CS345– ALGORITHMS-II

NITIN SAXENA

ASSIGNMENT 2

POINTS: 50

DATE GIVEN: 23-AUG-2019 DUE: 13-SEP-2019

Rules:

- You are strongly encouraged to work *independently*. That is the best way to understand & master the subject.
- Write the solutions on your own and honorably *acknowledge* the sources if any. `http://cse.iitk.ac.in/pages/AntiCheatingPolicy.html`
- Submit your solutions, before time, to your specific TA. The dynamic student-to-TA mapping would be announced by Pranav.
  Pranav Bisht `pbisht@cse`
  Ashish Dwivedi `ashish@cse`
  Abhibhav Garg `abhibhav@cse`
  Nikhil Shagrithaya `nshagri@cse`
  Abhimanyu `abhimanu@iitk` .
  Preferably, give the TA a printed copy of your LaTeXed or Word processed solution sheet.
- There will be a penalty if you write unnecessary or unrelated details in your solution. (Remember that in the exams you'll get limited space.)
- Clearly express the fundamental idea of your algorithm; sketch the main steps of the pseudocode; give a proof of correctness and that of the claimed time complexity. This decides the distribution of partial marks.
- Problems marked '0 points' are for practice.

Question 1: *[6 points]* For a set T define MinGap(T) to be the minimum positive difference between the elements of T. Eg. MinGap of the set $\{10, 5, 0, 40, 2, 100\}$ is 2.

Design a tree structure where this new operation can be implemented in $O(\log n)$ time along-with the standard tree operations.

Question 2: *[5 points]* Recall the Orthogonal Range Search algorithm done in the class. Modify it so that it outputs the *number* of points in the given rectangle in $O(\log^2 n)$ time.

Question 3: *[13 points]* Design a data structure D that stores a sequence $\big(e_1, e_2, \ldots, e_n\big)$ of numbers, remembering the array order, such that the following operations can be done in $O(\log n)$ time:

(1) Insert(D, i, x): Insert the number x in the i-th location. The sequence grows to size $n + 1$.
(2) Delete(D, i): Delete the number in the i-th location. The sequence shrinks to size $n - 1$.
(3) Report(D, i): Output the number in the i-th location.
(4) Min(D, i, j): Output the smallest number from the i-th to the j-th locations. Assume $1 \leq i \leq j \leq n$.

Question 4: *[12 points]* Given a list L of n rectangles in the plane, output the maximum number of rectangles that overlap at any point (maximized over all points in the plane). Your algorithm should have time complexity at most $O(n \log n)$.

Question 5: *[14 points]* Let us see a good example of the greedy paradigm.

You are given a complete binary tree representing an electronic circuit (with wires as edges). There are n leaves. Each edge has a positive number which represents the *delay* along that edge. An electric pulse originates at the root and travels all the way to the leaves. The *delay incurred along a path* is the sum of the delay on its edges. You need to ensure that the pulse reaches all the leaves at the same time. As a result the entire circuit becomes *synchronized*.

To achieve this objective, you are allowed to increase the delay along some edges. The delay *enhancement* is the sum of the increase in the delay of all the edges. The aim is to synchronize the given circuit such that the total delay enhancement in the circuit is minimum.

(1) Design a greedy step that transforms the given instance to a smaller instance of the same problem. You might like to generalize the problem for this purpose.
(2) Then, establish a relation (and prove it formally) between the optimal solution of the given instance and optimal solution of the smaller instance.

(3) Describe the fastest possible implementation of the algorithm based on the greedy step and the relation derived above.

Question 6: *[0 points]* You saw a polynomial multiplication algorithm that takes $O(n \log n)$ complex-operations for degree n. How do you implement addition/multiplication of complex numbers on a computer?

Question 7: *[0 points]* Prove that for an AVL tree T and an input x, Split(T, x) takes $O(\log n)$ time.

Why does Insert(T, x) requires at most two rotations while Delete(T, x) may require much more?

Question 8: *[0 points]* In the class you saw a method to test input rectangles for an overlap. Write the pseudocode, the proof of correctness and the time analysis.

Question 9: *[0 points]* In the Interval Tree, can you modify Search(T, i) operation so that it outputs *all* the intervals in T that overlap with i?

Question 10: *[0 points]* Is there a difference between divide-conquer and greedy paradigms?

□□□