Boran Yıldırım
21401947
CS 202-01

Question 3

Explain briefly how you implemented above ***levelorderTraverse***, ***span*** and ***mirror*** functions and analyze their worst-case running time complexities. Discuss whether or not each can be implemented to be asymptotically faster (e.g., $O(n)$ as opposed to $O(nlogn)$).

Firstly, ***levelorderTraverse*** function occurs from basically two functions. One is ***levelorder*** which prints all nodes with recursively at a given level and it is private. Other is ***levelorderTraversal*** which prints level order traversal of the tree and is called by users. ***levelorderTraversal*** makes use of ***levelorder*** to print nodes at all levels one by one starting from root. Time complexity of this algorithm is $O(n^2)$ in the worst case. If the Binary Search Tree is balanced, time complexity would be $O(nlogn)$.

Secondly, ***span*** function also occurs from two functions. One is ***span*** with two integer parameters, which is called by users for getting the number of nodes in the Binary Search Tree with data values within the range of the input parameters. Other is ***span*** which is private and is made use of public ***span*** function. Private ***span*** counts the data one by one within the range of input parameters. Time complexity of this algorithm is $O(n)$.

Finally, ***mirror*** function also occurs from two functions. One is ***mirror*** which is called by users without any parameters for changing the Binary Search Tree so that the roles of the left and right pointers are swapped at every node. Other is ***mirror*** which is private and gets the parameter root of the Binary Search Tree. When this function is called data value in a node will be less than the data values in its left subtree and greater than the data values in its right subtree. If it is applied again to the mirrored tree, the original Binary Search Tree is obtained. Time complexity of this algorithm is *O(n).*