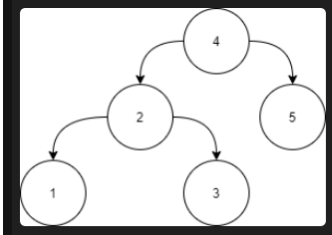
Validate BST

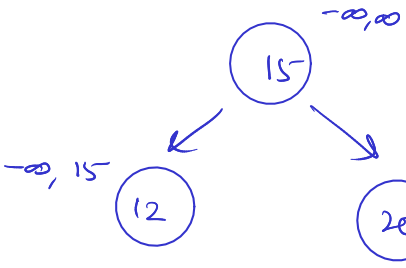
You have been given a binary tree of integers with N number of nodes. Your task is to check if that input tree is a BST (Binary Search Tree) or not.
A binary search tree (BST) is a binary tree data structure which has the following properties.

- The left subtree of a node contains only nodes with data less than the node's data.
- The right subtree of a node contains only nodes with data greater than the node's data.
- Both the left and right subtrees must also be binary search trees.

Example :



→ Approach 1:-



↓
Range in which this particular node can lie.

```
bool solve(BinaryTreeNode<int> *root, int lower, int higher) {
  if (root == NULL)
    return true;
  if (root->data < higher && root->data > lower) {

    bool left = solve(root->left, lower, root->data);
    bool right = solve(root->right, root->data, higher);
    return left && right;
  } else
    return false;
}

bool validateBST(BinaryTreeNode<int> *root) {
  int lower = INT_MIN, higher = INT_MAX;

  solve(root, lower, higher);
}
```

Time Complexity: O(N)

Space Complexity: O(N)

→ Approach 2:-

↳ Inorder traversal of BST is sorted.

↳ Now maintain a current & prev pointer & if the prev element is greater then it's not a BST.

```
bool helper(BinaryTreeNode<int>* root, BinaryTreeNode<int>* &prev)
{
    if (root != NULL)
    {
        if (!helper(root -> left, prev))
        {
            return false;
        }
        if (prev != NULL && root -> data <= prev -> data)
        {
            return false;
        }
        prev = root;
        return helper(root -> right, prev);
    }
    return true;
}
bool validateBST(BinaryTreeNode<int>* root)
{
    BinaryTreeNode<int>* prev = NULL;

    return helper(root, prev);
}
```

Time Complexity: O(N)

Space Complexity: O(N)