RANDOM INSERTION INTO A PRIORITY QUEUE STRUCTURE

by

Thomas Porter
Istvan Simon

STAN-C S-74-460
OCTOBER 1974

COMPUTER SC IENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UN IVERS ITY

Random Insertion into a Priority Queue Structure

by

Thomas Porter
Istvan Simon*/

Abstract

The average number of levels that a new element moves up when
inserted into a heap is investigated. Two probabilistic models, under
which such an average might be computed are proposed. A "lemma of
conservation of ignorance" is formulated and used in the derivation of
an exact formula for the average in one of these models. It is shown
that this average is bounded by a constant and its asymptotic behavior
is discussed. Numerical data for the second model is also provided and
analyzed.

Keywords and phrases: Priority queue, heap insertion, heap sort,
analysis of algorithms.

CR Categories: 5.25, 5.31

Random Insertion into a Priority Queue Structure

1.   Introduction

In this paper we investigate the average number of levels that a
new element moves up when inserted into an (n-1) -heap to form an
n-heap. An n-heap [Williams - 1964, Knuth - 1973] is a complete
binary tree of n nodes such that the key associated with each node
is larger than the keys of both of its sons. Given an (n-1) -heap,
a new node can be inserted by placing it initially at the bottom of
the tree, thereby creating a complete binary tree of n nodes, and
then repeatedly comparing the key of the inserted node, x , with the
key of its father, y , exchanging the two nodes if x > y . If at
any stage x < y the resulting binary tree is an n-heap. Since a
complete binary tree of n nodes has $\lfloor \lg(n) \rfloor + 1$ levels, $^{*/}$ the
inserted node moves up at most $\lfloor \lg(n) \rfloor$ levels. Hence we can create
an n-heap by repeated application of this process in less than $n \lfloor \lg n \rfloor$
operations. This suggests that the heap insertion method just described
could be used in the heap creation phase of heapsort [Knuth - 1973,
Section 5.2.3, Algorithm H]. One might expect that the average behavior
of the heap insertion method is still much better. Actually Williams'
INHEAP routine in his original paper is essentially the insertion method
just described, and he states without proof in the comment accompanying
his routine that the average number of exchanges is two. In one of the
models proposed in this paper we shall prove that the average is bounded
by a constant less than two.

$^{*/}$ $\lg n$ denotes $\log_2 n$.

To avoid ambiguities we state now the precise description of the insertion algorithm. In this description we make use of the well known compact representation of a complete binary tree in an array k , where k[1] is the root, and k[j] has left son k[2j] and right son k[2j+1] . We also assume that each node consists only of its key. If there are other fields of the node besides the key, the corresponding modification of Algorithm I is trivial, and it obviously has no effect on the average we are investigating here.

Algorithm I. This algorithm inserts the n-th node into a heap. The heap is stored in k[1],k[2],...,k[n-1] and k[n] is the node to be inserted.

I1. [Initialize.] Set p ← n , q ← ⌊n/2⌋ , k ← k[n] .

I2. [Sift it up.] While q > 0 and k[q] < k do
                begin k[p] ← k[q], p ← q , q ← ⌊p/2⌋ ___

I3. [Insert.] Set k[p] ← k .   □

2.   The Models

Let H(n) denote the number of n-heaps with n given distinct keys, k_1, k_2, ● ... . The following closed form is known [Knuth - 1973, Section 5.2.3] for H(n) :

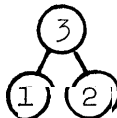$$H(n) = \frac{n!}{\prod_{1 \le i \le n} s_i} , \tag{1}$$

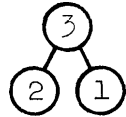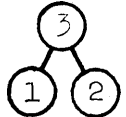where s_i is the size of the sub-tree rooted at node k[i] .

<u>Definition</u>:   Let A(n) denote the average number of levels the n-th
node is moved up by Algorithm I.
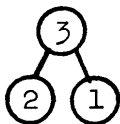
To find A(n) we consider two models:

<u>Model 1</u>:   We assume that each of the $H(n-1)$ possible heaps with the
(n-1) elements already in the heap $[k_1, k_2, \ldots, k_{n-1}]$ is equally likely,
and that the key of the n-th node k_n is equally likely to occur in any
of the n intervals determined by $k_1, k_2, \ldots, k_{n\ 1}$.   In Section 4 we
shall derive a simple recursive formula for $A(n)$ in this case.
Furthermore, we shall prove several properties of this formula in
Section 5.

The assumption that each of the H(n-1) possible heaps is equally
likely is justified if one uses the heap creation algorithm of heapsort
[Knuth -1973, Section 5.2.3, Algorithm, H, heap creation phase] to build
the (n-1) -heap and then applies Algorithm I to insert the last element.
It is shown [Knuth -1973, Section 5.2.3, Theorem H] that each (n-1) -heap
occurs with equal probability in this case.   One could hope that a
similar theorem would hold for heap creation by repeated application of
Algorithm I.   Unfortunately, this is not so, for certain heaps will
occur more-often than others if-we apply Algorithm I successively to a
random permutation of 1,2,...,n .   To see this, let us compute the
probability distribution for the two possible heaps when n = 3 . It
is easily verified that the permutations   1 2 3 ,   1 3 2 ,   2 1 3

