# 题目

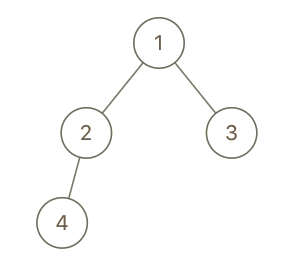
在二叉树中，根节点位于深度0处，每个深度为k的节点的子节点位于深度k+1处。

如果二叉树的两个节点深度相同，但父节点不同，则它们是一对堂兄弟节点。

我们给出了具有唯一值的二叉树的根节点root，以及树中两个不同节点的值x和y。

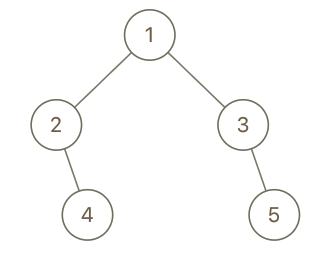
只有与值x和y对应的节点是堂兄弟节点时，才返回true。否则，返回false。

示例 1：



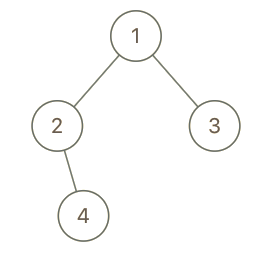
输入：root = [1,2,3,4], x = 4, y = 3输出：false

示例 2：



输入：root = [1,2,3,null,4,null,5], x = 5, y = 4输出：true

示例 3：



输入：root = [1,2,3,null,4], x = 2, y = 3

输出：false

提示：

二叉树的节点数介于2到100之间。

每个节点的值都是唯一的、范围为1到100的整数。

# 分析

这道题目当然用BFS或者DFS都可以做，但是用DFS是不如BFS优雅的。

细节是魔鬼，本题有两个关键点：

1、深度相同

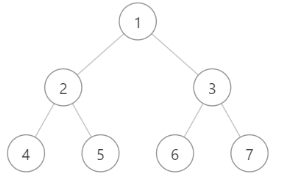
2、父节点不同

在确定深度上面，BFS一直很可以的。

具体思路如下：

因为在BFS中，我们使用的是层序遍历，如果每次遍历一层，那么这一层的元素的深度是相同的。

例子：（第i层深度为i）



第一层：[1]

第二层：[2,3]

第三层：[4,5,6,7]

因此我们在每一层，看看是否有出现 x 和 y，其中分为以下三种情况：

1、x 和 y 都没出现 → 那只能往下一层继续找了

2、x 和 y 只出现一个 → 两个元素的深度不同，不可能为兄弟，返回false

3、x 和 y 都出现了，好耶，但是还不够好

x 和 y 父节点相同 → 不是堂兄弟，是亲兄弟，不行，返回false

x 和 y 父节点不同 → 满足题目条件了，好耶，返回true

众所周知，BFS需要用到队列，那么我们应该如何设计队列的数据类型？

在这里，我采用了 pair<TreeNode\*, TreeNode\*>（其实pair<TreeNode\*, int>也可以），其中pair中，第一个元素记录指向当前结点的指针，第二个元素记录指向当前结点的父结点的指针，这样就可以完美应对上面所说的三种情况了。

**代码：**

class Solution {

public:

bool isCousins(TreeNode\* root, int x, int y) {

queue<pair<TreeNode\*, TreeNode\*>> que;

que.push({root, nullptr});

while (!que.empty()) {

// 记录当前层的节点数

int n = que.size();

// 记录是否找到x,y

bool foundX = false, foundY = false;

// 记录当前匹配的x,y的父节点

TreeNode\* parentX = nullptr, \*parentY = nullptr;

for (int i = 0; i < n; i++) {

auto [node, parent] = que.front();

que.pop();

if (node->val == x) {

foundX = true;

parentX = parent;

}

if (node->val == y) {

foundY = true;

parentY = parent;

}

if (node->left) que.push({node->left, node});

if (node->right) que.push({node->right, node});

}

// 如果找到x,y并且父节点不同

if (foundX && foundY && parentX != parentY) {

return true;

}

// 只找到x或y

if (foundX || foundY) {

return false;

}

}

return false;

}

};