Course Information

Course Overview

This course is designed as a deep dive into the design, analysis, implementation, and theory of data structures. Over the course of the quarter, we'll explore fundamental techniques in data structure design (isometries, amortization, randomization, word-level parallelism, etc.). In doing so, we'll see a number of classic data structures, as well as some more modern ones. By the time we've finished, we'll have seen some truly beautiful strategies for solving problems efficiently.

Instructor

Keith Schwarz (htiek@cs.stanford.edu)
Office: Gates 178
Office Phone: (650) 723-4350

TAs

Anton de Leon
Ryan Smith
Michael Zhu

Email

The course staff can be reached at cs166-spr1819-staff@lists.stanford.edu. Please don't hesitate to send us emails! We're here because we genuinely love this material and want to share it with you. If you have any questions on the material, or if you're interested in exploring more advanced content, please get in touch with us. We'd be happy to help out.

Lectures

Tuesdays and Thursdays from 3:00PM – 4:20PM in room 420-041 (building 420, room 041). Lectures will not be recorded, so you're encouraged to attend!

Units

CS166 is offered for either three or four units. Undergraduates are required to enroll for four units, while graduate students can enroll for either three or four units. The course content and requirements are the same in the three-unit and four-unit versions of the course and the unit flexibility is purely to help graduate students stay under unit limits.

Website

The course website is http://cs166.stanford.edu and it's loaded with resources for this course. There, you'll find all the course handouts, the syllabus, links to readings, and all sorts of other resources.

Office Hours

We'll be holding a few sets of office hours each week. We'll post a calendar to the course website once they're set up.

Prerequisites

The prerequisites for this course are CS161 and CS107.

From CS161, we expect you to feel comfortable designing and analyzing nontrivial algorithms and writing proofs of correctness. Mathematically, you should be comfortable using asymptotic notation (o, O, Θ, Ω, and ω), solving recurrence relations, manipulating inequalities, and simplifying summations. We'll also expect that you're comfortable with divide-and-conquer, greedy algorithms, and dynamic programming; that you're familiar with randomized algorithms (and related concepts like universal families of hash functions); and that you're comfortable writing correctness proofs for algorithms of each of these types. You should also feel comfortable with standard algorithms like Dijkstra's algorithm, Prim's algorithm, quicksort, etc.

The CS161 prerequisite, by transitivity, also means we assume you have the equivalent of CS103 (discrete mathematics, automata, and proofwriting) and CS109 (probability and basic combinatorics) as well. If you have never written a formal mathematical proof, or if the phrase “linearity of expectation” doesn’t ring a bell, you may want to come talk to us before jumping into CS166.

From CS107, we expect that you're comfortable writing and testing nontrivial programs and working from the command line. You should also feel comfortable with binary representations of numbers. We'll expect that you've at least heard of the memory hierarchy and are comfortable with the idea that not all memory accesses take the same amount of time. Additionally, we expect that you'll be comfortable writing code in both C and C++.

If you're unsure whether CS166 is the right place for you, please feel free to get in touch with the course staff.

Readings

The *recommended* reading for this course is *Introduction to Algorithms, Third Edition* by Cormen, Leiserson, Rivest, and Stein. This is an excellent textbook to have on-hand if you're doing anything related to algorithms and data structures, and we hope you find it useful. Copies are on reserve in the Engineering Library.

Additionally, there will be a variety of readings posted online (papers, course notes, slides, articles, etc.) Check the website for details on the readings for each lecture. I will try to present the salient features of each data structure in lecture, so depending on your learning style, you may find it useful to do the readings right before or right after lecture.

Assignments

Over the course of the quarter, there will be six problem sets. These problem sets will contain a mixture of theoretical questions and coding questions that are relevant to the week's material. You may either work on the problem sets individually or in pairs. If you work in a pair, you will turn in a joint problem set submission and both members of the pair will receive the same grade. More details are in an upcoming handout.

Midterm

There will be a take-home midterm exam in Week Nine of the quarter. The midterm will be given out on Tuesday, May 28th and is due on Thursday, May 30th.

Final Project

Over the course of the quarter, you will be asked to complete a final project in which you'll research one or more advanced data structures and present your findings. You’ll submit an initial project proposal at the end of Week Five, will submit a checkpoint detailing your initial progress in Week Seven, and will present your overall project in Week Ten. We’ll provide more details in a follow-up handout once we get closer to the project proposal deadline.

Grading

Your grade is computed from three subscores, computed as follows:

Assignment Score = (Points Earned) / (Points Possible),

Midterm Score = (Points Earned) / (Points Possible),

Project Score = (Points Earned) / (Points Possible).

Your raw score in CS166 is then computed as

(Assignment Score + Midterm Score + Project Score) / 3 – Late Penalty.

(The late penalty accounts for late problem set submissions and is described in the Problem Set Policies handout). We then apply a grading curve to raw scores to assign letter grades. We never assign letter grades that are lower than the decile of your raw score; for example, a 90% will never map to anything lower than an A-.

Your final grade will be determined solely according to the grading curve as applied to the raw score computed above. We do not offer any make-up work.

Incompletes

If you have a medical or family emergency and cannot complete the work in this course, you may contact Keith (not the TAs) to request an incomplete. We reserve incompletes only for emergencies, so we do not grant incomplete grades for poor performance on the assignments or exams, nor do we offer incompletes for busy work schedules.

In order to be eligible for an incomplete, you must have completed all of the assignments (except possibly the most-recently-due assignment) and must have a satisfactory academic performance as determined by the course instructor.

All incompletes are worked out on a case-by-case basis, and the instructor retains final discretion to approve or reject any requests for an incomplete.