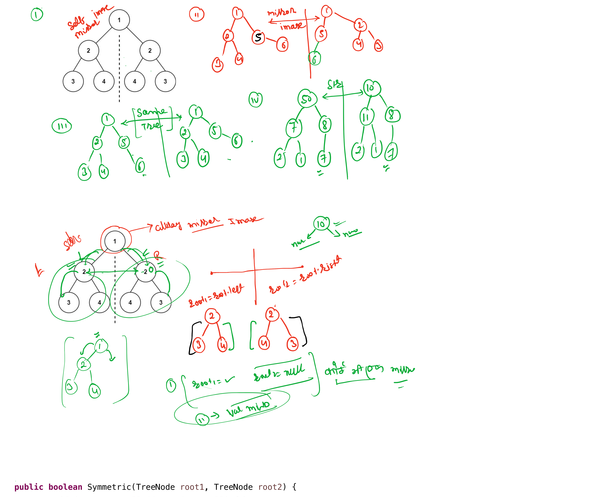
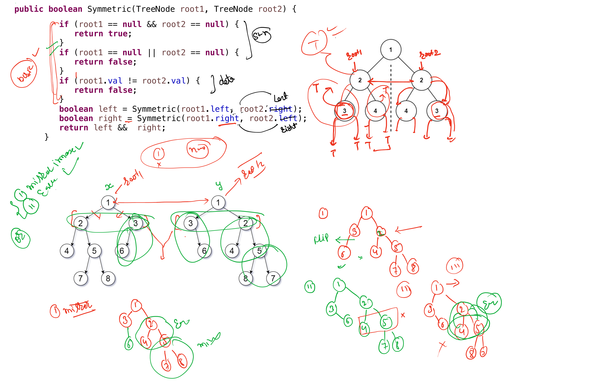
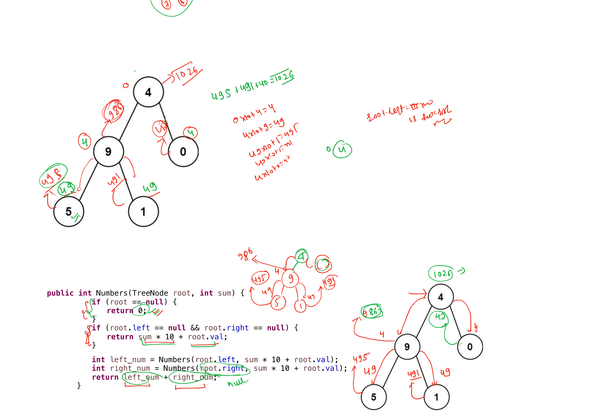
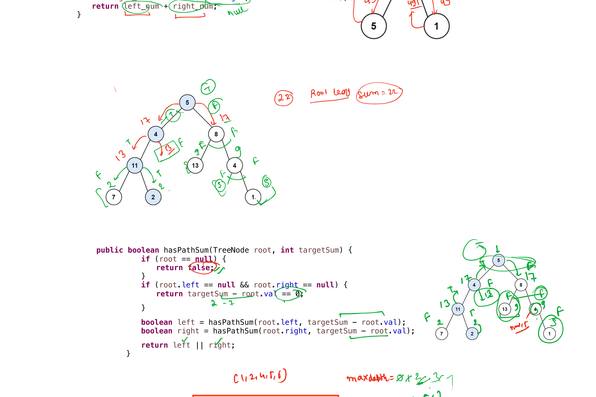


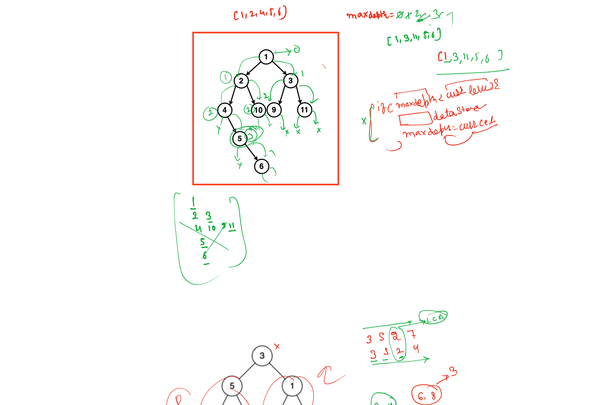

```
public boolean Symmetric(TreeNode root1, TreeNode root2) {
    if (root1 == null && root2 == null) {
        return true;
    }
    if (root1 == null || root2 == null) {
        return false;
    }
    if (root1.val != root2.val) {
        return false;
    }
    boolean left = Symmetric(root1.left,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





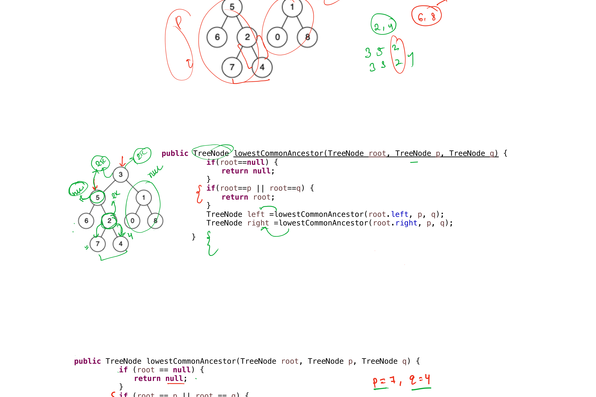


```
public int Numbers(TreeNode root, int sum) {
    if (root == null) {
        return 0;
    }
    if (root.left == null && root.right == null) {
        return sum * 10 + root.val;
    }
    int left_num = Numbers(root.left, sum * 10 + root.val);
    int right_num = Numbers(root.right, sum * 10 + root.val);
    return left_num + right_num;
}
```



```
public boolean hasPathSum(TreeNode root, int targetSum) {
        if (root == null) {
            return false;
        }
        if (root.left == null && root.right == null) {
            return targetSum - root.val == 0;
        }
        boolean left = hasPathSum(root.left, targetSum - root.val);
        boolean right = hasPathSum(root.right, targetSum - root.val);

        return left || right;
    }
```




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