Final Exam Review

Below are the “question headers” that might appear on the final exam. Obviously some questions are complete as given, and if those are selected, will appear verbatim on the final. For other questions where additional information (a “problem instance”) is missing, I will select one for the final. I am willing to create example problems for any of these questions to aid your studying, so please just ask.

**Asymptotic Analysis**

1. Give the “best fit” running time in “Big O” notation for the function given below.

2. Order the following Big O functions from smallest to largest.

3. Consider the scenario discussed below. Which algorithm would you go with and why? *Practice Note: this will involve a problem statement with two choices for algorithms, one of which is faster asymptotically but slower on smaller problem instances. You will have to decide if the size of the problem you are solving is large enough to make the asymptotically-faster algorithm the one to choose and justify your response.*

**Greedy Algorithms**

4. What is the general pattern of a greedy algorithm?

5. Provide a reasonable selection function to use for a greedy algorithm which solves the problem described below.

**Divide and Conquer**

6. What are the two requirements of a problem that suggests we can use dynamic programming to solve it?

7. Describe the difference between top-down and bottom-up dynamic programming solutions. Include a mention of what constructs you are likely to use for each in a typical high-level language like Python.

8. Write the solution to an instance of size \( n \) of the problem below in terms of solutions to smaller instances (i.e. create a recurrence).

9. Would dynamic programming help in the problem above? Why or why not?

**P and NP**

10. Give a problem which is in \( P \) (including the inputs and solution (“outputs”)).

11. Give a problem which is \( NP \)-Complete (including the inputs and solution (“outputs”)).

12. What does it mean for a problem to be in \( \{P, NP\} \)?

13. What can we say about the running time of an existing algorithm that solves an \( NP \)-Complete problem exactly?

14. Give two things we can try if we wish to have an efficient algorithm for an \( NP \)-Complete problem we are trying to (more or less) solve.
Optimizing Code

15. Give an example problem where the choice of the implementation of a particular data structure has a huge impact on the performance of the algorithm using it.

16. Suggest a few changes to the Python function below that could decrease the running time of the function.

Randomized Algorithms

17. What is the general premise behind the use of randomized algorithms? Practice Note: What I’m looking for here is something along the lines of how we can trade strict guarantees in running time or other resource usage for efficiency – for instance, if we allow our algorithm to be wrong sometimes, we can significantly decrease the space or time complexity.

18. What is the difference between a Las Vegas and a Monte Carlo randomized algorithm?

Linear Programming

19. Give the objective function and constraints for the following problem.