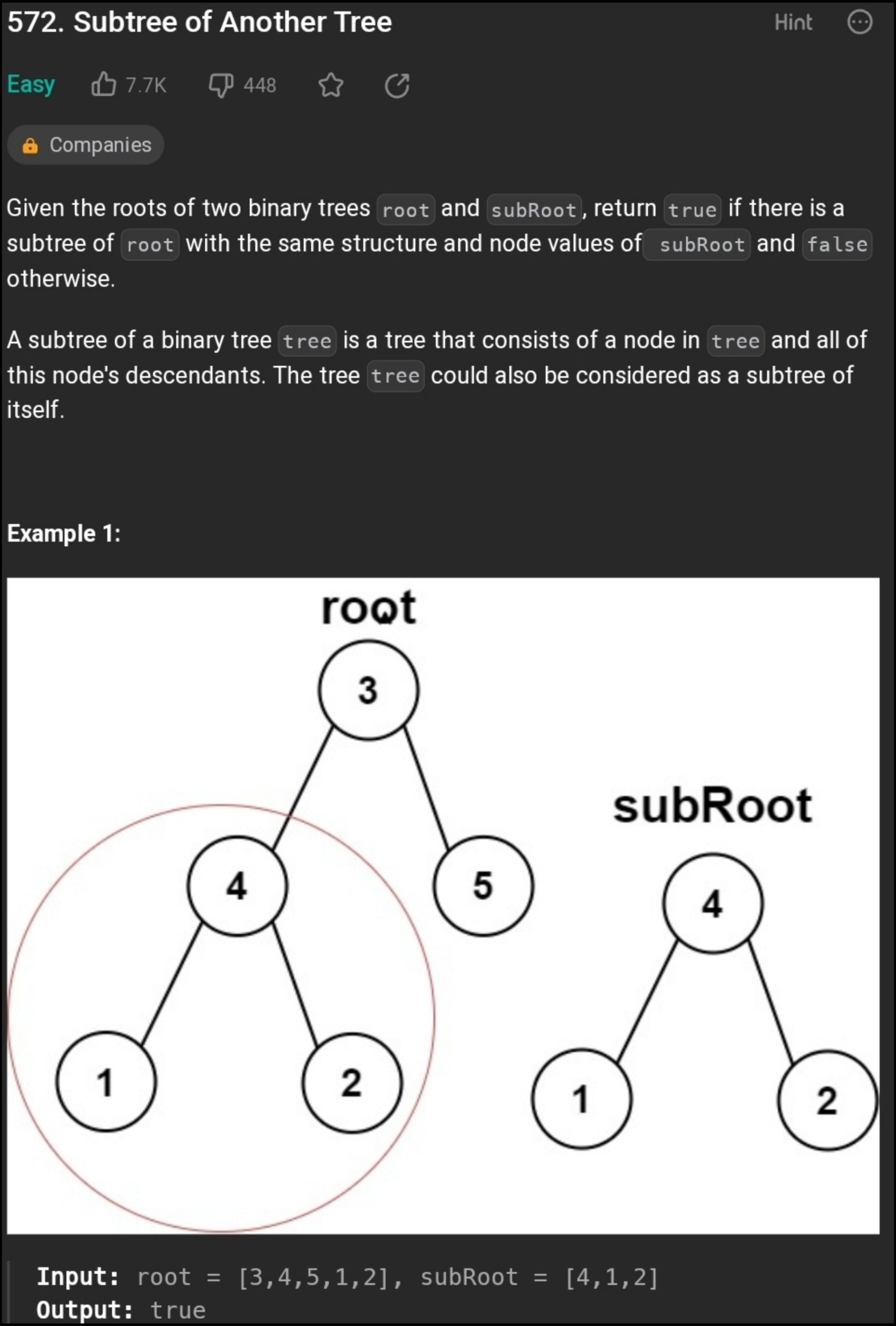


572. Subtree of Another Tree

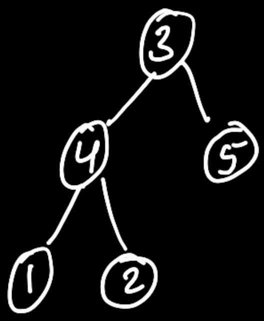
Given the roots of two binary trees `root` and `subRoot`, return `true` if there is a subtree of `root` with the same structure and node values of `subRoot` and `false` otherwise.

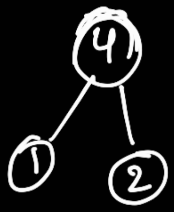
A subtree of a binary tree `tree` is a tree that consists of a node in `tree` and all of this node's descendants. The tree `tree` could also be considered as a subtree of itself.

Example 1:

```
Input: root = [3,4,5,1,2], subRoot = [4,1,2]
Output: true
```

Approach 1: Recursive implementation


Tree


Sub Tree

→ do any Traversal and at any node if Tree → val is equal to SubTree → val Then run isSameTree() algo on that nodes hoping they would be same. if it returns false just move along in the traversal and do the same at every node.

```
isSubtree( Tree ,Subtree)
{
    if( Tree is null)
        return false

    if( Tree val is equal to Subtree val)
        if( isSameTree( tree , subtree))
            return true

    return isSubtree( Tree→left ,SubTree)
              ||
           isSubtree( Tree→right , SubTree)
}
```

}

$$T(n) : O(n) + O(k\,\overset{\text{height of SubTree}}{h})$$

where

k = no. of nodes in tree having value same as Subroot→val.

Approach 2: Iterative Implementation

The order of visiting nodes is not an issue.

```
s.Push(Tree)

while(s is not empty)
{
        auto node = s.top()
            s.pop()

        if(node is null)
            don't do anything

        else if (node→val is equal to
                        SubRoot→val)
        {
            if (isSameTree(node, SubRoot)
                return true
        }

        else {
```

```
else {
        s.push(node->left)
        s.push(node->right)

}
```

$T(n) : O(n) + O(kh)$

$S(n) : O(h)$