大学计算机教育国外著名教材系列 影印版

The Algorithm Design Manual
Second Edition

算法设计手册
（第2版）

Steven S. Skiena 著

清华大学出版社 Springer

大学计算机教育国外著名教材系列（影印版）

The Algorithm Design Manual

Second Edition

算法设计手册

（第 2 版）

Steven S. Skiena　著

Stony Brook University

清华大学出版社
北　京

图书在版编目(CIP)数据

算法设计手册 = The Algorithm Design Manual：第 2 版：英文 /（德）斯基恩纳（Skiena, S. S.）著. —影印本. —北京：清华大学出版社，2009.9
（大学计算机教育国外著名教材系列）
ISBN 978-7-302-20727-6

Ⅰ. 算…　Ⅱ. 斯…　Ⅲ. 电子计算机－算法设计－高等学校－教材－英文　Ⅳ. TP301. 6

中国版本图书馆 CIP 数据核字（2009）第 141326 号

责任印制： 王秀菊

出版发行： 清华大学出版社　　　　**地　　址：** 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　　　**邮　　编：** 100084
社　总　机： 010-62770175　　　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者： 北京四季青印刷厂
装 订 者： 北京鑫海金澳胶印有限公司
发 行 者： 全国新华书店
开　　本： 185×230　　　　**印张：** 45.5
版　　次： 2009 年 9 月第 1 版　　　　**印　　次：** 2009 年 9 月第 1 次印刷
印　　数： 1~3000
定　　价： 69.00 元

出 版 说 明

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

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

清华大学出版社

Preface

Most professional programmers that I've encountered 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. Designing correct, efficient, and implementable algorithms for real-world problems requires access to two distinct bodies of knowledge:

- *Techniques* – Good algorithm designers understand several fundamental algorithm design techniques, including data structures, dynamic programming, depth-first search, backtracking, and heuristics. Perhaps the single most important design technique is *modeling*, the art of abstracting a messy real-world application into a clean problem suitable for algorithmic attack.

- *Resources* – Good algorithm designers stand on the shoulders of giants. Rather than laboring from scratch to produce a new algorithm for every task, they can figure out what is known about a particular problem. Rather than re-implementing popular algorithms from scratch, they seek existing implementations to serve as a starting point. They are familiar with many classic algorithmic problems, which provide sufficient source material to model most any application.

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 guide to techniques for 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 the first edition of *The Algorithm Design Manual* has received since its initial publication in 1997. It has been recognized as a unique guide to using algorithmic techniques to solve problems that often arise in practice. But much has changed in the world since the *The Algorithm Design Manual* was first published over ten years ago. Indeed, if we date the origins of modern algorithm design and analysis to about 1970, then roughly 30% of modern algorithmic history has happened since the first coming of *The Algorithm Design Manual*.

Three aspects of *The Algorithm Design Manual* have been particularly beloved: (1) the catalog of algorithmic problems, (2) the war stories, and (3) the electronic component of the book. These features have been preserved and strengthened in this edition:

- *The Catalog of Algorithmic Problems* – Since finding out what is known about an algorithmic problem can be a difficult task, I provide a catalog of the 75 most important 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. To aid in problem identification, we include a pair of "before" and "after" pictures for each problem, illustrating the required input and output specifications. One perceptive reviewer called my book "The Hitchhiker's Guide to Algorithms" on the strength of this catalog.

  The catalog is *the* most important part of this book. To update the catalog for this edition, I have solicited feedback from the world's leading experts on each associated problem. Particular attention has been paid to updating the discussion of available software implementations for each problem.

- *War Stories* – In practice, algorithm problems do not arise at the beginning of a large project. Rather, they typically arise as subproblems when it becomes clear that the programmer does not know how to proceed or that the current solution is inadequate.

  To provide a better perspective on how algorithm problems arise in the real world, we include a collection of "war stories," or tales from our experience with 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.

  This edition retains all the original war stories (with updates as appropriate) plus additional new war stories covering external sorting, graph algorithms, simulated annealing, and other topics.

- *Electronic Component* – Since the practical person is usually looking for a program more than an algorithm, we provide pointers to solid implementations whenever they are available. We have collected these implementations at one central website site (*http://www.cs.sunysb.edu/~algorith*) for easy retrieval. We have been the number one "Algorithm" site on Google pretty much since the initial publication of the book.

