CS630 Graduate Algorithms

November 5, 2024

by Dora Erdos and Jeffrey Considine

- Counting Sketches

Great Duck Island

Image source: https://www.coa.edu/live/files/739-great-duck-island-eno-stationpdf

Leach’s Storm Pestrel (not a duck)

“It breeds on inaccessible islands in the colder northern areas of the Atlantic and Pacific Oceans. It nests in colonies close to the sea in well concealed areas such as rock crevices, shallow burrows, or even logs. It lays a single white egg, which often has a faint ring of purple spots at the large end. This storm petrel is strictly nocturnal at the breeding sites to avoid predation by gulls and skuas, and even avoids coming to land on clear, moonlit nights.”

Lots of research into its nesting habits, but the birds don’t make it easy to monitor them.

Solution: sensor networks?

Image/quote source: https://en.wikipedia.org/wiki/Leach's_storm_petrel

Sensor Networks and Low Powered Aggregation

Lots of small devices taking measurements in places inconvenient to monitor.

- Very low power, low memory.
- May not be plugged into power.
- May not have persistent internet connectivity.
- If any connectivity, often low bandwidth and high loss rates.

Sensor Networks and Low Powered Aggregation

Lots of small devices taking measurements in places inconvenient to monitor.

- Very low power, low memory.
- May not be plugged into power.
- May not have persistent internet connectivity.
- If any connectivity, often low bandwidth and high loss rates.

But what about the ducks?

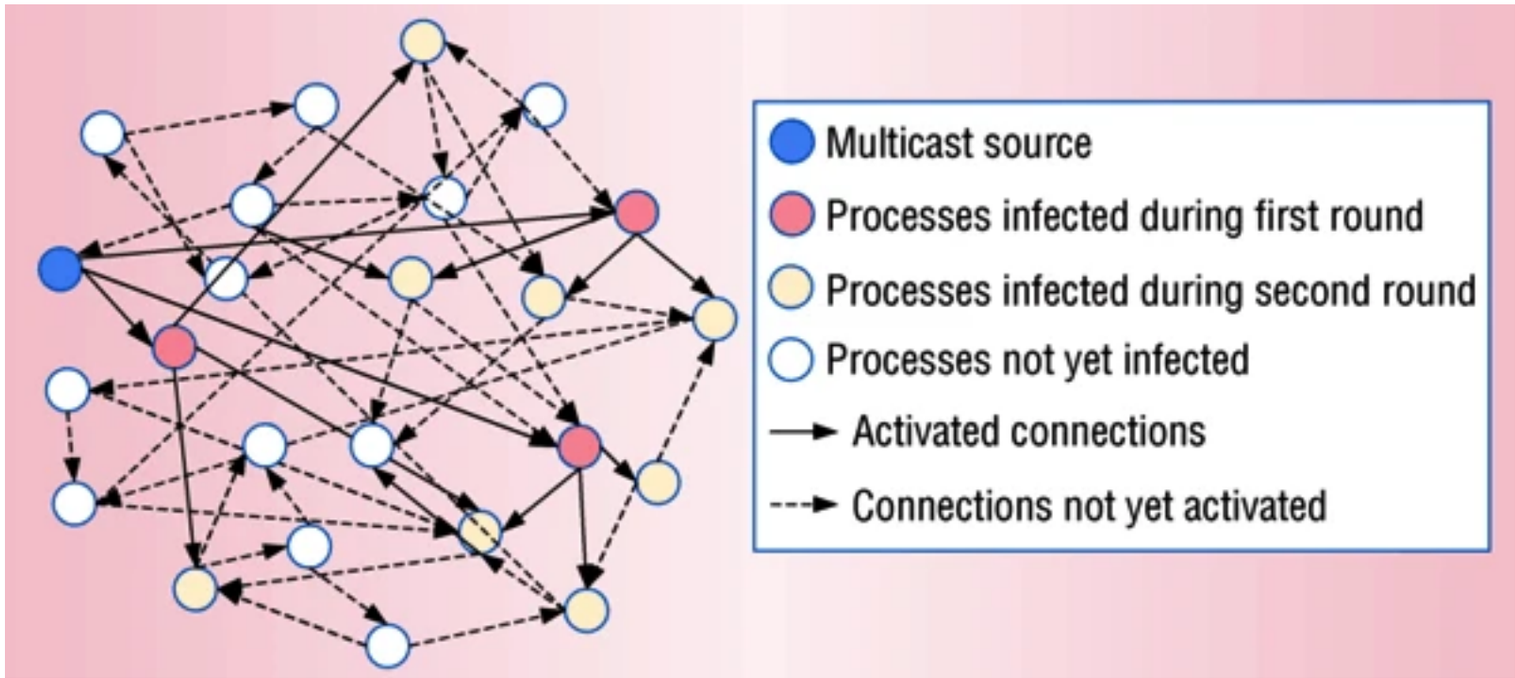
- Scientists walked around Great Duck Island and put these sensors in all the holes they could find where they thought a ~~duck~~ pestrel might hide.
- Each sensor node had a temperature sensor to guess whether a pestrel was in the hole, and a small transmitter.

