COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2024.

Lecture 8

Summary

Last Class:

- Finish up Bloom Filters and optimization of number of hash functions.
- Start on streaming algorithms.
- Introduce the frequent items problem and its applications.
- Start on the Count-Min sketch algorithm for frequent items.

This Class:

- Analysis of Count-Min sketch .
- Start on distinct items counting problem.

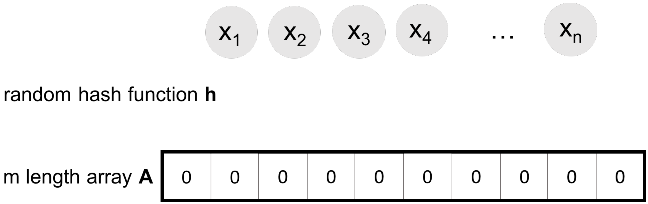
(ϵ, k)**-Frequent Items Problem**: Consider a stream of n items $x_1, \ldots, x_n$. Return a set F of items, including all items that appear at least $\frac{n}{k}$ times and only items that appear at least $(1-\epsilon) \cdot \frac{n}{k}$ times.

- To solve this problem, it suffices to estimate the frequency $f(x)$ of each item x up to error $\pm \frac{\epsilon n}{k}$.
- Will discuss later how to maintain the list of top items in small space.

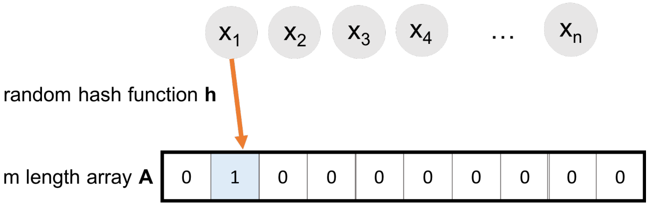
Frequent Elements with Count-Min Sketch

Count-min sketch:

Count-min sketch:

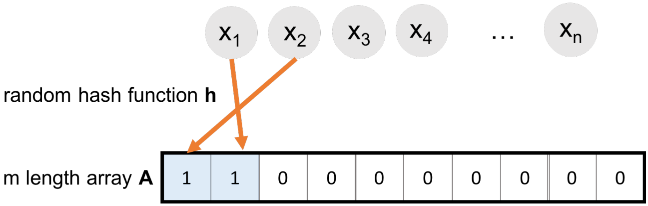


Count-min sketch:



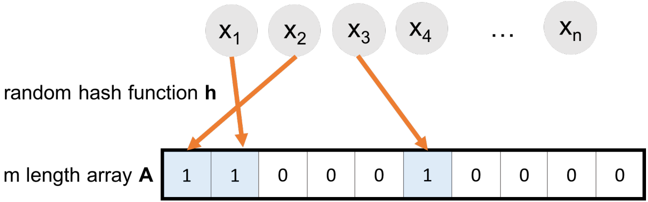
Frequent Elements with Count-Min Sketch

Count-min sketch:

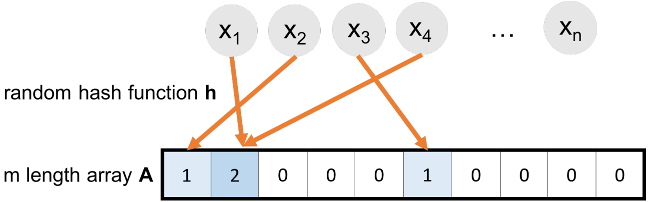


Frequent Elements with Count-Min Sketch

Count-min sketch:

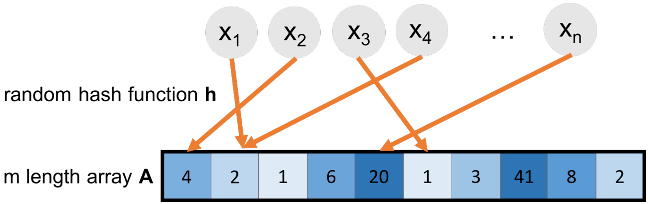


Count-min sketch:

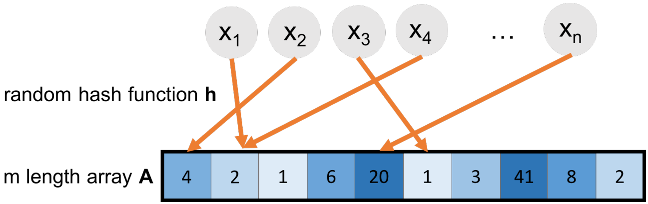


Frequent Elements with Count-Min Sketch

Count-min sketch:

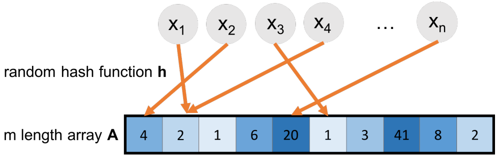


Count-min sketch:



Will use $A[\mathbf{h}(x)]$ to estimate $f(x)$, the frequency of x in the stream. I.e., $|\{x_i : x_i = x\}|$.

Count-Min Sketch Accuracy



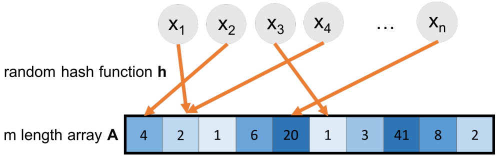
Use $A[\mathbf{h}(x)]$ to estimate $f(x)$.

Claim 1: We always have $A[\mathbf{h}(x)] \geq f(x)$.

- $A[\mathbf{h}(x)]$ counts the number of occurrences of any y with $\mathbf{h}(y) = \mathbf{h}(x)$, including x itself.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy



Use $A[\mathbf{h}(x)]$ to estimate $f(x)$.

Claim 1: We always have $A[\mathbf{h}(x)] \geq f(x)$.

- $A[\mathbf{h}(x)]$ counts the number of occurrences of any y with $\mathbf{h}(y) = \mathbf{h}(x)$, including x itself.
- $A[\mathbf{h}(x)] = f(x) + \sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)$.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathrm{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathrm{h}(y) = \mathrm{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). h: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathrm{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathrm{h}(y) = \mathrm{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathrm{h}(y) = \mathrm{h}(x)} f(y)\right] =$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). h: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}}.$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}}.$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y)$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathrm{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathrm{h}(y) = \mathrm{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathrm{h}(y) = \mathrm{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathrm{h}(y) = \mathrm{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). h: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}}.$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

What is a bound on probability that the error is $\geq \frac{2n}{m}$?

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

What is a bound on probability that the error is $\geq \frac{2n}{m}$?

Markov's inequality: $\Pr\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y) \geq \frac{2n}{m}\right] \leq \frac{1}{2}$.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

$$A[\mathbf{h}(x)] = f(x) + \underbrace{\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)}_{\text{error in frequency estimate}} .$$

Expected Error:

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \Pr(\mathbf{h}(y) = \mathbf{h}(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

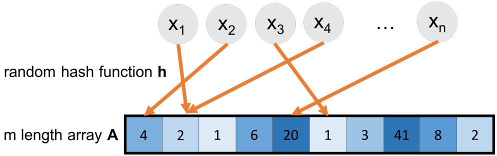
What is a bound on probability that the error is $\geq \frac{2n}{m}$?

Markov's inequality: $\Pr\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y) \geq \frac{2n}{m}\right] \leq \frac{1}{2}$.

