The Height of List-tries and TST

N. Broutin, L. Devroye

2007 Conference on Analysis of Algorithms, AofA 07 DMTCS proc. **AH**, 2007, 271–282

The Height of List-tries and TST

N. Broutin[1] and L. Devroye[1†]

[1] *School of Computer Science, McGill University, 3480 University Street, H3A 2K6 Montreal, Canada.*
Email: {nbrout,luc}@cs.mcgill.ca.

received 17th February 2007, revised 19th January 2008, accepted tomorrow.

We characterize the asymptotics of heights of the trees of (dlB59) and the ternary search trees (TST) of (BS97). Our proof is based on a new analysis of the structure of tries that distinguishes the bulk of the tree, called the *core*, and the long trees hanging down the core, called the *spaghettis*.

Keywords: Tries, branching process, height, de la Briandais, TST.

1 Introduction

Tries are data structures used to manipulate and store strings by taking advantage of the digital character of words. They were introduced by (dlB59). Apparently, the term of *trie* was coined by (Fre60) as a part of the word "retrieval". Their properties and uses are reviewed by (Knu73) and more recently by (Szp01). Consider n sequences of characters (or strings) from an alphabet $\mathcal{A} = \{1, \dots, d\}$. Each one of the sequences carves a path in an infinite rooted d-ary position tree T_∞ where the children of each node are labeled with the characters of $\mathcal{A}$: starting from the root, the characters are used in order to move down the tree. If all the sequences are distinct, the corresponding paths in T_∞ are distinct as well. The trie T_n is defined to be the smallest subtree of T_∞ such that the paths corresponding to the sequences are distinct within T_n.

In this paper, we are interested in tries built from n independent sequences. Each sequence is an infinite sequence of independent and identically distributed (i.i.d.) characters distributed like A, where $\mathbf{P}\{A = i\} = p_i$. We assume without loss of generality that $1 > p_1 \geq p_2 \geq \cdots \geq p_d > 0$. One of the quantities of interest under this model is the height H_n of a of a trie built from n independent such sequences. It is well known that the height satisfies

$$\frac{H_n}{\log n} \xrightarrow[n\to\infty]{} \frac{2}{\log(1/Q)} \quad \text{in probability,} \tag{1}$$

where

$$Q = \sum_{i=1}^{d} p_i^2. \tag{2}$$

†Research of the authors was supported by NSERC Grant A3456 and a James McGill fellowship.

(See R81; Dev84; Pit85; Szp91; Szp01).

The trie is only an *abstract data structure*, that is, it does not specify the implementation (see CFV98; CFV01). The usual implementation of a trie uses an array for the branching structure of a node (Fre60). Although this always ensures constant-time shunting of the strings, the space required may become an issue for large alphabets: many pointers would be left unused. To avoid this, one can replace the array by variable size structures. De la Briandais (1959) proposed to use linked-lists, and we shall call the implementation a *list-trie*. More recently, building on early ideas of (Cla64), (BS97) developed an elegant structure based on binary search trees. It is known as the *bst-trie*, ternary search trie or TST for short.

List-tries and TST aim at a trade-off between storage space and speed, and the access time to children is no longer constant. In particular, the height of the tree and the worst-case search time are different in general. Both these structures may be seen as *high-level* tries whose edges are weighted to reflect the internal *low-level* structure of a node (see Figure 2). This point of view has been taken by (CFV98; CFV01) who thoroughly analyzed these *hybrid* implementations of tries. In particular, they analyzed the average size and average depth. The question of the worst-case search time in hybrid-tries was left open. The worst-case search time is then given by the *weighted height* of the tree, we shall call it the height in the following. This paper addresses the question of the (weighted) height of hybrid tries, and of list-tries and TST, in particular.

2 Weighted tries

2.1 An embedding to construct weighted tries

In this section we propose an embedding to construct the weighted tries. Strictly speaking, since we are interested in the weighted height, we only construct the sequence of weighted depths of the nodes. Note that our embedding is only *one* way to build tries with the desired distribution. We will see in Section 4 that hybrid tries, like list-tries and TST, can be seen as weighted tries. Our construction emphasizes an underlying structure consisting of independent random variables. However, in the coupled tries built from the embedding, the random variables are dependent in general. Our construction goes in two steps: we first give a *shape* to the tree, and then assign the weights to the edges based on the shape. Consider the distribution $\{p_1, \ldots, p_d\}$ over the alphabet $\mathcal{A}$.

