LEETCODE WEEKLY CHALLENGE

653. Two Sum IV - Input is a BST

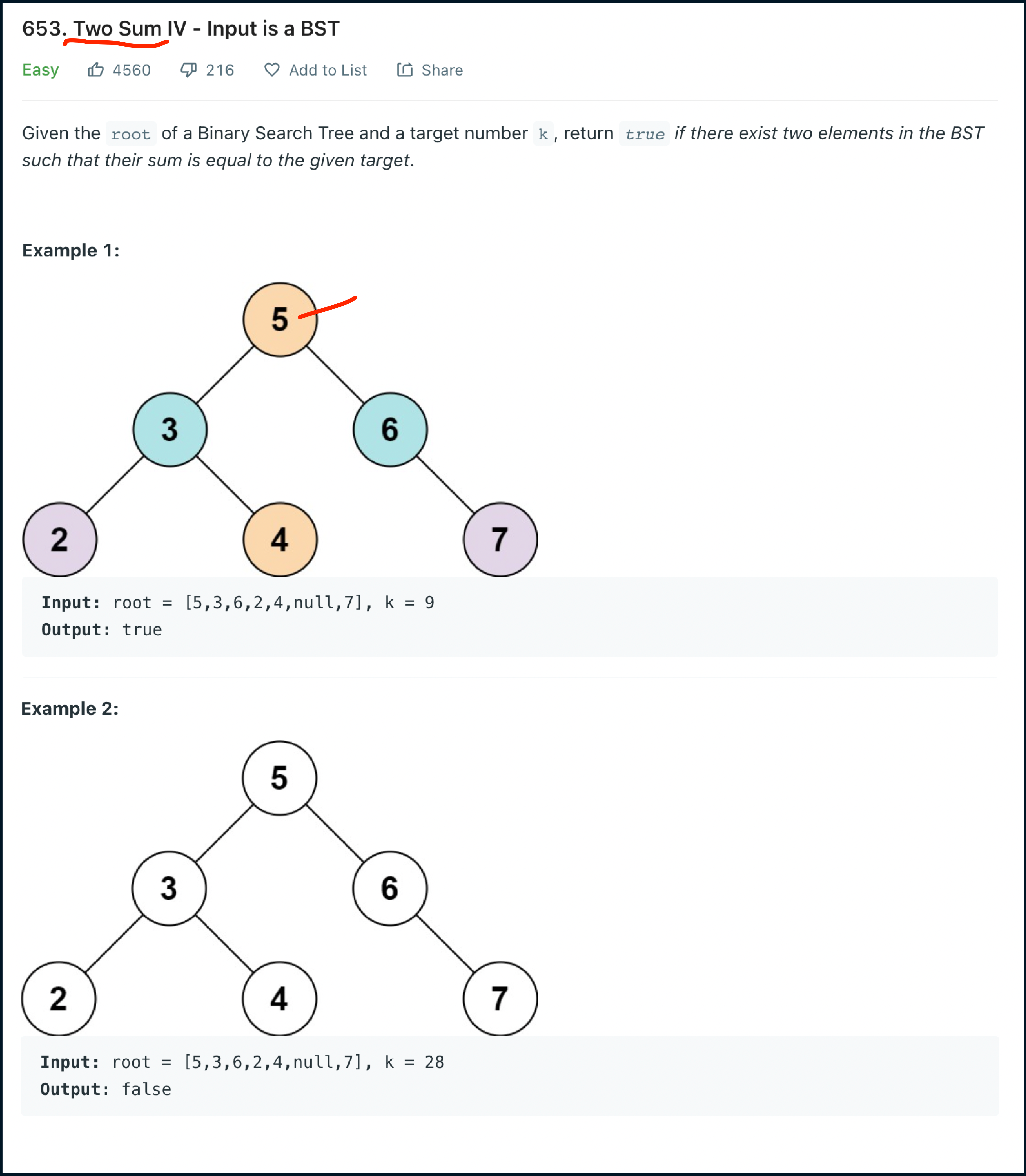
Using Hashset Approach

PROGRAMMER FELLAS

</> Programmer fellas

