CSC373 F24: Assignment 1

Due: September 30, by midnight

Guidelines: (read fully!!)

- Your assignment solution must be submitted as a *typed* PDF document. Scanned handwritten solutions, solutions in any other format, or unreadable solutions will **not** be accepted or marked. You are encouraged to learn the LaTeX typesetting system and use it to type your solution. See the course website for LaTeX resources.
- Your submission should be no more than 8 pages long, in a single-column US Letter or A4 page format, using at least 9 pt font and 1 inch margins.
- To submit this assignment, use the MarkUs system, at URL `https://markus.teach.cs.toronto.edu/`
- This is a *group assignment*. This means that you can work on this assignment with *at most one other* student. You are *strongly encouraged* to work with a partner. Both partners in the group should work on and arrive at the solution together. Both partners receive the same mark on this assignment.
- Work on all problems together. If one of you writes the solution to a (sub-)problem, then the other student should proof read it. Both members of a group are responsible for understanding all solutions to all problems in their submission.
- You **may not** consult any other resources except: your partner; your class notes; your textbook and assigned readings. *Consulting any other resource, or collaborating with students other than your group partner, is a violation of the academic integrity policy!*
- You may use any data structure, algorithm, or theorem previously studied in class, or in one of the prerequisites of this course, by just referring to it, and without describing it. This includes any algorithm, or theorem we covered in lecture, in a tutorial, or in any of the assigned readings. Be sure to give a *precise reference* for the data structure/algorithm/result you are using.
- Unless stated otherwise, you should justify all your answers using rigorous arguments. Your solution will be marked based both on its completeness and correctness, and also on the clarity and precision of your explanation.

Question 1. (11 marks)

In this question you are given as input n intervals $I_1 = [a_1, b_1]$, ..., $I_n = [a_n, b_n]$ on the real line. Each intervals I_j is specified by the two numbers a_j and b_j. We assume that $a_j < b_j$ for all j. We also assume that no two intervals share any of their endpoints, i.e., all the numbers $a_1, \ldots, a_n, b_1, \ldots, b_n$ are distinct. The intervals are given in two arrays $A[1\,..\,n]$ and $B[1\,..\,n]$ where $A[j] = a_j$ and $B[j] = b_j$.

We will say that intervals I_j and I_k *cross* if $I_j \cap I_k \neq \emptyset$ but neither interval contains the other one. In other words, I_j and I_k cross if exactly one of the endpoints of I_k is contained in I_j.

Part a. (3 marks)

Suppose that there exists some number x so that $a_j < x$ for all j, and $b_j > x$ for all j. Give an algorithm running in worst case time complexity $O(n \log n)$ to compute the number of pairs $j < k$ such that I_j and I_k cross. Justify your answer.

HINT: use one of the divide and conquer algorithms from class.

Part b. (8 marks)

Give a divide and conquer algorithm that computes the number of pairs $j < k$ such that I_j and I_k cross, without making the assumption from the previous subquestion. Your algorithm should run in worst case time complexity $O(n \log^2 n)$. Justify your answer.

Part c. (7 marks)

(Bonus question - Optional). Modify your algorithm so that it runs in worst case time complexity $O(n \log n)$. Justify your answer.

HINT: Try to call any subroutine that takes time $O(n \log n)$ only once, rather than recursively.

Question 2. (19 marks)

In this question, you are tasked with deciding the locations for k grocery stores, to serve n houses on a street. Suppose we model the street as the interval from 0 to 1, and the locations of the houses are given as real numbers $x_1, \ldots, x_n \in [0, 1]$. You can assume that $x_1 < x_2 < \ldots < x_n$, i.e., the locations are sorted from left to right, and that they are given to you as an array $x[1\,..\,n]$ where $x[i] = x_i$. Each week, one of the people living in each house travels once to their closest grocery store to buy their groceries. Your goal is to compute the locations $y_1, \ldots, y_k \in [0, 1]$ of the k grocery stores that minimize the total distance each household travels every week, given x and k as input.

Part a. (4 marks)

Prove that if $k = 1$, then the optimal location of the single grocery store is the median of $x_1, \ldots, x_n$. I.e., the total distance each household travels to the grocery store at location y

$$|x_1 - y| + \ldots + |x_n - y|$$

is minimized by choosing y to be the median of $x_1, \ldots, x_n$. Here we define the median as follows: recalling that $x_1 < \ldots < x_n$ are sorted, the median is $x_{\lceil (n+1)/2 \rceil}$. So, the median of $x_1 < x_2$ is x_2, and the median of $x_1 < x_2 < x_3$ is x_2 as well. (This may not be how you have seen medians defined for n even; any of the standard definitions would work, but please stick to the one above.)

