HZ BOOKS
华章教育

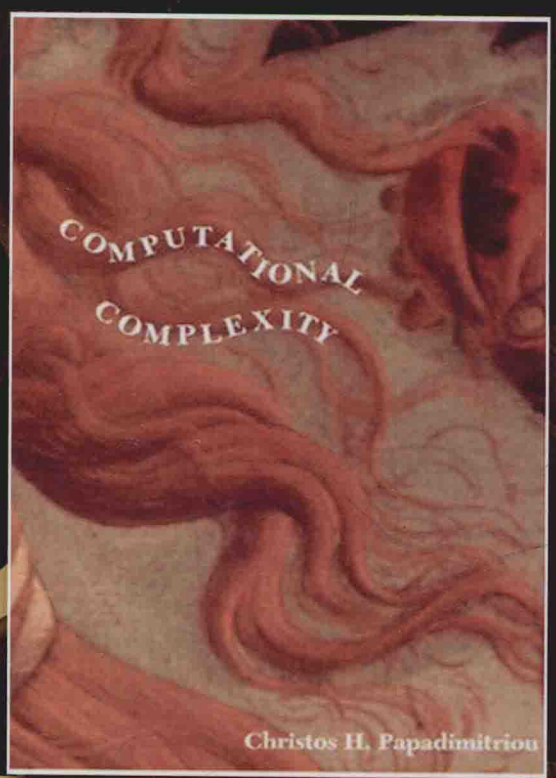
计 算 机 科 学 丛 书

PEARSON

计算复杂性

[美] 克里斯特斯 H. 帕帕季米特里乌（Christos H. Papadimitriou） 著
朱洪 彭超 卜天明 等译

Computational Complexity



机械工业出版社
China Machine Press

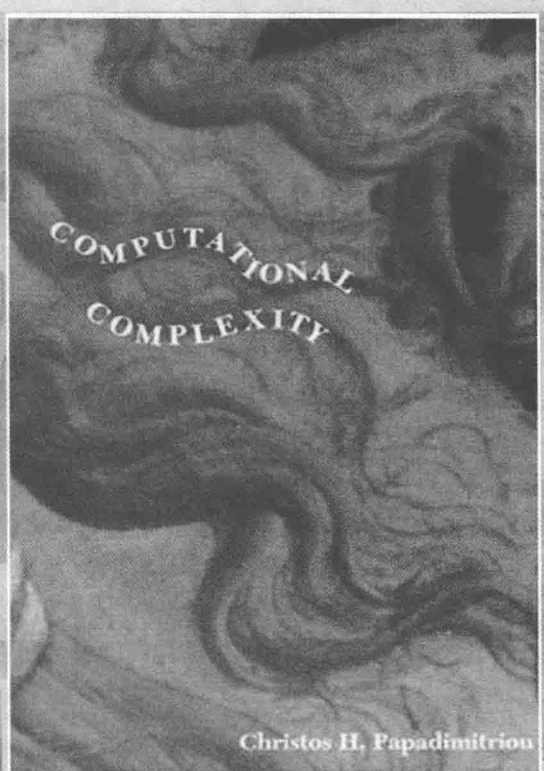
计 算 机 科 学 丛 书

计算复杂性

[美] 克里斯特斯 H. 帕帕季米特里乌（Christos H. Papadimitriou） 著

朱洪 彭超 卜天明 等译

Computational Complexity



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

计算复杂性 /（美）帕帕季米特里乌（Papadimitriou, C. H.）著，朱洪等译．—北京：机械工业出版社，2015.10
（计算机科学丛书）
书名原文：Computational Complexity

ISBN 978-7-111-51735-1

I. 计… II. ① 帕… ② 朱… III. 计算复杂性 IV. TP301.5

中国版本图书馆 CIP 数据核字（2015）第 237779 号

本书版权登记号：图字：01-2015-0263

计算复杂性是计算机科学中思考为什么有些问题用计算机难以解决的领域，是理论计算机科学研究的重要内容。本书重点介绍复杂性的计算、问题和逻辑。本书包含五部分：第一部分介绍算法，包括问题与算法、图灵机、不可判定性；第二部分介绍逻辑学，包括布尔逻辑、一阶逻辑和逻辑中的不可判定性；第三部分介绍 **P** 和 **NP**，包括复杂性类之间的关系、归约和完备性、**NP** 完全问题、**coNP** 和函数问题、随机计算、密码学、可近似性等；第四部分介绍 **P** 内部的计算复杂性，包括并行计算、对数空间；第五部分介绍 **NP** 之外的计算复杂性类，包括多项式谱系、计数的计算、多项式空间等。

本书不需要数学预备知识。每章最后一节包括相关的注解、参考文献和问题，很多问题涉及更深的结论和研究。本书适合于作为高年级本科生和低年级研究生计算复杂性理论课程的教材。

