
Computational Complexity: A Modern Approach

Draft of a book: Dated January 2007
Comments welcome!

Sanjeev Arora and Boaz Barak
Princeton University
complexitybook@gmail.com

Not to be reproduced or distributed without the authors' permission

This is an Internet draft. Some chapters are more finished than others. References and attributions are very preliminary and we apologize in advance for any omissions (but hope you will nevertheless point them out to us).

Please send us bugs, typos, missing references or general comments to
complexitybook@gmail.com — **Thank You!!**

DRAFT

DRAFT

About this book

Computational complexity theory has developed rapidly in the past three decades. The list of surprising and fundamental results proved since 1990 alone could fill a book: these include new probabilistic definitions of classical complexity classes ($\mathbf{IP} = \mathbf{PSPACE}$ and the \mathbf{PCP} Theorems) and their implications for the field of approximation algorithms; Shor's algorithm to factor integers using a quantum computer; an understanding of why current approaches to the famous \mathbf{P} versus \mathbf{NP} will not be successful; a theory of derandomization and pseudorandomness based upon computational hardness; and beautiful constructions of pseudorandom objects such as extractors and expanders.

This book aims to describe such recent achievements of complexity theory in the context of the classical results. It is intended to both serve as a textbook as a reference for self-study. This means it must simultaneously cater to many audiences, and it is carefully designed with that goal. Throughout the book we explain the context in which a certain notion is useful, and *why* things are defined in a certain way. Examples and solved exercises accompany key definitions. We assume essentially no computational background and very minimal mathematical background, which we review in Appendix A.

We have also provided a *web site* for this book at <http://www.cs.princeton.edu/theory/complexity/> with related auxiliary material. This includes web chapters on automata and computability theory, detailed teaching plans for courses based on this book, a draft of all the book's chapters, and links to other online resources covering related topics.

The book is divided into three parts:

Part I: Basic complexity classes. This volume provides a broad introduction to the field. Starting from the definition of Turing machines and the basic notions of computability theory, this volume covers the basic time and space complexity classes, and also includes a few more modern topics such as probabilistic algorithms, interactive proofs and cryptography.

Part II: Lower bounds on concrete computational models. This part describes lower bounds on resources required to solve algorithmic tasks on concrete models such as circuits, decision trees, etc. Such models may seem at first sight very different from Turing machines, but looking deeper one finds interesting interconnections.

Part III: Advanced topics. This part is largely devoted to developments since the late 1980s. It includes average case complexity, derandomization and pseudorandomness, the \mathbf{PCP} theorem and hardness of approximation, proof complexity and quantum computing.

Almost every chapter in the book can be read in isolation (though we recommend reading Chapters 1, 2 and 7 before reading later chapters). This is important because the book is aimed

at many classes of readers:

- *Physicists, mathematicians, and other scientists.* This group has become increasingly interested in computational complexity theory, especially because of high-profile results such as Shor’s algorithm and the recent deterministic test for primality. This intellectually sophisticated group will be able to quickly read through Part I. Progressing on to Parts II and III they can read individual chapters and find almost everything they need to understand current research.
- *Computer scientists (e.g., algorithms designers) who do not work in complexity theory per se.* They may use the book for self-study or even to teach a graduate course or seminar.
- *All those —professors or students— who do research in complexity theory or plan to do so.* They may already know Part I and use the book for Parts II and III, possibly in a seminar or reading course. The coverage of advanced topics there is detailed enough to allow this.

This book can be used as a textbook for several types of courses. We will provide several teaching plans and material for such courses on the book’s web site.

