Search in a Binary Search Tree (Day 47)

Problem | Submissions | Leaderboard | Discussions

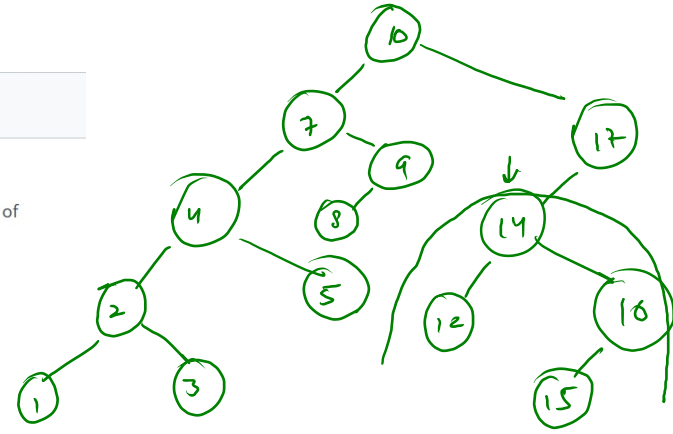
You are given preorder of the Binary search tree construct BST , Now you have root of a binary search tree (BST) and an integer val. Find the node in the BST that the node's value equals val and return the level order of the subtree rooted with that node. If such a node does not exist, return null.

Input Format

11

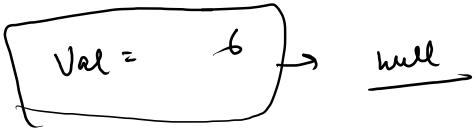
4 2 1 n n 3 n n 7 n n

2

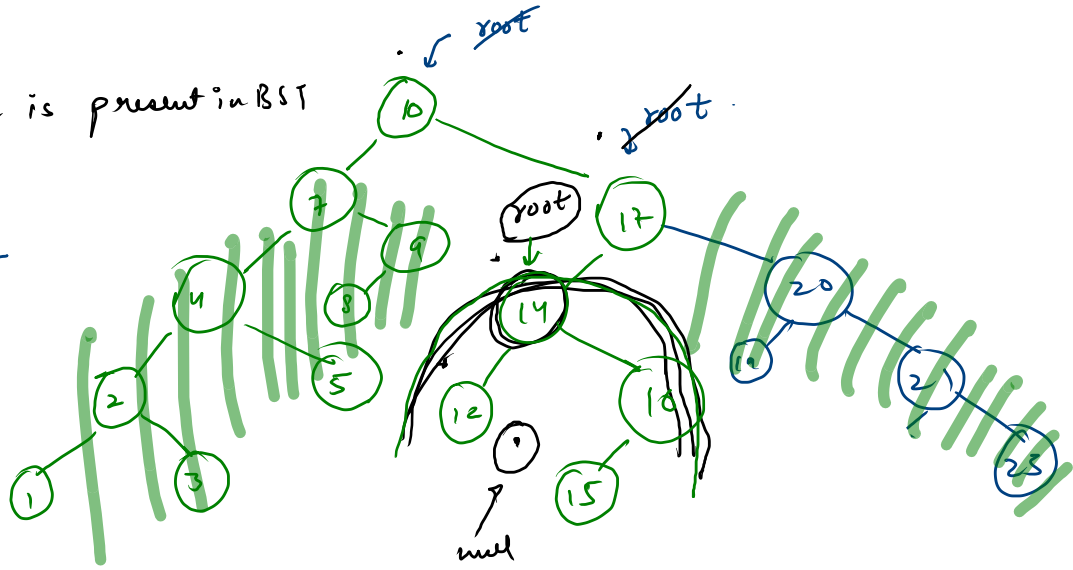


Val = 14

14 12 16 15



Val = 14 → when val is present in BST



14 > 10 → right

14 < 17 → left

14 == 14 ✓

Val = 13

if (root == null)

return null.


```java
public static Node serachInBST(Node root, int val){
    if(root == null){
        return null;
    }

    if(root.data == val){
        return root;
    }
    else if(root.data>val){
        return serachInBST(root.left,val);
    } else {
        return serachInBST(root.right,val);
    }
}
```

1038. Binary Search Tree to Greater Sum Tree

Hint

Medium 3.6K 151

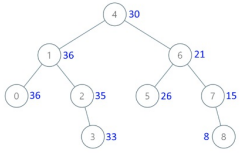
eBay Amazon Facebook ...

