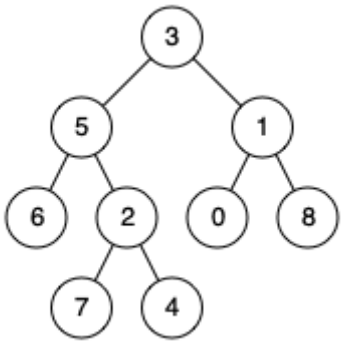
二叉树的最近公共祖先

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。

题解

1、 如果结点 p、q 都存在且为左右结点，那么根结点 root 就是最近公共祖先；

2、 如果结点 p、q 都存在且都为左结点，那么在根结点 root 的左子树查找；

3、 如果结点 p、q 都存在且都为右结点，那么在根结点 root 的右子树查找。

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
```

```
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q
) {
        if(root==nullptr||p==root||q==root)/* 二叉树为空或者结点 p/q 为根结
点 */
            return root;

        /* 在根结点的左右子树查找 */
        TreeNode* left=lowestCommonAncestor(root->left,p,q);
        TreeNode* right=lowestCommonAncestor(root->right,p,q);

        /* 结点 p/q 不在左子树中，就在右子树中查找，如果能找到，就返回在右子树中
找到的结点（反之亦然） */
        /* 结点 p/q 分别存在左右两颗子树， 根结点为最近公共祖先 */
        /* 结点 p/q 在左右子树都找不到，则它们没有最近公共祖先 */
        return left == nullptr ? right : right == nullptr ? left : root;
    }
};
```