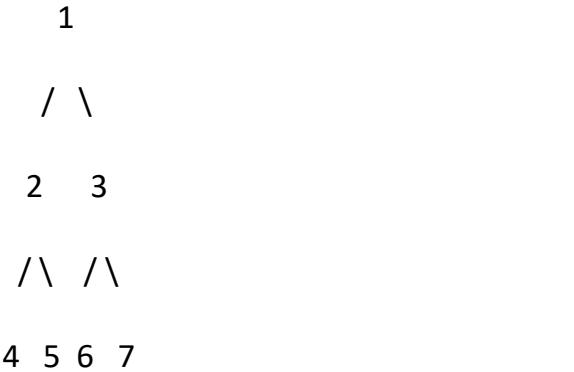
# Assignment 20 Solution - Tree| DSA

**Question-1**

Given a binary tree, your task is to find subtree with maximum sum in tree.

Examples:

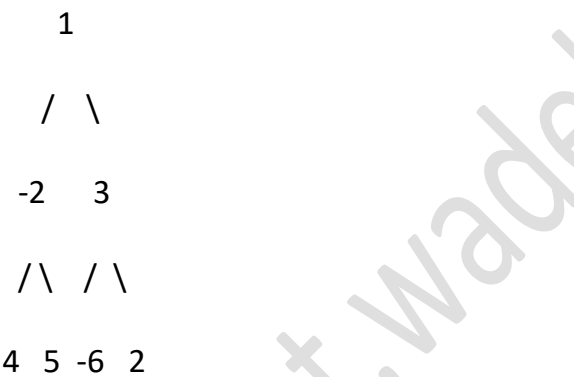
Input1 :



Output1 : 28

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

Input2 :



Output2 : 7

Subtree with largest sum is :

Also, entire tree sum is also 7.

**Solution Code:**

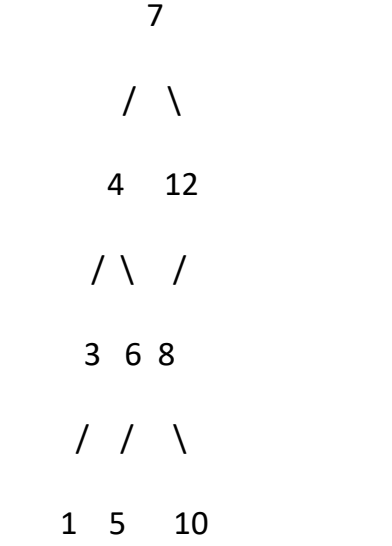
```
package in.ineuron.pptAssignment20;

//Definition of a binary tree node
class TreeNode {
        int val;
```

```
        TreeNode left;
        TreeNode right;

        TreeNode(int val) {
            this.val = val;
            left = null;
            right = null;
        }
    }

    class Result {
        int maxSum;

        Result(int maxSum) {
            this.maxSum = maxSum;
        }
    }

    class BinaryTree {
        private TreeNode root;

        BinaryTree(TreeNode root) {
            this.root = root;
        }

        private int findMaxSubtreeSum(TreeNode node, Result result) {
            if (node == null)
                return 0;

            // Recursively calculate sum of left and right subtrees
            int leftSum = findMaxSubtreeSum(node.left, result);
            int rightSum = findMaxSubtreeSum(node.right, result);

            // Update the maximum sum if the current subtree's sum is greater
            int currentSum = node.val + leftSum + rightSum;
            result.maxSum = Math.max(result.maxSum, currentSum);

            // Return the sum of the current subtree
            return currentSum;
        }

        int findMaxSubtreeSum() {
            Result result = new Result(Integer.MIN_VALUE);
            findMaxSubtreeSum(root, result);
            return result.maxSum;
        }
    }
```

```
public class MaxSubtreeSum_1 {
    public static void main(String[] args) {
        // Create the binary tree
        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.left.left = new TreeNode(4);
        root.left.right = new TreeNode(5);
        root.right.left = new TreeNode(6);
        root.right.right = new TreeNode(7);

        // Create an instance of BinaryTree and find the maximum subtree sum
        BinaryTree tree = new BinaryTree(root);
        int maxSum = tree.findMaxSubtreeSum();
        System.out.println("Maximum subtree sum: " + maxSum);
    }
}
```

**Question-2**

Construct the BST (Binary Search Tree) from its given level order traversal.
Example:
Input: arr[] = {7, 4, 12, 3, 6, 8, 1, 5, 10}
Output: BST:




**Solution Code:**

```
package in.ineuron.pptAssignment20;

//Definition of a binary tree node
class TreeNode02 {
	int val;
	TreeNode02 left;
	TreeNode02 right;

	TreeNode02(int val) {
		this.val = val;
		left = null;
		right = null;
	}
}

class BSTConstruction {
	TreeNode02 constructBST(int[] arr) {
		if (arr == null || arr.length == 0)
			return null;

		TreeNode02 root = new TreeNode02(arr[0]);
		for (int i = 1; i < arr.length; i++)
			insertNode(root, arr[i]);

		return root;
	}
```

```
        void insertNode(TreeNode02 root, int value) {
            if (value < root.val) {
                if (root.left == null)
                    root.left = new TreeNode02(value);
                else
                    insertNode(root.left, value);
            } else {
                if (root.right == null)
                    root.right = new TreeNode02(value);
                else
                    insertNode(root.right, value);
            }
        }

        void printBST(TreeNode02 root) {
            if (root == null)
                return;

            printBST(root.left);
            System.out.print(root.val + " ");
            printBST(root.right);
        }
    }

    public class BinarySearchTreeContr_2 {
        public static void main(String[] args) {
            int[] arr = { 7, 4, 12, 3, 6, 8, 1, 5, 10 };

            BSTConstruction bst = new BSTConstruction();
            TreeNode02 root = bst.constructBST(arr);

            System.out.println("BST:");
            bst.printBST(root);
        }
    }
```

**Question-3**

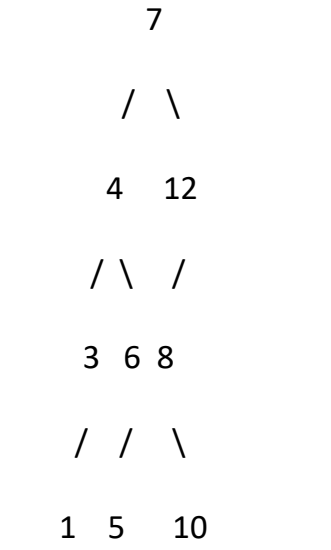
Given an array of size n. The problem is to check whether the given array can represent the level order traversal of a Binary Search Tree or not.

Examples:

Input1 : arr[] = {7, 4, 12, 3, 6, 8, 1, 5, 10}

Output1 : Yes

For the given arr[], the Binary Search Tree is:




Input2 : arr[] = {11, 6, 13, 5, 12, 10}

Output2 : No

The given arr[] does not represent the level order traversal of a BST.

**Solution Code:**

```
package in.ineuron.pptAssignment20;

import java.util.LinkedList;
import java.util.Queue;

class BSTLevelOrderValidation {
	boolean isLevelOrderBST(int[] arr) {
		if (arr == null || arr.length == 0)
			return true;

		int n = arr.length;
		Queue<Integer> queue = new LinkedList<>();
		queue.offer(arr[0]);

		int i = 1;
		while (i < n) {
			int current = queue.poll();

			// Find the index where the right subtree starts
			int index = -1;
			for (int j = i; j < n; j++) {
```

```
                        if (arr[j] > current) {
                            index = j;
                            break;
                        }
                    }

                // Check if the elements in the right subtree are greater than the current node
                    if (index != -1) {
                        for (int j = index; j < n; j++) {
                            if (arr[j] < current)
                                return false;
                        }

                        // Enqueue the right subtree elements
                        for (int j = index; j < n; j++)
                            queue.offer(arr[j]);

                        i = index + 1;
                    } else {
                        // Enqueue the remaining elements
                        for (int j = i; j < n; j++)
                            queue.offer(arr[j]);

                        break;
                    }
                }

                return true;
            }
        }

        public class BSTOrderTraversal_3 {
            public static void main(String[] args) {
                int[] arr = { 7, 4, 12, 3, 6, 8, 1, 5, 10 };

                BSTLevelOrderValidation bst = new BSTLevelOrderValidation();
                boolean isValid = bst.isLevelOrderBST(arr);

                System.out.println("Can represent level order traversal of a BST? " + (isValid ?
                "Yes" : "No"));
            }
        }
```

**GitHub Link :** https://github.com/devavratwadekar/ineuron_ppt_ProgramAssignmentCode/tree/master/PPTAssignment-20