Should Tables Be Sorted?

ANDREW CHI-CHIH YAO

Stanford University, Stanford, California

ABSTRACT Optimality questions are examined in the following information retrieval problem. Given a set S of n keys, store them so that queries of the form, "Is $x \in S$?" can be answered quickly It is shown that in a rather general model including all the commonly used schemes, $\lceil \lg(n + 1) \rceil$ probes to the table are needed in the worst case, provided the key space is sufficiently large The effects of smaller key space and arbitrary encoding are also explored

KEY WORDS AND PHRASES information retrieval, lower bound, optimality, query, Ramsey's theorem, search strategy, sorted table

CR CATEGORIES. 3 74, 4 34, 5 25, 5 31

1. *Introduction*

Given a set S of n distinct keys from a key space $M = \{1, 2, \ldots, m\}$, a basic information retrieval problem is to store S so that *membership* queries of the form, "Is j in S?" can be answered quickly. Two commonly used schemes are the sorted table and the hash table. In the first case, a query can be answered in $\lceil \lg(n + 1) \rceil$ probes by means of a binary search.[1] The hash table scheme has a good average-case cost, but requires $O(n)$ probes in the worst case for typical hashing schemes. Looking at various alternative methods, one gets the feeling that $\sim\log n$ probes must be necessary in the worst case, if the key space M is large and we only use about minimal storage space. Our purpose is to study the truth of this statement. The question is nontrivial, as the existence of hashing suggests the possibility of schemes drastically different from, and perhaps superior to, the sorted table.

Before presenting technical results, let us try to put the subject of this paper in perspective. In the literature, efficient methods have been devised to perform various primitives in data manipulations [1, 8, 20]. For example, a sequence of n "DELETE," "INSERT," "MIN" instructions can be performed in $O(n \log n)$ time. In recent years, lower bounds to the complexity of these problems have also begun to receive attention (e.g., [7, 9, 15, 18]). Since efficient data structures may utilize the full power of a random-access machine (e.g., [20]), it is of special interest to study the complexity problems in general models that are equipped with some address-computing capabilities. This paper is one step in that direction, studying perhaps the simplest of such data structuring problems. It is hoped that one can derive interesting results for other problems in similar frameworks. (For related study regarding bitwise-random-access machines, see [5, 6, 10].)

[1] lg denotes logarithm with base 2

This research was supported in part by the National Science Foundation under Grant MCS 77-05313

Author's address: Department of Computer Science, Stanford University, Stanford, CA 94305

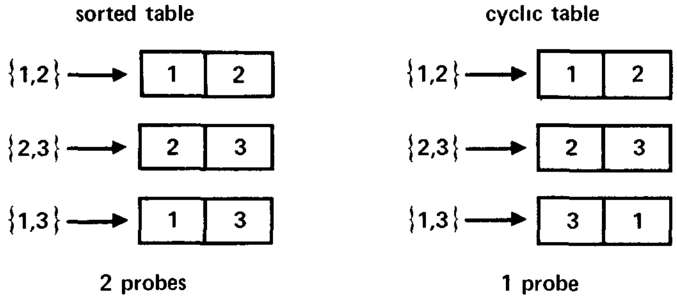

FIG 1 The sorted table is not optimal for $n = 2$, $m = 3$

2. *The Wisdom of Using Sorted Tables*

In this section we show that for large key space, $\lceil \lg(n + 1) \rceil$ probes are required to answer the membership problem in a rather general model. This model encompasses all common schemes such as hashing, sorted tables, and linked list structures. For clarity, we first prove the result in a simplified model. The general result will be given in Theorem 1′.

The Basic Model. Let the key space be $M = \{1, 2, \ldots, m\}$. We are interested in storing n distinct keys of M into a table of size n. A *table structure* $\mathscr{T}$ specifies how any particular set of n keys are to be placed in the table T. A *search strategy* $\mathscr{S}$ corresponding to $\mathscr{T}$ specifies, for any given key K, how to perform a series of probes $T(i_1) = ?$, $T(i_2) = ?$, ... into the table T, until one can claim whether K is in T or not. The search strategy is fully adaptive, in the sense that each probing location can depend on K and on all the previous probing results. The *cost* $c(\mathscr{T}, \mathscr{S})$ of a (table structure, search strategy) pair is measured by the number of probes needed in the worst case. The complexity $f(n, m)$ is the minimum cost achievable by any such pair. Clearly $f(n, m) \leq \lceil \lg(n + 1) \rceil$.

To get some feeling for possible improvements over the sorted table scheme and on its ultimate limitation, we look at the simple case $n = 2$, $m = 3$. It is easy to see that two probes are needed to decide whether $K = 2$ is in T if a sorted table is used. However, the "cyclic" table in Figure 1 allows us to answer any query in just one probe, as the first entry of T determines the entire table. Note that these are the only two nonisomorphic table structures (up to the renaming of keys and table locations) for this case.

Thus, a sorted table is not optimal for $n = 2$, $m = 3$. We now show, however, that a sorted table is optimal as soon as $n = 2$, $m = 4$ (hence for all $n = 2$, $m > 4$).

Any table structure for $n = 2$, $m = 4$ can be uniquely represented as a directed graph on four labeled vertices $\{1, 2, 3, 4\}$. We draw an edge $i \rightarrow j$ if the pair $\{i, j\}$ is stored as $\boxed{i \mid j}$. For example, the graph in Figure 2 represents a table structure with $\{1, 4\}$ stored as $\boxed{1 \mid 4}$ and $\{2, 4\}$ as $\boxed{4 \mid 2}$. For any three vertices in the graph, the edges between them may or may not form a directed cycle. It is not hard to show that for any such graph on four vertices, there exist three vertices among which the edges are acyclic. In Figure 2, $\{1, 3, 4\}$ is such a set of three vertices. If we consider the set of keys corresponding to these vertices as a subspace with $m = 3$, we find that we are storing these keys as a "permuted" sorted table, that is, one that differs from the sorted table only in a new ordering $3 \prec 1 \prec 4$ of the elements (Figure 3). But this means that any searching strategy for this table structure must make two probes in the worst case. This proves that $f(2, 4) \geq 2$, and hence the sorted table is optimal for $n = 2$, $m \geq 4$.

