Invited lecture, STACS06, Marseille, February 2006.
Proceedings in *Lecture Notes in Computer Science*.

THE UBIQUITOUS DIGITAL TREE

PHILIPPE FLAJOLET

Abstract. The *digital tree* also known as *trie* made its first appearance as a general-purpose data structure in the late 1950's. Its principle is a recursive partitioning based on successive bits or digits of data items. Under various guises, it has then surfaced in the management of very large data bases, in the design of efficient communication protocols, in quantitative data mining, in the leader election problem of distributed computing, in data compression, as well as in some corners of computational geometry. The algorithms are invariably very simple, easy to implement, and in a number of cases surprisingly efficient. The corresponding quantitative analyses pose challenging mathematical problems and have triggered a flurry of research works. Generating functions and symbolic methods, singularity analysis, the saddle-point method, transfer operators of dynamical systems theory, and the Mellin transform have all been found to have a bearing on the probabilistic behaviour of trie algorithms. We offer here a perspective on the rich algorithmic, analytic, and probabilistic aspects of tries, culminating with a connection between a sorting problem and the Riemann hypothesis.

While, in the course of the 1980s and 1990s, a large portion of the theoretical computer science community was massively engaged in worst-case design and analysis issues, the discovery of efficient algorithms continued to make tangible progress. Such algorithms are often based on simple and elegant ideas, and, accordingly, their study is likely to reveal structures of great mathematical interest. Also, efficiency is much better served by probabilistic analyses[1] than by the teratological constructions of worst-case theory. I propose to illustrate this point of view by discussing a fundamental process shared by algorithmics, combinatorics, and discrete probability theory—the digital tree process. Because of space-time limitations, this text, an invited lecture at STACS'06, cannot be but a brief guide to a rich subject whose proper development would require a book of full length.

1. The basic structure

Consider first as domain of our data items the set of all infinitely long binary strings, $\mathcal{B} = \{\mathtt{0},\mathtt{1}\}^\infty$. The goal is to devise a data structure in which elements of $\mathcal{B}$ can be stored and easily retrieved. Given a finite set $\omega \subset \mathcal{B}$ like

$$\omega = \big\{\mathtt{110100}\cdots,\quad \mathtt{01011}\cdots,\quad \mathtt{01101}\cdots\big\},$$

a natural idea is to form a *tree* in which the left subtree will contain all the elements starting with `0`, all elements starting with `1` going to the right subtree. (On the example, the last two strings would then go to the left subtree, the first one to the

[1]To be mitigated by common sense and a good feel for algorithmic engineering, of course!

right subtree.) The splitting process is repeated, with the next bit of data becoming discriminant. Formally, given any $\omega \subset \mathcal{B}$, we define

$$\omega \setminus \mathtt{0} := \{\sigma \mid \mathtt{0}\,\sigma \in \Omega\}, \qquad \omega \setminus \mathtt{1} := \{\sigma \mid \mathtt{1}\,\sigma \in \Omega\}.$$

The motto here is thus simply "filter and shift left". The *digital tree* or *trie* associated to ω is then defined by the recursive rule:

$$\text{trie}(\omega) := \begin{cases} \emptyset & \text{if } \omega = \emptyset \\ \sigma & \text{if } \omega = \{\sigma\} \\ \langle \bullet,\ \text{trie}(\omega \setminus \mathtt{0}),\ \text{trie}(\omega \setminus \mathtt{1})\rangle. \end{cases} \tag{1}$$

The tree trie(ω) makes it possible to search for elements of ω: in order to access $\sigma \in \mathcal{B}$, simply follow a path in the tree dictated by the successive bits of σ, going left on a 0 and right on a 1. This continues till either an external node containing one element, or being an empty node, is encountered. Insertion proceeds similarly (split an external node if the position is already occupied), while deletion is implemented by a dual process (merging a node with its newly vacant brother). The tree trie(ω) can be either constructed from scratch by a sequence of insertions or built by a top down procedure reflecting the inductive definition (1). In summary:

> *Tries serve to implement* dictionaries*, that is, they support the operations of insertion, deletion, and query.*

A trie thus bears some resemblance to the Binary Search Tree (BST), with the basic BST navigation based on relative order being replaced by decisions based on values (bits) of the data items:

$$\text{BST:}\quad \langle x,\ \text{“}< x\text{”},\ \text{“}> x\text{”}\,\rangle; \qquad \text{Trie:}\quad \langle \bullet,\ \text{“}= \mathtt{0}\text{”},\ \text{“}= \mathtt{1}\text{”}\,\rangle.$$

Equivalently, if infinite binary strings are interpreted as $[0,1]$ real numbers, the separation at the root is based on the predicates $< \frac{1}{2}, \geq \frac{1}{2}$. Like for the BST, a left to right traversal of the external nodes provides the set ω in sorted order: the resulting sorting algorithm is then essentially isomorphic to *Radix Exchange Sort* [43]. (This parallels the close relationship that BSTs entertain with the Quicksort algorithm.) The books by Knuth [43] and Sedgewick [54] serve as an excellent introduction to these questions.

There are many basic variations on the trie principle (1).

- *Multiway branching.* The alphabet $\{\mathtt{0},\mathtt{1}\}$ has been so far binary. An m-ary alphabet can be accommodated by means of *multiway branching*, with internal nodes being m-ary.
- *Paging.* Recursion may be halted as soon in the set ω has cardinality less than some fixed threshold b. The standard case is $b = 1$. The general case $b \geq 1$ corresponds to "bucketing" or *paging* and is used for retrieval from secondary memory.
- *Finite-length data.* Naturally occurring data tend to be of finite length. The trie can then be implemented by appending a *terminator symbol* to each data item, which causes branching to stop immediately.
- *Digital search trees (DSTs).* These are a hybrid between BSTs and tries. Given a *sequence* of elements of $\mathcal{B}$, place the first element at the root of the tree, partition the rest according to the leading digit and proceed recursively. DSTs are well described and analysed in Knuth's volume [43].

Their interest as a general purpose data structure has faded, but they have been found to play an important rôle in connection with data compression algorithms.

— *Patricia tries.* They are obtained from tries by adding skip fields in order to collapse sequences of one way branches.

Let us point out at this stage that, as a general purpose data structure, tries and their kin are useful for performing not only dictionary operations, but also set intersection and set union. This fact was recognized early by Trabb Pardo [58]. The corresponding algorithms are analysed in [27], which also contains a thorough discussion of the algebra of finite-length models.

Complexity issues. Under the basic model of infinitely long strings, the worst-case complexity of the algorithms can be arbitrarily large. In the more realistic case of finite-length strings, the worst-case search cost may equal the length of the longest item, and this may well be quite a large quantity. Like for many probabilistic algorithms, what is in fact relevant is the "typical" behaviour of the trie, measured either on average or in probability under realistic data models. *Analysis of algorithms* plays here a critical rôle in helping us decide in which contexts a trie can be useful and how parameters should be dimensioned for best effect. This is the topic we address next.

2. Random tries

The field of analysis of algorithms has evolved over the past two decades. The old-style recurrence approach is nowadays yielding ground to modern "symbolic methods" that replace the study of sequences of numbers (counting sequences, probabilities of events, average-case values or moments of parameters) by the study of *generating functions*. The algebra of series and the analysis of functions, mostly *in the complex domain* $\mathbb{C}$, then provide precise asymptotic information on the original sequence. For an early comparative analysis of tries and digital search trees in this perspective, see [29].

