Comptage probabiliste : entre mathématique et informatique

Philippe Flajolet, INRIA, Rocquencourt

http://algo.inria.fr/flajolet
Routers in the range of Terabits/sec ($10^{14}$b/s).

Google indexes 6 billion pages & prepares to index 100 Petabytes of data ($10^{17}$B).

What about content?

**Message:** Can get a few key characteristics, QUICK and EASY

Combinatorics + algorithms + probabilities + analysis are useful!
From Estan-Varghese-Fisk: traces of attacks
Need number of active connections in time slices.

Incoming/Outgoing flows at 40Gbits/second.
Code Red Worm: 0.5GBytes of compressed data per hour (2001).
CISCO: in 11 minutes, a worm infected 500,000,000 machines.
The situation is like listening to a play of Shakespeare and at the end estimate the number of different words.

Rules: Very little computation per element scanned, very little auxiliary memory.

From Durand-Flajolet, LogLog Counting (ESA-2003):
Whole of Shakespeare, \( m = 256 \) small “bytes” of 4 bits each = 128 bytes

Estimate \( n^\circ \approx 30,897 \) vs \( n = 28,239 \) distinct words. Error: +9.4% w/ 128 bytes!
Uses:

— **Routers:** intrusion, flow monitoring & control

— **Databases:** Query optimization, cf \( M \cup M' \) for multisets; Estimating the size of queries & "sketches".

— **Statistics gathering:** on the fly, fast and with little memory even on “unclean” data \( \simeq \) layer 0 of “*data mining*”. 
This talk:

- **Estimating characteristics of large data streams**
  - sampling; size & cardinality & nonuniformity index ($F_1, F_0, F_2$)
  - power of randomization via hashing
    - Gains by a factor of $>400$ (Palmer et al.)

- **Analysis of algorithms**
  - generating functions, complex asymptotics, Mellin transforms
    - Nice problems for theoreticians.

- **Theory and Practice**
  - Interplay of analysis and design $\rightsquigarrow$ super-optimized algorithms.
Problems on Streams

Given: $S = a$ large stream $S = (r_1, r_2, \ldots, r_\ell)$ with duplicates
- $\|S\|$ = length or size: total # of records ($\ell$)
- $|S|$ = cardinality: # of distinct records ($c$)

◊ How to estimate size, cardinality, etc?

More generally, if $f_v$ is frequency of value $v$: $F_p := \sum_{v \in \mathbb{D}} (f_v)^p$.

Cardinality is $F_0$; size is $F_1$; $F_2$ is indicator of nonuniformity of distribution;
“$F_\infty$” is most frequent element (Alon, Matias, Szegedy, STOC96)

◊ How to sample?
- with or without multiplicity

◊ How to find icebergs, mice, elephants?
1 ICEBERGS—COMBINATORICS HELPS!

Definition: A $k$–iceberg is present in proportion $> 1/k$.

One pass detection of icebergs for $k = 2$ using 1 registers is possible.

— Trigger a gang war: equip each individual with a gun.
— Each guy shoots a guy from a different gang, then commits suicide: Majority gang survives.
— Implement sequentially & adapt to $k \geq 2$ with $k - 1$ registers. (Karp et al. 2003)
How to find an integer while posing few questions?

— Ask if in (1—2), (2—4), (4—8), (8—16), etc?
— Conclude by binary search: cost is $2^{\log_2 n}$.

A paradigm for unbounded search:

• **Ethernet** proceeds by period doubling + randomization.
• **Wake up process for mobile communication** (OCAD: Lavault$^+$)
• **Adaptive data structures**: e.g., extendible hashing tables.

♥ **Approximate Counting**
Approximate counting—probabilities help!

The oldest algorithm (Morris CACM:1977), analysis (F, 1985).
Maintain $F_1$, i.e., counter subject to $C := C + 1$.

**Alg:** Approximate Counting
Initialize: $X := 1$;
Increment: do $X := X + 1$ with probability $2^{-X}$;
Output: $2^X - 2$.

