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

using System;
using System.Collections.Generic;

public class Program {
  // O(n^2) time | O(d) space - where n is the number of
  // nodes in each array, respectively, and d is the depth
  // of the BST that they represent
  public static bool SameBsts(List<int> arrayOne, List<int> arrayTwo) {
    return areSameBsts(arrayOne, arrayTwo, 0, 0, Int32.MinValue, Int32.MaxValue);
  }

  public static bool areSameBsts(List<int> arrayOne, List<int> arrayTwo, int rootIdxOne,
    int rootIdxTwo, int minVal, int maxVal) {
    if (rootIdxOne == -1 || rootIdxTwo == -1) return rootIdxOne == rootIdxTwo;

    if (arrayOne[rootIdxOne] != arrayTwo[rootIdxTwo]) return false;

    int leftRootIdxOne = getIdxOfFirstSmaller(arrayOne, rootIdxOne, minVal);
    int leftRootIdxTwo = getIdxOfFirstSmaller(arrayTwo, rootIdxTwo, minVal);
    int rightRootIdxOne = getIdxOfFirstBiggerOrEqual(arrayOne, rootIdxOne, maxVal);
    int rightRootIdxTwo = getIdxOfFirstBiggerOrEqual(arrayTwo, rootIdxTwo, maxVal);

    int currentValue = arrayOne[rootIdxOne];
    bool leftAreSame = areSameBsts(arrayOne, arrayTwo, leftRootIdxOne, leftRootIdxTwo,
        minVal, currentValue);
    bool rightAreSame = areSameBsts(arrayOne, arrayTwo, rightRootIdxOne,
        rightRootIdxTwo, currentValue, maxVal);

    return leftAreSame && rightAreSame;
  }

  public static int getIdxOfFirstSmaller(List<int> array, int startingIdx, int minVal) {
    // Find the index of the first smaller value after the startingIdx.
    // Make sure that this value is greater than or equal to the minVal,
    // which is the value of the previous parent node in the BST. If it
    // isn't, then that value is located in the left subtree of the
    // previous parent node.
    for (int i = startingIdx + 1; i < array.Count; i++) {
      if (array[i] < array[startingIdx] && array[i] >= minVal) return i;
    }
    return -1;
  }

  public static int getIdxOfFirstBiggerOrEqual(List<int> array, int startingIdx, int maxVal) {
    // Find the index of the first bigger/equal value after the startingIdx.
    // Make sure that this value is smaller than maxVal, which is the value
    // of the previous parent node in the BST. If it isn't, then that value
    // is located in the right subtree of the previous parent node.
    for (int i = startingIdx + 1; i < array.Count; i++) {
      if (array[i] >= array[startingIdx] && array[i] < maxVal) return i;
    }
    return -1;
  }
}

```