Can the sensor nodes work together to get a bird count back to base?

Epidemic/Gossip Algorithms

Basic idea:

- Send all the information you have to everyone you can contact.
- Can be an effective way to distribute certain kinds of information.
  - Good for “X happened”.
  - Less good for “A is alive, B is alive, C is alive, D is alive, …”



Epidemic Information Dissemination in Distributed Systems (2004)

TopHat

Space Requirements for Exact Counting with Duplicates

Theorem:

- Counting up to n items exactly requires at least n bits.
- Exactly n bits if there are n possible items.
- Lower bound is worse (higher) if the universe is big $\sim n \log_2 U$ for big U

Space Requirements for Exact Counting with Duplicates

Theorem:

- Counting up to n items exactly requires at least n bits.
- Exactly n bits if there are n possible items.
- Lower bound is worse (higher) if the universe is big $\sim n \log_2 U$ for big U

Any fewer bits than this bound, and you can be tricked by duplicates.

- 2^n possibilities needed to figure out which items already included.
  - If two are more different sets have the same representation, you can force bad updates for some of them.
  - Suppose all the sets sharing a representation have cardinality k.
  - If queried now, the answer should be k.
  - If we pick an item in some but not all the sets to “add”, does the answer increase?
  - Sometimes it should, sometimes it should not.

Approximate Set Representations

Can we devise a “sketch” of a set that can answer approximate queries?

- Want sketch to be small.
- Want fast inserts.
- Want fast sketch unions.

These let us run a cheap epidemic protocol.

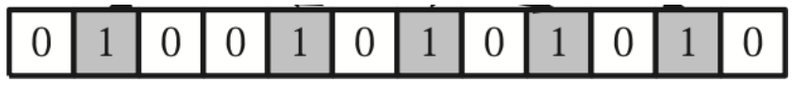
- Want fast size queries.

Performed back at duck base,
so maybe can loosen this up.

- Have we seen anything like this?

How Many Items Were Inserted in this Bloom Filter?

A Bloom filter with 3 hash functions and some inserts…



5 / 12 bits set, so at least 2 inserts.

Could be an unbounded number of super lucky inserts?

Ill-posed question: How many items were inserted in this Bloom filter?

Better question: What estimation method can I use so that if I insert n items and run the estimator, I get an unbiased estimate of n?

Intuitive idea for Bloom filters

- If pm out of m bits are still zero and recalling $p \approx e^{-kn/m}$ after n insertions, estimate $n \approx -\ln p(m/k)$

Beware the non-linearity when reasoning about estimators.

A Linear-Time Probabilistic Counting Algorithm for Database Applications (1990)

Bloom Filter Suitability as Counting Sketches

Bloom filters typically use space linear in the number of items inserted.

- So similar to the 1 bit per item minimum for exact counting with limited set of items.
- Bloom filters can be made smaller, but all bits get set to one around $m = n/\log n$ bits.
  - Neither useful for set membership nor counting.

- We want something a lot smaller.
- How small?

