CS 5633 Analysis of Algorithms – Spring 05

3/3/05

4. Homework

Due **3/10/05** before class

1. **Red-black trees (6 points)**
   Find a sequence of numbers which, when incrementally inserted into a red-black tree, causes the following sequence of rotations:

   right, left, right, right.

   You may start with an initially non-empty tree, and you may insert numbers that do not cause any rotations. But there should not be any additional rotations performed.

   Draw the sequence of trees that you obtain after each insertion. For each such tree indicate the node that violates the red-black tree condition, indicate the nodes that participate in the rotation, the type of the rotation, and the subtrees that correspond to each other before and after the rotation.
   Hint: Use a red-black tree demo from the web.

2. **Hashing (6 points)**
   - What is the best-case running time for inserting n elements into an initially empty hash table, using open addressing with linear probing?
   - What is the worst-case running time for inserting n elements into an initially empty hash table, using open addressing with linear probing?
   - What is the best-case running time for inserting n elements into an initially empty hash table, using chaining?
   - What is the worst-case running time for inserting n elements into an initially empty hash table, using chaining?

3. **Amortized insertion into a red-black tree (6 points)**
   Consider a red-black tree which provides only the operations INSERT(x) a nd DELETEALL(). INSERT(x) inserts element x, and DELETEALL() deletes and frees the memory of all elements in the tree.

   Consider an arbitrary sequence of n of these operations. Since DELETEALL() takes $O(n)$ time in the worst case, the runtime for the whole sequence of operations could be $O(n^2)$.

   Analyze the amortized runtimes of INSERT(x) and DELETEALL() using the **accounting method**. *Hint: Read chapter 17.1; although it uses an aggregate analysis, the approach is very similar.*

4. **Buildheap (3 points)**
   Consider the analysis of the runtime of BUILD-MAXHEAP on page 133. It is shown that the runtime is $O(n)$.

   Argue why the proof corresponds to an amortized analysis. What type of amortized analysis is being used?

FLIP OVER TO BACK PAGE $\Longrightarrow$

5. **B-trees (6 points)**

- What is the maximum number of keys that can be stored in a B-tree with minimum degree k and height h?
- The CPU time of B-Tree-Search is $O(k \log_k n)$. Show that, if B-Tree-Search is changed to use **binary search** instead of linear search on the key, then the CPU time is only $O(\log n)$, which is independent of k. *Hint: Write down the runtimes and remember some log-formulas, and the k-s will cancel.*