ICS621 Homework 5: Amortized weight-balanced trees

Problem 17-3 from CLRS. Consider an ordinary binary search tree augmented by adding to each node x the attribute $x.size$ giving the number of keys stored in the subtree rooted at x. Let α be a constant in the range $1/2 \leq \alpha < 1$. We say that a given node x is a α-balanced if $x.left.size \leq \alpha \cdot x.size$ and $x.right.size \leq \alpha \cdot x.size$. The tree as a whole is α-balanced if every node in the tree is α-balanced. The following amortized approach to maintaining weight-balanced trees was suggested by G. Varghese.

a) A 1/2-balanced tree is, in a sense, as balanced as it can be. Given a node x in an arbitrary binary search tree. show how to rebuild the subtree rooted at x so that it becomes 1/2-balanced. Your algorithm should run in time $\Theta(x.size)$, and it can use $O(x.size)$ auxiliary storage.

b) Show that performing a search in an n-node α-balanced binary search tree takes $O(\lg n)$ worst-case time.

For the remainder of this problem, assume that the constant α is strictly greater than 1/2. Suppose that we implement Insert and Delete as usual for an n-node binary search tree, except that after every such operation, if any node in the tree is no longer α-balanced, then we "rebuild" the subtree rooted at the highest such node in the tree so that it becomes 1/2-balanced. We shall analyze this rebuilding scheme using the potential method. For a node in a binary search tree T, we define

$$\Delta(x) = |x.lefr.size - x.right.size|,$$

and we define the potential of T as

$$\Phi(T) = c \sum_{x \in T : \Delta(x) \geq 2} \Delta(x),$$

where c is a sufficiently large constant that depends on α.

c) Argue that any binary search tree has nonnegative potential and that a 1/2-balanced tree has potential 0.

d) Suppose that m units of potential can pay for rebuilding an m-node subtree. How large must c be in terms of α in order for it to take $O(1)$ amortized time to rebuild a subtree that is not α-balanced?

e) Show that inserting a node into or deleting a node from an n-node α-balanced tree costs $O(\lg n)$ amortized time.