=x$ 的随机一元 3 次多项式；

2. 得到 $\mathbf{F}$ 上所有的一元（$3d$ 次）多项式 g_1，…，g_m 使得“至少有 $6\sqrt{d|\mathbf{F}|}$ 个 t 使得 $g_i(t)=f(q(t))$”对任意的 $i\in[m]$ 成立。

3. 随机选择 $r\in\mathbf{F}$。假设 $y_0=P(q(r))$ 是给定的值。

4. 如果存在唯一的 $i\in[m]$ 使得 $g_i(r)=y_0$，则输出 $g_i(0)$；否则，算法终止但不输出任何值。

上面的虚拟算法输出 $P(x)$ 的概率至少是 0.9。事实上，在通过 x 的随机 3 次曲线上，除 x 之外的其他点是两两独立的[⊖]，因此切比雪夫不等式表明：函数 f 与多项式 P 在曲线上的 $8\sqrt{d|\mathbf{F}|}$ 个点上取相同值的概率至少为 0.99。（这一点很容易用点的两两独立性和 A.2.4 节的切比雪夫不等式加以证明。）于是，算法第 2 步得到的 g_1，…，g_m 中含有定义式为 $g(t)=P(q(t))$ 的多项式 $g:\mathbf{F}\to\mathbf{F}$。习题 19.15 要求读者证明 g_1，…，g_m 中至多有 $\sqrt{|\mathbf{F}|}/4d$ 个多项式。由于任意两个 $3d$ 次多项式至多在 $3d+1$ 个点上取相同的值，因此随机选择 $r\in\mathbf{F}$ 时 $g(r)\neq g_i(r)$ 对任意 $g_i\in\{g_1,\cdots,g_m\}(g_i\neq g)$ 成立的概率至少为 $1-\frac{(3d+1)\sqrt{|\mathbf{F}|}/4d}{|\mathbf{F}|}>0.99$。因而，算法找到 g 且输出 $g(0)=P(x)$ 的概率也至少是 0.99。 ■ 396

19.6.3　拼接纠错码的局部列表解码

如果 $E_1:\{0,1\}^n\to\Sigma^m$ 和 $E_2:\Sigma\to\{0,1\}^t$ 是两个纠错码，并且它们都存在局部列表解码算法，则二者的拼接纠错码 $E_2\circ E_1:\{0,1\}^n\to\{0,1\}^{mt}$ 也存在局部列表解码算法。同 19.4.3 节一样，$E_2\circ E_1$ 的局部列表解码算法调用 E_1 的局部列表解码算法，并且在 E_1 需要随机访问函数值时再调用 E_2 的局部列表解码算法。更具体地说，假设 E_1 的局部列表解码算法的输入编号取自集合 I_1 并且能够处理错误率 $1-\varepsilon_1$，而 E_2 的局部列表解码算法的输入编号取自集合 I_2 并且能够处理错误率 $1/2-\varepsilon_2$。则 $E_2\circ E_1$ 的局部列表解码算法分别取 $i_1\in I_1$ 和 $i_2\in I_2$ 作为输入编号，然后在 i_1 上调用 E_1 的局部列表解码算法，每当它需要随机访问函数值时就在 i_2 上调用 E_2 的局部列表解码算法来给出相应的函数值。（参见 19.4.3 节。）我们断言，$E_2\circ E_1$ 的局部列表解码算法能够处理的错误率是 $1/2-\varepsilon_1\varepsilon_2|I_2|$。事实上，如果 y 与某个码字 $E_2\circ E_1(x)$ 在比例为 $\varepsilon_1\varepsilon_2|I_2|$ 的坐标分量上取相同的值，则 y 中存在比例为 $\varepsilon_1|I_2|$ 的区组使得它们与码字 $E_1(x)$ 的相应区组至少在比例为 $1/2+\varepsilon_2$ 的坐标分量上取相同的值。于是，由均值论证法可知，存在编号 i_1 使得：给定 i_2 作为输入，调用 E_2 的局部列表解码算法所得的输出结果与码字 $E_1(x)$ 中比例为 ε_1 的符号是一致的。这又意味着存在 i_2 使得：给定 (i_1,i_2) 和坐标编号 j，$E_2\circ E_1$ 的局部列表解码算法将高概率地输出 x_j。

19.6.4　局部列表解码算法综合运用于难度放大

正如前面指出的那样，利用局部列表解码算法可以将仅在最坏复杂性意义下的难解函数变换为另一个函数使得不能以显著高于 1/2 的概率计算它。

⊖ 利用 A.5 节介绍的多项式的基本事实，推广定理 8.15 的证明（它由随机线性函数得到了两两独立的函数）过程就可以证明该结论；也可以参见习题 8.4。

定理 19.27（最坏难度转变为强平均难度）　如果 $f\in \mathbf{E}=\mathbf{DTIME}(2^{\Theta(n)})$ 使得 $\mathrm{H}_{wc}(f)(n)\geqslant S(n)$ 对某个时间可构造的非减的 $S:\mathbf{N}\to\mathbf{N}$ 成立，则存在函数 $g\in\mathbf{E}$ 和常数 $c>0$ 使得 $\mathrm{H}_{wc}(g)(n)\geqslant S(n/c)^{1/c}$ 对充分大的 n 成立。

证明概要　同 19.4.4 一样，对于任意的 n，我们将函数 f 在 $\{0,1\}^n$ 上的限制视为一个长度为 N 的位串，其中 $N=2^n$。然后，将它用里德-穆勒纠错码和沃尔什-哈达玛纠错码的拼接纠错码进行编码，得到一个位串 $g\in\{0,1\}^{N'}$，再把 $g\in\{0,1\}^{N'}$ 视为从 $\{0,1\}^{n'}$ 到 $\{0,1\}$ 的函数，其中 $n'=\lceil\log N'\rceil$。我们设置参数使得 $n'=O(n)$，最后证明存在某个 $\varepsilon>0$ 使得：纠错码的局部列表解码算法可以在 $S(n)^{\varepsilon}$ 时间内将“能够在 $\{0,1\}^{n'}$ 中比例为 $1/2+1/S(n)^{\varepsilon}$ 的输入上正确计算函数 g 的任意一个算法”变换为“在 $\{0,1\}^n$ 中所有输入上正确计算函数 f 的一个线路”。

为了完成定理的证明，我们需要说明上述过程可以通过恰当地选取里德-穆勒纠错码的参数来实现，这些参数包括域 $\mathbf{F}$，多项式的次数 d 和多项式中变量的个数 ℓ。我们取定 $|\mathbf{F}|=S(n)^{\delta}$，其中 $\delta>0$ 是某个较小的常数，$d=\sqrt{|\mathbf{F}|}$。对于 ℓ，我们选取足够大的值使得 f 确实能够表示成里德-穆勒纠错码的输入。由于里德-穆勒纠错码的输入是域中元素构
397 成的一个长度为 $\binom{d+\ell}{\ell}\geqslant(d/\ell)^{\ell}$ 的元组，读者可以验证取 $\ell=2\log N/\log d$ 足以满足要求。（不失一般性，我们可以假设 $d>\log^2 N$，这是因为当 $S(n)=n^{\Omega(1)}$ 时，定理是平凡的。类似地，我们还可以假设 $S(n)<N$。）里德-穆勒纠错码的输出是 $\mathbf{F}$ 中的 $|\mathbf{F}|^{\ell}$ 个元素，而每个元素又被沃尔什-哈达玛纠错码编码为长度为 $|\mathbf{F}|$ 的位串，因此拼接编码之后最终得到的位串的总长度是 $N'=|\mathbf{F}|^{\ell+1}$。根据我们对参数的取法可知，$N'=N^{o(1)}$，因而，我们确实有 $\log N'=O(\log N)$。沃尔什-哈达玛纠错码和里德-穆勒纠错码的编码时间都是 $\mathrm{poly}(|\mathbf{F}|,d,\ell)$。在我们的参数设定下，该时间复杂度是 $S(n)^{c\delta}$，其中 c 是一个（独立于 δ 的）绝对常数。我们能够用局部列表解码算法处理错误率 $1/2-S(n)^{-\delta/3}$，并且列表的长度等于 $|\mathbf{F}|^{O(\ell)}=N^{o(1)}$，进而列表元素的标号仅需 $O(\log N)$ 个二进制位来表示。于是，取足够小的 δ 使得它是 ε 的函数，并且在列表解码算法中固化 $O(\log N)$ 规模的标号，则可以将“在比例为 $1/2+1/S(n)^{\varepsilon}$ 的输入上正确计算 g 的任意规模为 S' 的线路”变换为“在所有输入上均能正确计算 f 的一个规模为 $S(n)^{\varepsilon}\cdot S'$ 的线路”。■

本章学习内容

- 姚期智 XOR 引理可用于将仅具有温和难度（不能以概率 0.99 被计算）的布尔函数变换为具有强难度（不能以概率 0.51 被计算）的布尔函数。XOR 引理的证明过程表明，每个具有温和难度的难解函数都存在一个极难计算的输入的“难度核”。
- 纠错码是一种映射，它能将任意两个不同的字符串映射为两个在很多分量上取值不同的字符串。具有局部解码算法的纠错码可以用来将最坏复杂性下的难解函数变换为平均复杂性下具有温和难度的函数。
- 二进制字母表上的纠错码的编码距离至多是 1/2。编码距离为 δ 的纠错码的唯一性解码算法能够处理错误率至多为 $\delta/2$。纠错码的列表解码算法能够处理的错误率几乎可以达到 δ，但是它的输出不是一个消息而是较短的候选消息列表。
- 纠错码的局部列表解码算法可以用来将一个仅在最坏复杂性下难解的函数变换为一个平均复杂性下强难解的函数。

本章注记和历史

姚期智 XOR 引理是他对论文[Yao82a]进行演讲报告时陈述和证明的。自那以后，人们发表了该引理的好几个证明，其中第一个公开发表的证明源自勒维(Levin)[Lev87](参见综述[GNW95])。拉塞尔·因帕利亚佐(Russell Impagliazzo)的难度核引理在论文[Imp95a]中被证明，19.1.2 节给出的证明源自诺姆·尼散(Noam Nisan)。

纠错码的研究是一个极其优美而有用的领域，而本章只是很肤浅地讨论了其中的一些主题。该研究领域的诞生源自两篇几乎同时发表的开创性论文，这两篇论文分别是香农(Shannon)的[Sha48]和汉明(Hamming)的[Ham50]。马杜·苏丹(Madhu Sudan)的讲义(在他的主页上可以找到)为理论计算机科学家初学纠错码提供了很好的材料，也可以参考综述[Sud01]。

里德-所罗门纠错码是欧文·里德(Irving Reed)和古斯塔夫·所罗门(Gustave Solomon)在 1960 年发明的[RS60]。里德-所罗门纠错码的第一个高效解码算法源自彼得森(Petersen)[Pet60]。(有意思的是，该算法的发明甚至早于 **P** 的形式化定义，它是最早发明的几个非平凡的多项式时间算法之一。)19.3 节介绍的算法是格默尔(Gemmell)和苏丹(Sudan)[GS92]对波利坎普-韦尔奇解码算法[BW86]的简化。 398

里德-穆勒纠错码是穆勒(Muller)发明的[Mul54]，它的第一个解码算法是里德(Reed)在[Ree54]中给出的。里德-穆勒纠错码的第一个局部解码算法源自毕威尔(Beaver)和费根鲍姆(Feigenbaum)[BF90]，而利普顿[Lip91]发现了该局部解码算法意味着积和式(参见 8.6.2 节)的最坏复杂性与平均复杂性之间存在联系。巴拜(Babai)、福特劳(Fortnow)和伦德(Lund)[BFL90]观察到，在多线性扩张下这种联系也存在于 **PSPACE** 和 **EXP** 之间。巴拜等人[BFNW93]证明了这种联系可用于在最坏复杂性假设下的去随机化(derandomization)。19.4.2 节介绍的里德-穆勒局部解码算法属于格默尔(Gemmell)等人[GLR+91]。

里德-所罗门纠错码的第一个局部列表解码算法是苏丹(Sudan)给出的[Sud96]，后来该算法又由古鲁斯瓦米(Guruswami)和苏丹进行了改进[GS88]。最近，帕尔瓦雷什(Parvaresh)和瓦尔迪(Vardy)[PV05]为里德-所罗门纠错码的一种变形给出了一个局部列表解码算法，它能处理更高的错误率。该算法被古鲁斯瓦米和鲁德拉(Rudra)进行了改进[GR06]，改进后的结果在更大的字母表上实现了错误率和列表解码半径之间的最佳平衡。

在规模相当的线路上实现强难度放大(定理 19.27)率先由因帕利亚佐(Impagliazzo)和维格德尔森(Wigderson)证得[IW97]，他们在这篇论文中还给出了姚期智 XOR 引理的去随机化形式。我们给出的证明是苏丹(Sudan)、特雷维山(Trevisan)和瓦德翰(Vadhan)[STV99]给出的另一种证明，正是他们率先建立了纠错码和难度放大之间的明确联系，也正是他们率先定义了局部列表解码算法并将它运用于难度放大。沃尔什-哈达玛纠错码的第一个局部列表解码算法是戈德赖希(Goldreich)和勒维(Levin)[GL89]率先提出的(尽管该算法当时并未使用现在的正式术语来描述)。19.6 节介绍的里德-穆勒纠错码的局部列表解码算法是[STV99]所给算法的一种变形。

奥唐纳(O'Donnell)[O'D04]研究了习题 19.8 提出的问题，并给出了 **NP** 的一个难度放大引理。要了解该问题更精准的难度放大结果，请参阅希利(Healy)、瓦德翰(Vadhan)和瓦尔拉(Viola)的论文[HVV04]。

习题

19.1　设 X_1，…，X_k 是独立的随机变量，其中 $X_i=1$ 的概率为 $1-\delta$ 而 $X_i=0$ 的概率为 δ。令 $X=\sum_{i=1}^{k} X_i \pmod 2$。证明：$\Pr[X=1]=1/2+(1-2\delta)^k/2$。

19.2　证明：如果存在密度为 δ 的分布 H 使得 $\Pr_{x\in_R H}[C(x)=f(x)]\leqslant 1/2+\varepsilon$ 对规模至多为 $S\leqslant\sqrt{\varepsilon^2\delta 2^n/100}$ 的任意线路 C 均成立，则存在规模至少为 $\frac{\delta}{2}2^n$ 的子集 $I\subseteq\{0,1\}^n$ 使得 $\Pr_{x\in_R I}[C(x)=f(x)]\leqslant 1/2+2\varepsilon$ 对规模至多为 S 的任意线路 C 成立。

399

19.3　设 H 是 $\{0,1\}^n$ 上密度为 δ 的分布，也就是说，$\Pr[H=x]\leqslant 1/(\delta 2^n)$ 对任意 $x\in\{0,1\}^n$ 成立。

(a) 令 G 是如下定义的分布：$\Pr[G=x]=(2^{-n}-\delta\Pr[H=x])/(1-\delta)$ 对任意 $x\in\{0,1\}^n$ 成立。证明：G 确实是一个分布（亦即，所有元素的概率都是非负值，并且所有元素的概率之和等于 1）。

(b) 令 U 是如下的分布：以概率 δ 选取服从分布 H 的元素，以概率 $(1-\delta)$ 选取服从分布 G 的元素。证明：U 是 $\{0,1\}^n$ 上的均匀分布。

19.4　证明姚期智 XOR 引理（定理 9.2）在一般 k 上也成立。

19.5　证明如下形式的超平面分离定理：如果 C，$D\subseteq\mathbf{R}^n$ 是两个不相交的凸集，其中 C 是闭集而 D 是紧集（亦即，有界的闭集），则存在非零向量 $z\in\mathbf{R}^n$ 和实数 $a\in\mathbf{R}$ 使得

$$x\in C\Rightarrow\langle x,z\rangle\geqslant a$$

$$x\in D\Rightarrow\langle x,z\rangle\leqslant a$$

19.6　利用习题 19.5 给出的超平面分离定理，证明最小-最大定理（参见注记 19.4）。

19.7　（[CG85]）称 $\{0,1\}^n$ 上的分布 D 是 K-扁平的，如果它是 $\{0,1\}^n$ 中某个大小至少为 K 的子集上的均匀分布。证明：对任意 k，密度为 2^{-k} 的任意分布 H 均是某些 2^{n-k}-扁平分布的一个凸组合。也就是说，存在 N 个 2^{n-k}-扁平分布 D_1，…，D_N 和满足 $\sum\alpha_i=1$ 的非负实数 α_1，…，α_N 使得 H 等价于如下的分布：以概率 α_i 选取 i，然后从分布 D_i 中抽取一个随机元素。

19.8　假设已知 **NP** 包含一个函数，它对所有多项式规模线路而言均是弱难解的。利用 XOR 引理，你能推断出 **NP** 中存在强难解的函数吗？为什么能或者为什么不能？

19.9　对于任意 $\delta<1/2$ 和充分大的 n，证明：存在函数 $E:\{0,1\}^n\to\{0,1\}^{n/(1-H(\delta))}$ 是距离为 δ 的纠错码，其中 $H(\delta)=\delta\log(1/\delta)+(1-\delta)\log(1/(1-\delta))$。

19.10　证明：距离为 1/2 的任意纠错码 $E:\{0,1\}^n\to\{0,1\}^m$ 必然满足 $2^n<10m$。证明：如果 $\delta>1/2$ 且 E 是一个距离为 δ 的纠错码，则 $2^n<10/(\delta-1/2)$。

19.11　设 $E:\{0,1\}^n\to\{0,1\}^m$ 是纠错码并且存在两个不同的位串 x^1，$x^2\in\{0,1\}^n$ 使得 $\Delta(E(x^1),E(x^2))\leqslant\delta$。证明：$E$ 不存在能够处理 $\delta/2$ 错误或更多错误的解码算法。也就是说，不存在函数 $D:\{0,1\}^m\to\{0,1\}^n$ 使得它能够在任意 $x\in\{0,1\}^n$ 和满足 $\Delta(y,E(x))\leqslant\delta/2$ 的 y 上得到 $D(y)=x$。

19.12　设 $E:\{0,1\}^n\to\{0,1\}^m$ 是编码距离为 δ 的一个纠错码。用显而易见的方法将 E 变形为另一个纠错码 $E':\{0,1,2,3\}^{n/2}\to\{0,1,2,3\}^{m/2}$。证明：$E'$ 的编码距离也是 δ。证明：上述过程的逆过程不成立。也就是说，请给出一个编码距离为 δ

的纠错码 E'：$\{0,1,2,3\}^{n/2}\to\{0,1,2,3\}^{n/2}$ 使得它在二进制字母表上对应的纠错码的编码距离等于 2δ。

400

19.13　完成定理 19.15 的证明。亦即，说明在输入序列 $(a_1,b_1),\cdots,(a_m,b_m)$ 中存在 $t(t\geqslant\frac{m}{2}+\frac{d}{2}+1)$ 个序对与 d 次多项式 G 相符的情况下如何恢复出 G。

19.14　证明定理 19.17。

19.15　设 $f:\mathbf{F}\to\mathbf{F}$ 是任意函数。假设整数 $d\geqslant 0$ 和数 ε 满足 $\varepsilon>2\sqrt{\frac{d}{|\mathbf{F}|}}$。证明：至多存在 $2/\varepsilon$ 个 d 次多项式在至少 ε 比例的点上与 f 取相同的值。

19.16　证明：如果 $Q(x,y)$ 是某个域 $\mathbf{F}$ 上的二元多项式，$P(x)$ 是 $\mathbf{F}$ 上使得 $Q(P(x),x)=0$ 的一元多项式，则 $Q(x,y)=(y-P(x))A(x,y)$ 对某个多项式 $A(x,y)$ 成立。

19.17　(线性纠错码)我们称纠错码 $E:\{0,1\}^n\to\{0,1\}^m$ 是线性的，如果任意 $x,x'\in\{0,1\}^n$ 均满足 $E(x+x')=E(x)+E(x')$，其中 $+$ 是按位相加模 2。线性纠错码 E 可以表示为一个 $m\times n$ 的矩阵 A，使得 $E(x)=Ax$ 对任意 $x\in\{0,1\}^n$ 成立，其中 x 视为列向量。

(a) 证明：线性纠错码 E 的编码距离等于非零 x 取遍 $\{0,1\}^n$ 时 $E(x)$ 中所有 1 的个数占总长度的比例的最小值。

(b) 证明：对于任意 $\delta>0$，存在编码距离为 δ 的线性纠错码 $E:\{0,1\}^n\to\{0,1\}^{\lfloor 1.1n(1-H(\delta))\rfloor}$，其中 $H(\delta)=\delta\log(1/\delta)+(1-\delta)\log(1/(1-\delta))$。

(c) 证明：对某个 $\delta>0$，存在编码距离为 δ 的线性纠错码 $E:\{0,1\}^n\to\{0,1\}^{poly(n)}$ 使得它有多项式时间的编码算法和解码算法。(要解决本问题，需要了解域 $GF(2^s)$，参见附录 A。)

(d) 我们称线性纠错码 $E:\{0,1\}^n\to\{0,1\}^m$ 是 ε-偏差的，如果在任意非零 $x\in\{0,1\}^n$ 上，$E(x)$ 中所有 1 的个数占总长度的比例都介于 $1/2-\varepsilon$ 和 $1/2+\varepsilon$ 之间。证明：对任意 $\varepsilon>0$，存在有多项式时间编码算法的 ε-偏差线性纠错码 $E:\{0,1\}^n\to\{0,1\}^{poly(n/\varepsilon)}$。

19.18　回顾一下，对于任意 m，存在含有 2^m 个元素的有限域 $\mathbf{F}=GF(2)^m$ 使得 $\mathbf{F}$ 的每个元素都可以表示为 $GF(2)$ 上的 m 维向量，并且 $\mathbf{F}$ 的加法就对应于向量的按位 XOR 操作(参见附录 A)。因此，对于任意 $a\in\mathbf{F}$，操作 $x\mapsto a\times x$(其中 $\times$ 是 $\mathbf{F}$ 的乘法)是 $GF(2)^m$ 上的线性操作。并且，给定 a 的具体表示形式之后，该线性操作可高效地计算。

(a) 证明：对于任意非零元素 $x\in\mathbf{F}$，如果在 $\mathbf{F}$ 中均匀随机地选取 a，则 $a\times x$ 将均匀分布于 $\mathbf{F}$ 上。

(b) 证明：对于任意非零元素 $x\in\mathbf{F}$，当随机选取 $a\in_R\mathbf{F}$ 时，$a\times x$ 的 m 维向量表示至多含有 $m/10$ 个 1 的概率小于 $2^{-m/10}$。由此得出结论：存在 $a\in\mathbf{F}$ 使得将任意 $x\in\{0,1\}^{m/10}$ 映射为 $a\times(x\circ 0^{0.9m})$ 的函数是一个编码距离至少为 0.1 的纠错码，其中 $\circ$ 表示拼接。

(c) 证明：存在常数 c，$\delta>0$ 使得在任意 n 上都存在编码距离至少为 δ 的显式纠错码 $E:\{0,1\}^n\to\{0,1\}^m$。

401

第20章

Computational Complexity: A Modern Approach

去随机化

上帝不跟宇宙玩掷骰子。

——艾尔伯特·爱因斯坦(Albert Einstein)

任何人，如果他认为算术方法可以产生随机数字，那他是在犯罪。

——约翰·冯·诺依曼(John von Neumann)，摘自克鲁斯(Knuth)的引文，1981

随机方法是计算机科学中的一种令人振奋和功能强大的典型方法，正如我们在第7章所见，随机方法通常可以用来为许多计算问题设计出最简单、最高效的算法。事实上，在计算机科学的很多领域(如，分布式算法和密码学)中，随机方法对完成某些任务或者高效地完成某些计算任务而言是必需的。因此，人们自然地得到(许多计算机科学家最初都这样认为)：至少对某些问题而言，随机方法在本质上是必需的——也就是说，不可能将随机算法转换为确定型算法而不引起计算效率的显著降低。这种猜想的一种具体的形式是**BPP**⊈**P**(**BPP**的定义参见第7章)。但出人意料的是，最近的研究提供了越来越多的证据表明该猜想很可能是错误的。正如在本章中即将看到的那样，在非常合理的复杂性假设下，确实有可能存在一种方法可以消除**BPP**类型的任意概率算法中的随机性(亦即，将概率算法转换成确定算法)而只引起算法的计算效率下降一个多项式因子。因此，目前大多数研究者都开始相信**BPP**=**P**。注意，这并不意味着随机方法在任何场合都不再有用。事实上，我们在第8章中已经看到：随机方法在交互式证明的定义中发挥着关键的作用。

20.1节定义伪随机数产生器，它是我们消除概率算法随机性的主要工具。本章定义的伪随机数产生器是第9章定义的安全伪随机数产生器的宽松形式。与第9章的情形相比，本章的伪随机数产生器在构造过程中将使用更好的参数和更弱的假设条件。20.2节将在"存在线路平均复杂性较高的显式函数"这一假设条件下来构造一个具体的伪随机数产生器。第19章已经说明了如何由线路最坏复杂性较高的函数来构造线路平均复杂性较高的函数。

我们在讨论过程中所假设的线路下界都是人们普遍承认的线路下界，尽管要证明这些线路下界似乎仍然遥不可及。人们自然地要问：这种假设条件对去随机化而言是不是必需的。20.3节将证明，如果只假设**BPP**≠**EXP**，则我们至少仍有可能得到部分去随机化的相关结论。而20.4节将证明，完全消除**BPP**的随机性需要我们证明线路下界。

虽然我们还不能证明出足够强的线路下界，但正如我们在密码学中的做法一样，我们在处理去随机化时也可以使用猜想的难解问题来代替实际可证得的难解问题，最终得到一个双赢的局面。也就是说，如果猜想的难解问题确实是一个难解的，则去随机化将得以实现；如果去随机化行不通，则会为猜想的难解问题设计出高效的算法。

例20.1（多项式恒等测试） 我们用一个例子来解释去随机化的概念。7.2.3节给出了一个随机算法来测试给定的多项式(表示为算术线路的形式)是否恒等于零多项式，我们考虑如何消除这个算法中的随机性。如果总次数为d的n元非零多项式P定义于充分大的域$\mathbf{F}$(如，$|\mathbf{F}|>10d$)，则多数向量$\boldsymbol{u}\in\mathbf{F}^n$都满足$P(\boldsymbol{u})\neq 0$(参见引理7.5)。因此，为了验

证$P\equiv 0$，只需简单地随机选取$\boldsymbol{u}\in_R\mathbf{F}^n$然后在$\boldsymbol{u}$上计算$P$的值。事实上，不难证明，存在$m^2$个向量$\boldsymbol{u}^1,\cdots,\boldsymbol{u}^{m^2}$使得：对每个可以用规模为$m$的算术线路来计算的非零多项式$P$，都存在$i\in[m^2]$满足$P(\boldsymbol{u}^i)\neq 0$。

这就给出了设计确定型算法的一种比较自然的想法：给出一个确定型算法，它以$m\in\mathbf{N}$为输入，在$\mathrm{poly}(m)$时间内输出满足上述性质的m^2个向量$\boldsymbol{u}^1,\cdots,\boldsymbol{u}^{m^2}$。实现上述想法应该不是很难，毕竟绝大多数的向量组都满足该性质，找出其中一组向量能有多难？但出人意料的是，实现该想法事实上非常困难。也就是说，如果不使用任何复杂性假设，我们不知道如何才能找出满足上述性质的一组向量。20.4 节将证明，找到这样一组向量(或为多项式恒等测试设计出其他确定型算法)实际上将会得出某些非平凡的线路下界。 ◀

20.1　伪随机数产生器和去随机化

我们用于研究去随机化的主要工具是伪随机数产生器，它是第 9 章定义的安全伪随机数产生器的变形，主要的区别在于：本章的伪随机数产生器的时间复杂度可以是指数时间。也就是说，伪随机数产生器的运行时间可以比"区分真随机数和伪随机数的算法(简称区分器(Distinguisher))"的运行时间更长。另一个区别是，本章将使用线路作为区分器，而不再像第 9 章一样使用图灵机作为区分器。也就是说，本章使用的区分器是非一致的。第二个区别不是本质的，正如第 9 章的注记中指出的那样，安全伪随机数产生器也可以使用线路作为区分器。

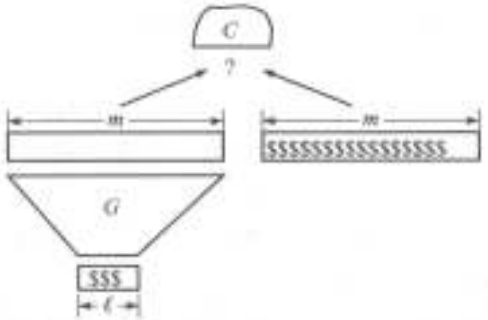
403

定义 20.2（伪随机数产生器）　称$\{0,1\}^n$上的分布R为一个(S,ε)-伪随机源(其中$S\in\mathbf{N}$，$\varepsilon>0$)，如果规模至多为S的任意线路C均使得

$$|\Pr[C(R)=1]-\Pr[C(U_n)=1]|<\varepsilon$$

其中U_n是$\{0,1\}^n$上的均匀分布。

令$S:\mathbf{N}\to\mathbf{N}$是一个函数。$2^n$时间内可计算的函数$G:\{0,1\}^*\to\{0,1\}^*$称为一个$S(\ell)$-伪随机数产生器，如果$|G(z)|=|S(z)|$对任意$z\in\{0,1\}^*$成立，并且在任意$\ell\in\mathbf{N}$上$G(U_\ell)$是一个$(S(\ell)^3,1/10)$-伪随机源。



在$S(\ell)$-伪随机数产生器的定义中，常数 3 和 1/10 是任意的，为方便计可以任意调换。为避免出现麻烦，在使用$S(\ell)$-伪随机数产生器时，我们要求$S:\mathbf{N}\to\mathbf{N}$是时间可构造的，并且是非递减单调的(亦即，$S(\ell')\geqslant S(\ell)$对任意$\ell'\geqslant\ell$成立)。

20.1.1　用伪随机数产生器实现去随机化

伪随机数产生器与模拟概率算法的运行之间的联系其实非常直接。

引理 20.3　假设存在$S(\ell)$-伪随机数产生器，其中$S:\mathbf{N}\to\mathbf{N}$是一个时间可构造的非

递减函数。那么，对于任意的多项式时间可计算的函数 ℓ：$\mathbf{N}\to\mathbf{N}$，$\mathbf{BPTIME}(S(\ell(n)))\subseteq \mathbf{DTIME}(2^{c\ell(n)})$均对某个常数 c 成立。

为了获得证明引理 20.3 的一些启发性认识，在证明引理 20.3 之前，我们先看看在不同的 S 上引理 20.3 在去随机化方面到底蕴含着什么样的结论。这些结论可以总结为下面简单的推论，其证明留作习题 20.1。

推论 20.4

1. 如果存在 2^{n^ε}-伪随机数产生器(其中 $\varepsilon>0$ 是某个常数)，则 $\mathbf{BPP}=\mathbf{P}$。

2. 如果存在 2^{ℓ}-伪随机数产生器(其中 $\varepsilon>0$ 是某个常数)，则 $\mathbf{BPP}\subseteq \mathbf{QuasiP}=\mathbf{DTIME}(2^{\mathrm{polylog}(n)})$。

404 3. 如果对任意 $c>1$ 都存在 ℓ-伪随机数产生器，则 $\mathbf{BPP}\subseteq\mathbf{SUBEXP}=\bigcap_{c>0}\mathbf{DTIME}(2^{n^c})$。

引理 20.3 的证明　语言 $L\in\mathbf{BPTIME}(S(\ell(n)))$，则根据定义 7.2 可知，存在输入 $x\in\{0,1\}^*$ 上的运行时间为 $cS(\ell(n))$(其中 c 是常数)的概率算法 A 满足

$$\Pr_{r\in_R\{0,1\}^m}[A(x,r)=L(x)]\geqslant 2/3 \tag{20.1}$$

其中 $m\leqslant S(\ell(n))$，并且当 $x\in L$ 时 $L(x)=1$，否则 $L(x)=0$。

证明引理的主要思想如下。如果我们将算法 A 使用的真随机数 r 替换为伪随机数产生器 G 产生的伪随机数 $G(z)$，其中 $z\in\{0,1\}^{\ell(n)}$ 是随机选取的，则由于算法 A 的运行时间仅为 $cS(\ell(n))$，因此大部分时间它将无法区分真随机数和伪随机数，进而(20.1)式中的概率在采用伪随机数之后不会下降到低于 $2/3-0.1>0.5$ 的程度。因此，为了消除 A 中的随机性，我们无需枚举所有 $r\in\{0,1\}^m$，而只需在所有 $z\in\{0,1\}^{\ell(n)}$ 上枚举伪随机数 $G(z)$，并查验是不是多数的伪随机数使得 A 进入接受状态。这样，去随机化后的算法的运行时间是 $2^{O(\ell(n))}$，而不是平凡的 2^m。

现在，我们将上述思想形式化。在输入 $x\in\{0,1\}^n$ 上，我们的确定型算法 B 穷举所有 $z\in\{0,1\}^{\ell(n)}$，并计算 $A(x,G(z))$，最后输出占半数以上的那个答案[⊖]。我们断言，当 n 充分大时，在 z 的所有选择中使得 $A(x,G(z))=L(x)$ 成立的 z 所占的比例至少为 $2/3-0.1$。由于我们可以将所有可能的 $2^{\ell(n)}$ 个 $G(z)$ 固化在算法中，因此该断言就证明了 $L\in\mathbf{DTIME}(2^{\ell(n)})$。

假设断言是错误的，则存在无穷个 x 使得 $\Pr[A(x,G(z))=L(x)]<2/3-0.1$。这样，我们将可以构造一个区分器来区分伪随机数产生器：只需利用库克-勒维变换(如同定理 6.6 的证明过程一样)构造一个线路来计算函数 $r\mapsto A(x,r)$，其中 x 是固化在线路中的。(这个"固化"过程是本证明中用到非一致性的地方。)该线路的规模为 $O(S(\ell(n)))^2$，当 n 充分大之后，$O(S(\ell(n)))^2\leqslant S(\ell(n))^3$。■

引理 20.3 解释了为什么定义 20.2 要求"伪随机数产生器的运行时间是随机种子长度的指数"就行了。其原因在于，消除随机化之后的算法要枚举长度为 ℓ 的所有随机种子，因此算法的时间复杂度至少是(ℓ 的)指数时间，即使伪随机数产生器本身的时间复杂度低于 $\exp(\ell)$ 时也是如此。还需注意到，允许伪随机数的时间复杂度为 $\exp(\ell)$ 还意味着：伪随机数产生器必须"骗过"时间复杂度低于 $\exp(\ell)$ 的区分器。相比之下，密码学中定义的安全伪随机数产生器(参见第 9 章定义 9.8)要求产生器的时间复杂度是某个固定的多项式时间，

⊖ 如果 $m<S(\ell(n))$，则 $A(x,G(z))$ 表示算法 A 在 x 上以 $G(z)$ 的前 m 个位作为随机数时得到的输出。

还需要"骗过"时间复杂度为多项式的任意区分器。这两种定义的区别源自定义这两种伪随机数产生器的目的互不相同。在密码学中，伪随机数产生器由诚实的用户使用，而区分器则是对密码系统展开攻击的敌手，因此，假设攻击者比系统的正常用户投入了更多的计算资源是非常合理的。在消除随机性时，伪随机数产生器是由去随机化之后的算法来使用，而区分器是正在被消除随机性的概率算法。此时，合理的假设是：去随机化之后的算法会比原来的概率算法运行更长时间。当然，同安全伪随机数相比，允许伪随机数产生器的时间复杂度为指数会让我们证明这种产生器的存在性变得更容易一些，而且情况也确实如 405
此。如果我们进一步放宽条件，在定义中不再对伪随机数产生器的效率提任何要求，那么这种伪随机数产生器的存在性将是平凡的(习题 20.2)。但是，对去随机化而言这种伪随机数产生器将毫无用处。

我们将借助复杂性假设来构造伪随机数产生器。较强的复杂性假设将等量地得到较强的伪随机数产生器(亦即，$S(\ell)$-伪随机数产生器中的 S 更大)。最强的假设(但仍然是合理的假设)将得到 $2^{O(\ell)}$-伪随机数产生器，它将蕴含 **BPP**=**P**。

20.1.2　难度与去随机化

我们在"存在显式的难解函数"这一假设条件下来构造伪随机数产生器。本章，我们利用平均难度的相关假设。根据第 19 章得到的结果，我们也可以仅依赖于最坏难度假设来构造伪随机数产生器。平均难度和最坏难度指的都是计算函数所需的布尔线路的最小规模。回顾一下，函数 $f:\{0,1\}^*\to\{0,1\}$ 的平均难度 $\mathrm{H}_{\mathrm{avg}}(f)$ 指的是使得"规模不超过 S 的任意线路 C 都满足 $\Pr_{x\in_R\{0,1\}^n}[C(x)=f(x)]<1/2+1/S$"成立的最大 S(参见定义 19.1)。对于函数 $f:\{0,1\}^*\to\{0,1\}$，我们用 $\mathrm{H}_{\mathrm{avg}}(f)(n)$ 表示"f 在 $\{0,1\}^n$ 上的限制"的平均难度。

例 20.5　下面给出一些函数和它们的难度，其中有些难度已被证明，有些难度仍是人们的猜测。

1. 如果 $f:\{0,1\}^*\to\{0,1\}$ 是随机函数(亦即，对于任意 $x\in\{0,1\}^*$，通过随机独立地投掷一枚无偏硬币来产生函数值 $f(x)$)，f 的平均难度和最坏难度以很高的概率都是指数(参见习题 20.3)。特别地，随着 n 的增大，$\mathrm{H}_{\mathrm{wrs}}(f)(n)$ 和 $\mathrm{H}_{\mathrm{avg}}(f)(n)$ 超过 $2^{0.99n}$ 的概率都趋向于 1。

2. 如果 $f\in$ **BPP**，则由于 **BPP**$\subseteq\mathbf{P}_{/\mathrm{poly}}$，故 $\mathrm{H}_{\mathrm{wrs}}(f)$ 和 $\mathrm{H}_{\mathrm{avg}}(f)$ 都是多项式有界的。

3. 一个合理的猜测是：3SAT 的最坏难度是指数的，亦即，$\mathrm{H}_{\mathrm{wrs}}(3\mathrm{SAT})(n)\geqslant 2^{\Omega(n)}$。一个相对较弱的假设是 **NP**$\not\subseteq\mathbf{P}_{/\mathrm{poly}}$。在这种假设下，$\mathrm{H}_{\mathrm{wrs}}(3\mathrm{SAT})(n)$ 不以任何多项式为上界。目前，人们还不清楚输入服从均匀分布时 3SAT 的平均复杂度是多少。但是，无论如何，3SAT 的平均复杂度都将依赖于布尔公式表示为位串的具体方法。

4. 在人们普遍接受的密码学假设下，**NP** 会含有平均复杂性下的难解函数。如果 g 是一个单向置换(其定义参见第 9 章)，它的逆函数不能被任意具有多项式规模的线路以多项式概率求得，则由定理 9.12 可知，将 $x,r\in\{0,1\}^n$ 映射为 $g^{-1}(x)\odot r$ 的函数 f 将具有超多项式的平均难度，亦即，$\mathrm{H}_{\mathrm{avg}}(f)\geqslant n^{\omega(1)}$。◀

本节的主要定理是利用平均难度的函数来构造伪随机数产生器。 406

定理 20.6　(由平均难度构造伪随机数产生器)　设 $S:\mathbf{N}\to\mathbf{N}$ 是时间可构造的非递减

函数。如果存在函数 $f\in \mathbf{DTIME}(2^{O(n)})$ 使得 $H_{avg}(f)(n)\geqslant S(n)$ 对任意 n 成立，则存在 $S(\delta\ell)^{\delta}$-伪随机数产生器，其中 $\delta>0$ 是某个常数。

结合定理 20.6 和定理 19.27，我们将得到如下定理。它提供了更强的证据来表明：只要具体地给出了定理 20.6 所要求的难解函数，则"消除概率算法的随机性是有可能实现的"这一猜想就是正确的。

定理 20.7（在最坏难度假设下消除随机性）　设 S：$\mathbf{N}\to\mathbf{N}$ 是时间可构造的非递减函数。如果存在函数 $f\in \mathbf{DTIME}(2^{O(n)})$ 使得 $H_{wrs}(f)(n)\geqslant S(n)$ 对任意 n 成立，则存在 $S(\delta\ell)^{\delta}$-伪随机数产生器，其中 $\delta>0$ 是某个常数。特别地，下列推论成立：

1. 如果存在 $f\in \mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 和 $\varepsilon>0$ 使得 $H_{wrs}(f)\geqslant 2^{\varepsilon n}$ 成立，则 $\mathbf{BPP}=\mathbf{P}$。
2. 如果存在 $f\in \mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 和 $\varepsilon>0$ 使得 $H_{wrs}(f)\geqslant 2^{\varepsilon^{\ell}}$ 成立，则 $\mathbf{BPP}\subseteq\mathbf{QuasiP}$。
3. 如果存在 $f\in \mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 使得 $H_{wrs}(f)\geqslant n^{\omega(1)}$ 成立，则 $\mathbf{BPP}\subseteq\mathbf{SUBEXP}$。

在定理 20.7 的推论 2 和推论 3 中，我们可以将 $\mathbf{E}$ 替换为 $\mathbf{EXP}=\mathbf{DTIME}(2^{\mathrm{poly}(n)})$。事实上，对于任意 $f\in \mathbf{DTIME}(2^{n^{\delta}})$，如果令函数 g 将输入 $x\in\{0,1\}^*$ 映射到"f 应用到 x 的前 $|x|^{1/\varepsilon}$ 个位上得到的输出"，则 g 属于 $\mathbf{DTIME}(2^{n})$ 并且满足 $H_{avg}(g)(n)\geqslant H_{avg}(f)(n^{1/\varepsilon})$。因此，如果存在 $f\in \mathbf{EXP}$ 满足 $H_{avg}(f)(n)\geqslant 2^{n^{\delta}}$，则存在常数 $\delta'>0$ 和函数 $g\in \mathbf{E}$ 满足 $H_{avg}(g)(n)\geqslant 2^{n^{\delta'}}$。于是，推论 2 中的 $\mathbf{E}$ 可以替换为 $\mathbf{EXP}$。类似的观察结果在推论 3 上也成立。注意，$\mathbf{EXP}$ 包含了很多复杂性类，人们相信其中很多复杂性类（如 $\mathbf{NP}$，$\mathbf{PSPACE}$，$\oplus\mathbf{P}$ 等等）都含有难解问题。

评注 20.8　尼散(Nisan)和维格德尔森(Wigderson)[NW88]率先在平均难度假设下构造出了伪随机数产生器，但他们并未证明上面所述的定理 20.6。证明定理 20.6 的另有其人。乌曼斯(Umans)经过一系列的工作[BFNW93，IW97，ISW99，STV99，Su01]，最终在[Uma03]中证明了该定理。尼散和维格德尔森仅证明了在定理的假设条件下存在 $S'(\ell)$-伪随机数产生器，其中 $S'(\ell)=S(n)^{\delta}$ 对某个常数 $\delta>0$ 成立并且 n 满足 $n>\delta\sqrt{\ell\log S(n)}$。注意，这一结论仍可以得出定理 20.7 中的三个推论。本书只证明这种较弱的形式。

20.2　定理 20.6 的证明：尼散-维格德尔森构造

我们如何利用难解函数来构造伪随机数产生器呢？作为准备，我们先看两个示意性例子(Toy Example)。第一个例子将说明如何由难解函数构造出输出仅比输入长一个二进制位的伪随机数产生器。然后，第二个例子再说明如何构造出输出仅比输入长两个二进制位的伪随机数产生器。当然，这两种构造都不足以证明定理 20.6，但它们却确实展示了难解性与随机性之间的深刻联系。

407

20.2.1　两个示意性例子

利用姚期智定理将输入长度扩展一个位

下面的引理用难解函数来构造一个示意性的伪随机数产生器，它能实现将输入的长度扩展一个位。

引理 20.9（1-位随机数产生器）　假设存在 $f\in \mathbf{E}$ 使得 $H_{wrs}(f)\geqslant n^{\varepsilon}$。那么，存在 $S(\ell)$-伪随机数产生器 G，其中 $S(\ell)=\ell+1$。

证明 伪随机数产生器 G 非常简单。对于任意 $z\in\{0,1\}^\ell$，我们令

$$G(z)=z\circ f(z)$$

(其中 $\circ$ 表示拼接)。G 的输出长度和时间复杂度显然都满足 $(\ell+1)$-伪随机数产生器的要求。为了证明 G 的输出的每个位可以取自一个 $((\ell+1)^2,1/10)$-随机源，我们只需利用第 9 章的姚期智引理[⊖]证明：产生器的每个位的伪随机性由不可预测性蕴含。

定理 20.10 (重述定理 9.11) 设 Y 是 $\{0,1\}^m$ 上的一个分布。如果存在 $S>10n$ 和 $\varepsilon>0$ 使得规模至多为 $2S$ 的任意线路 C 和任意 $i\in[m]$ 都满足

$$\Pr_{r\in_R Y}[C(r_1,\cdots,r_{i-1})=r_i]\leqslant\frac{1}{2}+\frac{\varepsilon}{m}$$

则 Y 是一个 (S,ε)-伪随机源。

根据定理 20.10，要证明引理 20.9，只需证明不存在规模为 $2(\ell+1)^3<\ell^4$ 的线路 C 和整数 $i\in[\ell+1]$ 使得

$$\Pr_{z\in G(U_\ell)}[C(r_1,\cdots,r_{i-1})=r_i]>\frac{1}{2}+\frac{1}{20(\ell+1)}\tag{20.2}$$

注意，对任意 $i\leqslant\ell$，$G(z)$ 的第 i 个位都严格服从均匀分布，并且与前面的 $i-1$ 个位都是相互独立的。因此任意规模的线路都不能以大于 1/2 的概率预测到它。当 $i=\ell+1$ 时，等式(20.2)就变成了

$$\Pr_{z\in_R\{0,1\}^\ell}[C(z)=f(z)]>\frac{1}{2}+\frac{1}{20(\ell+1)}>\frac{1}{2}+\frac{1}{\ell^4}$$

它在 $H_{\max}(f)\geqslant n^4$ 的假设条件下不可能成立。 ■

利用均值原理将输入长度扩展两个位

我们继续"婴儿学步"的过程，考虑下一个自然的示意性例子：造出一个伪随机数产生器将输入扩展两个位。这一构造由下面的引理完成。 408

引理 20.11 (2-位伪随机数产生器) 假设存在 $f\in\mathbf{E}$ 使得 $H_{\mathrm{avg}}(f)\geqslant n^4$。那么，存在 $(\ell+2)$-伪随机数产生器 G。

证明 这种构造也非常自然。对任意 $z\in\{0,1\}^\ell$，我们令

$$G(z)=z_1\cdots z_{\ell/2}\circ f(z_1,\cdots,z_{\ell/2})\circ z_{\ell/2+1}\cdots z_\ell\circ f(z_{\ell/2+1},\cdots,z_\ell)$$

同样，G 的输出长度和时间复杂度显然都满足 $\ell+2$-伪随机数产生器的要求。要证明 $G(U_\ell)$ 是 $((\ell+2)^2,1/10)$-随机源，我们再次使用定理 20.10。因此，我们只需证明不存在规模为 $2(\ell+1)^3$ 的线路 C 和整数 $i\in[\ell+2]$ 使得

$$\Pr_{z\in G(U_\ell)}[C(r_1,\cdots,r_{i-1})=r_i]>\frac{1}{2}+\frac{1}{20(\ell+2)}\tag{20.3}$$

同样，如果 $G(z)$ 的第 i 位真是随机选取的，则(20.3)式不可能在 i 上成立。因此，我们只需考虑 $i=\ell/2+1$ 和 $i=\ell+2$ 的情形。利用引理 20.9 一样的论证方法，可以证明当 $i=\ell/2+1$ 时(20.3)式不成立。对于 $i=\ell+2$，等式(20.3)将变成

$$\Pr_{r,r'\in_R\{0,1\}^{\ell/2}}[C(r\circ f(r)\circ r')=f(r')]>\frac{1}{2}+\frac{1}{20(\ell+2)}\tag{20.4}$$

这种分析似乎存在一些问题，因为 C 的输入中出现了二进制位 $f(r)$，但 C 自身却无法计算 $f(r)$(因为 f 是一个难解函数)。难道输入 $f(r)$ 不能帮助 C 来预测 $f(r')$ 吗？答案是

⊖ 虽然姚期智引理表述为一致图灵机的形式，但其证明过程很容易扩展到线路上来，参见习题 20.5。

不能，因为 r 和 r' 是相互独立的。正式地讲，我们可以用如下原理(参见 A.2.2 节)来完成证明：

均值原理(Averaging Principle)：如果随机事件 A 依赖于两个独立的随机变量 X，Y，则存在 X 的某个取值 x 使得 $\Pr_Y[A(x,Y)] \geqslant \Pr_{X,Y}[A(X,Y)]$。

运用均值原理可知，如果(20.4)式成立，则存在位串 $r \in \{0,1\}^{\ell/2}$ 使得

$$\Pr_{r' \in_R \{0,1\}^{\ell/2}}[C(r \circ f(r) \circ r') = f(r')] > \frac{1}{2} + \frac{1}{20(\ell+2)}$$

(注意，上式的概率只与 r' 的随机选择有关。)如果情况果真如此，则我们可以将 $\ell/2+1$ 个二进制位 $r \circ f(r)$“固化”到线路 C 中，由此得到一个规模至多为 $2(\ell+1)^2 < (\ell/2)^4$ 的线路 D，它使得

$$\Pr_{r' \in_R \{0,1\}^{\ell/2}}[D(r') = f(r')] > \frac{1}{2} + \frac{1}{20(\ell+2)}$$

这与 f 是难解函数矛盾。 ■

将输入扩展两个位以上

将输入扩展两个位的伪随机数产生器仍不足以实现我们的目标。我们可以推广引理 20.11 的证明过程，得到将输入扩展 k 个位的伪随机数产生器，这只需令

409

$$G(z_1,\cdots,z_\ell) = z^1 \circ f(z^1) \circ z^2 \circ f(z^2) \circ \cdots \circ z^k \circ f(z^k) \tag{20.5}$$

其中 z^i 表示将 z 划分为 k 个区组(每个区组含 ℓ/k 个位)之后的第 i 个区组。然而，无论我们将 k 取多大，也无论 f 是多么难解的函数，这种方式构造得到的伪随机数产生器的输出长度都不可能是输入长度的 2 的幂次倍。注意，要证明定理 20.6，我们所要的伪随机数产生器的输出长度必须是输入长度的指数！显然，构造过程还需要采用其他新思想。

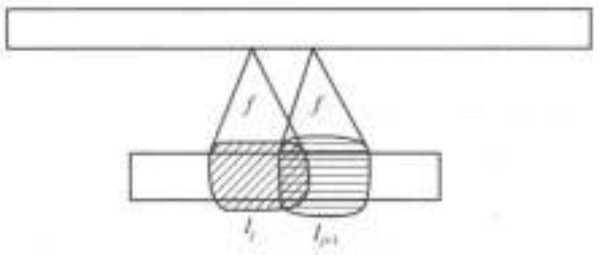
20.2.2　尼散-维格德尔森构造

我们的新思想仍受(20.5)式的启发，但我们不再将 z^1，…，z^k 取为相互独立的位串(或等价地说，是输入 z 中互不重叠的一些区组)，而是通过组合设计将 z^1，…，z^k 取为部分依赖的位串(一些有重叠的区组)。这种做法使得 k 可以取非常大的值，并且它还使得我们可以从伪随机数产生器的输出中删掉原始输入中的二进制位，而只将 $f(z^1) \circ f(z^2) \circ \cdots \circ f(z^k)$ 作为输出。为了证明方法的正确性，类似于前面示意性例子中的证明过程，我们也将使用姚期智定理，只是由于位串之间的依赖性，我们需要采用更仔细的方法来固定输入中的某些二进制位。

定义 20.12　(NW 伪随机数产生器)　设 $\mathcal{I} = \{I_1, I_2, \cdots, I_m\}$ 是由 $[\ell]$ 的子集构成的集族，其中 $|I_j| = n$ 对任意 j 成立，并且 $f: \{0,1\}^n \to \{0,1\}$ 是一个函数。$(\mathcal{I}, f)$-NW 伪随机数产生器指的是函数 $\mathrm{NW}^f_{\mathcal{I}}: \{0,1\}^\ell \to \{0,1\}^m$，它将任意 $z \in \{0,1\}^\ell$ 映射为

$$\mathrm{NW}^f_{\mathcal{I}}(z) = f(z_{I_1}) \circ f(z_{I_2}) \circ \cdots \circ f(z_{I_m}) \tag{20.6}$$

其中对于 $z \in \{0,1\}^\ell$ 和 $I \subseteq [\ell]$，z_I 表示 z 限制在 I 中坐标位置上得到的位串。



集族和函数应满足的条件

我们将看到，为了使 NW 伪随机数产生器产生伪随机数，函数 f 需要具有一定的难度，而集族则需要来自组合设计。组合设计的定义如下。

定义 20.13（组合设计）　设 d，n，ℓ 满足 $\ell > n > d$。称 $[\ell]$ 的子集构成的集族 $\mathcal{I}=\{I_1, I_2, \cdots, I_m\}$ 是一个 (ℓ, n, d) 组合设计，如果 $|I_j|=n$ 对任意 j 成立并且 $|I_j \cap I_k| \leqslant d$ 对任意 $j \neq k$ 成立。

下面的引理告诉我们如何高效地构造组合设计（引理的证明将推迟到本节末尾）。 410

引理 20.14（组合设计的构造）　存在如下的算法 A，它以 (ℓ, n, d) 为输入，其中 $n > d$ 且 $\ell > 10n^2/d$，其时间复杂度为 $2^{O(\ell)}$ 并输出一个由 $[\ell]$ 的 $2^{d/10}$ 个子集构成的 (ℓ, n, d)-组合设计 $\mathcal{I}$。

下一个引理表明，如果 f 是一个难解函数而 $\mathcal{I}$ 是一个参数足够好的组合设计，则 $\mathrm{NW}_{\mathcal{I}}^{f}(U_r)$ 将是一个伪随机源的分布。

引理 20.15（NW 产生器的伪随机性）　如果 $\mathcal{I}$ 是一个 (ℓ, n, d)-组合设计，其中 $|\mathcal{I}|=2^{d/10}$，并且函数 $f: \{0,1\}^n \to \{0,1\}$ 满足 $\mathrm{H}_{\mathrm{wo}}(f) > 2^{d/2}$，则分布 $\mathrm{NW}_{\mathcal{I}}^{f}(U_r)$ 是 $(\mathrm{H}_{\mathrm{wo}}(f)/10, 1/10)$-随机源。

证明　令 $S=\mathrm{H}_{\mathrm{wo}}(f)$。由姚期智定理可知，要证明分布 $\mathrm{NW}_{\mathcal{I}}^{f}(U_r)$ 是 $(S/10, 1/10)$-伪随机源，只需证明在任意 $i \in [2^{d/10}]$ 上均不存在规模为 $S/2$ 的线路 C 使得

$$\Pr_{\substack{z \sim U_r \\ R=\mathrm{NW}_{\mathcal{I}}^{f}(z)}}[C(R_1, \cdots, R_{i-1}) = R_i] \geqslant \frac{1}{2} + \frac{1}{10 \cdot 2^{d/10}} \tag{20.7}$$

为了利用反证法，假设(20.7)式对某个线路 C 和某个 i 成立。将 $\mathrm{NW}_{\mathcal{I}}^{f}$ 的定义代入，则等式(20.7)将变成

$$\Pr_{z \sim U_r}[C(f(Z_{I_1}), \cdots, f(Z_{I_{i-1}})) = f(Z_{I_i})] \geqslant \frac{1}{2} + \frac{1}{10 \cdot 2^{d/10}} \tag{20.8}$$

令 Z_1 和 Z_2 分别表示 Z 中坐标位置属于 I_i 和 $[\ell] \setminus I_i$ 的所有二进制位构成的位串，则 Z_1，Z_2 是独立的随机变量，并且(20.8)式将变成

$$\Pr_{\substack{z_1 \sim U_n \\ z_2 \sim U_{r-n}}}[C(f_1(Z_1, Z_2), \cdots, f_{i-1}(Z_1, Z_2)) = f(Z_1)] \geqslant \frac{1}{2} + \frac{1}{10 \cdot 2^{d/10}} \tag{20.9}$$

其中对于任意 $j \in [2^{d/10}]$ 而言，f_j 将 f 运用到 Z_1 中坐标位置属于 $I_j \cap I_i$ 和 Z_2 中坐标位置属于 $I_j \setminus I_i$ 的所有二进制位构成的位串上。由均值原理可知，如果(20.9)成立，则存在位串 $z_2 \in \{0,1\}^{r-n}$ 使得

$$\Pr_{z_1 \sim U_n}[C(f_1(Z_1, z_2), \cdots, f_{i-1}(Z_1, z_2)) = f(Z_1)] \geqslant \frac{1}{2} + \frac{1}{10 \cdot 2^{d/10}} \tag{20.10}$$

现在，我们好像遇到了麻烦。因为所有 $f_j(Z_1, z_2)$（其中 $j \leqslant i-1$）都依赖于 Z_1，这不免让人担心它们组合在一起将会包含 Z_1 的足够多的信息，因此当某个线路看到所有 $f_j(Z_1, z_2)$ 后很可能会预测出 $f(Z_1)$。利用 $\mathcal{I}$ 是一个组合设计而 f 是一个难解函数，我们可以证明所担心的情况不会发生。

由于 $|I_j \cap I_i| \leqslant d$ 对任意 $j \neq i$ 成立，故函数 $Z_1 \mapsto f_j(Z_1, z_2)$（其中 z_2 是固定的）至多依赖于 Z_1 中的 d 个位，进而这个函数可以很容易用规模为 $d2^d$ 的线路来计算（甚至还可以用规模为 $O(2^d/d)$ 的线路来计算）。因此，如果(20.9)式成立，则存在一个规模为 411
$2^{d/10} \cdot d2^d + S/2 < S$ 的线路 B 使得

$$\Pr_{z_i \sim U_n}[B(Z_i) = f(Z_i)] \geqslant \frac{1}{2} + \frac{1}{10 \cdot 2^{d/10}} > \frac{1}{2} + \frac{1}{S} \tag{20.11}$$

但这与 $S = \mathrm{H}_{\mathrm{wc}}(f)$ 的事实相矛盾。 ■

引理 2.15 的证明过程表明，如果 $\mathrm{NW}_I^f(U_r)$ 和均匀分布 $U_{2^{d/10}}$ 可以被某个线路 D 区分，则存在(规模是 D 的规模和 2^d 的多项式的)线路 B 以显著高于 1/2 的概率来计算函数 f。实际上，线路 B 可以这样实现：它将线路 D 作为一个黑盒，并在一些选定的输入上调用这个黑盒。NW 伪随机数产生器(以及其他伪随机数产生器)的这种性质在几种场合下大有用处。特别地，习题 20.7 要求在人们广泛认同的复杂性假设下，利用该性质来证明复杂性类 **AM** 等于 **NP**，其中 **AM** 包含具有常数个回合的交互式证明的所有语言(参见第 8 章)。在第 21 章中，我们还将利用该性质在伪随机数产生器的基础上构造随机提取器(Randomness Extractor)。

综合应用引理 20.14 和引理 20.15 来证明定理 20.6

正如评注 20.8 中说明的那样，我们并不证明定理 20.6 所表述的完整结论，而是只证明它的一种较弱形式——如果给定 $f \in \mathbf{DTIME}(2^{O(n)})$ 并且 S：$\mathbf{N} \to \mathbf{N}$ 满足 $\mathrm{H}_{\mathrm{wc}}(f) \geqslant S$，则我们可以构造一个 $S'(\ell)$-伪随机数产生器，其中 $S'(\ell) = S(n)^{\varepsilon}$ 对某个 $\varepsilon > 0$ 成立且 n 满足 $n \geqslant \varepsilon \sqrt{\ell \log S(n)}$。在输入 $z \in \{0,1\}^{\ell}$ 上，我们的伪随机数产生器如下进行操作：

- 令 n 是满足 $\ell > \dfrac{100n^2}{\log S(n)}$ 的最大整数。于是，$\ell \leqslant \dfrac{100(n+1)^2}{\log S(n+1)} \leqslant \dfrac{200n^2}{\log S(n)}$，进而 $n \geqslant \sqrt{\ell \log S(n)/200}$。
- 令 $d = \log S(n)/10$。
- 运行引理 20.14 中的算法得到一个 (ℓ, n, d)-组合设计 $\mathcal{I} = \{I_1, \cdots, I_{2^{d/10}}\}$。
- 输出 $\mathrm{NW}_I^f(z)$ 的前 $S(n)^{1/10}$ 个二进制位。

上面的伪随机数产生器将函数 f 调用了 $2^{d/10}$ 次，这些调用总共花费 $2^{O(n)+d}$ 个计算步骤。将 n 减小一个恰当的常数因子，我们可以确保该伪随机数产生器的运行时间以 2^{ℓ} 为上界。而且，由于 $2^{\ell} \leqslant S(n)^{1/10}$，引理 20.15 意味着分布 $\mathrm{NW}_I^f(U_\ell)$ 是一个 $(S(n)/10, 1/10)$-伪随机源。 ■

组合设计的构造

要完成整个证明，剩下的事情就是说明如何才能构造出具有所需参数的组合设计。

引理 20.14 的证明　(组合设计的构造)在满足 $\ell > 10n^2/d$ 的输入 ℓ，d，n 上，我们的算法 A 将利用贪心思想构造一个含有 $2^{d/10}$ 个子集的 (ℓ, n, d)-组合设计。

从 $\mathcal{I} = \varnothing$ 开始，在构造了 $\mathcal{I} = \{I_1, \cdots, I_m\}$ 之后，如果 $m < 2^{d/10}$，则搜索 $[\ell]$ 的所有规模为 n 的子集，找出满足条件(*)"$|I \cap I_j| \leqslant d$ 对任意 $j \in [m]$ 成立"的第一个集合 I，并将它加入 $\mathcal{I}$ 中。

显然，算法 A 的运行时间为 $\mathrm{poly}(m)2^{\ell} = 2^{O(\ell)}$。因此，我们只需证明算法 A 不会遇到困难。也就是说，只需证明：如果 $\ell = 10n^2/d$ 且 $\{I_1, \cdots, I_m\}$ 是由 $[\ell]$ 的规模为 n 的子集构成的集族，其中 $m < 2^{d/10}$，则必然存在一个规模为 n 的子集 $I \subseteq [\ell]$ 满足条件(*)。我们如下地证明这一点：以概率 $2n/\ell$ 将每个元素 $x \in [\ell]$ 随机独立地放入 I，则

412

$$\Pr[|I| \geqslant n] \geqslant 0.9 \tag{20.12}$$

$$\text{对任意 } j \in [m],\ \Pr[|I \cap I_j| \geqslant d] \leqslant 0.5 \cdot 2^{-d/10} \tag{20.13}$$

由于 $|I|$ 的数学期望等于 $2n$，而 $|I \cap I_j|$ 的数学期望是 $2n^2/\ell < d/5$，故(20.13)式和(20.12)式都可以由切尔诺夫界得到。此外，又由于 $m \leqslant 2^{d/10}$，再结合(20.13)式和

(20.12)式则意味着"集合 I 满足(*)并且$|I|\geqslant n$"的概率至少为 0.4。由于我们将 I 中多余的元素移除不会影响条件(*)，这就完成了证明。 ■

20.3　一致假设下的去随机化

线路下界的证明是臭名昭著的难题。尽管人们为之努力了几十年，但如今却仍未找出 **NP** 中的需要规模超过 $5n$ 的线路才能计算的函数，更遑论需要超线性规模或超多项式规模的线路才能计算的函数。自然地，人们不禁要问：这种线路下界对去随机化而言是否是必需的[⊖]。注意，定义 20.2 所定义的伪随机数产生器很容易推导出线路下界，参见习题 20.4。因此，人们除了能用伪随机数产生器来证明 **BPP**=**P** 之外，也不能排除能用其他方法证明这一结论的可能性。

下面的引理表明，在某种程度上讲，用其他方法来证明 **BPP**=**P** 确实是有可能的。该引理是说，在一致难度的假设(亦即，**BPP**≠**EXP**)下，我们也可以通过非平凡的方法消除 **BPP** 的随机性。

定理 20.16（一致去随机化[IW98]）　假设 **BPP**≠**EXP**。那么，对于任意 $L\in$ **BPP**，都存在亚指数时间(亦即，$2^{n^{\varepsilon}}$)的确定型算法 A 使得它在无穷个 n 上满足

$$\Pr_{x\in_U\{0,1\}^n}[A(x)=L(x)]\geqslant 1-1/n$$

定理 20.16 表明，除非随机方法是灵丹妙药(亦即，任意具有指数时间算法的计算问题，如 3SAT，TQBF，积和式等，都能找到概率型多项式时间算法)，否则我们至少可以部分地消除 **BPP** 的随机性。也就是说，我们能用亚指数时间的确定型算法以较低的平均复杂性来模拟概率型多项式时间算法。事实上，定理 20.16 的结论还可以进一步加强为：存在亚指数时间算法 A，它不仅在输入服从均匀分布时能以 $1-1/n$ 的概率求解 L，而且在输入服从任意可多项式时间抽样的分布时也能以同样的概率求解 L。这就是说，虽然这种算法模拟过程可能有失效的时候，但却很难实际地找出令模拟失效的输入！ 413

定理 20.16 的证明概要　这里，我们只给出定理 20.16 的证明概要。首先，我们注意到证明过程可以假设 **EXP**$\subseteq$**P**$_{/poly}$；因为否则的话，**EXP** 中的某个语言将具有超多项式规模的线路复杂度，进而这个语言可以用来构造出一个足够强的伪随机数产生器使得定理 20.16 的结论成立。(上述推理可以由定理 20.7 和习题 20.8 得出。)其次，由于 **EXP**$\subseteq$**P**$_{/poly}$，因此 **EXP** 含于多项式分层中(参见定理 6.20 和后面的引理 20.18)。但这又意味着 **EXP**=**PH**，进而由户田定理(定理 17.14)和瓦利安特定理(定理 17.11)可知，在多项式时间归约下积和式函数 perm 是 **EXP** 完全的。此外，定理 20.16 的假设条件还意味着 perm 不属于 **BPP**。这是整个证明的关键，因为 perm 是一个具有向下自归约性(参见第 8 章)的非常特殊的函数。

接下来的想法是，将 perm 作为难解函数来构造一个伪随机数产生器 G 使其输出的长度是超多项式的。构造过程可以根据定理 19.27 和定理 20.6 的证明来完成，在此我们略去其细节(构造过程需要注意的问题是：积和式函数的输出不是一个二进制位，但这一问题可以用第 9 章中的戈德赖希-勒维定理来解决)。下面考虑伪随机数产生器 G 的正确性证明，可以证明我们可以得到一个算法 T，它能将"区分 G 输出的分布和均匀分布的任意区

⊖ 注意，我们对图灵机下界也知之甚少。但是，从经验上看，证明 **BPP**≠**EXP** 这样的结论似乎比证明线路下界要容易一些，并且证明这种结论的第一步自然是在不做这种下界假设的前提下得出去随机化的相关结论。

分器"转变为一个"计算 perm_n 的具有多项式规模的线路 C_n"(其中 perm_n 表示积和式函数在长度为 n 的输入上的限制)。算法 T 与在标准 NW 伪随机数产生器的证明过程(定理 20.6 的证明过程)中给出的变换算法非常相似。算法 T 的计算效率不能满足定理证明的要求,唯一的原因在于,它需要在随机选定的一些输入上计算难解函数(这里指的是积和式函数)的函数值,而这些函数值正是我们要"固化"到区分器中的信息[⊖]。

为了利用反证法,我们假设定理 20.16 的结论不成立。这意味着:存在一个概率算法 A 使得:在除有限种输入长度之外的其余所有输入长度上,用 G 都无法高概率地(相对于输入的随机选择而言)消除 A 的随机性。这意味着不仅存在着一族多项式规模的线路 $\{D_n\}$ 能在除有限种输入长度之外的其余所有输入长度上区分 G 输出的分布和均匀分布,而且还存在一个多项式时间的概率型算法在输入 1^n 上能至少以 $1/n$ 的概率输出线路 D_n(习题 20.9)。现在,我们简化地假设该概率算法以较高的概率(不妨设概率为 $1-1/n^2$)输出区分器 D_n[⊖]。将这个概率算法代入产生 G 的伪随机性的证明过程中,则意味着存在一个多项式时间的概率算法 T 来"学习"积和式函数:以 perm_n(perm 在长度为 n 的输入上的限制)为神喻,算法 T 可以在 $\mathrm{poly}(n)$ 时间内输出一个能够计算 perm_n 且规模为 $\mathrm{poly}(n)$ 的线路。

然而,利用算法 T,我们能够构造一个不用神喻却也能计算积和式函数的多项式时间概率算法! 为了在长度为 n 的输入上计算积和式,我们可以归纳地计算线路 C_1,…,C_n。给定线路 C_{n-1},我们可以利用积和式函数的向下自归约性在长度为 n 的输入上计算积和式(参见 8.6.2 节和后面引理 20.19 的证明),这样就实现了算法 T 的神喻,进而可以构造出 C_n。由于我们假设 $\mathbf{BPP}\neq\mathbf{EXP}$,并且在 $\mathbf{EXP}\subseteq\mathbf{P}_{/\mathrm{poly}}$ 的假设下积和式函数是 **EXP** 完全的,由此产生了矛盾。■

414

20.4　去随机化需要线路下界

20.3 节证明了线路下界可用于去随机化。但是,线路下界的获得却非常棘手,因此我们希望去随机化不需借助线路下界。本节将证明这是不可能的。也就是说,只要证明了 $\mathbf{BPP}=\mathbf{P}$ 或者证明了 **BPP** 中的某个特定问题(如 ZEROP 问题——判定给定的多项式是否为零多项式)属于 **P**,则我们必然能得到布尔线路或代数线路的超多项式下界。这一结论既有乐观的一面,也有悲观的一面。从乐观的一面讲,我们可以认为 **BPP** 的去随机化是很难的;从悲观的一面讲,我们将有理由怀疑人们在 **BPP** 的去随机化方面所做的所有努力。因为一旦去随机化获得成功,则将会达到"一石二鸟"的效果——既完成了去随机化,又得到了线路下界。

回顾一下定义 16.7,我们称定义在整数上的函数 f 属于 $\mathbf{AlgP}^{\mathbb{Z}}_{\mathrm{poly}}$(或简记为 $\mathbf{AlgP}_{\mathrm{poly}}$),如果 f 可以用多项式规模的代数线路来计算,其中代数线路的每个门被标记为+,−和×。[⊖]我们令 perm 表示整数矩阵上积和式的计算问题。此外,还需注意,在多项式恒等测试问题 ZEROP 中,问题的输入是用算术线路表示的一个多项式,问题要求判定输入的多项式是

⊖ 定理 20.6 只证明了:存在一些输入,只要将这些输入及其相应的答案"固化"到区分器中就可以得到计算难解函数的线路。但是,由于证明过程使用了概率方法/均值论证法,不难证明随机选定的一些输入也可以确保所得线路以较高的概率正确计算难解函数。

⊖ 这个新的概率可以用如下事实得出:积和式是一个次数较低的多项式,进而它具有一定的子纠错性或自测试性。参见 8.6.2 节和 19.4.2 节。

⊖ 下面使用的线路是代数线路在有理数和实数上的扩展,此时线路中的操作还可以使用除法。

否是一个恒等于零的多项式(参见例 20.1 和 7.2.3 节)。问题 ZEROP 属于 **coRP**⊆**BPP**，我们将证明：如果 ZEROP 属于 **P**，则我们将得到某个超多项式的线路下界。

定理 20.17（去随机化蕴含线路下界[KI03]）　如果 ZEROP∈**P**，则要么 **NEXP**⊄$\mathbf{P}_{/poly}$，要么 perm∉$\mathbf{AlgP}_{/poly}$。

已经证明，即使将定理 20.17 的假设条件放宽为 ZEROP∈$\bigcap_{\varepsilon>0}$**NTIME**(2^{n^ε})，定理仍然成立。这就是说，即使仅用亚指数非确定型时间算法来消除 **BPP** 的随机性，我们仍可以得到超多项式的线路下界。定理 20.17 的证明要借助本书前面介绍的很多结论。(这是"第三代"复杂性结论中的一个很好的例子，它的证明需要巧妙地结合 20 世纪 60、70 年代得到的"经典"复杂性结果和 20 世纪 90 年代以来得到的较新的结果。)下面的引理是证明过程需要的第一个结果。

引理 20.18（[BFL90]，[BFNW93]）　**EXP**⊆$\mathbf{P}_{/poly}$⇒**EXP**=**MA**。　415

注意，**MA** 是由能用亚瑟(Arthur)和梅林(Merlin)之间仅含一个回合的交互式证明系统证明的所有语言构成的复杂性类(参见定义 8.10)。

引理 20.18 的证明　假设 **EXP**⊆$\mathbf{P}_{/poly}$。根据迈耶定理(定理 6.20)可知，**EXP** 坍塌到多项式分层中的第二层 Σ_2。因此，在我们的假设下，Σ_2=**PH**=**PSPACE**=**IP**=**EXP**⊆$\mathbf{P}_{/poly}$。于是，任意 L∈**EXP** 均存在交互式证明。而且，由于我们的假设条件蕴含了 **EXP**=**PSPACE**，因此我们可以只使用 TQBF 的交互式证明系统，它的证明者是一个 **PSPACE** 图灵机。在 **PSPACE**⊆$\mathbf{P}_{/poly}$的条件下，这个交互式证明系统的证明者还可以替换为多项式规模的线路族$\{C_n\}$。现在，我们可以看到上述交互式证明系统实际上可以在一个回合内完成：在长度为 n 的输入上，梅林将一个多项式规模的线路 C 发送给亚瑟，亚瑟认为 C 就是证明者判定 L 成员资格的策略 C_n。然后，亚瑟模拟对 L 的交互式证明过程，模拟过程中他将 C 视为证明者并通过投掷硬币来模拟验证者。注意，如果输入不属于语言 L，则任意证明者都无法以非零的概率让验证者相信输入属于 L。特别地，这个条件对 C 所描述的证明者也成立。于是，我们得到了 L 的一个 **MA** 协议。这就意味着 **EXP**⊆**MA**，进而 **EXP**=**MA**。■

我们的第二个引理建立了多项式恒等测试和积和式计算与复杂性类 $\mathbf{P}^{perm}$ 之间的联系。

引理 20.19　如果 ZEROP∈**P** 且 perm∈$\mathbf{AlgP}_{/poly}$，则 $\mathbf{P}^{perm}$∈**NP**。

引理 20.19 的证明　假设 perm 存在规模为 n^c 的线路，并且 ZEROP 存在多项式时间算法。令 L∈$\mathbf{P}^{perm}$是任意的语言，不妨设它可以用以 perm 为神喻的图灵机 M 在 n^c 时间内判定。我们下面为 L 构造一个 **NP** 图灵机 N。

设 x 是长度为 n 的输入。显然，M 在 x 上完成计算的过程中它向 perm 神喻咨询的每个问题的长度至多为 $m=n^c$。因此，N 可以如下使用非确定性：它先猜测 m 个代数线路 C_1，C_2，…，C_m，其中 C_i 的规模为 i^c，N 希望线路 C_i 能在 $i\times i$ 的矩阵上求解 perm，它可以在 poly(m)时间内验证情况是否果真如此。验证过程从验证 C_1 开始，该验证是平凡的。归纳地，在验证了 C_1，…，C_{t-1} 的正确性之后，N 在验证 C_t 的正确性时可以借助积和式的向下自归约性。也就是说，对于 $t\times t$ 的矩阵 A

$$\mathrm{perm}(A)=\sum_{i=1}^{t}a_{1i}\mathrm{perm}(A_{1,i})$$

其中 $A_{1,i}$ 是从 A 中删除第 1 行和第 i 列之后得到的 $(t-1)\times(t-1)$ 矩阵。因此，如果已知

C_{n-1}是正确的，则在验证C_n的正确性时可以将上式中的perm(A)替换为$C_t(A)$，同时将perm(A_{11})也替换为$C_{n-1}(A_{11})$。由此得到一个恒等式，它的两端各自与输入长度为t^2的一个代数线路相关联，该恒等式可以借助ZERO#P的算法在poly(t)时间内进行验证。重复上述验证过程，N就可以验证C_1，C_2，…，C_m中每个C_t的正确性。然后，N可以利用线路C_1，C_2，…，C_m来模拟M^{poly}在x上的运行过程。 ■

