```
class Node:
    def __init__(self, data=None):
        self.data = data
        self.right = None
        self.left = None


class BST:
    def __init__(self):
        self.root = None

    def insert(self, data):
        if self.root is None:
            self.root = Node(data)
        else:
            self._insert(data, self.root)

    def _insert(self, data, cur_node):
        if data < cur_node.data:
            if cur_node.left is None:
                cur_node.left = Node(data)
            else:
                self._insert(data, cur_node.left)
        elif data > cur_node.data:
            if cur_node.right is None:
                cur_node.right = Node(data)
            else:
                self._insert(data, cur_node.right)
        else:
            print("Value is already present in tree.")

    def find(self, data):
        if self.root:
            is_found = self._find(data, self.root)
            if is_found:
                return True
            return False
        else:
            return None

    def _find(self, data, cur_node):
        if data > cur_node.data and cur_node.right:
            return self._find(data, cur_node.right)
        elif data < cur_node.data and cur_node.left:
            return self._find(data, cur_node.left)
        if data == cur_node.data:
            return True

    def inorder_print_tree(self):
        if self.root:
            self._inorder_print_tree(self.root)

    def _inorder_print_tree(self, cur_node):
        if cur_node:
            self._inorder_print_tree(cur_node.left)
            print(str(cur_node.data))
            self._inorder_print_tree(cur_node.right)

    def is_bst_satisfied(self):
        if self.root:
            is_satisfied = self._is_bst_satisfied(self.root, self.root.data)

            if is_satisfied is None:
                return True
            return False
        return True

    def _is_bst_satisfied(self, cur_node, data):
```

```
        if cur_node.left:
            if data > cur_node.left.data:
                return self._is_bst_satisfied(cur_node.left, cur_node.left.data)
            else:
                return False
        if cur_node.right:
            if data < cur_node.left.data:
                return self._is_bst_satisfied(cur_node.right, cur_node.right.data)
            else:
                return False


bst = BST()
bst.insert(8)
bst.insert(3)
bst.insert(10)
bst.insert(1)
bst.insert(6)
bst.insert(9)
bst.insert(11)

tree = BST()
tree.root = Node(1)
tree.root.left = Node(2)
tree.root.right = Node(3)

# print(bst.find(4))
# print(bst.inorder_print_tree())
print(bst.is_bst_satisfied())
print(tree.is_bst_satisfied())
```