# COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Spring 2026.

Lecture 7

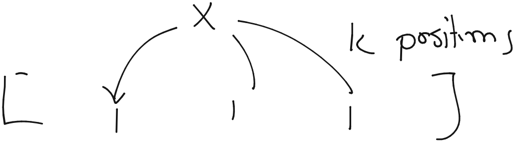
Logistics: - Pset 2 Released. Due Sunday 3/8

- Quiz 4 will be released this evening.
- Midterm 1 in two weeks 3/12. (in class)

# Summary

Last Class:

- Bloom filters for storing a set with a small false positive rate.
- Use roughly $O(n \log(1/\delta))$ bits of space to achieve false positive rate $\delta$, as compared to $n \cdot (\textit{item size})$ space for exact storage.




$\delta = .05$

$\log(1/\delta) = \log(20) \approx 6$ bits of space per item

# Summary

**Last Class:**

- Bloom filters for storing a set with a small false positive rate.
- Use roughly $O(n \log(1/\delta))$ bits of space to achieve false positive rate $\delta$, as compared to $n \cdot (\textit{item size})$ space for exact storage.

**This Class:**

- Intro to streaming algorithms – computation when space is limited.
- Frequent items estimation via Count-Min sketch

# Streaming Algorithms

**Stream Processing:** Have a massive dataset $X$ with $n$ items $x_1, x_2, \ldots, x_n$ that arrive in a continuous stream. Not nearly enough space to store all the items (in a single location).

- Still want to analyze and learn from this data.
- Typically must compress the data on the fly, storing a data structure from which you can still learn useful information.
- Often the compression is randomized. E.g., bloom filters.
- Compared to traditional algorithm design, which focuses on minimizing runtime, the big question here is how much space is needed to answer queries of interest.

# Some Examples

- **Sensor data:** images from telescopes (30 terabytes per night from the Vera C. Rubin Observatory), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors, electrical grid monitoring.

# Some Examples

- **Sensor data:** images from telescopes (30 terabytes per night from the Vera C. Rubin Observatory), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors, electrical grid monitoring.
- **Internet Traffic**: 8.5 billion Google searches, billions of ad-clicks and other logs from instrumented webpages, IPs routed by network switches, ...

## Some Examples

- Sensor data: images from telescopes (30 terabytes per night from the Vera C. Rubin Observatory), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors, electrical grid monitoring.
- Internet Traffic: 8.5 billion Google searches, billions of ad-clicks and other logs from instrumented webpages, IPs routed by network switches, ...
- Datasets in Machine Learning: When training e.g. a neural network on a large dataset (ImageNet with 14 million images or LLaMA-2 on trillions of tokens of text), the data is typically processed in a stream due to storage limitations.

# Some Examples

- **Sensor data:** images from telescopes (30 terabytes per night from the Vera C. Rubin Observatory), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors, electrical grid monitoring.
- **Internet Traffic**: 8.5 billion Google searches, billions of ad-clicks and other logs from instrumented webpages, IPs routed by network switches, ...
- **Datasets in Machine Learning:** When training e.g. a neural network on a large dataset (ImageNet with 14 million images or LLaMA-2 on trillions of tokens of text), the data is typically processed in a stream due to storage limitations.

# The Frequent Items Problems

$k$-Frequent Items (Heavy-Hitters) Problem: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

$k = 10$

# The Frequent Items Problems

$k$-Frequent Items (Heavy-Hitters) Problem: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|
| 5 | 12 | 3 | 3 | 4 | 5 | 5 | 10 | 3 |

# The Frequent Items Problems

**$k$-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return ~~any~~ all items ~~at~~ that appear~~s~~ at least $\frac{n}{k}$ times.

$k = 3$

$9/3 = 3$ time

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|
| 5 | 12 | 3 | 3 | 4 | 5 | 5 | 10 | 3 |

$\{5, 3\}$

# The Frequent Items Problems

**$k$-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

$\boxed{k << n}$

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|
| 5 | 12 | 3 | 3 | 4 | 5 | 5 | 10 | 3 |

- What is the maximum number of items that can be returned?

  a) $n$ b) $k$ c ) $n/k$ d) $\log n$

# The Frequent Items Problems

*k*-**Frequent Items (Heavy-Hitters) Problem**: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

5 : 3
10 : 1
3 : 3
12 : 1
4 : 1

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|
| 5 | 12 | 3 | 3 | 4 | 5 | 5 | 10 | 3 |

- What is the maximum number of items that can be returned?
  a) $n$ b) $k$ c) $n/k$ d) $\log n$

- Trivial with $O(n)$ space – store the count for each item and return the one that appears $\geq n/k$ times.
- Can we do it with less space? I.e., without storing all $n$ items?

