

COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2026.

Lecture 8

In-Class Quiz 4

Question 1: Let **A** be a bloom filter that uses $k = 4$ hash functions and currently has exactly $1/3$ of its bits set to 1. Let w be some item that has not yet been inserted into **A**. What is $\Pr[\text{query}(w) = 1]$? I.e., what is the probability that w is a false positive?

$$\Pr[\text{query}(w) = 1] = \Pr\left(\bigcap_{i=1}^4 A[h_i(w)] = 1\right) = \left(\frac{1}{3}\right)^4 = \frac{1}{81}$$

Question 2: A Chernoff bound can be applied to bound the maximum number of items in a hash table bucket when the hash function used is: (pick all that apply)

- sum of independent random variables*
- $S_i = \sum_{j=1}^n S_{ij}$ — $S_{ij} = 1$ if item j hashes to bucket i
- (a) ~~2-universal~~ (b) ~~pairwise independent~~ (c) fully random
- $\Pr[h(x) = h(y)] \leq \frac{1}{n}$ S_{ij}, S_{ik}, S_{ie}

Give one to two sentences explaining your choice.

$$\Pr[A[h_i(w)] = 1] = \frac{1}{3}$$



- Problem Set 2 is posted and due next Friday 10/16 at 11:59pm.
- Midterm 1 is **in class** Monday, 10/19.
- Study guide will be posted this week.
- Past midterms are posted in Canvas. Note that most (pre S26) cover a bit more material than we have seen – e.g., we will not cover the **Johnson-Lindenstrauss lemma** or **low-distortion embeddings** before Midterm 1.
- If you need extended time on the midterm, please reach out by the end of this week to make a plan.

Summary

Last Class:

- Overview of streaming algorithms.
- The (ϵ, k) -frequent items problem and its applications.
- Count-Min sketch algorithm for frequent items.

n items, $k, \epsilon > 0$
at least any item appearing $\geq \frac{n}{k}$ times
don't output any item ... $< (1-\epsilon)\frac{n}{k}$ times

Summary

Last Class:

- Overview of streaming algorithms.
- The (ϵ, k) -frequent items problem and its applications.
- Count-Min sketch algorithm for frequent items.

This Class:

- Finish up Count-Min sketch
- Distinct items counting via min-hashing.
- ✳ Success probability boosting via the **median trick**.

Frequent Elements

Frequent Elements

(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. 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.

Frequent Elements

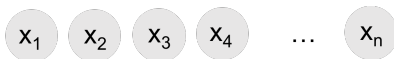
(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. Return a set F of items, including all items that appear at least $\frac{n}{k}$ times and only items that appear at least $(1 - \epsilon) \cdot \frac{n}{k}$ times.

Solution Count-Min Sketch:

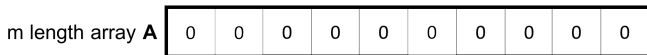
Frequent Elements

(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. 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.

Solution Count-Min Sketch:



random hash function h

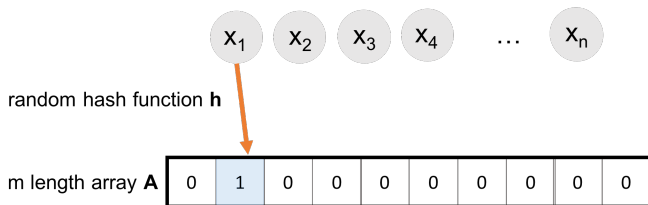


Use $A[h(x)]$ to estimate $f(x)$, the frequency of x in the stream.
i.e., $|\{x_i : x_i = x\}|$.

Frequent Elements

(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. 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.

Solution Count-Min Sketch:

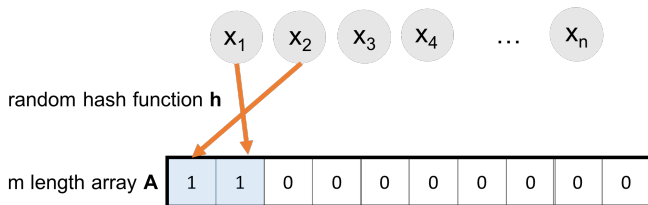


Use $A[h(x)]$ to estimate $f(x)$, the frequency of x in the stream.
i.e., $|\{x_i : x_i = x\}|$.

Frequent Elements

(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. 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.

