Probabilistic Counting Algorithms
for Data Base Applications

PHILIPPE FLAJOLET

INRIA, Rocquencourt, 78153 Le Chesnay, France

AND

G. NIGEL MARTIN

*IBM Development Laboratory, Hursley Park,
Winchester, Hampshire SO212JN, United Kingdom*

This paper introduces a class of probabilistic counting algorithms with which one can estimate the number of distinct elements in a large collection of data (typically a large file stored on disk) in a single pass using only a small additional storage (typically less than a hundred binary words) and only a few operations per element scanned. The algorithms are based on statistical observations made on bits of hashed values of records. They are by construction totally insensitive to the replicative structure of elements in the file; they can be used in the context of distributed systems without any degradation of performances and prove especially useful in the context of data bases query optimisation.   *(C)* 1985 Academic Press, Inc.

1. INTRODUCTION

As data base systems allow the user to specify more and more complex queries, the need arises for efficient processing methods. A complex query can however generally be evaluated in a number of different manners, and the overall performance of a data base system depends rather crucially on the selection of appropriate *decomposition strategies* in each particular case.

Even a problem as trivial as computing the intersection of two collections of data *A* and *B* lends itself to a number of different treatments (see, e.g., [7]):

1.   Sort *A*, search each element of *B* in *A* and retain it if it appears in *A*;

2.   sort *A*, sort *B*, then perform a merge-like operation to determine the intersection;

3.   eliminate duplicates in *A* and/or *B* using hashing or hash filters, then perform Algorithm 1 or 2.

Each of these evaluation strategy will have a cost essentially determined by the number of records *a, b* in *A* and *B*, and the number of *distinct* elements α, β in *A* and *B*, and for typical sorting methods, the costs are:

182

for strategy 1: $O(a \log \alpha + b \log \alpha)$;

for strategy 2: $O(a \log a + b \log b + a + b)$....

In a number of similar situations, it appears thus that, apart from the *sizes* of the files on which one operates (i.e., the number of records), a major determinant of efficiency is the *cardinalities* of the underlying sets, i.e., the *number of distinct elements* they comprise.

The situation gets much more complex when operations like projections, selections, multiple joins in combination with various boolean operations appear in queries. As an example, the relational system *system R* has a sophisticated query optimiser. In order to perform its task, that programme keeps several statistics on relations of the data base. The most important ones are the sizes of relations as well as the number of *different* elements of some key fields [8]. This information is used to determine the selectivity of attributes at any given time in order to decide the choice of keys and the choice of the appropriate algorithms to be employed when computing relational operators. The choices are made in order to minimise a certain cost function that depends on specific CPU and disk access costs as well as sizes and cardinalities of relations or fields. In system *R*, this information is periodically recomputed and kept in catalogues that are companions to the data base records and indexes.

In this paper, we propose efficient algorithms to estimate the cardinalities of multisets of data as are commonly encountered in data base practice. A trivial method consists in determining card(M) by building a list of all elements of M without replication; this method has the advantage of being exact but it has a cost in number of disk accesses and auxiliary storage (at least $O(a)$ or $O(a \log a)$ if sorting is used) that might be higher than the possible gains which one can obtain using that information.

The method we propose here is probabilistic in nature since its result depends on the particular hashing function used and on the particular data on which it operates. It uses minimal extra storage in core and provides practically useful estimates on cardinalities of large collections of data. The accuracy is inversely related to the storage: using 64 binary words of typically 32 bits, we attain a typical accuracy of 10%; using 256 words, the accuracy improves to about 5%. The performances do not degrade as files get large: with 32 bit words, one can safely count cardinalities well over 100 million. The only assumption made is that records can be hashed in a suitably pseudo-uniform manner. This does not however appear to be a severe limitation since empirical studies on large industrial files [5] reveal that *careful* implementations of standard hashing techniques do achieve practically uniformity of hashed values. Furthermore, by design, our algorithms are totally insensitive to the replication structures of files: as opposed to *sampling techniques*,[1]

---

[1] The simplest sampling algorithm is: take a sample of size N_0 of a file of size N. Estimate the cardinality v_0 of the sample using any direct algorithm and return $v_0(N/N_0)$ as estimate of the cardinality of the whole file.

the result will be the same whether elements appear a million times or just a few times.

From a more theoretical standpoint, these techniques constitute yet another illustration of the gains that may be achieved in many situations through the use of probabilistic methods. We mention here Morris' approximate counting algorithm [6] which maintains approximate counters with an expected constant relative accuracy using only

$$\log_2 \log_2 n + O(1)$$

bits in order to count up to n. Morris' algorithm (see [2] for a detailed analysis that has analogies to the present paper) may be used to reduce by a factor of 2 the memory size necessary to store large statistics on a large number of events in computer systems.

The structure of the paper is as follows: in Section 2, we describe a basic counting procedure called COUNT that forms the basis of our algorithms. It may be worth noting that non-trivial analytic techniques enter the justification, and actually the design, of the algorithms; these techniques are also developed in Section 2. Section 3 presents the actual counting algorithms based on this COUNT procedure and on the probabilistic tools of Section 2. Finally, Section 4 concludes with several indications on contexts in which the methods may be used: most notably they can be employed *on the fly* as well as in the context of *distributed processing* with minimal exchanges of information between processors and without any degradation of performances. Preliminary results about this work have been reported in [3].

2. A Probabilistic Counting Procedure and Its Analysis

The Basic Counting Procedure

We assume here that we have at our disposal a hashing function *hash* of the type:

function hash(x: records): scalar range $[0 \cdots 2^L - 1]$,

that transforms records into integers sufficiently uniformly distributed over the scalar range or equivalently over the set of *binary strings* of length L. For y any non-negative integer, we define bit(y, k) to be the kth bit in the binary repesentation of y, so that

$$y = \sum_{k \geqslant 0} \text{bit}(y, k) 2^k.$$

We also introduce the function $\rho(y)$ that represents the position of the least significant 1-bit in the binary representation of y, with a suitable convention for $\rho(0)$:

$$\rho(y) = \min_{k \geqslant 0} \text{bit}(y, k) \neq 0 \qquad \text{if} \quad y > 0$$

$$= L \qquad \qquad \text{if} \quad y = 0.$$

(Thus ranks are numbered starting from *zero*.)

We observe that if the values hash(x) are uniformly distributed, the pattern $0^k 1 \cdots$ appears with probability 2^{-k-1}. The idea consists in recording observations on the occurrence of such pattrns in a vector $BITMAP[0 \ldots L-1]$. If M is the multiset whose cardinality is sought, we perform the following operations:

> **for** $i := 0$ **to** $L - 1$ **do** $BITMAP[i] := 0$;
>
> **for all** x **in** M **do**
>
> > **begin**
> >
> > *index* := $\rho(hash(x))$;
> >
> > **if** $BITMAP[index] = 0$ **then** $BITMAP[index] := 1$;
> >
> > **end**;

Thus $BITMAP[i]$ is equal to 1 iff after execution a pattern of the form $0^i 1$ has appeared amongst hashed values of records in M. Notice that *by construction*, vector $BITMAP$ only depends on the set of hashed values and not on the particular frequency with which such values may repeat themselves.

From the remarks concerning pattern probabilities, we should therefore expect, if n is the number of distinct elements in M that $BITMAP[0]$ is accessed approximately $n/2$ times, $BITMAP[1]$ approximately $n/4$ times .... Thus at the end of an execution, $BITMAP[i]$ will almost certainly be *zero* if $i \gg \log_2 n$ and *one* if $i \ll \log_2 n$ with a *fringe* of zeros and ones for $i \approx \log_2 n$. As an example, we took as M the on-line documentation corresponding to Volume 1 of the manual of the Unix system on one of our installations. M consists here of 26692 lines of which 16405 were distinct. Considering these lines as records and hashing them through standard multiplicative hashing over 24 bits ($L = 24$), we found the following $BITMAP$ vector:

$$111111111111001100000000$$

The leftmost value *zero* appears in position 12 and the rightmost value *one* in position 15 while $2^{14} = 16384$.

