The Analysis of Hybrid Trie Structures

Julien Clément
Algorithms project, INRIA Rocquencourt

October 20, 1997

[summary by Frédéric Cazals]

1. Introduction

Tries are a general-purpose data structure of the dictionary type, that is supporting the three main operations Insert, Delete and Query. To see how they are defined, let $\mathcal{A} = \{a_j\}_{j=1}^{r}$ be an alphabet and S be a set of strings defined over $\mathcal{A}$. The trie associated to S is recursively defined by the rule

$$\text{trie}(S) = \langle \text{trie}(S\backslash a_1), \text{trie}(S\backslash a_2), \ldots, \text{trie}(S\backslash a_r) \rangle$$

where $S\backslash\alpha$ refers to the contents of S consisting of strings that start with α and stripped of their initial letter, and the recursion stops as soon as S contains one element.

Searching a trie T for a key w just requires tracing a path down the trie as follows: at depth i, the ith digit of w is used to orientate the branching. (Insertions and deletions are handled in the same way.) To complete the description, we need to specify which search structure is used to choose the correct sub-trie within a node. The main possibilities are:

1. the "array-trie" which uses an array of pointers to sub-tries. This solution is relevant for small alphabets only, otherwise too many empty pointers are created;
2. the "list-trie" that remedies the high-storage requirement of the "array-trie" by using a linked list of sub-tries instead. The drawback is a higher cost for the traversal;
3. the "bst-trie" which uses binary-search trees (bst) as a trade-off between the time efficiency of arrays and the space efficiency of lists.

In particular, the bst-trie can be represented as a ternary tree where the search on letters is conducted like in a standard binary search tree, while the tree descent is performed by following an escape pointer upon equality of letters. We shall refer to this data structure as a ternary search trie or tst. An example trie with its basic representation and the equivalent ternary search trie over the alphabet $\mathcal{A} = \{a, b, c\}$ is represented on fig. 1 and 2. As is well known, the performances

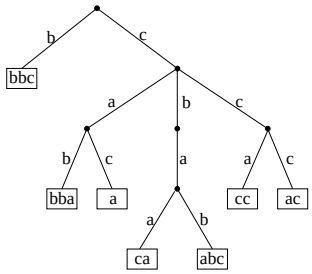

FIGURE 1. Basic trie

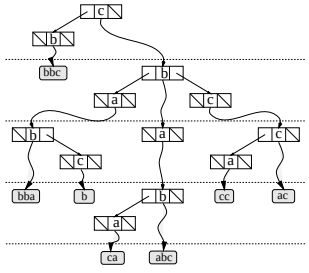

FIGURE 2. Its ternary search trie representation

of tries depend upon the probabilistic properties of the strings processed. More precisely, we shall work with two types of models:

- *The models for the infinite strings inserted in the tries.* These models depend upon the number of strings inserted—either a fixed number n or the output of a Poisson random variable $\mathcal{P}(n)$—and also on the way the characters are emitted after one another—either independently or with some memory scheme such as a Markov process or a continued fraction.
- *The models for the finite keys inserted in the tst nodes.* Examples of such models are the multiset model $\{a_1^{n_1}, a_2^{n_2}, \ldots, a_r^{n_r}\}$, the Poisson model $\mathcal{P}(n, p_i)$ or the Bernoulli model $\mathcal{B}(n, p_i)$. It should be emphasized that since the infinite strings are drawn independently, their ith letters are also independent, which is matched by the previous models.

The quantities we are interested in to capture the tries performances are defined as follows:

Definition 1. The comparison path length of a tst t is defined as the sum of the distances from the root to the external nodes, expressed in number of comparison pointers. Similarly, for a string s, the search cost $R(s,t)$ is defined as the number of comparison pointers followed when accessing s in t. More precisely,

$$L(t) = l(\text{root}(t)) + \sum_{i=1}^{r} L(t_i) \quad \text{and} \quad R(a_i s, t) = r_i(\text{root}(t)) + R(s, t_i) \tag{1}$$

with $l()$ the number of external nodes in the sub-tries pointed at by comparison pointers and $r_i()$ the cost of searching a_i in the bst present at the root of t.

