MIT/LCS/TM-69

IMPROVED BOUNDS ON THE COSTS OF OPTIMAL AND BALANCED BINARY SEARCH TREES

Paul J. Bayer

November 1975

TM-69

IMPROVED BOUNDS ON THE COSTS OF
OPTIMAL AND BALANCED BINARY SEARCH TREES

by

Paul J. Bayer

November 1975

This research was supported by the National
Science Foundation under research
grant GJ-43634X, contract number DCR 74-12997

IMPROVED BOUNDS ON THE COSTS OF OPTIMAL AND BALANCED BINARY SEARCH TREES

by

Paul Joseph Bayer

Submitted to the Department of Electrical Engineering and Computer Science on May 9, 1975 in partial fulfillment of the requirements for the Degree of Master of Science.

ABSTRACT

A binary search tree can be used to store data in a computer system for retrieval by name. Different elements in the tree may be referenced with different probabilities. If we define the cost of the tree as the average number of elements which must be examined in searching for an element, then different trees have different costs. We show that two particular types of trees, weight balanced trees and min-max trees, which are easily constructed from the probability distribution on the elements, are close to optimal. Specifically, we show that for any probability distribution with entropy H,

$$H - \log_2 H - (\log_2 e - 1) \le C_{Opt} \le \begin{cases} C_{WB} \le H + 2 \\ C_{MM} \le H + 2 \end{cases}$$

where C_{Opt}, C_{WB}, and C_{MM} are the optimal, weight balanced, and min-max costs. We gain some added insight by deriving an expression for the expected value of the entropy of a random probability distribution.

THESIS SUPERVISOR: Ronald L. Rivest
TITLE: Assistant Professor of Computer Science and Engineering

Acknowledgements

I would like to gratefully acknowledge the assistance of some friends. My supervisor Ron Rivest allowed me to find my own way through this research and offered valuable criticism. Bob Cassels assisted in the preparation of this manuscript. Fred Hennie has been invaluable in helping direct the last four years of my education. And Terry, thank you for helping make this a sanely executed endeavor.

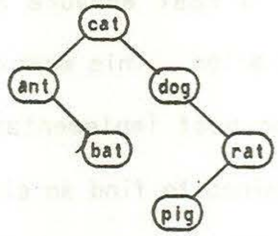
This research was supported by the National Science Foundation under research grant GJ-43634X.

Table of Contents

1.0 Introduction

Binary search trees can be used for data storage in a computer system when each piece of data is to be referenced by a name from some ordered set of names. Informally, a binary search tree is a binary tree in which each node is labelled with a name, such that in any subtree, the names in the left subtree are all less than the root name, and the names in the right subtree are greater than the root name.

For example, we can have a tree of animal names ordered alphabetically:



The algorithm for finding a name x in the tree is:

1) If the tree is empty, then the search fails.
2) Compare x with the root name.
3) If they are equal, then x has been found.
4) If x comes earlier alphabetically, then recursively search the left

subtree for x.

5) If x comes later, then recursively search the right subtree for x.

So to find "bat" in the above tree, it is compared with "cat" and found to come earlier. Then it is compared with "ant" and found to come later. Finally, it is compared with "bat", and is found. More information on binary search trees including applications can be found in Knuth [8], Severance [13], and Nievergelt [10]. The last has a good bibliography of the area.

One quantity that is important in determining whether a binary search tree should be used in a particular application is the average time needed to find a name. We will be dealing with trees abstractly, and so will be interested in a cost measure defined on the trees, which is independent of implementation. This measure, which accurately reflects the average search time for most implementations, is the average number of nodes which must be examined to find an element. We call this quantity the cost of the tree. The cost depends not only on the structure of the tree, but also on the probability distribution on the names, which indicates their frequency of reference. The cost also depends on the possibility that a name searched for is not in the tree. Note, also, that the actual names and the data stored with the names have no effect on our cost measure, so we will assume that the names are 1, 2, 3, ...

From a set of n names it is possible to construct $^{(2n)!}/_{(n!\ (n+1)!)}$

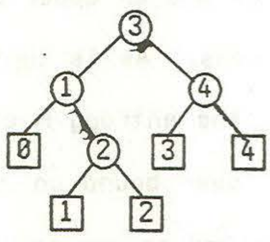
or about 4^n different trees. The optimal tree is the one of lowest cost (actually there may be many such trees). We are also interested in the weight balanced tree, in which the root of each subtree is chosen to most equally balance the probabilities contained in each of its subtrees, and in the min-max tree, in which each root is chosen to minimize the larger of the probabilities in each of its subtrees. The best algorithm known for constructing optimal trees runs in time $O(n^2)$ (Knuth [7]). Hu and Tucker [5] have an algorithm for a restricted case, which builds an optimal tree in time $O(n \log n)$. Fredman [2] has recently discovered an algorithm which can be used to build weight balanced and min-max trees in time $O(n)$. Since we will show that weight balanced and min-max trees are close to optimal, it might often be better to use Fredman's algorithm to build one of these trees instead of using the optimal tree. Walker and Gotlieb [14] and Bruno and Coffman [1] have empirically shown that weight balanced trees are close to optimal.

We show that these trees are good by deriving a lower bound on the cost of the optimal tree and an upper bound on the cost of the weight balanced and min-max trees. As is typical in information theory, our bounds are in terms of the entropy H of the probability distribution. Melhorn [9] derived a lower bound on the cost of the optimal tree of $H/_{\log 3}$, and an upper bound on the cost of the weight balanced tree of $3.42\,H + 2$. Rissanen [12] treated a special case corresponding to always having an unsuccessful search, and proved an upper bound on the cost of the weight balanced tree of H + 3. Our upper bound proofs are improvements on his. We improve on all previous bounds with

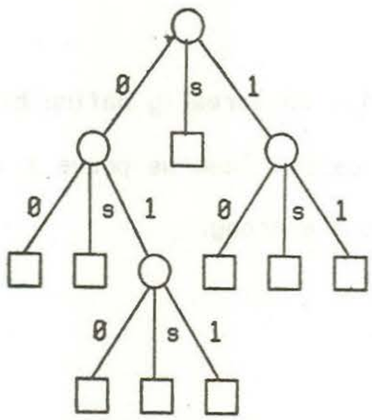
$$H - \log_2 H - (\log_2 e - 1) \le C_{Opt} \le \begin{cases} C_{WB} \le H + 2 \\ \\ C_{MM} \le H + 2 \end{cases}$$

where C_{Opt}, C_{WB}, and C_{MM} are the optimal, weight balanced, and min-max costs respectively.

