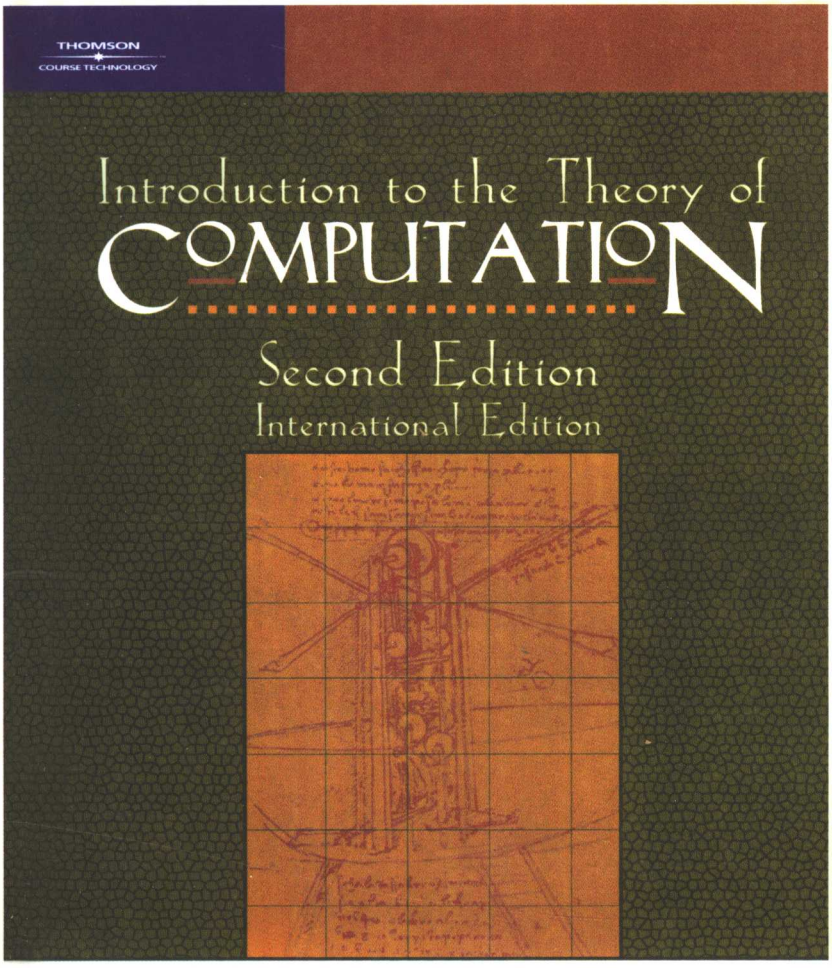
HZ BOOKS
THOMSON
经 典 原 版 书 库

计算理论导引

新版

（英文版 · 第2版）



The content of this text differs from the U.S. version

(美) Michael Sipser 著
麻省理工学院

机械工业出版社
China Machine Press

经 典 原 版 书 库

计算理论导引

（英文版 · 第2版）

Introduction to the Theory of Computation

(Second Edition)

(美) Michael Sipser
麻省理工学院 著

机械工业出版社
China Machine Press

Michael Sipser: Introduction to the Theory of Computation, Second Edition (ISBN 0-619-21764-2).

981-265-459-3

本书版权登记号：图字：01-2005-4496

图书在版编目（CIP）数据

计算理论导引（英文版·第2版）/（美）塞普瑟（Sipser, M.）著；　—北京：机械工业出版社，2006.1

（经典原版书库）

书名原文：Introduction to the Theory of Computation, Second Edition

ISBN 7-111-17327-9

Ⅰ.计…　Ⅱ.塞…　Ⅲ.电子计算机—算法理论—英文　Ⅳ.TP301.6

中国版本图书馆CIP数据核字（2005）第102232号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

北京京北制版厂印刷·新华书店北京发行所发行

2006年1月第1版第1次印刷

718mm×1020mm 1/16·28.75印张

印数：0 001- 3 000册

定价：49.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

PREFACE TO THE FIRST EDITION

TO THE STUDENT

Welcome!

