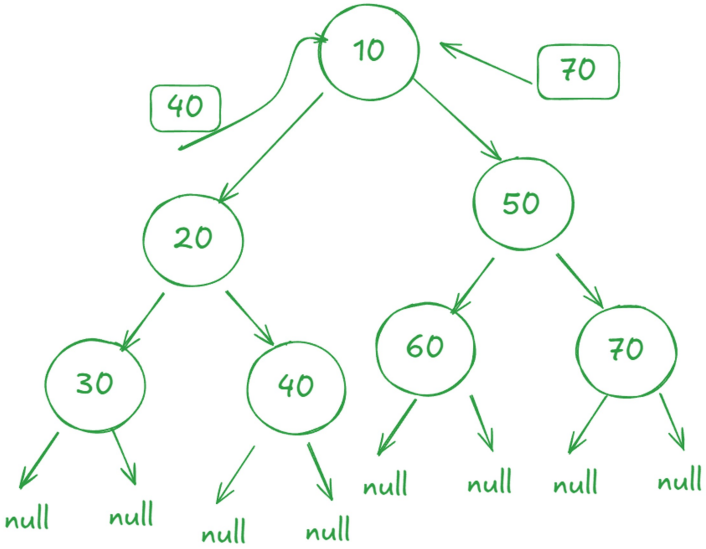
Introduction To Binary Tree [Day 18]



94. Binary Tree Inorder Traversal Solved

Easy Topics Companies

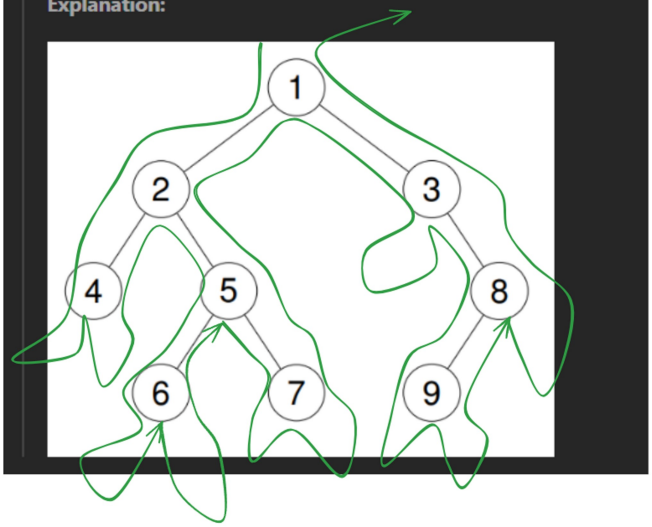
Given the `root` of a binary tree, return *the inorder traversal of its nodes' values.*

Example 2:

Input: `root = [1,2,3,4,5,null,8,null,null,6,7,9]`

Output: `[4,2,6,5,7,1,3,9,8]`

Explanation:



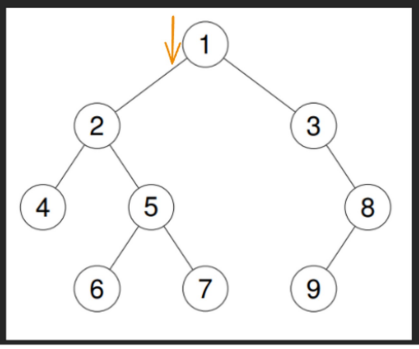
list.add(node.data)

```java
public List<Integer> inorderTraversal(TreeNode root) {
    List<Integer> result = new ArrayList<>();
    inorderHelper(root , result);
    return result;
}

private void inorderHelper(TreeNode node , List<Integer> result){
    if(node != null){
        inorderHelper(node.left , result);
        result.add(node.val);
        inorderHelper(node.right , result);
    }
}
```

Input: root = [1,2,3,4,5,null,8,null,null,6,7,9]

Output: [4,2,6,5,7,1,3,9,8]