The algebra of generating functions. Let (f_n) be a numeric sequence; its *exponential generating function* (EGF) is by definition the formal sum

$$f(z) = \sum_{n\geq 0} f_n \frac{z^n}{n!}. \tag{2}$$

(We systematically use the same groups of letters for numeric sequences and their EGFs.) Consider a parameter ϕ defined inductively over tries by

$$\phi[\tau] = t[\tau] + \phi[\tau_0] + \phi[\tau_1]. \tag{3}$$

There, the trie τ is of the form $\langle \bullet, \tau_0, \tau_1\rangle$, and the quantity $t(\tau)$, called the "toll" function, often only depends on the number of items stored in τ (so that $t(\tau) = t_n$ if τ contains n data). Our goal is to determine the expectation ($\mathbb{E}$) of the parameter ϕ, when the set ω on which the trie is built comprises n elements.

The simplest probabilistic model assumes bits in strings to be identically independently distributed,

$$\mathbb{P}(\mathtt{0}) = p, \qquad \mathbb{P}(\mathtt{1}) = q = 1 - p,$$

the n strings of ω furthermore being drawn independently. This model is known as the *Bernoulli model*. The *unbiased model* also known as *uniform model* corresponds

to the further condition $\mathbb{P}(\mathtt{0}) = \mathbb{P}(\mathtt{1}) = \frac{1}{2}$. Under the general Bernoulli model, the inductive definition (3) admits a direct translation

$$\phi(z) = t(z) + e^{qz}\phi(pz) + e^{pz}\phi(qz). \tag{4}$$

There $\phi(z)$ is the EGF of the sequence (ϕ_n) and $\phi_n = \mathbb{E}_n(\phi)$ is the expectation of the parameter $\phi[\cdot]$ taken over all trees comprising n data items. The verification from simple rules of series manipulation is easy: it suffices to see that, given n elements, the probability that k of them go into the left subtree (i.e, start with a $\mathtt{0}$) is the binomial probability $p^k q^{n-k}\binom{n}{k}$, so that, as regards expectations,

$$\phi_n = t_n + \sum_k p^k q^{n-k}\binom{n}{k}(\phi_k + \phi_{n-k}).$$

For the number of binary nodes in the tree, a determinant of storage complexity, the toll is $t_n = 1 - \delta_{n0} - \delta_{n1}$. For path length, which represents the total access cost of all elements, it becomes $t_n = n - \delta_{n0}$. The functional equation (4) can then be solved by iteration. Under the unbiased Bernoulli model, we have for instance

$$\phi(z) = t(z) + 2e^{z/2}t(\frac{z}{2}) + 4e^{3z/4}t(\frac{z}{4}) + \cdots .$$

Then, expansion around $z = 0$ yields coefficients, that is expectations. We quote under the unbiased Bernoulli model

> *The expected size (number of binary nodes) and the expected path length of a trie built out of n uniform independent random keys admit the explicit expressions*

$$S_n = \sum_{k\geq 0} 2^k \left(1 - (1-2^{-k})^n - \frac{n}{2^k}(1-2^{-k})^{n-1}\right), \quad P_n = n\sum_{k\geq 0}\left(1 - (1-2^{-k})^{n-1}\right).$$

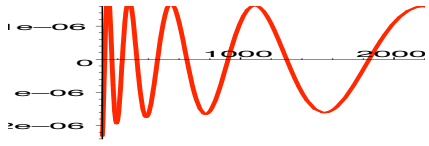
This result has been first discovered by Knuth in the mid 1960's.

Asymptotic analysis and the Mellin transform. A plot of the averages, S_n and P_n, is instructive. It strongly suggests that S_n is asymptotically linear, $S_n \sim cn(?)$, while $P_n \sim n\lg n(?)$, where $\lg x := \log_2 x$. As a matter of fact, the conjecture on size is false, but by an amazingly tiny amount. What we have is the following property:

> *The expected size (number of binary nodes) and the expected path length of a trie built out of n uniform independent random keys*

$$S_n = \frac{n}{\log 2}\left(1 + \epsilon(\lg n)\right) + o(n), \qquad P_n = n\lg n + O(n). \tag{5}$$

> *There $\epsilon(x)$ is a continuous period function with amplitude $< 10^{-5}$:*



We can only give here a few indications on the proof techniques and refer the reader to our long survey [24]. The idea, suggested to Knuth by the great analyst De Bruijn, is to appeal to the theory of integral transforms. Precisely, the *Mellin*

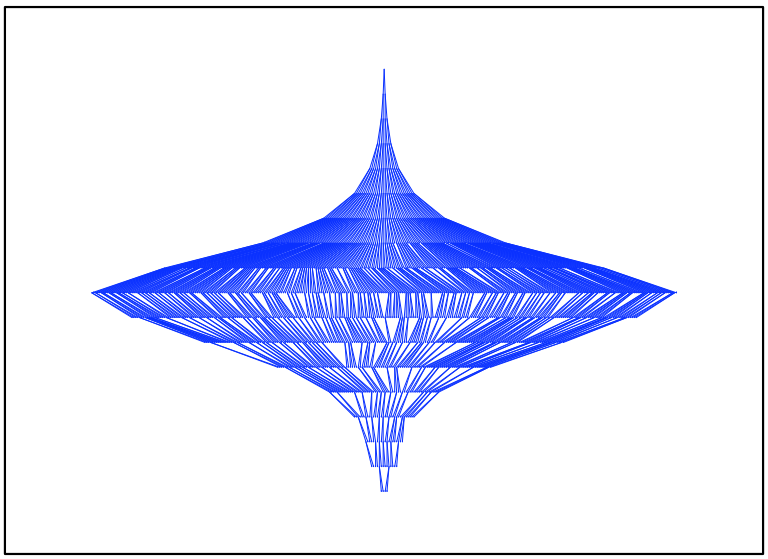


FIGURE 1. A random trie of size $n = 500$ built over uniform data.

transform associates to a function $f(x)$ (with $x \in \mathbb{R}_{\geq 0}$) the function $f^\star(s)$ (with $s \in \mathbb{C}$) defined by

$$f^\star(s) = \mathcal{M}[f(x), x \mapsto s] := \int_0^\infty f(x) x^{s-1}\, dx.$$

For instance $\mathcal{M}[e^{-x}] = \Gamma(s)$, the familiar Gamma function [61]. Mellin tranforms have two strikingly powerful properties. First, they establish a correspondence between the asymptotic expansion of a function at $+\infty$ (resp. 0) and *singularities* of the transform in a right (resp. left) half-plane. Second, they factorize *harmonic sums*, which correspond to a linear superposition of models taken at different scales.

Consider the function $s(x) = e^{-x} S(x)$, where $S(z)$ is the EGF of the sequence (S_n) of expectations of size. (This corresponds to adopting a Poisson rather than Bernoulli model; such a choice does not affect our asymptotic conclusions since, as can be proved elementarily [43, p. 131], $S_n - s(n) = o(n)$.) A simple calculation shows that

$$s(x) = \sum_{k \geq 0} 2^k \left[1 - \left(1 + \frac{x}{2^k}\right) e^{-x/2^k}\right], \qquad s^\star(s) - \frac{(1+s)\Gamma(s)}{1 - 2^{1+s}}.$$

The asymptotic estimates (5) result from there, given that a pole at α of the transform corresponds to a term $x^{-\alpha}$ in the asymptotic expansion of the original function. It is the existence of complex poles at

$$s = -1 + \frac{2ik\pi}{\log 2}, k \in \mathbb{Z},$$

that, in a sense, "creates" the periodic fluctuations present in $s(x)$ (and hence in S_n).

