Solution 1

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

#include <vector>
#include <climits>
using namespace std;

vector<vector<int>> buildSequence(vector<int> array, vector<int> sequences,
                                  int currentIdx, int sum);

// O(n^2) time | O(n) space
vector<vector<int>> maxSumIncreasingSubsequence(vector<int> array) {
  vector<int> sequences(array.size(), INT_MIN);
  vector<int> sums = array;
  int maxSumIdx = 0;
  for (int i = 0; i < array.size(); 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]);
}

vector<vector<int>> buildSequence(vector<int> array, vector<int> sequences,
                                  int currentIdx, int sum) {
  vector<vector<int>> sequence = {{}, {}};
  sequence[0].push_back(sum);
  while (currentIdx != INT_MIN) {
    sequence[1].insert(sequence[1].begin(), array[currentIdx]);
    currentIdx = sequences[currentIdx];
  }
  return sequence;
}

```