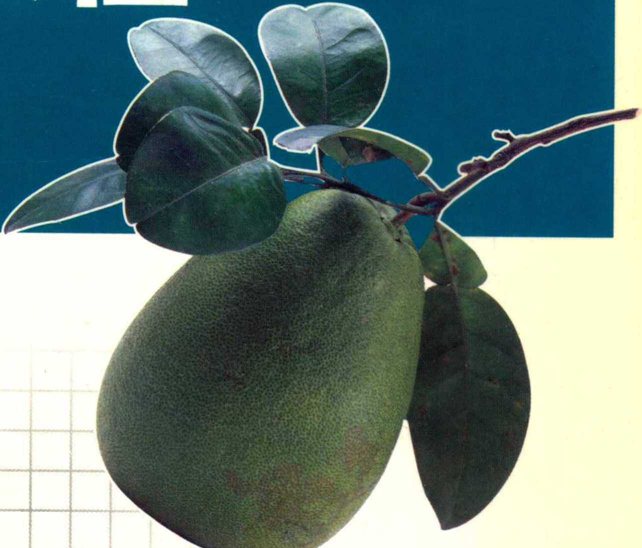
大学计算机教育国外著名教材系列（影印版）

PEARSON
Addison Wesley

COMPUTATIONAL COMPLEXITY

计算复杂性



Christos H. Papadimitriou 著

清华大学出版社

627347

大学计算机教育国外著名教材系列(影印版)

TP301.5
1

Computational Complexity

TP3.0
146

计算复杂性

Christos H. Papadimitriou

University of California, San Diego

清 华 大 学 出 版 社
北 京

图书在版编目(CIP)数据

计算复杂性＝Computational Complexity/帕帕李米特里乌(Papadimitriou, C. H.)著.—影印本.—北京：清华大学出版社，2004.9
(大学计算机教育国外著名教材系列)
ISBN 7-302-08955-8

Ⅰ.计…　Ⅱ.帕…　Ⅲ.计算复杂性－高等学校－教材－英文　Ⅳ.TP301.5

中国版本图书馆 CIP 数据核字(2004)第 075361 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦
http://www.tup.com.cn　　邮　　编：100084
社 总 机：010-62770175　　客户服务：010-62776969

责任编辑：龙啟铭
印 刷 者：北京季蜂印刷有限公司
装 订 者：三河市召亮装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×230　印张：34.25
版　　次：2004 年 9 月第 1 版　2004 年 9 月第1 次印刷
书　　号：ISBN 7-302-08955-8/TP・6335
印　　数：1～3000
定　　价：59.00 元

出版说明

进入 21 世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书，受到国内读者的欢迎和支持。跨入 21 世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列(影印版)”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列(影印版)”做得更好，更适合高校师生的需要。

清华大学出版社

影印版序

计算复杂性理论的研究开始于 20 世纪 70 年代，在 30 多年发展中已经成为了计算机科学理论中最主要的研究领域之一。有关的研究成果不仅对计算机科学，也对应用数学的发展产生了重要的影响。Christos H. Papadimitriou 教授是当代计算复杂性理论方向最著名的专家之一，他的“Computational Complexity”一书被国外许多大学选做计算复杂性理论课程的教材。

本书主要包含算法图灵机、可计算性等有关计算复杂性理论的基本概念；布尔逻辑、一阶逻辑、逻辑中的不可判定性等复杂性理论的基础知识；P 与 NP、NP 完全等各种复杂性的概念及其之间的关系等复杂性理论的核心内容；近 10 年来特别活跃的随机算法、近似算法、并行算法及其复杂性理论的成果；NP 之外如多项式空间等复杂性类的介绍。

本书是第一本全面阐述计算复杂性理论及其进展的教科书，其主要特点如下。

1. 首先从实际应用的问题求解和算法出发，引入了计算复杂性理论的基本概念，接着介绍了计算复杂性理论的逻辑基础，然后阐述 NP 完全性理论的主要内容。这种有机地将计算复杂性理论与应用及逻辑紧密结合的体系，不但为入门的读者提供了进入该领域的捷径，也为已经接触这一理论的人员提供了更为广泛的视角。

2. 内容丰富。不但包含了计算机复杂性理论的核心内容——NP 完全性理论，也反映了近年来在概率计算、近似计算、并行计算等复杂性理论的新进展，并且引入了大量的应用实例，是一本全面阐述计算机复杂性理论的专著。

3. 体系严谨，证明简洁，叙述深入浅出，配有大量的练习，文献引用丰富。不但适合作为研究生或本科高年级学生的教材，也适合从事算法和计算机复杂性研究的人员参考。

随着计算机科学的飞速发展，计算机复杂性理论的研究和应用越来越受到关注，许多高校开设了相关的研究生或本科生课程。和其他研究领域的计算机书籍相比，这方面的书籍少得可怜，其中适合做教材的就更少了。本书作为优秀教材的引进和出版恰好填补了这方面的缺憾。

