COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor
Lecture 9
MinHash maps sets to $[0, 1]$ such that for any two sets:

$$\Pr(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Basic approach to speeding up nearest neighbor search:
  - Combine MinHash with a function $g: [0, 1] \rightarrow [m]$.
  - Store each set $A$ in location $g(\text{MinHash}(A))$ of a table.
  - When we want to find sets similar to $B$, only compare $B$ to sets stored in location $g(\text{MinHash}(B))$ of the table.
  - Assuming $g$ has no collisions, the probability that one of the stored sets $A$ is found in this location is $J(A, B)$, i.e., near one for similar sets and near zero for dissimilar sets.
• MinHash maps set to $[0, 1]$ such that for any two sets:

$$Pr(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
• MinHash maps set to $[0, 1]$ such that for any two sets:

$$\Pr(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

• Basic approach to speeding up nearest neighbor search:
  • Combine MinHash with a function $g : [0, 1] \rightarrow [m]$
  • Store each set $A$ in location $g(\text{MinHash}(A))$ of a table.
  • When we want to find sets similar to $B$, only compare $B$ to sets stored in location $g(\text{MinHash}(B))$ of the table.
  • Assuming $g$ has no collisions, the probability that one of the stored sets $A$ is found in this location is $J(A, B)$, i.e., near one for similar sets and near zero for dissimilar sets.
We want to balance a small probability of false negatives (a high hit rate) with a small probability of false positives (a small query time.)
We want to balance a small probability of false negatives (a high hit rate) with a small probability of false positives (a small query time.)

Create $t$ hash tables. Each is indexed into not with a single MinHash value, but with $r$ values, appended together. A length $r$ signature.
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$. 
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$. 

![Diagram](image-url)
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$. 

![Graph showing the hit probability for different Jaccard similarities with $r = 10$ and $t = 10$.](image-url)
Using \( t \) repetitions each with a signature of \( r \) MinHash values, the probability that \( x \) and \( y \) with Jaccard similarity \( J(x, y) = s \) match in at least one repetition is: \( 1 - (1 - s^r)^t \).
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.

$r$ and $t$ are tuned depending on application.
For example: Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$. 

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are 10,000 near matches with $J(x, y) \in [.7, .9]$.
- With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y)$ is $1 - (1 - .9)^{25/50}$.
- Hit probability for $J(x, y) \geq .9$ is $\geq 1 - (1 - .9)^{20/40} \approx .98$.
- Hit probability for $J(x, y) \leq .7$ is $\leq 1 - (1 - .7)^{20/40} \approx .007$.

Expected Number of Items Scanned: (proportional to query time) $\leq 10 + .98 \times 10,000 + .007 \times 9,989,990 \approx 80,000 \ll 10,000,000$. 

$S$-CURVE EXAMPLE
For example: Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$.

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are 10,000 near matches with $J(x, y) \in [.7, .9]$. 

Expected Number of Items Scanned: (proportional to query time) $\leq 10 + .98 \times 10,000 + .007 \times 9,990 \approx 80,000 \ll 10,000,000$. 

**S-Curve Example**

**For example:** Consider a database with 10,000,000 audio clips. You are given a clip \( x \) and want to find any \( y \) in the database with \( J(x, y) \geq 0.9 \).

- There are 10 **true matches** in the database with \( J(x, y) \geq 0.9 \).
- There are 10,000 **near matches** with \( J(x, y) \in [0.7, 0.9] \).

With signature length \( r = 25 \) and repetitions \( t = 50 \), hit probability for \( J(x, y) = s \) is \( 1 - (1 - s^{25})^{50} \).
For example: Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$.

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are 10,000 near matches with $J(x, y) \in [.7, .9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y) = s$ is $1 - (1 - s^{25})^{50}$.

- Hit probability for $J(x, y) \geq .9$ is $\geq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \in [.7, .9]$ is $\leq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \leq .7$ is $\leq 1 - (1 - .7^{20})^{40} \approx .007$
**S-curve example**

For example: Consider a database with 10,000,000 audio clips. You are given a clip \( x \) and want to find any \( y \) in the database with \( J(x, y) \geq .9 \).

- There are 10 true matches in the database with \( J(x, y) \geq .9 \).
- There are 10,000 near matches with \( J(x, y) \in [.7, .9] \).