- *Undergraduate Theory of Computation Course.* Part I may be suitable for an undergraduate course that is an alternative to the more traditional *Theory of Computation* course currently taught in most computer science departments (and exemplified by Sipser’s excellent book with the same name [SIP96]). Such a course would have a greater emphasis on modern topics such as probabilistic algorithms and cryptography. We note that in contrast to Sipser’s book, the current book has a quite minimal coverage of computability and no coverage of automata theory, but we provide web-only chapters with more coverage of these topics on the book’s web site. The prerequisite mathematical background would be some comfort with mathematical proofs and elementary probability on finite sample spaces, topics that are covered in typical “discrete math”/“math for CS” courses currently offered in most CS departments.
- *Advanced undergraduate/beginning graduate introduction to complexity course.* The book can be used as a text for an introductory complexity course aimed at undergraduate or non-theory graduate students (replacing Papadimitriou’s 1994 book [Pap94] that does not contain many recent results). Such a course would probably include many topics from Part I and then a sprinkling from Parts II and III, and assume some background in algorithms and/or the theory of computation.
- *Graduate Complexity course.* The book can serve as a text for a graduate complexity course that prepares graduate students interested in theory to do research in complexity and related areas. Such a course can use parts of Part I to review basic material, and then move on to the advanced topics of Parts II and III. The book contains far more material than can be taught in one term, and we provide on our website several alternative outlines for such a course.

We hope that this book conveys our excitement about this new field and the insights it provides in a host of older disciplines.

DRAFT

Contents

About this book	iii
Introduction	p0.1 (1)
I Basic Complexity Classes	p0.9 (9)
1 The computational model —and why it doesn't matter	p1.1 (11)
1.1 Encodings and Languages: Some conventions	p1.2 (12)
1.1.1 Representing objects as strings	p1.2 (12)
1.1.2 Decision problems / languages	p1.3 (13)
1.1.3 Big-Oh notation	p1.3 (13)
1.2 Modeling computation and efficiency	p1.4 (14)
1.2.1 The Turing Machine	p1.5 (15)
1.2.2 Robustness of our definition.	p1.9 (19)
1.2.3 The expressive power of Turing machines.	p1.12 (22)
1.3 Machines as strings and the universal Turing machines.	p1.12 (22)
1.3.1 The Universal Turing Machine	p1.13 (23)
1.4 Uncomputable functions.	p1.15 (25)
1.4.1 The Halting Problem	p1.15 (25)
1.5 Deterministic time and the class P .	p1.17 (27)
1.5.1 On the philosophical importance of P	p1.17 (27)
1.5.2 Criticisms of P and some efforts to address them	p1.18 (28)
1.5.3 Edmonds' quote	p1.19 (29)
Chapter notes and history	p1.20 (30)
Exercises	p1.21 (31)
1.A Proof of Theorem 1.13: Universal Simulation in $O(T \log T)$ -time	p1.25 (35)
2 NP and NP completeness	p2.1 (39)
2.1 The class NP	p2.2 (40)
2.1.1 Relation between NP and P	p2.3 (41)
2.1.2 Non-deterministic Turing machines.	p2.4 (42)
2.2 Reducibility and NP-completeness	p2.5 (43)