Solution Count-Min Sketch:

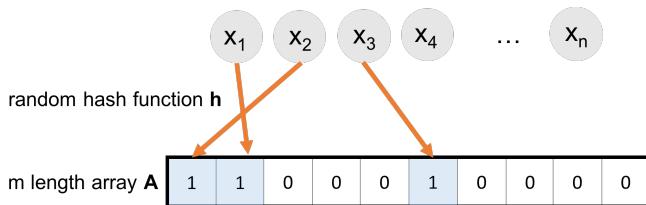


Use $A[h(x)]$ to estimate $f(x)$, the frequency of x in the stream.
i.e., $|\{x_i : x_i = x\}|$.

Frequent Elements

(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. 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.

Solution Count-Min Sketch:

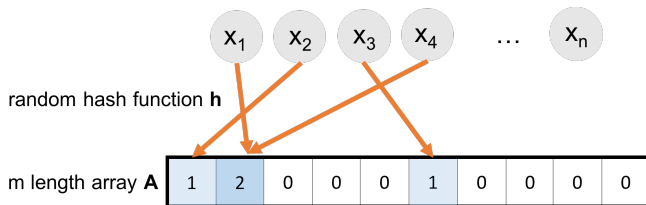


Use $A[h(x)]$ to estimate $f(x)$, the frequency of x in the stream.
i.e., $|\{x_i : x_i = x\}|$.

Frequent Elements

(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. 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.

Solution Count-Min Sketch:

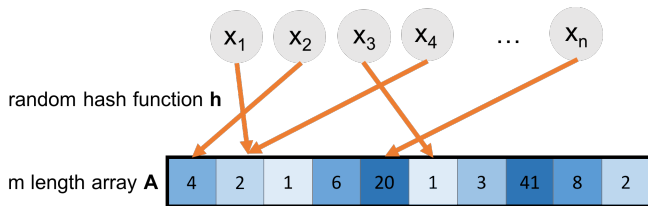


Use $A[h(x)]$ to estimate $f(x)$, the frequency of x in the stream.
I.e., $|\{x_i : x_i = x\}|$.

Frequent Elements

(ϵ, k) -Frequent Items Problem: Consider a stream of n items $x_1, \dots, x_n$. 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.

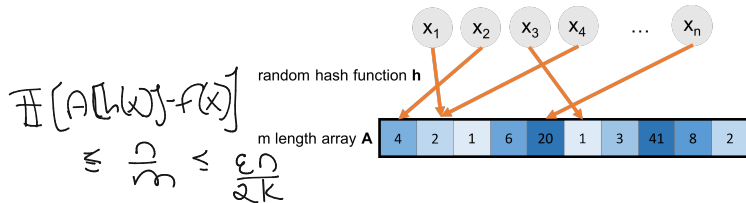
Solution Count-Min Sketch:



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

$$A[h(x)] \geq f(x)$$

Count-Min Sketch Accuracy



Claim: For any x , with probability at least $1/2$, if we set $m = \frac{2k}{\epsilon}$,

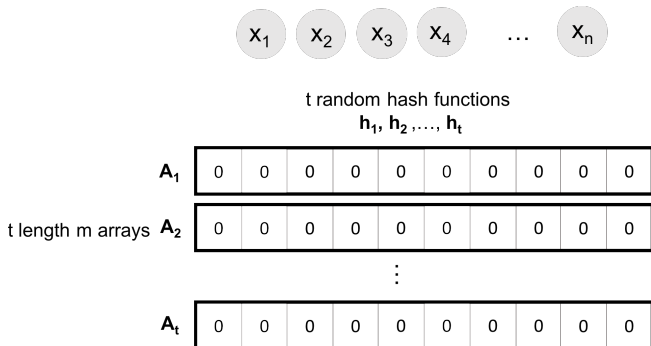
$$f(x) \leq A[h(x)] \leq f(x) + \frac{\epsilon n}{k}$$

this error is small enough to distinguish frequent from infrequent items.

