The Digital Tree Process

Julien Clément[1] Mark Daniel Ward[2],*

CNRS, research team Algorithms of GREYC[1]

Department of Statistics, Purdue University[2]

December 16, 2011

Dedicated to the memory of Philippe Flajolet.

*Supported by NSF Science & Technology Center grant CCF-0939370.

Introduction

- Discovered by De la Briandais (1959) et Fredkin (1960)
- Principles attributed to Thue (1912) by Knuth.
- TRIE ≡ TRee + reTRIEval.
- Based on the thumb rule in a dictionary
- Dynamical structure
- The analyses of digital tree processes pervade Philippe Flajolet's work.

Techniques

Trie Memory*

EDWARD FREDKIN, *Bolt Beranek and Newman, Inc., Cambridge, Mass.*

Introduction

Trie memory is a way of storing and retrieving information.[1] It is applicable to information that consists of function-argument (or item-term) pairs—information conventionally stored in unordered lists, ordered lists, or pigeonholes.

The main advantages of trie memory over the other memory plans just mentioned are shorter access time, greater ease of addition or up-dating, greater convenience in handling arguments of diverse lengths, and the ability to take advantage of redundancies in the information stored. The main disadvantage is relative inefficiency in using storage space, but this inefficiency is not great when the store is large.

In this paper several paradigms of trie memory are described and compared with other memory paradigms, their advantages and disadvantages are examined in detail, and applications are discussed.

Many essential features of trie memory were mentioned by de la Briandais [1] in a paper presented to the Western Joint Computer Conference in 1959. The present development is essentially independent of his, having been described in memorandum form in January 1959 [2], and it is fuller in that it considers additional paradigms (finite-dimensional trie memories) and includes experimental results bearing on the efficiency of utilization of storage space.

Basic Paradigm of Trie Memory

Let us consider first a simple abstract form of trie memory. Suppose that we need to keep track of a set of words, a set of sequences of alphabetic characters. The words are of various lengths. From time to time, additions to the set and deletions from it must be made. What we have to remember is just which ones, of all the possible finite sequences of alphabetic characters, are currently in the set. That is, given a word, we must be able to determine whether or not it is at present a member. In this example, each word is an argument, and the corresponding function is simply the binary variable of which the admissible values are *member* and *nonmember*.

At the outset, before a storage is begun, the trie is merely a collection of *registers*. Except for two special registers, which we may call α and δ, every register has a *cell* for each member (type) of the ensemble of alphabetic characters. If we let that ensemble include a "space" to indicate the end of a word (argument), each register must have 27 cells.

Each cell has space for the address of any register in the memory. Cells in the trie that are not yet being used to represent stored information always contain the address of the special α register. A cell thus represents stored information if it contains the address of some register other than α. The information it represents is its own name, "A" for the A cell, "B" for the B cell, etc., and the address of the next register in the sequence.

Storage of words of alphabetic characters is illustrated in Fig. 1. For the sake of simplicity, the ensemble of characters has been restricted to the first five letters of the alphabet and ∇ for "space." Suppose that we want to store DAB, BAD, BADE, BE, BED, BEAD, CAB, CAD, and A. We may follow the procedure illustrated in Fig. 1, in which the rows represent registers, each one

FIG. 1. Schematic representation of storage of words in trie memory

* The work reported here was begun at the MIT Lincoln Laboratory and completed at Bolt Beranek and Newman, Inc., with contractual support from the IBM Federal Systems Division. The

[1] ED. NOTE: "Trie" is apparently derived from reTRIEval.

most pervasive assistance.

[1] ED. NOTE: "Trie" is apparently derived from reTRIEval.

Tries

- Tries as a *data structure*
- Tries as a *partitioning digital process*

The digital tree process gave rise to many algorithmic variants:

1. PATRICIA trees,
2. digital search trees
3. LC-tries
4. hybrid trie structures (e.g., array-trie, bst-trie, list-trie)

E.g., *Digital Search Trees Revisited*, PF and R. Sedgewick (1986); and *The Analysis of Hybrid Trie Structures* (1998) and *Dynamical Sources in Information Theory* (2001), both by J. Clément, PF, and B. Vallée.

Fully Dynamic Dictionary Structure

Tries are dynamic data structures that store randomly generated words.
"Dynamic" because tries grow as more words are inserted.

