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

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1 Solution 2

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

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
    func sameBSTs(_ arrayOne: [Int], _ arrayTwo: [Int]) -> Bool {
        return areSameBSTs(arrayOne, arrayTwo, 0, 0, Int.min, Int.max)
    }

    func areSameBSTs(_ arrayOne: [Int], _ arrayTwo: [Int], _ rootIdxOne: Int, _ rootIdxTwo: Int, _ minVal: Int, _ maxVal: Int) -> Bool {
        if rootIdxOne == -1 || rootIdxTwo == -1 {
            return rootIdxOne == rootIdxTwo
        }

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

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

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

        return leftAreSame && rightAreSame
    }

    func getIdxOfFirstSmaller(_ array: [Int], _ startingIdx: Int, _ 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 in (startingIdx + 1) ..< array.count {
            if array[i] < array[startingIdx], array[i] >= minVal {
                return i
            }
        }
        return -1
    }

    func getIdxOfFirstBiggerOrEqual(_ array: [Int], _ startingIdx: Int, _ 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 in (startingIdx + 1) ..< array.count {
            if array[i] >= array[startingIdx], array[i] < maxVal {
                return i
            }
        }
        return -1
    }
}
```