The height and size of random hash trees and random pebbled hash trees

Luc Devroye
School of Computer Science
McGill University
Montreal, Canada H3A 2K6
luc@cs.mcgill.ca

Abstract. The random hash tree and the N-tree were introduced by Ehrlich in 1981. In the random hash tree, n data points are hashed to values $X_1, \ldots, X_n$, i.i.d. random variables taking values that are uniformly distributed on $[0,1]$. Place the X_i's in n equal-sized buckets as in hashing with chaining. For each bucket with at least two points, repeat the same process, keeping the branch factor always equal to the number of bucketed points. If H_n is the height of tree obtained in this manner, we show that $H_n/\log_2 n \to 1$ in probability. In the random pebbled hash tree, we remove one point randomly and place it in the present node (as with the digital search tree modification of a trie) and perform the bucketing step as above on the remaining points (if any). With this simple modification,

$$\frac{H_n}{\sqrt{\frac{2\log n}{\log\log n}}} \to 1$$

in probability. We also show that the expected number of nodes in the random hash tree and random pebbled hash tree is asymptotic to $2.3020238\ldots n$ and $1.4183342\ldots n$ respectively.

Keywords and phrases. Data structures, probabilistic analysis, hashing with chaining, hash tables, N-trees, random hash trees, expected complexity.

CR Categories: 3.74, 5.25, 5.5.

Research of the author was sponsored by NSERC Grant A3456 and by FCAR Grant 90-ER-0291.

1. Introduction.

In this paper, we analyze the height of the random hash tree (Ehrlich, 1981) defined as follows. We are given $X_1, \ldots, X_n$, n independent uniform $[0,1]$ random numbers. If $n > 1$, we partition $[0,1]$ into n equal intervals of length $1/n$ each, and place all points in the intervals. Let $N_1, \ldots, N_n$ be the cardinalities of the intervals (thus, $\sum_i N_i = n$). Repeat the partition process for every interval containing at least two points, and keep going until no further divisions are possible. The root node represents $[0,1]$, and each node represents a given interval. All internal nodes have at least two of the X_i's, while all leaf nodes have one or zero of the X_i's. If this structure is to be used for storing the data, then two important quantities are the number of nodes in the tree, S_n and the height of the tree, H_n. The former quantity obviously measures the storage, while the latter is the worst-case search time. In addition, S_n is also proportional to the time needed to construct the hash tree. The model is appropriate for situations in which a hash function can be constructed that delivers a uniform $[0,1]$ random variate. This may be a debatable hypothesis.

The distributive partitioning method invented by Dobosiewicz (1978) led Ehrlich (1981) to define the N-tree. In N-trees, the X_i's are unrestricted on the real line, and given that a node has $k \geq 2$ of the X_i's, it spawns k equal child intervals of $[\min_i X_i, \max_i X_i]$. Note that there are always at least two nonempty subintervals (the first and the last one), so that the size of the N-tree is limited to $n(n+1)/2$. A basic study of N-trees and random hash trees was performed by Tamminen (1983). The results of Tamminen will be summarized in the next section. However, Tamminen did not study H_n. Hashing with several levels of buckets has been known since Fagin, Nievergelt, Pippenger and Strong (1979) as extendible hashing. Its analysis was subsequently refined in several papers, including papers by Tamminen (1981) and Flajolet (1983). However, these structures differ fundamentally from the trees studied here.

The random hash tree and its modifications studied here are vaguely related to random tries (see Pittel, 1985, for the main properties). We will show that the height H_n of the random hash tree is with high probability close to $\log_2 n$, which is rather disappointing. The reason for this phenomenon is the same reason why random binary tries have height close to $2\log_2 n$. This prompted us to consider a modification similar to the modification of a trie into the digital search tree of Coffman and Eve (1970): if n points belong to an interval associated with a node, remove one point uniformly at random and place it in the node ("pebble" the node). If $n > 1$, the interval is partitioned equally into $n-1$ child subintervals and the $n-1$ remaining points are placed in their subintervals. This process is repeated until all leaf nodes have cardinality zero or one.

