Grundkurs Algorithmen und Datenstrukturen in JAVA

Andreas Solymosi • Ulrich Grude

Grundkurs Algorithmen und Datenstrukturen in JAVA

Eine Einführung in die praktische Informatik

6., aktualisierte und ergänzte Auflage

Springer Vieweg

Andreas Solymosi
Beuth-Hochschule für Technik Berlin
Berlin, Deutschland

Ulrich Grude
Beuth-Hochschule für Technik Berlin
Berlin, Deutschland

ISBN 978-3-658-17545-0 ISBN 978-3-658-17546-7 (eBook)
DOI 10.1007/978-3-658-17546-7

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist Teil von Springer Nature
Die eingetragene Gesellschaft ist Springer Fachmedien Wiesbaden GmbH
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Vorwort zur 6. Auflage

Nachdem die 3. Auflage vergriffen war, haben wir das Buch für die Version 5 von Java überarbeitet, um die Möglichkeiten der ausdrucksreicheren Sprache auszunutzen: Generizität, Aufzählungstypen und die Zählschleife („verbesserte for-Schleife“) wurden benutzt, um die Algorithmen noch verständlicher darzustellen. Jetzt, vor der 6. Auflage, wurde Java nochmals bereichert: Die „Lambdas“ kamen in der Version 8 dazu, die viele Algorithmen wiederum vereinfachen. Also wurde das Buch weiter ergänzt. Leser, die diese neuen Sprachelemente noch nicht kennen, können mit Hilfe der (knappen) Erklärungen und Fußnoten einen Einblick bekommen, um sich mit ihnen vertraut zu machen. Die Sachkundigen können gerne vorschlagen, wie die dargestellten Algorithmen mit dem neuen Paradigma („Funktionale Programmierung“) noch prägnanter formuliert werden können – die Möglichkeiten sind fast unerschöpflich.

Es ist noch zu betonen, dass viele der dargebotenen Algorithmen nicht für den Alltagsgebrauch gedacht sind, da die Standardbibliotheken von Java viel flexiblere (und oftmals auch effizientere) Versionen oder Alternativen anbieten. Vielmehr sind sie didaktisch nützlich, um zu lernen, wie solche Programme funktionieren. Dann kann der Java-Programmierer auch Lösungen für Probleme entwickeln, für die es keine Standardklassen gibt.

In die 6. Auflage wurden auch die dankend entgegengenommenen Korrekturhinweise eingearbeitet sowie der Online-Service aktualisiert. Hier können die im Buch vorgestellten Programme sowie die Lösungen einiger Übungen heruntergeladen werden:

`http://public.beuth-hochschule.de/oo-plug/A&D`

Auf der Internetseite finden sich auch einige Kapitel aus älteren Auflagen sowie eine Email-Adresse für Hinweise auf Fehler, Verbesserungsvorschläge und andere Bemerkungen.

Einleitung

Das Fach „Algorithmen und Datenstrukturen“ deckt „klassische Themen“ der Ausbildung von Informatikern ab. Es gibt viele Lehrbücher, die klassische Algorithmen (wie Sortierverfahren usw.) und klassische Datenstrukturen (wie Reihungen,[1] verkettete Listen, Bäume usw.) mehr oder weniger verständlich vorstellen. Die meisten – insbesondere die besten – von ihnen wurden vor einiger Zeit geschrieben, deswegen verwenden sie typischerweise auch eine „klassische“ Programmiersprache (wie Algol, Pascal, C o. ä.) und keine neuere Sprache wie Java.

Vermutlich verbreitet sich Java heute schneller als alle anderen Programmiersprachen. Dies hat im Wesentlichen zwei Gründe:

- die Plattformunabhängigkeit, die ihre Verwendung im Internet ermöglicht
- die Objektorientierung, die moderne Programmentwicklungstechniken und -paradigmen unterstützt.

Java wird auch zunehmend als erste Unterrichtssprache verwendet, z. B. in den Informatikstudiengängen an der *Beuth Hochschule Berlin*. So gibt es immer mehr Studenten, die noch keine andere Programmiersprache beherrschen. Um sie in *Algorithmen und Datenstrukturen* fortlaufend unterrichten zu können, wurde dieses Lehrbuch entwickelt.