Our results are also applicable to some coding problems in information theory. If we consider encoding source messages using a prefix code over the ternary alphabet {0,1,s}, with the restriction that s can appear only at the end of a code word, then every binary search tree corresponds to such a code. For example



corresponds to the code tree



Our lower bounds hold for all such codes. Our upper bounds hold for codes constructed as our trees are constructed. See Gallager [3] for more information on coding applications.

The remainder of this thesis is organized as follows. Chapter 2 contains definitions and some useful preliminary lemmas. The lower bounds are presented in Chapter 3. Chapter 4 gives the upper bounds. Added insight is given in Chapter 5 by showing how large the entropy can be expected to be.

2.0 Preliminaries

In this section we formally define binary search trees, and define the notations to be used. Then we prove a simple but useful lemma about trees and a lemma about entropy.

2.1 Definitions

Define a binary search tree T with n nodes (1), ... ,(n) and n+1 leaves $\boxed{0}$, ... ,$\boxed{n}$ as a binary tree such that if (i) is the root of a subtree t then all nodes (k) and leaves $\boxed{k}$ in the left subtree of t satisfy $k < i$, and all nodes (k) and leaves $\boxed{k}$ in the right subtree of t satisfy $k \geq i$. The level of (i), l_i, is the number of nodes from the root of T to (i), counting the root and (i). The level of $\boxed{i}$, l_i', is the level of the parent of $\boxed{i}$. A probability distribution over the nodes and leaves is a sequence of non-negative real numbers $p_1, \ldots, p_n, q_0, \ldots, q_n$ such that $\sum_{1 \leq k \leq n} p_k + \sum_{0 \leq k \leq n} q_k = 1$. The entropy of a probability distribution is defined by $H(x_1, \ldots, x_m) = \sum_{1 \leq k \leq m} x_k \log {}^1/_{x_k}$. (Unless otherwise specified, all log's in this thesis are to the base 2. Define $0 \log {}^1/_0 = 0$.) Given a probability distribution and a tree T, we define:

(1) C_T = the cost of T

= the expected number of comparisons made when searching for an element

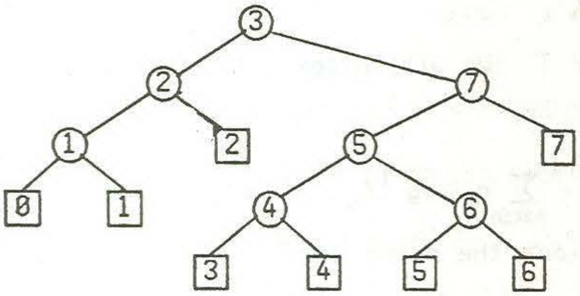
$$= \sum_{1 \le k \le n} p_k l_k + \sum_{0 \le k \le n} q_k l_k'$$

(2) H_T = the entropy of T (We will often omit the subscript T on C_T and H_T.)

$$= \sum_{1 \le k \le n} p_k \log {}^1/_{p_k} + \sum_{0 \le k \le n} q_k \log {}^1/_{q_k}.$$

In our formal definitions the nodes correspond to the elements stored in the tree. The leaves represent the positions in the tree where unsuccessful searches terminate. That is, $\boxed{k}$ is the termination point of a search for some name between the k<u>th</u> name in the tree and the k+1<u>st</u> name in the tree. The levels correspond to the number of nodes which must be examined before a search terminates, either successfully or unsuccessfully. The probability p_k is the probability that a search is for the k<u>th</u> element in the tree. The probability q_k is the probability that a search is for a name that falls between the k<u>th</u> and k+1<u>st</u> elements in the tree. With these definitions, then, the cost as defined is indeed the average number of comparisons made during a search.

For example:



$l_3 = 1$

$l_2 = l_7 = l_2' = l_7' = 2$

$l_1 = l_5 = l_0' = l_1' = 3$

$l_4 = l_6 = l_3' = l_4' = l_5' = l_6' = 4$

If $p_1 = \dots = p_7 = {}^1/_{14}$, $q_0 = \dots = q_7 = {}^1/_{16}$ then $C = {}^{19}/_{14} + {}^{26}/_{16}$, and $H = {}^{(\log 14)}/_2 + {}^{(\log 16)}/_2$.

For the sum of the probabilities of the nodes and leaves in the sequence $\boxed{0}$, (1), ... ,(n),$\boxed{n}$ between (i) and (j) including the endpoints, we will use the notation $P((i),(j)) = \sum_{i\le k\le j} p_k + \sum_{i\le k\le j-1} q_k$, and analogously for $P(\boxed{i},\boxed{j})$, $P((i),\boxed{j})$, $P(\boxed{i},(j))$. Also, for each subtree t of T with nodes and leaves $\boxed{i-1}$,(i), ... ,(j),$\boxed{j}$ define:

(1) r_t is the root of t,

(2) L_t is the left subtree of t,

(3) R_t is the right subtree of t,

(4) S_t is the set of all subtrees of t, including t, but not including the subtrees composed of single leaves,

(5) $P_t = P(\boxed{i-1},\boxed{j})$ is the total probability in t,

(6) $L_t((k)) = P(\boxed{i-1},\boxed{k-1})$ is the value that P_{L_t} would have if (k) were the

root of t.

(7) $R_t(\text{k}) = P(\boxed{k}, \boxed{j})$ is analogous to (6).

(8) $H_t = \sum_{i \le k \le j} {}^{p_k}/_{P_t} \log {}^{P_t}/_{p_k} + \sum_{i-1 \le k \le j} {}^{q_k}/_{P_t} \log {}^{P_t}/_{q_k}$ is the normalized entropy of t,

(9) $C_t = \sum_{i \le k \le j} {}^{p_k}/_{P_t} (l_k - l_{r_t} + 1) + \sum_{i-1 \le k \le j} {}^{q_k}/_{P_t} (l_k' - l_{r_t} + 1)$ is the cost of t when viewed as an isolated, normalized tree,

(10) $E_t = H({}^{p_{r_t}}/_{P_t}, {}^{P_{L_t}}/_{P_t}, {}^{P_{R_t}}/_{P_t})$ is the entropy of the split at t.

If $P_t = 0$, then $H_t = C_t = E_t = 0$. Note that if t is $\boxed{i}$ then r_t, L_t, R_t, E_t are undefined, $S_t = \emptyset$, $P_t = q_i$, and $H_t = C_t = 0$.

