


Texts in Computer Science

Series Editors

David Gries, Department of Computer Science, Cornell University, Ithaca, NY, USA

Orit Hazzan , Faculty of Education in Technology and Science, Technion—Israel Institute of Technology, Haifa, Israel

Titles in this series now included in the Thomson Reuters Book Citation Index!

'Texts in Computer Science' (TCS) delivers high-quality instructional content for undergraduates and graduates in all areas of computing and information science, with a strong emphasis on core foundational and theoretical material but inclusive of some prominent applications-related content. TCS books should be reasonably self-contained and aim to provide students with modern and clear accounts of topics ranging across the computing curriculum. As a result, the books are ideal for semester courses or for individual self-study in cases where people need to expand their knowledge. All texts are authored by established experts in their fields, reviewed internally and by the series editors, and provide numerous examples, problems, and other pedagogical tools; many contain fully worked solutions.

The TCS series is comprised of high-quality, self-contained books that have broad and comprehensive coverage and are generally in hardback format and sometimes contain color. For undergraduate textbooks that are likely to be more brief and modular in their approach, require only black and white, and are under 275 pages, Springer offers the flexibly designed Undergraduate Topics in Computer Science series, to which we refer potential authors.

More information about this series at <http://www.springer.com/series/3191>

Steven S. Skiena

The Algorithm Design Manual

Third Edition

 Springer

Steven S. Skiena
Department of Computer Science
Stony Brook University
Stony Brook, NY, USA

ISSN 1868-0941 ISSN 1868-095X (electronic)
Texts in Computer Science
ISBN 978-3-030-54255-9 ISBN 978-3-030-54256-6 (eBook)
<https://doi.org/10.1007/978-3-030-54256-6>

1st edition: © Springer-Verlag New York 1998

2nd edition: © Springer-Verlag London Limited 2008, Corrected printing 2012

3rd edition: © The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2020

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Many professional programmers are not well prepared to tackle algorithm design problems. This is a pity, because the techniques of algorithm design form one of the core practical *technologies* of computer science.

This book is intended as a manual on algorithm design, providing access to combinatorial algorithm technology for both students and computer professionals. It is divided into two parts: *Techniques* and *Resources*. The former is a general introduction to the design and analysis of computer algorithms. The Resources section is intended for browsing and reference, and comprises the catalog of algorithmic resources, implementations, and an extensive bibliography.

To the Reader

I have been gratified by the warm reception previous editions of *The Algorithm Design Manual* have received, with over 60,000 copies sold in various formats since first being published by Springer-Verlag in 1997. Translations have appeared in Chinese, Japanese, and Russian. It has been recognized as a unique guide to using algorithmic techniques to solve problems that often arise in practice.

Much has changed in the world since the second edition of *The Algorithm Design Manual* was published in 2008. The popularity of my book soared as software companies increasingly emphasized algorithmic questions during employment interviews, and many successful job candidates have trusted *The Algorithm Design Manual* to help them prepare for their interviews.

Although algorithm design is perhaps the most classical area of computer science, it continues to advance and change. Randomized algorithms and data structures have become increasingly important, particularly techniques based on hashing. Recent breakthroughs have reduced the algorithmic complexity of the best algorithms known for such fundamental problems as finding minimum spanning trees, graph isomorphism, and network flows. Indeed, if we date the origins of modern algorithm design and analysis to about 1970, then roughly 20% of modern algorithmic history has happened since the second edition of *The Algorithm Design Manual*.

The time has come for a new edition of my book, incorporating changes in the algorithmic and industrial world plus the feedback I have received from hundreds of readers. My major goals for the third edition are:

- To introduce or expand coverage of important topics like hashing, randomized algorithms, divide and conquer, approximation algorithms, and quantum computing in the first part of the book (Practical Algorithm Design).
- To update the reference material for all the catalog problems in the second part of the book (The Hitchhiker’s Guide to Algorithms).
- To take advantage of advances in **color printing** to produce more informative and eye-catching illustrations.