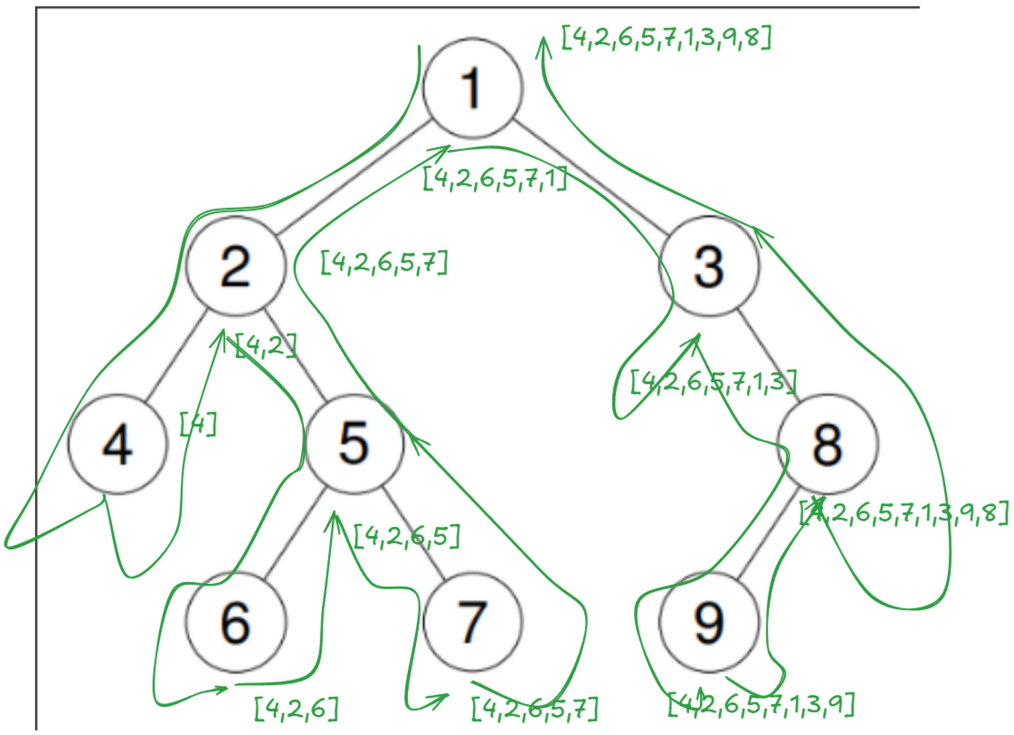
Explanation:



inorder(1,[4,2,6,5,7,1, 3, 9 ,8],3)

result 4 2 6 5 7 1 3 9 , 8

Call Stack



144. Binary Tree Preorder Traversal

Solved

Easy | Topics | Companies

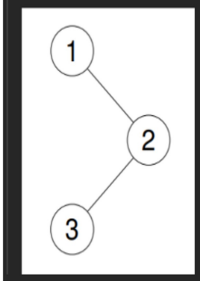
Given the `root` of a binary tree, return *the preorder traversal of its nodes' values.*

Example 1:

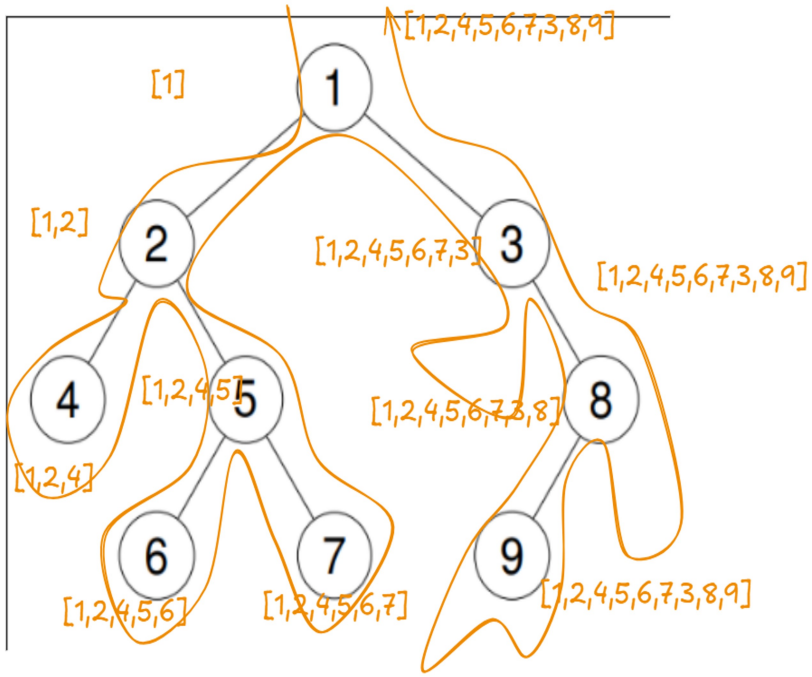
Input: `root = [1,null,2,3]`

Output: `[1,2,3]`

Explanation:



Example 2:




```java
class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> result = new ArrayList<>();
        preorderHelper(root, result);
        return result;
    }
    private void preorderHelper(TreeNode node, List<Integer> result){
        if(node!= null){
            result.add(node.val);
            preorderHelper(node.left , result);
            preorderHelper(node.right , result);
        }
    }
}
```

145. Binary Tree Postorder Traversal

Solved

Easy | Topics | Companies

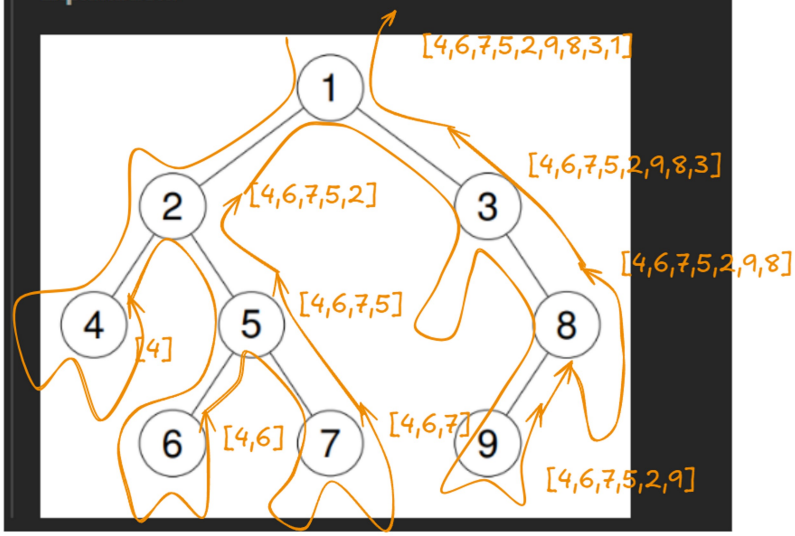
Given the `root` of a binary tree, return *the postorder traversal of its nodes' values.*

Example 2:

Input: root = [1,2,3,4,5,null,8,null,null,6,7,9]

Output: [4,6,7,5,2,9,8,3,1]

Explanation:




```java
public List<Integer> postorderTraversal(TreeNode root) {
    List<Integer> result = new ArrayList<>();
    postOrderHelper(root , result);
    return result;
}

private void postOrderHelper(TreeNode node , List<Integer> result){
    if(node == null){
        return;
    }
    postOrderHelper(node.left , result);
    postOrderHelper(node.right , result);
    result.add(node.val);
}
```

Pre

node.val
left
right

Inorder

left
node.val
right

PostOrder

Left
Right
node.val

104. Maximum Depth of Binary Tree

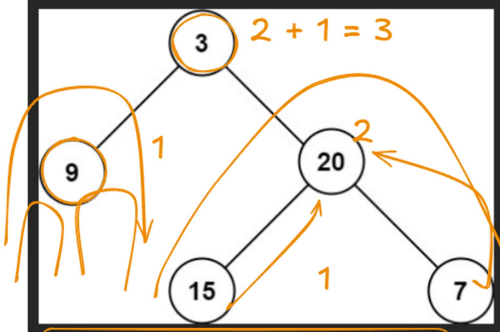
Solved

Easy | Topics | Companies

Given the `root` of a binary tree, return *its maximum depth*.

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

Example 1:



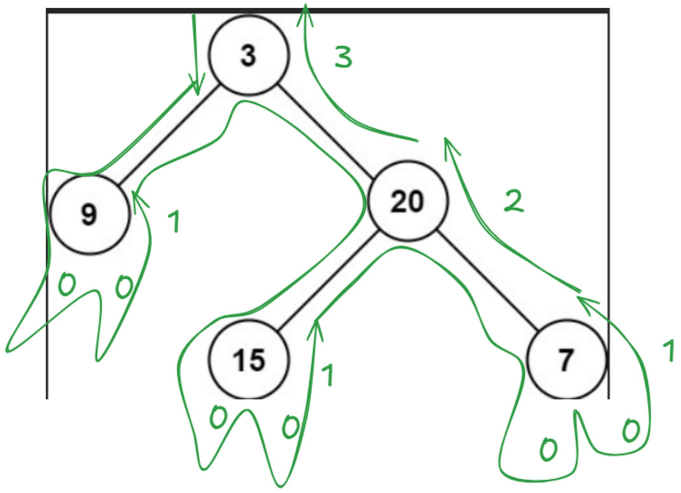
Max(left,right) + 1;

