演算法
Algorithms

沈錳坤
Man-Kwan Shan
Department of Computer Science
National Chechi University

加簽

- 修課同學必須具備資科系
  - 計程一、
  - 計程二、
  - 資料結構

  此三門課之基礎
- 遞補名單中的同學
  - 資科系學、碩、博請列印加簽單 2/16 前送交資科系辦
  - 雙主修、輔系同學請列印加簽單與三門課修課證明 2/16 前送交資科系辦
- 未在遞補名單中的資科系大學部重修同學
  - 請填寫個人姓名、學號、email，今天下課後交給老師
- 其他特殊情形請在下課後找老師
- 加簽優先順序：資科學>資科博>雙>資科碩=輔

旁聽

- 在教室空間允許的情況下，歡迎旁聽
- 旁聽同學請將學號email給謹任助教:

  111753124 @nccu.edu.tw
- 由助教手動加入wm5課程網站

Course Information

- Instructor
  - 沈錳坤 (Man-Kwan Shan)
  - Email: mkshan@nccu.edu.tw
  - Tel: 02-29393091-67622
  - Office: 大仁樓 200410
  - Office Hour: Thu. 12:00~13:00, or by appointment

- Teaching Assistant
  - 巫謹任(111753124@nccu.edu.tw)
  - 田詠恩 (108703030@nccu.edu.tw
  - Office: 大仁樓 200407

課程目標

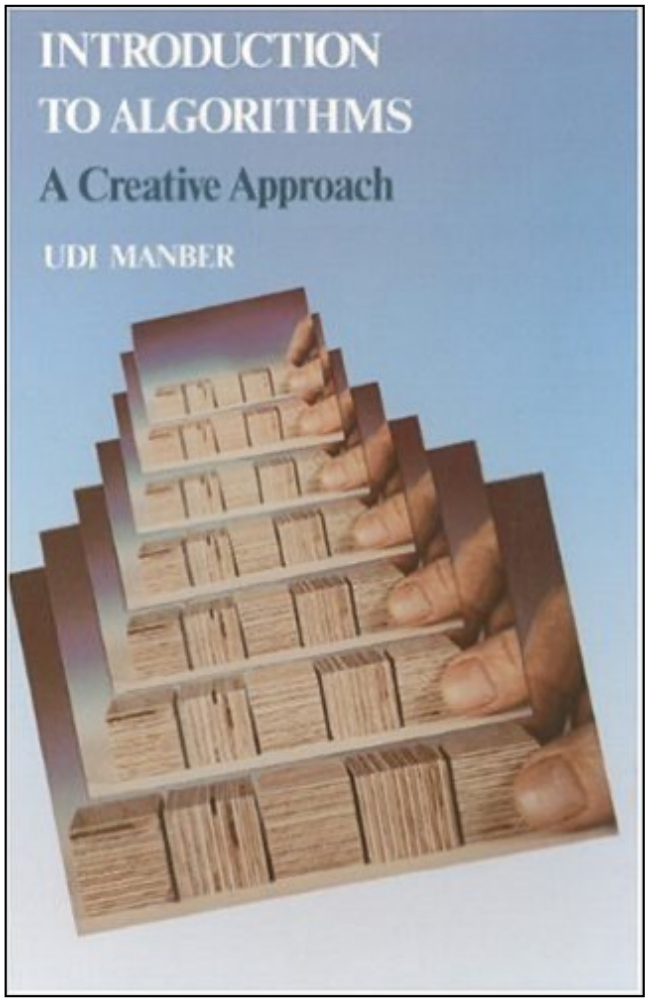
- Objective
  - to learn methods for algorithm design
  - understand principles behind the methods
  - know how to apply these methods
  - know when to apply these methods

課程定位

- Focus on
  - 瞭解演算法背後的思維
  - 學習如何構思設計演算法
  - 學習如何應用演算法
- Not a course about
  - 熟悉知名演算法
  - 程式解題 (快速正確的解題)
  - AI 演算法

Text Book

- **Introduction to Algorithms: A Creative Approach, by Udi Manber.**



Author : Udi Manber

- Professor, Univ. of Arizona
- Chief Scientist, Yahoo !, 1998.
- Chief Algorithms Officer, Amazon, 2002.
- Vice President of Engineering, Google, 2006.
- Vice President of Engineering, Youtube, 2014.
- National Institutes of Health, 2015.
- Technical Advisor, Institute for Computational Health Sciences, University of California, San Francisco, 2017.

