Algebraic methods for trie statistics

Philippe Flajolet, Dominique Sotteau, Mireille Regnier

▶ **To cite this version:**

Philippe Flajolet, Dominique Sotteau, Mireille Regnier. Algebraic methods for trie statistics. [Research Report] RR-0298, INRIA. 1984. inria-00076259

HAL Id: inria-00076259
https://inria.hal.science/inria-00076259

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(3) 954 90 20

Rapports de Recherche

N° 298

ALGEBRAIC METHODS FOR TRIE STATISTICS

Philippe FLAJOLET
Mireille REGNIER
Dominique SOTTEAU

Mai 1984

ALGEBRAIC METHODS FOR TRIE STATISTICS

Philippe FLAJOLET

INRIA
Rocquencourt
78153-Le Chesnay (France)

Mireille REGNIER

INRIA
Rocquencourt
78153-Le Chesnay (France)

Dominique SOTTEAU

LRI
Universite Paris-Sud
91405-Orsay (France)

ABSTRACT

Tries are a data structure commonly used to represent sets of binary data. They also constitute a convenient way of modelling a number of algorithms to factorise polynomials, to implement communication protocols or to access files on disk. We present here a systematic method for analysing, in the average case, trie parameters through generating functions and conclude with several applications.

PAPIER RECUPERE ET RECYCLE

Résumé: Les arbres digitaux (ou *tries*) sont à la base de structures de données utilisées pour représenter des ensemble de données binaires. Ils sont également à la base de modèles intervenant dans la factorisation de polynomes, l'implémentation de protocoles de communication ou l'accès direct à des fichiers rangés sur disque. Nous présentons ici un ensemble systématique de méthodes permettant d'obtenir les valeurs moyennes de nombreux paramètres de ces arbres avec applications à l'analyse d'algorithmes.

Abstract: *Tries are a data structure commonly used to represent sets of binary data. They also constitute a convenient way of modelling a number of algorithms to factorise polynomials, to implement communication protocols or to access files on disk. We present here a systematic method for analysing, in the average case, trie parameters through generating functions and conclude with several applications.*

1. INTRODUCTION

Digital Searching methods comprise a variety of techniques used for sorting or retrieving data by taking advantage of their binary representations. In many cases, these techniques constitute an attractive alternative to comparison-based methods that relie on the existence of an ordering on the universe of data to be processed.

The *trie* structure is probably the most well-known amongst digital structures. It is a tree representation of sets of digital (*e.g.* binary) sequences that has been introduced by de la Briandais and Fredkin [9] † and bears analogies to binary search trees [11, 15, 17, 20, 1, 19] Operations like insert, delete, query, union, intersection... can be performed efficiently on this representation of sets. Tries also appear as a structure underlying *Radix Exchange Sort* (a digital analogue of Quicksort, [11],p. 131).

Tries have been proposed as an efficient way of maintaining *indexes* for externally stored files. When combined with hashing techniques (to ensure a uniform distribution of elements on which tries are built) they lead to *dynamic hashing* schemes like the Dynamic Hashing Method of [12] or Extendible Hashing [2].

Another use of tries is for *multi-dimensional* searching; the problem there is to retrieve records with several fields when only some of these fields are specified in a query. Under the form of *k-d-tries* (multi-dimensional tries) they constitute an elegant solution to the problem of *Partial Match* or *Secondary Key* retrieval. This variety of tries has been described by Rivest [18] who assigns their origin to Mc Creight. Used in conjunction with ideas taken from dynamic hashing techniques they lead to the so-called *Grid-File* algorithms [14] that have been proposed as a physical access method for files.

As a representation of binary sequences, tries are also of frequent occurrence in several applications. As an example, Huffman's algorithm may be viewed as a progressive construction of a trie. Situations where they appear to be a convenient model are for instance polynomial factorisation [8], communication protocols [3] or some simulation algorithms [7]. In many such cases, rather intricate parameters of binary sequences have simple formulations when expressed in terms of tries.

Our objective here is to describe a general set-up in which statistical analyses on tries can be conveniently performed. We show how to derive in a concise and synthetic way generating functions for average values of a large number of parameters of interest in the context of the analysis of algorithms. In this manner, we are able to present in a systematic manner (and sometimes extend) a number of analyses otherwise often obtained at some computational effort, and show that they can be reduced to a few simple paradigms.

† The name *trie* was coined by Fredkin apparently from a combination of tree and retrieval.

Our methodology consists in first establishing a few basic and easy to prove lemmas; these lemmas, given in Sections 2,3, relate under various probabilistic models some structural definitions of trie parameters to functional equations of some sort over generating functions of average values for which general resolution methods are also available. We thus have an *algebra of parameters* on tries which is mapped on an *algebra of generating functions*. In this way, the process of analysis is reduced to finding expressions for parameters of interest as combinations of a few buiding blocks for which mapping lemmas are available, and obtaining generating function expressions (whence expressions for average values) becomes an almost mechanical process. Most notably, the recourse to recurrences on average values which constitutes the basic technique usually employed is completely eliminated. This permits to analyzé in a simple way rather complex parameters of tries. The usefulness of this approach is demonstrated on several examples in Sections 4,5.

We do not address here the problem of the asymptotic evaluation of trie parameters (*cf* [11],p.131), but occasionally mention some of the estimates to make clear the implications of the analyses. The key method there consists in using Mellin transform techniques and some systematisation of its use is also possible.

1.1. Trie Representations of sets

Assume we want to represent data from a universe of *binary strings*, sometimes called *records* or *keys*, of a fixed length $s \geq 0$:

$$\mathbf{B}^{(s)} = \{0;1\}^s .$$

A subset $\omega \in \mathbf{B}^{(s)}$ decomposes into two subsets $\omega/0$, $\omega/1$ of $\mathbf{B}^{(s-1)}$ defined by:

$$\omega/0 = \{v \in \mathbf{B}^{(s-1)} \mid 0v \in \omega\}$$

$$\omega/1 = \{v \in \mathbf{B}^{(s-1)} \mid 1v \in \omega\} .$$

The definition of a trie is based on this decomposition.

Definition: *To a subset* $\omega \subset \mathbf{B}^{(s)}$, *we associate a tree trie* (ω) *as follows:*

(1) *If* $|\omega|=0$, *then trie* (ω) *is the empty tree.*[†]

(2) *If* $|\omega|=1$, *then trie* (ω) *is a tree formed with a unique leaf labelled* ω.

(3) *If* $|\omega| \geq 2$, *then trie* (ω) *is obtained by appending a root to the recursively defined subtrees trie* $(\omega/0)$ *and trie* $(\omega/1)$.

As an example, the trie associated to:

[†] We let $|\omega|$ denote the number of elements (cardinality) of ω

$$\omega = \{a, b, c, d, e, f\}$$

with

$$a = 01011,\ b = 01101,\ c = 10110,\ d = 11000,\ e = 11011,\ f = 11110$$

is displayed in Figure 1.

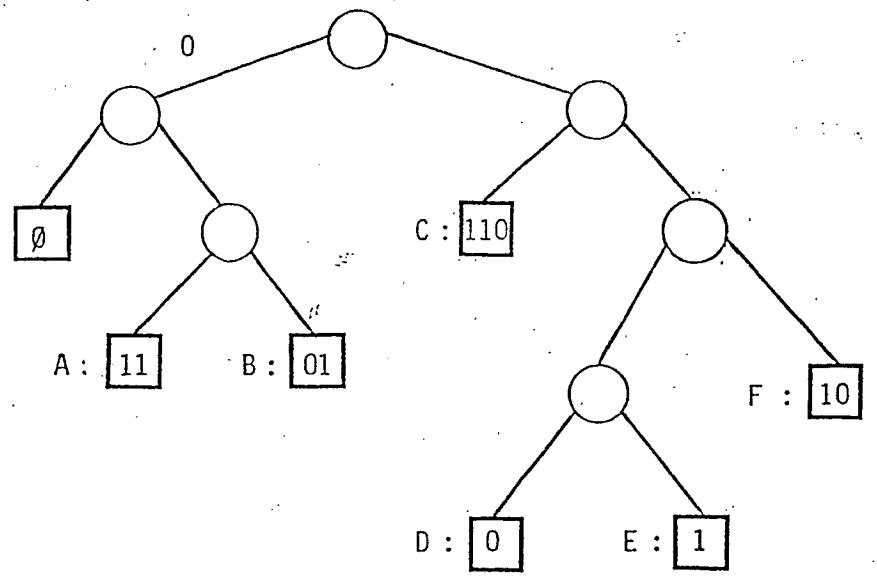

Figure 1: *A trie constructed on 6 binary sequences of length 5.*

The way of constructing the trie is clear from the definition: we recursively partition the set to be represented according to bits of highest weight until groups of cardinality at most one, represented by their remaining bits, have been individuated.

There is accordingly a simple way to recover the original set from the tree: simply read off all branches from the root to leaves, interpreting a left going edge as a 0 and a right going edge as a 1, appending for each branch the binary string stored at the leaf. For instance, in the above tree, going left-right-right (*i.e.* reading 011), we find leaf B that contains the information 01, and this corresponds to the key $b = 01101$.