The tree thus obtained is called the random pebbled hash tree. We will show that this minor modification causes a major improvement in H_n, which is with high probability close to $\sqrt{2 \log n / \log\log n}$.

Assuming that each bucketing operation is available at unit cost, the expected time for unsuccessful and for successful search (assuming all points are equally likely to be probed for) is $O(1)$ for both random hash tree and random pebbled hash tree, but the expected worst-case search time for the latter tree is much better than for classical hash structures in view of the behavior of H_n. For example, for standard hashing with chaining under the above model, $M_n \sim \log n / \log\log n$ in probability (Gonnet, 1981; see Devroye, 1985, for distributions with a bounded density), where M_n is the worst-case search time (or, equivalently, the maximal size of any chain).

The random pebbled hash tree is also superior to random binary search trees, where the height is in probability of the order of $\log n$ (under a simpler computational model, however). It also compares favorably with fusion trees (Fredman and Willard, 1990) for standard dictionary operations. Unfortunately, hash trees are not appropriate without modifications for fully dynamic situations. A brief section is devoted to this issue.

There are hash structures with better expected worst-case search and insert times. Azar, Broder, Karlin and Upfal (1994) have shown that the worst chain length in multihashing with $d > 1$ hash functions and insertion into the shortest chain leads to a maximum occupancy of about $\log_d \log n$ with high probability. This leads to expected worst-case search times about $d \log_d \log n$, which are much better than with random pebbled hash trees. However, multihashing is not an option when the table is to be used for sorting and order-preserving hash functions are called for. Also, the performance of multihashing is easily matched by the BBST structure (bucketing followed by a binary search tree) discussed below.

For a survey of known results on hashing and tries, we refer to Gonnet and Baeza-Yates (1991) and Vitter and Flajolet (1990). Throughout the paper, $B(n,p)$ denotes a binomial (n,p) random variable.

2. Survey of known results on N-trees and random hash trees.

Assume that the X_i's are uniformly distributed on $[0,1]$. Then Tamminen (1983) shows the following for N-trees:

A. $\sup_n \frac{\mathbf{E}S_n}{n} \le 2$.

B. $1.64 \le \liminf_n \frac{\mathbf{E}S_n}{n} \le \limsup_n \frac{\mathbf{E}S_n}{n} \le 1.70$.

C. Let A_i be the depth of X_i in the N-tree so that $(1/n)\sum_{i=1}^n A_i$ is the average successful search time. Then

$$\mathbf{E}\left\{\frac{1}{n}\sum_{i=1}^n A_i\right\} \le 2$$

for all n.

D. $1.71 \le \liminf_n \mathbf{E}\left\{\frac{1}{n}\sum_{i=1}^n A_i\right\} \le \limsup_n \mathbf{E}\left\{\frac{1}{n}\sum_{i=1}^n A_i\right\} \le 1.80$.

If the X_i's have a density f bounded by $\|f\|_\infty$, then for the random hash tree, Tamminen (1983) obtained the following results:

A. $\sup_n \frac{\mathbf{E}S_n}{n} \le 3\|f\|_\infty$.

B. $\limsup_n \frac{\mathbf{E}S_n}{n} \le 4$.

C. Let A_i be the depth of X_i in the random hash tree so that $(1/n)\sum_{i=1}^n A_i$ is the average successful search time. Then

$$\limsup_n \mathbf{E}\left\{\frac{1}{n}\sum_{i=1}^n A_i\right\} \le 4\ .$$

Note in particular that the asymptotic bounds in parts B and C do not depend upon the density. Tamminen (1983) also offers heuristic arguments for densities with unbounded support and so-called hybrid trees, where the first level is split as in an N-tree and all other splits are as in a random hash tree.

3. The height of the random hash tree.

In this section, we assume that $X_1, \ldots, X_n$ are i.i.d. and have common density f on $[0,1]$. Let H_n be the height of the random hash tree. We show the following.

