

国外计算机科学教材系列

算法设计技巧与分析

Algorithms Design Techniques and Analysis

英文版

[沙特] M. H. Alsuwaiyel 著



电子工业出版社

Publishing House of Electronics Industry

www.phei.com.cn

国外计算机科学教材系列

算法设计技巧与分析

(英文版)

Algorithms Design Techniques and Analysis

[沙特] M.H. Alsuwaiyel 著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

版 权 声 明



World Scientific

Copyright © 1999 by World Scientific Publishing
Co. Pte. Ltd.

All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system now known or to be invented, without written permission from the Publisher.

本书英文版由新加坡World Scientific公司出版, World Scientific公司已将英文版独家版权授予中国电子工业出版社及北京美迪亚电子信息有限公司。本书仅限于在中国大陆销售。未经许可,不得以任何形式和手段复制或抄袭本书内容。

版权贸易合同登记号: 01-2002-5160

图书在版编目(CIP)数据

算法设计技巧与分析(英文版)/ (沙特)阿苏外耶(Alsuwaiyel, M. H.)著. —北京: 电子工业出版社, 2003.1

书名原文: Algorithms Design Techniques and Analysis

ISBN 7-5053-8084-2

I. 算… II. 阿… III. 算法设计—英文 IV. TP301.6

中国版本图书馆CIP数据核字(2002)第083117号

责任编辑: 吴 源

印 刷: 李家史山印刷厂

出版发行: 电子工业出版社 <http://www.phei.com.cn>

北京市海淀区万寿路173信箱 邮编: 100036

北京市海淀区翠微东里甲2号 邮编: 100036

经 销: 各地新华书店

开 本: 850×1168 1/32 印张: 17 字数: 440 千字

版 次: 2003年1月第1版 2003年1月第1次印刷

定 价: 40.00元

凡购买电子工业出版社的图书, 如有缺损问题, 请向购买书店调换, 若书店售缺, 请与本社发行部联系。联系电话: (010) 68279077

出版前言

多年来，我一直想寻找一本适合中国计算机系学生用的算法方面的国外教材。尽管有些不错的国外教材在中国出版，但总有篇幅过多、内容略显陈旧或数据结构内容夹杂其中等等这样或那样的不甚满意之处。

去年我有幸看到世界科学图书出版社出版的由M.H.Alsuwaiyel撰写的《Algorithms Design Techniques and Analysis》，它是以国际著名算法专家，我国台湾出身的李德财教授所主编的系列丛书——Lecture Notes Series on Computing——中的一本。虽然此书不是美国的大学教材，而是沙特阿拉伯的大学计算机系教材。但是我很快就被该书的组织简明、概括，且包含当前市面上算法书较少涉及的概率算法和近似算法的基本内容所吸引。它是一本适合本科生学习算法的好书。

该书涉及数据结构的部分较少，即使有一些，表述上也很快与算法中比较复杂的集合查找和合并运算等相结合，使读者不会感到和已经学过的数据结构相重复。这比较适合中国大学计算机系中数据结构和算法分成两门开设的实际状况。

对于想了解NP完全问题基本概念的读者，本书的篇幅给了他们基本的但又清楚的描绘。本书还包括计算几何一章，其取材也是适中的。

概率算法和近似算法是近20年算法研究迅猛发展的领域，本书给予了足够的重视。这也是本书特色之一，是我向中国学生特别推荐的主要原因。

本书的另一个特色是以算法的设计技术为纲，讲述一个又一个的算法技术，然后分析其算法复杂性。

我希望本书的出版能满足短期内暂时无合适中文算法教材的空白。诚挚地推荐给中国的广大算法老师采用本书作为教材。

朱 洪

2002年10月1日

复旦大学

Preface

The field of computer algorithms has flourished since the early 1960's when the first users of electronic computers started to pay attention to the performance of programs. The limited resources of computers at that time resulted in additional impetus for devising efficient computer algorithms. After extensive research in this field, numerous efficient algorithms for different problems emerged. The similarities among different algorithms for certain classes of problems have resulted in general algorithm design techniques. This book emphasizes most of these algorithm design techniques that have proved their utility in the solution to many problems. It may be considered as an attempt to cover the most common techniques in the design of sequential algorithms. Each technique is presented as follows. First, the context in which that technique can be applied. Second, the special characteristics of that technique that set it apart. Third, comparison with other techniques, whenever possible; finally, and most importantly, illustration of the technique by applying it to several problems.