THE SHAPE OF THE TRIE. The shape of the trie is simply an ordinary trie: Each string defines an infinite path in T_∞. The *cardinality* N_u of a node $u \in T_\infty$ is the number of strings whose paths in T_∞ intersect u. Then, the trie T_n is constructed by pruning T_∞ down to every node of cardinality at most one. The sequences are distinct with probability one, and the strings define distinct paths in T_∞. Therefore, the trie T_n is almost surely finite. The tree T_n constitutes the *shape* of the weighted trie. We define $E_i = -\log p_i$. For the edge e between u and its i-th child in T_∞, we let $p_e = p_i$ and $E_e = -\log p_e$.

DIFFERENT TYPES OF NODES. There are 2^d *types* of nodes, each type being characteristic of the *branching structure* of the node. The branching structure of every node $u \in T_n$ is described by a d-vector τ_u: if $u_1, \ldots, u_d$ are the d children of u, then we define

$$\tau_u = (\mathbf{1}[N_{u_1} \geq 1], \mathbf{1}[N_{u_2} \geq 1], \ldots, \mathbf{1}[N_{u_d} \geq 1])\,.$$

The vector τ_u indicates which one of the d edges down u are part of some path in T_n, and influences the cost of accessing the children of u.

THE WEIGHTS. Consider a sequence of random vectors $\{\mathcal{Z}^\tau : \tau \in \{0,1\}^d\}$, where $\mathcal{Z}^\tau = (Z_1^\tau, \dots, Z_d^\tau)$. The vectors Z^τ describe the cost of accessing the children of the nodes. We assume that

- for all $\tau \in \{0,1\}^d$, $\max\{Z_i^\tau : 1 \le i \le d\} \le d$, and
- for all types τ such that τ is a permutation of $(1,0,\dots,0)$, $Z^\tau = (1,\dots,1)$.

Each node of T_∞ is assigned an independent copy of the whole sequence. Consider a node $u \in T_\infty$, and its sequence $\{\mathcal{Z}^\tau\}$. Weights are associated with the edges to u's children based on its type τ_u. The edge e_i between u and its i-th child in T_∞ is given the weight

$$Z_{e_i} \;=\; Z_i^{\tau_u} \;=\; \sum_{\tau\in\{0,1\}^d} Z_i^\tau \cdot \mathbf{1}[\tau_u = \tau].$$

We use the notations Z_i^τ and Z_e interchangeably. It should always be clear whether a subscript refers to an index or an edge. Let $\pi(u)$ be the set of edges on the path from u up to the root in T_∞. The *weighted depth* of a node u is defined by $D_u = \sum_{e\in\pi(u)} Z_e$. A tree that may be constructed by this procedure is called a *weighted trie*. We are interested in the weighted height of T_n:

$$H_n = \max\{D_u : u \in T_n\} = \max\{D_u : N_u \ge 2\} + 1.$$

Surprisingly, the first asymptotic term of H_n depends on two parameters only: the distribution $\{p_1,\dots,p_d\}$, and $\mathcal{Z} = \mathcal{Z}^{(1,\dots,1)}$. In particular, the first order asymptotics of H_n stays the same if we modify $\{\mathcal{Z}^\tau, \tau \in \{0,1\}^d\}$ in such a way that $\mathcal{Z}$ remains unchanged. This may be explained using the structure of a trie.

2.2 The structure of a trie

The profile of ordinary tries can be explained by distinguishing a so-called *core*, that constitutes the bulk of the trie, and *spaghetti*-like trees hanging down the core (BD07a). This distinction is justified by the following Lemma, which is at the heart of our proofs:

Lemma 1 *Let T_n be a random trie. Let $m = m(n) \to \infty$ such that $m = o(\log n)$. There exists $\omega = \omega(n) \to \infty$, as $n \to \infty$ such that:*
(a) with probability $1 - n^\omega$, all the nodes u with $N_u \ge \log^2 n$ have $\tau_u = (1,\dots,1)$,
(b) the maximum number of nodes with $N_u \ge m(n)$ and $\tau_u \ne (1,\dots,1)$ on a path down the root is $o(\log n)$ with probability $1 - n^{-\omega}$, and
(c) the maximum number of nodes with $N_u \le m(n)$ and degree at least two on a path down the root is at most $m(n) = o(\log n)$.

