# Crash Course on Data Stream Algorithms

## Part I: Basic Definitions and Numerical Streams

Andrew McGregor
University of Massachusetts Amherst

# Goals of the Crash Course

- *Goal:* Give a flavor for the theoretical results and techniques from the 100's of papers on the design and analysis of stream algorithms.

# Goals of the Crash Course

- *Goal:* Give a flavor for the theoretical results and techniques from the 100's of papers on the design and analysis of stream algorithms.

*"When we abstract away the application-specific details, what are the basic algorithmic ideas and challenges in stream processing? What is and isn't possible?"*

# Goals of the Crash Course

- *Goal:* Give a flavor for the theoretical results and techniques from the 100's of papers on the design and analysis of stream algorithms.

*"When we abstract away the application-specific details, what are the basic algorithmic ideas and challenges in stream processing? What is and isn't possible?"*

- *Disclaimer:* Talks will be theoretical/mathematical but shouldn't require much in the way of prerequisites.

# Goals of the Crash Course

- *Goal:* Give a flavor for the theoretical results and techniques from the 100's of papers on the design and analysis of stream algorithms.

  *"When we abstract away the application-specific details, what are the basic algorithmic ideas and challenges in stream processing? What is and isn't possible?"*

- *Disclaimer:* Talks will be theoretical/mathematical but shouldn't require much in the way of prerequisites.
- *Request:*

# Goals of the Crash Course

- *Goal:* Give a flavor for the theoretical results and techniques from the 100's of papers on the design and analysis of stream algorithms.

*"When we abstract away the application-specific details, what are the basic algorithmic ideas and challenges in stream processing? What is and isn't possible?"*

- *Disclaimer:* Talks will be theoretical/mathematical but shouldn't require much in the way of prerequisites.
- *Request:*
  - If you get bored, ask questions. . .

# Goals of the Crash Course

- *Goal:* Give a flavor for the theoretical results and techniques from the 100's of papers on the design and analysis of stream algorithms.

  *"When we abstract away the application-specific details, what are the basic algorithmic ideas and challenges in stream processing? What is and isn't possible?"*

- *Disclaimer:* Talks will be theoretical/mathematical but shouldn't require much in the way of prerequisites.
- *Request:*
  - If you get bored, ask questions...
  - If you get lost, ask questions...

# Goals of the Crash Course

- *Goal:* Give a flavor for the theoretical results and techniques from the 100's of papers on the design and analysis of stream algorithms.

*"When we abstract away the application-specific details, what are the basic algorithmic ideas and challenges in stream processing? What is and isn't possible?"*

- *Disclaimer:* Talks will be theoretical/mathematical but shouldn't require much in the way of prerequisites.
- *Request:*
  - If you get bored, ask questions...
  - If you get lost, ask questions...
  - If you'd like to ask questions, ask questions...

# Outline

# Outline

# Data Stream Model

- *Stream:* $m$ elements from universe of size $n$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad 3, 5, 3, 7, 5, 4, \ldots$$

# Data Stream Model

- *Stream:* $m$ elements from universe of size $n$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad 3, 5, 3, 7, 5, 4, \ldots$$

- *Goal:* Compute a function of stream, e.g., median, number of distinct elements, longest increasing sequence.

# Data Stream Model

- *Stream:* $m$ elements from universe of size $n$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad 3, 5, 3, 7, 5, 4, \ldots$$

- *Goal:* Compute a function of stream, e.g., median, number of distinct elements, longest increasing sequence.
- *Catch:*
  1. Limited working memory, sublinear in $n$ and $m$

# Data Stream Model

- *Stream:* $m$ elements from universe of size $n$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad 3, 5, 3, 7, 5, 4, \ldots$$

- *Goal:* Compute a function of stream, e.g., median, number of distinct elements, longest increasing sequence.
- *Catch:*
  1. Limited working memory, sublinear in $n$ and $m$
  2. Access data sequentially

