University of Zagreb
Faculty of Electrical Engineering and Computing
Consumer Computing Lab

Analysis of massive datasets

Authors: Marin Šilić, Klemo Vladimir, Adrian Kurdija
Ac. year. 2017/2018

2. Lab exercise

The goal of this exercise is to implement the estimation of count of ones in a bit stream using **Datar-Gionis-Indyk-Motwani (DGIM)** approach.

2.1 DGIM algoritm

DGIM estimates the number of ones in a given bit stream, inside a fixed-sized window of last **N** bits, without keeping the stream in memory. The number N will be given and, for this exercise, it will not exceed 10^6, while the actual stream of zeros and ones can be **smaller or greater** than N.

The algorithm is based on keeping track of **buckets** of sizes (count of 1s) equal to **powers of 2** (the largest bucket is not greater than $\mathbf{\log_2 N}$). In the variation of DGIM used in this exercise, we allow at most **two** buckets of equal size (it is the simplest variation). This requirement should be kept by iteratively merging the buckets while the requirement is not met (i.e., while there are three buckets of equal size). The buckets must not overlap and each digit 1 in the input stream must be covered by exactly one bucket. You are required to **delete** the buckets outside of the observed window of the last N bits. For this purpose, each bucket contains a **timestamp** corresponding to the newest 1 in the bucket. The timestamps are positive integers starting with **t=0**.

The buckets are used to answer queries of the form "***how many 1s are there among the last k bits?***", where $1 \leq k \leq N$. The estimate of the answer should be calculated by steps from the literature:
(1) find the largest bucket *z* with the timestamp still within the last *k* bits,
(2) sum up the sizes of all buckets with more recent timestamps than *z*,
(3) add half the size of bucket *z* to the sum (*rounding to the lower integer*).

For more details on the algorithm, please check:

- AVSP_Data_Streams.pdf (lecture)
- Maintaining stream statistics over sliding windows (paper)

- Mining Massive Datasets - Ch. 4 (Mining Data Streams (book)

Note: Although the limits of the evaluation system for this exercise allow the given stream to be kept in memory, the purpose of the algorithm is not to do it, as the streams are infinite in practice.

2.2 Input and output

From *standard input (stdin)*[1] you should read the input in the following format:

```
N
bits or query
...
bits or query
```

Where:

- `N` - window size ($1 < N < 10^6$)
- `bits` – a sequence of at most 80 digits (0 or 1) in the same line, for readability purposes
- `query` - a query in form "q *k*", where *k* is the size of the query ($1 \leq k \leq N$)

Each line ends with a newline character (**\n**). The first line contains the integer N. Then the **arbitrary** number of lines follow, containing either input stream bits (a sequence of 0s and 1s without spaces) or query (character "q", space, integer) which should be answered.

Note: there is only **one** stream in the input file, even though it is broken into multiple lines and interrupted by queries. Also, queries do not affect the timestamps.

Input example:

```
100
1010101101
1110101011
q 20
1000010010
q 3
```

[1] In other words, don't use files in your code, use the basic input/print (or cin/cout) functions. A file can (and will) be redirected to standard input by a command such as *my_program.exe* < *my_input.txt* in command prompt. End of input can be recognized, for example, as the EOF character.

Window size is N = 100. Total stream size is 10*3=30 bits. The first bit is 1 (t=0), the second bit is 0 (t=1), the third bit is 1 (t=2), etc. After 20th bit (at t=19) the first query (`q 20`) should be answered. After 30th bit, the second query (`q 3`, t=29) should be aswered. The output should contain one line for each query:

```
11
0
```

Notes:

- Time limit is 20 seconds, and inputs will have at most 10^5 lines.
- The input point for solutions in Java should be in the **DGIM class**. The input point for the solutions in Python should be in the **DGIM.py** file.

2.3 Sample test case

The sample input file along with the expected output can be found here. Please check the correctness of your solution on the given example before submitting it. The evaluation of this task will be performed on 5 input files, including the sample one.