★★★★★ **The approach taken by the writer is unique and to my opinion better than any other Algorithm's books I've seen**

By modern c++'ser on January 31, 2015

Format: Paperback | **Verified Purchase**

if you want to learn Algorithms, you should read CLRS introduction to Algorithms.
But if you want to learn how to design algorithms yourself then the book you need to read is this one by Udi Manber.
The approach taken by the writer is unique and to my opinion better than any other Algorithm's books I've seen.
The writer tries to teach the reader how to design algorithms through the usage of mathematical induction - hence the words "A creative approach".
I find this book to be so good, that even though I've read most of the chapters already I keep coming back to it.
This is a must read for anyone who aspires to be a computer scientist.

★★★★★ **I always wanted a book which would teach how to ...**

By Ravi on January 29, 2016

Format: Paperback | **Verified Purchase**

I always wanted a book which would teach how to design algorithms, rather than just state the algorithm. CLRS while comprehensive states the algorithm first and then formally establishes correctness. While this is how most papers/books are written, I am not a big fan of such an approach. I instead prefer a tight connection between an algorithm and its proof of correctness. Udi is a master of this, and he does this very well in this book.

I must admit I have not yet read a large portion of this book. But I have read a majority of Chapter 5, which is the core chapter of this book, and I am already impressed. Thanks a lot Udi Manber for writing this jewel.

