CARL HANSER VERLAG

Christian Wagenknecht

Algorithmen und Komplexität

3-446-22314-2

www.hanser.de

Vorwort

Dieser Text ist aus Vorlesungen über Algorithmen und Komplexität hervorgegangen, die der Autor seit 1993 im Fachbereich Informatik an der Hochschule Zittau/Görlitz (FH) gehalten hat. Sie wurden mehrfach überarbeitet und durch Begleitmaterial im Web ergänzt. Auch auf dieses Material kann man über die Homepage dieses Buches

`http://www.inf.hs-zigr.de/~wagenkn/AuK-Buch/`

zugreifen. Diese Seite enthält Verweise auf die *erforderliche Software*, einen *interaktiven Kurs* zur Einführung in die Arbeit mit SCHEME, sämtliche *im Text vorkommenden Programme* mit zusätzlichen Kommentaren, Musterlösungen zu ausgewählten Übungsaufgaben und *didaktisch aufbereitetes Material*, das bestimmte Inhalte des Buches ergänzt.

Von dem interaktiven SCHEME-Einführungskurs sollte man *vor* dem Lesen des Buches wenigstens die ersten 3 Themen bearbeiten. Mit zunehmendem Arbeitsfortschritt am Text kann sich der Lesende nach Bedarf weiteren Kursthemen zuwenden. Dabei ist zu beachten, dass das angebotene Kursmaterial ständig weiterentwickelt wird.

Im vorliegenden Buch werden nur *klassische Themen* behandelt. Der Schwerpunkt liegt also nicht auf inhaltlicher Vollständigkeit – hierfür gibt es Standardliteratur, wie beispielsweise [9], [14], [18], [29], [30], [31], [52] –, sondern auf *didaktischer Sorgfalt*. Natürlich wurde der Text dieses Buches von den vielen im Literaturverzeichnis genannten Titeln beeinflusst. Besonders hervorzuheben sind [10], [41], [50] und [51].

Nach der Überzeugung des Autors kommt es in erster Linie darauf an, dass die vermittelten Konzepte und Phänomene von den Studierenden *verinnerlicht* werden. Aufgrund wachsender Stofffülle wird dafür oft sehr wenig Zeit eingeräumt, manches dem Selbststudium überlassen.

Dabei wird oftmals übersehen, dass Lernende für die Erarbeitung und Aneignung ihnen zunächst unbekannter Themen *Orientierungshilfen* und *Handlungsanleitungen* benötigen. Dies geschieht in diesem Buch durch kommentierte Beispiele, ausgearbeitete Analyseergebnisse, vorgefertigte Programme, weiterführende Arbeitsaufträge, aufeinander abgestimmte Übungsaufgaben u.v.m. Die behandelten Themen sollen dadurch vertieft werden, dass der Studierende diesen Angeboten folgt und dementsprechend aktiv wird.

Dieses Ziel kann nicht durch einen papiergebundenen Lehrtext allein erreicht werden. Eine *Software-gestützte Arbeitsumgebung* ermöglicht *interaktives Arbeiten* am Lehrgegenstand. *Selbstgesteuerte, orientierend-angeleitete, interaktive Tätigkeit ist der einzige Weg, um Wissen dauerhaft zu erwerben und dessen Anwendung einzuüben.*

Diesen „Hauptsatz der Didaktik“ für das Thema „Algorithmen und Komplexität“ mit modernen Mitteln umzusetzen, erforderte die Erschließung einer Softwareumgebung, die folgenden Anforderungen genügt:

Einfache Bedienung: Der Softwaregebrauch darf als didaktisches Vehikel nicht ins Zentrum der Bemühungen des Lernenden rücken. Bekannte Systeme oder solche mit gewohnten Merkmalen sind daher zu bevorzugen.

Konsistenz: Die im Einzelnen beteiligten Systeme müssen zueinander passen und ohne nennenswerten Zusatzaufwand zusammenspielen.

Sprachliche Ausdrucksmittel mit wählbaren Abstraktionsstufen: Ein und denselben Sachverhalt auf verschiedenen Abstraktionsniveaus sprachlich zu beschreiben hilft Einsichten zu entwickeln und zu vertiefen. Ein programmiersprachliches System muss daher erweiterbar und damit an spezielle Sprachbedürfnisse anpassbar sein sowie – bei möglichst einfacher Syntax – über semantisch leistungsfähige Sprachelemente verfügen, die abstrakte Konzepte in knapper Form bereitstellen.

