Frequency moments and Counting Distinct Elements

Lecture 05
September 8, 2020
Part I

Frequency Moments
**Streaming model**

- The input consists of $m$ objects/items/tokens $e_1, e_2, \ldots, e_m$ that are seen one by one by the algorithm.
- The algorithm has “limited” memory say for $B$ tokens where $B < m$ (often $B \ll m$) and hence cannot store all the input.
- Want to compute interesting functions over input.

**Examples:**

- Each token in a number from $[n]$
- High-speed network switch: tokens are packets with source, destination IP addresses and message contents.
- Each token is an edge in graph (graph streams)
- Each token in a point in some feature space
- Each token is a row/column of a matrix
Frequency Moment Problem(s)

- A fundamental class of problems
Frequency Moment Problem(s)

- A fundamental class of problems

Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound).

Example: $n = 5$ and stream is $4, 2, 4, 1, 1, 1, 4, 5$
Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound).

Given a stream let $f_i$ denote the frequency of $i$ or number of times $i$ is seen in the stream.

Consider vector $f = (f_1, f_2, \ldots, f_n)$.

For $k \geq 0$ the $k$’th frequency moment is $F_k = \sum_i f_i^k$. We can also consider the $\ell_k$ norm of $f$ which is $(F_k)^{1/k}$.

Example: $n = 5$ and stream is $4, 2, 4, 1, 1, 1, 4, 5$

$m = 8$ \hfill $f_1 = 3$ \hfill $f_2 = 1$ \hfill $f_4 = 3$ \hfill $f_5 = 1$
Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound).

Given a stream let $f_i$ denote the frequency of $i$ or number of times $i$ is seen in the stream. Consider vector $f = (f_1, f_2, \ldots, f_n)$.

For $k \geq 0$ the $k$’th frequency moment $F_k = \sum_i f_i^k$. Important cases/regimes:

- $k = 0$: $F_0$ is simply the number of distinct elements in the stream.
- $k = 1$: $F_1$ is the length of the stream, which is easy.
- $k = 2$: $F_2$ is fundamental in many ways as we will see.
- $k = 1$: $F_1$ is the maximum frequency (heavy hitters problem).

For $0 < k < 1$ and $1 < k < 2$.
Frequency Moments

- Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound).
- Given a stream let $f_i$ denote the frequency of $i$ or number of times $i$ is seen in the stream. Consider vector $f = (f_1, f_2, \ldots, f_n)$.
- For $k \geq 0$ the $k$’th frequency moment $F_k = \sum_i f_i^k$.

Important cases/regimes:

- $k = 0$: $F_0$ is simply the number of distinct elements in stream
Frequency Moments

- Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound).
- Given a stream let $f_i$ denote the frequency of $i$ or number of times $i$ is seen in the stream. Consider vector $f = (f_1, f_2, \ldots, f_n)$.
- For $k \geq 0$ the $k$’th frequency moment $F_k = \sum_i f_i^k$.

Important cases/regimes:

- $k = 0$: $F_0$ is simply the number of distinct elements in stream.
- $k = 1$: $F_1$ is the length of stream which is easy.
Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound)

Given a stream let $f_i$ denote the frequency of $i$ or number of times $i$ is seen in the stream Consider vector $f = (f_1, f_2, \ldots, f_n)$

For $k \geq 0$ the $k$’th frequency moment $F_k = \sum_i f_i^k$.

Important cases/regimes:

- $k = 0$: $F_0$ is simply the number of distinct elements in stream
- $k = 1$: $F_1$ is the length of stream which is easy
- $k = 2$: $F_2$ is fundamental in many ways as we will see
Frequency Moments

- Stream consists of \( e_1, e_2, \ldots, e_m \) where each \( e_i \) is an integer in \([n]\). We know \( n \) in advance (or an upper bound).
- Given a stream let \( f_i \) denote the frequency of \( i \) or number of times \( i \) is seen in the stream. Consider vector \( f = (f_1, f_2, \ldots, f_n) \).
- For \( k \geq 0 \) the \( k \)'th frequency moment \( F_k = \sum_i f_i^k \).

Important cases/regimes:

- \( k = 0 \): \( F_0 \) is simply the number of *distinct elements* in stream.
- \( k = 1 \): \( F_1 \) is the length of stream which is easy.
- \( k = 2 \): \( F_2 \) is fundamental in many ways as we will see.
- \( k = \infty \): \( F_\infty \) is the maximum frequency (heavy hitters prob).
Frequency Moments

- Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound).
- Given a stream let $f_i$ denote the frequency of $i$ or number of times $i$ is seen in the stream. Consider vector $f = (f_1, f_2, \ldots, f_n)$.
- For $k \geq 0$ the $k$’th frequency moment $F_k = \sum_i f_i^k$.