Models. Relative to the unbiased model (unbiased 0-1 bits in independent data), the expected size estimate expresses the fact that storage occupation is at most of linear growth, despite the absence of convergence to a constant occupation ratio. The path length estimate means that the trie is nearly optimal in some information theoretic sense, since an element is typically found after $\sim \lg n$ binary questions. The profile of random trie under this model is displayed in Figure 1.

The ϵ-fluctuation, with an amplitude of 10^{-5}, in the asymptotic behaviour of size tends to be quite puzzling to programmers. Undeniably, such fluctuations will never be detected on simulations not to mention executions on real-life data. However, mathematically, their presence implies that most elementary strategies for analysing trie algorithms are doomed to failure. (See however [55, p. 403] for an elementary approach.) It is a fact that *no coherent theory of tries can be developed without taking such fluctuations into account*. For instance, the exact order of the variance of trie size and trie path length *must* involve them [41, 42]. As a matter of fact, some analyses, which were developed in the late 1970s and ignored fluctuations, led to wrong conclusions, even regarding the *order of growth* of important characteristics of tries.

Back to modelling issues, the uniform model seems at first sight to be of little value. It is however fully justified in situations where elements are accessed via *hashing* and the indications it provides are precious: see for instance the discussion of dynamic and extendible hashing in Section 4. Also, the Mellin transform technology is equally suitable for extracting asymptotic information from the baised Bernoulli model ($p \neq q$). In that case, it is found that, asymptotically[2]

$$S_n \approx \frac{n}{H}, \qquad P_n \sim \frac{n}{H} \log n, \tag{6}$$

where $H \equiv H(p,q) = -p \log p - q \log q$ is the *entropy* function of the Bernoulli (p,q) model. The formulæ admit natural generalizations to m-ary alphabets.

The estimates (6) indicate that trees become less efficient roughly in proportion to entropy. For instance, for a four symbol alphabet, where each letter has probability larger than 0.10, (this is true of most $\{\mathsf{A},\mathsf{G},\mathsf{C},\mathsf{T}\}$ genomic sequences), the degradation of performance is by less than a factor of 1.5 (i.e., a 50% loss at most). In particular linear storage and logarithmic access costs are preserved. Equally importantly, more realistic and considerably more general data models can be analysed precisely: see Section 8 relative to dynamical sources, which encapsulate the framework of Markov chains as a particular case.

Amongst the many fascinating techniques that have proved especially fruitful for trie analyses, we should also mention: Rice's method from the calculus of finite differences [30, 43]; *analytic depoissonization* specifically developed by Jacquet and Szpankowski [40], which has led to marked successes in the analysis of dictionary-based compression algorithms. Complex analysis, that is, the theory of *analytic* (holomorphic) functions is central to most serious works in the area. Books that discuss relevant methods include those of Sedgewick-Flajolet [31], Hofri [35], Mahmoud [45], and Szpankowski [57].

[2]The symbol '$\sim$' is used throughout in the strict sense of asymptotic equivalence; the symbol '$\approx$' is employed here to represent a numerical approximation up to tiny fluctuations.

3. Multidimensional tries

Finkel and Bentley [21] adapted the BST to multidimensional data as early as 1974. Their ideas can be easily transposed to tries. Say you want to maintain sets of points in d-dimensional space. For $d = 2$, this gives rise to the standard quadtrie, which associates to a finite set $\omega \subset [0,1]^2$ a tree defined as follows.

(*i*) If $\mathrm{card}(\omega) = 0$, then $\mathrm{quadtrie}(\omega) = \emptyset$;
(*ii*) if $\mathrm{card}(\omega) = 1$, then $\mathrm{quadtrie}(\omega)$ consists of a single external node containg ω;
(*iii*) else, partition ω into the four subsets determined by their position with respect to the center $(\frac{1}{2}, \frac{1}{2})$ of space, and attach a root to the subtrees recursively associated to the four subsets $(NW, NE, SW, SE$, where NE stands for North-East, etc).

A moment's reflection shows that the quadtrie is equivalent to the 4-way trie built over an alphabet of cardinality 4: given any point $P = (x, y)$, write its coordinates in binary, $x = x_1 x_2, \ldots$ and $y = y_1 y_2 \cdots$, then encode the pair of coordinates "in parallel" over the alphabet $\{a, b, c, d\}$, where, say, $a = (0,0)$, $b = (0,1)$, $c = (1,0)$, $d = (1,1)$. The quadtrie is none other than the 4-way trie built over the set of such encodings.

Another idea of Bentley [6] gives rise to *k-d-tries*. For $d = 2$, associate to each point $P = (x, y)$, where $x = x_1 x_2, \ldots$ and $y = y_1 y_2 \cdots$, the binary string $z = x_1 y_1 x_2 y_2 \cdots$ obtained by interleaving bits of both coordinates. The k-d-trie is the binary trie built on the z-codes of points.

Given these equivalences, the analytic methods of Section 2 apply verbatim:

> *Over uniform independent data, the d-dimensional quadtrie requires on average $\approx cn$ pointers, where $c = 2^d / \log 2^d$; for k-d-tries this number of pointers is $\approx c'n$, where $c' = 2/\log 2$. The mean number of bit accesses needed by a search is $\sim \log_2 n$.*

Roughly, multidimensional tries grant us fast access to multidimensional data. The storage requirements of quad-tries may however become prohibitive when the dimension of the underlying data space grows, owing to a large number of null pointers that carry little information but encumber memory.

Quadtries and k-d-tries also serve to implement *partial-match queries* in an elegant way. This corresponds to the situation, in d-dimensional space, where s out of d coordinates are specified and all points matching the s known coordinates are to be retrieved[3]. Put otherwise, one wants to reconstruct data given partial knowledge of their attributes. It is easy to set up recursive procedures reflected by inductive definitions for the cost parameters of such queries. The analytic methods of Section 2 are then fully operational. The net result, due to Flajolet and Puech [26], is

> *The mean number of operations needed to retrieve objects of d-dimensional space, when s out of d of their coordinates are known, is $O(n^{1-/s/d})$. The estimate holds both for quadtries and for k-d-tries.*

[3]For an excellent discussion of spatial data structures, we redirect the reader to Samet's books [53, 52].

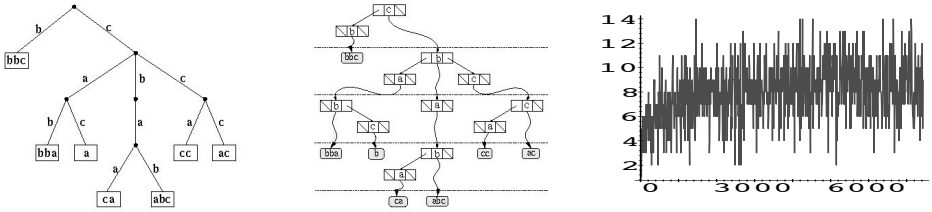


FIGURE 2. Left: a trie. Middle: a corresponding TST. Right: cost of TST search on *Moby Dick* (number of letter comparisons against number of words scanned).

In contrast, quadtrees and k-d-trees (based on the BST concept) require

$$O(n^{1-s/d+\theta(s/d)})$$

operations for some function $\theta > 0$. For instance, for comparison-based structures, the case $s = 1$ and $d = 2$, entails a complexity $O(n^\alpha)$, where $\alpha = \frac{\sqrt{17}-3}{2} \doteq 0.56155$, which is of higher order than the $O(\sqrt{n})$ attached to bit-based structures. The better balancing on bit based structures pays—at least on uniform enough data.

Devroye [11] has provided an insightful analysis of tries ($d = 1$) under a density model, where data are drawn independently according to a probability density function spread over the unit interval. A study of multidimensional search along similar lines would be desirable.

