# 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, 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](mailto:htiek@cs.stanford.edu))

## TAs
Francisco Pernice
Jose Calinawan Francisco

I'm excited to have Francisco and Jose on staff this quarter – they're veterans of CS166!

## Email
The course staff can be reached at cs166-spr2122-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
Lectures are on Tuesdays and Thursdays from 3:15PM – 4:45PM in STLC 111.

The first two weeks of the quarter will be conducted online. For the first week of the quarter, lectures will be online over Zoom. For the second week of the quarter, lectures will be prerecorded.

Aside from this, lectures will not be recorded this quarter.

## 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](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. We will not host much on Canvas aside from the lecture Zoom links for the first week of the quarter and videos for the first two weeks.

## Office Hours
It's important to us that everyone in the course feels that they can easily get in touch with the course staff when they have questions. We will be holding a large number of office hours time slots throughout the week. The office hours calendar can be found online through the course website.
Prerequisites  The prerequisites for this course are CS161 and CS107.

From CS161, you should be able to design and analyze nontrivial algorithms and write proofs of correctness. You should be comfortable using asymptotic notation ($\Theta$, $O$, $\Omega$, and $o$), solving recurrence relations, manipulating inequalities, and simplifying summations. We'll also expect that you're comfortable with divide-and-conquer algorithms, 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, mergesort, etc. and feel comfortable with their runtimes and proofs of correctness.

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 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. We understand that not everyone has a copy of this book or can get a copy, and that's okay. There's nothing in that textbook that you'll absolutely need for the course, and it's mostly there as a reference in case you'd like to look at certain topics in more depth.

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  Routine assignments in CS166 this quarter come in two flavors. First, there are problem sets. Problem sets can be completed either individually or in a pair. Whether you work individually or in a pair, you'll submit exactly one copy of the problem set. You're welcome to talk to the course staff about the problem set questions and get help and support as needed.

Second, there are individual assessments. Think of the individual assessments as mini take-home exams. As the name suggests, you must complete the individual assessments individually. You're welcome to ask the course staff clarifying questions on the individual assessments, but otherwise we'll leave it up to you to figure out how to solve these problems.
Research Project  CS166 culminates in a research project in which, working with a team, you'll do a deep dive into a data structure topic of your choice. You and your teammates’ job will be to become the resident experts on the subject, do something “interesting” with your topic, and present the topic and your findings. The format of how, exactly, you'll do this is still being hammered out.

Most quarters we’ve offered CS166 the projects have had groups of three, though during the remote quarters of the previous years we had teams of four. Either way, during the first few weeks of CS166 you may want to chat with other folks in the course and think about who you might want to work with.

Exams  We will not be holding any exams this quarter. That role will be played by the individual assessments given out over the quarter.

Grading  Your grade in CS166 is determined as follows:

- Problem Sets: 20%
- Individual Assessments: 30%
- Research Project: 50%

We use raw scores for each component and apply a grading curve at the end of the quarter that can only help. We never curve down, so a 90% will always map to some flavor of A.

One caveat: in order to earn a passing grade in CS166, you must earn a passing score in each of the three above components. We will determine what “passing” means at the end of the quarter, but plan on setting the passing threshold for the problem sets and individual assessments at a 60% average score. Usually, the research projects we see are excellent and the two checkpoints serve to help guide projects on a successful path, so we think it’s unlikely that we’ll need to set any specific targets there.

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 project, 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.