CSCI 3104: Algorithms
Spring 2020

	Section 100	Section 200
Instructor:	Lijun Chen	Joshua Grochow
Lectures:	TTh 9:30-1045a in GOLD A2B70	MWF 1-1:50p in BESC 180
Email:	lijun.chen@colorado.edu	jgrochow@colorado.edu
Office hours:	T 1:00-3:00pm in ECCR 1B11, or by appointment	M 2:00-3:00pm and W 4:00-5:00pm in ECES 118, or by appointment
TAs:	Robby Green, Brian Groenke, Isabella Huang, Marcus Hughes, Michael Levet, Zhiyuan Liu, Amatullah Sethji, Yichen Wang, Wanshan Yang	
Recitations:	See Canvas page	
CAs:	Nicholas Auer, Alex Book, Nicole Dong, Alici Edwards, Luke Ingalls, Ian Jorquera, +1 TBA	
CA office hours:	see Canvas page	

Prerequisites: data structures (CSCI 2270), discrete structures (CSCI 2824), calculus I & II, and programing experience in a language such as C, C++, Python, or Java

Recommended (not required) text: *Algorithm Design* by Kleinberg & Tardos (any edition)

Course objectives:
- Become familiar with standard algorithms and their pros and cons in practice
- Learn to prove an algorithm's correctness and time and space complexity
- Practice adapting and combining algorithms to solve real problems
- Learn strategies for designing new algorithms for emerging applications
- Communicate clearly about algorithm design, correctness, and complexity analysis

Course structure and grades

This course uses a standards-based or mastery-based grading scheme. For a brief introduction to standards-based grading, see, e.g., the first 6 minutes of https://www.youtube.com/watch?v=G587eEZjRJA.

1. There are 24 content standards (listed below) and 1 participation standard.
2. Within each content standard, you will have several opportunities (at least 4) to demonstrate that you have mastered that standard. To have mastered a standard, you must demonstrate mastery **twice**.
3. You earn points towards your participation standard by doing the online mini-quizzes, and by attending recitation. If you earn enough points, you gain “mastery” in the participation standard.
4. Your final grade is determined by how many standards you master, as follows:

Minimum Standards mastered:	21	20	19	18	17	16	15	14	13	12	11	10-
Letter grade:	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F

Your answer to each question on an assignment will be given a rating between 0 and 4:

0: No meaningful attempt made
1: Attempt made, but not yet really understanding
2: Near mastery. Just missing a detail or two.
3: Mastery.
4: Perfection.

The numerical values are ***only for feedback*** and have ***no effect on your grade***. ***The only relevant thing from a grade point of view is whether you got at least a 3 (mastery), or at most a 2 (not mastery).***

Weekly assignments. Roughly each week (excepting spring break and midterm weeks), you will have

- An online mini-quiz
- An in-recitation quiz (starting in week 3)
- A problem set

The online mini-quizzes are designed to be quick and easy, to give you immediate feedback, to see that you are following along with the basic material. The in-recitation quiz and the problem set are both opportunities to demonstrate mastery of one or more standards. On the problem set, we will not say specifically which problems go with which standards (though you can probably guess pretty accurately). It is possible that a single problem could count towards more than one standard. On quizzes and exams, each problem will be clearly marked with which standard it corresponds to.

Exams. There are three exams

- Midterm 1, in week 8 (exact date, time, location TBA). Covers material through week 6, standards 1-14
- Midterm 2, in week 13 (exact date, time, location TBA). Covers material through week 12, standards 15-24.
- Final exam, during finals week. Covers all material.

On the exams, questions will be clearly marked with which standard they correspond to. If you have already demonstrated mastery twice in some standard, you do not need to attempt questions relating to that standard again.

Overall, you thus have 4 default opportunities to demonstrate mastery in each standard:

1. On a problem set
2. On an in-recitation quiz
3. On a midterm
4. On the final

Office hours requizzes. These provide you potential additional opportunities to demonstrate mastery in a given standard. They are called “requizzes” because they are requizzing a standard that you have already been quizzed on, but the questions will be all new.

