SIAM J. COMPUT.
Vol. 14, No. 4, November 1985

SEARCHING SEMISORTED TABLES*

HELMUT ALT† AND KURT MEHLHORN‡

Abstract. A "semisorted table" is a one-dimensional array containing n data, which are not necessarily sorted, but can appear in p different permutations of the ascending order. We consider the problem of searching in such a table without knowing, in which one of the p permutations the data are stored (SST). It is shown that any deterministic search algorithm for SST needs at least $\sqrt[n]{p}$ comparisons in the worst case. This lower bound is generalized to average case performance even for nondeterministic algorithms. Some examples are given where the lower bound is tight.

Key words. searching, implicit data structures, comparison trees

1. Introduction. This paper deals with searching semisorted tables (SST), by which we mean the following problem:

Let U be an infinite, totally ordered universe. Let Π be a set of permutations of $\{1, \cdots, n\}$, and $p = |\Pi|$. Assume that n elements of U may be stored in an array A according to any permutation $\pi \in \Pi$ of their ascending order. How many comparisons are necessary to search for a given $x \in U$?

Note, that the restriction of our search problem to data stored in an array is not essential. In fact, even if they are stored in some arbitrary data structure, for a given problem instance the assignment of elements of the universe U to the memory locations of that structure will be unique and never will be changed. So we can give these locations a fixed numbering $1, \cdots, n$, such that the statement that data are "stored according to the permutation π of their ascending order" is well defined.

Lower and upper bounds for the degenerate cases of the problem are well known:

If $p = 1$, i.e., the data can appear in only one fixed permutation, a (modified) binary search can be done and $\lfloor \log n \rfloor + 1$ comparisons are necessary and sufficient. If $p = n!$, i.e., data can appear in any permutation, linear search has to be done with n comparisons being necessary and sufficient. Our problem is to find lower bounds for any p somewhere between 1 and $n!$.

In § 2 a comparison tree argument will show that $\Omega(\sqrt[n]{p})$ is a lower bound in the worst case. This argument and, thus, the lower bound are then extended to *nondeterministic* algorithms, and finally to the *average* complexity of SST. The final section of the paper shows that these lower bounds are tight for some, but not for all cases.

Note, that our lower bound only depends on the number p of possible arrangements of data, and does not restrict the way to arrange them. One possible restriction is, for example, to consider only data structures, where all possible arrangements of data are compatible with a given partial order. Most of the standard comparison based data structures like all versions of binary search trees, sorted arrays etc. have that property, but others, like, e.g., rotated lists [MS], do not. For data structures satisfying the partial order restriction an exact lower bound for searching has been found in a recent paper by Linial and Saks [LS]. They show that any search algorithm for a given data structure has to make at least $\log N$ comparisons, where N is the number of ideals of the underlying partial order. That method can be applied to instances of SST which satisfy a given partial order and gives better lower bounds in some of these cases (cf. second example in § 4).

* Received by the editors July 5, 1983, and in revised form May 30, 1984.

† Department of Computer Science, The Pennsylvania State University, University Park, Pennsylvania 16802.

‡ Fachbereich 10, Universität des Saarlandes, 6600 Saarbrücken, West Germany.

The SST problem by itself seems interesting to us, but there also exists one seemingly important application: Finding lower bounds for a case of partial match retrieval in implicit data structures (cf. [MS], [R]), as has been done in [AMM].

The deterministic worst case lower bound for the SST problem was investigated concurrently and independently by S. Cook ([B]).

The model of computation used here requires that comparisons are done only between the element x searched for and some array element. It is an interesting open question, if the lower bounds still hold, if comparisons between different array elements are allowed, as well.

2. A worst case lower bound for SST. Let us now define the SST problem more formally:

Let U be an infinite, totally ordered universe, let $S=\{x_i|1\leqq i\leqq n\}\subseteq U$ where $x_1<x_2<\cdots<x_n$. An array $A[1:n]$ is used to store S. If π is a permutation of $\{1,\cdots,n\}$ then we say S is stored according to π, if and only if $A[\pi(i)]=x_i$.

Let Π be a set of permutations of $\{1,\cdots,n\}$. Then SST(Π) is the following problem:

input: some $x\in U$

problem: decide, if $x\in S$ under the precondition that S is stored according to some $\pi\in\Pi$ (which one is not known though).

