1976

Analysis of a Heuristic for Full Trie Minimization

Douglas E. Comer
Purdue University, comer@cs.purdue.edu

Report Number:
77-217

ANALYSIS OF A HEURISTIC FOR FULL TRIE MINIMIZATION

Douglas Comer
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

December 1976

CSD-TR 217

Analysis of a Heuristic for Full Trie Minimization

Douglas Comer
Purdue University
W. Lafayette, Indiana

ABSTRACT:

A trie is a distributed-key search tree in which records from a file correspond to leaves in the tree. Retrieval consists of following a path from the root to a leaf, where the choice of an edge at each node is determined by attribute values of the key. For full tries, those in which all leaves lie at the same depth, the problem of finding an ordering of attributes which yields a minimum size trie is NP-Complete. Since the problem has practical implications, heuristic solutions are of interest. We consider a "greedy" heuristic and derive a worst case bound on the approximation it produces. A file is given for which the heuristic performs badly, producing tries of high cost.

Keywords and Phrases: trie index, trie size, heuristic procedure

CR Categories: 3.74, 4.35, 4.34, 5.25

Author's address: Computer Science Department, Purdue University, W. Lafayette, IN 47907

1: Introduction

A trie defined by Fredkin (1960), is an implementation of a distributed-key search tree in which records from a file correspond to leaves in the tree. Retrieval is carried out by following a path from the root of the tree to a leaf, the choice of a new edge at each node being determined by attribute values of the key. If all records in the file have the same number of attributes, then each path in the trie will be of the same length, and all leaves will lie at the same depth. This is called a full trie and has the property that the size of the trie is determined by the order in which attributes are tested.

Formal definitions of a trie are given in Comer and Sethi (1977). In the same paper, the problem of finding an ordering of the attributes which produces the minimum size trie is shown to be difficult in a precise sense. More formally, the problem is shown to be NP-Complete.[1] Since, at present, there is no known efficient algorithm for problems in this class, optimal solutions take exponential time. Even for a small file, such solutions are often too expensive to be feasible. Yet the problem of trie minimization is of practical interest. Rotwitt and deMaine (1973) and Yao (1976) consider an alternative: procedures which are computationally efficient but which yield solutions which are "close" to the optimal in some sense. Such procedures are often derived from "rule of thumb" practices and are called heuristics.

The focus of this paper will be on the analysis of tries built by heuristics. We seek a worst case bound on the approximation that the

[1] Aho et al (1973) provides a reasonable introduction to NP-Complete problems.

given heuristic produces. Such a bound provides an absolute limit on the size of the tries produced, but warns that they could be that bad.

To measure the performance of a heuristic, let the size of a trie produced by the heuristic be S_h. Let S_o and S_w denote the sizes of an optimal (smallest) and worst (largest) trie, respectively. The cost criterion used to judge the heuristic will be

$$COST = S_h/S_o$$

Heuristics which have minimum cost are desirable. Although this cost does not include the computational time of the heuristic itself, we assume that it is the sole criterion for judging the performance. Only efficient procedures, those for which the running time is a low-degree polynomial in the size of the file, will be considered and differences in the amount of work required between any two heuristic procedures will be considered insignificant.

Heuristics for full trie minimization are intended to produce low cost tries by minimizing the breadth of the trie. Figure 1.1 shows best and worst possible tries for a file of r records and k attributes. Intuitively, if a trie has fewer nodes than the worst case, it must have fewer nodes at depths near the root. The best case has fewer nodes at all depths except the depth of the leaves.

One way to produce a small trie, then, is to choose attributes in an order which minimizes the number of nodes at each depth. This optimizes the trie locally by restricting growth on a level-by-level basis. Of course, this does not guarantee the minimum size trie; it is only an attempt to do so. The idea of local minimization is formalized in the following.

DEFINITION 1 The GREEDY HEURISTIC for full trie minimization is given by the following procedure. While building the trie, select at each depth an attribute which adds the smallest number of nodes to the next depth. □

Note that the GREEDY heuristic requires at most $O(rk^2)$ to compute and therefore, meets the criteria for an efficient procedure defined above.

We will characterize the best and worst case tries, showing the maximum improvement that can be expected from any heuristic. The GREEDY heuristic will then be examined to see how it performs. One might expect that this heuristic provides a good approximation to minimum tries. We will show, however, that it can produce high cost tries under certain circumstances.

The rest of this paper is organized as follows: in section 2 a "restricted file" will be defined. These files have no trivial or isomorphic attributes and will be considered when analyzing heuristics. Following this (r,k)-FAT and (r,k)-THIN trees will be defined in sections 2 and 3. It will be shown that they are the largest and smallest tries, respectively, indexing a binary restricted file. Thus, the ratio of the size of an (r,k)-FAT tree to an (r,k)-THIN tree derived in section 4 is a bound on the performance of any heuristic. A modified (r,k)-FAT tree will be defined in section 5 which can be produced by the GREEDY heuristic. It will be shown that the worst case approximation for the GREEDY heuristic is not bounded by a constant but grows as the number of attributes, k.

2: A Worst case Trie

The smallest and largest full tries for a file of r records and k attributes are shown in Figure 1.1. The best trie has k internal nodes while the worst trie has r(k-1)+1 internal nodes. The ratio of sizes of worst to best, S_w/S_o is:

$$S_w/S_o = r((k-1)/k) + 1/k$$

which results in a factor of r for most k.

Files which allow tries as small as k nodes are not realistic because they have k-1 attributes which contain no information. Since we seek to model the files one might encounter, let us rule out trivial attributes -- those which carry no information; and isomorphic attributes -- those which are duplicates. Files which do not have these two types of attributes will be called restricted. In the rest of this paper, the term "file" will mean restricted file. We wish to know the ratio S_w/S_o for full tries indexing restricted files.

In the analysis which follows we will characterize smallest and largest tries indexing a binary restricted file. Attention to the binary case is motivated by two reasons. On one hand, since information is represented in binary in most computers, one can view operations on a binary file as operations on the binary encoding of a more general case. On the other hand, it is desirable to obtain information about this simple case as a prelude to understanding files of higher degree such as ternary. Recall that if one attribute has a ternary value set while all others are binary, then the file is of degree 3.

