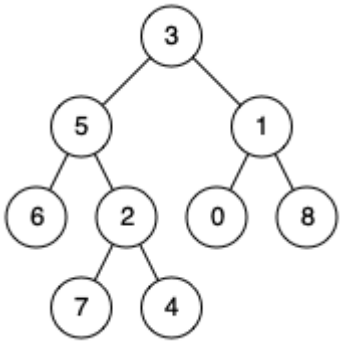
第15题——二叉树的最近共祖先

给定一个二叉树, 找到该树中两个指定节点的最近公共祖先。

百度百科中最近公共祖先的定义为：“对于有根树 T 的两个结点 p、q，最近公共祖先表示为一个结点 x，满足 x 是 p、q 的祖先且 x 的深度尽可能大（一个节点也可以是它自己的祖先）。”

例如，给定如下二叉树: root = [3,5,1,6,2,0,8,null,null,7,4]



示例 1:

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

```
输出: 3
```

解释: 节点 5 和节点 1 的最近公共祖先是节点 3。 示例 2:

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

```
输出: 5
```

解释: 节点 5 和节点 4 的最近公共祖先是节点 5。因为根据定义最近公共祖先节点可以为节点本身。

回溯算法(1)

所有节点的值都是唯一的。 p、q 为不同节点且均存在于给定的二叉树中。

解题思路： 祖先的定义： 若节点 pp 在节点 rootroot 的左（右）子树中，或 p = rootp=root ，则称 rootroot 是 pp 的祖先。

最近公共祖先的定义： 设节点 rootroot 为节点 p, qp,q 的某公共祖先，若其左子节点 root.leftroot.left 和右子节点 root.rightroot.right 都不是 p,qp,q 的公共祖先，则称 rootroot 是 “最近的公共祖先” 。

根据以上定义，若 rootroot 是 p, qp,q 的 最近公共祖先 ， 则只可能为以下情况之一：

pp 和 qq 在 rootroot 的子树中，且分列 rootroot 的 异侧（即分别在左、右子树中）； p = rootp=root ，且 qq 在 rootroot 的左或右子树中； q = rootq=root ，且 pp 在 rootroot 的左或右子树中；

考虑通过递归对二叉树进行后序遍历，当遇到节点 pp 或 qq 时返回。从底至顶回溯，当节点 p, qp,q 在节点 rootroot 的异侧时，节点 rootroot 即为最近公共祖先，则向上返回 rootroot 。

递归解析： 终止条件： 当越过叶节点，则直接返回 nullnull； 当 rootroot 等于 p, qp,q ，则直接返回 rootroot； 递推工作： 开启递归左子节点，返回值记为 leftleft； 开启递归右子节点，返回值记为 rightright； 返回值： 根据 leftleft 和 rightright ，可展开为四种情况； 当 leftleft 和 rightright 同时为空： 说明 rootroot 的左 / 右子树中都不包含 p,qp,q ，返回 nullnull； 当 leftleft 和 rightright 同时不为空： 说明 p, qp,q 分列在 rootroot 的 异侧 （分别在 左 / 右子树），因此 rootroot 为最近公共祖先，返回 rootroot； 当 leftleft 为空， rightright 不为空： p,qp,q 都不在 rootroot 的左子树中，直接返回 rightright 。具体可分为两种情况： p,qp,q 其中一个在 rootroot 的 右子树 中，此时 rightright 指向 pp（假设为 pp ）； p,qp,q 两节点都在 rootroot 的 右子树 中，此时的 rightright 指向 最近公共祖先节点； 当 leftleft 不为空， rightright 为空： 与情况 3. 同理； 观察发现， 情况 1. 可合并至 3. 和 4. 内。

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
struct TreeNode* lowestCommonAncestor(struct TreeNode* root, struct TreeNode* p, struct TreeNode*
q){
    if(root==q || root==p || !root)
    {
        return root;
    }
    struct TreeNode* left=lowestCommonAncestor(root->left,p,q);
    struct TreeNode* right=lowestCommonAncestor(root->right,p,q);
    if(left!=NULL && right!=NULL)
    {
        return root;//已找到
    }
    if(left==NULL && right!=NULL)
    {
        return right;//右子树不为空，返回右子树继续递归
    }
    else if(left!=NULL && right==NULL)
    {
        return left;//左子树不为空，返回左子树继续递归
    }
    else
```

```
    {
        return NULL;//左右子树都为空，返回NULL
    }
}
```