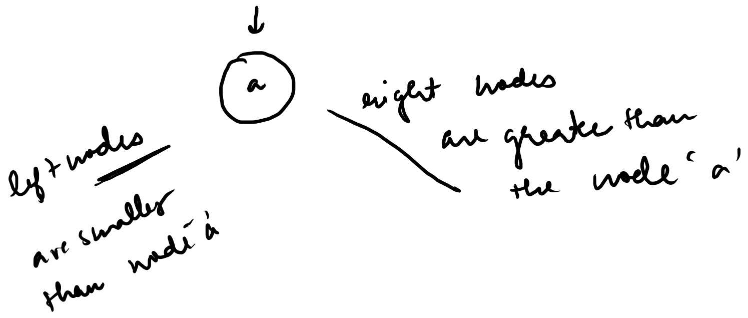
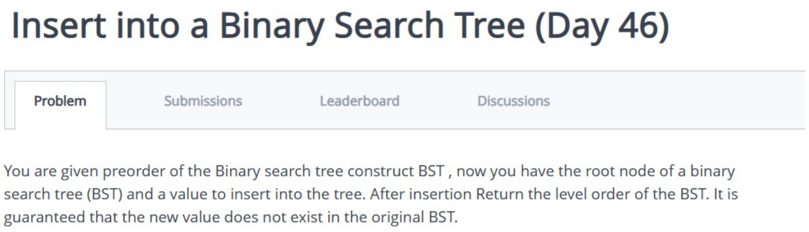
Starts @ 9:10pm

B.S.T

↳


a
right nodes are greater than the node 'a'
left nodes are smaller than node 'a'

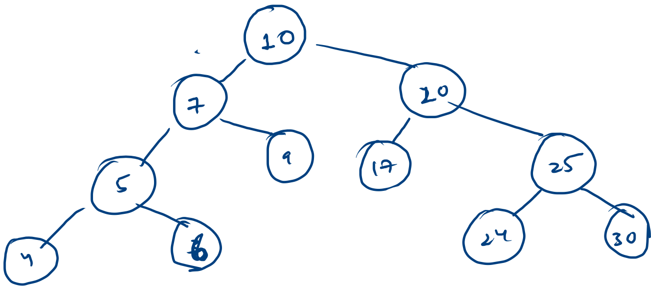
$$\begin{tikzcd}
\rightarrow O(n) \arrow[d] \\
\rightarrow O(\log n)
\end{tikzcd}$$



Insert into a Binary Search Tree (Day 46)

You are given preorder of the Binary search tree construct BST , now you have the root node of a binary search tree (BST) and a value to insert into the tree. After insertion Return the level order of the BST. It is guaranteed that the new value does not exist in the original BST.

Value ⇒ 18



8 → data

19

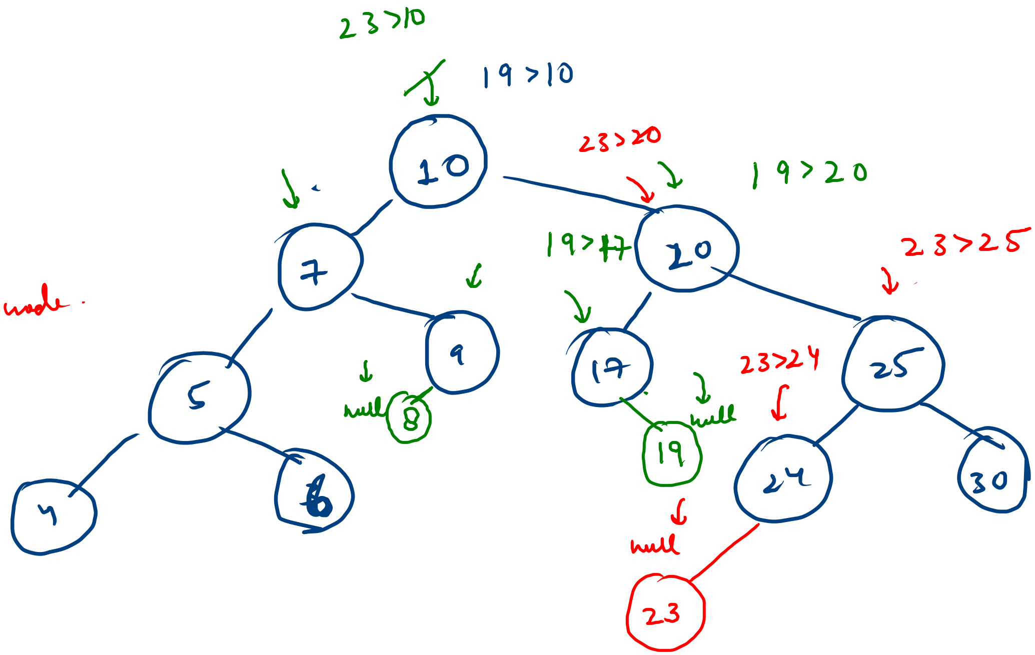
23 if (root == null)
return create a node.