You are about to embark on the study of a fascinating and important subject: the theory of computation. It comprises the fundamental mathematical properties of computer hardware, software, and certain applications thereof. In studying this subject we seek to determine what can and cannot be computed, how quickly, with how much memory, and on which type of computational model. The subject has obvious connections with engineering practice, and, as in many sciences, it also has purely philosophical aspects.

I know that many of you are looking forward to studying this material but some may not be here out of choice. You may want to obtain a degree in computer science or engineering, and a course in theory is required—God knows why. After all, isn't theory arcane, boring, and worst of all, irrelevant?

To see that theory is neither arcane nor boring, but instead quite understandable and even interesting, read on. Theoretical computer science does have many fascinating big ideas, but it also has many small and sometimes dull details that can be tiresome. Learning any new subject is hard work, but it becomes easier and more enjoyable if the subject is properly presented. My primary objective in writing this book is to expose you to the genuinely exciting aspects of computer theory, without getting bogged down in the drudgery. Of course, the only way to determine whether theory interests you is to try learning it.

Theory is relevant to practice. It provides conceptual tools that practitioners use in computer engineering. Designing a new programming language for a specialized application? What you learned about *grammars* in this course comes in handy. Dealing with string searching and pattern matching? Remember *finite automata* and *regular expressions*. Confronted with a problem that seems to require more computer time than you can afford? Think back to what you learned about *NP-completeness*. Various application areas, such as modern cryptographic protocols, rely on theoretical principles that you will learn here.

Theory also is relevant to you because it shows you a new, simpler, and more elegant side of computers, which we normally consider to be complicated machines. The best computer designs and applications are conceived with elegance in mind. A theoretical course can heighten your aesthetic sense and help you build more beautiful systems.

Finally, theory is good for you because studying it expands your mind. Computer technology changes quickly. Specific technical knowledge, though useful today, becomes outdated in just a few years. Consider instead the abilities to think, to express yourself clearly and precisely, to solve problems, and to know when you haven't solved a problem. These abilities have lasting value. Studying theory trains you in these areas.

Practical considerations aside, nearly everyone working with computers is curious about these amazing creations, their capabilities, and their limitations. A whole new branch of mathematics has grown up in the past 30 years to answer certain basic questions. Here's a big one that remains unsolved: If I give you a large number, say, with 500 digits, can you find its factors (the numbers that divide it evenly), in a reasonable amount of time? Even using a supercomputer, no one presently knows how to do that in all cases *within the lifetime of the universe!* The factoring problem is connected to certain secret codes in modern cryptosystems. Find a fast way to factor and fame is yours!

TO THE EDUCATOR

This book is intended as an upper-level undergraduate or introductory graduate text in computer science theory. It contains a mathematical treatment of the subject, designed around theorems and proofs. I have made some effort to accommodate students with little prior experience in proving theorems, though more experienced students will have an easier time.

My primary goal in presenting the material has been to make it clear and interesting. In so doing, I have emphasized intuition and "the big picture" in the subject over some lower level details.

For example, even though I present the method of proof by induction in Chapter 0 along with other mathematical preliminaries, it doesn't play an important role subsequently. Generally I do not present the usual induction proofs of the correctness of various constructions concerning automata. If presented clearly, these constructions convince and do not need further argument. An induction may confuse rather than enlighten because induction itself is a rather sophisticated technique that many find mysterious. Belaboring the obvious with

an induction risks teaching students that mathematical proof is a formal manipulation instead of teaching them what is and what is not a cogent argument.

A second example occurs in Parts Two and Three, where I describe algorithms in prose instead of pseudocode. I don't spend much time programming Turing machines (or any other formal model). Students today come with a programming background and find the Church–Turing thesis to be self-evident. Hence I don't present lengthy simulations of one model by another to establish their equivalence.