# Data Stream Model

- *Stream:* $m$ elements from universe of size $n$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad 3, 5, 3, 7, 5, 4, \ldots$$

- *Goal:* Compute a function of stream, e.g., median, number of distinct elements, longest increasing sequence.
- *Catch:*
  1. Limited working memory, sublinear in $n$ and $m$
  2. Access data sequentially
  3. Process each element quickly

# Data Stream Model

- *Stream:* $m$ elements from universe of size $n$, e.g.,

$$\langle x_1, x_2, \ldots, x_m \rangle \quad = \quad 3, 5, 3, 7, 5, 4, \ldots$$

- *Goal:* Compute a function of stream, e.g., median, number of distinct elements, longest increasing sequence.
- *Catch:*
  1. Limited working memory, sublinear in $n$ and $m$
  2. Access data sequentially
  3. Process each element quickly
- Origins in 70s but has become popular in last ten years because of growing theory and very applicable.

# Why's it become popular?

- *Practical Appeal:*
  - Faster networks, cheaper data storage, ubiquitous data-logging results in massive amount of data to be processed.
  - Applications to network monitoring, query planning, I/O efficiency for massive data, sensor networks aggregation...

# Why's it become popular?

- *Practical Appeal:*
  - Faster networks, cheaper data storage, ubiquitous data-logging results in massive amount of data to be processed.
  - Applications to network monitoring, query planning, I/O efficiency for massive data, sensor networks aggregation...
- *Theoretical Appeal:*
  - Easy to state problems but hard to solve.
  - Links to communication complexity, compressed sensing, embeddings, pseudo-random generators, approximation...

# Outline

# Sampling and Statistics

- Sampling is a general technique for tackling massive amounts of data

# Sampling and Statistics

- Sampling is a general technique for tackling massive amounts of data
- *Example:* To compute the median packet size of some IP packets, we could just sample some and use the median of the sample as an estimate for the true median. Statistical arguments relate the size of the sample to the accuracy of the estimate.

# Sampling and Statistics

- Sampling is a general technique for tackling massive amounts of data
- *Example:* To compute the median packet size of some IP packets, we could just sample some and use the median of the sample as an estimate for the true median. Statistical arguments relate the size of the sample to the accuracy of the estimate.
- *Challenge:* But how do you take a sample from a stream of unknown length or from a "sliding window"?

# Reservoir Sampling

- *Problem:* Find uniform sample $s$ from a stream of unknown length

# Reservoir Sampling

- *Problem:* Find uniform sample $s$ from a stream of unknown length
- *Algorithm:*
  - Initially $s = x_1$
  - On seeing the $t$-th element, $s \leftarrow x_t$ with probability $1/t$

# Reservoir Sampling

- *Problem:* Find uniform sample $s$ from a stream of unknown length
- *Algorithm:*
  - Initially $s = x_1$
  - On seeing the $t$-th element, $s \leftarrow x_t$ with probability $1/t$
- *Analysis:*
  - What's the probability that $s = x_i$ at some time $t \geq i$?

# Reservoir Sampling

- *Problem:* Find uniform sample $s$ from a stream of unknown length
- *Algorithm:*
  - Initially $s = x_1$
  - On seeing the $t$-th element, $s \leftarrow x_t$ with probability $1/t$
- *Analysis:*
  - What's the probability that $s = x_i$ at some time $t \geq i$?

$$\mathbb{P}\left[s = x_i\right] = \frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \ldots \times \left(1 - \frac{1}{t}\right) = \frac{1}{t}$$

# Reservoir Sampling

- *Problem:* Find uniform sample $s$ from a stream of unknown length
- *Algorithm:*
  - Initially $s = x_1$
  - On seeing the $t$-th element, $s \leftarrow x_t$ with probability $1/t$
- *Analysis:*
  - What's the probability that $s = x_i$ at some time $t \geq i$?

$$\mathbb{P}[s = x_i] = \frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \ldots \times \left(1 - \frac{1}{t}\right) = \frac{1}{t}$$

  - To get $k$ samples we use $O(k \log n)$ bits of space.