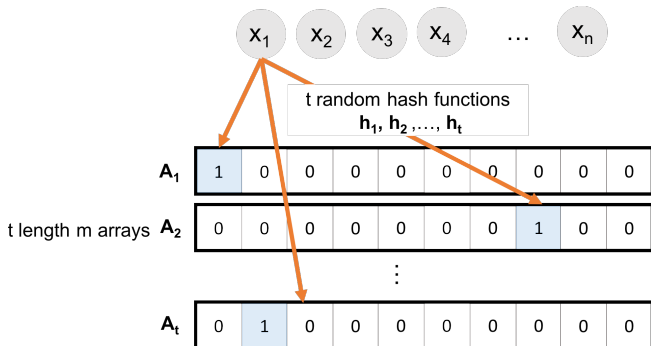
Accurate enough to identify (ϵ, k) -frequent elements. How can we improve the success probability? **Repetition.**

$f(x)$: 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.

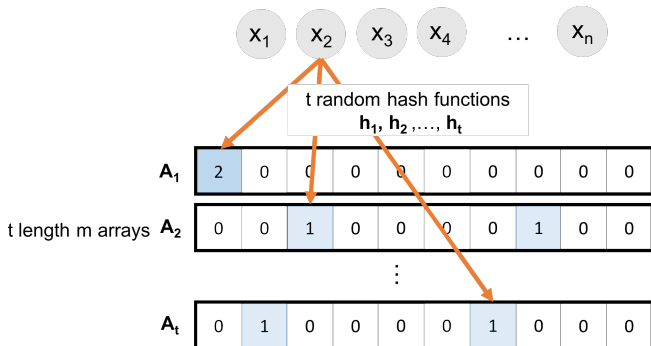
Count-Min Sketch Repetition



Count-Min Sketch Repetition

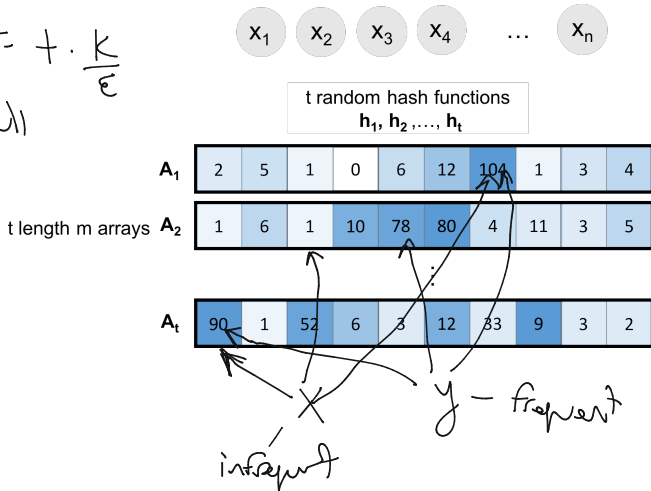


Count-Min Sketch Repetition

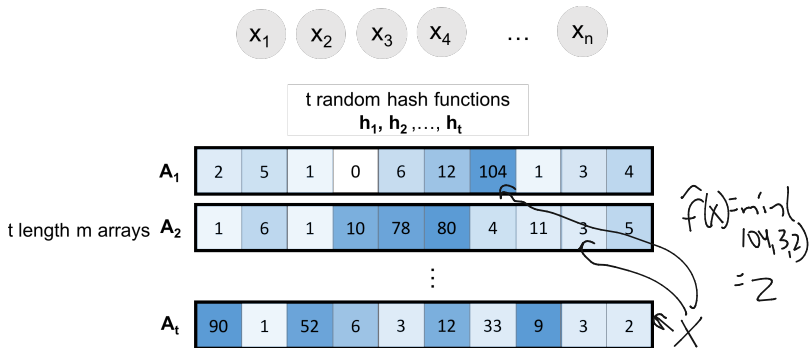


Count-Min Sketch Repetition

space
 $t \cdot m = t \cdot \frac{k}{\epsilon}$
very small

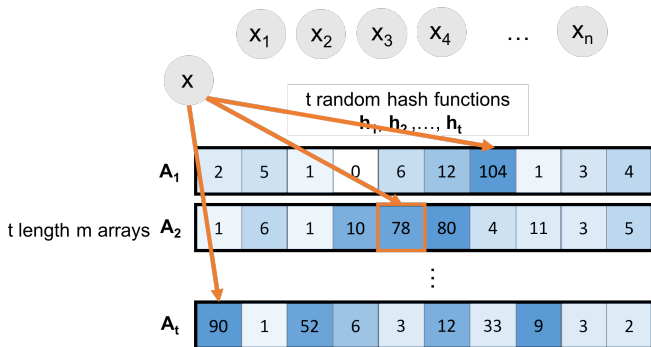


Count-Min Sketch Repetition



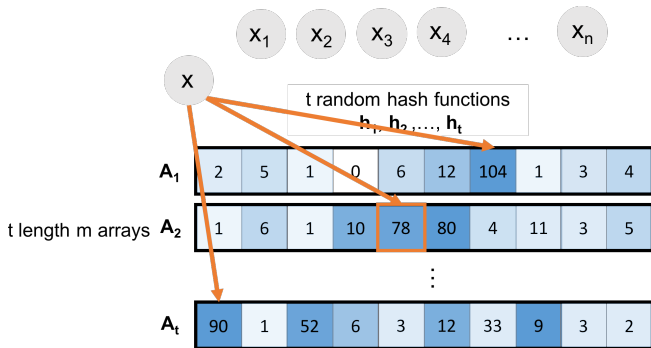
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (Count-min sketch)

Count-Min Sketch Repetition



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

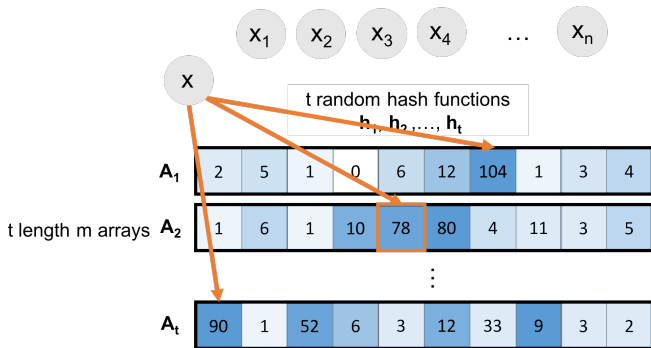
Count-Min Sketch Repetition



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

