

国外著名高等院校
信息科学与技术优秀教材



计算机科学概论(第9版) Computer Science

An Overview 9th Edition

〔美〕 J. Glenn Brookshear 著



(英文版)



人民邮电出版社
POSTS & TELECOM PRESS

国外著名高等院校信息科学与技术优秀教材

计算机科学概论 (第9版)

(英文版)

Computer Science: An Overview, 9th Edition

[美] J. Glenn Brookshear 著

人民邮电出版社
北京

图书在版编目 (CIP) 数据

计算机科学概论: 第9版. 英文 / (美) 布鲁克希尔
(Brookshear, J. G.) 著. —北京: 人民邮电出版社,
2007.10

国外著名高等院校信息科学与技术优秀教材
ISBN 978-7-115-16492-6

I. 计... II. 布... III. 电子计算机—高等学校—教材—
英文 IV. TP3

中国版本图书馆 CIP 数据核字 (2007) 第 098509 号

版 权 声 明

Original edition, entitled **COMPUTER SCIENCE: AN OVERVIEW**, 9th Edition, 0321387015 by **BROOKSHEAR, J. GLENN**, published by Pearson Education, Inc, publishing as Addison-Wesley, Copyright © 2007 by Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

China edition published by **PEARSON EDUCATION ASIA LTD.**, and **POSTS & TELECOMMUNICATIONS PRESS** Copyright © 2007.

This edition is manufactured in the People's Republic of China, and is authorized for sale only in People's Republic of China excluding Hong Kong, Macau and Taiwan.

仅限于中华人民共和国境内 (不包括中国香港、澳门特别行政区和中国台湾地区) 销售。

本书封面贴有 Pearson Education (培生教育出版集团) 激光防伪标签。无标签者不得销售。

国外著名高等院校信息科学与技术优秀教材 计算机科学概论 (第9版) (英文版)

- ◆ 著 [美] J. Glenn Brookshear
责任编辑 李 际
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
三河市海波印务有限公司印刷
新华书店总店北京发行所经销
- ◆ 开本: 800×1000 1/16
印张: 38.25
字数: 870 千字 2007 年 10 月第 1 版
印数: 1—4 000 册 2007 年 10 月河北第 1 次印刷

著作权合同登记号 图字: 01-2007-3016 号

ISBN 978-7-115-16492-6/TP

定价: 49.00 元

读者服务热线: (010)67132705 印装质量热线: (010)67129223

内容提要

本书是计算机科学概论课程的一本经典教材，是作者多年教学经验的结晶，是国际上众多名校的指定教材。本书涉及计算机科学的方方面面，介绍了计算机硬件、软件、数据组织和计算理论等四个方面的内容，包括编码及计算机体系结构的基本原理、操作系统、计算机网络、算法、程序设计语言、数据结构和数据库、人工智能以及计算理论等。本书在内容编排上，在力求保持学科广度的同时，还兼顾主题的深度，并把握了最新的技术趋势。书中配有大量的图、表和示例以增强读者对知识的掌握，并提供了丰富的习题以加强学生的参与性——本书包含 1000 多个问题，用于复习、扩展讨论过的内容，或者提示以后会涉及的有关主题。


本书既适合国内的大专院校用作计算机基础课教材，也可以供有意在计算机方面发展的非计算机专业读者作为入门参考。

序 言

在计算机学科的大学教育中，有一门重要的课程，即该学科各专业的学生都必须学习的一门专业基础课，在我国一般称作“计算机概论”。学生从中学进入大学，开始正规而系统地学习计算机专业课程，需要首先对计算机科学技术的基础知识有一个概括而准确的了解，否则其他任何一门专业课的教学都会遇到许多障碍。所以这门课程对于计算机软件与理论、计算机体系结构、计算机应用技术等专业的教学都是非常重要的。

