

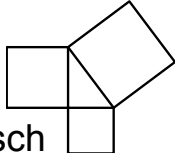
Jochen Ziegenbalg
Oliver Ziegenbalg
Bernd Ziegenbalg

Algorithmen

von Hammurapi bis Gödel

Mit Beispielen aus den Computeralgebra Systemen
Mathematica und Maxima

Verlag
Harri
Deutsch



Die Autoren

Prof. Dr. Jochen Ziegenbalg, geb. am 24.06.1944 in Dresden, ist Hochschullehrer für Mathematik und Informatik an der Pädagogischen Hochschule Karlsruhe. Näheres zu seiner Person und seiner Hochschullehrertätigkeit ist unter der Internet-Adresse

<http://www.ziegenbalg.ph-karlsruhe.de>

zu finden. Er ist per „electronic mail“ unter ziegenbalg@ph-karlsruhe.de

erreichbar und er freut sich über Kommentare und Anregungen jeglicher Art.

Oliver Ziegenbalg, geb. am 19.09.1971 in Böblingen, hat an der Universität Karlsruhe Wirtschaftsmathematik und an der Hochschule für Gestaltung, Karlsruhe, Medienkunst / Film studiert. Er ist jetzt als freier Drehbuchautor tätig.

Bernd Ziegenbalg, geb. am 19.09.1971 in Böblingen, hat an der Universität Karlsruhe Mathematik und Informatik und an der Universität Mainz Journalismus studiert. Er ist jetzt für raufeld medien, Berlin, tätig.

Die Autoren sind in der Reihenfolge ihres Alters aufgeführt.

Bibliografische Informationen der Deutschen Nationalbibliothek

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

ISBN 978-3-8171-1864-9

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches – oder von Teilen daraus – sind vorbehalten.

Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) reproduziert oder unter Verwendung elektronischer Systeme verarbeitet werden. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

Der Inhalt des Werkes wurde sorgfältig erarbeitet. Dennoch übernehmen Autoren, Herausgeber und Verlag für die Richtigkeit von Angaben, Hinweisen und Ratschlägen sowie für eventuelle Druckfehler keine Haftung.

1. Auflage: 1996 Spektrum Akademischer Verlag, Heidelberg Berlin Oxford

2., verbesserte Auflage 2007

3., überarbeitete Auflage 2010

© Wissenschaftlicher Verlag Harri Deutsch GmbH, Frankfurt am Main, 2010

Druck: fgb – freiburger graphische betriebe: www.fgb.de

Printed in Germany

Ergänzende Materialien, Lösungen für ausgewählte Aufgaben, Druckfehlerberichtigungen und ähnliches sind zu finden unter der Adresse:

<http://www.ziegenbalg.ph-karlsruhe.de>

Vorwort zur 3. Auflage

Mit der dritten Auflage dieses Buches geht (neben der stets anfallenden Verbesserung von Druckfehlern) eine Reihe von Änderungen inhaltlicher Art einher. Rein äußerlich ist das Buch etwas schlanker geworden. Dies ist der nunmehr durchgängigen Darstellung der Algorithmen in der Syntax der Computeralgebra Systeme *Mathematica*¹ und *Maxima*² geschuldet. Auf die in den ersten beiden Auflagen dieses Buches verwendete, seinerzeit gut verfügbare Programmiersprache *Pascal* wurde in der dritten Auflage verzichtet, da mit den in jüngster Zeit entwickelten, außerordentlich mächtigen, flexiblen, reichhaltig ausgestatteten und benutzerfreundlichen Programmiersystemen *Mathematica* und *Maxima* (um nur einige der inzwischen gut verfügbaren Computeralgebra Systeme zu nennen) modernere Werkzeuge zur Verfügung stehen. Computeralgebra Systeme sind heute für Aufgaben im Bereich von Mathematik und Naturwissenschaften Programmierwerkzeuge der ersten Wahl. Es sind in der Regel voll entwickelte, universelle algorithmische Sprachen (wenn man so will: Programmiersprachen) mit einer besonderen Stärke im Bereich der Symbolverarbeitung. Für viele mathematische Zwecke unerlässlich ist ihre den herkömmlichen Programmiersprachen weit überlegene numerische Genauigkeit; für zahlentheoretische Probleme ist die exakte Ganzzahlarithmetik unverzichtbar.