2.3	The Cook-Levin Theorem: Computation is Local	p2.6 (44)
2.3.1	Boolean formulae and the CNF form.	p2.7 (45)
2.3.2	The Cook-Levin Theorem	p2.7 (45)
2.3.3	Warmup: Expressiveness of boolean formulae	p2.8 (46)
2.3.4	Proof of Lemma 2.12	p2.9 (47)
2.3.5	Reducing SAT to 3SAT.	p2.11 (49)
2.3.6	More thoughts on the Cook-Levin theorem	p2.11 (49)
2.4	The web of reductions	p2.12 (50)
	In praise of reductions	p2.16 (54)
	Coping with NP hardness.	p2.16 (54)
2.5	Decision versus search	p2.17 (55)
2.6	coNP , EXP and NEXP	p2.18 (56)
2.6.1	coNP	p2.18 (56)
2.6.2	EXP and NEXP	p2.19 (57)
2.7	More thoughts about P , NP , and all that	p2.20 (58)
2.7.1	The philosophical importance of NP	p2.20 (58)
2.7.2	NP and mathematical proofs	p2.20 (58)
2.7.3	What if P = NP ?	p2.21 (59)
2.7.4	What if NP = coNP ?	p2.21 (59)
	Chapter notes and history	p2.22 (60)
	Exercises	p2.23 (61)
3	Diagonalization	p3.1 (65)
3.1	Time Hierarchy Theorem	p3.2 (66)
3.2	Space Hierarchy Theorem	p3.2 (66)
3.3	Nondeterministic Time Hierarchy Theorem	p3.3 (67)
3.4	Ladner's Theorem: Existence of NP -intermediate problems.	p3.4 (68)
3.5	Oracle machines and the limits of diagonalization?	p3.6 (70)
	Chapter notes and history	p3.8 (72)
	Exercises	p3.9 (73)
4	Space complexity	p4.1 (75)
4.1	Configuration graphs.	p4.2 (76)
4.2	Some space complexity classes.	p4.4 (78)
4.3	PSPACE completeness	p4.5 (79)
4.3.1	Savitch's theorem.	p4.8 (82)
4.3.2	The essence of PSPACE : optimum strategies for game-playing.	p4.8 (82)
4.4	NL completeness	p4.10 (84)
4.4.1	Certificate definition of NL : read-once certificates	p4.12 (86)
4.4.2	NL = coNL	p4.13 (87)
	Chapter notes and history	p4.14 (88)
	Exercises	p4.14 (88)

DRAFT

5	The Polynomial Hierarchy and Alternations	p5.1 (91)
5.1	The classes Σ_2^P and Π_2^P	p5.1 (91)
5.2	The polynomial hierarchy.	p5.3 (93)
5.2.1	Properties of the polynomial hierarchy.	p5.3 (93)
5.2.2	Complete problems for levels of PH	p5.4 (94)
5.3	Alternating Turing machines	p5.5 (95)
5.3.1	Unlimited number of alternations?	p5.6 (96)
5.4	Time versus alternations: time-space tradeoffs for SAT .	p5.6 (96)
5.5	Defining the hierarchy via oracle machines.	p5.8 (98)
	Chapter notes and history	p5.9 (99)
	Exercises	p5.10 (100)
6	Circuits	p6.1 (101)
6.1	Boolean circuits	p6.1 (101)
6.1.1	Turing machines that take advice	p6.5 (105)
6.2	Karp-Lipton Theorem	p6.6 (106)
6.3	Circuit lowerbounds	p6.7 (107)
6.4	Non-uniform hierarchy theorem	p6.8 (108)
6.5	Finer gradations among circuit classes	p6.8 (108)
6.5.1	Parallel computation and NC	p6.9 (109)
6.5.2	P -completeness	p6.10 (110)
6.6	Circuits of exponential size	p6.11 (111)
6.7	Circuit Satisfiability and an alternative proof of the Cook-Levin Theorem	p6.12 (112)
	Chapter notes and history	p6.13 (113)
	Exercises	p6.13 (113)
7	Randomized Computation	p7.1 (115)
7.1	Probabilistic Turing machines	p7.2 (116)
7.2	Some examples of PTMs	p7.3 (117)
7.2.1	Probabilistic Primality Testing	p7.3 (117)
7.2.2	Polynomial identity testing	p7.4 (118)
7.2.3	Testing for perfect matching in a bipartite graph.	p7.5 (119)
7.3	One-sided and zero-sided error: RP , coRP , ZPP	p7.6 (120)
7.4	The robustness of our definitions	p7.7 (121)
7.4.1	Role of precise constants, error reduction.	p7.7 (121)
7.4.2	Expected running time versus worst-case running time.	p7.10 (124)
7.4.3	Allowing more general random choices than a fair random coin.	p7.10 (124)
7.5	Randomness efficient error reduction.	p7.11 (125)
7.6	BPP \subseteq P /poly	p7.12 (126)
7.7	BPP is in PH	p7.13 (127)
7.8	State of our knowledge about BPP	p7.14 (128)
	Complete problems for BPP ?	p7.14 (128)
	Does BPTIME have a hierarchy theorem?	p7.15 (129)