THE CORE OF A TRIE. What we call the core here should not be confused with the graph-theoretic core, which happens to be empty for trees (see e.g., JŁR00). The core of the trie is defined to be the set of nodes $u \in T_\infty$ for which $N_u \ge m(n)$, for $m(n) \to \infty$ and $m(n) = o(\log n)$. The core is denoted by $\mathcal{C}$. Since $m \to \infty$, a nodes in the core has type $\tau = (1,\dots,1)$ with probability $1 - o(1)$. As a consequence, in a weighted trie, the distribution of weights in the core should be closely approximated by $\mathcal{Z} = \mathcal{Z}^{(1,\dots,1)}$. The core can be described by its *logarithmic profile*

$$\phi(\alpha,t) = \lim_{n\to\infty} \frac{\log \mathbf{E}P_m(t\log n, \alpha\log n)}{\log n} \qquad \forall t,\alpha > 0, \tag{3}$$

where $P_m(k,h)$ denotes the number of nodes u, k levels away from the root with $N_u \geq m(n)$ and $D_u \geq h$. In other words, assuming for now that the limit in (3) exists, we have $\mathbf{E}P_m(t\log n, \alpha \log n) = n^{\phi(\alpha,t)+o(1)}$, as $n \to \infty$. The function $\phi(\cdot,\cdot)$ is characterized in Theorem 2.

HANGING SPAGHETTIS. The *spaghettis* are the trees remaining when pulling out the core from the trie. They lie in the part of the trie where the nodes do not have d children any more: the types of the nodes may take all the values in $\{0,1\}^d$. However, the weighted height of a *long* spaghetti is close to its unweighted height, or number of levels. To see this, observe that the nodes not in the core have cardinality at most $m(n) = o(\log n)$, so each spaghetti stores at most $m(n)$ sequences. Each time the type τ is not a permutation of $(1,0,\ldots,0)$, i.e., the node is truly branching, at least one string is put aside from the longest path. This can happen at most $o(\log n)$ times, and hence the heights with and without the branching nodes differ by at most $o(\log n)$. If the number of levels is $\Theta(\log n)$, as is the case for the highest ones, the difference is negligible.

Both the core and the spaghettis contribute significantly to the height of a weighted trie. By figuring out what the core looks like, we can determine *when* the spaghettis take over. Roughly speaking, we then know if an edge's weight can be approximated by a component of $\mathcal{Z}$ or simply remain unweighted. The main result of this paper is the following theorem.

Theorem 1 *Consider a weighted trie built from n independent sequences consisting of i.i.d. characters distributed as A, where $\mathbf{P}\{A=i\} = p_i$, $1 \leq i \leq d$. Let H_n be its weighted height. Let $\phi(\alpha,t)$ be the logarithmic weighted profile of the core of T_n. Let*

$$c = \sup\left\{\alpha + \frac{\phi(\alpha,t)}{\log(1/Q)} : \phi(\alpha,t) \geq 0\right\},$$

where $Q = \sum_{i=1}^d p_i^2$. Then $H_n = c\log n + o(\log n)$ in probability, as $n \to \infty$.

Remark. (a) The respective contributions of the core and spaghettis are α and $\phi(\alpha,t)/\log(1/Q)$.
(b) The joint between the core and the spaghettis along the longest path, i.e., the level at which the longest path leaves the core, occurs far from the bottom of the core (see Figure 1).

3 The height of weighted tries

3.1 The shape of the core

Consider a weighted trie defined as in section 2. We consider $m = m(n) \to \infty$ with $m(n) = o(\log n)$. Let $\mathscr{L}_k$ be the set of nodes k levels away from the root in T_∞. Let $P_m(k,h)$ be the number of nodes $u \in \mathscr{L}_k$ with $D_u \geq h$ and $N_u \geq m$. Since $m \to \infty$, for n large enough, we have $m \geq b$ and

$$P_m(k,h) = \sum_{u \in \mathscr{L}_k} \mathbf{1}[N_u \geq m, D_u \geq h].$$

The asymptotic properties of the expected profile are directly tied to large deviation theory (DZ98). The random vector of interest here is $(Z,E) = (Z_K, -\log p_K)$, where K is uniform in $\{1,\ldots,d\}$ and $\mathcal{Z} = \mathcal{Z}^{(1,\ldots,1)} = (Z_1,\ldots,Z_d)$. For $\lambda,\mu \in \mathbb{R}$, the associated *generating function of the cumulants* is

$$\Lambda(\lambda,\mu) = \log \mathbf{E}\left[e^{\lambda Z + \mu E}\right].$$

Then, for $x, y \in \mathbb{R}$, we define the convex dual $\Lambda^\star$ of Λ by,

$$\Lambda^\star(x,y) = \sup_{\lambda,\mu}\{\lambda x + \mu y - \Lambda(\lambda,\mu)\}.$$