Besides giving extra intuition and suppressing some details, I give what might be called a classical presentation of the subject material. Most theorists will find the choice of material, terminology, and order of presentation consistent with that of other widely used textbooks. I have introduced original terminology in only a few places, when I found the standard terminology particularly obscure or confusing. For example I introduce the term *mapping reducibility* instead of *many–one reducibility*.

Practice through solving problems is essential to learning any mathematical subject. In this book, the problems are organized into two main categories called *Exercises* and *Problems*. The Exercises review definitions and concepts. The Problems require some ingenuity. Problems marked with a star are more difficult. I have tried to make both the Exercises and Problems interesting challenges.

THE FIRST EDITION

Introduction to the Theory of Computation first appeared as a Preliminary Edition in paperback. The first edition differs from the Preliminary Edition in several substantial ways. The final three chapters are new: Chapter 8 on space complexity; Chapter 9 on provable intractability; and Chapter 10 on advanced topics in complexity theory. Chapter 6 was expanded to include several advanced topics in computability theory. Other chapters were improved through the inclusion of additional examples and exercises.

Comments from instructors and students who used the Preliminary Edition were helpful in polishing Chapters 0–7. Of course, the errors they reported have been corrected in this edition.

Chapters 6 and 10 give a survey of several more advanced topics in computability and complexity theories. They are not intended to comprise a cohesive unit in the way that the remaining chapters are. These chapters are included to allow the instructor to select optional topics that may be of interest to the serious student. The topics themselves range widely. Some, such as *Turing reducibility* and *alternation*, are direct extensions of other concepts in the book. Others, such as *decidable logical theories* and *cryptography*, are brief introductions to large fields.

FEEDBACK TO THE AUTHOR

The internet provides new opportunities for interaction between authors and readers. I have received much e-mail offering suggestions, praise, and criticism,

and reporting errors for the Preliminary Edition. Please continue to correspond! I try to respond to each message personally, as time permits. The e-mail address for correspondence related to this book is

sipserbook@math.mit.edu.

A web site that contains a list of errata is maintained. Other material may be added to that site to assist instructors and students. Let me know what you would like to see there. The location for that site is

http://www-math.mit.edu/~sipser/book.html.

ACKNOWLEDGMENTS

I could not have written this book without the help of many friends, colleagues, and my family.

I wish to thank the teachers who helped shape my scientific viewpoint and educational style. Five of them stand out. My thesis advisor, Manuel Blum, is due a special note for his unique way of inspiring students through clarity of thought, enthusiasm, and caring. He is a model for me and for many others. I am grateful to Richard Karp for introducing me to complexity theory, to John Addison for teaching me logic and assigning those wonderful homework sets, to Juris Hartmanis for introducing me to the theory of computation, and to my father for introducing me to mathematics, computers, and the art of teaching.

This book grew out of notes from a course that I have taught at MIT for the past 15 years. Students in my classes took these notes from my lectures. I hope they will forgive me for not listing them all. My teaching assistants over the years, Avrim Blum, Thang Bui, Andrew Chou, Benny Chor, Stavros Cosmadakis, Aditi Dhagat, Wayne Goddard, Parry Husbands, Dina Kravets, Jakov Kučan, Brian O'Neill, Ioana Popescu, and Alex Russell, helped me to edit and expand these notes and provided some of the homework problems.

Nearly three years ago, Tom Leighton persuaded me to write a textbook on the theory of computation. I had been thinking of doing so for some time, but it took Tom's persuasion to turn theory into practice. I appreciate his generous advice on book writing and on many other things.

I wish to thank Eric Bach, Peter Beebee, Cris Calude, Marek Chrobak, Anna Chefter, Guang-Ien Cheng, Elias Dahlhaus, Michael Fischer, Steve Fisk, Lance Fortnow, Henry J. Friedman, Jack Fu, Seymour Ginsburg, Oded Goldreich, Brian Grossman, David Harel, Micha Hofri, Dung T. Huynh, Neil Jones, H. Chad Lane, Kevin Lin, Michael Loui, Silvio Micali, Tadao Murata, Christos Papadimitriou, Vaughan Pratt, Daniel Rosenband, Brian Scassellati, Ashish Sharma, Nir Shavit, Alexander Shen, Ilya Shlyakhter, Matt Stallmann, Perry Susskind, Y. C. Tay, Joseph Traub, Osamu Watanabe, Peter Widmayer, David Williamson, Derick Wood, and Charles Yang for comments, suggestions, and assistance as the writing progressed.

