1. Lowest Common Ancestor of a Binary Tree

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the [definition of LCA on Wikipedia](https://en.wikipedia.org/wiki/Lowest_common_ancestor): “The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**).”

Given the following binary tree: root = [3,5,1,6,2,0,8,null,null,7,4]

img

img

**Example 1:**

Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1  
Output: 3  
Explanation: The LCA of nodes 5 and 1 is 3.

**Example 2:**

Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 4  
Output: 5  
Explanation: The LCA of nodes 5 and 4 is 5, since a node can be a descendant of itself according to the LCA definition.

**Note:**

* All of the nodes’ values will be unique.
* p and q are different and both values will exist in the binary tree.

**解** 对于root节点，如果左右子树分别有一个节点，则root是一个公共祖先

**解1** 递归

class Solution {  
public:  
 TreeNode \*ans;  
 TreeNode\* lowestCommonAncestor(TreeNode\* root, TreeNode\* p, TreeNode\* q) {  
 LCA(root, p, q);  
 return ans;  
 }  
 bool LCA(TreeNode \*root, TreeNode \*p, TreeNode \*q){  
 if(root == NULL)return false;  
 int l = LCA(root->left, p, q) ? 1 : 0; // p或者q在左子树  
 int r = LCA(root->right, p, q) ? 1 : 0; // p或者q在右子树  
 int mid = (root == p || root == q) ? 1 : 0; //找到了p或者q  
 if(mid + l + r >= 2)ans = root; // =2时，左右各一个；>2时，一个是另一个的先驱节点  
 return mid + l + r > 0; // >0表明找到了p或者q  
 }  
};