and   3 1 2   are transformed into the heap   , while   2 3 1

and   3 2 1   are transformed into   . Hence   is twice

4

as likely as   (tree: 3 at top, children 2 and 1)   if we assume that each permutation is equally
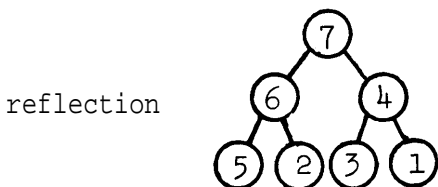
likely.   An even more striking example of this fact is that the heap
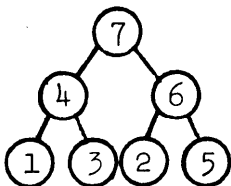
(tree: 7 at top; children 4 and 6; 4's children 1 and 3; 6's children 2 and 5)   is generated by 228 **7-permutations** while its

reflection   (tree: 7 at top; children 6 and 4; 6's children 5 and 2; 4's children 3 and 1)   is generated **by** only 12 , a ratio of 19:1 !

Therefore we are led to our second model in a natural way.

Model 2: We assume that the H(n-1) heaps occur with a probability
distribution induced when Algorithm I is applied successively n-1
times to a random permutation of $1,2,\ldots,n$. Then we determine the
average A(n) when applying Algorithm I once more to insert K_n .

      In Section 6 we shall discuss the average according to Model 2.
Although we do not provide an exact formula for Model 2, we do present
numerical data that shows results relatively close to those of Model 1.

3.   Combinatorial Preliminaries

      In this section we present a **combinatorial** lemma about k-arrangements
which appears to be useful in a variety of situations.   In particular
we shall use it to prove Theorem 1 in the next section.   The result
was suggested to us by D. E. **Knuth** and has been previously used in the
solution of several problems, but to our knowledge has not been precisely
formulated before.

Throughout this section, n and k denote fixed positive
integers with $n \geq k$.

Definition:

(1) A k-arrangement of n objects is an ordered k-tuple of these
objects. A k-arrangement of k objects is called a k-permutation.
We shall consider only k-arrangements of $\{1,2,\ldots,n\}$.

(2) Let σ be the set of k-arrangements of $\{1,2,\ldots,n\}$ and let π
be the set of k-permutations of $\{1,2,\ldots,k\}$. The function
$f\colon \sigma \to \pi$ such that $f((a_1,\ldots,a_k)) = (b_1,\ldots,b_k)$ implies

$$a_i < a_j \iff b_i < b_j$$

is called the <u>renumbering function</u> preserving relative order. It
is obvious that for each n and k the renumbering function f
preserving relative order is unique.

3) A property P of k-arrangements is said to depend only on the
relative order of its elements when $P(a) \iff P(f(a))$ for all
$a \in \sigma$. (Note that since $\pi \subseteq \sigma$, $f(a) \in \sigma$ and hence we can talk
about P applied to $f(a)$.)

Examples:

(1) The property that the first element of the k-arrangement is the
largest one is a property that depends only on the relative order
of its elements.

(2) The property that the first k-1 elements of the k-arrangement
form a (k-1) -heap is also such a property.

Definition: A random variable over a certain space with uniform
probability distribution will be called simply a random variable over
that space.

We observe that the renumbering function f induces a partition
over the set σ , where two k-arrangements belong to the same equivalence
class, if and only if they are both mapped into the same k-permutation
by f .

Lemma 1:$^*/$ Let P be a property of k-arrangements that depends only
on the relative order of its elements. A random k-arrangement of
$\{1,2,\ldots,n\}$ satisfying P remains random under the renumbering
function f preserving relative order.

Proof: All we have to prove is that an equal number of k-arrangements
that satisfy P are mapped by f into each k-permutation that
satisfies P . Notice that since P depends only on the relative
order of the elements of the k-arrangements, those that satisfy P
are always mapped into k-permutations that satisfy P . Furthermore,
if any k-arrangement of an equivalence class satisfies P , then all
k-arrangements of that class satisfy P . Hence we may simply show
that each equivalence class has the same number of elements. But there
are exactly k! equivalence classes each corresponding to a k-permutation.
Now consider any k-subset of $\{1,2,\ldots,n\}$. Permuting its elements in
every possible order, it follows immediately from the definition of f
that exactly one of these k! permutations falls into each equivalence

---

$^*/$ L. Guibas has suggested that this lemma be named a "Principle of
Conservation of Ignorance", because the randomness is preserved
through the renumbering process.

class. Thus, since there are $\binom{n}{k}$ k-subsets of $\{1,2,\ldots,n\}$, there
are exactly $\binom{n}{k}$ elements in each equivalence class. $\square$

4.   The Analysis by Model 1

In this section we derive a formula for A(n) in Model 1. The
input to Algorithm I can be thought of as a complete binary tree of n
nodes, $k_1, k_2, \ldots, k_n$, such that

(i) The nodes $k_1, k_2, \ldots, k_{n-1}$ form an (n-1) -heap.