J. Glenn Brookshear 著的《计算机科学概论》(Computer Science: an Overview)就是这样一本适合作为上述课程教材的好书。该书在美国哈佛大学、加州大学等各所大学被采用。自第一版之后，作者根据计算机科学技术的新发展不断地对该书进行更新和补充，目前已经是第9版。书中介绍了计算机硬件、软件、数据库和计算理论等方面的内容。对这些内容的论述深浅适当，文字通俗易懂而又保持简练和准确；每一节都带有精心挑选的习题；给出的插图也颇具匠心，能够很好地表现书中阐述的内容。总之，这是一本很值得引进和推广的好教材。

在我国，改革开放以来计算机科学技术的学科建设和教材建设一直在稳步发展。各高校的教师为此付出了大量心血，写作出版了许多高质量的教材。其中有许多教材不但具有很好的学术水平，而且适合我国的国情与文化背景。同时，学习和借鉴国际上先进的科学技术和优秀文化，是培养人才的需要。有选择地引进国外的优秀教材，必将有效地促进我国教育事业的健康发展。这本书的影印出版，将对我国的计算机专业基础课的教学和教材建设起到良好的作用。它也可以用于非计算机专业的计算机教学和面向计算机产业界的技术培训。



中国科学院 院士
北京大学 信息与工程科学学部 主任

前言

本书是计算机科学的入门综述。在力求保持学科广度的同时，还兼顾主题的深度，对所涉及的主题给出中肯的评价。

读者

本书面向计算机科学以及其他各个学科的学生。大多数计算机科学专业的学生在最初的学习中都有这样一个误解，认为计算机科学就是程序设计和浏览网页，因为这基本上就是他们所看到的一切。实际上计算机科学远非如此。因此，在入门阶段，学生们需要了解他们主攻的这门学科所涉及内容的广度，这也正是本书的宗旨。本书使学生们对计算机科学有一个总体的概念——在这个基础上，他们可以谙熟该领域今后其他课程的特点以及课程之间的相互关系。事实上，本书采用的综述方式也是自然科学入门教程的常见模式。

其他学科的学生如果想融入这个技术化社会，也需要具备这些宽泛的知识背景。适用于他们的计算机科学课程提供的应该是对整个领域很实用的认识，而不仅仅是培训学生如何上网和使用一些流行的软件。当然这种培训也有其适用的地方，而本书的目的是用作教科书。正如一句中国谚语所说，“授人以鱼，不如授人以渔”。

本书先前的 8 个版本已经很成功地作为教科书广为使用，用户包括从高中生到研究生各个教育层次众多专业的学生。本版仍将以此为目标。

第 9 版更新

第 9 版最突出的变化是第 4 章（组网及因特网）、第 7 章（软件工程）和第 10 章（人工智能）。尽管这几章目录变化很小，但是内容已经扩充、更新而且重新排序。其中，第 4 章涉及组网基础、XML、HTML、安全的内容都有扩充；第 7 章改动较大，并包含了对 UML 的更严谨的介绍；第 10 章进行了较大范围的重写。

其他章节的变化有：第 1 章，加入闪存设备一节，重写数据压缩一节，用 LZW 压缩代替了 LZ77 压缩，加入压缩音频和视频的内容；第 2 章，加入 USB 和 FireWire 的材料；第 3 章，重写概述以及安全性小节；第 6 章，删除链接和装载的部分。另外，本书还有许多小的变化，是为了让所讨论的主题更加清晰明了、与时俱进、准确切题。

章节安排

本书内容遵循主题自底向上的排列，即由具体到抽象地推进——这是一种很利于教学的顺序，每一个主题自然而然地引导出下一个主题。本书首先介绍的是信息编码及计算机体系结构的基本原理（第 1 章和第 2 章）；进而是操作系统（第 3 章）和计算机网络（第 4 章）的学习，接着探讨了算法、程序设计语言及软件开发（第 5 章至第 7 章），然后探索数据结构和数据库（第 8 章和第 9 章）方面的问题，接着考察了计算