**Theorem:** Count till $n$ probabilistically using $\log_2 \log n + \delta$ bits, with accuracy about $0.59 \cdot 2^{-\delta/2}$.
Beats information theory(!?): 8 bits for counts $\leq 2^{16}$ w/ accuracy $\approx 15\%$. 
10 runs of APCO: value of $X$ ($n = 10^3$)
Methodology:

Paths in graphs \(\mapsto\) Generating Functions:

\[(f_n) \mapsto f(z) := \sum_{n} f_n z^n.\]

Here: Symbolically describe all paths:

\[
(a_1)^* b_1 (a_2)^* b_2 (a_3)^* \quad \text{since} \quad \frac{1}{1-f} = 1 + f + f^2 + \cdots \simeq (f)^*.
\]

Perform probabilistic valuation \(a_j \mapsto q^j; b_j \mapsto 1 - q^j\):

\[
H_3(z) = \frac{q^{1+2}z^2}{(1 - (1 - q)z)(1 - (1 - q^2)z)(1 - (1 - q^3)z)}.
\]

\(\Diamond\) (Prodinger’94) Euler transform \(\xi := z/(1-z)\): \(z H_k(z) = \frac{q^{(k)}_2 \xi^{k-1}}{(1 - \xi q) \cdots (1 - \xi q^k)}\).

Exact moments of \(X\) and estimate \(q^X\) via Heine’s transformation of \(q\)-calculus: mean is unbiased, variance \(\sim 0.59\).
Partial fraction expansions $\sim$ asymptotic distribution = quantify typical behaviour + risk! (Exponential tails $\gg$ Chebyshev ineq.)

We have $\mathbb{P}_n(X = \ell) \sim \phi(n/2^{\ell})$, where $(q)_{j} := (1 - q) \cdots (1 - q_j)$.}

\[
\phi(x) := \sum_{j \geq 0} \frac{(-1)^j q^{(j)} e^{-xq^{-j}}}{(q)_{\infty}(q)_{j}}
\]

Fluctuations: $\ldots, \frac{n}{2L}, \ldots, \frac{n}{4}, \ldots$ depend on $L = \lceil \log_2 n \rceil$.

cf. Szpankowski, Mahmoud, Fill, Prodinger, $\ldots$

Analyse storage utilization via Mellin transform
Approximate Counting

Mean\( (X) - \log_2 n: \)

\[
E(X) - \log_2(n) \pm 0.273954
\]

\[
E(X) - \log_2(n) \pm 0.273952
\]

\[
E(X) - \log_2(n) \pm 0.27395
\]

\[
E(X) - \log_2(n) \pm 0.273948
\]

\[
E(X) - \log_2(n) \pm 0.273946
\]

The Mellin transform (F. Régnier Sedgewick 1985); (FLGoDu 1995)

\[
f^*(s) := \int_0^\infty f(x)x^{s-1} \, dx.
\]

Mapping properties (complex analysis): \( \text{Asympt}(f) \leftrightarrow \text{Singularities}(f^*). \)

\( \text{♣} \) dyadic superpositions of models: \( F(x) = \sum \phi \left( \frac{x}{2^\ell} \right) \sim F^*(s) = \frac{f^*(s)}{1 - 2^s}. \)

\( \Rightarrow \) Standard asymptotic terms + (small) fluctuations.
Cultural flashes

— Morris (1977): Counting a large number of events in small memory.
— The power of probabilistic machines & approximation (Freivalds IFIP 1977)
— The FTP protocol: Additive Increase Multiplicative Decrease (AIMD) leads to similar functions (Robert et al, 2001)
— Probability theory: Exponentials of Poisson processes (Yor et al, 2001)
Randomization and hashing

**Theme:** randomization is a major algorithmic paradigm.

- Cryptography (implementation, attacks)
- Combinatorial optimization (smoothing, random rounding).
- **Hashing** and direct access methods
  - Produce (seemingly) uniform data from actual ones;
  - Provide reproducible chance
Can get random bits from nonrandom data: **Works fine!**

To be or not to be...  

Justifies: 

__The Angel__  

__Daemon__

__Model__
3 COUPON COLLECTOR COUNTING

Let a flow of people enter a room.

— *Birthday Paradox*: It takes on average 23 to get a *birthday collision*

— *Coupon Collector*: After 365 persons have entered, expect a *partial collection* of $\sim 231$ different days in the year; it would take more than 2364 to reach a full collection.

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st birthday coll.</td>
<td>complete coll.</td>
</tr>
<tr>
<td>$\mathbb{E}_n(B) \sim \sqrt{\frac{\pi n}{2}}$</td>
<td>$\mathbb{E}_n(C) = nH_n \sim n \log n$</td>
</tr>
<tr>
<td>$\approx ne^{-1}$</td>
<td></td>
</tr>
</tbody>
</table>

**BP**: Suppose we *didn’t know* the number $N$ of days in the year but could identify people with the same birthday. Could we *estimate* $N$?
**Coupon Collector Counting**

**First Counting Algorithm**: Estimate cardinalities $\equiv \#$ of distinct elements. Motivated by query optimization in data bases. (Whang$^+$, ACM TODS 1990)

**Alg**: Coupon Collector Counting

Set up a table $T[1..m]$ of $m$ bit-cells.
— for $x$ in $S$ do mark cell $T[h(x)]$;
Return $-m \log V$, where $V$ := fraction of empty cells.