Darüber hinaus bieten Computeralgebra Systeme meist eine exzellente Graphik- und Audio-Unterstützung. Ihre hochgradige Interaktivität macht den Einstieg besonders für den gelegentlichen Nutzer vergleichsweise angenehm. Viele Unannehmlichkeiten bzw. Unzulänglichkeiten älterer Programmiersysteme, wie z.B. starre und oft sehr restriktiv gehandhabte Datentypen, inkorrekte numerische Ergebnisse (aufgrund der eingebauten „Festwort“-Arithmetik) und ähnliches mehr gehören damit der Vergangenheit an.

Für das, was sie leisten, sind praktisch alle Computeralgebra Systeme preiswert. Aber besonders angenehm ist natürlich der Umstand, dass es darunter auch sehr leistungsfähige kostenlose Open Source

¹ Nähere Informationen zu *Mathematica*: <http://www.wolfram.com>

² Nähere Informationen zu *Maxima*: <http://maxima.sourceforge.net>

Beispiele und weitere Informationen: <http://www.ziegenbalg.ph-karlsruhe.de>

Systeme gibt – wie z.B. das in diesem Buch verwendete System Maxima. Für die Nutzbarkeit von Maxima ist es hilfreich, dass nunmehr (mit wxMaxima) eine recht intuitiv bedienbare Programmieroberfläche zur Verfügung steht. Natürlich gibt es auch in Maxima eine Reihe von Fußangeln für Anfänger. Einige Hinweise, die Erstbenutzern das Leben erleichtern sollen, finden sich im Internet unter der Adresse www.ziegenbalg.ph-karlsruhe.de.

Da sich Computeralgebra Systeme oft an den Paradigmen der Programmiersprache Lisp orientieren (vgl. 8.7.2), unterstützen sie in der Regel besonders gut die Technik des *funktionalen* Programmierens. Dazu gehört primär eine angemessene Implementierung des Funktionskonzepts in Verbindung mit Listenverarbeitung, Rekursion und Symbolverarbeitung. Aber auch wenn Algorithmen durchgängig mit Hilfe von Computeralgebra Systemen realisiert werden, ist damit noch nicht die Frage entschieden, welches *Programmierkonzept* (Programmierparadigma, vgl. 8.3) dabei im Vordergrund stehen soll. So unterstützt z.B. Mathematica auch das *regelbasierte* Programmieren sehr gut. Im vorliegenden Buch gehen die Präferenzen eindeutig in Richtung des funktionalen Programmierens; allerdings nicht in dogmatischer Form. Dort, wo es angemessen erscheint, werden auch imperative Lösungen aufgezeigt (so z.B. beim Sieb des Eratosthenes). Vom Konzept der „anonymen Funktionen“, das aus der Welt der funktionalen Programmierung stammt und mittlerweile auch in anderen Programmiersprachen Eingang gefunden hat, wird praktisch kein Gebrauch gemacht.

Wenn in diesem Buch einige besonders wichtige Algorithmen (wie z.B. der Euklidische Algorithmus) parallel in mehreren algorithmischen Varianten dargestellt werden, so soll dies der Vermittlung von Kritik- und Urteilsfähigkeit dienen. Denn urteilsfähig wird man nicht allein dadurch, dass man einen bestimmten Programmierstil möglichst gut erlernt und beherrscht, sondern dadurch, dass man darüber hinaus (zumindest exemplarisch) zur Kenntnis nimmt, welche anderen wichtigen Sprachkonzepte es noch gibt.