With signature length \( r = 25 \) and repetitions \( t = 50 \), hit probability for \( J(x, y) = s \) is \( 1 - (1 - s^{25})^{50} \).

- Hit probability for \( J(x, y) \geq .9 \) is \( \geq 1 - (1 - .9^{20})^{40} \approx .98 \)
- Hit probability for \( J(x, y) \in [.7, .9] \) is \( \leq 1 - (1 - .9^{20})^{40} \approx .98 \)
- Hit probability for \( J(x, y) \leq .7 \) is \( \leq 1 - (1 - .7^{20})^{40} \approx .007 \)

**Expected Number of Items Scanned:** (proportional to query time)

\[
\leq 10 + .98 \times 10,000 + .007 \times 9,989,990 \approx 80,000
\]
For example: Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$.

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are 10,000 near matches with $J(x, y) \in [.7, .9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y) = s$ is $1 - (1 - s^{25})^{50}$.

- Hit probability for $J(x, y) \geq .9$ is $\geq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \in [.7, .9]$ is $\leq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \leq .7$ is $\leq 1 - (1 - .7^{20})^{40} \approx .007$

**Expected Number of Items Scanned:** (proportional to query time)

\[
\leq 10 + .98 \times 10,000 + .007 \times 9,989,990 \approx 80,000 \ll 10,000,000.
\]
Repetition and s-curve tuning can be used for fast similarity search with other similarity metrics:

Cosine Similarity:

$$\cos(\theta(x, y)) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}.$$
Repetition and s-curve tuning can be used for fast similarity search with other similarity metrics:

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.
Repetition and s-curve tuning can be used for fast similarity search with other similarity metrics:

• LSH schemes exist for many similarity/distance measures: hamming distance, \textit{cosine similarity}, etc.
Repetition and s-curve tuning can be used for fast similarity search with other similarity metrics:
- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

\[ \cos(\theta(x, y)) = 1 \text{ when } \theta(x, y) = 0^\circ \text{ and } \cos(\theta(x, z)) = 0 \text{ when } \theta(x, z) = 90^\circ, \text{ and } \cos(\theta(x, z)) = -1 \text{ when } \theta(x, z) = 180^\circ. \]
Repetition and $s$-curve tuning can be used for fast similarity search with other similarity metrics:

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

**Cosine Similarity:** $\cos(\theta(x, y))$
Repetition and $s$-curve tuning can be used for fast similarity search with other similarity metrics:

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

**Cosine Similarity:** $\cos(\theta(x, y))$

- $\cos(\theta(x, y)) = 1$ when $\theta(x, y) = 0^\circ$ and $\cos(\theta(x, y)) = 0$ when $\theta(x, y) = 90^\circ$, and $\cos(\theta(x, y)) = -1$ when $\theta(x, y) = 180^\circ$
Repetition and s-curve tuning can be used for fast similarity search with other similarity metrics:

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

**Cosine Similarity:** \( \cos(\theta(x, y)) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2} \).

- \( \cos(\theta(x, y)) = 1 \) when \( \theta(x, y) = 0^\circ \) and \( \cos(\theta(x, y)) = 0 \) when \( \theta(x, y) = 90^\circ \), and \( \cos(\theta(x, y)) = -1 \) when \( \theta(x, y) = 180^\circ \)
SimHash Algorithm: LSH for cosine similarity.
SimHash Algorithm: LSH for cosine similarity.

\[ \text{SimHash}(x) = \text{sign}(\langle x, t \rangle) \text{ for a random vector } t. \]

What is \( \text{Pr}[\text{SimHash}(x) = \text{SimHash}(y)] \)?
SimHash Algorithm: LSH for cosine similarity.

$\text{SimHash}(x) = \text{sign}(\langle x, t \rangle)$ for a random vector $t$. 

What is $Pr[\text{SimHash}(x) = \text{SimHash}(y)]$?
SimHash Algorithm: LSH for cosine similarity.
SimHash Algorithm: LSH for cosine similarity.

SimHash(x) = sign(⟨x, t⟩) for a random vector t.
SimHash Algorithm: LSH for cosine similarity.

\[ \text{SimHash}(x) = \text{sign}(\langle x, t \rangle) \] for a random vector \( t \).

