



PARISTIC
NOV 2006



*Quantitative Algorithmics
of Massive Data Sets*

**Algorithmique quantitative
de grands flux de données**

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).

Can get a few key characteristics, QUICK and EASY

This talk:

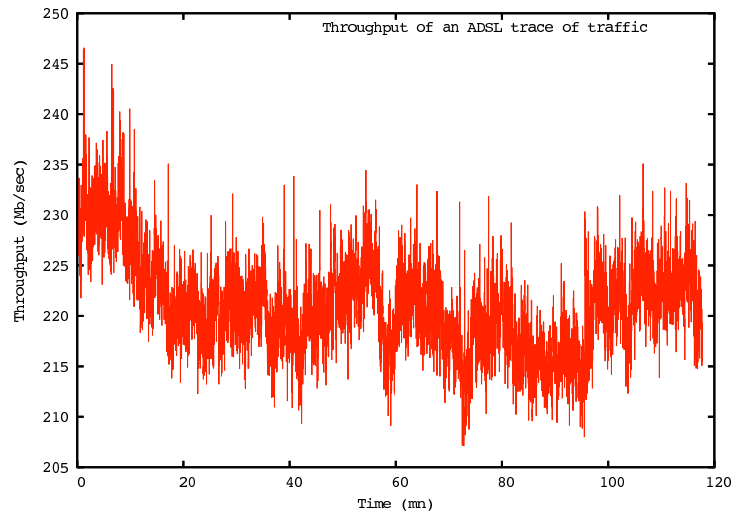


Zoom on some recent research directions \neq geopolitics.

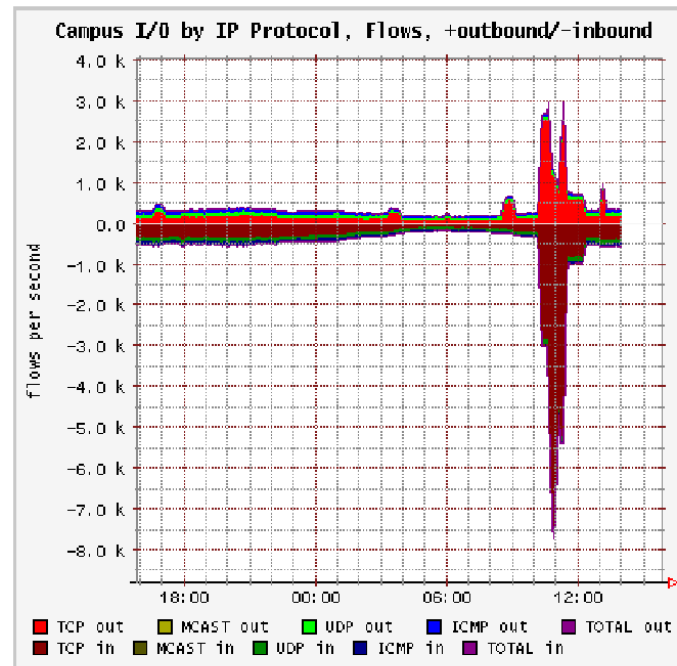
+ Interplay between **theory and practice**.

Combinatorics + algorithms + probabilities + analysis are useful!

Traces of attacks: Number of active connections in time slices.



(Raw ADSL traffic)



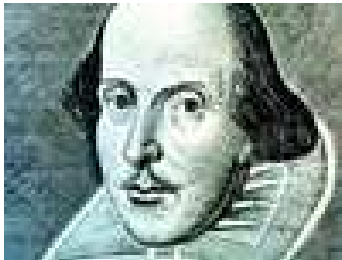
(Attack)

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.

Left: ADSL FT@Lyon 1.5×10^8 packets (21h–23h). Right: (Estan-Varghese-Fisk) different incoming/outgoing connections

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 = **Stream algorithms**.



