ON THE COMPUTATIONAL COMPLEXITY OF FINDING THE MAXIMA OF A SET OF VECTORS

H. T. Kung

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa.

April, 1974

This research was supported in part by the National Science Foundation under Grant GJ32111 and the Office of Naval Research under Contract N0014-67-A-0314-0010, NR 044-422.

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 $U_1 \times U_2 \times \ldots \times U_d$. A partial ordering is defined on V in a natural way. We consider the problem of finding all maximal elements of V with respect to the partial ordering. 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 $d \geq 2$. Previous results are $C_d(n) \leq O(n \log_2 n)$ for $d = 2,3$. In this paper, we show

1. $C_d(n) \leq O(n(\log_2 n)^{d-2})$ for $d \geq 4$,
2. $C_d(n) \geq \lceil \log_2 n! \rceil$ for $d \geq 2$.

1. INTRODUCTION

Let $U_1, U_2, \ldots, U_d$ be totally ordered sets and let V be a set of n dimensional vectors in $U_1 \times U_2 \times \ldots \times U_d$. Let $x_i(v)$ denote the ith component of any vector 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 $i = 1, \ldots, d$, where $\leq_i$ is the total ordering on V_i. (We shall also write $\leq$ for $\leq_i$. The context should make clear the meaning of $\leq$.) We consider the problem of finding all maximal elements of V. The computational complexity of the problem is defined to be

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

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.

If d = 1, V is a totally ordered set. It is obvious that

$$C_1(n) = n-1.$$

If $d > 1$, V is a <u>partially</u> ordered set. It is not difficult to convince oneself that to find the maximal elements of a <u>general</u> partially ordered set, any algorithm requires order n^2 comparisons in the worst case. However, for the special partial ordering "$\leq$" on V, we can do better. Recently, Luccio and Preparata [1] have shown that

(1.1) $C_d(n) \leq O(n \log n)$ for d = 2 and 3.

(In this paper, all logarithms are to base 2 and all comparisons are between components of the vectors in V.)

It remained an open problem to show whether such reduction is attainable for $d \geq 4$. In this paper, we prove

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

and

(1.3) $C_d(n) \geq \lceil \log n! \rceil$ for $d \geq 2$.

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

In Section 2 we prove (1.3). In Section 3 we describe the basic recursive procedure for obtaining the upper bound in (1.2). This procedure leads to the problem of finding, from a given set, the elements which are not less than any element in another given set. Upper bounds on the number of comparisons for solving this problem are established by another recursive procedure, in the final section.

2. LOWER BOUND

Lemma 2.1

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

Proof

Let A_d denote an algorithm which finds the maxima of n d-dimensional vectors with at most $C_d(n)$ comparisons. It suffices to show that an algorithm A_{d-1} can be constructed from A_d such that A_{d-1} finds the maxima of n (d-1)-dimensional vectors and uses the same number of comparisons as A_d does. Let V_{d-1} be a set of n (d-1)-dimensional vectors. Define a set V_d of n d-dimensional vectors by

$$V_d = \{(v_1, v_2, \ldots, v_{d-1}, v_{d-1}) \mid (v_1, \ldots, v_{d-1}) \in V_{d-1}\}.$$

Let A_{d-1} be constructed from A_d by replacing every comparison between the dth components of two vectors in the algorithm A_d by the comparison between the (d-1)st components of the vectors. Then A_{d-1} and A_d will be same for the set V_d. Since A_d finds the maxima of V_d, so does A_{d-1}. Observe that $(v_1, \ldots, v_{d-1}, v_{d-1})$ is a maximum of V_d if and only if $(v_1, \ldots, v_{d-1})$ is a maximum of V_{d-1}. Therefore A_{d-1} finds the maxima of V_{d-1}. Furthermore, by the definition of A_{d-1}, it is clear that A_d and A_{d-1} use the same number of comparisons. We have proven the lemma. ■

Let S(n) denote the maximum number of comparisons used by the algorithm that sorts n records the fastest in the worst case. We have the following

Lemma 2.2

$S(n) \leq C_2(n)$.

<u>Proof</u>

Consider any algorithm which finds the maxima of n 2-dimensional vectors. Let $v_1,\ldots,v_n$ be 2-dimensional vectors such that for all i, $x_1(v_i)$ are distinct and for all i, j,

(2.1) $x_1(v_i) > x_1(v_j)$ if and only if $x_2(v_i) < x_2(v_j)$.

We apply the algorithm to the set $\{v_1,v_2,\ldots,v_n\}$.

For each v_i the algorithm must determine whether v_i is a maximal element or not. To prove v_i is maximal the algorithm must establish the relationships that, for each $j \neq i$, either $x_1(v_i) > x_1(v_j)$ or $x_2(v_i) > x_2(v_j)$. By (2.1) we know that all v_i are maximal elements. The algorithm must establish the relationships that either $x_1(v_i) > x_1(v_j)$ or $x_1(v_i) < x_1(v_j)$ between all pairs (i,j). This implies the algorithm will sort $x_1(v_1),\ldots,x_1(v_n)$. Therefore, $S(n) \leq C_2(n)$. ■

It is well known (for example, see Knuth [2, §5.3.1]) that

$$S(n) \geq \lceil \log n! \rceil.$$

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

<u>Theorem 2.1</u>

<u>For any $d \geq 2$,</u>

$$\underline{C_d(n) \geq C_{d-1}(n) \geq \ldots \geq C_2(n) \geq \lceil \log n! \rceil},$$

<u>so that about n log n comparisons are needed for finding the maxima of n d-dimensional vectors in the worst case.</u>

3. ALGORITHMS FOR FINDING THE MAXIMA OF A SET OF VECTORS

In this and the following sections we shall construct algorithms to achieve the upper bounds asserted in (1.2). In the rest of the paper, we assume that for any two vectors u,v in V, R or S, $x_i(u) \neq x_i(v)$ for all i. Under this assumption it will be easier to describe the ideas of the algorithms. The algorithms can be obviously modified if the assumption is removed (see [1]). 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 $v_1, \ldots, v_n$ so that

$$(3.1) \quad x_1(v_1) > x_1(v_2) > \ldots > x_1(v_n).$$

(Note that sorting 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 $\{v_1, \ldots, v_{n/2}\}$ and $\bar{S}$, the set of the maxima of $\{v_{n/2+1}, \ldots, v_n\}$. Observe that by (3.1) 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 $\leq$ any element in $\bar{R}$. Therefore, we have the following algorithm:

<u>Algorithm 3.1</u>

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

The number of comparisons required by Algorithm 3.1 depends on those required to find $\bar{T}$. Define

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

where R and S are any sets consisting of r and s, respectively, d-dimensional vectors, and $c_d(A,R,S)$ is the number of comparisons used by any algorithm A for finding the elements in S which are not $\le$ any element in R. Hence $\bar{T}$ can be found in $C_d(n/2,n/2)$ comparisons, since $|\bar{R}|,|\bar{S}| \le n/2$. Observe, however, that because of the relation (3.1), for $u \in \bar{R}$, $v \in \bar{S}$, $u \ge v$ if and only if $x_i(u) \ge x_i(v)$ for $i = 2,\ldots,d$. To find $\bar{T}$, 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 $C_{d-1}(n/2,n/2)$ instead of $C_d(n/2,n/2)$ comparisons. Therefore, by Algorithm 3.1, we obtain the following recurrence relation on $C_d(n)$:

(3.2) $C_d(n) \le 2C_d(n/2) + C_{d-1}(n/2,n/2)$.

In the following section, we shall show (Theorem 4.2) that

(3.3) $C_d(r,s) \le (\alpha_d r+\beta_d s)(\log r)(\log s)^{d-3} + dr$

for $d \ge 3$, where α_d and β_d are constants. By (3.3), we have

(3.4) $C_{d-1}(n/2,n/2) \le O(n(\log n)^{d-3})$ for $d \ge 4$.

Therefore, from (3.2) and (3.4), we obtain the main result of the paper:

<u>Theorem 3.1</u>

<u>$C_d(n) \le O(n(\log n)^{d-2})$ for $d \ge 4$.</u>

4. UPPER BOUNDS ON $C_d(r,s)$

This section deals with the proof of the following result: For $d \geq 3$

$$(4.1) \quad C_d(r,s) \leq (\alpha_d r+\beta_d s)(\log r)(\log s)^{d-3} + dr.$$

We shall first prove (4.1) for $d = 3$ and then use induction on d to prove (4.1) for all d. We shall first describe the key idea used in the induction.

Let R and S be two sets consisting of r and s, respectively, d-dimensional vectors. Assume $d \geq 4$. Without loss of generality we assume that the elements of R have been arranged as $u_1,\ldots,u_r$ and the elements of S as $v_1,\ldots,v_s$ so that

$$(4.2) \quad \begin{aligned} &x_1(u_1) > x_1(u_2) > \ldots > x_1(u_r), \\ &x_1(v_1) > x_1(v_2) > \ldots > x_1(v_s). \end{aligned}$$

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

$$(4.3) \quad x_1(u_k) \geq x_1(v_{s/2}) > x_1(u_{k+1}).$$

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

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

Recall that our problem is to find all elements in S which are not less than any element in R. We let $\begin{bmatrix} R \\ S \end{bmatrix}$ denote this problem. It is trivial to see that the problem $\begin{bmatrix} R \\ S \end{bmatrix}$ can be done by doing four subproblems, $\begin{bmatrix} R_1 \\ S_1 \end{bmatrix}$, $\begin{bmatrix} R_2 \\ S_1 \end{bmatrix}$, $\begin{bmatrix} R_1 \\ S_2 \end{bmatrix}$ and $\begin{bmatrix} R_2 \\ S_2 \end{bmatrix}$. Observe that the problem $\begin{bmatrix} R_2 \\ S_1 \end{bmatrix}$ is trivial, since by (4.2) and (4.3) we know there is no element in R_2 which is greater that any element in S_1. Thus, we do not have to worry about the problem $\begin{bmatrix} R_2 \\ S_1 \end{bmatrix}$. Furthermore, observe that by (4.2) and (4.3) 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 the previous section, to do the problem $\begin{bmatrix} R_1 \\ S_2 \end{bmatrix}$ we only have to consider (d-1)-dimensional vectors rather than d-dimensional vectors. Thus, to solve the problem $\begin{bmatrix} R \\ S \end{bmatrix}$ for d-dimensional vectors, we can instead solve the three subproblems:

1. the problem $\begin{bmatrix} R_1 \\ S_1 \end{bmatrix}$ for d-dimensional vectors,
2. the problem $\begin{bmatrix} R_2 \\ S_2 \end{bmatrix}$ for d-dimensional vectors,
3. the problem $\begin{bmatrix} R_1 \\ S_2 \end{bmatrix}$ for (d-1)-dimensional vectors.

Therefore, we have shown

(4.4) $C_d(r,s) \le C_d(k,s/2) + C_d(r-k,s/2) + C_{d-1}(k,s/2)$.

In the rest of the section we shall first prove (4.1) for d = 3 then use (4.4) to prove (4.1) for general d by induction.

Theorem 4.1

$C_3(r,s) \le (\alpha_3 r+\beta_3 s)(\log r)$

for constants α_3 and β_3.

Proof

Let $\{v_1,\ldots,v_s\}$ be the elements of S. We establish the theorem by exhibiting an algorithm which is adapted from a result in [1].

Algorithm 4.1

This algorithm finds all elements in S which are not less than any element in R for d = 3.

1. Arrange the elements of R as a sequence $u_1,\ldots,u_r$ such that

$$x_1(u_1) > x_1(u_2) > \ldots > x_1(u_r).$$

2. Arrange the elements of S as a sequence $v_1,\ldots,v_s$ such that if a(j) is the largest value of the index i such that $x_1(u_i) \ge x_1(v_j)$ then

$$a(1) \le a(2) \le \ldots \le a(s).$$

($x_1(u_0)$ is defined to be ∞.)

3. Set $j \leftarrow 1$.

4. If $a(j) = 0$, v_j is not less than any element in R and go to step 9.

5. Construct $T_{a(j)}$, the set of maxima of $\{(x_2(u_i), x_3(u_i)) \mid i = 1,\ldots,a(j)\}$, and arrange its elements as a sequence $w_1,\ldots,w_\nu$ such that

$$x_2(w_1) > x_2(w_2) > \ldots > x_2(w_\nu).$$

6. If $x_2(v_j) > x_2(w_1)$, v_j is not less than any element in R and go to step 9.

7. Determine the largest value i^* of the index i such that

$$x_2(w_i) \geq x_2(v_j) \text{ for } w_i \in T_{a(j)}.$$

8. If $x_3(v_j) > x_3(w_{i^*})$, v_j is not less than any element in R and go to step 9.

9. If $j < s$, $j \leftarrow j+1$ and return to step 4.

10. Terminate the algorithm.

Step 5 can be efficiently performed by using, for example, an AVL binary tree [2, §6.2.3] as the information structure which stores the elements of $T_{a(j)}$. For details of this information structure and for the proof of the validity of the algorithm, see [1]. We now estimate the number of comparisons used in the algorithm. It is shown in [1] that the total number of comparisons needed for step 5 is $\leq O(r \log r)$. Clearly, step 1 also takes $O(r \log r)$ comparisons. By using the binary search technique, steps 2 and 7 take $O(s \log r)$ comparisons. Hence the total number of comparisons for the whole algorithm is $O(r \log r) + O(s \log r)$. ■

Theorem 4.2

For $d \geq 3$,

(4.5) $C_d(r,s) \leq (\alpha_d r+\beta_d s)(\log r)(\log s)^{d-3} + dr,$

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

(α_3, β_3 are given by Theorem 4.1.)