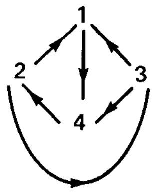

FIG 2 A typical table structure for $n = 2$, $m = 4$

3	1
1	4
3	4

FIG 3 The "permuted" sorted table corresponding to $\{1, 3, 4\}$ from Figure 2

The preceding statement generalizes to any fixed n; that is, the sorted table scheme is optimal for any fixed n, provided that the key space is large enough.

THEOREM 1. *For every n there exists an $N(n)$ such that $f(n, m) = \lceil lg(n + 1)\rceil$ for all $m \geq N(n)$.*

PROOF. We need the following lemma, which can be proved by an adversary argument.

LEMMA 1. *If a table structure stores the keys of a table in sorted order (or according to some fixed permutation), then $\lceil lg(n + 1)\rceil$ probes are needed in the worst case by any search strategy provided that $m \geq 2n - 1$ and $n \geq 2$.*

PROOF. We will construct an adversary strategy to show that $\lceil \lg(n + 1)\rceil$ probes are required to search for the key value $K = n$ of the space $\{1, 2, \ldots, m\}$. The construction is by induction on n. For $n = 2$ and $m \geq 3$ it is easy to see that two probes are required. Let $n_0 > 2$. Assume the induction hypothesis to be true for all $n < n_0$; we will prove it for $n = n_0$, $m \geq 2n_0 - 1$, and $K = n_0$. By symmetry, assume that the first probe position p satisfies $p \leq \lceil n_0/2\rceil$. The adversary answers $T(p) = p$. Then the key n_0 may be in any position i, where $\lceil n_0/2\rceil + 1 \leq i \leq n_0$. In fact, $T(\lceil n_0/2\rceil + 1)$ through $T(n_0)$ is a sorted table of size $n' = \lfloor n_0/2\rfloor$ which may contain any subset of $\{\lceil n_0/2\rceil + 1, \lceil n_0/2\rceil + 2, \ldots, m\}$, and hence, in particular, any subset of the key space $M' = \{\lceil n_0/2\rceil + 1, \lceil n_0/2\rceil + 2, \ldots, m - \lceil n_0/2\rceil\}$. The size m' of M' satisfies

$$\begin{aligned} m' &= m - 2\left\lceil \frac{n_0}{2} \right\rceil \geq (2n_0 - 1) - 2\left\lceil \frac{n_0}{2} \right\rceil \\ &\geq 2\left\lfloor \frac{n_0}{2} \right\rfloor - 1 = 2n' - 1, \end{aligned}$$

and the desired key n_0 has relative value $K' = n_0 - \lceil n_0/2\rceil = n'$ in the key space M'. By the induction hypothesis, $\lceil \lg(n' + 1)\rceil$ more probes will be required. Hence the total number of probes is at least $1 + \lceil \lg(n' + 1)\rceil = 1 + \lceil \lg(\lfloor n_0/2\rfloor + 1)\rceil \geq \lceil \lg(n_0 + 1)\rceil$. This completes the induction step. □

To prove Theorem 1, the idea is to show that if m is large enough, then for any table structure $\mathscr{T}$ there is a set S_0 of $2n - 1$ keys with the following property: Given any n-key subset $A \subseteq S_0$, the table structure always arranges the keys of A according to some fixed permutation. Lemma 1 will then imply the $\lceil \lg(n + 1)\rceil$ bound.

To this end, let us partition $\mathscr{A}$, the family of n-key subsets of M, into $n!$ parts as follows. For each $A = \{j_1 < j_2 < \cdots < j_n\} \in \mathscr{A}$, let T_A be the table formed under $\mathscr{T}$. We assign A to the group $\sigma(i_1, i_2, \ldots, i_n)$ if $T_A(i_1) = j_1$, $T_A(i_2) = j_2, \ldots, T_A(i_n) = j_n$. The collection $\{\sigma(i_1, i_2, \ldots, i_n) \mid (i_1, i_2, \ldots, i_n)$ is a permutation of $\{1, 2, \ldots, n\}\}$ forms a partition of $\mathscr{A}$.

CLAIM. *If m is sufficiently large, then there exists a set of $2n - 1$ keys $S_0 \subseteq \{1, 2, \ldots, m\}$ such that for all n-key subsets $A \subseteq S_0$ we have $A \in \sigma(i_1, i_2, \ldots, i_n)$, where $(i_1, i_2, \ldots, i_n)$ is a fixed permutation.*

By our earlier discussion this would imply Theorem 1. It remains to prove the claim. We make use of the following famous combinatorial theorem (see, e.g., [3]).

RAMSEY'S THEOREM. *For any k, r, t, there exists a finite number $R(k, r, t)$ such that the following is true. Let $S = \{1, 2, \ldots, m\}$ with $m \geq R(k, r, t)$. If we divide the family of all r-element subsets of S into t parts, then at least one part contains all the r-element subsets of some k elements of S.*

Our claim follows from Ramsey's Theorem by choosing $r = n$, $t = n!$ and $k = 2n - 1$. This proves Theorem 1 with $N(n) = R(2n - 1, n, n!)$. □

Generalization. As we mentioned at the beginning of this section, Theorem 1 holds under more general conditions. In the general setting a table may contain "pointers" and duplicated keys. Formally, we have a universe M of m keys, a set P of p special symbols (pointers), and an array T containing q cells. Let $S \subseteq M$ be any subset of n keys. We store S in T, where each cell may contain any element in the set $S \cup P$. Each key in S may appear several times, or may not appear at all. A rule for determining the above assignment is a table structure $\mathscr{T}$. Defining search strategies $\mathscr{S}$ as before, we measure the cost $c(\mathscr{T}, \mathscr{S})$ by the number of probes needed to answer the membership query in the worst case. The complexity $f(n, m, p, q)$ is the minimum cost achievable by such a pair.

THEOREM 1'. *For any n, p, q, there exists an $N(n, p, q)$ such that $f(n, m, p, q) = \lceil lg(n + 1)\rceil$ for all $m \geq N(n, p, q)$.*

PROOF. As the proof is very similar to that of Theorem 1, we shall only sketch it. Clearly we need only prove that $f(n, m, p, q) \geq \lceil \lg(n + 1)\rceil$ for all large m.