From Durand-Flajolet, LogLog Counting (ESA-2003):

Whole of Shakespeare, $m = 256$ small "bytes" of 4 bits each = 128bytes

```
ghfffghfghgghggggghghheehfhfhgghghghhfgffffhhhiigfhhffgfiihfhhh  
igigighfghihfffghigihghigfhhgeegeghgghhhgghhfhidiigihighihehhfgg  
hfgighigffghdieghhhggghhfhghhfiieffghghihifgggffihgihfggighgiiif  
fjgfgjhhjiihfjhgehghfhfhjhiggghghihigghhihiggiighgfhlgjfgjjmfl
```

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

- **Routers & Networks**: attacks, flow monitoring & control
- **Databases**: Query optimization = estimating the size of queries; also “sketches”.
- “**Raw**” **statistics on massive data**: on the fly, fast and with little memory even on textual data \in “*data mining*”.
(Gains by a factor of 400 in data mining of the internet graph.)



Produce **short algorithms & programs** with $\mathcal{O}(10)$ instructions based on **mathematically oriented engineering**.

- Estimating characteristics of large data streams

- **counters**: “beats” information theory;
- **sampling**: avoid negative effects
- **cardinality estimators**;
- gather statistics for **mice and elephants**;
- **profile** indicators (F_2 , etc)

↷ ◇ Gains by factors in range 100-1000 (!)

- No analysis \implies no algorithm! Algorithms are inherently **probabilistic**.

- analytic combinatorics, complex asymptotics, Mellin transforms
- **Interplay of analysis and design** \rightsquigarrow highly optimized algorithms.

1 ICEBERGS (& COMBINATORICS)

abcdaaaabcafcccaacccaaacacbbbbbaabbbbbbb



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, Shenker, Papadimitriou. 2003)

2 APPROXIMATE COUNTING

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 $2 \log_2 n$.

The **dyadic paradigm** for unbounded search:

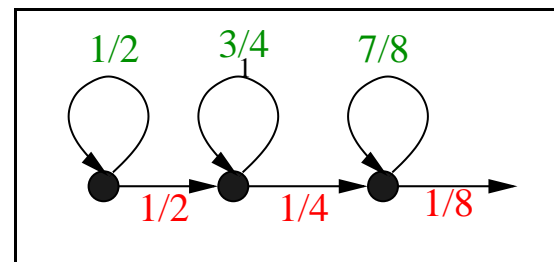
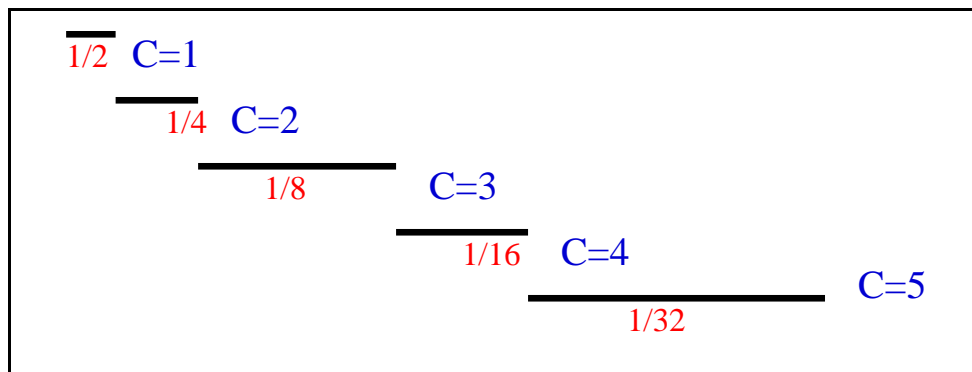
- Ethernet proceeds by period doubling + randomization.
- Wake up procedures for mobile communication (Lavault⁺)
- Adaptive data structures: e.g., extendible hashing tables.

♡ Approximate Counting

Approximate counting & randomization

The oldest algorithm (Morris CACM:1977), analysis (F, 1985).

Maintain **counter** subject to $X := X + 1$.



ALGORITHM: Approximate Counting

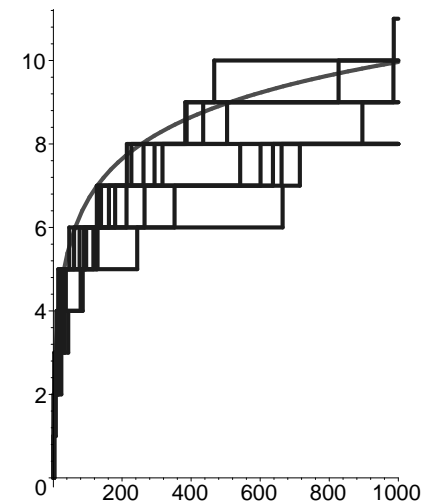
Initialize: $C := 1$;

Increment: do $C := C + 1$ with probability 2^{-C} ;

Output: $2^C - 2$.