# Priority Sampling for Sliding Windows

- *Problem:* Maintain a uniform sample from the last $w$ items

# Priority Sampling for Sliding Windows

- *Problem:* Maintain a uniform sample from the last $w$ items
- *Algorithm:*
  1. For each $x_i$ we pick a random value $v_i \in (0, 1)$

# Priority Sampling for Sliding Windows

- *Problem:* Maintain a uniform sample from the last $w$ items
- *Algorithm:*
  1. For each $x_i$ we pick a random value $v_i \in (0, 1)$
  2. In a window $\langle x_{j-w+1}, \ldots, x_j \rangle$ return value $x_i$ with smallest $v_i$

# Priority Sampling for Sliding Windows

- *Problem:* Maintain a uniform sample from the last $w$ items
- *Algorithm:*
  1. For each $x_i$ we pick a random value $v_i \in (0, 1)$
  2. In a window $\langle x_{j-w+1}, \ldots, x_j \rangle$ return value $x_i$ with smallest $v_i$
  3. To do this, maintain set of all elements in sliding window whose $v$ value is minimal among subsequent values

# Priority Sampling for Sliding Windows

- *Problem:* Maintain a uniform sample from the last $w$ items
- *Algorithm:*
  1. For each $x_i$ we pick a random value $v_i \in (0, 1)$
  2. In a window $\langle x_{j-w+1}, \ldots, x_j \rangle$ return value $x_i$ with smallest $v_i$
  3. To do this, maintain set of all elements in sliding window whose $v$ value is minimal among subsequent values
- *Analysis:*

# Priority Sampling for Sliding Windows

- *Problem:* Maintain a uniform sample from the last $w$ items
- *Algorithm:*
  1. For each $x_i$ we pick a random value $v_i \in (0, 1)$
  2. In a window $\langle x_{j-w+1}, \ldots, x_j \rangle$ return value $x_i$ with smallest $v_i$
  3. To do this, maintain set of all elements in sliding window whose $v$ value is minimal among subsequent values
- *Analysis:*
  - The probability that $j$-th oldest element is in $S$ is $1/j$ so the expected number of items in $S$ is

$$1/w + 1/(w-1) + \ldots + 1/1 = O(\log w)$$

# Priority Sampling for Sliding Windows

- *Problem:* Maintain a uniform sample from the last $w$ items
- *Algorithm:*
  1. For each $x_i$ we pick a random value $v_i \in (0, 1)$
  2. In a window $\langle x_{j-w+1}, \ldots, x_j \rangle$ return value $x_i$ with smallest $v_i$
  3. To do this, maintain set of all elements in sliding window whose $v$ value is minimal among subsequent values
- *Analysis:*
  - The probability that $j$-th oldest element is in $S$ is $1/j$ so the expected number of items in $S$ is

$$1/w + 1/(w-1) + \ldots + 1/1 = O(\log w)$$

  - Hence, algorithm only uses $O(\log w \log n)$ bits of memory.

# Other Types of Sampling

- *Universe sampling:* For a random $i \in_R [n]$, compute

$$f_i = |\{j : x_j = i\}|$$

# Other Types of Sampling

- *Universe sampling:* For a random $i \in_R [n]$, compute

$$f_i = |\{j : x_j = i\}|$$

- *Minwise hashing:* Sample $i \in_R \{i : \text{there exists } j \text{ such that } x_j = i\}$

