Sublinear Time and Space Algorithms 2024A – Lecture 2
Frequency Vectors, Distinct Elements, Frequency Moments, and the AMS algorithm*

Robert Krauthgamer

1 Frequency-vector model

A famous and common setting for data-stream problems lets the input be a stream of m items from a universe $[n] = \{1, \dots, n\}$; the stream $\sigma = (\sigma_1, \dots, \sigma_m)$ implicitly defines a *frequency vector* $x \in \mathbb{R}^n$, where coordinate x_i counts the frequency of item $i \in [n]$ in the stream.

Example: The sequence of IP addresses observed by a router. Here, $n = 2^{32}$ is huge but the vector x is sparse (many zeros).

Remark: In this setting, it is common to assume $m = \text{poly}(n)$, hence one machine word can store value in the ranges $[n]$ and $[m]$. The usual goal is to achieve storage requirement polylog(n).

Example Problems: Two classical computational problems ask for the most frequent item and for the number of distinct items, which can be expressed in terms of the frequency vector x as $\|x\|_\infty$ and $\|x\|_0$, respectively.

Suppose we are guaranteed that one item appears more than half the time, i.e., there exists (unknown) $i \in [n]$ such that $x_i > m/2$. Design a streaming algorithm with $O(\log n)$ storage that finds this item i. Hint: Store only two items.

Can you provide a $(1 + \epsilon)$-approximation to its frequency? Can you extend it from 2 to every k (i.e., frequency $> m/k$)?

Variations and further questions (we will discuss only some of these):

- $\|x\|_0$ (distinct elements)
- heavy hitters ($\|x\|_\infty$ when it is guaranteed to be "large")
- $\|x\|_2$ (reflects the probability that two random items from the stream are equal)
- more generally $\|x\|_p$

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

- ℓ_p-sampling
- item deletions (turnstile updates to x), now even $\|x\|_1$ is interesting
- sliding window (always refer to the w most recent items, for a parameter w known in advance)
- multiple passes over the input

2 Distinct Elements

Problem Definition: Let $x \in \mathbb{R}^n$ be the frequency vector of the input stream, and let $\|x\|_0 = |\{i \in [n] : x_i > 0\}|$ be the number of distinct elements in the stream. It's also called the F_0-moment of σ.

Naive algorithms: Storage $O(n)$ (a bit for each possible item) or $O(m \log n)$ (list of seen items) bits.

Algorithm FM [Flajolet and Martin, 1985]:

It employs a "hash" function $h : [n] \to [0, 1]$ where each $h(i)$ is drawn independently from a uniform distribution on $[0, 1]$. (This is an "idealized" description, because even though we can generate n truly random bits, we cannot store and re-use them.)

Idea: We will see exactly $\mathrm{d}^* = \|x\|_0$ distinct hashes, and since they are random, by symmetry their *minimum* should be around $1/(\mathrm{d}^* + 1)$.

1. Init: $z = 1$ and a hash function h
2. Update: When item $i \in [n]$ is seen, update $z = \min\{z, h(i)\}$
3. Output: $1/z - 1$

Storage requirement: $O(1)$ words (not including randomness); we will discuss implementation issues later.

Denote by $\mathrm{d}^* := \|x\|_0$ the true value, and let Z denote the final value of z (to emphasize it is a random variable).

Lemma 1: $\mathbb{E}[Z] = 1/(\mathrm{d}^* + 1)$.

Note: This is the expectation of Z and not of its inverse $1/Z$ (as used in the output).

Proof: We will use a trick to avoid the integral calculation (which is actually straightforward). Choose an additional random value X uniformly from $[0, 1]$ (for sake of analysis only), then by the law of total expectation

$$\mathbb{E}[Z] = \mathop{\mathbb{E}}_{Z}[\Pr_{X}[X < Z \mid Z]] = \mathop{\mathbb{E}}_{Z}[\mathop{\mathbb{E}}_{X}[\mathbb{1}_{\{X<Z\}} \mid Z]] = \mathbb{E}[\mathbb{1}_{\{X<Z\}}] = 1/(\mathrm{d}^* + 1).$$

Lemma 2: $\mathbb{E}[Z^2] = \frac{2}{(d^*+1)(d^*+2)}$ and thus $\mathrm{Var}[Z] \le (\mathbb{E}[Z])^2 \le O(1/d^{*2})$.

Exer: Prove this lemma using the above trick with two new random values (and/or prove both by calculating the integral).