Why min instead of taking the average?

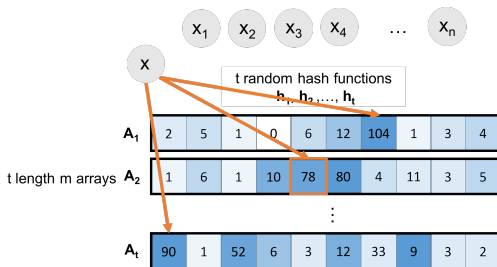
Count-Min Sketch Repetition



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

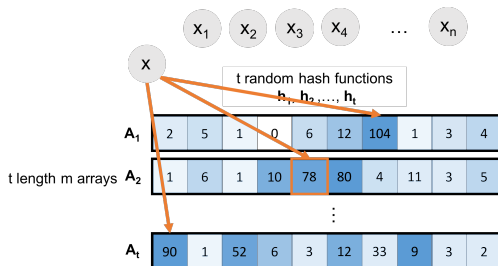
Why min instead of taking the average? The minimum estimate is always the most accurate since they are all overestimates of the true frequency.

Count-Min Sketch Analysis



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

Count-Min Sketch Analysis

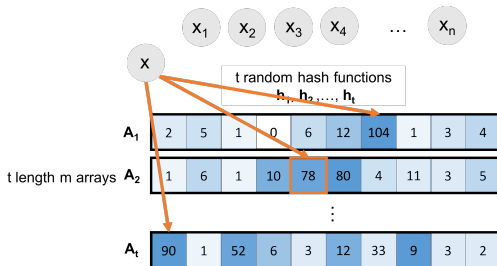


Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$

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

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

Count-Min Sketch Analysis



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

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

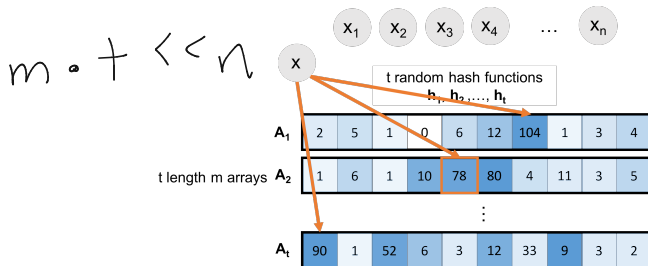
$$f(x) \leq A_i[h_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