Es wendet sich an folgende Zielgruppen:

- Studenten von Informatikstudiengängen
- Schüler mit Leistungskurs Informatik
- Auszubildende in IT-Berufen mit Schwerpunkt Software
- Programmierer und
- Interessierte an anspruchsvollen Algorithmen.

Es ist geeignet sowohl als Lehrmaterial für Vorlesungen und Kurse wie auch zum Selbststudium.

[1] Felder, arrays.

Der Leser sollte möglichst die folgenden Voraussetzungen erfüllen:

- Erfahrung im Erstellen einfacherer Programme
- Kenntnisse der Programmiersprache Java
- insbesondere die Behandlung von Reihungen[1] und Datenstrukturen, die durch Referenzen (Zeiger) miteinander verkettet sind

Nicht erforderlich sind (vertiefte) Kenntnisse der Standardbibliothek und der fortgeschrittenen Mechanismen wie Polymorphie, Ausnahmebehandlung, abstrakte Klassen u. ä. Zum Erlernen der Sprache Java werden zum Beispiel folgende Lehrbücher empfohlen:

Solymosi, *Schmiedecke:* Programmieren mit Java
Vieweg Verlag 2001, ISBN 3-528-25697-4 (3. Auflage)
Grude: Java ist eine Sprache
Vieweg Verlag 2005, ISBN 3-528-05914-1

Die meisten guten Bücher[2] zum Thema *Algorithmen und Datenstrukturen* haben einen hohen akademischen Anspruch. Es gibt nur einige, die als Lehrbücher (z. B. als Begleitliteratur für Hochschulvorlesungen) geeignet sind. Die Mehrzahl von diesen ist jedoch für Universitätsstudiengänge entstanden. Für Fachhochschulen, wo dem theoretischen Ansatz die Praxisrelevanz vorangestellt wird, sind sie oft zu anspruchsvoll. Die Informatikstudenten an Fachhochschulen sind weniger an mathematischen Beweisen als an verständlich formulierten Algorithmen interessiert.

Insbesondere wurde auf die *Lesbarkeit* der Programme in den meisten – hauptsächlich älteren – Lehrbüchern kein Schwerpunkt gelegt. In der Zwischenzeit[3] wissen wir aber: Im Allgemeinen ist das Lesen von Programmen oftmals schwerer als das Schreiben. Bei der Entwicklung der Beispielprogramme dieses Lehrbuchs wurde auf die Lesbarkeit Acht gegeben:

- Wahl der Bezeichner
- angemessene Kommentare
- inhärente[4] Strukturierung
- konsequentes Druckbild (Einrückung, Klammerung, Schriftarten).

Hierdurch soll der Leser schneller den Sinn, den Ablauf und das Prinzip der Algorithmen erfassen können. Auch Studenten sollen sich daran gewöhnen, Programme mit hohem Lesbarkeitsgrad zu schreiben.

Beim Verfassen dieses Lehrbuchs wurden des Weiteren folgende Ziele verfolgt:

1. Einige wichtige und bekannte Algorithmen (z. B. die Algorithmen *Quicksort* und *Heapsort* zum Sortieren von Reihungen, der *Knuth-Morris-Pratt-Algorithmus* zum Suchen einer Zeichenkette in einem Text, *Baumdurchläufe* usw.) werden vorgestellt.

[2] s. Literaturverzeichnis und Empfehlungen.

[3] durch die Entwicklung des *Software Engineering.*

[4] sich aus dem Wesen der Aufgabe ergebende.

2. Der Leser soll dabei Erfahrungen sammeln, wie man Algorithmen schreiben, lesen und wie man den dynamischen Ablauf eines Algorithmus darstellen kann.

   Einen Algorithmus kann man z. B. in natürlicher Sprache, in „Pseudocode“, als Ablaufdiagramm („Kästchen mit Pfeilen dazwischen“), als Struktogramm („geschachtelte Kästchen“), als Java-Programm, als Assembler-Programm usw. darstellen.