(2)

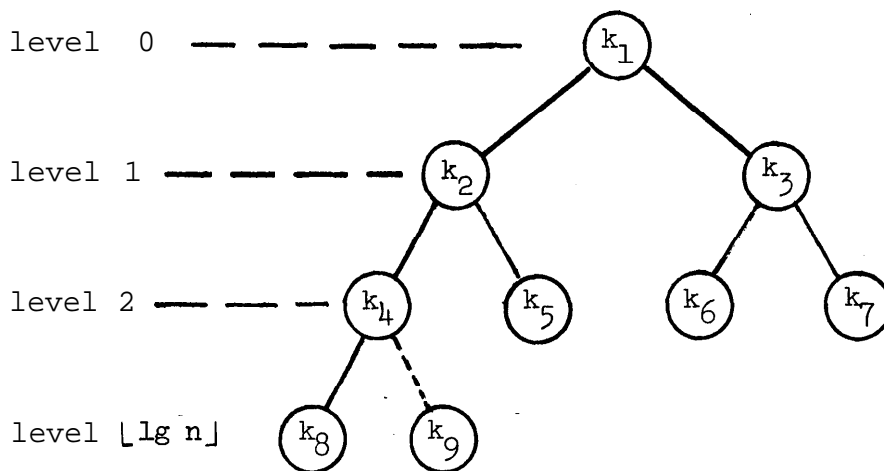(ii) The n-th node, k_n , of the binary tree is the new node

   to be inserted.

Figure 1.   Input to Algorithm I when n = 9 .

The situation is depicted in Figure 1 when n = 9 . The nodes connected
with solid lines form the (n-1) -heap, and the broken edge connecting

8

k_9 to the tree is used to indicate that k_9 is the node to be inserted into the heap. Thus, at this point k_9 is the only node that might violate the heap condition key(son) < key(father) . It is important to notice that as long as the relative order of $k_1, k_2, \ldots, k_9$ is preserved, the values of the keys themselves are irrelevant as far as the complexity of Algorithm I is concerned for this input.In other words, Algorithm I will execute exactly the same sequence of instructions for two inputs $k_1, k_2, \ldots, k_n$ and $k'_1, k'_2, \ldots, k'_n$ satisfying condition (2), , provided that their relative order is the same. Therefore we may as well assume that

$$\{k_1, k_2, \ldots, k_n\} = \{1, 2, \ldots, n\} = M_n \quad . \tag{3}$$

We now prove the following formula for the average A(n) in Model 1:

Theorem 1:   The average A(n) under the assumptions of Model 1 satisfies the recurrence relation

$$A(n) = \frac{1}{n} \cdot L + \frac{n-1}{n} \cdot A(n - 2^{\ell}) \qquad \text{for } n > 2$$

where $L = \lfloor \lg n \rfloor$ and $\ell = \lfloor \lg(\frac{2}{3}n) \rfloor$ and

$A(1) = 0$.

Proof: Let $L = \lfloor \lg n \rfloor$ be the level of the inserted node, and let T denote the subtree of the root that contains k_n . Let T' be the other subtree and let $\hat{T}$ be the subtree T without the node k_n .   Thus $\hat{T}$ and T' are both heaps. Each possible value of k_n occurs with probability 1/n . If $k_n = n$, Algorithm I will move it up exactly L levels, since in this case the inserted node has maximum key, and therefore it will be at the root when the algorithm terminates. If $k_n \neq n$, then n is already at the root and k_n will eventually

9

settle at some level in T .   Therefore we may write

$$A(n) \quad = \quad \frac{1}{n} \cdot L + \frac{n-1}{n} \; A[T] \tag{4}$$

where A[T] denotes the average number of levels k_n is moved up in

the **subtree** T .   The **subtree** T has exactly n-2' nodes, where

$a = \lfloor \lg(\frac{2}{3} n) \rfloor$. (Note that ℓ = L-1 if T is the left **subtree** of

the root and ℓ = L if it is the right one.)

   Our aim now is to show that

$$A[T] \quad = \quad A(n - 2^{\ell}) \quad . \tag{5}$$

Fixing a particular T and varying the possible heap arrangements of

the nodes of T' we see that each T occurs exactly $H(2^{\ell}-1)$ times

among the original (n-1) -heaps, which are assumed equally likely by

hypothesis.   Therefore each possible T is equally likely and we can

compute A[T] over the space of possible T-s . Keeping also in mind

that k_n will eventually settle at some level in T , since $k_n \in M_{n-1}$

and n is already at the root, this means that everything works as if

we were inserting k_n into $\hat{T}$ rather than into the original (n-1) -heap.

Let us therefore assume from now on that our input is T . The input T

has $k = n - 2^{\ell}$ nodes chosen from $M_{n\;1}$, and it can be regarded as a

k-arrangement of $\{1,2,\ldots,n-1\}$ such that its first (k-1) elements

form a heap.   As we have seen before, this is a property of k-arrangements

that depends only on the relative order of its elements, and hence a

-random T remains random under the renumbering function f preserving

relative order, by Lemma 1.   Furthermore by observation (3) above, the

renumbering process preserving relative order does not change the number

of levels that the inserted node is moved up by Algorithm I. Hence we

can compute the average $A[T]$ over the space of the renumbered **trees.**
But this is precisely $A(n - 2^\ell)$.