# The Frequent Items Problem

Applications of Frequent Items:

- Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
- Finding very frequent IP addresses sending requests (to detect DoS attacks/network anomalies).
- ‘Iceberg queries’ for all items in a database with frequency above some threshold.

Generally want very fast detection, without having to scan through database/logs. I.e., want to maintain a running list of frequent items that appear in a stream.

# Frequent Itemset Mining

**Association rule learning:** A very common task in data mining is to identify common associations between different events.

mmDS


2

- Identified via frequent itemset counting. Find all sets of $t$ items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.

# Frequent Itemset Mining

**Association rule learning:** A very common task in data mining is to identify common associations between different events.


- Identified via frequent itemset counting. Find all sets of $t$ items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.

# Approximate Frequent Elements

**Issue:** No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency $n/k$ (should be output) and $n/k - 1$ (should not be output).

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | ... | $x_{n-n/k+1}$ | ... | $x_n$ |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 12 | 9 | 27 | 4 | 101 | | 3 | | 3 |

n/k-1 occurrences

# Approximate Frequent Elements

**Issue:** No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency $n/k$ (should be output) and $n/k - 1$ (should not be output).

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | ... | $x_{n-n/k+1}$ | ... | $x_n$ |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 12 | 9 | 27 | 4 | 101 | ... | 3 | ... | 3 |

n/k-1 occurrences

**$(\epsilon, k)$-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.

# Approximate Frequent Elements

strictly less than $O(n)$

**Issue:** No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency $n/k$ (should be output) and $n/k - 1$ (should not be output).

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | ... | $x_{n-n/k+1}$ | ... | $x_n$ |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 12 | 9 | 27 | 4 | 101 | ... | 3 | ... | 3 |

n/k-1 occurrences

**$(\epsilon, k)$-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.

- An example of relaxing to a 'promise problem': for items with frequencies in $[(1-\epsilon) \cdot \frac{n}{k}, \frac{n}{k}]$ no output guarantee.

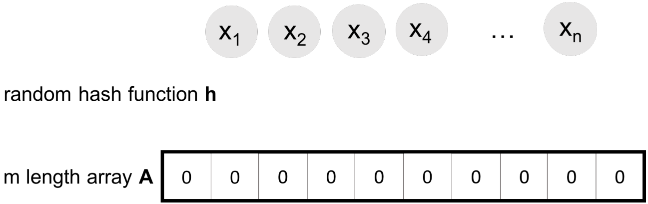
frequent items | $\frac{n}{k}$ borderline items | $(1-\varepsilon)\frac{n}{k}$ infrequent items

# Frequent Elements with Count-Min Sketch

**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

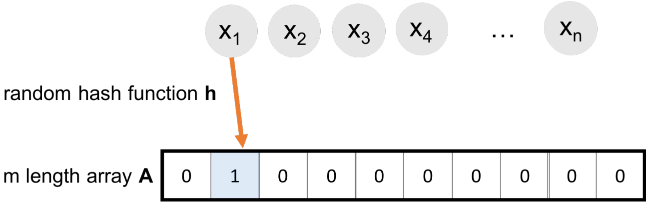
# Frequent Elements with Count-Min Sketch

**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.



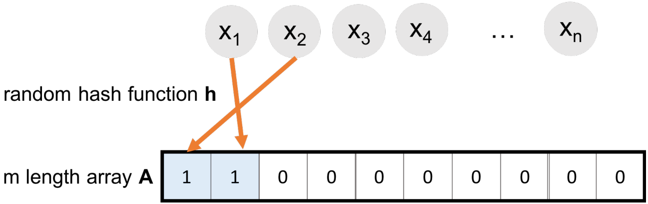
# Frequent Elements with Count-Min Sketch

