On Finding the Maxima of a Set of Vectors

H. T. KUNG

Carnegie-Mellon University, Pittsburgh, Pennsylvania

F. LUCCIO

Università di Pisa, Pisa, Italy

F. P. PREPARATA

University of Illinois, Urbana, Illinois

ABSTRACT. Let $U_1, U_2, \ldots, U_d$ be totally ordered sets and let V be a set of n d-dimensional vectors in $U_1 \times U_2 \ldots \times U_d$. A partial ordering is defined on V in a natural way The problem of finding all maximal elements of V with respect to the partial ordering is considered The computational complexity of the problem is defined to be the number of required comparisons of two components and is denoted by $C_d(n)$. It is trivial that $C_1(n) = n - 1$ and $C_d(n) \leq O(n^2)$ for $d \geq 2$ In this paper we show: (1) $C_2(n) = O(n \log_2 n)$ for $d = 2, 3$ and $C_d(n) \leq O(n(\log_2 n)^{d-2})$ for $d \geq 4$, (2) $C_d(n) \geq \lceil \log_2 n! \rceil$ for $d \geq 2$

KEY WORDS AND PHRASES: maxima of a set of vectors, computational complexity, number of comparisons, algorithm, recurrence

CR CATEGORIES. 5.25, 5.31, 5.39

1. *Introduction*

Let $U_1, U_2, \cdots, U_d$ be totally ordered sets and let V be a set of n d-dimensional vectors in the Cartesian product $U_1 \times U_2 \times \cdots \times U_d$. For any vector v in V, let $x_i(v)$ denote the ith component of v. A partial ordering $\leq$ is defined on V in a natural way, that is, for v, $u \in V$, $v \leq u$ if and only if $x_i(v) \leq_i x_i(u)$ for all $i = 1, \cdots, d$, where $\leq_i$ is the total ordering on U_i. (We shall often write $\leq$ for $\leq_i$. The context should make clear the meaning of $\leq$.) For $v \in V$, v is defined to be a *maximal element* (or, briefly, a *maximum*) of V if there does not exist $u \in V$ such that $u \geq v$ and $u \neq v$. We consider the problem of finding all maximal elements of V. The computational complexity of the problem is defined to be

$$C_d(n) = \min_A \max_V c_d(A, V),$$

This research was supported in part by the National Science Foundation under Grant GJ32111, the Office of Naval Research under Contract N0014-67-A-0314-0010, NR044-422, and the National Research Council of Italy (C.N.R), through the Istituto di Elaborazione dell'Informazione, Pisa, Italy.

Authors' addresses: H. T. Kung, Department of Computer Science, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA 15213; F. Luccio, Istituto di Scienze dell'Informazione, Università di Pisa, 56100 Pisa, Italy; F. P. Preparata, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801

where $c_d(A, V)$ is the number of comparisons used by any algorithm A on any such set V. In other words, $C_d(n)$ is the maximum number of comparisons used by the algorithm that solves the problem the fastest in the worst case. We are interested in obtaining the upper and lower bounds on $C_d(n)$ for all d. We assume that n, the number of vectors in V, is much greater than d, the dimension of V.

This problem arises in a number of applications, typically in pattern classification and in operations research, and is of interest only when $d > 1$, that is, when V is a *partially* ordered set. In fact when $d = 1$, i.e. V is a totally ordered set, we trivially have $C_1(n) = n - 1$. It is not difficult to realize that any algorithm designed to find the maxima of a *general* partially ordered set requires $O(n^2)$ comparisons in the worst case. A natural question is whether the particular structure of the partial ordering on V can be exploited to obtain a faster algorithm. This question is answered affirmatively in this paper, where we show[1]

$$C_d(n) \leq O(n \log n) \quad \text{for} \quad d = 2, 3, \tag{1.1}$$

$$C_d(n) \leq O(n (\log n)^{d-2}) \quad \text{for} \quad d \geq 4, \quad \text{and} \tag{1.2}$$

$$C_d(n) \geq \lceil \log n! \rceil \quad \text{for} \quad d \geq 2.^2 \tag{1.3}$$

Since $\log n!$ is approximately $n \log n$, the bounds in (1.1) and (1.3) are sharp for $d = 2$ and 3, with respect to the order of magnitude. It remains an open problem to show whether the bounds in (1.2) are sharp for $d \geq 4$.