<div align="right">☐</div>

5. Some Properties of the Average A(n)

In this section we shall explore some of the properties of the average $A(n)$ derived in Section 4. In particular we shall show that at any level L of the tree the leftmost node, 2^L , has the largest average. We then proceed to prove-that $A(2^L)$ is always bounded by a constant and approaches this constant as L approaches infinity, thus showing that $A(n)$ is bounded by a constant for all n . We then derive a closed formula for n of the form $2^{L+1} - 1$, which is the rightmost node at level L , and show that actually in this case $A(n)$ is bounded by 1 and approaches 1 as L approaches infinity. Finally we examine the asymptotic behavior of $A(n)$, where n varies along an arbitrary path of the tree.

<u>Theorem 2</u>: If n_1 and n_2 are two nodes at the same level L such that n_1 is to the left of n_2 (i.e., $n_1 < n_2$), and if $A(n_1 - 2^{\ell_1}) \geq A(n_2 - 2^{\ell_2})$ where $\ell_i = \lfloor \lg(\frac{2}{3} n_i) \rfloor$, then $A(n_1) > A(n_2)$.

<u>Proof</u>: By Theorem 1,

$$A(n_i) = \frac{L}{n._i} + (1 - \frac{1}{n_i}) A(n_i - 2^{\ell_i}) \quad .$$

Since $A(n_1 - 2^{\ell_1}) \geq A(n_2 - 2^{\ell_2})$,

<div align="center">11</div>

$$A(n_1) - A(n_2) \geq \left(\frac{1}{n_1} - \frac{1}{n_2}\right)L - \left(\frac{1}{n_1} - \frac{1}{n_2}\right)A(n_2 - 2^{\ell_2})$$

$$= (L - A(n_2 - 2^{\ell_2}))\left(\frac{1}{n_1} - \frac{1}{n_2}\right) .$$

Now $L - A(n_2 - 2^{\ell_2}) > 0$, because $(n_2 - 2^{\ell_2})$ is a node at level L-1 , and it can move up at most L-1 levels. It follows that $A(n_1) > A(n_2)$. $\qquad\square$

Corollary 1: At each level L the leftmost node 2^L has the largest average value.

Proof: The proof is by induction. At level 0 the result is trivial. Now assume the corollary at level L-1 . Let n_1 be the leftmost node at level L , and let n_2 be a node to its right at the same level. Then $n_2 - 2^{\ell_2}$ is some node at level L-1 , and $n_1 - 2^{\ell_1}$ is the leftmost node at that level (since $\ell_1 = \lfloor \lg(\frac{2}{3} n_1) \rfloor = \lfloor \lg(\frac{2}{3} 2^L) \rfloor = L-1$, and $n_1 - 2^{\ell_1} = 2^{L-1}$). Therefore $A(n_1 - 2^{\ell_1}) > A(n_2 - 2^{\ell_2})$ by the induction hypothesis, and the corollary follows from Theorem 2. $\qquad\square$

We now examine the average at the leftmost nodes at each level. Since $A(n) = A(n - 2^{\ell}) + \frac{L - A(n - 2^{\ell})}{n}$ it follows that $A(n) > A(n - 2^{\ell})$ for all n . In particular if n is the leftmost node, i.e., $n = 2^L$, then $n - 2^{\ell} = 2^{L-1}$, hence $A(2^L)$ is a monotonically increasing sequence with L . It is not difficult to show that this sequence has a limit λ , and hence it is bounded by this limit. By virtue of Corollary 1 this

implies that $A(n) < \lambda$ for all n , i.e., λ is a constant that

bounds $A(n)$. Furthermore λ is the best conceivable such bound

since $\lim\limits_{L \to \infty} A(2^L) = \lambda$.

We shall now determine a closed formula for $A(2^L)$ and use it to

derive an expression convenient for the numerical computation of λ .

Theorem 3: The bound λ satisfies the equalities

$$\lambda = \lim_{L \to \infty} A(2^L) = \sum_{j \geq 1} \frac{1}{2^j - 1} = 1.6066951524 \ldots$$

Proof: We have $A(2^L) = \dfrac{L}{2^L} + (1 - 2^{-L})A(2^{L-1})$ by Theorem 1. Let a_L

denote $A(2^L)$ and let $b_L = \dfrac{a_L}{\prod\limits_{1 \leq j \leq L} (1 - 2^{-j})}$ Then

$a_L = \dfrac{L}{2^L} + (1 - 2^{-L})a_{L-1}$ and

$$\frac{a_L}{\prod\limits_{1 \leq j \leq L} (1 - 2^{-j})} = \frac{L}{2^L \prod\limits_{1 \leq j \leq L} (1 - 2^{-j})} + \frac{a_{L-1}}{\prod\limits_{1 \leq j \leq L-1} (1 - 2^{-j})} ,$$

i.e.,

$$b_L = \frac{L}{2^L \prod\limits_{1 \leq j \leq L} (1 - 2^{-j})} + b_{\ldots} \qquad (6)$$

Iterating equation (6) we get

$$b_L = b_0 + \sum_{1 \leq i \leq L} \frac{i}{2^i \prod\limits_{1 \leq j \leq i} (1 - 2^{-j})} . \qquad (7)$$

13

But $b_0 = a_0 = 0$, hence (7) yields

$$a_L = \sum_{1 \le i \le L} \frac{i \prod_{1 \le j \le L} (1-2^{-j})}{2^i \prod_{1 \le j \le i} (1-2^{-j})} \qquad (8)$$