THEOREM 1. *For any increasing sequence a_n (however fast), there exists a density f for which $\mathbf{P}\{H_n \geq a_n\} \to 1$ as $n \to \infty$.*

PROOF. Let F be the distribution function for f, supported on $[0,1]$. Let N_1 be the number of points in $[0, 1/n]$, N_2 the number of points in $[0, 1/n^2]$ and so forth. Clearly,

$$[N_{a_n} \geq 2] \subseteq [H_n \geq a_n] \ .$$

But, putting $p = 1 - F(1/n^{a_n})$, we see that

$$\mathbf{P}\{N_{a_n} < 2\} = p^n + np^{n-1} \leq (n+1)e^{-(n-1)F(1/n^{a_n})} \to 0$$

if $nF(1/n^{a_n})/\log n \to \infty$. It suffices to pick F such that $F(1/n^{a_n}) = 1/\sqrt{n}$ for all n. Then let f be a histogram with breakpoints at $1/n^{a_n}$. Conclude that $\mathbf{P}\{H_n \geq a_n\} \to 0$. $\square$

THEOREM 2. *When f is the uniform distribution on $[0,1]$, we have*

$$\frac{H_n}{\log_2 n} \to 1 \ \textit{in probability} \ .$$

PROOF. For a lower bound, it suffices to consider a subtree of the random pebbled hash tree. To do so, we consider only those nodes at depth one that contain precisely two points. Let L be the number of these nodes. Observe that

$$\mathbf{E}\{L\} = (n-1)\mathbf{P}\{B(n-1,1/(n-1)) = 2\} = (n-1)\binom{n-1}{2}\frac{1}{(n-1)^2}\left(\frac{n-2}{n-1}\right)^{n-2} \sim \frac{n}{2e}$$

as $n \to \infty$. Furthermore, if one of the data points is changed, L changes by at most two. Thus, by McDiarmid's version of Azuma's inequality (McDiarmid, 1990),

$$\mathbf{P}\{L < \mathbf{E}\{L\}/2\} \le e^{-\frac{\mathbf{E}^2\{L\}}{8n}} = e^{-\frac{n}{32e^2+o(1)}} .$$

Each of the subtrees rooted at these nodes is independent of the other ones. Both points in one of these nodes are placed in the same subtree with probability $1/2$. Thus, a subtree has height at least $k-1$ with probability $1/2^{k-1}$. Therefore,

$$\begin{aligned}
\mathbf{P}\{H_n \le k\} &\le \mathbf{E}\left\{(1-1/2^{k-1})^L\right\} \\
&\le \mathbf{E}\left\{e^{-\frac{L}{2^{k-1}}}\right\} \\
&\le e^{-\frac{\mathbf{E}\{L\}}{2^k}} + \mathbf{P}\{L < \mathbf{E}\{L\}/2\} \\
&\le e^{-\frac{n}{(2e+o(1))2^k}} + e^{-\frac{n}{32e^2+o(1)}} \\
&\to 0 ,
\end{aligned}$$

as $n \to \infty$ if $k = \lfloor (1-\epsilon)\log_2 n \rfloor$ for $\epsilon \in (0,1)$.

For the upper bound, let $N_1, \ldots, N_n$ be the cardinalities of the children of the root (so that $\sum_i N_i = n$ unless $n = 1$). If X_i and X_j find themselves in the same child node of the root, then they will stay together at depth 2 with probability $1/2$, at depth 3 with probability $1/2^2$, and at depth k with probability $1/2^{k-1}$. Let $A_{m,k}$ be the event that for the m-th child of the root, one of the pairs of points in the node stays together to depth k. Clearly,

$$\begin{aligned}
\mathbf{P}\{H_n > k\} &\le \mathbf{P}\{\cup_{m=1}^n A_{m,k}\} \\
&\le n\mathbf{P}\{A_{1,k}\} \\
&\le n\mathbf{E}\left\{\frac{\binom{N_1}{2}}{2^{k-1}}\right\} \\
&= \frac{n-1}{2^k}
\end{aligned}$$

as N_1 is binomial $(n, 1/n)$. Therefore, for $\epsilon > 0$,

$$\lim_{n\to\infty} \mathbf{P}\{H_n > (1+\epsilon)\log_2 n\} = 0 . \square$$

4. The height of the random pebbled hash tree.

In this section, we assume that $X_1, \ldots, X_n$ are i.i.d. and have common density f on $[0,1]$. Let H_n be the height of the random pebbled hash tree. Clearly, $H_n \le n-1$. In a random pebbled hash tree we interchangeably speak of nodes, intervals (each node represents an interval), and cardinality (the number of X_i's that fall in a node's interval). We show the following.