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：盛思源　　责任校对：殷　虹
印　　刷：中国电影出版社印刷厂　　版　　次：2016 年 1 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：21.25
书　　号：ISBN 978-7-111-51735-1　　定　　价：119.00 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

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

华章科技图书出版中心

译者序

Computational Complexity

经过多年努力，本书终于翻译完成了。作者 Papadimitriou 为著名的理论计算机科学家，对计算复杂性有着比较全面而深刻的见解，因此本书内容也颇为深奥。在长期的翻译和学习中，我们体会到该书对于从事计算理论研究的相关人员来说确实是一本优秀的基础参考书，认真学习本书的读者必能受益匪浅。我们在翻译过程中也是收获良多，补习了不少原先忽略的知识。本书总结了截至 1994 年计算理论领域中的最新研究成果，其中有概率可验证证明（PCP）、零知识证明（Zeroknowledge proof system）等当时极其前沿并且对日后的学科发展起到关键性作用的内容，本书对这些专门知识给予了简洁的阐释。

读者通过本书的阅读和习题，可以对 20 世纪 90 年代中期的计算理论发展脉络有一个比较清楚的了解，从而打下良好的研究基础。但是，由于目前已经是 21 世纪 10 年代了，而计算理论在最近 20 多年里又有了蓬勃的发展，所以译者建议想进一步了解计算理论的读者可以补充阅读以下书籍：

- Sanjeev Arora and Boaz Barak. Computational Complexity: A Modern Approach. Cambridge University Press, 2009.
- Oded Goldreich. Computational Complexity: A Conceptual Perspective. Cambridge University Press, 2008.
- Mikhail J. Atallah and Marina Blanton. Algorithms and Theory of Computation Handbook. CRC Press, 2009.

对于 NP 难优化问题的近似算法和难近似性，译者推荐补充阅读以下两本书：

- Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela and Marco Protasi. Complexity and Approximation: Combinatorial Optimization Problems and Their Approximabiltiy Properties. Springer, 2003.
- Vijay V. Vazirani. Approximation Algorithm. Springer, 2001.

近 30 年来，量子计算、量子复杂性和算法得到长足的发展，并且具有远大的潜力，译者推荐如下两本书和一篇综述：

- Michael A. Nielson and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2000.
- Jozef Gruska. Quantum Computing, McGraw-Hill, New York, 1999.
- Lance Fortnow. One Complexity Theorist's View of Quantum Computing. Theoretical Computer Science, Vol 292, pp. 597-610, 2003.

在参数复杂性和算法机制设计方面，译者推荐两篇代表性综述：

- Rod Downey. Basic Parametric Complexity II: Intractability. Victoria University. Wellington. Isaac Newton Institute, Cambridge, LATA, March 2012.
- Noam Nisan and Amir Ronen. Algorithmic mechanism design. Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 129-140, 1999.

最后，作为补充阅读，译者还推荐以下两本中文书籍：

- 堵丁柱，葛可一，王洁．计算复杂性导论．高等教育出版社，2002.
- 堵丁柱，葛可一，胡晓东．近似算法的设计与分析．高等教育出版社，2011.

全书译者分工如下：前言和第 1、2、3 章由倪盛宇和朱洪翻译，第 4、5、6 章由彭超翻译，第 7、8、11、12、14、20 章由朱洪翻译，第 9 章由王怡慧翻译，第 10、13 章由陈崇琛翻译，第 15、16、17、18 章由卜天明翻译，第 19 章由卜天明、陈崇琛、王怡慧、朱洪各译一节。由于译者较多，译稿难免有不统一和不准确之处，欢迎读者随时通过电子邮件 hzhu@fudan.edu.cn 提出宝贵意见。

译者

前　言

Computational Complexity

我仅仅希望简单叙述
请赋予我这一特权
因为我们已经被灌输了带有这么多音乐的歌声
音乐正在沉沦
而我们的艺术变得如此矫饰
以至于装饰品已经腐蚀了她的容颜
是时候说一些简单的语言了
因为明天我们的心灵将起帆远航
——Giorgos Seferis

本书适合作为低年级研究生或者高年级本科生学习计算复杂性理论的教材。计算复杂性是计算机科学中思考为什么有些问题用计算机难以解决的领域。这个领域以前几乎不存在，而现在却迅速扩展，并构成了理论计算机科学研究活动的主要内容。现在没有一本书可以全面介绍复杂性——当然也包括这本书在内。本书只是包含了我认为可以清楚和相对简单地表示的结果以及在我看来是复杂性领域的中心内容。

我认为复杂性是计算（复杂性类）和应用（问题）之间复杂而核心的部分。开篇就向读者灌输这一观点有点为时过早，不过我还是要冒险一试，而且这也将是全书 20 章中反复强调的观点。完全性的结论明显是这一进展的中心环节。逻辑也是如此，它能很好地表达和抓住计算这一概念，是非常重要的应用。因此计算、问题和逻辑是贯穿本书的三大主脉络。

