# 题目

给定一个二叉树，它的每个结点都存放一个 0-9 的数字，每条从根到叶子节点的路径都代表一个数字。

例如，从根到叶子节点路径 1->2->3 代表数字 123。

计算从根到叶子节点生成的所有数字之和。

说明: 叶子节点是指没有子节点的节点。

**示例 1:**

输入: [1,2,3]

1

/ \

2 3

输出: 25

解释:

从根到叶子节点路径 1->2 代表数字 12.

从根到叶子节点路径 1->3 代表数字 13.

因此，数字总和 = 12 + 13 = 25.

**示例 2:**

输入: [4,9,0,5,1]

4

/ \

9 0

 / \

5 1

输出: 1026

解释:

从根到叶子节点路径 4->9->5 代表数字 495.

从根到叶子节点路径 4->9->1 代表数字 491.

从根到叶子节点路径 4->0 代表数字 40.

因此，数字总和 = 495 + 491 + 40 = 1026.

# 分析

## 方法一：递归法

**思路：**

递归函数设计：

void dfs(TreeNode\* root, int number)

root是当前遍历的节点，number是从根节点到当前节点路径表示的数。

递归边界：

遍历到叶节点时，将根节点到当前叶节点路径维护的数字加入答案中。

递归过程如下：

1、从根节点开始往下遍历，维护一个从根节点到当前节点路径表示的数number，初始值为0。

2、遍历当前节点root，将root->val追加到number中，即执行number = number\*10+root->val操作。

3、如果左子树不为空，递归到左子树。如果右子树不为空，递归到右子树。

4、当遍历到叶节点时，将number加入到数字之和res中。

5、最后返回res。

时间复杂度分析：O(n)，其中n是二叉树的节点个数，对每个节点访问一次。

代码：

class Solution {

public:

int res = 0;

int sumNumbers(TreeNode\* root) {

dfs(root, 0);

return res;

}

void dfs(TreeNode\* root, int number)

{

number = number \* 10 + root->val;

if(!root->left && !root->right) res += number;

//遍历到叶节点，将number加入res中

if(root->left) dfs(root->left,number); //递归左子树

if(root->right) dfs(root->right,number); //递归右子树

}

};

另一种写法：

class Solution {

public:

int sumNumbers(TreeNode\* root) {

return helper(root,0);

}

int helper(TreeNode\* root, int i){

if (root == nullptr) return 0;

int temp = i \* 10 + root->val;

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

return temp;

return helper(root->left, temp) + helper(root->right, temp);

}

};