# 题目

给你一个有根节点的二叉树，找到它最深的叶节点的最近公共祖先。

回想一下：

叶节点是二叉树中没有子节点的节点

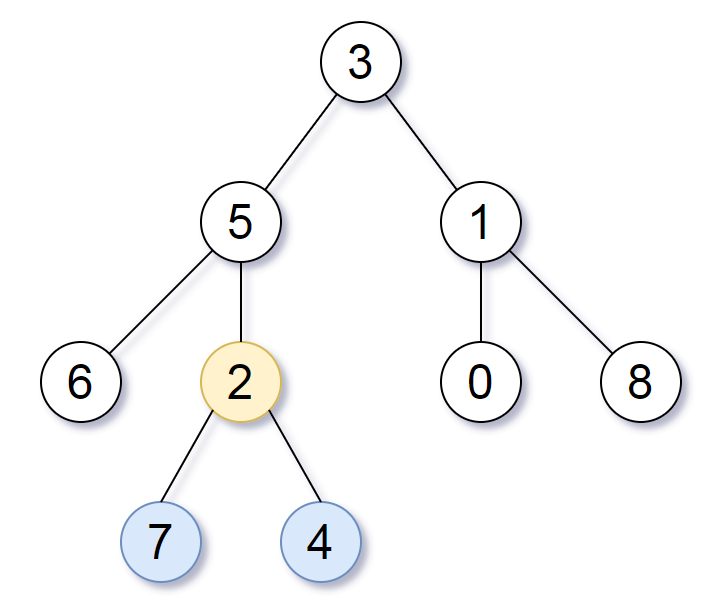
树的根节点的深度为 0，如果某一节点的深度为d，那它的子节点的深度就是 d+1

如果我们假定A是一组节点 S 的最近公共祖先，S中的每个节点都在以A为根节点的子树中，且A 的深度达到此条件下可能的最大值。

注意：本题与力扣 865 重复：

https://leetcode-cn.com/problems/smallest-subtree-with-all-the-deepest-nodes/

示例 1：



输入：root = [3,5,1,6,2,0,8,null,null,7,4]

输出：[2,7,4]

解释：

我们返回值为 2 的节点，在图中用黄色标记。

在图中用蓝色标记的是树的最深的节点。

注意，节点 6、0 和 8 也是叶节点，但是它们的深度是 2 ，而节点 7 和 4 的深度是 3 。

示例 2：

输入：root = [1]

输出：[1]

解释：根节点是树中最深的节点，它是它本身的最近公共祖先。

示例 3：

输入：root = [0,1,3,null,2]

输出：[2]

解释：树中最深的叶节点是 2 ，最近公共祖先是它自己。

提示：

给你的树中将有1到1000个节点。

树中每个节点的值都在1到1000之间。

每个节点的值都是独一无二的。

# 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 {

private:

    TreeNode \*helper(TreeNode \*root,int &height)

    {

        if(nullptr==root)

        {

            height = -1;

            return nullptr;

        }

        int leftHeight;

        int rightHeight;

        TreeNode \*left = helper(root->left,leftHeight);

        TreeNode \*right = helper(root->right,rightHeight);

        height = max(leftHeight,rightHeight)+1;

        if(leftHeight==rightHeight)  return root;

        if(leftHeight>rightHeight) return left;

        return right;

    }

public:

    TreeNode\* lcaDeepestLeaves(TreeNode\* root) {

        int height = 0;

        return helper(root,height);;

    }

};