2. Tools Used to Perform the Analysis

2.1. **Left-to-Right Maxima, Shuffle Product and Formal Laplace Transform.** Let w be a word of $\mathcal{A}^\star$. The ith letter of w denoted w_i is called a left-to-right maximum if $w_i \geq w_j, j = 1, \ldots, i-1$. For example, the permutation $a_2 a_3 a_1 a_5 a_4$ has three left-to-right maxima, respectively a_2, a_3 and a_5. If one builds a bst from a permutation, the left-to-right maxima are naturally in bijection with the nodes located on the rightmost branch. For the analysis to be performed in section 3.1, we therefore investigate left-to-right maxima.

Clearly, all the possible decompositions of words by sets of left-to-right maxima are encoded by the regular expression

$$\mathcal{A}^\star = \prod_{j=1}^{r} (\varepsilon + a_j (a_1 + \cdots + a_j)^\star).$$

Marking a_i by the two variables $z x_i$ together with u if it is a left-to-right maximum yields the generating function

$$N_{\max}(z, u, x_1, x_2, \ldots, x_r) = \prod_{j=1}^{r} \left(1 + \frac{z u x_j}{1 - z(x_1 + \cdots + x_j)}\right)$$

whose coefficient $[z^n u^k x_1^{n_1} \ldots x_r^{n_r}]$ is the number of words of length n that have k maxima and n_j occurrences of the letter a_j. A similar formula holds for $N_{\min}$, the generating function of minima.

Another tool we shall use is the shuffle product ш from which a word can be decomposed into words contained certain letters only:

$$(\mathcal{A} \backslash \{\alpha\})^\star = (a_1 + \cdots + a_{\alpha-1})^\star \text{ш} (a_{\alpha+1} + \cdots + a_r)^\star.$$

This product corresponds to the following operation on generating function

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

A nice way to handle it is through the formal Laplace transform defined by $\mathcal{L}\left[\sum_n f_n \frac{z^n}{n!}\right] = \sum_n f_n z^n$. In particular we have $f(z) \text{ш} g(z) = \mathcal{L}\left[\mathcal{L}^{-1}[f(z)] \cdot \mathcal{L}^{-1}[g(z)]\right]$.

3. Main Results

3.1. **Search Costs in Bst.** We analyze the cost of searching a letter a_α in the bst $bst(w)$ built from the letters of a word w from $\mathcal{A}^\star$. More precisely: given a letter a_α and a word w, the search cost $c_\alpha(w)$ is defined as the number of edges on the branch corresponding to a_α in the bst built from the letters of w. In particular, we are interested in condensing the cost related informations in the formal sum

$$C_\alpha := \sum_{w \in \mathcal{A}^\star} u^{c_\alpha(w)} \cdot w \tag{2}$$

whose exponent of u refers to the search cost $c_\alpha(w)$. To see how $c_\alpha(w)$ can be evaluated, observe that $w = \text{prefix}(w)\ \ a_\alpha?\ \ \text{suffix}(w)$ where $a_\alpha?$ means that the letter a_α may be absent. The interest of this decomposition is to show that the cost of searching a_α in the bst built from w is chargeable to prefix(w). And since prefix(w) can be expressed as the shuffle product on the sets of letters $a_1, \ldots, a_{\alpha-1}$ with $a_{\alpha+1}, \ldots, a_r$, the formal sum (2) yields the value of $C_\alpha(z, u, x_1, \ldots, x_r)$:

$$(N_{\max}(z, u, x_1, \ldots, x_{\alpha-1}) \text{ш} N_{\min}(z, u, x_{\alpha+1}, \ldots, x_r)) \left(1 + \frac{z x_\alpha}{1 - z(x_1 + \cdots + x_r)}\right).$$

This form condenses all the information on costs. For example, the generating function of average costs is obtained by differentiating with respect to u and setting $u = 1$. For example:

Theorem 1. *The mean search cost of the letter a_α in a bst built from the Poisson model is*

$$\mathrm{E}[C_\alpha] = \sum_{\substack{\min(u,v) \le j \le \max(u,v) \\ j \neq \alpha}} \frac{n_j}{P_{[j,\alpha]}} \left(1 - e^{-z P_{[j,\alpha]}}\right), \qquad \textit{with} \quad P_{[u,v]} = \sum_j p_j.$$