From the above definitions with some algebraic manipulation, we get

$$C_t = 1 + {}^{P_{L_t}}/_{P_t}\, C_{L_t} + {}^{P_{R_t}}/_{P_t}\, C_{R_t} \qquad (1)$$

$$H_t = E_t + {}^{P_{L_t}}/_{P_t}\, H_{L_t} + {}^{P_{R_t}}/_{P_t}\, H_{R_t} \qquad (2)$$

unless $P_t = 0$, in which case $C_t = H_t = 0$.

2.2 Tree Lemma

The following simple lemmas about trees are useful:

Lemma 2.1 If f(t) is a function defined on all subtrees of T by

$$f(t) = \begin{cases} 0 \text{ if } t \text{ is a leaf or } P_t = 0 \\ g(t) + {}^{P_{L_t}}/_{P_t}\, f(L_t) + {}^{P_{R_t}}/_{P_t}\, f(R_t) \quad \text{otherwise} \end{cases}$$

for some function g, then

$$f(t) = \begin{cases} 0 \text{ if } t \text{ is a leaf or } P_t = 0 \\ {}^1/_{P_t} \sum_{t' \in S_t} P_{t'} g(t') \quad \text{otherwise.} \end{cases}$$

Proof (by induction on the structure of the tree)

Basis If t is a leaf then the lemma is clearly true.

Induction If $P_t = 0$ then it is clearly true. If $P_t \neq 0$ and if $P_{L_t} \neq 0$ and $P_{R_t} \neq 0$ then

$$\begin{aligned} f(t) &= g(t) + {}^{P_{L_t}}/_{P_t}\ f(L_t) + {}^{P_{R_t}}/_{P_t}\ f(R_t) \\ &= {}^1/_{P_t}\ (P_t\ g(t) + \sum_{t' \in S_{L_t}} P_{t'} g(t') + \sum_{t' \in S_{R_t}} P_{t'} g(t')) \\ &\qquad \text{(by the induction hypothesis)} \\ &= {}^1/_{P_t} \sum_{t' \in S_t} P_{t'} g(t') \\ &\qquad (\text{since } S_t = \{t\} \cup S_{L_t} \cup S_{R_t}). \end{aligned}$$

If $P_{L_t} = 0$ ($P_{R_t} = 0$), then for all $t' \epsilon S_{L_t}$ ($t' \epsilon S_{R_t}$), $P_{t'} = 0$, and the lemma is true. □

Lemma 2.2 If f is defined as in Lemma 2.1 then $f(T) = \sum_{t \epsilon S_T} P_t g(t)$.

Proof This is Lemma 2.1 with $P_T = 1$. □

Lemma 2.3 $C = \sum_{t \epsilon S_T} P_t$, $H = \sum_{t \epsilon S_T} P_t E_t$.

Proof See equations (1) and (2) above and Lemma 2.2. □

2.3 Entropy Lemma

The following lemma about entropy is useful:

Lemma 2.4 (1) If $x_1+x_2+x_3 = 1$, then $H(x_1,x_2,x_3) \geq H(x_1+x_2,x_3)$.
(2) If $x \leq {}^1/_2$, then $H(x,1-x) \geq 2x$.

Proof The proof is straightforward. See Gallager [3]. $\square$

3.0 Lower Bounds on the Cost of the Optimal Tree

The problem of determining a good lower bound on the cost of a binary search tree seems not to have been studied in great detail. Melhorn [9] derived the bound $C \geq {}^{H}/_{\log 3}$, with a proof involving complex manipulations. In fact, we can get the same bound by noting, as we did before, that every binary search tree corresponds to a ternary code tree with the same cost and entropy. Then a theorem from information theory yields the same bound (Gallager [3], p. 50). (Note also that this theorem also yields $C \geq H$ when $\sum p_k = 0$.) What seems to have discouraged further work is that this bound is achievable. However, it is achievable only for $H < 3 \log 3$ as we shall see later.

We present next an easy lower bound which is better than the bound above. The proof is easier than that for our best bound, and we can give an argument for its plausibility.

The average amount of information which must be learned in finding an element in the tree is H bits. Each of the C comparisons, except the last, results in an answer of < or >. This gives one bit of information per comparison, or C - 1 bits total. The last comparison indicates that the search has ended. In other words, it tells how many levels had to be searched. In the optimal tree, the average number of levels is less than or equal to log (n+1). This means that the last comparison yields essentially log log (n+1) bits of information. Therefore, H is about (C-1) + log log (n+1). In fact we prove that $C \geq H - P\,(\log\log(n+1)-1)$, where $P = \sum p_k$.

As a lemma, we prove a Kraft-like inequality for this type of tree.

Lemma 3.1 In any binary search tree T

$$\sum_{0\le k\le n} 2^{-l_k'} \le 1$$

$$\sum_{1\le k\le n} 2^{-l_k} \le (\log(n+1))/2.$$

Proof (by induction on n)

Basis If n = 0 then T is $\boxed{0}$ and

$$\sum_{0\le k\le 0} 2^{-l_k'} = 2^{-0} = 1$$

$$\sum_{1\le k\le 0} 2^{-l_k} = 0.$$

Induction (n > 0) . T is made up of the root node (r), the left subtree, with r-1 nodes and r leaves, and the right subtree, with n-r nodes and n-r+1 leaves. The values of l_k and l_k' are one greater in the whole tree than the corresponding values in the subtrees. Thus, inductively we have

$$\sum_{0\le k\le r-1} 2^{-l_k'} \le 1/2$$

$$\sum_{1\le k\le r-1} 2^{-l_k} \le (\log r)/4$$

$$\sum_{r\le k\le n} 2^{-l_k'} \le 1/2$$

$$\sum_{r+1\le k\le n} 2^{-l_k} \le (\log(n-r+1))/4.$$

Also, $2^{-l_r} = 1/_2$. So

$$\sum_{0\le k\le n} 2^{-l_k'} \le 1$$

$$\sum_{1\le k\le n} 2^{-l_k} \le (1 + {}^{(\log r + \log (n-r+1))}/_{2})/_{2} = (\log (2\ (r(n-r+1))^{1/2}))/_{2}$$

But $(n+1)^2 - 4\ (r(n-r+1)) = n^2 - 4nr + 2n - 4r + 4r^2 + 1$

$$= ((n-r) - (r-1))^2 \ge 0.$$

So $\sum_{1\le k\le n} 2^{-l_k} \le {}^{(\log (n+1))}/_{2}.$ $\square$

<u>Theorem 3.2</u> In any binary search tree

$$C \ge H - P\ (\log \log (n+1) - 1)$$

where $P = \sum p_k$.