内容

快速浏览目录，第 1 章介绍问题和算法——因为当复杂性与简单性比较时，复杂性最好理解。第 2 章讨论图灵机，同时明确我们的方式将不依赖于机器。第 3 章介绍非确定性（它不仅是复杂性的最高形式，而且还具有重大的方法学影响）。

接着讨论逻辑。这一部分最可能会被复杂性理论同行视为另类。但是它对于我看待复杂性的观点非常重要，对于计算机科学非常基本，又很少作为走向计算机科学家的成功之路看待，所以我感到我必须做一次尝试。第 4 章介绍布尔逻辑（包括 Horn 子句的算法属性，以及布尔电路和香农定理）。第 5 章介绍一阶逻辑及其模型论和证明论，还包括完全性定理，以及足够的二阶逻辑以引出随后的 NP 的 Fagin 特征——非常有用但是往往被忽视，其意义相当于 Cook 定理。第 6 章是对 Gödel 不完全性定理的独立证明，该证明是逻辑表达计算早期的重要例子。

然后重点介绍复杂性。第 7 章介绍已知的复杂性类之间的关系——包括 Savitch 和 Immerman-Szelepscényi 关于空间复杂性的定理。第 8 章介绍归约和完全性概念，紧接着，

作为例子，介绍 Cook 定理和电路值问题的 P 完全性，同时比较用逻辑表示 **P** 和 **NP** 的特征。第 9 章包含很多 **NP** 完全的结果，同时介绍各种证明方法。第 10 章讨论 **coNP** 和函数问题。第 11 章介绍随机算法、与之对应的复杂性类以及用现实随机源的实现方法。电路和它们与复杂性、随机化的关系也在此介绍。第 12 章很简短，粗略介绍密码学和协议。第 13 章讨论近似算法，以及最近通过概率可验证性证明得出的一些不可行性方面的结果。另一方面，第 14 章讨论 $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ 问题的结构性方面，比如，中间度、同构、稠密性和谕示。它还包含了 Razborov 关于单调电路的下界证明。

第 15 章进一步关注 **P**、并行算法及其复杂性，第 16 章重点讨论对数空间，包括无向图路径的随机游走算法。最后，除了 **NP** 以外，第 17 章给出多项式谱系（包括优化问题的 Krentel 特征）；第 18 章讲述计数问题和关于积和式的 Valiant 定理；第 19 章介绍多项式空间的许多方面（最有趣的是关于交互式协议的 Shamir 定理）；本书最后对难解性领域做了简短展望。

本书并没有特别的数学基础要求——除了要有一定程度的“数学成熟度”，而数学成熟度这个名词，一般不在序言中给予定义。所有的定理都从基本原理给予证明（除了第 13 章关于近似性引用了两个定理外），同时更多的相关结果在每章最后一节中说明。证明和构造经常会比文献里讲述的简单得多。实际上，本书包含了多个与复杂性相关的主题或专题简介：基础数论（用来证明 Pratt 定理），Solovay-Strassen 素数测定和 RSA 密码协议（第 10、11、12 章）；基础概率（第 11 章和其他章节）；组合数学和概率方法（第 10、13、14 章）；递归理论（第 3、14 章）；逻辑（第 4、5、6 章）。由于复杂性问题总是和相对应的算法概念的全面发展联系在一起（第 1 章的有效算法，第 11 章的随机算法，第 13 章的近似算法和第 15 章的并行算法），所以本书也可以作为算法引论——虽然仅仅粗略分析，但是可以应用在各种情况。

注解和问题

每章的最后一节包含了相关的文献、注解、练习和问题。很多问题涉及更深的结论和课题。就我看来，这是一章中最重要的部分（经常也是最长的），读者应该将它作为本书的一部分来阅读。它经常给出历史观点，并把该章放到了更广泛的领域中。所有这些题目都是可做的，至少在提示下去图书馆查阅答案（我已经发现这样做至少对我的学生来说，不亚于另一次智商测验）。对这些题目没有标记难易，不过对于真正的难题还是给出了警示标记。

教学

本书的重点显然是复杂性，所以我们将它设计成（以及用作）计算机科学家关于计算理论的入门级读物。我和我的同事在过去的三年中用它作为加州大学圣地亚哥分校硕士研究生第一年为期 10 周的教材。前两周学习前 4 章，这些内容对于本科生来说，一般都已熟悉。逻辑学安排在紧接着的 3 周中，经常省略完全性证明。剩下的 5 周学习第 7 章，作为 NP 完全性的严格训练（不包括在该校的算法课内），选择第 11～14 章中的一两节。一学