Theorem 2 *Let $m = m(n) \to \infty$ with $m = o(\log n)$. Let $k \sim t \log n$ and $h \sim \alpha \log n$ for some positive constants t and α. Let*

$$\phi(\alpha,t) = t\log d - t \cdot \inf\left\{\Lambda^\star(x,y) : x > \frac{\alpha}{t}, y < \frac{1}{t}\right\}. \tag{4}$$

If $\phi(\alpha,t) > -\infty$, then $\mathbf{E}P_m(k,h) = n^{\phi(\alpha,t)+o(1)}$, as $n \to \infty$.

Remarks. (a) Observe that Theorem 2 justifies the definition of $\phi(\cdot,\cdot)$ in (3).
(b) The constraint that $m(n)$ is $o(\log n)$ is only used to ensure that the spaghettis each store $o(\log n)$ only. Theorem 2 still holds with $m(n)$ as large as $n^{o(1)}$.

Unlike the profile of *unweighted* tries (Dev02; Dev05; PHNS06), that of *weighted* tries does not seem concentrated. However, it is log-concentrated in the sense of the following theorem, and the true profile $P_m(k,h)$ is close enough to $\mathbf{E}P_m(k,h)$:

Theorem 3 *Let $m = m(n) \to \infty$ as $n \to \infty$ such that $m = o(\log n)$. Let $k \sim t \log n$ and $h \sim \alpha \log n$ for some positive constants t and α. Then, for all $\epsilon > 0$, as $n \to \infty$,*

$$\mathbf{P}\left\{P_m(k,h) \le n^{\phi(t,\alpha)-\epsilon}\right\} \xrightarrow[n\to\infty]{} 0, \quad \textit{and} \quad \mathbf{P}\left\{P_m(k,h) \ge n^{\phi(t,\alpha)+\epsilon}\right\} \le n^{-\epsilon+o(1)}.$$

3.2 How long is a spaghetti?

The behavior of the spaghettis is radically different from the one observed in the core. This is because the number of strings stored by a single spaghetti is at most $m(n)$. We first look at the profile, not of a single trie, but of a forest of independent tries.

Let $T^1, T^2, \dots, T^n$ be n independent tries. We assume that T^i is a weighted trie on $m_i = m_i(n)$ sequences generated by a memoryless source with distribution $\{p_1, \dots, p_d\}$. Also, we assume that for all i, $m/d \le m_i \le m$. The roots of T^i, $1 \le i \le n$, all lie at level zero. Then, we let $P^s(k,h)$ count the number of nodes u at level k with $D_u \ge h$ lying in any T^i. Since T^i is a trie, we only count the nodes for which $N_u \ge 2$. We are interested in $\mathbf{E}P^s(k,h)$ when $k \sim \rho \log n$ and $h \sim \gamma \log n$. Recall that the difference between the number of layers and the weighted height is at most $dm(n) = o(\log n)$. Hence, $\mathbf{E}P_m(k,h) = o(1)$ if $h \ge k + \Omega(\log n)$. Also, when $h \le k + o(\log n)$, only the number of levels is relevant. For n large enough that $m/d \ge 2$, by linearity of expectation, we have

$$nQ^k \le \mathbf{E}P_m(k,h) \le nm^2Q^k,$$

since two strings are identical up to the k-th character with probability Q^k. In other words, we have

Theorem 4 *Let T^i, $1 \le i \le n$, be a forest of n independent tries. Let T^i store $m_i = m_i(n)$ sequences. Assume that $m/d \le m_i \le m$ for all $1 \le i \le n$. Let $k \sim \rho \log n$ and $h \sim \gamma \log n$, as $n \to \infty$, for positive constants ρ and γ. Then,*

$$\mathbf{E}P^s(k,h) = \begin{cases} n^{1+\rho\log Q+o(1)} & \textit{if} \quad \rho \ge \gamma \\ o(1) & \textit{otherwise.} \end{cases}$$

as $n \to \infty$.