Let $\mathscr{T}$ be any table structure. To each n-key subset S we assign a q-tuple $(i_1, i_2, \ldots, i_q)$ with $1 \leq i_l \leq n + p$, where $i_l = k$ if $T[l]$ contains the kth smallest key in S and $i_l = n + j$ if $T[l]$ contains the jth pointer. This partitions the family of all n-key subsets into $(n + p)^q$ classes. If $m \geq R(2n - 1, n, (n + p)^q)$, then by Ramsey's theorem there exists a set S_0 of $2n - 1$ keys all of whose n-key subsets are in the same class. By definition, all tables for n-key subsets $S \subseteq S_0$ contain identical pointers in each location, and hence tables are distinguished only by the keys stored in the tables. Now, in these tables the set of locations containing a given key depends only on the relative ranking of the key in the n-key subset. Therefore, from the viewpoint of search strategies, these are sorted tables (with possible missing keys). By Lemma 1, it takes $\lceil \lg(n + 1)\rceil$ probes in the worst case. As $\mathscr{T}$ is arbitrary, this proves the theorem. □

We may further allow the set S to have nonunique representations as a table (as is the case of hash tables and search trees), since this obviously will not improve the worst case cost. Thus the present model allows for the use of linked lists, search trees, all common hashing techniques, etc.

3. *When Is One Probe Sufficient?*

The numbers $N(n)$ in Theorem 1 are extremely large even for moderate n. Thus the result is not too useful in practical terms. It is of interest to understand $f(n, m)$ for smaller m. We therefore ask the following equivalent question: Given n, k, what is the maximum m such that $f(n, m) = k$? Call this number $g(n, k)$. Hence, if and only

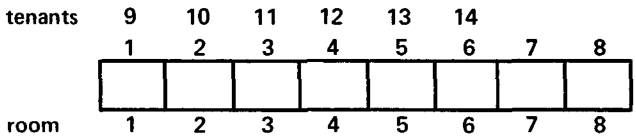

FIG 4 The association between tenants and rooms in the proof of Theorem 2

if there are more than $g(n, k)$ possible keys, then we have to use more than k probes in the worst case. The determination of $g(n, k)$ is difficult, but we can determine it in one special case.

THEOREM 2

$$g(n, 1) = \begin{cases} 3 & \textit{if} \quad n = 2, \\ 2n - 2 & \textit{if} \quad n > 2. \end{cases}$$

PROOF. We shall give a proof for the lower bound to $g(n, 1)$ by exhibiting a 1-probe table structure for the asserted number of keys. The other part of the proof, that is, that no table structure can achieve a 1-probe search for a larger key space, involves a lengthy case analysis and is left to Appendix A.

For the case $n = 2$, $m = 3$, the "cyclic" table discussed earlier has an obvious 1-probe search strategy. Now let $n > 2$ and $m = 2n - 2$; we describe a table structure allowing a 1-probe search strategy.

Consider the situation as m people sharing an apartment building with n rooms. We need a method so that no matter which n people appear at the same time, we can assign them in such a way that it is possible to determine if person j is present by looking up the occupant of one particular room (dependent on j).

We use K_j to stand for the person j ($1 \le j \le m$). Let us call K_j and K_{n+j} the *tenants* of room j, for $1 \le j \le n - 2$; K_j is the *lower tenant* and K_{n+j} the *upper tenant*. For room $n - 1$, K_{n-1} is a lower tenant, and for room n, K_n is a lower tenant. There are no upper tenants for these two special rooms. (See Figure 4.)

When a group of n people shows up, we make the assignment by the following steps.

(i) If room j ($1 \le j \le n - 2$) has only one tenant present, assign that tenant to the room.
(ii) If a room j ($1 \le j \le n - 2$) has both tenants present, let the upper tenant go to a room which has no tenants present.
(iii) Those people left unassigned are either tenants whose upper tenants are also present, or are keys K_{n-1}, K_n. We assign them so that they do not occupy the rooms of which they are tenants (e.g., a cyclic shift will do).

The last step can always be accomplished, for we can argue that if there is at least one person left in (iii), then there are at least two. Indeed, either (a) assuming neither K_{n-1} nor K_n is present, then at least two rooms j ($1 \le j \le n - 2$) have both tenants present; (b) assuming exactly one of K_{n-1}, K_n is present, then there must be another j ($1 \le j \le n - 2$) with both tenants present; or (c) both K_{n-1} and K_n are present. For example, assume in Figure 5 that the group {1, 2, 3, 6, 7, 9, 10, 12} shows up. Steps (i)–(iii) are illustrated.

To answer if K_j is in the table, we look at the room of which it is a tenant.

(a) If K_j is there, then it is in the table.
(b) If an upper tenant of some other room is there, then K_j is not in the table.
(c) If a lower tenant of some other room is there, then K_j is in the table.

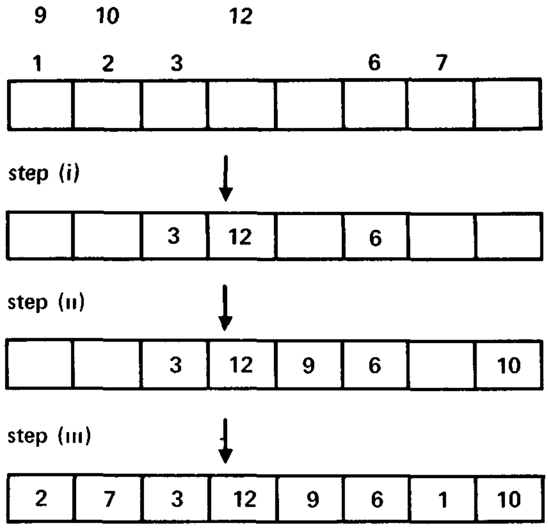

FIG. 5. An illustration of steps (i)–(iii) in the assignment.

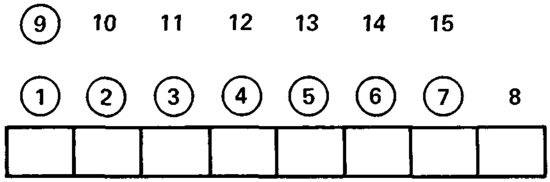

FIG. 6. Failure of the 1-probe scheme with $2n - 1$ keys.

