CS1570

Claire Mathieu Manas Korimilli

Fall 2026

Classes	1pm–2:20pm on Tuesdays and Thursdays.
Location	CIT 477
Course Website	`https://csci1570.github.io/`
Professor's Office Hours	Tuesdays 11am-12pm and Thursdays 9:30am-10:30am in CIT 419
Prerequisites	CSCI 0500 or CSCI 1010 or minimum score of WAIVE in 'Graduate Student PreReq'.

Objective. The design and analysis of algorithms is one of the foundational subjects of Computer science. You will learn to go through a sequence of steps:

1. What question do I want to solve?
2. How do I solve it?
3. How good is my solution?

For the first question, decades of experience have enabled scientists to distill a clean set of core questions that are representative of some pervasive issues arising in Computer science. We will go over a large collection of those standard problems. You are welcome to propose some problems, and together we can try to see if they are suitable for an algorithmic approach.

For the second question, the development of Computer science has shown that a few paradigms are necessary in the culture of all computer scientists. You have probably already been exposed to them (greedy, divide and conquer, dynamic programming, randomization): in this course we will go over those rigorously. Some more advanced techniques have been developed for slightly more specialized applications, and we will explore a few of those as well.

The third question has one standard answer (prove that the algorithm is correct and that its worst-case runtime is a low-degree polynomial in the size of the input), but it actually deserves a lot more scrutiny and there are many nuances, many criteria that can be used to define a "good" solution.

In this course, we prove theorems about the efficiency of the solutions presented. It requires enough mathematical maturity that you are able to read a short proof and decide whether or not the proof is correct. If you are uncertain about that, you will suffer. It requires enough programming maturity

that, given an algorithm, you can see how, given enough time, you could write a computer program implementing the algorithm. Although the course has no implementation, the runtime analyses require the ability to look at high-level pseudocode and imagine, if you were to implement a line of pseudocode with a short block of code, how efficient that block of code would be. The implementation is not part of the course but is always lurking as the main motivation underneath all the questions we study.

The teaching style is interactive. During lectures, together we will pose questions and explore possible paths towards solutions – including deadends, sometimes. That is part of the creative process. Sometimes we will cover less material than I would have liked, because we explore some side paths, but it is part of the process of learning how to have ideas. For that, class attendance and active participation are critical. To help you focus on the lecture and avoid the temptations and distractions of whatever is on your computer screen, I encourage you to, instead of using a laptop, making do with pen and paper. (Perhaps afterwards you can clean up your handwritten notes with an app to translate them into text files).

At the end of the course, you will minimally have an understanding of a broad set of algorithms and analysis techniques, and have some basic toolkits that should be part of the knowledge of any person with a degree in computer science. Ideally, you will have begun to develop the ability to take a question, formulate some version of it as a formal algorithmic problem, try out some paradigms of algorithmic design and choose a reasonable way to go about solving it, propose an algorithm to solve the problem, and have some tools to try to understand mathematically if your algorithm is any "good": you will have become a problem-solver!

I would also like it if you developed some presentation skills: having figured out, roughly, how to do something, what is the "right" way to communicate about it?

lecture	day	date	subject
1	Th	09/10	Part I
2	Tu	09/15	
3	Th	09/17	
4	Tu	09/22	
5	Th	09/24	
6	Tu	09/29	
7	Th	10/01	
8	Tu	10/06	Part II (Tested on Midterm 2)
9	Th	10/08	Midterm 1
10	Tu	10/13	
11	Th	10/15	
12	Tu	10/20	
13	Th	10/22	
14	Tu	10/27	
15	Th	10/29	
-	Tu	11/03	No Class (Election day)
16	Th	11/05	Midterm 2
17	Tu	11/10	Part III
18	Th	11/12	
19	Tu	11/17	
20	Th	11/19	
21	Tu	11/24	
-	Th	11/26	No Class (Thanksgiving)
22	Tu	12/01	
23	Th	12/03	
24	Tu	12/08	Course Recap (Reading Period)
	Th	12/10	No Class (Reading Period)
	Mon	12/14	Final Exam

Outline. Part I reviews four basic methods to design algorithms: greedy algorithms (with proofs of correctness); divide and conquer (with runtime analysis); dynamic programming; random sampling.