★★★★★ **Complementary to the CLR.**

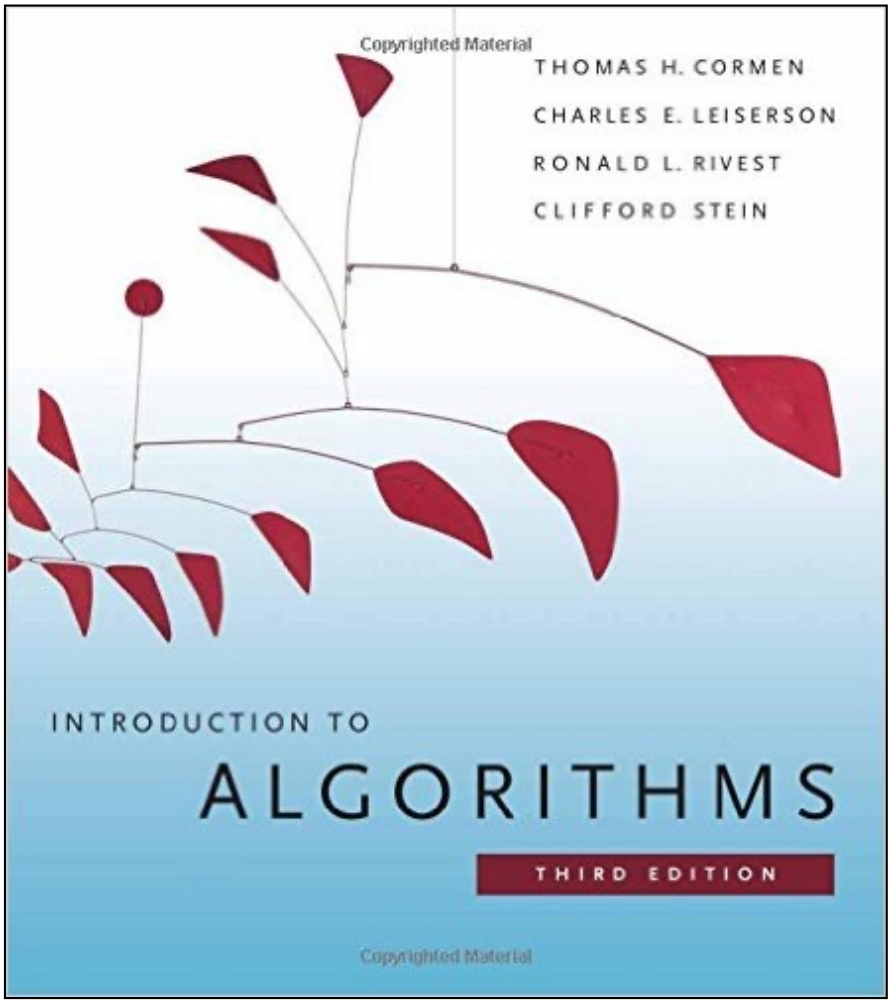
By Komtanoo Pinpimai on April 20, 2008

Format: Paperback | **Verified Purchase**

I wonder why this book is not as popular as it should be. Although the CLR is the mandatory book of most introduction to algorithm classes, it does not say much of how they came up with those algorithms which is the role of this book. The unique interesting thing is it uses induction to explain how each algorithm was developed, however I guess it's not the primary objective of the author. He wanted readers to read the description of the problems that those algorithms try to solve, and learn to apply induction to solve them on their own. If you like solving puzzle, you will love it.

Text Book

- **Introduction to Algorithms, by T. H. Cormen, C. E. Leiserson, R. L. Rivest, & C. Stein.**



Grading

- Grading (Tentative)
  - Midterm exam. (30%~35%)
  - Final exam. (30%~35%)
  - Homework (30%~35%

  * Midterm & Final exam.: normalization by interpolation
- Homework (Tentative)
  - 6~7 programming assignments (in C Language)
  - 1~3 exercises

課程網站

1. 政大wm5
   - 上課投影片
   - 計中自動建立修課同學帳號
   - 投影片 != 課本 (投影片是上課輔助工具)
   - 投影片會在上課前上傳
   - 投影片課後會根據同學學習效果來更新修飾，不影響內容
2. Online Judge
   - 繳交程式
   - 同學自己登記暱稱帳號

程式能力檢定 2023 Spring

2021/03/21 (二) 17:30~21:40上機考試
(2021/03/17 (五) 18:00 報名截止)

2021/05/23 (二) 17:30~21:40上機考試

Importance of Algorithms

- Data Structures + Algorithms + Coding Skill = Programming
- Foundation of Computer Science
- 研究所入學考、博士班資格考必考科目
- Interview of Software Engineer

Computational Thinking

- Computational thinking is about looking at a problem in a way that a computer can help us to solve it.
  - Logical reasoning: predicting and analysing
  - Algorithms: making steps and rules
  - Decomposition: breaking down into parts
  - Abstraction: removing unnecessary detail
  - Patterns and generalization: spotting and using similarities
  - Evaluation: making judgement

If you didn't pay attention during college lectures, it can come back to bite you when you least expect it.

Immigration officers are now asking people to solve computer science problems to verify their antecedents, claims a Nigerian software engineer. "I was just asked to balance a Binary Search Tree by JFK's airport immigration," tweeted Celestine Omin, who's a based out of Lagos, and was traveling to the US. "I was too tired to think of the BST solution," he rues, saying that he'd just been on a 23 hour flight.

I and my gf live in SF. We went to meet my gf family in Seattle. My gf dad is principal engineer at Msft. It was dinner time. We started with casual conversation about tech jobs and stuff. From there we started talking about interview process and out of nowhere he asked me how i would solve an array question. He acted like it was casual conversation but I could see how he was evaluating my answer. Initially I was taken back a little but then I got hold of my senses. After all I have years of LC experience. This was day to shine. I started answering with brute force solution followed by optimal solution. Talked about time and space complexity. In the end he was somewhat satisfied.

They are asking us to visit again during Christmas break. I wonder if I should prepare for system design.

Girl is Asian. I m not.