Three aspects of *The Algorithm Design Manual* have been particularly beloved: (1) the hitchhiker’s guide to algorithms, (2) the war stories, and (3) the electronic component of the book. These features have been preserved and strengthened in this third edition:

- *The Hitchhiker’s Guide to Algorithms* – Since finding out what is known about an algorithmic problem can be a difficult task, I provide a catalog of the seventy-five most important algorithmic problems arising in practice. By browsing through this catalog, the student or practitioner can quickly identify what their problem is called, what is known about it, and how they should proceed to solve it.

I have updated every section in response to the latest research results and applications. Particular attention has been paid to updating discussion of available software implementations for each problem, reflecting sources such as GitHub, which have emerged since the previous edition.

- *War stories* – To provide a better perspective on how algorithm problems arise in the real world, I include a collection of “war stories”, tales from my experience on real problems. The moral of these stories is that algorithm design and analysis is not just theory, but an important tool to be pulled out and used as needed.

The new edition of the book updates the best of the old war stories, plus adds new tales on randomized algorithms, divide and conquer, and dynamic programming.

- *Online component* – Full lecture notes and a problem solution Wiki is available on my website www.algorist.com. My algorithm lecture videos have been watched over 900,000 times on YouTube. This website has been updated in parallel with the book.

Equally important is what is not done in this book. I do not stress the mathematical analysis of algorithms, leaving most of the analysis as informal arguments. You will not find a single theorem anywhere in this book. When more details are needed, the reader should study the cited programs or references. The goal of this manual is to get you going in the right direction as quickly as possible.

To the Instructor

This book covers enough material for a standard *Introduction to Algorithms* course. It is assumed that the reader has completed the equivalent of a second programming course, typically titled *Data Structures* or *Computer Science II*.

A full set of lecture slides for teaching this course are available online at www.algorist.com. Further, I make available online video lectures using these slides to teach a full-semester algorithm course. Let me help teach your course, through the magic of the Internet!

I have made several pedagogical improvements throughout the book, including:

- *New material* – To reflect recent developments in algorithm design, I have added new chapters on randomized algorithms, divide and conquer, and approximation algorithms. I also delve deeper into topics such as hashing. But I have been careful to heed the readers who begged me to keep the book of modest length. I have (painfully) removed less important material to keep total expansion by page count under 10% over the previous edition.
- *Clearer exposition* – Reading through my text ten years later, I was thrilled to find many sections where my writing seemed ethereal, but other places that were a muddled mess. Every page in this manuscript has been edited or rewritten for greater clarity, correctness and flow.
- *More interview resources* – *The Algorithm Design Manual* remains very popular for interview prep, but this is a fast-paced world. I include more and fresher interview problems, plus coding challenges associated with interview sites like LeetCode and Hackerrank. I also include a new section with advice on how to best prepare for interviews.
- *Stop and think* – Each of my course lectures begins with a “Problem of the Day,” where I illustrate my thought process as I solve a topic-specific homework problem – false starts and all. This edition had more Stop and Think sections, which perform a similar mission for the reader.
- *More and better homework problems* – The third edition of *The Algorithm Design Manual* has more and better homework exercises than the previous one. I have added over a hundred exciting new problems, pruned some less interesting problems, and clarified exercises that proved confusing or ambiguous.
- *Updated code style* – The second edition featured algorithm implementations in C, replacing or augmenting pseudocode descriptions. These have generally been well received, although certain aspects of my programming have been condemned by some as old fashioned. All programs have been revised and updated, and are structurally highlighted in color.
- *Color images* – My companion book *The Data Science Design Manual* was printed with color images, and I was excited by how much this made

concepts clearer. Every single image in the *The Algorithm Design Manual* is now rendered in living color, and the process of review has improved the contents of most figures in the text.

Acknowledgments

Updating a book dedication every ten years focuses attention on the effects of time. Over the lifespan of this book, Renee became my wife and then the mother of our two children, Bonnie and Abby, who are now no longer children. My father has left this world, but Mom and my brothers Len and Rob remain a vital presence in my life. I dedicate this book to my family, new and old, here and departed.