Let $P = \prod_{j \ge 1} (1-2^{-j})$ and $P_i = \prod_{1 \le j \le i} (1-2^{-j})$. If L approaches

infinity, a_L approaches λ , hence

$$\lambda = P \sum_{i \ge 1} \frac{i}{2^i P_i} \qquad (9)$$

By Euler's partition formula [Knuth - 1973, exercise 5.1.1-16],

$$\prod_{j \ge 0} \frac{1}{(1-q^j z)} = \sum_{i \ge 0} \frac{z^i}{\prod_{1 \le j \le i} (1-q^j)} \qquad \text{Setting } z = \tfrac{1}{2} x \text{ and}$$

$q = \tfrac{1}{2} = 2^{-1}$ yields

$$R(x) = \prod_{j \ge 1} \frac{1}{1-2^{-j}x} = 1 + \sum_{i > 1} \frac{x^i}{2^i P_i} \qquad (10)$$

Let $r_j(x) = \dfrac{1}{1-2^{-j}x}$. Then $r'_j(x) = \dfrac{d}{dx} r_j(x) = \dfrac{2^{-j}}{(1-2^{-j}x)^2}$.

Taking derivatives of (10) and noting that $R'(x) = \dfrac{d}{dx} \left(\prod_{j \ge 1} r_j(x) \right) =$

$R(x) \sum_{j \ge 1} \dfrac{r'_j(x)}{r_j(x)}$ we have $R'(x) = R(x) \cdot \sum_{j \ge 1} \dfrac{1}{2^j (1-2^{-j}x)} = \sum_{i \ge 1} \dfrac{i x^{i-1}}{2^i P_i}$.

In particular for $x = 1$, noting that $R(1) = \dfrac{1}{P}$, we have

$$\frac{1}{P} \sum_{j \geq 1} \frac{1}{2^j-1} = \sum_{i \geq 1} \frac{i}{2^i P_i} \quad \text{Finally, by (9),}$$

$$\lambda = \sum_{j \geq 1} \frac{1}{2^j-1} \quad . \tag{11}$$

$\square$

The sum (11) is rapidly converging and can be used to find λ numerically.

We can also derive a closed formula for the average at the rightmost node of each level.

Theorem 4: If n is the rightmost node at level L , (i.e., $n = 2^{i+1}-1$), then $A(n) = 1 - \frac{L+1}{n}$.

Proof: A simple induction on L . $\square$

The proof of the following corollary is now trivial:

Corollary 2: If n is of the form $2^{L+1}-1$ then

(i) $A(n) < 1$

(ii) $\lim_{L \to \infty} A(n) = 1$.

Theorems 3 and 4 give asymptotic values for the average A(n) along two particular paths down the tree. We can describe any path -on the tree by a binary sequence α where the L-th element of the sequence, α_L , is 0 if we go from level L-1 to L taking the left branch and 1 if the right branch is taken. With this convention the number 1 followed by the first L bits of α , in binary, gives

15

the node at level L along that path. We denote this node by $2^L \cdot \alpha$. */

Hence the limit $A(a) = \lim_{L \to \infty} A(2^L \cdot \alpha)$, if it exists, gives the

asymptotic value of the average along the path defined by α . If

the limit does not exist we say that A(a) is undefined. For example

$a^{(0)} = 000 \ldots$ refers to the path of leftmost nodes at each level,

and thus $A(\alpha^{(0)}) = \lambda$ according to Theorem 3, while $\alpha^{(1)} = 111 \ldots$

refers to the path of rightmost nodes at each level, and thus

A(&)) = 1 by Corollary 2.

Definition:

(1) Two binary sequences α and β have the same tail if there exists

   an i and j such that $\alpha_{i+k} = \beta_{j+k}$ for all $k \geq 0$.

(2) If $m = n-2^\ell$, where $\ell = \lfloor \lg(\frac{2}{3} n) \rfloor$ then we say that the average

   at node n depends on the average at node m .

   This definition is motivated by Theorem 1.

Theorem 5:   If α and β are two binary sequences that have the

same tail, then $A(\alpha)$ and $A(\beta)$ are either both undefined or both

defined and equal.

---

*/ This notation is motivated by regarding α as associated with the

   real number $\hat{\alpha} = 1 + \sum_{L \geq 1} \frac{\alpha_L}{2^L}$. Then the node denoted by $2^L \bullet a$ is

   clearly the node $\lfloor 2^L \cdot \hat{\alpha} \rfloor$. Note however that this correspondence
   between α and $\hat{\alpha}$ is not 1-1 , for sequences that are of the form
   $\alpha_1 \alpha_2 \ldots \alpha_k 1 0 0 0 \ldots$ and $\alpha_1 \alpha_2 \ldots \alpha_k 0 1 1 1 \ldots$ correspond to the
   same real while defining distinct paths on the tree.