Flajolet and Martin Sketches (FM sketches)

Initialize:

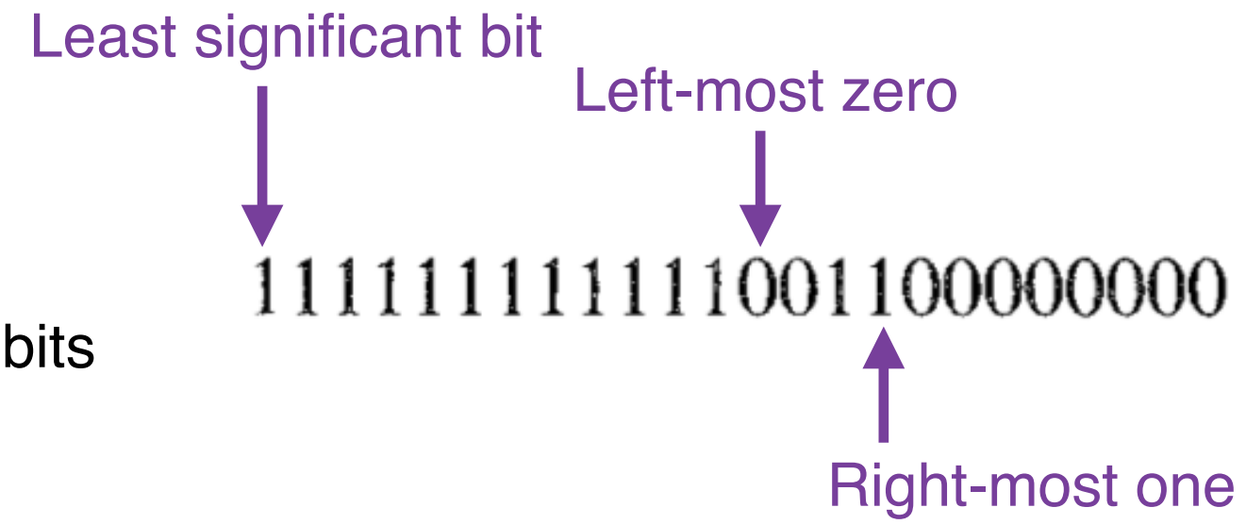
- Set array B of L bits to zero.

Insert(x):

- Compute $h(x)$.
- Find least significant bit i set to one in $h(x)$.
- Set $B[i]$ to one.

About insertions…

- 1/2 of all insertions set $B[0]$.
- 1/4 of all insertions set $B[1]$.
- After n insertions, most of first $\log_2 n$ bits are likely set to one.



Probabilistic counting algorithms for database applications (1985)

Flajolet and Martin Sketches (FM sketches)

Let R to the index of the left-most zero.

- $E[R] = \log_2 \varphi n$
  where $\varphi = 0.77351\ldots$
- $\sigma(R) \approx 1.12$

This is an estimator for the logarithm of n,
not n itself.

- About 1.5 bit standard deviation.
- Unbiased estimates of logarithm
  can give biased estimates of number.

Probabilistic counting algorithms for database applications (1985)

Abridged Analysis of FM Sketches

If we insert n items, how likely is $B[0]$ to be set?

Abridged Analysis of FM Sketches

If we insert n items, how likely is $B[0]$ to be set?

$$P[B[0] = 0] = \frac{1}{2^n}$$

Abridged Analysis of FM Sketches

If we insert n items, how likely is $B[\lfloor \log_2 n \rfloor]$ to be set?

Abridged Analysis of FM Sketches

If we insert n items, how likely is $B[\lfloor \log_2 n \rfloor]$ to be set?

Ignoring the rounding,

- $P[B[\lfloor \log_2 n \rfloor] = 0] \approx \left(1 - \frac{1}{n}\right)^n \quad 1/e$

- Expected number of times that bit is set to one is ~1.

Abridged Analysis of FM Sketches

If we insert n items, how likely is $B[\lceil 2\log_2 n \rceil]$ to be set?

$$P(B[\lceil 2\log_2 n \rceil] = 1) \leq \frac{1}{n^2} \cdot n = \frac{1}{n}$$

Abridged Analysis of FM Sketches

If we insert n items, how likely is $B[\lfloor 0.5 \log_2 n \rfloor]$ to be set?

$$P(B[\lfloor 0.5 \log_2 n \rfloor] = 0 \approx \frac{1}{e^{\sqrt{n}}}$$

So almost certainly 1.