<u>Proof</u> With Lemma 3.1 we can prove the theorem in the same way as the variable length source coding theorem is proved in Gallager [3] (p. 50). That is:

$H - P\ (\log \log (n+1)-1) - C$

$$= \sum_{1\le k\le n} p_k \log ({}^{2\ 2^{-l_k}}/_{(p_k \log (n+1))}) + \sum_{0\le k\le n} q_k \log ({}^{2^{-l_k'}}/_{q_k})$$

$$\le (\sum_{1\le k\le n} p_k ({}^{2\ 2^{-l_k}}/_{(p_k \log(n+1))} - 1) + \sum_{0\le k\le n} q_k ({}^{2^{-l_k'}}/_{q_k} - 1)) \log e$$

(since $\log z \le (z-1) \log e$)

$$= ({}^{2}/_{\log (n+1)} \sum 2^{-l_k} - 1 + \sum 2^{-l_k'} - 1) \log e$$

$$\le ((1 - 1) + (1 - 1)) \log e$$

(from Lemma 3.1)

$$= 0. \square$$

We will see in section 5.0 that H is almost always close to $\log(2n+1)$. Therefore, this bound is generally better than $^{H}/_{\log 3}$.

Another attack on the lower bound comes from Lemma 2.3. Using $H(x_1,\ldots,x_m) \le \log m$, we get $E_t \le \log 3$ for all t, so

$$H = \sum_{t\in S_T} P_t E_t \le \sum_{t\in S_T} P_t \log 3 = C \log 3,$$

giving still another proof of the information theoretic bound. We can do better as follows:

<u>Lemma 3.3</u> For any real number b and any $x_1, x_2, x_3 \ge 0$ such that $x_1+x_2+x_3 = 1$, $H(x_1,x_2,x_3) \le \log(2+2^{-b}) + bx_1$.

<u>Proof</u> The function $H(x_1,x_2,x_3)$ is a concave function (Gallager [3], p. 85) and is less than or equal to $f(x_1) = H(x_1, {}^{1-x_1}/_2, {}^{1-x_1}/_2)$ (Gallager [3], p. 508). The graph of the function $f(x_1)$ can be bounded from above by a line with slope b tangent to f. Solving $^{df}/_{dx_1} = -(\log x_1 + \log e) + (\log(1-x_1) + \log e) - 1 = b$ gives $x_1 = {}^1/_{(2^{b+1}+1)}$, $f({}^1/_{(2^{b+1}+1)}) = \log(2+2^{-b}) + {}^b/_{(2^{b+1}+1)}$. The equation of the line with slope b tangent at that point yields the lemma. $\square$

<u>Theorem 3.4</u> For any b, $C \ge {}^{(H - bP)}/_{\log(2+2^{-b})}$, where $P = \sum_{1\le k\le n} p_k$.

<u>Proof</u> From Lemma 3.3 we get

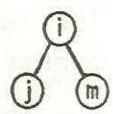
$$E_t = H({}^{p_{r_t}}/_{P_t}, {}^{P_{L_t}}/_{P_t}, {}^{P_{R_t}}/_{P_t}) \le \log(2+2^{-b}) + b\, {}^{p_{r_t}}/_{P_t}.$$

So $H = \sum_{t\in S_T} P_t E_t \le \sum_{t\in S_T} P_t \log(2+2^{-b}) + \sum_{t\in S_T} P_t\, b\, {}^{p_{r_t}}/_{P_t}$. And from Lemma 2.3

$$H \le C \log (2+2^{-b}) + b \sum_{t \in S_T} p_{r_t}.$$

But each node is the root of one subtree, so $\sum_{t \in S_T} p_{r_t} = \sum_{1 \le k \le n} p_k$ and the theorem is proved. $\square$

The bound of Theorem 3.4 is tight for all b since there exist trees which come arbitrarily close to the bound. Specifically, the complete tree with 2^k-1 nodes, in which all occurrences of three nodes in the form



satisfy ${}^{p_i}/_{p_j} = {}^{p_i}/_{p_m} = 2 + 2^{-b}$, and in which the q's are 0, has $C = a^{-1} \sum_{1 \le i \le k} i\ 2^{i-1}\ x^{-i}$ where $x = 2 + 2^{-b}$, $a = \sum_{1 \le i \le k} 2^{i-1}\ x^{-i} = {}^{(x^k - 2^k)}/_{x^k(x-2)}$, and $\log a = b + \log (1-(1+2^{-b-1})^{-k})$. Also,

$$\begin{aligned} H &= \sum_{1 \le i \le k} 2^{i-1}\ x^{-i}\ a^{-1} \log (ax^i) \\ &= a^{-1} \log a \sum_{1 \le i \le k} 2^{i-1}\ x^{-i} + a^{-1} \log x \sum_{1 \le i \le k} i\ 2^{i-1}\ x^{-i} \\ &= \log a + C \log x. \end{aligned}$$

This yields

$$C = {}^{(H - b - \log (1 - (1+2^{-b-1})^{-k}))}/_{\log (2+2^{-b})}$$

which approaches the bound of Theorem 3.4 as $k \to \infty$. But $\log a < b$, and $C = ((k-1)2^k + x^{k+1})(x^k-2^k)^{-1}(x-2)^{-1} < 2^{b+1} + 1$, so $H < b + (2^{b+1}+1) \log (2+2^{-b})$. For any value of b for which H exceeds this bound, the corresponding lower bound for C in Theorem 3.4 cannot be achieved.

This leads us to try to find the value of b (as a function of H) which maximizes the bound $f(b) = {}^{(H-bP)}/_{\log\,(2+2^{-b})}$. If we look at

$$f'(b) = {}^{\left({}^{(H-bP)}/_{(2^{b+1}+1)} - P \log\,(2+2^{-b})\right)}/_{(\log\,(2+2^{-b}))^2}$$

there seems to be no good closed form solution to $f'(b) = 0$. The value $b = \log {}^{H}/_{2P}$ is close to the solution, so we get:

<u>Theorem 3.5</u> If $H \geq 1$, then $C \geq H - P\,(\log {}^{H}/_{P} + \log e - 1)$.

