

Inhaltsverzeichnis

1	Algorithmen-Grundlagen und Algorithmen-Implementierung	1
1.1	Laufzeitanalyse von Algorithmen	1
1.1.1	Landau-Symbole	1
1.1.2	Worst-Case, Average-Case und amortisierte Laufzeit	4
1.1.3	Praktisch lösbar vs. exponentielle Laufzeit	4
1.2	Implementierung von Algorithmen	6
1.2.1	Rekursive vs. iterative Implementierung	6
1.2.2	Warum Rekursion (statt Iteration)?	12
1.2.3	„Kochrezept“ für das Entwickeln eines rekursiven Algorithmus	12
1.3	Nicht-destructive vs. In-place Implementierung	13
1.3.1	Warum nicht-destructive Implementierungen?	14
1.4	Repräsentation von Datenstrukturen	14
1.4.1	Repräsentation als Klasse	15
1.4.2	Repräsentation als Liste	15
1.4.3	Repräsentation als Dictionary	15
2	Sortieralgorithmen	17
2.1	Insertion Sort	17
2.1.1	Implementierung: nicht-destructiv	17
2.1.2	In-place Implementierung	19
2.1.3	Laufzeit	19
2.2	Mindestlaufzeit von Sortieralgorithmen	21
2.3	Quicksort	22
2.3.1	Divide-And-Conquer-Algorithmen	22
2.3.2	Funktionsweise von Quicksort	23
2.3.3	Laufzeit	26
2.3.4	In-Place-Implementierung	27
2.3.5	Eliminierung der Rekursion	30
2.4	Mergesort	33
2.5	Heapsort und Priority Search Queues	34
2.5.1	Repräsentation von Heaps	34
2.5.2	Heaps als Priority Search Queues	35
2.5.3	Konstruktion eines Heaps	39
2.5.4	Heapsort	43

3	Suchalgorithmen	47
3.1	Binäre Suchbäume	49
3.1.1	Repräsentation eines binären Suchbaums	50
3.1.2	Suchen, Einfügen, Löschen	51
3.1.3	Laufzeit	56
3.2	AVL-Bäume	57
3.2.1	Einfügeoperation	58
3.2.2	Grundlegende Balancierungsoperationen: Rotationen	59
3.3	Rot-Schwarz-Bäume	63
3.3.1	Einfügen	64
3.3.2	Löschen	69
3.4	Hashing	72
3.4.1	Hash-Funktionen	73
3.4.2	Kollisionsbehandlung	77
3.4.3	Implementierung in Python	79
3.5	Bloomfilter	85
3.5.1	Grundlegende Funktionsweise	85
3.5.2	Implementierung	87
3.5.3	Laufzeit und Wahrscheinlichkeit falsch-positiver Antworten	89
3.5.4	Anwendungen von Bloomfiltern	91
3.6	Skip-Listen	93
3.6.1	Implementierung	94
3.6.2	Laufzeit	98
3.7	Tries	100
3.7.1	Die Datenstruktur	100
3.7.2	Suche	102
3.7.3	Einfügen	103
3.8	Patricia-Tries	104
3.8.1	Datenstruktur	104
3.8.2	Suche	105
3.8.3	Einfügen	106
3.9	Suchmaschinen	108
3.9.1	Aufbau einer Suchmaschine	108
3.9.2	Invertierter Index	109
3.9.3	Implementierung	109
3.9.4	Erweiterte Anforderungen	111
4	Heaps	115
4.1	Binäre Heaps	116
4.1.1	Repräsentation binärer Heaps	116
4.1.2	Einfügen eines Elements	117
4.1.3	Minimumsextraktion	117
4.1.4	Erhöhen eines Schlüsselwertes	118

4.2	Binomial-Heaps	119
4.2.1	Binomial-Bäume	120
4.2.2	Repräsentation von Binomial-Bäumen	120
4.2.3	Struktur von Binomial-Heaps	121
4.2.4	Repräsentation von Binomial-Heaps	122
4.2.5	Verschmelzung zweier Binomial-Bäume	122
4.2.6	Vereinigung zweier Binomial-Heaps	123
4.2.7	Einfügen eines Elements	126
4.2.8	Extraktion des Minimums	126
4.3	Fibonacci Heaps	127
4.3.1	Struktur eines Fibonacci-Heaps	128
4.3.2	Repräsentation in Python	129
4.3.3	Amortisierte Laufzeit und Potenzialfunktion	131
4.3.4	Verschmelzung	131
4.3.5	Einfügen	132
4.3.6	Extraktion des Minimums	133
4.3.7	Erniedrigen eines Schlüsselwertes	136
4.3.8	Maximale Ordnung eines Fibonacci-Baums	141
4.4	Pairing-Heaps	142
4.4.1	Struktur und Repräsentation in Python	142
4.4.2	Einfache Operationen auf Pairing-Heaps	143
4.4.3	Extraktion des Minimums	144
5	Graphalgorithmen	147
5.1	Grundlegendes	147
5.1.1	Wozu Graphen?	147
5.1.2	Repräsentation von Graphen	149
5.2	Breiten- und Tiefensuche	152
5.2.1	Breitensuche	152
5.2.2	Tiefensuche	154
5.2.3	Topologische Sortierung	159
5.3	Kürzeste Wege	161
5.3.1	Der Dijkstra-Algorithmus	162
5.3.2	Der Warshall-Algorithmus	165
5.4	Minimaler Spannbaum	169
5.4.1	Problemstellung	169
5.4.2	Der Algorithmus von Kruskal	170
5.4.3	Union-Find-Operationen	174
5.5	Maximaler Fluss in einem Netzwerk	178
5.5.1	Netzwerke und Flüsse	178
5.5.2	Der Algorithmus von Ford-Fulkerson	179
5.5.3	Korrektheit des Ford-Fulkerson-Algorithmus	182

