■ **Exercise 1 (Streaming algorithm for frequent items).** We want to design a streaming algorithm that finds all the items in a stream of n items with frequency strictly greater than n/k for some fixed k. Consider the following algorithm:

Algorithm 1 Misra-Gries algorithm for frequent items

Initialize: $A :=$ empty dictionnary
for $i = 1..n$ **do**
  if $x_i \in$ keys(A) **then**
    $A[x_i] := A[x_i] + 1$
  else
    if # keys$(A) < k - 1$ **then**
      $A[x_i] := 1$
    else
      for each $a \in$ keys(A) **do**
        $A[a] := A[a] - 1$
        if $A[a] == 0$ **then**
          Remove a from A
Output: On query a, if $a \in$ keys(A), then report $\hat{f}_a := A[a]$, else report $\hat{f}_a := 0$.

We denote by $f_a = \#\{i : x_i = a\}$ the frequency of a in the stream.

► ***Question* 1.1)** *Show that for all a,* $f_a - \dfrac{n}{k} \leqslant \hat{f}_a \leqslant f_a$.
▷ Hint. *Show that the decrement loop is performed at most $\frac{n}{k}$ times while reading the stream.*
Answer. ▷ For the analysis purposes, we associate to every increment of a value of A, the corresponding item in the stream. Every time a decrement is made in A, we bar the corresponding items in the stream, including the item at the origin at the decrement. It follows that every decrement loop correspond to baring k (unbarred) items in the stream. As there are n items in the stream, the decrement loop is performed at most n/k times in total.

Now, $A[a]$ is incremented at most f_a times, thus $\hat{f}_a \leqslant f_a$. Furthermore, every time item a is read in the stream, either the value of $A[a]$ is increased by 1 or is unchanged and the decrement loop is run. Every time an item $b \neq a$ is read, either $A[a]$ is unchanged or it is decreased by 1 if the decrement loop is performed. It follows that $A[a]$ is at least f_a minus the number of times the decrement loop is performed, which implies that $\hat{f}_a \geqslant f_a - n/k$. ◁

► ***Question* 1.2)** *Conclude that one can find the items with frequency larger than n/k with two passes on the stream.*
Answer. ▷ According the inequality proven above, if $f_a > n/k$, then $\hat{f}_a > 0$ which implies that a belongs to A. Thus all the frequent items belong to A. One can compute the exact frequency of each of these k items in a second pass to determine which in the items of A have indeed a frequency $> n/k$. The total number of bits needed is $O(k \log n)$. ◁

► ***Question* 1.3)** *Let* $\hat{n} = \sum_{a \in \text{keys}(A)} A[a]$. *Show that for all a,* $f_a - \dfrac{n - \hat{n}}{k} \leqslant \hat{f}_a \leqslant f_a$.
Answer. ▷ Recall the baring scheme in the answer to question 1.1. Just remark that $\hat{n}$ items are "unbarred" at the end of the algorithm since they correspond to values in A that have not been decreased. As every decrement loop bars k items in the stream, there has been in fact no more than $(n - \hat{n})/k$ executions of the decrement loop. We then conclude as in question 1.1. ◁

■ **Exercise 2 (Streaming algorithm for counting triangles).** We want to estimate the number of triangles in a graph given as a stream of its edges. Let us consider the following algorithm (we assume that the number of vertices and edges, n and m resp., are known).

Algorithm 2 Counting triangles

```
Pick an edge uv uniformly at random in the stream
Pick a vertex w ∈ [n] ∖ {u, v} at uniformly at random
if edges uw and vw appear after edge uv in the stream then
    output m(n − 2)
else
    output 0
```