First paragraph of *Moby Dick* (H. Melville)

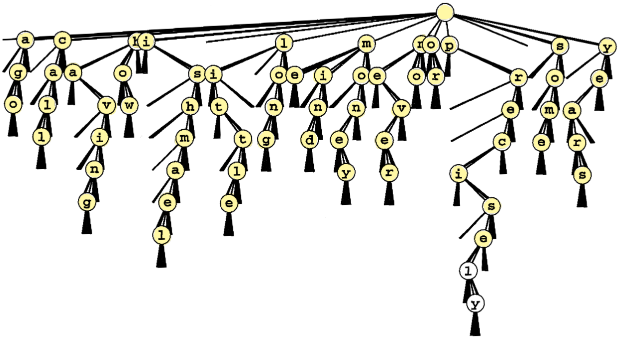


figure ©R. Sedgewick

Parameters: size (memory usage), external path length (searching), height

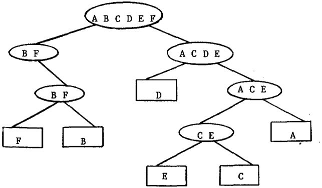
Motivations

Conceptually, the digital tree process can appear at unexpected places, and Philippe liked that a lot.

- recursive definition $\rightarrow$ functional equations via generating functions.
- database management; data mining
- data compression; closely related to Lempel-Ziv schemes
- efficient communication protocols; conflict resolution
- leader election and connections to distributed computing
- probabilistic counting
- hashing; bucket sorting (e.g., >1 string per leaf)
- polynomial factorization
- dictionary
- sorting and searching [Knuth 73]
- set intersection, set union [Trabb Pardo 78]
- multiway branching for generalized (non-binary) alphabet

Recursive Definition

A Recursive Partitioning Process of Computer Science (with D. Sotteau, 2nd World Conf. Math. at the Service of Man, 1982):



(Uncompressed!) Trie for sequences

$A = 111\ \ldots \qquad D = 10\ldots$

$B = 011\ \ldots \qquad E = 1100\ldots$

$C = 1101\ldots \qquad F = 010\ldots$

The splitting groups are "the 'heads' group and the 'tails' group."

Much later, for instance, in *The Ubiquitous Digital Tree* (STACS, 2006), he defines a trie recursively. For a set of strings ω,

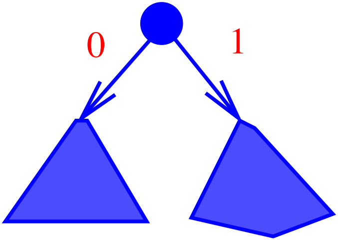
$$\text{trie}(\omega) := \begin{cases} \emptyset & \text{if } \omega = \emptyset, \\ \sigma & \text{if } \omega = \{\sigma\}, \\ \langle \bullet, \text{trie}(\omega \setminus 0), \text{trie}(\omega \setminus 1) \rangle & \end{cases} \tag{1}$$

"The motto here is thus simply 'filter and shift left'."

Recursive Description of Tries

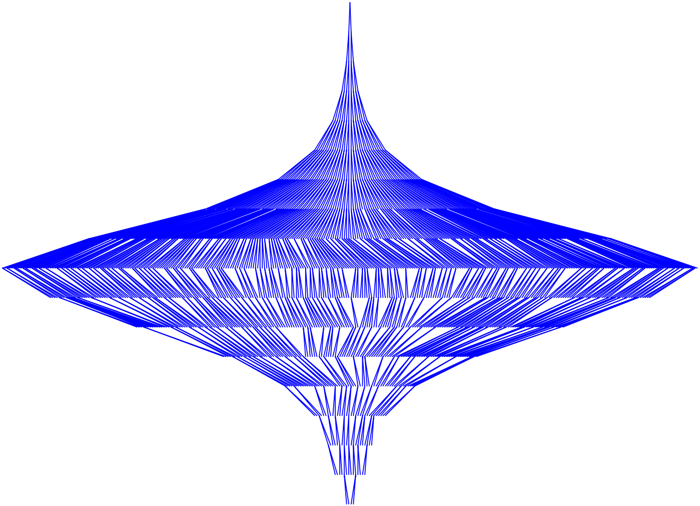
From Philippe's "Saga of Digital Trees", part of the Colloquium for Jacques Morgenstern in 2003:

> Compare-exchange based on successive *bits* of data.
>
> `place 0's on left, 1's on right;`
>
> `recurse.`
>
> The trie splitting process (Fredkin)



Example

One of Philippe's tries (TikZ rendering, used in several of his talks) built on 500 uniform binary sequences, with 741 internal nodes, and height 17:



Probability Models

The probability models for tries assume independence between the strings stores at the leaves.

