572. Subtree of Another Tree

<https://leetcode.com/problems/subtree-of-another-tree/>

1. **Listen**

**Problem Statement:**

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.

**Input:**

roots of two binary trees

root

and

subRoot

**Goal:**

discern if there is a subtree of root with the same structure and node values of subRoot

in other words, see if the subRoot tree is in the root tree

**Return:**

return true if there is a subtree of root with the same structure and node values of subRoot

return false otherwise.

1. **Examples**

Example 1:

A picture containing text, clock

Description automatically generated

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

**Output:** true

Example 2:

Diagram, shape

Description automatically generated

**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].
* -104 <= root.val <= 104
* -104 <= subRoot.val <= 104

**Test Cases:**

* true – there is a subtree present in root
* false – there is not a subtree present in root
* true – subtree and root are equal

1. **Brute Force**

Solution 1:

We already know how to see if two trees are equal.

We could abstract this as a function: isSameTree(TreeNode p, TreeNode q)

We could use this function on every node in the tree to test every possible subtree for equality.

isSameTree runs in

Time Complexity:

O(m) time, where m is the number of nodes in the subtree

Space Complexity:

Worst Case: O(m) if subtree is skewed

Average Case: O(logm) if subtree is balanced

We traverse over every node in the tree and run isSameTree.

If we assume:

***n*** is the number of nodes in the main tree

***m*** is the number of nodes in the sub tree

By combining the runtime of iterating over all the nodes in the main tree and doing isSameTree for each node:

Time Complexity:

O(nm) if subtree is skewed

O(nlogm) if subtree is balanced

1. **Optimize**

Solution 2:

1. **Walkthrough**
2. **Implement**
3. **Test**