Im vorliegenden Buch wurden funktional formulierte Programme eher in Mathematica und imperativ formulierte Programme (vgl. 8.3) eher in Maxima realisiert, dessen Syntax (besonders im Bereich der Kontrollstrukturen) der von Pascal oft recht ähnlich ist. Die Form der imperativen Lösung wurde besonders dann gewählt, wenn der Datentyp des Feldes verwendet wurde. Dies soll aber nicht heißen, dass

Maxima primär das imperative Programmierparadigma favorisiert; im Gegenteil; als ein auf der Programmiersprache Lisp basierendes System unterstützt es natürlich auch die funktionale Programmierung in hervorragender Weise.

Dieses Buch hat das Ziel, in die Grundlagen und die Geistesgeschichte des Algorithmierens einzuführen. Wenn man heute einen Algorithmus formuliert hat, dann gibt es gute Gründe dafür, ihn auch laufen lassen zu wollen. Dazu muss man den Algorithmus in die Form eines lauffähigen Programms bringen, und so wird im folgenden in der Regel auch verfahren. Das Buch versteht sich aber nicht als eine systematische Einführung in eine konkrete Programmiersprache – dazu gibt es eine Fülle von Spezialliteratur.

Programmieren heißt immer auch, dass man sich mit Fragen der *Effizienz* auseinanderzusetzen hat (vgl. Kapitel 5). In diesem Buch stand bei der Formulierung der Programme vorrangig die gute Verständlichkeit im Vordergrund, also die „kognitive“ Effizienz (vgl. 5.2) und nicht primär die Laufzeit- oder Speicherplatzeffizienz.

Schließlich möchte ich an dieser Stelle denjenigen danken, die den Entstehungsprozess dieses Buches unterstützt und gefördert haben: zuallererst meiner Frau Barbara für ihre Geduld und moralische Unterstützung, meinen Söhnen Oliver und Bernd für vielfältige produktive Diskussionen und die Einbringung ihrer Kenntnisse über evolutionäre Algorithmen und neuronale Netze. Christian Stellfeldt, Thomas Borys, Elke Bernsee und vielen meiner Studierenden sei dafür gedankt, dass sie das Manuskript kritisch gelesen und kommentiert haben. Last not least danke ich Herrn Klaus Horn vom Verlag Harri Deutsch für seine kompetente, konstruktive und freundliche Begleitung des Herstellungsprozesses.

Berlin, im Juli 2010

Jochen Ziegenbalg

Inhalt

Einleitung	11
1 Vorbemerkungen: Stellenwert des Themas, Hintergründe, Begründungs- und Bedeutungszusammenhänge, Ziele	18
2 Begriffsbestimmungen	24
2.1 Zum Begriff des Algorithmus	24
2.2 Zum Begriff der Informatik	31
3 Historische Bezüge	35
3.1 Ein Exkurs zur Geschichte der Algorithmik und Informatik	35
3.1.1 Zur Entwicklung der schriftlichen Rechenverfahren	48
3.1.2 Algebraisierung – die Idee der „ars magna“	51
3.1.3 Zur Geschichte der Rechenmaschinen	52
3.2 Vier klassische Algorithmen	56
3.2.1 Das sumerisch-babylonische Wurzelziehen bzw. Heron-Verfahren	57
3.2.2 Der Euklidische Algorithmus	62
3.2.3 Das Sieb des Eratosthenes	72
3.2.4 Die Approximation von π nach Archimedes	78
3.3 Algorithmisches Definieren und Beweisen	88
3.3.1 Die Unendlichkeit der Primzahlmenge	94
3.3.2 Inkommensurabilitätsbeweise	95
4 Fundamentale heuristische Strategien des algorithmischen Problemlösens	99
4.1 Elementare Methoden	99
4.1.1 Die Methode der rohen Gewalt (brute force method)	100
4.1.2 Die gierige Strategie (greedy strategy)	102
• Ägyptische Bruchrechnung	102
• Arbeitsplanung (job scheduling)	107
• Konstruktion eines minimalen Gerüsts (minimal spanning tree)	109
4.2 Methoden, die sich stark am Einsatz von Computern orientieren	114
4.2.1 Modularität	114
4.2.2 Rekursion	118
• Das Turm-von-Hanoi Spiel	121

