A PROBABILISTIC METHOD FOR ANALYZING SEARCH TREES

By

DALE LEROY MITCHELL, JR.

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1973

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
July, 1975

Thesis
1975
M681p
cop. 2

A PROBABILISTIC METHOD FOR ANALYZING

SEARCH TREES

Thesis Approved:

James R. Van Doren
Thesis Advisor

[illegible]

[illegible]

[illegible]
Dean of the Graduate College

PREFACE

This thesis presents a probabilistic method for analyzing search trees and applies this method to some constrained classes of limited branching search trees. It is hoped that this method of analysis may help in gaining insight into the expected behavior of search trees when they are utilized as storage structures in information storage and retrieval.

I would especially like to thank my major advisor, Dr. James Van Doren for providing the foundation on which this study is built and for the guidance, time, and interest he lent toward its completion.

Thanks to Dr. Grace and Dr. Hedrick both as committee members and as instructors who provided a scholastic atmosphere in which it was a pleasure to learn.

A note of thanks to Jane Van Wye for her patience, care, and the long hours she devoted to typing this manuscript for submission.

I am indebted to Eugene Bailey for all the instruction, tolerance, and friendship he has unselfishly given to me.

I am most grateful to my parents for their love and devotion, and for providing me with security and encouragement in all my educational pursuits. I am also grateful to Sheila for her care and understanding during the past two years.

I wish to express my appreciation to the United States Army for enabling me to attend graduate school through their graduate fellowship program.

I want to include a general expression of gratitude to all my friends, associates, fellow students and professors who have made my years at Oklahoma State University an enjoyable and educational experience.

TABLE OF CONTENTS

LIST FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Storage and retrieval of information in a computing system require the physical structuring of data in a form which may be readily accessed and maintained while making efficient use of available storage space. Tree structuring is one method of organizing large amounts of information to attain these ends. The physical data structure, or storage structure, of any given set of information records refers specifically to the scheme in which those records are arranged in a storage medium. Only the physical arrangement of data in tree structures will be considered in this paper.

The basic concept of tree structuring involves a set of indices to a set of data records which are based on the collating sequence of keys identifying the records. Starting with a general index, a search is made for an appropriate pointer to a more specific index. The process is then repeated at this level and as many subsequent levels as necessary to locate a specific element of the original data set. Knowing what the expected performance of a given tree structure will be if it is implemented makes possible wise choices of the physical data structures used in information systems. This study presents a method involving probabilistic analysis for obtaining the expected asymptotic performance characteristics of several constrained classes of tree structures which are suitable for organizing information in internal storage. The classes of search trees covered and their order of treatment will be as follows: 1) B-trees of branching order three (3-2 trees); 2) Symmetric Binary B-trees, henceforward called SBB-trees; and, 3) balanced binary trees (also known as AVL trees). A brief description of these structures is presented here. Further detail will be added in later chapters as needed.

The general class of tree structures known as B-trees was introduced in 1972 by Bayer and McCreight (1) and has since been the object of continued interest and research (e.g., Bayer (2) and Davis (3)). B-trees have a predetermined maximum number of branches from each node and are maintained such that all paths from the root node to the leaf nodes are of uniform length. 3-2 tree is the name given to that subclass of B-trees which has a maximum of three branches and two keys for each node. The example 3-2 tree shown in Figure 1 has a height of three which means that the contents of three nodes must be inspected to determine the insertion position for a key that is not in the tree. For any tree structure the term "root" applies to the node in the tree from which all other nodes descend. The term "leaf" refers to nodes which have no descendants. Furthermore, the descendants of a given node are its "sons" and it is their "father." This genealogical terminology may be carried to any level in a tree structure to describe the relationship between two nodes. For 3-2 trees as well as all classes of B-trees, the leaf nodes are all on the same level with respect to the root node.

SBB-trees, as described by Bayer (2), are directed binary trees which may contain two types of pointers, δ-pointers or vertical pointers and ρ-pointers or horizontal pointers. Figure 2 shows an

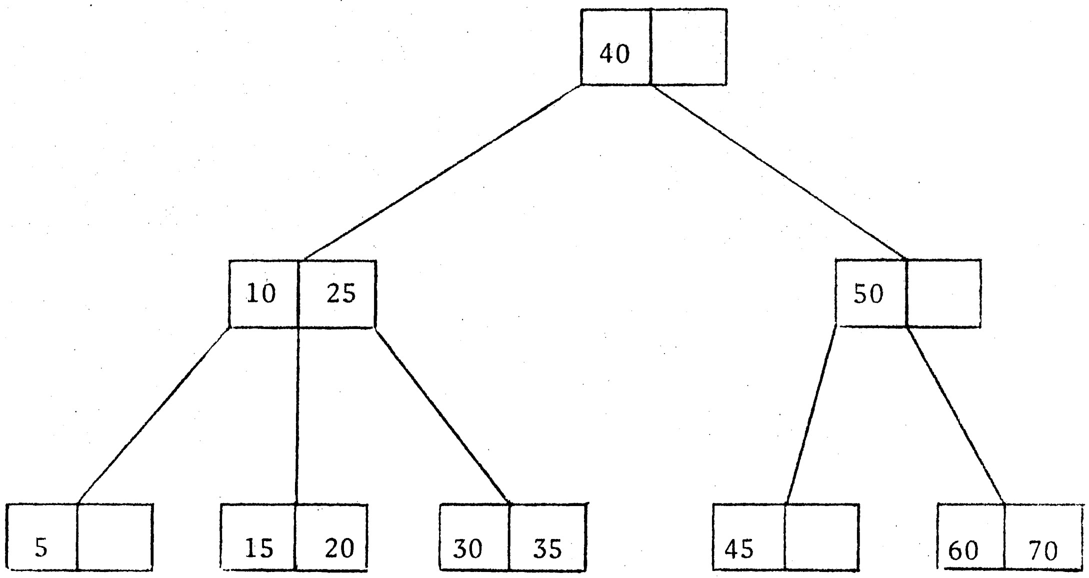


Figure 1. A 3-2 Tree

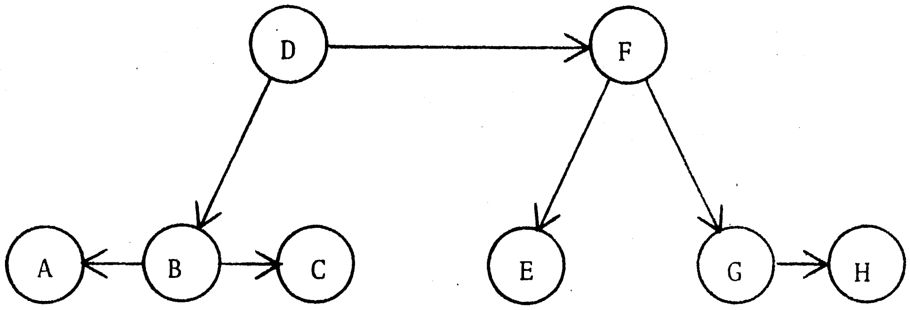
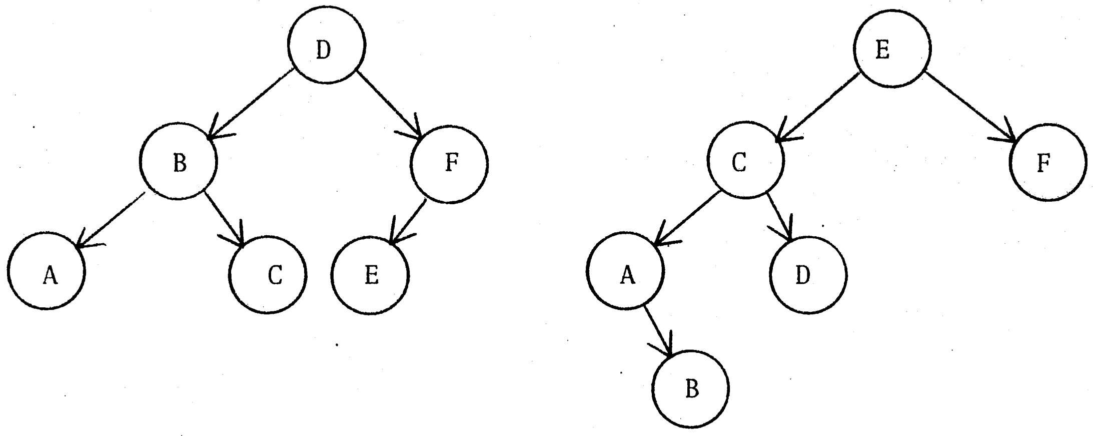


Figure 2. An SBB-Tree

example SBB-tree. Each node may have zero, one, or two sons; thus, SBB-trees are binary trees. Since the number of downward or δ-levels from the root node to any leaf in the tree is constant, SBB-trees can be viewed as a modified class of B-trees. A more thorough description of SBB-trees is presented in Chapter III.

AVL trees were developed by G. M. Adelson-Velskii and E. M. Landis as a constrained class of binary trees in which the maximum number of levels in the left and right subtrees of any node in the AVL tree could differ by no more than one. The binary tree in Figure 3a is an AVL tree. The binary tree in Figure 3b is not an AVL tree, because the left subtree of node E has a maximum height of three while its right subtree has a maximum height of one. AVL trees are sometimes referred to as balanced trees since the constraints which define them tend to maintain a fairly even left-right dispersion of nodes. Analysis of AVL trees by Knuth (7), and by Van Doren and Gray (10) has shown that AVL trees are effective structures for organizing and maintaining data. The following chapters will explore probabilistic methods for analyzing some aspects of the performance of all of the above data structures.

Development of the basic method of analysis is presented in Chapter II with application to 3-2 trees. It is further extended in Chapter III to cover SBB-trees. SBB-trees are thoroughly described in this chapter and are interpreted, for purposes of analysis, as a constrained class of 4-3 trees. Chapter IV deals with the analysis of AVL trees and their relationship to SBB-trees. A summary including conclusions and suggestions for further research is presented in Chapter V. Explanations and proofs for several probability theorems



a.) An AVL Tree b.) A Non-AVL Tree

Figure 3. Comparison of Balanced and Unbalanced Binary Trees

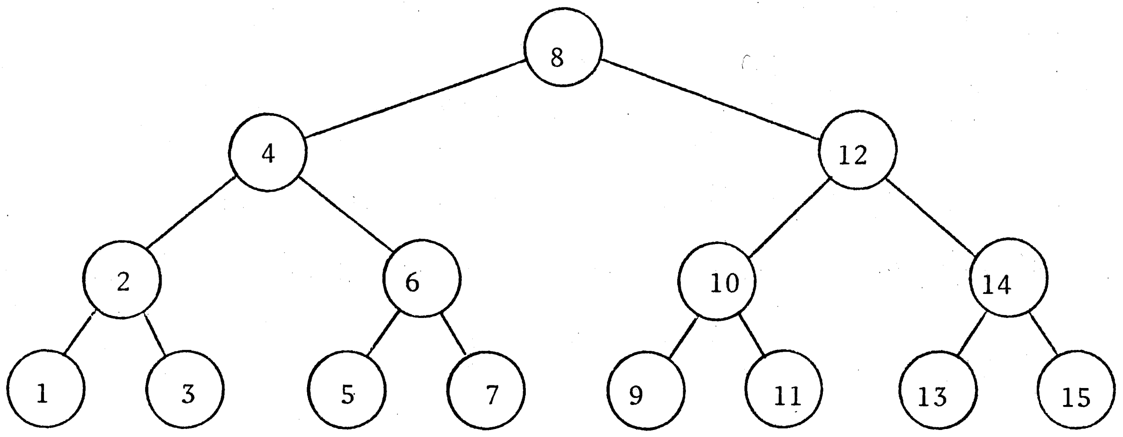
used in the analysis are contained in an Appendix.

The purpose in arranging the material in this order is to gain continuity in presentation and to obtain a logical order of development so that each succeeding chapter builds on the previously covered topics. The method of analysis developed in Chapter II for 3-2 trees is applied in Chapters III and IV to SBB-trees and AVL trees, respectively. The concepts and analysis dealing with SBB-trees in Chapter III are correlated with AVL trees in Chapter IV and are utilized to investigate some of the properties of AVL trees.

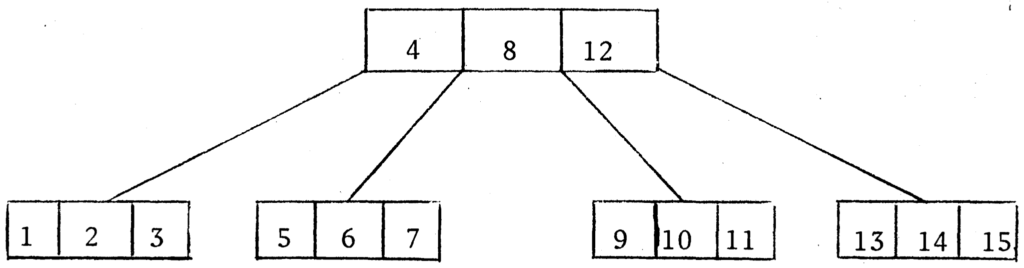
CHAPTER II

A PROBABILISTIC ANALYSIS OF 3-2 TREES

The performance of a given tree structure in a particular information system is dependent upon such factors as the arrangement and type of hardware being used, the type and amount of data stored in each data element, etc. However, there are two factors inherent in the structure of any tree which may be viewed independently of specific applications. These factors are the number of node levels in the paths from the root node to the leaf nodes, and the number of descendants that any node in the tree may have. The number of levels in the longest path of a tree is referred to as its height. The maximum number of sons that a tree node may have is referred to as its degree of branching. Tree height and degree of branching are inversely related. For example, the tree in Figure 4a has a branching degree of two and a height of four. In Figure 4b the same information is contained in a tree of height two where the degree of branching has been increased to four. Although the second tree requires fewer node accesses from root to leaf, the effort required to determine which pointer to follow out of a node has been increased. In this study only trees of very limited branching (≤ 4) will be treated. Therefore, the primary area of interest and analysis will be to determine something about the expected number of tree levels. 3-2 trees, as previously stated, comprise a class of B-trees for which the maximum degree of branching is three. The following analysis of 3-2 trees will illustrate a method of analysis for tree structures in general which makes use of a probabilistic process called a Markov chain.



a.) Tree with Height Four and Degree of Branching Two



b.) Tree with Height Two and Degree of Branching Four

Figure 4. Comparison of Tree Height and Degree of Branching

Observations About 3-2 Trees

As depicted in the example of Figure 5, a 3-2 tree may have one or two keys per node and two or three branches from any node. For convenience in description, an additional set of "external" nodes can be considered to exist as the sons of the leaf nodes at the lowest internal level of the tree. These nodes are indicated by the small

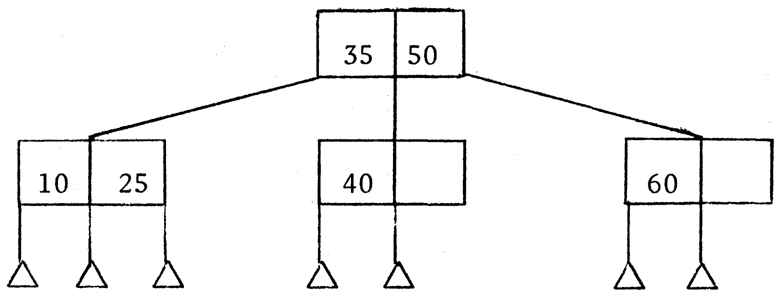
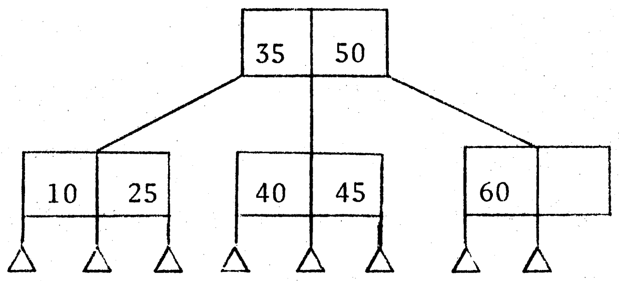


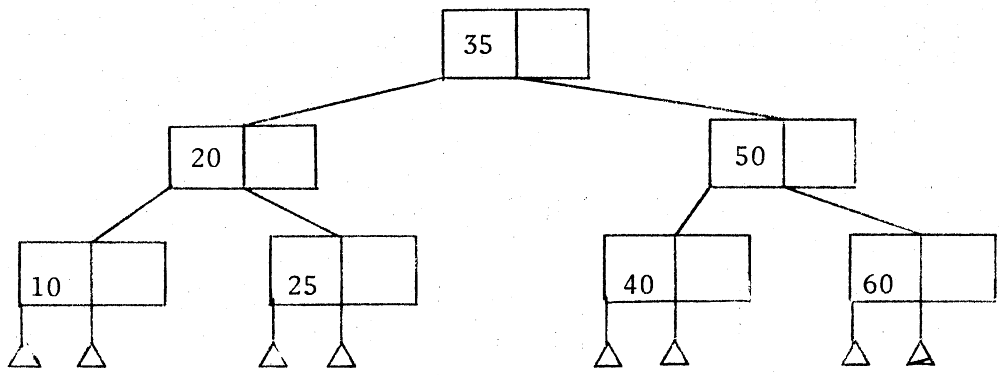
Figure 5. 3-2 Tree Showing External Node Positions

triangles in Figure 5 but their contents are not important. First it should be noted that for any B-tree the number of external nodes will be one greater than the number of keys in the tree. This follows if the external nodes are viewed as possible insertion positions for new keys and the observation is made that, as for a linear list, it is possible to insert a new key between any two keys already in the tree, as well as one before the first key and one after the last key. In the tree of Figure 5 there are six keys and seven external nodes. Secondly it may be observed that the external nodes for a 3-2 tree fall into two distinct classes, those with one key in their father node and those with two keys in their father node. The observation and classification of external node types is an important subject with respect to the probabilistic analysis presented.

The next observations made concern the effects on the external nodes of a 3-2 tree that occur when a new key is inserted in the tree. Algorithms for updating B-trees when insertion takes place have been described by Bayer and McCreight (1) and by Davis (3). There are many variations and refinements of the basic operations which may improve tree performance in some instances, but only the basics will be considered here. A new key is inserted at the internal leaf node level once the appropriate position relative to the keys already present in the tree has been determined. For example, if the key 45 is to be inserted in the 3-2 tree of Figure 5, the root node is examined and key 45 is found to be between key 35 and key 50. The corresponding pointer is followed to the node containing key 40 and this node is examined. Since key 45 is greater than key 40 and the node being examined is a leaf node, the new key is inserted as shown in Figure 6a. If the key to be inserted had been less than key 40 but greater than key 35, insertion would still occur in the same node with the new key being placed in the key slot occupied by key 40 and key 40 being moved to the right most key slot in the node. In both cases the node is changed from a one-key node to a two-key node which changes the classification of the external nodes that are its descendants. The net effect of this process is to destroy two external



a.) After Insertion of Key 45



b.) After Insertion of Key 20

Figure 6. Effects of Insertion on the 3-2 Tree of Figure 5

nodes with one key in their father and to create three external nodes with two keys in their father. Each insertion of a new key adds one to the total number of external nodes maintaining the relationship of N+1 external nodes for a tree containing N keys. Comparison of Figures 5 and 6a illustrates these observations. The tree of Figure 5 has seven external nodes, three external nodes with two-key fathers, and four external nodes with one-key fathers. The tree of Figure 6a has eight external nodes, two external nodes with one-key fathers and six external nodes with two-key fathers. The only exception to the insertion process described above occurs when the insertion position for a new key falls inside a node that already contains two keys. When this situation arises the overfull node is split into two nodes containing one key each and the middle key of the original three is inserted in the father node. If the father node becomes overfull as a result of this insertion, it also may be split and the process may be propagated up as many tree levels as necessary. To illustrate this process key 20 will be inserted in the tree of Figure 5 resulting in the restructured tree of Figure 6b. Since key 20 falls between key 10 and key 25, its insertion creates an overfull node which is split. Key 20 is the middle key so it is promoted for insertion in its father node. This also creates an overfull node which is split, with the middle key, key 35, being inserted in a new root node. The original node where insertion occurred was changed from a two-key node into two one-key nodes thus causing the destruction of three external nodes with two-key fathers and the creation of four external nodes with one-key fathers. The tree in Figure 5 has seven external nodes, three with two-key fathers, and four with one-key fathers as compared to the tree in Figure 6b which has eight external nodes, all eight having one-key fathers.

Concept of States

Since the external nodes of a 3-2 tree fall into one or the other of two classes, a useful concept is the "state" of an external node. A given external node may be in state #1, having one key in its father; or, it may be in state #2, having two keys in its father. Furthermore, upon insertion of a new key a given external node may remain in its current state or, as shown above, it may pass to the other state. This information may be represented as the directed graph of Figure 7. In order to analyze further the expected behavior of the tree, weighting factors must be determined for each of the digraph paths. These weights will be the probabilities that the corresponding state transitions will occur when an insertion is made. The following section develops the state transition probabilities for a generalized 3-2 tree.

State Transition Probabilities

For an arbitrary 3-2 tree containing N keys, it has been shown that there will be N+1 external nodes. The fraction of these nodes which are in state #1 at a particular time will be represented by P_{N+1} and the fraction that are in state #2 will be represented by Q_{N+1}. Since these states are mutually exclusive and collectively exhaustive, the sum of P_{N+1} and Q_{N+1} is one. Therefore, if a new key is equally likely to be inserted in the tree at any of the external node positions, P_{N+1} and Q_{N+1} may be viewed as the elements of a probability vector describing the tree in terms of its external nodes. The condition that keys to be inserted in the tree be chosen randomly from a uniform distribution insures that each of the external node positions has the same probability as an insertion candidate. A tree constructed by repeated insertion of keys chosen in this manner is said to be randomly generated. For a randomly generated 3-2 tree containing N keys, P_{N+1} and Q_{N+1} represent the probabilities that a given external node is in state #1 or state #2 respectively; also, $(N+1) \times P_{N+1}$ and $(N+1) \times Q_{N+1}$ represent the probable number of external nodes that are in each state.