期的课程可以涵盖以上 4 章。如果你想跳过逻辑学部分，可以加上第 15 章（然而，我相信这样做会错过本书相当好的一部分内容）。

本书至少还可以用于两门课程：前 9 章的主题对于计算机科学家很关键，所以它可以自豪地替代高年级本科生初级理论课程中的自动机和形式语言（特别是，因为现在的编译课程都已独立出来）。我也两次使用后面的 11 章作为理论方向的第二学期课程，其目标是带领有兴趣的研究生进入复杂性的研究课题——或者至少帮助他们成为计算机理论会议上见多识广的听众。

感谢

我关于复杂性的想法是我的老师、学生和同事长期鼓舞和启迪的结果。我非常感谢所有这些人：Ken Steiglitz、Jeff Ullman、Dick Karp、Harry Lewis、John Tsitsiklis、Don Knuth、Steve Vavasis、Jack Edmonds、Albert Meyer、Gary Miller、Patrick Dymond、Paris Kanellakis、David Johnson、Elias Koutsoupias（他也在图表、最后检查和索引上给予我很多帮助）、Umesh Vazirani、Ken Arrow、Russell Impagliazzo、Sam Buss、Milena Mihail、Vijay Vazirani、Paul Spirakis、Pierluigi Crescenzi、Noga Alon、Stathis Zachos、Heather Woll、Phokion Kolaitis、Neil Immerman、Pete Veinott、Joan Feigenbaum、Lefteris Kirousis、Deng Xiaotie、Foto Afrati、Richard Anderson，最主要的是 Mihalis Yannakakis 和 Mike Sipser。他们阅读了本书的草稿并提出了建设性意见、想法和建议——否则就会让我为他们的沉默而紧张。在所有对我的课件提出评论的学生中，我记得名字的只有 David Morgenthaller、Goran Gogic、Markus Jacobsson 和 George Xylomenos（但我记住了其余人的笑容）。最后，感谢 Richard Beigel、Matt Wong、Wenhong Zhu 和他们在耶鲁的复杂性班，他们找出了本书初稿中的许多错误。自然，我对剩下的错误负责——尽管我认为我的朋友当初可以找出更多的错误。

我非常感激 Martha Sideri 的鼓励和支持，以及她的注解、看法和封面设计。

我在加州大学圣地亚哥分校工作时完成本书，但这期间我也访问了 AT&T 公司的贝尔实验室、Bonn 大学、Saarbrücken 的 Max-Planck 研究所、Patras 大学和那里的计算机学院以及巴黎 Sud 大学。我对于算法和复杂性的研究受到美国国家科学基金、Esprit 项目 AlCOM 以及加州大学圣地亚哥分校信息和计算机科学主席 Irwin Mark 和 Joan Klein Jacobs 的资助。

与 Addison-Wesley 的 Tom Stone 及其同事一起完成本书出版是愉快的。最后，我使用了 Don Knuth 的 TeX 排版，我的宏是从 Jeff Ullman 很多年前给我的那些中演变而来的。

Christos H. Papadimitriou

目 录

Computational Complexity

第四部分 P 内部的计算复杂性类

第五部分 NP 之外的计算复杂性类

第一部分
Computational Complexity

算　法

算法书都是以复杂性作为最后一章结束的，反之本书以复述算法的某些基本方面作为开始。前3章旨在厘清几个简单的要点：计算问题不仅仅是有待解决的对象，它们本身就是值得研究的课题。问题和算法可以通过数学方法形式化和分析（比如，分别对应于语言和图灵机），而精确的形式化并不重要。多项式时间计算是计算问题中一个重要的期望属性，和直观意义上的实际可解决性联系在一起。许多不同模式的计算可以在多项式效率损失的情况下相互模拟——非确定性是唯一的例外，它看起来需要指数时间来模拟。还有一些问题根本没有算法，无论怎样总是束手无策。

第1章
Computational Complexity

问题与算法

算法是解决问题的一步步详细的方法。但是问题是什么？这一章介绍三个重要例子。

1.1 图的可达性问题

设图 $G=(V,E)$ 由有穷结点集 V 和边集 E 构成，它是结点对集合（如图 1.1 所示，图都是有向有穷图）。许多计算问题都是关于图的。图的最基本问题是：给定图 G 和两个结点 1，$n\in V$，是否存在从 1 到 n 的通路？我们称这为 REACHABILITY（可达性）[⊖]。例如，图 1.1 确实有一条从 1 到 $n=5$ 的通路，即 (1，4，3，5)。如果我们把边 (4，3) 的方向反过来，就不存在这样的通路了。

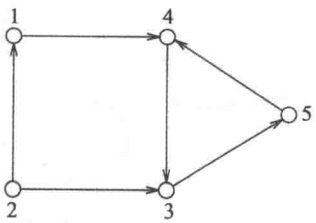


图 1.1 图

像大多数有趣的问题一样，REACHABILITY 有着无穷多的实例。每一个实例都是一个数学对象（在该例子中，是一个图和它的两个结点），我们对此提问并期待回答。提出的特定问题是这类问题的特征。注意，REACHABILITY 提出的问题只要求“是”或者“否”这样的回答。这样的问题称为判定问题。在复杂性理论上，我们通常发现只考虑判定问题而不需要各种形式答案的问题非常方便。所以判定性问题在本书扮演着很重要的角色。

