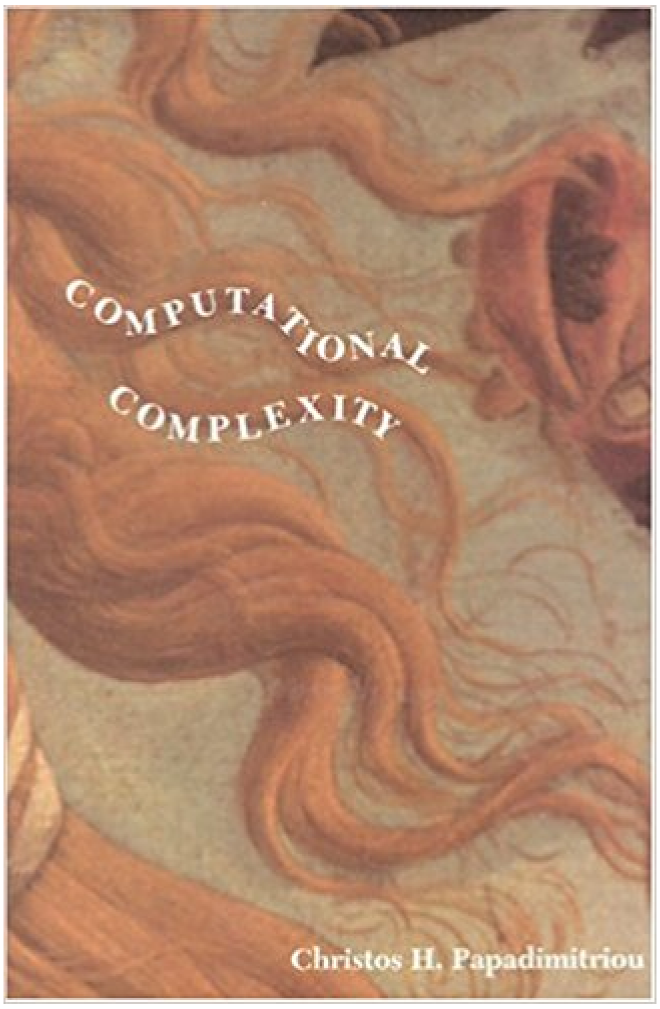
Computational Complexity



Synopsis

This modern introduction to the Theory of Computer Science is the first unified introduction to Computational Complexity. I+ offers a comprehensive and accessible treatment of the theory of algorithms and complexity - the elegant body of concepts and methods developed by computer scientists over the past 30 years for studying the pe@ormance and limitations of computer algorithms. The book is self-contained in that it develops all necessary mathematical prerequisites from such diverse fields such as computability, logic, number theory and probability.

Book Information

Paperback: 523 pages

Publisher: Pearson; 1 edition (December 10, 1993)

Language: English

ISBN-10: 0201530821

ISBN-13: 978-0201530827

Product Dimensions: 6.2 x 1.2 x 9 inches

Shipping Weight: 1.9 pounds (View shipping rates and policies)

Average Customer Review: 3.7 out of 5 stars   19 customer reviews

Best Sellers Rank: #195,461 in Books (See Top 100 in Books)   #31 inÂ Books > Computers & Technology > Hardware & DIY > Microprocessors & System Design > Computer Design   #97 inÂ Books > Science & Math > Mathematics > Pure Mathematics > Logic   #920 inÂ Books > Textbooks > Computer Science > Programming Languages

Customer Reviews

This new text offers a comprehensive and accessible treatment of the theory of algorithms and complexity - the elegant body of concepts and methods developed by computer scientists over the past 30 years for studying the performance and limitations of computer algorithms. Among topics covered are: reductions and NP-completeness, cryptography and protocols, randomized algorithms, and approximability of optimization problems, circuit complexity, the "structural" aspects of the P=NP question, parallel computation, the polynomial hierarchy, and many others. Several sophisticated and recent results are presented in a rather simple way, while many more are developed in the form of extensive notes, problems, and hints. The book is surprisingly self-contained, in that it develops all necessary mathematical prerequisites from such diverse field as computability, logic, number theory, combinatorics, and probability. Features First unified introduction to computational complexity. Integrates computation, applications, and logic throughout.

Provides an accessible introduction to logic, including Boolean logic, first-order logic, and second-order logic. Includes extensive exercises including historical notes, references, and challeging problems.   0201530821B04062001

explains the concepts well

This is a good introductory book of computational theory for students in computer science, good juniors, seniors and first year graduates. The book is well presented, fit for self studies, and covered most contents of computability and complexity. The book is slightly old, some of the latest result are not included, e.g., a P-algorithm of solving "prime problem" was found in 2001. This book is not good for advanced researchers in theoretical computer science, it is way to shallow. Compared with Martin Davis's book, this is easier to understand, equally well presented. Be sure not to get the $8-9 version, that is not the book, although under the same title.I am a research in theoretical algorithms.

This is a great book on complexity theory at a graduate school level.I would recommend this book to anyone interested in the field of complexity theory.

This book is excellent. However, you need strong training in the kind of reasoning used in math and CS theory before you can read it. The subject gets very abstract, and may be hard to follow (and that's not Papadimitriou's fault).I would recommend it for people who have already read Sipser's book (working on the exercises), for example.

[2016 review] Given the rapidity of technological development, the CS beginner may be surprised to learn that older textbooks contain much that is valuable: as with the rise of functional programming languages that derive their inspiration from Lisp, one of the first programming languages, there's a fair amount of "going back to the future". Christos Papadimitriou's *Computational Complexity* is a good illustration of this principle. Complexity theory is one of the most "scientific" parts of computer science, and with a $1 million prize (still) on offer for a solution to the P and NP problem there's no lack of attention to its basic concepts; recursive functions are well and good but if we want to do it on a computer, even a modern umpteen-core wonder, we need to be able to do it "fast". What budding computer scientists may lack is the logical background necessary to make sense of the complexity classes and what they mean for programming practice.Papadimitriou is professor of logic at UC-Berkeley and a gifted expositor of logical concepts: he recently wrote the text for *Logicomix*, a graphic novel about the founding of modern logic in the early 20th century. What you will find in this decades-old text is the logic you need to understand computer science and the computer science you need to understand logic: not only the dedicated CS student, but philosophers and other people with "theoretical" interests in computational paradigms will benefit immensely from reading this (most "theory of computation" texts are designed to slingshot students into compiler construction and reinforce the principles of algorithms, so they leave out the negative result of the *Entscheidungsproblem* for first-order logic, the reason modern computation exists in the first place). The layout of the different complexity classes, beginning with the "tractable" P and going on into the well-named "complexity zoo", is explained in readable prose and understandable diagrams.You need not be a genius to learn complexity from this book, only willing to be reflective about the principles of practices you know more "intuitively". Still relevant.

Yes,it is generally "hard" for undergradute students even grad. students. If you are taking course "Theory of computation", I would like to recommend the Sipser's or Cohen's books for reading supplement. But you should keep reading this book ! IMO, this book covers so many topics, that it becomes too dense to read. It means you should read it carefully and slowly. For example, it introduces the "reduction" in some previous chapters but without precise defintion and therefore misses the more important part :how to do the reduction correctly and what is the "reseasonable" reduction ? You will find the concept of "reduction" is not very easy to catch if you refer to the Sipser's or Ullman's books. Many friends and me could not go through more than 20 pages of this book in the beginning. But we were keeping on reading and surveying some "easy books". Finally, we understood most half parts of this book. Moreover, if some readers prepare to study more advanced and recent topics, this book is the must.

Computational complexity is not an easy subject; the mathematical maturity necessary is high, and it covers many subtopics ranging from graph theory to logic to number theory. Papadimitriou is more able than any other author I'm acquainted with at making this subject vaguely clear.Unfortunately, he's occasionally given to the standard flights of fancy that people of his subject are known for; it's a shame that most of them won't acknowledge that there's a big difference between an "efficient" algorithm and a reasonably effective one that actually gets the job done, for example. I wish that authors of complexity books would occasionally include a chapter or two by people who are actually designing real algorithms as a counterfoil.

We used this book for one semester when I was in the graduate school. This is one of the computer science related books that actually have enough substance to have some intellectual value.I found this volume entertaining years after leaving graduate school and working in the industry as an engineer. The topics addressed in this book is actually quite intriguing--the best time to reduce programming complexity is before one actually programs. I believe any serious programmer should be able to estimate the complexity, both space and time, on the algorithm he is designing.In the real world, one does not encounter nontrivial algorithms very often, and from a practical perspective, this books is not quite useful.However, when you really get bored, this is something that could entertain your brain a little.

Download to continue reading...

FAQ & Help