HINT: There are many ways to prove this. One possibility is to show that, for $x_1 \leq \ldots \leq x_n$,

$$|x_1 - y| + \ldots + |x_n - y| \geq |x_n - x_1| + |x_{n-1} - x_2| + \ldots + |x_{\lceil (n+1)/2 \rceil} - x_{\lfloor (n+1)/2 \rfloor}|,$$

and that the two sides of the inequality are equal when y is the median.

Part b. (5 marks)

For $1 \leq i \leq j \leq n$, let $M[i, j]$ equal $|x_i - y| + \ldots + |x_j - y|$, where y is the median of $x_i, \ldots, x_j$. Give an algorithm that computes all values of $M[i, j]$ for all $1 \leq i \leq j \leq n$ in worst case time complexity $O(n^2)$ (i.e., constant time per pair of i and j). Justify your answer.

Part c. (6 marks)

Give an algorithm that computes the minimum total travel distance achievable with k store locations. I.e., you need to output the minimum of

$$\min_{j=1}^{k} |x_1 - y_j| + \ldots + \min_{j=1}^{k} |x_n - y_j| \tag{1}$$

over all choices of $y_1, \ldots, y_k \in [0, 1]$. Your algorithm should run in worst case time complexity $O(kn^2)$. Use bottom-up dynamic programming, and be precise about your choice of subproblems, and the optimal substructure you are using, i.e., the recurrence (Bellman equation) satisfied by the subproblems, and its base cases. Give pseudocode for your algorithm, and justify your answers.

HINT: You can also equivalently write the objective (1) as

$$\min_{X_1,\ldots,X_k} \left\{ \min_{y_1 \in [0,1]} \left(\sum_{i \in X_1} |x_i - y_1| \right) + \ldots + \min_{y_k \in [0,1]} \left(\sum_{i \in X_k} |x_i - y_k| \right) \right\},$$

where the first minimum is over all ways to partition $x_1, \ldots, x_n$ into *disjoint* subsets $X_1, \ldots, X_k$. In other words, rather than directly choosing the store locations $y_1, \ldots, y_k$, we first choose the set of households X_j which will go to store j, and then we choose the location y_j of the j-th store to minimize the total distance the households in X_j need to travel.

Part d. (4 marks)

Modify your algorithm from the previous problem to also output the choice of store locations $y_1, \ldots, y_k$ that minimizes the total distance travelled. Your algorithm should still run in worst case time complexity $O(n^2 k)$. Justify your answer.

Question 3. (14 marks)

The Galactic Research Society (GRS) has decided to organize an interstellar expedition. The president insists that exactly k members, including herself, join the expedition, and all selected members are required to participate.

There are n GRS members, and they are organized in a strict hierarchical structure (i.e., a tree), with the president at the root. Every society member is represented as a node in the tree. Every member except the president has a supervisor (their parent in the tree), and members supervise at most two other members (their children in the tree).

The society's HR office has determined a "tension coefficient" between each member and their supervisor. This is a real number, and represents the degree to which there is tension between the member and their supervisor: a large positive number means their relationship is quite tense and there is potential for conflict, and a negative number with large absolute value means they get along very well.

Your task is to use a dynamic programming algorithm to choose exactly k members to join the expedition such that the total tension is minimized. You can assume that k ($1 \leq k \leq n$) is given to you, and also that the GRS hierarchy is given to you as a rooted binary tree $T = (V, E)$, and each edge $e = (u, v) \in E$ of the tree is labeled by the tension coefficient $t_e = t_{u,v}$. The total tension of a set $S \subseteq V$ of k GRS members equals the sum of the t_e for all $e = (u, v)$ where u and v are both in S. I.e., we add up the tension coefficients over all pairs of a society member and their supervisor who are both selected for the expedition.

Part a. (7 marks)

Define the subproblem structure you are using, and specify a recurrence satisfied by each subproblem, as well as the base cases of the recurrence. The recurrence should allow you to compute the value of a subproblem quickly (certainly in polynomial time), assuming the values of "smaller" subproblems have already been computed. Justify the recurrence.

Part b. (3 marks)

Using your recurrence from the previous question, give a dynamic programming algorithm to compute the smallest possible total tension of a set S of k GRS members that includes the president (root of the tree). Your algorithm should run in time polynomial in the total number of GRS members n. Give pseudocode for your algorithm, and analyze its worst case running time.

Part c. (4 marks)

Modify your algorithm from the previous subproblem to also output a set S of k GRS members including the president, that minimizes the total tension. Analyze the modified algorithm's worst case running time.