我们对于解决问题的算法感兴趣。下一章将介绍图灵机，一种表达任意算法的形式化模型。目前，让我们粗略地描述算法。例如，REACHABILITY 可以被所谓的搜索算法解决。这个算法按照如下方式工作：整个算法中，我们维护一个结点集 S，初始化 $S=\{1\}$。每个结点或者有标记或者没有被标记。结点 i 的标记表示 i 曾经（或者当前）属于 S。开始时，只有结点 1 有标记。在算法循环的每一步，我们选择一个结点 $i\in S$，并且将它从 S 中移除。然后我们逐一考察从 i 出去的每一边 (i,j)。如果结点 j 没有标记，那么我们将它标记，并放入集合 S。这个过程继续直到 S 为空。最后，如果结点 n 被标记，我们回答“是”；否则，就是“否”。

显然这个熟悉的算法解决了 REACHABILITY。需要证明的是，某个结点被标记当且仅当从结点 1 到它存在通路。两个方向都可以容易地用归纳证明（见问题 1.4.2）。然而，我们的描述也显然漏掉了一些重要的细节。比如，作为算法的输入，图如何表示？既然恰当的表示方法依赖于特定的算法模型，这就将等到我们有了特定模型才能确定。这一讨论（见 2.2 节）的关键在于表达方式并不重要。你可以假定此时图已经用邻接矩阵（见图 1.2）给出，所有的元素都可以被算法随机获取[⊜]。

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$
图 1.2 邻接矩阵

⊖ 在复杂性理论中，计算问题不仅是解答问题，而且问题本身就是我们要关注的数学对象。当问题被当作数学对象处理时，它们的名字用大写字母表示。

⊜ 实际上，在第 7 章，我们将看到 REACHABILITY 在复杂性理论中一些重要的应用，在那里，图将间接地给出。也就是说，邻接矩阵的每一元素可以从输入数据中计算出来。

算法本身也有不明确的地方：在 S 中怎样选择元素 $i\in S$？选择的方式可能严重影响搜索的风格。比如，如果总是选择待在 S 中最久的结点（换句话说，我们作为队列实现 S），那么搜索结果是广度优先的，得到的是最短路径。如果 S 按照栈来维护（我们选择最后加入的结点），那就是深度优先搜索。其他维护 S 的方式会导致完全不同的搜索。但是，对所有选择该算法都是正确的。

而且，该算法还是有效的。为了说明这一点，注意，当每行对应的顶点被选中的时候，矩阵的每个元素只被访问一次。因此，我们最多进行 n^2 次操作来处理那些选中顶点的边（毕竟，图最多只有 n^2 条边）。假定其他所需要的简单操作（从 S 集合中选择元素，标记顶点，判断顶点是否有标记）可以在恒定时间内完成，我们可以说，该搜索算法判断图中两结点是否连通的时间最多正比例于 n^2，或者说，$\mathcal{O}(n^2)$。

刚才用到的 $\mathcal{O}$ 记号以及相关记号，在复杂性理论中非常有用，所以这里插入一段来形式化地定义它们。

定义 1.1 记 $\boldsymbol{N}$ 为非负整数集合。在复杂性理论中，我们关注的是 $\boldsymbol{N}$ 到 $\boldsymbol{N}$ 的函数，像 n^2、2^n 和 n^3-2n+5。我们使用字母 n 作为函数的标准变量。即便函数的值是非整数或者负数（如$\sqrt{n}$、$\log n$ 和$\sqrt{n}-4\log^2 n$）我们将始终考虑非负整数。也就是说，这些例子中任何记为 $f(n)$ 的函数实际上表示 $\max\{\lceil f(n)\rceil,0\}$。

所以，令 f 和 g 是 $\boldsymbol{N}$ 到 $\boldsymbol{N}$ 的函数。我们记为 $f(n)=\mathcal{O}(g(n))$（读作“$f(n)$ 是大 O$g(n)$”），如果存在正整数 c 和 n_0，使得对于所有的 $n\geqslant n_0$，$f(n)\leqslant c\cdot g(n)$。$f(n)=\mathcal{O}(g(n))$ 通俗的意思是 f 增长得像 g 一样或者更慢。如果正好相反，则我们记为 $f(n)=\Omega(g(n))$，即如果 $g(n)=\mathcal{O}(f(n))$。最后 $f(n)=\Theta(g(n))$ 的意思是 $f(n)=\mathcal{O}(g(n))$ 并且 $f(n)=\Omega(g(n))$，后者表示的是 f 和 g 有着相同的增长率。

