

Aditya Y. Bhargava

# Algorithmen kapiere

**Visuell lernen und verstehen**  
mit Illustrationen, Alltagsbeispielen und Python-Code

Übersetzung aus dem Amerikanischen  
von Knut Lorenzen



# Inhaltsverzeichnis



	<b>Vorwort</b> .....	11
	<b>Einleitung</b> .....	13
	Überblick .....	14
	Verwendung dieses Buchs .....	15
	Wer sollte dieses Buch lesen? .....	15
	Konventionen und Downloads .....	16
	<b>Über den Autor</b> .....	16
	<b>Danksagungen</b> .....	17
1	<b>Einführung in Algorithmen</b> .....	19
1.1	Einführung .....	19
1.1.1	Performance .....	20
1.1.2	Problemlösungen .....	20
1.2	Binäre Suche .....	21
1.2.1	Eine bessere Suchmethode .....	23
	✍  Übungen .....	27
1.2.2	Laufzeit .....	28
1.3	Landau-Notation .....	29
1.3.1	Die Laufzeiten von Algorithmen nehmen unterschiedlich schnell zu .....	29
1.3.2	Visualisierung verschiedener Laufzeiten .....	32
1.3.3	Die Landau-Notation beschreibt die Laufzeit im Worst Case .....	33

1.3.4	Typische Laufzeiten gebräuchlicher Algorithmen . . . . .	34
	✎ Übungen . . . . .	35
1.3.5	Das Problem des Handlungsreisenden . . . . .	36
1.4	Zusammenfassung . . . . .	38
2	<b>Selectionsort</b> . . . . .	39
2.1	Die Funktionsweise des Arbeitsspeichers . . . . .	40
2.2	Arrays und verkettete Listen . . . . .	42
	2.2.1 Verkettete Listen . . . . .	43
	2.2.2 Arrays . . . . .	44
	2.2.3 Terminologie . . . . .	45
	✎ Übung . . . . .	46
	2.2.4 Einfügen in der Mitte einer Liste . . . . .	47
	2.2.5 Löschen . . . . .	48
	✎ Übungen . . . . .	49
2.3	Selectionsort . . . . .	51
	Beispielcode . . . . .	55
2.4	Zusammenfassung . . . . .	56
3	<b>Rekursion</b> . . . . .	57
3.1	Rekursion . . . . .	58
3.2	Basisfall und Rekursionsfall . . . . .	61
3.3	Der Stack . . . . .	62
	3.3.1 Der Aufruf-Stack . . . . .	63
	✎ Übung . . . . .	66
	3.3.2 Der Aufruf-Stack mit Rekursion . . . . .	66
	✎ Übung . . . . .	70
3.4	Zusammenfassung . . . . .	70
4	<b>Quicksort</b> . . . . .	71
4.1	Teile und herrsche . . . . .	72
	✎ Übungen . . . . .	79
4.2	Quicksort . . . . .	80
4.3	Landau-Notation im Detail . . . . .	85
	4.3.1 Mergesort und Quicksort im Vergleich . . . . .	86
	4.3.2 Average Case und Worst Case im Vergleich . . . . .	88
	✎ Übungen . . . . .	92
4.4	Zusammenfassung . . . . .	92
5	<b>Hashtabellen</b> . . . . .	93
5.1	Hashfunktionen . . . . .	96
	✎ Übungen . . . . .	99

5.2	Anwendungsfälle . . . . .	100
5.2.1	Hashtabellen zum Nachschlagen verwenden . . . . .	100
5.2.2	Doppelte Einträge verhindern . . . . .	102
5.2.3	Hashtabellen als Cache verwenden . . . . .	104
5.2.4	Zusammenfassung . . . . .	107
5.2.5	Kollisionen . . . . .	107
5.3	Performance . . . . .	110
5.3.1	Der Auslastungsfaktor . . . . .	112
5.3.2	Eine gute Hashfunktion . . . . .	114
	✎ Übungen . . . . .	114
5.4	Zusammenfassung . . . . .	115
<b>6</b>	<b>Breitensuche . . . . .</b>	<b>117</b>
6.1	Einführung in Graphen . . . . .	118
6.2	Was ist ein Graph? . . . . .	120
6.3	Breitensuche . . . . .	121
6.3.1	Den kürzesten Pfad finden . . . . .	124
6.3.2	Warteschlangen . . . . .	126
	✎ Übungen . . . . .	127
6.4	Implementierung des Graphen . . . . .	127
6.5	Implementierung des Algorithmus . . . . .	130
6.5.1	Laufzeit . . . . .	135
	✎ Übung . . . . .	135
6.6	Zusammenfassung . . . . .	138
<b>7</b>	<b>Der Dijkstra-Algorithmus . . . . .</b>	<b>139</b>
7.1	Anwendung des Dijkstra-Algorithmus . . . . .	140
7.2	Terminologie . . . . .	145
7.3	Eintauschen gegen ein Klavier . . . . .	147
7.4	Negativ gewichtete Kanten . . . . .	154
7.5	Implementierung . . . . .	157
	✎ Übung . . . . .	167
7.6	Zusammenfassung . . . . .	168
<b>8</b>	<b>Greedy-Algorithmen . . . . .</b>	<b>169</b>
8.1	Das Stundenplanproblem . . . . .	169
8.2	Das Rucksackproblem . . . . .	172
	✎ Übungen . . . . .	174
8.3	Das Mengenüberdeckungsproblem . . . . .	174
8.3.1	Approximationsalgorithmen . . . . .	175
	✎ Übungen . . . . .	181

