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>
<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>
**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>
**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) $\log n$
**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) \( \frac{n}{k} \)  
  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.

<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) \( n/k \)
  - 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?
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).

<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>$x_{n-k+1}$</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
**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.

Let $x$ be an element in the stream. We use $\mathbf{A} = \{h(x)\}$ to estimate $f(x) = |\{i : x_i = x\}|$, the frequency of $x$ in the stream.

Random hash function $h$

$m$ length array $\mathbf{A}$

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

\[ A[h(x)] \] to estimate \( f(x) = |\{ i : x_i = x \}| \), the frequency of \( x \) in the stream.
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

Random hash function $h$

$m$ length array $A$
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

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

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

- **Random hash function** $h$
- **m length array** $A$: 
  - $4, 2, 1, 6, 20, 1, 3, 41, 8, 2$
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

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

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

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

Error in frequency estimate

**Markov's inequality:**

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

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

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

**Expected Error:**

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

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

**Markov's inequality:**

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

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

Expected Error:

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

\( f(x) \): 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

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

**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.
**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 $1/2$,

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

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 $1/2$,

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

How can we improve the success probability?

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

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

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 Accuracy

Estimate $f(x)$ with $\hat{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (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!
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)

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>2</td>
<td>1</td>
<td>90</td>
</tr>
<tr>
<td>$x_2$</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$x_3$</td>
<td>1</td>
<td>10</td>
<td>52</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0</td>
<td>78</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>80</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>104</td>
<td>11</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Estimate $f(x)$ with $	ilde{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 $\geq 1/2$:

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

---

**Diagram:**

- A sketch with \( x_1, x_2, x_3, x_4, \ldots, x_n \) and \( t \) random hash functions \( h_1, h_2, \ldots, h_t \).
- Three arrays \( A_1, A_2, A_3 \):
  - \( A_1 \): \( 2, 5, 1, 0, 6, 12, 104, 1, 3, 4 \)
  - \( A_2 \): \( 1, 6, 1, 10, 78, 80, 4, 11, 3, 5 \)
  - \( A_3 \): \( 90, 1, 52, 6, 3, 12, 33, 9, 3, 2 \)
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 $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$. 

\[ x_1 \quad x_2 \quad x_3 \quad x_4 \quad \ldots \quad x_n \]

\[ A_1 \]
\[ \begin{array}{cccccccc}
2 & 5 & 1 & 0 & 6 & 12 & 104 & 1 & 3 & 4
\end{array} \]

\[ A_2 \]
\[ \begin{array}{cccccccc}
1 & 6 & 1 & 10 & 78 & 80 & 4 & 11 & 3 & 5
\end{array} \]

\[ \vdots \]

\[ A_t \]
\[ \begin{array}{cccccccc}
90 & 1 & 52 & 6 & 3 & 12 & 33 & 9 & 3 & 2
\end{array} \]
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 $Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$? $1 - 1/2^t$.

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

![Diagram of Count-Min Sketch with arrays and hash functions]
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 $O\left(\log(1/\delta) \cdot \frac{k}{\epsilon}\right)$ space.
**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.

- Accurate enough to solve the $(\epsilon, k)$-Frequent elements problem: Can 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: Can 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?
**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.

- Accurate enough to solve the $(\epsilon, k)$-Frequent elements problem: Can 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? $\delta = 0.01/|U|$ ensures

\[
\Pr[\text{there exists } x \in U \text{ with a bad estimate}] \\
\leq \sum_{x \in U} \Pr[\text{estimate for } x \text{ is bad}] \\
\leq \sum_{x \in U} 0.01/|U| = 0.01
\]
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 look up the estimated frequency for $x \in U$?

One approach:

- Maintain a set $F$ while processing the stream:
  - At step $i$:
    - Add $i$th stream element to $F$ if it's estimated frequency is $i/k$ and it isn't already in $F$.
    - Remove any element from $F$ whose estimated frequency is $< i/k$.
- Store at most $k$ items at once and have all items with frequency $\geq n/k$ stored at the end of 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 look up the estimated frequency for $x \in U$?

**One approach:**

- Maintain a set $F$ while processing the stream:
- At step $i$:
  - Add $i$th stream element to $F$ if it’s estimated frequency is $\geq i/k$ and it isn’t already in $F$.
  - Remove any element from $F$ whose estimated frequency is $< i/k$.
- Store at most $k$ items at once and have all items with frequency $\geq n/k$ stored at the end of the stream.
Questions on Frequent Elements?