Same Tree

Given two binary trees, write a function to check if they are equal or not.

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

Solution 1

```
public boolean isSameTree(TreeNode p, TreeNode q) {
    if(p == null && q == null) return true;
    if(p == null || q == null) return false;
    if(p.val == q.val)
        return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
    return false;
}
```

written by micheal.zhou original link here

Solution 2

```
//
// Algorithm for the recursion:
// 1)
// If one of the node is NULL then return the equality result of p an q.
// This boils down to if both are NULL then return true,
// but if one of them is NULL but not the other one then return false
// 2)
// At this point both root nodes represent valid pointers.
// Return true if the root nodes have same value and
// the left tree of the roots are same (recursion)
// and the right tree of the roots are same (recursion).
// Otherwise return false.
//

bool isSameTree(TreeNode *p, TreeNode *q) {
    if (p == NULL || q == NULL) return (p == q);
    return (p->val == q->val && isSameTree(p->left, q->left) && isSameTree(p->righ
t, q->right));
}
```

written by satyakam original link here

Solution 3

the idea is to use stack for preorder traverse

```
public boolean isSameTree(TreeNode p, TreeNode q) {
         Stack<TreeNode> stack_p = new Stack <> ();
         Stack<TreeNode> stack_q = new Stack <> ();
         if (p != null) stack_p.push( p ) ;
         if (q != null) stack_q.push( q ) ;
         while (!stack_p.isEmpty() && !stack_q.isEmpty()) {
             TreeNode pn = stack_p.pop() ;
             TreeNode qn = stack_q.pop() ;
             if (pn.val != qn.val) return false ;
             if (pn.right != null) stack_p.push(pn.right) ;
             if (qn.right != null) stack_q.push(qn.right) ;
             if (stack_p.size() != stack_q.size()) return false ;
             if (pn.left != null) stack_p.push(pn.left) ;
             if (qn.left != null) stack_q.push(qn.left) ;
             if (stack_p.size() != stack_q.size()) return false ;
         }
         return stack_p.size() == stack_q.size() ;
     }
```

written by scott original link here