Our Solution(s) Run Code

Solution 1 Solution 2

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

package main

import "math"

// 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
func SameBSTs(arrayOne, arrayTwo []int) bool {
  return areSameBSTs(arrayOne, arrayTwo, 0, 0, math.MinInt32, math.Max
}

func areSameBSTs(arrayOne, arrayTwo []int, rootIdxOne, rootIdxTwo int,
  if rootIdxOne == -1 || rootIdxTwo == -1 {
    return rootIdxOne == rootIdxTwo
  }

  if arrayOne[rootIdxOne] != arrayTwo[rootIdxTwo] {
    return false
  }

  leftRootIdxOne := getIdxOfFirstSmaller(arrayOne, rootIdxOne, minVal)
  leftRootIdxTwo := getIdxOfFirstSmaller(arrayTwo, rootIdxTwo, minVal)
  rightRootIdxOne := getIdxOfFirstBiggerOrEqual(arrayOne, rootIdxOne,
  rightRootIdxTwo := getIdxOfFirstBiggerOrEqual(arrayTwo, rootIdxTwo,

  currentValue := arrayOne[rootIdxOne]
  leftAreSame := areSameBSTs(arrayOne, arrayTwo, leftRootIdxOne, leftR
  rightAreSame := areSameBSTs(arrayOne, arrayTwo, rightRootIdxOne, rig

  return leftAreSame && rightAreSame
}
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

func SameBSTs(arrayOne, arrayTwo []int) bool {
  // Write your code here.
  return false
}

```

Our Tests

Custom Output Submit Code

```
}

func (s *TestSuite) TestCase23() *TestCase {
    arrayOne := []int{3, 2, 1, 4, 5, 6, 7}
    arrayTwo := []int{3, 2, 1, 4, 5, 6, 7}
    return &TestCase{1, SameMirrorArray(arrayOne, arrayTwo), true}
}

func (s *TestSuite) TestCase24() *TestCase {
    arrayOne := []int{7, 6, 5, 4, 5, 6, 7}
    arrayTwo := []int{7, 6, 5, 4, 5, 6, 7}
    return &TestCase{2, SameMirrorArray(arrayOne, arrayTwo), true}
}

func (s *TestSuite) TestCase25() *TestCase {
    arrayOne := []int{90, 35, 6, 12, 90, 90, 5, 2}
```

Run or submit code when you're ready.