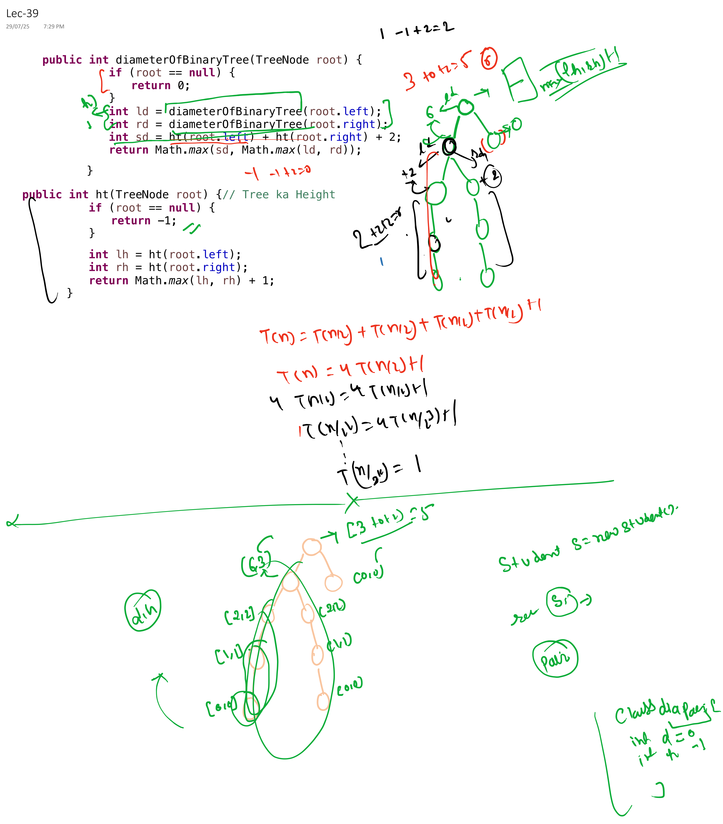
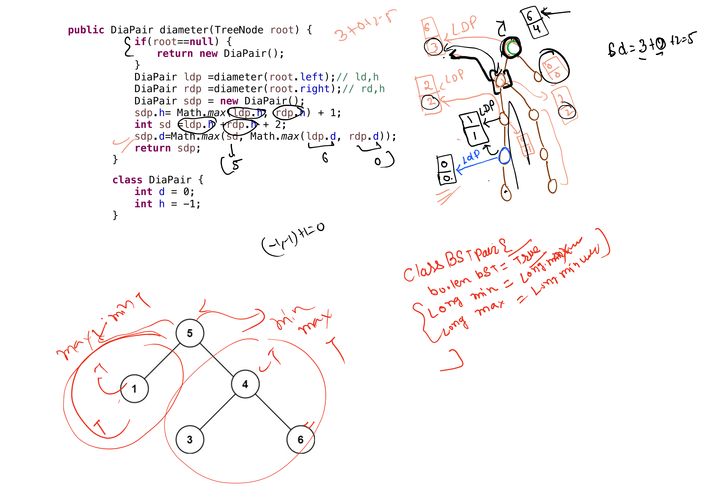
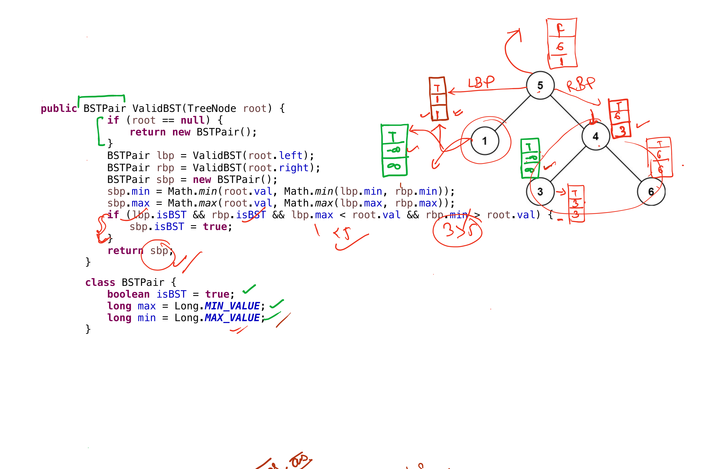
```java
public int diameterOfBinaryTree(TreeNode root) {
        if (root == null) {
            return 0;
        }
        int ld = diameterOfBinaryTree(root.left);
        int rd = diameterOfBinaryTree(root.right);
        int sd = ht(root.left) + ht(root.right) + 2;
        return Math.max(sd, Math.max(ld,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

    }

public int ht(TreeNode root) {// Tree ka Height
        if (root == null) {
            return -1;
        }

        int lh = ht(root.left);
        int rh = ht(root.right);
        return Math.max(lh, rh) + 1;
    }
```

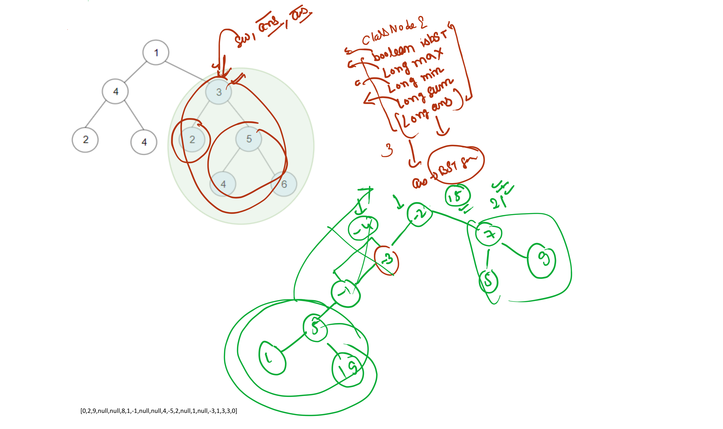


```java
public DiaPair diameter(TreeNode root) {
        if(root==null) {
            return new DiaPair();
        }
        DiaPair ldp =diameter(root.left);// ld,h
        DiaPair rdp =diameter(root.right);// rd,h
        DiaPair sdp = new DiaPair();
        sdp.h= Math.max(ldp.h, rdp.h) + 1;
        int sd = ldp.h + rdp.h + 2;
        sdp.d=Math.max(sd, Math.max(ldp.d, rdp.d));
        return sdp;
    }

    class DiaPair {
        int d = 0;
        int h = -1;
    }
```



```java
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(root.val, Math.min(lbp.min, rbp.min));
        sbp.max = Math.max(root.val, Math.max(lbp.max, rbp.max));
        if (lbp.isBST && rbp.isBST && lbp.max < root.val && rbp.min > root.val) {
            sbp.isBST = true;
        }
        return sbp;
    }

    class BSTPair {
        boolean isBST = true;
        long max = Long.MIN_VALUE;
        long min = Long.MAX_VALUE;
    }
```





[0,2,9,null,null,8,1,-1,null,null,4,-5,2,null,1,null,-3,1,3,3,0]