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

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1 Solution 2

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

import java.util.*;

class Program {
  // O(nlogn) time | O(n) space
  public static List<Integer> longestIncreasingSubsequence(int[] array) {
    int[] sequences = new int[array.length];
    int[] indices = new int[array.length + 1];
    Arrays.fill(indices, Integer.MIN_VALUE);
    int length = 0;
    for (int i = 0; i < array.length; i++) {
      int num = array[i];
      int newLength = binarySearch(1, length, indices, array, num);
      sequences[i] = indices[newLength - 1];
      indices[newLength] = i;
      length = Math.max(length, newLength);
    }
    return buildSequence(array, sequences, indices[length]);
  }

  public static int binarySearch(int startIdx, int endIdx, int[] indices, int[] array, int num) {
    if (startIdx > endIdx) {
      return startIdx;
    }
    int middleIdx = (startIdx + endIdx) / 2;
    if (array[indices[middleIdx]] < num) {
      startIdx = middleIdx + 1;
    } else {
      endIdx = middleIdx - 1;
    }
    return binarySearch(startIdx, endIdx, indices, array, num);
  }

  public static List<Integer> buildSequence(int[] array, int[] sequences, int currentIdx) {
    List<Integer> sequence = new ArrayList<Integer>();
    while (currentIdx != Integer.MIN_VALUE) {
      sequence.add(0, array[currentIdx]);
      currentIdx = sequences[currentIdx];
    }
    return sequence;
  }
}

```