# Other Types of Sampling

- *Universe sampling:* For a random $i \in_R [n]$, compute

$$f_i = |\{j : x_j = i\}|$$

- *Minwise hashing:* Sample $i \in_R \{i : \text{there exists } j \text{ such that } x_j = i\}$
- *AMS sampling:* Sample $x_j$ for $j \in_R [m]$ and compute

$$r = |\{j' \geq j : x_{j'} = x_j\}|$$

# Other Types of Sampling

- *Universe sampling:* For a random $i \in_R [n]$, compute

$$f_i = |\{j : x_j = i\}|$$

- *Minwise hashing:* Sample $i \in_R \{i : \text{there exists } j \text{ such that } x_j = i\}$
- *AMS sampling:* Sample $x_j$ for $j \in_R [m]$ and compute

$$r = |\{j' \geq j : x_{j'} = x_j\}|$$

  Handy when estimating quantities like $\sum_i g(f_i)$ because

$$\mathbb{E}\left[m(g(r) - g(r-1))\right] = \sum_i g(f_i)$$

# Outline

# Sketching

- Sketching is another general technique for processing streams

# Sketching

- Sketching is another general technique for processing streams
- Basic idea: Apply a linear projection "on the fly" that takes high-dimensional data to a smaller dimensional space. Post-process lower dimensional image to estimate the quantities of interest.

# Estimating the difference between two streams

- *Input:* Stream from two sources $\langle x_1, x_2, \ldots, x_m \rangle \in ([n] \cup [n])^m$

# Estimating the difference between two streams

- *Input:* Stream from two sources $\langle x_1, x_2, \ldots, x_m \rangle \in ([n] \cup [n])^m$
- *Goal:* Estimate difference between distribution of red values and blue values, e.g.,

$$\sum_{i \in [n]} |f_i - g_i|$$

where $f_i = |\{k : x_k = i\}|$ and $g_i = |\{k : x_k = i\}|$

# $p$-Stable Distributions and Algorithm

- *Defn:* A $p$-stable distribution $\mu$ has the following property:

$$\text{for } X, Y, Z \sim \mu \text{ and } a, b \in \mathbb{R}: \qquad aX + bY \sim (|a|^p + |b|^p)^{1/p} Z$$

e.g., Gaussian is 2-stable and Cauchy distribution is 1-stable

# $p$-Stable Distributions and Algorithm

- *Defn:* A $p$-stable distribution $\mu$ has the following property:

$$\text{for } X, Y, Z \sim \mu \text{ and } a, b \in \mathbb{R}: \qquad aX + bY \sim (|a|^p + |b|^p)^{1/p} Z$$

  e.g., Gaussian is 2-stable and Cauchy distribution is 1-stable
- *Algorithm:*
  - Generate random matrix $A \in \mathbb{R}^{k \times n}$ where $A_{ij} \sim$ Cauchy, $k = O(\epsilon^{-2})$.

# $p$-Stable Distributions and Algorithm

- *Defn:* A $p$-stable distribution $\mu$ has the following property:

  $$\text{for } X, Y, Z \sim \mu \text{ and } a, b \in \mathbb{R} : \qquad aX + bY \sim (|a|^p + |b|^p)^{1/p} Z$$

  e.g., Gaussian is 2-stable and Cauchy distribution is 1-stable
- *Algorithm:*
  - Generate random matrix $A \in \mathbb{R}^{k \times n}$ where $A_{ij} \sim$ Cauchy, $k = O(\epsilon^{-2})$.
  - Compute sketches $Af$ and $Ag$ incrementally

# $p$-Stable Distributions and Algorithm

- *Defn:* A $p$-stable distribution $\mu$ has the following property:

$$\text{for } X, Y, Z \sim \mu \text{ and } a, b \in \mathbb{R}: \quad aX + bY \sim (|a|^p + |b|^p)^{1/p} Z$$

  e.g., Gaussian is 2-stable and Cauchy distribution is 1-stable
- *Algorithm:*
  - Generate random matrix $A \in \mathbb{R}^{k \times n}$ where $A_{ij} \sim$ Cauchy, $k = O(\epsilon^{-2})$.
  - Compute sketches $Af$ and $Ag$ incrementally
  - Return median$(|t_1|, \ldots, |t_k|)$ where $t = Af - Ag$