Important cases/regimes:

- $k = 0$: $F_0$ is simply the number of distinct elements in stream.
- $k = 1$: $F_1$ is the length of stream which is easy.
- $k = 2$: $F_2$ is fundamental in many ways as we will see.
- $k = \infty$: $F_{\infty}$ is the maximum frequency (heavy hitters prob).
- $0 < k < 1$ and $1 < k < 2$
Frequency Moments

- Stream consists of $e_1, e_2, \ldots, e_m$ where each $e_i$ is an integer in $[n]$. We know $n$ in advance (or an upper bound).
- Given a stream let $f_i$ denote the frequency of $i$ or number of times $i$ is seen in the stream. Consider vector $f = (f_1, f_2, \ldots, f_n)$.
- For $k \geq 0$ the $k$’th frequency moment $F_k = \sum_i f_i^k$.

Important cases/regimes:

- $k = 0$: $F_0$ is simply the number of distinct elements in stream.
- $k = 1$: $F_1$ is the length of stream which is easy.
- $k = 2$: $F_2$ is fundamental in many ways as we will see.
- $k = \infty$: $F_\infty$ is the maximum frequency (heavy hitters prob).
- $0 < k < 1$ and $1 < k < 2$
- $2 < k < \infty$
## Frequency Moments: Questions

### Estimation

Given a stream and $k$ can we estimate $F_k$ exactly/approximately with small memory?
## Frequency Moments: Questions

### Estimation

Given a stream and $k$ can we estimate $F_k$ exactly/approximately with small memory?

### Sampling

Given a stream and $k$ can we sample an item $i$ in proportion to $f_i^k$?
Frequency Moments: Questions

**Estimation**
Given a stream and $k$ can we estimate $F_k$ exactly/approximately with small memory?

**Sampling**
Given a stream and $k$ can we sample an item $i$ in proportion to $f_i^k$?

**Sketching**
Given a stream and $k$ can we create a sketch/summary of small size?
Frequency Moments: Questions

**Estimation**
Given a stream and $k$ can we estimate $F_k$ exactly/approximately with small memory?

**Sampling**
Given a stream and $k$ can we sample an item $i$ in proportion to $f_i^k$?

**Sketching**
Given a stream and $k$ can we create a sketch/summary of small size?

Questions easy if we have memory $\Omega(n)$: store $f$ explicitly. Interesting when memory is $\ll n$. Ideally want to do it with $\log^c n$ memory for some fixed $c \geq 1$ (polylog$(n)$). Note that $\log n$ is roughly the memory required to store one token/number.
Need for approximation and randomization

For most of the interesting problems $\Omega(n)$ lower bound on memory if one wants exact answer or wants deterministic algorithms.
Need for approximation and randomization

For most of the interesting problems $\Omega(n)$ lower bound on memory if one wants exact answer or wants deterministic algorithms. Hence

- focus on $(1 \pm \epsilon)$-approximation or constant factor approximation
- and randomized algorithms
Need for approximation and randomization

For most of the interesting problems $\Omega(n)$ lower bound on memory if one wants exact answer or wants deterministic algorithms. Hence

- focus on $(1 \pm \epsilon)$-approximation or constant factor approximation
- and randomized algorithms
Relative approximation

Let $g(\sigma)$ be a real-valued non-negative function over streams $\sigma$.

**Definition**

Let $A(\sigma)$ be the real-valued output of a randomized streaming algorithm on stream $\sigma$. We say that $A$ provides an $(\alpha, \beta)$ relative approximation for a real-valued function $g$ if for all $\sigma$:

$$\Pr \left[ \left| \frac{A(\sigma)}{g(\sigma)} - 1 \right| > \alpha \right] \leq \beta.$$ 

Our ideal goal is to obtain a $(\epsilon, \delta)$-approximation for any given $\epsilon, \delta \in (0, 1)$. 
Additive approximation

Let \( g(\sigma) \) be a real-valued function over streams \( \sigma \). If \( g(\sigma) \) can be negative, focus on additive approximation.

**Definition**

Let \( \mathcal{A}(\sigma) \) be the real-valued output of a randomized streaming algorithm on stream \( \sigma \). We say that \( \mathcal{A} \) provides an \((\alpha, \beta)\) additive approximation for a real-valued function \( g \) if for all \( \sigma \):

\[
\Pr \left[ |\mathcal{A}(\sigma) - g(\sigma)| > \alpha \right] \leq \beta.
\]

When working with additive approximations some normalization/scaling is typically necessary. Our ideal goal is to obtain a \((\epsilon, \delta)\)-approximation for any given \( \epsilon, \delta \in (0, 1) \).
Part II

Estimating Distinct Elements
Distinct Elements

Given a stream $\sigma$ how many distinct elements did we see?

