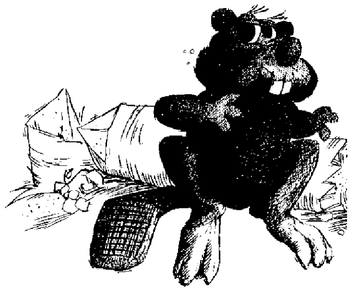


Search Trees and Bubble Memories

Philippe Flajolet
Thomas Ottmann
Derick Wood

Data Structuring Group
CS-83-02

November, 1983

SEARCH TREES AND BUBBLE MEMORIES[1]

Philippe Flajolet[2]

Thomas Ottmann[3]

Derick Wood[4]

ABSTRACT

We consider the storage of binary search trees in major-minor loop configurations of bubble memories. This leads, under reasonable assumptions, to the investigation of two cost measures for binary search trees, free search cost FCOST, and root-reset search cost RCOST. We analyze the average case behaviour of both cost measures and characterize their associated minimal cost trees. The average case analyses are themselves of interest since they are examples of the application of a recently developed methodology.

1. INTRODUCTION

Because bubble memory devices are now a practical proposition, the mathematical analysis of their properties is a useful and fruitful exercise, for example see Chandra and Wong (1979), Chung, Luccio and Wong (1979, 1980), and Bongiovanni and Wong (1981).

One area of concern is the representation of standard data structures in various bubble memory configurations. This representation or encoding problem has been much studied for standard memory configurations, for example see Rosenberg (1978), Rosenberg and Snyder (1978) and Standish (1980). Recently Bongiovanni and Wong (1980) have considered the representation of tree search in bubble memories. Their concern is related to yet different from ours; related because they consider the (implicit) representation of trees, and different because they are concerned with fixed

---

[1] This research was partially carried out under a Natural Sciences and Engineering Research Council of Canada Grant No. A-5692 and partially under NATO Grant No. RG 155.81.

[2] INRIA, Domaine de Voluceau, Rocquencourt, F-78150 Le Chesnay, France.

[3] Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, D-7500 Karlsruhe, West Germany.

[4] Data Structuring Group, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

or static trees, whereas our concern is the dynamic behaviour of explicit representations of trees.

We study the representation of binary search trees in a major-minor loop bubble memory configuration. Abstracting this, in Section 2, we are led to the problem of representing a binary search tree in a two-way circular list, see Vaishnavi and Wood (1982). We then make the reasonable assumption that comparison time far outweighs bubble movement time (or entry point movement time in our abstraction), which leads naturally to the concepts of root-reset search cost and free search cost for binary search trees whose analyses we then undertake.

We first derive in Section 3 some basic properties of both cost measures including the characterization of their associated minimal cost trees. Second we derive the average distance between two nodes in a binary tree and in a binary search tree of n nodes, often called shape or static analysis and search or dynamic analysis, respectively, giving the corresponding average free search costs. The techniques used for these derivations are those introduced in Flajolet (1981), which we explain in some detail. Then in Section 5 we analyse the average behavior of root-reset search cost. Fairley (1973), analyzed the average behavior for complete binary trees under this cost measure, which he called random entry search cost.

Finally in Section 6 we compare our results with those for the usual cost measure on binary search trees.

2. BINARY SEARCH TREES IN BUBBLE MEMORIES

The typical major-minor loop configuration for bubble memories is shown in Figure 2.1.

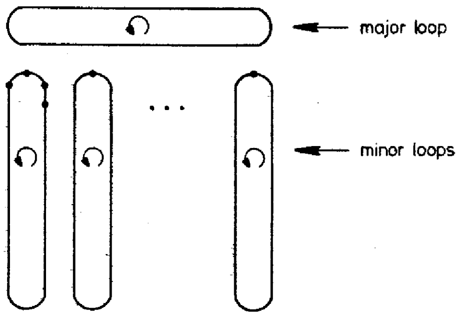


Figure 2.1: Major-Minor Loop

The bubbles in the minor loops are simultaneously rotated in either a clockwise or anti-clockwise direction. One position in each minor loop is designated as a transfer position. At any time the values in these designated positions can either be transmitted to or changed by their corresponding bubbles in the major loop. The major loop also has a designated transfer position or "window" to the "outside world".

To clarify its use, consider a table of m keys each of n bits. This could be stored in n minor loops where the loops contain at least m bubbles. A "row" of bubbles in the n minor loops represents a single key. A linear search for a key k of n bits might proceed as follows:

(1) Let $i = 1$.

(2) Transfer key l in the "windows" of the minor loops to the major loop.

(3) Do bit by bit comparison of k, in main memory, with l in the major loop through its window, rotating one position at a time as long as the comparison is successful.

(4) If the comparison is successful stop with 'SUCCESS'.

(5) Increase i by one. If $i \leq m$, rotate minor loops by one position and goto step 2, otherwise stop with 'FAILURE'.

As is well known the rotate operation is unusually fast, viz. approximately 10^6 positions per second. Hence the dominant primitive operations in the above algorithm are the major-minor transfer and the comparison operations. Since one major-minor transfer is needed for each comparison, henceforth the comparison operation will be assumed to include the major-minor transfer operation. This means that we may consider a two-way circular list, see Figure 2.2, to be the abstraction of a major-minor loop configuration, where link chasing is inexpensive compared with the examination of a node and a comparison with its contents.

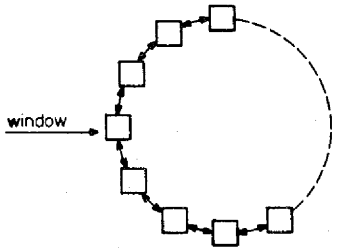


Figure 2.2

In this setting each node of the list represents the whole key (collected across the minor loops).

Let us now turn to the representation of binary search trees in such a circular list, and hence in a major-minor loop configuration. Each node of the binary search tree is assigned to a unique node of the circular list such that the assigned nodes are contiguous. The left and right links in each node of the tree are then converted into offset values, that is move clockwise or anti-clockwise by q nodes. In Figure 2.4 we have represented the tree of Figure 2.3 in symmetric order, and since each son appears later than (clockwise of) its father all offsets are positive.

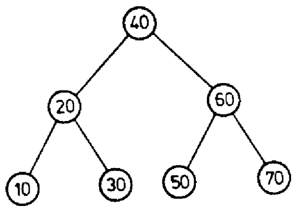
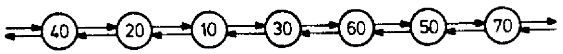


Figure 2.3



position	1	2	3	4	5	6	7
left offset	+1	+1	0	0	+1	0	0
right offset	+4	+2	0	0	+2	0	0

Figure 2.4

The search strategy may now mirror that for binary search trees by assuming that the circular list is always reset to the root of the tree (position 1 in Figure 2.4). The offset denotes the number of nodes to be skipped in either a clockwise or anti-clockwise direction.

Observe that the offsets are not, and cannot be, of equal value, hence we may interpret them as different length edges. For example in Figure 2.5 the tree of Figure 2.3 is drawn with the offset values of Figure 2.4 as the edge lengths. Bongiovanni and Wong (1980) consider binary search trees with variable edge lengths induced by such a storage representation. It is not clear whether or not the minimal cost tree is Figure 2.3 is also minimal with respect to this edge expansion. However Vaishnavi and Wood (1982) show that this is indeed the case and moreover any seven node tree is equally costly under the new cost measure. However this is not the case when the keys and gaps have unequal probabilities.

As pointed out above, because of the dominance of the comparison operation over the rotation operation, our main interest is not this variable edge length model. But having dismissed the rotation operations necessary along the search path there remains the reset rotation, which we would like to

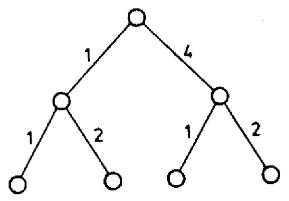


Figure 2.5: Variable Edge Lengths

avoid. We avoid it by starting a new search at the node where the previous search terminated.

To do this effectively a new searching strategy is necessary. At least two reasonable ones exist, see Section 6 for a further discussion of this issue.

Method 1: Search from the entry node as if it is the root node. If the search is unsuccessful begin again at the root of the tree. If the second search is unsuccessful then the search key is not present.

Method 2: Check if the search key is within the interval specified by the subtree of the entry node. If it is then search in the usual way, otherwise back-up to the father if one exists and repeat the process. If no father exists then the search process is at the root and the search key is not in the tree.

Method 1 has been studied previously in a limited way by Fairley (1973): We call such search trees, *root-reset search trees*. Method 2 has not, as far as we are aware, been studied previously: we call such trees, *free search trees*. The associated cost measures we call *root-reset search cost RCOST*, and *free search cost FCOST*, respectively. Two reasonable bubble memory representations suggest themselves for Method 1. Either each leaf contains its offset from the root or each node contains its offset from its father; we will assume the former. For Method 2 the latter representation is the appropriate one.

Both representations easily support insertions; deletion is more difficult, since it leaves gaps in the representation. Since each node is also 'linked' to its father when using Method 2, the tree has bi-directed edges, in other words undirected edges. Moreover although the trees are ordered, since they are search trees, their behavior is closer to that of unordered, unoriented trees, that is free trees, Knuth (1968). This is the reason for calling them *free* search trees. Such trees are of independent interest, for example the notion of *rivalling* of processors is possible on free search trees, Mühlbacher (1982).

3. PRELIMINARY DEFINITIONS AND RESULTS

In the present section we define the two new search cost measures on binary search trees and characterize minimal cost search trees under these measures.

Let T be a binary search tree with root ρ. For each node u in T let $T(u)$ denote the subtree of T rooted at u, and let $val(u)$ denote the value associated with u. As is usual we distinguish between *internal nodes,* which have two successors and *leaf nodes* which have none.

Then for any two nodes u and v in T let *the distance from u to v,* denoted by *$dist(u, v)$,* be defined as the length of the shortest path from u to v in T. Similarly let *the value distance of x from u,* denoted by *$vdist(u, x)$* denote the length of the standard search path for the value x in T beginning at node u. If x appears in $T(u)$ at node v then $vdist(u, x) = dist(u, v)$, otherwise it is the distance from u to the leaf representing an unsuccessful search for x.

We can now define the *reset distance of v from u* in T, denoted by *$rdist(u, v)$,* as either $dist(u, v)$ if v is in $T(u)$ or $vdist(u, val(v)) + dist(\rho, v)$ otherwise.

These distance measures lead to three associated cost measures for binary search trees. The usual search cost measure is defined as:

$$COST(T) = \sum_{u \text{ in } T} dist(\rho, u)$$

while the *free search cost measure* is captured by:

$$FCOST(T) = \sum_{u,\, v \text{ in } T} dist(u, v).$$

and the *root-reset search cost measure* by:

$$RCOST(T) = \sum_{u,\, v \text{ in } T} rdist(u, v).$$

As with standard binary search trees we may consider the corresponding *extended* cost measures. For this purpose trees are assumed to be extended by the addition of external nodes to all leaves and semi-leaves. Extended binary trees contain only binary and nullary nodes. The nullary or external nodes correspond to unsuccessful searches in the tree. We now obtain:

$$ECOST(T) = \sum_{u \text{ in } T} dist(\rho, u), \quad \textit{where } u \textit{ is a leaf}$$

$$EFCOST(T) = \sum_{u,\, v \text{ in } T} dist(u, v), \quad \textit{where } u \text{ and } v \textit{ are leaves}$$

$$ERCOST(T) = \sum_{u,\, v \text{ in } T} rdist(u, v), \quad \textit{where } u \textit{ is an internal node}$$
$$\text{and } v \textit{ is a leaf}.$$

The cost and extended cost measures for the first two measures are closely related as we summarize in the following:

Proposition 3.1 Let T be a binary search tree with n internal nodes, where $n \geq 0$. Then

(i) $ECOST(T) = COST(T) + 2n$,

(ii) $EFCOST(T) = FCOST(T) + 4n^2 + 6n + 2$.

Proof: By induction on n, (ii) depends upon (i), which is, of course, a well-known result. □

The exact nature of the relationship between ERCOST (T_n) and RCOST (T_n) continues to elude us; however it is almost certainly of the form ERCOST $(T_n) \leq$ RCOST $(T_n) + f(n)$, where $f(n)$ is some function of n.

Before considering the average case analysis of FCOST and RCOST we close the present section by stating the characterizations of minimal cost trees under both FCOST and RCOST and sketching their proofs.

We first consider FCOST minimal trees.

Let T be a binary tree. Then the *diameter of T*, denoted *diam(t)*, is defined as:

$$diam(T) = max(\{dist(u, v): u, v \text{ in } T\}).$$

A tree T of n nodes has *minimal diameter* if for all T' with n nodes $diam(T') \geq diam(T)$.

It is not too surprising that trees with minimal FCOST have minimal diameter. However this is insufficient to provide minimal FCOST. Let T be a minimal diameter tree, then it can be pictured as in Figure 3.1; where subtrees a and b have height k, c has height l, and $l = k - 1$, k or $k + 1$. The node u is termed the *centroid* of T. There can be at most two such centroids, in which case $l = k + 1$. We call a tree of height h *perfect* if it has minimal height and has 2^{h+1} leaves.

We say T is *clustered* if a and b are perfect, c is minimal height and moreover if c is not perfect then it only has frontier nodes on at most two levels, but those at distance l from the root of c are grouped (or clustered) in the rightmost (or leftmost) positions on that level. See Figure 3.2 for an example of both cases. **Lemma 3.2** *Let T be as in Figure 3.3, where T_n denotes the perfect binary tree of height $n \geq 0$, and the root of T_n has distance $t + 2$ from the root of $T_{n-1}, t \geq 0$. Adding a node to T_{n-1} in T yields T^1 and to T_n in T yields T^2. Then $FCOST(T^1) - FCOST(T^2) = (2^n - 2)(t + 1)$.*

Proof: Since we are only interested in the difference between the FCOST of T^1 and T^2, we only need to consider the contribution of the extra node in T^1 and T^2. Now this contribution to $FCOST(T^1)$ is twice:

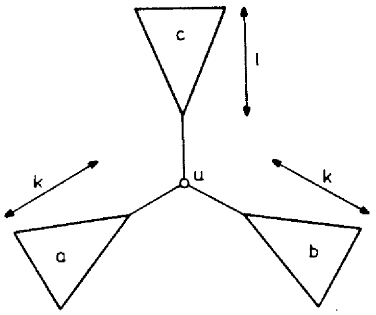


Figure 3.1

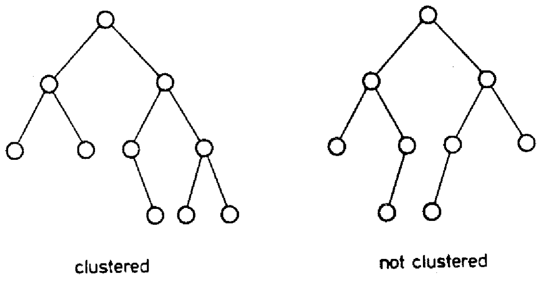


Figure 3.2

$$
\begin{aligned}
&[3x(2^1-1) + COST(T_1)] + [4.(2^2-1) + COST(T_2)] + \cdots \\
&\quad + [n(2^{n-2}-1) + COST(T_{n-2})] \\
&\quad + [(n + 1 + t).(2^n-1) + COST(T_n)] \\
&\quad + 1 + 2 + \cdots + n + t.
\end{aligned}
$$

Similarly the contribution to $FCOST(T^2)$ is twice:

$$
\begin{aligned}
&[3.(2^1-1) + COST(T_1)] + \cdots + [n.(2^{n-2}-1) + COST(T_{n-2})] \\
&\quad + [(n + 2)(2^{n-1}-1) + COST(T_{n-1})] \\
&\quad + [(n+2+t)(2^{n-1}-1) + COST(T_{n-1})] \\
&\quad + 1 + 2 + \cdots + n + t + 1 .
\end{aligned}
$$

Hence

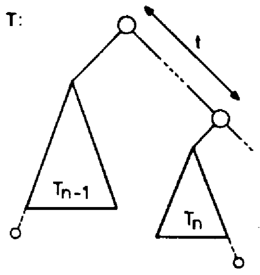


Figure 3.3

$$\begin{aligned} FCOST(T^1) - FCOST(T^2) &\\ &= 2[(n+t+1)(2^n-1) + COST(T_n)] \\ &\quad - 2[(n+1)(2^{n-1}-1) + COST(T_{n-1})] \\ &\quad - 2[(n+t+2)(2^{n-1}-1) + COST(T_{n-1})] - 2(n+t+1) \\ &= (2^n-2)(t+1) \end{aligned}$$

since

$$COST(T_n) = 2^n - 2 + 2\,COST(T_{n-1})\ . \quad \square$$

Theorem 3.3 *A binary search tree T with n internal nodes, $n \geq 0$, has minimal FCOST and EFCOST if and only if it has minimal diameter and is clustered.*

Proof: We will provide proof sketches in both cases.

If: By induction on n. Since T has minimal diameter and is clustered it has a subtree T' of $n-1$ nodes which also satisfies these conditions. Hence by the inductive assumption T' has minimal FCOST. Now T' can be viewed as in Figure 3.1. Consider the three cases height(c) = $k-1$, k, and $k+1$ separately. We will sketch the case height(c) = k only, the remaining two cases being left to the interested reader. If c is perfect then the additional node given to T' can be added anywhere. Hence assume c is not perfect. Now if a node is not added at a clustered position then consider the smallest tree S enclosing it and a clustered position. Compare the FCOST associated with these two choices. Since S is in one of the forms displayed in Figure 3.4, in both cases by invoking Lemma 3.2 we see that the chosen position is less costly than the clustered position within S. However there are greater than $(2^l - 2)$ nodes in $T' - S$ in the first case and greater than a $2(2^l - 2)$ contribution in the second case to the cost of the chosen position with respect to all of T'. Hence T must have minimal FCOST.

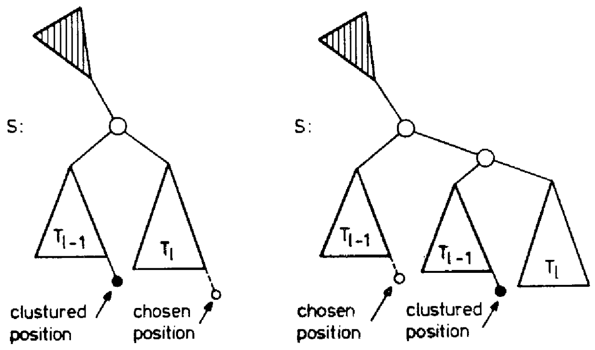


Figure 3.4

Only if: Again we prove this by induction on n. Clearly a tree with 0 or 1 node(s) satisfies the required conditions. Therefore consider a T with n nodes, $n > 1$ having minimal FCOST, but which does not satisfy the required conditions. Now if T has minimal diameter, then we can obtain a contradiction via Lemma 3.2 and the inductive assumption. Hence assume T does not have minimal diameter. Again if there is a subtree T' of T with $n - 1$ nodes having minimal diameter, we easily obtain a contradiction. Finally if there is no subtree T' of T with minimal diameter, then there exists S with $n - 1$ nodes and minimal FCOST satisfying:

$$diam(S) < diam(T') \leq diam(T).$$

Construct a tree U from S with N nodes and minimal FCOST. On the one hand if $diam(U) = diam(T)$, in which case $n = 3.2^p + 2$ for some $p \geq 0$, we obtain a contradiction either to the assumptions on T' or to the minimal FCOST of T. On the other hand if $diam(U) < diam(T)$ we may consider the largest tree V_0 which is a subtree of both U and T having minimal cost (and hence satisfying both conditions, by the inductive assumption). Then in U there is a sequence of minimal cost subtrees $U_1, \cdots, U_p$ such that $V_0 = U_1$, $U = U_p$ and the number of nodes in U_i is $i - 1$ more than in V_0. Then $FCOST(U_1) < FCOST(U_2) < \cdots < FCOST(U_p)$ and similarly there are $T_1, \ldots, T_p$ such that $T_1 = V_0$, $T_p = T$,

$FCOST(T_1) < FCOST(T_2) < \cdots < FCOST(T_p)$ and

$FCOST(U_2) < FCOST(T_2)$,

$FCOST(U_3) < FCOST(T_3)$,

.....

$FCOST(U_p) < FCOST(T_p)$,

since none of $T_2, \ldots, T_p$ satisfy the conditions of the Theorem. That is $FCOST(U) < FCOST(T)$ yielding a contradiction. □

Theorem 3.4 *A binary search tree T with n internal nodes $n \geq 0$, has minimal RCOST and ERCOST iff it has minimal COST.*

Proof: By induction on n. Clearly the proposition holds for $n = 0$ and $n = 1$. Assume it holds for all $n \leq k$, where $k \geq 1$, and consider a tree T with $n = k + 1$ internal nodes. Let u be a internal node in T with only leaves as sons. There must be at least one such node. Let T' be T with u replaced by a leaf.

We first establish the following:

Claim: Adding an internal node u to a T' leads to minimal RCOST if and only if the resulting tree is also minimal COST.

Proof of Claim: Consider T' in Figure 3.5, where $dist(\rho, u) = 1 + dist(\rho, v)$. Replace u by w to give T_u' and v by w to given T_v'. We prove that $ERCOST(T_v') < ERCOST(T_u')$.

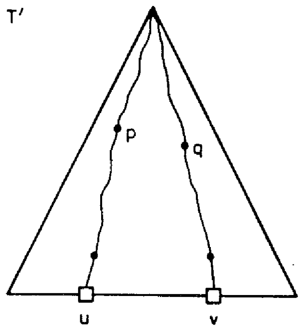


Figure 3.5

Now $ERCOST(T_u') - ERCOST(T_v') =$ difference in cost with respect to the two paths p and q in T_u' and T_v'. The contribution of all other nodes is the same in each tree. Therefore we obtain:

$$\begin{aligned} ERCOST(T_u') - ERCOST(T_v') &= 2(m+1) + 2(n-2m-1) + \\ &\quad (\textit{difference between nodes on } p \text{ and } q, \\ &\quad \textit{where } m \textit{ is the number of internal nodes on path } q.) \\ ERCOST(T_u') - ERCOST(T_v') &> 2(m+1) + 2(n-2m-1)+2m, \\ &\geq 2n > 0, \quad \textit{as desired.} \quad \square \end{aligned}$$

Returning to the proof of the Theorem if T is minimal height then T has minimal RCOST by the claim. In this case, there is a minimal height T' obtained by deleting an internal node of T. On the other hand, if T has minimal RCOST then T' obtained by deleting a node u in T must also have minimal RCOST. This follows by a similar argument to the one in the proof of the claim. $\square$

4. AVERAGE CASE ANALYSIS OF FCOST

Most families of trees of use in computer science can be generated by the iterative application of a set of *constructors* to trivial trees. Examples include planar trees, called simply trees by Knuth (1968), labelled non-planar trees, and unlabelled non-planar trees, and tournament trees together with their binary search tree counterparts.

Following Flajolet (1981) the situation can be informally described as follows: Given a family of trees F, we have for each integer $r \geq 0$ a constructor K_r: $F^r \rightarrow 2^F$, that constructs a set of trees (which in some cases consists of a single element) from an r-tuple of trees by appending a root to them and possibly reorganizing the labels. The set of trees F then satisfies the equation

$$F = \sum_{r \geq 0} K_r(F, F, \ldots F).$$

This recursive definition can, in the cases that we consider here, be translated into an equation over *multisets*. If we now consider F as a multiset of elements with multiplicity 1, there exists a set of constant $\omega_r \geq 0$ such that

$$F = \sum_{r \geq 0} \omega_r K_r(F, F, \ldots F)$$

where the equation is now an equation over multisets, and the K_r are extended to multisets by multi-linearity. In practice, $\omega_r = 1$ or $\frac{1}{r!}$. For the classes described above, it so happens that the recursive definitions can be translated into equations over *generating functions*. More precisely, let A, B, C, D... be multisets; let A_n, B_n, C_n, D_n... be the corresponding numbers of elements of *size* n, each element being counted with its multiplicity. For adequately chosen generating functions of the type

$$A(z) = \sum_{n \geq 0} \lambda_n A_n z^n$$

where the λ_n are a reference sequence of real numbers depending on the classes of trees considered (here $\lambda_n = 1$ or $\frac{1}{n!}$), the constructors K_r have *images:* if

$$E = K_r(A, B, C, \ldots),$$

then the corresponding generating functions $E(z)$, $A(z)$, $B(z)$, $C(z)$... satisfy an equation

$$E(z) = \Phi_r(A(z), B(z), C(z) \ldots)$$

for some functional Φ_r. In other words, if α is the morphism that associates to each multiset its corresponding generating function, then the diagram in Figure 4.1 commutes.

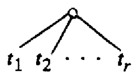
$$\begin{tikzcd}
A \arrow[d, "\alpha"] & B \arrow[d, "\alpha"] & C \arrow[d, "\alpha"] \arrow[r, "K_r"] & E \arrow[d, "\alpha"] \\
A(z) & B(z) & C(z) \arrow[r, "\Phi_r"'] & E(z)
\end{tikzcd}$$

Figure 4.1

Examples:

a) Unlabelled planar trees

For each r, $K_r(t_1, t_2, \ldots, t_r)$ is the unique tree obtained by appending a root to $t_1, t_2, \ldots, t_r$.

$t_1, t_2, \ldots, t_r \xrightarrow{K_r}$ 

The family of all planar unlabelled trees then satisfies the equation

$$G = \sum_{r \geq 0} K_r(G, G, \ldots G)$$

valid as an equation over multisets.

If we take as the size of a tree, the number of nodes in the tree, the morphism α is simply

$$\alpha(a) = A(z) = \sum_{n \geq 0} A_n z^n$$

where again A_n is the number of trees of size n in A. The relation

$$E = K_r(a_1, a_{2,\ldots,} a_r)$$

translates into

$$E(z) = zA_1(z)A_2(z)\ldots A_r(z).$$

Thus the image of the constructor K_r is nothing other than a variant of the Cauchy product.

Let $G(z)$ be the generating function associated with the family of unlabelled planar trees, then $G(z)$ satisfies the equation:

$$G(z) = z + zG(z) + zG^2(z) + zG^3(z) + \ldots,$$

that is

$$G(z) = \frac{z}{1 - G(z)}$$

whence

$$G(z) = 1 - \frac{\sqrt{1-4z}}{2} \quad \text{and} \quad G_{n+1} = \frac{1}{n+1}\binom{2n}{n}.$$

b) Binary trees

These are defined by

$$B = K_0 + K_2(B, B)$$

K_0 and K_2 being as above. It is customary to take the number of internal nodes in the tree to be the *size* of a tree. The image of K_0 is less than 1, and the equation over the corresponding generating function becomes

$$B(z) = 1 + zB^2(z).$$

whence

$$B(z) = 1 - \frac{\sqrt{1-4z}}{2z} \quad \text{and} \quad B_n = \frac{1}{n+1}\binom{2n}{n}.$$

c) Tournament trees

A tournament tree is a binary tree the internal nodes of which are labelled as consecutive integers starting from 1, in such a way that labels are to be found in increasing order along each branch. The corresponding defining equation is

$$T = L_0 + L_2(T, T),$$

Here L_0 constructs the empty tournament tree and $L_2(t_1, t_2)$, where t_1, t_2 are in T, is the set of those trees formed from t_1 and t_2 by appending a root with label 1, and by distributing labels from the set $[2..|t_1| + |t_2| + 1]$ in a manner consistent with the ordering in t_1 and t_2.

Now for a multiset A, the morphism α is

$$\alpha(A) = \sum_{n \geq 0} A_n \frac{z^n}{n!} .$$

The relation

$$E = L_2(A_1, A_2)$$

translates into

$$E(z) = \int_0^z A_1(z)A_2(z)dz,$$

and thus the image of L_2 is the integral of a Cauchy product. There are clearly $n!$ tournaments of size n, and the exponential generating function

$$\alpha(T) = T(z) = \sum_{n \geq 0} n! \frac{z^n}{n!}$$

which is equal to $\dfrac{1}{1 - z}$ satisfies the equation

$$T(z) = 1 + \int_0^z T^2(z)\, dz,$$

as is to be expected.

We can, after these preliminaries return to our main theme for which we need to consider the following three parameters for trees - whether binary trees or tournaments.

(i) The *size* of the tree is the number of its internal nodes.

(ii) The COST of a tree, which for manipulative convenience we denote by $p(t)$.

(iii) The FCOST of a tree, which, again for manipulative convenience we denote by $d(t)$.

COST has the inductive definition:

$$p\Big(\begin{array}{c}\circ\\ \diagup \;\; \diagdown\\ t_1 \quad t_2\end{array}\Big) = p(t_1) + |t_1| + p(t_2) + |t_2|,$$

and similarly for FCOST:

$$d\Big(\genfrac{}{}{0pt}{}{\overset{\circ}{\wedge}}{t_1 \quad t_2}\Big) = d(t_1) + d(t_2) + 2(p(t_1) + |t_1|)(|t_2| + 1) + 2(p(t_2) + |t_2|)(|t_1| + 1)$$

Now define the corresponding multisets for binary trees

$$SB = \sum |t| . t$$
$$PB = \sum p(t) . t$$
$$DB = \sum d(t) . t$$

where the t runs over B.

And in the same way:

$$ST = \sum |t| . t$$
$$PT = \sum p(t) . t$$
$$DT = \sum d(t) . t$$

where now t runs over T. The inductive definitions of COST and FCOST yield the equations

$$\begin{aligned} PB &= K_2(PB + SB, B) + K_2(B, PB + SB) \\ DB &= K_2(DB, B) + K_2(B, DB) + K_2(PB + SB, B + SB) \\ &\quad + K_2(B + SB, PB + SB) \\ PT &= L_2(PT + ST, T) + L_2(T, PT + ST) \\ DT &= L_2(DT, T) + L_2(T, DT) + L_2(PT + ST, T + ST) \\ &\quad + L_2(T + ST, PT + ST) . \end{aligned}$$

It now remains to translate these equations into equations over generating functions using the schemes outlined above, and then extract the Taylor coefficients that give explicit enumeration results.

The equation relative to COST translates into

$$PB(z) = 2zB(z).(PB(z) + DB(z))$$

with PB and DB the *ordinary* generating functions associated to PB and DB, namely

$$PB(z) = \sum_{n \geq 0} PB_n z^n \text{ and } DB_n = \sum_{n \geq 0} DB_n z^n .$$

The function $B(z) = \sum_{n \geq 0} B_n z^n$ is already known; as to $DB(Z)$ have:

$$DB(z) = \sum nB_n z^n = z\frac{dB(z)}{dz} = \frac{d}{dz}(zB(z)) - B(z)$$

The above equation can be solved for PB giving

$$PB(z) = \frac{2zB(z)DB(z)}{1 - 2zB(z)} = \frac{2zB(z)SB(z)}{\sqrt{(1-4z)}}$$

which makes it possible to obtain the explicit expression for PB_n. See Knuth (1968) for a different derivation.

The equation relative to FCOST reads

$$DB(z) = 2zB(z)DB(z) + 4z(B(z) + SB(z))(PB(z) + SB(z))$$

which again can be solved for $DB(z)$:

$$DB(z) = \frac{4z}{\sqrt{(1-4z)}}(B(z) + SB(z))(PB(z) + SB(z)).$$

It is to be noticed first that

$$B(z) + DB(z) = \frac{d}{dz}(zB(z)) = \frac{1}{\sqrt{1-4z}}$$

then

$$\begin{aligned} PB(z) + SB(z) &= \frac{SB(z)}{\sqrt{1-4z}}(2zB(z) + \frac{1}{\sqrt{1-4z}}) \\ &= \frac{SB(z)}{\sqrt{1-4z}} \end{aligned}$$

Putting everything together, we obtain

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

Hence with the value of the Taylor coefficients:

$$[z^n]\frac{4z}{(1-4z)^2} = n4^n$$

$$[z^n]\frac{2}{(1-4z)} = 2.4^n$$

$$[z^n]2(1-4z)^{-3/2} = (n+1)\binom{2n+2}{n+1} = 2(n+1)(2n+2)B_n,$$

the final closed form expression

$$DB_n = 4^n(n+2) - (2n+1)(2n+2)\,B_n$$

is obtained.

Now the average FCOST, assuming all B_n are equally likely, is simply

$$\frac{DB_n}{B_n}.$$

We have seen that the same equations apply to tournament trees with the labelling constructor L_2 replacing K_2. Translating into generating functions, we have

$$PT(z) = 2\int_0^z (PT(z) + ST(z))\,T(z)\,dz$$

Here, PT, ST and DT are now exponential generating functions

$$PT(z) = \sum PT_n \frac{z^n}{n!} \quad etc\ldots,$$

and

$$T(z) = \frac{1}{1-z}$$

whence

$$ST(z) = \frac{z}{(1-z)^2}$$

The equation can be solved by differentiating in z, solving the differential equation without the second term then applying the variation of constant method. We obtain:

$$PT(z) = \frac{1}{(1-z)^2}\ [2\ln(1-z) - 2z],$$

whose coefficients can be compared to the expression given by Knuth (1973). This gives rise to $DT(z)$:

$$DT(z) = 2\int_0^z DT(z)T(z)\,dz + 4\int_0^z (PT(z) + DT(z))(T(z) + DT(z))\,dz.$$

Differentiating again, and substituting values:

$$\frac{dPT(z)}{dz} = \frac{2PT(z)}{1-z} + \frac{4}{(1-z)^4}\left(2\ln\frac{1}{1-z} - z\right).$$

The equation without the second term has solution $\frac{1}{1-z^2}$ so that we set $PT(z) = \frac{u(z)}{(1-z)^2}$ and substitute in the equation:

$$\frac{du(z)}{dz} = \frac{8}{(1-z)^2}\ln\frac{1}{1-z} - \frac{4z}{(1-z)^2}$$

This can be integrated directly and one finds

$$u(z) = \frac{8}{1-z}\ln\frac{1}{1-z} + 4\ln\frac{1}{1-z} - \frac{12z}{1-z},$$

so that

$$PT(z) = \frac{8}{(l-z)^3}\ln\frac{1}{1-z} + \frac{4}{(1-z)^2}\ln\frac{1}{1-z} - \frac{12z}{(1-z)^3}$$

The simplest way to obtain the coefficients PT_n is to introduce the series

$$H(z) = \sum_{n\geq 0} H_n z^n = \frac{1}{1-z}\ln\frac{1}{1-z},$$

whose coefficients are the harmonic numbers

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n},$$

and express $PT(z)$ as a linear combination of

$$H(z), \frac{dH(z)}{dz} \text{ and } \frac{d^2H(z)}{dz^2}.$$

One easily finds

$$PT(z) = 4\frac{d^2H(z)}{dz^2} + 4\frac{dH(z)}{dz} - \frac{8(2-z)}{(1-z)^3}.$$

Extracting coefficients we obtain the closed form expression

$$\frac{PT_n}{n!} = 4(n+1)(n+3)H_n - 4n(3n+5),$$

that is the average FCOST in a tournament.

We now make use of the following equivalence principle in order to carry these results over to binary search trees, see Françon (1979) or Vuillemin (1980).

The equivalence principle: For any binary tree β,

- let $f_I(\beta)$ be the frequency of appearance of β as the "shape" of binary search trees when all $n!$ sequences of insertions of keys $1,2,\ldots n$ are performed;
- let $f_T(\beta)$ be the number of tournament trees that have β as their "shape".

Then the following equality holds

$$f_I(\beta) = f_T(\beta).$$

Thus all parameters over binary trees that are only a function of the shape of the tree can be evaluated by looking at the corresponding values for tournament trees. We have just proved:

Theorem 4.1

(i) *For binary trees, the average FCOST for trees of size n is*

$$(n+1)(n+2)\left(4^n/\binom{2n}{n}\right) - 2(n+1)(2n+1)$$

(ii) *For binary search trees the average FCOST of trees of size n is*

$$4(n+1)(n+3)H_n - 4n(3n+5).$$

5. AVERAGE CASE ANALYSIS OF RCOST

Again for manipulative convenience we denote RCOST (t) by $v(t)$ and let

$$v(t) = \sum_{q,x \text{ in } t} d(q, x, t) \tag{5.1}$$

where $d(q, x, t)$ denotes rdist (q, s) in t, where $val(s) = x$.

We are interested in the quantities

$$p_n^+ = \sum_{|t|=n} \sum_{q,x \text{ in } t} d(q, x, t) \tag{5.2}$$

$$p_n^- = \sum_{|t|=n} \sum_{q \text{ in } t} d(q, x, t) \tag{5.3}$$

where in this last summation (5.2), x ranges over all leaves of t. These quantities are such that

$$\frac{1}{n^2} p_n^+ \quad \text{and} \quad \frac{1}{n(n+1)} p_n^-$$

represent the average costs of a search with a random entry point, with a result either positive (5.2) or negative (5.3). We shall deal only with (5.2), hence we will write p_n for p_n^+ in the following.

We need to give for an inductive definition for v. As usual let t be decomposed into

$$t_1 \quad t_2 \text{ with } |t_1| = n_1 \text{ and } |t_2| = n_2\,;$$

we say that an internal node of t is of type 1 if it belongs to t_1, of type 2 if it belongs to t_2 and of type 0 if it coincides with the root, and similarly for values x.

Also let

$$v^{\alpha\beta}(t) = \sum{}^{\alpha\beta} d(q, x, t) = \sum_{\substack{type(q)=\alpha \\ type(x)=\beta}} d(q, x, t) \tag{5.4}$$

We decompose (5.1) in all possible ways

$$\Sigma = \Sigma^{00} + (\Sigma^{01} + \Sigma^{02}) + (\Sigma^{10} + \Sigma^{20}) + (\Sigma^{11} + \Sigma^{22}) + (\Sigma^{12} + \Sigma^{21}),$$

where terms have been grouped for later application of symmetries.

By definition we have

$$\Sigma^{00} d(q, x, t) = 1; \tag{5.5}$$

and we consider separately $\Sigma^{01}, \Sigma^{10}, \Sigma^{11}$ and Σ^{12}.

First for v^{01}, we find:

$$v^{01}(t) = n_1 + p(t_1) \tag{5.6}$$

A search of type 01 will first visit the root (which can happen in n_1 ways when all nodes of t_1 are searched), then proceeds as in the conventional search of t_1.

As to v^{10}, we observe that the search for the root starting from an entry point q in t_1 will result in following the rightmost branch from q and ultimately visiting the root. Thus

$$v^{10}(t) = \sum_{q \text{ in } t_1} rb(q, t_1) + n_1$$

where $rb(q, t)$ is the length; measured as the number of internal nodes; of the rightmost branch from q in t. Writing

$$w(t) = \sum_{q \text{ in } t} rb(q, t) \tag{5.7}$$

(we shall later return to w), we find

$$v^{10}(t) = w(t_1) + n_1 \tag{5.8}$$

The case of v^{11} is simple: we wish to sum over all pairs of nodes in t_1 the cost of a search; this is precisely $v(t_1)$, therefore:

$$v^{11}(t) = v(t_1). \tag{5.9}$$

Lastly we deal with v^{12}: a search from an entry point q in t_1 with an exit point in t_2 will follow the rightmost branch of q in t_1, then go through the root of t and ultimately descend in t_2. Thus:

$$v^{12}(t) = \sum_{\substack{q \text{ in } t_1 \\ x \text{ in } t_2}} (rb(q, t_1) + 1 + dist(root(t_2), x)$$

$$v^{12}(t) = n_2 w(t_1) + n_1 n_2 + n_1 p(t_2) \tag{5.10}$$

Summarizing the information gathered in (5.5)-(5.10) and using obvious symmetries we find that:

$$\begin{aligned} v(t)^1 \;=\; & 1 + [n_1 + p(t_1) + n_2 + p(t_2)] \\ & + [w(t_1) + n_1 + w(t_2) + n_2] + [v(t_1) + v(t_2)] \\ & + [n_2 w(t_1) + n_1 n_2 + n_1 p(t_2) + n_1 w(t_2) + n_1 n_2 + n_2 p(t_1)]. \end{aligned}$$

or, re-arranging the terms slightly:

$$\begin{aligned} v(t) = {} & v(t_1) + v(t_2) + 2n + 2n_1 n_2 - 1 \\ & + (n_1 + 1)p(t_2) + (n_2 + 1)p(t_1) + (n_1 + 1)w(t_2) \\ & + (n_2 + 1)\, w(t_1) \end{aligned} \tag{5.11}$$

This is the main equation. Recall that the inductive equations for p have been given in Section 4 (as well as the corresponding averages); as to w, equation (5.7) leads to

$$w(t) = w(t_1) + w(t_2) + rb(t), \tag{5.12}$$

and rb itself satisfies

$$rb(t) = rb(t_2) + 1. \tag{5.13}$$

We now translate these equations in terms of generating functions; for this we use

$$\sum_{n \ge 0} nz^n = \frac{z}{(1-z)^2} \quad \text{and} \quad \sum_{n \ge 0} (n+1)\, z^n = \frac{1}{(1-z)^2}$$

and with obvious notation, (5.11) becomes:

$$\frac{dV(z)}{dz} = \frac{2}{1-z} V(z) + \frac{2}{(1-z)^3} + \frac{2z^2}{(1-z)^4} \tag{5.14}$$
$$- \frac{1}{1-z} + \frac{2z}{(1-z)^2} P(z) + \frac{2z}{(1-z)^2} W(z) .$$

Similarly (5.12) becomes:

$$\frac{dW(z)}{dz} = \frac{2W(z)}{1-z} + RB(z), \tag{5.15}$$

and $RB(z)$ can be obtained from (5.13):

$$\frac{dRB(z)}{dz} = \frac{RB(z)}{1-z} + \frac{1}{(1-z)^2} .$$

By solving this we derive the result

$$RB(z) = \frac{1}{1-z} \log(1-z)^{-1} \tag{5.16}$$

which is equivalent to the classical fact that a tournament of size n has an average of $H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$ nodes on its rightmost branch.

Now the differential equations (5.14) and (5.15) can be solved by using (5.16) and (5.11) can be solved. We first find from (5.15) that

$$W(z) = \frac{2z}{(1-z)^2} - \frac{1}{1-z} \log(1-z)^{-1} \tag{5.17a}$$

whence

$$W_n = 2n - H_{n.} \tag{5.17b}$$

We can now solve (5.14): V is the solution of an equation of the form

$$V' = \frac{2}{1-z} V + r(z),$$

whence, by variation of constants, for some K:

$$V = \frac{1}{(1-z)^2} [K + \int_0^z r(z)(1-z)^2 dz].$$

The initial condition $V(0) = 0$ implies $K = 0$ whence

$$V(z) = \frac{1}{(1-z)^2}\int_0^z r(z)(1-z)^2 dz. \qquad (5.18)$$

For polynomials R and S, we can write

$$r(z) = \frac{R(z)}{(1-z)^4} + \frac{S(z)}{(1-z)^4}\log(1-z)^{-1}.$$

So as to avoid tedious calculation, we are content with the first two terms in the expansion of V_n. We have:

$$V(z) = \frac{R(1)}{(1-z)^2}\int_0^z \frac{dz}{(1-z)^2} + \frac{S(1)}{(1-z)^2}\int_0^z \log(1-z)^{-1}\frac{dz}{(1-z)^2} + \mathrm{O}\left(\frac{\log(1-z)^{-1}}{(1-z)^2}\right).$$