What property of **h** is required to show this bound? a) fully random b) pairwise independent c) 2-universal d) locality sensitive

> $f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy

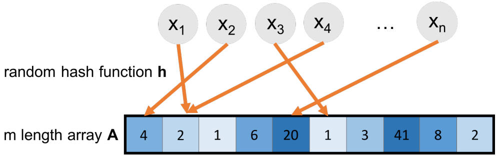


Claim: For any x, with probability at least 1/2,

$$f(x) \leq A[\mathbf{h}(x)] \leq f(x) + \frac{2n}{m}.$$

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy



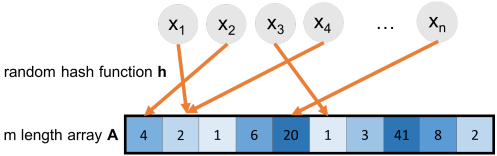
Claim: For any x, with probability at least 1/2,

$$f(x) \leq A[\mathbf{h}(x)] \leq f(x) + \frac{2n}{m}.$$

To solve the (ϵ, k)-Frequent elements problem, set $m = \frac{2k}{\epsilon}$.

$f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy



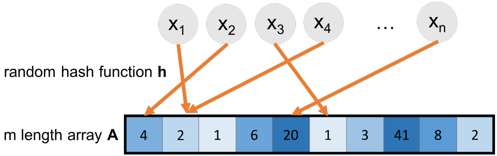
Claim: For any x, with probability at least 1/2,

$$f(x) \leq A[\mathbf{h}(x)] \leq f(x) + \frac{2n}{m}.$$

To solve the (ϵ, k)-Frequent elements problem, set $m = \frac{2k}{\epsilon}$. How can we improve the success probability?

> $f(x)$: frequency of x in the stream (i.e., number of items equal to x). $\mathbf{h}$: random hash function. m: size of Count-min sketch array.

Count-Min Sketch Accuracy



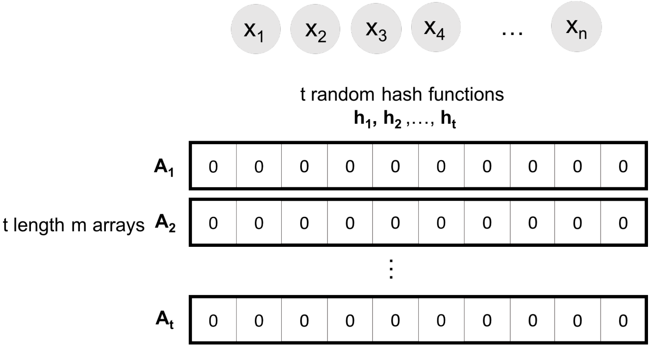
Claim: For any x, with probability at least 1/2,

$$f(x) \le A[\mathbf{h}(x)] \le f(x) + \frac{2n}{m}.$$

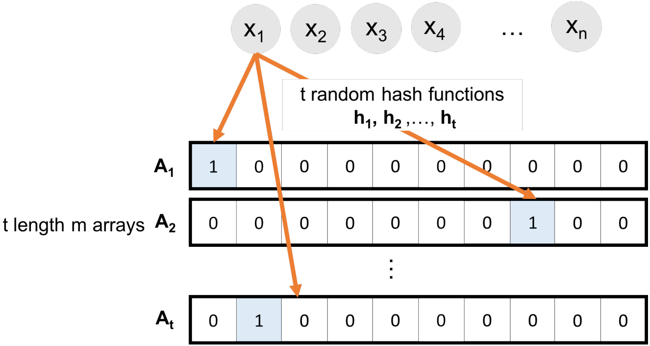
To solve the (ϵ, k)-Frequent elements problem, set $m = \frac{2k}{\epsilon}$. How can we improve the success probability? Repetition.

> $f(x)$: frequency of x in the stream (i.e., number of items equal to x). **h**: random hash function. m: size of Count-min sketch array.