I would like to thank several people for their concrete contributions to this new edition. Michael Alvin, Omar Amin, Emily Barker, and Jack Zheng were critical to building the website infrastructure and dealing with a variety of manuscript preparation issues. Their roles were played by Ricky Bradley, Andrew Gaun, Zhong Li, Betson Thomas, and Dario Vlah on previous editions. The world's most careful reader, Robert Piché of Tampere University, and Stony Brook students Peter Duffy, Olesia Elfimova, and Robert Matsibekker read early versions of this edition, and saved both you and me the trouble of dealing with many errata. Thanks also to my Springer-Verlag editors, Wayne Wheeler and Simon Rees.

Several exercises were originated by colleagues or inspired by other texts. Reconstructing the original sources years later can be challenging, but credits for each problem (to the best of my recollection) appear on the website.

Much of what I know about algorithms I learned along with my graduate students. Several of them (Yaw-Ling Lin, Sundaram Gopalakrishnan, Ting Chen, Francine Evans, Harald Rau, Ricky Bradley, and Dimitris Margaritis) are the real heroes of the war stories related within. My Stony Brook friends and algorithm colleagues Estie Arkin, Michael Bender, Jing Chen, Rezaul Chowdhury, Jie Gao, Joe Mitchell, and Rob Patro have always been a pleasure to work with.

Caveat

It is traditional for the author to magnanimously accept the blame for whatever deficiencies remain. I don't. Any errors, deficiencies, or problems in this book are somebody else's fault, but I would appreciate knowing about them so as to determine who is to blame.

Steven S. Skiena
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-2424
<http://www.cs.stonybrook.edu/~skiena>
August 2020

Contents

I	Practical Algorithm Design	1
1	Introduction to Algorithm Design	3
1.1	Robot Tour Optimization	5
1.2	Selecting the Right Jobs	8
1.3	Reasoning about Correctness	11
1.3.1	Problems and Properties	11
1.3.2	Expressing Algorithms	12
1.3.3	Demonstrating Incorrectness	13
1.4	Induction and Recursion	15
1.5	Modeling the Problem	17
1.5.1	Combinatorial Objects	17
1.5.2	Recursive Objects	19
1.6	Proof by Contradiction	21
1.7	About the War Stories	22
1.8	War Story: Psychic Modeling	22
1.9	Estimation	25
1.10	Exercises	27
2	Algorithm Analysis	31
2.1	The RAM Model of Computation	31
2.1.1	Best-Case, Worst-Case, and Average-Case Complexity . .	32
2.2	The Big Oh Notation	34
2.3	Growth Rates and Dominance Relations	37
2.3.1	Dominance Relations	38
2.4	Working with the Big Oh	39
2.4.1	Adding Functions	40
2.4.2	Multiplying Functions	40
2.5	Reasoning about Efficiency	41
2.5.1	Selection Sort	41
2.5.2	Insertion Sort	42
2.5.3	String Pattern Matching	43
2.5.4	Matrix Multiplication	45
2.6	Summations	46
2.7	Logarithms and Their Applications	48

2.7.1	Logarithms and Binary Search	49
2.7.2	Logarithms and Trees	49
2.7.3	Logarithms and Bits	50
2.7.4	Logarithms and Multiplication	50
2.7.5	Fast Exponentiation	50
2.7.6	Logarithms and Summations	51
2.7.7	Logarithms and Criminal Justice	51
2.8	Properties of Logarithms	52
2.9	War Story: Mystery of the Pyramids	54
2.10	Advanced Analysis (*)	57
2.10.1	Esoteric Functions	57
2.10.2	Limits and Dominance Relations	58
2.11	Exercises	59
3	Data Structures	69
3.1	Contiguous vs. Linked Data Structures	69
3.1.1	Arrays	70
3.1.2	Pointers and Linked Structures	71
3.1.3	Comparison	74
3.2	Containers: Stacks and Queues	75
3.3	Dictionaries	76
3.4	Binary Search Trees	81
3.4.1	Implementing Binary Search Trees	81
3.4.2	How Good are Binary Search Trees?	85
3.4.3	Balanced Search Trees	86
3.5	Priority Queues	87
3.6	War Story: Stripping Triangulations	89
3.7	Hashing	93
3.7.1	Collision Resolution	93
3.7.2	Duplicate Detection via Hashing	95
3.7.3	Other Hashing Tricks	96
3.7.4	Canonicalization	96
3.7.5	Compaction	97
3.8	Specialized Data Structures	98
3.9	War Story: String 'em Up	98
3.10	Exercises	103
4	Sorting	109
4.1	Applications of Sorting	109
4.2	Pragmatics of Sorting	113
4.3	Heapsort: Fast Sorting via Data Structures	115
4.3.1	Heaps	116
4.3.2	Constructing Heaps	118
4.3.3	Extracting the Minimum	120
4.3.4	Faster Heap Construction (*)	122
4.3.5	Sorting by Incremental Insertion	124