4.2.3	Das Prinzip „Teile und Herrsche“ (divide et impera, divide and conquer)	123
	• Quicksort	124
4.3	Methoden, die im Zusammenhang mit der Bearbeitung von Bäumen und Graphen zur Anwendung kommen	127
4.3.1	Systematisches Durchlaufen von Baumstrukturen	128
	• Algorithmus Tiefensuche	132
	• Algorithmus Breitensuche	135
	• Das Rucksack-Problem (knapsack problem)	136
4.3.2	Backtracking	141
	• Das Damenproblem	141
4.4	Die gezielte mathematische Analyse	145
	• Das NIM-Spiel	146
4.5	Probabilistische Verfahren, Modellbildung und Simulation	148
	• Das Sammlerproblem	152
	• Das Ziegenproblem	154
4.6	Parallelität	158
5	Effizienz von Algorithmen	159
5.1	Iteration und Rekursion unter dem Gesichtspunkt der Effizienz	160
5.2	Kognitive Effizienz	165
5.3	Das Prinzip von „Teile und Herrsche“ unter dem Aspekt der Effizienz	166
	• Schnelles Potenzieren	167
5.4	Das Horner-Schema	172
	• Erster Exkurs: Stellenwertsysteme	175
	• Zweiter Exkurs: Die Einkommensteuer	177
5.5	Die Zeitkomplexität des Euklidischen Algorithmus	178
5.6	Einige wichtige Funktionstypen zur Beschreibung der Effizienz von Algorithmen	180
5.7	Algorithmisch aufwendige Probleme	183
	• Das Königsberger Brückenproblem	184
	• Eulersche und Hamiltonsche Wege	184
	• Das Traveling Salesman Problem	189
	• Entscheidungsprobleme und Optimierungsprobleme	189
	• Die Komplexitätsklassen P, NP und NP-vollständig	190

6	Korrektheit von Computerergebnissen	
	Korrektheit von Algorithmen	195
6.1	Fehler in der Arithmetik von Computern	197
6.2	Partielle und totale Korrektheit von Algorithmen	205
6.3	Formale Methoden	206
7	Grenzen des Computers / Grenzen der Algorithmisierbarkeit	210
7.1	Entwicklung der wissenschaftstheoretischen Grundideen	213
7.2	Formalisierung des Algorithmus-Begriffs / der Begriff der Berechenbarkeit	225
7.3	Einige konkrete, algorithmisch nicht lösbare Probleme	226
	• Das Halteproblem	227
8	Programmierung	230
8.1	Zum Verhältnis von „Maschinensprachen“ und „höheren“ Programmiersprachen	231
8.2	Wie werden die in einer höheren Programmiersprache geschriebenen Programme verarbeitet?	236
8.3	Paradigmen des Programmierens Programmiersprachen-Familien	238
8.4	Die wichtigsten Kontrollstrukturen in strukturierten Programmiersprachen	241
	• Die Anweisungsfolge (Sequenz)	242
	• Die Fallunterscheidung (Auswahl, Verzweigung)	242
	• Die Wiederholung (Schleife)	243
	• Kontrollstrukturen und Modularität	244
	• Der Sprungbefehl	244
	• Strukturiertes Programmieren	245
	• Flussdiagramme	245
8.5	Die wichtigsten Datenstrukturen	248
	• Numerische Datentypen	251
	• Der Datentyp des Feldes	252
	• Der Datentyp des Verbunds	252
	• Der Datentyp der Liste	253
8.6	Modulares Programmieren mit Prozeduren und Funktionen	255
	• Prozeduren	256
	• Funktionen	257

