

PromptScratchpadOur Solution(s)Video Explanation

Run Code

Solution 1Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 import java.util.*;
4
5 class Program {
6     // O(n^2) time | O(d) space - where n is the number of
7     // nodes in each array, respectively, and d is the depth
8     // of the BST that they represent
9     public static boolean sameBsts(List<Integer> arrayOne, List<Integer> arrayTwo) {
10         return areSameBsts(arrayOne, arrayTwo, 0, 0, Integer.MIN_VALUE, Integer.MAX_VALUE);
11     }
12
13     public static boolean areSameBsts(
14         List<Integer> arrayOne,
15         List<Integer> arrayTwo,
16         int rootIdxOne,
17         int rootIdxTwo,
18         int minVal,
19         int maxVal) {
20         if (rootIdxOne == -1 || rootIdxTwo == -1) return rootIdxOne == rootIdxTwo;
21
22         if (arrayOne.get(rootIdxOne).intValue() != arrayTwo.get(rootIdxTwo).intValue()) return false;
23
24         int leftRootIdxOne = getIdxOffFirstSmaller(arrayOne, rootIdxOne, minVal);
25         int leftRootIdxTwo = getIdxOffFirstSmaller(arrayTwo, rootIdxTwo, minVal);
26         int rightRootIdxOne = getIdxOffFirstBiggerOrEqual(arrayOne, rootIdxOne, maxVal);
27         int rightRootIdxTwo = getIdxOffFirstBiggerOrEqual(arrayTwo, rootIdxTwo, maxVal);
28
29         int currentValue = arrayOne.get(rootIdxOne);
30         boolean leftAreSame =
31             areSameBsts(arrayOne, arrayTwo, leftRootIdxOne, leftRootIdxTwo, minVal, currentValue);
32         boolean rightAreSame =
33             areSameBsts(arrayOne, arrayTwo, rightRootIdxOne, rightRootIdxTwo, currentValue, maxVal);
34
35         return leftAreSame && rightAreSame;
36     }
37
38     public static int getIdxOffFirstSmaller(List<Integer> array, int startingIdx, int minVal) {
39         // Find the index of the first smaller value after the startingIdx.
40         // Make sure that this value is greater than or equal to the minVal,
41         // which is the value of the previous parent node in the BST. If it
42         // isn't, then that value is located in the left subtree of the
43         // previous parent node.
44         for (int i = startingIdx + 1; i < array.size(); i++) {
45             if (array.get(i).intValue() < array.get(startingIdx).intValue()
46                 && array.get(i).intValue() >= minVal) return i;
47         }
48         return -1;
49     }
50
51     public static int getIdxOffFirstBiggerOrEqual(List<Integer> array, int startingIdx, int maxVal) {
52         // Find the index of the first bigger/equal value after the startingIdx.
53         // Make sure that this value is smaller than maxVal, which is the value
54         // of the previous parent node in the BST. If it isn't, then that value
55         // is located in the right subtree of the previous parent node.
56         for (int i = startingIdx + 1; i < array.size(); i++) {
57             if (array.get(i).intValue() >= array.get(startingIdx).intValue()
58                 && array.get(i).intValue() < maxVal) return i;
59         }
60         return -1;
61     }
62 }
63
```