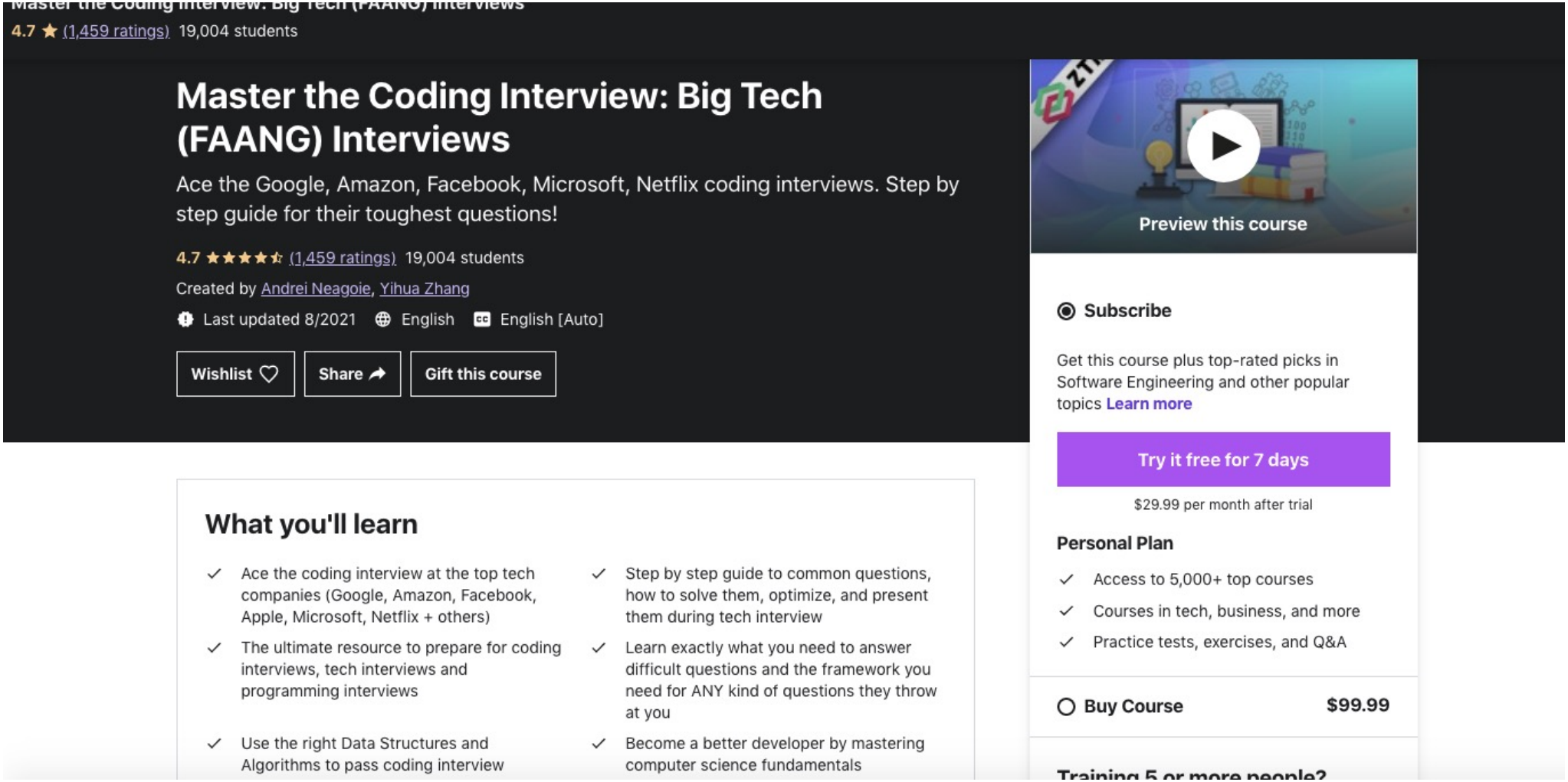
Course Contents

- Introduction
- Mathematical Induction
- Design of Algorithm by Induction
- Algorithms Involving Sequence and Sets
- Graph algorithms
- Geometric Algorithms
- Reductions
- NP-Completeness

1. Review of Algorithm Design Strategy
   . Divide and Conquer
   . Greedy Algorithm
   . Dynamic Programming
2. Algorithm Design by Induction
   . Evaluating Polynomial
   . Finding One-To-One Mapping
   . Celebrity Problem
   . Skyline Problem
   . Balance Factor in Binary Tree
   . Finding Maximum Consecutive Subsequence
   . Knapsack Problem
3. Algorithms Involving Sets
   . Binary Search
   . Finding Minimum in Cyclic Sorted Sequence
   . Binary Search of Unknown Size
   . Stuttering-Subsequence
   . Solving Equation: Bisection Algorithm
   . Interpolation Search
   . Bucket sort, Radix sort
   . Insertion sort, Selection sort, Bubble sort
   . Merge sort, Quick sort, Heap sort
   . Low bound of Comparison-based Sorting
   . Finding both Maximum and Minimum Elements
   . Finding the K-th Smallest Element