► ***Question* 2.1)** *Show that* $\mathbb{E}[\text{output}] = \#\mathcal{T}$ *where* $\mathcal{T}$ *denotes the set of triangles in the graph:* $\mathcal{T} = \{\{u, v, w\} \subset [n] : uv, vw, wu \in \text{edges}(G)\}$.
▷ Hint. *What is the probability that the algorithm outputs* $m(n-2)$*?*
Answer. ▷ For all $T \in \mathcal{T}$, let $X_T = \mathbb{1}(T \text{ is detected by the algorithm})$. Then, output $= \sum_{T\in\mathcal{T}} m(n-2)\cdot X_T$ and $\mathbb{E}[\text{output}] = \sum_{T\in\mathcal{T}} m(n-2)\cdot\mathbb{E}[X_T]$. Now, $\mathbb{E}[X_T] = \Pr\{X_T = 1\}$. Consider a triangle $T = \{u, v, w\}$ and suppose without loss of generality that u, v, and w are named such that the edges uv, uw, and vw appear in the stream in that precise order. Triangle T will be detected by the algorithm if and only if edge uv is selected in the first phase of the algorithm and w is selected in the second phase, which occur with probability $1/m$ for the first event and $1/(n-2)$ for the second. It follows that for all triangle T, $\Pr\{X_T = 1\} = 1/m(n-2)$. Thus, $\mathbb{E}[\text{output}] = \sum_{T\in\mathcal{T}} m(n-2)/m(n-2) = \#\mathcal{T}$. ◁

Assume that we know a lower bound t on $\#\mathcal{T}$.

► ***Question* 2.2)** *Design an one-pass* (ε, δ)*-estimator for counting the number of triangles in the graph given as a stream using* $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\cdot\frac{mn}{t})$ *bits of memory.*
▷ Hint. *Compute the variance for the output of the previous algorithm.*
Answer. ▷ According to the previous question, since at most one triangle is detected at a time by the algorithm: $\Pr\{\text{output} = m(n-2)\} = \sum_{T\in\mathcal{T}} \Pr\{X_T = 1\} = \#\mathcal{T}/m(n-2)$. It follows that $\mathbb{E}(\text{output}^2) = m^2(n-2)^2\cdot\#\mathcal{T}/m(n-2) = m(n-2)\#\mathcal{T}$. Thus, $\mathbb{V}\text{ar}[\text{output}] = \#\mathcal{T}\cdot(m(n-2) - \#\mathcal{T})$.

Let $X_{11}, \ldots, X_{k\ell}$ the results of $k\ell$ (parallel) independent runs of the algorithm and $Y_1, \ldots, Y_k$ be the averages of each lot ℓ values: $Y_j = \frac{X_{j1}+\cdots+X_{j\ell}}{\ell}$ for $j = 1..k$. Then, by independence, $\mathbb{V}\text{ar}(Y_j) = \frac{\mathbb{V}\text{ar}(\text{output})}{\ell} = \frac{\#\mathcal{T}\cdot(m(n-2)-\#\mathcal{T})}{\ell}$ for all $j = 1..k$. By Chebyshev's inequality, $\Pr\{|Y_j - \#\mathcal{T}| \geqslant \varepsilon\#\mathcal{T}\} \leqslant \frac{\#\mathcal{T}\cdot(m(n-2)-\#\mathcal{T})}{\ell\varepsilon^2(\#\mathcal{T})^2} \leqslant \frac{mn}{\ell\varepsilon^2\#\mathcal{T}} \leqslant \frac{1}{4}$ as soon as $\ell \geqslant \frac{4mn}{t\varepsilon^2}$. Let Z be the median of $Y_1, \ldots, Y_k$. If $Z \notin (1\pm\varepsilon)\#\mathcal{T}$, then at least $k/2$ values among $Y_1, \ldots, Y_k$ are outside $(1\pm\varepsilon)\mathcal{T}$, and if $\xi_j = \mathbb{1}(Y_j \notin (1\pm\varepsilon)\#\mathcal{T})$, this occurs by Hoeffding's inequality with probability at most : $\Pr\{|Z - \#\mathcal{T}| \geqslant \varepsilon\#\mathcal{T}\} \leqslant \Pr\{\xi_1 + \cdots + \xi_k \geqslant \frac{k}{2}\} \leqslant \Pr\{\xi_1 + \cdots + \xi_k - \mathbb{E}[\xi_1 + \cdots + \xi_k] \geqslant \frac{k}{4}\} \leqslant \exp(-\frac{2(k/4)^2}{k}) \leqslant \delta$ as soon as $k \geqslant 8\ln\frac{1}{\delta}$.

It follows that we get a one-pass (ε, δ)-estimator for counting the number of triangles in the graph using at most $O(\frac{1}{\varepsilon^2}\ln\frac{1}{\delta}\cdot\frac{mn}{t})$ bits of memory (since we only need to remember if $X_{ij} > 0$ or $= 0$).
◁

Note that it can be shown that there is no $o(n^2)$-space algorithm that approximates multiplicatively the number of triangles in a graph unless some lower bound is known on the number of triangles.