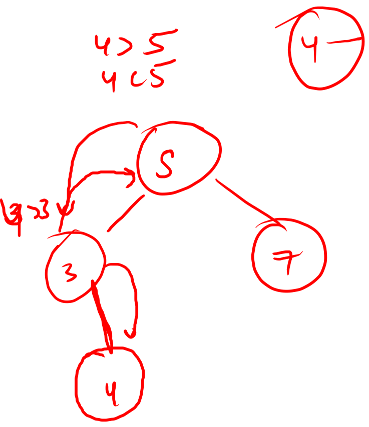
if (data > root.data)

→ right side

else if (data < root.data)

→ left side



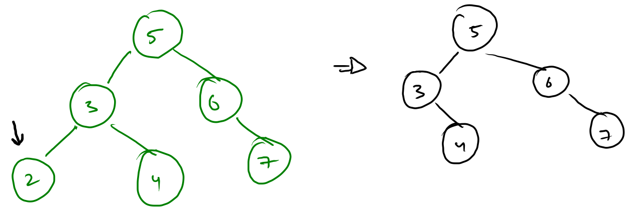


```
public static Node insertNode(Node root, int data){
    if(root == null){
        return new Node(data, null,null);
    }

    if(data>root.data){
        root.right = insertNode(root.right,data);
    } else if(data<root.data){
        root.left = insertNode(root.left,data);
    }

    return root;
}
```

Imp: Delete a node in BST

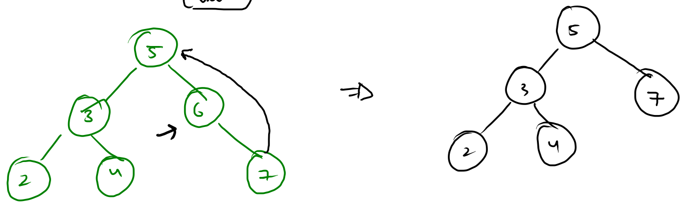
val = 2

Case-01 → When removing node is a leaf node.



Case-02 → When removing node has 1 child.

$\boxed{val=6}$

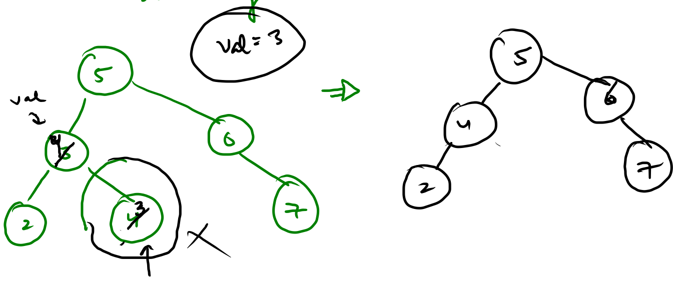


⇒ Attaching removing node child to the parent of removing node.