So the logarithmic profile of our forest is $1 + \rho \log Q$. Theorem 5 claims that the deepest node occurs when this logarithmic profile vanishes. Let $H^1, \ldots, H^n$ be the weighted heights of $T^1, \ldots, T^n$, respectively, and define $S_n = \max\{H^i : 1 \le i \le n\}$. Then:

Theorem 5 *Assume that* $p_1 < 1$. *Assume that* $m(n) \to \infty$ *and* $m(n) = o(\log n)$. *Let Then,* $S_n \sim \log_{1/Q} n$ *in probability, as* $n \to \infty$. *Furthermore, for every* $\epsilon > 0$, *there exists* $\delta > 0$ *such that, as* $n \to \infty$,

$$\mathbf{P}\left\{\frac{S_n}{\log n} \ge \frac{1}{\log(1/Q)} + \epsilon\right\} = O\left(n^{-\delta}\right). \tag{5}$$

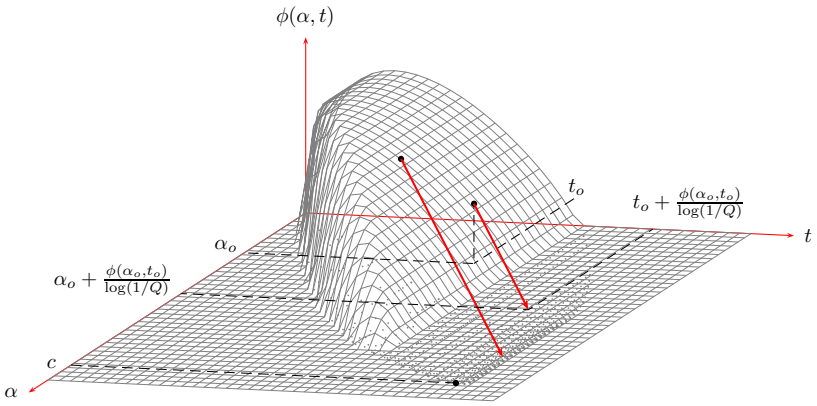

Fig. 1: A geometric interpretation for the height: each point $(\alpha, t, \phi(\alpha, t))$ of the logarithmic profile of the core throws a line whose direction is given by $(1, 1, \log Q)$. The line intersects the plane $\phi = 0$ at $(\alpha - \phi(\alpha, t)/\log Q, t - \phi(\alpha, t)/\log Q, 0)$. The constant c is the largest coordinate of one of these point measured along the α-axis. So spaghettis correspond to parallel rays that are originated on the profile. In particular the ray leading to the furthest projection (longest path) is born far from the deepest level of the core.

3.3 Projecting the profile of the core

In this section, we give a sketch of the proof of Theorem 1. Consider a weighted trie T_n. The spaghettis are rooted at a node $u \in \partial\mathcal{C}$, the external node-boundary of the core $\mathcal{C}$ in T_n (the nodes $u \in \partial\mathcal{C}$ are the children of some node v in the core, but are not themselves in the core). Then, if we write W_u for the height of the subtree rooted at u, we have

$$H_n = \max\{D_u + W_u : u \in \partial\mathcal{C}\} = \max_{h,k}\{h + W_u : u \in \partial C \cap \mathscr{L}_k, D_u \ge h\},$$

where the nodes in ∂C have been split into groups depending on their level k and weighted depth h. Then, we can rewrite

$$H_n = \sup_{h,k}\{h + \max\{W_u : u \in \partial\mathcal{C}, D_u \ge h, u \in \mathscr{L}_k\}\}.$$

We have thus separated the contributions of the core from that of the spaghettis,

$$h \qquad \text{and} \qquad \max\{W_u : u \in \partial\mathcal{C}, D_u \ge h, u \in \mathscr{L}_k\},$$

respectively. Only the latter expression requires more thought. The set of nodes $u \in \partial C \cap \mathscr{L}_k$ with $D_u \geq h$ roughly contains roughly $n^{\phi(\alpha,t)+o(1)}$ nodes by Theorem 2. Hence $\max\{W_u : u \in \partial\mathcal{C}, D_u \geq h, u \in \mathscr{L}_k\}$ is $S_{n'}$ with $n' = n^{\phi(\alpha,t)+o(1)}$, that is, $\phi(\alpha,t)/\log(1/Q)$ by Theorem 5. This explains why $H_n \sim c \log n$, where

$$c = \sup\left\{\alpha + \frac{\phi(\alpha,t)}{\log(1/Q)}\right\}.$$

Complete proofs of the theorems can be found in (Bro07) or (BD07b).

Theorem 1 can be interpreted by projecting the logarithmic profile $\phi(\alpha,t)$ on the horizontal plane going through the origin. See Figure 1.

4 Application: the height of hybrid tries

Let $\mathcal{A} = \{1,\dots,d\}$ be the alphabet. Let $\{A^i, 1 \leq i \leq n\}$ be the n strings. In the following, we distinguish the *nodes* that constitute the high-level trie structure from the *slots* which make the low-level structure of a node, whether this latter be a linked-list or a binary search tree.

