We will describe a way to generate a stream from left to right and we want to minimize the memory needed by the algorithm to accomplish this task. One can show that any deterministic algorithm that approximates the value of $\sum_{i=0}^{m-1} (f_i(x))^2$, as the zero-th moment of the frequencies of each element of $[m]$ in the stream. Let us denote by $S_x = \{ x_i : i \in [n] \}$ the set of the values in the stream $x$. Note that $F_0(x) = \#S_x$. (We may drop the $x$ when the context is clear.)

The streaming constraint is that the algorithm will see every $x_i$ only once as it reads the stream from left to right and we want to minimize the memory needed by the algorithm to accomplish this task. One can show that any deterministic algorithm that approximates the value of $F_0$ within 10% requires at least $\Omega(n)$ bits of memory. Here, we will design a randomized algorithm that accomplishes this task using only $O(\log n + \log m)$ bits of memory.

We start with an hypothetical algorithm using uniform real random numbers and a hypothetical family of hash functions and then see how to turn it into an effective algorithm.

Assume that we are given a random function $h : [m] \to \{0, 1\}$, i.e. such that for every $x \in [m]$, $h(x)$ is a (fixed) independent uniform random real in $\{0, 1\}$. The algorithm proceeds as follows: when reading the stream, record in memory the minimum value $\mu$ so far of the $h(x_i)$'s, and output $1/\mu - 1$ at the end.

**Exercise 1 (Pairwise independent random bits).** We will describe a way to generate $n$ pairwise independent uniform random bits $X_1, \ldots, X_n$ using only $\ell = \lceil \log_2 n \rceil$ “true” uniform independent random bits $Y_1, \ldots, Y_{\ell}$.

**Question 1.1** Let $(G, \cdot)$ be a finite group, $X$ a random variable over $G$ and $U$ an independent uniform random variable over $G$. Show that $X \cdot U$ is an uniform random variable over $G$ independent from $X$.

Let $\{i\} = \{j : j\text{-th bit of } i \text{ written in binary is } 1\} \subseteq \{1, \ldots, \ell\}$ such that $i = \sum_{j \in [\ell]} 2^{j-1}$ for all $i \in \{1, \ldots, n\}$. Consider $Y_1, \ldots, Y_{\ell}$ uniform independent random bits. We then set $X_i = \bigoplus_{j \in [\ell]} Y_j$ for $i = 1 \ldots n$, where $a \oplus b$ denote the XOR of $a$ and $b$ (i.e. their sum modulo 2). For instance: $13 = 1101$ in binary, thus $X_{13} = Y_4 \oplus Y_3 \oplus Y_1$.

**Exercise 2 (A streaming algorithm for counting the number of distinct values).** We are given a stream of numbers $x_1, \ldots, x_n \in [m]$ and we want to compute the number of distinct values in the stream: $F_0(x) = \#\{ x_i : i \in [n] \}$. (Note that if $f_a(x) = \#\{ i : x_i = a \}$, we can express $F_0(x) = \sum_{a=0}^{m-1} f_a(x)$, as the zero-th moment of the frequencies of each element of $[m]$ in the stream). Let us denote by $S_x = \{ x_i : i \in [n] \}$ the set of the values in the stream $x$. Note that $F_0(x) = \#S_x$. (We may drop the $x$ when the context is clear.)

The streaming constraint is that the algorithm will see every $x_i$ only once as it reads the stream from left to right and we want to minimize the memory needed by the algorithm to accomplish this task. One can show that any deterministic algorithm that approximates the value of $F_0$ within 10% requires at least $\Omega(n)$ bits of memory. Here, we will design a randomized algorithm that accomplishes this task using only $O(\log n + \log m)$ bits of memory.

We start with an hypothetical algorithm using uniform real random numbers and a hypothetical family of hash functions and then see how to turn it into an effective algorithm.

Assume that we are given a random function $h : [m] \to \{0, 1\}$, i.e. such that for every $x \in [m]$, $h(x)$ is a (fixed) independent uniform random real in $\{0, 1\}$. The algorithm proceeds as follows: when reading the stream, record in memory the minimum value $\mu$ so far of the $h(x_i)$'s, and output $1/\mu - 1$ at the end.

**Question 2.1** Show that $\Pr\{ \mu \geq t \} = (1 - t)^{F_0}$.

**Question 2.2** Show that $\mathbb{E}[\mu] = \frac{1}{F_0 + 1}$.

However, the following fact seems to imply that the algorithm is wrong.
**Question 2.3**  Show that $\mathbb{E}[1/\mu] = \infty$.

But, fortunately:

**Question 2.4**  Compute $\mathbb{V}ar(\mu)$ and show that $\mathbb{V}ar(\mu) \leq 2 \mathbb{E}[\mu]^2$.

**Question 2.5**  Design and analyze a $(\varepsilon, \delta)$-estimator for $F_0$. Still, what is the expected value of its output? Is there a paradox here?