Ternary search tries (TST). Multiway tries start require a massive amount of storage when the alphabet cardinality is large. For instance, a dictionary that would contain the word APPLE should have null pointers corresponding to the non-existent forms APPLA, APPLB, APPLC, etc. When attempting to address this problem, Bentley and Sedgewick [5] made a startling discovery: it is possible to design a highly efficient hybrid of the trie and the BST. In essence, you build an m-way trie (with m the alphabet cardinality), but implement the local decision structure at each node by a BST. The resulting structure, known as a TST, is simply a *ternary tree*, where, at each node, a comparison between letters is performed. Upon equality, go down one level, i.e., examine the next letter of the item to be retrieved; else proceed to the left or the right, depending on the outcome of the comparison between letters. It's as simple as that!

The TST was analysed by Clément, Flajolet, and Vallée [7, 8]. Quoting from [7]: *Ternary search tries are an efficient data structure from the information theoretic point of view since a search costs typically about* $\log n$ *comparisons. For an alphabet of cardinality 26, the storage cost of ternary search tries is about 9 times smaller than standard array-tries.* (Based on extensive natural language data.)

4. HASHING AND HEIGHT

Paged tries also known as *bucket tries* are digital trees defined like in (1), but with recursion stopped as soon as at most b elements have been isolated. This technique is useful in the context of a two-level memory. The tree itself can then be stored in core-memory as an index. Its end-nodes then point to pages or buckets in secondary memory. The technique can then be applied to hashed values of records, rather than

records themselves which may be rather non-uniformly distributed in practice. The resulting algorithm is known as *dynamic hashing* and is due to Larson [44]. It is interesting to note that it was first discovered without an explicit reference to tries, the author viewing it as an evolution of hashing with separate chaining (in a paged environment), and with the splitting of buckets replacing costly chains of linked pages. The analysis methods of Section 2 show that the mean number of pages is

$$\approx \frac{n}{b \log 2}.$$

In other words, the pages are approximately 69% filled, a score that is comparable to the one of B–trees.

For very large data bases, the index of dynamic hashing may become too large to fit in primary memory. Fagin *et al.* [16] discovered an elegant way to remedy the situation, known as *extendible hashing* and based on the following principle:

> **Perfect tree embedding.** Let a tree τ of some height H be given, with only the external nodes of τ containg information. Form the perfect tree P of height H (i.e., all external nodes are at distance H from the root). The tree τ can be embedded into the perfect tree with any information being pushed to the external nodes of P. (This in general involves duplications.) The perfect tree with decorated external nodes can then be represented as an array of dimension 2^H, thereby granting direct access to its leaves.

In this way, in most practical situations, only two disc accesses suffice to reach any item stored in the structure—one for the index, which is a paged array, the other for the referenced page itself. This algorithm is the definite solution to the problem of maintaining very large hashed tables.

In its time, extendible hashing posed a new problem to analysts. Is the size of the index of linear or of superlinear growth? That question brings the analysis of height into our algorithmic games. General methods of combinatorial enumeration [31, 56, 33, 62] are relevant to derive the basic equations. The starting point is ($[z^n]f(z)$ represents the coefficient of z^n in the expansion of $f(z)$ at 0)

$$\mathbb{P}_n(H \le h) = n![z^n] e_b\left(\frac{z}{2^h}\right)^{2^h}, \qquad e_b(z) := 1 + \frac{z}{1} + \cdots + \frac{z^b}{b!}.$$

The problem is thus to extract coefficients of large index in the large power of a fixed function (here, the truncated exponential, $e_b(z)$). The saddle point method [10, 31] of complex analysis comes to mind. It is based on Cauchy's coefficient formula,

$$[z^n]f(z) = \frac{1}{2i\pi}\int_{O^+} f(z)\,\frac{dz}{z^{n+1}},$$

which relates *values* of an analytic function to its coefficients, combined with the choice of a contour that crosses a saddle point of the integrand (Figure 3). The net result of the analysis [22] is the following:

> *Height of a paged b-trie is of the form*
>
> $$\left(1 + \frac{1}{b}\right)\log n + O(1)$$
>
> *both on average and in probability. The limit distributions are in the form of a double exponential function.*

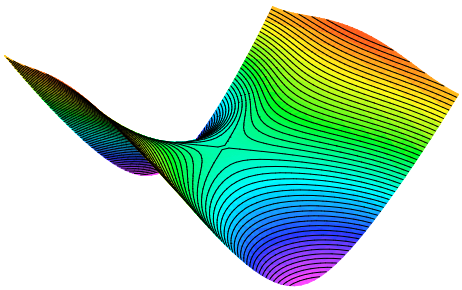


FIGURE 3. A saddle point of the modulus of an analytic function.

> *The size* (2^H) *of the extendible-hashing index is on average of the form* $C(\log n)n^{1+1/b}$*, with* $C(\cdot)$ *a bounded function. In particular, it grows non-linearly with* n*.*

(See also Yao [63] and Régnier [50] for earlier results under the Poisson model.)

The ideas of extendible hashing are also susceptible of being generalized to higher dimensional data: see Régnier's analysis of *grid-file* algorithms in [51].

Level compressed tries. Nilsson and Karlsson [48] made a sensation when they discovered the "*LC trie*" (in full: Level Compressed trie): they demonstrated that their data structure could handle address lookup in routing tables with a standard PC in a way that can compete favorably with dedicated hardware embedded into routers. One of their beautifully simple ideas consists in compressing the perfect tree contained in a trie (starting from the root) into a single node of high degree—this principle is then used recursively. It is evocative of a partial realization of extendible hashing. The decisive advantage in terms of execution time stems from the fact that chains of pointers are replaced by a single array access, while the search depth decreases to $O(\log\log n)$ for a large class of distributions [12, 13, 48]

5. Leader election and protocols

Tries have found unexpected applications as an abstract structure underlying several algorithms of distributed computing. We discuss here leader election and the tree protocol due to Capetanakis-Tsybakov-Mikhailov (also known as the CTM protocol or the stack protocol). In both cases, what is assumed is a shared channel on which a number of stations are hooked. At any discrete instant, a station can broadcast a message of unit duration. It can also sense the channel and get a ternary feedback: $\mathsf{0}$ for silence; $\mathsf{1}$ for a succesful transmission; $\mathsf{2}^+$ for a collision between an unknown number of individuals.

The leader election protocol in its bare version is as follows:

> **Basic leader election.** At time $t = 0$ the group G of all the n stations[4] on the channel are ready to start a round for electing a reader. Each one transmits its name (an identifier) at time 1. If $n = 0$, the channel has remained silent and nothing happens. If $n = 1$, the channel fedback is $\mathsf{1}$ and the corresponding individual is elected. Else all contenders in G flip a coin. Let G_H (resp G_T) be the subgroup of those who flipped head (resp. tails). Members of the group G_T withdraw instantly from the competition. Members of G_H repeat the process over the next time slot.

[4]The number n is unknown.

We expect the size of G to decrease by roughly a factor of 2 each time, which suggests that the number of rounds should be close to $\log_2 n$. The basic protocol described above may fail with probability close to 0.27865, see [55, p. 407], but it is easily amended: it suffices to let the last nonempty group of contenders (likely to be of small size) start again the process, and repeat, until a group of size 1 comes out.

This leader election protocol is a perfect case for the analytic methods evoked earlier. The number of rounds for instance coincides with the leftmost branch of tree, a parameter easily amenable to the analytic techniques described in Section 2. The complete protocol has been analysed thoroughly by Prodinger [49] and Fill *et al.* [20]. Fluctuations are once more everywhere to be found.

