Data Structures - Assignment no. 5

Remarks:

- Write both your name and your ID number very clearly on the top of the exercise. Write your exercises in pen, or in clearly visible pencil. Please write very clearly.
- Recall that 80% of the theoretical exercises must be submitted. The exercises can and must be worked on and submitted alone.
- Give correctness and complexity proofs for every algorithm you write.
- For every question where you are required to write pseudo-code, also explain your solution in words.

1. Describe an algorithm that receives as input a sorted array that contains \( n \) different real numbers, and returns

   (a) a 2-4+ tree whose keys are these numbers.
   (b) an RBT tree whose keys are these numbers.

   The algorithms should run in \( O(n) \) time.

2. Insert the keys 23, 34, 4 and 39 to the 2-4+ tree depicted in Figure 1. Then delete keys 10 and 20. Now draw the resulting tree.

3. Suggest a data structure based on RBT that supports the following operation and given time complexities.

   - \( \text{Init}(x_1, \ldots, x_n) \) - Init the DS with \( n \) real numbers (unordered) in \( O(n \log n) \) time.
   - \( \text{Insert}(x) \) - Insert \( x \) to the DS in \( O(\log n) \) time.
   - \( \text{findMin}() \) - Return the value of the minimal element in \( O(1) \) time.
   - \( \text{findMax}() \) - Return the value of the maximal element in \( O(1) \) time.
   - \( \text{findMed}() \) - Return the value of the median element in \( O(\log n) \) time.
   - \( \text{DelMin}() \) - Remove the minimal element in \( O(\log n) \) time.
   - \( \text{DelMax}() \) - Remove the maximal element in \( O(\log n) \) time.
   - \( \text{DelMed}() \) - Remove the median element in \( O(\log n) \) time.

4. Suppose that a node \( x \) is inserted into a red-black tree with RB-INSERT and then immediately deleted with RB-DELETE. Is the resulting red-black tree the same as the initial red-black tree?

5. (a) You are given a 2-4+ search tree where the root has exactly two children, \( u \) and \( v \). Let \( X \) be the number of descendants of \( v \), and \( Y \) be the number of descendants of \( u \). (In other words, \( X \) is the size of the subtree of \( v \), and \( Y \) is the size of the subtree of \( u \)). Is it necessarily true that \( X \leq 2008 \cdot Y \)? Explain your answer.

   (b) Solve the same question for an R-B tree
6. Suppose you do a sequence of $m$ insertions and deletions on a 2-4+ tree where you get a pointer to the leaf that has to contain the new item in case of insert, or contains the item to be deleted in case of delete. The 2-4+ trees contain at most $n$ elements when we start performing the sequence. Prove that it takes $O(n + m)$ time to perform the sequence.

7. Write an ADT that supports the operations:
   - $Init(S)$ that receives an array $S$ of size $n$, such that each cell contains the age and salary of some worker
   - $MaxSalary(i, j)$ which returns the age of the oldest worker in $S$ whose salary is between $i$ and $j$, for some reals $i$ and $j$.

   Assume that $MaxSalary(i, j)$ refers to the array $S$ in the last call to $Init(S)$, and returns 0 if $Init$ was never called. You don’t need to prove your answers in this question.

   (a) A call to $Init(S)$ should take $O(n \log n)$ time W.C, and a call to $MaxSalary(i, j)$ should take $O(\log n)$ time W.C.

   (b) A call to $MaxSalary(i, j)$ should take $O(1)$ time W.C., and a call to $Init(S)$ can take any finite amount of time.

Figure 1: A 2-4+ tree. (Recall that a 2-4+ tree is a 2-4 tree where the real set elements are only the keys that are at the leaves, and the rest of the elements are just pivot elements to aid in searching.)