⇒ Find the given remain node

Case-03

When removing node has 2 children

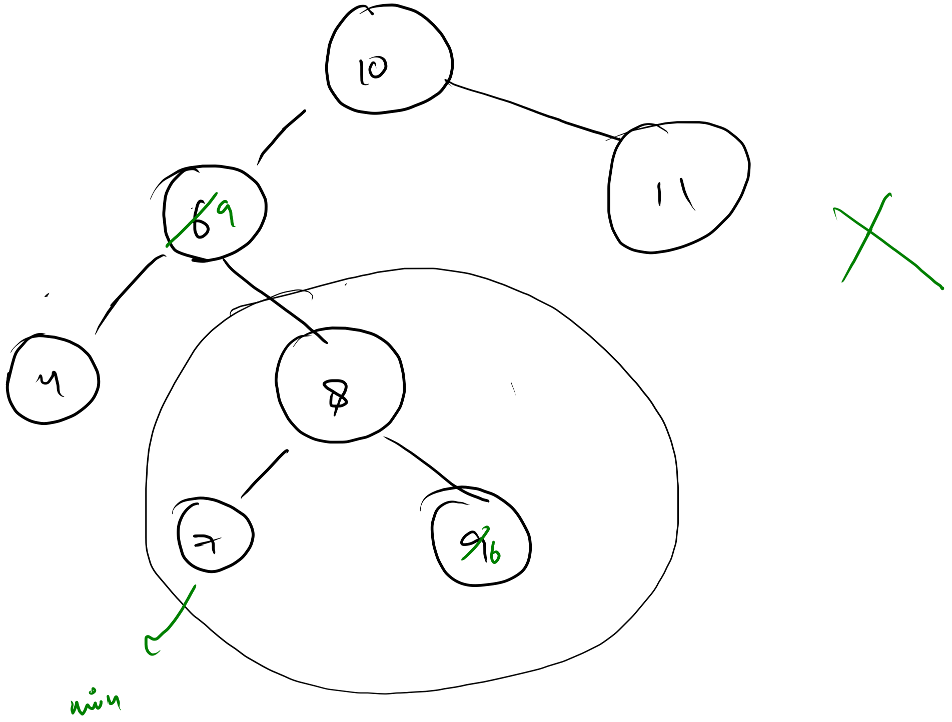


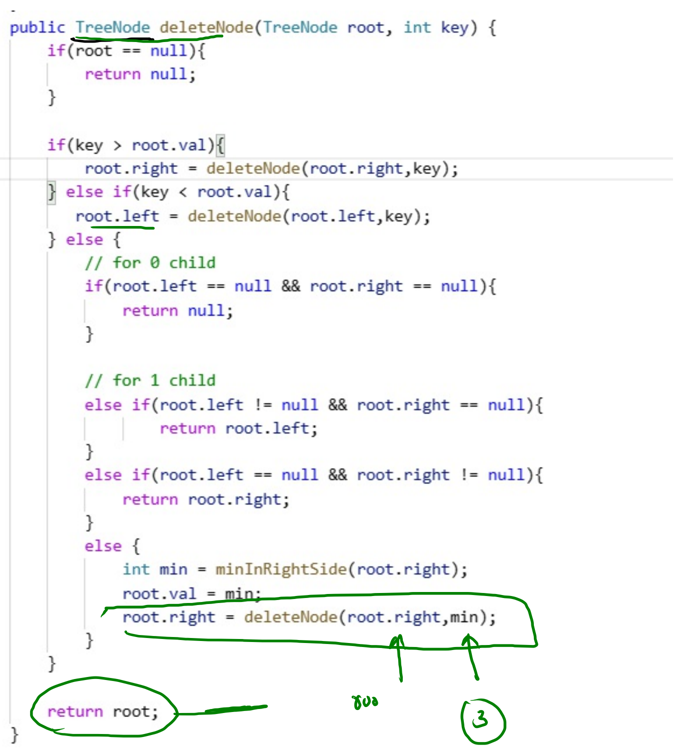
Step-01 → Find the min^n in the right side of removing node. (minNode)

Step-02 Swap the data of removing node and minNode.

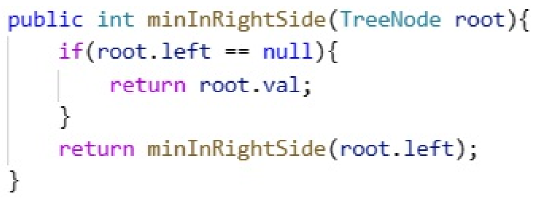
Step-03 → Apply the same logic as of for 0 child & 1 child.

