Proceedings of the 1991 Nineteenth Annual Computer Science Conference, March 5-7, 1991, San Antonio, Texas, pp.61-70

Fringe Analysis of Binary Search Trees with Minimal Internal Path Length

Faris N. Abuali
Roger L. Wainwright

Department of Mathematical and Computer Sciences
The University of Tulsa
Tulsa, Oklahoma 74104-3189

ABSTRACT

Binary search trees are well known as a method for organizing information which supports efficient insert, delete, and search operations. Considerable attention has been given recently to maintaining the balance in binary search trees. A recent algorithm has been developed which maintains a balanced tree by displacing keys, when necessary, using an in-order shift algorithm (ISA). This tree has minimum internal path length, and we call such a tree an ISA tree. In this paper we investigate the cost of maintaining an ISA tree. We use fringe analysis to investigate the behavior of such trees. Several theorems and lemmas are presented describing the nature of ISA trees and how they grow and maintain minimum internal path length. We use this anaylsis to determine the expected number of times an in-order shift will be required in the construction of an ISA tree. Furthermore, we define the cost of each possible in-order shift operation, and determine empirically the expected cost to construct and maintain an ISA tree. We present some interesting patterns that emerge in the construction of these trees from one level to another. We conclude that most of the time maintainance of an ISA tree is quite inexpensive. However, in certain instances it can be quite expensive. The cost can be justified, however, if searching is the predominant activity, and inserting is relatively infrequent in comparison.

Key Words Binary Search Trees, In-order Shifting, Internal Path Length, Expected values, Optimal Searching, Fringe Analysis

1. INTRODUCTION

Binary search trees are a method of organizing information which enables search, insert, and delete operations to be performed efficiently. Binary search trees can have a worst case search path length of $n-1$, and a best case search path length of $\log_2 n$, where n is the number of nodes in the tree. A binary search tree that grows in a random manner will have an expected search path length of $1.386 \log_2 n$ [5]. Thus, significant search time improvement (perhaps as much as 38% over the random case) can occur if a binary search tree is kept at minimum height at all times. Considerable attention has been given to maintaining the balance in binary search trees. Generally, the strategies can be divided into either global rebalancing strategies or local rebalancing strategies. Local rebalancing strategies detect an imbalance in the search tree and restores the balance of the tree to within acceptable limits. The most popular local rebalancing strategy is the "AVL rotations" for height balanced trees. Other local rebalancing strategies include BB trees or weight balanced trees [6], and a strategy geared on the total internal path length of the tree [4]. Global rebalancing strategies are generally much more expensive to perform than local rebalancing strategies. All of the nodes of the tree must be inspected in order to restore the balance. Various global rebalancing strategies can be found in [2,3,7].

Gerasch [3] recently introduced a balancing algorithm for binary search trees. This algorithm has an overall run-time performance that falls between that of the local and global rebalancing algorithms. The trees produced by this algorithm have minimum internal path length and are balanced with respect to both the number of complete levels and the number of levels in the left and right subtrees of each node in the tree. He presents an insertion algorithm that maintains the minimal internal path length by displacing keys when necessary, in an in-order fashion, until a vacant position

is found somewhere in the last level of the tree. In this paper we will refer to this algorithm as the ISA algorithm (in-order shift algorithm). He shows that the ISA algorithm produces trees that are optimal for searching while exhibiting run-time behavior between logarithmic and linear in the number of nodes in the tree. Linear time to perform the ISA algorithm is the worst-case behavior. It is assumed the equal likelihood of searches for each key. The cost of maintaining the tree using the ISA algorithm to achieve optimal search performance is expensive. However this is easily justified if searching is the predominant activity, and the insertion of keys in the tree is relatively infrequent in comparison. In this paper, we will further investigate optimal binary search trees with minimal internal path length, using the ISA algorithm presented by Gerasch[3] to maintain optimal balance. The rest of the paper is summarized as follows. In Section 2 we define an ISA tree, and use fringe analysis [1] to present several observations and lemmas concerning the behavior of ISA trees. In Section 3 we analyze the construction of ISA trees. We give theoretical results for both the expected number of in-order shifts required to build an ISA tree, as well as the expected cost of construction. Conclusions are presented in Section 4.

2. FRINGE ANALYSIS FOR ISA TREES

Definition 1. An ISA tree with n nodes ($n > 0$) is a binary search tree that is nearly complete to level L. That is, the tree is complete to level $L-1$ and level L is either empty or is the last non-empty level of the tree. Also the insertion algorithm that maintains minimal internal path length is the ISA algorithm described in [3.]

We define the root of a tree to be at level 0. Clearly, binary trees that are nearly complete to level L are those trees with minimal internal and external path lengths.

Definition 2. Given an ISA tree with n nodes, we define four types of nodes in the fringe (leaf area) of the tree as follows:

1. E-nodes are external nodes.
2. R-nodes are internal nodes on level $L-1$ that have two external nodes as children.
3. S-nodes are internal nodes on level $L-1$ that have one external node and one internal node as children.
4. K-nodes are internal nodes on level L that have two external nodes as children. The number of K-nodes is the number of nodes on level L, and is easily determined. An ISA tree with n nodes that is complete to level $L-1$ will necessarily have $n-(2^L-1)$ K-nodes.