What is \( \Pr[\text{SimHash}(x) = \text{SimHash}(y)] \)?
What is \( \Pr [\text{SimHash}(x) = \text{SimHash}(y)] \)?
What is \( \Pr [SimHash(x) = SimHash(y)] \)?

\( SimHash(x) \neq SimHash(y) \) when the plane separates \( x \) from \( y \).
What is \( \Pr [\text{SimHash}(x) = \text{SimHash}(y)] \)?

\( \text{SimHash}(x) \neq \text{SimHash}(y) \) when the plane separates \( x \) from \( y \).
What is $Pr[SimHash(x) = SimHash(y)]$?

$SimHash(x) \neq SimHash(y)$ when the plane separates $x$ from $y$.

- $Pr[SimHash(x) \neq SimHash(y)] = \frac{\theta(x,y)}{180}$ for small $\theta(x,y)$. 

---

**SIMHASH FOR COSINE SIMILARITY**
What is $\Pr[\text{SimHash}(x) = \text{SimHash}(y)]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$.

- $\Pr[\text{SimHash}(x) \neq \text{SimHash}(y)] = \frac{\theta(x,y)}{180}$
- $\Pr[\text{SimHash}(x) = \text{SimHash}(y)] = 1 - \frac{\theta(x,y)}{180} \approx \cos \theta$ for small $\theta$. 
Questions on MinHash and Locality Sensitive Hashing?
**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-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.

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( x_3 )</td>
<td>( x_4 )</td>
<td>( x_5 )</td>
<td>( x_6 )</td>
<td>( x_7 )</td>
<td>( x_8 )</td>
<td>( x_9 )</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>
$k$-Frequent Items (Heavy-Hitters) Problem: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item that appears at least $\frac{n}{k}$ times.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>
The Frequent Items Data Stream Problem

$k$-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.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- What is the maximum number of items that must be returned?
  - a) $n$
  - b) $k$
  - c) $n/k$
  - d) $\log n$
**k-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of \( n \) items \( x_1, \ldots, x_n \) (with possible duplicates). Return any item that appears at least \( \frac{n}{k} \) times.

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_6 )</th>
<th>( x_7 )</th>
<th>( x_8 )</th>
<th>( x_9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- What is the maximum number of items that must be returned?
  
a) \( n \)  
b) \( k \)  
c) \( n/k \)  
d) \( \log n \)
The Frequent Items Data Stream Problem: Consider a stream of \( n \) items \( x_1, \ldots, x_n \) (with possible duplicates). Return any item that appears at least \( \frac{n}{k} \) times.

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( x_3 )</td>
<td>( x_4 )</td>
<td>( x_5 )</td>
<td>( x_6 )</td>
<td>( x_7 )</td>
<td>( x_8 )</td>
<td>( x_9 )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

- What is the maximum number of items that must be returned?
  
  a) \( n \)  
  b) \( k \)  
  c) \( \frac{n}{k} \)  
  d) \( \log n \)

- Trivial with \( O(n) \) space: Store the count for each item and return the one that appears \( \geq \frac{n}{k} \) times.

- Can we do it with less space? I.e., without storing all \( n \) items?
Applications of Frequent Items:
Applications of Frequent Items:

• Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
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).
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.
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. That is we want to maintain a running list of frequent items that appear in a stream.
**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).
**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).

\[
\begin{array}{cccccc}
   x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \ldots & x_{n-k+1} & \ldots & x_n \\
   3   & 12  & 9   & 27  & 4   & 101 & \ldots & 3 & \ldots & 3 \\
\end{array}
\]
**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).

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>...</th>
<th>$x_{n-n/k+1}$</th>
<th>...</th>
<th>$x_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>9</td>
<td>27</td>
<td>4</td>
<td>101</td>
<td>...</td>
<td>3</td>
<td>...</td>
<td>3</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>...</th>
<th>$x_{n-n/k+1}$</th>
<th>...</th>
<th>$x_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>9</td>
<td>27</td>
<td>4</td>
<td>101</td>
<td>...</td>
<td>3</td>
<td>...</td>
<td>3</td>
</tr>
</tbody>
</table>

$n/k$-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.
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.

