Course Information

Course Overview

Are there "laws of physics" in computing? Are there fundamental restrictions to what computers can and cannot do? If so, what do these restrictions look like? What would make one problem intrinsically harder to solve than another? And what would such restrictions mean for our ability to computationally solve meaningful problems?

In CS103, we'll explore the answers to these important questions. We'll begin with an introduction to mathematical proofs and discrete structures, which will enable us to model problems that arise in computer science. In the course of doing so, we'll explore mathematical logic, discrete structures, and the mathematical nature of infinity.

We'll continue by exploring finite automata (mathematical models of computers with finite memory) and from there will explore context-free grammars and Turing machines (mathematical models of computers with unbounded memory). As we explore these models, we'll see their strengths and their weaknesses and will explore questions like "what does it mean to solve a problem?" and "why does this problem seem to resist a solution?" Finally, we'll conclude with a quick introduction to complexity theory and explore what we know – and what we don't – about efficient computation.

In the course of the quarter, you'll see some of the most impressive (and intellectually beautiful) mathematical results of the last 150 years. You'll see what proof-based mathematics is all about and will gain confidence using mathematics to model and solve problems. You'll learn about various discrete structures that arise throughout computer science. You'll learn how to think about computation itself and how to show that certain problems are impossible to solve. Finally, you'll get a sense of what lies on the frontier of computer science, especially with regards to the $\mathbf{P \overset{?}{=} NP}$ problem.

Instructor

Ryan Smith (rsmith20@cs.stanford.edu)

TAs

TBD

Website

The course website is cs103.stanford.edu and it's loaded with resources for this course. There, you'll find all the handouts and lecture slides, along with additional links you may find useful. I would suggest periodically polling the website to stay on top of any important developments in the course.

Email

The course staff can be reached at cs103-sum1920-staff@lists.stanford.edu. Please don't hesitate to email us! 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.

Campuswire   We will have a class Campuswire forum you can use to ask questions about the material and to get help and advice on the problem sets and discussion problems. Our policies regarding Campuswire use are covered in our Problem Set Policies handout. Campuswire functions similarly to Piazza that many of you may be familiar with.

Lectures   Monday, Wednesdays and Fridays, 9:30AM – 11:20AM online via Zoom. Lectures will be recorded although you are encouraged to watch live if you are able to. Links to live and recorded lectures will be present on Canvas, Stanford's online portal for classes. It can be found at canvas.stanford.edu.

Units   If you are an undergraduate or are taking this course through SCPD, you need to enroll in CS103 for five units (these are department and university policies, respectively). If you are a matriculated graduate student, you may enroll for anywhere between three and five units, depending on what best fits into your schedule. Regardless of how many units you are enrolled for, the course content and requirements will be the same. The unit flexibility is simply to make life easier for matriculated graduate students.

Prerequisites   CS103 has CS106B/X as a prerequisite or corequisite. This means that if you want to take CS103, you must either have completed or be concurrently enrolled in one of CS106B or CS106X (or have equivalent background experience).

Over the course of the quarter, we will be giving out a number of programming assignments to help you better understand the concepts from the course. Those assignments will assume a familiarity with C++ and programming concepts (especially recursion) at a level that's beyond what's typically covered in CS106A. The timing on these assignments is designed so that they'll sync up with what's covered in CS106B/X.

Although CS103 is a course on the mathematical theory behind computer science, the only actual math we'll need as a prerequisite is high-school algebra. We'll build up all the remaining mathematical machinery we need as we go. We've released another handout detailing the mathematical prerequisites for this course, so if you have any questions, check it out and see what you find!

Office Hours   Ryan and the TAs will be holding lots of office hours during the week so that you can stop by and ask questions about the material. Feel free to stop on by if you need any help. We'll post a schedule during the first week of class.

Readings   There are online course notes for the first few weeks of material. They go into a lot more depth than what we're going to end up covering in CS103, but hopefully you'll find them useful for getting a deeper understanding of the material. The course notes are still a work in progress, so please feel free to contact us with corrections of all sorts – logic errors, grammatical issues, formatting problems, etc. We also will release a bunch of handouts over the quarter to provide additional supplementary reading material. Additionally, we'll release a number of graphical guides to various concepts covered throughout the quarter.

There are two recommended textbooks for this quarter. The first is **How to Read and Do Proofs** by Daniel Solow, which is a great resource for learning how to approach mathematical problem-solving. The second is **Introduction to the Theory of Computation, Third Edition** by Michael Sipser. You might find this book useful in the second half of the quarter. Some of the readings in the syllabus are taken from this book, but

we will not directly test you on any material in Sipser that is not covered as well in lecture or the problem sets.

Problem Sets   There will be seven total problem sets in CS103, given out about once per week. With the exception of Problem Set 0, which must be done individually, you are welcome to work on them individually or in pairs. Our full policies with regards to problem sets (late policy, regrades, etc.) are in the Problem Set Policies handout.

Exams   In addition to problem sets, there will be a midterm and a final exam. The midterm will be on Thursday July 23rd and the final will be on Friday, August 14th. Both are 3 hour exams, and are open-everything except for communicating with others. There will be a 24 hour window in which to start these exams. Once you start, you will have 3 hours to complete the exam + 10 minutes to submit it on Gradescope. The 24 hour period starts at 8:30AM.

Online Changes   This class is being offered in an online-only format, which changes some course aspects such as exams and finding problem set partners. The course staff is considering different solutions to these challenges and is committed to providing the best possible version of CS 103 that we can. Specific details will be available during the first week of class but be assured that we will be as accommodating as we can in this unique situation.

