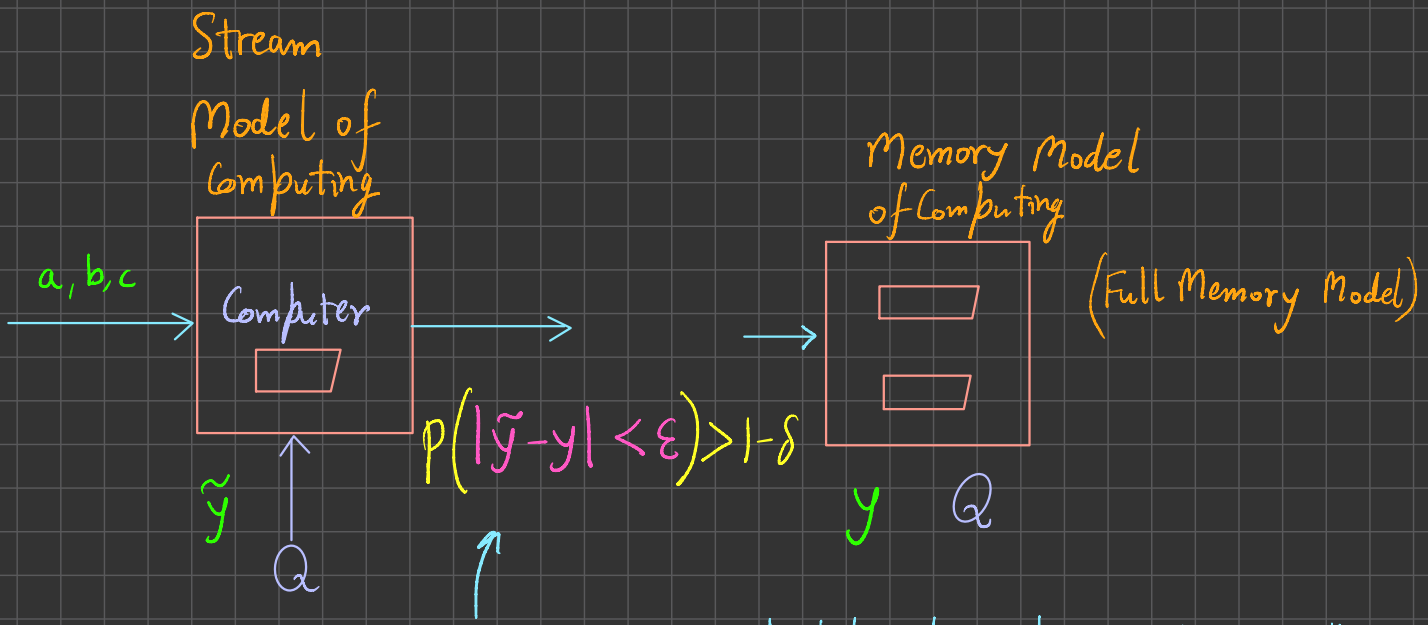
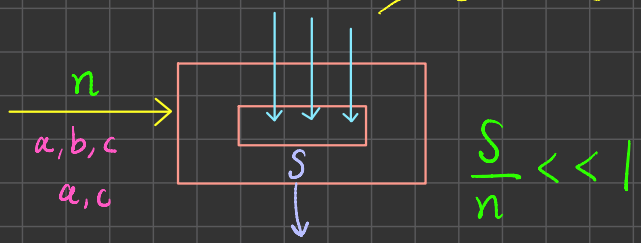
Streaming data algorithms:



Full memory is computationally not equivalent to the other model,

So look for lowering the prediction error.