Expect C near $\log_2 n$ after n steps, then use only $\log_2 \log n$ bits.

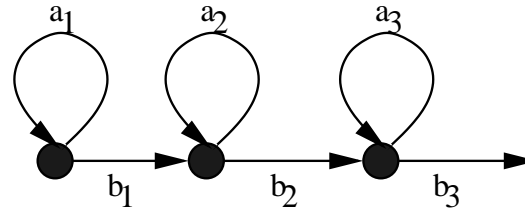
10 runs of of APCO: value of C ($n = 10^3$)



Theorem: Using alternate bases, 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 $\approx 15\%$.

Symbolic methodology:



♥ Generating Functions:

$$(f_n) \mapsto f(z) := \sum_n f_n z^n.$$

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

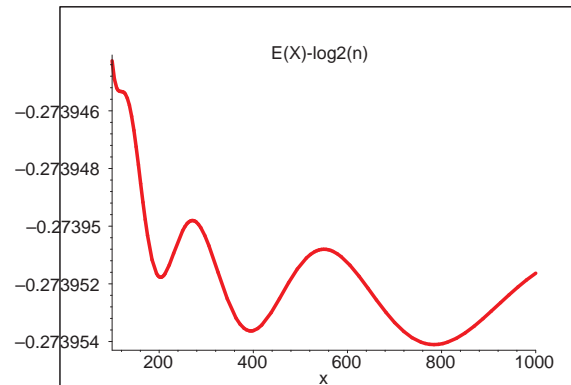
Perform probabilistic valuation $a_j \mapsto q^j$; $b_j \mapsto 1 - q^j$:

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

cf F.+Sedgewick, *Analytic Combinatorics* C.U.P., 2007.

Approximate Counting

Mean(X) $- \log_2 n$:



Asymptotic methodology: Mellin transform (FISE*)

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

Need **singularities** in **complex plane**.

♣ Dyadic superpositions of models \implies Asymptotics with **fluctuations**.

Mellin: Probabilistic counting, loglog counting + Lempel-Ziv compression (Jacquet-Szpa) + dynamic hashing + tree protocols (Jacquet+) + Quadtries &c.

Cultural flashes

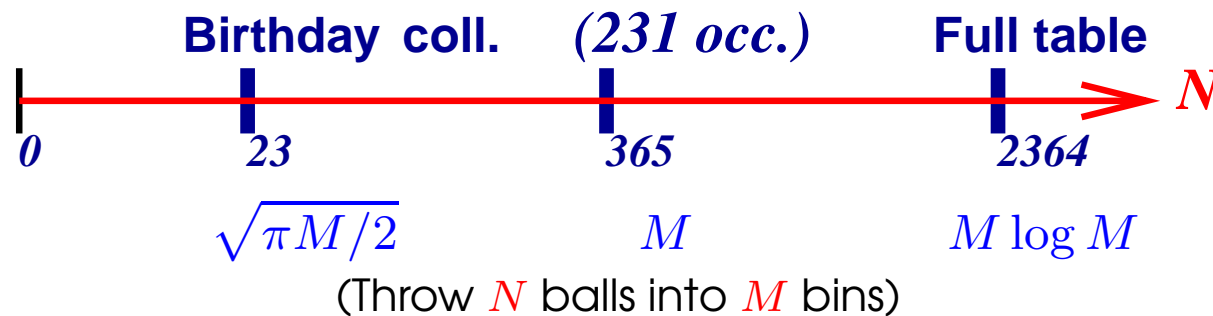
- Morris (1977): Counting a large number of events in small memory.
- The power of probabilistic machines & approximation (Freivalds)
- The TCP protocol: Additive Increase Multiplicative Decrease (AIMD) leads to similar functions (Robert et al, 2001+)
- Probability theory: Exponentials of Poisson processes (Yor et al, 2001)
- **Ethernet is unstable** (Aldous 1986) but **tree protocols are stable!** Cf (Jacquet+).

Probabilities: Coupons & Birthdays

Let a flow of people enter a room.

