COMPSCI 514: Algorithms for Data Science

Cameron Musco
University of Massachusetts Amherst. Fall 2022.
Lecture 8

Summary

Last Class:

- Finish up Bloom filter analysis and optimization of parameters.
- Start on streaming algorithms and distinct elements estimation via hashing.

Summary

Last Class:

- Finish up Bloom filter analysis and optimization of parameters.
- Start on streaming algorithms and distinct elements estimation via hashing.

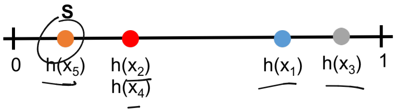
This Class:

- Analysis of the distinct elements algorithm.
- The median trick for boosting success probability.
- Sketch of the ideas behind practical algorithms for distinct elements estimation.

Hashing for Distinct Elements

Min-Hashing for Distinct Elements: 2 4 4 1 1 3 5 5

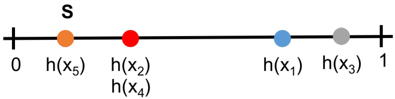
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Hashing for Distinct Elements

Min-Hashing for Distinct Elements:

- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$

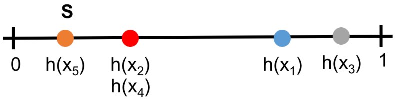


- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.

Hashing for Distinct Elements

Min-Hashing for Distinct Elements:

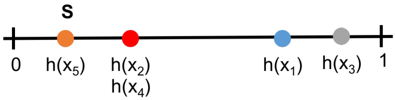
- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.
- Intuition: The larger d is, the smaller we expect s to be.

Performance in Expectation

s is the minimum of d points chosen uniformly at random on $[0, 1]$.
Where $d = \#$ distinct elements.



Performance in Expectation

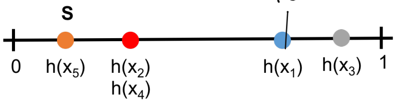
s is the minimum of d points chosen uniformly at random on $[0, 1]$.
Where $d = \#$ distinct elements.

$\mathbb{E}s = \frac{1}{2}$

$d = 1$

$\mathbb{E}s = \frac{1}{3}$ $d = 2$

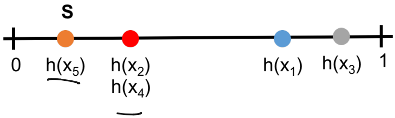
2/3



$d = 1$

$$\mathbb{E}[s] = \frac{1}{d+1}$$

s is the minimum of d points chosen uniformly at random on $[0, 1]$.
Where $d = \#$ distinct elements.



non-negative

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^{\infty} \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

in range

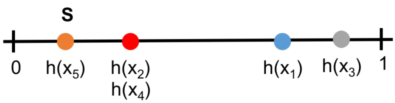
$$\mathbb{E}X = \sum_s s \cdot \Pr(X = s)$$

$$= \sum_s \Pr(X \geq s)$$

in range

Performance in Expectation

s is the minimum of d points chosen uniformly at random on $[0, 1]$.
Where $d = \#$ distinct elements.



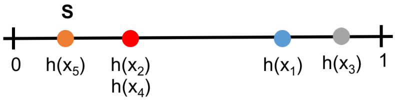
$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

- So our estimate $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ is correct if **s** exactly equals its expectation.

$$\frac{1}{\frac{1}{d+1}} - 1 = d$$

Performance in Expectation

s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.



$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \quad (\text{using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus})$$

- So our estimate $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ is correct if **s** exactly equals its expectation. Does this mean $\mathbb{E}[\widehat{\mathbf{d}}] = d$? No!

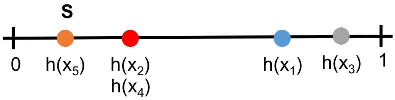
$\mathbb{E}[s] = \frac{1}{d+1}$

$\frac{1}{\mathbb{E}[s]} - 1 = d$

$\mathbb{E}\left[\frac{1}{s} - 1\right] \neq \frac{1}{\mathbb{E}[s]} - 1$

Performance in Expectation

s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.

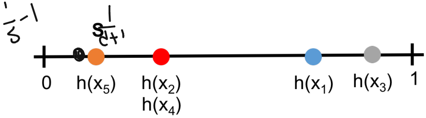


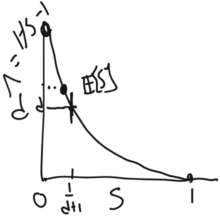
$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

- So our estimate $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ is correct if **s** exactly equals its expectation. Does this mean $\mathbb{E}[\widehat{\mathbf{d}}] = d$? No, but:

Performance in Expectation

s is the minimum of d points chosen uniformly at random on $[0, 1]$. Where $d = \#$ distinct elements.