416 定理20.17的证明过程的核心是下面的引理，该引理本身也具有独立的意义。

引理20.20 ([IKW01]) **NEXP**$\subseteq$**P**$_{poly}$⇒**NEXP**=**EXP**。

证明 我们证明定理的逆否命题。也就是说，我们假设**NEXP**≠**EXP**，则我们将可以证明**NEXP**$\not\subseteq$**P**$_{poly}$。设$L\in$**NEXP****EXP**(在我们的假设条件下，这样的语言是存在的)。由于$L\in$**NEXP**，故存在常数$c>0$和关系R使得

$$x\in L\Leftrightarrow\exists y\in(0,1)^{2^{|x|^c}}\text{ 使得 }R(x,y)\text{ 成立}$$

其中$R(x,y)$是否成立可以在$2^{|x|^{2c}}$时间内被验证。

现在，考虑用如下方法尝试在确定型指数时间内求解L。对任意常数$D>0$，令M_D是如下的图灵机：在输入$x\in\{0,1\}^n$上，M_D枚举输入长度为n^c并且仅输出一个位的所有规模为n^{100D}的布尔线路C。对每个这样的线路C，令tt(C)是一个长度为2^n的位串，它对应C所计算的函数的真值表。如果$R(x,\mathrm{tt}(C))$对C成立，则M_D停机并输出1。如果$R(x,\mathrm{tt}(C))$对所有C都不成立，则M_D输出0。由于M_D的运行时间是$2^{n^{100D}+n^c}$，并且我们假设了$L\notin$**EXP**，故M_D不能判定L的成员资格。因而，对任意D均存在无穷的输入序列$\mathcal{X}_D=\{x_i\}_{i\in\mathbb{N}}$使得$M_D(x_i)$输出0但$x_i\in L$(注意，$M_D$只会犯单面错误)。这也就是说，对于序列$\mathcal{X}_D$中的任意$x$和满足$R(x,y)$的任意$y$，位串$y$所表示的函数(其输入长度为$n^c$)不能用规模为$n^{100D}$的线路来计算，其中$n=|x|$。利用最坏难性假设下的伪随机数产生器(定理20.7)，我们可以用这样一个位串y来构造一个ℓ^D-伪随机数产生器。这种方法被称作"简单证据法(Easy Witness Method)"[Kab00]，它获得这个名字的缘由是它证明了：除非输入x存在可以由小规模线路计算的简单证据y(亦即，满足$R(x,y)=1$的位串)，否则x的任意证据都可以用于去随机化。

现在，如果**NEXP**$\subseteq$**P**$_{poly}$，则**EXP**$\subseteq$**P**$_{poly}$，进而由引理20.18可得**EXP**$\subseteq$**MA**。也就是说，**EXP**中的任意语言都存在如下的交互式证明系统：梅林(Merlin)通过向亚瑟(Arthur)发送"证明"来证明给定的长度为n的输入位串属于语言，然后亚瑟利用概率算法在至多n^D步骤内验证所接收到的"证明"，其中D是某个常数。于是，如果n是序列$\mathcal{X}_D$中某个输入位串的长度，则给定$|x|=n$的$x\in\mathcal{X}_D$之后亚瑟使用的概率算法可以如下替换为一个时间复杂度为poly(n^D)$2^{n^{2c}}$的非确定型算法，其中不再涉及随机选择：这个非确定型算法先猜测一个使得$R(x,y)=1$的位串y，然后用y构造一个伪随机数产生器来验证梅林给出的证明。

这意味着，存在一个绝对常数$c>0$使得：**EXP**中的每个语言在无穷个输入上的成员资格都可以用**NTIME**(2^{n^c})时间的算法来判定，只是该算法需要以长度为n的串作为建言。由于我们假设了**NEXP**$\subseteq$**P**$_{poly}$，因此**EXP**中的每个语言在无穷个输入上的成员资格也可以用规模为$n^{c'}$的线路族来判定，其中c'是一个绝对常数。但是，利用标准的对角线方法，我们很容易构造一个属于**DTIME**($2^{O(n^{c'})}$)$\subseteq$**EXP**的语言使得它几乎在所有输入上都不能被这样的线路计算。 ■

套用引理20.18(**EXP**$\subseteq$**P**$_{poly}$⇒**EXP**=**MA**)的证明过程似乎可以得到引理20.20的更简

单的证明，因为这只需将证明过程中 TQBF 的交互式证明系统替换为 **NEXP** 的多证明者交互式证明系统即可。但是，人们目前仍不清楚如何才能实现 **NEXP** 的多证明者交互式证明系统中证明者的策略。从直观上看，引起这种问题的根源在于，**NEXP** 论断可能存在多个 [417]
"证明"，但我们却不知道如何才能确保多个证明者都采用同一个"证明"。

至此，我们完成了证明定理 20.17 的所有准备工作。

定理 20.17 的证明　为了导出矛盾，假设下列各式均为真：

$$\text{ZEROP} \in \mathbf{P} \tag{20.14}$$

$$\mathbf{NEXP} \subseteq \mathbf{P}_{/poly} \tag{20.15}$$

$$\text{perm} \in \mathbf{AlgP}_{/poly} \tag{20.16}$$

(20.15)式和引理 20.18、引理 20.20 一起将意味着 **NEXP**＝**EXP**＝**MA**。现在，注意 **MA**⊆**PH**，并且由户田定理(定理 17.14)可知 **PH**⊆$\mathbf{P}^{\#\mathbf{P}}$。再由瓦利安特定理(17.11)可知 perm 是 #**P** 完全的。因此，在我们的假设条件下有

$$\mathbf{NEXP} \subseteq \mathbf{P}^{\text{perm}} \tag{20.17}$$

由于我们假设 ZEROP∈**P**，引理 20.19 和(20.16)式、(20.17)式一起蕴含了 **NEXP**⊆**NP**。这与非确定时间分层定理(定理 3.2)矛盾。因此，(20.14)式，(20.15)式和(20.16)不可能同时成立。 ■

本章学习内容

- 在某种线路下界假设下，存在可以将所有概率算法去随机化的伪随机数产生器。
- 特别地，如果我们合理地假设 **E**＝**DTIME**($2^{O(n)}$)中的某个函数具有指数级的平均线路复杂度，则 **BPP**＝**P**。
- 证明 **BPP**＝**P** 需要先证明某种类型的线路下界。

本章注记和历史

正如第 9 章注记中指出的那样，伪随机数产生器最早的研究出现在密码学领域，率先对它开展研究的人包括萨米尔(Shamir)[Sha81]，布卢姆(Blum)和米卡利(Micali)[BM82]，以及姚期智[Yao82a]。特别地，姚期智率先指出了伪随机数产生器可以用于 **BPP** 的去随机化。他证明了，如果安全伪随机数产生器存在，则 **BPP** 可以部分地去随机化，亦即 **BPP**⊆$\bigcap_{\varepsilon>0}$**DTIME**($2^{n^{\varepsilon}}$)。尼散(Nisan)和维格德尔森(Wigderson)在他们的开创性论文[NW88]中证明了在明显更弱的复杂性假设下也可以将 **BPP** 去随机化，并且还证明了在人们广泛认同的复杂性假设下仍有可能将 **BPP** 完全地去随机化(亦即，证明 **BPP**＝**P**)。至那以后，人们开展了大量的工作来改进去随机化和削弱去随机化需要的假设条件(也请参阅第 19 章的注记)。特别地，人们已经证明，最坏难度假设是以完成去随机化(参见第 19 章及其注记)。这种研究的中心目标被因帕利亚佐(Impagliazzo)和维格德尔森(Wigderson)[IW97]实现，他们证明了如果 **E**＝**DTIME**($2^{O(n)}$)中存在具有指数级线路复杂度的函数，则 **BPP**＝**P**。 [418]

伪随机数产生器与难度假设之间的最佳依赖关系(定理 20.6)源自乌曼斯(Umans)[Uma03](参见评注 20.8)。有意思的是，他给出的伪随机数产生器直接基于最坏难度假设(而非平均难度假设)，并且实际上还借助了难度方法中的局部解码技术。具体地讲，乌曼斯的构造用到了第 19 章中的里德-穆勒纠错码，这种构造还以萨尔蒂尔(Shaltiel)和乌曼

斯之前发表的一篇论文[SU01]为基础，该文在同样的参数设置下构造了碰集产生器(Hitting Set Generator)(伪随机数产生器的宽松形式)。安德列夫(Andreev)，克莱门蒂(Clementi)和罗林(Rolim)[ACR96]证明了这种碰集产生器足以用于将 **BPP** 去随机化([GVW00]对此给出了一个较简单的证明)。

因帕利亚佐和维格德尔森[IW98]率先基于 **EXP** 函数的一致难度得出了去随机化的结果(亦即，定理 20.16)。这一结果表明，完成 **BPP**=**P**(至少 **BPP**≠**EXP**)的证明有希望不必等到线路下界的证明取得进展之后再进行了。然而，因帕利亚佐，卡巴涅茨(Kabanets)和维格德尔森[IKW01]却证明了 **MA**(它是 **BPP** 的诺言形式)的去随机化将得到 **NEXP** 的下界，同时卡巴涅茨和因帕利亚佐[KI03]还证明了定理 20.17。也就是说，他们证明了，一旦将 **BPP** 去随机化则必然会得到一些线路下界。

习题

20.1　证明推论 20.4。

20.2　证明：存在常数 $\varepsilon>0$ 和函数 $G:\{0,1\}^*\to\{0,1\}^*$ 满足定义 20.2 中 2^n-伪随机数产生器需要的除计算效率之外的所有条件。

20.3　用计数方法(亦即，概率方法)证明：对于任意充分大的 n，存在函数 $f:\{0,1\}^n\to\{0,1\}$ 使得 $\mathrm{H}_{\mathrm{avg}}(f)\geqslant 2^{n/10}$。

20.4　证明：如果 $S(\ell)$-伪随机数产生器存在，则存在函数 $f\in\mathbf{DTIME}(2^{O(n)})$ 使得 $\mathrm{H}_{\mathrm{wrs}}(f)(n)\geqslant S(n)$。

20.5　证明定理 20.10。

20.6　证明：如果存在 $f\in\mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 和 $\varepsilon>0$ 使得 $\mathrm{H}_{\mathrm{avg}}(f)(n)\geqslant 2^{\varepsilon n}$ 对任意 $n\in\mathbf{N}$ 成立，则 **MA**=**NP**。

20.7　我们定义神喻布尔线路是含有一种特殊门的布尔线路，这种特殊的门称为神喻门，它用 ORACLE 来标记，其扇入度是无界的。对于布尔线路 C 和语言 $O\subseteq\{0,1\}^*$，我们定义 $C^O(x)$ 是 C 在 x 上的输出，其中每个神喻门在输入 q 上输出 1 当且仅当 $q\in O$。

(a) 证明：如果任意 $f\in\mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 都能用以 SAT 作神喻的多项式规模的线路来计算，则多项式分层将塌陷。

419

(b) 对于函数 $f:\{0,1\}^*\to\{0,1\}$ 和 $O\subseteq\{0,1\}^*$，定义函数 $\mathrm{H}^O_{\mathrm{avg}}(f)$ 将任意 $n\in\mathbf{N}$ 映射为满足 $\Pr_{x\in_R\{0,1\}^n}[C^O(x)=f(x)]\leqslant 1/2+1/S$ 的最大 S。证明：如果存在 $f\in\mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 和 $\varepsilon>0$ 满足 $\mathrm{H}^{\#\mathrm{SAT}}_{\mathrm{avg}}(f)(n)\geqslant 2^{\varepsilon n}$，则 **AM**=**NP**。

20.8　证明：如果 $\mathbf{EXP}\not\subseteq\mathbf{P}_{/\mathrm{poly}}$，则定理 20.16 的结论成立。

20.9　设 $G:\{0,1\}^*\to\{0,1\}^*$ 是 $S(\ell)$-长度的伪随机数产生器，它无法消除一个特定的 **BPP** 算法 A 在平均情况下的随机性。也就是说，在满足 $\Pr[A(x)=L(x)]\geqslant 2/3$ 的某个语言 $L\in\mathbf{BPP}$ 上，对于充分大的 n，如果随机选取 $x\in_R\{0,1\}^n$ 则有 $\Pr[A(x,G(U_{\ell(n)}))=L(x)]\leqslant 1/2$(其中 $\ell(n)$ 使得 $S(\ell(n))=m(n)$ 成立，而 $m(n)$ 是算法 A 在长度为 n 的输入上运行时所使用的随机带的长度)。证明：存在如下的多项式时间概率算法 D，它在输入 1^n 上输出一个线路 D_n 使得 D_n 相对于算法 D 的随机选择而言以至少为 $1/(2n)$ 的概率满足

$$\left\|E[D_n(G(U_{\ell(n)}))]-E[D_n(U_{m(n)})]\right\|>0.1\ .$$

420　20.10　(van Melkebeek 2000，参见[IKW01])证明：如果 **NEXP**=**MA**，则 $\mathbf{NEXP}\subseteq\mathbf{P}_{/\mathrm{poly}}$。

伪随机构造：扩张图和提取器

在干草堆中找到干草能有多难？

——霍华德·卡洛夫(Howard Karloff)

概率方法是一种强有力的工具，它可以用来证明具有某种所需性质的对象(比如，函数或图)是存在的。本书前面的章节已经多次遇到概率论证法。例如，第6章用它证明了具有较高线路复杂度的函数的存在性，第19章用它证明了具有良好性质的纠错码的存在性。然而，在某些情况下，仅证明对象的存在性是不够的，我们需要明确而高效地构造出所需的对象。本章讨论的方法用于构造两类著名(并且与复杂性相关)的伪随机对象——扩张图和提取器。这两类对象在计算机科学中都非常重要，因为它们可以用来替换或减小某些场合所需的随机性。这不禁又让我们联想到第20章研究过的去随机化。事实上，纵观全章，我们将看到这两类对象与去随机化之间的几个联系。但是，与第20章的一个重大区别是，本章证明的所有结果都是无条件的。换句话说，本章证明的所有结果都不依赖于未经证明的假设条件。与扩张图相关的另一个主题是纠错码的构造以及与之相关的难度放大方面的结果(其中，难度放大参见第19章)。本章注记简明扼要地讨论了纠错码、扩张图、伪随机数产生器、提取器之间的许多深刻而迷人的联系。

扩张图指的是一种图，它的连通性(指的是顶点子集A，B之间边的条数的多少)类似于"随机"图的连通性。从这个意义上讲，扩张图是"伪随机图"或者"类随机图"。扩张图具有十分广泛的应用——从快速分类网络，到度量空间理论中的反例构造，再到**PCP**定理的证明。扩张图的研究与邻接矩阵特征值的研究之间的联系密不可分。21.2节为扩张图的研究奠定基础，我们说明如何利用邻接矩阵的特征值来分析图上的概率游走。然后，21.2节给出扩张图的两种等价定义。此外，本小节还会讨论扩张图在概率算法的随机高效错误减小方面的应用。21.3节给出扩张图的一种显式构造。最后，21.4节利用这种构造来为无向连通性问题设计一个确定型的对数空间算法。

我们的第二种显式构造与如下的问题相关。虽然随机算法被建模为随机独立地投掷一系列无偏硬币，但实践中的随机源却是非完美的，它们通常相互关联而且是偏斜的。从哲学角度讲，人们还不清楚宇宙中是否真的存在能够产生随机二进制位的无偏随机源。因此，研究者们提出了量化的方法来表示产生随机位的随机源的非完美性以及如何用这样的随机源来运行随机算法。

21.5节定义弱随机源。这一定义刻画了能用于运行随机算法的非完美随机源所需要具备的最小"随机性"。本小节还定义了随机性提取器(或简称提取器)，它是从弱随机源中提取无偏随机二进制位的一种算法，本节还给出了提取器的显式构造。这些结果的一个哲学推论是：多项式时间随机图灵机(及相应的复杂性类**BPP**)真正存在于现实世界当且仅当弱随机源真正存在于现实世界。

21.6节利用提取器来去除概率型对数空间计算中的随机性，但是去随机化将导致计算所需的空间有所增加。值得强调的是，与第19章和第20章的结果相比，本章的去随机化(同本章其他结果一样)是无条件的，它们不使用任何未经证明的假设条件。

本章讨论的构造和分析都比较复杂。读者可能会问，为什么得到显式构造如此困难呢？毕竟，基于概率方法的存在性的证明不仅已经说明了具有所需性质的对象确实是存在的，而且实际上还说明了大量的对象都具有所需的性质。正如卡洛夫所说(参见本章开头的引言)，构造一个具有所需性质的对象能有多难？或许这个任务真的很难也不足为奇，毕竟，虽然我们已知几乎所有布尔函数都具有指数级的线路下界，但真正从 **NP** 中找出一个具有指数线路下界的布尔函数却将会表明 **P**≠**NP**!

21.1　随机游走和特征值

本节研究图上的随机游走。利用初等线性代数，我们建立图的邻接矩阵的特征值和图上的随机游走的性质之间的联系。作为推论，我们将得到 7.7 节中介绍的关于无向连通性问题的空间高效的随机游走算法的正确性。这里，我们只讨论正则图，其中每个顶点都具有相同的度。但是，我们允许图中包含自环和平行边(也称为重边)。大多数定义和结果都可以恰当地推广到非正则无向图上。

一些线性代数知识

我们将使用线性空间 $\mathbf{R}^n$ 的一些基本性质。A.5 节介绍了这些知识，这里我们仅对它们进行快速浏览。如果 $u, v \in \mathbf{R}^n$ 是两个向量，则它们的内积定义为 $\langle u, v\rangle = \sum_{i=1}^{n} u_i v_i$。如果 $\langle u, v\rangle = 0$，则称 u 和 v 正交，表示为 $u \perp v$。$v \in \mathbf{R}^n$ 的 L_2-范数 $\|v\|_2$ 定义为 $\sqrt{\langle v, v\rangle} = \sqrt{\sum_{i=1}^{n} v_i^2}$。$L_2$-范数等于 1 的向量称为单位向量。一个简单而有用的事实是勾股定理(Pythagoream Theorem)，亦即，如果 u 和 v 正交，则 $\|u+v\|_2^2 = \|u\|_2^2 + \|v\|_2^2$。$v$ 的 L_1-范数 $|v|_1$ 定义为 $\sum_{i=1}^{n} |v_i|$。L_1-范数和 L_2-范数都满足如下的基本性质：(1) $\|v\| \geqslant 0$，并且 $\|v\| = 0$ 当且仅当 v 是零向量；(2) $\|av\| = |a|\,\|v\|$ 对任意 $a \in \mathbf{R}$ 成立；(3) $\|u+v\| \leqslant \|u\| + \|v\|$。$L_1$-范数和 L_2-范数之间的关系可以描述为如下论断，其证明留作习题 21.1。

422

论断 21.1　任意向量 $v \in \mathbf{R}^n$ 均满足，$|v|_1/\sqrt{n} \leqslant \|v\|_2 \leqslant |v|_1$。

21.1.1　分布向量和参数 $\lambda(G)$

设 G 是一个 d-正则的 n-顶点图，p 是 G 的所有顶点上的一个概率分布。我们可以将 p 视为 $\mathbf{R}^n$ 中的一个(列)向量，其中 p_i 是顶点 i 在分布中的概率。注意，p 的 L_1-范数等于 1。现在，令 q 是如下随机变量的分布：先根据分布 p 从 G 中随机选择一个顶点 i，然后随机地取 i 在 G 中的一个相邻顶点。通过简单计算就可以得出分布 q，因为“该随机变量等于 j 的概率 q_j”等于“j 的所有相邻顶点 i 在 p 中的概率 p_i 与 $1/d$ 的乘积”之和(因为顶点 i 关联了 d 条边，当第一步选中 i 之后，下一步随机选择通过边 $\overline{ij}$ 取中 j 的概率等于 $1/d$)。于是，$q = Ap$，其中 $A = A(G)$ 是一个矩阵，其中对于任意顶点 i，j 而言 $A_{i,j}$ 等于介于 i 和 j 之间的边的条数除以 d。换句话说，A 是 G 的邻接矩阵乘以 $1/d$。我们称 A 是 G 的随机游走矩阵。注意，A 是一个对称矩阵[⊖]，它的所有元素均介于 0 和 1 之间，并且

⊖ 如果 $A_{i,j} = A_{j,i}$ 对任意 i，j 成立，则称 A 是对称矩阵。也就是说，对称矩阵满足 $A = A^{\mathrm{T}}$，其中 A^{T} 表示 A 的转置(参见 A.5 节)。

各行、各列元素之和恰好等于 1。满足这种性质的矩阵统称为对称随机矩阵(Symmetric Stochastic Matrix)。

矩阵 A 和图 G 上的随机游走之间的联系非常直接。亦即，对于任意 $\ell \in \mathbf{N}$ 和 $i \in [n]$，向量 $A^\ell e^i$(其中向量 e^i 的第 i 个坐标分量等于 1，而其余坐标分量都等于 0)表示从顶点 i 出发进行 ℓ 步随机游走之后所达到的顶点的概率分布 X_ℓ。

定义 21.2 (参数 $\lambda(G)$) 用 $\mathbf{1}$ 表示向量 $(1/n, 1/n, \cdots, 1/n)$，它对应于均匀分布。用 $\mathbf{1}^\perp$ 表示与 $\mathbf{1}$ 正交的所有向量构成的集合，也就是说，$v \in \mathbf{1}^\perp$ 当且仅当 $\langle v, \mathbf{1} \rangle = (1/n)\sum_i v_i = 0$。

参数 $\lambda(A)$，也可以表示为 $\lambda(G)$，指的是 $\|Av\|_2$ 在满足 $\|v\|_2 = 1$ 的所有向量 $v \in \mathbf{1}^\perp$ 上达到的最大值。

与特征值的联系

$\lambda(G)$ 的值称为 G 的第二大特征值。获得这种称呼的原因如下。由于 A 是一个对称矩阵，故 A 的所有特征值 $\lambda_1, \cdots, \lambda_n$ 对应的特征向量 $v^1, \cdots, v^n$ 构成了 $\mathbf{R}^n$ 的一组正交基底(参见 A.5.3 节)。$\lambda_1, \cdots, \lambda_n$ 经过恰当排序后可以确保 $|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n|$。注意，$A\mathbf{1} = \mathbf{1}$。事实上，对任意 i，$(A\mathbf{1})_i$ 等于 A 的第 i 行与向量 $\mathbf{1}$ 的内积。又由于 A 的每行元素之和等于 1，故该内积等于 $1/n$。于是，$\mathbf{1}$ 是 A 的特征向量，它对应的特征值等于 1。可以证明，对称随机矩阵的所有特征值的绝对值都至多为 1(参见习题 21.5)。于是，我们可以假设 $\lambda_1 = 1$ 且 $v^1 = \mathbf{1}$。再由 $\mathbf{1}^\perp = \mathrm{span}\{v^2, \cdots, v^n\}$，因此前面定义的参数 $\lambda(G)$ 将在(单位化之后的)v^2 上达到最大值，进而 $\lambda(G) = |\lambda_2|$。量 $1 - \lambda(G)$ 称为图 G 的谱鸿沟(Spectral Gap)。注意，有些教科书不是用随机游走矩阵来定义参数 $\lambda(G)$，而是直接用标准的邻接矩阵(未做归一化处理)来定义它。此时，$\lambda(G)$ 是介于 0 和 d 之间的数，相应地谱鸿沟被定义为 $d - \lambda(G)$。特征值和特征向量的相关知识(全部被附录涵盖)可以为阅读本章提供有用的背景知识，但也不是严格必需才能读懂相关的结论和证明。

下面的引理解释了参数 $\lambda(G)$ 的重要性。

引理 21.3 设 G 是一个 n-顶点正则图并且 p 是 G 的所有顶点上的一个概率分布，则 $\|A^\ell p - \mathbf{1}\|_2 \leqslant \lambda^\ell$。

证明 由 $\lambda(G)$ 的定义可知，$\|Aw\|_2 \leqslant \lambda \|v\|_2$ 对任意 $v \perp \mathbf{1}$ 均成立。注意，如果 $v \perp \mathbf{1}$，则 $Av \perp \mathbf{1}$，因为 $\langle \mathbf{1}, Av \rangle = \langle A^\mathrm{T}\mathbf{1}, v \rangle = \langle \mathbf{1}, v \rangle = 0$(这是由于 $A = A^\mathrm{T}$ 且 $A\mathbf{1} = \mathbf{1}$)。因此，A 将 $\mathbf{1}^\perp$ 映射到它自身。再注意到，除 $\mathbf{1}$ 之外的所有其他特征向量张成了子空间 $\mathbf{1}^\perp$，并且 A 将每个特征向量的 L_2-范数至少缩小因子 λ。因此，A 将子空间 $\mathbf{1}^\perp$ 中的每个向量的 L_2-范数也至少缩小因子 λ。进而，A^ℓ 将子空间 $\mathbf{1}^\perp$ 中的每个向量的 L_2-范数至少缩小了因子 λ^ℓ。于是，我们得到 $\lambda(A^\ell) \leqslant [\lambda(A)]^\ell$。(事实上，根据 $\lambda(A)$ 的特征值定义，还可以证明 $\lambda(A^\ell) = [\lambda(A)]^\ell$。)

对于任意向量 p，它都可以分解成平行于 $\mathbf{1}$ 和正交于 $\mathbf{1}$ 的两个部分，进而 p 可以表达成 $p = \alpha\mathbf{1} + p'$，其中 $p' \perp \mathbf{1}$ 而 α 是一个数。如果 p 是一个概率分布，则 $\alpha = 1$，因为 p' 的各个分量之和等于 0。因此，

$$A^\ell p = A^\ell(\mathbf{1} + p') = \mathbf{1} + A^\ell p'$$

由于 $\mathbf{1}$ 与 p' 正交，故 $\|p\|_2^2 = \|\mathbf{1}\|_2^2 + \|p'\|_2^2$；特别地，$\|p'\|_2 \leqslant \|p\|_2$。由于 p 是一个概率向量，故 $\|p\|_2 \leqslant |p|_1 = 1$(参见论断 21.1)。于是，$\|p'\|_2 \leqslant 1$ 并且

$$\|A^\ell p - \mathbf{1}\|_2 = \|A^\ell p'\|_2 \leqslant \lambda^\ell. \tag{21.1}$$ ■

事实证明，任意图都具有显著的谱鸿沟。

引理 21.4 如果 G 是每个顶点上均含有自环的正则连通图，则 $\lambda(G)\leqslant 1-\frac{1}{12n^2}$。

证明 令 $\varepsilon=\frac{1}{6n^2}$，$\boldsymbol{u}\perp\mathbf{1}$ 是任意单位向量，并且 $\boldsymbol{v}=A\boldsymbol{u}$。我们需要证明 $\|\boldsymbol{v}\|_2\leqslant 1-\varepsilon/2$。为此，只需证明 $1-\|\boldsymbol{v}\|_2^2\geqslant\varepsilon$。（事实上，如果 $\|\boldsymbol{v}\|_2>1-\varepsilon/2$，则 $\|\boldsymbol{v}\|_2^2>1-\varepsilon$，进而 $1-\|\boldsymbol{v}\|_2^2<\varepsilon$。）由于 $\boldsymbol{u}$ 是单位向量，故 $1-\|\boldsymbol{v}\|_2^2=\|\boldsymbol{u}\|_2^2-\|\boldsymbol{v}\|_2^2$。我们断言，$\|\boldsymbol{u}\|_2^2-\|\boldsymbol{v}\|_2^2=\sum_{i,j}A_{i,j}(u_i-v_j)^2$，其中 i,j 取遍 1 到 n 之间的所有值。事实上，

424

$$\begin{aligned}\sum_{i,j}A_{i,j}(u_i-v_j)^2&=\sum_{i,j}A_{i,j}u_i^2-2\sum_{i,j}A_{i,j}u_iv_j+\sum_{i,j}A_{i,j}v_j^2\\&=\|\boldsymbol{u}\|_2^2-2\langle A\boldsymbol{u},\boldsymbol{v}\rangle+\|\boldsymbol{v}\|_2^2=\|\boldsymbol{u}\|_2^2-2\|\boldsymbol{v}\|_2^2+\|\boldsymbol{v}\|_2^2=\|\boldsymbol{u}\|_2^2-\|\boldsymbol{v}\|_2^2\end{aligned}$$

上述等式是由于 A 的各行、各列元素之和都等于 1，并且 $\|\boldsymbol{v}\|_2^2=\langle\boldsymbol{v},\boldsymbol{v}\rangle=\langle A\boldsymbol{u},\boldsymbol{v}\rangle=\sum_{i,j}Au_iv_j$。

于是，只需证明 $\sum_{i,j}A_{i,j}(u_i-v_j)^2\geqslant\varepsilon$。由于 $\boldsymbol{u}$ 是所有分量之和等于 0 的单位向量，因此必然存在顶点 i，j 使得 $u_i>0$，$u_j<0$ 并且至少其中一个分量的绝对值 $\geqslant\frac{1}{\sqrt{n}}$，这意味着 $u_i-u_j\geqslant\frac{1}{\sqrt{n}}$。而且，由于 G 是连通图，故 G 中必然存在从 i 到 j 的至多含有 $D+1$ 个顶点的路径，其中 D 是 G 的直径[⊖]。通过对顶点重命名，我们可以假设 $i=1$，$j=D+1$，而顶点 2，3，…，D 依次是 1 到 $D+1$ 之间的路径顺序经过的顶点。现在，我们有

$$\begin{aligned}\frac{1}{\sqrt{n}}\leqslant u_1-u_{D+1}&=(u_1-v_1)+(v_1-u_2)+\cdots+(v_D-u_{D+1})\\&\leqslant|u_1-v_1|+|v_1-u_2|+\cdots+|v_D-u_{D+1}|\\&\leqslant\sqrt{(u_1-v_1)^2+(v_1-u_2)^2+\cdots+(v_D-u_{D+1})^2}\ \sqrt{2D+1}\end{aligned}\tag{21.2}$$

其中最后一个不等式是根据论断 21.1 将向量 $(u_1-v_1,\ v_1-u_2,\ \cdots,\ v_D-u_{D+1})$ 的 L_1-范数替换为 L_2-范数后得到的。令 d 是正则图 G 的顶点度，则(21.2)式表明，

$$\sum_{i,j}A_{i,j}(u_i-v_j)^2\geqslant\frac{1}{dn(2D+1)}\tag{21.3}$$

因为(21.3)式左端是非负项之和，而(21.2)式意味着形如 $A_{i,i}(u_i-v_i)^2$ 和 $A_{i,i+1}(v_i-u_{i+1})^2$（其中 $i=1,2,\cdots,D$）的所有项至少对求和过程贡献 $\frac{1}{dn(2D+1)}$（注意，由于 $A_{i,i}$ 和 $A_{i,i+1}$ 分别表示刻画了顶点 i 上的自环数和介于 i 与 $i+1$ 之间的边数，故二者都大于等于 $1/d$）。

将直径的平凡上界 $D\leqslant n-1$ 代入(21.3)式，即可证得 $\lambda(G)\leqslant 1-\frac{1}{4dn^2}$。要证明引理所述的结论，需要利用如下事实：任意 d-正则的 n-顶点图必然满足 $D\leqslant 3n/(d+1)$。我们将这一事实的证明留作习题 21.4。■

引理 21.4 的证明可以进一步增强以证明类似的结论在任意连通的非二分图上也成立

⊖ 图 G 的直径指的是任意两个顶点之间距离的最大值，其中两个顶点间的距离指的是这两个顶点之间的最短路径的长度。注意，连通 n-顶点图的直径不会超过 $n-1$。

（连通的非二分图这一条件排除了仅在各个顶点上包含自环的图）。注意，非二分图这一条件是必需的，事实上，如果 A 是二分图上的随机游走，则我们可以找到一个向量 v 使得 $Av=-v$（参见习题 21.3）。

425

21.1.2　无向连通性问题的随机算法的分析

引理 21.3 和引理 21.4 组合在一起意味着，至少在正则图上，如果顶点 s 和顶点 t 是连通的，则从 s 出发进行足够长的随机游走在多项式时间内访问过 t 的概率必然很高。

论断 21.5　设 G 是每个顶点上均具有自环的 d -正则 n -顶点图，s 是 G 的一个顶点，并且 $\ell>24n^3\log n$。如果 X_ℓ 表示从 s 出发随机游走 ℓ 步之后达到的顶点所服从的分布，则对于任意与 s 连通的顶点 t 必有 $\Pr[X_\ell=t]>\frac{1}{2n}$。

证明　根据引理 21.3 和引理 21.4，如果我们只考虑 n -顶点图 G 中 s 所在的连通分支，则对于这个连通分支上的任意概率分布 $\boldsymbol{p}$ 和 $\ell>24n^3\log n$，必有 $\|A^\ell\boldsymbol{p}-\mathbf{1}\|_2<\left(1-\frac{1}{12n^3}\right)^{24n^3\log n}<\frac{1}{n^2}$，其中 $\mathbf{1}$ 是该连通分支上的均匀分布。但这又意味着，对每个坐标分量的标号 i，有 $|[A^\ell\boldsymbol{p}-\mathbf{1}]_i|<\frac{1}{n^2}$。进而，该连通分支中每个顶点在分布 $A^\ell\boldsymbol{p}$ 中的概率至少是 $1/n-1/n^2\geqslant 1/(2n)$。 ■

注意，推论 21.5 意味着，如果将 $24n^3\log n$ 步的随机游走再重复 $O(n\log n)$ 遍（或者等价地，从 s 出发进行 $100n^3\log^2 n$ 步随机游走），则我们将高概率地访问过与 s 连通的所有顶点。

21.2　扩张图

扩张图是极其有用的组合对象，本书中几次用到了它。扩张图有两种等价的定义方式。从较高的层次上看，扩张图的两种等价的定义如下。

组合定义：度为常数的正则图 G 称为一个*扩张图*，如果对于顶点数少于 G 中顶点总数的一半的任意顶点子集 S，与 S 关联的所有边中将有常数比例的边介于 S 和 S 在 G 中的补集之间，参见图 21-1。

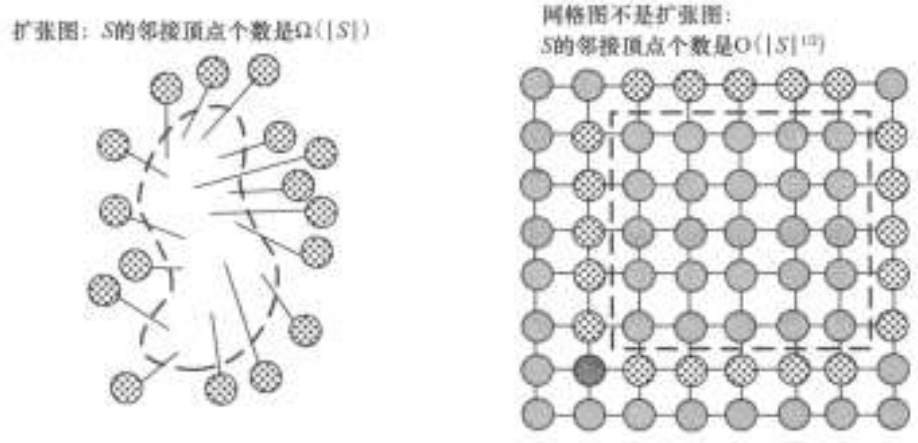


图 21-1　在边扩张图中，任意不太大的顶点子集 S 至少通过 $\Omega(|S|)$ 条边与 S 之外的顶点相邻。网格图（以及任意的平面图）都不是扩张图，因为网格图的 $k\times k$ 区域中的所有顶点在该区域之外只存在 $O(k)$ 个相邻顶点

426

代数定义：度为常数的正则图 G 称为一个扩张图，如果参数 $\lambda(G)$ 与 1 之间的距离超过一个常数，亦即，$\lambda(G)\leqslant 1-\varepsilon$ 对某个常数 $\varepsilon>0$ 成立。

常数的含义

常数指的是独立于图的顶点数的数值。我们经常会讨论无穷图族中的图，此时，常数对整个图族而言都是相同的，它与图族中的顶点数无关。下面，我们精确地给出扩张图的两种定义，并证明它们的等价性。

21.2.1　代数定义

扩张图的代数定义如下。

定义 21.6　（(n,d,λ)-扩张图）　如果 G 是一个 d -正则的 n -顶点图并且 $\lambda(G)\leqslant\lambda$ 对某个数 $\lambda<1$ 成立，则我们称 G 是一个 (n,d,λ)-扩张图。

如果存在常数 $d\in\mathbf{N}$ 和常数 $\lambda<1$ 使得 G_n 是 (n,d,λ)-扩张图对图族 $\{G_n\}_{n\in\mathbf{N}}$ 中的任意 G_n 都成立，则称 $\{G_n\}_{n\in\mathbf{N}}$ 是一个扩张图族。

有些教科书将 (n,d,λ)-扩张图简称为 (n,d,λ)-图。前面已经提到，有些教科书使用未作归一化的邻接矩阵，此时 λ 介于 0 和 d 之间。d -正则 n -顶点图能够取到的最小 λ 值是 $(1-o(1))\dfrac{2\sqrt{d-1}}{d}$，其中 $o(1)$ 表示顶点数增大时趋于 0 的一个函数。上述表达式称为阿龙-波普潘纳界（Alon-Boppana Bound），而参数 $\lambda(G)$ 达到该下界的图 G 则称为拉马努詹图（Ramanujan Graph），参见习题 21.9 和 21.10。

显式构造

在 21.2.2 节中我们将看到，利用概率方法不难证明扩张图族的存在性。然而，这并不能得到扩张图族的显式构造，但很多应用却需要这种显式构造。扩张图族 $\{G_n\}_{n\in\mathbf{N}}$ 称为显式的，如果存在一个多项式时间算法能够在输入 1^n 上输出 G_n 的邻接矩阵（或等价地，输出 G_n 的随机游走矩阵）。扩张图族 $\{G_n\}_{n\in\mathbf{N}}$ 称为强显式的，如果存在一个多项式时间算法在输入 (n,v,i)（其中 $v\in[n]$ 且 $i\in[d]$）上输出顶点 v 的第 i 个相邻顶点（的编号）。注意，在强显式的情况下，算法的输入长度和输出长度都是 $O(\log n)$，因此算法的时间复杂度实际上是 polylog(n)。

幸运的是，人们已经找到了扩张图的几种显式构造和强显式构造，其中一些构造简单而高效。但对构造过程的分析却是不平凡的，需要用到一些深奥的数学知识[⊖]。21.3 节将给出扩张图的一种强显式构造，而它的分析只用到了初等方法。21.4 节将利用这种构造
427 作为工具来对 UPATH 问题的随机游走算法进行去随机化。

21.2.2　组合扩张和扩张图的存在性

现在，我们再给出扩张图的一种组合定义，它大致上等价于定义 21.6。这种组合定义的优点在于：利用概率方法容易证明非显式扩张图族的存在性。这种定义对某些应用而言也非常有用。

⊖ 例如，下面给出了扩张图的一种简单的显式构造：所有顶点是介于 0 到 $p-1$ 之间的数，其中 p 是一个素数，每个数 x 有三条边，分别连接 $x+1$，$x-1$ 和 x^{-1}（其中的运算是相对于模 p 而言的，并且 $0^{-1}=0$）。这种显式构造的分析将用到一些深奥的数学知识（如 Selberg 3/16 定理），参见[HLW06]的第 11.1.2 节。

定义 21.7（组合(边)扩张图）　d-正则 n-顶点图 $G=(V, E)$ 称为一个 (n, d, ρ)-组合边扩张图，如果在满足 $|S|\leqslant n/2$ 的任意顶点子集 S 上均有

$$|E(S,\overline{S})|\geqslant \rho d|S|$$

其中 $\overline{S}$ 表示 S 的顶点补集，并且对于任意顶点子集 S、T 而言 $E(S, T)$ 表示满足 $i\in S$，$j\in T$ 的所有边 $\overline{ij}$ 构成的集合。

注意，在组合定义下，ρ 越大则扩张图越好。我们通常省略前缀"组合"，而将所定义的术语简写为"边扩张图"。当 ρ 是独立于 n 的正常数时，我们还用更宽松的术语"扩张图"来表示 (n, d, ρ)-组合边扩张图。利用概率方法可以证明下面的定理。(习题 21.11 要求读者证明定理的稍弱的形式。)

定理 21.8（扩张图的存在性）　设 $\varepsilon>0$ 是一个常数。则存在 $d=d(\varepsilon)$ 和 $N\in\mathbf{N}$ 使得：对于任意 $n>N$，均存在 $(n, d, 1/2-\varepsilon)$-边扩张图。

当 $\rho>1/2$ 时不存在 (n, d, ρ)-边扩张图(习题 21.13)，从这种意义上讲，定理 21.8 是紧的。下面的定理建立了组合扩张图和定义 21.6 之间的联系。

定理 21.9（组合扩张度与代数扩张度[㊀]）

1. 如果 G 是一个 (n, d, λ)-扩张图，则存在 $(n, d, (1-\lambda)/2)$-边扩张图。

2. 如果 G 是一个 (n, d, ρ)-边扩张图，则它的第二大特征值(求取绝对值)至多是 $1-\frac{\rho^2}{2}$。而且，如果 G 在所有顶点上都有自环，则它也是一个 $(n, d, 1-\varepsilon)$-扩张图，其中 $\varepsilon=\min\left\{\frac{2}{d}, \frac{\rho^2}{2}\right\}$。

定理 21.9 中"G 在所有顶点上都有自环"这一条件再次用来排除 G 是二分图的情况。二分图可以是很好的边扩张图，但是它有一个特征值等于 -1，进而二分图的谱鸿沟等于 0。

21.2.3　代数扩张图蕴含组合扩张图

定理 21.9 的第一部分可以立刻由如下的引理证得。　428

引理 21.10　如果 G 是一个 (n, d, λ)-扩张图，并且 G 的顶点子集 S 和 T 互补，则

$$|E(S,T)|\geqslant(1-\lambda)\frac{d|S||T|}{|S|+|T|}$$

证明　令 $x\in\mathbf{R}^n$ 表示如下的向量：

$$x_i=\begin{cases}+|T| & i\in S\\ -|S| & i\in T\end{cases}$$

注意，$\|x\|_1^2=|S||T|^2+|T||S|^2=|S||T|(|S|+|T|)$ 并且 $x\perp\mathbf{1}$。

令 $Z=\sum_{i,j}A_{i,j}(x_i-x_j)^2$。一方面，$Z=\frac{2}{d}|S||T|(|S|+|T|)^2$，因为满足 $i\in S$，$j\in T$ 的任意边 $\overline{ij}$ 在求和过程中出现了两次，每次为总和贡献的值是 $\frac{1}{d}(|S|+|T|)^2$。另一方面，

$$Z=\sum_{i,j}A_{i,j}x_i^2-2\sum_{i,j}A_{i,j}x_ix_j+\sum_{i,j}A_{i,j}x_j^2=2\|x\|_2^2-2\langle x,Ax\rangle$$

(上式利用了 A 的各行各列元素之和等于 1)。由于 $x\perp\mathbf{1}$ 且 $\|Ax\|_2\leqslant\lambda\|x\|_2$，故我们得到

㊀ 原书未明确定义"扩张度"，但后文将多次使用这一术语。对于代数扩张图而言，扩张度指的是定义 21.6 中的 $\lambda(G)$；对于组合边扩张图而言，扩张度指的是定义 21.7 中的 ρ。——译者注

$$\frac{1}{d}|E(S,T)|(|S|+|T|)^2 \geqslant (1-\lambda)\|x\|_2^2$$

将$\|x\|_2^2=|S||T|(|S|+|T|)$代入上式就完成了引理的证明。 ■

在代数扩张图中，我们还可以估计介于任意两个不太大的顶点子集 S 和 T 之间的边的条数，即使它们不是互不相交的。

引理 21.11（扩张图平稳引理(Expander Mixing Lemma)）　如果 $G=(V, E)$是一个(n, d, λ)-扩张图，且 $S, T\subseteq V$，则

$$\left||E(S,T)|-\frac{d}{n}|S||T|\right|\leqslant \lambda d\sqrt{|S||T|}$$

扩张图平稳引理很好地解释了为什么扩张图是"伪随机的"。在随机的 d -正则图中，$|E(S, T)|$的数学期望约等于$\frac{d}{n}|S||T|$。平稳引理断言，在扩张图中，当 S, T 都充分大时，$|E(S, T)|$将非常接近于上述数学期望。我们将引理 21.11 的证明留作习题 21.14。

21.2.4　组合扩张图蕴含代数扩张图

现在，我们证明定理 21.9 的第二部分。设 d -正则的 n -顶点图 $G=(V, E)$上对于满足$|S|\leqslant n/2$ 的任意顶点子集 S 均存在$\rho d|S|$条边介于 S 和$\overline{S}=V\setminus S$之间，并且 A 是 G 的随机游走矩阵。

设 λ 是 A 的第二大特征值(未取绝对值)。我们需要证明$\lambda\leqslant 1-\rho^2/2$。由特征值的定义
429 可知，存在向量 $u\perp \mathbf{1}$ 使得 $Au=\lambda u$。记 $u=v+w$，其中 v 在 u 取正值的分量上等于 u 相应的分量值，在其他分量上取值为 0；w 在 u 取负值的分量上等于 u 相应的分量值，在其他分量上取值为 0。由于 $u\perp \mathbf{1}$，故 v，w 都不是零向量。假设 v 至多有 $n/2$ 个分量取非零值(否则，用$-u$ 代替 u 就可以满足该假设)。令 $Z=\sum_{i,j}A_{i,j}|v_i^2-v_j^2|$。定理 21.9 的第二部分(除"而且"部分之外)可立刻由下面的两个论断得出。

论断 1　$Z\geqslant 2\rho\|v\|_2^2$。

论断 2　$Z\leqslant\sqrt{8(1-\lambda)}\|v\|_2^2$。

论断 1 的证明　将 v 的所有分量排序使得$v_1\geqslant v_2\geqslant\cdots\geqslant v_n$(当 $i>n/2$ 后，$v_i=0$)。然后，利用 $v_j^2-v_i^2=\sum_{k=i}^{j-1}(v_k^2-v_{k+1}^2)$，得到

$$Z=\sum_{i,j}A_{i,j}|v_i^2-v_j^2|=2\sum_{i<j}A_{i,j}\sum_{k=i}^{j-1}(v_k^2-v_{k+1}^2)$$

注意，对于满足 $i\leqslant k<j$ 的每条边 $\overline{ij}$，求和项 $v_k^2-v_{k+1}^2$都要在上述和式中出现一次(其权值等于 $2/d$)。由于当 $k>n/2$ 时有 $v_k=0$，故由 G 的扩张度可知，上式意味着

$$Z=\frac{2}{d}\sum_{k=1}^{n/2}|E(\{1,\cdots,k\},\{k+1,\cdots,n\})|(v_k^2-v_{k+1}^2)\geqslant\frac{2}{d}\sum_{k=1}^{n/2}\rho k(v_k^2-v_{k+1}^2)$$

但是，重新排列求和项的顺序(并利用 $k>n/2$ 时 $v_k=0$)可知，上式的最后一个求和过程的结果是

$$\frac{2}{d}d\rho\sum_{k=1}^{n/2}kv_k^2-(k-1)v_k^2=2\sum_{k=1}^{n}v_k^2=2\rho\|v\|_2^2$$ ■

论断 2 的证明　由于 $Au=\lambda u$ 且$\langle v, w\rangle=0$，故

$$\langle Av, v\rangle+\langle Aw, v\rangle=\langle A(v+w), v\rangle=\langle Au, v\rangle=\langle\lambda(v+w), v\rangle=\lambda\|v\|_2^2$$

由于$\langle Aw, v\rangle$不是正数，故$\langle Av, v\rangle/\|v\|_2^2\geqslant\lambda$，这意味着

$$(1-\lambda)\geqslant 1-\frac{\langle A\boldsymbol{v},\boldsymbol{v}\rangle}{\|\boldsymbol{v}\|_2^2}=\frac{\|\boldsymbol{v}\|_2^2-\langle A\boldsymbol{v},\boldsymbol{v}\rangle}{\|\boldsymbol{v}\|_2^2}=\frac{\sum_{i,j}A_{i,j}(\boldsymbol{v}_i-\boldsymbol{v}_j)^2}{2\|\boldsymbol{v}\|_2^2} \tag{21.4}$$

其中最后一个等式是由于 $\sum_{i,j}A_{i,j}(\boldsymbol{v}_i-\boldsymbol{v}_j)^2=\sum_{i,j}A_{i,j}\boldsymbol{v}_i^2-2\sum_{i,j}A_{i,j}\boldsymbol{v}_i\boldsymbol{v}_j+\sum_{i,j}A_{i,j}\boldsymbol{v}_j^2=2\|\boldsymbol{v}\|_2^2-2\langle A\boldsymbol{v},\boldsymbol{v}\rangle$，这里用到 A 的各行、各列元素之和都等于 1。 430

将(21.4)式最后一项的分子和分母同时乘以 $\sum_{i,j}A_{i,j}(\boldsymbol{v}_i+\boldsymbol{v}_j)^2$，则新的分子变为

$$\Big(\sum_{i,j}A_{i,j}(\boldsymbol{v}_i-\boldsymbol{v}_j)^2\Big)\Big(\sum_{i,j}A_{i,j}(\boldsymbol{v}_i+\boldsymbol{v}_j)^2\Big)\geqslant\Big(\sum_{i,j}A_{i,j}(\boldsymbol{v}_i-\boldsymbol{v}_j)(\boldsymbol{v}_i+\boldsymbol{v}_j)\Big)^2$$

上式利用了柯西-西瓦兹不等式[⊖]。再利用恒等式 $(a-b)(a+b)=a^2-b^2$，得到

$$(1-\lambda)\geqslant\frac{\Big(\sum_{i,j}A_{i,j}(\boldsymbol{v}_i^2-\boldsymbol{v}_j^2)\Big)^2}{2\|\boldsymbol{v}\|_2^2\sum_{i,j}A_{i,j}(\boldsymbol{v}_i+\boldsymbol{v}_j)^2}=\frac{Z^2}{2\|\boldsymbol{v}\|_2^2\Big(\sum_{i,j}A_{i,j}\boldsymbol{v}_i^2+2\sum_{i,j}A_{i,j}\boldsymbol{v}_i\boldsymbol{v}_j+\sum_{i,j}A_{i,j}\boldsymbol{v}_j^2\Big)}$$

$$=\frac{Z^2}{2\|\boldsymbol{v}\|_2^2(2\|\boldsymbol{v}\|_2^2+2\langle A\boldsymbol{v},\boldsymbol{v}\rangle)}\geqslant\frac{Z^2}{8\|\boldsymbol{v}\|_2^4}$$

最后一个不等式如下得到：由 A 是对称随机矩阵可知，$\|A\boldsymbol{v}\|_2\leqslant\|\boldsymbol{v}\|_2$ 对任意 $\boldsymbol{v}$ 成立，这意味着 $\langle A\boldsymbol{v},\boldsymbol{v}\rangle\leqslant\|\boldsymbol{v}\|_2^2$。

对于"而且"部分，只需注意到，在 $d-1$ -正则图的所有顶点上添加一个自环等价于将它的随机游走矩阵 A 变换为矩阵 $\frac{d-1}{d}A+\frac{1}{d}I$，其中 I 是单位矩阵。由于 A 的最小特征值（未取绝对值）至少是 -1，因此新矩阵的最小特征值至少是 $-\frac{d-1}{d}+\frac{1}{d}=-1+\frac{2}{d}$。 ■

21.2.5　用扩张图设计纠错码

在构造扩张图之前，我们先看看扩张图在随机算法上的一个应用。回顾一下，在 7.4.1 节，我们曾通过将概率算法重复运行 k 次后再取过半数的答案来把算法的错误概率从 1/3 减小到 $2^{-\Omega(k)}$。如果概率算法使用了 m 个随机位，则 k 次重复运行将使用 $m\cdot k$ 个随机位。似乎很难构想出一种使用较少随机位就能实现概率算法重复调用的方法。然而，利用扩张图，我们能够仅用 $m+O(k)$ 个随机位就把概率算法的错误概率降低到同样的水平。

思想很简单：从一个强显式扩张图族中取一个 $(M=2^m,\ d,\ 1/10)$-扩张图 G，其中 d 是一个常数。（注意，利用图的乘幂方法可以将任意扩张图变换为参数 $\lambda<1/10$ 的扩张图，参见 21.3 节。）从 G 中随机选取一个顶点 v_1，然后从 v_1 出发在 G 上进行 $k-1$ 步随机游走，将访问过的顶点依次记为 v_2，…，v_k（注意，在顶点的邻域中随机选择一个顶点需要 $O(\log d)=O(1)$ 个随机位）。将 v_1，…，v_k 分别视为随机位串，亦即，将 G 的顶点集 $[M]$ 等同于概率算法的可选随机位串集 $\{0,1\}^m$。这样利用 v_1，…，v_k 调用概率算法 k 遍并输出过半数的答案。

为简单计，我们这里只分析概率算法仅犯单面错误的情况。例如，考虑这样的 **RP** 算法：如果给定的输入不属于相应的语言，则算法绝不可能"接受"；如果给定的输入属于相 431

⊖ 柯西-西瓦兹不等式(Cauchy-Schwarz Inequality)断言，$\langle\boldsymbol{x},\boldsymbol{y}\rangle\leqslant\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2$ 对于任意向量 $\boldsymbol{x}$，$\boldsymbol{y}$ 均成立。这里，我们使用双重下标 (i,j)，并且 $\boldsymbol{x}_{i,j}=\sqrt{A_{i,j}}(\boldsymbol{v}_i^2-\boldsymbol{v}_j^2)$ 而 $\boldsymbol{y}_{i,j}=\sqrt{A_{i,j}}(\boldsymbol{v}_i^2+\boldsymbol{v}_j^2)$。

应的语言，则算法将以 2/3 的概率"接受"。(可以类似地要求 **coRP** 算法。)对于这样的概率算法，如果在某一个 v_i 上接受某个输入，则算法的重复调用过程也接受该输入；如果给定的输入不属于相应的语言，则算法的重复调用过程永远也不会接受该输入。如果输入属于语言，我们将导致算法拒绝该输入的随机位串称为"劣性"的，用$\mathcal{B}\subseteq[M]$表示该输入的所有"劣性"随机位串构成的集合。我们知道$|\mathcal{B}|\leqslant\frac{M}{3}$。将$\beta=1/3$和$\lambda=1/10$代入下面的引理即可知道，上面的概率算法重复调用过程至多以$2^{-\Omega(k)}$的概率拒绝属于语言的输入。

定理 21.12　(扩张图的随机游走)　设 G 是一个(n, d, λ)-扩张图，$\mathcal{B}\subseteq[n]$使得$|\mathcal{B}|\leqslant\beta n$对某个$\beta\in(0, 1)$成立。如果随机变量 $X_1, \cdots, X_k$ 表示 G 上起始于 X_1 的 $k-1$ 步随机游走并且 X_1 均匀随机地取自$[n]$，则

$$\Pr[\forall_{1\leqslant i\leqslant k}X_i\in\mathcal{B}]\leqslant((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}$$

注意，如果 λ 和 β 都是小于 1 的常数，则$(1-\lambda)\sqrt{\beta}+\lambda$ 也小于 1。

证明　对于$1\leqslant i\leqslant k$，令 B_i 表示随机事件$X_i\in\mathcal{B}$。这样，定理 21.12 实际上要求给出如下概率的上界

$$\Pr[\wedge_{i=1}^{k}B_i]=\Pr[B_1]\Pr[B_2\mid B_1]\cdots\Pr[B_k\mid B_1,\cdots,B_{k-1}]\tag{21.5}$$

令 B 是从 $\mathbf{R}^n$ 到 $\mathbf{R}^n$ 的线性变换，它在变换过程中将不位于$\mathcal{B}$的坐标分量置为 0。也就是说，对于任意$i\in[n]$，如果$i\in\mathcal{B}$则$(B\boldsymbol{u})_i=\boldsymbol{u}_i$；否则，$(B\boldsymbol{u})_i=0$。不难验证，对于$[n]$上的任意概率分布 $\boldsymbol{p}$，向量 $B\boldsymbol{p}$ 的所有坐标分量之和等于根据 $\boldsymbol{p}$ 随机选择的顶点属于$\mathcal{B}$的概率。而且，如果我们将 $B\boldsymbol{p}$ 归一化，则所得的概率向量是在已知随机选择的顶点属于$\mathcal{B}$的条件下 $\boldsymbol{p}$ 的条件概率分布。

因此，如果令 $\mathbf{1}=(1/n, \cdots, 1/n)$表示$[n]$上的均匀分布，$\boldsymbol{p}^i\in\mathbf{R}^n$ 表示在 $B_1, \cdots, B_i$ 发生的条件下 X_i 的概率分布，则

$$\boldsymbol{p}^1=\frac{1}{\Pr[B_1]}B\mathbf{1}$$

$$\boldsymbol{p}^2=\frac{1}{\Pr[B_2\mid B_1]\Pr[B_1]}BAB\mathbf{1}$$

一般地，

$$\boldsymbol{p}^i=\frac{1}{\Pr[B_i\mid B_{i-1}\cdots B_1]\cdots\Pr[B_1]}(BA)^{i-1}B\mathbf{1}$$

由于任意概率向量 $\boldsymbol{p}$ 均满足$|\boldsymbol{p}|_1=1$，所以(21.5)式左端的概率等于

432

$$|(BA)^{k-1}B\mathbf{1}|_1\tag{21.6}$$

运用 L_1-范数和 L_2-范数之间的关系(论断 21.1)，则只要证明了下面的(21.7)式就得到了(21.6)的上界。

$$\|(BA)^{k-1}B\mathbf{1}\|_2\leqslant\frac{((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}}{\sqrt{n}}\tag{21.7}$$

要证明(21.7)式，我们需要如下的定义和引理。

定义 21.13　(谱范数)　在任意矩阵 A 上，A 的谱范数，记为$\|A\|$，定义为$\|A\boldsymbol{v}\|_2$在满足$\|\boldsymbol{v}\|_2=1$的所有 $\boldsymbol{v}$ 上达到的最大值。

习题 21.5 和习题 21.6 要求读者证明任意随机游走矩阵的谱范数都等于 1，并且$\|A+B\|\leqslant\|A\|+\|B\|$和$\|AB\|\leqslant\|A\|\|B\|$对$n\times n$的任意矩阵 A，B 成立。

引理 21.14　如果 A 是(n, d, λ)-扩张图 G 的随机游走矩阵，并且 J 是每个顶点上都有自环的 n-顶点团的随机游走矩阵(亦即，$J_{i,j}=1/n$ 对任意 i，j 成立)，则

$$A=(1-\lambda)J+\lambda C \tag{21.8}$$

其中$\|C\|\leqslant 1$。

注意，对于任意概率向量 $\boldsymbol{p}$，$J\boldsymbol{p}$ 都是均匀分布。因此，引理 21.14 告诉我们，从某种意义下讲，我们可以将(n, d, λ)-扩张图上的单步随机游走视为以$(1-\lambda)$的概率按均匀分布进行游走，而以 λ 的概率按其他分布进行游走。当然，这种看法肯定是不准确的，因为 d-正则图上的单步随机游走只能达到当前顶点的 d 个相邻顶点之一。但是，就我们的分析目的而言，条件(21.8)式已经够用了[⊖]。

引理 21.14 的证明　事实上，直接定义 $C=\frac{1}{\lambda}(A-(1-\lambda)J)$。我们需要证明$\|C\boldsymbol{v}\|_2\leqslant\|\boldsymbol{v}\|_2$ 对任意 $\boldsymbol{v}$ 成立。将 $\boldsymbol{v}$ 分解为 $\boldsymbol{v}=\boldsymbol{u}+\boldsymbol{w}$，其中 $\boldsymbol{u}=\alpha\mathbf{1}$ 且 $\boldsymbol{w}\perp\mathbf{1}$，$\alpha\in\mathbf{R}$。由 $A\mathbf{1}=\mathbf{1}$ 且 $J\mathbf{1}=\mathbf{1}$可得，$C\boldsymbol{u}=\frac{1}{\lambda}(\boldsymbol{u}-(1-\lambda)\boldsymbol{u})=\boldsymbol{u}$。现在，令 $\boldsymbol{w}'=A\boldsymbol{w}$。则$\|\boldsymbol{w}'\|_2\leqslant\lambda\|\boldsymbol{w}\|_2$ 且(正如我们在引理 21.3 中所见一样)$\boldsymbol{w}'\perp\mathbf{1}$。换句话说，$\boldsymbol{w}$ 的所有分量之和等于 0，这意味着 $J\boldsymbol{w}=0$。由此可得，$C\boldsymbol{w}=\frac{1}{\lambda}\boldsymbol{w}'$。由于 $\boldsymbol{w}'\perp\boldsymbol{u}$，故$\|C\boldsymbol{v}\|_2^2=\left\|\boldsymbol{u}+\frac{1}{\lambda}\boldsymbol{w}'\right\|_2^2=\|\boldsymbol{u}\|_2^2+\left\|\frac{1}{\lambda}\boldsymbol{w}'\right\|_2^2\leqslant\|\boldsymbol{u}\|_2^2+\|\boldsymbol{w}\|_2^2=\|\boldsymbol{v}\|_2^2$，其中两次运用勾股定理由 $\boldsymbol{u}\perp\boldsymbol{w}$ 得到$\|\boldsymbol{u}+\boldsymbol{w}\|_2^2=\|\boldsymbol{u}\|_2^2+\|\boldsymbol{w}\|_2^2$。■

回到定理 21.12 的证明上来。我们可以记 $BA=B((1-\lambda)J+\lambda C)$，因而$\|BA\|\leqslant(1-\lambda)\|BJ\|+\lambda\|BC\|$。由于 J 的输出总是形如 $\alpha\mathbf{1}$ 的向量，故不难验证$\|B\mathbf{1}\|_2=\sqrt{\frac{\beta n}{n^2}}=\frac{\sqrt{\beta}}{\sqrt{n}}=\sqrt{\beta}\|\mathbf{1}\|_2$，$\|BJ\|=\sqrt{\beta}$。同时，因为 B 只是将输入向量的某些分量变为 0，故$\|B\|\leqslant 1$，这意味着$\|BC\|\leqslant 1$。因此，$\|BA\|\leqslant(1-\lambda)\sqrt{\beta}+\lambda$。这意味着$\|(BA)^{k-1}B\mathbf{1}\|_2\leqslant((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}\frac{\sqrt{\beta}}{\sqrt{n}}$，由此得到(21.7)式。■

下面的定理成功地削减了犯双面错误的概率算法的错误概率，我们略去该定理的证明(但习题 21.12 讨论了该定理的证明)。

433

定理 21.15（扩张图的切尔诺夫界）　设 G 是一个(n, d, λ)-扩张图，$B\subseteq[n]$使得$|B|=\beta n$。如果随机变量 X_1，…，X_k 表示 G 上起始于 X_1 的 $k-1$ 步随机游走并且 X_i 均匀随机地取自$[n]$。对于任意 $i\in[k]$，如果 $X_i\in B$ 则定义 $B_i=1$；否则，定义 $B_i=0$。那么，对于 $\delta>0$，有

$$\Pr\left[\left|\frac{\sum_{i=1}^{k}B_i}{k}-\beta\right|>\delta\right]\leqslant 2\mathrm{e}^{-(1-\lambda)\delta^2k/4}$$

21.3　扩张图的显式构造

本节构造一个非常显式的扩张图族。我们要使用的主要工具是图的几种乘积。图的乘积操作将两个输入图 G，G' 变换为另一个图 H。通常，我们感兴趣的是 G，G' 的性质和 H 的性质之间的关系。具体地讲，本节主要关注图的三个参数——顶点数(表示为 n)，顶点

⊖　用代数的观点看，(21.8)式并不是说在单步随机游走中都以 $1-\lambda$ 的概率按均匀分布游走，这是因为 C 未必是一个随机矩阵，它可能含有负值元素。

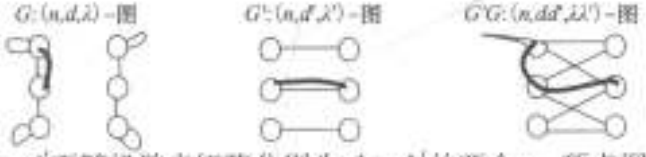
的度(表示为 d)和随机游走矩阵的第二大特征值(表示为 λ)，讨论它们在各种图乘积操作下的变化。然后，我们利用这几种乘积构造一个显式的强扩张图族。在下一节还要利用这几种乘积为无向连通性问题设计一个对数空间算法。

21.3.1　旋转映射

以前，我们总用邻接矩阵来表示图，本章则用随机游走矩阵来表示图。如果图是 d-正则的，则我们可以用旋转映射来表示它。如果 G 是一个度为 d 的 n-顶点图，则可以先为图中的每个顶点关联数字 $1, \cdots, d$，然后定义旋转映射 $\hat{G}$ 是从 $[n]\times[d]$ 到 $[n]\times[d]$ 的映射，它将序对 (v, i) 映射为序对 (u, j) 以表明 u 是 v 的第 i 个相邻顶点且 v 是 u 的第 j 个相邻顶点。显然，旋转映射是 $[n]\times[d]$ 上的一个置换(既是一对一的又是满的)。有的读者可能会问为什么不恰当地标识顶点使得 $\hat{G}(u, i)=(v, i)$(也就是说，u 是 v 的第 i 个相邻顶点且 v 也是 u 的第 i 个相邻顶点)呢？虽然事实上这是可以实现的，但它需要全局的计算。对我们的应用而言，这种方法太复杂了，因为扩张图的显式构造需要通过空间受限的计算来实现。

下面，我们描述图乘积操作，它将两个图变换为一个图。我们将为图乘积操作选用最自然的图表示方法。但是，对读者而言，在图的其他表示方法(例如，随机游走矩阵表示或旋转映射表示)下研究图乘积操作的等价实现将是十分有益的练习。

21.3.2　矩阵乘积和路径乘积



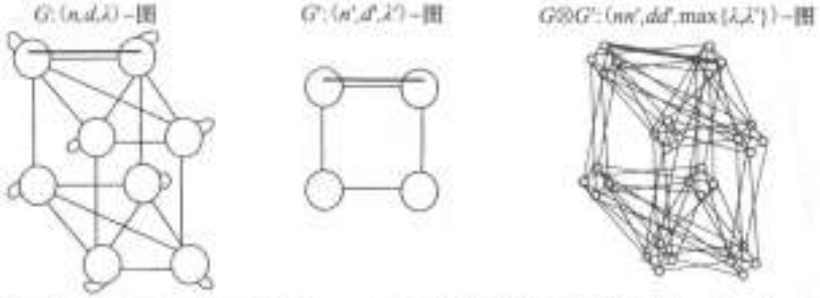
434

对于度分别为 d，d' 而随机游走矩阵分别为 A，A' 的两个 n-顶点图 G，G'，定义图 $G'G$ 的随机游走矩阵为 $A'A$。也就是说，从 u 到 v 的每条长度为 2 路径(其中路径的第一条边取自 G，而路径的第二条边取自 G')都在图 $G'G$ 中对应一条边 (u, v)。这样，图 $G'G$ 有 n 个顶点，并且每个顶点的度都等于 dd'。通常，我们感兴趣的是 $G=G'$ 的情况。此时，图的乘积操作也称为图平方。更一般地，我们用 G^k 表示图 $G\cdot G\cdot\cdots\cdot G$(重复 k 次)。我们在引理 21.3 中已经见过这种情形，类似地对 $G'G$ 进行分析可以得到下面的引理(其证明留作习题 21.8)。

引理 21.16　(矩阵相乘改进扩张度)　$\lambda(G'G)\leqslant\lambda(G')\lambda(G)$。

注意，根据 G' 和 G 的旋转映射，可以很容易计算出 $G'G$ 的旋转映射。

21.3.3　张量积



如果图 G 和图 G' 的顶点数分别为 n，n' 而每个顶点的度分别为 d，d'，并且 $\hat{G}$：$[n]\times$

$[d]\to[n]\times[d]$和$\hat{G}'$：$[n']\times[d']\to[n']\times[d']$分别是 G 和 G' 的旋转映射，则 G 和 G' 的张量积 $G\otimes G'$ 是顶点度为 dd' 的 nn'-顶点图，它的旋转映射 $\widehat{G\otimes G'}$ 是 $([n]\times[n'])\times([d]\times[d'])$ 上如下定义的置换

$$\widehat{G\otimes G'}((u,v),(i,j))=(u',v'),(i',j')$$

其中$(u',i')=\hat{G}(u,i)$而$(v',j')=\hat{G}'(v,j)$。也就是说，$G\otimes G'$ 的每个顶点是一个顶点序对，序对中的一个顶点取自 G 而另一个顶点则取自 G'。从 $G\otimes G'$ 的顶点 (u,v) 开始按 (i,j) 进行一步游走到达顶点 (u',v') 相当于在 G 和 G' 上分别独立地进行一步游走。亦即，u' 是 G 中顶点 u 的第 i 个相邻顶点，而 v' 是 G' 中顶点 v 的第 j 个相邻顶点。

用随机游走矩阵来描述张量积也非常容易。如果 $n\times n$ 的矩阵 $A=(a_{i,j})$ 是 G 的随机游走矩阵并且 $n'\times n'$ 的矩阵 $A'=(a'_{i',j'})$ 是 G' 的随机游走矩阵，则张量积 $G\otimes G'$ 的随机游走矩阵是一个 $nn'\times nn'$ 的矩阵，记为 $A\otimes A'$，它的第 (i,i') 行第 (j,j') 列的取值是 $a_{i,j}\cdot a'_{i',j'}$。也就是说，$A\otimes A'$ 含有 A' 的 n^2 个复制，其中第 (i,j) 个复制被缩放 $a_{i,j}$ 倍。

$$A\otimes A'=\begin{bmatrix} a_{1,1}A' & a_{1,2}A' & \cdots & a_{1,n}A' \\ a_{2,1}A' & a_{2,2}A' & \cdots & a_{2,n}A' \\ \vdots & \vdots & & \vdots \\ a_{n,1}A' & a_{n,2}A' & \cdots & a_{n,n}A' \end{bmatrix}$$

435

张量积还可以用图的术语描述为：图 G 的每个顶点在 $G\otimes G'$ 中对应一个包含 n' 个顶点的聚簇。在这种刻画下，如果 u 和 v 是 G 的两个相邻顶点，则 u 在 $G\otimes G'$ 中对应的聚簇和 v 在 $G\otimes G'$ 中对应的聚簇之间将存在一个二分图形式的 G'。也就是说，如果 (i,j) 是 G' 的一条边，则在 u 对应的聚簇中的第 i 个顶点和 v 对应的聚簇的第 j 个顶点之间将存在一条边。

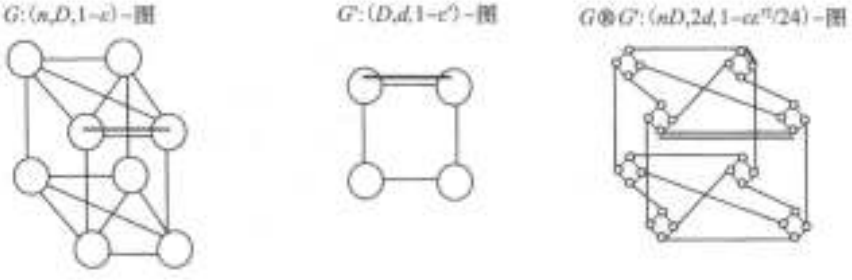
引理 21.17（张量乘积保持扩张度）　如果 $\lambda=\lambda(G)$ 且 $\lambda'=\lambda(G')$，则 $\lambda(G\otimes G')\leqslant\max\{\lambda,\lambda'\}$。

引理 21.17 给出的上界背后的直观思想如下。在 $G\otimes G'$ 上进行一步游走相当于在 G 和 G' 上分别独立地进行一步游走。因此，如果 G 和 G' 上的随机游走在 T 步内收敛到均匀分布，则 $G\otimes G'$ 上的随机游走也将在 T 步内收敛到均匀分布。

引理 21.17 的证明　定理的结论可以立刻由张量积和特征值的一些基本事实得出（参见习题 21.22）。如果 $\lambda_1,\cdots,\lambda_n$ 是 A 的所有特征值（其中 A 是 G 的随机游走矩阵），并且 $\lambda'_1,\cdots,\lambda'_{n'}$ 是 A' 的所有特征值（其中 A' 是 G' 的随机游走矩阵），则 $A\otimes A'$ 的所有特征值都是形如 $\lambda_i\lambda'_j$ 的数，因此除 1 之外 $A\otimes A'$ 的最大特征值是 $1\cdot\lambda(G')$ 或 $\lambda(G)\cdot 1$。■

注意，我们不用任何关于特征值的知识就可以证明 $\lambda(G\otimes G')\leqslant\lambda(G)+\lambda(G')$（参见习题 21.23）。这个较弱的上界就能满足我们显式构造扩张图的需求。

21.3.4　替换乘积



在矩阵乘积和张量积中，乘积图的顶点度都大于输入图的顶点度。下面定义的乘积可以让一个输入图的顶点度减小。设 G，G' 是两个图，其中 G 有 n 个顶点且顶点度等于 D，而 G' 有 D 个顶点且顶点度等于 d。G 和 G' 的平衡替换乘积(下面简称替换乘积)，记为 $G\textcircled{R}G'$，是如下得到的 $2d$ -度的 nD -顶点图：

1. 对于 G 的每个顶点 u，图 $G\textcircled{R}G'$ 中含有 G' 的一个复制(包括顶点和边)。

2. 如果 u，v 是 G 中的相邻顶点，并且 v 是 u 的第 i 个相邻顶点而 u 是 v 的第 j 个相
436 邻顶点(也可以说，从 u 出发的第 i 边关联到顶点 v 而从 v 出发的第 j 边关联到顶点 u)，则图 $G\textcircled{R}G'$ 中 u 对应的 G' 拷贝的第 i 个顶点和 v 对应的 G' 拷贝的第 j 个顶点之间添加 d 条平行边。

上面的第 2 个条件也可以要求顶点之间只引入一条边而不是 d 条平行边。事实上，有些教科书中的"替换乘积"指的就是这种变形。在替换乘积中引入平行边可以确保从顶点 v 出发在 $G\textcircled{R}G'$ 上随机游走一步时可以以 1/2 的概率停留在 v 所在的聚簇中而以 1/2 的概率离开 v 所在的聚簇。

替换乘积也可以如下简单地描述为旋转映射。由于 $G\textcircled{R}G'$ 有 nD 个顶点且顶点度为 $2d$，故它的旋转映射可以视为 $([n]\times[D])\times([d]\times\{0,1\})$ 上的一个置换。旋转映射的输入包含四个数 u，v，i，b，其中 $u\in[n]$，$v\in[D]$，$i\in[d]$ 且 $b\in\{0,1\}$。如果 $b=0$，则旋转映射输出 u，$\hat{G}'(v,i)$，b；如果 $b=1$，则它输出 $\hat{G}(u,v)$，i，b。也就是说，根据 b 等于 0 还是 1，旋转映射把输入中的 v 相应地当作 G' 的顶点或者 G 中的一条边的编号。

用随机游走矩阵的话说，替换乘积可以描述为：

$$A\textcircled{R}A' = 1/2\,\hat{A} + 1/2(I_n \otimes A') \tag{21.9}$$

其中 A，A' 分别是 G 和 G' 的随机游走矩阵，而 $\hat{A}$ 是 G 的旋转映射对应的置换矩阵。也就是说，$\hat{A}$ 是 $(nD)\times(nD)$ 的矩阵，它的第 (i,j) 列除了在 (i',j') 的位置上取 1 之外其余元素全为 0，其中 $(i',j')=\hat{G}(i,j)$。

