CS146
Lecture 0

GREENSHEET

INSTRUCTOR: KATERINA POTIKA
CS SJSU

Information

- Lectures:

  - Section 1: TTh 10:30-11:45 a.m.

Office Hours

- TTh: 9:30-10:15PM
- T 1:30-2:00PM
- or by appointment

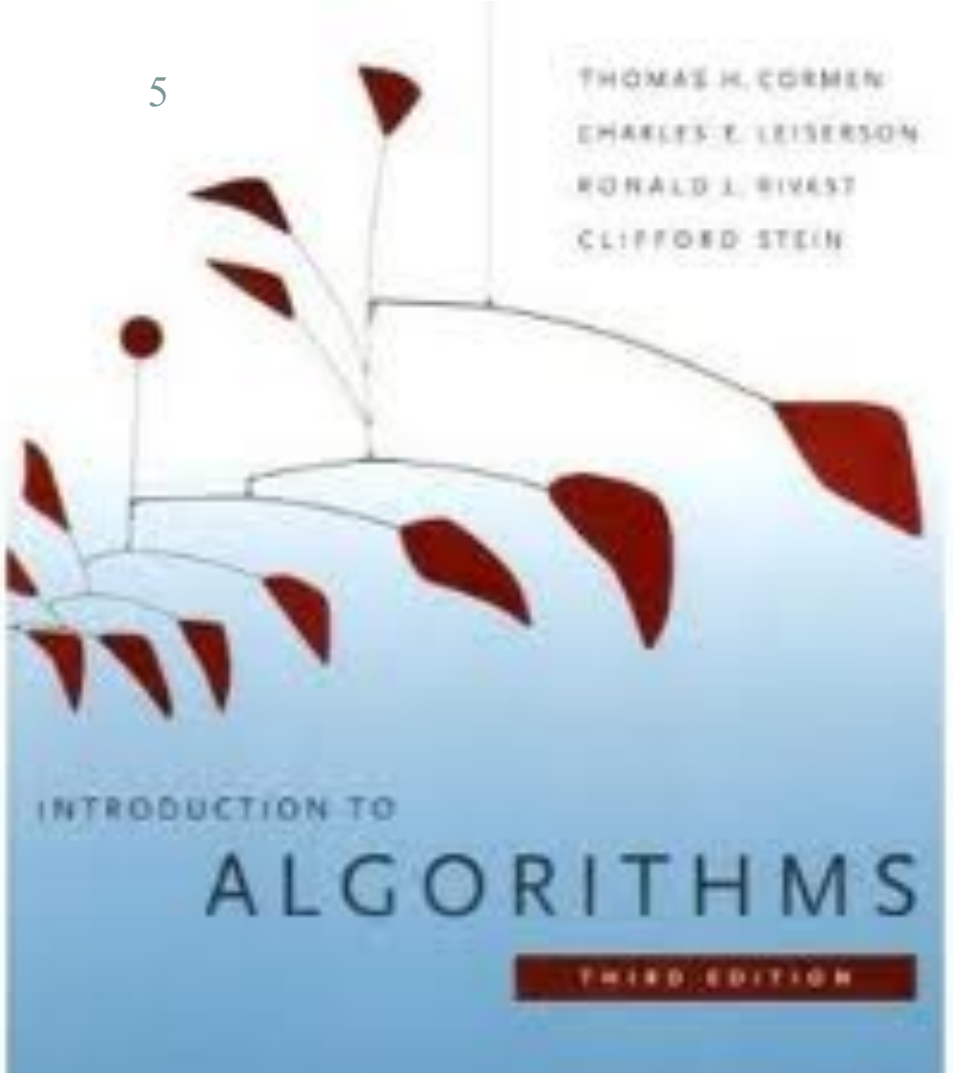
- Office: MH215

Prerequisites

- CS46B or CS49J
- Math30
- Math42

- Need your Transcripts (highlight prereqs) send by 1/31!