It is straightforward to verify the correctness of the answers. This proves $g(n, 1) \geq 2n - 2$ for $n > 2$.

It remains to prove the upper bounds for $g(n, 1)$. We have shown that $g(2, 1) < 4$ in Section 2. The proof of $g(n, 1) \leq 2n - 2$ for $n \geq 3$ is left to Appendix A. □

Note that the above scheme only allows one to determine in one probe whether an element is in the table, but not the location of the element (when it is in). It would be interesting to study schemes that can determine the stored locations of the elements and to determine how large a key space can be accommodated under this stronger requirement.

Remark. It is somewhat surprising that the 1-probe schemes used in the above proof are optimal, as they look quite arbitrary. In particular, why do we need *two* special rooms $n - 1$ and n? Figure 6 shows that the scheme fails if we have only *one* special room (and $2n - 1$ keys). The arrival of keys $1, 2, \ldots, n - 1, n + 1$ will make the accommodation impossible.

4. *Searching in Two Probes*

How strong is Theorem 1'? It appears to be a robust result, considering its generality. However, the following surprising result demonstrates that it depends heavily on the fact that keys outside of the set S may *not* be present in the table.

THEOREM 3. *There exists a number $N'(n)$ such that if $m \geq N'(n)$, then by adding one extra cell in a sorted table, the search can always be accomplished in two probes. (The content in the extra cell is allowed to be any integer between* 1 *and m.)*

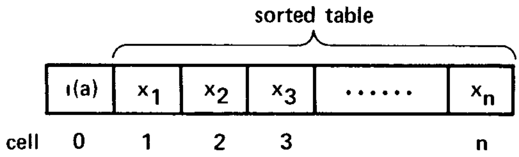

FIG 7 A 2-probe table

PROOF. We define a concept called "k-separating systems." Let $M = \{1, 2, \ldots, m\}$ and $n > 0$ an integer. An *n-separator* $F = (A_1, A_2, \ldots, A_n)$ is an ordered n-tuple of subsets $A_i \subseteq M$ which are mutually disjoint. An *n-separating system* for M is a family of n-separators such that for any n elements $x_1 < x_2 < \cdots < x_n$ of M, there exists (not necessarily unique) a member $F = (A_1, A_2, \ldots, A_n) \in \mathscr{F}$ with $x_i \in A_i$ for $i = 1, 2, \ldots, n$. Let us use $\psi(x_1, x_2, \ldots, x_n)$ to denote this F. For $y \in \bigcup_{j=1}^{n} A_j$, use $J(F, y)$ to denote the j with $y \in A_j$.

We now show how to design a 2-probe structure with the help of an n-separating system $\mathscr{F}$ for M. Let $\mathscr{F} = \{F_1, F_2, \ldots, F_l\}$. For each n-tuple $a = (x_1 < x_2 < \cdots < x_n)$ drawn from M, let $F_{i(a)} = \psi(x_1, x_2, \ldots, x_n)$. For the moment assume that $|\mathscr{F}| = l \leq m$. We organize the table as shown in Figure 7.

To test if a number $y \in M$ is in the table, one first probes at cell 0 to find $i(a)$, then makes a second probe at position $J(F_{i(a)}, y)$. The number y is in the table if and only if it is in this location.

Reason. Let $F_{i(a)} = (A_1, A_2, \ldots, A_n)$; if y is in the table, then $y \in A_j$ with $j = J(F_{i(a)}, y)$, and hence must be in the jth cell.

It remains to examine the condition that $l \leq m$. We need the following combinatorial lemma.

LEMMA 2. *There exists an n-separating system $\mathscr{F}$ for S with $|\mathscr{F}| \leq 4^{n^2} \cdot (\lg m)^{n-1}$.*

PROOF. See Appendix B. □

It follows from the lemma that if $4^{n^2} \cdot (\lg m)^{n-1} \leq m$, then the 2-probe scheme works. The condition is satisfied if $m \geq N'(n) = 2^{16n^2}$. This proves Theorem 3. □

Bob Tarjan [private communication] has improved the bound $N'(n)$ in Theorem 3 to $\exp(cn \log n)$ by a somewhat different construction.

In the proof of Theorem 3, the table structure used has a "directory" at cell 0. To retrieve a key y, one consults the directory to probe a cell which would contain j if and only if y is in the table. (Tarjan's construction also follows this pattern.) It is of interest to find tight bounds on m, n for such table structures (call them *canonical 2-probe structures*) to exist. Define a *primitive n-separating system* $\mathscr{F}$ for $M = \{1, 2, \ldots, m\}$ to be a family of n-separators such that for any n distinct elements $x_1, x_2, \ldots, x_n$ of M, there exists a member $F = (A_1, A_2, \ldots, A_n) \in \mathscr{F}$ with each A_i containing exactly one x_j. Let $b(m, n)$ be the minimum size of such a primitive n-separating system. It can be shown that $m \geq b(m, n)$ is a necessary and sufficient condition for a canonical 2-probe structure to exist. Ron Graham [private communication] has shown that asymptotically $b(m, n) \leq \sqrt{n}\, e^n \log m$ by a nonconstructive argument which implies the existence of a canonical 2-probe structure whenever $m \geq \exp(cn)$ for some constant $c > 0$.

5. *Conclusions*

We have discussed the complexity of the "membership" retrieval problem. The main conclusions are, roughly, that when the word size is large, sorted tables are optimal structures if *only* the addressing power of a random-access machine can be used, but far from optimal once arbitrary encoding of the information is allowed in the table. These results are mainly of theoretical interest, although Theorem 3 suggests that there may be fast retrieval schemes in more practical situations. The Ramsey type technique used in the proof of Theorem 1 may have wider applications. Ron Rivest [private communication] has used it to prove a conjecture concerning [13]. Below we mention some subjects for future research.

We have proved the optimality of sorted tables in a rather general framework (Theorem 1′). It would be nice if the threshold value $N(n)$ could be substantially lowered. Also, the exact determination of quantities such as $g(n, 2)$ poses challenging mathematical questions.

