CS120 Fall 2022:
Introduction to Algorithms and their Limitations

Syllabus

Course website: https://harvard-cs-120.github.io/cs120/

Lecture times: TuTh 9:45-11am Eastern, Science & Engineering Complex (SEC), 150 Western Ave., Allston, Room 1.321

Synchronous course preview: Tuesday 8/23 1-2pm Eastern via Zoom (will be recorded). The Zoom link is available from Canvas. Please email the course staff (below) if you are not able to access it.

Shopping period office hours: Adam and/or Salil will hold virtual advising office hours most days during the course registration period.

- Adam: drop-in office hours listed on https://csadvising.seas.harvard.edu/contact/ .
- Salil: cs120 drop-in session from 5-6pm ET on Tue 8/23; 15min sign-up slots at other times. Zoom and sign-up links at http://salil.seas.harvard.edu/.

No enrollment cap: Unlike Fall 2021, there is no enrollment cap for CS 120 this semester.

Course Staff

Instructors:
Adam Hesterberg (he)
ahesterberg@seas.harvard.edu
Salil Vadhan (he/him/his; they/them/theirs)
http://salil.seas.harvard.edu/, salil_vadhan@harvard.edu

Head Teaching Fellow: Zach Ratliff (he/him/his)
zacharyratliff@g.harvard.edu

Patel Fellow (1:1 tutoring): Helen Cho (she/her/hers)
Reach out through this form, hcho@college.harvard.edu

Teaching Fellows: Silvia Casacuberta Puig, Jennifer Guo, Noa Kligfeld, Jennifer Liang, Michelle Lu, Andrew Nickerson, Phevos Paschalidis, Olivia Wenzel, Ayana Yaegashi, Iris Yan, Anurag Mitra, Benji Kan, Kathy Zhong, Patrick Song

Faculty Coordinator: Allison Choat (she/her/hers; they/them/theirs)
achoat@seas.harvard.edu

Whenever possible, please post questions for the course staff on Ed (privately if needed) rather than emailing us, so that we can all see the question and responses.

Overview

Looking at the world around us, we see computers solving problems on incredibly large scales: finding webpages relevant to our internet searches and returning them in sorted order, computing the quickest way to reach a destination given current traffic conditions, matching people on dating sites. How is this possible? More computing power? Intensive application-specific engineering? While these certainly have had a role to play, in CS120, you will see and learn how to use general algorithm design principles that cut across application domains and remain relevant even as computing technology changes.

First among these principles is mathematical abstraction, whereby we capture the essence of a computational problem (as well as the notion of a "computer") so that we can develop and analyze solutions independent of an implementation. Given these mathematical abstractions, we can apply a toolkit of basic algorithmic techniques in the search for solutions, and then gain certainty in their correctness and efficiency through rigorous mathematical proofs. Furthermore, the powerful concept of reductions will allow us to identify relationships between computational problems that seem very different on the surface, and thus automatically transfer solutions from one to another.

At the same time, some important computational problems have defied the search for algorithmic solutions. Computer scientists would love to have debugging tools that determine whether their programs can crash, natural scientists would love to have simulators that quickly determine the energy-minimizing states of physical or biological systems, and university registrars would love to be able to automatically schedule classes in a way that optimally maximizes the use of the best classrooms. Why have no scalable algorithms been found for these problems?

In the last part of CS120, we will see that many important computational problems are inherently unsolvable—they have no general algorithmic solution whatsoever! Others are solvable, but have no efficient algorithm—the minimum computation time inherently grows exponentially with the size of the problem instance. Uncovering these phenomena (known as "uncomputability" and "intractability", respectively) are unique benefits of a mathematically rigorous approach to algorithms. While we may sometimes be satisfied with empirical demonstrations of the

performance of an algorithm we have found, a proof seems to be the only way to convince ourselves that there is no algorithm whatsoever.

To summarize, CS120 aims to give you the power of using mathematical abstraction and rigorous proof to understand computation. Thus equipped, you should be able to design and use algorithms that apply to a wide variety of computational problems, with confidence about their correctness and efficiency, as well as recognize when a problem may have no algorithmic solution. At the same time, we hope you will gain an appreciation for the beautiful mathematical theory of computation that is independent of (indeed, predates) the technology on which it is implemented.