We propose to use the position of the leftmost zero in $BITMAP$ (ranks start at 0) as an indicator of $\log_2 n$. Let R be this quantity, we shall see that under the assumption that hashed values are uniformly distributed, the expected value of R is close to:

$$\mathbf{E}(R) \approx \log_2 \varphi n, \qquad \varphi = 0.77351 \cdots. \tag{1}$$

so that our intuition is justified. In fact the "correction factor" φ plays quite an important role in the design of the final algorithms we propose here. We shall also

prove that under reasonable probabilistic assumptions, the standard deviation of R is close to

$$\sigma(R) \approx 1.12 \qquad (2)$$

so that an estimate based on (1) will typically be one binary order of magnitude off the exact result, a fact that calls for more elaborate algorithms to be developed in Section 3.

Probability Distributions

We now proceed to justify rigorously the above claims (1) and (2) concerning the distribution of the value of parameter R in the basic counting procedure.

PROBABILISTIC MODEL. We let **B** denote the set of infinite binary strings. The model assumes that bits of elements of **B** are uniformly and independently distributed. Equivalently strings can be considered as real numbers over the interval $[0; 1]$, and the model assumes that the numbers are uniformly distributed over the interval. Functions bit and ρ are extended to **B** trivially. We denote by R_n the random variable defined over $\mathbf{B}^n$ (assuming independence) that is, the analogue of the parameter R above:

$R_n(x_1, x_2,..., x_n) = r$ iff (i) for all $0 \leqslant j < r$ there is an i such that $\rho(x_i) = j$ and (ii) for all i $\rho(x_i) \neq r$.

We also introduce the following notations concerning the probability distribution of R_n under the uniform model:

$$p_{n,k} = \Pr(R_n = k); \qquad q_{n,k} = \Pr(R_n \geqslant k)$$
$$\bar{R}_n = \mathbf{E}(R_n) = \sum_{k \geqslant 0} k p_{n,k}$$
$$\sigma_n^2 = \mathbf{E}((R_n - \bar{R}_n)^2) = \sum_{k \geqslant 0} k^2 p_{n,k} - \bar{R}_n^2,$$

and we let $v(n)$ denote the number of ones in the binary representation of n, so that for instance $v(13) = v((1101)_2) = 3$. We have

THEOREM 1. *The probability distribution of R_n is characterised by:*

$$q_{n,k} = \sum_{j=0}^{2^k} (-1)^{v(j)} \left(1 - \frac{j}{2^k}\right)^n.$$

Proof. For each integer $k \geqslant 0$, we define the following *events* (i.e., subsets of **B**):

$$E_k = \{x \mid \rho(x) = k\}; \qquad K_k = \{x \mid \rho(x) \geqslant k\}.$$

Thus, for each k, $E_0, E_1,..., E_{k-1}, K_k$ form a disjoint and complete set of events. When n elements are drawn from **B**, the formal polynomial:

$$P_k^{(n)} = (E_0 + E_1 + \cdots + E_{k-1} + K_k)^n \tag{3}$$

represents the set of all possible events in the following sense: if we expand (3) taken as a non-commutative polynomial in its indeterminates, interpreting the sums as (disjoint) unions of events and the products as successions of events (each monomial has degree n), we obtain a complete and disjoint representation of $\mathbf{B}^n$.

We are interested in obtaining from $P_k^{(n)}$ an expression for the polynomial $Q_k^{(n)}$ that represents in a similar fashion the succession of all events corresponding to $R_n \geqslant k$. Polynomial $Q_k^{(n)}$ is formed by a subset of the noncommutative monomials appearing in $P_k^{(n)}$.

Let us start with a few examples. If $k = 0$, we have: $P_0^{(n)} = (K_0)^n$ and $Q_0^{(n)} = P_0^{(n)}$. If $k = 1$,

$$P_1^{(n)} = (E_0 + K_1)^n, \qquad Q_1^{(n)} = (E_0 + K_1)^n - K_1^n,$$

since Q is obtained from P is this case by taking out from P the monomial K_1^n corresponding to the situation where all strings drawn have a ρ-value at least 1. For $k = 2$ now, we have

$$Q_2^{(n)} = (E_0 + E_1 + K_2)^n - (E_1 + K_2)^n - (E_0 + K_2)^n + K_2^n,$$

since we have to take out from P the cases where either ρ-value 1 or ρ-value 0 does not appear but in so doing, we have eliminated the case where all ρ-values are at least 2 (i.e., K_2) twice.

In general, for P a polynomial in the indeterminates $E_1, E_2,...,$ the polynomial Q formed with monomials of degree at most 1 in each of the indeterminates E_j can be obtained from P by the *inclusion–exclusion* type formula:

$$Q = P - \sum_i P[E_i \to 0] + \sum_{i \neq j} P[E_i, E_j \to 0] - \sum_{i \neq j \neq k} P[E_i, E_j, E_k \to 0] + \cdots, \tag{4}$$

where the notation $P[x, y \to 0]$ means the replacement of x, y by 0 in P. Thus $Q_k^{(n)}$ can in general be obtained by applying (4) to the expression of $P_k^{(n)}$ given by (3).

To evaluate the probabilities $q_{n,k}$, all we have to do is to take the measures μ of the events described by polynomial Q using the rules:

$$\mu(E_j) = \frac{1}{2^{j+1}}; \qquad \mu(K_k) = \frac{1}{2^k},$$

using additivity of measure μ over disjoint sets of events as well as the relation $\mu(A \cdot B) = \mu(A) \cdot \mu(B)$ since trials in **B** are assumed to be independent. On our pevious examples, we find in this way:

$$q_{n,0} = 1; \qquad q_{n,1} = 1 - (\tfrac{1}{2})^n; \qquad q_{n,2} = 1 - (\tfrac{3}{4})^n - (\tfrac{1}{2})^n + (\tfrac{1}{4})^n,$$

and in general:

$$q_{n,k} = 1 + \xi_1 + \xi_2 + \xi_3 + \cdots \tag{5}$$

where

$$\xi_t = (-1)^t \sum \left(1 - \frac{1}{2^{i_1}} - \frac{1}{2^{i_2}} - \cdots - \frac{1}{2^{i_t}} \right),$$

and the sum extends to all t-tuples of integers $i_1, i_2, \ldots$, of distinct integers in the interval $[1 \cdots k]$. Notice that by changing the summation indexes to $l_j = k - i_j$, ξ_t can be rewritten as:

$$\xi_t = (-1)^t \sum \left(1 - \frac{2^{l_1} + 2^{l_2} + \cdots 2^{l_t}}{2^k} \right)^n$$

where now the l_j are distinct integers over the interval $[0 \cdots k - 1]$. In other words, we have shown that

$$\xi_t = (-1)^t \sum_{\substack{\nu(j) = t \\ j \leqslant 2^k}} \left(1 - \frac{j}{2^k} \right)^n. \tag{6}$$

Using (6) inside (5) completes the proof of the theorem. ∎

We now turn to the derivation of asymptotic forms for these probabilities.

THEOREM 2. *The distribution of R_n satisfies the following estimates:*

(i) *If* $k < \log_2 n - 2 \log_2 \log n$, *then*

$$q_{n,k} = 1 - O(n e^{-\log^2 n});$$

(ii) *If* $k \leqslant \frac{3}{2} \log_2 n$, *then*

$$q_{n,k} = \prod_{j=0}^{\infty} (1 - e^{-2^j(n/2^k)}) + O\left(\frac{\log^6 n}{\sqrt{n}} \right)$$

$$= \sum_{j \geqslant 0} [(-1)^{\nu(j)} e^{-jn/2^k}] + O\left(\frac{\log^6 n}{\sqrt{n}} \right);$$

(iii) *If* $k \geqslant \frac{3}{2} \log_2 n + \delta$, *with* $\delta \geqslant 0$, *the tail of the distribution is exponential*:

$$q_{n,k} = O\left(\frac{2^{-\delta}}{\sqrt{n}} \right).$$

Proof. The main device here consists in using repeatedly the exponential approximation:

$$(1 - a)^n \approx e^{-na}.$$

inside the terms that form the expression of $q_{n,k}$:

$$q_{n,k} = \sum_{j=0}^{2^k} (-1)^{v(j)} \left(1 - \frac{j}{2^k}\right)^n. \tag{7}$$

We set

$$t(j, n, k) = \left(1 - \frac{j}{2^k}\right)^n:$$

(i)   The case when $k \leqslant \log_2 n - 2 \log_2 \log n$. Pulling out the 1 corresponding to the first term $(j = 0)$ in (7), and noticing that, as j increases, the terms $t(j, n, k)$ decrease, we find

$$1 - q_{n,k} \leqslant 2^k \left(1 - \frac{1}{2^k}\right)^n.$$

Since $2^k < n/\log^2 n$, we have $\log(1 - 1/2^k)^{-1} > \log^2 n/n$, and the above inequality becomes

$$1 - q_{n,k} \leqslant n e^{-\log^2 n},$$

as was to be established.