When arbitrary encoding was allowed, we obtained a rather curious result (Theorem 3). In either of the extreme cases $m \approx n$ and $m \geq 2^{16n^2}$, one needs at most two probes to decide if an item is in a table. In the former case the addressing power, and in the latter case the encoding power, contribute to fast retrieval. It would be interesting to study the problem for intermediate values of m. Tarjan and Yao [19] have shown that when m grows at most polynomially in n, one can retrieve in $O(1)$ probes with a $O(n)$-cell table. The question is still open for other ranges of m, say, $m = 2^{\sqrt{n}}$. Another direction of research is to study the effect of restricting the decoding procedures.

A main theme of this paper has been to discuss the membership problem in a word-length-independent framework (by letting $m \to \infty$). We list some open problems of prime importance in this framework, which are indirectly related to the membership problem.

(1) It is easy to construct similar models for more complex data manipulation problems such as executing a sequence of "INSERT," "DELETE," "MIN." We conjecture that, unlike the membership problem, nonconstant lower bounds exist even if arbitrary encoding is allowed.

(2) The Post Office Problem [4, 14]. Consider n points $v_1, v_2, \ldots, v_n$ on an $m \times m$ lattice (with $m \to \infty$). Can we encode them in cn cells so that, given any point on the lattice, we can find the nearest v_i in $O(1)$ probes? In fact, this problem is unresolved even in the one-dimensional case.

(3) Sorting Networks. In the usual Boolean networks for sorting n inputs in $\{0, 1\}$, it is known [11] that one need only use $O(n)$ gates $\wedge$, $\vee$, $\neg$. If we consider gates that are functions from $M \times M$ to M, can we build a sorting network for n inputs from M with $O(n)$ gates as $m \to \infty$? In general, the study of such networks for function computation would be interesting. See Vilfan [21] for some discussions on the formula size problems.

Appendix A. Proof of Optimality in Theorem 2

In this appendix we complete the proof of Theorem 2 by showing that $g(n, 1) \leq 2n - 2$ for $n \geq 3$. For convenience, the inductive proof is organized in the following way. We first prove that for any $n \geq 3$ and $m = 2n - 1$, a table structure allowing a 1-probe search induces a 1-probe table structure for $n' = n - 1$ and $m' = 2n' - 1$. Then we demonstrate that for $n = 3$ and $m = 2n - 1 = 5$, there cannot be any 1-probe table structure. This immediately implies $g(n, 1) < 2n - 1$ for all $n \geq 3$, completing the proof.

Suppose there is a 1-probe table structure $\mathscr{T}$ for n, $m = 2n - 1$, where $n \geq 3$. For $1 \leq j \leq 2n - 1$, let l_j be the location to examine when key j is to be retrieved. Clearly, some location will be l_j for at least two distinct j. Without loss of generality, assume that $l_1 = l_2 = 1$, that is, the content in $T(1)$ determines whether key 1 and/or key 2 are in the table. For $i = 1, 2$ let Y_i denote the set of keys j such that $T(1) = j$ implies the presence of key i in the table, and let $N_i = \{1, 2, \ldots, m\} - Y_i$. Certainly, $T(1) \in N_i$ if and only if key i is not in the table. Note that $1 \in Y_1$ and $2 \in Y_2$. We distinguish four possibilities:

Case I. $2 \in Y_1$, $1 \in Y_2$;
Case II. $2 \in N_1$, $1 \in N_2$;
Case III. $2 \in Y_1$, $1 \in N_2$;
Case IV. $2 \in N_1$, $1 \in Y_2$.

We shall show that these cases either are impossible or imply the existence of a 1-probe table structure for $n' = n - 1$ and $m' = 2n' - 1$. The following simple fact is relevant.

FACT A1. $|N_i| \geq n - 1$ *for* $i = 1, 2$.

PROOF. Otherwise, let $Y_i' \subseteq Y_i - \{i\}$ with $|Y_i'| = n$. The table T storing Y_i' will have $T[1] \in Y_i$, contradicting the absence of key i. □

LEMMA A1. *Case I is impossible.*

PROOF. By Fact A1, $|N_1| \geq n - 1$. Let $x_1, x_2, \ldots, x_{n-1} \in N_1$. Then the set $\{1, x_1, x_2, \ldots, x_{n-1}\}$ cannot be satisfactorily arranged in a table T. A key x_j in cell 1 would imply the absence of key 1, and key 1 in cell 1 would imply the presence of key 2. □

LEMMA A2. *Case II is impossible.*

PROOF. By Fact A1, $|N_1| \geq n - 1$. Let $2, x_1, x_2, \ldots, x_{n-2} \in N_1$. Then the set $\{1, 2, x_1, x_2, \ldots, x_{n-2}\}$ cannot be arranged in a table T. A key x_j or 2 in cell 1 would imply the absence of key 1, and key 1 in cell 1 would imply the absence of key 2. □

LEMMA A3. *Cases III and IV both imply the existence of a* 1-*probe table structure for* $n' = n - 1$ *and* $m' = 2n' - 1$.

PROOF. We need only prove the lemma for case III; case IV merely switches the roles of keys 1 and 2 in case III.

CLAIM A1. $|N_2| = n - 1$.

PROOF. By Fact A1, $|N_2| \geq n - 1$. Suppose $|N_2| > n - 1$; let $1, x_1, x_2, \ldots, x_{n-1} \in N_2$. Then there is no way to accommodate $\{2, x_1, x_2, \ldots, x_{n-1}\}$ in a table T. A key x_j in cell 1 would imply the absence of key 2, and key 2 in cell 1 would imply the presence of key 1. We conclude that $|N_2| = n - 1$. □

Because of Claim A1 we can write $N_2 = \{1, 3, 4, \ldots, n\}$ and $Y_2 = \{2, n + 1, n + 2, \ldots, 2n - 1\}$, renaming the keys in $\{3, 4, \ldots, 2n - 1\}$ if necessary.

CLAIM A2. $Y_1 = \{1, 2\}$.

