



→ `return self.dfs(root, subRoot)`

`dfs(a, b)`

```
def dfs(self, root, subRoot):
    if (root == None and subRoot != None): → False
        return False
    return self.compare(root, subRoot) or self.dfs(root.left, subRoot) or self.dfs(root.right, subRoot)
```

```
def compare(self, root, otherRoot):
    if root == None and otherRoot == None:
        return True
    if (root != None and otherRoot != None and root.val != otherRoot.val)
    or (root == None and otherRoot != None)
    or (otherRoot == None and root != None):
        return False ✓
    #print("comparing "+str(root.val)+" and "+str(otherRoot.val))
    return self.compare(root.left, otherRoot.left) and self.compare(root.right, otherRoot.right)
```

```
def dfs(self, root, subRoot):
    if (root == None and subRoot != None):
        return False
    return self.compare(root, subRoot) or self.dfs(root.left, subRoot) or self.dfs(root.right, subRoot)
```

```
def compare(self, root, otherRoot):
    if root == None and otherRoot == None:
        return True
    if (root != None and otherRoot != None and root.val != otherRoot.val)
    or (root == None and otherRoot != None)
    or (otherRoot == None and root != None):
        return False
    #print("comparing "+str(root.val)+" and "+str(otherRoot.val))
    return self.compare(root.left, otherRoot.left) and self.compare(root.right, otherRoot.right)
```

T and T

```
6         self.right = right
7     class Solution(object):
8         def isSubtree(self, root, subRoot):
9             """
10             :type root: TreeNode
11             :type subRoot: TreeNode
12             :rtype: bool
13             """
14             return self.dfs(root, subRoot)
15
16         def dfs(self, root, subRoot):
17             if (root == None and subRoot != None):
18                 return False
19
20             return self.compare(root, subRoot) or self.dfs(root.left, subRoot) or self.dfs(root.right, subRoot)
21
22         def compare(self, root, otherRoot):
23             if root == None and otherRoot == None:
24                 return True
25
26             if (root != None and otherRoot != None and root.val != otherRoot.val)
27             or (root == None and otherRoot != None)
28             or (otherRoot == None and root != None):
29                 return False
30             #print("comparing "+str(root.val)+" and "+str(otherRoot.val))
31             return self.compare(root.left, otherRoot.left) and self.compare(root.right, otherRoot.right)
```