The results (1.1) for $d = 2$ and $d = 3$ were originally obtained by Luccio and Preparata [3]. Their technique, however, did not generalize to a larger number of dimensions. The general results (1.2) were later obtained by Kung [2] with a different technique, where their algorithm for $d = 3$ is used as one of the important components. The lower bound results (1.3) were also originally given in [2]. Hence the present paper is a combination of papers [2] and [3].

The paper is organized as follows. In Section 2 we prove (1.3). In Section 3 we describe a technique which achieves (1.1). In Section 4 we describe the basic recursive procedure for obtaining (1.2), which is based on a merge-like algorithm described in Section 5. Upper bounds on the number of comparisons for solving this problem are established by another recursive procedure, in Section 5.

2. *Lower Bound*

LEMMA 2.1. $C_{d-1}(n) \leq C_d(n)$ *for* $d \geq 2$.

PROOF. Consider a set V_{d-1} of n $(d - 1)$-dimensional vectors in $U_1 \times \cdots \times U_{d-1}$ Let each vector in V_{d-1} be extended by the same element of U_d and let V_d be the set of these d-dimensional vectors. Then it is clear that v is a maximum of V_{d-1} if and only if the vector extended from v is a maximum of V_d. Hence for finding the maxima of V_{d-1} it suffices to find the maxima of V_d Therefore, $C_{d-1}(n) \leq C_d(n)$. $\square$

LEMMA 2.2. $C_2(n) \geq \lceil \log n! \rceil$.

PROOF. Let $(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)$ be n 2-dimensional vectors, where $a_1, a_2, \cdots, a_n$ are n distinct elements from a totally ordered set. Define an ordering on $b_1, b_2, \cdots, b_n$ by the following rule: for any i, j,

$$b_i < b_j \quad \text{if} \quad a_j < a_i . \tag{2.1}$$

Hence the ordering for b_i, b_j is detected by comparing a_i to a_j.

[1] In this paper, all logarithms are to base 2 and all comparisons are between components of the vectors in V

[2] Yao [4] has shown that $C_d(n) \geq S(n) + n - 1$ where $S(n)$ is the minimal number of comparisons to sort n keys. Since $S(n)$ is about $n \log n$, she has slightly improved the lower bound in (1 3). Note that for $d = 2$, her lower bound is sharp and is achieved by our algorithm in Section 3.

Consider any algorithm for finding the maxima of n 2-dimensional vectors. Apply this algorithm to the vectors $(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)$. We shall use the well-known information-theoretic argument (Knuth [1]) to show the algorithm requires at least $\lceil \log n! \rceil$ comparisons in the worst case. It suffices to show that the (binary) comparison tree associated with the algorithm has at least $n!$ leaves, i.e. one for each ordering of $a_1, a_2, \cdots, a_n$.

Each leaf can be associated with a directed graph (Hasse diagram), in which there exists an arc from node a_i to node a_j if and only if $a_i < a_j$ was the result of a comparison on the path from the root to the leaf. For each (a_i, b_i) the algorithm must determine whether (a_i, b_i) is a maximal element or not. To determine that (a_i, b_i) is maximal, the results of all comparisons in the algorithm must be sufficient to decide that for any other vector (a_j, b_j), either $a_j < a_i$ or $b_j < b_i$, i.e. $a_i < a_j$, holds. Since by (2.1) all (a_i, b_i) are maximal, the transitive closure of the directed graph at a leaf then must have an arc between *every* pair of nodes. Hence this transitive closure determines the ordering of $a_1, a_2, \cdots, a_n$. Therefore, each leaf is associated with a unique ordering of $a_1, a_2, \cdots, a_n$. This implies that there are at least $n!$ leaves. $\square$

Therefore, by Lemmas 2.1 and 2.2, we have shown the following:

THEOREM 2.1. *For any $d \geq 2$, $C_d(n) \geq C_{d-1}(n) \geq \cdots \geq C_2(n) \geq \lceil \log n! \rceil$, so that $O(n \log n)$ comparisons are needed for finding the maxima of n d-dimensional vectors in the worst case.*

3. *Algorithms for $d = 2, 3$*

In this section we shall present algorithms which achieve $C_2(n) = O(n \log n)$ and $C_3(n) = O(n \log n)$. In the subsequent sections we shall use a modification of the algorithm for $d = 3$ to achieve the general upper bounds asserted in (1.2). Here and hereafter, we assume that for any two vectors, u, v in the sets V, R, or S defined below, $x_i(u) \neq x_i(v)$ for all i. This simplifying assumption helps bring out the central ideas of the algorithms, while the modifications required by the unrestricted case are straightforward.