Alg. is indep. of replications.
Let $n$ be sought cardinality. Then $\alpha := n/m$ is filling ratio.
Expect $V \approx e^{-\alpha}$ empty cells by classical analysis of occupancy. Distribution is concentrated. Invert!
Count cardinalities till $N_{\text{max}}$ using $\frac{1}{10}N_{\text{max}}$ bits, for accuracy (standard error) $= 2\%$.

Generating functions for occupancy; Stirling numbers; basic depoissonization.


4 SAMPLING

Classical sampling (Vitter, ACM TOMS 1985)

**ALG:** Reservoir Sampling (with multiplicities)
Sample $m$ elements from $S = (s_1, \ldots, s_N)$; ($N$ unknown a priori)
Maintain a cache (reservoir) of size $m$;
— for each coming $s_{t+1}$:
  place it in cache with probability $m/(t+1)$; drop random element;
Can we sample values (i.e., without multiplicity)?
Algorithm due to (Wegman, ca 1984, unpub.), analysed by (F.1990).

Sample of size \( \leq b \):
depth \( d = 0, 1, 2, \ldots \)

\[ h(x) = 00...0 \]

\[ c x a s d \]

\[ c s d \]

\[ h(x) = 0... \]

\[ s d f h \]

\[ h(x) = 00... \]

**Alg:** Adaptive Sampling (without multiplicities)
Get a sample of size \( m \) from \( S' \)’s values.

Set \( b := 4m \) (bucket capacity);
— Oversample by adaptive method;
– Get sample of \( m \) elements from the \((b \equiv 4m)\) bucket.
Analysis.

View collection of records as a set of bitstrings.

Digital tree aka trie, paged version:

\[
\begin{cases}
    \text{Trie}(\omega) \equiv \omega \text{ if } \text{card}(\omega) \leq b \\
    \text{Trie}(\omega) = \text{Trie}(\omega \setminus 0) \text{Trie}(\omega \setminus 1) \text{ if } \text{card}(\omega) > b
\end{cases}
\]

(Underlies dynamic and extendible hashing, paged DS, etc)

Depth in Adaptive Sampling is length of leftmost branch;
Bucket size is # of elements in leftmost page.)
**Bonus:** Second Counting Algorithm for cardinalities.

Let $d :=$ sampling depth; $\xi :=$ sample size.

**Theorem [F90]:** $X := 2^d \xi$ estimates the cardinality of $S$ using $b$ words of memory, in a way that is unbiased and with standard error $\approx 1.20/\sqrt{b}$.

- $1.20 \approx 1/\sqrt{\log 2}$: with $b = 1,000W$, get 4% accuracy.
- Distributional analysis by (Louchard RSA 1997).
- Related to folk algorithm for leader election on channel: “Talk, flip coin if noisy; sleep if Tails; repeat!”
- Related to “tree protocols with counting” $\gg$ Ethernet. Cf (Greenberg-F-Ladner JACM 1987).
Cardinality Estimators

\[ F_0 = \text{Number of different values} \]

- 1983–1985: (F-Martin, FOCS+JCSS) Probabilistic Counting
- 1984–1990: (Wegner) (F90 COMP) Adaptive Sampling
- 1996: (Alon et al, STOC) \( F_p \) statistics \( \sim \text{later} \)
- 2000: (Indyk FOCS) Stable Law Counting \( \sim \text{later} \)
- 2001: (Estan-Varghese SIGCOMM) Multiresolution Bitmap
- 2003: (Durand-F ESA) Loglog Counting
- 2005: (Giroire) MinCount
- 2006: (DFFM) HyperLoglog

Note: suboptimal algorithms can be useful!
5 PROBABILISTIC COUNTING

Third Counting Algorithm for cardinalities:

**Algorithm (Alg):** Probabilistic Counting
Input: a stream $S$; Output: cardinality $|S|$
For each $x \in S$ do /* $\rho \equiv$ position of leftmost 1-bit */
   Set BITMAP[$\rho$(hash($w$))] := 1; od;
Return $P$ where $P$ is position of first 0.
— $P$ estimates $\log_2(\varphi n)$ for $\varphi \approx 0.77351$

— **Average** over $m$ trials $A = \frac{1}{m}[A_1 + \cdots + A_m]$; return $\frac{1}{\varphi}2^A$.

— In fact, use **stochastic averaging**, which needs only one hash function: $S \mapsto (S_{000}, \ldots, S_{111})$.