$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

$s \leq \frac{1}{d+1}(1+\varepsilon)$

$\hat{d} = \frac{1}{s} - 1 \geq \frac{1}{\frac{1}{(d+1)(1+\varepsilon)}} - 1$

$\geq (1-2\varepsilon)d$

- So our estimate $\hat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ is correct if **s** exactly equals its expectation. Does this mean $\mathbb{E}[\hat{\mathbf{d}}] = d$? No, but:
- **Approximation is robust:** if $|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \leq \epsilon \cdot \mathbb{E}[\mathbf{s}]$ for any $\epsilon \in (0, 1/2)$ and a small constant $c \leq 4$:

$|\hat{d} - d| \leq < \varepsilon d$

$$(1 - c\epsilon)d \leq \hat{\mathbf{d}} \leq (1 + c\epsilon)d$$

$\hat{d} = \frac{1}{s} - 1$

Initial Concentration Bound

So question is how well s concentrates around its mean.

$$\mathbb{E}[\mathsf{s}] = \frac{1}{d+1}$$

s: minimum of d distinct hashes chosen randomly over $[0,1]$, computed by hashing algorithm. $\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of $\#$ distinct elements d.

Initial Concentration Bound

So question is how well $\mathbf{s}$ concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ and } \mathrm{Var}[\mathbf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

$\mathbf{s}$: minimum of d distinct hashes chosen randomly over $[0,1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

Initial Concentration Bound

So question is how well s concentrates around its mean.

$$\mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ and } \mathsf{Var}[\mathsf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

Chebyshev's Inequality:

$$\Pr\left[|\mathsf{s} - \mathbb{E}[\mathsf{s}]| \geq \epsilon\mathbb{E}[\mathsf{s}]\right] \leq \frac{\mathsf{Var}[\mathsf{s}]}{(\epsilon\mathbb{E}[\mathsf{s}])^2} \qquad .$$

s: minimum of d distinct hashes chosen randomly over $[0,1]$, computed by hashing algorithm. $\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Initial Concentration Bound

So question is how well s concentrates around its mean.

$$\mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ and } \mathrm{Var}[\mathsf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

Chebyshev's Inequality:

$$\Pr\left[|\mathsf{s} - \mathbb{E}[\mathsf{s}]| \geq \epsilon \mathbb{E}[\mathsf{s}]\right] \leq \frac{\mathrm{Var}[\mathsf{s}]}{(\epsilon \mathbb{E}[\mathsf{s}])^2} = \frac{1}{\epsilon^2}. \geq 1$$

> s: minimum of d distinct hashes chosen randomly over $[0,1]$, computed by hashing algorithm. $\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Initial Concentration Bound

So question is how well s concentrates around its mean.

$$\mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ and } \mathrm{Var}[\mathsf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus}).}$$

Chebyshev's Inequality:

$$\Pr\left[|\mathsf{s} - \mathbb{E}[\mathsf{s}]| \geq \epsilon \mathbb{E}[\mathsf{s}]\right] \leq \frac{\mathrm{Var}[\mathsf{s}]}{(\epsilon \mathbb{E}[\mathsf{s}])^2} = \frac{1}{\epsilon^2}.$$

Bound is vacuous for any $\epsilon < 1$.

s: minimum of d distinct hashes chosen randomly over $[0,1]$, computed by hashing algorithm. $\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Initial Concentration Bound

So question is how well $\mathbf{s}$ concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ and } \mathrm{Var}[\mathbf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus}).}$$

Chebyshev's Inequality:

$$\Pr\left[|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \geq \epsilon \mathbb{E}[\mathbf{s}]\right] \leq \frac{\mathrm{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2} = \frac{1}{\epsilon^2}.$$

Bound is vacuous for any $\epsilon < 1$. How can we improve accuracy?

