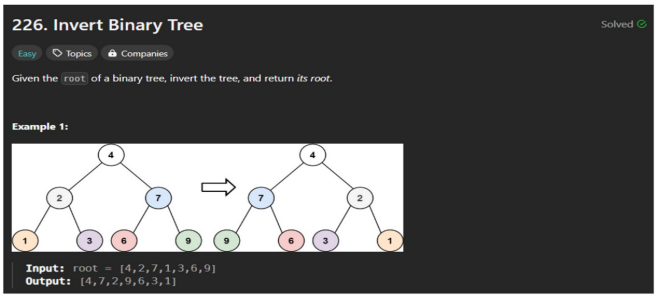
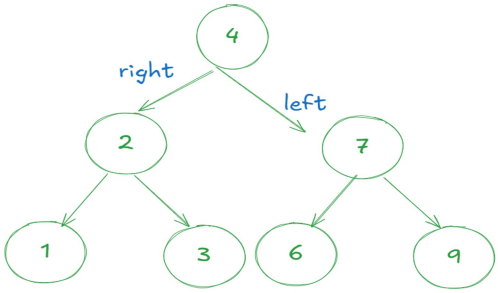
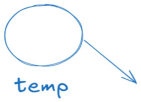
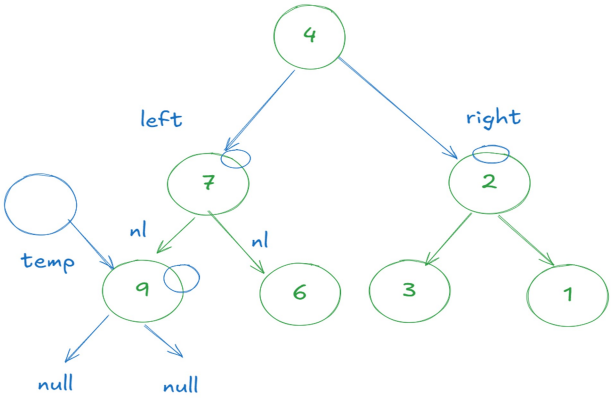
Binary Tree + Binary Search Tree [Day 19]






```
temp = root.left;
root.left = root.right
root.right = temp;
```





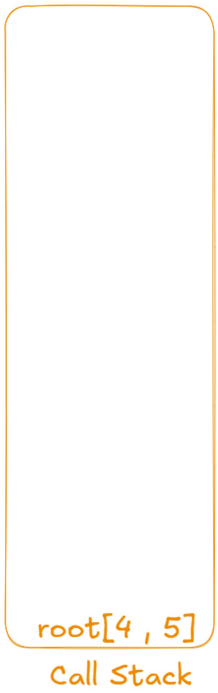
```java
class Solution {
    public TreeNode invertTree(TreeNode root) {
        if(root == null){
            return null;
        }

        TreeNode temp = root.left;
        root.left = root.right;
        root.right = temp;

        invertTree(root.left);
        invertTree(root.right);
        return root;
    }
}
```

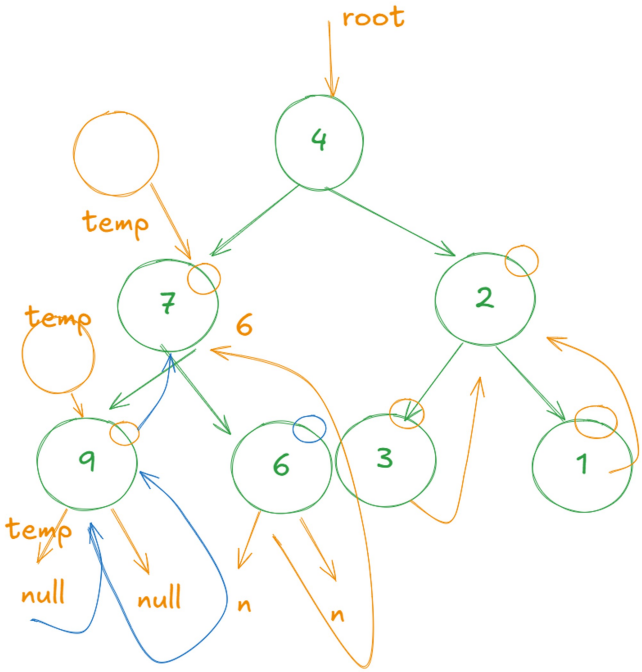
1

2

3

4





return root.