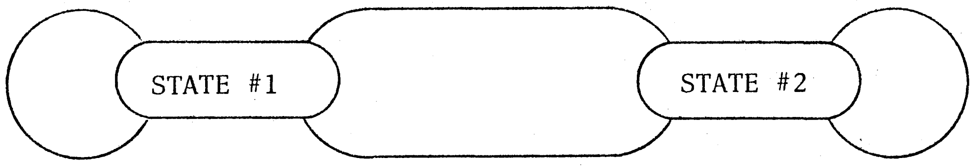


Figure 7. Digraph Representation of State Transitions

The state transition probabilities are derived by relating the state of a 3-2 tree described by (P_{N+1}, Q_{N+1}) with the state of the same tree as it existed immediately prior to the last key insertion. At that time there were N external nodes (one less than the current number) and the fractions of external nodes, P_N and Q_N in each of the two possible states was somewhat different. The number of external nodes in state #1 in the previous tree plus or minus the number of state #1 nodes created or destroyed by the insertion process. In equation form this would appear as:

$$(N+1) \times P_{N+1} = (N x P_N) - (2 x P_N) + (4 x Q_N) \qquad (2.1)$$

Stated verbally this equation says that the probable number of external nodes in state #1 for the tree after insertion is equal to NxP_N, the probable number in that state prior to insertion, minus the probable number of state #1 nodes destroyed by insertion, $2xP_N$, plus the probable number of state #1 nodes created by insertion, $4xQ_N$. Similarly for external nodes of state #2 the probability equation would be:

$$(N+1) x Q_{N+1} = (N x Q_N) + (3 x P_N) - (3 x Q_N) \qquad (2.2)$$

The correctness of the coefficients in (2.1) and (2.2) can be verified graphically by comparison of the trees in Figures 5 and 6. Also, the sum of equations (2.1) and (2.2) must be equal to N+1.

Algebraic manipulation of equations (2.1) and (2.2) results in the following matrix equation:

$$(P_{N+1}, Q_{N+1}) = (P_N, Q_N) \cdot \begin{bmatrix} \frac{N-2}{N+1} & \frac{3}{N+1} \\ \frac{4}{N+1} & \frac{N-3}{N+1} \end{bmatrix} \tag{2.3}$$

The general form of this equation is:

$$S_{N+1} = S_N \cdot T(N) \tag{2.4}$$

where S_N and S_{N+1} represent respectively the probability vectors for the external node states of a given 3-2 tree before and after the insertion process, and T(N) is a probability transition matrix that is dependent on N. The elements of T(N) may be interpreted as the weights for the state transition paths of the digraph in Figure 7 as follows:

$$\begin{aligned} \frac{N-2}{N+1} &= \text{transition probability from state \#1 to state \#1} \\ \frac{3}{N+1} &= \text{transition probability from state \#1 to state \#2} \\ \frac{4}{N+1} &= \text{transition probability from state \#2 to state \#1} \\ \frac{N-3}{N+1} &= \text{transition probability from state \#2 to state \#2} \end{aligned} \tag{2.5}$$

Now that the values have been determined defining the transition between consecutive states for the insertion process, it will be advantageous to determine whether the probability vector S_{N+1} approaches some set of values asymptotically; that is, whether the state of the tree stabilizes after a large number of insertions. For this analysis the construction of a 3-2 tree by repeated insertions of randomly generated keys will be viewed as a particular type of probabilistic process known as a Markov chain. Alternatively it would be possible, as shown by Liu (8), to interpret the recurrence relationships as a class of nonlinear difference equations; however, the terminology of Markov chains is used in the analysis given below.

Concepts of Markov Chains

A discrete finite stochastic process is a process that occurs in a finite number of steps, where the outcome of each step may depend on the outcomes of the preceding steps. For example, if there are three colored golf balls, one red, one yellow, and one green, in a box, and they are drawn at random without replacement, the possible outcomes after three draws are as shown in Figure 8. The probability of each possible outcome of a given step is known when the outcomes of all previous steps are known. If it is known that the outcome of the first step in the above example was that a yellow ball was drawn, then the possible outcomes of the next step are drawing a red ball and drawing a green ball, each with one-half probability of occurrence. If the probabilities for the possible outcomes of a particular step of a stochastic process depend only on the outcome of the immediately preceding step then the process is called a Markov process. Additionally, if the transition probabilities are constant for every step in the process, the process is called a homogeneous Markov chain. If the transition probabilities vary according to some process related parameter then the Markov chain is nonhomogeneous. Detailed formal definitions of all of these processes and their relationships are given by Kemeny and Snell (5) and by Parzen (9). The insertion of a key in a 3-2 tree as pictured in Figure 7 and defined by matrix equations (2.3) and (2.4) is a finite nonhomogeneous Markov chain which is dependent on N, the number of keys in the tree.

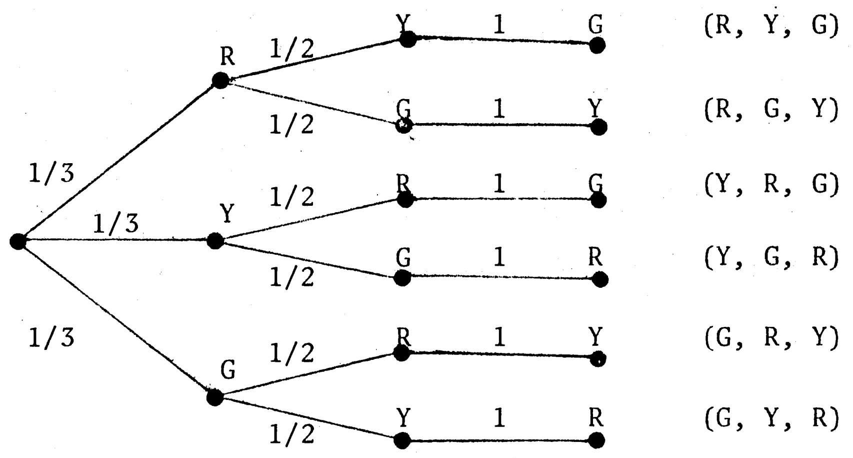


Figure 8. A Discrete Finite Stochastic Process

The property of this Markov chain which is of interest is known as ergodicity, the property of stabilization after a large number of steps regardless of the initial state. If the Markov chain model of 3-2 tree insertion is ergodic the fraction of external nodes in each state will stabilize in a statistical sense after a sufficient number of insertions regardless of the initial fractional values. Such long term values may give valuable insight into the expected performance of a 3-2 tree.

To examine the manner in which a Markov chain proceeds, the 3-2 tree of Figure 6a will serve as an initial state. This tree has seven keys and eight external nodes, six with two-key fathers and two with one-key fathers. Placing these values into equation (2.3) yields:

$$(P_9, Q_9) = (2/8, 6/8)\cdot\begin{bmatrix} 5/8 & 3/8 \\ 4/8 & 4/8 \end{bmatrix} \tag{2.6}$$

Solving this equation for the values of P_9 and Q_9 yields:

$$P_9 = 17/32 \quad , \quad Q_9 = 15/32 \tag{2.7}$$

which are the probable fractions of external nodes in states #1 and #2 respectively for the tree after an eighth key has been inserted. Using these values for the next step in the chain (insertion of the ninth key) yields:

$$(P_{10}, Q_{10}) = (17/32, 15,32)\cdot\begin{bmatrix} 6/9 & 3/9 \\ 4/9 & 5/9 \end{bmatrix} = (2/8, 6/8)\cdot\begin{bmatrix} 5/8 & 3/8 \\ 5/8 & 4/8 \end{bmatrix}\cdot\begin{bmatrix} 6/9 & 3/9 \\ 4/9 & 5/9 \end{bmatrix} \tag{2.8}$$

From (2.8) it can be seen that the probability vector values after the n-th step of the process will be:

$$(P_{N+1}, Q_{N+1}) = (P_i, Q_i)\cdot T(i)\cdot T(i+1)\cdot\ldots\cdot T(n) \tag{2.9}$$

or

$$(P_{N+1}, Q_{N+1}) = (P_i, Q_i)\cdot \prod_{k=i}^{n} \cdot\, T(k) \tag{2.10}$$

for some initial probability vector (P_i, Q_i). Following the development in Theorem 2 of the appendix a Markov chain will be ergodic if the product of the series of transition matrices approaches some limit in which all the rows are identical (all elements of a column are identical) as the number of steps in the chain approaches infinity. Equation (2.10) would then be:

$$(P_{N+1}, Q_{N+1}) = (P_i, Q_i)\cdot\begin{bmatrix} a & b \\ a & b \end{bmatrix} \tag{2.11}$$

solving for P_{N+1} and Q_{N+1} yields:

$$P_{N+1} = a \cdot P_i + a \cdot Q_i = a \cdot (P_i + Q_i) = a \tag{2.12}$$

$$Q_{N+1} = b \cdot P_i + b \cdot Q_i = b \cdot (P_i + Q_i) = b \tag{2.13}$$

which show that regardless of the initial values of the state probability vector, the long term values will be stable. The long term 3-2 tree insertion process can thus be expressed as:

$$\alpha = \alpha \cdot T(N) \tag{2.14}$$

Where the values of α are the steady state values of P and Q. In effect this says that once statistical stability is reached the system remains statistically stable regardless of additional steps.

3-2 Tree Analysis

Expanding equation (2.14) for the T(N) obtained in equation (2.3) and utilizing the relationship:

$$P + Q = 1 \tag{2.15}$$

the stable state values of P and Q may be solved for as follows:

$$((N-2)/(N+1)) \cdot P + (4/(N+1)) \cdot Q = P \tag{2.16}$$

$$(3/(N+1)) \cdot P + ((N-3)/(N+1)) \cdot Q = Q \tag{2.17}$$

Multiplying through by (N+1) in (2.16) and (2.17) yields:

$$((N-2) \cdot P) + (4 \cdot Q) = P \cdot (N+1) \tag{2.18}$$

$$(3 \cdot P) + ((N-3) \cdot Q) = Q \cdot (N+1) \tag{2.19}$$

Algebraic manipulation gives:

$$4 \cdot Q = 3 \cdot P \tag{2.20}$$

and from (2.15)

$$Q = 1 - P \tag{2.21}$$

so that by substitution:

$$4\cdot(1-P) = 3\cdot P \tag{2.22}$$

giving:

$$P = 4/7 \tag{2.23}$$

and:

$$Q = 3/7 \tag{2.24}$$

The validity of these values may be tested by substituting them for P_N and Q_N in equation (2.3) and noting that the resulting values of P_{N+1} and Q_{N+1} are likewise, 4/7 and 3/7.

If for a randomly generated 3-2 tree of N keys (where N is large), the expected fraction of external nodes with one-key fathers is 4/7 and the expected fraction of external nodes with two-key fathers is 3/7 then the expected number of external nodes in each class will be $4/7\cdot(N+1)$ and $3/7\cdot(N+1)$ respectively. Furthermore, the fact that there are two external nodes for each one-key father indicates that the expected number of one-key nodes at the bottom internal level of the tree is $1/2\cdot4/7\cdot(N+1)$ or $2/7\cdot(N+1)$. Likewise, since there are three external nodes for each two-key father, the expected number of two key nodes, at the bottom internal level is $1/3\cdot3/7\cdot(N+1)$ or $1/7\cdot(N+1)$. The expected total number of nodes at the bottom internal level would therefore be $2/7(N+1)+1/7(N+1)$ or $3/7(N+1)$. Adding the number of one-key nodes to two times the number of two-key nodes gives $2/7\cdot(N+1)+2\cdot1/7\cdot(N+1)$ or $4/7\cdot(N+1)$ which is the expected number of keys in the bottom internal level. Subtracting this quantity from N, the number of keys in the tree, gives $N-4/7(N+1)$ or $3/7N-4/7$ expected keys in the upper levels of the tree.

Extension of Analysis To Upper Tree Levels

The specific information gained from the preceding analysis is primarily related only to the bottom internal level of the tree. In order to extend these concepts to levels above the lowest one the states of the nodes in these levels must be used in defining the states of the external nodes on which the analysis is performed. For the analysis of the bottom two levels of a 3-2 tree it would be possible to define each external node by a four-tuple (m, n, o, p), where m is the number of keys in the grandfather node and n, o, and p are the numbers of keys in each of the sons of the grandfather. From such an n-tuple the probability that an external node has a given number of keys in its father can be determined by a simple ratio. For example, if an external node with two keys in its grandfather is represented by (2, 1, 2, 2) then the probability that its father is a two-key node is the ratio of the number of external nodes with two-key fathers in this situation (6) to the total number of external nodes below the two-key grandfather (8), or .75. Analysis of this type yields eight distinct states for the external nodes of a 3-2 tree resulting in an eight-by-eight transition matrix. This analysis would still be reasonable but, as before, its results do not apply to the tree as a whole unless the tree is of only two levels. Since the method of analysis is fundamentally based on trees containing a large number of keys and since the number of external node states increases cominatorially as additional levels are considered, it is easy to see that exact analysis of a whole tree structure would quickly grow out of hand. Although analysis of this magnitude is not completely infeasible, it definitely calls for the application of non-trivial electronic computing techniques.

In order to obtain a more simplified though somewhat less mathematically precise means of analyzing an entire tree, an assumption is made that the behavior of the nodes at each level of the tree structure is probabilistically independent of the other levels in the tree. This means that for any level in the tree the next lower level may be viewed as external and the level in question may be analyzed as the bottom internal level. This is not a completely accurate assumption since it is assumed that any insertion position at the actual bottom level of the tree is equally likely to be filled. In reality, however, this approximation is very close to true for randomly generated trees containing a large number of keys. To support this concept the results of the Markov analysis applied to an entire 3-2 tree can be compared to the results of empirical testing obtained by Davis (3, p. 41) for 3-2 trees generated in the same manner. Davis found the tree utilization to be approximately two-thirds for such trees where utilization is defined as:

$$\text{Utilization} = \#\text{Keys} \; / \; \#\text{Key Slots} \qquad (2.25)$$

Utilization is an important performance measure because it relates the storage space that a structure occupies to the amount of that storage space which actually contains useful information. For B-trees, utilization is inversely related to tree height. Calculation of the utilization of the bottom internal level of a 3-2 tree from the results of the last section is accomplished by dividing the expected number of keys by the product of the expected number of nodes and two key slots per node.

$$\frac{4/7(N+1)}{2\cdot 3/7(N+1)} = 2/3 \qquad (2.26)$$

Extending the behavior of the bottom level to the entire tree gives an expected tree utilization of two-thirds which is consistent with the empirical results.

Using the two-thirds as the expected tree utilization for a 3-2 tree, the average number of keys per tree node will be 1.33. Thus, the average number of branches from a node will be 2.33. For a 3-2 tree of m levels the exected number of keys can then be computed as:

$$N = (1.33)\cdot(2.33)^0+(1.33)\cdot(2.33)^1+\ldots+(1.33)\cdot(2.33)^{m-1} \qquad (2.27)$$

This geometric progression may be solved by multiplying through by (2.33) and solving for N in the following sequence:

$$(2.33)\cdot N = (1.33)\cdot(2.33)^1+(1.33)\cdot(2.33)^2+\ldots+(1.33)\cdot(2.33)^m \qquad (2.28)$$

$$(2.33)\cdot N-N = (1.33)\cdot(2.33)^m-(1.33) \qquad (2.29)$$

$$N = (2.33)^m-1 \qquad (2.30)$$

Thus for a 3-2 tree of N keys the expected number of levels is given by:

$$m = \log_{2.33}(N+1) \qquad (2.31)$$

Comparison of this result with the empirical results obtained by Davis (3) lends additional credence to the approximations given by extension of the analysis to upper tree levels. For a sample of eight randomly generated 3-2 trees containing 300 keys and built according to the same maintenance rules as the 3-2 trees analyzed above, Davis obtained an average utilization of .674052 (3, p. 41). This compares with .666... or 2/3 obtained in equation (2.26) above. For the same sample the tree height was always seven levels (3, p. 49) which is the smallest integer greater than 6.74, the expected number of levels in a 300 key B-tree as calculated from equation (2.31) above.

CHAPTER III

ANALYSIS OF SYMMETRIC BINARY B-TREES

This chapter is devoted to the analysis of a particular data structure defined by Bayer (2) in 1972 as a symmetric binary B-tree. There are two main reasons for analyzing this tree structure in this study. Firstly, it provides a vehicle for illustrating the expansion of the analysis techniques developed in chapter two to B-trees of higher order branching. This is accomplished by interpreting SBB-trees as B-trees of branching order four (4-3 trees). Secondly, due to the binary structure of SBB-trees and their special relationships to AVL trees, the analysis in this chapter will serve as a basis for the detailed comparison of AVL trees and SBB-trees which is developed in chapter four. Possible extensions of probabilistic analysis to AVL and other binary tree structures will be viewed in the light of these comparisons.

Description of SBB-Trees

SBB-trees are directed binary trees in which the keys are ordered such that all nodes in the left subtree of any given node contain keys less than its key and all nodes in the right subtree contain keys greater than its key. The terms node and key may be used interchangeably here since there is exactly one key per node. A pointer between two nodes may be one of two types as shown in Figure 9.

δ-pointers point vertically or downward to the next node while ρ-pointers point horizontally to an adjacent node. The conditions related to nodes and pointers that must be met for a binary tree structure to be an SBB-tree are: 1) there must be the same number of δ-pointers in the paths from the root node to each leaf node, 2) there must be two descendants from each node except those which have no internal node

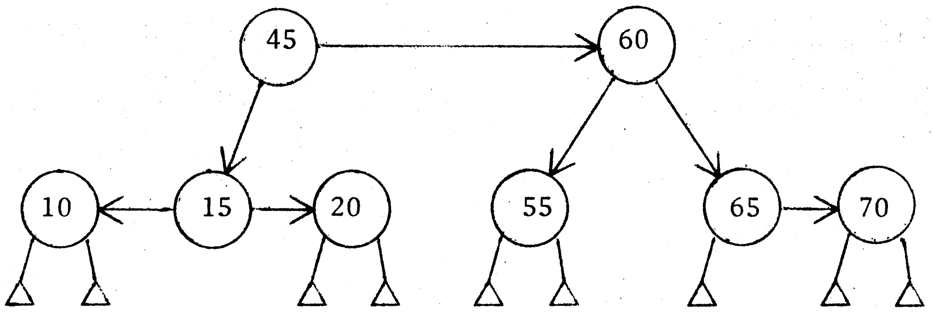


Figure 9. An Example SBB-Tree

descendants pointed to by δ-pointers, and 3) there may never be two consecutive ρ-pointers in a path from the root node to any leaf node. The tree in Figure 9 has two internal δ-levels; therefore, it may be said that the δ-height (h) of the tree is two. This means that a path from the root node, 45, to any leaf node will contain exactly one δ-pointer. The actual height (k) of an SBB-tree, however, is the maximum number of nodes in any path from the root node to the leaf nodes. Thus, the height of the SBB-tree of Figure 9 is four since the longest path contains nodes 45, 60, 65, and 70. The relationship between the height (k) and number of δ-levels (h) is dependent upon the number of ρ-pointers in the longest path of the tree and can be expressed as:

$$h \leq k \leq 2h$$

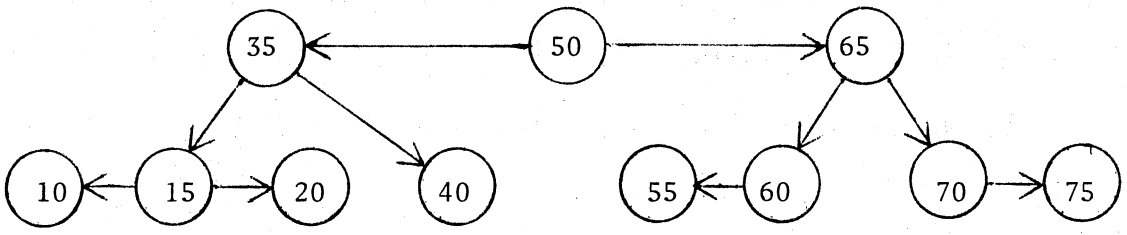
(2, p. 292). Furthermore, the number of δ-levels for an SBB-tree of N keys has been shown by Bayer (2, p. 294) to be bounded by the following relation:

$$\log_2(N+1) \leq h \leq 2\log_2(N+2)-2$$

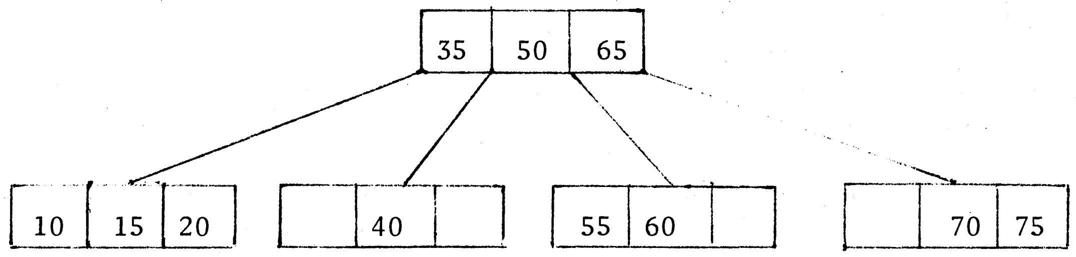
The identification of two types of pointers and the conditions determining when they are used provide a means of constraining a binary tree from becoming excessively unbalanced. It is shown in chapter four that another class of constrained binary trees, AVL trees, is a proper subset of the class of SBB-trees.

