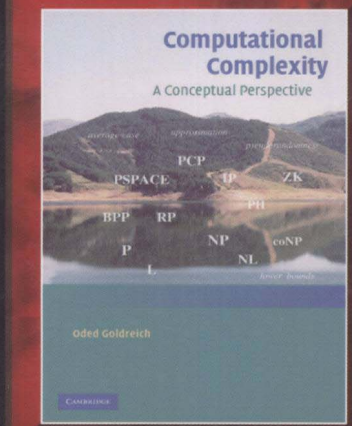
TURING 图灵原版计算机科学系列

CAMBRIDGE

Computational Complexity
A Conceptual Perspective

计算复杂性
（英文版）

[以] Oded Goldreich 著



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵原版计算机科学系列

Computational Complexity
A Conceptual Perspective

计算复杂性
(英文版)

[以] Oded Goldreich 著

人民邮电出版社
北京

图书在版编目（CIP）数据

计算复杂性：英文／（以）戈德赖希（Goldreich, O.）著．—北京：人民邮电出版社，2010.4

（图灵原版计算机科学系列）

书名原文：Computational Complexity: A Conceptual Perspective

ISBN 978-7-115-22400-2

I. ①计… II. ①戈… III. ①计算复杂性－英文 IV. ①TP301.5

中国版本图书馆CIP数据核字（2010）第031182号

内 容 提 要

本书是理论计算机科学领域的名著。书中对计算任务的固有复杂性研究进行了一般性介绍，涉及了复杂性理论的很多子领域，涵盖了NP完整性、空间复杂性、随机性和计数、伪随机数生成器等内容，还在附录里面给出了现代密码学基础等内容。

本书内容严谨，可读性强，适合作为高年级本科生、研究生的教材，对涉及计算复杂性的专业人员也是理想的技术参考书。

图灵原版计算机科学系列

计算复杂性（英文版）

◆ 著 [以] Oded Goldreich

责任编辑 杨海玲

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子函件 315@ptpress.com.cn

网址 http://www.ptpress.com.cn

三河市海波印务有限公司印刷

◆ 开本：787×1092 1/16

印张：39.25

字数：754千字 2010年4月第1版

印数：1－2 000册 2010年4月河北第1次印刷

著作权合同登记号 图字：01-2010-0211号

ISBN 978-7-115-22400-2

定价：99.00元

读者服务热线：（010）51095186 印装质量热线：（010）67129223

反盗版热线：（010）67171154

版 权 声 明

Preface

The quest for efficiency is ancient and universal, as time and other resources are always in shortage. Thus, the question of which tasks can be performed efficiently is central to the human experience.

A key step toward the systematic study of the aforementioned question is a rigorous definition of the notion of a task and of procedures for solving tasks. These definitions were provided by computability theory, which emerged in the 1930s. This theory focuses on computational tasks, and considers automated procedures (i.e., computing devices and algorithms) that may solve such tasks.

In focusing attention on computational tasks and algorithms, computability theory has set the stage for the study of the computational resources (like time) that are required by such algorithms. When this study focuses on the resources that are necessary for *any* algorithm that solves a particular task (or a task of a particular type), the study becomes part of the theory of Computational Complexity (also known as Complexity Theory).[1]

Complexity Theory is a central field of the theoretical foundations of computer science. It is concerned with the study of the *intrinsic complexity of computational tasks.* That is, a typical complexity theoretic study refers to the computational resources required to solve a computational task (or a class of such tasks), rather than referring to a specific algorithm or an algorithmic schema. Actually, research in Complexity Theory tends to *start with and focus on the computational resources themselves*, and addresses the effect of limiting these resources on the class of tasks that can be solved. Thus, Computational Complexity is the general study of what can be achieved within limited time (and/or other limited natural computational resources).