Abridged Analysis of FM Sketches

The first $\log_2 n - \log_2 \log n$ bits set w.h.p.

No bits after first $2 \log_2 n$ bits set w.h.p.

So small range where first (least-significant) zero bit will be found…

- Will skip those details now.

Abridged Analysis of FM Sketches

THEOREM 3.A. *The average value of parameter* R_n *satisfies*:

$$\bar{R}_n = \log_2(\varphi n) + P(\log_2 n) + o(1),$$

where constant $\varphi = 0.77351\ldots$ *is given by*

$$\varphi = 2^{-1/2} e^{\gamma} \frac{2}{3} \prod_{p=1}^{\infty} \left[\frac{(4p+1)(4p+2)}{(4p)(4p+3)} \right]^{(-1)^{\nu(p)}}$$

and $P(u)$ *is a periodic and continuous functions of* u *with period* 1 *and amplitude bounded by* 10^{-5}.

THEOREM 3.B. *The standard deviation of* R_n *satisfies*

$$\sigma_n^2 = \sigma_\infty^2 + Q(\log_2 n) + o(1),$$

where $\sigma_\infty = 1.12127\ldots$ *and* $Q(u)$ *is a periodic function with mean value* 0 *and period* 1.

We can mention in passing for σ_∞ the "closed form" expression

$$\sigma_\infty^2 = \frac{1}{12(\log 2)^2} [2\pi + \log 2 - 12N'(0) - 12N''(0)] - 2 \sum_{k>0} |p_k|^2,$$

where the p_k are the Fourier coefficients of $P(u)$ defined above.

Accuracy-Space Tradeoffs with FM Sketches

The variation estimating the log of n is about one bit…

- Can we do better?

Easy answer - repeat the sketch m times.

$$\mathbf{E}(A) \approx \log_2 \varphi n; \qquad \sigma(A) \approx \sigma_\infty / \sqrt{m}.$$

But all computation increases by m too.

Stochastic Averaging

An alternative to making m copies of the sketch.

- Instead, make m copies of the sketch.
- And use hashing to insert into just one of them.

"Probabilistic Counting with Stochastic Averaging" or PCSA

- Uneven bucket assignments lead to slight bias and increased variance, but generally preferred over basic version.
- Applicable to other sketch methods too.

THEOREM 4. *The estimate Ξ_n of algorithm PCSA has average value that satisfies*[2]:

$$\mathbf{E}[\Xi_n] = \frac{n}{\varphi}\left[\frac{1}{\log 2} N\left(-\frac{1}{m}\right) \Gamma\left(-\frac{1}{m}\right)(1-2^{-1/m})\right]^m + nP_m(\log_2 n) + o(n);$$

the second moment of Ξ_n satisfies

$$\mathbf{E}[\Xi_n^2] = \frac{n^2}{\varphi^2}\left[\frac{1}{\log 2} N\left(-\frac{2}{m}\right) \Gamma\left(-\frac{2}{m}\right)(1-2^{-2/m})\right]^m + n^2 Q_m(\log_2 n) + o(n^2).$$

In the above expressions P_m and Q_m represent periodic functions with period 1, mean value 0 and amplitude bounded by 10^{-5}.

Combining FM Sketches

FM sketches are like Bloom filters in that each insertion just sets a bit to one.

- Same handling of union operations.
- $FM(A \cup B) = FM(A) \cup FM(B)$

Small size and easy combination makes them perfect for epidemic protocols.

- $O(m \log n)$ bits.
- Actual sketch size depends to be chosen beforehand.

Most importantly, these are good enough for counting ducks!

TopHat

Next idea

What is the distribution of the maximum bit set in an FM sketch?

Next idea

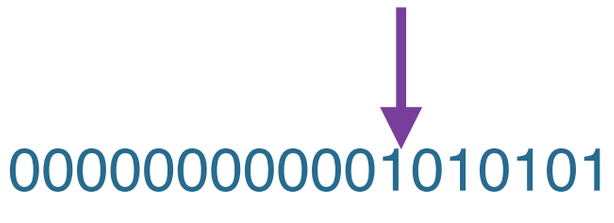
What is the distribution of the maximum bit set in an FM sketch?