To earn the opportunity to do an OH requiz, you must first do ***corrections & reflections*** on a quiz in which you previously did not demonstrate mastery. This means you take your original quiz answers, show how to change them to the correct answer (the “corrections” part), and then discuss, in plain English, full sentences, what led you to the wrong answer on the original quiz. For example, what do you know now that you didn’t when you took the quiz? If you were confused about something when you took the quiz, in what way were you confused, and how do you now understand the topic? Etc. Merely saying “I put the wrong answer” is not sufficient.

There are several rules that must be followed around the logistics for OH requizzes:

1. To earn the opportunity for an OH requiz, you must do corrections & reflections on a quiz on the same standard you want to be requizzed on.
2. You can only take up to three (3) OH requizzes. (You may requiz the same standard more than once, but those will count towards your three.)

3. **Requiz requests must be made at least 2 weeks in advance.** As the semester progresses and we see how many such requests we are getting, **we may make this advanced notice longer or shorter,** which we will announce in class and on Canvas.

Standards

1. Loop invariants / proof by induction
2. Asymptotic notation
3. Asymptotic analysis: correctly writing down equation for counting operations given pseudocode w/ simple independent loops
4. Asymptotic analysis: correctly writing down equation for counting operations given pseudocode w/ nested dependent loops
5. Recurrence relation - write down correct recurrence for recursive algorithm
6. Recurrence relation – solve by plugging in
7. Recurrence relation – solve by tree method
8. Divide & conquer: principles (when does it apply vs not)
9. QuickSort: expected vs worst-case behavior
10. Hash tables: load factor, when to apply vs balanced binary tree
11. Greedy: exchange argument for correctness (mastery can be demonstrated either using interval scheduling or Huffman coding)
12. MST: safe & useless edges
13. MST generic algorithm (for mastery, must also master at least one of Kruskal's or Prim's)
14. Shortest paths: breadth-first search for unweighted
15. Shortest paths: generic weighted algorithm (for mastery must master at least one of Dijkstra or Bellman-Ford)
16. Max flow: residual graph
17. Max flow: applying max-flow (/reducing to max-flow, e.g. bipartite matching)
18. Dynamic programming: principles (when does it apply vs not)
19. Dynamic programming: write down recurrence for computing local solution
20. Dynamic programming: template / order of filling in the table
21. Dynamic programming: backtracking to find solutions
22. Dynamic programming: using DP for counting
23. NP-completeness: basic definitions (P, NP, NP-hard, NP-complete, reduction)
24. NP-completeness: reductions
25. Participation

Schedule of topics

Subject to change – which we will notify you of - but this should be essentially correct.

Week 1: Proofs & loop invariants
Week 2: Asymptotic analysis, start divide & conquer w/ MergeSort
Week 3: Divide & conquer, recurrence relations, start QuickSort
Week 4: Finish QuickSort. Hash tables.
Week 5: Greedy 1: Interval scheduling, Huffman coding
Week 6: Greedy 2: Breadth-first search, greedy shortest paths, start Min Spanning Tree
Week 7: Greedy 3: Max flow/min cut
Week 8: Review + midterm (covering through Week 6)
Week 9: Dynamic programming 1: Introduction and starting examples
Week 10: Dynamic programming 2: Bellman-Ford
Spring Break
Week 11: Dynamic programming 3: string alignment
Week 12: P versus NP: definitions, reductions
Week 13: Review + midterm
Week 14: P versus NP & approximation algorithms

Week 15: Special topic + review

Problem sets

The course staff will evaluate more than 10,000 pages of homework this semester. For this to run smoothly, we need your help.

- Problem sets are due on Friday at 11:55p, via the class Canvas page
- No credit will be given for assignment submitted in any other way (e.g., email) or after the deadline
- Along with the problem set, we will release an answer template in PDF and LATEX formats. All submissions must strictly conform to this template. We rely on Gradescope to help us with the logistics of grading, and answers that are not in the specified regions may not be considered. If you need more space for an answer, that is fine and will be graded as usual - please attach a separate sheet to that problem.
- The answer templates have a space for you name and ID. If your name and ID are not in this box, you will not get credit.
- Solutions must be detailed and clear. The clearer your explanation, the more likely you are to receive full credit for a correct answer.