Interpretation of SBB-Trees as 4-3 Trees

Knuth (7, p. 469) explains how the concept of δ-pointers and ρ-pointers as developed by Bayer may be applied to give a binary tree representation of 3-2 trees. These same ideas can be extended to SBB-trees and 4-3 trees. If all of the keys in a given SBB-tree which are directly connected by ρ-pointers are grouped together in multi-key nodes as shown in Figure 10, the resulting structure is a 4-3 B-tree. This class of B-tree has a maximum of four downward branches and three keys per node. Although the ρ-pointers have been eliminated, the information they represented has been retained by letting the relative position of the keys within the 4-3 tree nodes be such that the key previously pointed to by a δ-pointer from the next higher δ-level occupies the center key slot in the new 4-3 node and the keys that were connected to it by ρ-pointers occupy the appropriate left or right key slot. This information is important because it is used to determine the proper action to take during maintenance operations. Now that the structure of SBB-trees has been presented the next step will be to observe how this structure is maintained when new keys are inserted.



a.) An SBB-Tree



b.) 4-3 B-Tree Interpretation of above SBB-Tree

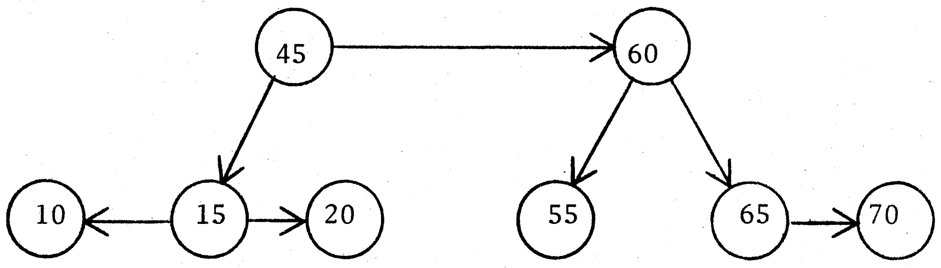
Figure 10. 4-3 Tree Interpretation of an SBB-Tree

Maintenance of SBB-trees

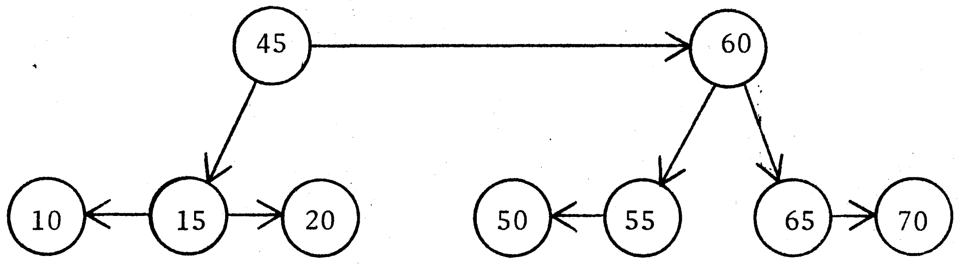
The maintenance algorithms for SBB-trees have been defined by Bayer (2, pp. 297-301) in a somewhat cryptic though mathematically quite precise manner. The following is a verbal description of these algorithms and their extensions to 4-3 trees.

Insertion of keys in an SBB-tree follows much the same procedure as insertion for any other type of tree structure. A traversal of the tree is made searching for the key which is to be inserted. Assuming that there is not a duplicate key in the tree, an external node, which represents an insertion position, will be encountered. The new key is inserted at this position with a ρ-pointer connecting it to its father. The result of this process for inserting key 50 in the tree of Figure 11a is shown in Figure 11b. After the insertion process has been completed, the resultant tree requires no further maintenance if it still conforms to the conditions defining SBB-trees. However, if the father node of the newly inserted key was connected to its father by a ρ-pointer, the insertion creates two successive ρ-pointers which are not allowed by the SBB-tree definition. This situation is depicted in Figure 11c where key 35 has been inserted in the tree of Figure 11a. When two successive ρ-pointers occur the tree must be restructured so that it again fits the SBB-tree definition.

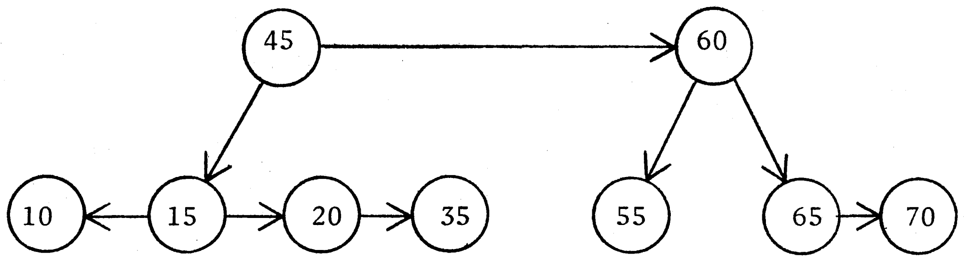
The method of accomplishing restructuring is called "splitting" (2, p. 297) and involves different changes in pointer type, pointer direction, and node relationships for the different cases that arise involving ρ-pointers. These operations for the lowest level in an SBB-tree are pictured in Figure 12. The basic situations which arise and are not allowed are consecutive ρ-pointers in the same direction and consecutive ρ-pointers in opposite directions. Figure 12a shows two of the possible cases of the first situation that may arise



a.) An SBB-Tree

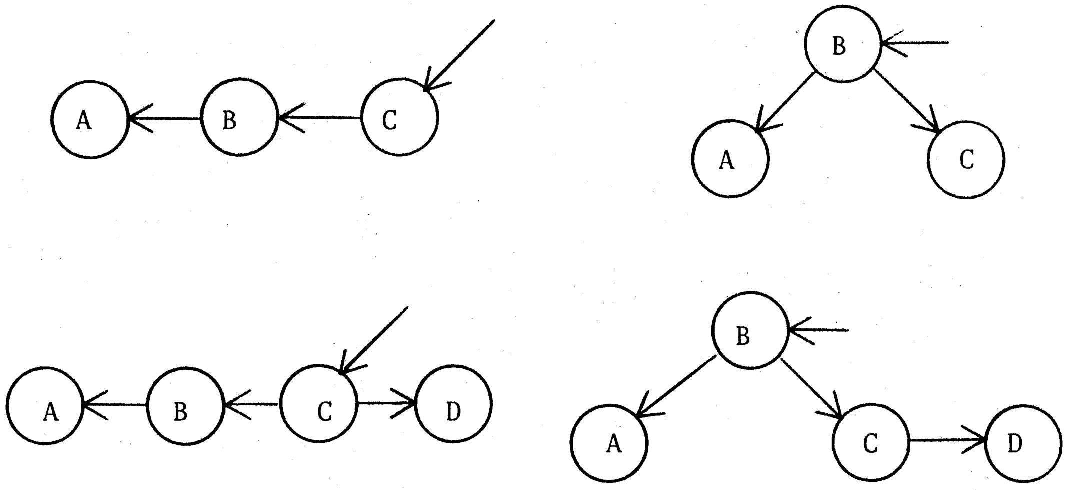


b.) The Tree of a.) After Key 50 Is Inserted

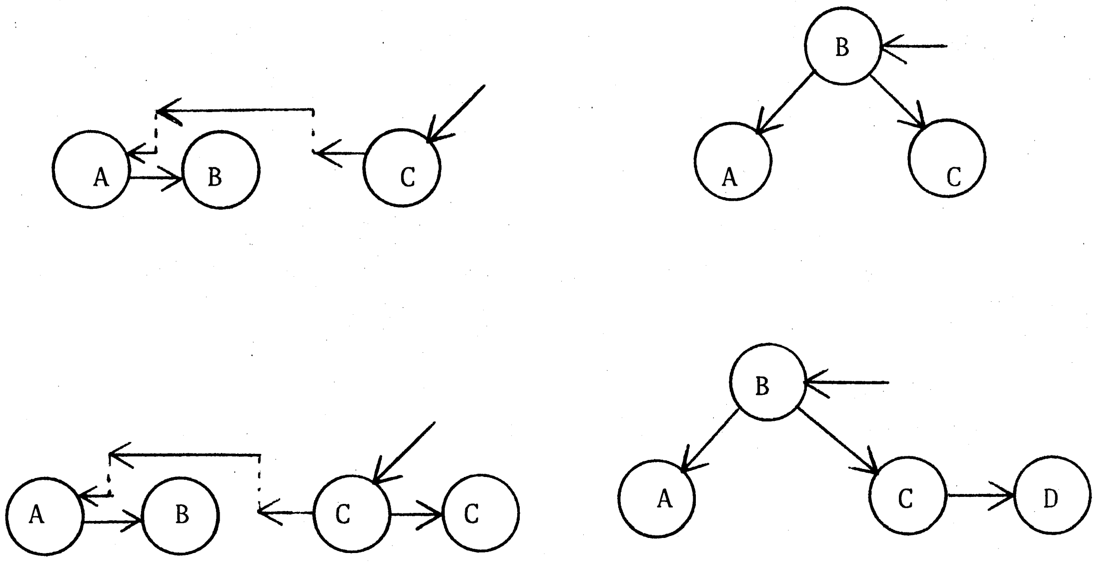


c.) The Tree of a.) After Key 35 Is Inserted

Figure 11. Insertion in an SBB-Tree



a.) Consecutive ρ-Pointers in Same Direction

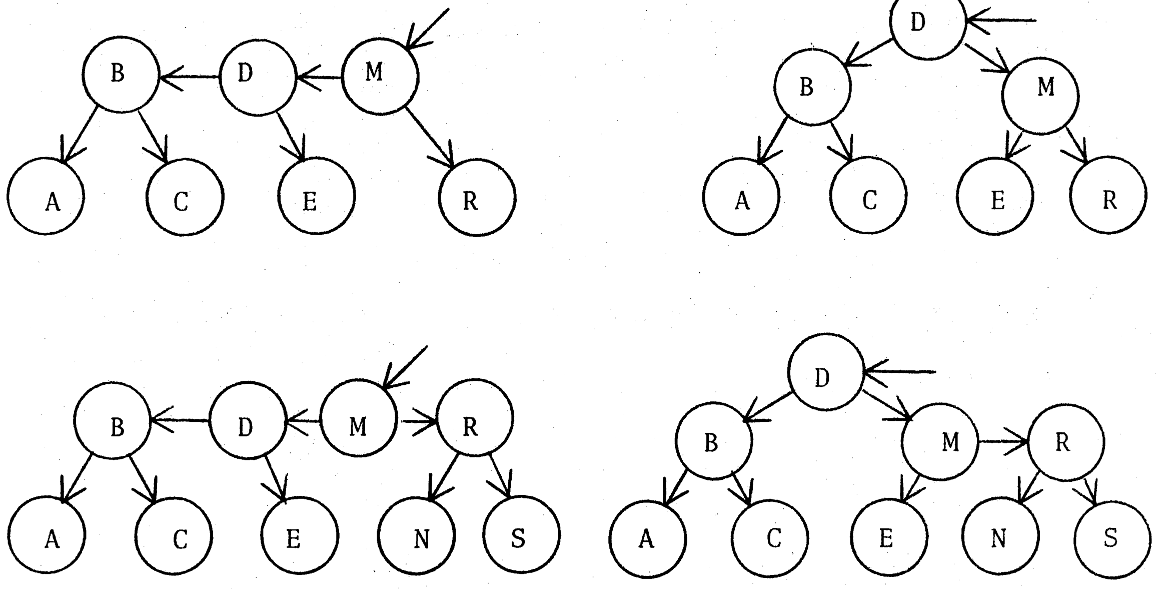


b.) Consecutive ρ-Pointers in Opposite Directions

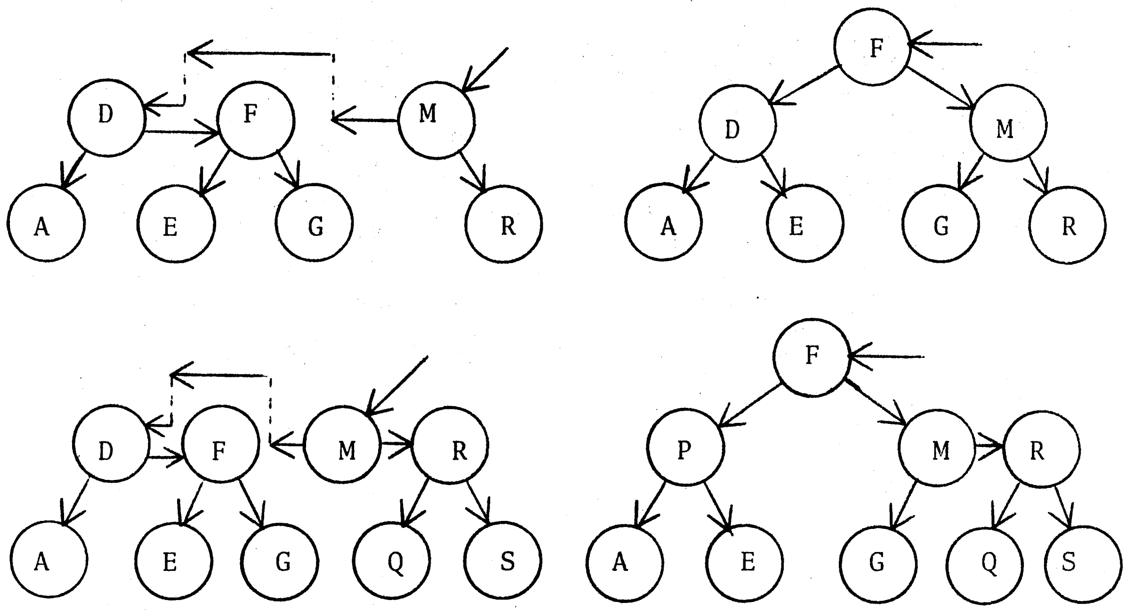
Figure 12. Lowest Level Splitting Operations for SBB-Tree

and it also shows the restructured configuration of the nodes after splitting. In both instances node B is elevated to the next higher δ-level; the ρ-pointer from B to A becomes a δ-pointer; the ρ-pointer from C to B is reversed to point from B to C and it becomes a δ-pointer; and the δ-pointer which previously pointed to C becomes a ρ-pointer pointing to B. This new ρ-pointer that is created at the next higher δ-level in the tree may cause a consecutive ρ-pointer in the same direction are the left-right symmetric versions of the two situations in Figure 12a. These would occur if the consecutive pointers were to the right of node C instead of to the left and corresponding splitting operations would be applied. For the situations involving consecutive ρ-pointers in opposite directions as shown in Figure 12b, node B is elevated to the next higher δ-level, and δ-pointers from B to A and B to C replace the ρ-pointers from C to A and A to B. Like cases in Figure 12a, the δ-pointer which originally pointed to C becomes a ρ-pointer to B at the next higher δ-level. Symmetric versions of the situations in Figure 12b are also possible and would be handled in a symmetric manner.

If promotion creates a consecutive ρ-pointer situation in the next higher δ-level when any of the above splitting operations are performed, the situation at that level is handled in much the same manner. This splitting operation may propagate several δ-levels up through an SBB-tree. Figure 13 shows examples of splitting at a higher level which correspond to the situations for the lowest level shown in Figure 12. Again, symmetric versions are possible in all instances. The only additional factor which must be taken into account at higher levels is the redistribution of the subtrees of the



a.) Consecutive ρ-Pointers in Same Direction



B.) Consecutive ρ-Pointers in Opposite Directions

Figure 13. Higher Level Splitting Operations for SBB-Trees

restructured nodes. For example, in the first case of Figure 13a the subtree of node D consisting of node E becomes a subtree of node M after splitting. The use of the insertion and splitting processes in combination makes possible the addition of nodes to an SBB-tree while maintaining the SBB-tree defining conditions. Since the tree of Figure 11c is no longer an SBB-tree following the insertion of key 35, the splitting process may be applied resulting in the SBB-tree of Figure 14.

The maintenance procedures described for SBB-trees may be extended in a straightforward manner to the 4-3 tree interpretation. If a new key may be inserted in an empty slot in a leaf node then no further action is necessary. However, if an empty key slot is not available to hold the new key a splitting operation analogous to the SBB-tree splitting is performed. The 4-3 trees of Figures 15a, 15b, and 15c correspond to the SBB-trees of Figures 11a, 11b, and 14 respectively and demonstrate the maintenance of the 4-3 tree interpretation for insertion cases involving one-key nodes and three-key nodes. The two situations involving insertion in nodes containing two keys are shown in Figure 16. It is important to note that insertion in the empty key slot of a two-key node does not call for a node split, but insertion on the opposite side does. This premature splitting of a 4-3 tree node before it is full must be taken into consideration in the formulation of the state transition equations.

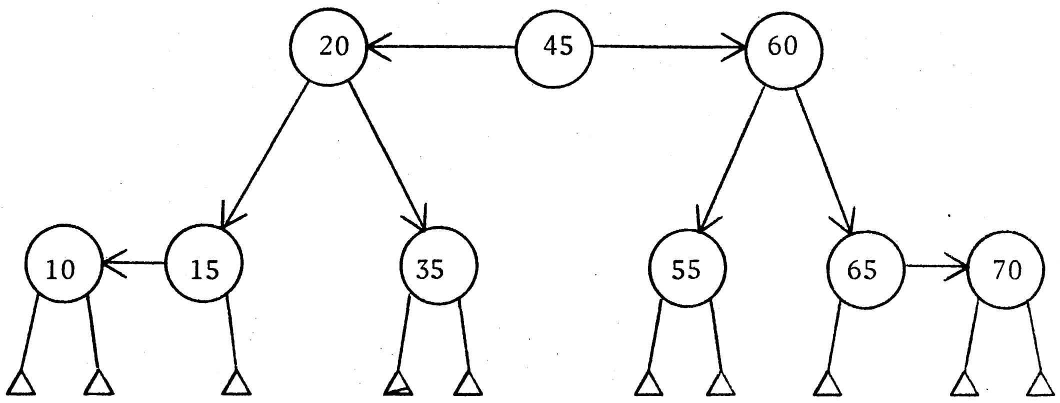
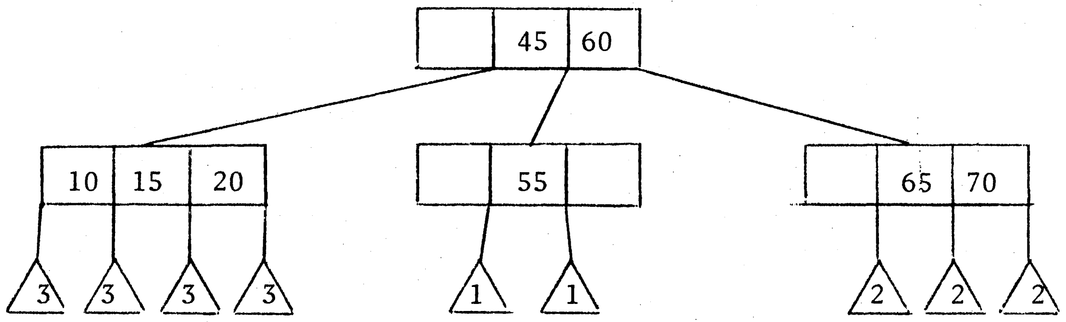
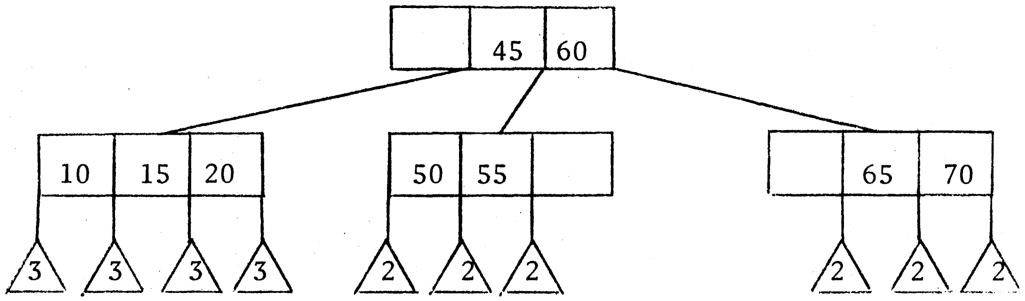