The following simple property of binary restricted files will be used extensively.

LEMMA 1 Let F be a binary restricted file of r records and k attributes such that there exists a full trie indexing F. Then

$$\lceil \log_2 r \rceil \leq k < 2^{r-1} \qquad (1)$$

PROOF: $(k \geq \lceil \log_2 r \rceil)$

Suppose $k < \lceil \log_2 r \rceil$. Let t be an integer such that $2^{t-1} < r \leq 2^t$. Then $\lceil \log_2 r \rceil = t$. Recall that a binary tree with depth k can have at most 2^k leaves. So any trie indexing F can have at most 2^{t-1} leaves. This is a contradiction since $r > 2^{t-1}$ and therefore $k \geq \log_2 r$.

$(k < 2^{r-1})$

Suppose $k \geq 2^{r-1}$. Think of the binary values in each column as a binary number of r bits. Clearly, one half of these are isomorphic up to a renaming of bits. Of the 2^{r-1} remaining values, one of them is all zeroes (or all ones). Therefore there are $2^{r-1} - 1$ nonisomorphic, nontrivial values. Since by assumption $k \geq 2^{r-1}$, at least one of the values must be repeated. This is a contradiction and it must hold that $k < 2^{r-1}$. ∎

Since two positive integers r and k which satisfy equation (1) appear so often throughout the paper, this condition is given a name as follows:

DEFINITION 2 A pair of integers (r,k) is <u>valid</u> iff:

1. r, k > 0, and
2. $\lceil \log_2 r \rceil \leq k < 2^{r-1}$

An (r,k)-File is a valid file iff (r,k) is a valid pair of integers and the file has r records and k attributes. □

We have commented on the tries shown in Figure 1.1 which are the best and worst tries for an unrestricted file. We will now consider worst case tries for binary restricted files. For a binary file, trees of the shape in Figure 1.1 are prohibited since a node may have at most two descendants. The point to note is that the worst case trie for an unrestricted file distinguishes the records as early as possible. An early splitting will also occur in a worst case trie for a binary file but will be slightly slower due to the binary constraint. Consider the trie shown in Figure 2.1 for a file of eight records and seven attributes. The first three depths form a complete binary tree, distinguishing all records as fast as possible. The remaining levels contain only chains as in the worst trie for an unrestricted file. Of course, this example is for a tree with the number of leaves a power of two. Tries with this shape are defined for arbitrary number of leaves in the following definition.

DEFINITION 3 Let (r,k) be valid integers and let t be an integer such that $2^t < r \leq 2^{t+1}$. Then an (r,k)-FAT tree is a binary tree such that:

1. Each node at depth d, $0 \leq d < t$, has two sons.
2. $r - 2^t$ nodes at depth t have two sons and the remaining $2^{t+1} - r$ nodes have one son.
3. Each node at depth d, $t+1 \leq d \leq k-1$, has exactly one son. □

The following lemma shows that an (r,k)-FAT tree is as large as any trie indexing a binary restricted file of r records and k attributes.

LEMMA 2 Let F be a valid (r,k) binary restricted file and let T be a full trie indexing F. If A is an (r,k)-FAT tree then

$$|A| \geq |T|$$

where $|T|$ denotes the size of tree T.

PROOF:

Suppose that $|T| > |A|$. Since both trees have all leaves at the same depth, there must be a first depth, d, at which T has more nodes than A. Let t be an integer such that $2^t < r \leq 2^{t+1}$. Two cases arise.

Case 1: $d < t$. Since each node in A at depth less than t has two sons, T cannot have more nodes than A and still be a binary tree.

Now consider case 2.

Case 2: $d \geq t$. By the definition of A there are r nodes at depth t and each one has one son for all depths d, $t < d \leq k-1$. Therefore, if T has more nodes than A, it must have more than r leaves. This is a contradiction, and the Lemma holds. □

3: Smallest Tries for Binary Restricted Files

The following defines a binary tree called an (r,k)-THIN tree. First, an i-STEM is defined which is the slowest growing trie for a binary file with no isomorphic or trivial attributes. Following this an (r,k) THIN tree will be defined as an i-STEM with a FOREST of binary trees rooted in its n leaves. Finally, it will be shown that the ratio S_w/S_o is attained by an (r,k)-FAT tree and an (r,k)-THIN tree.

Consider a binary restricted file. We wish to characterize the most slowly growing full trie for such a file. Since there are no duplicate or isomorphic attributes, only a finite number of attributes may be selected before a new node is added to the trie. A minimum growth trie for the file shown in Figure 3.1a will have a shape as shown in Figure 3.1b. This trie shows an exponentially increasing number of levels between the appearance of a new node.

The following gives a formal definition of a tree with this shape which will be shown to meet the lower bound on the growth of a full trie for a binary restricted file.

DEFINITION 4 Let i be a nonnegative integer, and let t be an integer such that $2^t \leq i < 2^{t+1}$. An i-STEM is a binary tree such that:

1. All leaves lie at depth i.
2. The rightmost node at depth 2^j-1, $0 \leq j \leq t$, has two sons and all other nonleaf nodes have exactly one son. □

Examples of i - STEMS are shown in Figure 3.2.

To see how we arrive at the shape of an i-STEM, consider a binary restricted file. The first selection of an attribute from the file must split it into two parts. The second attribute tested must break at least one of these groups into two or it would be isomorphic with the first. But the third selection could be such that it did not further divide the sets of records and yet was not isomorphic to the first two attributes. Figure 3.3 shows a sample file. Testing left-to-right distinguishes the first set at depth one, the second set at depth two, and no set at depth three. Following this at least one set must be distinguished at depth four.

Using these ideas we will show that an i-STEM is as small as the slowest growing trie indexing a binary restricted file.

LEMMA 3 Let F be a valid (r,k) binary restricted file and let T be a full trie indexing F. Then T must have at least $\lfloor \log_2 i \rfloor + 2$ nodes at depth i.

PROOF:

Claim: If A is a k-STEM then T has at least as many nodes as A at depths d, $0 \leq d \leq k$.

