hash functions

Count-Min Sketches

Sriram Sankaranarayanan

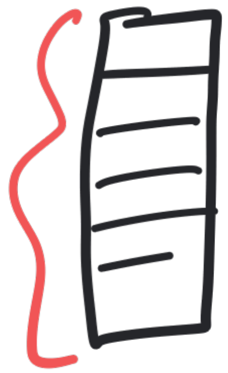
Data Structures and Algorithms

$x_1, x_2, \boxed{x_3} \ldots \boxed{x_N}$

"string" . . .

all that words in a Tolstoy's W&P.

Counter



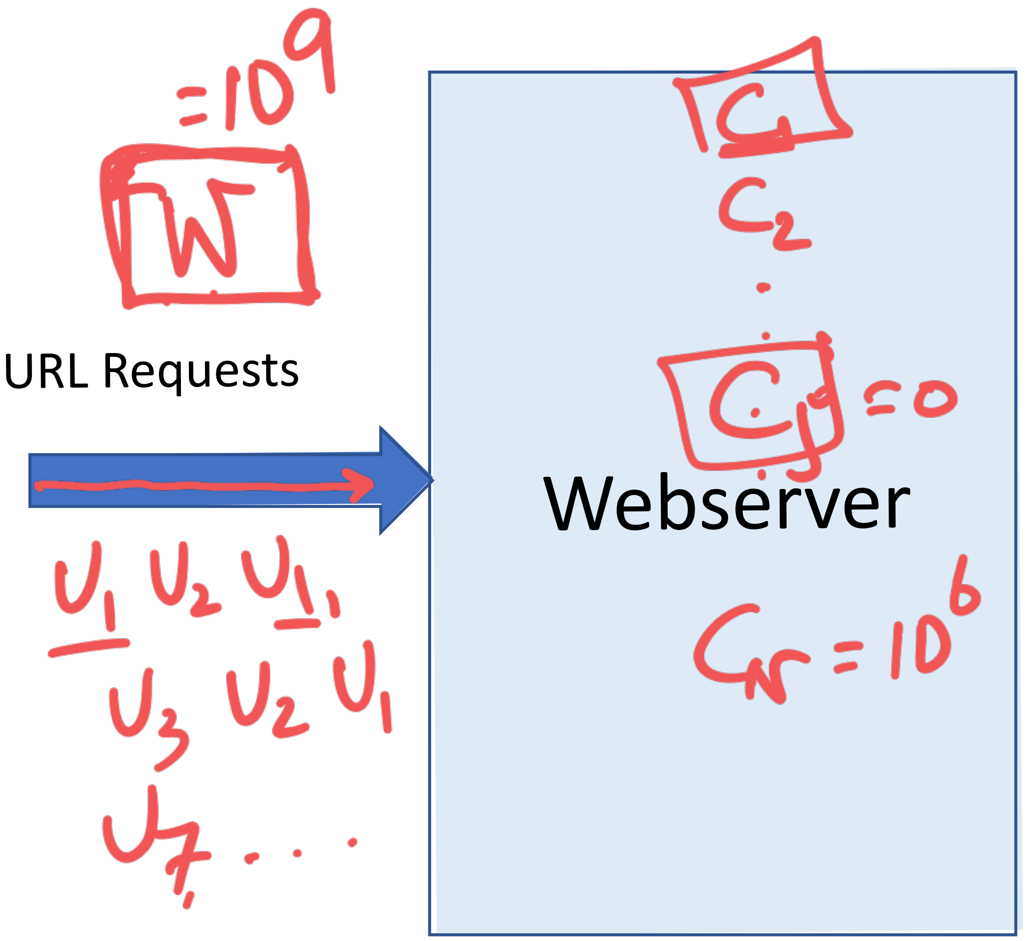
x_i

"Piece"

approx count

"Close" to the real count.

Problem: Count Items in a Data Stream



Problem: Keep count of how often each URL is requested.