The high-level tree between the nodes of a hybrid trie is just the ordinary trie. Only the low-level structure of the nodes differ from the array-based implementation. Consider a node $u \in T_\infty$. The subtree rooted at u stores a subset of the strings A^i, $1 \leq i \leq n$. Let $\mathcal{N}_u \subset \{1,\dots,n\}$ be the set of their indices, and write $N_u = |\mathcal{N}_u|$. As in ordinary tries, for a node u at level k in T_∞, only the k-th characters of each sequence are used. Moreover, only set $\mathcal{A}_u \subset \mathcal{A}$ matters, which consists of the characters appearing at the k-th position in the sequences A^i, $i \in \mathcal{N}_u$. In an hybrid trie, the order in which the strings are used to build the trie matters, and we let σ_u be the permutation $\mathcal{A}_u$ where the characters are ordered by first appearance. The internal structure of the node u is built by successive insertions of the elements of σ_u into an originally empty linked list, or binary search tree. For list-tries we then have a random linked list on $\mathcal{A}_u$ and for TST a binary search tree on $\mathcal{A}_u$. This is shown in Figure 2. The costs of branching to a subtree is then given by the number of edges to cross *inside the node* to the subtree. The distribution clearly depends on the type τ_u of node u, and when $|\mathcal{A}_u| = 1$, the cost is simply 1. This explains why hybrid tries are weighted tries in the sense of Section 2.

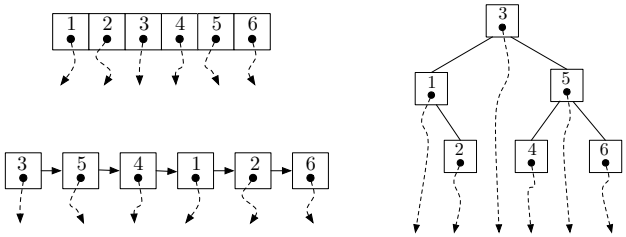

Fig. 2: The different node structures used for the standard (top-left), list (bottom-left) and bst-trie (right) when the order of appearance of the characters is 3, 5, 4, 1, 2, 6. The dashed arrows represent the pointers to further levels of the trie.

4.1 List-tries

In the list-trie of (dlB59), the cost of branching to a character a is just the index of a is the permutation σ_u. For every node u, for which $\mathcal{A}_u = \mathcal{A}$, σ_u is distributed as the sequence (in order) of first appearance of characters in an infinite string generated by the source. This fully describes the distribution of $\mathcal{Z} =$

$\mathcal{Z}^{(1,\dots,1)} = (Z_1, \dots, Z_d)$. Then Z_i is the index of i in σ, and $(Z, E) = (Z_K, -\log p_K)$, where K is uniform in $\{1, \dots, d\}$.

Theorem 6 *Let H_n be the weighted height of a list-trie on n sequences generated by the memoryless source with probabilities $\{p_1, \dots, p_d\}$. Then, $H_n \sim c \log n$ in probability, as $n \to \infty$, where*

$$c = \sup \left\{ \alpha + \frac{\phi(t, \alpha)}{\log(1/Q)} : \alpha, t > 0 \right\},$$

and $\phi(\cdot, \cdot)$ is the logarithmic profile of the trie weighted with (Z, E) is described above.

It seems difficult to explicitly obtain $\phi(\alpha, t)$ for general $\{p_1, \dots, p_d\}$.

Example: symmetric list-tries. We assume here that $p_1 = p_2 = \cdots = p_d$. Then, the permutation σ is just a uniform random permutation. For any $\lambda, \mu \in \mathbb{R}$, we have

$$\Lambda(\lambda, \mu) \;=\; \log \mathbf{E}\left[e^{\lambda Z}\right] + \mu \log d \;=\; \log \left(\sum_{i=1}^{d} e^{i\lambda} \right) + (\mu - 1) \log d.$$

Then, $\Lambda^\star(x, y) = \sup_{\lambda,\mu}\{\lambda x + \mu y - \Lambda(\lambda, \mu)\}$. For $x \in [1, d]$, there exists $\lambda = \lambda(x)$ such that

$$x = \frac{\partial \Lambda(\lambda, \mu)}{\partial \lambda} = \frac{\sum_{i=1}^{d} i e^{i\lambda}}{\sum_{i=1}^{d} e^{i\lambda}}. \tag{6}$$

