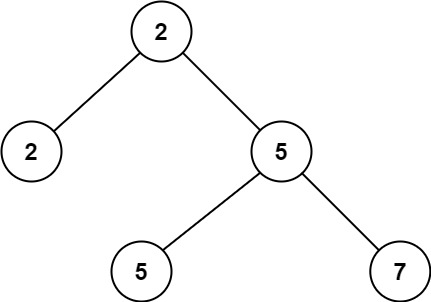
# 题目

给定一个非空特殊的二叉树，每个节点都是正数，并且每个节点的子节点数量只能为2或0。如果一个节点有两个子节点的话，那么该节点的值等于两个子节点中较小的一个。

更正式地说，root.val = min(root.left.val, root.right.val)总成立。

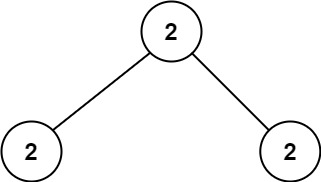
给出这样的一个二叉树，你需要输出所有节点中的第二小的值。如果第二小的值不存在的话，输出-1。

示例 1：



输入：root = [2,2,5,null,null,5,7]输出：5解释：最小的值是 2 ，第二小的值是 5 。

示例 2：



输入：root = [2,2,2]

输出：-1

解释：最小的值是 2, 但是不存在第二小的值。

提示：

树中节点数目在范围 [1, 25]内

1 <= Node.val <= 231 - 1

对于树中每个节点 root.val == min(root.left.val, root.right.val)

# 分析

/\*\*

\* 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 findSecondMinimumValue(TreeNode\* root) {

minnum = root->val; //记录最小值

dfs(root); //找第二小的值

return ans;

}

private:

int minnum;

int ans = -1; //答案，如果找不到就输出-1

void dfs(TreeNode\* root){

if(root->val > minnum && (root->val < ans || ans == -1)){

//查看这个节点是否是第二小的

ans = root->val;

return;

}

if(root->val >= ans && ans != -1)

//如果最小的都比当前答案大，则不用查下去了

return;

if(root->left) //查这个节点的左节点和右节点

dfs(root->left);

if(root->right)

dfs(root->right);

}

};