Frequency moments and Counting Distinct Elements

Lecture 06
January 31, 2019
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
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.
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 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 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.
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.
**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.
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?

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 $D$ 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 $D$ to be empty

$k ← 0$

While (stream is not empty) do

Let $e$ be next item in stream

If ($e \not\in D$) then

Insert $e$ into $D$

$k ← k + 1$

EndWhile

Output $k$

Which dictionary data structure?
DistinctElements

Initialize dictionary $D$ to be empty

$k \leftarrow 0$

While (stream is not empty) do

Let $e$ be next item in stream

If ($e \not\in D$) then

Insert $e$ into $D$

$k \leftarrow k + 1$

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

**DistinctElements**

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

1. $y \leftarrow 1$
2. While (stream is not empty) do
   1. Let $e$ be next item in stream
   2. $y \leftarrow \min(z, h(e))$
3. EndWhile
4. 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

$$E[Y^2] = \frac{2}{(k+1)(k+2)} \quad \text{and} \quad \text{Var}(Y) = \frac{k}{(k+1)^2(k+2)} \leq \frac{1}{(k+1)^2}.$$
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 $O(\log \frac{1}{\delta})$ 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 \( \frac{1}{Z} - 1 \)

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

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

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

Hence

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

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

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

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

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

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

Hence $Pr\left[|\left(\frac{1}{Z} - 1\right) - k| \geq O(\epsilon)k\right] \leq \eta$. 
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: $E[Z] = \frac{1}{(k+1)}$ and $\text{Var}(Z) \leq \frac{1}{h} \left(\frac{1}{(k+1)^2}\right)$.

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

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

Do not have idealized hash function.

- Use $h : [n] \rightarrow [N]$ 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
Algorithm from BJKST

**BJKST-DistinctElements:**
- $\mathcal{H}$ is a 2-universal hash family from $[n]$ to $[N = n^3]$
- choose $h$ at random from $\mathcal{H}$
- $t \leftarrow \frac{c}{\epsilon^2}$
- While (stream is not empty) do
  - $a_i$ is current item
  - Update the smallest $t$ hash values seen so far with $h(a_i)$
- endWhile
- Let $v$ be the $t$’th smallest value seen in the hash values.
- Output $tN/v$. 

**Memory:**
- $t = O(1/\epsilon^2)$ values so $O(\log n/\epsilon^2)$ bits. Also $O(\log n)$ bits to store hash function.

**Processing time per element:**
- $O(\log(1/\epsilon))$ comparisons of $\log n$ bit numbers by using a binary search tree. And computing hash value.
Algorithm from BJKST

**BJKST-DistinctElements:**

- \( \mathcal{H} \) is a 2-universal hash family from \([n]\) to \([N = n^3]\)
- choose \( h \) at random from \( \mathcal{H} \)
- \( t \leftarrow \frac{c}{\epsilon^2} \)

While (stream is not empty) do
  - \( a_i \) is current item
  - Update the smallest \( t \) hash values seen so far with \( h(a_i) \)
endWhile

Let \( v \) be the \( t' \)th smallest value seen in the hash values.
Output \( tN/v \).

- Memory: \( t = O(1/\epsilon^2) \) values so \( O(\log n/\epsilon^2) \) bits. Also \( O(\log n) \) bits to store hash function
- Processing time per element: \( O(\log(1/\epsilon)) \) comparisons of \( \log n \) bit numbers by using a binary search tree. And computing hash value.
Intuition for algorithm/analysis

If $h$ were truly random we expect minimum hash value to be around $\frac{N}{(d + 1)}$

$t$’th minimum hash value $v$ to be around $\frac{tN}{(d + 1)}$.

Hence $\frac{tN}{v}$ should be around $d + 1$

We will assume that $d > c\epsilon^2$ for otherwise we can keep track of the exact count of distinct elements. How?

$t$’th hash value more robust estimator than minimum hash value and incorporates the averaging trick to reduce variance
Analysis

Let $d$ be actual number of distinct values in a given stream (assume $d > c/\epsilon^2$). Let $D$ be the output of the algorithm which is a random variable.