What if we just save that index?

- Only takes $\log_2 \log_2 n$ space?

LogLog Sketches

The left-most one bit in a hash.
Track the maximum index over
all item hashes.



Loglog Counting of Large Cardinalities (2003)

LogLog Sketches

Algorithm LogLog($\mathfrak{M}$: Multiset of hashed values; $m \equiv 2^k$)
Initialize $M^{(1)}, \ldots, M^{(m)}$ to 0;
let $\rho(y)$ be the rank of first 1-bit from the left in y;
for $x = b_1 b_2 \cdots \in \mathfrak{M}$ do
set $j := \langle b_1 \cdots b_k \rangle_2$ (value of first k bits in base 2)
set $M^{(j)} := \max(M^{(j)}, \rho(b_{k+1} b_{k+2} \cdots)$;
return $E := \alpha_m m 2^{\frac{1}{m} \sum_j M^{(j)}}$ as cardinality estimate.

Main idea:

- Instead of finding first hash bit set to one and setting it in the sketch too, just store its index.
  - Logarithmic sketch size → log log index size
  - Enough bits for cardinality → enough bits to index within cardinality bits
  - OR set bits → aggregate maximum index

Loglog Counting of Large Cardinalities (2003)

Super LogLog

Algorithm tweaks to reduce outliers

Truncation Rule. When collecting register values in order to produce the final estimate, retain only the $m_0 := \lfloor \theta_0 m \rfloor$ smallest values and discard the rest. There θ_0 is a real number between 0 and 1, with $\theta_0 = 0.7$ producing near-optimal results. The mean of these registers is computed and the estimate returned is $m_0 \widetilde{\alpha}_m 2^{\frac{1}{m_0} \sum^{\star} M^{(j)}}$, where $\Sigma^{\star}$ indicates the truncated sum. The modified constant $\widetilde{\alpha}_m$ ensures that the algorithm remains unbiased.

Restriction Rule. Use register values that are in the interval $[0\,..B]$, where $\left\lceil \log_2 \left(\frac{N_{\max}}{m} \right) + 3 \right\rceil \leq B$.

Fact 1. *Combining the basic* LogLog *counting algorithm, the Truncation Rule and the Restriction Rule yields the super-*LogLog *algorithm that estimates cardinalities with a standard error of* $\approx \frac{1.05}{\sqrt{m}}$ *when* m *"small bytes" are used. Here a small byte has size* $\left\lceil \log_2 \left\lceil \log_2 \left(\frac{N_{\max}}{m} \right) + 3 \right\rceil \right\rceil$, *that is, 5 bits for maximum cardinalities* $N_{\max}$ *well over* 10^8.

Loglog Counting of Large Cardinalities (2003)

Methods So Far

Algorithm	*Std. Err.* (σ)	*Memory units*	$n = 10^8$, $\sigma = 0.02$
Adaptive Sampling	$1.20/\sqrt{m}$	Records ($\geq$24–bit words)	10.8 kbytes
Prob. Counting	$0.78/\sqrt{m}$	Words (24–32 bits)	6.0 kbytes
Multires. Bitmap	$\approx 4.4/\sqrt{m}$	Bits	4.8 kbytes
LogLog	$1.30/\sqrt{m}$	"Small bytes" (5 bits)	2.1 kbytes
Super-LogLog	$1.05/\sqrt{m}$	"Small bytes" (5 bits)	1.7 kbytes

Loglog Counting of Large Cardinalities (2003)

HyperLogLog Sketches

"estimating the number of distinct elements (the cardinality) of very large data ensembles. Using an auxiliary memory
of m units (typically, "short bytes"), HYPERLOGLOG performs a single pass over the data and produces an estimate √
of the cardinality such that the relative accuracy (the standard error) is typically about 1.04/ m. This improves on the best previously known cardinality estimator, LOGLOG, whose accuracy can be matched by consuming only 64% of the original memory."