— Analysis provides

$$\varphi = \frac{e^\gamma}{\sqrt{2}} \prod_{m \geq 2}^* m^{\epsilon(m)}, \quad \epsilon(m) := (-1)^\sum \text{bits}(m).$$

$\epsilon(19) = \epsilon(\langle 10011 \rangle_2) = (-1)^3 = -1$. **Standard error** is $0.78/\sqrt{m}$ for $m$ Words of $\log_2 N$ bits. + **Exponential Tails $\gg$ Chebyshev**.

(AMS96) and subsequent literature claim wrongly that several hash functions are needed!
Theorem [FM85]: Prob. Count. is asymptotically unbiased. Accuracy is $\frac{0.78}{\sqrt{m}}$ for $m$ Words of size $\log_2 N$. E.g. 1,000W = 4kbytes $\sim$ 2.5% accuracy.

Proof: trie analysis

$1 \cdot (e^{x/8} - 1)(e^{x/4} - 1)(e^{x/2} - 1)\epsilon(n)\epsilon(n)
(1 - q)(1 - q^2)(1 - q^4) \cdots = \sum_n (-1)^{\sum \text{bits}(n)} q^n.$

Distribution:

$Q(x) := e^{-x/2} \prod_{j=0}^{\infty} (1 - e^{-x2^j})$

$P_n(X = \ell) \sim Q\left(\frac{n}{2\ell}\right)$

+ Mellin requires $N(s) := \sum_{n \geq 1} \frac{\epsilon(n)}{n^s}$. One finds $\log_2 \varphi \equiv -\Gamma'(1) - N'(0) + \frac{1}{2}$, &c.
Data mining of the Internet graph
(Palmer, Gibbons, Faloutsos, Siganos 2001)

Internet graph: 285k nodes, 430k edges.

For each vertex $v$, define ball $B(v; R)$ of radius $R$.

Want: histograms of $|B(v, R)|$ \( R = 1 \ldots 20 \)

Get it in minutes of CPU rather than a day (400× speedup)

+ Sliding window usage (Motwani et al) Fusy & Giroire for MinCount.
Fourth Counting Algorithm for cardinalities: (Durand-F, 2003/DFFM, 2006)

Claim: the best algorithm on the market!

- Hash values and get $\rho(h(x)) = \text{position of leftmost 1-bit} = \text{a geometric RV } G(x)$.

- To set $S$ associate $R(S) := \max_{v \in S} G(v)$.

- Max of geometric RVs are well-known (Prodinger*).
$R(s)$ estimates $\sim \log(\hat{\varphi} \text{card}(S))$, with $\hat{\varphi} := e^{-\gamma \sqrt{2}}$.

• Do stochastic averaging with $m = 2^\ell$:
E.g., $S \cong \langle S_{00}, S_{01}, S_{10}, S_{11} \rangle$: count separately.
Return $\frac{m}{\hat{\varphi}} 2^{\text{Average}}$.

++ Switch to Coupon Collector Counting for small and large cardinalities.
++ Optimize by pruning discrepant values $\sim$ superLogLog or better by harmonic means $\sim$ superLogLog (≪ Chassaing-Gerin, 2006)
**Theorem.** LogLog needs \( m \) “bytes”, each of length \( \log_2 \log N \).

Accuracy is: \( \frac{1.30}{\sqrt{m}} \).

**Proof:** Generating Functions + Saddle-point depoissonization (Jacquet-Szpankowski) + Mellin. \( 1.30 \equiv \sqrt{\frac{1}{12} \log^2 2 + \frac{1}{6} \pi^2} \).

Whole of *Shakespeare*:

\( m = 256 \) small “bytes” of 4 bits each = 128 bytes

Error is +9.4% for 128 bytes(!!)
An aside: **Analytic depoissonization** (JaSz95+)

- **Problem:** Recover asympt. $f_n$ from $f(z) = \sum_n f_n z^n$?

- Intuition: “with luck” $f_n \sim \phi(n)$ where $\phi(z) := e^{-z} f(z)$ is Poisson g.f.

(Here: “Luck” means good lifting of $\phi(z)$ to $\mathbb{C} \equiv$ Poisson flow of complex rate!)

\[
f_n = \frac{n!}{2i\pi} \oint f(z) \frac{dz}{z^{n+1}} \approx \phi(n)
\]
Features: Errors \( \approx \text{Gaussian} \), seldom more than \( 2 \times \) standard error. Algorithm *scales down* (for small cardinalities) and Algorithm *scales up* (large memory size): **HYBRIDIZE with Collision Counting.**

Mahābhārata: 8MB, 1M words, 177601 diff. HTTP server: 400Mb log pages 1.8 M distinct req.