The following *model of computation* will be used:

Algorithms are based on comparisons of the form $x?\ A[i]$ where x is the element searched for and $i\in\{1,\cdots,n\}$. So any such algorithm can be illustrated by a comparison tree T, each node of which is labelled by some $i\in\{1,\cdots,n\}$, meaning that at this point x is compared to $A[i]$. If the outcome of that comparison is "$<$" ("$>$") the algorithm proceeds using the left (right) subtree, if it is "$=$" the algorithm halts giving a positive answer.

Certainly a computation (= path starting from the root) in T only depends on the permutation of Π in which the data are stored and the relative position of x within the data but not on the particular choice of data. (For example, a search for 3 in a table containing 4 3 5 1 2 in that order would take the same path in the comparison tree as searching for 5 in the table 7 5 9 2 3.)

Now a lower bound for SST(Π) will be shown, depending on the number p of elements of Π and on the number n of data.

THEOREM 1. *For all $n\in N$, Π any set of p permutations of $\{1,\cdots,n\}$, any algorithm solving* SST(Π) *makes at least $\sqrt[n]{p}$ comparisons in the worst case.*

Proof of Theorem 1. Call a sequence $i_1,\cdots,i_n$ $(1\leqq k\leqq n)$ *valid* if there exists a $\pi\in\Pi$ such that $\pi(j)=i_j$ $(1\leqq j\leqq k)$, i.e., if the k smallest elements may be stored in table positions $i_1,\cdots,i_k$. Denote by ε the empty sequence over $\{1,\cdots,n\}$ and let it be valid by definition. Clearly a permutation π is in Π exactly if $\pi(1),\cdots,\pi(n)$ is valid. Let T be a comparison tree solving SST(Π) and let s be its depth, i.e., $s+1$ is the maximum number of comparisons for a search.

We will show that the number of permutations for which T works correctly is at most $(s+1)^n$.

LEMMA 1. *Any valid sequence of length k $(0\leqq k<n)$ can be extended in at most $s+1$ ways to a valid sequence of length $k+1$.*

Proof. Let $i_1,\cdots,i_k$ (*the empty sequence* ε if $k=0$) be a valid sequence and $\pi\in\Pi$ some permutation which makes it valid (i.e., $\pi(j)=i_j$, $1\leqq j\leqq k$; any $\pi\in\Pi$ will do in the case $k=0$). Assume that data are stored in the table according to π, i.e., $A[\pi(i)]=x_i$. In particular the k smallest elements are stored in ascending order in $A[i_1],\cdots,A[i_k]$.

Consider the path in T which is taken by a search for some $x \in U$ with $x_k < x < x_{k+1}$, i.e., $x \notin S$, $x > A[i]$ if $i \in \{i_1, \cdots, i_k\}$, $x < A[i]$ otherwise ($x < A[i]$ for all i if $k = 0$). Such an x exists w.l.o.g. since U is infinite and the algorithm only depends on the permutation in which data are stored (i.e., we can assume that there are "gaps" between the data stored in the table.) This path now depends only on $i_1, \cdots, i_k$ (is unique if $k = 0$). In fact it can be described by the rule: start in the root, take the $>$-branch if the current node is labelled by some i_j ($1 \leqq j \leqq k$), take the $<$-branch otherwise (in the case $k = 0$ always take the $<$-branch).

Let $L = \{l_1, \cdots, l_t\}$ be the set of labels encountered on the above path which are not in $\{i_1, \cdots, i_k\}$, i.e., on the path they are the ones, whose left child is visited next (see Fig. 1).

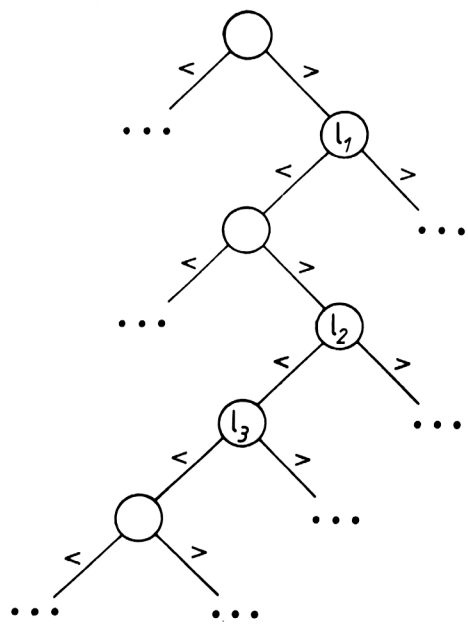

FIG. 1

Let $i_{k+1} = \pi(k+1)$, i.e., $A[i_{k+1}]$ contains x_{k+1}.

