# 题目

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

# 分析

## 方法一：递归法

**思路：**

首先此题是一道递归题，如果是求所有节点之和的话，那么我们递归就是先求左孩子为根节点的树的和，再就右孩子为根节点的树的和，然后再将左右之和加上本身节点的值就求出了一个树的节点之和，那么怎么求左右节点树的和呢，此时就要用到递归了，我们把左右孩子又作为根节点，然后再用同样的方法来求他们的左右孩子，最后加上就完了。但是 如果不加限制我们就会出现一直递归的情况，所以我们需要加上限制条件，可以加上当传入的节点返回0即可。

class Solution {

public:

int sumOfLeftLeaves(TreeNode\* root) {

if(root == NULL) return 0;

return sumOfLeftLeaves(root->left)+

sumOfLeftLeaves(root->right)+root->val;

}

};

此题加上一些限制条件，我们只需要稍加改变即可，首先就是我们只求叶子节点之和（左孩子右孩子为空就可以判断是否为叶子节点），然后需要判断是否是左边的叶子节点，所以我们只需要判断每个节点左边节点是否为空，如果不为空是否为叶子节点，如果成立我们就返回这个左孩子的值再加上向右边孩子进行递归计算的值（右孩子中也有可能有左叶子节点）。

class Solution {

public:

int sumOfLeftLeaves(TreeNode\* root) {

if(root == NULL) return 0;

if(root->left!=NULL&&root->left->right==NULL

&&root->left->left==NULL)

return sumOfLeftLeaves(root->right)+root->left->val;

return sumOfLeftLeaves(root->left)+sumOfLeftLeaves(root->right);

}

};

**代码：**

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class Solution {

public:

int sumOfLeftLeaves(TreeNode\* root) {

if(nullptr==root)

return 0;

//终止条件

if(nullptr!=root->left &&

nullptr==root->left->left && nullptr==root->left->right)

return sumOfLeftLeaves(root->right)+root->left->val;

//递归调用左右子树

return sumOfLeftLeaves(root->left)+sumOfLeftLeaves(root->right);

}

};

**另一种写法：**

class Solution {

public:

bool isLeafNode(TreeNode\* node) {

return !node->left && !node->right;

}

int dfs(TreeNode\* node) {

int ans = 0;

if (node->left) {

ans += isLeafNode(node->left) ? node->left->val : dfs(node->left);

}

if (node->right && !isLeafNode(node->right)) {

ans += dfs(node->right);

}

return ans;

}

int sumOfLeftLeaves(TreeNode\* root) {

return root ? dfs(root) : 0;

}

};

**另一种写法：**

/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     TreeNode \*left;

 \*     TreeNode \*right;

 \*     TreeNode() : val(0), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

 \* };

 \*/

class Solution {

public:

    int sumOfLeftLeaves(TreeNode\* root) {

        if(nullptr==root)

        {

            return 0;

        }

        else

        {

            return dfs(root);

        }

    }

    bool isLeafNode(TreeNode \*node)

    {

        if(!node->left && !node->right)

            return true;

        return false;

    }

    int dfs(TreeNode \*node)

    {

        int ret = 0;

        if(node->left)

        {

            if(isLeafNode(node->left))

            {

                ret += node->left->val;

            }

            else

            {

                ret += dfs(node->left);

            }

        }

        if(node->right)

        {

            if(isLeafNode(node->right))

            {

                ret += 0;//右叶子节点不计算

            }

            else

            {

                ret += dfs(node->right);

            }

        }

        return ret;

    }

};

说明：如果需要计算所有叶子节点的值，则代码如下：

/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     TreeNode \*left;

 \*     TreeNode \*right;

 \*     TreeNode() : val(0), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

 \* };

 \*/

class Solution {

public:

    int sumOfLeftLeaves(TreeNode\* root) {

        if(nullptr==root)

        {

            return 0;

        }

        else

        {

            return dfs(root);

        }

    }

    bool isLeafNode(TreeNode \*node)

    {

        if(!node->left && !node->right)

            return true;

        return false;

    }

    int dfs(TreeNode \*node)

    {

        int ret = 0;

        if(node->left)

        {

            if(isLeafNode(node->left))

            {

                ret += node->left->val;

            }

            else

            {

                ret += dfs(node->left);

            }

        }

        if(node->right)

        {

            if(isLeafNode(node->right))

            {

                ret += node->right->val;

            }

            else

            {

                ret += dfs(node->right);

            }

        }

        return ret;

    }

};