<table>
<thead>
<tr>
<th>( m )</th>
<th>( 2^6 ) (50by)</th>
<th>( 2^{10} ) (0.8kby)</th>
<th>( 2^{14} ) (12kb)</th>
<th>( 2^{18} ) (200kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs:</td>
<td>8.9%</td>
<td>2.6%</td>
<td>1.2%</td>
<td>0.32%</td>
</tr>
<tr>
<td>( \sigma ):</td>
<td>11%</td>
<td>2.8%</td>
<td>0.7%</td>
<td>0.36%</td>
</tr>
</tbody>
</table>
Summary

*Analytic results* ($\lg \equiv \log_2$): Alg/Mem/Accuracy

<table>
<thead>
<tr>
<th></th>
<th>CouponCC</th>
<th>AdSamp</th>
<th>ProbC</th>
<th>LogLog</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\approx \frac{N}{10}$ bits</td>
<td>$m \cdot \lg N$ Words</td>
<td>$m \cdot \frac{N}{m}$ Words</td>
<td>$m \cdot \lg \frac{N}{m}$ Bytes</td>
<td></td>
</tr>
<tr>
<td>$\approx 2%$</td>
<td>$\frac{1.20}{\sqrt{m}}$ W</td>
<td>$\frac{0.78}{\sqrt{m}}$ W</td>
<td>$\approx \frac{1.30-1.05}{\sqrt{m}}$ By</td>
<td></td>
</tr>
</tbody>
</table>

$F_0$ statistics, $N = 10^8$ & 2% error

— Coupon Collector Counting = 1 Mbyte + used for corrections
— Adaptive Sampling = 16 kbytes + sampling, unbiased
— Probabilistic Counting: = 8 kbytes + sliding window
— Multiresolution bitmap (analysis?) = 5 kbytes?
— MinCount ©Giroire = 4 kbytes + sliding window
— Loglog Counting = 2 kbytes + mice/elephants

+ Sliding window, Mice/elephant problem (A. Jean-Marie, O. Gandouët)
FREQUENCY MOMENTS: $F_2$

Recall: Alon, Matias, Szegedy (STOC 1996)***

$$F_2 := \sum_v (f_v)^2,$$

where $f_v$ is frequency of value $v$.

An elegant idea: $\text{flip}(x) \equiv \epsilon(x) = \pm 1$ based on $\text{hash}(x)$.

**Alg:** F2;
Initialize $Z := 0$;
For each $x$ in $S$ do $Z := Z + \text{flip}(x)$.
Return $Z^2$.
Collect $m$ $Z$-values and average, with $T$-transform.
\[
\mathbb{E}(Z^2) = \mathbb{E}\left( \sum_{x \in S} \varepsilon(v) \right)^2 = \mathbb{E}\left( \sum_j f_j \cdot \varepsilon(j) \right)^2 = \sum_j (f_j)^2.
\]

(Actually, AMS prove stronger complexity result by complicated (impractical?) algorithm.) (What about stochastic averaging?)
**Indyk’s $F_p$ algorithm**

A beautiful idea of Piotr Indyk (FOCS 2000)*** for $F_p$, $p \in (0, 2)$.

- Stable law of parameter $p \in (0, 2)$: $E(e^{i t X}) = e^{-|t|^p}$.

No second moment; no 1st moment if $p \in (0, 1)$.

\[ c_1 X_1 + c_2 X_2 \sim \mu X, \text{ with } \mu := (c_1^p + c_2^p)^{1/p}. \]

---

**ALG: $F_p$**

Initialize $Z := 0$;

For each $x$ in $S$ do $Z := Z + \text{Stable}_\alpha(x)$.

Return $Z$.

Estimate $F_p$ parameter from $m$ copies of $Z$-values.

---

Remark: Use of $\log(|Z|)$ to estimate seems better than median(?)
8 CONCLUSIONS

For streams, using practically $O(1)$ storage, one can:
— Sample positions and even distinct values;
— Estimate $F_0, F_1, F_2, F_p$ ($0 < p \leq 2$) even for huge data sets;
— Need no assumption on nature of data.

The algorithms are based on randomization $\mapsto$ Analysis fully applies
— They work exactly as predicted on real-life data;
— They often have a wonderfully elegant structure;
— Their analysis involves beautiful methods for AofA: “Symbolic modelling by generating functions, Singularity analysis, Saddle Point and analytic depoissonization, Mellin transforms, stable laws and Mittag-Leffler functions, etc.”
That’s All, Folks!