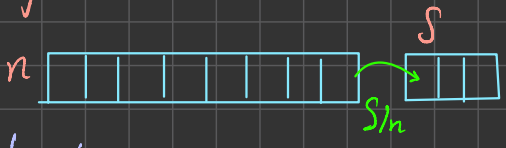
Sampling



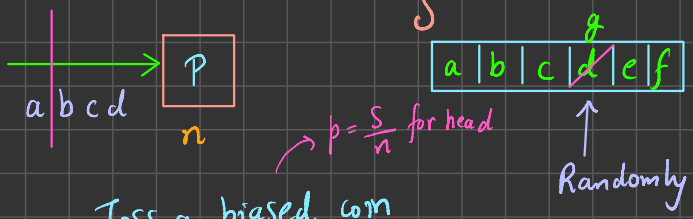
$\frac{S}{n} << 1$

So sample is a representation of all n elements

Faithful – Unbiased

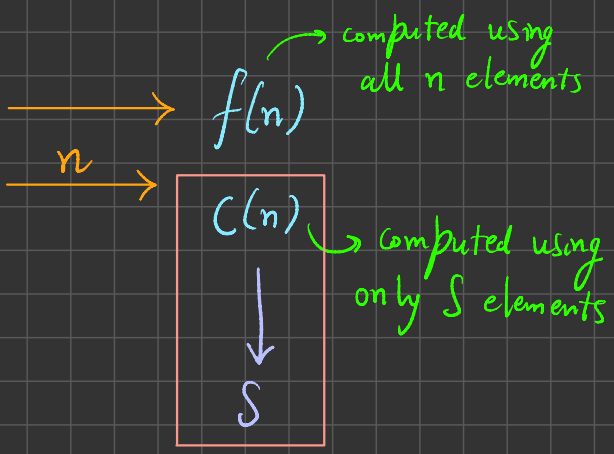


Reservoir Sampling



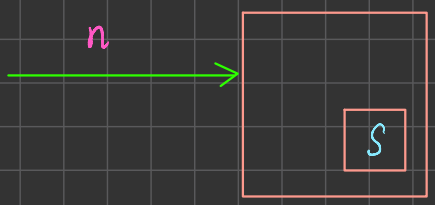
Toss a biased coin and replace the element with $(S+1)^{st}$ if heads on coin toss.
if head, then put the new element in S, else discard new element

General idea: Sketches



$f \sim C$

Membership:



Q: has K ever appeared in stream of n elements?

Solving this exactly requires to store all n elements!

K

Bloom Filter: $h: K \rightarrow i \in S$

0 False negative
>0 False positives

$h_1(K) \wedge h_2(K) \wedge \ldots \wedge h_m(K)$

Both increasing m or S, lowers false positives.

Turns the bit on always

K'?

If K' ever appeared, then detected, but it might happen that some stray K' $\pi h_i(K')$ is also 1, which results in false positives

Count distinct

$a \in U, \ |U| = N$

$S = a_0 \ a_1 \ \dots \ a_n$ (stream)

m – memory available $(m << n, m << N)$

Given that m is upto $\log N$, we want to approximate distinct elements' count in S.

Flajolet-Martin algorithm:

Uses a sketch – bit string – $\log_2 N$

↓

produces from 1–N (say q)

Now convert q to bit string and let $r(a)$ be the number of trailing zeros in this bit string of q.

$a_0 \longrightarrow r(a_0)$

$a_1 \longrightarrow r(a_1)$

$a_m \longrightarrow r(a_m)$

$R \ (\max r(a_i))$

↳ # trailing zeros in bit vector of $h(a_i)$

Then,

approx. count distinct $= 2^R$

↓ more precisely

$$C - \varepsilon \leq E(2^R) \leq C + \varepsilon$$

Actual count

$h_1 \quad \cdots \quad h_K$ → K is of order $\log N$ (else contradicts memory constraint)

$2^{R_1} \quad \cdots \quad 2^{R_K}$

$\downarrow \quad \downarrow \quad \cdots \quad \downarrow$

$m_1 \quad \cdots \quad m_K$

break K into grps. of size $\lfloor \log N \rfloor$ and take avg. of each grp.

$avg_1 \quad avg_2 \quad \cdots \quad avg_p$

Take median of these as the best estimate

Moment Query:

$S = a_0 \cdots a_n \quad a_i \in |U|_N$ → U is a set with $|U| = N$

$\downarrow$

$V_1, V_2, \ldots, V_N$ → distinct elements in N.

$$m_d(S) := \frac{1}{n} \sum_{i=1}^{N} \left(1 + c(V_i) + c(V_i)^2 + \cdots + c(V_i)^d \right)$$

$c(V_i) = \#V_i$ appears in S.

