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 dictionary
  - for \( i = 1 \ldots n \) do
    - if \( x_i \in \text{keys}(A) \) then
      \[ A[x_i] := A[x_i] + 1 \]
    - else if \# keys(A) < \( k - 1 \) then
      \[ A[x_i] := 1 \]
    - else
      - for each \( a \in \text{keys}(A) \) do
        \[ A[a] := A[a] - 1 \]
        - if \( A[a] == 0 \) then Remove \( a \) from \( A \)
    - Output: On query \( a \), if \( a \in \text{keys}(A) \), 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 \( a \), \( f_a - \frac{n}{k} \leq \hat{f}_a \leq f_a \).

**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 barring \( k \) (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 \( \hat{f}_a \leq f_a \). 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 \( \hat{f}_a \geq f_a - n/k \).

**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 \( a \), \( f_a - \frac{n - \hat{n}}{k} \leq \hat{f}_a \leq f_a \).

**Answer.** Recall the baring scheme in the answer to question [3]. 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 [4].
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

1. Pick an edge \( uv \) uniformly at random in the stream.
2. Pick a vertex \( w \in [n] \setminus \{u, v\} \) at uniformly at random.
3. If edges \( uw \) and \( vw \) appear after edge \( uv \) in the stream then
   - **output** \( m(n - 2) \)
4. Else
   - **output** 0

**Question 2.1** Show that \( \mathbb{E}[\text{output}] = \#T \) where \( T \) denotes the set of triangles in the graph: 
\[
T = \{ \{u, v, w\} : uv, uw, wu \in \text{edges}(G) \}.
\]

Hint. What is the probability that the algorithm outputs \( m(n - 2) \)?

**Answer.** For all \( T \in T \), let \( X_T = 1 \) if \( T \) is detected by the algorithm. Then, \( \mathbb{E}[\text{output}] = \sum_{T \in T} \mathbb{E}[X_T] \). Now, \( \mathbb{E}[X_T] = \mathbb{P}(X_T = 1) \). 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 occurs with probability \( \frac{1}{m} \) for the first event and \( \frac{1}{n(n - 2)} \) for the second. It follows that for all triangle \( T \), \( \mathbb{P}(X_T = 1) = \frac{1}{m} \cdot \frac{1}{n(n - 2)} = \frac{\#T}{m(n - 2)} \). Thus, \( \mathbb{E}[\text{output}] = \sum_{T \in T} \frac{\#T}{m(n - 2)} = \#T \).

Assume that we know a lower bound \( \ell \) on \( \#T \).

**Question 2.2** Design an one-pass \((\varepsilon, \delta)\)-estimator for counting the number of triangles in the graph given as a stream using \( O\left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \cdot \frac{m}{n}\right) \) 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: \( \mathbb{P}(\text{output} = m(n - 2)) = \sum_{T \in T} \mathbb{P}(X_T = 1) = \#T / m(n - 2) \). It follows that \( \mathbb{E}[(\text{output})^2] = m^2(n - 2)^2 \cdot \#T / m(n - 2) = m(n - 2) \#T \). Thus, \( \mathbb{V}[\text{output}] = \#T \cdot (m(n - 2) - \#T) \).

Let \( X_{k1}, \ldots, X_{k\ell} \) be the results of \( k\ell \) (parallel) independent runs of the algorithm and \( Y_1, \ldots, Y_k \) be the averages of each \( \ell \) values: \( Y_j = \frac{X_{j1} + \cdots + X_{j\ell}}{\ell} \) for \( j = 1..k \). Then, by independence, \( \mathbb{V}[Y_j] = \frac{\mathbb{V}[\text{output}]}{\ell} = \frac{\#T \cdot (m(n - 2) - \#T)}{\ell^2} \) for all \( j = 1..k \). By Chebyshev's inequality, \( \mathbb{P}\{|Y_j - \#T| \geq \varepsilon \#T\} \leq \frac{\#T \cdot (m(n - 2) - \#T)}{\ell^2 \varepsilon^2 \#T} \leq \frac{mn}{\ell^2 \varepsilon^2} \leq \frac{1}{4} \) as soon as \( \ell \geq 4mn / \varepsilon^2 \).

Let \( Z \) be the median of \( Y_1, \ldots, Y_k \). If \( Z \not\in (1 \pm \varepsilon) \#T \), then at least \( k / 2 \) values among \( Y_1, \ldots, Y_k \) are outside \((1 \pm \varepsilon) \#T \), and if \( Z \in (1 \pm \varepsilon) \#T \), then this occurs by Hoeffding's inequality with probability at most: \( \mathbb{P}\{|Z - \#T| \geq \varepsilon \#T\} \leq \mathbb{P}\{\xi_1 + \cdots + \xi_k \geq k \#T / 2\} \leq \exp\left(-\frac{2(k/4)^2}{k}\right) \leq \delta \) as soon as \( k \geq 8 \ln \frac{1}{\delta} \).

It follows that we get a one-pass \((\varepsilon, \delta)\)-estimator for counting the number of triangles in the graph using at most \( O\left(\frac{1}{\varepsilon^2} \ln \frac{1}{\delta} \cdot \frac{m}{n}\right) \) bits of memory (since we only need to remember if \( X_{ij} \neq 0 \) 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.