Figure 1 illustrates an example ISA tree nearly complete to level $L=3$ with the R, K, and S-nodes identified.

Lemma 1. Let T be an ISA tree with n nodes, R R-nodes, K K-nodes, S S-nodes, and E E-nodes, then $2*(R+K)+S=n+1$.

Proof. In any binary tree the number of external nodes equals the number of internal nodes plus one (ie., $E = n+1$). It follows from Definition 2, that each S-node, contributes one E-node, and each K-node and R-node each contribute two E-nodes.

Lemma 2. Let T be an ISA tree with n nodes, R R-nodes, K K-nodes, S S-nodes, and E E-nodes, then an insertion which falls into an external node of an S-node produces the following changes in the values of R, K, S, E, and n: R remains unchanged, $S \Leftarrow S-1$ (the number of S-nodes decreases by 1), $K \Leftarrow K+1$ (the number of K-nodes increases by 1), $E \Leftarrow E+1$, and $n \Leftarrow n+1$.

Proof. The proof follows from Definition 2. A node inserted into an external node of an S-node becomes a K-node, and the parent node is no longer an S-node. The values for E and n naturally increase by one.

Lemma 3. Let T be an ISA tree with n nodes, R R-nodes, K K-nodes, S S-nodes, and E E-nodes, then an insertion which falls into an external node of an R-node produces the following changes in the values of $R, K, S, E, and\ n$: $R \Leftarrow R-1$, $S \Leftarrow S+1$, $K \Leftarrow K+1$, $E \Leftarrow E+1$, and $n \Leftarrow n+1$.

Proof. The proof follows from Definition 2. A node inserted into an external node of an R-node becomes a K-node, and the R-node (parent node) changes to an S-node. The values for E and n naturally increase by one.

Lemma 4. Let T be an ISA tree with n nodes, R R-nodes, K K-nodes, S S-nodes, and E E-nodes, then an insertion which falls into an external node of a K-node produces the following changes in the values of $R, K, S, E, and\ n$: (Note the in-order shift algorithm is required to rebalance the tree except when level L is

empty.)

Case 1. If after the ISA algorithm is performed, a new node is inserted into an external node of an S-node then R remains unchanged, $S \Leftarrow S-1$, and $K \Leftarrow K+1$.

Case 2. If after the ISA algorithm is performed, a new node is inserted into an external node of an R-node then $R \Leftarrow R-1$, $S \Leftarrow S+1$, and $K \Leftarrow K+1$.

Case 3. If $S=0$, and $R=0$ (ie., $2K=n+1$), then a new level of the tree begins and $R \Leftarrow K-1$, $S \Leftarrow 1$, and $K \Leftarrow 1$.

In all cases $E \Leftarrow E+1$, and $n \Leftarrow n+1$.

Proof. The proof follows directly from Lemmas 2, and 3.

Lemma 5. Let T be an ISA tree with n nodes, R R-nodes, K K-nodes, and S S-nodes then the probability of inserting a node in the external leaves of R, K, and S-nodes are $2R/(n+1)$, $2K/(n+1)$, and $S/(n+1)$, respectively.

Proof. Assuming the equal likelihood of inserting each key, this follows directly from Lemma 1.

Thus the probability of inserting a node into a K-node is $2K/(n+1)$. However, this will cause an imbalance in the tree and the ISA algorithm will be used to rebalance the tree. Thus all insertions into an ISA tree ultimately result in the insertion into either an S-node or an R-node, and never into a K-node. The only exception is when the ISA tree is complete and the inserted node is the first node on the next level. This is the same as case 3 described in Lemma 4.

Lemma 6. Let T be an ISA tree with n nodes, R R-nodes, K K-nodes, and S S-nodes, then the probability that inserting a node results in an insertion into the leaf of an R-node or an S-nodes is given by $2R/(2R+S)$, and $S/(2R+S)$, respectively.

Proof. Throughout this paper we assume the equal likelihood of inserting each key. There are $2R+S$ external nodes available for an insertion. The probability the insertion occurs in an external node of an S-node is clearly $S/(2R+S)$, and the probability the insertion will occur in an R-node is $2R/(2R+S)$.

Alternatively, we can calculate the probability of inserting in an external node of an S-node as $S/(n+1)+2K/(n+1)*S/(n+1)$ if we consider K-nodes as well. This reduces to $S/(2R+S)$. Similarly, we can calculate the probability of inserting in an external node of an R-node as $2R/(n+1)+2K/(n+1)*2R/(n+1)$, if we consider K-nodes. This reduces to $2R/(2R+S)$ as expected.

3. CONSTRUCTION OF ISA TREES

3.1 Expected Number of In-order Shifts

Clearly ISA trees offer minimum search times. It is important to consider the cost required for maintaining ISA trees. To the authors' knowledge this has not been fully investigated. In this section we will analyze the cost of the construction and maintenance of an ISA tree. We will first investigate the expected number of S-nodes and R-nodes found in an ISA tree. The expected number of K-nodes is predictable. Next we will determine the expected number of in-order shifts required during the construction of an ISA tree. Finally we will investigate the cost for each in-order shift operation and determine theoretically the expected cost to construct an ISA tree.