Programming assignments (of which there are only a few)

- We *strongly recommend* you use Python – as it lets you focus more on the algorithm and less on language details like pointers, classes, etc. – but you are welcome to use C, C++, or Java instead.
- Your code must be runnable and readable (good variable names, formatted, commented)
- You must submit source files (e.g., file.c, file.java, file.py, etc) in addition to your solutions PDF, and it must be runnable/compilable.
- Unless specifically allowed, all parts of all algorithms and data structures must be implemented from scratch (that is, no libraries; if you use Python or another modern language, be sure you are not accidentally invoking non-trivial libraries; garbage collection features and static arrays are okay; “dictionary” data structures are not). If you are unsure about what counts as “non-trivial,” ask for clarification.

Rules for formatting and submitting your work

1. On-time rule: Late submission are not accepted.
2. Canvas rule: All submissions must be made through the class Canvas page.
3. One-file rule: Your solutions to the problem set must be a single PDF file that matches template. **The only exception to this is source code files, which should be submitted as separate text files formatted for the appropriate programming language.**
4. Name rule: Your name and ID must be in the specified box at the top of page 1
5. Figure rule: All figures, graphs, charts, and tables must be labeled correctly. No credit for figures with unlabeled axes or unlabeled data series.
6. Source-code rule: For problem sets with programming portions, you must submit your source code, and it must be runnable/compilable.
7. Filename rule: Name your file as Lastname-Firstname-MMDD-PSX.pdf
    - Name: last name then first name (Lastname-Firstname)
    - Birthday: 2-digit month and 2-digit day (MMDD)
    - Problem set: PS then the problem set you are submitting solutions for (X)
    - Source files must also conform to this pattern

   For example, Prof. Grochow’s solutions for problem set one would be Grochow-Joshua-0314-PS1.pdf and his Python source code would be Grochow-Joshua-0314-PS1.py

What to write

- Answer the right question.
- Justify your answer. Unless the problem specifically says otherwise, **every homework problem requires an explanation.** Without one, even a perfectly correct solution is worth nothing. In particular, the sentence "It's obvious!" is not an explanation— "obvious" is often a synonym for "false"!
- Answer the question completely. When a problem says to give an algorithm, you must do several things to receive full credit.
  - If necessary, formally restate the problem to make it clear exactly what problem you are solving.
  - Give a concise pseudocode description of your algorithm.
  - Explain what your pseudocode does.
  - Justify the correctness of your algorithm.
  - Describe the correct algorithm.
  - Analyze your algorithm's running time. This may be as simple as "There are two nested loops from 1 to n, so the running time is $O(n_2)$." Or, it may require setting up and solving a summation or a recurrence, in which case you will need to prove your answer is correct.
  - Describe the fastest algorithm you can think of, even if the problem does not include the words "fast" or "efficient." Very slow or brute force algorithms usually do not demonstrate mastery of a new algorithmic technique. Not every problem will tell you what to aim for (figuring that out is part of what you're learning).

Some problems may deviate from these default requirements. For example, you may be asked to give an algorithm that uses a particular approach, even though another approach is more efficient. (Answer the right question!) Some problems may ask you to analyze the space used by your algorithm in addition to the running time.

Advice for writing up your solutions

The most important thing you can do is to make it easy for the grading staff to figure out what you mean within the short time they have to grade your solution. Solutions that are difficult to read or understand make it harder for us to see whether you have mastered a standard.

