CSCI 332, Fall 2024
Homework 11

Due before class on Tuesday, December 3rd, 2024–that is, due at 9:30am Mountain Time

---

Submission Requirements

- Type or clearly hand-write your solutions into a PDF format so that they are legible and professional. Submit your PDF on Gradescope.
- Do not submit your first draft. Type or clearly re-write your solutions for your final submission.
- Use Gradescope to assign problems to the correct page(s) in your solution. If you do not do this correctly, we will ask you to resubmit.
- You may work with a group of up to three students and submit ***one single document*** for the group. Just be sure to list all group members at the top of the document. When submitting a group assignment to Gradescope, only one student needs to upload the document; just be sure to select your groupmates when you do so.

Academic Integrity

Remember, you may access ***any*** resource in preparing your solution to the homework. However, you ***must***

- write your solutions in your own words, and
- credit every resource you use (for example: "Bob Smith helped me on this problem. He took this course at UM in Fall 2020"; "I found a solution to a problem similar to this one in the lecture notes for a different course, found at this link: www.profzeno.com/agreatclass/lecture10"; "I asked ChatGPT how to solve part (c)"; "I put my solution for part (c) into ChatGPT to check that it was correct and it caught a missing case.") If you use the provided LaTeX template, you can use the `sources` environment for this. Ask if you need help!

Grading

Remember, submitted homeworks are graded for completeness, not correctness. Correctness is evaluated using homework quizzes. Each submitted problem will be graded out of six points according to the following rubric:

- Does the solution address the correct problem?
- Does the solution make a reasonable attempt at solving the problem, even if not fully correct?
- Is the presentation neat?
- Is the explanation clear?

- Does the solution list collaborators or sources, or state that the student did not use any collaborators or outside resources?
- Is the solution written in the student's own voice (not copied directly from an outside resource)?

1. You're considering getting into trading cryptocurrencies, but first, you'd like to see how much money you could make. To do so, you will need to process historical datasets. Given n days of prices for a given cryptocurrency, you could have made the maximum amount of money by buying on a low price day and selling a later, high price day. For example, you may see a dataset where $n = 5$ and the prices were $p_1 = \$450, p_2 = \$300, p_3 = \$355, p_4 = \$505, p_5 = \$500$. The most money you could have made is by buying on day 2 and selling on day 4 for a \$205 profit.

   (a) There are some inputs that do not allow you to make any money. What property would such an input have?

   (b) Give a brute-force, $\Theta(n^2)$ algorithm to find the maximum amount of money you could have made (or report that no money could be made).

   (c) Although this problem has similarities to the 1-dimensional closest points problem, it is not exactly the same. Explain why simply sorting the list and checking adjacent entries would not be a correct algorithm for this problem.

   (d) We would like to write a divide and conquer algorithm for this problem. Just as we have done for many other problems, we will do this by dividing the array in half, using recursion to find the optimal solution in each half, and then finding a way to merge the optimal solutions from each half into one single optimal solution.

   Let the array you are trying find the max profit for be A. If you divide array in A into two halves, L and R (for left and right), the max profit is made in exactly one of these ways:

   - Buying and selling on days in L
   - Buying and selling on days in R
   - ???

   What is the third way that you could make the max profit from the prices in A?

   (e) Use your observation above to give a $\Theta(n \log n)$ algorithm for the max profit problem. Note: if you aren't sure about (d), get help before trying this one!

   (f) Write a recurrence relation the runtime of your algorithm. (You may assume that the length of the array is a power of 2, if that is easiest.)

   (g) Give a closed-form solution for the runtime of your algorithm. (You don't have to show how you got this.)