CIS 313 Introduction to Data Structures
Winter 2007

Assignment 4

due Wednesday, February 14, 2007

1. Explain how to use an AVL tree to sort n elements in $O(n \lg n)$ time. **[6 points]**
2. Insert into an initially empty AVL tree, in the order given, the following values: 10, 15, 17, 16, 22, 25, 20, 19, 21, 23, 9, 5, 8. Show (some of) the intermediate steps. **[6 points]**
3. Insert the values above into an initially empty (2,4)-tree. **[6 points]**
4. Let T and U be two (2,4)-trees storing n and m items, respectively, such that any item in T has a key less than the keys of all items in U. Describe an $O(\lg n + \lg m)$ method for *joining* the trees into a single tree that stores all the items in T and U. The original T and U may be destroyed in the process. **[8 points]**
5. Suppose that an AVL implementation stores the height of the subtree in each node to calculate the balance factors. Also suppose that one byte (8 bits, unsigned) are allocated for this purpose. What is the fewest number of nodes in an AVL tree that could cause one of these fields to overflow? What is the largest number of nodes that could be accomodated without overflow? (Use the g_k, related to the Fibonacci numbers, from class.) **[6 points]**

Total: 32 points

Notes:

- *(Q1)* All of the recursive tree traversal methods - preorder, inorder, and postorder - can be done in $O(n)$ time. (See p 255 of text.)
- *(Q5)* With 8 bits, we can store all values from 0 up to $2^8 - 1 = 255$.
- *(Q5)* We had defined g_k as the minimum number of nodes in an AVL tree of height k. We then saw that $g_k = F_{k+3} - 1$, where F_k is the k^{th} Fibonacci number.