1.2. Operations on tries

The recursive construction of a trie, usually represented in the form of a linked structure, closely mimics the definition given above. Once $trie(\omega)$ has been constructed, we can perform various operations[11, 15]:

query: to determine whether u is in ω, follow a branch taking directions corresponding to the successive letters of u until a leaf is encountered, then compare with the remaining bits.

insert: same as query; when a leaf is encountered, split it to obtain the new trie.

delete: a dual of the insertion procedure.

union-intersection: when sets are represented as tries, these operations can be implemented by means of the recursive definitions:

$$union(\psi,\vartheta) \equiv 0.union(\psi/0,\vartheta/0) \cup 1.union(\psi/1,\vartheta/1)$$

$$inter(\psi,\vartheta) \equiv 0.inter(\psi/0,\vartheta/0) \cup 1.inter(\psi/1,\vartheta/1)\,,$$

with adequate initial conditions.

The costs of these operations are largely determined by the number of pointer chains followed (or bits inspected). They usually admit *inductive definitions* of a simple form over the tree structure. A prototype is the size of the trie representation measured in the number of its internal nodes, which satisfies:†

$$in(\omega) = \text{if } |\omega|\leq 1 \text{ then } 0 \text{ else } 1+in(\omega/0)+in(\omega/1),$$

or equivalently

$$in(\omega) = 1+in(\omega/0)+in(\omega/1)-\delta_{|\omega|,0}-\delta_{|\omega|,1}.$$

together with the initial condition $in(\omega)=0$ if $|\omega|\leq 1$.

Our purpose in what follows is to describe a method for obtaining estimates of average values of such parameters when the number of elements in the set is a fixed integer n.

1.3. Trie Indexes

In order to save storage (reduce the number of pointers used), Sussenguth [21], followed by Knuth [11,ex. 6.3.20] proposed using a sequential storage discipline whenever reaching a subfile of b or less keys. The corresponding tree which we shall call a *b-trie* thus consists of:

† It is clear that such inductive definitions may be expressed either in terms of the left-subtree/right-subtree decomposition of trees or in terms of the equivalent decomposition of sets ω into $\omega/0$ and $\omega/1$.

(i) a skeleton tree formed with the internal nodes

(ii) leaves containing between 0 and b records.

This idea may be used to access files on some secondary storage device. The skeleton tree then becomes the *index* or *directory* and small subfiles are stored in *pages*, with b being the page capacity determined by physical characteristics of the device (in practice b ranges between a few tens and a few hundreds). When used in conjunction with hashing, the resulting algorithm is exactly Larson's Dynamic Hashing method.

Finally if the index is itself too large to fit in core, it may be paged as an array that represents its embedding into a perfect tree. This algorithm constitutes the Extendible Hashing method.

Naturally, the operations described in Section 1.2 are easily adapted to such representations of files. Notice, for instance, that a query with Dynamic Hashing requires only one disk probe, Extendible Hashing which can be used even for very large files requiring only two accesses. These strategies thus guarantee an almost *direct access* to external files, whence their practical interest.

2. THE UNIFORM MODEL

Our objective is to obtain estimates of expected values of a number of parameters on tries (size, path length, height, ...) as a function of the number n of elements on which the trie is built. In order to do so, we must first make precise what our probabilistic assumptions are. Following [11, 15], we retain two models.

1. *The finite key model:* Keys are to be of some fixed length s (s a non-negative integer). All sets of n elements are assumed to be equally likely. Since the number of these is:

$$b_n^{(s)} = \binom{2^s}{n}$$

the probability of each set is thus $(b_n^{(s)})^{-1}$.

2. *The infinite key model:* it is also occasionally called the Bernoulli model. Keys are assumed to be infinitely long strings of zeros and ones *i.e.* from the universe:

$$\mathbf{B}^{(\infty)} = \{0,1\}^{\infty} .$$

Alternatively, keys can be conceived of as real numbers over the real interval[0;1] (the correspondence between $\mathbf{B}^{(\infty)}$ and the real interval is bijective except for a set of measure 0). The infinite key model assumes that n keys are drawn uniformly and independently over the interval [0;1].

We consider *parameters* (also called *valuations*) on sets of strings. These are here usually parameters of the trie representation of sets.

Notations: *Let $v(\omega)$ be a parameter of sets $\omega \subset \mathbf{B}^{(s)}$, $s \leq \infty$ (or of trie (ω)). We define the quantities:*

$$v_n^{(s)} = \sum_{\substack{\omega \subset \mathbf{B}^{(s)} \\ |\omega| = n}} v(\omega) \quad \text{if } s < \infty ,$$

$$v_n^{(\infty)} = E[v(\omega) \mid \omega \subset \mathbf{B}^{(\infty)}, |\omega| = n]^{\dagger} .$$

In other words, $v_n^{(s)}$ represents the cumulated value of parameter v over all n-subsets of $\mathbf{B}^{(s)}$, and $v_n^{(\infty)}$ is the expectation of random variable v on a random n-subset of $\mathbf{B}^{(\infty)}$. (Notice that one can prove that any parameter that is polynomially bounded in the size of the trie has a finite expectation under the infinite key model).

Notation: *We define the ordinary generating function of the $v_n^{(s)}$ as:*

$$v^{(s)}(x) \equiv \sum_{n=0}^{2^s} v_n^{(s)} x^n ,$$

and the exponential generating function of the $v_n^{(\infty)}$ as:

$$v^{(\infty)}(x) = \sum_{n \geq 0} v_n^{(\infty)} \frac{x^n}{n!} .$$

In the sequel, we adhere to the convention of denoting parameters, corresponding cumulated values, expectations and generating functions by the same group of letters, as we have done above. We also make some use of the classical notation:

$$[x^n] f(x)$$

to represent the coefficient of x^n in the Taylor expansion of $f(x)$.

2.1. The Finite Model

In the finite model, the universe of keys is the set $\mathbf{B}^{(s)}$. The universe of sets is thus

$$\mathbf{P}^{(s)} = \{\omega \subset \mathbf{B}^{(s)}\} ,$$

and the finite model consists in assuming a uniform distribution on the elements of $\mathbf{P}^{(s)}$ of cardinality n.

---

† Here $E[X]$ denotes the expectation of the random variable X.

The definitions of additive and multiplicative valuations on tries can be translated directly into recurrence equations over generating functions as the following lemma shows:

Lemma 1 [*The additive-multiplicative translation lemma; finite model*]: *To the following schemas defining valuations on tries*

$$a(\omega) = \lambda b(\omega) \qquad (i)$$

$$a(\omega) = b(\omega)+c(\omega) \qquad (ii)$$

$$a(\omega) = b(\omega/0).c(\omega/1)\,, \qquad (iii)$$

there corresponds the following relations on generating functions:

$$a^{(s)}(x) = \lambda b^{(s)}(x) \qquad (i)$$

$$a^{(s)}(x) = b^{(s)}(x)+c^{(s)}(x) \qquad (ii)$$

$$a^{(s)}(x) = b^{(s-1)}(x).c^{(s-1)}(x)\,. \qquad (iii)$$

Proof: Relations (i) and (ii) follow from the linearity of generating functions and expectations. Relation (iii) can be established without using recurrences by writing:

$$a^{(s)}(x) = \sum_{\omega\in\mathbf{P}^{(s)}} a^{(s)}(\omega)x^{|\omega|} \qquad (2.1)$$

$$= \sum_{\omega\in\mathbf{P}^{(s)}} b(\omega/0)x^{|\omega/0|}c(\omega/1)x^{|\omega/1|} \qquad (2.2)$$

$$= \sum_{\omega_0\in\mathbf{P}^{(s-1)}} b(\omega_0)x^{|\omega_0|} \sum_{\omega_1\in\mathbf{P}^{(s-1)}} c(\omega_1)x^{|\omega_1|} \qquad (2.3)$$

$$= b^{(s-1)}(x)c^{(s-1)}(x). \qquad (2.4)$$

Note that the transition from (2.2) to (2.3) results from the standard isomorphisms:

$$\mathbf{B}^{(s)} \sim (0+1)\,\mathbf{B}^{(s-1)}\,,$$

$$\mathbf{P}^{(s)} \sim \mathbf{P}^{(s-1)}\times\mathbf{P}^{(s-1)}\,. \qquad \blacksquare$$

Lemma 2: [*The translation lemma for initial valuations; finite model*]: *The valuations:*

$$a(\omega) = 1 \qquad (i)$$

$$b(\omega) = \delta_{|\omega|,p} \qquad (ii)$$

$$c(\omega) = |\omega| \qquad (iii)$$

(δ denotes the Kronecker symbol) have corresponding generating functions:

$$a(x) = (1+x)^{2^s} \qquad (i)$$

$$b^{(s)}(x) = \binom{2^s}{p}x^p \qquad (ii)$$

$$c^{(s)}(x) = 2^s x(1+x)^{2^s-1} \qquad \textit{(iii)}$$

The above correspondences have a number of direct implications. For instance, if:

$$a(\omega) = b(\omega/0) \tag{2.5}$$

which we may rewrite (using 1_u to denote the valuation identically equal to 1) as

$$a(\omega) = b(\omega/0).1_{\omega/1} .$$

we get:

$$a^{(s)}(x) = (1+x)^{2^s-1}.b^{(s-1)}(x) \tag{2.6}$$

An important pattern in analyses is relative to parameters that are recursively defined over the tree structure. By (2.6), if a parameter v satisfies the inductive definition:

$$v(\omega) = v(\omega/0)+v(\omega/1)+w(\omega) , \tag{2.7}$$

then:

$$v^{(s)}(x) = 2(1+x)^{2^s-1}v^{(s-1)}(x)+w^{(s)}(x) . \tag{2.8}$$

Equations of the form (2.7) are solved by iterating (or unwinding) the recurrence, and one has trivially:

Lemma 3: [*The iteration lemma for the finite model*] *The solution to the recurrence:*

$$v^{(s)}(x) = \alpha_s(x)v^{(s-1)}(x)+\beta_s(x) ,$$

where α,β *are known and* $v^{(0)}(x)=\beta_0(x)$ *has the explicit solution:*

$$v^{(s)}(x) = \sum_{j=0}^{s}[\,\beta_j(x).\prod_{k=j+1}^{s}\alpha_j(x)\,] .$$

Application of this lemma to the special case of (2.8) results in the solution:

$$v^{(s)}(x) = \sum_{j=0}^{s}2^{s-j}w^{(j)}(x)(1+x)^{2^s-2^j} .$$

2.2. The Infinite Model

Our treatment of the infinite model closely follows what we have done in the preceding section. The universe of keys is now the set:

$$\mathbf{B}^{(\infty)}$$

and the universe of sets is:

$$\mathbf{P}^{(\infty)} = \{\omega \subset \mathbf{B}^{(\infty)} \mid \omega \text{ finite}\}.$$

The basic property here is that for a random set ω of n elements, the probability that the size of $\omega/0$ be equal to k and the size of $\omega/1$ be equal to $n-k$ is simply the Bernoulli probability:

$$\beta_{n,k} = \frac{1}{2^n}\binom{n}{k}.$$

We have;

Lemma 4: [*The additive-multiplicative translation lemma; infinite model*]: *If valuations on tries satisfy the relations:*

$$a(\omega) = \lambda b(\omega) \qquad (i)$$

$$a(\omega) = b(\omega) + c(\omega) \qquad (ii)$$

$$a(\omega) = b(\omega/0).c(\omega/1), \qquad (iii)$$

then the corresponding generating functions are related by:

$$a^{(\infty)}(x) = \lambda b^{(\infty)}(x) \qquad (i)$$

$$a^{(\infty)}(x) = b^{(\infty)}(x) + c^{(\infty)}(x) \qquad (ii)$$

$$a^{(\infty)}(x) = b^{(\infty)}\left(\frac{x}{2}\right).c^{(\infty)}\left(\frac{x}{2}\right). \qquad (iii)$$

Proof: Again (i) and (ii) are trivial. Relation (iii) is proven by:

$$a^{(\infty)}(x) \equiv \sum_{n\geq 0} a_n^{(\infty)} \frac{x^n}{n!} = \sum_{n,k\geq 0} \beta_{n,k} b_k c_{n-k} \frac{x^n}{n!}$$

$$= \sum_{n_1\geq 0} b_{n_1} \frac{x^{n_1}}{n_1!} \sum_{n_2\geq 0} c_{n_2} \frac{x^{n_2}}{n_2!}$$

$$= b^{(\infty)}\left(\frac{x}{2}\right).c^{(\infty)}\left(\frac{x}{2}\right).$$

The product form comes from the fact that when ω is a random n-subset of $\mathbf{B}^{(\infty)}$, then if $\omega/0$ is conditioned to be of cardinality k, it is a random k-subset of $\mathbf{B}^{(\infty)}$. ■

Lemma 5 [*The translation lemma for initial valuations; infinite model*]: *The valuations:*

$$a(\omega) = 1 \qquad (i)$$

$$b(\omega) = \delta_{|\omega|,p} \qquad (ii)$$

$$c(\omega) = |\omega| \qquad (iii)$$

have corresponding generating functions:

$$a^{(\infty)}(x) = e^x \qquad (i)$$

$$b^{(\infty)}(x) = \frac{x^p}{p!} \qquad \textit{(ii)}$$

$$c^{(\infty)}(x) = xe^x \ . \qquad \textit{(iii)}$$

We again have the important special cases corresponding to (2.5),(2.7). If

$$a(\omega) = b(\omega/0)$$

then

$$a^{(\infty)}(x) = e^{x/2}b^{(\infty)}(\frac{x}{2}) \ ; \qquad (2.9)$$

similarly, if $v(\omega)$ is a recursively defined parameter:

$$v(\omega) = v(\omega/0)+v(\omega/1)+w(\omega)$$

then

$$v^{(\infty)}(x) = 2e^{x/2}v^{(\infty)}(\frac{x}{2})+w^{(\infty)}(x) \ . \qquad (2.10)$$

Equations of the form (2.10) may be solved by iteration, and we have in analogy to Lemma 3:

Lemma 6 [*The iteration lemma; infinite case*]: *Let $\alpha(x)$ and $\beta(x)$ be two entire functions such that $\alpha(0)=c$ and $\beta(x)=O(x^d)$ as $x\to 0$. The difference equation:*

$$f(x) = \alpha(x)f(\frac{x}{2})+\beta(x)$$

where f is the unknown function and α, β satisfy the "contraction condition":

$$c\,2^{-d}<1 \ ,$$

with the initial conditions on $f(x)$:

$$f(0) = f'(0) = f''(0) = \cdots = f^{(d-1)}(0) = 0$$

has a unique entire solution given by:

$$f(x) = \sum_{j\geq 0}[\ \beta(\frac{x}{2^j})\prod_{k=0}^{j}\alpha(\frac{x}{2^k})\ ] \ .$$

Proof: Iterating the basic equation, one gets:

$$f(x) = \beta(x)+\alpha(x)f(\frac{x}{2})$$

$$= \beta(x)+\alpha(x)\beta(\frac{x}{2})+\alpha(x)\alpha(\frac{x}{2})f(\frac{x}{4})$$

$$= \cdots$$

The initial conditions together with the contraction condition ensure the convergence of the infinite sum that one obtains in the limit. ∎

We have again an important special case corresponding to (2.10) where two equivalent expressions can be derived.

Lemma 7 [*Iteration Lemma for the infinite model; special case*] *The difference equation:*

$$f(x) = ce^{x/2} f\left(\frac{x}{2}\right) + \beta(x)$$

admits provided the initial condition and the contraction conditions of Lemma 6 are satisfied the solution:

$$f(x) = \sum_{j \geq 0} c^j \beta\left(\frac{x}{2^j}\right) e^{x\left(1-\frac{1}{2^j}\right)} . \qquad (i)$$

Alternatively, the Taylor expansion of $f(x)$*:*

$$f(x) = \sum_{n \geq 0} f_n \frac{x^n}{n!}$$

can be obtained as:

$$f_n = \sum_k \left\{ {n \atop k} \right\} \frac{\beta_k^*}{1 - c\,2^{-k}} , \text{ where } \beta_k^* = k![x^k] e^{-x} \beta(x) . \qquad (ii)$$

Proof: Part (i) is a direct application of Lemma 6. For part (ii), we set $f^*(x) = e^{-x} f(x)$; $f^*(x)$ satisfies:

$$f^*(x) = c f^*\left(\frac{x}{2}\right) + \beta^*(x) . \qquad (2.11)$$

Identifying coefficients of x^n in (2.11) gives the relation $f_n^* = c\,2^{-n} f_n^* + \beta_n^*$. Relation (ii) then follows since the coefficients of $f(x)$ are convolutions of those of $f^*(x)$ by e^x. ■

Notice that in applications, the initial condition on $f(x)$ can be by-passed by subtracting from f adequate combinations of functions of the form: $x^m e^{px}$.

The reader may compare this approach to the treatment of recurrences occurring in the analysis of trie parameters via the use of *binomial transforms* in [11,ex. 5.2.2.36-38].

Notice, as a final remark in this section, the following relation between the finite and the infinite models [7]:

$$v^{(\infty)}(x) = \lim_{s \to \infty} v^{(s)}\left(\frac{x}{2^s}\right) ,$$

which is clear on the coefficients and explains many of the analogies between the two models.

3. ALTERNATIVE MODELS

The way taken in Section 2 which allows for a systematic *translation mechanism* from parameter definitions to generating functions may be extended to a diversity of models. Since the proof techniques in each case differ only little from what we have encountered, we only briefly sketch here the kind of results that can be attained.

3.1. Multiway Tries

In many applications, one may wish to take advantage of the decomposition of records into characters or bytes instead of bits. The resulting trees have then a branching degree corresponding to the cardinality of the alphabet which is an integer m, $m \geq 2$.

The definition of multiway tries mimics closely that of binary tries; if the alphabet is assimilated to the integer interval $[1..m]$, we consider the sets:

$$\mathbf{M}^{(s)} = [1..m]^s; \quad \mathbf{M}^{(\infty)} = [1..m]^\infty .$$

and tries are now defined recursively via the decomposition:

$$\omega \equiv 1.(\omega/1) \cup 2.(\omega/2) \cup \cdots \cup m.(\omega/m) ,$$

for any $\omega \subset \mathbf{M}^{(s)}$, with $\omega/j \subset \mathbf{M}^{(s-1)}$. Sum and product rules remain valid as before (with m-ary products if a valuation is a product of m valuations on subtrees).