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

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



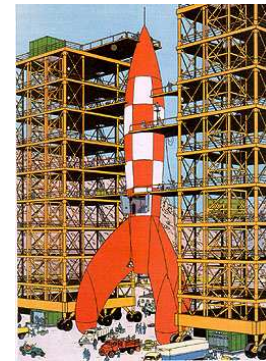
Semi-classical occupancy problems; generating functions; saddle points.

Birthday Paradox Counting (Gedanken!)



Take your space ship to remote planet.

How many days (M) in the year?



Gedanken Algorithm:

— Interview people:

39 Propergol, 42 Sargol, 11 Mordor, ..., 42 Sargol

— Record B = time till first birthday collision;

— $\mathbb{E}(B) \sim \sqrt{\frac{\pi M}{2}} \implies$ return estimate $M \approx \frac{2B^2}{\pi}$;

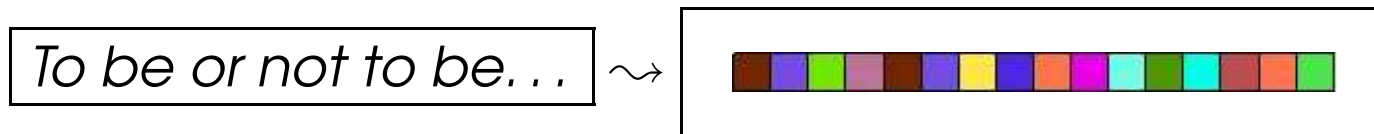
— do some maths and improve!

Sublinear algorithms ...

Randomization and hashing

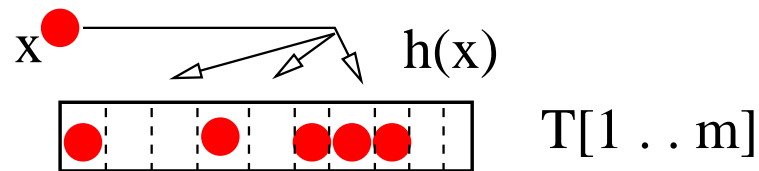
Randomization is a major algorithmic paradigm.

- Cryptography (implementation, attacks, cf RSA)
- Combinatorial optimization (smoothing, random rounding).
- **Hashing** and direct access methods
 - Produce (seemingly) **uniform data** from actual ones;
 - Provide **reproducible chance**



3 HIT COUNTING

First Counting Algorithm: Estimate *cardinalities* \equiv # of **distinct** elements.
Based on Coupon Collector phenomena & motivated by *query optimization in data bases*. (Whang⁺, ACM TODS 1990)



ALGORITHM Hit 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.

— Algorithm is *independent of replications*.

— Let n be sought cardinality and $\alpha := n/m$, *filling ratio*. Expect

$$V \approx e^{-\alpha} \quad \text{fraction of empty cells,}$$

by classical occupancy theory.

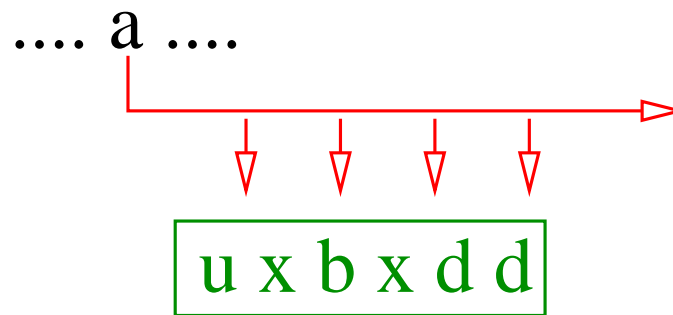
— Distribution is concentrated. *Invert: $n \approx m \log(1/V)$.*

Linear storage complexity: Count cardinalities till N_{\max} using $\frac{1}{10} N_{\max}$ bits, for accuracy (standard error) = 2%.

Generating functions for occupancy; Stirling numbers; basic deois-sonization.