7.9	Randomized reductions	p7.15 (129)
7.10	Randomized space-bounded computation	p7.15 (129)
	Chapter notes and history	p7.17 (131)
	Exercises	p7.18 (132)
7.A	Random walks and eigenvalues	p7.21 (135)
7.A.1	Distributions as vectors and the parameter $\lambda(G)$.	p7.21 (135)
7.A.2	Analysis of the randomized algorithm for undirected connectivity.	p7.24 (138)
7.B	Expander graphs.	p7.25 (139)
7.B.1	The Algebraic Definition	p7.25 (139)
7.B.2	Combinatorial expansion and existence of expanders.	p7.27 (141)
7.B.3	Error reduction using expanders.	p7.29 (143)
8	Interactive proofs	p8.1 (147)
8.1	Warmup: Interactive proofs with a deterministic verifier	p8.1 (147)
8.2	The class IP	p8.3 (149)
8.3	Proving that graphs are <i>not</i> isomorphic.	p8.4 (150)
8.4	Public coins and AM	p8.5 (151)
8.4.1	Set Lower Bound Protocol.	p8.6 (152)
	Tool: Pairwise independent hash functions.	p8.7 (153)
	The lower-bound protocol.	p8.9 (155)
8.4.2	Some properties of IP and AM .	p8.10 (156)
8.4.3	Can GI be NP -complete?	p8.11 (157)
8.5	IP = PSPACE	p8.11 (157)
8.5.1	Arithmetization	p8.12 (158)
8.5.2	Interactive protocol for $\#\text{SAT}_D$	p8.12 (158)
	Sumcheck protocol.	p8.13 (159)
8.5.3	Protocol for TQBF : proof of Theorem 8.17	p8.14 (160)
8.6	The power of the prover	p8.15 (161)
8.7	Program Checking	p8.16 (162)
8.7.1	Languages that have checkers	p8.17 (163)
8.8	Multiprover interactive proofs (MIP)	p8.18 (164)
	Chapter notes and history	p8.19 (165)
	Exercises	p8.20 (166)
8.A	Interactive proof for the Permanent	p8.21 (167)
8.A.1	The protocol	p8.23 (169)
9	Complexity of counting	p9.1 (171)
9.1	The class $\#\mathbf{P}$	p9.2 (172)
9.1.1	The class PP : decision-problem analog for $\#\mathbf{P}$.	p9.3 (173)
9.2	$\#\mathbf{P}$ completeness.	p9.4 (174)
9.2.1	Permanent and Valiant's Theorem	p9.4 (174)
9.2.2	Approximate solutions to $\#\mathbf{P}$ problems	p9.8 (178)
9.3	Toda's Theorem: $\mathbf{PH} \subseteq \mathbf{P}^{\#\text{SAT}}$	p9.9 (179)

