CSC265 W23: Assignment 2

Due: October 20, 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 7 pages long, in a single-column US Letter or A4 page format, using at least 10 pt font and 1 inch margins.
- To submit this assignment, use the MarkUs system, at URL `https://markus.teach.cs.toronto.edu/2023-09`
- 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. For each problem, one of you should write the solution, and one should proof-read and revise it. The first page of your submission must list the *name*, *student ID*, and *UTOR email address* of both group members. It should also list, for each problem, which group member wrote the problem, and which group member proof-read and revised it.
- 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 data structure, 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. (9 marks)

Let $M[1\,..\,n][1\,..\,n]$ be a matrix (two dimensional array), where the rows and columns are indexed from 1 through n, and the entries are integers. Suppose that the entries of M are sorted in non-decreasing order from left to right, and from top to bottom, i.e., $M[i][j] \leq M[i+1][j]$ and $M[i][j] \leq M[i][j+1]$ whenever the indexes are within range. We consider the MATRIX-SEARCH problem, in which you are given as input a matrix M satisfying the properties above, and an integer x, and the goal is to output either indexes i and j such that $M[i][j] = x$, or $\perp$ if no such indexes exist.

Prove that any algorithm in the decision tree model, which can access M and x only though comparisons queries, must have worst-case time complexity $\Omega(n)$. Any comparison query made by the algorithm can ask one of the following

- whether $M[i][j] < M[k][l]$ for some valid indexes i, j, k, l;
- whether $M[i][j] = M[k][l]$ for some valid indexes i, j, k, l;
- whether $x < M[i][j]$ for some valid indexes i, j;
- whether $x > M[i][j]$ for some valid indexes i, j;
- whether $x = M[i][j]$ for some valid indexes i, j.

Your proof should establish that any algorithm which only accesses M and x through these types of queries must make at least $\Omega(n)$ queries in the worst case. Use the adversary argument.

HINT: You may want to first show that it takes at least $\Omega(n)$ queries to find x in an unsorted one-dimensional array of size n, and then show that the MATRIX-SEARCH is at least as hard. To make the proofs easier, do not try to make the constants in your proof tight.

Question 2. (18 marks)

Your goal in this problem is to design a data structure that maintains a set of disjoint horizontal line segments in 2 dimensions. I.e. it maintains a set S where each element s of S has three integer attributes $s.xl$, $s.xr$, and $s.y$. Here s represents a horizontal line segment connecting a point with x-coordinate $s.xl$ and y-coordinate $s.y$ with a point with x-coordinate $s.xr$ and y-coordinate $s.y$. You can assume that $s.xl \leq s.xr$ for any segment s, and that every two segments s and s' in S have distinct y coordinates, i.e., $s.y \neq s.y'$.

The data structure should support inserting and deleting segments into S, as well as a query operation VERTICAL(x, y). The operation should return the segment $s \in S$ with largest y-coordinate that intersects a vertical ray starting from the point (x, y) and directed downwards, if one exists. I.e., the segment s returned by VERTICAL(x, y) should satisfy that $s.y \leq y$, $s.xl \leq x \leq s.xr$, and that s is the segment in S with largest value of $s.y$ that satisfies these properties. If no segment satisfies the properties, VERTICAL(x, y) returns NIL. See Figure 1 below.

Part a. (8 marks)

In this subquestion we consider the special case of the problem in which there exists a value ℓ so that, for all $s \in S$, $s.xl = \ell$. I.e., the left endpoint of each segment lies on the vertical line $x = \ell$. Under this assumption, design a data structure that supports INSERT(s) (which inserts s into S), DELETE(y) (which deletes the unique $s \in S$ with y-coordinate equal to y, if one exists), and VERTICAL(x, y) (as described above). Each operation should have worst-case time complexity $O(\log n)$ where n is the size of S.

Use an augmented AVL tree to design your data structure. Describe what key you use to build the AVL tree, what additional variables you augment it with, and how these additional variables are updated during inserts and deletes. Justify the correctness and worst case running time bound of your data structure operations.

Part b. (4 marks)

Show how to use the answer to the previous subquestion to give a data structure for the special case when there exists some number ℓ for which each segment $s \in S$ intersects the vertical line $x = \ell$. I.e., each $s \in S$

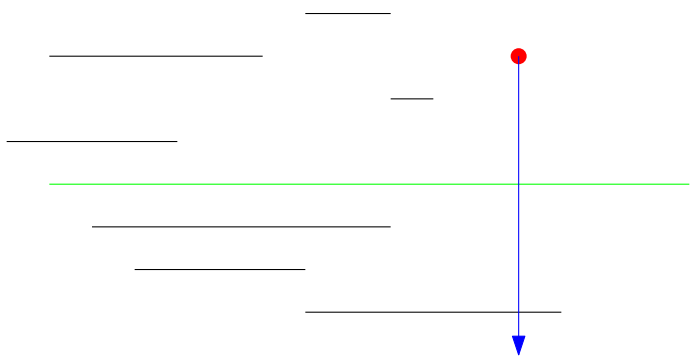

Figure 1: The figue shows in black the set S of horizontal segments. The red dot has coordinates (x, y). If VERTICAL(x, y) is called it would return the green segment since it is the highest that intersects the blue ray shooting down from the red dot.

satisfies $s.xl \leq \ell \leq s.xr$. Under this assumption, your data structure should support the three operations INSERT, DELETE, and VERTICAL in worst-case time complexity $O(\log n)$.

Part c. (6 marks)

In this final subquestion, we consider the case when the x-coordinates of all segments $s \in S$ are in the range $\{1, \ldots, U\}$. I.e., you can assume that any $s \in S$ satisfies $s.xl \in \{1, \ldots, U\}$ and $s.xr \in \{1, \ldots, U\}$. You can assume that U is an integer power of 2. Under this assumption, describe a data structure that supports the three operations INSERT, DELETE, and VERTICAL, so that INSERT and DELETE run in worst-case time complexity $O(\log(n) + \log(U))$ and VERTICAL runs in worst-case time complexity $O(\log(n) \log(U))$. You can use the two subproblems above. You can initialize your data structure for an empty S in worst case running time $O(U)$.

HINT: Buld a complete binary tree whose leaves are $\{1, \ldots, U\}$. Each node of the tree stores a subset of S using the data structure from the previous subquestion.