The characters within the string may be indep., uniform (i.e., $p = q = 1/2$), indep., biased ($p \neq q$), or have Markov dependence, or even have a dynamical source.

Philippe's work also laid a foundation for analysis of suffix trees, in which the strings are dependent: they are suffixes of a common string. [E.g., Nicodème; Jacquet, Szpankowski; J. Fayolle, MDW.]

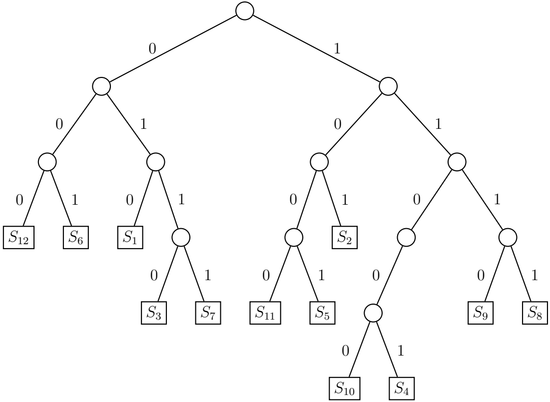
First-order, expected behavior of suffix tree parameters often agrees with analogous behavior for tries built over independently-generated strings.

First-order variance, higher moments, and also second-order terms of expectation, are often different in suffix trees vs tries built over independent strings.

$X_1, X_2, X_3, \ldots = 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, \ldots$

Suffixes of the sequence:

$S_1 = 0, 1, 0, 1, 1, 0, \ldots$
$S_2 = 1, 0, 1, 1, 0, 0, \ldots$
$S_3 = 0, 1, 1, 0, 0, 1, \ldots$
$S_4 = 1, 1, 0, 0, 1, 1, \ldots$
$S_5 = 1, 0, 0, 1, 1, 1, \ldots$
$S_6 = 0, 0, 1, 1, 1, 1, \ldots$
$S_7 = 0, 1, 1, 1, 1, 0, \ldots$
$S_8 = 1, 1, 1, 1, 0, 0, \ldots$
$S_9 = 1, 1, 1, 0, 0, 0, \ldots$
$S_{10} = 1, 1, 0, 0, 0, 0, \ldots$
$S_{11} = 1, 0, 0, 0, 0, 0, \ldots$
$S_{12} = 0, 0, 0, 0, 0, 1, \ldots$



“A Recursive Partitioning Process of Computer Science”

We see Philippe building on algebraic properties, e.g., set constructions, multisets, translations to generating functions, and complex-valued analysis [inheriting from De Bruijn, Knuth, Henrici, others].

“The power [of these results] comes from the fact that most parameters of interest on trees are definable as additive-multiplicative combinations of similar or simpler parameters on subtrees, so that a large number of equations can be written *systematically*.”
[PF, D. Sotteau 1982; emphasis added]

Philippe uses generating functions to unify earlier analyses of

- the family of sets whose associated tree has height $\leq k$
- generalized versions for leaves with $\leq b$ nodes (buckets)
- total number of nodes in tree when height is $\leq k$
- path length

“A Recursive Partitioning Process of Computer Science”

In 1982, Philippe was already synthesizing connections among different analysis of tries

- Collision resolution in networks [G. Fayolle, PF, M. Hofri 1982]
  - Time to resolve n collisions in an open stack protocol network: $\alpha_n = An + n\phi(n) + O(n/\log n)$
- Digital sorting and searching [Knuth 1973]
- Dynamic Hashing [PF, J. M. Steyaert 1982]
- Polynomial Factorization [PF, J. M. Steyaert 1982]

The function $\phi(n)$ is a fluctuating function with small amplitude.

Some Trie Parameters

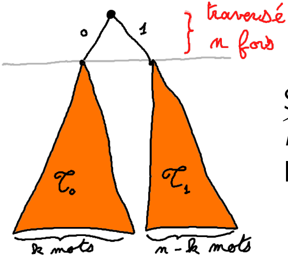
Commonalities: Trie arise in unexpected situations. The analysis often concerns asymptotic properties of a trie parameter (often called valuations by Philippe), e.g.,

- path length: sum (over all leaves) of distances from the root to the leaves,
- total number of nodes: also called the size of the tree,
- height: maximum distance from root to a leaf,
- number of unary nodes (leaves),

Philippe was a master at synthesizing ideas and making generalizations of results, especially results that were previously only known in special cases or certain situations.

Basic recurrence (binary case)

Unbiased memoryless source $p = 1/2$.