DRAFT

9.3.1	The class $\oplus\mathbf{P}$ and hardness of satisfiability with unique solutions.	p9.9 (179)
	Proof of Theorem 9.15	p9.11 (181)
9.3.2	Step 1: Randomized reduction from \mathbf{PH} to $\oplus\mathbf{P}$	p9.11 (181)
9.3.3	Step 2: Making the reduction deterministic	p9.13 (183)
9.4	Open Problems	p9.14 (184)
	Chapter notes and history	p9.14 (184)
	Exercises	p9.15 (185)
10	Cryptography	p10.1 (187)
10.1	Hard-on-average problems and one-way functions	p10.2 (188)
10.1.1	Discussion of the definition of one-way function	p10.4 (190)
10.1.2	Random self-reducibility	p10.5 (191)
10.2	What is a random-enough string?	p10.5 (191)
10.2.1	Blum-Micali and Yao definitions	p10.6 (192)
10.2.2	Equivalence of the two definitions	p10.8 (194)
10.3	One-way functions and pseudorandom number generators	p10.10 (196)
10.3.1	Goldreich-Levin hardcore bit	p10.10 (196)
10.3.2	Pseudorandom number generation	p10.13 (199)
10.4	Applications	p10.13 (199)
10.4.1	Pseudorandom functions	p10.13 (199)
10.4.2	Private-key encryption: definition of security	p10.14 (200)
10.4.3	Derandomization	p10.15 (201)
10.4.4	Tossing coins over the phone and bit commitment	p10.16 (202)
10.4.5	Secure multiparty computations	p10.16 (202)
10.4.6	Lowerbounds for machine learning	p10.17 (203)
10.5	Recent developments	p10.17 (203)
	Chapter notes and history	p10.17 (203)
	Exercises	p10.18 (204)
II	Lowerbounds for Concrete Computational Models	p10.21 (207)
11	Decision Trees	p11.2 (211)
11.1	Certificate Complexity	p11.4 (213)
11.2	Randomized Decision Trees	p11.6 (215)
11.3	Lowerbounds on Randomized Complexity	p11.6 (215)
11.4	Some techniques for decision tree lowerbounds	p11.8 (217)
11.5	Comparison trees and sorting lowerbounds	p11.9 (218)
11.6	Yao's MinMax Lemma	p11.9 (218)
	Exercises	p11.9 (218)
	Chapter notes and history	p11.10 (219)

12 Communication Complexity	p12.1 (221)
12.1 Definition	p12.1 (221)
12.2 Lowerbound methods	p12.2 (222)
12.2.1 Fooling set	p12.2 (222)
12.2.2 The tiling lowerbound	p12.3 (223)
12.2.3 Rank lowerbound	p12.4 (224)
12.2.4 Discrepancy	p12.5 (225)
A technique for upperbounding the discrepancy	p12.6 (226)
12.2.5 Comparison of the lowerbound methods	p12.7 (227)
12.3 Multipart communication complexity	p12.8 (228)
Discrepancy-based lowerbound	p12.9 (229)
12.4 Probabilistic Communication Complexity	p12.10 (230)
12.5 Overview of other communication models	p12.10 (230)
12.6 Applications of communication complexity	p12.11 (231)
Exercises	p12.11 (231)
Chapter notes and history	p12.12 (232)
13 Circuit lowerbounds	p13.1 (235)
13.1 \mathbf{AC}^0 and Håstad's Switching Lemma	p13.1 (235)
13.1.1 The switching lemma	p13.2 (236)
13.1.2 Proof of the switching lemma (Lemma 13.2)	p13.3 (237)
13.2 Circuits With "Counters": \mathbf{ACC}	p13.5 (239)
13.3 Lowerbounds for monotone circuits	p13.8 (242)
13.3.1 Proving Theorem 13.9	p13.8 (242)
Clique Indicators	p13.8 (242)
Approximation by clique indicators.	p13.9 (243)
13.4 Circuit complexity: The frontier	p13.11 (245)
13.4.1 Circuit lowerbounds using diagonalization	p13.11 (245)
13.4.2 Status of \mathbf{ACC} versus \mathbf{P}	p13.12 (246)
13.4.3 Linear Circuits With Logarithmic Depth	p13.13 (247)
13.4.4 Branching Programs	p13.13 (247)
13.5 Approaches using communication complexity	p13.14 (248)
13.5.1 Connection to \mathbf{ACC}^0 Circuits	p13.14 (248)
13.5.2 Connection to Linear Size Logarithmic Depth Circuits	p13.15 (249)
13.5.3 Connection to branching programs	p13.15 (249)
13.5.4 Karchmer-Wigderson communication games and depth lowerbounds	p13.15 (249)
Chapter notes and history	p13.17 (251)
Exercises	p13.18 (252)
14 Algebraic computation models	p14.1 (255)
14.1 Algebraic circuits	p14.2 (256)
14.2 Algebraic Computation Trees	p14.4 (258)
14.3 The Blum-Shub-Smale Model	p14.8 (262)