PROOF. Otherwise, let $\{1, 2, x\} \subseteq Y_1$. If $x \in \{3, 4, \ldots, n\}$, then we cannot arrange the set $\{x, n + 1, n + 2, \ldots, 2n - 1\}$ in T, since $T[1] = x$ would imply the presence of key 1 and $T[1] = n + j$ would imply the presence of key 2. If $x \in \{n + 1, n + 2, \ldots, 2n - 1\}$, then we cannot arrange the set $\{x, 2, 3, \ldots, n\}$ in T by a similar reasoning. □

It follows from Claim A2 that $N_1 = \{3, 4, \ldots, 2n - 1\}$.

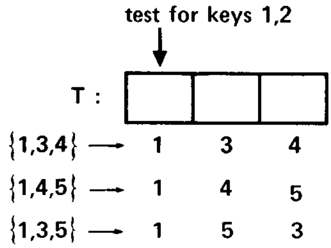

FIG 8 A partial configuration for the 1-probe table structure

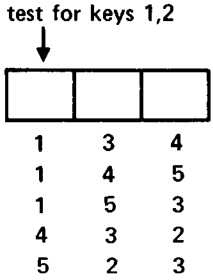

FIG 9 Our knowledge about the table structure after taking Claim A5 into consideration

CLAIM A3. *In a table T formed from an n-key subset* $\{1, x_1, x_2, \ldots, x_{n-1}\}$, *where* $x_j \neq 2$ *for all j, key* 1 *always appears in cell* 1.

PROOF. Otherwise $T[1] = x_j$ for some j, implying the absence of key 1. □

CLAIM A4. *For* $3 \leq j \leq 2n - 1$, $l_j \neq 1$.

PROOF. By Claim A3, any n-key subset S_0 with $1 \in S_0$, $2 \notin S_0$ will have key 1 in cell 1. Therefore, the key stored in $T[1]$ cannot decide if $j \in S_0$. □

Consider the set of tables for storing all the n-key subsets $\{1, x_1, x_2, \ldots, x_{n-1}\}$ with $x_j \neq 2$ for all j. Because of Claims A3 and A4, cell 1 always contains key 1, and if we eliminate cell 1 from all these tables, we are left with a 1-probe table structure for all the $(n-1)$-key subsets of $\{3, 4, \ldots, 2n-1\}$. This proves Lemma A3. □

We have completed the first part of the proof for $g(n, 1) \leq 2n - 2$, namely, the existence of a 1-probe table structure for n, $m = 2n - 1$ $(n \geq 3)$ implies the existence of such a structure for $n' = n - 1$, $m' = 2n' - 1$.

It remains to prove that no 1-probe table structure exists for $n = 3$, $m = 5$. Assume that such a structure exists; we proceed to demonstrate a contradiction. By the preceding analysis we can assume that $l_1 = l_2 = 1$, $l_3, l_4, l_5 \neq 1$, $Y_1 = \{1, 2\}$, $N_1 = \{3, 4, 5\}$, $Y_2 = \{2, 4, 5\}$, and $N_2 = \{1, 3\}$.

As the naming of keys 4 and 5 is still arbitrary, we can assume that the tables storing sets $\{1, 3, 4\}$, $\{1, 3, 5\}$, $\{1, 4, 5\}$ are as shown in Figure 8. (Note that key 1 has to be in cell 1, and the remaining have to be in a cyclic order.) Next consider how the table structure arranges $S = \{2, 3, 4\}$ and $\{2, 3, 5\}$. Keys 2 and 3 cannot be in cell 1 because $T[1] = 2$ would imply $1 \in S$ and $T[1] = 3$ would imply $2 \notin S$. Thus the arrangements can only be

$$\{2, 3, 4\} \rightarrow \begin{array}{lll} \text{either} & \text{(a)} & (4, 2, 3), \\ \text{or} & \text{(b)} & (4, 3, 2), \end{array}$$

and

$$\{2, 3, 5\} \rightarrow \begin{array}{lll} \text{either} & \text{(a)}' & (5, 2, 3), \\ \text{or} & \text{(b)}' & (5, 3, 2), \end{array}$$

where (i, j, k) means that cells 1, 2, 3 contain keys i, j, k, respectively. There are four possibilities, namely, (a) × (a)′, (a) × (b)′, (b) × (a)′, and (b) × (b)′.

CLAIM A5. *Only* (*b*) × (*a*)′ may be *possible.*

test for keys 1,2	test for key 5	test for key 4
1	3	4
1	4	5
1	5	3
4	3	2
5	2	3
3	5	4

FIG 10 More knowledge about the table structure

test for keys 1,2	test for key 5	test for key 4
1	3	4
1	4	5
1	5	3
4	3	2
5	2	3
3	5	4
2	5	1
2	1	4

FIG 11 Adding (2, 5, 1) and (2, 1, 4) to the structure

PROOF. If (a) × (a)′ or (b) × (b)′, then one cannot test in one probe whether key 4 is in the table (recall that $l_4 \neq 1$). If (a) × (b)′, again one cannot test in one probe whether key 4 is in the table — if $l_4 = 2$, then the tables (1, 3, 4) and (5, 3, 2) cannot be distinguished, and if $l_4 = 3$, then (1, 5, 3) and (4, 2, 3) cannot be distinguished. Therefore, the table structure must contain the tables shown in Figure 9. □

How is the {3, 4, 5} arranged as a table? One cannot put key 4 or 5 into cell 1 since that would imply the presence of key 2. Also, the arrangement as (3, 4, 5) would make it impossible to test for key 3 (since there is a (1, 4, 5)). Thus it has to be arranged as (3, 5, 4).

We now assert that $l_5 = 2$ and $l_4 = 3$. Testing for key 5 at cell 3 cannot distinguish (1, 3, 4) and (3, 5, 4), and testing for key 4 at cell 2 cannot distinguish (1, 5, 3) and (3, 5, 4). Our knowledge about the 1-probe table structure thus far is summarized in Figure 10.

To fill in the slots for {1, 2, 4} and {1, 2, 5}, we note that key 2 has to be put into cell 1 since both keys 1 and 2 are here. The only possibility for {1, 2, 5} is (2, 5, 1); the alternative (2, 1, 5) would jeopardize the test for key 4, since (1, 4, 5) is already there. This also means that $T[3] = 1$ implies the absence of key 4. It follows that {1, 2, 4} has to be arranged as (2, 1, 4). The known part of the table structure is shown in Figure 11.