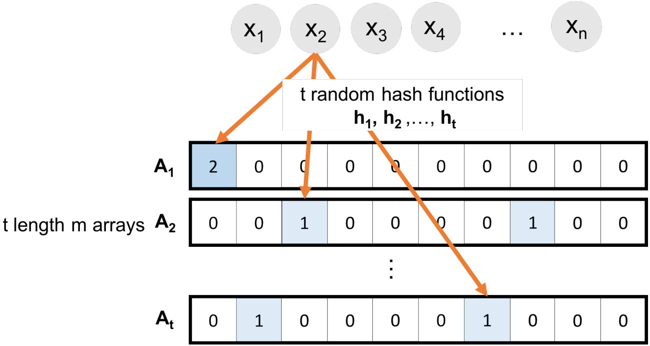
Count-Min Sketch Repetition



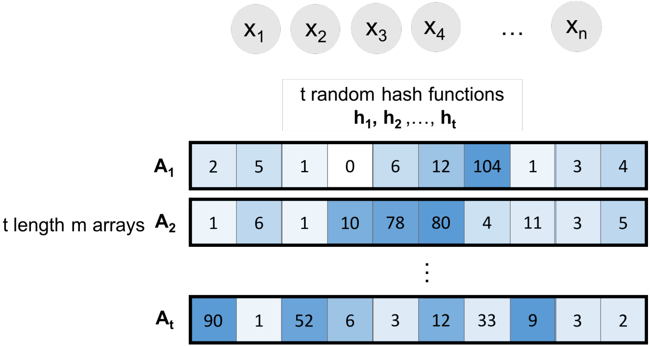
Count-Min Sketch Repetition



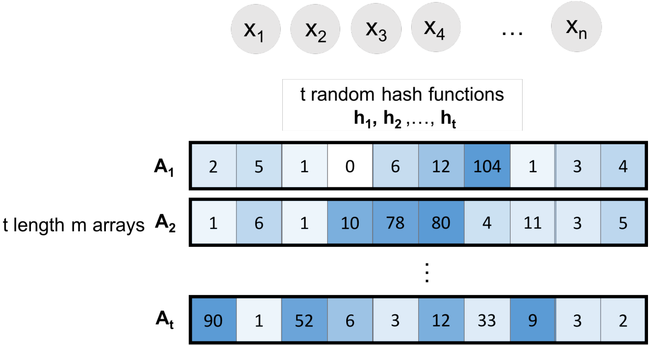
Count-Min Sketch Repetition



Count-Min Sketch Repetition

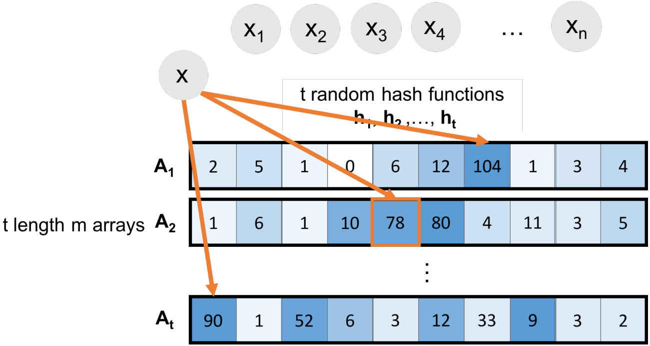


Count-Min Sketch Repetition



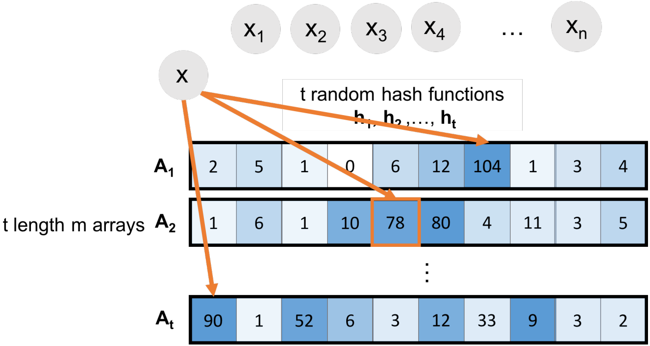
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (Count-min sketch)

Count-Min Sketch Repetition



Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (Count-min sketch)