DRAFT

14.3.1 Complexity Classes over the Complex Numbers	p14.9 (263)
14.3.2 Hilbert’s Nullstellensatz	p14.10 (264)
14.3.3 Decidability Questions: Mandelbrot Set	p14.10 (264)
Exercises	p14.11 (265)
Chapter notes and history	p14.11 (265)
III Advanced topics	p14.13 (267)
15 Average Case Complexity: Levin’s Theory	p15.1 (269)
15.1 Distributional Problems	p15.2 (270)
15.1.1 Formalizations of “real-life distributions.”	p15.3 (271)
15.2 DistNP and its complete problems	p15.4 (272)
15.2.1 Polynomial-Time on Average	p15.4 (272)
15.2.2 Reductions	p15.5 (273)
15.2.3 Proofs using the simpler definitions	p15.8 (276)
15.3 Existence of Complete Problems	p15.10 (278)
15.4 Polynomial-Time Samplability	p15.10 (278)
Exercises	p15.11 (279)
Chapter notes and history	p15.11 (279)
16 Derandomization, Expanders and Extractors	p16.1 (281)
16.1 Pseudorandom Generators and Derandomization	p16.3 (283)
16.1.1 Hardness and Derandomization	p16.5 (285)
16.2 Proof of Theorem 16.10: Nisan-Wigderson Construction	p16.7 (287)
16.2.1 Warmup: two toy examples	p16.8 (288)
Extending the input by one bit using Yao’s Theorem.	p16.8 (288)
Extending the input by two bits using the averaging principle.	p16.9 (289)
Beyond two bits:	p16.10 (290)
16.2.2 The NW Construction	p16.10 (290)
Conditions on the set systems and function.	p16.11 (291)
Putting it all together: Proof of Theorem 16.10 from Lemmas 16.18 and 16.19	p16.12 (292)
Construction of combinatorial designs.	p16.13 (293)
16.3 Derandomization requires circuit lowerbounds	p16.13 (293)
16.4 Explicit construction of expander graphs	p16.16 (296)
16.4.1 Rotation maps.	p16.17 (297)
16.4.2 The matrix/path product	p16.17 (297)
16.4.3 The tensor product	p16.18 (298)
16.4.4 The replacement product	p16.19 (299)
16.4.5 The actual construction.	p16.21 (301)
16.5 Deterministic logspace algorithm for undirected connectivity.	p16.22 (302)
16.6 Weak Random Sources and Extractors	p16.25 (305)
16.6.1 Min Entropy	p16.25 (305)
16.6.2 Statistical distance and Extractors	p16.26 (306)