(ii)   The case when $k \leqslant \frac{3}{2} \log_2 n$. We set here $\varepsilon(n) = \log^2 n/n$. When $j > \varepsilon(n)2^k$, for k in the given range, $t(j, n, k)$ is $O(e^{-\log^2 n})$; since there are less than 2^k such terms, and $2^k = O(n^{3/2})$, we get

$$q_{n,k} = \sum_{j < \varepsilon(n)2^k} (-1)^{v(j)} t(j, n, k) + O(n^{3/2} e^{-\log^2 n}). \tag{8}$$

We let $q'_{n,k}$ denote the sum that appears in (8), and we define similarly

$$q''_{n,k} = \sum_{j < \varepsilon(n)2^k} (-1)^{v(j)} e^{-nj/2^k}.$$

For $j < \varepsilon(n)2^k$, we have

$$|t(j, n, k) - e^{-nj/2^k}| = O(e^{-nj/2^k}(e^{O(-nj^2/2^{2k})} - 1))$$
$$= O(n\varepsilon^2(n)).$$

so that, since q' and q'' comprise $2^k \varepsilon(n)$ terms,

$$|q'_{n,k} - q''_{n,k}| = O(n2^k \varepsilon^3(n)), \tag{9}$$

a quantity which is $O(\log^6 n/\sqrt{n})$. To derive the final expression, all we have to do is to "complete the sum" in $q''_{n,k}$; we set

$$q''_{n,k} = \sum_{k \geqslant 0} (-1)^{v(j)} e^{-nj/2^k} + E \tag{10}$$

where the error term E satisfies

$$|E| < \sum_{j > \varepsilon(n)2^k} e^{-nj/2^k}$$

$$= O\left(\frac{e^{-n\varepsilon(n)}}{1 - e^{-n/2^k}}\right) = O\left(\frac{2^k}{n} e^{-\log^2 n}\right) = O(n^{1/2} e^{-\log^2 n}). \tag{11}$$

Combining Eqs. (8), (9), (10), (11) therefore establishes the sum expression that appears in claim (ii) of the statement. To derive the product form, we appeal to the general identity

$$\sum_{j \geq 0} (-1)^{v(j)} q^j \equiv \prod_{m \geq 0} (1 - q^{2^m}).$$

(iii)   The case when $k = \frac{3}{2} \log_2 n + \delta$. We bound the probabilities $q_{n,k}$ by observing that since the p-value $k - 1$ is taken at least once:

$$\Pr(R_n \geq k) \leq 1 - (1 - 1/2^k)^n,$$
$$< 1 - \exp(-2 \cdot n/2^k). \tag{12}$$

In the range of values of k considered, the last expression is $O(n/2^k)$, which is itself of order $O(2^{-\delta}/\sqrt{n})$; thus the proof of part (iii) is now completed. ∎

For the sequel we introduce the real function:

$$\psi(x) = \prod_{j \geq 0} (1 - e^{-x2^j}) = \sum_{j \geq 0} (-1)^{v(j)} \exp(-jx). \tag{13}$$

Thus Theorem 2 expresses essentially the existence of a sort of *limiting distribution* for the probability distribution of R_n, as n gets large:

$$q_{n,k} \approx \psi\left(\frac{n}{2^k}\right); \qquad p_{n,k} \approx \psi\left(\frac{n}{2^k}\right) - \psi\left(\frac{n}{2^{k+1}}\right). \tag{14}$$

Table I describes the values of the probabilities compared to the approximation given by (14). It shows excellent agreement between the $q_{n,k}$'s and their approximations. It also reveals that the tail decreases sharply (actually a decrease faster than that of Theorem 2 may be established).

Asymptotic Analysis

From Theorem 2 follows that

LEMMA 1.   *The expectation $\bar{R}_n$ of R_n satisfies*

$$\bar{R}_n = \sum_{k \geq 1} k \left[\psi\left(\frac{n}{2^k}\right) - \psi\left(\frac{n}{2^{k+1}}\right) \right] + O\left(\frac{1}{n^{0.49}}\right), \tag{15}$$

TABLE I

Values of Exact Probabilities $(q_{n,k})$ and of the Approximations (9) (in Italics)

	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$	$k = 11$
100	0.0019	0.0439	0.200	0.3452	0.2767	0.1088	0.0212	0.0020
	0.0016	*0.0417*	*0.1985*	*0.3476*	*0.2789*	*0.1087*	*0.0209*	*0.0020*
	$k = 7$	$k = 8$	$k = 9$	$k = 10$	$k = 11$	$k = 12$	$k = 13$	$k = 14$
1000	0.0004	0.0201	0.1389	0.3166	0.3216	0.1586	0.0388	0.0047
	0.0004	*0.0200*	*0.1387*	*0.3167*	*0.3219*	*0.1586*	*0.0388*	*0.0047*
	$k = 10$	$k = 11$	$k = 12$	$k = 13$	$k = 14$	$k = 15$	$k = 16$	$k = 17$
10000	0.0001	0.0076	0.0863	0.2673	0.3469	0.2150	0.0659	0.0101
	0.0001	*0.0076*	*0.0863*	*0.2673*	*0.3469*	*0.2150*	*0.0659*	*0.0101*

Note. $n = 100 \approx 2^{6.64}$, $n = 1000 \approx 9^{9.96}$, and $n = 10000 \approx 2^{13.28}$.

Thus the problem of estimating $\bar{R}_n$ asymptotically reduces to that of estimating the sum in (15), i.e., the function

$$F(x) = \sum_{k \geqslant 1} k \left[\psi\left(\frac{x}{2^k}\right) - \psi\left(\frac{x}{2^{k+1}}\right) \right], \tag{16}$$

for large x. To that purpose we appeal to Mellin transform techniques whose introduction in the context of analysis of algorithms is due to De Bruijn (see [4, pp. 131 *et seq.*]). The Mellin transform of a function $f(x)$ defined for $x \geqslant 0$, x real, is by definition the complex function $f^*(s)$ given by

$$f^*(s) \equiv M[f(x); s] = \int_0^\infty f(x) x^{s-1} \, dx. \tag{17}$$

We succinctly recall the salient properties of the Mellin transform, referring the reader to [1] for precise statements. The Mellin transform of a function f is defined in a strip of the complex plane that is determined by the asymptotic behaviours of f at 0 and ∞. It satisfies the important functional property

$$M[f(ax); s] = a^{-s} f^*(s). \tag{18}$$

Finally there is a complex inversion formula

$$f(x) = \frac{1}{2i\pi} \int_{c-i\infty}^{c+i\infty} f^*(s) x^{-s} \, ds \tag{19}$$

where c is chosen in the strip where the integral in (17) is absolutely convergent.

The interest of the inversion formula is that, in many cases, it can be evaluated by means of the *residue* theorem, each residue corresponding to a *term in the asymptotic expansion* of f.