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

always holds

$$= \Pr\left[f(x) \leq f(x) + \frac{\epsilon n}{k}\right] = 1 - \Pr\left[\tilde{f}(x) > f(x) + \frac{\epsilon n}{k}\right] \geq 1 - \frac{1}{2^t} = 1 - \Pr\left[\bigcap_{i=1}^t A_i[h_i(x)] > f(x) + \frac{\epsilon n}{k}\right] \approx 1 - \frac{1}{2^t}$$

Count-Min Sketch Analysis



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

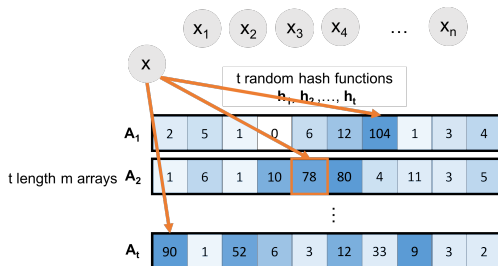
- For every x and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$ with probability $\geq 1/2$:

$$f(x) \leq A_i[h_i(x)] \leq f(x) + \frac{\epsilon n}{k}$$

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

$$\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}] = 1 - \Pr[\tilde{f}(x) \geq f(x) + \frac{\epsilon n}{k}] \leq \frac{1}{2^t}$$

Count-Min Sketch Analysis



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

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

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

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

$$\frac{1}{2} + \dots = \delta$$

- 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 $O(\log(1/\delta) \cdot k/\epsilon)$ space.

to m

(independent of n)

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 – distinguish between items with frequency $\frac{n}{k}$ and those with frequency $(1 - \epsilon)\frac{n}{k}$.

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 $O(\log(1/\delta) \cdot k/\epsilon)$ space.

- Accurate enough to solve the (ϵ, k) -Frequent elements problem – distinguish 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?

↳ union bound

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:

1 1 2 3 4 5
1 drop 1

- 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.

$i = n$

Distinct Elements

Distinct Elements

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

E.g.,

1, 5, 7, 5, 2, 1 $\rightarrow$ 4 distinct elements

Distinct Elements

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

E.g.,

1, 5, 7, 5, 2, 1 $\rightarrow$ 4 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 $x_1, \dots, x_n$, estimate the number of distinct elements in the stream.

E.g.,

1, 5, 7, 5, 2, 1 $\rightarrow$ 4 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 $x_1, \dots, x_n$, estimate the number of distinct elements.

— Store a set (bloom filter)
 ↳ when a new item comes in
 increment counter if not in set
 ↳ $O(d)$

$$-(1-\epsilon)d \leq \tilde{d} \leq (1+\epsilon)d$$

$$.95d \leq \tilde{d} < 1.05d$$

Hashing for Distinct Elements

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

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

- Let $h : U \rightarrow [0, 1]$ be a random hash function (with a real valued output)