112. Path Sum

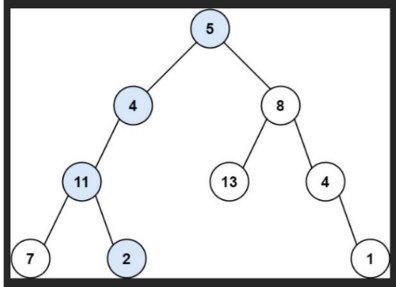
Solved

Easy Topics Companies

Given the `root` of a binary tree and an integer `targetSum`, return `true` if the tree has a **root-to-leaf** path such that adding up all the values along the path equals `targetSum`.

A **leaf** is a node with no children.

Example 1:

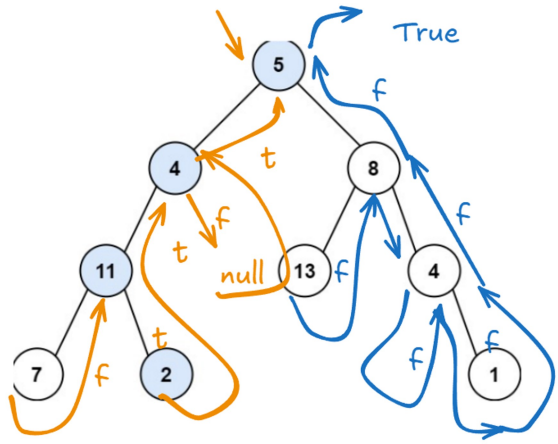


Input: root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22
Output: true
Explanation: The root-to-leaf path with the target sum is shown.

Short Circuit :

And Logical - F && (F/T*) -- F

OR Logical - T || (T/F*) -- T



targetSum = 22

```
if(root == null){
    return false;   1
}
if(root.left == null && root.right == null){
    return root.val == targetSum;          2
}
int remainingSum = targetSum - root.val;   3
return hasPathSum(root.left , remainingSum) || hasPathSum(root.right , remainingSum);
                  4                                    5
```

root(5,22,5)

Call Stack

Work

Remaining Sum = 5

target - root.val

100. Same Tree

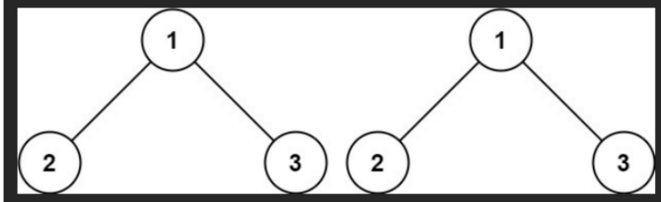
Solved

Easy | Topics | Companies

Given the roots of two binary trees `p` and `q`, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

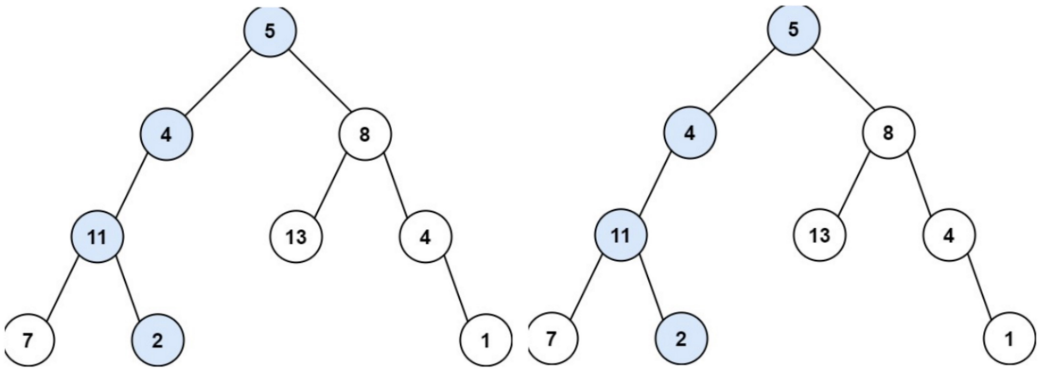
Example 1:



Input: p = [1,2,3], q = [1,2,3]
Output: true

```
class Solution {
    public boolean isSameTree(TreeNode p, TreeNode q) {
        if(p == null && q == null){
            return true;
        }
        if(p == null || q == null){
            return false;
        }
        if(p.val != q.val){
            return false;
        }
        return isSameTree(p.left , q.left) && isSameTree(p.right , q.right);
    }
}
```

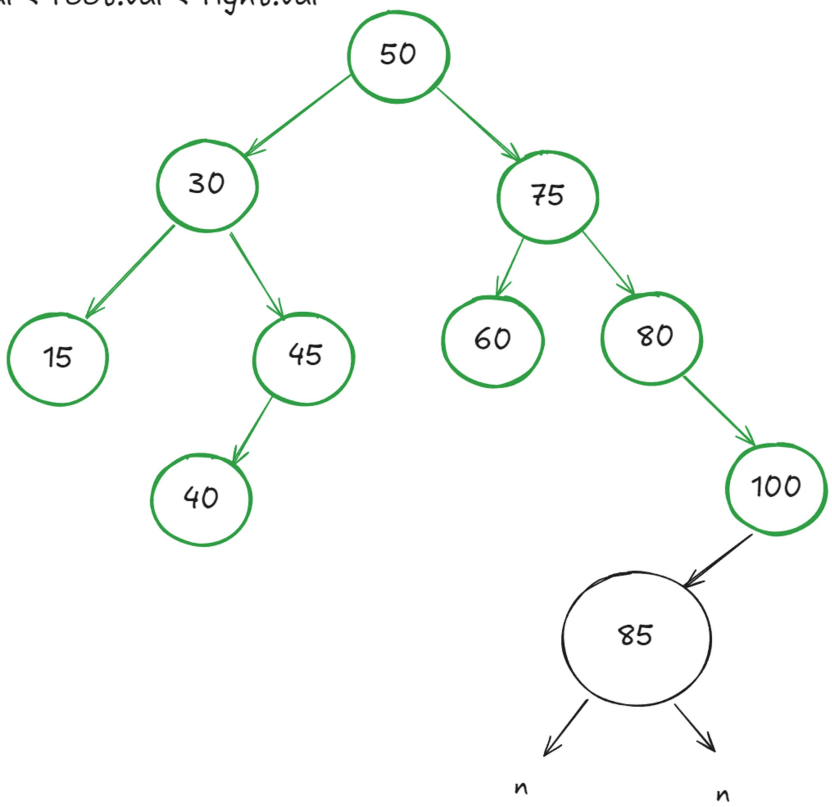
1 2 3 4 5



Introduction To BST [Binary Search Tree]

At most 2 child

left.val < root.val < right.val



701. Insert into a Binary Search Tree

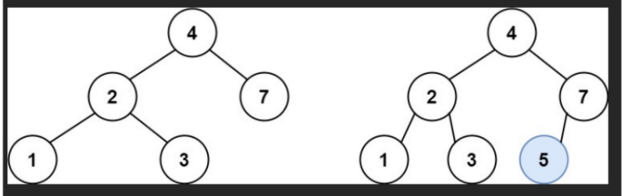
Solved

Medium | Topics | Companies

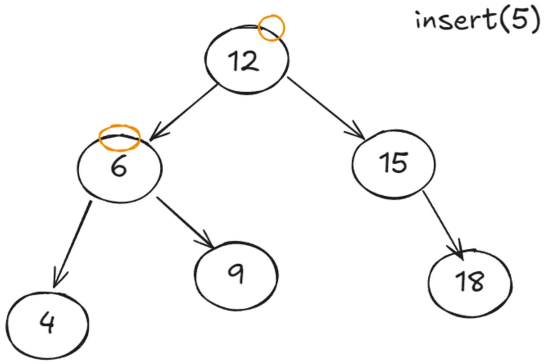
You are given the `root` node of a binary search tree (BST) and a `value` to insert into the tree. Return *the root node of the BST after the insertion*. It is **guaranteed** that the new value does not exist in the original BST.

Notice that there may exist multiple valid ways for the insertion, as long as the tree remains a BST after insertion. You can return **any of them**.