random hash function $h$

$m$ length array $A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

random hash function $h$

$m$ length array $A$

$X_1 \quad X_2 \quad X_3 \quad X_4 \quad \ldots \quad X_n$

$0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

A \[\{h(x)\}\] to estimate \(f(x)\), the frequency of \(x\) in the stream. I.e.,

\[|\{x_i : x_i = x\}|\]

m length array \(A\)

\[
\begin{array}{cccccccccc}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

random hash function \(h\)
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

![Diagram of Count-min sketch](image)

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

![Diagram of Count-min sketch](image)
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

```
\begin{array}{cccccc}
1 & 2 & 0 & 0 & 0 & 1 \\
\end{array}
```

random hash function \( h \)

m length array A

\( x_1 \quad x_2 \quad x_3 \quad x_4 \quad \ldots \quad x_n \)
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

![Diagram showing count-min sketch](image)

- Random hash function $h$ maps elements $x_1, x_2, x_3, x_4, \ldots, x_n$ to indices in an $m$-length array $A$.
- The count-min sketch estimates the frequency of elements in the stream by taking the minimum count at each index.

| $A$ | 4 | 2 | 1 | 6 | 20 | 1 | 3 | 41 | 8 | 2 |
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

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

Claim 1: We always have $A[h(x)] \geq f(x)$. Why?

---

$f(x)$: 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.
Use $A[h(x)]$ to estimate $f(x)$.

**Claim 1:** We always have $A[h(x)] \geq f(x)$. Why?

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

---

$f(x)$: 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.
Use $A[h(x)]$ to estimate $f(x)$.

**Claim 1:** We always have $A[h(x)] \geq f(x)$. Why?

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

---

$f(x)$: 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

\[ A[\mathbf{h}(x)] = f(x) + \sum_{y \neq x: \mathbf{h}(y) = \mathbf{h}(x)} f(y) \text{.} \]

\( f(x) \) : 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.
A[\mathbf{h}(x)] = f(x) + \sum_{y \neq x : \mathbf{h}(y) = \mathbf{h}(x)} f(y) .

\textbf{Expected Error:}

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

\( f(x) \): 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

\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \]

\textbf{Expected Error:}

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

- \( f(x) \): 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

\[ A[h(x)] = f(x) + \sum_{y \neq x : h(y) = h(x)} f(y). \]

**Expected Error:**

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

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

\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \]

**Expected Error:**

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

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

\( f(x) \): 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

\[
A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y)
\]

Expected Error:

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

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

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 \( x \)). \( h \): random hash function. \( m \): size of Count-min sketch array.
**COUNT-MIN SKETCH ACCURACY**

\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y). \]

**Expected Error:**

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

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

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

---

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

\[ A[h(x)] = f(x) + \sum_{y \neq x : h(y) = h(x)} f(y). \]

**Expected Error:**

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

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

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

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

\[ A[\mathbf{h}(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \, . \]

**Expected Error:**

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

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

**Markov’s inequality:**

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

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

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

- $f(x)$: 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.
Claim: For any $x$, with probability at least $1/2$,

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

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

**f(x):** frequency of $x$ in the stream (i.e., number of items equal to $x$). $h$: random hash function. $m$: size of Count-min sketch array.
**Claim:** For any $x$, with probability at least $1/2$, 

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

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 $x$).
- $h$: random hash function.
- $m$: size of Count-min sketch array.
**Claim:** For any \( x \), with probability at least \( 1/2 \),

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

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 \( x \)). **h:** random hash function. **m:** size of Count-min sketch array.
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median?

The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i(h_i(x))$. (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
count-min sketch accuracy

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

Why min instead of mean or median?
The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median?
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$
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 \( \geq 1/2 \):
  \[
  f(x) \leq A_i[h_i(x)] \leq f(x) + \frac{\epsilon n}{k}.
  \]
Estimate $f(x)$ by $	ilde{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 $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$?
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 $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$?
  $$1 - 1/2^t.$$
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 $Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$.

- To get a good estimate with probability $\geq 1 - \delta$, set $t = \log(1/\delta)$. 

**Count-Min Sketch Analysis**

![Diagram of Count-Min Sketch](image)
Upshot: Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{en}{k}$ with probability $\geq 1 - \delta$ in $O(\log(1/\delta) \cdot k/\epsilon)$ space.
**Upshot:** Count-min sketch lets us estimate the frequency of every 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 – distinguish 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 every 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 – distinguish 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?
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:

• 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 Elements?