CSC 378F 2000
Enriched Assignment 2

due: Wednesday, October 18, 2000

1. CLR problem 14-1 (page 278).

2. Let $\frac{1}{2} < \alpha < 1$. A node x in a binary tree is *α-balanced* if

$$\begin{aligned} size(left[x]) &\leq \alpha \cdot size(x) \text{ and} \\ size(right[x]) &\leq \alpha \cdot size(x), \end{aligned}$$

   where the size of a node is the number of nodes (including itself) in the subtree rooted at that node.

   A binary tree is *α-balanced* if all its nodes are α-balanced.

   A node in an n-node binary tree is *deep* if it has depth greater than $\lfloor \log_{(1/\alpha)} n \rfloor$.

   (a) Prove that $\lfloor \log_{(1/\alpha)} n \rfloor$ comparisons are sufficient to search an n-node α-balanced binary search tree.

   (b) Prove that if a node is not α-balanced, then its heavy subtree contains at least two more nodes than its light subtree.

   (c) Prove that any deep node in a binary tree has an ancestor that is not α-balanced.

   (d) Describe an algorithm that, given a binary search tree with parent pointers and given a deep node in that tree, finds an ancestor y of x that is not α-balanced in $O(size(y))$ time.

   Prove that your algorithm is correct and runs in the required time.

   Let T be an n-node binary search tree with no deep nodes. Suppose an insertion is performed on T and the new node x is deep. Then rebuild the subtree rooted at an α-unbalanced ancestor y of x so that the resulting subtree is a binary search tree, contains the same nodes, and is $\frac{1}{2}$-balanced.

   (e) Explain how this can be accomplished in $O(size(y))$ time, given n, even if T does not have parent pointers.

   (f) Prove that the resulting tree has no deep nodes.

3. (a) Describe an algorithm that, given an unordered list of n intervals, decides, in $O(n \log n)$ worst case time, whether it contains a pair of intervals that overlap. Justify why your algorithm is correct and runs in the required time.

   (b) Suppose R and R' are rectangles whose edges are parallel to the x and y axes. Define what it means for R and R' to overlap in terms of their left and right x coordinates and top and bottom y coordinates.

   (c) Describe an algorithm that, given an unordered list of n rectangles whose edges are parallel to the x and y axes and which are specified by their left and right x coordinates and top and bottom y coordinates, decides, in $O(n \log n)$ worst case time, whether it contains a pair of rectangles that overlap.

   Justify why your algorithm is correct and runs in the required time.

   Hint: Consider the left and right x coordinates of all rectangles in order from smallest to largest. For each, do some processing.