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，该题为所有叶子节点路径的和，可以使用深度遍历来解决，若是有子节点，将本节点的值，传递给子节点，子节点相加。若没有子节点，将本节点的值和SUM相加进行总和统计。当传入的节点就为空时，返回0。

```
void dfs(TreeNode* root,int num,int &sum){
    if(root->right==NULL&&root->left==NULL){
        sum=sum+num*10+root->val;
        return;
    }
    if(root->right!=NULL){
        dfs(root->right,num*10+root->val,sum);
    }
    if(root->left!=NULL){
        dfs(root->left,num*10+root->val,sum);
```

```
    }
}
class Solution {
public:
    int sumNumbers(TreeNode* root) {
        if(root==NULL){
            return 0;
        }
        int sum=0;
        dfs(root,0,sum);
        return sum;
    }
};
```