Lemma \[ \Pr[D < (1 - \epsilon)d] \leq \frac{1}{6}. \]

Lemma \[ \Pr[D > (1 + \epsilon)d] \leq \frac{1}{6}. \]

Hence \[ \Pr[|D - d| \geq \epsilon d] < \frac{1}{3}. \]

Can do median trick to reduce error probability to $\delta$ with $O(\log 1/\delta)$ parallel repetitions.
Let $d$ be actual number of distinct values in a given stream (assume $d > c/\epsilon^2$). Let $D$ be the output of the algorithm which is a random variable.

**Lemma**

$$\Pr[D < (1 - \epsilon)d] \leq 1/6.$$  

**Lemma**

$$\Pr[D > (1 + \epsilon)d] \leq 1/6.$$  

Hence $\Pr[|D - d| \geq \epsilon d] < 1/3$. Can do median trick to reduce error probability to $\delta$ with $O(\log 1/\delta)$ parallel repetitions.
Analysis

Lemma

Since $N = n^3$ the probability that there are no collisions in $h$ is at least $1 - 1/n$.

Left as an exercise.

Recall

Lemma

$X = X_1 + X_2 + \ldots + X_k$ where $X_1, X_2, \ldots, X_k$ are pairwise independent. Then $\text{Var}(X) = \sum_i \text{Var}(X_i)$. 
Lemma

\[ \Pr[D < (1 - \epsilon)d] \leq 1/6. \]

Let \( b_1, b_2, \ldots, b_d \) be the distinct values in the stream. Recall \( D = tN/v \) where \( v \) is the \( t \)'th smallest hash value seen.

\( D < (1 - \epsilon)d \) iff \( v > \frac{tN}{(1-\epsilon)d} \). Implies less than \( t \) hash values fell in the interval \([1..\frac{tN}{(1-\epsilon)d}]\).
Let $b_1, b_2, \ldots, b_d$ be the distinct values in the stream. Recall $D = \frac{tN}{v}$ where $v$ is the $t$’th smallest hash value seen.

$D < (1 - \epsilon)d$ iff $v > \frac{tN}{(1 - \epsilon)d}$. Implies less than $t$ hash values fell in the interval $[1..\frac{tN}{(1 - \epsilon)d}]$. What is the probability of this event?

Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1 - \epsilon)d}$.

And $X = \sum_{i=1}^{d} X_i$

$$\Pr[D < (1 - \epsilon)d] = \Pr[X < t].$$
Analysis

Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1-\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$,
$E[X_i] = \Pr[X_i = 1] = \frac{t}{(1-\epsilon)d} \geq (1 + \epsilon)\frac{t}{d}$. (ignoring some rounding issues for clarity of calculations)
Analysis

Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1-\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

- Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$, $\mathbb{E}[X_i] = \Pr[X_i = 1] = \frac{t}{(1-\epsilon)d} \geq \frac{(1 + \epsilon)t}{d}$. (ignoring some rounding issues for clarity of calculations)

- $\mathbb{E}[X] \geq (1 + \epsilon)t$. 

By Chebyshev:

$$\Pr[X < t] \leq \Pr[|X - \mathbb{E}[X]| > \epsilon t] \leq \frac{\text{Var}(X)}{\epsilon^2 t^2} \leq \frac{(1 + 3\epsilon/2)t}{c}.$$ 

Choose $c$ sufficiently large to ensure ratio is at most $1/6$. 

Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1-\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

- Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$, $E[X_i] = \Pr[X_i = 1] = \frac{t}{(1-\epsilon)d} \geq (1 + \epsilon)t/d$. (ignoring some rounding issues for clarity of calculations)
- $E[X] \geq (1 + \epsilon)t$.
- $X_i$ is a binary rv hence $\text{Var}(X_i) \leq E[X_i] \leq (1 + 3\epsilon/2)t/d$. 