Val = 8



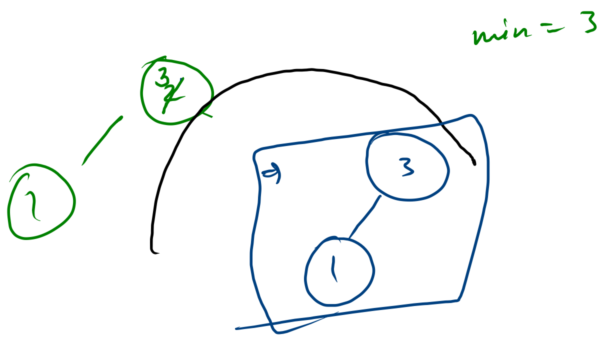


```
public TreeNode deleteNode(TreeNode root, int key) {
    if(root == null){
        return null;
    }

    if(key > root.val){
        root.right = deleteNode(root.right,key);
    } else if(key < root.val){
       root.left = deleteNode(root.left,key);
    } else {
        // for 0 child
        if(root.left == null && root.right == null){
            return null;
        }

        // for 1 child
        else if(root.left != null && root.right == null){
                return root.left;
        }
        else if(root.left == null && root.right != null){
            return root.right;
        }
        else {
            int min = minInRightSide(root.right);
            root.val = min;
            root.right = deleteNode(root.right,min);
        }
    }

    return root;
}
```



```
public int minInRightSide(TreeNode root){
    if(root.left == null){
        return root.val;
    }
    return minInRightSide(root.left);
}
```

8 - 10 mins   Code + Dry Run



Range Sum of BST 1 (Day 46)

Problem | Submissions | Leaderboard | Discussions

You are given preorder of the Binary search tree construct BST, Now you have 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].

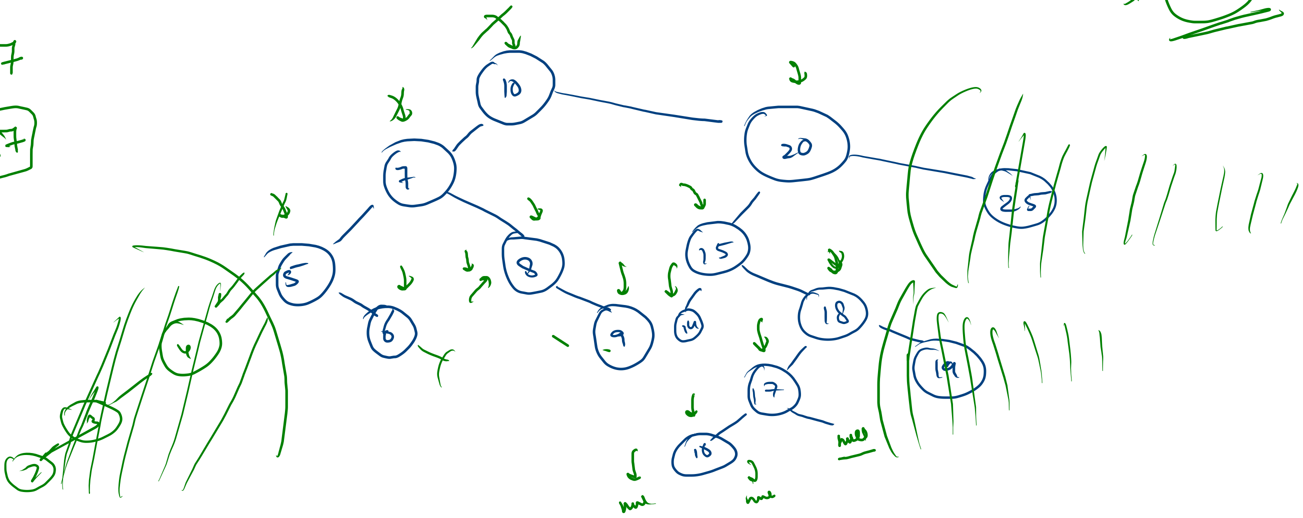
$\text{sum} = \underline{10 + 7 + 8 + 9 + 15 + 14 + 17}$

$\underline{+16}$

$\rightarrow \boxed{\text{ans}}$

low = 7

$\boxed{\text{high} = 17}$



```java
public static int rangeSumOfBST(Node root, int low, int high){
    if(root == null) {
        return 0;
    }
    int sum = 0;
    if(root.data >=low && root.data<=high){
        sum += root.data;
    }

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

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

    return sum;
}
```

⇒ ① Find the given removing node in BST

⇒ ② Check no. of children of removing node.

⇒ ③ → 0 → step-1

1 → step-2

2 → step-3