Vorwort

Dieses Buch soll einen Überblick über die wichtigsten der heute auf Computern eingesetzten Algorithmen geben und dem zunehmenden Kreis der Personen, die entsprechende Kenntnisse auf diesem Gebiet benötigen, die grundlegenden Verfahren vermitteln. Es eignet sich als Lehrbuch für das zweite bis vierte Studienjahr der Informatik, nachdem sich die Studenten bereits grundlegende Programmierkenntnisse angeeignet haben und mit Computersystemen vertraut sind, aber noch bevor sie eine Spezialrichtung zu weiterführenden Gebieten der Informatik oder Computeranwendungen eingeschlagen haben. Das Buch eignet sich auch für das Selbststudium oder als Nachschlagewerk für diejenigen, die sich mit der Entwicklung von Computersystemen oder Anwendungsprogrammen beschäftigen, da es Implementierungen nützlicher Algorithmen und detaillierte Informationen zu den Leistungsmerkmalen dieser Algorithmen enthält. Durch die breit angelegte Darstellungsweise stellt das Buch außerdem eine geeignete Einführung in die genannten Gebiete dar.

Den Text für diese neue Ausgabe habe ich komplett überarbeitet und mehr als tausend neue Übungen, mehr als hundert neue Abbildungen und Dutzende neuer Programme hinzugefügt. Außerdem sind die Abbildungen und Programme jetzt mit umfangreichen Kommentaren versehen. Das neue Material behandelt einerseits neue Themen und geht anderseits auf viele der klassischen Algorithmen detaillierter ein. Neu ist auch, dass sich das gesamte Buch verstärkt auf abstrakte Datentypen stützt. Damit erweitert sich der Einsatzbereich der Programme und sie entsprechen mehr den modernen Programmierumgebungen. Leser älterer Ausgaben des Buchs finden eine Fülle neuer Informationen sowie reichhaltiges Lehrmaterial, das effizienten Zugriff auf wesentliche Konzepte bietet.

Aufgrund der zahlreichen Erweiterungen haben wir die neue Ausgabe auf zwei Bände aufgeteilt (die jeder für sich den Umfang der alten Ausgabe haben). Der vorliegende Band bildet den ersten Teil. Er behandelt grundlegende Konzepte, Datenstrukturen, Sortieralgorithmen und Suchalgorithmen; der zweite Band umfasst erweiterte Algorithmen und Anwendungen, die auf den hier entwickelten grundlegenden Abstraktionen und Verfahren aufbauen. In dieser Ausgabe ist nahezu das gesamte Material zu den Grundlagen und Datenstrukturen neu.

Dieses Buch ist nicht nur für Programmierer und Informatikstudenten gedacht. Fast jeder Computerbenutzer ist daran interessiert, dass Programme schneller laufen oder sich größere Probleme lösen lassen. Die Algorithmen in diesem Buch repräsentieren das Wissen, das sich im Verlauf der letzten 50 Jahre angesammelt hat und beim effizienten Einsatz des Computers für eine breite Vielfalt von Anwendungen unverzichtbar geworden ist. Die hier beschriebenen Verfahren sind zum wesentlichen Bestandteil der wissenschaftlichen Forschung geworden, ob es sich nun um Simulationen von N-Körper-Problemen in der Physik oder die Untersuchung von genetischen

Strukturen in der Molekularbiologie handelt. Auch im Bereich von Datenbanksystemen bis hin zu Suchmaschinen im Internet bilden diese Verfahren einen wichtigen Teil moderner Softwaresysteme. Mit der zunehmenden Verbreitung des Computereinsatzes nimmt auch der Einfluss vieler der hier behandelten grundlegenden Verfahren zu. Dieses Buch soll eine Quelle sein für Studenten und Profis, die an den grundlegenden Algorithmen als Werkzeuge für die Entwicklung von Anwendungen interessiert sind und diese Algorithmen intelligent nutzen möchten.