4 SAMPLING

Classical sampling (Vitter, ACM TOMS 1985)



ALGORITHM Reservoir Sampling (with multiplicities)

Sample m elements from $S = (s_1, \dots, s_N)$; (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;

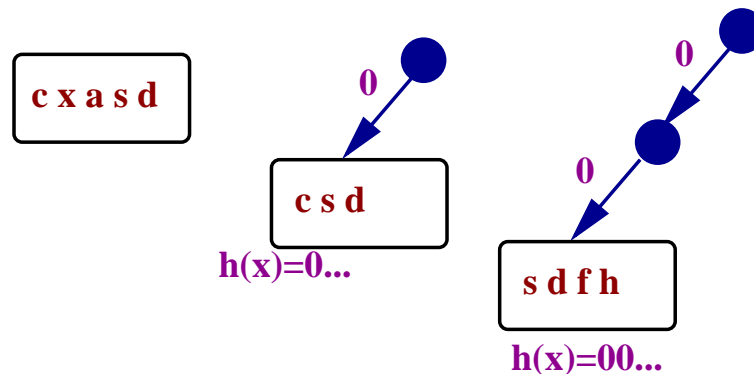
Sample *values* (i.e., without multiplicity)? (Wegman+F).

ALGORITHM: Adaptive Sampling (*without multiplicities*)

Get a sample of size $\leq m$ according to **values**.

— Increase sampling depth and decrease sampling probabilities.

Sample of size $\leq m$:
depth $d = 0, 1, 2, \dots$



Also: get ***b*-sample** by mild oversampling ($m = 4b$).

Analysis makes use of randomness digital trees, generating functions and Mellin transforms.

Adaptive Sampling = Second counting algorithm for *cardinalities*.

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

Theorem: $X := 2^d \xi$ estimates the cardinality of S using m words: unbiased, with standard error $\approx 1.20/\sqrt{m}$.

- $1.20 \doteq 1/\sqrt{\log 2}$: with $b = 1,000W$, get 4% accuracy.
- 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 = Number of different values

- 1983–1985: (F-Martin, FOCS+JCSS) Probabilistic Counting
- 1987–1990: (Whang et al) Hit Counting
- 1984–1990: (Wegner) (F90 COMP) Adaptive Sampling
- 1996: (Alon et al, STOC) F_p statistics \leadsto later
- 2000: (Indyk FOCS) Stable Law Counting \leadsto 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!**

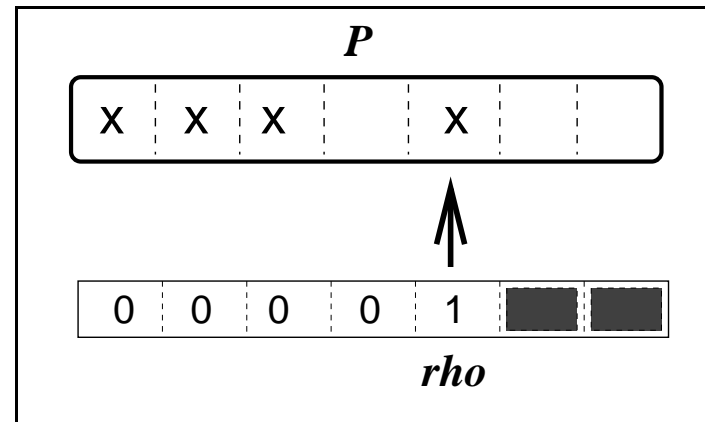
5 **PROBABILISTIC COUNTING**

Observables of hashed values!

Third Counting Algorithm

for cardinalities

$$\begin{array}{lcl} 0.1xxxx & \rightarrow & \mathbb{P} = \frac{1}{2} \\ 0.01xxx & \rightarrow & \mathbb{P} = \frac{1}{4} \\ 0.001xx & \rightarrow & \mathbb{P} = \frac{1}{8} \end{array} \quad :$$



ALGORITHM: Probabilistic Counting

Input: a stream S ; Output: cardinality $|S|$

For each $x \in S$ do /* $\rho \equiv$ position of leftmost 1-bit */

Set $\text{BITMAP}[\rho(\text{hash}(w))] := 1$; od;

Return P where P is position of first 0.

— P estimates $\log_2(\varphi n)$ for $\varphi \doteq 0.77351 = \frac{e^\gamma}{\sqrt{2}} \prod_{m \geq 2}^* m^{\epsilon(m)}$, $\epsilon(m) := (-1)^{\sum \text{bits}(m)}$.

A. Average over m trials $A = \frac{1}{m}[A_1 + \dots + A_m]$; return $\frac{1}{\varphi} 2^A$. Error $\approx \frac{1}{\sqrt{m}}$.