机技术通过人工智能(第10章)在未来的应用,最后以计算理论导引(第11章)作为结束。

本书编排顺序自然连贯,但各个章节仍保持很强的独立性,可以单独查阅,或者根据不同学习顺序重新排列。事实上,本书已作为各类课程的教材,内容选择的顺序是多种多样的。其中一种教法是先介绍第5章和第6章(算法和程序设计语言),然后按照需要返回到前面章节。我还有人是从第11章有关可计算性的内容开始的。此外,本书还曾被用作“毕业班”的教材,用作学生进入其他领域学习的主干课程。

在目录中,本书已经用星号标识出了选学章节。其中有些章节讨论更专门的话题,有些是对传统内容的深入探究。此举仅仅是为那些想采取不同阅读顺序的人提供建议。当然,还有其他读法。尤其对于那些寻求快速阅读的读者,我建议采取下面的阅读顺序:

章 节	标 题	章 节	标 题
1.1~1.4	数据编码和存储基础	7.1~7.2	软件工程
2.1~2.3	计算机体系结构和机器语言	8.1~8.2	数据抽象
3.1~3.3	操作系统	9.1~9.2	数据库系统
4.1~4.3	组网及因特网	10.1~10.3	人工智能
5.1~5.4	算法和算法设计	11.1~11.2	计算理论
6.1~6.4	程序设计语言		

在本书中有几条贯穿始终的主线。主线之一是计算机科学是不断发展变化的。本书从历史发展的角度反复呈现各个主题,讨论其当前的状况,并指出研究方向。另一条主线是抽象的作用以及用抽象工具控制复杂性的方式。该主线在第0章引入,然后在操作系统体系结构、算法开发、程序设计语言、软件工程、数据表示和数据库系统等内容中反复体现。

致教师

本教材所包含的内容要多于通常情况一个学期所能讲授的,因此一定要果断地略掉不适合你教学需要的那些主题,或者根据需要重新调整讲授顺序。你会发现,尽管本书有它固有的结构体系,但各个主题在很大程度上又是相对独立的,可以根据需要做出选择。我写本书的目的是把它作为一种课程的资源,而非课程的定义。本人喜欢把某些主题留作阅读作业,鼓励学生自己学习,而不在课堂讲授。我认为,如果认为所有的东西都一定要在课堂上讲,那就低估学生的能力了。我们应该教会他们独立学习。

关于本书自底向上、从具体到抽象的组织结构,我觉得有必要多言几句。作为学者,我们总以为学生会欣赏我们对于学科的观点,这些观点是我们在某一领域多年工作中形成的。但作为老师,我认为我们最好从学生的视角提供教材。这就是为什么本书首先介绍的是数据的表示/存储、计算机体系结构、操作系统以及组网,因为这些都是学生们最容易产生共鸣的主题——他们大多都听说过 JPEG、MP3 这些术语,用 CD 和 DVD 刻录过资料,买过计算机配件,应用过某一操作系统,或者上过因特网。我发现,从这些主题开始讲授这门课程,我的学生找到了许多困惑他们多年的问题的答案,而且开始把这门课看作是实践课程而不是纯理论的课程。由此出发就会很自然地过渡到较抽象的内容上,例如算法、算法结构、程序设计语言、软件开发方法、可计算性以及复杂性等。而这些内容就是我们这些从事该领域的人所认为的计算机科学的主要内容。正如我前面所说的,并不是强求大家都按此顺序讲课,只是我鼓励你们如此尝试一下而已。

我们都知道,学生能学到的东西要远远多于我们直接传授的,而且潜移默化地传授要更容易吸收。当

要“传授”问题求解时，这就更是如此。学生不可能通过学习问题求解的方法而变成问题的解决者。他们只有通过解决问题——还不仅仅是那些精心设计过的“教科书式的问题”才能成为问题的解决者。因此我在本书中加入了大量的问题，其中有一些问题是有意模棱两可的——意味着正确答案不只一个。我建议你们采用并充分拓展这些问题。

