# 题目

给定一个二叉搜索树, 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，给定如下二叉搜索树:  root = [6,2,8,0,4,7,9,null,null,3,5]



**示例 1:**

输入: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8

输出: 6

解释: 节点 2 和节点 8 的最近公共祖先是 6。

**示例 2:**

输入: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4

输出: 2

解释: 节点 2 和节点 4 的最近公共祖先是 2, 因为根据定义最近公共祖先节点可以为节点本身。

**说明:**

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搜索树中。

# 分析

## 方法一：递归法

**思路：**

**代码：**

/\*\*

\* 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(NULL==root || root==p || root==q)

return root;

TreeNode \*leftNode = lowestCommonAncestor(root->left,p,q);

TreeNode \*rightNode = lowestCommonAncestor(root->right,p,q);

if(NULL==leftNode)return rightNode;

if(NULL==rightNode)return leftNode;

return root;

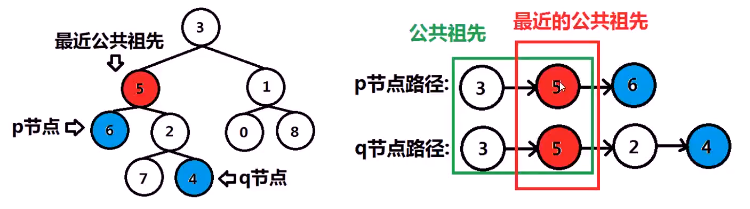
}

};

## 方法二：迭代法

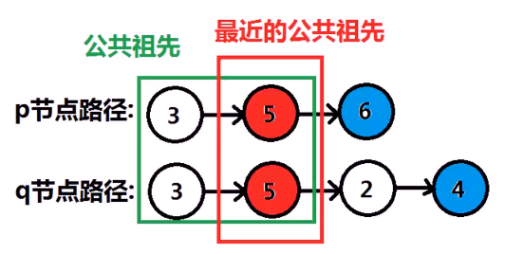
**思路：**

1. 两个节点的公共祖先一定在从根节点至这两个节点的路径上；
2. 由于求公共祖先中的最近公共祖先，那么即同时出现在这两条路径上的离根节点最远的节点（或离两个最近）；
3. 最终算法即：求p节点路径，q节点路径，两路径上最后一个相同的节点。



根据上述求根节点到某节点路径的算法：

1. 求出较短路径的长度n；
2. 同时遍历p节点的路径与q节点的路径，遍历n个节点，最后一个发现的相同节点，即最近公共祖先。



**代码：**



**测试：**