Theorem 1. Let T be an ISA tree with n nodes, then the expected number of S-nodes (denoted by $\overline{S(n)}$), in the tree is given by the following recurrence relation:

$$\overline{S(n+1)} = \frac{n-2K-1}{n-2K+1}\ \overline{S(n)} + 1,$$

$$2^L < n < 2^{L+1}-1, \quad L \geq 1.$$

and

$$S(2^L) = 1.$$
$$S(2^{L+1}-1) = 0.$$

Proof. Define Pn(S,R,K) to be the probability that an ISA tree with n nodes will have S S-nodes, R R-nodes, and K K-nodes. The expected number of S-nodes in this tree is given by

$$\overline{S(n)} = \sum_{S,R,K} Pn(S,R,K)\ *S.$$

The probability the number of S-nodes decreases by one when a node is inserted into an ISA tree is $S/(2R+S)$, and the probability the number of S-nodes

increases by one is $2R/(2R+S)$. This follows directly from Lemmas 2, 3 and 6. Thus,

$$\begin{aligned}\overline{S(n+1)} &= \sum Pn(S,R,K) \\ &\quad [(S-1)*(S/(2R+S)) + (S+1) \\ &\quad *2R/(2R+S)] \\ &= \sum Pn(S,R,K) \\ &\quad [S^2 - S + 2R*S + 2R)/(2R+S)]\end{aligned}$$

Since $2R = n+1-2K-S$, we get the following:

$$\begin{aligned}&= \sum Pn(S,R,K) \\ &\quad \frac{[S^2-S+(n+1-2K-S)S+n+1-2K-S]}{n+1-2K} \\ &= \sum Pn(S,R,K) \\ &\quad \frac{[n-2K-1)*S+(n+1-2K)]}{n+1-2K}\end{aligned}$$

Thus

$$\overline{S(n+1)} = \frac{n-2K-1}{n-2K+1}\ \overline{S(n)} + 1$$

If $n = 2^L$ then the ISA tree is complete to level $L-1$ and only one node appears at level L. Thus this tree has $K = 1$ and necessarily $S = 1$, therefore $S(2^L) = 1$. Furthermore, if $n = 2^{L+1} - 1$ the ISA tree is complete to level L and necessarily has no S-nodes, thus $S(2^{L+1} - 1) = 0$.

When the first node is added to each new level of an ISA tree the number of S-nodes is exactly one. To determine the expected number of S-nodes as additional nodes are added to the level, the above recurrence formula can be used. Finally, as the last node is added to a level the number of S-nodes becomes zero. An equivalent formula for calculating $\overline{S(n)}$, is given

$$\overline{S(n)} = \sum_{i=0}^{n-2^L} ((2^L - 1) - (2*i))/(2^L - 1)$$

$$for\ 2^L \le n \le (2^L + 2^{L-1}) - 1.$$

Note that the sequence is symmetric about $n = (2^L + 2^{L-1}) - 1$.

Theorem 2. Let T be an ISA tree with n nodes, then the expected number of R-nodes (denoted by $\overline{R(n)}$), in the tree is given by the following recurrence relation:

$$\overline{R(n+1)} = \frac{n-2K-1}{n-2K+1}\ \overline{R(n)},$$

$$2^L < n < 2^{L+1} - 1, \quad L \ge 1.$$

and

$$R(2^L) = 2^{L-1} - 1$$

$$R(2^{L+1} - 1) = 0$$

Proof. Define $Pn(S,R,K)$ to be the probability of an ISA tree with n nodes has S S-nodes, R R-nodes, and K K-nodes. The expected number of R-nodes in this tree is given by

$$\overline{R(n)} = \sum_{S,R,K} Pn(S,R,K) * R.$$

When a node is inserted into an ISA tree, the probability the number of R-nodes decreases by one is $2R/(2R+S)$, and the probability the number of R-nodes remains unchanged is $S/(2R+S)$. This follows directly from Lemmas 2, 3, and 6.

Thus,

$$\begin{aligned}&\overline{R(n+1)} \\ &= \sum Pn(S,R,K)\ (R*(S/(2R+S)) \\ &\quad +(R-1)*2R/(2R+S)) \\ &= \sum Pn(S,R,K)\ ((R*S+2R^2-2R) \\ &\quad /(2R+S))\end{aligned}$$

Since $S = n+1-2K-2R$, we get the following:

$$\begin{aligned}&= \sum Pn(S,R,K)\ \frac{(R(n+1-2K-2R)+2R^2-2R)}{(2R+n+1-2R-2K)} \\ &= \sum Pn(S,R,K)\ \frac{(R*n+R-2K*R-2R^2+2R^2-2R)}{n+1-2K} \\ &= \sum Pn(S,R,K)\ \frac{(n-1-2K)}{(n+1-2K)} * R \\ &= \frac{n-2K-1}{n-2K+1}\ \overline{R(n)}\end{aligned}$$

If the $N = 2^L$ then the ISA tree is complete to level $L-1$ and only one node appears at level L. Thus this tree has $K = 1$ and $S = 1$ and from Lemma 1 $R = 2^{L-1} - 1$. Furthermore if $n = 2^{L+1} - 1$ the ISA tree is complete to level L and necessarily has no R-nodes, thus $R(2^{L+1} - 1) = 0$.