如果 $D\gg d$，则替换乘积的顶点度将显著地小于 G 的顶点度。下面的引理表明，顶点度的大幅减小并不会严重地破坏图的扩张度。

引理 21.18（替换乘积的扩张度）　如果 $\lambda(G)\leqslant 1-\varepsilon$ 且 $\lambda(H)\leqslant 1-\delta$，则 $\lambda(G\textcircled{R}H)\leqslant 1-\frac{\varepsilon\delta^2}{24}$。

引理 21.18 背后的直观含义如下。将输入图 G 视为一个良好的扩张图，其唯一缺点是顶点度 D 太大。这意味着，在 G 上进行 k 步随机游走需要 $O(k\log D)$ 个随机二进制位。但是，正如我们在 21.2.5 中所见，有时借助扩张图可以使得所需的随机二进制位的数量减少一些。为此，一种自然的思想是利用 D 个顶点上度 $d\ll D$ 的扩张图 G' 来产生随机游走要使用的边的编号。这种直观的需求促使人们定义 $G\textcircled{R}G'$，这样，$G\textcircled{R}G'$ 上的随机游走大致等价于用扩张图 G' 的随机游走来产生 G 上随机游走所需的边的编号。具体地讲，$G\textcircled{R}G'$ 上的每步随机游走可以如下进行：先投掷一枚硬币，根据投掷结果采取两种策略，要么在 G' 上进行一步随机游走，要么将 G' 上的当前顶点当作边的编号在 G 上进行一步随机游走。习题 21.24 给出了替换乘积的另一种直观含义，它考虑的是替换乘积得出的组合(边)扩张度与输入图的边扩张度之间的函数关系。

引理 21.18 的证明　只需证明 $\lambda(G\textcircled{R}H)^2\leqslant 1-\frac{\varepsilon\delta^2}{8}$。由于 $\lambda(F^t)=\lambda(F)^t$ 对任意图 F 都成立，因而只需证明 $\lambda((G\textcircled{R}H)^3)\leqslant 1-\frac{\varepsilon\delta^2}{8}$。令 $n\times n$ 矩阵 A 是 G 的随机游走矩阵，并且 G 的旋转映射 $\tilde{G}$ 对应的置换矩阵是 $(nD)\times(nD)$ 的矩阵 $\hat{A}$。再令 $D\times D$ 的矩阵 B 是 H 的随机游走矩阵，而 C 是 $(G\textcircled{R}H)^3$ 的随机游走矩阵。则，(21.9)式意味着 137

$$C=(1/2\,\hat{A}+1/2(I_n\otimes B))^3 \tag{21.10}$$

同时，引理 21.14 还意味着 $B=(1-\delta)B'+\delta J_D$ 对某个范数至多为 1 的矩阵 B' 和元素全为 $1/D$ 的 $D\times D$ 矩阵 J_D 成立。将该等式代入(21.10)式，展开所有的项，然后将除 $\frac{1}{2}\delta(I_n\otimes J_D)\frac{1}{2}\hat{A}\frac{1}{2}\delta(I_n\otimes J_D)$ 之外的其余各项整理在一起。读者可以验证，所有这些项都对应范数至多为 1 的矩阵。因而，(21.10)式变为

$$C=\left(1-\frac{\delta^2}{8}\right)C'+\frac{\delta^2}{8}(I_n\otimes J_D)\hat{A}(I_n\otimes J_D) \tag{21.11}$$

其中 C' 是某个范数至多为 1 的 $(nD)\times(nD)$ 矩阵。引理可以由下面的论断得出。

论断：$(I_n\otimes J_D)\hat{A}(I_n\otimes J_D)=A\otimes J_D$。

证明　事实上，等式左端是如下 nD -顶点图的随机游走矩阵，该图上从顶点 (i,j) 出发的一步随机游走将涉及：(1)随机选取 $k\in[D]$；(2)令 i' 是 G 上顶点 i 的第 k 个相邻顶点；(3)在 $[D]$ 中随机选择 j'，并移动达到顶点 (i',j')。这一个过程可以等价地描述为"随机选择 G 中顶点 i 的相邻顶点 i' 并在 $[D]$ 中随机选取 j'"，而这恰好是在 $A\otimes J_D$ 表示的图上进行随机游走。■

上述论断得出了引理，因为 $\lambda(A\otimes J_D)\leqslant\max\{\lambda(A),\lambda(J_D)\}=\max\{\lambda(A),0\}$。将它代入(21.11)式，同时注意 $\lambda(C')\leqslant 1$ 对范数至多为 1 的任意矩阵都成立，就可以得出引理的结论。■

21.3.5　显式构造

本小节运用前面介绍的 3 种图乘积来显式地构造一个强扩张图族，也就是说，我们将证明下面的定理。

定理 21.19（扩张图的显式构造）　存在强显式的 (λ,d)-扩张图族，其中 $d\in\mathbf{N}$，$\lambda<1$ 是某个常数。

注意，定理 21.19 再结合矩阵乘积/路径乘积，就可以用来在任意 $\lambda>0$ 上构造强显式的 (λ,d)-扩张图族，只不过 d 可能是依赖于 λ 的任意大的常数。

证明　我们先证明较弱的结论：我们非常显式地构造一个图族 $\{G_n\}$，但 G_n 不是 k 个顶点上的图而是 c^k 个顶点上的图，其中 c 是一个常数。也就是说，我们构造的图族并没有为每个 n 包含一个 n -顶点扩张图，而是只为 c 的幂次形式的 n 包含了一个 n -顶点扩张图。然后，我们再概要地说明如何改进我们构造的图族来得到另一个图族使得它为每个 n 都包含了一个 n -顶点扩张图。438

构造过程是一个递归过程。也就是说，我们从某个规模有限的图 G_1 开始(利用蛮力搜索算法可以找到 G_1)，然后再由 G_{k-1} 来构造 G_k。从较高的层次上看，前面介绍的三种图

乘积将在构造过程中发挥不同的作用。张量积可以用来增加 G_{k-1} 的顶点个数，但这可能造成顶点度的增大并使得图的扩张度受到一些损失。替换乘积可以用来大幅度削减顶点度，但图的扩张度可能受到损失。最后，矩阵乘积/路径乘积可以用来弥补损失的扩张度，而这只造成顶点度小幅度地增大。实际的构造过程定义如下。

- 令 H 是一个 $(D=(2d)^{100}, d, 0.01)$-扩张图，它可以用蛮力搜索算法来得到。（我们取充分大的 d 以确保这样的扩张图是存在的。）我们令 G_1 是一个 $((2d)^{100}, 2d, 1/2)$-扩张图而 G_2 是一个 $((2d)^{200}, 2d, 1/2)$-扩张图。（同样，G_1，G_2 也可以用蛮力算法找到。）
- 对于 $k>2$，定义

$$G_k = (G_{\lfloor \frac{k}{2} \rfloor} \otimes G_{\lceil \frac{k}{2} \rceil})^{50} \textcircled{R} H$$

我们证明下面的论断：

论断：对任意的 k，G_k 是一个 $((2d)^{100k}, 2d, 1-1/50)$-扩张图。而且，存在一个 poly($k$)时间算法，它在 G_k 中给定的顶点编号 i 和 $[2d]$ 中给定的编号 j 上输出 G_k 上顶点 i 的第 j 个相邻顶点。

证明　我们先用归纳法证明论断的第一个部分。通过直接验证可知，它在 $k=1$，2 时成立。当 $k>2$ 时，如果用 n_k 表示 G_k 的顶点个数，则 $n_k = n_{\lfloor k/2 \rfloor} n_{\lceil k/2 \rceil} (2d)^{100}$。由归纳假设可知 $n_{\lfloor k/2 \rfloor} = (2d)^{100\lfloor k/2 \rfloor}$ 并且 $n_{\lceil k/2 \rceil} = (2d)^{100\lceil k/2 \rceil}$，再结合 $\lfloor (k-1)/2 \rfloor + \lceil (k-1)/2 \rceil = k-1$，即可得到 $n_k = (2d)^{100k}$。G_k 中任意顶点 j 的度为 $2d$ 也很容易证明。事实上，如果图 G 的顶点度为 $2d$，则 $G \otimes G$ 的顶点度为 $(2d)^2$，继而 $(G \otimes G)^{50}$ 的顶点度为 $(2d)^{100}$，进而 $(G \otimes G)^{50} \textcircled{R} H$ 的顶点度为 $(2d)$。特征值的分析也可以由归纳法得到。事实上，如果 $\lambda(G) < 1 - 1/50$，则 $\lambda((G \otimes G)^{50}) < 1/e < 1/2$；因此，由引理 21.18 可知，$\lambda((G \otimes G)^{50} \textcircled{R} H) \leqslant 1 - \frac{1}{2}(0.99)^2/24 \leqslant 1 - 1/50$。

要证明“而且”部分，只需注意到存在一个自然的算法来计算 G_k 的旋转映射，该算法需要递归调用 $G_{\lfloor k/2 \rfloor}$ 的旋转映射 100 次，因此它的运行时间约等于 $n^{\log 100}$。■

上述构造和分析过程得到了一个扩张图族，但是其中只为 c 的幂次形式的 n 包含了 n-顶点扩张图，其中 c 是常数。要完成定理的证明，只需进一步注意到，将 (n, d, λ)-扩张图中规模至多为 c 的一些顶点集合合并为一些“超级顶点”就可以把它改造成一个 (n', cd, λ')-扩张图，其中 $\lambda' < 1$ 是依赖于 λ，d 的常数并且 n' 是满足 $n/c \leqslant n' \leqslant n$ 的任意自然数。■

评注 21.20　无论从顶点度与扩张度之间的关系上讲还是从构造过程的运行时间上讲（特别地，单是初始阶段的蛮力搜索就要花费 2^{200} 步），定理 21.19 的证明过程得到的数值界限都非常差。其中的部分原因是，出于教学目的，我们只给出了这个构造过程的最简单的形式，而没有对它进行各种优化。但是，即使我们进行了这样的优化，它也不是目前最高效的扩张图显式构造方法。

439

扩张图存在各种可用于实践的非常高效的显式构造方法（例如[LPS86, Mar88]）。但是，对这些构造方法的分析通常都要用到深奥的数论知识。同时，替换乘积及其孪生操作（拉链乘积）除了用来证明定理 21.15 之外，还有很多其他应用。其中一个应用是用它来为无向连通性问题设计确定型对数空间算法，下一节将讨论这个应用。另一个应用是用它来

构造组合顶点扩张图(Combinatorial Vertex Expander)使得它在小规模顶点集上具有较好的扩张度(由参数 λ 表示)[CRVW02]，也可以参见习题 21.15。

21.4　无向连通性问题的确定型对数空间算法

本节介绍由莱茵戈尔德(Reingold)得到的一个结果。该结果表明，至少无向图中的 s-t-连通性问题 UPATH 的随机游走算法(参见第 7 章)——这个最著名的随机对数空间算法——可以被完全去随机化。

定理 21.21　(莱茵戈尔德定理(Reingold Theorem))$\mathrm{UPATH} \in \mathbf{L}$。

莱茵戈尔德给出了从 s 出发的 $\mathrm{poly}(n)$ 个游走构成的集合使得：如果 s，t 是连通的，则该集合中至少有一个游走能够到达 t。利用概率方法和推论 21.5 可以证明这种规模较小的游走集合的存在性。关键点在于，莱茵戈尔德对游走的枚举可以通过确定型算法在对数空间内完成。

证明思路

同以往一样，我们感兴趣的是含平行边的无向图，并且我们只在 $d=4$ 的情况下研究如何验证 d-正则图的连通性。这不失一般性。事实上，如果某个顶点的度 $d''<3$，则可以在该顶点上添加若干个自环使得该顶点的度达到 d；如果某个顶点的度 $d'\geqslant 3$，则可以将该顶点替换为一个由 d' 个顶点构成的环，原来与该顶点关联的 d' 条边可以分别关联到环上的一个顶点上。当然，对数空间的图灵机没有足够的空间来存储修改后的图，但是它可以假装图已经按照上述方式进行了修改。因为当它要访问图的时候，它可以随时根据需要执行相应的修改。(用定义 4.16 中的正式术语来讲，图的上述修改是隐式可计算的。)事实上，后面的证明过程中还要在图上执行一系列的其他局部修改，其中每个修改都可以用对数空间算法根据需要而随时执行。

首先，我们观察到查验连通性在扩张图中是很容易进行的。具体地讲，如果图 G 的所有连通分支都是扩张图，则存在一个数 $\ell=O(\log n)$ 使得：如果 s，t 在 G 中是连通的，在 s，t 之间必然存在长度至多为 ℓ 的路径。事实上，引理 21.3 表明，在任意的正则 n-顶点图 G 中，随机游走访问的第 ℓ 个顶点所服从的分布与均匀分布之间的统计距离(或 L_1 距离)至多为 $\sqrt{n}\lambda^{\ell}$。特别地，这表明：如果 G 的连通分支 H 是一个扩张图且 $\lambda(H)$ 与 1 之间 [440]
的距离存在下界，则从 H 中顶点 u 出发进行 $\ell=O(\log n)$ 步随机游走时将以正数概率访问 H 中所有顶点。

莱茵戈尔德算法背后的思想是通过对数空间隐式可计算的方式将 G 转换为另一个图 G' 使得 G 的每个连通分支都变为 G' 中的一个扩张图，但 G 中的两个不连通的顶点在 G' 中仍然保持不连通。

21.4.1　连通性问题的对数空间算法(定理 21.21 的证明)

通过在顶点上添加自环，我们可以假设输入图 G 的顶点度为 d^{20}，其中常数 d 充分大以至于存在(d^{20}，$d/2$，0.01)-扩张图 H。由于 H 的规模是常数，我们可以将它整个存储在 $O(1)$ 内存空间中[⊖]。令 $G_0=G$，对于 $k>1$，定义 $G_k=(G_{k-1}\textcircled{r}H)^{20}$，其中$\textcircled{r}$是 21.3.4 节

[⊖] 我们既可以用显式构造算法来构造它，也可以用穷举搜索算法枚举顶点数为 d^{20} 的所有图来找出它。

定义的替换乘积操作。

如果 G_{k-1} 是一个顶点度为 $d^{(20)}$ 的 N-顶点图，则 G_{k-1} ⓡ H 是顶点度为 d 的 $d^{(20)}N$-顶点图。因此 $G_k=(G_{k-1}$ ⓡ $H)^{20}$ 是顶点度为 d 的 $d^{(20)}N$-顶点图。注意，如果两个顶点在图 G_{k-1} 是连通的，则它们在图 G_k 中也是连通的；同样，如果两个顶点在图 G_{k-1} 是不连通的，则它们在图 G_k 中也是不连通的。一个重要的观察结果是：$G_{10\log n}$ 是一个扩张图。因此，UP-ATH 问题在 $G_{10\log n}$ 上很容易求解。具体地讲，下面的论断成立。

论断：对任意 $k>0$，G_k 的每个连通分支都是 $(d^{(20)}n,\ d^{(20)},\ 1-\varepsilon)$-扩张图，其中 $\varepsilon=\min\{1/20,\ 1.5^k/(12n^2)\}$，而 n 是 $G=G_0$ 中顶点的个数。

证明　事实上，由引理 21.16 和引理 21.18 可知，对任意 $\varepsilon<1/20$ 和顶点度等于 D 的图 F，如果 $\lambda(F)\leqslant 1-\varepsilon$，则 $\lambda(F$ ⓡ $H)\leqslant 1-\varepsilon/25$，进而 $\lambda((F$ ⓡ $H)^{20})\leqslant 1-2\varepsilon$。再由引理 21.4 可知，$G$ 的每个连通分支的扩张度至多为 $1-\dfrac{1}{12n^2}$（注意，每个连通分支中顶点的个数均至多为 n）。 ■

于是，$G_{10\log n}$ 的每个连通分支都是扩张度至多为 $1-1/20$ 的扩张图。因此，为了判断 s、t 在 $G_{10\log n}$ 中是否连通，我们只需枚举 $G_{10\log n}$ 中始于 s 且长度至多为 $l=O(\log n)$ 的路径，查看其中是否有一条路径能够访问 t。困难在于，算法的输入图是 G 而不是 $G_{10\log n}$。相对简单一些的问题是考虑：给定 G 之后，我们的算法能否在 G_k（其中 $k=10\log n$）上执行单步的随机游走。也就是说，给定 G_k 中某个顶点 s 的一种表示形式和一个编号 $i\in[d^{(20)}]$，我们的算法需要只利用对数空间计算 s 在 G_k 中的第 i 个相邻顶点。不难看到，如果我们能够在对数空间中执行单步的随机游走，则引入一个计数器并且重复计算每步随机游走所需的空间，就可以很容易地通过重复单步随机游走来完成 l-步随机游走。

441

由于图 G_k 等于图 $(G_{k-1}$ ⓡ $H)^{20}$，因此只需说明我们能够在对数空间中完成 G_{k-1} ⓡ H 上的单步随机游走（然后，将同样的过程重复 50 遍就可以完成 G_k 上的单步随机游走）。现在，根据替换乘积的定义，G_{k-1} ⓡ H 上的每个顶点都是一个序对 $\langle u,\ v\rangle$，其中 u 是 G_{k-1} 的一个顶点而 v 是 H 的一个顶点。$\langle u,\ v\rangle$ 的每个相邻顶点表示为序对 $\langle b,\ i\rangle$，其中 $b\in\{0,\ 1\}$ 而 $i\in[d/2]$。如果 $b=0$，则 $\langle u,\ v\rangle$ 的第 $\langle b,\ i\rangle$ 个相邻顶点是 $\langle u,\ v'\rangle$，其中 v' 是 v 在 H 中的第 i 个相邻顶点。如果 $b=1$，则 $\langle u,\ v\rangle$ 的第 $\langle b,\ i\rangle$ 个相邻顶点是 $\langle u',\ v'\rangle$，它是 G_{k-1} 的旋转映射在 $\langle u,\ v\rangle$ 上的像。（也就是说，u' 是 u 在 G_{k-1} 中的第 v 个相邻顶点，而 v' 是 G_{k-1} 中 u 在 u' 的邻域中的编号。上述过程已经给出了计算 G_k 的旋转映射的显而易见的递归算法。令 s_k 表示该递归算法计算 G_k 的旋转映射时所需空间的大小，我们看到 s_k 满足方程 $s_k=s_{k-1}+O(1)$，这意味着 $s_{10\log n}=O(\log n)$。[⊖]

21.5　弱随机源和提取器

不管随机算法中还存在什么样的哲学困难，现在我们都假设它已经足够令人满意了，进而也无需基于一些未经证明的假设条件来将随机算法去随机化了。即使这样，我们依然

⊖ 实现算法的过程中，在算法的递归调用时需要小心地避免对输入进行复制。正确的做法是，将编号 k、顶点以及边标号都放入全局内存空间中，让所有的递归调用过程都可以访问全局内存。否则，所实现的算法的空间复杂度将变成 $O(\log\log\log n)$。更多实现细节，请参考原始论文[Rei05]或[Gol08 的第 5.2.4 节]。

要面对如下事实：现实世界中恐怕很难存在像“投掷一系列互不关联且无偏的硬币”这样的随机源和不可预测性。那我们能用现实世界中的“弱随机源”来运行随机算法吗？

21.5.1　最小熵

对于初学者，我们需要定义弱随机源的含义。历史上，人们对此给出了好几种定义，例 21.23 回顾了这些定义。下面的定义源自 D. 朱克曼(D. Zukerman)，该定义涵盖了之前的所有定义。

定义 21.22　设 X 是一个随机变量。X 的最小熵，记为 $H_\infty(X)$，指的是使得“$\Pr[X=x]\leqslant 2^{-k}$ 对 X 的定义域中的任意 x 成立”的最大实数值 k。

如果 X 是 $\{0,1\}^n$ 上满足 $H_\infty(X)\geqslant k$ 的概率分布，则称 X 是一个 (n,k)-随机源。

不难看到，如果 X 是 $\{0,1\}^n$ 上的随机变量，则 $H_\infty(X)\leqslant n$，而且 $H_\infty(X)=n$ 当且仅当 X 服从均匀分布 U_n。本节的目标是在给定的 $H_\infty(X)$ 尽可能小的随机源 X 上确保随机算法能够顺利运行。不难证明，采用了 k 个随机二进制位的随机算法在一般的模拟运行中需要使用最小熵至少为 k 的概率分布 X(参见习题 21.18)，从这个意义上讲，模拟随机算法运行时要求所使用的分布的最小熵满足一定的最低要求。 442

例 21.23　本例将会表明，最小熵是一个非常一般的概念，它可以用来为许多其他的“非完美随机源”进行建模。下面讨论 $\{0,1\}^n$ 上的一些概率分布 X。

- (冯·诺依曼模型：偏斜硬币) X 由 n 次独立的硬币投掷构成，每次硬币投掷时得到 1 的概率是 $\delta<1/2$ 而得到 0 的概率是 $1-\delta$。不难验证[⊖]，$H_\infty(X)=\log(1/(1-\delta))n$。
- (桑沙-瓦兹拉尼随机源(Santha-Vazirani Source)) X 具有如下的性质：对于任意 $i\in[n]$，和任意位串 $x\in\{0,1\}^{i-1}$，在 $X_1=x_1$，…，$X_{i-1}=x_{i-1}$ 的条件下，$\Pr[X_i=0]$ 和 $\Pr[X_i=1]$ 的概率都介于 δ 和 $1-\delta$ 之间。这种随机源是冯·诺依曼模型的推广，它可以用来对股票市场的价格波动进行建模，因为当前的股票价格有限地依赖于它的历史价格。不难验证，$H_\infty(X)\geqslant\log(1/(1-\delta))n$。
- (位定随机源和广义位定随机源) 在位定随机源 X 中，有一个满足 $|S|=k$ 的子集 $S\subseteq[n]$ 使得：X 中坐标编号属于 S 的所有随机二进制位服从 $\{0,1\}^k$ 上的均匀分布，并且 X 中坐标编号属于 $[n]\setminus S$ 的所有二进制位构成一个固定的位串(不妨设是全 0 位串)。则 $H_\infty(X)=k$。如果 X 中坐标编号属于 $[n]\setminus S$ 的所有二进制位是坐标编号属于 S 的所有随机二进制的确定型函数，则称 X 是广义位定随机源，此时，$H_\infty(X)=k$ 仍然成立。例如，如果 X 的奇数二进制位相互独立且服从均匀分布，并且在任意的偶数位置 $2i$ 上都有 $X_{2i}=X_{2i-1}$，则 $H_\infty(X)=\left\lceil\frac{n}{2}\right\rceil$。这种模型可以用来刻画高频率采样数据(例如，对于每分钟才变化一次的物理事件，每秒钟对它进行一次采样)。
- (线性子空间) 如果 X 是 $GF(2)^n$ 的 k 维线性子空间上的均匀分布，则 $H_\infty(X)=k$。

⊖ 事实上，随着 n 增大，X 接近于最小熵等于 $H(\delta)n$ 的概率分布，其中 H 是香农熵函数 $H(\delta)=\delta\log\frac{1}{\delta}+(1-\delta)\log\frac{1}{1-\delta}$。同样的结论对下面定义的桑沙-瓦兹拉尼随机源也成立。要进一步了解该结论以及具有同样形式的更一般的结论，请参阅[DFR'07]。

（此时，X 是位定随机源的推广。你能说明这是为什么吗？）

- （子集上的均匀分布）如果 X 是集合 $S\subseteq\{0,1\}^n$ 上的均匀分布，其中 $|S|=2^k$，则 $H_\infty(X)=k$。后面将会看到，子集上的均匀分布是一种非常一般的分布，它本质上刻画了满足 $H_\infty(X)=k$ 的所有分布。◀

21.5.2　统计距离

现在，我们形式地定义"从(n,k)-随机源中提取随机位"的含义。更准确地讲，我们要从(n,k)-随机源中提取几乎随机的位。我们将使用统计距离（参见 A.2.6 节）的概念来量化两个分布之间的接近程度。回顾一下，如果 X，Y 是值域 Ω 上的两个概率分布，则 X 和 Y 之间的统计距离 $\Delta(X,Y)$ 指的是

443

$$\max_{f:\Omega\to\{0,1\}} |E[f(X)]-E[f(Y)]| \tag{21.12}$$

人们已经证明，$\Delta(X,Y)=1/2|x-y|_1$，其中 x，y 分别是 $\mathbf{R}^\Omega$ 中表示分布 X 和 Y 的向量。对于任意 $\varepsilon>0$，如果分布 X 和 Y 满足 $\Delta(X,Y)\leqslant\varepsilon$，则称 X 和 Y 是 ε-近的，记为 $X\approx_\varepsilon Y$。

21.5.3　随机性提取器的定义

现在，我们定义随机性提取器——也就是能将"(n,k)-随机源转换为几乎均匀的分布"的函数。提取器需要使用少量额外的真随机位，这些真随机位称为随机种子。在下面的定义中，我们用 d 表示随机种子的长度。

定义 21.24（随机性提取器）　函数 $\mathrm{Ext}:\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$ 称为(k,ε)-提取器，如果在任意(n,k)-随机源 X 上，$\mathrm{Ext}(X,U_d)$的分布与 U_m 是 ε-近的。（对任意 ℓ 而言，U_ℓ 表示$\{0,1\}^\ell$ 上的均匀分布。）

为什么提取器需要额外的输入？

前面已经指出，提取器的目标是要在不使用完美无偏硬币的条件下运行随机算法。但是，单独的提取器似乎无法达成这一目标，因为我们只能保证在额外给定服从均匀分布的随机种子时提取器的输出才接近于均匀分布。"提取器需要额外的输入"可以有两种解释。其一，提取器要求提供额外的输入是必要的。事实上，对于任意函数 $\mathrm{Ext}:\{0,1\}^n\to\{0,1\}^m$和任意的$k\leqslant n-1$，必然存在$(n,k)$-随机源使得 $\mathrm{Ext}(X)$的第一个位是常数（亦即，概率为 1 地等于某个 $b\in\{0,1\}$）。于是，$\mathrm{Ext}(X)$的分布与均匀分布之间的统计距离至少是 1/2（习题 21.17）。其二，如果提取器的第二个输入的长度 t 充分小（比如，$t=O(\log n)$），则我们在模拟随机算法的运行时将只需枚举可能的 2^t 个随机种子，而无需再使用真随机位。显然，要使提取器是非平凡的，d 必须比较小。在最平凡的情况下，$d\geqslant m$，此时提取器可以忽略它的第一个输入而直接输出随机种子！

21.5.4　提取器的存在性证明

可以证明，如果忽略计算效率问题，则存在非常好的提取器。

定理 21.25　对于任意 k，$n\in\mathbf{N}$ 和 $\varepsilon>0$，存在(k,ε)-提取器 $\mathrm{Ext}:\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$ 使得 $d=\log n+2\log(1/\varepsilon)+O(1)$。

证明　如果(n,k)-随机源 X 是$\{0,1\}^n$ 的某个规模为 2^k 的子集上的均匀分布，则称 X 是扁平的。习题 19.7 证明了，每个(n,k)-随机源都是若干个(n,k)-扁平随机源的凸

组合。由于分布 Y_1，…，Y_N 的凸组合得到的分布与分布 U 之间的统计距离至多是 $\max\Delta(Y_i, U)$(习题 21.19)，故只需给出一个函数 Ext 使得当 X 是(n, k)-扁平随机源时 $\mathrm{Ext}(X, U_d)$与均匀分布之间非常接近。

现在，我们用概率方法来证明这种提取器确实存在。将 Ext 取为从$\{0,1\}^n\times\{0,1\}^d$到$\{0,1\}^m$ 的随机函数。令 X 是一个(n, k)-扁平随机源，而 $f:\{0,1\}^n\to\{0,1\}$是一个 444
函数。如果 Ext 是随机选取的，则数学期望 $E[f(\mathrm{Ext}(X, U_d))]$可以通过 f 在 $2^n\times 2^d$ 个点上的函数值来计算。因此，由切尔诺夫界可知，上述数学期望偏离 $E[f(U_n)]$超过 ε 的概率至多为 $2^{-\varepsilon^2 2^{k-d}/n^2}$。这意味着，若 $d>\log n+2\log(1/\varepsilon)+3$，则上述概率不超过 $2^{-2n(2^k)}$。又由于扁平分布的个数不超过$(2^n)^{2^k}$ 个并且从$\{0,1\}^n$ 到$\{0,1\}$的函数也不超过 2^{2^n} 个，因此由合并界限可知，存在 Ext 的一种选择使得

$$|E[f(\mathrm{Ext}(X,U_d))]-E[f(U_n)]|<\varepsilon$$

对每个(n, k)-扁平随机源和任意函数 $f:\{0,1\}^n\to\{0,1\}$都成立。换句话说，对任意(n, k)-扁平随机源 X 而言，$\mathrm{Ext}(X, U_d)$与 U_n 是 ε -近的。进而，对任意(n, k)-随机源 X 而言，$\mathrm{Ext}(X, U_d)$与 U_n 也是 ε -近的。 ■

[NZ93，RST97]证明了存在绝对常数 c 使得任意非平凡的(k, ε)-提取器(亦即，$\varepsilon<1/2$ 且输出长度大于随机种子的长度)必须满足 $d\geqslant\log(n-k)+2\log(1/\varepsilon)-c$。在此意义下，定理 21.25 的证明过程给出的提取器是最优的。

21.5.5　基于哈希函数构造提取器

定理 21.25 给出的非显式提取器用处不大，因为大多数应用需要显式的提取器，也就是多项式时间内可计算的提取器。利用两两独立的哈希函数可以得到这样一个显式的提取器(尽管它需要较长的随机种子)。

回顾一下 8.2.2 节，从$\{0,1\}^n$ 到$\{0,1\}^m$ 的函数族$\mathcal{H}$称为两两独立的，如果对$\{0,1\}^n$中任意$x\neq x'$和任意 y，$y'\in\{0,1\}^m$ 均有"$h(x)=y$ 且$h(x')=y'(h\in_R\mathcal{H})$"的概率为 2^{-2m}。这种函数族存在显式的构造方法，其中每个函数 h 都可以描述为长度为 $n+m$ 的位串(为简单计，我们将这样的位串也表示为 h)。于是，从函数族中随机选择一个函数就相当于从$\{0,1\}^{n+m}$中随机选择一个位串。下面这个著名的引理表明，在恰当的参数设置下，映射 $x, h\mapsto h(x)\circ h$(其中 $\circ$ 表示串接)是一个提取器。从参数值的角度看，这个提取器并不是非常好，但是，它在很多场合中却非常有用。

引理 21.26　(残余哈希引理(Leftover Hash Lemma)[BBR88，ILL89])　*如果 $m=k-2\log(1/\varepsilon)$，则对于任意(n, k)-随机源 X 都有*

$$\Delta(H(X)\circ H, U_m\circ H)<\varepsilon$$

其中 H 是从$\{0,1\}^n$ 到$\{0,1\}^m$ 的两两独立哈希函数族中随机选取的一个函数(的位串表示)。

证明　将 H 等同于 U_l，其中 $l=n+m$ 是哈希函数的位串表示的长度。我们研究 $H(X)\circ H$ 的冲突概率。亦即，随机选取 $h, h'\in_R\mathcal{H}$和 $x, x'\in_R X$ 时 $h(x)\circ h=h'(x')\circ h'$的概率。这个概率不超过 $h=h'$的概率(它等于 2^{-l})乘以 $h(x)=h(x')$的概率。$h(x)=h(x')$的概率又不超过 2^{-k} 与 2^{-m} 之和，前者是(n, k)-随机源 X 使得 $x=x'$的概率上界，后者是 $x\neq x'$但随机选择的哈希函数 $h\in_R\mathcal{H}$满足 $h(x)=h(x')$的概率。因此，$H(X)\circ H$ 的冲突概率至 445
多为 $2^{-l}(2^{-k}+2^{-m})=2^{-(l+m)}+2^{-l-k}$。

现在，将分布 $H(X)\cdot H$ 视为概率向量 $p\in\mathbf{R}^{[\cdot]}$，则它的冲突概率恰好是 p 的 L_2-范数的平方。我们可以记 $p=\mathbf{1}+w$，其中 $\mathbf{1}$ 对应于均匀分布 $U_n\cdot H=U_{n+\cdot}$ 的概率向量，而 w 与 $\mathbf{1}$ 正交。（对于一般的向量 p，我们只能得到 $p=a\mathbf{1}+w$，其中 a 是一个实数。但是由于 p 是一个概率向量，故必然有 $a=1$，否则等式右端各个分量之和将不等于 1。）于是，由勾股定理可知，$\|p\|_2^2=\|\mathbf{1}\|_2^2+\|w\|_2^2$。又由于 $\|\mathbf{1}\|_2^2=2^{-[\cdot]}$，故我们得到

$$\|w\|_2^2=\|p-\mathbf{1}\|_2^2=\|p\|_2^2-\|\mathbf{1}\|_2^2\leqslant(2^{-(n+k)}+2^{-(n-k)})-2^{-[\cdot]}=2^{-n-k}$$

利用 L_1-范数和 L_2-范数之间的关系（论断 2.1.1），我们看到

$$\begin{aligned}\Delta(H(X)\cdot H,U_{[\cdot]n})=1/2\|p-\mathbf{1}\|_1&\leqslant\frac{1}{2}2^{([\cdot]+n)/2}\|p-\mathbf{1}\|_2\\&\leqslant 2^{n/2+k/2-\log(1/\varepsilon)}2^{-n/2-k/2}\\&<\varepsilon\end{aligned}$$

■

21.5.6　基于扩张图的随机游走构造提取器

我们也可以用扩张图来显式地构造提取器。

引理 21.27　设 $\varepsilon>0$。对于任意 n 和 $k\leqslant n$，存在显式的 (k,ε)-提取器 $\mathrm{Ext}:\{0,1\}^n\times\{0,1\}^t\to\{0,1\}^m$，其中 $t=O(n-k+\log 1/\varepsilon)$。

证明　假设 X 是一个 (n,k)-随机源，a 是 X 的一个抽样样本，并且 G 是一个 $(2^n,d,1/2)$-扩张图，其中 d 是一个常数（参见定义 21.6 和定理 21.19）。

取真随机串 z 作为随机种子，其长度 $t=\log d(n/2-k/2+\log 1/\varepsilon+1)=O(n-k+\log 1/\varepsilon)$。我们将 z 视为 G 上始于标号为 a 的顶点的长度为 $l=n/2-k/2+\log 1/\varepsilon+1$ 的随机游走。也就是说，我们将 z 视为取自 $[d]$ 的 l 个标号，它给出了随机游走所采取的各个步骤。提取器的输出 $\mathrm{Ext}(a,z)$ 是这个随机游走最后到达的顶点的标号。

沿用定理 21.3 的证明过程（参见(21.1)式）可以看到，如果用 p 表示 X 的概率向量并且用 A 表示 G 的随机游走矩阵，则

$$\|A^lp-\mathbf{1}\|_2\leqslant 2^{-l}\|p-\mathbf{1}\|_2$$

又由于 X 是 (n,k)-随机源，故 $\|p\|_2^2$ 表示 X 的冲突概率且 $\|p\|_2^2$ 不超过 2^{-k}。进而，$\|p-\mathbf{1}\|_2\leqslant\|p\|_2+\|\mathbf{1}\|_2\leqslant 2^{-k/2}+2^{-n/2}\leqslant 2^{-k/2+1}$。因此，在我们选取的 l 上，有

$$\|A^lp-\mathbf{1}\|_2\leqslant 2^{-n/2+k/2-\log(1/\varepsilon)+1}2^{-k/2+1}\leqslant\varepsilon 2^{-n/2}$$

446　最后，借助 L_1-范数和 L_2-范数之间的关系即可完成定理证明。■

21.5.7　由伪随机数产生器构造提取器

多年以来，人们显式构造的随机性提取器的性能参数远远小于定理 21.25 中非显式提取器给出的最优性能参数。例如，在 $k=n^\varepsilon$（其中 $\varepsilon>0$ 是任意小的常数）时，人们还从未给出一种显式的提取器使得我们能够用长度约为 k 的位串作为 (n,k)-随机源的随机种子来执行多项式时间随机算法。（一般地，将 k 视为 n 的函数时，k 越小，则提取器的构造就越难。直观上看，如果 $n\gg k$，则很难用 k 个二进制位来“提炼”隐藏在 n 个二进制位中的随机性。）为了实现上述目标，我们应当考虑构造这样的提取器：它使用长度为 $O(\log n)$ 的随机种子从 (n,n^δ)-随机源中提取随机性，并且提取器的输出至少具有多项式长度（亦即，输出中至少含有 n^δ 个二进制位，其中 $\delta>0$ 是某个数）[⊖]。1999 年，特雷维山（Trevisan）说明

⊖ Ta-Shma 的工作[TS96]接近于该目标，他们构造了一个提取器，其中的随机种子的长度略高于超对数长度。

了如何利用改进的提取器构造方法来实现上述目标。与特雷维山的结果本身相比，他的思想更有价值。他证明了，如果用恰当的方式来看待第 19 章和第 20 章中的伪随机数产生器，则它们实际上就是随机性提取器。这个结果让人大感意外，因为这些伪随机数产生器依赖于难度假设(比如，线路复杂度较高的函数在 $\mathbf{E}=\mathbf{DTIME}(2^{O(n)})$ 中的存在性)。因此，伪随机数产生器在构造随机性提取器时似乎没什么用处，因为我们构造的提取器需要无条件地得到分析而不能依赖于任何未经证明的复杂性假设。

经过深入思考，我们认识到伪随机数产生器和随机性提取器的上述区别主要是由它们要面对的"敌手"或"区分器"的类型不同而造成的。伪随机数产生器要面对的敌手是所有计算能力受限的算法(亦即，能够被具有指定规模的线路计算的所有算法)。另一方面，提取器要面对的敌手则是所有的布尔函数，这是因为随机性提取器输出 $\{0,1\}^m$ 上的一个分布 $\mathcal{D}$，它与均匀分布 U_m 之间的统计距离不能超过 ε：这意味着 $|\Pr_{x\in\mathcal{D}}[D(x)=1]-\Pr_{x\in U_m}[D(x)=1]|<\varepsilon$ 需要对任意 $D:\{0,1\}^m\to\{0,1\}$ 成立。

特雷维山还进一步注意到，我们除了可以常规地认为伪随机数产生器只有一个输入之外，也可以认为它有两个输入：一个较短的随机种子和函数 f 的真值表，其中 f 是候选的难解函数。我们所得到的定理表明，如果 f 确实是一个难解函数，则由它构造的伪随机数构造器将是正确的。这些定理的证明过程实际上是构造性的——它们将"用于区分伪随机数产生器的输出和真随机串"的区分器 D 转换为一个计算 f 的线路 A。线路 A 把区分器 D 当做黑盒来使用。因此，即使 D 是一个不能用小规模线路来计算的任意的函数，则我们也可以运用上述的变换过程。这正是特雷维山论证过程的关键。

具体地讲，要实现我们的想法，需要用最坏复杂度较高的函数 f 来构造伪随机数产生器(如定理 20.7 所述)，这是一种较强的构造过程。如果存在区分器 D 可以区分伪随机数产生器的输出和均匀分布，则伪随机数产生器的正确性证明过程将得到一个算法，它能够在所有输入上计算候选的难解函数 f。形式地，我们有如下的算法，其中 G^f 指的是将 f 作为黑盒来使用的算法 G。

447

定理 21.28　(定理 20.7 的构造性形式)　对于任意时间可构造函数 $S:\mathbf{N}\to\mathbf{N}$(亦即，"安全性参数")，均存在常数 c 和算法 G，R 满足下列条件：

- 在输入函数 $f:\{0,1\}^l\to\{0,1\}$ 和输入位串 $z\in\{0,1\}^d$ 上，算法 G 在 2^{cd} 时间内输出一个长度为 $m=S(l)^{1/c}$ 的位串 $G^f(z)$。
- 如果函数 $D:\{0,1\}^m\to\{0,1\}$ 满足 $|E[D(G^f(U_d))]-E[D(U_m)]|>1/10$，则存在一个长度至多为 $S(l)^{1/c}$ 的建言位串 a 使得 $R^D(a,x)=f(x)$ 对任意输入 x 都成立，并且 R 的运行时间至多为 $S(l)^c$。

定理 21.28 中算法 R 就是第 20 章证明伪随机数产生器的正确性时使用的归约算法。

下面是特雷维山提取器的构造。设 G 是定理 21.28 中的算法，X 是一个 (n,k)-随机源。不失一般性，假设 n 是 2 的方幂，亦即，$n=2^l$。令"安全性参数"$S(l)$ 表示 k。给定随机源产生的任意位串 f 和随机种子 $z\in\{0,1\}^{c\log n}$，提取器将 f 视为从 $\{0,1\}^l$ 到 $\{0,1\}$ 的函数，并且提取器输出

$$\mathrm{Ext}(f,z)=G^f(z) \tag{21.13}$$

于是，给定长度为 n 的位串和长度为 $c\log n$ 的随机种子，Ext 产生 $S(l)^{1/c}=k^{1/c}$ 个二进制位。下面证明 Ext 是一个提取器。

论断 21.29　对任意 k，n，(21.13)式定义的函数是一个 $(k,1/5)$-提取器。

证明　若不然，则存在 (n,k)-随机源 X 和布尔函数 D，它至少以 1/5 的偏斜度区分

了分布 $\mathrm{Ext}(X, U_d)$和均匀分布 U_m，其中 $m=S(l)^{1/c}$。因此，在随机选择的 $f\in_R X$ 面言，"区分器 D 以 1/10 的偏斜度区分 $G'(U_d)$和 U_m"的概率至少是 1/10。我们称使得"区分器 D 以 1/10 的偏斜度区分 $G'(U_d)$和 U_m"的 f 为"劣性的"。注意，对每个劣性 f，均存在建言 $a\in\{0,1\}^{d(l)}$ 使得映射 $x\mapsto R^D(x,a)$恰好计算了函数 f。由于 R^D 是一个确定型算法，这意味着劣性 f 的个数至多等于 a 的选择种数 $2^{d(l)}$。再由 X 是(n,k)-随机源，故在分布 X 下任何一个位串的概率都不超过 2^{-k}。因此，随机选取的 f 是劣性的概率至多为 $2^{d(l)}2^{-k}\ll 1/10$，这与 D 是良好的区分器矛盾。 ■

评注 21.30 对上述构造过程仍感困惑的读者应该弄清伪随机数产生器 G 的内部结构，以便清楚地理解上述构造过程。实际上，提取器 Ext 非常简单。给定弱随机源产生的位串 $f\in\{0,1\}^n$，提取器 Ext 先用纠错码(具体地讲，应该是可列表解码的纠错码)将 f 变换为位串 $\hat{f}\in\{0,1\}^{\mathrm{poly}(n)}$。直观上讲，这就将 f 的随机性"分散"到整个位串 $\hat{f}$上。然后，提取器利用 20.2.2 节中的尼数-维格德尔森伪随机数产生器来选取 $\hat{f}$的一个坐标子集。也就是说，$\hat{f}$被看成定义在 $s=O(\log n)$个二进制位上的函数，用长度为 $t=O(s)$的随机种子 z 来产生输出 $\hat{f}(z_{I_1})\cdots\hat{f}(z_{I_m})$，其中 $I_1,\cdots,I_m$ 都是$[t]$的规模为 s 的子集并且 $I_1,\cdots,$ 448
I_m 构成一个组合设计(参见定义 20.13)。

21.6　空间受限计算的伪随机数产生器

本节说明如何用提取器为空间受限的随机计算构造一个伪随机数产生器，它使得随机的对数空间计算仅用 $O(\log^2 n)$个随机二进制位就可以完成。在此，我们强调这个伪随机数产生器不使用任何未经证明的复杂性假设。

这里，我们的目标是将随机的对数空间计算(如 **BPL** 算法或 **RL** 算法)去随机化。回顾一下第 4 章中空间受限图灵机的格局图的概念。如果我们取定对数空间图灵机的长度为 n 的一个输入，则格局图的规模为 $\mathrm{poly}(n)$。如果对数空间图灵机是随机的，则它在执行过程中利用硬币投掷的随机输入结果在格局图中完成状态转移(亦即，每个格局有两条出边，每条出边被采用的概率都等于 1/2)。为了消除计算过程的随机性，我们将对数空间图灵机所用的随机位串替换为"伪随机数产生器"的输出(只是这个伪随机数产生器是为对数空间计算量身定制的而已)，并证明对数空间图灵机无法"区分"随机位串和伪随机数产生器的输出(也就是说，对数空间图灵机最后终止于接受状态的概率不会发生变化)。

定理 21.31 (尼散伪随机数产生器(Nisan's Pseudorandom Generator[Nis90]))　对于任意 d 而言，均存在 $c>0$ 和 $\mathrm{poly}(n)$时间可计算的函数 $g:\{0,1\}^{c\log^2 n}\to\{0,1\}^{n^d}$(亦即，"伪随机数产生器")使得：对于空间受限的任意图灵机 M，只要它在长为 n 的输入上具有规模$\leqslant n^d$的格局图，则

$$\left|\Pr_{r\in\{0,1\}^{n^d}}[M(x,r)=1]-\Pr_{z\in\{0,1\}^{c\log^2 n}}[M(x,g(z))=1]\right|<\frac{1}{10} \tag{21.14}$$

通过枚举尼散定理中随机数产生器 g 的长度为 $O(\log^2 n)$的所有输入，我们就可以在 $O(\log^2 n)$空间内模拟任意 **BPL** 算法的运行。注意，虽然塞维奇定理(定理 4.14)也表明**BPL**$\subseteq$**SPACE**$(\log^2 n)$，但它并没有得到这样的伪随机数产生器。事实上，可以进

一步加强定理 21.31 以证明：**BPL** 中的任何问题都可以用多项式时间算法在 $O(\log^2 n)$ 空间内判定，但我们在此并不打算证明这一结论。萨克斯(Saks)和周世宇(Shiyu Zhou)[SZ95]通过改进尼散的思想证明了 $\mathbf{BPL}\subseteq\mathbf{SPACE}(\log^{1.5} n)$，该结论导致很多研究者对 $\mathbf{BPL}=\mathbf{L}$ 的猜想(亦即，随机性根本无助于任何对数空间计算)。事实上，我们在 21.4 节中已经看到，无向连通性问题的著名的随机游走算法可以在对数空间中去随机化。

尼散构造和 $\mathbf{BPL}=\mathbf{L}$ 这一猜想背后的直观思想是相同的。由于对数空间图灵机只能对随机位串进行读操作而且它的内存只能存放 $O(\log n)$ 个二进制位，因此它只能"记住"见过的 $O(\log n)$ 个位。为了有效地利用有限的存储空间，我们需要使用下面的引理。它表明了 449
如何对仅了解少量信息的随机位串进行回收复用。

引理 21.32（回收复用引理）　如果 $f:\{0,1\}^n\to\{0,1\}^s$ 是任意函数，$\mathrm{Ext}:\{0,1\}^n\times\{0,1\}^t\to\{0,1\}^m$ 是 $(k,\varepsilon/2)$-提取器，其中 $t=n-(s+1)-\log\frac{1}{\varepsilon}$，则

$$\Delta(f(X)\circ U_t, f(X)\circ \mathrm{Ext}(X,U_t))<\varepsilon$$

其中 X 是在 $\{0,1\}^n$ 上服从均匀分布的随机变量。

让我们讨论 $s\ll n$ 且 $n=m$ 的情形，以此解释为什么引理 21.32 被称为回收复用引理。假设我们用长度为 n 的随机位串 X 来产生 $f(X)$。由于 $f(X)$ 的长度是 s，因此通常情况下 $\{0,1\}^s$ 中的每个位串在 f 之下都存在很多原像。因此，任何人在仅看到 $f(X)$ 的情况下都只能了解到关于 X 的非常有限的信息。更正式地讲，对于任意选定的 $f(X)$，在 f 下能产生这一函数值的所有 X 构成的集合可以视为一个弱随机源。引理 21.32 是说，如果利用恰当的提取器(其随机种子的长度仅为 $t=O(s+\log(1/\varepsilon))$)将引理 21.27 运用到 X 上，则我们将得到一个长度为 m 的新位串 $\mathrm{Ext}(X,z)$，它看上去是一个完全随机的(即使对知晓 $f(X)$ 的人而言亦是如此)。

证明　对 $v\in\{0,1\}^s$，我们用 X_v 表示在 $f^{-1}(v)$ 上服从均匀分布的随机变量。这样，$\Delta(f(X)\circ W, f(X)\circ \mathrm{Ext}(X,z))$ 可以表达为

$$\frac{1}{2}\sum_{v,w}\left|\Pr[f(X)=v\wedge W=w]-\Pr[f(X)=v\wedge \mathrm{Ext}(X,z)=w]\right|$$

$$=\sum_v \Pr[f(X)=v]\cdot\Delta(W,\mathrm{Ext}(X_v,z)) \qquad (21.15)$$

令 $V=\{v:\Pr[f(X)=v]\geqslant\varepsilon/2^{s+1}\}$。如果 $v\in V$，则我们可以将 X_v 视为 (n,k)-随机源，其中 $k\geqslant n-(s+1)-\log\frac{1}{\varepsilon}$。因此，根据提取器的定义可知，$\mathrm{Ext}(X_v,r)\approx_{\varepsilon/2} W$，进而 $v\in V$ 对(21.15)式的贡献至多为 $\varepsilon/2$。所有 $v\notin V$ 对(21.15)式的总贡献的上界为 $\sum_{v\notin V}\Pr[f(X)=v]\leqslant 2^s\times\frac{\varepsilon}{2^{s+1}}=\varepsilon/2$，由此即得引理。 ■

现在，我们给出回收复用引理在尼散构造中的用处。设 M 是一个对数空间图灵机。固定输入的规模为 n。则，对某个 $d\geqslant 1$ 而言，M 在输入上的格局图至多包含 n^d 个格局，并且 M 的运行时间 $L\leqslant n^d$(参见图 21-2)。由于未使用的随机二进制位可以被忽略，故不失一般性，我们可以假设 M 在每个计算步骤中使用一个随机位。此外，只要给 M 配备一条独立的带作为计数器，则我们还可以假设格局图是分层的：它一

共分为 L 层，第 i 层包含在第 i 个计算步骤时所能够到达的所有格局。第 1 层只含起始结点；最后一层只含“接受”和“拒绝”这两个结点；其余每层含有 $W=n^d$ 个结点。第 i 层的每个结点有两条出边，它们分别连接到第 $i+1$ 层的一个结点。图灵机在该结点上执行计算步骤时，先从随机串中取出下一个随机位，再根据该随机位的取值相应地沿着一条出边到达第 $i+1$ 层的结点。有时，我们称 L 是格局图的长度，而称 W 是格局图的宽度。

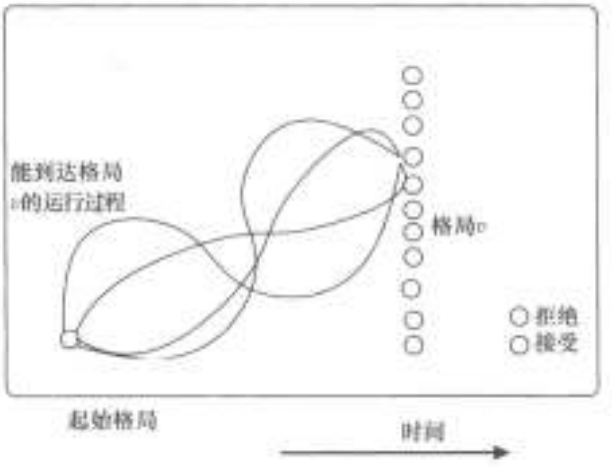


图 21-2　随机图灵机 M 的格局图。v 是中间层上的一个结点/格局。很多不同的随机串都能使得概率图灵机从起始格局到达格局 v

为简单计，我们先说明如何将所需的随机位的个数削减一半。试想将计算过程的 L 个步骤分为前、后两半，每半使用 $L/2$ 个随机位。假设我们用一个长度为 $L/2$ 的位串来执行前一半的计算步骤，完成之后图灵机的格局 v 恰好处于格局图的中间分层上。现在，关于 X，我们所能了解的信息只有：格局 v 在中间分层中的编号，它可以表示为长为 $d\log n$ 的位串。于是，前一半计算步骤的分支计算过程可以视为从 $\{0,1\}^{L/2}$ 到 $\{0,1\}^{d\log n}$ 的函数。回收复用引理使得我们可以用一个长度为 $O(\log n)$ 的随机种子来重新回收 X，得到另一个长度为 $L/2$ 的几乎随机的位串 $\mathrm{Ext}(X,z)$，它可以作为执行后一半计算步骤时所需的随机串。这样，在执行所有 L 个步骤时我们只使用了 $L/2+O(\log n)$ 个随机位，大致上将所需的随机位削减了一半。递归地使用类似的思想，尼散的伪随机数产生器用 $O(\log n\log L)$ 个随机位就能执行所有 L 个计算步骤。

现在，我们正式地定义尼散伪随机数产生器。

定义 21.33（尼散伪随机数产生器(Nisan's Generator)）　对某个 $r>0$ 和任意 $k\geqslant 0$，设 $\mathrm{Ext}_k:\{0,1\}^n\times\{0,1\}^r\to\{0,1\}^n$ 是提取器函数。与任意整数 $k\geqslant 0$ 关联的尼散伪随机数产生器 $G_k:\{0,1\}^n\to\{0,1\}^{2^k}$ 递归地定义如下(其中 $|a|=(k-1)r$，$|z|=r$)：

$$G_k(a\circ z)=\begin{cases}z_1(\text{即 } z \text{ 的第 1 个位}) & k=1\\ G_{k-1}(a)\circ G_{k-1}(\mathrm{Ext}_{k-1}(a,z)) & k>1\end{cases}$$

现在，我们用尼散伪随机数产生器来证明定理 21.31。我们只需证明，图灵机分别在真随机串和伪随机串下从起始结点出发最后进入接受结点的概率是相似的。但是，我们后面证明的结论还考虑了中间步骤，它比所需的结论更强一些。

如果 u 是格局图中的一个结点，s 是一个长度为 2^r 的位串，则图灵机从 u 出发把 s 当作随机串进行相应的计算之后，图灵机会到达格局图中的一个顶点，我们将这个顶点记为 $f_{u,r}(s)$。于是，如果 s 服从某个分布 $\mathcal{D}$，则 $f_{u,r}\mathcal{D}$ 定义了从 u 开始进行 2^r 个计算步骤之后计算格局所服从的分布。

引理 21.34　设 $r=O(\log n)$。对任意 $k\leqslant d\log n$，$\mathrm{Ext}_k:\{0,1\}^n\times\{0,1\}^r\to\{0,1\}^n$ 是 $(kr-2d\log n,\varepsilon)$-提取器，$G_k$ 是定义 21.33 所定义的尼散伪随机数产生器。对于上一自

然段描述的任意图灵机及其格局图中的任意结点 u，有

$$\Delta(f_{u,2^t}(U_{2^t}),f_{u,2^t}(G_k(U_m)))\leqslant 3^t\varepsilon \tag{21.16}$$

其中 U_l 表示 $\{0,1\}^l$ 上的均匀分布。

用引理 21.34 证明定理 21.31：设 $u=u_0$ 是起始格局，$2^t=L$ 是计算的总长度。选取 ε 使得 $3^t\varepsilon<1/10$，也就是说，$1/\varepsilon=O(\log L)=O(\log n)$。用 21.5.6 节中的 Ext_k 作为提取器，令 $r=O(\log n)$，则随机种子的长度 $kr=O(r\log L)=O(\log^2 n)$。■

引理 21.34 的证明　令 ε_k 是(21.16)式左端在所有概率型对数空间图灵机上达到的最大值。如果我们能用数学归纳法证明 $\varepsilon_k\leqslant 2\varepsilon_{k-1}+2\varepsilon$，则引理 21.34 就成立。基本情况 $k=1$ 是平凡的。在归纳步骤中，我们需要给出两个分布 $f_{u,2^t}(\mathcal{D}_1)$，$f_{u,2^t}(\mathcal{D}_5)$之间的统计距离的上界。为此，我们引入两个分布$\mathcal{D}_2$，$\mathcal{D}_3$，然后应用三角不等式($\Delta(\cdot,\cdot)$满足三角不等式，因为它定义了分布函数上的距离)得到：

$$\Delta(f_{u,2^t}(\mathcal{D}_1),f_{u,2^t}(\mathcal{D}_5))\leqslant\sum_{i=1}^{4}\Delta(f_{u,2^t}(\mathcal{D}_i),f_{u,2^t}(\mathcal{D}_{i+1})) \tag{21.17}$$

其中的四个分布的分别是

$$\begin{aligned}
\mathcal{D}_1&=U_{2^t}\\
\mathcal{D}_2&=G_k(U_m)\\
\mathcal{D}_3&=U_{2^{t-1}}\circ G_{k-1}(U_{\ell(k-1)r})\\
\mathcal{D}_4&=G_{k-1}(U_{\ell(k-1)r})\circ G_{k-1}(U'_{\ell(k-1)r}) \qquad (U,U'\text{ 表示独立的相同分布})\\
\mathcal{D}_5&=G_{k-1}(U_{\ell(k-1)r})\circ G_{k-1}(U_{\ell(k-1)r})
\end{aligned}$$

我们依次给出(21.17)式中各个求和项的上界。

论断 1：$\Delta(f_{u,2^t}(\mathcal{D}_1),f_{u,2^t}(\mathcal{D}_2))\leqslant\varepsilon_{k-1}$。

将 $\Pr[f_{u,2^{t-1}}(U_{2^{t-1}})=w]$表示为 $p_{u,w}$，将 $\Pr[f_{u,2^{t-1}}(G_{k-1}(U_{\ell(k-1)r}))=w]$表示为 $q_{u,w}$。根据归纳假设可知，

$$\frac{1}{2}\sum_w|p_{u,w}-q_{u,w}|=\Delta(f_{u,2^{t-1}}(U_{2^{t-1}}),f_{u,2^{t-1}}(G_{k-1}(U_{\ell(k-1)r})))\leqslant\varepsilon_{k-1}$$

由于$\mathcal{D}_1=U_{2^t}$可以视为$U_{2^{t-1}}$的两个独立的复制，故我们有

$$\Delta(f_{u,2^t}(\mathcal{D}_1),f_{u,2^t}(\mathcal{D}_2))=\sum_v\frac{1}{2}\Big|\sum_w p_{u,w}p_{w,v}-\sum_w q_{u,w}q_{w,v}\Big| \tag{452}$$

其中 w，v 是分别位于与结点 u 距离 2^{t-1} 个层和 2^t 个层的结点

$$\begin{aligned}
&=\sum_w p_{u,w}\frac{1}{2}\sum_v|p_{w,v}-q_{w,v}|\\
&\leqslant\varepsilon_{k-1} \qquad (\text{利用归纳假设和}\textstyle\sum_w p_{u,w}=1)
\end{aligned}$$

论断 2：$\Delta(f_{u,2^t}(\mathcal{D}_3),f_{u,2^t}(\mathcal{D}_4))\leqslant\varepsilon_{k-1}$。

证明过程与论断 1 类似，本处从略。

论断 3：$\Delta(f_{u,2^t}(\mathcal{D}_4),f_{u,2^t}(\mathcal{D}_5))\leqslant 2\varepsilon$。

我们运用回收复用引理来证明该论断。令函数 $g_u:\{0,1\}^{\ell(k-1)r}\to[1,W]$定义为 $g_u(a)=f_{u,2^{t-1}}(G_{k-1}(a))$。(换做自然语言也就是说，将尼散伪随机数产生器运用到随机种子 a 上，将产生的位串作为概率型对数空间图灵机的随机位串，从结点 u 开始执行 2^{t-1} 个计算步骤之后到达结点 $g_u(a)$)。设 X，$Y\in U_{\ell(k-1)r}$和 $z\in U_r$。根据回收复用引理可知，

$$g_u(X)\circ Y\approx_\varepsilon g_u(X)\circ\mathrm{Ext}_{k-1}(X,z)$$

引理 A.21 的第 5 个结论表明，如果在上式两端的第二项用一个(确定型)函数作用到其上，则等价关系仍成立。因此，

$$g_w(X) \cdot g_w(Y) \approx_\varepsilon g_w(X) \cdot g_w(\mathrm{Ext}_{k-1}(X,z))$$

其中 w 是 u 之后与 u 距离 2^{t-1} 层的任意结点。上式的左端对应于 $f_{u,t}(\mathcal{D}_1)$（也就是说，$\Pr[f_{u,t}(\mathcal{D}_1)=v]=\sum_w \Pr[g_w(X)=w \wedge g_w(Y)=v]$），而右端则恰好是 $f_{u,t}(\mathcal{D}_2)$。这就完成了引理的证明。 ■

本章学习内容

- 证明随机对象具有某种良好的性质通常比较容易，但是要显式地构造出具有这种性质的对象往往不是一件容易的事。而且，一旦找到了对象的显式构造，则这种显式构造通常极其有用。
- 图上随机游走的性质往往与邻接矩阵的特征值密切相关。邻接矩阵归一化之后得到的矩阵称为随机游走矩阵。因此，也可以说，图上随机游走的性质与随机游走的特征值密切相关。
- 扩张图族是由具有常数度的一些图构成的集合，其中每个图的第二大特征值与 1 之间的距离存在下界。用概率方法很容易证明扩张图族的存在性，而且人们也找到了扩张图族的显式构造方法。
- 扩张图上的 ℓ-步随机游走在某种程度上讲是“伪随机的”，它非常类似于用一定的标准随机选择 ℓ 个顶点。这一事实在很多场合中大有用处，它既可以用来为 **BPP** 算法设计随机高效的错误概率削减过程，也可以用来为无向图连通性问题设计对数空间
453 算法。
- 提取器是一种概率分布变换函数，它作用到具有较大的最小熵的概率分布上，所输出的分布接近于均匀分布。
- 虽然随机性提取器不使用任何未经证明的复杂性假设或下界，但是对于具有“黑盒”的伪随机数产生器的正确性分析却可以用来构造随机性提取器。

本章注记和历史

扩张图的定义最先由巴萨雷戈(Bassalygo)和平斯克尔(Pinsker)[BP73]给出，并且平斯克尔[Pin73]用概率方法证明了扩张图的存在性。他们定义扩张图的动机是，找出一个显式的图来代替加拉格尔(Gallager)[Gal73]所给出的纠错码构造方法中的随机图。马古利斯(Margulis)[Mar73]率先给出了扩张图族的显式构造，但他只证明了图族中任意图 G 的参数 $\lambda(G)$ 与 1 之间的距离存在下界，却未能明确地给出这个下界。盖贝尔(Gabber)和加利尔(Galil)[GG79]改进了马古利斯的分析并给出了 $\lambda(G)$ 的明确上界，该上界后来又进一步被金波(Jimbo)和马罗卡(Marouka)改进[JM85]。卢博茨基(Lubotzky)，菲利普斯(Phillips)和萨那克(Sarnak)[LPS86]以及马古利斯[Mar88]构造了拉马努詹图(Ramanujan Graph)，这种扩张图使得 λ 参数和顶点度之间的依赖关系达到了最优。d-正则图的第二大特征值的阿龙-波普潘纳下界最初出现在[Alo86]中，[Nil04]给出了误差项 $o(1)$ 的更紧界限。

扩张图的(基于特征值的)代数定义和组合定义之间的关系由多德扎克(Dodziuk)，阿龙(Alon)，米尔曼(Milman)等人在一系列论文[Dod84，AM84，AM85，Alo86]中逐步建立。辛克莱(Sinclair)和杰鲁姆(Jerrum)[SJ88]将这种关系推广到了一般的可逆马尔可夫链上。所有

这些结果都可以视为齐格尔(Cheeger)[Che70]在紧黎曼流形(Compact Riemannian Manifold)上得到的一个结果的离散形式。

引理 21.4(每个连通图都存在显著的谱鸿沟)源自阿龙和苏达科夫(Sudakov)[AS00a]，它是洛瓦兹(Lovász)的书[Lov07]中问题 11.29 的改进形式。引理 21.11(扩张图平稳引理)源自阿龙和 Chung[AC86](但该论文中要求 $T=V\setminus S$)。

卡普(Karp)，皮朋吉尔(Pippenger)和西普赛尔(Sipser)[KPS85]率先将扩张图用于去随机化。具体地说，他们借助扩张图证明了如何再多用 $O(k)$ 个随机位将 **RP** 算法的错误概率从 1/3 削减到 $1/\sqrt{k}$。奥伊陶伊(Ajtai)，科姆罗斯(Komlos)和塞梅尔雷迪(Szemeredi)[AKS87]率先将扩张图的随机游走用于随机化，他们的结果表明在确定型对数空间内用 $\log^2 n/\log\log n$ 个随机位就可以模拟 **RL** 算法的运行。科恩(Cohen)和维格德尔森(Wigderson)[CW89]以及因帕利亚佐(Impagliazzo)和朱克曼(Zukerman)[IZ89]独立地证明了如何借助[AKS87]中的分析和额外的 $O(k)$ 个随机位来将 **RP** 算法和 **BPP** 算法的错误概率从 1/3 削减到 2^{-k}(参见 21.2.5 节)。这种随机游走的改进分析由吉尔曼(Gillman)给出，他还证明了扩张图的切尔诺夫界(定理 21.15)。[Kah97，WX05，Hea06]还讨论了一些其他的改进。

21.3 节讨论的扩张图的显式构造源自莱茵戈尔德(Reingold)，瓦德翰(Vadhan)和维格德尔森[RVW00]，但我们给出的形式取自[RV05，RTV06]。格罗莫夫(Gromov)[Gro83]在特殊的超方体图上分析了替换乘积的扩张度，马丁(Martin)和兰德尔(Randall)[MR00]则在一般的图上分析了替换乘积的扩张度。在某种程度上讲，超方体图和一般的图在替换乘积的扩张度分析上是有区别的。 454

胡利(Hoory)，利尼亚尔(Linial)和维格德尔森[HLW06]精彩地介绍了扩张图以及它在计算机科学中的应用。阿龙和斯宾塞(Spencer)的书[AS00b]也包含了扩张图的几个结果。

随机性提取问题最早在 20 世纪 50 年代被冯·诺依曼[vN51]考虑过，他希望能够从偏斜硬币的随机独立的投掷结果中提取随机性。布卢姆[Blu84]将其结果推广到马尔可夫链上。桑沙(Santha)和瓦兹拉尼(Vazirani)[SV84]将随机性提取推广到了更具一般性的随机源，这种随机源现在被称为桑沙-瓦兹拉尼随机源(参见例 21.23)，他们的提取方法需要增加随机种子并且需要允许提出器的输入与均匀分布之间存在较小的统计距离。U. V. 瓦兹拉尼和 V. V. 瓦兹拉尼[VV85]说明了如何用桑沙-瓦兹拉尼随机源来模拟 **RP** 算法的运行。齐沃尔(Chor)和戈德赖希(Goldreich)[CG85]改进了[SV84，VV85]的分析，并推广了随机源的种类。特别地，他们引入了最小熵的概念并研究了区组随机源(Block Source)。在区组随机源中，每个区组都具有显著的最小熵，即使之前的区组是已知的。他们还研究了如何从最小熵较大的两个或两个以上的独立随机源中提取随机性，也就是说，从 $k>n/2$ 的几个 (n,k)-随机源中提出随机性。朱克曼(Zukerman)[Zuc90]推进了"用最小熵较大的单个随机源来模拟概率算法的运行"这一目标，并注意到了最小熵模型涵盖了所有现存的提取器模型(要了解那之前人们提出的各种提取器模型，请参阅[SZ94])。朱克曼还率先在 $k=\Omega(n)$ 的假设下用 (n,k)-随机源实现了对概率算法的模拟。同时，我们还注意到：在最小熵模型提出之前，提取器已经出现在了西普赛尔[Sip86]给出的"有条件的去随机化"方面的工作中，他给出的去随机化要求假设提取器(的某种变形)是存在的，但是他采用其他方式来描述提取器。

提取器也被显式地实现并应用于密码学中(只是随机种子的长度比较长)。这种提取器

既可以用哈希函数来构造，也可以用因帕利亚佐、勒维和卢比[ILL89]提出的回收复用引理(引理 21.26)来构造，还可以用早先由班尼特(Bennet)、布拉萨尔(Brassard)和罗伯特(Robert)[BBR88]给出的前驱性方法来构造。后来，尼散(Nisan)[Nis90]证明了，基于哈希函数构造的提取器(尤其是[VV85]给出的提取器)可以用来为对数空间计算构造伪随机数产生器使得它的优良性能能够被证明。尼散和朱克曼[NZ93]率先定义了提取器。他们还构造了一种新提取器，并用这种提取器得到了如下结果：概率算法所需的随机位的数量可以从空间复杂度的多项式规模削减到空间复杂度的线性规模。自那以后，有一系列漂亮的工作都致力于改进提取器的几个参数，在此过程中，人们提出很多重要的工具，这些工具在理论计算机科学的其他领域中也得到了应用。特别地，古鲁斯瓦米(Guruswami)等人[Guv07](稍微改进了吕及人(Chi-Jen Lu，台湾人)等人的工作)构造了一个提取器，它的随机种子的长度和输出的长度都是定理 21.25 中非显式最优提取器相应参数的常数倍。关于提取的构造及其应用，[Sha02]是一个很好的综述(但它稍显陈旧)。

特雷维山(Trevisan)[Tre99]首先认识到伪随机数构造器可以用来构造提取器(参见 21.5.7 节)，但他最初的理解目前已经得到了极大的扩展。人们现在已经知道，在不同领域中实际上研究了三个非常相似的组合对象：伪随机数产生器(属于密码学和去随机化领域)，提取器(属于弱随机源领域)和可列表解码的纠错码(属于编码理论和信息论领域)。
455
在这三个组合对象中，构造出其中一个对象通常也意味着构造出了其他两个对象。关于这方面的内容，请参阅瓦德翰(Vadhan)[Vad07]。

定理 21.31 是尼散[Nis90]证明的，他还证明了所有的 **BPL** 算法都可以同时在多项式时间和对数空间内被模拟。我们给出的证明源自因帕利亚佐、尼散和维格德尔森[INW94]，但其中基于提取器的观点却属于莱斯(Raz)和莱茵戈尔德(Reingold)。萨克斯(Saks)和周世宇(Shiyu Zhou)[SZ95]扩展了尼散的技术，证明了每个 **BPL** 问题都存在空间复杂度为 $O(\log^{1.5} n)$的算法。

或许是由于无向图连通性问题是 **RL** 问题最重要的例子，各种文献一直对它给予了特殊的关照。尼散、塞梅尔雷迪和维格德尔森[NSW92]首次为无向图连通性问题给出了空间复杂度为 $n(\log^2 n)$(确切地说是 $O(\log^{1.5} n)$)的确定型算法。正如前面所说，这个结果后来被[SZ95]推广到所有 **RL** 问题上。阿尔莫尼(Armoni)等人[ASTWZ97]将无向图连通性问题的空间复杂度改进到 $O(\log^{1.5} n)$。无向图连通性问题的确定型空间复杂度最终被莱茵戈尔德[Rei05]解决，他证明了该问题属于 **L**(定理 21.21)。特里福诺夫(Trifonov)[Tri05]同时独立地证明了一个相对较弱的结果，他为无向图连通性问题给出了一个空间复杂度为 $O(\log n \log\log n)$的算法。

习题

21.1　利用柯西-西瓦兹不等式(Cauchy-Schwarz Inequality)证明论断 21.1，其中柯西-西瓦兹不等式断言：$|\langle u, v\rangle| \leqslant \|u\|_2 \|v\|_2$ 对任意向量 $u, v \in \mathbf{R}^n$ 成立。

21.2　(a) 证明霍尔德不等式(Hölder's Inequality，参见 A.5.4 节)：对于任意满足 $\frac{1}{p}+\frac{1}{q}=1$的任意 p，q，均有$\|u\|_p \|v\|_q \geqslant \sum_{i=1}^{n} |u_i v_i|$。注意，柯西-西瓦兹不等式是霍尔德不等式在 $p=q=2$ 时的特殊情况。

(b) 对任意向量 $v \in \mathbf{R}^n$，定义 $\|v\|_\infty = \max_{i \in [n]} |v_i|$。证明 $\|v\|_\infty$ 是一种范数，且对任意向量 v，有

$$\|v\|_\infty = \lim_{p\to\infty}\left(\sum_{i=1}^{n}|v_i|^p\right)^{1/p}$$

(c) 证明：$\|v\|_2 \leqslant \sqrt{\|v\|_1\|v\|_\infty}$ 对任意向量 $v\in\mathbf{R}^n$ 成立。

21.3 证明：如果 G 是一个 n -顶点二分图，则存在向量 $v\in\mathbf{R}^n$ 使得 $Av=-v$，其中 A 是 G 的随机游走矩阵。

21.4 证明：对于任意 d -正则的 n -顶点图，G 的直径至多为 $3n/(d+1)$，其中直径指的是不同顶点 i，j 在图 G 中最短路径的长度所达到的最大值。

21.5 回顾一下，矩阵 A 的谱范数 $\|A\|$ 定义为 $\|Av\|_2$ 在所有单位向量 v 上达到的最大值。设 A 是一个对称随机矩阵，也就是说，A 的元素都是非负值，$A=A^{\mathrm{T}}$ 且各行、各列元素之和都等于 1。证明：$\|A\|\leqslant 1$。 456

21.6 令 A，B 是两个 $n\times n$ 的矩阵。

(a) 证明：$\|A+B\|\leqslant\|A\|+\|B\|$。

(b) 证明：$\|AB\|\leqslant\|A\|\|B\|$。

21.7 设 A，B 是两个对称随机矩阵，证明：$\lambda(A+B)\leqslant\lambda(A)+\lambda(B)$。

21.8 证明引理 21.16。

21.9 (a) 证明：如果概率分布 X 的定义域(support)的大小至多为 d，则该分布的冲突概率至少为 $1/d$。

(b) 证明：如果 G 是一个 (n,d,λ)-扩张图，X 是 G 的第一个顶点的随机相邻顶点服从的概率分布，则 X 的冲突概率至多为 λ^2+1/n。

(c) 证明：$\lambda\geqslant\sqrt{\frac{1}{d}-\frac{1}{n}}=\frac{1}{\sqrt{d}}+o(1)$，其中 $o(1)$ 表示随 n 的增大而趋于 0 的量。

21.10 回顾一下，矩阵 A 的迹 $\mathrm{tr}(A)$ 指的是它的所有对角线元素之和。

(a) 证明：如果 $n\times n$ 矩阵 A 的所有特征值为 $\lambda_1,\cdots,\lambda_n$，则 $\mathrm{tr}(A)=\sum_{i=1}^{n}\lambda_i$。

(b) 证明：如果 A 是 n -顶点图 G 的随机游走矩阵并且 $k\geqslant 1$，则 $\mathrm{tr}(A^k)$ 是如下随机事件的概率的 n 倍：均匀随机地选择一个顶点 i，从 i 出发随机游走 k 步恰好回到顶点 i。

(c) 证明：对任意 d -正则图 G，$k\in\mathbf{N}$ 和 G 的顶点 i，始于顶点 i 且长度为 k 的路径终止于顶点 i 的概率大于等于 T_d 的概率，其中 T_d 是以 i 为根且深度为 k 的 $(d-1)$-完全树。(在 $(d-1)$-完全树中，每个内结点的度都等于 d，它有一个父结点和 $d-1$ 个子结点。)

(d) 证明：对于偶数 k，始于 T_d 的根结点 v 且长度为 k 的路径终止于顶点 v 的概率至少为 $2^{k-4\log d-3/2+o(k)}$。

(e) 证明：对于度为 d 的任意 n -顶点图 G，$\lambda(G)\geqslant\frac{2}{\sqrt{d}}(1+o(1))$，其中 $o(1)$ 是依赖于 n 和 d 的项并且随着 n 和 d 增大 $o(1)$ 趋于 0。

21.11 设 (n,d)-随机图是按如下方法得到的 n -顶点图 G：随机选取从 $[n]$ 到 $[n]$ 的 d 个排列 $\pi_1,\cdots,\pi_d$；如果存在 $1\leqslant i\leqslant d$ 使得 $v=\pi_i(u)$，则在 G 的顶点 u，v 之间引入一条边 (u,v)。证明：对于任意 n 和 $d\geqslant 2$，(n,d)-随机图是 $(n,2d,1/10)$-扩张图的概率为 $1-o(1)$(亦即，该概率趋于 1)。

