ON THE OPTIMALITY OF ELIAS'S ALGORITHM FOR PERFORMING BEST-MATCH SEARCHES

Ronald L. RIVEST

INFORMATION PROCESSING 74 – NORTH-HOLLAND PUBLISHING COMPANY (1974)

ON THE OPTIMALITY OF ELIAS'S ALGORITHM FOR PERFORMING BEST-MATCH SEARCHES

Ronald L. RIVEST

IRIA-LABORIA
Domaine de Voluceau
78150, Rocquencourt, France

We examine a hash-coding algorithm due to Elias for retrieving from a file of fixed-length binary records all the "best-matches" (using the Hamming distance) to a given input word. We prove that no balanced hash-coding algorithm can examine fewer buckets on the average than Elias's algorithm does, assuming that every record is equally likely to appear in the file.

1. INTRODUCTION.

We consider hash-coding algorithms for retrieving from a file of k-bit words all the "best-matches" to a given input word x (under the Hamming distance metric). An algorithm based on error-correcting codes due to Prof. Peter Elias of MIT (according to [7]) is shown to minimize the expected number of buckets examined, over all "balanced" hash-coding algorithms, if every record is independently equiprobable to appear in F. (Burkhard and Keller have apparently independently rediscovered the idea [1]). We also analyze its efficiency for a sample application.

We shall use the following notation. Let R_k denote the set of all the 2^k different k-bit words (or records) so that our file F is a nonempty subset of R_k. We restrict our attention to binary records only to simplify the analysis ; similar results presumably hold for nonbinary records. For $x \in R_k$ let x(i) denote the i-th bit of x, for $1 \le i \le k$. Let d(x,y) denote the Hamming distance between words $x,y \in R_k$; this is the number of bit positions i such that $x(i) \neq y(i)$. The set $\beta(x,F)$ of best-matches (or nearest neighbors) to x in F is defined to be the set of all records $y \in F$ which have minimum distance d(x,y). That is,

$$\beta(x,F) =_{\text{def}} \{y \in F \mid \neg(\exists z \in F)\ d(x,z) < d(x,y)\}. \quad (1)$$

Note that $\beta(x,F)$ is {x} if $x \in F$. The complement of $x \in R_k$ is denoted by $\bar{x}$; we have $\bar{x}(i) \neq x(i)$ for $1 \le i \le k$. If the binary number denoted by x is less than that denoted by y, we write $x < y$.

Our problem is to organize the file F on a random-access memory so that the average time to calculate $\beta(x,F)$ is small, averaging over all records $x \in R_k$. This problem models a large number of practical applications. For example, a typical pattern recognition problem fits this schema : the input word x is known to be a noisy version of some word in F and we wish to guess which one (the "decoding" problem). One might want, say, to organize a dictionary so as to be able to correct mistyped characters. Viewed as an "associative" retrieval problem, we want to calculate the "nearest associates" $\beta(x,F)$ to the input query x. We restrict ourselves here to the best-match retrieval problem ; the "partial-match" retrieval problem (finding all $y \in F$ agreeing with a partially specified query x) is also studied in the author's thesis [5].

One can clearly minimize the average search time by precomputing the response $\beta(x,F)$ to each query x, but this requires exorbitant amounts of storage and makes file modification difficult. For algorithms using reasonable amounts of storage, Prof. Marvin Minsky conjectured that "for large data sets with long word lengths there are no practical alternatives to large searches that inspect large parts of the memory" [3]. We show that hash-coding algorithms can often calculate $\beta(x,F)$ very quickly, and that Elias's hash-coding algorithm is in a certain sense the optimal hash-coding algorithm for our problem.

2. THE ALGORITHM.

We consider balanced hash-coding schemes utilizing chaining to handle collisions (see [2]). These schemes maintain b distinct lists (or buckets) $L_1, \ldots, L_b$ of records in F.

A record $x \in F$ is stored on list $L_{h(x)}$, where h is an auxiliary "hash" function mapping R_k onto {1, ..., b}. The b lists $L_1, \ldots, L_b$ are thus disjoint and satisfy $\cup_{1 \le i \le b} L_i = F$. Collisions (two or more records hashing to the same bucket address) are no problem since each list L_i can be of arbitrary length. We denote by B_i the set of records in R_k which might appear on list L_i. That is,

$$B_i =_{\text{def}} \{x \in R_k \mid h(x) = i\}, \quad (2)$$

so that $L_i = B_i \cap F$ for $1 \le i \le b$, $\cup_{1 \le i \le b} B_i = R_k$ and $B_i \cap B_j = \emptyset$ for $i \neq j$. We call B_i the block (of the partition $\{B_1, \ldots, B_b\}$ of R_k) corresponding to the list L_i. A hash function h is said to be balanced if $|B_i| = |R_k|/b$ for $1 \le i \le b$.

Hash-coding schemes are known to be very efficient for handling "exact match" queries (that is, determing whether the input word x is in F) -see [2], for example. Here we give the natural generalization for handling best-match queries. Let us extend our distance notation so that d(x,S) denotes the minimum distance from x to any point in set S (that is, $d(x,S) =_{\text{def}} \min_{y \in S} d(x,y)$). The idea is to examine L_i in order of increasing corresponding distances $d(x,B_i)$ until a nonempty L_{i_o} is found. The value $\delta = \min_{y \in L_{i_o}} d(x,y)$ then gives an upper bound for $d(x,\beta(x,F))$, the distance from x to its nearest neighbors in F. We then only need to look at those unexamined lists L_j such that $d(x,B_j) \le \delta$ in order to calculate $\beta(x,F)$. The upper bound δ can be decreased every time records in F closer than δ are found, in a typical dynamic programming fashion, further reducing the number of buckets yet to be examined. We now present the algorithm precisely, using the notation $x^{(\delta)}$ (respectively $S^{(\delta)}$) to denote the set of points $y \in R_k$ such that $d(y,x) = \delta$ (resp. $d(y,S) = \delta$), for points $x \in R_k$ and sets $S \subseteq R_k$.

```
procedure SEARCH (record x) ;
begin comment find all best-matches to x ;
  integer δ,j ; record set S ;
  comment S contains the current best-matches to x,
     at distance δ from x ;
   δ: = ∞; S := null ; j := -1 ;
  for j := j+1 while j ≤ min(δ,k) do
    for each i ∈ {1, ..., b} such that d(x,B_i) = j do
       begin comment examine list L_i ;
        if L_i ≠ null then δ := min(δ,d(x,L_i)) ;
        S := x^(δ) ∩ (S ∪ L_i) ;

       end ;
  print(x,δ,S) ;
  comment now S = β(x,F) and d(x,β(x,F)) = δ;
end.
```

The efficiency of the above algorithm depends on the hash function h used. A typical pseudo-random hash function would perform poorly since each block B_i could be expected to contain some word near to x, causing list L_i to be examined with high probability. It is more reasonable to make each block B_i a compact subset of R_k, so that $d(x,B_i)$ is large for many points $x \in R_k$. Elias suggests (according to [7]) using a (k,w) perfect error-correcting code with minimum distance $2\lambda + 1$ as a basis for a suitable hash function. Such a code is a subset C of R_k of size 2^w such that for any distinct pair c_i, c_j of codewords in C we have $d(c_i,c_j) \geq 2\lambda + 1$ and furthermore, for each $x \in R_k$ there is a unique codeword c_i with $d(x,c_i) \leq \lambda$. We define the hash function h associated with the code C by eq.(2) and $B_i = \{x \in R_k \mid d(x,c_i) \leq \lambda\}$ for $1 \leq i \leq b$, where $b = 2^w$. Each block B_i is a "sphere" of radius λ centered at c_i. This is intuitively appealing since records relevant to the same search are likely to be stored together. While perfect (k,w) error-correcting codes do not always exist (see [6]) we would expect nearly equivalent performance from the best (k,w) error-correcting code available.

The actual amount of work performed is also dependent on the particular file F being searched. We wish to consider the average work done over all queries x and files F, assuming that each query $x \in R_k$ is equally likely and that each file F of the same size is equally likely. More precisely, we assume that there is a probability p such that any record $x \in R_k$ appears in F with independent probability p. Thus the expected size of F is $p2^k$ and the probability that $|F|=n$ is $\binom{2^k}{n} p^n (1-p)^{2^k-n}$. We thus ignore second-order "correlation" effects between records.

We will only consider balanced hash functions, for which the average number of buckets examined is an accurate measure of the total work performed, since the expected size of each list is the same ($E(|L_i|) = p2^k/b$ for $1 \leq i \leq b$). We define our cost measure $\alpha(h)$ to be the average number of lists examined by the procedure SEARCH over all queries x and files F (using the preceding assumption about the probability distribution for F). The number of lists examined is a realistic cost measure when the file is stored on a secondary storage device such that the time to access an arbitrary list L_i exceeds the time to read it. Requiring the hash function to be balanced also eliminates consideration of such degenerate solutions as storing all records on a single list (so that $\alpha(h) = 1$).

3. A SAMPLE APPLICATION.

An an example, suppose we have a file F of approximately 2^{15} 23-bit words. If we use the Golay perfect $(k = 23, w = 12)$ error-correcting code with minimum distance $2\lambda + 1 = 7$ between codewords (see [4]), our hash function h will have $2^{12} = 4096$ buckets, each containing about 8 records. The following formula gives the average number of buckets examined to find all best-matches to an input query x.

$$\alpha(h) = \sum_{0 \leq i \leq 3} \binom{23}{i} 2^{-11} \sum_{0 \leq j \leq 23} v(j)\, \tau(i,j) \qquad (3)$$

Here we take as separate cases the distance i of x from the center of its bucket, the codeword $c_{h(x)}$; the factor $\binom{23}{i} 2^{-11}$ is the probability of this distance occurring. The factor v(j) is the probability that the nearest record to x in F is at distance j ; this is given by the formula

$$v(j) = (1 - (1-p)^{\binom{23}{j}}) \cdot \emptyset(k,j-1) \qquad (4)$$

We use V(k,j) to denote the number $\sum_{0 \leq i \leq j} \binom{k}{i}$ of points in a k-dimensional sphere with radius j and $\emptyset(k,j) = (1-p)^{V(k,j)}$ to denote the probability that it is empty (that is, it contains no points in F). Finally, the value of $\tau(i,j)$ in eq.(3) denotes the average number of blocks B such that $d(x,B) \leq j$, given that $d(x,c_{h(x)}) = i$; the relevant values of $\tau(i,j)$ were calculated with a computer program to obtain table 1 of $\alpha(h)$ versus p. For our application p is 2^{-8} ; we see that only 37 buckets need be examined on the average, or about 9 % of the file.

Table 1

Average number of buckets examined versus $\log_2(p)$

$\log_2(p)$	$\alpha(h)$
0	1.000
-1	3.162
-2	4.259
-3	5.450
-4	8.545
-5	12.86
-6	17.14
-7	24.51
-8	36.97
-9	51.85
-10	75.16
-11	109.8
-12	157.0
-13	227.5
-14	325.1
-15	463.5
-16	653.5
-17	909.8
-18	1244
-19	1665
-20	2153
-21	2612
-22	2701
$-\infty$ (p=0)	4096

4. OPTIMAL BUCKET SHAPES.

To prove the optimality of Elias's algorithm, we consider the probability $\Psi(B_i)$ that each list L_i will be examined. Note that this is necessarily only a function of the set B_i of records that could possibly appear on L_i, and not of L_i itself (since we don't know what L_i is until we examine it). Since $\alpha(h) = \sum_{1\le i\le b} \Psi(B_i)$, $\alpha(h)$ will be minimal if each $\Psi(B_i)$ is minimal.

Theorem. The probability $\Psi(B)$ that a list L will be examined during the search for all best-matches $\beta(x,F)$ of x in a file F is minimized over all blocks of size $|B|$ when the block B associated with list L is a "sphere" -that is, when B consists of a center $c \in R_k$ and all points y such that $d(c,y) \le \lambda$ for some λ.

Proof. List L will be examined if and only if the file F does not contain any words y such that $d(x,y) < d(x,B)$ (that is, any words y closer to x that the nearest word in B). Under our assumption that each word $y \in R_k$ is in F with probability p, the probability that L is examined is exactly $\emptyset(k,d(x,B)-1)$. Averaging over all queries $x \in R_k$ we have

$$\Psi(B) \underset{\text{def}}{=} 2^{-k} \sum_{x \in R_k} \emptyset(k,d(x,B)-1) \qquad (5)$$

$$= 2^{-k} \sum_{0\le\delta\le k} |B^{(\delta)}| \cdot \emptyset(k,\delta-1) \qquad (6)$$

(Remember that $B^{(\delta)}$ is the set of points x such that $d(x,B) = \delta$.) Since $\emptyset(k,\delta-1)$ is a decreasing function of δ, we can show that $\Psi(B) < \Psi(C)$ for any blocks B and C whenever the vector

$(|B^{(0)}|, |B^{(1)}|, \ldots, |B^{(k)}|)$ is lexicographically less that the corresponding vector for C (that is, whenever there is an m, $0\le m\le k$ such that $|B^{(\delta)}| = |C^{(\delta)}|$ for $0\le\delta<m$ but $|B^{(m)}| < |C^{(m)}|$). Since $|B^{(m)}| - |C^{(m)}| = \sum_{m<\delta\le k} (|C^{(\delta)}| - |B^{(\delta)}|)$ we have

$$\Psi(C) - \Psi(B) = 2^{-k}(|C^{(m)}| - |B^{(m)}|)\cdot\emptyset(k,m-1) \qquad (8)$$

$$+ 2^{-k} \sum_{m<\delta\le k} (|C^{(\delta)}| - |B^{(\delta)}|)\cdot\emptyset(k,\delta-1)$$

$$\ge 2^{-k}(|C^{(m)}| - |B^{(m)}|)(\emptyset(k,m-1) - \emptyset(k,m)) \qquad (9)$$

$$\ge 0 \qquad (10)$$

Since $|B^{(0)}| = |B|$, for two blocks B and C of the same size, the list associated with block B will be examined less frequently than C's list if $|B^{(1)}| < |C^{(1)}|$. Note that $|B^{(1)}|$ is the discrete analog of the surface area of a region B, so that what we would like to show is that a sphere has minimal "surface area".

We first extend our definition of a sphere to include subsets $S \subseteq R_k$ of size n, where n is not $V(k,\lambda)$ for any λ.

Definition. The sphere $S_{n,c}$ of size n about $c \in R_k$ is defined by

(i) $S_{1,c} = \{c\}$, and

(ii) $S_{n+1,c} = S_{n,c} \cup \{x\}$, where x is that point in $S_{n,c}^{(1)}$ with minimal pair $(\sigma_c(x), x \oplus c)$ where $\sigma_c(x)$ denotes $\min\{i \mid x \in S_{i,c}^{(1)}\}$, using a lexicographic ordering between pairs. Here $x \oplus c$ denotes the k-bit word obtained by performing a component-wise "exclusive-or" of x and c (this has d(x,c) ones in it, in the positions where x and c differ).

Note that either $S_{i,c} \subseteq (S_{j,c} \cup S_{j,c}^{(1)})$ or $(S_{j,c} \cup S_{j,c}^{(1)}) \subset S_{i,c}$ for all i,j. When $n = V(k,\lambda)$ for some λ the definition corresponds to the usual one, giving all points x within distance λ of c, since if $d(x,c) < d(y,c)$ then x will be reached before y.

We prove our theorem by induction on the length k of the words under consideration. For $k = 1$ all the nonempty subsets $\{0\},\{1\},\{0,1\}$ of R_1 are spheres ; since $\Psi(\{0\}) = \Psi(\{1\})$ trivially, the theorem holds.

Now suppose that we have an arbitrary block $B \subseteq R_k$ for $k \ge 2$. We show that B can be transformed into a sphere by a finite sequence of operations such that $B^{(1)}$ is nonincreasing at each step. Thus for all spheres S, $S^{(1)}$ will be minimal over all blocks of size $|S|$.

Let $B_{0,i}$ (respectively $B_{1,i}$) denote the subset of R_{k-1} obtained from B by taking each word $x \in B$ such that the $x(i) = 0$ (resp. $x(i) = 1$) and deleting this i-th bit. Thus $|B| = |B_{0,i}| + |B_{1,i}|$ and B is uniquely determined by $B_{0,i}$ and $B_{1,i}$. For each point $x \in R_k$, denote by $\varphi_i(x)$ the corresponding point in R_{k-1} obtained by deleting the i-th bit of x.

Suppose $B \subseteq R_k$ is not a sphere about the point c. Then at least one of following three cases hold :

(i) there is an i such that at least one of $B_{0,i}$, $B_{1,i}$ is not a sphere in R_{k-1} about $\varphi_i(c)$, or

(ii) there exists a pair of records $x \in B$, $y \in R_k - B$ such that $\sigma_c(x) > \sigma_c(y)$, but cases (i) and (iii) do not apply, or

(iii) there exists a pair of records $x \in B$, $y \in R_k - B$ such that $\sigma_c(x) = \sigma_c(y)$ and $x \oplus c > y \oplus c$, but case (i) does not apply.

Note that if (ii) or (iii) occur, then we must have $x = \bar{y}$ to exclude case (i) as well, since if $x(i) = y(i)$ then $\varphi_i(x)$ and $\varphi_i(y)$ are both in $B_{x(i),i}$, which is thus not a sphere about $\varphi_i(c)$. In addition the pair (x,y) must be unique, otherwise we could find another pair (x,y) having $x \ne \bar{y}$.

Our transformation of a nonspherical B depends on which of the three cases occurs. With (ii) or (iii) we merely set $B \leftarrow B \cup \{y\} - \{x\}$; in either case it is easy to verify that B immediately becomes a sphere and that $B^{(1)}$ is nonincreasing. For case (iii) to occur we must have $k = 2$; for case (ii) to occur we must have $y^{(1)} \subseteq B^{(1)}$, since y must either be the only point in $S^{(1)}_{\sigma_c(y),c}$ or must have the largest value $y \oplus c$ of any point in $S^{(1)}_{\sigma_c(y),c}$, so that all of y's neighbors have already been assigned their σ values.

When case (i) occurs, we modify B so that the corresponding sets $B_{0,i}$ and $B_{1,i}$ for the postulated i (at least one of which is nonspherical)) become spheres, conserving their sizes. To show that $|B^{(1)}|$ is nonincreasing we use the following fact.

$$|B^{(1)}| = |B^{(1)}_{0,i}| + |B_{1,i} - (B_{0,i} \cup B^{(1)}_{0,i})| + |B^{(1)}_{1,i}| + |B_{0,i} - (B_{1,i} \cup B^{(1)}_{1,i})| \qquad (11)$$

Remember that $B^{(1)}$ is a subset of R_k but that the sets on the right-hand side of (11) are all in R_{k-1}. The first two terms count those $x \in B^{(1)}$ such that $x(i) = 0$; the last two terms count those with $x(i) = 1$. The first and third terms count those $x \in B^{(1)}$ such that there exists a $y \in B$ with $x(i) = y(i)$; the other terms handle the case that the only $y \in B$ adjacent to x has $y(i) \neq x(i)$.

Making $B_{0,i}$ and $B_{1,i}$ to be spheres about $\varphi_i(c)$ means that $|B^{(1)}_{0,i}|$ and $|B^{(1)}_{1,i}|$ become minimal, by our induction hypothesis. If we assume (without loss of generality) that $|B_{0,i}| \geq |B_{1,i}|$, then our operation forces $B_{1,i} \subseteq B_{0,i}$, causing the second term $|B_{1,i} - (B_{0,i} \cup B^{(1)}_{0,i})|$ of (11) to drop to zero. Finally, either $B_{0,i} \subseteq (B_{1,i} \cup B^{(1)}_{1,i})$ occurs, in which case $|B^{(1)}|$ clearly becomes minimal, or else we get $(B_{1,i} \cup B^{(1)}_{1,i}) \subseteq B_{0,i}$. In this case, we get $|B^{(1)}| = |B^{(1)}_{0,i}| + |B_{0,i} - B_{1,i}|$, which is necessarily minimal among the set of blocks B for which $|B_{0,i}|$ and $|B_{1,i}|$ are at their current values. The value of $|B^{(1)}|$ is thus nonincreasing at each step.

