Homework 8 - Optimization

Algorithms I - CSCI 700 - Fall 2010 - Prof. Rosenberg

Due October 26 at 6:30pm

Problem 1) (15 points) Insert the following items into an empty AVL tree: [1, 4, 5, 9, 7, 8, 2]. Show what the AVL tree looks like after each item is inserted. That is, insert the item into the tree, and perform any required rotations to balance the tree. It is only required to show the tree after the rotations are complete and the tree is balanced. This will require that you draw 7 trees.

Problem 2) Red Black Trees

2a. (15 points) Show the red-black trees that result after successively inserting the keys [41, 38, 31, 12, 19, 8] into an empty red-black tree. This will require that you draw 6 trees. One after each insertion (and any necessary recoloring and balancing).

2b. (15 points) Consider a red-black tree formed by inserting \( n \) nodes with RB-INSERT as described in class and in Cormen 13.3 (p. 280). Argue that if \( n > 1 \), that the tree contains at least one red node.

Problem 3) Merging Binary Search Trees

3a. (25 points) Given two binary search trees, \( T \) and \( S \), write pseudocode to “merge” the elements in \( T \) and \( S \). That is, display the elements of \( T \) and \( S \) in order. For example, if \( T \) was in Figure ??, and \( S \) in Figure ??, then the output would be 1, 2, 3, 4, 5, 6. Feel free to use PRINT to output each element. However, no penalty will be given for constructing an output array containing the elements.

3b. (10 points) Assume there are \( k \) elements in \( T \), and \( l \) elements in \( S \), and \( n = k + l \). Describe the running time of your MERGETREES(T,S) algorithm in terms of \( k \), \( l \), and/or \( n \).
Problem 4) 2-3 Trees.

4a. (15 points) A 2-3 tree with height = 2 can contain at most 8 elements. This comes from a root with 2 keys, and 3 children. Each of the three children then contain two keys. Present 8 elements whose successive insertion into an empty 2-3 tree will lead to a 2-3 tree with height = 2.

4b. (5 points) Draw the resulting tree from the previous question.