PROOF of Claim: From the above discussion, there is only one depth possible in T with no new nodes after the second record is distinguished. Now assume that there are $2^{j-1}-1$ depths possible with no new nodes after the j^{th} record has been distinguished, for $2 \leq j \leq p$. Suppose that the p^{th} record is distinguished. Think of the assignment of values to records as assigning bit values to a p-bit binary number. There are only 2^p possible assignments, and one-half of these were used after the $p\text{-}1^{st}$ record was distinguished. Therefore, only 2^{p-1} assignments can be made before a duplication occurs, and only 2^{p-1} additional levels can appear in the trie before another record must be distinguished. Therefore, T must have an attribute tested which distinguishes the $p+1^{st}$ record at depth 2^{p-1}. Since A meets this criterion, the size of A is less than or equal to the size of T. □Claim.

Since the growth of A is a lower bound on the growth of any full trie, T, the number of nodes at depth i in A is a lower bound on the number of nodes in T at that depth. We will show that A has $\lfloor \log_2 i \rfloor + 2$ nodes at depth i.

For depths one, two, and three A has two, three, and three nodes, respectively. Let t be a positive integer such that $2^t \leq i < 2^{t+1}$ and assume that for all j, $0 \leq j < t$, A has $\lfloor \log_2 j \rfloor + 2$ nodes at depths 2^j to $2^{j+1}-1$. Consider i in the range 2^t to $2^{t+1}-1$. Since only one node at depth 2^t-1 had an additional son, there must be $(\lfloor \log_2 2^t-1 \rfloor +2)+1$ nodes at each depth. But

$$\lfloor \log_2 2^t-1 \rfloor +3 = t-1+3 = t+2 = \lfloor \log_2 i \rfloor +2.$$

Therefore, the lemma holds by induction. □

Since an i-STEM is the most slowly growing full trie, one might think that it would be a minimum size tree of $\lfloor \log_2 i \rfloor +2$ leaves, each at depth i. To see that this is not the case, consider the trees shown in Figure 3.4. The second tree consists of an i-STEM which is shorter than the first and each node just before a leaf has two sons which are leaves. Extending this one more level would yield an even shorter tree. A minimum size full trie will be characterized which uses this idea of continuing an i-STEM with lower levels which are complete binary subtrees.

Consider an i-STEM in which the leaves have been made the roots of a forest of binary trees. At each depth there must be at least $n(i) = \lfloor \log_2 i \rfloor + 2$ nodes. If there are to be r leaves at depth k, where $r > n(k)$, then there must be some depth p at which the tree begins to grow more rapidly than an i-STEM. Since we wish to delay this splitting as long as possible, p is to be maximized. To see how the best value for p is obtained, observe that in a binary tree of q leaves at least $\lceil \log_2 q \rceil$ depths are required to distinguish all q leaves. In the case of an i-STEM, the r leaves can be divided into a forest, F, of n(i) binary trees. Since

all trees in the forest are of the same depth, we will need $d = \max_{t \in F} \lceil \log_2(\text{leaves in } t) \rceil$ depths to distinguish all leaves. It is easily seen that d can be minimized by distributing the leaves as evenly as possible among all trees in the forest. The largest tree will have $\lceil r/n(i) \rceil$ leaves. Thus, at depth i in the i-STEM, the number of additional depths needed to distinguish all r leaves is $\lceil \log_2 \lceil r/n(i) \rceil \rceil = \lceil \log_2 (r/n(i)) \rceil$.

An (r,k)-THIN tree will be defined in terms of an i-STEM in which the leaves form the roots of a forest of binary trees. It will then be shown that the (r,k)-THIN tree is minimized when all trees in the forest are complete binary trees of equal size.

DEFINITION 5: Let r, k be a valid pair of integers and let n(i) be defined by $n(i) = \lfloor \log_2 i \rfloor + 2$. Let p be the maximum integer such that

$$p + \lceil \log_2 (r/n(p)) \rceil = k$$

Then an <u>(r,k)-THIN tree</u> is a binary tree which consists of a p-STEM in which the n(p) leaves form the roots of a forest of n(p) binary trees such that the largest binary tree has $\lceil r/n(p) \rceil$ leaves. □

Note that the (r,k)-THIN tree defined here is not minimum for arbitrary r and k since the exact shape of the binary trees in the forest is not specified. While a minimum size forest can be characterized (see Comer 1976), we are interested only in a bound on S_w/S_o. The following lemma shows that it is sufficient to consider only those cases where the trees in the forest of the (r,k)-THIN tree are all complete binary trees of the same size.

LEMMA 4: Let (r,k) be a valid pair of integers, and let T and F be an (r,k)-THIN and (r,k)-FAT tree, respectively. Then the ratio $|T|/|F|$ is minimized when all trees in the forest of the (r,k)-THIN tree are complete binary trees of equal size.

PROOF:

Suppose they are not complete binary trees. Let p be the depth in T at which the roots of the forest lie. Then for large enough k, F will have r nodes at depth p. Now consider a new pair of trees, T' and F' which are (r+1,k)-THIN and (r+1,k)-FAT trees, respectively. T' will still have the same number of nodes at depth p as T because the trees in the forest were not complete binary trees. But F' will have r+1 nodes at depth p. Thus the ratios of the sizes of T' to F' is $(|T|+q)/(|F|+q+1)$ which is smaller than $|T|/|F|$. This is a contradiction and the trees must all be complete binary trees.

Now suppose that the forest has two trees of different sizes. Since the same argument implies that $|T|/|F|$ is not minimum, the Lemma holds. □

THEOREM 1: Let (r,k) be a valid pair of integers and let F be a binary restricted file with r records and k attributes. Let T be an (r,k)-THIN tree in which the trees of the forest are all complete binary trees of the same size, and let A be a full trie indexing F. Then

$$|T| \leq |A|$$

where $|T|$ denotes the size of T.

PROOF: Suppose $|A|<|T|$. Since A and T both have all leaves at depth k, there must be a first depth, d, such that A has fewer nodes at depth d than T. Let p be the depth of the roots of the forest of binary trees in T. Now two cases arise:

Case 1: $d \leq p$. From Lemma 3, a p-STEM is the slowest growing trie for a binary restricted file. Therefore, A cannot have fewer nodes than T at depth d.

Case 2: d>p. Since T has complete binary trees rooted at depth p, if A has fewer nodes at depth d, then A would have fewer leaves than T. But this is a contradiction.