The tree protocol was invented around 1977 independently in the USA and in the Soviet Union. For background, references, and results, we recommend the special issue of the *IEEE Transactions on Information Theory* edited by Jim Massey [46]. The idea is very simple: instead of developing only the leftmost branch of a trie, develop cooperatively the whole trie.

> **Basic tree protocol.** Let G be the group of stations initially waiting to transmit a message. During the first available slot, all stations of G transmit. If the channel feedback is `0` or `1`, transmission is complete. Else, G is split into G_H, G_T. All the members of G_H are given precedence and resolve their collisions between themselves, by a recursive application of the protocol. Once this phase has been completed, the group G_T takes its turn and proceeds similarly.

Our description presupposes a perfect knowledge of the system's state by all protagonists at every instant. It is a notable fact that the protocol can be implemented in a fully decentralized manner, each station only needing to monitor the channel feedback (and maintain a priority stack, in fact, a simple counter). The time it takes to resolve the contention between n initial colliders coincides with the total number of nodes in the corresponding trie (think of stations as having predetermined an infinite sequence of coin flips), that is, $2S_n + 1$ on average. The unbiased Bernoulli model is *exactly* applicable, given a decent random number generator. All in all, the resolution of a collision of multiplicity n takes times asymptotic to (cf Equation (5))

$$\frac{2}{\log 2} n,$$

upon neglecting the usual tiny fluctuations. In other words, the service time per customer is about $2/\log 2$. By standard queuing theory arguments, the protocol is demonstrably stable for all arrival rates λ satisfying $\lambda < \lambda_{\max}$, where

$$\lambda_{\max} = \frac{\log 2}{2} \pm \cdot 10^{-5} \doteq 0.34657.$$

In contrast, the Ethernet protocol has been proved unstable by Aldous in a stunning study [1].

We have described above a simplified version (the one with so-called "blocked arrivals") of the tree protocol. An improved version allows competitors to enter the game as soon as they are ready. This "free arrivals" version leads to nonlocal functional equations of the form

$$\psi(z) = t(z) + \psi(\lambda + pz) + \psi(\lambda + qz),$$

whose treatment involves interesting properties of iterated functions systems (IFS) and associated Dirichlet series: see G. Fayolle *et al* [17, 18] and the account in Hofri's book [35]. The best protocol in this class (Mathys-Flajolet [47]) was largely discovered thanks to analytic techniques, which revealed the following: *a throughput of* $\lambda_{\max} = 0.40159$ *is achievable when combining free arrivals* and *ternary branching.*

6. Probabilistic counting algorithms

A problem initially coming from query optimization in data bases led Nigel Martin and me to investigate, at an early stage, the following problem: *Given a multiset* M *of data of sorts, estimate the number of* distinct *records, also called* cardinality*, that* M *contains.* The cardinality estimation problem is nowadays of great relevance to data mining and to network management. (We refer to [23] for a general discussion accompanied by references.)

The idea consists in applying a hash function h to each record. Then bits of hashed values are observed. The detection of patterns in observed hashed values can serve as a fair indicator of cardinality. Note that, by construction, such algorithms are totally insensitive to the actual structure of repetitions in the original file (usually, no probabilistic assumption regarding these can be made). Also, once a hash function of good quality has been chosen, the hashed values can legitimately be identified with uniform random strings. This makes it possible to trigger a virtuous cycle, involving probabilistic analysis of observables and suitably tuned cardinality estimators.

The original algorithm, called *probabilistic counting* [25], was based on a simple observable: the length L of the longest initial run of 1-bits in $h(M)$. This quantity can be computed with an auxiliary memory of a single 32 bit word, for reasonable file cardinalities, say $n \leq 10^9$. We expect $L_n \approx \log_2 n$, which suggests 2^{L_n} as a rough estimator of n. The analysis of L_n is attached to that of tries—we are in a way developing the leftmost branch of a pseudo-trie to which the methods of Section 2 apply perfectly. It involves the Thue-Morse sequence, which is familiar to aficionados of combinatorics on words. However, not too surprisingly, the rough estimate just described is likely to be typically off, by a little more than one binary order of magnitude, from the actual (unknown) value of n. Improvements are called for.

The idea encapsulated into the complete Probabilistic Counting algorithm is to emulate at barely any cost the effect of m independent experiments. There is a simple device, called "stochastic averaging" which makes it possible to do so by distribution into buckets and then averaging. The resulting algorithm *estimates cardinalities using* m words *of memory, with a relative accuracy of about* $\frac{0.78}{\sqrt{m}}$. It is pleasant to note that a multiplicative correction constant, provided by a Mellin transform analysis,

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

(γ is Euler's constant, $\varepsilon(p) \in \{-1, +1\}$ indicates the parity of the number of 1-bits in the binary representation of p) enters the very design of the algorithm by ensuring that it is free of any systematic bias.

```
eddcdfdddddfcfdeeeeeefeedfedeffeffdefefeb fedefceffdefd
fefeecfdedeeededffefeffeecddefcfcddccedddcfddedeccdefdd
fcedddfdfedecddfedcfedcdfdeedegddcfededggfffdggdfgfegdg
ddddegddffededceeeefdedgfgdddeefdceeeefeeddeedefcfffdh
          hcgdccgchdfdchdehdgeeegfeedccfdedfddf
```

FIGURE 4. The LOGLOG algorithm asociates to a text a signature, from which the number of differents words can be inferred. Here, the signature of Hamlet uses $m = 256$ bytes, with which the cardinality of the vocabulary is estimated to an accuracy of 6.6%.

Recently, Marianne Durand and I were led to revisit the question, given the revival of interest in the area of network monitoring and following stimulating exchanges with Estan and Varghese (see, e.g., [15]). We realized that a previously neglected observable, the position $\widetilde{L}$ of the rightmost 1-bit in hashed values, though it has inferior probabilistic properties (e.g., a higher variance), can be maintained as a register in binary, thereby requiring very few bits. Our algorithm [14], called LOGLOG Counting *estimates cardinalities using* m bytes *of memory, with a relative accuracy of about* $\frac{1.3}{\sqrt{m}}$. Given that a word is four bytes, the overall memory requirement is divided by a factor of 3, when compared to Probabilistic Counting. This is, to the best of my knowledge, the most efficient algorithm available for cardinality estimation. Once more the analysis can be reduced to trie parameters and Mellin transform as well as the saddle point method play an important part.

For a highly valuable complexity-theoretic perspective on such questions see the study [2] by Alon, Matias, and Szegedy. In recent years, Piotr Indyk and his collaborators have introduced radically novel ideas in quantitative data mining, based on the use of *stable* distributions, but these are unfortunately outside of our scope, since tries do not intervene at all there.

7. SUFFIX TRIES AND COMPRESSION

Say you want to compress a piece of text, like the statement of Pythagoras' Theorem:

```
In any right triangle, the area of the square whose side is the hypotenuse
(the side of the triangle opposite the right angle) is equal to the sum
of the areas of the squares of the other two sides.
```

It is a good idea to notice that several words appear repeated. They could then be encoded once and for all by numbers. For instance:

```
1the,2angle,3triangle,4area,5square,6side,7right|In any 7 3, 1 4 of 1
5 whose 6 is 1 hypotenuse (1 6 of 1 3 opposite 1 7 2) is equal to 1 sum
of 1 4s of 1 5s of 1 other two 6s.
```

A dictionary of frequently encountered terms has been formed. That dictionary could even be recursive as in

```
1the,2angle,3tri2,4area,5square,6side,7right| ...
```

Around 1977–78, Lempel and Ziv developed ideas that were to have a profound impact of the field of data compression. They proposed two algorithms that make it possible to build a dictionary on the fly, and in a way that adapts nicely to the contents of the text. The first algorithm, known as LZ'78, is the following:

LZ'78 algorithm. Scan the text left to right. The text is segmented into phrases. At any given time, the cursor is on a yet unsegmented portion of the text. Find the longest phrase seen so far that matches the continuation of the text, starting from the cursor. A new phrase is created that contains this longest phrase plus one new character. Encode the new phrase with the rank of the previously matching phase and the new character.