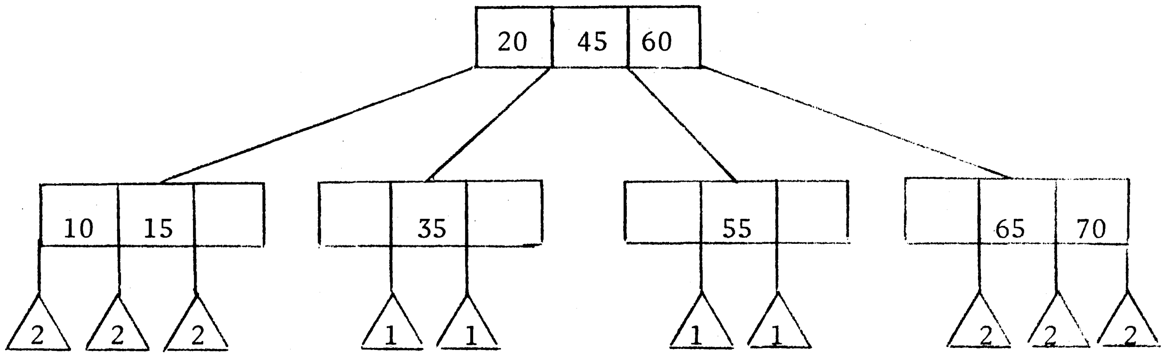
Figure 14. SBB-Tree of Figure 11c After Splitting Process



a.) 4-3 Tree Interpretation of Figure 11a

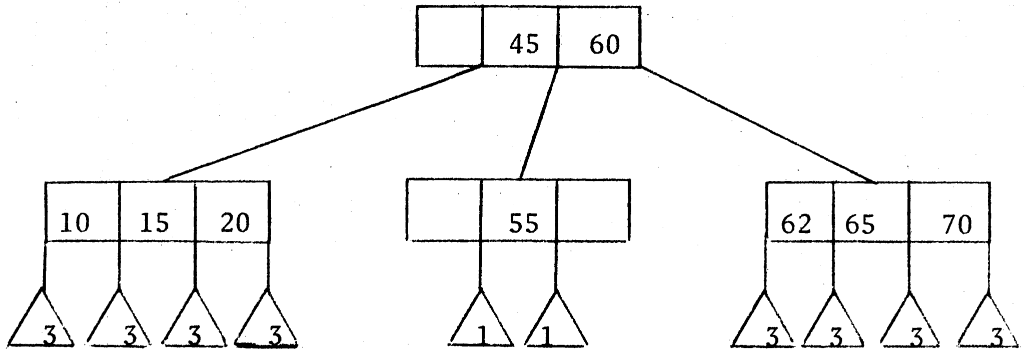


b.) Tree of a.) After Insertion of Key 50

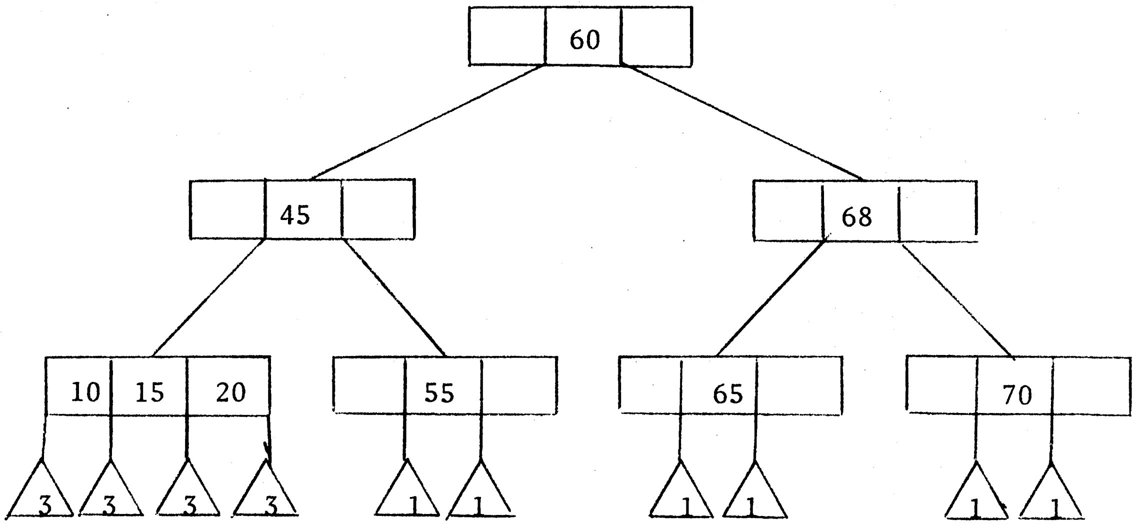


c.) Tree of a.) After Insertion of Key 35

Figure 15. Insertion in One Key and Three Key Nodes for a 4-3 Tree Interpretation of an SBB-Tree



a.) Tree of Figure 15a After Insertion of Key 62



b.) Tree of Figure 15a After Insertion of Key 68

Figure 16. Insertion in Two-Key Nodes for a 4-3 Tree Interpretation of an SBB-Tree

State Transition Equations for a 4-3 Tree Interpretation

Following the same procedure that was used to develop the state transition equations for 3-2 trees in the preceding chapter, the first step is to categorize the external nodes. Again basing the states on the number of keys in the father node, there will be three distinct states into which an external node may fall. External nodes with one, two, and three keys in their father nodes will be referred to as being in classes 1, 2, and 3 respectively. Examples of nodes in each of these classes can be seen in the trees of Figure 15.

Next, the relationships between these states before and after insertion are examined. It is easily seen that insertion in a one-key node as depicted in Figure 15b, always results in a two-key node so that two class 1 external nodes are destroyed and three class 2 external nodes are created. For insertion in a three-key node the result is always a split which causes the loss of four class 3 nodes, and the creation of two class 1 nodes and three class 2 nodes. This can be seen in Figure 15c. Note that there is no way for an insertion to transform class 1 nodes into class 3 nodes.

The logic for determining the transitions caused by insertion in a two-key father node are not quite so simple. Observing the two-key leaf node at the right in Figure 15a it can be seen that there are three external nodes for this leaf, each of which is assumed to be equally probable as the target of an insertion. This assumption is fundamental to the method of analysis. Based on observation of these external nodes as insertion positions, it follows that there is a one-third probability of a new key being inserted in the empty key slot to the left of key 65 and there is a two-thirds probability of insertion to the right of key 65 causing a node split. If no split occurs as in Figure 16a, three class 2 external nodes will be destroyed and four class 3 external nodes will be created. Figure 16b shows that a split will cause the loss of three class 2 external nodes and the creation of four class 1 external nodes.

Representing the fraction of external nodes in each state as P_{N+1}, and Q_{N+1}, and R_{N+1} respectively for a randomly generated 4-3 tree interpretation of an SBB-tree of N keys, the corresponding number of external nodes in each class will be $(N+1)\cdot P_{N+1}$, $(N+1)\cdot Q_{N+1}$, and $(N+1)\cdot R_{N+1}$. Relating the current state of the tree to the state immediately prior to the last key insertion when the number of external nodes was N, results in the following set of state transition equations:

$$(N+1)\cdot P_N = (N\cdot P_N)-(2\cdot P_N)+((8/3)\cdot Q_N)+(2\cdot R_N) \qquad (3.1)$$

$$(N+1)\cdot Q_N = (N\cdot Q_N)+(3\cdot P_N)-(3\cdot Q_N)+(3\cdot R_N) \qquad (3.2)$$

$$(N+1)\cdot R_N = (N\cdot R_N) \qquad + ((4/3)\cdot Q_N)-(4\cdot R_N) \qquad (3.3)$$

The general format of these equations indicates that the probable number of external nodes in a given class after insertion is equal to the probable number in that class before insertion plus and/or minus the probable number of external nodes in that class created and/or destroyed by insertion in a particular type of father node. Special note should be made of the coefficients of Q_N in the first and third equations where the two-thirds and one-third probability factors, obtained for the two possible occurrences dealing with insertion in a two key node are incorporated.

Rewriting the transition equations in matrix format yields:

$$(P_{N+1}, Q_{N+1}, R_{N+1}) = (P_N, Q_N, R_N) \cdot \begin{bmatrix} \frac{N-2}{N+1} & \frac{3}{N+1} & \frac{0}{N+1} \\ \frac{8/3}{N+1} & \frac{N-3}{N+1} & \frac{4/3}{N+1} \\ \frac{2}{N+1} & \frac{3}{N+1} & \frac{N-4}{N+1} \end{bmatrix} \qquad (3.4)$$

The elements of the three-by-three matrix are all nonnegative and its row sums are all one; therefore, it is a probability transition matrix and equation (3.4) defines a nonhomogeneous Markov chain.

Extending the concepts of state digraphs and their relationship to Markov chain models to the 4-3 tree interpretation of SBB-trees results in the digraph of Figure 17. Note that there is no path from state #1 to state #3 because a transition of that type is not possible. This missing path corresponds to the zero entry in the state transition matrix of equation (3.4). The other elements of the state transition matrix may be assigned to the digraph paths as follows:

$\frac{N-2}{N+1}$ = transition probability from state #1 to state #1

$\frac{3}{N+1}$ = transition probability from state #1 to state #2

$\frac{8/3}{N+1}$ = transition probability from state #2 to state #1

$\frac{N-3}{N+1}$ = transition probability from state #2 to state #2 (3.5)

$\frac{4/3}{N+1}$ = transition probability from state #2 to state #3

$\frac{2}{N+1}$ = transition probability from state #3 to state #1

$\frac{3}{N+1}$ = transition probability from state #3 to state #2

$\frac{N-4}{N+1}$ = transition probability from state #3 to state #3

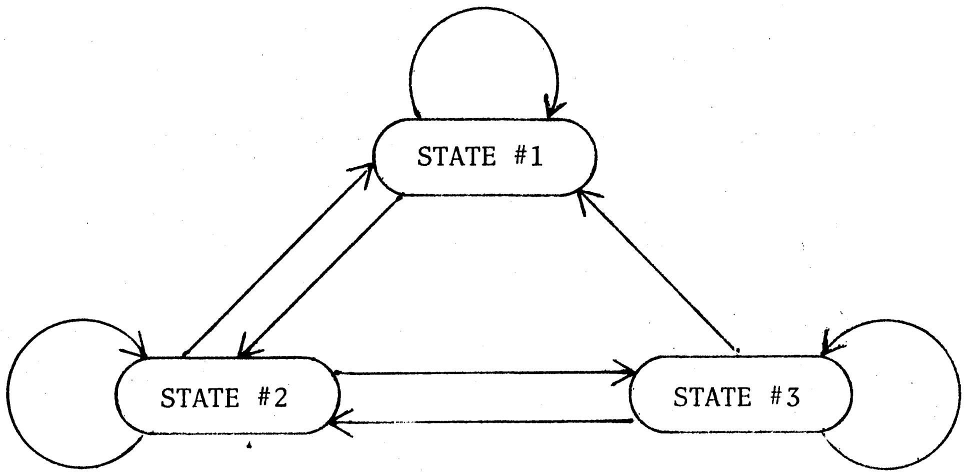


Figure 17. State Transition Digraph for 4-3 Tree Interpretation of SBB-Trees

Analysis of the 4-3 Tree Interpretation for SBB-Trees

Determination of the steady state values of the state probability vectors in equation (3.4) proceeds as for 3-2 trees. As stated in the Appendix the theorems and development of the method using Markov chain concepts on two-dimensional transition matrices with nonzero entries can be extended in a straightforward manner to transition matrices of any dimension and the restriction to nonzero entries can be relaxed if paths of equal length exist from every state to every other state in the state transition digraph. For the 4-3 tree interpretation of SBB-trees this may be accomplished by a path of length two as can be seen in Figure 17. The set of equations (3.1)-(3.3) and the relationship

$$P + Q + R = 1 \tag{3.6}$$

are used to solve for P_{N+1}, Q_{N+1}, and R_{N+1} at the limit where the vectors $(P_{N+1}, Q_{N+1}, R_{N+1})$ and (P_N, Q_N, R_N) are equal. Thus, ignoring subscripts and rewriting equation (3.2) yields:

$$(N \cdot Q)+Q = (N \cdot Q)+3 \cdot P-3 \cdot Q+3 \cdot R \tag{3.7}$$

subtracting $(N \cdot Q)$ from both sides and restructuring gives:

$$Q = 3 \cdot (P+R)-3 \cdot Q \tag{3.8}$$

or substituting (1-Q) for (P+R) from (3.6):

$$Q = 3 - (6 \cdot Q) \tag{3.9}$$

therefore Q=3/7. Note that the N terms can be removed from any of the equations (3.1)-(3.3) in the manner that they were removed from (3.7) to obtain (3.8) above. This indicates that the equations are independent of N. Removing the N terms and substituting 3/7 for Q in equation (3.3) results in 4/35 for the value of R. Finally,

$$P = 1 - Q - R = 1 - 3/7 - 4/35 = 16/35 \tag{3.10}$$

Thus, the long term values of the probability state vector are:

$$P_{N+1} = 16/35,\ Q_{N+1} = 3/7,\ R_{N+1} = 4/35 \tag{3.11}$$

Through the use of these values the number of keys in each type of father node, and thus the number of each type of father node in the bottom internal level of the tree can be determined. The number of keys in one key father nodes is:

$$P \cdot (N+1) \cdot 1/2 = 16/35 \cdot (N+1) \cdot 1/2 = 8/35 \cdot (N+1) \tag{3.12}$$

where the factor 1/2 is the ratio of internal keys per external nodes for a father node of this type. Similarly for two-key and three-key father nodes the number of keys is:

$$Q\cdot(N+1)\cdot 2/3 = 3/7\cdot(N+1)\cdot 2/3 = 2/7\cdot(N+1) \qquad (3.13)$$

$$R\cdot(N+1)\cdot 3/4 = 4/35\cdot(N+1)\cdot 3/4 = 3/35\cdot(N+1) \qquad (3.14)$$

It follows directly that the number of nodes of each type is:

$(8/35) \cdot (N + 1)$ nodes containing one key (3.15)

$(1/7) \cdot (N + 1)$ nodes containing two keys (3.16)

$(1/35) \cdot (N + 1)$ nodes containing three keys (3.17)

The total number of keys in the lowest internal level is obtained by summing the values in (3.12), (3.13), and (3.14):

$$(8/35+2/7+3/35)\cdot(N+1) = 21/35\cdot(N+1) = 3/5\cdot(N+1) \qquad (3.18)$$

Likewise, the number of nodes in the bottom internal level is the sum of (3.15), (3.16), and (3.17):

$$(8/35+1/7+1/35)\cdot(N+1) = 14/35\cdot(N+1) = 2/5\cdot(N+1) \qquad (3.19)$$

The number of keys in the upper levels of the tree is one less than the number of nodes in the lowest level, or:

$$2/5\cdot(N+1)-1 = 2/5\cdot N=3/5 \qquad (3.20)$$

The utilization of the bottom internal level is:

$$3/5\cdot(N+1)\text{keys}/3\cdot 2/3\cdot(N+1)\text{key slots} = .5 \qquad (3.21)$$

This utilization, when thought of in terms of individual nodes, means that the average number of keys per node at the bottom internal level is 1.5. The average degree of branching from these nodes is one greater than the average number of keys or 2.5. This can be confirmed by dividing the total number of branches out of the bottom internal level by the expected number of nodes in that level:

$$(N + 1) / ((2/5) \cdot (N + 1)) = 5/2 = 2.5 \qquad (3.22)$$

It must be remembered that the above analysis applies rigorously only to the bottom internal level of a tree. The bottom level of the 4-3 tree analyzed above is actually an interpretation of the bottom δ-level of an SBB-tree.

Analysis of Upper Tree Levels

An attempt to thoroughly analyze more than the lowest level of a 4-3 tree using Markov chain analysis quickly grows out of manageable proportions due to the excessive number of distinct states in which an external node may be classified. For analysis in terms of only one additional level above the lowest there would be 117 different combinations of grandfather and father nodes and the addition of higher levels increases this number combinatorially.

In order to simplify the analysis, the assumption of the probabilistic independence of tree levels that was made in chapter two for 3-2 trees will be applied here to SBB-trees interpreted as 4-3 trees. As before, it must be pointed out that this assumption is not valid in a rigorous mathematical sense but approximates the true situation. In effect it yields an approximate model of behavior.

Making use of this simplifying assumption allows the extension of the results obtained for the behavior of the bottom internal δ-level of an SBB-tree to the SBB-tree as a whole. It is then possible to estimate the expected number of δ-levels for an SBB-tree of N keys. If the average number of keys per tree node is 1.5, the average branching from each node is 2.5 and the expected number of keys (N) for a tree of m levels can be expressed approximately as the geometric progression:

$$N = (1.5)\cdot(2.5)^0+(1.5)\cdot(2.5)^1+\ldots+(1.5)\cdot(2.5)^{m-1} \quad (3.23)$$

Multiplying the above through by 2.5 gives:

$$2.5\cdot N = (1.5)\cdot(2.5)^1+(1.5)\cdot(2.5)^2+\ldots+(1.5)\cdot(2.5)^m \quad (3.24)$$

subtracting (23) from (24) yields:

$$(2.5)\cdot N-N = (1.5)\cdot(2.5)^m-(1.5) \quad (3.25)$$

or:

$$N = (2.5)^m - 1 \quad (3.26)$$

Thus for an SBB-tree of N keys the approximate expected number of δ-levels is given by:

$$m = \log_{2.5} (N + 1) \quad (3.27)$$

This result, however, ignores the ρ-levels within a given δ-level which add additional key comparisons in the search paths of an SBB-tree. If the expected number of key comparisons necessary to traverse a given δ-level is calculated, this value will be equal to the average number of binary tree levels present in an SBB-tree δ-level. The traversal of a one-key node requires one comparison and occurs with a probability of 16/35 according to the value of P_{N+1} in (3.11). The traversal of a three-key node requires two comparisons and occurs with a probability of 4/35. Traversal of a three key node occurs with a probability of 3/7; however, one-third of these traversals will require only one node comparison while two-thirds will require two comparisons. This is due to the fact that of the three equally likely δ-pointers out of the δ-level, two are subsequent to the ρ-pointer within the δ-level. Thus the average number of node comparisons for each δ-level is the sum of the average number of one node comparisons,

$$16/35 + 5/35 = 21/35 = 3/5 \quad (3.28)$$

and twice the number of two node comparisons,

$$2 \cdot (4/35 + 10/35) = 28/35 = 4/5 \tag{3.29}$$

The result is 7/5 or 1.4 binary tree levels per SBB-tree δ-level, or $1.4 \cdot \log_{2.5}(N+1)$ binary tree levels for an SBB-tree of N keys. Knuth (7) has shown that the expected path length to an external node for completely balanced binary trees is approximately $\log_2 N$. Since completely balanced trees are a subset of SBB-trees in the binary tree sense it would be expected that for an identical number of keys the expected path length to an external node in an SBB-tree would be somewhat larger. Forming the ratio of the two values and simplifying yields:

$$\frac{1.4 \cdot \log_{2.5} N}{\log_2 N} = (1.4) \frac{\log_{10} 2}{\log_{10} 2.5} \approx 1.06 \tag{3.30}$$

The analysis thus seems to give an intuitively plausible approximation to the expected performance of a randomly built SBB-tree but it needs to be subjected to empirical testing.

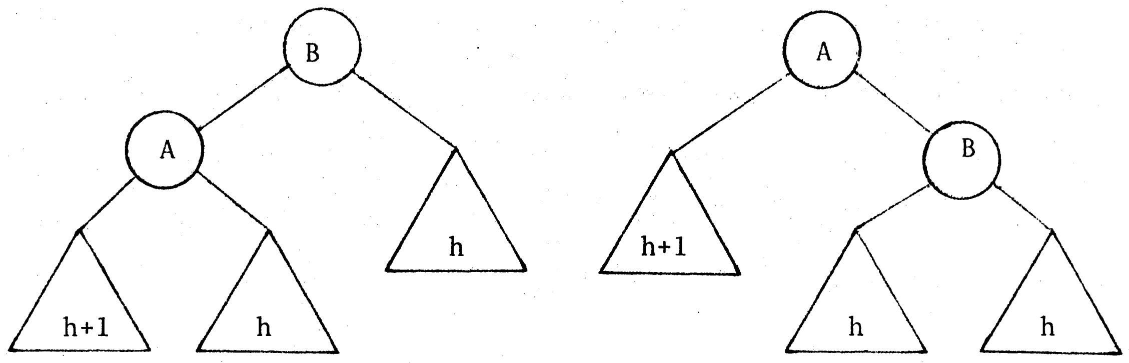
The fact that the similar analysis of 3-2 trees in Chapter II is supported by empirical evidence lends credence to the results given here, however.

CHAPTER IV

ANALYSIS OF AVL TREES

As stated in chapter one, AVL trees are a class of balanced binary trees for which the maximum number of levels in the left and right subtrees of any node differ by no more than one. If this difference becomes greater than one at any node when a new key is inserted in the tree, then a restructuring process must be applied to nodes such that the proper balance is regained. Basically there are two operations used to accomplish this rebalancing, rotation and double rotation.

Rotation is used as shown in Figure 18, where triangular nodes represent arbitrary subtrees of height h or h+1 as labeled. In Figure 18a, the left subtree of node B, which has node A as its root node, has a height of h+2 while the right subtree has a height of h. In order to rebalance the tree, node A, the root of the left subtree, becomes the root of the tree, while node B, the previous root of the tree, is rotated to the position of root node for the right subtree. The right subtree of node A before rotation becomes the left subtree of node B after rotation. The result is a balanced tree with root node A. This operation will work unless a tree becomes unbalanced in the manner shown in Figure 19a. If a simple rotation were used in this case the resulting tree would also be unbalanced because instead of shortening the subtree of maximum height, it would merely shift it



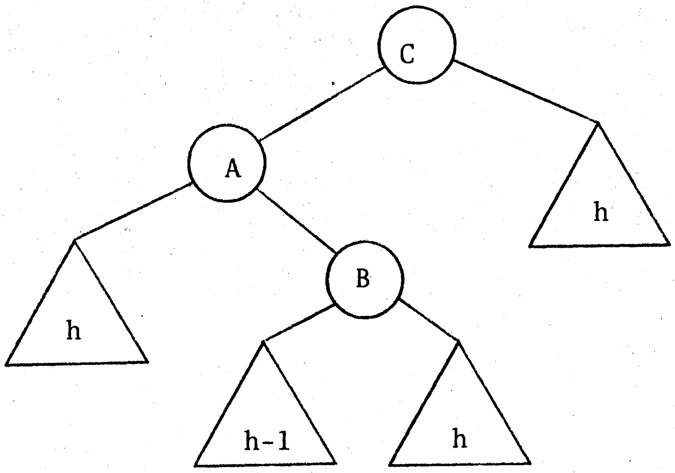
a.) Unbalanced AVL Tree b.) Rebalanced AVL Tree

Figure 18. AVL Tree Rotation Process

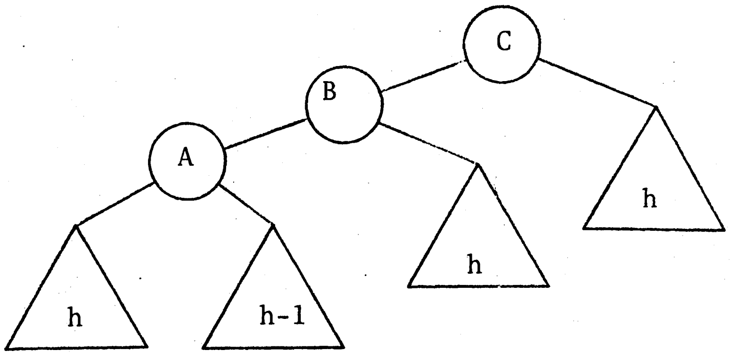
to the other side of the tree. To avoid this problem the double rotation process shown in Figure 19b and 19c is used. A simple rotation is applied first to the largest subtree of the unbalanced tree and then to the tree itself. The result (Figure 19c) is a balanced AVL tree with root node B. The rotate and double rotate operations also apply to the reflections of the trees of Figures 18 and 19 and may be used at any level of an AVL tree where an imbalance occurs.

