**CS 229r Information Theory in Computer Science** April 11, 2019

Lecture 20

*Instructor: Madhu Sudan* *Scribe: Boriana Gjura*

# 1 Today: Streaming Algorithms

Moving away from communication complexity, in this lecture we will explore other applications of information theory. In particular, we introduce streaming algorithms and illustrate the model by elegant algorithms that compute the frequency of moments of an input stream. We conclude by proving lower bounds, relying on results from multi-party lower bounds of set-disjointness.

# 2 Model

Streaming algorithms process an input stream, given as a sequence of variables, which can be examined in *only a few passes*, sometimes just one. Such algorithms find many practical applications in networking, such as in routers for internet traffic, computing popular IT addresses, etc.

Unless otherwise noted, we will consider single-pass algorithms only. We are interested in what we can compute in this model when the space is small: $S = poly(\log m, \log n)$.

```
Streaming Model:
    Given: An input stream x_1, x_2, ..., x_m, where x_i ∈ [n] for all i ∈ [m].
    Goal:  Compute f(x_1, ..., x_m) for some f : [n]^m → Γ.
    Restriction: Small space S.
        Algorithm:
            State update:    A : {0,1}^S · [n] → {0,1}^S
            Output function: g : {0,1}^S → Γ
```

*A few notes on randomness.*

The state update algorithm A is wlog deterministic, for if necessary, we can store randomness on the description of the states. We do not worry about how complicated A is

We want $f(x_1, \ldots, x_m)$ to be derivable from $\sigma_m$, such that $f(x_1, \ldots, x_m) = g(\sigma_m)$, where

$$\begin{aligned} \sigma_0 &= \overline{0}, \\ \sigma_i &= A(\sigma_{i-1}, x_i), \quad i \in [m]. \end{aligned}$$

In randomized streaming algorithms we can allow $\sigma_0$ to be random, in which case we will require only that the algorithm is correct most of the time, say:

$$\mathbb{P}_{\sigma_0}\big[f(x_1, \ldots, x_m) = g(\sigma_m)\big] \leq \frac{2}{3}.$$

# 3 Example algorithms: frequency moments

The frequency of $i \in [n]$ is defined naturally as $f_i(x_1, \ldots, x_m) = \big|\{j \,|\, x_j = i\}\big|$. We want to compute:

```
K-Frequency Moment:
    Compute F_k(x_1, ..., x_m) = Σ_{i∈[n]} f_i^k(x_1, ..., x_m).
```

## 3.1 Quick overview

We want to compute the $k^{th}$ frequency moments. We first present elegant algorithms for $k = 0, 2$, and then give a clever (but-not-so-intuitive!) algorithm for higher values of $k$. Most of the calculations are left as an exercise to the reader.

*Advance warning!* Without worrying much, we will use a lot of randomness in the following algorithms (you will see this when we pick uniformly random hash functions!). This is not necessary and can be fixed.

Trivially, $F_1 = m$ corresponds to the number of elements in the input stream. A more interesting example is the $0^{th}$ moment, which corresponds to the the number of distinct elements in the stream, as seen below:

$$\begin{aligned} F_0(x_1, \ldots, x_m) &= \lim_{k \to 0} F_k(x_1, \ldots, x_m) \\ &= \big|\{i \,|\, f_i > 0\big| \end{aligned}$$

Can we compute $k^{th}$ moments with non-trivial space? Today we will see the following results.

| **Brief history of the problem** | | |
|---|---|---|
| trivial | $F_1$ | |
| 1985, Flajolet-Martin, 1 | approximate $F_0$ | polylog(n,m) |
| 1996, Alon-Matias-Szegedy, 2 | approximate $F_2$ | polylog(n,m) |
| 2003, BJKS, 3 | approximate $F_k$ | $n^{1-2/k}$ |

The algorithms have the following general structure.

- Find an unbiased estimator of $F_k$, which might have a high variance.
- Run the algorithm a (constant) $\times$ variance times.
- Report the median, using concentration inequalities this gives us the bounds we need.

## 3.2 Algorithm for $F_0$.

```
[Flajolet-Martin '85]
    Pick random h : [n] → [0,1] uniformly at random.
    Compute min_{j∈[m]}{h(x_j)} = h_min.
    Output: 1/h_min.
```

The reader can verify that indeed $\mathbb{E}\big[\frac{1}{h_{min}}\big] \approx F_0$!

The intuitive idea is that if $h$ is uniform, then the $m$ inputs of the stream will be (roughly) uniformly distributed in the interval $[0, 1]$, which then implies that the number of distinct elements is, in expectation, $\frac{1}{h_{min}}$.

This leads to an $O\big(\frac{1}{\varepsilon^2} \log^c n\big)$ algorithm to output $(1 \pm \varepsilon)$-approximation to $F_0$.

*Can we get rid of the expectation?* Remember that the exponential distribution corresponds is the probability distribution that describes the time between events in a Bernoulli process. This way of thinking about it is useful because the process is memoryless (nothing from the past carries to the future). This is particularly useful when we are doing minimum analysis, because the minimum will be distributed as $exp(1/p)$, where $p$ is the success probability of the Bernoulli process. To make the above algorithm cleaner, we can take $h(i)$ i.i.d. $\sim exp(n)$.

*Slightly better algorithm?* Instead of the minimal element, we can pick the $t^{th}$ smallest and output $\frac{t}{h_{t_{min}}}$.

