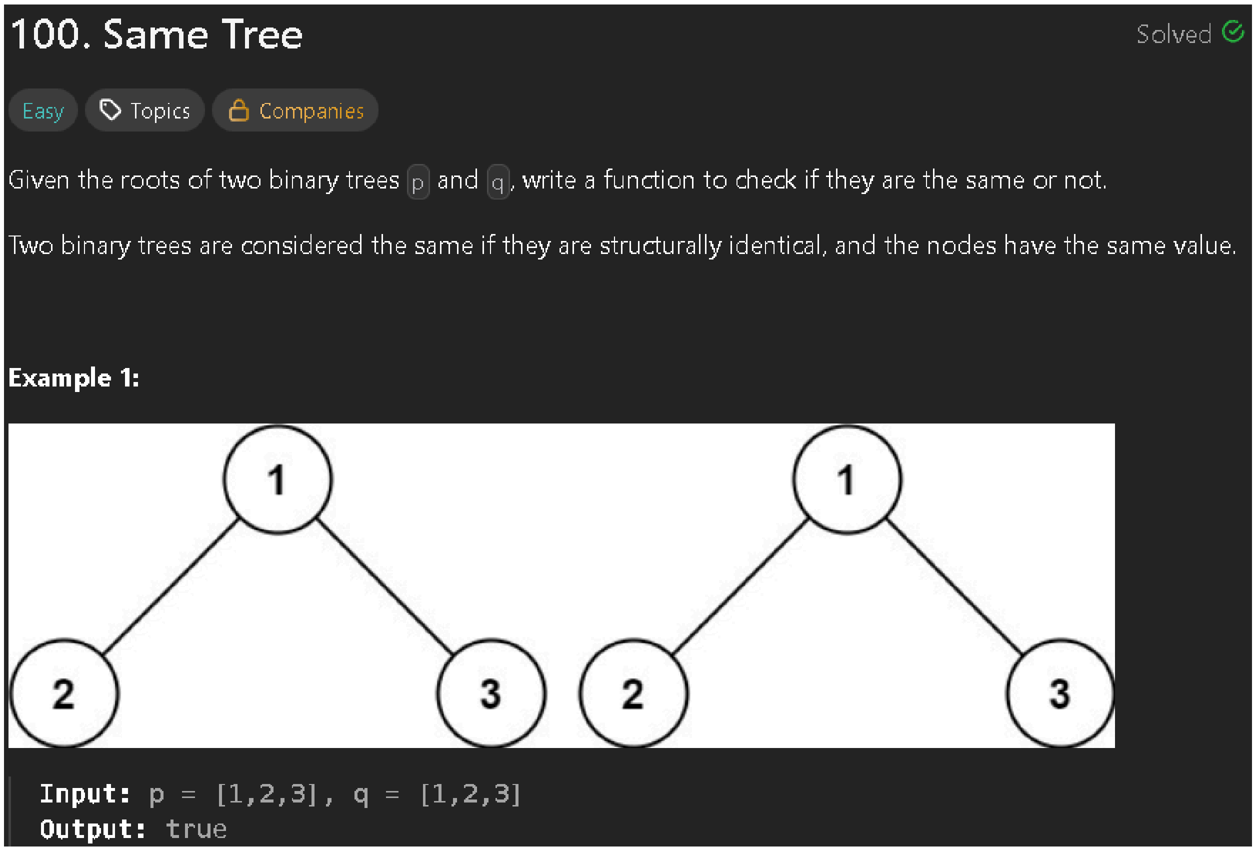


100. Same Tree

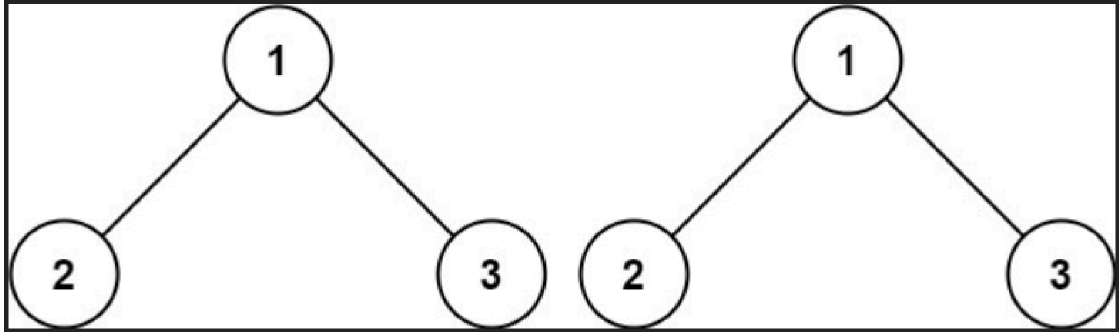
Solved

Easy Topics Companies

