Back to Chapter

Week 1: May 1st–May 7th
Problems appear at midnight, Pa

First Bad Version
Jewels and Stones
Ransom Note
Number Complement
First Unique Character in a ...
Majority Element
Cousins in Binary Tree

Week 2: May 8th–May 14th
Problems appear at midnight, Pa

Week 3: May 15th–May 21st
The first problem for this section

Previous Next

Go to Discuss

Cousins in Binary Tree

Solution

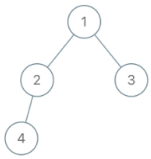
In a binary tree, the root node is at depth `0`, and children of each depth `k` node are at depth `k+1`.

Two nodes of a binary tree are *cousins* if they have the same depth, but have **different parents**.

We are given the `root` of a binary tree with unique values, and the values `x` and `y` of two different nodes in the tree.

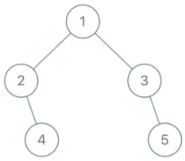
Return `true` if and only if the nodes corresponding to the values `x` and `y` are cousins.

Example 1:



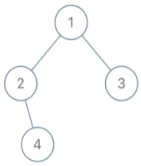

```
Input: root = [1,2,3,4], x = 4, y = 3
Output: false
```

Example 2:




```
Input: root = [1,2,3,null,4,null,5], x = 5, y = 4
Output: true
```

Example 3:




```
Input: root = [1,2,3,null,4], x = 2, y = 3
Output: false
```

Note:

1. The number of nodes in the tree will be between `2` and `100`.
2. Each node has a unique integer value from `1` to `100`.

Java

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    class Node{
        TreeNode root;
```

```
        int depth;
        TreeNode parent;
        public Node(TreeNode root,TreeNode parent,int depth){
            this.root=root;
            this.parent=parent;
            this.depth=depth;
        }
    }
    public boolean isCousins(TreeNode root, int x, int y) {
        if(root==null) return true;
        Queue<Node> queue=new LinkedList<>();
        queue.add(new Node(root,null,0));
        int depth1=0;
        int depth2=0;
        int parent1=0;
        int parent2=0;
        while(!queue.isEmpty()){
            Node node=queue.poll();
            int depth=node.depth;
            TreeNode temp=node.root;
            TreeNode tempP=node.parent;
            if(temp!=null && depth1==0 && temp.val==x && tempP!=null){
                depth1=depth;
                parent1=tempP.val;
            }
            if(temp!=null && depth2==0 && temp.val==y && tempP!=null){
                depth2=depth;
                parent2=tempP.val;
            }
            if(temp.left!=null) queue.add(new Node(temp.left,temp,depth+1));
            if(temp.right!=null) queue.add(new Node(temp.right,temp,depth+1));
        }
        if(depth1==depth2 && parent1!=parent2){
            return true;
        }
        return false;
    }
}
```

Custom Testcase (Contribute)

Run Code Submit