
Computational Complexity: A Modern Approach

Draft of a book in preparation: Dated December 2004
Comments welcome!

Sanjeev Arora

Not to be reproduced or distributed without the author's permission

*I am turning lecture notes from my graduate complexity course into a book.
Some chapters are more finished than others. References and attributions are
largely missing.*

About this book

Computational complexity theory has developed rapidly in the past three decades. The list of surprising and fundamental results proved in the 1990s alone could fill a book: these include new probabilistic definitions of classical complexity classes ($IP = PSPACE$ and the 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 P versus 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 is a graduate level text that aims to describe such recent achievements of complexity theory in the context of the classical results. It assumes undergraduate level knowledge of the theory of computation (e.g., from Sipser's book *Theory of Computation*). It is divided into three parts: (Part A) **Basic complexity classes:** This part covers broadly the same ground—but more quickly—as Papadimitriou's text from the early 1990s. (Part B) **Lower-bounds for concrete computational models:** This concerns lowerbounds 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 the Turing machine used in Part A, but looking deeper one finds interesting interconnections. (Part C) **Advanced topics:** This constitutes the latter half of the book and is largely devoted to the abovementioned developments from the 1990s (a couple of results date from the late 1980s).

Any successful text in this area must simultaneously cater to many audiences, and the book is carefully designed with that goal. The book chapters can largely be read in isolation except for the 100 pages in Part A. While writing the book I am keeping the following classes of readers in mind:

- *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 generally be interested in the 150-page synopsis of basic complexity theory in Part A. Progressing on to Parts B and C, 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 selfstudy or even to teach a graduate course as a way of learning the latest results. Such a course would probably include many topics from Part A and then a sprinkling from the rest of the book. I plan to include detailed course-plans they can follow (e.g., if they plan to teach an advanced result such as the PCP Theorem, they may wish to prepare the students by teaching other results first).
- *All those —professors or students— who do research in complexity theory or plan to do so.* They will largely know Part A and will use the book for Parts B and C. They would typically use Part C in a seminar or reading course. The coverage of advanced topics here is detailed enough to make this possible. For example, teaching all results related to the PCP Theorems in this book takes me at least 12 lectures of 1.5 hours each (though I will include sample teaching plans for introducing this topic in fewer lectures) and the chapter on Derandomization and Extractors takes me another 4 to 5 lectures.

The use of examples and exercises throughout the book is intended to make it suitable both for selfstudy and as a text. The appendix will include probabilistic and algebraic facts needed in the book.

Contents

1	The computational model —and why it doesn't matter	13
1.1	The Turing machine	14
1.1.1	Nondeterministic TMs	15
1.2	Turing machine: a universal computational model	16
1.2.1	The Church-Turing Thesis and its strong form.	17
I	Basic Complexity Classes	19
2	P, NP and all that	21
2.0.2	The philosophical importance of P	22
2.0.3	The philosophical importance of NP	23
2.1	Reducibility and NP-completeness	24
2.1.1	The Cook-Levin Theorem: Computation is Local	25
2.1.2	The web of reductions	27
2.2	coNP	27
2.3	EXPTIME and NEXPTIME	28
2.4	Decision versus search	29
2.5	List of some problems in P and NP	29
2.6	More thoughts about P, NP, and all that	30
2.6.1	Why 3SAT?	30
2.6.2	What if $P = NP$?	30
2.6.3	In praise of reductions	31
3	Space complexity	35
3.1	PSPACE completeness	37
3.1.1	The essence of PSPACE: optimum strategies for game-playing.	40
3.2	NL completeness	41
3.3	Two suprising algorithms	43
4	Diagonalization	49
4.1	Time Hierarchy Theorem	49
4.2	Space Hierarchy Theorem	50