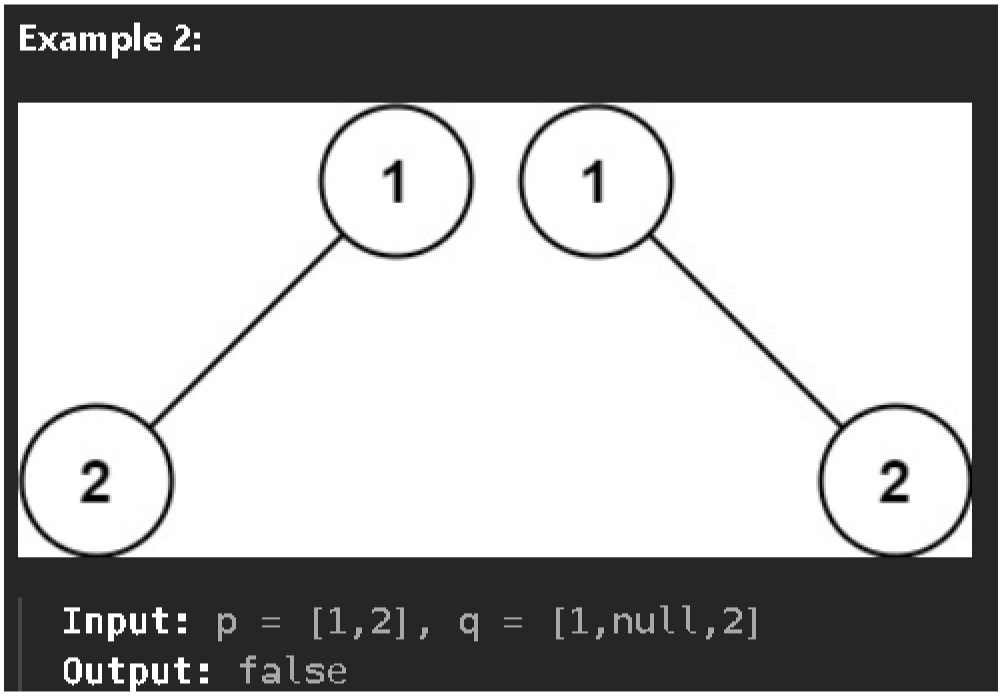
Given the roots of two binary trees `p` and `q`, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

