# 题目

给定一个二叉树，它的每个结点都存放着一个整数值。

找出路径和等于给定数值的路径总数。

路径不需要从根节点开始，也不需要在叶子节点结束，但是路径方向必须是向下的（只能从父节点到子节点）。

二叉树不超过1000个节点，且节点数值范围是[-1000000,1000000]的整数。

**示例：**

root = [10,5,-3,3,2,null,11,3,-2,null,1], sum = 8

10

/ \

5 -3

/ \ \

3 2 11

/ \ \

3 -2 1

返回3。和等于8的路径有:

1. 5 -> 3

2. 5 -> 2 -> 1

3. -3 -> 11

# 分析

## 方法一：递归法

代码：

class Solution {

public:

    int count = 0;

    int pathSum(TreeNode \*root, int targetSum)

    {

        if (!root)

            return 0;

        dfs(root,targetSum);            //以root为起始点查找路径

        pathSum(root->left, targetSum);  //左子树递归

        pathSum(root->right, targetSum); //右子树递归

        return count;

    }

    void dfs(TreeNode \*root, int sum)

    {

        if (!root)

            return;

        sum -= root->val;

        if (sum == 0) //注意不要return,因为不要求到叶节点结束,所以一条路径下面还可能有另一条

            count++;  //如果找到了一个路径全局变量就+1

        dfs(root->left, sum);

        dfs(root->right, sum);

    }

};

**另一种写法：**

/\*\*

\* 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 pathSum(TreeNode\* root, int sum) {

int res = 0;

if(!root) return res;

if(root->val == sum) res++;

//和root节点无关的

res += pathSum(root->left,sum);

res += pathSum(root->right,sum);

//和root有关的，这里不能直接调用pathSum,

//否则递归以后造成“断开的路径”

res += pathSum\_nroot(root->left, sum - root->val);

res += pathSum\_nroot(root->right, sum - root->val);

return res;

}

int pathSum\_nroot(TreeNode\* root, int sum) {

//这个函数只找包含root的路径

if(!root) return 0;

int res = 0;

if(root->val == sum) res++;

res += pathSum\_nroot(root->left,sum - root->val); //自递归

res += pathSum\_nroot(root->right,sum - root->val);

return res;

}

};

或：

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

public int pathSum(TreeNode root, int sum) {

if(root == null){

return 0;

}

int result = countPath(root,sum);

int a = pathSum(root.left,sum);

int b = pathSum(root.right,sum);

return result+a+b;

}

public int countPath(TreeNode root,int sum){

if(root == null){

return 0;

}

sum = sum - root.val;

int result = sum == 0 ? 1:0;

return result + countPath(root.left,sum) + countPath(root.right,sum);

}

}

## 方法二：迭代法