Further, we provide recommendations to make it easier to identify the correct code for the job. With these implementations available, the critical issue in algorithm design becomes properly modeling your application, more so than becoming intimate with the details of the actual algorithm. This focus permeates the entire book.

Equally important is what we do not do in this book. We 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. We assume 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 is available online at *http://www.algorist.com* . Further, I make available online audio and video lectures using these slides to teach a full-semester algorithm course. Let me help teach your course, by the magic of the Internet!

This book stresses design over analysis. It is suitable for both traditional lecture courses and the new "active learning" method, where the professor does not lecture but instead guides student groups to solve real problems. The "war stories" provide an appropriate introduction to the active learning method.

I have made several pedagogical improvements throughout the book. Textbook-oriented features include:

- *More Leisurely Discussion* – The tutorial material in the first part of the book has been *doubled* over the previous edition. The pages have been devoted to more thorough and careful exposition of fundamental material, instead of adding more specialized topics.

- *False Starts* – Algorithms textbooks generally present important algorithms as a fait accompli, obscuring the ideas involved in designing them and the subtle reasons why other approaches fail. The war stories illustrate such development on certain applied problems, but I have expanded such coverage into classical algorithm design material as well.

- *Stop and Think* – Here I illustrate my thought process as I solve a topic-specific homework problem—false starts and all. I have interspersed such problem blocks throughout the text to increase the problem-solving activity of my readers. Answers appear immediately following each problem.

- *More and Improved Homework Problems* – This edition of *The Algorithm Design Manual* has twice as many homework exercises as the previous one. Exercises that proved confusing or ambiguous have been improved or replaced. Degree of difficulty ratings (from 1 to 10) have been assigned to all problems.

- *Self-Motivating Exam Design* – In my algorithms courses, I promise the students that *all* midterm and final exam questions will be taken directly from homework problems in this book. This provides a "student-motivated exam," so students know exactly how to study to do well on the exam. I have carefully picked the quantity, variety, and difficulty of homework exercises to make this work; ensuring there are neither too few or too many candidate problems.

- *Take-Home Lessons* – Highlighted "take-home" lesson boxes scattered throughout the text emphasize the big-picture concepts to be gained from the chapter.

- *Links to Programming Challenge Problems* – Each chapter's exercises will contain links to 3-5 relevant "Programming Challenge" problems from *http://www.programming-challenges.com*. These can be used to add a programming component to paper-and-pencil algorithms courses.

- *More Code, Less Pseudo-code* – More algorithms in this book appear as code (written in C) instead of pseudo-code. I believe the concreteness and reliability of actual tested implementations provides a big win over less formal presentations for simple algorithms. Full implementations are available for study at *http://www.algorist.com* .

- *Chapter Notes* – Each tutorial chapter concludes with a brief notes section, pointing readers to primary sources and additional references.

Acknowledgments

Updating a book dedication after ten years focuses attention on the effects of time. Since the first edition, Renee has become my wife and then the mother of our two children, Bonnie and Abby. 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. Andrew Gaun and Betson Thomas helped in many ways, particularly in building the infrastructure for the new *http://www.cs.sunysb.edu/∼algorith*and dealing with a variety of manuscript preparation issues. David Gries offered valuable feedback well beyond the call of duty. Himanshu Gupta and Bin Tang bravely taught courses using a manuscript version of this edition. Thanks also to my Springer-Verlag editors, Wayne Wheeler and Allan Wylde.

A select group of algorithmic sages reviewed sections of the Hitchhiker's guide, sharing their wisdom and alerting me to new developments. Thanks to:

> Ami Amir, Herve Bronnimann, Bernard Chazelle, Chris Chu, Scott Cotton, Yefim Dinitz, Komei Fukuda, Michael Goodrich, Lenny Heath, Cihat Imamoglu, Tao Jiang, David Karger, Giuseppe Liotta, Albert Mao, Silvano Martello, Catherine McGeoch, Kurt Mehlhorn, Scott A. Mitchell, Naceur Meskini, Gene Myers, Gonzalo Navarro, Stephen North, Joe O'Rourke, Mike Paterson, Theo Pavlidis, Seth Pettie, Michel Pocchiola, Bart Preneel, Tomasz Radzik, Edward Reingold, Frank Ruskey, Peter Sanders, Joao Setubal, Jonathan Shewchuk, Robert Skeel, Jens Stoye, Torsten Suel, Bruce Watson, and Uri Zwick.