8.7	Diskussion einiger konkreter Programmiersprachen	261
8.7.1	Die Familie der ALGOL-ähnlichen Programmiersprachen	261
	• Pascal	263
8.7.2	Programmiersprachen aus dem Bereich der „Künstlichen Intelligenz“	265
	• Lisp	265
	• Logo	267
	• Scheme	268
	• Prolog	270
	• Computeralgebra Systeme	274
8.7.3	Das Phänomen der Interaktivität	274
	• BASIC	275
8.8	Programmierumgebungen, Betriebssysteme, Benutzerschnittstellen und Anwendersysteme	277
9	Evolutionäre Algorithmen und neuronale Netze	280
9.1	Evolutionäre Algorithmen	281
	• Die Methode der evolutionären Algorithmen – erläutert am Traveling Salesman Problem	287
9.2	Neuronale Netze	297
9.2.1	Backpropagation-Netze	309
9.2.2	Rückgekoppelte Netze (Hopfield-Netze)	315
	• Mustererkennung mit rückgekoppelten Netzen	318
9.2.3	Selbstorganisierende Netze (Kohonen-Netze)	323
	• Selbstorganisierende Netze am Beispiel des Traveling Salesman Problems	328
	Bildquellen	334
	Literaturverzeichnis	335
	Index	343

Einleitung

Die folgende Abbildung soll die Verwobenheit des Themas „Algorithmen“ mit weiteren zentralen Themen unserer Geistesgeschichte symbolisieren.



Um diese Verwobenheit geht es im vorliegenden Buch. Die Darstellung versinnbildlicht, warum die Themenkreise *Algorithmen* und *Computer* auch zum Kanon unserer Allgemeinbildung gehören sollten: Sie stehen, wie im folgenden auszuführen sein wird, in einer regen Wechselwirkung mit weiteren zentralen Themen unseres geistigen Lebens.

Die bislang zum Themenkomplex *Mathematik, Algorithmen, Computer, Informatik* vorgelegten Bildungsangebote stellen ein wildes

Konglomerat von Beiträgen dar, die sich zwischen den Polen *zeitloser mathematischer Grundkenntnisse* und *extrem vergänglicher Computerkenntnisse* bewegen. Besonders was die computerbezogenen Kenntnisse betrifft, kommt zudem eine Woge bisher nicht gekannter Kommerzialisierung auf die Bildungslandschaft zu.

Allein schon deshalb, aber auch, weil die für Unterricht und Studium zur Verfügung stehende Zeit eine sehr knappe Ressource ist, wird es notwendig, die Kriterien und Inhalte der Allgemeinbildung immer wieder kritisch zu durchdenken.

Der Computer (einschließlich seiner Software) ist ein hochgradig komplexes und mit einer gewissen, außerordentlich zeitabhängigen Beliebigkeit konstruiertes Gerät, das uns aber in die Lage versetzt, zeitlose fundamentale Ideen der Mathematik und Informatik zu realisieren. Bildungsrelevant im Zusammenhang mit dem Computer sind diese bleibenden grundlegenden Ideen – nicht jedoch die extrem schnell veraltenden Kenntnisse über dieses oder jenes Detail seiner Hardware, über gerade gängige Formen dieses oder jenes Betriebssystems oder die Besonderheiten graphischer bzw. sonstiger „Benutzerschnittstellen“.

Wenn hier die Auffassung vertreten wird, dass Algorithmen im Zentrum unserer Wissenschafts- und Kulturgeschichte stehen, dann soll damit nicht gesagt sein, dass nur sie dieses Zentrum sind. Die Abbildung auf der vorigen Seite enthält zentrale bildungswirksame Begriffe; es sind Facetten im Gesamtmosaik dessen, was Bildung ausmacht. Dieses „Bildungsmosaik“ ist außerordentlich vielgestaltig; die Frage, was zu einer guten, richtigen, angemessenen Bildung gehört, wird von den unterschiedlichsten Standpunkten her permanent diskutiert – aber kaum jemand wird heute in Frage stellen, dass Kenntnisse in Mathematik, Informatik, Naturwissenschaften und Technik *auch* dazu gehören.

Für viele Menschen ist die Algorithmik (also die Lehre von und die Beschäftigung mit den Algorithmen) ein esoterisches Spezialphänomen unserer vom Computer durchdrungenen Zeit, das nur interessant zu sein scheint für hochgradig spezialisierte Berufsgruppen, insbesondere Computerprogrammierer. Dieses Bild ist jedoch nicht nur unzutreffend; das Gegenteil ist eher richtig. Mit den ihnen innewohnenden Eigenschaften der Elementarität, Konstruktivität und Beziehungshaltigkeit (Vernetztheit und Vernetzbarkeit), mit ihrer Förderung experimenteller und operativer Arbeitstechniken stehen Algorithmen im

methodologischen Zentrum mathematischer Bildung – und weit darüber hinaus.