By Chebyshev:

$\Pr[X < t] \leq \Pr[|X - E[X]| > \epsilon t] \leq \frac{\text{Var}(X)}{\epsilon^2 t^2} \leq \frac{(1 + 3\epsilon/2)}{c}$.
Analysis

Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1-\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

- Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$, 
  $E[X_i] = \Pr[X_i = 1] = \frac{t}{(1-\epsilon)d} \geq (1 + \epsilon) t / d$. (ignoring some rounding issues for clarity of calculations)

- $E[X] \geq (1 + \epsilon) t$.

- $X_i$ is a binary rv hence $\text{Var}(X_i) \leq E[X_i] \leq (1 + 3\epsilon/2) t / d$.

- $X_1, X_2, \ldots, X_d$ are pair-wise independent random variables hence $\text{Var}(X) = \sum_i \text{Var}(X_i) \leq (1 + 3\epsilon/2) t$. 

By Chebyshev:

$$\Pr[X < t] \leq \Pr[|X - E[X]| > \epsilon t] \leq \frac{\text{Var}(X)}{\epsilon^2 t^2} \leq \frac{1 + 3\epsilon/2}{c}$$

Choose $c$ sufficiently large to ensure ratio is at most $1/6$. 

Chandra (UIUC)  
CS498ABD  
Spring 2019  
25 / 28
Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1-\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

- Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$, $E[X_i] = \Pr[X_i = 1] = \frac{t}{(1-\epsilon)d} \geq (1 + \epsilon) \frac{t}{d}$. (ignoring some rounding issues for clarity of calculations)
- $E[X] \geq (1 + \epsilon) t$.
- $X_i$ is a binary rv hence $Var(X_i) \leq E[X_i] \leq (1 + 3\epsilon/2) \frac{t}{d}$.
- $X_1, X_2, \ldots, X_d$ are pair-wise independent random variables hence $Var(X) = \sum_i Var(X_i) \leq (1 + 3\epsilon/2) t$.

By Chebyshev:

$$Pr[X < t] \leq Pr[|X - E[X]| > \epsilon t] \leq \frac{Var(X)}{\epsilon^2 t^2} \leq \frac{(1 + 3\epsilon/2)}{c}$$
Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1-\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

- Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$, $E[X_i] = Pr[X_i = 1] = \frac{t}{(1-\epsilon)d} \geq (1 + \epsilon)t/d$. (ignoring some rounding issues for clarity of calculations)

- $E[X] \geq (1 + \epsilon)t$.

- $X_i$ is a binary rv hence $Var(X_i) \leq E[X_i] \leq (1 + 3\epsilon/2)t/d$.

- $X_1, X_2, \ldots, X_d$ are pair-wise independent random variables hence $Var(X) = \sum_i Var(X_i) \leq (1 + 3\epsilon/2)t$.

By Chebyshev:

$$Pr[X < t] \leq Pr[|X - E[X]| > \epsilon t] \leq Var(X)/\epsilon^2 t^2 \leq (1 + 3\epsilon/2)/c$$

Choose $c$ sufficiently large to ensure ratio is at most $1/6$. 
Lemma

\[ \text{Pr}[D > (1 + \epsilon)d] \leq 1/6]. \]

Let \( b_1, b_2, \ldots, b_d \) be the distinct values in the stream. Recall \( D = tN/v \) where \( v \) is the \( t \)'th smallest hash value seen.

\( D > (1 + \epsilon)d \) iff \( v < \frac{tN}{(1+\epsilon)d} \). Implies \textit{more than} \( t \) hash values fell in the interval \([1..\frac{tN}{(1+\epsilon)d}]\).
Let \( b_1, b_2, \ldots, b_d \) be the distinct values in the stream. Recall \( D = tN/v \) where \( v \) is the \( t' \)th smallest hash value seen.

\[
D > (1 + \epsilon)d \iff v < \frac{tN}{(1+\epsilon)d}.
\]

Implies more than \( t \) hash values fell in the interval \([1..\frac{tN}{(1+\epsilon)d}]\). What is the probability of this event?

Let \( X_i \) be indicator for \( h(b_i) \leq \frac{tN}{(1+\epsilon)d} \).