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.

It would be rude not to thank important contributors to the original edition. Ricky Bradley and Dario Vlah built up the substantial infrastructure required for the WWW site in a logical and extensible manner. Zhong Li drew most of the catalog figures using xfig. Richard Crandall, Ron Danielson, Takis Metaxas, Dave Miller, Giri Narasimhan, and Joe Zachary all reviewed preliminary versions of the first edition; their thoughtful feedback helped to shape what you see here.

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, Jie Gao, and Joe Mitchell have always been a pleasure to work and be with. Finally, thanks to Michael Brochstein and the rest of the city contingent for revealing a proper life well beyond Stony Brook.

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 11794-4400
http://www.cs.sunysb.edu/∼skiena
April 2008

Contents

I Practical Algorithm Design **1**

1 Introduction to Algorithm Design **3**
- 1.1 Robot Tour Optimization 5
- 1.2 Selecting the Right Jobs 9
- 1.3 Reasoning about Correctness 11
- 1.4 Modeling the Problem 19
- 1.5 About the War Stories 22
- 1.6 War Story: Psychic Modeling 23
- 1.7 Exercises 27

2 Algorithm Analysis **31**
- 2.1 The RAM Model of Computation 31
- 2.2 The Big Oh Notation 34
- 2.3 Growth Rates and Dominance Relations 37
- 2.4 Working with the Big Oh 40
- 2.5 Reasoning About Efficiency 41
- 2.6 Logarithms and Their Applications 46
- 2.7 Properties of Logarithms 50
- 2.8 War Story: Mystery of the Pyramids 51
- 2.9 Advanced Analysis (*) 54
- 2.10 Exercises 57

3 Data Structures **65**
- 3.1 Contiguous vs. Linked Data Structures 66

3.2 Stacks and Queues 71
3.3 Dictionaries 72
3.4 Binary Search Trees 77
3.5 Priority Queues 83
3.6 War Story: Stripping Triangulations 85
3.7 Hashing and Strings 89
3.8 Specialized Data Structures 93
3.9 War Story: String 'em Up 94
3.10 Exercises 98

4 Sorting and Searching **103**
4.1 Applications of Sorting 104
4.2 Pragmatics of Sorting 107
4.3 Heapsort: Fast Sorting via Data Structures 108
4.4 War Story: Give me a Ticket on an Airplane 118
4.5 Mergesort: Sorting by Divide-and-Conquer 120
4.6 Quicksort: Sorting by Randomization 123
4.7 Distribution Sort: Sorting via Bucketing 129
4.8 War Story: Skiena for the Defense 131
4.9 Binary Search and Related Algorithms 132
4.10 Divide-and-Conquer 135
4.11 Exercises 139

5 Graph Traversal **145**
5.1 Flavors of Graphs 146
5.2 Data Structures for Graphs 151
5.3 War Story: I was a Victim of Moore's Law 155
5.4 War Story: Getting the Graph 158
5.5 Traversing a Graph 161
5.6 Breadth-First Search 162
5.7 Applications of Breadth-First Search 166
5.8 Depth-First Search 169
5.9 Applications of Depth-First Search 172
5.10 Depth-First Search on Directed Graphs 178
5.11 Exercises 184

6 Weighted Graph Algorithms **191**
6.1 Minimum Spanning Trees 192
6.2 War Story: Nothing but Nets 202
6.3 Shortest Paths 205
6.4 War Story: Dialing for Documents 212
6.5 Network Flows and Bipartite Matching 217
6.6 Design Graphs, Not Algorithms 222
6.7 Exercises 225

7 Combinatorial Search and Heuristic Methods **230**
7.1 Backtracking 231
7.2 Search Pruning 238
7.3 Sudoku 239
7.4 War Story: Covering Chessboards 244
7.5 Heuristic Search Methods 247
7.6 War Story: Only it is Not a Radio 260
7.7 War Story: Annealing Arrays 263
7.8 Other Heuristic Search Methods 266
7.9 Parallel Algorithms 267
7.10 War Story: Going Nowhere Fast 268
7.11 Exercises 270