Leaf and Semi-leaf Node Analysis

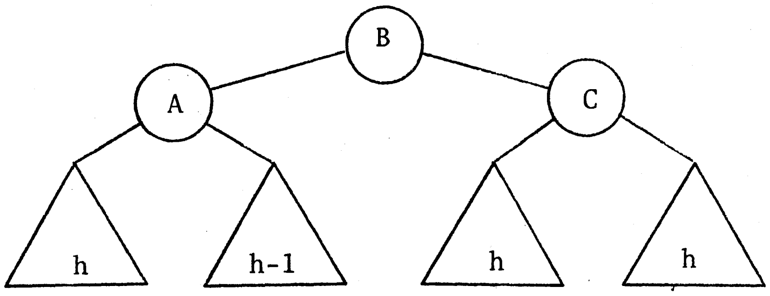
The concept of external node classification may be applied to AVL trees. A leaf and semi-leaf analysis suggests three classes of external nodes as pictured in Figure 20. Those nodes labeled 1 have leaf node fathers and semi-leaf node grandfathers. Those nodes labeled 2 have semi-leaf node fathers. The remainder of the nodes are labeled 3 and have leaf node fathers but their grandfathers are not



a.) Unbalanced AVL Tree



b.) AVL Tree After First Rotation Step



c.) AVL Tree After Second Rotation Step

Figure 19. AVL Tree Double Rotation Process

semi-leaf nodes.

Close observation of the relationships between these classes yields the fact that every semi-leaf node has one son that is an external node of class 2 and two grandsons that are external nodes of class 1. Therefore, there are always twice as many class 1 external nodes as there are class 2 external nodes. Using the method of analysis as extended in chapter three, P_{N+1}, Q_{N+1}, and R_{N+1} will represent the fraction of external nodes in classes 1, 2, and 3 respectively for an arbitrary AVL tree containing N keys. As before, for a tree of N keys, there will be N+1 external nodes. The possible changes that can occur in external node states when a key is inserted in an AVL tree are displayed in Figure 21. The trees shown result from inserting a new key in an external node from each of the tree classes in the tree of Figure 20. Figure 21a indicates that insertion in a class 1 external node causes the loss of two class 1 nodes, the loss of one class 2 node, the addition of four class 3 nodes, and the addition of one to the total number of external nodes. This particular insertion requires a double rotation, but the results would be the same for a simple rotation. For insertion in a class 2 external node, Figure 21b shows that two class 1 nodes are lost, one class 2 node is lost, four class 3 nodes are gained, and one external node is added to the total. This demonstrates that insertion in a class 2 external node has the same effect on an AVL tree with respect to its set of external nodes as insertion in a class 1 node. Finally, insertion in a class 3 external node has the effect of destroying two class 3 nodes, creating two class 1 nodes, creating one class 2 node, and creating one additional external node in the system. This is depicted in Figure 21c. Rotation

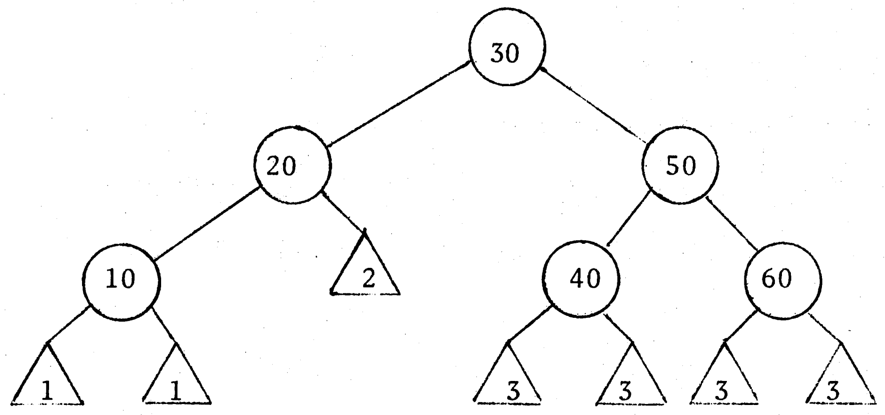


Figure 20. Example AVL Tree with Classified External Nodes

and double rotation occurring at levels higher than the lowest possible rotation level will cause arrangement of subtrees, but as shown in Figure 22, these will not affect the number of external nodes in any given class. If these observations are used to formulate the state transition equations for an AVL tree containing N-1 keys before insertion and N keys after insertion the resulting set of equations is:

$$(N+1)\cdot P_{N+1} = (N\cdot P_N)-(2\cdot P_N)-(2\cdot Q_N)+(2\cdot R_N) \tag{4.1}$$

$$(N+1)\cdot Q_{N+1} = (N\cdot Q_N)-(1\cdot P_N)-(1\cdot Q_N)+(1\cdot R_N) \tag{4.2}$$

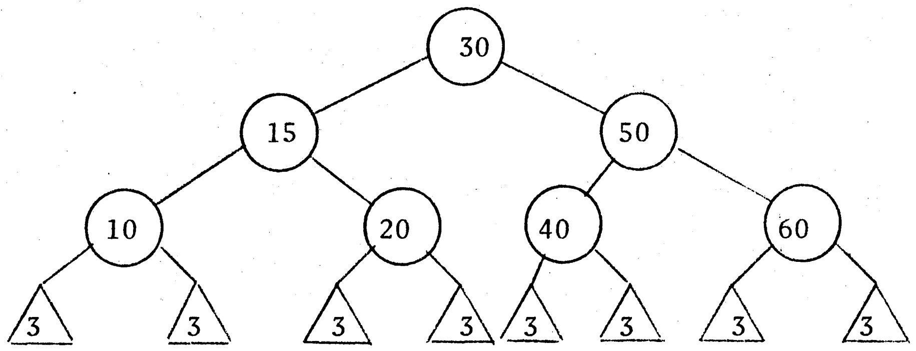
$$(N+1)\cdot R_{N+1} = (N\cdot R_N)+(4\cdot P_N)+(4\cdot Q_N)-(2\cdot R_N) \tag{4.3}$$

In order to simplify the analysis, this set of three equations can be reduced to two equations by utilizing the relationship between class 1 and class two external nodes:

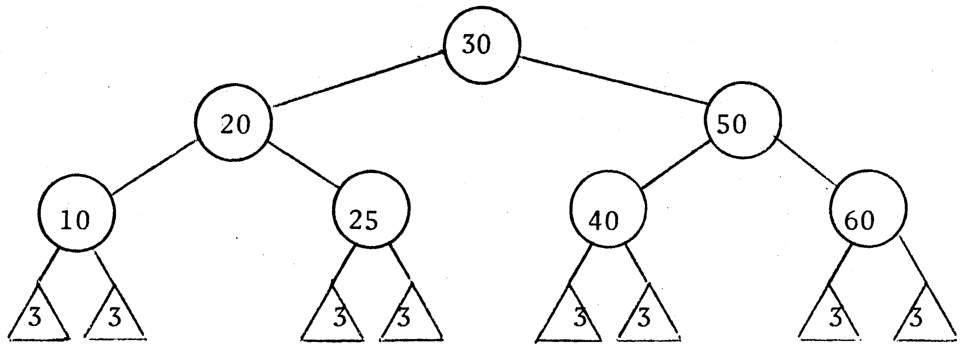
$$P = 2Q \tag{4.4}$$

The simplification can be accomplished by defining

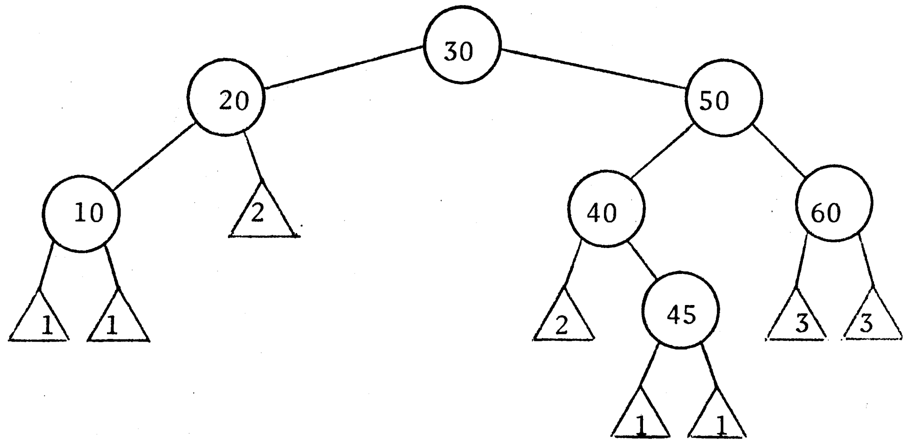
$$S = P + Q = 3\cdot Q \tag{4.5}$$



a.) After Inserting Key 15 in a Class 1 External Node



b.) After Inserting Key 25 in a Class 2 External Node



c.) After Inserting Key 45 in a Class 3 External Node

Figure 21. AVL Trees Resulting from Insertion in Each of the External Node Classes of Figure 20

and substituting for P plus Q in the sum of equations (4.1) and (4.2) and in equation (4.3). The resulting equations are:

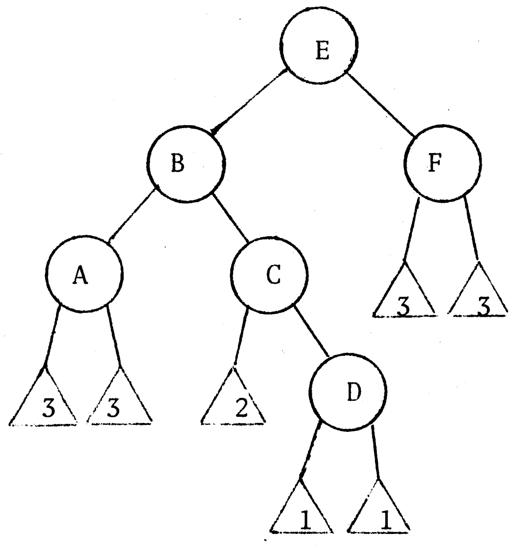
$$(N+1)\cdot S_{N+1} = (N\cdot S_N)-(3\cdot S_N)+(3\cdot R_N) \qquad (4.6)$$

$$(N+1)\cdot R_{N+1} = (N\cdot R_N)+(4\cdot S_N)-(2\cdot R_N) \qquad (4.7)$$

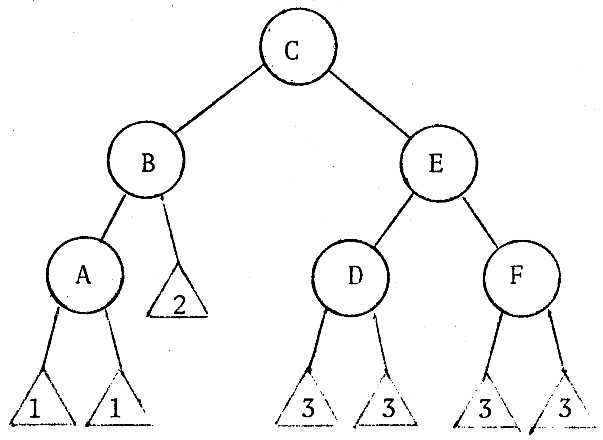
Rearranging (4.6) and (4.7) in matrix format yields:

$$(S_{N+1}, R_{N+1}) = (S_N, R_N)\cdot \begin{bmatrix} \frac{N-3}{N+1} & \frac{4}{N+1} \\ \\ \frac{3}{N+1} & \frac{N-2}{N+1} \end{bmatrix} \qquad (4.8)$$

An interesting observation can be made at this point by comparing the transition matrix in equation (4.8) above with the transition matrix derived for 3-2 trees in Chapter II. The two matrices are identical.



a.) Before Restructuring



b.) After Restructuring

Figure 22. Comparison of the Number of External Nodes In the Various Classes Following Restructuring at a Level Higher than the Lowest Possible Rotation Level

This means that the stable probability vector values for (4.8) are the same as those for 3-2 trees, (3/7, 4/7). Using equations (4.4) and (4.5) to transform the value obtained for S, 3/7, back into values for P and Q, gives the long term probabilities for all of the original three classes of external nodes as:

$$P_{N+1} = 2/7 \quad , \quad Q_{N+1} = 1/7 \quad , \quad R_{N+1} = 4/7 \qquad (4.9)$$

It may further be observed that the probability that an external node descends from a leaf node is equal to the sum of P_{N+1} and R_{N+1}:

$$P_{N+1} + R_{N+1} = 6/7 \qquad (4.10)$$

It follows that the probable number of external nodes descending from leaf nodes for an AVL tree of N keys asymptotically is:

$$(N+1)\cdot 6/7 \qquad (4.11)$$

and since there are two such descendants from each leaf node the probable number of leaf nodes would be:

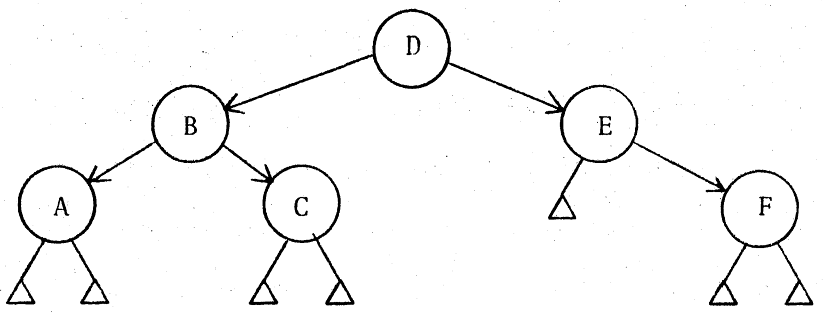
$$(N+1)\cdot 3/7 \qquad (4.12)$$

For support of the validity of the above analysis reference is made to an exhaustive combinational analysis of ten key balanced trees by Knuth (7, p. 462). Knuth's analysis gives the exact probability of one-seventh that when the tenth item is inserted in the tree, no imbalance will occur at its father node. This corresponds to insertion at a class 2 external node position for which the probability value obtained above is also one-seventh. Insertion in a class 1 external node position, for which the probability shown in equations (4.9) is two-sevenths, corresponds to the situations where insertion causes an imbalance at the grandfather node. Knuth also obtained two-sevenths as the exact probability for a ten key balanced tree.

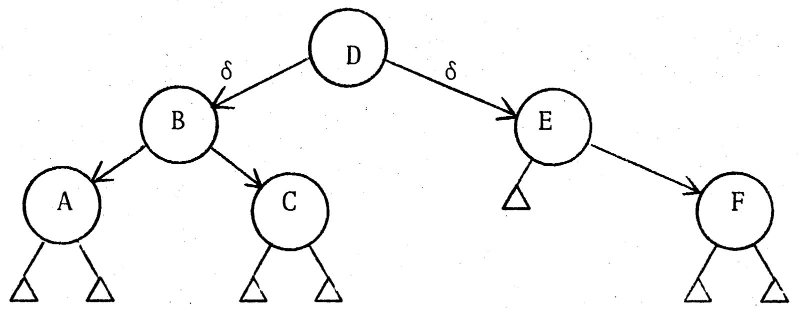
Balanced Trees as a Proper Subset of SBB-Trees

One of the properties of the class of SBB-trees as they were defined in the preceding chapter is that it contains the class of balanced trees (AVL trees) as a proper subset. This means that all AVL trees can be represented as SBB-trees, but not all SBB-trees are balanced trees. Bayer (2) proves these facts by presenting an algorithm which will translate all possible AVL trees into SBB-tree format and by giving an example of an SBB-tree that is not balanced to show that the subset is a proper subset. A description of the algorithm for translating an AVL tree to SBB-tree format and some examples of various AVL trees and their SBB-tree counterparts follows.

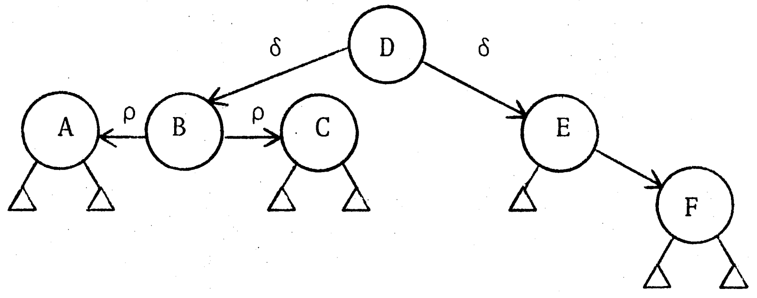
To convert a given balanced tree to an SBB-tree, the height of the tree must first be determined. If the height of the tree is odd then the pointers from the root node to the left and right subtrees are both converted to δ-pointers (downward pointers). If the height of the tree is even then the heights of the left and right subtrees must be determined. Once they are determined, they are compared; if they are equal then the pointers from the root node to the left and right subtrees are both converted to ρ-pointers (horizontal pointers). If the subtree heights are not equal the pointer from the root node to the subtree of greater height is converted to a ρ-pointer. The pointer to the lesser subtree becomes a δ-pointer. The algorithm is then applied recursively to the subtrees until all pointers in the balanced tree have been set. To demonstrate this, a step-by-step conversion of the balanced tree of Figure 23a is shown in subsequent trees of Figures 23 (b-d). The tree of Figure 23a with root node D has a height of three which is odd so both pointers to the subtrees of node D are converted to δ-pointers as in Figure 23b. The subtree with root node B is of even height (two) and both of the subtrees of node B are of equal height (one); therefore, the pointers to the subtrees of node B are both ρ-pointers as in Figure 23c. The right subtree of node D which has root node E is also of height two but its left subtree zero height while its right subtree has a height of one. The pointer to the larger right subtree thus becomes a ρ-pointer. The pointer to the left subtree, which is an external node in this case, is a δ-pointer. These steps are shown in Figure 23d. Leaf nodes can always be considered as trees of height one so that pointers to external nodes are always δ-pointers. The proof that this algorithm works for all balanced trees is given by Bayer (2, p. 295). By observing the SBB-tree of Figure 24 it is easily seen that not all SBB-tree are balanced trees. The path from root node B to leaf node G contains four nodes while the path from B to A contains only two.



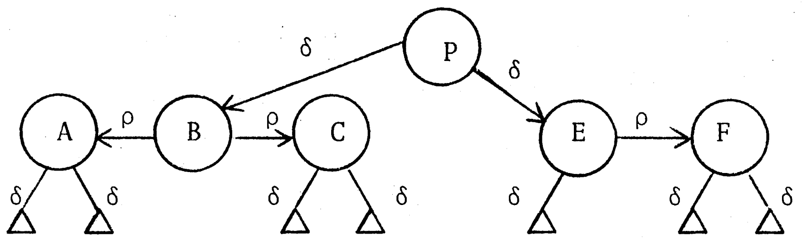
a.) Example Balanced Tree



b.) Pointers from Node D Converted to SBB-Tree Format



c.) Pointers from Node B Converted to SBB-Tree Format



c.) Pointers from Node B Converted to SBB-Tree Format

Figure 23. Conversion of AVL Pointers to SBB-Tree Format

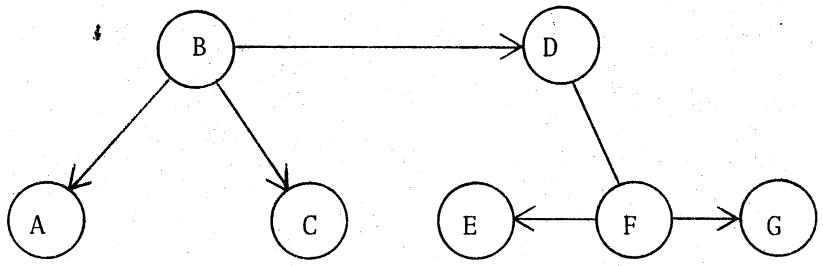


Figure 24. An SBB-Tree That is not a Balanced Tree

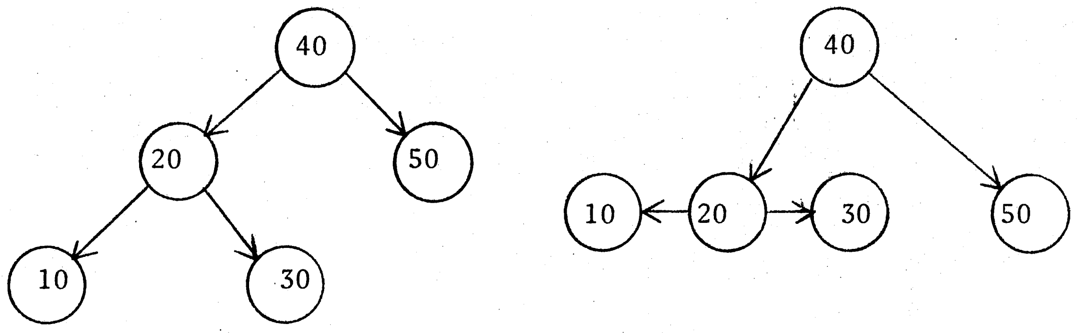
SBB-Tree Interpretation of AVL Maintenance Operations

As shown in the previous section, any balanced tree can be represented as an SBB-tree. However, the analysis of SBB-trees presented in the preceding chapter cannot be applied directly to balanced trees because the maintenance algorithms on which the state transition equations are predicated allow the SBB-trees to become unbalanced when insertions are made. An example of this incongruity is outlined in the trees in Figure 25. Insertion of key 5 in the balanced tree of Figure 25a results in the balanced tree of Figure 25b. The SBB-trees derived from both of these balanced trees by application of the algorithm of the last section are shown to the right of each tree. For comparison the SBB-tree obtained by inserting key 5 in the SBB-tree of Figure 25a using the maintenance algorithms described in chapter three is shown in Figure 25c. Not only does this tree fail to correspond to the SBB-tree of Figure 25b, but it fails to remain a balanced tree. The correlation between insertion in an AVL tree and the resulting effects on SBB-tree models of these balanced trees must be investigated before an SBB-tree state transition model can be derived that is directly applicable to AVL analysis.