ALGORITHM 3.1

This algorithm finds the maxima of a set of d-dimensional vectors $V = \{v_1, \cdots, v_n\}$. Given a d-dimensional vector u, by u^* we denote its projection on the coordinates $x_2, \cdots, x_d$. We assume that a test for the conditions "$u < T$" is available, where u is a $(d-1)$-dimensional vector, T is a set of $(d-1)$-dimensional vectors, and "$u < T$" means that there is a $w \in T$ such that $u \leq w$

1 Arrange the elements of V as a sequence $v_1, \cdots, v_n$ such that

$$x_1(v_1) > x_1(v_2) > \cdots > x_1(v_n).$$

2. Set $i \leftarrow 1$ and $T_0 \leftarrow \varnothing$. ($T_0, T_1, \cdots$, are sets of $(d-1)$-dimensional vectors.)
3 If $v_i^* < T_{i-1}$, set $T_i \leftarrow T_{i-1}$, else set $T_i \leftarrow$ maxima $(T_{i-1} \cup v_i^*)$.
4 If $i = n$, halt; else set $i \leftarrow i + 1$ and return to step 3

THEOREM 3.1. *The vector v_i is a maximal element of V if and only if $v_i^* \in T_n$.*

PROOF. Assume inductively that $T_{i-1} = \{w_1, \cdots, w_r\}$ is the set of the maxima of $\{v_k^* \mid k = 1, \cdots, i-1\}$. From step 1 in Algorithm 3.1 we know that $x_1(v_i) < x_1(v_j)$ for $j = 1, 2, \cdots, i-1$. But for any v_j $(j = 1, \cdots, i-1)$ there is some $w \in T_{i-1}$ such that $v_j^* \leq w$. The condition $v_i^* \nless T_{i-1}$ means that for any $w \in T_{i-1}$ there is at least one coordinate $x_k (k = 2, \cdots, d)$ such that $x_k(w) < x_k(v_i)$; thus we have $x_k(v_j) \leq x_k(w) < x_k(v_i)$. The two conditions $x_1(v_i) < x_1(v_j)$ and $x_k(v_i) > x_k(v_j)$ for $j = 1, 2, \cdots, i-1$ show that v_i is a maximal element in T_i. Conversely, the condition $v_i^* < T_{i-1}$ means that there is at least one $w \in T_{i-1}$ such that $x_k(v_i) \leq x_k(w)$ for $k = 2, \cdots, d$. We also know that w coincides with v_h^*, for some h in the range $[1, i-1]$; recalling that $x_1(v_i) < x_1(v_h)$, we conclude that $v_i \leq v_h$, i.e. v_i is not a maximum. $\square$

Next we estimate the running time of Algorithm 3.1. In addition to step 1, which requires $O(n \log n)$ comparisons, the work is essentially due to step 3 (implementation of the test $v_i^* < T_{i-1}$ and, when required, the construction of T_i). We analyze these two operations for $d = 2$ and $d = 3$, separately.

$d = 2$. In this case $v_j^* = x_2(v_j)$, whence T_{i-1} consists of only one element $\max_{j=1}^{i-1} x_2(v_j)$ which is denoted by w_{i-1}. Therefore the test $v_i^* < T_{i-1}$ reduces to the single comparison between $x_2(v_i)$ and w_{i-1}; moreover when $x_2(v_i) > w_{i-1}$, the construction of T_i is trivial, since $w_i = x_2(v_i)$. In summary the number of required comparisons is $O(n \log n) + O(n)$, i.e., $O(n \log n)$.