我要放在“潜移默化学习”这一类主题中谈论的还有职业道德、伦理和社会责任感。我认为这种内容不可能独立成章，而是应该在有所涉及时讨论，这是本书的编排方法。你们会发现，在 3.5 节、4.5 节、7.8 节、9.7 节和 10.7 节分别在操作系统、组网、软件工程、数据库系统和人工智能的上下文中提及了安全、隐私、责任和社会意识的问题。而且，0.6 节就通过总结一些比较著名的理论而引入这一主线——这些理论都企图把伦理上的决断建立在哲学的坚实基础上。同时你还会发现，每一章都包含了“社会问题”小节，这些问题将鼓励学生思考现实社会与教材内容的关系。

感谢你对本书感兴趣。无论你是否选用本书作为教材，我都希望你喜欢它，认同它是一部不错的计算机科学教育文献。

教学特色

本书是多年教学经验的结晶，因此在教学法方面考虑较多。最主要的是提供了丰富的问题以加强学生的参与性——在第 9 版里包含 1000 多个问题。分为“问题与练习”、“复习题”和“社会问题”。“问题与练习”列在每节末尾（除了第 0 章），用于复习、延伸刚刚讨论过的内容，或者提示以后会涉及的有关主题。这些问题的答案在附录 F 中。

“复习题”列在每章的末尾（除了第 0 章没有以外）。它们是课后作业，内容覆盖整章，在书中不给出答案。

“社会问题”也列在每章的末尾，供思考讨论。许多可以用来开展课外研究，可要求学生提交简短的书面或口头报告。

在每章的末尾还设有“课外阅读”，它列出了与本章主题有关的参考资料。同时，前言以及正文中所列的网址也非常适合查找相关资料。

补充材料

本书的许多补充材料可以从配套网站 www.aw.com/brookshear 上找到。以下内容面向所有读者。

- 每章活动帮助加深理解本教材的主题，并提供机会了解其他相关主题。
- 第2章使用的样机的软件模拟程序。
- 每章的“自测题”帮助读者复习本书中的内容。

除此之外，教师还可以登录 Addison-Wesley 的教师资源中心 (www.aw.com/ric) 申请获得下面的教辅资料：

- 包含“复习题”答案的教师指导。
- PowerPoint 幻灯片讲稿。
- 测试题库。

你也许还想看一下我的个人网站 www.msccs.mu.edu/~glennb，不是很正式（体现了我某一时的灵感和幽默），但你或许能找到些有用的信息。

致学生

我有一点点偏执 (我的一些朋友说远不是一点点), 所以写本书时, 我经常不接受他人的建议, 其中许多人认为一些内容对于初学者过于高深。我相信即使学术界把它们归为“高级论题”, 但只要与主题相关就是合适的。读者需要的是一本全面介绍计算机科学的教科书, 而不是“缩了水”的版本——只包括那些被简化了的、被认为是适合初学者的主题。因此我不回避任何主题, 相反, 我还力求寻找更好的解释。我力图在一定深度上向读者展示计算机科学最真实的一面。就好比对待菜谱里的那些调味品一样, 你可以有选择地略过本书的一些主题, 但我写这些主题是为了在你想要的时候供你“品尝”, 而且我也鼓励你们去尝试。

我还要指出的是, 在任何与技术打交道的课程中, 目前学到的细节可能是以后不需要知道的。这个领域是发展变化的——这是使人兴奋的方面。本书将从现实及历史的角度展现本学科的图景。有了这些背景知识, 你们就会和技术一起成长。我希望你们现在就开始行动起来, 不局限于课本的内容进行探索。要学会学习。