# $p$-Stable Distributions and Algorithm

- *Defn:* A $p$-stable distribution $\mu$ has the following property:

$$\text{for } X, Y, Z \sim \mu \text{ and } a, b \in \mathbb{R}: \qquad aX + bY \sim (|a|^p + |b|^p)^{1/p} Z$$

  e.g., Gaussian is 2-stable and Cauchy distribution is 1-stable
- *Algorithm:*
  - Generate random matrix $A \in \mathbb{R}^{k \times n}$ where $A_{ij} \sim$ Cauchy, $k = O(\epsilon^{-2})$.
  - Compute sketches $Af$ and $Ag$ incrementally
  - Return median$(|t_1|, \ldots, |t_k|)$ where $t = Af - Ag$
- *Analysis:*
  - By the 1-stability property for $Z_i \sim$ Cauchy

$$|t_i| = |\sum_j A_{i,j}(f_j - g_j)| \sim |Z_i| \sum_j |f_j - g_j|$$

# $p$-Stable Distributions and Algorithm

- *Defn:* A $p$-stable distribution $\mu$ has the following property:

$$\text{for } X, Y, Z \sim \mu \text{ and } a, b \in \mathbb{R}: \qquad aX + bY \sim (|a|^p + |b|^p)^{1/p} Z$$

  e.g., Gaussian is 2-stable and Cauchy distribution is 1-stable
- *Algorithm:*
  - Generate random matrix $A \in \mathbb{R}^{k \times n}$ where $A_{ij} \sim$ Cauchy, $k = O(\epsilon^{-2})$.
  - Compute sketches $Af$ and $Ag$ incrementally
  - Return $\text{median}(|t_1|, \ldots, |t_k|)$ where $t = Af - Ag$
- *Analysis:*
  - By the 1-stability property for $Z_i \sim$ Cauchy

$$|t_i| = |\sum_j A_{i,j}(f_j - g_j)| \sim |Z_i| \sum_j |f_j - g_j|$$

  - For $k = O(\epsilon^{-2})$, since $\text{median}(|Z_i|) = 1$, with high probability,

$$(1 - \epsilon) \sum_j |f_j - g_j| \leq \text{median}(|t_1|, \ldots, |t_k|) \leq (1 + \epsilon) \sum_j |f_j - g_j|$$

# A Useful Multi-Purpose Sketch: Count-Min Sketch

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$
- *Range Sums:* Estimate $\sum_{i \leq k \leq j} f_k$ when $i, j$ aren't known in advance

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$
- *Range Sums:* Estimate $\sum_{i \leq k \leq j} f_k$ when $i, j$ aren't known in advance
- *Find k-Quantiles:* Find values $q_0, \ldots, q_k$ such that

$$q_0 = 0, \quad q_k = n, \quad \text{and} \quad \sum_{i \leq q_j - 1} f_i < \frac{jm}{k} \leq \sum_{i \leq q_j} f_i$$

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$
- *Range Sums:* Estimate $\sum_{i \leq k \leq j} f_k$ when $i, j$ aren't known in advance
- *Find k-Quantiles:* Find values $q_0, \ldots, q_k$ such that

$$q_0 = 0, \quad q_k = n, \quad \text{and} \quad \sum_{i \leq q_j - 1} f_i < \frac{jm}{k} \leq \sum_{i \leq q_j} f_i$$

- *Algorithm:* Count-Min Sketch
  - Maintain an array of counters $c_{i,j}$ for $i \in [d]$ and $j \in [w]$

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$
- *Range Sums:* Estimate $\sum_{i \leq k \leq j} f_k$ when $i, j$ aren't known in advance
- *Find k-Quantiles:* Find values $q_0, \ldots, q_k$ such that

$$q_0 = 0, \quad q_k = n, \quad \text{and} \quad \sum_{i \leq q_j - 1} f_i < \frac{jm}{k} \leq \sum_{i \leq q_j} f_i$$