21.12 本习题研究如何将 21.2.5 节中的错误概率削减过程扩展到犯双面错误的

BPP 算法上。

(a) 证明：在定理 21.12 的假设下，对于任意子集 $I\subseteq[k]$，均有

$$\Pr[\forall_{i\in I}X_i\in B]\leqslant((1-\lambda)\sqrt{\beta}+\lambda)^{|I|}$$

(b) 得出结论：如果 $|\mathcal{B}|<n/100$ 且 $\lambda<1/100$，则存在 $I\subseteq[k]$ 使得 $|I|>k/10$ 且 $\forall_{i\in I}X_i\in\mathcal{B}$ 的概率至多为 $2^{-k/100}$。

(c) 利用(b)中的结论给出一个错误概率削减过程，它的输入是能用 m 个随机二进制位以 0.99 的概率判定语言 L 的任意 **BPP** 算法 A，它能将 A 转变为另一个能用 $m+O(k)$ 个随机二进制位以 $1-2^{-k}$ 的概率判定语言 L 的算法 B。

457

21.13　证明：在任意 d -正则的 n -顶点图中，均存在含 $n/2$ 个顶点的子集 S 使得 $E(S,\overline{S})\leqslant dn/4$。由此得出结论：当 $\rho>1/2$ 时，不存在 (n,d,ρ)-边扩张图。

21.14　证明扩张图平稳引理(引理 21.11)。

21.15　[Tan84]如果顶点度为 d 的 n -顶点图 G 使得 $|\Gamma(S)|\geqslant c|S|$ 的对任意不太大的顶点子集 S 成立，则称 G 的顶点扩张度为 c。(这里，$\Gamma(S)=\{u:u\in\overline{S}$ 且存在 $v\in S$ 使得 (u,v) 是 G 的一条边$\}$。“不太大”的含义是形如 $|S|\leqslant n/(10d)$ 的不等式成立。)本习题表明，当第二大特征值为 $\frac{2}{\sqrt{d}}(1+o(1))$ 时图的顶点扩张度大约为 $d/4$。[Kah92]证明了这种图的顶点扩张度实际上大约为 $d/2$，并且能够构造反例来说明该界限是紧的。相比之下，随机 d -正则图的顶点扩张度是 $1-o(1)$。

(a) 证明：如果 $\boldsymbol{p}$ 是一个概率向量，则 $\|\boldsymbol{p}\|_2^2$ 等于“根据 $\boldsymbol{p}$ 随机抽取的 i 和 j 满足 $i=j$”的概率。

(b) 证明：如果图 G 的随机游走矩阵为 A，S 是 G 的一个顶点子集，并且随机向量 $\mathbf{s}$ 表示 S 上的均匀分布，则 $\|A\mathbf{s}\|_2^2\geqslant 1/|\Gamma(S)|$，其中 $\Gamma(S)$ 表示 S 的相邻顶点构成的集合。

(c) 证明：如果 G 是 (n,d,λ)-扩张图，且 S 是含有 ϵn 个顶点的顶点子集，则

$$|\Gamma(S)|\geqslant\frac{|S|}{\lambda^2(1-\epsilon)+\epsilon}$$

21.16　设 G 是一个图而 S 是 G 的顶点子集，则收缩 S 的含义是如下地将图 G 转变为另一个图 H：将 G 中位于 S 的所有顶点替换为一个新顶点 s，并且为 G 中满足 $u\in S$ 的每条边 $\overline{uv}$ 在 H 中引入一条边 $\overline{sv}$。设 G 是一个 (n,d,ρ)-边扩张图。在 G 中取大小为 c 的 k 个互不相交的顶点子集 $S_1,\cdots,S_k$，收缩这些子集；然后如有必要则在其他顶点上引入自环以确保所得的图是正则的；将最后得到的图记为 H，它含有 $n'=n-(c-1)k$ 个顶点并且每个顶点的度为 cd。证明：H 是一个 $(n',cd,\rho/(2c))$-边扩张图。利用本结论完成定理 21.19 的证明。

21.17　证明：对于任意函数 $\mathrm{Ext}:\{0,1\}^r\to\{0,1\}^n$，均存在 $(n,n-1)$-随机源 X 和 $b\in\{0,1\}$ 使得 $\Pr[\mathrm{Ext}(X)_1=b]=1$，其中 $\mathrm{Ext}(X)_1$ 表示 $\mathrm{Ext}(X)$ 的第一个二进制位。证明上述结论表明了 $\Delta(\mathrm{Ext}(X),U_n)\geqslant 1/2$。

21.18　(a) 证明：存在下述的 $\mathrm{poly}(n)$ 时间概率型算法 A。A 的输入服从分布 X，其中 $H_\infty(X)\geqslant n^{100}$，并且它还可以通过黑盒访问函数 $f:\{0,1\}^n\to\{0,1\}$。A 的输出如下。如果 $E[f(U_n)]\geqslant 2/3$，则算法 A 至少以 0.99 的概率输出 1；如果 $E[f(U_n)]<1/3$，则算法 A 至少以 0.99 的概率输出 0。我们称这种算法是*函数近似器*(Function Approximator)。

(b) 证明：不存在多项式时间的确定型函数近似器 A 使得它无需使用其他随机输入(也就是说，不存在确定型函数近似器)。

(c) 证明：对于任意概率分布 X，如果 $\Delta(X, Y)>1/10$ 对满足 $H_{-}(Y)\geqslant n/2$ 的任意 Y 都成立，则不存在多项式时间函数近似器以 X 为输入。由此得出结论，要对 **BPP** 算法进行黑盒模拟必须使用最小熵较大的概率分布。 458

21.19　称分布 Y 是分布 Y_1，…，Y_N 的凸组合，如果存在和为 1 的非负实数 a_1，…，a_N 使得 Y 等同于如下分布：以 a_i 的概率随机选取 i，然后根据分布 Y_i 随机选取一个元素。证明：如果分布 Y 是分布 Y_1，…，Y_N 的凸组合，则对任意分布 X 都有

$$\Delta(X,Y)\leqslant\sum_i a_i\Delta(X,Y_i)\leqslant\max_i\Delta(X,Y_i).$$

21.20　假设布尔函数 f 是 (S,ε)-难解的。均匀随机地选择 x_1，x_2，…，x_m，并输出 $f(x_1)f(x_2)\cdots f(x_m)$。令 D 是 $f(x_1)f(x_2)\cdots f(x_m)$ 服从的 m 长位串的概率分布。证明：D 与长为 m 的位串上的均匀分布之间的统计距离至多为 εm。

21.21　证明引理 21.26。

21.22　设 A 是一个 $n\times n$ 矩阵，它的所有特征向量是 u^1，…，u^n，对应的特征值依次是 λ_1，…，λ_n。设 B 是一个 $m\times m$ 矩阵，它的所有特征向量是 v^1，…，v^m，对应的特征值依次是 a_1，…，a_m。证明：$A\otimes B$ 的所有特征向量都形如 $u^i\otimes v^j$，它对应的特征值是 $\lambda_i\cdot a_j$。

21.23　不利用对称矩阵可以对角化这一事实，证明：在任意两个图 G，G' 上，均有 $\lambda(G\otimes G')\leqslant\lambda(G)+\lambda(G')$。

21.24　设 G 是度为 D 的 n-顶点图，它具有 ρ-边扩张度，其中 $\rho>0$。(也就是说，对于满足 $|S|\leqslant n/2$ 的任意顶点子集 S，介于 S 和它的补集之间的边的条数至少为 $\rho D|S|$。)设 G' 是度为 d 的 D-顶点图，它具有 ρ'-边扩张度，其中 $\rho'>0$。证明：$G\textcircled{R}G'$ 的边扩张度至少是 $\rho^2\rho'^2/1000$。 459

第 22 章

Computational Complexity: A Modern Approach

PCP 定理的证明和傅里叶变换技术

常数的改进很多时候都可以如下地获得：从之前的协议[⊖]中提取某种重要的性质，然后将原来的协议作为黑盒来使用并添加一些概念上全新的构造。在某种程度上讲，我们的论文也将沿着这种思路来进行……长编码具有通用性，因为其他任何二进制编码都是它的子编码。因此，这种编码的存在性无可置疑，但让人感到意外的是这种铺张浪费的编码居然也十分有用。

——约翰·哈斯塔德(Johan Hästad)，1997

我们在第 11 章中已经看到，**PCP** 定理意味着很多优化问题的近似求解也是 **NP** 难的。本章完整地证明 **PCP** 定理。第 11 章还指出，**PCP** 定理不足以证明其他几个类似的结果，要证明这些结果，我们需要更强(或者直接说是不同)的"**PCP** 定理"。本章还将概述这些结果和它们的证明过程。两个重要的结果是莱斯(Raz)的并行重复定理(Parallel Repetition Theorem)(参见 22.3 节)和哈斯塔德(Hästad)的 3-位 **PCP** 定理(Three-Bit PCP Theorem)(定理 22.16)。莱斯定理得到了大规模字母表上的 2CSP 问题的强难度结论。哈斯塔德定理表明，**NP** 语言的证明只需查验其中的 3 个二进制位就可以被概率地验证。哈斯塔德定理的一个推论是，在任意 $\epsilon>0$ 上计算 MAX-3SAT 问题的$(7/8+\epsilon)$-近似解都是 **NP** 难的。由于人们已经知道 MAX-3SAT 问题的 7/8-近似解可以在多项式时间内求得(参见例 11.2 和习题 11.3)，这就表明在 **P**≠**NP** 的假设条件下 MAX-3SAT 问题的可近似性在 7/8 上由"容易近似"突变为"难以近似"。这样的结果通常称为阈值结果。人们只在其他少数几个问题上得到了阈值结论。

哈斯塔德定理以我们研究过的其他几个结果为基础，这些结果包括了(标准的)**PCP** 定理和莱斯定理。哈斯塔德定理还要使用哈斯塔德引入的一种方法，该方法利用傅里叶变换来分析验证者的接受概率。这种傅里叶分析在理论计算机科学的其他领域中也非常有用。我们将在 22.5 中介绍这种技术，并借助它来证明 11.5 节中给出的线性测试算法的正确性，这样就证明了 11.5 节给出的结论 **NP**⊆**PCP**(poly(n), 1)。然后，我们再利用傅里叶分析来证明哈斯塔德的 3-位 **PCP** 定理。

22.8 节证明 SET-COVER 问题的近似难度。22.2.3 节证明，计算 MAX-INDSET 问题的 $n^{1-\epsilon}$-近似解是 **NP** 难的。22.9 节简明扼要地概述人们已经证得的其他 **PCP** 定理，包括那些基于唯一性游戏假设(Unique Game Conjecture)的 **PCP** 定理。

460

22.1 非二进制字母表上的约束满足问题

本章将会经常使用 $q\text{CSP}_W$ 问题，它是定义 11.11 中的二进制字母表上的 qCSP 问题在大小为 W 的字母表上的推广。

定义 22.1 ($q\text{CSP}_W$)) 对于自然数 q，$W\geq 1$，$q\text{CSP}_W$ 问题的定义类似于定义 11.11

⊖ 这里主要论述的是交互式证明协议中的可靠性参数和完备性参数。——译者注

中 qCSP 问题的定义，只是它的字母表是$[W]=\{1, 2, \cdots, W\}$而不再是$\{0, 1\}$，因此它的每个约束都是从$[W]^q$到$\{0, 1\}$的映射。

对于 $\rho<1$，同二进制字母表上 ρ-GAPqCSP 的定义一样(参见定义 11.13)，我们也可以类似地定义诺言问题 ρ-GAPqCSP$_W$。

例 22.2 3SAT 是 qCSP$_W$ 在 $q=3$ 和 $W=2$ 时的特殊情况，其中的每个约束都是相关文字的 OR 操作。

类似地，**NP** 完全问题 3COL 也可以视为 2CSP$_3$的特殊情形，其中每条边(i, j)对应变量 u_i，u_j上的一个约束，该约束被满足当且仅当 $u_i \neq u_j$。图是 3-可着色的当且仅当所有变量在$\{0, 1, 2\}$中的赋值使得所有约束被满足。 ◀

22.2　PCP 定理的证明

本节证明 **PCP** 定理。我们给出迪纳尔(Dinur)的证明[Din06]，该证明将[AS92, ALM+92]中的原始证明简化了将近一半。22.2.1 节勾勒出证明的主要步骤。22.2.2 节给出关键的证明步骤—迪纳尔的鸿沟放大技术。22.2.5 节给出证明过程的另一个关键步骤，它源自 **PCP** 定理的原始证明[ALM+92]，11.5 节在证明 **NP**$\subseteq$**PCP**$(\log n, 1)$时已经交代了它的核心思想。

22.2.1　PCP 定理的证明思路

我们已经看到，**PCP** 定理等价于定理 11.14，该定理断言 ρ-GAPqCSP 对某个常数 q 和 $\rho<1$ 是 **NP** 难的。考虑 $\rho=1-\varepsilon$ 的情形，其中 ε 不一定是常数还可以是约束个数 m 的函数。由于能够被满足的约束的个数始终是一个正整数，故如果 φ 是不可满足的，则 $\mathrm{val}(\varphi)\leqslant 1-1/m$。于是，鸿沟问题$(1-1/m)$-GAP3CSP 是 3SAT 问题的推广并且是 **NP** 难的。**PCP** 定理的证明过程从上述观察开始，递归地证明当 ε 越来越大时$(1-\varepsilon)$-GAP3CSP 仍然是 **NP** 难问题，直到 ε 变为独立于 m 的某个绝对常数。这个过程可以形式化为下面的定义和引理。

461

定义 22.3 设函数 f 将 CSP 实例映射为 CSP 实例。我们称 f 是一个CL-归约(它是完全线性膨胀归约(Complete Linear-blowup Reduction)的简称)，如果 f 是多项式时间可计算的，并且它在任意 CSP 实例 φ 上均满足：

完全性：如果 φ 是可满足的，则 $f(\varphi)$也是可满足的；

线性膨胀：如果 m 是 φ 中约束的个数，则新的 qCSP 实例 $f(\varphi)$至多有字母表 W 上的 Cm 个约束，其中 C 和 W 都是依赖于 φ 的约束参数和字母表的大小(但不依赖于约束个数和变量总个数)。

引理 22.4 (**PCP** 主引理)　存在常数 $q_0\geqslant 3$，$\varepsilon_0\geqslant 0$ 和一个 CL-归约 f 使得，对于二进制字母表上的任意 q_0CSP 实例 φ 和任意 $\varepsilon<\varepsilon_0$，实例 $\psi=f(\varphi)$是(二进制字母表上的)一个满足如下条件的 q_0CSP 实例：

$$\mathrm{val}(\varphi)\leqslant 1-\varepsilon \Rightarrow \mathrm{val}(\psi)\leqslant 1-2\varepsilon$$

引理 22.4 可以繁凑地描述如下。

	约束参数	字母表	约束个数	值
原始值	q_0	二进制字母表	m	$1-\varepsilon$
	⇓	⇓	⇓	⇓
引理 22.4 得到的值	q_0	二进制字母表	Cm	$1-2\varepsilon$

引理 22.4 使得 *PCP* 定理的证明变得非常容易。

由引理 22.4 证明定理 11.5

令 $q_0 \geqslant 3$ 是引理 22.4 中的常数。正如前面的观察结果所述，判定问题 q_0CSP 是 **NP** 难问题。为了证明 **PCP** 定理，只需将 q_0CSP 问题归约到 GAPq_0CSP 问题。令 φ 是 q_0CSP 问题的一个实例，m 是 φ 中约束的个数。如果 φ 是可满足的，则 $\mathrm{val}(\varphi)=1$；否则，$\mathrm{val}(\varphi)\leqslant 1-1/m$。我们利用引理 22.4 来放大这个鸿沟。具体而言，将引理 22.4 得到的 f 在 φ 上递归地运用 $\log m$ 次，最后得到 ψ。注意，如果 φ 是可满足的，则 ψ 也是可满足的；但是，如果 φ 不是可满足的(进而 $\mathrm{val}(\varphi)\leqslant 1-1/m$)，则 $\mathrm{val}(\psi)\leqslant 1-\min(2\varepsilon_0, 1-2^{\log m}/m)=1-2\varepsilon_0$。注意，$\psi$ 的规模至多为 $C^{\log m}m$，这是 m 的多项式。因此，我们得到了一个从 q_0CSP问题到 $(1-2\varepsilon_0)$-GAPq_0CSP 问题的保持鸿沟的归约，因此 **PCP** 定理成立。 ■

本节的剩余部分通过结合两种变换来证明引理 22.4。第一种变换用于放大给定 CSP 实例的鸿沟(其中，鸿沟(Gap)指的是不被满足的约束总数占所有约束总数的比例)，但这将使得字母表的规模增大。第二种变换将字母表再变换回二进制字母表，但这将使得鸿沟略为减小。这两种变换由下面的两个引理给出。

引理 22.5（鸿沟放大[Din06]）　对于任意 l，$q\in\mathbf{N}$，存在数 $W\in\mathbf{N}$，$\varepsilon_0>0$ 和一个 *CL*-归约 $g_{l,q}$ 使得：对于二进制字母表上 qCSP 问题的任意实例 φ，实例 $\psi=g_{l,q}(\varphi)$ 的约束参数等于 2 并且所用的字母表的大小至多为 W，并且在任意 $\varepsilon\leqslant\varepsilon_0$ 上还满足

462

$$\mathrm{val}(\varphi)\leqslant 1-\varepsilon \Rightarrow \mathrm{val}(\psi)\leqslant 1-l\varepsilon$$

引理 22.6（字母表削减）存在常数 q_0 和一个 *CL*-归约 h 使得：对 CSP 的任意实例 φ，如果 φ 在非二进制字母表 $\{0,\cdots,W-1\}$ 上的约束参数等于 2，则 $\psi=h(\varphi)$ 是二进制字母表上约束参数等于 q_0 的 CSP 实例，并且它满足

$$\mathrm{val}(\varphi)\leqslant 1-\varepsilon \Rightarrow \mathrm{val}(h(\varphi))\leqslant 1-\varepsilon/3$$

引理 22.5 和引理 22.6 一起蕴含了引理 22.4，这只需令 $f(\varphi)=h(g_{\varepsilon,q_0}(\varphi))$。事实上，如果 φ 是可满足的，则 $f(\varphi)$ 也是可满足的。如果 $\mathrm{val}(\varphi)\leqslant 1-\varepsilon$ 对 $\varepsilon<\varepsilon_0$ 成立，其中 ε_0 是引理 22.5 在 $l=6$，$q=q_0$ 时得到的数，则 $\mathrm{val}(g_{\varepsilon,q_0}(\varphi))\leqslant 1-6\varepsilon$，进而 $\mathrm{val}(h(g_{\varepsilon,q_0}(\varphi)))\leqslant 1-2\varepsilon$。引理 22.5 和引理 22.6 的复合运用可以描述为下表。

	约束参数	字母表	约束个数	值
原始	q_0	二进制字母表	m	$1-\varepsilon$
	⇓	⇓	⇓	⇓
引理 22.5($l=6$，$q=q_0$)	2	W	Cm	$1-6\varepsilon$
	⇓	⇓	⇓	⇓
引理 22.6	q_0	二进制字母表	$C'Cm$	$1-2\varepsilon$

22.2.2　迪纳尔鸿沟放大：引理 22.5 的证明

要证明引理 22.5，我们需要给出一个函数 g 将 qCSP 问题的任意实例映射为较大的字母表 $\{0,\cdots,W-1\}$ 上 2CSP$_W$ 问题的一个实例，使得其中不能被满足的约束在所有约束中

所占的比例增加。从"证明验证"的观点(11.3 节)看，不能被满足的约束在所有约束中所占的比例仅涉及可靠性参数。因此，初看起来，我们的任务似乎仅仅是削减 **PCP** 验证者的"可靠性"参数，而这很容易根据(定理 11.5 后面的)评注 11.6 中第 5 项描述的方法来完成——重复验证者的操作 2 次(或更一般地，重复操作 k 次)。这种直接的想法的问题在于，**PCP** 验证者的操作重复运行 k 次之后所对应的 CSP 实例不是一个 2CSP 实例，而是一个约束参数为 2^k 的 CSP 实例，这是因为 **PCP** 验证者的判定过程依赖于证明中的 2^k 个量。在 22.3 节中，我们将看到另一种重复 **PCP** 验证者操作的方法——并行重复。它确实可以得到 2CSP 实例，但是它却使得 CSP 实例的规模大幅度增加。但是，这里我们需要一个 CL-归约，也就是说，CSP 实例的规模只能增长一个常数因子。设计这种 CL-归约的关键在于扩张图中的随机游走。由于扩张图中的随机游走具有独立的研究价值，22.2.3 节先单独描述它。

22.2.3　扩张图、随机游走和 INDSET 的近似难度

迪纳尔的证明用到了第 21 章所讨论的扩张图。这里，为了本章的需要，我们重述扩张图的要点。作为本章运用扩张图的示例，我们还用它来证明 MAX-INDSET 的近似难度。

第 21 章为任意正则图 G 定义了参数 $\lambda(G)\in[0,1]$(参见定义 21.2)。对于任意 $c\in(0,1)$，如果正则图 G 满足 $\lambda(G)\leqslant c$，则称之为 c-扩张图。如果 $c<0.9$，我们还省略称呼中的前缀 c 而直接称 G 是一个扩张图。注意，常数 0.9 的选取是任意的。正如第 21 章中定理 21.19 所证明的那样，对于任意 $c\in(0,1)$，均存在常数 d 和一个算法使得：在给定的 $n\in\mathbf{N}$ 上，算 463
法在 $\mathrm{poly}(n)$ 时间内输出一个 d-正则 n-顶点的 c-扩张图。

本章需要的主要性质是，对任意正则图 $G=(V,E)$ 和满足 $|S|\leqslant|V|/2$ 的每个顶点子集 $S\subseteq V$ 具有

$$\Pr_{(u,v)\in E}[u\in S,v\in S]\leqslant\frac{|S|}{|V|}\left(\frac{1}{2}+\frac{\lambda(G)}{2}\right) \tag{22.1}$$

习题 22.1 要求读者证明(22.1)式。我们要使用的另一个性质是 $\lambda(G^\ell)=\lambda(G)^\ell$ 对任意 $\ell\in\mathbf{N}$ 成立，其中图 G^ℓ 是图 G 的邻接矩阵的 ℓ 次幂所对应的图(亦即，G^ℓ 中的每条边对应于 G 中长度为 ℓ 的路径)。因此，(22.1)式蕴含了

$$\Pr_{(u,v)\in E(G^\ell)}[u\in S,v\in S]\leqslant\frac{|S|}{|V|}\left(\frac{1}{2}+\frac{\lambda(G)^\ell}{2}\right) \tag{22.2}$$

例 22.7　作为扩张图上随机游走的一个应用，我们证明 INDSET 的一个近似难度，它强于定理 11.15 给出的近似难度。具体而言，我们要证明的结论可以表述为下面的引理，由它立刻可以得到：在 n-顶点图上计算 MAX-INDSET 问题的 $n^{-\varepsilon}$-近似解是 **NP** 难的，因为 $m=\mathrm{poly}(n)$。(22.9.2 节概述了 MAX-INDSET 问题的更强的难度结果。)下面 $\overline{\alpha}(F)$ 表示图 F 中最大独立集的大小占图 F 的顶点数的比例。有趣的是，引理更加高效地实现了定理 11.15 中的"自我改进"思想。◀

引理 22.8　*对任意 $\lambda>0$，存在多项式时间可计算的一个归约 f 将 n-顶点图 F 映射为 m-顶点图 H 使得*

$$(\overline{\alpha}(F)-2\lambda)^{\log m}\leqslant\overline{\alpha}(H)\leqslant(\overline{\alpha}(F)+2\lambda)^{\log m}$$

证明　我们用随机游走更高效地定义推论 11.17 的证明过程中所用的"图乘积"。设 G 是一个 d-正则的 n-顶点扩张图，其中 d 是独立于 n 的常数，并令 $\lambda=\lambda(G)$。为记号上的方便起见，我们假设 G，F 的顶点集相同。我们如下地将 F 映射为具有 $nd^{\log m-1}$ 个顶

点的图 H：

- H 的每个顶点对应于 λ -扩张图 G 中的一条长度为 $(\log n-1)$ 的路径。
- 对于 H 的两个顶点 u，v，它们对应的路径分别记为 $(u_1, \cdots, u_{\log n})$ 和 $(v_1, \cdots, v_{\log n})$。如果图 F 在顶点集 $\{u_1, \cdots, u_{\log n}, v_1, \cdots, v_{\log n}\}$ 的任意两个顶点之间存在一条边，则在 u 和 v 之间为 H 引入一条边。

不难验证，对于 H 中的任意独立集，如果将 F 中出现在相应路径上的顶点全部取出来，则得到 F 的一个独立集。利用这一观察结果和习题 22.2 即可证得引理。 ■

22.2.4　迪纳尔鸿沟放大

如果 qCSP$_w$ 实例 φ 满足下列 3 个性质，则称它是“精细的”。

464 性质 1：φ 的约束参数 q 等于 2(但它的字母表可以是非二进制的)。

性质 2：如果将 φ 的约束图 G 定义如下：顶点集是 $[n]$，其中 n 是 φ 中所含变量的个数；φ 中依赖于变量 u_i，u_j 的每个约束在图 G 中对应一条边 (i, j)，图 G 可以包含平行边(或重边)和自环。那么，G 是一个 d -正则图(其中 d 是独立于字母表大小的常数)，并且任意顶点的关联边有一半是自环。

性质 3：φ 的约束图 G 是扩张图。亦即，$\lambda(G)\leqslant 0.9$。

可以证明，在证明引理 22.5 时，我们可以假设引理中的 CSP 实例 φ 是精细的而不失一般性。这是由于，存在一个比较简单的 CL -归约将任意 qCSP 实例变换为精细的实例。(参见 22.A 节，我们注意到，这个 CL -归约会将可靠性鸿沟缩减一个依赖于 q 的因子，但这个因子可以通过在下面的引理 22.9 中选择较大的 t 值来消除。)引理 22.5 证明的剩余部分将在精细的 2CSP 实例上使用“乘幂”操作。这个操作可以用下面的引理来描述。

引理 22.9（乘幂）　存在算法将满足性质 1 至性质 3 的任意 2CSP$_w$ 实例 φ 转换为另一个 2CSP 实例 φ'(其中 $t\geqslant 1$ 是整数)使得

1. φ' 是如下的 2CSP$_{w'}$ 实例：$W'<W^{d^t}$ 并且 φ' 含有 $d^{(1+\sqrt{t})t}n$ 个约束，其中 d 是 φ 的约束图的顶点度而 n 是 φ 中变量的个数。

2. 如果 φ 是可满足的，则 φ' 也是可满足的。

3. 对于任意 $\varepsilon<\dfrac{1}{d\sqrt{t}}$，如果 $\mathrm{val}(\varphi)\leqslant 1-\varepsilon$，则 $\mathrm{val}(\varphi')\leqslant 1-\varepsilon'$，其中 $\varepsilon'=\dfrac{\sqrt{t}}{10^3 dW^t}\varepsilon$。

4. 由 φ 计算 φ' 可以在 m 和 W^{d^t} 的多项式时间内完成。

证明　设 φ 是 n 个变量上具有 $m=nd$ 个约束的 2CSP$_w$ 实例，同前面一样用 G 表示 φ 的约束图。要证明引理 22.9，我们先说明由 φ 如何构造 φ'。构造过程的主要思想是利用具有某种长度的随机游走来实现约束图的“乘幂”操作。

φ' 的构造

公式 φ' 中的变量个数与 φ 中的变量相同，但 φ' 的变量 $\boldsymbol{y}=y_1, \cdots, y_n$ 在大小为 $W'<W^{d^t}$ 的字母表上取值。于是，每个新变量 y_i 的取值都是 $\{0, \cdots, W'-1\}$ 上的一个 d^t 元组。为了避免在后面的证明中引起混淆，我们仅将这些新变量称为“变量”，而将 φ 中的变量都称为“老变量”。

我们可以这样认为，变量 y_i 的取值给出了如下的每个老变量 u_j 在 $\{0, \cdots, W-1\}$ 上的一个取值：顶点 j 和顶点 i 之间在图 G 中存在一条长度至多为 $t+\sqrt{t}$ 的路径（参见图 22-1）。也就是说，对于以 i 为中心以 $t+\sqrt{t}$ 为半径的球体中的所有 j，y_i 的取值为变量 u_j 给出了一个取值。由于 G 是 d -正则图，该球体中所含的结点 j 至多有 $d^{t+\sqrt{t}+1}$ 个，而 $d^{t+\sqrt{t}+1}$ 小于 $d^{|V|}$。因此，y_i 的取值可以用大小为 W' 的字母表来编码。

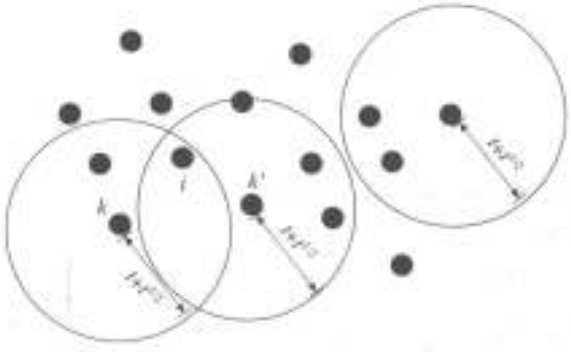


图 22-1　CSP 实例 ψ' 的 n 个变量上都在大小至多为 $W^{d^{t+\sqrt{t}+1}}$ 的字母表上取值。新变量 y_i 的一个赋值表示了 ψ 中的一组老变量的赋值，这组老变量在 ψ 的约束图中对应的结点全部位于以 i 为中心以 $t+\sqrt{t}$ 为半径的球体中。如果结点 j 同时位于分别以 i 和 i' 为中心以 $t+\sqrt{t}$ 为半径的球中，则 y_i 和 $y_{i'}$ 可能为 u_j 申明不同的取值。从这个意义上讲，y_1，…，y_n 的赋值可能是不一致的

下面，我们通常会说 y_i 的赋值"申明"了老变量 u_j 的一个取值。当然，另一个变量 $y_{i'}$ 也可能会申明 u_j 的另一个不同的取值。正是所有 u_j 取值的潜在不一致性使得引理证明的其余部分是非平凡的。事实上，证明的任务正是要设计 2CSP_W 实例 ψ' 的各个约束使得这种不一致性得以显露出来。

对于图 G 中长度为 $2t+1$ 的任意路径 $p=(i_1, \cdots, i_{2t+2})$，我们在 ψ' 中引入一个约束 C_p（参见图 22-2）。约束 C_p 将只依赖于变量 y_{i_1} 和 $y_{i_{2t+2}}$，因此，我们所构造的是一个 2CSP_W 实例。C_p 为假当且仅当存在某个 $j\in[2t+1]$ 使得

1. i_j 位于以 i_1 为中心以 $t+\sqrt{t}$ 为半径的球体中。

2. i_{j+1} 位于以 i_{2t+2} 为中心以 $t+\sqrt{t}$ 为半径的球体中。

3. 如果 w 表示 y_{i_1} 为老变量 u_{i_j} 申明的值而 w' 表示 $y_{i_{2t+2}}$ 为老变量 $u_{i_{j+1}}$ 申明的值，则序对 (w, w') 不满足 ψ 中同时包含老变量 u_{i_j} 和 $u_{i_{j+1}}$ 的约束。

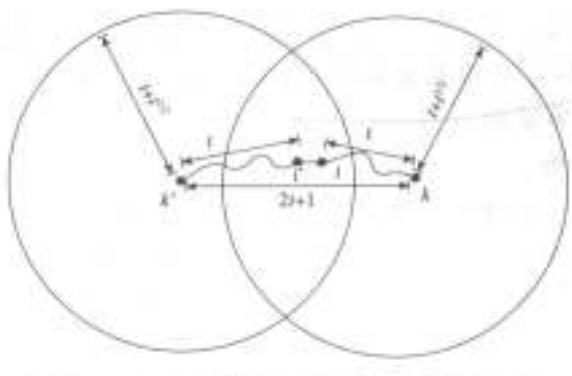


图 22-2　对于 ψ 的约束图中的每一条长度为 $2t+1$ 的路径，ψ' 中都有一个约束。该约束用于查验在分别以路径的两个端点为中心以 $t+\sqrt{t}$ 为半径的球体中的两个相继顶点对应的变量在 ψ 的约束上是不是不一致的

下面依次给出几个观察结果。首先，2CSP_W 实例 ψ' 的构造可以在 m 和 w^{d^t} 的多项式时间内完成，因此引理 22.9 的第 4 部分可以直接得到。其次，对于老变量 u_1，u_2，…，u_n 的任意赋值，我们可以按照如下的标准方式将它"升级"为 y_1，…，y_n 的一个标准赋值：对于变量 y_i，用"以 i 为中心以 $t+\sqrt{t}$ 为半径的球体中所有 j"对应的所有变量 u_j 构成的向量来对 y_i 赋值。如果 u_1，u_2，…，u_n 的赋值满足 ψ，则按这种标准形式得到的赋值也满足 ψ'，这是由于该赋值满足图 G 中长度为 $2t+1$ 的每条路径对应的约束。因此，引理 22.9 的第 2 部分也可以直接得到。剩下的只需证明引理的第 3 部分，这是整个证明最困难的部分。我们需要证明，如果 $\text{val}(\psi)\leqslant 1-\varepsilon$，则 y_1，…，y_n 的任意赋值都至

多只能满足 ψ' 中 $1-\varepsilon'$ 比例个约束，其中 $\varepsilon<\dfrac{1}{d\sqrt{t}}$ 而 $\varepsilon'=\dfrac{\sqrt{t}}{10^5 dW^5}\varepsilon$。

相对多数赋值

为了证明引理 22.9 的第 3 部分，我们先说明如何将 ψ' 的一个赋值 $\boldsymbol{y}$ 转换为 ψ 的一个赋值 $\boldsymbol{u}$，然后再论证：如果 ψ 中只有（比例为 ε 的）"少量"约束不能被 $\boldsymbol{u}$ 满足，则 ψ' 中将有（比例为 $\varepsilon'=\Omega(\sqrt{t}\varepsilon)$ 的）"很多"约束不能被 $\boldsymbol{y}$ 满足。

从现在起，我们在 ψ' 的所有变量上取定一个任意的赋值 $\boldsymbol{y}=y_1, \cdots, y_n$。正如前面指出的那样，各个 y_i 的值相互之间可能是不一致的，因此不能显而易见地得到各个老变量 u_i 的值。要从 $y_1, \cdots, y_n$ 中提取各个老变量 u_i 的单一取值，下面的概念非常关键。

对于 ψ 的每个变量 u_i，我们定义取值于 $\{0, \cdots, W-1\}$ 的随机变量 Z_i 来表示如下过程得到的结果：在 G 上从顶点 i 出发进行 t 步随机游走到达顶点 k，输出 y_k 为 u_i 申明的值。在 Z_i 的所有取值中，令 z_i 是概率最大的取值。如果两个取值的概率都达到最大，则 z_i 可以取其中任意一个。我们称赋值 $z_1, \cdots, z_n$ 是相对多数赋值（Plurality Assignment），参见图 22-3。注意，$Z_i=z_i$ 的概率至少是 $1/W$。



图 22-3　ψ' 的一个赋值 $\boldsymbol{y}$ 如下诱导产生 ψ 的一个相对多数赋值 $\boldsymbol{u}$。考虑在 ψ 的约束图上从顶点 i 出发进行 t 步随机游走能够达到的所有顶点 k，在 y_k 为 u_i 申明的所有值中，取概率最大者作为 u_i 的赋值

由于 $\mathrm{val}(\psi)\leqslant 1-\varepsilon$，故 ψ 的任意赋值至少使得 ε 比例的约束不能被该赋值满足。因此，任意的相对多数赋值也至少使得 ψ 中 ε 比例的约束不能被满足。因此，存在 ψ 的 $\varepsilon m=\varepsilon nd/2$ 个约束构成的子集 F 使得 $\boldsymbol{x}=z_1, \cdots, z_n$ 不满足 F 中的任意一个约束。下面，我们将利用集合 F 来证明：ψ' 中将有至少 $\varepsilon'=\Omega(\sqrt{t}\varepsilon)$ 比例的约束不能被 $\boldsymbol{y}$ 满足。

467

分析

后续的证明将在如下的概率空间中定义随机事件：在 G 中长度为 $2t+1$ 的所有路径中，随机选择一条路径 $(i_1, \cdots, i_{2t+2})$（这相当于在 ψ' 的所有约束中随机选择一个约束）。对于 $j\in\{1, 2, \cdots, 2t+1\}$，路径 $(i_1, \cdots, i_{2t+2})$ 中的第 j 条边 (i_j, i_{j+1}) 称为真确的，如果 y_{i_j} 为老变量 u_{i_j} 申明了相对多数取值并且 $y_{i_{j+1}}$ 为老变量 $u_{i_{j+1}}$ 申明了相对多数取值。注意，如果路径 $(i_1, \cdots, i_{2t+2})$ 有一条真确边并且这条边在 ψ 中对应的约束属于集合 F，则根据 F 的定义可知，路径 $(i_1, \cdots, i_{2t+2})$ 在 ψ' 中对应的约束不能被 $\boldsymbol{y}$ 满足。我们的目标是证明：在所有的路径中，比例至少为 ε' 的路径中将包含这样的真确边。

整个证明过程由下列几个论断完成。

论断 22.10　对于 G 的每条边 e 和任意的 $j\in\{1, 2, \cdots, 2t+1\}$，

$$\Pr[e \text{ 是路径} (i_1, \cdots, i_{2t+2}) \text{ 的第 } j \text{ 条边}]=\frac{1}{|E|}=\frac{2}{nd}$$

证明　容易验证，在 d-正则图中，如果我们随机选择顶点 i_1，然后从 i_1 出发进行 $2t+1$ 步随机游走，则随机游走经过的第 j 条边就是随机取值 G 的一条边。 ■

下一个论断证明，粗略地位于路径中部的边（具体地讲，位于中部大小为 $\delta\sqrt{t}$ 的区间内）很有可能是真确的。

论断 22.11　令 $\delta<\frac{1}{100W}$，对于 G 的每条边 e 和任意的 $j\in\{t, t+1, \cdots, t+\delta\sqrt{t}\}$，

$$\Pr[(i_1,\cdots,i_{2t+2}) \text{ 的第 } j \text{ 条边是真确边} \mid e \text{ 是路径} (i_1,\cdots,i_{2t+2}) \text{ 的第 } j \text{ 条边}]\geqslant\frac{1}{2W^2}$$

证明　主要的直观认识是，由于 G 有一半的边都是自环，因此长度属于 $[t-\delta\sqrt{t}, t+\delta\sqrt{t}]$ 的随机游走从统计角度看非常类似于长度为 t 的随机游走。

正式地讲，用另一种稍微不同的观点来观察所选择的路径，就可以得到论断的证明。根据前一个论断可知，第 j 步使用边 $e=(i_j, i_{j+1})$ 且长度为 $2t+1$ 的任意随机游走都是由两个随机游走拼接而成的，其中一个随机游走从 i_j 出发进行 j 步游走，另一个随机游走则从 i_{j+1} 出发进行 $2t-j$ 步游走（当然，这两个随机游走是随机独立的）。用 i_1 和 i_{2t+2} 分别表示这两个随机游走的另一个端点。这样，我们需要证明

$$\Pr[(y_{i_1} \text{ 申明了 } u_{i_1} \text{ 的相对多数取值}) \wedge (y_{i_{2t+2}} \text{ 申明了 } u_{i_{j+1}} \text{ 的相对多数取值})]\geqslant\frac{1}{2W^2} \tag{22.3}$$

由于相对多数赋值是用长度恰好等于 t 的路径来定义的，故如果 j 恰好等于 t，则 (22.3) 式左端将至少为 $\frac{1}{W}\times\frac{1}{W}=\frac{1}{W^2}$。（得出这个结论的关键在于抵达 y_{i_1} 和 $y_{i_{2t+2}}$ 的两个随机游走是相互独立的。）

然而，论断中的 j 在集合 $\{t, t+1, \cdots, t+\delta\sqrt{t}\}$ 中变化，因此这两个随机游走的长度都介于 $t-\delta\sqrt{t}$ 和 $t+\delta\sqrt{t}$ 之间。此时，我们也能对任意 j 证明 (22.3) 式左端的取值跟 $1/W^2$ 相差无几。 468

由于每个顶点的关联边都有一半是自环，故我们可以如下地看待始于顶点 i 且长度为 l 的随机游走：(1) 投掷 l 枚硬币，将头面朝上的硬币的个数记为 S_l；(2) 在图上“真正”地随机游走（亦即，不使用自环）S_l 个步骤。注意，l 枚硬币中头面朝上的个数 S_l 服从二项分布。

可以证明，对于任意 t 和 δ，S_t 和 $S_{t+\delta t}$ 之间的统计距离至多为 10δ（参见习题 22.3）。换句话说，

$$\frac{1}{2}\sum_m|\Pr[S_t=m]-\Pr[S_{t+\delta t}=m]|\leqslant 10\delta$$

由此可知，从边 e 出发随机游走 t 步后所到达的顶点所服从的分布与从 e 出发随机游走 $t+\delta\sqrt{t}$ 步后所到达的顶点所服从的分布之间的统计距离也很近；同样，从边 e 出发随机游走 t 步后所到达的顶点所服从的分布与从 e 出发随机游走 $t-\delta\sqrt{t}$ 步后所到达的顶点所服从的分布之间的统计距离也很近。因此，(22.3) 式左端的表达式至少为

$$\left(\frac{1}{W}-10\delta\right)\left(\frac{1}{W}-10\delta\right)\geqslant\frac{1}{2W^2}$$

这就完成了对论断的证明。 ■

现在，我们用随机变量 V 表示路径 $(i_1, \cdots, i_{2t+2})$ 中间的 $\delta\sqrt{t}$ 条边中属于集合 F 的真确边[⊖]的条数。由于只要 $(i_1, \cdots, i_{2t+2})$ 中包含一条这样的边即可表明该路径在 ψ 中对应的

⊖ 这里，“属于 F 的真确边”指的是真确边端点表示的变量在 ψ 中对应的约束属于集合 F。——译者注

约束不能被 y 满足，故我们的目标是证明 $\Pr[V>0]\geqslant\varepsilon'$。

前面的两个引理表明，在路径$\langle i_1, \cdots, i_{2t+2}\rangle$中间的 $\delta\sqrt{t}$ 条边中，每条边是属于 F 真确边的概率为 $\frac{|F|}{|E|}\times\frac{1}{2W^2}$。于是，由数学期望的线性性质可知

$$E[V]\geqslant\delta\sqrt{t}\times\frac{|F|}{|E|}\times\frac{1}{2W^2}=\frac{\delta\sqrt{t}\varepsilon}{2W^2}$$

这说明 $E[V]$比较大，但这还不能说明引理成立，因为数学期望比较大时 V 仍有可能在大多数随机路径上都等于 0。为了排除这种情况，我们考虑 V 的二阶矩。这是整个证明过程中唯一用到"约束图是扩张图"这一事实的地方。

论断 22.12　$E[V^2]\leqslant 30\varepsilon\delta\sqrt{t}d$。

证明　令随机变量 V' 表示路径$\langle i_1, \cdots, i_{2t+2}\rangle$中间的 $\delta\sqrt{t}$ 条边中属于 F 的边的条数。由于，V 表示路径$\langle i_1, \cdots, i_{2t+2}\rangle$中间的 $\delta\sqrt{t}$ 条边中属于 F 的真确边的条数，故 $V\leqslant V'$。于是，只需证明 $E[V'^2]\leqslant 30\varepsilon\delta\sqrt{t}d$。为此，我们利用扩张图的平稳性和 F 含有约束图中 ε 比例的边这一事实。

具体地讲，对于 $j\in\{t, t+1, \cdots, t+\delta\sqrt{t}\}$，令 I_j 是一个示性随机变量，如果路径$\langle i_1, \cdots, i_{2t+2}\rangle$的第 j 条边属于 F，则 I_j 取 1，否则，I_j 取 0。于是，$V'=\sum_{j\in\{t,t+1,\cdots,t+\delta\sqrt{t}\}} I_j$。令 S 是 F 中所有约束在约束图 G 中对应的边的所有端点构成的集合，于是 $|S|/n\leqslant d\varepsilon$[⊖]。

469

$$\begin{aligned}
E[V'^2]&=E\Big[\sum_{i,j} I_iI_j\Big]\\
&=E\Big[\sum_j I_j^2\Big]+E\Big[\sum_{i\neq j} I_iI_j\Big]\\
&=\varepsilon\delta\sqrt{t}+E\Big[\sum_{j\neq j'} I_jI_{j'}\Big] && \text{（数学期望的线性性质和论断 22.10）}\\
&=\varepsilon\delta\sqrt{t}+2\sum_{j<j'}\Pr[\text{（第 }j\text{ 条边属于 }F\text{）}\wedge\text{（第 }j'\text{ 条边属于 }F\text{）}]\\
&\leqslant\varepsilon\delta\sqrt{t}+2\sum_{j<j'}\Pr[\text{（路径的第 }j\text{ 个顶点属于 }S\text{）}\wedge\text{（路径的第 }j'\text{ 个顶点属于 }S\text{）}]\\
&\leqslant\varepsilon\delta\sqrt{t}+2\sum_{j<j'}\varepsilon d(\varepsilon d+\lambda(G)^{j'-j}) && \text{（利用(22.2)式）}\\
&\leqslant\varepsilon\delta\sqrt{t}+2\varepsilon^2\delta\sqrt{t}d^2+2\varepsilon\delta\sqrt{t}d\sum_{k\geqslant 1}(\lambda(G))^k\\
&\leqslant\varepsilon\delta\sqrt{t}+2\varepsilon^2\delta\sqrt{t}d^2+20\varepsilon\delta\sqrt{t}d && \text{（利用 }\lambda(G)\leqslant 0.9\text{）}\\
&\leqslant 30\varepsilon\delta\sqrt{t}d && \text{（利用引理 22.9 的假设 }\varepsilon<\frac{1}{d\sqrt{t}}\text{）}\blacksquare
\end{aligned}$$

最后，由于 $\Pr[V>0]\geqslant E[V]^2/E[V^2]$对任意非负的随机变量都成立(参见习题 22.4)，故我们得到 $\Pr[V>0]\geqslant\frac{\sqrt{t}}{10^3dW^2}\varepsilon=\varepsilon'$。引理 22.9 得证。 ■

⊖ 原文对 S 的定义是"Let S be the set of vertices that have at least one end point in F"，句子的含义不甚明了，译文将它定义得更明确。由于 $|F|=\varepsilon nd/2$，故 F 中所有约束对应 G 中 $\varepsilon nd/2$ 条边。每条边至多有两个端点(自环除外)，故 $|S|\leqslant 2\varepsilon nd/2=\varepsilon nd$。亦即 $|S|/n\leqslant d\varepsilon$。——译者注

22.2.5　字母表削减：引理 22.6 的证明

有趣的是，引理 22.6 的证明过程的主要部分仍然是 11.5 节中介绍的具有指数规模的概率可验证明(**PCP**)。习题 22.5 讨论了引理的另一种证明。

设 φ 是引理 22.6 中所述的 2CSP 实例，它的 n 个变量是 u_1，…，u_n，它的字母表是 $\{0,\cdots,W-1\}$ 并且它的 m 个约束是 C_1，C_2，…，C_m。将每个变量的取值都视为是 $\{0,1\}^{\log W}$ 中的一个位串。约束 C_i 包含的两个变量(不妨设)为 u_j，u_k，则约束 C_i 可以视为一个线路，它的输入是 u_j，u_k 所表示的位串，而且约束 C_i 被满足当且仅当该线路输出 1。当 C_i 取遍所有约束时，将这种线路的规模的上界记为 l。注意，l 至多不超过 $2^{2\log W}<W^3$。不失一般性，我们假设所有的线路具有相同的规模。

削减字母表的思想是，将上面得到的每个线路改写为一个规模较小的 CSP 实例，并将每个老变量替换为一组新变量。[AS92]将这种技术称为验证者的复合(Verifier Composition)。最近，这种技术的一种变形被人们称作 **PCP** 的逼近(PCP of Proximity)。这两个术语都源于概率可验证明(**PCP**)的"证明验证"观点(参见 11.2 节)。我们先将结论表述为定理 11.9 的一个直接推论，然后再用 CSP 观点来表述它。 470

推论 22.13　(**PCP** 的逼近(PCP of Proximity))　*存在一个以输入长度为 $2k$ 且规模为 l 的任意线路 C 为输入的验证者 V 满足*

1. *如果长度都为 k 的两个位串 u_1，u_2 使得 $u_1\circ u_2$ 是线路 C 的满足性赋值，则存在长度为 $2^{\mathrm{poly}(l)}$ 的位串 π_3 使得 V 以概率 1 接受位串 $\mathrm{WH}(u_1)\circ\mathrm{WH}(u_2)\circ\pi_3$。*

2. *对于任意的 3 个位串 π_1，π_2，π_3，其中 π_1，π_2 的长度为 2^k，如果 V 至少以 1/2 的概率接受 $\pi_1\circ\pi_2\circ\pi_3$，则 π_1，π_2 将分别 0.99-接近于 $\mathrm{WH}(u_1)$，$\mathrm{WH}(u_2)$，其中 u_1，u_2 是某两个"长度为 k 且使得 $u_1\circ u_2$ 是线路 C 的满足性赋值"的位串。*

3. *V 的时间复杂度是 $\mathrm{poly}(l)$，它使用 $\mathrm{poly}(l)$ 个随机位并且仅查验输入位串中的 $O(1)$ 个位。*

在证明推论 22.13 之前，我们先说明如何用它来按要求削减字母表。首先，我们注意到，如果用约束满足(CSP)的观点看推论 22.13，(参见表 11.1)，则变量对应于 π_1，π_2，π_3 中的各个位而验证者则对应于这些变量上大小为 $2^{\mathrm{poly}(l)}$ 的 CSP 实例。约束参数是验证者需要从证明中读取的二进制位的个数，它是独立于 W 和 ε 的固定常数。被满足的约束在所有约束中所占的比例则是验证者的接受概率。

回过头来考虑字母表有待削减的 CSP 实例，我们将取值于字母表 $\{0,\cdots,W-1\}$ 的每个原始变量 u_i 替换为 2^W 个取值于 $\{0,1\}$ 的变量构成的序列 $U_i=(U_{i,1},\cdots,U_{i,2^W})$，这组变量的一个有效的赋值构成 u_i 的取值的沃尔什-哈达玛编码。对于每个老约束 $C_i(u_j,u_k)$，我们从约束满足的观点来运用推论 22.13，将 C_i 视为一个线路，它的赋值正有待于我们进行验证。于是，对于每个原始的约束 C_i，我们有由 $2^{\mathrm{poly}(l)}$ 个取值于 $\{0,1\}$ 的新变量构成的向量 Π_i，它在推论 22.13 中扮演 π_3 的角色，而 U_j 和 U_k 则分别扮演 π_1，π_2 的角色。C_i 对应的新约束族记为 $\mathcal{C}_i$。正如前面所注意到的那样，新约束中的约束参数是独立于 W 和 ε 的固定常数。

整个新 CSP 实例是所有新约束族的并集 $\bigcup_{i=1}^{m}\mathcal{C}_i$，参见图 22-4。显然，如果老的 CSP 实例是可满足的，则这个新的 CSP 实例也是可满足的。现在证明，如果某个赋值满足新 CSP 实例中超过 $1-\varepsilon/3$ 比例的新约束，则我们可以构造原始实例的一个赋值使得它能满

足其中 $1-\varepsilon$ 比例的老约束。这可以运用如下的规则将每个新变量族 U_i 的赋值进行"解码"来完成：如果 U_i 与某个线性函数 $\mathrm{WH}(a_i)$ 是 0.99 -接近的，则用 a_i 作为老变量 u_i 的赋值；否则，用任意的位串作为 u_i 的赋值。现在，看看任意老约束 $C_i(u_i, u_j)$ 是否被满足。如果 U_i，U_j 的解码结果 a_i，a_j 不满足 C_i，则推论 22.13 意味着 C_i 中至少有一半的约束不能被满足。于是，如果 δ 表示未被满足的老约束在所有老约束中所占的比例，则 $\delta/2\leqslant\varepsilon/3$，亦即 $\delta<2\varepsilon/3$。由此证得引理 22.6。

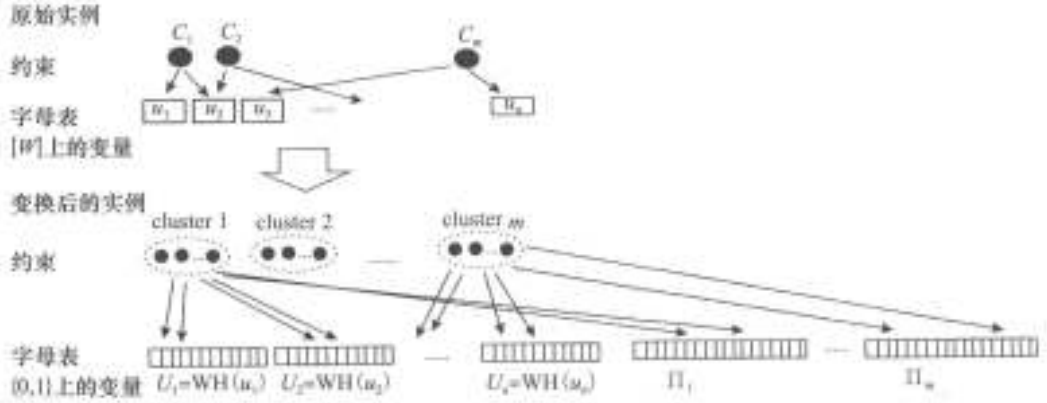


图 22-4　字母表削减变换将字母表 $\{0,\cdots,W-1\}$ 上的 2CSP 实例 φ 映射为二进制字母表上的一个 qCSP 实例 ψ。φ 的每个变量被映射为一组二进制变量，这组变量的赋值恰好是原变量的取值的沃尔什哈达玛编码。φ 中依赖于变量 u_i，u_j 的一个约束 C_t 被映射为一族约束，这族约束是原约束的 **PCP** 逼近。这族约束除依赖于 u_i，u_j 对应的所有新变量之外，还另外依赖于一族辅助变量，这族辅助变量的赋值包含了"能够表明 u_i，u_j 满足约束 C_t"的 **PCP** 逼近证明

因此，证明了推论 22.13 也就完成了引理 22.6 的整个证明。

推论 22.13 的证明　证明过程将用到从 CKT-SAT 到 QUADEQ 的归约（参见 11.5.2 节和习题 11.15）。该归约将具有 l 个输入位的线路 C 变换为 l 个变量上的 $O(l)$ 个方程构成的 QUADEQ 实例，其中 QUADEQ 实例的每个变量对应于线路的一个输入位。因此，
471 QUADEQ 实例的任意解恰由 l 个位构成，其前 k 位恰好是线路 C 的满足性赋值。

验证者希望 π_3 包含了定理 11.19 的验证者在"QUADEQ 实例的证明"中希望看到的所有信息，亦即沃尔什-哈达玛编码码字 $\mathrm{WH}(w)$ 对应的线性函数 f 和码字 $\mathrm{WH}(w\otimes w)$ 对应的线性函数 g，其中 w 满足 QUADEQ 实例。验证者采用定理 11.19 的证明中给出的过程来验证 f 和 g。

然而，在当前的情况中，验证者还有两个输入位串 π_1，π_2，它们也可以视为两个函数 π_1，π_2：$\mathrm{GF}(2)^k\to\mathrm{GF}(2)$。验证者验证它们分别 0.99 -接近于两个线性函数（不妨记为 $\tilde{\pi}_1$，$\tilde{\pi}_2$）。然后再验证 $\tilde{f}$ 所编码的位串的前 $2k$ 个分别与 $\tilde{\pi}_1$，$\tilde{\pi}_2$ 编码的位串相同。为此，验证者利用沃尔什-哈达玛编码的性质来执行如下的拼接测试。

拼接测试

我们给定了三个线性函数 π_1，π_2 和 f，它们分别编码了长度为 k，k 和 l 的位串。用 $\boldsymbol{u}$ 和 v 分别表示 π_1，π_2 所编码的位串，亦即，$\pi_1=\mathrm{WH}(\boldsymbol{u})$ 且 $\pi_2=\mathrm{WH}(v)$。用 w 表示 f 所编码的位串，则我们需要通过仅查验这三个函数中的 $O(1)$ 个位来验证 $\boldsymbol{u}\cdot v$ 确实是 w 的前 $2k$ 个位。根据随机子和原理，如果 $\boldsymbol{u}\cdot v$ 不是 w 的前 $2k$ 个位，则下述过程将至少以 1/2 的概率拒绝 π_1，π_2 和 f：随机选取 $\boldsymbol{x}$，$\boldsymbol{y}\in\{0,1\}^k$，用 $\boldsymbol{XY}\in\mathrm{GF}(2)^l$ 表示前 k 个位等于 $\boldsymbol{x}$ 且

接下来的 k 个位等于 y 而其余位全等于 0 的位串，如果 $f(XY)=\pi_1(x)+\pi_2(y)$，则接受，否则拒绝。 ■

22.3　2CSP$_W$的难度：鸿沟和字母表大小之间的平衡

2CSP$_W$问题经常出现在各种高级 **PCP** 定理的证明中。(标准的)**PCP** 定理表明，存在常数 W 和常数 $v<1$ 使得计算 2CSP$_W$问题的 v-近似解是 **NP**-难的(参见定义 22.1)。 472

推论 22.14（**PCP** 定理的推论）　*存在常数 $v<1$ 和常数 W 使得 v-GAP2CSP$_W$是* **NP** *难的。*

在高级 **PCP** 定理中，我们希望证明推论 22.14 的结论在 v 较小但 W 却不太大的情况下仍成立。(注意，如果允许 W 取 $\exp(n)$，则即使在 $v=0$ 的情况下 v-GAP2CSP$_W$也是 **NP** 难的!)乍一看，用于"鸿沟放大"的引理 22.5 似乎就能用来证明上述结论，但事实上这却行不通。原因有两点。首先，引理 22.5 不允许 v 小于某个固定的常数；其次，引理 22.5 大幅度地增大了字母表的大小。下面的定理通过常数 c 实现了鸿沟和字母表大小这两个常数之间的最佳平衡。在进一步构造概率可验证明时，一种有益的做法通常是，只限制在某些 2CSP 实例构成的一个特殊子类上进行讨论，这些实例满足所谓的投影性质(Projection Property)。也就是说，对于任意的约束 $\varphi_i(y_1, y_2)$和 y_1的任意取值，存在 y_2的唯一取值使得 $\varphi_i(y_1, y_2)=1$。投影性质还可以表述为，对于任意约束 φ_i，存在函数 $h_i:[W]\to[W]$ 使得：约束 φ_i被(u, v)满足当且仅当 $h(u)=v$。

2CSP 问题的一个实例称为是正则的，如果它的每个变量都出现在同样数量的约束中。

定理 22.15（莱斯[Raz95]）　*存在 $c>1$ 使得在任意 $t>1$ 时 ε-GAP2CSP$_W$都是* **NP** *难的，其中 $\varepsilon=2^{-t}$，$W=2^t$。在具有投影性质的正则 2CSP 实例上，上述结论也成立。*

费格(Feige)和吉莉安(Kilian)[FK93]证明了定理 22.15 的较弱形式，其证明过程比定理 22.15 的证明过程简单。这种较弱的形式对很多应用而言已经足矣，包括哈斯塔德3-位 **PCP** 定理(参见 22.4 节)。

22.3.1　莱斯的证明思想：并行重复

设 φ 是推论 22.14 的归约产生的 2CSP$_W$实例。在某个 $v<1$ 上，φ 具有如下性质：要么 $\mathrm{val}(\varphi)=1$，要么 $\mathrm{val}(\varphi)=v<1$，但要判定具体是哪种情形成立则十分困难。用一种显而易见的"乘幂"思想可以削减鸿沟并且保持约束参数 2 不变。令 φ''表示如下的实例。它的每个变量都是 φ 中变量构成的 t 元组，它的每个约束也对应 φ 中约束构成的一个 t 元组。也就是说，对于 φ 中的 t 个约束构成的 t 元组 $\varphi_1(y_1, z_1)$，$\varphi_2(y_2, z_2)$，…，$\varphi_t(y_t, z_t)$，在新实例中对应一个约束参数为 2 的约束，它的两个新变量分别是$(y_1, y_2, \cdots, y_t)$和$(z_1, z_2, \cdots, z_t)$并且这个新约束被描述为布尔函数

$$\bigwedge_{i=1}^{t}\varphi_i(y_i, z_i)$$

(用自然语言来讲就是，新约束被满足当且仅当构成它的 t 个老约束被满足。)

用证明验证的观点看，新 2CSP 实例对应于 t 个验证者的并行执行。因此，莱斯定理也被称为并行重复定理。

φ 的满足性赋值很容易组织成一些 t 元组使得它们构成 φ''的满足性赋值。而且，

给定 φ 的一个赋值，如果它满足 φ 中比例为 v 的约束，则容易看到由该赋值得到的这
473 些相应的 r 元组可以满足 φ'' 中比例至少为 v' 的约束。对多数的研究者而言，似乎都可以"显而易见"的看到：任何赋值都不可能使得 φ'' 中被满足的约束的比例真正高于 v'。后来，人们找到了一个简单的反例使得 φ'' 中被满足的约束的比例确实比 v' 高（参见习题 22.6）。但是，莱斯证明了：任何赋值都不可能使得 φ'' 中被满足的约束的比例高于 v'，其中 c 依赖于字母表的大小 W。这个证明非常难，尽管人们已经对它做了一些简化（参见章节注记和本书的网站）。

22.4　哈斯塔德 3 位 PCP 定理和 MAX-3SAT 的难度

在第 11 章中，我们证明了 **NP**=**PCP**($\log n$, 1)。换句话说，**NP** 语言的成员资格的证明只需查验其中 $O(1)$ 个位就可以验证证明的真伪。现在，我们感兴趣的是在保证可靠性约为 1/2 的前提下使得需要查验的二进制位的个数尽可能小。下面的定理表明，需要查验的二进制位的个数可以被削减到 3，而且验证者的判定过程也仅需简单地查验这 3 个位的奇偶性。

定理 22.16（哈斯塔德 3-位 **PCP** 定理[Hås97]）　对任意 $\delta>0$ 和任意语言 $L\in$ **NP**，存在 L 的 **PCP** 验证者 V 仅查验证明中的 3 个二进制位并且具有完备性参数 $1-\delta$ 和至多为 $1/2+\delta$ 的可靠性参数。

而且，V 使用的测试是线性的。也就是说，给定一个证明 $\pi\in\{0,1\}^n$，V 根据某种分布选择元组 $(i_1, i_2, i_3)\in[m]^3$ 和 $b\in\{0,1\}$，V 接受证明 π 当且仅当 $\pi_{i_1}+\pi_{i_2}+\pi_{i_3}=b \pmod 2$。

22.4.1　MAX-3SAT 的近似难度

首先，我们注意到定义 22.16 与所谓的 MAX-E3LIN 问题的近似难度密切相关。MAX-E3LIN 是 $3CSP_2$ 的子类，实例中的每个约束要求相应的 3 个变量满足给定的奇偶性。MAX-E3LIN 的每个实例也可以看作，给定一个模 2 的线性方程组，其中每个方程至多含有 3 个变量，要求计算能够被同时满足的方程的最大个数。我们断言，定理 22.16 意味着在任意 $v>0$ 上计算 MAX-E3LIN 的 $1/2+v$-近似解是 **NP** 难的。这个结论是一个阈值结果，因为 MAX-E3LIN 有一个简单的 1/2-近似算法。（该近似算法使用的观察结果类似于我们在第 11 章中得到 MAX-3SAT 的近似算法时所用的观察结果。也就是说，从数学期望的角度看，变量的随机赋值满足一半的方程。根据这一观察结果，可以设计出一个确定型算法使得它输出的变量赋值能够至少满足一半的方程。）

为了证明上面关于 MAX-E3LIN 的近似难度的论断，我们可以利用 11.3 节的方法将定理 22.16 的验证者变换为一个等价的 CSP 实例。由于验证者在证明的 3 个二进制位上查验奇偶性，因此所得的等价 CSP 实例是 MAX-E3LIN 实例，而且要么它的 $(1-\delta)$ 比例的约束能够被同时满足要么至多只有 $1/2+\delta$ 比例的约束能够被同时满足。由于判定这两种情况中究竟哪种情况成立是 **NP**-难的，由此可知计算 MAX-E3LIN 的 ρ-近似解也是 **NP** 难
474 的，其中 $\rho=\dfrac{1/2+\delta}{1-\delta}$。由于 δ 可以任意小，因此 ρ 可以任意地接近于 1/2。由此可知，对于任意 $v>0$，计算 MAX-E3LIN 的 $(1/2+v)$-近似解是 **NP** 难的。

此外，我们还注意到，如果 **P**≠**NP**，则定理 22.16 的完备性参数必然小于 1。这是由于，用高斯消去法可以在多项式时间内判定是否存在一个解满足所有的方程。也就是说，MAX-E3LIN 的满足性问题可以在多项式时间内求解。

现在，我们证明 MAX-3SAT 的近似难度。正如前面已经提到的那样，这个结论也是一个阈值结果。

推论 22.17　对任意 $\varepsilon>0$，计算 MAX-3SAT 问题的 $(7/8+\varepsilon)$-近似解是 **NP** 难的。

证明　我们将 MAX-E3LIN 归约到 MAX-3SAT。给定上述归约产生的 MAX-E3LIN 实例，我们希望判定实例中 $1-v$ 比例的方程能被同时满足还是该实例中至多只有 $1/2+\delta$ 比例的方程能被同时满足。实例中的每个线性约束都可以轻而易举地表示成 4 个 3CNF 公式。例如，线性约束 $a+b+c=0 \pmod 2$ 等价于子句 $(\bar{a}\vee b\vee c)$，$(a\vee\bar{b}\vee c)$，$(a\vee b\vee\bar{c})$，$(\bar{a}\vee\bar{b}\vee\bar{c})$。如果 a，b，c 满足线性约束，则它们必然也满足上面的四个子句，否则它们将至多只能满足其中 3 个子句。由此得出结论，如果 MAX-E3LIN 实例中 $1-v$ 比例的方程能被同时满足，则 MAX-3SAT 实例中 $1-\varepsilon$ 比例的子句可以同时被满足，其中 $\varepsilon=\frac{1}{4}v$。如果 MAX-E3LIN 实例中至多只有 $1/2+\delta$ 比例的方程能被同时满足，则 MAX-3SAT 实例中可以同时被满足的子句的比例至多为 $1-(\frac{1}{2}-v)\times\frac{1}{4}=\frac{8}{7}+\frac{v}{4}$。当 v 减小时，$\frac{8}{7}+\frac{v}{4}$ 与 $1-\varepsilon$ 之间的比值趋向于 7/8。由于定理 22.16 表明，在 MAX-E3LIN 实例上对任意 v 区分上述两种情况都是 **NP** 难的，故推论的结论成立。■

22.5　工具：傅里叶变换

定理 22.16 的证明用到了傅里叶变换。连续傅里叶变换在数学上和工程中都极其有用。类似地，离散傅里叶变换也被广泛地应用于算法分析和复杂性理论中，特别是在各种概率可验证明(**PCP**)的构造和分析中。在概率可验证明(**PCP**)中，傅里叶变换技术主要是将傅里叶变换作用到输入的“证明位串”上来计算验证者的最大接受概率。(注记 22.21 以更广阔的视角讨论了傅里叶变换在组合论证方法和概率论证方法中的用途。)傅里叶变换技术十分精美，用它可以给出 MAX-INDSET，MAX-3SAT 以及许多其他问题的“紧”近似难度。

我们从简单的例子开始介绍傅里叶变换。首先，本节分析 GF(2) 上的线性测试，由此完成定理 11.21 的证明。然后，22.6 节介绍长编码并说明如何测试其成员资格。这些思想将在 22.7 节用于证明哈斯塔德 3-位 **PCP** 定理。

22.5.1　$GF(2)^n$ 上的傅里叶变换

$GF(2)^n$ 上的傅里叶变换是研究布尔超方体上的函数的有力工具。在本章，用集合 $\{+1,-1\}=\{\pm1\}$ 代替集合 $\{0,1\}$ 将带来极大的方便。要将 $\{0,1\}$ 变换为 $\{\pm1\}$，只需使用映射 $b\mapsto(-1)^b$(亦即，$0\mapsto+1$，$1\mapsto-1$)。这样，超方体就变成了 $\{\pm1\}^n$，而不再是以前的 $\{0,1\}^n$。注意，这个映射将 XOR(亦即，GF(2) 上的加法)操作变成了 **R** 上的乘法操作。 475

从 $\{\pm1\}^n$ 到 **R** 的所有函数定义了一个 2^n 维的希尔伯特空间(亦即，带有内积的向量空间)。在这个空间中，加法和数乘以自然的方式进行定义。也就是说，$(f+g)(x)=$

$f(x)+g(x)$，$(af)(x)=af(x)$对任意 f，$g:\{\pm1\}^n\to\mathbf{R}$ 和 $a\in\mathbf{R}$ 都成立。函数 f，g 的内积$\langle f,g\rangle$定义为 $\mathrm{E}_{x\in\{\pm1\}^n}[f(x)g(x)]$。这种内积称为期望内积。

该空间的标准基是函数集合$\{\mathbf{e}_x\}_{x\in\{\pm1\}^n}$，其中 $\mathbf{e}_x(y)=1$，如果 $x=y$；否则，$\mathbf{e}_x(y)=0$。标准基是正交基，并且任意函数 $f:\{\pm1\}^n\to\mathbf{R}$ 都可以用标准基表示为 $f=\sum_x a_x\mathbf{e}_x$，其中系数 $a_x=f(x)$ 对于任意 $x\in\{\pm1\}^n$ 成立。

该空间的傅里叶基是另一组标准正交基。任意子集 $\alpha\subseteq[n]$在傅里叶基中对应函数 $\chi_\alpha(x)=\prod_{i\in\alpha}x_i$。我们将 χ 定义为处处为 1 的函数。傅里叶基实际上就是沃尔什-哈达玛编码(参见 11.5.1 节)的另一种形式：基中的向量对应于 GF(2)上的线性函数。为了看清这一点，形如 $\mathbf{b}\mapsto\mathbf{a}\odot\mathbf{b}$(其中 $\mathbf{a}$，$\mathbf{b}\in\{0,1\}^n$)的任意线性函数在我们的变换之下被映射为函数 $x\in\{\pm1\}^n\to\prod_{i:a_i=1}x_i$。为了验证傅里叶基确实是 $\mathbf{R}^{2^n}$的标准正交基，只需注意到，随机子和原理意味着在任意 α，$\beta\subseteq[n]$上均有$\langle\chi_\alpha,\chi_\beta\rangle=\delta_{\alpha,\beta}$，其中 $\delta_{\alpha,\beta}$等于 1 当且仅当 $\alpha=\beta$，否则 $\delta_{\alpha,\beta}$等于 0。

评注 22.18　注意，用$\{-1,1\}$的观点看，基函数可以视为多变量线性多项式(亦即，每个变量的次数都等于 1 的多变量多项式)。于是，形如 $f:\{-1,1\}^n\to\mathbf{R}$ 的任意函数都存在傅里叶展开这一事实可以重述为"每个这样的函数都可以用多变量线性多项式来表示"。这种表示本质上等同于第 8 章和第 11 章的多项式表示。

由于傅里叶基是标准正交基，故任意函数 $f:\{\pm1\}^n\to\mathbf{R}$ 都可以表示为 $f=\sum_{\alpha\subseteq[n]}\hat{f}_\alpha\chi_\alpha$。我们称$\hat{f}_\alpha$是 f 的傅里叶系数。我们会经常使用如下简单的引理。

引理 22.19　任意函数 f，$g:\{\pm1\}^n\to\mathbf{R}$ 均满足

1. $\langle f,g\rangle=\sum_\alpha\hat{f}_\alpha\hat{g}_\alpha$。

2.(**巴塞弗恒等式**(Parseval's Identity))$\langle f,f\rangle=\sum_\alpha\hat{f}_\alpha^2$。

证明　第 2 个性质由第 1 个性质可得。为证明第 1 个性质，只需将内积展开得到

$$\langle f,g\rangle=\Big\langle\sum_\alpha\hat{f}_\alpha\chi_\alpha,\sum_\beta\hat{g}_\beta\chi_\beta\Big\rangle=\sum_{\alpha,\beta}\hat{f}_\alpha\hat{g}_\beta\langle\chi_\alpha,\chi_\beta\rangle$$

$$=\sum_{\alpha,\beta}\hat{f}_\alpha\hat{g}_\alpha\delta_{\alpha,\beta}=\sum_\alpha\hat{f}_\alpha\hat{g}_\alpha$$

■

476

例 22.20　本例给出几个特殊函数的傅里叶变换。

1. 3 个变量上的多数函数 M 定义为：如果 u_1，u_2，u_3 中至少有两个变量取 $+1$，则 $M(u_1,u_2,u_3)$等于 $+1$；否则，$M(u_1,u_2,u_3)$等于 -1。它可以表示为$\frac{1}{2}u_1+\frac{1}{2}u_2+\frac{1}{2}u_3-\frac{1}{2}u_1u_2u_3$。于是，它有 3 个傅里叶系数等于$\frac{1}{2}$，有 1 个傅里叶系数等于$-\frac{1}{2}$，其余傅里叶系数等于 0。

2. 如果 $f(u_1,u_2,\cdots,u_n)=u_i$(亦即，$f$ 是一个坐标函数。坐标函数的概念我们在讨论长编码时还会再遇到)，则 $f=\chi_{\{i\}}$。因此，$\hat{f}_{\{i\}}=1$ 且$\hat{f}_\alpha=0$ 对任意 $\alpha\neq\{i\}$成立。

3. 如果 f 是 n 个位上的随机布尔函数，则$\hat{f}_\alpha$是随机变量，它是 2^n个二项变量之和(其中每个二项变量等概率地取 1 和 -1)。于是，$\hat{f}_\alpha$看上去有点像均值为 0 且标准差为 $2^{n/2}$的

正态分布变量。因此，随机布尔函数的 2^n 个傅里叶系数都以较高概率取值于区间 $\left[-\frac{\mathrm{poly}(n)}{2^{n/2}}, \frac{\mathrm{poly}(n)}{2^{n/2}}\right]$。◀

22.5.2　从较高层面看傅里叶变换和 PCP 之间的联系

在概率可验证明(**PCP**)中，我们感兴趣的是布尔值函数(亦即，从 GF(2)^n 到 GF(2)的函数)。在我们的变换下，这种函数变为了从$\{\pm 1\}^n$到$\{\pm 1\}$的函数。因此，如果函数 $f:\{\pm 1\}^n \to \mathbf{R}$ 使得 $f(x)\in\{\pm 1\}$对任意 $x\in\{\pm 1\}^n$成立，则我们称 f 是布尔函数。注意，如果 f 是布尔函数，则$\langle f, f\rangle = E_x[f(x)^2]=1$。

从较高的层面上看，在概率可验证明(**PCP**)的可靠性证明中，我们可以用傅里叶变换来证明：如果验证者高概率地接受证明 π，则 π 将"接近于"某种"良好形式"(这里，"接近于"和"良好形式"都要依赖于具体的概率可验证明系统)。通常，我们将验证者的接受概率关联到某种数学期望的形式$\langle f, g\rangle = E_x[f(x)g(x)]$，其中 f 和 g 都是出现在概率可验证明中的布尔函数。然后，我们再用类似于"引理 22.19 中结论"的技术将验证者的接受概率与 f，g 的傅里叶系数关联起来。这使得我们可以这样来完成论证：如果验证者能以较高的概率接受所做的测试，则 f 和 g 几乎没有相对较大的傅里叶系数。这样，我们就得到了 f，g 的一些非平凡的有用信息。这是由于，在一般的函数或随机函数中，所有傅里叶系数都比较小而且大致相等。

22.5.3　GF(2)上线性测试的分析

现在，我们证明定理 11.21，继而完成 **PCP** 定理的证明。回顾一下，线性测试要求在给定的函数 f：GF(2)^n→GF(2)上测试 f 是否在大多数输入上与某个线性函数取相同的值。为完成线性测试，可以先随机选取 x，$y\in$GF(2)^n，然后接受当且仅当 $f(x+y)=f(x)+f(y)$。