In the finite case, the generating function describing the universe of all subsets of $\mathbf{M}^{(s)}$ becomes:

$$\sum_{\omega \subset \mathbf{M}^{(s)}} x^{|\omega|} = (1+x)^{m^s} . \tag{3.1}$$

In particular, if

$$v(\omega) = w(\omega/1) , \tag{3.2}$$

we have:

$$\begin{aligned} v^{(s)}(x) &= w^{(s-1)}(x)((1+x)^{m^{s-1}})^{m-1} \\ &\equiv w^{(s-1)}(x)(1+x)^{m^s - m^{s-1}} . \end{aligned} \tag{3.3}$$

In the infinite case, for instance, (3.2) leads to

$$\begin{aligned} v^{(\infty)}(x) &= w^{(\infty)}\left(\frac{x}{m}\right)\left(e^{x/m}\right)^{m-1} \\ &= w^{(\infty)}\left(\frac{x}{m}\right)e^{x\left(1-\frac{1}{m}\right)} . \end{aligned} \tag{3.4}$$

3.2. Biased Bits

To model more closely some applications, as for instance when tries are built out of textual data, one also considers *non-uniform probability distributions* on bits or characters of strings. We shall study here the infinite model only. Starting with the binary case, the model assumes that bits of keys are taken independently from some discrete distribution:

$$Pr(0\text{-}bit) = p; \quad Pr(1\text{-}bit) = q \equiv 1-p \ . \tag{3.5}$$

In other terms, for $u \in B^{(\infty)}$, we have:

$$Pr(\,|u_1 u_2 \cdots u_l|_0) = \binom{l}{k} p^k q^{n-k} \ .$$

($|u|_0$ denotes the number of zeros in u).

Additive properties of generating functions still hold. The main difference lies in multiplicative valuations, for which:

$$v(\omega) = w(\omega/0)t(\omega/1) \tag{3.6}$$

translates into:

$$v^{(\infty)}(x) = w^{(\infty)}(px).t^{(\infty)}(qx) \ . \tag{3.7}$$

This biased model also extends easily to m-ary tries, as we have been considering in Section 3.1. If the probability distribution on an m-ary alphabet is $(p_1, p_2, \cdots, p_m)$ with $\sum p_i = 1$, then

$$v(\omega) = w_1(\omega/1).w_2(\omega/2) \cdots w_m(\omega/m) \tag{3.8}$$

translates into:

$$v^{(\infty)}(x) = w_1^{(\infty)}(p_1 x).w_2^{(\infty)}(p_2 x). \cdots .w_m^{(\infty)}(p_m x) \ . \tag{3.9}$$

3.3. Allowing Repetitions

The definition of tries associated to *sets* of binary –or other– sequences can also be extended to *multisets* where elements may appear repeated several times. In order to do so, we only need to allow leaves to contain several elements that are identical. In practice the situation occurs for instance when constructing tries on a single field of composite records. Although records are usually all distinct, some values of a specified field are likely to occur several times (many people live in New-York City!).

Our universe of "files" has now become in the binary case the family $\mathbf{Q}^{(s)}$ of all multisets over $B^{(s)}$ which, using notations from formal language theory may be rewritten as:

$$\mathbf{Q}^{(s)} = \prod_{\alpha \in B^{(s)}} \alpha^* \tag{3.10}$$

with:

$$\alpha^* = \phi+\alpha+\alpha^2+\alpha^3+\cdots . \tag{3.11}$$

Taking as a measure of the size $|.|$ of a multiset the number of its elements counted with their multiplicities, the generating function that describes the universe of multisets $\mathbf{Q}^{(s)}$ is found to be:

$$\sum_{\omega\in\mathbf{Q}^{(s)}} z^{|\omega|} = \left(\frac{1}{1-x}\right)^{2^s} . \tag{3.12}$$

Notice in passing the formal analogy between definitions (3.10),(3.11) and equation (3.12).

Equation (3.12) is also consistent with the obvious counting result:

$$card\{\omega\in\mathbf{Q}^{(s)} \mid |\omega|=n\} = \binom{2^s+n-1}{n} . \tag{3.13}$$

Sum and product rules again apply and it is only in subtree valuations that the form (3.12) of the "universal" polynomial has to be taken into account.

For instance, if

$$v(\omega) = w(\omega/0) ,$$

then under this model:

$$v^{(s)}(x) = w^{(s-1)}(x)(1-x)^{-(2^{s-1})} .$$

3.4. Several Sets

Set-theoretic operations like union , intersection, ... take as arguments several sets. In order to analyse them in the average case, we should therefore enter the sizes of the arguments as parameters. Restricting here the discussion to the case of *two* sets, we consider valuations of the form:

$$v : \mathbf{P}^{(s)}\times\mathbf{P}^{(s)} \to \mathbf{R} .$$

The cumulated values of v:

$$v_{m,n}^{(s)} = \sum_{\substack{\xi,\eta\in\mathbf{P}^{(s)} \\ |\xi|=m,|\eta|=n}} v(\xi,\eta)$$

can be attained through the *bivariate* generating functions:

$$v^{(s)}(x,y) \equiv \sum_{m,n} v_{m,n}^{(s)} x^m y^n = \sum_{\xi,\eta\in\mathbf{P}^{(s)}} v(\xi,\eta) x^{|\xi|} y^{|\eta|} .$$

As we shall see in Section 4.2, a relation like:

$$v(\xi,\eta) = w(\xi/0,\eta/0)$$

translates into:

$$v^{(s)}(x,y) = (1+x)^{2^{s-1}}(1+y)^{2^{s-1}}w^{(s-1)}(x,y) .$$

Proof techniques are highly simplified if one uses the way taken in Section 2.1. This permits us in particular to give a detailed analysis of trie union and trie intersection.

3.5. The Poisson Model

The Poisson model has been used to obtain expressions that are sometimes easier to handle than corresponding expressions under the Bernoulli (infinite key) model. A typical example is the treatment of directory size in Extendible Hashing [2] that will be discussed in Section 4.3.

Under this model, the number N of elements on which a trie is constructed is a Poisson distributed random variable with average n (n being a parameter). The keys themselves are uniformly distributed over the real interval [0;1]. We have:

Lemma 8: *If a parameter v has under the Bernoulli model for n keys an expected value $v_n^{(\infty)}$, then under a Poisson model of parameter ν, its expectation satisfies:*

$$v_\nu^P = \sum_{k=0}^{\infty} e^{-\nu}\frac{\nu^k}{k!}v_k^{(\infty)} = e^{-\nu}v^{(\infty)}(\nu) .$$

Lemma 8 is a trivial consequence of the form of the Poisson probability distribution. It shows that the *values* at positive real points of *generating functions* under the Bernoulli model are directly related to the *Poisson expectations*. Thus Lemmas 4,5 translate *verbatim* into schemes that permit to determine from the shape of valuations the expected values of trie parameters under the Poisson model.

4. APPLICATIONS TO DIGITAL SEARCH

4.1. Simple Operations on Tries

We analyse here the storage efficiency of tries (and of some of their variants), as well as the time cost of a basic search. Our aim in this section is to provide a uniform framework for a number of results that are to be found in

[11]. (See in particular Sect. 5.2.2 and ex. 5.2.2.36-38; Sect 6.3 and ex. 6.3.20,31-34)

Multiway Tries

Our first analysis is relative to the storage occupation of multiway tries. In the case of an alphabet of cardinality m, assimilated to the integer interval $[1..m]$, the determinant parameter is the number of internal nodes of the trie. This parameter admits, as we have seen, the inductive definition:

$$in(\omega) = in(\omega/1)+in(\omega/2)+\cdots+in(\omega/m)+1-\delta_{|\omega|,0}-\delta_{|\omega|,1} \qquad (4.1)$$

for $\omega\subset\mathbf{B}^{(s)}$, with $s\geq 1$ or $s=\infty$. For $\omega\subset\mathbf{B}^{(0)}$, we have $in(\omega)=0$.

In the finite case, we find from Section 3.1 the recurrence relation:

$$in^{(s)}(x) = (1+x)^{m^s}-1-m^s x + m(1+x)^{m^s-m^{s-1}}in^{(s-1)}(x) \qquad (4.2)$$

for $s\geq 1$, with $in^{(0)}(x)=0$.In the infinite case, we get a difference equation:

$$in^{(\infty)}(x) = m\, e^{x(1-\frac{1}{m})}in^{(\infty)}(\frac{x}{m})+e^x-1-x, \qquad (4.3)$$

Equation (4.2) is readily solved by unwinding the recurrence, and we obtain:

$$in^{(s)}(x) = \sum_{j=1}^{s} m^{s-j}(1+x)^{m^s-m^j}[(1+x)^{m^j}-1-m^j x]\,. \qquad (4.4)$$

Taking Taylor coefficients of formula (4.4), we obtain the explicit form of the total number of nodes in all tries formed with n distinct keys of length s over an m-ary alphabet:

$$in_n^{(s)} = \sum_{j=1}^{s} m^{s-j}\begin{bmatrix} m^s \\ n \end{bmatrix}-m^{s-j}\begin{bmatrix} m^s-m^j \\ n \end{bmatrix}-m^s\begin{bmatrix} m^s-m^j \\ n-1 \end{bmatrix}$$

Solving under the infinite model is even simpler. By the methods of Section 2, we find for the exponential generating function of expected values of the number of internal nodes of tries the relation:

$$in^{(\infty)}(x) = \sum_{k\geq 0} m^k[e^x-e^{x(1-\frac{1}{m^k})}-\frac{x}{m^k}e^{x(1-\frac{1}{m^k})}]\,. \qquad (4.5)$$

Taking again Taylor coefficients in (4.5), we find for the expectation of in under the infinite key model the form:

$$in_n^{(\infty)} = \sum_{k\geq 0} m^k[1-(1-\frac{1}{m^k})^n-\frac{n}{m^k}(1-\frac{1}{m^k})^{n-1}]\,. \qquad (4.6)$$

