概念必须弄清楚，什么是左叶子（二叉树的某个属性跟根节点有关）

不管某个属性是否与父节点有关其实我们都可以遍历所有节点然后根据某个条件来计算属性

404.左叶子之和

计算给定二叉树的所有左叶子之和。

示例：

```
    3
   / \
  9  20
    /  \
   15   7
```

在这个二叉树中，有两个左叶子，分别是 9 和 15，所以返回 24

> 如果某节点的左节点不为空，且该左节点是叶子节点则称该左节点为左叶子
> **因此该节点是否是左叶子节点第一是叶子节点，且必须要通过节点的父节点来判断其节点是否是父节点的左节点**

判断左叶子节点的代码为:

如果该节点的左节点不为空，该节点的左节点的左节点为空，该节点的左节点的右节点为空，则找到了一个左叶子

```
if(node->left!=NULL&&node->left->left==NULL&&node->left->right==NULL){
 //左叶子节点的处理逻辑

}
```

迭代法(用前序遍历算法)

```
class Solution {
public:
    int sumOfLeftLeaves(TreeNode* root) {
 stack<TreeNode*> sta;
 int sum=0;
 if(root==NULL) return sum;
 sta.push(root);
```

```
while(!sta.empty()){
TreeNode* node=sta.top();
 sta.pop();
 if(node->left!=NULL&&node->left->left==NULL&&node->left->right==NULL){
 sum+=node->left->val;
 }
 if(node->right)sta.push(node->right);
 if(node->left) sta.push(node->left);
 }
 return sum;
    }
};
```

513.找树左下角的值

给定一个二叉树，在树的最后一行找到最左边的值。

示例 1:

输入:

```
    2
   / \
  1   3
```

输出:
1

在层次遍历时,节点拓展时每一层while循环中
第一个被拓展的节点就是每层的最左节点
最后一个被拓展的节点就是每层最右节点.

```
class Solution {
public:
    int findBottomLeftValue(TreeNode* root) {
 queue<TreeNode*> que;
 int result=0;
 if(root==NULL) return result;
 while(!que.empty()){
 int size=que.size();
```

```
9  for(int i=0;i<size;i++){
10  TreeNode* node=que.front();
11  que.pop();
12  if(i==0) result=node->val;
13  if(node->left) que.push(node->left);
14  if(node->right) que.push(node->right);
15  }
16  }
17  return result;
18      }
19 };
```