Although the main theme of the book is algorithm design techniques, it also emphasizes the other major component in algorithmic design: the analysis of algorithms. It covers in detail the analysis of most of the algorithms presented. Chapter 2 covers most of the mathematical tools that are helpful in analyzing algorithms. Chapter 11 is an introduction to the field of computational complexity, and Chapter 12 covers the basics of establishing lower bounds on the solution of various problems. These chapters are indispensable for the design of efficient algorithms.

The focus of the presentation is on practical applications of the design techniques. Each technique is illustrated by providing an adequate num-

ber of algorithms to solve some problems that quite often arise in many applications in science and engineering.

The style of presentation of algorithms is straightforward, and uses pseudocode that is similar to the syntax of structured programming languages, e.g. **if-then-else**, **for** and **while** constructs. The pseudocode is sometimes intermixed with English whenever necessary. Describing a portion of an algorithm in English is indeed instructive; it conveys the idea with minimum effort on the part of the reader. However, sometimes it is both easier and more formal to use a pseudocode statement. For example, the function of the assignment statement

$$B[1..n] \leftarrow A[1..n]$$

is to replace each entry $B[i]$ with $A[i]$ for all $i, 1 \leq i \leq n$. Neither the **for ...end** for construct nor plain English is more concise or easier to state than this notation.

The book is divided into seven parts. Each part consists of chapters that cover those design techniques that have common characteristics or objectives. Part 1 sets the stage for the rest of the book, in addition to providing the background material that is needed in subsequent chapters. Part 2 is devoted to the study of recursive design techniques, which are extremely important, as they emphasize a fundamental tool in the field of computer science: recursion. Part 3 covers two intuitive and natural design techniques: the greedy approach and graph traversals. Part 4 is concerned with those techniques needed to investigate a given problem and the possibility of either coming up with an efficient algorithm for that problem, or proving its intractability. This part covers NP-completeness, computational complexity and lower bounds. In Part 5, techniques for coping with hard problems are presented. These include backtracking, randomization and finding approximate solutions that are reasonable and acceptable using a reasonable amount of time. Part 6 introduces the concept of iterative improvement using two important problems that have received extensive attention, which resulted in increasingly efficient algorithms: the problem of finding a maximum flow in a network and the problem of finding a maximum matching in an undirected graph. Finally, Part 7 is an introduction to the relatively new field of computational geometry. In one chapter, the widely used technique of geometric sweeping is presented with examples of important problems in that field. In the other chapter, the versatile tool of

the Voronoi diagram is covered, and some of its applications are presented.

The book is intended as a text in the field of the design and analysis of algorithms. It includes adequate material for two courses in algorithms. Chapters 1 through 10 provide the core material for an undergraduate course in algorithms at the junior or senior level. Some of the material may be skipped such as the amortized analysis of the union-find algorithms, and the linear time algorithms in the case of dense graphs for the shortest path and minimum spanning tree problems. The instructor may find it useful to add some of the material in the following chapters such as backtracking, randomized algorithms, approximation algorithms or geometric sweeping. The rest of the material is intended for a graduate course in algorithms.

The prerequisites for this book have been kept to the minimum; only an elementary background in discrete mathematics and data structures are assumed.

The author is grateful to King Fahd University of Petroleum & Minerals (KFUPM) for their support and providing facilities for the preparation of the manuscript. This book writing project has been funded by KFUPM under Project ics/algorithm/182. The Author would like to thank those who have critically read various portions of the manuscript and offered many helpful suggestions, including the students of the undergraduate and graduate Algorithms courses at KFUPM. Special thanks go to S. Albassam, H. Almuallim, and S. Ghanta for their valuable comments.