4.4	War Story: Give me a Ticket on an Airplane	125
4.5	Mergesort: Sorting by Divide and Conquer	127
4.6	Quicksort: Sorting by Randomization	130
4.6.1	Intuition: The Expected Case for Quicksort	132
4.6.2	Randomized Algorithms	133
4.6.3	Is Quicksort Really Quick?	135
4.7	Distribution Sort: Sorting via Bucketing	136
4.7.1	Lower Bounds for Sorting	137
4.8	War Story: Skiena for the Defense	138
4.9	Exercises	140
5	Divide and Conquer	147
5.1	Binary Search and Related Algorithms	148
5.1.1	Counting Occurrences	148
5.1.2	One-Sided Binary Search	149
5.1.3	Square and Other Roots	150
5.2	War Story: Finding the Bug in the Bug	150
5.3	Recurrence Relations	152
5.3.1	Divide-and-Conquer Recurrences	153
5.4	Solving Divide-and-Conquer Recurrences	154
5.5	Fast Multiplication	155
5.6	Largest Subrange and Closest Pair	157
5.7	Parallel Algorithms	159
5.7.1	Data Parallelism	159
5.7.2	Pitfalls of Parallelism	160
5.8	War Story: Going Nowhere Fast	161
5.9	Convolution (*)	162
5.9.1	Applications of Convolution	163
5.9.2	Fast Polynomial Multiplication (**)	164
5.10	Exercises	166
6	Hashing and Randomized Algorithms	171
6.1	Probability Review	172
6.1.1	Probability	172
6.1.2	Compound Events and Independence	174
6.1.3	Conditional Probability	175
6.1.4	Probability Distributions	176
6.1.5	Mean and Variance	176
6.1.6	Tossing Coins	177
6.2	Understanding Balls and Bins	179
6.2.1	The Coupon Collector's Problem	180
6.3	Why is Hashing a Randomized Algorithm?	181
6.4	Bloom Filters	182
6.5	The Birthday Paradox and Perfect Hashing	184
6.6	Minwise Hashing	186
6.7	Efficient String Matching	188

6.8	Primality Testing	190
6.9	War Story: Giving Knuth the Middle Initial	191
6.10	Where do Random Numbers Come From?	192
6.11	Exercises	193
7	Graph Traversal	197
7.1	Flavors of Graphs	198
7.1.1	The Friendship Graph	201
7.2	Data Structures for Graphs	203
7.3	War Story: I was a Victim of Moore's Law	207
7.4	War Story: Getting the Graph	210
7.5	Traversing a Graph	212
7.6	Breadth-First Search	213
7.6.1	Exploiting Traversal	216
7.6.2	Finding Paths	217
7.7	Applications of Breadth-First Search	217
7.7.1	Connected Components	218
7.7.2	Two-Coloring Graphs	219
7.8	Depth-First Search	221
7.9	Applications of Depth-First Search	224
7.9.1	Finding Cycles	224
7.9.2	Articulation Vertices	225
7.10	Depth-First Search on Directed Graphs	230
7.10.1	Topological Sorting	231
7.10.2	Strongly Connected Components	232
7.11	Exercises	235
8	Weighted Graph Algorithms	243
8.1	Minimum Spanning Trees	244
8.1.1	Prim's Algorithm	245
8.1.2	Kruskal's Algorithm	248
8.1.3	The Union-Find Data Structure	250
8.1.4	Variations on Minimum Spanning Trees	253
8.2	War Story: Nothing but Nets	254
8.3	Shortest Paths	257
8.3.1	Dijkstra's Algorithm	258
8.3.2	All-Pairs Shortest Path	261
8.3.3	Transitive Closure	263
8.4	War Story: Dialing for Documents	264
8.5	Network Flows and Bipartite Matching	267
8.5.1	Bipartite Matching	267
8.5.2	Computing Network Flows	268
8.6	Randomized Min-Cut	272
8.7	Design Graphs, Not Algorithms	274
8.8	Exercises	276