- *Algorithm:* Count-Min Sketch
  - Maintain an array of counters $c_{i,j}$ for $i \in [d]$ and $j \in [w]$
  - Construct $d$ random hash functions $h_1, h_2, \ldots h_d : [n] \to [w]$

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$
- *Range Sums:* Estimate $\sum_{i \leq k \leq j} f_k$ when $i, j$ aren't known in advance
- *Find k-Quantiles:* Find values $q_0, \ldots, q_k$ such that

$$q_0 = 0, \quad q_k = n, \quad \text{and} \quad \sum_{i \leq q_j - 1} f_i < \frac{jm}{k} \leq \sum_{i \leq q_j} f_i$$

- *Algorithm:* Count-Min Sketch
  - Maintain an array of counters $c_{i,j}$ for $i \in [d]$ and $j \in [w]$
  - Construct $d$ random hash functions $h_1, h_2, \ldots h_d : [n] \to [w]$
  - Update counters: On seeing value $v$, increment $c_{i,h_i(v)}$ for $i \in [d]$

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$
- *Range Sums:* Estimate $\sum_{i \leq k \leq j} f_k$ when $i, j$ aren't known in advance
- *Find k-Quantiles:* Find values $q_0, \ldots, q_k$ such that

$$q_0 = 0, \quad q_k = n, \quad \text{and} \quad \sum_{i \leq q_j - 1} f_i < \frac{jm}{k} \leq \sum_{i \leq q_j} f_i$$

- *Algorithm:* Count-Min Sketch
  - Maintain an array of counters $c_{i,j}$ for $i \in [d]$ and $j \in [w]$
  - Construct $d$ random hash functions $h_1, h_2, \ldots h_d : [n] \to [w]$
  - Update counters: On seeing value $v$, increment $c_{i,h_i(v)}$ for $i \in [d]$
  - To get an estimate of $f_k$, return

$$\tilde{f}_k = \min_i c_{i,h_i(k)}$$

# A Useful Multi-Purpose Sketch: Count-Min Sketch

- *Heavy Hitters:* Find all $i$ such that $f_i \geq \phi m$
- *Range Sums:* Estimate $\sum_{i \leq k \leq j} f_k$ when $i, j$ aren't known in advance
- *Find k-Quantiles:* Find values $q_0, \ldots, q_k$ such that

$$q_0 = 0, \quad q_k = n, \quad \text{and} \quad \sum_{i \leq q_j - 1} f_i < \frac{jm}{k} \leq \sum_{i \leq q_j} f_i$$

- *Algorithm:* Count-Min Sketch
  - Maintain an array of counters $c_{i,j}$ for $i \in [d]$ and $j \in [w]$
  - Construct $d$ random hash functions $h_1, h_2, \ldots h_d : [n] \to [w]$
  - Update counters: On seeing value $v$, increment $c_{i,h_i(v)}$ for $i \in [d]$
  - To get an estimate of $f_k$, return

$$\tilde{f}_k = \min_i c_{i,h_i(k)}$$

- *Analysis:* For $d = O(\log 1/\delta)$ and $w = O(1/\epsilon^2)$

$$\mathbb{P}\left[f_k - \epsilon m \leq \tilde{f}_k \leq f_k\right] \geq 1 - \delta$$

# Outline

# Counting Distinct Elements

- *Input:* Stream $\langle x_1, x_2, \ldots, x_m \rangle \in [n]^m$
- *Goal:* Estimate the number of distinct values in the stream up to a multiplicative factor $(1 + \epsilon)$ with high probability.

