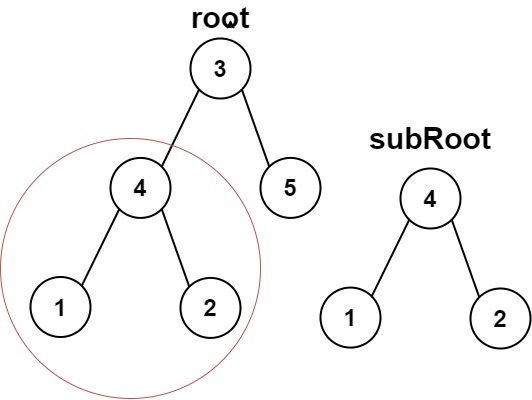
Given the roots of two binary trees root and subRoot, return true if there is a subtree of root with the same structure and node values of subRoot and false otherwise.

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.

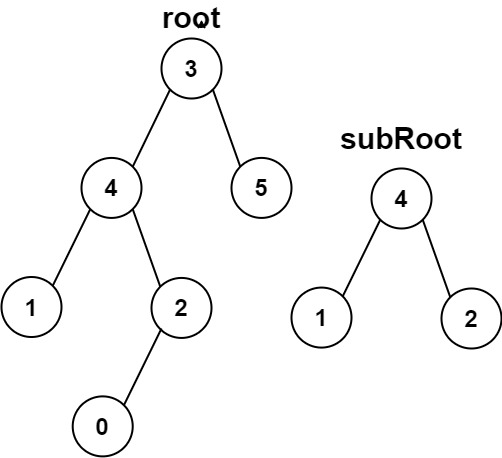
**Example 1:**



**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

Solution:

/\*\*

\* 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 isSubtree(TreeNode s, TreeNode t) {

if(s == null)

return false;

else if(isSameTree(s, t))

return true;

else{

return isSubtree(s.left, t) || isSubtree(s.right, t);

}

}

public boolean isSameTree(TreeNode s, TreeNode t){

if(s == null || t == null)

return s== null && t == null;

else if(s.val == t.val)

return isSameTree(s.left, t.left) && isSameTree(s.right, t.right);

else

return false;

}

}

**Time Complexity:** Time worst-case complexity of above solution is O(mn) where m and n are number of nodes in given two trees.