<u>Proof</u> Substituting $\log {}^{H}/_{2P}$ for b in Theorem 3.4, we get

$$C \geq {}^{(H - P \log {}^{H}/_{P} + P)}/_{(1 + \log {}^{(H + P)}/_{H})}$$

$$= {}^{(P - H \log {}^{(H+P)}/_{H} + P \log {}^{H}/_{P} \log {}^{(H+P)}/_{H})}/_{(1 + \log {}^{(H+P)}/_{H})} + H - P \log {}^{H}/_{P}$$

By using $\log z \leq (z-1) \log e$ (Gallager [3], p. 23), we have $\log {}^{(H+P)}/_{H} \leq {}^{P}/_{H} \log e$. Also, $P \log {}^{H}/_{P} \log {}^{(H+P)}/_{H} \geq 0$. This gives

$$C \geq H - P \log {}^{H}/_{P} + {}^{(P - P \log e)}/_{(1 + \log {}^{(H+P)}/_{H})}.$$

Since $\log e > 1$, $P - P \log e < 0$, and since $\log {}^{(H+P)}/_{H} \geq 0$, we get $C \geq H - P \log {}^{H}/_{P} + P - P \log e.$ $\square$

Note that if $P = 0$, we get the classical information theoretic bound $C \geq H$. The bound is least when $P = 1$ and

$$C \geq H - \log H - \log e + 1.$$

This bound beats the bound ${}^{H}/_{\log 3}$ for $H \geq 11$. We have found values for b that result in a bound which beats ${}^{H}/_{\log 3}$ for smaller H, but they exhibit the same asymptotic behavior (in H).

4.0 Upper Bounds for Balanced Trees

In this section we show that various balanced tree schemes are good by establishing upper bounds on the costs of such trees. Balanced, here, means balanced in probability. Knuth [7] first proposed weight balanced trees as an area for research. Melhorn [9] has published an upper bound for weight balanced trees, but the bound presented here is better. The discovery by Fredman [2] of an algorithm for constructing balanced trees in linear time has generated special interest in such trees. The best known algorithm for constructing optimal trees runs in time $O(n^2)$ (Knuth [7]).

Throughout this section we will be talking about a subtree t made up of $\boxed{i-1}$, (i), ... , (j), $\boxed{j}$, and so will omit subscripts when the context is clear.

We would like to formally capture the idea of balanced trees. A logical starting point is to select as root the node (k) closest to the center of the probability in P_t. Unfortunately, even if (k) is exactly in the center (that is $R((k)) \le {}^{P_t}/_2$, $L((k)) \le {}^{P_t}/_2$), it might not be the node which gives the most even split between the left and right subtree probabilities. For example, if $n = 3$, $p_1 = {}^5/_8$, $p_2 = {}^1/_{16}$, $p_3 = {}^5/_{16}$, and all the q's are zero, then (1) falls in the center of the probability, but $R((1))-L((1)) = {}^6/_{16} > L((2))-R((2)) = {}^5/_{16}$. (It is this anomaly which motivated the idea of min-max trees, in which the same situation does not occur.) We now define formally the notion of balanced, which avoids this problem, and which facilitates the proofs which follow.

The <u>middle leaf</u> of t is the leaf closest to the middle in probability. Formally, $\boxed{m}$ is defined by

Let $K = \{k \mid i-1 \le k \le j$ and $\min(P(\boxed{i-1},\boxed{k}),P(\boxed{k},\boxed{j}))$ is maximum (or, equivalently, $\min(L(k+1),R(k))$ is maximum)$\}$. Let a equal this maximum. If there exists a $k \in K$ for which $P(\boxed{k},\boxed{j}) = a$, then m is the smallest such k. Otherwise, m is the largest $k \in K$ for which $P(\boxed{i-1},\boxed{k}) = a$.

The node r is said to <u>generally balance</u> t, if $r = m$ or $r = m+1$. A tree T is <u>generally balanced</u> if, for all $t \in S_T$, r_t generally balances t. Generally balanced trees are provably good in cost relative to the optimal cost, and include weight balanced and min-max trees.

The following lemma describes the structural implications of this definition. It will often be used implicitly in the proofs which follow, especially (1) and (2).

<u>Lemma 4.1</u> If $\boxed{m}$ is the middle leaf then

(1) For all k, $i \le k \le m$, $R(k) > L(k)$

(2) For all k, $m+1 \le k \le j$, $R(k) \le L(k)$

(3) For all k, $i \le k \le m$, $L(k) < {}^{P_t}/_2$

(4) For all k, $m+1 \le k \le j$, $R(k) \le P_t/2$

(5) $L(m+1) \ge (P_t - p_{m+1})/2$

(6) $R(m) \ge (P_t - p_m)/2$

Proof (1) If $R(m) > L(m+1)$ then

$$R(k) \ge R(m) > L(m+1) \ge L(k).$$

So assume $R(m) \le L(m+1)$ and assume that $R(k) \le L(k)$ for contradiction. Then

$$L(m+1) \ge L(k+1) \ge L(k) \ge R(k) \ge R(m).$$

But then $\boxed{k}$ would be the middle leaf.

(2) Analogous to (1).

(3) This follows from (1) and the fact that

$$L(k) + R(k) \le P_t.$$

(4) Analogous to (3).

(5) If $L(m+1) \ge R(m)$ then, since

$$L(m+1) + R(m) \ge P_t,$$

$$L(m+1) \ge P_t/2 \ge (P_t - p_{m+1})/2.$$

So assume $L(m+1) < R(m)$, and for contradiction assume that $(P_t - p_{m+1})/2 > L(m+1)$. We have

$$R(m+1) + L(m+1) + p_{m+1} = P_t$$

so (using (4)),

$$L(m+2) \ge P_t/2 \ge R(m+1) > (P_t - p_{m+1})/2 > L(m+1).$$

But then $\boxed{m+1}$ would be the middle leaf.

(6) Analogous to (5).$\square$

A node r is said to weight balance t if r minimizes $|L(r) - R(r)|$. A node r is said to min-max balance t if r minimizes $\max(L(r), R(r))$.

Theorem 4.2 (1) At least one of the nodes which generally balances t weight balances t.

(2) At least one of the nodes which generally balances t min-max balances t.

Proof (1) We will show that if $r < m$ then

$$|L(r) - R(r)| \geq |L(m) - R(m)|$$

and similarly for $r > m+1$. Then one of m or $m+1$ must weight balance t. For $r < m$, from Lemma 4.1 (1) we get

$$|L(r) - R(r)| = R(r) - L(r)$$

and similarly for m. So

$$R(r) - L(r) - R(m) + L(m) =$$
$$P(\boxed{r}, m) + P(r, \boxed{m-1}) \geq 0$$

and so

$$|L(r) - R(r)| \geq |L(m) - R(m)|$$

The analogous argument holds for $r > m+1$.