LEMMA 2. *The Mellin transform of* $F(x)$ *is for* $-1 < \text{Re}(s) < 0$:

$$F^*(s) = \frac{2^s}{1 - 2^s} N(s)\, \Gamma(s),$$

where $\Gamma(s)$ *is the Euler Gamma function and* $N(s)$ *is an entire function that is the analytic continuation of the function defined for* $\text{Re}(s) > 1$ *by*

$$N(s) = \sum_{j \geq 1} \frac{(-1)^{\nu(j)}}{j^s}.$$

Proof. Let $\psi_1(x) \equiv \psi(x) - 1$. The transform of ψ_1 is for $\text{Re}(s) > 1$:

$$
\begin{aligned}
\psi_1^*(s) &= \sum_{j \geq 1} (-1)^{\nu(j)} j^{-s} \Gamma(s) \\
&= N(s)\, \Gamma(s),
\end{aligned}
\tag{20}
$$

as follows from the basic functional property (18), and the fact that the transform of $\exp(-x)$ is the Gamma function $\Gamma(s)$. Similarly, for $\psi_2(x) = \psi(x) - \psi(x/2)$ and $\text{Re}(s) > 1$, we get

$$\psi_2^*(s) = M[\psi(x) - \psi(x/2); s] = \psi_1^*(s)(1 - 2^s). \tag{21}$$

Since $\psi(x) - \psi(x/2)$ is exponentially small both at 0 and ∞, the transform ψ_2^* is actually analytic for all complex s; since:

$$N(s) = \frac{\psi_2^*(s)}{\Gamma(s)(1 - 2^s)}. \tag{22}$$

we find that $N(s)$ is analytic for all s except possibly for the points $s = 2ik\pi/\log 2$, where the denominator of (22) vanishes. However, direct calculations in Lemma 3 below show that $N(s)$ is analytic for $\text{Re}(s) > -1$, so that $N(s)$ is analytic everywhere.

Now, using again the basic functional property,

$$
\begin{aligned}
F^*(s) &= \psi_2^*(s) \sum_{k \geq 1} k 2^{ks} \\
&= \psi_2^*(s) \frac{2^s}{(1 - 2^s)^2},
\end{aligned}
\tag{23}
$$

where (23) is valid for $\text{Re}(s) < 0$. Putting together (20), (21), (22), (23) establishes the claim of the lemma.

We now need to establish some more constructive properties of $N(s)$ for $\mathrm{Re}(s) < 0$, establishing in passing the analytic continuation property of $N(s)$ used in the proof of Lemma 2.

LEMMA 3. *The function $N(s)$ satisfies $N(0) = -1$. Furthermore, for $s = \sigma + it$ and $\sigma > -0.99$, it satisfies*

$$N(s) = O(1 + |s|^3).$$

Proof. Terms in the definition of $N(s)$ may be grouped 4 by 4; using the property

$$v(4j) = v(j); \; v(4j+1) = v(4j+2) = 1 + v(j); \; v(4j+3) = 2 + v(j),$$

we find

$$
N(s) = -1^{-s} - 2^{-s} + 3^{-s} \\
+ \sum_{j \geqslant 1} \frac{(-1)^{v(j)}}{(4j)^s} \left[1 - \frac{1}{\left(1 + \frac{1}{4j}\right)^s} - \frac{1}{\left(1 + \frac{2}{4j}\right)^s} + \frac{1}{\left(1 + \frac{3}{4j}\right)^s} \right]. \quad (24)
$$

We observe that the general term in the above sum is $O(j^{-\sigma-2})$ as j gets large. This confirms that $N(s)$ is defined and analytic when $\sigma > -1$. To obtain the bounds on $N(s)$, we split the sum (24): the terms such that $j < |s|^2$ contribute at most $O(1 + |s|^4)$ to the sum; and since

$$1 - (1 + u)^{-s} - (1 + 2u)^{-s} + (1 + 3u)^{-s} = O(|s|^2 u^2)$$

uniformly in s and u when $u < 1/|s|^2$, we find that the contribution of terms such that $j > |s|^2$ is

$$O\left(|s|^2 \sum_{j > |s|^2} j^{-\sigma-2}\right) = O(|s|^2),$$

uniformly in s when $\sigma > -0.99$, say. Finally substituting $s = 0$ in (24) gives $N(0) = -1$. ∎

We can now come back to the asymptotic study of $F(x)$ and hence of $\bar{R}_n$ using the inversion formula (19).

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

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

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

Proof. By Lemma 1, the problem reduces to obtaining an asymptotic expansion of $F(x)$ as $x \to \infty$ up to $o(1)$ terms. The principle consists in evaluating the complex integral of the form (19) by residues. From the inversion theorem for Mellin transforms, we have

$$F(x) = \frac{1}{2i\pi} \int_{-1/2 + i\infty}^{-1/2 + i\infty} F^*(s) \, x^{-s} \, ds. \tag{25}$$

We consider for k a positive integer the rectangle contour Γ_k defined by its corner points (and traversed in that order)

$$\Gamma_k = [-1/2 - i(2k+1)\,\pi/\log 2; \; -1/2 + i(2k+1)\,\pi/\log 2;$$
$$1 - i(2k+1)\pi/\log 2; \; 1 + i(2k+1)\pi/\log 2].$$

By Cauchy's residue theorem, we have

$$\frac{1}{2i\pi} \int_{\Gamma_k} F^*(s) \, x^{-s} \, ds = - \sum_{s \, in \, \Gamma_k} \operatorname{Res}(F^*(s)x^{-s}).$$

For fixed x, as k gets large, the integral along the segment $[-1/2 - i(2k+1)\,\pi/\log 2; \; -1/2 + i(2k+1)\,\pi/\log 2]$ tends to $F(x)$ by (25). From Lemma 3 and the exponential decrease of $\Gamma(s)$ towards $i\infty$, the integrals along $[-1/2 + i(2k+1)\,\pi/\log 2; \quad 1 - i(2k+1)\pi/\log 2]$ and $[1 + i(2k+1)\pi/\log 2; -1/2 - i(2k+1)\pi/\log 2]$ tend to zero exponentially fast (as functions of m). As to the integral along $[1 - i(2k+1)\pi/\log 2; \; 1 + i(2k+1)\pi/\log 2]$, it stays bounded in absolute value by

$$\frac{1}{2\pi} \int_{-\infty}^{+\infty} |F^*(1+it)| \, x^{-1} \, dt < \frac{K}{x},$$

for some constant K. (Again the exponential decrease of $\Gamma(s)$ guarantees convergence of the above integral.) We have thus found that, by letting $m \to \infty$:

$$F(x) = - \sum_{\operatorname{Re}(s) = 0} \operatorname{Res}(F^*(s)x^{-s}) + O\left(\frac{1}{x}\right). \tag{26}$$

(The sum of residues is also absolutely convergent because of the decrease of $\Gamma(s)$ towards $i\infty$.) It only remains to evaluate the residues in (26). $F^*(s)$ has a double pole at $s = 0$ and simple poles at each $\chi_k = 2ik\pi/\log 2$, with k an integer different from 0, and we find easily

$$-\operatorname{Res}(F^*(s)x^{-s}; s = 0) = \log_2 x + \frac{\gamma}{\log 2} + \frac{N'(0)}{\log 2} - \frac{1}{2},$$

which we may rewrite as $\log_2 \varphi x$, and

$$-\text{Res}(F^*(s) \, x^{-s}; s = \chi_k) = (1/\log 2) \, \Gamma(\chi_k) \, N(\chi_k) \, x^{-\chi_k},$$

which is of the form $p_k \, e^{-2ik\pi \log_2 x}$.