The (half-century) history of Complexity Theory has witnessed two main research efforts (or directions). The first direction is aimed toward actually establishing concrete lower bounds on the complexity of computational problems, via an analysis of the evolution of the process of computation. Thus, in a sense, the heart of this direction is a "low-level" analysis of computation. Most research in circuit complexity and in proof complexity falls within this category. In contrast, a second research effort is aimed at exploring the connections among computational problems and notions, without being able to provide absolute statements regarding the individual problems or notions. This effort may be

[1] In contrast, when the focus is on the design and analysis of specific algorithms (rather than on the intrinsic complexity of the task), the study becomes part of a related subfield that may be called Algorithmic Design and Analysis. Furthermore, Algorithmic Design and Analysis tends to be sub-divided according to the domain of mathematics, science, and engineering in which the computational tasks arise. In contrast, Complexity Theory typically maintains a unity of the study of tasks solvable within certain resources (regardless of the origins of these tasks).

viewed as a "high-level" study of computation. The theory of NP-completeness as well as the studies of approximation, probabilistic proof systems, pseudorandomness, and cryptography all fall within this category.

The current book focuses on the latter effort (or direction). The main reason for our decision to focus on the "high-level" direction is the clear *conceptual significance* of the known results. That is, many known results in this direction have an extremely appealing conceptual message, which can be appreciated also by non-experts. Furthermore, these conceptual aspects can be explained without entering excessive technical detail. Consequently, the "high-level" direction is more suitable for an exposition in a book of the current nature.[2]

The last paragraph brings us to a discussion of the nature of the current book, which is captured by the subtitle (i.e., "a conceptual perspective"). Our main thesis is that *Complexity Theory is extremely rich in conceptual content*, and that *this content should be explicitly communicated in expositions and courses on the subject*. The desire to provide a corresponding textbook is indeed the motivation for writing the current book and its main governing principle.

This book offers a conceptual perspective on Complexity Theory, and the presentation is designed to highlight this perspective. It is intended to serve as an introduction to the field, and can be used either as a textbook or for self-study. Indeed, the book's primary target audience consists of students who wish to learn Complexity Theory and educators who intend to teach a course on Complexity Theory. Still, we hope that the book will be useful also to experts, especially to experts in one sub-area of Complexity Theory who seek an introduction to and/or an overview of some other sub-area.

It is also hoped that the book may help promote general interest in Complexity Theory and make this field acccessible to general readers with adequate background (which consists mainly of being comfortable with abstract discussions, definitions, and proofs). However, we do expect most readers to have a basic knowledge of algorithms, or at least be fairly comfortable with the notion of an algorithm.

The book focuses on several sub-areas of Complexity Theory (see the following organization and chapter summaries). In each case, the exposition starts from the intuitive questions addressed by the sub-area, as embodied in the concepts that it studies. The exposition discusses the fundamental *importance* of these questions, the *choices* made in the actual formulation of these questions and notions, the *approaches* that underlie the answers, and the *ideas* that are embedded in these answers. Our view is that these ("non-technical") aspects are the core of the field, and the presentation attempts to reflect this view.

We note that being guided by the conceptual contents of the material leads, in some cases, to technical simplifications. Indeed, for many of the results presented in this book, the presentation of the proof is different (and arguably easier to understand) than the standard presentations.

Web site for notices regarding this book. We intend to maintain a Web site listing corrections of various types. The location of the site is

`http://www.wisdom.weizmann.ac.il/~oded/cc-book.html`

[2]In addition, we mention a subjective reason for our decision: The "high-level" direction is within our own expertise, while this cannot be said about the "low-level" direction.

Organization and Chapter Summaries

This book consists of ten chapters and seven appendices. The chapters constitute the core of this book and are written in a style adequate for a textbook, whereas the appendices provide either relevant background or additional perspective and are written in the style of a survey article. The relative length and ordering of the chapters (and appendices) do not reflect their relative importance, but rather an attempt at the best logical order (i.e., minimizing the number of forward pointers).

Following are brief summaries of the book's chapters and appendices. These summaries are more novice-friendly than those provided in Section 1.1.3 but less detailed than the summaries provided at the beginning of each chapter.

Chapter 1: Introduction and Preliminaries. The introduction provides a high-level overview of some of the content of Complexity Theory as well as a discussion of some of the characteristic features of this field. In addition, the introduction contains several important comments regarding the approach and conventions of the current book. The preliminaries provide the relevant background on *computability theory*, which is the setting in which complexity theoretic questions are being studied. Most importantly, central notions such as search and decision problems, algorithms that solve such problems, and their complexity are defined. In addition, this part presents the basic notions underlying non-uniform models of computation (like Boolean circuits).