现在，我们用$\{\pm 1\}$代替 GF(2)并重新表述线性测试。此时，线性函数变成了傅里叶基函数。对于任意 x，$y\in\{\pm 1\}^n$，将它们的对应分量相乘得到的向量表示为 xy。也就是说，$xy=(x_1y_1, \cdots, x_ny_n)$。注意，在任意基函数上有 $\chi_S(xy)=\chi_S(x)\chi_S(y)$。

对于两个布尔函数 f，g，它们的内积$\langle f, g\rangle$等于"f，g 取相同函数值的输入所占比例"与"f，g 取值不同的输入所占比例"之差。由此可知，对任意 $\epsilon\in[0, 1]$和任意函数 f，$g:\{\pm 1\}^n\to\{\pm 1\}$而言，$f$ 和 g 在 $\frac{1}{2}+\frac{1}{\epsilon}$比例的输入上取相同的值当且仅当$\langle f, g\rangle=\epsilon$。因此，如果 f 有一个很大的傅里叶系数，则 f 在多数输入上的取值与某个基函数的取值相同。从 GF(2)的观点看，f 接近于一个线性函数。这意味着，定理 11.21 所表述的线性测试的正确性可以重述为下面的定理 22.22。 477

注记 22.21　(自相关函数，等周问题，相变)

虽然用傅里叶变换来证明 **PCP** 定理让人颇感意外，但追溯历史你将看到这其实是很自然的。本注记阐述这种观点。要进一步了解这个话题的相关背景，请读者参阅卡莱(Kalai)和萨弗拉(Safra)的综述[KS06]，奥唐奈(O'Donnell)和莫塞尔(Mossell)放在网上的讲义，以及其他一些资料。

傅里叶变换作为一种经典的有用工具，它通常用于证明如下形式的结论："如果一个

函数以某种结构化的方式与它自身相关，则它必然属于某个很小的函数族。"在概率可验证明(**PCP**)中，函数 $f:\{0,1\}^n\to\{0,1\}$ 是"自相关的"的含义是：如果我们在 f 上执行指定的验证者算法时只需查验其中的少数几个二进制位，则验证者将以合理的概率接受。例如，在 GF(2)上进行线性测试时，验证者接受测试的概率是 $E_{x,y}[I_{x,y}]$，其中 $I_{x,y}$ 是随机事件 $f(x+y)=f(x)+f(y)$ 的示性随机变量。

傅里叶变换的另一个经典的应用是研究等周问题(Isoperimetry)。等周问题是"最小表面积问题"的子问题。一个简单的例子是，在 $\mathbf{R}^2$ 上具有指定面积的所有连通区域中，圆具有最小的周长。同样，等周问题也可以看成是对"自相关性"的研究。事实上，考虑特定空间中点集的特征函数，只要将空间中形成"给定表面积"的点集的边界在某个方向上微微小修改，则特征函数的取值将由 1 变成 0。

22.7 节将要出现的哈斯塔德"噪音"操作符出现在数学家尼尔森(Nelson)，伯纳密(Bonamie)，贝克纳(Beckner)等人的工作中。他们用它来研究超收缩估计(Hypercontractive Estimate)。这个一般性的研究主题也是用函数的"自相关"行为来刻画函数的性质。考察函数 f 与函数 $T_\rho(f)$ 的相关性，其中函数 $T_\rho(f)$(大致上讲)是在给定点的球形小邻域内计算 f 函数值的平均值。可以证明，f 的范数和 $T_\rho(f)$ 的范数是相关的。虽然这一事实在哈斯塔德的证明中并未用到，但在 22.9 节概述其他 **PCP** 定理时却非常有用。习题 22.10 粗浅地讨论了这种相关性。

傅里叶变换和超收缩估计在研究随机图的相变(Phase Transition)时也非常有用(例如，参见弗里德古特(Friedgut)[Fri99])。在最简单的随机图模型 $G(n,p)$ 中，每条可能的边独立地以概率 p 出现在图中。相变指的是 p 的一个取值，它使得当概率值从小于 p 变化到 p 时随机图将从几乎都不具有某种特定的性质变化为几乎都具有这种性质。例如，人们已经证明了，存在一个常数 c 使得 $p=c\log n/n$ 前后，"随机图是连通的"的概率将从 0 跃变为 1。傅里叶变换之所以可以用来研究随机图的相变，是因为相变问题是特殊"图"上的等周问题，该"图"中的每个"顶点"是一个 n-顶点图，"顶点"之间存在边意味着相应的图可以通过增、删少数几条边互相转换。注意，增、删少数几条边对应于 p 的微小变动。

478 最后，我们提一下傅里叶变换在 22.9.4 节和 22.9.5 节中的几个有趣的应用。这些应用涉及超方体$\{0,1\}^n$上的等周问题。我们也可以在图上研究等周问题。我们可以将顶点子集的"表面积"定义为"离开该顶点子集的边的条数"，也可以按其他方式定义"表面积"。在超方体图和类超方体图中，傅里叶变换可以用来证明等周定理。这是因为，此时子集 $S\subseteq\{0,1\}^n$ 不是别的，而是一个布尔函数，该函数在 S 上取值为 1 而在其他输入上取值为 -1。如果图是 D-正则的且 $|S|=2^{n-1}$，则

$$E_{(x,y)\in E}[f(x)f(y)]=\frac{1}{2^nD}(|E(S,S)|+|E(\overline{S},\overline{S})|-2|E(S,\overline{S})|)$$

上式表明，离开顶点子集 S 的边在所有边中所占的比例是 $1/2-E[f(x)f(y)]/2$。这种表达式可以用傅里叶变换来分析，参见习题 22.11(b)。

定理 22.22　如果函数 $f:\{\pm1\}^n\to\{\pm1\}$ 满足 $\Pr_{x,y}[f(xy)=f(x)f(y)]\geqslant\frac{1}{2}+\varepsilon$，则存在 $\alpha\subseteq[n]$ 使得 $\hat{f}_\alpha\geqslant 2\varepsilon$。

证明　定理的假设条件可以重述为 $E_{x,y}[f(xy)f(x)f(y)]\geqslant\left(\frac{1}{2}+\varepsilon\right)-\left(\frac{1}{2}-\varepsilon\right)=2\varepsilon$。

注意，从现在开始，我们不再需要 f 是布尔函数的假设了，而只需要 f 满足 $\langle f, f\rangle = 1$ 即可。

将 f 进行傅里叶展开，得到

$$2\varepsilon \leqslant \underset{x,y}{E}[f(xy)f(x)f(y)] = \underset{x,y}{E}[(\sum_{\alpha}\hat{f}_{\alpha}\chi_{\alpha}(xy))(\sum_{\beta}\hat{f}_{\beta}\chi_{\beta}(x))(\sum_{\gamma}\hat{f}_{\gamma}\chi_{\gamma}(y))]$$

由于 $\chi_{\alpha}(xy)=\chi_{\alpha}(x)\chi_{\alpha}(y)$，故上式最后一项等于

$$= \underset{x,y}{E}[\sum_{\alpha,\beta,\gamma}\hat{f}_{\alpha}\hat{f}_{\beta}\hat{f}_{\gamma}\chi_{\alpha}(x)\chi_{\alpha}(y)\chi_{\beta}(x)\chi_{\gamma}(y)]$$

利用数学期望的线性性质，可得

$$= \sum_{\alpha,\beta,\gamma}\hat{f}_{\alpha}\hat{f}_{\beta}\hat{f}_{\gamma}\underset{x,y}{E}[\chi_{\alpha}(x)\chi_{\alpha}(y)\chi_{\beta}(x)\chi_{\gamma}(y)]$$

$$= \sum_{\alpha,\beta,\gamma}\hat{f}_{\alpha}\hat{f}_{\beta}\hat{f}_{\gamma}\underset{x}{E}[\chi_{\alpha}(x)\chi_{\beta}(x)]\underset{y}{E}[\chi_{\alpha}(y)\chi_{\gamma}(y)] \quad (\text{因为 } x,y \text{ 是独立的})$$

由正交性可知 $E_x[\chi_{\alpha}(x)\chi_{\beta}(x)]=\delta_{\alpha,\beta}$，因此上式可以化简为

$$= \sum_{\alpha}\hat{f}_{\alpha}^{3}$$

$$\leqslant (\max_{\alpha}\hat{f}_{\alpha})\times(\sum_{\alpha}\hat{f}_{\alpha}^{2})$$

由于 $\sum_{\alpha}\hat{f}_{\alpha}^{2} = \langle f, f\rangle = 1$，由此可得 $\max_{\alpha}\hat{f}_{\alpha} \geqslant 2\varepsilon$，这就证明了定理。 ■ 479

22.6　坐标函数、长编码及其测试

哈斯塔德 3-位 **PCP** 定理用到了一种编码方法——长编码。设 $W \in \mathbf{N}$，对于函数 $f: \{\pm 1\}^{W} \to \{\pm 1\}$，如果存在 $w \in [W]$ 使得 $f(x_1, x_2, \cdots, x_W) = x_w$，则称 f 是坐标函数。换句话说，$f = \chi_w$。[⊖]（注意，不同于前几节，本节用 W 代替 n 来表示变量的个数，这样做是为了与 22.7 节用 W 表示 $2\mathrm{CSP}_W$ 中的字母表大小保持一致。）

定义 22.23（长编码）　$[W]$ 的长编码将任意 $w \in [W]$ 编码为函数 $\chi_w: \{\pm 1\}^{W} \to \{\pm 1\}$ 的所有函数值表。

注意，正常情况下 w 只需用 $\log W$ 个位就可以出来，但我们的编码却用长度为 2^W 个位的表来表示它，它是 $\log W$ 的双重指数！编码的这种低效性正是我们将它称为“长编码”的原因。

长编码的成员资格测试问题类似于之前定义的沃尔什-哈达玛编码的测试问题。也就是说，给定一个函数 $f: \{\pm 1\}^{W} \to \{\pm 1\}$，我们需要判定是否存在一个 $w \in [W]$ 使得 f 和 χ_w 具有高度的一致性，亦即，$\hat{f}_w$ 是否显著地大于 0。习题 22.5 讨论这种测试，但它仍不足以证明哈斯塔德定理，因为哈斯塔德定理要求只查验 3 个位。下面给出一个只查验 3 个位的测试方法，但必须牺牲一定的性能，故我们只能得到较弱的性能保障：如果测试能以较高概率通过，则在 $|\alpha|$ 较小的某个子集上 f 与 χ_α 具有高度的一致性（虽然 $|\alpha|$ 比较小，但它不必等于 1）。这种较弱的结论就足以用来证明定理 22.16。

⊖ 有些教科书用独裁函数（Dictatorship Function）来称呼这种函数，因为某个变量（独裁变量）完全确定了函数的输出。这种称呼源自社会选择理论（Social Choice Theory），它研究各种选举过程。该论域的研究也由于采用了注记 22.21 中的傅里叶分析思想而取得了长足的进步。

令 $\rho>0$ 是某个任意小的常数。我们的测试先均匀随机地选取 $x,y\in_R(\pm1)^W$，然后再根据如下的分布随机地选取一个向量 $z\in(\pm1)^W$：对于每个坐标位置 $i\in[W]$，以概率 $1-\rho$ 取 $z_i=1$，以概率 ρ 取 $z_i=-1$。因此，“z 中 ρ 比例的坐标位等于 -1 且 $1-\rho$ 比例的坐标位等于 $+1$”的概率比较高。我们将 z 视为“噪音”向量。测试最终接受当且仅当 $f(x)f(y)=f(xyz)$。注意，除了用到了噪音向量 z 之外，上述测试与线性测试非常相似。

假设 $f=\chi_{\{w\}}$。则，由于 $b\cdot b=1$ 对 $b\in(\pm1)$ 恒成立，故

$$f(x)f(y)f(xyz)=x_w y_w(x_w y_w z_w)=1\cdot z_w$$

因此，测试接受当且仅当 $z_w=1$，而这发生的概率等于 $1-\rho$。现在，我们证明该结论的某种逆命题。

480

引理 22.24　如果测试以 $1/2+\delta$ 的概率接受，则 $\sum_{\alpha}\hat{f}_\alpha^2(1-2\rho)^{|\alpha|}\geqslant 2\delta$。

证明　如果测试以 $1/2+\delta$ 的概率接受，则 $\mathrm{E}[f(x)f(y)f(xyz)]=2\delta$。将 f 替换为它的傅里叶展开，我们有

$$\begin{aligned}2\delta&\leqslant \mathop{E}_{x,y,z}\Big[\Big(\sum_\alpha \hat{f}_\alpha\chi_\alpha(x)\Big)\Big(\sum_\beta \hat{f}_\beta\chi_\beta(y)\Big)\Big(\sum_\gamma \hat{f}_\gamma\chi_\gamma(xyz)\Big)\Big]\\&=\mathop{E}_{x,y,z}\Big[\sum_{\alpha,\beta,\gamma}\hat{f}_\alpha\hat{f}_\beta\hat{f}_\gamma\chi_\alpha(x)\chi_\beta(y)\chi_\gamma(x)\chi_\gamma(y)\chi_\gamma(z)\Big]\\&=\sum_{\alpha,\beta,\gamma}\hat{f}_\alpha\hat{f}_\beta\hat{f}_\gamma\mathop{E}_{x,y,z}\left[\chi_\alpha(x)\chi_\beta(y)\chi_\gamma(x)\chi_\gamma(y)\chi_\gamma(z)\right]\end{aligned}$$

标准正交性意味着 $\chi_\alpha(x)\chi_\beta(y)\chi_\gamma(x)\chi_\gamma(y)\chi_\gamma(z)=0$，除非 $\alpha=\beta=\gamma$。因此，上式等于

$$=\sum_\alpha \hat{f}_\alpha^2\mathop{E}_z[\chi_\alpha(z)]$$

再由 $E_z[\chi_\alpha(z)]=E_z[\prod_{w\in\alpha}(z_w)]$，它等于 $\prod_{w\in\alpha}E[z_w]=(1-2\rho)^{|\alpha|}$，这是由于 z 的各个坐标位是随机独立地选取的。由此可得

$$2\delta\leqslant\sum_\alpha\hat{f}_\alpha^2(1-2\rho)^{|\alpha|}$$

■

引理 22.24 的结论让我们回想起定理 22.22 证明过程中的计算结果，只是多出了一个因子 $(1-2\rho)^{|\alpha|}$。该因子抵消了大 α 对应的 $\hat{f}_\alpha$ 对不等式左端的贡献，这使得我们得知小 α 对应的 $\hat{f}_\alpha$ 对不等式左端的贡献比较大。这一观察可以形式化为下面的推论（其证明过程只涉及简单的计算，我们将它留作习题 22.8）。

推论 22.25　如果 f 以 $1/2+\delta$ 的概率通过长编码测试，则存在满足 $|\alpha|\leqslant k$ 的 $\alpha\subseteq[W]$ 使得 $\hat{f}_\alpha\geqslant 2\delta-\varepsilon$，其中 $k=\frac{1}{2\rho}\log\frac{1}{\varepsilon}$。

22.7　定理 22.16 的证明

现在，我们证明哈斯塔德定理。证明的出发点是定理 22.15 给出的 $2\mathrm{CSP}_w$ 实例 φ。因此我们知道，φ 要么是可满足的，要么 φ 中比例至多为 ε 的约束是可满足的，其中 ε 是任意小的数。设 W 是字母表的大小，n 是变量的个数，m 是约束的个数。我们将 φ 的每个赋值视为从 $[n]$ 到 $[W]$ 的函数 π。由于给定的 2CSP 实例 φ 满足投影性质，故其中的每个约束 φ_i 都可以等价地视为一个函数 $h_i:[W]\to[W]$，该约束被赋值 π 满足当且仅当 $\pi(j)=h_i(\pi(i))$。

哈斯塔德验证者将使用长编码，而且这些编码还是双折叠的（Bifolded）。下面先定义双折叠性，它是一种技巧性的性质。之所以定义这种性质，是因为我们观察到坐标函数使

得 $\chi_{\{w\}}(-v)=-\chi_{\{w\}}(v)$对任意向量 v 成立。

定义 22.26　如果函数 $f:\{\pm1\}^W\to\{\pm1\}$使得 $f(-v)=-f(v)$对任意 $v\in\{\pm1\}^W$成立，则称 f 是双折叠的。

481

（注意，数学上通常将这种函数称为奇函数，但是在概率可验证明的相关文献中却使用"折叠(folding)"作为这种函数的标准术语，它表达了更宽泛的含义。）

只要概率可验证明(**PCP**)中的证明 π 是长编码的一个码字，则我们可以不失一般性地假设该证明对应的函数是一个双折叠函数。因为，验证者可以在任意一对输入 v，$-v$ 上识别一个代表(比如，首个符号是+1 的符号串)作为 v 并定义 $f(-v)$为 $-f(v)$。这种约定的好处(虽然证明过程并未用到这一点)是，将双折叠函数表示为位串时可以节省一半的长度。我们将使用下面的引理。

引理 22.27　如果函数 $f:\{\pm1\}^W\to\{\pm1\}$是双折叠的并且 $\hat{f}_a\neq0$，则 $|a|$必然是一个奇数(特别地，$|a|\neq0$)。

证明　由傅里叶系数的定义可知，

$$\hat{f}_a=\langle f,\chi_a\rangle=\underset{v}{E}\left[f(v)\prod_{i\in a}v_i\right]$$

如果 $|a|$是偶数，则 $\prod_{i\in a}v_i=\prod_{i\in a}(-v_i)$。因此，如果 f 是双折叠的，则 v 和 $-v$ 对上式的最末一项的贡献将互相抵消，因此整个数学期望将等于 0。 ■

哈斯塔德验证者

现在，我们给出哈斯塔德验证者 V_H。验证者希望给定的证明 π是 φ 的满足性赋值，其中 n 个变量中的每个变量的取值都被编码为(双折叠的)长编码。因此，该证明由 $n2^W$个二进制位构成(如果利用双折叠性，则该证明也可以表示为 $n2^{W-1}$个二进制位)。V_H将给定的证明视为 n 个函数 f_1，…，f_n，其中每个函数都将 $\{\pm1\}^W$映射到 $\{\pm1\}$。验证者 V_H从 $2\mathrm{CSP}_W$实例 φ 中随机选择一个约束，不妨记为 $\varphi_c(i,j)$。然后，验证者 V_H(只使用 3 个位!)验证[W]中由 f_i，f_j所编码的两个值满足 φ_c。换句话说，V_H验证 f_i，f_j所编码的值 w，u 满足 $h(w)=u$，其中 $h:[W]\to[W]$是描述 φ_c的函数。现在，我们给出哈斯塔德测试，它就是我们前面见过的长编码测试。

基本的哈斯塔德测试

输入：函数 f，$g:\{\pm1\}^W\to\{\pm1\}$和函数 $h:[W]\to[W]$。

目标：验证 f，g 是否是两个满足 $h(w)=u$ 的值 w，u 的长编码。

测试：对于 $u\in[W]$，令 $h^{-1}(u)$表示集合 $\{w:h(w)=u\}$。注意，$\{h^{-1}(u):u\in[W]\}$构成了[W]的一个划分。对于位串 $y\in\{\pm1\}^W$，定义 $\mathcal{H}^{-1}(y)$是 $\{\pm1\}^W$中的一个位串使得：对于任意 $w\in[W]$而言，$\mathcal{H}^{-1}(y)$的第 w 个符号等于 $y_{h(w)}$。换句话说，对于任意 $u\in[W]$，符号 y_u出现在 $\mathcal{H}^{-1}(y)$中由 $h^{-1}(u)$给出的所有坐标位置上。V_H均匀随机地选取 v，$y\in\{\pm1\}^W$，再如下地随机选取 $z\in\{\pm1\}^W$，其中 $z_i=+1$的概率为 $1-\rho$而 $z_i=-1$的概率为 ρ。最后，如果

$$f(v)g(y)=f(\mathcal{H}^{-1}(y)vz) \tag{22.4}$$

则 V_H接受，否则 V_H拒绝。

482

将 $\{\pm1\}$翻译回 $\{0,1\}$。注意，V_H测试实际上就是线性测试。由于 V_H接受当且仅当 $\pi[i_1]+\pi[i_2]+\pi[i_3]=b$对某个 i_1，i_2，$i_3\in[n2^W]$和 $b\in\{0,1\}$成立。(b 实际上可以取定为 1，因为我们确保了 V_H的双折叠性。)

现在，由于 ρ，ϵ 是任意小的数，故由下面的论断可以立刻证得定理。（具体地讲，令 $\rho=\epsilon^{1/3}$ 就可以使得完备性参数至少为 $1-\epsilon^{1/3}$ 且可靠性参数至多为 $1/2+\epsilon^{1/3}$。）

论断 22.28（主）　如果 φ 是可满足的，则存在一个证明使得 V_H 以概率 $1-\rho$ 接受。如果 $\mathrm{val}(\varphi)\leqslant\epsilon$，则 V_H 不会以大于 $1/2+\delta$ 的概率接受任何证明，其中 $\delta=\sqrt{\epsilon/\rho}$。

本节后面的部分全部是对论断 22.28 的证明。

V_H 的完备性：易证

如果 φ 是可满足的，则任意给定一个满足性赋值 $\pi:[n]\to[W]$，这 n 个值都可以通过用双折叠的长编码分别进行编码来得到 V_H 所需要的证明。（前面已经指出，坐标函数是双折叠函数。）欲证 V_H 以 $1-\rho$ 的概率接受该证明，只需证明基本的哈斯塔德测试以 $1-\rho$ 的概率接受每个约束。

假设 f，g 是满足 $h(w)=u$ 的整数 w，u 的长编码，则根据 $x^2=1$ 对任意 $x\in\langle\pm1\rangle$ 成立，有

$$\begin{aligned}f(v)g(y)f(\mathcal{H}^{-1}(y)vz)&=v_uy_u(\mathcal{H}^{-1}(y)_vv_uz_u)\\&=v_uy_u(y_{h(u)}v_uz_u)=z_u\end{aligned}$$

因此，V_H 接受当且仅当 $z_u=1$，而 $z_u=-1$ 成立的概率恰好是 $1-\rho$。

V_H 的可靠性：难证

我们先证明，如果基本的哈斯塔德测试以显著大于 $1/2$ 的概率接受两个函数 f，g，则 f，g 的傅里叶变换必然是相关的。为了形式化地进行论证，我们在任意 $a\subseteq[W]$ 上定义

$$h_\Sigma(a)=\{u\in[W]_\Sigma\mid h^{-1}(u)\cap a\text{ 是奇数}\}\tag{22.5}$$

注意，特别地，对任意 $t\in h_\Sigma(a)$，至少存在一个 $w\in a$ 使得 $h(w)=t$。

在下面的引理中，δ 可以取负数。这是我们唯一用到双折叠性的地方。

引理 22.29　如果 f，$g:\langle\pm1\rangle^W\to\langle\pm1\rangle$ 是双折叠函数，并且 $h:[W]\to[W]$ 使得它们通过(22.4)式中基本哈斯塔德测试的概率至少为 $1/2+\delta$，则

$$\sum_{v\subseteq[W],|v|=1}\hat f_v\hat g_{h_\Sigma(v)}(1-2\rho)^{|v|}\geqslant2\delta\tag{22.6}$$

483

证明　根据假设可知，f，g 满足 $E[f(v)f(v\,\mathcal{H}^{-1}(y)z)g(y)]\geqslant2\delta$。将 f，g 替换为它们的傅里叶展开，我们得到

$$\begin{aligned}2\delta&\leqslant\mathop{E}_{x,y,z}\Big[\Big(\sum_\alpha\hat f_\alpha\chi_\alpha(v)\Big)\Big(\sum_\beta\hat g_\beta\chi_\beta(y)\Big)\Big(\sum_\gamma\hat f_\gamma\chi_\gamma(v\,\mathcal{H}^{-1}(y)z)\Big)\Big]\\&=\sum_{\alpha,\beta,\gamma}\hat f_\alpha\hat g_\beta\hat f_\gamma\mathop{E}_{x,y,z}[\chi_\alpha(v)\chi_\beta(y)\chi_\gamma(v)\chi_\gamma(\mathcal{H}^{-1}(y))\chi_\gamma(z)]\end{aligned}$$

根据正交性，上式等于

$$\begin{aligned}&=\sum_{\alpha,\beta}\hat f_\alpha\hat g_\beta\mathop{E}_{y,z}[\chi_\beta(y)\chi_\alpha(\mathcal{H}^{-1}(y))\chi_\alpha(z)]\\&=\sum_{\alpha,\beta}\hat f_\alpha\hat g_\beta(1-2\rho)^{|\alpha|}\mathop{E}_{y}[\chi_\alpha(\mathcal{H}^{-1}(y))\chi_\beta(y)]\end{aligned}\tag{22.7}$$

这是因为长编码测试的分析中已经得到 $\chi_\alpha(z)=(1-2\rho)^{|\alpha|}$。现在，我们有

$$\begin{aligned}\mathop{E}_{y}[\chi_\alpha(\mathcal{H}^{-1}(y))\chi_\beta(y)]&=\mathop{E}_{y}\Big[\prod_{u\in\alpha}\mathcal{H}^{-1}(y)_u\prod_{u\in\beta}y_u\Big]\\&=\mathop{E}_{y}\Big[\prod_{u\in\alpha}y_{h(u)}\prod_{u\in\beta}y_u\Big]\end{aligned}$$

它等于 1，如果 $h_\Sigma(\alpha)=\beta$；否则，它等于 0。于是，(22.7)式可以化简为

$$\sum_{\alpha} \hat{f}_{\alpha}^2 \hat{g}_{h_2(\alpha)}(1-2\rho)^{|\alpha|}$$

最后，注意到，由于我们假设 f，g 都是双折叠的，故傅里叶系数 $\hat{f}_{\emptyset}$ 和 $\hat{g}_{\emptyset}$ 都等于 0。从上式删除 $\hat{f}_{\emptyset}$ 和 $\hat{g}_{\emptyset}$ 就得到引理的结论。 ■

下面的引理完成论断 22.28 的证明，由此完成对哈斯塔德 3-位 **PCP** 定理的证明。

引理 22.30　假设 φ 是一个满足 $\mathrm{val}(\varphi)\leqslant\varepsilon$ 的 $2\mathrm{CSP}_W$ 实例。如果 ρ 和 δ 满足 $\rho\delta^2>\varepsilon$，则验证者 V_H 接受任何证明的概率都不会超过 $1/2+\delta$。

证明　假设 V_H 至少以 $1/2+\delta$ 的概率接受长度为 $n2^W$ 的证明 $\tilde{\pi}$。我们构造 φ 的所有变量的概率型赋值 π 使得 φ 中被 π 满足的约束的比例的数学期望至少为 $\rho\delta^2$。进而，根据概率论证法可知，φ 的所有变量存在一个具体的赋值 π 使得 φ 中被它满足的约束的比例至少为 $\rho\delta^2$。在 $\rho\delta^2>\varepsilon$ 的情况下，这与 $\mathrm{val}(\varphi)\leqslant\varepsilon$ 这一假设条件矛盾。

构造 π 所需的概率分布

我们可以认为，$\tilde{\pi}$ 对任意 $i\in[n]$ 给出了一个函数 $f_i:\{\pm1\}^W\to\{\pm1\}$。概率型赋值 π 的构造分两个步骤完成。我们先用 f_i 如下地定义 $[W]$ 上的一个概率分布 $\mathcal{D}_i$：以概率 $\hat{f}_{\alpha}^2$ 随机选择 $\alpha\subseteq[W]$（其中 $f=f_i$），然后从 α 中随机选择 w。分布 $\mathcal{D}_i$ 是良定义的，因为 $\sum_{\alpha}\hat{f}_{\alpha}^2=1$ 并且（由双折叠性可知）空集对应的傅里叶系数 $\hat{f}_{\emptyset}$ 等于 0。然后，根据分布 $\mathcal{D}_i$ 随机选取 $\pi[i]$。于是，赋值 π 是服从乘积分布 $\prod_{i=1}^{n}\mathcal{D}_i$ 的随机变量。我们希望证明

$$\mathop{E}_{\pi}\Big[\mathop{E}_{r\in[m]}[\pi\text{ 满足第 }r\text{ 个约束}]\Big]\geqslant\rho\delta^2 \tag{22.8}$$

分析

对于任意 φ_r（其中 $r\in[m]$），用 $1/2+\delta_r$ 表示条件概率：在 V_H 已经选择了 φ_r 的前提下，哈斯塔德测试接受 $\tilde{\pi}$ 的概率。（注意，δ_r 有可能取负数。）这样，V_H 的接受概率可以表达成 $E_r[\frac{1}{2}+\delta_r]$。由此可得，$E_r[\delta_r]=\delta$。下面证明

$$\Pr_{\pi}[\pi\text{ 满足 }\varphi_r]\geqslant\rho\delta_r^2 \tag{22.9}$$

由(22.9)式和数学期望的线性性质可得，(22.8)式的左端至少是 $\rho E_{r\in[m]}[\delta_r^2]$。再由 $E[X^2]\geqslant E[X]^2$ 对任意随机变量都成立，可知 $\rho(E_r[\delta_r])^2\geqslant\rho\delta^2$。综上所述，要证明(22.8)式，只需证明(22.9)式。

令 $\varphi_r(i,j)$ 是第 r 个约束，并且 h 是描述该约束的函数。于是，

$$\pi\text{ 满足 }\varphi_r\qquad\text{当且仅当}\qquad h[\pi[i]]=\pi[j]$$

令 I_r 是随机事件 $h[\pi[i]]=\pi[j]$ 的示性随机变量。从现在起，我们使用如下的简记法：$f=f_i$，$g=f_j$。两个变量的随机赋值 $\pi[i]\in_R\mathcal{D}_i$，$\pi[j]\in_R\mathcal{D}_j$ 满足 $\pi[j]=h[\pi[i]]$ 的概率是多少呢？注意，这两个变量的随机赋值是这样获得的：以概率 $\hat{f}_{\alpha}^2$ 随机选择 α，以概率 $\hat{g}_{\beta}^2$ 随机选择 β，然后均匀随机地选择 $\pi[i]\in_R\alpha$，$\pi[j]\in_R\beta$。假设 α 先于 β 被选取。于是，$\beta=h_2(\alpha)$ 的条件概率等于 $\hat{g}_{h_2(\alpha)}^2$。在 $\beta=h_2(\alpha)$ 的条件下，"$\pi[i]\in_R\mathcal{D}_i$，$\pi[j]\in_R\mathcal{D}_j$ 满足 $\pi[j]=h[\pi[i]]$"的条件概率至少是 $1/|\alpha|$。原因如下。根据定义可知，$h_2(\alpha)$ 中的每个 u 都使得 $|h^{-1}(u)\cap\alpha|$ 是奇数，而奇数不可能等于 0！因此，无论我们选取哪个 $\pi[j]\in_R h_2(\alpha)$，都存在 $w\in\alpha$ 使得 $h(w)=\pi[j]$，而且将这样的 w 取为 $\pi[i]$ 的条件概率至少是 $1/|\alpha|$。综上所述，我们得到

$$\sum_{\alpha}\frac{1}{|\alpha|}\hat{f}_{\alpha}^2\hat{g}_{h_2(\alpha)}^2\leqslant\mathop{E}_{\mathcal{D}_i,\mathcal{D}_j}[I_r] \tag{22.10}$$

(22.10)式的左端与引理 22.29 中的表达式非常相似(但不完全相同)。根据引理 22.29 可知

$$2\delta_r \leqslant \sum_{\alpha} \hat{f}_{\alpha}^2 \hat{g}_{A_2(\alpha)}(1-2\rho)^{|\alpha|}$$

但是，由于容易看到 $(1-2\rho)^{|\alpha|} \leqslant \frac{2}{\sqrt{\rho|\alpha|}}$ 成立，故我们有

485

$$2\delta_r \leqslant \sum_{\alpha} \hat{f}_{\alpha}^2 |\hat{g}_{A_2(\alpha)}| \frac{2}{\sqrt{\rho|\alpha|}}$$

调整上式可得

$$\delta_r \sqrt{\rho} \leqslant \sum_{\alpha} \hat{f}_{\alpha}^2 |\hat{g}_{A_2(\alpha)}| \frac{1}{\sqrt{|\alpha|}}$$

将柯西-西瓦兹不等式 $\sum_i a_i b_i \leqslant (\sum_i a_i^2)^{1/2}(\sum_i b_i^2)^{1/2}$ 运用到上式(其中 $\hat{f}_{\alpha}|\hat{g}_{A_2(\alpha)}|\frac{1}{\sqrt{|\alpha|}}$ 相当于 a_i，而 $\hat{f}_{\alpha}$ 则相当于 b_i)，我们得到

$$\delta_r \sqrt{\rho} \leqslant \sum_{\alpha} \hat{f}_{\alpha}^2 |\hat{g}_{A_2(\alpha)}| \frac{1}{\sqrt{|\alpha|}} \leqslant (\sum_{\alpha} \hat{f}_{\alpha}^2)^{1/2} (\sum_{\alpha} \hat{f}_{\alpha}^2 \hat{g}_{A_2(\alpha)}^2 \frac{1}{|\alpha|})^{1/2} \tag{22.11}$$

由于 $\sum_{\alpha} \hat{f}_{\alpha}^2 = 1$，故将(22.11)式两端同时取平方，再结合(22.10)式，可以知道在任意 r 上均有

$$\delta_r^2 \rho \leqslant \underset{\mathcal{D}_1 \cdot \mathcal{D}_r}{E}[I_r]$$

这就证明了(22.9)式，进而完成了定理 22.16 的证明。 ■

22.8　SET-COVER 的近似难度

在 SET-COVER 问题中，我们给定一个基础集合 $\mathcal{U}$ 和它的一族子集 S_1，S_2，…，S_n，并且这族子集的并集等于 $\mathcal{U}$，我们需要找出一个最小的子集族 I 使得 $\bigcup_{i\in I} S_i = \mathcal{U}$。满足 $\bigcup_{i\in I} S_i = \mathcal{U}$ 的任意子集族 I 称为 $\mathcal{U}$ 的一个集合覆盖，它的大小指的是 $|I|$。如果集合覆盖问题的一个算法在任意实例上都能输出一个大小至多为 OPT/ρ 的集合覆盖，其中 $\rho<1$ 且 OPT 是该实例的最小集合覆盖的大小，则称该算法是 SET-COVER 问题的一个 ρ-近似算法。

定理 22.31 ([LY94])　如果存在常数 $\rho>0$ 使得 SET-COVER 问题存在 ρ-近似算法，则 **P=NP**。具体地讲，对于任意 ε，$W>0$，存在一个多项式时间的变换算法 f 将 $2\mathrm{CSP}_W$ 实例转换为 SET-COVER 的实例使得：如果 $2\mathrm{CSP}_W$ 实例是正则的并且满足投影性质，则

$$\mathrm{val}(\varphi)=1 \Rightarrow f(\varphi) \text{ 有大小为 } N \text{ 的集合覆盖}$$

$$\mathrm{val}(\varphi)<\varepsilon \Rightarrow f(\varphi) \text{ 不存在大小为 } \frac{N}{4\sqrt{\varepsilon}} \text{ 的集合覆盖}$$

其中 N 依赖于 φ。

事实上，我们还可以证明更强的结论，参见证明后面的评注。

定理 22.31 的证明需要下面的构件。

定理 22.32　((k,l)-集合构件)　一个 (k,l)-集合构件包含基础集 B 和它的 l 个满足如下条件的子集 C_1，C_2，…，C_l：如果取自子集族 C_1，$\overline{C_1}$，C_2，$\overline{C_2}$，…，C_n，$\overline{C_n}$ 的至多

k 个集合构成$\mathcal{B}$的集合覆盖，则必然存在某个 i 使得 C_i 和$\overline{C_i}$都属于该集合覆盖。

下面的引理留作习题 22.13。

[486]

引理 22.33　存在一个算法，它以任意 k，ℓ 为输入并且在 $\mathrm{poly}(k, 2^\ell)$ 时间内输出一个(k, ℓ)-集合构件。

定理 22.31 的证明　现在我们证明定理 22.31。我们将定理 22.15 得到的 $2\mathrm{CSP}_W$ 实例归约为 SET-COVER 实例。

设 φ 是一个使得 $\mathrm{val}(\varphi)=1$ 和 $\mathrm{val}(\varphi)<\varepsilon$ 之一成立的 $2\mathrm{CSP}_W$ 实例，其中 ε 是某个任意小的常数。假设 φ 有 n 个变量和 m 个约束。令 Γ_i 表示第 i 个变量在其中作为第一个变量出现的所有约束构成的集合，Δ_i 表示第 i 个变量在其中作为第二个变量出现的所有约束构成的集合。

构造

构造一个(k, W)-集合构件$(\mathcal{B}; C_1, C_2, \cdots, C_W)$，其中 $k>2/\sqrt{\varepsilon}$。由于 φ 的每个变量都取值于$[W]$中，故我们可以将每个集合 C_u 与一个可能的变量取值 u 相关联。

SET-COVER 的实例如下。基础集是$[m]\times\mathcal{B}$，我们可以将它视为$\mathcal{B}$的 m 个复制，每个复制对应于 φ 的一个约束。SET-COVER 问题实例中子集合的个数为 nW。对于 φ 中的每个变量 $i\in[n]$和每个可能的取值 $u\in[W]$，实例中含有一个子集 $S_{i,u}$；$S_{i,u}$是下列集合的并集：$\{r\}\times C_u$(任意 $r\in\Delta_i$)和$\{r\}\times\mathcal{B}\setminus C_{\pi(i)}$(任意 $r\in\Gamma_i$)。正是由于我们按照这种方式使用补集操作才得以将(满足投影性质的)2CSP 实例映射为 SET-COVER 实例。

分析

如果 $2\mathrm{CSP}_W$ 实例 φ 是可满足的，则我们可以给出一个大小为 n 的集合覆盖。令 π：$[n]\to W$ 是一个满足性赋值，其中 $\pi[i]$是第 i 个变量的取值。我们断言，由 n 个子集构成的集族$\{S_{i,\pi[i]}: i\in[n]\}$是一个集合覆盖。为了说明它确实是一个集合覆盖，只需证明其中所有子集的并集包含了$\{r\}\times\mathcal{B}$，其中 r 是任意约束。但这是显然的，因为如果 i 是 r 的第 1 个变量而 j 是 r 的第 2 个变量，则根据 $S_{i,\pi[i]}$和 $S_{j,\pi[j]}$的定义可知，$S_{i,\pi[i]}$包含了$\{r\}\times C_{\pi[i]}$并且 $S_{j,\pi[j]}$包含了$\{r\}\times\mathcal{B}\setminus C_{\pi[j]}$。因此，$S_{i,\pi[i]}$和 $S_{j,\pi[j]}$的并集包含了$\{r\}\times\mathcal{B}$。

另一方面，如果 $2\mathrm{CSP}_W$ 实例 φ 中仅有小于 ε 比例的约束是可被同时满足的，则我们断言所构造的 SET-COVER 实例的任意集合覆盖必然含有至少 nT 个子集。其中 $T=4\dfrac{1}{\sqrt{\varepsilon}}$。我们用反证法证明这个论断。假设 SET-COVER 实例存在规模小于 nT 的集合覆盖 I，则我们可以如下地构造 $2\mathrm{CSP}_W$ 实例 φ 的一个概率型赋值。对于任意变量 i，如果集合 $S_{i,u}$属于集合覆盖 I，则称 u 与变量 i 相关。我们从所有与变量 i 相关的值中随机地取出一个作为变量 i 的取值。由于我们对 k 的取法确保了 $8T<k$，故我们只需证明下面的论断成立。

论断：如果 $8T<k$，则 φ 中被上述随机赋值满足的约束的个数的数学期望大于$\dfrac{m}{16T^2}$。

证明　如果与某个变量相关的值少于 $4T$ 个，则称该变量是良性的。与每个变量相关的值的个数的平均值小于 T，因此，只有不到四分之一的变量使得它相关的值的个数大于 $4T$。于是，只有不到四分之一的变量不是良性的。

由于 2CSP 实例 φ 是正则的，故每个变量出现在相同个数的子句中。进而，含有非良性变量的约束在 φ 的所有约束中所占的比例小于 $2\times1/4=1/2$。因此，在超过一半的约束中，两个变量都是良性的。令 r 是这样一个约束并且 i，j 是它的两个变量。

[487]

于是，$\{r\}\times B$被并集$\bigcup S_{i,u}$和$\bigcup S_{j,v}$覆盖，其中并操作分别作用在与 i 和 j 相关的所有值上。由于$8T<k$，故(k, W)-集合构件的定义将意味着基础集B的任意大小小于$8T$的集合覆盖必然包含两个互补的子集。由此，我们断言，存在分别与 i 和 j 相关的两个值 u，v 使得$h(u)=v$。于是，当我们从分别与 i 和 j 相关的所有值中随机选择一个作为 i 和 j 的取值时，为 i 选中 u 并且为 j 选中 v 的概率至少是$\frac{1}{4T}\times\frac{1}{4T}=\frac{1}{16T^2}$。亦即，$\varphi$的第 r 个约束被随机赋值满足的概率至少是$\frac{1}{16T^2}$。因此，由数学期望的线性性质可知，论断成立。继而定理 22.31 成立。 ■

评注 22.34 同样的证明过程实际上可以得到如下更强的结论：存在常数 $c>0$ 使得，如果在 $\alpha=c/\log n$ 上 SET-COVER 存在 α-近似算法，则 $\mathbf{NP}\subseteq\mathbf{DTIME}(n^{O(\log\log n)})$。证明这一结论的思想是使用莱斯并行重复定理，其中重复次数 t 是超常数的，因此可靠性参数是 $1/\log n$。但是，归约过程的时间复杂度是 $n^{O(t)}$，它也略高于多项式时间。

22.9　其他 PCP 定理概述

正如本章开头的介绍所说，证明各种问题的不可近似性需要证明新的 **PCP** 定理。我们已经看到了一个例子，即哈斯塔德 3-位 **PCP** 定理。本节概述 **PCP** 定理的一些变形，它们都是人们已经证得的结果。

22.9.1　具有亚常数可靠性参数的 PCP 定理

在我们对定理 22.15 的表述方式中，可靠性参数是任意小的常数 2^{-t}。从定理的证明过程容易看到，由于用于证明 **NP** 难度的归约需要考虑由约束构成的所有 t 元组，故该归约的时间复杂度是 n^t。因此，如果 t 大于任何常数，则归约的时间复杂度将变为超多项式时间。尽管如此，有几个难度结果仍然在 t 上使用了超常数的值。这些结果的目的不是为了证明 **NP** 难度，而是为了在"**NP** 不存在 $n^{\log^c n}$ 时间的确定型算法"这一假设条件下证明良好的近似算法不存在。（值得注意的是，该假设条件仍是十分可信的。）在评注 22.34 中，我们已经提到了这种结果。

目前的一个待决问题是，在比任意常数还小的 ρ 上证明"求 2CSP 的 ρ-近似解"是 **NP** 难的。如果所考虑的不是 2CSP 而是 3CSP 或 4CSP，则用较大的字母表确实可以在概率可验证明（**PCP**）中得到低于任意常数的可靠性参数并且保持验证者的运行时间是多项式的[RS97]。通常，这些结果就足以满足各种应用了。

22.9.2　平摊的查验复杂度

某些应用要求二进制字母表上的概率可验证明（**PCP**）系统的查验次数（它可以是任意大的常数）和可靠性参数之间具有较紧的关系。这种关系可以通过一个相关的参数来表示，这个常数称为无约束位复杂度（Free Bit Complexity）[FK93，BS94]。该参数如下定义。假设查验次数是 q。在验证者取定随机位串和 q 个查验地址之后，这 q 个地址所包含的位串存在 2^s 种可能的情况。如果验证者仅在其中 t 个位串上进入接受状态，则我们称相应的概率可验证明（**PCP**）系统的无约束位参数（Free Bit Parameter）

为 log$t$(注意，这个数不一定是整数)。事实上，为了证明 MAX-INDSET 和 MAX-CLIQUE 的近似难度，只需考虑平摊无约束位复杂度(Amortized Free Bit Complexity)[BGS95]。平摊无约束位复杂度定义为 $\lim_{s\to 0} f_s/\log(1/s)$，其中 f_s 是验证者为了确保可靠性参数不超过 s 而完备性参数至少为 1/2 所需的无约束位的个数。哈斯塔德构造的概率可验证明系统的平摊无约束位复杂度趋向于 0[Has96]。也就是说，对于任意 $\varepsilon>0$，他为 **NP** 语言构造的 **PCP**-验证者使用了 $O(\log n)$ 个随机位和 ε 个平摊的无约束位，并且完备性参数是 1。然后，他利用这个概率可验证明(**PCP**)系统证明了在任意小的 $\varepsilon>0$ 上求 MAX-INDSET 的 $n^{1-\varepsilon}$-近似解是**NP**难的。当然，他在证明过程中借助了[FGL+91，FK93，BS94，BGS95]等工作中的基本框架。

22.9.3　2 位测试和高效傅里叶分析

在哈斯塔德定理给出的研究方向上，最新的研究进展用到了源自傅里叶分析的更强大的思想。哈斯塔德证明中的傅里叶分析难以用上"所查验的函数是布尔函数"这一事实。但是，卡恩(Kahn)，卡莱(Kalai)和利尼亚尔(Linial)[KKL88]，弗里德古特(Friedgut)[Fri99]，布尔干(Bourgain)[Bou02]这几篇论文已经深入地研究了布尔函数的傅里叶系数的性质，后来的研究结果表明这些性质在分析 **PCP**-验证者时非常有用。(也可以参见注记 22.21.)这些性质的主要优点是用它设计的验证者将只需查验证明中的 2 个位。这种验证者可以用来证明很多图问题的近似难度，包括 VERTEX-COVER，MAX-CUT 和 SPARSEST-CUT。

该方向上取得的这些新结果都沿用了哈斯塔德的整体证明思想，也就是说，它们都在证明：如果验证者以较高的概率接受某个给定的函数，则该函数将只有少数几个较大的傅里叶系数(推论 22.25 的证明就是这样的例子)。但是，哈斯塔德基本测试(甚至 22.6 节中的长编码测试)本质上需要验证者查验证明中的 3 个位，我们试着简要地说明其中的原因。为简单计，我们考虑长编码测试。我们只对长编码测试进行了简单的分析就得到了引理 22.24 的结论：

$$\sum_{\alpha} \hat{f}_\alpha^2 (1-2\rho)^{|\alpha|} \geqslant 2\delta$$

其中 $1/2+\delta$ 是函数被验证者接受的概率。基于这一事实，推论 22.25 断言函数 f 至少有一个傅里叶系数至少为 $c=c(\delta,\rho)>0$。推论 22.25 是哈斯塔德证明中的一个关键步骤，因为它使我们得知了：f 与长编码的少量码字之间存在一定的不易察觉的联系。

我们也可以设计一个类似的 2-位测试。上述分析的第一个部分可以套用，只是结论中的立方要换成平方，得到

$$\sum_{\alpha} \hat{f}_\alpha^2 (1-2\rho)^{|\alpha|} \geqslant 2\delta \tag{22.12}$$

如果 f 不是布尔函数，则(22.12)式不足以让我们得出结论：f 的某个傅里叶系数至少是 $c=c(\delta,\rho)>0$。但是，布尔干(Bourgain)给出的下述引理却使得我们可以在布尔函数 f 上得出这样的结论。如果函数 $f:\{0,1\}^n\to\{0,1\}$ 的取值只依赖于 n 个输入变量中的 k 个，则我们称函数 f 是一个 k-子团。注意，巴塞弗恒等式表明，k-子团至少有一个傅里叶系数 [489]
大于等于 $1/2^k$。下面的引理表明，如果布尔函数 f 使得在 $t>1/2$ 时(22.12)式的左端大于等于 $1-\rho'$，则 f 接近于一个 k-子团而且 k 是比较小的值。

引理 22.35 ([Bou02])　对于任意 $\varepsilon,\delta>0$ 和整数 r，均存在一个常数 $k=k(r,\varepsilon,\delta)$

使得：如果

$$\sum_{S:|S|>k} \hat{f}_S^2 < \frac{1}{2^{1/\varepsilon}}$$

则 f 与某个 k-子团在 $1-\delta$ 比例的输入上取相同的函数值。

我们怀疑，傅里叶分析在 **PCP** 构造中还有其他的用处。

22.9.4　唯一性游戏和阈值结果

哈斯塔德的思想可以得出几个问题的近似阈值。但是，为 SET-COVER 和 MAX-CUT 等其他一些问题寻找近似阈值仍是待决问题。2002 年，科特(Khot)[Kho02]提出了一个新的复杂性理论猜想，这个猜想被称为唯一性游戏猜想(Unique Game Conjecture)，简称 UGC。它强于 **P**≠**NP**，但仍然与人们现有的知识是一致的。该猜想考虑 $2\mathrm{CSP}_W$ 中约束函数是[W]上的置换映射时的特殊情况。换句话说，如果约束 φ_r 的两个变量是 i，j，则约束函数 h 是从[W]到[W]的双射。这样，所有变量的赋值 u_1，u_2，…，u_n 满足约束 φ_r 当且仅当 $u_j=h(u_i)$。根据 UGC 可知，对于任意常数 ε，$\delta>0$，存在字母表大小 $W=W(\varepsilon,\delta)$ 使得：不存在多项式时间算法以满足 $\mathrm{val}(\cdot)>1-\varepsilon$ 的任意 $2\mathrm{CSP}_W$ 实例为输入并输出一个赋值使得输入实例中 δ 比例的约束被满足。[⊖]

科特指出，现有的算法技术似乎都无法设计出这种算法。(最近几年，人们为此付出了大量的努力，都未能找出这样的算法，这似乎进一步肯定了科特的看法。)科特还指出，唯一性游戏猜想蕴含了几个较强的近似难度。简明扼要地讲，这是由于，如果我们从具有唯一性约束的 $2\mathrm{CSP}_W$ 实例开始，则哈斯塔德的傅里叶分析技术(结合引理 22.35 中布尔函数的傅里叶分析的结论)可以发挥更大的作用。

随后，在假设 UGC 为真的条件下，人们借助前面提到的高级傅里叶分析技术，证明了大量关于近似难度的阈值结果。例如，UGC 意味着，VERTEX-COVER 问题在任意 $\delta>0$ 上均不存在多项式时间的 $1/2+\delta$ -近似算法[KR08]。类似地，MAX-CUT 问题在任意 $\delta>0$ 上也不存在 $0.878+\delta$ -近似算法[KKMO05，MO005]。这两个结果都是阈值结果，因为人们已经为这两个问题找到了达到相应近似比的近似算法。

因此，证明或否证唯一性游戏猜想都非常有意义。算法设计者们尝试了利用半正
490 定规划中的巧妙的工具来否证这个猜想。目前，唯一性游戏猜想是否成立似乎仍处于一个十分微妙的境地。但人们已经知道，要确定该猜想是否成立，只需进一步考虑“约束函数 h 是线性函数(亦即，所有约束都是仅含两个变量的模 W 的方程)”的这种特殊情况。

22.9.5　与等周问题和度量空间嵌入之间的联系

度量空间(X,d)由点集 X 和距离函数 d 构成，其中 d 将取自 X 的任意点对映射为一个非负实数并且满足：(a)$d(i,j)=0$ 当且仅当 $i=j$；(b)$d(i,j)+d(j,k)\geqslant d(i,k)$，这个不等式称为三角不等式。空间$(X,d)$到空间$(Y,d')$的一个嵌入是一个函数 $f:X\to Y$。嵌入的扭曲度定义为 $\dfrac{d'(f(i),f(j))}{d(i,j)}$，$\dfrac{d(i,j)}{d'(f(i),f(j))}$ 这两个数量

⊖　事实上，科特(Khot)给出的 UGC 更强，它断言求解这个问题是 NP-难的。

在所有点对(i, j)上达到的最大值。在算法设计和数学中，找出将一个度量空间族嵌入到另一个度量空间族所需的最小扭曲度具有非常重要的意义。一种有意义的特殊情况是，考虑宿主空间(Y, d')是某个 n 上的 L_1-度量空间$\mathcal{R}^n$的子集。也就是说，此时距离是通过 L_1-范数来定义的。布尔干(Bourgain)证明了，含有 n 个点的任意度量空间嵌入到 L_1-空间都需要$O(\log n)$的扭曲度。这一结果对于 SPARSEST-CUT 等图划分问题的算法设计而言非常重要。在为 SPARSEST-CUT 设计算法时，人们找到了一种称为 L_1^2 的范数。高伊曼斯(Goemans)和利尼亚尔(Linial)曾经猜想 L_1^2-空间嵌入到 L_1-空间只需 $O(1)$的扭曲度。如果这个猜想成立，则 SPARSEST-CUT 问题将存在 $O(1)$-近似算法。科特(Khot)和维什诺伊(Vishnoi)[KV05]用一个有趣的 L_1^2-空间来否证了上述猜想，这个空间的构造受到本章讨论的高级 **PCP** 定理的启发。构造过程的主要思想如下。由于 L_1-空间和割之间存在密切的联系，因此在构造图时必须对割的结构和等周性质进行紧控制。因此，科特和维什诺伊使用了一种类似于超方体的图，并利用傅里叶分析来证明它的等周性质。(参见注记 22.21。)

科特和维什诺伊的工作促进了人们在度量空间嵌入方面的研究，继而得到了其他扭曲度下界。

22.A　将 qCSP 实例转换成"精细"实例

利用本小节给出的 3 个论断，我们可以将 qCSP 的任意实例 φ 转换成一个精细的 2CSP 实例 ψ。

论断 22.36　*存在 CL-归约将 qCSP 的任意实例 φ 映射为一个 $2CSP_{\mathcal{P}}$ 实例 ψ 使得*

$$\mathrm{val}(\varphi) \leqslant 1-\varepsilon \Rightarrow \mathrm{val}(\psi) \leqslant 1-\varepsilon/q$$

证明　给定 n 个变量 $u_1, \cdots, u_n$ 上含有 m 个约束的 qCSP 的实例 φ，我们如下构造变量 $u_1, \cdots, u_n, y_1, \cdots, y_m$ 上的 $2CSP_{\mathcal{P}}$ 实例 ψ。直观上看，变量 y_i 将用于刻画"第 i 个约束的 q 个变量的赋值"应该满足的限制条件，新增的约束则用于验证赋值的一致性。也就是说，如果第 i 个约束含有变量 u_j，则新增的约束将用于确保 u_j 的赋值确实与 y_i 刻画的限制条件是一致的。具体地讲，对于 φ 中依赖于变量 $u_{i_1}, \cdots, u_{i_q}$ 的约束 φ_i，我们添加 q 个约束 $\{\psi_{i,j}\}_{j\in[q]}$ 使得：$\psi_{i,j}(y_i, u_j)$ 为真当且仅当 y_i 所表示的 $u_{i_1}, \cdots, u_{i_q}$ 的赋值满足 φ_i 并且 $u_j \in \{0,1\}$ 与赋值 y_i 是一致的。注意，ψ 含有 qm 个约束。 [491]

显然，如果 φ 是可满足的，则 ψ 也是可满足的。假设 $\mathrm{val}(\varphi) \leqslant 1-\varepsilon$，并且 $u_1, \cdots, u_n, y_1, \cdots, y_m$ 是中 ψ 所有变量的一个赋值。于是，存在一个规模至少为 εm 的子集 $S \subseteq [m]$ 使得 $u_{i_1}, \cdots, u_{i_q}$ 不满足约束 φ_i 对 $\forall i \in S$ 成立。因此，对于任意 $i \in S$，必然至少存在一个 $j \in [q]$ 使得 $u_{i_1}, \cdots, u_{i_q}, y_i, \cdots, y_m$ 不满足约束 $\psi_{i,j}$。 ■

论断 22.37　*存在绝对常数 d 和一个 CL-归约将 $2CSP_W$ 的任意实例 φ 映射为 $2CSP_W$ 的另一个实例 ψ 使得*

$$\mathrm{val}(\varphi) \leqslant 1-\varepsilon \Rightarrow \mathrm{val}(\psi) \leqslant 1-\varepsilon/(100Wd)$$

并且 ψ 的约束图是 d-正则图。亦即，ψ 的每个变量恰好出现在 d 个约束中。

证明　设 φ 是一个 $2CSP_W$ 实例，$\{G_n\}_{n\in\mathbb{N}}$ 是一个显式的 d-正则扩张图族。我们的目标是确保 j 中的每个变量至多出现 $d+1$ 次(如果某个变量出现的次数少于这个数量，则我们可以通过添加仅含该变量的若干个约束来确保条件成立)。假设变量 u_i 出现在 k 个约束

中，其中 $k>1$。我们把 u_i 变为 k 个变量 y_i^1，…，y_i^k。在包含 u_i 的每个原始约束中，我们将 u_i 替换成 y_i^1，…，y_i^k 中的一个变量 y_i^j，使得包含 u_i 的所有原始约束经过替换之后所使用的 y_i^j 各不相同。然后，对于 G_n 中的每条边 $\langle j, j'\rangle$，我们再添加一个约束来要求"y_i^j 等于 $y_i^{j'}$"。对原始实例 φ 中的每个变量重复上述操作，使得每个变量恰好出现在至少 d 个相等约束和一个原始约束中。将最后得到的 2CSP 实例称为 ψ。注意，如果 φ 含有 m 个约束，则 ψ 至多含有 $m+dm$ 个约束。

显然，如果 φ 是可满足的，则 ψ 也是可满足的。假设 $\mathrm{val}(\varphi)\leqslant 1-\varepsilon$，并且 $\boldsymbol{y}$ 是中 ψ 所有变量的任意赋值。我们需要证明 ψ 中至少有 $\frac{\varepsilon m}{100W}$ 个约束不能被 $\boldsymbol{y}$ 满足。注意，对于在 φ 中出现了 k 次的任何变量 u_i，$\boldsymbol{y}$ 都包含了对 k 个变量 y_i^1，…，y_i^k 的赋值。根据 $\boldsymbol{y}$，我们可以如下地计算 j 中所有变量的一个赋值 $\boldsymbol{u}$：u_i 的赋值取 y_i^1，…，y_i^k 的赋值中出现次数大于 1 的值。令 t_i 表示变量 y_i^1，…，y_i^k 中赋值与 u_i 的赋值不一致的变量个数。注意，$0\leqslant t_i\leqslant k(1-1/W)$，其中 W 是字母表的大小。如果 $\sum_{i=1}^{n} t_i \geqslant \frac{\varepsilon}{4}m$，则得欲所证。事实上，由(22.1)式(参见 22.2.3 节)可知，此时 ψ 中至少有 $\sum_{i=1}^{n}\frac{t_i}{10W}\geqslant\frac{\varepsilon}{40W}m$ 个约束不能被 $\boldsymbol{y}$ 满足。

现在，假设 $\sum_{i=1}^{n} t_i<\frac{\varepsilon}{4}m$。由于 $\mathrm{val}(\varphi)\leqslant 1-\varepsilon$，故存在一个规模至少为 εm 的约束子集 S 使得我们定义的赋值 $\boldsymbol{u}$ 不满足 S 中的每个约束。这些约束也出现在 ψ 中，由于我们假设了 $\sum_{i=1}^{n} t_i<\frac{\varepsilon}{4}m$，故这些约束中至多有一半的约束使得 $\boldsymbol{y}$ 和 $\boldsymbol{u}$ 对其中的变量 u_i 赋值不相同。因此，赋值 ψ 中至少有 $\frac{\varepsilon}{2}m$ 个约束不能被 $\boldsymbol{y}$ 满足。 ■

论断 22.38　*存在绝对常数 d 和一个 CL-归约将满足 $d\geqslant d'$ 且具有 d'-正则约束图的任意 $2\mathrm{CSP}_W$ 实例 φ 映射为一个 $2\mathrm{CSP}_\Sigma$ 实例 ψ 使得*

$$\mathrm{val}(\varphi)\leqslant 1-\varepsilon\Rightarrow\mathrm{val}(\psi)\leqslant 1-\varepsilon/(10d)$$

[492] *并且 ψ 的约束图是 $4d$-正则扩张图而且图中任意顶点的关联边有一半是自环。*

证明　由 22.2.3 节可知，存在常数 d 和一个显式图族 $\{G_n\}_{n\in\mathbb{N}}$ 使得：对任意 n 而言，G_n 是 d-正则 n-顶点的 0.1-扩张图。

设 φ 是论断 22.38 中具有 d'-正则约束图的 $2\mathrm{CSP}_W$ 实例。通过在顶点上添加自环，我们可以假设 φ 的约束图中顶点度等于 d。此时，φ 的鸿沟至多缩小因子 d。现在，对于 G_n 中的每条边(其中 n 是 φ 中变量的个数)，我们再引入一个"空"约束——恒为真的约束。此外，我们在 G_n 的每个顶点上再添加 $2d$ 个自环，相应地我们引入 $2d$ 个空约束。我们将最终得到的实例记为 ψ。添加空约束使得实例中不能被满足的约束的个数占约束总数的比例至多缩减因子 4。而且，由于任意扩张图 H 都满足 $\lambda(H)\leqslant 1$ 并且 λ 参数还满足亚可加性(习题 21.7)，故 $\lambda(\psi)\leqslant\frac{3}{4}+\frac{1}{4}\lambda(G_n)\leqslant 0.9$，其中 $\lambda(\psi)$ 表示 ψ 的约束图的 λ 参数。 ■

本章学习内容

- **PCP** 定理是一种归约，它能将 **NP** 问题的 YES 实例和 NO 实例之间的鸿沟放大。本

章给出了迪纳尔对 **PCP** 定理的证明，它通过一系列小的组合步骤来实现鸿沟放大。而 **PCP** 定理的原始证明则利用代数工具和纠错码（几乎）一次到位地实现了鸿沟放大。

- 离散傅里叶变换是分析布尔函数的强有力的工具。在分析函数在"噪音"干扰下的行为时，离散傅里叶分析特别有用。
- **PCP** 定理可以立刻证得近似难度方面的一些结果。但是，要获得更强的近似难度，通常需要用更复杂的归约或者其他 **PCP** 定理。

本章注记和历史

正如第 11 章的注记所指出的那样，**PCP** 定理的证明首次于 1992 年发表在论文[AS92，ALM+92]的会议版本中。十几年以来，桑杰夫·阿罗拉等人给出的原始证明不断被简化。原始证明（其实还包括 **MIP**=**NEXP** 的证明）的整体思想都与定理 11.19 的证明非常相似。（事实上，在原始证明的简化过程中，定理 11.19 是原始证明中唯一幸存下来的部分。）但是，原始证明除了用到了沃尔什-哈达玛编码之外，还用到了基于低次多变量多项式的编码。这些编码也用到了类似于线性测试和局部解码的过程，但是这些过程的正确性证明相对而言却要难一些。原始证明还借用了自测试（Self-testing）和自纠错编程（Self-Correcting Program）[BLR90，RS92]等领域的直观思想，8.6 节曾对这两个领域进行了概述。在桑杰夫·阿罗拉等人给出的原始证明中，字母表削减过程也更复杂一些。改写后的原始证明可以在本书的网络版草稿中找到，本书的网站提供了这个草稿版本。我们从本书正式出版的版本中删除了 **PCP** 定理的原始证明，而采用了迪纳尔（Dinur）的证明。但我们认为，**PCP** 定理的原始证明有其独特的存在价值，并且它也可能会在将来的研究中继续发挥作用。 493

迪纳尔（Dinur）的主要贡献是简化了鸿沟放大引理的证明，他的结果允许我们递归地改进概率可验证明（**PCP**）系统的可靠性参数，从接近于 1 的值改进到距离 1 超过一个正常数的值。这样，概率可验证明（**PCP**）系统才得以采用较小的字母表。事实上，字母表削减过程是整个证明中唯一用到"证明验证"观点的部分，我们期待在未来几年的研究中字母表削减过程也能找到纯组合构造。一个相关的待决问题是为 **MIP**=**NEXP** 找出迪纳尔式的证明过程。

我们还注意到，迪纳尔的一般性策略不禁让人联想到扩张图的拉链构造和第 20 章中莱茵戈尔德（Reingold）为无向连通性给出的确定型对数空间算法。这意味着，不同研究领域之间的联系还有待人们建立和完善。

正如第 11 章末尾的注记中已经指出的那样，帕帕迪米特里奥（Papadimitriou）和杨纳卡卡斯（Yannakakis）[PY88]曾在 1988 年前后证明了：如果在某个 $\rho<1$ 上求 MAX-3SAT 问题的 ρ-近似解是 **NP** 难的，则一大批问题的 ρ'-近似解的计算也将是 **NP** 难的，其中 ρ' 依赖于具体的问题。因此，当 **PCP** 定理被发现之后，人们关注的焦点变成了为各种计算问题确定其准确的近似阈值，参见[BS94，BGS95]。几年之后，哈斯塔德在 MAX-CLIQUE [Hås96]和 MAX-3SAT[Hås97]建立了阈值结果，这标志着人们对近似难度的理解获得了巨大的突破。

并行重复问题源自福特劳（Fortnow），龙佩尔（Rompel）和西普赛尔（Sipser）的论文[FRS88]。在该论文中，他们错误地断言 $\mathrm{val}(\varphi^{t'})=\mathrm{val}(\varphi)^t$ 对任意 2CSP 实例 φ 和任意 $t\in\mathbf{N}$ 都成立。但是，福特劳[For89]很快找出了一个反例（参见习题 22.6）。在莱斯（Raz）的论文发表之前，拉皮多特（Lapidot）和萨米尔（Shamir）的论文[LS91]，费格（Feige）和洛瓦兹（Lovasz）

的论文[FL92]，都得出了 2CSP 的近似难度，但所用归约的时间复杂度都是超多项式的。韦尔比茨基(Verbitsky)[Ver94]，费格和吉蒋安(Kilian)[FK93]证明了莱斯定理(定理 22.15)的较弱形式。莱斯的证明所采用的技术扩展了莱兹波偌夫(Razborov)[Razb90]在通信复杂性中建立的技术。整个证明很漂亮，但却相当复杂。但最近，霍伦施泰因(Holenstein)[Hol07]对莱斯的证明进行了一些简化，这个简化的证明在本书的网络版草稿中可以找到。

引理 22.8 中 MAX-INDSET 的近似难度可以进一步改进为：对于任意 $\epsilon>0$，在 n 顶点图上求 MAX-INDSET 的 $n^{1-\epsilon}$-近似解是 **NP** 难的。这个结果源自[Hás96]，它的基础是其他人所做的一大批工作[FGL+91，AS92，ALM+92，BS94，BGS95]。引理 22.8 中基于扩张图的归约源自[AFWZ95]。注意，MAX-INDSET 的 $1/n$-近似解是很容易求得的：输出任意一个单独的顶点，它肯定是一个独立集。因此，改进后的结论也是一个阈值定理。

SET-COVER 问题的近似难度源自伦德(Lund)和杨纳卡卡斯[LY94]，该论文首次明确地使用了 $2CSP_W$ 的投影性质。[Aro94，ABSS93]指出了这种技术的重要性，这两篇论文的作者在证明其他结果时称这种归约技术为标签覆盖(Label Cover)。该技术目前已经成为 **PCP** 文献中的一种普遍技术。

费格[Fei96]曾经证明了 SET-COVER 问题的一个阈值结果：在任意 $\delta>0$ 上，求 SET-COVER 的 $(1+\delta)/\ln n$-近似解是 **NP** 难的，同时 SET-COVER 也存在一个简单的 $(1/\ln n)$-近似算法。

要了解如何证明基本的近似难度，请参阅阿罗拉和伦德在 1995 年前后发表的综述[AL95]。
494 要了解基于傅里叶分析得到的研究结果，请参阅科特(Khot)最近发表的综述[Kho05]。

习题

22.1　证明(22.1)式给出的等式。

22.2　设 $G=(V, E)$是一个 λ-扩张图，其中 $\lambda\in(0, 1)$。S 是 V 的一个满足 $|S|=\beta|V|$ 的子集，其中 $\beta\in(0, 1)$。令随机变量构成的元组 $(X_1, \cdots, X_l)$表示 G 中均匀选取的一条长度为 $l-1$ 的路径。证明：

$$(\beta-2\lambda)^l \leqslant \Pr[\forall_{i\in[l]} X_i \in S] \leqslant (\beta+2\lambda)^l$$

22.3　设 S_t表示投掷 t 枚匀质硬币所对应的二项分布，亦即，$\Pr[S_t=k]=\binom{t}{k}2^{-t}$。证明：对任意 $\delta<1$，S_t 和 $S_{t+\delta t}$之间的统计距离都不超过 10δ。(统计距离的定义请参见 A.2.6 节。)

22.4　证明：在任意非负随机变量 V 上都有 $\Pr[V>0]\geqslant E[V]^2/E[V^2]$。

22.5　本题考虑用另一种方法证明字母表削减引理(引理 22.6)，这里我们将使用长编码而不再用沃尔什-哈达玛编码。我们已经看到，集合 $\{0, \cdots, W-1\}$ 上的一个长编码是一个如下的函数 $LC: \{0, \cdots, W-1\}\to\{0, 1\}^{2^W}$；对于任意的 $i\in\{0, \cdots, W-1\}$和一个函数 $f: \{0, \cdots, W-1\}\to\{0, 1\}$(我们将函数 f 等同于取自$[2^W]$的一个编号)，$LC(i)$的第 f 个位 $LC(i)_f$就是 $f(i)$。如果函数 $L: \{0, 1\}^{2^W}\to\{0, 1\}$使得 $L=LC(i)$对某个 $i\in\{0, \cdots, W-1\}$成立，则称函数 L 是长编码的一个码字。

(a) 证明：长编码 LC 是一个编码距离等于 1/2 的纠错码。也就是说，对任意 $i\neq j\in\{0, \cdots, W-1\}$而言，$LC(i)$和 $LC(j)$的分数汉明距离是 1/2。

(b) 证明：LC 是可以局部解码的。也就是说，给出一个如下的算法。它的输入包含两个部分：其一是对函数 $L:\{0,1\}^W\to\{0,1\}$ 的随机访问能力，其中 L 与 $LC(i)$ 是 $(1-\varepsilon)$-接近的；其二是函数 $f:\{0,\cdots,W-1\}\to\{0,1\}$，算法需要在只查验 L 的函数值 2 次的条件下至少以 0.9 的概率输出 $LC(i)_f$。

(c) 设 $L=LC(i)$ 对某个 $i\in\{0,\cdots,W-1\}$ 成立。证明：在任意 $f:\{0,\cdots,W-1\}\to\{0,1\}$ 上均有 $L(f)=1-L(\overline{f})$，其中 $\overline{f}$ 是 f 的负函数(也就是说，$\overline{f}(i)=1-f(i)$ 在任意 $i\in\{0,\cdots,W-1\}$ 上成立)。

(d) 设 T 是允许对 $L:\{0,1\}^W\to\{0,1\}$ 进行随机访问的算法，它完成如下操作：

(1) 选择从 $\{0,\cdots,W-1\}$ 到 $\{0,1\}$ 的随机函数 f；

(2) 如果 $L(f)=1$，则输出 TRUE；

(3) 否则，如下选择函数 $g:\{0,\cdots,W-1\}\to\{0,1\}$：对于任意 $i\in\{0,\cdots,W-1\}$，若 $f(i)=0$ 则令 $g(i)=0$，否则令 $g(i)$ 是 $\{0,1\}$ 中的一个随机值。

(4) 如果 $L(g)=0$，则输出 TRUE；否则，输出 FALSE。

证明：如果 L 是长编码的一个码字(即，$L=LC(i)$ 对某个 i 成立)，则 T 以概率 1 输出 TRUE。

证明：如果 L 是一个非零线性函数但不是长编码的码字，则 T 至多以 0.9 的概率输出 TRUE。 495

(e) 证明：LC 是局部可验证的。也就是说，给出一个允许对 $L:\{0,1\}^W\to\{0,1\}$ 进行随机访问的算法使得它只对 L 的函数值进行常数次查验，并且：如果 L 是长编码的码字，则算法以概率 1 输出 TRUE；如果 L 与长编码的任意码字都不是 0.9-接近的，则算法至少以 1/2 的概率输出 FALSE。

(f) 利用上面的测试，给出字母表削减引理(引理 22.6)的另一种证明。

22.6 ([For89, Fei89])考虑大小为 4 的字母表上的如下 2CSP 实例 φ(这里，我们将字母表等同于 $\{0,1\}^2$)。实例 φ 有四个变量 $x_{0,0}$，$x_{0,1}$，$x_{1,0}$，$x_{1,1}$ 和四个约束 $C_{0,0}$，$C_{0,1}$，$C_{1,0}$，$C_{1,1}$。约束 $C_{a,b}$ 的两个变量是 $x_{a,*}$ 和 $x_{*,b}$，该约束为真当且仅当 $x_{0,a}=x_{1,b}$ 且 $x_{a,b}\in\{0a,1b\}$。

(a) 证明：$\mathrm{val}(\varphi^{*t})=\mathrm{val}(\varphi)$，其中 φ^{*t} 是 22.3.1 节中在字母表 W^t 上定义的 2CSP 实例，它是 φ 的 t 次并行重复。

(b) 证明：对任意的 t 均有 $\mathrm{val}(\varphi^{*t})=\mathrm{val}(\varphi)^{t/2}$。

22.7 (唯一性游戏的可解性)第 22.9.4 节中曾遇到唯一性游戏，它是 $2CSP_W$ 问题的特殊情形，其实例的每个约束 φ_r 的函数 h 都是 $[W]$ 上的置换。换句话说，如果 φ_r 中的两个变量是 i 和 j，则变量赋值 $u_1,u_2,\cdots,u_n$ 满足约束 φ_r 当且仅当 $u_j=h(u_i)$。证明：存在一个多项式时间算法在任意给定的唯一性游戏实例上输出该实例的一个满足性赋值(如果满足性赋值存在的话)。

22.8 证明推论 22.25。

22.9 证明论断 22.36 的证明过程所得的概率可验证明(**PCP**)系统(参见第 11 章)满足投影性质。

22.10 本习题讨论布尔函数的噪音敏感度，这个概念与 22.5.3 节中注记 22.21 的部分内容联系密切。令 $f:\{\pm1\}^n\to\{\pm1\}$ 且 $I\subseteq[n]$，而 M_I 是如下的分布：$\mathbf{z}\in_R M_I$ 可以这样随机选取，当 $i\in I$ 时 z_i 独立地以 1/2 的概率取 +1 而以 1/2 的概率取 −1，当 $i\notin I$ 时 $z_i=+1$。f 在 I 上的方差定义为 $\Pr_{\mathbf{x}\in_R\{\pm1\}^n,\mathbf{z}\in_R M_I}[f(\mathbf{x})\neq f(\mathbf{xz})]$。

假设 f 在 I 上的方差小于 ε。证明：存在一个函数 $g:\{\pm1\}^n\to\mathbf{R}$ 使得：(1)g 不依赖于 I 中的坐标位置；(2)g 与 f 是 10ε-接近的(即，$\Pr_{x\in_R\{\pm1\}^n}[f(x)\neq g(x)]<10\varepsilon$)。你能构造出值域为 $\{\pm1\}$ 的这样一个函数 g 吗？

22.11　对于 $f:\{\pm1\}^n\to\{\pm1\}$ 和 $x\in\{\pm1\}^n$，我们定义 $N_f(x)$ 是如下坐标位置 i 的个数：如果令 y 是翻转 x 的第 i 个坐标分量得到的向量(亦即，$y=xe^i$，其中 e^i 除了在第 i 个坐标分量上取 -1 之外，其他坐标分量都等于 $+1$)，则 $f(x)\neq f(y)$。我们定义 f 的平均敏感度 $\mathrm{as}(f)$ 为 $N_f(x)$ 在 $x\in_R\{\pm1\}^n$ 上的数学期望。

(a) 证明：在任意平衡函数 $f:\{\pm1\}^n\to\{\pm1\}$(即 $\Pr[f(x)=+1]=1/2$)上，$\mathrm{as}(f)\geqslant1$。