(2) For $r \leq m$, $\max(L(r), R(r)) = R(r)$ by Lemma 4.1 (1). So we

need to show that for $r < m$, $R(r) \geq R(m)$. But this is clearly true since $r < m$. And analogously for $r > m+1$. $\square$

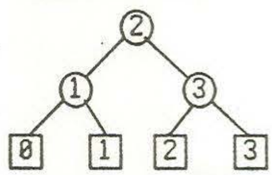
We can therefore define the weight balanced root of t as m if it weight balances t, and otherwise, $m+1$. Define the min-max root of t as m if it min-max balances t, and otherwise $m+1$. Then the weight balanced tree is the tree in which the root of each subtree is the weight balanced root. The min-max tree is the tree in which the root of each subtree is the min-max root.

In general we are interested in specific sub-types of generally balanced trees, defined by the rules for choosing between m and $m+1$ for the root. Of greatest interest are those sub-types which have rules that can be computed in constant time for each subtree, perhaps with the benefit of some linear time pre-conditioning of the entire tree. In this case, Fredman's algorithm can be used for constructing the trees in linear time. Both weight balanced trees and min-max trees are such trees.

We believe intuitively that, on the average, min-max trees are better than weight balanced trees. The following is an argument for this claim. If, in any subtree t, the min-max and weight balanced roots differ (say m is the min-max root), then $R(m) \leq L(m+1)$ (by the min-max definition and Lemma 4.1 (2)). But it is also true that $L(m) \leq R(m+1)$ or m would be the weight balanced root. That is, each subtree of the min-max tree has less total probability than the the subtree on the other side of the weight balanced tree. Therefore, the probability that a search of t will stop at the root is greater in the min-max case than in the weight balanced case. If the cost of a subtree were a monotone function of the weight, then the min-max tree would be uniformly better.

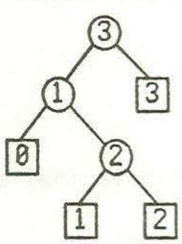
Unfortunately, there is not a strict hierarchy between these two kinds of trees. Consider $n = 3$, $p_1 = q_0 = {}^{1}/_{6}$, $p_2 = q_2 = {}^{1}/_{8}$, $p_3 = q_1 = 0$, $q_3 = {}^{5}/_{12}$, then the min-max tree is

T_A:



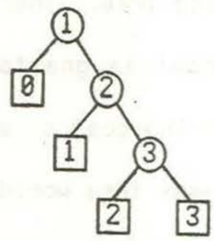
and $C = {}^{45}/_{24}$. The weight balanced tree is

T_B:



and $C = {}^{44}/_{24}$. The weight balanced tree has lower cost. But if $p_1 = {}^{5}/_{12}$, $p_2 = q_3 = {}^{1}/_{6}$, $q_1 = {}^{1}/_{4}$, $p_3 = q_0 = q_2 = 0$, then the min-max tree is

T_C:



and $C = {}^{21}/_{12}$, while T_A is the weight balanced tree with $C = {}^{22}/_{12}$. The min-max tree is better. Finally, neither type is necessarily optimal, since if $p_1 = p_3 = {}^{1}/_{2}$ and all the others are zero, then T_A is both the min-max tree and the weight balanced tree with $C = 2$. However, T_B is optimal with $C = {}^{3}/_{2}$.

4.1 Generally Balanced Trees

We can prove an upper bound on the cost of a generally balanced tree.

Lemma 4.3 Let $\boxed{m}$ be the middle leaf of t with root r.

If $m = r$ and $m \neq j$ then either

(A) $P_t/2 > P_L + p_m \geq (P_t - (2q_m + p_{m+1}))/2$

or (B) $P_t/2 \geq P_R \geq (P_t - p_m)/2$.

If $m = r = j$ then either

(C) $P_t/2 > P_L + p_m \geq (P_t - q_m)/2$

or (D) $P_t/2 \geq P_R \geq (P_t-p_m)/2$.

And the symmetric formulas hold for $m+1 = r$.

Proof (A), (B), (D) follow easily from Lemma 4.1 (5), (6). (C) is easy since when $m = r = j$, $P_R = q_m$, so

$$P_L + p_m = P_t - P_R = P_t - q_m \geq (P_t-q_m)/2. \square$$

Theorem 4.4 In a generally balanced tree $C \leq H + 3$. And the bound is tight.

Proof We will bound E_t for each t. Assume (A) in Lemma 4.3 holds. Then

$$E_t \geq H((P_L+p_m)/P_t, P_R/P_t) \geq 2\,(P_L+p_m)/P_t \geq 1 - (2q_m+p_{m+1})/P_t$$

by Lemma 2.4. Let $b_t = 2q_m + p_m$ in this case. Similarly we have

$$E_t \geq 1 - p_m/P_t, \text{ letting } b_t = p_m \text{ for (B) and (D)}$$

$$E_t \geq 1 - q_m/P_t, \text{ letting } b_t = q_m \text{ for (C)},$$

and the symmetric formulas hold for $m+1 = r$. Then

$$H = \sum_{t \in S_T} P_t E_t \geq \sum P_t - \sum b_t = C - \sum b_t.$$

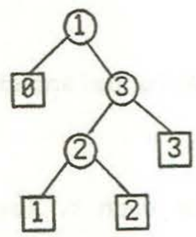
Let us see how q_k can appear in $\sum b_t$. If (k) is higher in the tree than (k+1), then q_k can appear in $2q_k+p_{k+1}$ (A) when (k) is the root, and it can appear in q_k (C) when (k+1) is the root. If (k) is lower than (k+1), the result is symmetric. Thus the coefficient of q_k in $\sum b_t$ is ≤ 3. Examining p_k, it can appear in (A) at most once when (k-1) is the root and at most once when (k+1) is the root. And it can appear in one of (B) or (D), but not both. So the coefficient of p_k in $\sum b_t$ is ≤ 3.

Therefore,

$$H \geq C - \sum b_t \geq C - (3 \sum_{1 \leq k \leq n} p_k + 3 \sum_{0 \leq k \leq n} q_k) = C - 3.$$

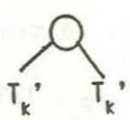
The bound is tight, since we can define a family of generally balanced trees as follows:

T_0:



with $p_1 = p_3 = q_3 = 0$, $q_0 = q_2 = 2\epsilon$, $q_1 = \epsilon$, $p_2 = 1-5\epsilon$,

T_{k+1}:



where the root probability is 0 and, T_k' is T_k with all probabilities scaled down by $^1/_2$. These trees are generally balanced since $\boxed{1}$ is the middle leaf of T_0, and each T_k for $k > 0$ is perfectly balanced. Then for T_k, $C \to k+3$, $H \to k$, as $\epsilon \to 0$. □

The upper bound of Theorem 4.4 is not especially interesting, since we can do better for weight balanced and min-max trees. The proof technique is of interest, however, since it will be used, with more careful, detailed analysis of each subtree, to get upper bounds on these two kinds of trees.