Path length in multiway tries can be analysed in very similar terms. From the definition:

$$pl(\omega) = pl(\omega/1)+pl(\omega/2)+\cdots+pl(\omega/m)+|\omega|-\delta_{|\omega|,1}\,, \qquad (4.7)$$

we find the equations corresponding to the finite and infinite models:

$$pl^{(s)}(x) = m.pl^{(s-1)}(x)(1+x)^{m^s-m^{s-1}}+m^s x[(1+x)^{m^s-1}-1] \qquad (4.8)$$

$$pl^{(\infty)}(x) = m.e^{x(1-\frac{1}{m})}pl^{(\infty)}(\frac{x}{m})+x(e^x-1)\,. \qquad (4.9)$$

Solutions may be obtained as before, and summarising these analyses, we find [11]:

Theorem A: *The expectation of the number of nodes in an m-ary trie constructed with n keys is:*

$$in_n^{(s)} = \sum_{j=1}^{s} m^{s-j}\begin{Bmatrix} m^s \\ n \end{Bmatrix} - m^{s-j}\begin{Bmatrix} m^s-m^j \\ n \end{Bmatrix} - m^s\begin{Bmatrix} m^s-m^j \\ n-1 \end{Bmatrix},$$

$$in_n^{(\infty)} = \sum_{k\geq 0} m^k[1-(1-\frac{1}{m^k})^n - \frac{n}{m^k}(1-\frac{1}{m^k})^{n-1}]\,.$$

The expected value of path length is:

$$pl_n^{(s)} = m^s \sum_{j=1}^{s}[\begin{Bmatrix} m^s-1 \\ n-1 \end{Bmatrix} - \begin{Bmatrix} m^s-m^j \\ n-1 \end{Bmatrix}]$$

$$pl_n^{(\infty)} = n\sum_{j\geq 0}[1-(1-\frac{1}{m^j})^{n-1}]\,.$$

In particular, the expected cost, measured in the number of bit inspections, of a positive search is

$$\frac{1}{n\begin{Bmatrix} m^s \\ n \end{Bmatrix}}pl_n^{(s)},\ \frac{1}{n}pl_n^{(\infty)}.$$

Binary Representations of Multiway Tries

In the case of multiway tries, the asymptotic analysis of the number of nodes reveals that storing a file of n elements necessitates about $\frac{m}{\log m}n$ pointers, which may be quite expensive when m is large (many such pointers toward the low levels in the tree are likely to be null). For that reason, it may prove necessary to use a binary representation of tries, where each node is linked to its leftmost son and to its immediate right brother. The price to be paid is an increased time cost, since access to subtrees is now done in a sequential way. Such a representation is displayed in Figure 2.

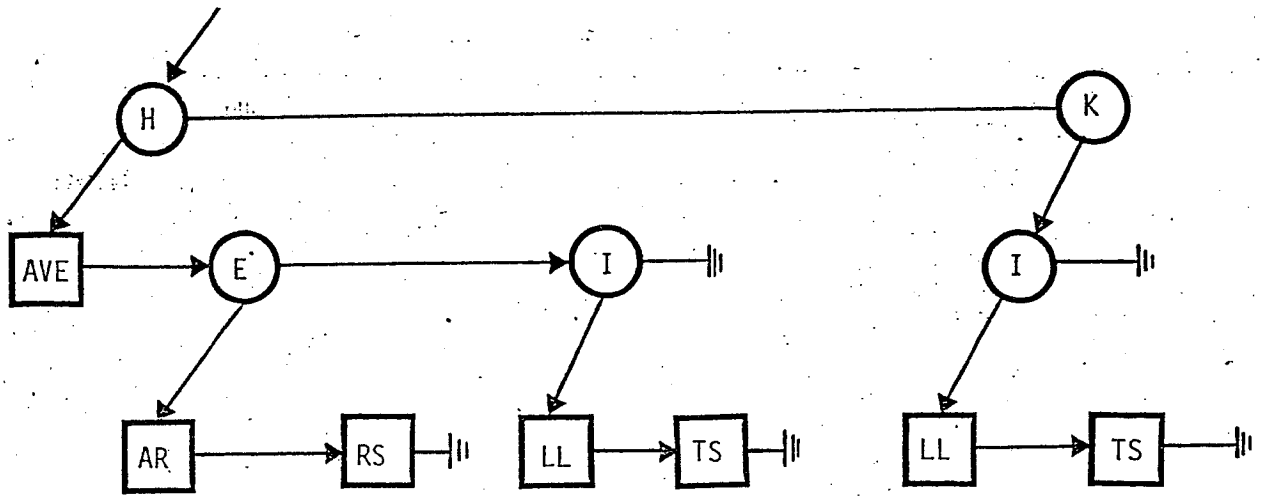

Figure 2: The binary tree encoding of a trie associated to the file {HAVE,HEAR,HERS,HILL,HITS,KILL,KITS}.

There are several conceivable implementations of this structure. In the one shown in Figure 2, an internal node of the original trie necessitates two pointers while external nodes only use one pointer. The problem of the storage occupation of this structure is thus solved by our previous analyses. We propose here to analyse the cost of a positive search under the infinite model.

Let $bp(\omega)$be the total number of pointers traversed when searching all keys in the binary tree representation of $trie(\omega)$. An inductive definition of this quantity is obtained by observing that the cost of accessing the subtree corresponding to ω/k is equal to 1 plus the number of non-empty subsets ω/j, for $0 \le j < k$. Hence:†

$$bp(\omega) = |\omega| + \sum_{k=1}^{m} [\, bp(\omega/k) + |\omega/k| \sum_{j=0}^{k-1} \chi(|\omega/j| \ge 1) \,] \qquad (4.10)$$

From there, the translation to generating functions is immediate, and, for the infinite model, we have:

$$bp^{(\infty)}(x) = x(e^x - 1) + me^{x(1-\frac{1}{m})} bp(\infty)(\frac{x}{m}) + \frac{m(m-1)}{2}(e^{\frac{x}{m}} - 1)\frac{x}{m} e^{\frac{x}{m}} (e^{\frac{x}{m}})^{m-2}$$

$$= x[\frac{m+1}{2} e^x - \frac{m-1}{2} e^{x(1-\frac{1}{m})} - 1] + me^{x(1-\frac{1}{m})} bp^{(\infty)}(\frac{x}{m}). \qquad (4.11)$$

† We let $\chi(P)$ denote the function whose value is 0 if P is false and 1 if P is true

Solving, we obtain:

Theorem B: *The average number of pointers of a binary tree representation of a multiway trie of n elements is:*

$$2in_n^{(\infty)}+n \ .$$

The average number of pointers followed in a positive search is $\frac{1}{n}bp_n^{(\infty)}$, *where:*

$$bp_n^{(\infty)} = n \sum_{k\geq 0} [\frac{m+1}{2} - \frac{m-1}{2}(1-\frac{1}{m^{k+1}})^{n-1} - (1-\frac{1}{m^k})^{n-1}] \ .$$

Patricia Trees

Patricia Trees are a compact representation of tries due to R. Morrison in which one-way branching is avoided by means of skip fields. Our description follows [11,pp.497-498]. We propose here to analyse under the infinite model the cost of both positive and negative queries.

The access cost of a leaf in a Patricia tree is therefore exactly the number of binary nodes traversed in the corresponding trie. Thus the cost of a positive query in a tree of n elements is $\frac{1}{n}$ times the "path length through binary nodes" in the underlying trie. This modified path length is defined by the recursion:

$$ppl(\omega) = \begin{cases} ppl(\omega/0)+ppl(\omega/1)+|\omega/0|+|\omega/1| & \text{if } |\omega/0|.|\omega/1|\neq 0 \\ ppl(\omega/0) & \text{if } |\omega/1|=0 \\ ppl(\omega/1) & \text{if } |\omega/0|=0 \end{cases}$$

This definition can be trivially rephrased as an additive-multiplicative combination of standard valuations. From there, we obtain the equation (terms correspond to those of the above definition):

$$ppl^{(\infty)}(x) = 2(e^{x/2}-1)ppl^{(\infty)}(\frac{x}{2}) + 2e^{x/2}\frac{x}{2}(e^{x/2}-1) + 2ppl^{(\infty)}(\frac{x}{2}) \ , \quad (4.12)$$

which, after simplification gives:

$$ppl^{(\infty)}(x) = 2e^{x/2}ppl^{(\infty)}(\frac{x}{2})+x(e^x-e^{x/2}) \ . \quad (4.13)$$

The case of a negative search leads to a difference equation of a new type. Let $pns(\omega)$ denote the expected cost of a random (*i.e.* negative with probability 1) search in the Patricia tree built on ω. This cost, again measured in the number of pointers followed, has the definition:

$$pns(\omega) = \begin{cases} \frac{1}{2}pns(\omega/0)+\frac{1}{2}pns(\omega/1)+1 & \text{if } |\omega/0|.|\omega/1|\neq 0 \\ pns(\omega/0) & \text{if } |\omega/1|=0 \\ pns(\omega/1) & \text{if } |\omega/0|=0 \end{cases}$$