1^{st} moment → $\frac{\sum c(V_i)}{n}$

Only $\log N$ memory available, so we need approx. for moment computation.

2^{nd} moment → $\frac{\sum c(V_i)^2}{n}$

[Surprise number] a number of interest

X – random variable

$x = a_t, x_2, \dots, x_K$ → K different values caught at different points of times.

$S = a_0 \cdots a_t \cdots a_n$

m' → frequency of x over the rest of the stream. n

$$c_{approx}(x) = \frac{n(m')}{n-t}$$

use these for each of x_i for moment calculations.

General:

$a \in U, |U| = N$

$S = \{a_0, a_1, \dots, a_n\} \longrightarrow q(a)$

$|M| = m$ | sketch $\quad \hat{q}(a)$

$m = O(\log N, \log n)$ and $\Pr(|\hat{q}(a) - q(a)| \le \varepsilon) \ge 1 - \delta$

Frequency query (count)

– additive stream → values once appeared, don't disappear

– add/delete stream

Histogram vector : $[c_1 \quad \cdots \quad c_N]$

$q(a) = (1, c_a)$ if a appears at least εn times in S (# of a; this currently working defn. of frequent not top K.)

0 else

(Heavy hitters)

↳ So we only want know the "frequent" terms/heavy hitter in stream

Even this is impossible to be answered exactly with memory constraint, so an approx $\hat{q}(a)$ is found.

$$\Pr\left(\left|\hat{c}(a) - c(a)\right| \leq \varepsilon\right) \geq 1 - \delta$$

Problem: Given a stream S, $|S| = m$, $a \in U$

if $c(a) \geq m/2$, then $q(a) = c(a)$, $q(a) = 0$

Majority algorithm → works only on additive stream.

Break down of algo:

Initialization data str.

Maintainance / Process updates

Query $q(a)$

Frequency query (count):

Misra-Gries } sketches
Freq. Count } are linear

i.e. union of streams lead to addition of sketches.

Initialization: Counter = 0, value = a_0 (in-hand)

Processing: (i) a_i : a_i == value, counter ++

(ii) $a_i \neq$ value, counter --

(iii) if counter == 0, value = a_i

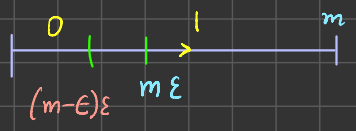
Sketch stores a_i, counter → max size is m

⇒ requires $\log m$ size.

Problem: if $c(a) \geq m\varepsilon$ then $q(a)=1$

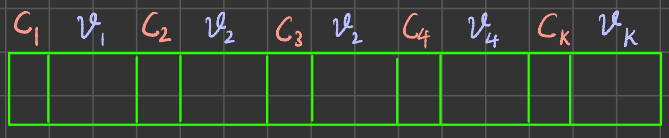
$c(a) \leq (m-\epsilon)\varepsilon$ then $q(a)=0$

(another epsilon)



if $q(a)$ is 1, find $\hat{c}(a)$, such that $Pr\left[|\hat{c}(a)-c(a)| \leq \varepsilon'\right] \geq 1-\delta$

Misra-Gries algorithm:



Initialization: $Counter_1 = 0$, $value_1$; $Counter_2 = 0$, $value_2$; ..... $Counter_K = 0$, $value_K$;

Processing: for a_i,

if any $Counter_p == 0$, update $value_p = a_i$, and set the corresponding counter to 1.
(aka already exists)

if $a_i = V_p$, then increment

else decrement all the counters → In space saving algo., we decrement only the one with smallest counter.
↳ leads to overestimate of count

Query: if for a_i, the counter $\neq 0$, then $q(a_i)=1$

and corresponding counter value is an estimate of frequency

Count Min Sketch :

w, d are independent of stream size.

→ for each stream value, w updates happen

Rest from slides

C.T. :

Basic Streaming models

Bloom filter

Count distinct

Flajolet Martin

Misra - Gries

Freq. count

Count min

Ref: Ullman / mmds

Graham Cormode lecs. } for full proofs

↳ Sketches, Streaming and Big data