- **Write carefully**. You will only be graded on what you write, not what you mean. We cannot read your mind. If your solution is ambiguous, we will choose the interpretation that makes the solution wrong.
- **Write clearly.** If we cannot decipher your answer quickly, it is nearly impossible for us to give you credit for it, even if it is correct. Write your answers clearly. Separate them from other problems. Box your results. Solve problems in the order they are listed. (If you have poor handwriting, try typesetting your solutions with LATEX, and don't submit your first draft.)
- **Explain your solution**. Unless the problem specifies it, giving pseudocode alone, with no accompanying verbal explanation, is insufficient. A complete solution is a verbal explanation of the algorithm's generic behavior with supporting pseudocode (or something functionally equivalent). Pseudocode provides precision where English does not. English provides context and interpretability where pseudocode does not.
- **Give generic solutions, not examples**. Solutions that include phrases like "and so on", "etc.", "do this for all X", or ". . ." typically indicate to us a lack of understanding. Those phrases indicate precisely where you should have used iteration, recursion, or induction but did not. If your solution does not work on every valid input, then it is not a correct solution. A complete solution explains why there is no such input on which your algorithm fails.
- **Make it understandable**. Start by describing the key ideas or insights behind your solution before going into its details. The reader should understand your approach almost immediately without looking at the proofs or pseudocode. A clearly written solution that includes all the main ideas but omits some low-level might demonstrate mastery, whereas a complete, correct, but opaque solution may not. If you cannot communicate the ideas clearly, we cannot tell whether you have really mastered them.
- **Be succinct.** Your solution should be long enough to convey exactly why your answer is correct, yet short enough to be easily understood.
- **Numerical experiments:** Some programming problems will require you to conduct numerical experiments. For instance, to show that an algorithm takes O(n log n) time, you will need to measure the number of atomic operations at multiple values of n, plot the measured values versus n, and then plot the asymptotic function showing that the function matches the data. Plotting the average number of

operations for a given value of n will almost always improve your results. To get a good trend, we recommend using a dozen or so exponentially spaced values of n, e.g., n = {2_4, 2_5, . . . , 2_{16}, . . .}. When presenting your results, you must explain your experimental design.

Getting help

- Attend the lectures, attend your recitation section, and come to office hours. These are provided specifically to help you learn the material. Take advantage of them.
- Use the Piazza forum. For class discussion and Q&A. Rather than emailing questions to the teaching staff, please post your questions on our Piazza forum.
- Do not abuse email. Do not ask for help with specific parts of the problem sets via email. If every student in class sent us 1 email a day, and it took only 5 minutes to respond to each, the course staff would be spending over 33 hours a *day* (!) just responding to email. Email should be reserved for high-priority issues only.

Course policies

Working in teams

- All problem sets must be completed individually.
- We encourage you to form study groups to discuss the problems with each other.
- Each student must independently write up their own solution—the one they submit to demonstrate mastery for themselves. Working at a whiteboard or on paper together is fine, even if you solve the problem that way, but writing up a joint solution is not.
- Examinations must be completed individually.

Intellectual Honesty and Plagiarism

- Intellectual dishonesty or plagiarism of any form, at any level, will not be tolerated.
- **Discussing problems with other students is encouraged, but** you must list your collaboration on the page where you give the solution. If you discussed it with 20 other people, then all 20 names should appear in your solution. If someone was particularly helpful, say so. Be generous; if you're not sure whether someone should be included in your list of collaborators, include them. For discussions in class, in section, or in office hours, where collecting names is impractical, it's okay to write something like "discussions in class." There is no penalty for discussing problems with other students.
- **Copying from any source in any way is strictly forbidden.** This includes both the Web and other students (past or present). If you are unsure about whether something is permitted, please see me before the assignment is due.

  There will be a zero-tolerance policy to violations of this policy. Violators will be removed from the class, given a grade of F, and reported to the University Honor Council.

- **Write everything in your own words and cite all outside resources.** You are strongly encouraged to use outside resources, but you must write your solutions yourself. We are not interested in seeing Wikipedia's or anyone else's solution. The only sources you are not required to cite are the textbook and lecture notes, and the prerequisite material. As a small bonus for having read the syllabus carefully, I will award 1 participation point if you send me an email containing a photo of a dinosaur (any kind is fine).

Regrade requests

On Gradescope, there is a "regrade" button, but this should ***only*** be used if you received a zero because you received "no meaningful attempt was made" on a given question (or part of a question), and yet you did make a meaningful attempt. Regrade requests through Gradescope for any other purpose will receive a pointer to this policy and then marked resolved.

On the other hand, if you want the *logic* of your answer regraded, you must email one of the professors with a clear explanation of what your solution was, what rating it received, and why you think it deserved a higher rating. In that situation, we reserve the right to personally regrade your entire problem set, and the rating on each problem might go up or go down.

Exceptional circumstances

In the case of exceptional circumstances, e.g., documented illness or injury that preclude submitting an assignment on time, we have several possible remedies, depending on the situation. Examples of ***unexceptional*** circumstances include registering late, travel for job interviews or conferences or fun, forgetting the homework deadline, or simply not finishing on time.

