The Analysis of Hybrid Trie Structures*

Julien Clément[1,2] Philippe Flajolet[1] Brigitte Vallée[2]

[1] Algorithms Project, INRIA-Rocquencourt, F-78150 Le Chesnay (France)

[2] GREYC, Université de Caen, F-14032 Caen (France)

Abstract

This paper provides a detailed analysis of various implementations of digital tries, including the "ternary search tries" of Bentley and Sedgewick. The methods employed combine symbolic uses of generating functions, Poisson models, and Mellin transforms. Theoretical results are matched against real-life data and justify the claim that ternary search tries are a highly efficient dynamic dictionary structure for strings and textual data.

Introduction

Digital trees, usually called *tries*, are both an abstract structure and a data structure that can be superimposed on a set of strings over some fixed alphabet. As an abstract structure, they are based on a splitting according to letters encountered in strings: if S is a set of strings, and $\mathcal{A} = \{a_j\}_{j=1}^{r}$ is the alphabet, then the trie associated to S is defined recursively by the rule:

$$\text{trie}(S) = \langle \text{trie}(S \setminus a_1), \ldots, \text{trie}(S \setminus a_r) \rangle,$$

where $S \setminus \alpha$ means the subset of S consisting of strings that start with α, stripped of their initial letter α; recursion is halted as soon as S contains less than 2 elements. The advantage of the trie is that it only maintains the minimal prefix set of characters that is necessary to distinguish all the elements of S.

Clearly the tree trie(S) supports the search for any string w in the set S by following an access path dictated by the successive letters of w. By similar means, the trie implements insertions and deletions, so that it is a fully dynamic *dictionary* data type. In addition, tries efficiently support set-theoretic operations like union and intersection [17], as well as partial match queries or interval search [7, 14], and suitable adaptations make them a method of choice for complex text processing tasks [9, Ch. 7]. These various applications justify considering the trie structure as one of the central general-purpose data structures of computer science [9, 11, 13, 15].

When it comes to implementation, several options are possible depending on the decision structure chosen to guide descent in subtrees. Three major choices present themselves.

— The "array-trie" uses an *array* of pointers to access subtrees directly; this solution is adequate only when the cardinality of the alphabet is small (typically for binary strings) since otherwise it creates a large number of null pointers.

— The "list-trie" structure remedies the high storage cost of array-tries by linking sister subtrees at the expense of replacing direct array access by a *linked list* traversal.

— The "bst-trie" uses *binary search trees* (bst) as subtree access method, with the goal of combining advantages of array-tries in terms of time cost, and list-tries in terms of storage cost.

This paper is devoted to the analysis of such *hybrid trie* structures, especially the list-trie and the bst-trie, many properties of basic (array) tries being already known [10, 11, 13, 16]. Our motivation comes in fact from a recent paper of Bentley and Sedgewick [2] who, following early ideas of Clampett [4], developed an elegant implementation of bst-tries, under the name of *ternary search trie*, or *tst* for short. The basic idea of [2, 4] is to represent the bst-trie as a ternary tree where search on letters is conducted like in a standard binary search tree over the alphabet set $\mathcal{A}$, while trie descent is performed by following an escape pointer whenever equality of letters of detected. In this way, the code is especially compact and, in simulations, the implementation constants appear to be particularly small. Bentley and Sedgewick report that, in practical situations, their data structure can be more efficient than hashing while offering considerably wider functionality. Our goal, as analysts, is to examine this claim and precisely quantify what goes on.

*Work supported in part by the Long Term Research Project ALCOM-IT (# 20244) of the European Union.

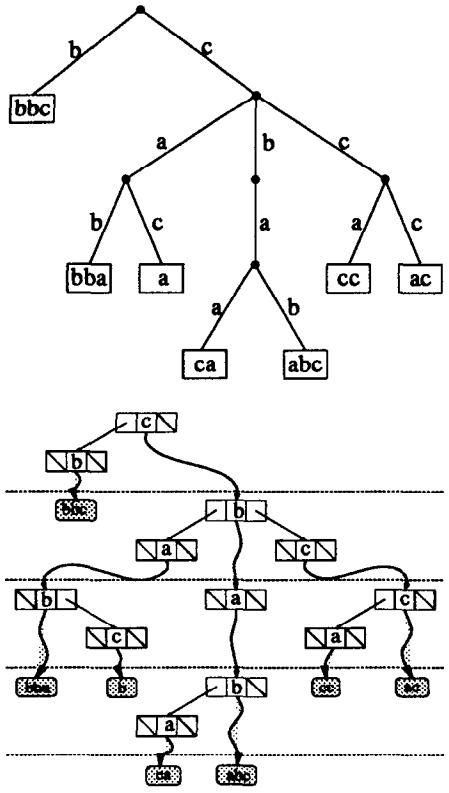


Figure 1: The basic trie representation (left) and the ternary search trie representation (right) of the sequence of strings S = (**cbab***abc*, **cab***bba*, **ccc***ac*, **cac***b*, **b***bbc*, **cbaa***ca*, **cca***cc*) over the alphabet $\mathcal{A} = \{a, b, c\}$.

More precisely, *all* our analytical models assume that n infinitely long keys are drawn *independently* from a common source, the universe of keys being the set $\mathcal{U} = \mathcal{A}^\infty$. The assumption of independence, which implies "random order", is the crucial one. Though not universally valid, it is satisfied in many situations (*e.g.*, successive words in a text like this paper are very weakly alphabetically correlated!) while providing a convenient basis for general comparisons between alternative data structures. (The assumption of infinitely long keys is only a technical convenience, as it is known to approximate reality quite well for reasonably large data bases of strings [17].) We consider a variety of quite general and fairly realistic *sources* for textual data, corresponding to probabilistic models on the universe $\mathcal{U} = \mathcal{A}^\infty$ from which individual keys are drawn. They include simple sources, the memoryless source with independent letters in strings, sources with a Markovian dependency between letters (the analysis focusses on these two cases), and even more general sources, like continued fraction representations of real numbers where the dependency between digits is no longer of a local nature.