**Example:** in a network switch, during some time window how many distinct destination (or source) IP addresses were seen in the packets?

$$1, 1, 1, 1, \ldots$$
Distinct Elements

Given a stream $\sigma$ how many distinct elements did we see?

**Example:** in a network switch, during some time window how many distinct destination (or source) IP addresses were seen in the packets?

Offline solution?
Distinct Elements

Given a stream $\sigma$ how many distinct elements did we see?

**Example:** in a network switch, during some time window how many distinct destination (or source) IP addresses were seen in the packets?

Offline solution? via Dictionary data structure
DistinctElements

Initialize dictionary $D$ to be empty

$k ← 0$

While (stream is not empty) do

Let $e$ be next item in stream

If ($e ∉ D$) then

Insert $e$ into $D$

$k ← k + 1$

EndWhile

Output $k$
DistinctElements

Initialize dictionary $\mathcal{D}$ to be empty
$k \leftarrow 0$

While (stream is not empty) do
    Let $e$ be next item in stream
    If ($e \not\in \mathcal{D}$) then
        Insert $e$ into $\mathcal{D}$
        $k \leftarrow k + 1$

EndWhile

Output $k$

Which dictionary data structure?
Offline Solution

**DistinctElements**

- Initialize dictionary $\mathcal{D}$ to be empty
- $k \leftarrow 0$
- While (stream is not empty) do
  - Let $e$ be next item in stream
  - If ($e \not\in \mathcal{D}$) then
    - Insert $e$ into $\mathcal{D}$
    - $k \leftarrow k + 1$
- EndWhile
- Output $k$

Which dictionary data structure?
- Binary search trees: space $O(k)$ and total time $O(m \log k)$
- Hashing: space $O(k)$ and expected time $O(m)$. 
Hashing based idea

- Use hash function \( h : [n] \rightarrow [N] \) for some \( N \) polynomial in \( n \).
- Store only the minimum hash value seen. That is \( \min_{e_i} h(e_i) \).
  Need only \( O(\log n) \) bits since numbers are in range \([N]\).

\[
\log N = (\log n)
\]
Hashing based idea

- Use hash function \( h : [n] \rightarrow [N] \) for some \( N \) polynomial in \( n \).
- Store only the minimum hash value seen. That is \( \min_{e_i} h(e_i) \).
  Need only \( O(\log n) \) bits since numbers are in range \([N] \).

**Question:** why is this good?

- Assume *idealized* hash function: \( h : [n] \rightarrow [0, 1] \) that is fully random over the real interval
Hashing based idea

- Use hash function $h : [n] \rightarrow [N]$ for some $N$ polynomial in $n$.
- Store only the minimum hash value seen. That is $\min_{e_i} h(e_i)$. Need only $O(\log n)$ bits since numbers are in range $[N]$.

**Question:** why is this good?

- Assume *idealized* hash function: $h : [n] \rightarrow [0, 1]$ that is fully random over the real interval
- Suppose there are $k$ distinct elements in the stream

$$1, 12, 100, 5, 1, 5, \frac{2}{3}, 100, 5.$$
Hashing based idea

- Use hash function $h : [n] \rightarrow [N]$ for some $N$ polynomial in $n$.
- Store only the minimum hash value seen. That is $\min_{e_i} h(e_i)$. Need only $O(\log n)$ bits since numbers are in range $[N]$.

**Question:** why is this good?

- Assume *idealized* hash function: $h : [n] \rightarrow [0, 1]$ that is fully random over the real interval
- Suppose there are $k$ distinct elements in the stream
- What is the expected value of the minimum of hash values?
Analyzing idealized hash function

**Lemma**

Suppose $X_1, X_2, \ldots, X_k$ are random variables that are independent and uniformly distributed in $[0, 1]$ and let $Y = \min_i X_i$. Then $\mathbb{E}[Y] = \frac{1}{(k+1)}$.

\[ \mathbb{E}[Y] = \int_0^1 \left( \sum_{i=1}^k \frac{(k-1)!}{(k'!)} \frac{(k-1)!}{(k'!)} dt \right) \]

\[ = \left( \sum_{i=1}^k \frac{(k-1)!}{(k'!)} \frac{(k-1)!}{(k'!)} \right) \int_0^t dt = k \int_0^t y^{k-1} dy = \frac{1}{k+1}. \]
Lemma

Suppose $X_1, X_2, \ldots, X_k$ are random variables that are independent and uniformly distributed in $[0, 1]$ and let $Y = \min_i X_i$. Then $E[Y] = \frac{1}{(k+1)}$.

