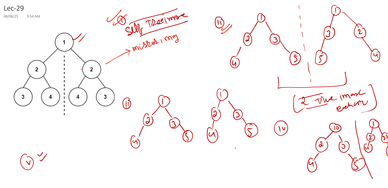
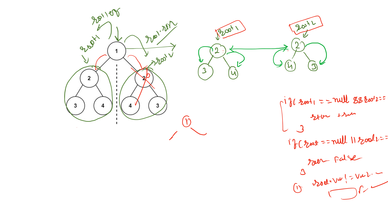
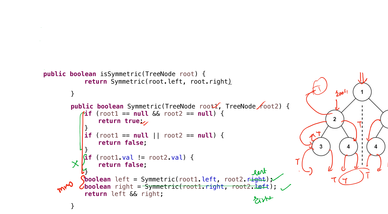
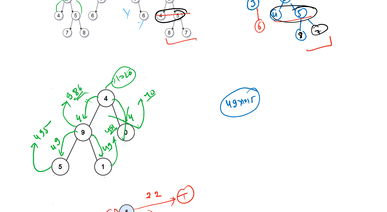
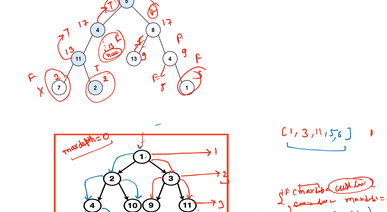


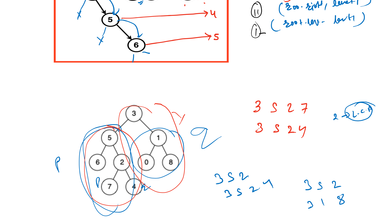


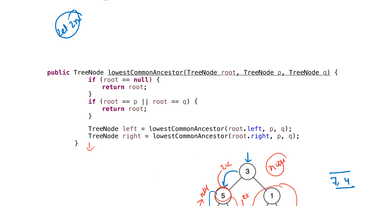


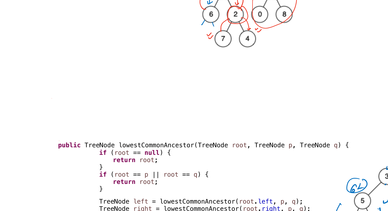
```
public boolean isSymmetric(TreeNode root) {
        return Symmetric(root.left, root.right)
    }

    public boolean Symetric(TreeNode root1, TreeNode root2) {
        if (root1 == null && root2 == null) {
            return true;
        }
        if (root1 == null || root2 == null) {
            return false;
        }
        if (root1.val != root2.val) {
            return false;
        }
        boolean left = Symetric(root1.left, root2.right);
        boolean right = Symetric(root1.right, root2.left);
        return left && right;
    }
```

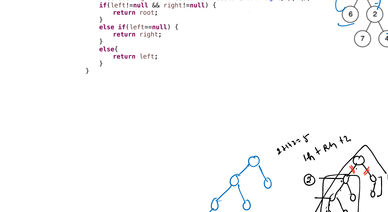




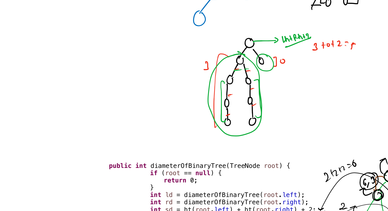




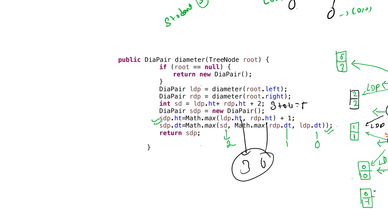
```
public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if (root == null) {
            return root;
        }
        if (root == p || root == q) {
            return root;
        }

        TreeNode left = lowestCommonAncestor(root.left, p, q);
        TreeNode right = lowestCommonAncestor(root.right, p, q);
    }
```



```
public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if (root == null) {
            return root;
        }
        if (root == p || root == q) {
            return root;
        }

        TreeNode left = lowestCommonAncestor(root.left, p, q);
        TreeNode right = lowestCommonAncestor(root.right, p, q);
        if(left!=null && right!=null) {
            return root;
        }
        else if(left==null) {
            return right;
        }
        else{
            return left;
        }
    }
```



```
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(rd, ld));
    }

    public int ht(TreeNode root) {
        if (root == null) {
            return -1;
        }
        int lh = ht(root.left);
        int rh = ht(root.right);
        return Math.max(lh, rh) + 1;
    }
```



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