Umfang

Das Buch enthält 16 Kapitel, die in vier Hauptteile gruppiert sind: *Grundlagen*, *Datenstrukturen*, *Sortieren* und *Suchen*. Die hier angegebenen Beschreibugen sollen den Lesern die wesentlichen Eigenschaften einer möglichst breiten Palette von fundamentalen Algorithmen vermitteln. Die vorgestellten Algorithmen werden seit Jahren in der Praxis angewendet und sind Grundpfeiler des Wissens sowohl für den angehenden Programmierer als auch für den Informatikstudenten. Es werden geniale Verfahren beschrieben, die von Binomialwarteschlangen bis zu Patricia-Tries reichen und sich alle auf grundlegende Prinzipien der Informatik beziehen. Der zweite Band besteht aus vier zusätzlichen Teilen, die sich mit Strings, Geometrie, Graphen und erweiterten Themen beschäftigen. Mit diesen Büchern verfolge ich das Ziel, die grundsätzlichen Verfahren aus diesen unterschiedlichen Gebieten zusammenzubringen, um die besten bekannten Verfahren für Problemlösungen durch Computer zugänglich zu machen.

Das Material in diesem Buch werden Sie vor allem zu schätzen wissen, wenn Sie bereits einen oder zwei Informatikkurse absolviert haben oder auf gleichwertige Programmiererfahrungen zurückblicken können: einen Kurs zur Programmierung in einer höheren Programmiersprache wie zum Beispiel C++, Java oder C und vielleicht einen weiteren Kurs, der sich mit grundlegenden Konzepten von Programmiersystemen beschäftigt. Dieses Buch ist folglich für jeden gedacht, der sich in einer modernen Programmiersprache auskennt und über die grundlegenden Merkmale moderner Computersysteme Bescheid weiß. Im Text finden Sie viele Literaturhinweise, die Ihnen helfen sollen, Lücken in Ihrem Hintergrundwissen zu schließen.

Der größte Teil des Stoffes, der sich mit analytischen Ergebnissen auseinander setzt, ist in sich abgeschlossen (oder als »über das Anliegen dieses Buchs hinausgehend« gekennzeichnet), sodass nur wenig mathematische Vorkenntnisse für den Hauptteil des Buchs notwendig sind, obwohl ein gewisses mathematisches Grundwissen zweifellos hilfreich ist.

Einsatz als Unterrichtsmittel

Für den Unterricht bietet der Stoff des Buchs einen weiten Spielraum, je nach dem Anliegen des Dozenten und der Vorbereitung der Studenten. Das Buch enthält genügend Grundlagenmaterial, um Wissen zu Datenstrukturen an Neueinsteiger zu vermitteln, und auch die erweiterten Themen sind umfassend dargestellt, sodass sie sich für den Unterricht zur Algorithmenanalyse bei fortgeschritteneren Studenten eignen. Manche Dozenten betonen eher den Aspekt der Implementierungen und der praktischen Belange, andere legen mehr Wert auf Analysen und theoretische Konzepte.

Parallel zum Buch habe ich verschiedene Studienmaterialen entwickelt, wozu unter anderem Folien für Vorträge, Programmieranweisungen, Hausaufgaben, Beispielexamen und interaktive Übungen für Studenten gehören. Auf diese Materialien können Sie über die Homepage zur Originalausgabe des Buchs unter `http://www.awl.com/cseng/titles/0-201-35088-2` zugreifen. Besuchen Sie auch die Website zu dieser deutschsprachigen Ausgabe unter `www.pearson-studium.de`.

Ein Einführungskurs zu Datenstrukturen und Algorithmen könnte die grundlegenden Datenstrukturen in Teil 2 sowie ihren Einsatz in den Implementierungen von Teil 3 und 4 herausarbeiten. Ein Kurs zum Entwurf und zur Algorithmenanalyse kann sich mit dem Grundlagenstoff von Teil 1 und Kapitel 5 beschäftigen und dann die Vorgehensweise darstellen, nach der die Algorithmen in den Teilen 3 und 4 eine gute asymptotische Leistung erzielen. Ein Kurs zur Softwaretechnik kann auf das Material zu mathematischen und erweiterten algorithmischen Themen verzichten und den Schwerpunkt darauf legen, wie sich die im Buch angegebenen Implementierungen in größere Programme oder Systeme einbinden lassen. In einem Kurs zu Algorithmen kann man einen Überblick über die Ansätze geben und die Konzepte aus allen diesen Bereichen einführen.