8 Dynamic Programming **273**
8.1 Caching vs. Computation 274
8.2 Approximate String Matching 280
8.3 Longest Increasing Sequence 289
8.4 War Story: Evolution of the Lobster 291
8.5 The Partition Problem 294
8.6 Parsing Context-Free Grammars 298
8.7 Limitations of Dynamic Programming: TSP 301
8.8 War Story: What's Past is Prolog 304
8.9 War Story: Text Compression for Bar Codes 307
8.10 Exercises 310

9 Intractable Problems and Approximation Algorithms **316**
9.1 Problems and Reductions 317
9.2 Reductions for Algorithms 319
9.3 Elementary Hardness Reductions 323
9.4 Satisfiability 328
9.5 Creative Reductions 330
9.6 The Art of Proving Hardness 334
9.7 War Story: Hard Against the Clock 337
9.8 War Story: And Then I Failed 339
9.9 P vs. NP 341
9.10 Dealing with NP-complete Problems 344
9.11 Exercises 350

10 How to Design Algorithms **356**

II The Hitchhiker's Guide to Algorithms **361**

11 A Catalog of Algorithmic Problems **363**

12 Data Structures **366**
12.1 Dictionaries 367
12.2 Priority Queues 373
12.3 Suffix Trees and Arrays 377
12.4 Graph Data Structures 381
12.5 Set Data Structures 385
12.6 Kd-Trees 389

13 Numerical Problems **393**
13.1 Solving Linear Equations 395
13.2 Bandwidth Reduction 398
13.3 Matrix Multiplication 401
13.4 Determinants and Permanents 404
13.5 Constrained and Unconstrained Optimization 407
13.6 Linear Programming 411
13.7 Random Number Generation 415
13.8 Factoring and Primality Testing 420
13.9 Arbitrary-Precision Arithmetic 423
13.10 Knapsack Problem 427
13.11 Discrete Fourier Transform 431

14 Combinatorial Problems **434**
14.1 Sorting 436
14.2 Searching 441
14.3 Median and Selection 445
14.4 Generating Permutations 448
14.5 Generating Subsets 452
14.6 Generating Partitions 456
14.7 Generating Graphs 460
14.8 Calendrical Calculations 465
14.9 Job Scheduling 468
14.10 Satisfiability 472

15 Graph Problems: Polynomial-Time **475**
15.1 Connected Components 477
15.2 Topological Sorting 481
15.3 Minimum Spanning Tree 484
15.4 Shortest Path 489
15.5 Transitive Closure and Reduction 495
15.6 Matching 498
15.7 Eulerian Cycle/Chinese Postman 502
15.8 Edge and Vertex Connectivity 505
15.9 Network Flow 509
15.10 Drawing Graphs Nicely 513

15.11 Drawing Trees 517
15.12 Planarity Detection and Embedding 520

16 Graph Problems: Hard Problems **523**
16.1 Clique 525
16.2 Independent Set 528
16.3 Vertex Cover 530
16.4 Traveling Salesman Problem 533
16.5 Hamiltonian Cycle 538
16.6 Graph Partition 541
16.7 Vertex Coloring 544
16.8 Edge Coloring 548
16.9 Graph Isomorphism 550
16.10 Steiner Tree 555
16.11 Feedback Edge/Vertex Set 559

17 Computational Geometry **562**
17.1 Robust Geometric Primitives 564
17.2 Convex Hull 568
17.3 Triangulation 572
17.4 Voronoi Diagrams 576
17.5 Nearest Neighbor Search 580
17.6 Range Search 584
17.7 Point Location 587
17.8 Intersection Detection 591
17.9 Bin Packing 595
17.10 Medial-Axis Transform 598
17.11 Polygon Partitioning 601
17.12 Simplifying Polygons 604
17.13 Shape Similarity 607
17.14 Motion Planning 610
17.15 Maintaining Line Arrangements 614
17.16 Minkowski Sum 617

18 Set and String Problems **620**
18.1 Set Cover 621
18.2 Set Packing 625
18.3 String Matching 628
18.4 Approximate String Matching 631
18.5 Text Compression 637
18.6 Cryptography 641
18.7 Finite State Machine Minimization 646
18.8 Longest Common Substring/Subsequence 650
18.9 Shortest Common Superstring 654

19 Algorithmic Resources **657**
19.1 Software Systems . . . 657
19.2 Data Sources . . . 663
19.3 Online Bibliographic Resources . . . 663
19.4 Professional Consulting Services . . . 664

Bibliography **665**

Part I

Practical Algorithm Design