Chapter 2: P, NP, and NP-Completeness. The P versus NP Question can be phrased as asking whether or not finding solutions is harder than checking the correctness of solutions. An alternative formulation asks whether or not discovering proofs is harder than verifying their correctness, that is, is proving harder than verifying. It is widely believed that the answer to the two equivalent formulations is that finding (resp., proving) is harder than checking (resp., verifying); that is, it is believed that P is different from NP. At present, when faced with a hard problem in NP, we can only hope to prove that it is not in P assuming that NP is different from P. This is where the theory of NP-completeness, which is based on the notion of a reduction, comes into the picture. In general, one computational problem is reducible to another problem if it is possible to efficiently solve the former when provided with an (efficient) algorithm for solving the latter. A problem (in NP) is NP-complete if any problem in NP is reducible to it. Amazingly enough, NP-complete problems exist, and furthermore, hundreds of natural computational problems arising in many different areas of mathematics and science are NP-complete.

Chapter 3: Variations on P and NP. Non-uniform polynomial time (P/poly) captures efficient computations that are carried out by devices that handle specific input lengths. The basic formalism ignores the complexity of constructing such devices (i.e., a uniformity condition), but a finer formalism (based on "machines that take advice") allows us to quantify the amount of non-uniformity. This provides a generalization of P. In contrast, the Polynomial-time Hierarchy (PH) generalizes NP by considering statements expressed by a quantified Boolean formula with a fixed number of alternations of existential and universal quantifiers. It is widely believed that each quantifier alternation adds expressive power to the class of such formulae. The two different classes are related by showing that if NP is contained in P/poly then the Polynomial-time Hierarchy collapses to its second level (i.e., Σ_2).

Chapter 4: More Resources, More Power? When using "nice" functions to determine an algorithm's resources, it is indeed the case that more resources allow for more tasks to be performed. However, when "ugly" functions are used for the same purpose, increasing the resources may have no effect. By nice functions we mean functions that can be computed without exceeding the amount of resources that they specify. Thus, we get results asserting, for example, that there are problems that are solvable in cubic time but not in quadratic time. In the case of non-uniform models of computation, the issue of "nicety" does not arise, and it is easy to establish separation results.

Chapter 5: Space Complexity. This chapter is devoted to the study of the space complexity of computations, while focusing on two rather extreme cases. The first case is that of algorithms having logarithmic space complexity, which seem a proper and natural subset of the set of polynomial-time algorithms. The second case is that of algorithms having polynomial space complexity, which in turn can solve almost all computational problems considered in this book. Among the many results presented in this chapter are a log-space algorithm for exploring (undirected) graphs, and a log-space reduction of the set of directed graphs that are *not* strongly connected to the set of directed graphs that are strongly connected. These results capture fundamental properties of space complexity, which seems to differentiate it from time complexity.

Chapter 6: Randomness and Counting. Probabilistic polynomial-time algorithms with various types of failure give rise to complexity classes such as $\mathcal{BPP}$, $\mathcal{RP}$, and $\mathcal{ZPP}$. The results presented include the emulation of probabilistic choices by non-uniform advice (i.e., $\mathcal{BPP} \subset \mathcal{P}/\text{poly}$) and the emulation of two-sided probabilistic error by an $\exists\forall$-sequence of quantifiers (i.e., $\mathcal{BPP} \subseteq \Sigma_2$). Turning to counting problems (i.e., counting the number of solutions for NP-type problems), we distinguish between exact counting and approximate counting (in the sense of relative approximation). While any problem in $\mathcal{PH}$ is reducible to the exact counting class $\#\mathcal{P}$, approximate counting (for $\#\mathcal{P}$) is (probabilistically) reducible to $\mathcal{NP}$. Additional related topics include $\#\mathcal{P}$-completeness, the complexity of searching for unique solutions, and the relation between approximate counting and generating almost uniformly distributed solutions.

Chapter 7: The Bright Side of Hardness. It turns out that hard problems can be "put to work" to our benefit, most notably in cryptography. One key issue that arises in this context is bridging the gap between "occasional" hardness (e.g., worst-case hardness or mild average-case hardness) and "typical" hardness (i.e., strong average-case hardness).

