Assignment 2

Satyanand
14EC10049

Traversals

Inorder Traversal

Algorithm

- Traverse the left subtree, i.e., call Inorder(left-subtree)
- Visit the root.
- Traverse the right subtree, i.e., call Inorder(right-subtree)

Preorder Traversal

Algorithm

- Visit the root.
- Traverse the left subtree, i.e., call Inorder(left-subtree)
- Traverse the right subtree, i.e., call Inorder(right-subtree)

Postorder Traversal

Algorithm

- Traverse the left subtree, i.e., call Inorder(left-subtree)
- Traverse the right subtree, i.e., call Inorder(right-subtree)
- Visit the root.

Analysis of Traversals

Here one can notice that each node if visited only once. There is constant time overhead for visiting one node hence in concurs that all traversals have O(n) time complexity.

Maximum sum path from root to a leaf

Algorithm

We can solve this problem recursively by passing the sum of all nodes from root to each node as parameter to the recurive calls to its child.If the node is a leaf then we check if this has the sum greater than such other nodes.If it is so then we update the maximum sum.To be able to output the path we just need the leaf node of the maximum sum path.With that we can go backward to the root hence obtaining the path.

- Start by assigning $max\_sum = INT_MIN$
- Call the routine maxPath() for root of the tree with $current\_sum$=0
- recursively call it for its both child by adding this node's sum to $current\_sum$
- If the node is a leaf node compare this to max_sum
- If it is greater than max_sum update the max_sum and update the leaf node

Analysis

Here the equation for recursive call is T(n) = T(left child) + T(right child) + O(1) But since each node is visited only once and there is constant time required for each node's operation the time complexity of finding target leaf is O(n).

The path printing subroutine visits each node of the path once and requires constant operation for each node it also has time complexity of O(n).

Maximum sum path between any two nodes

Algorithm

We can solve this question also recursively. First let us see for a node the ways the path goes through the node

1. Node only
2. Max path through Left Child + Node

3. Max path through Right Child + Node
4. Max path through Left Child + Node + Max path through Right Child

To obtain the path back we just need three nodes

1. the highest node in the path
2. the lowest node in the left path from the highest node
3. the lowest node in the right path from the highest node

To regenerate the path we start from the left lowest node and reach top node going upwareds and the from right lowest reach top node but now printing in the top down manner recursively.

Analysis

The complexity is the same as in previous problem. Here the equation for recursive call is T(n) = T(left child) + T(right child) + O(1) But since each node is visited only once and there is constant time required for each node's operation the time complexity of finding target leaf is O(n).

The path printing subroutine visits each node of the path once and requires constant operation for each node it also has time complexity of O(n).