Our main results are as follows. For both the memoryless source (m) and the Markov source (M), and either the list-trie or the ternary search trie, there exist computable constants $K_{\mathcal{S}}$, dependent upon the source $\mathcal{S} = \mathrm{m}, \mathrm{M}$, such that the expected cost (measured by the number of letter comparisons) of a random search in a tree built on n keys is of the form,

$$K_{\mathcal{S}} \log n + \mathcal{O}(1). \tag{0.1}$$

For instance, the ternary search trie applied to data provided by a Markov source (M) with transition probabilities $p_{j\,|\,i}$ (the probability of letter j occuring, given that the preceding letter is i) has

$$K_{\mathrm{M}} = \frac{1}{H_{\mathrm{M}}} \left(2 \sum_k \sum_{i<j} \frac{\pi_k\, p_{k\,|\,i}\, p_{k\,|\,j}}{p_{k\,|\,i} + \cdots + p_{k\,|\,j}} \right),$$

$$H_{\mathrm{M}} = -\sum_{k,j} \pi_k p_{j\,|\,k} \log p_{j\,|\,k},$$

where $\{\pi_k\}$ is the stationary probability of the chain, and H_{M} is none other than the entropy of the Markov chain.

In summary, we establish rigorously here that, on average, hybrid trie structures have logarithmic access costs under a variety of uses and we characterize precisely the dependence of implied constants on basic properties of the source model.

1 Search trees

The ternary search trie (tst) is, as we saw, a trie structure whose nodes are binary search trees (bst) over the character alphabet $\mathcal{A} = \{a_1, \ldots, a_r\}$, augmented with "escape links" for trie descent, in case of equality. Thus, the analysis of tst's requires first a dedicated analysis of bst's.

A word $w \in \mathcal{A}^\star$ is a sequence of letters regarded here as a sequence of insertions of letters (with possibly multiple occurrences) into a binary search tree, denoted bst(w), where letters of w are stored without repetition in nodes of the bst. The natural order $a_1 < \cdots < a_r$ is adopted throughout the paper. The independence assumption of our tst analysis model implies that one should endow $\mathcal{A}^\star$ with a probabilistic model where letters are drawn independently according to some (possibly nonuniform) probability distribution $\{p_j\}_{j=1}^r$.

Given such a word w, and a letter a_α, we define the *search cost* $c_\alpha[w]$ as the number of edges on the branch corresponding to a_α in bst(w). (Equivalently, if a_α occurs in w, $c_\alpha[w]$ is minus one plus the number of

nodes on the branch of a_α, while if a_α does not occur, then $c_\alpha[w]$ equals the number of nodes on that same branch.) The *cumulated search cost* $d_\alpha[w]$ is defined as $|w|_\alpha \cdot c_\alpha[w]$ and it represents the total cost for searching all the occurrences of a_α in the tree bst(w).

Our treatment is based on generating functions and it allows in fact for the simultaneous treatment of three probabilistic models:

— the multiset model, where w is obtained by a random permutation of the multiset $\{a_1^{n_1}, a_2^{n_2}, \dots, a_r^{n_r}\}$;

— the Bernoulli model based on a fixed probability distribution $\{p_j\}_{j=1}^r$ on letters of $\mathcal{A}$ and a fixed length of words, n;

— the Poisson model $\mathcal{P}(z, \{p_j\})$, which is similar but where the length N of w is itself a random variable that obeys a Poisson law of parameter z, $\Pr\{N = n\} = e^{-z}\frac{z^n}{n!}$.

The multiset model and the Bernoulli model have been considered earlier by Burge [3] and by Allen and Munro [1]. We re-derive and extend their results. In particular, consideration of the Poisson model is needed for the analysis of hybrid tries in the following sections.

THEOREM 1.1. (SEARCH COST IN BST'S) *The mean search cost and mean cumulated cost corresponding to α in a binary search tree, under the multiset, Bernoulli, and Poisson model, are given by*

$$\text{Multiset:}\quad \mathbf{E}[c_\alpha] = \sum_{j\neq\alpha} \frac{n_j}{N_{[j,\alpha]}}$$

$$\mathbf{E}[d_\alpha] = \sum_{j\neq\alpha} \frac{n_\alpha n_j}{N_{[j,\alpha]}} = n_\alpha \mathbf{E}[c_\alpha]$$

$$\text{Bernoulli:}\quad \mathbf{E}[c_\alpha] = \sum_{j\neq\alpha} \frac{p_j}{P_{[j,\alpha]}}\left[1-(1-P_{[j,\alpha]})^n\right]$$

$$\mathbf{E}[d_\alpha] = \sum_{j\neq\alpha} \frac{p_\alpha p_j}{P_{[j,\alpha]}^2}\left[nP_{[j,\alpha]} + (1-P_{[j,\alpha]})^n - 1\right]$$

$$\text{Poisson:}\quad \mathbf{E}[c_\alpha] = \sum_{j\neq\alpha} \frac{p_j}{P_{[j,\alpha]}}\left(1-e^{-zP_{[j,\alpha]}}\right)$$

$$\mathbf{E}[d_\alpha] = \sum_{j\neq\alpha} \frac{p_\alpha p_j}{P_{[j,\alpha]}^2}\left[zP_{[j,\alpha]} + e^{-zP_{[j,\alpha]}} - 1\right],$$

with $N_{[u,v]} = \sum_j n_j$, $P_{[u,v]} = \sum_j p_j$, and the index j ranges from $\min(u,v)$ *to* $\max(u,v)$.

