# 题目

给定一个根为 root 的二叉树，每个节点的深度是该节点到根的最短距离。

如果一个节点在整个树的任意节点之间具有最大的深度，则该节点是最深的。

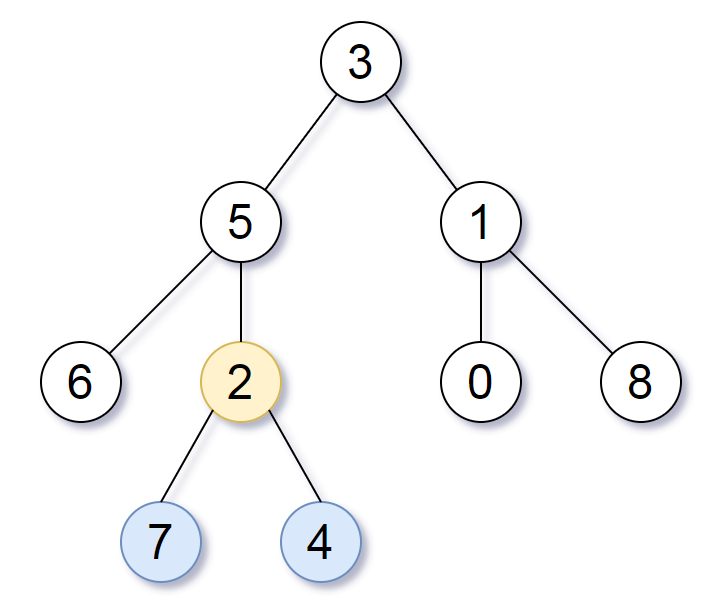
一个节点的子树是该节点加上它的所有后代的集合。

返回能满足以该节点为根的子树中包含所有最深的节点 这一条件的具有最大深度的节点。

注意：本题与力扣1123重复：

https://leetcode-cn.com/problems/lowest-common-ancestor-of-deepest-leaves/

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 5、3 和 2 包含树中最深的节点，但节点 2 的子树最小，因此我们返回它。

示例 2：

输入：root = [1]

输出：[1]

解释：根节点是树中最深的节点。

示例 3：

输入：root = [0,1,3,null,2]

输出：[2]

解释：树中最深的节点为 2 ，有效子树为节点 2、1 和 0 的子树，但节点 2 的子树最小。

提示：

树中节点的数量介于1和 500之间。

0 <= Node.val <= 500

每个节点的值都是独一无二的。

# 分析

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\* subtreeWithAllDeepest(TreeNode\* root) {

        int height = 0;

        return helper(root,height);;

    }

};