16.6.3	Extractors based upon hash functions	p16.27 (307)
16.6.4	Extractors based upon random walks on expanders	p16.28 (308)
16.6.5	An extractor based upon Nisan-Wigderson	p16.28 (308)
16.7	Applications of Extractors	p16.31 (311)
16.7.1	Graph constructions	p16.31 (311)
16.7.2	Running randomized algorithms using weak random sources	p16.32 (312)
16.7.3	Recycling random bits	p16.33 (313)
16.7.4	Pseudorandom generators for spacebounded computation	p16.33 (313)
	Chapter notes and history	p16.37 (317)
	Exercises	p16.38 (318)
17	Hardness Amplification and Error Correcting Codes	p17.1 (321)
17.1	Hardness and Hardness Amplification.	p17.1 (321)
17.2	Mild to strong hardness: Yao's XOR Lemma.	p17.2 (322)
	Proof of Yao's XOR Lemma using Impagliazzo's Hardcore Lemma.	p17.3 (323)
17.3	Proof of Impagliazzo's Lemma	p17.4 (324)
17.4	Error correcting codes: the intuitive connection to hardness amplification	p17.8 (328)
17.4.1	Local decoding	p17.10 (330)
17.5	Constructions of Error Correcting Codes	p17.12 (332)
17.5.1	Walsh-Hadamard Code.	p17.12 (332)
17.5.2	Reed-Solomon Code	p17.13 (333)
17.5.3	Concatenated codes	p17.14 (334)
17.5.4	Reed-Muller Codes.	p17.15 (335)
17.5.5	Decoding Reed-Solomon.	p17.16 (336)
	Randomized interpolation: the case of $\rho < 1/(d + 1)$	p17.16 (336)
	Berlekamp-Welch Procedure: the case of $\rho < (m - d)/(2m)$	p17.16 (336)
17.5.6	Decoding concatenated codes.	p17.17 (337)
17.6	Local Decoding of explicit codes.	p17.17 (337)
17.6.1	Local decoder for Walsh-Hadamard.	p17.17 (337)
17.6.2	Local decoder for Reed-Muller	p17.18 (338)
17.6.3	Local decoding of concatenated codes.	p17.19 (339)
17.6.4	Putting it all together.	p17.20 (340)
17.7	List decoding	p17.21 (341)
17.7.1	List decoding the Reed-Solomon code	p17.22 (342)
17.8	Local list decoding: getting to BPP = P	p17.23 (343)
17.8.1	Local list decoding of the Walsh-Hadamard code.	p17.24 (344)
17.8.2	Local list decoding of the Reed-Muller code	p17.24 (344)
17.8.3	Local list decoding of concatenated codes.	p17.26 (346)
17.8.4	Putting it all together.	p17.26 (346)
	Chapter notes and history	p17.27 (347)
	Exercises	p17.28 (348)

DRAFT

18 PCP and Hardness of Approximation	p18.1 (351)
18.1 PCP and Locally Testable Proofs	p18.2 (352)
18.2 PCP and Hardness of Approximation	p18.5 (355)
18.2.1 Gap-producing reductions	p18.6 (356)
18.2.2 Gap problems	p18.6 (356)
18.2.3 Constraint Satisfaction Problems	p18.7 (357)
18.2.4 An Alternative Formulation of the PCP Theorem	p18.8 (358)
18.2.5 Hardness of Approximation for 3SAT and INDSET.	p18.9 (359)
18.3 $n^{-\delta}$ -approximation of independent set is NP-hard.	p18.11 (361)
18.4 NP \subseteq PCP(poly(n), 1): PCP based upon Walsh-Hadamard code	p18.13 (363)
18.4.1 Tool: Linearity Testing and the Walsh-Hadamard Code	p18.13 (363)
18.4.2 Proof of Theorem 18.21	p18.15 (365)
18.4.3 PCP's of proximity	p18.17 (367)
18.5 Proof of the PCP Theorem.	p18.19 (369)
18.5.1 Gap Amplification: Proof of Lemma 18.29	p18.21 (371)
18.5.2 Alphabet Reduction: Proof of Lemma 18.30	p18.27 (377)
18.6 The original proof of the PCP Theorem.	p18.29 (379)
Exercises	p18.29 (379)
19 More PCP Theorems and the Fourier Transform Technique	p19.1 (385)
19.1 Parallel Repetition of PCP's	p19.1 (385)
19.2 Håstad's 3-bit PCP Theorem	p19.3 (387)
19.3 Tool: the Fourier transform technique	p19.4 (388)
19.3.1 Fourier transform over $GF(2)^n$	p19.4 (388)
The connection to PCPs: High level view	p19.6 (390)
19.3.2 Analysis of the linearity test over $GF(2)$	p19.6 (390)
19.3.3 Coordinate functions, Long code and its testing	p19.7 (391)
19.4 Proof of Theorem 19.5	p19.9 (393)
19.5 Learning Fourier Coefficients	p19.13 (397)
19.6 Other PCP Theorems: A Survey	p19.14 (398)
19.6.1 PCP's with sub-constant soundness parameter.	p19.14 (398)
19.6.2 Amortized query complexity.	p19.15 (399)
19.6.3 Unique games.	p19.15 (399)
Exercises	p19.15 (399)
20 Quantum Computation	p20.1 (401)
20.1 Quantum weirdness	p20.2 (402)
20.1.1 The 2-slit experiment	p20.2 (402)
20.1.2 Quantum entanglement and the Bell inequalities.	p20.3 (403)
20.2 A new view of probabilistic computation.	p20.5 (405)
20.3 Quantum superposition and the class BQP	p20.8 (408)
20.3.1 Universal quantum operations	p20.13 (413)
20.3.2 Spooky coordination and Bell's state	p20.13 (413)