> $\mathbf{s}$: minimum of d distinct hashes chosen randomly over $[0,1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

Improving Performance

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Improving Performance

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathrm{h} : U \to [0, 1]$ be a random hash function
- $\mathrm{s} := 1$
- For $i = 1, \ldots, n$
    - $\mathrm{s} := \min(\mathrm{s}, \mathrm{h}(x_i))$
- Return $\widehat{\mathrm{d}} = \frac{1}{\mathrm{s}} - 1$

Improving Performance

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathsf{h}_1, \mathsf{h}_2, \ldots, \mathsf{h}_k : U \to [0,1]$ be random hash functions
- $\mathsf{s} := 1$
- For $i = 1, \ldots, n$
  - $\mathsf{s} := \min(\mathsf{s}, \mathsf{h}(x_i))$
- Return $\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$

Improving Performance

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_i))$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

Improving Performance

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
    - For j=1,...,k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

Improving Performance

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,...,k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

Improving Performance

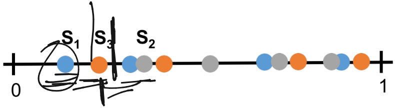
Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,...,k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

$\frac{1}{d+1} = \mathbb{E} S$

$k \cdot d$



Analysis

$s = \frac{1}{k}\sum_{j=1}^{k} s_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[s_j] = \frac{1}{d+1}$$

$$\mathrm{Var}[s_j] \leq \frac{1}{(d+1)^2}$$

s_j: minimum of d distinct hashes chosen randomly over $[0, 1]$. $s = \frac{1}{k}\sum_{j=1}^{k} s_j$.
$\widehat{d} = \frac{1}{s} - 1$: estimate of # distinct elements d.

Analysis

$s = \frac{1}{k}\sum_{j=1}^{k} s_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[s_j] = \frac{1}{d+1} \implies \mathbb{E}[s] = \frac{1}{d+1}$$

$$\mathrm{Var}[s_j] \leq \frac{1}{(d+1)^2}$$

s_j: minimum of d distinct hashes chosen randomly over $[0, 1]$. $s = \frac{1}{k}\sum_{j=1}^{k} s_j$.
$\widehat{d} = \frac{1}{s} - 1$: estimate of # distinct elements d.

Analysis

$\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathsf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathsf{s}_j] \leq \frac{1}{(d+1)^2}$$

s_j: minimum of d distinct hashes chosen randomly over $[0, 1]$. $\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$.
$\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of $\#$ distinct elements d.

Analysis

$\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathsf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathsf{s}_j] \le \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathsf{s}] = \frac{1}{k(d+1)^2}$$

s_j: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$.
$\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Analysis

$\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathsf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathsf{s}_j] \leq \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathsf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

s_j: minimum of d distinct hashes chosen randomly over $[0, 1]$. $\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$.
$\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Analysis

$\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathsf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathsf{s}_j] \leq \frac{1}{(d+1)^2} \implies \underline{\mathrm{Var}[\mathsf{s}] \leq \frac{1}{k \cdot (d+1)^2}} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[|\mathsf{s} - \mathbb{E}[\mathsf{s}]| \geq \epsilon\mathbb{E}[\mathsf{s}]\right] \leq \frac{\mathrm{Var}[\mathsf{s}]}{(\epsilon\mathbb{E}[\mathsf{s}])^2}$$

s_j: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$.
$\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Analysis

$s = \frac{1}{k}\sum_{j=1}^{k} s_j$. Have already shown that for $j = 1, \ldots, k$:

$Var(s) = \frac{1}{k^2} \cdot \sum Var(s_j) = \frac{1}{k^2} \cdot k \cdot Var(s_j)$

$$\mathbb{E}[s_j] = \frac{1}{d+1} \implies \mathbb{E}[s] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[s_j] \le \frac{1}{(d+1)^2} \implies \mathrm{Var}[s] \le \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[|s - \mathbb{E}[s]| \ge \epsilon \mathbb{E}[s]\right] \le \frac{\mathrm{Var}[s]}{(\epsilon \mathbb{E}[s])^2} = \frac{\cancel{\mathbb{E}[s]^2}/k}{\epsilon^2 \cancel{\mathbb{E}[s]^2}} = \frac{1}{k \cdot \epsilon^2}$$

> s_j: minimum of d distinct hashes chosen randomly over $[0, 1]$. $s = \frac{1}{k}\sum_{j=1}^{k} s_j$.
> $\widehat{d} = \frac{1}{s} - 1$: estimate of # distinct elements d.