Thus summing the residues, and using (26), we find the announced asymptotic form for $F(x)$ (and hence $\bar{R}_n$), with $P(u)$ given by

$$P(u) = \sum_{k \in \mathbf{Z}/\{0\}} p_k \, e^{-2ik\pi u}.$$

The gory details of the bound on the amplitude of $P(u)$ are left for the Appendix. ∎

We can evaluate the standard deviation of R_n in a similar fashion. Let S_n be the second moment of R_n: $S_n = E(R_n^2)$. As before, S_n is approximated by the function $G(n)$ where

$$G(x) = \sum_{k \geqslant 1} k^2 \left[\psi \left(\frac{x}{2^k} \right) - \psi \left(\frac{x}{2^{k+1}} \right) \right],$$

whose transform is found to be for $\text{Re}(s) < 0$

$$G^*(s) = \frac{2^s(1 + 2^s)}{(1 - 2^s)^2} \, \Gamma(s) \, N(s),$$

which now has a triple pole at $s = 0$. Computing $G(z)$ is done from $G^*(s)$ via the inversion theorem followed by residue calculations, and one finds:

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

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

where $\sigma_\infty = 1.12127...$ 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

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

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

3. PROBABILISTIC COUNTING ALGORITHMS

We have seen in the previous section that the result R of the *COUNT* procedure has an average close to $\log_2 \varphi n$, with a standard deviation close to 1.12. Actually the values of

$$\lambda(n) = (1/\varphi) 2^{R_n}$$

are amazingly close to n as the following instances show:

$$\lambda(10) = 10.502; \qquad \lambda(100) = 100.4997; \qquad \lambda(1000) = 1000.502.$$

This observation justifies the hope of obtaining very good estimates on n from the observation of parameter R, using the correction factor φ. However, the dispersion of results corresponds to a typical error of 1 binary order of magnitude which is certainly too high for many applications.

The simplest idea to remedy this situation consists in using a set $\mathbf{H}$ of m hashing functions, where m is a *design parameter* and computing m different *BITMAP* vectors. In this way, we obtain m estimates $R^{\langle 1 \rangle}$, $R^{\langle 2 \rangle}$,..., $R^{\langle m \rangle}$. One then considers the average

$$A = \frac{R^{\langle 1 \rangle} + R^{\langle 2 \rangle} + \cdots + R^{\langle m \rangle}}{m}. \tag{27}$$

When n distinct elements are present in the file, the random variable A has expectation and standard deviation that satisfy

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

Thus we may expect 2^A to provide an estimate of n with a typical error (measured by the standard deviation of the estimates) of relative value $\approx K/\sqrt{m}$.

Such an algorithm using direct averaging has indeed provably good performances (with an expected relative error of about 10% if $m = 64$) but it has the disadvantage of requiring the calculation of a number of hashing functions, so that the CPU cost per element scanned gets essentially multiplied by a factor of m.

It turns out that an effect very similar to straight averaging may be achieved by a device that we call *stochastic averaging*. The idea consists in using the hashing function in order to distribute each record into one of m lots, computing $\alpha = h(x) \bmod m$. We update only the corresponding *BITMAP* vector of address α with the "rest" of the information contained in $h(x)$, namely $h(x) \operatorname{div} m \equiv \lfloor h(x)/m \rfloor$. At the end, we determine as before the $R^{\langle j \rangle}$'s and compute their average A by (27). Hoping for the distirbution of records into lots to be even enough, we may thus expect that about n/m elements fall into each lot so that $(1/\varphi)2^A$ should be a reasonable approximation of n/m.

The corresponding algorithm is called *Probabilistic Counting with Stochastic Averaging*, or PCSA for short. It is described in Fig. 1. We claim that its cost per element scanned is handly distinguishable from that of the *COUNT* procedure and its relative accuracy improves with m roughly as $0.78/\sqrt{m}$. In the sequel, we shall call *standard error* the quotient of the standard deviation of an estimate of n by the value of n; this quantity is thus a precise indication of the *expected relative accuracy* of an algorithm estimating n. Neglecting periodic fluctuations of extremely small amplitude (less than 10^{-5}), we shall call the *bias* of an algorithm the ratio between the estimate of n and the exact values of n for large n. Standard error and bias of

```
program PCSA;

const nmap = 64;  {with nmap = 64, accuracy is typically 10%}
    {nmap corresponds to variable m in the analysis}
    φ = 0.77351 {the magic constant};  maxlength = 32;
    {with maxlength =32 (==L), one can count up to 10⁸.}

var M: multiset of data of type records;
        x: records; hashedx, index, α, R, S, Ξ: integer;
        BITMAPS: array [0..nmap-1;0..maxlength-1] of integer;

function getelement(var x:records);
{reads an element x of type records from file M}
function hash(x:records):integer;
{hashes a record x into an integer over scalar range [0..2^maxlength −1]}
function ρ(y:integer):integer;
{returns the position of the first 1−bit in y; ranks start at 0.}

begin
while not eof(M) do
    begin
    getelement(x); hashedx:=hash(x);
    α:=hashedx mod nmap; index:=ρ(hashedx div nmap);
    if BITMAP[α,index]=0 then BITMAP[α,index]:=1;
    end;
S = 0;
for i:=0 to nmap-1 do
    begin
    R:=0; while (BITMAP[i,R]=1) and (R<maxlength) do R:=R+1; S:=S+R;
    end;
Ξ:=trunc(nmap /φ *2 **(S/ nmap ));
{Result Ξ of the PCSA programme that estimates n}
end.
```

FIG. 1.   Probabilistic counting with stochastic averaging (PCSA).

TABLE II

Bias and Standard Error of PCSA for Several Values of
m, the Number of *BITMAP* Vectors Used

m	Bias	% Standard error
2	1.1662	61.0
4	1.0792	40.9
8	1.0386	28.2
16	1.0191	19.6
32	1.0095	13.8
64	1.0047	9.7
128	1.0023	6.8
256	1.0011	4.8
512	1.0005	3.4
1024	1.0003	2.4

algorithm PCSA for various values of the design parameter m are displayed in Table II.

In the remainder of this section, we are going to justify these claims rigorously and in particular show how the estimates of Table II are deduced.

We let Ξ denote the random variable computed by PCSA with m *BITMAP*s and let Ξ_n denote this random variable when n distinct elements are present in the file; we denote by $\mathbf{E}[\Xi_n]$ the average value of Ξ_n and $\sigma(\Xi_n)$ the standard deviation of Ξ_n. We propose to establish:

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}.

THEOREM 5. *Using the notation $u(n) \simeq v(n)$ to express the property*

$$\exists n_0\ \forall n > n_0\ |u(n) - v(n)| < 10^{-5}$$

one has the following characterisations of the bias and standard error of algorithm PCSA:

$$\frac{\mathbf{E}[\Xi_n]}{n} \simeq (1 + \varepsilon(m))$$

$$\frac{\sigma[\Xi_n]}{n} \simeq \eta(m),$$

where quantities $\varepsilon(m)$ and $\eta(m)$ satisfy as m gets large:

$$\varepsilon(m) \sim \lambda/2m$$

$$\eta(m) \sim \lambda^{1/2}/\sqrt{m},$$

where

$$\lambda = \frac{\pi^2}{12} - \frac{\gamma^2}{2} - N'(0)^2 - N''(0) + \frac{\log^2 2}{12}.$$

---

[2] The error terms in Theorem 4 and the n_0 in Theorem 5 are *not* uniform in m.

The Analysis of Algorithm PCSA

We now proceed with the proof of Theorem 4. We start with an estimate of $E[\beta^{R_n}]$ for $1 \leqslant \beta \leqslant 2$ that is needed throughout the rest of this section and prove

LEMMA 4. *Setting $\beta = 2^{1/q}$, with $q \geqslant 1$, one has for fixed q*

$$E[\beta^{R_n}] = n^{1/q}(d_q + P_q(\log_2 n)) + o(n^{1/q}),$$

where

$$d_q = -\frac{1}{\log 2}(1 - 2^{-1/q}) N\left(-\frac{1}{q}\right)\Gamma\left(-\frac{1}{q}\right)$$

and P_q is a periodic function of amplitude less than 10^{-5}.