$$x_1 \rightarrow 0.98765 \dots$$

- $s := 1$

$$x_2 \rightarrow 0.0351 \dots$$

- For $i = 1, \dots, n$

- $s := \min(s, h(x_i))$

$$x_3 \rightarrow 0.3765$$

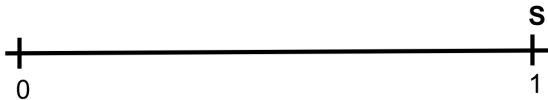
- Return $\tilde{d} = \frac{1}{s} - 1$

Hashing for Distinct Elements

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

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

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

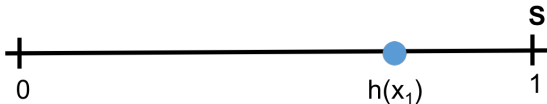


Hashing for Distinct Elements

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

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

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

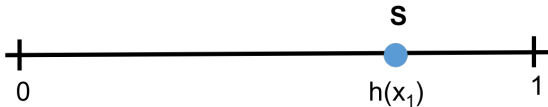


Hashing for Distinct Elements

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

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

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

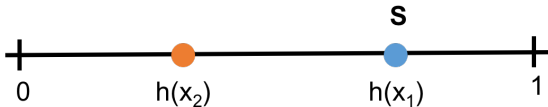


Hashing for Distinct Elements

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

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

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

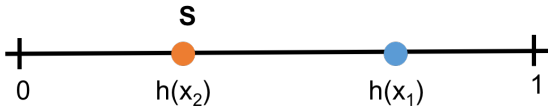


Hashing for Distinct Elements

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

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

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

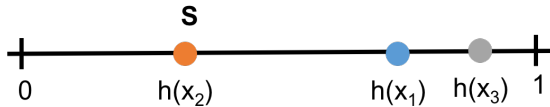


Hashing for Distinct Elements

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

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

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



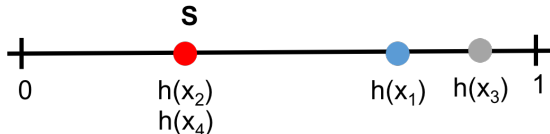
Hashing for Distinct Elements

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

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

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

$$x_4 = x_2$$

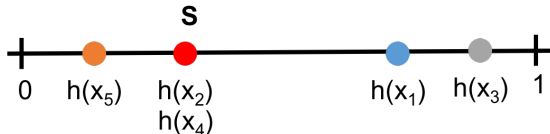


Hashing for Distinct Elements

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

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

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

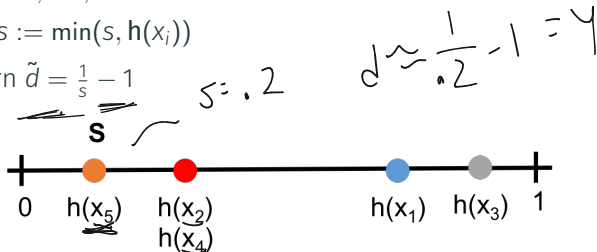


Hashing for Distinct Elements

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

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

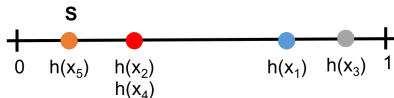
- Let $h : U \rightarrow [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \dots, n$
 - $s := \min(s, h(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Hashing for Distinct Elements

Min-Hashing for Distinct Elements:

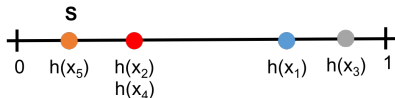
- Let $h : U \rightarrow [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \dots, n$
 - $s := \min(s, h(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Hashing for Distinct Elements

Min-Hashing for Distinct Elements:

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

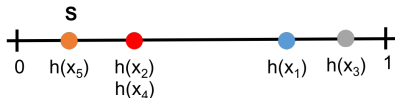


- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0, 1]$. Where
 $d = \#$ distinct elements.

Hashing for Distinct Elements

Min-Hashing for Distinct Elements:

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

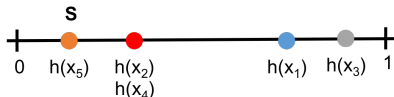


- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0, 1]$. Where $d = \#$ distinct elements.
- Intuition: The larger d is, the smaller we expect s to be.

Hashing for Distinct Elements

Min-Hashing for Distinct Elements:

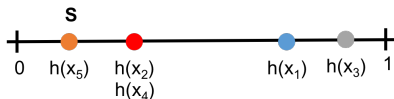
- Let $h : U \rightarrow [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \dots, n$
 - $s := \min(s, h(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



- 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

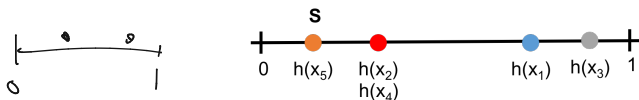
s is the minimum of d points chosen uniformly at random on $[0, 1]$.
Where $d = \#$ distinct elements.



Performance in Expectation

s is the minimum of d points chosen uniformly at random on $[0, 1]$.

Where $d = \#$ distinct elements.

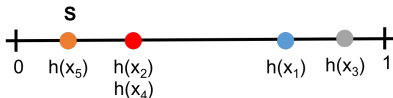


$$\mathbb{E}[s] = \frac{1}{d+1}$$

$$\begin{aligned} d &= 1 \\ \mathbb{P}[s] &= \frac{1}{2} \\ d &= 2 \\ \mathbb{P}[s] &= \frac{1}{3} \end{aligned}$$

Performance in Expectation

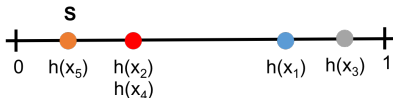
s is the minimum of d points chosen uniformly at random on $[0, 1]$.
Where $d = \#$ distinct elements.



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

Performance in Expectation

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



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

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

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

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