When the first node is added to each new level of an ISA tree the number of R-nodes is exactly, $R(2^L) = 2^{L-1} - 1$. To determine the expected number of R-nodes as additional nodes are added to the level, the

above recurrence formula can be used. Finally, as the last node is added to a level the number of R-nodes becomes zero. An equivalent formula for calculating $\overline{R(n)}$ on a given level is given by

$$\overline{R(n)} = (2^{(L-1)} - 1)$$

$$- \sum_{i=2^L}^{n-1} (2^{(L+1)} - i - 2)/(2^L - 1)$$

Consider the ISA tree with $n = 9$, and $K = 2$ in Figure 1. In this particular tree there are two S-nodes, and two R-nodes. However, it is possible with $K = 2$ to have no S-nodes, and three R-nodes. Intuitively the expected number of S-nodes and the expected number of R-nodes can be determined as follows. In this tree there are eight possible positions to place the two K nodes. This represents 28 combinations. In four of these cases both K-nodes have the same parent resulting in no S-nodes and three R-nodes. In the remaining 24 cases the two K-nodes have different parent nodes resulting in two S-nodes and two R-nodes. Therefore the expected number of S-nodes in an ISA tree with $n = 9$ nodes is calculated to be $(24*2+4*0)/28$, which is $12/7$. $\overline{S(9)}$ can also be determined by the above recurrence formula. In this case $n = 8$, $\overline{S(8)} = 1$, and $\overline{S(9)} = [(8 - 2 * 1 - 1)/(8 - 2 * 1 + 1)] * 1 + 1$, which is also $12/7$. Furthermore, the expected number of R-nodes in an ISA tree with $n = 9$ nodes is calculated to be $(24 * 2 + 4 * 3)/28$, which is $15/7$. $\overline{R(9)}$ can also be determined by the above recurrence formula. In this case $n = 8$, $\overline{R(8)} = 3$, and $\overline{R(9)} = [(8 - 2 * 1 - 1)/(8 - 2 * 1 + 1)] * 3$, which is $15/7$ as expected. Table I summarizes various values associated with ISA trees that are nearly complete to level four. In addition, Figure 2 gives the number of K-nodes, and the expected number of S-nodes and R-nodes in graphical form for the same ISA trees depicted in Table I.

Lemma 7. Let T be an ISA tree with n nodes. The probability that the ISA algorithm will be required when a node is inserted into the tree is $2K/(n + 1)$.

Proof. An insertion of a node into an ISA tree with n nodes can occur in any of the $n + 1$ E-nodes. The only insertions that require the ISA algorithm are insertions that fall in the external nodes of a K-node, and each K-node has two external nodes (Lemma 4). Therefore assuming the equal likelihood of each key, the probability that the ISA algorithm will be required when inserting a node into and ISA tree with n nodes is $2K/(n + 1)$. Consider Figure 1 with $n = 9$. The probability that the tenth node inserted into this tree will require the ISA algorithm is clearly $4/10$.

Theorem 3. Let T be an ISA tree nearly complete to level L with n nodes, then the expected number of ISA (in-order shifts) required to construct the tree is given by:

$$\left(\sum_{i=0}^{L-1} \sum_{j=1}^{2^i-1} \frac{2j}{2^i + j} \right) + \sum_{j=1}^{K-1} \frac{2j}{2^L + j}$$

Proof. An ISA tree complete to level $L - 1$ with n nodes has $n - (2^L - 1)$ K-nodes (Definition 2). The expected number of times the ISA algorithm will be performed in the construction of an n node ISA tree, denoted as ISA(n), is the sum of the probabilities that the ISA algorithm is required for each insertion of nodes $1..n$ as the tree is constructed. The probability that the ISA algorithm is required for each new insertion is given in Lemma 7 as $2K/(n + 1)$.

Consider the expected number of in-order shifts required for inserting nodes in a given level of an ISA tree, say level m. There are 2^m nodes to insert on level m and the number of K-nodes range from 1 to 2^m. Furthermore, the number of nodes of the tree, n, range from 2^m to $2^{m+1} - 1$. The expected number of in-order shifts required to insert the nodes on level m is given by

$$\sum_{K=1}^{2^m-1} \frac{2K}{2^m + K}$$

Finally, the expected number of in-order shifts for an ISA tree complete to level $L-1$ with K K-nodes on the partially completed level L is given by the following formula, which is the desired result.

$$\left(\sum_{m=0}^{L-1} \sum_{K=1}^{2^m-1} \frac{2K}{2^m + K} \right) + \sum_{j=1}^{K-1} \frac{2j}{2^L + j}$$

3.2 Imperical Cost Analysis

The expected number of times the ISA algorithm is required in the construction of an ISA tree with n nodes is fairly easy to calculate as shown above. However, the expected cost of the construction of an ISA tree is much more difficult to analyze. We define the cost of the ISA algorithm as the minimum number of nodes that change values during the in-order shift. In most cases an in-order shift can be performed either to the right or the left of the insertion point, and we will assume the

minimum cost shift will always be used. Furthermore, we define cost as the number of nodes that change value rather than the number of nodes along the path of the in-order shift. In this case we assume the tree has a bidirectional thread connecting the nodes in an in-order fashion. Note the actual work needed to create a new node and locate its position in the tree is not included in the cost (ie., only the cost of the in-order shifting itself is considered in this analysis).