Let $h : \mathcal{D} \to [0,1] \equiv \{0,1\}^\infty$ *hash data from domain* $\mathcal{D}$ *to the binary domain.*
Let $\rho(s)$*, for* $s \in \{0,1\}^\infty$*, be the position of the leftmost 1-bit* ($\rho(0001\cdots) = 4$).

Algorithm HYPERLOGLOG (**input** $\mathcal{M}$: multiset of items from domain $\mathcal{D}$).
assume $m = 2^b$ with $b \in \mathbb{Z}_{>0}$;
initialize a collection of m registers, $M[1], \ldots, M[m]$, to $-\infty$;

for $v \in \mathcal{M}$ **do**
  set $x := h(v)$;
  set $j = 1 + \langle x_1 x_2 \cdots x_b \rangle_2$; {*the binary address determined by the first b bits of* x}
  set $w := x_{b+1} x_{b+2} \cdots$; **set** $M[j] := \max(M[j], \rho(w))$;

compute $Z := \left(\sum_{j=1}^{m} 2^{-M[j]} \right)^{-1}$; {*the "indicator" function*}

return $E := \alpha_m m^2 Z$ with α_m as given by Equation (3).

HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm (2007)

Decentralized Network Size Counting

Count the nodes in a decentralized network by gossiping with a count sketch…

- Like the duck counting, but skipping the duck check and counting machines.

A common variant:

- Use timestamps instead of bits (so FM-targeted).
- Threshold the timestamp to count nodes seen since the threshold.

Database Distinct Count Queries

Approximate counting is a fast alternative to COUNT(DISTINCT(…)) in SQL.

- Native implementations for Amazon RedShift and Redis.
- Plugins available for MySQL and Postgres.

https://docs.aws.amazon.com/redshift/latest/dg/hyperloglog-functions.html

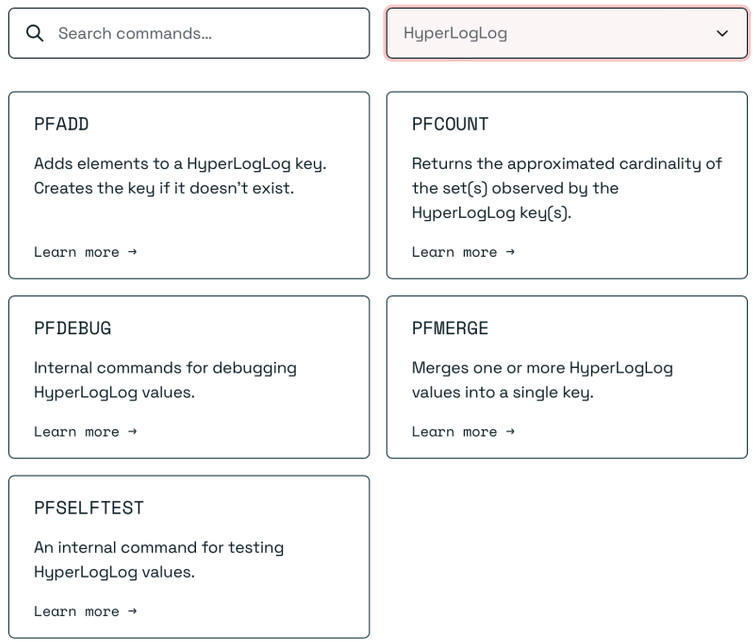
HyperLogLog functions

PDF | RSS

Following, you can find descriptions for the HyperLogLog functions for SQL that Amazon Redshift supports.

Topics

- HLL function
- HLL_CREATE_SKETCH function
- HLL_CARDINALITY function
- HLL_COMBINE function
- HLL_COMBINE_SKETCHES function



https://redis.io/docs/latest/commands/?group=hyperloglog

Generalization to Sum Sketches

The same sketch constructions can be applied to summing data instead of counting data.

- Basic idea for integers:
  - Simulate inserting many v items at once.
  - Do not need to compute v hashes and inserts.
  - Just need to realize the same distributions of bits being set.
  - Some implementation nuances depending on how you want to handle different values for same key…

- More complex operations possible.
  - Fixed point easy.
  - Variance possible in theory, but higher error rates expected.

Approximate Reconciliation Techniques for Sensor Databases (2004)