9	Combinatorial Search	281
9.1	Backtracking	281
9.2	Examples of Backtracking	284
9.2.1	Constructing All Subsets	284
9.2.2	Constructing All Permutations	286
9.2.3	Constructing All Paths in a Graph	287
9.3	Search Pruning	289
9.4	Sudoku	290
9.5	War Story: Covering Chessboards	295
9.6	Best-First Search	298
9.7	The A* Heuristic	300
9.8	Exercises	303
10	Dynamic Programming	307
10.1	Caching vs. Computation	308
10.1.1	Fibonacci Numbers by Recursion	308
10.1.2	Fibonacci Numbers by Caching	309
10.1.3	Fibonacci Numbers by Dynamic Programming	311
10.1.4	Binomial Coefficients	312
10.2	Approximate String Matching	314
10.2.1	Edit Distance by Recursion	315
10.2.2	Edit Distance by Dynamic Programming	317
10.2.3	Reconstructing the Path	318
10.2.4	Varieties of Edit Distance	321
10.3	Longest Increasing Subsequence	324
10.4	War Story: Text Compression for Bar Codes	326
10.5	Unordered Partition or Subset Sum	329
10.6	War Story: The Balance of Power	331
10.7	The Ordered Partition Problem	333
10.8	Parsing Context-Free Grammars	337
10.9	Limitations of Dynamic Programming: TSP	339
10.9.1	When is Dynamic Programming Correct?	340
10.9.2	When is Dynamic Programming Efficient?	341
10.10	War Story: What's Past is Prolog	342
10.11	Exercises	345
11	NP-Completeness	355
11.1	Problems and Reductions	355
11.1.1	The Key Idea	356
11.1.2	Decision Problems	357
11.2	Reductions for Algorithms	358
11.2.1	Closest Pair	358
11.2.2	Longest Increasing Subsequence	359
11.2.3	Least Common Multiple	359
11.2.4	Convex Hull (*)	360
11.3	Elementary Hardness Reductions	361

11.3.1	Hamiltonian Cycle	362
11.3.2	Independent Set and Vertex Cover	363
11.3.3	Clique	366
11.4	Satisfiability	367
11.4.1	3-Satisfiability	367
11.5	Creative Reductions from SAT	369
11.5.1	Vertex Cover	369
11.5.2	Integer Programming	371
11.6	The Art of Proving Hardness	373
11.7	War Story: Hard Against the Clock	375
11.8	War Story: And Then I Failed	377
11.9	P vs. NP	379
11.9.1	Verification vs. Discovery	380
11.9.2	The Classes P and NP	380
11.9.3	Why Satisfiability is Hard	381
11.9.4	NP-hard vs. NP-complete?	382
11.10	Exercises	383
12	Dealing with Hard Problems	389
12.1	Approximation Algorithms	389
12.2	Approximating Vertex Cover	390
12.2.1	A Randomized Vertex Cover Heuristic	392
12.3	Euclidean TSP	393
12.3.1	The Christofides Heuristic	394
12.4	When Average is Good Enough	396
12.4.1	Maximum k -SAT	396
12.4.2	Maximum Acyclic Subgraph	397
12.5	Set Cover	397
12.6	Heuristic Search Methods	399
12.6.1	Random Sampling	400
12.6.2	Local Search	402
12.6.3	Simulated Annealing	406
12.6.4	Applications of Simulated Annealing	410
12.7	War Story: Only it is Not a Radio	411
12.8	War Story: Annealing Arrays	414
12.9	Genetic Algorithms and Other Heuristics	417
12.10	Quantum Computing	418
12.10.1	Properties of “Quantum” Computers	419
12.10.2	Grover’s Algorithm for Database Search	420
12.10.3	The Faster “Fourier Transform”	422
12.10.4	Shor’s Algorithm for Integer Factorization	422
12.10.5	Prospects for Quantum Computing	424
12.11	Exercises	426
13	How to Design Algorithms	429
13.1	Preparing for Tech Company Interviews	433

