Analysis of a Heuristic for Full Trie Minimization

DOUGLAS COMER
Purdue University

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 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.

This paper considers a "greedy" heuristic for constructing low-cost tries. It presents simulation experiments which show that the greedy method tends to produce tries with small size, and analysis leading to a worst case bound on approximations produced by the heuristic. It also shows a class of files for which the greedy method may perform badly, producing tries of high cost.

Key Words and Phrases: trie index, trie size, heuristic
CR Categories: 3.74, 4.33, 4.34

1. INTRODUCTION

A trie, defined by Fredkin [6], 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 trie to a leaf, the choice of a new edge at each node being determined by an attribute value 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. Such tries are called *full tries* and have the property that the size of the trie is determined by the order in which attribute values are tested. Full tries are useful for storing information when there is a high probability of unsuccessful search because the entire key can be checked in the trie index without searching the file.

Comer and Sethi [4] show that the problem of finding an ordering of the attributes which produces a minimum size trie 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, optimum 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. DeMaine

[1] Aho, Hopcroft, and Ullman [1] provide a reasonable reference to NP-complete problems.

Author's address: Department of Computer Science, Purdue Unversity, West Lafayette, IN 47907.

and Rotwitt [5] and Yao [7] consider an alternative: procedures which are computationally efficient but which yield solutions that are close to optimal in some sense. Such procedures are often derived from "rule of thumb" practices and are called *heuristics.*

Comer [3] presents and analyzes heuristics for the class of tries in which leaf chains are removed; such tries are useful only as indexes because not all information is present in the trie itself. This work analyzes tries in which all information from a file is kept in the trie. It focuses on one method of minimizing space, called the *greedy* heuristic, providing experimental evidence that it performs well on the average, and a bound on the worst case tries produced. While a bound provides an absolute limit on the size of the tries produced, it also gives a warning about how far from optimum they could be. The paper goes on to exhibit a class of files for which the greedy heuristic may produce high cost tries. Fortunately, the type of files on which greedy misbehaves seldom occur in practical applications.

To measure the performance of a heuristic, let S_h denote the size of a trie produced by the heuristic, and let S_o denote the size of an optimum (smallest) trie for the same file. The cost of the heuristic is

$$\text{cost} = \frac{S_h}{S_o}.$$

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

Heuristics for full trie minimization are intended to produce low cost tries by minimizing the breadth of the trie. Figure 1 shows the best and worst possible tries for a file of r records and k attributes. Intuitively, the best trie consists of a long, thin spine with all branching just before the leaves, while the worst trie branches as wide as possible just below the root.

One way to produce a small trie, then, is to choose attributes in an order which minimizes the number of nodes at each level. This optimizes the trie locally by restricting growth on a level-by-level basis as the trie is grown. Of course, local optimization does not guarantee a minimum trie; the global optimum may require some levels to branch more than the minimum amount to reduce branching later. The idea of local minimization is formalized in the following:

Definition. The *greedy heuristic for full trie minimization* is given by the following procedure. While building the trie from the root down, select at each level an attribute which adds the smallest number of nodes to the next level.

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

This paper characterizes the best and worst possible full tries for a reasonable class of files, deriving a bound on the ratio of the size of the largest trie to the size of the smallest trie for any file in the class. It examines tries produced by the greedy heuristic to see how well the heuristic performs. While simulation exper-

iments indicate that the greedy method does well on typical files, a class of files is described for which the heuristic may produce high cost tries.

The rest of the paper is organized as follows. Section 2 gives the necessary definitions; Section 3 presents the simulation results and some interpretation; Section 4 defines an (r, k)-FAT tree and shows it to be as large as any trie for a binary restricted file; Section 5 defines an (r, k)-THIN tree and shows it to be as small as any trie for a binary restricted file. Finally, Section 6 defines a modified (r, k)-FAT tree and shows a class of files for which the greedy heuristic can produce tries as large as the modified trees.

2. DEFINITIONS

Definition. Let $A_1, A_2, \ldots, A_k$ be a finite set of *attributes*, where attribute A_i takes on values from the finite set V_i, $1 \leq i \leq k$. A *file* F is a subset of $V_1 \times V_2 \times \cdots \times V_k$ and a *record* is an element of F. The *alphabet size* of F is given by $\max\{|V_1|, |V_2|, \ldots, |V_k|\}$, where $|V|$ represents the number of elements in V. Files with alphabet size 2 will be referred to as *binary files.*

In the sequel it will be convenient to think of a file as a two-dimensional array with r rows, one for each record, and k columns, one for each attribute. Thus we may refer to the value of the jth attribute in the ith record of F as $F_{i,j}$. Likewise, it will be convenient to think of elements of F as integers.

Definition. Let F be a file of r records and k attributes, and let $V = \{1, 2, \ldots, a\}$, where a is the alphabet size of F. Attribute m is *trivial* in F iff $\forall i$, $1 \leq i \leq r$, $F_{i,m} = v$, where v is some fixed value from V. Attribute m is *isomorphic* to attribute n iff $\exists$ an automorphism $S: V \rightarrow V$ such that $\forall i$, $1 \leq i \leq r$, $F_{i,m} = S(F_{i,n})$. F is a *restricted* file iff F contains no trivial attributes and no distinct pair of isomorphic attributes.

Graph definitions used throughout the paper are standard (see [1]).

Definition. A *full trie* for a file F is a tree with all leaves at depth[2] k such that the following hold:

(1) Let $A_1, A_2, \ldots, A_k$ be the attributes of F, and let π be a permutation of 1, 2, $\ldots$, k. All edges leaving a node at depth $i - 1$ have distinct labels chosen from $V_{\pi(i)}$ for all i, $1 \leq i \leq k$.
(2) The labels encountered on each path from the root to a leaf correspond to an element of F, and, for each element of F there is such a path.

The *size* of a trie is the number of nonleaf nodes in it.

3. SIMULATION RESULTS FOR THE GREEDY HEURISTIC

The analysis in [5] shows that the greedy heuristic produces optimum tries for a file with a record for each possible attribute combination, and summarizes experiments that indicate a wide class of heuristics including greedy perform well on randomly generated files. This paper extends the experiments to include small files and peaked (nonuniform) distributions.

Tables I–IV present a sample of simulation studies for randomly generated files

[2] The root of a tree lies at depth 0; children of a node at depth $i - 1$ lie at depth i.

Table I

Size	Degrees	Overall Best	Overall Worst	Overall Mean	Greedy Heuristic Best	Greedy Heuristic Worst	Greedy Heuristic Mean	% Gr. Opt.	% Opt. Greedy
Uniform Distribution									
50	10 12 14 16	149	157	153	149	149	149	100.0	100.0
50	20 22 24 26	162	170	166	162	166	164	33.3	100.0
50	10 15 20 25	154	167	161	154	154	154	100.0	100.0
50	10 10 30 30	152	173	162	152	155	153	33.3	100.0
50	20 20 20 21	162	168	165	162	162	162	100.0	100.0
50	30 30 30 31	170	175	173	170	170	170	100.0	100.0
50	30 30 30 34	169	175	173	169	169	169	100.0	100.0
50	15 15 15 15	154	161	158	154	159	155	50.0	100.0
50	20 20 20 20	158	168	164	158	158	158	100.0	100.0
50	26 26 26 26	163	172	169	163	163	163	100.0	100.0
50	30 30 30 30	172	174	173	172	172	172	100.0	100.0
50	31 31 31 31	170	175	173	170	172	171	50.0	100.0
50	32 32 32 32	171	176	174	171	174	172	25.0	100.0
50	33 33 33 33	170	178	174	170	170	170	100.0	100.0
50	34 34 34 34	167	176	172	167	167	167	100.0	100.0
Normal Distribution $\mu = 25, \sigma = 12.5$									
50	10 12 14 16	103	124	113	103	103	103	100.0	100.0
50	20 22 24 26	143	157	149	143	143	143	100.0	100.0
50	10 15 20 25	112	146	129	112	112	112	100.0	100.0
50	10 10 30 30	115	154	135	115	115	115	100.0	100.0
50	20 20 20 21	139	147	143	141	141	141	0.0	0.0
50	30 30 30 31	150	161	155	150	150	150	100.0	100.0
50	30 30 30 34	152	161	155	152	152	152	100.0	25.0
50	15 15 15 15	103	110	106	103	103	103	100.0	100.0
50	20 20 20 20	139	147	143	141	141	141	0.0	0.0
50	26 26 26 26	150	161	155	150	150	150	100.0	100.0
50	30 30 30 30	150	161	155	150	150	150	100.0	100.0
50	31 31 31 31	150	161	155	150	150	150	100.0	100.0
50	32 32 32 32	159	165	162	159	161	160	50.0	100.0
50	33 33 33 33	159	165	162	159	161	160	50.0	100.0
50	34 34 34 34	159	165	162	159	161	160	50.0	100.0
Normal Distribution $\mu = 25, \sigma = 6.25$									
50	10 12 14 16	103	110	106	103	103	103	100.0	100.0
50	20 22 24 26	103	110	106	103	103	103	100.0	100.0
50	10 15 20 25	103	110	106	103	103	103	100.0	100.0
50	10 10 30 30	103	110	106	103	103	103	100.0	100.0
50	20 20 20 21	103	110	106	103	103	103	100.0	100.0
50	30 30 30 31	103	110	106	103	103	103	100.0	100.0
50	30 30 30 34	103	124	113	103	103	103	100.0	100.0
50	15 15 15 15	103	110	106	103	103	103	100.0	100.0
50	20 20 20 20	103	110	106	103	103	103	100.0	100.0
50	26 26 26 26	103	110	106	103	103	103	100.0	100.0
50	30 30 30 30	103	110	106	103	103	103	100.0	100.0
50	31 31 31 31	103	110	106	103	103	103	100.0	100.0
50	32 32 32 32	139	147	143	141	141	141	0.0	0.0
50	33 33 33 33	139	147	143	141	141	141	0.0	0.0
50	34 34 34 34	139	147	143	141	141	141	0.0	0.0

Table II

Uniform Distribution		Overall			Greedy Heuristic			% Gr.	% Opt.
Size	Degrees	Best	Worst	Mean	Best	Worst	Mean	Opt.	Greedy
100	10 12 14 16	278	295	286	278	278	278	100.0	100.0
100	20 22 24 26	308	318	313	308	308	308	100.0	100.0
100	10 15 20 25	284	313	299	284	284	284	100.0	100.0
100	10 10 30 30	275	323	300	275	275	275	100.0	100.0
100	20 20 20 21	304	314	308	304	304	304	100.0	100.0
100	30 30 30 31	319	325	322	319	319	319	100.0	100.0
100	30 30 30 34	322	330	324	322	322	322	100.0	50.0
100	15 15 15 15	290	296	293	290	294	291	50.0	100.0
100	20 20 20 20	304	313	306	304	304	304	100.0	100.0
100	26 26 26 26	313	320	316	314	314	314	0.0	0.0
100	30 30 30 30	321	325	323	321	321	321	100.0	100.0
100	31 31 31 31	322	325	323	322	323	322	66.7	100.0
100	32 32 32 32	325	330	326	325	325	325	100.0	100.0
100	33 33 33 33	324	331	327	324	324	324	100.0	66.7
100	34 34 34 34	323	330	326	323	324	323	33.3	100.0
Normal Distribution $\mu = 50, \sigma = 25$									
100	10 12 14 16	178	215	198	178	178	178	100.0	100.0
100	20 22 24 26	260	282	271	260	260	260	100.0	100.0
100	10 15 20 25	198	261	230	198	198	198	100.0	100.0
100	10 10 30 30	201	275	239	201	201	201	100.0	100.0
100	20 20 20 21	249	260	255	249	251	250	50.0	100.0
100	30 30 30 31	279	288	284	279	283	281	50.0	100.0
100	30 30 30 34	279	301	288	279	283	281	50.0	100.0
100	15 15 15 15	170	175	172	170	171	170	57.1	100.0
100	20 20 20 20	249	260	255	249	251	250	50.0	100.0
100	26 26 26 26	279	288	284	279	283	281	50.0	100.0
100	30 30 30 30	279	288	284	279	283	281	50.0	100.0
100	31 31 31 31	279	288	284	279	293	281	50.0	100.0
100	32 32 32 32	299	307	302	299	301	300	50.0	100.0
100	33 33 33 33	299	307	302	299	301	300	50.0	100.0
100	34 34 34 34	299	307	302	299	301	300	50.0	100.0
Normal Distribution $\mu = 50, \sigma = 12.5$									
100	10 12 14 16	170	175	172	170	171	170	57.1	100.0
100	20 22 24 26	170	175	172	170	171	170	57.1	100.0
100	10 15 20 25	170	175	172	170	171	170	57.1	100.0
100	10 10 30 30	170	175	172	170	171	170	57.1	100.0
100	20 20 20 21	170	175	172	170	171	170	57.1	100.0
100	30 30 30 31	170	175	172	170	171	170	57.1	100.0
100	30 30 30 34	178	215	198	178	178	178	100.0	100.0
100	15 15 15 15	170	175	172	170	171	170	57.1	100.0
100	20 20 20 20	170	175	172	170	171	170	57.1	100.0
100	26 26 26 26	170	175	172	170	171	170	57.1	100.0
100	30 30 30 30	170	175	172	170	171	170	57.1	100.0
100	31 31 31 31	170	175	172	170	171	170	57.1	100.0
100	32 32 32 32	249	260	255	249	251	250	50.0	100.0
100	33 33 33 33	249	260	255	249	251	250	50.0	100.0
100	34 34 34 34	249	260	255	249	251	250	50.0	100.0

Table III

Uniform Distribution					Overall			Greedy Heuristic			% Gr.	% Opt.
Size	Degrees				Best	Worst	Mean	Best	Worst	Mean	Opt.	Greedy
500	10	12	14	16	1058	1164	1107	1058	1058	1058	100.0	100.0
500	20	22	24	26	1309	1366	1335	1309	1309	1309	100.0	100.0
500	10	15	20	25	1115	1309	1209	1115	1115	1115	100.0	100.0
500	10	10	30	30	1056	1391	1226	1056	1056	1056	100.0	100.0
500	20	20	20	21	1282	1311	1295	1282	1287	1283	66.7	100.0
500	30	30	30	31	1402	1425	1414	1402	1415	1406	50.0	100.0
500	30	30	30	34	1402	1428	1418	1402	1421	1408	66.7	100.0
500	15	15	15	15	1177	1191	1181	1177	1177	1177	100.0	83.3
500	20	20	20	20	1274	1290	1284	1274	1287	1279	40.0	100.0
500	26	26	26	26	1351	1382	1369	1351	1366	1357	50.0	100.0
500	30	30	30	30	1401	1425	1410	1401	1404	1402	50.0	100.0
500	31	31	31	31	1406	1428	1418	1406	1417	1411	40.0	100.0
500	32	32	32	32	1411	1438	1423	1411	1424	1414	80.0	100.0
500	33	33	33	33	1432	1449	1438	1432	1437	1433	40.0	100.0
500	34	34	34	34	1421	1456	1435	1421	1437	1426	50.0	100.0
Normal Distribution $\mu = 250, \sigma = 12.5$												
500	10	12	14	16	458	551	514	458	458	458	100.0	100.0
500	20	22	24	26	931	1062	993	931	931	931	100.0	100.0
500	10	15	20	25	614	844	729	614	614	614	100.0	100.0
500	10	10	30	30	672	953	794	672	672	672	100.0	100.0
500	20	20	20	21	856	873	865	856	862	858	66.7	100.0
500	30	30	30	31	1071	1096	1081	1071	1071	1071	100.0	100.0
500	30	30	30	34	1073	1137	1108	1073	1073	1073	100.0	100.0
500	15	15	15	15	382	392	385	382	382	382	100.0	100.0
500	20	20	20	20	856	873	865	856	862	858	66.7	100.0
500	26	26	26	26	1071	1096	1081	1071	1071	1071	100.0	100.0
500	30	30	30	30	1071	1096	1081	1071	1071	1071	100.0	100.0
500	31	31	31	31	1071	1096	1081	1071	1071	1071	100.0	100.0
500	32	32	32	32	1180	1200	1193	1180	1180	1180	100.0	100.0
500	33	33	33	33	1180	1200	1193	1180	1180	1180	100.0	100.0
500	34	34	34	34	1180	1200	1193	1180	1180	1180	100.0	100.0
Normal Distribution $\mu = 250, \sigma = 62.5$												
500	10	12	14	16	382	392	185	382	382	382	100.0	100.0
500	20	22	24	26	382	392	385	382	382	382	100.0	100.0
500	10	15	20	25	382	392	385	382	382	382	100.0	100.0
500	10	10	30	30	382	392	385	382	382	382	100.0	100.0
500	20	20	20	21	382	392	385	382	382	382	100.0	100.0
500	30	30	30	31	382	392	385	382	382	382	100.0	100.0
500	30	30	30	34	458	551	514	458	458	458	100.0	100.0
500	15	15	15	15	382	392	385	382	382	382	100.0	100.0
500	20	20	20	20	382	392	385	382	382	382	100.0	100.0
500	26	26	26	26	382	392	385	382	382	382	100.0	100.0
500	30	30	30	30	382	392	385	382	382	382	100.0	100.0
500	31	31	31	31	382	392	385	382	382	382	100.0	100.0
500	32	32	32	32	856	873	865	856	862	858	66.7	100.0
500	33	33	33	33	856	873	865	856	862	858	66.7	100.0
500	34	34	34	34	856	873	865	856	862	858	66.7	100.0

Table IV

Uniform Distribution Size	Degrees	Overall Best	Overall Worst	Overall Mean	Greedy Heuristic Best	Greedy Heuristic Worst	Greedy Heuristic Mean	% Gr. Opt.	% Opt. Greedy
5000	10 12 14 16	6274	7069	6643	6274	6274	6274	100.0	100.0
5000	20 22 24 26	9389	9795	9583	9389	9389	9389	100.0	100.0
5000	10 15 20 25	7446	8956	8130	7446	7446	7446	100.0	100.0
5000	10 10 30 30	7383	9623	8367	7383	7383	7383	100.0	100.0
5000	20 20 20 21	9081	9176	9115	9081	9081	9081	100.0	100.0
5000	30 30 30 31	10467	10543	10505	10467	10488	10481	33.3	50.0
5000	30 30 30 34	10484	10676	10583	10484	10486	10484	66.7	100.0
5000	15 15 15 15	7565	7615	7595	7565	7604	7582	50.0	100.0
5000	20 20 20 20	9053	9088	9069	9053	9074	9060	50.0	100.0
5000	26 26 26 26	10013	10063	10042	10013	10051	10028	50.0	100.0
5000	30 30 30 30	10454	10491	10477	10454	10476	10460	50.0	100.0
5000	31 31 31 31	10560	10579	10570	10560	10573	10565	40.0	100.0
5000	32 32 32 32	10645	10680	10661	10645	10662	10651	33.3	100.0
5000	33 33 33 33	10760	10794	10781	10760	10781	10770	50.0	100.0
5000	34 34 34 34	10823	10883	10855	10823	10870	10850	25.0	50.0
Normal Distribution $\mu = 2500, \sigma = 1250$									
5000	10 12 14 16	1277	1464	1400	1277	1277	1277	100.0	100.0
5000	20 22 24 26	4879	5386	5115	4879	4879	4879	100.0	100.0
5000	10 15 20 25	2234	2818	2530	2234	2234	2234	100.0	100.0
5000	10 10 30 30	2736	3508	3085	2736	2736	2736	100.0	100.0
5000	20 20 20 21	3835	3877	3850	3835	3835	3835	100.0	100.0
5000	30 30 30 31	6365	6416	6389	6365	6365	6365	100.0	100.0
5000	30 30 30 34	6545	6975	6829	6545	6545	6545	100.0	100.0
5000	15 15 15 15	916	928	921	916	921	917	40.0	100.0
5000	20 20 20 20	3835	3877	3850	3835	3835	3835	100.0	100.0
5000	26 26 26 26	6365	6416	6389	6365	6365	6365	100.0	100.0
5000	30 30 30 30	6365	6416	6389	6365	6365	6365	100.0	100.0
5000	31 31 31 31	6365	6416	6389	6365	6365	6365	100.0	100.0
5000	32 32 32 32	7947	7987	7965	7947	7947	7947	100.0	100.0
5000	33 33 33 33	7947	7987	7965	7947	7947	7947	100.0	100.0
5000	34 34 34 34	7947	7987	7965	7947	7947	7947	100.0	100.0
Normal Distribution $\mu = 2500, \sigma = 625$									
5000	10 12 14 16	916	928	921	916	921	917	40.0	100.0
5000	20 22 24 26	916	928	921	916	921	917	40.0	100.0
5000	10 15 20 25	916	928	921	916	921	917	40.0	100.0
5000	10 10 30 30	916	928	921	916	921	917	40.0	100.0
5000	20 20 20 21	916	928	921	916	921	917	40.0	100.0
5000	30 30 30 31	916	928	921	916	921	917	40.0	100.0
5000	30 30 30 34	1277	1464	1400	1277	1277	1277	100.0	100.0
5000	15 15 15 15	916	928	921	916	921	917	40.0	100.0
5000	20 20 20 20	916	928	921	916	921	917	40.0	100.0
5000	26 26 26 26	916	928	921	916	921	917	40.0	100.0
5000	30 30 30 30	916	928	921	916	921	917	40.0	100.0
5000	31 31 31 31	916	928	921	916	921	917	40.0	100.0
5000	32 32 32 32	3835	3877	3850	3835	3835	3835	100.0	100.0
5000	33 33 33 33	3835	3877	3850	3835	3835	3835	100.0	100.0
5000	34 34 34 34	3835	3877	3850	3835	3835	3835	100.0	100.0

of 50, 100, 500, and 5000 records. In each table, the columns labeled "degrees" give the size of the sets from which attribute values were chosen. Thus, the first file in Table I contains 50 records, each with 4 attributes, where the first attribute takes on values between 1 and 10, the second between 1 and 12, the third between 1 and 14, and the fourth between 1 and 16. For the uniform distribution experiments, the values in a particular attribute were selected with equal probability from the entire range. For a skewed distribution, values were generated to approximate a normal distribution with mean and standard deviation as shown.

One must remember that while a particular implementation of the greedy heuristic will choose one single attribute at each stage, there may be two or more attributes which both produce the minimal branching at that stage. Thus Tables I–IV compare the best, worst, and average tries that could be produced by greedy with the best, worst, and average tries for all possible attribute orders.

The data support several observations.

(1) The greedy heuristic tends to produce low cost tries. In all experiments, the mean size of tries produced by the greedy heuristic was smaller than the mean size of all possible tries. Furthermore, the size of the worst trie produced by the greedy heuristic was seldom as large as the size of the worst possible trie.
(2) The greedy heuristic sometimes fails to produce an optimum trie. In some simulations only a portion of the possible greedy tries were nonoptimum; in other cases all possible greedy tries were nonoptimum. Failure occurred relatively infrequently, however, and even when it did, the average greedy trie was smaller than the average overall trie.
(3) The percentage of optimum tries which are also greedy tries is usually larger than the percentage of greedy tries which are optimum. Thus, while the set of all greedy tries tends to be larger than the set of all optimum tries, it tends to include most of the optimum ones.

The experiments suggest that the greedy heuristic performs well on typical files; the next sections analyze its performance in more detail.

4. LARGEST TRIES FOR BINARY RESTRICTED FILES

The smallest and largest full tries for a file of r records and k attributes are shown in Figure 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

$$\frac{S_w}{S_o} = r\frac{k-1}{k} + \frac{1}{k}$$

which results in a factor of r for most k.

Files which allow tries as small as k nodes are not realistic, however, because they have $k - 1$ trivial attributes, attributes which contain no information. Since we seek to model the files one might encounter in practice, we consider only restricted files, that is, those with no trivial or isomorphic attributes. In the rest of this paper, the term "file" will mean restricted file.

The analysis which follows characterizes the smallest and largest tries indexing a binary restricted file. Attention to the binary case is motivated by two factors.

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 representation of a more general case. On the other hand, it is desirable to obtain information about the simple binary case as a prelude to understanding files of higher degree. Observe, for example, that if one attribute of a file has a ternary value set while all others are binary, the file has alphabet size 3, even though analysis of the binary case applies directly.

The following simple property of binary restricted files is 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*

$$\lceil lg\ r \rceil \leq k < 2^{r-1}. \tag{1}$$

PROOF

a. $(k \geq \lceil \lg r \rceil)$.[3] Suppose $k < \lceil \lg r \rceil$. Let t be an integer such that $2^{t-1} < r \leq 2^t$. Then $\lceil \lg r \rceil = t$. Recall that a binary tree with maximum 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 $k \geq \lg r$.

b. $(k < 2^{r-1})$. Suppose $k \geq 2^{r-1}$. Think of a file as a two-dimensional array, and view the binary values in each column as a binary number of r bits. Clearly, one-half of the possible numbers are isomorphic up to a renaming of bits. Of the 2^{r-1} remaining values, one of them is all zeros (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 eq. (1) appear so often throughout the paper, this condition is given a name.

Definition. A pair of integers (r, k) is *valid* iff

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

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.

The tries shown in Figure 1 are the best and worst tries for an unrestricted file. For a binary restricted file, tries of the shape in Figure 1 are prohibited because a node may have at most two direct descendants. In the worst case trie for a binary restricted file, rapid branching occurs as early as possible but will be slightly slower due to the binary constraint.

Consider the trie shown in Figure 2 for a binary file of eight records and seven attributes. The first three levels form a complete binary tree, distinguishing all records as early as possible. The remaining levels contain only chains as in the worst case trie for an unrestricted file. Of course, this example shows a tree where the number of leaves is a power of two. Rapidly branching trees with an arbitrary number of leaves are defined by the following.

[3] lg is used throughout this paper to indicate $\log_2$.

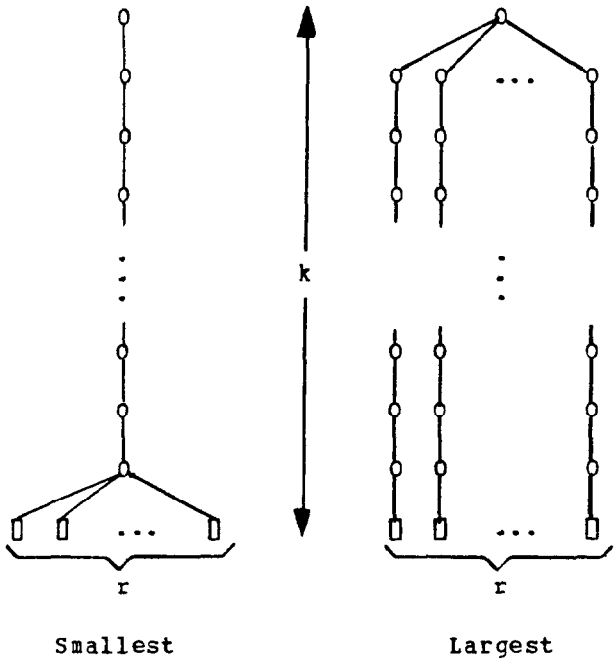


Fig. 1. The smallest and largest tries for an unrestricted file of r records and k attributes.

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

(1) each node at depth d, $0 \leq d < t$, has two children;
(2) $r - 2^t$ nodes at depth t have two children and the remaining $2^{t+1} - r$ nodes have one child; and
(3) each node at depth d, $t + 1 \leq d \leq k - 1$, has exactly one child.

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 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 children, 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. Moreover, each node at depth d has one child for all 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. □

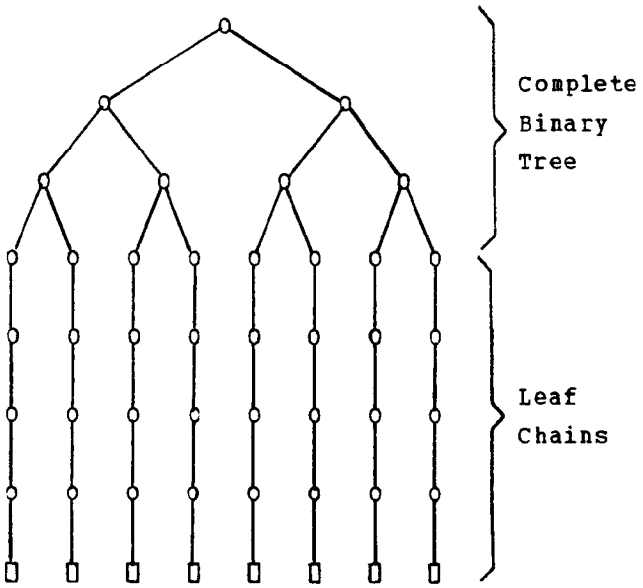


Fig. 2. The largest trie indexing a binary restricted file of eight records and seven attributes.

5. SMALLEST TRIES FOR BINARY RESTRICTED FILES

This section defines a class of binary trees which are as small as any trie indexing a binary restricted file, and identifies the shape of tries for which the ratio of sizes of the largest to smallest trie is maximized. It begins by characterizing the most slowly growing trie for a binary restricted file.

The smallest trie for an unrestricted file exhibits no splitting until the level just before the leaves as shown in Figure 1. A binary restricted file has no trivial attributes, however, so tries indexing binary restricted files cannot consist of a single chain of nodes. The absence of trivial or isomorphic attributes implies that only a finite number of attributes may be tested before a binary branch must appear in the trie. In fact, a minimum growth trie for a binary restricted file has the shape shown in Figure 3. This trie exhibits an exponentially increasing number of levels between the appearance of a binary branch.

The following gives a formal definition of the slowly growing trees described above. We show that tries indexing a binary restricted file cannot have fewer nodes at any depth than these trees.

Definition. 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 children and all other nonleaf nodes have exactly one child.

Examples of i-STEMS are shown in Figure 4.

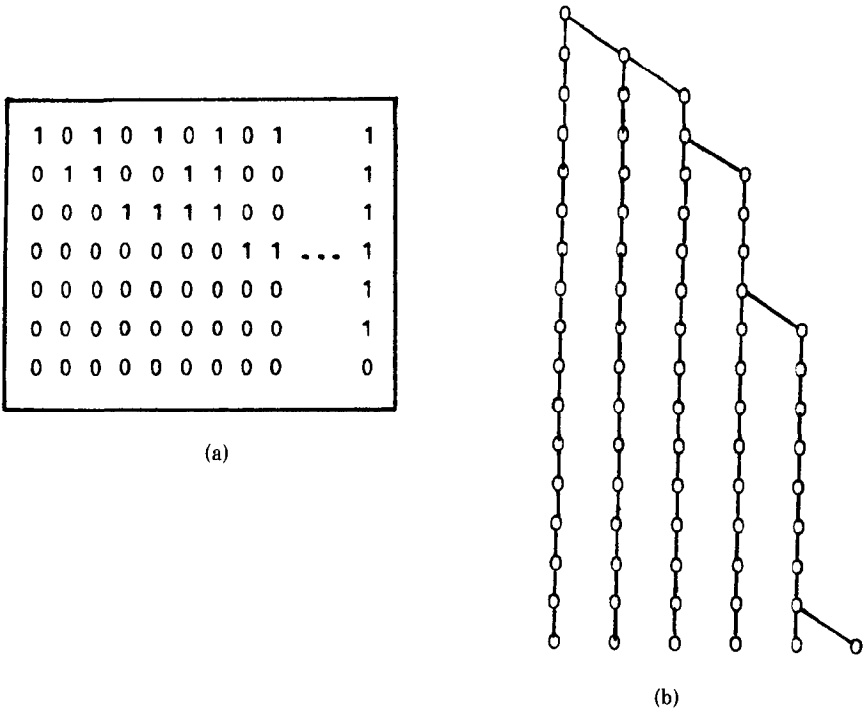


Fig. 3. (a) A binary restricted file. (b) The minimum growth full trie indexing it.

To understand the shape of an i-STEM, consider a binary restricted file. Because there are no trivial attributes, the first attribute tested must split the records into two groups. The second attribute must subdivide at least one of these groups into two or it would be isomorphic to the first. The third selection, however, could be such that it did not further divide the sets of records and yet still not be isomorphic to either of the first two attributes; Figure 5 shows an example. In Figure 5, testing the attributes from left to right distinguishes the first set of records at depth one, the second set at depth two, but no set at depth three. Following the first three selections, at least one more set must be distinguished no matter which attribute is tested. Using these ideas we 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. T must have at least $\lfloor lg\ i \rfloor + 2$ 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 level possible in T with no binary branching after the second set of records is distinguished.

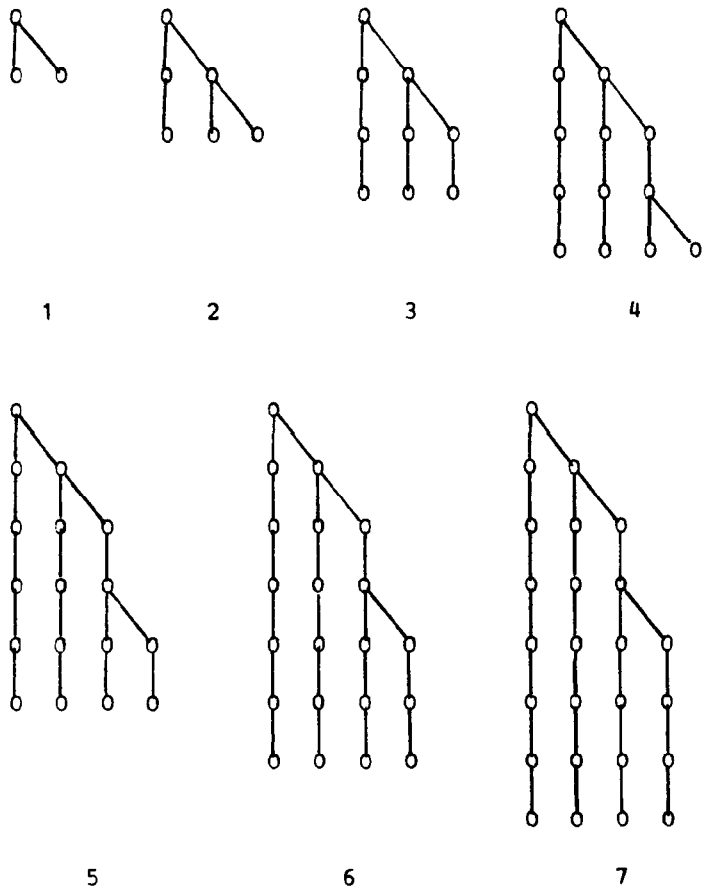


Fig. 4. Examples of i-STEMS for $1 \leq i \leq 7$.

Now assume that there are $2^{j-1} - 1$ levels possible with no branching nodes after the jth set of records has been distinguished, for $2 \leq j < p$. Suppose that the pth set of records is distinguished. Think of the assignment of values to sets of records as assigning bit values to a p-bit number. There are only 2^p possible assignments, and one-half of these were used after the $p - 1$st set was distinguished. Therefore, only 2^{p-1} additional levels can appear in the trie before another set of records must be distinguished. Thus T must have an attribute tested at depth 2^{p-1} which distinguishes the $p + 1$st set. 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 show that A has $\lfloor \lg i \rfloor + 2$ 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 $\lfloor \lg 2^j \rfloor + 2$ 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 $2^t - 1$ has an additional child, there

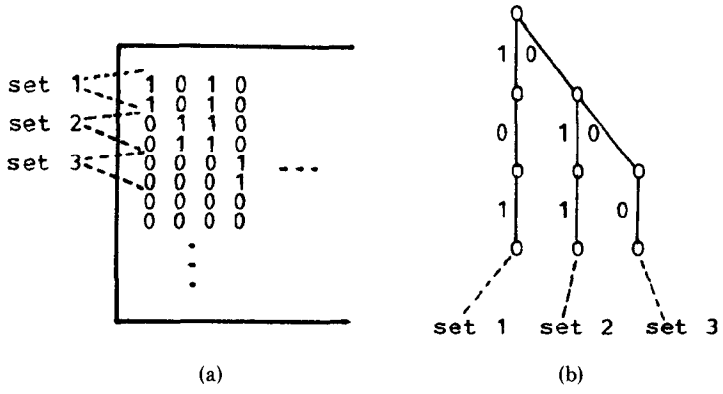


Fig. 5. (a) Part of a binary restricted file. (b) A slowest growing full trie for that file obtained by testing the attributes left to right.

must be $(\lfloor \lg 2^t - 1 \rfloor + 2) + 1$ nodes at each depth. But

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

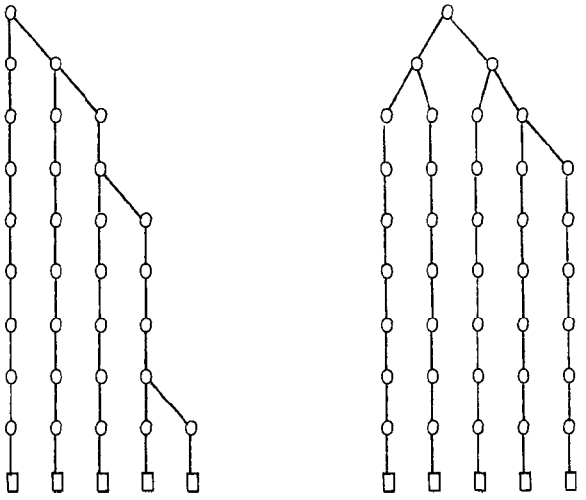
Therefore, the lemma holds by induction. □

Since an i-STEM is the most slowly growing full trie, one might think that the ratio of the size of an (r, k)-FAT tree to the size of a k-STEM would provide a bound on the ratio of the sizes of the worst case to best case tries for a binary restricted file with a given number r of records. Figure 6 shows, however, that this is not the case. In Figure 6a, the 9-STEM with 5 leaves has 30 internal nodes and the (5, 9)-FAT tree has 37 internal nodes. The modified 4-STEM in Figure 6b has only 9 internal nodes while the (5, 4)-FAT tree has 12. Since $37/30 < 12/9$, the trees in Figure 6b have a higher cost ratio than those in Figure 6a. Relating this example back to the tries shown in Figure 1, one can see that the ratio of sizes of the largest to smallest trie is maximized when the i-STEM has lower levels which are complete binary subtrees. The rapidly growing subtrees correspond to the rapid growth just before the leaves in the trie of Figure 1.

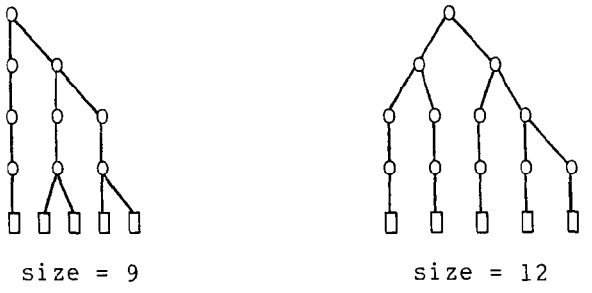
Consider an i-STEM in which the leaves have been made the roots of a forest of binary trees. At each depth in the i-STEM, there must be at least $n(i) = \lfloor \lg i \rfloor + 2$ nodes. If there are to be r leaves at depth k, where $r > n(k)$, there must be some depth p at which the tree begins to grow more rapidly than an i-STEM. Since one should delay splitting as long as possible, p should be maximized. To see how the best value for p is obtained, observe that at least $\lceil \lg q \rceil$ levels are required in a binary tree of 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 lie at the same depth, we need

$$d = \max_{t \in F} \lceil \lg(\text{leaves in } t) \rceil$$

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



(a)



(b)

Fig. 6. (a) A 9-STEM and a (5, 9)-FAT tree compared to (b) a 3-STEM with binary trees rooted in its leaves and a (5, 4)-FAT tree.

of additional depths needed to distinguish all r leaves is $\lceil \lg \lceil r/n(i) \rceil \rceil = \lceil \lg(r/n(i)) \rceil$.

An (r, k)-THIN tree is defined in terms of an i-STEM in which the leaves form the roots of a forest of binary trees. It is then shown that the ratio of the size of an (r, k)-FAT tree to the size of an (r, k)-THIN tree is maximized when all trees in the forest are complete binary trees of equal size. Finally, the section concludes by showing that no trie indexing a binary restricted file is smaller than an (r, k)-THIN tree.

LEMMA 4. *Let (r, k) be a valid pair of integers. Then there exists a positive integer i satisfying*

$$k = i + \left\lceil lg \frac{r}{\lfloor lg\ 4i \rfloor} \right\rceil.$$

PROOF. By inspection the statement holds for $k = \lceil \lg r \rceil$ when $i = 1$. It is sufficient to show that the sequence given by

$$i + \lceil \lg r - \lg(\lg\lfloor 4i \rfloor) \rceil - \lceil \lg r \rceil$$

for $i = 1, 2, \ldots$, has no two terms which differ by more than 1 and is nondecreasing. It is nondecreasing because $i \geq \lg((\lg i) + 2)$ for $i \geq 1$.

Suppose that the jth and $j + 1$st terms differ by more than 1.

$$(j + 1 + \lceil \lg r - \lg\lfloor \lg 4(j + 1) \rfloor \rceil - \lceil \lg r \rceil) - (j + \lceil \lg r - \lg\lfloor \lg 4j \rfloor \rceil - \lceil \lg r \rceil) > 1.$$

Simplifying,

$$\lceil \lg r - \lg\lfloor \lg 4(j + 1) \rfloor \rceil - \lceil \lg r - \lfloor \lg 4j \rfloor \rceil > 0$$

or

$$\lceil \lg r - \lg\lfloor \lg 4(j + 1) \rfloor \rceil > \lceil \lg r - \lg\lfloor \lg 4j \rfloor \rceil.$$

Since $\lceil a - b \rceil > \lceil a - d \rceil$ implies $b < d$ for positive a, b, and d,

$$\lg\lfloor \lg(4(j + 1)) \rfloor < \lg\lfloor \lg(4j) \rfloor$$

provided that

$$r \geq \lg(4(j + 1)). \tag{2}$$

If (2) holds, $\lg 4(j + 1) < \lg(4j)$ implies that $4(j + 1) < 4j$ which is a contradiction. Thus Lemma 4 holds provided (2) is true. Now suppose that $r < \lg(4(j + 1))$. Then

$$r \leq \lg(4(j + 1)) - 1.$$

So

$$r + 1 \leq \lg(4(j + 1))$$

and

$$2^{r-1} \leq j + 1 \leq k$$

which is a contradiction since (r, k) is a valid pair. Therefore, Lemma 4 holds. □

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

$$p + \left\lceil \lg \frac{r}{n(p)} \right\rceil = k.$$

Then an (r, k)-*THIN tree* 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 exact shape of the binary trees in the forest is not specified. While a forest which leads to a smallest (r, k)-THIN tree can be characterized (see [2]), the analysis in this paper is limited to finding a bound on S_w/S_o; we consider a subset of THIN trees for that reason. Lemma 5 establishes conditions on r and k which allow one to find a worst case bound on the ratio of sizes of an (r, k)-FAT tree to the size of an (r, k)-THIN tree.

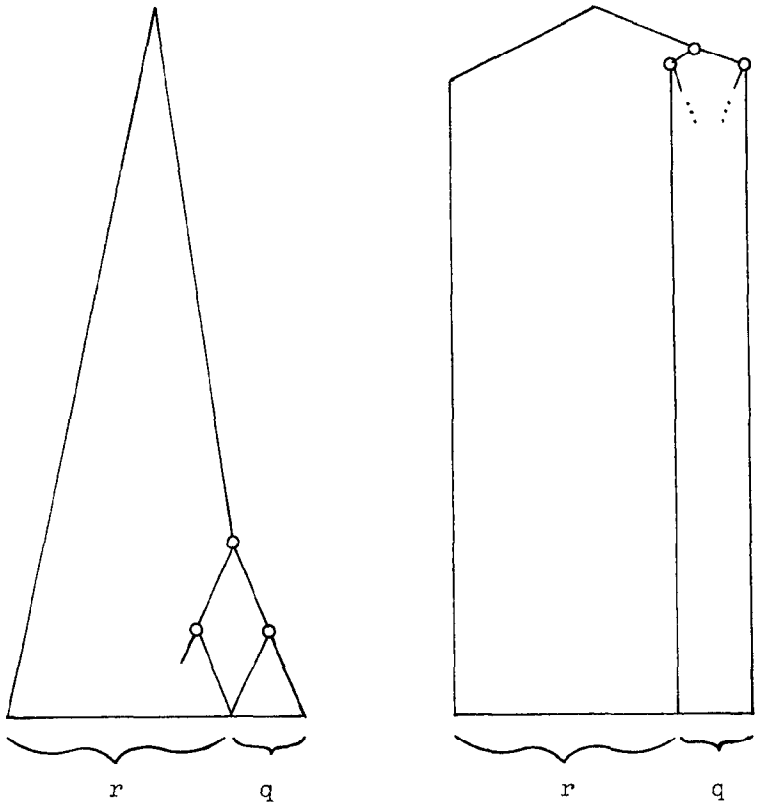


Fig. 7. An (r, k)-THIN tree, T, and an (r, k)-FAT tree that have been extended to an $(r + q, k')$-THIN tree and an $(r + q, k')$-FAT tree. q is the minimum number of leaves necessary to complete a binary subtree of T.

LEMMA 5. *Consider the set of ratios*

$$R = \{|F|/|T| \mid (r, k) \text{ is a valid pair, } F \text{ is an } (r, k)\text{-FAT tree, and } T \text{ is an } (r, k)\text{-THIN tree}\}.$$

If R achieves a maximum then it does so for a pair (r, k) such that all subtrees in the forest of the (r, k)-THIN tree are complete binary subtrees.

PROOF. Suppose not. Then there exists a valid pair, (r, k), T, an (r, k)-THIN tree, and F, an (r, k)-FAT tree, such that $|F|/|T|$ is maximum, but not all subtrees in the forest of T are complete. We show that there exists a valid pair, (r', k') producing a THIN tree, T', in which all subtrees are complete binary trees, and a FAT tree F' such that $|F'|/|T'| > |F|/|T|$. Hence, a contradiction arises.

Consider T and F as shown in Figure 7. Let q be the number of additional leaves necessary to complete the smallest binary subtree of T that is not complete. Clearly, $q \leq r$ or all subtrees would be complete.

Let $r' = r + q$, and let $k' = r + 1 + \lg r$. The pair (r', k') is valid for $r > 3$ (the

cases $r = 2$ and $r = 3$ hold by inspection). Now consider F', an (r', k')-FAT tree and T', an (r', k')-THIN tree. The size of T' is the size of T plus the $q - 1$ internal nodes added in the complete subtree of q leaves. The size of F has been increased by at least $q(k' - \lceil \lg(q + r) \rceil)$, so the ratio $|F'|/|T'|$ can be bounded from below by

$$\frac{|F'|}{|T'|} \geq \frac{|F| + q(k' - \lceil \lg(q + r) \rceil)}{|T| + q - 1}$$
$$\geq \frac{|F| + q(k' - \lceil \lg(q + r) \rceil)}{|T| + q}.$$

To prove that $|F'|/|T'| > |F|/|T|$, it suffices to show that

$$X = \frac{|F'| + q(k' - \lceil \lg(q + r) \rceil)}{|T| + q} > \frac{|F|}{|T|}$$

$$X \geq \frac{|F| + q(k' - \lg(q + r))}{|T| + q}$$

$$\geq \frac{|F| + q(k' - \lg(2r))}{|T| + q}$$

$$= \frac{|F| + q(r + \lg r + 1 - \lg r - 1)}{|T| + q}$$

$$= \frac{|F| + qr}{|T| + q}.$$

From Figure 1 and the fact that F is a binary tree we have

$$\frac{|F|}{|T|} < r.$$

So

$$\frac{|F| + qr}{|T| + q} = \frac{|T|\,|F| + |T|rq}{|T|(|T| + q)}$$

$$> \frac{|T|\,|F| + |F|q}{|T|(|T| + q)}$$

$$= \frac{|F|(|T| + q)}{|T|(|T| + q)} = \frac{|F|}{|T|}.$$

Repeated application of the above transformation will eventually result in a THIN tree with complete binary subtrees. Since each transformation increases the ratio of sizes of FAT and THIN trees, Lemma 5 holds. □

Lemma 5 limits the tries one needs to consider in order to obtain a worst case bound on the ratio S_w/S_o. Theorem 1 summarizes the importance of (r, k)-THIN trees of this limited form. The next section demonstrates a class of files for which there exist (r, k)-THIN tries and (r, k)-FAT tries.

THEOREM 1. *Let (r, k) be a valid pair of integers and let T be an (r, k)-THIN tree in which the trees of the forest are all complete binary trees of the same size. Let F be a binary restricted file of r records and k attributes, 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, A has fewer leaves than T. This is a contradiction.

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

6. A BOUND ON S_w/S_o FOR BINARY RESTRICTED FILES

This section provides a bound on the ratio of the size of an (r, k)-FAT tree, S_w, to an (r, k)-THIN tree, S_o. The 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 next section shows that there exists an infinite class of files for which the ratio of best to worst tries achieves the bound.

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 $i = 2^{n-1} - 1$ can be obtained by summing the number of nodes at each level.

$$\begin{aligned} S_o &= 1 + 2*2^0 + 3*2^1 + 4*2^2 + \cdots + n*2^{n-2} \\ &= 1 + \sum_{i=2}^{n} i*2^{i-2} = \frac{1}{2} + \frac{1}{4}(n2^{n+2} - (n+1)2^{n+1} + 2). \end{aligned}$$

Simplifying, we get

$$\frac{1}{2}((n-1)2^n + 1) + \frac{1}{2} = (n-1)2^{n-1} + 1.$$

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 is then

$$S_o = (n-1)2^{n-1} + 1 + r - 2n \tag{3}$$

where n is the number of leaves in the i-STEM. The definition of THIN trees provides the exact relation between r, k, and n. Throughout the remainder of this paper n is used to refer to the number of subtrees in the forest of a THIN tree. Observe that for the THIN trees with complete forests, $r = n2^t$, where t is the

height of the trees in the forest. From Lemma 3, the i-STEM in a THIN tree has $\lfloor \lg i \rfloor + 2 = \lfloor \lg 4i \rfloor$ leaves, and so it must hold that $k = t + i \simeq t + 2^{n-1} - 1$.

The size of an (r, k)-FAT tree (excluding the leaves) can easily be computed since it consists of a complete binary tree of $\lfloor \lg r \rfloor$ levels of a complete binary tree followed by $k - \lfloor \lg r \rfloor - 1$ levels at which exactly r nodes appear. Let $p = \lfloor \lg r \rfloor$. 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 above, there exist integers n and t such that $r = n2^t$, so

$$\begin{aligned} S_w &= 2^{t+1+\lfloor \lg n \rfloor} - 1 + n2^t(2^{n-1} - 2 - \lfloor \lg n \rfloor) \\ &\leq 2^t(2n - 1 + n(2^{n-1} - 2 - \lfloor \lg n \rfloor)). \\ &\quad 2^t(n(2^{n-1} - \lfloor \lg n \rfloor) - 1). \end{aligned} \tag{4}$$

From eqs. (3) and (4), one can obtain a bound on the worst case performance of any heuristic. Since $2^{n-1} \gg \lg n$,

$$\begin{aligned} S_w &\leq 2^t(n2^{n-1} - 1) = r2^{n-1} - 2^t \\ &= r2^{n-1} - \frac{r}{n} = r\left(2^{n-1} - \frac{1}{n}\right) \leq r2^{n-1} \end{aligned}$$

therefore,

$$\frac{S_w}{S_0} \leq \frac{r2^{n-1}}{(n-1)2^{n-1} + r - 2n + 1}. \tag{5}$$

This ratio is approximately $r/(n-1)$ for large r and n. Since $r/(n-1) \geq 2^t$, S_w/S_0 is not bounded above by a constant, but grows as the size of the input file.

Files of the form shown in Figure 8 demonstrate that the worst case bound is achievable.

Definition. Let n, t be positive integers. An *(n, t)-WC file* is a binary file of $r = n2^t$ records and $k = t + 2^{n-1} - 1$ attributes constructed as shown in Figure 8.

Lemma 6 shows that tries exist for an (n, t)-WC file that are (r, k)-THIN trees and (r, k)-FAT trees.

LEMMA 6. *Given n, $t > 1$, and F an (n, t)-WC file, there exist full tries T and A indexing F which are an (r, k)-THIN tree and an (r, k)-FAT tree, respectively.*

PROOF

a. (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 from set Q left to right, producing 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 set P left to right dividing up these sets, doubling the number of divisions at each depth. T will consist of n subtrees of 2^t leaves each, where each subtree is a complete binary tree rooted in one of the leaves of the $(2^{n-1} - 1)$-STEM. By definition, T is an (r, k)-THIN tree for appropriate r and k.

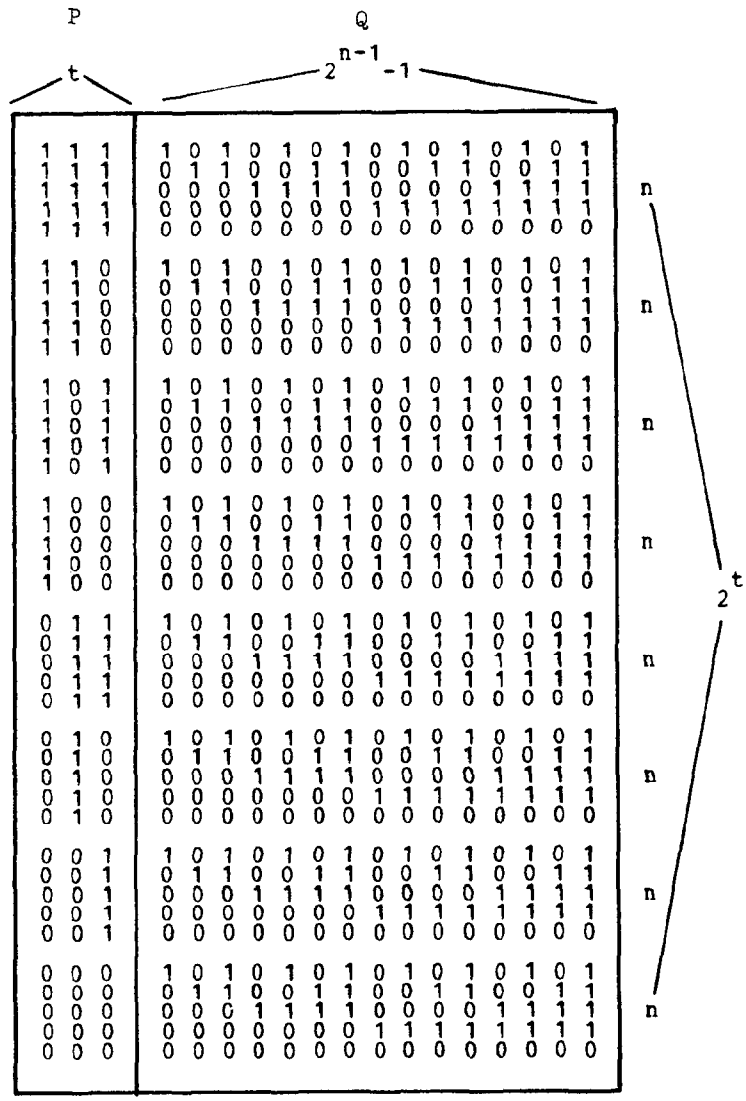


Fig. 8. The construction of an (n, t)-WC file. The example file has $n = 5$ and $t = 3$.

b. (For A). Construct A in the following way. Choose the t attributes from set P left to right producing a complete binary tree of depth t. Exactly one record from each of the n blocks will be associated with each leaf in this part of the tree. Since the blocks are all identical, they each contain all $2^{n-1} - 1$ possible attribute values; select those attributes which divide the sets of records represented in half, then in quarters, and so on. These selections will form complete binary subtrees rooted at each of the 2^t nodes formed by selections in q. That is, the selections lead to a complete binary tree with r leaves. Selection of the remaining attributes in any order will produce an (r, k)-FAT tree. □

7. A WORST CASE BOUND FOR THE GREEDY HEURISTIC

This section shows that the greedy heuristic may produce relatively large files when presented with an (n, t)-WC file. The choice of the first attribute turns out to be crucial; a correct choice will lead the greedy heuristic to an optimum (THIN) trie, but a bad choice may lead to a tree which is almost as large as the worst (FAT) tree. The details are given in Theorem 2.

THEOREM 2. *Let $n, t > 1$, 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 6 there exists an (r, k)-THIN trie indexing F, so S_o is given by the size of an (r, k)-THIN tree. Now consider the selections which can lead greedy to a worst case trie. Referring to Figure 8, form a trie as follows: Choose the t attributes from set P left to right, dividing the records into 2^t sets. As in the (r, k)-FAT trie, a complete binary tree will be formed. Following the selections from set P, continue to select attributes from set Q left to right. The selections from Q form 2^t $(2^{n-1} - 1)$-STEMS rooted in the 2^t nodes at depth t. The end result is a modified (r, k)-FAT trie in which the first levels grow as rapidly as possible, but later levels grow more slowly.

The greedy heuristic allows the selections described above. Since there are no trivial attributes in the file, any attribute may be selected first; its choice is important. If the leftmost attribute is selected, any second attribute is allowed because each will increase the breadth of the trie by two. If part of the attributes from set P have been selected left to right, the next attribute in P can be selected by the greedy heuristic because any remaining attribute choice will split each node. After all selections in P are complete, the greedy heuristic will choose the "best" order for attributes in set Q, producing a STEM for each subtree.

Analysis which follows shows that the size of the modified (r, k)-FAT tree described by the above selection is such that S_h/S_o approaches S_w/S_o asymptotically. Therefore, Theorem 2 holds. □

We now consider the size of the modified (r, k)-FAT tree produced by the greedy heuristic. As shown in Figure 9, the difference between the tree in question and an (r, k)-FAT tree is that at depth t the modified trie stops exponential growth and has 2^t i-STEMS as subtrees. The (r, k)-FAT tree, on the other hand, continues with complete binary subtrees until all r records have been distinguished. So the difference between a FAT tree and the modified FAT tree occurs only in the 2^t subtrees. The point to note is that the subtrees of the (r, k)-FAT tree are themselves $(r/n, k - t)$-FAT trees while the subtrees of the trie are $(k - t)$-STEMS. Lemma 7 shows that the ratio of the size of an (r, k)-FAT tree to the size of a k-STEM approaches 1 for large r and k. Thus the cost of the greedy heuristic asymptotically approaches the worst cost possible.

LEMMA 7. *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 1 for large r and appropriate k, where $|T|$ denotes the size of T.*

PROOF. From the preceding analysis, we have that

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

and

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

and for $k = 2^{r-1} - 1$,

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

So

$$\frac{|T|}{|S|} = \frac{r(2^{r-1} - \lceil \lg r \rceil - 1) - 1}{(r-1)2^{r-1} + 1}$$
$$\leq \frac{r(2^{r-1} - \lceil \lg r \rceil)}{(r-1)2^{r-1}}$$

and for $r \gg 1$,

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

which approaches 1 as r grows. □

We conclude that a k-STEM is approximately the size of an (r, k)-FAT tree for large r and k. This result is intuitively unappealing. In a sense it claims that for a file of r records and k attributes the slowest growing trie, a k-STEM, and the fastest growing trie, an (r, k)-FAT tree, are approximately the same size. To see why this happens, observe that the last $k/2$ levels of the k-STEM have r nodes. So for large k, a k-STEM is at least one-half the size of an (r, k)-FAT tree. Of the remaining levels, $k/4$ of them have $r - 1$ nodes, $k/8$ have $r - 2$, and so on. Thus, while a k-STEM grows slowly, it has many levels which are almost as wide as an (r, k)-FAT tree when k is large.

The above arguments show that when $k \simeq 2^{r-1}$, the largest and smallest tries are close to the same size. It is interesting to note that the opposite extreme, when $k \simeq \lg r$, has the same property. One can visualize a THIN tree being stretched and compressed as k changes. As k decreases, the THIN tree shrinks but the binary subtrees expand upward. Eventually, k reaches $\lg(r)$ and there is only one possible trie: the complete binary tree. As k increases, the THIN tree grows longer while the subtrees grow smaller until only a k-STEM remains. The cost of a heuristic will be maximized for $k \simeq r$, and will decrease to 1 for very large or very small k.

8. SUMMARY AND CONCLUSIONS

The problem of constructing a minimum size trie by reordering attributes is computationally difficult. This paper has explored a simple heuristic procedure for trie construction, called greedy, which employs a local optimization in an attempt to generate low cost tries. The greedy heuristic is a member of a class of heuristics which all produce optimum tries for files with all possible attribute combinations. Experimental results show that the greedy heuristic tends to produce tries with smaller average size than randomly chosen tries. For some

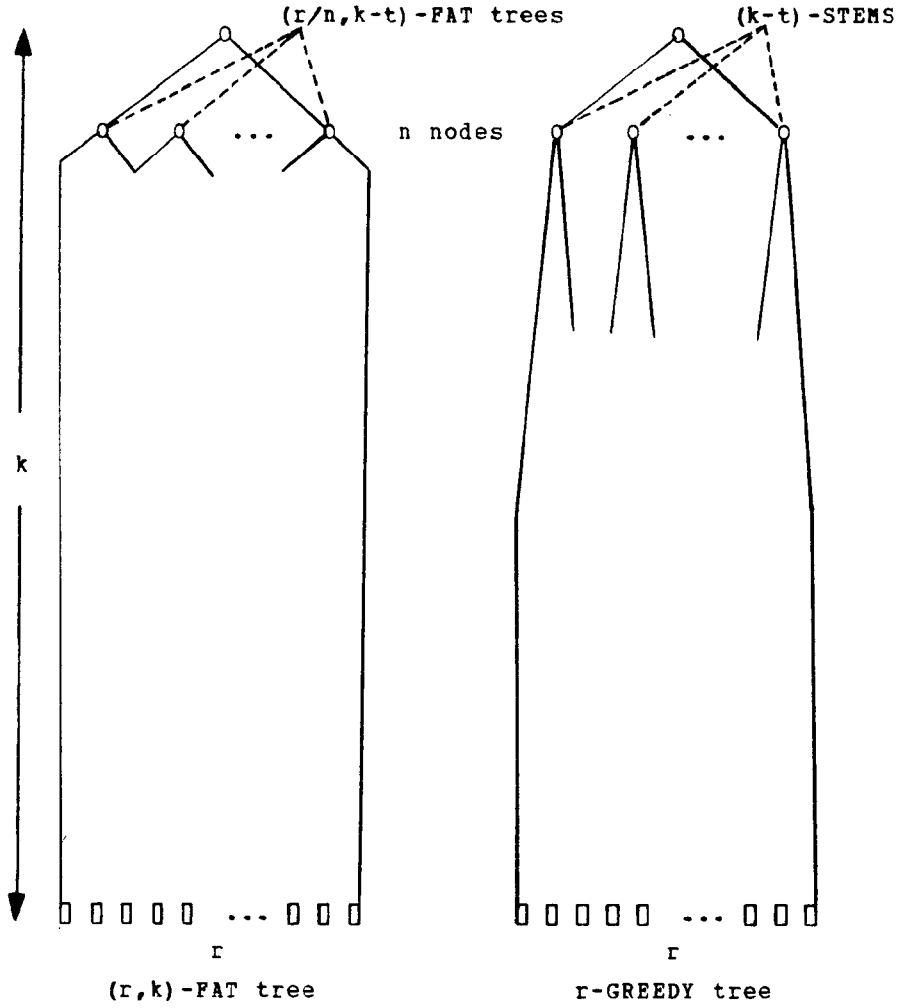


Fig. 9. The shape of a largest (FAT) trie, and the shape of a worst case trie for the greedy heuristic. The greedy trie differs from the FAT trie in that there are n subtrees which are $(k - t)$-STEMS instead of n FAT subtrees.

files, the greedy method may produce a nonoptimum trie, however, because it cannot guarantee success.

The paper defines a class of restricted files which have no trivial or isomorphic attributes, and characterizes the largest and smallest tries for binary restricted files. From the size of the largest and smallest tries, we obtained a bound on the ratio of the size of the worst trie to the size of the best possible trie for a binary restricted file, and demonstrated a class of files which obtained that ratio.

Analysis of the greedy heuristic shows that it could produce tries which approach the worst case. We conclude that the greedy heuristic can perform badly, even though it seems to do well on many files.

Several problems remain open. First, the analysis here concentrates on the binary case. Extensions to files of greater alphabet size would be interesting. Second, the worst case occurs when the greedy heuristic is presented with a set of attributes which all have equal branching. It seems that a more sophisticated technique which employs bounded lookahead might pay off dramatically in lowering the worst case cost. For example, a lookahead of 3 for WC files would force greedy to choose the best attribute order. Third, the worst case occurs when k is almost equal to r. Since in practice one would expect k to be closer to $(\lg r)^2$, it would be interesting to know whether restrictions on k would change the bound significatly. Last, only a static file has been considered. A host of questions relating to updates can be asked. Given a file and an optimum trie, how does the trie deviate from the optimum after a sequence of "delete" or "insert" operations? Phrased another way, at what time during a sequence of updates would it be cost effective to reorganize the index?

ACKNOWLEDGMENT

The author wishes to thank the referees for several helpful suggestions.

REFERENCES

1. AHO, A., HOPCROFT, J., AND ULLMAN, J. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.
2. COMER, D. Trie structured index minimization. Ph.D. Dissertation, The Pennsylvania State Univ., University Park, Pa., 1976.
3. COMER, D. Heuristics for trie index minimization. *ACM Trans. Database Syst. 4,* 3 (Sept. 1979), 383–395.
4. COMER, D., AND SETHI, R. The complexity of trie index construction. *J. ACM 24,* 3 (July 1977), 428–440.
5. DEMAINE, P.A.D., AND ROTWITT, T. Storage optimization of tree structured files representing descriptor sets. In *Proc. ACM SIGFIDET Workshop on Data Description, Access, and Control,* Nov. 1971, pp. 207–217.
6. FREDKIN, E. Trie memory. *Commun. ACM 3,* 9 (Sept. 1960), 490–499.
7. YAO, S.B. A model for combined attribute index organizations. In *Proc. 5th Texas Conf. Computing Systems,* Austin, Tex., Oct. 1976, pp. 127–130.

Received April 1978; revised September 1980; accepted November 1980