   Einen Ablauf des Algorithmus kann man z. B. durch eine Folge von „Momentaufnahmen“ darstellen, oder durch spezielle Diagramme und Grafiken. Man kann einen solchen Ablauf auch an einer Tafel „mit Kreide und Schwamm“ oder auf einem Papier mit „Bleistift und Radiergummi“ vorführen. Zeichentrickfilme oder Computeranimationen sind auch sehr geeignet, ihre Herstellung ist aber meistens aufwändig.
3. Die Studenten von Lehrveranstaltungen dieses Fachs sollen auch üben, genau und verständlich über Algorithmen zu sprechen und zu schreiben. Sie sollen einige wichtige Fachbegriffe (z. B. *Algorithmus*, *B-Baum*, *Halde* usw.) kennen lernen und in ihr aktives Vokabular aufnehmen. Insbesondere sollen sie sich auch mit der Komplexität von Algorithmen befassen und ein Verständnis für die Algorithmen-Klassen P und NP gewinnen.
4. Sie sollen theoretisch und praktisch etwas über das Verhältnis von (abstrakten) Algorithmen und (konkreter) Programmierung erfahren.

Vertrautheit mit abstrakten Algorithmen ist eine notwendige, aber keineswegs hinreichende Voraussetzung für „gute Programmierung im Alltag“.

Die in diesem Buch abgedruckten Beispielprogramme stehen im Internet unter der folgenden Adresse zur Verfügung:

`http://www.beuth-hochschule.de/~oo-plug/A&D`

Fragen, Verbesserungsvorschläge und Kritik können an die folgenden Adressen gesendet werden:

`solymosi@beuth-hochschule.de` oder `grude@beuth-hochschule.de`

Die Autoren A. S. und U. G.

Danksagungen

Ich danke den Studenten (auch von anderen Hochschulen), die durch aufmerksames Lesen des Buches einige Fehler und Unklarheiten entdeckt und uns Hinweise gegeben haben. Außerdem gebührt meiner Frau Dr. I. Solymosi und unseren vier Kindern Dank, die die Belastungen der Arbeit an so einem Buch ertragen und mitgetragen haben. Und nicht zuletzt danke ich Gott, meinem Schöpfer und liebendem Vater, der mich nicht nur mit Fähigkeiten für meinen Beruf ausgestattet, sondern durch sein persönliches Opfer auch erlöst hat.

Andreas Solymosi

Inhaltsverzeichnis

1 Begriffsbildung 1
1.1 Algorithmus 1
1.2 Komplexität 5
1.3 Verbrauch und Komplexität 5

2 Gleichwertige Lösungen 9
2.1 Maximale Teilsumme 9
2.1.1 Summen und Teilsummen 10
2.1.2 Aufgabenstellung 10
2.1.3 Intuitive Lösung 10
2.1.4 Zeitkomplexität der Lösung 11
2.1.5 Zeit für Raum 13
2.1.6 Teile und herrsche 15
2.1.7 Die optimale Lösung 19
2.1.8 Messergebnisse 20
2.1.9 Gleichwertigkeit von Algorithmen 22
2.2 Komplexitätsformel 23
2.3 Datenstrukturen 24
2.3.1 Reihungen 25
2.3.2 Verkettete Listen 27
2.3.3 Gleichwertigkeit von Datenstrukturen 30
2.3.4 Berechnung von Ausdrücken 33

3 Rekursion und Wiederholung 35
3.1 Rekursive Algorithmen 35
3.1.1 Fakultät 36
3.1.2 Die Fibonacci-Zahlen 37
3.1.3 Die Ackermann-Funktion 39
3.1.4 Die mathematische Induktion 40
3.1.5 Permutationen 42

3.2 Abarbeitung von Datenstrukturen ... 43
3.2.1 Iterative Abarbeitung von rekursiven Datenstrukturen ... 43
3.2.2 Rekursive Abarbeitung von rekursiven Datenstrukturen ... 45
3.2.3 Rekursive Abarbeitung von Reihungen ... 46
3.2.4 Iteratoren ... 48
3.2.5 Lambda-Ausdrücke ... 48
3.3 Rekursive Kurven ... 51
3.3.1 Schneeflockenkurve ... 52
3.3.2 Die Pfeilspitzenkurve ... 54
3.3.3 Die Hilbert-Kurve ... 57
3.3.4 Ersetzen der Rekursion durch Wiederholung ... 60
3.4 Zurückverfolgung ... 62
3.4.1 Labyrinth ... 62
3.4.2 Der Weg des Springers ... 63
3.4.3 Die acht Damen ... 66