We consider two conjectures that are related to $\mathcal{P} \neq \mathcal{NP}$. The first conjecture is that there are problems that are solvable in exponential time but are not solvable by (non-uniform) families of small (say, polynomial-size) circuits. We show that these types of worst-case conjectures can be transformed into average-case hardness results that yield non-trivial derandomizations of $\mathcal{BPP}$ (and even $\mathcal{BPP} = \mathcal{P}$). The second conjecture is that there are problems in NP for which it is easy to generate (solved) instances that are hard to solve for other people. This conjecture is captured in the notion of *one-way functions*, which are functions that are easy to evaluate but hard to invert (in an average-case sense). We show that functions that are hard to invert in a relatively mild average-case sense yield functions that are hard to invert almost everywhere, and that the latter yield predicates that are very hard to approximate (called *hard-core predicates*). The latter are useful for the construction of general-purpose pseudorandom generators, as well as for a host of cryptographic applications.

Chapter 8: Pseudorandom Generators. A fresh view of the *question of randomness* was taken in the theory of computing: It has been postulated that a distribution is pseudorandom if it cannot be told apart from the uniform distribution by any efficient procedure. The paradigm, originally associating efficient procedures with polynomial-time algorithms, has been applied also with respect to a variety of limited classes of such distinguishing procedures. The archetypical case of pseudorandom generators refers to efficient generators that fool any feasible procedure; that is, the potential distinguisher is any probabilistic polynomial-time algorithm, which may be more complex than the generator itself. These generators are called general-purpose, because their output can be safely used in any efficient application. In contrast, for purposes of derandomization, one may use pseudorandom generators that are somewhat more complex than the potential distinguisher (which represents the algorithm to be derandomized). Following this approach and using various hardness assumptions, one may obtain corresponding derandomizations of $\mathcal{BPP}$ (including a full derandomization; i.e., $\mathcal{BPP} = \mathcal{P}$). Other forms of pseudorandom generators include ones that fool space-bounded distinguishers, and even weaker ones that only exhibit some limited random behavior (e.g., outputting a pairwise independent sequence).

Chapter 9: Probabilistic Proof Systems. Randomized and interactive verification procedures, giving rise to *interactive proof systems*, seem much more powerful than their deterministic counterparts. In particular, interactive proof systems exist for any set in $\mathcal{PSPACE} \supseteq \text{co}\mathcal{NP}$ (e.g., for the set of unsatisfied propositional formulae), whereas it is widely believed that some sets in $\text{co}\mathcal{NP}$ do *not* have NP-proof systems. Interactive proofs allow the meaningful conceptualization of *zero-knowledge proofs*, which are interactive proofs that yield nothing (to the verifier) beyond the fact that the assertion is indeed valid. Under reasonable complexity assumptions, every set in $\mathcal{NP}$ has a zero-knowledge proof system. (This result has many applications in cryptography.) A third type of probabilistic proof system underlies the model of PCPs, which stands for *probabilistically checkable proofs*. These are (redundant) NP-proofs that offer a trade-off between the number of locations (randomly) examined in the proof and the confidence in its validity. In particular, a small constant error probability can be obtained by reading a constant number of bits in the redundant NP-proof. The PCP Theorem asserts that NP-proofs can be efficiently transformed into PCPs. The study of PCPs is closely related to the study of the complexity of approximation problems.

Chapter 10: Relaxing the Requirements. In light of the apparent infeasibility of solving numerous useful computational problems, it is natural to seek relaxations of those problems that remain useful for the original applications and yet allow for feasible solving procedures. Two such types of relaxation are provided by adequate notions of approximation and a theory of average-case complexity. The notions of approximation refer to the computational problems themselves; that is, for each problem instance we extend the set of admissible solutions. In the context of search problems this means settling for solutions that have a value that is "sufficiently close" to the value of the optimal solution, whereas in the context of decision problems this means settling for procedures that distinguish yes-instances from instances that are "far" from any yes-instance. Turning to average-case complexity, we note that a systematic study of this notion requires the development of a non-trivial conceptual framework. One major aspect of this framework is limiting the class of distributions in a way that, on the one hand, allows for various types of natural distributions and, on the other hand, prevents the collapse of average-case hardness to worst-case hardness.