Proof. (i) We start with a strenghening of bounds on the tail of the distribution of R_n. Consider the probability $\Pr[R_n \geqslant k]$ where $k = \frac{5}{4}\log_2 n + \delta$, with $\delta > 0$. When $R_n \geqslant k$, positions $(k-1)$ and $(k-2)$ of *BITMAP* are set to 1, an event that has probability

$$1 - \left(1 - \frac{1}{2^k}\right)^n - \left(1 - \frac{1}{2^{k-1}}\right)^n + \left(1 - \frac{1}{2^k} - \frac{1}{2^{k-1}}\right)^n$$

a quantity which is

$$1 - e^{-n/2^k + O(n/2^{2k})} - e^{-n/2^{k-1} + O(n/2^{2k})} + e^{-3n/2^{k-1} + O(n/2^{2k})}$$

or $O(n/2^{2k})$, which in the given range of values of k is $O(n^{-3/2}4^{-\delta})$. Thus

$$\sum_{k > (5/4)\log_2 n} 2^k p_{n,k} = O\left(n^{5/4 - 3/2} \sum_{\delta \geqslant 0} 4^{-\delta} 2^\delta\right) = O(n^{-1/4}), \tag{28}$$

and the same bound applies if 2 is replaced by β in the above sum.

We now consider the error that comes from the replacement of the $p_{n,k}$ by their asymptotic equivalent for "small" k. From the bounds of Theorem 2, one finds

$$\sum_{k \leqslant (5/4)\log_2 n} \beta^k \left[p_{n,k} - \psi\left(\frac{n}{2^k}\right) + \psi\left(\frac{n}{2^{k+1}}\right)\right] = O\left(\frac{n^{5/4q}}{n^{0.49}}\right) = O(n^{0.76/q}), \tag{29}$$

a quantity which is $\ll n^{1/q}$. Thus completing the sum and defining the function

$$H(x) = \sum_{k \geqslant 1} \beta^k \left[\psi\left(\frac{x}{2^k}\right) - \psi\left(\frac{x}{2^{k+1}}\right)\right],$$

we have from (28), (29):

$$E[\beta^{R_n}] = H(n) + O(n^{0.76/q}).$$

The asymptotic behaviour of H is determined by Mellin transform techniques as before; the transform of function H is

$$H^*(s) = \frac{\beta 2^s}{1 - \beta 2^s} \, \Gamma(s) \, N(s).$$

H^* has poles at $s = -1/q + 2ik\pi/\log 2$ and we find the claim of the lemma, using the inversion theorem with

$$d_q = -\operatorname{Res}(H^*(s); s = -1/q). \quad \blacksquare$$

The next step in the proof of Theorem 4 is to establish that algorithm PCSA behaves asymptotically as though the n elements were perfectly distributed in m groups.

LEMMA 5. *If n elements are distributed into m cells (m fixed), where the probability that any element goes to a given cell has probability $1/m$, then the probability that at least one of the cells has a number of elements N satisfying*

$$|N - n/m| > \sqrt{n} \log n$$

is $O(e^{-h \log^2 n})$ for some constant $h > 0$.

Proof. Set $p = 1/m$, $q = 1 - 1/m$; let N_1 be the number of elements that fall into cell 1. N_1 obeys a binomial distribution

$$\Pr(N_1 = k) = \binom{n}{k} p^k q^{n-k}, \tag{30}$$

and taking logarithms of (30), for $k = pn + \delta$ and $\delta \ll n$, one finds

$$\Pr(N_1 = pn + \delta) = \exp\left(-\frac{\delta^2 + O(\delta)}{2npq} + O\left(\frac{\delta^3}{n^2}\right)\right).$$

If $\delta = \sqrt{n} \log n$, the probability (30) is exponentially small. We conclude the proof by observing that the binomial distribution is unimodal and

$$\Pr\left[\bigcup_{1 \leqslant j \leqslant m} \left| N_j - \frac{n}{m} \right| > \sqrt{n} \log n \right] < m \Pr\left[\left| N_1 - \frac{n}{m} \right| > \sqrt{n} \log n \right]. \quad \blacksquare$$

We can now conclude the proof of the first part of Theorem 4. Let S denote the sum $R^{\langle 1 \rangle} + R^{\langle 2 \rangle} + \cdots + R^{\langle m \rangle}$. We have

$$\Pr(S = k) = \sum_{\substack{n_1 + n_2 + \cdots + n_m = n \\ k_1 + k_2 + \cdots + k_m = k}} \frac{1}{m^n} \binom{n}{n_1, n_2, \ldots, n_m} p_{n_1, k_1} \, p_{n_2, k_2} \cdots p_{n_m, k_m}. \tag{31}$$

Thus

$$\mathbf{E}(2^{S/m}) = \sum_{n_1 + n_2 + \cdots + n_m = n} \frac{1}{m^n} \binom{n}{n_1, n_2, \ldots, n_m} \mathbf{E}(2^{R_{n_1}/m}) \mathbf{E}(2^{R_{n_2}/m}) \cdots \mathbf{E}(2^{R_{n_m}/m}). \qquad (32)$$

Call E the quantity (32), and E_C the sum of the terms in (32) such that for all $j, 1 \leqslant j \leqslant m$:

$$\left| n_j - \frac{n}{m} \right| < \sqrt{n} \log n.$$

From Lemmas 4, 5, $E - E_C$ is $O(ne^{-h\log^2 n})$. As to the central contribution E_C it is bounded by

$$(\mathbf{E}[2^{(1/m)R_{n/m} - \sqrt{n}\log n}])^m < E_C < (\mathbf{E}[2^{(1/m)R_{n/m} + \sqrt{n}\log n}])^m,$$

so that finally

$$\mathbf{E}[2^{S/m}] = (\mathbf{E}[2^{(1/m)R_{n/m}}])^m + o(n). \qquad (33)$$

or

$$\mathbf{E}(\Xi_n) = \frac{m}{\varphi} (\mathbf{E}[2^{(1/m)R_{n/m}}])^m + o(n). \qquad (34)$$

Equation (34) combined with Lemma 5 is sufficient to establish the estimates on Ξ_n from Theorem 4, provided we check that the amplitudes of the periodic fluctuations do not grow with m, a fact that can be proved using the methods described in the Appendix.

Estimates on the second moment of Ξ_n are derived in exactly the same way through the equality

$$\mathbf{E}(\Xi_n^2) = \frac{m^2}{\varphi^2} (E[2^{(2/m)R_{n/m}}])^m + o(n^2). \qquad (35)$$

Dependence of Results on the Number of BITMAPs

We finally conclude with an indication of the (easy) proof of Theorem 5. From Theorem 4, all we need is to determine the asymptotic behaviour of the quantities

$$\alpha(m) = \frac{1}{\varphi} \left[\frac{1}{\log 2} N\left(-\frac{1}{m} \right) \Gamma\left(-\frac{1}{m} \right) (1 - 2^{-1/m}) \right]^m, \qquad (36)$$

$$\beta(m) = \frac{1}{\varphi^2} \left[\frac{2}{\log 2} N\left(-\frac{2}{m} \right) \Gamma\left(-\frac{2}{m} \right) (2 - 2^{-2/m}) \right]^m, \qquad (37)$$

$$\gamma(m) = (\beta(m) - \alpha^2(m))^{1/2}, \qquad (38)$$

as m gets large since we neglect the effect of the small periodic fluctuations. This is achieved by performing standard (but tedious) asymptotic expansions of (36), (37), (38) for large m. (This task as been carried out with the help of the MACSYMA system for symbolic computations.) We find that the bias and standard error are *for all values of m* closely approximated by the formulae

$$\text{bias:} \quad 1 + 0.31/m \tag{39}$$

$$\text{standard error:} \quad 0.78/\sqrt{m}. \tag{40}$$

4. IMPLEMENTATION ISSUES

There are three factors to be taken into account when applying algorithm PCSA:

(i)   The choice of the hashing function.

(ii)   The choice of the length of the *BITMAP*-vectors, L.

(iii)   The number, nmap, of *BITMAP* used (corresponding to quantity m in our analyses).

Also corrections of two types may be introduced:

(iv)   Corrections to the systematic bias of Table II.

(v)   Corrections for initial nonlinearities of the algorithm.

We briefly proceed to discuss these issues here.

