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

using System;
using System.Collections.Generic;

public class Program {
  // O(nlogn) time | O(n) space
  public static List<int> LongestIncreasingSubsequence(int[] array) {
    int[] sequences = new int[array.Length];
    int[] indices = new int[array.Length + 1];
    Array.Fill(indices, Int32.MinValue);
    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<int> buildSequence(int[] array, int[] sequences, int currentIdx) {
    List<int> sequence = new List<int>();
    while (currentIdx != Int32.MinValue) {
      sequence.Insert(0, array[currentIdx]);
      currentIdx = sequences[currentIdx];
    }
    return sequence;
  }
}

```