Unique URLs: U_1 U_2 U_3 U_4 U_5 U_6 .. U_N $N \approx 10^6$

Have N distinct counters: C(1),…, C(N)

Each time URL $\boxed{U_j}$ requested: Increment counter C(j)

Problem: N is humungous. not known in advance

Most URLs are requested very few times.

A few URLs are requested a lot of times.

Advantage: *Approximate count within 10% of actual answer (say) is acceptable.*

Approximate Counting Data-Structure

W = size of stream

Stream of data: $x_1 x_2 x_3 x_4, \cdots, x_W$

Each element of the stream : $\boxed{x_j} \in \{1, j, \ldots, N\}$

Return $approxCount(j)$: must be within $\boxed{\epsilon W}$ of *true count*

with probability $\boxed{\text{at least } \delta}$

ϵ

$10^{-6} \times 10^{9}$

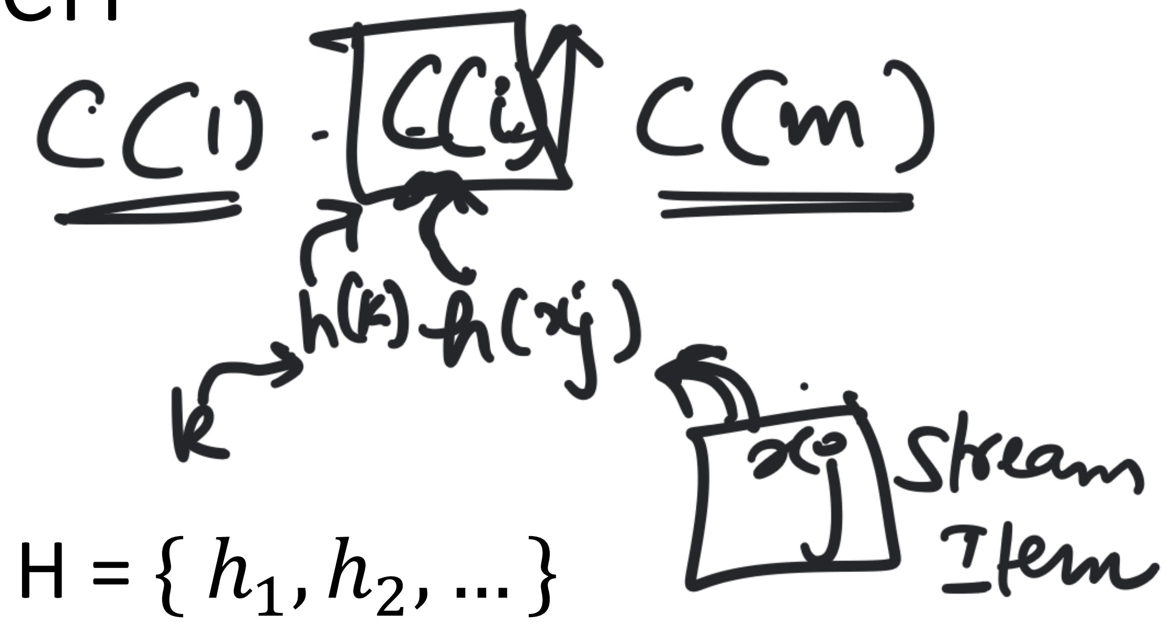
≈ 1000

1B

- Typical Numbers: $\cancel{N \sim 10^8}$, $W \sim 10^9$, $\epsilon \sim 10^{-6}$, $\boxed{\delta = 0.99}$
- From a stream of nearly 1 billion items each having a number between 1 and 100Million, count how often each item occurs where the count is within 1000 of the true count at least 99% of the time.

how to choose m.