4. Algorithms Involving Sequences
   . KMP Algorithm
   . String Editing Distance
   . Longest Common Subsequence
   . Dynamic Time Warping (Optional)
   . Huffman Tree
5. Graph Algorithms
   . Graph Traversal (DFS, BFS)
   . Finding Connected Components
   . Finding a Cycle
   . Topological Ordering
   . Unweighted Shortest Path
   . Weighted Shorts Path over Acyclic graph
   . Weighted Shortest Path over Cyclic graph (Dijkstra's Algorithm)
   . Floyd–Warshall algorithm for All Pair Shortest Path
   . Minimum Spanning Tree (Prim's and Kruskal's Algorithms)
   . Bipartite Graph Matching
   . Network Flow & Augmenting Path
   . Graph Coloring (Optional)
   . Bi-connected Component
6. Geometric Algorithms
   . Determining a Point inside Polygon
   . Constructing Simple Polygons
   . Convex hull (straightforward, gift wrapping, Graham's Scan algorithms)
   . Closest Pair Algorithm

7. Reduction
   . Reduction Techniques
   . Reduction of Cyclic String Matching
   . Reduction of Systems of Distinct Representatives
   . Reduction of Sequence Comparison
8. NP Completeness
   . Finite State Machines
   . Turing Machines
   . Decision Problems vs. Optimization Problems
   . P, NP, NP-Hard and NP-Complete
   . P = NP ?
   . Satisfibility Problem
   . Proof of NP-Completeness (Clique Problem, Vector Cover Problem)
   . Branch and Bound


Master the Coding Interview: Big Tech (FAANG) Interviews
4.7 ★ (1,459 ratings) 19,004 students
Master the Coding Interview: Big Tech (FAANG) Interviews
Ace the Google, Amazon, Facebook, Microsoft, Netflix coding interviews. Step by step guide for their toughest questions!
4.7 (1,459 ratings) 19,004 students
Created by Andrei Neagoie, Yihua Zhang
Last updated 8/2021
English
English [Auto]
Wishlist
Share
Gift this course
Preview this course
Subscribe
Get this course plus top-rated picks in Software Engineering and other popular topics Learn more
Try it free for 7 days
$29.99 per month after trial
Personal Plan
Access to 5,000+ top courses
Courses in tech, business, and more
Practice tests, exercises, and Q&A
Buy Course
$99.99
What you'll learn
Ace the coding interview at the top tech companies (Google, Amazon, Facebook, Apple, Microsoft, Netflix + others)
The ultimate resource to prepare for coding interviews, tech interviews and programming interviews
Use the right Data Structures and Algorithms to pass coding interview
Step by step guide to common questions, how to solve them, optimize, and present them during tech interview
Learn exactly what you need to answer difficult questions and the framework you need for ANY kind of questions they throw at you
Become a better developer by mastering computer science fundamentals

2. Data Structures Used:

- *Arrays*
- *Hash Tables*
- *Singly linked lists*
- *Doubly linked lists*
- *Stacks*
- *Queues*
- *Binary Trees*
- *Binary Search Trees*
- *Tries*
- *N-ary Trees*
- *Min/Max Heaps*
- *Priority Queues*
- *2-D Arrays/ Matrices*
- *Graphs*
- *Adjacency List*
- *Adjacency Matrix*
- *Interface Design*

3. Algorithmic Paradigms Used:

- *Recursion*
- *Sorting*
- *Searching*
- *Tree Traversals*
- *Graph Traversals*
- *Breadth First Search*
- *Depth First Search*
- *Divide and Conquer*
- *Greedy Method*
- *Dynamic Programming*
- *Backtracking*

4. Specific Algorithms Used:

- *Hoare's Quickselect Algorithm*
- *Floyd's Tortoise and Hare Cycle Detection Algorithm*
- *Bellman-Ford Algorithm*
- *Dijkstra's Algorithm*
- *Topological Sort*

What is an algorithm ?

algorithm

noun

Word used by programmers when they do not want to explain what they did.

Conclusions

- Algorithm: a step by step procedure for solving a problem
- Steps of design & implementation of algorithms
  - design
  - proof of correctness
  - analysis
  - implementation
- Methodology for algorithm design in the Book by Udi Manber
  - based on mathematical induction
  - extending solutions of small problems to solutions of large problems
  - Not a universal approach