THEOREM 3. *Consider a random pebbled hash tree. For any monotonically decreasing sequence $a_n \downarrow 0$ (however slow), there exists a density f for which $\mathbf{P}\{H_n \ge na_n\} \to 1$ as $n \to \infty$.*

Theorem 3 shows that no good universal results are possible for H_n unless the density of the X_i's is suitably restricted. As we may often assume that the hash function is very good, we will assume that the X_i's are uniformly distributed on $[0,1]$. It is worthwhile to note that Theorem 3 remains valid for the N-tree as well.

PROOF. We take a density f that decreases monotonically on $[0,1]$ and has distribution function F. Remove one data point. Let N_1 be the number of points in $[0, 1/n]$. Remove one point again, and let N_2 be the number of points in $[0, 1/n^2]$ and so forth. Assume without loss of generality that na_n is integer-valued and strictly increasing to ∞. Clearly,

$$[N_{na_n} \ge 2] \subseteq [H_n \ge na_n] \ .$$

But N_k is binomial $(N_{k-1}-1, F(1/n^k)/F(1/n^{k-1}))$, which is stochastically greater than a binomial $(N_{k-1}, F(1/n^k)/F(1/n^{k-1}))$ random variable minus one. Therefore, N_k is stochastically greater than a binomial $(n, F(1/n^k))$ random variable minus k. Thus, for n large enough, setting $p = F(1/n^{na_n}) = \sqrt{a_n + 1/n}$, we have, by Chebyshev's inequality,

$$\begin{aligned}
\mathbf{P}\{N_{na_n} < 2\} &\le \mathbf{P}\{B(n,p) \le na_n + 1\} \\
&= \mathbf{P}\{B(n,p) \le np^2\} \\
&\le \frac{np(1-p)}{(np(1-p))^2} \\
&\sim \frac{1}{n\sqrt{a_n + 1/n}} \\
&\to 0 \ .
\end{aligned}$$

Now, take for f a histogram whose distribution function satisfies $F(1/n^{na_n}) = \sqrt{a_n + 1/n}$ for all n large enough. $\square$

Theorem 4. *When f is the uniform distribution on $[0,1]$, we have*

$$\frac{H_n}{\sqrt{\frac{2\log n}{\log\log n}}} \to 1 \text{ in probability} .$$

5. Proof of Theorem 4.

Lemma 1. *For $t > 0$ and $t \ge c$,*

$$\mathbf{P}\{B(n, c/n) \ge t\} \le e^{t-c-t\log t+t\log c} .$$

Proof. We write $B = B(n, c/n)$. By Chernoff's bounding method, for $\lambda > 0$,

$$\begin{aligned}
\mathbf{P}\{B \ge t\} &\le \mathbf{E}\left\{e^{\lambda B-\lambda t}\right\} \\
&= \left(1 + \left(e^{\lambda} - 1\right)\frac{c}{n}\right)^n e^{-\lambda t} \\
&\le e^{c\left(e^{\lambda}-1\right)-\lambda t} .
\end{aligned}$$

The upper bound is minimized when $e^{\lambda} = t/c$. This yields the desired inequality. □