Basic Idea of Count-Min Sketch



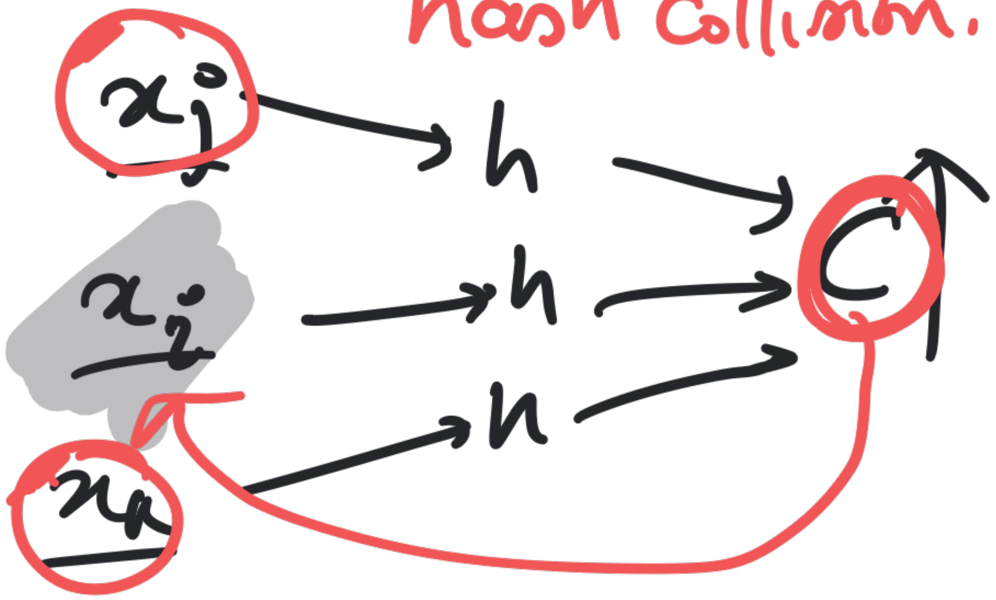
Use m counters: $\underline{C(1)} \dots \underline{C(m)}$

We will choose m later (expect $m \ll \overset{W}{\cancel{M}}$)

Draw a hash function at random from family H = { $\underline{h_1, h_2, \dots}$ }

$$h_i: \{1, .., \underset{N}{\cancel{M}}\} \to \{\underline{1, \dots m}\}$$

hash collision.



Stream Item $\boxed{x_j}$: $\text{Increment}(C\left(h(x_j)\right))$

$$approxCount(\boxed{k}) = \boxed{C(h(k))}$$

Count-Min Sketch Error Analysis

- $approxCount(j) \geq count(j)$

$$\Pr_{\substack{h \in H \\ i \neq j}} \left(h(i) = h(j) \right) \leq \boxed{\frac{c}{m}}$$

$$E(approxCount(j)) = Count(j) + \sum_{\substack{i=1 \\ i \neq j}}^{N} \boxed{\Pr(h(i) = h(j))} \times Count(i).$$

$$\leq Count(j) + \frac{c}{m} \cdot \boxed{\sum_{i=1}^{N} Count(i)}$$

$$\leq Count(j) + \frac{c}{m} W.$$

$$E(approxCount(j) - Count(j)) \leq \frac{c}{m} W$$

Count-Min Sketch: Choosing m

$$\mathbb{E}(\text{error}(j)) \le \boxed{\frac{cW}{m}}.$$

$$\Pr(\boxed{\text{error}(j)} \ge \underbrace{\varepsilon W}_{t}) \le \frac{c\cancel{W}}{m \cdot \varepsilon \cancel{W}} \quad \text{(M. Ineq.)}$$

approxCount - Count
≥ 0

$$\le \frac{c}{m\varepsilon}.$$

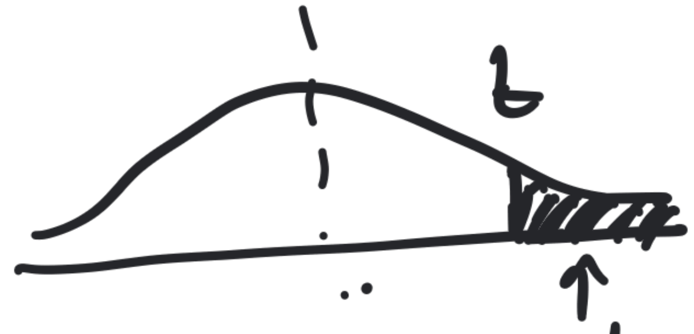
$$\le \boxed{\frac{1}{e}}.$$

Not good enough

2.72...