To show that a sequence of the above operations will transform any block B of size n into the sphere $S_{n,c}$ in a finite number of steps, it is only necessary to note that the value $\sum_{x \in B} \{j | x \in (S_{j,c} - S_{j-1,c})\}$ is strictly decreasing, but bounded below by $\binom{n+1}{2}$. Thus $B = S_{n,c}$ after a finite number of steps. This proves that a spherical B has minimal surface area $B^{(1)}$.

Finally, we remark that $B \cup B^{(1)}$ is a sphere whenever B is a sphere, so that $(B \cup B^{(1)})^{(1)} = B^{(2)}$ is also minimal. By induction on the index j of the $|B^{(j)}|$'s, the vector $(|B^{(1)}|, |B^{(2)}|, \ldots, |B^{(k)}|)$ is lexicographically minimal over all blocks of size $|B|$, thus proving that the chance $\Psi(B)$ that a list L is examined is minimized when B is a sphere. □

Since Elias's algorithm has a spherical B_i for each list L_i, it minimizes the average number of buckets examined over all balanced hash-coding algorithms with the same block size.

5. CONCLUSION.

The preceding theorem shows that in a sufficiently idealized setting one may prove a data structure optimal for a given application. We have shown that the best strategy in certain situations for organizing a file for best-match retrieval is to divide the record space into spherical domains B_i, so that each subset L_i of the file (with $L_j \subseteq B_i$) is necessarily very compact and cluster-like. This result gives a certain amount of justification for selecting clustering and similar organizational methods for practical applications.

The results presented here raise the interesting question of what to do in slightly more general situations. It is probably possible to generalize this result to nonbinary records in a straightforward manner, for example, but changing the distance metric or modifying the assumption that each record is equally likely to appear in F gives a new problem for which no results are known. An exact determination of the number of buckets examined on the average would also be desirable, but seems to require determination of the behavior of the function τ described in §3.

ACKNOWLEDGMENT.

The author would like to thank professors Donald E. Knuth, Robert W. Floyd and David A. Klarner for their assistance and encouragement. The author was supported by NSF grant GJ - 922 during the development of these results, and by IRIA-LABORIA during the preparation of the manuscript. This work represents a portion of the author's Ph.D. thesis [5].

REFERENCES.

[1] W.A. Burkhard and R.M. Keller, Some approaches to best-match file searching, CACM 16, April 1973, 230-236.

[2] Donald E. Knuth, The art of computer programming vol.3 (sorting and searching), Addison-Wesley, 1972.

[3] Marvin Minsky and Seymour Papert, Perceptrons : an introduction to computational geometry, The MIT Press, Cambridge, Mass, 1969.

[4] William W.Peterson and E.J. Weldon, Error-correcting codes, MIT Press, 1972.

[5] Ronald L. Rivest, Analysis of associative retrieval algorithm, Ph.D. thesis, Stanford University, 1974.

[6] Aimo Tietäväinen, On the non-existence of perfect codes over finite fields, SIAM J. Appl. Math. 24, Jan. 1973, 88-96.

[7] Terry Welch, Bounds on the information retrieval efficiency of static file structures, Project MAC Report MAC-TR-88, MIT, June 1971.

$$|S^{(i)}| = |B_{0,i}^{(i)}| + |B_{1,i} - (B_{0,i} \cup B_{0,i}^{(i)})|$$

$$+ |B_{1,i}^{(i)}| + |B_{0,i} - (B_{1,i} \cup B_{1,i}^{(i)})| \qquad (11)$$

Remember that $S^{(i)}$ is a subset of B_k but that the sets on the right-hand side of (11) are all in B_{k-1}. The first two terms count those $x \in S^{(i)}$ such that $x(i) = 0$; the last two terms count those with $x(i) = 1$. The first and third terms count those $x \in S^{(i)}$ such that there exists a $y \in B$ with $x(i) \neq y(i)$; the other terms handle the case that the only $y \in B$ adjacent to x has $y(i) \neq x(i)$.

