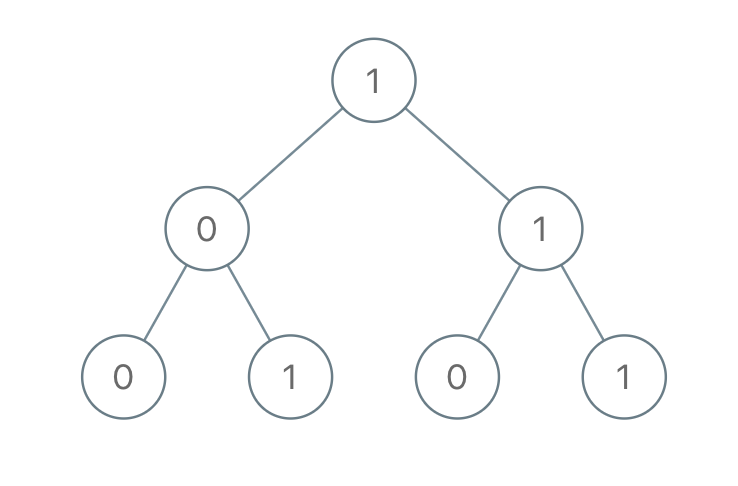
# 题目

给出一棵二叉树，其上每个结点的值都是 0 或 1 。每一条从根到叶的路径都代表一个从最高有效位开始的二进制数。例如，如果路径为 0 -> 1 -> 1 -> 0 -> 1，那么它表示二进制数 01101，也就是 13 。

对树上的每一片叶子，我们都要找出从根到该叶子的路径所表示的数字。

以 10^9 + 7 为模，返回这些数字之和。

**示例：**



输入：[1,0,1,0,1,0,1]

输出：22

解释：(100) + (101) + (110) + (111) = 4 + 5 + 6 + 7 = 22

**提示：**

树中的结点数介于 1 和 1000 之间。

node.val 为 0 或 1 。

# 分析

## 方法一：递归法/深度优先遍历

class Solution {

public:

const long M = 1e9 + 7;

void dfs(TreeNode\* root, long p, long& res) {

if (root == NULL) return;

p = (p << 1) | root->val;

if (root->left == NULL && root->right == NULL) {

res += p;

res %= M;

return;

}

dfs(root->left, p, res);

dfs(root->right, p, res);

}

int sumRootToLeaf(TreeNode\* root) {

long res = 0;

long p = 0;

dfs(root, p, res);

return res;

}

};