We claim: $i_{k+1} \in L$. Assume otherwise. Then x would not be compared to x_{k+1} and the outcome of all comparisons would be the same if we traverse T with x_{k+1} instead of x. So the same path would be taken and T would give the same answer for x_{k+1} as for x, namely that it is not in S, a contradiction. Now L depends on $i_1, \cdots, i_k$ only and it has $\leqq s+1$ elements, since they are some of the labels of one particular path in T. So the valid sequence $i_1, \cdots, i_k$ can be extended in at most $s+1$ ways to a valid sequence $i_1, \cdots, i_k, i_{k+1}$, which proves Lemma 1.

Now, applying Lemma 1 repeatedly, we have that there are at most $(s+1)^k$ valid sequences of length k ($1 \leqq k \leqq n$). Since all permutations in Π give different valid sequences of length n we have $P \leqq (s+1)^n$, i.e., the number of comparisons $s+1 \geqq \sqrt[n]{p}$, which proves Theorem 1.

3. Nondeterministic and average case lower bounds. In this section we want to show that the lower bound of § 2 also holds for nondeterministic algorithms and for the average search time.

A nondeterministic comparison based search, in one step of the computation compares, just like a deterministic one, the element z searched for to some table entry $A[i]$. But for any outcome "<" or ">" it may have several choices to which entry to compare x next. So the comparison tree is not necessarily binary any more. The answers the algorithm gives at the end of a computation may be "yes" ($x \in S$) "no", or "undetermined". We require the algorithm to be complete and consistent: If $x \in S$ there must not be any computation for x answering "no" and there must be at least one computation answering "yes". If $x \notin S$ there must not be any computation answering "yes" but at least one answering "no".

Of course, if $x \in S$, this can be determined by a nondeterministic algorithm in constant time: Choose nondeterministically any table position i, answer "yes" if $x = A[i]$. We show that the same lower bound as in § 2 holds in the nondeterministic case for $x \notin S$.

THEOREM 2. *For all $n \in N$, Π any set of p permutations of $\{1, \cdots, n\}$, any algorithm solving* SST (Π) *makes at least $\sqrt[n]{p}$ comparisons in the worst case, i.e., for any set $S \subset U$, $|S| = n$ there exists an $x \in U$ such that the shortest computation searching for x in S and leading to a "yes" or "no"-answer has length $\geqq \sqrt[n]{p}$.*

Proof. The proof is essentially the same as the one for Theorem 1.

Assume $0 \leqq k < n$, $i_1, \cdots, i_k$ is a valid sequence, $\pi \in \Pi$ a permutation making it valid, and data are stored according to π. Now again any search for $x \in U$ with $x_k < x < x_{k+1}$ leading to a "no" answer has to compare x with x_{k+1}. So only the positions appearing on every path in the comparison tree leading to a "no" answer for x can be used to store x_{k+1}. So if s_{k+1} is the shortest length of such a path, there are at most s_{k+1} possibilities to extend the above valid sequence to one of length $k+1$. So there are at most $s_1, s_2, \cdots, s_n$ valid sequences of length n, we need at least p to make the algorithm work for all permutations in Π. So there exists and $i \in \{1, \cdots, n\}$ with $s_i \geqq \sqrt[n]{p}$. So the shortest search for an x with $x_{i-1} < x < x_i$ makes at least $\sqrt[n]{p}$ comparisons.

Next it will be shown that the lower bound of Theorems 1 and 2 even holds for the average case of unsuccessful searches, even for nondeterministic algorithms. Any unsuccessful search can be associated with an interval $(x_i, x_{i+1}) = \{y \in U | x_i < y < x_{i+1}\}$, namely the one containing the element x searched for.