感谢你们对我的信任, 选择了我的这本书。作为作者, 我有责任创作出值得一读的书稿。我希望你们看到我已经尽到了这份责任。

致谢

首先我要感谢那些支持本书——阅读并使用本书前几个版本的人们, 我感到很荣幸。

随着每一次新版本的问世, 给本书提出建议的审稿人和顾问也越来越多。如今, 这份名单包括 J. M. Adams、C.M.Allen、D.C.S.Allison、R.Ashmore、B.Auernheimer、P.Bankston、M.Barnard、P.Bender、K.Bowyer、P.W.Brashear、C.M.Brown、B.Calloni、M.Clancy、R.T.Close、D.H.Cooley、L.D.Cornell、M.J.Crowley、F.Deek、M.Dickerson、M.J.Duncan、S.Fox、N.E.Gibbs、J.D.Harris、D.Hascom、L.Heath、P.B.Henderson、L.Hunt、M.Hutchenreuther、L.A.Jehn、K.K.Kolberg、K.Korb、G.Krenz、J.Liu、T.J.Long、C.May、W.McCown、S.J.Merrill、K.Messersmith、J.C.Moyer、M.Murphy、J.P.Myers, Jr.、D.S.Noonan、S.Olariu、G.Rice、N.Rickert、C.Riedesel、J.B.Rogers、G.Saito、W.Savitch、R.Schlaflly、J.C.Schlimmer、S.Sells、G.Sheppard、Z.Shen、J.C.Simms、M.C.Slaterry、J.Slimick、J.A.Slomka、D.Smith、J.Solderitsch、R.Steigerwald、L.Steinberg、C.A.Struble、C.L.Struble、W.J.Taffe、J.Talbur、P.Tonellato、P.Tromovitch、E.D.Winter、E.Wright、M.Ziegler, 还有一位匿名的朋友。我向他们的每一位致以我最真诚的谢意。

尤其要感谢 Roger Eastman, 他对第 10 章 (人工智能) 的修订起了很大的作用。我想你们会发现他的见解极大地改善了主题的陈述。同时他也为本书网站提供了许多辅助性资料, 我非常感激他为此所做出的努力。

我同时要感谢为本项目做出贡献的 Addison-Wesley 的员工。他们不仅是很好的合作伙伴, 而且还是很好的朋友。如果你们打算写一本教材, 可以考虑交给 Addison-Wesley 出版。

我还要感谢我的夫人 Earlene 和我的女儿 Cheryl, 感谢她们这么多年对我的鼓励。当然 Cheryl 已经长大, 几年以前已经离家开始独自生活。Earlene 还陪在我身边。我是一个幸运的人。1998 年 12 月 11 日的早晨, 我突发心脏病, 由于她及时把我送到了医院, 我逃过了一劫。(对于年轻一代的你们, 我有必要解释一下, 躲过心脏病的一劫有点像你们又获准延期提交课后作业。)

最后, 我要感谢我的父母, 本书即是给他的献礼。我用下面一句赞语作为结束, 就不要署名了: “我们儿子的书真的非常好, 人人都应该阅读。”



Chapter 0 Introduction 1

- 0.1 The Role of Algorithms 2
- 0.2 The Origins of Computing Machines 4
- 0.3 The Science of Algorithms 9
- 0.4 Abstraction 10
- 0.5 An Outline of Our Study 11
- 0.6 Social Repercussions 13

Chapter 1 Data Storage 19

- 1.1 Bits and Their Storage 20
- 1.2 Main Memory 27
- 1.3 Mass Storage 30
- 1.4 Representing Information as Bit Patterns 37
- *1.5 The Binary System 44
- *1.6 Storing Integers 49
- *1.7 Storing Fractions 56
- *1.8 Data Compression 61
- *1.9 Communication Errors 66

Chapter 2 Data Manipulation 79

- 2.1 Computer Architecture 80
- 2.2 Machine Language 83
- 2.3 Program Execution 89
- *2.4 Arithmetic/Logic Instructions 97
- *2.5 Communicating with Other Devices 102
- *2.6 Other Architectures 107

Chapter 3 Operating Systems 119

- 3.1 The History of Operating Systems 120
- 3.2 Operating System Architecture 124
- 3.3 Coordinating the Machine's Activities 131
- *3.4 Handling Competition Among Processes 134
- 3.5 Security 139

**Asterisks indicate suggestions for optional sections.*

Chapter 4 Networking and the Internet 147

- 4.1 Network Fundamentals 148
- 4.2 The Internet 157
- 4.3 The World Wide Web 164
- *4.4 Internet Protocols 174
- 4.5 Security 180