4.3	Nondeterministic Time Hierarchy Theorem	51
4.4	Can diagonalization resolve P vs NP?	52
5	Alternating TM's and Polynomial Hierarchy	57
5.0.1	Complete problems for levels of PH	60
5.1	Alternating Turing machines	60
5.1.1	Unlimited number of alternations?	61
6	Circuits	65
6.0.2	Turing machines that take advice	68
6.1	Karp-Lipton Theorem	68
6.2	Circuit lowerbounds	70
6.3	Finer gradations among circuit classes	70
6.3.1	Parallel computation and NC	71
6.3.2	P-completeness	73
6.4	Circuits of exponential size	73
7	Randomized Computation	77
7.1	RP, coRP, BPP, ZPP	78
7.1.1	The robustness of our definitions	79
7.1.2	Alternative definitions	80
7.1.3	Error reduction and recycling of random bits	80
7.2	Some examples of PTMs	81
7.3	$BPP \subseteq P/poly$	84
7.4	BPP is in PH	85
7.5	State of our knowledge about BPP	86
7.5.1	Complete problems for BPP?	86
7.5.2	Does BPTIME have a hierarchy theorem?	86
7.6	Randomized reductions	87
7.7	Randomized space-bounded computation	87
7.7.1	Universal traversal sequences	89
8	Complexity classes having to do with counting	91
8.1	$\#P$ and $\#P$ -completeness	92
8.1.1	Permanent and Valiant's Theorem	93
8.1.2	Approximate solutions to $\#P$ problems	97
8.2	Toda's Theorem: $PH \subseteq P^{\#SAT}$	98
8.2.1	Step 1: Randomized reduction from PH to $\oplus SAT$	99
8.2.2	Step 2: Making the reduction deterministic	102
8.3	Open Problems	103
9	Cryptography	105
9.1	Problems that are hard on average	106
9.1.1	One-way functions: definition and discussion	108
9.1.2	Random self-reducibility	108
9.2	What is a random-enough string?	109

9.2.1	Blum-Micali and Yao definitions	110
9.2.2	Equivalence of the two definitions	111
9.3	One-way functions and pseudorandom number generators	113
9.3.1	Goldreich-Levin hardcore bit	114
9.3.2	Pseudorandom number generation	116
9.4	Applications	117
9.4.1	Tossing coins over the phone and bit commitment	117
9.4.2	Derandomization	118
9.4.3	Pseudorandom functions	119
9.4.4	Lowerbounds for machine learning	119
9.5	Recent developments	120
10	Interactive proofs	123
10.1	Warmup: Interactive proofs with a deterministic verifier	124
10.2	IP	124
10.2.1	Zero Knowledge and Cryptography	127
10.2.2	Public coins and AM	127
10.2.3	Some properties of IP and AM	129
10.3	IP = PSPACE	130
10.3.1	Arithmetization	131
10.3.2	Interactive protocol for $\#SAT_L$	131
10.3.3	Protocol for TQBF: proof of Theorem 10.4	133
10.4	Interactive proof for the Permanent	134
10.5	The power of the prover	137
10.6	Program Checking	138
10.6.1	Languages that have checkers	139
10.7	Multiprover interactive proofs (MIP)	140
II	Lowerbounds for Concrete Computational Models	143
11	Decision Trees	147
11.1	Certificate Complexity	149
11.2	Randomized Decision Trees	151
11.3	Distributional Complexity	152
12	Communication Complexity	155
12.1	Lowerbound methods	155
12.1.1	Crossing sequences	156
12.1.2	The tiling lowerbound	156
12.1.3	Rank lowerbound	158
12.2	Applications	158

13 Circuit lowerbounds	161
13.1 AC^0 and Håstad Switching Lemma	161
13.1.1 Proof of Håstad Switching Lemma	165
13.2 Circuits With “Counters”: ACC	167
13.3 Lowerbounds for monotone circuits	170
13.4 Circuit complexity: The frontier	174
13.4.1 Circuit lowerbounds using diagonalization	174
13.4.2 Status of ACC versus P	175
13.4.3 Linear Circuits With Logarithmic Depth	176
13.4.4 Branching Programs	177
13.5 Possible approaches using communication complexity	178
13.5.1 Connection to ACC^0 Circuits	179
13.5.2 Connection to Linear Size Logarithmic Depth Circuits	179
13.5.3 Connection to branching programs	180
13.5.4 Karchmer-Wigderson communication games and depth lower- bounds	180
14 Algebraic computation models	185
14.1 Algebraic Computation Trees	186
14.2 Algebraic circuits	190
14.3 The Blum-Shub-Smale Model	190
14.3.1 Complexity Classes over the Complex Numbers	191
14.3.2 Hilbert’s Nullstellensatz	192
14.3.3 Decidability Questions: Mandelbrot Set	193
III Advanced topics	195
15 Average Case Complexity: Levin’s Theory	197
15.1 Distributional Problems	198
15.1.1 Formalizations of “real-life distributions.”	199
15.2 DistNP and its complete problems	200
15.2.1 Polynomial-Time on Average	200
15.2.2 Reductions	201
15.2.3 Proofs using the simpler definitions	204
15.3 Existence of Complete Problems	206
15.4 Polynomial-Time Samplability	206
16 Amplification of Hardness	209
16.1 Hardness and how it is amplified with Yao’s Lemma	209
16.2 Proof of Impagliazzo’s Lemma	212
16.3 Hard functions from worst-case assumptions	214