比如，非常容易看出，如果 $p(n)$ 是一个 d 阶多项式，那么 $p(n)=\Theta(n^d)$。也就是说，多项式的增长率由多项式第一个非零项决定。或许复杂性理论最重要和有用的事实为：如果 $c>1$ 是整数，$p(n)$ 是多项式，那么 $p(n)=\mathcal{O}(c^n)$，但是 $p(n)=\Omega(c^n)$ 并不正确。也就是说，*任何多项式增长严格慢于任何指数函数*（证明见问题 1.4.9）。降低指数，这一性质同样意指 $\log n=\mathcal{O}(n)$ ——实际上，$\log^k n=\mathcal{O}(n)$ 对于任意 k 都成立。 □

4
~
5

多项式时间算法

回到 REACHABILITY 的 $\mathcal{O}(n^2)$ 算法，不用吃惊这个简单问题被这个简单算法圆满解决——事实上，我们对 $\mathcal{O}(n^2)$ 的估计还悲观了，尽管这一点在此并不重要，见问题 1.4.3。然而，明确我们满意之处是重要的：时间增长率是 $\mathcal{O}(n^2)$。在本书中，多项式增长率视为可接受的时间要求，它作为问题满意解决的标志。相反，指数增长率 2^n，或者更糟糕的 $n!$ 就需要注意了。如果尝试了一个又一个算法，问题始终不能在多项式时间内解决，我们通常认为这个问题是难解的，不可能有实际有效的算法。那么本书介绍的方法就起作用了。

多项式和非多项式时限性的分划线（dichotomy），以及认为多项式算法即是直观意义上的“实际可行计算”是有争议的。存在着非多项式的有效算法和实践上低效的多项式算法[⊖]。比

⊖ 实际上，线性规划这一重要例子提供了上述的两种反例（见 9.5.34 节的讨论和参考文献）。这个基本问题的一个广泛使用的经典算法是*单纯形法*，最坏情况下是指数，但实际运行中性能很好。事实上，它的期望性能被证明是多项式时间的。与此相反，问题的第一个多项式算法，*椭球算法*，缓慢得不切实际。但是线性规划的例子事实上支持（而不是反对）复杂性理论的方法学：看起来可能暗示了实际解决的问题确实是多项式时间的——尽管多项式时间算法和经验上的好算法不必然一致。

如，一个 n^{80} 的算法可能只有有限的实际价值，而一个指数增长的算法（如 $2^{\frac{n}{100}}$）（或者更有意思的是，亚指数函数，如 $n^{\log n}$）将有用得多。

然而，强有力的论据偏向于多项式范式。首先，多项式增长率最终会优于指数增长
6 率，所以后者只会在问题的有限实例中作为首选项——当然，有穷结点集可能把实际出现的例子都包含进去，或者在我们宇宙的界限内……更重要的是，经验表明那种极端的增长率算法，如 n^{80} 和 $2^{\frac{n}{100}}$，在实际中很少发生。多项式算法一般有小的指数和合理的系数，而指数算法通常不现实。

另一个潜在的批评是，我们的观点只考察最坏情况下的算法性能。最坏情况下指数的算法性能可能对应输入数据中很小的一小部分，而平均情况下性能可能是满意的。当然，与最坏情况相反，分析算法的期望值，会给算法的性态带来更多信息。遗憾的是，在实践中我们很少知道问题的输入概率分布（也就是说，各种可能出现的实例作为输入的概率有多大）因此，不可能有真实情况的平均情况分析。另外，如果我们想解决某个特定实例，算法表现糟糕，那么知道我们遇到了小概率的例外情况对我们没有帮助和安慰[⊖]。

当我们选择多项式算法作为数学概念来代表非形式化意义下的“实际上有效的计算”时，不用惊讶所面临的种种批评。在数学领域，任何尝试用数学概念（如 C_∞）来表达直观意义下的实际想法（像实分析中的“光滑函数”）都会包括某些不想要的东西，而把一些有理由应包含在内的东西排除在外。最终，我们选择的论据只能是：*采用多项式最坏情况下的性能作为我们有效性的评价标准成就了出色而有用的理论：它抓住了实际计算的内涵，没有这种简化就不可能成功。*实际上，多项式有着不少数学上的方便：它们形成了一个稳定的函数类，在加法、乘法和左右代入（见问题 7.4.4）下不变。此外，对多项式函数取对数只相差一个常数（即 $\Theta(\log n)$），这一点在某些情况下比较方便，比如，当我们讨论空间约束时。

由此我们考虑解决 REACHABILITY 搜索算法的空间要求。算法需要存储集合 S 和
7 每个结点的“标记”。因为最多有 n 个标记，S 不会大于 n，所以算法使用 $\mathcal{O}(n)$ 空间。我们应该庆贺吗？就空间而言，我们倾向于比时间更为严格——有些时候也确实可以节约许多空间。我们将在 7.3 节看到另一个 REACHABILITY 算法，不同于深度优先搜索和广度优先搜索（它可以称为*中间优先搜索*），所用的空间大大少于 n，特别是，它使用了 $\mathcal{O}(\log^2 n)$ 空间。