Chapter 5 Algorithms 195

- 5.1 The Concept of an Algorithm 196
- 5.2 Algorithm Representation 199
- 5.3 Algorithm Discovery 207
- 5.4 Iterative Structures 213
- 5.5 Recursive Structures 224
- 5.6 Efficiency and Correctness 233

Chapter 6 Programming Languages 251

- 6.1 Historical Perspective 252
- 6.2 Traditional Programming Concepts 261
- 6.3 Procedural Units 272
- 6.4 Language Implementation 280
- *6.5 Object-Oriented Programming 289
- *6.6 Programming Concurrent Activities 295
- *6.7 Declarative Programming 298

Chapter 7 Software Engineering 311

- 7.1 The Software Engineering Discipline 312
- 7.2 The Software Life Cycle 315
- 7.3 Software Engineering Methodologies 319
- 7.4 Modularity 321
- 7.5 Tools of the Trade 328
- 7.6 Testing 336
- 7.7 Documentation 338
- 7.8 Software Ownership and Liability 339

Chapter 8 Data Abstractions 349

- 8.1 Data Structure Fundamentals 350
- 8.2 Implementing Data Structures 355
- 8.3 A Short Case Study 370
- 8.4 Customized Data Types 375
- *8.5 Classes and Objects 379
- *8.6 Pointers in Machine Language 381

Chapter 9 Database Systems 391

- 9.1 Database Fundamentals 392
- 9.2 The Relational Model 397

- *9.3 Object-Oriented Databases 408
- *9.4 Maintaining Database Integrity 411
- *9.5 Traditional File Structures 414
- 9.6 Data Mining 423
- 9.7 Social Impact of Database Technology 425

Chapter 10 Artificial Intelligence 435

- 10.1 Intelligence and Machines 436
- 10.2 Perception 441
- 10.3 Reasoning 447
- 10.4 Additional Areas of Research 460
- 10.5 Artificial Neural Networks 464
- 10.6 Robotics 473
- 10.7 Considering the Consequences 475

Chapter 11 Theory of Computation 485

- 11.1 Functions and Their Computation 486
- 11.2 Turing Machines 488
- 11.3 Universal Programming Languages 493
- 11.4 A Noncomputable Function 499
- 11.5 Complexity of Problems 504
- *11.6 Public-Key Cryptography 513

Appendixes 525

- A ASCII 527
- B Circuits to Manipulate Two's Complement Representations 529
- C A Simple Machine Language 533
- D High-Level Language Program Examples 535
- E The Equivalence of Iterative and Recursive Structures 543
- F Answers to Questions & Exercises 545

Index 585

Introduction

In this preliminary chapter we consider the scope of computer science, develop a historical perspective, and establish a foundation from which to launch our study.

0.1 The Role of Algorithms

0.2 The Origins of Computing Machines

0.3 The Science of Algorithms

0.4 Abstraction

0.5 An Outline of Our Study

0.6 Social Repercussions

Computer science is the discipline that seeks to build a scientific foundation for such topics as computer design, computer programming, information processing, algorithmic solutions of problems, and the algorithmic process itself. It provides the underpinnings for today's computer applications as well as the foundations for tomorrow's applications.

This book provides a comprehensive introduction to this science. We will investigate a wide range of topics including most of those that constitute a typical university computer science curriculum. We want to appreciate the full scope and dynamics of the field. Thus, in addition to the topics themselves, we will be interested in their historical development, the current state of research, and prospects for the future. Our goal is to establish a functional understanding of computer science—one that will support those who wish to pursue more specialized studies in the science as well as one that will enable those in other fields to flourish in an increasingly technical society.

0.1 The Role of Algorithms

We begin with the most fundamental concept of computer science—that of an algorithm. Informally, an **algorithm** is a set of steps that defines how a task is performed. (We will be more precise later in Chapter 5.) For example, there are algorithms for cooking (called recipes), for finding your way through a strange city (more commonly called directions), for operating washing machines (usually displayed on the inside of the washer's lid or perhaps on the wall of a laundromat), for playing music (expressed in the form of sheet music), and for performing magic tricks (Figure 0.1).

