CS 5633 Analysis of Algorithms – Spring 04

2/25/04

7. Homework

Due **3/10/04** before class

1. **Amortized insertion into a red-black tree (4 points)**
   Consider a red-black tree which provides only the operations INSERT(x) and 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.*

2. **Queue from two stacks (6 points)**
   Assume we are given an implementation of a stack, in which PUSH and POP operations take constant time each. We now implement a queue using two stacks A and B as follows:

   ENQUEUE(x):
   • Push x onto stack A

   DEQUEUE():
   • If stack B is nonempty, return B.POP()
   • Otherwise pop all elements from A and while doing so push them, except the last one, onto B. Return the non-pushed element.

   a) (1 point) Demonstrate the behavior for the sequence of operations ENQUEUE(1), ENQUEUE(2), ENQUEUE(3), DEQUEUE(), ENQUEUE(4), ENQUEUE(5), DEQUEUE():, DEQUEUE():, DEQUEUE(): by drawing A and B after each operation.

   b) (2 points) Why is the algorithm correct? Argue which invariants hold for A and B.

   c) (3 points) Prove that the amortized runtime of ENQUEUE and DEQUEUE each is $O(1)$. Use the accounting method to show it.

FLIP OVER TO BACK PAGE $\Longrightarrow$

3. **Buildheap (2 points)**
In analyzing the runtime of buildheap, in what sense did we actually use an amortized analysis? What kind of amortized analysis did we use?

4. **Red-black trees (4 points)**
Find a sequence of numbers which, when incrementally inserted into a red-black tree, causes the following sequence of rotations:

left, right, left, left, right, right.

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.

5. **Delete in a red-black tree (4 points)**

a) Consider the red-black tree on slide number 23 of the red-black tree slides. Show how the deletion algorithm on page 288 works on that tree when element 22 is deleted. Draw all intermediate steps.

b) Consider again the red-black tree on slide number 23 of the red-black tree slides. Now delete element 15. What is the resulting red-black tree?

c) Consider a red-black tree T. Now insert an element, and then delete it again. Is the resulting red-black tree equal to T?

6. **B-tree construction (4 points)**

Insert the keys $\{1, 2, \ldots, 16\}$ incrementally into an initially empty B-tree with minimum degree $t = 2$. Draw the B-trees that are obtained in that way. You may draw only those trees right after a major change like a node split.

7. $2-3-4$ **trees (4 points)**
A $2-3-4$ tree is a B-tree with minimum degree $t = 2$.

a) Argue that every $2-3-4$ tree corresponds to a red-black tree, and vice versa. *Hint: Find a canonical mapping from an arbitrary* $2-3-4$ *tree to a red-black tree, and another mapping from an arbitrary red-black tree to a* $2-3-4$ *tree.*

b) Do the insertion procedures in the $2-3-4$ and in the corresponding red-black tree also correspond to each other?

General hint: Draw examples to see what's going on.