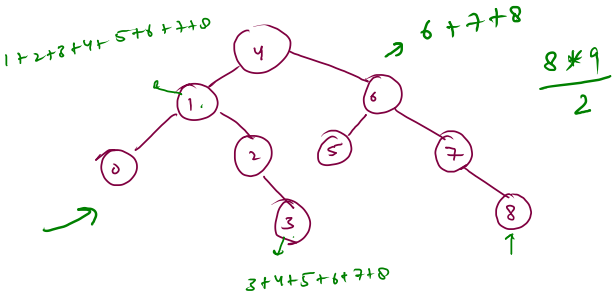
Given the `root` of a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus the sum of all keys greater than the original key in BST.

As a reminder, a *binary search tree* is a tree that satisfies these constraints:

- The left subtree of a node contains only nodes with keys **less than** the node's key.
- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

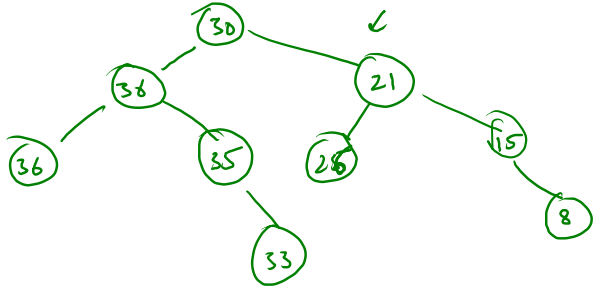
Example 1:

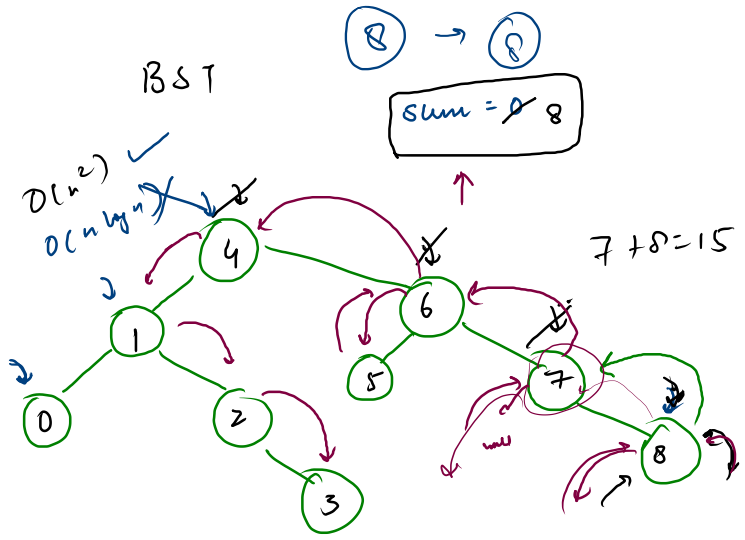




BST → GST

GST

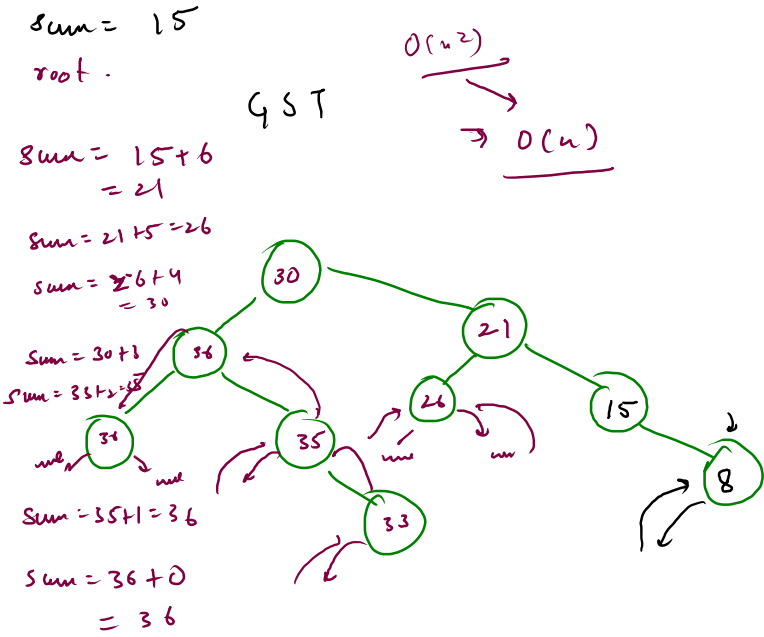


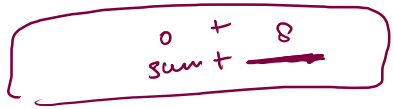


(1) Reach to the rightmost node.

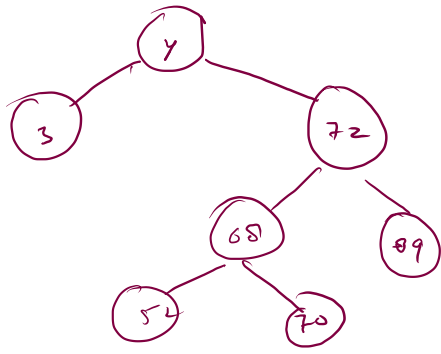
(2) (a) sum += root.val
(b) root.val = sum
→ update sum variable, update root.val.

(3) call to left side

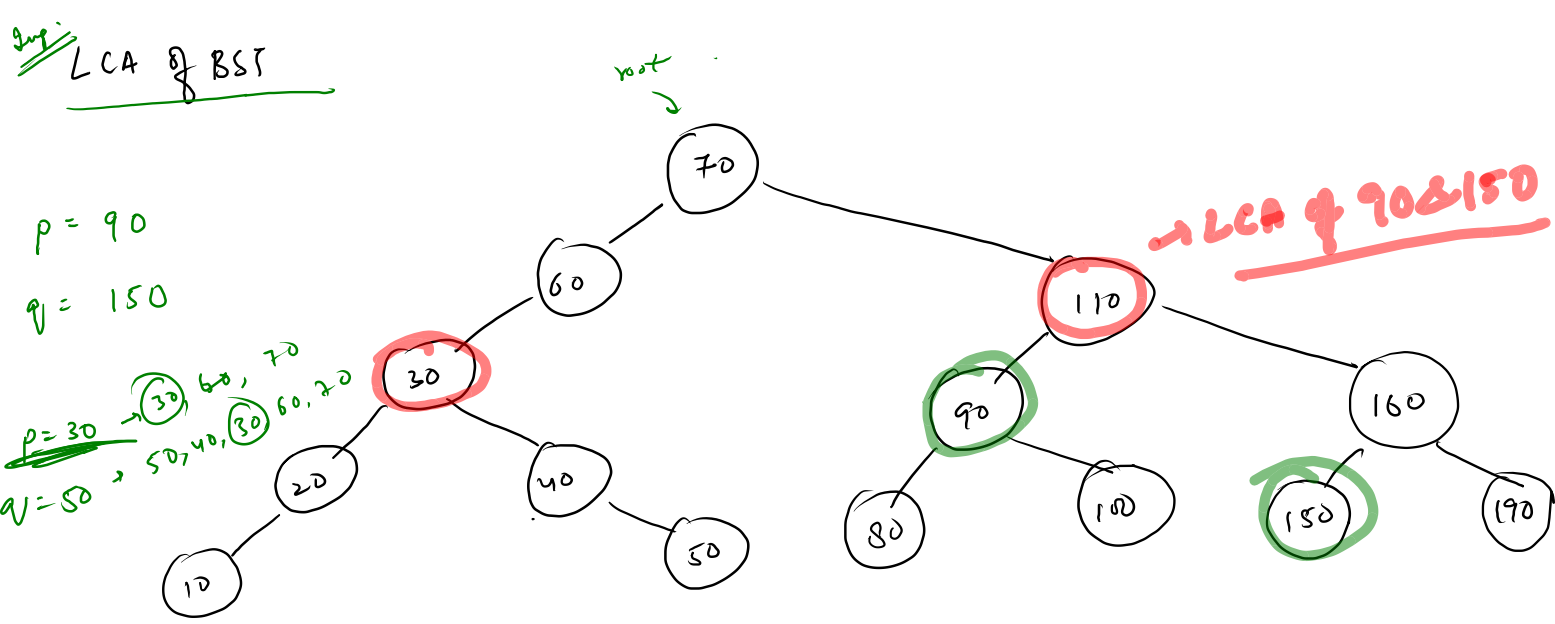





```java
class Solution {
    public static int sum ;

    public static void helper(TreeNode root) {
        if(root == null) return;

        helper(root.right);
        root.val = sum;
        helper(root.left);
    }
    public TreeNode bstToGst(TreeNode root) {
        sum = 0;
        helper(root);
        return root;
    }
}
```