The following people provided additional comments that have improved this book: Isam M. Abdelhameed, Eric Allender, Shay Artzi, Michelle Ather-

ton, Rolfe Blodgett, Al Briggs, Brian E. Brooks, Jonathan Buss, Jin Yi Cai, Steve Chapel, David Chow, Michael Ehrlich, Yaakov Eisenberg, Farzan Fallah, Shaun Flisakowski, Hjalmtyr Hafsteinsson, C. R. Hale, Maurice Herlihy, Vegard Holmedahl, Sandy Irani, Kevin Jiang, Rhys Price Jones, James M. Jowdy, David M. Martin Jr., Manrique Mata-Montero, Ryota Matsuura, Thomas Minka, Farooq Mohammed, Tadao Murata, Jason Murray, Hideo Nagahashi, Kazuo Ohta, Constantine Papageorgiou, Joseph Raj, Rick Regan, Rhonda A. Reumann, Michael Rintzler, Arnold L. Rosenberg, Larry Roske, Max Rozenoer, Walter L. Ruzzo, Sanatan Sahgal, Leonard Schulman, Steve Seiden, Joel Seiferas, Ambuj Singh, David J. Stucki, Jayram S. Thathachar, H. Venkateswaran, Tom Whaley, Christopher Van Wyk, Kyle Young, and Kyoung Hwan Yun.

Robert Sloan used an early version of the manuscript for this book in a class that he taught and provided me with invaluable commentary and ideas from his experience with it. Mark Herschberg, Kazuo Ohta, and Latanya Sweeney read over parts of the manuscript and suggested extensive improvements. Shafi Goldwasser helped me with material in Chapter 10.

I received expert technical support from William Baxter at Superscript, who wrote the LaTeX macro package implementing the interior design, and from Larry Nolan at the MIT mathematics department, who keeps everything running.

It has been a pleasure to work with the folks at PWS Publishing in creating the final product. I mention Michael Sugarman, David Dietz, Elise Kaiser, Monique Calello, Susan Garland and Tanja Brull because I have had the most contact with them, but I know that many others have been involved, too. Thanks to Jerry Moore for the copy editing, to Diane Levy for the cover design, and to Catherine Hawkes for the interior design.

I am grateful to the National Science Foundation for support provided under grant CCR-9503322.

My father, Kenneth Sipser, and sister, Laura Sipser, converted the book diagrams into electronic form. My other sister, Karen Fisch, saved us in various computer emergencies, and my mother, Justine Sipser, helped out with motherly advice. I thank them for contributing under difficult circumstances, including insane deadlines and recalcitrant software.

Finally, my love goes to my wife, Ina, and my daughter, Rachel. Thanks for putting up with all of this.

Cambridge, Massachusetts
October, 1996

Michael Sipser

PREFACE TO THE SECOND EDITION (INTERNATIONAL)

Judging from the email communications that I've received from so many of you, the biggest deficiency of the first edition is that it provides no sample solutions to any of the problems. So here they are. Every chapter now contains a new *Selected Solutions* section that gives answers to a representative cross-section of that chapter's exercises and problems. To make up for the loss of the solved problems as interesting homework challenges, I've also added a variety of new problems. Instructors may request an Instructor's Manual that contains additional solutions by contacting the sales representative for their region designated at `www.course.com`.

This second edition (international) had been designed with certain non-domestic markets in mind. Though it covers the same topics, it differs from the standard second edition and not intended for use as a substitute for the standard second editon.