Textbook


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

Important Dates (fixed)

- Midterms (40% of grade)
  - Thursday, 3/2
  - Thursday, 4/13
- Final (40% of grade)
  - Section 1: Monday 5/22 9:45-12pm

Remaining 20% of grade

- 15% Programming assignments (at most 4)
- 5% Quizzes
- 0% Homework Assignments
- 0% Reading Assignments

Final Grade

- Students must obtain >50% in each component of the course in order to be eligible for a grade of C- or better.

Remember:

- Students must obtain a passing grade (>50%) in all components of the course in order to pass the class
  - Failing to do so will result in them failing the class :(
- late assignments will be accepted with a penalty check syllabus.
  - An extension will be granted only if a student has a written medical excuse (doctor's note).
- The exam dates are final and there will be no makeup exams
- No email submission of assignments

CS146 Web site @ canvas

- Contains/Will be uploaded throughout the semester:
  - Announcements (News)
  - Green Sheet (Syllabus)
  - Lecture slides
  - Homework Assignments
  - Project Assignments
  - Upload your Assignments
  - Important dates
  - Online forum
    - Discussion forum
    - Email (may forward to your regular email)

It is your responsibility to visit the web site regularly (every 1-2 days) in order to access important information related to the course!

ID please…

- Please say your name every time you speak
- Also, upload your photo on your profile @ canvas

Course Objectives I

- To ensure that students are familiar with ways to implement elementary data structures and their associated algorithms.

Course Objectives II

- To introduce students to the implementation of more complex data structures and their associated algorithms.

Course Objectives III

- To acquaint students with advanced sorting techniques.

Course Objectives IV

- To teach students how to determine the time complexity of algorithms.

Course Objectives V

- To introduce students to algorithm design techniques.

Student Learning Objectives I

- Upon successful completion of this course, students will be able to understand the implementation of
  - lists,
  - stacks,
  - queues,
  - search trees,
  - heaps,
  - union-find ADT, and
  - graphs and
  - be able to use these data structures in programs they design

Student Learning Objectives II

- Upon successful completion of this course, students will be able to prove
  - basic properties of trees and graphs

Student Learning Objectives III

- Upon successful completion of this course, students will be able to perform
  - breadth-first search and
  - depth-first search on directed as well as undirected graphs

Student Learning Objectives IV

- Upon successful completion of this course, students will be able to use
  - advanced sorting techniques (heapsort, mergesort, quicksort)

Student Learning Objectives V

- Determine the running time of an algorithm in terms of asymptotic notation

Student Learning Objectives VI

- Solve recurrence relations representing the running time of an algorithm designed using a divide-and-conquer strategy

Student Learning Objectives VII

- Understand the basic concept of NP-completeness and realize that they may not be able to efficiently solve all problems they encounter in their careers

Student Learning Objectives VIII

- Understand algorithms designed using
  - greedy,
  - divide-and-conquer, and
  - dynamic programming techniques

Data structures

- Way to store and organize data
- No single data structure works universal
- Know the strengths and limitation of each
- Appropriate use in algorithms
- Examples: primitive types (integer, character), array, record, union etc

Abstract Data Types

- An **abstract data type** is a **type** with associated operations, but whose representation is hidden.

- Examples: queue, stack, priority queue, list, set, map, graph etc

Algorithm

- Algorithm
  - A finite sequence of precise instructions that leads to a solution
  - It takes an input and produces an output
- A tool to solve computational problems
- From the name of the Persian Mathematician al-Khwārizmī

Technique for designing algorithms

- Various exist that can help us
  - Brute-force search or exhaustive search
  - Divide and conquer
  - Greedy
  - Dynamic Programming
- No ‘cookbook’ for algorithms
- Help to prove correctness and efficiency of your algorithm

Hard Problems

- We mostly will see *efficient algorithms*
- Can we find efficient algorithms for all problems?
- Efficient : Time +Space
- NP-complete problems: no efficient solution
- NP-complete problems are interesting…