Angemessenheit: Eine Arbeitsumgebung soll aus spezialisierten Komponenten, die einen bestimmten Zweck erfüllen, aufgebaut sein. Oftmals wird übersehen, dass die von Studierenden zu treffende Entscheidung, das zur Lösung einer bestimmten Aufgabe wirklich passende Werzeug einzusetzen, ein Lehrgegenstand ist.

Verfügbarkeit: Eine noch so gute Arbeitsumgebung ist unbrauchbar, wenn sie nicht vollständig zur Verfügung steht und stabil funktioniert.

Die Softwarekomponenten der hier verwendeten Arbeitsumgebung stehen entweder *kostenlos* zur Verfügung oder gehören zum Basisangebot der Computerpools einer jeden Hochschule und können dort (unter *Windows und/oder Linux*) im Allgemeinen kostenlos genutzt werden. Unsere Arbeitsumgebung umfasst:

1. `gnuplot` zur *Visualisierung*, insbesondere für Schaubilder von Aufwandsfunktionen. Alternativ kann ein separater Funktionsplotter verwendet oder auf das Visualisierungsangebot anderer bereits integrierter Komponenten zurückgegriffen werden. Aus Gründen der Dokumentation (kommentiertes Sitzungsprotokoll) muss darauf geachtet werden, dass die zur Speicherung verwendeten Dateiformate zueinander passen. Soweit vorhanden leisten auch grafische Taschenrechner gute Dienste.
2. `Maple` bzw. `MuPAD` als Werkzeug für symbolisches Rechnen und ggf. zur Visualisierung.
3. CHEZ SCHEME, sprich: Schejih Skiem, als Programmiersystem. Alle im Text enthaltenen Programme wurden mit Chez Scheme entwickelt und getestet. Im Allgemeinen laufen sie auch mit DrScheme, ein System mit didaktisch angelegter Umgebung, das besonders für Studierende ohne Vorkenntnisse in funktionsorientierter Programmierung empfohlen wird.
4. ein beliebiges Tabellenkalkulationssystem, z.B. `Excel`, für tabellarisch organisierte numerische Berechnungen.

Auf den ersten Blick mag der Einsatz mehrerer Anwendungsprogramme verwirren und die Erwartung zusätzlicher Schwierigkeit hervorrufen. Dies ist aber nicht der Fall, denn wir nutzen aus, dass die genannten Werkzeuge für bestimmte Zwecke spezialisiert, klein und leicht zu bedienen sind. Schließlich schlägt man einen Nagel mit dem Hammer in die Wand und nur im Notfall mit der Zange, wenn kein Hammer zur Hand ist.

„Supersoftware“, die lt. Begleitbrief alles kann und über Knöpfe, hinter denen alles steckt, bedient wird, wäre hier nicht die erste Wahl, falls sie denn wirklich existiert.
Erfreulich und in der Sache förderlich ist die Beobachtung, dass Studierende der Informatik bereits im Grundstudium in der Lage sind, sich sehr rasch in den Gebrauch neuer Werkzeuge einzuarbeiten und die dabei erworbenen Fähigkeiten über längere Zeit zu bewahren. Den Studierenden hat die Benutzung der unterschiedlichen Tools keinerlei Schwierigkeiten bereitet. Manuals und how-to-do's sind im Web allgegenwärtig.
Die meisten Programme wurden mit SCHEME geschrieben. SCHEME ist eine Sprache aus der Lisp-Familie und zeichnet sich durch semantisch mächtige Ausdrucksmittel bei extrem einfacher Syntax aus. Dies erkennt man allein schon daran, dass die angegebenen SCHEME-Programme keinen größeren Umfang einnehmen als hierfür üblicherweise verwendeter Pseudocode, der – im Unterschied zu den SCHEME-Prozeduren – in der angegebenen Form nicht direkt ausführbar wäre.
Für Leser, die noch nicht über Kenntnisse in der Programmierung mit SCHEME verfügen, wird folgendes Vorgehen empfohlen:

1. Absolvieren Sie einen interaktiven Einstiegskurs in die funktionsorientierte Programmierung mit Scheme, zu dem Sie über die o.g. Buch-Homepage gelangen. Hierfür benötigen Sie lediglich den von Ihnen bevorzugten WWW-Browser (ohne lokale SCHEME-Installation) und ca. eine[1] Stunde Zeit.
2. Installieren Sie das (Chez-)SCHEME-System auf Ihrem Rechner. Stellen Sie sicher, dass auch die Hilfen (SCHEME-Manual usw.) zum späteren Nachschlagen verfügbar sind.
3. Kopieren Sie die angegebenen Prozeduren in das SCHEME-System und experimentieren Sie damit: Folgen Sie zunächst den Aufrufbeispielen im Text, modifizieren Sie die Aufrufe und kommentieren Sie die Prozedurdefinitionen.
4. Entwickeln Sie selbst SCHEME-Prozeduren nach dem Vorbild der im Text angegebenen Beispiele. Folgen Sie dabei den Übungsaufgaben.