B. Stochastic averaging needs only *one* hash function.

$$x \mapsto h(x) \longrightarrow \begin{cases} 00 & \text{count } n_{00} \\ 01 & \text{count } n_{01} \\ 10 & \text{count } n_{10} \\ 11 & \text{count } n_{11}. \end{cases}$$

Implementation: Hash + switch + update.

Theorem [FM]: Prob. Count. is *asymptotically unbiased*. Accuracy is $\frac{0.78}{\sqrt{m}}$ for m Words of size $\log_2 N$.

Data mining of the Internet graph

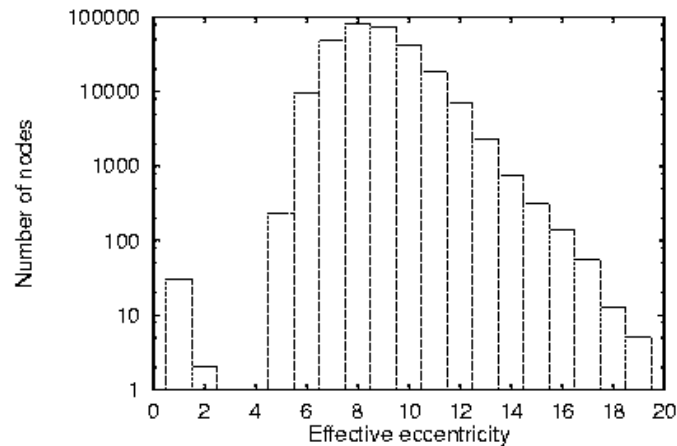
(Palmer, Gibbons, Faloutsos², Siganos 2001)

Internet graph: 285k nodes, 430k edges.

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

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

Get CPU **400 speedup factor** (minutes rather than day)



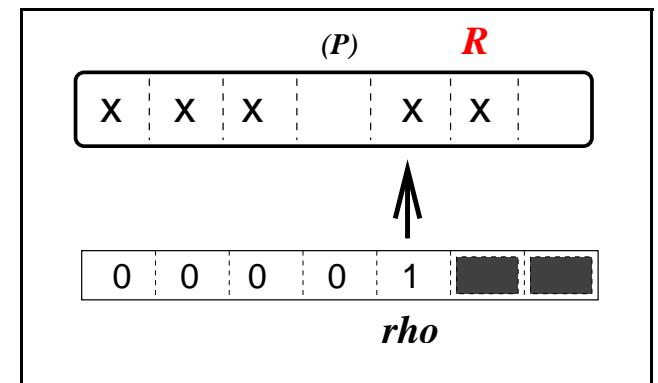
b) Histogram of diameters

+ **Sliding window usage** (Motwani et al) (Fusy & Giroire for MinCount).

6 LOGLOG COUNTING

Fourth Counting Algorithm for cardinalities: (Durand-F, 2003/DFFM, 2006)

- Hash values $\rightarrow \rho(h(x)) =$ position of leftmost 1-bit = a geometric RV $G(x)$.
- To set S associate $R(S) := \max_{v \in S} G(v)$.



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

- Do stochastic averaging with $m = 2^\ell$:

Return $\frac{m}{\hat{\varphi}} 2^{\text{Average}}$.

Engineering HyperLogLog

©Durand+F+Fusy+Meunier, 2006.

++ **Optimize** by limiting effect of discrepant values: use **harmonic means of $2^R \rightsquigarrow$ HyperLogLog** (\ll Chassaing-Gerin, 2006). Asymptotically near-optimal?.

+ Correct **nonasymptotic regime** by switching to **Hit Counting**.

+ Extend **beyond nominal capacity** by random allocation correction.

Theorem. HyperLogLog needs m “bytes”, each of length $\log_2 \log N$.
Accuracy is: $\frac{1.05}{\sqrt{m}}$.

Distributed implementation: exchange registers (a few kilobytes) and can even **compress**, to **reduce storage by extra factor of 3**.