16

<u>Proof:</u>   Let n_0 be a node at level i and let n_1 be the node

at level i-1 such that the average at n_0 depends on the average

at $n_1 = n_0 - 2^{\ell_0}$. It is an immediate consequence of Theorem 1 that

the average at node $2n_0$ (node $2n_0+1$), the left (right) son of n_0

depends on the average at node $2n_1$ (node $2n_1+1$) the left (right)

son of n_1 . Now given $\delta^{(0)}$ such that $2^i \delta^{(0)} = n_0$ let $\delta^{(1)}$ be

the binary sequence **satisfying** $2^{i-1} \cdot \delta^{(1)} = n_1$ and $\delta^{(0)}_{i+k} = \delta^{(1)}_{i-1+k}$ for all

$k \geq 0$. Then $A(2^{i+k} \cdot \delta^{(0)}) = \dfrac{i+k}{2^{i+k} \cdot \delta^{(0)}} + \left(1 - \dfrac{1}{2^{i+k} \cdot \delta^{(0)}}\right) A(2^{i-1+k} \cdot \delta^{(1)})$.

Letting k approach infinity we have

$$\frac{i+k}{2^{i+k} \cdot \delta^{(0)}} \to 0$$

$$\left(1 - \frac{1}{2^{i+k} \cdot \delta^{(0)}}\right) \to 1 \quad .$$

Hence if $A(\delta^{(0)}) = \lim_{L \to \infty} A(2^L \cdot \delta^{(0)})$ exists then so does

$A(\delta^{(1)}) = \lim_{L \to \infty} A(2^L \cdot \delta^{(1)})$ and $A(\delta^{(0)}) = A(\&))$.

The above construction, given a node n_0 at level i and a path

$\delta^{(0)}$ passing through n_0 constructs a corresponding path $\delta^{(1)}$

passing through n_1 at level i-1 , such that the average at n_0

depends on the average at n_1 and $A(\delta^{(0)})$ and $A(\delta^{(1)})$ are either

-both undefined, or both defined and equal. So i successive applications

of this construction reduces $\delta^{(0)}$ to a path $\delta^{(1)}$ such that

$\delta^{(0)}_{i+k} = \delta^{(i)}_{k}$ for all $k \geq 0$, and $A(\delta^{(i)})$ and $A(\delta^{(0)})$ are either

both undefined or both defined and equal. Consequently if α and β

have the same tail then they are reduced by this process to the same
path δ such that $\mathbf{a}_{i+k} = \beta_{j+k} = \delta_k$, for k > 0 for some i and j ,
and $A(\delta)$, $A(\alpha)$ and A(@) are either all undefined or all defined
and equal.   □

Corollary **3:** If α is a binary sequence ending with an infinite
sequence of 0's , then A(a) exists and A(a) = λ .

, Proof: Immediate from Theorems **3** and **5.**

Corollary **4:** If α is a binary sequence ending with an infinite
sequence of 1's , then $A(\alpha)$ exists and $A(\alpha)$ = 1 .

Proof: Immediate from Corollary 2 and Theorem **5.**

Corollary **5:**   There exists an α such that $A(\alpha)$ is undefined.

Proof: Let δ be any sequence ending with an infinite sequence
of 0's .   Since A(6) = λ , given any ϵ > 0 , there exists an N
such that for L >N , $|\lambda - A(2^L \cdot \delta)| < \epsilon$. Similarly for any
sequence y ending with an infinite sequence of 1's , since
$A(\gamma) = 1$, there exists an **N'** such that for L > **N'** ,
$|1 - A(2^L \cdot \gamma)| < \epsilon$. Now 1 < λ , so let ϵ > 0 be such that
$1 + \epsilon < \lambda - \epsilon$.   Then construct a as follows.

   Step (1): Let $\alpha_1 = \alpha_2 = \ldots = \alpha_{k_1} = 0$ where k_1 is the least
      integer such that $|\lambda - A(2^{k_1} \cdot \alpha)| < \epsilon$. Note that $2^{k_1} \cdot \alpha$
      is determined by the first k_1 bits of α only, so
      this condition is well defined.

18

Step (2): Now let $\alpha_{k_1+1} = \alpha_{k_1+2} = \ldots = \alpha_{k_2} = 1$ where k_2 is

the least integer greater than k_1 such that

$$\left| 1 - A(2^{k_2}\alpha) \right| < \epsilon .$$

Now at any odd step (r) add sufficiently many 0's to have

$\left| \lambda - A(2^{k_r} \bullet \quad a > 1 \right| < \epsilon$ and at any even step (s) add sufficiently many 1's

to have $\left| 1 - A(2^{k_s} \cdot \alpha) \right| < \epsilon$. It is clear that $A(\alpha)$ must be undefined

for such an α .

cl

Corollary 3 asserts that no matter which node of the tree we start

at, if we always take the left branch the asymptotic value of the

average is λ , while if we always take the right branch then the

asymptotic value of the average is 1 by Corollary 4.

The relation of two sequences α and β having the same tail is

clearly an equivalence relation. In virtue of Theorem **5** the asymptotic

value of the average is invariant **over** any equivalence class. Corollary

3 and 4 give two distinct equivalence classes that have two distinct

asymptotic values, and Corollary **5** shows that there are equivalence,

classes over which the asymptotic value of the average is undefined.

We conjecture that indeed the only two equivalence classes with defined

asymptotic values are those mentioned above.

