The Analysis of an Improved Symmetric Binary B-tree Algorithm

NIVIO ZIVANI
Departamento de Ciência da Computação, UFMG, CP 702, 30,000 Belo Horizonte, MG, Brazil

HENK J. OLIVIÉ
Stedelijke Industriële Hogeschool, Paardenmarkt 94, B-2000 Antwerp, Belgium

GASTON H. GONNET
Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L3G1, Canada

We present an improved version of the original insertion and deletion algorithm for symmetric binary B-trees. We perform the analysis of the insertion algorithm using a fringe analysis method, and obtain bounds on the expected number of transformations per insertion and on the expected number of balanced nodes. We also examine empirically the expected costs to search, insert, and delete keys in symmetric binary B-trees.

1. INTRODUCTION

B-trees were presented in 1972 as a dictionary structure primarily for secondary storage.[1] A special case of B-trees, called 2–3 trees, was introduced by John Hopcroft in 1970 (see Ref. 2, p. 468). In a 2–3 tree every internal node contains either one or two keys, and all leaves appear at the same level. Unlike B-trees, 2–3 trees are more appropriate for use in primary store than secondary.

Bayer proposed a binary representation for 2–3 trees, as shown in Fig. 1.1*[3] Note that the binary representation for 2–3 trees has an asymmetry: the left edges always point to a node at the next level, while the right edges point either to a node at the same level or point to one at the next level. Removing the asymmetry of the binary B-trees leads to the symmetric binary B-trees (abbreviated as SBB trees).[4]

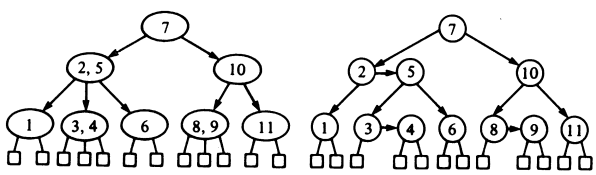

Figure 1.1. A 2–3 tree and the corresponding binary B-tree.

SBB trees are binary trees with two kinds of edges, namely vertical edges and horizontal edges (called δ-edges and ρ-edges respectively, by Bayer), such that:[4]

(i) all paths from the root to every leaf node contain the same number of vertical edges;
(ii) there are no successive horizontal edges;
(iii) all leaf nodes are endpoints of vertical edges.

Fig. 1.2 shows a representation of an SBB tree.

For SBB trees two kinds of heights need to be distinguished: the vertical height h, required for the uniform height constraint, and calculated by counting only vertical edges in any path from root to leaf, and the ordinary height k, required to determine the maximum number of key comparisons and calculated by counting all edges in a maximal path from root to leaf. The ordinary height k is larger than the vertical height h wherever the tree has some horizontal edges. In particular, for a given SBB tree with n internal nodes, we have

$$h \leqslant k \leqslant 2h,$$

and

$$\log(n+1) \leqslant k \leqslant 2\log(n+2) - 2.^{4*}$$

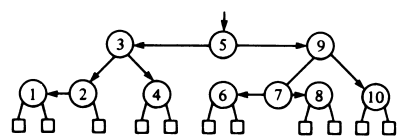

Figure 1.2. An SBB tree with h-height 2 and k-height 4.

In 1972 Bayer introduced the SBB trees and the maintenance algorithms, and showed that the class of AVL trees is a proper subset of the class of SBB trees.[4] Later Wirth presented an implementation of the insertion algorithm using Pascal.[5] An SBB tree can be seen as a binary representation for a 2–3–4 tree as defined by Guibas and Sedgewick (1978),[6] in which 'supernodes' may contain up to three keys and four sons. For example, such a 'supernode' (with keys 3, 5, and 9 and with sons containing keys 2, 4, 7 and 10) can be seen in the SBB tree of Fig. 1.2.

Huddleston and Mehlhorn showed that the number of nodes revisited to restore the SBB property, counted from the father of the node inserted into the tree to the node where the retreat terminated, is constant on the average.[7] In another paper they used SBB trees as a basic date structure for representing linear lists.[8] The University of Washington's ESP text editor developed by Fisher, Ladner, Robertson and Sandberg uses SBB trees as a basic data structure.[9]

Olivié proposed a new set of transformations for SBB trees and son trees.[10, 11] Ziviani and Tompa (1982) showed

* In this paper the internal nodes are circular and the leaves or external nodes are square.

* All logarithms are taken to base two.

experimentally that (i) the new insertion algorithm uses less transformations per insertion and produces SBB trees with smaller height than the original algorithm proposed in Ref. 4, and (ii) SBB trees require less work than AVL trees to maintain balance, and the search time is only slightly longer.[12] In Olivié and Schrapp (1982)[13] a close relationship is shown between son trees and half balanced binary trees.[14] Due to this relationship Tarjan (1983) was able to give insertion and deletion algorithms for SBB trees requiring only a constant number of single rotations after an insertion or a deletion.[15]

The purpose of this paper is to present the analysis of the improved algorithm for SBB trees. The analytical results of the insertion algorithm are obtained using a technique called fringe analysis.[16,17] We use a fringe analysis method based on a way of describing the composition of a fringe in terms of tree collections. The expected costs to search, insert and delete keys are examined empirically.

In section 2 of this paper the original algorithm proposed in Ref. 4 and the new algorithm are studied. In section 3 a fringe analysis of SBB trees is performed. In Section 4 a performance evaluation of SBB trees is presented.

2. THE ALGORITHMS

The algorithms to insert or to delete a key from an SBB tree use local transformations on the path of insertion to preserve the balance of the tree. The process of insertion of a new key consists of three parts.

(i) Follow the search path until it is verified that the key is not in the tree. This is equivalent to finding the place of insertion, which is located after the lowest vertical pointer in the tree, and is represented by a leaf node.

(ii) Replace the leaf node by an internal node with two leaves as sons and containing the new key. This node is raised to the next higher level by changing the edge that points to the new node from vertical to horizontal.

(iii) Retreat along the search path and check the pointers at each node. Depending on the tree's status prior to insertion two successive horizontal pointers may result, and a transformation may become necessary. If a transformation is performed a node may again be raised to a higher level and this may require further transformations to obtain a uniform height.

The process of deletion of a key consists of three parts.

(i) Follow the search path until it is verified that the key is in the tree.