# Algorithm

- *Algorithm:*
  1. Apply random hash function $h : [n] \longrightarrow [0, 1]$ to each element

# Algorithm

- *Algorithm:*
  1. Apply random hash function $h : [n] \to [0, 1]$ to each element
  2. Compute $\phi$, the $t$-th smallest value of the hash seen where $t = 21/\epsilon^2$

# Algorithm

- *Algorithm:*
  1. Apply random hash function $h : [n] \rightarrow [0, 1]$ to each element
  2. Compute $\phi$, the $t$-th smallest value of the hash seen where $t = 21/\epsilon^2$
  3. Return $\tilde{r} = t/\phi$ as estimate for $r$, the number of distinct items.

# Algorithm

- *Algorithm:*
  1. Apply random hash function $h : [n] \to [0, 1]$ to each element
  2. Compute $\phi$, the $t$-th smallest value of the hash seen where $t = 21/\epsilon^2$
  3. Return $\tilde{r} = t/\phi$ as estimate for $r$, the number of distinct items.
- *Analysis:*
  1. Algorithm uses $O(\epsilon^{-2} \log n)$ bits of space.

# Algorithm

- *Algorithm:*
  1. Apply random hash function $h : [n] \to [0, 1]$ to each element
  2. Compute $\phi$, the $t$-th smallest value of the hash seen where $t = 21/\epsilon^2$
  3. Return $\tilde{r} = t/\phi$ as estimate for $r$, the number of distinct items.
- *Analysis:*
  1. Algorithm uses $O(\epsilon^{-2} \log n)$ bits of space.
  2. We'll show estimate has good accuracy with reasonable probability

$$\mathbb{P}\left[|\tilde{r} - r| \leq \epsilon r\right] \leq 9/10$$

# Accuracy Analysis

1. Suppose the distinct items are $a_1, \ldots, a_r$

# Accuracy Analysis

1. Suppose the distinct items are $a_1, \ldots, a_r$
2. *Over Estimation:*

$$\mathbb{P}\left[\tilde{r} \geq (1+\epsilon)r\right] = \mathbb{P}\left[t/\phi \geq (1+\epsilon)r\right] = \mathbb{P}\left[\phi \leq \frac{t}{r(1+\epsilon)}\right]$$

# Accuracy Analysis

1. Suppose the distinct items are $a_1, \ldots, a_r$
2. *Over Estimation:*

$$\mathbb{P}\left[\tilde{r} \geq (1+\epsilon)r\right] = \mathbb{P}\left[t/\phi \geq (1+\epsilon)r\right] = \mathbb{P}\left[\phi \leq \frac{t}{r(1+\epsilon)}\right]$$

3. Let $X_i = 1[h(a_i) \leq \frac{t}{r(1+\epsilon)}]$ and $X = \sum X_i$

$$\mathbb{P}\left[\phi \leq \frac{t}{r(1+\epsilon)}\right] = \mathbb{P}\left[X > t\right] = \mathbb{P}\left[X > (1+\epsilon)\mathbb{E}\left[X\right]\right]$$

# Accuracy Analysis

1. Suppose the distinct items are $a_1, \ldots, a_r$
2. *Over Estimation:*

$$\mathbb{P}\left[\tilde{r} \geq (1+\epsilon)r\right] = \mathbb{P}\left[t/\phi \geq (1+\epsilon)r\right] = \mathbb{P}\left[\phi \leq \frac{t}{r(1+\epsilon)}\right]$$

3. Let $X_i = 1[h(a_i) \leq \frac{t}{r(1+\epsilon)}]$ and $X = \sum X_i$

$$\mathbb{P}\left[\phi \leq \frac{t}{r(1+\epsilon)}\right] = \mathbb{P}\left[X > t\right] = \mathbb{P}\left[X > (1+\epsilon)\mathbb{E}\left[X\right]\right]$$

4. By a Chebyshev analysis,

$$\mathbb{P}\left[X > (1+\epsilon)\mathbb{E}\left[X\right]\right] \leq \frac{1}{\epsilon^2 \mathbb{E}\left[X\right]} \leq 1/20$$