Dhahran, Saudi Arabia

M. H. Alsuwaiyel

Contents

Preface

vii

PART 1 Basic Concepts and Introduction to Algorithms **1**

Chapter 1 Basic Concepts in Algorithmic Analysis	5
1.1 Introduction	5
1.2 Historical Background	6
1.3 Binary Search	8
1.3.1 Analysis of the binary search algorithm	10
1.4 Merging Two Sorted Lists	12
1.5 Selection Sort	14
1.6 Insertion Sort	15
1.7 Bottom-Up Merge Sorting	17
1.7.1 Analysis of bottom-up merge sorting	19
1.8 Time Complexity	20
1.8.1 Order of growth	21
1.8.2 The O -notation	25
1.8.3 The Ω -notation	26
1.8.4 The Θ -notation	27
1.8.5 Examples	29
1.8.6 Complexity classes and the o -notation	31
1.9 Space Complexity	32
1.10 Optimal Algorithms	34

1.11	How to Estimate the Running Time of an Algorithm	35
1.11.1	Counting the number of iterations	35
1.11.2	Counting the frequency of basic operations	38
1.11.3	Using recurrence relations	41
1.12	Worst case and average case analysis	42
1.12.1	Worst case analysis	44
1.12.2	Average case analysis	46
1.13	Amortized Analysis	47
1.14	Input Size and Problem Instance	50
1.15	Exercises	52
1.16	Bibliographic Notes	59
Chapter 2	Mathematical Preliminaries	61
2.1	Sets, Relations and Functions	61
2.1.1	Sets	62
2.1.2	Relations	63
2.1.2.1	Equivalence relations	64
2.1.3	Functions	64
2.2	Proof Methods	65
2.2.1	Direct proof	65
2.2.2	Indirect proof	66
2.2.3	Proof by contradiction	66
2.2.4	Proof by counterexample	67
2.2.5	Mathematical induction	68
2.3	Logarithms	69
2.4	Floor and Ceiling Functions	70
2.5	Factorial and Binomial Coefficients	71
2.5.1	Factorials	71
2.5.2	Binomial coefficients	73
2.6	The Pigeonhole Principle	75
2.7	Summations	76
2.7.1	Approximation of summations by integration	78
2.8	Recurrence Relations	82
2.8.1	Solution of linear homogeneous recurrences	83
2.8.2	Solution of inhomogeneous recurrences	85
2.8.3	Solution of divide-and-conquer recurrences	87
2.8.3.1	Expanding the recurrence	87
2.8.3.2	Substitution	91

2.8.3.3	Change of variables	95
2.9	Exercises	98
Chapter 3 Data Structures		103
3.1	Introduction	103
3.2	Linked Lists	103
3.2.1	Stacks and queues	104
3.3	Graphs	104
3.3.1	Representation of graphs	106
3.3.2	Planar graphs	107
3.4	Trees	108
3.5	Rooted Trees	108
3.5.1	Tree traversals	109
3.6	Binary Trees	109
3.6.1	Some quantitative aspects of binary trees	111
3.6.2	Binary search trees	112
3.7	Exercises	112
3.8	Bibliographic Notes	114
Chapter 4 Heaps and the Disjoint Sets Data Structures		115
4.1	Introduction	115
4.2	Heaps	115
4.2.1	Operations on heaps	116
4.2.2	Creating a heap	120
4.2.3	Heapsort	124
4.2.4	Min and max heaps	125
4.3	Disjoint Sets Data Structures	125
4.3.1	The union by rank heuristic	127
4.3.2	Path compression	129
4.3.3	The union-find algorithms	130
4.3.4	Analysis of the union-find algorithms	132
4.4	Exercises	134
4.5	Bibliographic Notes	137
PART 2 Techniques Based on Recursion		139
Chapter 5 Induction		143