Whole of Shakespeare:



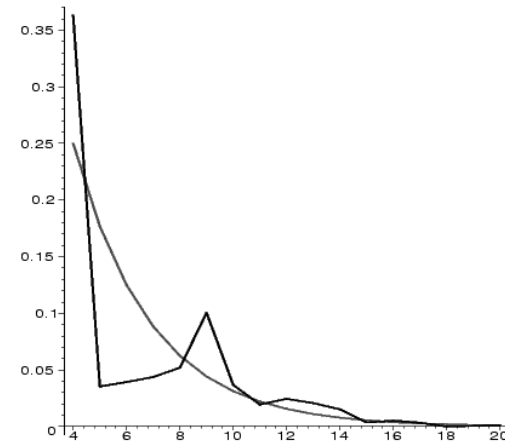
$m = 256$ small "bytes" of 4 bits each = 128bytes

```
ghfffghfghgghggggghghheehfhfhhgghghghhfghfffhhiigfhhffgfiihfhhh  
igigighfghfffghigihghigfhhgeegeghgghhhgghhfhidiigihighihehhfghg  
hfgighigffghdieghhhggghhfhghfiiheffghghihifgggffihgihfggighgiiif  
fjgfgjhhjiihfjgehghfhhfhjhiggghghihigghihihgiighghfhlgjfgjjmfl
```

Estimate $n^\circ \approx 30,897$ against $n = 28,239$ distinct words

Error is +9.4% for 128 bytes(!!)

Features: Errors \approx *Gaussian*, seldom more than $2\times$ standard error.



Mahābhārata: 8MB, 1M words, 177601 diff.

HTTP server: 400Mb log pages 1.8 M distinct req.

m	2^6 (50by)	2^{10} (0.8kby)	2^{14} (12kb)
Obs:	8.9%	2.6%	1.2%
σ :	11%	2.8%	0.7%

Example: Mining the human genome

(by Frédéric Giroire)

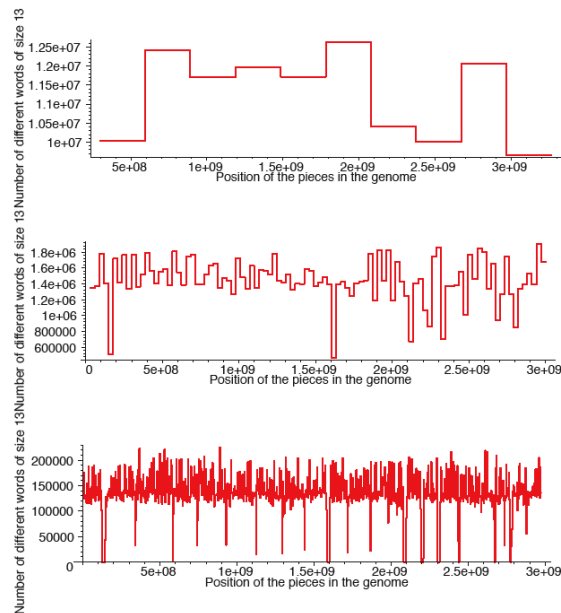
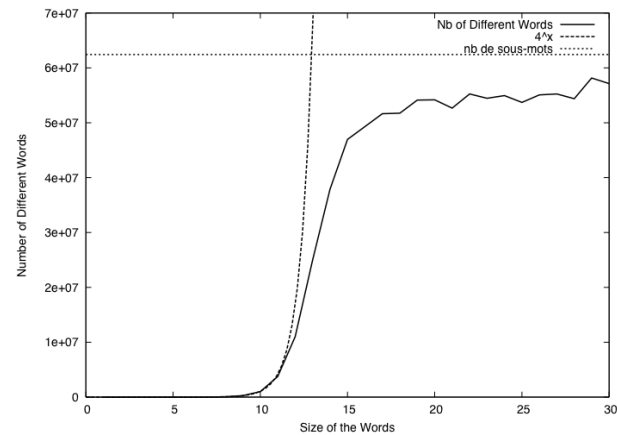


Figure 1: Mots différents de taille 13 dans des séquences consécutives du génome le 3M, 30M et 300M de paires de bases.



Patterns of length 13 in sections of human genome (3Gbp).
Number of different patterns of length ℓ in Chr. 20 (60Mbp).

