Solution 1

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

import java.util.*;

class Program {
  // O(n^2) time | O(n) space
  public static List<List<Integer>> maxSumIncreasingSubsequence(int[] array) {
    int[] sequences = new int[array.length];
    Arrays.fill(sequences, Integer.MIN_VALUE);
    int[] sums = 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<Integer>> buildSequence(
      int[] array, int[] sequences, int currentIdx, int sums) {
    List<List<Integer>> sequence = new ArrayList<List<Integer>>();
    sequence.add(new ArrayList<Integer>());
    sequence.add(new ArrayList<Integer>());
    sequence.get(0).add(sums);
    while (currentIdx != Integer.MIN_VALUE) {
      sequence.get(1).add(0, array[currentIdx]);
      currentIdx = sequences[currentIdx];
    }
    return sequence;
  }
}

```