A number of readers would have liked more coverage of certain "standard" topics, particularly the Myhill–Nerode Theorem and Rice's Theorem. I've partially accommodated these readers by developing these topics in the solved problems. I did not include the Myhill–Nerode Theorem in the main body of the text because I believe that this course should provide only an introduction to finite automata and not a deep investigation. In my view, the role of finite automata here is for students to explore a simple formal model of computation as a prelude to more powerful models, and to provide convenient examples for subsequent topics. Of course, some people would prefer a more thorough treatment, while others feel that I ought to omit all reference to (or at least dependence on) finite

automata. I did not include Rice's Theorem in the main body of the text because, though it can be a useful "tool" for proving undecidability, some students might use it mechanically without really understanding what is going on. Using reductions instead, for proving undecidability, gives more valuable preparation for the reductions that appear in complexity theory.

I am indebted to my teaching assistants, Ilya Baran, Sergi Elizalde, Rui Fan, Jonathan Feldman, Venkatesan Guruswami, Prahladh Harsha, Christos Kapoutsis, Julia Khodor, Adam Klivans, Kevin Matulef, Ioana Popescu, April Rasala, Sofya Raskhodnikova, and Iuliu Vasilescu who helped me to craft some of the new problems and solutions. Ching Law, Edmond Kayi Lee, and Zulfikar Ramzan also contributed to the solutions. I thank Victor Shoup for coming up with a simple way to repair the gap in the analysis of the probabilistic primality algorithm that appears in the first edition.

I appreciate the efforts of the people at Course Technology in pushing me and the other parts of this project along, especially Alyssa Pratt and Aimee Poirier. Many thanks to Gerald Eisman, Weizhen Mao, Rupak Majumdar, Chris Umans, and Christopher Wilson for their reviews. I'm indebted to Jerry Moore for his superb job copy editing and to Laura Segel of ByteGraphics (lauras@bytegraphics.com) for her beautifully precise rendition of the figures.

The volume of email I've received has been more than I expected. Hearing from so many of you from so many places has been absolutely delightful, and I've tried to respond to all eventually—my apologies for those I missed. I've listed here the people who made suggestions that specifically affected this edition, but I thank everyone for their correspondence.

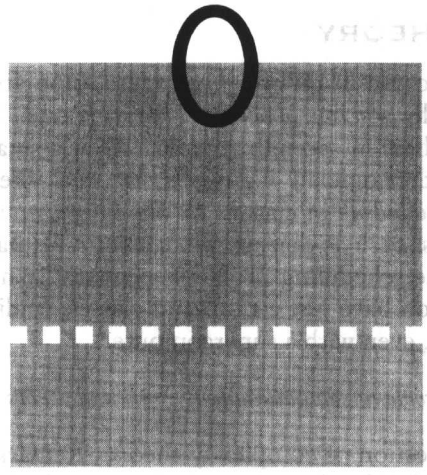
Luca Aceto, Arash Afkanpour, Rostom Aghanian, Eric Allender, Karun Bakshi, Brad Ballinger, Ray Bartkus, Louis Barton, Arnold Beckmann, Mihir Bellare, Kevin Trent Bergeson, Matthew Berman, Rajesh Bhatt, Somenath Biswas, Lenore Blum, Mauro A. Bonatti, Paul Bondin, Nicholas Bone, Ian Bratt, Gene Browder, Doug Burke, Sam Buss, Vladimir Bychkovsky, Bruce Carneal, Soma Chaudhuri, Rong-Jaye Chen, Samir Chopra, Benny Chor, John Clausen, Allison Coates, Anne Condon, Jeffrey Considine, John J. Crashell, Claude Crepeau, Shaun Cutts, Susheel M. Daswani, Geoff Davis, Scott Dexter, Peter Drake, Jeff Edmonds, Yaakov Eisenberg, Kurtcebe Eroglu, Georg Essl, Alexander T. Fader, Farzan Fallah, Faith Fich, Joseph E. Fitzgerald, Perry Fizzano, David Ford, Jeannie Fromer, Kevin Fu, Atsushi Fujioka, Michel Galley, K. Ganesan, Simson Garfinkel, Travis Gebhardt, Peymann Gohari, Ganesh Gopalakrishnan, Steven Greenberg, Larry Griffith, Jerry Grossman, Rudolf de Haan, Michael Halper, Nick Harvey, Mack Hendricks, Laurie Hiyakumoto, Steve Hockema, Michael Hoehle, Shahadat Hossain, Dave Isecke, Ghaith Issa, Raj D. Iyer, Christian Jacobi, Thomas Janzen, Mike D. Jones, Max Kanovitch, Aaron Kaufman, Roger Khazan, Sarfraz Khurshid, Kevin Killourhy, Seungjoo Kim, Victor Kuncak, Kanata Kuroda, Suk Y. Lee, Edward D. Legenski, Li-Wei Lehman, Kong Lei, Zsolt Lengvarszky, Jeffrey Levetin, Baekjun Lim, Karen Livescu, Thomas Lasko, Stephen Louie, TzerHung Low, Wolfgang Maass, Arash Madani, Michael Manapat, Wojciech Marchewka, David M. Martin Jr.,