Die Entstehung von Algorithmen hat zunächst einmal gar nichts mit dem Phänomen des Computers zu tun. Wie in den historischen Ausführungen (in Kapitel 3) noch zu zeigen sein wird, sind die Algorithmen sehr viel älter als die Computer. Der Entstehung „des“ Computers, wie wir ihn heute kennen, liegt geradezu das Motiv zugrunde, ein Gerät zur Ausführung der schon früher entwickelten Algorithmen zur Verfügung zu haben. Bis zum heutigen Tag gibt es viele auf dem Gebiet der Algorithmik arbeitende Mathematiker und Informatiker, für die der Computer bestenfalls am Rande eine Rolle spielt. Der renommierte Mathematikhistoriker H. M. Edwards vertritt die wohlbegründete Auffassung, dass alle Mathematik bis zum Auftreten der Zeitgenossen von Leopold Kronecker (1823–1891) algorithmischer Natur war (vgl. Edwards 1987).

Einen der frühesten schriftlichen Belege für Algorithmen bietet die von Euklid im dritten Jahrhundert v. Chr. unter dem Titel „Die Elemente“ verfasste Gesamtdarstellung des mathematischen Wissens seiner Zeit (die neben vielen anderen natürlich auch den „Euklidischen“ Algorithmus enthält). Euklids Werk war eines der erfolgreichsten Lehrbücher in der Bildungsgeschichte der Menschheit; es wurde bis in das 19. Jahrhundert hinein als Lehrbuch für Mathematik verwendet – ein Lehrbuch mit einer „Lebenszeit“ von über 2000 Jahren!

Im Bildungszusammenhang ist weiterhin der Umstand bemerkenswert, dass auch der Algorithmus-Begriff aus einer ureigenen pädagogischen Intention heraus entstanden ist. Der Begriff des Algorithmus geht zurück auf die Angabe des Geburtsorts („al Khowarizm“) desjenigen persisch-arabischen Gelehrten, der im neunten Jahrhundert ein wissenschaftsgeschichtlich äußerst einflussreiches Buch zur Verbreitung der „indischen“ Ziffern, also der Zahldarstellung im Zehnersystem, geschrieben hat (man vergleiche dazu auch die Ausführungen zur Etymologie in Kapitel 2). Die Art und Weise, wie wir heute die Zahlen schreiben und mit ihnen umgehen, geht entscheidend auf dieses Buch zurück, das mithin neben Euklids Elementen einen der wichtigsten Ecksteine in der Bildungsgeschichte der Menschheit darstellt.

Die Algorithmik hat, etwas vereinfachend ausgedrückt, zwei „Gesichter“. Zum einen gehört dazu der Entwurf, die Konstruktion von (neuen) Algorithmen sowie das Verstehen, Nachvollziehen und Ana-

lysieren vorgegebener, insbesondere klassischer Algorithmen. Soll die Beschäftigung mit der Algorithmik jedoch nicht nur theoretischer Natur bleiben, so gehört zum anderen aber auch die Abarbeitung von fertigen, vorgegebenen Algorithmen zum Gesamtgebiet der Algorithmik.