PROOF. Our approach here relies on a symbolic description of parameters by *generating functions* (gf's) and is, perhaps, of independent interest. We proceed by stages and describe the search for a letter a_α in bst(w) by: a search of the maximum along the rightmost branch of bst$(w_{\leq\alpha})$, where $w_{<\alpha}$ means w restricted to elements of index smaller than α; a dual search of the minimum along the leftmost branch of bst$(w_{\geq\alpha})$. Each one-sided search is described by a regular expression and corresponding multivariate rational functions. The combination is achieved by a shuffle product that involves formal Laplace transforms.

(*i*) *Extrema analysis.* The first problem to be solved is thus the analysis of length of the rightmost branch in a tree built on random words, or equivalently the analysis of left-to-right maxima (also called "records"). Given the alphabet $\mathcal{A}$, the regular expression decomposition

$$\mathcal{A}^\star = \prod_{j=1}^r \left(\{\epsilon\} + a_j(a_1 + a_2 + \cdots a_j)^\star\right)$$

expresses precisely all the possible decompositions of words by sets of left-to-right maxima. Accordingly, the multivariate generating function, with $\mathbf{x} = x_1, x_2, \dots, x_r$,

$$N_{\max}(z,u,\mathbf{x}) = \prod_{j=1}^r \left(1 + \frac{zux_j}{1 - z(x_1 + \cdots + x_j)}\right), \tag{1.2}$$

is such that the coefficient of $[z^n u^k x_1^{n_1}\cdots x_r^{n_r}]$ equals the number of words of length n that have k maxima and n_j occurrences of letter a_j. This property is based on the systematic correspondence between operations on formal languages and operators on gf; for instance $(1-g)^{-1} = 1+g+g^2+\cdots$ generates arbitrary repetitions of elements enumerated by g, which corresponds to the star operation on languages. Dually, the multivariate gf for minima is

$$N_{\min}(z,u,\mathbf{x}) = \prod_{j=1}^r \left(1 + \frac{zux_j}{1 - z(x_j + \cdots + x_r)}\right). \tag{1.3}$$

(*ii*) *Search costs.* Consider next the search cost of some fixed letter α. The formal sum

$$C_\alpha := \sum_{w\in\mathcal{A}^\star} u^{c_\alpha[w]} \cdot w$$

when interpreted as a multivariate generating function (reading zx_j for the letter a_j) is such that the coefficient $[z^n u^k x_1^{n_1}\cdots x_r^{n_r}]$ represents the number of words w with length n, n_j occurrences of letter j, and search cost equal to k. Now one has the shuffle ('ш') decomposition [5, 12],

$$(\mathcal{A}\setminus\{\alpha\})^\star = (a_1 + \cdots + a_{\alpha-1})^\star \text{ ш } (a_{\alpha+1} + \cdots + a_r)^\star, \tag{1.4}$$

meaning that each word decomposes into into subwords $< \alpha$ and $> \alpha$, shuffled in all possible ways.

It is a known and easy result that a shuffle of languages over disjoint alphabets corresponds to an operation on generating functions (also denoted by '⧢') that is defined by

$$\left(\sum_n f_n z^n\right) \text{⧢} \left(\sum_n g_n z^n\right) = \sum_n \left(\sum_{k=0}^{n} \binom{n}{k} f_k g_{n-k}\right) z^n,$$

since the binomial in the convolution enumerates all possible shuffles. Equipped with this operation, we have $(\mathbf{x}_{u..v} = x_u, \dots, x_v)$,

(1.5)
$$C_\alpha(z, u, \mathbf{x}) = (N_{\max}(z, u, \mathbf{x}_{1..\alpha-1}) \text{⧢} N_{\min}(z, u, \mathbf{x}_{\alpha+1..r})) \cdot \left(1 + \frac{z x_\alpha}{1 - z(x_1 + \cdots + x_r)}\right),$$

where the last factor takes into account trailing sequences that may contain α.

(iii) Explicit forms. Eq. (1.5) condenses all the information on costs, including the full distribution. The gf of average costs is as usual obtained by differentiating with respect to u and setting $u = 1$. The rest of the computation is carried out by means of Laplace transforms, partial fraction expansions, and logarithmic derivatives. The formal Laplace transform $\mathcal{L}$ is defined by

$$\mathcal{L}[\sum_n f_n \frac{z^n}{n!}] = \sum_n f_n z^n,$$

and satisfies

$$f(z) \text{⧢} g(z) = \mathcal{L}[\mathcal{L}^{-1}[f(z)] \cdot \mathcal{L}^{-1}[g(z)]].$$

The result for the multiset model then derives from extracting coefficients in rational multivariate gf's. For the Bernoulli model, one sets $x_j \rightarrow p_j$ and extracts the coefficient $[z^n]$ from the univariate rational gf. For the Poisson model, the following general principle is used. If f_n is the expectation under the Bernoulli model of index n, then the corresponding expectation under the Poisson model is

$$\text{(1.6)} \qquad \sum_n f_n e^{-z} \frac{z^n}{n!} = e^{-z} \mathcal{L}^{-1}\left[\sum_n f_n z^n\right].$$

It then suffices to use the obvious correspondences and relations

$$\mathcal{L}[e^{az}] = \frac{1}{1-az}, \quad \mathcal{L}[z e^{az}] = \frac{z}{(1-az)^2},$$

$$\frac{1}{1-az} \text{⧢} \frac{1}{1-bz} = \frac{1}{1-(a+b)z}.$$

Computations (details omitted) complete the proof of Theorem 1.1. □

With the generating function framework, we have full access to the distribution of extrema and search costs under three models. Variances in particular have explicit expressions of a form similar to the means. Our results also specialize easily to the random permutation model (corresponding to each $n_j = 1$ in the multiset model) where one has Gaussian limit laws for records and search costs, as found earlier by Goncharov, Lynch, and Louchard (see [13]). From our gf expressions and continuity theorems, it may be shown that the Gaussian laws survive for a variety of letter distributions, like generalized Zipf laws.