Appendix A: Glossary of Complexity Classes. The glossary provides self-contained definitions of most complexity classes mentioned in the book. The glossary is partitioned into two parts, dealing separately with complexity classes that are defined in terms of algorithms and their resources (i.e., time and space complexity of Turing machines) and complexity classes defined in terms of non-uniform circuits (and referring to their size and depth). In particular, the following classes are defined: $\mathcal{P}$, $\mathcal{NP}$, co$\mathcal{NP}$, $\mathcal{BPP}$, $\mathcal{RP}$, co$\mathcal{RP}$, $\mathcal{ZPP}$, $\#\mathcal{P}$, $\mathcal{PH}$, $\mathcal{E}$, $\mathcal{EXP}$, $\mathcal{NEXP}$, $\mathcal{L}$, $\mathcal{NL}$, $\mathcal{RL}$, $\mathcal{PSPACE}$, $\mathcal{P}$/poly, $\mathcal{NC}^k$, and $\mathcal{AC}^k$.

Appendix B: On the Quest for Lower Bounds. This brief survey describes the most famous attempts at proving lower bounds on the complexity of natural computational problems. The first part, devoted to Circuit Complexity, reviews lower bounds for the *size* of (restricted) circuits that solve natural computational problems. This represents a program whose long-term goal is proving that $\mathcal{P} \neq \mathcal{NP}$. The second part, devoted to Proof Complexity, reviews lower bounds on the length of (restricted) propositional proofs of natural tautologies. This represents a program whose long-term goal is proving that $\mathcal{NP} \neq$ co$\mathcal{NP}$.

Appendix C: On the Foundations of Modern Cryptography. This survey of the foundations of cryptography focuses on the paradigms, approaches, and techniques that are used to conceptualize, define, and provide solutions to natural security concerns. It presents some of these conceptual tools as well as some of the fundamental results obtained using them. The appendix augments the partial treatment of one-way functions, pseudorandom generators, and zero-knowledge proofs (included in Chapters 7–9). Using these basic tools, the appendix provides a treatment of basic cryptographic applications such as encryption, signatures, and general cryptographic protocols.

Appendix D: Probabilistic Preliminaries and Advanced Topics in Randomization. The probabilistic preliminaries include conventions regarding random variables as well as three useful inequalities (i.e., Markov's Inequality, Chebyshev's Inequality, and Chernoff Bound). The advanced topics include constructions and lemmas regarding families of hashing functions, a study of the sample and randomness complexities of estimating the

average value of an arbitrary function, and the problem of randomness extraction (i.e., procedures for extracting almost perfect randomness from sources of weak or defected randomness).

Appendix E: Explicit Constructions. Complexity Theory provides a clear perspective on the intuitive notion of an explicit construction. This perspective is demonstrated with respect to error-correcting codes and expander graphs. Starting with codes, the appendix focuses on various computational aspects, and offers a review of several popular constructions as well as a construction of a binary code of constant rate and constant relative distance. Also included are a brief review of the notions of locally testable and locally decodable codes, and a useful upper bound on the number of codewords that are close to any single sequence. Turning to expander graphs, the appendix contains a review of two standard definitions of expanders, two levels of explicitness, two properties of expanders that are related to (single-step and multi-step) random walks on them, and two explicit constructions of expander graphs.

Appendix F: Some Omitted Proofs. This appendix contains some proofs that were not included in the main text (for a variety of reasons) and still are beneficial as alternatives to the original and/or standard presentations. Included are a proof that $\mathcal{PH}$ is reducible to $\#\mathcal{P}$ via randomized Karp-reductions, and the presentation of two useful transformations regarding interactive proof systems.

Appendix G: Some Computational Problems. This appendix includes definitions of most of the specific computational problems that are referred to in the main text. In particular, it contains a brief introduction to graph algorithms, Boolean formulae, and finite fields.