Analysis

$\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathsf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathsf{s}_j] \leq \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathsf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[|\mathsf{s} - \mathbb{E}[\mathsf{s}]| \geq \underbrace{\overset{.05}{\epsilon}\mathbb{E}[\mathsf{s}]}\right] \leq \frac{\mathrm{Var}[\mathsf{s}]}{(\epsilon\mathbb{E}[\mathsf{s}])^2} = \frac{\mathbb{E}[\mathsf{s}]^2/k}{\epsilon^2\mathbb{E}[\mathsf{s}]^2} = \frac{1}{\underline{k \cdot \epsilon^2}} \underline{\leq \overset{.01}{\delta}}$$

How should we set k if we want an error with probability at most δ?

$$k \geq \frac{1}{\epsilon^2 \delta} \qquad \frac{1}{.05^2 \cdot .01}$$

s_j: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$.
$\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Analysis

$\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$. Have already shown that for $j = 1, \dots, k$:

$$\mathbb{E}[\mathsf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathsf{s}_j] \leq \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathsf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[|\mathsf{s} - \mathbb{E}[\mathsf{s}]| \geq \epsilon \mathbb{E}[\mathsf{s}]\right] \leq \frac{\mathrm{Var}[\mathsf{s}]}{(\epsilon \mathbb{E}[\mathsf{s}])^2} = \frac{\mathbb{E}[\mathsf{s}]^2/k}{\epsilon^2 \mathbb{E}[\mathsf{s}]^2} = \frac{1}{k \cdot \epsilon^2}$$

How should we set k if we want an error with probability at most δ?

$k = \frac{1}{\epsilon^2 \cdot \delta}$.

> s_j: minimum of d distinct hashes chosen randomly over $[0, 1]$. $\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$.
> $\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of # distinct elements d.

Analysis

$\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathsf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathsf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathsf{s}_j] \leq \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathsf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[|\mathsf{s} - \mathbb{E}[\mathsf{s}]| \geq \epsilon \mathbb{E}[\mathsf{s}]\right] \leq \frac{\mathrm{Var}[\mathsf{s}]}{(\epsilon \mathbb{E}[\mathsf{s}])^2} = \frac{\mathbb{E}[\mathsf{s}]^2/k}{\epsilon^2 \mathbb{E}[\mathsf{s}]^2} = \frac{1}{k \cdot \epsilon^2} = \frac{\epsilon^2 \cdot \delta}{\epsilon^2} = \delta.$$

How should we set k if we want an error with probability at most δ?

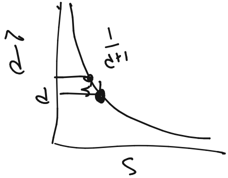
$k = \frac{1}{\epsilon^2 \cdot \delta}$.

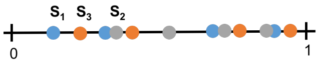
s_j: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathsf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathsf{s}_j$.
$\widehat{\mathsf{d}} = \frac{1}{\mathsf{s}} - 1$: estimate of $\#$ distinct elements d.

Space Complexity

Hashing for Distinct Elements:

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0,1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,..., k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$





- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.

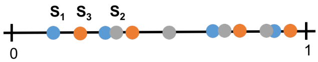
ε = relative error

δ = failure prob.

Space Complexity

Hashing for Distinct Elements:

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0,1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,..., k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$



- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.

d bits

- Space complexity is $k = \frac{1}{\epsilon^2 \cdot \delta}$ real numbers $s_1, \ldots, s_k$.

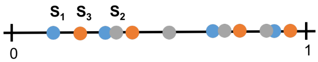
$\varepsilon = \delta = .01$ $\qquad \frac{1}{.01^3} = 1$ mil. $\qquad \frac{1}{.01^2} \cdot 5 = 50{,}000$

Space Complexity

Hashing for Distinct Elements:

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0,1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
    - For j=1,..., k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

$E\, S = \frac{1}{d+1}$



- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.
- Space complexity is $k = \frac{1}{\epsilon^2 \cdot \delta}$ real numbers $s_1, \ldots, s_k$.
- $\delta = 5\%$ failure rate gives a factor 20 overhead in space complexity.

Improved Failure Rate

How can we improve our dependence on the failure rate δ?

Improved Failure Rate

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/5$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{5}{\epsilon^2}$ hash functions.

Improved Failure Rate

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/5$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{5}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ be the outcomes of the t trials, return $\widehat{\mathrm{d}} = \mathit{median}(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

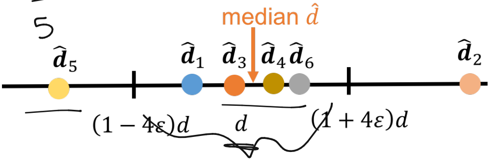
Improved Failure Rate

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/5$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{5}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ be the outcomes of the t trials, return $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

$\frac{1}{5}$

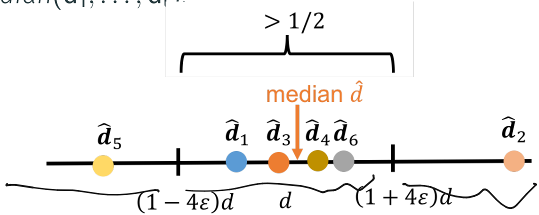


Improved Failure Rate

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/5$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{5}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ be the outcomes of the t trials, return $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.