Learning Outcomes

By the end of the course, we hope that you will all have the following skills:

- To mathematically abstract computational problems and models of computation
- To design and implement algorithms using a toolkit of algorithmic techniques
- To recognize and formalize inherent limitations of computation
- To rigorously analyze algorithms and their limitations via mathematical proof
- To appreciate the technology-independent mathematical theory of computation as an intellectual endeavor as well as its relationship with the practice of computing
- To engage effectively in a collaborative theoretical computer science learning community, supporting your peers' learning as well as your own
- To clearly communicate mathematical proofs about computation to peers, conveying both high-level intuition and formal details.

Prerequisites

Experience with proofs and discrete mathematics at the level of Computer Science 20, and (Python) programming at the level of CS32/CS50. If you haven't taken these courses, we recommend using Problem Set 0 (to be posted during registration period) and the CS20 Placement Self-Assessment to gauge your preparation.

Curricular Context

First offered in Fall 2021, CS120 is a new introductory course in theoretical computer science, adding to CS121 (limits & models of computation) and CS124 (algorithms) in our curriculum. We have added this course in response to growing enrollments in 121 and 124 and to offer a greater variety of introductory theory courses to meet student interests and background. CS120 is meant to provide a gentler entry into theoretical computer science, which can be taken before CS121 or CS124, but these courses will *not* assume CS120 as a prerequisite. See this FAQ for a more detailed comparison between CS 120, 121, and 124, and the CS Advising site for how CS120 fits into the old and new CS concentration requirements.

Expectations and Format

This is the second offering of CS120, and the course is still somewhat experimental. By joining the class, you will be partners in the continued development and improvement of the course. We will solicit feedback from you periodically and expect to make adjustments based on it, so everything below is subject to change.

The main planned components of the course are as follows:

Main class meetings (TuTh 9:45-11am): About a third of our class meetings will begin with an approx. 30min *Sender-Receiver Exercise (SRE)* done in pairs, where the partners, the "sender" and "receiver," engage in an interactive dialogue to develop a joint understanding of a mathematical proof (related to recent class content) that the sender has studied beforehand.

The remainder of our class time will consist of interactive lectures on new material, with occasional in-class polls/surveys using the Ed platform (so please bring a web-enabled device that you can use to answer these questions).

Regular attendance in class is required. The SREs will develop important skills such as deconstructing and communicating proofs. Additionally, your peers will be depending on your participation and these will be a source of highlights to include in your participation portfolios (described below).

Sections (weekly, times TBD): Sections will be held weekly, at times TBD to accommodate as many students as possible. The sections will consist mainly of guided work on practice problems and interactive review of difficult concepts. You will find sections most useful if you have read and started thinking about the problem sets and the corresponding material in advance. Attendance is optional.

Office hours (every week, at multiple times TBD): The teaching staff will have regular office hours, to help guide you as you work through the material. Before coming to office hours with questions about the problem set (other than clarifications on what is being asked), it is important to invest real effort in trying to solve the problems without staff assistance. If you get stuck, the teaching staff will help you get unstuck, not by giving hints, but by probing your thought process and understanding of the content and/or reviewing material and related problems. The goal is to help you build the skills that will enable you to discover solutions for yourself.

Problem sets: There will be approximately 10 weekly problem sets, typically due Wednesdays at 11:59pm. Some of the problems may require deep thought, so you are strongly encouraged to begin them early and to brainstorm in groups of 2-3 students. (See Collaboration Policy below.)

Asynchronous discussions: We will use the *Ed* platform for discussions, Q&A, and announcements.

Participation portfolio:[1] CS120 is a collective learning experience, where we all should be working to support each others' progress. Thus, we expect you all to maintain a portfolio of the best examples of where your participation helped advance your classmates' learning or your own learning or helped improve the class. Four times during the semester we will ask you to submit highlights from your portfolio. These can be based on any form of participation: in class/section/office hours, in discussions on Ed, in our feedback surveys, in your collaborations on problem sets, etc.