Acknowledgments

My perspective on Complexity Theory was most influenced by Shimon Even and Leonid Levin. In fact, it was hard not to be influenced by these two remarkable and highly opinionated researchers (especially for somebody like me who was fortunate to spend a lot of time with them).[1]

Shimon Even viewed Complexity Theory as the study of the limitations of algorithms, a study concerned with natural computational resources and natural computational tasks. Complexity Theory was there to guide the engineer and to address the deepest questions that bother an intellectually curious computer scientist. I believe that this book shares Shimon's perspective of Complexity Theory as evolving around such questions.

Leonid Levin emphasized the general principles that underlie Complexity Theory, rejecting any "model-dependent effects" as well as the common coupling of Complexity Theory with the theory of automata and formal languages. In my opinion, this book is greatly influenced by these perspectives of Leonid.

I wish to acknowledge the influence of numerous other colleagues on my professional perspectives and attitudes. These include Shafi Goldwasser, Dick Karp, Silvio Micali, and Avi Wigderson. Needless to say, this is but a partial list that reflects influences of which I am most aware.

The year I spent at Radcliffe Institute for Advanced Study (of Harvard University) was instrumental in my decision to undertake the writing of this book. I am grateful to Radcliffe for creating such an empowering atmosphere. I also wish to thank many colleagues for their comments and advice (or help) regarding earlier versions of this text. A partial list includes Noga Alon, Noam Livne, Dieter van Melkebeek, Omer Reingold, Dana Ron, Ronen Shaltiel, Amir Shpilka, Madhu Sudan, Salil Vadhan, and Avi Wigderson.

Lastly, I am grateful to Mohammad Mahmoody Ghidary and Or Meir for their careful reading of drafts of this manuscript and for the numerous corrections and suggestions that they have provided.

Relation to previous texts. Some of the text of this book has been adapted from previous texts of mine. In particular, Chapters 8 and 9 were written based on my surveys [90, Chap. 3] and [90, Chap. 2], respectively; but the exposition has been extensively revised

[1] Shimon Even was my graduate studies adviser (at the Technion, 1980–83), whereas I had a lot of meetings with Leonid Levin during my postdoctoral period (at MIT, 1983–86).

to fit the significantly different aims of the current book. Similarly, Section 7.1 and Appendix C were written based on my survey [90, Chap. 1] and books [91, 92]; but, again, the previous texts are very different in many ways. In contrast, Appendix B was adapted with relatively little modifications from an early draft of a section in an article by Avi Wigderson and myself [107].

List of Figures

1.1 Dependencies among the advanced chapters. *page* 9
1.2 A single step by a Turing machine. 23
1.3 A circuit computing $f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2, x_1 \wedge \neg x_2 \wedge x_4)$. 38
1.4 Recursive construction of parity circuits and formulae. 42
2.1 Consecutive computation steps of a Turing machine. 74
2.2 The idea underlying the reduction of CSAT to SAT. 76
2.3 The reduction to G3C – the clause gadget and its sub-gadget. 81
2.4 The reduction to G3C – connecting the gadgets. 82
2.5 The world view under $\mathcal{P} \neq \text{co}\mathcal{NP} \cap \mathcal{NP} \neq \mathcal{NP}$. 97
3.1 Two levels of the Polynomial-time Hierarchy. 119
4.1 The Gap Theorem – determining the value of $t(n)$. 137
5.1 Algorithmic composition for space-bounded computation. 147
5.2 The recursive procedure in $\mathcal{NL} \subseteq \text{DSPACE}(O(\log^2))$. 167
5.3 The main step in proving $\mathcal{NL} = \text{co}\mathcal{NL}$. 170
6.1 The reduction to the Permanent – tracks connecting gadgets. 207
6.2 The reduction to the Permanent – the gadget's effect. 207
6.3 The reduction to the Permanent – A Deus ex Machina gadget. 208
6.4 The reduction to the Permanent – a structured gadget. 209
6.5 The reduction to the Permanent – the box's effect. 209
7.1 The hard-core of a one-way function – an illustration. 250
7.2 Proofs of hardness amplification: Organization. 258
8.1 Pseudorandom generators – an illustration. 287
8.2 Analysis of stretch amplification – the i^{th} hybrid. 300
8.3 Derandomization of $\mathcal{BPL}$ – the generator. 321
8.4 An affine transformation affected by a Toeplitz matrix. 327
8.5 The LFSR small-bias generator. 330
8.6 Pseudorandom generators at a glance. 335
9.1 Arithmetization of CNF formulae. 360
9.2 Zero-knowledge proofs – an illustration. 369
9.3 The PCP model – an illustration. 380
9.4 Testing consistency of linear and quadratic forms. 387
9.5 Composition of PCP system – an illustration. 389
9.6 The amplifying reduction in the second proof of the PCP Theorem. 397

