Tutorium #12

03/02/2023

A **wiggle sequence** is a sequence where the differences between successive numbers strictly alternate between positive and negative. The first difference (if one exists) may be either positive or negative. A sequence with one element and a sequence with two non-equal elements are trivially wiggle sequences.

- For example, `[1, 7, 4, 9, 2, 5]` is a **wiggle sequence** because the differences `(6, -3, 5, -7, 3)` alternate between positive and negative.
- In contrast, `[1, 4, 7, 2, 5]` and `[1, 7, 4, 5, 5]` are not wiggle sequences. The first is not because its first two differences are positive, and the second is not because its last difference is zero.

A **subsequence** is obtained by deleting some elements (possibly zero) from the original sequence, leaving the remaining elements in their original order.

Given an integer array `nums`, return *the length of the longest **wiggle subsequence** of* `nums`.

A company produces custom build curtain rods (Gardinenstangen). Therefore, they have poles of length m which can be cut according to the wishes of their customers. Each production process consists of n orders, where each order i has a corresponding length $0 < x_i \leq m$. These are stored in an array x, ordered by the time, when the customer has stated the order. The production works according to the following algorithm.

procedure greedyPole(x : Array of $\mathbb{R}_{>0}$, $n : \mathbb{N}$, $m : \mathbb{N}$)
    $u := 0$ *//number of used poles*
    $\mathcal{R} := \emptyset$ *//here we store remaining pieces of poles*
    for $i = 0, .., n-1$ **do**
        if $\exists \ell \in \mathcal{R}$ with $\ell \geq x[i]$ **then**
            $\mathcal{R} = \mathcal{R} \setminus \{\ell\}$ *//cut from remaining piece*
            if $\ell - x[i] > 0$ **then**
                $\mathcal{R} = \mathcal{R} \cup \{\ell - x[i]\}$
        else
            $u++$ *//take a new pole*
            $\mathcal{R} = \mathcal{R} \cup \{m - x[i]\}$

To save money the goal is to minimize the used poles in the production process. The company is interested in how good the algorithm is compared to the optimal way of cutting the poles.

Problem 1 (4 points)

Show how the problem can be solved in $O(n \log n)$.

Solution:
The problem can be solved analogue to the bin packing problem. For this some approximation algorithms exist which run in $\mathrm{O}(n \log n)$.
For example the *BestFitDecreasing* Algorithm or even the *FirstFitDecreasing* algorithm solve the problem. Both are in $\mathrm{O}(n \log n)$. The important part in both heuristics is to sort the elements x_i in descending order in advance.