LCA of BST



p = 90

q = 150

p = 30 → 30, 60, 70

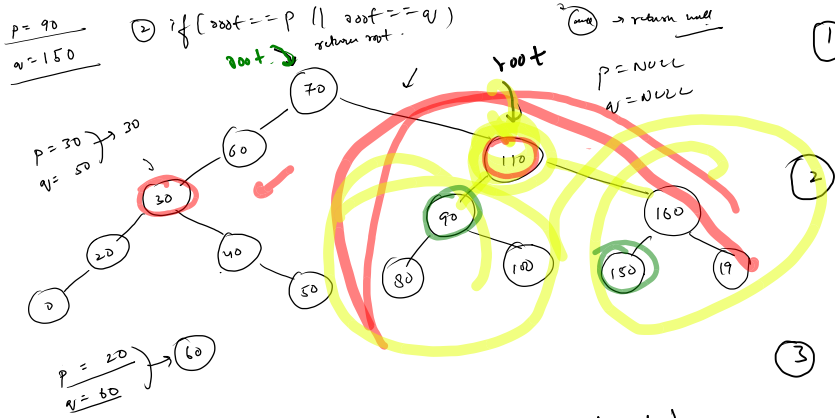
q = 50 → 50, 40, 30, 60, 70

90 110 70

150 160 110 70

p = 50

q = 200 → null



① When root is null
→ return null;

② When root is equal to either 'p' or 'q'
↳ return root.

③ When root != null but p == NULL || q == NULL
↳ return NULL

if (p.data < root.data && q.data < root.data)
return LCA (root.left, p, q)

elseif (p.data > root.data && q.data > root.data)
return LCA (root.right, p, q)

else →
return root.

