Data Structures - Assignment no. 4, March 29, 2006

Remarks:

- **Please write your exercises in pen, or in clearly visible pencil. Please write very clearly.**
- **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. 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.

3. (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 $O(n)$.

4. **(Corrected 3/4/2006)** Describe a data structure that implements a dictionary ADT. (The dictionary ADT maintains a set of keys, S, and supports the operations $insert(x)$, $delete(x)$ and $find(x)$). Let n be the number of operations performed on the data structure **since it was created**[1]. The data structure should implement insert and delete in time $O(1)$ worst-case, and find in time $O(n \log n)$ worst case. Also, the amortized complexity of all operations should be $O(\log n)$. (In other words, the worst-case time of performing n operations should be $O(n \log n)$). Describe the data structure (no need to give pseudocode), and prove your claims about the running time.

[1] Originally we said here that n is the maximum possible size of the data structure, which made the question unsolvable: Think about performing a sequence of n inserts, followed by delete,insert,delete,insert, and so on.

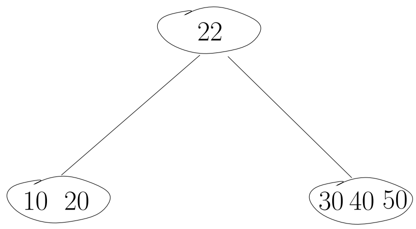

Figure 1: A 2-4+ tree. (Recall that a 2-4+ tree is a 2-4 tree where the real set elements are only the keys that are at the leaves, and the rest of the elements are just pivot elements to aid in searching.)