Part II goes over some more advanced algorithmic gems: online data structures such as union-find (with amortized analysis); text compression (Lempel-Ziv etc.); bipartite matchings, flows, and linear programming (simplex); Markov Chain Monte Carlo methods (without runtime analysis).

Part III, time allowing, addresses selected topics such as: clustering; beyond worst case models; approximation algorithms; energy minimization; stable matching; etc.

Please let me know during the first couple of weeks if there is excessive redundancy between this course and some material covered in courses that you have already taken, or conversely, if I am assuming knowledge that many of you do not have. In that case, I will then revise the syllabus accordingly.

Textbooks. Here are a few books that I will occasionally use for the course. None of them are required.

- Algorithm Design by Jon Kleinberg and Eva Tardos. Pros: it is particularly well thought out for combinatorial optimization algorithms. It likes to introduce problems by telling stories that provide at least a semblance of motivation and help the reader understand the point of what is being studied. It has lots of exercises, of varying degrees of difficulty, and some of them are amusing. Cons: fat and expensive; some of the proofs are, I believe, longer than they need be. Sometimes it feels a bit wordy. Overall: I enjoy it, but it is not perfect.

- Algorithms by Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani. Pros: it is relatively short and inexpensive. It has an interesting selection of problems, exhibiting some taste. The proofs are often elegant, highlighting the clever ideas and skipping the tedious details. I love the style of the book. For the reader who is at ease with the material, looking at it is a good way to get a quick refresher on the subject presented. Cons: some standard topics are omitted; in particular it is somewhat weak on combinatorial optimization. More importantly, there are bugs. I have not seen any fatal flaws, but sometimes elegance has come at the price of correctness, with some (easy) special cases forgotten, and other such silly mistakes (the kind of mistakes that an LLM would catch.) So in order to use it for more than getting a hint or reminder, the reader needs the ability to notice and correct the annoying little mistakes. Overall: I enjoy it, but it is not perfect.

- Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Pros: the standard algorithms textbook for decades at this point. Everybody who has a CS degree knows about it. The sequence of editions has cleaned up the writeup and eliminated the bugs. It is very complete. The algorithms are presented in enough detail that implementation ought to be relatively easy. Cons: big and expensive. The proofs are complete but a bit too detailed for my taste; I find them a bit wordy, although the insecure reader might be more comfortable going through the details line by line. Similarly, I don't like the pseudocode used to present the algorithms, which I feel is overly detailed, so that it obscures the elegance of the underlying ideas. Overall: I don't particularly enjoy it, but I find that I keep going back to it as the default reference on the subject.

- Algorithms Illuminated: Omnibus Edition by Tim Roughgarden. I have not yet taught with this book, but have heard good things about it. It is relevant to a significant portion of the course.

Grading. The course grade will be obtained from a weighted average of the final exam (40%), the two midterms (25% each), and the homeworks (10%).

There will be about 9 homeworks, roughly one per week. Each homework is given a grade of 100% if something is turned in by the deadline and 0% otherwise, irrespective of the content of the homework. Thus using AI has no effect on the homework grades. We will still provide feedback on your homework as if it were an exam.

Homeworks require mental effort. It is hard to make yourself do it when there are other demands on your time. But for most people it is much easier to get to work if they are not alone doing it. For that reason, we ask that (unless you absolutely are allergic to the idea) you do the homeworks in pairs after shopping period. Each homework after shopping period will be done by two students working together and the pair will turn in a single homework bearing both of their names. Homework 1 and Homework 2 will be done individually.

Homeworks must be typeset in LaTeX and submitted on Gradescope by 10:00pm (Eastern Time) on the due date.

The two midterms and the final are in class, closed book. However, to alleviate the stress of not having access to course material, each student is allowed a 1-page handwritten summary of material of their choice. Preparing that page is also a chance for each student to review the course content and select what seems most important.

The problems on the midterms and final will be mostly taken from the homework problems, but at least 1 in each midterm/final will be more challenging.

Each of the homeworks will have a challenging bonus problems for students to try, but these bonus problems are guaranteed not to show up on the midterms and final.