And \( X = \sum_{i=1}^{d} X_i \)

\[
\Pr[D > (1 + \epsilon)d] = \Pr[Y > t].
\]
Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1+\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

- Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$, $E[X_i] = \Pr[X_i = 1] = \frac{t}{(1+\epsilon)d} \leq (1 - \epsilon/2)t/d$. (ignoring some rounding issues for clarity of calculations)
- $E[X] \leq (1 - \epsilon/2)t$.
- $X_i$ is a binary rv hence $Var(X_i) \leq E[X_i] \leq (1 - \epsilon/2)t/d$.
- $X_1, X_2, \ldots, X_d$ are pair-wise independent random variables hence $Var(X) = \sum_{i} Var(X_i) \leq (1 - \epsilon/2)t$. 

By Chebyshev:
\[
\Pr[X > t] \leq \Pr[|X - E[X]| > \frac{\epsilon t}{2}] \leq \frac{4 Var(X)}{\epsilon^2 t^2} \leq 4(1 - \epsilon/2)/\epsilon^2 t^2
\]

Choose $c$ sufficiently large to ensure ratio is at most $1/6$. 

Let $X_i$ be indicator for $h(b_i) \leq \frac{tN}{(1+\epsilon)d}$. And $X = \sum_{i=1}^{d} X_i$

- Since $h(b_i)$ is uniformly distributed in $\{1, \ldots, N\}$, $E[X_i] = \Pr[X_i = 1] = \frac{t}{(1+\epsilon)d} \leq (1 - \epsilon/2)t/d$. (ignoring some rounding issues for clarity of calculations)
- $E[X] \leq (1 - \epsilon/2)t$.
- $X_i$ is a binary rv hence $\text{Var}(X_i) \leq E[X_i] \leq (1 - \epsilon/2)t/d$.
- $X_1, X_2, \ldots, X_d$ are pair-wise independent random variables hence $\text{Var}(X) = \sum_i \text{Var}(X_i) \leq (1 - \epsilon/2)t$.

By Chebyshev:

$$\Pr[X > t] \leq \Pr[|X - E[X]| > \epsilon t/2] \leq 4 \text{Var}(X)/\epsilon^2 t^2 \leq 4(1 - \epsilon/2)/c$$
Let \( X_i \) be indicator for \( h(b_i) \leq \frac{tN}{(1+\epsilon)d} \). And \( X = \sum_{i=1}^{d} X_i \)

- Since \( h(b_i) \) is uniformly distributed in \( \{1, \ldots, N\} \),
  \[ E[X_i] = \Pr[X_i = 1] = \frac{t}{(1+\epsilon)d} \leq \frac{1 - \epsilon/2}{t/d}. \] (ignoring some rounding issues for clarity of calculations)

- \( E[X] \leq (1 - \epsilon/2)t \).

- \( X_i \) is a binary rv hence \( \text{Var}(X_i) \leq E[X_i] \leq (1 - \epsilon/2)t/d \).

- \( X_1, X_2, \ldots, X_d \) are pair-wise independent random variables hence \( \text{Var}(X) = \sum_i \text{Var}(X_i) \leq (1 - \epsilon/2)t \).

By Chebyshev:

\[
\Pr[X > t] \leq \Pr[|X - E[X]| > \epsilon t/2] \leq 4 \text{Var}(X)/\epsilon^2 t^2 \leq 4(1 - \epsilon/2)/c
\]

Choose \( c \) sufficiently large to ensure ratio is at most \( 1/6 \).
Summary on Distinct Elements

- with $O\left(\frac{1}{\epsilon^2} \log(1/\delta) \log n\right)$ bits algorithm output estimate $D$ such that $|D - d| \leq \epsilon d$ with probability at least $(1 - \delta)$

- Best known memory bound: $O\left(\frac{\log(1/\delta)}{\epsilon^2} + \log n\right)$ bits and for any fixed $\delta$ this meets lower bound within constant factors. Both lower bound and upper bound quite technical — potential reading for projects.

- Continuous monitoring: want estimate to be correct not only at end of stream but also at all intermediate steps. Can be done with $O\left(\frac{\log \log n + \log(1/\delta)}{\epsilon^2} + \log n\right)$ bits.

- Deletions allowed! Can also be done. More on this later.