p = 90
q = 150

90 < 70 ✗
150

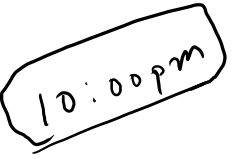
90 > 70 ✓

150 > 70 ✓

90 < 110 ✓

150 < 110 ✗

90 > 110 ✗

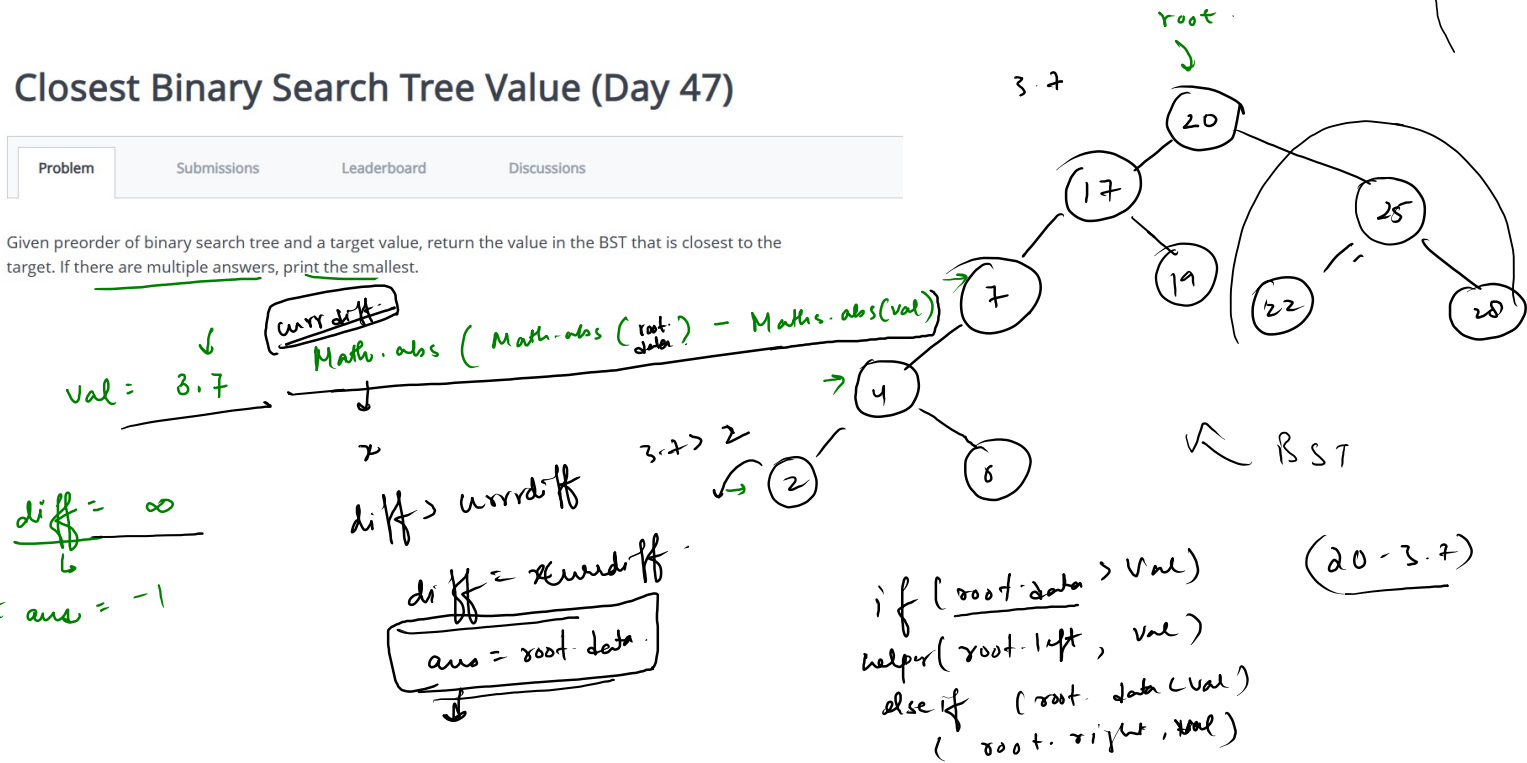


```java
public static Node LCA(Node root, Node p, Node q){
    if(root == null) {
        return root;
    }

    if(p == null || q == null) {
        return null;
    }

    if(p.data == root.data || q.data == root.data){
        return root;
    }

    if(p.data > root.data && q.data > root.data){
        return LCA(root.right,p,q);
    }
    else if(p.data <root.data && q.data <root.data){
        return LCA(root.left,p,q);
    }
    else {
        return root;
    }


}
```