**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.



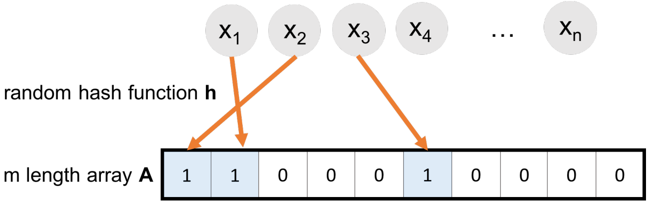
# Frequent Elements with Count-Min Sketch

**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

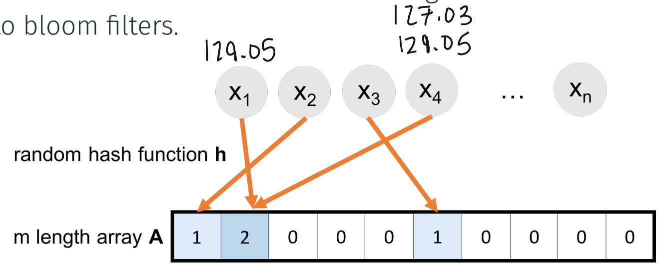


# Frequent Elements with Count-Min Sketch

**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

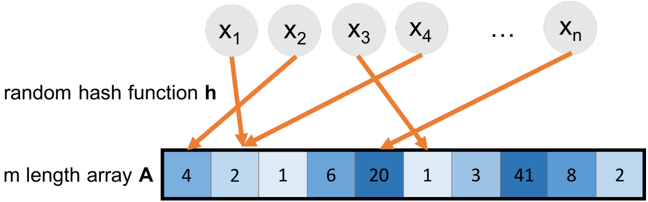


**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.



# Frequent Elements with Count-Min Sketch

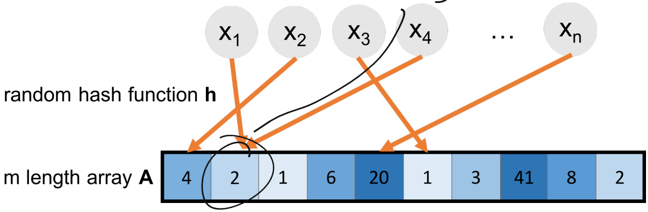
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.



# Frequent Elements with Count-Min Sketch

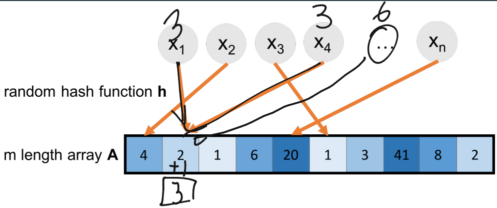
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

129.05




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




$f(3)=2$

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

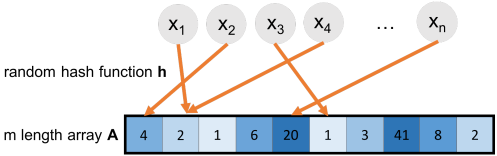
$h(3)=h(6)$

**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 $x$). $\mathrm{h}$: random hash function. $m$: size of Count-min sketch array.

# Count-Min Sketch Accuracy

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

Expected Error:

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

$f(x)$: frequency of $x$ in the stream (i.e., number of items equal to $x$). $\mathsf{h}$: 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:

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

$$\Pr(h(y) = h(x)) \cdot f(y) + \Pr(h(y) \neq h(x)) \cdot 0$$

$f(x)$: 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

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

Expected Error:

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

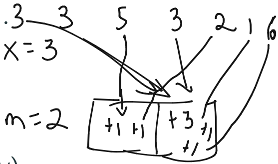
$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

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

$f(x)$: 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

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




**Expected Error:**

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

$$\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1$$

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

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

$$\mathbb{E}[E(x)] \leq \frac{n}{m} \qquad \Pr\left(E(x) \geq \frac{2n}{m}\right) \leq \frac{\frac{n}{m}}{\frac{2n}{m}} \leq \frac{1}{2}$$

$$\mathbb{E}\Big[\sum_{y \neq x} f(y) \cdot \mathbb{1}(h(x) = h(y))\Big] = \sum_{y \neq x} f(y) \cdot \mathbb{E}[\mathbb{1}(h(x) = h(y))] = \sum_{y \neq x} f(y) \cdot \Pr(h(y) = h(x))$$

$f(x)$: 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:**

$$\mathbb{E}\left[\sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y)\right] = \sum_{y \neq x} \underbrace{\Pr(\mathbf{h}(y) = \mathbf{h}(x))}_{\leq \frac{1}{m}} \cdot f(y)$$

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

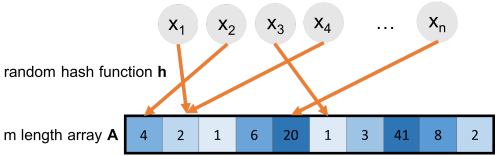
What is a bound on probability that the error is $\geq \frac{2n}{m}$?