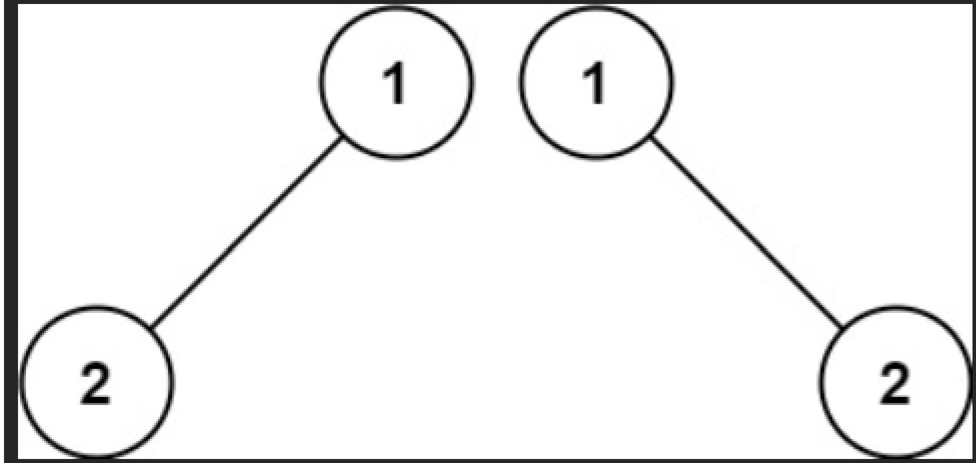
Example 1:



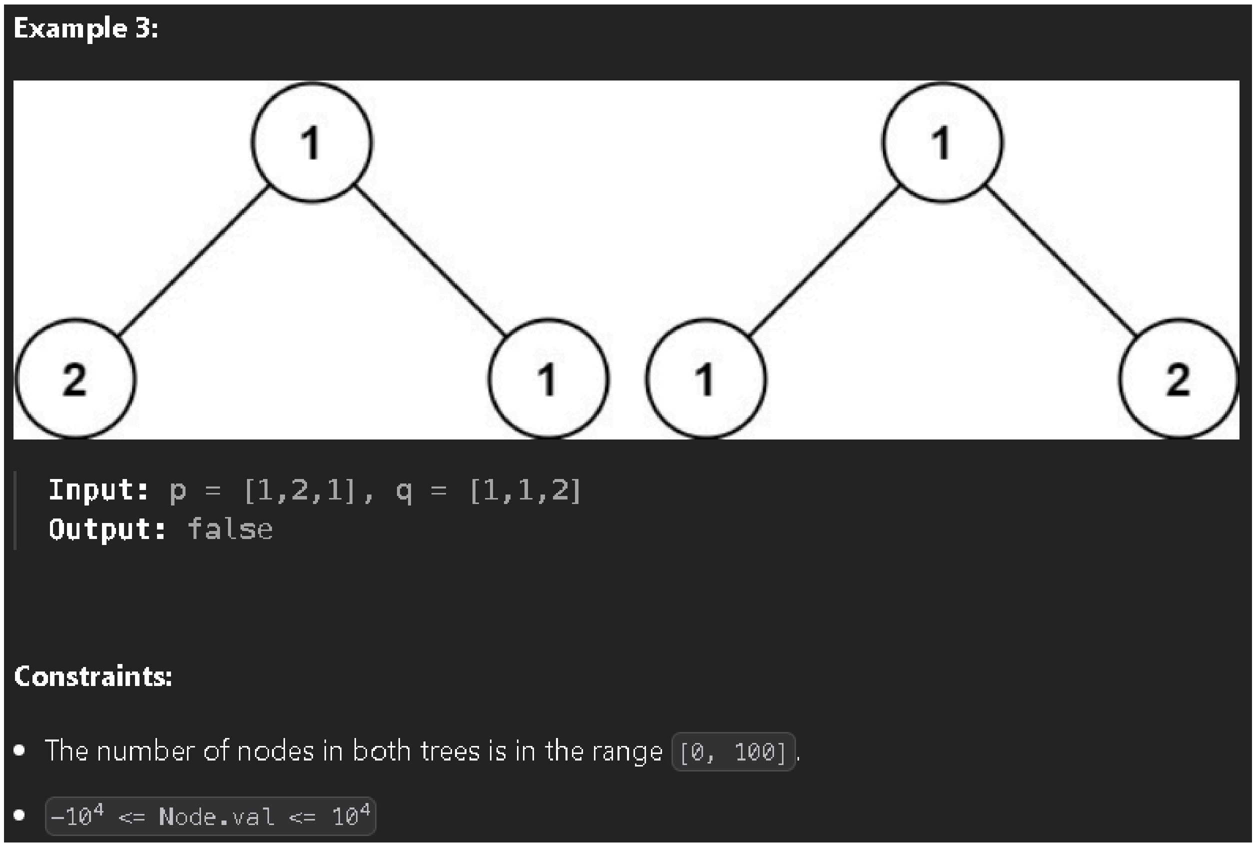
```
Input: p = [1,2,3], q = [1,2,3]
Output: true
```

