Helwan University
Faculty of Computers and Information

Review Problems 3

Department of Computer Science
Data Structures
Spring 2007

1. Write the recursive function `PostorderRecursive` to traverse a tree.
2. Write a function that swaps a tree. A tree is swapped by swapping the two children of every node in the tree. For example, if you draw the tree on a transparent paper, the swaped tree will be the tree that appears from the other side of the paper.
3. Write the function `LeafCount` that counts the number of leaf-nodes in a tree.
4. Write the function `TreeDepth` recursively to calculate the depth of a tree.
5. Can you rewrite the previous function iteratively; it is a little bit tough (do not spend too much time on it.)
6. Write the iterative function `PreorderIterative` to traverse a tree.
7. Write the function `TreeSize` iteratively to calculate the number of nodes in a tree.
8. Rewrite the previous function recursively.
9. Prove that the condition `if(!StackEmpty(&s))`, in the function `InorderIterative` explained in lectures, is redundant. That is, the condition is always 1; hence, removing it and writing the body of the `if`-statement directly will have no effect on the function.
10. Redefine the structure `Tree` in the way that enables us to design the functions `TreeSize` and `TreeDepth` in $O(1)$ complexity.
    (a) Modify the function `CreateTree`
    (b) Modify the iterative version of the function `InsertTree`
    (c) Rewrite the recursive function `DestroyTree` to match the new type definition.
11. Write a function that replaces an element, if exists, in a tree with another one—both elements, of course, have the same key value; then, the function makes that element a leaf node by deleting all of its successors (its children and the children of its children, ...etc).

    Very Important Note: In this problem you learn that it is not necessary to have our solution inclusively recursive or iterative. Rather, our recursive function my be just a piece in our solution (recall Problem 6 in "Review Problems 2".)