Vorwort

Was ist Theoretische Informatik? Insbesondere, was sollten die Inhalte einer Einführung in die Theoretische Informatik sein? Diese Fragen stellten sich mir im Sommer 1995, als ich die Vorlesungen Informatik III und Informatik IV im Grundstudium des Diplomstudienganges Informatik an der Universität Bonn, die ich im WS 1995/96 bzw. im SS 1996 gehalten habe, konzipierte. Die Antwort, die sich ergab, war zunächst methodischer Natur. Im theoretischen Teil des Grundstudiums sollten die Studierenden lernen, Probleme fundiert und präzise zu bearbeiten. Wenn dies anhand von Inhalten geschieht, die Relevanz für die Anwendungen haben, um so besser. Üblicherweise werden in der einführenden Theorieveranstaltung die Gebiete Automatentheorie und Formale Sprachen sowie Berechenbarkeits- und Komplexitätstheorie gelesen. Dies ist dann Inhalt von einer der oben bezeichneten Vorlesungen. Inhalt der anderen Vorlesung ist meistens eine Einführung in die Algorithmik. Ich habe mich aus didaktischen Gründen dafür entschieden, diese übliche strikte Trennung der Inhalte aufzuheben und beide Bereiche zusammen zweisemestrig zu konzipieren. Dies wurde dadurch bestärkt, daß das Gebiet Algorithmik als Teilgebiet der Theoretischen Informatik angesehen wird. Die konkrete Stoffauswahl wurde von der Idee beeinflußt, daß die gewählten Inhalte möglichst auch für die Praxis von Relevanz sein sollten. So sind z.B. schnelle Fouriertransformationen für die Praxis viel wichtiger als das Postsche Korrespondenzproblem.

Teil I des Buches behandelt zunächst die Berechenbarkeit von Funktionen. Dabei wird zwischen theoretischer und praktischer Berechenbarkeit unterschieden. Theoretisch berechenbar ist eine Funktion f, wenn es ein effektives Rechenverfahren gibt, das für eine beliebige Eingabe x aus dem Definitionsbereich den Funktionswert $f(x)$ berechnet. In Kapitel 1 wird dieser intuitive Berechenbarkeitsbegriff formalisiert. Dies führt zur Einführung der μ-rekursiven Funktionen, der turingberechenbaren Funktionen und dem Beweis der Äquivalenz dieser beiden Formalisierungen. Beim Beweis der Äquivalenz kommt die grundlegende Beweistechnik der Simulation zur Anwendung. Mittels Diagonalisierung, eine weitere grundlegende Beweistechnik, wird gezeigt, daß es nichtberechenbare Funktionen gibt. Kapitel 1 ist das einzige Kapitel, das keinen unmittelbaren Anwendungsbezug hat. Denoch meine ich, daß ein Informatiker sich einmal mit der Frage, was im Prinzip berechenbar ist, auseinandergesetzt haben sollte und habe dies auch bewußt an den Anfang des Buches gestellt.

Eine Funktion ist auch dann theoretisch berechenbar, wenn jedes effektive Rechenverfahren mindestens eintausend Jahre benötigt, um den Funktionswert $f(x)$ zu berechnen. Eine derartige Funktion ist ganz gewiß nicht praktisch berechenbar. Diese Beobachtung führt zum Begriff der praktischen Berechenbarkeit, die in Kapitel 2 des Buches behandelt wird. Als praktisch berechenbar wird eine Funktion f angesehen, wenn es ein Computerprogramm gibt, das für ein beliebiges x aus dem Definitionsbereich den Funktionswert $f(x)$ auf einem Rechner in Polynomzeit berechnet. Für viele Funktionen, die in der Praxis auftreten, kennt man zwar kein derartiges Computerprogramm, jedoch kann man bei gegebenem x und gegebener Herleitung für $f(x)$ in polynomieller Zeit überprüfen, ob die Herleitung korrekt ist. Dies führt zur Begriffsbildung der NP-vollständigen Probleme, die in gewissem Sinn „schwierigste" derartige Probleme sind. In der Tat sind auch viele Funktionen der Praxis NP-vollständig und verlangen eine andere Behandlung als Probleme, für die bekannt ist, daß sie praktisch berechenbar sind.

In Teil II des Buches werden grundlegende Werkzeuge, die häufig bei der Entwicklung von Algorithmen benötigt werden, vorgestellt. Immer müssen Algorithmen auch Mengenverwaltungsprobleme lösen. Darum ist Kapitel 3 der Entwicklung von Datenstrukturen zur Lösung von Mengenverwaltungsproblemen gewidmet. Neben einfachen Datenstrukturen - wie Feldern und Listen - werden Bäume, insbesondere balancierte Bäume, Hashing, Priority Queues und Datenstrukturen zur Organisation von disjunkten Mengen behandelt. Viele Probleme lassen sich als ein Problem auf Graphen formalisieren, zu deren Lösung dann ein geordnetes Betrachten des Graphen notwendig ist. Daher beschäftigt sich Kapitel 4 mit Durchmusterungsmethoden für Graphen. In der Praxis erhält ein Informatiker häufig in der Sprache des Anwendungsbereiches eine Problemstellung mit der Aufgabe, ein Computerprogramm zu entwickeln, welches das Problem löst. Üblicherweise wird zunächst mittels Abstraktion aus dem Anwendungsproblem eine mathematische Formulierung erstellt. Zur Lösung des formalen Problems entwickelt man einen Algorithmus, der dann in einer geeigneten Programmiersprache auf dem Rechner implementiert wird. Für die Entwicklung von Algorithmen gibt es allgemeine algorithmische Paradigmen, deren Anwendbarkeit jeweils von gewissen Eigenschaften abhängt. Kapitel 5 ist derartigen Paradigmen gewidmet. Diese umfassen Divide-and-conquer, dynamische Programmierung, Backtracking, Branch-and-bound, Greedyalgorithmen und Approximationsalgorithmen.