Grading   We compute final grades based on two subscores: one for problem sets and one for the exams.
Your problem set score is computed as

$$\text{PSet Score} = (\text{Points Earned} / \text{Non-Extra-Credit Points Possible})^{0.5}.$$

In other words, we will take your raw composite problem set score, then take the square root. This has the effect of boosting your problem set grades. For example, if you have a 81% raw score on the problem sets, you'd end up with a 90% for your assignment score. Similarly, if you had a 64% raw score on problem sets, you'd end up with an 80% for your assignment score. The problem sets are where you will do most of the learning in this course, and it's important that you complete each of them. Therefore, we do not drop your lowest problem set score.

Your exam score is computed as an unmodified raw score weighted equally between both exams:

$$\text{Exam Score} = 0.5 * (\text{Midterm \%} + \text{Final \%})$$

Note that, in particular, this means that we do not curve exam scores.

We will compute your final raw score at the end of the quarter as

$$\text{Raw Score} = 0.5 \cdot \text{Pset Score} + 0.5 \cdot \text{Exam Score} - \text{Late Penalty}.$$

(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. Historically, we've used the median raw score as the B/B+ cutoff. 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.

Honor Code We want to foster a collaborative and supportive atmosphere in CS103. This is why, for example, we will have so many office hours sections and why we let you work in pairs on the assignments. We expect you to abide by the letter and the spirit of the Stanford Honor Code in CS103. You are required to read and abide by the policies detailed in our handout on the Honor Code as it applies in CS103, which among other things discusses our expectations for what is and is not permissible collaboration on the problem sets.

We hope that you will respect the Honor Code, comport yourself with integrity, and work to create a learning environment where everyone feels supported.

Incomplete Policy If a serious medical or family emergency arises and you cannot complete the work in this course, you may contact Ryan – 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 solid academic performance in the course, as determined by the instructor. The instructor has the final say in whether to grant or deny incompletes. While the above criteria indicate certain cases in which incompletes will not be granted, there are no situations in which the instructor is obligated to offer an incomplete.

Useful Websites
http://web.stanford.edu/class/archive/cs/cs103/cs103.1208/   the course website
Campuswire.com   discussion site
Gradescope.com   submissions site
canvas.stanford.edu   home of videos
overleaf.com   LaTeX editing website.

Course Outline

Dates and topics below are tentative and subject to change.

Part One: Discrete Mathematics			
Date	**Topics**	**Readings**	**Assignments**
Monday, June 22	Can computers solve all problems? • Set Theory • The Limits of Computing	Notes, Ch. 1 Online Guides	PS 0 Out
Wednesday, June 24	How do we prove results with certainty? • Direct Proofs	Notes, Ch. 2	
Friday, June 26	How do we prove something without directly proving it? • Proof by Contradiction • Proof by Contrapositive	Notes, Ch. 2	PS0 Due PS1 Out
Monday, June 29	How can we formalize our reasoning? • Propositional Logic		PS1 Checkpoint Due
Wednesday, July 1	How can we reason about collections of objects? • First-Order Logic, Part I		
Friday, July 3	Holiday, no class		PS1 Due PS2 Out
Monday, July 6	How do we rigorously define key terms? • First-Order Logic, Part II	Handouts Online Guides	
Wedndsay, July 8	How do we model relationships between objects? • Binary Relations Equivalence Relations	Notes, Ch. 5	
Friday, July 10	What does it mean to compare two objects? • Fundamental Theorem of Equivalence Relations • Strict Order Relations	Notes, Ch. 5	PS2 Due PS3 Out
Monday, July 13	How do we model transformations and associations? • Functions • Injections, Surjections, and Bijections	Handouts Notes, Ch. 6	PS3 Checkpoint Due
Wednesday, July 15	How do we model network structures? • Graphs • The Pigeonhole Principle	Notes, Ch. 4	
Friday, July 17	How can we reason about sequential processes? • Mathematical Induction	Notes, Ch. 3	PS3 Due PS4 Out
Monday, July 20	How does recursion relation to mathematical proof? • Variations on Induction • Complete Induction	Notes, Ch. 3 Handouts	
Part Two: Computability Theory			
Wednesday, July 22	How do we mathematically model computers? • DFAs • NFAs	Sipser 1.1	Midterm Tomorrow, July 23

Friday, July 24	How can we transform machines? • Equivalence of DFAs and NFAs • Closure Properties of Regular Languages	Sipser 1.3	
Monday, July 27	Can we generate new programs from old programs? • Regular Expressions • Equivalence of Regular Expressions and NFAs		
Wednesday, July 29	Can computers with finite memory solve all problems? • Nonregular Languages • The Myhill-Nerode Theorem	Sipser 2.1	
Friday, July 31	How do natural and formal languages overlap? • Context-Free Grammars • Context-Free Languages	Sipser 3.1	PS4 Due PS5 Out
Monday, August 3	How do we model realistic computers? • Turing Machines • Designing Turing Machines	Sipser 4.1, 4.2, 6.1	
Wednesday, August 5	What does it mean to solve a problem with a computer? • **R** and **RE** Languages • The Universal Turing Machine, Self-Reference	Online Guides	
Friday, August 7	What is the full scope of computing power? • Verifiers • Unrecognizability	Sipser 7.2 Sipser 7.3	PS5 Due PS6 Out
Monday, August 10	How do we measure the difficulty of problems? • The P versus NP Question • Reducibility, Part I		
Wednesday, August 12	Final Class: What makes hard problems hard? • Reducibility, Part II • NP-Completeness	Sipser 7.4	PS6 Due No late submissions