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

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1 Solution 2

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

// 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
function sameBsts(arrayOne, arrayTwo) {
  return areSameBsts(arrayOne, arrayTwo, 0, 0, -Infinity, Infinity);
}

function areSameBsts(arrayOne, arrayTwo, rootIdxOne, rootIdxTwo, minVal, maxVal) {
  if (rootIdxOne === -1 || rootIdxTwo === -1) return rootIdxOne === rootIdxTwo;

  if (arrayOne[rootIdxOne] !== arrayTwo[rootIdxTwo]) return false;

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

  const currentValue = arrayOne[rootIdxOne];
  const leftAreSame = areSameBsts(arrayOne, arrayTwo, leftRootIdxOne, leftRootIdxTwo, minVal, currentValue);
  const rightAreSame = areSameBsts(arrayOne, arrayTwo, rightRootIdxOne, rightRootIdxTwo, currentValue, maxVal);

  return leftAreSame && rightAreSame;
}

function getIdxOfFirstSmaller(array, startingIdx, 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 (let i = startingIdx + 1; i < array.length; i++) {
    if (array[i] < array[startingIdx] && array[i] >= minVal) return i;
  }
  return -1;
}

function getIdxOfFirstBiggerOrEqual(array, startingIdx, 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 (let i = startingIdx + 1; i < array.length; i++) {
    if (array[i] >= array[startingIdx] && array[i] < maxVal) return i;
  }
  return -1;
}

exports.sameBsts = sameBsts;
```