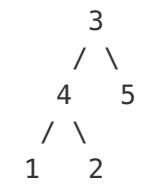
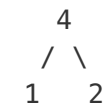
Subtree of Another Tree

Given two non-empty binary trees **s** and **t**, check whether tree **t** has exactly the same structure and node values with a subtree of **s**. A subtree of **s** is a tree consists of a node in **s** and all of this node's descendants. The tree **s** could also be considered as a subtree of itself.

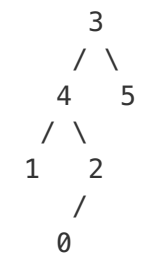
Example 1:
Given tree s:



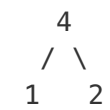
Given tree t:



Return **true**, because t has the same structure and node values with a subtree of s.
Example 2:
Given tree s:



Given tree t:



Return **false**.

Solution 1

For each node during pre-order traversal of `s`, use a recursive function `isSame` to validate if sub-tree started with this node is the same with `t`.

```
public class Solution {
    public boolean isSubtree(TreeNode s, TreeNode t) {
        if (s == null) return false;
        if (isSame(s, t)) return true;
        return isSubtree(s.left, t) || isSubtree(s.right, t);
    }

    private boolean isSame(TreeNode s, TreeNode t) {
        if (s == null && t == null) return true;
        if (s == null || t == null) return false;

        if (s.val != t.val) return false;

        return isSame(s.left, t.left) && isSame(s.right, t.right);
    }
}
```

written by shawngao original link here

Solution 2

```
public class Solution {
 public boolean isSubtree(TreeNode s, TreeNode t) {
        String spreorder = generatepreorderString(s);
        String tpreorder = generatepreorderString(t);

        return spreorder.contains(tpreorder) ;
    }
    public String generatepreorderString(TreeNode s){
        StringBuilder sb = new StringBuilder();
        Stack<TreeNode> stacktree = new Stack();
        stacktree.push(s);
        while(!stacktree.isEmpty()){
           TreeNode popelem = stacktree.pop();
           if(popelem==null)
              sb.append(",#"); // Appending # inorder to handle same values but not
subtree cases
           else
              sb.append(","+popelem.val);
           if(popelem!=null){
                stacktree.push(popelem.right);
                stacktree.push(popelem.left);
           }
        }
        return sb.toString();
    }
}
```

written by giridhar_bhageshpur original link here

Solution 3

```
public boolean isSubtree(TreeNode s, TreeNode t) {
    return serialize(s).contains(serialize(t)); // Java use a naive contains algorit
hm so to ensure linear time,
                                                // replace with KMP algorithm
}

public String serialize(TreeNode root) {
    StringBuilder res = new StringBuilder();
    serialize(root, res);
    return res.toString();
}

private void serialize(TreeNode cur, StringBuilder res) {
    if (cur == null) {res.append(",#"); return;}
    res.append("," + cur.val);
    serialize(cur.left, res);
    serialize(cur.right, res);
}
```

written by compton_scatter original link here