In Teil III des Buches werden Lösungen für elementare Probleme entwickelt, die häufig selbst als Teilprobleme bei der Lösung von Problemen auftreten. Dabei kommen die in Teil II des Buches vorgestellten Werkzeuge zum Tragen. Einer der wichtigsten Anwendungsbereiche von Automatentheorie und formalen Sprachen ist der Übersetzerbau. Daher werden im Kapitel 6 die Automatentheorie und formalen Sprachen mit diesem konkreten Anwendungsbereich im Hintergrund behandelt. Dies führt dazu, daß sich die Inhalte aus der lexikalischen Analyse und der Syntaxanalyse eines Compilers motivieren. Bei der Entwicklung von Algorithmen, die für die Scanner- und die Parsergenerierung benötigt werden, wird großer Wert auf deren Effizienz bzw. auf die der generierten Werkzeuge gelegt. Dies schlägt sich z.B. bei der Minimierung von endlichen Automaten oder bei der Konstruktion von $LL(k)$-Parsern nieder. Algorithmen für grundlegende Probleme auf Graphen, wie z.B. Wege- oder auch Netzwerkflußprobleme, und für Probleme auf Strings, wie z.B. Stringmatching, werden in den Kapiteln 7 und 8 behandelt. In Kapitel 9 wird der Nutzen von Fouriertransformationen anhand der Polynommultiplikation aufgezeigt und dann Algorithmen für schnelle Fouriertransformationen entwickelt. Viele in der Praxis auftretende Probleme lassen sich als lineares bzw. als ganzzahliges lineares Programm formulieren. Kapitel 10 bietet daher eine Einführung in die lineare und ganzzahlige lineare Programmierung.

Das Buch ist an Personen gerichtet, die über grundlegende Kenntnisse der Analysis und der linearen Algebra, wie sie im 1. Semester in den Vorlesungen Analysis I bzw. Lineare Algebra I vermittelt werden, verfügen. Insofern wurde auf einen Anhang „Mathematische Grundlagen " verzichtet. Die im Buch benötigten graphentheoretischen Grundlagen stehen zu Beginn des 4. Kapitels. Beweise werden, mit Ausnahme der Beweise von Behauptungen und Lemmata, die selbst Teil eines Beweises sind, mit $\blacksquare$ abgeschlossen. Deren Ende wird mit $\square$ angezeigt. Beispiele, die noch fortgeführt werden, sind mit $\diamondsuit$ abgeschlossen. Ansonsten ist das Ende eines Beispiels mit $\blacklozenge$ markiert. Das Buch enthält Übungsaufgaben, die zum Teil dort im Text stehen, wo sie bearbeitet werden sollen. Weitere Übungsaufgaben stehen jeweils am Kapitelende. Schwere Übungsaufgabe sind mit * und sehr schwere Übungsaufgaben mit ** markiert. Um die Vertiefung in den bearbeiteten Teilbereichen zu vereinfachen, werden ausführliche Literaturhinweise gegeben. Bei der Vielzahl von ausgezeichneter Literatur habe ich sicherlich das eine oder andere gute Buch übersehen. Dies bitte ich zu entschuldigen.

Der Inhalt des Buches ist zu umfangreich, um in den oben genannten zwei Vorlesungen behandelt zu werden. Meine Empfehlung ist Teil I, Teil II und aus Teil III die Kapitel 6.1 - 6.3, 7, 8 und 9 zu behandeln. Gegebenenfalls würde ich die Kapitel 7 und 8 kürzen. Meine Absicht ist, das Buch sukzessive zu verbessern. Daher bin ich für Anregungen und Hinweise jeglicher Art dankbar und habe hierfür die Email-Adresse

blum.buch@informatik.uni-bonn.de

einrichten lassen.

Mein Dank gilt den Kollegen Herrn Michael Clausen und Herrn Martin Dietzfelbinger, die Teile des Buches gelesen haben, sowie meinen Mitarbeitern Frau Maria Nikolaidou und Herrn Claus Rick, die das Buch insgesamt gelesen haben. Deren wertvolle Hinweise haben zur Verbesserung des Buches in vielerlei Hinsicht beigetragen. Desweiteren danke ich Herrn Harald Schwarz für die Erstellung der Druckvorlage. Dem Oldenbourg Verlag und insbesondere der Lektorin, Frau Margarete Metzger gilt mein Dank für die gute Zusammenarbeit.

Bonn, im August 1998 Norbert Blum