Consider the average cost computation (expected cost) going from an ISA tree of size n to size $n+1$. It is obvious the expected cost for inserting the first node on any given level is zero. Thus we will consider cost as it pertains to a given level, such as level 3 in Figure 1. To compute the expected cost for going from a tree of size 8 to a tree of size 9, we have to consider the eight possible ISA trees for $n = 8$. Furthermore for these 8 possible trees one must consider all of the possible placements of one additional node, and the actual cost of any insertions that require the ISA algorithm.

We have devised an enumeration scheme to uniquely identify all possible ISA trees to assist in cost analysis. Given an ISA tree with n nodes, and K K-nodes on the lowest level we represent the unique shape of the ISA tree by a series of integers that partitions the tree by indicating the positions of each K-node and empty position as they appear on the level from left to right. The key to this notation is that the number of digits in the sequence is the number of partitions (about empty locations) and is always one plus the number of empty positions on the lowest level of the tree. Each non-zero integer indicates the number of consecutive K-nodes in a row. Thus the sum of the digits always adds to the number of K-nodes. Furthermore, two consecutive non-zero integers implies an empty position (partition) between them, else a single digit would be used. Also a zero represents a partition and indicates the presents of an E-node. For example the configuration of Figure 1 is uniquely represented by 0100100. If instead both K-nodes were located in the leftmost two positions, then the representation would be 2000000. A 3-partition sequence of 321 uniquely defines an ISA tree with $n = 13$ and on the last level the 8 positions left to right are $KKKEKKEK$-nodes. We know this because a 3-digit sequence represents two empty positions on the last level, and the sum of the digits represents six K-nodes. Thus the level of concern on this ISA tree has 8 locations, implying a total of 7 nodes on complete levels higher in the tree. The reader is encouraged to verify that the sequence 10400301 represents uniquely an ISA tree with $n = 24$, and the lowest level of the tree left to right has $KEEKKKKEEEKKKEEK$-nodes.

There are two types of in-order insertions: those that must be performed in one direction called End-insertions, and those that have the option of going either to the left or to the right of the insertion location, called Mid-insertions. The cost associated with each of these types of inserts are denoted by Costend and Costmid, respectively. End-insertions occur at K-nodes that appear at either end of the level. Mid-insertions occur elsewhere in the level. The cost of an ISA algorithm for Costends are $2k(2k+1)/2$, and the cost of an ISA algorithm for Costmids are $k\ (k+1)$, where k represents the number of consecutive K-nodes in the partition. This is the number of nodes that change values during the in-order shifting. Consider again the expected cost of going from an ISA tree with 8 nodes to 9 nodes. The eight possible ISA trees with $n = 8$ are represented by the partition sequences: 10000000, 01000000, 00100000, ..., 00000001. The first and last of these sequences each have Costends of $2*(2+1)/2$. The other six partitions each have a Midcost of $1*(1+1)$. The total sum of the possible costs is 18. Most insertions in this example cost nothing, however. Thus, the expected or average cost is the total cost divided by the product of the number of possible configurations times the number of external nodes. Thus the expected cost associated with the ISA algorithm in going from an ISA tree with 8 nodes to a tree with 9 nodes is $18/(8$ configurations $*9E$-nodes$) = 0.25$.

4. CONCLUSIONS

Binary search trees with minimal internal path length offer optimal search times assuming equal likelihood for searches. We have defined such trees as ISA trees, which use an in-order shift algorithm to maintain a balanced tree. We used fringe analysis to imperically determine the cost of construction and maintainence of ISA trees. As a summary, Table I depicts the number of K-nodes, the expected number of S-nodes and R-nodes, the expected number of in-order shifts, and the expected cost required in the construction of ISA trees for $n = 15$ to 31. Figure 3 graphically illustrates for $n < 32$ the incremental expected number of in-order shifts required in going from $n-1$ to n nodes. Figure 3 also illustrates the accumulative (total) expected number of in-order shifts required for an ISA tree with n nodes. The incremental expected number of in-order shifts ranges from [0..1). Notice the increment is nearly linear for each new level of the tree.

Figure 4 illustrates the incremental expected cost in going from $n-1$ to n nodes in an ISA tree, for $n < 32$. As expected the cost is zero for the first node on each

level, and the expected cost increases slowly until the level begins to fill up. The expected cost exhibits exponential behavior as the level becomes complete. However, the expected cost behaves in a near linear fashion for the first half of the nodes inserted on each level. We have noticed that the expected cost of inserting the last $L-1$ nodes on a given level L is as much or more than inserting all of the other nodes on the level combined. For example in Table I, where $L = 4$, inserting the last three nodes ($n = 29, 30, 31$) costs as much or more than inserting nodes 16 through 28. Furthermore the expected cost of inserting the last node on each level appears to be nearly linear. The expected cost for the last insertion on a given level can be estimated by the linear relationship: $\overline{cost(n)} = (n-7)/3 + 2.43$. See $n = 7, 15$, and 31 in Figure 4.