(ii) Delete the node according to the following: (*a*) the key to be deleted belongs to a node with at most one son, deletion in this case is straightforward; (*b*) the key to be deleted belongs to a node with two sons, it is to be replaced by the rightmost key of its left subtree.

(iii) Retreat along the search path and restore the SBB tree property.

Fig. 2.1 shows the transformations proposed by Bayer.[4] Symmetric transformations may also occur. On the left side we have the situation where a node has been lifted up one level, so that there are two successive horizontal edges. On the right side the result of the restructuring is shown.

Fig. 2.2. shows a new set of transformations. In the new transformations for SBB trees the edges marked by a star in Fig. 2.1 are considered explicitely: only if this edge is horizontal is a node lifted up to the next higher level.

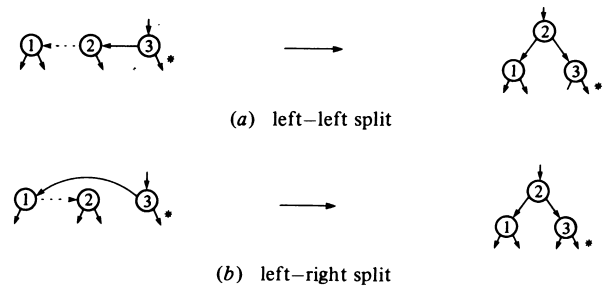

Figure 2.1. The two transformations as proposed by Bayer.[4] The edges marked with stars may be either vertical or horizontal.

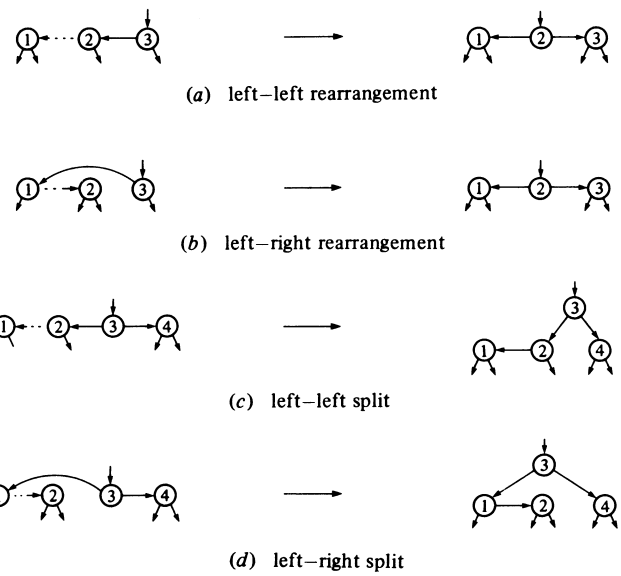

Figure 2.2. The four transformations as proposed by Olivié.[10]

The left–left rearrangement and the left–right rearrangement require the modification of 3 and 5 pointers respectively, and the split transformations require the modification of two bits. The difference between a vertical and a horizontal pointer is indicated by one bit. Symmetric transformations may also occur.

When a split transformation occurs, the height of the transformed subtree is one more than the height of the original subtree, and this may cause other transformations along the search path to the root. The retreat along the search path terminates when either a vertical pointer is found or a rearrangement transformation is performed. As the height of the rearranged subtree is the same as the height of the original subtree, at most one rearrangement transformation per insertion may be performed.

The implementation of the original algorithm can be found in Bayer (1972) using Algol, and in Wirth (1976) using Pascal.[4,5] The implementation of the new algorithm can be found in Olivié (1980) and in Ziviani and Tompa (1982) using Pascal.[10,12] Bayer, Wirth and Olivié used two bits per node in their algorithms to indicate whether the right and left pointers are vertical or horizontal. Ziviani and Tompa implemented only one bit per node, as suggested by Guibas and Sedgewick:[6] the information whether the left (right) pointer of a node is vertical or horizontal is stored in the left (right) son.

Ziviani and Tompa (1982) have shown empirically that the insertion algorithm for SBB trees based on the transformations shown in Fig. 2.2. performs better than the original algorithm based on the transformations

Table 3.1. Summary of the SBB tree results

Tree collection size...	First-order analysis* 3	Second-order analysis† 30
$\frac{\bar{b}(N)}{N}$	$\left[0.51+\frac{0.51}{N}, 0.86-\frac{0.14}{N}\right]$	$\left[0.57+\frac{0.57}{N}, 0.72-\frac{0.28}{N}\right]$
$r(N)$	[0.29, 0.63]	[0.36, 0.56]
$Pr\{0\ s(N)\}$	0.66	0.66
$Pr\{1$ or more $s(N)\}$	0.34	0.34
$\bar{f}(N)$	$0.66N+0.66$	$0.85N+0.85$

* For $N \geqslant 6$.
† Results are approximated to $O(N^{-5.96})$.

shown in Fig. 2.1. In the next section we present an analysis of the algorithm based on the new transformations shown in Fig. 2.2.

3. ANALYTICAL RESULTS

Consider an SBB tree T with N keys and consequently $N+1$ external nodes. These N keys divide all possible key values into $N+1$ intervals. An insertion into T is said to be a *random insertion* if it has an equal probability of being in any of the $N+1$ intervals defined above. A *random SBB tree* with N keys is an SBB tree constructed by making n successive random insertions into an initially empty tree. In this paper we assume that all trees are random trees.

We now define certain complexity measures.

(i) Let $b(N)$ be the expected number of completely k-balanced* nodes in an SBB tree after the random insertion of N keys into the initially empty tree.

(ii) Let $r(N)$ be the expected number of rearrangements required during the insertion of the $(N+1)$st key into a random SBB tree with N keys.

(iii) Let $Pr\{0\ s(N)\}$ be the probability that zero splits occur during the insertion of the $(N+1)$st key into a random SBB tree with N keys.

(iv) Let $Pr\{1\ or\ more\ s(N)\}$ be the probability that one or more splits occur during the insertion of the $(N+1)$st key into a random SBB tree with N keys.

(v) Let $m(N)$ be the maximum number of rearrangements that may occur outside the fringe in an SBB tree during the insertion of the $(N+1)$st key into a random SBB tree with n keys.

(vi) Let $\bar{f}(N)$ be the expected number of nodes in the fringe of an SBB tree after the random insertion of N keys into the initially empty tree.

In section 3.1 the fringe analysis technique is introduced. In Section 3.2 a small tree collection of SBB trees of h-height 1 is studied. In Section 3.3. a bigger tree collection of SBB trees of h-height 2 is proved to be a closed collection and results on the complexity measures are obtained.

Table 3.1 shows a summary of the results obtained for SBB trees.

3.1. Fringe analysis technique

In section 3.1.1 we introduce the concepts necessary to describe the Markov chain used to model the insertion process. In section 3.1.2. we study the matrix recurrence relation involved in the Markov process. More details may be found in Ref. 20 or 17.

* A node in an SBB tree is k-balanced when the k-height of the left subtree is equal to the k-height of the right subtree.

3.1.1. The Markov process

Let us define a *tree collection* as a finite collection $C = \{T_1, \ldots, T_m\}$ of trees. The collection of SBB trees with three leaves or fewer forms a tree collection, as shown in Fig. 3.1.1.1.

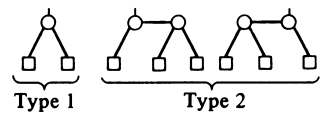

Figure 3.1.1.1. Tree collection of SBB trees with three leaves or fewer.

The *fringe* of a tree consists of one or more subtrees that are isomorphic to members of a tree collection C. Typically, the fringe will contain all subtrees that meet this definition; for example the fringe of a SBB tree that corresponds to the tree collection of Fig. 3.1.1.1 is obtained by deleting all nodes at a distance greater than 1 from the leaves. Fig. 3.1.1.2 shows an instance of an SBB tree with eleven keys in which the fringe that corresponds to the tree collection of Fig. 3.1.1.1 is encircled.

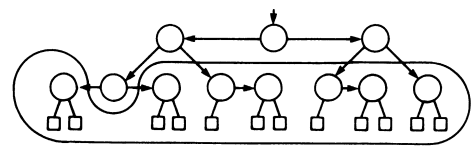

Figure 3.1.1.2. An SBB tree and its fringe that corresponds to the tree collection of Fig. 3.1.1.1.

The composition of the fringe can be described in several ways. One possible way is to consider the probability that a randomly chosen leaf of the tree belongs to each of the members of the corresponding tree collection. In other words, the probability p is

$$p_i(N) = \frac{\text{Expected number of leaves of type } i \text{ in an } n\text{-key tree}}{n+1} \quad (1)$$

Yao (1978) describes the fringe in a different way. His description of the composition of the fringe considers the expected number of trees of type i, while we describe it

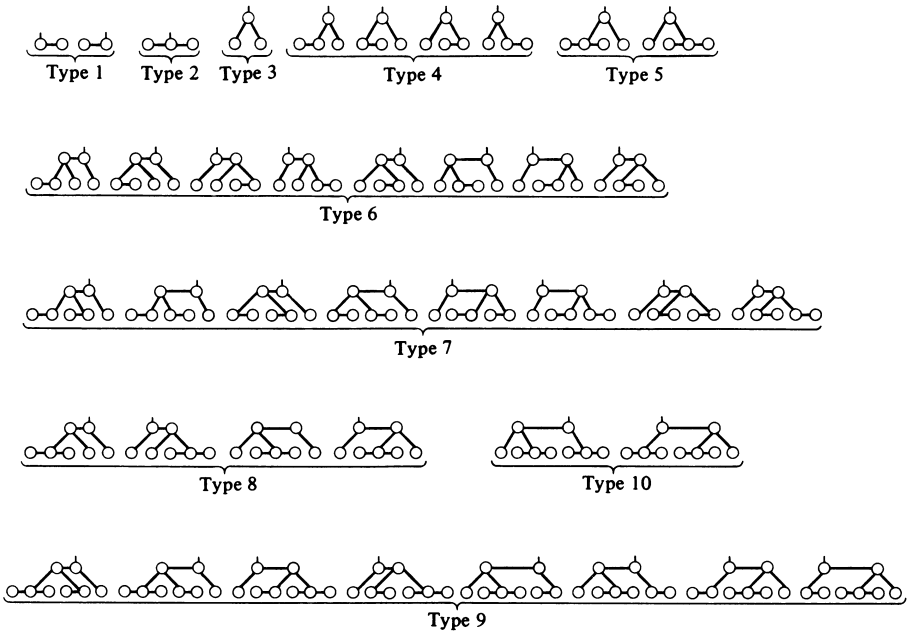

Figure 3.1.1.3. Tree collection of SBB trees with 10 types (leaves not shown).

in terms of leaves as in Equation (1). As we shall see our description of the composition of the fringe simplifies the notation necessary to present the fringe analysis technique, and also makes easier the task of finding which complexity measures can be obtained from the analysis of each search tree.

In fringe analysis problems we always deal with a collection $C = \{T_1, \ldots, T_m\}$ of trees. We now introduce some concepts about the fringes of search trees.

Definition 3.1.1.1. A tree collection $C = \{T_1, \ldots, T_m\}$ is *weakly closed* if for all $j \in [1, \ldots, m]$ an insertion into T_j always leads to one or more T_i, $i \in [1, \ldots, m]$.

Definition 3.1.1.2. A tree collection $C = \{T_1, \ldots, T_m\}$ is *closed* when (i) C is weakly closed and (ii) the effect of an insertion on the composition of the fringe is determined only by the subtree of the fringe where the insertion is performed.

The tree collection of Fig. 3.1.1.1 is an example of a closed tree collection. This can be shown informally by noting that an insertion into a type 1 tree always leads to a type 2 tree, and an insertion into a type 2 tree always leads to two type 1 tree, and in this case a rearrangement transformation occurs with probability 2/3.

On the other hand the collection of SBB trees shown in Fig. 3.1.1.3 is weakly closed but not closed. This is because an insertion into a type 2 tree of Fig. 3.1.1.3, when the type 2 tree is part of the fringe of an SBB tree, may cause a split transformation followed by a rearrangement transformation higher in the tree, and the composition of the fringe depends on this rearrangement at the higher level. Fig. 3.1.1.4 shows an instance of a SBB tree where an insertion into a type 2 tree does not lead to a type 4 tree as expected.

Definition 3.1.1.3. A tree collection $C = \{T_1, \ldots, T_m\}$ is *ambiguous* when a tree in C appears as a subtree

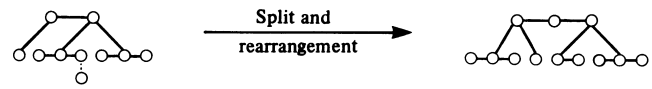

Figure 3.1.1.4. Example of an insertion that unexpectedly changes the fringe of an SBB (dotted edge shows the point of insertion).

of another tree in C. Fig. 3.1.1.5 shows a SBB tree collection that is ambiguous, since a tree of type 1 is a subtree of trees of type 3.

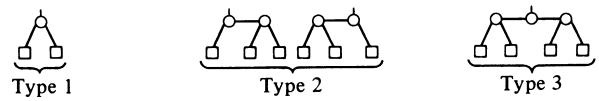

Figure 3.1.1.5. Tree collection of SBB trees with more than 1 and less than 5 leaves.

The transitions between trees of a tree collection can be used to model the insertion process. In an insertion of a key into the type 1 tree shown in Figure 3.1.1.1 two leaves of type 1 are lost and three leaves of type 2 are obtained. In an insertion of a key into the type 2 tree three leaves of the type 2 are lost and four leaves of type 1 tree are obtained.

Clearly the probability that an insertion in one type of a tree collection C leads to another type of C depends only on the two types involves, and so the process is a Markov process (cf. Refs. 18 and 19). A sequence $\{X_N\} = \{X_0, X_1 \ldots\}$ of random variables taking values on a state space S is a Markov chain if

$$Pr\{X_N = i \mid X_{N-1} = j, X_{n-2} = j_1, \ldots, X_0 = j_{N-1}\} = Pr\{X_N = i \mid X_{N-1} = j\},$$

for all $i, j, j_1, \ldots, J_{N-1} \in S$. The current value of X_N depends on the history of the process only through the most recent value X_{N-1}.

To illustrate this fact consider the tree collection of SBB trees shown in Figure 3.1.1.1. In this context, let X_N and Y_N be respectively the numbers of type 1 and type 2 leaves

after the Nth insertion. Since the tree collection is closed, the value of X_N depends only on the value of X_{N-1} and as a consequence $\{X_N\}$ (or equivalently $\{Y_N\}$) is a Markov chain.

The transition probabilities of the chain $\{X_N\}$ are given by

$$Pr\{X_N = i \mid X_{N-1} = j\} = \begin{cases} \dfrac{j}{N} & i = j-2 \\ \dfrac{N-j}{N} & i = j+4 \end{cases}$$

while those of Y_N are

$$Pr\{Y_N = i \mid Y_{N-1} = j\} = \begin{cases} \dfrac{j}{N} & i = j-3 \\ \dfrac{N-j}{N} & i = j+3 \end{cases}$$

Let

$$j_N = E(X_n) \quad \text{and} \quad k_N = E(Y_N).$$

Then

$$\begin{aligned} j_N = E(X_N) &= E[E(X_N \mid X_{N-1}, Y_{N-1})] \\ &= E\left[\frac{X_{N-1}}{N}(X_{N-1}-2)+\frac{Y_{N-1}}{N}(X_{N-1}+4)\right] \\ &= j_{N-1}-\frac{2}{N}j_{N-1}+\frac{4}{N}k_{N-1} \end{aligned}$$

and similarly

$$k_N = k_{N-1}-\frac{3}{N}k_{N-1}+\frac{3}{N}j_{N-1}.$$

But, by definition

$$\begin{aligned} j_{N-1} &= Np_1(N-1); \quad j_N = (N+1)p_1(N); \\ k_{N-1} &= Np_2(N-1); \quad k_N = (N+1)p_2(N). \end{aligned}$$

Substituting these equations into the previous equations we get

$$P_1(N) = \frac{(N-2)p_1(N-1)+4p_2(N-1)}{N+1}$$

and

$$p_2(N) = \frac{3p_1(N-1)+(N-3)p_2(N-1)}{N+1}$$

In matrix notation

$$\begin{bmatrix} p_1(N) \\ p_2(N) \end{bmatrix} = \begin{bmatrix} \dfrac{N-2}{N+1} & \dfrac{4}{N+1} \\ \dfrac{3}{N+1} & \dfrac{N-3}{N+1} \end{bmatrix} \begin{bmatrix} p_1(N-1) \\ p_2(N-1) \end{bmatrix}$$

or

$$\begin{bmatrix} p_1(N) \\ p_2(N) \end{bmatrix} = \left[I+\frac{H}{N+1}\right] \begin{bmatrix} p_1(N-1) \\ p_2(N-1) \end{bmatrix}$$

where

$$H = \begin{bmatrix} -3 & 4 \\ 3 & -4 \end{bmatrix} \quad \text{and} \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Thus the probability of an insertion occurring in each of the subtrees of the fringe can be obtained from the steady-state solution of a matrix recurrence relation in a Markov chain. In general, let $p(N)$ be an m-component column vector containing $p_i(N)$. Then

$$p(N) = \left[I+\frac{H}{N+1}\right]p(N-1), \tag{2}$$

where I is the $m \times m$ identity matrix, and H is the transition matrix.

Extensions to other tree collections with more than two types requires consideration of a vector process $\{\mathbf{X}_N\}$, where x_{jN} is equal to the number of type j leaves at time N.

3.1.2. The matrix recurrence relation

In fringe analysis problems we always deal with a tree collection $C = \{T_1, \ldots, T_m\}$ of trees. Let L_i be the number of leaves of T_i. An insertion into the kth leaf, $k \in [1, \ldots, L_j]$ of T_i will generate $l_{ij}(k)$ leaves of type T_i. Let $p_i(N)$ be defined as in Equation 3.1.1 (1). Then Equation 3.1.1 (2) can be written as

$$p(N) = \left[I+\frac{H_2-H_1-I}{N+1}\right]p(N-1) \tag{1}$$

where

$$H_2 = \left[\frac{1}{L_j}\sum_{k=1}^{L_j} l_{ij}(k)\right]_{1 \leqslant i,j \leqslant m}, \quad H_1 = \operatorname{diag}(L_1, \ldots, L_m),$$

and I is the $m \times m$ identity matrix.

Definition 3.1.2.1. Consider a fringe analysis problem. Equation (1) is the associated recursion equation, where $H = H_2 - H_1 - I = (h_{ij})$ is its transformation matrix. We have

$$h_{ij} = \frac{1}{L_j}\sum_{k=1}^{L_j} l_{ij}(k) - \delta_{ij}(L_j+1)$$

where δ_{ij} is the Kronecker symbol.

Intuitively, the element in the diagonal of H represent the number of leaves lost due to an insertion minus one, and off-diagonal elements represent the number of leaves obtained for each type times the probability that each type is reached in a transition.

Definition 3.1.2.2. A fringe analysis is *connected* if there is an $l \in [1 \ldots m]$ such that $\det(H_{ll}) \neq 0$, where H_{ll} is matrix H with the lth column and lth row deleted.

The following theorem shows that the real part of the eigenvalues of the transition matrix are non-positive. The proof of this theorem and all the following theorems may be found in Ref. 17 or 20.

Theorem 3.1.2.1. Consider a connected fringe analysis problem with a $m \times m$ transition matrix H as in Definition 3.1.2.1. Let $\lambda_1, \ldots, \lambda_m$ be the eigenvalues of H. We can order them so that $\lambda_1 = 0$ and $0 > Re\,\lambda_2 \geqslant Re\,\lambda_2 \geqslant \ldots \geqslant Re\,\lambda_m$.

Definition 3.1.2.3.
Let

$$T_j \rightarrow T_i \quad \text{if} \quad \sum_{l=1}^{L_j} l_{ij}(k) > 0, \text{ i.e. } T_j \text{ can produce } T_i.$$

The symbol $\overset{*}{\rightarrow}$ is the reflexive transitive closure of $\rightarrow$.

The following theorem describes a test for connectedness.

Theorem 3.1.2.2. A fringe is connected if and only if there is a T_i such that

$$T_j \overset{*}{\rightarrow} T_i \quad \text{for all} \quad j \in [1 \ldots m].$$

It remains to solve Equation (1) for connected fringe analysis problems. In a previous version of the proof of the convergence of the matrix recurrence relation (Ref. 21, Lemma 2.1, p. 4) the eigenvalues of the transition matrix are assumed to be pairwise distinct. The following theorem extends the proof to the general case.[22]

Theorem 3.1.2.3. Let H be the $m \times m$ transition matrix of a connected fringe analysis problem. Let $\lambda_1, \ldots, \lambda_m$ be the eigenvalues of H, where $\lambda_1 = 0 > Re\,\lambda_1 \geqslant Re_3 \geqslant \ldots \geqslant Re\lambda_m$, and let q be the right eigenvector of H corresponding to $\lambda_1 = 0$. Then for every vector $p(0)$ there is a c such that

$$|p(N) - cq| = O(N^{Re\,\lambda_2})$$

where $p(N)$ is defined by Equation (1).

It is important to note the following.

(i) Consider a $m \times m$ transition matrix H of a connected fringe analysis problem. Theorem 3.1.2.3. says that $p(N)$, the m-component column vector solution of Equation (1), converges to the solution of

$$Hq = 0, \quad \text{as} \quad N \rightarrow \infty,$$

where q is also an m-component column vector that is independent of N, and

$$p(N) = \alpha_1 q = O(N^{Re\,\lambda_2}), \tag{3}$$

where q is the right eigenvector of H corresponding to eigenvalue $\lambda_1 = 0$. Furthermore, the eigenvalues of H do not need to be pairwise distinct.

(ii) Let $A_i(N)$ be the expected number of trees of type i in a random search tree with N keys. Let L_i be the number of leaves of the type i tree. We observe that Equation 3.1.1 (1) can be written as

$$P_i(N) = \frac{A_i(N) L_i}{N+1}. \tag{4}$$

3.2. First-order analysis.

The analysis of the lowest level of the SBB tree can be carried out by considering the tree collection of SBB trees with four or less leaves and h-height 1, as shown in Fig. 3.2.1. The first step necessary to perform the first-order analysis is to show that the SBB tree collection of Fig. 3.2.1. is closed (Definition 3.1.1.2).

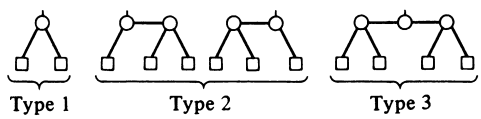

Figure 3.2.1. Tree collection of SBB trees with four or less leaves and h-height 1.

Theorem 3.2.1. The SBB tree collection shown in Fig. 3.2.1. is closed.

Proof. An insertion into type 1 always leads to a type 2 tree. An insertion into a type 2 tree always leads to a type 3 tree, and in this case a rearrangement transformation occurs with probability 2/3. The only case where a split transformation occurs is after an insertion into type 3, and in this case a rearrangement transformation may take place higher in the tree. But even if this rearrangement transformation takes place higher it causes no problem because the smallest subtrees that may be moved around are exactly type 1 and type 2 subtrees, which are the types one would obtain anyway if an insertion is performed into a type 3 tree. □

Theorem 3.2.1 says that all the transitions in the tree collection of Fig. 3.2.1 are well defined, so that the theorems presented in Section 3.1.2. can be applied. Thus

$$H = \begin{bmatrix} -3 & 0 & 2 \\ 3 & -4 & 3 \\ 0 & 4 & -5 \end{bmatrix}$$

From Equation 3.1.2 (2) we have $Hp(N) = 0$, and therefore $p_1(\infty) = 8/35$, $p_2(\infty) = 15/35$, and $p_3(\infty) = 12/35$. Since the eigenvalues of H are 0, -5 and -7, we observe that $p_1(N) = 8/35$, $p_2(N) = 15/35$, and $p_3(N) = 12/35$, for $N \geqslant 6$.

Theorem 3.2.2. The expected number of completely k-balanced nodes in a random SBB tree with N keys is bounded by

$$\left[\frac{p_1}{2} + \frac{p_2}{3} + 3\frac{p_3}{4}\right](N+1) \leqslant \bar{b}(N) \leqslant N - \left[\frac{p_2}{3}\right](N+1).$$

Proof. The lower bound is obtained by using Equation 3.1.2 (4) and observing Fig. 3.2.1. The upper bound comes from the fact that $\bar{b}(N)$ plus the expected number of k-unbalanced nodes is equal to N, and the expected number of k-unbalanced nodes in this case is $p_2/3(N+1)$. □

Corollary.

$$\frac{18}{35} + \frac{18}{35N} \leqslant \frac{\bar{b}(N)}{N} \leqslant \frac{6}{7} - \frac{1}{7N} \quad \text{for} \quad N \geqslant 6.$$

Theorem 3.2.3. The expected number of rearrangements in a random SBB tree with N keys is bounded by

$$\frac{2}{3}p_2 \leqslant r(N) \geqslant 1 - (p_1 + \frac{1}{3}p_2).$$

Proof. The lower bound is obtained by observing that a rearrangement transformation happens when an insertion is performed into the type 2 shown in Fig. 3.2.1, with probability 2/3. The upper bound is obtained by observing that the maximum number of rearrangements per insertion is 1. □

Corollary. $2/7 \leqslant r(N) \leqslant 22/35$ for $N \geqslant 6$.

Lemma 3.2.4. The probabilities that no split or one or more splits occur on the $(N+1)$st random insertion into a random SBB tree with N keys are, respectively

$$Pr\{0\ s(N)\} = p_1 + p_2$$
$$Pr\{1 \text{ or more } s(N)\} = p_3.$$

Proof. By observing Figure 3.2.1. □

Corollary.

$$Pr\{0\ s(N)\} = \frac{23}{35} \quad \text{for} \quad N \geqslant 6$$

$$Pr\{1 \text{ or more } S(N)\} = \frac{12}{35} \quad \text{for} \quad N \geqslant 6.$$

Lemma 3.2.5. The probability of a rearrangement occurring outside the fringe during the insertion of the $(N+1)$st key into a random SBB tree with N keys is $m(N) = p_3$. Furthermore, no more than one rearrangement may occur.

Proof. An insertion into type 3 shown in Fig. 3.2.1 causes a split transformation, which may cause a rearrangement higher in the tree. □

Corollary. $m(N) = 12/35$ for $N \geqslant 6$.

Lemma 3.2.6. The expected number of nodes in the fringe of an SBB tree with N keys that corresponds to the tree collection of Figure 3.2.1 is

$$\tilde{f}(N) = \left[\frac{p_1}{2} + 2\frac{p_2}{3} + 3\frac{p_3}{4}\right](N+1).$$

Proof. By observing Fig. 3.2.1. □

Corollary. $\tilde{f}(N) = 23/35N + 23/35$ for $N \geqslant 6$.

3.3. Second-order analysis

We can improve the bounds obtained in the previous section by considering a larger tree collection. Fig. 3.3.1 shows a tree collection of SBB trees with 30 types and h-height 2.

Theorem 3.3.1. The SBB tree collection of Fig. 3.3.1 is closed.

Proof. When an insertion into an SBB tree causes no rearrangement transformation then there is no problem. When an insertion does cause a rearrangement transformation somewhere in the tree then we consider two possible cases.

(i) The rearrangement transformation occurs at a node that belongs to one of the trees of the tree collection shown in Fig. 3.3.1. This case obviously causes no problem.

(ii) The rearrangement transformation occurs at a node outside the fringe. By examining Fig. 3.3.1 we can see that there is no type that contains two opposite horizontal pointers at the second level. This fact implies that in order to have a transformation out of the fringe we must (*a*) have at least any two subtrees from the tree collection of Fig. 3.3.1 sharing the same root; and (*b*) an insertion into one of these two subtrees must cause a split transformation which will cause the rearrangement transformation higher in the tree. However, this rearrangement transformation has no effect on the composition of the fringe because the fringe of the transformed subtree is entirely contained in the subtrees that are moved around during the transformation.

Fig. 3.3.2 illustrates this fact. Assume that T_1, T_2, T_3 and T_4 contain subtrees that belong to the tree collection of Fig. 3.3.1. Suppose that an insertion into T_2 (*or* T_3) causes a split transformation (resulting in subtree T_2'), which will

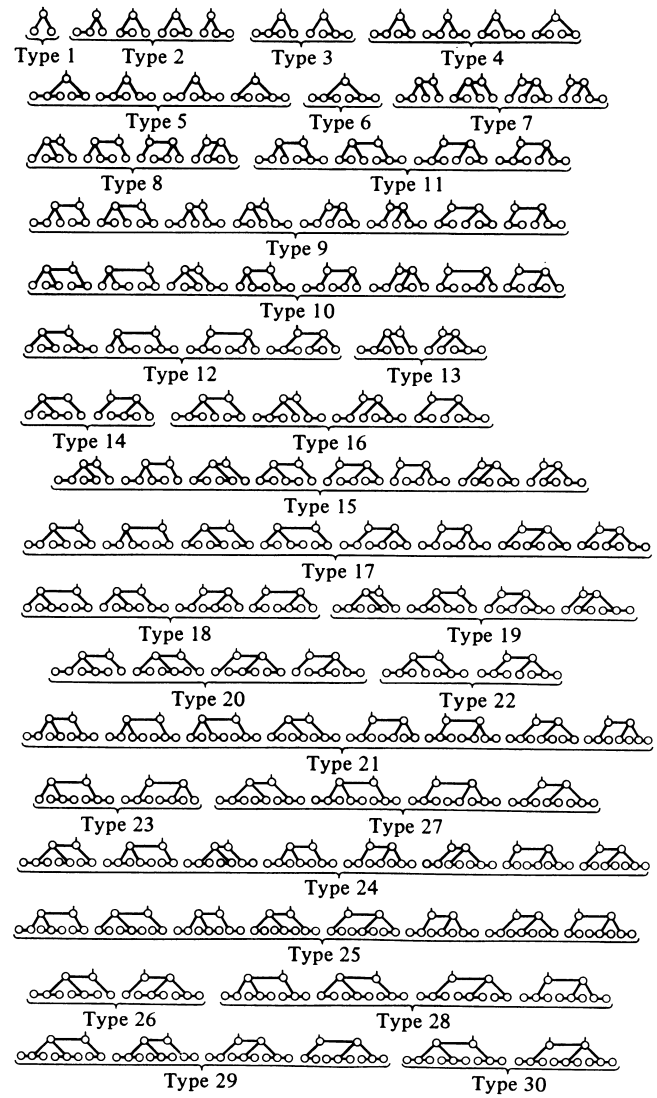

Figure 3.3.1. Tree collection of SBB trees with 30 types (leaves not shown).