We assume that all these intervals have the same access-probability. Let Π be a set of permutations of $\{1, \cdots, n\}$, for which we consider SST (Π). To Π a tree T_Π is associated in the following way:

The nodes of T_Π except the root, are labelled with numbers of $\{1, \cdots, n\}$. There are exactly $p = |\Pi|$ leaves representing the permutations in Π. Call the leaf associated with $\pi \in \Pi$, l_π.

For any $\pi \in \Pi$ the sequence of labels on the path from the root to l_π exactly corresponds to the permutation π. (So T_Π has height n.)

As an example consider: $n = 7$, Π the set of all permutations of $\{1, \cdots, 7\}$ not displacing $1, \cdots, 4$. So $p = 3! = 6$. A suitable search strategy is to do binary search on the first 4 positions and linear search on the others. The corresponding search tree (a square denotes an unsuccessful search), is shown in Fig. 2.

The tree T_Π, in this case, is shown in Fig. 3.

Furthermore, by definition, the labelling $i_1, \cdots, i_k$ ($k \geqq 0$) of any path in T_Π starting in the root, is a valid sequence. By the proofs of Theorems 1 and 2 a valid sequence of length k ($0 \leqq k < n$) can only be extended to one of length $k+1$ by using the nodes on a certain path in the comparison tree. This path corresponds to an unsuccessful search for an element $x \in U$ with $x_k < x < x_{k+1}$. So the outdegree of the node in T_Π corresponding to the valid sequence $i_1, \cdots, i_k$ is a lower bound on the

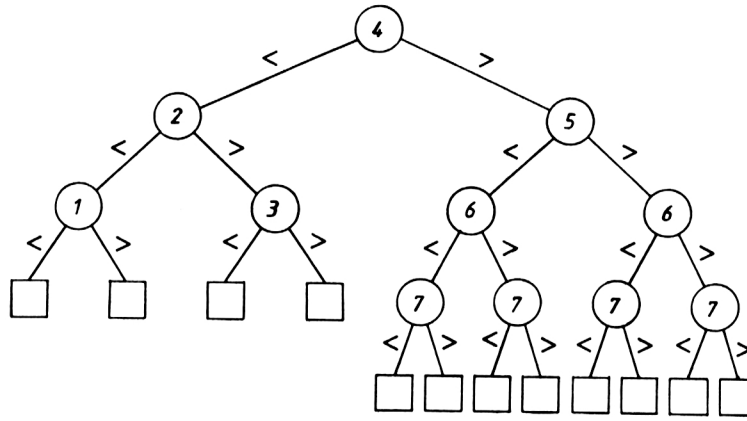

FIG. 2

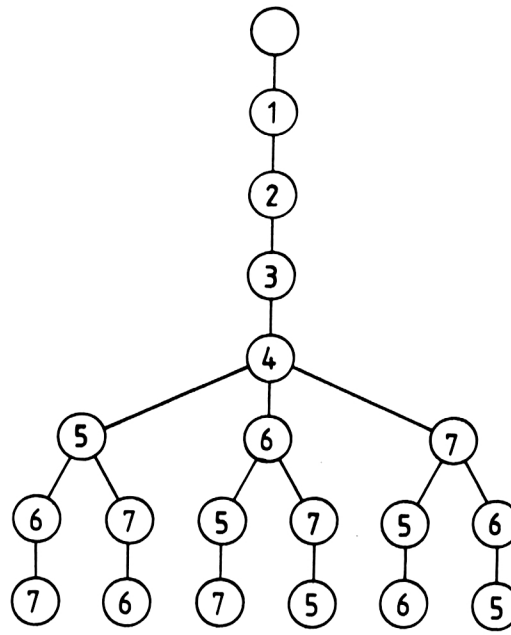

FIG. 3

number of comparisons for the above unsuccessful search, assuming that the elements $x_1, \cdots, x_k$ are stored in positions $i_1, \cdots, i_k$.

For a permutation $\pi \in \Pi$ let $c(\pi)$ be the average number of comparisons for searches for elements $x \in U$ with $x_k < x < x_{k+1}$ $(0 \leqq k < n)$ assuming that all "gaps" are equally likely.