The "error" terms (valid around $z = 1$) only represent effectively computable functions which we do not wish to evaluate here. We notice that $R(1) = 4$ and $S(1) = 4$. Integrating and simplifying, we find

$$V(z) = \frac{4}{(1-z)^3}\log(1-z)^{-1} + \mathrm{O}\left(\frac{\log(1-z)^{-1}}{(1-z)^2}\right) \qquad (5.19)$$

Now the n-th Taylor coefficient of $V(z)$ is:

$$V_n = [z^n]\frac{4}{(1-z)^3}\log(1-z)^{-1} + [z^n]\,\mathrm{O}\left(\frac{\log(1-z)^{-1}}{(1-z)^2}\right)$$

and the last term can be seen to be $\mathrm{O}(n\log n)$ (this either follows from the explicit form available for this general term or from a Darboux-like theorem). Equivalently

$$V_n = [z^n]\frac{d^2}{dz^2}\left[\frac{2}{1-z}\log(1-z)^{-1} - \frac{3}{1-z}\right] + \mathrm{O}(n\log n)$$

$$= (n+1)(n+2)\,2H_{n+2} - 3(n+1)(n+2) + \mathrm{O}(n\log n),$$

so that finally we have:

Theorem 5.1 *For binary search trees, the average RCOST of trees of size n is:*

$$2(n+1)(n+2)\,H_n - 3(n+1)(n+2) + \mathrm{O}(n\log n).$$

Therefore the average cost of a (random) successful search with a random entry point is:

$$2H_n - 3 + \mathrm{O}\left(\frac{\log n}{n}\right).$$

It is to be noticed that our method can also provide a closed-form expression (as a rational combination of H_n and n) if a more precise estimate is required.

6. CONCLUDING REMARKS

The average COST for binary trees of n nodes is:

$$(n + 1)(4^n/\binom{2n}{n}) - (3n + 1) \tag{6.1}$$

and the average COST for binary search trees of n nodes is:

$$2(n + 1)\, H_n - 3n. \tag{6.2}$$

Recall that for COST these values should be divided by n to give the expected number of nodes visited in one search, and for FCOST and RCOST the corresponding values should be divided by n^2 to give the expected number of nodes visited in one search.

Comparing (6.1) divided by n with Theorem 4.1(i) divided by n^2 and comparing leading terms we find that the ratio of the average distance apart to the average distance from the root is, approximately:

$$\frac{n + 2}{n}\,. \tag{6.3}$$

In other words the average distance between nodes in a binary tree of n nodes differs by only $2/n$ from the average distance from the root.

A similar comparison of (6.2) with Theorem 4.1(ii) yields

$$\frac{2(n + 3)}{n} \tag{6.4}$$

that is the average distance between nodes in a binary search tree of n nodes is approximately twice the average distance from the root.

Finally comparing (6.2) with Theorem 5.1 in a similar manner yields:

$$\frac{n + 2}{n} \tag{6.5}$$

once more. That is the extra distance involved in a root-reset search of a binary search tree with n nodes is, approximately, $2/n$ times the average distance from the root.

Of these comparisons perhaps (6.5) is the most surprising result. However it confirms Fairley's partial results, namely a root-reset search visits at most 2 extra nodes. This follows from the observation that most nodes are within 2 levels of the leaves. Comparison (6.4) similarly reflects the observations that most nodes are close to the leaf level and, on average, $n/2$ will be found in each subtree of the root.

However the expectedness or otherwise of these results is not the focal point of this paper. Rather it is that the statistics can be analyzed using the general approach of Flajolet (1981), to provide these results. These analyses confirm the utility of Flajolet's methodology.

Let us mention some open problems before closing this paper. Two mechanical issues are the evaluation of p_n (see Section 5), and the evaluation of the average RCOST for binary trees of n nodes (c.f. Theorem 4.1(i). A nontrivial problem is the average case analysis of the diameter of a binary tree and binary search tree. Only recently, in Flajolet and Odlyzko (1982), has the average height of a binary tree of n nodes been determined. The average height of a binary search tree remains a tantalizing open problem. It appears to us that average diameter is even more difficult to determine than the average height.

REFERENCES

Beausoleil, W.F., Brown, D.T., and Phelps, B.E., Magnetic Bubble Memory Organization, *IBM Journal of Research and Development 16* (1972), 587-591.

Bongiovanni, G., and Wong, C.K., Tree Search in Major/Minor Loop Magnetic Bubble Memories, IBM Yorktown Research Center Report RC 8160, 1980.

Bonyhard, P.I., and Nelson, T.J., Dynamic Data Reallocation in Bubble Memories, *The Bell System Technical Journal 52* (1973), 307-317.

Chandra, A.K., and Wong, C.K., The Movement and Permutation of Columns in Magnetic Bubble Lattice Files, *IEEE Transactions on Computers C-27* (1979), 8-15.

Chung, K.M., Luccio, F., and Wong, C.K., A Tree Storage Scheme for Magnetic Bubble Memories, IBM Yorktown Research Center Report RC 8116, 1979.

Chung, K.M., Luccio, F., and Wong, C.K., A New Permutation Algorithm for Bubble Memories, *Information Processing Letters 10* (1980), 226-230.

Fairley, R.E., Random EnTry Searching of Binary trees, University of Colorado, Boulder, Computer Science Report CU-CS-035-73, 1973.

Flajolet, P., Analyse d'Algorithms de Manipulation d'Arbres et de Fichiers. *Cahiers du B.U.R.O.*, nos. 34-35, Paris, 1981.

Flajolet, P., and Odlyzko, A., The Average Height of Binary Trees and Other Simple Trees, *Journal of Computer and System Sciences 25* (1982), 171-213.

Françon, J., *Combinatoire des Structures de Données.* Doctoral dissertation, Université de Strasbourg, 1979.

Knuth, D.E., *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, Addison-Wesley Publishing Co., Reading, Mass., 1968.

Mühlbacher, J., private communication, 1982.

Rosenberg, A.L., Data EnCoding and Their costs, *Acta Informatica 9* (1978), 273-292.

Rosenberg, A.L., and Snyder, L., Bounds on the Costs of Data Encodings, *Mathematical Systems Theory 12* (1978), 9-39.

Standish, T.A., *Data Structure Techniques*, Addison-Wesley Publishing Co., Reading, Mass., 1980.

Vaishnavi, V.K., and Wood, D., Encoding Search Trees in Lists, *International Journal of Computer Mathematics 10* (1982), 237-246.

Vuillemin, J., A Unifying Look at Data Structures, *Communications of the ACM 23* (1980), 229-239.