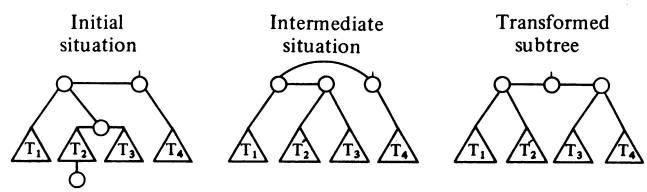

Figure 3.3.2. Left–right rearrangement (an insertion into T_2 transforms it into T'_2).

cause a left–right rearrangement transformation at the next level. Clearly the subtrees T_1, T_2', T_3 and T_4 are moved around without any modification in their structures. □

The matrix H can be easily obtained by observing Fig. 3.3.1. From Equation 3.1.2 (2) we have $H\,p(N) = 0$, and therefore

$p_1 = 874463196/49525503055$ $\quad p_8 = 183823695/5660057492$
$p_2 = 1620896463/9905100611$ $\quad p_9 = 394971148/9905100611$
$p_3 = 1102942170/9905100611$ $\quad p_{10} = 394971148/9905100611$
$p_4 = 783174402/9905100611$ $\quad p_{11} = 7311869433/198102012220$
$p_5 = 176215347/1415014373$ $\quad p_{12} = 7311869433/198102012220$
$p_6 = 516201576/9905100611$ $\quad p_{13} = 122549130/9905100611$
$p_7 = 183823695/5660057492$ $\quad p_{14} = 122549130/9905100611$

$p_{15} = 163398840/9905100611$ $p_{23} = 7420293/460702354$
$p_{16} = 1608763167/99051006110$ $p_{24} = 114212043/9905100611$
$p_{17} = 1072508778/49525503055$ $p_{25} = 114212043/9905100611$
$p_{18} = 1608763167/99051006110$ $p_{26} = 50133735/9905100611$
$p_{19} = 330882651/39620402444$ $p_{27} = 340437933/28300287460$
$p_{20} = 330882651/39620402444$ $p_{28} = 340437933/28300287460$
$p_{21} = 4946862/230351177$ $p_{29} = 360039042/49525503055$
$p_{22} = 7420293/460702354$ $p_{30} = 78121098/9905100611$

Since the eigenvalues of H are 0, -5, $-5.96\pm7.03i$, $-6.4\pm2.49i$, -7, -8, $-8.5\pm2.34i$, -9, -9, -9, -10, -10, -10, -10, $-10.52\pm4.39i$, $-10.79\pm2.38i$, -11, -11, -11, -12, $-12.66\pm1.59i$, -13.73, $-14.80\pm4.22i$, the asymptotic values of $p(N)$ obtained from Equation 3.1.2 (3) are approximated to $0(N^{-5.96})$.

Theorem 3.3.2. The expected number of completely k-balanced nodes in a random SBB tree with n keys is bounded by

$$0.57097+\frac{0.57097}{N}+0(N^{-5.96}) \leqslant \frac{\bar{b}(N)}{N} \leqslant 0.71632-\frac{0.28368}{N}+0(N^{-5.96}).$$

Proof. Similar to the proof of Theorem 3.2.2. □

Theorem 3.3.3. The expected number of rearrangements in a random SBB tree with N keys is bounded by

$$0.35921+0(N^{-5.96}) \leqslant r(N) \leqslant 0.55672+0(N^{-5.96}).$$

Proof. Similar to the proof of Theorem 3.2.3. □

Lemma 3.3.4. The probability of a rearrangement occurring outside the fringe during the insertion of the $(N+1)$st key into a random SBB tree with N keys is

$$m(N) = 0.19751+0(\mathrm{N}^{-5.96}).$$

Proof. Similar to the proof of Theorem 3.2.5. □

Lemma 3.3.5. The expected number of nodes in the fringe of a SBB tree with N keys that corresponds to the tree collection of Fig. 3.3.1 is

$$\bar{f}(N) = 0.85465N+0.85465+0(N^{-5.96})$$

Proof. Similar to the proof of Theorem 6.2.6. □

We end this section with the following remarks.

(i) The results on split transformations obtained in the first-order analysis cannot be improved in the second-order analysis, because there is no possibility of a split transformation at the second level of any type in the tree collection shown in Fig. 3.3.1. (This fact is the key point that permits the proof Theorem 3.3.1.)

(ii) A third-order analysis seems difficult to obtain because of the large number of types involved in a tree collection of SBB trees with h-height 3. A tree collection of SBB trees of h-height 3 which have as subtrees the 30 types of the h-height 2 tree collection of Fig. 3.3.1 contains $n[n(n+1)/2] = 13950$ types, since $n = 30$.

4. Experimental results

The performance evaluation of SBB trees was obtained by means of a sufficient number of repetitions (for different tree sizes) of the following experiment: a permutation of an ordered list is presented one key at a time to the procedure that inserts keys into an initially empty tree; this is followed by the presentation of another permutation to the procedure that deletes keys from the tree just constructed. In other words, there are N insertions followed by N deletions.

Consequently, all observations of inserting (deleting) the ith ($i \leqslant N$) node into a tree are independent events, which eliminates correlation in the simulation results. A similar type of design was used in an experimental study of AVL trees.[23] In order to generate random permutations of an ordered list we used the algorithm presented in Ref. 24. After insertions of the Nth node into the tree the following values are tabulated:

(i) the average number of comparisons in an unsuccessful search (C'_n);

(ii) the average number of comparisons in a successful search (C_n);

(iii) the length of the longest path (i.e. the ordinary height);

Table 4.1 presents results for trees of various sizes.

Table 4.1. SBB tree statistics (expected number of comparisons)

n	C'_n	Variance
5	2.6667 ± 0.0003	0
10	3.5509 ± 0.0032	0.0005
50	5.8549 ± 0.0051	0.0030
100	6.8621 ± 0.0053	0.0022
500	9.2234 ± 0.0059	0.0014
1000	10.2435 ± 0.0063	0.0010
5000	12.6056 ± 0.0073	0.0010
10000	13.6273 ± 0.0084	0.0009

(*a*) Expected unsuccessful search

n	C_n	Variance
5	2.2000 ± 0.0003	0
10	2.9057 ± 0.0035	0.0005
50	4.9720 ± 0.0051	0.0031
100	5.9307 ± 0.0054	0.0023
500	8.2419 ± 0.0059	0.0014
1000	9.2537 ± 0.0062	0.0010
5000	11.6081 ± 0.0073	0.0010
10000	12.6287 ± 0.0083	0.0009

(*b*) Expected successful search

n	Longest path	Variance
5	3.0000 ± 0.0198	0
10	4.0229 ± 0.0222	0.0225
50	7.0089 ± 0.0163	0.0311
100	8.0933 ± 0.0330	0.0849
500	11.0267 ± 0.0259	0.0261
1000	12.1400 ± 0.0684	0.1216
5000	15.0143 ± 0.0280	0.0143
10000	16.1800 ± 0.1076	0.1506

(*c*) Expected worst case search

In addition, the following values are tabulated in order to estimate the cost of maintaining the properties of SBB trees

(*a*) on insertion:

(i) the percentage of insertions that caused a transformation to be performed, and

(ii) the number of nodes revisited to restore the tree property, counted from the father of the node inserted into the tree to the node at which the retreat terminated.