However, there is now no way to test for key 3! If we probe at cell 2, the two tables (3, 5, 4) and (2, 5, 1) cannot be distinguished; if we probe at cell 3, the tables (1, 3, 4) and (2, 1, 4) will look the same. This contradicts the definition of a table structure allowing a 1-probe search strategy.

We have thus proved that no 1-probe table structure can exist for $n = 3$, $m = 5$. This completes the proof for $g(n, 1) < 2n - 1$ $(n \geq 3)$ and hence Theorem 2. □

Appendix B. Proof of Lemma 2

Let $m \geq k \geq 2$ and $S = \{1, 2, \ldots, m\}$. We shall construct a k-separating system $\mathscr{F}$ for S such that $|\mathscr{F}| \leq 4^{k^2}(\lg m)^{k-1}$.

We agree that the 0-*separating system* is $\varnothing$, and the 1-*separating system* for any T is $\{T\}$. The system $\mathscr{F}$ will be recursively constructed in the lexicographic order of (k, m). Divide S consecutively into k almost equal blocks $S_1, S_2, \ldots, S_k$ with $|S_i| = m_i = \lfloor (m + i - 1)/k \rfloor$. We define $\mathscr{F}$ as the union of the following families of k-

separators, to be described in a moment: $\mathscr{A}$ and $\mathscr{B}(n_1, n_2, \ldots, n_k)$, where $0 \le n_i < k$ are integers satisfying $\sum_i n_i = k$.

Let $F_i = (A_{i1}, A_{i2}, \ldots, A_{ik})$ be a k-separator for the set S_i $(1 \le i \le k)$. The *direct sum* $F_1 \oplus F_2 \oplus \cdots \oplus F_k$ is the k-separator $(A_1, A_2, \ldots, A_k)$, where $A_j = \bigcup_i A_{ij}$. Let $t > 0$ and, for each $1 \le i \le k$, $\mathscr{F}_i = \{F_{i1}, F_{i2}, \ldots, F_{it}\}$ be a family of k-separators for S_i. Define the *direct sum* $\mathscr{F}_1 \oplus \mathscr{F}_2 \oplus \cdots \oplus \mathscr{F}_k$ to be the family of k-separators for S, $\mathscr{F} = \{F_1, F_2, \ldots, F_t\}$, where $F_j = F_{1j} \oplus F_{2j} \oplus \cdots \oplus F_{kj}$ for $1 \le j \le t$. We now construct $\mathscr{A}$ as follows. Let $\mathscr{F}_i$ $(1 \le i \le k)$ be a k-separating system for S_i, constructed recursively.[2] For each j, add arbitrary k-separators into $\mathscr{F}_j$ so that the resulting family $\mathscr{F}'_j$ has $t = \max_i |\mathscr{F}_i|$ elements. We now define $\mathscr{A} = \mathscr{F}'_1 \oplus \mathscr{F}'_2 \oplus \cdots \oplus \mathscr{F}'_k$. For each $x_1 < x_2 < \cdots < x_k$ there is clearly a k-separator $F = (A_1, A_2, \ldots, A_k) \in \mathscr{A}$ that "separates" the x's (i.e., such that $x_j \in A_j$ for all j) if all x_j are in the same block S_i.

For each $(n_1, n_2, \ldots, n_k)$ that satisfies $0 \le n_i < k$ and $\sum_i n_i = k$, the family of separators $\mathscr{B}(n_1, n_2, \ldots, n_k)$ is constructed as follows. The family $\mathscr{B}(n_1, n_2, \ldots, n_k)$ is empty if there is some i such that $n_i > m_i$. Otherwise, for each $1 \le i \le k$ let $\mathscr{F}''_i$ be an n_i-separating system for S_i, recursively constructed. Denote by $\mathscr{B}(n_1, n_2, \ldots, n_k)$ the family of all k-separators of the form $F = (A_{11}, A_{12}, \ldots, A_{1n_1}, A_{21}, \ldots, A_{2n_2}, \ldots, A_{kn_k})$, where each $(A_{i1}, A_{i2}, \ldots, A_{in_i}) \in \mathscr{F}''_i$. For any $x_1 < x_2 < \cdots < x_k$ in S such that exactly n_i of the x's are in S_i for each i, there is clearly some k-separator in $\mathscr{B}(n_1, n_2, \ldots, n_k)$ that separates the x's.

Let $\mathscr{F} = \mathscr{A} \cup (\bigcup_{n_i} \mathscr{B}(n_1, n_2, \ldots, n_k))$. Then $\mathscr{F}$ is a k-separating system for S, as implied by the properties of $\mathscr{A}$ and $\mathscr{B}$ stated above. Let $f_k(n)$ denote the size of $\mathscr{F}$ constructed this way. Then by definition,

$$f_k(m) = \max\left\{ f_k\left(\left\lceil \frac{m}{k} \right\rceil\right), f_k\left(\left\lfloor \frac{m}{k} \right\rfloor\right)\right\} + \sum_{\substack{0 \le n_i < k \\ \sum n_i = k}} \prod_{i=1}^{k} f_{n_i}(m_i), \qquad \text{for} \quad m \ge k \ge 2. \tag{B1}$$

We adopted in (B1) the convention that $f_0(m_i) = 1$ and $f_{n_i}(m_i) = 0$ if $n_i > m_i$.

FACT B1. *For each $k \ge 2$, $f_k(m)$ is a nondecreasing function of m.*

PROOF. Using (B1), one can prove it by induction on (k, m), lexicographically. □

We shall now prove, by induction on k, the following formula:[3]

$$f_k(m) \le 4^{k^2} (\lg m)^{k-1} \qquad \text{for} \quad m \ge k \ge 1. \tag{B2}$$

The formula is obviously true for $k = 1$. Let $k > 1$; we shall prove (B2) assuming that it is true for all smaller values of k. First we prove the following fact.