Therefore, we have

$$\Lambda^\star(x, y) = \begin{cases} \lambda x - \log \left(\sum_{i=1}^{d} e^{i\lambda} \right) + \log d & \text{if } x \in [1, d], y = \log d \\ \infty & \text{otherwise.} \end{cases}$$

For instance, for $d = 2$, we can find an explicit expression for $\Lambda^\star$: for $x \in [1, d]$,

$$\Lambda^\star(x, \log 2) = x \log \left(\frac{1 - x}{x - 2} \right) - \log \left(\frac{1 - x}{2 - x} + \left(\frac{1 - x}{2 - x} \right)^2 \right) + \log 2.$$

With more manipulation, we obtain for this specific example

$$c = c(d) = \frac{\log \left(\sum_{i=1}^{d} d^i \right)}{\log^2 d} \sim \frac{d}{\log d},$$

for large d. Numerical values for $c = c(d)$ can be found in Table 1. Observe that $c(d)$ is not monotonic in d, and that the minimum height is reached for an alphabet of size four.

d	2	3	4	5	10	20
$c(d)$	3.72931…	3.03539…	3.03304…	3.19269…	4.36281…	6.68187…

Tab. 1: Some numerical values of $c = c(d)$ characterizing the height of symmetric list-tries.

4.2 Ternary search trees

In the ternary search trees introduced by (BS97), the implementation of a node uses a binary search tree. The cost of branching to a character $i \in \mathcal{A}$ at a node u is one plus the depth of i in the binary search built from the (non-uniform) random permutation σ_u. When the node u is of type $\tau_u = (1, \dots, 1)$, the permutation σ_u is distributed as σ, the ordered list of first appearances of characters in an infinite string generated by the memoryless source with distribution $\{p_1, \dots, p_d\}$.

Let Z_i be distributed as the depth of i in the binary search tree built from σ. Then, $\mathcal{Z}$ is distributed as $(Z_1, \dots, Z_d)$ and $(Z, E) = (Z_K, -\log p_K)$, where K is uniform in $\{1, \dots, d\}$. By Theorem 1, we obtain:

Theorem 7 *Let H_n be the weighted height of a TST on n sequences. Let*

$$c = \sup\left\{\alpha + \frac{\phi(\alpha, t)}{\log(1/Q)} : \alpha, t > 0\right\},$$

where $\phi(\alpha, t)$ is the logarithmic profile of the core of a trie weighted by (Z, E) described above. Then, $H_n \sim c \log n$ in probability, as $n \to \infty$.

The random vector (Z, E) seems complicated to describe for general distributions $p_1, \dots, p_d$. Some parameters like the average value and the variance of Z_i, $1 \le i \le d$, have been studied by (CFV98; CFV01) and (AC06).

Example: Symmetric TST. We assume here that $p_1 = p_2 = \cdots = p_d$. In this case, the permutation σ is just a uniform random permutation. Hence, Z_i is the depth of the key i in a random binary search tree. Observe that unlike in the case of list-tries, the Z_i, $1 \le i \le d$, do *not* have the same distribution. This is easily seen, since, for instance as $d \to \infty$, $\mathbf{E}Z_1 \sim \log d$ whereas $\mathbf{E}Z_{\lfloor d/2 \rfloor} \sim 2 \log d$. However, we are only interested in the distribution of Z, that is, the depth of a uniform random node. This distribution is known exactly, and is due to (Lyn65) and (BS84):

$$\mathbf{P}\{Z = k\} = \frac{2^{k-1}}{d \cdot d!} \sum_{j=k}^{d} \begin{bmatrix} d \\ j \end{bmatrix}, \tag{7}$$

where $\begin{bmatrix} n \\ k \end{bmatrix}$ denotes the Stirling number of the first kind with parameter n and k (see SF96; Mah92). Using (7), it is possible to compute the cumulant generating function Λ, and $\phi(\alpha, t)$. Observe that the when $d = 2$, the TST is equivalent to the list-tries. In general, one obtains

$$\begin{aligned} c = c(d) &= \frac{1}{\log d} + \frac{1}{\log^2 d} \log\left(\sum_{i=1}^{d} \sum_{j=i}^{d} \frac{2^{i-1}}{d \cdot d!} \begin{bmatrix} d \\ j \end{bmatrix} d^i\right) \\ &= \frac{1}{\log d} + \frac{1}{\log^2 d} \log\left(\frac{(2d) \cdot (2d+1) \cdots (3d-1) - d!}{d!(2d-1)}\right) \sim \frac{d \log(27/4)}{\log^2 d} \end{aligned}$$

(see, e.g., Mah92, p. 79). Numerical values for the constant $c = c(d)$ are given in Table 2.

d	2	3	4	5	10	20
$c(d)$	3.72931…	2.89698…	2.72474…	2.70765…	3.05001…	3.88868…

Tab. 2: Some numerical values of $c = c(d)$ characterizing the height of symmetric ternary search trees.