This definition can be translated as before into:

$$pns^{(\infty)}(x) = (e^{x/2}-1)pns^{(\infty)}(\frac{x}{2})+(e^{x/2}-1)^2+2pns^{(\infty)}(\frac{x}{2}) .$$

$$= (1+e^{x/2})pns^{(\infty)}(\frac{x}{2})+(e^{x/2}-1)^2 . \qquad (4.14)$$

This equation can be solved by the iteration method described in Section 2:

$$pns^{(\infty)}(x) = \sum_{k\geq 0} (e^{\frac{x}{2^{k+1}}}-1)^2 \prod_{j=0}^{k-1}(1+e^{\frac{x}{2^j}}) ,$$

which using the identity:

$$(1+q)(1+q^2)(1+q^4)\cdots(1+q^{2^{j-1}}) = \frac{(1-q^{2^j})}{(1-q)}$$

yields the explicit form:

$$pns^{(\infty)}(x) = \sum_{k\geq 1} \frac{1-e^{\frac{x}{2^k}}}{1+e^{\frac{x}{2^k}}}(1-e^x) . \qquad (4.15)$$

To extract the Taylor coefficients of $pns^{(\infty)}(x)$ from (4.15), our route now follows that of Knuth [11,ex. 6.3.34.a]. We notice that for a Taylor series:

$$f(x) = \sum_{n\geq 1} f_n x^n ,$$

one obtains by expanding then aggregating the coefficients of x^n:

$$\sum_{k\geq 1} f(\frac{x}{2^k}) = \sum_{n\geq 1} f_n \frac{x^n}{2^n-1} .$$

Thus, from the standard definition of the Bernoulli numbers:

$$\sum_{n\geq 1} B_{n+1}\frac{x^n}{(n+1)!}(2^{n+1}-1) = \tanh(\frac{x}{2}) = \frac{e^x-1}{e^x+1} ,$$

we can expand the factor of $(1-e^x)$ in (4.15). The coefficients of the function pns are finally obtained by multiplying the expansion so obtained by the expansion of $(1-e^x)$. We have thus proved [11]:

Theorem C: *The expected costs of a positive and a negative search in a Patricia tree with n keys are:*

$$\frac{1}{n}pl_n^{(\infty)} - 1 ,$$

$$pns_n^{(\infty)} = \frac{2}{n+1} - 3 + 2\sum_{k\geq 1} \frac{1}{k+1}\left\{ {n \atop k} \right\} \frac{B_{k+1}}{2^k-1} .$$

4.2. Set Theoretic Operations on Tries

Union and intersection can be efficiently performed on trie representations of sets. As we saw in section 1.2, the algorithms are based on a simultaneous traversal of tries. We propose here to give a precise analysis of *trie intersection.* Our results improve on Trabb Pardo's [15] who only had an approximate analysis.

The parameters of the analysis are the cardinalities m,n of both sets whose intersection is to be computed *and* the cardinality of their intersection k. This way of proceeding follows Trabb Pardo's approach and is motivated by the fact that random sets tend to have very small intersections. Thus taking also the size of the intersection into account yields more informative results.

Our statistical model thus assumes all pairs of sets:

$$\mathbf{I}^{(s)}_{m,n,k} = \{(\xi,\eta) \mid \xi,\eta\subset\mathbf{B}^{(s)}, |\xi|=m, |\eta|=n, |\xi\cap\eta|=k\}$$

to be equally likely. By considering $\mathbf{I}^{(s)}_{m,n} = \bigcup_k \mathbf{I}^{(s)}_{m,n,k}$, *i.e.* summing our expressions over k, our results give the analysis of trie intersection performed on random sets of cardinalities m,n.

The intersection algorithm is obtained from the recursive definition of intersection:

$$inter(\xi,\eta) = 0.inter(\xi/0,\eta/0) \cup 1.inter(\xi/1,\eta/1).$$

with the initial conditions:

if $|\xi|=0$ *or* $|\eta|=0$ then report the empty set as intersection;

if $|\xi|=1$ then search for ξ in η;

if $|\eta|=1$ then search for η in ξ.

The *cost* of the intersection will be taken here to be the number of nodes traversed simultaneously in both tries. Extension to more complex cost measures is also possible by our methods. This cost admits an inductive definition that closely reflects the structure of the procedure:

$$ti(\xi,\eta) = 1+ti(\xi/0,\eta/0)+ti(\xi/1,\eta/1)-\chi(|\xi|\le 1 \text{ or } |\eta|\le 1).$$

The first problem we encounter is to determine the number of input configurations, *i.e.* the quantity

$$I^{(s)}_{m,n,k}=card(\mathbf{I}^{(s)}_{m,n,k}).$$

Let us define the function:

$$I^{(s)}(x,y,t) = \sum_{\xi,\eta\subset\mathbf{B}^{(s)}} x^{|\xi|}y^{|\eta|}t^{|\xi\cap\eta|};$$

we have:

$$I^{(s)}_{m,n,k} = [x^m y^n t^k] I^{(s)}(x,y,t).$$

To determine the quantity $I^{(s)}(x,y,t)$, we apply the techniques of Sections 2,3.

We write:

$$I^{(s)}(x,y,t) = \sum_{\xi,\eta\subset\mathbb{B}^{(s)}} x^{|\xi/0|+|\xi/1|}y^{|\eta/0|+|\eta/1|}t^{|\xi/0\cap\eta/0|+|\xi/1\cap\eta/1|}$$

$$= \sum_{\xi_0,\eta_0,\xi_1,\eta_1\subset\mathbb{B}^{(s-1)}} x^{|\xi_0|+|\xi_1|}y^{|\eta_0|+|\eta_1|}t^{|\xi_0\cap\eta_0|+|\xi_1\cap\eta_1|}$$

$$= [I^{(s-1)}(x,y,t)]^2 .$$

Since we have the initial value:

$$I^{(0)}(x,y,t) = (1+x+y+txy) ,$$

we get:

$$I^{(s)}(x,y,t) = (1+x+y+txy)^{2^s} . \tag{4.16}$$

The same process can be applied to multiplicative valuations over subtrees. If

$$v(\xi,\eta) = w(\xi/0,\eta/0) \tag{4.17}$$

we find:

$$v^{(s)}(x,y,t) = (1+x+y+txy)^{2^s-1}w^{(s-1)}(x,y,t) . \tag{4.18}$$

Applying this paradigm (4.17)-(4.18) to the equation defining ti, we have:

$$ti^{(s)}(x,y,t) = (1+x+y+txy)^{2^s-1}ti^{(s-1)}(x,y,t)$$

$$+(1+x+y+txy)^{2^s}-X^{(s)}(x,y,t) , \tag{4.19}$$

where

$$X^{(s)}(x,y,t) = \sum_{\substack{\xi,\eta\subset\mathbb{B}^{(s)}\\ |\xi|\le 1,|\eta|\le 1}} x^{|\xi|}y^{|\eta|}t^{|\xi\cap\eta|} . \tag{4.20}$$

Determining $X^{(s)}(x,y,t)$ is routinely obtained by considering all possible cases, and we find:

$$X^{(s)}(x,y,t) = 2^s(1+x+y+txy)[(1+x)^{2^s-1}+(1+y)^{2^s-1}-1]$$

$$-(2^s-1)[(1+x)^{2^s}+(1+y)^{2^s}-1-2^sxy] . \tag{4.21}$$

Whence solving by means of Lemma 3 the explicit form:

$$ti^{(s)}(x,y,t) = (2^s-1)(1+x+y+txy)^{2^s}-\sum_{j=1}^{s}2^{s-j}(1+x+y+txy)^{2^s-2^j}X^{(j)}(x,y,t) . \tag{4.22}$$

There now remains the task of extracting coefficients of the polynomials that appear in equation (4.22). To that purpose, we define:

$$I_{m,n,k}[\alpha,\gamma] = [x^my^nt^k](1+x)^\alpha(1+x+y+txy)^\gamma \tag{4.23}$$

$$J_{m,n,k}[\beta,\gamma] = [x^my^nt^k](1+y)^\beta(1+x+y+txy)^\gamma .$$

Expanding first in t then in x and y the polynomials of (4.23), we immediately find:

$$I_{m,n,k}[\alpha,\gamma] = \binom{\gamma}{k}\binom{\gamma-k}{n-k}\binom{\gamma+\alpha-n}{m-k} \tag{4.23}$$

$$J_{m,n,k}[\beta,\gamma] = \binom{\gamma}{k}\binom{\gamma-k}{m-k}\binom{\gamma+\beta-m}{n-k} \tag{4.24}$$

The quantities $I_{m,n,k}[\alpha,\gamma]$ generalise the $I^{(s)}_{m,n,k}$. In particular, from (4.23), we have:

$$I^{(s)}_{m,n,k} = I_{m,n,k}[0,2^s] = J_{m,n,k}[0,2^s] = \binom{2^s}{k}\binom{2^s-k}{m-k}\binom{2^s-m}{n-k}, \tag{4.25}$$

a fact which could have been deducted by direct reasoning.

Using (4.23),(4.24) in (4.22), we finally obtain:

Theorem D: *The cumulated cost of the trie intersection algorithm applied to two sets of cardinalities m and n whose intersection has cardinality k is:*

$$ti^{(s)}_{m,n,k} = (2^s-1)I_{m,n,k}[0,2^s]$$

$$-2^s\sum_{j=1}^{s}\{\, I_{m,n,k}[2^j-1,2^s-2^j+1]+J_{m,n,k}[2^j-1,2^s-2^j+1]$$

$$-I_{m,n,k}[0,2^s-2^j+1]+(2^j-1)I_{m-1,n-1,k}[0,2^s-2^j]\,\}$$

$$+\sum_{j=1}^{s}(2^s-2^j)\,\{\, I_{m,n,k}[2^s,2^s-2^j]+J_{m,n,k}[2^s,2^s-2^j]-I_{m,n,k}[0,2^s-2^j]\,\}.$$

The expected cost, assuming a uniform distribution over $\mathbf{I}^{(s)}_{m,n,k}$ *is:*

$$\frac{1}{I_{m,n,k}[0,2^s]}ti^{(s)}_{m,n,k}.$$

One could analyse in a similar way the cost of trie union as well as take into account the cost of other operations (pointer traversals, bit inspections, tests...).

