437. Path Sum III
We will use pre sum and traverse the tree in pre-order.TC is O(n)

```
from collections import defaultdict
class Solution:
    def pathSum(self, root: TreeNode, sum: int) -> int:
        memo = defaultdict(int)
        memo[0] = 1
        self.result = 0
        def traverse(node, cur):
            if not node:
                return
            cur = cur + node.val
            self.result += memo[cur - sum]
            memo[cur] += 1
            traverse(node.left, cur)
            traverse(node.right, cur)
            memo[cur] -= 1

        traverse(root, 0)
        return self.result
```

124. Binary Tree Maximum Path Sum
We will traverse our tree in post order. We will return the largest path and store the maximum path sum. TC is O(n)

```
class Solution:
    def maxPathSum(self, root: TreeNode) -> int:
        self.max_sum = -float('inf')

        def traverse(node):
            if not node:
                return 0
            left = max(0, traverse(node.left))
            right = max(0, traverse(node.right))
            self.max_sum = max(self.max_sum, node.val + left + right)

            return max(left, right) + node.val

        traverse(root)
        return self.max_sum
```

543. Diameter of Binary Tree
We will traverse our tree in post order and return the longest path, recording the largest path. TC is O(n)

```
class Solution:
    def diameterOfBinaryTree(self, root: TreeNode) -> int:
        self.max_node = 0
        if not root:
            return 0
        def traverse(node):
            if not node:
                return 0
            left = traverse(node.left)
            right = traverse(node.right)
            self.max_node = max(self.max_node, left + right + 1)
            return max(left, right) + 1
        traverse(root)
        return self.max_node - 1
```

687. Longest Univalue Path

It's the same as the previous one except we will set left or right to zero if values are not equal. TC is O(n)

```
class Solution:
    def longestUnivaluePath(self, root: TreeNode) -> int:
        self.max_node = 0
        if not root:
            return 0
        def traverse(node):
            if not node:
                return (0, 0)
            left_val, left_num = traverse(node.left)
            right_val, right_num = traverse(node.right)
            if node.val != left_val:
                left_num = 0
            if node.val != right_val:
                right_num = 0
            self.max_node = max(self.max_node, left_num + right_num + 1)
            return (node.val, 1 + max(left_num, right_num))
        traverse(root)
        return self.max_node - 1
```

129. Sum Root to Leaf Numbers

We will traverse the whole tree in preorder and always accumulate the number, we will add it to result when we encounter the leaf node. TC is O(n)

```
class Solution:
    def sumNumbers(self, root: TreeNode) -> int:
        self.result = 0
```

```
def helper(cur, node):
    if not node:
        return
    cur = cur * 10 + node.val
    if not node.left and not node.right:
        self.result += cur
    helper(cur, node.left)
    helper(cur, node.right)

helper(0, root)
return self.result
```