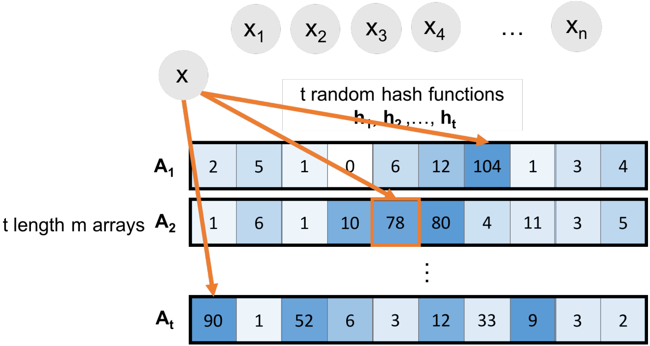
Count-Min Sketch Repetition



Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (Count-min sketch)

Why min instead of taking the average?

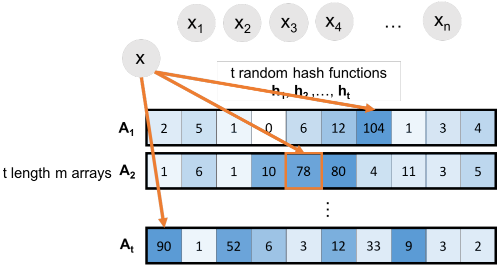
Count-Min Sketch Repetition



Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (Count-min sketch)

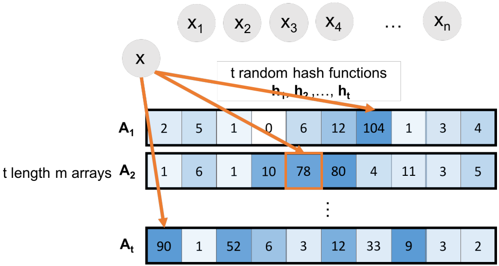
Why min instead of taking the average? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!

Count-Min Sketch Analysis



Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

Count-Min Sketch Analysis

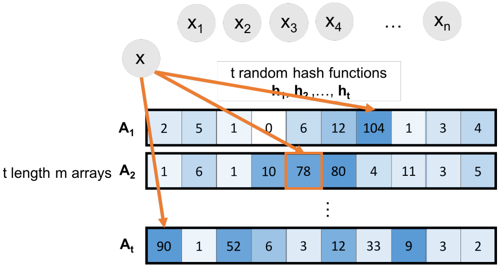


Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

Count-Min Sketch Analysis



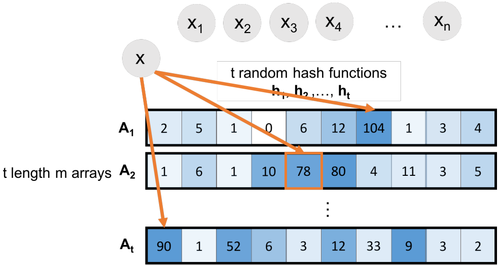
Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$?

Count-Min Sketch Analysis



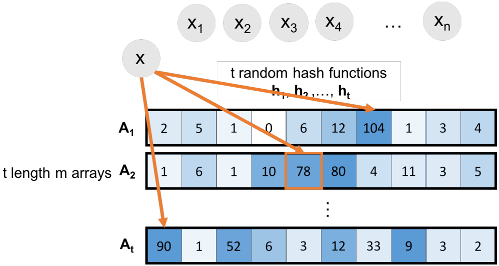
Estimate $f(x)$ by $\tilde{f}(x) = \min_{i\in[t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$.

Count-Min Sketch Analysis



Estimate $f(x)$ by $\tilde{f}(x) = \min_{i\in[t]} A_i[\mathbf{h}_i(x)]$

- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:

$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$.
- To get a good estimate with probability $\geq 1 - \delta$, set $t = \log_2(1/\delta)$.