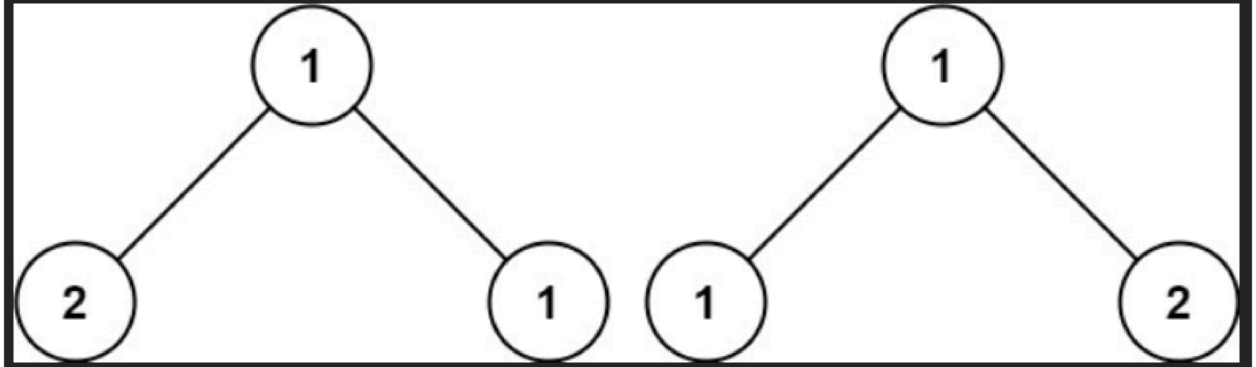


Example 2:



```
Input: p = [1,2], q = [1,null,2]
Output: false
```


Example 3:



```
Input: p = [1,2,1], q = [1,1,2]
Output: false
```

Constraints:

- The number of nodes in both trees is in the range `[0, 100]`.
- `-10^4 <= Node.val <= 10^4`

Python:

```
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def isSameTree(self, p: Optional[TreeNode], q: Optional[TreeNode]) -> bool:
        # If both are None, trees are same
        if not p and not q:
            return True
        # If one is None and the other isn't, trees are different
        if not p or not q:
            return False
        # If values don't match, trees are different
        if p.val != q.val:
            return False
        # Recursively check left and right subtrees
        return self.isSameTree(p.left, q.left) and self.isSameTree(p.right, q.right)
```

JavaScript:

```
/**
 * Definition for a binary tree node.
*/
function TreeNode(val, left, right) {
    this.val = (val===undefined ? 0 : val)
    this.left = (left===undefined ? null : left)
    this.right = (right===undefined ? null : right)
}
/**
 * @param {TreeNode} p
 * @param {TreeNode} q
 * @return {boolean}
 */
var isSameTree = function(p, q) {
    // Case 1: Both are null
  if (p === null && q === null) return true;

  // Case 2: One is null but the other is not
  if (p === null || q === null) return false;

  // Case 3: Values are different
  if (p.val !== q.val) return false;

  // Case 4: Recurse on left and right subtrees
  return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
};
```

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 {
    public boolean isSameTree(TreeNode p, TreeNode q) {
        // Case 1: Both are null
        if (p == null && q == null) {
            return true;
        }

        // Case 2: One of them is null
        if (p == null || q == null) {
            return false;
        }

        // Case 3: Values are different
        if (p.val != q.val) {
            return false;
        }

        // Recursively check left and right subtrees
        return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
    }
}
```