Data Structures - Assignment no. 4, March 21, 2007

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. Insert the keys 25, 33, 9 and 35 to the 2-4+ tree depicted in Figure 1. Then delete keys 10 and 20. Now draw the resulting tree.

2. (a) Give an algorithm that is given a binary tree T of n vertices with keys at the nodes, and determines whether T is a binary search tree. The algorithm should run in time $O(n)$. Give: (i) pseudo-code; (ii) an explanation of the algorithm; (iii) an explanation why it is correct; and (iv) an explanation why the running time is indeed $O(n)$.

   (b) Describe an algorithm that given a sorted array of size n builds a 2-4+ tree that contains the same keys as the array. The algorithm should run in time $O(n)$. Give: (i) a description of the algorithm; (ii) an explanation why it is correct; and (iii) an explanation why the running time is indeed $O(n)$.

3. Consider a 2-4 search tree, where instead of having up to 4 children for every node, you have up to $\log n$ children for every node. Suppose that each node is kept by a small data structure that can do whichever manipulations you like, in constant time.

   (a) What is the depth of this tree?

   (b) What are the running times of the usual operations (insert, delete, find) for this tree?

   Note 1: The assumption that a node can do whichever manipulations you like in $O(1)$ time is not very realistic. One way to justify it is considering the case where you have a very fast subprocessor, capable of handling small amounts of data, but very quickly. You program it in advance to perform all the manipulations you like on $\log n$-sized nodes.

   Note 2: You might want to use the fact that $\log_a b = \frac{\log a}{\log b}$, so $\log_{\log n} n = \frac{\log n}{\log \log n}$.

4. (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 $X \leq 2006 \cdot Y$? Explain your answer.

   (b) Solve the same question for an R-B tree

5. **Challange Question. Not to be Submitted.**
Bounded-Balance Trees. In a binary tree, define the *size* of a vertex v, denoted by S_v, as the number of vertices in v's subtree. A binary search tree is called a *bounded-balance (BB)* tree, if for every vertex v, it holds that $S_{v.left} \geq \lfloor \frac{S_v}{10} \rfloor$ and $S_{v.right} \geq \lfloor \frac{S_v}{10} \rfloor$, where $v.left$ is v's left child, and $v.right$ is v's right child.

   (a) Prove that if T is a BB tree containing n nodes, then the depth of T is $O(\log n)$.

   (b) We define a BB tree data structure as follows: The data structure is a normal binary search tree, with an additional field *size* at each vertex. The field $v.size$ holds S_v.

   Consider a sub-tree with root v, where $v.left$ and $v.right$ are both BB trees, and also $S_v = 10x, S_v.right = x - 1, S_v.left = 9x$. Thus, v's subtree is not a BB tree, but there is just one small violation. Show how to perform rotations that will correct this violation. (Note: a little case-analysis is needed. In a certain case, you may need to perform two rotations, not just one).

   (c) Conclude from the last section that you can perform lookup, insertion and deletion in a BB tree, all in time $O(\log n)$.

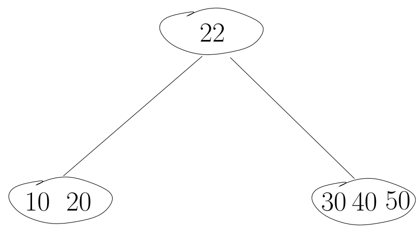

Figure 1: A 2-4+ tree. (Recall that a 2-4+ tree is a 2-4 tree where the real set elements are only the keys that are at the leaves, and the rest of the elements are just pivot elements to aid in searching.)