Closest Binary Search Tree Value (Day 47)

Problem | Submissions | Leaderboard | Discussions

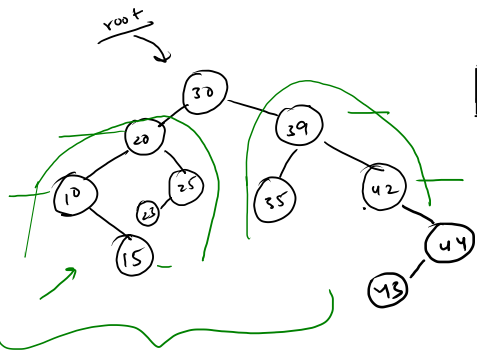
Given preorder of binary search tree and a target value, return the value in the BST that is closest to the target. If there are multiple answers, print the smallest.

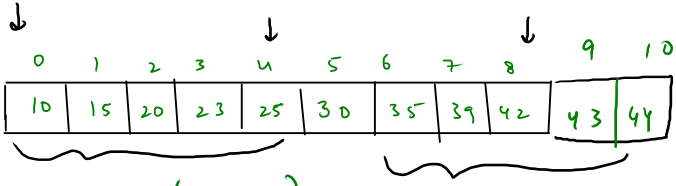


```java
public static void helper(Node root, double val) {
    if(root == null) return;

    double currDiff = Math.abs(Math.abs(root.data) - Math.abs(val));

    if(diff>currDiff){
        diff = currDiff;
        ans = root.data;
    }

    if(root.data > val){
        helper(root.left,val);
    } else if(root.data < val){
        helper(root.right,val);
    }
}

public static int closestNode(Node root, double val){
  diff = Double.MAX_VALUE;
    helper(root,val);
    return ans;
}
```

Sort

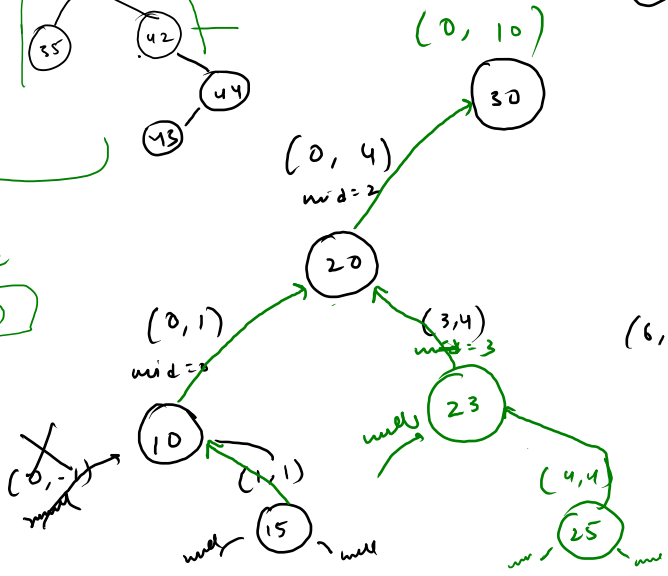
Construct BST using Inorder Traversal





Balancing Tree

$O(\log n)$



$0+1 = 0$

$mid = \frac{(0+10)}{2} = 5$

if (low > high)
return mid)

→ mid

→ (low, mid-1)

→ (mid+1, high)

(6, 10)
mid = 8
42

(6, 7)

(9, 10)

```java
public static Node helper(int in[], int low,int high){
    if(low > high) return null;

    int mid = (low+high)/2;

    Node node = new Node(in[mid],null,null);
    node.left = helper(in,low,mid-1);
    node.right = helper(in,mid+1,high);

    return node;
}

public static Node constructFromInorder(int [] inorder){
    int n = inorder.length;

    return helper(inorder,0,n-1);
}
```