**Exercise.** Verify all the sentences above :)

### 3.3 Algorithm for $F_2$

In a similar spirit, we have the following algorithm for $F_2$.

```
[Alon-Matias-Szegedy]
    Pick h : [n] → {+1, −1} uniformly at random.
    Compute V = Σ_{j=1}^{m} h_{min}.
    Output: V^2.
```

The calculation below shows that $F_2(x_1, \ldots, \times_m) \approx V^2$.

$$\begin{aligned}
\mathbb{E}[V^2] &= \mathbb{E}_h\Big[\big(\sum_{j=1}^{m} h(x_j)\big)\big(\sum_{k=1}^{m} h(x_k)\big)\Big] \\
&= \sum_{j=1}^{m}\sum_{k=1}^{m} \mathbb{E}_h\big[h(x_j)h(x_k)\big] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} \mathbb{1}_{x_j=x_k=i} \\
&= \sum_{i=1}^{n}\Big(\sum_{j=1}^{m} \mathbb{1}_{x_j=i}\Big)^2 \\
&= \sum_{i=1}^{n} f_i^2 \\
&= F_2
\end{aligned}$$

Note that above $h(x_j)h(x_k) = 1$ iff $x_j = x_k$, 0 otherwise.

### 3.4 Arbitrary $k$

What about $F_3, F_4$ and so on? [1] We cannot use the same tricks as earlier, therefore the algorithm below has a different flavor.

```
[AMS Algorithm]
    Pick j ∈ [m] uniformly at random.
    Let i = a_j.
    Let f^t = |{j' | a_{j'} = i, j' ≥ j}| \\count how many times it appears later
    Output: X = (f^t)^k − (f^t − 1)^k.
```

The claim is that $\mathbb{E}[X] = F_k$, and the proof of this statement is left as an **exercise** to the reader. This doesn't seem to have as nice of an intuitive explanation, but it works out!

One can also check that $V[X] \approx n^{1-\frac{1}{k}}$.

## 4 Lower bounds

We now shift our attention to lower bounds for the problems we just considered. We rely heavily on the lower bounds of set disjointness shown in [3].

[1] This is well-defined for fractional $k$ as well, but in this lecture we will take $k \in \mathbb{N}$ for simplicity.

### 4.1 T-party communication model

The model consists of $t$ parties: $P_1, P_2, \ldots, P_t$ that share a common randomness R. $P_1$ has access to $x_1$, and communicates $m_1$ to $P_2$. For all other $i$, $P_i$ has access to $X_i$ and $m_{i-1}$, and communicates $m_i$ to $P_{i+1}$. At the end, $P_t$ communicates $m_t$. It is natural to think of $x_1, \ldots, x_t$ as coming from an input stream, and of $m_t$ as $m_t = f(x_1, \ldots, x_t)$.

We say the model is one way if the communication happens in this form: $P_1 \to P_2 \to \ldots P_t$. We say the model is r-pass one-way if the communication looks like $P_1 \to \ldots P_t \to P_1 \to \ldots P_t \ldots P_1 \to \ldots P_t$, r times.

Observe that: One-way CC $\geq$ r-pass one-way CC $\geq$ arbitrary-CC (exercise).

It is fairly easy to see that *t-pass CC gives a lower bound for trivial algorithms.* Indeed, divide the stream into $t$ pieces. $P_i$ gets the $i^{th}$ piece. Together, the parties simulate a $s$-space algorithm to get $O(s)$ ammortized time for the problem.

### 4.2 t-party lower bounds of t-set-disjointness

We will now define a stronger notion of set disjointness.

[t-set-disjointness]
YES: $x_1, \ldots, x_t \in \{0,1\}^n$
and $\exists\ i$ such that $x_1^{(i)} \ldots x_t^{(i)} = 1$.
and $\forall i' \neq i$, $\sum_{j=1}^{t} x_j^{(i')} \leq 1$, otherwise disjoint.
NO: $x_1, \ldots, x_t \in \{0,1\}^k$, $\forall i$ : $\sum_{j=1}^{t} x_j^{(t)} \leq 1$. \\ strongly disjoint.
Distinguish YES from NO.

We will make use of the following theorem.

**Theorem.** One way CC $\geq \Omega\left(\frac{n}{t^2}\right)$. Arbitrary CC $\geq \Omega\left(\frac{n}{t^3}\right)$

### 4.3 Streaming Lower Bound for $F_3$

Set $t = (2n)^{1/k}$, we will see shortly why this is useful. Given $x_1, \ldots, x_t \in \{0,1\}^k$, we transform it to $a_1 \ldots a_m$ where we list all elements in $x_1$, then all elements of $x_2$ and so on.

YES: $F_k \geq t^k$, i.e. there is some element in all t sets, and therefore $F_k \geq t^k \geq 2n$ by our choice of $t$.

NO: $t_i \leq 1$ for all $i$, which implies $F_k \leq n$.

This gives the desired lower bounds.

## References


[1] Philippe Flajolet and Nigel G. Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences* 31.2:182-209, 1985 http://algo.inria.fr/flajolet/Publications/src/FlMa85.pdf

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences.* 58(1):137147, 1999 https://dl.acm.org/citation.cfm?id=237823

[3] Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences* 68 (2004) 702–732 http://people.seas.harvard.edu/~madhusudan/courses/Spring2016/papers/BJKS.pdf