6	Formale Sprachen und Parser	185
6.1	Formale Sprachen und Grammatiken	185
6.1.1	Formales Alphabet, formale Sprache	185
6.1.2	Grammatik, Ableitung, akzeptierte Sprache, Syntaxbaum	186
6.2	Repräsentation einer Grammatik in Python	190
6.2.1	Berechnung der FIRST-Mengen	192
6.2.2	Berechnung der FOLLOW-Mengen	195
6.3	Recursive-Descent-Parser	197
6.3.1	Top-Down-Parsing	197
6.3.2	Prädiktives Parsen	198
6.3.3	Implementierung eines Recursive-Descent-Parsers	199
6.3.4	Vorsicht: Linksrekursion	201
6.4	Ein LR-Parsergenerator	202
6.4.1	LR(0)-Elemente	203
6.4.2	Die Hülleoperation	203
6.4.3	Die GOTO-Operation	204
6.4.4	Erzeugung des Präfix-Automaten	205
6.4.5	Berechnung der Syntaxanalysetabelle	208
6.4.6	Der Kellerautomat	210
7	Stringmatching	213
7.1	Primitiver Algorithmus	213
7.2	Stringmatching mit endlichen Automaten	214
7.3	Der Knuth-Morris-Pratt-Algorithmus	216
7.3.1	Suche mit Hilfe der Verschiebetabelle	217
7.3.2	Laufzeit	219
7.3.3	Berechnung der Verschiebetabelle	220
7.4	Der Boyer-Moore-Algorithmus	221
7.4.1	Die Bad-Character-Heuristik	221
7.4.2	Die Good-Suffix-Heuristik	224
7.4.3	Implementierung	227
7.4.4	Laufzeit	228
7.5	Der Rabin-Karp-Algorithmus	228
7.5.1	Rollender Hash	229
7.5.2	Implementierung	231
7.6	Der Shift-Or-Algorithmus	232
7.6.1	Implementierung	234
8	Schwere Probleme und Heuristiken	237
8.1	Das Travelling-Salesman-Problem	237
8.1.1	Lösung durch Ausprobieren	237

8.1.2	Lösung durch Dynamische Programmierung	238
8.1.3	Laufzeit	240
8.2	Heuristiken für das Travelling-Salesman-Problem	241
8.3	Greedy-Heuristiken	241
8.3.1	Nearest-Neighbor-Heuristik	241
8.3.2	Nearest-, Farthest-, Random-Insertion	242
8.3.3	Tourverschmelzung	244
8.4	Lokale Verbesserung	246
8.4.1	Die 2-Opt-Heuristik	247
8.4.2	Die 2.5-Opt-Heuristik	248
8.4.3	Die 3-Opt- und k -Opt-Heuristik	250
8.5	Ein Genetischer Algorithmus	255
8.5.1	Knoten-Cross-Over	255
8.5.2	Kanten-Cross-Over	255
8.5.3	Die Realisierung des genetischen Algorithmus	257
8.6	Ein Ameisen-Algorithmus	258
8.6.1	Erster Ansatz	260
8.6.2	Verbesserte Umsetzung	263
A	Python Grundlagen	267
A.1	Die Pythonshell	267
A.2	Einfache Datentypen	267
A.2.1	Zahlen	267
A.2.2	Strings	268
A.2.3	Variablen	268
A.2.4	Typisierung	268
A.2.5	Operatoren	269
A.3	Grundlegende Konzepte	270
A.3.1	Kontrollfluss	270
A.3.2	Schleifenabbruch	272
A.3.3	Anweisungen vs. Ausdrücke	273
A.3.4	Funktionen	274
A.3.5	Referenzen	276
A.4	Zusammengesetzte Datentypen	277
A.4.1	Listen	277
A.4.2	Sequenzen	279
A.4.3	Tupel	282
A.4.4	Dictionaries	283
A.4.5	Strings (Fortsetzung)	285
A.4.6	Mengen: Der <i>set</i> -Typ	286
A.5	Funktionale Programmierung	287
A.5.1	Listenkomprehensionen	288

A.5.2	Lambda-Ausdrücke	290
A.5.3	Die <i>map</i> -Funktion	291
A.5.4	Die <i>all</i> - und die <i>any</i> -Funktion	292
A.5.5	Die <i>enumerate</i> -Funktion	292
A.5.6	Die <i>reduce</i> -Funktion	293
A.6	Vergleichen und Sortieren	295
A.6.1	Vergleichen	295
A.6.2	Sortieren	296
A.7	Objektorientierte Programmierung	298
A.7.1	Spezielle Methoden	301
B	Mathematische Grundlagen	303
B.1	Mengen, Tupel, Relationen	303
B.1.1	Mengen	303
B.1.2	Tupel	303
B.1.3	Relationen	304
B.1.4	Vollständige Induktion	306
B.1.5	Summenformel	306
B.2	Fibonacci-Zahlen	307
B.3	Grundlagen der Stochastik	309
B.3.1	Wahrscheinlichkeitsraum	309
B.3.2	Laplacesches Prinzip	310
B.3.3	Zufallsvariablen und Erwartungswert	311
B.3.4	Wichtige Verteilungen	312
B.4	Graphen, Bäume und Netzwerke	314
B.4.1	Graphen	314
B.5	Potenzmengen	316
B.5.1	Permutationen	317
B.5.2	Teilmengen und Binomialkoeffizient	319
Literaturverzeichnis		321
Index		323