Even though the maintainence of an ISA tree can be quite expensive in certain instances, most of the time it is relatively inexpensive. The cost can easily be justified if searching is the predominant activity, and inserting is relatively infrequent in comparison. As a topic of further research in this area, we are looking into the feasability of relaxing the requirement of inserting all nodes on one level before going to the next. Since inserting the last $L-1$ nodes on each level, L, is extremely expensive, we are studying the effects of not inserting these nodes until the expense is within some acceptable range.

REFERENCES

[1] M.R. Brown, A Partial Analysis of Random Height Balanced Trees, SIAM J. Comput. 8 (1) (1979) 33-41.

[2] H. Chang and S.S. Iyengar, Efficient Algorithms to Globally Balance a Binary Search Tree, Commun. ACM 27 (7) (1984) 695-702.

[3] T.E. Gerasch, An Insertion Algorithm for a Minimal Internal Path Length Binary Search Tree, Commun. ACM 31 (5) (1988) 579-585.

[4] G.H. Gonnet, Balancing Binary Trees by Internal Path Reduction, Commun. ACM 26 (12) (1983) 1074-1081.

[5] D.E. Knuth, The Art of Computer Programming, vol. 3, Sorting and Searching (Addison-Wesley 1973).

[6] J. Nievergelt and E.M. Reingold, Binary Search Trees of Bounded Balance, Siam J. Comput. 2 (1) (1973) 33-43.

[7] Q.F. Stout and B.L. Warren, Tree Rebalancing in Optimal Time and Space, Commun. ACM 29 (9) (1986) 902-908.

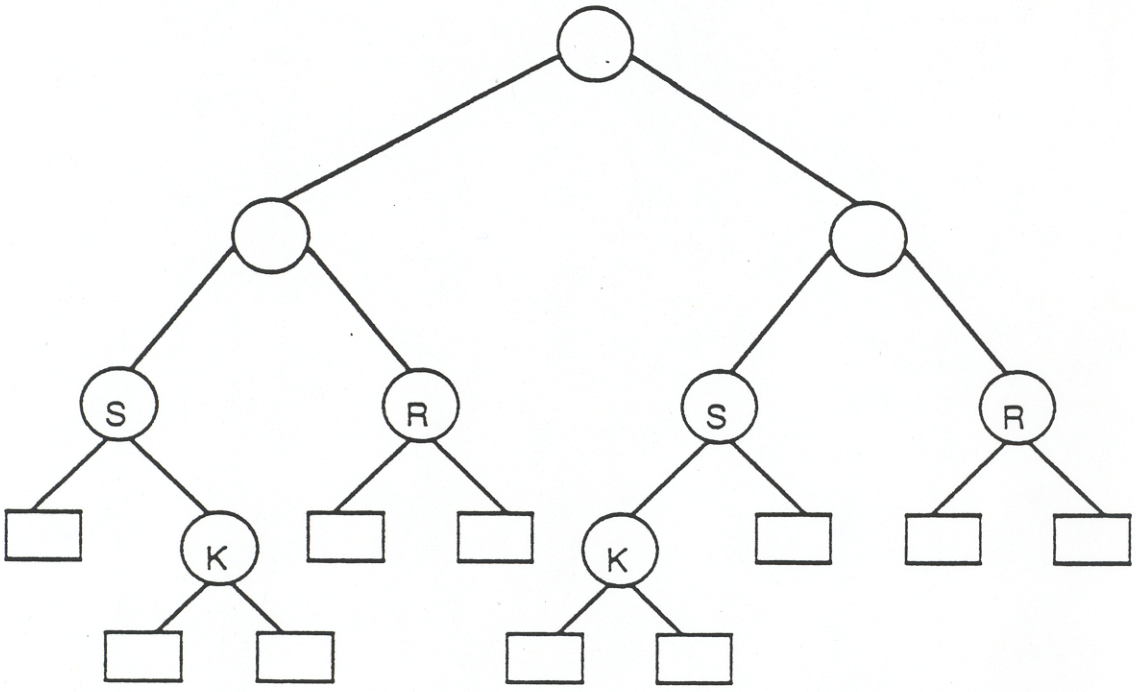

Figure 1. A Sample ISA Tree with 9 nodes.

Table I
Various Expected Values Associated with ISA Trees

n	K	S(n)	R(n)	ISA(n) Increment	ISA(n) Total	Cost(n) Increment	Cost(n) Total
15	8	0	0	0.93	6.98	5.13	17.93
16	1	1	7	0	6.98	0	17.93
17	2	28/15	91/15	0.12	7.10	0.13	18.05
18	3	39/15	78/15	0.22	7.32	0.25	18.31
19	4	48/15	66/15	0.32	7.64	0.39	18.69
20	5	55/15	55/15	0.40	8.04	0.53	19.22
21	6	60/15	45/15	0.48	8.52	0.69	19.91
22	7	63/15	36/15	0.55	9.06	0.86	20.77
23	8	64/15	28/15	0.61	9.67	1.07	21.84
24	9	63/15	21/15	0.67	10.34	1.31	23.15
25	10	60/15	1	0.72	11.06	1.61	24.76
26	11	55/15	10/15	0.77	11.83	1.99	26.75
27	12	48/15	6/15	0.82	12.64	2.50	29.26
28	13	39/15	3/15	0.86	13.50	3.23	32.49
29	14	28/15	1/15	0.90	14.40	4.35	36.83
30	15	1	0	0.93	15.33	6.30	43.13
31	16	0	0	0.97	16.30	10.48	53.61