Input: root = [3,9,20,null,null,15,7]
Output: 3

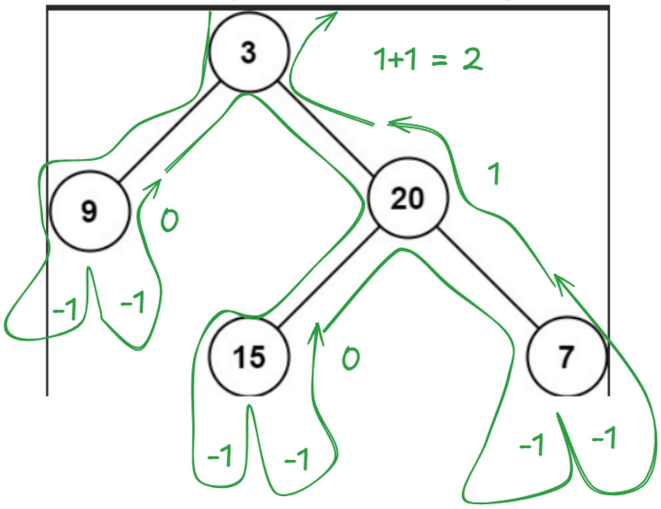
```java
class Solution {
    public int maxDepth(TreeNode root) {
        if(root == null){
            return 0;
        }
        int leftDepth = maxDepth(root.left);
        int rightDepth = maxDepth(root.right);

        return Math.max(leftDepth , rightDepth) + 1;
    }
}
```

-1

Node Depth



Depth Based on Edges



0 = Math.max(-1,-1) + 1 = 0

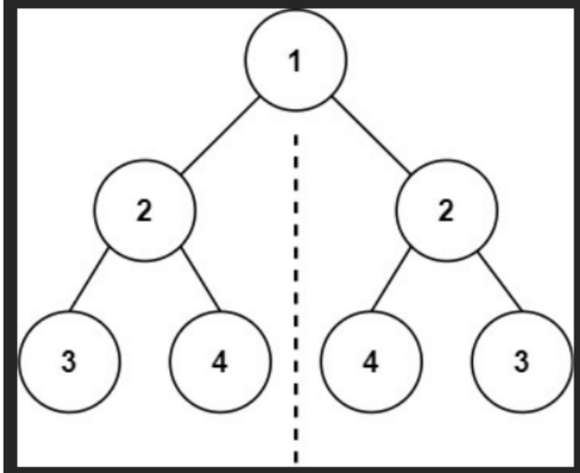
101. Symmetric Tree

Solved

Easy | Topics | Companies

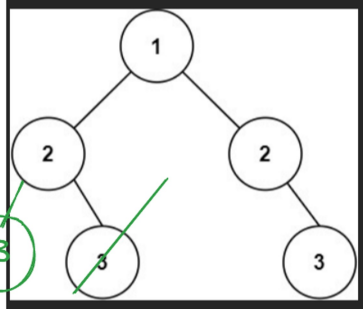
Given the `root` of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

Example 1:




```
Input: root = [1,2,2,3,4,4,3]
Output: true
```

Example 2:



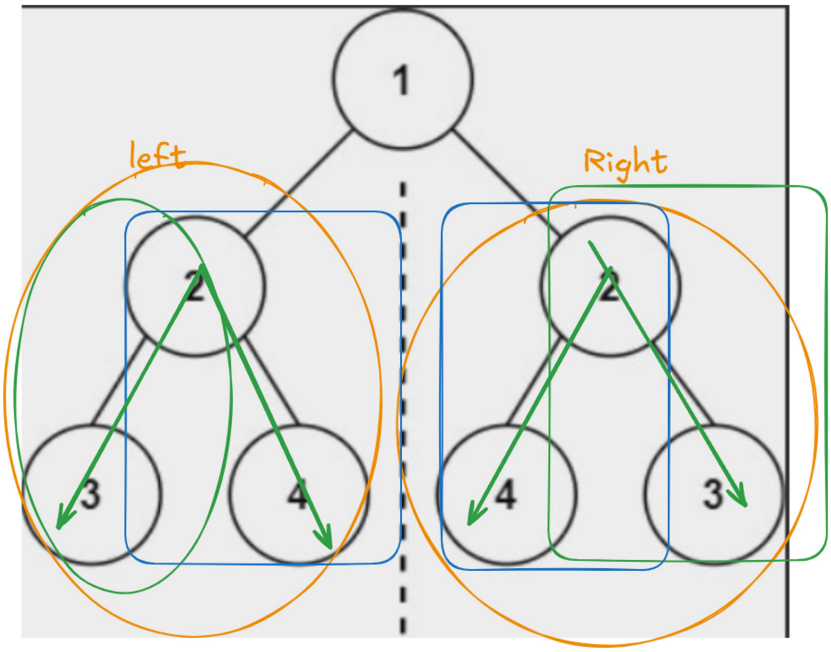

```
Input: root = [1,2,2,null,3,null,3]
Output: false
```