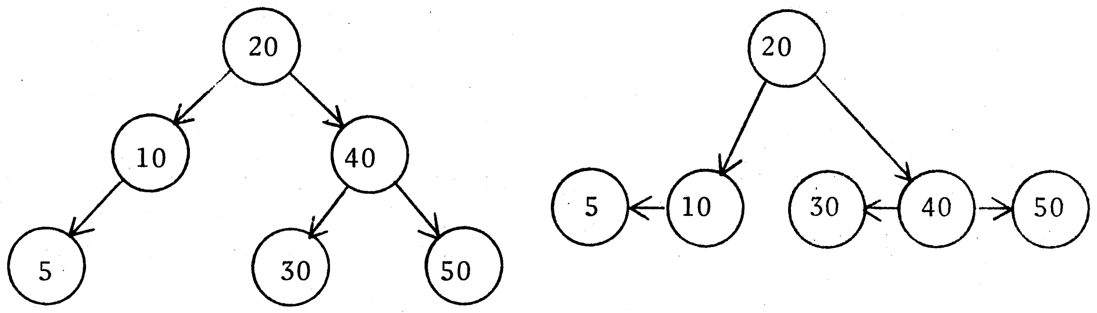
There are four basic situations in which insertion occurs in an AVL tree. In order of inspection they are: 1) insertion below a leaf node that has a semi-leaf father, 2) insertion below a semi-leaf, 3) insertion below a leaf that has a brother node which is not a leaf, and 4) insertion below a leaf that has a brother node which is a leaf.

Inspecting first the case of insertion below a leaf node that has a semi-leaf father as pictured in Figure 26a, it is evident that this type insertion will always cause an imbalance at the grandfather node of the newly inserted node, node 50 in the example. A rotation or double rotation at this node is all that is required to rebalance the tree as shown in Figure 26a-b. Inspection of the resulting SBB-trees which are represented as 4-3 trees in Figure 26 shows that the corresponding maintenance operation for these trees is merely the creation of a three key node from a two key node by internally shifting the old keys and inserting the new one. Figure 27 demonstrates the second possible insertion situation. This situation will never cause an AVL tree imbalance and its corresponding SBB-tree operation is insertion in the empty key slot of a two key node. The similarity between the situations of Figures 26 and 27 was evidenced in the section of this chapter dealing with leaf and semi-leaf analysis when the transition equations for these two types of insertion were combined to simplify the analysis.

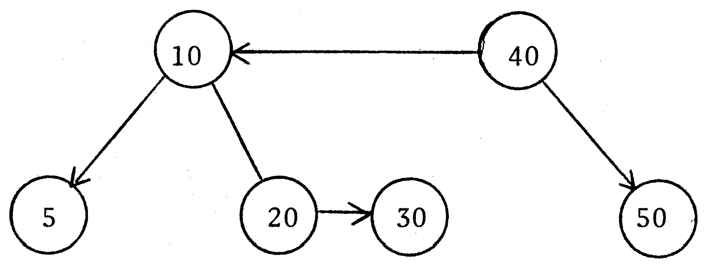
The third insertion situation--insertion below a leaf node which has a brother that is not a leaf node--will never cause an imbalance because the subtree containing the non-leaf brother has a greater



a.) Example AVL Tree and Corresponding SBB-Tree

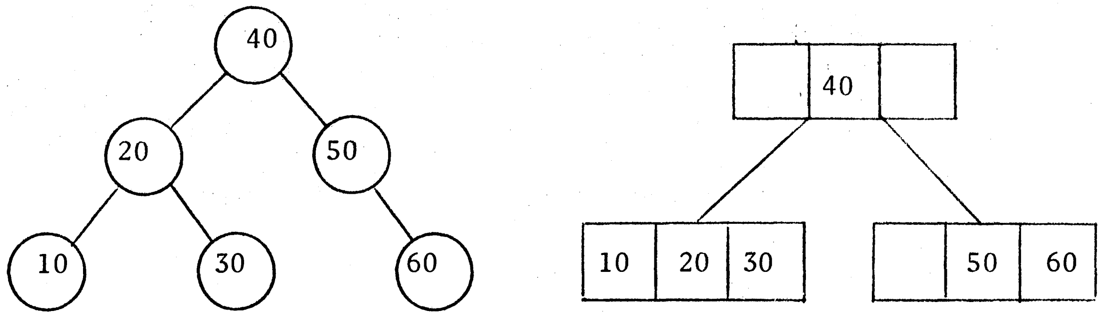


b.) AVL Tree and Corresponding SBB-Tree Obtained by Inserting Key 5 in the AVL Tree of a.)

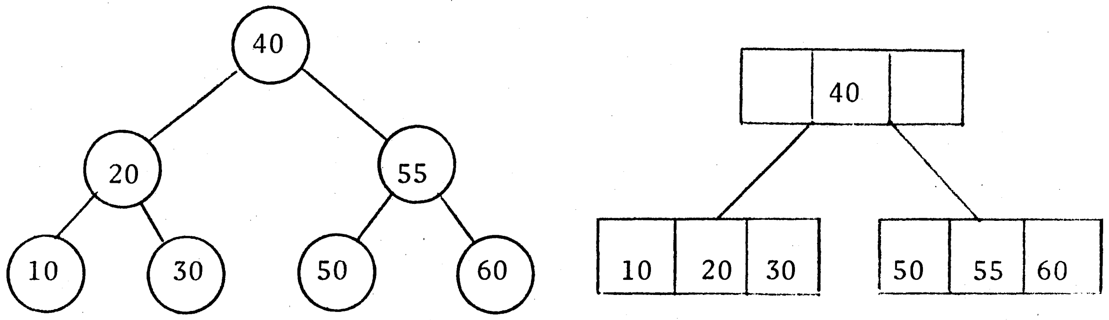


c.) SBB-Tree Obtained by Inserting Key 5 in the SBB-Tree of a.)

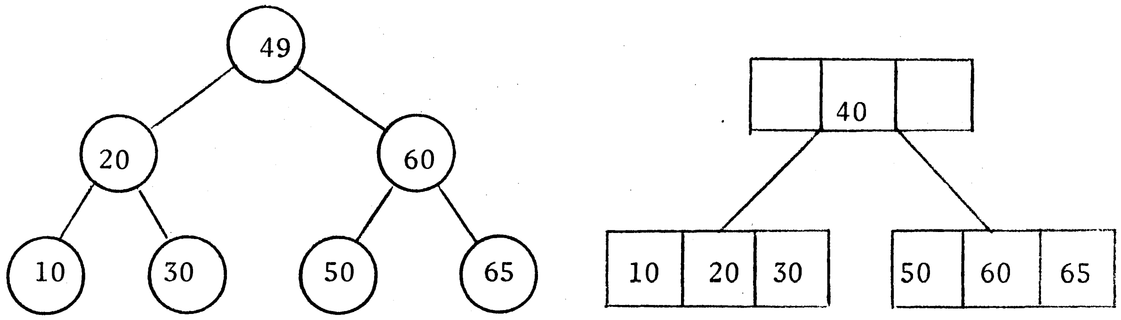
Figure 25. Comparison of Key Insertion Results for AVL Trees and SBB-Trees



a.) Example AVL Tree and Corresponding SBB-Tree



b.) AVL tree of a.) and Corresponding SBB-Tree After
Inserting Key 55 and Performing a Double Rotation



c.) AVL Tree of a.) and Corresponding SBB-Tree After
Inserting Key 65 and Performing a Single Rotation

Figure 26. Insertion Below a Leaf Node that has a Semi-Leaf Father

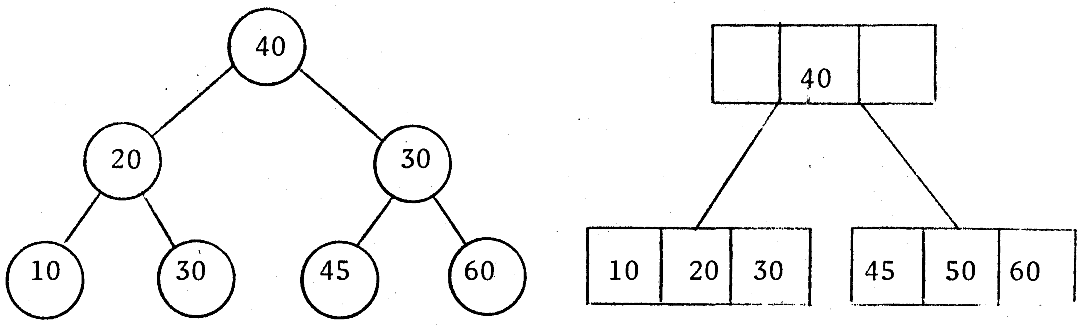
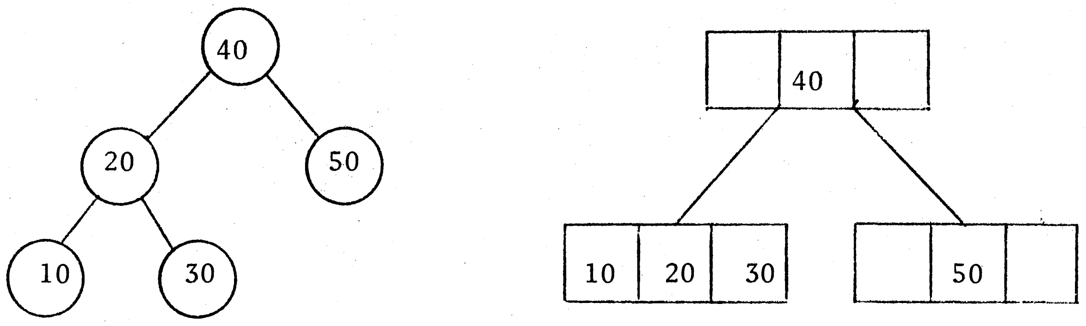
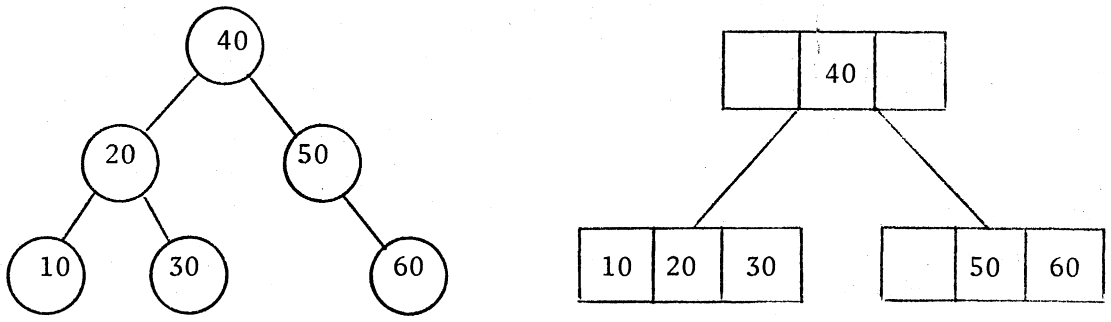


Figure 27. AVL Tree of 26a and Corresponding SBB-Tree After Inserting Key 45 Showing Insertion Below a Semi-Leaf Node



a.) Example AVL Tree and Corresponding SBB-Tree

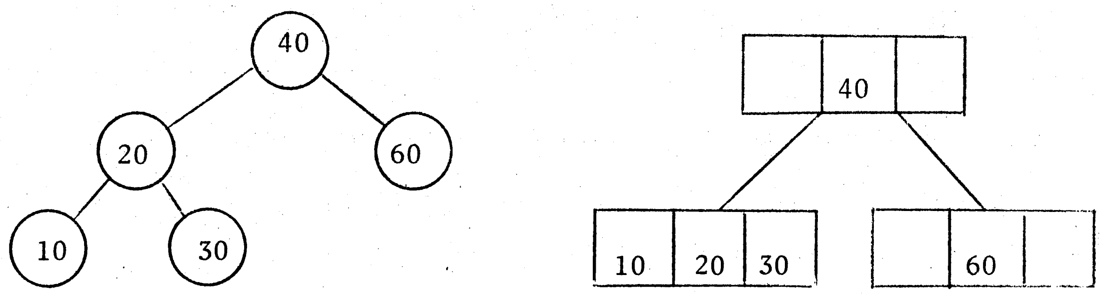


b.) AVL Tree of a.) and Corresponding SBB-Tree After Inserting Key 60

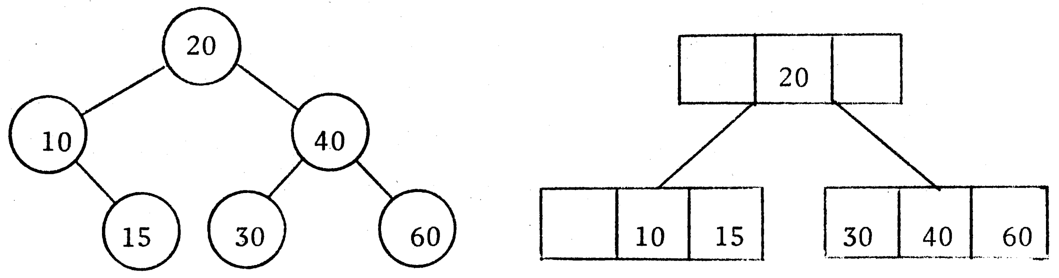
Figure 28. Insertion Below a Leaf Node that has a Brother Which is not a Leaf Node

height than the subtree in which the insertion is made. Insertion of key 60 below the node containing key 50 as depicted in Figure 28 demonstrates this situation. The corresponding SBB-tree operation will always be inserted in one of the empty key slots of a one key node. This third situation as well as the first and second situations have required only simple internode maintenance in the corresponding SBB-tree interpretations. This, however, is not the case for the fourth situation.

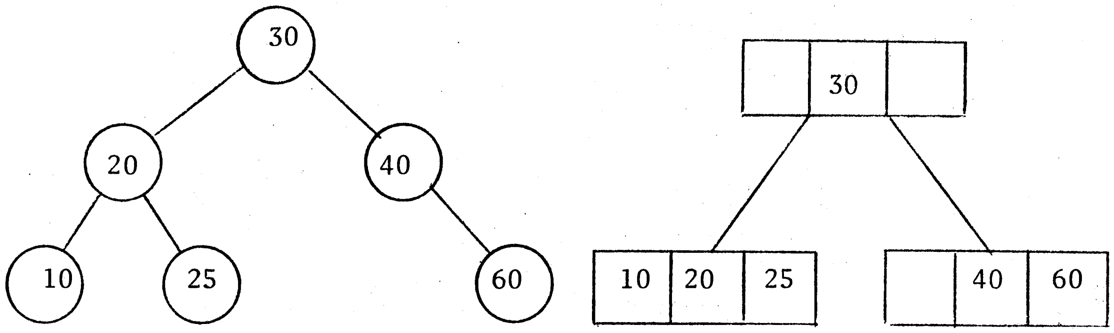
Insertion below a leaf node which has a brother node that is also a leaf may cause an AVL tree imbalance in some instances and not in others. If an imbalance does occur it will always occur at some node higher in the tree than the grandfather of the node being inserted. The trees diagrammed in Figure 29 show the different situations that may occur when an imbalance arises. If a single rotation is required to restructure the AVL tree, the corresponding SBB-tree operation will be an overflow of two keys from the node in which the insertion occurs to its immediately adjacent brother. This is shown in Figure 29b. If a double rotation is required to restructure the AVL tree after insertion there are two possible SBB-tree operations that may occur. These are shown in Figures 29c and 29d. In the first case key 25 is inserted in the left subtree of the AVL tree of Figure 29a. After restructuring it remains in the left subtree and the corresponding SBB-tree operation is that of a one-key overflow to an immediately adjacent brother node. In the second case key 35 is inserted in the left subtree of the AVL tree of Figure 29a but after restructuring it has been shifted to the right subtree. This corresponds to a two key overflow in the SBB-tree representation. The fact that the number



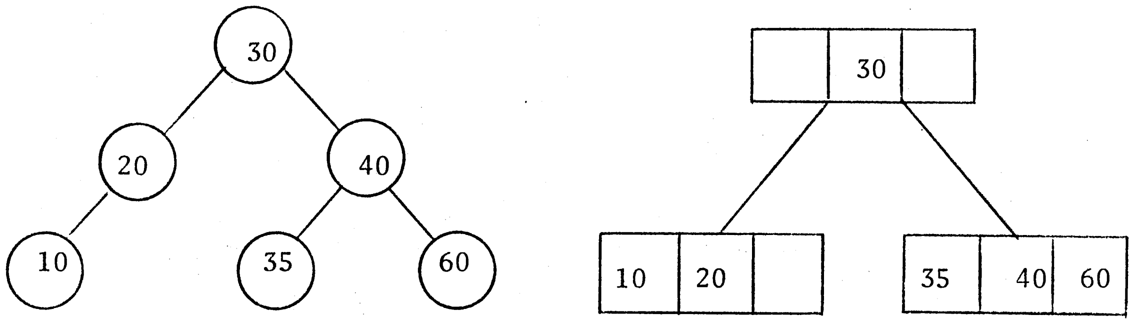
a.) Example AVL Tree and Corresponding SBB-Tree



b.) AVL Tree of a.) and Corresponding SBB-Tree After Inserting Key 15 and Performing a Single Rotation



c.) AVL Tree of a.) and Corresponding SBB-Tree After Inserting Key 25 and Performing a Double Rotation

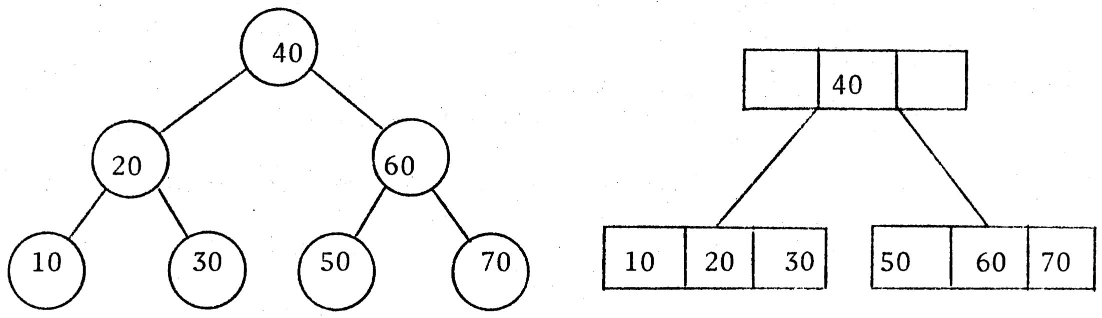


d.) AVL Tree of a.) and Corresponding SBB-Tree After Inserting Key 35 and Performing a Double Rotation

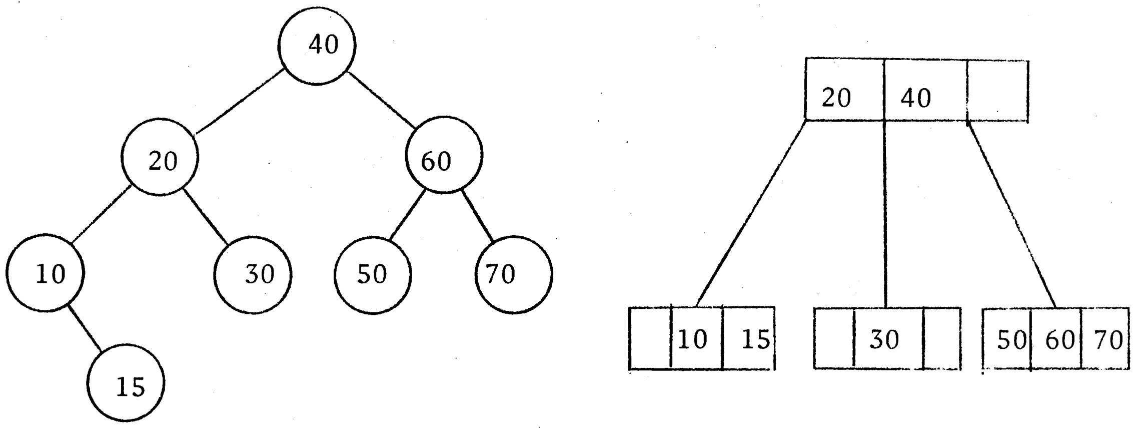
Figure 29. Insertion Involving an Imbalance at a Node Above the Grandfather Level

of keys in the second level node of the SBB-tree representation is not changed in any of the above three situations is significant. It means that the second level may be ignored in forming transition relationships for these cases. This, however, is not the situation for insertions of this type which do not cause imbalance in the AVL tree.

