Our Solution(s) Run Code

Solution 1

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <climits>
using namespace std;

class BST {
public:
  int value;
  BST *left;
  BST *right;

  BST(int val);
  BST &insert(int val);
};

bool validateBst(BST *tree);
bool validateBstHelper(BST *tree, int minValue, int maxValue);

// O(n) time | O(d) space
bool validateBst(BST *tree) {
  return validateBstHelper(tree, INT_MIN, INT_MAX);
}

bool validateBstHelper(BST *tree, int minValue, int maxValue) {
  if (tree->value < minValue || tree->value >= maxValue) {
    return false;
  }
  if (tree->left != NULL &&
      !validateBstHelper(tree->left, minValue, tree->value)) {
    return false;
  }
  if (tree->right != NULL &&
      !validateBstHelper(tree->right, tree->value, maxValue)) {
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
class BST {
public:
  int value;
  BST *left;
  BST *right;

  BST(int val);
  BST &insert(int val);
};

// O(n) time | O(d) space
bool validateBst(BST *tree) {
  // Write your code here.
  return false;
}

```

Our Tests

Custom Output Submit Code

```
    value = val,
    left = Nil,
    right = Nil,
}

// Average: O(log(n)) time | O(log(n)) space
// Worst: O(n) time | O(n) space
fn (mut t BST) insert(val int) {
  if val < value {
    if left == Nil {
      mut newBST := new_BST(val)
      left = newBST
    } else {
      left.insert(val)
    }
  } else {
    if right == Nil {
      mut newBST := new_BST(val)
      right = newBST
    } else {
      right.insert(val)
```

Run or submit code when you're ready.