5.1	Introduction	143
5.2	Two Simple Examples	144
5.2.1	Selection sort	144
5.2.2	Insertion sort	145
5.3	Radix Sort	145
5.4	Integer Exponentiation	148
5.5	Evaluating Polynomials (Horner's Rule)	149
5.6	Generating Permutations	150
5.6.1	The first algorithm	150
5.6.2	The second algorithm	152
5.7	Finding the Majority Element	154
5.8	Exercises	155
5.9	Bibliographic Notes	158
Chapter 6	Divide and Conquer	161
6.1	Introduction	161
6.2	Binary Search	163
6.3	Mergesort	165
6.3.1	How the algorithm works	166
6.3.2	Analysis of the mergesort algorithm	167
6.4	The Divide and Conquer Paradigm	169
6.5	Selection: Finding the Median and the k th Smallest Element	172
6.5.1	Analysis of the selection algorithm	175
6.6	Quicksort	177
6.6.1	A partitioning algorithm	177
6.6.2	The sorting algorithm	179
6.6.3	Analysis of the quicksort algorithm	181
6.6.3.1	The worst case behavior	181
6.6.3.2	The average case behavior	184
6.6.4	Comparison of sorting algorithms	186
6.7	Multiplication of Large Integers	187
6.8	Matrix Multiplication	188
6.8.1	The traditional algorithm	188
6.8.2	Recursive version	188
6.8.3	Strassen's algorithm	190
6.8.4	Comparisons of the three algorithms	191
6.9	The Closest Pair Problem	192
6.9.1	Time complexity	194

6.10 Exercises	195
6.11 Bibliographic Notes	202

Chapter 7 Dynamic Programming 203

7.1 Introduction	203
7.2 The Longest Common Subsequence Problem	205
7.3 Matrix Chain Multiplication	208
7.4 The Dynamic Programming Paradigm	214
7.5 The All-Pairs Shortest Path Problem	215
7.6 The Knapsack Problem	217
7.7 Exercises	220
7.8 Bibliographic Notes	226

PART 3 First-Cut Techniques 227

Chapter 8 The Greedy Approach 231

8.1 Introduction	231
8.2 The Shortest Path Problem	232
8.2.1 A linear time algorithm for dense graphs	237
8.3 Minimum Cost Spanning Trees (Kruskal's Algorithm)	239
8.4 Minimum Cost Spanning Trees (Prim's Algorithm)	242
8.4.1 A linear time algorithm for dense graphs	246
8.5 File Compression	248
8.6 Exercises	251
8.7 Bibliographic Notes	255

Chapter 9 Graph Traversal 257

9.1 Introduction	257
9.2 Depth-First Search	257
9.2.1 Time-complexity of depth-first search	261
9.3 Applications of Depth-First Search	262
9.3.1 Graph acyclicity	262
9.3.2 Topological sorting	262
9.3.3 Finding articulation points in a graph	263
9.3.4 Strongly connected components	266
9.4 Breadth-First Search	267
9.5 Applications of Breadth-First Search	269

9.6 Exercises	270
9.7 Bibliographic Notes	273

PART 4 Complexity of Problems 275

Chapter 10 NP-Complete Problems 279

10.1 Introduction	279
10.2 The Class P	282
10.3 The Class NP	283
10.4 NP-Complete Problems	285
10.4.1 The satisfiability problem	285
10.4.2 Vertex cover, independent set and clique problems	288
10.4.3 More NP-complete Problems	291
10.5 The Class co-NP	292
10.6 The Class NPI	294
10.7 The Relationships Between the Four Classes	295
10.8 Exercises	296
10.9 Bibliographic Notes	298

Chapter 11 Introduction to Computational Complexity 299

11.1 Introduction	299
11.2 Model of Computation: The Turing Machine.	299
11.3 k -tape Turing Machines and Time complexity	300
11.4 Off-Line Turing Machines and Space Complexity	303
11.5 Tape Compression and Linear Speed-Up	305
11.6 Relationships Between Complexity Classes	306
11.6.1 Space and time hierarchy theorems	309
11.6.2 Padding arguments	311
11.7 Reductions	313
11.8 Completeness	318
11.8.1 NLOGSPACE-complete problems	318
11.8.2 PSPACE-complete problems	319
11.8.3 P-complete problems	321
11.8.4 Some conclusions of completeness	323
11.9 The Polynomial Time Hierarchy	324
11.10 Exercises	328
11.11 Bibliographic Notes	332