Anders Martinson, Lyle McGeoch, Alberto Medina, Kurt Mehlhorn, Nihar Mehta, Albert R. Meyer, Thomas Minka, Mariya Minkova, Daichi Mizuguchi, G. Allen Morris III, Damon Mosk-Aoyama, Xiaolong Mou, Paul Muir, German Muller, Donald Nelson, Gabriel Nivasch, Mary Obelnicki, Kazuo Ohta, Thomas M. Oleson, Jr., Curtis Oliver, Owen Ozier, Rene Peralta, Alexander Perlis, Holger Petersen, Detlef Plump, Robert Prince, David Pritchard, Bina Reed, Nicholas Riley, Ronald Rivest, Robert Robinson, Christi Rockwell, Phil Rogaway, Max Rozenoer, John Rupf, Teodor Rus, Larry Ruzzo, Brian Sanders, Cem Say, Kim Schioett, Joel Seiferas, João Carlos Setubal, Geoff Lee Seyon, Mark Skandera, Bob Sloan, Geoff Smith, Marc L. Smith, Stephen Smith, Alex C. Snoeren, Guy St-Denis, Larry Stockmeyer, Radu Stoleru, David Stucki, Hisham M. Sueyllam, Kenneth Tam, Elizabeth Thompson, Michel Toulouse, Eric Tria, Chittaranjan Tripathy, Dan Trubow, Hiroki Ueda, Giora Unger, Kurt L. Van Etten, Jesir Vargas, Bienvenido Velez-Rivera, Kobus Vos, Alex Vrenios, Sven Waibel, Marc Waldman, Tom Whaley, Anthony Widjaja, Sean Williams, Joseph N. Wilson, Chris Van Wyk, Guangming Xing, Vee Voon Yee, Cheng Yongxi, Neal Young, Timothy Yuen, Kyle Yung, Jinghua Zhang, Lilla Zollei.

Most of all I thank my family—Ina, Rachel, and Aaron—for their patience, understanding, and love as I sat for endless hours here in front of my computer screen.

Cambridge, Massachusetts ***Michael Sipser***
December, 2004

CONTENTS



INTRODUCTION

We begin with an overview of those areas in the theory of computation that we present in this course. Following that, you'll have a chance to learn and/or review some mathematical concepts that you will need later.

0.1 AUTOMATA, COMPUTABILITY, AND COMPLEXITY

This book focuses on three traditionally central areas of the theory of computation: automata, computability, and complexity. They are linked by the question:

What are the fundamental capabilities and limitations of computers?

This question goes back to the 1930s when mathematical logicians first began to explore the meaning of computation. Technological advances since that time have greatly increased our ability to compute and have brought this question out of the realm of theory into the world of practical concern.

In each of the three areas—automata, computability, and complexity—this question is interpreted differently, and the answers vary according to the interpretation. Following this introductory chapter, we explore each area in a separate part of this book. Here, we introduce these parts in reverse order because starting from the end you can better understand the reason for the beginning.