Lemma 2. *If B is a binomial $(n, c/n)$ random variable and $t > 0$, then, for $t \ge c$,*

$$\mathbf{E}\left\{BI_{B\ge t}\right\} \le ce^{t-c-t\log t+t\log c} .$$

PROOF. By simple bounding, we have for $\lambda > 0$,

$$\begin{aligned}\mathbf{E}\left\{BI_{B\geq t}\right\} &\leq \mathbf{E}\left\{Be^{\lambda B-\lambda t}\right\} \\ &= n\mathbf{E}\left\{\frac{c}{n}e^{\lambda B'-\lambda t}\right\} \\ &= c\mathbf{E}\left\{e^{\lambda B'-\lambda t}\right\}\end{aligned}$$

where B' is binomial $(n-1, c/n)$. Here we made use of the linearity of expectation. By the Chernoff bound used in Lemma 1, we have

$$\begin{aligned}\mathbf{E}\left\{BI_{B\geq t}\right\} &\leq c\left(1+\left(e^{\lambda}-1\right)\frac{c}{n}\right)^{n-1} e^{-\lambda t} \\ &\leq c\left(1+\left(e^{\lambda}-1\right)\frac{c}{n-1}\right)^{n-1} e^{-\lambda t} \\ &\leq ce^{c\left(e^{\lambda}-1\right)-\lambda t}\ .\end{aligned}$$

The upper bound is minimized when $e^{\lambda} = t/c$. $\square$

We are now ready to prove Theorem 4. For the upper bound, take $\epsilon > 0$ and define $k = \left\lceil (1+\epsilon)\sqrt{\frac{2\log n}{\log\log n}}\right\rceil$. The tree is pruned by omitting the root node if its cardinality is less than k, all nodes at depth 1 with cardinality less than $k-1$, and in general all nodes at depth d of cardinality less than $k-d$. We call this tree the pruned tree. To compute $\mathbf{P}\{H_n \geq k\}$, the pruned tree and the random pebbled hash tree are equivalent as all deleted nodes are roots of subtrees that cannot reach past depth k. The expected number of nodes in the pruned tree at depth one is

$$(n-1)\mathbf{P}\{B(n-1, 1/(n-1)) \geq k-1\} \leq (n-1)e^{k-1-1-(k-1)\log(k-1)} = \frac{n-1}{e}\ \left(\frac{e}{k-1}\right)^{k-1}\ .$$

where we used Lemma 1. Let the L nodes at depth one have cardinalities $N_1, \ldots, N_L$. Given N_1, the first node spawns an expected number of nodes at depth two equal to

$$(N_1-1)\mathbf{P}\{B(N_1-1, 1/(N_1-1)) \geq k-2\} \leq (N_1-1)e^{k-2-1-(k-2)\log(k-2)} = \frac{N_1-1}{e}\ \left(\frac{e}{k-2}\right)^{k-2}\ .$$

Given all the cardinalities, we thus have an expected number of nodes at depth 2 not exceeding

$$\frac{\sum_{i=1}^{L}(N_i-1)}{e}\ \left(\frac{e}{k-2}\right)^{k-2}\ .$$

But

$$\mathbf{E}\left\{\sum_{i=1}^{L} N_i\right\} = \mathbf{E}\left\{\sum_{i=1}^{n-1} M_i I_{M_i\geq k-1}\right\} \leq ne^{k-1-1-(k-1)\log(k-1)}\ ,$$

where $M_1, \ldots, M_{n-1}$ are the cardinalities of the $n-1$ intervals in the first partition. Here we used Lemma 2 with $c=1$. Therefore, the expected number of nodes at depth two does not exceed

$$\frac{n}{e^2}\left(\frac{e}{k-1}\right)^{k-1}\left(\frac{e}{k-2}\right)^{k-2} .$$

By induction, the expected number of nodes at depth $k-1$ does not exceed

$$\frac{n}{e^{k-1}}\prod_{i=1}^{k-1}\left(\frac{e}{i}\right)^i = ne^{(k-1)(k-2)/2-\sum_{i=1}^{k-1} i\log i} = ne^{-(1/2+o(1))k^2\log k} .$$

Thus,

$$\mathbf{P}\{H_n \geq k\} \leq ne^{-(1/2+o(1))k^2\log k} \to 0$$

for the given choice of k.

