Algorithms, Spring 2012-13, Homework 4
due Friday, 5 April 2013, 14:00

Problem 1

Given is a sequence of integers $a_1, a_2, \ldots, a_n$. Give an $O(n^2)$ algorithm that finds the longest possible subsequence of $a_1, a_2, \ldots, a_n$ that alternates even and odd integers.

For example, for sequence $1, 5, 6, 2, 4, 7$, one of such longest subsequences is $5, 2, 7$.

Problem 2

Given is a sequence of integers $a_1, a_2, \ldots, a_n$. Give an $O(n^2)$ algorithm that finds the number of all subsequences of $a_1, a_2, \ldots, a_n$ that alternate even and odd integers.

For example, for sequence $1, 5, 6, 2, 4, 7$, all such subsequences are

- of length one: 1, and 5, and 6, and 2, and 4, and 7 (a total of 6 subsequences of length one),
- of length two: $1, 6$, and $1, 2$, and $1, 4$, and $5, 6$, and $5, 2$, and $5, 4$, and $6, 7$, and $2, 7$, and $4, 7$ (a total of 9 subsequences of length two), and
- of length three: $1, 6, 7$, and $1, 2, 7$, and $1, 4, 7$, and $5, 6, 7$, and $5, 2, 7$, and $5, 4, 7$ (a total of 6 subsequences of length three).

Hence, the total number of such subsequences is 21.

Problem 3

You've been in awe of Mt Srđ and you would like to build switchback trails up several other Croatian mountains. Digging up the stones and creating the path would be fun but you do not have the time to do that. Fortunately, a local company called Zollstock (ok, maybe not so local after all...) heard of your plans and offered to help. Each of their products, commonly referred to as a zollstock, consists of straight pieces of a firm material held together by joints; every joint connects exactly two pieces and all pieces are connected to form a single zig-zag object. Most customers want their zollstocks to consist of equal length pieces, typically used for measuring distance (called "folding ruler" in English). As the company has several huge zollstocks with pieces of varying length that they cannot sell, they offered them to you for free. It turns out that the material is as wide as a path and it forms a good walking surface. So, you decided to build each hiking trail from one of the zollstocks.

The last thing that remains is to make sure that you comply with the local requirements:

- Whenever the switchback trail turns, it has to turn by 165 degrees. This means that when using the zollstock, the joint is either going to go straight (0 degree turn) or turn back by 165 degrees.

- For each mountain, there is a specified width that indicates how far to the left or right can the trail get from the starting position. This distance conveniently already accounts for 15 degree slant.

  For example, suppose that the trail consists of 50 meters going left, then 165 degree turn, then 20 meters going right, then 0 degree turn, then 10 meters going right, then 165 degree turn, and then 100 meters going left. Then, overall the trail goes 120 meters to the left of the starting position and 0 meters to the right of the starting position. If the specified width is a maximum of 80 meters, the mentioned trail does not comply. However, if the trail goes 50 meters to the left, continues 20 and then 10 meters to the left, then turns and goes 100 meters to the right, it would have gone an overall 80 meters to the left and 20 meters to the right of the starting position, complying with the maximum of 80 meters to either side.

Given the number of zollstocks and mountains you volunteered to deal with, you decided to write the following helper program. The input consists of positive integers $a_1, a_2, \dots, a_n$ specifying a single zollstock, and a positive integer W, the maximum possible width to either side. The i-th piece is of length a_i for $i \in \{1, 2, \dots, n\}$, and the i-th joint connects the i-th and the $(i+1)$-st piece, for $i \in \{1, 2, \dots, n-1\}$.

Your program first outputs whether it is possible to stay within the required width range. If yes, it then outputs a sequence of left and right directions, the i-th direction refers to the i-th piece of the zollstock, defining a switchback trail that complies with the width requirement. Make sure that the running time of your code is $O(nW)$.

Remark: Do not worry about the height of the mountain. It is easy to check whether a given zollstock is an appropriate length for a given mountain height and this has been already done before you use your program.

Note: Heart of the solution

For some of the problems on this homework (and other problems in the future) you will need to use dynamic programming. To explain how your algorithm works, describe the "heart of the algorithm" (you do not need to include any other explanation). Recall that the heart of the algorithm consists of three parts:

- *What:* A precise verbal description of the meaning of your dynamic programming array, see the slides for examples.
- *How:* A mathematical formula that describes how to compute the value of each cell of the array (using previously computed array values).
- *Where:* to find the answer. I.e., what is the return value of the algorithm.