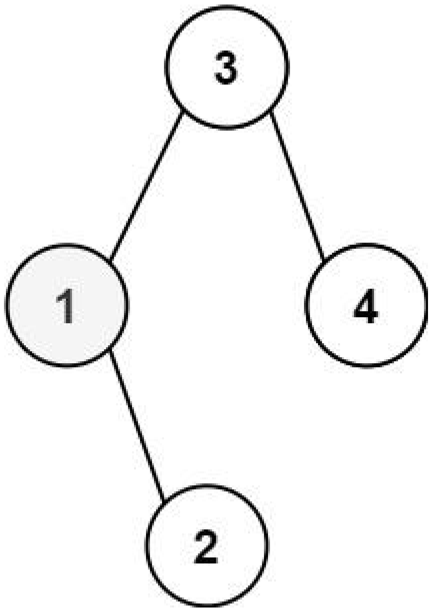
230. Kth Smallest Element in a BST

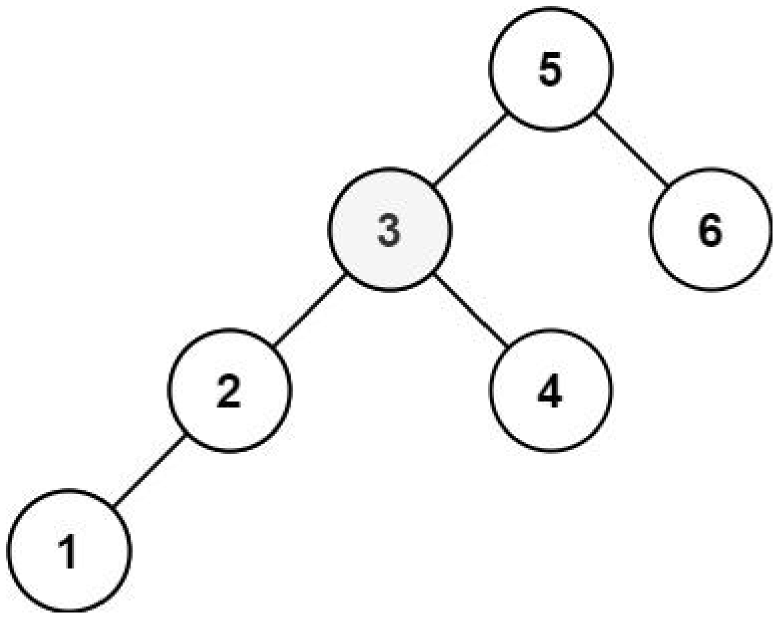
Given the root of a binary search tree, and an integer k, return *the* kth (**1-indexed**) *smallest element in the tree*.

Example 1:



Input: root = [3,1,4,null,2], k = 1
Output: 1

Example 2:



Input: root = [5,3,6,2,4,null,null,1], k = 3
Output: 3

```
 /**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
```

Constraints:

- The number of nodes in the tree is n.
- $1 <= k <= n <= 10^4$
- $0 <= Node.val <= 10^4$

(1) Problem
    (a) Find the kth smallest element of a BST
    (b) Inputs
(2) Plan

```
Root, counter = 0
Find the leftmost node
Counter = 1
Keep going until the counter = k
    If right node
        Counter++
        If children
            If left Node
                Recurs
            If right Node
                Recurs
    Else go to parentNode



Go leftmost
Go right
    Leftmost recursion()
Go up/parent
    Right
        Leftmost recursion()
```

(3) Psuedocode

```
Function kthSmallest(root, k) { // root =3, k=2
    Const [count, value] = helper(root, k, count=0);
    Return value;
}

Function helper(root, k, count=0) {
    Let value = root.val;
    If (root.left) {
        [count, value] = helper(root.left, k, count); // 1-node // 2
    }

    count +=1; // didn't
    If (k === count) return [count, value]; // 2
```