2 Ternary search tries

In this section, we complete the analysis of ternary search tries (tst's) built on a random n-tuple $S = (s_1, \dots, s_n)$ of infinite strings from the universe of keys $\mathcal{U} = \mathcal{A}^\infty$. The set $\mathcal{U}$ is itself endowed with a probability measure, the source model, and we center the discussion on *memoryless* and *Markov sources*, though more general "dynamical" models are amenable to our approach. The two main results are Theorems 2.1 and 2.2 below that give an exact analysis and an asymptotic analysis of search costs.

More precisely, we consider the two parameters that characterize a positive search and a random search (or equivalently, a negative search, since a random search fails with probability 1). In the course of a search in a tst, different kinds of links are followed: the "comparison pointers" (with outcome '<' or '>') and the "descent pointers" that correspond to an escape to the next level upon equality of characters. The analysis of the cost induced by equality pointers is the same as that of standard tries and essentially known. So, we concentrate here on the *comparison cost* induced by the bst access structures present at each node that predominates. Path length that describes the total cost of all positive searches grows like $n \log n$ and the cost of a random (negative) search grows like $\log n$. The purpose of this section is precisely to characterize the constants involved.

Two complementary methods are used. The exact analysis is performed through an intermediate Poisson model, where the number N of strings is itself a random variable that obeys a Poisson law of parameter z. The asymptotic analysis further relies on Mellin transforms and Dirichlet series associated to the source model.

Exact analysis. We define the (comparison) *path length* $L(t)$ of a tst t as the sum of the distances of all external nodes to the root of the tree, where

distance is measured in the number of comparison pointers; see Fig. 1. Similarly, for s a string, we define the (comparison) *search cost* $R(t,s)$ as the number of comparison pointers followed when accessing the string s in the tst t. The comparison costs decompose according to the underlying trie structure,

$$(2.7)\qquad \begin{aligned} L(t) &= \ell(\mathsf{Root}(t)) + \sum_{i=1}^{r} L(t_i) \\ R(a_i\cdot s,t) &= r_i(\mathsf{Root}(t)) + R(s,t_i), \end{aligned}$$

where t_i denotes the subtree of t relative to the letter a_i. The terms $\ell(\mathsf{Root}(t))$ and $r_i(\mathsf{Root}(t))$ are so-called toll functions and they represent respectively the traversal cost (in number of links) and the cost of searching for the letter a_i in the bst present at the root of t.

Consider the Poisson model of rate z. It is a well known property that Poisson flows in an interval lead to (independent) Poisson flows in (disjoint) subintervals. Thus, the number N_h of strings that have a given prefix h obeys a Poisson law of parameter $p_h z$, where p_h is the probability that a random element of $\mathcal{A}^\infty$ starts with the string h, a quantity that depends of course on the source. Let $p_{i|h}$ be the probability that a random string conditioned to start with the prefix h has its letter following h that equals a_i. Then, the probabilistic behaviour of the tst that corresponds to the "place" associated to h is described by a Poisson model of rate zp_h with individual letter probabilities $\{p_{i|h}\}$. Then, Theorem 1 applies locally to this place: it suffices to replace in the statement the Poisson model $\mathcal{P}(z,\{p_i\})$ (rate z and letter probabilities $\{p_i\}$) by the Poisson model $\mathcal{P}(zp_h,\{p_{i|h}\})$. Denoting by $\mathbf{E}[d_\alpha,\mathcal{P}(zp_h,\{p_{i|h}\})]$ and $\mathbf{E}[c_\alpha,\mathcal{P}(zp_h,\{p_{i|h}\})]$ the expected values of d_α and c_α in this local Poisson model, one has from Theorem 1:

$$(2.8)\qquad \begin{aligned} \mathbf{E}[\ell(h)] &= \sum_{\alpha=1}^{r} \mathbf{E}[d_\alpha,\mathcal{P}(zp_h,\{p_{i|h}\})] \\ \mathbf{E}[r_\alpha(h)] &= \mathbf{E}[c_\alpha,\mathcal{P}(zp_h,\{p_{i|h}\})]. \end{aligned}$$

What remains to be done is to relate the quantities of (2.8) to the expectations of the global parameters R and L. The recursion formula (2.7) unwinds, and the expectations under a Poisson model of rate z are found to satisfy

$$\begin{aligned} \mathbf{E}_z[L] &= \sum_{h\in\mathcal{A}^*} \mathbf{E}[\ell(h)] \\ \mathbf{E}_z[R] &= \sum_{h\in\mathcal{A}^*} p_h \sum_{\alpha=1}^{r} p_{\alpha|h}\,\mathbf{E}[r_\alpha(h)]. \end{aligned}$$

Taking the individual expectations from (2.8), we then get

$$(2.9)\qquad \begin{aligned} \mathbf{E}_z[L] &= 2\sum_{h\in\mathcal{A}^*}\sum_{i<j} \frac{p_{h\cdot i}\,p_{h\cdot j}}{P_{h\cdot[i,j]}^{\,2}}\left[zP_{h\cdot[i,j]} - 1 + e^{-zP_{h\cdot[i,j]}}\right], \\ \mathbf{E}_z[R] &= 2\sum_{h\in\mathcal{A}^*}\sum_{i<j} \frac{p_{h\cdot i}\,p_{h\cdot j}}{P_{h\cdot[i,j]}}\left[1 - e^{-zP_{h\cdot[i,j]}}\right], \end{aligned}$$

where $P_{h\cdot[i,j]} = \sum_{k=i}^{j} p_{h\cdot k}$. (To simplify notations, we identify a letter with its index and use has been made of the relations $p_h\,p_{\alpha|h} = p_{h\cdot\alpha}$.)

