Summary

Time Complexity: O(N), where N is the number of nodes in the tree, since we visit each node only once.
Space Complexity: O(1), since we use constant space (couple of variables) irrespective of number of nodes
Problem Pattern

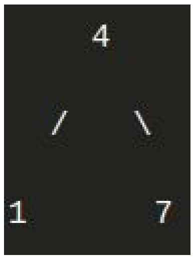
1. What are the properties of a Binary Search Tree that we must verify, given that the tree is surely a Binary Tree?

The requirements of a BST are:
- All the keys of the nodes in the left subtree of a node are supposed to be less than the value of the node.
- All the keys of the nodes in the right subtree of a node are supposed to be greater than the value of the node.

2. How can we check a BST for validity?
3. We can create a recursive function which iterates for all nodes and checks for validity.

Problem Approach

1. We can create a function such as checkBST(curr, minimum, maximum) which takes in the current node, and the range of values allowed for the subtree rooted at the current node.
2. For eg, say the tree looks like



```
     4

   /   \

1       7
```

Then we call the function checkBST(root, -inf, inf), as any value is possible for the root node. Next we recurse for checkBST(root->left, -inf, 3) and checkBST(root->right, 5, inf), because we know that all values less than 4 are allowed for the left subtree, and all values greater than 4 are allowed for the right subtree.

3. Whenever we find a node with a value not within the range we return false. If the current node is within range, we return the value of checkBST(left subtree) AND checkBST(right subtree).

4. The time complexity of the solution is O(N), since we iterate for each node in the tree only once.

Problem Pseudocode

```
isValidBST(self, root):

        return helper(root, -inf, inf)


helper(node, lower, upper):

        if not node:

                return True



        val = node.val

        if val <= lower or val >= upper:

                return False

        if not helper(node.right, val, upper):

                return False

        if not helper(node.left, lower, val):

                return False

        return True
```