For a tree T and any leaf l of T let $P_T(l)$ be the set of ancestors of l. Then by the considerations above

$$c(\pi) \geqq \frac{1}{n} \sum_{v \in P_{T_\Pi}(l_\pi)} \deg(v). \tag{1}$$

So, additionally assuming that all permutations are equally likely, the average number of comparisons for an unsuccessful search is

$$\frac{1}{p}\sum_{\pi\in\Pi} c(\pi).$$

This expression is because of (1) greater than or equal $s(T_\Pi)$ where for any tree T with p leaves we define

$$s(T)=\frac{1}{p}\sum_{l \text{ leaf of } T}\frac{1}{\text{depth}\,(l)}\sum_{v\in P_T(l)}\deg(v)$$

which is the average outdegree of T's nodes weighted according to their number of descendants.

CLAIM. *For any tree T with p leaves all having depth $h\geqq 1$:*

$$p\leqq s(T)^h.$$

Proof. (by induction on h). If $h=1$, the tree has the form of Fig. 4.

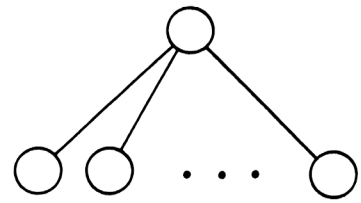

FIG. 4

Assume that the root has outdegree k. Then $p=s(T)=k$ and thus, the claim is true.

For the *inductive step* assume that we have a tree T of height $h+1$ and that its root v_0 has outdegree k. Let $T_1,\cdots,T_k$ be the subtrees of the root and $p_1,\cdots,p_k$ their numbers of leaves respectively (see Fig. 5).

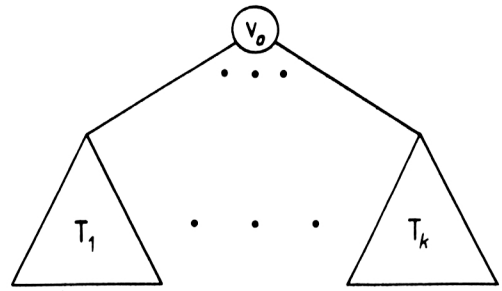

FIG. 5

Then

$$\begin{aligned} s(T)&=\frac{1}{p(h+1)}\sum_{l \text{ leaf of } T}\sum_{v\in P_T(l)}\deg(v)\\ &=\frac{1}{p(h+1)}\sum_{i=1}^{k}\sum_{l \text{ leaf of } T_i}\left[\sum_{v\in P_{T_i}(l)}\deg(v)+\deg(v_0)\right] \qquad (2)\\ &=\frac{1}{p(h+1)}\sum_{i=1}^{k}[p_ik+p_ihs(T_i)] \end{aligned}$$

since $\deg(v_0)=k$ and by definition of $s(T_i)$.

We have to show

$$p \leqq s(T)^{h+1}$$

or, because of (2)

$$p^{h+2} \leqq \left[\frac{1}{h+1}\left(kp + h \sum_{i=1}^{k} p_i s(T_i)\right)\right]^{h+1}.$$

Since by inductive hypothesis $p_i \leqq s(T_i)^h$ and hence $s(T_i) \geqq (p_i)^{1/h}$ it suffices to show

$$p^{h+2} \leqq \left[\frac{1}{h+1}\left(kp + h \sum_{i=1}^{k} p_i^{1+1/h}\right)\right]^{h+1}.$$

Now note that

$$\sum_{i=1}^{k} p_i^{1+1/h}$$

is minimal (subject to the constraint $\sum_{i=1}^{k} p_i = p$) if $p_i = p/k$ for all i.

Therefore, it suffices to show:

$$p^{h+2} \leqq \left[\frac{1}{h+1}(kp + hk(p/k)^{1+1/h})\right]^{h+1}$$

or

$$p \leqq \left[\frac{1}{h+1}(k + h(p/k)^{1/h})\right]^{h+1}. \tag{3}$$

The left-hand side of this inequation does not depend on k. Therefore, the proof is finished, if it is possible to prove inequation (3) for that k, for which the right-hand side is minimal. So, consider

$$f(k) = k + h(p/k)^{1/h}.$$

Then, the derivative

$$f'(k) = 1 - p^{1/h} \cdot k^{-1/(h+1)}$$

equals 0, if and only if

$$k^{h+1} = p,$$

i.e.,

$$k = p^{1/(h+1)}.$$

So, by (3), it suffices to show

$$p \leqq \left[\frac{1}{h+1}(p^{1/(h+1)} + h(p^{1-1/(h+1)})^{1/h})\right]^{h+1}.$$

The right-hand side equals

$$\left[\frac{1}{h+1}(p^{1/(h+1)} + hp^{1/(h+1)})\right]^{h+1} = [p^{1/(h+1)}]^{h+1} = p,$$

which finishes the proof of the claim.