The transition from the Poisson model to a model with the cardinality n of the data set being fixed is then simply achieved by the formal dictionary

$$e^{-az} \rightarrowtail (1-a)^n, \quad ze^{-az} \rightarrowtail n(1-a)^{n-1},$$

in accordance with the principles of the previous section; see (1.6).

THEOREM 2.1. (EXACT COSTS) *The (comparison) path length and the (comparison) cost of a random search in a ternary search trie made of n keys have expectations given by ($P_{h\cdot[i,j]} = \sum_{k=i}^{j} p_{h\cdot k}$)*

$$\begin{aligned} \mathbf{E}_n[L] &= 2\sum_{h\in\mathcal{A}^*}\sum_{i<j} \frac{p_{h\cdot i}\,p_{h\cdot j}}{P_{h\cdot[i,j]}^{\,2}}\left[nP_{h\cdot[i,j]} - 1 + (1-P_{h\cdot[i,j]})^n\right], \\ \mathbf{E}_n[R] &= 2\sum_{h\in\mathcal{A}^*}\sum_{i<j} \frac{p_{h\cdot i}\,p_{h\cdot j}}{P_{h\cdot[i,j]}}\left[1 - (1-P_{h\cdot[i,j]})^n\right]. \end{aligned}$$

It is a noteworthy feature that this theorem makes absolutely no assumption on the source model and is solely a consequence of independence. Quantities like p_h admit more or less complicated expressions in each particular case. For the memoryless model, one has $p_h = p_1^{m_1}\cdots p_r^{m_r}$, where m_i is the number of letters i in h, so that the sum over all h is an r-fold sum in disguise; for Markov models, one has to resort to matrix forms. Nonetheless, these formulae together with their Poisson counterparts constitute a useful input to asymptotic analysis where a phenomenon of "asymptotic simplification" takes place (as usual!).

Asymptotic analysis. The expectations in Theorem 2.1 and Equation (2.9) belong to the paradigm of harmonic sums [6] that are general sums of the form $F(z) = \sum_{k\in K} \lambda_k f(\mu_k z)$, Asymptotic analysis of such sums is classically done by the Mellin transform [6], that associates to a real function $g(x)$ the transform

$$g^\star(s) = \int_0^\infty g(x)x^{s-1}\,dx.$$

The Mellin transform $F^\star$ of the general harmonic sum F factorizes as $F^\star(s) = \Lambda(s)f^\star(s)$, where $\Lambda(s) = \sum_{k\in K} \lambda_k \mu_k^{-s}$ is the associated Dirichlet series. It is

known that the asymptotic behaviour of $F(x)$ is driven by the singularities of the Mellin transform $F^\star(s)$, hence eventually by singularities of the Dirichlet series $\Lambda(s)$.

The Poisson expectations in (2.9) prove more basic, and we start with them. Introduce the Dirichlet series

$$\Lambda(s) = 2 \sum_{h \in \mathcal{A}^\star} \sum_{i<j} \frac{p_{h\cdot i}\, p_{h\cdot j}}{P_{h\cdot[i,j]}{}^{2+s}}.$$

The Mellin transforms of the expectations $\mathbf{E}_z[L]$ and $\mathbf{E}_z[R]$ are, by the harmonic sum property, $-\Lambda(s)\Gamma(s)$, $-\Lambda(s-1)\Gamma(s)$, with definition domains $\langle -2, -1\rangle$ and $\langle -1, 0\rangle$. Singularities are needed to complete the analysis.

Two kinds of sources are considered here: memoryless sources and sources based upon Markov chains. A memoryless source (sometimes called Bernoulli) produces infinite strings where a letter a_i has probability p_i to appear independently of past history. A Markov source produces letters with an initial distribution and with transition probabilities $p_{j|i}$, the probability of producing j given i. (Only first-order Markov models are considered here for notational simplicity.) In both cases, the Dirichlet series $\Lambda(s)$ has an explicit expression. For the memoryless source, one has

$$\begin{aligned}\Lambda(s) &= \left(\sum_{h \in \mathcal{A}^\star} p_h{}^{-s}\right)\left(2\sum_{i<j} \frac{p_i\, p_j}{P_{[i,j]}{}^{2+s}}\right)\\ &= \frac{1}{1-(p_1{}^{-s}+\cdots+p_r{}^{-s})}\left(2\sum_{i<j} \frac{p_i\, p_j}{P_{[i,j]}{}^{2+s}}\right).\end{aligned}$$

For the Markovian source, the first character distribution being $\{\widetilde{p}_k\}$, one has a matrix form,

$$\Lambda(s) = {}^t b_s\, (I - P_s)^{-1}\, \widetilde{p}_s, \tag{2.10}$$

where $\widetilde{p}_s = (\widetilde{p}_k{}^{-s})_{1\le k\le r}$, $P_s = (p_{j|i}{}^{-s})$ is the element-wise power of order $-s$ of the transition matrix, and b_s is the vector whose the kth component is

$$2\sum_{i<j} \frac{p_{i|k}\, p_{j|k}}{(p_{i|k}+\cdots+p_{j|k})^{2+s}}.$$

In both case, the function Λ has a simple pole at $s=-1$ and the residue turns out to be related to the entropy of the source, as is apparent in the memoryless situation.