3.2. **Exact Analysis.** In this section, we outline the analysis of the statistics introduced in def. 1. The crux of this analysis consists in using quantities that are independent from the source model the keys are generated by. To see how this works, consider the Poisson process of parameter z. The number N_h of strings that have a given prefix h obeys a Poisson law of parameter $p_h z$, with p_h the source dependent probability for a random string to start with the prefix h. Then, the probabilistic behavior of the tst that corresponds to the prefix h is described by a Poisson model of parameter $\{z p_h\}$ with individual letter probabilities $\{p_{i|h}\}$, with $p_{i|h}$ the conditional probability to have the prefix h followed by the letter a_i. Applying theorem 1 locally and unwinding the recurrences (1) yield

Theorem 2. *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*

$$\mathrm{E}[L]_n = 2 \sum_{h \in \mathcal{A}^\star} \sum_{i<j} \frac{p_{h \cdot i} p_{h \cdot j}}{P_h[i,j]^2} [n P_h[i,j] - 1 + (1 - P_h[i,j])^n],$$

$$\mathrm{E}[R]_n = 2 \sum_{h \in \mathcal{A}^\star} \sum_{i<j} \frac{p_{h \cdot i} p_{h \cdot j}}{P_h[i,j]} [1 - (1 - P_h[i,j])^n]$$

	array-trie (standard)	list-trie	bst-trie (tst)
Pointers	$\frac{r}{H_{\mathcal{S}}} n$	$\frac{2}{H_{\mathcal{S}}} n$	$\frac{3}{H_{\mathcal{S}}} n$
Path length	$\frac{1}{H_{\mathcal{S}}} n \log n$	$\frac{C^\star_{\mathcal{S}}}{H_{\mathcal{S}}} n \log n$	$\frac{C_{\mathcal{S}}}{H_{\mathcal{S}}} n \log n$
Search	$\frac{1}{H_{\mathcal{S}}} \log n$	$\frac{C^\star_{\mathcal{S}}}{H_{\mathcal{S}}} \log n$	$\frac{C_{\mathcal{S}}}{H_{\mathcal{S}}} \log n$

TABLE 1.

with $P_h[i,j] = \sum_{k=i}^{j} p_{k\cdot h}$ *and* $p_{h\cdot\alpha} = p_h p_{\alpha|h}$.

A noteworthy feature of this theorem is its independence from the source model since its derivation uses solely the independence of the digits processed. It can therefore be instantiated for the three models mentioned in section 1.

3.3. **Asymptotic Analysis.** We aim at finding asymptotic equivalents to the quantities of theorem 2. These quantities are harmonic sums amenable to a treatment with the Mellin transform [2].

The Mellin machinery applied to the formulae of theorem 2 requires evaluating the $p_{i|j}$ probabilities. This is done under two models: a memoryless (a.k.a. Bernoulli) source outputting infinite strings where the letter a_i has probability to appear independently of past history; and a Markov one producing letters with an initial distribution and with transition probabilities $p_{i|j}$. Singularity analysis on the Mellin transforms (combined with the so-called Dirichlet depoissonization) yields

Theorem 3. *The comparison external path length and random search cost for a ternary search tree built on n keys produced by a source S, either memoryless (m) or Markovian (M), have averages that satisfy*

$$\mathrm{E}[L]_n = \frac{C_S}{H_S} n \log n + O(n) \qquad \textit{and} \qquad \mathrm{E}[R]_n = \frac{C_S}{H_S} \log n + O(1)$$

where the entropy H_S and the quantity C_S are source-dependent constants.

3.4. **Comparative Studies.** Theorems 3 and 2 quantify precisely the access costs for tst. The same analysis can be carried out for the list-trie variant, while the parameters describing standard array-trie stem from Knuth's books. The results are summarized in table 1—with $C^\star_m$ and $C^\star_M$ constants known in closed forms—and show that the three structures have logarithmic costs and require linear space. In order to assess the relevance of these theoretical analyses, a simulation campaign was undertaken on Herman Melville's novel, *Moby Dick*. Its conclusions are [1]:

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

Bibliography

[1] Clément (Julien), Flajolet (Philippe), and Vallée (Brigitte). – The analysis of hybrid trie structures. In *Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Algorithms*. pp. 531–539. – Philadelphia, 1998.

[2] Flajolet (Philippe), Gourdon (Xavier), and Dumas (Philippe). – Mellin transforms and asymptotics: harmonic sums. *Theoretical Computer Science*, vol. 144, n° 1-2, 1995, pp. 3–58. – Special volume on mathematical analysis of algorithms.