1.2　最大流问题

下一个计算问题的例子是 REACHABILITY 的一般化。对应地，输入是称为网络的一种一般化图形（见图 1.3）。网络 $N=(V,E,s,t,c)$ 是有两个特殊结点 s（称为源）和 t（称为汇）的图 (V,E)。假定源没有入边，汇没有出边。而且每一条边 (i,j) 都对应一个正整数，称为容量 $c(i,j)$。N 的流是对于每一条边 (i,j) 指定的一个非负整数值 $f(i,j)\leqslant c(i,j)$，

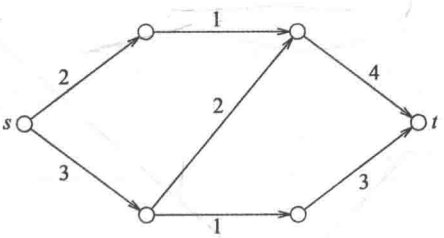


图 1.3　网络

⊖ 在第 11 章以及其他地方，我们会学到概率算法，但是随机源是算法的一部分，而不是输入随机。问题 12.3.10 给出了一种从复杂性理论角度处理平均性能的有趣方法。

且满足除了 s、t 外，其余结点的出边的 f 的和等于入边的 f 的和。流 f 的值定义为离开 s 的边的流之和（或者到达 t 的流之和，通过将所有结点的流守恒方程相加后，可知这两个量是相等的）。MAX FLOW（最大流）问题是：求给定网络 N，找出最大可能的流值。

MAX FLOW 明显不是判定性问题，因为它要求答案的信息量远多于“是”或者“否”。它是优化问题，因为它在诸多可能的解中，根据简单的代价准则，寻求最佳方案。然而，优化问题可以通过提供目标值，询问这个值是否可以取到，大致等价地转换为判定性问题。比如，在 MAX FLOW 的判定性版本中，记为 MAX FLOW(D)，给定网络 N 和整数 K（目标），问是否存在大于或者等于 K 的流。这个问题基本等价于 MAX FLOW，如果我们找到 MAX FLOW 的多项式时间算法，MAX FLOW(D) 也解决了；反过来也一样（后者，稍微不明显，可以用二分搜索得到结果，见例 10.4）。 8

解决 MAX FLOW 的多项式时间算法基于一个关于网络的基本事实，称为最大流最小割定理（同时该算法也提供了该定理的证明，见问题 1.4.11）。算法如下：假定我们已经有一个流 f，我们只想知道它是不是最优的。如果存在一个值大于 f 的流 f'，那么 $\Delta f = f' - f$ 就是一个正值流。Δf 的唯一问题就是在某些边 (i,j) 上出现负值。但是，这样的负值可以看成是边 (j,i) 上的正值流。这样的“逆流”必须至多等于 $f(i,j)$。另一方面，正分量 Δf 最多为 $c(i,j)-f(i,j)$。

另一种表述方法可以说，Δf 是派生网络 $N(f)=(V,E',s,t,c')$ 的流，这里 $E'=E-\{(i,j):f(i,j)=c(i,j)\}\cup\{(i,j):(j,i)\in E,\ f(j,i)>0\}$，当 $(i,j)\in E$ 时 $c'(i,j)=c(i,j)-f(i,j)$，当 $(i,j)\in E'-E$ 时 $c'(i,j)=f(j,i)$[⊖]。比如，对于图 1.3 中的网络和图 1.4a 中的流，$N(f)$ 如图 1.4b 所示。所以，判断 f 是否最优，等价于判定 $N(f)$ 中是否有正值流。而我们知道怎样来做：在正容量网络中求正值流就是求是否存在 s 到 t 的路径：REACHABILITY 的一个实例！

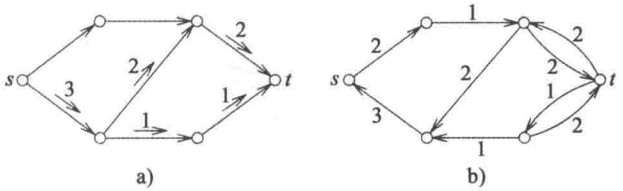


图 1.4 网络 $N(f)$ 的构造 9

因此，我们可以判断流是否已经最大。这就有了以下 MAX FLOW（最大流）的算法：从 N 中每个地方都是零值的流开始。重复构造 $N(f)$，并在 $N(f)$ 中寻找一条 s 到 t 的路径。如果路径存在，找出路径上沿着边的最小容量 c'，然后把这个流量的值加到出现在该路径上的所有边的 f 值上。结果很清楚是一个更大的流。当没有这样的路径时，f 就是最大流，算法结束。