6.   Remarks About Model 2

Table 1 shows the comparison between Model 2 and Model 1. The

values under Model 1 were computed using the recurrence relation of

Theorem 1.   For Model 2 the average for $n \leq 9$ was determined by

considering all possible inputs to the algorithm. For greater values

Table 1

Level	n	A(n) , according to the assumptions of:		
		Model 1	Model 2	Zig-Zag
0	1	0	0	0
1	2	0.50	0.50	0.50
	3	0.33	0.33	0.33
	4	0.88	0.92	0.59 ± 0.02
	5	0.67	0.67	0.83 ± 0.02
	6	0.75	0.75	0.66 ± 0.03
	7	0.57	0.55	0.83 ± 0.02
	8	1.14	1.24	1.08 + 0.03
	9	0.93	0.94	0.86 ± 0.03
	10	0.98	0.99 ± 0.03	1.07 ± 0.03
	11	0.79	0.81 ± 0.03	0.78 ± 0.03
	12	1.05	1.05 ± 0.03	1.07 ± 0.03
	13	0.85	0.87 ± 0.03	0.84 ± 0.03
	14	0.91	0.88 ± 0.03	1.05 ± 0.03
	15	0.73	0.77 ± 0.03	0.72 ± 0.03
	16	1.32	1.38 ± 0.04	0.82 ± 0.03
	17	1.11	1.14 ± 0.04	1.12 + 0.03
	18	1.14	1.14 + 0.04	1.00 ± 0.04
	19	0.96	0.94 ± 0.04	1.17 ± 0.03
	20	1.20	1.24 + 0.04	0.92 ± 0.04
	21	1.00	1.01 ± 0.04	1.26 ± 0.03
	22	1.05	1.01 ± 0.04	1.02 ± 0.04
	23	0.88	0.84 ± 0.04	1.18 + 0.04
	24	1.26	1.28 ± 0.04	0.83 ± 0.03
	25	1.05	1.08 ± 0.04	1.13 ± 0.03
	26	1.09	1.08 ± 0.04	0.99 ± 0.04
	27	0.91	0.91 ± 0.04	1.24 ± 0.04
	28	1.16	1.15 ± 0.04	0.91 ± 0.04
	29	0.95	0.96 ± 0.04	1.24 + 0.03
	30	1.01	1.00 ± 0.04	0.99 ± 0.04
	31	0.84	0.82 ± 0.04	1.23 + 0.04

Table 1 continued

Level	n	A(n) , according to the assumptions of:		
		Model 1	Model 2	Zig-Zag
5	32	1.43	1.50 ± 0.05	1.37 + 0.04
	33	1.22	1.27 ± 0.05	1.14 ± 0.04
	⋮			
	46	1.10	1.10 ± 0.04	1.19 + 0.03
	47	0.93	0.91 ± 0.04	0.93 ± 0.04
	48	1.40	1.43 ± 0.04	1.34 ± 0.04
	49	1.19	1.20 ± 0.04	1.10 ± 0.04
	⋮			
	62	1.08	1.03 ± 0.04	1.16 ± 0.04
	63	0.90	0.85 ± 0.04	0.88 ± 0.04
6	64	1.51	1.56 ± 0.05	0.96 ± 0.04
	65	1.30	1.33 ± 0.05	1.21 + 0.04
	⋮			
	94	1.13	1.08 ± 0.04	1.13 ± 0.05
	95	0.96	0.95 ± 0.04	1.35 + 0.04
	96	1.48	1.57 ± 0.05	0.96 ± 0.04
	97	1.27	1.30 ± 0.05	1.20 + 0.04
	⋮			
	126	1.12	1.05 ± 0.04	1.17 ± 0.05
	127	0.94	0.85 ± 0.04	1.39 ± 0.04
	128	1.55	1.74 ± 0.06	1.42 ± 0.05
	129	1.34	1.42 ± 0.05	1.20 ± 0.05
	⋮			
	190	1.15	1.09 ± 0.04	1.24 + 0.04
	191	0.98	0.91 ± 0.04	0.94 + 0.04
	192	1.53	1.63 ± 0.05	1.44 ± 0.05
	193	1.33	1.46 ± 0.06	1.20 ± 0.05
	⋮			
	254	1.14	1.01 ± 0.04	1.27 ± 0.04
	255	0.97	0.88 ± 0.04	0.92 ± 0.04

of n we simulated heap creation on 1000 randomly selected inputs, thus determining an estimate of the average and its interval of confidence. These results indicate that the average is relatively close in the two models. In general the behavior of Model 2 is more extreme than that of Model 1: the worst case, at each level, is now worse than the worst case in Model1 and asymptotically exceeds λ ; on the other hand the best case, at each level, is now better than in Model 1. In this section we give an intuitive explanation for this behavior, and suggest a method for smoothing out the difference between the worst and best case. We consider L-1 levels of the heap already created and examine what happens when inserting the nodes at level L . To simplify the notation we discuss the case where L = 3 , but the argument applies as well for the general case.