Der Entwurf von Algorithmen ist eine höchst kreative Aktivität; ihre Abarbeitung dagegen meist eine sehr langweilige Sache. Deshalb gehörte zur Algorithmik schon immer der Versuch, Maschinen zur Abarbeitung von Algorithmen zu konstruieren. Die Universalmaschine zur Abarbeitung von Algorithmen ist heute der Computer; er stellt ein unentbehrliches Werkzeug für das algorithmische Problemlösen dar. Die Formen des empirischen, experimentellen, ja sogar spielerischen Arbeitens, die auch in der im Prinzip deduktiven Wissenschaft der Mathematik von großer Bedeutung sind, hängen heute ganz massiv mit der Nutzung des Computers im Lern- und Bildungsprozess zusammen. Der Computer ist das wichtigste Werkzeug, um in der Mathematik Experimente durchzuführen. Er ist aber noch mehr. Vor allem im Zusammenhang mit dem *modularen* Arbeiten, also dem Arbeiten im Sinne des Baukastenprinzips, ist er ein Katalysator, der eine neue, intensivere begriffliche Durchdringung vieler klassischer und moderner Probleme ermöglicht und erzwingt. Die Umsetzung von Algorithmen auf einem Computer geschieht in der Form der *Programmierung* des Computers. Es gibt viele Formen und Varianten des Programmierens. Programmieren „mit Stil“ ist eine anspruchsvolle geistige Aktivität, die der Pädagoge H. von Hentig in der ZEIT vom 18. Mai 1984 mit eindrucksvollem Blick für das Wesentliche folgendermaßen beschreibt:

„Dies ist der Gewinn – aber er wird mir nur zuteil, wenn ich den Computer dazu verwende: als Abbild meiner Denkprozesse, die ich in ihm objektiviere und erprobe. Programmieren heißt eben dies.“

Die Entwicklung moderner Programmiersprachenkonzepte steht im engsten Zusammenhang mit den fundamentalen Paradigmen des Algorithmus-Begriffs und den philosophischen Grundfragen, die sich aus der Untersuchung der Grenzen der Algorithmisierbarkeit ergeben haben. Der österreichische Philosoph Ludwig Wittgenstein (1889–1951) hat sich in der ersten Hälfte des vorigen Jahrhunderts intensiv mit der Rolle der Sprache (und dabei insbesondere auch der formalen

Sprachen) für unser Denken befasst. Von ihm stammt der Ausspruch „Die Grenzen der Sprache sind die Grenzen der Welt“, der auch in der „Welt des Computers“ seine Berechtigung hat.

Wenn heute von der Rolle des Computers im Bildungswesen die Rede ist, dann geschieht dies aus ganz unterschiedlichen Perspektiven heraus. Eine davon, nämlich die Rolle des Computers als *Werkzeug für das algorithmische Arbeiten und Problemlösen*, haben wir bereits angesprochen. Daneben wird der Computer aber auch oft nur als *Präsentations-* und *Display-Medium* genutzt; das „nur“ wurde hier deshalb verwendet, weil dies eben nicht seine Einzigartigkeit als Werkzeug zur Unterstützung der menschlichen Denk- und Problemlösefähigkeiten ausmacht. Veranschaulichung und Visualisierung sind zweifellos wichtig für jede Form des Lernens; sie spielen insbesondere auch im Mathematikunterricht eine große Rolle. *Die Kunst des Sehens in der Mathematik* ist der Titel eines wohlbekannten Buches des italienischen Mathematikers Bruno de Finetti; schon der Titel macht die Bedeutung der Veranschaulichung von Mathematik deutlich. Sehen ist aber mehr als bloße „Visualisierung“ – sehen heißt auch: Zusammenhänge erkennen, Fundamentales identifizieren, Wichtiges von Unwichtigem trennen, komplexe Sachverhalte strukturieren können, zu einer „Gesamtschau“ fähig sein. So hat z.B. *Leonhard Euler* (1707–1783), einer der produktivsten Mathematiker aller Zeiten, in der Mathematik so viel „gesehen“, wie kaum ein anderer – obwohl er fast blind war.

Nichtsdestoweniger ist auch die reine Visualisierung wichtig. Aber wenn der Computer nur dazu verwendet wird, dann wird er in ähnlicher Weise benutzt wie ein Overhead-Display, wie ein (zugegebenermaßen sehr komfortables) Filmgerät, wie ein Videorecorder oder wie ähnliche Präsentationsmedien. Er hat diesen Geräten (wenn er richtig programmiert ist und sinnvoll verwendet wird) immer noch das Moment der „Interaktivität“ voraus, aber seine eigentliche Stärke als „Denkwerkzeug“ kommt durch diese Nutzungsformen nicht oder nur ungenügend zur Geltung.