(b) 令 f 是从 $\{\pm1\}^n$ 到 $\{\pm1\}$ 的平衡函数并且 $\mathrm{as}(f)=1$。证明：f 是一个坐标函数或者负坐标函数(亦即，$f(x)=x_i$ 或 $f(x)=-x_i$ 对某个 $i\in[n]$ 和任意 $x\in\{\pm1\}^n$ 成立)。(用注记 22.21 中等周问题的话来说就是：对于由超方体 $\{0,1\}^n$ 中一半顶点构成的顶点子集，如果恰有 2^{n-1} 条边离开该子集，则存在 i 使得该顶点子集就是由满足 $x_i=0$(或 $x_i=1$)的所有顶点构成的子集。)

496

22.12　([KM91])本习题要求读者用傅里叶分析来给出戈德赖希-勒维定理(定理 9.12)的另一种证明。

(a) 对于任意函数 $f:\{\pm1\}^n\to\mathbf{R}$，令 $\widetilde{f}_{\star\star}=\sum_{\beta\in\{0,1\}^{n-k}}\widehat{f}_{\star\beta}$，其中 $\cdot$ 表示字符串拼接操作，并且将 $\{0,1\}^k$ 中的位串以显而易见的方式等同于 $[n]$ 的子集合。证明：

$$\widetilde{f}_{\alpha\star}=\mathop{E}_{x,x'\in_R\{0,1\}^k,y\in_R\{0,1\}^{n-k}}[f(x\cdot y)f(x'\cdot y)]$$

(b) 证明：对任意 $\alpha\in\{0,1\}^k$，有

$$\widetilde{f}_{\alpha\star}=\mathop{E}_{x,x'\in_R\{0,1\}^k,y\in_R\{0,1\}^{n-k}}[f(x\cdot y)f(x'\cdot y)\chi_\alpha(x)\chi_\alpha(x')]\quad(22.13)$$

(c) 给出一个算法 Estimate，它的输入包括 $\alpha\in\{0,1\}^k$，$\varepsilon>0$ 和神喻函数 $f:\{\pm1\}^n\to\{\pm1\}$，要求算法在 $\mathrm{poly}(n,1/\varepsilon)$ 时间内以 $1-\varepsilon$ 的概率输出 f_α 的一个估计值使得其精度在 ε 误差范围内。

(d) 给出一个算法 LearnFourier，它的输入包括 $\varepsilon>0$ 和神喻函数 $f:\{\pm1\}^n\to\{\pm1\}$，要求算法在 $\mathrm{poly}(n,1/\varepsilon)$ 时间内输出由 $\mathrm{poly}(1/\varepsilon)$ 个字符串构成的集合 L 使得在任意 $\alpha\in\{0,1\}^n$ 上“如果 $|\widehat{f}_\alpha|>\varepsilon$ 则 $\alpha\in L$”至少以 0.9 的概率成立。

(e) 证明：上面的算法蕴含了定理 9.12。

22.13　证明引理 22.33，但只要求设计一个随机算法即可。

22.14　将习题 22.13 中的算法去随机化。

22.15　([ABSS93])习题 11.16 讨论了在有理系数的线性方程组中近似地找出有解的最大子方程组这一问题的难度。证明：存在 $\varepsilon>0$ 使得求该问题的 $n^{-\varepsilon}$-近似解是 **NP** 难的。

22.16　([PY88])假设我们只讨论 MAX-3SAT 的所有如下特殊实例：每个变量至多出现在 5 个子句中。证明：仍存在常数 $\rho<1$ 使得求该问题的 ρ-近似解是 **NP** 难的。

22.17　([PY88])MAX-CUT 问题的输入是一个图 $G=(V,E)$，要求将图的所有顶点划分为两个子集 S，$\overline{S}$ 使得介于这两个子集之间的边的条数 $|E(S,\overline{S})|$ 达到最大值。证明：存在常数 $\rho<1$ 使得求该问题的 ρ-近似解是 **NP** 难的。

497

为什么线路下界如此困难

证明下界的主要困难是算法的存在性。

——史蒂芬·卢吉奇(Steven Rudich)

为什么人们至今仍无法证明一般线路的较强下界呢？尽管人们在受限的线路族上证得了一些引人注目的下界(正如第 14 章所述)，但是人们在证明一般布尔线路的下界时却陷入了徒劳无功的困境。

1994 年，莱兹波偌夫(Razborov)和卢吉奇(Rudich)[RR94]用他们特有的视角解释了证明线路下界的现有方法的局限性。他们为线路下界定义了"自然数学证明"的概念。他们指出，线路下界的现行论证方法都采用了这种自然的数学证明。同时，他们还证明了，"用这种方法证得线路的更强下界"会违背 $\mathbf{P}\neq\mathbf{NP}$ 这一猜想的某种更强的形式。具体地讲，这种更强的猜想指的是：存在强单向函数使得任意亚指数时间算法都无法求出它的逆函数。现有证据表明，这种强单向函数可能确实存在(例如第 9 章中介绍的因数分解函数和离散对数函数等等都可能是强单向函数)。由此，他们得出结论，用现有方法证明一般线路的下界将存在固有的困难。

用现代的观点看，莱兹波偌夫和卢吉奇的结论类似于二十世纪 70 年代时人们得到的关于"对角线方法的局限性"的结论(参见第 3 章)。尤其令人惊讶的是，计算复杂性(亦即，强单向函数的存在性)在这里被用来阐明关于计算复杂性的一个元数学问题："为什么人们一直证明不了 $\mathbf{P}\neq\mathbf{NP}$？"这很好地印证了本书开头曾经指明的论断——计算上的难解性与数学上的可解性或可证明性之间是紧密联系的。

本章的组织如下。23.1 节定义自然证明，23.2 节讨论为什么这样的证明确实是"自然的"。然后，23.3 节在人们广泛认同的假设条件下证明"自然证明技术无法证得 $\mathbf{NP}\not\subseteq\mathbf{P}_{/poly}$"。

我们能设计出线路下界的证明技术来克服"自然证明障碍"吗？23.4 节将给出这样一个有意义的例子。最后，23.5 节厘清自然证明障碍的一些哲学误区和作者个人的见解。

23.1 自然证明的定义

设 $f:\{0,1\}^n\to\{0,1\}$ 是一个布尔函数且 $c\geqslant 1$ 是一个数。对"f 不存在规模为 n^c 的线路"的任何证明都可以视为对"f 具有'规模为 n^c 的任意线路都不具备的'某种性质"的阐述。也就是说，可以认为这种证明也就是给出布尔函数上的某个谓词 $\mathcal{P}$ 使得 $\mathcal{P}(f)=1$，但

$$\mathcal{P}(g)=0 \text{ 对任意 } g\in\mathbf{SIZE}(n^c) \text{ 成立} \tag{23.1}$$

条件(23.1)称为 n^c-有用性。谓词 $\mathcal{P}$ 称为自然的，如果它还满足如下的另外两个条件：

可构造性：存在一个时间复杂度为 $2^{O(n)}$ 的算法使得它在输入函数 $g:\{0,1\}^n\to\{0,1\}$ (的真值表)上输出 $\mathcal{P}(g)$。注意，由于真值表的大小为 2^n，故该算法的运行时间是输入大小的多项式。

广泛性：随机函数 $g:\{0,1\}^n\to\{0,1\}$ 满足 $\mathcal{P}(g)=1$ 的概率至少为 $1/n$。

我们将在 23.2 节中讨论设置上述两个条件的动机。现在，我们仅需注意到，广泛性

条件与 n^c-有用性条件之间并不矛盾，这是由于具有多项式规模线路的布尔函数在所有可能的布尔函数中只占少部分(参见定理 6.21 的证明)。下面的定理断言，在合理的假设条件下，自然证明无法用来证明"函数不属于 P_{poly}"。

定理 23.1 (自然证明[RR94])　假设亚指数强单向函数是存在的，那么存在常数 $c\in\mathbf{N}$使得 n^c-有用的自然谓词$\mathcal{P}$是不存在的。

单向函数的定义参见第 9 章(第 9.2 节)，亚指数强单向函数指的是这样一个单向函数，即使用时间复杂度为 2^{n^ε} 的算法也无法求得该单向函数的逆函数(其中 $\varepsilon>0$ 是某个固定的值)。人们广泛地承认，这种强单向函数是存在的。定理 23.1 的证明推迟到 23.3 节才给出。这里首先解释为什么将这种谓词称为"自然的"。

例 23.2 我们给出两个谓词，以帮助大家深入理解自然证明的定义。

第一个谓词要求，$\mathcal{P}(g)=1$ 当且仅当 n 位布尔函数 g 的线路复杂度大于 $n^{\log n}$。由于 $n^c=o(n^{\log n})$对任意常数 c 成立，故该谓词是 n^c-有用的。该谓词也满足广泛性，这是由于随机布尔函数满足这个谓词的概率几乎等于 1(参见定理 6.21 的证明)。但是，人们还不知道该谓词是否满足可构造性。这是由于用简单直接的算法来验证该谓词需要枚举规模为 $n^{\log n}$ 的所有线路，而这个过程的时间开销为 $2^{n^{\log n}}$。

第二个谓词要求，$\mathcal{P}'(g)=1$ 当且仅当 g 在所有的 n 位输入上正确地求解了判定问题
499 3SAT。该谓词是可构造的。事实上，为了计算这个谓词，只需枚举所有的 n 位输入，然后利用 3SAT 问题的时间复杂度为 2^n 的平凡算法来验证 g 在所有输入上给出了正确答案。如果 $3SAT\notin\mathbf{SIZE}(n^c)$(当然，这还是一个待决问题)，则$\mathcal{P}'$也满足 n^c-有用性，因为它在 $\mathbf{SIZE}(n^c)$ 中的所有函数上均输出 0。然而，$\mathcal{P}'$不满足广泛性，因为它仅在一个函数上输出 1。◀

23.2　为什么自然证明是自然的

现在我们回顾一下前面得出的一些线路下界，看看这些下界是否隐式地用到了自然证明。(当然，这就说明了"自然证明"这一术语确实是"自然的"。)

例 23.3 ($\mathbf{AC}^0$)　证明"奇偶性函数不能由 $\mathbf{AC}^0$ 线路来计算"(参见 14.1 节)的主要步骤如下：(a)证明每个 $\mathbf{AC}^0$ 线路在至多限制 $n-n^\varepsilon$ 个输入位之后就会被简化变成一个常值函数；(b)证明奇偶性函数在至多限制 $n-n^\varepsilon$ 个输入位之后不可能取常数值。

显然，我们可以在 $2^{O(n)}$ 时间内验证函数 $f:\{0,1\}^n\to\{0,1\}$ 是否满足(a)中定义的性质。这只需枚举所有需要限制的变量集合，并在每个可能的限制集上枚举所有变量可能的 0/1 取值。因此，上述证明满足"可构造性"条件。而且，不难证明，在随机函数上限定其输入中的至多 $n-n^\varepsilon$ 个位不可能使得它变成常值函数(习题 23.2)，因此上述的证明也满足"广泛性"条件。◀

例 23.4 (双方通信复杂性)　要证明函数 f 的双方通信复杂度较高，只需证明在第 13 章引入的 $n\times n$ 矩阵 $M(f)$中不存在较大的单色矩形，其中 M 中(x,y)位置上存储的元素是 $f(x,y)$。考虑用算法验证上述条件的复杂度，其中算法的输入是 $M(f)$(亦即，一个长度等于 2^{2n}的位串)。论断"$M(f)$存在 $k\times l$ 的单色矩形"是一个 **coNP** 论断。事实上，这个论断对一般的 f 而言是 **coNP**-完全的(因为该论断等价于二分团问题)。然而，第 13 章讨论的下界论证方法(比如，计算 $M(f)$的秩或者特征值)都具有多项式复杂度，因此这些方法都满足"可构造性"条件。通信复杂度的差异论证法不是多项式时间的，但差异可以在多

项式时间内被近似到 $O(1)$ 因子范围内(参见第 13 章的注记)，因此差异论证法也满足可构造性条件。

而且，这些下界论证方法中所使用的条件(比如，第二大的特征值较小，具有较高的秩，或者具有较小的差异等等)都可以被随机矩阵以较高的概率满足。因此，这些下界论证法也都满足广泛性条件。◀

可见，许多下界确实使用了自然证明。事实上，可以证明人们目前证得的所有"组合"线路下界都是自然的，包括第 12～16 章直接证明的关于线路的结构或通信协议的结构的所有下界。因此，人们不禁要问，这些下界可以用更一般的方法来证得吗？是什么固有的原因使得下界必须满足可构造性条件和广泛性条件呢？ 500

23.2.1　为什么要求可构造性

注意，数学上的"非构造性"证明往往通过对某些无穷集合展开讨论来阐明某个对象的存在性，而无需给出构造该对象的明确算法。这种"非构造性"证明也曾经饱受争议。但如今，多数数学家早已完全习惯于非构造性证明。

在自然证明中，我们要求的"可构造性"是一种非常强的形式。因为它不仅要求证明能够被一个有穷算法构造出来，而且还要求构造算法具有多项式时间复杂度。这使得许多证明虽然在数学上是可构造的，但在我们的定义下却不是可构造的。但出人意料的是，即使在我们更严格的定义下，组合数学上的许多证明仍然是可构造的，而且人们目前证得的所有线路下界也都是可构造的。

事实上，线路下界通常只依赖于组合数学中的证明技术，并且一般还只依赖于在我们的定义下是可构造的那些组合学证明技术。在几个情况中，虽然人们最初用组合学得出的结论不是"可构造的"，但后来却又为这些结论找到了构造性证明。最著名的例子是洛瓦兹局部引理(Lovász Local Lemma)，它在 1975 年被发现[EL75]，但它的算法形式直到 1991 年才被发现[Bec91]。有几个线路下界也是这样。对于 14.2 节中介绍的 $\mathbf{ACC}^0[q]$ 的下界，莱兹波偌夫(Razborov)和卢吉奇(Rudich)后来发现了它的"自然证明"。巴拜(Babai)等人最初在 1992 年[BNS89]给出了多方通信协议下界，它的原始形式不是构造性的，后来莱斯(Raz)[Raz00]给出了该下界的"自然证明"(参见 13.3 节)。

虽然非构造性证明技术也存在于组合数学中(比如概率方法，零点定理，拓扑论证法等等)，但人们还未能利用这些证明技术为明确的函数找出更佳的线路下界。对这些专题的进一步思考，请参阅 23.5 节。

23.2.2　为什么要求广泛性

为什么我们在证明具体函数(如奇偶性函数和 3SAT)的线路下界时所采用的性质也需要对随机函数以较高的概率成立呢？下面，我们尝试形式化地解释其中的原因。大致上讲，每当我们要证明一个具体的函数 $f_n:\{0,1\}^n\to\{0,1\}$ 不存在规模为 S 的线路时，实际上需要证明从 $\{0,1\}^n$ 到 $\{0,1\}$ 的所有函数中至少有一半不存在规模为 $S/2-10$ 的线路。这是由于，如果我们随机地选择函数 $g:\{0,1\}^n\to\{0,1\}$，并且记 $f_n=(f_n\oplus g)\oplus g$(其中函数 $g\oplus h$ 将 x 映射为 $g(x)\oplus h(x)$)，则不难发现：如果 $f_n\oplus g$ 和 g 都有规模小于 $S/2-10$ 的线路，则 f_n 将存在规模小于 S 的线路。由于 g 和 $f_n\oplus g$ 都是服从均匀分布的随机函数，故 f_n 的性路复杂性下界 S 将使得复杂性下界 $S/2-10$ 对半数的随机函数成立。

23.2.3　用复杂性测度看自然证明

事实证明，证明下界的大量更一般的技术所得到的性质也都同时满足可构造性和广泛
501 性(也就是说，这些性质是自然的)。为了使讨论更具体一些，我们将讨论的焦点集中在布尔公式上(参见图 23-1)，而且还要求在所讨论的布尔公式对应的布尔线路中所有逻辑门的入度都等于 2 而出度都等于 1。我们希望恰当地应用归纳法能够证明这种布尔公式的一个下界。假设我们有一个"复杂"函数，它需要用一个大规模的布尔公式线路才能被计算。由于布尔线路的输出是一个"复杂"函数，故输出门的两条入边对应的函数中至少有一个"比较复杂"(因为这两个函数通过单个逻辑门组合在一起得到了一个"复杂"函数)。下面，我们将上述直观认识形式化，并指出为什么我们能最终证明随机函数的公式复杂度下界。

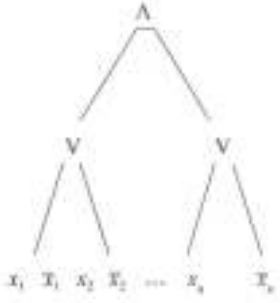


图 23-1　一个布尔公式

刻画"复杂性"的一种最显然的方法是用一个函数 μ 将 $\{0,1\}^n$ 上的每个布尔函数映射为一个非负整数。我们称 μ 是一个形式复杂度测度(Formal Complexity Measure)，如果它满足如下两条性质。其一，μ 在平凡函数上的取值很小，亦即，$\mu(x_i)\leqslant 1$ 且 $\mu(\bar{x}_i)\leqslant 1$ 对所有 i 都成立。其二，我们还要求

- $\mu(f\wedge g)\leqslant\mu(f)+\mu(g)$对任意 f，g 成立；
- $\mu(f\vee g)\leqslant\mu(f)+\mu(g)$对任意 f，g 成立。

例如，如下的函数 ρ 是一个平凡的形式复杂度测度。

$$\rho(f)=1+f\text{ 的最小公式的大小} \tag{23.2}$$

事实上，用数学归纳法容易证明如下的定理。

定理 23.5　如果 μ 是一个形式复杂度测度，则 $\mu(f)$是函数 f 的公式复杂度的下界。

这样，为了形式化地讨论前面勾勒出的归纳过程，只需定义一个形式复杂度测度 μ 使得 $\mu(3SAT)$是超多项式的。例如，可以定义 $\mu(f)$是"使得 f 与 3SAT 具有相同函数值的输入占所有输入的比例"，当然也可以采用这个形式复杂度测度的其他变形。一般而言，不难想象，只有在深入观察 3SAT 函数的基础上，才能定义出一个形式复杂度测度使得我们能够证明 3SAT 函数的好的下界。但是，下面的定理却表明，无论我们如何定义 3SAT 函数的形式复杂度测度，我们得到的下界也必然会对随机函数成立。

引理 23.6　假设 μ 是一个形式复杂度测度，并且存在函数 $f:\{0,1\}^n\to\{0,1\}$使得 $\mu(f)\geqslant S$对某个较大的数 S 成立。那么，在所有函数 $g:\{0,1\}^n\to\{0,1\}$中至少有 1/4 比
502 例的函数 g 满足 $\mu(g)\geqslant S/4$。

证明　证明过程是对前面的直观观察结果的形式化。对于随机函数 $g:\{0,1\}^n\to\{0,1\}$，我们令 $f=h\oplus g$，其中 $h=f\oplus g$。因此，$f=(\bar{h}\wedge g)\vee(h\wedge\bar{g})$，进而 $\mu(f)\leqslant\mu(g)+\mu(\bar{g})+\mu(h)+\mu(\bar{h})$。如果所有函数中有超过 3/4 比例的函数在给定的形式复杂度测度上都小于 $S/4$，则由合并界限可知"四个函数 g，$\bar{g}$，h，$\bar{h}$ 在给定的形式复杂度测度上都小于 $S/4$ 的概率"将大于 0。由此可得，$\mu(f)\leqslant S$。而这与引理的假设条件矛盾。 ■

事实上，下面更强的定理也成立。

定理 23.7　如果 $\mu(f)>S$，则对于任意 $\varepsilon>0$，在所有函数中至少有 $1-\varepsilon$ 比例的函数 g 使得下式成立

$$\mu(g) \geqslant \Omega\left(\frac{S}{(n+\log(1/\varepsilon))^2}\right)$$

证明定理 23.7 所采用的思想仍然是将 f 改写为少数几个函数的组合，这类似于前面证明引理的思想。引理 23.6 和定理 23.7 表明，利用 $2^{O(n)}$ 时间可计算的形式复杂度测度 μ 能够得出的所有下界都是自然的。

23.3　定理 23.1 的证明

现在，我们证明定理 23.1。证明过程要用到 9.5.1 节给出的一个关键事实，亦即由一个伪随机数产生器可以构造出一个伪随机函数族。记住，在伪随机函数族 $\{f_s\}_{s\in\{0,1\}^*}$ 中当 $s\in_R\{0,1\}^n$ 时，f_s 是从 $\{0,1\}^n$ 到 $\{0,1\}$ 的函数。伪随机函数族 $\{f_s\}_{s\in\{0,1\}^*}$ 具有如下的两个性质：(a)存在一个多项式时间算法，它在输入 s，x 上输出 $f_s(x)$；(b)任何多项式算法都无法以不可忽略的概率区分"通过神喻获得的函数值 $f_s(\cdot)$"(其中 $s\in_R\{0,1\}^n$ 是随机选取的)和"通过神喻获得的从 $\{0,1\}^n$ 到 $\{0,1\}$ 的随机函数的函数值"。

由于伪随机数产生器可以由单向函数构造得到[HILL99]，因此我们也可以假设伪随机函数族是由单向函数构造得到的。事实上，可以验证：从一个单向函数(它的逆函数不能被时间复杂度为 2^{n^ε} 的任意算法求得，其中 ε 是一个常数)出发，用归约技术可以得到一个伪随机函数族 $\{f_s\}_{s\in\{0,1\}^*}$，使得时间复杂度为 2^{n^ε} 的任意算法(其中 ε' 是一个常数)都无法区分 $f_s(\cdot)$(其中 $s\in_R\{0,1\}^n$)和从 $\{0,1\}^n$ 到 $\{0,1\}$ 的随机函数。

伪随机函数族与自然证明之间有什么必然的联系呢？假设$\mathcal{P}$是所有 n^c-有用的 n 位函数上的一个自然性质。该性质可视为一个算法(虽然它的时间复杂度是 $2^{O(n)}$)，该算法满足：(a)在线路复杂度低于 n^c 的函数上，算法输出 0；(b)在另一些函数上输出 1，且这些函数在所有函数中的比例是不可忽视的。因此，这样的算法有望以不可忽视的概率区分伪随机函数和真随机函数。这就是我们下面要证明的结论。

假设 $\{f_s\}$ 是前面提到的伪随机函数族，并且时间复杂度为 2^{n^ε} 的任意算法都无法区分 $f_s(\cdot)$和从 $\{0,1\}^n$ 到 $\{0,1\}$ 的随机函数。下面，我们利用自然性质$\mathcal{P}$来设计一个算法(下面称为区分器)，使得该算法能够以不可忽略的概率区分从 $\{0,1\}^n$ 到 $\{0,1\}$ 的随机函数和 $f_s(\cdot)$。注意，我们假设这两个函数的函数值都可以用神喻来计算。

503

给定计算未知函数 h 的函数值的神喻(其中的未知函数可能是某个 s 对应的函数 f_s，也可能是随机函数)，区分器取 $n=m^{1/c}$ 并计算函数 $g(x)=h(x0^{m-n})$ 的真值表。注意，区分器构造 g 的真值表的时间复杂度是 $2^{O(n)}$。然后，区分器在 g 上调用算法$\mathcal{P}$，并将$\mathcal{P}$的输出结果作为最终的计算结果输出。下面，我们考虑区分器在随机函数和 f_s 上的运行。第一种情况，假设 h 是随机函数。此时，g 是定义域为 $\{0,1\}^n$ 的随机函数。因此，$\mathcal{P}$输出 1 的概率至少为 $1/n$。第二种情况，h 是某个 s 对应的伪随机函数 f_s。于是，函数 g 的线路复杂度至多为 n^c。这是由于，映射 s，$x\mapsto f_s(x)$ 可以在 $\mathrm{poly}(m)$ 时间内被计算，进而映射 $x\mapsto g(x)$ 可以用一个规模为 $\mathrm{poly}(m)=n^c$ 的线路来计算(并且该线路中的 s 个输入位的取值已经固定)。(当然，区分器并不知道 s，也不知道这个线路，故我们只是断言这样的线路必然存在。)因此，在给定 g 的真值表之后，算法$\mathcal{P}$必然输出 0。

因此，我们构造的区分器至少以 $1/n$ 的概率区分了函数 f_s 和随机函数，并且区分器的时间复杂度是 $2^{O(n)}$，该复杂度低于 2^{n^ε}。从逆否命题的角度看，这就表明：如果伪随机函数是亚指数强的，则该函数不存在自然性质。

23.4　一个"不自然的"下界

不自然的证明能得出线路的下界吗？作为一个例子，这里用简单而陈旧的对角线方法来证明线路的一个下界！当然，证明过程还会用到其他技术。在给出结果之后，我们再解释这种方法为什么不是自然的。

讨论这一结果时，我们要用到*诺言问题*(Promise Problem)的概念。诺言问题指的是从$\{0,1\}^*$到$\{0,1\}$的一个部分定义的布尔函数。也就是说，诺言问题是一个函数$f:\{0,1\}^*\to\{0,1,\perp\}$，其中$\perp$表示函数值未定义。算法$A$求解了诺言问题$f$，指的是$A(x)=f(x)$在$f(x)\in\{0,1\}$时恒成立。注意，这并未要求算法$A$在$f(x)=\perp$的情况下输出什么值。我们可以将任何一个复杂性类的定义推广到诺言问题上。特别地，将8.2节中定义的复杂性类**MA**推广到诺言问题上之后，我们把所得的复杂性类记为**PromiseMA**。也就是说，诺言问题f属于**PromiseMA**指的是，存在一个概率型多项式时间算法A和多项式$p(\cdot)$使得，对任意$x\in\{0,1\}^*$均有：(a)如果$f(x)=1$，则存在$y\in\{0,1\}^{p(|x|)}$使得$\Pr[A(x,y)=1]\geqslant 2/3$；并且(b)如果$f(x)=0$，则对任意$y\in\{0,1\}^{p(|x|)}$都有$\Pr[A(x,y)=1]\leqslant 1/3$。关于**PromiseMA**，我们有下面的下界。

定理 23.8　([San07])　**PromiseMA**$\not\subseteq$**SIZE**(n^c)对任意$c\in\mathbf{N}$均成立，其中**SIZE**(n^c)表示由可以用n^c规模线路求解的所有诺言问题构成的集合。

证明　回顾一下，在第8.3节讨论的**PSPACE**的交互式证明中，证明者算法本身可以用多项式空间实现。这意味着，如果L是一个**PSPACE**-完全问题，则存在L的一个交互式证明使得，证明者在证明x属于L时仅花费多项式时间并且仅通过神喻访问语言L本身。事实上，人们已经证明，存在语言L_0使得：在长度为n的输入上，证明者(向神喻提出的)查询的长度至多为n[TV02]。这意味着，如果L_0可以用规模为$S(n)$的线路来判定，则在交互式证明系统中证明者可以将这个线路发送给概率型验证者。收到的线路后，概率型验证者可以自己独立地执行交互式协议。因此，如果$L_0\in$**SIZE**$(S(n))$，则存在L_0的时间$\mathrm{poly}(S(n))$的**MA**协议。(类似的推理过程在定理8.22和引理20.18中出现过。)

504

定义$S(n)$等于1加上在长度为n的输入上判定L_0的最小线路的规模。于是，如果$S(n)\leqslant\mathrm{poly}(n)$，则意味着**PSPACE**$\subseteq$**MA**。但此时，对任意的$c$而言，**MA**中显然存在不属于**SIZE**(n^c)的语言(参见第6章习题6.5)。事实上，即使我们假设在某个常数c上$S(n)\leqslant n^c$对无穷个n成立，则上述的推理过程仍成立。因此，我们假设$S(n)=n^{c(1)}$。注意，L_0存在一个时间复杂度为$\mathrm{poly}(S(n))$的**MA**协议，但却不存在规模为$S(n)$的线路。于是，只要$S(n)$是时间可构造的，则将上面的区分过程作用到新定义的语言$L_1=\{x01^{S(|x|)^{1/c}-|x|-1}:x\in L_0\}$上可知，$L_1$属于**MA**但却不属于**SIZE**$(n^c)$，由此得出引理。遗憾的是，我们不能假设$S(n)$是时间可构造的。因此，无法确保$L_1$属于**MA**。虽然如此，我们仍可以如下定义一个诺言问题f_1。f_1仅在形如$y=x01^{S(|x|)^{1/c}-|x|-1}$的输入上有定义，并且在这些输入上定义$f_1(y)=L_0(x)$。不难看出，$f_1\in$**PromiseMA**$\setminus$**SIZE**$(n^c)$。■

上面的证明是非自然的，这是由于该证明依赖于**PSPACE**$\not\subseteq$**SIZE**(n^c)的证明，而后者的证明却用到了对角线方法——对角线方法在本质上就是一个非自然的技术，因为它只关注一个非常具体的函数，继而肯定会违背广泛性条件。从另一个角度看，可以认为对角线方法是在证明"函数和每个小规模线路在某些输入上具有不一致表现"的某种性质——这种性质满足广泛性却又不满足可构造性。事实上，定理23.1表明，定理23.8不存在自然证

明，除非亚指数强的单向函数不存在。人们还证明了，定理 23.8 给出的下界不是相对性的[Aar06]。遗憾的是，很难将对角线方法或基于算术化的方法"向下推移"以证明 **NP** 函数的下界。

23.5　哲学观点

我们认为，自然证明以及具有这种自然性的其他负面结论都很有价值。当我们求解难题陷入困境时，一种有益的做法是尝试去证明：该难题无法求解或者无法用某种特定的方法求解。这样做可能会得到问题的某些深刻的性质。但如果不采用这种方法，可能永远也无法得到问题的这些性质。理解了求解问题时遇到的各种困难，我们才知道在求解问题时应该如何处理或者如何绕开这些困难。事实证明，这种研究方法非常有用。在复杂性理论中，或者在理论计算机科学这一更大的领域中，应用这种方法成功解决问题的例子不胜枚举。就下界的证明而言，自然证明范式表明，包含似真伪随机函数产生器的任何复杂性类都会给现有的下界证明方法造成困难。由于像 $\mathbf{NC}^1$ 和 $\mathbf{TC}^0$ 这样简单的复杂性类都包含似真的伪随机函数，因此，我们就不难理解，为什么证明下界这项工作会在复杂性类 $\mathbf{ACC}^0$ 上陷入停顿。

另一方面，由于自然证明成功地囊括了所有现存的下界证明方法，因此它在一定程度上也阻碍了研究者对线路下界的深入思考。其实，研究者们大可不必如此沮丧。在组合数学中，有些证明技术既不满足可构造性，也不满足广泛性。在我自己看来，可构造性很容易绕过去，我们在前一小节给出的非自然证明中看到了这一点。从组合数学这个更广阔的领域来看，一个相关的例子是克内泽尔图(Kneser Graph)色数的洛瓦兹下界[Lov78]。一般情况下，计算图的色数下界是 **coNP**-完全问题。洛瓦兹(Lovász)用拓扑证明方法(用到了著名的博苏克-乌拉姆不动点定理(Borsuk-Ulam Fixed-Point Theorem))准确地计算了克内泽尔图的色数。根据他给出的证明，人们得到了一个能在所有图上计算色数的算法[MZ04]——但该算法在一般图上运行于 **PSPACE** 内！因此，假如将这一算法视为线路下界的话，则以我们的观点看，该下界应该是"非构造性的"。但是，由于克内泽尔图的高度对称性，洛瓦兹给出的证明过程并不太复杂。这就告诉我们，不要盲目地相信"非构造性=难"。我们应该铭记相对技术的局限性结果给我们的教训(参见 3.4 节)。我们在第 8 章和第 11 章中已经看到，算术化技术——一种非相对技术——已经帮助我们证明了一大批结论，而这些结论用相对技术是无法证明的。某种"非自然的"技术很可能就是一把开启线路下界证明之门的钥匙，获得这把钥匙可能会使各种线路下界像洪水般冲破牢笼。

505

本章注记和历史

线路下界无意间最终与随机函数发生了关联，这一观察最早出现在莱兹波诺夫(Razborov)给出的关于近似方法的局限性的结果中[Razb89]。本章并未全面介绍莱兹波诺夫和卢吉奇在论文[RR94]中给出的思想。这篇论文观察到，即使在介于 $\mathbf{ACC}^0$ 和 $\mathbf{NC}^1$ 之间的复杂性类 $\mathbf{TC}^0$ 中，也存在伪随机函数产生器。这一结论表明，自然证明甚至可能无法区分 $\mathbf{TC}^0$ 和 **P**。莱兹波诺夫在亚模复杂性测度上观察到的结果(参见习题 23.4)很重要，因为现有的许多证明公式复杂性的方法都使用了亚模复杂性测度。由此可知，自然证明在证明亚模复杂性下界时也无能为力。23.4 节给出的下界源自圣哈南(Santhanam)[San07]，而类似的证明技术最早曾在证明"具有小规模建言的概率算法的分层定理"时使用过[Bar02，

FS04，GST04]。

与我们的相对乐观的观点相比，莱兹波偌夫自己曾在[Razb03]的绪言中认为自然证明给线路下界造成的困难是非常严重的。他注意到现有的下界方法都使用了受囿算术之类的弱算术理论。他曾经猜想，在这种逻辑系统下得出的任何线路下界都是自然的(因而，这种逻辑系统不大可能用来证明线路下界)。但是，即使在离散数学中，有些定理的证明过程也无需借助受囿算术就可以完成推理，比如博苏克-乌拉姆不动点定理。这种现象正是我们持乐观观点的原因。但是，也有部分其他的研究者比莱兹波偌夫还悲观，他们怀疑 **P** 与 **NP** 问题可能独立于数学。比如，有人认为 **P** 与 **NP** 独立于策梅洛-弗兰克尔集合论(Zermelo-Fraenkel Set Theory)。要了解这一问题的更多内容，请参阅阿伦森(Aaronson)的综述[Aar03]。

最近，阿伦森和维格德尔森(Wigderson)[AW08]证明了复杂性的一种称为代数化的
506 新障碍。对于复杂性类$C \subseteq D$，如果存在神喻 O 使得$C^{\tilde{O}} \subseteq D^{O}$(其中$\tilde{O}$表示布尔函数 O 在更大的域或环(比如整数环)上的低次扩张)，则C和D无法用"代数化技术"进行区分。粗略地讲，代数化技术能够刻画目前用算术化方法证得的所有结果(如 **IP**=**PSPACE** 和 **PCP** 定理等)。特别地，23.4 节证明的下界用到了代数化技术。但是[AW08]证明了，代数化方法甚至也无法证得 **NP** 语言的任何超线性的下界。

习题

23.1　证明定理 23.7。

23.2　证明：随机函数 $g:\{0,1\}^* \to \{0,1\}$ 将高概率地满足$\mathcal{P}(g)=1$(其中$\mathcal{P}$是例 23.3 中定义的性质)，但是固定 g 的 $n-n^{\varepsilon}$ 个输入位却无法使 g 取常数值。

23.3　离散对数问题 DISCRETELOG 要求在给定的素数 p，和 g，$y \in \mathbf{Z}_p^*$(其中 $g \neq 1$)上，找出 $x \in \mathbf{Z}_p^*$ 使得 $y=g^x \pmod p$。证明维格德尔森(Wigderson)的如下观察结果："求解离散对数问题需要规模为 $2^{n^{\varepsilon}}$ 的线路，其中 ε 是一个常数"这一个结论不存在自然证明。

23.4　(莱兹波偌夫(Razborov)[Razb92])一个亚模(submodular)复杂性测度是使得"$\mu(f \vee g)+\mu(f \wedge g) \leqslant \mu(f)+\mu(g)$对任意函数 f，g 成立"的复杂性测度 μ。证明：对任意的 n 位函数 f_n，亚模复杂性测度 μ 满足 $\mu(f_n)=O(n)$。

23.5　令 L 是由如下的所有三元组$\langle \varphi, P, i \rangle$构成的语言。三元组$\langle \varphi, P, i \rangle$中，$\varphi \pmod P$的第 i 个位等于 1，其中 P 是一个整数而 φ 是一个用到常量的表达式，φ 中的算术运算操作 $+$，$-$，$\cdot$ 和形如 $\sum_{x_j \in \{0,1\}}$ 或 $\prod_{x_j \in \{0,1\}}$ 的求和、求积运算满足下列性质：如果 φ 中出现的所有变量按出现的先后顺序依次为 x_1，…，x_n，则每个变量 x_i 在 φ 中的最后一次出现之前至多有一个 Π 运算符用到变量 x_j($j>i$)。证明：L 是 **PSPACE** 完全的，进而存在 L 的一个交互式证明使得：(1)证明者算法借助神喻来判定 L 的成员资格，它的运行时间是多项式时间；(2)证明者在证明 $x \in \{0,1\}^*$ 属
507 于 L 时，他向神喻提出的查询的长度至多为 n。

数学基础

本附录介绍了本书用到的数学基础。然而，多数的数学基础仅在少数地方用到。因此，读者可能仅需快速浏览 A.1、A.2 和 A.3 节，在需要时再回头阅读其他节。特别地，本书第一部分本质上仅需数学证明和离散数学中非常基础的概念，此外还需要一些概率知识。

本附录涉及的主题在很多教科书和网站上有更深入的讨论。事实上，几乎所有需要的数学基础均在"计算机科学离散数学"这门本科生课程中教授过，很多计算机系目前均设置了这门课程。涵盖这些数学基础的其他较好的资源包括帕帕迪米特里奥(Papadimitriou)和瓦兹拉尼(Vazirani)的讲义[PV06]，以及罗森(Rosen)的书[Ros06]。

最常用的数学工具是离散概率，这一领域较好的著作是阿龙(Alon)和斯宾塞(Spencer)的书[AS00b]。密特森迈克尔(Mitzenmacher)和尤普夫(Upfal)的书[MU05]，以及牟特瓦尼(Motwani)和拉加万(Raghavan)的书[MR95]均从算法角度阐述了概率论。

尽管算法知识对本书而言不是必需的，但却十分有益。下面这些书有助于读者了解算法的相关知识，最近出版的优秀书籍包括戴斯古普塔(Dasgupta)等人的书[DPV06]，以及克莱恩伯格(Kleinberg)和塔多斯(Tardos)的书[KT06]，稍早出版的优秀书籍是科曼(Cormen)等人的教材[CLRS01]。本书不需要可计算性和自动机理论等预备知识，然而熟悉这些理论也将十分有益，西普赛尔(Sipser)的书[SIP96]精辟地介绍了这些知识。舒普(Shoup)的书[Sho05]从计算机科学的角度讲述了代数和数论知识。

本书所需的数学基础就是能够比较舒适地阅读数学证明。数学证明必须是绝对令人信服的，但这并不意味着数学证明必然是过度形式化的和冗长乏味的。数学证明仅仅需要书写清楚，且不存在逻辑间隙。当你书写证明时，请力求清晰、简洁，而不要过多地使用形式化记号。当然，要相信一个结论为真，需要明确该结论的含义。这就是为什么数学(和本书)特别强调每个定义的精确性的原因。当你遇到任何定义时，确保你完全理解了该定义，这有时需要借助一些简单的例子。通常，理解了数学结论就完成了证明该结论一半以上的工作量。

A.1 集合、函数、序对、字符串、图、逻辑

集合。一个集合含有有穷或无穷个元素，其中元素不重复且无序。例如，{2，17，5}，$\mathbf{N}=\{1, 2, 3, \cdots\}$(自然数构成的集合)，$[n]=\{1, 2, \cdots, n\}$(从 1 到 n 的自然数构成的集合)，$\mathbf{R}$(实数构成的集合)。对于有穷集合 A，用 $|A|$ 表示 A 中元素的个数。集合上的一些操作包括：(1)并：$A \cup B=\{x: x \in A$ 或 $x \in B\}$；(2)交：$A \cap B=\{x: x \in A$ 且 $x \in B\}$；(3)差：$A \setminus B=\{x: x \in A$ 且 $x \notin B\}$。

函数。若 f 将集合 A 中的每个元素映射为集合 B 中的一个元素，则称 f 是从 A 到 B 的一个函数，记为 $f: A \to B$。如果 A 和 B 是有穷集合，则从 A 到 B 的函数有 $|B|^{|A|}$ 个。如果对任意 $x, w \in A$，只要 $x \neq w$ 就有 $f(x) \neq f(w)$，则称函数 f 是一对一的。如果 A 和 B 是有穷集合，则存在这种一对一的函数将蕴含 $|A| \leqslant |B|$。如果对每个 $y \in B$ 均存在 $x \in A$ 使得 $f(x)=y$，则称函数 f 是满的。如果 A 和 B 是有穷集合，则存在这种映上的

函数蕴含$|A|\geqslant|B|$。如果一个函数既是一对一的又是满的，则称之为一个置换。如果A和B是有穷集合，则存在从A到B的置换蕴含$|A|=|B|$。

序对和元组。如果A，B是两个集合，则$A\times B$表示所有满足$a\in A$，$b\in B$的有序对$\langle a,b\rangle$构成的集合。注意，如果A，B是有穷集合，则$|A\times B|=|A|\cdot|B|$。类似地，$A\times B\times C$定义为所有满足$a\in A$，$b\in B$，$c\in C$的有序对(a,b,c)构成的集合。对于$n\in\mathbf{N}$，用A^n表示集合$A\times A\times\cdots\times A$($n$个$A$)。我们常用$\{0,1\}^n$表示长度为$n$的二进制位的序列，而$\{0,1\}^*=\bigcup_{n\geqslant 0}\{0,1\}^n$($\{0,1\}^0$仅含一个元素，即长度为0的二进制位序列，我们称之为空字并记为ε)。正如0.1节那样，我们用二进制位串表示各种对象(如数、图、矩阵等)，对象x表示为$\langle x\rangle$(不要与下取整函数$\lfloor x\rfloor$混淆)。甚至，有时去掉符号$\langle\cdot\rangle$而直接用x表示对象及其表示。

图。一个图由一个顶点集V(通常假设为集合$[n]=\{1,\cdots,n\}$，n是正整数)和一个边集E构成，其中边集由V上的一些无序对(即大小为2的子集)构成。我们用$\overline{uv}$表示边$\{u,v\}$。对于$v\in V$，v的邻接点指的是所有满足$\overline{uv}$的顶点$u\in V$。在有向图中，边由有序顶点对构成。有时，为了强调边的有向性，用$\overrightarrow{uv}$表示有向图的边$\langle u,v\rangle$。n顶点图G可以表示为$n\times n$邻接矩阵A，其中如果边$\overrightarrow{ij}$出现在G中则A_{ij}等于1，否则A_{ij}等于0。无向图可以按下述方式视为有向图G：对于任意顶点u和v，G包含边$\overrightarrow{uv}$当且仅当G包含边$\overrightarrow{vu}$。因此，无向图可以表示为一个对称邻接矩阵A($A_{ij}=A_{ji}$对任意i，$j\in[n]$成立)。

布尔运算符。一个布尔变量是取值为非真即假的变量(有时，真用1表示而假用0表示)。布尔变量可以用逻辑运算符与($\wedge$)、或($\vee$)、非($\neg$，有时也用上划线表示)组合产生布尔表达式。例如，$(u_1\wedge\bar{u}_2)\vee\neg(u_3\bar{u}_1)$是布尔变量$u_1$，$u_2$，$u_3$上的一个布尔表达式。布尔运算符的定义正如人们常用的那样：$a\wedge b=$真，如果$a=$真且$b=$真，否则$a\wedge b=$假。$\bar{a}=\neg a=$真，如果$a=$假，否则$\bar{a}=\neg a=$假。$(a\vee b)=\neg(\bar{a}\wedge\bar{b})$。有时，我们还使
509 用其他逻辑运算符，例如异或($\oplus$)，但这些逻辑运算符总可以等价地替换为仅使用$\wedge$，$\vee$和$\neg$的逻辑表达式，例如$a\oplus b=(a\wedge\bar{b})\vee(\bar{a}\wedge b)$。对于定义在$n$个变量$u_1$，$u_2$，…，$u_n$上的一个逻辑公示$\varphi$和变量的每个赋值$u\in$(假，真)^n(或者等价地$\{0,1\}^n$)，用$\varphi(u)$表示$\varphi$中各个变量依次取$u$中的各个值时$\varphi$的值。如果存在$u$使得$\varphi(u)=$真，则称$\varphi$是可满足的。

量词。我们还经常使用量词$\forall$(对所有)和$\exists$(存在)。亦即，如果φ是依赖于变量x的值而取值为真或假的一个条件，则$\forall_x\varphi(x)$表示"在x的每种取值上φ均取值为真"这一个论断。如果A是一个集合，则用$\forall_{x\in A}\varphi(x)$表示"对于$x$取自集合$A$中的每个值，$\varphi$均取值为真"这一论断。存在量词$\exists$的定义与此类似。形式上，$\exists_x\varphi(x)$成立当且仅当$\neg(\forall_x\neg\varphi(x))$成立。

大O记号。我们经常使用0.3节中定义的大O记号(即O，Ω，Θ，o，ω)。

A.2　概率论

一个有穷离散概率空间指的是一个有穷集合$\Omega=\{\omega_1,\cdots,\omega_N\}$和一些满足$\sum_{i=1}^{N}p_i=1$的数值$p_1$，…，$p_N\in[0,1]$。一个随机元素是取自该空间的一个元素，其中$\omega_i$被取中的概率等于$p_i$。如果$x$取自样本空间$\Omega$，则表示为$x\in_R\Omega$。如果未说明分布类型，则假设$\Omega$

上的元素服从均匀分布$\left(\text{即 } p_i=\frac{1}{N}\text{对每个 } i \text{ 成立}\right)$。

空间 Ω 上的一个事件指的是子集 $A\subseteq\Omega$。事件 A 发生的概率，记为 $\Pr[A]$，等于 $\sum_{\omega_i\in A}p_i$。例如，考虑将一枚匀质的硬币投掷 n 次后产生的结果，则概率空间 Ω 为所有 2^n 种可能的结果(即 $\Omega=\{0,1\}^n$)，其中 $p_i=2^{-n}$ 对每个 $i\in[2^n]$ 成立。所有头面朝上(或者等价地说 1)的总次数为偶数的样本构成事件 A。此时，$\Pr[A]=1/2$(留作练习)。本书经常使用下面的简单界限，亦即合并界限。对于任意事件 $A_1, A_2, \cdots, A_n$，有

$$\Pr[\bigcup_{i=1}^{n}A_i]\leqslant\sum_{i=1}^{n}\Pr[A_i] \tag{A.1}$$

容斥原理

合并界限是一个更具一般性的原理的特例。事实上，容易注意到，如果集合 $A_1, \cdots, A_n$ 是相交的，则 $\bigcup_i A_i$ 的概率小于 $\sum_i p_i$，因为出现在两个集合中的元素的概率被重复累加。为避免重复累加，可以从总和中减去 $\sum_{i<j}\Pr[A_i\cap A_j]$。但这又可能漏算了一些元素的概率，因为我们可能减掉了出现在至少 3 个集合中的元素的概率。依次类推，我们得到下面的论断。

论断 A.1 (容斥原理) 对于任意 $A_1, \cdots, A_n$，

$$\Pr[\bigcup_{i=1}^{n}A_i]=\sum_{i=1}^{n}\Pr[A_i]-\sum_{1\leqslant i<j\leqslant n}\Pr[A_i\cap A_j]+\cdots+(-1)^{n-1}\Pr[A_1\cap\cdots\cap A_n]$$

而且，上述表达式是一个交错和，亦即如果仅取右端的前 k 个求和项，则当 k 是奇数时所得结果是左端的上界，当 k 是偶数时所得结果是左端的下界。

有时，我们会用到上述论断的如下推论。该推论也称为 Bonefforni 不等式。

推论 A.2 对于任意事件 $A_1, \cdots, A_n$，

$$\Pr[\bigcup_{i=1}^{n}A_i]\geqslant\sum_{i=1}^{n}\Pr[A_i]-\sum_{1\leqslant i<j\leqslant n}\Pr[A_i\cap A_j]$$

A.2.1 随机变量及其期望

随机变量是从概率空间到实数集 $\mathbf{R}$ 的映射。例如，如果 Ω 如前所述(即投掷 n 次硬币的所有可能的结果)，则头面朝上的次数可以表示为随机变量 X。

随机变量 X 的数学期望，表示为 $E[X]$，是它的加权平均值，亦即 $E[X]=\sum_{i=1}^{n}p_iX(\omega_i)$。由定义即得下面简单的论断。

论断 A.3 (期望的线性性质) 对于空间 Ω 上的随机变量 X，Y，用 $X+Y$ 表示将 ω 映射为 $X(\omega)+Y(\omega)$ 的随机变量，则

$$E[X+Y]=E[X]+E[Y]$$

上述论断意味着，前例中的随机变量 X 的期望为 $n/2$。事实上，$X=\sum_{i=1}^{n}X_i$，其中 X_i 取值为 1，如果第 i 次投掷的硬币头面朝上；否则，X_i 取值为 0。显然，$E[X_i]=1/2$ 对任意 i 成立。

对于实数 a 和随机变量 X，定义 aX 为将 ω 映射为 $a\cdot X(\omega)$ 的随机变量。注

意，$E[aX]=aE[X]$。

例 A.4 设从$[n]$中随机独立地选取 k 个数 x_1，…，x_k，如果将满足 $x_i=x_j$ 的无序对 $\{i, j\}$ 称为一次冲突，那么冲突的总次数的期望是多少？对于任意 $i\neq j$，定义随机变量 $Y_{i,j}$ 使得 $Y_{i,j}$ 等于 1，如果 $x_i=x_j$；否则，$Y_{i,j}$ 等于 0。由于选取 x_i 时，$x_i=x_j$ 的概率等于 $1/n$，因此 $E[Y_{i,j}]=1/n$。由于冲突的总次数等于 $Y_{i,j}$ 的和，其中求和下标均取自$[k]$且 $i\neq j$。因此由期望的线性性质可知，冲突的总次数的期望为

$$\sum_{1\leqslant i<j\leqslant k} E[Y_{i,j}]=\binom{k}{2}\frac{1}{n}$$

这意味着，只要 $\binom{k}{2}\geqslant n$，亦即 k 略大于 $\sqrt{2n}$，则必然至少发生一次冲突。这就是熟知的生

511 日悖论，因为它解释了“为什么在约有 27 名学生的班级中极可能有两名学生的生日是同一天，尽管一年有 365 天？”这一奇怪的现象。

另一方面，如果 $k\ll\sqrt{n}$，则由合并界限可知，出现一次冲突的概率至多为 $\binom{k}{2}/n\ll 1$。

注记：(1)我们有时也考虑值域不是实数集 $\mathbf{R}$ 而是复数集 $\mathbf{C}$ 或集合$\{0, 1\}^n$的随机变量；(2)而且，我们将空间 Ω 上的随机变量 X 表示成满足 $\omega\in_R\Omega$ 的一个分布 $X(\omega)$。例如，$\Pr_{x\in_R X}[X^2=1]$和 $\Pr[X^2=1]$均可以用来表示 $\omega\in_R\Omega$，$X(\omega)^2=1$。 ◀

A.2.2　均值论证法

下面简单的事实非常有用。

均值论证法：如果 a_1，a_2…，a_n是平均值为 c 的数，则存在 $a_i\geqslant c$。

等价地，上述结论可以用概率论的术语重述为：

引理 A.5（概率法）　如果 X 是取值为一个有穷集合的随机变量且 $E[X]=\mu$，则事件“$X\geqslant\mu$”发生的概率不等于 0。

下面的两个事实也很容易被证明。

引理 A.6　如果 a_1，a_2…，a_n是平均值为 c 的数，则取值大于等于 kc 的数在所有数中的比例至多为 $1/k$。

引理 A.7（马尔可夫不等式）　任意非负随机变量 X 满足

$$\Pr(X\geqslant kE[X])\leqslant\frac{1}{k}$$

随机变量 X 的取值远小于其期望的概率存在上界吗？是的，如果 X 是有界的。

引理 A.8　如果 a_1，a_2…，a_n是区间$[0, 1]$上平均值为 ρ 的数，则大于等于 $\rho/2$ 的数的比例至少为 $\rho/2$。

证明　令 γ 表示满足 $a_i\geqslant\rho/2$ 的 i 的比例，则 a_i的平均值的一个上界为 $\gamma\cdot 1+(1-\gamma)\rho/2$。因此，$\rho\leqslant\gamma+\rho/2$，这意味着 $\gamma\geqslant\rho/2$。 ■

更一般地，我们有下面的结论。

引理 A.9　若 $X\in[0, 1]$且 $E[X]=\mu$，则对任意 $c<1$ 有

512

$$\Pr[X\leqslant c\mu]\leqslant\frac{1-\mu}{1-c\mu}$$

例 A.10 假设你参加了大量考试，每次考试的成绩均介于 1 到 100 之间。如果你的平均成绩是 90，则所有考试中你至少有一半以上的成绩在 80 分以上。 ◀

A.2.3 条件概率和独立

如果已知事件 B 已经发生，则概率空间由 Ω 演变为 $\Omega \cap B$。其中每个元素的概率需要乘以因子 $1/\Pr[B]$ 才能确保所有元素的概率之和等于 1。因此，在事件 B 发生的前提下，事件 A 发生的概率(条件概率)，记为 $\Pr[A|B]$，等于 $\Pr[A \cap B]/\Pr[B]$。这里，总假设 B 具有正概率。

如果事件 A 和 B 满足 $\Pr[A \cap B]=\Pr[A]\Pr[B]$，则称 A 和 B 独立。注意，这意味着 $\Pr[A|B]=\Pr[A]$ 和 $\Pr[B|A]=\Pr[B]$。对于事件 A_1，…，A_n，如果对于任意子集 $S \subseteq [n]$ 有

$$\Pr[\bigcap_{i \in S} A_i] = \prod_{i \in S} \Pr[A_i] \tag{A.2}$$

则称 A_1，…，A_n 相互独立。如果(A.2)式仅对满足 $|S| \leqslant k$ 的任意子集 $S \subseteq [n]$ 成立，则称 A_1，…，A_n 为 k-独立。

对于随机变量 X 和 Y 和任意实数 x，$y \in \mathbf{R}$，如果事件 $\{X=x\}$ 和事件 $\{Y=y\}$ 是独立的，则称 X 和 Y 独立。随机事件相互独立和 k-独立的概念也可以类似地推广到随机变量 X_1，…，X_n 上。下面的论断成立。

论断 A.11 如果 X_1，…，X_n 相互独立，则

$$E[X_1 \cdots X_n] = \prod_{i=1}^{n} E(X_i)$$

证明

$$\begin{aligned}
E[X_1 \cdots X_n] &= \sum_x x \Pr[X_1 \cdots X_n = x] \\
&= \sum_{x_1, \cdots, x_n} x_1 \cdots x_n \Pr[X_1 = x_1 \text{ 且 } X_2 = x_2 \text{ 且 } \cdots \text{ 且 } X_n = x_n] \\
&= \sum_{x_1, \cdots, x_n} x_1 \cdots x_n \Pr[X_1 = x_1] \Pr[X_2 = x_2] \cdots \Pr[X_n = x_n] \quad \text{(由独立性)} \\
&= \left(\sum_{x_1} x_1 \Pr[X_1 = x_1]\right)\left(\sum_{x_2} x_2 \Pr[X_2 = x_2]\right) \cdots \left(\sum_{x_n} x_n \Pr[X_n = x_n]\right) \\
&= \prod_{i=1}^{n} E(X_i)
\end{aligned}$$

其中，求和符号作用于随机变量或其乘积在空间 Ω 上能够取到的所有实数上。 ■ 513

A.2.4 标准差的上界

在不同的条件下，我们可以给出随机变量"远离"其期望的概率的各种更好的上界。这些上界均是恰当地使用马尔可夫不等式产生的。

随机变量 X 的方差定义为 $\mathrm{Var}[X]=E[(X-E(X))^2]$。注意，由于方差是一个非负随机变量的期望，因此 $\mathrm{Var}[X]$ 总是非负的。而且，由期望的线性性质，可以导出 $\mathrm{Var}[X]=E[X^2]-(E[X])^2$。随机变量 X 的标准差定义为 $\sqrt{\mathrm{Var}[X]}$。

我们给出的第一个界限是切比雪夫不等式，它在仅知道方差时特别有用。

引理 A.12 (切比雪夫不等式) 若 X 是一个标准差为 σ 的随机变量，则对任意

$k>0$ 有

$$\Pr[|X-E[X]|>k\sigma]\leqslant 1/k^2$$

证明　将马尔可夫不等式应用到随机变量 $(X-E[X])^2$ 上。注意，根据方差的定义可知，$E[(X-E[X])^2]=\sigma^2$。■

切比雪夫不等式在随机变量 X 等于两两独立的随机变量 X_1，…，X_n 之和 $\sum_{i=1}^{n}X_i$ 时非常有用。这是由于下面的论断，其证明留作练习。

论断 A.13　如果 X_1，…，X_n 是两两独立的随机变量，则

$$\mathrm{Var}\Big(\sum_{i=1}^{n}X_i\Big)=\sum_{i=1}^{n}\mathrm{Var}(X_i)$$

下面的不等式有许多叫法，理论计算机科学领域常称之为切尔诺夫界（参见注记 7.11）。切尔诺夫界考虑下述情形的各种情况。假设投掷一枚匀质硬币 n 次。头面朝上的次数的期望为 $n/2$，该随机变量在其期望值的周围是如何分布的？如果在 1000 次投掷中有 625 次头面朝上，你感觉吃惊吗？我们给出的结论更具一般性，因为该结论中考虑的是投掷 n 枚硬币，而每枚硬币可能具有不同的期望（硬币的期望即是头面朝上的概率，匀质硬币的期望等于 1/2）。这种重复的随机实验常称为泊松实验。

定理 A.14　（切尔诺夫界）　设 X_1，X_2，…，X_n 是 $[0,1]$ 上（即取值非 0 即 1）相互独立的随机变量，且 $\mu=\sum_{i=1}^{n}\mathrm{Var}(X_i)$，则对于任意 $\delta>0$ 有

$$\Pr\Big[\sum_{i=1}^{n}X_i\geqslant(1+\delta)\mu\Big]\leqslant\left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}\tag{A.3}$$

$$\Pr\Big[\sum_{i=1}^{n}X_i\leqslant(1-\delta)\mu\Big]\leqslant\left[\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right]^{\mu}\tag{A.4}$$

[514]　通常，我们仅使用下面的推论。

推论 A.15　在上述假设下，对于任意 $c>0$ 有

$$\Pr\Big[\Big|\sum_{i=1}^{n}X_i-\mu\Big|\geqslant c\mu\Big]\leqslant 2\cdot e^{-\min(c^2,c)\mu/(4\ln 2)}$$

特别地，上述概率以 $2^{-\Omega(\mu)}$ 为上界（其中高阶函数记号 Ω 中的常数依赖于 c）。

证明　出人意料的是，切尔诺夫界也是由马尔可夫不等式证得的。我们仅证明第一个不等式，第二个不等式可以类似地证明。我们引入一个取值为正的哑变量 t 并注意到

$$E[\exp(tX)]=E\Big[\exp\Big(t\sum_i X_i\Big)\Big]=E\Big[\prod_i\exp(tX_i)\Big]=\prod_i E[\exp(tX_i)]\tag{A.5}$$

其中，$\exp(x)$ 等于 e^x，而最后一个等号成立是因为所有随机变量 X_i 是相互独立的。而且，

$$E[\exp(tX_i)]=(1-p_i)+p_ie^t$$

由于 $1+x\leqslant e^x$，因此，

$$\begin{aligned}\prod_i E[\exp(tX_i)]&=\prod_i[1+p_i(e^t-1)]\leqslant\prod_i\exp(p_i(e^t-1))\\&=\exp\Big(\sum_i p_i(e^t-1)\Big)=\exp(\mu(e^t-1))\end{aligned}\tag{A.6}$$

最后，将马尔可夫不等式应用到随机变量 $\exp(tX)$ 上，并利用（A.5）式和（A.6）式以及 t 非负的事实，得到

$$\Pr[X \geqslant (1+\delta)\mu] = \Pr[\exp(tX) \geqslant \exp(t(1+\delta)\mu] \leqslant \frac{E[\exp(tX)]}{\exp(t(1+\delta)\mu)} = \frac{\exp[(e^t-1)\mu]}{\exp(t(1+\delta)\mu)}$$

由于 t 是哑变量，我们可以让 t 取任意的值。经过简单的计算可知，上式右端在 $t=\ln(1+\delta)$ 时达到最小值，这恰好是定理所述的结论。 ■

所以，投掷 n 枚匀质硬币(头面朝上的概率为 1/2)，则有 N 枚硬币头面朝上的概率至多为 $2e^{-\alpha^2/4}$，其中 $|N-n/2|>\alpha\sqrt{n}$。特别地，在投掷 1000 枚匀质硬币的过程中观察到至少 625 枚硬币头面朝上的概率小于 5.3×10^{-7}。

A.2.5 其他不等式

詹森不等式

下面的不等式十分有用，它是不等式 $E[X^2]\geqslant E[X]^2$ 的推广。

引理 A.16 (詹森不等式) 如果函数 $f: \mathbf{R}\to\mathbf{R}$ 对任意 $p\in[0,1]$ 和任意 $x, y\in\mathbf{R}$ 均有 $f(px+(1-p)y)\leqslant p\cdot f(x)+(1-p)\cdot f(y)$，则称 f 是凸函数。对于任意随机变量 X 和任意凸函数 f 恒有 $f(E(X))\leqslant E(f(X))$。

二项式系数的近似

二项式随机变量 B_n 具有特殊意义，它表示投掷 n 枚均质硬币时头面朝上的硬币数。对任意 k，$\Pr[B_n=k]=2^{-n}\binom{n}{k}$，其中 $\binom{n}{k}=\frac{n!}{k!\ (n-k)!}$ 表示[n]的大小为 k 的子集的个数。显然，$\binom{n}{k}\leqslant n^k$。然而，有时我们需要 $\binom{n}{k}$ 的更准确的估计值。这可以用下面的近似来得到。

论断 A.17 对于任意 n 和 $k<n$ 有 $\left(\frac{n}{k}\right)^k\leqslant\binom{n}{k}\leqslant\left(\frac{ne}{k}\right)^k$。

二项式系数的最佳近似可以由斯特林公式(Stirling Formula)公式给出。

引理 A.18 (斯特林公式) 对于任意 n，有

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}} < n! < \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$$

为证明斯特林公式，可以对三个部分分别取自然对数，并利用积分 $\int_1^n \ln x dx = n\ln n - n+1$ 来近似 $\ln n! = \ln(1\cdot 2\cdot\cdots\cdot n)=\sum_{i=1}^{n}\ln i$ 的值。由斯特林公式可得到如下的推论。

推论 A.19 对任意 $n\in\mathbf{N}$ 和 $\alpha\in[0,1]$ 有

$$\binom{n}{\alpha n} = (1\pm O(n^{-1}))\frac{1}{\sqrt{2\pi n\alpha(1-\alpha)}}2^{H(\alpha)n}$$

其中 $H(\alpha)=\alpha\log(1/\alpha)+(1-\alpha)\log(1/(1-\alpha))$，并且低阶函数记号 O 中的常数与 n 和 α 均无关。

更有用的估计

利用初等积分可以得到下列不等式：

- 对任意 $x\geqslant 1$，$\left(1-\frac{1}{x}\right)^x\leqslant\frac{1}{e}\leqslant\left(1-\frac{1}{x+1}\right)^x$。

- 对于任意 k，$\sum_{i=1}^{n} i^k = \Theta\left(\frac{n^{k+1}}{k+1}\right)$。
- 对于任意 $k>1$，$\sum_{i=1}^{\infty} i^{-k} < O(1)$。
- 对于任意 $c,\varepsilon>0$，$\sum_{i=1}^{\infty} \frac{i^c}{(1+\varepsilon)^i} < O(1)$。
- 对于任意 n，$\sum_{i=1}^{n} \frac{1}{i} = \ln n \pm O(1)$。

A.2.6　统计距离

下面的概念十分有用，它考虑如何比较两个分布中哪一个更接近于第三个分布。

定义 A.20（统计距离）　Ω 是一个有穷集合。对于值域为 Ω 的两个随机变量 X 和 Y，它们的统计距离（也称方差距离）定义为 $\Delta(X,Y)=\max_{S\subseteq\Omega}\{|\Pr[X\in S]-\Pr[Y\in S]|\}$。

对于统计距离，有些教科书称之为全方差距离。下面的引理刻画了这种距离的性质，
516 它十分有用。

引理 A.21　令 X，Y，Z 是取值位于有穷集合 Ω 的三个分布，则

1. $\Delta(X,Y)\in[0,1]$，并且 $\Delta(X,Y)=0$ 当且仅当 X 等于 Y。
2. （三角不等式）$\Delta(X,Z)\leqslant\Delta(X,Y)+\Delta(Y,Z)$。
3. $\Delta(X,Y)=\frac{1}{2}\sum_{x\in\Omega}|\Pr[X=x]-\Pr[Y=x]|$。
4. $\Delta(X,Y)\geqslant\varepsilon$ 当且仅当存在布尔函数 $f:\Omega\to\{0,1\}$ 使得 $|E[f(X)]-E[f(Y)]|\geqslant\varepsilon$。
5. 对任意有穷集合 Ω' 和函数 $f:\Omega\to\Omega'$，$\Delta(f(X),f(Y))\leqslant\Delta(X,Y)$。（这里，$f(X)$ 是在 X 的样本上应用函数 f 得到的取值于 Ω' 上的分布。）

注意，第 3 条结论意味着，$\Delta(X,Y)$ 等于 X 和 Y 的 L_1-距离（参见 A.5.4 节）除以 2。亦即，如果将 X 视为 $\mathbf{R}^{\Omega}$ 上的一个向量，其中 $X_\omega=\Pr[X=\omega]$，并对任意向量 $\boldsymbol{v}\in\mathbf{R}^{\Omega}$ 定义 $|\boldsymbol{v}|_1=\sum_{\omega\in\Omega}|\boldsymbol{v}_\omega|$，则 $\Delta(X,Y)=1/2|X-Y|_1$。

引理 A.21 的证明　我们先证明第 3 条结论。对于 $\{0,1\}^n$ 上的任意两个分布 X，Y，令 S 是由满足 $\Pr[X=x]>\Pr[Y=y]$ 的字符串构成的集合。容易看到，S 的选取使得 $b(S)=\Pr[X\in S]-\Pr[Y\in S]$ 的值达到最大并且 $b(S)=\Delta(X,Y)$。这是由于，如果存在集合 T 使得 $b(T)<-b(S)$，则 T 的补集 $\overline{T}$ 将满足 $b(\overline{T})>b(S)$。然而，

$$\begin{aligned}
&\sum_{x\in\{0,1\}^n}|\Pr[X=x]-\Pr[Y=x]|\\
&=\sum_{x\in S}(\Pr[X=x]-\Pr[Y=x])+\sum_{x\notin S}(\Pr[Y=x]-\Pr[X=x])\\
&=\Pr[X\in S]-\Pr[Y\in S]+(1-\Pr[Y\in S])-(1-\Pr[X\in S])\\
&=2\Pr[X\in S]-2\Pr[Y\in S]
\end{aligned}$$

由此得到第 3 条结论。

由第 3 条结论即可得到三角不等式，因为 $\Delta(X,Y)=1/2|X-Y|_1$，并且 L_1-距离满足三角不等式。第 3 条结论也蕴含了第 1 条结论，因为 $|X-Y|_1=0$ 当且仅当 $X=Y$，而且 $|X-Y|_1\leqslant|X|_1+|Y|_1=1+1$。

第 4 条结论是统计距离的定义的重述，只是用满足"$f(x)=1$ 当且仅当 $x\in S$"的函数 $f:\{0,1\}^*\to\{0,1\}$ 来表示集合 $S\subseteq\{0,1\}^*$。第 5 条结论由第 4 条结论导出。注意，如果 $\Delta(X,Y)\leqslant\varepsilon$，则 $|E[g(f(X))]-E[g(f(Y))]|\leqslant\varepsilon$ 对任意函数 g 均成立。 ■

A.3 数论和群

整数指的是集合 $\mathbf{Z}=\{0,\pm1,\pm2,\cdots\}$，而自然数指的是它的子集 $\mathbf{N}=\{0,1,2,\cdots\}$[⊖]。最基本的事实是，任意整数 n 可以被一个非零整数 k 除，产生整数 l，r 使得 $n=kl+r$ 且 $r\in\{0,1,\cdots,k-1\}$。如果 $r=0$，则称 k 整除 n 并记为 $k|n$。n 的因数指的是能够整除 n 的正整数。 517

整数 n，m 的最大公因数，记为 $gcd(n,m)$，是满足 $d|n$ 和 $d|m$ 的最大整数 d。如果 n 和 m 的最大公因数等于 1，则称 n 和 m 互素。不难证明，下列事实成立。

- 如果非零整数 c 整除 n 和 m，则 $c|d$；
- n 和 m 的最大公因数是使得条件"存在整数 x 和 y 使得 $nx+my=d$"成立的最小正整数 d。
- 存在多项式时间(即 polylog(n,m)时间)算法以 n，m 为输入，以 n，m 的最大公因数 d 和满足 $nx+my=d$ 的整数 x，y 为输出。(这就是人们熟知的欧几里得算法。)

如果一个数 p 仅有因数 1 和 p，则称之为素数。下面是关于素数的基本事实。

- 任意正整数 n 均可以唯一地表示为若干个素数的乘积(忽略素数的顺序)，该乘积称为 n 的素因数分解。
- 如果 $gcd(p,a)=1$ 且 $p|ab$，则 $p|b$。特别地，如果素数 p 整除 $a\cdot b$，则 $p|a$ 或 $p|b$。

数论中的一个基本问题是问，素数有多少。下面是一个著名的结果。

定理 A.22 (素数定理(阿达马(Hadamard)，瓦勒·普桑 de la Vallée Poussin)，1896) 对于 $n>1$，令 $\pi(n)$表示介于 1 到 n 之间的素数的个数，则

$$\pi(n)=\frac{n}{\ln n}(1\pm o(1))$$

素数定理的原始证明用到了深奥的数学工具，甚至人们还曾猜测这应该是素数定理固有的。然而，1949 年，埃德斯(Erdős)和泽尔贝格(Selberg)(独立地)发现了素数定理的初等证明。对于计算机科学的绝大多数应用，掌握下面稍弱的结论就足够了，该结论是由切比雪夫证明的。

定理 A.23 $\pi(n)=\Theta\left(\frac{n}{\log n}\right)$。

证明 考虑数 $\binom{2n}{n}=\frac{2n!}{n!\ n!}$。由斯特林公式可知，$\log\binom{2n}{n}=(1-o(1))2n$。特别地，$n\leqslant\log\binom{2n}{n}\leqslant2n$。并且，$\binom{2n}{n}$的所有素因数均介于 0 到 $2n$ 之间，且每个素因数 p 出现的次数均不超过 $k=\left\lfloor\frac{\log 2n}{\log p}\right\rfloor$。事实上，对任意 n，因数 p 在 $n!$的素因数分解中出现的次数是 $\sum_i\left\lfloor\frac{n}{p^i}\right\rfloor$，这是由于因数 p 在$\{1,2,\cdots,n\}$的素因数分解中恰出现$\left\lfloor\frac{n}{p}\right\rfloor$次，因数 p^2 恰出

⊖ 有的教科书上 $\mathbf{N}$ 不含 0，在绝大多数情况下，这没有区别。

现$\left\lfloor \frac{n}{p^2} \right\rfloor$次，依次类推。因此，素因数 p 在$\binom{2n}{n}=\frac{2n!}{n!\ n!}$的素因数分解中出现的次数等于$\sum_i \left(\left\lfloor \frac{2n}{p^i} \right\rfloor - 2\left\lfloor \frac{n}{p^i} \right\rfloor\right)$，其中求和符号至多对 k 个加项求和(因为 $p^{k+1}>2n$)且每个求和项非
518 0 即 1。

因此，$\binom{2n}{n} \leqslant \prod_{\substack{1<p\leqslant 2n \\ p\text{ 是素数}}} p^{\left\lfloor \frac{\log 2n}{\log p} \right\rfloor}$，取对数后得到

$$n \leqslant \log\binom{2n}{n} \leqslant \sum_{\substack{1<p\leqslant 2n \\ p\text{ 是素数}}} \left\lfloor \frac{\log 2n}{\log p} \right\rfloor \log p \leqslant \sum_{\substack{1<p\leqslant 2n \\ p\text{ 是素数}}} \log 2n = \pi(2n)\log 2n$$

由此可知，$\pi(n)=\Omega\left(\frac{n}{\log n}\right)$。

为证 $\pi(n)=O\left(\frac{n}{\log n}\right)$，定义函数 $\vartheta(n)=\sum_{\substack{1<p\leqslant n \\ p\text{ 是素数}}} \log p$。只需证明$\vartheta(n)=O(n)$(留作练习!)。

为此，由于介于 $n+1$ 到 $2n$ 之间的素数至少能整除$\binom{2n}{n}$一次，进而$\binom{2n}{n} \geqslant \prod_{\substack{n+1<p\leqslant 2n \\ p\text{ 是素数}}} p$，取对数得到

$$2n \geqslant \log\binom{2n}{n} \geqslant \sum_{\substack{n+1<p\leqslant 2n \\ p\text{ 是素数}}} \log p = \vartheta(2n)-\vartheta(n)$$

由此得到递归方程$\vartheta(2n)\leqslant\vartheta(n)+2n$，解得$\vartheta(n)=O(n)$。 ■

A.3.1　群

群是一种抽象的数学结构，它刻画了诸如整数、矩阵、函数等数学对象的一些性质。形式上，群是具有二元运算的集合，不妨将二元运算记为$\star$，该运算满足结合律且每个元素均具有逆元素。亦即，$(G,\star)$是一个群，如果

1. 任意 a，b，$c\in G$，则$(a\star b)\star c=a\star(b\star c)$。

2. 存在单位元素 $id\in G$ 使得 $a\star id=a$ 对任意 $a\in G$ 成立。并且，对于任意 $a\in G$，存在 $b\in G$ 使得 $a\star b=b\star a=id$。(元素 b 称为 a 的逆元素，通常记为 a^{-1} 或 $-a$。)

例如，整数集在加法运算下构成一个群(0 是单位元素)；非零实数集在乘法运算下构成一个群(1 是单位元素)；从定义域 A 到其自身的所有函数构成的集合在函数复合操作下构成一个群。

通常，人们更习惯用加法(+)或乘法(·)表示群运算，而不用$\star$。这样，la(相应地 a^l)就表示在 a 上执行 l 次运算后产生的结果。

A.3.2　有限群

具有有穷个元素的群称为有限群。群 G 中元素的个数记为 $|G|$。下面是有限群的一些实例。

- 介于 0 到 $n-1$ 之间的整数集 $\mathbf{Z}_n$，在对模 n 的加法运算下构成一个群。特别地，$\mathbf{Z}_2$ 是集合{0，1}在异或运算下构成的群。
- $[n]$上的所有置换在函数复合操作下构成群 S_n。

- 长度为 n 的所有二进制位串在按位异或操作下构成群 $(\mathbf{Z}_2)^n$。一般地，对于任意两个群 G 和 H，定义 $G\times H$ 是一个群，其元素是所有满足 $g\in G$，$h\in H$ 的对 (g, h)。其运算是将 G 和 H 上的运算分别作用于元素相应的分量上。在此意义下，G^n 表示群 $G\times G\times\cdots\times G$（$n$ 个 G）。 519
- 对任意 n，集合 $\{k: 1\leqslant k\leqslant n-1, gcd(k, n)=1\}$ 在对模 n 的乘法运算下构成群 $\mathbf{Z}_n^*$。注意，如果 $gcd(k, n)=1$，则存在 x，y 使得 $kx+ny=1$。换句话说，$kx\equiv 1(\mathrm{mod}\ n)$。这意味着，在模 n 的乘法运算下 x 是 k 的逆元素；它还说明，逆元素可以利用欧几里得算法在多项式时间内求得。$\mathbf{Z}_n^*$ 的大小用 $\varphi(n)$ 表示，函数 $\varphi(n)$ 即是人们熟知的欧拉商函数。注意，如果 n 是素数，则 $\varphi(n)=n-1$。已经证明，对于任意 $n>6$ 有 $\varphi(n)\geqslant\sqrt{n}$。