4.3. Trie Indexes

We propose to analyse here the main parameters of trie indexes, when the keys are infinite.

We shall first evaluate the number of pages necessary to store records of the file; this quantity satisfies the recurrence:

$$page(\omega) = page(\omega/0)+page(\omega/1)-\chi(|\omega|\le b).$$

with the initial conditions: $page(\omega)=1$ if $|\omega|\le b$. This is an additive valuation on tries. It is amenable to the techniques previously described, and we find for the corresponding generating function the difference equation:

$$page^{(\infty)}(x) = 2e^{x/2}page^{(\infty)}(\frac{x}{2})+e^x-e_b(x), \tag{4.26}$$

where

$$e_b(x) = \sum_{j=0}^{b} \frac{x^j}{j!}.$$

Equation (4.26) does not satisfy the contraction condition of Lemma 6. However, the auxiliary function $\psi(x) = page^{(\infty)}(x) - e^x$ does. Function ψ is defined by a variant of equation (4.26):

$$\psi(x) = 2e^{x/2}\psi(\frac{x}{2}) + e^x - e_b(x)\ ,$$

which can be solved by iteration leading to an explicit form of $page^{(\infty)}$:

$$page^{(\infty)}(x) = e^x + \sum_{k\geq 0} 2^k e^{x(1-2^{-k})}(e^{x2^{-k}} - e_b(x2^{-k})) \qquad (4.27)$$

from which Taylor coefficients can be extracted.

The distribution of page occupation may be analysed in a similar manner. Let $page_p(\omega)$denote the number of pages containing p elements. This quantity can be defined by:

$$page_p(\omega) = [page_p(\omega/0) + page_p(\omega/1)]\chi(|\omega| > b) + \delta_{|\omega|,p}\ . \qquad (4.28)$$

Under this form, equation (4.28) does not fit directly into the schemes we have previously introduced. It can however be brought under the standard form of an additive-multiplicative valuation if we rewrite it as:

$$page_p(\omega) = page_p(\omega/0) + page_p(\omega/1) \qquad (4.29)$$
$$+\chi(|\omega| = p) - \chi(|\omega/0| = p).\chi(|\omega/1| \leq b-p) - \chi(|\omega/1| = p).\chi(|\omega/0| \leq b-p)\ .$$

From (4.29) follows the equation:

$$page_p^{(\infty)}(x) = 2e^{x/2} page_p^{(\infty)}(\frac{x}{2}) + \frac{x^p}{p!} - 2^{1-p}\frac{x^p}{p!}e_{b-p}(\frac{x}{2})\ . \qquad (4.30)$$

This equation can be solved as before and one finds:

Theorem E: *The expected number of pages in a binary trie index, when the page capacity is b is given by:*

$$page_n^{(\infty)} = 1 + \sum_{k=0}^{\infty} \qquad (4.31)$$
$$2^k[1 - (1-\frac{1}{2^k})^n - \binom{n}{1}(\frac{1}{2^k})(1-\frac{1}{2^k})^{n-1} \cdots - \binom{n}{b}(\frac{1}{2^k})^b(1-\frac{1}{2^k})^{n-b}]$$

The expected number of pages containing p elements is:

$$page_{p,n}^{(\infty)} = \delta_{n,p} + \binom{n}{p}\sum_{k\geq 1} 2^{(1-p)k}[(1-\frac{1}{2^k})^{n-p} - \qquad (4.32)$$
$$(1-\frac{1}{2^{k+1}})^{n-p} - \binom{n-p}{1}(\frac{1}{2^k})(1-\frac{1}{2^{k+1}})^{n-p-1} - \cdots - \binom{n-p}{b-p}(\frac{1}{2^k})^{b-p}(1-\frac{1}{2^{k+1}})^{n-b}]\ .$$

We now proceed with the evaluation of the depth of b-tries which is related to the size of the implementation of the index in extendible hashing.To that purpose, we introduce the *characteristic variables* $p_k^{(\infty)}(\omega)$, defined by:

$$p_k^{(\infty)}(\omega) = \begin{cases} 1 \text{ if the height of } \omega \text{ is at most } k \\ 0 \text{ otherwise .} \end{cases}$$

These quantities are purely multiplicative valuations that satisfy:

$$p_k^{(\infty)}(\omega) = p_{k-1}^{(\infty)}(\omega/0)p_{k-1}^{(\infty)}(\omega/1) . \tag{4.33}$$

Furthermore, the *expectation* $p_{k,n}^{(\infty)}$ of $p_k^{(\infty)}(\omega)$ over all ω of cardinality n is exactly the *probability* for an n-set of strings to have an associated trie of height at most k.

Thus these probabilities, under the Bernoulli model, have generating functions that satisfy:

$$\begin{aligned} p_k^{(\infty)}(x) &= p_{k-1}^{(\infty)}(\frac{x}{2})^2 \qquad (4.34) \\ &= p_0^{(\infty)}(x2^{-k})^{2^k} = e_b(x2^{-k})^{2^k}. \end{aligned}$$

With the general relations between the Bernoulli and Poisson models that have been given in Lemma 7, we thus find the values of these probabilities under the Poisson model to be:

$$p_k^P(n) = e^{-n}e_b(n2^{-k})^{2^k}. \tag{4.35}$$

From (4.34), (4.35), we have access to the expected height of b-tries, and we find:

Theorem F: *The average depth of a b-trie is under the Bernoulli model:*

$$de_n^{(\infty)} = \sum_{k\geq 0}(1-n![x^n]e_b(x2^{-k})^{2^k}),$$

and under the Poisson model of parameter n:

$$de_n^P = \sum_{k\geq 0}(1-e^{-n}e_b((n2^{-k})^{2^k}).$$

The asymptotic analysis of the results of Theorems E,F reveals that the expected number of pages fluctuates around

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

which corresponds to a load factor of the pages close to log2 [11, 12, 2]. The depth of a b-trie under either the Bernoulli or the Poisson model satisfies:

$$de_n = (1+\frac{1}{b})\log_2 n + O(1) .$$

In Extendible Hashing, the trie is embedded in a perfect tree, and represented as an array; the size of this array is exactly 2 raised to a power which is the

height of the trie. It has been analysed under both models by [8, 16] who show that it has a *non-linear growth* and fluctuates around:

$$\frac{[(b+1)!]^{-\frac{1}{b}}}{\log 2}\Gamma(1-\frac{1}{b})n^{1+\frac{1}{b}}.$$

5. MISCELLANEOUS APPLICATIONS

5.1. Multidimensional Search

Multi-dimensional tries or $k-d$-tries (in dimension k) are used to store and retrieve records from a k-dimensional universe. Throughout this section, we assume that each field is an infinitely long binary string. The universe of records is then simply:

$$(\mathbf{B}^{(\infty)})^k$$

There is a very natural mapping from $(\mathbf{B}^{(\infty)})^k$ to $\mathbf{B}^{(\infty)}$. To an element $\vec{u} \subset (\mathbf{B}^{(\infty)})^k$, $\vec{u} = (u^{[1]}, u^{[2]}, \cdots, u^{[k]})$, we associate the string:

$$v = sh(\vec{u}) = u_1^{[1]} u_1^{[2]} \cdots u_1^{[k]} u_2^{[1]} u_2^{[2]} \cdots u_2^{[k]} u_3^{[1]} u_3^{[2]} \cdots u_3^{[k]} \cdots ,$$

In other words, v is obtained from $\vec{u}$ by regularly shuffling the bits of the components of $\vec{u}$.

Given a finite set $\omega \in (\mathbf{B}^{(\infty)})^k$, the trie built on $sh(\omega)$ is by definition the $k-d$-trie associated to ω.

The use of $k-d$-tries should be clear. To retrieve a record when all its fields are known, follow a path in the tree guided by the bits of fields in a manner consistent with the definition of the shuffle function.

The interest of $k-d$-tries is to allow partial match retrieval to be performed often with reasonable efficiency. To retrieve a record with *some* of its fields specified, again follow a path in the tree guided by the bits specified in the query. For bits corresponding to unspecified fields, proceed with a search in both subtrees. This method is described in [18]. A *partial match search* is thus a succession of one-way and two-way branching dependent upon the *specification pattern* of the query.

Definition: *A specification pattern is a word of length k over the alphabet $\{S, *\}$. To any partial match query there corresponds a unique specification pattern obtained by associating an "S" to a specified field, and a "*" to an unspecified field in the query.*

Our purpose here is only to illustrate by means of an example extracted from [5] how previously discussed methods may be used to analyse the cost of partial match queries in k-d-tries. One has:

Theorem G: *The expected cost, measured in the number of internal nodes traversed, of a partial match query on a file of size n, represented as a k-d-trie with specification pattern π is:*

$$c_{\pi,n} = \sum_{l=0}^{k-1} [\, \delta_1 \delta_2 \cdots \delta_l \sum_{j\geq 0} 2^{j(k-s)} \tau_{j,l}(n) \,]$$

where $\delta_j = 1$ if $\pi_j = "S"$, $\delta_j = 2$ if $\pi_j = "*"$, and:

$$\tau_{j,l}(n) = 1-(1-\frac{1}{2^{kj+l}})^n - \frac{n}{2^{kj+l}}(1-\frac{1}{2^{kj+l}})^{n-1} ,$$

for j,l not both 0; $\tau_{0,0}(n)=0$.

