Frequency moments and Counting Distinct Elements

Lecture 06
January 31, 2019
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
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 $F_k = \sum_i f_i^k$. We can also consider the $\ell_k$ norm of $f$ which is $(F_k)^{1/k}$.

Example: $n = 5$ and stream is $4, 2, 4, 1, 1, 1, 4, 5$. 
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.
- $k = \infty$: $F_\infty$ is the maximum frequency (heavy hitters problem).

$1 < k < 2$ 

$2 < k < \infty$
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.
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.
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
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)
- $1 < k < 2$
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)
- $1 < k < 2$
- $2 < k < \infty$
Frequency Moments: Questions

**Estimation**

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

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

$$\Pr [ |A(\sigma) - g(\sigma)| > \alpha ] \leq \beta.$$

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?
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 $\mathcal{D}$ to be empty

$k \leftarrow 0$

While (stream is not empty) do

Let $e$ be next item in stream

If ($e \notin \mathcal{D}$) then

Insert $e$ into $\mathcal{D}$

$k \leftarrow k + 1$

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?
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 ($e \notin \mathcal{D}$) 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 $O(n)$ and expected time $O(m)$. 
Hashing based idea

- Use hash function $h : [n] \rightarrow [m]$ for some large $m \gg n$
- 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:

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?
Hashing based idea

- Use hash function $h : [n] \rightarrow [m]$ for some large $m \gg n$
- 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 $h : [n] \rightarrow [m]$ for some large $m \gg n$
- 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
Hashing based idea

- Use hash function \( h : [n] \rightarrow [m] \) for some large \( m \gg n \).
- 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?
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

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

**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 

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

EndWhile

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

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

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

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

$Pr\left[\left|Z - \frac{1}{k+1}\right| \geq \frac{\epsilon}{k+1}\right] \leq \eta$. 
Run basic estimator independently and in parallel \( h \) times to obtain \( X_1, X_2, \ldots, X_h \)

Let \( Z = \frac{1}{h} X_i \)

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 \( h = \frac{1}{(\eta \varepsilon^2)} \) and using Chebyshev:

\[
\Pr\left[ |Z − \frac{1}{k+1}| \geq \frac{\varepsilon}{k+1} \right] \leq \eta.
\]

Hence \( \Pr\left[ |(\frac{1}{Z} − 1) − k| \right] \geq O(\varepsilon)k \leq \eta. \)
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 \( \text{Var}(Z) \leq \frac{1}{h(k+1)^2} \).

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

Repeat \( O(\log 1/\delta) \) times and output median. Error probability < \( \delta \).
Algorithm via regular hashing

Do not have idealized hash function.

- Use $h : [n] \rightarrow [m]$ for large $m$ (say $m = n^3$)
- 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