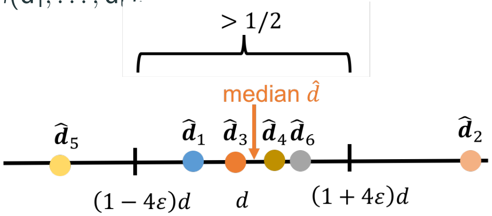
- If $> 1/2$ of trials fall in $[(1-4\epsilon)d, (1+4\epsilon)d]$, then the median will.

Improved Failure Rate

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/5$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{5}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ be the outcomes of the t trials, return $\widehat{\mathrm{d}} = \mathit{median}(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.



- If $> 1/2$ of trials fall in $[(1-4\epsilon)d, (1+4\epsilon)d]$, then the median will.
- Have $< 1/2$ of trials on both the left and right.

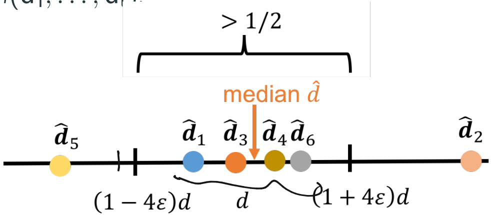
Improved Failure Rate

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/5$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{5}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ be the outcomes of the t trials, return $\widehat{\mathrm{d}} = median(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

$1 - \delta' = 4/5$



- If $> 2/3$ of trials fall in $[(1-4\epsilon)d, (1+4\epsilon)d]$, then the median will.
- Have $< 1/3$ of trials on both the left and right.

The Median Trick

- $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathrm{d}} = \mathit{median}(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

What is the probability that the median $\widehat{\mathrm{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

The Median Trick

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$.

The Median Trick

- $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathrm{d}} = median(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

What is the probability that the median $\widehat{\mathrm{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$.

$$\Pr\left(\widehat{\mathrm{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(X < \frac{2}{3} \cdot t\right)$$

The Median Trick

- $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathrm{d}} = median(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

What is the probability that the median $\widehat{\mathrm{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathrm{X}] =$

$$\Pr\left(\widehat{\mathrm{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathrm{X} < \frac{2}{3} \cdot t\right)$$

The Median Trick

- $\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least $4/5$.
- $\widehat{\mathsf{d}} = \mathit{median}(\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t)$.

What is the probability that the median $\widehat{\mathsf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\boxed{\mathbb{E}[\mathsf{X}] = \frac{4}{5} \cdot t.}$

$$\Pr\left(\widehat{\mathsf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathsf{X} < \frac{2}{3} \cdot t\right)$$

The Median Trick

- $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathrm{d}} = median(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

What is the probability that the median $\widehat{\mathrm{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[X] = \frac{4}{5} \cdot t$.

$\frac{5}{6} \cdot \frac{4}{5} = \frac{2}{3}$

$$\underline{\Pr\left(\widehat{\mathrm{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right)} \leq \Pr\left(X < \frac{5}{6} \cdot \mathbb{E}[X]\right)$$

The Median Trick

- $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least $4/5$.
- $\widehat{\mathrm{d}} = \mathit{median}(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

What is the probability that the median $\widehat{\mathrm{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathrm{X}] = \frac{4}{5} \cdot t$.

$$\Pr\left(\widehat{\mathrm{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathrm{X} < \frac{5}{6} \cdot \mathbb{E}[\mathrm{X}]\right) \leq \Pr\left(|\mathrm{X} - \mathbb{E}[\mathrm{X}]| \geq \frac{1}{6}\mathbb{E}[\mathrm{X}]\right)$$

The Median Trick

- $\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathsf{d}} = median(\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t)$.

What is the probability that the median $\widehat{\mathsf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[X] = \frac{4}{5} \cdot t$.

$$\Pr\left(\widehat{\mathsf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(X < \frac{5}{6} \cdot \mathbb{E}[X]\right) \leq \Pr\left(|X - \mathbb{E}[X]| \geq \frac{1}{6}\mathbb{E}[X]\right)$$

Apply Chernoff bound:

The Median Trick

- $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathrm{d}} = median(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

What is the probability that the median $\widehat{\mathrm{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\underline{\mathbb{E}[X] = \frac{4}{5} \cdot t}$.

