CS 252: Algorithms — Layla Oesper
Midterm 2 Review Sheet — Spring 2024

Important Details: Exam 1 will occur on Monday May 6, 2024, 11:10am - 12:20pm in CMC 210. This is a closed note and closed book exam.

Differences Between Exams and HW

You only have 70 minutes to complete an in class exam. As such, there will be some important differences between both the types of questions on an exam and what criteria is used for evaluation. In particular, the following are important to remember.

- You should expect *conceptual questions* that ask you to demonstrate your understanding of the underlying concepts we have been discussing. In particular, you should be prepared to:
    - Demonstrate understanding of algorithms (and basic underlying concepts and definitions) we have discussed by answering questions about them or applying them to novel data.
    - Answer questions that ask you to apply concepts or describe rationale behind them.
    - Provide basic analysis (running time, identify issues, etc.) of approaches similar, but not identical ones discussed previously.
    - Sketch out why you think statements related to approaches covered in class, or slight modifications to these approaches, are true or provide counterexamples that show them to be false.
- Remember that **you can always give me a brute force or naive algorithm** for any question I ask. Faster algorithms will get more points, but substantial points will be awarded for correct, but slow approaches.
- Presentation of your algorithm or ideas is less essential as you don't have time to write and revise multiple times. Pay attention to wording on questions that ask for "Give me the high level idea", rather than thinking that you need to give a fully fleshed out proof.

How can I study for the exam? I suggest starting early rather than trying to cram at the last minute. Here are a few suggestions on how you might most effectively use your study time.

- Go back through the problem set problems. If you missed points on something, check out the posted solutions and make sure you understand the problem.
- Go back through class notes/worksheets and re-work problems and examples from class. Pay special attention to anything that was particularly confusing at the time. In particular, I encourage you to first make an attempt at solving questions without looking at the solutions. This will make your practice better resemble what you'll need to do on the exam.
- Try as many problems from the book as you can. In particular, I suggest working through the "Solved Problems" before reading the solutions.
- Re-read any sections of the book that pertain to material you find confusing. The book does a great job of explaining many concepts. Now that you've tried your hand at some things, the reading may make more sense the second time through.
- Form a study group and go through problems together.
- Come to Layla's student hours or a prefect session!

Material on the Exam

The exam will focus on material covered in the readings, in class and on the problem sets since the last exam, but previous concepts that are foundational to the entire class are fair game. For example, we won't ask you about stable-matching but we may ask you to prove that certain functions or pieces of code have certain constraints on their running time (O, Ω, Θ).

The following table gives a more specific breakdown of both topics you should be familiar with, but also some tasks that you should feel comfortable doing on the exam. Note, I will do my best to stick to this on the exam, but it is **not a contract** and I can't guarantee that this is an exhaustive list of everything on the exam.

Topic	You should know or be able to ...
For any algorithm discussed ...	(1) Apply the algorithm to a provided input. (2) Analyze a proof of its correctness, including filling in missing pieces. (3) Pick appropriate data structures and analyze the resulting runtime of the implementation. (4) Determine how to apply the algorithm to similar, but not identical problems. (5) Know what problem the algorithms solves. (6) Explain the effect of small modifications to an algorithm or its implementation.
Greedy Algorithms	(1) Interval scheduling problems and algorithms. (2) Identify possible greedy approaches to solve a stated problem. (3) Provide counterexamples that show a stated greedy approach doesn't solve a stated problem. (4) Be able to explain and apply a Greedy stays-ahead argument.
Greedy Graph Algorithms	(1) Dijkstra's Algorithm. (2) Minimum spanning tree algorithms: Reverse-Delete, Kruskal, and Prim. (3) Understand and be able to apply the Cut-Rule and the Cycle-Rule. (4) Union-Find data-structure - know its operations and running times and explain how it might be useful for certain problems. (5) Understand the impact of assumptions to the shortest-path or minimum spanning tree problems can impact the stated output.
Foundational Material	(1) Know the definitions of Θ, Ω and O. (2) Be able to prove basic statements (or provide counterexamples) about the runtime of functions. (3) Order the growth rate of different functions. (4) Identify the worst-case running time of pieces of psuedo-code. (5) Know the basic operations and running times of standard data structures used in class, including heaps and priority queues.