# Accuracy Analysis

1. Suppose the distinct items are $a_1, \ldots, a_r$
2. *Over Estimation:*

$$\mathbb{P}[\tilde{r} \geq (1+\epsilon)r] = \mathbb{P}[t/\phi \geq (1+\epsilon)r] = \mathbb{P}\left[\phi \leq \frac{t}{r(1+\epsilon)}\right]$$

3. Let $X_i = 1[h(a_i) \leq \frac{t}{r(1+\epsilon)}]$ and $X = \sum X_i$

$$\mathbb{P}\left[\phi \leq \frac{t}{r(1+\epsilon)}\right] = \mathbb{P}[X > t] = \mathbb{P}[X > (1+\epsilon)\mathbb{E}[X]]$$

4. By a Chebyshev analysis,

$$\mathbb{P}[X > (1+\epsilon)\mathbb{E}[X]] \leq \frac{1}{\epsilon^2 \mathbb{E}[X]} \leq 1/20$$

5. *Under Estimation:* A similar analysis shows $\mathbb{P}[\tilde{r} \leq (1-\epsilon)r] \leq 1/20$

# Outline

# Some Other Results

*Correlations:*

- Input: $\langle (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \rangle$
- Goal: Estimate strength of correlation between $x$ and $y$ via the distance between joint distribution and product of the marginals.
- Result: $(1 + \epsilon)$ approx in $\tilde{O}(\epsilon^{-O(1)})$ space.

# Some Other Results

*Correlations:*

- Input: $\langle (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \rangle$
- Goal: Estimate strength of correlation between $x$ and $y$ via the distance between joint distribution and product of the marginals.
- Result: $(1+\epsilon)$ approx in $\tilde{O}(\epsilon^{-O(1)})$ space.

*Linear Regression:*

- Input: Stream defines a matrix $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^{d \times 1}$
- Goal: Find $x$ such that $\|Ax - b\|_2$ is minimized.
- Result: $(1+\epsilon)$ estimation in $\tilde{O}(d^2 \epsilon^{-1})$ space.

# Some More Other Results

*Histograms:*

- Input: $\langle x_1, x_2, \ldots, x_m \rangle \in [n]^m$
- Goal: Determine $B$ bucket histogram $H : [m] \to \mathbb{R}$ minimizing

$$\sum_{i \in [m]} (x_i - H(i))^2$$

- Result: $(1 + \epsilon)$ estimation in $\tilde{O}(B^2 \epsilon^{-1})$ space

# Some More Other Results

*Histograms:*

- Input: $\langle x_1, x_2, \ldots, x_m \rangle \in [n]^m$
- Goal: Determine $B$ bucket histogram $H : [m] \to \mathbb{R}$ minimizing

$$\sum_{i \in [m]} (x_i - H(i))^2$$

- Result: $(1 + \epsilon)$ estimation in $\tilde{O}(B^2 \epsilon^{-1})$ space

*Transpositions and Increasing Subsequences:*

- Input: $\langle x_1, x_2, \ldots, x_m \rangle \in [n]^m$
- Goal: Estimate number of transpositions $|\{i < j : x_i > x_j\}|$
- Goal: Estimate length of longest increasing subsequence
- Results: $(1 + \epsilon)$ approx in $\tilde{O}(\epsilon^{-1})$ and $\tilde{O}(\epsilon^{-1}\sqrt{n})$ space respectively

# Thanks!

- *Blog:* `http://polylogblog.wordpress.com`
- *Lectures:* Piotr Indyk, MIT

  `http://stellar.mit.edu/S/course/6/fa07/6.895/`
- *Books:*

  "Data Streams: Algorithms and Applications"
  S. Muthukrishnan (2005)

  "Algorithms and Complexity of Stream Processing"
  A. McGregor, S. Muthukrishnan (forthcoming)