17 Derandomization and Extractors	219
17.1 Pseudorandom Generators and Derandomization	221
17.1.1 Hardness and Derandomization	223
17.2 Proof of Theorem 17.2: Nisan-Wigderson Construction	223
17.2.1 The Pseudorandom Generator	224
17.3 Weak Random Sources and Extractors	228
17.3.1 An extractor based upon Nisan-Wigderson	230
17.3.2 Extractors based upon hash functions	233
17.3.3 Extractors based upon random walks on expanders	233
17.4 Applications of Extractors	234
17.4.1 Graph constructions	234
17.4.2 Running randomized algorithms using weak random sources	235
17.4.3 Recycling random bits	236
17.4.4 Pseudorandom generators for spacebounded computation	236
17.5 Derandomization requires circuit lowerbounds	241
17.5.1 Proof of Lemma 17.23: Easy witness method	243
17.6 A logspace algorithm for undirected connectivity	245
17.7 Technical Details	248
18 The PCP Theorems	253
18.1 Probabilistic checking of membership proofs	254
18.2 Relations between PCP and standard complexity classes	255
18.2.1 Two PCP Theorems	256
18.3 Approximation algorithms and hardness of approximation	257
18.4 Proof of PCP Theorem	259
18.4.1 $\text{ALG-SAT} \in \text{sPCP}(\log n, 1, \log^{O(1)} n)$	261
Low degree test	262
Reconstruction procedure	263
Proof of Theorem 18.6	265
18.4.2 $\text{ALG-SAT} \in \text{PCP}(\log^c n, 1)$: Constant-bit verifier	266
Linear function reconstruction procedure	267
Verifying the concatenation property of encodings	268
Converting a circuit to a system of quadratic equations	268
Verifying satisfiability of quadratic system by an encoded assignment	268
Proof of Theorem (18.9)	270
18.4.3 $\text{NP} \subseteq \text{PCP}(\log n, 1)$: sketch	270
18.5 Low Degree Test: Correctness	270
18.5.1 The case $m = 3$	271
18.5.2 The Bootstrapping	273
18.6 The Fourier transform technique	277
18.6.1 The discrete Fourier transform	277
The connection to PCPs: High level view	278
18.6.2 Analysis of the linearity test over $GF(2)$	279
18.6.3 Coordinate functions, Long code and its testing	280
18.6.4 Proof of Håstad's 3-bit PCP Theorem	282

18.7 Other PCP Theorems: A survey	287
18.8 More advanced techniques for constructing PCPs	288
18.9 Technical details: ALG-SAT is NP-complete	289
19 Hardness of approximating NP-hard optimization problems	295
19.1 MAX-SNP problems	295
19.2 CLIQUE	295
19.3 COLORING	295
19.4 SET-COVER	295
20 Quantum Computation	297
20.1 Quantum physics	297
20.2 Quantum superpositions	298
20.3 Classical computation using reversible gates	300
20.4 Quantum gates	301
20.4.1 Universal quantum gates	303
20.5 BQP	303
20.6 Factoring integers using a quantum computer	305
20.6.1 Uniform superpositions of eigenvectors of U	308
20.6.2 Uniform superposition suffices	309
21 Why are circuit lowerbounds so difficult?	311
21.1 Formal Complexity Measures	311
21.2 Natural Properties	313
21.3 Limitations of Natural Proofs	314
22 Logic in complexity theory	317
22.1 Logical definitions of complexity classes	318
22.1.1 Fagin's definition of NP	318
22.1.2 MAX-SNP	319
22.2 Proof complexity as an approach to NP versus coNP	319
22.2.1 Resolution	319
22.2.2 Frege Systems	319
22.2.3 Polynomial calculus	319
22.3 Is $P \neq NP$ unprovable?	319
23 The polynomial method	321
23.1 Robust interpolation of univariate polynomials	321
23.1.1 Berlekamp-Welch Procedure: $\rho < 1/2$	321
23.1.2 Sudan's list-decoding: $\rho > 1/2$	322
23.2 Self-correction for multivariate polynomials	323
23.3 Local Self-correction for multivariate polynomials	324
23.4 Application1: From worst-case to average-case hardness	324
23.4.1 An approach to hardness amplification using polynomial self-correction	324

A Appendix	327
A.1 Probability theory	327
A.1.1 The averaging argument	327
A.1.2 Deviation upperbounds	328
A.1.3 Sampling methods	330
A.1.4 The random subsum principle	331
A.2 Polynomials	331
A.2.1 Representing data by multivariate polynomials	332
A.3 Eigenvalues and expanders	333