> **Hint.** First, design an $(\varepsilon, \delta)$-estimator for $\mu$.

Unfortunately, such a random function $h$ requires storing $m$ reals in memory. The key to reduce the memory needed is to relax the independence of the hash value to pairwise independence only. In the following, we will approximate the minimum of the hash keys by recording only the position of their first non-zero bit in their binary writing. We proceed as follows.

Let $\ell = \lfloor \log_2 m \rfloor$ such that $2^{\ell-1} < m \leq 2^\ell$ and consider the field with $2^\ell$ elements $\mathbb{F}_{2^\ell}$. We identify $\mathbb{F}_{2^\ell}$ through canonical bijections to the set of bit-vectors $\{0, 1\}^\ell$ and to the set of integers $\{0, \ldots, 2^\ell - 1\}$ written in binary. For every pair $(a, b) \in \mathbb{F}_{2^\ell}$, consider the hash function $h_{ab} : \mathbb{F}_{2^\ell} \to \mathbb{F}_{2^\ell}$ defined as $h_{ab}(y) = a + b \cdot y$. For every $y \in \mathbb{F}(2^\ell) \equiv \{0, 1\}^\ell$, we denote by $\rho(y) = \max\{j \in [\ell] : y_1 = \cdots = y_j = 0\}$ the largest index $j$ such that the first $j$ bits of $y$, seen as a bit-vector, are all zero. Let us now consider the following streaming algorithm:

**Algorithm 2**  Streaming algorithm for $F_0$

Let $\ell = \lfloor \log_2 m \rfloor$, we identify each element $x_i \in [m]$ of the stream with its corresponding element in $\mathbb{F}_{2^\ell}$.

Pick uniformly and independently two random elements $a, b \in \mathbb{F}_{2^\ell}$.

Compute $R = \max_{i=1..n} \rho(h_{ab}(x_i))$.

return $2^R$.

**Question 2.6**  Show that for all $c \in \mathbb{F}_{2^\ell}$ and $r \in \{0, \ldots, \ell\}$, $\Pr_{a,b}\{\rho(h_{ab}(c)) \geq r\} = \frac{1}{2^r}$.

> **Hint.**  Show that $h_{ab}(c)$ is uniform in $\mathbb{F}_{2^\ell}$.

Let $W_r^c$ the indicator random variable for the event $\rho(h_{ab}(c)) \geq r$. Let $Z_r = \sum_{c \in S_a} W_r^c$, be the number of the values in the stream whose $r$ first bits of their hash key are all zero.

**Question 2.7**  Show that $\mathbb{E}[Z_r] = F_0/2^r$.

**Question 2.8**  Show that the random values $h_{ab}(0), \ldots, h_{ab}(2^\ell - 1)$ are uniform and pairwise independent.

> **Hint.**  Show that if $c \neq d$, then for all $\gamma, \delta \in \mathbb{F}_{2^\ell}$, $\Pr_{a,b}\{\rho(h_{ab}(c), h_{ab}(d)) = (\gamma, \delta)\} = \frac{1}{2^r 2^{\ell}}$.

**Question 2.9**  Show that $\mathbb{V}ar(Z_r) = F_0/2^r \left(1 - \frac{1}{2^r}\right) < \mathbb{E}[Z_r]$.

Fix some $\eta > 1$.

**Question 2.10**  Show that $\Pr\{Z_r > 0\} < \frac{1}{\eta}$ for all $r \in \{0, \ldots, \ell\}$ such that $2^r > \eta F_0$.

> **Hint.**  $Z_r$ is an integer and use Markov’s inequality.

**Question 2.11**  Show that $\Pr\{Z_r = 0\} < \frac{1}{\eta}$ for all $r \in \{0, \ldots, \ell\}$ such that $2^r < \eta F_0 / \eta$.

> **Hint.**  $Z_r$ is an integer and apply Chebyshev’s inequality.

**Question 2.12**  Conclude that for all $\eta > 2$, $\Pr\{2^R \in [F_0/\eta, \eta F_0]\} > 1 - \frac{2}{\eta}$. The algorithm outputs thus a $\eta$-approximation of $F_0$ with probability at least $1 - 2/\eta$ for all $\eta > 2$. How many bits of memory does it require?
We have thus obtained a $(\varepsilon, 2/(1 + \varepsilon))$-estimator for $F_0$ using $O(\log m)$ bits of memory for $\varepsilon > 1$. Getting a $(\varepsilon, \delta)$-estimator for $F_0$ in $O_{\varepsilon, \delta}(\log m + \log n)$ bits of memory for arbitrarily small $\varepsilon, \delta > 0$ requires a lot more work...