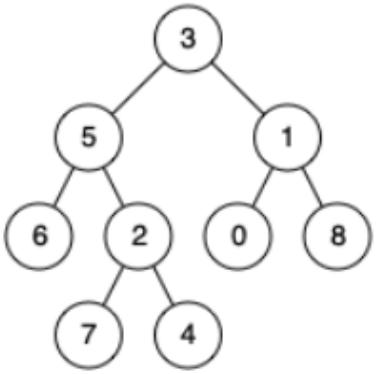
二叉树的最近公共祖先

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 的两个节点 p、q，最近公共祖先表示为一个节点 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**）。”

示例 1：

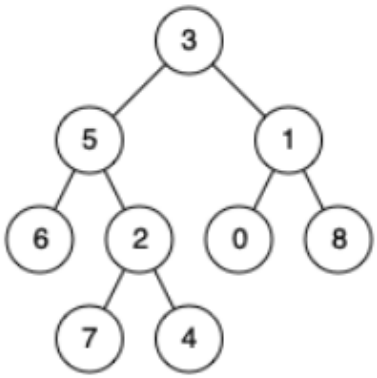


输入：`root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1`

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。

示例 3：

输入：root = [1,2], p = 1, q = 2

输出：1

```
/**
 * 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:
    bool findpath(TreeNode* root, TreeNode* target,stack<TreeNode*>& st)
    {
        if(root==nullptr)
            return false;
        st.push(root);
        if(root==target)
        {
            return true;
        }
        if(findpath(root->left,target,st))
            return true;
        if(findpath(root->right,target,st))
            return true;
        //当前节点不在查找路径上 删除
        st.pop();
        return false;
    }
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q
) {
        //通过栈实现
        stack<TreeNode*> path1;
        stack<TreeNode*> path2;
        //路径查找
        findpath(root,p,path1);
        findpath(root,q,path2);
        //路径裁剪
        while(path1.size()!=path2.size())
        {
            if(path1.size()>path2.size())
                path1.pop();
```

```
            else
                path2.pop();
        }
        //比对
        while(!path1.empty())
        {
            if(path1.top()==path2.top())
                return path1.top();
            else
            {
                path1.pop();
                path2.pop();
            }
        }
        return path1.top();
    }
};
```