8.4	NP-vollständige Probleme . . . . .	181
8.5	Das Problem des Handlungsreisenden – Schritt für Schritt . . . . .	183
8.5.1	Wie lassen sich NP-vollständige Probleme erkennen? . . . . .	187
	✍ Übungen . . . . .	189
8.6	Zusammenfassung . . . . .	189
<b>9</b>	<b>Dynamische Programmierung</b> . . . . .	<b>191</b>
9.1	Das Rucksackproblem . . . . .	191
9.1.1	Die einfache Lösung . . . . .	192
9.1.2	Dynamische Programmierung . . . . .	193
9.2	Häufig gestellte Fragen zum Rucksackproblem . . . . .	201
9.2.1	Was geschieht beim Hinzufügen eines Gegenstands? . . . . .	201
	✍ Übung. . . . .	204
9.2.2	Was geschieht, wenn die Reihenfolge der Zeilen geändert wird? . . . . .	204
9.2.3	Kann man das Gitter auch spaltenweise (statt zeilenweise) befüllen? . . . . .	205
9.2.4	Was geschieht, wenn man ein leichteres Objekt hinzufügt? . . . . .	205
9.2.5	Kann man Teile eines Gegenstands stehlen? . . . . .	206
9.2.6	Optimierung des Reiseplans . . . . .	206
9.2.7	Handhabung voneinander abhängiger Objekte . . . . .	208
9.2.8	Ist es möglich, dass die Lösung mehr als zwei Teil-Rucksäcke erfordert? . . . . .	209
9.2.9	Ist es möglich, dass die beste Lösung den Rucksack nicht vollständig füllt? . . . . .	209
	✍ Übung. . . . .	209
9.3	Der längste gemeinsame Teilstring . . . . .	210
9.3.1	Erstellen des Gitters . . . . .	211
9.3.2	Befüllen des Gitters . . . . .	212
9.3.3	Die Lösung . . . . .	213
9.3.4	Die längste gemeinsame Teilfolge . . . . .	214
9.3.5	Die längste gemeinsame Teilfolge – Lösung . . . . .	216
	✍ Übung. . . . .	217
9.4	Zusammenfassung . . . . .	217
<b>10</b>	<b>k-nächste Nachbarn</b> . . . . .	<b>219</b>
10.1	Klassifikation von Orangen und Grapefruits . . . . .	219
10.2	Entwicklung eines Empfehlungssystems . . . . .	221
10.2.1	Merkmalsextraktion . . . . .	223
	✍ Übungen . . . . .	227
10.2.2	Regression . . . . .	227

10.2.3	Auswahl geeigneter Merkmale . . . . .	230
	✍ Übung . . . . .	230
10.3	Einführung in Machine Learning. . . . .	231
10.3.1	OCR . . . . .	231
10.3.2	Entwicklung eines Spamfilters . . . . .	232
10.3.3	Vorhersage der Entwicklung des Aktienmarkts. . . . .	233
10.4	Zusammenfassung . . . . .	233
11	<b>Die nächsten Schritte</b> . . . . .	235
11.1	Bäume . . . . .	235
11.2	Invertierte Indizes . . . . .	238
11.3	Die Fourier-Transformation . . . . .	239
11.4	Nebenläufige Algorithmen . . . . .	240
11.5	MapReduce . . . . .	241
	11.5.1 Warum sind verteilte Algorithmen nützlich? . . . . .	241
	11.5.2 Die map-Funktion . . . . .	242
	11.5.3 Die reduce-Funktion . . . . .	242
11.6	Bloom-Filter und HyperLogLog . . . . .	243
	11.6.1 Bloom-Filter . . . . .	245
	11.6.2 HyperLogLog . . . . .	245
11.7	Die SHA-Algorithmen. . . . .	246
	11.7.1 Dateien vergleichen . . . . .	246
	11.7.2 Passwörter überprüfen. . . . .	247
11.8	Locality-Sensitive Hashing . . . . .	248
11.9	Diffie-Hellman-Schlüsselaustausch . . . . .	249
11.10	Lineare Programmierung . . . . .	250
11.11	Epilog . . . . .	251
	<b>Lösungen zu den Übungen</b> . . . . .	253
	Kapitel 1 . . . . .	253
	Kapitel 2 . . . . .	254
	Kapitel 3 . . . . .	257
	Kapitel 4 . . . . .	258
	Kapitel 5 . . . . .	259
	Kapitel 6 . . . . .	260
	Kapitel 7 . . . . .	262
	Kapitel 8 . . . . .	263
	Kapitel 9 . . . . .	264
	Kapitel 10 . . . . .	265
	<b>Stichwortverzeichnis</b> . . . . .	267