For instance, "abracadabra" is segmented as follows

$$^0\mathsf{a}|\ ^0\mathsf{b}|\ ^0\mathsf{r}|\ \mathsf{a}^1\mathsf{c}|\ \mathsf{a}^1\mathsf{d}|\ \mathsf{a}^1\mathsf{b}|\ \mathsf{r}^3\mathsf{a}|\ \mathsf{ab}^6\mathsf{r}|\ \mathsf{ac}^4\mathsf{a}|\ ^0\mathsf{d}|\ \mathsf{abr}^7\mathsf{a}|,$$

resulting in the encoding

$$\mathsf{0a0b0r1c1d1b3a6r4a0d7a},$$

As it is well known, the algorithm can be implemented by means of a trie whose nodes store the ranks of the corresponding phrases.

From a mathematical perspective, the tree built in this way obeys the same laws as a *digital search tree* (DST), so that we'll start with examining them. The DST parameters can be analysed on average by the methods of Section 2, with suitable adjustments [28, 29, 43, 45, 57]. For instance, an additive parameter ϕ associated to a toll function t gives rise, at EGF level, to a functional equation of the form (in the unbiased case)

$$\phi(z) = t(z) + 2\int_0^z e^{t/2} f\left(\frac{t}{2}\right)\, dt,$$

which is now a difference-differential equation. The treatement is a bit more difficult, but the equation eventually succumbs to the Mellin technology. In particular, path length under a general Bernoulli model is found to be satisfy

$$P_n^\circ = \frac{1}{H} n \log n + O(n), \tag{7}$$

with H the entropy of the model.

Back to the LZ'78 algorithm. Equation (7) means that when n phrases have been produced by the algorithm, the total number of characters scanned is $N \sim H^{-1} n \log n$ on average. Inverting this relation[5] suggest the following relation between number of characters read and number of phrases produced:

$$n \sim H \frac{N}{\log N}.$$

Since a phrase requires at most $\log_2 N$ bits to be encoded, the total length of the compressed text should be $\sim hn$ with $h = H/\log 2$ the *binary entropy*. This handwaving argument suggests the true fact: for a memoryless (Bernoulli) source, *the entropic (optimal) rate is achieved by LZ compression*.

The previous argument is quite unrigorous. In addition, information theorists are interested not only in dominant asymptotics but in quantifying *redundancy*, which measures the distance to the entropic optimum. The nature of stochastic fluctuations is also of interest in this context. Jacquet and Szpankowski have solved these difficult questions in an important work [39]. Their treatment starts from the bivariate generating function $P^\circ(z, u)$ of path length in DSTs, which satisfies a nonlinear functional equation,

$$\frac{P^\circ(z,u)}{\partial z} = P^\circ(pz, u) \cdot P^\circ(qz, u).$$

[5]This is in fact a renewal type of argument.

They deduce asymptotic normality via a combination of inductive bounds, bootstrapping, analytic depoissonization, and the Quasi-Powers Theorem of analytic combinatorics [31, 36, 57]. (Part of the treatment makes use of ideas developed earlier by Jacquet and Régnier in their work establishing asymptotic normality of path length and size in tries [37].) From there, the renewal argument can be put on a sound setting. A full characterization of redundancy results and the fluctuations in the compression rate are determined to be asymptotically normal.

Suffix trees and antidictionaries. Given an infinitely long text $T \in \mathcal{B} \equiv \{0,1\}^\infty$, the *suffix tree* (or suffx trie) of index n is the trie built on the first n suffixes of T. Such trees are important as an indexing tool for natural language data. They are also closely related to a variant of the LZ algorithm, known as LZ'77, that we have just presented. A new difficulty in the analysis comes from the back that suffix trees are tries built on data that are intrinsically *correlated*, due to the overlapping structure of suffixes of a single text. Jacquet and Szpankowski [38] are responsible for some of the early analyses in this area. Their treatment relies on the autocorrelation polynomial of Guibas and Odlyzko [34] and on complex-analytic techniques. Julien Fayolle [19] has recently extended this methodology. His methods also provide insight on the quantitative behaviour of a new scheme due to Crochemore *et al.* [9], which is based on the surprising idea of using *antidictionaries*, that is, a description of some of the patterns that are *avoided* by the text.

8. Dynamical sources

So far, tries have been analysed when data are provided by a *source*, but one of a simple type. A new paradigm in this area is Brigitte Vallée's concept of *dynamical sources*. Such sources are most likely constituting the widest class of models, which can be subjected to a complete analytic treatment. To a large extent, Vallée's ideas [59] evolved from the realization that methods, originally developed for the purpose of analysing *continued fraction* algorithms [60], could be of a much wider scope.

Consider a transformation T of the unit interval that is piecewise differentiable and expanding: $T'(x) > 1$. Such a transformation is called a *shift*. It consists of several branches, as does the multivalued inverse function T^{-1}, which is formed of a collection of contractions. Given an initial value x_0, the sequence of iterates $(T^j(x_0))$ can then be encoded by recording at each iteration which branch is selected—this is a fundamental notion of symbolic dynamics. For instance, the function $T(x) = \{2x\}$ (with $\{w\}$ representing the fractional part of w) generates in this way the binary representation of numbers; from a metric (or probabilistic) point of view, it also describes the unbiased Bernoulli model. Via a suitable design, any biased Bernoulli model is associated with a shift, which is a piecewise-linear function. Markov chains (of any order) also appear as a special case. Finally, the continued fraction representation of numbers itself arises from the transformation

$$T(x) := \left\{ \frac{1}{x} \right\} = \frac{1}{x} - \left\lfloor \frac{1}{x} \right\rfloor .$$

A dynamical source is specified by a shift (which determines a symbolic encoding) as well as by an initial density on the unit interval. The theory thus unites Devroye's density model, Bernoulli and Markov models, as well as continued fraction

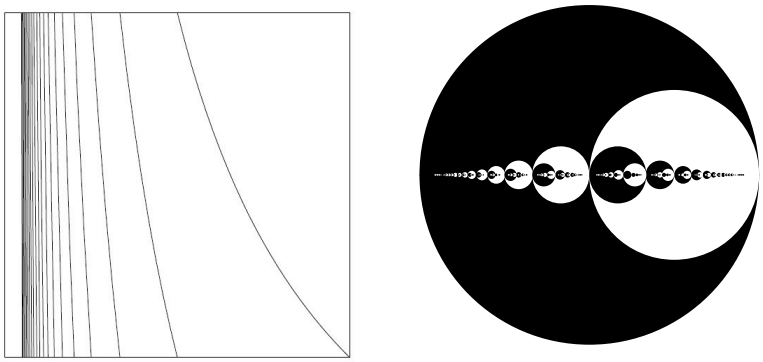


FIGURE 5. Dynamical sources: [left] the shift associated with continued fractions; [right] a rendering of fundamental intervals.

representions of real numbers and a good deal more. As opposed to earlier models, such sources take into account correlations between letters at an unbounded distance.

Tries under dynamical sources. Vallée's theory has been applied to tries, in particular in a joint work with Clément [8]. What it brings to the field is the unifying notion of *fundamental intervals* which are the subintervals of $[0, 1]$ associated to places corresponding to potential nodes of tries. Much transparence is gained by this way of viewing a trie process as a succession of refined partitions of the unit interval, and the main parameters of tries can be expressed simply in this framework.