DistinctElements

Assume ideal hash function $h : [n] \rightarrow [0, 1]$

$y \leftarrow 1$

While (stream is not empty) do

Let $e$ be next item in stream

$y \leftarrow \min(y, h(e))$

EndWhile

Output $\frac{1}{y} - 1$
Lemma

Suppose $X_1, X_2, \ldots, X_k$ are random variables that are independent and uniformly distributed in $[0, 1]$ and let $Y = \min_i X_i$. Then

$$E[Y] = \frac{1}{(k+1)}.$$
Analyzing idealized hash function

**Lemma**

Suppose $X_1, X_2, \ldots, X_k$ are random variables that are independent and uniformly distributed in $[0, 1]$ and let $Y = \min_i X_i$. Then

$$E[Y] = \frac{1}{(k+1)}.$$

**Lemma**

Suppose $X_1, X_2, \ldots, X_k$ are random variables that are independent and uniformly distributed in $[0, 1]$ and let $Y = \min_i X_i$. Then

$$E[Y^2] = \frac{2}{(k+1)(k+2)}$$

and

$$\text{Var}(Y) = \frac{k}{(k+1)^2(k+2)} \leq \frac{1}{(k+1)^2}.$$
Analyzing idealized hash function

Apply standard methodology to go from exact statistical estimator to good bounds:

- average $h$ parallel and independent estimates to reduce variance
- apply Chebyshev to show that the average estimator is a $(1 + \epsilon)$-approximation with constant probability
- use preceding and median trick with $O(\log 1/\delta)$ parallel copies to obtain a $(1 + \epsilon)$-approximation with probability $(1 - \delta)$
Averaging and reducing variance

1. Run basic estimator independently and in parallel $h$ times to obtain $X_1, X_2, \ldots, X_h$
2. Let $Z = \frac{1}{h} X_i$
3. Output $\frac{1}{Z} - 1$
Averaging and reducing variance

1. Run basic estimator independently and in parallel $h$ times to obtain $X_1, X_2, \ldots, X_h$
2. Let $Z = \frac{1}{h} X_i$
3. Output $\frac{1}{Z} - 1$

Claim: $E[Z] = \frac{1}{(k+1)}$ and $\text{Var}(Z) \leq \frac{1}{h} \frac{1}{(k+1)^2}$.
Averaging and reducing variance

1. Run basic estimator independently and in parallel $h$ times to obtain $X_1, X_2, \ldots, X_h$
2. Let $Z = \frac{1}{h} X_i$
3. Output $\frac{1}{Z} - 1$

Claim: $E[Z] = \frac{1}{(k+1)}$ and $\text{Var}(Z) \leq \frac{1}{h} \frac{1}{(k+1)^2}$.

Choosing $h = 1/(\eta \epsilon^2)$ and using Chebyshev:

$$\Pr\left[|Z - \frac{1}{k+1}| \geq \frac{\epsilon}{k+1}\right] \leq \eta.$$
Averaging and reducing variance

1. Run basic estimator independently and in parallel \( h \) times to obtain \( X_1, X_2, \ldots, X_h \)
2. Let \( Z = \frac{1}{h} X_i \)
3. Output \( \frac{1}{Z} - 1 \)

Claim: \( \mathbb{E}[Z] = \frac{1}{(k+1)} \) and \( \text{Var}(Z) \leq \frac{1}{h} \frac{1}{(k+1)^2} \).

Choosing \( h = \frac{1}{(\eta \epsilon^2)} \) and using Chebyshev:
\[
\Pr\left[ \left| Z - \frac{1}{k+1} \right| \geq \frac{\epsilon}{k+1} \right] \leq \eta.
\]
Hence \( \Pr\left[ \left| \left( \frac{1}{Z} - 1 \right) - k \right| \geq O(\epsilon)k \leq \eta. \right. \)
Averaging and reducing variance

1. Run basic estimator independently and in parallel $h$ times to obtain $X_1, X_2, \ldots, X_h$
2. Let $Z = \frac{1}{h}X_i$
3. Output $\frac{1}{Z} - 1$

Claim: $E[Z] = \frac{1}{(k+1)}$ and $\text{Var}(Z) \leq \frac{1}{h} \frac{1}{(k+1)^2}$.

Choosing $h = 1/(\eta \epsilon^2)$ and using Chebyshev:

$$\Pr \left[ \left| Z - \frac{1}{k+1} \right| \geq \frac{\epsilon}{k+1} \right] \leq \eta.$$  

Hence $\Pr \left[ \left| (\frac{1}{Z} - 1) - k \right| \right] \geq O(\epsilon) k \leq \eta$.

Repeat $O(\log \frac{1}{\delta})$ times and output median. Error probability $< \delta$. 

Chandra (UIUC)
Algorithm via regular hashing

Do not have idealized hash function.

- Use $h : [n] \rightarrow [N]$ for appropriate choice of $N$
- Use pairwise independent hash family $\mathcal{H}$ so that random $h \in \mathcal{H}$ can be stored in small space and computation can be done in small memory and fast

Several variants of idea with different trade offs between

- memory
- time to process each new element of the stream
- approximation quality and probability of success