Sublinear Time and Space Algorithms 2026A – Lecture 1
Data-stream algorithms, probabilistic counting*

Robert Krauthgamer

1 Introduction

Massive datasets: In today's era many datasets are massive (Big Data), and a major bottleneck is moving data between sites/processors/cores, or from disk/storage to memory, and so forth. We will focus on *modern algorithmic paradigms* that aim to be much more efficient than the classical goal of polynomial, or linear, time.

Approaches: Sublinear time or space (and indirectly, fewer random coins or probes to the input).

Tools:

Approximation and randomization (and derandomization!) will both play a crucial tool.

We will touch upon impossibility results that are information-theoretic (do not rely on computational complexity assumptions like P vs NP).

We will not study specific application domains, but rather focus on algorithmic techniques.

2 Streaming model

In the basic version of this model (also called the data-stream model), the input is a huge sequence (stream) that can be read only sequentially, and the algorithm's memory is small compared to the input size. Since the algorithm cannot access "earlier" input portions (nor store them in full), it must immediately process and "compress" the input.

Notation: We consider a stream of m items, denoted $\sigma = (\sigma_1, \ldots, \sigma_m)$. But sometimes (e.g., in today's class) m is not known in advance.

Typically, an algorithm in this model consists of three procedures: (1) initialization, (2) update (applied to each item in the stream), and (3) output.

*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.

Efficiency: The main measure of efficiency is the algorithm's memory size, referred to as *storage requirement* or *space complexity*. Running time, which is obviously very important as well, is often considered as a secondary measure to be optimized later.

3 Probabilistic Counting

Problem Definition: Given an input stream σ, compute (or approximate) its length m.

Naive solution: $O(\log m)$ bits, exact solution

Approximate solution: Observe that (multiplicative) 2-approximation, i.e., an estimate $\hat{m} \in [m, 2m]$, can be represented using $O(\log \log m)$ bits, but it seems difficult to update this estimate (deterministically).

Definition (randomized approximation): A random variable $\hat{Z}$ is called an (ϵ, δ)-*approximator* to a value $z \in \mathbb{R}$ if

$$\Pr[\hat{Z} \in (1 \pm \epsilon)z] \geq 1 - \delta.$$

Morris' Algorithm [Morris, 1978]:

Intuition: maintain a random counter $X \approx \log_2 m$.

1. Init: $X = 0$
2. Update: For each input item, increment X with probability $1/2^X$
3. Output: $2^X - 1$

Analysis: the proof proceeds in three steps.

1. Compute the expectation of our estimator $\mathbb{E}[2^X - 1]$.
2. Bound its variance $\mathrm{Var}[2^X - 1]$.
3. WHP the estimator is in the range of mean plus/minus a few standard deviations.

Lemma 1: Let X_m be the value of X after seeing m items. Then $\mathbb{E}[2^{X_m} - 1] = m$.

Proof: Was seen in class.

Lemma 2: $\mathrm{Var}[2^{X_m} - 1] \leq \dfrac{3m(m+1)}{2} + 1 = O(m^2)$.

The proof uses basic principles to bound $\mathrm{Var}[2^{X_m} - 1] = \mathrm{Var}[2^{X_m}] \leq \mathbb{E}[2^{2X_m}]$, and continues similarly to Lemma 1.

Exer: Verify that $\mathrm{Var}(2^{X_m}) = m(m-1)/2$.

Chebychev's inequality: Let X be a random variable with finite variance $\sigma^2 > 0$. Then

$$\forall t \geq 1, \quad \Pr\left[|X - \mathbb{E}X| \geq t\sigma\right] \leq \tfrac{1}{t^2}.$$

We would like to apply Chebychev's inequality but our current bound on the standard deviation is $O(m)$, which is a little too weak to be useful.

Algorithm Morris+:

Idea: maintain k basic estimators and report their average.

1. Run $k = O(1/\epsilon^2)$ independent copies of Morris' algorithm, keeping in memory $(X^{(1)}, \ldots, X^{(k)})$

2. Output: $\frac{1}{k}\sum_{i=1}^{k}(2^{X^{(i)}} - 1)$

Analysis:

Let $Y_i = 2^{X^{(i)}} - 1$ be the estimator we get from copy i. Then the algorithm's output is $Y = \frac{1}{k}\sum_i Y_i$.

$$\mathbb{E}[Y] = \tfrac{1}{k}\sum_i \mathbb{E}[Y_i] = \mathbb{E}[Y_1] = m.$$
$$\mathrm{Var}(Y) = \tfrac{1}{k^2}\sum_i \mathrm{Var}(Y_i) = \tfrac{1}{k^2} k\,\mathrm{Var}(Y_1) = O(m^2/k).$$

Setting $k = O(1/\epsilon^2)$, the standard deviation is bounded by $\epsilon m/3$, and we can now apply Chebychev's inequality to get that Algorithm Morris+ provides an $(\epsilon, 1/9)$-approximation to m.

Exer: Prove that with high probability the algorithm's storage is $O(\epsilon^{-2}\log\log m)$ bits.

Hint: Use Markov's inequality.

Markov's inequality: Let X be a nonnegative random variable with a positive finite expectation. Then

$$\forall t \geq 1, \quad \Pr\left[X \geq t \cdot \mathbb{E}X\right] \leq \tfrac{1}{t}.$$

Altogether: By a union bound, we get that with high probability the algorithm is both correct and uses low memory.

Exer: Prove Markov's inequality. (Hint: use the law of total expectation.)

Exer: Prove Chebychev's inequality. (Hint: use Markov's inequality.)

Exer: Generalize Morris' Algorithm to an algorithm that increments X with probability $\frac{1}{(1+a)^X}$ for fixed $0 < a \leq 1$, then use $a = \Theta(\epsilon^2)$ to improve the space complexity to $O(\log\frac{1}{\epsilon} \cdot \log\log m)$.