1. *Hashing functions.* Simulations on textual files (see below) ranging in size from a few kilobytes to about 1 megabyte indicate that standard multiplicative hashing leads to performances that do not depart in any detectable way from those predicted by the uniform model of Sections 2, 3. There, a record $x = (x_0, x_1, ..., x_p)$ formed of ASCII characters is hashed into

$$h(x) = \left(M + N \sum_{j=0}^{p} \text{ord}(x_j)\, 128^j \right) \bmod 2^L,$$

with $\text{ord}(\kappa)$ denoting the (standard ASCII) rank of character κ. This good agreement between theoretically predicted and practically observed performances is in accordance with empirical studies concerning standard hashing techniques and conducted on large industrial files by Lum *et al.* [5].

2. *Length of the BITMAP vector.* Since the probability ditribution of the R-parameter has a very steep distribution, it suffices to select L in such a way that

$$L > \log_2(n/\text{nmap}) + 4. \tag{41}$$

Thus, as already pointed out, with nmap $= 64$, taking $L = 16$ makes it possible to safely count cardinalities of files up to $n \sim 10^5$, and $L = 24$ can be used for car-

dinalities well beyond 10^7. The probabilities of obtaining underestimates because of such truncations (the probabilistic model assumes L to be infinite) can be computed from our previous results and when (41) is satisfied, the error introduced is below $5 \cdot 10^{-3}$.

3. *Number of BITMAPS.* The expected relative accuracy of the algorithm or *standard error* is by Theorems 4, 5 inversely proportional to $\sqrt{m}$, being closely approximated by

$$0.78/\sqrt{m}.$$

Thus nmap $= 64$ leads to a standard error of about 10%, and with nmap $= 256$, this error decreases to about 5% (see Table II).

4. *Bias.* The *bias* of algorithm PCSA as presented in Table II is negligible compared to the standard error as soon as nmap exceeds 32. If smaller values of nmap are to be used, it can be corrected using the results of Theorems 4, 5. For a practical use of the algorithm, it suffices to use the estimates of Theorem 5, which one achieves by changing the last instruction of the programme to

$$\varXi := \operatorname{trunc}(\text{nmap}/(\varphi*(1 + 0.31/\text{nmap}))*2**(S/\text{nmap})).$$

In so doing, we obtain an algorithm which apart from the small periodic fluctuations of amplitude less than 10^{-4} is an *asymptotically unbiased estimator* of cardinalities n.

5. *Initial non-linearities.* The asymptotic estimates which form the basis of the algorithm are extremely close to the actual average values as soon as n/nmap exceeds 10–20. If very small cardinalities were to be estimated, then based on the characterisation of probability distributions, *corrections* could be computed and introduced in the algorithm. (These corrections would be based on calculation of exact average values from our formulae instead of using the asymptotic estimates).

Simulations

We have conducted fairly extensive simulations of algorithm PCSA applied to textual data. The files called man_1, man_2,..., man_8 correspond to chapters of the on-line documantation available on one of our systems, and the versions $\text{man}_1 w$, $\text{man}_2 w$,... correspond to the files obtained from the preceding ones by segmentation into 5 character blocks. Standard multiplicative hashing was used as described by Eq. (41). We counted in each case the number of different records and compared with corresponding values estimated by algorithm PCSA (here, a record is a line of text for man_1,... and a 5 letter block for $\text{man}_1 w$,...). Some sample runs are reported in Table III, and they show good agreement between our estimates and actual values. The files are mixtures of text in English, names of commands and typesetting commands.

We have also taken these 16 files, and have subjected them to algorithm PCSA, varying the constants M and N in (41). This provides empirical values of the bias

TABLE III

Sample Executions of Algorithm PCSA on 6 Files with the Same Multiplicative Hashing Function

File	Card.	8	16	32	64	128	256
man 1	16405	17811	16322	14977	15982	16690	17056
		1.08	*0.99*	*0.91*	*0.97*	*1.01*	*1.03*
man 1.w	38846	40145	40566	40145	43290	41230	42592
		0.96	*1.01*	*0.96*	*1.07*	*1.02*	*1.06*
man 2	3149	2427	2887	3015	3015	2840	2982
		0.77	*0.91*	*0.95*	*0.95*	*0.90*	*0.94*
man 2.w	10560	10590	9711	9100	9100	10032	10734
		1.00	*0.91*	*0.86*	*0.86*	*0.95*	*1.01*
man 8	3075	4452	3744	3360	3252	3097	3106
		1.44	*1.21*	*1.09*	*1.05*	*1.00*	*1.01*
man 8.w	11334	10590	10590	10363	10705	10999	10676
		0.93	*0.93*	*0.91*	*0.94*	*0.97*	*0.94*

Note. The figure displays the file name, the exact cardinality, the estimated cardinality for nmap = 8, 16, 32, 64, 128, 256, and the ratio of estimated cardinalities to exact cardinalities (in italics).

and standard error of PCSA (averaging over 10 simulations × 16 files) that again appear to be in amazingly good agreement with the theoretical predictions. Such results are reported in Table IV and should be compared with Table II. (The correction for small values of nmap described above has been inserted into the algorithm PCSA of Fig. 1.)

Applications to Distributed Computing

Assume a fine F is partitioned into subfiles $F_1, F_2,..., F_s$, where the F_i and F_j need not be disjoint. Such a situation occurs routinely in the context of *distributed data bases.*

TABLE IV

Empirical Values of Bias and Standard Error Based on
160 Simulations

m	Bias	% Standard error
8	1.0169	31.92
16	1.0104	19.63
32	0.9798	12.98
64	0.9961	9.67
128	1.0035	6.68
256	1.0073	4.65

Note. Ten different hashing functions applied to the 16 files $man_1,...,$ $man_8.w$.

Then the global cardinality of file F may be determined as follows:

Process separately each of the s subfiles by algorithm PCSA. This gives rise to s BITMAP vectors, $BITMAP_1$,.... Each of the s processors sends its result to a central processor that computes the logical or of the s BITMAPs. The resulting BITMAP vector is then used to construct the estimate of n.

It is rather remarkable that the accuracy of the estimate is, by construction, not affected at all by the way records are spread amongst subfiles. The number of messages exchanged is small (being $O(s)$), and the algorithm results in a *net speed-up* by a factor of s.

Scrolling

The matrix of *BITMAP* vectors has a rather specific form: it starts with rows of *ones* followed by a fringe of rows consisting of mixed *zeros* and *ones* and followed by rows all *zeros*. This suggests naturally a more compact encoding of the bitmap that may be quite useful for distributed applications since it then minimises the sizes of messages exchanged by processors. The idea is to indicate the left boundary of the fringe, followed by a standard encoding of the fringe itself. For instance if the *BITMAP* matrix is

$$
\begin{array}{cccc|ccccc|cccc}
1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$

then, one only needs to represent the leftmost boundary of the fringe here 4), and the binary words 10100, 11000, 01011, 11010.

This technique amounts to keeping only a small window of the *BITMAP* matrix and scrolling it is necessary. For practical pruposes, a window of size 8 should suffice, so that the storage requirement of this version of PCSA becomes close to $\frac{1}{8}\log_2 n + $ nmap bytes.

Deletions

If instead of keeping only bits to record the occurrences of patterns of the form 0^k1, one also keeps the counts of such occurrences, one obtains an algorithm that can maintain running estimates of cardinalities of files subjected to arbitrary sequences of insertions and deletions. The price to be paid is however a somewhat increased storage cost.

5. CONCLUSION

Probabilistic counting techniques presented here are particular algorithmic solutions to be problem of estimating the cardinality of a multiset. It is quite clear 24

that other observable regularities on hashed values of records could have been used, in conjunction with direct or stochastic averaging. We mention is passing:

—the rank of the rightmost one in *BITMAP*: this parameter has a flatter distribution that results in an appreciably less accurate algorithm (in terms of standard error);

—the binary logarithm of the minimal hashed value encountered (hashed values being considered are real $[0; 1]$ numbers) provides an approximation to $\log_2 1/n$, but the resulting algorithm appears to be slightly less accurate than PCSA.

The common feature of all such algorithms is to estimate the cardinality n of a multiset in real time, using auxiliary storage $O(m \log_2 n)$ with a relative accuracy of the form:

$$\alpha/\sqrt{m}.$$

It might be of interest to determine whether appreciably better *storage/accuracy trade-offs* can be achieved (or to prove that this is not possible from an information-theoretic standpoint).