Summary

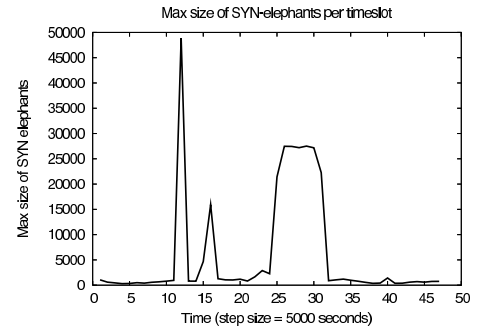
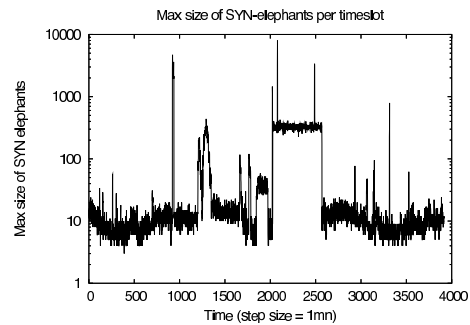
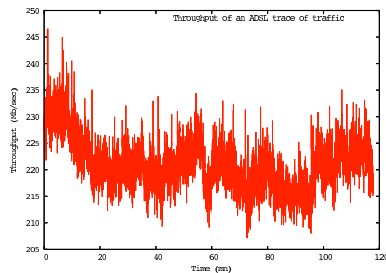
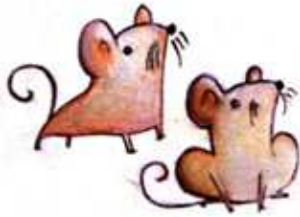
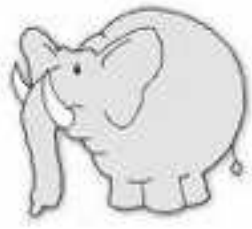
Analytic results ($\lg \equiv \log_2$): Alg/Mem/Accuracy

CouponCC	AdSamp	ProbC	LogLog
$\approx \frac{N}{10}$ bits	$m \cdot \lg N$ Words	$m \cdot \lg \frac{N}{m}$ Words	$m \cdot \lg \lg \frac{N}{m}$ Bytes
$\approx 2\%$	$\frac{1.20}{\sqrt{m}}$ W	$\frac{0.78}{\sqrt{m}}$ W	$\approx \frac{1.30 - 1.05}{\sqrt{m}}$ By

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

7 ELEPHANTS AND MICE



DoS attacks: Largest elephants; (here) elephant = number of packets to a given destination.

— By **LogLog Counting and p -filtering**: estimate $\text{card}(\Phi)$ and $\text{card}(\Phi_{pM} \cup \Phi_{pE})$ (Jean-Marie+Gandouët06).

— By **Bloom filters** (Azzana+Robert+Chabchoub)

8 PROFILING: FREQUENCY MOMENTS

Alon-Matias-Szegedy: $F_p := \sum_v (f_v)^p$ where $f_v :=$ frequency of value v .

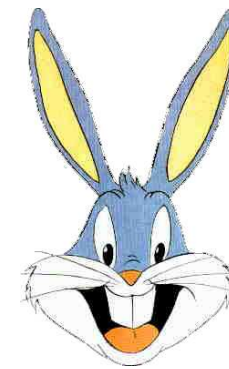
dim = n



→

dim $\approx \log n$

Johnson-Lindenstrauss
embeddings
dimension reduction
Indyk's beautiful ideas



Use of **random Gaussian** projections for F_2 ; **Stable laws** for $0 \leq p \leq 2$.

Indyk's F_p algorithm

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

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

$c_1X_1 + c_2X_2 \stackrel{\mathcal{L}}{\cong} \mu X$, with $\mu := (c_1^p + c_2^p)^{1/p}$.

ALGORITHM: Frequency Moments 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(?)

Conclusions

For huge data streams, using reduced/minimal storage, one can:

- Sample positions and (distinct) values;
- Count events and cardinalities (F_1, F_0);
- Estimate profiles of data (F_p), $0 < p \leq 2$;
- Get informations on elephants and mice.



The algorithms are based on randomization \mapsto Analysis fully applies

- They tend to work **exactly** as predicted on real-life data;
- They often have a very elegant structure;
- Their analysis involves numerous methods of AofA: “Symbolic modelling by generating functions, Singularity analysis, Saddle-point and analytic depois-sonization, Mellin transforms, Stable laws, etc.”

That's All, Folks!

