LEC-40
07/06/25 10:11 AM

BST

Binary Tree / Binary Search Tree

1. Size

Class Node {
int val
Node left
Node right
}

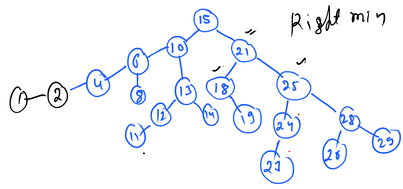
1. BST → BT ✓
2. BT → BST ✗

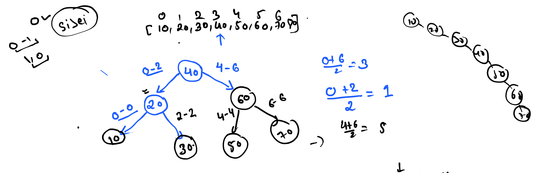
BT → in-order → Sorted → BST

Left Root Right

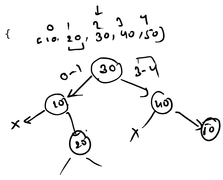
1 2 4 6 8 10 11 12 13 14 15 18 19 21 23 24 25 26 28 29

1)



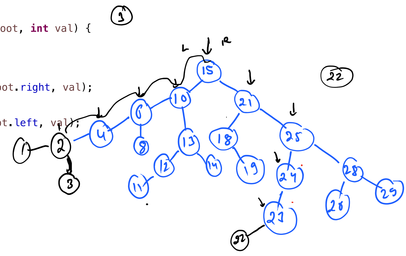



```
private Node Create_Tree(int[] in, int si, int ei) {
        // TODO Auto-generated method stub
        if (si > ei) {
            return null;
        }
        int mid = (si + ei) / 2;
        Node nn = new Node();
        nn.val = in[mid];
        nn.left = Create_Tree(in, si, mid - 1);
        nn.right = Create_Tree(in, mid + 1, ei);
        return nn;
    }
```



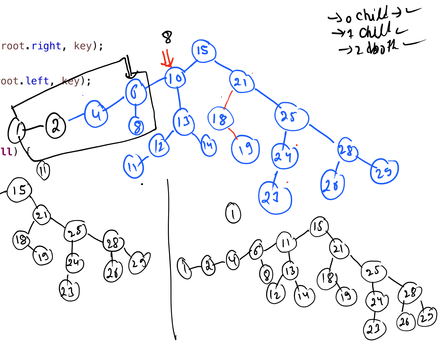

```
public TreeNode insertIntoBST(TreeNode root, int val) {
        if(root==null) {
            return new TreeNode(val);
        }
        if(root.val<val) {
            root.right=insertIntoBST(root.right, val);
        }
        else {
            root.left=insertIntoBST(root.left, val);
       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

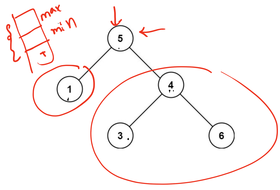
3)



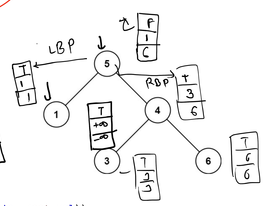

```
public TreeNode deleteNode(TreeNode root, int key) {
        if(root==null) {
            return null;
        }
        if(root.val<key) {
            root.right=deleteNode(root.right, key);
        }
        else if(root.val>key) {
            root.left=deleteNode(root.left, key);
        }
  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

        }
    }
```

→ 0 child
→ 1 child
→ 2 child

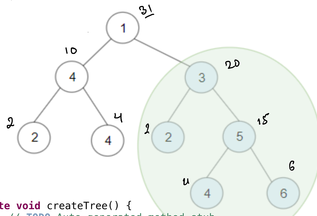




Class BSTPair {
boolean isBST = T
long min = Long.MAX
long max = Long.MIN
}




```
public BstPair ValidBST(TreeNode root) {
        if (root == null) {
            return new BstPair();
        }

        BstPair lbp = ValidBST(root.left);
        BstPair rbp = ValidBST(root.right);
        BstPair sbp = new BstPair();
        sbp.min = Math.min(lbp.min, Math.min(rbp.min, root.val));
        sbp.max = Math.max(lbp.max, Math.max(rbp.max, root.val));
        sbp.isbst = lbp.isbst && rbp.isbst && lbp.max < root.val && rbp.min > root.val;
        return sbp;

    }
```



ans = (ans ans sub / 2)

ans = (ans, ans)

```
private void createTree() {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        Queue<Node> q = new LinkedList<>();
        Node nn = new Node();
        nn.val = sc.nextInt();
        root = nn;
        q.add(nn);
    }
```

[10 5 7 8 -1 9 -1 -1 -1 16 -1 -1 -1]