Proof: Let $c_\pi(\omega)$ denote the expected cost of a *random* search (*i.e.* specified fields according to π in the query are drawn uniformly) in the *fixed* tree *trie*(ω). Letting $\pi^{<1>}, \pi^{<2>}, \cdots$ denote the successive left circular shifts of the word π; in particular $\pi^{<k>}=\pi^{<0>}=\pi$, $\pi^{<k+1>}=\pi^{<1>}, \cdots$. From the structure of the recursive search procedure, we find the recurrence:

$$c_{\pi^{<j>}}(\omega) = \frac{\delta_j}{2} c_{\pi^{<j+1>}}(\omega/0) + \frac{\delta_j}{2} c_{\pi^{<j+1>}}(\omega/1) + 1 - \delta_{|\omega|,0} - \delta_{|\omega|,1} , \qquad (5.1)$$

which is a direct reflection of the cyclical changes of the discriminating fields in a k-d-trie.

Equations (5.1) translate into a set of difference equations for corresponding generating functions:

$$c_{\pi^{<j>}}(x) = \delta_j c_{\pi^{<j+1>}}(\frac{x}{2}) + e^x - 1 - x , \quad j = 0 \cdots k-1 . \qquad (5.2)$$

The system of k equations (5.2) reduces by successive elimination of $c_{\pi^{<1>}}, c_{\pi^{<2>}}, \cdots$ with $c_{\pi^{<k>}}(x) = c_\pi(x)$ to:

$$c_\pi(x) = 2^{k-s} c_\pi(\frac{x}{2^k}) + a_\pi(x) , \qquad (5.3)$$

where s is the number of specified attributes in the query and $a_\pi(x)$ is a combination of exponential functions.

System (5.3) is then solved by iteration as explained in Section 2, and Theorem G is then proved. ■

The result of the asymptotic analysis of our previous estimates is that the average cost of a partial match query with s out of k attributes specified in a file of size n is

$$O(n^{1-\frac{s}{k}}) .$$

This result is to be compared to the corresponding cost of a search in a k-d-tree (the multidimensional analogue of binary search trees) which is:

$$O(n^{1-\frac{s}{k}+\vartheta(\frac{s}{k})})$$

for a non-zero function $\vartheta(\frac{s}{k})$. These analyses are presented in detail in [5]. They give support to Nievergelt' s claim [13] that in many situations digital structures tend to be more efficient than comparison based structures.

5.2. Biased Tries and Polynomial Factorisation

Tries constructed from a biased distribution may be analysed by the methods of Section 3.2. Additive parameters are no difficulty, especially if one uses methods of Lemma 7 (ii). See *e.g.* [11, ex. 5.5.2.53].

A generating function of a rather surprising form occurs in the analysis of the expected height of biased tries. Defining as in Section 4.3 the quantities $p_{n,k}$ to be the probability that a simple trie built on a binary alphabet has height at most k, we readily find from Section 3.2 and decomposition (4.33) the relation on the exponential generating function of the $p_{n,k}$:

$$p_k(x) = p_{k-1}(px).p_{k-1}(qx)\,, \tag{5.4}$$

which, combined to the initial condition:

$$p_0(x) = 1+x$$

shows that

$$p_k(x) = \prod_{j\geq 0}(1+p^j q^{k-j})^{\binom{k}{j}}.$$

Whence:

Theorem H: *The probability for a biased trie formed with n strings to have height at most k is:*

$$n![x^n]\prod_{j\geq 0}(1+p^j q^{k-j})^{\binom{k}{j}}.$$

In [8], the authors use a saddle point argument to show that the corresponding expected height is of order;

$$\frac{2\log n}{\log\,(p^2+q^2)^{-1}}.$$

This result is useful in the analysis of some refinements of Berlekamp's polynomial factorisation algorithms based on the construction of idempotents.

6. CONCLUSIONS

It should be clear by now that a large number of statistics on binary sequences can be analysed rather simply by the methods which we have previously developed. Other applications that we do not have space to describe here include the Probabilistic Counting algorithm of [4] or the analysis by [6] of the von Neumann-Knuth-Yao algorithm for generating an exponentially distributed variate.

More generally, consideration of general relationships between structural definitions of algorithms or combinatorial parameters and the corresponding equations satisfied by generating functions seems worthy of attention in the field of analysis of algorithms. It extends the approach of some recent works in combinatorial analysis by Foata and Schutzenberger, Rota, Jackson and Goulden [10]. That it is not restricted to tries will only be illustrated by means of a few simple examples.

Assume two valuations on binary trees are related by:

$$v(T) = w(left(T)) .$$

We can then set up various translation lemmas for corresponding generating functions of average values under several statistical models of tree formation. We cite:

(i): in the case of binary tries (as we have been considering), for exponential generating functions:

$$v(z) = e^{z/2} w(\frac{z}{2}) ;$$

(ii): in the case of binary search trees, for ordinary generating functions:

$$v(z) = \int_0^z w(t) \frac{dt}{1-t} ;$$

(iii): in the case of digital trees [Kn73], for exponential generating functions:

$$v(z) = \int_0^z e^{t/2} w(\frac{t}{2}) dt ;$$

(iv): in the case of planar binary trees, for generating functions of cumulated values:

$$v(z) = \frac{1-\sqrt{1-4z}}{2} w(z) .$$

Thus, for entire classes of valuations we can characterise the systems of functional equations that arise. This characterisation calls for:

-exact resolution methods; this is provided by iteration mechanisms in case (i), by the resolution of differential systems in (ii) and by the resolution of algebraic systems in case (iv);

-methods for pulling out (if possible directly from equations satisfied by generating functions) the asymptotic behaviour of coefficients; the available tools are: (i) Mellin transform techniques, (ii) contour integration in conjunction with the study of singular differential systems, (iii) Newton series and corresponding integral formulae, (iv) the Darboux method for relating singularities of functions to asymptotics of their Taylor coefficients.

References

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, Mass. (1983).
2. R. Fagin, J. Nievergelt, N. Pippenger, and R. Strong, "Extendible Hashing: A Fast Access Method for Dynamic Files," *A.C.M. T.O.D.S* **4** pp. 315-344 (1979).
3. G. Fayolle, P. Flajolet, and M. Hofri, "On a Functional Equation Arising in the Analysis of a Protocol for a Multiaccess Broadcast Channel," INRIA Res. Rep. 131 (April 1982).
4. P. Flajolet and P. N. Martin, "Probabilistic Counting," *Proc. 24th I.E.E.E. Symp. on F.O.C.S.*, (1983).
5. P. Flajolet and C. Puech, "Tree Structures for Partial Match Retrieval," *Proc. 24th I.E.E.E. Symp. on F.O.C.S.*, (November 1983).
6. P. Flajolet and N. Saheb, "Digital Search Trees and the Complexity of Generating an Exponentially Distributed Variate," in *Proc. Coll. on Trees in Algebra and Programming*, Lecture Notes in Comp. Sc., L' Aquila (1983). to appear
7. P. Flajolet and D. Sotteau, "A Recursive Partitioning Process of Computer Science," *Proc. II World Conference on Mathematics at the service of Man*, pp. 25-30 (1982). invited lecture
8. P. Flajolet and J-M. Steyaert, "A Branching Process Arising in Dynamic Hashing, Trie Searching and Polynomial Factorization," *Proceeding ICALP 82, Lect. Notes in Comp. Sc. 140*, pp. 239-251 (1982).
9. E. Fredkin, "Trie Memory," *Comm. ACM* **3** pp. 490-499 (1960).
10. I. P. Goulden and D. M. Jackson, *Combinatorial Enumeration*, John Wiley, New York (1983).
11. D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, Addison-Wesley, Reading, Mass. (1973).
12. P. A. Larson, "Dynamic Hashing," *BIT* **18** pp. 184-201 (1978).
13. J. Nievergelt, "Trees as Data and File Structures," *Proceedings CAAP 81, Lect. Notes in Comp. Sc. 112*, pp. 35-45 (1981).

14. J. Nievergelt, H. Hinterberger, and K. Sevcik, "The Grid File: an Adaptable Symmetric Multi-key File," Report 46, E.T.H. (1981).

15. L. Trabb Pardo, "Set Representation and Set Intersection," Report STAN-CS-78-681, Stanford University (1978).

16. M. Regnier, "Evaluation des performances du hachage dynamique," Thesis, Universite Paris-Sud (1983).

17. E. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms*, Prentice-Hall, Englewood Cliffs, N.J. (1977).

18. R. L. Rivest, "Partial Match Retrieval Algorithms," *S.I.A.M. J. on Comp.* **5** pp. 19-50 (1976).

19. R. Sedgewick, *Algorithms*, Addison-Wesley, Reading, Mass. (1983).

20. T. A. Standish, *Data Structure Techniques*, Addison-Wesley, Reading, Mass. (1980).

21. E. H. Sussenguth, "Use of Tree Structures for Processing Files," *Comm. A.C.M.* **6**(5) pp. 272-279 (1963).

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique