UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

Minimal-Cost Brother Trees

Thomas Ottmann
D. Stott Parker
Arnold L. Rosenberg
Hans W. Six
Derick Wood

Data Structuring Group
CS-82-54

December, 1982

Minimal-Cost Brother Trees†

Thomas Ottmann[1)]

D. Stott Parker[2)]

Arnold L. Rosenberg[3)]

Hans W. Six[1)]

Derick Wood[4)]

ABSTRACT

We investigate three cost measures for the recently introduced brother search trees. In particular we characterize node visit optimal, comparison-cost optimal and space-cost optimal 1-2 brother trees and present linear-time algorithms to construct optimal 1-2 brother trees for each cost measure. Furthermore we also consider, briefly, these cost measures for brother leaf search trees.

1. INTRODUCTION

In many data processing situations we are given a large set of keys as an initial configuration. Then the set is dynamically altered by inserting new keys and deleting unwanted keys. Furthermore, member operations and other queries which do not alter the set of keys are also posed. Queries of this latter type may far exceed the others. Data structures for which an arbitrary sequence of member, insert, and delete operations can be carried out efficiently are usually called dictionaries; see Aho, Hopcroft and Ullman [1974]. It is well known that dictionaries can be implemented in such a way that all three dictionary operations can be performed in time $O(\log n)$. Various balanced tree schemes are known which may be used for this task. Among them are the AVL trees of Adelson-Velskii and Landis [1962], the 2-3 trees of Hopcroft (see Aho, Hopcroft and Ullman [1974]), the brother (leaf-search) trees of Ottmann and Six [1976], and Ottmann, Six and Wood [1978] and the 1-2 brother trees of Ottmann and

† Work partially supported by Natural Sciences and Engineering Research Council of Canada Grants Nos. A-5692 and A-7700, a grant from the Deutsche Forschungsgemeinschaft (DFG) and NSF grants Nos. IST 80-12419 and MCS 81-16522.

1) Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, D-7500 Karlsruhe, W. Germany.

2) Computer Science Department, University of California, Los Angeles, California 90024, U.S.A.

3) Computer Science Department, Duke University, Durham, North Carolina 27706, U.S.A.

4) Computer Science Department, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.

Wood [1981]. The insertion procedure for a balanced-tree scheme can also be used to handle the initialization phase in the above mentioned data processing situation: By iteratively inserting the N keys of the given initial set, beginning with the empty tree, we obtain an initial tree in time $O(N \log N)$. However, this iterative insertion method does not utilize the often valid assumption that the initial set of N keys is given in lexicographic order. Therefore, a natural question arises, namely, how to construct efficiently a balanced tree which is optimal in some sense, when given a set of keys in lexicographic order. This problem has been solved for the class of 2-3 trees by Miller, Pippenger, Rosenberg and Snyder [1979] and by Rosenberg and Snyder [1978]. The present paper addresses this question for 1-2 brother trees and brother leaf-search trees. We characterize those trees which are optimal with respect to three different cost measures, the expected number of node-visits per access, the expected number of key comparisons per access, and the space cost. Linear-time algorithms to construct optimal trees are also designed.

2. BROTHER TREES, 1-2 BROTHER TREES AND THEIR COSTS

A *brother tree* is a rooted, oriented tree each of whose nonleaf nodes has either one or two sons. Each unary node must have a binary brother. All root-to-leaf paths have the same length.

There are two basic ways of representing a set of keys as the (values of the) nodes of a tree. Either the keys are stored at internal nodes and the leaves are not used, or the keys are stored at the leaves while the internal nodes contain separating or routing values to direct queries to the correct leaf. Which is the appropriate form of representation depends on the particular application. If, for example, a sequence of insert, delete, and min operations has to be performed (that is if we want to implement a priority queue by using brother trees) storing the keys at the leaves and assigning to each internal node the minimum value stored in the subtree of that node is appropriate.

The two ways of storing sets of keys lead to the class of 1-2 brother trees on the one hand and to the class of brother (leaf-search) trees on the other: In a *1-2 brother tree* a binary node has one key and both unary nodes and leaves have no keys. All keys resident in a binary node's left subtree are strictly less than the key resident at the node; all keys resident in a binary node's right subtree are strictly greater than the key resident at the node.

In a *brother leaf-search tree* the leaves contain the keys ordered from left to right in increasing order. The internal nodes contain separating or routing information which enables us to retrieve the keys stored at the leaves. Various assignments of such separators, that is routing schemes, (cf. Kwong and Wood [1980]), are possible: we may, for example, assign to each internal node the maximum value of its sons, or we may assign to each internal node the rightmost key in the left subtree of that node.

We simply speak of *brother trees* if we are only interested in structural properties and wish to disregard the method of representing keys.

We will make frequent use of some basic notions of trees. The *depth* of a node p in a tree is its distance from the root, that is, the number of edges on the path from the root to p. The *height* of a node p is the largest distance from p to a leaf in the subtree of the tree with root p. The *height of a tree* is the height of its root.

The root of a tree is said to be at level 0; the sons of a node at level l are said to be at level $l+1$.

Given a binary tree T of height h, with leaves on level h only. The *profile* $\pi(T)$ is the integer sequence $\pi(T)=\nu_0,\ldots,\nu_h$ where ν_i is the number of nodes at level i in T. The *detailed profile* $\Delta(T)$ is the sequence of pairs

$$\Delta(T)=<\omega_0,\beta_0>,\ldots,<\omega_h,\beta_h>$$

where each ω_j (resp., β_j) denotes the number of unary (resp., binary) nodes at level j in T. Here we are setting $\beta_h=\nu_h=N+1$, and $\omega_h=0$ by convention, where $N+1$ is the number of leaves of the tree.

From these definitions we immediately obtain:

$$\nu_0=1 \tag{2.1}$$

$$\nu_h = \beta_h = 1 + \sum_{i=0}^{h-1} \beta_i = N+1, \text{ the number of leaves of } T \tag{2.2}$$

$$\nu_i = \omega_i + \beta_i \quad , \quad 0 \leq i \leq h \tag{2.3}$$

$$\nu_{i+1} = \omega_i + 2\beta_i \quad , \quad 0 \leq i < h \ . \tag{2.4}$$

Proposition Let $\Delta = <\omega_0, \beta_0>, ..., <\omega_h, \beta_h>$ be the detailed profile of a binary tree of height h with leaves on level h only. Then Δ is the detailed profile of a brother tree if and only if

$$\beta_0 = 1 \tag{2.5}$$

$$\beta_i \geq \omega_{i+1} \quad (0 \leq i < h) \tag{2.6}$$

Proof: It is clear that the detailed profile of a brother tree fulfills conditions (2.5) and (2.6). It remains to show that there exists a brother tree having detailed profile Δ, where Δ is the detailed profile of a binary tree of height h with leaves on level h only which fulfills the conditions (2.5) and (2.6). Using (2.4) we obtain for all i, $0 \leq i < h$:

$$\beta_{i+1} + \omega_{i+1} = 2\beta_i + \omega_i \geq 2 \cdot \omega_{i+1} + \omega_i \ , \quad \text{by (2.6)} \ .$$

Thus

$$\beta_{i+1} \geq \omega_{i+1} + \omega_i \geq \omega_{i+1} \ .$$

From (2.5) we obtain $\beta_0 \geq \omega_0$. Summarizing we have

$$\beta_i \geq \omega_i \quad , \quad 0 \leq i \leq h \ . \tag{2.7}$$

Thus starting with level h we can now associate to every unary node a binary one on the same level. Each of these pairs can be provided with a binary father on the next higher level because of condition (2.6). Thus we ultimately obtain a brother tree having detailed profile Δ.

In analogy with the related cost measures for 2,3-trees, (see Miller, Pippenger, Rosenberg and Snyder [1979], and Rosenberg and Snyder [1978]) we define the node-visit cost NVCOST, the comparison-cost COMPCOST, and the space-cost SPACECOST of 1-2 brother trees: Let T be a 1-2 brother tree of height h with $\pi(T) = \nu_0, ..., \nu_h$ and $\Delta(T) = <\omega_0, \beta_0>, ..., <\omega_h, \beta_h>$.

$$\begin{aligned} \text{NVCOST}(T) &= \sum_{i=0}^{h-1} (i+1) \cdot \beta_i \\ &= h \cdot \nu_h - \sum_{i=0}^{h-1} \nu_i \quad \text{(by (2.3), (2.4))} \ . \end{aligned} \tag{2.8}$$

This definition implies that 1-2 brother trees having the same (detailed) profile have the same NVCOST.

The comparison-cost is (the number of keys times) the average number of key-comparisons needed to access a key in T. Since no key-comparison is

necessary to access the only son of a unary node, we define

$$\text{COMPCOST}(T) = \sum_{p \text{ binary}} bindist(p) \tag{2.9}$$

where $bindist(p)$ denotes the number of binary nodes on the path from the root to p. Finally, let

$$\text{SPACECOST}(T) = \sum_{i=0}^{h-1} \nu_i$$

$$= \textit{number of internal nodes of } T \quad . \tag{2.10}$$

Observe that for each 1-2 brother tree T we have:

$$\text{NVCOST}(T) + \text{SPACECOST}(T) = h \cdot \nu_h$$

$$= height(T) \cdot \textit{number of leaves of } T \quad .$$

A tree is called *optimal* with respect to a certain cost measure if it has the minimum cost among all trees with the same number of keys.

In Sections 3, 4 and 5 we will characterize the 1-2 brother trees which are optimal with respect to the three different cost measures introduced above. We will use the following abbreviations throughout

NVO for *node-visit optimal,*

CCO for *comparison-cost optimal,* and

SCO for *space-cost optimal.*

Since 1-2 brother trees are "expanded" height-balanced or AVL trees (see Ottmann, Six and Wood [1979]), it should be clear that the placement of the unary nodes plays a central role in determining optimality. We will show that to a certain extent there is a duality between NVO and SCO trees, in that their only differentiating feature is the number of unary nodes. On the one hand an NVO tree has as many unary nodes as possible, so they must necessarily appear close to the leaf level of the tree. On the other hand a SCO tree has as few unary nodes as possible, so they must appear close to the root. "NVO" is to 1-2 brother trees what "bushy" is to 2-3 trees, while "SCO" is similar to the notion of "scrawny" for 2-3 trees (see Miller, Pippenger, Rosenberg and Snyder [1979]). This analogy is not complete, since the "scrawny" 1-2 brother trees or Fibonacci trees of Ottmann and Wood [1981] are not SCO. However, as we shall demonstrate "SCO" is "scrawny" under the constraint that the trees' height be minimal. We will also show that CCO trees are characterized as those 1-2 brother trees, which have at most one unary node on each root-to-leaf path.

In Section 6 we compare the three cost measures for 1-2 brother trees.

For brother leaf-search trees the above-introduced cost measures have to be modified slightly in order to take account of the fact that keys are stored only at the leaves while internal nodes contain separators. In Section 7 we briefly discuss

appropriate modifications and their difficulties, and address the question of characterizing optimal brother leaf-search trees. Finally in Section 8 we mention some open problems, and give a brief history of this paper.

We conclude this section with an example:

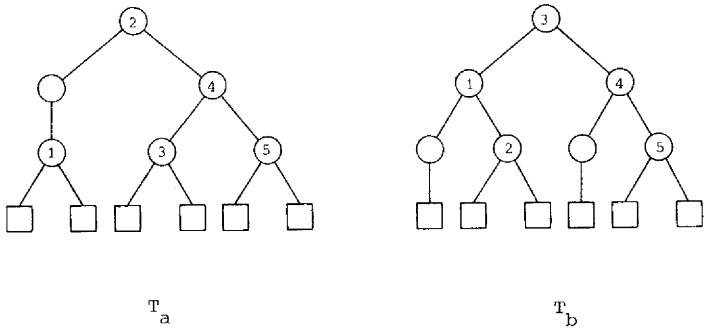


Figure 1

The trees T_a and T_b of Figure 1 are both 1-2 brother trees representing the set of keys $\{1, ..., 5\}$. Their detailed profiles are:

$$\Delta(T_a) = <0,1>, <1,1>, <0,3>, <0,6>$$
$$\Delta(T_b) = <0,1>, <0,2>, <2,2>, <0,6> \quad .$$

Their costs are:

	T_a	T_b
NVCOST	12	11
COMPCOST	11	11
SPACECOST	6	7

In fact, T_a is SCO but not NVO, and T_b is NVO but not SCO. Both trees are CCO. This example already shows that node-visit cost optimality and space cost optimality are independent of each other.

3. NODE-VISIT-OPTIMAL 1-2 BROTHER TREES

We characterize NVO 1-2 brother trees and design a linear-time algorithm to construct them. The first result gives a necessary condition for 1-2 brother trees to be NVO, namely, height-minimality.

Lemma 3.1: Let T and T' be 1-2 brother trees both having the same number of leaves (that is containing the same number of keys). Let T (respectively, T') be of height h (respectively, h'). If $h < h'$ then $\mathrm{NVCOST}(T) < \mathrm{NVCOST}(T')$.

Proof: Let $\pi(T) = \nu_0, \ldots, \nu_h$, $\pi(T') = \nu'_0, \ldots, \nu'_{h'}$, where $\nu_h = \nu'_{h'}$, $h < h'$. Then by (2.8)

$$\begin{aligned}
\mathrm{NVCOST}(T') - \mathrm{NVCOST}(T) &= h'\nu'_{h'} - \sum_{i=0}^{h'-1} \nu'_i - (h \cdot \nu_h - \sum_{i=0}^{h-1} \nu_i) \\
&= \nu'_{h'}(h' - h) - \sum_{i=0}^{h'-1} \nu'_i + \sum_{i=0}^{h-1} \nu_i \quad ; \text{ since } \nu_h = \nu'_{h'} \\
&\geq \nu'_{h'} + \sum_{i=0}^{h-1} \nu_i - \sum_{i=0}^{h'-1} \nu'_i \quad ; \text{ since } h' > h \quad .
\end{aligned}$$

By (2.2) both trees have the same number of binary nodes. This implies

$$\sum_{i=0}^{h-1} \nu_i - \sum_{i=0}^{h'-1} \nu'_i \geq \sum_{i=0}^{h-1} \beta_i - \sum_{i=0}^{h'-1} \nu'_i = -\sum_{i=0}^{h'-1} \omega'_i \quad .$$

Furthermore, the number of unary nodes of a 1-2 brother tree is always strictly less than the number of binary nodes in the tree, since each unary node must have a binary father and brother. This implies

$$-\sum_{i=0}^{h'-1} \omega'_i > -\sum_{i=0}^{h'-1} \beta'_i = -(\nu'_{h'} - 1) \quad .$$

Therefore we have $\mathrm{NVCOST}(T') - \mathrm{NVCOST}(T) \geq 1$.

The trees of Figure 1 (a), (b) illustrate that height minimality is not sufficient for NV-optimality of a 1-2 brother tree.

There does not appear to be a 1-2 brother tree correspondent to the 2, 3-tree notion of "dense profile" as in Miller, Pippenger, Rosenberg and Snyder [1979], with which to characterize NVO 1-2 brother trees. We demonstrate that the number of nodes at each level of a NVO tree is not completely specified by the number of nodes on the level below. Thus one cannot hope for a simple modification of the notion of denseness for 2-3 trees, namely $\nu_l = \min(3^l, \lfloor \nu_{l+1}/2 \rfloor)$ for $1 \leq l \leq h-1$, to something like $\nu_l = \min(2^l, \lfloor 2/3\, \nu_{l+1} \rfloor)$, $1 \leq l \leq h-1$, for 1-2 brother trees. We argue as follows.

We consider for example 1-2 brother trees with $N+1 = 18 \cdot 2^{h-5}$ leaves of height h. The following two detailed profiles are both profiles of equally costly 1-2 brother trees:

$$\Delta = <0, 2^0>, ..., <0, 2^{h-4}>, <2^{h-5}, 3 \cdot 2^{h-5}>, <3 \cdot 2^{h-5}, 4 \cdot 2^{h-5}>,$$
$$<4 \cdot 2^{h-5}, 7 \cdot 2^{h-5}>, <0, N+1>$$
$$\Delta' = <0, 2^0>, ..., <0, 2^{h-4}>, <2^{h-4}, 2^{h-4}>, <0, 3 \cdot 2^{h-4}>,$$
$$<3 \cdot 2^{h-4}, 3 \cdot 2^{h-4}>, <0, N+1>$$

Lemma 3.2 given below will show that both are profiles of NVO 1-2 brother trees. However, the numbers of nodes ν_{h-1} and ν'_{h-1} on level $h-1$ are different:

$$\nu_{h-1} = 4 \cdot 2^{h-5} + 7 \cdot 2^{h-5} = \frac{11}{18}(N+1)$$
$$\nu'_{h-1} = 3 \cdot 2^{h-4} + 3 \cdot 2^{h-4} = \frac{12}{18}(N+1)$$

This shows that for NVO 1-2 brother trees the number of nodes on level $h-1$ is not uniquely determined by the number of nodes $\nu_h = N+1$ on level h. Similar arguments carry over to level $h-2$ and to other numbers of leaves.

In order to characterize profiles of NVO 1-2 brother trees each three adjacent levels have to be considered:

Lemma 3.2: Let $\Delta = <\omega_0, \beta_0>, ..., <\omega_h, \beta_h>$ be the detailed profile of a NVO 1-2 brother tree of height h. Then for each k, $3 \leq k \leq h$, at least one of the following conditions must hold:

$$\omega_{k-2} = 0 \tag{3.1}$$
$$\beta_{k-2} = \omega_{k-1} \tag{3.2}$$
$$\beta_{k-1} = \omega_k \quad . \tag{3.3}$$

Proof: Assume the contrary, that is there exists a NVO 1-2 brother tree of height h and a k, $3 \leq k \leq h$, such that none of the conditions (3.1), (3.2), (3.3) is met. We show that there exists another 1-2 brother tree representing the same number of keys with the detailed profile

$$\Delta' = <\omega_0, \beta_0>, ..., <\omega_{k-2}-1, \beta_{k-2}+1>, <\omega_{k-1}+2, \beta_{k-1}-1>,$$
$$<\omega_k, \beta_k>, ..., <\omega_h, \beta_h>$$

which has better NV-cost. First note that $\omega_{k-2}-1 \geq 0$ because (3.1) does not hold by assumption. Furthermore, $\beta_{k-1} \geq \omega_k \geq 0$, because (2.6) holds for the given detailed profile Δ. Thus the assumption $\beta_{k-1} \neq \omega_k$ implies $\beta_{k-1} > \omega_k$, hence $\beta_{k-1}-1 \geq 0$. Therefore we know that all numbers occurring in Δ' are nonnegative. It remains to show only that Δ' obeys the constraints (2.4), (2.5), (2.6). In order to show (2.4) it suffices to prove

$$2\beta_{k-3} + \omega_{k-3} = (\beta_{k-2}+1) + (\omega_{k-2}-1) \ ,$$

$$2(\beta_{k-2}+1)+(\omega_{k-2}-1)=(\beta_{k-1}-1)+(\omega_{k-1}+2)\ ,$$

and

$$2(\beta_{k-1}-1)+(\omega_{k-1}+2)=\beta_k+\omega_k\ .$$

These equations are obtained trivially from the fact that condition (2.4) must hold for the given detailed profile Δ. Similarly, condition (2.5) must hold for Δ' because it holds for Δ.

Finally, proving (2.6) for Δ' reduces to showing the inequality $\beta_{k-2}+1\geq\omega_{k-1}+2$. It can be obtained from the assumption $\beta_{k-2}\neq\omega_{k-1}$ and from the fact that (2.6) holds for Δ.

Corollary 3.3: Let $\Delta=<\omega_0,\beta_0>,\ldots,<\omega_h,\beta_h>$ be the detailed profile of a NVO 1-2 brother tree. If there exists an l , $0<l<h$, such that $\beta_l>\omega_{l+1}$ and $\beta_{l-1}>\omega_l$ then for all l' , $0\leq l'\leq l-1$, $\omega_{l'}=0$ must hold.

Proof: Let us assume that $\beta_l>\omega_{l+1}$ and $\beta_{l-1}>\omega_l$ for an l, $0<l<h$.

We will show that $\omega_{l'}=0$ must hold for all l' , $0\leq l'\leq l-1$, by induction on l' . First the application of Lemma 3.2 for $k=l+1$ yields $\omega_{l-1}=0$ or $\beta_{l-1}=\omega_l$ or $\beta_l=\omega_{l+1}$. Hence, our assumptions imply that $\omega_{l-1}=0$ must hold. Next assume that $\omega_k=0$ for all k, $l'\leq k\leq l-1$. Using condition (2.6) this in particular implies that $\beta_{l'-1}>\omega_{l'}=0$ and $\beta_{l'}>\omega_{l'+1}=0$ must hold. Again applying Lemma 3.2 for $k=l'-1$ yields $\omega_{l'-1}=0$.

This corollary in particular implies that a NVO 1-2 brother tree must be complete binary up to level l (that is $\omega_0=\omega_1=\cdots=\omega_l=0$), if $\omega_l=\omega_{l-1}=0$.

Proposition 3.4: Let T be a 1-2 brother tree with $N+1$ leaves and detailed profile $\Delta=<\omega_0,\beta_0>,\ldots,<\omega_h,\beta_h>$. If $\beta_{h-2}=\omega_{h-1}$ then $N+1=3\beta_{h-2}+2\omega_{h-2}$.

Proof: We have $N+1=2\beta_{h-1}+\omega_{h-1}=2\beta_{h-1}+\beta_{h-2}$. However

$$2\beta_{h-2}+\omega_{h-2}=\beta_{h-1}+\omega_{h-1}=\beta_{h-1}+\beta_{h-2}\ ,$$

so

$$\beta_{h-2}+\omega_{h-2}=\beta_{h-1}\ .$$

Thus

$$N+1=2(\beta_{h-2}+\omega_{h-2})+\beta_{h-2}=3\beta_{h-2}+2\omega_{h-2}\ .$$

The next theorem constitutes one part of our result characterizing NVO 1-2 brother trees.

Theorem 3.5: Let T be a 1-2 brother tree with N keys and minimal possible height $h = \lceil \log_2(N+1) \rceil$, where $3 \cdot 2^{h-2} \leq N+1 \leq 2^h$. Then T is NVO if and only if its detailed profile $\Delta = <\omega_0, \beta_0>, \ldots, <\omega_h, \beta_h>$ satisfies

(i) $\beta_k = 2^k$, $\omega_k = 0$ for $0 \leq k \leq h-2$

(ii) $\beta_{h-1} = (N+1) - 2^{h-1}$, and $\omega_{h-1} = 2^h - (N+1)$.

Proof: Because NVCOST is a function of the (detailed) profile of 1-2 brother trees it suffices to show the "only if" part. Let T be a NVO 1-2 brother tree with detailed profile Δ. We claim first that $\omega_{h-2} = 0$. If not, Lemma 3.2 tells us that $\beta_{h-2} = \omega_{h-1}$ (since otherwise $\beta_{h-1} = \omega_h = 0$, which is impossible).

By Proposition 3.4 we obtain

$$\begin{aligned} N+1 &= 3\beta_{h-2} + 2\omega_{h-2} \\ &\leq 3 \cdot (2^{h-2} - \omega_{h-2}) + 2\omega_{h-2} \\ &< 3 \cdot 2^{h-2} \end{aligned}$$

a contradiction.

We claim second that $\omega_{h-3} = 0$. If not we again find by Lemma 3.2 that $\beta_{h-2} = \omega_{h-1}$, since we now know that $\omega_{h-2} = 0$. But then,

$$\begin{aligned} N+1 &= 3\beta_{h-2} \\ &= 2(2\beta_{h-3} + \omega_{h-3}) \\ &\leq 3(2(2^{h-3} - \omega_{h-3}) + \omega_{h-3}) \\ &< 3 \cdot 2^{h-2} \quad , \text{ also a contradiction} . \end{aligned}$$

From $\omega_{h-2} = 0$ and $\omega_{h-3} = 0$ we obtain $\beta_{h-3} > \omega_{h-2}$ and $\beta_{h-4} > \omega_{h-3}$. Now Corollary 3.3 tells us that $\omega_l = 0$ for all $l \leq h-4$. The theorem is thus complete except for β_{h-1} and ω_{h-1}. Since $\beta_{h-2} = 2^{h-2}$ we find

$$\beta_{h-1} + \omega_{h-1} = 2\beta_{h-2} = 2^{h-1}$$

and of course

$$2\beta_{h-1} + \omega_{h-1} = N+1 \quad .$$

These two equations form a linear system in β_{h-1} and ω_{h-1} which when solved gives the results stated in the theorem.

Before we prove the second half of our result characterizing NVO 1-2 brother trees we show the following Lemma:

Lemma 3.6: Let T be a 1-2 brother tree with N keys and minimal possible height $h = \lceil \log_2(N+1) \rceil \geq 5$ where $2^{h-1} < N+1 < 3 \cdot 2^{h-2}$. If T is NVO then T is a complete binary tree up to level $h-4$.

Proof: Let $\Delta = <\omega_0, \beta_0>, ..., <\omega_h, \beta_h>$ be the detailed profile of a NVO 1-2 brother tree T with height h, where h obeys the assumptions of the Lemma.

Claim 1: $\omega_{h-4} = 0$.

If not Lemma 3.2 tells us that $\beta_{h-4} = \omega_{h-3}$ or $\beta_{h-3} = \omega_{h-2}$.

Case 1: $\beta_{h-4} = \omega_{h-3}$ and $\beta_{h-3} \neq \omega_{h-2}$ (that is $\beta_{h-3} > \omega_{h-2}$).

We will show that there exists a 1-2 brother tree T' representing the same number of keys with detailed profile

$$\Delta' = <\omega_0, \beta_0>, ..., <\omega_{h-5}, \beta_{h-5}>, <\omega_{h-4}-1, \beta_{h-4}+1>, <\omega_{h-3}+1, \beta_{h-3}>,$$
$$<\omega_{h-2}+3, \beta_{h-3}-2>, <\omega_{h-1}-2, \beta_{h-2}+1>, <\omega_h, \beta_h>$$

which has better NVCOST. Application of Lemma 3.2 for $k = h-1$ and the assumptions $\omega_{h-3} = \beta_{h-4} \neq 0$ and $\beta_{h-3} \neq \omega_{h-2}$ yield $\beta_{h-2} = \omega_{h-1}$. This equation leads to $N+1 = 3\beta_{h-2} + 2\omega_{h-2}$ by Proposition 3.4.

Because $\beta_{h-2} \geq \omega_{h-2}$ the assumptions $\beta_{h-2} \leq 3$ and $h \geq 5$ lead to $N+1 < 17$, a contradiction. Thus we know that all numbers occurring in Δ' are nonnegative. It is easy to show that Δ' is the profile of a binary tree. Thus it remains to prove that Δ' is the detailed profile of a 1-2 brother tree. We need to show only that condition (2.6) holds for Δ'. The only nontrivial case is the proof of $\beta_{h-3} \geq \omega_{h-3} + 3$.

First we have

$$\beta_{h-3} + \omega_{h-3} \leq 2^{h-3} - \omega_{h-4} \quad ,$$

so

$$\beta_{h-2} + \omega_{h-2} = 2\beta_{h-3} + \omega_{h-3} \leq 2^{h-3} - \omega_{h-4} + \beta_{h-3} \quad .$$

From Proposition 3.4 we infer

$$\beta_{h-2} = \frac{1}{3}(N+1) - \frac{2}{3}\omega_{h-2} \quad .$$

Thus we obtain

$$\frac{1}{3}(N+1) - \frac{2}{3}\omega_{h-2} + \omega_{h-2} \leq 2^{h-3} - \omega_{h-4} + \beta_{h-3} \quad .$$

This is equivalent to

$$\beta_{h-3} \geq \frac{1}{3}\omega_{h-2} + \omega_{h-4} + \frac{1}{3}(N+1) - 2^{h-3}$$
$$> \frac{1}{3}\omega_{h-2} + \omega_{h-4} + \frac{1}{3}2^{h-1} - 2^{h-3} \quad ,$$

so

$$\beta_{h-3} > \frac{1}{3}\omega_{h-2} + \omega_{h-4} + \frac{1}{3}2^{h-3} \quad . \tag{3.4}$$

On the other hand we have

$$\begin{aligned}\beta_{h-3} + \omega_{h-3} &= 2\beta_{h-4} + \omega_{h-4} \\ &= \omega_{h-3} + \beta_{h-4} + \omega_{h-4} \quad ;\end{aligned}$$

by the first assumption of Case 1. Hence

$$\beta_{h-3} = \beta_{h-4} + \omega_{h-4} \leq 2^{h-4} \quad .$$

Now using the second assumption of Case 1 we obtain $\omega_{h-2} < 2^{h-4}$. Then

$$2\omega_{h-2} < 2^{h-3}$$

$$3\omega_{h-2} < 2^{h-3} + \omega_{h-2}$$

$$\omega_{h-2} < \frac{1}{3}2^{h-3} + \frac{1}{3}\omega_{h-2}$$

$$\frac{1}{3}2^{h-3} + \frac{1}{3}\omega_{h-2} + \omega_{h-4} > \omega_{h-2} + \omega_{h-4} \quad .$$

From the last inequality and (3.4) we obtain

$$\beta_{h-3} \geq \omega_{h-2} + \omega_{h-4} + 2 \geq \omega_{h-2} + 3 \quad .$$

This completes Case 1.

Case 2: $\beta_{h-3} = \omega_{h-2}$

We will show that there exists a 1-2 brother tree T' representing the same number of keys with detailed profile

$$\Delta' = <\omega_0, \beta_0>, \ldots, <\omega_{h-5}, \beta_{h-5}>, <\omega_{h-4}-1, \beta_{h-4}+1>,$$
$$<\omega_{h-3}+3, \beta_{h-3}-2>, <\omega_{h-2}-3, \beta_{h-2}+2>, <\omega_{h-1}+2, \beta_{h-1}-1>, <\omega_h, \beta_h>$$

which has better NV-Cost.

Clearly, $\beta_{h-1} > 0$ and thus $\beta_{h-1} \neq \omega_h$ ($\omega_h = 0$ by convention). Furthermore, the assumption $\beta_{h-3} = \omega_{h-2}$ implies $\omega_{h-2} \neq 0$. Thus, application of Lemma 3.2 for $k = h$ yields $\beta_{h-2} = \omega_{h-1}$. From this equation we obtain by Proposition 3.4

$$N+1 = 3\beta_{h-2} + 2\omega_{h-2} \quad .$$

In order to show that all numbers occurring in Δ' are nonnegative it remains to show that $\beta_{h-3} = \omega_{h-2} \geq 3$. We know

$$2\beta_{h-3} + \omega_{h-3} = \beta_{h-2} + \omega_{h-2} = \beta_{h-2} + \beta_{h-3} \quad ,$$

so

$$\beta_{h-2} = \beta_{h-3} + \omega_{h-3} \quad .$$

Again using Proposition 3.4 the last equation and the assumption of Case 2 imply:

$$\begin{aligned} N+1 &= 3\beta_{h-2}+2\omega_{h-2} \\ &= 3(\beta_{h-3}+\omega_{h-3})+2\beta_{h-3} \\ &= 5\beta_{h-3}+3\omega_{h-3} \quad . \end{aligned} \tag{3.5}$$

On the other hand

$$\beta_{h-3}+\omega_{h-3} \leq 2^{h-3}-\omega_{h-4} \ ,$$

so

$$\omega_{h-3} \leq 2^{h-3}-\omega_{h-4}-\beta_{h-3} \quad . \tag{3.6}$$

From (3.5) and (3.6) we obtain

$$N+1 \leq 2\beta_{h-3}+2^{h-1}-2^{h-3}-3\omega_{h-4} \ .$$

Using $2^{h-1} < N+1$ we obtain

$$2^{h-4}+\frac{3}{2}\omega_{h-4} < \beta_{h-3} \quad . \tag{3.7}$$

Since we have assumed $\omega_{h-4} > 0$ we know that $\beta_{h-3} \geq 3$.

It is easy to show that Δ' is the detailed profile of a binary tree. Thus it remains to prove that Δ' is the detailed profile of a 1-2 brother tree. For this we need to show only that condition (2.6) holds for Δ'. The only nontrivial case is the proof of $\beta_{h-4}+1 \geq \omega_{h-3}+3$.

Clearly, $\beta_{h-4}+\omega_{h-4} \leq 2^{h-4}$. Using (3.7) we obtain

$$\beta_{h-4}+\frac{5}{2}\omega_{h-4} < \beta_{h-3} \quad .$$

On the other hand

$$2\beta_{h-4}+\omega_{h-4} = \beta_{h-3}+\omega_{h-3} \ ,$$

so

$$2\beta_{h-4}+\omega_{h-4} > \beta_{h-4}+\frac{5}{2}\omega_{h-4}+\omega_{h-3} \quad .$$

Therefore

$$\beta_{h-4} > \omega_{h-3}+\frac{3}{2}\omega_{h-4} \quad .$$

Since $\omega_{h-4} > 0$ this implies $\beta_{h-4} \geq \omega_{h-3}+2$, completing the proof of Case 2 and of our first claim $\omega_{h-4}=0$.

Claim 2: $\omega_{h-5}=0$.

If not, $h > 5$ must hold and Lemma 3.2 tells us that $\beta_{h-4}=\omega_{h-3}$ (because

$\beta_{h-5} > \omega_{h-4} = 0$, by our first claim).

Again applying Lemma 3.2 for $k = h-1$ we have

$$\omega_{h-3} = 0 \quad \text{or} \quad \beta_{h-3} = \omega_{h-2} \quad \text{or} \quad \beta_{h-2} = \omega_{h-1} \quad . \tag{3.8}$$

Since $\omega_{h-4} = 0$ (by Claim 1) and $\omega_{h-5} > 0$ (by assumption), Corollary 3.3 tells us that $\omega_{h-3} > 0$.

We now argue that $\beta_{h-2} = \omega_{h-1}$ is true. If not, (3.8) tells us that $\beta_{h-3} = \omega_{h-2}$. Again applying Lemma 3.2 for $k = h$ we can infer from $\beta_{h-3} = \omega_{h-2}$ that $\beta_{h-2} = \omega_{h-1}$ - a contradiction.

We will now show that there exists a 1-2 brother tree T' representing the same number of keys with detailed profile

$$\Delta' = <\omega_0, \beta_0>, \ldots, <\omega_{h-6}, \beta_{h-6}>, <\omega_{h-5}-1, \beta_{h-5}+1>, <\omega_{h-4}, \beta_{h-4}+1>,$$
$$<\omega_{h-3}+1, \beta_{h-3}+1>, <\omega_{h-2}+9, \beta_{h-2}-6>, <\omega_{h-1}-6, \beta_{h-1}+3>, <\omega_h, \beta_h>$$

which has better NVCOST. First we conclude from $\beta_{h-2} = \omega_{h-1}$ that

$$\begin{aligned} 3\beta_{h-2} + 2\omega_{h-2} &= N+1 \; ; \quad \text{by Proposition 3.4} \\ &> 2^{h-1} \; ; \quad \text{by assumption ,} \end{aligned}$$

so

$$5\beta_{h-2} > 2^{h-1} \; ; \qquad \text{by (2.7)}$$

$$\beta_{h-2} > \frac{1}{5} 2^{h-1} \geq 6 \; ; \text{because} \quad h > 5 .$$

This shows that all numbers occurring in the detailed profile Δ' are nonnegative. In order to show that Δ' is the detailed profile of a 1-2 brother tree, it is sufficient to prove $\beta_{h-3} + 1 \geq \omega_{h-2} + 9$. We briefly recall what we know about the detailed profile Δ': $\omega_{h-5} > 0$, $\omega_{h-4} = 0$, $\beta_{h-4} = \omega_{h-3}$, $\beta_{h-2} = \omega_{h-1}$. As consequence we obtain furthermore: $\beta_{h-3} = \omega_{h-3} = \beta_{h-4}$, since $2\beta_{h-4} + \omega_{h-4} = \beta_{h-3} + \omega_{h-3}$. In order to get an upper bound for ω_{h-2}, the number of unary nodes on level $h-2$, we first show that there must be considerably more binary than unary nodes on level $h-2$:

$$\begin{aligned} \beta_{h-2} + \omega_{h-2} &= 2\beta_{h-3} + \omega_{h-3} = 3\beta_{h-4} \\ &= 2\beta_{h-4} + \beta_{h-3} \\ &\geq 2\beta_{h-4} + \omega_{h-2} \; ; \quad \text{by (2.6)} \end{aligned}$$

so, $\beta_{h-2} \geq 2\beta_{h-4}$ and $\omega_{h-2} \leq \beta_{h-4}$. Let $\beta_{h-2} = 2\omega_{h-2} + x$, where $x \geq 0$. Then

$$N+1 = 3\beta_{h-2} + 2\omega_{h-2} = 8\omega_{h-2} + 3x$$

that is $x = \frac{1}{3}(N+1) - \frac{8}{3}\omega_{h-2}$. Now

$$\begin{aligned} \beta_{h-2} + \omega_{h-2} &= 2\omega_{h-2} + x + \omega_{h-2} \\ &= \frac{1}{3}\omega_{h-2} + \frac{1}{3}(N+1) = 3\beta_{h-4} \; ; \end{aligned}$$

hence

$$\begin{aligned}\omega_{h-2}+2^{h-1} &< 9\beta_{h-4}\\ &\leq 8(2^{h-4}-\omega_{h-5})+\beta_{h-4}\\ &= \beta_{h-4}+2^{h-1}-8\omega_{h-5}\ .\end{aligned}$$

Since $\beta_{h-4}=\beta_{h-3}$ we obtain $\beta_{h-3}>\omega_{h-2}+8\omega_{h-5}\geq\omega_{h-2}+8$. This concludes the proof of our second claim.

Application of Corollary 3.3 yields the result stated in the Lemma.

We are now ready for the second half of our result characterizing NVO 1-2 brother trees.

Theorem 3.7: Let T be a 1-2 brother tree with N keys and minimal possible height $h=\lceil\log_2(N+1)\rceil$, where $2^{h-1}<N+1<3\cdot2^{h-2}$. Then T is NVO if and only if its detailed profile $\Delta=<\omega_0,\beta_0>,\ldots,<\omega_h,\beta_h>$ satisfies

(i) $\beta_k=2^k$, $\omega_k=0$ for $0\leq k\leq h-4$ and

(ii) $\beta_{h-2}=\omega_{h-1}$.

Proof: Let T be a NVO 1-2 brother tree with N keys and height $h=\lceil\log_2(N+1)\rceil$, where $2^{h-1}<N+1<3\cdot2^{h-2}$. Clearly, (i) holds because of Lemma 3.6. Application of Lemma 3.2 for $k=h$ yields $\omega_{h-2}=0$ or $\beta_{h-2}=\omega_{h-1}$ or $\beta_{h-1}=\omega_h$. The last alternative is false, because $\omega_h=0$ by definition.

We show that $\beta_{h-2}=\omega_{h-1}$. If not, then $\omega_{h-2}=0$ must hold. Application of Lemma 3.2 for $k=h-1$ yields $\omega_{h-3}=0$ or $\beta_{h-3}=\omega_{h-2}$ or $\beta_{h-2}=\omega_{h-1}$. Clearly, the first alternative must hold, because $\beta_{h-2}\neq\omega_{h-1}$ by assumption and $\beta_{h-3}\neq\omega_{h-2}=0$.

Thus we know that T must be complete binary up to level $h-2$. This implies $N+1\geq3\cdot2^{h-2}$ - a contradiction. This completes the proof of (ii).

In order to show the "only if" part assume that T is a 1-2 brother tree with minimal possible height h whose detailed profile Δ fulfills (i) and (ii). The equation $\beta_{h-2}=\omega_{h-1}$ implies (cf. Proposition 3.4) $(N+1)=3\beta_{h-2}+2\omega_{h-2}$ and $\beta_{h-1}=\beta_{h-2}+\omega_{h-2}$. Hence, we infer:

$$\begin{aligned}(\omega_{h-2}+\beta_{h-2})+(\omega_{h-1}+\beta_{h-1}) &= (\omega_{h-2}+\beta_{h-2})+(\beta_{h-2}+\beta_{h-2}+\omega_{h-2})\\ &= 3\beta_{h-2}+2\omega_{h-2}\\ &= N+1\ .\end{aligned}$$

Now we use the fact that T's NVCOST depends only on T's profile, that is on the numbers $\nu_k=\omega_k+\beta_k$, for $k=0,\ldots,h$. Thus, we obtain from (2.8):

$$\mathrm{NVCOST}(T)=h\cdot\nu_h-\sum_{k=0}^{h-1}\nu_k$$

$$= h(N+1) - \sum_{k=0}^{h-4} 2^k - [(\omega_{h-3} + \beta_{h-3}) + (\omega_{h-2} + \beta_{h-2}) + (\omega_{h-1} + \beta_{h-1})]$$

$$= h(N+1) - (2^{h-3} - 1) - (2^{h-3} + N + 1)$$

$$= (h-1)(N+1) - 2^{h-2} + 1 \quad .$$

This shows that the NVCOST of a 1-2 brother tree T with minimal possible height h whose detailed profile fulfils (i) and (ii) depends only on the number of leaves $N+1$. Now the "if" part of the Theorem tells us that T must be NVO.

An Algorithm for Constructing NVO 1-2 Brother Trees

Our characterization of NVO 1-2 brother trees directly leads to a linear-time algorithm for constructing such a tree T for a given sorted list of N keys. It should be clear that it suffices to construct a detailed profile of a NVO 1-2 brother tree. The construction in linear time of the skeleton of a brother tree from the given detailed profile is straightforward. The skeleton is then filled with the sorted list of N keys by traversing T in inorder and depositing the next key whenever a binary node is visited.

Procedure NVO-Profile

Input: Natural number N (of keys to be stored)

Output The detailed profile of a NVO 1-2 brother tree with N binary nodes and $N+1$ leaves.

```
begin
    h:=⌈log2(N+1)⌉;
    if N+1≥3·2^(h-2)
    then Δ=<0,2^0>,...,<0,2^(h-2)>,<2^h-(N+1),(N+1)-2^(h-1)>,<0,N+1>
    else {2^(h-1)<N+1<3·2^(h-2)}
    begin
        determine i, 0≤i≤3, such that
        (8+i)·2^(h-4)≤(N+1)<(9+i)2^(h-4);
        x := (9+i)·2^(h-4)-(N+1);
        y := (N+1)-(8+i)·2^(h-4);
        {note that(N+1)=(8+i)·x+(9+i)·y}
        case i of
        0:Δ:=<0,2^0>,...,<0,2^(h-4)>,<2^(h-4),2^(h-4)>,
              <x,2x+3y>,<2x+3y,3x+3y>,<0,N+1>;
        1:Δ:=<0,2^0>,...,<0,2^(h-4)>,<x,x+2y>,
              <2y,3x+2y>,<3x+2y,3x+4y>,<0,N+1>;
        2:Δ:=<0,2^0>,...,<0,2^(h-4)>,<0,2^(h-3)>,
              <2x+y,2x+3y>,<2x+3y,4x+4y>,<0,N+1>;
        3:Δ:=<0,2^0>,...,<0,2^(h-4)>,<0,2^(h-3)>,
              <x,3x+4y>,<3x+4y,4x+4y>,<0,N+1>
        end
    end
end.
```

It is easy to check that the above algorithm generates optimal profiles of 1-2 brother trees. These profiles are not necessarily unique.

4. SPACE OPTIMAL 1-2 BROTHER TREES

In this section we characterize those 1-2 brother trees which have minimum storage requirement among all 1-2 brother trees with the same number of keys (or, equivalently, of leaves). It turns out that the trees with the minimal number of nodes must have also minimal height, but not conversely. All 1-2 brother trees with the same number of keys (or leaves) have the same number of binary nodes. This means that a 1-2 brother tree has minimal SPACECOST if and only if it has the minimal number of unary nodes.

Our first observation concerning SCO 1-2 brother trees is that the total number of nodes strictly increases with the number of stored keys

Lemma 4.1: Let T_N and T_{N+1} be two SCO 1-2 brother trees with N and $N+1$ leaves respectively, (or, equivalently $N-1$ and N keys, respectively). Then the total number of nodes in T_N is strictly less than the total number of nodes in T_{N+1}.

Proof: Consider the SCO 1-2 brother tree T_{N+1}. Remove one of its keys (and leaves) by performing the *delete* procedure of Ottmann and Wood [1981]. This procedure restructures the tree in order to retain the brother tree structure. Inspection of this procedure shows that the total number of nodes (in T_{N+1}) is decreased. Let T_N' denote the resulting 1-2 brother tree with N leaves. Then we have:

$$\text{SPACECOST}(T_N) \leq \text{SPACECOST}(T_N') < \text{SPACECOST}(T_{N+1})$$

since we have removed (at least) one node from T_{N+1}.

Let a 1-2 brother tree T_{N+1} with $N+1$ leaves, $N \geq 0$, be constructed with the following profile:

$$\nu_h = N+1\,, \quad \nu_i = \left\lceil \nu_{i+1}/2 \right\rceil, \quad 0 \leq i < h\,, \quad \text{where} \quad h = \left\lceil \log_2(N+1) \right\rceil \qquad (4.1)$$

We first prove that T_{N+1} is SCO.

Theorem 4.2: For all $N \geq 0$, T_{N+1} constructed with profile given by (4.1) is SCO.

Proof: Clearly T_1 is SCO. Complete the proof by induction on the number of leaves of T_{N+1}. Assume T_{k+1} is SCO for all k, $0 \leq k < N$, for some $N \geq 0$. Consider T_{N+1}. If T_{N+1} is not SCO, then there exists a T'_{N+1} which is SCO, whose profile is different from (4.1). Let $\nu'_0, \ldots, \nu'_{h'}$ denote the profile of T'_{N+1}, where $h' \geq h$. Compare ν_h and $\nu'_{h'}, \nu_{h-1}$ and $\nu'_{h'-1}, \cdots$. Let $j \geq 1$ be the least integer such that $\nu_{h-j} \neq \nu'_{h'-j}$.

If $j > 1$ then we can replace the prefix of T'_{N+1} of height $h' - j + 1$ with the SCO tree with ν_{h-j+1} leaves, by our inductive assumption. If $j = 1$ then $\nu'_{h'-1} > \nu_{h-1}$, in which case $\text{SPACECOST}(T_{\nu'_{h'-1}}) > \text{SPACECOST}(T_{\nu_{h-1}})$, by

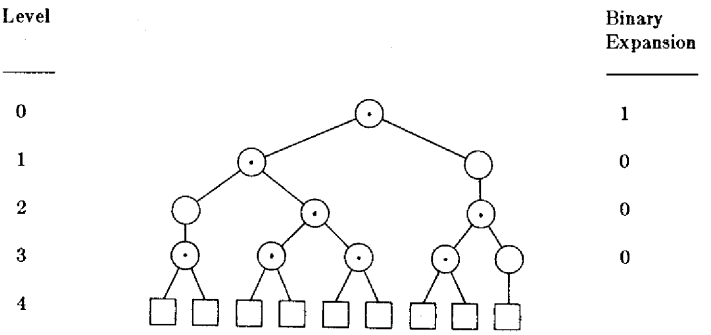
Lemma 4.1, contradicting the optimality of T'_{N+1}. Both cases lead to the conclusion that T_{N+1} is SCO. Hence the result.

The following corollary follows easily from Theorem 4.2:

Corollary 4.3: A 1-2 brother tree is SCO if and only if it has at most one unary node on each level (that is, if and only if its profile is given by (4.1)).

This also implies that a SCO 1-2 brother tree must be height-minimal. The converse is not true, as can be seen from Figure 1. Furthermore, SCO profiles are unique for any value of N, although this is not true for NVO profiles. Finally, we mention that we can read off from the binary expansion of the number N on which levels the unary nodes (if any) occur in SCO 1-2 brother trees. A value of 0 corresponds to a unary level.

Example: An 8-key SCO 1-2 brother tree must have exactly one unary node on levels 1, 2, 3. For $8 = (1000)_2$. Thus, the tree may be as follows:



As in Section 3 we simply generate the detailed profile of an SCO tree, from which a corresponding skeletal brother tree T is easily generated. The N keys are sorted and then T is filled in with an inorder traversal. In Section 5 we give an explicit SCO tree construction algorithm.

Procedure SCO Profile

Input: Natural number N (of keys to be stored).

Output: The detailed profile of an SCO 1-2 brother tree with N binary nodes and $N+1$ leaves.

begin $h := \lceil \log_2(N+1) \rceil$
$\omega_h := 0$; $\beta_h := N+1$; $\nu_h := N+1$
for $i := h-1$ **downto** 0 **do**
begin $\beta_i := \nu_{i+1}$ **div** 2;
$\omega_i := \nu_{i+1}$ **mod** 2 ;
$\nu_i := \omega_i + \beta_i$
end;
Let $\Delta = <\omega_0, \beta_0>, \ldots, <\omega_h, \beta_h>$
end;

5. COMPARISON COST OPTIMAL 1-2 BROTHER TREES

We will characterize structurally those 1-2 brother trees that have minimal comparison cost among all 1-2 brother trees with a given number of leaves (or, equivalently, of keys). Our characterization uses the close correspondence between brother trees and AVL trees (see Ottmann, Six and Wood [1979]). Consider a brother tree T. Let $contract(T)$ denote the tree obtained by replacing each unary node p in T by its only son σp. Then the resulting tree is height-balanced, that is for each node p, the balance factor of p, that is the height difference between the left and right subtree of p, is $+1$, 0 , or -1 . NVCOST and COMPCOST are defined for AVL trees in analogy to the related cost measures of 1-2 brother trees. Clearly, the two cost measures coincide for AVL trees. Further, if T is a 1-2 brother tree then COMPCOST(T) = NVCOST$(contract(T))$. Hence, characterizing the CCO 1-2 brother trees amounts to characterizing the NVO AVL trees.

Lemma 5.1: An AVL tree T is NVO (or, equivalently, CCO) if and only if all leaves of T occur on (at most) two adjacent levels.

Proof: Clearly, an AVL tree is NVO if and only if its internal path length is minimal. Knuth [1973, Section 2.3.4.5] has shown that a binary tree has minimal internal path length if and only if all its leaves occur on at most two adjacent levels.

Let T be a 1-2 brother tree. Then all leaves of $contract(T)$ occur on (at most) two adjacent levels if and only if every root to leaf path of T contains at most one unary node. This yields the desired characterization of CC0 1-2 brother trees:

Theorem 5.2 A 1-2 brother tree T is CCO if and only if every root-to-leaf path of T contains at most one unary node.

Proof: The stated condition is clearly sufficient by Lemma 5.1 and the preceding remark. Now consider a 1-2 brother tree T which contains a path with more than one unary node on it. Then $contract(T)$ is an AVL tree in which not all leaves appear on two adjacent levels; hence, $contract(T)$ is not NVO and, therefore, T cannot be CCO.

NVCOST and COMPCOST are independent cost measures. For example, the 17-leaf NVO tree must have two unary nodes on some root-to-leaf path, hence cannot be CCO. On the other hand, the 6-leaf tree T_a of Figure 1 is CCO but not NVO. In particular, COMPCOST is not a function of a tree's detailed profile. Figure 1 further shows that COMPCOST optimality does not imply SPACECOST optimality. However, each SCO 1-2 brother tree can be transformed to a (profile equivalent SCO and) CCO tree. Rather than providng this directly we give an algorithm which constructs an SCO 1-2 brother tree

which is simultaneously CCO, using the binary expansion of N.

Procedure SCO and CCO 1-2 Brother Tree.

Input: Natural number N (of keys to be stored).

Output: A 1-2 brother tree with N binary nodes which is simultaneously SCO and CCO.

```
begin h := ⌈log2(N+1)⌉;
    if N=0 then return else
    if N=1 then return else
    begin Create a binary node p, say;
        if the coefficient of 2^(h-2) in the binary expansion
            of N is 0 then
        begin { insert unary node }
            a) Attach a unary node q as the right son of p
                and attach a complete binary tree of height
                h-2 as q's only subtree.
            b) Attach a subtree of N-2^(h-2) binary nodes,
                as p's left subtree, constructed recursively.
        end else { both sons are binary }
                Attach a subtree of (N-1) div 2 binary nodes
                as p's left subtree, and one of N-1-(N-1) div 2
                binary nodes as p's right subtree; both constructed
                recursively.
    end
end;
```

6. A COMPARISON OF THE COST-MEASURES

The two cost measures NVCOST and SPACECOST are distinct in the sense that there are N-key 1-2 brother trees which are NVO but not SCO and 1-2 brother trees which are SCO but not NVO. However, the cost measures are not totally unrelated. We already know that NVO and SCO 1-2 brother trees both must have the minimal height $h = \lceil \log_2(N+1) \rceil$, where N is number of keys. Moreover NVO and SCO 1-2 brother trees are in one sense dual to each other, since, for a given height, in NVO trees the number of nodes is maximized while in SCO trees the number of nodes is minimized. Thus, for $N = 2^h - 1$ the complete binary tree of height h is both NVO and SCO. In general, the SPACECOST of the NVO 1-2 brother tree exceeds the SPACECOST of the SCO tree with the same number of keys; and the NVCOST of the SCO tree exceeds the NVCOST of the NVO tree. But, by how much may they differ?

Recall that for each 1-2 brother tree T of height h with N keys the following holds:

$$\text{NVCOST}(T) + \text{SPACECOST}(T) = h \cdot (N+1) \quad . \tag{6.1}$$

Let $T_{SCO}(N)$ denote the SCO 1-2 brother tree with N keys and height $h = \lceil \log_2(N+1) \rceil$, and let $T_{NVO}(N)$ denote the corresponding NVO tree.

Summing up the numbers of nodes in profiles for $T_{NVO}(N)$ given in Theorems 3.5 and 3.7, we find

$$\text{SPACECOST}(T_{NVO}(N)) = \begin{cases} N + 2^{h-2} & \text{if } 2^{h-1} < N+1 \leq 3 \cdot 2^{h-2} \\ 2^h - 1 & \text{if } 3 \cdot 2^{h-2} < N+1 \leq 2^h \end{cases} \quad . \tag{6.2}$$

From (6.1) we may then infer

$$\text{NVCOST}(T_{NVO}(N)) = \begin{cases} h \cdot (N+1) - N - 2^{h-2} & \text{if } 2^{h-1} < N+1 \leq 3 \cdot 2^{h-2} \\ h \cdot (N+1) - 2^h + 1 & \text{if } 3 \cdot 2^{h-2} < N+1 \leq 2^h \end{cases} \tag{6.3}$$

Corollary 4.3 and (6.1) also give us the results:

$$N \leq \text{SPACECOST}(T_{SCO}(N)) \leq N + (h-1) \tag{6.4}$$

$$(h-1)N + 1 \leq \text{NVCOST}(T_{SCO}(N)) \leq (h-1)N + h \quad . \tag{6.5}$$

Combining (6.2) with (6.4) we easily find

$$1 \leq \frac{\text{SPACECOST}(T_{NVO}(N))}{\text{SPACECOST}(T_{SCO}(N))} < \frac{3}{2}.$$

Similarly using (6.3) and (6.5) we find that, for all N ,

$$\frac{\text{NVCOST}(T_{SCO}(N))}{\text{NVCOST}(T_{NVO}(N))} \leq \frac{(h-1)N + h}{(h-1)N + h - 2^{h-2}}$$

so for $h \geq 3$ we obtain, since $N \geq 2^{h-1}$,

$$1 \leq \frac{\text{NVCOST}(T_{SCO}(N))}{\text{NVCOST}(T_{NVO}(N))} < \frac{4}{3} \quad .$$

In fact this bound can also be shown to hold for $h = 1$ or 2. Thus we have shown that, for any value of N, the SPACECOSTs and NVCOSTs of optimal trees are within a small constant factor of each other.

7. OPTIMAL BROTHER LEAF SEARCH TREES

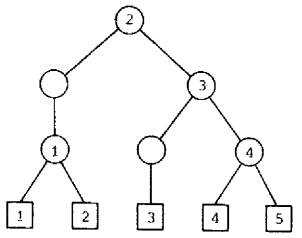
Recall that a brother leaf search has all the keys at the leaf level, while the internal binary nodes contain separating or routing information. Hence to access a key stored in a brother leaf search tree, each node on the path from the root to the appropriate leaf must be visited. Therefore, the node-visit cost of a brother leaf search tree T can be defined by

$$\mathrm{NVCOST}(T) = (\mathit{height}(T)+1)\cdot(\ \mathit{number\ of\ leaves\ of}\ T\ )\ .$$

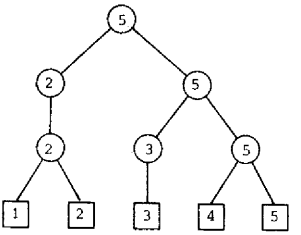
This implies that an N-key brother leaf search tree T, that is an N leaf tree, is NVO if and only if T has the minimal possible height $h = \lceil \log_2 N \rceil$.

This definition of the NVCOST and the resulting trivial characterization of NVO brother leaf search trees implicitly presupposes that separators are assigned to internal nodes in such a way that a search for a key in the tree only visits nodes which lie on the root to leaf path. This presupposition does not hold if we assign to each internal node the maximum value of its sons, as in Ottmann and Six [1976], since this implies the left sons of nodes on the search path are also visited. But if we assign to each (internal) binary node p the maximum value in the left subtree of p as in Mehlhorn [1977], then the presupposition does indeed hold. This indicates that the definition of NVCOST for brother leaf search trees is *dependent* on the routing scheme used, cf. Kwong and Wood [1980].

The routing scheme is also crucial to the definition of COMPCOST for a brother leaf search tree. This is because COMPCOST depends on the underlying search procedure. Consider, for example, the following two sample trees T_a and T_b, where T_a has separators which are the maximum value in the left subtree, and T_b has separators which are the maximum value in the right or only subtree. Note that by the results of Sections 3, 4, and 5 T_a and T_b are NVO, SCO, and CCO brother trees.



T_a



T_b

In order to retrieve a key x stored in the tree T_a we may perform $\mathit{search}_a(\mathit{root}(T_a), x)$ where search_a is the following procedure:

Procedure $search_a(p,x)$

Case 1 [p is a leaf]

if $value(p)=x$ **then return** (p) **else** nil;

Case 2 [p is unary, that is p has a single son σp]

$search_a(\sigma p,x)$

Case 3 [p is binary, that is p has a left son λp and a right son ρp]

if $x \leq value(p)$ **then** $search_a(\lambda p,x)$ **else**
$search_a(\rho p,x)$.

In order to access each key stored in T_a exactly once via $search_a$, a total number of 17 comparisons is required. The routing scheme for T_b suggests retrieving a key x by performing $search_b(root(T_b),x)$, where $search_b$ is obtained from $search_a$ by replacing Case 3 (where p is binary) by the following:

compare x and value(λp). (This comparison has one of three different possible outcomes)

Case 3.1 [$x = value(\lambda p)$]

if λp is a leaf **then return** (λp)
else $search_b(\lambda p, x)$;

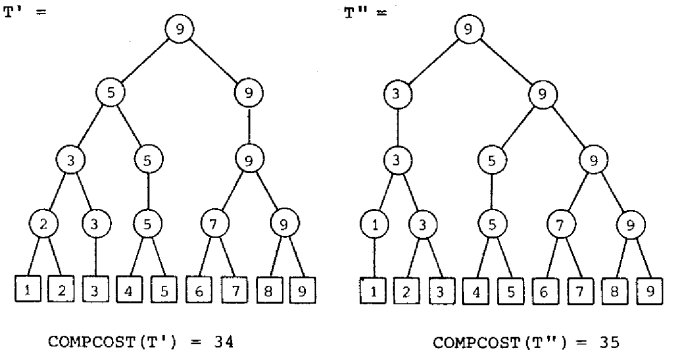
Case 3.2 [$x < value(\lambda p)$]

if λp is a leaf **then** nil **else** $search_b(\lambda p, x)$;

Case 3.3 [$x > value(\lambda p)$]

$search_b(\rho p, x)$.

This example shows why a uniform definition of COMPCOST is impossible for brother leaf search trees. We must consider a definition of COMPCOST with respect to a particular routing scheme (or a class of routing schemes). Furthermore even when a particular routing scheme is chosen the COMPCOST may not even be characterized by the detailed profile. For example, the following brother leaf search trees T' and T'' both have the same detailed profile, and the same routing scheme, but different COMPCOSTs:



Fortunately no such difficulties arise when defining SPACECOST for brother leaf search trees. Since the leaves are now used, a brother leaf search tree T with profile $\pi(T) = \nu_0, \ldots, \nu_h$, $h = height(T)$, has space-cost

$$\text{SPACECOST}(T) = \sum_{i=0}^{h} \nu_i \quad .$$

Since the SPACECOST of an N-key brother leaf search tree is $N + 1 + \sum_{i=0}^{h-1} \nu_i$, minimizing the SPACECOST amounts to minimizing the number of internal nodes. Thus we obtain the same characterization of SCO brother leaf search trees as already proved for 1-2 brother trees in Section 5.

8. CONCLUDING REMARKS AND HISTORY

We have characterized the 1-2 brother trees which are optimal with respect to one of three cost measures, NVCOST, COMPCOST and SPACECOST. In particular we have shown that SCO and CCO 1-2 brother trees having N binary nodes have the same detailed profile. Moreover an NVO 1-2 brother tree with N binary nodes has a SPACECOST differing by at most 50% from the SPACECOST of an SCO 1-2 brother tree with N binary nodes. Similarly the NVCOST of an SCO 1-2 brother tree differs by at most 33% from that of an NVO 1-2 brother tree with the same number of binary nodes. Thus the optimal trees according to one measure are nearly optimal with respect to the other measures.

We have also considered, albeit briefly, the optimality of brother leaf search trees. Although SPACECOST is defined and characterized as for 1-2 brother trees, we demonstrate that NVCOST and COMPCOST are both dependent upon the particular routing scheme that is used. Moreover we have shown that even when a routing scheme is chosen it is possible that the CCO and NVO trees cannot be characterized solely by detailed profiles.

This leads us to a number of open problems. First for the classical routing scheme (an internal binary node is assigned the maximum value in its left subtree) characterize NVO and CCO brother leaf search trees. Second do this for the other routing scheme mentioned here (an internal node is assigned the maximum value in its subtrees). Third, investigate 1-2 brother trees which are simultaneously SCO and NVO. Fourth, as in Rosenberg and Snyder [1981] consider the effects of insertions (and, possibly, deletions) on an initially optimal 1-2 brother tree. Fifth, investigate update schemes which maintain near optimality of 1-2 brother trees according to one of the desired measures. Sixth, how close are "random" 1-2 brother trees to being NVO, CCO and SCO? There are some preliminary results for some of these questions by one of us, namely H.W.Six.

Finally, in closing let us mention something of the history of the present paper. In 1978 the authors, apart from D. Stott Parker, submitted a first draft of this paper to SICOMP. However the results and proofs of Section 3, although correct, did not then have their present precise combinatorial flavor, along the lines of Miller et al [1978]. Fortunately the referee insisted through two further drafts that such an approach was possible. Eventually, after we argued that it was impossible, he produced a detailed proof outline of a new Section 3, which became, in essence, the present Section 3. This so changed the orientation of the paper that we felt that the referee should be included as a co-author. The referee was D. Stott Parker.

9. REFERENCES

Adelson-Velskij, G.M. and Landis, Y.M.: An Algorithm for the Organization of Information. *Doklady Akademij Nauk SSSR 146,* (1962), 263-266.

Aho, A.V., Hopcroft, J.E., and Ullman, J.D.: *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Co., Reading, Mass., 1974.

Knuth, D.E.: *The Art of Computer Programming, Vol. I, Fundamental Algorithms.* Addison-Wesley Publishing Co., Reading, Mass. 1969.

Mehlhorn, K.: *Effiziente Algorithmen.* Teubner Studienbücher Informatik, Stuttgart, 1977.

Miller, R.E., Pippenger, N.J., Rosenberg, A.L., and Snyder, L.: Optimal 2,3-trees, *SIAM Journal on Computing 8,* (1979), 42-59.

Kwong, Y.S., and Wood, D.: On B-trees: Routing Schemes and Concurrency, *Proceedings of the 1980 ACM/SIGMOD International Conference on Management of Data,* (1980), 207-213.

Ottmann, Th., and Six, H.W.: Eine neue Klasse von ausgeglichenen Binärbäumen, *Angewandte Informatik 18/9,* (1976), 395-400.

Ottmann, Th., Six, H.W., and Wood, D.: On the Correspondence Between AVL Trees and Brother Trees, *Computing 23,* (1979), 43-54.

Ottmann, Th., Six, H.W., and Wood, D.: Right Brother Trees, *Communications of the ACM 21,* (1981), 769-776.

Ottmann, Th., and Wood, D.: 1-2 Brother Trees or AVL Trees Revisited, *The Computer Journal 23,* (1981), 248-255.

Rosenberg, A.L., and Snyder, L.: Minimal Comparison 2,3-trees, *SIAM Journal on Computing 7,* (1978), 465-480.

Rosenberg, A.L., and Snyder, L.: Time- and Space- Optimality in B-trees, *ACM Transactions on Data Base Systems 6,* (1981), 174-183.