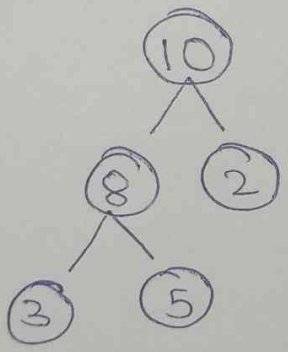
Haasha.M
1RV24MC041
I sem MCA 'A'

Largest SubTree Sum

The Largest subtree sum in a binary tree refers to the maximum sum obtained by adding up all the node value within a subtree of the tree. A subtree is any node along with all its descendant nodes in the tree.

1)



sub tree sum at 4 : 3
sub tree sum at 5 : 3
sub tree sum at 2 ; 8+3+5 = 16
sub tree sum at 3 ; 2
sub tree sum at 1 ; 10+16+2=28

Largest subtree sum = 28

∴, are positive, the largest subtree sum is equal to sum of all tree elements

To find the largest subtree sum in a binary tree follow these steps:

Step1: Understand the problem

- we need to find the maximum subtree sum in the entire tree.

Step2: Use postorder Traversal

- follow the order left->right->root, recursively calculate subtree sums and update the maximum sum found.

step3: Implement the Algorithm

1. Base case: if a node is null return 0
2. Recursive calls:- calculate the sum of tree
3. Compute current subtree sum:

   Sum = node value + left subtree Sum + right subtree sum.
4. update Maximum sum; keep highest subtree sum.
5. Return the subtree sum: Pass the sum up to the stack.

Application:

- used in data analysis.
- helps in optimizing tree structures in memory management
- used in network analysis to determine the most connected subnetworks

Code in java

```
Class TreeNode {
    int val;
    TreeNode left, right;
    Tree Node(int val){
        this.val=val;
        this.left=null;
        this.right=null;
    }
}
class Solution {
    Private int maxSum=Integer.MIN-VALUE;
    Public int longestSubtreeSum(Tree Node
    root){
     calculateSubtreeSum(root);
      return maxSum;
     }
     Private int calculateSubtreeSum(TreeNode
     node){
     if(node==null)return 0;
     int leftSum=calculateSubtreeSum(node.left);
     int rightSum=calculateSubtreeSum(node.
       right);
     int currentsum=node.val+leftSum+rightSum
     maxSum=Math.max(maxSum, currentSum);
     return currentsum;
}
```

```
Public static void main(String[] args){
    TreeNode root = new TreeNode(1);
    root.left = new TreeNode(2);
    root.right = new TreeNode(3)
    root.leftleft = new TreeNode(4);
    root.left.right=new TreeNode(5);

    Solution solution=new Solution();
    int result =Solution.largestsubtreeSum(roo
    System.out.println("Largest subtree sum:" +
      result);
  }
}
```