Exams: There will be an in-class, 75min, closed-book midterm exam on Tuesday October 4, and a standard 3-hour final exam scheduled by the Registrar.

Grading

Breakdown. Half of your grade in cs120 will be based on the problem sets. The midterm and final exams will comprise one third of your overall grade in CS120, weighted proportionally to their length (75:180). The remaining sixth of your grade will be based on your participation portfolio submissions.

Problem set grading rubric. The problem sets will be graded using the following rubric:

- N: Not assessable. The assignment is too fragmentary or incomplete for us to assess the level of effort and understanding reached while working on it.
- L: Learning. Significant effort is evident, but there remain important misconceptions or gaps that will limit your ability to apply the skills and/or knowledge in the future (whether in the rest of CS120, in later CS courses, or outside of your CS education).
- R-: Nearly ready to move on, but with review and revision recommended. You have achieved the main learning objectives of the assignment, but there are some gaps that should be filled in on your path to mastery.
- R: Ready to move on. Your work meets all of the learning objectives of this assignment, and contains only minor errors (which we will note in our feedback to you).
- R+: Beyond ready. The work is nearly perfect, goes beyond expectations in its clarity or insight, and/or explores optional extensions.

Exams will be given percentage grades.

The reason for this coarse grading scale is to move away from nitpicking over point deductions and focus on the end goal of acquiring sufficient understanding to competently apply what you have learned.

Revisions. The material in CS120 is largely cumulative, so we wish to see you all reach an R-range grade on all assignments. But mistakes and misunderstandings are a part of the

[1] The idea of Participation Portfolios is taken from Harvard's course AC 221, taught by Prof. Mike Smith.

learning process, so we will allow you to submit short revisions (in the form of short videos) that clearly explain the misconceptions or gaps in your problem sets and the corrected understanding you have developed by studying the solutions. These revisions can increase up to five of your problem set grades from L to R- or R- to R (not both). We encourage you to submit revisions on all of your L and R- grades (even if you receive more than five) to ensure that you have achieved the learning objectives of the assignments; at the end of the course, we will use your budget of five in the way that maximizes your final letter grade. To make sure you do not fall behind, there will be a deadline for revisions on each problem set, approximately one week after graded assignments are returned. We will not accept revisions on assignments that receive an N or R grade.

Participation portfolio grading rubric. For consistency, we will use the same N/L/R-/R/R+ scale for grading your participation portfolio submissions, but with the following interpretations:

- N: Your submission is too incomplete or deviates too far from the instructions for us to assess your level of participation.
- L: Your submission demonstrates a minimal amount of participation in the course and effort on the assignment, but is below expectations.
- R-: You seem to be meeting most of the expectations for participation, but have not followed all of the guidelines for assignment.
- R: Your submission demonstrates that you are a fully engaged member of the course's learning community and meets the specified guidelines.
- R+: Your submission goes beyond expectations, in the level of your positive contributions to the course's learning community and/or the thoughtfulness of your reflections on your participation.

Participation portfolio submissions cannot be revised.

Letter grades. The table below describes the expected performance on each of the course elements corresponding to different letter grades.

letter grade range	psets	exams	participation
A	all problem sets (revised) should have R-range grades with at most 4 R-s	indicate full mastery of the subject	a mix of R and R+ grades
B	problem sets (revised) should have at most one N grade, with at most two Ls and a majority of Rs or R+s	indicate good comprehension of the course material	a mix of R and R- grades

C	problem sets (revised) should have at most two Ns, with a majority of R-range grades	indicate an adequate and satisfactory comprehension of the course material	a mix of R- and L grades
D	problem sets (revised) should have more R-range grades than Ns	indicate some minimal command of the course material	a mix of L and N grades
Grades of R+ are equivalent to R for the letter grade ranges defined above, but the difference between R+ and R may affect final grade +/-s. Students whose scores are a mix of the above categories will be proportionally averaged—for instance, problem set grades including three Ls and an N but exams that indicate full mastery of the subject would likely yield a B-range grade.			