$Pr\left(X < \frac{2}{3}t\right)$

$$\Pr\left(\underline{\widehat{\mathrm{d}} \notin} [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(X < \underline{\frac{5}{6} \cdot \mathbb{E}[X]}\right) \leq \Pr\left(|X - \mathbb{E}[X]| \geq \frac{1}{6}\mathbb{E}[X]\right)$$

Apply Chernoff bound:

$$\Pr\left(\underline{|X - \mathbb{E}[X]|} \geq \underline{\frac{1}{6}\mathbb{E}[X]}\right) \leq \underline{2}\exp\left(-\frac{\frac{1}{6}^2 \cdot \frac{4}{5}t}{\underline{2} + \underline{1/6}}\right) = O\left(e^{-ct}\right).$$

The Median Trick

- $\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t$ are the outcomes of the t trials, each falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least 4/5.
- $\widehat{\mathsf{d}} = median(\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t)$.

What is the probability that the median $\widehat{\mathsf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let X be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[X] = \frac{4}{5} \cdot t$.

$$\Pr\left(\widehat{\mathsf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(X < \frac{5}{6} \cdot \mathbb{E}[X]\right) \leq \Pr\left(|X - \mathbb{E}[X]| \geq \frac{1}{6}\mathbb{E}[X]\right)$$

Apply Chernoff bound:

$$\leq \delta = e^{-ct}$$

$$\Pr\left(|X - \mathbb{E}[X]| \geq \frac{1}{6}\mathbb{E}[X]\right) \leq 2\exp\left(-\frac{\frac{1}{6}^2 \cdot \frac{4}{5}t}{2+1/6}\right) = O\left(e^{-ct}\right).$$

- Setting $t = O(\log(1/\delta))$ gives failure probability $e^{-\log(1/\delta)} = \delta$.

Median Trick

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns $\widehat{d} \in [(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least $1-\delta$.

Median Trick

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns $\widehat{\mathbf{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least $1-\delta$.

Total Space Complexity: t trials, each using $k = \frac{1}{\epsilon^2\delta'}$ hash functions, for $\delta' = 1/5$. Space is $\frac{5t}{\epsilon^2} = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ real numbers (the minimum value of each hash function).

Median Trick

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns $\widehat{\mathbf{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least $1-\delta$.

Total Space Complexity: t trials, each using $k = \frac{1}{\epsilon^2 \delta'}$ hash functions, for $\delta' = 1/5$. Space is $\frac{5t}{\epsilon^2} = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ real numbers (the minimum value of each hash function).

No dependence on the number of distinct elements d or the number of items in the stream n! Both of these numbers are typically very large.

Median Trick

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns $\widehat{\mathrm{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$ with probability at least $1-\delta$.

Total Space Complexity: t trials, each using $k = \frac{1}{\epsilon^2 \delta'}$ hash functions, for $\delta' = 1/5$. Space is $\frac{5t}{\epsilon^2} = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ real numbers (the minimum value of each hash function).

No dependence on the number of distinct elements d or the number of items in the stream n! Both of these numbers are typically very large.

A note on the median: The median is often used as a robust alternative to the mean, when there are outliers (e.g., heavy tailed distributions, corrupted data).

Distinct Elements in Practice

Our algorithm uses continuous valued fully random hash functions.

Distinct Elements in Practice

Our algorithm uses continuous valued fully random hash functions.
Can’t be implemented…

Distinct Elements in Practice

Our algorithm uses continuous valued fully random hash functions. Can't be implemented...

- The idea of using the minimum hash value of $x_1, \ldots, x_n$ to estimate the number of distinct elements naturally extends to when the hash functions map to discrete values.

Distinct Elements in Practice

Our algorithm uses continuous valued fully random hash functions. Can't be implemented...

- The idea of using the minimum hash value of $x_1, \ldots, x_n$ to estimate the number of distinct elements naturally extends to when the hash functions map to discrete values.
- Flajolet-Martin (LogLog) algorithm and HyperLogLog.

Distinct Elements in Practice

Our algorithm uses continuous valued fully random hash functions. Can't be implemented...

- The idea of using the minimum hash value of $x_1, \ldots, x_n$ to estimate the number of distinct elements naturally extends to when the hash functions map to discrete values.
- Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	1011000

Distinct Elements in Practice

Our algorithm uses continuous valued fully random hash functions. Can't be implemented...

- The idea of using the minimum hash value of $x_1, \ldots, x_n$ to estimate the number of distinct elements naturally extends to when the hash functions map to discrete values.
- Flajolet-Martin (LogLog) algorithm and HyperLogLog.

h(x_1)	**1010010**
h(x_2)	**1001100**
h(x_3)	**1001110**
⋮	
h(x_n)	**1011000**

Estimate # distinct elements based on maximum number of trailing zeros **m**.