This decomposition gives for path length

$$L_n = \underbrace{n}_{\text{toll}} + \sum_{k=0}^{n} \frac{\binom{n}{k}}{2^n} \left(L_k + L_{n-k}\right)$$

So that:
$\widehat{L}(z) = \sum_{n \geq 0} L_n \frac{z^n}{n!} = z(e^z - 1) + 2e^{z/2}\widehat{L}(z/2).$
Iterating we obtain

$$\widehat{L}(z) = \sum_{k \geq 0} z(e^z - e^{(1-\frac{1}{2^k})z}).$$

N.B. An harmonic sum ! (see talk by P. Dumas)

Functional Equations [PF, D. Sotteau 1982]

Fundamental generalization:

For recursively defined parameters, the translation schemes (17), (18) lead to functional difference equations of the form

$$\phi(x) = a(x)\,\phi\left(\frac{x}{2}\right) + b(x)\,. \tag{19}$$

The a, b are toll functions that depend on the trie parameter.

These can normally be solved by iteration, so that

$$\phi(x) = \sum_{k\geq 0} b(x2^{-k}) \prod_{j=0}^{k-1} a(x2^{-j})\,.$$

In the frequent case $a(x) = Ae^{cx}$, (20) further simplifies to

$$\phi(x) = \sum_{k\geq 0} A^k\, b(x2^{-k})\, \exp(c(1-2^{-k})x) \tag{20}$$

Analysis of Periodicities [PF, D. Sotteau 1982]

For the path length S_n:

Such periodicities are of frequent occurrence in the analysis of algorithms and they have here a clear origin in regularly spaced singularities of functions of the type $(1-2^s)^{-1}$. An alternative derivation, which avoids the exponential approximation is based on the observation that S_n is itself a harmonic sum :

$$S(x) = \sum (e^{-x\log(1-2^{-k})^{-1}} - 1)$$

Philippe was using saddle points, Mellin transform, and making precise characterization of periodicities, in 1982, *more than 10 years before the first Analysis of Algorithms meeting*.

$$
\begin{tikzcd}
\text{Mean Value in the Poisson model} \arrow[r, Rightarrow, "\text{AlgDePo}"] \arrow[d, Rightarrow, "\text{Mellin}"'] & \text{Mean Value in the Bernoulli model} \arrow[d, Rightarrow, "\text{Rice}"] \\
\text{Asymptotic Mean Value in the Poisson model} \arrow[r, Rightarrow, "\text{AnDePo}"] & \text{Asymptotic Mean Value in the Bernoulli model}
\end{tikzcd}
$$

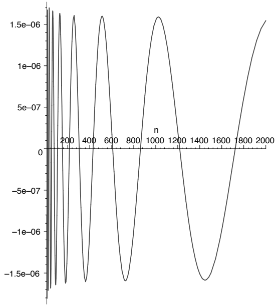
FIGURE 11. Possible ways to obtain the asymptotic mean value in the Bernoulli model from the exact mean value in the Poisson model.

Analysis of Periodicities

The precise analysis of trie parameters demands careful handling of fluctuations. E.g., the expected size (number of nodes) in a trie built over uniform binary sequences is

$$\frac{n}{\log 2}(1 + \epsilon(\lg n)) + o(n),$$

where $\epsilon(x)$ is again a fluctuating function with amplitude less than 10^{-5}.



Analysis of Periodicities

In the three decades since the early 1982 paper, Philippe became a master of the analysis of generating functions, and he went on to characterize many trie parameters under more complicated probabilistic assumptions.

We briefly outline the way that the analysis changes, depending on the branching probabilities $p_1, p_2, \ldots, p_m$ related to the m letters in the alphabet (e.g., $m = 2$ in the binary case)
When the trie is built over uniform binary sequences, the oscillating term arises from the appearance of complex poles, for instance, at

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

Non-Uniform Analysis

Oscillations also arise in the asymptotic growth of parameters of tries built over non-uniform tries (i.e., biased probability model).

E.g., the expected size of a trie with branching probabilities p, q such that $\frac{\log p}{\log q} \in \mathbb{Q}$ is

$$\sim \frac{n}{H}(1 + \epsilon(\ln n)),$$

where ϵ is oscillating with small amplitude, and

$$H = p \log(1/p) + q \log(1/q)$$

is the entropy of the source that generates the strings.

When $\frac{\log p}{\log q} \notin \mathbb{Q}$, the expected size is $\sim n/H$, with no oscillations in the leading term.