Therefore, the assumption was false and $|T| \leq |A|$. □

4: A Bound on the Ratio S_w/S_o for Binary Restricted Files

In this section a bound on the ratio of the size of an (r,k)-FAT tree, S_w, to an (r,k)-THIN tree, S_o, will be derived. This worst case bound will provide a measure of the maximum improvement that can be expected from any heuristic for tries indexing a binary restricted file. The bound will be computed for THIN trees in which the forest consists of complete binary trees of equal size. This will produce the worst case according to Lemma 4. To demonstrate that this bound is achievable, a file will be given in section 5 for which there exist tries approaching it asymptotically.

The size of an (r,k)-THIN tree can be obtained from the sum of the size of the i-STEM and the forest. The size of an i-STEM of n leaves (including the leaves) where $i = 2^{p-1}$ for some p can be obtained by summing the nodes at each level. This is

$$1 + 2*2^0 + 3*2^1 + 4*2^2 + \ldots + n*2^{n-2}$$

$$= 1 + \sum_{i=2}^{n} i*2^{i-2} = (1/2) + (1/4) * (n*2^{n+2} - (n+1)2^{n+1} + 2)$$

Simplifying, we get

$$(1/2)*((n-1)\ 2^n+1)+1/2 = (n-1)2^{n-1}+1$$

The size of a forest of n (complete) binary trees of f leaves each is n(f-1)-n (excluding the n roots and nf leaves). Since f = r/n, the size of the forest in an (r,k)-THIN tree is n(r/n-2) = r-2n. The size of an (r,k)-THIN tree, S_o, is then

$$S_o = (n-1)2^{n-1} + 1 + r - 2n \qquad (2)$$

where n is the number of leaves of the i-STEM. To relate this to r and k, observe that $r = n2^t$ for some integer t. t is the height of the trees in the forest. From Lemma 3, an i-STEM has $\lfloor \log_2 i \rfloor + 2 = \lfloor \log_2 4i \rfloor$ leaves. Thus, it must be true that $i \leq 2^{n-1}-1$. Therefore, $k = t + i = t + 2^{n-1} - 1$.

The size of an (r,k)-FAT tree (excluding the leaves) can be easily computed since it consists of a complete binary tree of $\lfloor \log_2 r \rfloor$ levels. Following this there are exactly r nodes at each of the remaining $k - \lfloor \log_2 r \rfloor - 1$ depths. Let $p = \lfloor \log_2 r \rfloor$ then the size of an (r,k)-FAT tree, S_w, is

$$S_w = 2^{p+1} - 1 + r(k - 1 - p)$$

From the discussion of (r,k)-THIN trees, there are integers t and n such that $r = n2^t$ so

$$\begin{aligned} S_w &= 2^{t+1+\lfloor \log n \rfloor} - 1 + n2^t(2^{n-1} - 2 - \lfloor \log_2 n \rfloor) \\ &= 2^t (2n - 1 + n(2^{n-1} - 2 - \lfloor \log_2 n \rfloor)) \\ &= 2^t (n(2^{n-1} - \lfloor \log_2 n \rfloor) - 1) \qquad (3) \end{aligned}$$

From equations (2) and (3), we can deduce a bound on the worst case performance of any heuristic. Since $2^{n-1} >> \log_2 n$,

$$S_w \leq 2^t(n2^{n-1} - 1) = r2^{n-1} - 2^t$$

$$= r(2^{n-1} - r/n) = r(2^{n-1} - 1/n) \leq r2^{n-1}$$

therefore,

$$S_w/S_o \leq (r2^{n-1})/((n-1)2^{n-1}+r-2n+1) \qquad (4)$$

This ratio is approximately $r/(n-1)$ for large r and n. Since $r/(n-1) \geq 2^t$, S_w/S_o is not bounded above by a constant, but grows as the size of the input file.

5: A Worst Case File for the GREEDY Heuristic

Consider a file of the form shown in Figure 5.1. These files represent a class of binary restricted files for which there exists a worst case tric. In addition, the GREEDY heuristic misbehaves when presented with a file from this class. To help relate these files to our previous analysis, we will use the parameters n and t as shown and refer to them as (n,t)-WC files. It should be clear that $r = n2^t$ and $k = t + 2^{n-1} - 1$.

First it will be shown that tries exist for an (n,t)-WC file which are (r,k)-THIN trees and (1,k)-FAT trees. Then the performance of the heuristic on this file will be analyzed.

LEMMA 5. Given n,t > 1, and F an (n,t)-WC file, there exists full tries T and A indexing F which are an (r,k)-THIN and (r,k)-FAT tree, respectively.

PROOF: (for T)

Construct T in the following way. By definition of F, all 2^t blocks of n records are identical. Select the 2^{n-1}-1 attributes left to right from the set Q yielding a $(2^{n-1}-1)$-STEM where each leaf in the stem represents a set of 2^t records, one from the first block, one from the second block, and so on. Select the final t attributes from N

left-to-right dividing up these sets, doubling the number of divisions at each depth. Thus, T consists of n subtrees of 2^t leaves, each of which is a complete binary tree, rooted in the n leaves of a $(2^{n-1}-1)$-STEM. By definition, T is an (r,k)-THIN tree.

(for A)

Construct A in the following way. Choose the t attributes from set P left-to-right yielding a complete binary tree of depth t. Associated with each leaf in this part of the tree will be n records, exactly one for each of the n blocks. Since these blocks are all identical and contain all $2^{n-1}-1$ possible attribute values, select in order attributes which divide the set in half, then in quarters, and so on. This will place a complete binary subtree at each of the 2^t nodes formed by selections in Q. Following this, all records will be distinguished and the remaining depths will have r nodes each. By definition this tree is an (r,k)-FAT tree, and the lemma holds. □

THEOREM 2. Let n,t be integers greater than one and let F be an (n,t)-WC file. The GREEDY heuristic can produce a trie for which S_h/S_o is approximately S_w/S_o.