Interactions. My preferred mode of interaction is in person. I will hold office hours and wish for each of you to come and introduce yourself to me at one of my office hours between the beginning of the semester and the first midterm. The HTA (Manas) and TA (Sunny) will be holding office hours every week. The course Ed Discussion is also available to ask about the course content or homeworks. If you have any issues or concerns, do not hesitate to reach out to me or to the TAs. If you have any issues or concerns, do not hesitate to reach out to me or to the TAs.

AI. This is a **no AI** course.

If you wish to use AI outside class for help with the content of the lectures or out of curiosity, to explore related questions, or for any other reason, you are welcome to use it if it helps you. For example, you can use AI as a tool to re-explain content from lectures. You are not allowed to use AI for the homeworks (or the midterms, or the final exam). No one wants to grade something that has been written by AI. It would be a waste of the grader's time. We want to see what you are able to come up with with your own brain, not with some combination of you and an AI agent.

Why does this matter, when, at this point of development of LLMs, any algorithmic skills that you might have developed by the end of the course will

most likely be superseded by an LLM's ability to spew out a solution to any of the algorithmic problems that we will discuss in the course? Well, this is about your education. Problem-solving is a necessary skill in technical fields. Later, when you interact with an LLM and it gives you some algorithmic solution, you need to be able to understand it and to verify for yourself that it is correct; and, should the LLM somehow become polluted by some nefarious agents and start giving you bad solutions, you need to realize it and to have the knowledge to sidestep it. Otherwise you become dependent on the AI. If you want to stay the one in control and use the AI as an assistant, you need to have a basic understanding of what is going on. That is part of what your education is about, and if *Brown* students do not do it, who will?

Additionally, some of you might want to become researchers in the future. Currently, an increasing collection of open problems from theoretical computer science are being solved by AI, but many of those solutions are found in a dialogue between human and computer, during which the human prods and steers the conversation. That requires maturity on the part of the human: experienced researchers are the ones that are best able to take advantage of the LLM's capabilities.

Moreover, recent research about learning and the impact of AI shows that the use of AI by students boosts their grades while they are using AI to assist them, but reduces their performance once they are asked to solve a problem by themselves: students who use AI for their homeworks seem to ultimately learn less.

Of course, knowing how to use AI, to understand its potential and limitations in the context of the design and analysis of algorithms, and to work with them, also has some value, but that would be for another course. Not this course.

Expected Time Commitment Each week, in addition to 2 hours 40 minutes in class (for a total of 24 lectures), students will spend on average as much time on reading the material. Students can expect to spend 8 hours on each of the 9 homework assignments. Each of the two midterms will last 1 hour and 20 minutes, and the final will last 2.5 hours. Preparation for each exam should take 10 hours. This translates to a minimum of 180 hours of work throughout the semester, in line with Brown University policy for 1-credit courses.

Accommodations Brown University is committed to full inclusion of all students. Please inform me early in the term if you may require accommodations or modification of any of course procedures. You may speak with me after class, during office hours, or by appointment. If you need accommodations around online learning or in classroom accommodations, please be sure to reach out to Student Accessibility Services (SAS) for their assistance (sas@brown.edu, 401-863-9588). If you encounter any digital materials in this course that are inaccessible, complete the Digital Accessibility Concern Reporting Form. For any accessibility or accommodation concerns, please contact the ADA/504 Coordinator at ada_504@brown.edu. Undergraduates in need of short-term academic

advice or support can contact an academic dean in the College by emailing college@brown.edu. Graduate students may contact one of the deans in the Graduate School by emailing graduate_school@brown.edu.

Diversity and Inclusion The course intends to provide a welcoming environment for all students. We especially welcome diverse ideas and perspectives during class discussions.

All members of the CS community, including faculty and staff, are expected to treat one another professionally. Toward this goal, TAs have undergone training in diversity and inclusion. However, despite our best efforts, we may accidentally slip up, so please feel free to speak to any member of the course staff with any concerns you have during the semester, and do not hesitate to contact Claire directly. We will take your concerns very seriously.

Incompletes In extraordinary situations, students may request to receive an incomplete grade. Such cases must be accompanied by a Dean's note and will be assessed case by case.

Student Well-Being Being a student can be very stressful. If you feel you are under too much pressure or there are psychological issues that are keeping you from performing well at Brown, we encourage you to contact Brown's Counseling and Psychological Services (CAPS). They provide confidential counseling. You may also find these resources from Brown Health and Wellness (and links therein) useful.