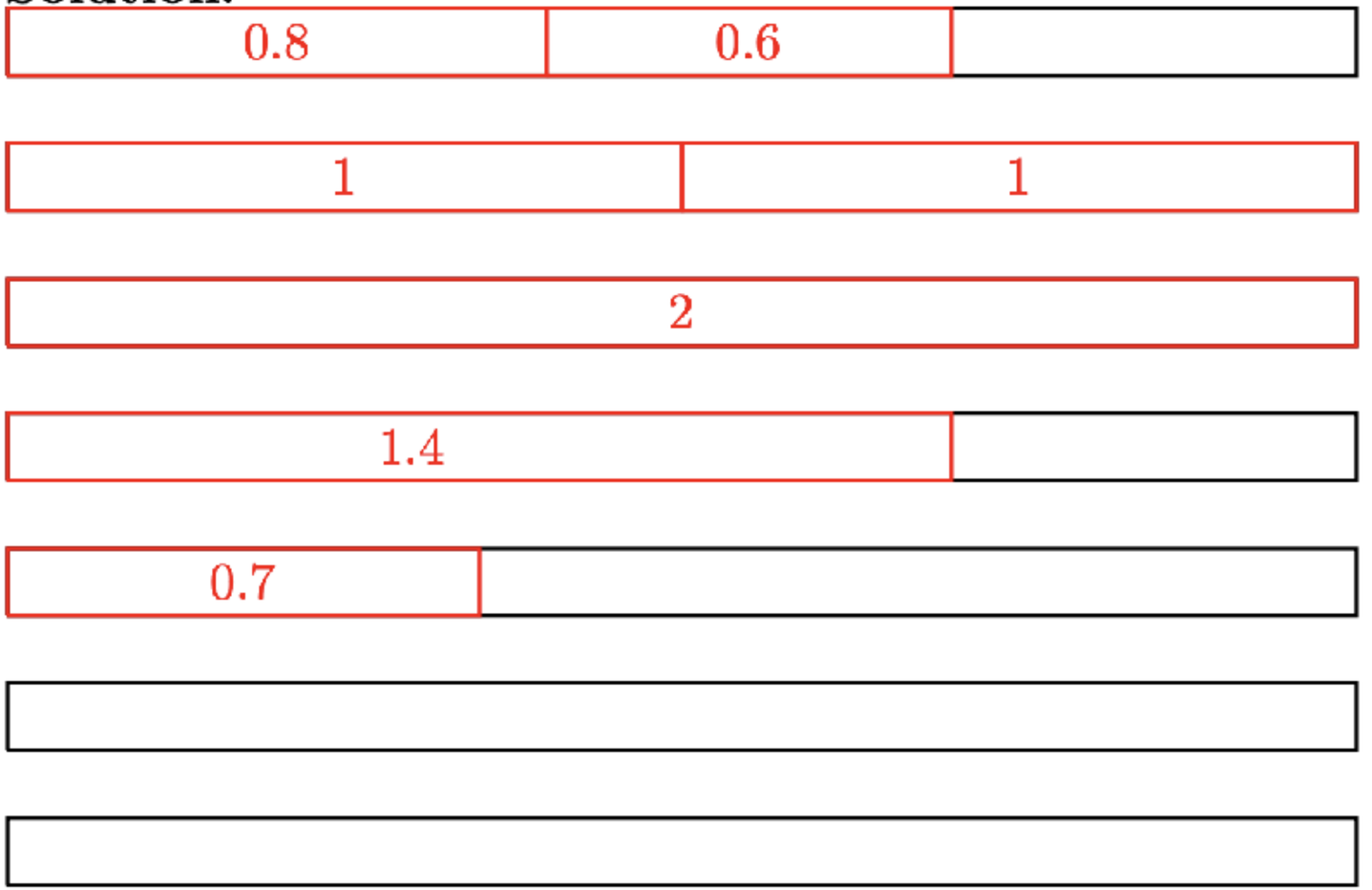
Problem 2 (4 points)

The company has the following order for which they use poles of length $m = 2$

$$x = (0.8, 0.6, 1, 2, 1.4, 1, 0.7).$$

Apply the procedure described above and mark the solution below using the poles drawn.
How many poles are used? Is this the optimal solution?

Solution:



Applying the described procedure results in using 5 poles.
The optimal solution only needs 4 poles.

Problem 3 (4 points)

Show that for this strategy the algorithm has an approximation factor of 2.

Solution:
The optimal solution can be bounded by the sum of all lengths divided by m:

$$f(x^*) \geq \frac{1}{m} \left\lceil \sum_{i=1}^{n} x_i \right\rceil$$

For the result of our greedy algorithm, there cannot be two poles of length less than $\frac{m}{2}$ left when the algorithm terminated. This is because otherwise the algorithm would have used the first of these rest

pieces to cut the other from. Therefore,

$$f(x) \leq \frac{2}{m} \left\lceil \sum_{i=1}^{n} x_i \right\rceil$$

combined this results in

$$\frac{f(x)}{f(x^*)} \leq 2.$$

Problem 4 (4 points)

Describe a family of instances such that the algorithm computes a $\frac{2m}{m+1}$ approximation on those instances.

Solution:
For $4m$ orders in the form

$$x = (\frac{m}{2}, \frac{1}{2m}, \frac{m}{2}, \frac{1}{2m}, ..., \frac{m}{2}, \frac{1}{2m})$$

the algorithm described above would result in using $2m$ poles.
The optimal solution would be to cut m poles in half and one pole has to be cut into pieces of size $\frac{1}{2m}$ resulting in using $m + 1$ poles.