Technically, a central rôle is played by Ruelle's *transfer operator*. Given a shift T with $\mathcal{H}$ the collection of its inverse branches, the transfer operator is defined over a suitable space of functions by

$$\mathcal{G}_s[f](x) := \sum_{h\in\mathcal{H}} |h'(x)|^s \, f \circ h(x).$$

For instance, in the continued fraction case, one has

$$\mathcal{G}_s[f](x) := \sum_{m\geq 1} \frac{1}{(m+x)^{2s}} f\left(\frac{1}{m+x}\right).$$

The quantity s there is a parameter that is *a priori* allowed to assume complex values. As Vallée showed, by considering iterates of $\mathcal{G}_s$, it then becomes possible to construct generating functions, usually of the Dirichlet type, associated with partitions into fundamental intervals. (In a way, the transfer operator is a supergenerating operator.) Equiped with these, it then becomes possible to express expectations and probability distributions of trie parameters, after a Mellin transform round. Then functional analysis comes into play (the operators have a discrete spectrum), to the effect that *asymptotic properties of tries built on dynamical sources are explicitly related to spectral properties of the transfer operator.*

As an example of unified formulæ, we mention here the mean value estimates of (6) which are seen to hold for an arbitrary dynamical source. The rôle of entropy in these formulæ comes out neatly—entropy is bound to be crucial under any dynamical source model. The analysis of height becomes almost trivial; the characteristic constant turns out to be in all generality none other than $\lambda_1(2)$, the dominant eigenvalue of operator $\mathcal{G}_2$.

We conclude this section with a brief mention of an algorithm due to Gosper, first described in the celebrated "Hacker's memorandum" also known as HAKMEM [4, Item 101A]. The problem is to compare two fractions $\frac{a}{b}$ and $\frac{c}{d}$. It is mathematically trivial, since

$$\frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd},$$

but the algorithms that this last formula suggests either involve going to multiprecision routines or operating with floating point arithmetics at the risk of reaching a wrong conclusion.

> **Gosper's comparison algorithm.** In order to compare $\frac{a}{b}$ and $\frac{c}{d}$, perform a continued fraction expansion of both fractions. Proceed in lazy mode. Stop as soon as a discrepant digit is encountered.

Gosper's solution operates within the set precision of data and is error-free (as opposed to the use of floating point approximations). For this and other reasons[6], it has been found to be of interest by the community of computational geometers engaged in the design of *robust* algorithms [3]. (It also makes an appearance the source code of Knuth's METAFONT, for similar reasons.)

In our perspective, the algorithm can be viewed as the construction of the digital tree associated to two elements accessible via their continued fraction representations. Vallée and I give a thorough discussions of the fascinating mathematics that surround its analysis in [32]. The algorithm extends to the lazy and robust comparison of a system of n fractions: it suffices to build, by lazy evaluation, the trie associated to continued fraction representations of the entries. What we found in [32] is the following result: *The expected cost of sorting n uniform random real numbers by lazy evaluation of their continued fraction representations satisfies*

$$P_n = K_0 n \log n + K_1 n + Q(n) + K_2 + o(1),$$

where ($\zeta(s)$ is the Riemann zeta function)

$$K_0 = \frac{6 \log 2}{\pi^2}, \qquad K_1 = 18\frac{\gamma \log 2}{\pi^2} + 9\frac{(\log 2)^2}{\pi^2} - 72\frac{\log 2\, \zeta'(2)}{\pi^4} - \frac{1}{2},$$

and $Q(u)$ is an oscillating function with mean value 0 whose order is

$$Q(u) = O\left(u^{\delta/2}\right), \quad \textit{where } \delta \textit{ is any number such that } \delta > \sup\left\{\Re(s) \;\middle|\; \zeta(s) = 0\right\}.$$

The Riemann hypothesis has just made an entry into the world of tries!

References

1. David J. Aldous, *Ultimate instability of exponential back-off protocol for acknowledgement-based transmission control of random access communication channels*, IEEE Transactions on Information Theory **33** (1987), no. 2, 219–223.
2. Noga Alon, Yossi Matias, and Mario Szegedy, *The space complexity of approximating the frequency moments*, Journal of Computer and System Sciences **58** (1999), no. 1, 137–147.
3. F. Avnaim, J.-D. Boissonnat, O. Devillers, F. P. Preparata, and M. Yvinec, *Evaluating signs of determinants using single-precision arithmetic*, Algorithmica **17** (1997), no. 2, 111–132.
4. M. Beeler, R. W. Gosper, and R. Schroeppel, *HAKMEM*, Memorandum 239, M.I.T., Artificial Intelligence Laboratory, February 1972, Available on the WorldWide Web at `http://www.inwap.com/pdp10/hbaker/hakmem/hakmem.html`.
5. Jon Bentley and Robert Sedgewick, *Fast algorithms for sorting and searching strings*, Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM Press, 1997.

[6]The sign question is equivalent to determining the orientation of a triangle in the plane.

6. Jon Louis Bentley, *Multidimensional binary search trees used for associative searching*, Communications of the ACM **18** (1975), no. 9, 509–517.
7. Julien Clément, Philippe Flajolet, and Brigitte Vallée, *The analysis of hybrid trie structures*, Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Algorithms (Philadelphia), SIAM Press, 1998, pp. 531–539.
8. Julien Clément, Philippe Flajolet, and Brigitte Vallée, *Dynamical sources in information theory: A general analysis of trie structures*, Algorithmica **29** (2001), no. 1/2, 307–369.
9. M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi, *Text compression using antidictionaries*, Automata, languages and programming (Prague, 1999), Lecture Notes in Computer Science, vol. 1644, Springer, Berlin, 1999, pp. 261–270.
10. N. G. de Bruijn, *Asymptotic methods in analysis*, Dover, 1981, A reprint of the third North Holland edition, 1970 (first edition, 1958).
11. Luc Devroye, *A probabilistic analysis of the height of tries and of the complexity of triesort*, Acta Informatica **21** (1984), 229–237.
12. Luc Devroye, *An analysis of random LC tries*, Random Structures & Algorithms **19** (2001), 359–375.
13. Luc Devroye and Wojtek Szpankowski, *Probabilistic behaviour of level compressed tries*, Random Structures & Algorithms **27** (2005), no. 2, 185–200.
14. Marianne Durand and Philippe Flajolet, LogLog *counting of large cardinalities*, Annual European Symposium on Algorithms (ESA03) (G. Di Battista and U. Zwick, eds.), Lecture Notes in Computer Science, vol. 2832, 2003, pp. 605–617.
15. Cristian Estan and George Varghese, *New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice*, ACM Transactions on Computer Systems **21** (2003), no. 3, 270–313.
16. R. Fagin, J. Nievergelt, N. Pippenger, and R. Strong, *Extendible hashing: A fast access method for dynamic files*, A.C.M. Transactions on Database Systems **4** (1979), 315–344.
17. Guy Fayolle, Philippe Flajolet, and Micha Hofri, *On a functional equation arising in the analysis of a protocol for a multiaccess broadcast channel*, Advances in Applied Probability **18** (1986), 441–472.
18. Guy Fayolle, Philippe Flajolet, Micha Hofri, and Philippe Jacquet, *Analysis of a stack algorithm for random access communication*, IEEE Transactions on Information Theory **IT-31** (1985), no. 2, 244–254, (Special Issue on Random Access Communication, J. Massey editor).
19. Julien Fayolle, *An average-case analysis of basic parameters of the suffix tree*, Mathematics and Computer Science III: Algorithms, Trees, Combinatorics and Probabilities (M. Drmota *et al.*, ed.), Trends in Mathematics, Birkhäuser Verlag, 2004, pp. 217–227.
20. James Allen Fill, Hosam M. Mahmoud, and Wojciech Szpankowski, *On the distribution for the duration of a randomized leader election algorithm*, The Annals of Applied Probability **6** (1996), no. 4, 1260–1283.
21. R. A. Finkel and J. L. Bentley, *Quad trees, a data structure for retrieval on composite keys*, Acta Informatica **4** (1974), 1–9.
22. Philippe Flajolet, *On the performance evaluation of extendible hashing and trie searching*, Acta Informatica **20** (1983), 345–369.
23. ———, *Counting by coin tossings*, Proceedings of ASIAN'04 (Ninth Asian Computing Science Conference) (M. Maher, ed.), Lecture Notes in Computer Science, vol. 3321, 2004, (Text of Opening Keynote Address.), pp. 1–12.
24. Philippe Flajolet, Xavier Gourdon, and Philippe Dumas, *Mellin transforms and asymptotics: Harmonic sums*, Theoretical Computer Science **144** (1995), no. 1–2, 3–58.
25. Philippe Flajolet and G. Nigel Martin, *Probabilistic counting algorithms for data base applications*, Journal of Computer and System Sciences **31** (1985), no. 2, 182–209.
26. Philippe Flajolet and Claude Puech, *Partial match retrieval of multidimensional data*, Journal of the ACM **33** (1986), no. 2, 371–407.
27. Philippe Flajolet, Mireille Régnier, and Dominique Sotteau, *Algebraic methods for trie statistics*, Annals of Discrete Mathematics **25** (1985), 145–188, In *Analysis and Design of Algorithms for Combinatorial Problems*, G. Ausiello and M. Lucertini Editors.
28. Philippe Flajolet and Bruce Richmond, *Generalized digital trees and their difference–differential equations*, Random Structures & Algorithms **3** (1992), no. 3, 305–320.
29. Philippe Flajolet and Robert Sedgewick, *Digital search trees revisited*, SIAM Journal on Computing **15** (1986), no. 3, 748–767.