"On the Performance Evaluation of Extendible Hashing and Trie Searching" (1983)

In b-tries, there are $\leq b$ strings per leaf (rather than one string per leaf).

The probability that a b-trie on n strings has height $\leq h$ is

$$\exp\left(-\frac{2^{b\,u(n)}2^{-b\delta}}{(b+1)!}\right)(1+O((\log n)^{b+1}/n^{1/b})),$$

where $u(n)$ is the fractional part of $(1+1/b)\log_2 n$, and $\delta = h - \lfloor(1+1/b)\log_2 n\rfloor$, and h is in a central region around $(1+1/b)\log_2 n$.

The average height is $\overline{H}_n = (1+1/b)\log_2 n + P((1+1/b)\log_2 n) + o(1)$ where P is periodic.

The average size is $\overline{S}_n = Q((1+1/b)\log_2 n)n^{1+1/b}(1+O(1/(\log n)^{b-1}))$, where Q is periodic.

Timeline: Asymptotic Analysis of Tries [adapted, PF 2010]

- 1965, De Bruijn, Knuth analyze tries built over uniform strings, $p = q = 1/2$; oscillations exhibited
- 1973: Knuth (TAOCP, vol. 2) discusses biased case
- 1982: PF makes connections among several types of related analysis; towards systematic treatments
- 1986: Fayolle, PF, Hofri study periodicity criterion, see also Schachinger [2000], and Jacquet, Szpankowski, Tang [2001]
- 1990-2000: Convergence to asymptotic regime often *wrongly assumed to be fast*. Caveats by Schachinger (2000).
- 2010: PF, Roux, Vallée, convergence to asymptotic regime is very slow and depends on fine arithmetic properties of probabilistic model.

Expected Size of Trie: *m*-ary Case

- If $\frac{\log p_j}{\log p_k}$ are rational for all $1 \leq j, k \leq m$, expected size has first-order periodicities: $\overline{S}_n = \frac{n}{H} + n\phi(\log n) + O(n^{1-A})$, for $A > 0$, where $H = \sum p_j \log(1/p_j)$ is the entropy of the source.
- If at least one $\frac{\log p_j}{\log p_k}$ is irrational, then expected size is: $\overline{S}_n = \frac{n}{H} + O(n\exp(-\sqrt[\theta]{\log n}))$, for $\theta > 1$. "This is better than $n/(\log n)^a$, any a; much worse than $n^{1-\epsilon}$, any ϵ." ("no oscillation, but poor error term")
- "For remaining 'Liouvillean sources' (rare), error term can come arbitrarily close to $o(n)$."

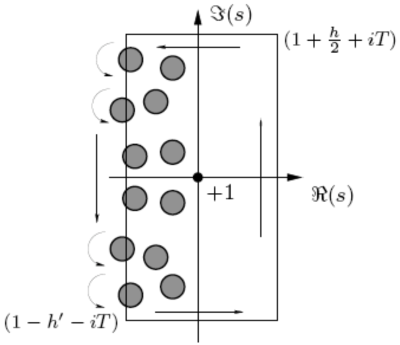
Of course, the set $\left\{(p_1, \ldots, p_m) \mid \text{at least one of } \frac{\log p_j}{\log p_k} \text{ is irrational}\right\}$ has measure 1 in the *m*-ary space of all *m*-tuples. So this is actually the general situation.

Singularity Analysis

The geometry of the poles

$$\frac{1}{1 - p_1^s - \cdots - p_m^s}$$

plays a central role in asymptotic analysis. See especially *Digital Trees and Memoryless Sources: from Arithmetics to Analysis*, PF, M. Roux, B. Vallée, 2010.

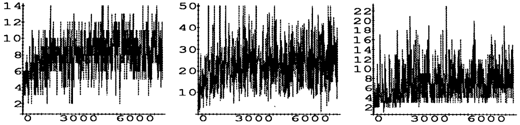


The importance of the geometry during inverse Mellin analysis. Need integration path avoiding poles and estimates of global contributions.

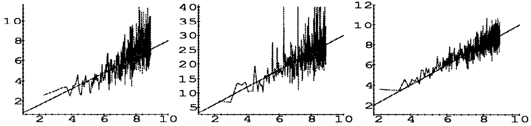
Verification with Data

Philip loved to check theoretical, asymptotic results using empirical sources or simulations. E.g., *The Analysis of Hybrid Tree Structures* by J. Clément, P. Flajolet, B. Vallée [SODA, 1998]

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:

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

Philippe, you are greatly missed.

Your friendship, guidance, and leadership will never be forgotten.