PROOF:

From Lemma 5 there exists an (r,k)-THIN trie indexing F, so S_o is the size of an (r,k)-THIN trie. Now consider the selections which lead to a worst case. Referring to Figure 5.1, form a trie as follows: choose the t attributes from P, dividing the records into 2^t sets. As in the (r,k)-FAT tree a complete binary subtree will be formed. Following these selections, continue to select attributes from Q in a left-to-right order. These selections yield 2^t $(2^{n-1}-1)$-STEMS rooted in the 2^t nodes at depth t. Note that the above selections produce a modified (r,k)-FAT tree in which

the first levels agree but in which later depths grow more slowly.

To see that such selections are allowed by the GREEDY heuristic, observe that any attribute may be selected first. Once the leftmost one has been selected, any second attribute is allowed because each will add two new nodes. If any part of set P has been selected left-to-right, it will always be true that the next one can be selected since any remaining attribute choice will split each node. After all selections in P are complete, the GREEDY heuristic will choose the "best" order from set Q producing a STEM for each subtree.

Analysis in the next section shows that the size of the modified (r,k)-FAT tree described here is such that S_h/S_o is approximately S_w/S_o. Therefore, the theorem holds. □

We now consider the size of the modified (r,k)-FAT tree produced by the GREEDY heuristic. As shown in Figure 5.2, the difference between the tree in question and an (r,k)-FAT tree is that at some depth t, the modified trie stops exponential growth and has 2^t i-STEMS as subtrees. The (r,k)-FAT tree, however, continues at this depth with complete binary subtrees until all r records have been distinguished. So there are 2^t subtrees which differ in the two trees. The point to note is that the subtrees of the (r,k)-FAT tree are themselves (r/n,k-t)-FAT trees and the subtrees produced by the GREEDY heuristic are (k-t)-STEMS. We will show that the ratio of a (r,k)-FAT tree to a k-STEM is approximately equal to one for large r and k. Thus, the tries produced by the GREEDY heuristic are "close to" (r,k)-FAT trees in size. The next lemma establishes this.

LEMMA 6. Let (r,k) be a valid pair, let T be an (r,k)-FAT tree, and let S be a k-STEM. Then $|T|/|S|$ is approximately one for large r and k,

where $|T|$ denotes the size of T.

PROOF:

From the preceding analysis, we have that

$$|S| = (r-1)2^{r-1}+1, \text{ and}$$

$$|T| = r(k- \lceil \log_2 r \rceil -1)+r-1 = r(k- \lceil \log_2 r \rceil)-1$$

and since in this case $k = 2^{r-1}-1$,

$$|T| = r(2^{r-1}- \lceil \log_2 r \rceil -1)-1$$

So

$$|T|/|S| = (r(2^{r-1}- \lceil \log_2 r \rceil -1)-1)/((r-1)2^{r-1}+1)$$

$$\leq (r(2^{r-1}- \lceil \log_2 r \rceil)/((r-1)2^{r-1})$$

and since $r >> 1$, $2^{r-1}- \lceil \log_2 r \rceil$ is approximately 2^{r-1},

$$|T|/|S| \leq (r2^{r-1})/((r-1)2^{r-1}) = r/(r-1)$$

which, for large r, is one. □

Having shown that the k-STEM produced in a modified (r,k)-FAT tree are approximately the size of an (r,k)-FAT tree, we conclude the analysis of the GREEDY heuristic by stating that it allows tries which are close to the worst possible for binary restricted files. This result may be intuitively unappealing. In a sense it claims that for binary trees of r leaves at depth k, the slowest growing binary tree, a k-STEM, and the fastest growing tree, an (r,k)-FAT tree, are approximately the same size. To see why this happens, think of the k-STEM. The last k/2 depths have r nodes. So for large k, it is at least one half of the size of an (r,k)-FAT tree. Of the remaining depths, k/4 of them have r-1 nodes, k/8 have r-2, and so on.

Thus, a k-STEM grows slowly but has many levels which are almost the size of an (r,k)-FAT tree. It is the large portion of levels at which many nodes appear that account for the size.

6: Summary

We have defined an (r,k)-THIN tree which is a binary tree that is as small as any full trie indexing a binary restricted file. Furthermore, we demonstrated an (r,k)-FAT tree which is a binary tree that is as large as any full trie for a binary restricted file. An upper bound on the ratio S_w/S_o was obtained from the sizes of an (r,k)-FAT and (r,k)-THIN tree. It was shown that this bound was attainable by demonstrating a file for which both an (r,k)-FAT and (r,k)-THIN trie existed.

The GREEDY heuristic for full trie minimization was introduced for the approximation of minimum tries. This heuristic operates by choosing attributes which produce minimum splitting in a local sense. It was shown that there existed files for which this heuristic could produce tries such that the ration S_h/S_o was not bounded by a constant.

References

Aho, A.; Hopcroft, J.; and Ullman, J., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.

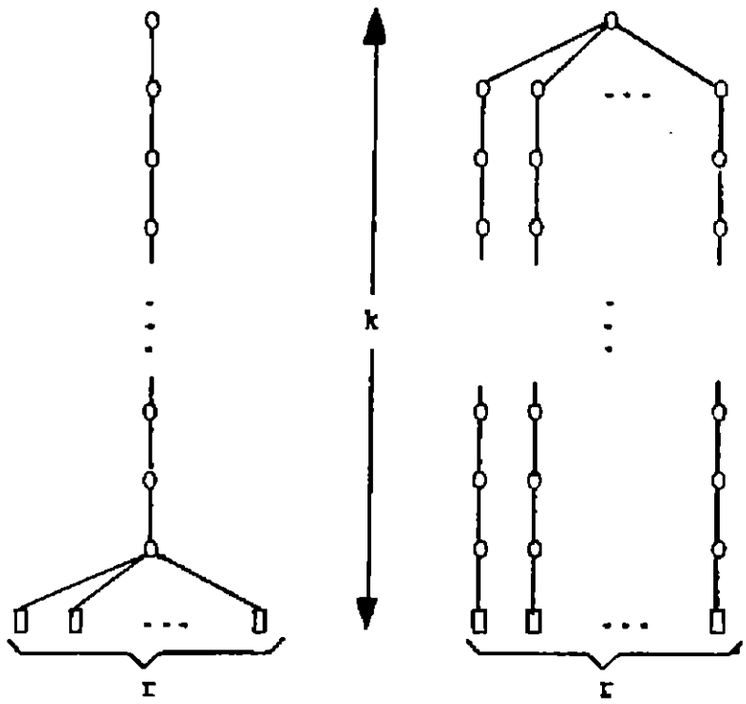
Comer, D., Trie Structured Index Minimization, Ph.D. dissertation, The Pennsylvania State University, 1976.