For a matching lower bound, we argue by embedding and prune the tree even further. For $\epsilon \in (0,1)$, define $k = \left\lfloor (1-\epsilon)\sqrt{\frac{2\log n}{\log\log n}}\right\rfloor$. Of all nodes at depth one, we keep only those of exact cardinality k. These nodes spawn children at depth two, of which we keep only the first child and only if it is of cardinality precisely $k-1$. Continuing in this manner, the process either becomes extinct, or it survives up to depth k with at least one node of cardinality 1. In the latter case, $H_n \geq k$. A node at depth one has progeny that survives to depth k with probability

$$\frac{1}{(k-1)^{k-1}(k-2)^{k-2}\cdots 2^2 1^1} \stackrel{\text{def}}{=} q .$$

Clearly, $q = e^{-(1/2+o(1))k^2\log k}$. Given that there are L nodes at depth one in the pebbled hash tree, we have

$$\mathbf{P}\{H_n < k | L\} \leq (1-q)^L .$$

Now, L is binomial $(n-1,p)$, where $p = \mathbf{P}\{B = k\}$ and B is binomial $(n-1, 1/(n-1))$. It is easy to verify that for $n \geq 3$,

$$\begin{aligned} p &= \binom{n-1}{k}\frac{1}{(n-1)^k}\left(1-\frac{1}{(n-1)}\right)^{n-1-k} \geq \frac{1}{k!}\left(\frac{n-1-k}{n-1}\right)^k\left(1-\frac{1}{n-1}\right)^{n-1} \\ &\geq \frac{1}{4\ k!}\left(1-\frac{k}{n-1}\right)^k \geq \frac{1-k^2/(n-1)}{4\ k!} \stackrel{\text{def}}{=} q' . \end{aligned}$$

Hence L is stochastically greater than B', a binomial $(n-1,q')$ random variable. Thus,

$$\begin{aligned} \mathbf{P}\{H_n < k\} &\leq \mathbf{E}\left\{(1-q)^L\right\} \leq \mathbf{E}\left\{(1-q)^{B'}\right\} \\ &= (q'(1-q)+1-q')^{n-1} = (1-qq')^{n-1} \leq e^{-(n-1)qq'} \to 0 \end{aligned}$$

when $nqq' \to \infty$. This is easily verified for our choice of k. $\square$

6. The size of random hash trees.

The second parameter of primary interest is S_n. We will only look at $s_n = \mathbf{E}S_n$. By linearity of expectation, we have

$$s_n = \begin{cases} 1\ , & 0 \le n \le 1 \\ 1 + n\sum_{i=0}^{n} \mathbf{P}\{B(n,1/n) = i\} s_i\ , & n \ge 2. \end{cases}$$

Note that we provide storage for empty bins as well. The recurrence given above yields $s_0 = s_1 = 1$, $s_2 = 1 + 2(3/4 + s_2/4)$, so that $s_2 = 5$. We have

Theorem 5. *For the random hash tree,*

$$\lim_{n\to\infty} \frac{\mathbf{E}S_n}{n} = 2.3020238\ldots\ .$$

The limiting constant is

$$\frac{1}{e}\sum_{i=0}^{\infty} \frac{s_i}{i!}\ ,$$

where $s_0, s_1, \ldots$ is given by the recurrence above.

Proof. The values s_n can be shown to be approximable by the values t_n, where

$$t_n = \frac{n}{e}\sum_{i=0}^{\infty} \frac{s_i}{i!}\ .$$

Indeed, note that

$$\frac{s_n - t_n}{n} = \frac{1}{n} + \sum_{i=0}^{n} \left(\mathbf{P}\{B(n,1/n) = i\} - \frac{1}{e\ i!} \right) s_i - \sum_{i>n} \frac{s_i}{e\ i!}\ .$$