4 Suchen ... 73
4.1 Textsuche ... 73
4.2 Suchen in Sammlungen ... 77
4.3 Suchen in einer Reihung ... 78
4.3.1 Suchen in einer unsortierten Reihung ... 79
4.3.2 Lineares Suchen in einer sortierten Reihung ... 80
4.3.3 Binäres Suchen in einer sortierten Reihung ... 81
4.4 Suchen in einer verketteten Liste ... 83
4.4.1 Lineares Suchen in einer unsortierten Liste ... 84
4.4.2 Lineares Suchen in einer sortierten Liste ... 85
4.4.3 Listen funktional ... 86
4.5 Hash-Tabellen ... 88
4.5.1 Funktionalität ... 88
4.5.2 Datenorganisation ... 89
4.5.3 Hash-Funktionen ... 92
4.5.4 Weitere Aspekte ... 97
4.6 Zeitkomplexitäten beim Suchen ... 97

5 Sortierverfahren ... 101
5.1 Die Problemstellung ... 101
5.1.1 Präzisierung des Problems und Grundbegriffe ... 103
5.1.2 Zeitbedarf und Zeitkomplexität ... 104
5.1.3 Sortieralgorithmen in Java-Standardbibliotheken ... 105
5.1.4 Entwurfsmuster Strategie ... 107
5.2 Quadratische Sortierverfahren ... 108
5.2.1 Sortieren durch Vertauschen benachbarter Elemente ... 108
5.2.2 Sortieren durch Einfügen ... 111
5.2.3 Sortieren durch Auswählen ... 112

5.3 Unterquadratische Verfahren ... 114
5.4 Rekursive Verfahren ... 116
5.4.1 Quicksort ... 116
5.4.2 Sortieren mit Mischen ... 119
5.5 Logarithmische Verfahren ... 120
5.5.1 Halde ... 120
5.5.2 Die Haldenbedingung ... 121
5.5.3 Senken ... 121
5.5.4 Zwei Phasen des Heap Sorts ... 122
5.5.5 Sortieren auf der Halde ... 123
5.6 Sortieren von Listen ... 126
5.7 Externe Sortierverfahren ... 129
5.7.1 Mischen ... 129
5.7.2 Sortierkanal ... 131
5.7.3 Mischkanal ... 133
5.7.4 Fibonacci-Mischen ... 133

6 Baumstrukturen ... 137
6.1 Binärbaum ... 137
6.1.1 Definition ... 137
6.1.2 Suchen im sortierten Binärbaum ... 140
6.1.3 Darstellung von Binärbäumen ... 141
6.2 Sortieren mit Binärbäumen ... 142
6.2.1 Binärbaum als Halde ... 143
6.2.2 Senken im Binärbaum ... 143
6.2.3 Baumsort ... 146
6.2.4 Durchwandern eines Binärbaums ... 147
6.3 Operationen für Binärbäume ... 149
6.3.1 Binärbaum aus Knoten ... 149
6.3.2 Eintragen in einen sortierten Binärbaum ... 150
6.3.3 Löschen in Binärbäumen ... 151
6.4 Ausgeglichene Bäume ... 154
6.4.1 Eintragen in ausgeglichene Bäume ... 154
6.4.2 Löschen in ausgeglichenen Bäumen ... 159
6.5 2-3-4-Bäume ... 161
6.5.1 Definition ... 161
6.5.2 Spalten ... 162
6.5.3 Einfügen ... 163
6.6 Rot-Schwarz-Bäume ... 165
6.7 B-Bäume ... 170
6.8 Bäume in den Standardklassen ... 173

7 Klassen von Algorithmen 175
7.1 Was ist ein algorithmisches Problem? 175
7.2 Theoretische Lösbarkeit von Problemen 180
7.2.1 Definitionen 180
7.2.2 Beispiele 181
7.2.3 Das Halteproblem 184
7.2.4 Das Kachelproblem 186
7.2.5 Das Paligrammproblem 187
7.2.6 Gleichwertigkeit von Grammatiken 189
7.3 Praktische Lösbarkeit von Problemen 190
7.3.1 Das zweite Kachelproblem 191
7.3.2 Das Rucksackproblem 191
7.3.3 Das Aufteilungsproblem 192
7.3.4 Das Problem des Handelsreisenden 192
7.3.5 Hamiltonsche Wege durch einen Graphen 192
7.3.6 Das Erfüllbarkeitsproblem 193
7.4 Die Klassen P und NP 194
7.5 Ist P = NP? 196
7.6 Übersicht über Problemklassen 197

Verzeichnisse 199
Literaturverzeichnis 199
Klassische Literatur zum Thema 200
Java 8 201
Programmverzeichnis 201

Stichwortverzeichnis 205

Abbildungsverzeichnis

Abb. 2.1 LIFO als Reihung mit drei Elementen ... 25
Abb. 2.2 Eintragen in eine rückwärts verkettete Liste ... 28
Abb. 2.3 Eintragen in eine vorwärts verkettete Liste ... 29
Abb. 2.4 Löschen eines Elements aus der verketteten Liste ... 30

Abb. 3.1 Berechnung von f_5 ... 38
Abb. 3.2 Türme von Hanoi ... 41
Abb. 3.3 Permutationen ... 43
Abb. 3.4 Rekursive Abarbeitung einer Liste ... 45
Abb. 3.5 Rekursive Abarbeitung einer Reihung ... 47
Abb. 3.6 Monsterkurve ... 51
Abb. 3.7 Initiator (S_0) und Generator (S_1) der Schneeflockenkurve ... 52
Abb. 3.8 Drei Annäherungen der Schneeflockenkurve ... 55
Abb. 3.9 Vollständige Schneeflockenkurve ... 55
Abb. 3.10 Initiator und Generator der Pfeilspitzenkurve ... 55
Abb. 3.11 Drei Annäherungen der Pfeilspitzenkurve ... 56
Abb. 3.12 Initiator und Generator der Hilbert-Kurve ... 57
Abb. 3.13 Drei Annäherungen der Hilbert-Kurve ... 59
Abb. 3.14 Initiator und Generator Sierpinski-Kurve ... 59
Abb. 3.15 Labyrinth ... 63
Abb. 3.16 Wege im Labyrinth ... 63
Abb. 3.17 Erlaubte Züge des Springers ... 64
Abb. 3.18 Ein Weg des Springers auf einem `5 × 5`-Feld ... 65
Abb. 3.19 Durch eine Dame bedrohte Felder ... 66
Abb. 3.20 Die acht Damen ... 68
Abb. 3.21 Stabile Liebesbeziehung ... 71

Abb. 4.1 Endlicher Automat für das Muster `babaabbb` ... 76
Abb. 4.2 Hash-Tabelle ... 90
Abb. 4.3 Kollisionsbehandlung mit verketteter Liste ... 92

Abb. 5.1 Bubble Sort ... 110
Abb. 5.2 Straight Insertion ... 112
Abb. 5.3 Straight Selection ... 113
Abb. 5.4 Shell Sort ... 115
Abb. 5.5 Quick Sort ... 118
Abb. 5.6 Heap Sort ... 125

Abb. 6.1 Zwei verschiedene Bäume ... 138
Abb. 6.2 Knotenarten im Baum ... 139
Abb. 6.3 Die Ebenen eines Baumes ... 139
Abb. 6.4 Ein voller Binärbaum ... 139
Abb. 6.5 Ein kompletter Binärbaum ... 139
Abb. 6.6 Binärbäume verschiedener Tiefen und Gewichte ... 140
Abb. 6.7 Ein Binärbaum mit sieben Knoten und zwei Pseudoknoten ... 142
Abb. 6.8 Ein Baum mit eingezeichneten Schlüsseln ... 143
Abb. 6.9 Ein kompletter Binärbaum ... 144
Abb. 6.10 Ein an der Wurzel „gestörter sortierter“ Binärbaum ... 144
Abb. 6.11 Senken ... 145
Abb. 6.12 Reparierter Binärbaum ... 145
Abb. 6.13 Sortierter Binärbaum ... 148
Abb. 6.14 „inorder“ Traversieren ... 148
Abb. 6.15 Eintragen in einen sortierten Binärbaum ... 150
Abb. 6.16 Fall 1 (kein Nachfolger: löschen) ... 151
Abb. 6.17 Fall 2 (ein Nachfolger: umhängen) ... 151
Abb. 6.18 Fall 3 (zwei Nachfolger: austauschen) ... 151
Abb. 6.19 Löschen im sortierten Binärbaum ... 153
Abb. 6.20 Fall 1 – einfaches Rotieren (Die Zeichnung stammt aus [Wir]; die symmetrischen Fälle werden nicht dargestellt.) ... 155
Abb. 6.21 Fall 2 – doppeltes Rotieren (Nur einer der gekennzeichneten Äste (auf dem Baum `B`) ist zu lang.) ... 156
Abb. 6.22 Ein sortierter 2-3-4-Baum ... 161
Abb. 6.23 Spalten eines Wurzelknotens ... 162
Abb. 6.24 Spalten des Unterknotens eines 2-er-Knotens ... 162
Abb. 6.25 Spalten des Unterknotens eines 3-er-Knotens ... 163
Abb. 6.26 Einfügen im 2-3-4-Baum ... 164
Abb. 6.27 „Rote“ und „schwarze“ Kanten ... 165
Abb. 6.28 Ein Rot-Schwarz-Baum ... 166
Abb. 6.29 Rotieren im Rot-Schwarz-Baum ... 167
Abb. 6.30 Symmetrisches Rotieren im Rot-Schwarz-Baum ... 167
Abb. 6.31 Spalten, Fall 1 ... 168
Abb. 6.32 Spalten, Fall 2 ... 168

Abb. 6.33 Spalten, Fall 2.1 168
Abb. 6.34 Spalten, Fall 2.2 169
Abb. 6.35 Spalten, Fall 2.3 170
Abb. 6.36 Knoten eines B-Baums mit Schlüsseln und Verweisen 171
Abb. 6.37 Sortierkriterium im B-Baum 172

Abb. 7.1 Kachel 186
Abb. 7.2 Auslegen von Kacheln 186
Abb. 7.3 Anschlussbedingung 186
Abb. 7.4 Erfüllbarer Kachelkatalog 187
Abb. 7.5 Unerfüllbarer Kachelkatalog 187
Abb. 7.6 Das Problem des Handelsreisenden 193
Abb. 7.7 Graph ohne und mit hamiltonschem Weg 193

Tabellenverzeichnis

Tab. 1.1 Zeitverbrauch für einen Schritt 6
Tab. 1.2 Anzahl der Schritte 7

Tab. 2.1 Veränderung des Aktienwerts 10
Tab. 2.2 Zeitverbrauch in hundertstel Sekunden 21

Tab. 3.1 Wertetabelle der Fakultät 36
Tab. 3.2 Wertetabelle der Fibonacci-Zahlen 37

Tab. 4.1 Suchen im Text 74
Tab. 4.2 Informationsverlust bei der Textsuche 75
Tab. 4.3 „next-Tabelle“ für das Muster `babaabbb` 75
Tab. 4.4 Laufzeiten für Primzahlen 88
Tab. 4.5 Konstante Hash-Funktion 94
Tab. 4.6 Zweiwertige Hash-Funktion 94
Tab. 4.7 Alphabetische Hash-Funktion 95
Tab. 4.8 Hash-Funktion aus drei Zeichen 96
Tab. 4.9 Hash-Funktion nach Formel 97
Tab. 4.10 Zeitbedarf der Operationen (Anzahl der Schritte) in einer Sammlung der Größe *n* 98
Tab. 4.11 Zeitkomplexität von Operationen 99

Tab. 5.1 Halde 121
Tab. 5.2 Keine Halde 121
Tab. 5.3 Eine „an der Spitze gestörte“ Halde 121
Tab. 5.4 „Reparierte“ Halde 122
Tab. 5.5 Anzahl und Länge (in 1000) der Sequenzen mit vier Dateien 134
Tab. 5.6 Anzahl und Länge (in 1000) der Sequenzen mit sechs Dateien 134
Tab. 5.7 Fibonacci-Mischen mit sechs Dateien 135
Tab. 5.8 Fibonacci-Mischen mit drei Dateien 135
Tab. 5.9 Fibonacci-Zahlen der 4. Ordnung 135

Tab. 7.1 Lösbares Paligrammproblem 188
Tab. 7.2 Lösung des Paligrammproblems aus der Tab. 7.1 188
Tab. 7.3 Unlösbares Paligrammproblem..... 188