Es ist selbstverständlich, dass dieses Buch in einem Spannungsverhältnis zur *Bildungsinformatik* steht. An Stelle einer expliziten Definition dieses etwas schillernden Begriffs versteht sich das vorliegende Buch als Beitrag zu dessen impliziter Klärung durch Beschreibung eines ihrer wichtigsten Themenkreise. Es ist wohl richtig, dass Algo-

rithmik nicht gleich Bildungsinformatik ist; aber sie ist eine ihrer wichtigsten Facetten.

Wodurch hebt sich nun aber eine Darstellung zur Bildungsinformatik von einer zur „reinen“ Informatik ab, wodurch unterscheidet sich dieses Buch von Darstellungen zur Informatik – ohne den *Bildungs-Zusatz*? Dies geschieht vorrangig durch die verstärkte Berücksichtigung geisteswissenschaftlicher Themenkreise. So wurde im vorliegenden Buch besonderer Wert gelegt auf die ausführliche Darstellung historischer Entwicklungen, auf philosophische Hintergründe und Zusammenhänge, auf die Heuristik des algorithmischen Problemlösens und auf eine exemplarische, am konkreten Beispiel orientierte Entwicklung vieler abstrakter Ideen der Informatik. Die Beispiele sind zudem so ausgewählt, dass sie im weitesten Sinne als bildungsrelevant bezeichnet werden können. Hierzu gehören viele schulisch und historisch bedeutsame Algorithmen: das Verfahren des babylonischen Wurzelziehens, der Euklidische Algorithmus, das Sieb des Eratosthenes, das Verfahren von Archimedes zur Approximation der Kreiszahl π und vieles mehr. Bildungsrelevant ist aber auch eine Reihe von Themen, die im Kanon unseres derzeitigen Schulwissens eher unterrepräsentiert sind, wie z.B. der gesamte für die Informationsverarbeitung enorm wichtige Themenkomplex: *Bäume, Netze und Graphen*. Auch die Würdigung des Themas *heuristische Strategien* lässt im derzeitigen Schulsystem sehr zu wünschen übrig. Die Behandlung der Strategien des Problemlösens hat im Schulsystem keinen zentralen „Ort“. Es gibt kein Schulfach *Heuristik*, und der Einführung eines solchen Faches soll hier auch nicht das Wort geredet werden, denn heuristische Strategien sind in der Regel fächerübergreifender Natur. Heuristik müsste inhaltsbezogen im Fachunterricht thematisiert werden. Dies geschieht aber bestenfalls nur sporadisch; zum einen, weil die Lehrpläne (ob tatsächlich oder vermeintlich, sei dahingestellt) keinen Raum dafür bieten, zum anderen, weil sich auch die Studierenden nur in den seltensten Fällen während ihres Studiums explizit mit heuristischen Fragen befassen – bzw. befassen müssen. Heuristik lässt sich nur schwer in Lehr- und Studienpläne gießen. Und es ist nicht einfach, Kenntnisse in Heuristik abzu prüfen. Dennoch ist ein Wissen über heuristische Strategien von großer Bedeutung für die Zukunft der Lernenden – vielleicht wichtiger als manches fachliche Detail, das sehr bald wieder vergessen wird.

Eine gewisse Sonderrolle spielt in diesem Zusammenhang jedoch die Informatik und speziell das Teilgebiet der „künstlichen Intelligenz“ (es ist hier nicht der geeignete Ort, um auf die Problematik dieses schillernden Begriffs näher einzugehen). Im Rahmen der Versuche, das Phänomen des „intelligenten Verhaltens“ zu simulieren, kommt die Thematisierung heuristischer Verfahren auf ganz natürliche Weise ins Spiel. Die Informatik ist also möglicherweise nicht der einzige, aber auf jeden Fall ein ausgezeichneter Ort, um der Heuristik eine geeignete Heimstatt zu geben – und wenn dies erfolgt, dann ist das ein wesentlicher Schritt, damit aus einer *reinen* Informatik eine *Bildungs*-Informatik wird.