Chapter 12 Lower Bounds	335
12.1 Introduction	335
12.2 Trivial Lower Bounds	335
12.3 The Decision Tree Model	336
12.3.1 The search problem	336
12.3.2 The sorting problem	337
12.4 The Algebraic Decision Tree Model	339
12.4.1 The element uniqueness problem	341
12.5 Linear Time Reductions	342
12.5.1 The convex hull problem	342
12.5.2 The closest pair problem	343
12.5.3 The Euclidean minimum spanning tree problem	344
12.6 Exercises	345
12.7 Bibliographic Notes	346

PART 5 Coping with Hardness **349**

Chapter 13 Backtracking	353
13.1 Introduction	353
13.2 The 3-Coloring Problem	353
13.3 The 8-Queens Problem	357
13.4 The General Backtracking Method	360
13.5 Branch and Bound	362
13.6 Exercises	367
13.7 Bibliographic notes	369

Chapter 14 Randomized Algorithms	371
14.1 Introduction	371
14.2 Las Vegas and Monte Carlo Algorithms	372
14.3 Randomized Quicksort	373
14.4 Randomized Selection	374
14.5 Testing String Equality	377
14.6 Pattern Matching	379
14.7 Random Sampling	381
14.8 Primality Testing	384
14.9 Exercises	390
14.10 Bibliographic Notes	392

Chapter 15	Approximation Algorithms	393
15.1	Introduction	393
15.2	Basic Definitions	393
15.3	Difference Bounds	394
15.3.1	Planar graph coloring	395
15.3.2	Hardness result: the knapsack problem	395
15.4	Relative Performance Bounds	396
15.4.1	The bin packing problem	397
15.4.2	The Euclidean traveling salesman problem	399
15.4.3	The vertex cover problem	401
15.4.4	Hardness result: the traveling salesman problem	402
15.5	Polynomial Approximation Schemes	404
15.5.1	The knapsack problem	404
15.6	Fully Polynomial Approximation Schemes	407
15.6.1	The subset-sum problem	408
15.7	Exercises	410
15.8	Bibliographic Notes	413

PART 6 Iterative Improvement for Domain-Specific Problems 415

Chapter 16	Network Flow	419
16.1	Introduction	419
16.2	Preliminaries	419
16.3	The Ford-Fulkerson Method	423
16.4	Maximum Capacity Augmentation	424
16.5	Shortest Path Augmentation	426
16.6	Dinic's Algorithm	429
16.7	The MPM Algorithm	431
16.8	Exercises	434
16.9	Bibliographic Notes	436

Chapter 17	Matching	437
17.1	Introduction	437
17.2	Preliminaries	437
17.3	The Network Flow Method	440
17.4	The Hungarian Tree Method for Bipartite Graphs	441

17.5 Maximum Matching in General Graphs	443
17.6 An $O(n^{2.5})$ Algorithm for Bipartite Graphs	450
17.7 Exercises	455
17.8 Bibliographic Notes	457

PART 7 Techniques in Computational Geometry 459

Chapter 18 Geometric Sweeping 463

18.1 Introduction	463
18.2 Geometric Preliminaries	465
18.3 Computing the Intersections of Line Segments	467
18.4 The Convex Hull Problem	471
18.5 Computing the Diameter of a Set of Points	474
18.6 Exercises	478
18.7 Bibliographic Notes	480

Chapter 19 Voronoi Diagrams 481

19.1 Introduction	481
19.2 Nearest-Point Voronoi Diagram	481
19.2.1 Delaunay triangulation	484
19.2.2 Construction of the Voronoi diagram	486
19.3 Applications of the Voronoi Diagram	489
19.3.1 Computing the convex hull	489
19.3.2 All nearest neighbors.	490
19.3.3 The Euclidean minimum spanning tree	491
19.4 Farthest-Point Voronoi Diagram	492
19.4.1 Construction of the farthest-point Voronoi diagram	493
19.5 Applications of the Farthest-Point Voronoi Diagram	496
19.5.1 All farthest neighbors	496
19.5.2 Smallest enclosing circle	497
19.6 Exercises	497
19.7 Bibliographic Notes	499

Bibliography	501
------------------------	-----

Index	511
-----------------	-----