SUBSCRIBE

Problem

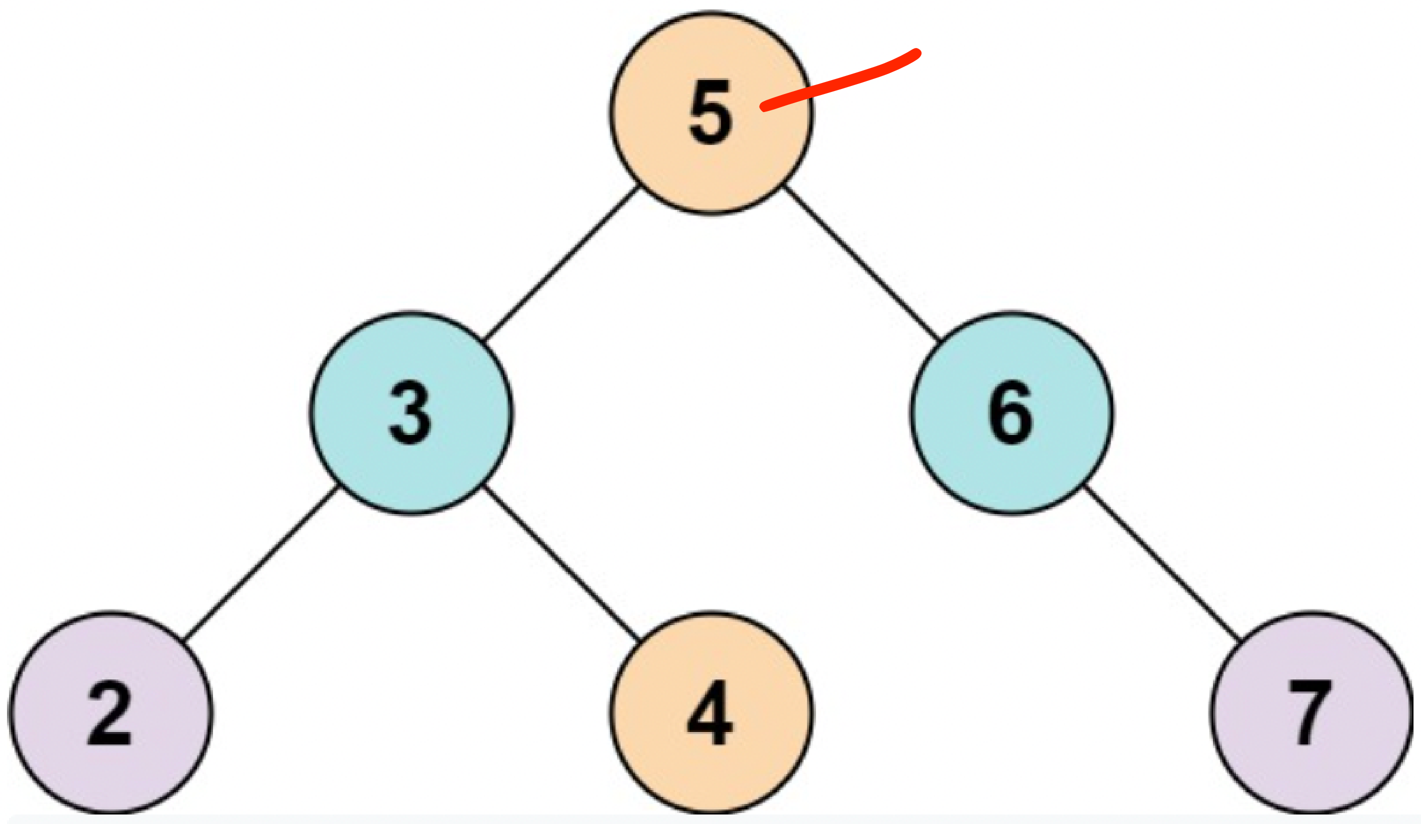


653. Two Sum IV - Input is a BST

Easy 4560 216 Add to List Share

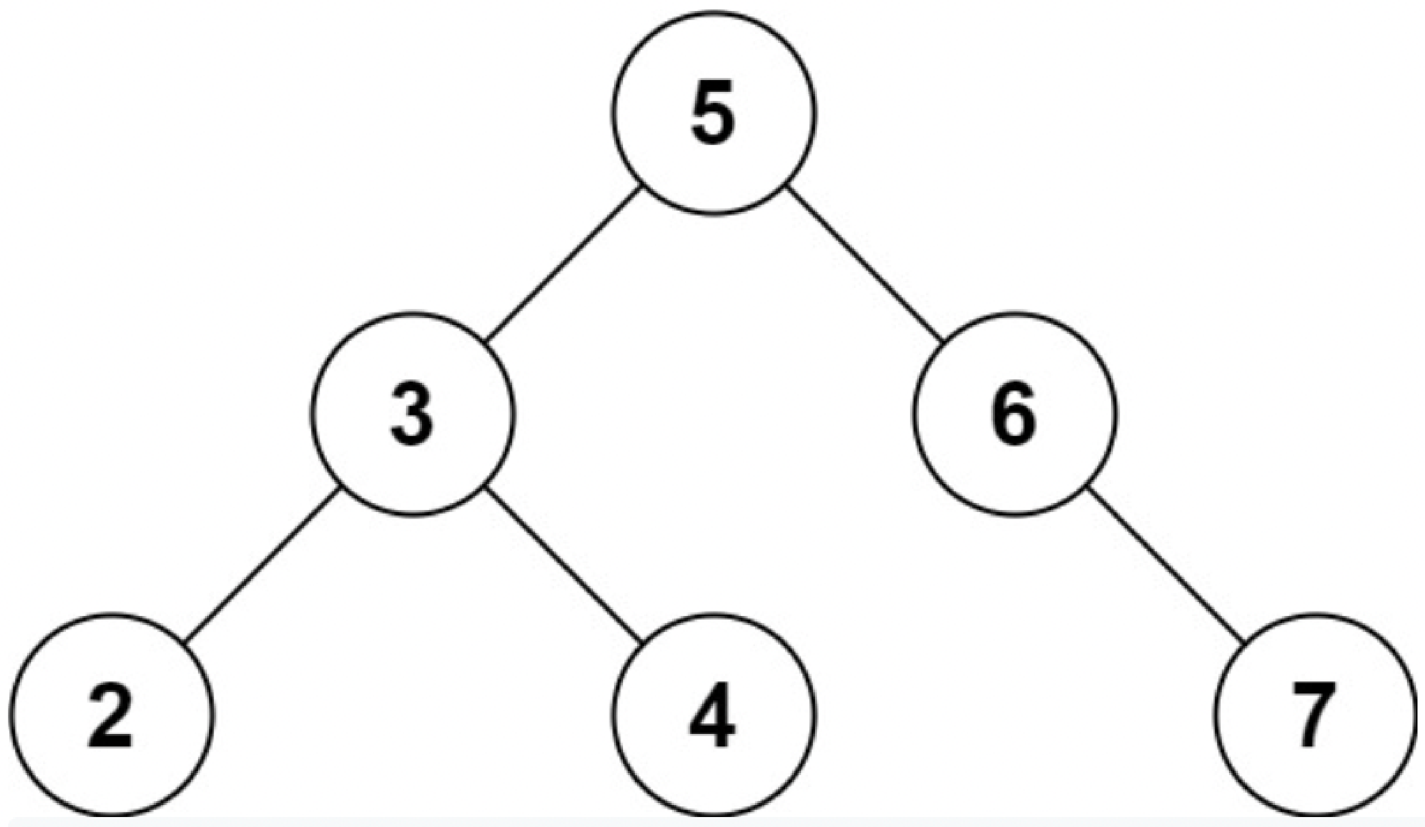
Given the `root` of a Binary Search Tree and a target number `k`, return `true` *if there exist two elements in the BST such that their sum is equal to the given target*.