$$\therefore \boxed{m = \frac{ec}{\varepsilon}}$$

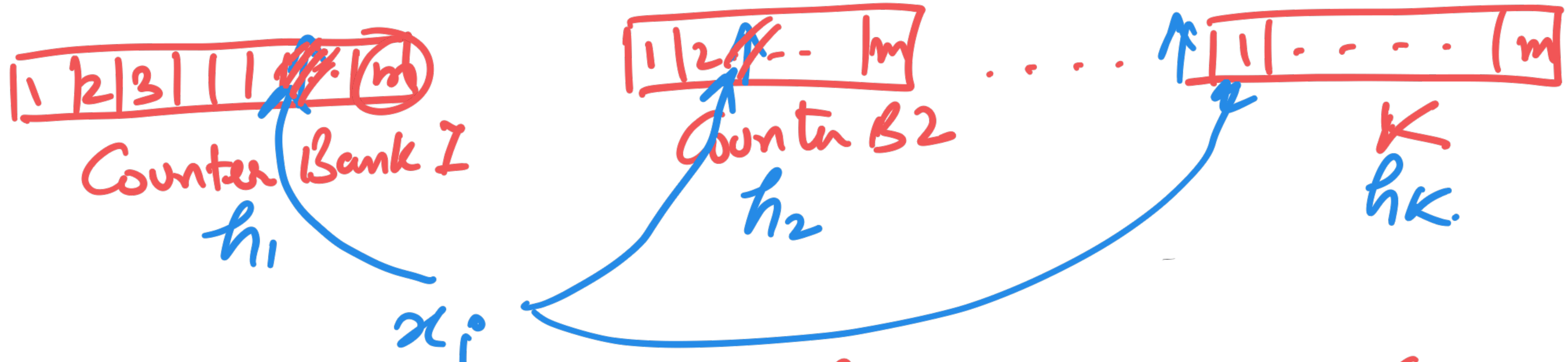
E(x)



Markov. Ineq.

$$\Pr(x \ge t) \le \frac{E(x)}{t}$$

Count-Min Sketch: Reducing Error Probability



$$\text{approx Count}(k) = \underline{\min}\left(C_1(h_1(k)),\ C_2(h_2(k)),\ \ldots,\ C_K(h_K(k))\right)$$

$$\underline{Pr(\text{error}_j(h_j(k)) \geq \boxed{\varepsilon W})} \leq 1/e$$

All of them make

$$(1/e)^{K} \leq 1-\delta \quad \boxed{K \geq \frac{-\ln(1-\delta)}{-\ln(0.01)}}$$

Count-Min Sketch: Overall Algorithm

- Initialize K counter-banks with hash functions : $h_1, h_2, \ldots, h_K$ randomly chosen

- Stream $\boxed{\text{item } x_j}$
  - Increment $C_1(h_1(x_j))$, $C_2(h_2(x_j))$, … , $C_K(h_K(x_j))$

$\Theta(K)$

- Query count of k
  - $approxCount(k) = \min(C_1(h_1(k)), C_2(h_2(k)), \ldots C_K(h_K(k)))$

$\Theta(K)$

$K \approx S$

Count-Min Sketch: Some actual numbers

1 Billion

- Stream of 1 Billion Items 10^9.
- $\epsilon = \boxed{10^{-6}}$ (tolerate error of upto 1000)
- $\delta = \boxed{0.9}$

for 90% of my queries I wish the answer to be within 1000 of actual answer

- $m = \frac{e}{\epsilon} \approx \boxed{3 \times 10^6}, K = -\ln(1-\delta) \approx \boxed{3}$

$\boxed{10M} \sim 10^7$.

- Use 3 banks of 3 million counters, each.
  - Guarantees that `approxCount` will be within 1000 of true at least answer 90% of the time.

- Merkle : Blockchain -
- Locality Sensitive . .

. . .