For the analysis with a fixed data set cardinality n, we appeal to a technique a "Dirichlet depoissonization". It is based on the observation that the averages in Theorem 2.1 are also harmonic sums, but with a more complicated Dirichlet series,

$$\widehat{\Lambda}(s) = 2 \sum_{h \in \mathcal{A}^\star} \sum_{i<j} \frac{p_{h\cdot i}\, p_{h\cdot j}}{P_{h\cdot[i,j]}{}^{2}} \left(\log \frac{1}{1-P_{h\cdot[i,j]}}\right)^{-s},$$

and on the property that dominant poles of $\widehat{\Lambda}(s)$ are exactly the same for $\widehat{\Lambda}(s)$ and $\Lambda(s)$.

THEOREM 2.2. (ASYMPTOTIC COSTS) *The (comparison) external path length path and random search for a ternary search tree built on n keys produced by a source $\mathcal{S}$, either memoryless (m) or Markovian (M), have averages that satisfy*

$$\begin{aligned}\mathbf{E}_n[L] &= \frac{1}{H_{\mathcal{S}}} C_{\mathcal{S}} \cdot n \log n + O(n)\\ \mathbf{E}_n[R] &= \tfrac{1}{H_{\mathcal{S}}} C_{\mathcal{S}} \cdot \log n + O(1),\end{aligned}$$

where the entropy $H_{\mathcal{S}}$ and the quantity $C_{\mathcal{S}}$ are source-dependent constants

$$\begin{aligned}H_{\mathrm{m}} &= \sum_i -p_i \log p_i\\ H_{\mathrm{M}} &= \sum_k \pi_k \sum_j -p_{j|k} \log p_{j|k}\\ C_{\mathrm{m}} &= 2\sum_{i<j} \frac{p_i p_j}{p_i+\cdots+p_j}\\ C_{\mathrm{M}} &= 2\sum_k \pi_k \sum_{i<j} \frac{p_{i|k} p_{j|k}}{p_{i|k}+\cdots+p_{j|k}}.\end{aligned}$$

Standard tries under a Markov model have been analysed by Jacquet and Szpankowski [10] who additionally obtained limit distributions. The analysis conducted here provides an extension of their logarithmic cost estimates to tst's. It even applies to non-Markovian sources, for instance continued fraction representations of real numbers (where the alphabet $\mathbb{N}$ is now infinite), and more generally so-called "dynamical sources" [18]. Standard (array) tries have been analysed in the continued fraction context in [8]. From [8, 18] and methods of this paper, ternary search tries applied to continued fraction representations still lead to logarithmic costs, for instance,

$$\begin{aligned}\mathbf{E}_n[L] &= K_{\mathrm{CF}}\, n \log n + \mathcal{O}(n)\\ K_{\mathrm{CF}} &= \frac{12}{\pi^2}\left[\frac{1}{4} + \sum_{k\ge 2} \frac{1}{k^2-1} \log \frac{k+1}{2}\right].\end{aligned}$$

The proof is based on the use of transfer operators $\mathcal{G}_s$ that vastly generalize the matrix forms like (2.10) that are associated with Markov chains.

3 Comparative studies

As detailed in the introduction, tries exist in three major versions: the array-trie (the standard version), the list-trie, and the bst-trie (the ternary search trie implementation). These should be compared in terms

	array-trie (standard)	list-trie	bst-trie (tst)
Pointers	$\frac{r}{H_S} n$	$\frac{2}{H_S} n$	$\frac{3}{H_S} n$
Path length	$\frac{1}{H_S} n \log n$	$\frac{C_S^\star}{H_S} n \log n$	$\frac{C_S}{H_S} n \log n$
Search	$\frac{1}{H_S} \log n$	$\frac{C_S^\star}{H_S} \log n$	$\frac{C_S}{H_S} \log n$

Figure 2: A comparative table of three trie implementations under three cost measures. The corresponding constants are given in Theorem 2.2 for $C_S = C_{\mathrm{m}}, C_{\mathrm{M}}$ and Eq. (3.12) for $C_S^\star = C_{\mathrm{m}}^\star, C_{\mathrm{M}}^\star$.

of time and space complexity, that is in terms of storage utilization and access cost. Theorems 2.1 and 2.2 quantify precisely the access cost for tst's. The reference parameters are the number of internal nodes I°, the path length L°, and the search cost R° of *standard array-tries*, whose analysis stems from Knuth's books,

$$\begin{aligned} \mathbf{E}_n[I^\circ] &\approx \frac{1}{H_S} n \\ \mathbf{E}_n[L^\circ] &\sim \frac{1}{H_S} n \log n \\ \mathbf{E}_n[R^\circ] &\sim \frac{1}{H_S} \log n. \end{aligned} \tag{3.11}$$

These estimates are valid for both the memoryless (m) and the Markov (M) source [10, 11, 13] , H_S is the source entropy, and the approximation sign '$\approx$' in size estimates conceals oscillations of a minute amplitude arising from complex poles of Dirichlet series.