Let us first assume that heap H is random. (See Figure 2.) When k_8 is inserted it can exchange with k_4 , k_2 , and k_1 . These are the same nodes that k_9 will encounter, hence k_9 will be competing with numbers greater than or equal to k_8 's competition. Consequently the average at node 9 will be smaller than that at node 8 . Let us now look at the leftmost node of the right subtree, k_{12} . When it is inserted it will be compared with k_6 , k_3 and k_1 . The only one of these nodes that could possibly have been affected by previous insertions at this level is k_1 . But k_{12} is compared to k_1 only when it is greater than k_3 ; therefore we might expect that the average at 12 is only slightly smaller than the average at 8 . By this same kind of reasoning the average at 15 should be the smallest at this level. The above discussion shows that the average will have an undulatory behavior at each level.
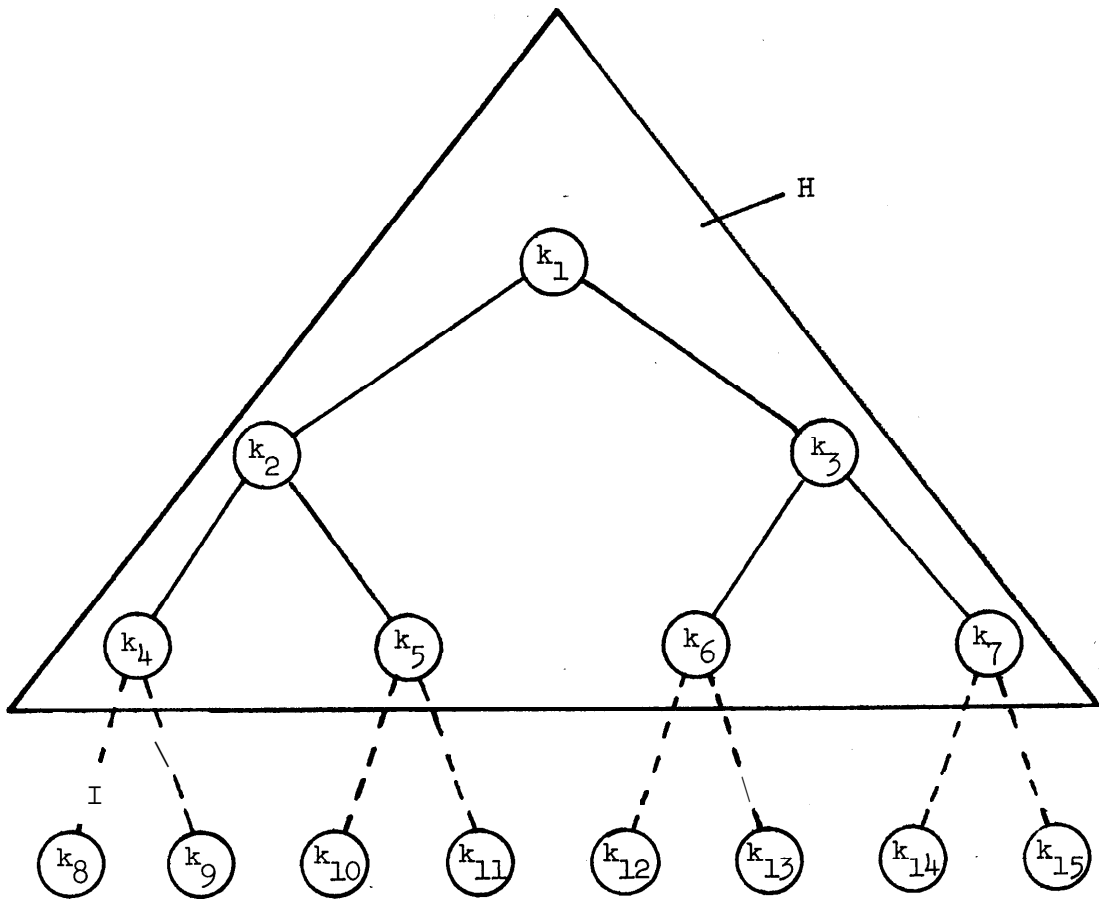
Figure2

Actually the heap H , as we know, will not be random, but this will only accentuate such behavior. The heap H is not random because large keys tend to drift to the right, thus further lessening the averages of the elements in the right **subtree.** To see this, consider the input as a random permutation of $\{1,2,\ldots,15\}$. We know that after all nodes have been inserted, 15 will be at the root, and we shall examine the chances of k_2 or k_3 being 14 . We have several cases to consider. The number 14 will settle as the left son k_2 if

(1) 15 and 14 both enter the tree on the left;

(2) one of them enters at the root and the other on the left;

(3) 14 enters on the left and 15 enters previously on the right;

(4) 14 enters on the right and 15 enters later on the left.

Similarly there are four corresponding cases where 14 settles as the right son k_3 . Comparing these cases we find that the difference between the probabilities that 14 settles at k_3 rather than k_2 is the probability that 14 enters on the same level as 15 but on the opposite side.

In order to dampen the latter effect we suggest a "zig-zag" method, alternating the direction of insertion at each level. Table 1 also shows the averages found by this "zig-zag" method, when even levels are inserted from right to left. The effect is to balance the tree more by upsetting the ordinary drift of large elements to the right.

7.   Acknowledgments

References

[Knuth - 1968]: D. E. Knuth, The Art of Computer programming, Vol. 1
Addison-Wesley, 1968.

[Knuth - 1973]: D. E. Knuth, The Art of Computer Programming, Vol. 3
Addison-Wesley, 1973.

[Williams - 1964]: J. W.J. Williams, Algorithm 232: HEAPSORT, CACM 7
(1964), 347-348.