Distinct Elements in Practice

Our algorithm uses continuous valued fully random hash functions. Can't be implemented...

- The idea of using the minimum hash value of $x_1, \ldots, x_n$ to estimate the number of distinct elements naturally extends to when the hash functions map to discrete values.
- Flajolet-Martin (LogLog) algorithm and HyperLogLog.

h(x$_1$)	**1010010**
h(x$_2$)	**1001100**
h(x$_3$)	**1001110**
⋮	
h(x$_n$)	**1011000**

Estimate # distinct elements based on maximum number of trailing zeros **m**.
The more distinct hashes we see, the higher we expect this maximum to be.

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

a) $O(1)$ b) $O(\log d)$ c) $O(\sqrt{d})$ d) $O(d)$

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
$\vdots$	
$h(x_n)$	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } x \text{ trailing zeros}) =$$

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } x \text{ trailing zeros}) = \frac{1}{2^x}$$

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log d \text{ trailing zeros}) = \frac{1}{2^{\log d}}$$

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
$\vdots$	
$h(x_n)$	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log d \text{ trailing zeros}) = \frac{1}{2^{\log d}} = \frac{1}{d}.$$

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
$\vdots$	
$h(x_n)$	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log d \text{ trailing zeros}) = \frac{1}{2^{\log d}} = \frac{1}{d}.$$

So with d distinct hashes, expect to see 1 with $\log d$ trailing zeros.

Expect $\mathbf{m} \approx \log d$.

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

h(x_1)	1010010
h(x_2)	1001100
h(x_3)	1001110
⋮	
h(x_n)	1011000

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log d \text{ trailing zeros}) = \frac{1}{2^{\log d}} = \frac{1}{d}.$$

So with d distinct hashes, expect to see 1 with $\log d$ trailing zeros.

Expect $\mathbf{m} \approx \log d$. **m** takes $\underline{\log \log d}$ bits to store.

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

h(x_1)	**1010010**
h(x_2)	**1001100**
h(x_3)	**1001110**
⋮	
h(x_n)	**1011000**

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log d \text{ trailing zeros}) = \frac{1}{2^{\log d}} = \frac{1}{d}.$$

So with d distinct hashes, expect to see 1 with $\log d$ trailing zeros.

Expect $\mathbf{m} \approx \log d$. $\mathbf{m}$ takes $\log \log d$ bits to store.

Total Space: $O\left(\frac{\log \log d}{\epsilon^2}\right)$ for an ϵ approximate count.

LogLog Counting of Distinct Elements

Flajolet-Martin (LogLog) algorithm and HyperLogLog.

h(x$_1$)	**1010010**
h(x$_2$)	**1001100**
h(x$_3$)	**1001110**
⋮	
h(x$_n$)	**1011000**

Estimate # distinct elements based on maximum number of trailing zeros **m**.

With d distinct elements, roughly what do we expect **m** to be?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log d \text{ trailing zeros}) = \frac{1}{2^{\log d}} = \frac{1}{d}.$$

So with d distinct hashes, expect to see 1 with $\log d$ trailing zeros.

Expect $\mathbf{m} \approx \log d$. **m** takes $\log \log d$ bits to store.

Total Space: $O\left(\frac{\log \log d}{\epsilon^2}\right)$ for an ϵ approximate count.

Note: Careful averaging of estimates from multiple hash functions.

LogLog Space Guarantees

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\text{space used } = O\left(\frac{\log\log d}{\epsilon^2}\right)$$

LogLog Space Guarantees

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\begin{aligned} \text{space used } &= O\left(\frac{\log\log d}{\epsilon^2}\right) \\ &= \frac{1.04 \cdot \lceil \log_2 \log_2 d \rceil}{\epsilon^2} \text{ bits}^1 \end{aligned}$$

1. 1.04 is the constant in the HyperLogLog analysis. Not important!

LogLog Space Guarantees

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\begin{aligned}\text{space used } &= O\left(\frac{\log\log d}{\epsilon^2}\right)\\ &= \frac{1.04 \cdot \lceil \log_2 \log_2 d \rceil}{\epsilon^2} \text{ bits}^{1}\\ &= \frac{1.04 \cdot 5}{.02^2} = 13000 \text{ bits} \approx 1.6\ kB!\end{aligned}$$

1. 1.04 is the constant in the HyperLogLog analysis. Not important!