屈婉玲

北京大学信息学院教授

Preface

I wish nothing else but to speak simply
please grant me this privilege
because we have burdened our song with so much music
that it is slowly sinking
and our art has become so ornate
that the makeup has corroded her face
and it is time to say our few simple words
because tomorrow our soul sails away

Giorgos Seferis

This book is an introduction to the theory of computational complexity at a level appropriate for a beginning graduate or advanced undergraduate course. Computational complexity is the area of computer science that contemplates the reasons why some problems are so hard to solve by computers. This field, virtually non-existent only 20 years ago, has expanded tremendously and now comprises a major part of the research activity in theoretical computer science. No book on complexity can be comprehensive now—certainly this one is not. *It only contains results which I felt I could present clearly and relatively simply, and which I consider central to my point of view of complexity.*

At the risk of burdening the reader so early with a message that will be heard rather frequently and loudly throughout the book's twenty chapters, my point of view is this: I see complexity as the intricate and exquisite interplay between *computation* (complexity classes) and *applications* (that is, problems). Completeness results are obviously central to this approach. So is logic, a most important application that happens to excel in expressing and capturing computation. Computation, problems, and logic are thus the three main currents that run through the book.

Contents

For a quick look at the table of contents, Chapter 1 introduces problems and algorithms—because complexity is best understood when contrasted with simplicity. Chapter 2 treats Turing machines, while at the same time making the point that our approach is very much machine-independent. Chapter 3 is an introduction to undecidability (not only the highest form of complexity, but also a major methodological influence).

Next comes logic. I am aware that this is the part that will seem most alien and unusual to my fellow complexity theorists. But it is so relevant to my way of viewing complexity, so basic for computer science in general, and so rarely treated in a way accessible to computer scientists, that I felt I had to give it a try. Chapter 4 is about Boolean logic (including the algorithmic properties of Horn clauses, and circuits up to Shannon's theorem). Then in Chapter 5 comes first-order logic, its model theory and its proof theory up to the completeness theorem, and enough second-order logic to usher in later Fagin's characterization of NP—a very useful and often overlooked parallel of Cook's theorem. Chapter 6 is a self-contained proof of Gödel's incompleteness theorem, an important early instance of logic expressing computation.

Then complexity is taken on in earnest. Chapter 7 is an exposition of the known relations between complexity classes —including the theorems by Savitch and Immerman-Szelepscényi on space complexity. In Chapter 8 is an introduction to reductions and the concept of completeness, immediately exemplified by Cook's theorem and the P-completeness of the circuit value problem; the parallel of the characterizations of P and NP in terms of logic is also pursued here. Chapter 9 contains many NP-completeness results, sprinkled with tips on proof methodology. Chapter 10 discusses coNP and function problems, while Chapter 11 introduces randomized algorithms, the complexity classes they define, and their implementation in terms of realistic random sources. Circuits and their relationship with complexity and randomization are also introduced there. Chapter 12 is a brief and rather informal introduction to the important subject of cryptography and protocols. Chapter 13 discusses approximation algorithms, and the recent impossibility results based on probabilistically checkable proofs. On the other hand, Chapter 14 surveys "structural" aspects of the $\mathrm{P} \overset{?}{=} \mathrm{NP}$ question, such as intermediate degrees, isomorphism and density, and oracles; it also contains a proof of Razborov's lower bound for monotone circuits.

For a more careful look inside P, parallel algorithms and complexity are the subject of Chapter 15, while Chapter 16 concentrates on logarithmic space, including the random walk algorithm for undirected paths. Finally, beyond NP we find the polynomial hierarchy in Chapter 17 (with Krentel's characterization of optimization problems); counting problems and Valiant's theorem on permanents in Chapter 18; and the many facets of polynomial space (not the least interesting of which is Shamir's recent theorem on interactive protocols) in Chapter 19. The book ends with a glimpse to the intractable territory beyond that.

