Algorithms, Section 2
Spring 2003
Homework 6a
Due Thursday, March 6

1. **[10 Points] Sorting with Trees.** (This is a short review problem.)

   (a) Imagine that we have a binary search tree with n elements in it. Recall that an inorder traversal of the tree is a recursive traversal which takes a node v as input and does the following: Recursively traverse the left subtree of v, print out the value of v, recursively traverse the right subtree of v. Note that in a binary search tree, an inorder traversal beginning at the root prints out the contents of the tree in increasing order. Explain briefly why an inorder traversal of a tree with n elements takes $O(n)$ time regardless of how imbalanced the tree may be.

   (b) Professor Lai, who now teaches at the Massachusetts Institute of Typography, wants to use inorder traversals to get a fast sorting algorithm. Here is his idea: Given an unsorted array of n elements, first insert them one-at-a-time into an initially empty binary search tree. Then do an inorder traversal of the binary search tree. Each time the inorder traversal prints out a number, put that in the next available position in a new array. The new array will be the sorted version of the original array! What is the worst-case running time of Professor Lai's algorithm? Poor Professor Lai. No tenure again!

   (c) Assume that instead of using regular binary search trees in Professor Lai's approach, we were to use red-black trees. Analyze the running time of this "Red-Black Sorting algorithm". Explain your analysis.

   (d) How about if B-Trees had been used instead of red-black trees? What would the running time be for the "B-Tree Sorting Algorithm" ?

2. **[20 Points] Join Operation on Red-Black Trees.** CLRS Exercise 13-2, parts a through f (page 295). Part a is modified; only argue why it works for insert, considering all three cases (including the one that might turn the root into a red node, and the resulting fix that insert would need to make to address this).