The exact and asymptotic methods of Section 2 apply to *list-tries*, with the simpler analysis of linked lists being substituted for the analysis of bst's in Section 1. The (comparison) path length $L^\star$ and the search cost $R^\star$ of list-tries are found to satisfy

(3.12)

$$\mathbf{E}_n[L^\star] \sim \frac{C_S^\star}{H_S} n \log n, \quad \mathbf{E}_n[R^\star] \sim \frac{C_S^\star}{H_S} \log n$$
$$C_{\mathrm{m}}^\star = \sum_i (i-1) p_i, \quad C_{\mathrm{M}}^\star = \sum_k \pi_k \sum_i (i-1) p_{i|k}.$$

Thus, all three structures have logarithmic access costs, require linear space, albeit with different constants. Globally, the situation for all three data structures, our two main models, and the three complexity measures is summarized by the table of Fig. 2.

It is interesting to note that some of these properties are dependent upon a finite alphabet cardinality as the continued fraction model induces an average path length of $\mathcal{O}(n(\log n)^2)$.

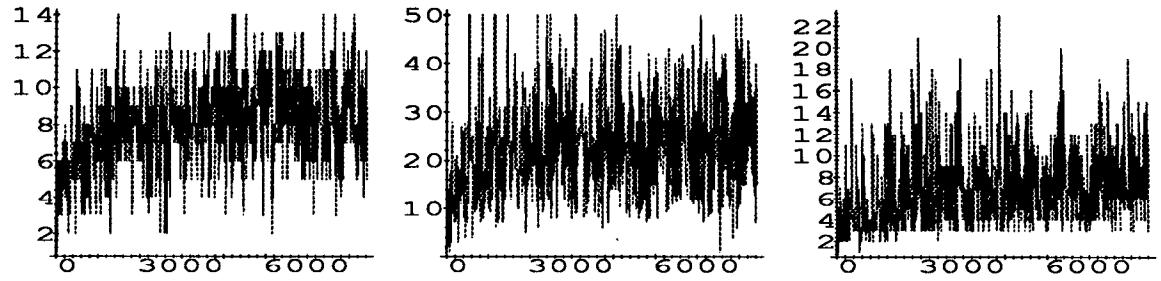
Experimental data. In order to assess the relevance of these analyses, we have conducted a simulation campaign on large real textual data. When we started, we were hopeful that the data structure alternatives could behave *qualititatively* in a way compatible with the theoretical models, but did not expect *quantitatively* much predictive power given the simplicity of our models (infinite strings, first-order Markov chains, asymptotic regime). The real situation turned out to be a little better.

As reference data, we have taken Herman Melville's novel *Moby Dick* whose complete text is available on the World Wide Web. The whole book comprises 1,295,000 characters and 217,000 words —contiguous alphabetical character strings— whose lengths range between 1 (the word 'a') and 20 ('uninterpenetratingly'). Various experiments have been conducted, like splitting the text into halves, filtering out short words (of length ≤ 5, say), etc. There are altogether 17,448 *distinct* words in the whole novel. One can then easily build a Bernoulli model and a Markov model on the corpus. The theoretical estimates for the entropy and the C constants are then determined by Theorems 2.1 and 2.2:

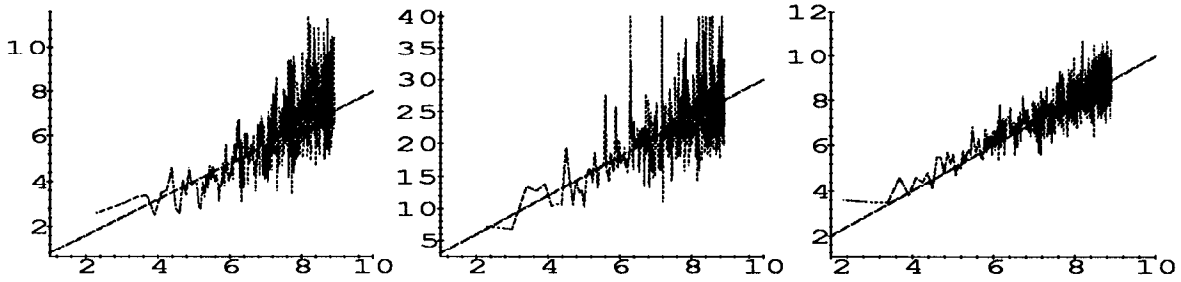
$$\begin{aligned} H_{\mathrm{m}} &= 2.896, \quad H_{\mathrm{M}} = 2.271 \\ C_{\mathrm{m}}^\star &= 10.649, \quad C_{\mathrm{M}}^\star = 10.447 \\ C_{\mathrm{m}} &= 3.298, \quad C_{\mathrm{M}} = 2.295. \end{aligned}$$

(As anticipated, the Markov model leads to a lower entropy estimate and the values found here are in fair agreement with estimates based on other texts [19].) For search costs $R^\circ, R^\star, R$, in standard, list, and bst tries containing n data items, this yields respectively

Moby Dick data. The evolution of insertion costs [left: array-trie, middle: list-trie, right: bst-trie], or equivalently negative search costs, shows an unclear tendency to increase as the number of data items n increases, and there is a fairly large variability of numerical data,



The presentation obtained by plotting against $\log n$ the costs averaged over successive batches of 10 insertions exhibits more clearly the logarithmic trends,



and leads to empirical formulæ for the search costs in array-tries, list-tries, and bst-tries:

$$\mathbf{E}_n[R^\circ] \approx 0.8 \log n, \qquad \mathbf{E}_n[R^*] \approx 3.0 \log n, \qquad \mathbf{E}_n[R] \approx 1.0 \log n.$$

Finally, the evolution of path length divided by $n \log n$, as insertions proceeds [bottom: standard trie, middle: bst-tries, top: list-tries], provides another view of the data. Here, the curves go by pairs corresponding to the two halves of the corpus.

