Intro to Analysis of Algorithms
Welcome
Chapter 0

Michael Soltys

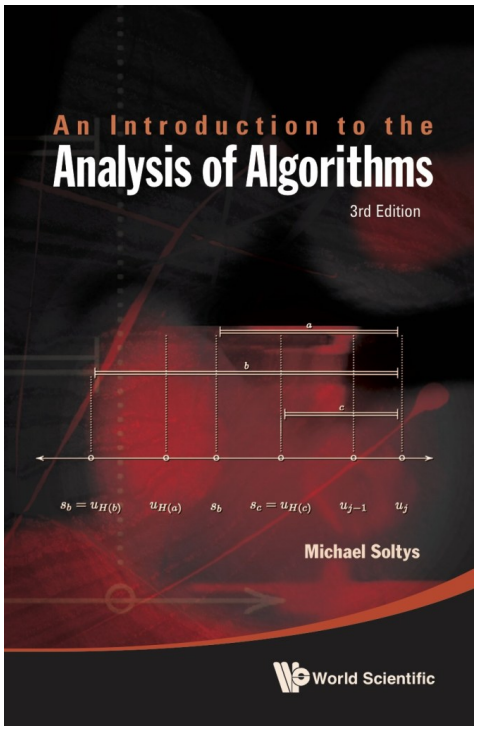
CSU Channel Islands

[**Git** Date:2019-09-11 Hash:d1ba29b Ed:3rd]

Course

An Introduction to the Analysis of Algorithms

`http://www.msoltys.com/book`



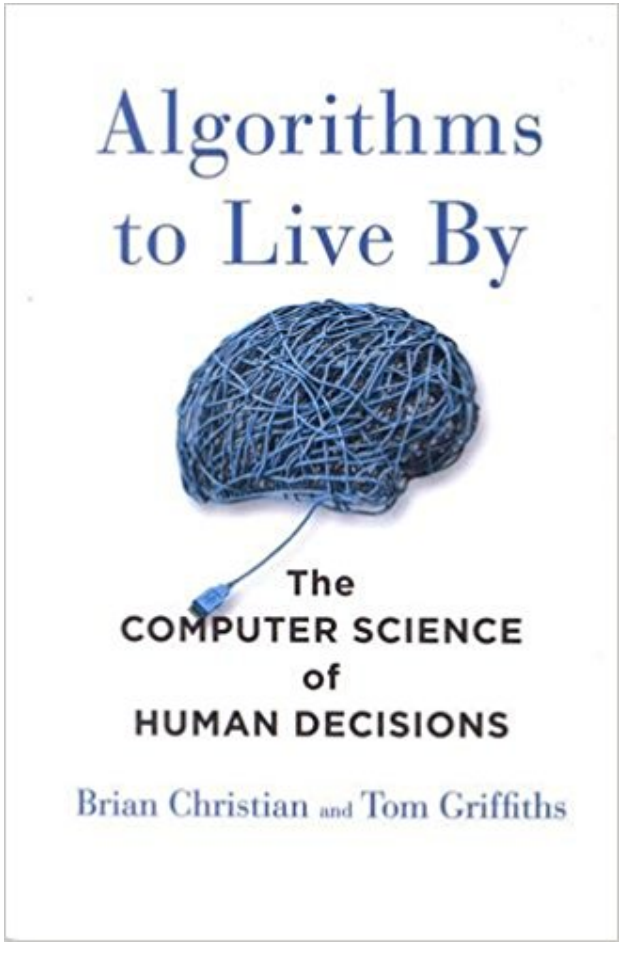
Outline

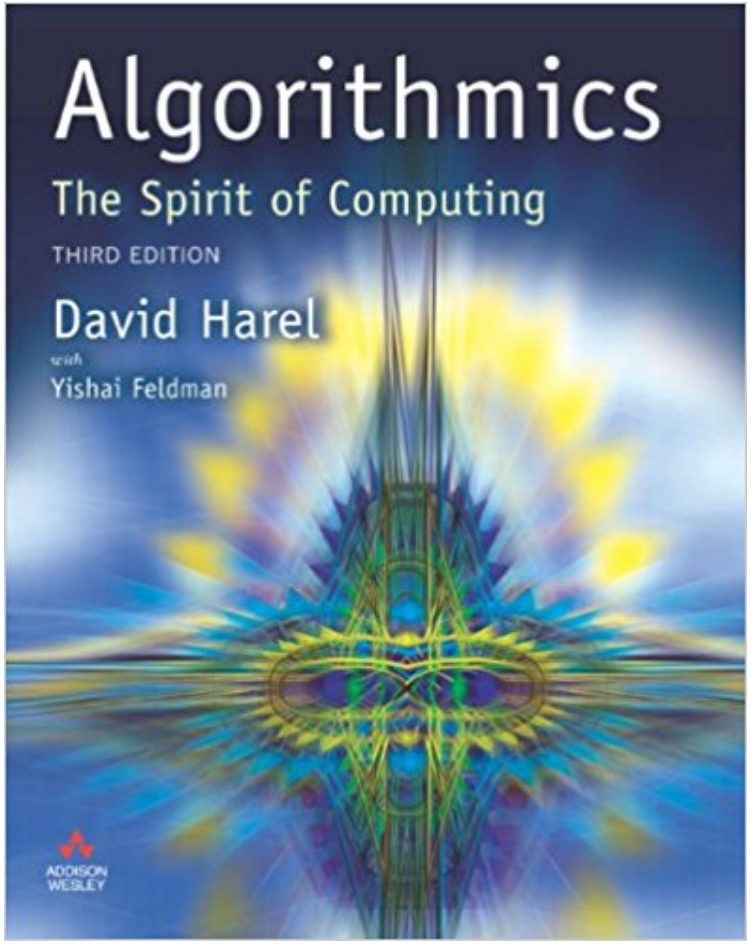
1. Preliminaries: division, Euclid, Palindromes, etc.; complexity, and ranking algorithms
2. Greedy algorithms: spanning trees, jobs, shortest path, Huffman codes, etc.
3. Divide and Conquer: mergesort, Savitch's algorithm, etc.
4. Dynamic Programming: longest monotone subsequence, knapsack problem, activity selection, etc.

---

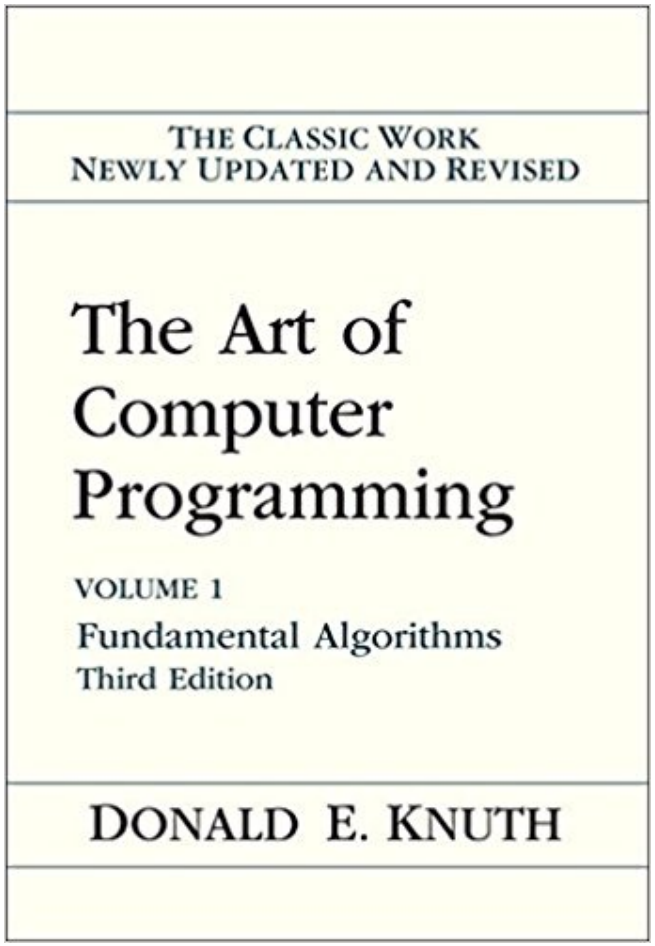
5. Online Algorithms
6. Randomized Algorithms
7. Parallel Algorithms

Great introductions to Algorithms

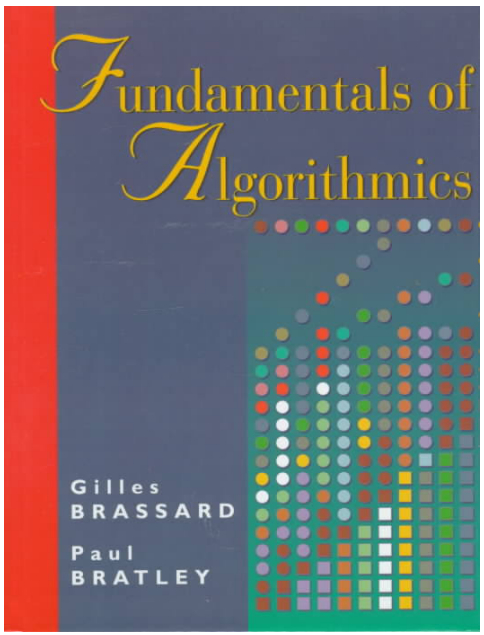


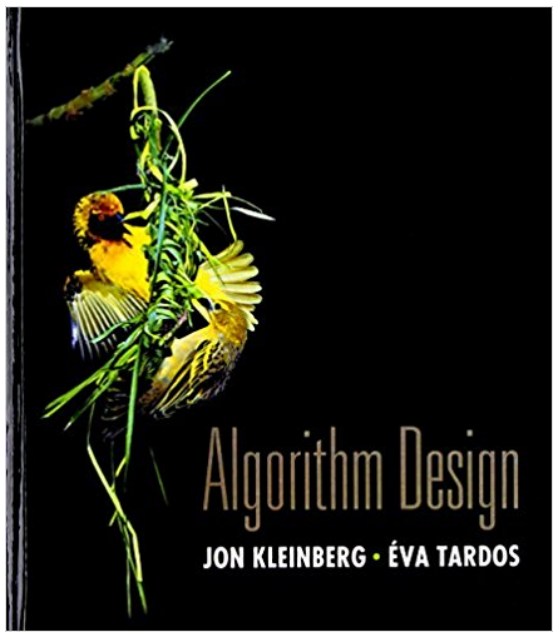


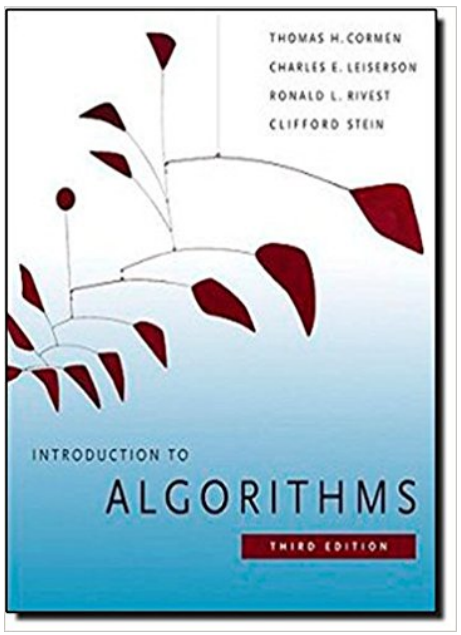
A classic


THE CLASSIC WORK
NEWLY UPDATED AND REVISED
The Art of Computer Programming
VOLUME 1
Fundamental Algorithms
Third Edition
DONALD E. KNUTH

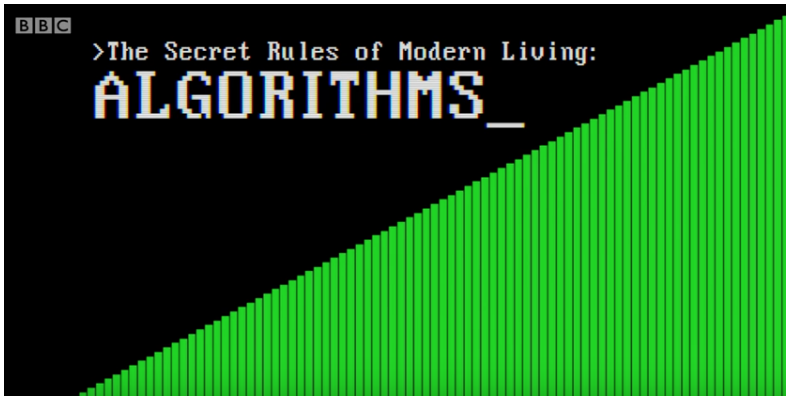
References


Fundamentals of Algorithmics
Gilles BRASSARD
Paul BRATLEY


Algorithm Design
JON KLEINBERG • ÉVA TARDOS


THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN
INTRODUCTION TO
ALGORITHMS
THIRD EDITION

A BBC Documentary by Marcus Du Sautoy



Student Learning Outcome (SLO)

Measured for the COMP 354 assessment
(ABET accreditation requirement)

> *Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.*

Rubric

Performance Indicator	**Unsatisfactory**	**Developing**	**Satisfactory**	**Exemplary**
1. **Algorithmic design: principle of computing**	no understanding of problem, no solution	problem understood, but solution wrong	problem understood and a solution given	problem understood and best solution given
2. **Performance analysis: computational complexity**	no understanding of what is requested	understanding of worst-case but no Big-O estimate	worst-case analysis and a Big-O estimate given	worst-case analysis resulting in tight Big-O estimate
3. **Proof of correctness: Mathematics as other discipline that helps identify solution**	no understanding of how to approach the proof	providing general direction but no details	an outline of the proof given and aspects of framework	a complete proof, with framework of pre/post-condition and invariants

All three rows will be measured by the corresponding question on the final exam:

A Design Question: A problem is posed, and the students must choose one of the three basic algorithm design techniques to solve it, and present the solution in clear and correct pseudo-code.

A Performance Question: An algorithm is posed, and the student must evaluate its time and/or space complexity in terms of worst-case performance expressed in Big-O notation, and tradeoffs, e.g., optimization versus speed, or time resources versus space resources.

Proof of correctness Question: The student will be given a problem, and an algorithmic solution will be requested, together with the proof of correctness of the algorithm; the student will be required to tie the algorithmic solution to the problem, and to show that the algorithm solves that problem.

All questions will designate how to measure the answer (as unsatisfactory, developing, satisfactory or exemplary).