Comer, D.; and Sethi, R., "The Complexity of Trie Index Construction," JACM, to appear.

Fredkin, E., "Trie Memory," CACM, Vol. 3:9 (Sept. 1960), 490-499.

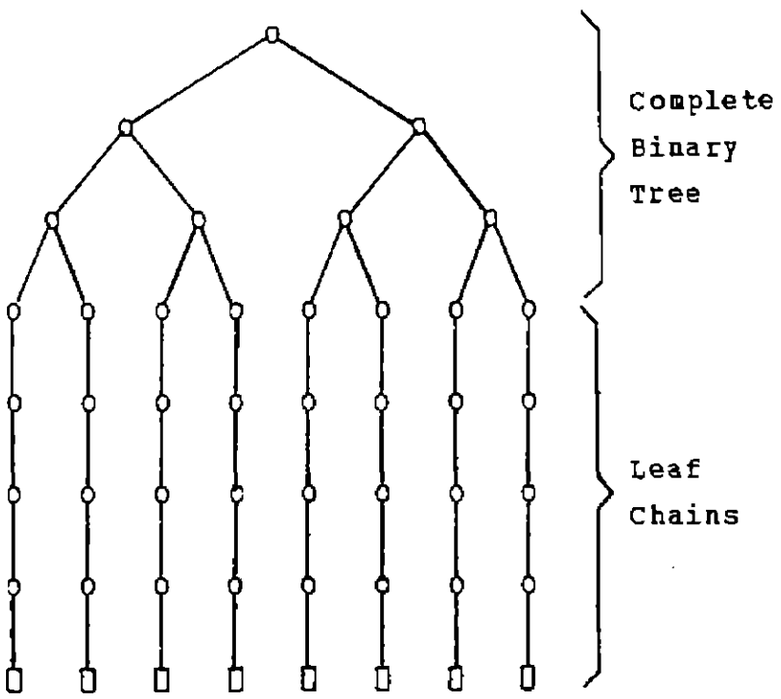
Rotwitt, T.; and deMaine, P. A. D., "Storage Optimization of Tree Structured Files Representing Descriptor Sets," Proc. 1971 ACM-SIGFIDET Workshop on Data Description Access and Control, 207-217.

Yao, S. B., "A Model for Combined Attribute Index Organizations," Proc. Fifth Texas Conference on Computing Systems, Austin Texas, Oct. 1976, 127-130.



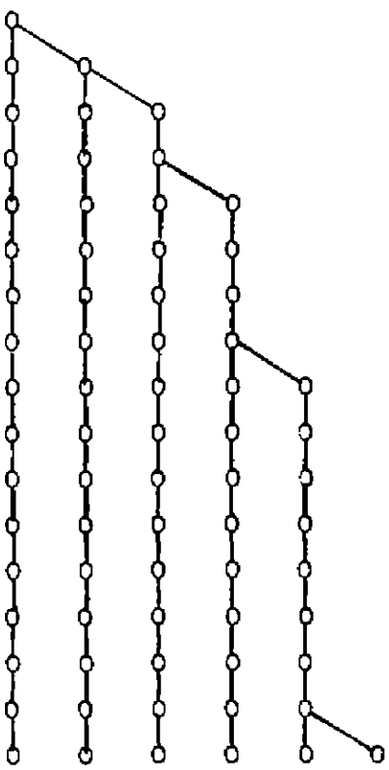
The smallest and largest full tries for an unrestricted file of r records and k attributes.

Figure 1.1



The largest trie indexing a binary restricted file of 8 records and 7 attributes.