Making $B_{0,i}$ and $B_{1,i}$ to be spheres about e_i means that $|B_{0,i}^{(i)}|$ and $|B_{1,i}^{(i)}|$ become minimal, by our induction hypothesis. If we assume (without loss of generality) that $|B_{0,i}| \geq |B_{1,i}|$, then our operation forces $B_{1,i} \subseteq B_{0,i}$, causing the second term $|B_{1,i} - (B_{0,i} \cup B_{0,i}^{(i)})|$ of (11) to drop to zero. Finally, either $B_{0,i} \subseteq (B_{1,i} \cup B_{1,i}^{(i)})$ obtains, in which case $S^{(i)}$ clearly becomes minimal, or else we get $B_{1,i} \cup B_{1,i}^{(i)} \subseteq B_{0,i}$; in this case, $|S^{(i)}| = |B_{0,i}^{(i)}| + |B_{0,i} - B_{1,i}|$, which gets its value minimal among the set of blocks B for which $|B_{0,i}|$ and $|B_{1,i}|$ are at their current values. The value of $|B^{(i)}|$ is thus non-increasing at each step.

To show that a sequence of the above operations will transform any block B of size n into the sphere $S_{n,k}$ in a finite number of steps, it is only necessary to note that the value [illegible] $x \in B$ is strictly increasing, but bounded [illegible]. Thus $B = S_{n,k}$ after a finite number of steps. This proves that a sphere [illegible] has minimal surface area $B^{(1)}$.

Finally, we remark that $B \cup B^{(1)}$ is a sphere whenever B is a sphere, so that $(B \cup B^{(1)})^{(1)} = B^{(2)}$ is also minimal, by induction on the index [illegible] the $|B^{(i)}|$. The vector $(|B^{(1)}|, |B^{(2)}|, \ldots, |B^{(k)}|)$ is lexicographically minimal over all blocks of size $|B|$, thus proving that the chance $\Psi(B)$ that [illegible] is a minimized is minimized when B is a sphere.

Since Elias's algorithm has a sphere as L_i for each list L_i, it minimizes the average number of buckets examined over all balanced [illegible] algorithms with the same bucket size.

5. CONCLUSION.

The preceding theorem shows that [illegible] the idealized setting one may prove a data structure optimal for a given application. We have shown that the best strategy in certain situations for arranging a file for best-match retrieval is to divide the record space into spherical domains L_i, so that each subset L_i [illegible] (with [illegible]) is necessarily very compact and clustered. This result gives a certain amount of justification for selecting clustering and similar organizational methods for practical applications.

The results presented here leave the interesting question of what to do in slightly more general situations. It is probably possible to generalize this result [illegible] in a straightforward manner, for example, [illegible] changing the distance metric or modifying the assumption that each record is equally likely to appear in F gives a new problem for which no results are known. An exact determination of the number of buckets examined on the average would also be desirable, but seems to require determination of the behavior of the function r described in §3.

ACKNOWLEDGMENT.

The author would like to thank professors Donald E. Knuth, Robert W. Floyd and David A. Klarner for their assistance and encouragement. The author was supported by NSF grant GJ - 922 during the development of these results, and by IRIA-LABORIA during the preparation of the manuscript. This work represents a portion of the author's Ph.D. thesis [5].

REFERENCES.

[1] W.A. Burkhard and R.M. Keller, Some approaches to best-match file searching, CACM 16, April 1973, 230-236.
[2] Donald E. Knuth, The art of computer programming vol.3 (sorting and searching), Addison-Wesley, 1973.
[3] Marvin Minsky and Seymour Papert, Perceptrons : an introduction to computational geometry, The MIT Press, Cambridge, Mass, 1969.
[4] William W. Peterson and E.J. Weldon, Error-correcting codes, MIT Press, 1972.
[5] Ronald L. Rivest, Analysis of associative retrieval algorithms, Ph.D. thesis, Stanford University, 1974.
[6] Aimo Tietäväinen, On the non-existence of perfect codes over finite fields, SIAM J. Appl. Math. 24, Jan. 1973, 88-96.
[7] Terry Welch, Bounds on the information retrieval efficiency of static file structures, Project MAC Report MAC-TR-88, MIT, June 1971.