Example 1:



```
Input: root = [5,3,6,2,4,null,7], k = 9
Output: true
```

Example 2:



```
Input: root = [5,3,6,2,4,null,7], k = 28
Output: false
```

Algorithm

Approach : Using HashSet

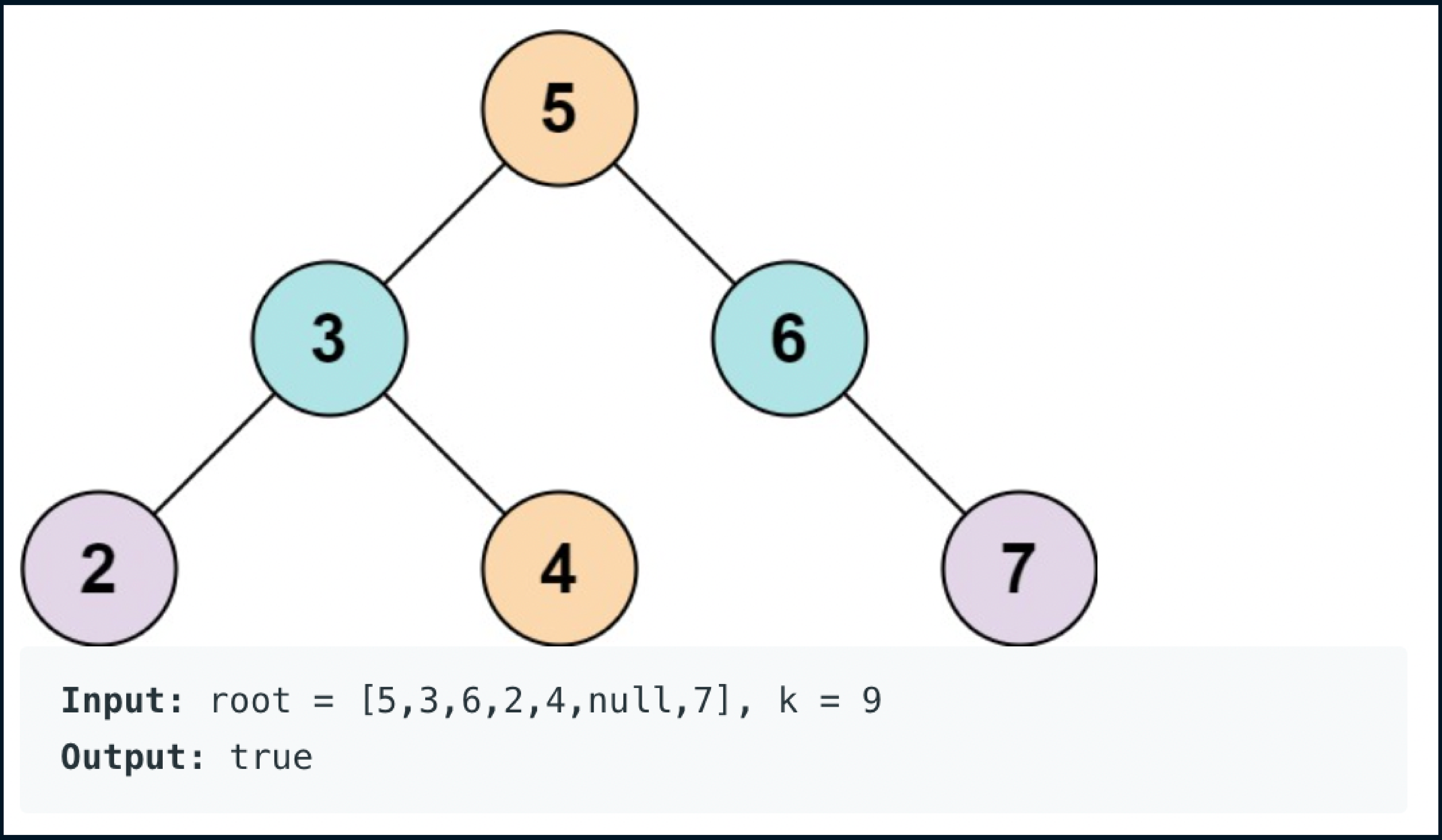
The simplest solution will be to traverse over the whole tree and consider every possible pair of nodes to determine if they can form the required sum kk. But, we can improve the process if we look at a little catch here.

If the sum of two elements x + y$x+y$ equals kk, and we already know that xx exists in the given tree, we only need to check if an element yy exists in the given tree, such that y = k - x$y=k-x$. Based on this simple catch, we can traverse the tree in both the directions(left child and right child) at every step. We keep a track of the elements which have been found so far during the tree traversal, by putting them into a set.

For every current node with a value of pp, we check if k-p$k-p$ already exists in the array. If so, we can conclude that the sum kk can be formed by using the two elements from the given tree. Otherwise, we put this value pp into the set.

If even after the whole tree's traversal, no such element pp can be found, the sum kk can't be formed by using any two elements.

