CSC265 Fall 2021 Homework Assignment 6

due Wednesday, October 27, 2021

For any node x in a binary tree, let $n(x)$ denote the number of nodes in the subtree rooted at x. We say that the binary tree is *unbalanced* at x if either x's left subtree or x's right subtree contains more than $2n(x)/3$ nodes.

1. Prove that if x is a node at depth greater than $\lfloor \log_{3/2} n \rfloor$ in a binary tree with n nodes, then the binary tree is unbalanced at some proper ancestor of x.

2. Given an arbitrary binary search tree with n nodes, explain how to construct, in $O(n)$ time, a binary search tree with the same nodes such that, for each node, the number of nodes in its left and right subtrees differ by at most one. Briefly justify why your algorithm is correct and runs in the required time.

Consider a binary search tree augmented with a single variable *size* which is the number of nodes in the tree. (Note that each node x of the binary search tree only has three fields, *x.key*, *x.left*, and *x.right*.) Suppose that each time an INSERT is successfully performed, *size* is incremented. If the newly inserted node, x, has depth greater than $\lfloor \log_{3/2} size \rfloor$, then the ancestors of x are examined in order, starting from the parent of x until an ancestor a of x is reached such that the binary tree is unbalanced at a. Finally, the subtree rooted at a is replaced by a binary search tree with the same nodes, as described in question 2, i.e., for each node in the subtree, the number of nodes in its left and right subtrees differ by at most one.

3. Explain how the node a can be found in $O(n(a))$ time.

4. Suppose that a sequence of n INSERT operations are performed, beginning with an empty tree. Prove that the height of the resulting tree is at most $\lfloor \log_{3/2} n \rfloor$.

5. Use the accounting method to prove that the allocated cost of each INSERT in a sequence of n INSERT operations, beginning with an empty tree, is $O(\log n)$. Maintain the credit invariant that each node x in the tree contains at least $3(|n(x.left) - n(x.right)| - 1)$ credits. Remember to explicitly state what a credit can pay for.