Der Text des Buches enthält *breite rechte Ränder* mit herausgezogenen Stichworten. Diese Ränder haben nicht in jedem Falle die Aufgabe, einen neuen tragenden Begriff anzuzeigen bzw. dessen Einführung im zugehörigen Textteil zu betonen. Die Marginalien sollen vielmehr die *Gliederung des Textes* optisch unterstützen, sodass beim Zurückschlagen eine schnelle Orientierung möglich wird. Außerdem bieten breite Ränder Platz für Notizen. Lehrenden helfen diese Stichpunkte, um den „roten Faden“ ihrer Vorlesung im Auge zu behalten.
Zum Verständnis dieses Buches werden vom Leser *Grundkenntnisse in der Programmierung* sowie die *Fähigkeit und Bereitschaft zur Anwendung mathematischer Methoden* erwartet, wie sie bis zum Abitur bzw. in den Anfangssemestern an Hochschulen

[1] Aus dieser Angabe ist der Schluss zu ziehen, dass sich die Lernkurve für die Einarbeitung in die Programmierung mit SCHEME deutlich von der anderer Programmiersprachen, etwa für Java, unterscheidet. Schon nach kurzer Zeit werden Sie in der Lage sein, SCHEME-Programme zu verstehen und einfache Prozeduren selbstständig zu schreiben. Die Schwierigkeiten des Anfängers liegen erfahrungsgemäß im mentalen Kampf gegen eingeschliffene Denkweisen, was sich insbesondere beim Entwurf rekursiver Prozeduren offenbart.

vermittelt werden. Obwohl es im Text einige (wenige) Bezüge zu Inhalten aus anderen Gebieten der theoretischen Informatik gibt, werden dementsprechende Vorkenntnisse nicht vorausgesetzt. Für Kapitel 10 gilt dies naturgemäß nur mit Einschränkung.
Nach Ansicht des Autors ist das Buch für *Studierende der Informatik* im Grundstudium an *Fachhochschulen* ebenso geeignet, wie an *Berufsakademien* und *Universitäten*. Darüber hinaus kann es *Weiter- und Fortbildungsmaßnahmen* unterstützen. Dies betrifft die *vorlesungsbegleitende* Verwendung des Materials ebenso wie das *Selbststudium*. Auch *Lehrende gymnasialer Oberstufenkurse* sollen hier inhaltliche und didaktische Anregungen finden.
Ich danke allen Studierenden, die in irgendeiner Form durch individuelle Beiträge an der Entstehung des Manuskripts oder Web-Materials beteiligt waren. Jegliche Bemerkungen und Hinweise sind auch weiterhin willkommen!
Herrn Professor Kerner gebührt mein Dank dafür, dass er mich schon als Student mit Aspekten der theoretischen Informatik vertraut machte, und zwar in einer Weise, die neben der rein wissenschaftlichen auch immer die didaktische Sicht integrierte. Über die Jahre folgten viele Diskussionen.
Dem Einfluss von Herrn Professor Löthe verdanke ich die Bekanntschaft mit SCHEME, deren Nutzung ich inzwischen zu einem didaktischen Vehikel für das Studium der theoretischen Informatik weiterentwickelt habe.
Nicht unerwähnt seien die geduldigen Versuche meines Sohnes Martin, mich mit den Grundlagen der Molekulargenetik vertraut zu machen.
Verlagsseitig gilt Frau Hotho und Herrn Professor Lutz (Hrsg.) mein besonderer Dank für klare Hinweise nach gründlicher Durchsicht des Manuskripts und unkomplizierte Zusammenarbeit. Frau Hotho versteht es treffsicher, durch verlässliche Aussagen einen Rahmen zu schaffen, in dem die Motivation zur (nicht immer kreativen) Arbeit an einem Buch nie vertrocknet. Schließlich gilt auch für dieses Projekt: “It’s easier said than done.”

> ... and if you don’t believe it, try proving that it’s easier done than said, and you’ll see that “it’s easier said that ‘it’s easier done than said’ than it is done”, which really proves that “it’s easier said than done”.

Christian Wagenknecht
c.wagenknecht@hs-zigr.de Februar 2003