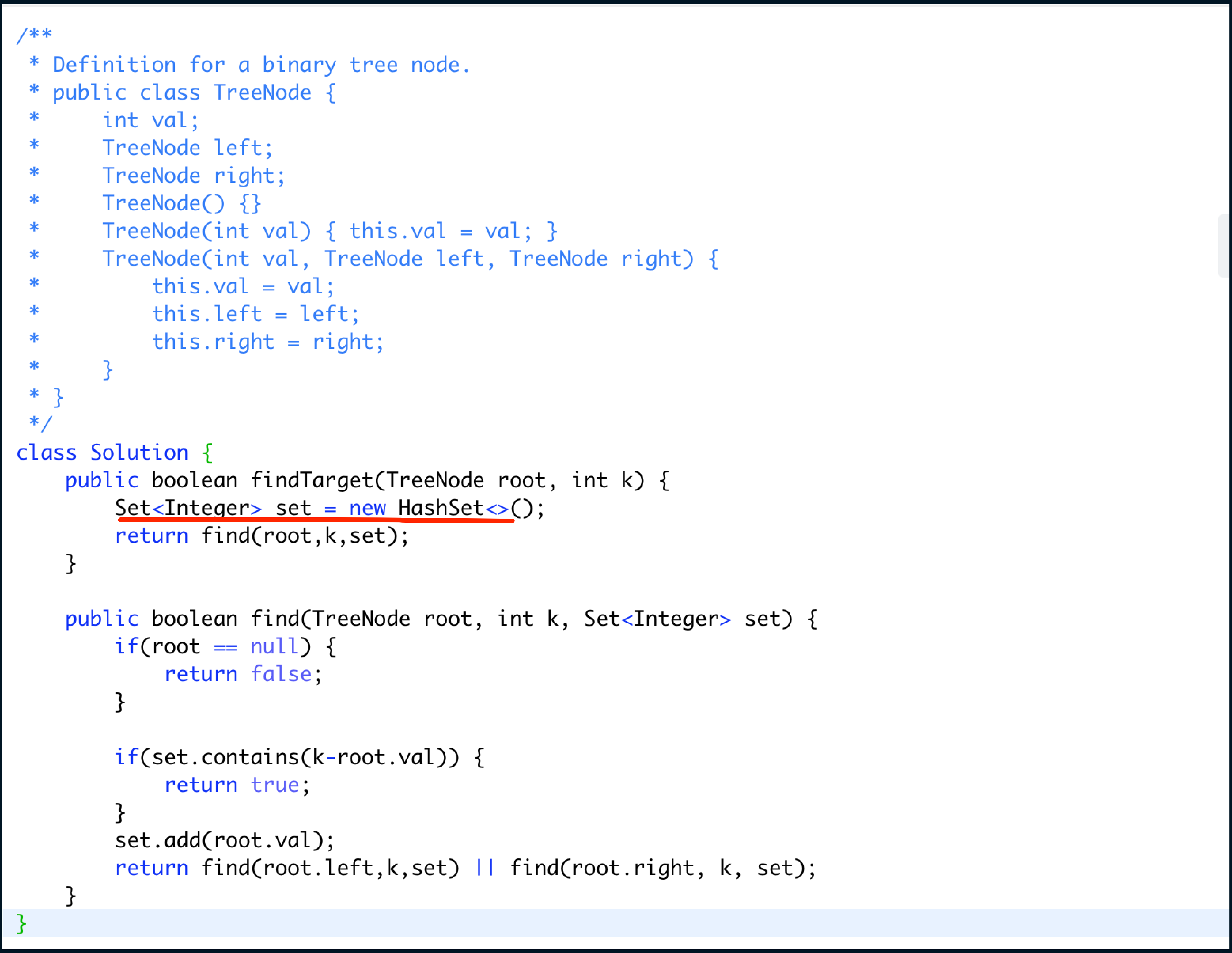
Explanation:



Input: root = [5,3,6,2,4,null,7], k = 9
Output: true

1. **We will create Set and store all the node value.**
2. At starting we will check current node value - target value (root.val - target) in set if its available than return true otherwise we will add into set.
3. We will traverse left and right node till we found the target val.

Solution



```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public boolean findTarget(TreeNode root, int k) {
        Set<Integer> set = new HashSet<>();
        return find(root,k,set);
    }

    public boolean find(TreeNode root, int k, Set<Integer> set) {
        if(root == null) {
            return false;
        }

        if(set.contains(k-root.val)) {
            return true;
        }
        set.add(root.val);
        return find(root.left,k,set) || find(root.right, k, set);
    }
}
```