Count-Min Sketch

Upshot: Count-min sketch lets us estimate the frequency of each item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O\left(\log(1/\delta) \cdot k/\epsilon\right)$ space.

Count-Min Sketch

Upshot: Count-min sketch lets us estimate the frequency of each item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O\left(\log(1/\delta) \cdot k/\epsilon\right)$ space.

- Accurate enough to solve the (ϵ, k)-Frequent elements problem – distinquish between items with frequency $\frac{n}{k}$ and those with frequency $(1-\epsilon)\frac{n}{k}$.

Upshot: Count-min sketch lets us estimate the frequency of each item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O(\log(1/\delta) \cdot k/\epsilon)$ space.

- Accurate enough to solve the (ϵ, k)-Frequent elements problem – distinquish between items with frequency $\frac{n}{k}$ and those with frequency $(1 - \epsilon)\frac{n}{k}$.
- How should we set δ if we want a good estimate for all items at once, with 99% probability?

Identifying Frequent Elements

Count-min sketch gives an accurate frequency estimate for every item in the stream. But how do we identify the frequent items without having to store/look up the estimated frequency for all elements in the stream?

Identifying Frequent Elements

Count-min sketch gives an accurate frequency estimate for every item in the stream. But how do we identify the frequent items without having to store/look up the estimated frequency for all elements in the stream?

One approach:

- When a new item comes in at step i, check if its estimated frequency is $\geq i/k$ and store it if so.
- At step i remove any stored items whose estimated frequency drops below i/k.
- Store at most $O(k)$ items at once and have all items with frequency $\geq n/k$ stored at the end of the stream.

Questions on Frequent Items?

Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements in the stream. E.g.,

$$1, 5, 7, 5, 2, 1 \rightarrow 4 \text{ distinct elements}$$

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements in the stream. E.g.,

$$1, 5, 7, 5, 2, 1 \rightarrow 4 \text{ distinct elements}$$

Applications:

- Distinct IP addresses clicking on an ad or visiting a site.
- Distinct values in a database column (for estimating sizes of joins and group bys).
- Number of distinct search engine queries.
- Counting distinct motifs in large DNA sequences.

Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements in the stream. E.g.,

$$1, 5, 7, 5, 2, 1 \to 4 \text{ distinct elements}$$

Applications:

- Distinct IP addresses clicking on an ad or visiting a site.
- Distinct values in a database column (for estimating sizes of joins and group bys).
- Number of distinct search engine queries.
- Counting distinct motifs in large DNA sequences.

Google Sawzall, Facebook Presto, Apache Drill, Twitter Algebird

Hashing for Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Hashing for Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

- 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

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

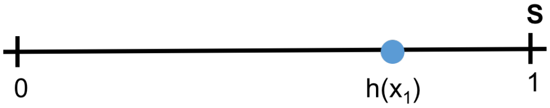
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
    - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$

Hashing for Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

- 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

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

- 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

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

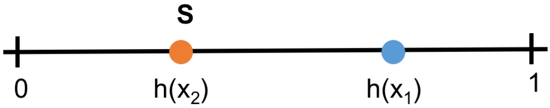
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
    - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$

Hashing for Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

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

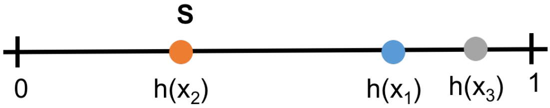


Hashing for Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

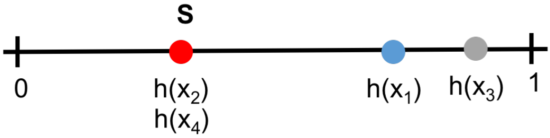
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
    - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