20.4 Quantum programmer's toolkit	p20.15 (415)
20.5 Grover's search algorithm.	p20.16 (416)
20.6 Simon's Algorithm	p20.21 (421)
20.6.1 The algorithm	p20.21 (421)
20.7 Shor's algorithm: integer factorization using quantum computers.	p20.22 (422)
20.7.1 Quantum Fourier Transform over \mathbb{Z}_M	p20.23 (423)
Definition of the Fourier transform over \mathbb{Z}_M	p20.23 (423)
Fast Fourier Transform	p20.24 (424)
Quantum Fourier transform: proof of Lemma 20.20	p20.24 (424)
20.7.2 The Order-Finding Algorithm.	p20.25 (425)
Analysis: the case that $r M$	p20.26 (426)
The case that $r \nmid M$	p20.26 (426)
20.7.3 Reducing factoring to order finding.	p20.28 (428)
20.8 BQP and classical complexity classes	p20.29 (429)
Chapter notes and history	p20.29 (429)
Exercises	p20.31 (431)
20.A Rational approximation of real numbers	p20.32 (432)
21 Logic in complexity theory	p21.1 (433)
21.1 Logical definitions of complexity classes	p21.2 (434)
21.1.1 Fagin's definition of NP	p21.2 (434)
21.1.2 MAX-SNP	p21.3 (435)
21.2 Proof complexity as an approach to NP versus coNP	p21.3 (435)
21.2.1 Resolution	p21.3 (435)
21.2.2 Frege Systems	p21.4 (436)
21.2.3 Polynomial calculus	p21.4 (436)
21.3 Is P \neq NP unprovable?	p21.4 (436)
22 Why are circuit lowerbounds so difficult?	p22.1 (437)
22.1 Formal Complexity Measures	p22.1 (437)
22.2 Natural Properties	p22.3 (439)
22.3 Limitations of Natural Proofs	p22.5 (441)
22.4 My personal view	p22.6 (442)
Exercises	p22.7 (443)
Chapter notes and history	p22.7 (443)
Appendices	p22.9 (445)
A Mathematical Background.	pA.1 (447)
A.1 Mathematical Proofs	pA.1 (447)
A.2 Sets, Functions, Pairs, Strings, Graphs, Logic.	pA.3 (449)
A.3 Probability theory	pA.4 (450)
A.3.1 Random variables and expectations.	pA.5 (451)

A.3.2	The averaging argument	pA.6 (452)
A.3.3	Conditional probability and independence	pA.7 (453)
A.3.4	Deviation upperbounds	pA.7 (453)
A.3.5	Some other inequalities.	pA.9 (455)
	Jensen's inequality.	pA.9 (455)
	Approximating the binomial coefficient	pA.9 (455)
	More useful estimates.	pA.10 (456)
A.4	Finite fields and groups	pA.10 (456)
	A.4.1 Non-prime fields.	pA.11 (457)
	A.4.2 Groups.	pA.11 (457)
A.5	Vector spaces and Hilbert spaces	pA.12 (458)
A.6	Polynomials	pA.12 (458)

DRAFT