**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 $x$). **h**: random hash function. $m$: size of Count-min sketch array.

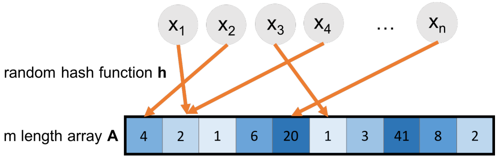
# Count-Min Sketch Accuracy




**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.




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

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

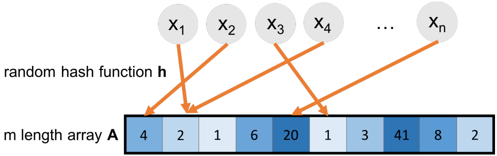
To solve the $(\epsilon, k)$-Frequent elements problem, set $\boxed{m = \frac{2k}{\epsilon}.}$

$$\frac{2n}{m} = \frac{2n}{\left(\frac{2k}{\varepsilon}\right)} = \frac{\varepsilon n}{k}$$

$\geq \frac{n}{k}$ $\leq (1-\varepsilon)\frac{n}{k}$

$f(x)$: 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



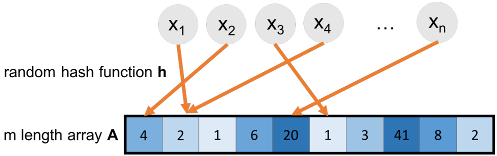

**Claim:** For any $x$, with probability at least 1/2,

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

To solve the $(\epsilon, k)$-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



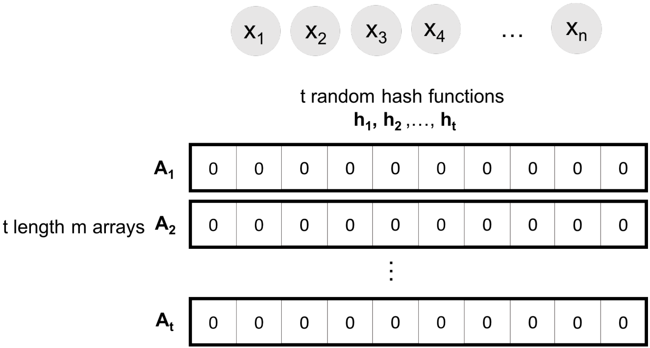

**Claim:** For any $x$, with probability at least 1/2,

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

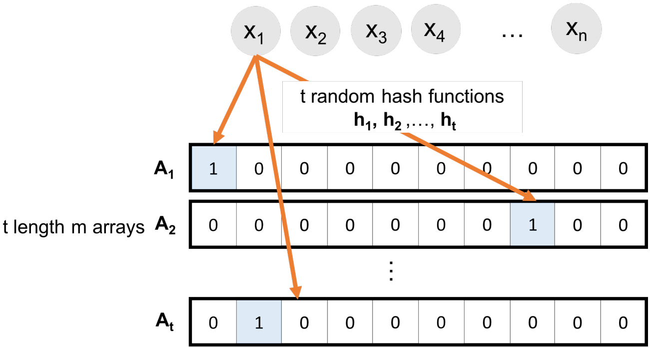
To solve the $(\epsilon, k)$-Frequent elements problem, set $m = \frac{2k}{\epsilon}$. How can we improve the success probability? Repetition.