$d = 3$. In this case we assume inductively that the two-dimensional elements of T_{i-1} are arranged as a sequence $w_1, \cdots, w_\nu$ such that $x_2(w_1) > x_2(w_2) > \cdots > x_2(w_\nu)$. We carry out the test $v_i^* < T_{i-1}$ as follows: "Determine the largest value j^* of the index j such that $x_2(w_j) \geq x_2(v_i)$ for $w_j \in T_{i-1}$; $v_i^* < T_{i-1}$ if and only if $x_3(v_i) \leq x_3(w_{j^*})$." The critical operation is the determination of j^*. This is most readily done by adopting an AVL tree [1, Sec. 6.2.3] as the information structure which stores the elements of T_{i-1}. Since the length of the longest path in an AVL tree with ν vertices is upper bounded by $1.44 \log (\nu + 1)$, the determination of j^* requires at most $O(\log \nu) \leq O(\log n)$ comparisons, and $O(n \log n)$ comparisons are needed by the tests $v_i^* < T_{i-1}$. When $v_i^* \nless T_{i-1}$, the construction of T_i can be carried out as follows: "Compare $x_3(v_i)$ with $x_3(w_h)$ for $h = j^* + 1, \cdots, q$, where q is the smallest value of the index k such that $x_3(v_i) < x_3(w_k)$. To obtain T_i, remove $w_{j^*+1}, \cdots, w_{q-1}$ from T_{i-1} and insert v_i^*, i.e. set $T_i = T_{i-1} - \{w_h \mid j^* < h < q\} + v_i^*$." Each insertion into or deletion from the AVL tree requires at most $O(\log n)$ comparisons. Since in the worst case at most n vectors are to be inserted and at most $(n - 1)$ vectors are to be deleted in the entire execution of the algorithm, the number of comparisons required by the algorithm for $d = 3$ is $O(n \log n)$. Thus, we have proved that $O(n \log n)$ is an upper bound to $C_d(n)$ for $d = 2$ and 3. Together with the lower bound proved in Section 2, we have:

THEOREM 3.2. *$C_2(n) = O(n \log n)$ and $C_3(n) = O(n \log n)$.*

Theorem 3.2 establishes (1.1). It is easily realized that Algorithm 3.1 fails to achieve (1.2) for $d > 3$. Another technique incorporating a modification of Algorithm 3.1 will be developed in Section 5 to achieve the general bound (1.2).

4. *A General Algorithm for $d > 3$*

Without loss of generality, we assume that $n = 2^r$ for some positive integer r, and that the elements of V have been arranged as a sequence $v_1, \cdots, v_n$ so that

$$x_1(v_1) > x_1(v_2) > \cdots > x_1(v_n). \tag{4.1}$$

(Note that this sorting operation takes $O(n \log n)$ comparisons.)

Like many other "fast" algorithms (e.g. FFT), our algorithms will first solve two subproblems and then combine the results of the subproblems. We shall first find $\bar{R}$, the set of the maxima of $\{v_1, \cdots, v_{n/2}\}$ and $\bar{S}$, the set of the maxima of $\{v_{n/2+1}, \cdots, v_n\}$. Observe that by (4.1) the elements of $\bar{R}$ are also maximal elements of V, but the elements in $\bar{S}$ are not necessarily maximal elements of V. In fact, an element in $\bar{S}$ is a maximal element of V if and only if it is not less than or equal to any element in $\bar{R}$. Therefore, we have the following algorithm:

ALGORITHM 4.1

We define a recursive procedure for finding the set V_M of the maxima of $V = \{v_1, \cdots, v_n\}$ To find V_M, we find $\bar{R}$, the set of the maxima of $\{v_1, \cdots, v_{n/2}\}$, find $\bar{S}$, the set of the maxima of $\{v_{n/2+1}, \cdots, v_n\}$, and then find $\bar{T}$, the set of elements in $\bar{S}$ which are not less than or equal to any element in $\bar{R}$. Then set $V_M \leftarrow \bar{R} \cup \bar{T}$.

The number of comparisons required by Algorithm 4.1 depends on the number of comparisons required to find $\bar{T}$. Define

$$F_d(r, s) = \min_{A} \max_{\substack{|R| = r \\ |S| = s}} f_d(A, R, S),$$

where R and S are any sets consisting of r and s, respectively, d-dimensional vectors, and $f_d(A, R, S)$ is the number of comparisons used by any algorithm A for finding the elements in S which are not less than or equal to any element in R. Hence $\bar{T}$ can be found in $F_d(n/2, n/2)$ comparisons, since $|\bar{R}|, |\bar{S}| \leq n/2$. Observe, however, that because of the relation (4.1), for $u \in \bar{R}$, $v \in \bar{S}$, $u \geq v$ if and only if $x_i(u) \geq x_i(v)$ for $i = 2, \cdots, d$. To find $\bar{T}$, the first components of the vectors do not have to be considered. We end up with considering $(d-1)$-dimensional vectors. Hence $\bar{T}$ can be found in $F_{d-1}(n/2, n/2)$ instead of $F_d(n/2, n/2)$ comparisons. Therefore, by Algorithm 4.1, we obtain the following recurrence relation on $C_d(n)$:

$$C_d(n) \leq 2C_d(n/2) + F_{d-1}(n/2, n/2). \tag{4.2}$$

In Section 5, we shall show (Theorem 5.2) that

$$F_d(r, s) \leq (\alpha_d r + \beta_d s)(\log r)(\log s)^{d-3} + dr \tag{4.3}$$

for $d \geq 3$, where α_d and β_d are constants. By (4.3), we have

$$F_{d-1}(n/2, n/2) \leq O(n(\log n)^{d-3}) \quad \text{for} \quad d \geq 4. \tag{4.4}$$

Therefore, from (4.2) and (4.4) we obtain $C_d(n) \leq O(n(\log n)^{d-2})$, which yields the central result:

THEOREM 4.1. *$C_d(n) \leq O(n(\log n)^{d-2})$ for $d \geq 4$.*

5. *Upper Bounds on $F_d(r, s)$*

This section deals with the proof of the following result.

$$F_d(r, s) \leq (\alpha_d r + \beta_d s)(\log r)(\log s)^{d-3} + dr, \quad \text{for} \quad d \geq 3. \tag{5.1}$$

Let R and S be two sets consisting of r and s, respectively, d-dimensional vectors. Assume $d \geq 3$. Without loss of generality we assume that the elements of R have been arranged as $u_1, \cdots, u_r$ and the elements of S as $v_1, \cdots, v_s$ so that

$$x_1(u_1) > x_1(u_2) > \cdots > x_1(u_r), \qquad x_1(v_1) > x_1(v_2) > \cdots > x_1(v_s). \tag{5.2}$$

Also, we assume that $s = 2^m$ for some positive integer m. Define $x_1(u_0) = \infty$ and $x_1(u_{r+1}) = -\infty$. Using binary search we find k, $0 \leq k \leq r$, such that

$$x_1(u_k) \geq x_1(v_{s/2}) > x_1(u_{k+1}). \tag{5.3}$$

We now divide R into two subsets R_1 and R_2 such that $R_1 = \{u_i \mid 1 \leq i \leq k\}$ and $R_2 = \{u_i \mid k < i \leq r\}$. Also divide S into two subsets S_1 and S_2 such that $S_1 = \{v_i \mid 1 \leq i \leq s/2\}$ and $S_2 = \{v_i \mid s/2 < i \leq s\}$.

$$\begin{array}{l}
R_1\begin{cases} u_1 = (x_1(u_1), x_2(u_1), \cdots, x_d(u_1)), \\ \quad\vdots \\ u_k = (x_1(u_k), x_2(u_k), \cdots, x_d(u_k)). \end{cases} \\
\hdashline
R_2\begin{cases} u_{k+1} = (x_1(u_{k+1}), x_2(u_{k+1}), \cdots, x_d(u_{k+1})), \\ \quad\vdots \\ u_r = (x_1(u_r), x_2(u_r), \cdots, x_d(u_r)). \end{cases} \\
\hline
S_1\begin{cases} v_1 = (x_1(v_1), x_2(v_1), \cdots, x_d(v_1)), \\ \quad\vdots \\ v_{s/2} = (x_1(v_{s/2}), x_2(v_{s/2}), \cdots, x_d(v_{s/2})). \end{cases} \\
\hdashline
S_2\begin{cases} v_{s/2+1} = (x_1(v_{s/2+1}), x_2(v_{s/2+1}), \cdots, x_d(v_{s/2+1})), \\ \quad\vdots \\ v_s = (x_1(v_s), x_2(v_s), \cdots, x_d(v_s)). \end{cases}
\end{array}$$