II The Hitchhiker's Guide to Algorithms	435
14 A Catalog of Algorithmic Problems	437
15 Data Structures	439
15.1 Dictionaries	440
15.2 Priority Queues	445
15.3 Suffix Trees and Arrays	448
15.4 Graph Data Structures	452
15.5 Set Data Structures	456
15.6 Kd-Trees	460
16 Numerical Problems	465
16.1 Solving Linear Equations	467
16.2 Bandwidth Reduction	470
16.3 Matrix Multiplication	472
16.4 Determinants and Permanents	475
16.5 Constrained/Unconstrained Optimization	478
16.6 Linear Programming	482
16.7 Random Number Generation	486
16.8 Factoring and Primality Testing	490
16.9 Arbitrary-Precision Arithmetic	493
16.10 Knapsack Problem	497
16.11 Discrete Fourier Transform	501
17 Combinatorial Problems	505
17.1 Sorting	506
17.2 Searching	510
17.3 Median and Selection	514
17.4 Generating Permutations	517
17.5 Generating Subsets	521
17.6 Generating Partitions	524
17.7 Generating Graphs	528
17.8 Calendrical Calculations	532
17.9 Job Scheduling	534
17.10 Satisfiability	537
18 Graph Problems: Polynomial Time	541
18.1 Connected Components	542
18.2 Topological Sorting	546
18.3 Minimum Spanning Tree	549
18.4 Shortest Path	554
18.5 Transitive Closure and Reduction	559
18.6 Matching	562
18.7 Eulerian Cycle/Chinese Postman	565
18.8 Edge and Vertex Connectivity	568

18.9 Network Flow	571
18.10 Drawing Graphs Nicely	574
18.11 Drawing Trees	578
18.12 Planarity Detection and Embedding	581
19 Graph Problems: NP-Hard	585
19.1 Clique	586
19.2 Independent Set	589
19.3 Vertex Cover	591
19.4 Traveling Salesman Problem	594
19.5 Hamiltonian Cycle	598
19.6 Graph Partition	601
19.7 Vertex Coloring	604
19.8 Edge Coloring	608
19.9 Graph Isomorphism	610
19.10 Steiner Tree	614
19.11 Feedback Edge/Vertex Set	618
20 Computational Geometry	621
20.1 Robust Geometric Primitives	622
20.2 Convex Hull	626
20.3 Triangulation	630
20.4 Voronoi Diagrams	634
20.5 Nearest-Neighbor Search	637
20.6 Range Search	641
20.7 Point Location	644
20.8 Intersection Detection	648
20.9 Bin Packing	652
20.10 Medial-Axis Transform	655
20.11 Polygon Partitioning	658
20.12 Simplifying Polygons	661
20.13 Shape Similarity	664
20.14 Motion Planning	667
20.15 Maintaining Line Arrangements	671
20.16 Minkowski Sum	674
21 Set and String Problems	677
21.1 Set Cover	678
21.2 Set Packing	682
21.3 String Matching	685
21.4 Approximate String Matching	688
21.5 Text Compression	693
21.6 Cryptography	697
21.7 Finite State Machine Minimization	702
21.8 Longest Common Substring/Subsequence	706
21.9 Shortest Common Superstring	709

22 Algorithmic Resources	713
22.1 Algorithm Libraries	713
22.1.1 LEDA	713
22.1.2 CGAL	714
22.1.3 Boost Graph Library	714
22.1.4 Netlib	714
22.1.5 Collected Algorithms of the ACM	715
22.1.6 GitHub and SourceForge	715
22.1.7 The Stanford GraphBase	715
22.1.8 Combinatorica	716
22.1.9 Programs from Books	716
22.2 Data Sources	717
22.3 Online Bibliographic Resources	718
22.4 Professional Consulting Services	718
23 Bibliography	719
Index	769