Applying the claim to the tree T_Π gives

$$p^{1/n} \leqq s(T_\Pi).$$

So, since $s(T_\Pi)$ is a lower bound on the average number of comparisons for unsuccessful searches in SST (Π), we have the following result:

THEOREM 3. *For all $n \in N$, Π any set of p permutations of $\{1, \cdots, n\}$, any nondeterministic algorithm solving* SST (Π) *makes at least $\sqrt[n]{p}$ comparisons on the average.*

4. Tightness of the lower bounds. Theorems 1 through 3, of course, only give nontrivial lower bounds, if the number p of possible permutations is "sufficiently high". In fact, p needs to grow more than exponentially in n, in order to have $\sqrt[n]{p}$ not bounded above by a constant.

One case in which the lower bounds are tight, is $p = n!$. So every possible permutation is allowed and by linear search we have $O(n)$ worst case and average algorithms. On the other hand Theorems 1 and 3 give an $\Omega(\sqrt[n]{n!}) = \Omega(n)$ lower bound, showing that linear search is asymptotically optimal. (Of course, this lower bound can be shown by a much easier argument.) Linear search can be generalized in the following way:

Let $s = \Omega(\log n)$ and assume w.l.o.g. that s divides n. Let Π be the set of all permutations of $\{1, \cdots, n\}$ obtained in the following way:

$\{1, \cdots, n\}$ is broken up into n/s blocks of size s. The first block contains $\{1, \cdots, s\}$, the second one contains $\{s+1, \cdots, 2s\}$ etc. Within a block the order of elements is arbitrary.

So $p = (s!)^{n/s}$ and the lower bounds from Theorems 1 and 3 are

$$\sqrt[n]{p} = (s!)^{1/s} = \Theta(s).$$

On the other hand, we have the following algorithm to solve SST (Π):

Do a binary search on positions $1, s+1, 2s+1$, etc., i.e., the first positions of the blocks in order to find the only two blocks which possibly may contain the element searched for. Then do a linear search in these blocks.

This algorithm performs in the worst case and on the average

$$O(\log(n/s) + s) = O(\log n - \log s + s)$$
$$= O(s)$$

comparisons since $s = \Omega(\log n)$. So the lower bound given by Theorems 1 through 3 are tight in all these cases. Note, that while s ranges from $\log n$ to n, p ranges from $\Theta(2^{n \log \log n})$ to $\Theta(n!)$.

But not in every case are the lower bounds of this paper tight. As a counter example let Π be the set of all permutations, where the elements $\{1, \cdots, n/2\}$ are in their original positions, the other ones are permuted arbitrarily. Clearly

$$p = (n/2)!$$

and

$$\sqrt[n]{p} = \sqrt[n]{(n/2)!} = \Theta[(n/2)^{1/2}] = \Theta(\sqrt{n}).$$

On the other hand it is clear, that no algorithm can do better than $\Theta(n)$ even on the average, because on the second half of the data a linear search has to be performed.

REFERENCES

[AMM] H. ALT, K. MEHLHORN AND J. I. MUNRO, *Partial match retrieval in implicit data structures*, Inform. Proc. Lett., 19 (1984), pp. 61-65.

[B] A. BORODIN, Oral communication.

[LS] N. LINIAL AND M. E. SAKS, *Information bounds are good for search problems on ordered data structures*, Proc. 24th IEEE Symposium on Foundations of Computer Science, Tucson, AZ, 1983, pp. 473-475.

[M] J. I. MUNRO, *A multikey search problem*, Proc. 17th Annual Allerton Conference, October 1979.

[MS] J. I. MUNRO AND H. SUWANDA, *Implicit data structures*, Proc. 11th Annual ACM Symposium on Theory of Computing, May 1979, pp. 108-117.

[R] R. L. RIVEST, *Partial match retrieval algorithms*, this Journal, 5 (1976), pp. 19-50.

[Y] A. C. YAO, *Should tables be sorted?*, Proc. IEEE Symposium on Foundations of Computer Science, 1978, pp. 22-27.