(*b*) on deletion.

(iii) the percentage of deletions that caused a transformation to be performed, and

(iv) the number of nodes revisited to restore the tree property, counted from the node to be deleted from the tree to the node at which the retreat terminated. Sometimes the retreat terminates immediately at the node to be deleted; this is not counted as a retreat. For instance, if the node to be deleted is pointed at by a horizontal pointer, the only operation to perform is to replace that pointed by nil. (If the node to be deleted has two subtrees, it is first interchanged with the rightmost node of its left subtree before deletion.)

Table 4.2. Insertion and deletion statistics for SBB trees of 10000 nodes

	Mean	Variance
Insertion		
Rearrangements	0.3880 ± 0.0011	0
Splits	0.5119 ± 0.0007	0
Nodes revisited	2.4109 ± 0.0019	0.0001
Deletion		
Rearrangements	0.2091 ± 0.0011	0
Splits	0.0018 ± 0.0001	0
Nodes revisited	0.8596 ± 0.0014	0

Table 4.2 presents the insertion and deletion results. These results are actually for trees of 10000 nodes only because they have been shown to be asymptotically independent of the number of nodes in the tree. (In fact, for trees greater than approximately 50 nodes, the results approach these.)

5. CONCLUSIONS

As a practical structure symmetric binary B-trees have been considered as an option for representing dictionary information. In Section 2 we present an improved version of the insertion and deletion algorithms for SBB trees.

In the first part of Section 3 we present the fringe analysis technique. We use the theorem which shows that the matrix recurrence relation related to fringe analysis problems converges to the solution of a linear system involving the transition matrix, even when the eigenvalues of the transition matrix are not pairwise distinct.[20]

In the remainder of Section 3 we accomplish a higher-order analysis of the improved insertion algorithm for SBB trees. We obtain a closed tree collection containing 30 types, and derive results on the expected number of transformations per insertion and on the expected number of balanced nodes.

In Section 4 we present experimental results for the improved versions of the insertion and deletion algorithms. We examine empirically the expected costs to search, insert and delete keys from SBB trees.

Acknowledgements

The work of the first author was supported by the Financiadora de Estudos e Projetos, Brazil (grant no. DCC-FINEP-383), and of the third by the Natural Sciences and Engineering Research Council of Canada (grant no. A-3353).

REFERENCES

1. R. Bayer and E. McCreight, Organization and maintenance of large ordered indexes. *Acta Informatica* **1** (3), 173–189 (1972).
2. D. E. Knuth, *The Art of Computer Programming*, vol. 3. Addison-Wesley, Reading, Mass. (1973).
3. R. Bayer, Binary B-trees for virtual memory. *Proc. 1971 ACM SIGFIDET Workshop*, San Diego 219–235 (1971).
4. R. Bayer. Symmetric binary B-trees: data structure and maintenance algorithms. *Acta Informatica* **1** (4), 290–306 (1972).
5. N. Wirth. *Algorithms + Data Structures = Programs*. Prentice-Hall, New Jersey (1976).
6. L. J. Guibas and R. Sedgewick, A dichromatic framework for balanced trees. *19th Annual Symposium on Foundations of Computer Science* (1978).
7. S. Huddleston and K. Mehlhorn, *Robust Balancing in B-Trees. Lecture Notes in Computer Science* 104, Springer-Verlag 234–244 (1981).
8. S. Huddleston and K. Mehlhorn, A new data structure for representing sorted lists, *Acta Informatica* **17** 157–184 (1982).
9. R. E. Ladner, Private communication (1980).
10. H. Olivié, *Symmetric Binary B-Trees Revisited. Report 80–01*, Interstedelijke Industriële Hogeschool Antwerpen-Mechelen, Antwerp, Belgium (1980).
11. H. Olivié, On random son-trees. *International Journal of Computer Mathematics* **9** 287–303 (1981).
12. N. Ziviani and F. Tompa, A look at symmetric binary B-trees. *INFOR – Canadian Journal of Operational Research and Information Processing* **20** (2), 65–81 (1982).
13. H. Olivié and M. Schrapp, *Path-Length Balanced Trees. Report 82–02*, Stedelijke Industriële Hogeschool Antwerpen-Mechelen, Antwerpen, Belgium (1982).
14. H. Olivié, A new class of balanced search trees: half balanced binary search trees. *RAIRO Theoretical Informatics* **16** (1) 51–71 (1982).
15. R. E. Tarjan, Updating a balanced search tree in 0(1) rotations. *Information Processing Letters* **16** (5) 253–257 (1983).
16. A. Yao, On random 2–3 trees. *Acta Informatica* **9** 159–170 (1978).
17. N. Ziviani, The fringe analysis of search trees. *PhD. Thesis, Report CS*-82-15, Department of Computer Science, University of Waterloo, Canada (1982).
18. D. R. Cox and H. D. Miller. *The Theory of Stochastic Processes*. Chapman and Hall, London (1965).
19. W. Feller. *An Introduction to Probability Theory and its Application*, vol. 1. Wiley, New York (1968).
20. B. Eisenbarth, N. Ziviani, G. H. Gonnet, K. Mehlhorn and D. Wood. The Theory of Fringe Analysis and its application to 2–3 trees and B-trees. *Information and Control* **55** (1–3) 125–174 (1982).
21. G. H. Gonnet, N. Ziviani and D. Wood, *An Analysis of 2–3 trees and B-trees. Report CS*-81-21, Department of Computer Science, University of Waterloo, Canada (1981).
22. B. Eisenbarth, Allgemeine Fringe-analysis und ihre Anwendung zur Untersuchung der Overflowtechnik bei B-Bäumen. *Master's Thesis*, Universität des Saarlandes, Saarbrucken, West Germany (Adviser K. Mehlhorn) (1981).
23. P. L. Karlton, S. H. Fueller, R. E. Scroggs and E. B. Kaehler. Performance of height-balanced trees. *CACM* **19** (1) 23–28 (1967).
24. R. Durstenfeld. Random Permutation. *Algorithm 235, CACM* **7** (7) 420 (1964).