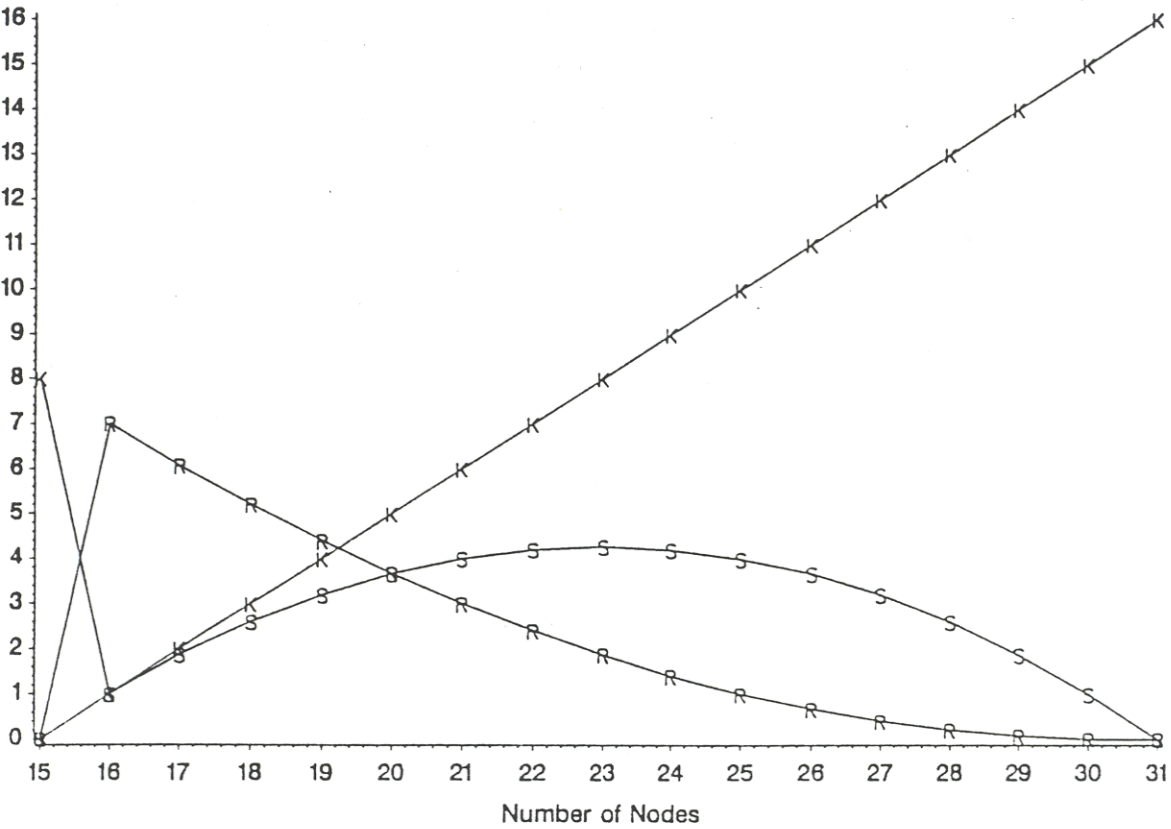

Fig. 2 Expected Number of S, R, and K – nodes

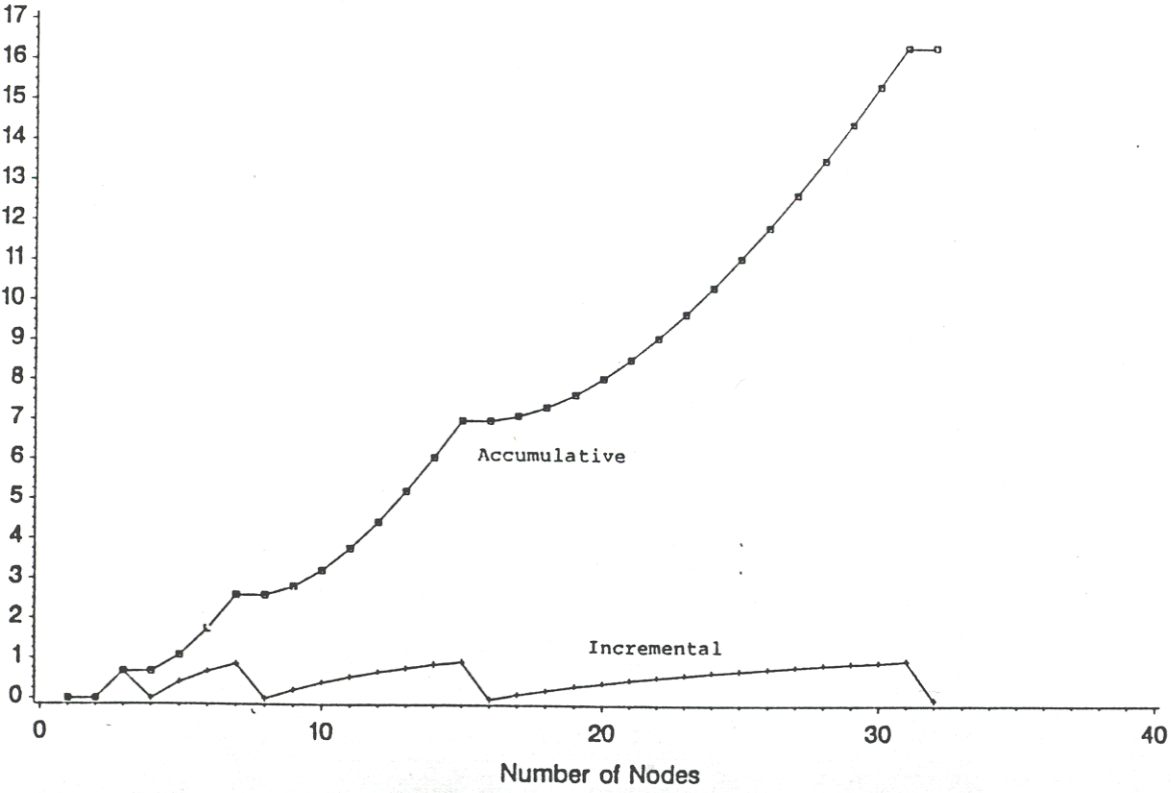