There are no real mathematical prerequisites —besides of course a certain level of "mathematical maturity," a term always left safely undefined in prefaces. All theorems are proved from first principles (with the exception of two theorems that are quoted without proofs in Chapter 13 on approximability), while many more relevant results are stated in the "notes and problems" sections. The proofs and constructions are often much simpler than one finds in the literature. In fact, the book contains brief and gentle introductions to several subjects as they relate to complexity: Enough elementary number theory to prove Pratt's theorem, the Solovay-Strassen primality test, and the RSA cryptographic protocol (Chapters 10, 11, and 12); elementary probability (Chapters 11 and elsewhere); combinatorics and the probabilistic method (Chapters 10, 13, and 14); recursion theory (Chapters 3 and 14); and, of course, logic (Chapters 4, 5, and 6). Since complexity questions always follow a reasonably comprehensive development of the corresponding algorithmic ideas (efficient algorithms in Chapter 1, randomized algorithms in Chapter 11, approximation algorithms in Chapter 13, and parallel algorithms in Chapter 15), the book is also a passable introduction to algorithms—although only rough analysis, enough to establish polynomiality, is attempted in each case.

Notes and Problems

Each chapter ends with a section that contains intertwined references, notes, exercises, and problems; many of the problems are hand-held tours of further results and topics. In my view this is perhaps the most important section of the chapter (it is often by far the longest one), and you should consider reading it as part of the text. It usually gives historical perspective and places the chapter within the broader field. Most problems are followed with a parenthetical hint and/or references. All are doable, at least following the hint or looking up the solution in the library (I have found that this is often at least as valuable to my students as passing yet another IQ test). There is no difficulty code for problems, but you are warned about any really hard ones.

Teaching

For all its obvious emphasis on complexity, this book has been designed (and used) as a general introduction to the theory of computation for computer scientists. My colleagues and I have used it over the past three years in a ten-week course mainly intended for first-year computer science masters students at the University of California at San Diego. Two weeks are usually enough for a review of the first four chapters, generally familiar from the students' undergraduate training. Logic takes the next three weeks, often without the proof of the completeness theorem. The remaining five weeks are enough for Chapter 7, a serious drill on NP-completeness (not included in algorithms courses at UCSD), and a selection of one or two among Chapters 11 through 14. A semester-length course could cover all four of these chapters. If you must skip logic, then Chapter 15 on parallelism can be added (however, I believe that a good part of the book's message and value would be lost this way).

There are at least two other courses one can teach from this book: The subjects in the first nine chapters are, in my view, so central for computer scientists, that they can replace with pride the usual automata and formal languages in an advanced undergraduate introductory theory course (especially since many compiler courses are nowadays self-contained in this respect). Also, I have used twice the last eleven chapters for a second course in theory; the goal here is to bring interested graduate students up to the research issues in complexity—or, at least, help them be an informed audience at a theory conference.

Debts

My thinking on complexity was shaped by a long process of exciting and inspiring interactions with my teachers, students, and colleagues (especially those who were all three). I am immensely grateful to all of them: 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 (who also helped me a lot with the figures, the final check, and the index), 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; and above all, Mihalis Yannakakis and Mike Sipser. Many of them read early drafts of this book and contributed criticisms, ideas, and corrections—or made me nervous with their silence. Of all the students in the course who commented on my lecture notes I can only remember the names of David Morgenthaller, Goran Gogic, Markus Jacobsson, and George Xylomenos (but I do remember the smiles of the rest). Finally, many thanks to Richard Beigel, Matt Wong, Wenhong Zhu, and their complexity class at Yale for catching many errors in the first printing. Naturally, I am responsible for the remaining errors—although, in my opinion, my friends could have caught a few more.

I am very grateful to Martha Sideri for her sweet encouragement and support, as well as her notes, ideas, opinions, and help with the cover design.

I worked on writing this book at the University of California, San Diego, but also during visits at AT&T Bell Laboratories, the University of Bonn, the Max-Planck-Institut at Saarbrücken, the University of Patras and the Computer Technology Institute there, and the Université de Paris Sud. My research on algorithms and complexity was supported by the National Science Foundation, the Esprit project ALCOM, and the Irwin Mark and Joan Klein Jacobs chair for information and computer sciences at the University of California at San Diego.

It was a pleasure to work on this project with Tom Stone and his colleagues at Addison-Wesley. Finally, I typeset the book using Don Knuth's TEX, and my macros evolved from those Jeff Ullman gave me many years ago.

Christos H. Papadimitriou
La Jolla, summer of 1993

Contents

PART

I ALGORITHMS

Any book on algorithms ends with a chapter on complexity, so it is fitting to start this book by recounting some basic facts about algorithms. Our goal in these three chapters is to make a few simple but important points: Computational problems are not only things that have to be solved, they are also objects that can be worth studying. Problems and algorithms can be formalized and analyzed mathematically—for example as languages and Turing machines, respectively—and the precise formalism does not matter much. Polynomial-time computability is an important desirable property of computational problems, akin to the intuitive notion of practical solvability. Many different models of computation can simulate one another with a polynomial loss of efficiency—with the single exception of nondeterminism, which appears to require exponential time for its simulation. And there are problems that have no algorithms at all, however inefficient.