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

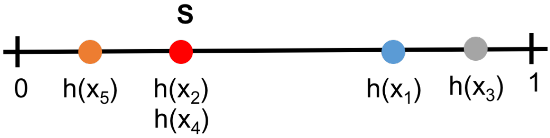


Hashing for Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

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

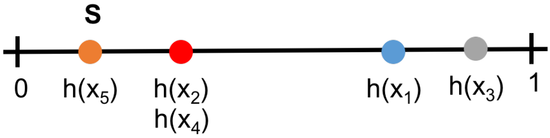


Hashing for Distinct Elements

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

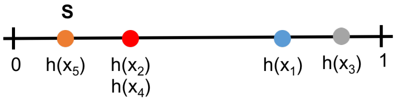
Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

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



Min-Hashing for Distinct Elements:

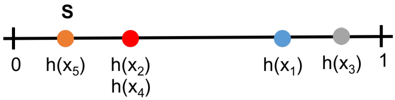
- Let $\mathbf{h} : U \rightarrow [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
    - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



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

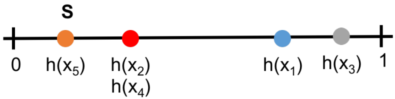


- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0, 1]$. Where $d = \#$ distinct elements.

Hashing for Distinct Elements

Min-Hashing for Distinct Elements:

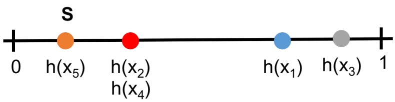
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
    - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



- 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.

Min-Hashing for Distinct Elements:

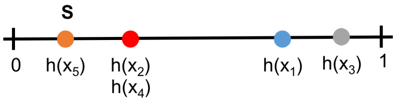
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



- 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.
- Same idea as Flajolet-Martin algorithm and HyperLogLog, except they use discrete hash functions.

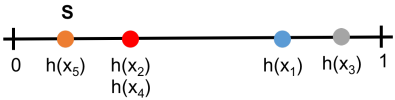
Performance in Expectation

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



Performance in Expectation

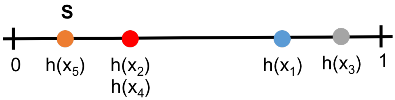
s is the minimum of d points chosen uniformly at random on $[0, 1]$.
Where $d = \#$ distinct elements.



$\mathbb{E}[s] =$

Performance in Expectation

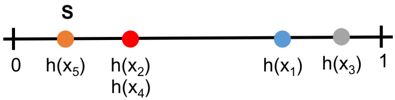
s is the minimum of d points chosen uniformly at random on $[0, 1]$. Where $d = \#$ distinct elements.



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

Performance in Expectation

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

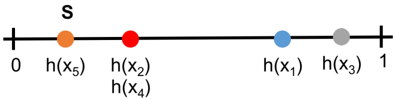


$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

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

Performance in Expectation

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

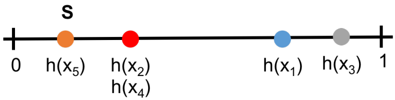


$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

- So our estimate $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ is correct if $\mathbf{s}$ exactly equals its expectation. Does this mean $\mathbb{E}[\widehat{\mathbf{d}}] = d$?

Performance in Expectation

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

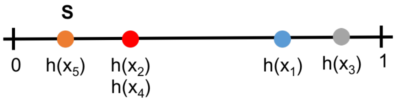


$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

- 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

$\mathbf{s}$ 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 $\mathbf{s}$ exactly equals its expectation. Does this mean $\mathbb{E}[\widehat{\mathbf{d}}] = d$? 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 $c \leq 4$:

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

.

Initial Concentration Bound

So question is how well s concentrates around its mean.

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

$$.$$

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.

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

Chebyshev's Inequality:

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

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.