Cases where the fourth type of insertion occurs in AVL trees and no imbalance arises correspond in the SBB-tree sense to insertion in a full node where the adjacent brother node does not have enough empty key slots to accept an overflow. Figure 30 illustrates one example of such a situation. The node in which insertion was attempted was full so that a node split occurred with key 20 which previously occupied the center position of the node being promoted to the next δ-level up. This instance in which the number of keys in a node in the second level is changed makes impossible an SBB-tree analysis such as the one in chapter three which involved only the lowest internal level. The need for maintaining a knowledge of the number of keys in brother nodes at the bottom level due to overflow considerations is the basic cause of this problem. Figure 31 demonstrates that insertion situations of this last type described may cause node changes in any level above the bottom internal level; therefore, an analysis based on the two lowest internal levels would still not give a completely accurate result. Here as before it is the categorization of nodes that determines the degree of difficulty of the analysis to be done.

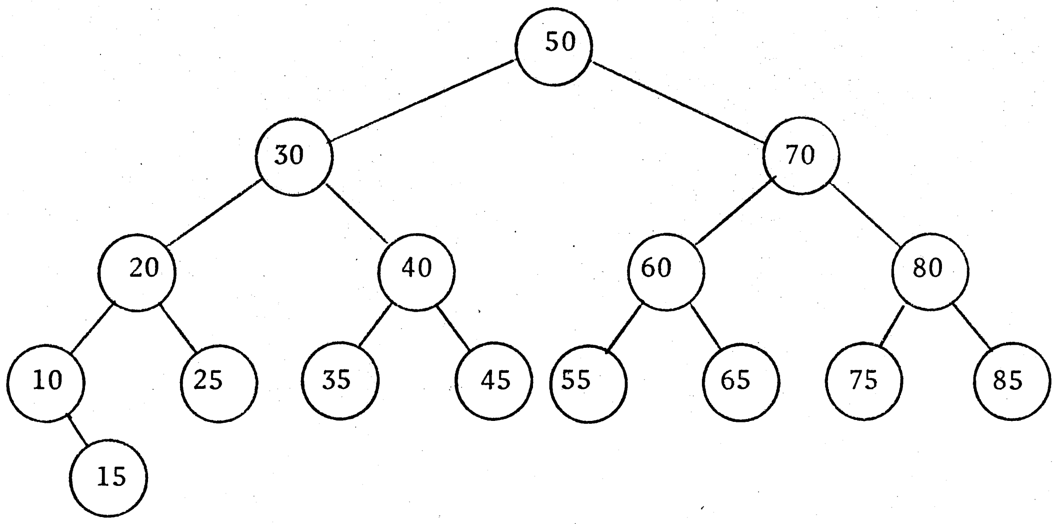


a.) An Example AVL Tree and Corresponding SBB-Tree

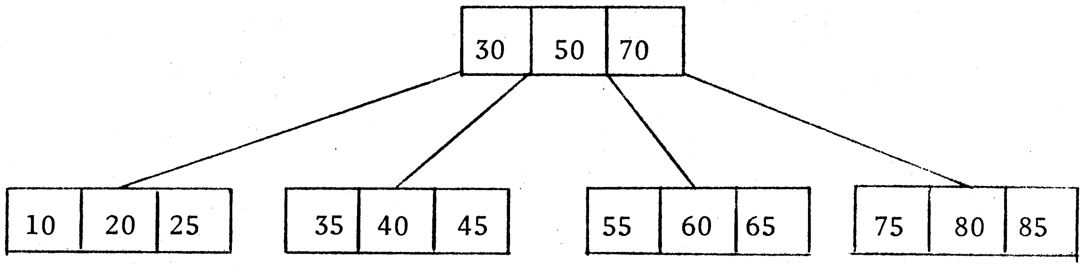


b.) AVL Tree of a.) and Corresponding SBB-Tree After Inserting Key 15

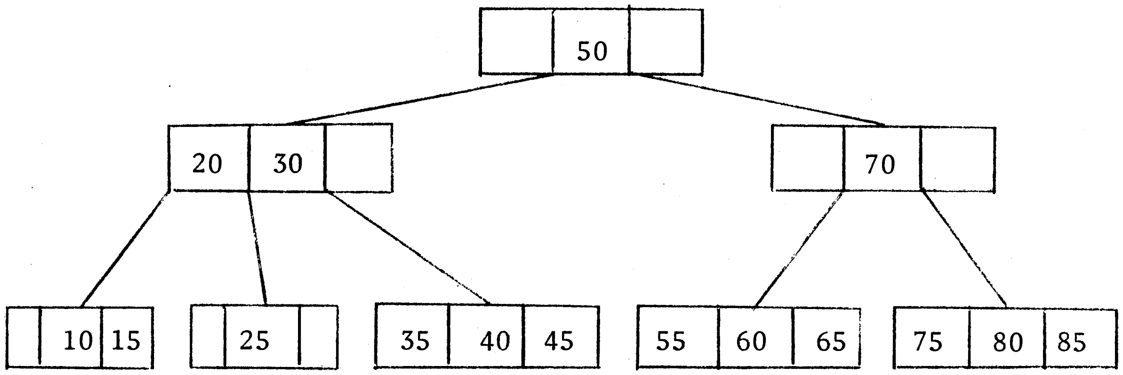
Figure 30. Insertion Involving No Imbalance in AVL Tree, but Involving a Node Split in the Corresponding SBB-Tree



a.) An Example AVL Tree



b.) SBB-Tree Corresponding to the Tree of a.) Before Insertion of Key 15



c.) SBB-Tree Corresponding to the Tree of a.) After Insertion of Key 15

Figure 31. AVL Insertion with Corresponding SBB-Tree Maintenance at Higher Levels

Further Analysis of AVL Trees

It has been shown above that an AVL tree may be interpreted as an SBB-tree which in turn may be interpreted as a 4-3 Tree. Using a 4-3 tree model for AVL trees and ignoring any rotations that may occur at AVL tree levels higher than three above the inserted node, a probabilistic analysis may be conducted in the following manner.

States are assigned to the external nodes according to their AVL tree relationships. They are not assigned as before, according to the number of keys in the father node of the B-tree representation. An external node is in state #1 if it has a leaf node father and a semi-leaf node grandfather. It is in state #2 if it has a semi-leaf node father. If an external node has a leaf node father which has a leaf node brother than it is in state #3. State #4 consists of external nodes having a leaf node father which has a semi-leaf node brother and state #5 consists of external nodes having a leaf node father whose brother has two sons. Examples of external nodes in each of these states can be seen in Figure 32.

Formulation of the state transition equations proceeds as in previous analyses relating the state of the (N+1) external nodes after insertion to the states of N external nodes immediately before an insertion. The resulting equations are:

$$(N+1)\cdot P_{N+1} = (N\cdot P_N)-(2\cdot Q_N)+(2\cdot R_N)+(2\cdot U_N)+(2\cdot V_N) \qquad (4.13)$$

$$(N+1)\cdot Q_{N+1} = (N\cdot Q_N)-(1\cdot P_N)-(1\cdot Q_N)+(1\cdot R_N)+(1\cdot U_N)+(1\cdot V_N) \qquad (4.14)$$

$$(N+1)\cdot R_{N+1} = (N\cdot R_N)+(4\cdot P_N)+(4\cdot Q_N)-(4\cdot R_N)+(4\cdot V_N) \qquad (4.15)$$

$$(N+1)\cdot U_{N+1} = (N\cdot U_N)+(2\cdot R_N)-(2\cdot U_N)-(2.(3/2)\cdot U_N) \qquad (4.16)$$

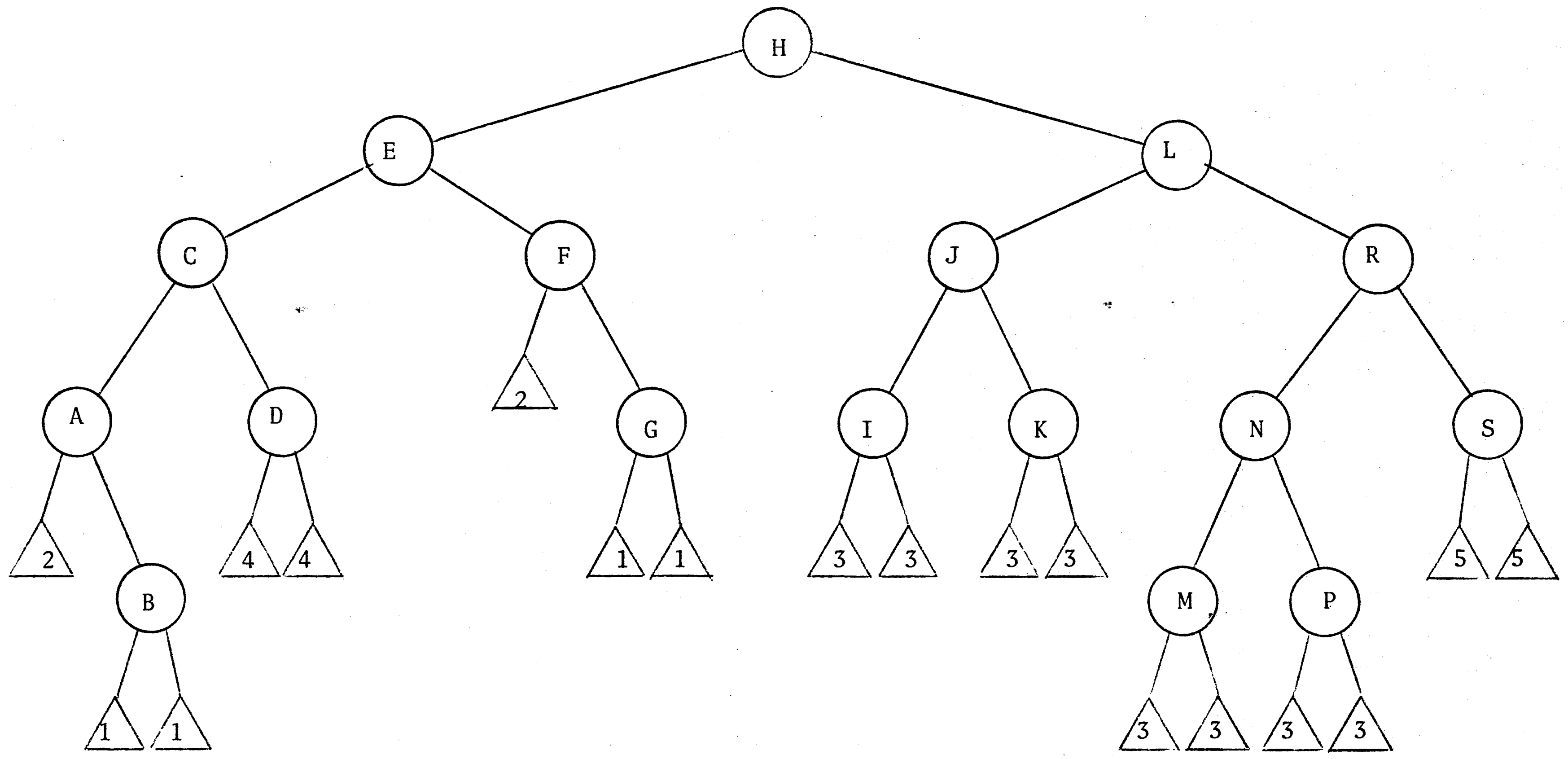


Figure 32. Classification of AVL Tree External Nodes

$$(N+1)\cdot V_{N+1} = (N\cdot V_N)+(2\cdot(3/2)\cdot U_N)-(2\cdot V_N)-(2\cdot(4/2)\cdot V_N) \qquad (4.17)$$

The coefficients obtained for equations (4.13) and (4.14) are straightforward, but those obtained for equations (4.15), (4.16), and (4.17) require further explanation. In equation (4.15) the insertion of a new node in an external node position which is in either state #1 or state #2 will cause the addition of four external nodes in state #3 (terms $(4\cdot P_N)$ and $(4\cdot Q_N)$. The insertion of a new node in a state #3 external node position causes the loss of four state #3 external nodes unless those nodes are arranged in the situation depicted by the subtree with root node R in Figure 32. In this situation restructuring occurs and the four state #3 external nodes are retained. Writing this in terms of external node probabilities, four external nodes are subtracted for all cases of insertion in state #3 external nodes as $(-4\cdot R_N)$ and those cases where the state #3 and state #5 nodes occur in combination are added back as $(4\cdot V_N)$. The same type of situation arises in equation (4.16). Two state #4 external nodes are gained whenever a new node is inserted in a state #3 external node position (term $(2\cdot R_N)$) and two state #4 external nodes are lost every time a new node is inserted in a state #4 external node position (term $(-2\cdot U_N)$). Two state #4 external nodes will also be lost when insertion occurs in a state #1 or state #2 external node position and these external nodes are arranged in combination with a state #4 external node as in the subtree with root node C in Figure 32. This situation arises with probability U_N and as can be seen in Figure 32 there are three external node positions below nodes A and B and only two below node D. Multiplying this ratio by the probability of the situation gives $(3/2\cdot U_N)$ for the probability of losing two state #4 external nodes (term $(-2\cdot(3/2)\cdot U_N)$). Two state #4 external nodes will not be lost if insertion occurs at state #1 and state #2 external nodes which are without a corresponding state #4 external node as is the case for the external nodes below nodes G and F in Figure A. Any time insertion occurs as in the last situation discussed for equation (4.16) two state #5 external nodes will be created (term $(2\cdot(3/2)\cdot U_N)$ in equation (4.17)). Two state #5 external nodes are destroyed when insertion occurs in a state #5 external node position (term $(2\cdot V_N)$). Two such external nodes are also lost upon insertion in a state #3 external node when such an insertion causes restructuring. This situation, the same as discussed for equation (4.15), occurs with probability V_N and the ratio of state #3 external nodes to state #5 external nodes is four to two. The resulting term in equation (4.17) is $(-2\cdot(4/2)\cdot V_N)$.

Analysis conducted earlier in this chapter on the leaf and semi-leaf nodes of AVL trees showed the limiting probability for state #1 and state #2 nodes to be three-sevenths (P+Q = 3/7) and the probability for the remaining states to be four-sevenths (R+U+V = 4/7). Using these results, equations (4.13)-(4.17) can be solved for the limiting probabilities of states #1-#5 as follows:

Rewriting equations (4.16) and (4.17) gives:

$$U = (1/3)\cdot R \tag{4.18}$$

$$V = (3/7)\cdot U = (3/7)\cdot(1/3)\cdot R = (1/7)\cdot R \tag{4.19}$$

Substituting the expressions for U and V into the relationship

$$R+U+V = 4/7 \tag{4.20}$$

gives:

$$R+(1/3)\cdot R+(1/7)\cdot R = 4/7 \tag{4.21}$$

Solving for R yields:

$$R = 12/31 \tag{4.22}$$

Substitution of the value of R in equations (4.18) and (4.19) yields:

$$U = 4/31 \tag{4.23}$$

$$V = 24/217 \tag{4.24}$$

As in the previous analysis of states #1 and #2:

$$P = 2/7 \tag{4.25}$$

$$Q = 1/7 \tag{4.26}$$

Relating these results to the 4-3 tree model, it can be seen from Figure 33 that (P+Q) is the probability that an external node has a two-key father node; that (U+V) is the probability that an external node has a one-key father node; and that R is the probability that an external node has a three-key father node. Thus:

$$P+Q = 3/7 \tag{4.27}$$

$$U+V = 40/217 \tag{4.28}$$

$$R = 12/31 \tag{4.29}$$

For a tree of N keys the expected number of keys in one-key father nodes in the bottom internal level is:

$$(U+V)\cdot(N+1)\cdot(1/2) = (40/217)\cdot(N+1)\cdot(1/2) = 20/217 \tag{4.30}$$

where the factor 1/2 is the ratio of internal keys to external nodes for a father node of this type. Similarly for two- and three-key father nodes the expected number of keys will be:

$$(P+Q)\cdot(N+1)-(2/3) = (3/7)\cdot(N+1)\cdot(2/3) = 2/7\cdot(N+1) \tag{4.31}$$

$$(R)\cdot(N+1)\cdot(3/4) = (12/39)\cdot(N+1)\cdot(3/4) = 9/31\cdot(N+1) \tag{4.32}$$

Dividing each of the results in equations (4.30)-(4.32) by the number of keys in the corresponding node type gives the expected number of nodes of each type in the bottom internal level as:

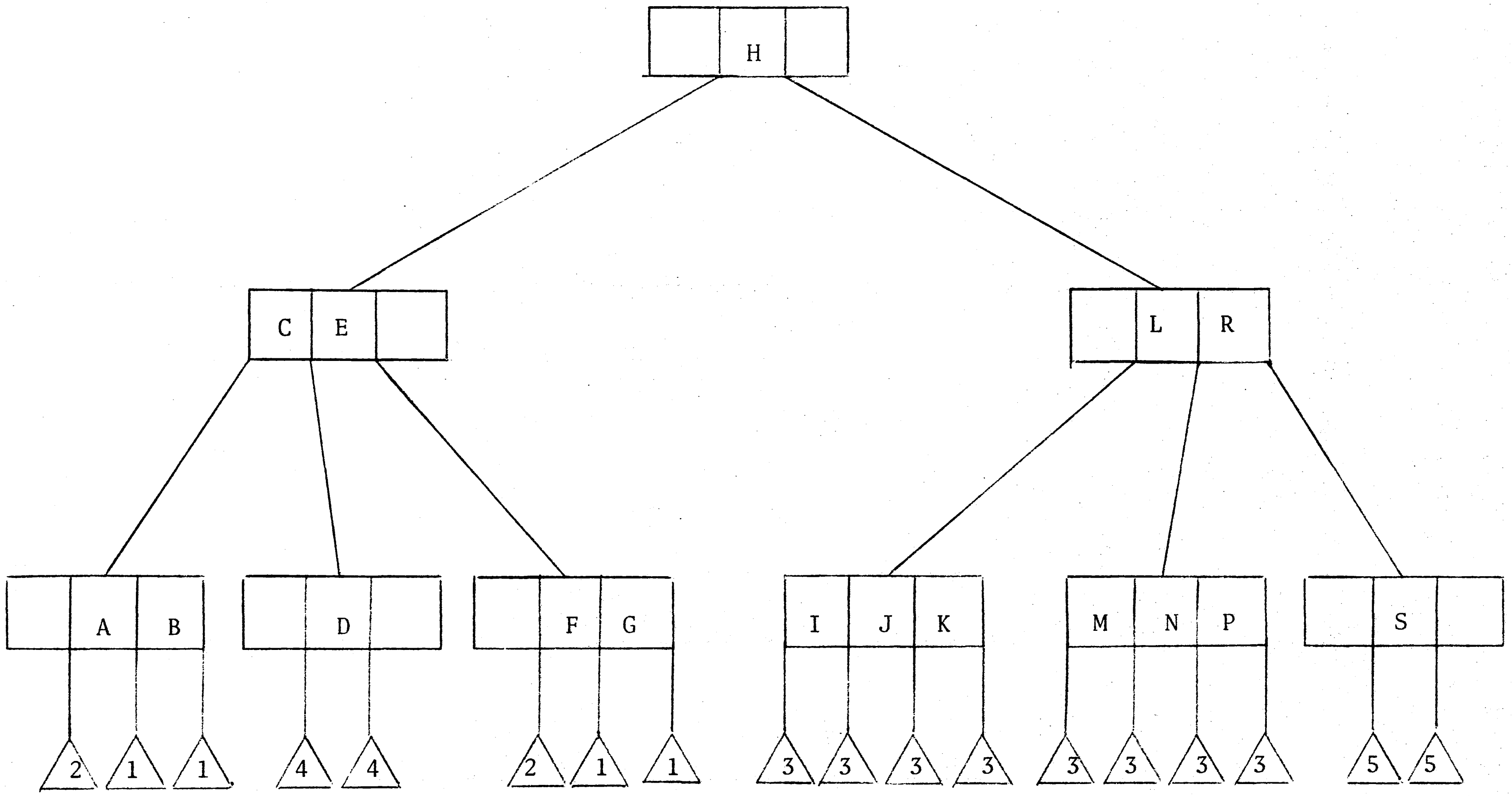


Figure 33. 4-3 Tree Model of the AVL Tree in Figure 32

$$(20/217)\cdot(N+1) \text{ nodes containing one key} \tag{4.33}$$

$$(1/7)\cdot(N+1) \text{ nodes containing two keys} \tag{4.34}$$

$$(3/31)\cdot(N+1) \text{ nodes containing three keys} \tag{4.35}$$

Summing the values in (4.30)-(4.32) gives the expected number of keys in the bottom internal level as:

$$(20/217+2/7+9/31)\cdot(N+1) \approx 145/217\cdot(N+1) \tag{4.36}$$

Likewise the expected number of nodes in the bottom internal level is the sum of values from (4.33)-(4.35):

$$(20/217+1/7+3/31)\cdot(N+1) = 72/217\cdot(N+1) \tag{4.37}$$

The utilization at the bottom internal level is:

$$145/217\cdot(N+1)\text{Keys}/3\cdot(72/217)\cdot(N+1)\text{Key slots} \approx .6713 \tag{4.38}$$

From the utilization the average number of keys per node can be calculated as:

$$.6713\cdot 3 = 2.014 \text{ keys/node} \tag{4.39}$$

The average branching from a node is one greater than this or:

$$2.014+1 = 3.014 \text{ external nodes/node} \tag{4.40}$$

Assuming probabilistic independence of levels and extending the above results to all levels of the tree allows approximation of an expected number of levels in a tree of (N+1) keys. Following a development similar to that used in Chapter three to arrive at an expression of the approximate expected number of tree levels (m) gives:

$$m = \log_{3.014}(N+1) \tag{4.41}$$

for the 4-3 tree interpretation of AVL trees. The average number of binary tree levels or key comparisons per 4-3 tree level may be calculated also following the logic of chapter three as the average number of one-key comparisons plus twice the average number of two key comparisons, or:

$$40/217+(1/3)\cdot(3/7)+\ (2/3)\cdot(3/7)+(12/31)\ \cdot 2 = 1.673 \qquad (4.42)$$