4.2 Min-max Trees

We can get an upper bound on the cost of min-max trees with an easy modification of the proof for generally balanced trees.

Lemma 4.5 Let $\boxed{m}$ be the middle leaf of t with min-max root r.

If $m = r$ and $m \neq j$ then either

(A) $P_t/2 > P_L + p_m \geq (P_t - q_m)/2$

or (B) $P_t/2 \geq P_R \geq (P_t - p_m)/2$.

If $m = r = j$ then either

(C) $P_t/2 > P_L + p_m \geq (P_t - q_m)/2$

or (D) $P_t/2 \geq P_R \geq (P_t - p_m)/2$.

And the symmetric formulas hold for $m+1 = r$.

Proof (B), (C), (D) are the same as in Lemma 4.3. To get (A), assume $m = r$, $m \neq j$, $P_t/2 > P_L + p_m$, and, for contradiction, assume $P_L + p_m < (P_t - q_m)/2$. Then $\max(P_L, P_R) = P_R > (P_t + q_m)/2$. But

$$\max(L(m+1), R(m+1)) = L(m+1) = P_L + p_m + q_m < (P_t + q_m)/2.$$

However then $m+1$ would be the min-max root of t. $\square$

<u>Theorem 4.6</u> In a min-max tree $C \le H + 1 + \sum_{0 \le k \le n} q_k$. And the bound is tight.

<u>Proof</u> As in Theorem 4.4 we define

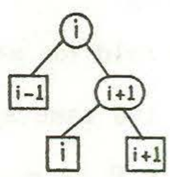
$$b_t = q_m \text{ for (A) and (C)}, \; b_t = p_m \text{ for (B) and (D)}.$$

Looking at $\sum b_t$, for each k, q_k can appear at most once in (A) and at most once in (C), while p_k can appear at most once in (B) or (D). This gives

$$\sum_{t \in S_T} b_t \le \sum_{1 \le k \le n} p_k + 2 \sum_{0 \le k \le n} q_k.$$

And $H \ge C - (1 + \sum_{0 \le k \le n} q_k)$.

If $\sum q_k = 0$, then the bound is tight, since the complete tree with 2^k-1 nodes, where $p_1 = p_3 = \ldots = p_n = 2^{1-k}$ and all the other p's and q's are 0, is a min-max tree and has $C = k$, $H = k-1$. To get the bound when $\sum q_k = Q$, we replace about $2^{k-1}Q$ of the nodes having non-zero probability with



where $q_i = 2^{1-k}$, and the others have zero probability. Then the entropy stays the same, and the cost increases by Q. $\square$

4.3 Weight Balanced Trees .

We can prove an upper bound for weight balanced trees which is similar to that for min-max trees. Using the same scheme as before:

Lemma 4.7 Let $\boxed{m}$ be the middle leaf of t with weight balanced root r. If m = r and m ≠ j then either

(A) $P_t/2 > P_L + p_m \geq (P_t - (q_m + p_{m+1}/2))/2$

or (B) $P_t/2 \geq P_R \geq (P_t - p_m)/2$.

If m = r = j then either

(C) $P_t/2 > P_L + p_m \geq (P_t - q_m)/2$

or (D) $P_t/2 \geq P_R \geq (P_t - p_m)/2$.

And the symmetric formulas hold for m+1 = r.

Proof (B), (C), (D) are from Lemma 4.3. For (A), weight balanced means

$$L(m+1) - R(m+1) \geq R(m) - L(m).$$

Collecting terms:

$$P(\boxed{i-1}, m) + q_m - (P_t - P(\boxed{i-1}, m) - q_m - p_{m+1}) - (P_t - P(\boxed{i-1}, m)) + P(\boxed{i-1}, m) - p_m \geq 0$$

or

$$4\ (P_L + p_m) \geq 2P_t - 2q_m - p_{m+1} + p_m$$

$$P_L + p_m \geq (P_t - (q_m + p_{m+1}/2))/2. \square$$

Theorem 4.8 In a weight balanced tree, $C \le H + 2$. And the bound is tight.

Proof As in Theorem 4.4 we have:

$$b_t = q_m + p_{m+1}/2 \text{ for (A)}, \; b_t = p_m \text{ for (B) and (D)},$$
$$b_t = q_m \text{ for (C)}.$$

In $\sum b_t$, for each k, q_k can appear at most once each in (A) and (C), while p_k can appear at most twice in (A) and at most once in (B) or (D). This gives

$$\sum b_t \le 2 \sum_{1\le k\le n} p_k + 2 \sum_{0\le k\le n} q_k = 2.$$

And $H \ge C - 2$.

The tree of Theorem 4.6 in which $Q = 1$ is weight balanced and has $C = k+1$, $H = k-1$. $\square$

This bound is equal to the min-max bound in the worst case, $Q = 1$. In fact, one easily proved consequence of the weight balanced definition is that if $Q = 1$, then the weight balanced tree is the same as the min-max tree. Since we have not shown that the bound of Theorem 4.8 can be achieved for all values of Q, we might conjecture that the bound can be lowered to the min-max bound for all Q. This is not the case. Namely, the tree T_A from before, with $p_1 = 2/3 - \epsilon$, $p_3 = 1/3 + \epsilon$ and all the others zero, has $C = 2$, $H \to \log 3 - 2/3$ as $\epsilon \to 0$. In this case $C - H$ is about 1.08.

5.0 The Expected Value of the Entropy

In this section we try to get some idea of what value we can expect for the entropy. It will be easiest to talk about entropy measured with natural logarithms. That is,

$$H_e(p, \ldots ,p_n) = \sum_{1\le k\le n} -p_k \ln p_k = H(p_1, \ldots ,p_n) \ln 2.$$

We are interested in knowing how large the entropy of a random probability distribution can be expected to be. To learn this, we derive an expression for the expected value of the entropy, given that all distributions are equally likely. Specifically, we show that the expected value of the entropy is

$$\bar{H}(p_1, \ldots ,p_n) = H_n - 1$$

where $H_n = \sum_{1\le k\le n} {}^1/_k$. Our first proof of this involved integrating the value of the entropy over all probability distributions, and then dividing the result by the volume of the region of integration. Ronald L. Rivest suggested the simple proof that appears here.

