Data Structures 10/26/2016

0145-343-001

Note Taker: Jai Punjwani

ANNOUNCEMENTS

Notes:

PowerPoint: <http://home.adelphi.edu/~siegfried/cs343/343l5.pdf>

* Traversing a tree can be done via preorder, postorder, or inorder traversal
* These traversal methods are particularly useful for infix, prefix, postfix parsing
* Q: Is it beneficial to use an array implementation for a tree rather than a pointer implementation?
* A: To support languages that do not have pointers, array implementations are more suited. However, most of the time, pointers will suffice
* Implementation of tree usually contains: root, right, left
* Example: Given 10 distinct numbers between 0 and 100. We read them down in a shuffled order. The first number starts as the root of the tree. Then if the next number is less than the root, it becomes the left node. If it is greater than the root, it becomes the right node.
* Takeaway: If we were to search a tree that is sorted in the above fashion, we do not have to traverse through each element!
* Having a balanced tree can save a lot of work

Array Implementation of Tree (example):

* Imagine an array of indices 0 to 7 filled by the letters A through G respectively. What if we delete a letter? We need a Boolean to indicate whether that array index is empty. Or possibly a dummy value