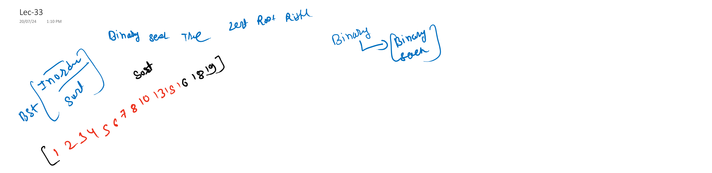
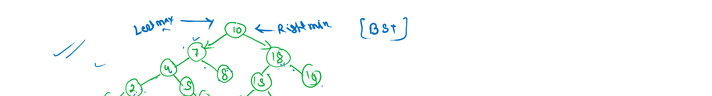
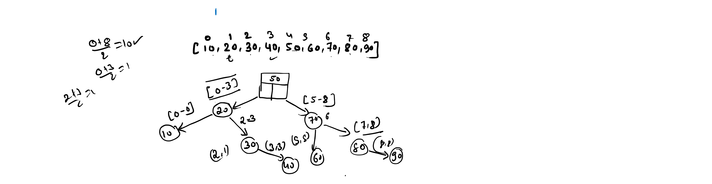
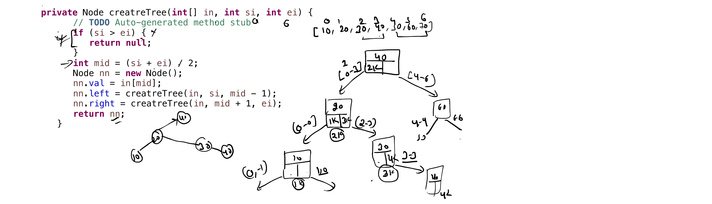
Lec-33

30/01/24 1:10 PM





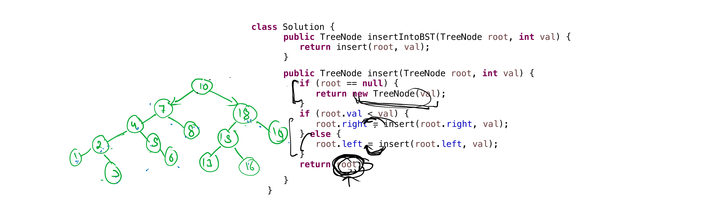




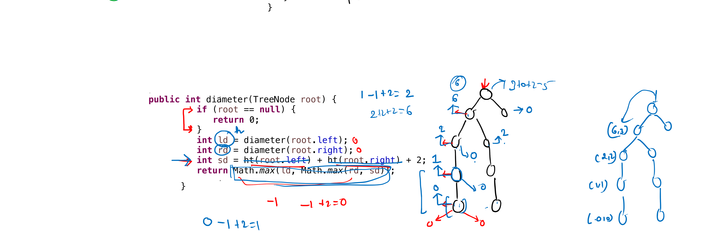
```
private Node creatreTree(int[] in, int si, int ei) {
    // TODO Auto-generated method stub
    if (si > ei) {
        return null;
    }
    int mid = (si + ei) / 2;
    Node nn = new Node();
    nn.val = in[mid];
    nn.left = creatreTree(in, si, mid - 1);
    nn.right = creatreTree(in, mid + 1,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



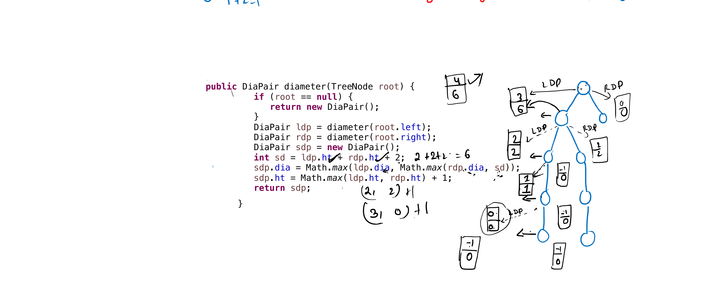
```
class Solution {
    public TreeNode insertIntoBST(TreeNode root, int val) {
        return insert(root, val);
    }

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



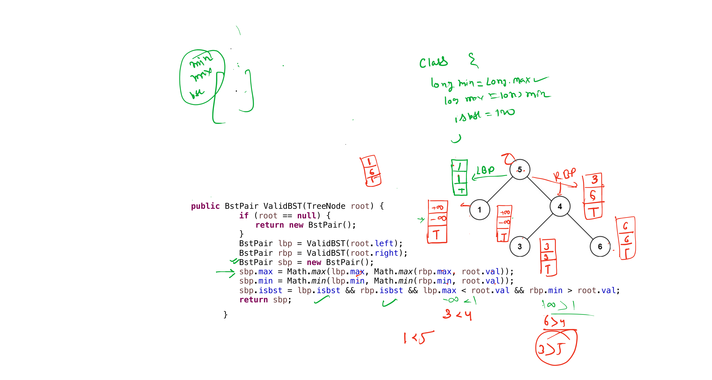
```
public int diameter(TreeNode root) {
    if (root == null) {
        return 0;
    }
    int ld = diameter(root.left);
    int rd = diameter(root.right);
    int sd = ht(root.left) + ht(root.right) + 2;
    return Math.max(ld, Math.max(rd, sd));
}
```



```
public DiaPair diameter(TreeNode root) {
    if (root == null) {
        return new DiaPair();
    }
    DiaPair ldp = diameter(root.left);
    DiaPair rdp = diameter(root.right);
    DiaPair sdp = new DiaPair();
    int sd = ldp.ht + rdp.ht + 2;
    sdp.dia = Math.max(ldp.dia, Math.max(rdp.dia, sd));
    sdp.ht = Math.max(ldp.ht, rdp.ht) + 1;
    return sdp;
}
```



```
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.max = Math.max(lbp.max, Math.max(rbp.max, root.val));
    sbp.min = Math.min(lbp.min, Math.min(rbp.min, root.val));
    sbp.isbst = lbp.isbst && rbp.isbst && lbp.max < root.val && rbp.min > root.val;
    return sbp;
}
```



```
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;
        } else {
            int max = max(root.left);
            root.left = deleteNode(root.left, max);
            root.val = max;
        }
    }
    return root;
}
```