Algorithm FM+:

1. Run $k = O(1/\varepsilon^2)$ independent copies of algorithm FM, keeping in memory $Z_1, \ldots, Z_k$ (and functions $h^1, \ldots, h^k$)

2. Output: $1/\bar{Z} - 1$ where $\bar{Z} = \frac{1}{k}\sum_{i=1}^{k} Z_i$

As before, averaging reduces the standard deviation by factor $\sqrt{k}$, and then applying Chebyshev's inequality to $\bar{Z}$, WHP

$$\bar{Z} \in (1 \pm 3/\sqrt{k})\,\mathbb{E}[Z] = (1 \pm 3/\sqrt{k}) \cdot 1/(\mathrm{d}^* + 1)$$

in which case its inverse is $1/\bar{Z} \in (1 \pm \varepsilon)(\mathrm{d}^* + 1)$.

Storage requirement: $O(k) = O(1/\varepsilon^2)$ words (not including randomness); we will discuss implementation issues later.

Exer: Can we change h to be a random permutation $h : [n] \to [n]$?

Remark: The storage can be improved similarly to the probabilistic counting. It suffices to store a $(1+\varepsilon)$-approximation of z, which can reduce the number of bits from $O(\log n)$ (in a "typical" implementation of the real-valued hashes) to $O(\log \log n)$. A particularly efficient 2-approximation is to store the number of zeros in the beginning of z's binary representation.

Remark: Notice this algorithm does not work under deletions.

3 Frequency Moments and the AMS algorithm

ℓ_p-norm problem: Let $x \in \mathbb{R}^n$ be the frequency vector of the input stream, and fix a parameter $p > 0$.

Goal: estimate its ℓ_p-norm $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$. We focus on $p = 2$.

Theorem 1 [Alon, Matthias, and Szegedy, 1996]: One can estimate the ℓ_2 norm of a frequency vector $x \in \mathbb{R}^n$ within factor $1+\varepsilon$ [with high constant probability] using storage requirement of $s = O(\varepsilon^{-2})$ words. In fact, the algorithm stores a linear sketch of dimension s.

Algorithm AMS (also known as Tug-of-War):

1. Init: choose $r_1, \ldots, r_n$ independently at random from $\{-1, +1\}$

2. Update: maintain $Z = \sum_i r_i x_i$

3. Output: to estimate $\|x\|_2^2$ report Z^2

The sketch Z is linear in x, and thus the update step can indeed be implemented in a streaming fashion. Indeed, if the sketch is some linear map $L : \mathbb{R}^n \to \mathbb{R}^s$, then it can be updated by $L(x + e_i) = L(x) + L(e_i)$.

Storage requirement: $O(\log(nm))$ bits, not including randomness; we will discuss implementation issues a bit later.

Analysis: We saw in class that $\mathbb{E}[Z^2] = \sum_i x_i^2 = \|x\|_2^2$, and $\mathrm{Var}(Z^2) \le 2(\mathbb{E}[Z^2])^2$.

Algorithm AMS+:

1. Run $k = O(1/\varepsilon^2)$ independent copies of Algorithm AMS, denoting their Z values by $Z_1, \ldots, Z_k$, and output the mean of these copies $\tilde{Y} = \frac{1}{k}\sum_j Z_j^2$.

Observe that the sketch $(Z_1, \ldots, Z_k)$ is still linear.

Storage requirement: $O(k) = O(1/\varepsilon^2)$ words (for constant success probability), not including randomness.

Analysis: We saw in class that

$$\Pr[|\tilde{Y} - \mathbb{E}\,\tilde{Y}| \ge \varepsilon\,\mathbb{E}\,\tilde{Y}] \le \tfrac{\mathrm{Var}(\tilde{Y})}{\varepsilon^2(\mathbb{E}\,\tilde{Y})^2} = \tfrac{\mathrm{Var}(Z^2)/k}{\varepsilon^2(\mathbb{E}\,Z^2)^2} \le \tfrac{2}{k\varepsilon^2}.$$

Choosing appropriate $k = O(1/\varepsilon^2)$ makes the probability of error an arbitrarily small constant.

Notice it actually gives a $(1 \pm \varepsilon)$-approximation to $\|x\|_2^2$, which is immediately yields a $(1 \pm \varepsilon)$-approximation to $\|x\|_2$.

Exer: What would happen in the accuracy analysis if the r_i's were chosen as standard gaussians $N(0,1)$?