Algorithms and Data structures 2020: Second assignment

November 2020

1 Instructions

You are allowed to submit a solution in either Java, C++, C or Python 3. You are only allowed to use the Standard Library corresponding to your selected language. You will find a set of examples to test your solution on Brightspace. Besides handing in code, we would like to receive a report – **of at least 2 and at most 10 pages** – in which you explain your algorithm and analyse its correctness and runtime complexity.

The deadline for sending in your solution is January 7 (2021), 23:59, Nijmegen time. You can submit your solutions via Brightspace. You are allowed to work in groups of two persons. Only one team member has to submit a solution; the names of both team members must be mentioned in the report.

2 Grading

Grades will be determined as follows. You may earn up to 100 points for your solution:

- 20 points for the explanation of your algorithm.
- 10 points for the correctness analysis.
- 10 points for the complexity analysis.
- 50 points for the test results. We will be running several tests and you will get points for every correct answer within the time limit.
- 10 points for the quality of the code.

The grade is the total number of points divided by 10. If you have any questions, do not hesitate to contact Timo Maarse, `t.maarse@student.ru.nl`.

3 Saving Money in the Supermarket

Dutch people are known for their thriftiness (cheapness).[1] Therefore, if you *really* want to impress your Dutch friends with your knowledge of algorithms and data structures, explain them how they may save some cents in the Albert Heijn (AH).

As you know, when you pay (by cash) in the AH, the total sum is rounded to multiples of 5 cents. For example, when you buy AH rijstwafels, costing 39 cents, you have to pay 40 cents. On the other hand, when you buy a carton of AH ijsthee perzik, costing 61 cents, you only have to pay 60 cents. The Dutch exploit this fact, by reordering products and grouping them (with the checkout divider/beurtbalkje) such that every group is rounded down optimally, saving some cents.

Suppose you want to save money but you are not willing to reorder the products (that is way too much of a hassle). So you have to solve the following problem: Given n products in order with costs $c_1, \ldots, c_n$ and given k dividers, can you place the dividers in such a way that the total amount you have to pay is minimized?

Figure 1: Our queen Maxima was a victim of Dutch cheapness when a loyal subject offered her flowers that were on sale.

3.1 Input

You are given the following data (via `stdin`):

- One line with the number of products: n ($1 \leq n \leq 10\,000$) and the number of dividers: k ($0 \leq k \leq 100$).
- One line with all the n costs in cents: $c_1, \ldots, c_n$ ($1 \leq c_i \leq 50000$). The products come in order they are put on the conveyor belt (so c_1 is closest to the cashier).

3.2 Output

You should return (via `stdout`):

- One line with the minimal amount (in cents) you have to pay to buy all the products, using up to k dividers.

[1]Source: `http://stuffdutchpeoplelike.com/2015/04/03/no-67-dutch-cheapness-thriftiness-cheap/`.

3.3 Examples

Sample input 1:

```
5 1
10 23 43 637 45
```

Sample output 1:

```
755
```

Sample input 2:

```
6 2
1 1 1 1 1 1
```

Sample output 2:

```
0
```