

Kurt Mehlhorn and Peter Sanders

Algorithms and Data Structures

The Basic Toolbox

August 21, 2008

Springer

ERHEBUNG
KOPF

PRELIMINARY
COPY

To all algorithmicists

ERHEBUNG
KOPF

Preface

Algorithms are at the heart of every nontrivial computer application. Therefore every computer scientist and every professional programmer should know about the basic algorithmic toolbox: structures that allow efficient organization and retrieval of data, frequently used algorithms, and generic techniques for modeling, understanding, and solving algorithmic problems.

This book is a concise introduction to this basic toolbox, intended for students and professionals familiar with programming and basic mathematical language. We have used the book in undergraduate courses on algorithmics. In our graduate-level courses, we make most of the book a prerequisite, and concentrate on the starred sections and the more advanced material. We believe that, even for undergraduates, a concise yet clear and simple presentation makes material more accessible, as long as it includes examples, pictures, informal explanations, exercises, and some linkage to the real world.

Most chapters have the same basic structure. We begin by discussing a problem as it occurs in a real-life situation. We illustrate the most important applications and then introduce simple solutions *as informally as possible and as formally as necessary* to really understand the issues at hand. When we move to more advanced and optional issues, this approach gradually leads to a more mathematical treatment, including theorems and proofs. This way, the book should work for readers with a wide range of mathematical expertise. There are also advanced sections (marked with a *) where we *recommend* that readers should skip them on first reading. Exercises provide additional examples, alternative approaches and opportunities to think about the problems. It is highly recommended to take a look at the exercises even if there is no time to solve them during the first reading. In order to be able to concentrate on ideas rather than programming details, we use pictures, words, and high-level pseudocode to explain our algorithms. A section “implementation notes” links these abstract ideas to clean, efficient implementations in real programming languages such as C++ and Java. Each chapter ends with a section on further findings that provides a glimpse at the state of the art, generalizations, and advanced solutions.

Algorithmics is a modern and active area of computer science, even at the level of the basic toolbox. We have made sure that we present algorithms in a modern

way, including explicitly formulated invariants. We also discuss recent trends, such as algorithm engineering, memory hierarchies, algorithm libraries, and certifying algorithms.

We have chosen to organize most of the material by problem domain and not by solution technique. The final chapter on optimization techniques is an exception. We find that presentation by problem domain allows a more concise presentation. However, it is also important that readers and students obtain a good grasp of the available techniques. Therefore, we have structured the final chapter by techniques, and an extensive index provides cross-references between different applications of the same technique. Bold page numbers in the Index indicate the pages where concepts are defined.

Karlsruhe, Saarbrücken,
February, 2008

*Kurt Mehlhorn
Peter Sanders*

UNIVERSITÄT
DUISBURG
ESSEN
COPY

Contents

1	Appetizer: Integer Arithmetics	1
1.1	Addition	2
1.2	Multiplication: The School Method	3
1.3	Result Checking	6
1.4	A Recursive Version of the School Method	7
1.5	Karatsuba Multiplication	9
1.6	Algorithm Engineering	11
1.7	The Programs	13
1.8	Proofs of Lemma 1.5 and Theorem 1.7	16
1.9	Implementation Notes	17
1.10	Historical Notes and Further Findings	18
2	Introduction	19
2.1	Asymptotic Notation	20
2.2	The Machine Model	23
2.3	Pseudocode	26
2.4	Designing Correct Algorithms and Programs	31
2.5	An Example – Binary Search	34
2.6	Basic Algorithm Analysis	36
2.7	Average-Case Analysis	41
2.8	Randomized Algorithms	45
2.9	Graphs	49
2.10	P and NP	53
2.11	Implementation Notes	56
2.12	Historical Notes and Further Findings	57
3	Representing Sequences by Arrays and Linked Lists	59
3.1	Linked Lists	60
3.2	Unbounded Arrays	66
3.3	*Amortized Analysis	71
3.4	Stacks and Queues	74

3.5	Lists Versus Arrays	77
3.6	Implementation Notes	78
3.7	Historical Notes and Further Findings	79
4	Hash Tables and Associative Arrays	81
4.1	Hashing with Chaining	83
4.2	Universal Hashing	85
4.3	Hashing with Linear Probing	90
4.4	Chaining Versus Linear Probing	92
4.5	*Perfect Hashing	92
4.6	Implementation Notes	95
4.7	Historical Notes and Further Findings	97
5	Sorting and Selection	99
5.1	Simple Sorters	101
5.2	Mergesort – an $O(n \log n)$ Sorting Algorithm	103
5.3	A Lower Bound	106
5.4	Quicksort	108
5.5	Selection	114
5.6	Breaking the Lower Bound	116
5.7	*External Sorting	118
5.8	Implementation Notes	122
5.9	Historical Notes and Further Findings	124
6	Priority Queues	127
6.1	Binary Heaps	129
6.2	Addressable Priority Queues	133
6.3	*External Memory	139
6.4	Implementation Notes	141
6.5	Historical Notes and Further Findings	142
7	Sorted Sequences	145
7.1	Binary Search Trees	147
7.2	(a, b) -Trees and Red–Black Trees	149
7.3	More Operations	156
7.4	Amortized Analysis of Update Operations	158
7.5	Augmented Search Trees	160
7.6	Implementation Notes	162
7.7	Historical Notes and Further Findings	164
8	Graph Representation	167
8.1	Unordered Edge Sequences	168
8.2	Adjacency Arrays – Static Graphs	168
8.3	Adjacency Lists – Dynamic Graphs	170
8.4	The Adjacency Matrix Representation	171
8.5	Implicit Representations	172

8.6	Implementation Notes	172
8.7	Historical Notes and Further Findings.....	174
9	Graph Traversal	175
9.1	Breadth-First Search	176
9.2	Depth-First Search.....	178
9.3	Implementation Notes	188
9.4	Historical Notes and Further Findings.....	189
10	Shortest Paths	191
10.1	From Basic Concepts to a Generic Algorithm	192
10.2	Directed Acyclic Graphs.....	195
10.3	Nonnegative Edge Costs (Dijkstra’s Algorithm)	196
10.4	*Average-Case Analysis of Dijkstra’s Algorithm	199
10.5	Monotone Integer Priority Queues.....	201
10.6	Arbitrary Edge Costs (Bellman–Ford Algorithm)	206
10.7	All-Pairs Shortest Paths and Node Potentials	207
10.8	Shortest-Path Queries	209
10.9	Implementation Notes	213
10.10	Historical Notes and Further Findings.....	214
11	Minimum Spanning Trees	217
11.1	Cut and Cycle Properties	218
11.2	The Jarník–Prim Algorithm	219
11.3	Kruskal’s Algorithm	221
11.4	The Union–Find Data Structure.....	222
11.5	*External Memory.....	225
11.6	Applications	228
11.7	Implementation Notes	231
11.8	Historical Notes and Further Findings.....	231
12	Generic Approaches to Optimization	233
12.1	Linear Programming – a Black-Box Solver	234
12.2	Greedy Algorithms – Never Look Back	239
12.3	Dynamic Programming – Building It Piece by Piece	243
12.4	Systematic Search – When in Doubt, Use Brute Force	246
12.5	Local Search – Think Globally, Act Locally	249
12.6	Evolutionary Algorithms	259
12.7	Implementation Notes	261
12.8	Historical Notes and Further Findings.....	262
A	Appendix	263
A.1	Mathematical Symbols	263
A.2	Mathematical Concepts	264
A.3	Basic Probability Theory	266
A.4	Useful Formulae	269

XII Contents

References 273

Index 285

PRELIMINARY COPY