Before a machine such as a computer can perform a task, an algorithm for performing that task must be discovered and represented in a form that is compatible with the machine. A representation of an algorithm is called a **program**. For the convenience of humans, computer programs are usually printed on paper or displayed on computer screens. For the convenience of machines, programs are encoded in a manner compatible with the technology of the machine. The process of developing a program, encoding it in machine-compatible form, and inserting it into a machine is called **programming**. Programs, and the algorithms they represent, are collectively referred to as **software**, in contrast to the machinery itself, which is known as **hardware**.

The study of algorithms began as a subject in mathematics. Indeed, the search for algorithms was a significant activity of mathematicians long before the development of today's computers. The goal was to find a single set of directions that described how all problems of a particular type could be solved. One of the best known examples of this early research is the long division algorithm for finding the quotient of two multiple-digit numbers. Another example is the Euclidean algorithm, discovered by the ancient Greek mathematician Euclid, for finding the greatest common divisor of two positive integers (Figure 0.2).

Once an algorithm for performing a task has been found, the performance of that task no longer requires an understanding of the principles on which the algorithm is based. Instead, the performance of the task is reduced to the process of merely following directions. (We can follow the long division algorithm to find a quotient or the Euclidean algorithm to find a greatest common divisor without understanding why the

Figure 0.1 An algorithm for a magic trick

Effect: The performer places some cards from a normal deck of playing cards face down on a table and mixes them thoroughly while spreading them out on the table. Then, as the audience requests either red or black cards, the performer turns over cards of the requested color.

Secret and Patter:

- Step 1. From a normal deck of cards, select ten red cards and ten black cards. Deal these cards face up in two piles on the table according to color.
- Step 2. Announce that you have selected some red cards and some black cards.
- Step 3. Pick up the red cards. Under the pretense of aligning them into a small deck, hold them face down in your left hand and, with the thumb and first finger of your right hand, pull back on each end of the deck so that each card is given a slightly *backward* curve. Then place the deck of red cards face down on the table as you say, "Here are the red cards in this stack."
- Step 4. Pick up the black cards. In a manner similar to that in step 3, give these cards a slight *forward* curve. Then return these cards to the table in a face-down deck as you say, "And here are the black cards in this stack."
- Step 5. Immediately after returning the black cards to the table, use both hands to mix the red and black cards (still face down) as you spread them out on the tabletop. Explain that you are thoroughly mixing the cards.
- Step 6. As long as there are face-down cards on the table, repeatedly execute the following steps:
 - 6.1. Ask the audience to request either a red or a black card.
 - 6.2. If the color requested is red and there is a face-down card with a concave appearance, turn over such a card while saying, "Here is a red card."
 - 6.3. If the color requested is black and there is a face-down card with a convex appearance, turn over such a card while saying, "Here is a black card."
 - 6.4. Otherwise, state that there are no more cards of the requested color and turn over the remaining cards to prove your claim.

Figure 0.2 The Euclidean algorithm for finding the greatest common divisor of two positive integers

Description: This algorithm assumes that its input consists of two positive integers and proceeds to compute the greatest common divisor of these two values.

Procedure:

Step 1. Assign M and N the value of the larger and smaller of the two input values, respectively.

Step 2. Divide M by N, and call the remainder R.

Step 3. If R is not 0, then assign M the value of N, assign N the value of R, and return to step 2; otherwise, the greatest common divisor is the value currently assigned to N.

algorithm works.) In a sense, the intelligence required to solve the problem at hand is encoded in the algorithm.

It is through this ability to capture and convey intelligence (or at least intelligent behavior) by means of algorithms that we are able to build machines that perform useful tasks. Consequently, the level of intelligence displayed by machines is limited by the intelligence that can be conveyed through algorithms. We can construct a machine to perform a task only if an algorithm exists for performing that task. In turn, if no algorithm exists for solving a problem, then the solution of that problem lies beyond the capabilities of machines.

