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

import java.util.*;

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 boolean sameBsts(List<Integer> arrayOne, List<Integer> arrayTwo) {
    return areSameBsts(arrayOne, arrayTwo, 0, 0, Integer.MIN_VALUE, Integer.MAX_VALUE);
  }

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

    if (arrayOne.get(rootIdxOne).intValue() != arrayTwo.get(rootIdxTwo).intValue()) 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.get(rootIdxOne);
    boolean leftAreSame =
        areSameBsts(arrayOne, arrayTwo, leftRootIdxOne, leftRootIdxTwo, minVal, currentValue);
    boolean rightAreSame =
        areSameBsts(arrayOne, arrayTwo, rightRootIdxOne, rightRootIdxTwo, currentValue, maxVal);

    return leftAreSame && rightAreSame;
  }

  public static int getIdxOfFirstSmaller(List<Integer> 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.size(); i++) {
      if (array.get(i).intValue() < array.get(startingIdx).intValue()
          && array.get(i).intValue() >= minVal) return i;
    }
    return -1;
  }

  public static int getIdxOfFirstBiggerOrEqual(List<Integer> 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.size(); i++) {
      if (array.get(i).intValue() >= array.get(startingIdx).intValue()
          && array.get(i).intValue() < maxVal) return i;
    }
    return -1;
  }
}

```