Sublinear Time and Space Algorithms 2018B – Lecture 1
Data-stream algorithms, probabilistic counting, reservoir sampling, frequency vectors*

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: Limited 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 $\sigma = (\sigma_1, \ldots, \sigma_m)$.

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

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: randomly maintain $X \approx \log_2 m$.

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

Analysis: the proof proceeds in three steps.

1. Compute the expected value of our estimator $\mathbb{E}[2^X - 1]$.
2. Bound the variance of the estimator $\mathrm{Var}[2^X - 1]$.
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 $\mathrm{Var}[2^{X_n} - 1] = \mathrm{Var}[2^{X_n}] \leq \mathbb{E}[2^{2X_n}]$, and continues similarly to Lemma 1.

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

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 deviations is $O(m)$, which is a little too weak to be useful.

Algorithm Morris+:

Idea: average k basic estimators.

1. Run k 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 $Y = \frac{1}{k}\sum_i Y_i$.

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

Setting $k = O(1/\epsilon^2)$, the standard deviation is only $2\epsilon n$, and we can now apply Chebychev's inequality. Thus, Algorithm Morris+ provides an $(\epsilon, 1/4)$-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

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

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

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

4 Reservoir Sampling

Problem definition: Pick a uniformly random item from the stream.

Reservoir Sampling [Vitter, 1985]:

1. Init: $s = \text{null}$

2. When the next item σ_j is read, toss a biased coin and with probability $1/j$ let $s = \sigma_j$ in the stream (note we need to maintain j)

3. Output: s

Lemma 3: This algorithm uses storage $O(\log(n+m))$, and its output is a uniform item from the stream, i.e., each item σ_j ends up being output with probability $1/m$.

Note that items appearing many times are output with high probability.

Exer: Prove this lemma.

5 Frequency-vector model

A famous and common setting for data-stream problems lets the input be a stream of m items from a universe $[n] = \{1, \ldots, n\}$; the stream $\sigma = (\sigma_1, \ldots, \sigma_m)$ 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, $n = 2^{256}$ is huge but the vector x is sparse (many zeros).

Remark: In this setting, it is common to assume $m = \text{poly}(n)$, and the usual goal is to achieve storage requirement $\text{polylog}(n)$.

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 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 guarantee to be "large")
- $\|x\|_2$ (reflects the probability that two random items from the stream are equal)
- more generally $\|x\|_p$
- ℓ_p-sampling
- item deletions (turnstile updates to x), now even $\|x\|_1$ is interesting
- sliding window
- multiple passes over the input