AlgoExpert   Quad Layout   Go   12px   Sublime   Monokai   00:00:00

Prompt   Scratchpad   Our Solution(s)   Video Explanation   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.MaxInt32)
}

func areSameBSTs(arrayOne, arrayTwo []int, rootIdxOne, rootIdxTwo int, minVal, maxVal int) bool {
  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, maxVal)
  rightRootIdxTwo := getIdxOfFirstBiggerOrEqual(arrayTwo, rootIdxTwo, maxVal)

  currentValue := arrayOne[rootIdxOne]
  leftAreSame := areSameBSTs(arrayOne, arrayTwo, leftRootIdxOne, leftRootIdxTwo, minVal, currentValue)
  rightAreSame := areSameBSTs(arrayOne, arrayTwo, rightRootIdxOne, rightRootIdxTwo, currentValue, maxVal)

  return leftAreSame && rightAreSame
}

func getIdxOfFirstSmaller(array []int, startingIdx, minVal int) int {
  // 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 i := startingIdx + 1; i < len(array); i++ {
    if array[i] < array[startingIdx] && array[i] >= minVal {
      return i
    }
  }
  return -1
}

func getIdxOfFirstBiggerOrEqual(array []int, startingIdx, maxVal int) int {
  // 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 i := startingIdx + 1; i < len(array); i++ {
    if array[i] >= array[startingIdx] && array[i] < maxVal {
      return i
    }
  }
  return -1
}
```