30. ———, *Mellin transforms and asymptotics: finite differences and Rice's integrals*, Theoretical Computer Science **144** (1995), no. 1–2, 101–124.
31. Philippe Flajolet and Robert Sedgewick, *Analytic combinatorics*, October 2005, Chapters I–IX of a book to be published, 688p.+x, available electronically from P. Flajolet's home page.
32. Philippe Flajolet and Brigitte Vallée, *Continued fractions, comparison algorithms, and fine structure constants*, Constructive, Experimental, and Nonlinear Analysis (Providence) (Michel Théra, ed.), Canadian Mathematical Society Conference Proceedings, vol. 27, American Mathematical Society, 2000, pp. 53–82.
33. Ian P. Goulden and David M. Jackson, *Combinatorial enumeration*, John Wiley, New York, 1983.
34. L. J. Guibas and A. M. Odlyzko, *String overlaps, pattern matching, and nontransitive games*, Journal of Combinatorial Theory. Series A **30** (1981), no. 2, 183–208.
35. Micha Hofri, *Analysis of algorithms: Computational methods and mathematical tools*, Oxford University Press, 1995.
36. Hsien-Kuei Hwang, *On convergence rates in the central limit theorems for combinatorial structures*, European Journal of Combinatorics **19** (1998), no. 3, 329–343.
37. Philippe Jacquet and Mireille Régnier, *Trie partitioning process: Limiting distributions*, CAAP'86 (P. Franchi-Zanetacchi, ed.), Lecture Notes in Computer Science, vol. 214, 1986, Proceedings of the 11th Colloquium on Trees in Algebra and Programming, Nice France, March 1986., pp. 196–210.
38. Philippe Jacquet and Wojciech Szpankowski, *Autocorrelation on words and its applications: analysis of suffix trees by string-ruler approach*, Journal of Combinatorial Theory. Series A **66** (1994), no. 2, 237–269.
39. ———, *Asymptotic behavior of the Lempel-Ziv parsing scheme and digital search trees*, Theoretical Computer Science **144** (1995), no. 1–2, 161–197.
40. ———, *Analytical de-Poissonization and its applications*, Theoretical Computer Science **201** (1998), no. 1-2, 1–62.
41. Peter Kirschenhofer and Helmut Prodinger, *On some applications of formulæ of Ramanujan in the analysis of algorithms*, Mathematika **38** (1991), 14–33.
42. Peter Kirschenhofer, Helmut Prodinger, and Wojciech Szpankowski, *On the variance of the external path length in a symmetric digital trie*, Discrete Applied Mathematics **25** (1989), 129–143.
43. Donald E. Knuth, *The art of computer programming*, 2nd ed., vol. 3: Sorting and Searching, Addison-Wesley, 1998.
44. P. A. Larson, *Dynamic hashing*, BIT **18** (1978), 184–201.
45. Hosam M. Mahmoud, *Evolution of random search trees*, John Wiley, New York, 1992.
46. James L. Massey (ed.), *Special issue on random-access communications*, vol. IT-31, IEEE Transactions on Information Theory, no. 2, March 1985.
47. Peter Mathys and Philippe Flajolet, *Q–ary collision resolution algorithms in random access systems with free or blocked channel access*, IEEE Transactions on Information Theory **IT-31** (1985), no. 2, 217–243.
48. S. Nilsson and G. Karlsson, *IP–address lookup using LC tries*, IEEE Journal on Selected Areas in Communications **17** (1999), no. 6, 1083–1092.
49. Helmut Prodinger, *How to select a loser*, Discrete Mathematics **120** (1993), 149–159.
50. Mireille Régnier, *On the average height of trees in in digital search and dynamic hashing*, Information Processing Letters **13** (1982), 64–66.
51. ———, *Analysis of grid file algorithms*, BIT **25** (1985), 335–357.
52. Hanan Samet, *Applications of spatial data structures*, Addison–Wesley, 1990.
53. ———, *The design and analysis of spatial data structures*, Addison–Wesley, 1990.
54. Robert Sedgewick, *Algorithms in C, Parts 1–4*, third ed., Addison–Wesley, Reading, Mass., 1998.
55. Robert Sedgewick and Philippe Flajolet, *An introduction to the analysis of algorithms*, Addison-Wesley Publishing Company, 1996.
56. Richard P. Stanley, *Enumerative combinatorics*, vol. II, Cambridge University Press, 1998.
57. Wojciech Szpankowski, *Average-case analysis of algorithms on sequences*, John Wiley, New York, 2001.

58. Luis Trabb Pardo, *Set representation and set intersection*, Tech. report, Stanford University, 1978.
59. Brigitte Vallée, *Dynamical sources in information theory: Fundamental intervals and word prefixes*, Algorithmica **29** (2001), no. 1/2, 262–306.
60. Brigitte Vallée, *Euclidean dynamics*, October 2005, 69p. Submitted to *Discrete and Continuous Dynamical Systems*.
61. E. T. Whittaker and G. N. Watson, *A course of modern analysis*, fourth ed., Cambridge University Press, 1927, Reprinted 1973.
62. Herbert S. Wilf, *Generatingfunctionology*, Academic Press, 1990.
63. Andrew Chi Chih Yao, *A note on the analysis of extendible hashing*, Information Processing Letters **11** (1980), 84–86.

Algorithms Project, INRIA Rocquencourt, F-78153 Le Chesnay, France
E-mail address: Philippe.Flajolet AT inria.fr