For practical purposes, algorithm PCSA is quite satisfactory. It consumes only a few operations per element scanned (may be 20 or 30 assembly language instructions), has good accuracy described at length in the previous sections, and may be used to gather statistics on files *on the fly* (therefore eliminating the additonal cost of disk accesses). On a VAX 11/780 running Berkeley Unix, a non-optimised version in Pascal used for our tests is already typically twice as fast as the standard system sorting routine. A version of the algorithm has been implemented at IBM San Jose in the context of the System R^* Project.

APPENDIX: THE AMPLITUDE OF PERIODIC FLUCTUATIONS

The purpose of this Appendix is to show how the fluctuations, in the form of Fourier series, that appear in Theorems 3, 4, 5 can be precisely bounded. Notice that the problem reduces to showing that the Fourier coefficients have sufficiently small values.

All these Fourier coefficients are values of functions of the form:

$$\Gamma(s) \, N(s) \, \omega(s),$$

with $\omega(s)$ a "well-behaved" function, taken at points $\chi_k = \sigma + 2ik\pi/\log 2$ and k is a non-zero integer. Quantity σ depends on the particular problem considered: $\sigma = 0$ in Theorem 3, $\sigma = -1/m$ in Theorems 4, 5.

We shall only give a proof in the case of Theorem 3.A, the other proofs being entirely similar. We thus need to find bounds for the Fourier series:

$$P(u) = \sum_{k \in \mathbf{Z}/\{0\}} p_k \, e^{-2ik\pi u}$$

with

$$p_k = \frac{1}{\log 2} \Gamma\left(\frac{2ik\pi}{\log 2}\right) N\left(\frac{2ik\pi}{\log 2}\right).$$

The behaviour of the gamma function along the imaginary axis is known:

$$|\Gamma(it)| = \sqrt{\pi/t \sinh \pi t}$$

so that it decreases very fast when going away from the real axis. For instance, one finds with $\chi_k = 2ik\pi/\log 2$:

$$|\Gamma(\chi_1)| = 5.45249 \cdot 10^{-7}; \qquad |\Gamma(\chi_2)| = 2.52468 \cdot 10^{-13}.$$

Thus all that is required is effective bounds on $|N(it)|$. These follows easily by refining the approach taken in the proof of Lemma 3.

Define for x and t real, the function (see Eq. (24)):

$$f(x, t) = 1 - (1 + x)^{-it} - (1 + 2x)^{-it} + (1 + 3x)^{-it}.$$

LEMMA.   *For $t \geq 1$ and $x < 3/2t$, one has:*

$$|f(x, t)| \leq 16x^2t^2.$$

Proof.   The proof depends on the following easy observations: for $y \geq 0$:

$$\log(1 + y) \leq y \tag{1}$$

and for $|u| < \frac{1}{2}$:

$$|e^u - 1 - u| \leq |u|^2. \tag{2}$$

which follows immediately from the inequality:

$$|e^u - 1 - u| \leq \frac{|u|^2}{2}\left(1 + \frac{|u|}{3} + \frac{|u|^2}{12} + \cdots\right)$$

$$\leq \frac{|u|^2}{2(1 - |u|)}. \tag{3}$$

Thus rewriting the definition of f in exponential form

$$f(x, t) = 1 - e^{-it\log(1 + x)} - e^{-it\log(1 + 2x)} + e^{-it\log(1 + 3x)}$$

we find using (2) that, when $3xt < \frac{1}{2}$,

$$f(x, t) = -it \log \frac{1 + 3x}{(1 + x)(1 + 2x)} + R$$

where the remainder R satisfies

$$|R| \leqslant t^2(\log^2(1 + x) + \log^2(1 + 2x) + \log^2(1 + 3x))$$
$$\leqslant 14t^2x^2. \tag{5}$$

Now since

$$\left| \log \frac{1 + 3x}{(1 + x)(1 + 2x)} \right| \leqslant 2x^2, \tag{6}$$

we obtain

$$|f(x, t)| \leqslant 2tx^2 + 14t^2x^2 \leqslant 16x^2t^2. \quad \blacksquare$$

The above lemma can be used for two purposes: (1) bounding the values of $|N(it)|$ for large t; (2) bounding the truncation errors when estimating $N(t)$ from the sum of its first few terms.

COROLLARY. *For all $t \geqslant 1$, $N(it)$ satisfies*

$$|N(it)| \leqslant t^2 + 7t + 7. \tag{7}$$

Proof. Consider the form (24) of $N(it)$. With the notations of the lemma, it is

$$N(it) = -1^{-it} - 2^{-it} - 3^{-it} + \sum_{j \geqslant 1} \frac{(-1)^{\nu(j)}}{(4j)^{it}} \alpha(j, it) \tag{8}$$

where $\alpha(j, it) = f(1/4j, t)$. Define $j_0(t) = \max(\lfloor t/6 \rfloor, 1)$ so that $1/4j_0 \leqslant 3/2t$. Splitting the sum in (8) as $\sum_{j \geqslant 1} = \sum_{1 \leqslant j \leqslant j_0} + \sum_{j_0 < j}$ and applying the trivial bound $|f(x, t)| \leqslant 4$ to the first sum and the bound of the lemma to the other one, we find

$$|N(it)| \leqslant 3 + 4j_0 + 16t^2 \sum_{j > j_0} \frac{1}{(4j)^2}$$
$$\leqslant 3 + 4j_0 + \frac{t^2}{j_0} \leqslant 7 + 7t + t^2. \quad \blacksquare \tag{9}$$

The modulus of $N(\chi_1)$ is found by direct numerical computations to be less than 6, and one has

$$N(\chi_1) \approx -4.42 - 3.99i; \qquad N(\chi_2) \approx -6.55 - 3.17i; \qquad N(\chi_3) \approx +2.75 + 1.77i.$$

Thus using these values, one can check that $|p_1| < 0.5 \ 10^{-6}$, $|p_2| < 10^{-9}$, and that the p_k with $k > 2$ are much smaller and exponentially decreasing with the basis of the exponential equal to $e^{-\pi^2/\log 2} \approx 0.6584 \ 10^{-6}$.

ACKNOWLEDGMENTS

Note added in proof. The sequence $(-1)^{v(p)}$ that occurred repeatedly here is the classical Morse–Thue sequence. Using the Dirichlet generating function $N(s)$, Allouche *et al.* (Automates finis et séries de Dirichlet, *J. Inform. Math.*, Publ. Math. Université de Caen, 1985) have obtained several interesting properties of that sequence, including a proof of a curious identity of Shallit (compare with our Theorem 3A):

$$\frac{1}{\sqrt{2}} = \frac{2}{3} \prod_{p=1}^{\infty} \left[\frac{(4p+1)(4p+4)}{(4p+2)(4p+3)} \right]^{(-1)^{v(p)}}.$$

REFERENCES

1. G. Doetsch, "Handbuch der Laplace-Transformation," Birkhauser, Basel, 1950.
2. P. Flajolet, Approximate counting: A detailed analysis, *BIT* **25** (1985), 113–134.
3. P. Flajolet and N. Martin, Probabilistic counting, in "Proc. 24th IEEE Sympos. Foundations of Computer Science, Nov. 1983," pp. 76–82.
4. D. E. Knuth, "The Art of Computer Programming: Sorting and Searching," Addison–Wesley, Reading, Mass., 1973.
5. V. Y. Lum, P. S. T. Yuen, and M. Dodd, Key to address transformations: A fundamental study based on large existing formatted files, *Comm. ACM* **14** (1971), 228–239.
6. R. Morris, Counting large numbers of events in small registers, *Comm. ACM* **21** (1978), 840–842.
7. I. Munro and P. Spira, Sorting and searching in multisets, *SIAM J. Comput* **5**, No. 1 (1976), 1–8.
8. P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access Path Selection in A Relational Database Management System," Report RJ-2429, IBM San Jose Res. Lab., Aug. 1979.