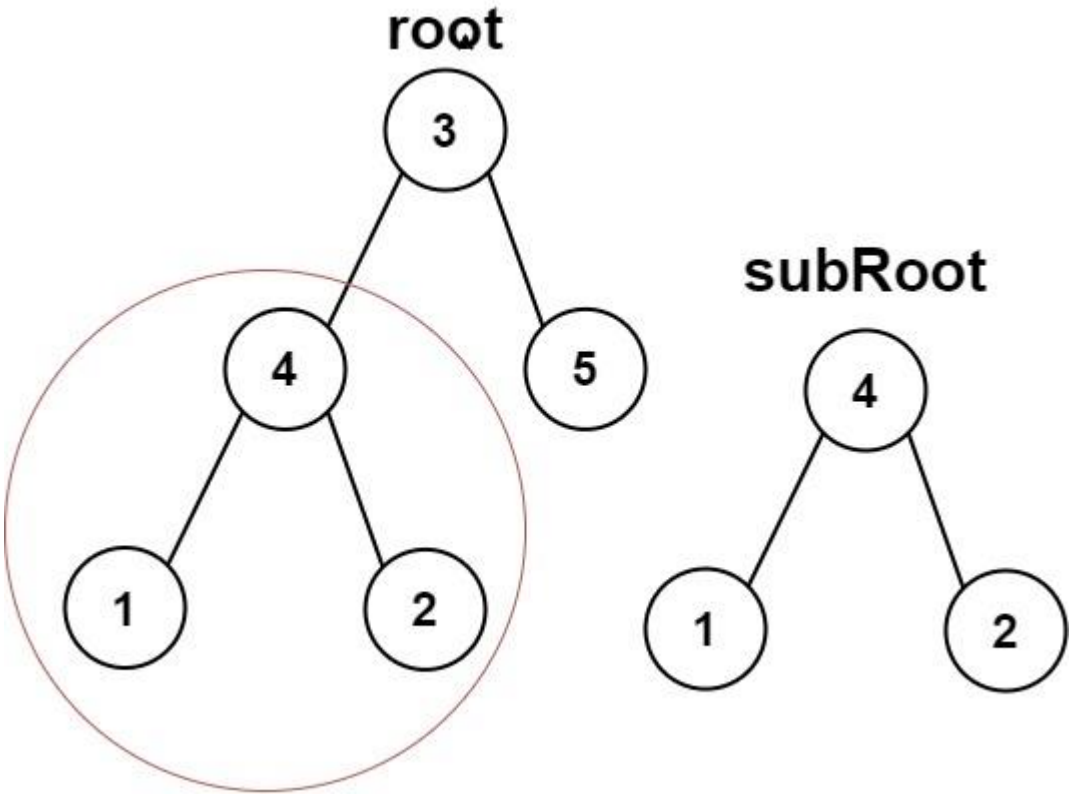
572. Subtree of Another Tree

Given the roots of two binary trees root and subRoot, return true if there is a subtree of root with the same structure and node values of subRoot and false otherwise.

A subtree of a binary tree tree is a tree that consists of a node in tree and all of this node's descendants. The tree tree could also be considered as a subtree of itself.

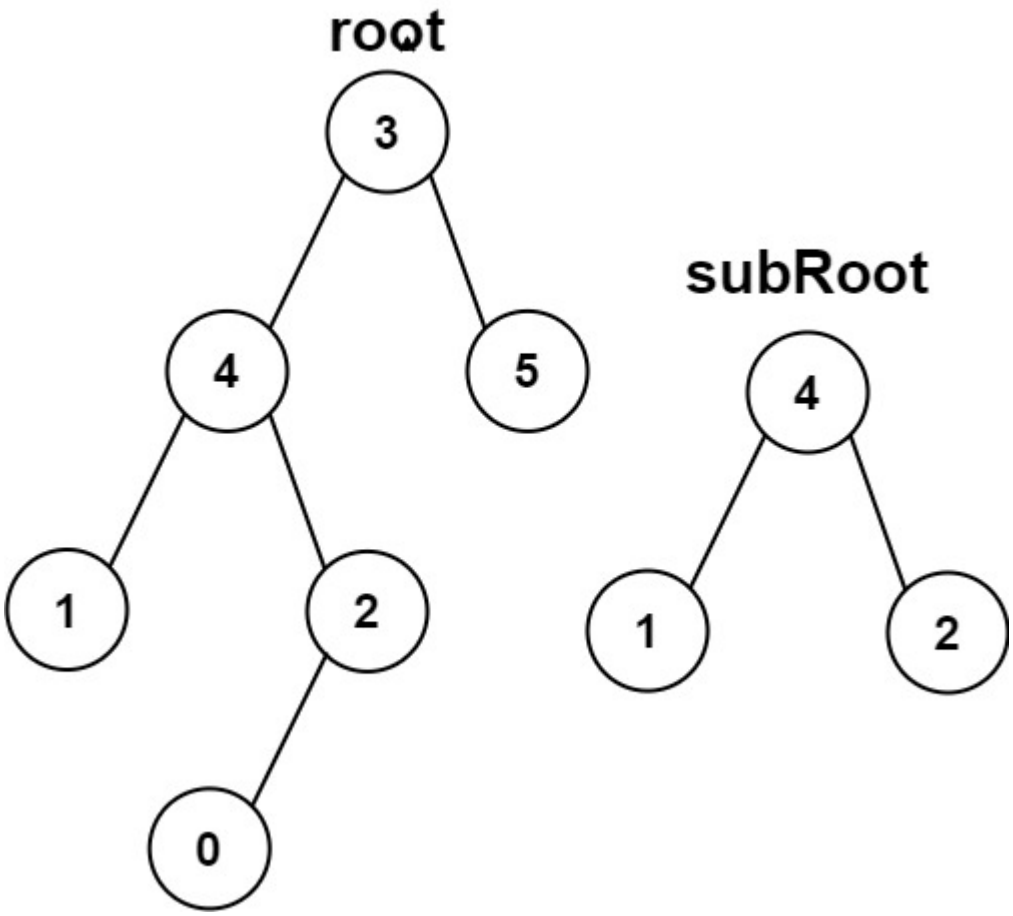
Example 1:



Input: root = [3,4,5,1,2], subRoot = [4,1,2]

Output: true

Example 2:



Input: root = [3,4,5,1,2,null,null,null,null,0], subRoot = [4,1,2]

Output: false

Constraints:

- The number of nodes in the root tree is in the range [1, 2000].
- The number of nodes in the subRoot tree is in the range [1, 1000].
- -10^4 <= root.val <= 10^4
- -10^4 <= subRoot.val <= 10^4

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution(object):
    def isSubtree(self, root, subRoot):
        """
        :type root: TreeNode
        :type subRoot: TreeNode
        :rtype: bool
        """
        if not root:
            return False
        if self.isSameTree(root,subRoot):
            return True
        return self.isSubtree(root.left, subRoot) or
self.isSubtree(root.right,subRoot)

    def isSameTree(self,root,subRoot):
        if root and subRoot:
            return root.val == subRoot.val and
self.isSameTree(root.left, subRoot.left) and
self.isSameTree(root.right, subRoot.right)
        return root is subRoot
```