```
public class SearchForElementInBinarySearchTree {
	static Node search(Node root, int x) {
		if (root == null) {
			return null;
		}
		if (root.data == x) {
			return root;
		} else if (root.data < x) {
			return search(root.right, x);
		} else {
			return search(root.left, x);
		}
	}

	public static void main(String[] args) {
		Node root = new Node(2);
		root.left = new Node(1);
		root.right = new Node(4);
		root.right.left = new Node(3);
		root.right.right = new Node(6);
		System.out.println(search(root, 3).data);
	}
}


public class InsertBST {
	public static Node insert(Node root, int data) {
		if (root == null) {
			root = new Node(data);
			return root;
		}
		if (data < root.data) {
			root.left = insert(root.left, data);
		} else {
			root.right = insert(root.right, data);
		}
		return root;
	}

 }
public class IsBST {
	public boolean isBST(Node root) {
		if (root == null) {
			return true;
		}
		if (root.left != null && root.left.data > root.data) {
			return false;
		}
		if (root.right != null && root.right.data > root.data) {
			return false;
		}
		return isBST(root.left) && isBST(root.right);
	}
}
```

On the below line: -

```
if (root.left != null && root.left.data > root.data)
```

it should be: -

```
if (root.left != null && max(root.left.data) > root.data)
```

So, now the TC will become **O(n^2)**

So, the comparison should not be with children, but it should be with parent.

```
public class IsBSTBetter {
      public boolean isBST(Node root) {
            return isBSTHealper(root, 0, Integer.MAX_VALUE);
      }

      public boolean isBSTHealper(Node root, int min, int max) {
            if (root == null) {
                  return true;
            }
            if (root.data <= min || root.data > max) {
                  return false;
            }
            return isBSTHealper(root.left, min, root.data) &&
                        isBSTHealper(root.right, root.data, max);

      }
}
```