One integration formula that we need is:

Lemma 5.1 $\int_0^b x^m \ln x \, (b-x)^n \, dx$

$$= {}^{(n!\ m!\ b^{m+n+1})}/_{(m+n+1)!} \left(\ln b - \sum_{1\le i\le n+1} {}^1/_{(m+i)}\right).$$

Proof (Induction on n)

<u>Basis</u> If $n = 0$ then

$$\int_0^b x^m \ln x \, dx = x^{m+1} \left({}^{(\ln x)}/_{(m+1)} - {}^{1}/_{(m+1)^2} \right) \Big|_0^b$$

(CRC [12], integral 390, p. 334)

$$= {}^{b^{m+1}}/_{(m+1)} \left(\ln b - {}^{1}/_{(m+1)} \right).$$

<u>Induction</u> $(n > 0)$ Integrating by parts with

$$u = (b-x)^n, \quad du = -n(b-x)^{n-1} \, dx$$

$$v = x^{m+1} \left({}^{(\ln x)}/_{(m+1)} - {}^{1}/_{(m+1)^2} \right), \quad dv = x^m \ln x \, dx$$

we get

$$\int_0^b x^m \ln x \, (b-x)^n \, dx$$

$$= x^{m+1} \left({}^{(\ln x)}/_{(m+1)} - {}^{1}/_{(m+1)^2} \right) (b-x)^n \Big|_0^b +$$

$$\int_0^b x^{m+1} \left({}^{(\ln x)}/_{(m+1)} - {}^{1}/_{(m+1)^2} \right) n \, (b-x)^{n-1} \, dx$$

$$= {}^{n}/_{(m+1)} \int_0^b x^{m+1} \ln x \, (b-x)^{n-1} \, dx - {}^{n}/_{(m+1)^2} \int_0^b x^{m+1} \, (b-x)^{n-1} \, dx$$

$$= {}^{n}/_{(m+1)} \, {}^{((n-1)! \, (m+1)! \, b^{m+n+1})}/_{(m+n+1)!}$$

$$\left(\ln b - \sum_{1 \le i \le n} {}^{1}/_{(m+i+1)} \right) -$$

$${}^{n}/_{(m+1)^2} \, {}^{((n-1)! \, (m+1)! \, b^{m+n+1})}/_{(m+n+1)!}$$

(by induction and Gradshteyn [4], integral 3.191.1, p.284)

$$= {}^{(n! \, m! \, b^{m+n+1})}/_{(m+n+1)!} \left(\ln b - \sum_{1 \le i \le n+1} {}^{1}/_{(m+i)} \right). \square$$

The main theorem is:

<u>Theorem 5.2</u> $\bar{H}(p_1, \ldots, p_n) = H_n - 1.$

<u>Proof</u> For the uniform distribution over all n-tuples $(p_1, \ldots, p_n)$ such that $\sum p_k = 1$, the density function for p_k is $(n-1)(1-p_k)^{n-2}$. That is,

$$\text{Prob}(p_k \le x) = \int_0^x (n-1)(1-p_k)^{n-2}\, dp_k.$$

Using this density and summing over all k, we get

$$\bar{H} = n(n-1) \int_0^1 (1-p_k)^{n-2} (-p_k \ln p_k)\, dp_k.$$

Then if we apply Lemma 5.1, we get

$$\bar{H} = H_n - 1. \square$$

The consequences of this theorem are interesting. First, from Knuth ([6], p.74), we know that

$$H_n = \ln n + \gamma + O(n^{-1})$$

where $\gamma = 0.577\ldots$ is Euler's constant. From Gallager ([3] p. 23) we know that

$$H_e(p_1, \ldots, p_n) \le \ln n.$$

So the average entropy is always within 0.61 bits of the maximum possible entropy. This means that in a situation where the probability distribution is not known, the entropy is probably high.

6.0 Conclusions

To summarize, we have shown that weight balanced and min-max trees are near optimal by proving:

$$H - \log_2 H - (\log_2 e - 1) \le C_{Opt} \le \begin{cases} C_{WB} \le H + 2 \\ \\ C_{MM} \le H + 2. \end{cases}$$

As a result, we have

$$C_{WB} < C_{Opt} + \log H + 2.45$$
$$C_{MM} < C_{Opt} + \log H + 2.45.$$

These two bounds can probably be improved, either by improving the lower bound on C_{Opt}, or by trying a different approach, such as bounding C_{WB} and C_{MM} in terms of C_{Opt}. We conjecture that $C_{MM} \le C_{Opt}$ + constant is possible.

A number of other problems are open for research. One of these problems is the analysis of the average case, in the sense of what can be expected with a real application. One aspect of this analysis could be more empirical testing. An associated problem is that of comparing weight balanced and min-max trees, since only in the average could there be a strict relation between them. More generally, the question of the best scheme for choosing between the two generally balanced roots is open for research. A lower bound on the complexity of building the optimal tree would also be of interest.

References

[1] Bruno, J., and E. G. Coffman. "Nearly optimal binary search trees." Proc. IFIP Congress 71, North-Holland Publishing Co., Amsterdam, 1972, pp. 99-103.

[2] Fredman, M. L. "Two applications of a probabilistic search technique: sorting x + y and building balanced search trees." Proc. 7th Annual ACM Symp. on Theory of Computing, 1975, pp. 240-244.

[3] Gallager, R. G. Information Theory and Reliable Communication, Wiley, New York, 1968.

[4] Gradshteyn, I. S., and I. M. Ryzhik. Table of Integrals, Series, and Products, Academic Press, New York, 1965.

[5] Hu, T. C., and A. C. Tucker. "Optimum binary search trees." SIAM J. Applied Math. 21, 4, 1971, pp. 514-532.

[6] Knuth, D. E. The Art of Computer Programming, Volume 1: Fundamental Algorithms, Addison-Wesley, Reading, Mass., 1968.

[7] Knuth, D. E. "Optimum binary search trees." Acta Informatica 1, 1971, pp. 14-25.

[8] Knuth, D. E. The Art of Computer Programming, Volume 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

[9] Melhorn, K. "A note on binary search trees." Technical Report,

University of Saarbrucken, 1974.

[10] Nievergelt, J. "Binary search trees and file organization." Computing Surveys 6, 3 (Sept.), 1974, pp. 195-207.

[11] Rissanen, J. "Bounds for weight balanced trees." IBM Journal of Research and Development, March 1973, pp. 101-105.

[12] Selby, S. M. (ed.). Standard Mathematical Tables, The Chemical Rubber Co., 1965.

[13] Severance, D. G. "Identifier search mechanisms: a survey and generalized model." Computing Surveys 6, 3 (Sept.), 1974, pp. 175-194.

[14] Walker, W. A., and C. C. Gotlieb. "A top down algorithm for constructing nearly optimal lexicographic trees." R. C. Read (ed.), Graph Theory and Computing, Academic Press, New York, 1972, pp. 303-323.