G 的子群是 G 的一个子集，它自身构成一个群（亦即，在群运算和求逆运算下封闭）。下面的结论非常有用。

定理 A.24 如果 G 是有限群且 H 是 G 的子群，则 $|H|$ 整除 $|G|$。

证明 考虑所有形如 $aH=\{ah: h\in H\}$ 的集合构成的集族，其中 $a\in G$ 是任意的。（这里，将群运算视为乘法。）易知，映射 $x\mapsto ax$ 是一对一的。因此，$|aH|=|H|$ 对任意 a 成立。于是，只需证明上述集族构成 G 的一个划分。显然，该集族覆盖 G（因为 $a\in aH$ 对任意 $a\in G$ 成立）。因此，只需证明，对任意 a，b，要么 $aH=bH$，要么 aH 和 bH 不相交。事实上，假设存在 x，$y\in H$ 使得 $ax=by$，则对任意 $az\in aH$ 有 $az=(byx^{-1})z$。再由 $yx^{-1}z\in H$ 可知，$az\in bH$。 ■

推论 A.25（费尔马小定理） 对任意 n 和 $x\in\{1, 2, \cdots, n-1\}$，则 $x^{\varphi(n)}\equiv 1(\mathrm{mod}\ n)$。特别地，如果 n 是素数，则 $x^{n-1}\equiv 1(\mathrm{mod}\ n)$。

证明 考虑集合 $H=\{x^l: l\in\mathbf{Z}\}$。显然，$H$ 是 $\mathbf{Z}_n^*$ 的子群。因此，$|H|$ 整除 $\varphi(n)$。而且，H 的大小是满足 $x^k\equiv 1(\mathrm{mod}\ n)$ 的最小整数 k。事实上，这样的 k 必然存在，因为在模 n 的条件下整数序列 1，x，x^2，x^3，$\cdots$ 均在群 $\mathbf{Z}_n^*$ 中，故存在整数 i，j 使得 $i<j$ 且 $x^i=x^j$，这意味着 $x^{j-i}\equiv 1(\mathrm{mod}\ n)$。因此，上述整数序列形如 1，x，x^2，$\cdots$，x^{k-1}，1，x，x^2，$\cdots$，这意味着 $|H|=k$。

由于 $x^{|H|}\equiv 1(\mathrm{mod}\ n)$ 且 $\varphi(n)$ 是 $|H|$ 的倍数，因此 x 的 $\varphi(n)$ 次方模 n 显然也得 1。 ■

群 G 中元素 x 的阶是使得 x^k 等于单位元素的最小 k 值。上面的证明过程表明，在有限群 G 中，任意元素的阶均是有限值，并且元素的阶整除 G 的大小。G 中阶为 $|G|$ 的元素称为 G 的生成元素，这是因为此时子群 $\langle x^1, x^2, \cdots\rangle$ 即为群 G ㊀。如果群 G 存在生成元素，则称 G 是周期的。例如，$\langle 0, \cdots, n-1\rangle$ 在模 n 的加法运算下构成的周期群 $\mathbf{Z}_n$，元素 1 是该群的一个生成元素（其他与 n 互素的数也是该群的生成元素，留作练习）。 520

A.3.3 中国剩余定理

令 $n=pq$，其中 p，q 互素。中国剩余定理（也称 CRT 定理）断言，（在模 n 的乘法运算下）群 $\mathbf{Z}_n^*$ 同构于群 $\mathbf{Z}_p^*\times\mathbf{Z}_q^*$（模 p 的乘法和模 q 的乘法分别作用于整数对的第一分量和第二分量）。

㊀ 更一般的定义是，如果子群 $\langle x^l: l\in\mathbf{Z}\rangle$ 等于群 G，则称 x 是 G 的生成元素。该定义对有限群也成立。

定理 A.26　如果 $n=pq$，其中 p，q 互素，则将 x 映射为 $\langle x(\bmod p), x(\bmod q)\rangle$ 的函数 f 是 $\mathbf{Z}_n^*$ 到 $\mathbf{Z}_p^* \times \mathbf{Z}_q^*$ 的一对一的映射。而且，在 $f(xy)=f(x)f(y)$ 的意义下 f 是一个同构映射，其中 xy 对 n 取模而 $f(x)f(y)$ 分别对 p 取模和对 q 取模。

证明　定理的后面部分容易验证，因此仅证明 f 是一对一的映射。为此，往证若 $f(x)=f(x')$ 则 $x=x'$。由于 $f(x-x')=f(x)-f(x')$，故只需证明，如果 $x=0(\bmod p)$（即 $p|x$）且 $x=0(\bmod q)$（即 $q|x$），则 $x=0(\bmod n)$（即 $pq|x$）。为此，将 $p|x$ 记为 $x=pk$。于是，由 $gcd(p, q)=1$ 和 $q|x$ 可知 $q|k$，这意味着 $pq|x$。 ■

中国剩余定理可以直接地推广为更一般的形式。亦即，对于任意 $n=p_1p_2\cdots p_k$，其中所有 p_i 两两互素，则 $\mathbf{Z}_n^*$ 与 $\mathbf{Z}_{p_1}^* \times \mathbf{Z}_{p_2}^* \times \cdots \times \mathbf{Z}_{p_k}^*$ 同构。这意味着，对于任意 n，群 $\mathbf{Z}_n^*$ 同构于若干个形如 $\mathbf{Z}_q^*$ 的群的乘积，其中 q 是素数的幂（即形如 p^i 的数，其中 p 是素数）。由于所有形如 $\mathbf{Z}_q^*$（其中 q 是奇素数的幂）的群均是周期群，而形如 $\mathbf{Z}_{2^i}^*$ 的群要么是周期群要么是两个周期群的乘积，因此中国剩余定理还可以进一步推广为：任意阿贝尔群 G 均同构于若干个周期群的乘积 $G_1 \times G_2 \times \cdots \times G_k$。

A.4　有限域

集合 $\mathbf{F}$ 及其上的加法运算（$+$）和乘法运算（$\cdot$）称为一个域，如果加法和乘法满足结合律、交换律和乘法分配律并且在加法和乘法运算下元素均分别存在逆元素，其中加法和乘法的单位元素分别记为 0 和 1。换句话说，$\mathbf{F}$ 是域，如果 $\mathbf{F}$ 在加法运算和单位元素 0 下是一个阿贝尔群，并且 $\mathbf{F}\setminus\{0\}$ 在乘法运算和单位元素 1 下是一个阿贝尔群，而且加法运算和乘法运算满足分配律 $a(b+c)=ab+ac$。

人们较熟悉的域包括实数域（$\mathbf{R}$），有理数域（$\mathbf{Q}$），复数域（$\mathbf{C}$），此外还存在有限域。回顾一下，对于素数 p，集合 $\{0, \cdots, p-1\}$ 在模 p 的加法运算下构成阿贝尔群，集合 $\{1, \cdots, p-1\}$ 在模 p 的乘法运算下也构成阿贝尔群。因此，集合 $\{0, \cdots, p-1\}$ 在上述两种运算下构
521 成一个域，记为 GF(p)。这类域中最简单的莫过于 GF(2)，它实际是集合 $\{0, 1\}$，其乘法是逻辑与操作（$\wedge$）而加法是异或操作。

对于任意有限域 $\mathbf{F}$，存在数 l 使得“$x+x+\cdots+x$（l 个 x）等于 F 的 0 元素”对任意 $x\in\mathbf{F}$ 成立（留作练习）。数 l 称为域 $\mathbf{F}$ 的特征数。对于任意素数 q，域 GF(q) 的特征数等于 q。

A.4.1　非素域

易知，如果 n 不是素数，则集合 $\{0, \cdots, n-1\}$ 在模 n 的加法和乘法下不是一个域，因为该集合中存在非零元素 x，y 使得 $x\cdot y=n=0(\bmod n)$。然而，即便 n 不是素数，仍然存在大小为 n 的有限域。具体地讲，对于任意素数 q 和 $k\geqslant 1$，存在恰有 q^k 个元素的域，记为 GF(q^k)。本书很少用到这种域，但下面仍概要地介绍其构造。

任意素数 q 和 k，域 GF(q) 上存在 k 次不可约多项式 P（如果多项式 P 不能表示成两个次数更低的多项式 P'，P'' 的乘积，则称 P 是不可约的）。令 GF(q^k) 为 GF(q) 上的所有 $(k-1)$ 次多项式构成的集合，其中每个多项式均可以表示为 k 个系数组成的向量。我们规定多项式的加法和乘法完成之后对多项式 P 求余式。注意，加法对应于 GF(q) 上的 k 维向量的标准的加法，并且加法和乘法均容易在 poly(k, log$q$) 时间内完成（多项式 S 对多项式 P 求余式可以采用类似于整数长除法的算法）。可以证明，无论如何选取不可约多项式 P，

通过对元素重命名，最终将产生同一个域。并且，存在确定性算法在 poly(q, k)时间内构造出 GF(q)上的一个 k 次不可约多项式。此外，还存在概率算法(和确定性算法，但其分析依赖于未被证明的假设)在 poly($\log q$, k)时间内构造出这样的不可约多项式(参见书[Sho05])。

对于本书，最重要的有限域是 GF(2^k)，它是集合$\{0,1\}^k$，其加法是按位异或操作，其乘法是对一个不可约多项式求余式的多项式乘法，所用的不可约多项式可以在 poly(k)时间内确定。实际上，我们绝大多数时候根本不使用 GF(2^k)的乘法，而仅使用其加法(亦即，将 GF(2^k)视为向量空间，见下节)。

A.5 线性代数基础

对于域 $\mathbf{F}$ 和 $n\in\mathbf{N}$，我们用 $\mathbf{F}^n$ 表示由 $\mathbf{F}$ 的元素构成的长度为 n 的元组(或向量)构成的集合。如果 $\boldsymbol{u}$，$\boldsymbol{v}\in\mathbf{F}^n$ 且 $x\in\mathbf{F}$，则我们用 $\boldsymbol{u}+\boldsymbol{v}$ 表示由 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 的对应分量相加得到的向量，并用 $x\boldsymbol{u}$ 表示将 $\boldsymbol{u}$ 的每个分量乘以 x 后得到的向量。

$\mathbf{F}^n$ 上的一系列向量 $\boldsymbol{u}^1$，…，$\boldsymbol{u}^k$ 是线性无关的，如果方程 $x_1\boldsymbol{u}^1+\cdots+x_k\boldsymbol{u}^k=\mathbf{0}$(其中 $\mathbf{0}$ 表示分量全为 0 的向量)只有解 $x_1=x_2=\cdots=x_k=0$。可以证明，如果向量 $\boldsymbol{u}^1$，…，$\boldsymbol{u}^k$ 线性无关则 $k\leqslant n$(留作练习)。$\mathbf{F}^n$ 上线性无关的 n 个向量构成的集合称为 $\mathbf{F}^n$ 的一个基。不难看到， 522
如果 $\boldsymbol{u}^1$，…，$\boldsymbol{u}^n$ 是 $\mathbf{F}^n$ 的基，则任意 $\boldsymbol{v}\in\mathbf{F}^n$ 均可以表达成 $\boldsymbol{u}^1$，…，$\boldsymbol{u}^n$ 的线性组合 $\boldsymbol{v}=\sum_i x_i\boldsymbol{u}^i$ 且这种线性组合是唯一的。$\mathbf{F}^n$ 的标准基指的是 $\boldsymbol{e}^1$，…，$\boldsymbol{e}^n$，其中 $\boldsymbol{e}^i_j$ 等于 1，如果 $i=j$；否则 $\boldsymbol{e}^i_j$ 等于 0。

如果子集 $S\subseteq\mathbf{F}^n$ 在加法和标量乘法下是封闭的(亦即 $\boldsymbol{u}$，$\boldsymbol{v}\in S$ 和 x，$y\in\mathbf{F}$ 蕴含 $x\boldsymbol{u}+y\boldsymbol{v}\in S$)，则称 S 是一个子空间。子空间 S 的维数，记为 $\dim(S)$，定义为使得"S 中存在 k 个线性无关向量"成立的最大 k 值。S 中的 $\dim(S)$ 个线性无关的向量构成的集合称为 S 的一个基。易知，S 中任意向量可以表示成基的线性组合。

如果函数 $f:\mathbf{F}^n\to\mathbf{F}^m$ 满足 $f(\boldsymbol{u}+\boldsymbol{v})=f(\boldsymbol{u})+f(\boldsymbol{v})$，则称 f 是线性的。不难验证，线性函数具有下列性质：

- 如果 $\boldsymbol{u}^1$，…，$\boldsymbol{u}^n$ 是 $\mathbf{F}^n$ 的基，则对任意 $\boldsymbol{v}\in\mathbf{F}^n$ 有 $f(\boldsymbol{v})=\sum_i x_i f(\boldsymbol{u}^i)$，其中 x_1，…，x_n 是使得 $\boldsymbol{v}=\sum_i x_i\boldsymbol{u}^i$ 成立的元素。于是，欲知 f 在各个点的取值，仅需知道它在基元素上的取值。
- 集合 $Im(f)=\{f(\boldsymbol{v}):\boldsymbol{v}\in\mathbf{F}^n\}$ 是 $\mathbf{F}^m$ 的子空间。
- 集合 $Ker(f)=\{\boldsymbol{v}:f(\boldsymbol{v})=0\}$ 是 $\mathbf{F}^n$ 的子空间。
- $\dim(Im(f))+\dim(Ker(f))=n$。

线性函数 $f:\mathbf{F}^n\to\mathbf{F}^m$ 通常表示为一个 $m\times n$ 的矩阵 A，其第 i 列是 $f(\boldsymbol{e}^i)$。$m\times n$ 的矩阵 A 与 $n\times k$ 的矩阵 B 的乘积是一个 $m\times k$ 的矩阵 $C=AB$，其中 $C_{i,j}=\sum_{l\in[n]}A_{i,l}B_{l,j}$。不难验证，如果矩阵 A 表示线性函数 $f:\mathbf{F}^n\to\mathbf{F}^m$ 而矩阵 B 表示线性函数 $g:\mathbf{F}^m\to\mathbf{F}^p$，则矩阵 C 表示线性函数 $h:\mathbf{F}^n\to\mathbf{F}^p$，它将向量 $\boldsymbol{v}$ 映射为 $g(f(\boldsymbol{v}))$。还可以证明，如果将 $\mathbf{F}^n$ 的元素表示成 $n\times 1$ 的矩阵(即列向量)，则 $f(\boldsymbol{v})=A\boldsymbol{v}$。

$n\times n$ 矩阵 A 的行列式，记为 $\det(A)$，等于 $\sum_{\sigma\in S_n}(-1)^{\mathrm{sgn}(\sigma)}\prod_{i=1}^{n}A_{i,\sigma(i)}$，其中 S_n 是$[n]$上的置换群，并且 $\mathrm{sgn}(\sigma)$在使得"$i<j$ 但 $\sigma(i)>\sigma(j)$"成立的(i,j)对的个数为奇数的情况下等

于 1，$\mathrm{sgn}(\sigma)$在其他情况下等于 0[⊖]。下述两个事实成立。

- $\det(AB)=\det(A)\det(B)$。直接计算即可验证该事实。
- 如果 A 是上三角矩阵(亦即，只要 $i>j$ 则 $A_{i,j}=0$)，则 $\det(A)=A_{1,1}A_{2,2}\cdots A_{n,n}$。事实上，在此情况下，对行列式的值具有非 0 贡献的置换 σ 必须满足：$\sigma(i)\geqslant i$ 对任意 i 成立，这样的置换必是恒等置换。

利用上述两个事实可以得到计算矩阵 A 的行列式的多项式时间算法，该算法就是著名的高斯消去法。该算法将矩阵 A 表示为 $E_1E_2\cdots E_mD$ 的形式，其中 D 是上三角矩阵，而所有 E_i 全是初等矩阵(一个矩阵乘以一个初等矩阵，相当于调换该矩阵的两列，或者将该矩阵的某列元素全部乘以域中的一个元素，或者将该矩阵的一个列加到另一个列上)。由于
523 初等矩阵的行列式很容易计算，因此 A 的行列式也很容易计算。

下面的引理将矩阵的行列式关联到该矩阵表示的线性函数上。

引理 A.27　如果线性函数 $f:\mathbf{F}^n\to\mathbf{F}^n$ 的矩阵表示是 A，则下列条件等价。

- 矩阵 A 的所有列构成 $\mathbf{F}^n$ 的一个基。
- f 是一对一的函数。
- $\dim(Im(f))=n$。
- $\dim(Ker(f))=0$。
- $\det(A)\neq 0$。
- 存在 $v\in\mathbf{F}^n$ 使得方程 $Ax=v$ 恰有一个解。
- 对任意 $v\in\mathbf{F}^n$，方程 $Ax=v$ 恰有一个解。

而且，如果 f 是一对一的函数，则映射 f^{-1} 是线性函数并且可以表示为一个 $n\times n$ 的矩阵 A^{-1}，该矩阵的(i,j)位置取值为 $\dfrac{\det(A_{-(i,j)})}{\det(A)}$，其中 $A_{-(i,j)}$ 是删除矩阵 A 的第 i 行和第 j 列之后剩下的$(n-1)\times(n-1)$的矩阵。

A.5.1　内积

向量空间 $\mathbf{R}^n$ 和 $\mathbf{C}^n$ 上的加性结构通常十分有用[⊖]。$\mathbf{C}^n$ 上的内积是一个函数，它将向量 u，v 映射为复数$\langle u,v\rangle$。内积满足下列条件：

- $\langle xu+yw,v\rangle=x\langle u,v\rangle+y\langle w,v\rangle$。
- $\langle v,u\rangle=\overline{\langle u,v\rangle}$，其中$\bar{z}$表示 z 的共轭复数(即，如果 $z=a+ib$ 则$\bar{z}=a-ib$)。
- 对于任意 u，$\langle u,u\rangle$是非负实数，并且$\langle u,u\rangle=0$ 当且仅当 $u=\mathbf{0}$。

本书将用到两种内积。一种是标准的内积，它将 x，$y\in\mathbf{C}^n$映射到 $\sum_{i=1}^{n}x_i\overline{y_i}$。另一种是期望内积或范式内积，它将 x，$y\in\mathbf{C}^n$映射到 $\frac{1}{n}\sum_{i=1}^{n}x_i\overline{y_i}$。空间 $\mathbf{R}^n$ 上也可以定义内积，这

⊖ 人们已经证明，任意置换 $\sigma\in S_n$ 均可以表示为若干个对换的复合，对换是一种特殊的置换，每个对换仅对调$[n]$中两个元素并将其他元素保持不变(冒泡排序算法可以证明上述结论)。如果对换 τ_1，…，τ_m 的复合等于置换 σ，则 $\mathrm{sgn}(\sigma)$等于$(-1)^m$。可以证明，$\mathrm{sgn}(\sigma)$不依赖于复合得到 σ 的对换。因此，在此含义下 $\mathrm{sgn}(\sigma)$也是良定义的。

⊖ 限定在实数域和复数域进行讨论的原因在于，它们的特征数为 0。这意味着，不存在整数 $k\in\mathbf{N}$ 和非零元素 $a\in\mathbf{F}$ 使得 $ka=0$(其中 ka 表示 a 与其自身相加 k 次)。读者可以证明，如果域 $\mathbf{F}$ 中存在这样的数，则 $\mathbf{F}^n$ 上不能定义内积。

只需从前面的定义中去掉共轭符号。

如果$\langle u, v\rangle=0$，则称u和v是垂直的，记为$u\perp v$。下面的结论成立。

引理 A.28 如果非零向量$u^1, \cdots, u^n$使得$u^i\perp v^j$对任意$i\neq j$成立，则它们是线性无关的。

证明 假设$\sum_i x_i u^i=\mathbf{0}$，将该向量与其自身做内积，得到

$$0=\langle\sum_i x_i u^i, \sum_j x_j u^j\rangle=\sum_{i,j} x_i \bar{x}_j\langle u^i, u^j\rangle=\sum_i |x_i|^2\langle u^i, u^i\rangle \qquad (A.7)$$

(A.7)式中最后的等式基于$\langle u^i, u^j\rangle=0$对任意$i\neq j$成立这一事实。然而，除非所有$x_i$均为0，(A.7)式右端将严格大于0。(注意，对于复数$x=a+ib$，$|x|=\sqrt{a^2+b^2}$且$|x|^2=x\bar{x}$。)■

$\mathbf{C}^n$上使得$\langle u^i, u^j\rangle=0$对任意$i\neq j$成立的非零向量$u^1, \cdots, u^n$称为$\mathbf{C}^n$的正交基。如果正交基还使得$\langle u^i, u^i\rangle=1$对任意$i$成立，则称该正交基为标准正交基。标准正交基由$n$个线性无关的向量构成，因此它也是空间$\mathbf{C}^n$的一个基，这意味着任意向量$v$均可以表示为$v=\sum_i x_i u^i$，由该等式与$u^i$做内积可得$x_i=\langle v, u^i\rangle$。

下面的恒等式是勾股定理的一般形式，它非常有用。

引理 A.29 (巴塞弗恒等式) 如果向量$u^1, \cdots, u^n$是$\mathbf{C}^n$的标准正交基，则对任意v有

$$\langle v, v\rangle=\sum_{i=1}^{n} |x_i|^2$$

其中$x_1, \cdots, x_n$是使得$v=\sum_i x_i u^i$成立的数。

证明 同引理 A.28 一样的证明过程，可得

$$\langle v, v\rangle=\langle\sum_i x_i u^i, \sum_j x_j u^j\rangle=\sum_i |x_i|^2\langle u^i, u^i\rangle$$ ■

定义了内积的向量空间也称为希尔伯特空间。

A.5.2 点积

如果空间$\mathbf{F}$无法定义内积，则可以定义点积。向量$u, v\in\mathbf{F}^n$的点积，记为$u\odot v$，定义为$\sum_{i=1}^{n} u_i v_i$。对于任意子空间$S\sqsubset\mathbf{F}^n$，定义$S^\perp=\{u: u\odot v=0, \forall v\in S\}$。下面的论断成立，其证明很简单，留作练习。

论断 A.30 $\dim(S)+\dim(S^\perp)=n$。

特别地，对于任意非零向量$u\in\mathbf{F}^n$，子空间$u^\perp$由所有满足$u\odot v=0$的向量v构成，其维数等于$n-1$，进而其大小为$|\mathbf{F}|^{n-1}$。作为推论，我们得出下面有用的事实。

论断 A.31 (随机子和原理) 对于任意非零的$u\in\mathrm{GF}(2)^n$(集合$\{0, 1\}^n$在模2的加法和乘法下构成的域)，有

$$\Pr_{v\in_R\mathrm{GF}(2)^n}[u\odot v=0]=1/2$$

A.5.3 特征向量和特征值

设A是一个$n\times n$的复数矩阵，$v\in\mathbf{C}^n$是一个非零向量，如果存在$\lambda\in\mathbf{C}$使得$Av=$

λv，则 v 称为 A 的一个特征向量，λ 称为 A 的一个特征值。如果矩阵 A 的特征向量 v_1，…，v_n 构成一个基，则称 A 是可对角化的，亦即存在可逆矩阵 P 使得 PAP^{-1} 是一个对角矩阵。

注意，矩阵 A 存在与特征值 λ 关联的特征向量当且仅当矩阵 $A-\lambda I$ 不可逆，其中 I 是单位矩阵。因此，特征值 λ 均是多项式 $p(x)=\det(A-\lambda I)$ 的根。由代数基本定理（复数多项式的根的个数等于该多项式的次数）可知，任意方阵至少存在一个特征值（不可逆方阵以 0 作为特征值）。

矩阵 A 的共轭转置，记为 A^*，是使得 $A^*_{i,j}=\overline{A}_{j,i}$ 对任意 i，j 成立的矩阵，其中上划线表示复数共轭操作。如果矩阵 A 满足 $A=A^*$，则称 A 是厄尔米特矩阵。仅含实数元素的厄尔米特矩阵称为对称矩阵。亦即，如果实数矩阵 A 满足 $A=A^{\mathrm{T}}$，其中 A^{T} 表示 A 的转置（即 $A^{\mathrm{T}}_{i,j}=A_{j,i}$），则 A 是对称矩阵。作为练习，请读者证明下面的等价条件：矩阵 A 是厄尔米特矩阵当且仅当

$$< Au, v > = < Av, u > \tag{A.8}$$

对任意向量 u，v 成立。

厄尔米特矩阵的一个重要而有用的性质由下面的定理给出。

定理 A.32　如果 A 是一个 $n \times n$ 的厄尔米特矩阵，则 A 的特征向量构成一个正交基。

证明　对 n 作数学归纳。我们已经知道，矩阵 A 存在与特征值 λ 关联的特征向量 v。现在，令 $S=v^{\perp}$ 表示由所有与 v 正交的向量构成的 $n-1$ 维子空间。我们断言，$Au \in S$ 对任意 $u \in S$ 成立。事实上，如果 $<u, v>=0$，则

$$< Au, v > = < u, Av > = \bar{\lambda} < u, v > = 0$$

因此，A 限制在 S 上得到 $n-1$ 维空间上满足（A.8）式的线性变换，由归纳假设，限制后的线性变换 A 的特征向量 v_2，…，v_n 构成 S 的一个正交基。将 v 添加进来即得到由 A 的特征向量构成的 n 维正交基。 ■

注意，如果 A 是实对称矩阵，则其所有特征值也是实数（即不含虚部）。事实上，如果 $Av=\lambda v$，则

$$\lambda < v, v > = < Av, v > = < v, Av > = \bar{\lambda} < v, v >$$

上式表明：对于非零向量 v 有 $\lambda=\bar{\lambda}$。这意味着，特征向量的所有分量也是实数，因为它们是由实系数线性方程解得的。

A.5.4　范数

$\mathbf{C}^n$ 空间的范数将每个向量 v 映射为一个实数 $\|v\|$，且

- 对于任意向量 v，$\|v\| \geqslant 0$，且 $\|v\|=0$ 当且仅当 $v=\mathbf{0}$。
- 如果 $x \in \mathbf{C}$，则 $\|xv\| = |x| \|v\|$。
- （三角不等式）对于任意向量 u，v，$\|u+v\| \leqslant \|u\| + \|v\|$。

对任意 $v \in \mathbf{C}^n$ 和 $p>1$，v 的 L_p 范数记为 $\|v\|_p$，等于 $\left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p}$。$p=2$ 的情况具有特别的意义，亦即 $\|v\|_2 = \sqrt{\sum_{i=1}^{n} |v_i|^2} = \sqrt{\langle v, v \rangle}$，这种范数称为欧几里得范数。$p=1$ 的情况也很有意义，这种范数仅用一条竖线表示，即 $|v|_1 = \sum_{i=1}^{n} |v_i|$。$p=\infty$ 时，范数表示

526

为 $\|\boldsymbol{v}\|_\infty=\lim_{p\to\infty}\|\boldsymbol{v}\|_p=\max_{i\in[n]}|v_i|$。

不同范数之间的联系可以通过霍尔德不等式来建立。霍尔德不等式断言，对于满足 $\frac{1}{p}+\frac{1}{q}=1$ 任意的 p，q 有 $\|\boldsymbol{u}\|_p\|\boldsymbol{v}\|_q\geqslant\sum_{i=1}^{n}|u_iv_i|$。为证明该不等式，首先注意到，由于可以对向量进行伸缩，故仅需考虑单位长度的向量。因此，仅需证明：如果 $\|\boldsymbol{u}\|_p=\|\boldsymbol{v}\|_q=1$ 则 $\sum_{i=1}^{n}|u_iv_i|\leqslant 1$。这是由于，$\sum_{i=1}^{n}|u_i\|v_i|=\sum_{i=1}^{n}|u_i|^{p(1/p)}|v_i|^{q(1/q)}\leqslant\sum_{i=1}^{n}\left(\frac{1}{p}|u_i|^p+\frac{1}{q}|v_i|^q\right)=\frac{1}{p}+\frac{1}{q}=1$，其中最后一个不等式成立是因为任意 a，$b>0$ 和任意 $\alpha\in[0,1]$ 均满足不等式 $a^\alpha b^{1-\alpha}\leqslant\alpha a+(1-\alpha)b$。

霍尔德不等式意味着，任意向量的 L_2 范数、L_1 范数和 L_∞ 范数满足如下关系（参见习题 21.2）。

$$|\boldsymbol{v}|_1/\sqrt{n}\leqslant\|\boldsymbol{v}\|_2\leqslant\sqrt{|\boldsymbol{v}|_1\|\boldsymbol{v}\|_\infty} \tag{A.9}$$

定义了范数的向量空间通常称为巴拿赫空间。

A.5.5 度量空间

对于任意集合 Ω，函数 $d:\Omega^2\to\mathbf{R}$ 称为 Ω 上的一个度量，如果 d 满足下列条件

1. $d(x,y)\geqslant 0$ 对任意 x，$y\in\Omega$ 成立，且 $d(x,y)=0$ 当且仅当 $x=y$。
2. $d(x,y)=d(y,x)$ 对任意 x，$y\in\Omega$ 成立。
3. （三角不等式）$d(x,z)\leqslant d(x,y)+d(y,z)$ 对任意 x，y，$z\in\Omega$ 成立。

这意味着，$d(x,y)$ 在某种测度下表示了 x 和 y 之间的距离。例如，如果 Ω 上定义了范数，则函数 $d(x,y)=\|x-y\|$ 是 Ω 上的一个度量。除此之外，也存在不是由范数定义的其他度量。例如，对任意图 G，可以如下定义 G 的顶点集上的一个度量：x 和 y 的距离等于这两个顶点之间的最短路径的长度。近年来，不同度量空间及其相互关系在理论计算机科学中的应用日益广泛。关于这方面内容，[Mat02]第 15 章给出了很好的综述。

A.6 多项式

下面列举一元多项式的几个事实。

定理 A.33 非零 d 次多项式至多存在 d 个相异的根。

证明 假设域 $\mathbf{F}$ 上的多项式 $p(x)=\sum_{i=0}^{d}c_ix^i$ 有 $d+1$ 个相异的根 a_1，…，a_{d+1}，则对 $j=1$，…，$d+1$ 均有

$$\sum_{i=0}^{d}a_j^i\cdot c_i=p(a_j)=0$$

这意味着方程组 $A\boldsymbol{y}=\boldsymbol{0}$ 有解 $\boldsymbol{y}=\boldsymbol{c}$，其中

$$A=\begin{bmatrix}1 & a_1 & a_1^2 & \cdots & a_1^d\\ 1 & a_2 & a_2^2 & \cdots & a_2^d\\ \cdots & \cdots & \cdots & \cdots & \cdots\\ 1 & a_{d+1} & a_{d+1}^2 & \cdots & a_{d+1}^d\end{bmatrix}$$

527

上述矩阵 A 是一个范德蒙特矩阵。可以证明，A 的行列式为

$$\det A = \prod_{i>j} (a_i - a_j)$$

由于 a_i 各不相同，行列式不等于 0。因此，$\operatorname{rank} A = d+1$。于是，方程组 $A\mathbf{y} = \mathbf{0}$ 只有平凡解 $\mathbf{0}$，这与 $\mathbf{c} \neq \mathbf{0}$ 矛盾。■

上述定理具有如下非平凡的推论。

推论 A.34 *在任意有限域 $\mathbf{F}$ 上，乘法群 $\mathbf{F}^*$ 是周期的。*

证明 多项式 $x^k - 1$ 至多有 k 个根这一事实表明，群 $\mathbf{F}^*$ 具有如下的(*)性质：对任意 k 值，满足 $x^k = 1$ 的元素至多有 k 个。下面用数学归纳法证明，满足(*)性质的任意群均是周期的。

设 $|G| = n$，我们分下列三种情况完成证明。

- n 是素数。此时，任意元素的阶要么为 1 要么为 n。由于阶为 1 的元素只有单位元素，因此 G 含有阶为 n 的元素，故 G 是周期的。
- $n = p^c$，其中 p 互素，$c > 1$。此时，如果不存在阶为 n 的元素，则所有元素的阶均能整数 p^{c-1}。因此，我们找到 $n = p^c$ 个满足 $x^{p^{c-1}} = 1$ 的元素，这与 G 满足(*)性质矛盾。
- $n = pq$，其中 p 和 q 互素。此时，令 H 和 F 是如下定义的 G 的子群，其中 $H = \{a: a^p = 1\}$，$F = \{b: b^q = 1\}$。于是，$|H| \leqslant p < n$ 而 $|F| \leqslant q < n$。此外，作为 G 的子群，H 和 F 均满足(*)性质。因此，由归纳假设，H 和 F 是生成元素分别 a 和 b 的周期群。我们断言，ab 生成群 G。事实上，令 c 是 G 的任意元素，由于 p 和 q 互素，因此存在 x，y 使得 $xq + yp = 1$，进而 $c = c^{xq+yp}$。再由 $(c^q)^p = 1$ 和 $(c^p)^q = 1$ 可知，c 是 H 的一个元素和 F 的一个元素的乘积，于是存在 $i \in \{0, \cdots, p-1\}$ 和 $j \in \{0, \cdots, q-1\}$ 使得 $c = a^i b^j$。因此，欲证 $c = (ab)^z$ 对某个 z 成立，仅需找出同时满足 $z = i \pmod p$ 和 $z = i \pmod q$ 的 z 值，而这可以通过中国剩余定理完成。■

定理 A.35 *对任意的对序列 (a_1, b_1)，$\cdots$，(a_{d+1}, b_{d+1})，存在唯一的次数不超过 d 的多项式 $g(x)$ 使得 $g(a_i) = b_i$ 对 $i = 1, 2, \cdots, d+1$ 均成立。*

证明 下面的拉格朗日插值多项式满足定理的要求。

528

$$\sum_{i=1}^{d+1} b_i \cdot \frac{\prod_{j \neq i} (x - a_j)}{\prod_{j \neq i} (a_i - a_j)}$$

如果两个多项式 $g_1(x)$，$g_2(x)$ 均满足定理，则它们的差 $p(x) = g_1(x) - g_2(x)$ 的次数不超过 d 并且在 $x = a_1$，$\cdots$，a_{d+1} 上均取值为 0。因此，由前一个定理可知，$p(x)$ 必为 0，即 $g_1(x)$，$g_2(x)$ 是相同的多项式。■

理论计算机科学界通常将下面的初等结果归功于施瓦兹(Schwartz)和兹佩尔(Zippel)，尽管该结果在他们之前早就被发现了(参见德米洛(DeMillo)和利普顿(Lipton)的著作[DLne])。

引理 A.36 *如果域 $\mathbf{F} = \mathrm{GF}(q)$ 上的非零多项式 $p(x_1, x_2, \cdots, x_m)$ 的次数不超过 d，则*

$$\Pr[p(a_1, a_2, \cdots, a_m) \neq 0] \geqslant 1 - \frac{d}{q}$$

其中的概率考虑 $a_1, a_2, \cdots, a_m \in \mathbf{F}$ 的所有取法。

证明 我们对 m 用数学归纳法。如果 $m=1$，则由定理 A.33 可知引理成立。假设引理在变量个数小于等于 $m-1$ 时成立。先将多项式 p 可以改写成

$$p(x_1, x_2, \cdots, x_m) = \sum_{i=0}^{d} x_1^i p_i(x_2, \cdots, x_m)$$

其中 p_i 的次数不超过 $d-i$。由于 p 不等于 0，所以至少存在一个 p_i 不等于 0。令 i 是使得 p_i 不等于 0 的最大下标，则由归纳假设得到

$$\Pr_{a_2, \cdots, a_m} [p_i(a_2, \cdots, a_m) \neq 0] \geqslant 1 - \frac{d-i}{q}$$

只要 $p_i(a_2, \cdots, a_m) \neq 0$，$p(x_1, a_2, \cdots, a_m)$ 就是一个非零的一元 i 次多项式，因此它至多在 x_1 的 i 个值上取值为 0。于是，

$$\Pr[p(a_1, a_2, \cdots, a_m) \neq 0] \geqslant \left(1 - \frac{i}{q}\right)\left(1 - \frac{d-i}{q}\right) \geqslant 1 - \frac{d}{q}$$

这就完成了归纳证明。 ■ 529

部分习题的提示

第 0 章

0.2 答案依次为：(a)n (b)n^2 (c)2^n (d)$\log n$ (e)n (f)$n\log n$ (g)$n^{\log n}$ (h)n^2

第 1 章

1.1 根据小学竖式加法。

1.5 利用论断 1.6 的证明。

1.6 证明定理 1.9 的证明过程所得的通用图灵机U可以调整为散漫图灵机。

1.12 **b.** 如有必要可能需要切换到$\mathcal{S}$的补集，这样就可以假设空函数 O(在任何输入上都没有定义的函数)属于$\mathcal{S}$，而且还可以假设存在一个在某个 x 上有定义的函数 f 不属于$\mathcal{S}$。由此给出一个计算 $f_{\mathcal{S}}$的算法使得它能够计算函数 HALT_x，其中 HALT_x在输入 a 上输入 1 当且仅当 M_a 在 x 上停机。然后，将 HALT 的计算归约为对 HALT_x的计算。由此根据定理 1.11 证得莱斯定理。

第 2 章

2.2 习题 1.14 已经证明了 CONNECTED 和 2COL 属于 **P**(只是当时将 2COL 称为 BIPARTITE)。习题 2.21 将证明 3COL 是 **NP**-完全的，因此它不太可能属于 **P**。

2.3 先证明，对于任意有理数矩阵 A，其行列式的值都可以表示成一个位串并且该位串的长度是矩阵的二进制表示的总长度的多项式。然后，利用克莱姆法则(Gramer's Rule)将方程组的解用行列式表示出来。

2.4 利用习题 2.3。

2.5 n 是素数的证明包括 $n-1$ 的所有素因子 q_1，…，q_l和相应的整数 a_1，…，a_l以及 q_1，…，q_l是素数的(递归)证明。

2.6 **b.** 直接改造定理 1.9 的证明就可以得到 $O(|a|T\log T)$时间的模拟过程。要实现更高效的模拟，主要思想是，先模拟 M 的运行，但不要实际读取工作带的内容，而是简单地非确定地猜测工作带上的内容并将这些猜测写下来。然后，查验各条带上的猜测都是一致的。

531

2.11 为什么这个语言属于 **NP**？布尔公式的满足性问题是数学命题吗？

2.13 **a.** 修改机器 M 使得它在输出 1 之前清空工作带上的内容并将两个读写头都移动到带的末端。这样，最后的快照和读写头的位置是唯一的。

2.15 从 INDSET 进行归约。

2.17 对于 EXACTLY ONE 3SAT，将文字 v_i在各个子句 C 中的每次出现都替换为一个新变量 $z_{i,C}$，子句也相应变化，并且可能还需要引入辅助变量，以确保：如果 v_i取 TRUE，则 $z_{i,C}$既可以取 TRUE 也可以取 FALSE；但是，如果 v_i取 FALSE，则 $z_{i,C}$只能取 FALSE。将 EXACTLY ONE 3SAT 归约为 SUBSET SUM 的方法是，给定公式 φ，将每个可能出现的问题 u_i映射为一个数 $\sum_{j\in S_i}(2n)^j$，其中 S_i是包含文字 u_i的所有子句，再令目标值 T取 $\sum_{j=1}^{n}(2n)^j$。这样就得到 SUBSET SUM 的一个实例。此外，还需要用一个技巧来保证子集和实例的解所用到的两个文字不会分别对应于一个变量和该变量的否定。

2.19 从 SAT 进行归约。

2.20 你可以将 $x\in\{0,1\}$这一约束表示为 $x^2=x$。

2.21 从 3SAT 进行归约。

2.22 从 SAT 进行归约。

2.30 如果从 3SAT 到某个一元语言 L 存在 n^i 时间的归约，则该归约只能将 3SAT 中规模为 n 的实例映射为形如 1^i 的实例，其中 $i\leqslant n^i$。利用上述观察和定理 2.18 中的向下自归约论证过程，得出 3SAT 的一个多项式时间算法。

2.31 考虑 SUBSET SUM 的指数时间递归算法，证明：在习题给定的条件下，用一张表格存储之前的算得的值，你就能将上述算法改造成一个多项式时间算法。

第 3 章

3.6 a. 要计算 $H(n)$，需要(1)在任意 $i\leqslant\log n$ 上计算 $H(i)$；(2)在长度至多为 $\log n$ 的输入上模拟至多 $\log\log n$ 个机器运行少于 $\log\log n(\log n)^{\log\log n}=o(n)$ 个步骤；(3)在长度不超过 $\log n$ 的输入上计算 SAT。因此，如果 $T(n)$ 表示计算 $H(n)$ 所需的时间，则 $T(n)\leqslant\log nT(\log n)+O(n^2)$，进而 $T(n)=O(n^2)$。

3.7 b. 如果 f 是从 SAT 到 SAT_H 的归约，且其时间复杂度为 $O(n^i)$。令 N 是"使得 $H(n)>i$ 对 $n>N$ 均成立"的一个数。下面的递归算法 A 可以在多项式时间内求解 SAT：在输入的公式 φ 上，如果 $|\varphi|\leqslant N$，则用蛮力算法计算得到输出；否则，计算 $x=f(\varphi)$。如果 x 不是形如 $\Psi01^{m}$ 的串，则输出 FALSE；否则，输出 $A(\Psi)$。

第 4 章

4.6 第 2 章证明库克-勒维定理时使用了散漫图灵机。你需要验证，评注 1.7 和习题 1.5 指出的过程所构造的散漫图灵机满足如下性质：在任何步骤中读写头的位置都能在对数空间内被计算出来。 532

4.7 利用习题 4.6。

第 5 章

5.1 利用 SAT 的 **NP**-完全性。

5.7 利用定理 5.11 的证明过程的思想来证明非平凡的方向 **EXP**⊆**APSPACE**。

5.13 b. 从 Σ_3-3SAT 进行归约，并且归约产生的子集族 S 可以运用相同的倍数。

第 6 章

6.1 a. 利用等式 $f(x_1,\cdots,x_n)=(x_n\wedge f(x_1,\cdots,x_{n-1},1))\vee(\overline{x_n}\wedge f(x_1,\cdots,x_{n-1},0))$ 为 f 递归地构建一个 $O(2^n)$ 的线路。

6.1 b. k 个位上只有 2^{2^k} 个函数，这意味着我们可以简单地用 $2^{2^k}\cdot(k2^{2^k})$ 个门来计算 $x_1,\cdots,x_k$ 上所有可能的函数。这样做之后，我们可以在 $O(2^{n-k})$ 个门上使用习题 6.1a 中的递归线路，每个递归线路含 $n-k$ 个层。令 $k=\log n-2$ 即得到所需的结果。

6.5 留意具有较高线路复杂度的函数的存在性证明，并试着证明：你能够计算量同交错常数次并且在字典序中最靠前的具有较高线路复杂度的函数。

6.7 留意习题 6.5 的结论。

6.9 给出一个递归的多项式时间算法 S 使得 S 在含有 n 个变量的公式 φ 和 $v\in\{0,1\}^*$ 上输出 1 当且仅当 φ 存在满足性赋值 v 满足 $v>u(v,u$ 都视为$[2^n]$中的数的二进制表示)。利用 SAT 到语言 L 的归约削减 S 的递归树中的概率。

6.12 a. 你可以用不同的处理器来计算 AB 中的不同元素。

6.12 b. 通过重复地取平方，$A^{2^k}=(A^{2^{k-1}})^2$。

6.12 c. 如果 A 是某个图的邻接矩阵，那么 A^n 中第 i 行第 j 列的元素表示什么含义？

6.13 线路忽略输入结点之后可以视为一个有向的二叉树。在规模为 m 的二叉树中，一定存在一个结点使得删除该结点之后得到的每棵子树都至多具有规模 $2m/3$。

6.16 先为矩阵乘法设计 **NC** 线路，然后用快速乘幂法递归 $\mathrm{poly}(\log n+\log r)$ 次来计算 A^r。然后，利用如下事实：矩阵的行列式等于其所有特征值的乘积，并且 $\mathrm{trace}(A^r)$ 等于 A 的所有特征值的 r 次方之和。而且，行列式和迹都是特征值的对称函数。

6.19 在你的归约中，将 CIRCUIT-EVAL 问题表示成线性规划，并利用 $x \vee y = 1$ 当且仅当 $x + y \geqslant 1$ 这一事实。注意，线性规划中的变量取实数值而不是布尔值。 [533]

第 7 章

7.3 利用 n 的二进制表示并重复地计算平方。

7.4 利用如下事实，如果 B_1，…，B_k 是 k 个独立的随机事件且每个事件发生的概率至多为 p，则事件 $B_1 \cup \cdots \cup B_k$ 的概率至多为 p^k。

7.5 将实数 p 视为建言串。如何得到它的每个位？

7.8 沿用证明卡普-利普顿定理(定理 6.19)的思想。

7.9 利用动态规划或矩阵乘法，试着计算机器最后终止于接受格局的概率。

7.11 c. 考虑始于 u 的无限步随机游走。如果 $E_v > K$，则由标准概率界(如，切尔诺夫界)可知，u 在该游走中出现的位置占游走的所有位置的比例小于 $2/K$。

7.11 d. 从 $k=1$ 的情况开始(此时，u，v 通过一条边连通)，$k>1$ 时的情形可以用数学期望的线性性质归约为 $k=1$ 时的情况。注意，$\mathbf{N}$ 上的随机变量 X 的数学期望等于 $\sum_{m \in \mathbf{N}} \Pr[X \geqslant m]$。因此，只需证明始于 u 的 ln^3 步随机游走未到达过 v 的概率随着 l 的增长而指数地下降。

第 8 章

8.1 c. 利用 **IP**=**PSPACE**。

8.5 首先注意到，在当前的集合下界协议中，我们让证明者选择哈希函数。考虑如下相对简单的任务：构造一个交互式证明协议来区分 $|S| > K$ 和 $|S| < \frac{1}{c}K$，其中 $c > 2$(它甚至可以是 K 的函数)。如果 c 充分大，则我们可以让证明者使用几个哈希函数 h_1，…，h_i。并且，可以证明，如果 i 充分大，则 $\bigcup_i h_i(S) = \{0,1\}^n$。如果用 S^l 代替 S(其中 S^l 是 S 的 l 次笛卡尔乘积)，则鸿沟还可以进一步放大。

8.7 先证明 **MAM**⊆**AM**，其中 **MAM** 是可用如下的 3 回合交互式证明系统证明的所有语言构成的集合：证明者先发送一个消息，而后验证者发送一些随机二进制位，然后证明再发送一个消息。我们可以将 **MAM** 协议变换为一个 **AM** 协议，这只需让验证者先发送他的随位串。这无损于证明系统的完备性。证明：如果我们先通过平行重复将可靠性错误率减小到一个充分小的值(使得该值是证明者消息长度的函数)，则变换后的交互式协议将是可靠的。

8.8 a. 证明：在这种情况下，由于乘积运算的存在，所构造的每个多项式的次数至多扩大为原来的 2 倍。

8.8 b. 如果 Ψ 还不是这种形式而是含有形如 $\forall x_j \cdots \forall x_{j'} p(x_i, \cdots)$ 的片段，其中 $j' > j > i$ 且 p 是含有变量 x_i 的公式(可能还包含其他变量)，则我们可以引入一个新变量 y_i 并将公式转换为它的等价形式 $\forall x_j \exists y_i$ 使得 $(y_i = x_j) AND \cdots \forall x_{j'} p(y_i, \cdots)$，从右向左递归地使用上述过程。

8.12 证明用两个验证者可以模拟 $\mathrm{poly}(n)$ 个验证者。模拟过程中，用一个证明者扮演 $m(n)$ 个证明者，另一个证明者只模拟从 $m(n)$ 个证明者中随机选出的一个证明者。然后，将这种模拟过程重复几次。 [534]

第 9 章

9.2 形如 $E_{0^n}(x)$ 的所有分布都具有相同的支持度(support)吗？

9.4 在 $\{0,1\}^{n^{10}}$ 上定义如下的分布 $\mathcal{D}$：从 $E_{0^n}(0^{n^{10}})$ 中随机选择 y，从 $\{0,1\}^n$ 中随机选取 k，然后令 $x = \Pi_k(y)$。给出一个函数 A 使得：如果我们置 $x_0 = 0^{n^{10}}$ 且 x_1 使得(9.11)式不成立，则在任意 $x \in \{0,1\}^{n^{10}}$ 上均有 $\Pr[\mathcal{D} = x] > 2^{-n}$。由此导出矛盾。

9.6 a. 利用填充技术。

9.7 证明：如果 $X^2 = Y^2 \pmod M$ 但 $X \neq \pm Y \pmod M$，则通过计算 M 和 $X - Y$ 之间的最大公因数(gcd)可以找出 M 的一个因数。然后，证明你可以用一个逆算法找出这样一对 X，Y。

9.8 对任意素数 p，$\mathbf{Z}_p^*$ 的生成元素 g 和 $x \in \{0, \cdots, p-1\}$，如果我们随机选择 $y \in_R \{0, \cdots, p-1\}$，

则 $g^r g^s \pmod p$ 服从 $\mathbf{Z}_p^*$ 上的均匀分布。

9.9 b. 将 $A(E_{U_n}(0^m))$ 作为算法 B。

9.9 c. 用前一小题中同样的算法 B。

9.10 利用定理 9.13 的证明思想。

9.13 你需要证明某个行列式不等于 0。

9.16 先证明，将语言 3COL 替换为语言 $L=\{(y, r, b): \exists x$ 使得 $y=f(x), b=r\odot x\}$ 时结论成立，其中 f 是单向函数。

第 10 章

10.2 先证明条件 3 成立当且仅当条件 1 成立当且仅当条件 4 成立。上述推导几乎立刻可以由定义和下面的事实得出：对任意矩阵 A，B 均有 $(AB)^*=B^*A^*$ 和 $(A^*)^*=A$ 成立。然后，证明条件 3 蕴含条件 2，这可以利用在基变换下范数不变的事实。最后，证明条件 2 蕴含条件 3，这可以通过证明下面的结论来完成：如果两个正交的单位向量 $\boldsymbol{v}$，$\boldsymbol{u}$ 被映射为非正交的单位向量，则 $\boldsymbol{u}+\boldsymbol{v}$ 的范数在该映射下不会保持不变。

10.5 在寄存器中再添加一个量子位，它表示如下的语义。如果这个量子位是 0，则其余量子位的振幅对应于原始算法中所有振幅的实部；如果这个量子位是 1，则其余量子位的振幅对应于原始算法中所有振幅的虚部。

10.10 先考虑 $x=2^k$ 对某个 k 成立的情形。然后，利用 x 的二进制表示形式给出处理一般 x 的算法。

10.12 利用如下事实：如果 N 和 A 互素，则存在整数 α，β 使得 $\alpha N+\beta A=1$。等式的两端同时乘以 B 即可得出结论。

10.15 令 $d=\gcd(r, M)$，$r'=r/d$，且 $M'=M/d$。现在，利用同 r 与 M 互素的情况时一样的论证过程，证明存在 $\Omega\left(\frac{r}{d\log r}\right)$ 个 $x\in\mathbf{Z}_M$ 满足要求，因此 $x+cM$（其中 c 是任意的）也满足要求。

535

第 11 章

11.3 证明：随机赋值所满足的子句占子句总数的比例的数学期望为 7/8。然后用马尔可夫不等式证明 $7/8-1/2m$（其中 m 是子句的个数）比例的子句被满足的概率至少是 $1/\mathrm{poly}(m)$。

11.4 利用条件数学的方法。给定变量 u_1，…，u_i 上的部分赋值，可以在多项式时间内计算出 u_{i+1}，…，u_n 的随机赋值所满足的子句的个数的数学期望。存在对 u_1，u_2，…依次赋值的一种方法使得下面的不变量总成立：被满足的子句的个数占子句总数的比例的数学期望至少为 7/8。（得到确定型算法的另一种方法是利用任意 3 个变量都相互独立的抽样空间来选取赋值，参见习题 11.14 的提示。）

11.8 利用题目所给条件推导出 SAT 的向下自归约性。

11.9 为 3SAT 设计一个验证者。一种平凡的思想是，3SAT 公式的证明由该公式的满足性赋值构成，验证者可以随机选择公式中的一个子句，然后从证明中读取该子句对应的三个变量的赋值，并验证所选的子句是否被读取到的 3 个位满足。这种思想不符合要求，为什么？更好的思想是，让 3SAT 公式的证明包含满足性赋值的多个复制，并且让验证者利用两两独立性在这些复制上执行上述验证过程，验证过程使用的复制可以是同一个也可以是不同的复制。

11.11 库克-勒维归约实际上将任意 $x\in\{0, 1\}^*$ 都变换为一个几乎所有子句都能同时满足的布尔公式。这是由于，所得的布尔公式几乎检查了 M 在任意输入 x 和任意串 u（即使 $M(x, u)=0$ 亦如此）上执行运算时各个状态转移之间的一致性。

11.12 先证明，如果问题实例中的所有数都取自 $[m]$，则该问题可以用动态规划算法在 $\mathrm{poly}(n, m)$ 时间内精确求解。然后，说明如何保留每个数的前 $O(\log(1/\varepsilon)+\log n)$ 个二进制位以便得到问题的一个近似算法。

11.14 同习题 11.4 一样，要将随机算法去随机化，既可以用条件数学期望，也可以用任意 q 个函数均相互独立的一族函数。这族函数可以借助 8.2.2 节中两两独立哈希函数族的构造方法来获得，只是需要将当时所采用的线性函数替换为 $GF(2^n)$ 上的 $q-1$ 次多项式。

11.15　说明 SAT 的满足性问题可以表达成二次方程组。

11.16　从 MAX-3SAT 进行归约。

第 12 章

12.1　令 x_1，…，x_n 满足 $f(x_i)\neq f(x_j)$。证明：存在 x_1，…，x_n 中由至少 $n/2^k$ 个元素构成的子集合 X 使得判定树的前 k 次查验在 X 中所有元素上将得到相同的答案。

12.2　用数学归纳法。

第 13 章

13.3　证明：不存在求解 PAL 的单带图灵机 M 使得：在形如 $x_{n/2}\cdots x_1 0^n x_1\cdots x_{n/2}$ 的任意输入和任意位置编号 $i\in[n/2+1,\cdots,3n/2-1]$ 上，M 扫描工作带上介于 i 和 $i+1$ 之间的位置的次数少于 $o(n)$ 次。否则的话，当 M 的读写头在工作带上的前 i 个位置上移动时，让爱丽丝模拟 M 的运行。当 M 的读写头在工作带上的其余位置上移动时，让波比模拟 M 的运行。这样就可以为相等函数设计出一个通信复杂度协议使得它在多于 $2^{n/2}/n$ 个输入上仅通信 $o(n)$ 个位。 535

13.4　同前一题一样，将问题转化为通信复杂度协议的设计问题。这里，爱丽丝和波比之间的通信传递工作带上的内容。（但是，在本题中输入带是只读的。）创建一个只存储 0 的“缓冲区”，迫使机器用 n 步骤才能处理爱丽丝和波比之间的一次通信。

13.5　对单色铺砌所用的所有矩形随意编号，令 $N=\chi(f)$。在集合 $\{1,\cdots,N\}$ 上定义图 G_R，G_C。其中 (i,j) 是 G_R 的边当且仅当矩形 i 和矩形 j 共享一个公共的行。相应地，(i,j) 是 G_C 的边当且仅当矩形 i 和矩形 j 共享一个公共的列。令 $\deg_R(\cdot)$ 和 $\deg_C(\cdot)$ 分别是这两个图的度函数。在每个步骤中，行参与方（Row Player）试图找到一个包含其输入且 $\deg_R(i)\leqslant 3|G_R|/4$ 的矩形 i。如果这样的矩形存在，则将 i 发送给列参与方（Column Player）。类似地，列参与方试图找到一个包含其输入且 $\deg_C(j)\leqslant 3|G_C|/4$ 的矩形 j。我们断言，如果这样的 i，j 确实可以找到，则问题就解决了。你能说明这是为什么吗？而且，你能证明这样的 i，j 总存在吗？注意，下面的事实或许有所帮助：如果 N 顶点图的最小度至少是 $N/2+1$，则任意两个顶点都会有公共的相邻顶点。

13.6　首先证明 $\mathrm{rank}(A+B)\leqslant\mathrm{rank}(A)+\mathrm{rank}(B)$ 对任意矩阵 A，B 成立。这意味着，如果 $A=\sum_{i=1}^{l}a_iB_i$ 对秩为 1 的矩阵 B_1，…，B_l 成立，则 $\mathrm{rank}(A)\leqslant l$。然后，利用“如果 A 的秩不超过 l，则 A 中存在 l 个行使得 A 的其余行都可以表示为这 l 个行的线性组合”这一事实将 A 表示为 l 个秩为 1 的矩阵 B_1，…，B_l 之和（矩阵 B_i 的所有行都是 A 的某个行的倍数）。

13.9　利用 $M'=J-2M$（其中 J 是元素全为 1 的矩阵）这一事实。

13.10　将问题转换为 ±1 矩阵的秩的计算问题。然后在实数域上计算矩阵的秩。直接在 GF(2) 上计算秩，你能证明同样的结论吗？

13.11　给出秩的下界。

13.12　利用 $-1^{abk}-1^{a'bk}-1^{ab'k}-1^{a'b'k}=-1^{(a+a')(b+b')k}$ 这一事实。

13.15　利用 7.2.3 节中的指纹技术。

13.16　要将线路转换成一个通信协议，将通信协议的参与双方不妨记为 OR 方和 AND 方，其中 OR 方持有的输入 x 满足 $f(x)=0$ 而 AND 方持有的输入 y 满足 $f(y)=1$。双方都知道他们的输入至少有一个位是不同的，他们需要借助线路来找出这个位。他们都在线路上计算各自输入的函数值。如果线路的顶端的门是 OR 门，则 OR 方知道这个 OR 门的两个子门都输出 0；而 AND 方则知道这个 OR 门至少有一个子门输出 1。因此，AND 方发送一个位给 OR 方告诉他哪个门输出了 1。双方按照上述方式深入线路对应的二叉树中。要将通信协议转换成线路，你可以类似地进行处理并使用数学归纳法。

13.19　将问题归约为不相交性的通信复杂度协议，其中爱丽丝读取输入中的（比方说）前 $n/4$ 个数据而波比读取其余输入数据。

第 14 章

14.1　原来线路中的每个门在新线路中都有一个孪生门，该孪生门计算原始线路门的否定。 537

14.3 先将 f 平凡地表示为一个 CNF 公式使得每个满足 $f(x)=0$ 的赋值 x 在该表示中对应一个子句。然后证明每个子句 C 都可以替换为如下的子句 D，D 中至多包含 C 中的 s 个文字，但仍然可以保证"若 $f(x)=0$，则 $D(x)=0$"对某个新得出的子句 D 成立。

14.4 利用等式 $\binom{n}{t+k}=\binom{n}{t}\binom{n-t}{k}/\binom{t+k}{t}$ 和估计式 $\left(\frac{n}{k}\right)^k\leqslant\binom{n}{k}\leqslant\left(\frac{en}{k}\right)^k$。

14.10 证明：如果 $I\subseteq[l]$，$x_1<x_2<\cdots<x_m$ 是取自 $[2^l-1]$ 的一个单调数列并且 x_{i-1} 和 x_i 取值相异的最高位所在的位置不属于 I，则 $x'_1<x'_2<\cdots<x'_m$ 仍然是一个单调数列，其中 x'_i 是将 x_i 中属于 I 的位置全部置 0 之后得到的数，由此得出结论 $m\leqslant 2^{l-|I|}$。

第 15 章

15.1 尝试模仿为 φ 寻找满足性赋值的指数时间算法。

15.2 对任意 j，在"消除"z 变量之后的辩驳中，如果第 j 个子句 $\overline{C}_j$ 仅用 y 变量就可以推导出来，则令 $d_z(e)=0$。证明：(1)每个 $d_z(e)$ 都可以用 z 上的规模为 $O(S^2)$ 的线路来计算；(2)在定理 15.4 的证明中可以取 $I(e)=d_z(e)$。

15.2 通过赋值操作 $z'_i=\neg z_i$ 和函数 $l'=\neg l$ 将本小题归约为上一小题讨论的情况。

15.4 难点在于完备性。一种相对简单的特殊情况是公理系统中含有 $0\leqslant X_i\leqslant 1$ 时的情况。此时，尝试证明 $D=2$ 时推导规则是完备的。证明上述情况之后再证明，在一般情况下，所有的归结证明都可以改写为一组平面分割证明使得其中的每个证明都属于 $D=2$ 时的特殊情况。

15.6 任意 $i\leqslant n+1$，$j\leqslant n$ 都对应一个变量 x_{ij}，它等于 1 当且仅当 i 被映射到 j。

第 16 章

16.4 a. 先在 n 是 k 的方幂的情况下证明结论的正确性。如果 $n=k^l$，你可以将 $k^l\times k^l$ 的矩阵划分为 k^2 个块，每个块的规模为 $k^{l-1}\times k^{l-1}$。先递归地调用分块矩阵的乘法，然后程序 $\prod_k$ 再将递归调用的结果合并起来得到最后的结果。

16.4 b. 我们无法提供找出这种程序的直觉思想。但是，由于我们只讨论 2×2 的矩阵，所以可以通过尝试和排除错误方案来找到它。

16.6 c. 注意，矩阵的行列式可以用其子式的行列式来表示，首先利用上述事实证明 $p(x)=(A_{1,1}-x)\det(M-xI)+\varepsilon ADJ(M-xI)c$，其中对于任意矩阵 B 而言，$ADJ(B)$ 的 (i,j) 位置上的元素等于 $(-1)^{i+j}$ 乘以删除 B 中第 i 行和第 j 列后所得子式的行列式(也就是说，对于任意非奇异矩阵 B，均有 $ADJ(B)=\det(B)B^{-1}$)。然后，利用凯莱-哈密顿定理(Caley-Hamilton Theorem)(该定理断言 $q_M(M)$ 等于零矩阵)将矩阵多项式 $ADJ(M-xI)$ 的系数用 q_M 的系数和 M 的方幂表达出来。

16.8 参见习题 16.9。

16.10 先证明只需计算 $k!$ 其中 k 是 n 的最小非平凡因数，而这又只需计算 $s!$ 其中 s 是 2 的最小方幂且 $s>k$。然后，注意到 $\binom{2r}{r}=\frac{(2r)!}{(r!)^2}$，因此只需对任意大的 r 计算 $\binom{2r}{r}$。但 $\binom{2r}{r}$ 仅仅是 $(x^2+1)^{2r}$ 中的一个项。x 取多大时，才能恰当地运用模操作从 $(x^2+1)^{2r}$ 中"读出" $\binom{2r}{r}$ 呢？ 538

16.12 这种机器的"程序"可以将任意精度的实数作为常数来使用。

第 17 章

17.5 当你需要估计某个字符串集合的大小时，请利用哈希和证明户田定理时所用的思想。如果你觉得本习题很难，你可能需要回头再看看第 8 章中的戈德瓦瑟-西普赛尔集合下界协议。要使得算法是确定型的，还需要利用证明 **BPP**$\subseteq$**PH**(定理 7.15)时所用的思想。

17.6 利用引理 17.17 的证明过程。

17.7 实数可以用有理数来近似，因此只需证明结论在"表示量子操作的矩阵是有理数矩阵"的情况下成立即可。

第 18 章

18.1　3-可着色图不可能含有 4 个顶点上的完全子图。

18.2　随机图中存在大小至少为 k 的独立集的概率至多为 $\binom{n}{k}2^{-\binom{k}{2}}$。

18.5　构造 CNF 公式上的一个可抽样的分布 $\mathcal{D}$ 使得根据 φ 在 $\mathcal{D}$ 中的概率可以计算得到公式 φ 的满足性赋值的个数。

18.6　利用如下事实：在任意非负随机变量 X 和 $d\geqslant 1$ 上，均有 $E[X^d]\geqslant E[X]^d$。

第 19 章

19.1　定义 $Y_i=(-1)X_i$ 且 $Y=\prod_{i=1}^{k}Y_i$。然后，利用独立随机变量乘积的数学期望等于它们的数学期望的乘积这一事实。

19.2　选取 $x\in\{0,1\}^n$ 使得它属于 I 的概率为 $\delta 2^n\Pr[H=x]$。证明：(1)$\Pr[|I|\geqslant\frac{\delta}{2}2^n]>1/2$；(2)对于任意线路 C，如果我们定义 $SUCCESS_C(I)$ 是随机选取 $x\in I$ 时 $C(x)=f(x)$ 成立的概率，则相对于 I 的随机选取而言 $SUCCESS_C(I)\geqslant 1/2+2\varepsilon$ 的概率小于 $\frac{1}{2}2^{-n}$。

19.3　b. 将 G，H，U 视为 2^n-维的概率向量或许会有所帮助。

19.5　取 z 是形如 $x-y$（其中 $x\in C$ 而 $y\in D$）的最短向量（可以证明 z 是存在的，而且是非零的。这只需利用 C，D 是闭集而且 D 是紧集的事实，因为这些条件意味着我们只需考虑 C 与充分大的球体的交集）。

19.6　注意，$\max_q\min_p qAp>c$ 当且仅当凸集 $D=\{Ap:\ p\in\{0,1\}^n\text{且}\sum_i p_i=1\}$ 与凸集 $C=\{x\in\mathbf{R}^m:\ \forall_{i\in[m]}x_i\leqslant c\}$ 不相交。用超平面分离定理证明，上述结论意味着存在概率向量 q 使得 $\langle q,y\rangle\geqslant c$ 对任意 $y\in D$ 成立。

19.7　假设存在某个概率密度为 2^{-n} 的分布 D 不是这样的凸组合，则可以用超平面分离定理如下地导出矛盾：根据 D 与标准超平面之间的内积的大小，将分布 D 中的各个项重新排列，不断修改权值直到得到一个扁平分布。

539

19.9　使用贪心策略，逐次地选择 E 的码字，确保每次选择的码字与之前选择的所有码字之间的距离都不会小于 δ。选择过程何时终止？

19.10　沿用约翰逊界限（定理 19.23）的证明过程，但需要将问题重述为如下形式：在保证所选的单位向量相互之间都比较远的条件下，你最多能从 $\mathbf{R}^n$ 中能够取出多少个单位向量？

19.14　参见定理叙述之前的讨论和定理 19.21 的证明。

19.15　第一个多项式在 ε 比例的点（不妨记为集合 S_1）上刻画函数 f，第二个多项式在 $\varepsilon-d/|\mathbf{F}|$ 比例的点（不妨记为集合 S_2）上刻画函数 f，其中 $S_1\cap S_2=\emptyset$。以此类推。

19.16　将 $Q(x,y)$ 视为变量 y 上的一元多项式，其系数是 x 的多项式（即，系数是环 $\mathbf{F}[x]$ 中的元素）。然后，用 $y-P(x)$ 去除 $Q(x,y)$ 得到 $Q(x,y)=(y-P(x))A(x,y)+R(x,y)$，其中 $R(x,y)$ 是余式并且 $R(x,y)$ 中 y 的次数小于 $y-P(x)$ 中 y 的次数。

19.17　b. 用概率方法。证明结论对随机矩阵成立。

19.17　c. 拼接 $GF(2^n)$ 上的里德-所罗门纠错码和沃尔什-哈达玛纠错码。

19.18　c. 将里德-所罗门纠错码和 19.18b 中得到的二进制编码进行拼接。注意，我们只将二进制编码运用到长度为 $O(\log n)$ 的输入上，这样我们就可以使用指数时间的编码算法和解码算法。

第 20 章

20.2　证明：在任意 n 上，将长度为 n 的位串映射为长度为 2^{n+1} 的位串的随机函数就可以高概率地满足习题要求的性质。

20.4　如果 G 是一个伪随机数产生器，则仅需考虑如下的函数 f：在输入 $x\in\{0,1\}^{n+1}$ 上，$f(x)=1$ 当

且仅当存在 $z\in(0,1)^r$ 使得 $G(z)=x$。

20.6 利用定理 20.6。

20.8 证明：定理 20.6 和定理 19.27 的证明过程蕴含了，给定函数 $f\in\mathbf{EXP}$，如果它的平均复杂度 $H_{\alpha\alpha}(f)$ 不是多项式有界的，则我们可以得到一个 $S(l)$-伪随机数产生器使得 S 也不是多项式有界的(亦即，对任意多项式 p，$S(l)>p(l)$ 必然在无穷多个 l 上成立)。

20.9 利用如下事实：算法 D 高概率地计算出一个能判定语言 L 的线路。

第 21 章

21.2 a. 利用如下事实，log 函数是凸函数(二阶导数非负)意味着 $a\log a+(1-a)\log b\leqslant\log(a\alpha+(1-a)b)$。

21.2 c. 表达式 $|\mathbf{v}|_1^2=\sum_{i,j}|\mathbf{v}_i||\mathbf{v}_j|$ 包含了 $\|\mathbf{v}\|_2^2$ 中出现的所有项和另外一些非负项。

21.4 证明如果在任意两个顶点之间的最短路径上再任意选择一个顶点，则这三个顶点的 $(d+1)$-邻域是互不相交的。

21.5 先证明 $\|A\|$ 不超过 n^2。然后利用等式 $\langle\mathbf{w},B\mathbf{z}\rangle=\langle B^{\mathrm{T}}\mathbf{w},\mathbf{z}\rangle$ 和不等式 $\langle\mathbf{w},\mathbf{z}\rangle\leqslant\|\mathbf{w}\|_2\|\mathbf{z}\|_2$ 来证明：对任意 $k\geqslant1$，A^k 也是随机矩阵并且 $\|A^{2k}\mathbf{v}\|_2\geqslant\|A^k\mathbf{v}\|$。

21.8 利用如下事实：如果 A 是一个图的随机游走矩阵且 $\mathbf{v}\perp\mathbf{1}$，则 $A\mathbf{v}\perp\mathbf{1}$。

21.10 d. 这样的路径可以如下获得：从树根出发远离树根前进 $k/2$ 步，再通过 $k/2$ 步返回树根。远离树根的每个步骤有 $d-1$ 种选择，因此得到因子 $2^{(k/2)\log(d-1)}$，返回树根时所做的选择另外产生一个约为 $\binom{k}{k/2}=2^{k-o(k)}$ 的因子。事实上，我们必须更加小心一些，因为如果我们已经在树根的位置上则不能再进行"返回树根"的动作。因此，我们必须确保在任何时候采取行动时绝不会产生多余的"返回树根"动作和"远离树根"动作。为此，我们可以这样做，让前 t 个步骤都"远离树根"，让最后的 t 个步骤都"返回树根"，其中 $t=100\log k\sqrt{k}$(它是 $o(k)$)。这样，我们将有 $\binom{k-2t}{k/2-2t}=2^{k-o(k)}$ 方法在路径中插入剩下的 $k/2-2t$ 步"返回树根"的动作。我们可以证明，这些方法中绝大多数都不会得到无效的路径。更具体地讲，我们可以看到，有效路径的条数恰好等于为 k 个数的乘法表达式添加括弧的方案个数。可以证明，这个数等于 $\frac{1}{k/2+1}\binom{k}{k/2}$，这个数也称为第 $k/2$ 个卡特兰数(Catalan Number)。

21.10 e. 利用前面几个小题的结论，证明 $1+(n-1)\lambda^k\geqslant n2^{k-k\log d/2-o(k)}$。对不等式两端同时取对数就得到需要的界限。

21.11 对满足 $|S|\leqslant n/2$ 的任意子集 $S\subseteq[n]$，试着给出介于 S，$\overline{S}$ 之间的边的条数偏离其数学期望超过某个阈值的概率上界。

21.13 用概率论证法。随机选取大小为 $n/2$ 的顶点子集 S。对任意两个相异的顶点 u，v 而言，"$u\in S$ 且 $v\in\overline{S}$"或"$v\in S$ 且 $u\in\overline{S}$"的概率至多为 1/2(如果 S 是可放回式地选取的，则该概率恰好等于1/2)。因此，由于图中共有 $dn/2$ 条边，因此 $E(S,\overline{S})$ 的数学期望至多为 $dn/4$。

21.14 你可以利用引理 21.14。

21.15 c. 证明：如果 $\mathbf{s}$ 是 S 上的均匀分布，则 $\|A\mathbf{s}\|_2^2\leqslant\|A\mathbf{1}\|_2^2+\lambda^2\|\mathbf{s}-\mathbf{1}\|_2^2$。

21.16 H 中大小至多为 $n'/2$ 的顶点子集 S 对应于 G 中大小至多为 $(1-1/(2c))n$ 的顶点子集 S'。利用 G 的扩张度来讨论介于 S' 的补集和 S 之间的边的条数。

21.18 b. 证明任意确定型算法必然需要查验指数次函数值。

21.18 c. 证明在给定的条件下存在一个大小至多为 2^{n-1} 的子集 S 使得 $\Pr[X\in S]\geqslant1/20$。

21.19 将定义在 M 元集合上的分布表示为 $\mathbf{R}^M$ 中的向量，然后利用 L_1-范数的三角不等式。

21.23 利用引理 21.14。

21.24 G 与 G' 的替换乘积的每个顶点子集，都可以看成 $G\circledS G'$ 的 n 个聚簇的子集(的并集)。在这些子集中，有些子集合包含了相应聚簇中 $1-\mu/10$ 比例以上的顶点。有些子集却只包含了相应聚簇中不

到 $1-\rho/10$ 比例的顶点。对这两类子集分别进行处理。在前者上利用 G 的扩张度，在后者上利用
541 G' 的扩张度。

第 22 章

22.1 在 $T=V\setminus S$ 上运用引理 21.10。

22.2 利用类似于定理 22.12 的证明过程所采用的技术来证明本习题。

22.3 利用斯特林公式(Stiring's Formula)来近似二项式系数中的阶乘。

22.4 考虑新定义的随机变量 V'，它等于 V 在 $V>0$ 的条件下的取值，并利用不等式 $\mathbf{E}[V']\geqslant\mathbf{E}[V]$。

22.5 e. 利用习题 22.5d 中的 T 测试，再结合线性测试、自纠错和一个简单的测试来排除恒零函数。

22.5 f. 要将 n 个变量上的 $2\mathrm{CSP}_q$ 公式 φ 变换为二进制字母表上的 $q\mathrm{CSP}$ 公式 Ψ，让 φ 中每个变量 u_i 对应 2^n 个变量 $u_i^1,\cdots,u_i^{2^n}$。在正确性证明中，这些变量的取值将对应 u_i 的值的长编码。然后，对 φ 的每个约束 φ_c，再添加 2^{n^2} 个变量 $y_c^1,\cdots,y_c^{2^{n^2}}$。在正确性证明中，这些变量的取值将对应 φ_c 中所有变量取值的长编码。对于 φ 中的每个约束，其中出现的变量记为 x 和 y，变换后的实例 φ 将用一系列约束来测试 x，y 的长编码，还要测试变量 x，y 的一致性，还要测试所有 x，y 确实构成了一个满足性赋值。

22.6 a. $\mathrm{val}(\varphi)=\mathrm{val}(\varphi*2)=1/2$。

22.12 a. 用傅里叶基将函数 f 表示出来，利用最基本的性质，以及 x，x' 和 y 之间的随机独立性。

22.12 b. 利用函数 $g(x,y)=f(x,y)\chi_x(x)$ 将问题转换为 22.12a 中的情形。

22.12 c. 你可以通过在随机选取的输入上计算相应函数的取值来估计(22.13)式给出的数学期望。

22.12 d. 考虑将深度为 n 的完全二叉树用长度$\leqslant n$ 的所有位串对每个结点进行标记(树根标记为空串，a 的两个孩子分别标记为 $a0$ 和 $a1$)。然后，用巴塞弗恒等式证明，在二叉树的每一层，至多存在 $1/\varepsilon^2$ 个位串 a 满足 $\hat{f}_{a,*}\geqslant\varepsilon$。利用 22.12c 得到的 Esitmate 过程从树根到叶子进行剪枝，扔掉满足 $\hat{f}_{a,*}<10\varepsilon$ 的所有 a 分支。最后，输出剩下的所有叶子。

22.13 证明算法随机选择一族子集即可。

22.14 需要构造 ε 偏斜的随机变量，虽然本书未涉及，但是用线性纠错码仍然可以构造出这种随机变量。

22.15 设法将习题 11.16 得到的鸿沟"放大"常数因子倍。你需要将一些方程进行重组来构造新的方程。

22.16 对于出现在 5 个以上的子句中的每个变量，引入一组新的变量。用这些新变量设计一个构件来确保，这些新变量在最优赋值中取相同的值。你可能需要借助扩张图。本习题本质上等同于论
542 断 22.37。

第 23 章

23.3 如果 DISCRETE LOG 对某个 p 而言在某些最坏的输入上是难解的，则它对这个 p 而言在绝大多数输入上也是难解的。因此，可以用它来构造出伪随机函数(假设将 p 用做一致的建言)。

23.4 为证明结论成立，只需证明结论对随机函数成立。对变量的个数用数学归纳法，并利用 f_* 和 $\overline{f}_*$ 都是随机函数这一事实。

543 23.5 参见习题 8.8。

参考文献

[Aar03] S. Aaronson. Is P versus NP formally independent? *Bulletin of the EATCS*, 81:109–136, 2003.