LogLog Space Guarantees

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\begin{aligned}\text{space used } &= O\left(\frac{\log\log d}{\epsilon^2}\right)\\ &= \frac{1.04 \cdot \lceil \log_2 \log_2 d \rceil}{\epsilon^2} \text{ bits}^1\\ &= \frac{1.04 \cdot 5}{.02^2} = 13000 \text{ bits} \approx 1.6\ kB!\end{aligned}$$

Mergeable Sketch: Consider the case (essentially always in practice) that the items are processed on different machines.

1. 1.04 is the constant in the HyperLogLog analysis. Not important!

LogLog Space Guarantees

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\begin{aligned}\text{space used} &= O\left(\frac{\log\log d}{\epsilon^2}\right)\\ &= \frac{1.04 \cdot \lceil \log_2 \log_2 d \rceil}{\epsilon^2} \text{ bits}^1\\ &= \frac{1.04 \cdot 5}{.02^2} = 13000 \text{ bits} \approx 1.6\ kB!\end{aligned}$$

Mergeable Sketch: Consider the case (essentially always in practice) that the items are processed on different machines.

- Given data structures (sketches) $HLL(x_1, \ldots, x_n)$, $HLL(y_1, \ldots, y_n)$ is is easy to merge them to give $HLL(x_1, \ldots, x_n, y_1, \ldots, y_n)$.

1. 1.04 is the constant in the HyperLogLog analysis. Not important!

LogLog Space Guarantees

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\begin{aligned} \text{space used } &= O\left(\frac{\log\log d}{\epsilon^2}\right) \\ &= \frac{1.04 \cdot \lceil \log_2 \log_2 d \rceil}{\epsilon^2} \text{ bits}^1 \\ &= \frac{1.04 \cdot 5}{.02^2} = 13000 \text{ bits} \approx 1.6\ kB! \end{aligned}$$

Mergeable Sketch: Consider the case (essentially always in practice) that the items are processed on different machines.

- Given data structures (sketches) $HLL(x_1, \ldots, x_n)$, $HLL(y_1, \ldots, y_n)$ is is easy to merge them to give $HLL(x_1, \ldots, x_n, y_1, \ldots, y_n)$. How?

1. 1.04 is the constant in the HyperLogLog analysis. Not important!

LogLog Space Guarantees

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\begin{aligned}\text{space used } &= O\left(\frac{\log\log d}{\epsilon^2}\right)\\ &= \frac{1.04\cdot\lceil\log_2\log_2 d\rceil}{\epsilon^2}\text{ bits}^1\\ &= \frac{1.04\cdot 5}{.02^2} = 13000\text{ bits} \approx 1.6\ kB!\end{aligned}$$

Mergeable Sketch: Consider the case (essentially always in practice) that the items are processed on different machines.

- Given data structures (sketches) $HLL(x_1,\ldots,x_n)$, $HLL(y_1,\ldots,y_n)$ is is easy to merge them to give $HLL(x_1,\ldots,x_n,y_1,\ldots,y_n)$. How?
- Set the maximum # of trailing zeros to the maximum in the two sketches.

1. 1.04 is the constant in the HyperLogLog analysis. Not important!

HyperLogLog In Practice

Implementations: Google PowerDrill, Facebook Presto, Twitter Algebird, Amazon Redshift.

HyperLogLog In Practice

Implementations: Google PowerDrill, Facebook Presto, Twitter Algebird, Amazon Redshift.

HyperLogLog In Practice

Implementations: Google PowerDrill, Facebook Presto, Twitter Algebird, Amazon Redshift.

Use Case: Exploratory SQL-like queries on tables with 100s billions of rows. $\sim$ 5 million count distinct queries per day.

HyperLogLog In Practice

Implementations: Google PowerDrill, Facebook Presto, Twitter Algebird, Amazon Redshift.

Use Case: Exploratory SQL-like queries on tables with 100s billions of rows. $\sim$ 5 million count distinct queries per day. E.g.,

- Count number if distinct users in Germany that made at least one search containing the word 'auto' in the last month.
- Count number of distinct subject lines in emails sent by users that have registered in the last week, in comparison to number of emails sent overall (to estimate rates of spam accounts).

HyperLogLog In Practice

Implementations: Google PowerDrill, Facebook Presto, Twitter Algebird, Amazon Redshift.

Use Case: Exploratory SQL-like queries on tables with 100s billions of rows. ~ 5 million count distinct queries per day. E.g.,

- Count number if distinct users in Germany that made at least one search containing the word 'auto' in the last month.
- Count number of distinct subject lines in emails sent by users that have registered in the last week, in comparison to number of emails sent overall (to estimate rates of spam accounts).

Traditional *COUNT*, *DISTINCT* SQL calls are far too slow, especially when the data is distributed across many servers.

Questions on distinct elements counting?