But for $0 \le i \le n$,

$$\begin{aligned} \mathbf{P}\{B(n,1/n) = i\} - \frac{1}{e\ i!} &= \binom{n}{i} \frac{(n-1)^{n-i}}{n^n} - \frac{1}{e\ i!} \le \frac{n^i}{i!} \frac{(n-1)^{n-i}}{n^n} - \frac{1}{e\ i!} \\ &= \frac{1}{i!} \frac{(n-1)^{n-i}}{n^{n-i}} - \frac{1}{e\ i!} \le \frac{e^{\frac{i}{n}-1}}{i!} - \frac{1}{e\ i!} \le \frac{i}{n\ i!} \end{aligned}$$

and for $0 \le i \le n$,

$$\begin{aligned} \mathbf{P}\{B(n,1/n) = i\} - \frac{1}{e\ i!} &\ge \frac{(n-i+1)^i}{i!} \frac{(n-1)^{n-i}}{n^n} - \frac{1}{e\ i!} \ge \left(1 - \frac{i-2}{n-1}\right)^i \frac{e^{-\frac{1}{1-1/n}}}{i!} - \frac{1}{e\ i!} \\ &\ge \frac{e^{-\frac{i^2}{n-i+1} - \frac{1}{1-1/n}}}{i!} - \frac{1}{e\ i!} \ge \frac{1}{e\ i!} \left(e^{-\frac{i^2}{n-i+1} - \frac{1}{n-1}} - 1 \right) \end{aligned}$$

$$\geq -\frac{1}{e\ i!}\left(\frac{i^2}{n-i+1}+\frac{1}{n-1}\right) \geq -\frac{1}{e\ i!}\left(\frac{i^2}{n/2}+i^2 I_{i\geq n/2}+\frac{1}{n-1}\right) .$$

Thus, we have $(s_n - t_n)/n \to 0$ if

$$\sum_{i=0}^{\infty} \frac{i^2 s_i}{i!} < \infty .$$

But that follows from the simple fact that $s_i = O(i)$, something that is easy to verify. Numerical computations show that

$$\frac{1}{e}\sum_{i=0}^{\infty} \frac{s_i}{i!} = 2.3020238\ldots . \square$$

For the random pebbled hash tree, the recurrences are slightly different. Indeed,

$$s_n = \begin{cases} 1, & 0 \leq n \leq 1 \\ 1+(n-1)\sum_{i=0}^{n-1} \mathbf{P}\{B(n-1,1/(n-1))=i\}s_i , & n>1. \end{cases}$$

The analysis is entirely similar as for Theorem 5, and we thus obtain

THEOREM 6. *For the random pebbled hash tree,*

$$\lim_{n\to\infty} \frac{\mathbf{E}S_n}{n} = 1.4183342\ldots .$$

The limiting constant is

$$\frac{1}{e}\sum_{i=0}^{\infty} \frac{s_i}{i!} ,$$

where $s_0, s_1, \ldots$ is given by the recurrence above.

In particular, note that random pebbled hash trees are smaller on the average than random N-trees.

7. BBST: bucketing followed by binary search trees.

Assume that we bucket n points into n equi-spaced buckets, and that within each bucket, we maintain a balanced binary search tree. Then, in view of the results of Gonnet (1981) and Devroye (1985), the expected worst-case search and insert times and indeed the expected value of the height of this hybrid structure is asymptotic to $\log_2 \log n$ for all bounded densities f on $[0,1]$. In fact, the height H_n satisfies $H_n/\log_2 \log n \to 1$ in probability. The expected average search time (if each element is equally likely to be probed for) and the expected unsuccessful search time (for a random element drawn independently from the same density f) are both $O(1)$.

8. Extension: m pebbles.

We may extend the analysis to m pebbles, leaving m randomly selected points in every node before bucketing. If there are $m+k$ points, then there are k (with possibly $k=1$) child nodes, each corresponding to a subinterval $1/k$-th of the length of the interval of the split node. This leads to random m-pebbled hash trees. A quick analysis not worth repeating here shows that in this case, $H_n \sim \sqrt{(2/m)\log n/\log\log n}$ in probability. However, the worst-case search time is roughly $(m+1)H_n$ when comparisons and bucket operations all take one time unit. Therefore, it seems wasteful to take $m>1$, and thus, the most important member of the family is the pebbled hash tree studied above. A rather obvious but unaesthetic modification however will improve matters exponentially. If we set $m = c\log n/\log\log n$ for some constant c, and pick the m pebbles as follows for a node representing interval $[a,b]$: find the m-th order statistic M in time linear in the number of points. The m points on $[a,M]$ are placed in a balanced binary search tree in time $m\log m = O(\log\log n)$. The remaining points on $(M,b]$ are bucketed as in a random hash tree, and the process is repeated. This tree has expected height $O(1)$ so that expected worst-case search times are $O(\log\log n)$. As this method is eclipsed by the simple BBST structure of the previous section, we will not analyze it in this paper.