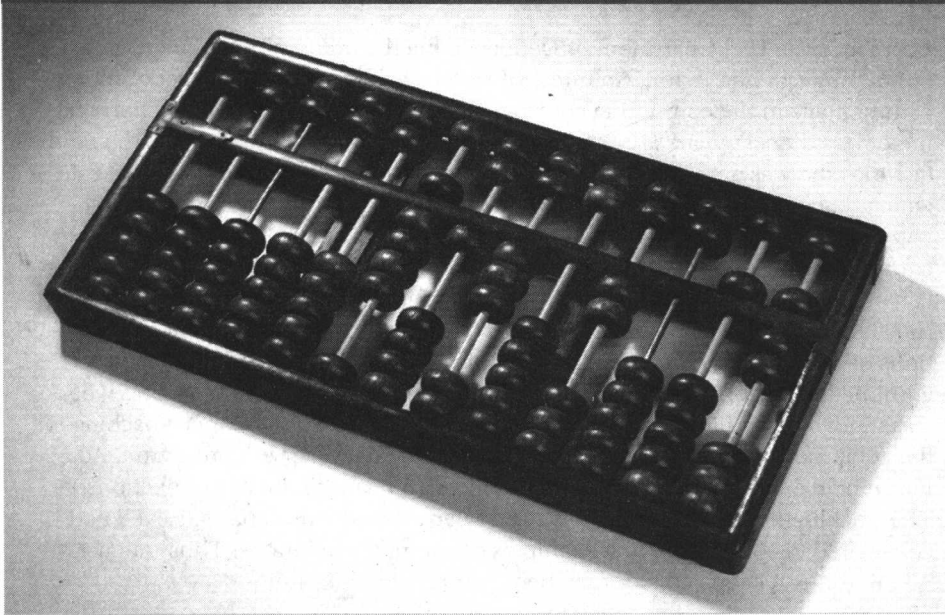
Identifying the limitations of algorithmic capabilities solidified as a subject in mathematics in the 1930s with the publication of Kurt Gödel's incompleteness theorem. This theorem essentially states that in any mathematical theory encompassing our traditional arithmetic system, there are statements whose truth or falseness cannot be established by algorithmic means. In short, any complete study of our arithmetic system lies beyond the capabilities of algorithmic activities.

This realization shook the foundations of mathematics, and the study of algorithmic capabilities that ensued was the beginning of the field known today as computer science. Indeed, it is the study of algorithms that forms the core of computer science.

0.2 The Origins of Computing Machines

Today's computers have an extensive genealogy. One of the earlier computing devices was the abacus. Its history has been traced as far back as the ancient Greek and Roman civilizations. The machine is quite simple, consisting of beads strung on rods that are in turn mounted in a rectangular frame (Figure 0.3). As the beads are moved back and

Figure 0.3 An abacus (photography by Wayne Chandler)



forth on the rods, their positions represent stored values. It is in the positions of the beads that this “computer” represents and stores data. For control of an algorithm’s execution, the machine relies on the human operator. Thus the abacus alone is merely a data storage system; it must be combined with a human to create a complete computational machine.

In more recent years, the design of computing machines was based on the technology of gears. Among the inventors were Blaise Pascal (1623–1662) of France, Gottfried Wilhelm Leibniz (1646–1716) of Germany, and Charles Babbage (1792–1871) of England. These machines represented data through gear positioning, with data being input mechanically by establishing initial gear positions. Output from Pascal’s and Leibniz’s machines was achieved by observing the final gear positions. Babbage, on the other hand, envisioned machines that would print results of computations on paper so that the possibility of transcription errors would be eliminated.

As for the ability to follow an algorithm, we can see a progression of flexibility in these machines. Pascal’s machine was built to perform only addition. Consequently, the appropriate sequence of steps was embedded into the structure of the machine itself. In a similar manner, Leibniz’s machine had its algorithms firmly embedded in its architecture, although it offered a variety of arithmetic operations from which the operator could select. Babbage’s Difference Engine (of which only a demonstration model was constructed) could be modified to perform a variety of calculations, but his Analytical Engine (the construction for which he never received funding) was designed to read instructions in the form of holes in paper cards. Thus Babbage’s Analytical Engine