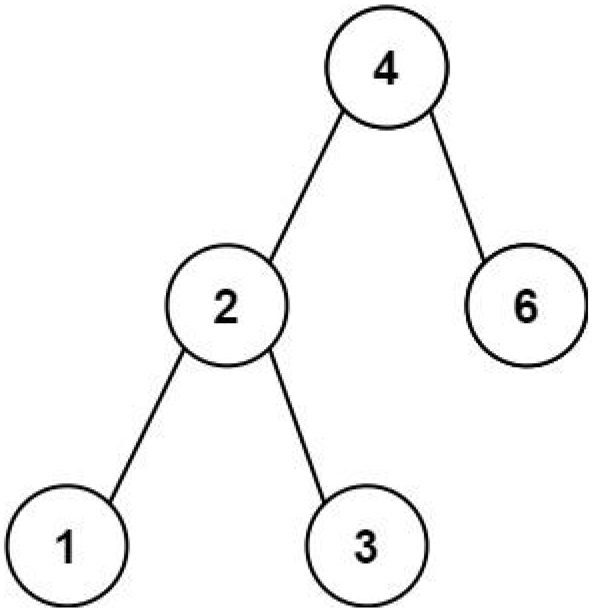
```
        // check if there's a right
        If (node.right) [count, value] = helper(node.right, k, count) // returned 2

        Return [count, value];
}
```

In-order traversal

530. Minimum Absolute Difference in BST

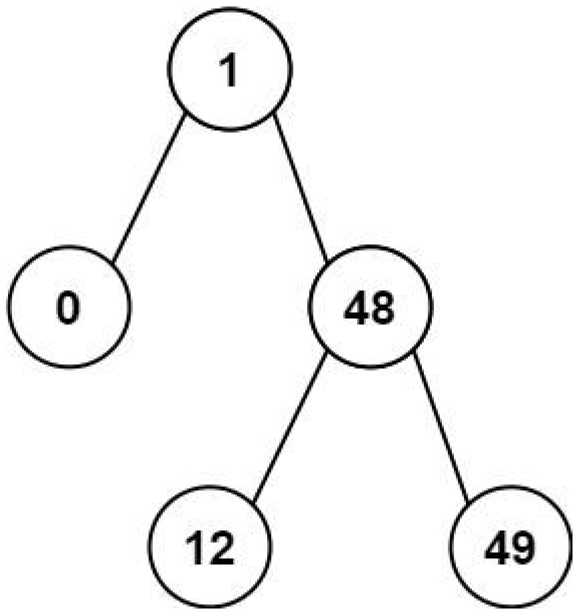
Given the root of a Binary Search Tree (BST), return *the minimum absolute difference between the values of any two different nodes in the tree*.

Example 1:



Input: root = [4,2,6,1,3]
Output: 1

Example 2:



Input: root = [1,0,48,null,null,12,49]
Output: 1

Constraints:

- The number of nodes in the tree is in the range [2, 10^4].
- 0 <= Node.val <= 10^5

Compute the difference between the current node and its children
Keep track of the smallest difference between two nodes
Traverse down the left and right paths

```
function minimumDifference(root, smallestDifference = infinity)
    if root.left
        currDifference = Math.abs(root.val - root.left)
        smallestDifference = smallestDifference > currDifference ? currDifference :
smallestDifference
        return minimumDifference(root.left, smallestDifference)
    if root.right
        currDifference = Math.abs(root.val - root.right)
        smallestDifference = smallestDifference > currDifference ? currDifference :
smallestDifference
        return minimumDifference(root.right, smallestDifference)
    return smallestDifference
```

```
function minimumDifference(root, smallestDifference = Infinity) {
        let leftDifference
        let rightDifference
        if (root.left) {
                const currDifference = Math.abs(root.val - root.left);
                smallestDifference = smallestDifference > currDifference ? currDifference :
smallestDifference;
                leftDifference = minimumDifference(root.left, smallestDifference)

        };
        if (root.right) {
                const currDifference = Math.abs(root.val - root.right);
                smallestDifference = smallestDifference > currDifference ? currDifference :
smallestDifference;
                rightDifference = minimumDifference(root.right, smallestDifference)

        };

}
```