108. Convert Sorted Array to Binary Search Tree

Convert Sorted Array to Binary Search Tree

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> TreeNode:
        if len(nums)==0:
            return None

        middle=len(nums)//2


        return TreeNode(
            val=nums[middle],
            left=self.sortedArrayToBST(nums[:middle]),
            right=self.sortedArrayToBST(nums[middle+1:]))
```

404. Sum of Left Leaves

Sum of Left Leaves

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def dfs(self, root, side):
        if not root:
            return

        if not root.left and not root.right:
            if side == -1:
                self.sum += root.val

        self.dfs(root.left, -1)
        self.dfs(root.right, 1)

    def sumOfLeftLeaves(self, root):
        self.sum = 0
        self.dfs(root, 0)
        return self.sum
```

538. Convert BST to Greater Tree

Convert BST to Greater Tree

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def convertBST(self, root: TreeNode) -> TreeNode:
        self.sum=0

        def dfs(root):
            if not root:
                return

            dfs(root.right)
            root.val=self.sum=self.sum+root.val
            dfs(root.left)

        dfs(root)
        return root
```

637. Average of Levels in Binary Tree

Average of Levels in Binary Tree

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def averageOfLevels(self, root: TreeNode) -> List[float]:
        if root is None:
            return

        queue=[root]
        result=[]

        while queue:
            result.append(sum([node.val for node in queue])/len(queue))

            new_queue=[]

            for node in queue:
                if node.left:
                    new_queue.append(node.left)

                if node.right:
                    new_queue.append(node.right)

            queue=new_queue
        return result
```

669. Trim a Binary Search Tree

Trim a Binary Search Tree

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def trimBST(self, root: TreeNode, L: int, R: int) -> TreeNode:
        if not root:
            return None

        if L<=root.val and R>=root.val:
            root.left=self.trimBST(root.left,L,R)
            root.right=self.trimBST(root.right,L,R)
            return root

        else:
            if root.val<R:
                return self.trimBST(root.right,L,R)

            if root.val>L:
                return self.trimBST(root.left,L,R)
```