5 Concluding remarks

Theorem 1 is not the most general one can obtain. Also, the weights for non-branching nodes $\mathcal{Z}^\sigma$, with σ a permutation of $(1, 0, \ldots, 0)$, may also be random instead of unit values. Then, the height satisfies

$$\frac{H_n}{\log n} \xrightarrow[n\to\infty]{} \sup\{\alpha + \gamma\phi(\alpha, t) : \alpha, t > 0\} \quad \text{in probability,}$$

where γ is a constant depending on $\{p_1, \ldots, p_d\}$ and the weights $\mathcal{Z}^\sigma$. Finally, similar results hold for b-tries, where the leaves are allowed to contain up to b strings. These extensions and complete proofs of the theorems can be found in (Bro07) or (BD07b).

6 Aknowledgement

We are very grateful to Julien Clément for bringing this problem to our attention and for his insightful comments.

References

[AC06] M. Archibald and J. Clément. Average depth in binary search tree with repeated keys. In *Fourth Colloquium on Mathematics and Computer Science*, pages 309–320, 2006.

[BD07a] N. Broutin and L. Devroye. The core of a trie. Manuscript, 2007.

[BD07b] N. Broutin and L. Devroye. Weighted height of random tries. Manuscript, 2007.

[Bro07] N. Broutin. *Shedding New Light on Random Trees*. Phd thesis, McGill University, Montreal, 2007.

[BS84] G.G. Brown and B.O. Shubert. On random binary trees. *Mathematics of Operations Research*, 9:43–65, 1984.

[BS97] J. L. Bentley and R. Sedgewick. Fast algorithm for sorting and searching strings. In *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 360–369, 1997.

[CFV98] J. Clément, P. Flajolet, and B. Vallée. The analysis of hybrid trie structures. In *9th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 531–539, Philadelphia, PA, 1998. SIAM Press.

[CFV01] J. Clément, P. Flajolet, and B. Vallée. Dynamical source in information theory: a general analysis of trie structures. *Algorithmica*, 29:307–369, 2001.

[Cla64] H. A. Clampett. Randomized binary searching with tree structures. *Communications of the ACM*, 7(3):163–165, 1964.

[Dev84] L. Devroye. A probabilistic analysis of the height of tries and of the complexity of triesort. *Acta Informatica*, 21:229–237, 1984.

[Dev02] L. Devroye. Laws of large numbers and tail inequalities for random tries and PATRICIA trees. *Journal of Computational and Applied Mathematics*, 142:27–37, 2002.

[Dev05] L. Devroye. Universal asymptotics for random tries and PATRICIA trees. *Algorithmica*, 42:11–29, 2005.

[dlB59] R. de la Briandais. File searching using variable length keys. In *Proceedings of the Western Joint Computer Conference, Montvale, NJ, USA*. AFIPS Press, 1959.

[DZ98] A. Dembo and O. Zeitouni. *Large Deviation Techniques and Applications*. Springer Verlag, second edition, 1998.

[Fre60] E. Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, 1960.

[JŁR00] S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs*. Wiley, New York, 2000.

[Knu73] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, Reading, MA, 1973.

[Lyn65] W.C. Lynch. More combinatorial properties of certain trees. *Computing J.*, 7:299–302, 1965.

[Mah92] H. Mahmoud. *Evolution of Random Search Trees*. Wiley, New York, 1992.

[PHNS06] G. Park, H.K. Hwang, P. Nicodème, and W. Szpankowski. Profile of tries. Manuscript, 2006.

[Pit85] B. Pittel. Asymptotic growth of a class of random trees. *The Annals of Probability*, 13:414–427, 1985.

[Rég81] M. Régnier. On the average height of trees in digital search and dynamic hashing. *Information Processing Letters*, 13:64–66, 1981.

[SF96] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithm*. Addison-Wesley, 1996.

[Szp91] W. Szpankowski. On the height of digital trees and related problems. *Algorithmica*, 6:256–277, 1991.

[Szp01] W. Szpankowski. *Average Case Analysis of Algorithms on Sequences*. Wiley, New York, 2001.