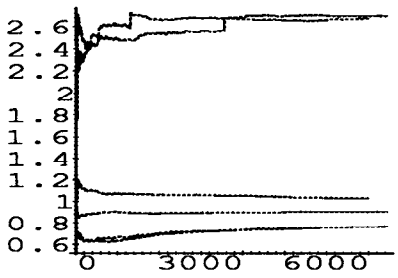


Figure 3: A display of the evolution of search costs and path lengths as a function of the number of strings inserted in standard and hybrid trie structures.

the predictions

memoryless source (m) :

$$0.345 \log n,\ 3.677 \log n,\ 1.138 \log n$$

Markov source (M) :

$$0.440 \log n,\ 4.600 \log n,\ 1.010 \log n.$$

We describe in some detail one experiment; see Fig. 3 for graphics. Taking the first half of the text and retaining words of length 6 or more, we obtain a collection P_1 of $n = 7437$ different words. The number of internal trie nodes is then 8444, and the path lengths of the standard array-trie (L°), list-trie ($L^\star$), and tst (L) appear to be, respectively, $L^\circ = 50894$, $L^\star = 178688$, $L = 59715$. A fit of the path length data and the individual insertion costs suggests approximate formulæ (Fig. 3)

$$\mathbf{E}_n[R^\circ] \approx 0.8 \log n$$

$$\mathbf{E}_n[R^\star] \approx 3.0 \log n,$$

$$\mathbf{E}_n[R] \approx 1.0 \log n.$$

for the search costs in the three structures.

The theoretical predictions turn out to be optimistic by a factor of about 2 for trie size (number of internal nodes) and trie path length, a fact to be probably assigned to the large number of closely resembling words in a dictionary based on raw written forms instead of "normal forms" (*e.g.*, here, aboriginal, aboriginally, aboriginalness). The predictions for list-tries are pessimistic by 20% (memoryless model) or 50% (Markov model). The predictions for bst-tries turn out to be quite close to reality (especially under the Markov model), a happy event.

Simplifying the discussion (Fig. 2 and 3), a concise practical conclusion is as follows:

> *Ternary search tries are an efficient data structure from the information theoretic point of view since a search costs typically about* $\log n$ *comparisons on real-life textual data. List-tries require about 3 times as many comparisons as ternary search tries that implement bst-tries. For an alphabet of cardinality 26, the storage cost of ternary search tries is about 9 times smaller than standard array-tries.*

This justifies considering ternary search tries as a method of choice for managing textual data. As expressed by Bentley and Sedgewick [2], "*Ternary search tries combine the best of two worlds: the low overhead of binary search trees (in terms of space and running time) and the character-based efficiency of tries*".

References

[1] ALLEN, B., AND MUNRO, I. Self-organizing binary search trees. *Journal of the ACM 25*, 4 (Oct. 1978), 526–535.

[2] BENTLEY, J., AND SEDGEWICK, R. Fast algorithms for sorting and searching strings. In *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), SIAM Press.

[3] BURGE, W. H. An analysis of binary search trees formed from sequences of nondistinct keys. *JACM 23*, 3 (July 1976), 451–454.

[4] CLAMPETT, H. A. Randomized binary searching with tree structures. *Communications of the ACM 7*, 3 (Mar. 1964), 163–165.

[5] FLAJOLET, P., GARDY, D., AND THIMONIER, L. Birthday paradox, coupon collectors, caching algorithms, and self–organizing search. *Discrete Applied Mathematics 39* (1992), 207–229.

[6] FLAJOLET, P., GOURDON, X., AND DUMAS, P. Mellin transforms and asymptotics: Harmonic sums. *Theoretical Computer Science 144*, 1–2 (June 1995), 3–58.

[7] FLAJOLET, P., AND PUECH, C. Partial match retrieval of multidimensional data. *Journal of the ACM 33*, 2 (1986), 371–407.

[8] FLAJOLET, P., AND VALLÉE, B. Continued fraction algorithms, functional operators, and structure constants. Research Report 2931, Institut National de Recherche en Informatique et en Automatique, July 1996. 33 pages. Invited lecture at the 7th Fibonacci Conference, Graz, July 1996; to appear in *Theoretical Computer Science*.

[9] GONNET, G. H., AND BAEZA-YATES, R. *Handbook of Algorithms and Data Structures: in Pascal and C*, second ed. Addison–Wesley, 1991.

[10] JACQUET, P., AND SZPANKOWSKI, W. Analysis of digital tries with Markovian dependency. *IEEE Transactions on Information Theory 37*, 5 (1991), 1470–1475.

[11] KNUTH, D. E. *The Art of Computer Programming*, vol. 3: Sorting and Searching. Addison-Wesley, 1973.

[12] LOTHAIRE, M. *Combinatorics on Words*, vol. 17 of *Encyclopedia of Mathematics and its Applications*. Addison–Wesley, 1983.

[13] MAHMOUD, H. *Evolution of Random Search Trees*. John Wiley, New York, 1992.

[14] RIVEST, R. L. Partial match retrieval algorithms. *SIAM Journal on Computing 5* (1976), 19–50.

[15] SEDGEWICK, R. *Algorithms*, second ed. Addison–Wesley, Reading, Mass., 1988.

[16] SEDGEWICK, R., AND FLAJOLET, P. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, 1996.

[17] TRABB PARDO, L. Set representation and set intersection. Tech. rep., Stanford University, 1978.

[18] VALLÉE, B. Dynamical systems and average-case analysis of general tries. Preprint, 1997. 16p.

[19] WELSH, D. *Codes and cryptography*. Oxford Science Publications. Oxford University Press, 1988.