Frühere Ausgaben dieses Buchs sind seit Jahren an Hochschulen und Universitäten auf der ganzen Welt als Lehrmaterial für den zweiten und dritten Kurs in der Informatik und als ergänzende Lektüre für andere Vorlesungen im Einsatz. An der Princeton-Universität haben wir die Erfahrung gemacht, dass sich die in die Breite gehende Behandlung des Stoffes aus diesem Buch in den Hauptfächern der Informatik als Einführung eignet, auf der spätere Vorlesungen zur Algorithmenanalyse, Systemprogrammierung und theoretischer Informatik aufbauen können, wobei auch Studenten aus anderen Fachbereichen eine große Menge an Verfahren vermittelt bekommen, die sie unmittelbar in der Praxis nutzen können.

Die Übungen – die in dieser Ausgabe zum größten Teil neu sind – lassen sich mehreren Kategorien zuordnen. Einige sind dafür vorgesehen, das Verständnis des im Text behandelten Stoffes zu testen; sie fordern den Leser einfach auf, ein Beispiel durchzuarbeiten oder beschriebene Konzepte anzuwenden. Bei anderen Übungen geht es darum, die Algorithmen zu implementieren und zusammenzufügen oder empirische Untersuchungen durchzuführen, um die Varianten der Algorithmen zu vergleichen und ihre Eigenschaften kennen zu lernen. Wieder andere Übungen liefern wichtige Informationen auf einer Detailebene, die über die Behandlung im Text hinausgeht. Jeder Leser kann davon profitieren, wenn er sich mit den Übungen auseinander setzt.

Algorithmen mit Praxisbezug

Jeder, der seinen Computer effizienter einsetzen möchte, kann dieses Buch als Nachschlagewerk oder für das Selbststudium nutzen. Leser mit Programmiererfahrung finden das ganze Buch hindurch Informationen zu speziellen Themen. Zum größten Teil können Sie die einzelnen Kapitel dieses Buchs unabhängig von anderen lesen, auch wenn in manchen Fällen bestimmte Algorithmen in einem Kapitel auf Verfahren aus einem vorherigen Kapitel zurückgreifen.

In diesem Buch untersuchen wir vor allem Algorithmen, die mit hoher Wahrscheinlichkeit von praktischem Interesse sind. Das Buch bietet Informationen zu den einschlägigen Werkzeugen, sodass der Leser die Algorithmen getrost implementie-

ren, debuggen und einsetzen kann, um ein Problem zu lösen oder die jeweilige Funktionalität in einer Anwendung bereitzustellen. Außerdem werden auf einer einheitlichen Menge von Beispielen vollständige Implementierungen der vorgestellten Verfahren angegeben und die Operationen dieser Programme erläutert.

Da wir mit echtem Code arbeiten und keinen Pseudocode schreiben, können Sie die Programme unmittelbar in der Praxis ausprobieren. Auf der Homepage der Originalausgabe finden Sie die Programmlistings zum Buch. Diese funktionsfähigen Programme sollen Ihnen unter anderem dabei helfen, die Algorithmen zu verstehen. Sehen Sie sich die Programme an, um sich mit den Details eines Algorithmus vertraut zu machen oder Möglichkeiten kennen zu lernen, wie man Datenstrukturen initialisiert, Grenzbedingungen testet und andere kritische Situationen behandelt, die oftmals nicht so leicht zu programmieren sind. Führen Sie die Programme aus, um sich ein Bild von der Arbeitsweise der Algorithmen zu machen und Leistungsmerkmale empirisch zu untersuchen. Vergleichen Sie Ihre Ergebnisse mit den Tabellen im Buch oder probieren Sie eigene Modifikationen aus.

An verschiedenen Stellen veranschaulichen empirische und analytische Ergebnisse, warum bestimmte Algorithmen zu bevorzugen sind. Wenn es von allgemeinem Interesse ist, werden auch Beziehungen der hier behandelten praktischen Algorithmen zu rein theoretischen Ergebnissen beschrieben. Auch wenn wir nicht besonders darauf hinweisen, stellen wir Querverbindungen zwischen der Algorithmenanalyse und der theoretischen Informatik her. Spezielle Informationen zu Leistungsmerkmalen von Algorithmen und Implementierungen werden das ganze Buch hindurch zusammengetragen, konzentriert dargestellt und erörtert.

Programmiersprache

Die Programmiersprache C++ dient als Grundlage für alle Implementierungen. Die Programme arbeiten mit vielen Standardkonstruktionen von C++, und im Text finden Sie knappe Beschreibungen zu den einzelnen Programmelementen.

Chris Van Wyk hat mit mir zusammen einen Stil der C++-Programmierung entwickelt, der auf Klassen, Schablonen und überladenen Operatoren beruht und unserer Ansicht nach einen effizienten Weg darstellt, um die Algorithmen und Datenstrukturen als echte Programme zu präsentieren. Wir haben uns um elegante, kompakte, effiziente und portable Implementierungen bemüht. Nach Möglichkeit sind wir diesem Stil gefolgt, sodass ähnliche Programme auch ähnlich aussehen.

Für viele Algorithmen in diesem Buch gelten die Ähnlichkeiten unabhängig von der Programmiersprache: Quicksort ist eben Quicksort (um einen prominenten Vertreter herauszugreifen), ob man ihn nun in Ada, Algol-60, Basic, C, C++, Fortran, Java, Mesa, Modula-3, Pascal, PostScript, Smalltalk oder in einer der anderen zahllosen Programmiersprachen und Umgebungen formuliert, in denen er sich als effizientes Sortierverfahren etabliert hat. Auf der einen Seite ist unser Code durch die Erfahrungen mit der Implementierung von Algorithmen in diesen und vielen anderen Sprachen geprägt (eine C-Version dieses Buchs ist bereits erschienen, eine Java-Version ist geplant), auf der anderen Seite gehen bestimmte Eigenschaften bei einigen dieser Sprachen auf Erfahrungen zurück, die die Entwickler mit einigen der im Buch betrachteten Algorithmen und Datenstrukturen gesammelt haben.

Kapitel 1 umfasst ein detailliertes Beispiel für diesen Ansatz, um effiziente C++-Implementierungen unserer Algorithmen zu entwickeln, und Kapitel 2 beschreibt

unsere Herangehensweise, um sie zu analysieren. Kapitel 3 und 4 beschreiben und rechtfertigen die grundlegenden Mechanismen, die wir für Implementierungen von Datentypen und abstrakten Datentypen verwenden. Diese vier Kapitel bilden das Gerüst für den übrigen Teil des Buchs.

Danksagung

Zu früheren Ausgaben dieses Buchs habe ich viele Rückmeldungen erhalten. Hunderte Studenten der Princeton-Universität und der Brown-Universität haben im Verlauf mehrerer Jahre unter Entwurfsversionen dieses Buchs gelitten. Mein besonderer Dank richtet sich an Trina Avery und Tom Freeman für ihre Hilfe bei der Herstellung der ersten Ausgabe, an Janet Incerpi für ihre Kreativität und ihren Einfallsreichtum, um unsere frühe und recht einfache Technik für den computergestützten Satz sowohl auf Hardware- als auch auf Softwareseite zum Laufen zu bringen, um die erste Ausgabe zu produzieren, an Marc Brown für seinen Anteil an der Forschung zur visuellen Darstellung von Algorithmen, der den Ursprung für viele Abbildungen im Buch bildet, und an Dave Hanson und Andrew Appel für ihre Bereitschaft, alle meine Fragen zu Programmiersprachen zu beantworten. Danken möchte ich auch den vielen Lesern, die mir Hinweise zu verschiedenen Ausgaben geliefert haben; hier seien Guy Almes, Jon Bentley, Marc Brown, Jay Gischer, Allan Heydon, Kennedy Lemke, Udi Manber, Dana Richards, John Reif, M. Rosenfeld, Stephen Seidman, Michael Quinn und William Ward genannt.

Um diese neue Ausgabe zu produzieren, hatte ich das Vergnügen, mit Peter Gordon, Debbie Lafferty und Helen Goldstein von Addison-Wesley zusammenzuarbeiten, die geduldig den Fortgang dieses Projekts betreut haben. Angenehm ist auch die Hilfe durch andere Mitarbeiter aus dem professionellen Team bei Addison-Wesley gewesen. Die Natur dieses Projekts machte das Buch zu einer gewissen Herausforderung für viele von ihnen und ich weiß ihr Entgegenkommen sehr zu schätzen.

Für das Manuskript dieses Buchs konnte ich drei neue Mentoren gewinnen und möchte ihnen besonderen Dank aussprechen. Als Erstes hat Steve Summit frühere Versionen des Manuskripts auf technischer Ebene sorgfältig überprüft und mir im wahrsten Sinne des Wortes Tausende Detailhinweise vor allem zu den Programmen geliefert. Steve hat meine Absicht klar erkannt, elegante, effiziente und wirksame Algorithmen bereitzustellen, und seine Kommentare haben mir nicht nur geholfen, eine gewisse Einheitlichkeit über die Implementierungen hinweg zu gewährleisten, sondern mich auch dabei unterstützt, viele von ihnen wesentlich zu verbessern. Zweitens hat mir auch Lyn Dupre Tausende von detaillierten Hinweisen zum Manuskript geliefert, die für mich unschätzbar waren, nicht nur, um grammatikalische Fehler zu korrigieren und zu vermeiden, sondern – vor allem – um einen einheitlichen und flüssigen Schreibstil zu finden, mit dem sich die überwältigende Masse des hier gebotenen technischen Materials verbinden lässt. Drittens hat Chris Van Wyk alle meine Algorithmen in C++ implementiert und debugged, zahlreiche Fragen über C++ beantwortet, bei der Entwicklung eines geeigneten C++-Programmierstils geholfen und das Manuskript gewissenhaft gegengelesen. Chris hat mir auch geduldig beigestanden, als ich viele seiner C++-Programme auseinander genommen und dann, als ich mehr über C++ von ihm gelernt hatte, wieder fast so zusammengesetzt habe, wie er sie geschrieben hatte. Ich bin sehr dankbar dafür, dass ich von Steve, Lyn und Chris lernen konnte – ihre Beiträge waren für die Entwicklung dieses Buchs wesentlich.

Vieles von dem, was ich hier niedergeschrieben habe, geht auf Vorlesungen und Veröffentlichungen meines Beraters Don Knuth an der Stanford-Universität zurück. Obwohl Don keinen direkten Einfluss auf diese Arbeit hatte, ist seine Präsenz im Buch zu spüren, denn er war es, der die Untersuchung von Algorithmen auf eine wissenschaftliche Grundlage stellte, die ein Werk wie dieses erst ermöglicht hat. Mein Freund und Kollege Philippe Flajolet, der die treibende Kraft bei der Entwicklung der Algorithmenanalysen zu einem ausgewachsenen Forschungsbereich darstellt, hatte einen ähnlichen Einfluss auf dieses Werk.

Sehr dankbar bin ich für die Unterstützung durch die Princeton-Universität, die Brown-Universität und das Institut National de Recherche en Informatique et Automatique (INRIA), wo ich den größten Teil der Arbeit an diesem Buch vollbracht habe, sowie durch das Institute for Defense Analyses und das Xerox Palo Alto Reserach Center, wo ich während eines Gastaufenthalts einen Teil der Arbeiten am Buch ausgeführt habe. Viele Teile des Buchs beruhen auf Forschungsarbeiten, die von der National Science Foundation und dem Office of Naval Research großzügig unterstützt wurden. Schließlich möchte ich Bill Bowen, Aaron Lemonick und Neil Rudenstine für Ihre Hilfe danken, ein akademisches Klima an der Princeton-Universität zu schaffen, in dem es mir trotz zahlreicher anderer Verpflichtungen möglich war, dieses Buch vorzubereiten.

Robert Sedgewick
Marly-le-Roi, Frankreich, 1983
Princeton, New Jersey, 1990, 1992
Jamestown, Rhode Island, 1997
Princeton, New Jersey, 1998

Vorwort des C++-Beraters

Es waren Algorithmen, die mich zur Informatik gebracht haben. Die Untersuchung von Algorithmen verlangt Denken auf verschiedene Art und Weise: *kreativ*, um eine Idee zu finden, die ein Problem löst, *logisch*, um ihre Korrektheit zu analysieren, *mathematisch*, um ihre Leistung zu analysieren, und *sorgfältig*, um die Ideen als detaillierte Folge von Schritten auszudrücken, sodass sie zu Software werden kann. Da meine größte Zufriedenheit bei der Untersuchung von Algorithmen auf die Erkenntnis zurückgeht, dass es sich um funktionsfähige Computerprogramme handelt, habe ich die Chance wahrgenommen, mit Bob Sedgewick an einem Algorithmenbuch zu arbeiten, das auf C++-Programmen basiert.

Bob und ich haben die Beispielprogramme mit den relevanten Merkmalen von C++ in unkomplizierter Weise geschrieben. Mit Klassen trennen wir die Spezifikation eines abstrakten Datentyps von den Details seiner Implementierung. Wir verwenden Schablonen und überladene Operatoren, sodass man unsere Programme für viele unterschiedliche Datentypen ohne Änderungen übernehmen kann.

Allerdings haben wir darauf verzichtet, viele andere C++-Verfahren vorzustellen. Beispielsweise lassen wir normalerweise zumindest einige der Konstruktoren weg, die notwendig sind, um eine Klasse zu einer »First Class« zu machen (wobei Sie aber in Kapitel 4 lernen, wie sich das problemlos nachholen lässt); die meisten Konstruktoren verwenden Zuweisungen anstelle der Initialisierung, um Werte in den Instanzvariablen zu speichern; wir verwenden Zeichenstrings im Stil von C und stützen uns nicht auf die Strings in der C++-Bibliothek; wir verwenden Wrapper-Klassen, die

aber nicht um jeden Preis auf »leichtgewichtig« getrimmt sind; und wir arbeiten mit einfachen anstelle von »intelligenten« Zeigern.

Diese Entscheidungen haben wir deshalb getroffen, damit der Blick auf die eigentlichen Algorithmen nicht durch technische, C++-spezifische Details versperrt wird. Nachdem Sie verstanden haben, wie die Programme in diesem Buch arbeiten, sind Sie gut gerüstet und können sich mit den erwähnten C++-Verfahren auseinander setzen, um die jeweiligen Effizienzkompromisse abzuschätzen und den Einfallsreichtum der Entwickler der Standard Template Library zu bewundern.

Ob es sich um Ihren ersten Kontakt mit der Untersuchung von Algorithmen handelt oder ob Sie eine alte Bekanntschaft wieder aufleben lassen – erfreuen Sie sich zumindest genauso daran wie ich, als ich mit Bob Sedgewick an den Programmen gearbeitet habe.

Danken möchte ich Jon Bentley, Brian Kernighan und Tom Szymanski, von denen ich jede Menge zur Programmierung gelernt habe, Debbie Lafferty für die Vorschläge, an diesem Projekt mitzuarbeiten, und den Bell Labs, der Drew-Universität und der Princeton-Universität für ihre institutionelle Unterstützung.

Christopher Van Wyk
Chatham, New Jersey, 1998

Hinweis: In diesem Buch wird aus Gründen der besseren Lesbarkeit des Textes grundsätzlich nur die männliche Form stellvertretend für weibliche und männliche Formen verwendet. Wir bitten alle Leserinnen um Verständnis für diese Vorgehensweise.

Für Adam, Andrew, Brett, Robbie
und besonders Linda

Hinweise zu den Übungen

Die Klassifizierung der Übungen ist ein zum Teil gefährliches Unterfangen, weil die Leser einen unterschiedlichen Kenntnisstand und Erfahrungsschatz mitbringen. Dennoch ist eine gewisse Führung angebracht, sodass viele Übungen mit einer von vier Markierungen gekennzeichnet sind, um Sie bei der Auswahl zu unterstützen.

Übungen, die *Ihr Verständnis des Stoffes testen*, sind mit einem Dreieck wie folgt gekennzeichnet:

▹ 9.54 Zeichnen Sie eine Binomialwarteschlange der Größe 29 mithilfe der Darstellung von Binomialbäumen.

Meistens beziehen sich derartige Übungen direkt auf Beispiele im Text und sollten keine besondere Schwierigkeit darstellen; ihre Lösung kann aber Tatsachen oder Konzepte erläutern, denen Sie beim ersten Lesen vielleicht ausgewichen sind.

Übungen, die *neue Informationen und Denkanstöße bieten*, sind wie folgt mit einem Kreis gekennzeichnet:

∘ 9.61. Implementieren Sie Binomialwarteschlangen, sodass Eigenschaft 9.7 gilt, indem Sie den Datentyp für Binomialwarteschlangen um die Größe der Warteschlange ergänzen und dann mithilfe der Größe die Schleifen steuern.

Derartige Übungen sollen Sie dazu anregen, über ein wichtiges Konzept nachzudenken, das sich auf den Stoff im Text bezieht, oder eine Frage zu beantworten, die vielleicht beim Lesen aufgetaucht ist. Diese Übungen sollten Sie sich auch dann ansehen, wenn Ihnen die Zeit fehlt, sie durchzuarbeiten.

Übungen, die möglicherweise eine *Herausforderung für Sie* darstellen, sind mit einem schwarzen Punkt versehen:

• 9.63. Implementieren Sie die Operation *Einfügen* für Binomialwarteschlangen, indem Sie ausschließlich die Operation *Verknüpfen* verwenden.

Die Lösung solcher Übungen kann je nach Ihren Erfahrungen oftmals recht zeitaufwändig sein. Im Allgemeinen ist es am produktivsten, diese Übungen nicht auf einmal, sondern in mehreren Sitzungen durchzuarbeiten.

Übungen, die wir (in Bezug auf die meisten anderen) als *sehr schwierig* eingestuft haben, sind mit zwei schwarzen Punkten hervorgehoben:

•• 9.64. Implementieren Sie die Operationen *Ändern der Priorität* und *Entfernen* für Binomialwarteschlangen. *Hinweis:* Sie müssen den Knoten eine dritte Verbindung hinzufügen, die im Baum aufwärts zeigt.

Derartige Übungen ähneln Fragen, denen Sie in der Fachliteratur begegnen, wobei Sie aber der Stoff im Buch darauf vorbereitet, dass Sie diese Übungen lösen können (und das möglicherweise auch erfolgreich).

Die Markierungen sollen in Bezug auf Ihre Programmierkenntnisse und mathematischen Fähigkeiten neutral sein. Übungen, die tiefer gehende Erfahrungen in der Programmierung oder der mathematischen Analyse verlangen, sind sicherlich als

solche zu erkennen. Alle Leser seien dazu ermutigt, ihr Verständnis der Algorithmen zu testen, indem sie sich mit diesen Übungen auseinander setzen. Trotzdem ist eine Übung wie die folgende für einen angehenden Programmierer oder einen Informatikstundenten relativ leicht zu lösen, während sie für einen Programmierneuling mit großem Arbeitsaufwand verbunden sein kann:

- 1.22. Modifizieren Sie Programm 1.4, um zufällige Ganzzahlpaare zwischen 0 und $N - 1$ zu generieren, statt die Paare von der Standardeingabe zu lesen. Durchlaufen Sie die Schleife, bis $N - 1$ *Vereinigungs*-Operationen ausgeführt worden sind. Führen Sie Ihr Programm für $N = 10^3$, 10^4, 10^5 und 10^6 aus und geben Sie die Gesamtzahl der für jeden Wert von N erzeugten Kanten aus.

In ähnlichem Sinne seien auch alle Leser dazu aufgefordert, ihre analytischen Kenntnisse und ihr Wissen über die Eigenschaften der Algorithmen einzuschätzen. Dennoch kann eine Übung wie die folgende für einen Wissenschaftler oder einen Mathematikstudenten ein Leichtes sein, während sie für Leser ohne Erfahrungen in der mathematischen Analyse eine Herausforderung darstellen kann:

1.12. Berechnen Sie den *mittleren* Abstand von einem Knoten zur Wurzel für den ungünstigsten Fall in einem Baum mit 2^n Knoten, den der gewichtete Quickunion-Algorithmus erstellt hat.

Durch die Vielzahl der angegebenen Übungen wird es Ihnen sicherlich nicht möglich sein, alle zu lesen und durchzuarbeiten; dennoch hoffe ich, dass Sie genügend Übungen finden, mit denen Sie Ihr Wissen zu den Sie interessierenden Themen vertiefen können, als sich lediglich nur mit dem Text zu beschäftigen.