10.1 Two types of average-case completeness. 446
10.2 Worst-case versus average-case assumptions. 451
E.1 Detail of the Zig-Zag product of G' and G. 562
F.1 The transformation of an MA-game into an AM-game. 579
F.2 The transformation of MAMA into AMA. 581

Contents

1 Introduction and Preliminaries 1

1.1 Introduction 1

1.1.1 A Brief Overview of Complexity Theory 2

1.1.2 Characteristics of Complexity Theory 6

1.1.3 Contents of This Book 8

1.1.4 Approach and Style of This Book 12

1.1.5 Standard Notations and Other Conventions 16

1.2 Computational Tasks and Models 17

1.2.1 Representation 18

1.2.2 Computational Tasks 18

1.2.3 Uniform Models (Algorithms) 20

1.2.4 Non-uniform Models (Circuits and Advice) 36

1.2.5 Complexity Classes 42

Chapter Notes 43

2 P, NP, and NP-Completeness 44

2.1 The P Versus NP Question 46

2.1.1 The Search Version: Finding Versus Checking 47

2.1.2 The Decision Version: Proving Versus Verifying 50

2.1.3 Equivalence of the Two Formulations 54

2.1.4 Two Technical Comments Regarding NP 55

2.1.5 The Traditional Definition of NP 55

2.1.6 In Support of P Different from NP 57

2.1.7 Philosophical Meditations 58

2.2 Polynomial-Time Reductions 58

2.2.1 The General Notion of a Reduction 59

2.2.2 Reducing Optimization Problems to Search Problems 61

2.2.3 Self-Reducibility of Search Problems 63

2.2.4 Digest and General Perspective 67

2.3 NP-Completeness 67

2.3.1 Definitions 68

2.3.2 The Existence of NP-Complete Problems 69

2.3.3 Some Natural NP-Complete Problems 71

2.3.4 NP Sets That Are Neither in P nor NP-Complete 81
2.3.5 Reflections on Complete Problems 85
2.4 Three Relatively Advanced Topics 87
2.4.1 Promise Problems 87
2.4.2 Optimal Search Algorithms for NP 92
2.4.3 The Class coNP and Its Intersection with NP 94
Chapter Notes 97
Exercises 99

3 Variations on P and NP 108
3.1 Non-uniform Polynomial Time (P/poly) 108
3.1.1 Boolean Circuits 109
3.1.2 Machines That Take Advice 111
3.2 The Polynomial-Time Hierarchy (PH) 113
3.2.1 Alternation of Quantifiers 114
3.2.2 Non-deterministic Oracle Machines 117
3.2.3 The P/poly Versus NP Question and PH 119
Chapter Notes 121
Exercises 122

4 More Resources, More Power? 127
4.1 Non-uniform Complexity Hierarchies 128
4.2 Time Hierarchies and Gaps 129
4.2.1 Time Hierarchies 129
4.2.2 Time Gaps and Speedup 136
4.3 Space Hierarchies and Gaps 139
Chapter Notes 139
Exercises 140

5 Space Complexity 143
5.1 General Preliminaries and Issues 144
5.1.1 Important Conventions 144
5.1.2 On the Minimal Amount of Useful Computation Space 145
5.1.3 Time Versus Space 146
5.1.4 Circuit Evaluation 153
5.2 Logarithmic Space 153
5.2.1 The Class L 154
5.2.2 Log-Space Reductions 154
5.2.3 Log-Space Uniformity and Stronger Notions 155
5.2.4 Undirected Connectivity 155
5.3 Non-deterministic Space Complexity 162
5.3.1 Two Models 162
5.3.2 NL and Directed Connectivity 164
5.3.3 A Retrospective Discussion 171
5.4 PSPACE and Games 172
Chapter Notes 175
Exercises 175