Heavy Hitters
Tail Bounds

Anna Karlin
Stream Model

- Input elements (e.g. Google queries) enter/arrive one at a time.
- We cannot possibly store the stream.

Question: How do we make critical calculations about the data stream using a limited amount of memory?
Sources of this kind of data

- Sensor data
  - E.g. millions of temperature sensors deployed in the ocean

- Image data from satellites or surveillance cameras
  - E.g. London

- Internet and web traffic
  - E.g. millions of streams of IP packets

- Web data
  - E.g. Search queries on Google, clicks on Bing, etc.
Example Applications

- Mining query streams
  - Google wants to know which queries are more frequent today than yesterday.
- Mining click streams
  - Facebook wants to know which of its ads are getting an unusual number of hits in the last hour.
- Mining social network news feeds
  - E.g., looking for trending topics on Twitter and Facebook, trending videos on TikTok
More Applications

- Sensor networks
  - Many sensors feeding into a central controller.

- IP packets
  - Gather congestion information for optimal routing
  - Detect denial-of-service attacks
**Problem**

- **Input:** sequence of \( n \) elements \( x_1, x_2, \ldots, x_n \) from a known universe \( U \) (e.g., 8-byte integers).

- **Goal:** perform a computation on the input, in a single left to right pass where
  
  - Elements processed in real time

  - Can’t store the full data. \( \Rightarrow \) minimal storage requirement to maintain working “summary”
Heavy Hitters: Keys that occur many times

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

Applications:
- Determining popular products
- Computing frequent search queries
- Identifying heavy TCP

\[ f_{32}^6 = 2 \]

Find all elements that occur more than \( \frac{n}{100} \)

Goal: Find/output all elements with \( f_x^* \geq \frac{n}{k} \) (These els are “heavy hitters”) \( (k = 100) \)
Output has size $O(k)$

Provably impossible to solve this problem exactly with sublinear space.

How many els can there be with $f_x^+ \geq \frac{n}{100}$?

\[ \begin{align*}
\text{a)} & \quad \infty \\
\text{b)} & \quad n \\
\text{c)} & \quad 100 \\
\text{d)} & \quad 10
\end{align*} \]

Modified goal: (solve $E$-HH problem)

1. If $f_x^n \geq \frac{n}{k}$, added to list, we output.

2. If some elt, say $y$, added to list, then w.p. $\geq 1 - \epsilon$

$$f_y^n \geq \frac{n}{k} - \epsilon n$$

Suppose $k = 20$, $\epsilon = 0.01$, $\delta = \frac{1}{2^{10}}$

\[ \begin{align*}
\text{a)} & \quad f_x^n \geq 0.05n \\
\text{b)} & \quad \text{for every } y \text{ in list w.p. } \geq 1 - \frac{1}{2^{10}}, f_y^n \geq 0.05n
\end{align*} \]
\[ k, \varepsilon, \delta \Rightarrow \text{determine } b, \varepsilon. \]
Count-min sketch

- Maintain a short summary of the information that still enables answering queries.

- Cousin of the Bloom filter
  - Bloom Filter solves the “membership problem”.
  - We want to extend it to solve a counting problem.
Count-Min Sketch

Want:
1. If \( f_x \geq \frac{n}{k} \), added to list.
2. If some elt, say \( y \), added to list, then \( \Pr \geq 1 - \delta \)
   \[ f_y \geq \frac{n}{k} - \epsilon \]

Designs specify \( k, \epsilon, \delta \rightarrow b, \xi \)

Keep 2D array \( \xi \) hash tables, each of size \( b \).
when elt \( x \) shows up.

**Update** \((x)\): 
\[
\text{for } 1 \leq j \leq \ell \text{ increment } t_j[h_j(x)]
\]

**Count** \((x)\): return 
\[
\min_{1 \leq j \leq \ell} t_j[h_j(x)]
\]

if **Count** \((x)\) \(\geq \frac{1}{\ell} \), add \(x\) to HT list

after \(x_1, \ldots, x_t\) arrived

\[
t_j[h_j(x)] \text{ guaranteed}
\]

- \(a) \quad \leq f_x^t
- \(b) \quad = f_x^t
- \(c) \quad \geq f_x^t
- \(d) \quad I \text{ don't know}

after \(x_1, \ldots, x_t\) arrived

\[
\text{Count}(x) \text{ guaranteed}
\]

- \(a) \quad \leq f_x^t
- \(b) \quad = f_x^t
- \(c) \quad \geq f_x^t
- \(d) \quad I \text{ don't know}
Assumptions

1. Hash functions behave like random maps
   \[ h_1, h_2 : U \rightarrow \{0, 1, \ldots, b-1\} \]
   \[ \forall x \neq y \quad \Pr(h_j(x) = h_j(y)) = \frac{1}{b} \]

2. Hash functions are independent of each other.

If hash fn \( h \) random, \( x \neq y \)
\[ \Pr(h(x) = h(y)) = \]
\[ a) \frac{1}{b} \]
\[ b) \frac{1}{b^2} \]
\[ c) 0 \]
\[ d) I don't know \]
Fix time \( t \). (\( x_{t-1}, x_t \) have just arrived)

\[
Z_j \equiv \sum_{j} h_j(x)
\]

\[
Z_j^t = f_x^t + \sum_{\text{distinct } y \neq x} f_y^t W_{xy}
\]

\[
E(\mathbb{E}(Z_j)) = f_x^t + \sum_{\text{distinct } y \neq x} f_y^t W_{xy}
\]

\[
E(W_{xy}) = \frac{1}{\theta} \sum_{x \neq y} \frac{h_j(x)}{h_j(y)}
\]

\[
\frac{t - f_x^t}{f_x^t} \leq t \leq n
\]
$E(z_i) \leq f_x^t - \frac{\sqrt{\text{Var}(x)}}{\beta}$

$E(z_i - f_x^t) \leq \frac{n}{\beta}$

$\Pr(X \geq \frac{2n}{\beta}) \leq \Pr(X \geq 2E(X))$

$X > 0$

Markov's Inequality:

If $X \geq 0$ r.v.

$\Pr(X \geq cE(X)) \leq \frac{1}{c}$
Count-Min Sketch

- Elegant small space data structure.
- Space used is independent of n.
- Is implemented in several real systems.
  - AT&T used in network switches to analyze network traffic.
  - Google uses a version on top of Map Reduce parallel processing infrastructure and in log analysis.
- Huge literature on sketching and streaming algorithms (algorithms like Distinct Elements, Heavy Hitters and many many other very cool algorithms).