Binary Search Trees:

Average and Worst Case Behavior*

Rainer Güttler, Kurt Mehlhorn, Wolfgang Schneider

Fachbereich 10 - Angewandte Mathematik und
Informatik der Universität des Saarlandes

D-6600 Saarbrücken

Abstract: We discuss several simple strategies for constructing binary search trees. Upper and lower bounds for the average and worst case search time in trees constructed according to these strategies are derived. Furthermore, different implementations are discussed and the results are applied to digital searching.

Key words: binary search, average behavior, worst case behavior, digital searching

*A preliminary version was presented at the GI-6. Jahrestagung, Stuttgart, Sept. 1976

I. Introduction and Survey of Known Results

"One of the popular methods for retrieving information by its 'name' is to store the names in a binary tree. We are given n names $B_1, B_2, \dots, B_n$ and 2n+1 frequencies $\beta_1, \dots, \beta_n$, $\alpha_o, \dots, \alpha_n$ with $\Sigma\beta_i + \Sigma\alpha_j = 1$. Here β_i is the frequency of encountering name B_i, and α_j is the frequency of encountering a name which lies between B_j and B_{j+1}, α_o and α_n have obvious interpretations" [Knuth 71] .
We may always assume w.l.o.g. that $\beta_i + \alpha_i + \beta_{i+1} \neq 0$ for all i. Otherwise, the i-th (or the (i+1)-th) key might as well be removed.

A binary search tree T is a tree with n interior nodes (nodes having two sons), which we denote by circles, and n+1 leaves, which we denote by squares. The interior nodes are labelled by the B_i in increasing order from left to right and the leaves are labelled by the intervals (B_j, B_{j+1}) in increasing order from left to right. Let b_i be the distance of interior node B_i from the root and let a_j be the distance of leaf (B_j, B_{j+1}) from the root. To retrieve a name X, b_i+1 comparisons are needed if $X = B_i$ and a_j comparisons are required if $B_j < X < B_{j+1}$.
Therefore we define the weighted path length of tree T as:

$$P = \sum_{i=1}^{n} \beta_i (b_i+1) + \sum_{j=o}^{n} \alpha_j \, a_j$$

D.E. Knuth [Knuth 71] gives an algorithm for constructing an optimum binary search tree, i.e. a tree with minimal weighted path length. His algorithm has $O(n^2)$ time complexity and $O(n^2)$ space complexity. Hu and Tucker [Hu & Tucker] consider the case that all names have frequency zero, i.e. $\beta_i = 0$ for $1 \leq i \leq n$. They give an algorithm with O(nlogn) time complexity for this case.

Approximation algorithms were considered early in the game [Bruno & Coffman, Walker & Gotlieb].

Walker and Gotlieb consider the following rule of thumb:

RULE I (Weight Balancing): Choose the root so as to equalize the weight of the left and right subtree as much as possible, then proceed similarly on the subtrees.

The weight of a subtree is the sum of the frequencies of all nodes and leaves in this subtree. They describe an implementation of this rule with time complexity O(nlogn) and space complexity O(n) and report that the rule tends to produce trees which are within a few percent of the optimum. (5%).

In order to define rule I more formally we need some more notation. For $1 \leq i \leq j \leq n$

$$w(i+1,i) = \alpha_i \quad \text{and}$$
$$w(i,j) = \alpha_{i-1} + \beta_i + \alpha_i + \ldots + \alpha_{j-1} + \beta_j + \alpha_j.$$

$w(i,j)$ is the weight of the tree with nodes $B_i,\ldots,B_j$ and leaves $(B_{i-1},B_i),\ldots,(B_j,B_{j+1})$. Rule I chooses as the root of the subtree with nodes $B_i,\ldots,B_j$ a node B_K, $i \leq k \leq j$, which minimizes

$$|w(i,k-1) - w(k+1,j)|$$

Ties are broken arbitrarily.

Recently several other rules of thumb were suggested.

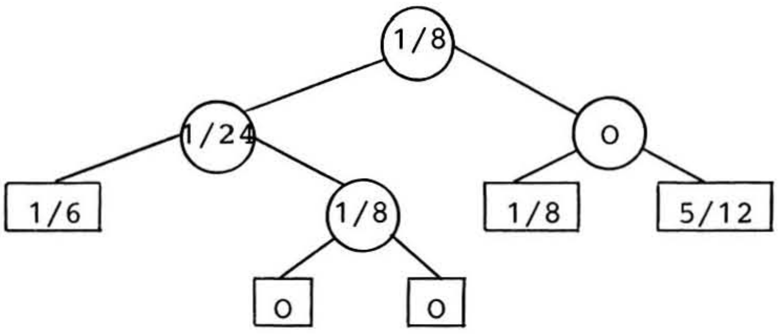
RULE II (Min-Max) [Bayer, Schnorr]: Choose the root so as to minimize the maximum of the weights of the left and right subtree, then proceed similarly on the subtrees.

More formally, rule II chooses as the root of the subtree with nodes $B_i,\ldots,B_j$ a node B_k, $i \leq k \leq j$, which minimizes

$$\max(w(i,k-1),\ w(k+1,j))$$

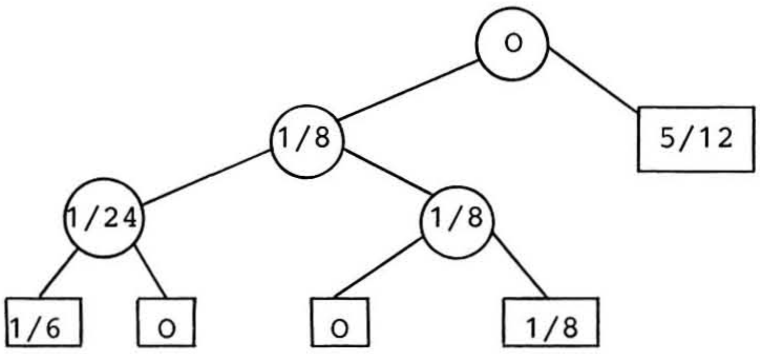
Again, ties are broken arbitrarily.

<u>Example:</u> Let $n=4$, $(\alpha_o, \beta_1, \ldots, \beta_4, \alpha_4)$ = $(1/6, 1/24, 0, 1/8, 0, 1/8, 1/8, 0, 5/12)$. The Min-Max tree is



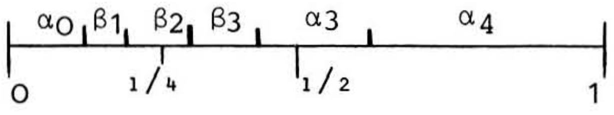
$P_{MM} = 2 = 48/24$

The weight balanced tree is



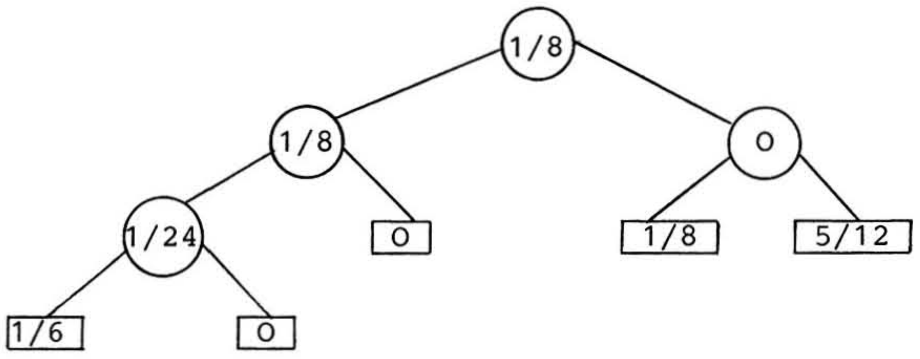
$P_{WB} = 49/24$

Both rules always choose nodes as the root which are close to the center of the distribution.



In our case the center of the distribution goes through the leaf with weight $\alpha_3 = 1/8$. The weight of the left subtree of the Min-Max tree is $1/6 + 1/24 + 1/8 = 1/3$. The center of the sub-distribution (1/6, 1/24, 0 1/8, 0) [strictly speaking this is not a distribution, since frequencies are not normalized] runs between the leaf with weight $\alpha_o = 1/6$ and the node with weight $\beta_1 = 1/24$. Therefore the node with

weight 1/24 is chosen as the root of the left subtree of the Min-Max tree.
We could instead look for a node, which is close to 1/4 of the entire distribution as root of the left subtree (close to 3/4 for the right subtree), close to 1/8, 3/8 (5/8, 7/8) in the subtrees of the next level and so on. In our example this strategy would choose the node with weight $\beta_2 = 1/8$ as the root of the left subtree.



$P_{BI} = 50/24$

This strategy may be formalized as

RULE III (BI-section) [Mehlhorn 77a]; we refer the reader to [Mehlhorn 77a] for an exact definition.

The first theoretical results about the behavior of binary search trees were obtained by Gilbert and Moore [Gilbert & Moore]. They consider the case that the weight is concentrated in the leaves, i.e. $\beta_i = 0$ for all i, and showed that in this case $H \leq P_{opt} \leq H+2$ where $H = \Sigma\beta_i \cdot \log 1/\beta_i + \Sigma\alpha_j \log 1/\alpha_j$ is the entropy of the frequency distribution.
Later Rissanen [Rissanen] showed $P_{MM} \leq H+2$ and Horibe [Horibe] showed $P_{MM} \leq H+2-(n+3)\cdot\alpha_{min}$, where $\alpha_{min} = \min_{0\leq j\leq n} \alpha_j$.

The general case was first considered by Mehlhorn [Mehlhorn 75] who proved $1/\log 3 \cdot H \leq P_{opt} \leq P_{WB} \leq 1.44\, H+2$. Bayer [Bayer] improved upon this. The best bounds presently known are:

$1/\log 3 \cdot H \leq P_{opt}$ [Mehlhorn 75]

$\max\{(H-d\Sigma\beta_i)/\log(2+2^{-d}); d \in \mathbb{R}\} \leq P_{opt}$ [Bayer]

$P_{WB} \leq H+2$ [Bayer]

$$P_{MM} \leq H + 1 + \Sigma\alpha_j \qquad \text{[Bayer]}$$

$$P_{BI} \leq \Sigma\beta_i \lfloor \log 1/\beta_i \rfloor + \Sigma\alpha_j \lfloor \log 1/\alpha_j \rfloor + 1 + \Sigma\alpha_j$$

$$\leq H + 1 + \Sigma\alpha_j \qquad \text{[Mehlhorn 77a]}$$

All bounds are achievable for a wide range of frequency distributions.

These results answer two important questions:

1) They give an a-priori test for the performance of binary search trees, i.e. they enclose the average path length in a narrow interval.

2) They prove that the approximation rules described above always produce nearly optimal search trees.

The importance of the approximation rules was increased by a recent result of Fredman [Fredman, Mehlhorn 77b]. He describes an implementation of the Min-Max, Weight-Balancing and Bisection Rule which runs in time O(n).

In sections II and III we study the Worst Case Behavior of search trees. We try to relate frequency and search time for every single node and leaf. In section II we derive upper bounds and in section III we derive lower bounds.

In section IV we propose another construction rule: the entropy rule. Its average and worst case behaviour are studied. Finally, in section V we apply our results to digital searching.

II. Worst Case Behavior: Upper Bounds

In the preceding section we surveyed results about the average case behavior of optimal and nearly optimal binary search trees. We pose the following question:

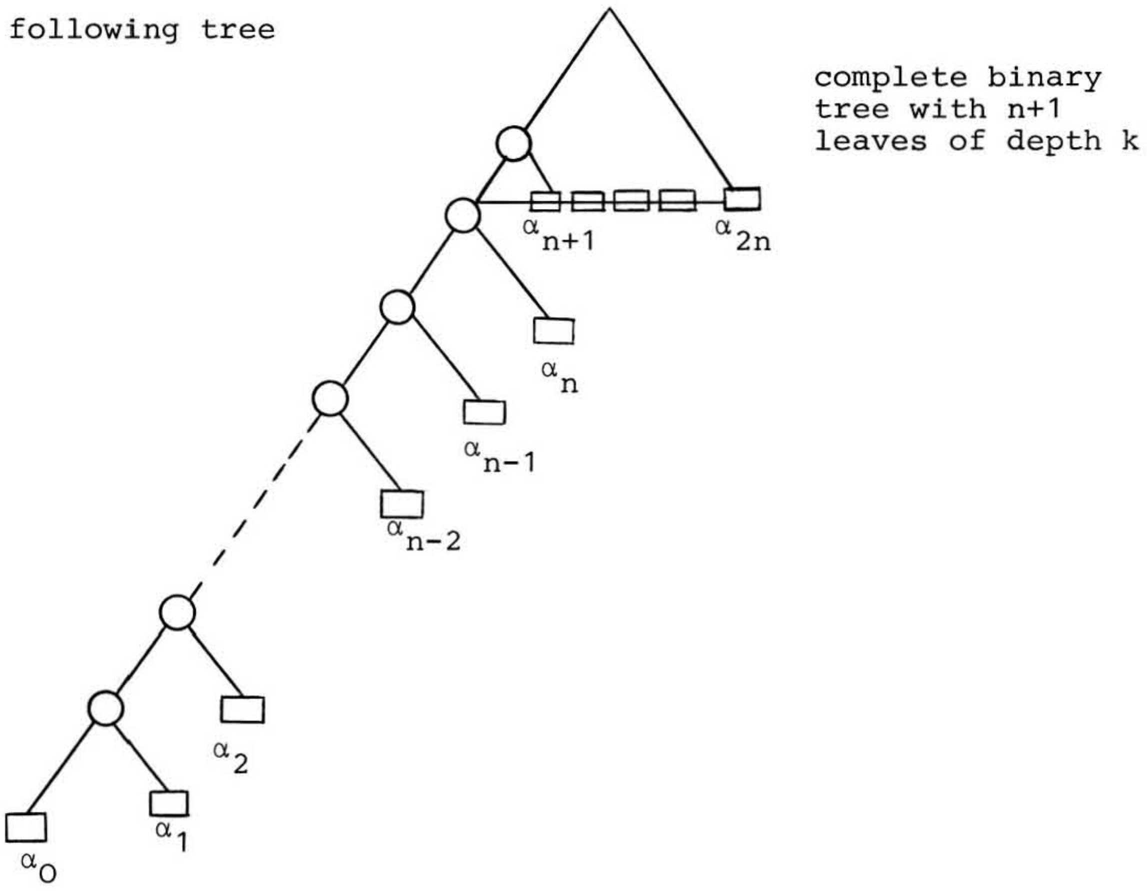
What can we say about the time needed for a single search (worst case behavior) ?

We give a simple example of a tree performing well on the average, but exhibiting extremely bad worst case behavior.

Consider $n = 2^k - 1$, $\beta_i = 0$ for all i, $\alpha_o = 2^{-k}$,

$\alpha_1 = \ldots = \alpha_n = \varepsilon$, $\alpha_{n+1} = \ldots = \alpha_{2n} = 2^{-k} - \varepsilon$ where ε is a small positive number. Then $H(\alpha_o,\ldots,\alpha_n) \approx \log(n+1)$.

The following tree



has weighted path length

$$P = n \cdot (2^{-j}-\varepsilon) \cdot k + \sum_{i=k+1}^{k+n} \varepsilon \cdot i + 2^{-k} \cdot (k+n)$$

$\approx \log(n+1) + 1$. The results of Gilbert and Moore tell us that $P_{opt} \geq H$, i.e. the tree is nearly optimal. However, a search ending in the left-most leaf takes n+logn steps, i.e. there is an exponential discrepancy between average and worst case behaviour: log n vs. n+log n.
This phenomenon does not occur in the trees constructed according to the rules decribed above.

<u>Theorem I</u> (Upper Bounds for the Worst Case Behaviour of Binary Search Trees) :

Let $(\alpha_o, \beta_1, \ldots, \beta_n, \alpha_n)$ be a frequency distribution and let b_i^{WB} (b_i^{MM}, b_i^{BI}) be the depth of node B_i in the tree constructed from that distribution according to the weight balancing (min-max, bisection) rule. Let furthermore b_i^{OPT} be the depth of node B_i in an optimal tree ($\hat{=}$ smallest weighted path length).
a_j^{WB}, a_j^{MM}, a_j^{BI} and a_j^{OPT} are defined analogously. Let $\delta = (1 + \sqrt{5})/2$ and $\varphi = 2/(\sqrt{17} -3)$. Then $(\log \delta)^{-1} \approx 1.44$ and $(\log \varphi)^{-1} \approx 1.20$.

<u>Case 1:</u> $\Sigma\alpha_j = 0$, i.e. all α_j are 0.

(1.1) $b_i^{WB} \leq (\log \varphi)^{-1} \log 1/\beta_i + 0.412$

(1.2) $b_i^{MM} \leq \log 1/\beta_i$

(1.3) $b_i^{BI} \leq \log 1/\beta_i$

(1.4) $b_i^{OPT} \leq$ open, but compare the remark following the proof.

<u>Case 2:</u> $\Sigma\beta_i = 0$, i.e. all β_i are 0

(2.1) $a_j^{WB} \leq (\log \delta)^{-1} \log 1/\alpha_j + 2$

2.2) $a_j^{MM} \leq (\log \delta)^{-1} \log 1/\alpha_j + 2$

(2.3) $a_j^{BI} \leq \log 1/\alpha_j + 2$

(2.4) $a_j^{OPT} \leq (\log \delta)^{-1} \log 1/\alpha_j + 2$

<u>Case 3:</u> $\Sigma\alpha_j \geq 0,\ \Sigma\beta_i \geq 0$

(3.1) $b_i^{WB} \leq (\log \delta)^{-1} \log 1/\beta_i + 0.157$

$a_j^{WB} \leq (\log \delta)^{-1} \log 1/\alpha_j + 2$

(3.2) $b_i^{MM} \leq (\log \delta)^{-1} \log 1/\alpha_i$

$a_j^{MM} \leq (\log \delta)^{-1} \log 1/\alpha_j + 2$

(3.3) $b_i^{BI} \leq \log 1/\beta_i$

$a_j^{BI} \leq \log 1/\alpha_j + 2$

(3.4) $b_i^{OPT} \leq (\log \delta)^{-1} \log 1/\beta_i$

$a_j^{OPT} \leq (\log \delta)^{-1} \log 1/\alpha_j + 2$

All bounds are achievable.

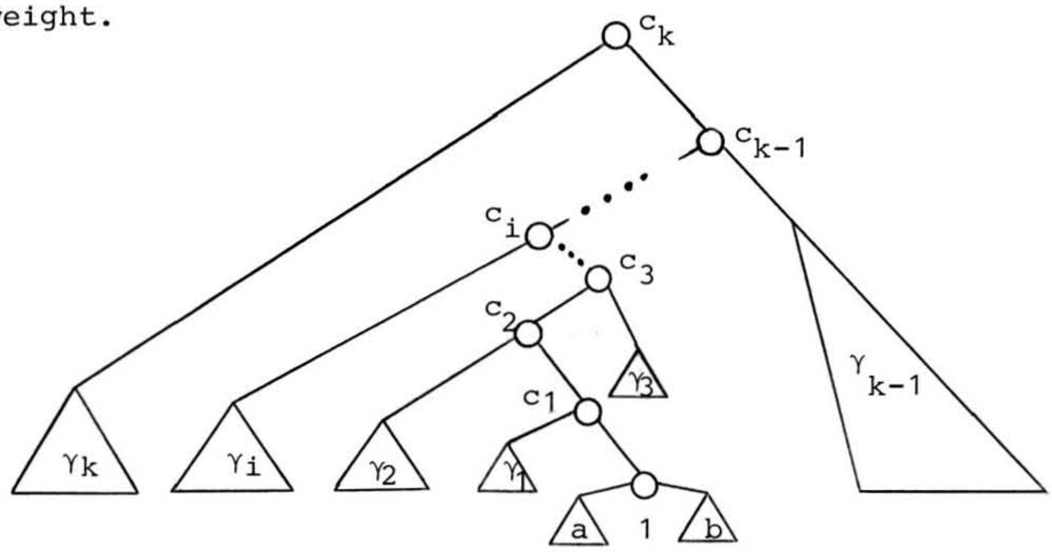
<u>Proof:</u> The bisection rule (cases (1.3),(2.3),(3.3)) was considered elsewhere [Mehlhorn 77a].

For this proof it is convenient to work with unnormalized frequencies. Let p_i, $1 \leq i \leq n$, and q_j, $0 \leq j \leq n$, be non-negative real numbers. We refer to p_i as the weight of node B_i and to q_j as the weight of leaf (B_j, B_{j+1}). By normalizing we obtain a frequency distribution

$$\beta_i = p_i/W \qquad \text{and} \qquad \alpha_j = q_j/W$$

where $W = \Sigma p_i + \Sigma q_j$.

Case 1.1: We want to compute the minimal weight of a tree which is constructed according to the weight balancing rule and features a node v of weight 1 at depth k. Let T_k^{WB} be such a tree of minimal weight and let W_k be its weight.

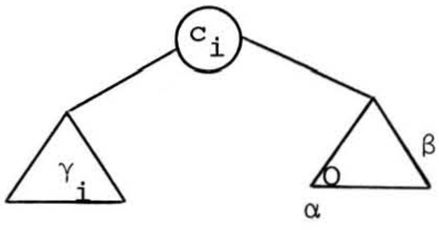


Consider the path form from node v with weight 1 to the root. Let c_i be the weight of the i-th node on this path ($1 \le i \le k$) and let γ_i be the weight of that subtree of the node with weight c_i which does not contain v. Let a (b) be the weight of the left (right) subtree of node v and let $\gamma_o = a+b$, $c_o = 1$.

Lemma A: For $i \geq 1$

(A) $$2\gamma_i + c_i \geq 2 \sum_{j=0}^{i-1} (c_j+\gamma_j) - \max_{0\le j\le i-1} (c_j,\gamma_j)$$

Proof: Consider the subtree with root $\boxed{c_i}$;



We may assume w.l.o.g. that v lies in the right subtree. Let α be the weight of the left-most node of positive weight in the right subtree and let β be the weight of the remainder of this subtree.
Then

$$\alpha + \beta = \sum_{j=0}^{i-1} (c_j + \gamma_j)$$

Since the node with weight α was not chosen as the root

$$|\alpha + \beta - \gamma_i| \leq |\gamma_i + c_i - \beta|$$

and hence

$$\alpha + \beta - \gamma_i \leq |\gamma_i + c_i - \beta|$$

Since

$$\alpha + \beta - \gamma_i \leq \beta - \gamma_i - c_i$$

leads to the contradiction $\alpha \leq - c_i \leq 0$
we have

$$\alpha + \beta - \gamma_i \leq \gamma_i + c_i - \beta$$

or

$$2\gamma_i + c_i \geq 2(\alpha+\beta) - \alpha$$

The node α is either one of the c_j's or it is an element of some γ_j. Hence

$$2\gamma_i + c_i \geq 2 \sum_{j=0}^{i-1} (c_i + \gamma_i) - \max_{0\leq j\leq i-1} (c_j,\gamma_j) \qquad \square$$

We want to solve the set A of recurrence relations.Let

(B) $$\delta_0 = 1$$
$$\delta_1 = \delta_0 - 1/2\delta_0 = 1/2$$
$$\delta_2 = \delta_0 + \delta_1 - 1/2\delta_0 = 1$$
$$\delta_i = \sum_{j=0}^{i-1} \delta_j - \delta_{i-1}/2 \qquad \text{for } i \geq 3$$

<u>Lemma C:</u> For $i \geq 0$

(C) $\gamma_i + c_i \geq \delta_i$

<u>Proof:</u> (by induction on i). The claim is obvious for $i = 0$. (Remember $c_0 = 1$). For $i > 0$:

$$\gamma_i + c_i \geq \gamma_i + c_i/2$$

$$\overset{(A)}{\geq} \sum_{j \neq \ell}^{i-1} (c_j + \gamma_j) + (c_\ell + \gamma_\ell)/2$$

where

$$c_\ell + \gamma_\ell = \max_{0 \leq j \leq i-1} (c_j + \gamma_j)$$

$$\geq \max_{0 \leq j \leq i-1} \delta_j = \begin{cases} \delta_0 & i=1 \\ \delta_0 & i=2 \\ \delta_{i-1} & i \geq 3 \end{cases}$$

Hence

$$\gamma_i + c_i \geq \sum_{j=0}^{i-1} \delta_j - \max_{0 \leq j \leq i-1} \delta_j/2$$

$$= \delta_i \qquad \square$$

Consider the following distribution of weights, $n = 2k+1$

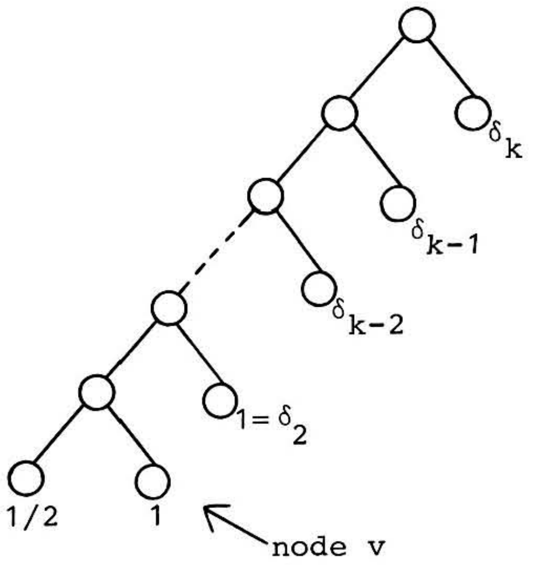
(D) $p_1 = \delta_1 = 1/2$

$p_3 = \delta_0 = 1$

$p_{2i+1} = \delta_i$ for $2 \leq i \leq k$

$p_{2i} = 0$ for $1 \leq i \leq k$

then the following tree can be constructed by the weight-balancing rule.



Proof: (by induction on k). The claim is obvious for k=0, k=1 and k=2. Assume k > 2. The weight difference between the left and right subtree of the tree drawn is

$\delta_k - \sum_{i=0}^{k-1} \delta_i = - \delta_{k-1}/2$. If one chooses the node with weight δ_k as the root then the difference is $\sum_{i=0}^{k-1} \delta_i \geq \delta_k \geq \delta_{k-1}/2$.
If one chooses the node with weight δ_{k-1} as the root then the difference is $\delta_k - \sum_{i=0}^{k-2} \delta_i = \delta_{k-1}/2$. Hence the tree drawn can be constructed by the weight-balancing rule. □

$W_K = \sum_{i=0}^{k} (c_i + \gamma_i)$ is the weight of T_k^{WB}. From Lemma C we infer $W_k \geq \sum_{i=0}^{k} \delta_i$ and from Lemma D we infer $W_k \leq \sum_{i=0}^{k} \delta_i$.
Hence

$$W_k = \sum_{i=0}^{k} \delta_i$$

It remains to compute the W_k's.

Lemma E: The W_k's satisfy the following recurrence equation:

$$W_0 = 1$$
$$W_1 = 3/2$$
$$W_2 = 5/2$$
$$W_k = 3/2\ W_{k-1} + 1/2\ W_{k-2} \qquad \text{for } k \geq 3.$$

<u>Proof:</u> simple calculation $\square$

Let $W(z) = \sum_{k \geq 0} W_k \cdot z^k$ be the generating function of the sequence $\{W_k\}_{k \geq 0}$. Then

$$W(z) \cdot (1-3z/2-z^2/2) = 1-z^2/4$$

and hence

$$W(z) = \frac{1 - z^2/4}{1-3z/2-z^2/2}$$

The roots of the denominator are

$$z_1 = (\sqrt{17}-3)/2 \qquad z_2 = (-\sqrt{17}-3)/2 \quad ,$$

the partial fraction expansion of $W(z)$ is

$$W(z) = \frac{\rho_1}{z_1-z} + \frac{\rho_2}{z_2-z}$$

with

$$\rho_1 = \frac{-5+3\sqrt{17}}{4\sqrt{17}}$$

$$\rho_2 = \frac{5+3\sqrt{17}}{4\sqrt{17}}$$

Then using $\frac{1}{z_1-z} = 1/z_1 \cdot \sum_{i \geq 0} (\frac{z}{z_1})^i$ we get

$$W_k = \frac{\rho_1}{z_1^{k+1}} + \frac{\rho_2}{z_2^{k+1}}$$

$$= z_1^{-(k+1)} (\rho_1 + \rho_2 (z_1/z_2)^{k+1})$$

Node v of weight 1 has depth k in tree T_k^{WB}. The weight of T_k^{WB} is W_k and this is minimal. Node v has relative weight $1/W_k$. Consider now any node B_i of depth b_i and frequency β_i in a tree constructed according to the weight balancing rule. Then

$$\beta_i \le 1/W_{b_i}$$

and hence

$$\log 1/\beta_i \ge \log W_{b_i}$$

$$= (b_i+1) \log 1/z_1 + \log(\rho_1+\rho_2(z_1/z_2)^{b_i+1})$$

Thus

$$b_i \le (\log 1/z_1)^{-1}[\log 1/\beta_i - \log(\rho_1+\rho_2(z_1/z_2)^{b_i+1})]-1$$

$$= (\log \varphi)^{-1} \log 1/\beta_i + d_{b_i}$$

where

$$\varphi = 1/z_1 = 2/(\sqrt{17}-3)$$

and

$$d_k = (\log \varphi)^{-1} [-\log(\rho_1+\rho_2(z_1/z_2)^{k+1})]-1$$

The sequence d_k converges to

$$-(\log \varphi)^{-1} \cdot \log \rho_1 - 1 \approx 0.396 ,$$

it reaches its maximal value for k = 2

$$d_2 \approx 0.412$$

<u>Case 1.2:</u> This case is simple. The Min-Max rule chooses as the root of the subtree with nodes $B_i,\dots,B_j$ a node B_k, $i \le k \le j$, which minimizes $\max(w(i,k-1),w(k+1,j))$. Apparently, it is always possible to choose B_k such that

$$\max(w(i,k-1),w(k+1,j)) \le w(i,j)/2$$

Consider any node B_i of depth b_i and frequency β_i in a tree

constructed according to the min-max rule.
Then by the above

$$\beta_i \leq 2^{-b_i}$$

and hence

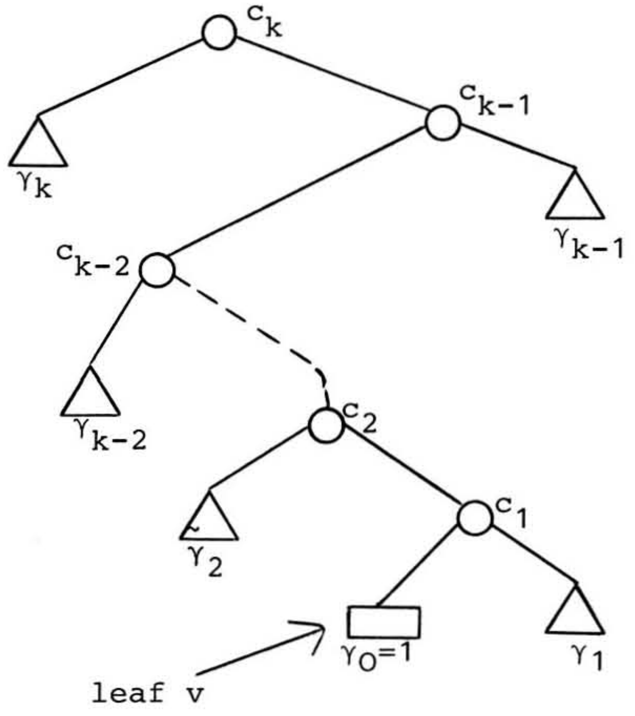
$$b_i \leq \log 1/\beta_i$$

The example $n = 2^k - 1$

$$\beta_i = \begin{cases} 0 & \text{for i even} \\ 2^{-k+1} & \text{for i odd} \end{cases}$$

shows that the bound is achievable.

<u>Case 2.1:</u> The proof is very similar to the proof of case (1.1). Let T_k^{WB} be a tree of minimal weight featuring a leaf of weight 1 at depth k. Let W_k be the weight of T_K^{WB}



We only state the main equations in the proof and leave the details to the reader.

Lemma A: For $i \geq 1$

(A) $$\gamma_i \geq \sum_{j=0}^{i-1} \gamma_j - \max_{0 \leq j \leq i-1} \gamma_j$$

Proof: As in case (1.1), but take α to be the weight of the left-most leaf of positive weight in the left subtree. □

Let

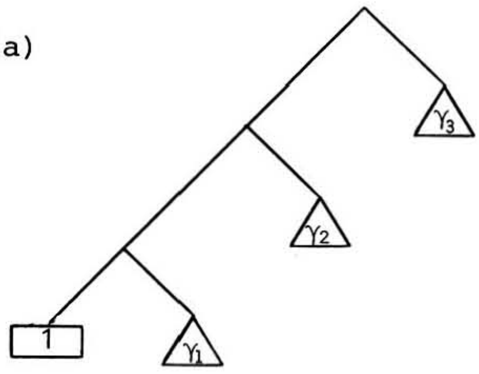
(B) $$\begin{aligned} \delta_0 &= 1 \\ \delta_1 &= 0 \\ \delta_2 &= 0 \\ \delta_3 &= 1 \\ \delta_i &= \sum_{j=0}^{i-2} \delta_j \qquad \text{for } i \geq 4 \end{aligned}$$

Lemma C: For $i \geq 0$

(C) $$\gamma_i \geq \delta_i$$

Proof: The claim is obvious for $i = 0,1,2$. For $i \geq 4$ the proof is a simple induction. It remains to consider the case $i = 3$. Up to symmetry there are four possible trees. Note that $\gamma_1, \gamma_2 > 0$ by our basic assumption that $\beta_i+\alpha_i+\beta_{i+1} \neq 0$ for all i.

(a)



Since it was possible to construct this tree we must have

$|1+\gamma_1-\gamma_2-\gamma_3| \geq 1+\gamma_1+\gamma_2-\gamma_3$

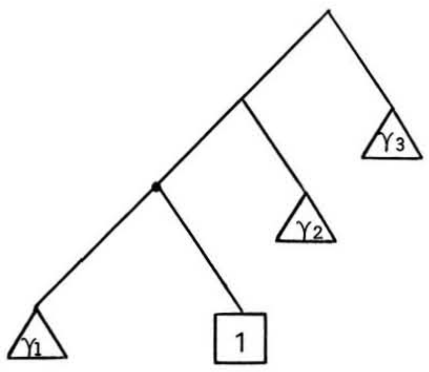
and hence

$\gamma_2+\gamma_3-1-\gamma_1 \geq 1+\gamma_1+\gamma_2-\gamma_3$

Thus

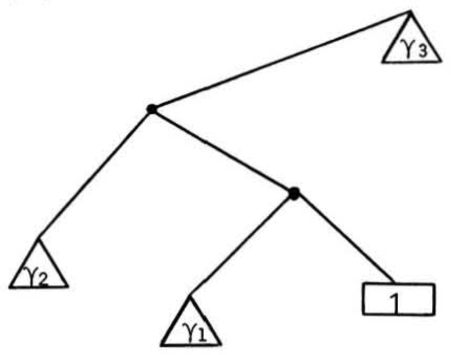
$\gamma_3 \geq 1 + \gamma_1 \geq 1$

(b)



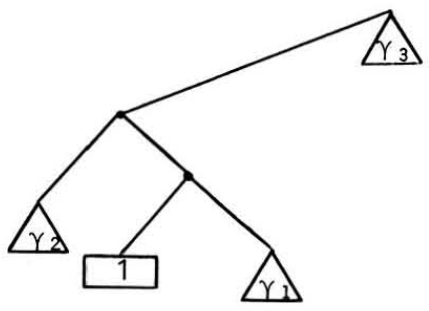
As in (a) one shows
$\gamma_3 \geq 1 + \gamma_1 \geq 1$

(c)



Since the left subtree was constructed according the weight-balancing rule we have
$1+\gamma_1-\gamma_2 \leq |\gamma_2+\gamma_1 - 1|$
and hence
$1+\gamma_1-\gamma_2 \leq \gamma_2 +\gamma_1 - 1$
Thus
$$\gamma_2 \geq 1$$
and by (A)
$$\gamma_3 \geq 1$$

(d)



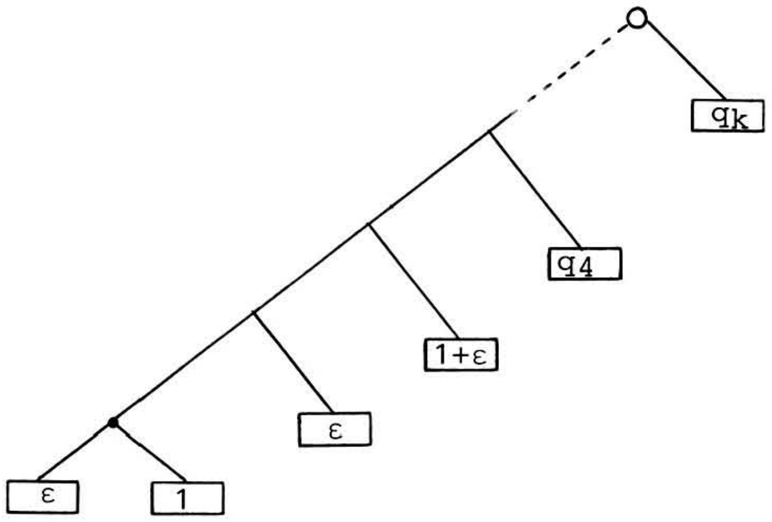
As in (a) one shows
$\gamma_3 \geq 1 + \gamma_2 \geq 1$

□

<u>Lemma D:</u> Consider the following distribution of weights,

$$q_o = \varepsilon$$
$$q_1 = 1$$
$$q_2 = \varepsilon$$
$$q_3 = 1 + \varepsilon$$
$$q_i = \sum_{j=0}^{i-2} q_j \qquad \text{for } i \geq 4$$

where ε is an arbitrary positive real. Then the following tree can be constructed by the weight-balancing rule.



□

Proof: As in case (1.1).

$W_k = \sum_{i=0}^{k} \gamma_i$ is the weight of T_k^{WB}. From Lemma C we infer $W_k \geq \sum_{i=0}^{k} \delta_i$ and from Lemma D we infer $W_k \leq \sum_{i=0}^{k} \delta_i + \varepsilon$ for any positive real ε. It remains to compute the sums $\Delta_k = \sum_{i=0}^{k} \delta_i$.

From now on one proceeds as in case (1.1) and obtains

$$\Delta_k = z_1^{-(k+1)} (\rho_1 + \rho_2 (z_1/z_2)^{k+1})$$

where

$$z_1 = (-1+\sqrt{5})/2 \qquad z_2 = (-1-\sqrt{5})/2$$

$$\rho_1 = (5-\sqrt{5})/10 \qquad \rho_2 = (5+\sqrt{5})/10$$

Hence

$$a_j \leq (\log \delta)^{-1} \log 1/\alpha_j + d_{a_j}$$

where

$$\delta = 1/z_1 = 2/(\sqrt{5} - 1) \quad \text{and}$$

$$d_k = (\log \delta)^{-1} [-\log(\rho_1+\rho_2(z_1/z_2)^{k+1})] - 1$$

The sequence $\{d_k\}_{k\geq 0}$ converges to $-(\log \delta)^{-1} \log \rho_1 - 1 \approx 1.67$.

It assumes it maximal value 2 for $k = 2$.

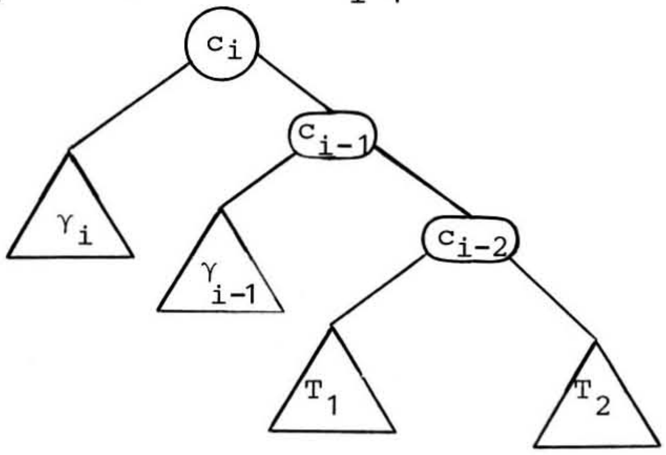
Case 2.2: The Min-Max rule and the weight-balancing rule are identical in the case that $\Sigma\beta_i = 0$. Hence the analysis of case 2.1 applies.

Case 2.4: Let T_k^{OPT} be an optimal (with respect to average search time) tree featuring a leaf v of weight 1 at depth k. Let W_k be the weight of T_k^{OPT}. Otherwise, we use the notation of case 2.1. Then $W_k = \sum_{i=0}^{k} \gamma_i$.

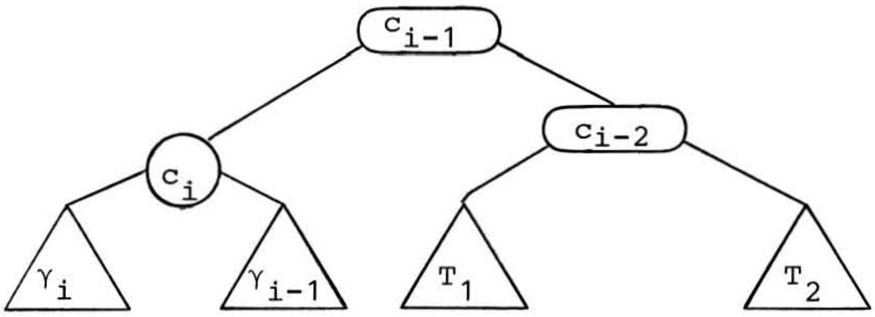
Lemma A: For $i \geq 3$: $\gamma_i \geq W_{i-2}$.

Proof: Consider the subtree with root c_i. We may assume w.l.o.g. that c_{i-1} is the right son of c_i. c_{i-2} is either right or left son of c_{i-1}.

a) c_{i-2} is right son of c_{i-1}

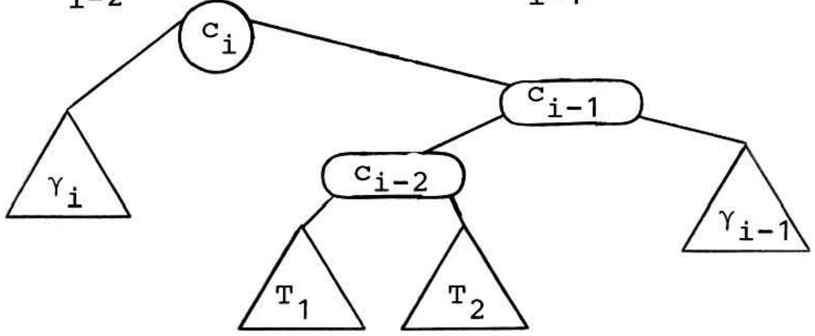


This tree must have no larger weighted path length than the following

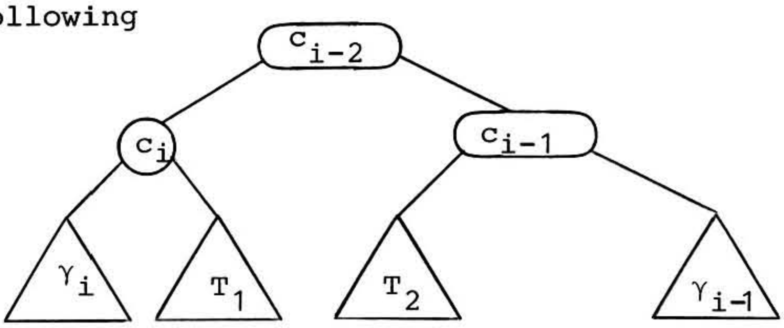


Hence $W_{i-2} \leq \gamma_i$ (Note that the c_j are 0)

b) c_{i-2} is the left son of c_{i-1}



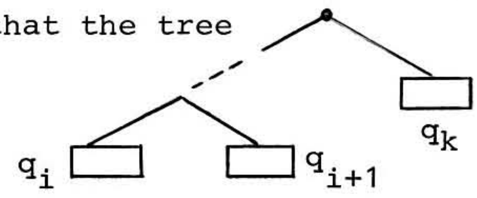
This tree must not have larger weighted path-length than the following



Hence $W_{i-2} \leq \gamma_i$.

It is also easy to show that $\gamma_2 \geq \gamma_1$ (by case analysis) and that $\gamma_1 > 0$ (by definition).

From now on the proof proceeds exactly as in case 2.1 except for the proof of Lemma D. There one shows by a very simple inductive argument that the tree



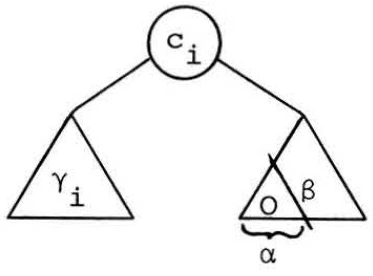
is optimal for the weights $q_i, q_{i+1}, \ldots, q_k$.

<u>Case 3.1:</u> We first derive the bound for b_i^{WB}. We use the notation of case 1.1 .

<u>Lemma A:</u> For $i \geq 1$

$$\text{(A)}\quad 2\gamma_i + c_i \geq 2\sum_{j=0}^{i-1}(c_j+\gamma_j) - \max_{0\leq j\leq i-1}(c_j+2\gamma_j)$$

<u>Proof:</u> Consider the subtree with root c_i.



We may assume w.l.o.g. that v lies in the right subtree. Consider the left-most node in the right subtree such that either the node itself or a leaf to its left which still belongs to the right subtree has positive weight. Let α' be the weight of that node, let α'' be the total weight of the leaves to the left of that node and let β be the weight of the remainder of the right subtree. Let $\alpha = \alpha' + \alpha'' > 0$. Since the node with weight α' was not choosen as the root

$$\alpha + \beta - \gamma_i \leq \gamma_i + c_i + \alpha'' - \beta$$

and hence

$$2\gamma_i + c_i \geq 2(\alpha+\beta) - \alpha' - 2\cdot\alpha''$$

α'' is by definition the weight of a single leaf. This leaf belongs to one of the γ_j's. Hence α' either also belongs to γ_j or is c_j. In either case

$$\alpha' + 2\alpha'' \leq \max_{0\leq j\leq i-1} (c_j+2\gamma_j)$$ □

Let (B)
$$\delta_0 = 1$$
$$\delta_1 = 1/2$$
$$\delta_2 = 1$$
$$\delta_i = \sum_{j=0}^{i-2} \delta_j \qquad \text{for } i \geq 3$$

Lemma C: For $i \geq 0$

(C) $\gamma_i + c_i \geq \delta_i$

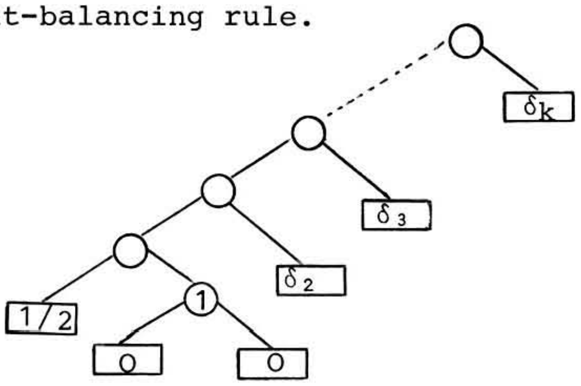
Proof: Obvious for $i = 0$, by case analysis for $i = 1$ and $i = 2$ and by induction for $i \geq 3$. □

Lemma D: Consider the following distribution of weights, $n = k+1$

$$q_0 = 1/2 = \delta_1,\ q_1 = q_2 = 0,\ q_i = \delta_{i-1} \text{ for } B \leq i \leq k+1$$

$$p_1 = 0,\ p_1 = 1,\ p_i = 0 \quad \text{for } 2 \leq i \leq k+1$$

Then the following tree may be constructed according to the weight-balancing rule.



Proof: Obvious □

$W_k = \sum_{i=0}^{k} (c_i + \gamma_i)$ is the weight of T_k^{WB}. As above, we infer $W_k = \sum_{i=0}^{k} \delta_i$. From now on one proceeds as in case 1.1 and obtains:

$$W_k = z_1^{-(k+1)} \; (\varrho_1 + \varrho_2 (z_1/z_2)^{k+1})$$

where $z_1 = (-1 + \sqrt{5})/2 \qquad z_2 = (-1 - \sqrt{5})/2$

$$\varrho_1 = (3\sqrt{5} + 5)/20 \qquad \varrho_2 = (-3\sqrt{5} + 5)/20$$

Hence
$$b_i \leq (\log \delta)^{-1} \log 1/\beta_i + d_{b_i}$$
where $d_k = (\log \delta)^{-1} \; [-\log(\varrho_1 + \varrho_2 (z_1/z_2)^{k+1})]-1$

The sequence $\{d_k\}_{k \geq 0}$ converges to $-(\log \delta)^{-1} \cdot \log \varrho_1 - 1 \approx 0.1125$. It assumes its maximal value 0.157 for k = 1.

Next we derive a bound on a_j^{WB}. We use the notation of case 2.1 .

Lemma A: For $i \geq 2$

(A) $\quad 2\gamma_i + c_i \geq 2 \cdot \sum_{j=0}^{i-1} (c_j+\gamma_j) - \max_{0 \leq j \leq i \cdot 1} (c_j+2\gamma_j)$

Proof: As above where the bound on b_i^{WB} was derived. Note however, that the argument cannot be applied to the subtree with root c_1 since the subtree which contains γ_o does not contain a node.

From now on the proof proceeds exactly as in case 2.1 .

Case 3.2: We only show how to derive the bound on b_i^{MM}, the bound on a_j^{MM} being similar. We use the notation of case 1.1.

Lemma A: For $i \geq 1$

(A) $$\gamma_i + c_i \geq \sum_{j=0}^{i-1} (c_j + \gamma_j) - \max_{0 \leq j \leq i-1} \gamma_j$$

Proof: Define α' and α'' as in case 3.1 . Then $\alpha' + \alpha'' = \alpha > 0$.

$$\alpha + \beta \leq \max (\gamma_i + c_i + \alpha'', \beta)$$

and hence

$$\alpha + \beta - \alpha'' \leq \gamma_i + c_i$$

α'' is by definition the weight of a single leaf. This leaf belongs to one of the γ_j's.

Hence $\alpha'' \leq \max_{0 \leq j \leq i-1} \gamma_j$

Let (B) $$\delta_0 = 1$$
$$\delta_1 = 1$$
$$\delta_2 = 2$$
$$\delta_i = \sum_{i=0}^{k-2} \delta_i \qquad \text{for } i \geq 2 .$$

From now on one proceeds exactly as in case 3.1 . One obtains

$$W_k = z_1^{-(k+1)} (\rho_1 + \rho_2 (z_1/z_2)^{k+1})$$

where $z_1 = (-1 + \sqrt{5})/2 \qquad z_2 = (-1-\sqrt{5})/2$

$\rho_1 = 2/\sqrt{5} \qquad \rho_2 = -2/\sqrt{5}$

Hence $b_i \leq (\log \delta)^{-1} \log 1/\beta_i + d_{b_i}$

where $d_k = (\log \delta)^{-1} [-\log(\rho_1 + \rho_2 (z_1/z_2)^{k+1})]-1$

The sequence d_k converges to

$-(\log \delta)^{-1} \log \rho_1 - 1 \approx -0.768.$

It assumes its maximal value for $k = 0$, $d_o = 0$.

Case 3.4:

We only derive a bound on b_i^{OPT}, the bound on a_j^{OPT} is derived similarly. Let T_k^{OPT} be an optimal tree featuring a node of weight 1 at depth k. Let W_k be the weight of T_k^{OPT}. Otherwise, we use the notation of case 1.1.

Then $W_k = \sum_{i=0}^{k} (c_i + \gamma_i)$.

Lemma A: For $i \geq 2$: $c_i + \gamma_i \geq W_{i-2}$

Proof: Analogous to the proof of Lemma a in case 2.4. We refer to that proof. In case a) of that proof

$$W_{i-2} + c_{i-1} \leq \gamma_i + c_i ,$$

in case b) of that proof

$$W_{i-2} + c_{i-2} \leq \gamma_i + c_i.$$

Note that the inequality is valid for $i \geq 2$ since $\textcircled{c_0}$ exists. □

It is also easy to show (by case analysis) that

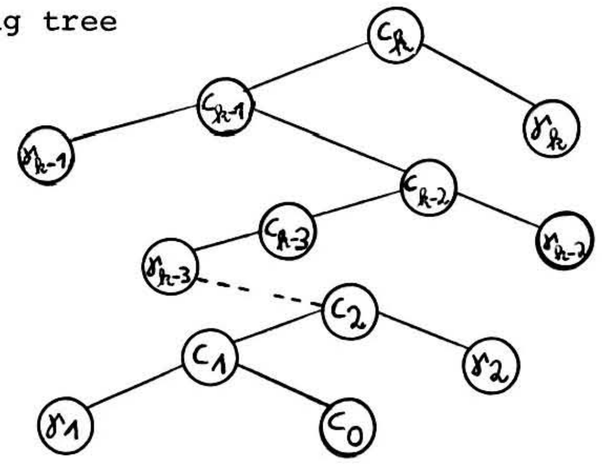
$$c_1 + \gamma_1 \geq c_0 = 1.$$

From now one proceeds exactly as in case 3.2 . □

Theorem 1 gives precise information about the worst case behavior of weight-balanced, Min-Max, Bisection and optimal trees. With respect to optimal trees the case $\Sigma\alpha_j = 0$ was left open. We can give only a partial answer in that case. Of course, case 3.4 applies and hence

$$b_i^{OPT} \leq (\log \delta)^{-1} \log 1/\beta_i$$

However, we do not know if this bound is achievable. We are only able to show that the multiplicative constant is at least 1.29. Thus the worst case behaviour of optimal trees is worse than that of nearly optimal trees. Consider the following zig-zag tree



with $\gamma_i = c_i$, $W_k = c_o + 2 \cdot \sum_{i=1}^{k} c_i$, $c_o = 1$,

$c_1 = 1/2$, $c_2 = 1$ and

$$2c_i = c_{i-2} + W_{i-2} \quad \text{for } i \geq 3.$$

Then the tree above is optimal. (The proof is a tedious case analysis).
Since $2c_i = W_i - W_{i-1}$ for $i \geq 1$

$$W_i - W_{i-1} = 1/2(W_{i-2} - W_{i-3}) + W_{i-2}$$

for $i \geq 3$. With $W(z) = \sum_{i \geq 0} W_i z^i$

$$W(z) = \frac{W_o + (W_1 - W_o)z + (W_2 - W_1 - 3/2W_o)z^2}{1 - z - 3/2z^2 + 1/2z^3}$$

$$= \frac{2 + 2z + z^2}{(z-z_1)(z-z_2)(z-z_3)}$$

where $z_1 = -1$, $z_2 = 2+\sqrt{2}$, $z_3 = 2-\sqrt{2}$. Hence

$$W_k \approx c \cdot (1/z_3)^k \approx c \cdot 1{,}7^k$$

for some constant c and large k. This shows that whenever

$$b_i^{OPT} \leq c_1 \cdot \log 1/\beta_i + c_2$$

holds for all optimal trees then $c_1 \geq (\log 1/1.7)^{-1} \approx 1.29$

III. Lower Bounds

In this section we derive lower bounds on the search time in binary trees. We prove a lower bound on the time required for a single search and give an alternate proof for Bayer's bound on the average search time. Our proofs are based on a well-known proof for the noise-less coding theorem [Kameda & Weihrauch].

For this section $(\alpha_0, \beta_1, \alpha_1, \ldots, \alpha_{n-1}, \beta_n, \alpha_n)$ is a fixed probability distribution and T is a search tree for this distribution. As before, b_i is the depth of node B_i and a_j is the depth of leaf (B_j, B_{j+1}).

<u>Lemma:</u> Let $c \in R$, $0 \le c < 1$, and

$$\bar\beta_i = ((1-c)/2)^{b_i} \cdot c , \qquad 1 \le i \le n$$

$$\bar\alpha_j = ((1-c)/2)^{a_j} , \qquad 0 \le j \le n$$

Then $\bar\beta_i, \bar\alpha_j \ge 0$ and $\Sigma\bar\beta_i + \Sigma\bar\alpha_j = 1$, i.e. $(\bar\alpha_0, \bar\beta_1, \bar\alpha_1, \ldots, \bar\beta_n, \bar\alpha_n)$ is a probability distribution.

<u>Proof:</u> A simple induction on n. □

<u>Theorem II (Bayer)</u> (A lower bound for the average search time):
Let $B = \Sigma\beta_i$, $P = \Sigma\beta_i(b_i+1) + \Sigma\alpha_j \cdot a_j$ and
$H = \Sigma\beta_i \cdot \log 1/\beta_i + \Sigma\alpha_j \log 1/\alpha_j$. Then

$$\max\{(H-dB)/\log(2+2^{-d});\ d \in \mathbb{R}\} \le P$$

<u>Proof:</u> Let $\bar\beta_i$ and $\bar\alpha_j$ be defined as in the lemma above. Then

$$b_i + 1 = 1 + (\log \bar\beta_i - \log c)/\log \bar c$$

$$a_j = \log \bar\alpha_j / \log \bar c$$

where

$$\bar c = (1-c)/2$$

Then $P = \Sigma\beta_i(b_i+1) + \Sigma\alpha_j a_j$

$$= B(1-\log c/\log \bar{c}) - 1/\log \bar{c}[-\Sigma\beta_i \log\bar{\beta}_i - \Sigma\alpha_j \log \bar{\alpha}_j]$$

$$\geq (1/\log(1/\bar{c}))[-B(\log \bar{c} - \log c) + H]$$

because of $-\Sigma x_i \log x_i \leq -\Sigma x_i \log \bar{x}_i$ for $\Sigma x_i = \Sigma\bar{x}_i = 1$, $x_i \geq 0$, $\bar{x}_i \geq 0$. [cf. Kameda & Weihrauch]. Taking $d = \log(\bar{c}/c)$ and observing that $c/\bar{c} = 2c/(1-c)$ ranges over all non-negative numbers as c ranges over [0,1] yields the claim. □

Unfortunately there is no closed form expression for d_{max} which maximizes the left hand side of the inequality in theorem II. Taking $d = 0$ gives $H/\log 3 \leq P$ [Mehlhorn 75] and taking $d = \log(P/2B)$ gives $H \leq P + B \log e - 1 + \log(P/B)$ [Bayer].

We now turn to the behavior of single searches. In its full form the inequality of theorem II is:

$$\Sigma\beta_i[(-\log \beta_i - d)/\log(2+2^{-d})] + \Sigma\alpha_j[(-\log \alpha_j)/\log(2+2^{-d})]$$

$$\leq \Sigma\beta_i[b_i+1] \qquad\qquad + \Sigma\alpha_j[a_j]$$

We want to show that the inequality holds "almost" componentwise for the expressions in square brackets. More precisely, let $h \in R$, $h > 0$ and

$$N_h = \{i; (-\log \beta_i - d - h)/\log(2+2^{-d}) \geq b_i + 1\}$$

and

$$L_h = \{j; (-\log \alpha_j - h)/\log(2+2^{-d}) \geq a_j\}$$

Then

$$\sum_{i\in N_h} \beta_i + \sum_{j\in L_h} \alpha_j \leq 2^{-h}$$

i.e. for nodes and leaves with total probability $\geq 1 - 2^{-h}$ the inequality of theorem II is "almost valid" componentwise. "Almost valid" means: Up to the additive factor $-h/\log(2+2^{-d})$. In order to prove the claim we set $d=\log(\bar{c}/c)$ with $\bar{c} = (1-c)/2$ and $0 \leq c-1$. Then $d = \log \bar{c} - \log c$ and $\log(2+2^{-d}) = \log(1/\bar{c})$.

Simple calculation shows that the definitions of N_h and L_h are equivalent to

$$N_h = \{i;\ \beta_i \leq 2^{-h}\ \bar{\beta}_i\}$$

and

$$L_h = \{j;\ \alpha_j \leq 2^{-h}\ \bar{\alpha}_j\}$$

where $\bar{\beta}_i$ and $\bar{\alpha}_j$ are defined as in the lemma above. Then

$$\begin{aligned} 1 &= \Sigma\bar{\beta}_i + \Sigma\bar{\alpha}_j \\ &\geq \sum_{i\in N_h} \bar{\beta}_i + \sum_{j\in L_h} \bar{\alpha}_j \\ &\geq 2^h \left[\sum_{i\in N_h} \beta_i + \sum_{j\in L_h} \alpha_j \right] \end{aligned}$$

<u>Theorem III:</u> (Lower Bound on the search time for single searches). Let $c,h \in R$ with $0 \leq c \leq 1$ and $h > 0$. Define $\bar{\beta}_i$ and $\bar{\alpha}_j$ as in the lemma above. Let

$$N_h = \{i;\ \beta_i \leq 2^{-h}\ \bar{\beta}_i\} \qquad \text{and}$$

$$L_h = \{j;\ \alpha_j \leq 2^{-h}\ \bar{\alpha}_j\}.$$

Then $\sum_{i\in N_h} \beta_i + \sum_{j\in L_h} \alpha_j \leq 2^{-h}$. □

It is worthwile to contrast this theorem with the upper bounds of theorem 1. For simplicity suppose $\Sigma\beta_i = 0$. Then the bisection rule yields a tree with the property that the search time for leaf (B_j,B_{j+1}) is bounded above by $\log 1/\alpha_j + 2$. Conversely, consider any tree for this distribution. Then for a set of leaves having weight $\geq 1/2$ $(3/4)$, i.e. for 50 (75) % of the searches, the search time is larger than $\log 1/\alpha_j - 1$ $(\log 1/\alpha_j - 2)$. Thus the bisection rule produces trees whose worst case behavior is close to optimal and so do the other rules.

IV. The Entropy Rule: Average and Worst Case Behavior, Experimental Results

In this section we propose yet another construction rule. It is based on information theoretic considerations. A comparison with a name is a decision with three possible outcomes: <,=,>. The probabilities of the three outcomes are the weight of the left subtree W_L, the weight of the root W_{ROOT} and the weight of the right subtree W_R respectively, the information gained by this comparison is equal to the entropy $H(W_L, W_{root}, W_R)$.

Rule IV (Entropy): Choose the root so as to maximize the local information gain $(H(W_L, W_{ROOT}, W_R)$, then proceed similarly on the subtrees.

This rule was also proposed by Horibe [Horibe 77b] . The pure entropy rule behaves quite well; compare the table of experimental results below. However, already a superficial analysis shows that the pure entropy rule has some undesirable properties. Consider the following example: $\beta_1 = \beta_2 = \varepsilon$, $\beta_3 = 1 - 2\varepsilon$, $\alpha_o = \alpha_1 = \alpha_2 = \alpha_3 = 0$. The entropy rule chooses B_2 as the root. The entropy tree has weighted path length $2 - \varepsilon$, the optimal tree has weighted path length $1 + 3\varepsilon$. Since similar situations happen quite frequently for small n we rather study a modified entropy rule. The modification is based on the following lemma.

Lemma: Let $(\alpha_o, \beta_1, \ldots, \beta_n, \alpha_n)$ be a probability distribution.

1) If $\beta_i \geq \max\ (\alpha_o + \beta_1 + \ldots + \beta_{i-1} + \alpha_{i-1}, \alpha_i + \beta_{i+1} + \ldots + \beta_n + \alpha_n)$

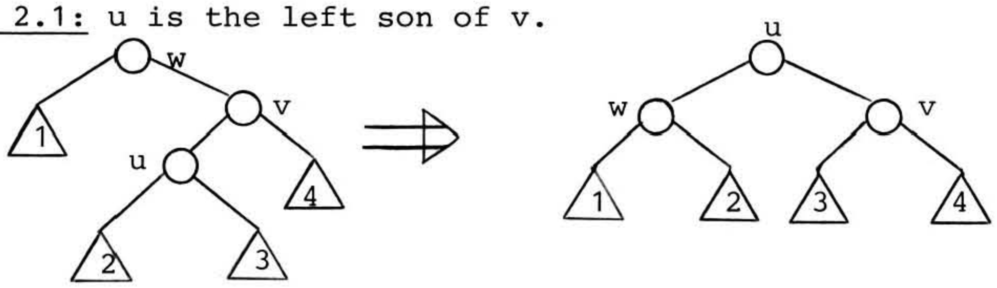
then there is an optimal tree with root B_i.

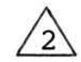
2) If $\alpha_j \geq \max\ (\alpha_o + \beta_1 + \ldots + \beta_j, \beta_{j+1} + \ldots + \alpha_n)$

then there is an optimal tree with leaf (B_j, B_{j+1}) at depth 2 or less.

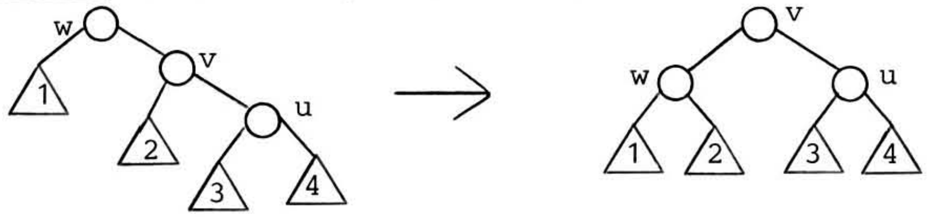
Proof: We only prove 2); the proof of case 1) is similar. Suppose that the depth of leaf (B_j, B_{j+1}) is ≥ 3. Let u,v,w be father, grandfather and grand-grandfather of leaf (B_j, B_{j+1}). Consider the subtree with root w. We may assume w.l.o.g. that v is the right son of w.

Case 2.1: u is the left son of v.



Then (B_j, B_{j+1}) is either the subtree  or . It is easy to see that the transformed tree is at least as good as the original one.

Case 2.2: u is the right son of v.



Again it is easy to see that the transformed tree is as least as good as the original one.

Hence in either case it is possible to move up leaf (B_j, B_{j+1}) one level without distroying optimality.

The Lemma suggests the following modification of the entropy rule.

RULE V (Modified Entropy): If a node B_i exists which satisfies clause 1 of the lemma then choose it as the root. If a leaf (B_j, B_{j+1}) exists which satisfies clause 2 of the lemma then choose the root among B_j and B_{j+1} . B_j is choosen if

$\alpha_o + \beta_j + \dots + \beta_j > \beta_{j+1} + \dots + \beta_n + \alpha_n$ and B_{j+1} otherwise.

In all other cases choose the root so as to maximize the local information gain $H(W_L, W_{root}, W_R)$. Then proceed similarly on the subtrees.

Next we present the results of some experiments with rules I to V.[Table I]. We applied rules I to V to the 200 distributions described by Gotlieb and Walker. We refer the reader to [Gotlieb & Walker] for a detailled description of the test data.

Note that $P_{ME} \leq P_{ENT} \leq P_{MM} \leq P_{WB} \leq P_{BI}$ in all but a few cases. The first (second, third, fourth) inequality does hold except in O(9, 4, O) cases. Only in 6 cases $P_{ME} \leq P_{MM}$ is not true.

Setting P_{OPT} to 100, the average and maximal values of $P_{ME}, P_{ENT}, P_{MM}, P_{WB}$ and P_{BI} are:

	P_{OPT}	P_{ME}	P_{ENT}	P_{MM}	P_{WB}	P_{BI}
average value of $P/P_{OPT} \cdot 100$	100	1o1.4	102.2	104.5	107.7	119.7
maximal value of $\frac{P \cdot 100}{P_{OPT}}$	100	104.2	105.6	109.0	128.6	154.7

	Optimal path length	Rule V Modified Entropy	Rule IV Entropy	Rule II Min-Max	Rule I Weight Balancing	Rule III Bisection
Set 1						
Case 1	4.29	4.32	4.45	4.55	4.93	6.09
2	6.61	6.63	6.63	7.02	7.09	7.65
3	5.87	5.94	6.01	6.11	6.12	7.19
4	6.07	6.17	6.17	6.32	6.42	7.17
5	6.64	6.69	6.71	6.79	6.82	7.40
6	5.96	6.01	6.02	6.31	6.59	7.21
7	5.83	5.91	5.92	6.14	6.33	7.21
8	6.26	6.41	6.52	6.72	6.85	7.65
9	7.09	7.34	7.47	7.45	7.44	8.30
10	7.34	7.39	7.39	7.51	7.52	7.65
Set 2						
Case 1	4.63	4.65	4.77	5.00	5.66	6.60
2	7.26	7.30	7.30	7.64	7.80	8.06
3	6.25	6.32	6.33	6.68	6.84	7.52
4	6.40	6.51	6.51	6.87	6.88	7.49
5	7.02	7.07	7.09	7.21	7.24	8.03
6	6.54	6.61	6.61	6.88	7.03	7.67
7	6.45	6.48	6.50	6.80	6.99	7.68
8	6.58	6.86	6.95	6.84	6.87	7.82
9	7.12	7.42	7.50	7.35	7.32	8.03
10	7.70	7.79	7.79	7.87	7.89	8.05
Set 3						
Case 1	4.00	4.04	4.08	4.23	4.54	6.14
2	6.42	6.46	6.49	6.56	6.99	7.37
3	5.56	5.61	5.69	5.83	5.85	6.63
4	5.76	5.84	5.85	5.98	6.21	6.58
5	6.22	6.33	6.40	6.28	6.37	6.86
6	5.65	5.68	5.75	6.02	6.25	7.10
7	5.59	5.62	5.65	5.91	6.02	7.08
8	6.09	6.23	6.38	6.39	6.62	7.65
9	7.01	7.31	7.38	7.22	7.26	8.39
10	6.97	7.04	7.05	7.10	7.09	7.32

	Optimal path length	Rule V Modified Entropy	Rule IV Entropy	Rule II Min-Max	Rule I Weight Balancing	Rule III Bisection
Set 4						
Case 1	5.04	5.04	5.08	5.38	5.88	6.67
2	7.42	7.50	7.50	7.69	7.85	8.05
3	6.55	6.61	6.61	6.81	6.98	7.56
4	6.65	6.74	6.74	7.03	7.18	7.57
5	7.27	7.32	7.35	7.45	7.46	8.02
6	6.55	6.56	6.59	6.87	7.08	7.67
7	6.75	6.81	6.82	7.05	7.18	7.78
8	6.73	6.92	7.05	6.91	6.96	8.02
9	7.11	7.39	7.50	7.29	7.31	8.15
10	7.87	7.91	7.91	8.02	8.03	8.16
Set 5						
Case 1	4.14	4.18	4.28	4.32	5.32	6.40
2	5.97	6.02	6.05	6.19	6.75	7.32
3	5.75	5.84	5.93	5.95	6.28	7.26
4	6.00	6.10	6.11	6.44	6.51	6.93
5	6.66	6.72	6.79	6.87	6.96	7.65
6	5.33	5.41	5.46	5.61	5.97	7.01
7	5.41	5.48	5.53	5.90	6.37	7.29
8	5.81	5.92	5.97	6.05	6.36	7.14
9	6.99	7.17	7.31	7.22	7.24	8.09
10	7.27	7.32	7.32	7.42	7.41	7.63

Table 1: Weighted path length P_{opt}, P_{WB}, P_{MM}, P_{BI}, P_{ENT} and P_{MENT}.

Remark: For comparison with the results of Gotlieb & Walker in this table we used their definition of the weighted path length:

$P = \sum_{i=1}^{n} \beta_i (b_i+1) + \sum_{j=0}^{n} \alpha_j (a_j+1)$ (instead of our definition of page 1).

We now turn to the analysis of the modified entropy rule.

Theorem III (Average case behavior of the entropy rule).

Case 1: $\Sigma\alpha_j = 0$: $P_{ME} \leq H + 1$

Case 2: $\Sigma\beta_i = 0$: $P_{ME} \leq H + 2$

Case 3: $\Sigma\alpha_i \geq 0$, $\Sigma\beta_i \geq 0$

$P_{ME} \leq c_1 \cdot H + c_2$

where $c_1 = 1/H(1/3, 2/3) \approx 1{,}08$ and $c_2 = 2$.

Proof: It is easy to see that in the case $\Sigma\beta_i = 0$ the entropy rule and modified entropy rule is identical with the min-max rule. Hence $P_{ME} \leq H + 2$ in that case.

We treat cases 1 and 3 together. We want to prove $P_{ME} \leq c_1 \cdot H + c_2$ for suitable constants c_1 and c_2 by induction on n.

n = 0 or n = 1 : Then $P \leq 1$ and hence $c_2 \geq 1$ suffices.

n > 1. Suppose B_i is the root of the tree constructed according to the modified entropy rule. Let $W_L = \alpha_o + \beta_1 + \ldots + \beta_{i-1} + \alpha_{i-1}$, $W_R = \alpha_i + \beta_{i+1} + \ldots + \beta_n + \alpha_n$,

$$(\alpha'_o, \beta'_o, \ldots, \beta'_{i-1}, \alpha'_{i-1}) = (\frac{\alpha_o}{W_L}, \frac{\beta_1}{W_L}, \ldots, \frac{\beta_{i-1}}{W_L}, \frac{\alpha_{i-1}}{W_L}) ,$$

$$(\alpha'_i, \beta'_{i+1}, \ldots, \beta'_n, \alpha'_n) = (\frac{\alpha_i}{W_R}, \frac{\beta_{i+1}}{W_R}, \ldots, \frac{\beta_n}{W_R}, \frac{\alpha_n}{W_R})$$

P_L the weighted path length of the left subtree and P_R the weighted path length of the right subtree. Then

$$P = 1 + W_L \cdot P_L + W_R \cdot P_R$$

$$P_L \leq c_1 \; H(\alpha'_o, \beta'_1, \ldots, \beta'_{i-1}, \alpha'_{i-1}) + c_2$$

$$P_R \leq c_1 \; H(\alpha'_i, \beta'_{i+1}, \ldots, \beta'_n, \alpha'_n) + c_2$$

Hence

$$P \leq c_1 [H(\alpha_o, \beta_1, \ldots, \beta_n, \alpha_n) - H(W_L, \beta_i, W_R)]$$

$$+ c_2 [W_L + W_R] + 1$$

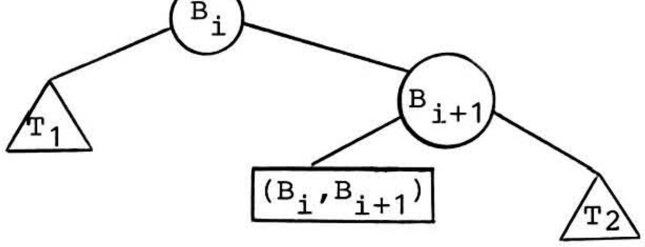
$$= c_1 \; H + c_2 + [1 - c_2 \beta_i - c_1 \; H(W_L, \beta_i, W_R)]$$

It remains to show

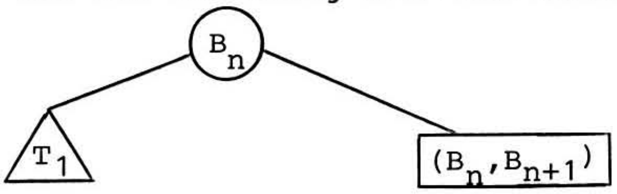
$$1 \leq c_2 \beta_i + c_1 \; H(W_L, \beta_i, W_R)$$

a) B_i was chosen as the root because $\beta_i \geq \max(W_L, W_R)$. Then W_L, $W_R \leq 1/2$. If $\beta_i \leq 1/2$ then $H(W_L, \beta_i, W_R) \geq H(1/2, 1/2, 0)$ and hence $c_1 \geq 1$ suffices. If $\beta_i > 1/2$ then $c_2 \geq 1$ does it.

b) B_i was chosen because $\alpha_i \geq W_L + \beta_i \geq W_R - \alpha_i$ (The symmetric case that $\alpha_{i-1} \geq \beta_i + W_R \geq W_L - \alpha_{i-1}$ is treated similar). Then either i<n and the following tree was constructed



or i = n and the following tree was constructed:



In the first case let P_i be the weighted path length of tree T_i , $i = 1,2$. Then

$$P = 1 + W_L \cdot P_1 + W_R \quad + (W_R - \alpha_i - \beta_{i+1}) \cdot P_2$$

$$\overset{\text{I.H.}}{\leq} \quad c_1[H(\alpha_o,\beta_1,\ldots,\beta_n,\alpha_n) - H(W_L,\beta_i,\alpha_i,\beta_{i+1}, W_R-\alpha_i-\beta_{i+1})]$$

$$+ c_2[W_L + W_R - \alpha_i - \beta_{i+1}] + 1 + W_R$$

$$\leq \quad c_1 \ H + c_2 + [1+W_R-c_2(\beta_i+\alpha_i+\beta_{i+1})-c_1H(W_L+\beta_i,\alpha_i,W_R-\alpha_i)]$$

If $\alpha_i < 1/2$ then $H(W_L + \beta_i,\alpha_i,W_R - \alpha_i) \geq 1$ and

$W_R \leq (W_R - \alpha_i) + \alpha_i \leq 2\alpha_i$. Hence $c_1 \geq 1$ and $c_2 \geq 2$ suffice.

If $\alpha_i \geq 1/2$ then we use $H(W_L + \beta_i , \alpha_i, W_R - \alpha_i)$

$= H(\alpha_i,1-\alpha_i) + (1-\alpha_i) \ H((W_L+\beta_i)/(1-\alpha_i), (W_R-\alpha_i)/(1-\alpha_i))$

$$\geq 2(1-\alpha_i) + (1-\alpha_i) \cdot 2 \cdot (W_R - \alpha_i)/(1 - \alpha_i)$$

$= 2(1 + (W_R - \alpha_i) - \alpha_i)$. Here we used $H(x,1-x) \geq 2x$ if $x \leq 1/2$. Hence

$1 + W_R - c_2 \ (\beta_i + \alpha_i + \beta_{i+1}) - c_1 \ H(W_L + \beta_i, \alpha_i,W_R - \alpha_i)$

$\leq (1 - 2c_1) + (W_R - \alpha_i)(1 - 2c_1) + \alpha_i(1 - c_2 + 2c_1)$

≤ 0 if $c_1 \geq 1$ and $c_2 \geq 2$.

In the second case let P_1 be the weighted path length of tree T_1. Then

$$P \leq 1 + W_L \cdot P_1$$

$$\overset{\text{I.H.}}{\leq} \quad c_1[H(\alpha_o,\beta_1,\ldots,\beta_n,\alpha_n) - H(W_L,\beta_n,\alpha_n)] + c_2W_L + 1$$

$$\leq \quad c_1 \cdot H + c_2 + [1 - c_2 \cdot \alpha_n]$$

From $W_L + \beta_n = 1 - \alpha_n \leq \alpha_n$ we infer $\alpha_n \geq 1/2$.

Hence $c_2 \geq 2$ suffices.

c) B_i was chosen because it maximizes $H(W_L, \beta_i, W_R)$. We distinguish two subcases.

c1) There exists a $j, 1 \leq j \leq n$, such that

$\alpha_o + \beta_1 + \ldots + \beta_{j-1} + \alpha_{j-1} \leq 1/2$ and

$\alpha_j + \beta_{j+1} + \ldots + \beta_n + \alpha_n \leq 1/2$. Since case a) does not apply $\beta_j \leq 1/2$. Hence

$$H(W_L, \beta_i, W_R) \geq H(\alpha_o + \beta_1 + \ldots + \alpha_{j-1}, \beta_j, \alpha_j + \beta_{j+1} + \ldots + \beta_n + \alpha_n)$$
$$\geq H(1/2,\ 1/2) = 1$$

Hence $c_1 \geq 1$ suffices.

c2) There exists no such j. Hence there exists a j such that

$\alpha_o + \beta_1 + \ldots + \alpha_{j-1} + \beta_j \leq 1/2$ and

$\beta_{j+1} + \alpha_{j+1} + \ldots + \beta_n + \alpha_n \leq 1/2$

Assume w.l.o.g. that

$\alpha_o + \beta_1 + \ldots + \alpha_{j-1} + \beta_j \geq \beta_{j+1} + \alpha_{j+1} + \ldots + \beta_n + \alpha_n$.

Since case b) does not apply we have

$\alpha_o + \beta_1 + \ldots + \alpha_{j-1} + \beta_j \geq \alpha_j$

and hence

$\alpha_o + \beta_1 + \ldots + \alpha_{j-1} + \beta_j \geq 1/3$

So we have

$$H(W_L, \beta_i, W_R) \geq H(\alpha_o + \beta_1 + \ldots + \alpha_{j-1}, \beta_j, \alpha_j + \beta_{j+1} + \ldots + \beta_n + \alpha_n)$$

$\geq H(\alpha_o+\beta_1+\ldots+\alpha_{j-1}+\beta_j, \alpha_j+\beta_{j+1}+\ldots+\beta_n+\alpha_n)$

$\geq H(1/3, 2/3)$

Hence $c_1 \geq 1/H(1/3, 2/3)$ suffices.

In the case that $\Sigma\alpha_j = 0$ only subcases a) and c1) arise. Hence $c_1 = c_2 = 1$ do the job. In the general case we have to choose $c_2 = 2$ and $c_1 = 1/H(1/3,2/3)$. □

The problem whether $P_{ME} \leq H + 2$ is true in the general case is open. In view of the empirical evidence a positive answer seems likely.

Next we investigate the worst case behavior of the entropy rule. Here we can only present results which are probably far from being final.

Theorem IV (Worst Case Behavior of Entropy Trees):

Let b_i^{ME} (a_j^{ME}) be the depth of node B_i (leaf(B_j,B_{j+1})) in a tree constructed according to the modified entropy rule Then

$$b_i^{ME} \leq (1/\log(1/1-\xi))\cdot\log 1/\beta_i + 1$$

$$a_j^{ME} \leq (1/\log(1/1-\xi))\cdot\log 1/\alpha_j + 2$$

where ξ is defined by $H(\xi,1-\xi) + \xi = H(1/3,2/3)$.

Then $\xi \approx 0.195$ and $1/\log(1/1-\xi)) \approx 3.19$.

Proof: The proof is similar to the one given in Mehlhorn [75]. We need the following Lemmas.

Lemma 1: Let T be a binary tree which is constructed to the modified entropy rule. Let B be an interior node with distance 2 from the root. Let w_o be the weight of T, w_1 be the weight of the direct subtree of T which contains B, and let w_2 be the total weight of the tree with root B. Then either

$$w_1 \leq (1 - \xi)w_o \quad \text{or} \quad w_2 \leq 1/3\, w_o$$

Proof: We distinguish three cases according to which clause of the definition of the modified entropy rule was used in the construction of T.

Case 1: There is a β_i with $\beta_i \geq \max(\alpha_o+\beta_1+\ldots+\beta_{i-1}+\alpha_{i-1}, \alpha_i+\beta_{i+1}+\ldots+\beta_n+\alpha_n)$.

Then B_i is taken as the root. Hence $\beta_i \geq w_1$ and $w_1 \leq 1/2$.

Case 2: There is a α_j with $\alpha_j \geq \max(\alpha_o+\beta_1+\ldots+\beta_j, \beta_{j+1}+\ldots+\beta_n+\beta_n)$.

Then the leaf (B_j,B_{j+1}) is at depth 1 or 2. If B is in the same direct subtree of T as (B_j,B_{j+1}) is in then $w_2 \leq 1/3$, otherwise $w_1 \leq 1/2$.

Case 3: The root was chosen because it maximizes the entropy. Assume w.l.o.g. that B is in the right subtree of T. Then

$$H(w_L, w_{root}, w_1/w_o) \geq H(1/3,\ 2/3)$$

by the argument used in the proof of theorem III.
Since

$$H(w_L, w_{root}, w_1/w_o) \leq H(1-w_1/w_o, w_1/w_o) + (1-w_1/w_o)$$

by the grouping theorem we have

$$(1-w_1/w_o) + H(1-w_1/w_o, w_1/w_o) \geq H(1/3,2/3)$$

Hence $(1-w_1/w_o) \geq \xi$ □

Lemma 2: Let T be a binary tree which is constructed according to the modified entropy rule and let B be an interior node with distance b from the root. Let w be the weight of the subtree with root B. Then

$$w \leq (1-\xi)^{(b-1)}$$

Proof: The claim is obvious for $b \leq 1$. Otherwise, let $B_{k_o}, B_{k_1}, \ldots, B_{k_b} = B$ be the nodes on the path from the root to B and let W_i be the weight of the subtree with root B_{k_i}.

We show: for all i either

$$w_{i-1} \leq (1-\xi)^{i-1}$$

or $$w_i \leq (1-\xi)^i$$

For i = 2 this follows from Lemma 1. If i > 2 then either $w_{i-2} \leq (1-\xi)^{i-2}$ or $w_{i-1} \leq (1-\xi)^{i-1}$ by induction hypothesis.

In the second case we are done. In the first case we apply Lemma 1 and obtain:

either $w_{i-1} \leq (1-\xi)\ w_{i-2} \leq (1-\xi)^{i-1}$

or $w_i \leq 1/3 \cdot w_{i-2} \leq (1-\xi)^i$

Hence

$$w = w_b \leq \min(w_b, w_{b-1}) \leq (1-\xi)^{(b-1)}$$ □

Together with the observation that the weight of node B_i (of leaf (B_j,B_{j+1})) is certainly not larger than the weight of the subtree with root B_i (with leaf (B_j,B_{j+1}) at depth 1) Lemma 2 yields the theorem. □

V. Implementations

In this section we do a comparative study of different implementations of the construction rules.

Rules I, II, III : Suppose, node B_i is taken as the root. Then the weight of the left subtree is $\sum_{j=0}^{i-1} \alpha_j + \sum_{j=1}^{i-1} \beta_j$, the weight of the right subtree is $\sum_{j=i}^{n} \alpha_j + \sum_{j=i+1}^{n} \beta_j$.

The weight of the left (right) subtree is an increasing (decreasing) function of i. Let i_o be the largest i such that $\sum_{j=0}^{i-1} \alpha_j + \sum_{j=1}^{i-1} \beta_j$ is $\leq 1/2$, let i_o be 0 if no such i exists. Then rules I and II choose either B_{i_o} or B_{i_o+1} as the root. For rule III we have to replace 1/2 by some number which is determined dynamically by the algorithm. Hence the problem of determining the root essentially reduces to the following problem :

<u>Problem:</u> Let $F : \{1,...,n\} \to \mathbb{R}$ be monotonically increasing with $0 \leq F(1) \leq F(n) \leq 1$. Find the largest i such that $F(i) \leq 1/2$, say i_o.

Three strategies were proposed to solve this problem.

<u>Binary Search:</u> We try $i = n/2$ first.

If $F(n/2) > 1/2$ then do a binary search on the left subinterval $[1, n/2]$, otherwise on the right subinterval.

Binary search determines i_o in $O(\log n)$ units of time.

<u>Linear Search:</u> Search for i_o simultaneously from both ends, i.e. try $i = 1,n,2,n-1,3,n-3,...$ in that order. Linear search finds i_o in $O(\min(i_o,n-i_o+1))$ units of time.

Binary search leads to the following recurrence relation for the worst case running time

$$T(n) \leq \max_{1 \leq i \leq n} [T(i-1) + T(n-i) + O(\log n)]$$

which has the solution

$$T(n) = O(n \log n)$$

The worst case behavior occurs if nodes with small ($i \approx 1$) or large ($i \approx n$) index are chosen repeatedly as the roots.

Linear search leads to the following recurrence relation for the worst case running time

$$T(n) \leq \max_{1 \leq i \leq n} [T(i-1) + T(n-i) + O(\min(i,n-i+1))]$$

which has the solution [Mehlhorn 75]

$$T(n) = O(n \log n)$$

The worst case behavior occurs if nodes B_i with $i \approx n/2$ are chosen repeatedly as the roots.

<u>Exponential Search + Binary Search</u> [Fredman] :

Search for i_0 with exponentially increasing steps from both ends, i.e. try 1, n, 1+1, n-1, 1+2, n-2, 1+4, n-4, 1+8, n-8,... in that order. This search determines an interval

$[1+2^p, 1+2^{p+1}]$ ($[n-2^{p+1}, n-2^p]$) for i_0 in $O(p)$ steps.

Then do a binary search on this interval. ($O(p)$ steps)).
This search method determines i_0 in $O(\min(\log i_0, \log(n-i_0)))$ units of time.

It leads to the following recurrence relation for the worst case running time

$$T(n) \leq \max_{1 \leq i \leq n} [T(i-1) + T(n-i) + O(\log \min(i,n-i))]$$

which has a linear solution [Fredman].

$$T(n) = O(n) .$$

RULE IV: Rule IV determines the root so as to maximize H(weight of the left subtree, weight of the root, weight of the right subtree). This function is not a monotone function of the index of the root. Therefore, one has to try every i, $1 \le i \le n$, in order to determine i_0. This leads to the following recurrence relation for the worst case running time.

$$T(n) \le \max_{1 \le i \le n} [T(i-1) + T(n-i) + n]$$

which has the solution

$$T(n) = O(n^2) .$$

So far we surveyed known results about the worst case behavior of different implementations. We turn now to average case behavior. We analyse the average case running time under the (conservative) assumption that the root index is uniformly distributed in the interval {1,...,n}. The same results hold under the assumption that all frequency distributions are equally likely, though calculations are more tedious.

Under the above assumption the following recurrence relations describe the average case behavior of the implementations.

Rules I, II and III:

Binary Search: $T(n) = 1/n \sum_{i=1}^{n} (T(i-1)+T(n-i)+\log n)$

solution : $T(n) = O(n)$

Linear Search: $T(n) = 1/n \sum_{i=1}^{n} (T(i-1)+T(n-i)+O(\min(i,n-i+1)))$

solution: $T(n) = O(n \log n)$

<u>Exponential Search:</u> $T(n) = 1/n \sum_{i=1}^{n} (T(i-1)+T(n-i)+ O(\log(\min(i,n-i+1)))$

solution: $T(n) = O(n)$

<u>Rule IV:</u> $T(n) = 1/n \sum_{i=1}^{n} (T(i-1)+T(n-i)+n)$

solution: $T(n) = O(n \log n)$

We summarize the running times in table 2.

	Worst Case	Average Case
Rules I, II, III		
Binary Search	$O(n \log n)$	$O(n)$
Linear Search	$O(n \log n)$	$O(n \log n)$
Expon. Search	$O(n)$	$O(n)$
Rule IV	$O(n^2)$	$O(n \log n)$

<u>Table 2:</u> Running times of different implementations.

VI. Application to digital search trees

Quite frequently the "names" B_i will be strings. Instead of basing the search method on comparisons between names, we can make use of the representations as a sequence of characters. A classic example of such a search method is the trie [cf. Knuth 73, Sec 6.3] . Suppose that the names $B_1,\dots,B_n$ are strings over a k character alphabet Σ . A trie is then a k-ary tree. For each prefix of a name B_i there is a node in the tree, the branching is done on the next character.

Different representations of the nodes of a trie were proposed: vectors of length k, linked lists or binary trees. The first alternative minimizes processing time, the two others save memory space.

From now on we restrict ourself to the case that the weight is concentrated in the keys. $(\Sigma\alpha_j = 0)$. This restriction simplifies the notation; the general case may be treated analogously. Consider a node of the trie. It corresponds to a string $w \in \Sigma^*$. Let p_w be the sum of the probabilities of all words B_i having w as a prefix. Then in node w the branch corresponding to character $a \in \Sigma$ is taken with probability p_{wa}/p_w.
Hotz [Hotz] proposed to represent each node by an optimal (or nearly so) binary search tree; he showed that this strategy works well in the case of "uniform distributions". We show that it performs well for all distributions.

Assume that we represent each node of a trie by a binary search tree whose weighted path length is bounded above by $c_1 H + c_2$ where H is the entropy of the associated frequency distribution.
Consider a search for $B_i = a_1a_2\dots a_{\ell_i}$ with $a_j \in \Sigma$, ℓ_i = length of string B_i. We search first for the character a_1 in the top

level tree, then for a_2 in the tree $T_{a_1}, \ldots$. The average search time is given by the expression (a comparison between two characters is assumed to take one unit of time):

$$\sum_{i=1}^{n} \beta_i (\text{\# of comparisons needed to find } B_i)$$

$$= \sum_{i=1}^{n} \beta_i \cdot \sum_{j=1}^{\ell_i} \left(\begin{matrix}\text{\# of comparisons needed to find} \\ a_j \text{ in } T_{a_1,\ldots,a_{j-1}}\end{matrix}\right)$$

$$= \sum_{v\in\Sigma^*} \cdot \sum_{a\in\Sigma} \left(\begin{matrix}\text{\# of comparisons needed to find} \\ a \text{ in } T_v\end{matrix}\right) \cdot p_{va}$$

$$= \sum_{v\in\Sigma^*} p_v \sum_{a\in\Sigma} \left(\begin{matrix}\text{\# of comparisons needed to find} \\ a \text{ in } T_v\end{matrix}\right) \cdot \frac{p_{va}}{p_v}$$

$$\leq \sum_{v\in\Sigma^*} p_v (c_2 + c_1 \cdot H(\frac{p_{va}}{p_v}, \frac{p_{vb}}{p_v}, \frac{p_{vc}}{p_v}, \ldots))$$

$$\leq c_2 \cdot \sum_{v\in\Sigma^*} p_v + c_1 \cdot \sum_{v\in\Sigma^*} p_v \, H(\frac{p_{va}}{p_v}, \frac{p_{vb}}{p_v}, \ldots)$$

$$\leq c_2 \cdot \sum_{i=1}^{n} \beta_i \ell_i + c_1 \, H(\beta_1,\ldots,\beta_n) \quad \text{(grouping theorem cf. Ash)}$$

$$\leq c_2 \cdot \overline{\ell} + c_1 \cdot H(\beta_1,\ldots,\beta_n)$$

where $\overline{\ell}$ is the weighted average length of the strings B_i.

Thm.V: Suppose that we represent each node of a trie by a binary search tree whose weighted path length is bounded above by $c_1 H + c_2$, then the average search time is bounded by

$$c_1 H(\beta_1,\ldots,\beta_n) + c_2 \, \overline{\ell}$$

where $\overline{\ell} = \sum_{i=1}^{n} \beta_i \cdot \text{length}(B_i)$ is the average length of the names B_i.

Consider the case that all names have length m. It is reasonable to assume that the time needed to compare two names of length m takes O(m) units of time. Using binary search trees based on comparison of entire names then results in an average search time of $O(m\cdot(c_1H+c_2))$. This contrasts sharply to the average search time of $O(c_1H + c_2\cdot m)$ achieved by the search method based on comparisons of characters.

We turn now to worst case behavior. Assume that we represent each node of a trie by a binary search tree whose worst case behavior is bounded by $c_1\cdot\log 1/\text{probability} + c_2$; i.e. a search for a name with probability p takes at most $c_1\cdot\log 1/p + c_2$ units of time. Then

$$\begin{aligned}
&\#\text{ of comparisons needed to find } B_i\\
&= \sum_{j=1}^{\ell_i} (\#\text{ of comparisons needed to find } a_j \text{ in } T_{a_1\ldots a_{j-1}})\\
&\le \sum_{j=1}^{\ell_i} \left(c_1\cdot\log\frac{p_{a_1\ldots a_{j-1}}}{p_{a_1\ldots a_j}} + c_2\right)\\
&\le c_1 \cdot \sum_{j=1}^{\ell i} (\log p_{a_1\ldots a_{j-1}} - \log p_{a_1\ldots a_j}) + c_2\ell_i\\
&\le c_1 \cdot (\log p_\varepsilon - \log p_{a_1\ldots a_{\ell_i}}) + c_2\ell_i\\
&\le c_1\ (0 + \log 1/\beta_i) + c_2\ell_i
\end{aligned}$$

<u>Thm.VI:</u> Assume that we represent each node of a trie by a binary search tree whose worst case behavior is bounded above by $c_1 \cdot \log 1/\text{probability} + c_2$, then a search for name B_i of length ℓ_i and frequency β_i takes at most

$$c_1 \cdot \log 1/\beta_i + c_2\ell_i$$

comparisons between characters.

The remark following the preceding theorem applies here as well.
For the special case that all names have equal probability 1/n and equal length m theorem VI yields the bound $c_1 \log n + c_2 m$.
This special case was considered previously by Fredman and v. Leeuwen.

Bibliography

[Ash] "Information Theory", Interscience Publishers,N.Y.1965

[Bayer,P.] "Improved Bounds on the Costs of Optimal and Balanced Binary Search Trees", to appear in ACTA INFORMATICA

[Bruno, J. & Coffman, E.G.] "Nearly Optimal Binary Search Trees" Proc. IFIP Congress 1971

[Fredman, M.L., 75] "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

[Fredman, M.L., 76] "How Good is the Information Theory Bound in Sorting, Theoretical Computer Science 1 (4), 355-362, 1976

[Gilbert, E.N. & Moore, E.F.] "Variable Length Encodings" Bell System Technical Journal, 1971

[Gotlieb, C.C. & Walker, W.A.] "A Top-Down Algorithm for Constructing Nearly Optimal Lexicographical Trees" Graph Theory and Computing, Academic Press, 1972

[Horibe, Y.] "An improved bound for weight balanced trees", Information and Control, 34, 1977.

[Hotz, G.] "Schranken für Balanced Trees bei ausgewogenen Verteilungen", Theoretical Computer Science 3, 1977

[Hu, T.C. & Tucker, A.C.] "Optimal Computer Search Trees and Variable Length Alphabetic Codes", Siam J. Applied Math. 21, 1971

[Kameda, Weihrauch] "Codierungstheorie", Bibliographisches Institut

[Knuth, D.E., 71] "Optimum Binary Search Trees", Acta Informatica I, 1971

[Knuth, D.E., 73] "The Art of Computer Programming", Vol. II, Addison Wesley, 1973

[Mehlhorn, K.,75a] "Nearly Optimal Binary Search Trees", Acta Informatica 5, 1975

[Mehlhorn, K., 77a]"Best Possible Bounds on the Weighted Path Length of Optimum Binary Search Trees" SIAM Journal of Computing, Vol. 6, No. 2, 1977

[Mehlhorn, K., 77b]"Effiziente Algorithmen", Teubner Studienbücher Informatik, 1977

[Meschkowski] "Differenzengleichungen", Vandenhoeck u. Ruprecht, Göttingen, 1959

[v. Leeuwen] "The complexity of data organization", Math. Center Tracts, Vol. 81, 1976

[Rissanen, J.],"Bounds for Weight Balanced Trees", IBM Journal of Research and Development, 1973