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

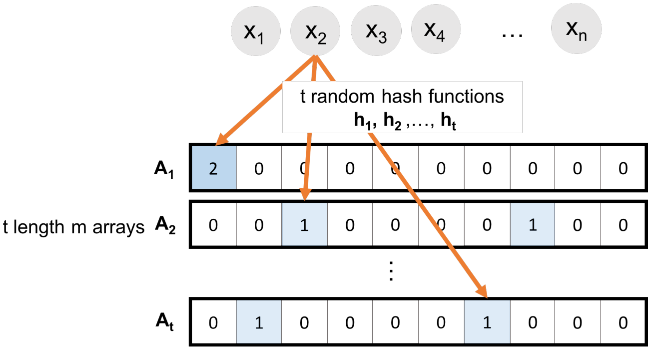
# Count-Min Sketch Repetition



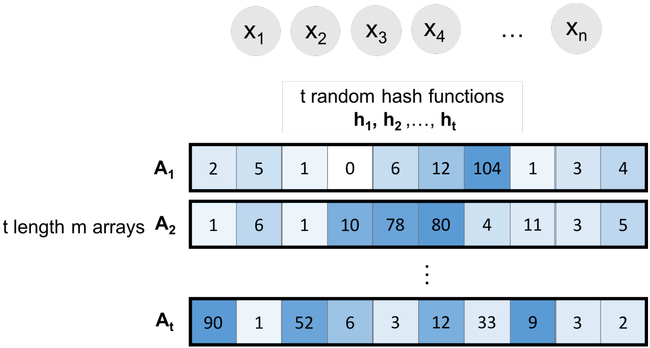
# Count-Min Sketch Repetition



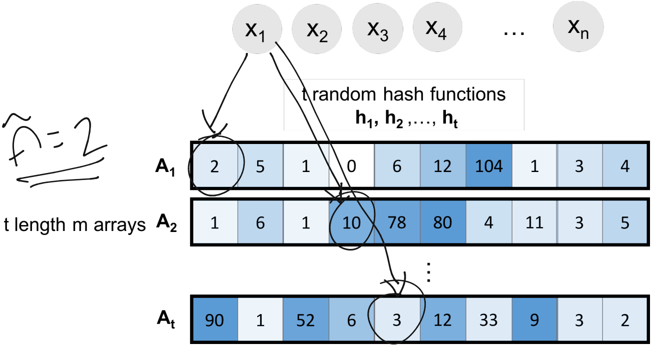
# Count-Min Sketch Repetition



# Count-Min Sketch Repetition

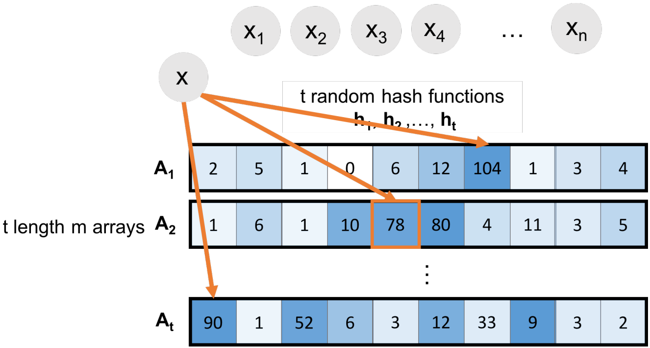


# Count-Min Sketch Repetition



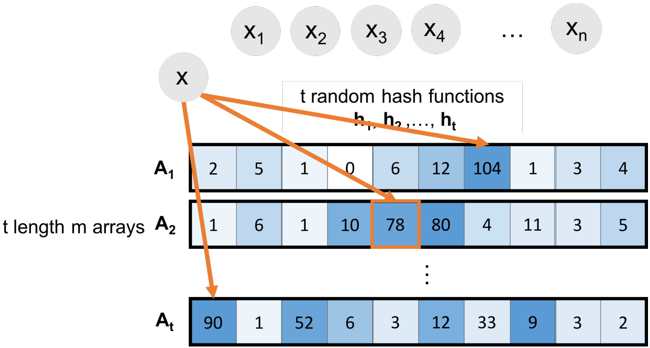

Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$. (Count-min sketch)

# Count-Min Sketch Repetition




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

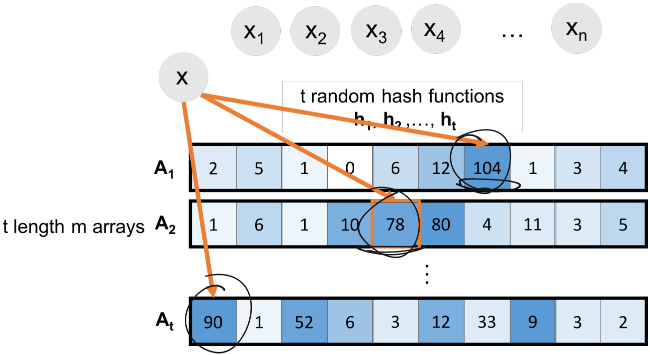
# Count-Min Sketch Repetition




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?