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

Chebyshev's Inequality:

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

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 $\mathsf{h} : U \to [0,1]$ be a random hash function
- $\mathsf{s} := 1$
- 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 $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \rightarrow [0, 1]$ be random hash functions
- $\mathbf{s} := 1$
- 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 $i = 1, \ldots, n$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_i))$
- 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 $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \rightarrow [0, 1]$ 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):

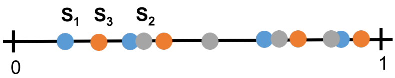
- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,...,k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$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



Analysis

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

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

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

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 $j = 1, \ldots, k$:

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

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

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 $j = 1, \ldots, k$:

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

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

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 $j = 1, \ldots, k$:

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

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

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 $j = 1, \ldots, k$:

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

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

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 $j = 1, \ldots, k$:

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

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

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

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

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

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

Chebyshev Inequality:

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

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:

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

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

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:

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

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 $j = 1, \ldots, k$:

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

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

Chebyshev Inequality:

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

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

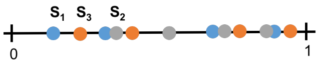
$k = \frac{1}{\epsilon^2 \cdot \delta}$.

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 $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

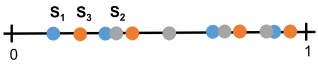


- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.

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 $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

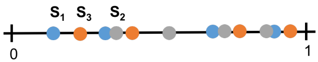


- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.
- Space complexity is $k = \frac{1}{\epsilon^2 \cdot \delta}$ real numbers $s_1, \ldots, s_k$.

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 $1 - \delta$.
- 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.

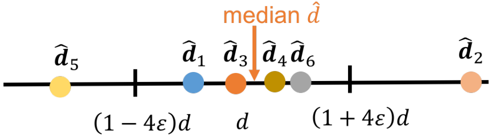
- Letting $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ be the outcomes of the t trials, return $\widehat{\mathrm{d}} = \mathit{median}(\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t)$.

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

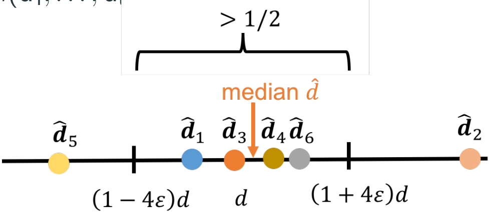


Improved Failure Rate

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 $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

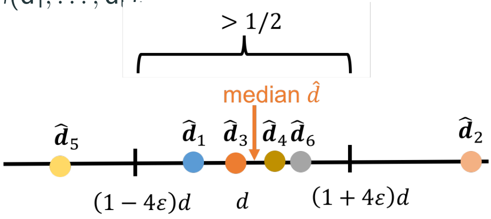


- If $> 1/2$ of trials fall in $[(1-4\epsilon)d, (1+4\epsilon)d]$, then the median will.

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 $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.



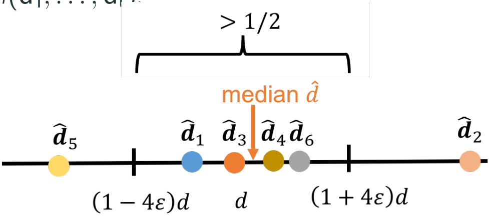
- If $> 1/2$ of trials fall in $[(1-4\epsilon)d, (1+4\epsilon)d]$, then the median will.
- Have $< 1/2$ of trials on both the left and right.

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



- If $> 2/3$ 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

- $\widehat{\mathrm{d}}_1, \ldots, \widehat{\mathrm{d}}_t$ 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 $[(1-4\epsilon)d, (1+4\epsilon)d]$.

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 $[(1-4\epsilon)d, (1+4\epsilon)d]$.

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

The Median Trick

- $\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t$ 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 $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[X] =$

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

The Median Trick

- $\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t$ 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 $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathsf{X}] = \frac{4}{5} \cdot t$.

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

The Median Trick

- $\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t$ 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 $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[X] = \frac{4}{5} \cdot t$.

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

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 $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[X] = \frac{4}{5} \cdot t$.

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

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

- $\widehat{\mathsf{d}}_1, \ldots, \widehat{\mathsf{d}}_t$ 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:

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

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:

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

- 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 $\widehat{\mathrm{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$ 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 $\widehat{\mathbf{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$ 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.

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).