Solution 1

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

using System;
using System.Collections.Generic;

public class Program {
  // O(n^2) time | O(n) space
  public static List<List<int> > MaxSumIncreasingSubsequence(int[] array) {
    int[] sequences = new int[array.Length];
    Array.Fill(sequences, Int32.MinValue);
    int[] sums = (int[]) array.Clone();
    int maxSumIdx = 0;
    for (int i = 0; i < array.Length; i++) {
      int currentNum = array[i];
      for (int j = 0; j < i; j++) {
        int otherNum = array[j];
        if (otherNum < currentNum && sums[j] + currentNum >= sums[i]) {
          sums[i] = sums[j] + currentNum;
          sequences[i] = j;
        }
      }
      if (sums[i] >= sums[maxSumIdx]) {
        maxSumIdx = i;
      }
    }
    return buildSequence(array, sequences, maxSumIdx, sums[maxSumIdx]);
  }

  public static List<List<int> > buildSequence(int[] array, int[] sequences, int currentIdx,
    int sums) {
    List<List<int> > sequence = new List<List<int> >();
    sequence.Add(new List<int>());
    sequence.Add(new List<int>());
    sequence[0].Add(sums);
    while (currentIdx != Int32.MinValue) {
      sequence[1].Insert(0, array[currentIdx]);
      currentIdx = sequences[currentIdx];
    }
    return sequence;
  }
}
```