Figure 2.1

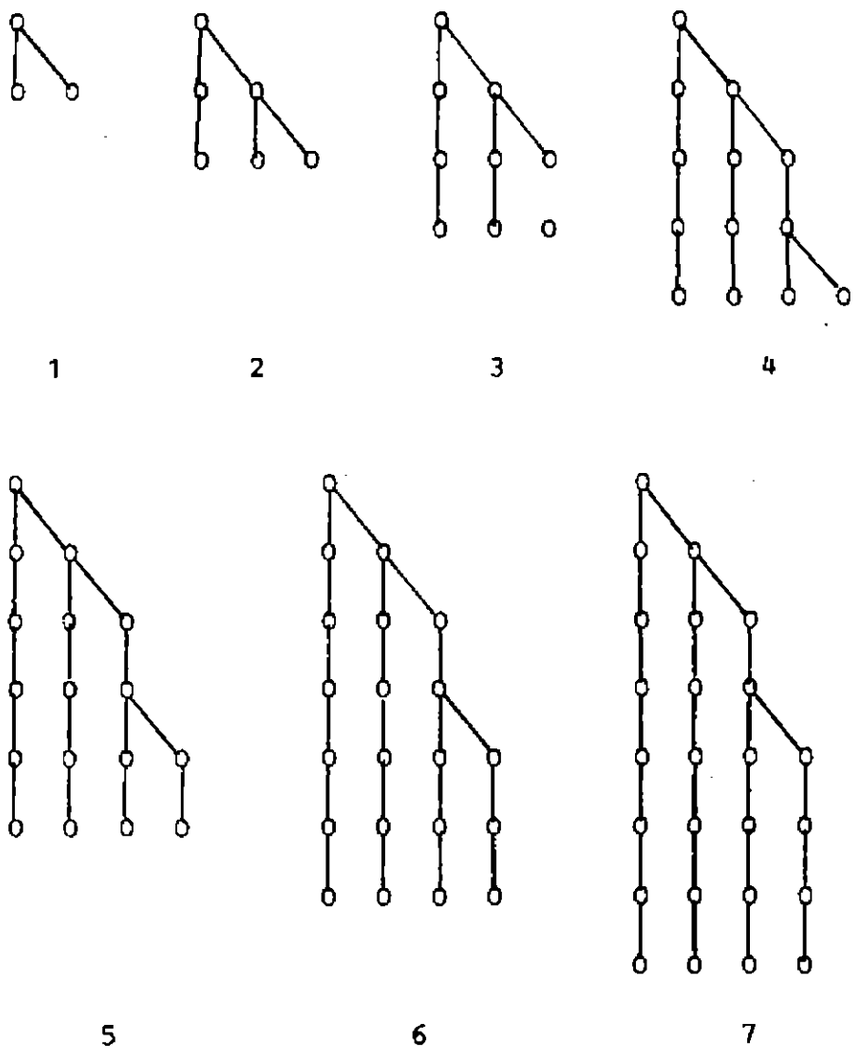


1	0	1	0	1	0	1	0	1		1
0	1	1	0	0	1	1	0	0		1
0	0	0	1	1	1	1	0	0		1
0	0	0	0	0	0	0	1	1	...	1
0	0	0	0	0	0	0	0	0		1
0	0	0	0	0	0	0	0	0		1
0	0	0	0	0	0	0	0	0		0

(a) (b)

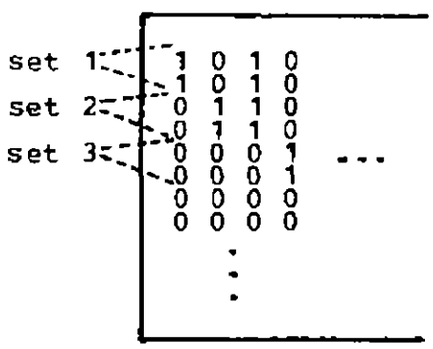
A restricted binary file and the minimum growth full trie indexing it.

Figure 3.1

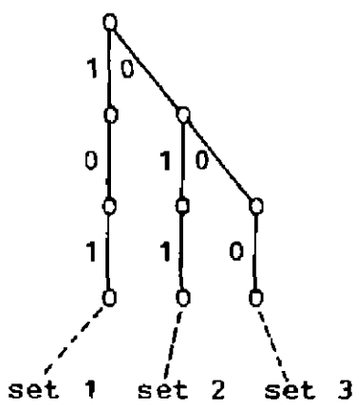


Examples of i-STEMS for $1 \leq i \leq 7$.

Figure 3.2



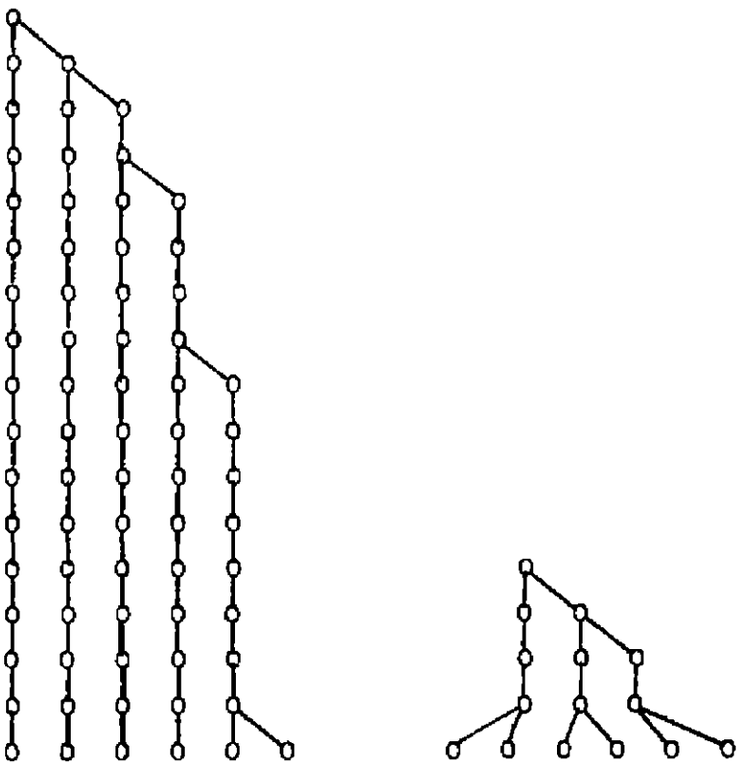
(a)



(b)

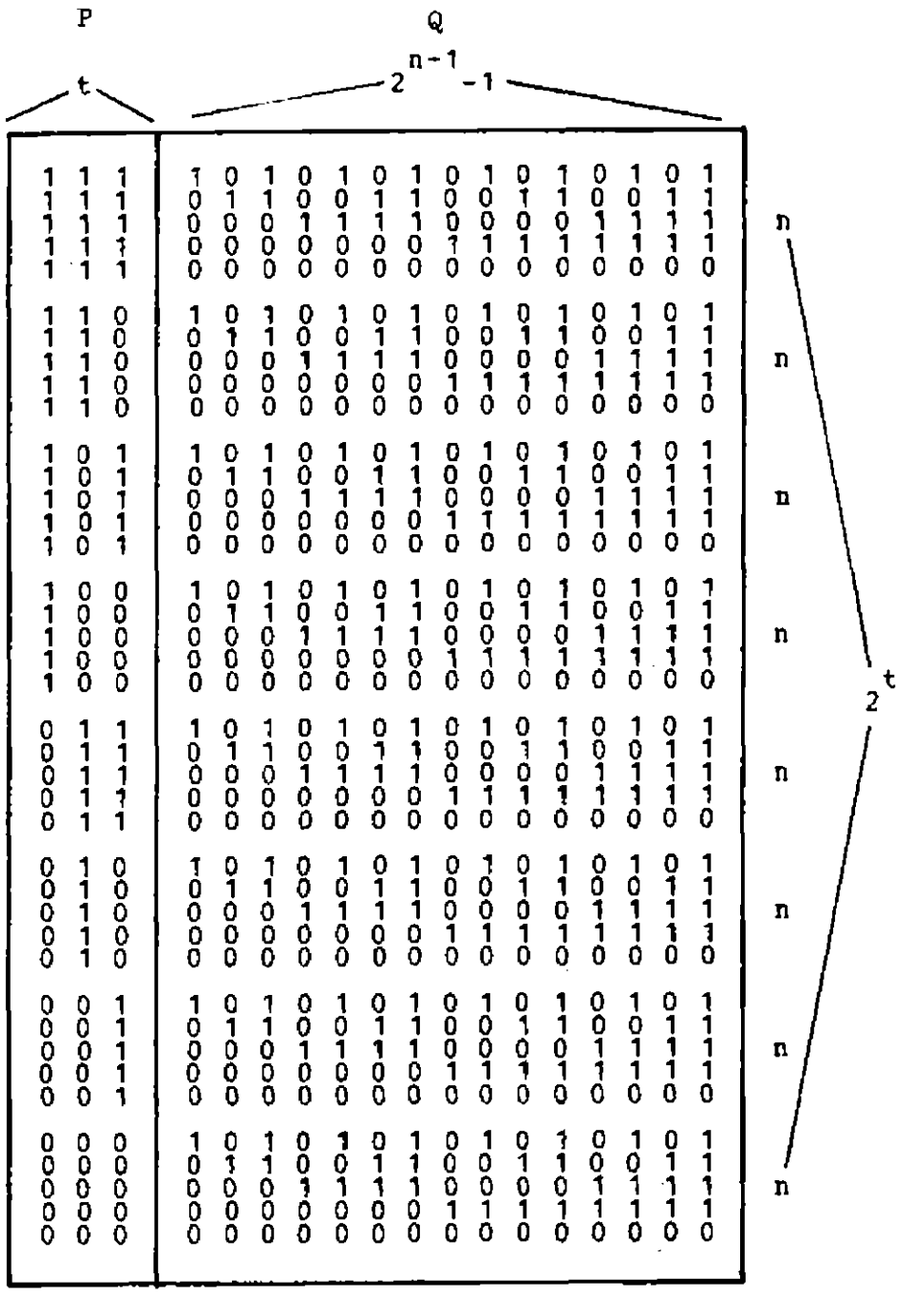
Part of a binary restricted file (a) and a slowest growing full trie for that file (b) obtained by testing attributes left-to-right.