If it is a homework and the homework can still be submitted before solutions are released, then we may choose to count it as though it were on time. If that does not apply, we may provide you the opportunity for an additional OH requiz (not counting towards your limit of 3) on the relevant standards.

Suggestions

Suggestions for improvement are welcome at any time. Any concern about the course should be brought first to our attention. Further recourse is available through the office of the Department Chair or one of the Academic Advisors, all accessible on the 7th floor of the Engineering Center Office Tower.

Honor Code

As members of the CU academic community, we are all bound by the CU Honor Code. I take the Honor Code very seriously, and I expect that you will, too. Any significant violation will result in a failing grade for the course and will be reported. Here is the University's statement about the matter:

All students enrolled in a University of Colorado Boulder course are responsible for knowing and adhering to the Honor Code. Violations of the policy may include: plagiarism, cheating, fabrication, lying, bribery, threat, unauthorized access to academic materials, clicker fraud, submitting the same or similar work in more than one course without permission from all course instructors involved, and aiding academic dishonesty. All incidents of academic misconduct will be reported to the Honor Code (honor@colorado.edu); 303-492-5550). Students found responsible for violating the academic integrity policy will be subject to nonacademic sanctions from the Honor Code as well as academic sanctions from the faculty member. Additional information regarding the Honor Code academic integrity policy can be found at the Honor Code Office website.

Special Accommodations

If you qualify for accommodations because of a disability, please submit your accommodation letter from Disability Services to your faculty member in a timely manner so that your needs can be addressed. Disability Services determines accommodations based on documented disabilities in the academic environment. Information on requesting accommodations is located on the Disability Services website. Contact Disability Services at 303-492-8671 or dsinfo@colorado.edu for further assistance. If you have a temporary medical condition or injury, see Temporary Medical Conditions under the Students tab on the Disability Services website.

In order for us to have time to arrange logistics, accommodation letters from Disability Services must be submitted at least two weeks in advance of the needed accommodation (or within the first two weeks of the semester).

Campus policy regarding religious observances requires that faculty make every effort to deal reasonably and fairly with all students who, because of religious obligations, have conflicts with scheduled exams, assignments or required attendance. In this class, I will make reasonable efforts to accommodate such needs if you notify me of their specific nature by the end of the 3rd week of class. See full details at http://www.colorado.edu/policies/observance-religious-holidays-and-absences-classes-andor-exams

Classroom Behavior

Students and faculty each have responsibility for maintaining an appropriate learning environment. Those who fail to adhere to such behavioral standards may be subject to discipline. Professional courtesy and sensitivity are especially important with respect to individuals and topics dealing with race, color, national origin, sex, pregnancy, age, disability, creed, religion, sexual orientation, gender identity, gender expression, veteran status, political affiliation or political philosophy. For more information, see the policies on classroom behavior and the Student Code of Conduct.

CU Boulder recognizes that students' legal information doesn't always align with how they identify. Students may update their preferred names and pronouns via the student portal; those preferred names and pronouns are listed on instructors' class rosters. In the absence of such updates, the name that appears on the class roster is the student's legal name. Class rosters are provided to the instructor with the student's legal name. I will gladly honor your request to address you by an alternate name or gender pronoun. Please advise me of this preference early in the semester so that we may make appropriate changes to our records. See policies at http://www.colorado.edu/policies/classbehavior.html and at http://www.colorado.edu/studentaffairs/judicialaffairs/code.html#student code

Discrimination and Harassment

The University of Colorado Boulder (CU Boulder) is committed to fostering a positive and welcoming learning, working, and living environment. CU Boulder will not tolerate acts of sexual misconduct, intimate partner abuse (including dating or domestic violence), stalking, or protected-class discrimination or harassment by members of our community. Individuals who believe they have been subject to misconduct or retaliatory actions for reporting a concern should contact the Office of Institutional Equity and Compliance (OIEC) at 303-492-2127 or cureport@colorado.edu. Information about the OIEC, university policies, anonymous reporting, and the campus resources can be found on the OIEC website.

Please know that faculty and instructors have a responsibility to inform OIEC when made aware of incidents of sexual misconduct, discrimination, harassment and/or related retaliation, to ensure that individuals impacted receive information about options for reporting and support resources.