Fig. 3 Incremental and Accumulative Expected Number of In – Order Shifts

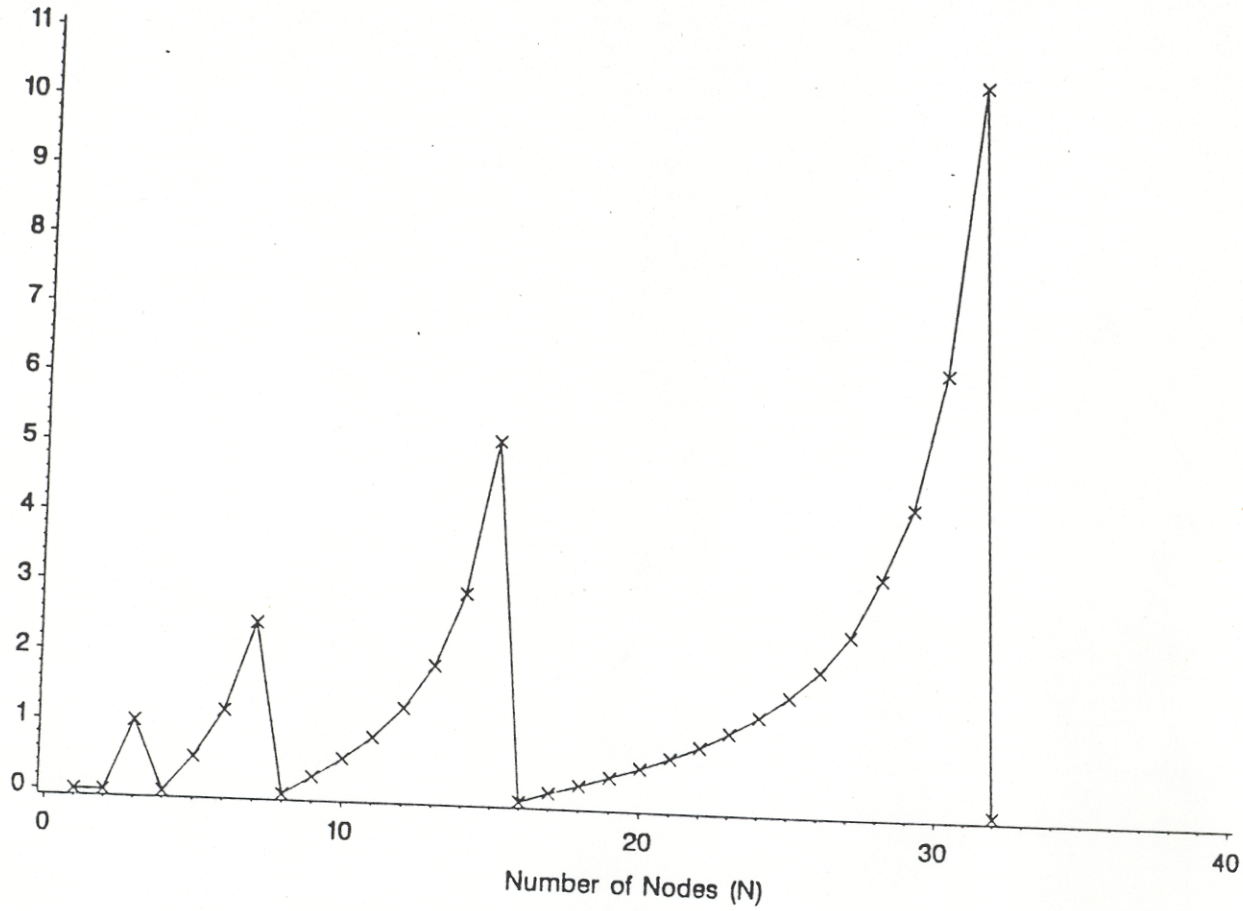

Fig. 4 Incremental Expected Cost from N−1 to N Nodes