Recall that our problem is to find all elements in S which are not less than any element in R. We let $\left[{R \atop S}\right]$ denote this problem. It is trivial to see that the problem $\left[{R \atop S}\right]$ can be solved by solving four subproblems, $\left[{R_1 \atop S_1}\right]$, $\left[{R_2 \atop S_1}\right]$, $\left[{R_1 \atop S_2}\right]$, and $\left[{R_2 \atop S_2}\right]$. Observe that the problem $\left[{R_2 \atop S_1}\right]$ is trivial, since by (5.2) and (5.3) we know there is no element in R_2 which is greater than any element in S_1. Thus, we do not have to worry about the problem $\left[{R_2 \atop S_1}\right]$. Furthermore, observe that by (5.2) and (5.3), the first component of any element in R_1 is greater than that of any element in S_2. Hence by the same reason as we used in Section 4, to do the problem $\left[{R_1 \atop S_2}\right]$ we only have to consider $(d-1)$-dimensional vectors rather than d-dimensional vectors. Thus, to solve the problem $\left[{R \atop S}\right]$ for d-dimensional vectors, we can instead solve the three subproblems:

(1) The problem $\left[{R_1 \atop S_1}\right]$ for d-dimensional vectors.

(2) The problem $\left[{R_2 \atop S_2}\right]$ for d-dimensional vectors.

(3) The problem $\left[{R_1 \atop S_2}\right]$ for $(d-1)$-dimensional vectors.

Therefore, we have shown

$$F_d(r, s) \leq F_d(k, s/2) + F_d(r-k, s/2) + F_{d-1}(k, s/2). \tag{5.4}$$

In the remainder of the section we shall first prove (5.1) for $d = 3$, and then use (5.4) to prove (5.1) for general d by induction.

THEOREM 5.1. *$F_3(r, s) \leq (\alpha_3 r + \beta_3 s)(\log r)$ for constants α_3 and β_3.*

PROOF. We establish the theorem by exhibiting an algorithm and evaluating its running time.

ALGORITHM 5 1

This algorithm accepts two sets R and S of 3-dimensional vectors with r and s elements, respectively, and finds all the elements of S which are not less than any element of R. This algorithm, which is very closely reminiscent of list-merge, is an adaptation of Algorithm 3.1 and adopts its notational conventions.

1 Arrange elements of R as a sequence $u_1, \cdots, u_r$ such that $x_1(u_1) > x_1(u_2) > \cdots > x_1(u_r)$. Define $x_1(u_{r+1}) = -\infty$.
2 Arrange the element of S as a sequence $v_1, \cdots, v_s$ with the property that $x_1(v_j) < x_1(u_i) \leq x_1(v_h) \Rightarrow j < h$. (*Comment:* The sequence $v_1, \cdots, v_s$ is formed by binary insertion of $x_1(v_j)$ in the sequence $x_1(u_1), \cdots, x_1(u_r)$)
3 Set $i \leftarrow 1$, $j \leftarrow 1$, and $T_0 \leftarrow \varnothing$ ($T_0, T_1, \cdots$ are sets of 2-dimensional vectors)
4 If $x_1(u_i) < x_1(v_j)$ go to step 7.
5 If $u_i^* < T_{i-1}$, set $T_i \leftarrow T_{i-1}$, else set $T_i \leftarrow$ maxima $(T_{i-1} \cup u_i^*)$
6 Set $i \leftarrow i + 1$ and go to step 4.
7. If $v_j^* < T_{i-1}$, discard v_j, else v_j is not less than any element in R
8. If $j = s$, halt, else set $j \leftarrow j + 1$ and go to step 4

The proof of the validity of Algorithm 5.1 closely parallels the one we presented for Algorithm 3.1, and will therefore be omitted. We now estimate the number of comparisons performed by the algorithm. Step 1 requires $O(r \log r)$ comparisons, and step 2 requires $O(s \log r)$ comparisons (s binary insertions into a set of cardinality r). We have shown in Section 3 that a test of the type "$w < T$" requires $O(\log \nu)$ comparisons if $|T| = \nu$; since step 7 is executed s times, step 5 is executed at most r times, and $\log \nu \leq \log r$, these tests require at most $O((r + s) \log r)$ comparisons. We have also shown in Section 3 that the total number of comparisons required to construct the sequence T_0, $T_1, \cdots$ is at most $O(r \log r)$. This shows that $F_3(r, s)$ is $O(r \log r) + O(s \log r)$. □

THEOREM 5.2. *For $d \geq 3$,*

$$F_d(r, s) \leq (\alpha_d r + \beta_d s)(\log r)(\log s)^{d-3} + dr, \tag{5.5}$$

where $\alpha_d = \alpha_3 + 3 + 4 + \cdots + (d-1)$ and $\beta_d = 2^{-(d-3)}\beta_3$.