这个算法需要花多少时间求解 MAX FLOW？算法的每次迭代（寻找路径并增加沿着该路径的流）花费 $O(n^2)$ 的时间，如上一节所述。而且最多有 nC 次循环，这里 C 表示网络中边的最大容量。原因如下：每次迭代流都会至少增加 1（记住，容量是整数），最大流不会超过 nC（从源最多有 n 条边出来）。所以最多有 nC 次循环。实际上，如图 1.5 所示，

⊖ 这里假设 N 没有互反的有向边对；否则，$N(f)$ 的定义会复杂一些。

现实结果几乎就这么糟糕：如果 REACHABILITY 的算法每次都返回一条包含边 (i,j) 或者 (j,i) 的路径，那么需要 $2C$ 次迭代。因此这个算法的总时间为 $\mathcal{O}(n^3C)$。

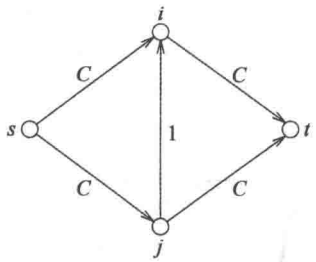


图 1.5　一个坏例子

尽管时间界有着完美的多项式形式，但颇有些令人不愉快。算法的运行时间线性地依赖于 C 要引起警觉。实例中的数字通过输入方很少的工作量就会变得很大（只是添加一些零）。也就是说，数字通常用二进制（或者十进制）表示，如此表示的长度是被表示数的对数。从真正意义而言，上述的界（以及算法）不属于多项式，因为时间按照输入长度的指数函数增长。

有简单而优雅的方法可以克服这个困难：假设我们每次增广的流不是 s 到 t 的任意路径，而是最短路径，也就是包括最少边的一条路径。我们可以通过搜索算法的广度优先版本来得到（注意这个策略如何轻易克服图 1.5 中的“坏例子”）。在 $N(f)$ 中一条 s 到 t 的路径上有最小容量 c' 的边称为瓶颈。不难证明，如果我们总是选择最短路径来增广流，那么 N 的每条边 (i,j)，或者其反向边（$N(f)$ 中每条边或者是 N 的边或者是它的反向边），最多 n 次迭代成为瓶颈（见问题 1.4.12）。因为最多有 n^2 条边，每次循环最少有一个瓶颈，所以最短路径版本算法循环次数最多为 n^3。可以直接推出这个算法解决 MAX FLOW 问题的时间是 $\mathcal{O}(n^5)$。

MAX FLOW 还有更快的算法，但是它们是在上述算法基础上使用一些并非本书核心内容的技巧。这些算法的时间要求小于 n^5。比如，已知存在多项式时间 n^3 的算法，或许还有更好的可能。对于稀疏网络，具有远少于 n^2 的边，甚至有更快的已知算法，见 1.4.13 节的参考文献。这个领域以往的经验暗示着一条规律：问题一旦找到了多项式算法，时间要求会得到一系列的提高，使得问题可以获得实际意义下更好的计算（MAX FLOW 就是很好的例子，见 1.4.13 节的参考文献）。重要的一步是打破“指数的障碍”，设计第一个多项式时间算法。我们刚才见证了 MAX FLOW 就是如此突破的。

空间消耗呢？即便算法的每次迭代都可以在较小的空间中完成（注意前一节最后的评论），我们仍然需要很多额外空间（大概为 n^2）来存储当前流。我们将在后继章节（第 16 章）看到，不太可能有显著提高。

二分图匹配问题

还有一个相关的有趣问题可以用类似的技巧解决。定义二分图为三元组 $B=(U,V,E)$，其中 $U=\{u_1,\cdots,u_n\}$ 为结点集，称为男孩集，$V=\{v_1,\cdots,v_n\}$ 称为女孩集，$E\subseteq U\times V$ 为边集。二分图如图 1.6a 所示（注意，二分图始终有相同数目的男孩和女孩）。完美匹配，或者简单地，匹配，是二分图中 n 条边的集合 $M\subseteq E$，使得对任意两条边 $(u,v),(u',v')\in M$，有 $u\neq u'$ 和 $v\neq v'$。也就是说，M 中没有两条边与同一个男孩或同一个女孩相邻（例子见图 1.6a）。直观地，匹配是为每个男孩分配一个不同的女孩，使得如果 v 分配给 u，那么 $(u,v)\in E$。最后，MATCHING 描述如下：给定二分图，它有匹配吗？

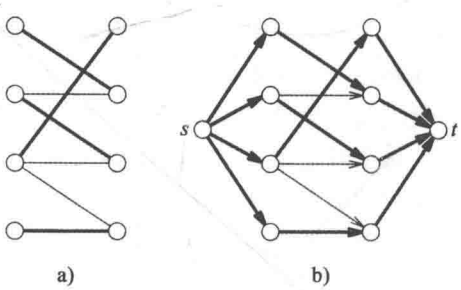


图 1.6　二分图 a）和伴随网络 b）