9. Dynamic data structures.

The data structures described above are of course useful in any static setting, in which case we have expected preprocessing time $O(n)$ and expected worst-case search times as given by the expected values of H_n in the analyses. Consider now the standard dictionary operations `insert` and `search`. We may introduce the load factor α, the number of elements stored divided by the branch factor of the root. The objective is to keep α at all times in a fixed range, such as $[1/2,2]$. As soon as α reaches a boundary, a complete rehash is performed to make $\alpha=1$. In an amortized sense, these rehash operations take $O(1)$ expected time. Expected worst-case search times remain asymptotically the same as for the static case. However, for a fully dynamic data structure with interspersed `delete` and `insert` operations, additional analysis is required.

10. Acknowledgment.

I would like to thank all referees for their suggestions.

11. Dedication.

This paper is dedicated to the memory of Markku Tamminen who died tragically in New York shortly after he finished his analysis of N-trees and random hash trees.

12. References.

Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced allocations (extended abstract)," in: *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pp. 593–602, 1994.

E. G. Coffman and J. Eve, "File structures using hashing functions," *Communications of the ACM*, vol. 13, pp. 427–436, 1970.

L. Devroye, "The expected length of the longest probe sequence when the distribution is not uniform," *Journal of Algorithms*, vol. 6, pp. 1–9, 1985.

L. Devroye, *Lecture Notes on Bucket Algorithms*, Birkhäuser Verlag, Boston, 1986.

W. Dobosiewicz, "Sorting by distributive partitioning," *Information Processing Letters*, vol. 7, pp. 1–6, 1978.

G. Ehrlich, "Searching and sorting real numbers," *Journal of Algorithms*, vol. 2, pp. 1–14, 1981.

R. Fagin, J. Nievergelt, N. Pippenger, and H. R. Strong, "Extendible hashing—a fast access method for dynamic files," *ACM Transactions on Database Systems*, vol. 4, pp. 315–344, 1979.

P. Flajolet, "On the performance evaluation of extendible hashing and trie search," *Acta Informatica*, vol. 20, pp. 345–369, 1983.

M. L. Fredman and D. E. Willard, "Blasting through the information theoretic barrier with fusion trees," in: *Proceedings of the Twenty Second Annual Symposium on Theory of Computing*, pp. 1–7, ACM Press, 1990.

G. H. Gonnet, "Expected length of the longest probe sequence in hash code searching," *Journal of the ACM*, vol. 28, pp. 289–304, 1981.

G. H. Gonnet and R. Baeza-Yates, *Handbook of Algorithms and Data Structures*, Addison-Wesley, Workingham, 1991.

C. McDiarmid, "On the method of bounded differences," in: *Surveys in Combinatorics 1989*, vol. 141, pp. 148–188, London Mathematical Society Lecture Notes Series, Cambridge University Press, Cambridge, 1989.

B. Pittel, "Asymptotical growth of a class of random trees," *Annals of Probability*, vol. 13, pp. 414–427, 1985.

M. Tamminen, "Analysis of N-trees," *Information Processing Letters*, vol. 16, pp. 131–137, 1983.

M. Tamminen, "Two levels are as good as any," *Journal of Algorithms*, vol. 6, pp. 138–144, 1985.

J. S. Vitter and P. Flajolet, "Average-case analysis of algorithms and data structures," in: *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, ed. J. van Leeuwen, pp. 431–524, MIT Press, Amsterdam, 1990.