(α_3, β_3 are given by Theorem 5.1.)

PROOF. We shall prove the theorem by induction on d. By Theorem 5.1, (5.5) holds for $d = 3$. Assume that (5.5) holds for $d = l - 1$. Without loss of generality, we assume that $s = 2^m$ for some positive integer m. Then we have

$$F_{l-1}(r, 2^m) \le (\alpha_{l-1} r + \beta_{l-1} 2^m)(\log r) m^{l-4} + (l-1)r. \tag{5.6}$$

By (5.4) we know that there exist $p_1(= k/r)$and q_1 $(= (r-k)/r)$ such that

$$F_l(r, 2^m) \le F_l(p_1 r, 2^{m-1}) + F_l(q_1 r, 2^{m-1}) + F_{l-1}(p_1 r, 2^{m-1}). \tag{5.7}$$

Note that

$$0 \le p_1,\ q_1 \le 1 \quad \text{and} \quad p_1 + q_1 = 1. \tag{5.8}$$

We shall use (5.6) and (5.7) to prove that

$$F_l(r, 2^m) \le (\alpha_l r + \beta_l 2^m)(\log r)\, m^{l-3} + lr,$$

that is, (5.5) for $d = l$. The proof below is elementary but tedious. The essential idea is to apply (5.7) recursively. It is not difficult to see from (5.7) that we can prove that

$$F_l(r, 2^m) \le \sum_{\substack{i_1=1\\ i_k=1,2}} [F_l(A_{i_1,\ldots,i_m} r, 1) + F_l(B_{i_1,\ldots,i_m} r, 1)]$$
$$+ \sum_{j=1}^{m} \sum_{\substack{i_1=1\\ i_k=1,2}} F_{l-1}(D_{i_1,\ldots,i_j} r, 2^{m-j}), \tag{5.9}$$

where $A_{i_1,\ldots,i_m}$, $B_{i_1,\ldots,i_m}$ and $D_{i_1,\ldots,i_j}$ are defined as follows:

$$A_{i_1,\ldots,i_m} = p_{i_1,\ldots,i_m} E_{i_1,\ldots,i_m}, \qquad B_{i_1,\ldots,i_m} = q_{i_1,\ldots,i_m} E_{i_1,\ldots,i_m},$$
$$D_{i_1,\ldots,i_j} = p_{i_1,\ldots,i_j} E_{i_1,\ldots,i_j}, \tag{5.10}$$

where $E_1 = E_2 = 1$ and the $E_{i_1,\ldots,i_j}$ are defined recursively by

$$E_{i_1,\ldots,i_j} = \begin{cases} p_{i_1,\ldots,i_{j-1}} E_{i_1,\ldots,i_{j-1}} & \text{if } i_j = 1, \\ q_{i_1,\ldots,i_{j-1}} E_{i_1,\ldots,i_{j-1}} & \text{if } i_j = 2, \end{cases} \tag{5.11}$$

and the $p_{i_1,\ldots,i_k}$, $q_{i_1,\ldots,i_k}$ are constants satisfying the following conditions like (5.8).

$$0 \le p_{i_1,\ldots,i_k},\ q_{i_1,\ldots,i_k} \le 1, \qquad p_{i_1,\ldots,i_k} + q_{i_1,\ldots,i_k} = 1. \tag{5.12}$$

We first establish some properties of $A_{i_1,\ldots,i_m}$, $B_{i_1,\ldots,i_m}$, $D_{i_1,\ldots,i_j}$, and $E_{i_1,\ldots,i_j}$.

