COSC-311 Fall 2018 Midterm 2 Topics

This is *not* a comprehensive study guide. There may be topics that we have discussed in class or that have come up on homework that are not on this list. You are responsible for all of the course material up to this point, including both in-class material and homework.

1. Divide-and-Conquer paradigm
   (a) Structure: three components to a divide-and-conquer algorithm
   (b) How to write a recurrence for the runtime of a divide-and-conquer algorithm
   (c) Solving recurrences using the Master Theorem

   (a) Problems and algorithms we've seen:
   - Mergesort
   - Quicksort
   - Largest sum subsequence
   - Probing binary trees (on HW2)
   - Database queries (on HW2)

2. Greedy paradigm
   (a) When to use it: greedy choice property and optimal substructure property
   (b) Problems and algorithms we've seen:
   - Making change (with standard coin denominations); choose largest denomination
   - Interval scheduling; earliest finish time
   - Single source shortest paths (with positive edge weights); Dijkstra's algorithm
   - Minimum spanning tree; Prim's algorithm and Kruskal's algorithm
   - Scheduling to minimize response time (on HW3)
   - Fractional knapsack (on HW3)

3. Dynamic programming
   (a) When to use it: optimal substructure property and inability to make a greedy choice that will lead to an optimal solution, overlapping subproblems
   (b) Two approaches: fill in an array of subproblem solutions from the bottom up, or top-down with memoization
   (c) Problems and algorithms we've seen:
   - Making change (with nonstandard coin denominations)
   - Weighted interval scheduling
   - Single source shortest paths (with negative edge weights); Bellman-Ford algorithm
   - Rod cutting (on HW3)

4. For each algorithm, know:

    - How the algorithm works (i.e., be able to run it by hand)
    - Why the algorithm works (I won't ask you to prove/argue about correctness on the exam, but I do expect you to understand why these algorithms do what they do)
    - Its asymptotic runtime and where the runtime analysis comes from (we haven't gone over the runtime analysis for every single algorithm in class, but I expect you to be able to derive runtimes)

FAQ

Q: What's the level of difficulty of the exam questions, relative to homework?

A: The exam questions will be closer to Mini HW than to HW. You have two weeks to do each homework assignment, and I expect that you're spending lots of time thinking deeply about the problems. You have 50 minutes to do the exam, and so the questions that I ask won't be as in-depth as homework questions.

Q: But what will the actual questions look like?

A: Here are some types of questions that I might ask (this is not a comprehensive list):

- Given a new algorithm that you haven't seen before, analyze its worst-case runtime.
- Given an input and a particular algorithm, run the algorithm on that input.
- Use the Master Theorem to solve a recurrence.
- Given a new problem, how would you adapt or modify an algorithm we've discussed to solve the new problem?

Q: How should I go about studying?

A: The most important recommendation I can give is to study actively. Don't simply read your notes and then conclude that you've mastered the material. Instead, go back and redo previous mini homeworks. Re-derive the solutions to examples we did in class without looking at your notes. Make up some functions and prove asymptotic bounds for them. Make up some unsorted arrays and sort them using different algorithms.

Q: Will the exam be curved?

A: I aim to write my exams so that the median falls somewhere around a B+ (i.e., in the high 80s), and I do not intend to curve the scores. However, I reserve the right to change my mind should I discover after the fact that I've substantially miscalibrated the difficulty of the exam.