if 3 is on other side it can be symmetric.

Change Status

Completed

Attendance Code: 5EE175CA

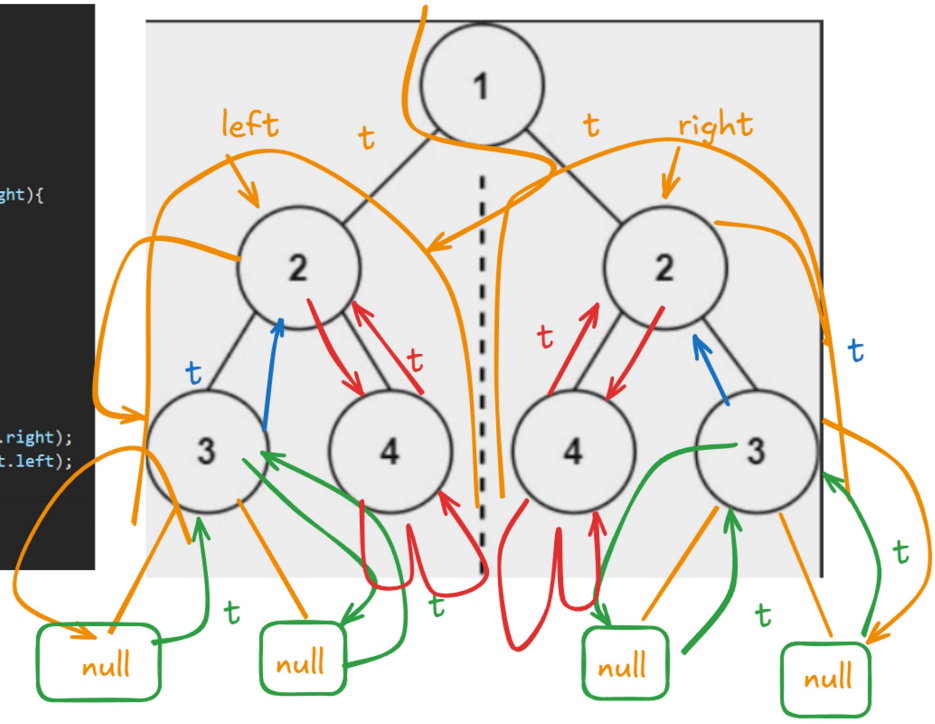


Mirror(left , right)

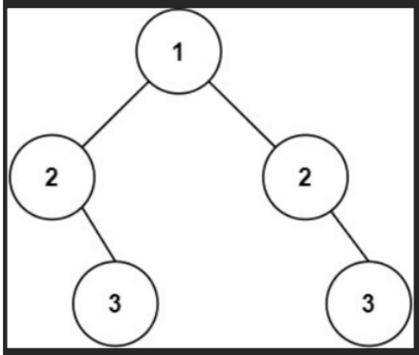
outermirror(left.left , right.right)

innerMirror(left.right , right.left)

```java
public boolean isSymmetric(TreeNode root) {
    if(root == null){
        return true;
    }
    return isMirror(root.left , root.right);
}

private boolean isMirror(TreeNode left , TreeNode right){
    if(left == null && right == null){
        return true;
    }
    if(left == null || right == null){
        return false;
    }
    if(left.val != right.val){
        return false;
    }
    boolean outerMirror = isMirror(left.left , right.right);
    boolean innerMirror = isMirror(left.right , right.left);

    return outerMirror && innerMirror;
}
```



Example 2:



Input: root = [1,2,2,null,3,null,3]
Output: false