[Aar05] S. Aaronson. NP-complete problems and physical reality. *SIGACT News*, 36, 2005.

[Aar06] S. Aaronson. Oracles Are Subtle But Not Malicious. *Proceedings of the 21st Annual IEEE Conference on Computational Complexity*, pages 340–354, 2006.

[Aar08] S. Aaronson. The limits of quantum computers. *Scientific American*, pages 62–69, Mar. 2008.

[AB87] N. Alon and R. B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[AB97] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error rate. *SIAM J. Comput.*, 38(4):1207–1282, 2008. Prelim version STOC '97.

[ABSS93] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997.

[AC86] N. Alon and F. R. K. Chung. Explicit construction of linear sized tolerant networks. *Discrete Mathematics*, 72:15–19, 1988. Prelim version Japan Conf on Graph Theory and Applications '86.

[ACR96] A. E. Andreev, A. E. F. Clementi, and J. D. P. Rolim. A new general derandomization method. *J. ACM*, 45(1):179–213, 1998. Prelim version ICALP '96.

[ACR+07] A. Ambainis, A. M. Childs, B. Reichardt, R. Spalek, and S. Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1}/2 + o(1)$ on a quantum computer. In *FOCS*, pages 363–372. IEEE, 2007.

[AD97] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293. ACM, 1997.

[adH88] F. M. auf der Heide. Fast algorithms for N-dimensional restrictions of hard problems. *J. ACM*, 35(3):740–747, 1988.

[ADH97] L. M. Adleman, J. Demarrais, and M.-D. A. Huang. Quantum computability. *SIAM J. Comput.*, 26(5):1524–1540, 1997.

STOC—ACM Symposium on Theory of Computing; FOCS—IEEE Annual Symposium on Foundations of Computer Science.

[Adl78] L. Adleman. Two theorems on random polynomial time. In *FOCS*, pages 75–83. IEEE, 1978.

[AFWZ95] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995.

[AG94] E. Allender and V. Gore. A uniform circuit lower bound for the permanent. *SIAM J. Comput.*, 23(5):1026–1049, 1994.

[AGIK07] D. Aharonov, D. Gottesman, S. Irani, and J. Kempe. The power of quantum systems on a line. In *FOCS*, pages 373–383. IEEE, 2007.

[AIK04] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. Prelim version FOCS '04.

[AIV93] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: The role of local checkability. Unpublished manuscript, available from the authors' Web pages, 1993.

[Ajt83] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[Ajt88] M. Ajtai. The complexity of the pigeonhole principle. In *FOCS*, pages 346–355. IEEE, 1988.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108. ACM, 1996.

[AKL+79] R. Aleliunas, R. M. Karp, L. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *FOCS*, pages 218–223. IEEE, 29–31 Oct. 1979.

[AKS87] M. Ajtai, J. Komlos, and E. Szemeredi. Deterministic simulation in LOGSPACE. In *STOC*, pages 132–140. ACM, 1987.

[AKS98] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. *Random Struct. Algorithms*, 13(3–4):457–466, 1998. Prelim version SODA '98.

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.

[AL95] S. Arora and C. Lund. Hardness of approximations. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, Chapter 10. PWS, 1995.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. Prelim version FOCS '92.

[Alo86] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[AM84] N. Alon and V. D. Milman. Eigenvalues, expanders and superconcentrators (extended abstract). In *FOCS*, pages 320–322. IEEE, 24–26 Oct. 1984.

[AM85] N. Alon and V. D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *J. Comb. Theory Series B*, 38:73–88, 1985.

[Amb04] A. Ambainis. Quantum search algorithms. *SIGACT News*, 35(2):22–35, 2004.

[AMS96] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. Prelim version STOC '96.

[AN04] N. Alon and A. Naor. Approximating the cut-norm via grothendieck's inequality. *SIAM J. Comput.*, 35(4):787–803, 2006. Prelim version in STOC '04.

[And85] A. E. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Math. Dokl.*, 31(3):530–534, 1985.

[AR04] D. Aharonov and O. Regev. Lattice problems in NP intersect coNP. *J. ACM*, 52:749–765, 2005. Prelim version FOCS '04.

[Aro94] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD thesis, UC Berkeley, 1994.

[Aro96] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45:753–782 1998. Prelim version FOCS '96.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, Jan. 1998. Prelim version FOCS '92.

[AS00a] N. Alon and B. Sudakov. Bipartite subgraphs and the smallest eigenvalue. *Combinatorics, Probability & Computing*, 9(1):1–12, 2000.

[AS00b] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley, 2000.

[ATSWZ97] R. Armoni, A. Ta-Shma, A. Widgerson, and S. Zhou. An $O(\log(n)^{4/3})$ space algorithm for (s,t) connectivity in undirected graphs. *J. ACM*, 47(2):294–311, Mar. 2000. Prelim version STOC '97.

[AUY83] A. V. Aho, J. D. Ullman, and M. Yannakakis. On notions of information transfer in VLSI circuits. In *STOC*, pages 133–139. ACM, 1983.

[AvDK+04] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM J. Comput.*, 37(1):166–194, 2007. Prelim version FOCS '04.

[AW08] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. In *STOC*, pages 731–740. ACM, 2008.

[Bab85] L. Babai. Trading group theory for randomness. In *STOC*, pages 421–429. ACM, 1985.

[Bab90] L. Babai. E-mail and the unexpected power of interaction. In *Proceedings, Fifth Annual Structure in Complexity Theory Conference*, pages 31–91. IEEE, 8–11 July 1990.

[Bab94] L. Babai. Transparent proofs and limits to approximations. In *First European Congress of Mathematicians*, 1994.

[Bar86] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, Feb. 1989. Prelim version STOC '86.

[Bar02] B. Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *RANDOM*, volume 2483 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2002.

[BB84] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, 175, 1984.

[BBBV97] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, Oct. 1997.

[BBC+98] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. Prelim version FOCS '98.

[BBR88] C. H. Bennett, G. Brassard, and J. Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, Apr. 1988.

[BBR92] D. A. M. Barrington, R. Beigel, and R. Rudich. Representing Boolean functions as polynomials modulo composite numbers. *Computational Complexity*, 4(4):367–382, 1994. Prelim version STOC '92.

[BC06] M. Braverman and S. Cook. Computing over the reals: Foundations for scientific computing. *Notices of the AMS*, 53(3):318–329, 2006.

[BCC86] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, Oct. 1988. Prelim versions by Brassard and Crépeau (CRYPTO '86, FOCS '86) and Chaum (CRYPTO '86).

[BCE+95] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of *NP* search problems. In *STOC*, pages 303–314. ACM, 1995.

[BCS97] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*. Springer Verlag, 1997.

[BCSS97] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer Verlag, 1997.

[BDCGL89] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, Apr. 1992. Prelim version Structures in Complexity '89.

[BdW02] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288:21–43, 2002.

[BE76] B. Bollobás and P. Erdös. Cliques in random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 80(41):419–427, 1976.

[Bec91] J. Beck. An algorithmic approach to the lovàsz local lemma. *Random Structures and Algorithms*, 2(4):367–378, 1991.

[Bel64] J. S. Bell. On the Einstein-Podolsky-Rosen paradox. *Physics*, 1(3):195–290, 1964.

[Ben87] C. H. Bennett. Demons, engines and the second law. *Scientific American*, 257(5):88–96, 1987.

[Berch] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett.*, 18(3):147–150, 1984.

[BF90] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *lncs*, pages 37–48. Springer, 22–24 Feb. 1990.

[BFL90] L. Babai, L. Fortnow, and L. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. Prelim version FOCS '90.

[BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–32. ACM, 1991.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BFT98] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity (CCC-98)*, pages 8–12. IEEE, 15–18 June 1998.

[BG94] M. Bellare and S. Goldwasser. The complexity of decision versus search. *SIAM J. Comput.*, 23(1):97–119, 1994.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the $\mathcal{P} =? \mathcal{NP}$ question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BGS95] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[BHZ87] R. B. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987.

[BK95] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *J. ACM*, 14(2):332–336, 1967.

[Blu84] M. Blum. Independent unbiased coin flips from a correlated biased source: A finite state Markov chain. In *FOCS*, pages 425–433. IEEE, 24–26 Oct. 1984.

[Blu87] M. Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

[BLY92] A. Björner, L. Lovász, and A. C. C. Yao. Linear decision trees: Volume estimates and topological bounds. In *STOC*, pages 170–177. ACM, 1992.

[BM82] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, Nov. 1984. Prelim version FOCS '82.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

[BMMS00] J. C. Birget, S. Margolis, J. Meakin, and M. Sapir. *Algorithmic Problems in Groups and Semigroups*. Birkhauser, 2000.

[BNS89] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992. Prelim version STOC '89.

[BO83] M. Ben-Or. Lower bounds for algebraic computation trees. In *STOC*, pages 80–86. ACM, 1983.

[BOGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *STOC*, pages 113–131. ACM, 2–4 May 1988.

[Bol01] B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.

[Bou02] J. Bourgain. On the distribution of the fourier spectrum of boolean functions. *Israel J. Math.*, 131(1):269–276, 2002.

[BP73] L. A. Bassalygo and M. S. Pinsker. The complexity of an optimal non-blocking commutation scheme without rezzzorganization. *Problemy Peredaci Informacii*, 9(1):84–87, 1973. Translated into English in *Problems of Information Transmission*, 9:64–66, 1974.

[BPR97] M. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *J. Symbolic Logic*, 62(3):708–728, 1997.

[Bra04] G. Brassard. Quantum communication complexity: A survey. In *ISMVL*, page 56. IEEE, 2004.

[Bru04] C. Bruce. *Schrodinger's Rabbits: Entering The Many Worlds Of Quantum*. Joseph Henry Press, 2004.

[BS90] R. B. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 1. Elsevier and MIT Press, 1990.

[BS94] M. Bellare and M. Sudan. Improved non-approximability results. In *STOC*, pages 184–193. ACM, 1994.

[BS96] E. Bach and J. Shallit. *Algorithmic Number Theory – Efficient Algorithms*, volume I. MIT Press, 1996.

[BS08] D. Boneh and V. Shoup. *A graduate course in applied cryptography*. 2008. To appear. Prelim drafts available on http://crypto.stanford.edu/ dabo/cryptobook/.

[BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *American Mathematical Society*, 21(1):1–46, 1989.

[BT91] R. Beigel and J. Tarui. On ACC. *Computational Complexity*, 4(4):350–366, 1994. Prelim version FOCS '91.

[Bus90] S. R. Buss. Axiomatizations and conservations results for fragments of bunded arithmetic. In *Logic and Computation, Contemporary Mathematics* 106, pages 57–84. American Math. Society, 1990.

[BV93] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. Prelim version STOC '93.

[BW86] E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470, 1986.

[BYJKS02] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. Prelim version FOCS '02.

[CA08] A. Chattopadhyay and A. Ada. Multiparty communication complexity of disjointness. *ECCC archive*, 2008. Report TR08–002.

[Can96] R. Canetti. More on BPP and the polynomial-time hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996.

[CDR86] S. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, 1986.

[CFL83] A. K. Chandra, M. L. Furst, and R. J. Lipton. Multi-party protocols. In *STOC*, pages 94–99. ACM, 25–27 Apr. 1983.

[CG85] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988. Prelim version FOCS '85.

[Cha94] B. Chazelle. A spectral approach to lower bounds with applications to geometric searching. *SIAM J. Comput.*, 27(2):545–556, 1998. Prelim version FOCS '94.

[Che70] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. In *Problems in Analysis (Papers dedicated to Salomon Bochner, 1969)*, pages 195–199. Princeton Univ. Press, 1970.

[CHSH69] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23(15):880–884, 1969.

[Chu36] A. Church. An unsolvable problem of elementary number theory. *Amer. J. Math.*, 58(2):345–363, 1936.

[Chu90] F. R. K. Chung. Quasi-random classes of hypergraphs. *Random Struct. Algorithms*, 1(4):363–382, 1990.

[Chv73] V. Chvâtal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

[CK00] P. Crescenzi and V. Kann. A compendium of NP optimization problems. http://www.nada.kth.se/~viggo/problemlist/, 2000. Web site tracking the tractability of many NP problems.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science*, pages 24–30. Elsevier/North-Holland, 1964.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd Ann. ACM Symp. Theory of Computing*, pages 151–158. ACM, 1971.

[Coo72] S. A. Cook. A hierarchy for nondeterministic time complexity. *J. Comput. Syst. Sci.*, 7(4):343–353, 1973. Prelim version STOC '72.

[Coo75] S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *STOC*, pages 83–97. ACM, 1975.

[CRVW02] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *STOC*, pages 659–668. ACM, 19–21 May 2002.

[CS88] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988.

[Csa76] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5:618–623, 1976.

[CSWY01] A. Chakrabarti, Y. Shi, A. Wirth, and A. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *FOCS*, pages 270–278. IEEE, 14–17 Oct. 2001.

[CT65] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Computing*, 19:297–301, 1965.

[CW77] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. Prelim version STOC '77.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *FOCS*, pages 14–19. IEEE, 30 Oct.–1 Nov. 1989.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Computation*, 9(3):251–280, 1990.

[Dav65] M. Davis. *The Undecidable*. Dover Publications, 1965.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. Prelim version STOC '91.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Roy. Soc. Lond. A*, A400:97–117, 1985.

[Deu89] D. Deutsch. Quantum computational networks. *Proc. Roy. Soc. Lond. A*, A425:73–90, 1989.

[DFR+07] I. Damgård, S. Fehr, R. Renner, L. Salvail, and C. Schaffner. A tight high-order entropic quantum uncertainty relation with applications. In *Proceedings of 27th CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 360–378. Springer, 2007.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(5):644–654, 1976.

[DHS94] M. Dietzfelbinger, J. Hromkovic, and G. Schnitger. A comparison of two lower-bound methods for communication complexity. *Theor. Comput. Sci*, 168(1): 39–51, 1996. Prelim version MFCS '94.

[Din06] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3), 2007. Prelim version '06.

[DJ92] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proc. Roy. Soc. Lond. A*, 439:553–558, Oct. 1992.

[DK00] D. Z. Du and K. I. Ko. *Theory of Computational Complexity*. Wiley, 2000.

[DL93] P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

[DLne] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[dLMSS56] K. de Leeuw, E. F. Moore, C. E. Shannon, and N. Shapiro. Computability by probabilistic machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 183–212. Princeton University Press, 1956. Annals of Mathematics Studies #34.

[Dod84] J. Dodziuk. Difference equations, isoperimetric inequality and transience of certain random walks. *Trans. Amer. Math. Soc.*, 284(2):787–794, 1984.

[DP60] M. Davis and H. Putnam. A computing procedure for quantification theory. *JACM*, 7(3):201–215, 1960.

[DPV06] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms*. McGraw Hill, 2006. Draft available from the authors' Web page.

[DR02] J. Daemen and V. Rijmen. *The Design of Rijndael: AES—The Advanced Encryption Standard*. Springer, 2002.

[DvM05] S. Diehl and D. van Melkebeek. Time-space lower bounds for the polynomial hierarchy on randomized machines. *SIAM J. Comput.*, 56:563–594, 2006. Prelim version ICALP '05.

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[EL75] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, 1975.

[Ell99] J. H. Ellis. The history of non-secret encryption. *Cryptologia*, 23(3):267–273, 1999.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47(10):777–780, 1935.

[ER59] P. Erdos and A. Renyi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959.

[ER60] P. Erdős and R. Rado. Intersection theorems for systems of sets. *J. Lond. Math. Soc.*, 35:85–90, 1960.

[Fei91] U. Feige. On the success probability of the two provers in one-round proof systems. In *Proceedings of the 6th Annual Conference on Structure in Complexity Theory, CSCT'91 (Chicago, Illinois, June 30–July 3, 1991)*, pages 116–123. IEEE, 1991.

[Fei95] U. Feige. A tight upper bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(1):51–54, 1995.

[Fei96] U. Feige. A threshold of ln for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[Fey82] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6&7):467–488, 1982.

[FG06] J. Flum and M. Grohe. *Parametrized Complexity*. Springer, 2006.

[FGG07] E. Farhi, J. Goldstone, and S. Gutmann. A quantum algorithm for the Hamiltonian NAND tree. *Arxiv preprint quant-ph/0702144*, 2007.

[FGL+91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996. Prelim version FOCS '91.

[Fis04] E. Fischer. The art of uninformed decisions: A primer to property testing. *Current Trends in Theoretical Computer Science: The Challenge of the New Century* (G. Paun, G. Rozenberg, and A. Salomaa, eds.), World Scientific Publishing (2004), Vol. I.

[FK93] U. Feige and J. Kilian. Two-prover protocols – Low error at affordable rates. *SIAM J. Comput.*, 30(1):324–34, 2000.

[FKO06] U. Feige, J. H. Kim, and E. Ofek. Witnesses for non-satisfiability of dense random 3CNF formulas. In *FOCS*, pages 497–508. IEEE, 2006.

[FL92] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems (extended abstract). In *STOC*, pages 733–744. ACM, 1992.

[FLvMV00] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52:835–865, 2005. Prelim version CCC '2000.

[FM05] G. S. Frandsen and P. B. Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Information Processing Letters*, 95(2):354–357, 2005.

[FO04] U. Feige and E. Ofek. Easily refutable subformulas of large random 3CNF formulas. In *Proceedings of 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *Lecture Notes in Computer Science*, pages 519–530. Springer, 2004.

[For89] L. J. Fortnow. *Complexity-Theoretic Aspects of Interactive Proof Systems*. PhD thesis, Massachussets Institute of Technology, 1989.

[For97a] L. Fortnow. Time-space tradeoffs for satisfiability. *J. Comput. Syst. Sci.*, 60(2):337–353, 2000. Prelim version CCC '97.

[For97b] L. Fortnow. Counting complexity. In L. Hemaspaandra and A. Selman, editors, *A Complexity Theory Retrospective II*. Springer Verlag, 1997.

[FR98] A. M. Frieze and B. Reed. Probabilistic analysis of algorithms. In M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, volume 16 of *Algorithms and Combinatorics*, pages 36–92. Springer-Verlag, 1998.

[Fri99] E. Friedgut. Sharp thresholds of graph properties, and the k-sat problem. *J. Amer. Math. Soc.*, 12(4):1017–1054, 1999. With an appendix by Jean Bourgain.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

[FS88] L. Fortnow and M. Sipser. Are there interactive protocols for coNP-languages? *Inf. Process. Lett.*, 28(5):249–251, 1988.

[FS04] L. Fortnow and R. Santhanam. Hierarchy theorems for probabilistic polynomial time. In *FOCS*, pages 316–324. IEEE, 2004.

[FSS81] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984. Prelim version FOCS '81.

[Gal63] R. G. Gallager. *Low-Density Parity-Check Codes*. The MIT Press, 1963.

[Gar72] M. R. Garey. Optimal binary identification procedures. *SIAM J. Appl. Math.*, 23(2):173–186, 1972.

[GG79] O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *J. Comput. Syst. Sci.*, 22(3):407–420, June 1981. Prelim version FOCS '79.

[GG98] O. Goldreich and S. Goldwasser. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000. Prelim version STOC '98.

[GGH+07] S. Goldwasser, D. Gutfreund, A. Healy, T. Kaufman, and G. N. Rothblum. Verifying and decoding in constant depth. In *STOC*, pages 440–449. ACM, 2007.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Prelim version FOCS '84.

[Gil74] J. T. Gill. Computational complexity of probabilistic Turing machines. In *STOC*, pages 91–95. ACM, 1974.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6(4):675–695, Dec. 1977.

[Gil93] D. Gillman. A chernoff bound for random walks on expander graphs. *SIAM J. Comput.*, 27(4):1203–1220, 1998. Prelim version FOCS '93.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[GK01] A. Goerdt and M. Krivelevich. Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In *STACS 2001*, pages 294–304. Springer, 2001.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32. ACM, 15–17 May 1989.

[GLR+91] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *STOC*, pages 32–42. ACM, 1991.

[GM82] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. Prelim version STOC '82.

[GMR84] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. Prelim version CRYPTO '84.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. Prelim version STOC '85.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991. Prelim version FOCS '86.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game—A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson. On yao's XOR-lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(50), 1995.

[Gol97] O. Goldreich. Notes on levin's theory of average-case complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(58), 1997.

[Gol04] O. Goldreich. *Foundations of Cryptography, Volumes 1 and 2*. Cambridge University Press, 2001, 2004.

[Gol08] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge Univerity Press, 2008. Online drafts available at http://www.wisdom.weizmann.ac.il/~oded/cc-drafts.html.

[Gom63] R. Gomory. An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent advances in mathematical programming*, pages 269–302. McGraw-Hill, 1963.

[Gow01] W. Gowers. A new proof of Szemerédi's theorem. *Geometric And Functional Analysis*, 11(3):465–588, 2001.

[Gow07] W. T. Gowers. Hypergraph regularity and the multidimensional Szemerédi theorem. *Ann. of Math. (2)*, 166(3):897–946, 2007.

[GR06] V. Guruswami and A. Rudra. Explicit capacity-achieving list-decodable codes. In *STOC*, pages 1–10. ACM, 2006.

[Gra66] R. Graham. Bounds for certain multiprocessor anomalies. *Bell Sys. Tech. J.*, 45:1563–1581, 1966.

[Gro83] M. Gromov. Filling Riemannian manifolds. *J. Differential Geom.*, 18(1):1–147, 1983.

[Gro96] L. K. Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219. ACM, 22–24 May 1996.

[GS87] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*. JAI Press, 1987. Extended Abstract in *Proc. 18th ACM Symp. on Theory of Computing*, 1986.

[GS92] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, 1992.

[GS98] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. Prelim version FOCS '98.

[GST04] O. Goldreich, M. Sudan, and L. Trevisan. From logarithmic advice to single-bit advice. *Electronic Colloquium on Computational Complexity (ECCC)*, 2004. Report TR04–093.

[GUV07] V. Guruswami, C. Umans, and S. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. In *Proc. of CCC*, pages 96–108. IEEE, 2007.

[GVW00] O. Goldreich, S. Vadhan, and A. Wigderson. Simplified derandomization of BPP using a hitting set generator. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(4), 2000.

[Haj90] P. Hajnal. On the power of randomness in the decision tree model. In *Proceedings, Fifth Annual Structure in Complexity Theory Conference*, pages 66–77. IEEE, 8–11 July 1990.

[Hak85] A. Haken. The intractability or resolution. *Theoretical Comput. Sci.*, 39:297–308, 1985.

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *Bell Sys. Tech. J.*, 29(2):147–160, 1950. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 2, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.

[Hås86] J. Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20. ACM, 28–30 May 1986.

[Hås96] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999.

[Hås97] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. Prelim version STOC '97.

[Hea06] A. Healy. Randomness-efficient sampling within NC^1. In *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 2006.

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Language, and Computation*. Addison–Wesley, 2nd edition, 2001.

[HO02] L. A. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion*. Springer-Verlag, 2002.

[Hoc97] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing, 1997.

[Hod83] A. Hodges. *Alan Turing: The Enigma*. Burnett Books, 1983.

[Hof82] C. M. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism*, volume 136 of *Lecture Notes in Computer Science*. Springer, 1982.

[Hoh30] G. Hoheisel. Primzahlprobleme in der analysis. *Sitzunsberichte der Königlich Preussischen Akademie der Wissenschaften zu Berlin*, 33:3–11, 1930.

[Hol07] T. Holenstein. Parallel repetition: Simplifications and the no-signaling case. In *STOC*, pages 411–419. ACM, 2007.

[HPV75] J. Hopcroft, W. Paul, and L. Valiant. On Time Versus Space. *J. ACM*, 24(2):332–337, 1977. Prelim version FOCS '75.

[HR76] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS66] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape Turing machines. *J. ACM*, 13(4):533–546, 1966.

[HVV04] A. Healy, S. Vadhan, and E. Viola. Using nondeterminism to amplify hardness. In *STOC*, pages 192–201. ACM, 13–15 June 2004.

[IKW01] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002. Prelim version CCC '01.

[IL90] R. Impagliazzo and L. A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *FOCS*, pages 812–823. IEEE, Oct. 1990.

[ILL89] R. Impagliazzo, L. A. Levin, and L. Luby. Pseudo-random generation from one-way functions. In *STOC*, pages 12–24. ACM, May 1989.

[ILMR05] K. Iwama, O. Lachish, H. Morizumi, and R. Raz. An explicit lower bound of $5n - o(n)$ for boolean circuits. Manuscript, 2006. Prelim versions by Raz and Lachish (STOC '01), and Iwama and Morizumi (manuscript, 2005).

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988.

[Imm99] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[Imp95a] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *FOCS*, pages 538–544. IEEE, 1995.

[Imp95b] R. Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference*, pages 134–147, 1995.

[Ing37] A. E. Ingham. On the difference between consecutive primes. *Quarterly Journal of Mathematics (Oxford Series)*, 8:255–266, 1937.

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *STOC*, pages 356–364. ACM, 23–25 May 1994.

[IPZ98] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. Prelim version FOCS '98.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Reducing the seed length in the nisan-wigderson generator. *Combinatorica*, 26(6):647–681, 2006. Prelim versions in FOCS '99 and STOC '00.

[IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the *XOR* lemma. In *STOC*, pages 220–229, New York, 1997. ACM.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. Prelim version FOCS '98.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *FOCS*, pages 248–253. IEEE, 30 Oct.–1 Nov. 1989.

[JM85] S. Jimbo and A. Maruoka. Expanders obtained from affine transformations. *Combinatorica*, 7(4):343–355, 1987. Prelim version STOC '85.

[Joh62] S. Johnson. A new upper bound for error correcting codes. *IRE Trans. Information Theory*, pages 203–207, 1962.

[Joh74] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[Joh84] D. S. Johnson. Solving np-hard problems quickly (on average). *J. Algorithms*, 5(2):284–299, 1984.

[JS90] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993. Prelim version ICALP '90.

[JSV01] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *J. ACM*, 51(4):671–697, 2004.Prelim version STOC '01.

[JVV86] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comp. Sci.*, 43(2–3):169–188, 1986.

[Kab00] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *J. Comput. Syst. Sci.*, 63(2):236–252, 2001. Prelim version CCC '00.

[Kah92] N. Kahale. Eigenvalues and expansion of regular graphs. *J. ACM*, 42:1091–1106, 1995. Prelim version FOCS '92.

[Kah96] D. Kahn. *The Codebreakers: The Story of Secret Writing*. Scribner, revised edition, 1996.

[Kah97] N. Kahale. Large deviation bounds for markov chains. *Combinatorics, Probability and Computing*, 6(04):465–474, 1997.

[Kan81] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1–3):40–56, 1982. Prelim version FOCS '81.

[Kan83] R. Kannan. Alternation and the power of nondeterminism. In *STOC*, pages 344–346. ACM, 1983.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, 1972.

[Kha79] L. C. Khačiyan. Polynomial algorithm for linear programming. *Soviet Doklady*, 244:1093–1096, 1979. Typed translation.

[Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *STOC*, pages 767–775. ACM, 2002.

[Kho05] S. Khot. Guest column: Inapproximability results via long code based PCPs. *SIGACT News*, 36(2):25–42, 2005.

[KI03] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *STOC*, pages 355–364. ACM, 2003.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 2–4 May 1988.

[Kin88] V. King. An omega(n^5/4) lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(1):23–32, 1991. Prelim version STOC '88.

[Kit97] A. Y. Kitaev. Quantum computations: Algorithms and error correction. *RMS: Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

[KKL88] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions (extended abstract). In *FOCS*, pages 68–80. IEEE, 1988.

[KKMO05] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

[KL80] R. Karp and R. Lipton. Turing machines that take advice. *L' Enseignement Mathématique*, 28:191–210, 1982.

[KL07] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[KM91] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993. Prelim version STOC '91.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[Knu69] D. E. Knuth. *Art of Computer Programming, Volume II*. Addison Wesley, 1969. Current edition: 1997.

[Knu73] D. E. Knuth. *The Art of Computer Programming, Vol. 3 : Sorting and Searching*. Series in Computer Science and Information Processing. Addison-Wesley, 1973. Current edition: 1997.

[Koi97] P. Koiran. Randomized and deterministic algorithms for the dimension of algebraic varieties. In *FOCS*, pages 36–45. IEEE, 1997.

[Koz97] D. C. Kozen. *Automata and Computability*. Springer-Verlag, 1997.

[Koz06] D. C. Kozen. *Theory of Computation*. Springer, 2006.

[KPS85] R. M. Karp, N. Pippenger, and M. Sipser. A time randomness tradeoff. In *AMS Conf. on Probabilistic Computational Complexity*, 1985. This paper gives the first example of deterministic amplification using expander graphs.

[KR81] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM J. Res. Dev.*, 31(2):249–260, 1987. Prelim version report TR-31-81, Harvard University, 1981.

[KR08] S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

[Kra94] J. Krajíček. Lower bounds to the size of constant-depth propositional proofs. *J. Symbolic Logic*, 59(1):73–86, 1994.

[Kra95] J. Krajíček. *Bounded Arithmetic, Propositional Logic and Complexity Theory*. Cambridge University Press, 1995.

[Kra97] J. Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *J. Symbolic Logic*, 62(2):457–486, 1997.

[KRW95] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3/4):191–204, 1995.

[KS99] A. R. Klivans and R. A. Servedio. Boosting and hard-core set construction. *Machine Learning*, 51(3):217–238, 2003. Prelim version FOCS '99.

[KS06] G. Kalai and S. Safra. Threshold phenomena and influence. In C. M. A. G. Percus, G. Istrate, editor, *Computational Complexity and Statistical Physics*. Oxford University Press, 2006.

[KSS83] J. Kahn, M. E. Saks, and D. Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4(4):297–306, 1984. Prelim version FOCS '83.

[KT06] J. Kleinberg and É. Tardos. *Algorithm Design*. Pearson Studium, 2006.

[Kuc95] L. Kucera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2–3):193–212, 1995.

[KUW85] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6:35–48, 1986. Prelim version STOC '85.

[KV89] M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994. Prelim version STOC '89.

[KV94] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[KV05] S. Khot and N. K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into l_1. In *FOCS*, pages 53–62. IEEE, 2005.

[KVS02] A. Y. Kitaev, M. Vyalyi, and A. Shen. *Classical and Quantum Computation*. AMS Press, 2002.

[KW88] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Mathematics*, 3(2):255–265, May 1990. Prelim version STOC '88.

[Lad75] R. E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17(4):215–217, 1983.

[Lea05] D. Leavitt. *The Man Who Knew too Much: Alan Turing and the Invention of the Computer*. Great Discoveries. W. W. Norton & Co., 2005.

[Lei91] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Array, Trees, Hypercubes*. Morgan Kaufmann, 1991.

[Lev73] L. A. Levin. Universal sequential search problems. *PINFTRANS: Problems of Information Transmission (translated from Problemy Peredachi Informatsii (Russian))*, 9, 1973.

[Lev86] L. A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[Lev87] L. A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. Prelim version FOCS '90.

[Lip90] R. J. Lipton. Efficient checking of computations. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *lncs*, pages 207–215. Springer, 22–24 Feb. 1990.

[Lip91] R. J. Lipton. New directions in testing. In *Distributed Computing and Cryptography*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. American Mathematics Society, 1991.

[Liv06] N. Livne. All natural NPC problems have average-case complete versions. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2006.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. John Wiley, 1985.

[LLMP90] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard. The number field sieve. In *STOC*, pages 564–572. ACM, 14–16 May 1990.

[Llo06] S. Lloyd. *Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos*. Knopf, 2006.

[Lov78] L. Lovàsz. Kneser's conjecture, chromatic number, and homotopy. *J. Combin. Theory Ser. A*, 25:319–324, 1978.

[Lov79] L. Lovàsz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computation Theory FCT '79*, pages 565–574. Akademie-Verlag, 1979.

[Lov07] L. Lovász. *Combinatorial Problems and Exercises*. AMS Chelsea Publishing, Providence, RI, second edition, 2007. Corrected reprint of 1993 second edition.

[LPS86] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988. Prelim version STOC '86.

[LRVW03] C.-J. Lu, O. Reingold, S. Vadhan, and W. Wigderson. Extractors: optimal up to constant factors. In *STOC*, pages 602–611. ACM, 2003.

[LS88] L. Lovász and M. E. Saks. Communication complexity and combinatorial lattice theory. *J. Comput. Syst. Sci.*, 47(2):322–349, 1993. Prelim version FOCS '88.

[LS91] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXP-time (extended abstract). In *FOCS*, pages 13–18. IEEE, 1991.

[LS07] T. Lee and A. Shraibman. Disjointness is hard in the multi-party number on the forehead model, 27 Dec. 2007. Comment: 21 pages.

[Lup58] O. Lupanov. The synthesis of contact circuits. *Dokl. Akad. Nauk SSSR*, 119:23–26, 1958.

[LW06] M. Luby and A. Wigderson. Pairwise independence and derandomization. *Found. Trends Theor. Comput. Sci.*, 1(4):237–301, 2006.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[LY02] L. Lovász and N. E. Young. Lecture notes on evasiveness of graph properties. *CoRR*, cs.CC/0205031, 2002. Informal publication.

[Maa84] W. Maass. Quadratic lower bounds for deterministic and nondeterministic one-tape turing machines. In *STOC*, pages 401–408. ACM, 1984.

[Mah80] S. R. Mahaney. Sparse complete sets of *NP*: solution of a conjecture of berman and hartmanis. *J. Comput. Syst. Sci.*, 25(2):130–143, 1982. Prelim version FOCS '80.

[Mar73] G. A. Margulis. Explicit constructions of concentrators. *Probl. Peredachi Inf.*, 9(4):71–80, 1973. English translation in *Problems of Information Transmission*, 9(4):325–332, 1973.

[Mar88] G. A. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Probl. Peredachi Inf.*, 24(1):51–60, 1988. English translation in *Problems of Information Transmission*, 24(1):39–46, 1988.

[Mat76] D. W. Matula. The largest clique size in a random graph. Technical report, Dept. of Computer Science, Southern Methodist University, 1976.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002. Online updated chapters available on http://kam.mff.cuni.cz/~matousek/dg.html.

[MG02] D. Micciancio and S. S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*. Kluwer Academic Publishers, 2002.

[MOO05] E. Mossel, R. O'Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *FOCS*, pages 21–30. IEEE, 2005.

[Mor73] J. Morgenstern. Note on a lower bound on the linear complexity of the fast fourier transform. *J. ACM*, 20(2):305–306, 1973.

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.

[MR00] R. Martin and D. Randall. Sampling adsorbing staircase walks using a new markov chain decomposition method. In *FOCS*, pages 492–502. IEEE, 12–14 Nov. 2000.

[MS72] A. R. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *FOCS*, pages 125–129. IEEE, 1972.

[MS82] K. Mehlhorn and E. M. Schmidt. Las Vegas is better than determinism in VLSI and distributed computing (extended abstract). In *STOC*, pages 330–337. ACM, 5–7 May 1982.

[MU05] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[Mul54] D. Muller. Application of boolean algebra to switching circuit design and to error detection. *IRE Trans. Electronic Computation*, EC-3:6–12, 1954.

[MVV87] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[MZ04] J. Matoušek and G. M. Ziegler. Topological lower bounds for the chromatic number: A hierarchy. *Jahresber. Deutsch. Math.-Verein.*, 106(2):71–90, 2004.

[NC00] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[Nil04] A. Nilli. Tight estimates for eigenvalues of regular graphs. *Electr. J. Comb.*, 11(1), 2004.

[Nis89] N. Nisan. CREW PRAMs and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991. Prelim version STOC '89.

[Nis90] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. Prelim version STOC '90.

[NS92] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994. Prelim version STOC '92.

[NSW92] N. Nisan, E. Szemeredi, and A. Wigderson. Undirected connectivity in $O(log^{1.5}n)$ space. In *FOCS*, pages 24–29. IEEE, Oct. 1992.

[NW88] N. Nisan and A. Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. Prelim version FOCS '88.

[NW94] N. Nisan and A. Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995. Prelim version FOCS '94.

[NZ93] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996. Prelim version STOC '93.

[O'D04] R. O'Donnell. Hardness amplification within NP. *J. Comput. Syst. Sci.*, 69(1):68–94, 2004.

[Ost54] A. M. Ostrowski. On two problems in abstract algebra connected with Horner's rule. *Studies in Mathematics and Mechanics*. Academic, 1954.

[Pap85] C. H. Papadimitriou. Games against nature. *J. Comput. System Sci.*, 31(2):288–301, 1985.

[Pap90] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. Prelim version FOCS '90.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pen89] R. Penrose. *The Emperor's New Mind*. Oxford University Press, 1989.

[Pen90] R. Penrose. The Emperor's New Mind. *Bull. Amer. Math. Soc. 23 (1990)*, 606–616, 1990.

[Pet60] W. W. Peterson. Encoding and error-correction procedures for bose-chaudhuri codes. *IEEE Trans. Information Theory*, 6:459–470, 1960.

[Pin73] M. S. Pinsker. On the complexity of a concentrator. *Proc. 7th Int. Teletraffic Cong.*, 1973.

[Pol65] S. L. Pollack. Conversion of limited-entry decision tables to computer programs. *Commun. ACM*, 8(11):677–682, 1965.

[Pra75] V. R. Pratt. Every prime has a succinct certificate. *SIAM J. Comput.*, 4:214–220, 1975.

[Pre97] J. Preskill. Fault tolerant quantum computation. *Arxive e-print*, 1997. arXiv:quant-ph/9712048v1.

[Pre98] J. Preskill. Reliable quantum computers. *Proc. Roy. Soc. A: Mathematical, Physical and Engineering Sciences*, 454(1969):385–410, 1998.

[PS82] C. H. Papadimitriou and M. Sipser. Communication complexity. *J. Comput. Syst. Sci.*, 28(2):260–269, 1984. Prelim version STOC '82.

[Pud97] P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *J. Symbolic Logic*, 62(3):981–998, 1997.

[PV05] F. Parvaresh and A. Vardy. Correcting errors beyond the guruswami-sudan radius in polynomial time. In *FOCS*, pages 285–294. IEEE, 2005.

[PV06] C. Papadimitriou and U. Vazirani. Lecture notes for CS70: Discrete mathematics for computer science, 2006. Available from the authors' home pages.

[PY82] C. H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *J. Comput. Syst. Sci.*, 28(2):244–259, 1982.

[PY88] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. Prelim version STOC '88.

[Rab72] M. O. Rabin. Proving simultaneous positivity of linear forms. *J. Comput. Syst. Sci.*, 6:639–650, 1972.

[Rab79] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Jan. 1979.

[Rab80] M. O. Rabin. A probabilistic algorithm for testing primality. *J. Number Theory*, 12(1):128–138, 1980.

[Raz95] R. Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. Prelim version STOC '95.

[Raz00] R. Raz. The BNS-chung criterion for multi-party communication complexity. *Computational Complexity*, 9(2):113–122, 2000.

[Raz02] R. Raz. On the complexity of matrix product. *SIAM J. Comput.*, 32(5):1356–1369, 2003. Prelim version STOC '02.

[Raz04] R. Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. In *STOC*, pages 633–641. ACM, 2004.

[Razb85a] A. A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Dokl. Akad. Nauk. SSSR*, 281(4):798–801, 1985. Translation in *Soviet Math. Dokl.* 31:354–357.

[Razb85b] A. A. Razborov. A lower bound on the monotone network complexity of the logical permanent. *Matematicheskie Zametki*, 37(6):887–900, 1985. In Russian. English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 37(6):485–493.

[Razb87] A. A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition (Russian), in Matematicheskie Zametki, 41(4):598–607, 1987. English Translation in *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[Razb89] A. A. Razborov. On the method of approximations. In *STOC*, pages 169–176. ACM, 1989.

[Razb90] A. A. Razborov. On the distributed complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992. Prelim version ICALP '90.

[Razb92] A. A. Razborov. On submodular complexity measures. In M. Paterson, editor, *Boolean Function Complexity*, pages 76–83. London Mathematical Society Lecture Note Series 169. Cambridge University Press, 1992.

[Razb95] A. A. Razborov. Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic. *Izvestiya of the RAN*, 59(1):201–224, 1995.

[Razb98] A. A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7:291–324, 1998.

[Razb01] A. A. Razborov. Improved resolution lower bounds for the weak pigeonhole principle. Technical Report TR01-055, Electronic Colloquium on Computational Complexity, 2001.

[Razb03a] A. A. Razborov. Resolution lower bounds for the weak functional pigeonhole principle. *Theoretical Computer Science*, 303(1):233–243, 2003.

[Razb03b] A. A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution, 2003. Unpublished manuscript, available from the author's Web page.

[Razb04a] A. A. Razborov. Resolution lower bounds for perfect matching principles. *J. Comput. Syst. Sci.*, 69(1):3–27, 2004.

[Razb04b] A. A. Razborov. Feasible proofs and computations: Partnership and fusion. In *Proceedings of the 31st International Colloquium, Lecture Notes in Computer Science*, 3142, pages 8–14. Springer-Verlag, 2004. Also appeared in Proceedings of the 19th LICS conference.

[Ree54] I. S. Reed. A class of multiple error-correcting codes and the decoding scheme. *IRE Trans. Information Theory*, PGIT-4:38–49, 1954.

[Reg06] O. Regev. Lattice-based cryptography. In *Proceedings of 26th CRYPTO, Lecture Notes in Computer Science*, 4117, pages 131–141. Springer, 2006.

[Rei05] O. Reingold. Undirected ST-connectivity in log-space. In *STOC*, pages 376–385. ACM, 2005.

[Rog87] H. Rogers Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987.

[Ros06] K. H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill, 2006.

[Rot93] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1–2):273–302, 1996. Prelim version IJCAI '93.

[RR94] A. A. Razborov and S. Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. Prelim version STOC '94.

[RR99] R. Raz and O. Reingold. On recycling the randomness of states in space bounded computation. In *STOC*, pages 159–168. ACM, 1999.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J. Soc. Industrial Applied Math.*, 8(2):300–304, 1960.

[RS91] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992, 11–15 Aug. 1991.

[RS92] R. Rubinfeld and M. Sudan. Self-testing polynomial functions efficiently and over rational domains. In *SODA*, pages 23–32, 1992.

[RS93] R. Raz and B. Spieker. On the "log rank"-conjecture in communication complexity. *Combinatorica*, 15(4):567–588, 1995. Prelim version FOCS '93.

[RS95] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7(2):152–162, 1998. Prelim version MIT Tech report, 1995.

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test. In *STOC*, pages 475–484. ACM, 1997.

[RSA78] R. L. Rivest, A. Shamir, and L. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[RTS97] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Mathematics*, 13(1):2–24, 2000. Prelim version FOCS '97.

[RTV06] O. Reingold, L. Trevisan, and S. Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *STOC*, pages 457–466. ACM, 2006.

[Rub90] R. Rubinfeld. *A Mathematical Theory of Self-Checking, Self-Testing and Self-Correcting Programs*. PhD thesis, UC Berkeley, 1990.

[RV76] R. L. Rivest and J. Vuillemin. On recognizing graph properties from adjacency matrices. *Theor. Comput. Sci.*, 3(3):371–384, 1976.

[RV05] E. Rozenman and S. Vadhan. Derandomized squaring of graphs. In *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*. LNCS, 2005.

[RVW00] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *FOCS*, pages 3–13. IEEE, 2000.

[RW93] A. A. Razborov and A. Wigderson. $n^{\Omega(\log n)}$ Lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom. *Inform. Process. Lett.*, 45(6):303–307, 1993.

[San07] R. Santhanam. Circuit lower bounds for Merlin-Arthur classes. STOC pages 275–283. ACM, 2007.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4:177–192, 1970.

[Sav72] J. E. Savage. Computational work and time on finite machines. *J. ACM*, 19(4):660–674, 1972.

[SBR02] M. V. Sapir, J.-C. Birget, and E. Rips. Isoperimetric and isodiametric functions of groups. *Ann. Math. (2)*, 156(2):345–466, 2002.

[Sch37] A. Scholz. Aufgabe 253. *Jahresber. DMV*, 1937.

[Sch44] E. Schroedinger. *What Is Life?* Cambridge University Press, 1944.

[Sch96] M. Schaefer. Deciding the Vapnik-Cervonenkis dimension is Σ_3^p-complete. *J. Comput. Syst. Sci.*, 58(2):177–182, 1999. Prelim version CCC '96.

[SG76] S. Sahni and T. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. J.*, 27(3):379–423, 1948.

[Sha49a] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell Sys. Tech. J.*, 28(1):59–98, 1949.

[Sha49b] C. E. Shannon. Communication theory of secrecy systems. *Bell Sys. Tech. J.*, 28:656–715, 1949.

[Sha79] A. Shamir. Factoring numbers in O(log n) arithmetic steps. *Inf. Process. Lett.*, 8(1):28–31, 1979.

[Sha81] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Computer Sys.*, 1(1):38–44, 1983. Prelim version presented at Crypto '81 and ICALP '81.

[Sha90] A. Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992. Prelim version FOCS '90.

[Sha02] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bul. EATCS*, 77:67–95, 2002.

[She92] A. Shen. IP = PSPACE: Simplified proof. *J. ACM*, 39(4):878–880, 1992.

[She07] A. A. Sherstov.The pattern matrix method for lower bounds on quantum communication. Technical Report CS-TR-07–46, The University of Texas at Austin, Department of Computer Sciences, Sept. 6 2007. Thu, 3 Jan 108 21:31:32 GMT.

[Shi03] Y. Shi. Both toffoli and controlled-NOT need little help to universal quantum computing. In *Quantum Information and Computation*, volume 3. Rinton Press, 2003.

[SHL65] R. E. Stearns, J. Hartmanis, and P. M. Lewis. Hierarchies of memory limited computations. In *FOCS*, pages 179–190. IEEE, 1965.

[Sho95] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):2493–2496, 1995.

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Sho05] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.

[Sim94] D. R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5):1474–1483, 1997. Prelim version FOCS '94.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *STOC*, pages 330–335. ACM, 25–27 Apr. 1983.

[Sip86] M. Sipser. Expanders, randomness, or time versus space. *J. Comput. Syst. Sci.*, 36:379–383, 1988. Prelim version CSCT '86.

[Sip92] M. Sipser. The history and status of the P versus NP question. In *STOC*, pages 603–618. ACM, 1992.

[Sip96] M. Sipser. *Introduction to the Theory of Computation*. PWS, 1996.

[SJ88] A. Sinclair and M. Jerrum. Approximative counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.*, 82(1):93–133, 1989. Prelim version International Workshop on Graph-Theoretic Concepts in Computer Science '88.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *STOC*, pages 1–9. ACM, 1973.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *STOC*, pages 77–82. ACM, 1987.

[Sni81] M. Snir. Lower bounds on probabilistic linear decision trees. *Theor. Comput. Sci.*, 38(1):69–82, 1985. Prelim version ICALP '81.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing (Arch. Elektron. Rechnen)*, 7:281–292, 1971.

[SS77] R. Solovay and V. Strassen. A fast Monte Carlo test for primality. *SIAM J. Comput.*, 6(1):84–85, 1977.

[ST01] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004. Prelim version STOC '01.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.

[Str72] V. Strassen. Berechnung und programm. I. *Act. Inf.*, 1:320–335, 1972. In German.

[Str73] V. Strassen. Vermeidung von divisionen. *J. reine angew. Math.*, 264:184–202, 1973. In German.

[STV99] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. Prelim version STOC '99.

[SU01] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005. Prelim version FOCS '01.

[SU02a] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: Part I. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 33, Sept. 2002.

[SU02b] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: Part II. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 33, Dec. 2002.

[Sud96] M. Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. Prelim version FOCS '96.

[Sud01] M. Sudan. Coding theory: Tutorial & survey. In *FOCS*, pages 36–53. IEEE, 2001.

[SV84] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986. Prelim version FOCS '84.

[SV85] S. Skyum and L. G. Valiant. A complexity theory based on boolean algebra. *J. ACM*, 32(2):484–502, 1985.

[SW86] M. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *FOCS*, pages 29–38. IEEE, Oct. 1986.

[SZ94] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. *SIAM J. Comput.*, 28(4):1433–1459, 1999. Prelim version FOCS '94.

[SZ95] M. Saks and S. Zhou. $BP_HSPACE(S) \subseteq DSPACE(S^{3/2})$. *J. Comput. Syst. Sci.*, 58(2):376–403, 1999. Prelim version FOCS '95.

[Sze76] E. Szemerédi. Regular partitions of graphs. *Problèmes combinatoires et théorie des graphes*, Orsay, 1976.

[Sze87] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Bulletin of the European Association for Theoretical Computer Science*, 33:96–100, Oct. 1987. Technical Contributions.

[Tan84] R. M. Tanner. Explicit construction of concentrators from generalized n-gons. *SIAM J. Algebraic Discrete Methods*, 5:287–293, 1984.

[Tar88] É. Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[Tra84] B. A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984. Also contains a good translation of [Lev73].

[Tre99] L. Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001. Prelim version STOC '99.

[Tre05] L. Trevisan. Inapproximability of combinatorial optimization problems. In V. Paschos, editor, *Optimisation Combinatiore*, volume 2. Hermes, 2005. English version available from author's Web page.

[Tri05] V. Trifonov. An $O(\log n \log \log n)$ space algorithm for undirected st-connectivity. In *STOC*, pages 626–633. ACM, 2005.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *STOC*, pages 276–285. ACM, May 1996.

[Tur36] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings, London Mathematical Society.*, pages 230–265, 1936. Published as *Proceedings, London Mathematical Society*, volume 2, number 42.

[TV02] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity, 2002. Proceedings. 17th IEEE Annual Conference on*, pages 103–112, 2002.

[TV06] T. Tao and V. H. Vu. *Additive Combinatorics*. Cambridge University Press, 2006.

[Uma98] C. Umans. The minimum equivalent DNF problem and shortest implicants. *J. Comput. Syst. Sci.*, 63(4):597–611, 2001. Prelim version FOCS '98.

[Uma03] C. Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003.

[Urq87] A. Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987.

[Vad99] S. P. Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, 1999. Revises version to appear in Springer Series on Information Security and Cryptography, 2009.

[Vad00] S. Vadhan. On transformation of interactive proofs that preserve the prover's complexity. In *STOC*, pages 200–207. ACM, 2000.

[Vad07] S. Vadhan. The unified theory of pseudorandomness. *SIGACT News*, 38(2), 2007.

[Val75a] L. G. Valiant. Graph-theoretic properties in computational complexity. *J. Comput. Syst. Sci.*, 13(3):278–285, 1976. Prelim version STOC '75.

[Val75b] L. G. Valiant. On non-linear lower bounds in computational complexity. In *STOC*, pages 45–53. ACM, 1975.

[Val79a] L. G. Valiant. Completeness classes in algebra. In *STOC*, pages 249–261. ACM, 1979.

[Val79b] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[Val79c] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[Vaz01] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[vDMV01] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *FOCS*, pages 279–287. IEEE, 14–17 Oct. 2001.

[Ver94] O. Verbitsky. Towards the parallel repetition conjecture. In *Structure in Complexity Theory Conference*, pages 304–307, 1994.

[VG99] J. Von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[vM07] D. van Melkebeek. *A survey of lower bounds for satisfiability and related Problems. Foundations and Trends in Theoretical Computer Science*, 2(3):197–303, 2007.

[vN45] J. von Neumann. First draft of a report on the EDVAC. Report for the U.S. Army Ordinance Department, University of Pensylvania, 1945. Reprinted in part in Brian Randell, editor, *The Origins of Digital Computers: Selected Papers*, Springer Verlag, 1982.

[vN51] J. von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 12:36–38, 1951.

[VSBR81] L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983. Prelim version Mathematical Foundations of CS '81.

[VV85] U. V. Vazirani and V. V. Vazirani. Random polynomial time is equal to slightly-random polynomial time. In *FOCS*, pages 417–428. IEEE, 1985.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(1):85–93, 1986.

[vzG88] J. von zur Gathen. Algebraic complexity theory. *Annual Reviews Computer Science*, 3:317–347, 1988.

[vzGG99] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[Wat03] J. Watrous. PSPACE has constant-round quantum interactive proof systems. *Theor. Comput. Sci*, 292(3):575–588, 2003.

[Weg87] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner Series in Computer Science, 1987. Online version available from http://eccc.hpi-web.de/eccc-local/ECCC-Books/wegener_book_readme.html.

[Wel93] D. J. A. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.

[Wig06] A. Wigderson. P, NP and mathematics – A computational complexity perspectivy. In *Proceedings of ICM '06*, 2006.

[Wil05] R. Williams. Better time-space lower bounds for SAT and related problems. In *IEEE Conference on Computational Complexity*, pages 40–49. IEEE, 2005.

[Wil07] R. Williams. Time-space tradeoffs for counting NP solutions modulo integers. Computational Complexity 17(2):179–219, 2008. Prelim version CCC '07.

[WX05] A. Wigderson and D. Xiao. A randomness-efficient sampler for matrix-valued functions and applications. In *FOCS*, pages 397–406. IEEE, 2005. See also correction in ECCC TR05-107 Revision 01.

[WZ82] W. K. Wooters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802f, 1982.

[Yam97] P. Yam. Bringing Schroedinger's cat back to life. *Scientific American*, pages 124–129, June 1997.

[Yan88] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. *J. Comput. Syst. Sci.*, 43(3):441–466, 1991. Prelim version STOC '88.

[Yao79] A. C. C. Yao. Some complexity questions related to distributive computing(prelim report). In *STOC*, pages 209–213. ACM, 1979.

[Yao82a] A. C. C. Yao. Theory and applications of trapdoor functions. In *FOCS*, pages 80–91. IEEE, 3–5 Nov. 1982.

[Yao82b] A. C. C. Yao. Protocols for secure computations. In *FOCS*, pages 160–164. IEEE, 3–5 Nov. 1982.

[Yao85] A. C. C. Yao. Separating the polynomial-time hierarchy by oracles. In *FOCS*, pages 1–10. IEEE, 1985.

[Yao87] A. C. C. Yao. Lower bounds to randomized algorithms for graph properties. *J. Comput. Syst. Sci.*, 42(3):267–287, 1991. Prelim version FOCS '87.

[Yao90] A. C. C. Yao. On ACC and threshold circuits. In *FOCS*, volume II, pages 619–627. IEEE, 22–24 Oct. 1990.

[Yao93] A. C. C. Yao. Quantum circuit complexity. In *FOCS*, pages 352–361. IEEE, 1993.

[Yao94] A. C. C. Yao. Decision tree complexity and betti numbers. In *STOC*, pages 615–624. ACM, 1994.

[Zak83] S. Zak. A Turing machine time hierarchy. *Theoret. Computer Sci.*, 26(3):327–333, 1983.

[Zuc90] D. Zuckerman. General weak random sources. In *FOCS*, volume II, pages 534–543. IEEE, 22–24 Oct. 1990.

术语索引

索引中的页码为英文原书页码，与书中页边标注的页码一致。

⊕ SATproblem(⊕ SAT 问题)，参见 computational problem
#CYCLE problem(#CYCLE 问题)，参见 computational problem
#SAT problem(#SAT 问题)，参见 computational problem
  #**P**-completeness(#**P**-完全性)，346
  Decision Version(判定形式)，158
2-Slit experiment(双缝实验)，202
3COL problem(3COL 问题)
  NP-completeness(**NP**-完全性)，66
3SAT problem(3SAT 问题)，参见 computational problem
3SAT problem(3SAT 问题)
  Average case(平均情况)，363

A

Adleman's Theorem(阿德尔曼定理)，136
Algorithmica(算法界)，参见 Impagliazzo's worlds
Alternative Turing machine(交错图灵机)，参见 Turing machine
Approximation algorithm(近似算法)，238
Arithmetization(算术化)，158
Average-case hardness(平均难度)，参见 hardness
Averaging principle(平均值原理)，147，310，377，409，512

B

Baker-Gill-Solovay Theorem(贝克-吉尔-索洛韦定理)，74
Bell's inequality(贝尔不等式)，207
Berman's Theorem(贝尔曼定理)，66
Big-Oh notation(大 O 记号)，3
Blum-Shub-Smale model(布卢姆-舒布-斯梅尔模型)，参见 Turing machine

C

Chebychev's inequality(切比雪夫不等式)，184
Chernoff bound(切尔诺夫界)，133
Church-Turing thesis(邱奇-图灵命题)，26
CIRCUIT-EVAL problem(CIRCUIT-EVAL 问题)
  P-completeness(**P**-完全性)，119
Circuit(线路)
  Algebraic(代数线路)，322
  Boolean(布尔线路)，107，215
    Counting lower bound(计数下界)，115
    DC Uniform(DC 一致性)，119
    Equivalence to straight-line programs(与直线程序的等价性)，109
    Uniform generation(一致生成)，111
  Lower bound(线路下界)，286-304，415，499
  Quantum(量子线路)，214
    Universal basis(量子通用基)，216
CKT-SAT problem(CKT-SAT 问题)，110
  NP-completeness(**NP**-完全性)，111
Clique problem(团问题)
  NP-completeness(**NP**-完全性)，65
CNF form(CNF 范式)，44
Coin tossing(硬币投掷)，192
Combinatorial auction problem(组合拍卖问题)
  NP-completeness(**NP**-完全性)，65
Combinatorial Design(组合设计)，410，412
Communication Complexity(通信复杂性)，271
Computability theory(可计算理论)，21
Computational problem(计算问题)
  ⊕SAT，353，355
  #CYCLE，342，343
  #SAT，342
  3COLproblem(3COL 问题)
    NP-completeness(**NP**-完全性)，65
  3SAT，44
  3SAT problem(3SAT 问题)
    Average case(平均情况)，363
  CIRCUIT-EVAL problem(CIRCUIT-EVAL 问题)
    P-completeness(**P**-完全性)，119

CKT-SAT problem(CKT-SAT 问题), 110
Exactly-One 3SAT
  NP-completeness(NP-完全性), 65
Clique problem(团问题)
  NP-completeness(NP-完全性), 65
Combinatorial auction problem(组合拍卖问题)
  NP-completeness(NP-完全性), 66
Constraint satisfaction(约束满足), 244
Disjointness(不相交性), 273
PATH, 82
PATH problem(PATH 问题)
  NL-completeness(NL-完全性)
Factoring(因数分解), 40, 221
Graph connectivity(图连通性), 25, 40
Graph isomorphism(图同构), 40, 148, 156, 165, 188, 242
Halting(停机), 23
Hamiltonian cycle problem(哈密顿环问题)
  NP-completeness(NP-完全性), 65
Independent set(独立集), 39
Integer programming(整数规划), 40
Integer programming problem(整数规划问题)
  NP-completeness(NP-完全性), 52
Linear programming(线性规划), 40
Max-cut problem(最大割问题)
  NP-completeness(NP-完全性), 65
Perfect matching(完美匹配), 130
Permanent(积和式), 166-169, 242, 323, 325, 414
Permanent(积和式)
  #P-completeness(#P-完全性), 347-351
Planted clique(植入团), 362
Polynomial identity testing(多项式恒等测试), 129, 403
Primality testing(素性测试), 40, 64, 128
Quadratic equations problem(二次方程组问题)
  NP-completeness(NP-完全性), 65
Quadratic Residuosity(二次剩余), 149
SAT, 38, 44, 44-50, 54-56, 62, 65, 71-73
Set cover(集合覆盖), 456
$\sum_i^3$SAT, 67
Subset sum(子集和), 40
Subset sum problem(子集和问题)
  NP-completeness(NP-完全性), 65
Subset sum problem(子集和问题)
  pseudo-polynomial time algorithm(伪多项式时间算法), 66
  Tautology(永真式), 56
  TQBF, 161, 165
  Traveling salesperson(旅行售货商), 40
  Unique SAT(唯一性 SAT), 354
  UPATH, 139, 426, 440
Configuration graph(格局图), 80
Constraint satisfaction problem(约束满足问题), 参见 computational problem
Continued functions(连续函数), 229
Cook-reductions(库克归约), 参见 reduction
Cook-Levin Theorem(库克-勒维定理), 45, 110
  Quantum analog(量子计算中的类似定理), 231
Cryptomania(密码界), 参见 Impagliazzo's worlds
Cutting planes proof system(分割平面证明系统), 313

D

Decision tree(判定树), 260
  Algebraic(代数判定树), 326
Designs(设计), 参见 Combinatorial Design
Determinant(行列式), 122, 130, 325, 346, 523
Discrepancy(差异), 275
Disjointness problem(不相交问题), 参见 computational problem
Distributional problem(分布问题), 364

E

Easy witness method(显然证据法), 416
Eigenvalue(特征值), 422-426, 526
Eigenvector(特征向量), 参见 Eigenvalue
Einstein-Podosky-Rosen Paradox(爱因斯坦-波多尔斯基-罗森悖论), 206
Encryption(密码)
  Perfect secrecy(完美安全密码), 175
  Private key(私钥密码), 178
  Public key(公钥密码), 178
Error correcting code(纠错码), 379
  concatenated(拼接纠错码), 384
  Gilbert-Varshamov bound(吉尔伯特-瓦尔沙莫夫界), 380
  Listing decoding(列表解码), 392
  Long code(长编码), 480
  Reed-Muller(里德-穆勒纠错码), 383
  Local decoder(局部解码算法), 387

Local list decoding(局部列表解码), 395
Reed-Solomon(里德-所罗门纠错码), 382
Berlekamp-Welch decoder(波利坎普-韦尔奇解码算法)
Sudan List decoder(苏丹列表解码算法)
Walsh-Hadamard(沃尔什-哈达玛纠错码), 212, 249-250, 382
Local decoder(局部解码算法), 250, 388
Local list decoding(局部列表解码), 183, 395
Local testing(局部测试), 477
Error reduction(错率归约), 132, 431
Exactly-One 3SAT
NP-completeness(**NP**-完全性), 65
Expander graph(扩张图), 426-431
Chernoff bound(扩张图的切尔诺夫界), 434
Error reduction(扩张图错率归约), 431
Explicit construction(扩张图的显式构造), 434-440
Use in **PCP**(扩张图在 **PCP** 中的应用), 463-470, 491
Expander mixing lemma(扩张图平稳引理), 429
Extractor(提取器), 442-453

F

Factoring problem(因素分解问题), 参见 computational problem
Feasible interpolation theorem(易用插值定理), 312
Fingerprinting(指纹技术), 130
Fourier transform(傅里叶变换), 223
$GF(2)^n$($GF(2)^n$上的傅里叶变换), 475
Cyclic group(周期群上的傅里叶变换), 223
Fast Fourier transform(快速傅里叶变换), 223
Quantum(量子傅里叶变换), 224
Frege proof system(弗雷格证明系统), 315

G

Gödel's Theorem(哥德尔定理), 23
Goldreich-levin Theorem(戈德赖希-勒维定理), 183, 395
Alternative proof(另一种证明), 497
Goldwasser-sipser Theorem(戈德瓦塞-西普塞尔定理), 151
Graph connectivity problem(图连通性问题), 参见 computational problem
Graph isomorphism problem(图同构问题), 参见 computational problem
Grover's algorithm(格罗弗算法), 216

H

Hadamard matrix(哈达玛矩阵), 参见 Error correcting code
Halting problem(停机问题), 参见 computational problem
Hamiltonian cycle problem(哈密顿环问题)
NP-completeness(**NP**-完全性), 65
Hamiltonian path problem(哈密顿路径问题)
NP-completeness(**NP**-完全性), 53
Hardcore lemma(难度核引理), 参见Impagliazzo's hardcore lemma
Hardness(难度)
Average-case(平均难度), 375, 406
Worst-case(最坏难度), 375
Håstad **PCP** Theorem(哈斯塔德 **PCP** 定理), 474
Håstad switching Theorem(哈斯塔德开关定理)
Heuristica(启发式界), 参见 Impagliazzo's worlds
Hierarchy theorem(分层定理)
deterministic time(确定型时间分层定理), 69
non-deterministic time(非确定型时间分层定理), 69
Non-uniform(非一致分层定理), 116
probabilistic(概率型分层), 138
space(空间分层定理), 82
Hilbert's Nullstellensatz(希尔伯特零点定理), 参见 Nullstellensatz

I

Immerman-Szelepcsényi Theorem(伊默曼-史泽勒普克森奈定理), 91
Impagliazzo's hardcore lemma(因帕利亚佐难度核引理), 376
Impagliazzo's world(因帕利亚佐五界)
Algorithmica(算法界), 369
Cryptomania(密码界), 370
Heuristica(启发式界), 369
Minicrypt(迷你密码界), 370
Pessiland(悲观界), 369
Independent set problem(独立集问题), 参见 computational problem
Hardness of approximation(近似难度), 247
NP-completeness(**NP**-完全性), 51
Integer programming problem(整数规划问题), 参见 computational problem
NP-completeness(**NP**-完全性), 52

Interactive proof(交互式证明), 146
Multi-prover proof(多证明者交互式证明), 163
IP=**PSPACE**, 158
Ising model(伊辛模型), 343

K

Karp reduction(卡普归约), 参见 reduction
Karp-Lipton theorem(卡普-利普顿定理), 113
k-wise independent hash functions(k-独立哈希函数), 参见 pairwise independence

L

Ladner's theorem(拉德纳尔定理), 71
Lattices(格), 196, 233, 364
Leftover hash lemma(残余哈希引理), 参见 pairwise independence
Levin reduction(勒维归约), 参见 reduction
Linear programming problem(线性规划问题), 参见 computational problem
Log rank conjecture(对数秩猜想), 283
Logical independence(逻辑独立), 75

M

Machine Learning(机器学习), 193
Mandelbrot set(曼德勃罗集), 333
Max-cut problem(最大割问题)
NP-completeness(**NP**-完全性), 65
Metric spaces(度量空间), 527
Meyer's theorem(迈耶定理), 114, 416
Min-Max Theorem(最小-最大定理), 379
Minicrypt(迷你密码界), 参见 Impagliazzo worlds
Multi-party secure computation(多方安全计算), 192

N

Natural proofs(自然证明), 499
Negligible function(可忽略函数), 176
Nisan-Wigderson generator(尼散-维格德尔森产生器), 参见 pseudorandom generator
NP-completeness(**NP**-完全性), 42
Nullstellensatz(零点定理), 60, 333
Proof system(零点证明系统), 314

O

One-time pad(一次一密), 175
One-way function 单向函数
Rabin(拉宾单向函数), 177
RSA(RSA 单向函数), 177
Universal(通用单向函数), 178
Oracle Turing Machines(神喻图灵机), 参见 Turing Machine

P

P-computable distribution(**P**-可计算分布), 365
P-sampleable distribution (**P**-可抽样分布), 365, 369
Pairwise independence(两两独立性), 396, 514
Extension to $k>2$(推广到 $k>2$), 536
Goldreich-Levin theorem(戈德赖希-勒维定理), 184
Hash functions(哈希函数的两两独立性), 152, 354, 445
Leftover hash lemma(残余哈希引理), 445
Parallel repetition theorem(并行重复定理), 473
PATH problem(PATH 问题), 参见 computational problem
PATH problem(PATH 问题)
NL-completeness(**NL**-完全性), 89
PCP theorem(**PCP** 定理), 241, 261
Perfect matching problem(完美匹配问题), 参见 computational problem
Permanent(积和式), 166-169, 242, 323, 325, 414
#**P**-completeness(#**P**-完全性), 347-351
Pessiland(悲观界), 参见 Impagliazzo worlds
Planted clique problem(植入团问题), 参见 computational problem
Polynomial calculus proof system(多项式演算证明系统), 314
Polynomial identity testing problem(多项式恒等测试问题), 参见 computational problem
Primality testing problem(多项式恒等测试问题), 参见 computational problem
Program checking(程序检验), 164
pseudorandom functions(伪随机函数), 189, 503
pseudorandom generator(伪随机数产生器)
Derandomization(去随机化), 404
From one-way functions(从单向函数构造), 180
From one-way permutation(从单向置换构造), 180
log-space(在对数空间内构造), 449
Nisan-Wigderson(尼散-维格德尔森构造), 410
Secure(安全伪随机数产生器), 180

Uniform(一致伪随机数产生器), 413
Yao's theorem(姚期智定理), 181
PSPACE, 158

Q

Quadratic equations problem(二次方程组问题)
NP-completeness(**NP**-完全性), 65
Quadratic residuosity problem(二次剩余问题), 参见 computational problem
Quantum register(量子寄存器), 参见 Qubit
Qubit(量子位), 204, 210

R

Rabin encryption(拉宾加密), 参见 one-way function
Rabinovitch's trick(拉比诺维茨技巧), 330
Random self reducibility(随机自归约性), 166
Random subsum principle(随机子和原理), 249
Random walk(随机游走), 139, 422-426
Razborov-smolensky theorem(莱兹波偌夫-斯莫伦斯基定理), 291
Real quadratic equations problem 实变量二次方程问题
NP-completeness(**NP**-完全性), 66
Reduction(归约)
Average-cage(平均归约), 366
Cook(库克归约), 65
Karp(卡普归约), 42
Levin(勒维归约), 50, 248
log-space(对数空间归约), 88
Parsimonious(简约卡普归约), 65
randomized(随机归约), 138
Search to decision(从搜索到判定的归约), 55
Replacement product(替换乘积), 436
Resolution(归结), 309
Reversible computation(可逆计算), 211
Rotation maps(旋转映射), 434
RSA encryption(RSA 加密), 参见 one-way function

S

SAT problem(SAT 问题), 参见 computational problem
Savitch's theorem(塞维奇定理), 86
Search to decision reduction(从搜索到判定的归约), 参见 reduction
Sensitivity of functions(函数的敏感度), 266
Set cover problem(集合覆盖问题), 参见 computational problem
Set lower bound protocol(集合下界协议), 152
Shannon secrecy(香农安全性), 参见 encryption
Shor's algorithm(肖尔算法), 221
$\sum$ SAT problem($\sum$ SAT 问题), 参见 computational problem
Simon's algorithm(西蒙算法), 219
Sipser-Gács theorem(西普赛尔-高奇定理), 136
Space-Hierarchy theorem(空间分层定理), 参见 Hierarchy theorem
Straight-line program(直线程序)
Algebraic(代数直线程序), 319
Boolean(布尔直线程序), 107
Subset sum problem(子集和问题)
NP-completeness(**NP**-完全性), 65
Pseudo-polynomial time algorithm(伪多项式时间算法), 66
Sumcheck protocol(子和验证协议), 159

T

Tautology problem(永真式问题), 参见 computational problem
Tensor product(张量积), 251, 284, 435
Time constructible functions(时间可构造函数), 16
Time/space tradeoff for SAT(SAT 的时空平衡), 101
Toda's theorem(户田定理), 352
TQBF problem(TQBF 问题), 参见 computational problem
TQBF problem(TQBF 问题), 84
Traveling salesperson problem(旅行商问题), 参见 computational problem
Turing Machine(图灵机), 12
Advice(建言图灵机), 112
Alternating(交错图灵机), 99
Blum-Shub-Smale model(布卢姆-舒布-斯梅尔模型), 331
Non-deterministic(非确定型图灵机), 41
Non-deterministic universal(非确定型通用图灵机), 64
Oblivious(散漫图灵机), 18
Oracle(神喻图灵机), 72
Probabilistic(概率型图灵机), 125
Single-tape(单带图灵机), 17
Space-bounded(空间受限图灵机), 78
Universal(通用图灵机), 20
Space-bounded Universal(空间受限通用图灵机), 93

U

Unary languages(一元语言)，66
Unique games conjecture(唯一性游戏猜想)，490
Unique SAT(唯一性 SAT)，354
Unitary matrices(酉阵)，209
UPATH problem(UPATH 问题)，参见 computational problem

V

Valiant-Vazirani theorem(瓦利安特-瓦兹拉尼定理)，217，354
Vertex cover problem 顶点覆盖问题
  2-approximation(2-近似)，67
  Hardness of approximation(近似难度)，247
  NP-completeness(**NP** 完全性)，65

W

Weak random sources(弱随机源)，133，442
Worst-case Hardness(最坏难度)，参见 hardness

X

XOR lemma(XOR 引理)，参见 Yao's XOR lemma

Y

Yao's min-max lemma(姚期智最小-最大引理)，265
Yao's XOR lemma(姚期智 XOR 引理)，375

Z

Zero knowledge(零知识)，187
Zig-zag product(拉链乘积)，440

复杂性类索引

索引中的页码为英文原书页码，与书中页边标注的页码一致。

⊕P，352
=P，344
  Approximation(近似)，351
  Completeness(完全性)，346
AC，118
AC^0，118，286
$AlgNP_{poly}$，323
$AlgP_{poly}$，323
ACC0，291
$ACC0(m_1, \cdots, m_k)$，291
AC^i，118
AM，150
AM，[k]，150
AP，100
BPL，139
BPP，125
  Alternative definition(交错定义)，126
  Completeness(完全性)，17
  Error reduction(错率归约)，132
BPTIME(T(n))，125
BQP，213
coNP，55
  Completeness(完全性)，56
DTIME(T(n))，25
DTIME(T(n))/a(n)，112
EXP，41，56
FP，344
IP，147
  Error reduction(错率归约)，147
IP，[k]，147
L，81，440
MA，150
MIP，163
NC，117
NEXP，56，164，243
NL，81
  Alternative(交错的)，90
NP，39
  Alternative definition(交错定义)，41
  Philosophy(哲学意义)，57
NP_C，332
NPSPACE，81
NTIME(T(n))，41
P，25
  completeness(完全性)，118
  Philosophy(哲学意义)，26
P_C，332
PCP(q(n)，r(n))，241
PH，97，119，136，352
  completeness(完全性)，98
  Oracle-based definition(基于神喻的定义)，102
PP，345
PSPACE，81，230
  completeness(完全性)，83
RL，139，449
RP，131
RTIME(T(n))，131
Σ_2^p，97
SIZE(T(n))，108
SPACE(S(n))，79
TISP(T(n)，S(n))，101
ZPP，131
ZTIME(T(n))，131

推荐阅读



作者：Thomas H. Cormen 等著
书号：978-7-111-40701-0，128.00元



作者：Brian W. Kernighan 等著
书号：978-7-111-12806-0，30.00元



作者：Y.Daniel Liang 著
书号：978-7-111-50990-4，85.00元



作者：Randal E. Bryant 等著
书号：978-7-111-32133-0，99.00元



作者：David A. Patterson，John L. Hennessy
中文版：978-7-111-50482-5，99.00元



作者：James F. Kurose 等著
书号：978-7-111-45373-8，79.00元



作者：Abraham Silberschatz 著
中文翻译版：978-7-111-37529-8，99.00元
本科教学版：978-7-111-40660-1，59.00元



作者：Jiawei Han 等著
中文版：978-7-111-39140-1，79.00元



作者：Ian H. Witten 等著
中文版：978-7-111-45381-6，79.00元

推荐阅读





自动机理论、语言和计算导论（原书第3版）

作者：John E. Hopcroft 等 译者：孙家骕 等 书号：7-111-24035-8 定价：49.00元

本书是关于形式语言、自动机理论和计算复杂性方面的经典教材，是三位理论计算大师的巅峰之作。书中涵盖了有穷自动机、正则表达式与语言、正则语言的性质、上下文无关文法及上下文无关语言、下推自动机、上下文无关语言的性质、图灵机、不可判定性以及难解问题等内容。

本书已被世界许多著名大学采用为计算机理论课程的教材或教学参考书，适合作为国内高校计算机专业高年级本科生或研究生的教材，还可供从事理论计算工作的研究人员参考。

计算理论导引（原书第3版）

作者：Michael Sipser 译者：段磊 唐常杰 等 书号：978-7-111-49971-8 定价：69.00元

本书由计算理论领域的知名权威Michael Sipser所撰写。他以独特的视角，系统地介绍了计算理论的三个主要内容：自动机与语言、可计算性理论和计算复杂性理论。作者以清新的笔触、生动的语言给出了直达的数学原理，而没有拘泥于某些低层次的细节。在证明之前，均有“证明思路”，帮助读者理解数学形式下蕴涵的概念。本书可作为计算机专业高年级本科生和研究生的教材，也可作为教师和研究人员的参考书。