FACT B2. *For $m = k^t$, where $t \ge 1$ is an integer, we have*

$$f_k(m) \le 4^{k^2} (\tfrac{1}{2} \lg m)^{k-1}.$$

PROOF. Using (B1), Fact B1, and the induction hypothesis, we have

$$f_k(m) \le f_k\left(\frac{m}{k}\right) + \sum_{\substack{0 \le n_i < k \text{ for all } i \\ \sum n_i = k}} 4^{\sum' n_i^2} (\lg m)^{\sum' (n_i - 1)}. \tag{B3}$$

In (B3), the summations $\sum'$ are over those i with $n_i \ne 0$. The second term in (B3) is at most

$$\binom{2k-1}{k-1} 4^{(k-1)^2+1} (\lg m)^{k-2} \le \frac{1}{2} \cdot 4^{k^2-k+2} (\lg m)^{k-2}.$$

[2] We agree that $\mathscr{F}_i = \varnothing$ if $k > |S_i|$. Also note that when $k > |S_i|$, any k-separator $(A_1, A_2, \ldots, A_k)$ for S_i must have some $A_j = \varnothing$

[3] We interpret 0^0 to be 1

Thus (B3) implies

$$\begin{aligned} f_k(m) &\leq f_k\left(\frac{m}{k}\right) + \frac{1}{2}\cdot 4^{k^2}\left(\frac{1}{4}\lg m\right)^{k-2} \\ &\leq f_k\left(\frac{m}{k^2}\right) + \frac{1}{2}\cdot 2\cdot 4^{k^2}\left(\frac{1}{4}\lg m\right)^{k-2} \\ &\ \ \vdots \\ &\leq \frac{1}{2}(\log_k m)\, 4^{k^2}\left(\frac{1}{4}\lg m\right)^{k-2} \\ &\leq 4^{k^2}\left(\frac{1}{2}\lg m\right)^{k-1}. \end{aligned}$$ □

For general m, let $k^{t-1} \leq m < k^t$ where $t \geq 2$. By Facts B1 and B2,

$$\begin{aligned} f_k(m) &\leq 4^{k^2}\left(\frac{1}{2}\lg k^t\right)^{k-1} \\ &\leq 4^{k^2}(\lg m)^{k-1}. \end{aligned}$$

This completes the inductive proof for (B2) and hence Lemma 2. □

A Bibliographic Note. The complexity of the membership problem was first raised in Minsky and Papert [10, pp. 215–221], where it was called the *exact match* problem. The model was formulated on a bitwise-access machine, with the complexity defined as the average number of bits needed to be examined for a random table. This model, especially the $n = 1$ case, was further examined by Elias and Flower [6], but the problem has not been solved completely even for this special case. Wordwise-access models were used in several recent papers. Sprugnoli's work [16] dealt with efficient hash functions and is closely related to the materials in Section 4 of the present paper. Tarjan [17] showed that tables of size $O(n)$ and retrieval time $O(\log^* n)$ can be achieved if m is at most polynomial in n; the retrieval time was improved to $O(1)$ by Tarjan and Yao [19]. Also see Bentley et. al [2], Gonnet [7], and Munro and Suwanda [12] for other recent studies on related problems.

ACKNOWLEDGMENT. I wish to thank Bob Tarjan for many helpful comments, which led me to include Theorem 1′ in the paper.

REFERENCES

1 AHO, A V., HOPCROFT, J.E , AND ULLMAN, J D *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass., 1974
2. BENTLEY, J , DETIG, D , GUIBAS, L., AND SAXE, J. An optimal data structure for minimal-storage dynamic member searching Unpublished manuscript.
3 BERGE, C *Graphs and Hypergraphs* North-Holland, Amsterdam, 1973
4 DOBKIN, D., AND LIPTON, R J Multidimensional search problems *SIAM J Comput 5* (1976), 181–186
5. ELIAS, P Efficient storage and retrieval by content and address of static files *J. ACM 21*, 2 (April 1974), 246–260
6 ELIAS, P., AND FLOWER, R A The complexity of some simple retrieval problems *J ACM 22*, 3 (July 1975), 367–379
7 GONNET, G H Average lower bounds for open-addressing hash coding. Proc Conf on Theoretical Computer Science, Waterloo, Ontario, Canada, August 1977, pp 159–162.
8 KNUTH, D E *The Art of Computer Programming, Vol 1. Fundamental Algorithms* Addison-Wesley, Reading, Mass , 1968

9 KNUTH, D E *The Art of Computer Programming, Vol 3 Sorting and Searching.* Addison-Wesley, Reading, Mass, 1973
10 MINSKY, M., AND PAPERT, S *Perceptrons.* MIT Press, Cambridge, Mass, 1969
11 MULLER, D E, AND PREPARATA, F P Bounds to complexities of networks for sorting and for switching *J ACM 22*, 2 (April 1975), 195–201
12 MUNRO, J I, AND SUWANDA, H Implicit data structures Proc 11th Ann. ACM Symp on Theory of Computing, Atlanta, Georgia, 1979, pp 108–117.
13 RIVEST, R.L Optimal arrangement of keys in a hash table *J. ACM 25*, 2 (April 1978), 200–209.
14 SHAMOS, M I. Geometric complexity Proc 7th Ann ACM Symp. on Theory of Computing, Albuquerque, N M, 1975, pp 224–233
15 SNYDER, L On uniquely representable data structures Proc 18th Ann IEEE Symp on Foundations of Computer Science, Providence, R I, 1977, pp 142–146
16 SPRUGNOLI, R Perfect hashing functions A single probe retrieving method for static sets *Commun ACM 20*, 11 (Nov 1977), 841–850
17 TARJAN, R E Storing a sparse table Stanford Computer Science Dep Rep STAN-CS-78-683, December 1978 (this is a preliminary version of [19])
18 TARJAN, R.E A class of algorithms which require nonlinear time to maintain disjoint sets *J Comput Syst Sci. 18* (1979), 110–127
19 TARJAN, R E, AND YAO, A C Storing a sparse table *Commun ACM 22*, 11 (Nov 1979), 606–611
20 VAN EMDE BOAS, P, KAAS, R, AND ZIJLSTRA, E Design and implementation of an efficient priority queue *Math Syst Theory 10* (1977), 99–127
21 VILFAN, B Lower bounds for the size of expressions for certain functions in d-ary logic *Theor Comput Sci 2* (1976), 249–269.

RECEIVED JULY 1979, REVISED JUNE 1980, ACCEPTED AUGUST 1980