

Inhaltsverzeichnis

1	Ein Anwendungsbeispiel	3
1.1	Die involvierten Daten	3
1.2	Das Mengenproblem	4
1.3	Das Sortierproblem	5
1.4	Das Kürzeste-Wege-Problem	7
1.5	Das Rundreiseproblem	10
1.6	Das Flussproblem	11
	Übungsaufgaben	13
2	Machbarkeit und Effizienz	17
2.1	Der Algorithmusbegriff	17
2.2	Grenzen der Algorithmik	21
2.3	Laufzeitüberlegungen	23
2.4	Schwierigkeit von Problemen	32
2.5	Zutaten für effiziente Algorithmen	36
	Übungsaufgaben	38
3	Einfache Ansätze	43
3.1	Datenstrukturen für das Mengenproblem	43
3.1.1	Unsortierte Ablage in einem Feld	43
3.1.2	Unsortierte Ablage in einer verketteten Liste	45
3.1.3	Unsortierte Ablage in einem dynamischen Feld	52
3.1.4	Laufzeitvergleich von Feld und Liste	54
3.2	Ein einfaches Sortierverfahren: Bubblesort	56
3.3	Backtracking als allgemeine Lösungsstrategie	59
3.4	Bedeutung für das Anwendungsszenario	62
	Übungsaufgaben	64
4	Verbesserung durch mehr Struktur	69
4.1	Sortiertes Ablegen der Daten	69
4.1.1	Sortiert in einem Feld (Array)	69
4.1.2	Sortiert in einer Liste	71
4.1.3	Sortieren durch sortiertes Einfügen	74
4.2	Skiplisten (Listen mit Abkürzungen)	77
4.3	Abkürzungen beim Sortieren (Shellsort)	81
4.4	Binäre Suchbäume	86
4.5	Strukturierte Verarbeitung von Graphen	95
4.5.1	Datenstrukturen für Graphen	95
4.5.2	Kürzeste Wege – ein vereinfachter Versuch	99

4.5.3	Ableiten von Strukturinformation	102
4.6	Rückführung auf ein einfacheres Problem	106
	Übungsaufgaben	110
5	Gierige Algorithmen	117
5.1	Idee der gierigen Algorithmen	117
5.2	Sortieren durch Auswählen	118
5.3	Kürzeste Wege: Dijkstra-Algorithmus	121
5.4	Minimale Spannbäume zur Approximation von Rundreisen .	125
5.5	Flussproblem: Ford-Fulkerson-Algorithmus	135
	Übungsaufgaben	140
6	Kleinster Schaden im Worst-Case	145
6.1	Verbesserung für den maximalen Fluss	145
6.2	Mengenproblem: balancierte Bäume	148
6.3	Sortieren mit bestmöglicher asymptotischer Laufzeit	165
	Übungsaufgaben	169
7	Teile und Beherrsche	175
7.1	Sortieren durch Mischen	175
7.2	Laufzeitanalyse bei »Teile und Beherrsche«	180
7.3	Quicksort	187
7.4	Suche nach dem Median	193
	Übungsaufgaben	199
8	Dynamisches Programmieren	205
8.1	Idee des dynamischen Programmierens	205
8.2	Alle kürzesten Wege nach Floyd-Warshall	207
8.3	Optimale Suchbäume	211
8.4	Straight-Mergesort	220
	Übungsaufgaben	223
9	Direkter Zugriff	227
9.1	Interpolationssuche	227
9.2	Sortieren durch Abzählen	229
9.3	Radix-Sort	232
9.4	Mengenproblem: Hash-Tabellen	235
9.4.1	Hash-Code	235
9.4.2	Hash-Funktion	238
9.4.3	Ansätze der Kollisionsbehandlung	239
9.4.4	Suchen, Einfügen und Löschen in Hash-Tabellen	242
9.4.5	Varianten des geschlossenen Hashings	245
	Übungsaufgaben	252
10	Prioritätswarteschlangen	257
10.1	Motivation	257

10.2 Binäre Heaps	259
10.2.1 Einfügen und Minimum-Löschen	260
10.2.2 Verringern eines Prioritätswerts	263
10.3 Heapsort	266
Übungsaufgaben	274
11 Extern gespeicherte Daten	277
11.1 Zugriff auf externe Speichermedien	277
11.2 Sortierproblem: Mehrphasen-Mergesort	279
11.3 Mengenproblem: B-Bäume	282
11.4 Mengenproblem: Erweiterbares Hashing	297
Übungsaufgaben	305
12 Selbstorganisation	309
12.1 Mengenproblem: Selbstorganisierende Liste	309
12.2 Rundreiseproblem: Evolutionäre Algorithmen	312
Übungsaufgaben	321
13 Zusammenfassung	325
13.1 Mengenproblem	325
13.2 Sortieren	327
13.3 Kürzeste Wege	328
13.4 Rundreise	329
13.5 Maximaler Fluss	330
Anhang	331
A Notation des Pseudo-Code	333
B Mathematische Grundlagen	339
B.1 Summen und Reihen	339
B.2 Fibonacci-Zahlen	339
B.3 Logarithmen	340
B.4 Fakultät	340
Literaturverzeichnis	341
Liste der Algorithmen	347
Stichwortverzeichnis	351

Algorithmen und Datenstrukturen

Weicker, K.; Weicker, N.

2013, IX, 356 S. 91 Abb. in Farbe., Softcover

ISBN: 978-3-8348-1238-4