The exams do not have predetermined cutoffs (e.g. percentages) corresponding to letter grades because, with this being a relatively new course, we do not have enough data from past exams to calibrate the difficulty level. We will draw the cutoffs for each exam by qualitatively assessing the level of mastery shown by the submitted exams at different scores.

Late days. You have a total of eight late days that you can use for problem sets or participation portfolios, using at most three on any one assignment. (That is, an assignment beyond 3 days late will automatically receive an N grade.) Do not needlessly use up your late days at the beginning of the semester; they are meant for accommodating unforeseen circumstances or crunches from your other classes or other aspects of your life. Any extensions beyond these late days require a note from your resident dean (or advisor, in the case of graduate students).

Diversity and Inclusion[2]

We would like to create a learning environment in our class that supports a diversity of thoughts, perspectives and experiences, and honors your identities (including race, gender, class, sexuality, socioeconomic status, religion, ability, etc.). We (like many people) are still in the process of learning about diverse perspectives and identities. If something was said in class (by anyone) that made you feel uncomfortable, please talk to us about it. If you feel like your performance in the class is being impacted by your experiences outside of class, please don't hesitate to come and talk with us. As a participant in course discussions, you should also strive to be open-minded and respectful of your classmates.

Health Accommodations[3]

If you have a physical or mental health condition that affects your learning or classroom experience, please let us know as soon as possible so that we can do our best to support your learning (at minimum, providing all of the accommodations listed in your DAO letter if you have one).

---

[2] Based on text by Dr. Monica Linden at Brown University.
[3] Based on text by Prof. Krzysztof Gajos at Harvard University.

Support Structures[4]

Everyone can benefit from support during challenging times. If you experience significant stress or worry, changes in mood, or problems eating or sleeping this semester, whether because of CS120 or other courses or factors, please do not hesitate to reach out to Adam, Salil, or other members of the course staff. Not only are we happy to listen and discuss how we can help you cope in CS120, we can also refer you to additional support structures on campus, including, but not limited to, the below.

- Bureau of Study Counsel
- InTouch
- Counseling and Mental Health Services, 617-495-2042
- Let's Talk
- Room 13, 617-495-4969

Collaboration Policy

Students are encouraged to discuss the course material and the homework problems with each other in small groups (2-3 people). Discussion of homework problems may include brainstorming and talking through possible solutions, but should not include one person telling the others how to solve the problem. In addition, each person must write up their solutions independently, and these write-ups should not be checked against each other or passed around. While working on your problem sets, you should not refer to existing solutions, whether from other students, past offerings of this course, materials available on the internet, or elsewhere. All sources of ideas, including the names of any collaborators, must be listed on your submitted homework along with a brief description of how they influenced your work.

In general, we expect all students to abide by the Harvard College Honor Code. We view us all (teaching staff and students) as engaged in a *shared mission* of learning and discovery, not an adversarial process. The assignments we give and the rules we set for them (such as the collaboration policy) are designed with the aim of maximizing what you take away from the course. We trust that you will follow these rules, as doing so will maximize your own learning (and thus performance on exams) and will maintain a positive educational environment for everyone in the class. We welcome and will solicit feedback from you about what more we can do to support your learning.

Content and Textbooks

The content covered in CS120 this semester will be very similar to Fall 2021, with some reordering to smooth out the pacing of the course. The lectures, lecture notes, SREs, problem sets, and section notes will capture all of the content that you are expected to learn during the

[4] Based on text in the Harvard CS50 Syllabus.

semester, but we recommend the following textbooks for supplementary reading and practice exercises:

1. Background (to review or fill in material from CS 20 and CS 50)
    a. Harry Lewis and Rachel Zax. Essential Discrete Mathematics for Computer Science.
    b. CS50 OpenCourseWare and the Python documentation.
2. Algorithms (approx. the first two-thirds of the course).
    a. Tim Roughgarden. Algorithms Illuminated, especially Volumes I & II.
    b. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to Algorithms, Fourth Edition.
3. Limits of Algorithms (approx. last third of the course)
    a. John MacCormick. What can be Computed?
    b. Michael Sipser. Introduction to the Theory of Computation.