Example 1:




```
Input: root = [4,2,7,1,3], val = 5
Output: [4,2,7,1,3,5]
```



```java
public TreeNode insertIntoBST(TreeNode root, int val) {
    if(root == null){
        return new TreeNode(val);
    }
    if(val < root.val){
       root.left = insertIntoBST(root.left , val);
    }else if(val > root.val){
        root.right = insertIntoBST(root.right , val);
    }
    return root;
}
```

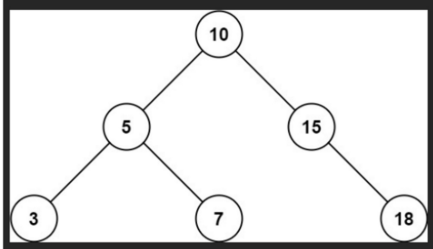
938. Range Sum of BST

Solved

Easy | Topics | Companies

Given the `root` node of a binary search tree and two integers `low` and `high`, return *the sum of values of all nodes with a value in the* ***inclusive*** *range* `[low, high]`.

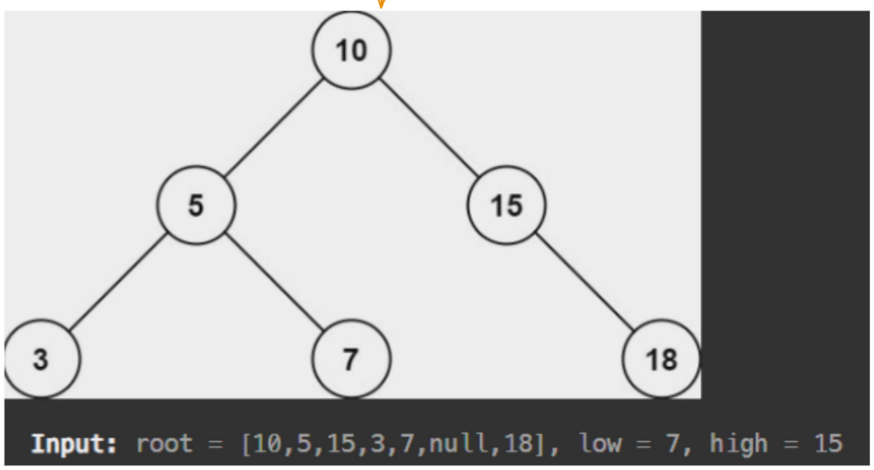
Example 1:



Input: root = [10,5,15,3,7,null,18], low = 7, high = 15
Output: 32
Explanation: Nodes 7, 10, and 15 are in the range [7, 15]. 7 + 10 + 15 = 32.

root reaches to null return 0;

sum = 32



```java
class Solution {
    public int rangeSumBST(TreeNode root, int low, int high) {
        if(root == null){
            return 0;
        }
        int sum = 0;
        if(root.val >= low && root.val <= high){
            sum += root.val;
        }

        if(root.val > low){
            sum += rangeSumBST(root.left , low , high);
        }

        if(root.val < high){
            sum += rangeSumBST(root.right , low , high);
        }

        return sum;
    }
}
```

Change Status

Completed

Attendance Code: 27F80679

700. Search in a Binary Search Tree

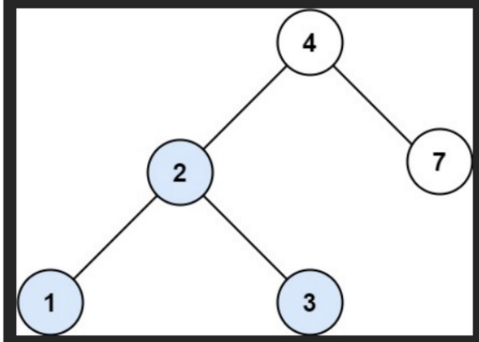
Solved

Easy | Topics | Companies

You are given the `root` of a binary search tree (BST) and an integer `val`.

Find the node in the BST that the node's value equals `val` and return the subtree rooted with that node. If such a node does not exist, return `null`.

Example 1:



Input: root = [4,2,7,1,3], val = 2
Output: [2,1,3]

```java
*/
class Solution {
    public TreeNode searchBST(TreeNode root, int val) {
        if(root == null || root.val == val){
         return root;
        }
        if(val < root.val){
         return searchBST(root.left , val);
        }

        return searchBST(root.right , val);
    }
}
```