Data Structures and Algorithms COMP-251 A

Problem Assignment #4

1. Balanced Search Trees

Consider the following sequence of keys: (5, 16, 22, 45, 2, 10, 18, 30, 50, 12, 1).

(a) Consider the insertion of items with this set of keys (in the order given) into an initially-empty (2,4) tree T. Draw T after each insertion.

(b) Show that a (2,4) tree can have a different structure when the order of the items inserted is changed. Use the keys used in (a) in a different order of insertion.

(c) Prove that the height of a (2,4) tree storing n items is $\Theta(\log n)$, i.e., prove that is it both $O(\log n)$ and $\Omega(\log n)$.

2. Element Uniqueness *(Problem 6.21 in Udi Manber text)*

The input is a set S of n real numbers. Design an $O(n)$ worst-case time algorithm to construct a number that is guaranteed **not** to be in S. Prove that $\Omega(n)$ is a lower bound on the worst-case complexity of this problem under the *comparison-based* model of computation.

3. Non-collinearity of Points

The input is a set S of n points in the plane in general position, i.e., no three points lie on a line. Invent an $O(n \log n)$ worst-case time algorithm to construct a new point X such that X is not collinear with any pair of points in S. Prove the **correctness** of your algorithm. Prove the **complexity** of your algorithm.

4. Maximal Points of a Set *(Problem 8.18(a) in Udi Manber text)*

The input is a set S of n points in the plane given by their x and y coordinates (real numbers). A point p in the plane is said to **dominate** another point q if **both** the x and y coordinates of p are greater than or equal to those of q. A point p is a **maximal** point of S if no other point of S dominates it. Invent an $O(n \log n)$ worst-case time algorithm for finding **all** the maximal points of a given set S. Prove the **correctness** of your algorithm. Prove the **complexity** of your algorithm.