$$\sum_{\substack{i_1=1\\ i_k=1,2}} E_{i_1,\ldots,i_j} = 1. \tag{5.13}$$

The proof of (5.13) follows from the fact that

$$\sum_{\substack{i_1=1\\ i_2=1,2}} E_{i_1,i_2} = E_{1,1} + E_{1,2} = p_1 + q_1 = 1 \quad \text{and}$$

$$\sum_{\substack{i_1=1\\ i_k=1,2}} E_{i_1,\ldots,i_j} = \sum_{\substack{i_1=1\\ i_k=1,2}} (p_{i_1,\ldots,i_{j-1}} E_{i_1,\ldots,i_{j-1}} + q_{i_1,\ldots,i_{j-1}} E_{i_1,\ldots,i_{j-1}})$$
$$= \sum_{\substack{i_1=1\\ i_k=1,2}} E_{i_1,\ldots,i_{j-1}}.$$

Note that by (5.10),

$$A_{i_1,\ldots,i_m} + B_{i_1,\ldots,i_m} = p_{i_1,\ldots,i_m} E_{i_1,\ldots,i_m} + q_{i_1,\ldots,i_m} E_{i_1,\ldots,i_m} = E_{i_1,\ldots,i_m}.$$

Hence by (5.13) we have

$$\sum_{\substack{i_1=1\\ i_k=1,2}} (A_{i_1,\ldots,i_m} + B_{i_1,\ldots,i_m}) = 1. \tag{5.14}$$

Similarly, we can show that

$$\sum_{\substack{i_1=1\\ i_k=1,2}} D_{i_1,\ldots,i_j} \le 1. \tag{5.15}$$

Furthermore, from (5.10), (5.11), and (5.12), it is trivial to see

$$A_{i_1, \ldots, i_m}, B_{i_1, \ldots, i_m}, D_{i_1, \ldots, i_j} \le 1.$$

Therefore, by (5.14),

$$\sum_{\substack{i_1=1\\ i_k=1,2}} [F_l(A_{i_1, \ldots, i_m} r, 1) + F_l(B_{i_1, \ldots, i_m} r, 1)] \le \sum_{\substack{i_1=1\\ i_k=1,2}} (lA_{i_1, \ldots, i_m} r + lB_{i_1, \ldots, i_m} r) = lr.$$

By (5.6) and (5.15), we have

$$\begin{aligned}
\sum_{j=1}^{m} \sum_{\substack{i_1=1\\ i_k=1,2}} F_{l-1}(D_{i_1, \ldots, i_j} r, 2^{m-j}) &\le \sum_{j=1}^{m} \sum_{\substack{i_1=1\\ i_k=1,2}} [(l-1)D_{i_1, \ldots, i_j} r + (\alpha_{l-1} D_{i_1, \ldots, i_j} r + \beta_{l-1} 2^{m-j})(\log r) m^{l-4}] \\
&\le \sum_{j=1}^{m} [(l-1)r + \alpha_{l-1} r + 2^{j-1} \beta_{l-1} 2^{m-j}](\log r) m^{l-4} \\
&\le [(\alpha_{l-1} + l - 1)r + (\beta_{l-1}/2) 2^m](\log r) m^{l-3}.
\end{aligned}$$

Hence by (5.9) we obtain that $F_l(r, 2^m) \le lr + (\alpha_l r + \beta_l 2^m)(\log r) m^{l-3}$, where $\alpha_l = \alpha_{l-1} + (l-1)$ and $\beta_l = \beta_{l-1}/2$. We have proven the theorem. □

ACKNOWLEDGMENTS. The authors wish to thank Dr. H. Freeman of the Polytechnic University of New York for bringing this problem to their attention and Dr. M. Schkolnick of Carnegie-Mellon University for his comments on the paper.

REFERENCES

1. KNUTH, D. E *The Art of Computer Programming, Vol 3: Sorting and Searching.* Addison-Wesley, Reading, Mass , 1973
2. KUNG, H. T. On the computational complexity of finding the maxima of a set of vectors Proc 15th Annual IEEE Symp. on Switching and Automata Theory, Oct. 1974, pp. 117–121 (also available as a Comput. Sci Dep. Rep , Carnegie-Mellon U , Pittsburgh, Pa , April 1974).
3. LUCCIO, F., AND PREPARATA, F. P. On finding the maxima of a set of vectors. Istituto di Scienze dell'Informazione, Università di Pisa, 56100 Pisa, Italy, Dec. 1973
4. YAO, F. F On finding the maximal elements in a set of plane vectors Comput. Sci. Dep Rep , U. of Illinois at Urbana-Champaign, Urbana, Ill., July 1974.

RECEIVED JANUARY 1975; REVISED APRIL 1975