Figure 3.3



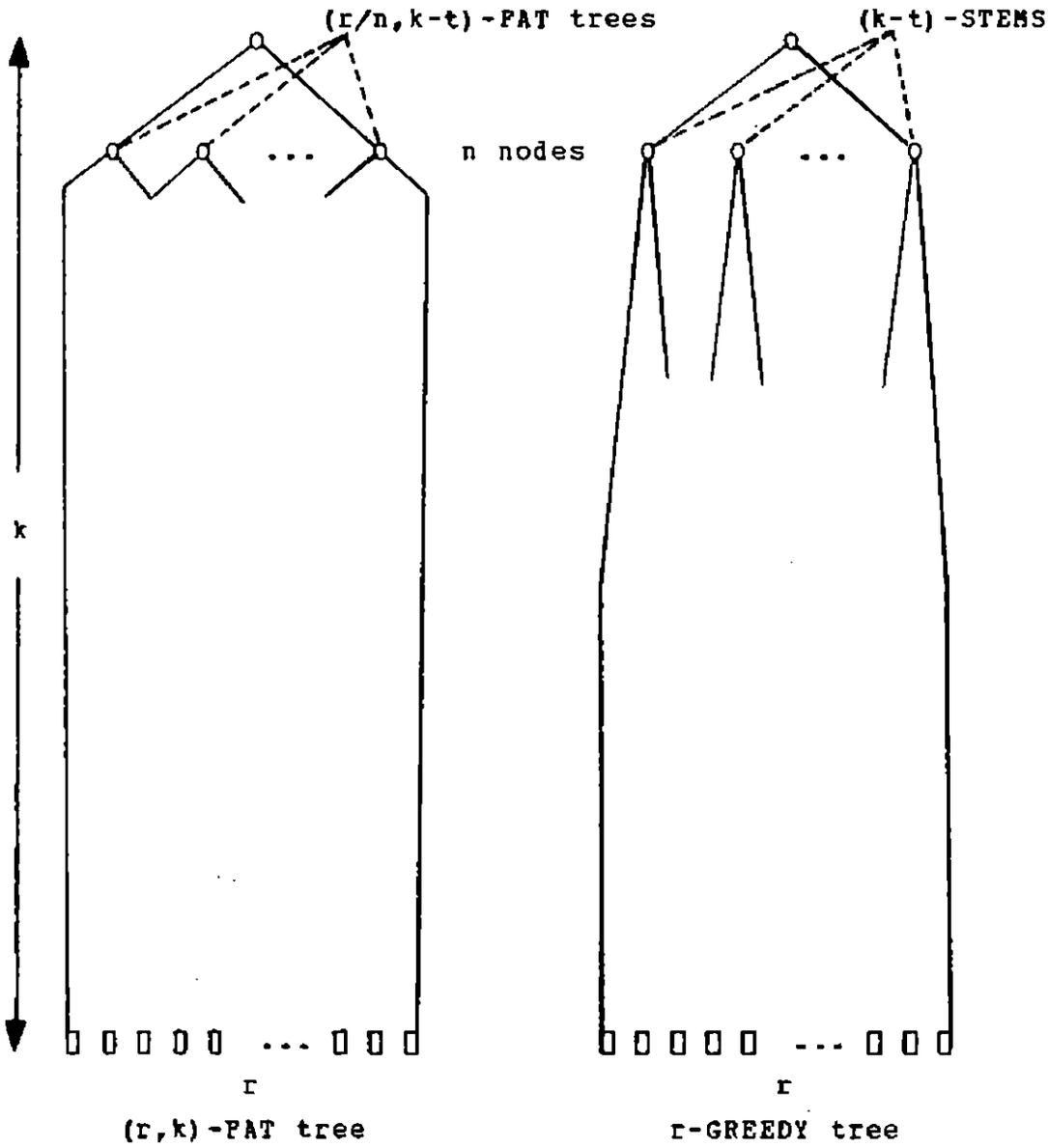
An i-STEM with six leaves (a) and an i-STEM with three leaves, each of which is the root of a binary tree.

Figure 3.4



A worst case binary restricted file for t = 3, n = 5 .

Figure 5.1



The shape of a worst case trie for the GREEDY heuristic. Note that it differs from an (r,k)-PAT tree in that there are n subtrees which are (k-t)-STEMS instead of PAT trees.

Figure 5.2