Data Structures - Assignment no. 3, 2008

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.**
- **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. Find the order of growth for the following recursively given functions. Explain your answer. Assume that $T(n)$ is constant for $n \leq 2$.

   (a) $T(n) = 8T(n/2) + 3n^2$
   (b) $T(n) = 2T(\sqrt{n}) + 1$
   (c) $T(n) = T(9n/10) + n$
   (d) $T(n) = 2T(n/4) + \sqrt{n}$
   (e) $T(n) = T(n-1) + 1/n$.
   (f) $T(n) = 2T(n/2) + n\log^4 n$ (Hint: The approach to solving this is similar to the approach to solving $T(n) = 2T(n/2) + n$).

2. The running time of an algorithm A is described by the recurrence $T(n) = 7T(n/2) + n^2$. A competing algorithm A' has a running time of $T'(n) = \mathbf{a}T'(n/4) + n^2$. What is the largest integer value for a such that A' is asymptotically faster than A?

3. 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.

4. Consider a B-tree with $b = (\log n)/2$. (Thus, every node has at most $\log n$ children and holds at most $\log n$ keys). 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 $\log_{\log n} n = \frac{\log n}{\log\log n}$.

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. Consider the following data structure that supports the operations Insert(x) and Find(x).

   - Insert(x) - Insert the element x.
   - Find(x) - Returns true if x is in the data structure and false otherwise.

   The data structure is implemented as follows. The elements are held in a collection of sorted arrays. The length of each array is a power of 2. Each array has different length, i.e. there are no two arrays with the same length. For example, if the data structure currently contains 22 elements then the elements are held in arrays of length 2, 4 and 16. Each array in the data structure is sorted, however the order between elements of different arrays is unknown.

   Insert(x) is implemented as follows. Create an array of size 1 that includes x and add it to the data structure. While there are 2 arrays of the same size merge them into one sorted array with double the size.

   Find(x) is implemented as a binary search in each of the arrays in the data structure.

   (a) What is the worst-case time complexity of Find(x) when the data structure contains n elements?

   (b) What is the worst-case time complexity of Insert(x) when the data structure contains n elements?

   (c) What are the *amortized* costs of the insert(x) and find(x) operations?

   (d) What is the total cost, in the worst case, of n operations on an initially empty data structure?

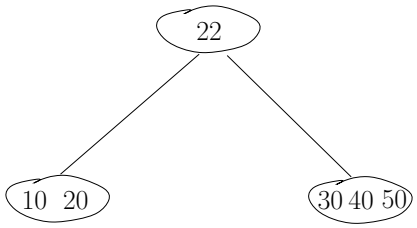

Figure 1: A 2-4 tree.