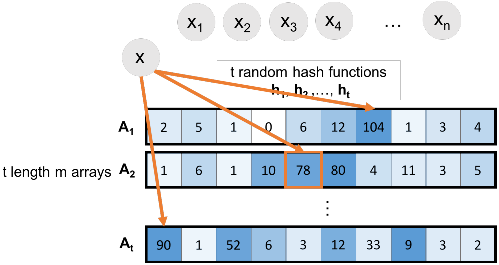
# Count-Min Sketch Repetition




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

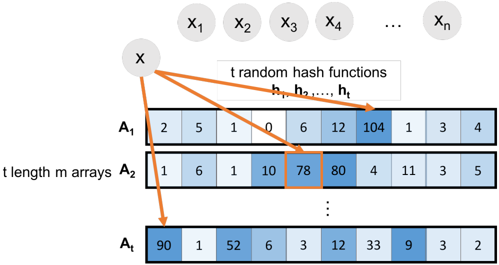
Why min instead of taking the average? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!

# Count-Min Sketch Analysis



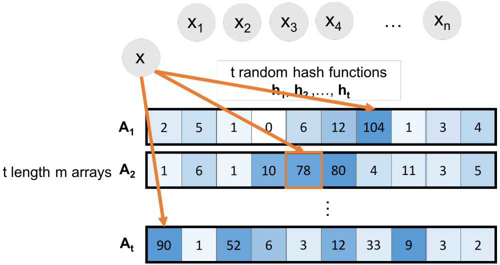

Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

# 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 $\geq 1/2$:
$$f(x) \leq A_i[\mathbf{h}_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$




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 $\geq 1/2$:

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

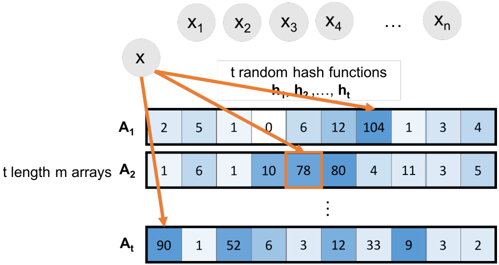
$E$

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

$Pr(E) = 1 - Pr(\bar{E}) = 1 - Pr(\bar{E}_1 \cap \bar{E}_2 \cap .. \bar{E}_t)$

$= 1 - \frac{1}{2^t}$

# Count-Min Sketch Analysis

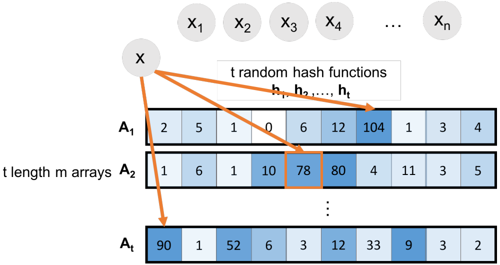



Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[\mathbf{h}_i(x)]$

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

# 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 $\geq 1/2$:

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

$\frac{1}{2}^t = \delta$

- 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 $t = \log_2(1/\delta)$.

enough to distinguish $\geq \frac{n}{k}$
$\leq (1-\varepsilon)\frac{n}{k}$

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

**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 $(\epsilon, k)$-Frequent elements problem – distinquish between items with frequency $\frac{n}{k}$ and those with frequency $(1-\epsilon)\frac{n}{k}$.

Example $k=2$

1 2 3 1 4 2 1 5 6 1 10 12 1 7 1 1

$n/2 \mid +n/4 \mid +n/4 \mid +n/4$

$O(\log(n/.01) \cdot k/\epsilon) = O(\log n \cdot k/\epsilon)$

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

$E_i$ be event get a bad estimate for $x_i$

$$\delta = \frac{.01}{n}$$

$$\Pr\left(\bigcup_{i=1}^{n} E_i\right) \leq \sum_{i=1}^{n} \Pr(E_i) = \frac{.01}{n} \cdot n \leq .01$$

# 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?

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:

counting bloom filter

- 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.