Multiplying the results of equations (4.41) and (4.42) gives an approximate expected number of binary tree levels for the 4-3 tree model of an AVL tree as:

$$1.673\cdot\log_{3.014}(N+1) \qquad (4.43)$$

Comparing this with the expression for expected path length to an external node for a completely balanced binary tree yields:

$$\frac{1.673\ \log_{3.014}(N+1)}{\log_{2.0}(N+1)} = (1.673)\ \frac{\log_{10}2.0}{\log_{10}3.014} \approx 1.055 \qquad (4.44)$$

This result is dependent on several assumptions:

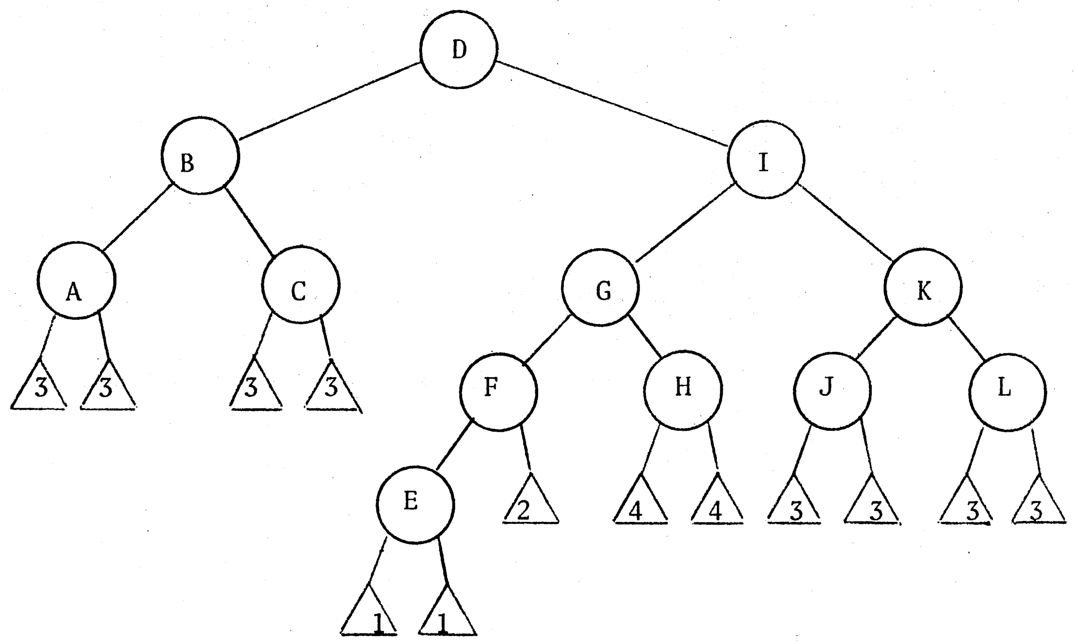
1) All 4-3 tree levels have identical asymptotic characteristics

2) All 4-3 tree levels are statistically independent

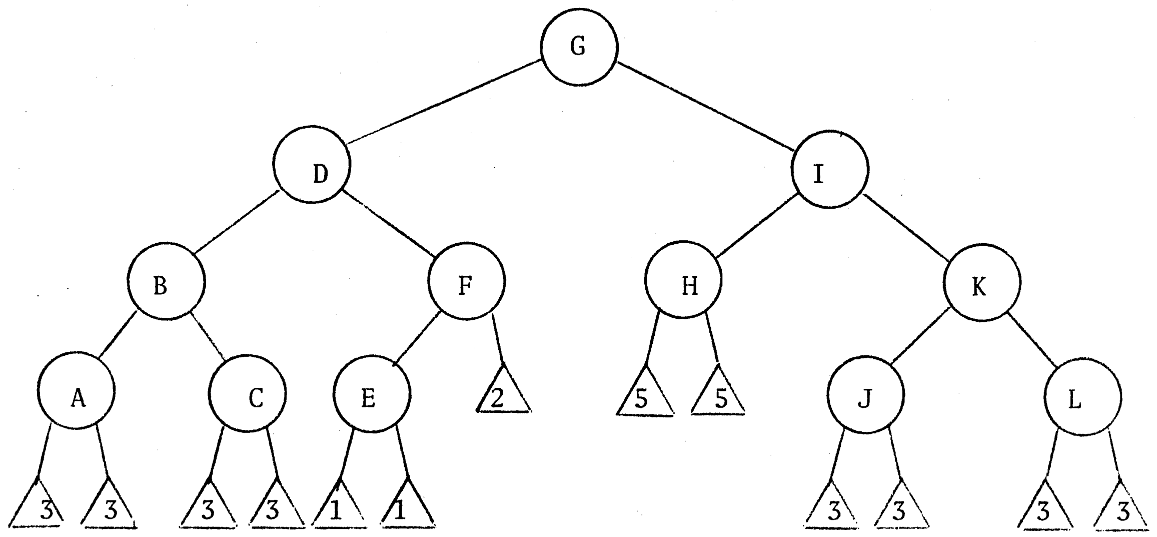
3) All AVL rotations are ignored at levels higher than three levels above the inserted node.

Qualitatively the effect of this last assumption can be examined in the example of Figure 34. Here a double rotation occurs at a level higher than three above the node creating the imbalance, node E. The only node states that are affected by the restructuring are states #4 and #5 and since these both correspond to one-key 4-3 tree nodes there is no immediate effect at this step on the 4-3 tree analysis. However, the probability that a state #4 or state #5 external node will be an insertion position at a later step in the process is affected and assumption 3) neglects this effect in the state transition equations.

Nevertheless, the above analysis lends additional credence to the conjecture offered by Knuth (7, p. 460) that the average external path length of balanced trees is approximately $\log_2 N+.75$ where N is the



a.) An Unbalanced Tree Showing External Node Classifications



b.) Tree of a.) After Double Rotation at Node D

Figure 34. Effect of Rebalancing at Higher AVL Tree Levels

number of keys internal to the tree.

It is known that the path length to an external node in a balanced binary tree is bounded by 1.4404 $\log_2(N+2)-0.328$ (7, p. 453), but the asymptotic average for this path length has never been rigorously determined. Empirical evidence concerning this average supports the conjecture mentioned above.

It is hoped that the analysis given here will be of some assistance in eventually resolving this matter.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER RESEARCH

A method has been presented in this study for analyzing search trees by utilizing probability concepts involving Markov chains. The development of this method is presented for randomly built 3-2 trees maintained without overflow. Extension of the method of analysis to the 4-3 tree interpretation of SBB-trees is then presented to provide a point of correlation between B-tree analysis and binary tree anlaysis as well as to show expanded applicability of the method. Finally, the behavior of balanced trees is inspected. In all cases only the simplest of maintenance operations and state transition relationships are considered. The application of analytical results to all levels of a tree structure is based upon simplifying assumptions. Suggestions that the results of this type analysis are credible are based on comparisons with results of limited empirical testing. Extension of the method of analysis to a multilevel analysis as recommended below should, however, produce more refined results.

There is a definite need for empirical testing of the SBB-tree data structure for comparison with the analytical results obtained in this paper. Some of the properties of this structure make it a prime candidate for further research. For example, the concepts of vertical and horizontal binary tree levels could be extended to obtain a correspondence among classes of binary search trees and other multikey-multibranch node structures. This might allow an analysis technique developed for one type of structure to be extended to another via the correspondence.

Several possible extensions of the method of analysis as presented offer areas for further investigation. The area of key deletion is one example. It may be desirable in many cases to know the expected performance of a tree structure under conditions of insertions and deletions of keys according to some statistically determined pattern. This could also involve extension of the method to include provisions for tree structures in which the probabilities of the insertion positions are not uniformly distributed. The analysis of B-trees of higher branching order could be accomplished in a straightforward manner with the only difficulties arising from the number of equations involved and from the solution of these equations for the stable state probability values. The alleviation of these problems might be accomplished by the development of some generalized algorithms for generating the state transition equations, given the characteristics of a tree structure, and the application of appropriate methods for solving such large sets of equations. The analysis of external nodes in terms of higher tree levels implies a dramatic increase in the number and complexity of the state transition relationships. However, the ability to extend the analysis to multiple levels of a tree structure would eliminate areas of uncertainty introduced by the assumption of probabilistic independence of levels. There is also a need to analyze B-tree structures utilizing such maintenance operations as overflow and multiway splitting. Comparisons with empirical results of testing on trees utilizing these types of operations would provide additional evidence concerning the usefulness of the method.

Other legitimate questions may be raised relating to how a structure proceeds from its initial state to a stable state via the Markov chain process. The concepts presented herein state that for a data structure with a given initial state, maintenance defined by a Markov chain will ultimately result in a stable state. Research relating rate of convergence to the initial state, the type of structure, the size of the structure, etc. is needed. In addition, there is need for research on other statistical factors besides the asymptotic averages on which this report has concentrated. Analytical determination of variance and comparison with results of empirical variances obtained in testing is a good example.

The number of areas in which additional work is desirable indicates in part the need for further research in the field of data structures analysis. Hopefully the topics covered in this study will provide insight on these matters.

BIBLIOGRAPHY

(1) Bayer, R. and E. McCreight. "Organization and Maintenance of Large Ordered Indexes." Acta Informatica, 1 (1972), 173-189.

(2) Bayer, R. "Symmetric Binary B-Trees: Data Structure and Maintenance Algorithms." Acta Informatica, 1 (1972), 290-306.

(3) Davis, William S. "Empirical Behavior of B-Trees." (Unpub. Masters thesis, Oklahoma State University, 1974.)

(4) Fisher, D. D. "Data Structures, Information Structures, and Information Processing." (Unpub. Classnotes, Oklahoma State University, 1974.)

(5) Kemeny, J. and J. Snell. Finite Markov Chains. New Jersey: D. VanNostrand Company, 1960.

(6) Knuth, D. E. The Art of Computer Programming. Vol. 1. Reading: Addison-Wesley, 1969.

(7) Knuth, D. E. The Art of Computer Programming. Vol. 3. Reading: Addison-Wesley, 1973.

(8) Liu, C. L. Introduction to Combinatorial Mathematics. New York: McGraw-Hill, 1968.

(9) Parzen, E. Stochastic Processes. San Francisco: Holden-Day Inc., 1962.

(10) Van Doren, J. R. and J. L. Gray. "An Algorithm for Maintaining Dynamic AVL Trees." In Information Systems, Vol. 4. New York: Plenum Press, 1974, 161-180.

APPENDIX

APPENDIX

The method of analysis described in chapter two and used throughout this study makes use of several theorems relating to nonhomogeneous Markov chains. The theorems and proofs given are the unpublished work of the author's advisor, Dr. James R. Van Doren. These theorems are extensions to nonhomogeneous Markov chains of theorems by Kemeny and Snell (5) for homogeneous Markov chains.

Theorem 1.

If T(n) is a 2x2 probability transition matrix which is dependent on n and which has no zero entries; and, if certain elements of T(n) are defined as follows:

1) ε_n is the smallest entry in T(n)
2) X is any two-component column vector
3) M_0 is the larger element of X
4) m_0 is the larger element of X
5) M_1 is the larger element of the vector $T(n) \cdot X$
6) m_1 is the smaller element of the vector $T(n) \cdot X$

the the following are true:

1) $M_1 \leq M_0$
2) $m_1 \geq m_0$
3) $M_1 - m_1 \leq (1-2\cdot\varepsilon_n)(M_0-m_0)$

Proof of Theorem 1.

Since T(n) is a probability transition matrix its row sums must equal one; thus, the elements of any row may be represented as (a, 1-a) where $\varepsilon_n \le a \le 1$ and $\varepsilon_n \le 1 - a \le 1$. The elements M_1 and m_1, of the product vector $T(n)\cdot X$, will therefore have the form

$$a\cdot m_0 + (1-a)\cdot M_0$$

and any convex combination of two numbers lies between them. If the element in question is M_1, then

$$\begin{aligned} M_1 &= a\cdot m_0 + (1-a)\cdot M_0 \\ &= M_0 - a(M_0 - m_0) \end{aligned}$$

Substituting ε_n for (a) where $a \ge \varepsilon_n > 0$ yields

$$M_1 \le M_0 - \varepsilon_n(M_0 - m_0) \le M_0$$

This proves part 1).

Similarly if the element in question is m_1, then

$$\begin{aligned} m_1 &= a\cdot m_0 + (1-a)\cdot M_0 \\ &= (a-1)\cdot M_0 + m_0 \\ &= (1-a)(M_0 - m_0) + m_0 \end{aligned}$$

Substituting ε_n for (1-a) where $(1-a) \ge \varepsilon_n > 0$ yields

$$m_1 \ge \varepsilon_n(M_0 - m_0) + m_0 \ge m_0$$

This proves part 2).

Reversing the second inequality

$$-m_1 \le -m_0 - \varepsilon_n(M_0 - m_0)$$

and adding it to the first inequality

$$M_1 \ge M_0 - \varepsilon_n(M_0 - m_0)$$

yields

$$\begin{aligned} M_1 - m_1 &\le M_0 - m_0 - 2\cdot\varepsilon_n(M_0 - m_0) \\ &\le (M_0 - m_0)(1 - 2\cdot\varepsilon_n) \end{aligned}$$

This proves Part 3).

Q. E. D.

Theorem 2

If $$P_n = \prod_{i=k}^{n} T(i) \quad \text{where } \{T(i): \ i=k,\dots,n\}$$

is a sequence of 2x2 probability matrices which have nonzero entries and are commutative; and, if the sequence:

$$\{\prod_{i=k}^{n} (1-2\cdot\varepsilon_i) \ , \ n=k,\dots.\}$$

where ε_i is the minimum entry in T(i), converges to zero,

Then the following are true:

1) The sequence of probability matrices P_n has limit

$$\lim_{n\to\infty} P_n = A$$

2) Each row of A has the same probability vector α where $\alpha = (a_1, a_2)$, $a_1+a_2 = 1$, and $a_1, a_2 > 0$.

Thus:

$$A = \begin{bmatrix} a_1 & a_2 \\ a_1 & a_2 \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha \end{bmatrix}$$

Proof of Theorem 2.

Let ρ_j be a two element column vector in which the j-th component is one and the other component is zero; and let M_n and m_n be the maximum and minimum components respectively of the column vector $P_n\cdot\rho_j$.

As defined above $P_n=P_{n-1}\cdot T(n)$ and due to the commutativity of the sequence, $P_n = T(n)\cdot P_{n-1}$. Hence

$$P_n\cdot\rho_j = T(n)\cdot P_{n-1}\cdot\rho j$$

where M_{n-1} and m_{n-1} are the maximum and minimum components of $P_{n-1}\cdot\rho_j$ by definition.

From Theorem 1

$$M_{n-1} \geq M_n \quad \text{and} \quad m_{n-1} \leq m_n$$

which may be extended from k to n as

$$M_k \geq M_{k+1} \geq \cdots \geq M_{n-1} \geq M_n$$

and

$$m_k \leq m_{k+1} \leq \cdots \leq m_{n-1} \geq m_n$$

Also from Theorem 1

$$M_n - m_n = (1-2\cdot\varepsilon_n)(M_{n-1}-m_{n-1})$$

for $n > k$

Defining d_n by:

$$d_n = M_n - m_n \qquad \text{for } n=k, \ldots.$$

where $\quad d_{k-1} = 1$

Then: $\quad d_n \leq (1-2\cdot\varepsilon_n)\cdot d_{n-1} \leq (1-2\cdot\varepsilon_n)(1-2\cdot\varepsilon_{n-1})\cdot d_{n-2} \leq \cdots$

Rewriting in product form:

$$d_n \leq d_{k-1} \prod_{i=k}^{n} (1-2\cdot\varepsilon_i) = \prod_{i=k}^{n} (1-2\cdot\varepsilon_i)$$

By hypothesis this sequence converges to zero; therefore d_n approaches zero as n approaches infinity.

Thus $P_n\cdot\rho_j$ approaches a vector with identical components, say a_j.

Since $P_n\cdot\rho_j$ is in fact the j-th column of P_n, P_n approaches a matrix with all rows identical.

$$P_n \rightarrow \begin{bmatrix} \alpha \\ \alpha \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_1 & a_2 \end{bmatrix}$$

Since a_j must follow the relationship $m_n \leq a_j \leq M_n$ by definition for all $n \geq K$, and $m_k > 0$ while $M_k > 1$, $0 < a_j < 1$.

Since the row sums of P_n are one this must also be true of the limit.

Q. E. D.

<u>Application of Theorem 2</u>:

If

$$T(i) = \begin{bmatrix} \frac{i-1}{i+2} & \frac{3}{i+2} \\ \frac{4}{i+2} & \frac{i-2}{i+2} \end{bmatrix}$$

the the matrix product $T(k)\cdot T(j) = T(j)\cdot T(k)$ for k and j, any two values of i, because:

$$T(i) = I + \frac{1}{i+2}\begin{bmatrix} -3 & 3 \\ 4 & -4 \end{bmatrix}$$

and $\frac{3}{i+2}$ will be the smallest element of T(i), ε_i.

$$(1-2\cdot\varepsilon_i) = (1-\frac{6}{i+2}) = \frac{i+6-2}{i+2} = \frac{i-4}{i+2} < \frac{i+1}{i+2}$$

Therefore: $$\prod_{i=k}^{n}(1-2\cdot\varepsilon_i) < \prod_{i=k}^{n}\frac{i+1}{i+2}$$

and since $$\prod_{i=k}^{n}\frac{n+1}{i+2} = \frac{k+1}{k+2}\cdot\frac{k+2}{k+3}\cdot\ \cdots\ \frac{n}{n+1}\ \frac{n+1}{n+2} = \frac{k+1}{n+2}$$

which approaches zero as n approaches infinity, the product $\prod_{i=k}^{n}(1-2\cdot\varepsilon_i)$ also approaches zero as n approaches infinity.

Thus the sequence $\prod_{i=k}^{n} T(i)$, $n=k,\ldots$ converges to some limiting probability matrix with identical rows.

Theorem 3

If T(i), P_n, α, and A are defined as in Theorem 2.

The the following are true:

1) For any probability vector δ

$$\alpha = \lim_{n\to\infty} \delta\cdot P_n$$

2) is the unique probability vector such that

$$\alpha\cdot T(i) = \alpha$$

3) $T(i)\cdot A = A\cdot T(i) = A$

Proof of Theorem 3.

If δ is a two element probability row vector then

$$\delta \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1$$

Since

$$A = \begin{bmatrix} a_1 & a_2 \\ a_1 & a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot (a_1, a_2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \alpha$$

the product

$$\delta \cdot A = \delta \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \alpha = \alpha$$

Since $\delta \cdot P_n$ approaches $\delta \cdot A$, $\delta \cdot P_n$ approaches α.

This proves part 1).

P_n and T(i) are commutative so that

$$P_n \cdot T(i) = T(i) \cdot P_n$$

From Theorem 2, part 1) P_N approaches A as n approaches ∞. Therefore:

$$P_N \cdot T(i) \text{ approaches } A \cdot T(i)$$

and $T(i) \cdot P_n$ approaches $T(i) \cdot A$

and $T(i) \cdot A = A \cdot T(i)$

Since T(i) is in essence a column of row probability vectors, it follows from Part 1) above that each row of $T(i) \cdot A$ is α

Therefore: $T(i) \cdot A = A \cdot T(i) = A$

This proves part 3).

Rewriting the equality of part 3) with $A = \begin{vmatrix} \alpha \\ \alpha \end{vmatrix}$ gives

$$\begin{bmatrix} \alpha \\ \alpha \end{bmatrix} \cdot T(i) = \begin{bmatrix} \alpha \\ \alpha \end{bmatrix}$$

Thus for each row of the product

$$\alpha \cdot T(i) = \alpha$$

This proves part 2).

The uniqueness of α can be shown since for any probability vector such that $\beta \cdot T(i) = \beta$, $\beta \cdot P_n$ approaches α according to Part 1) above. It follows that if $\beta \cdot T(i) = \beta$, $\beta \cdot P_n = \beta$ and therefore $\alpha = \beta$.

Q. E. D.

By straightforward modifications the above theorems may be extended to transition matrices of any dimension. Furthermore, the requirement that transition matrices have nonzero entries may be relaxed if it can be shown that there exists a transition path of the same length from any state to every state in the Markov chain. In effect this means that there must be a transition matrix with all nonzero entries which is the product of two or more transition matrices that may contain zero entries.

VITA

Dale LeRoy Mitchell, Jr.

Candidate for the Degree of

Master of Science

Thesis: A PROBABILISTIC METHOD FOR ANALYZING SEARCH TREES

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Russellville, Arkansas, February 10, 1951, the son of Mr. and Mrs. Dale L. Mitchell, Sr.

Education: Graduated from Edmond High School, Edmond, Oklahoma in May, 1969; received Bachelor of Science degree from Oklahoma State University, Stillwater, Oklahoma, in December, 1973, with a major in Electrical Engineering; completed requirements for the Master of Science degree at Oklahoma State University, Stillwater, Oklahoma, in July, 1975.

Professional Experience: Audio Engineer, Division of Public Information, Oklahoma State University, Stillwater, Oklahoma, September, 1970, to December, 1973; Registered Engineer in Training, Oklahoma Board of Registration for Professional Engineers and Land Surveyors, Oklahoma City, Oklahoma, January, 1974.