Proof

We shall prove the theorem by induction on d. By Theorem 4.1, (4.5) holds for $d = 3$. Assume that (4.5) holds for $d = \ell-1$. Without loss of generality, we assume that $s = 2^m$ for some positive integer m. Then we have

(4.6) $C_{\ell-1}(r,2^m) \leq (\alpha_{\ell-1}r+\beta_{\ell-1}2^m)(\log r)m^{\ell-4} + (\ell-1)r.$

By (4.4) we know that there exist $p_1 (= k/r)$ and $q_1 (= (r-k)/r)$ such that

(4.7) $C_\ell(r,2^m) \leq C_\ell(p_1r,2^{m-1}) + C_\ell(q_1r,2^{m-1}) + C_{\ell-1}(p_1r,2^{m-1}).$

Note that

(4.8) $0 \leq p_1, q_1 \leq 1$ and $p_1 + q_1 = 1.$

We shall use (4.6) and (4.7) to prove that

$$C_\ell(r,2^m) \leq (\alpha_\ell r+\beta_\ell 2^m)(\log r)\ m^{\ell-3} + \ell r,$$

that is, (4.5) for $d = \ell$. The proof below is elementary but tedious. The essential idea is to apply (4.7) recursively. It is not difficult to see from (4.7) we can prove that

$$(4.9)\quad C_\ell(r,2^m) \le \sum_{\substack{i_1=1\\ i_k=1,2}} [C_\ell(A_{i_1,\ldots,i_m} r,1) + C_\ell(B_{i_1,\ldots,i_m} r,1)] + \sum_{j=1}^{m} \sum_{\substack{i_1=1\\ i_k=1,2}} C_{\ell-1}(D_{i_1,\ldots,i_j} r, 2^{m-j}),$$

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

$$(4.10)\quad \begin{cases} A_{i_1,\ldots,i_m} = p_{i_1,\ldots,i_m} E_{i_1,\ldots,i_m}, \\ B_{i_1,\ldots,i_m} = q_{i_1,\ldots,i_m} E_{i_1,\ldots,i_m}, \\ D_{i_1,\ldots,i_j} = p_{i_1,\ldots,i_j} E_{i_1,\ldots,i_j}, \end{cases}$$

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

$$(4.11)\quad E_{i_1,\ldots,i_j} = \begin{cases} p_{i_1,\ldots,i_{j-1}} E_{i_1,\ldots,i_{j-1}} & \text{if } i_j = 1, \\ q_{i_1,\ldots,i_{j-1}} E_{i_1,\ldots,i_{j-1}} & \text{if } i_j = 2, \end{cases}$$

and the $p_{i_1,\ldots,i_k}$, $q_{i_1,\ldots,i_k}$ are constants satisfying the following conditions like (4.8):

$$(4.12)\quad \begin{cases} 0 \le p_{i_1,\ldots,i_k},\ q_{i_1,\ldots,i_k} \le 1, \\ p_{i_1,\ldots,i_k} + q_{i_1,\ldots,i_k} = 1. \end{cases}$$

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 $E_{i_1,\ldots,i_j}$.

$$(4.13)\quad \sum_{\substack{i_1=1\\ i_k=1,2}} E_{i_1,\ldots,i_j} = 1.$$

The proof of (4.13) follows from the fact that $\sum_{\substack{i_1=1\\ i_2=1,2}} E_{i_1,i_2} = E_{1,1} + E_{1,2}$ $= p_1 + q_1 = 1$ 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 (4.10),

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

Hence by (4.13), we have

$$(4.14) \quad \sum_{\substack{i_1=1\\ i_k=1,2}} (A_{i_1,\ldots,i_m} + B_{i_1,\ldots,i_m}) = 1.$$

Similarly, we can show that

$$(4.15) \quad \sum_{\substack{i_1=1\\ i_k=1,2}} D_{i_1,\ldots,i_j} \le 1.$$

Furthermore, from (4.10), (4.11) and (4.12), 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 (4.14),

$$\sum_{\substack{i_1=1\\ i_k=1,2}} [C_\ell(A_{i_1,\ldots,i_m} r,1) + C_\ell(B_{i_1,\ldots,i_m} r,1)]$$

$$\le \sum_{\substack{i_1=1\\ i_k=1,2}} (\ell A_{i_1,\ldots i_m} r + \ell B_{i_1,\ldots i_m} r)$$

$$= \ell r.$$

By (4.6) and (4.15), we have

$$\sum_{j=1}^{m} \sum_{\substack{i_1=1\\ i_k=1,2}} C_{\ell-1}(D_{i_1,\ldots,i_j} r, 2^{m-j})$$

$$\le \sum_{j=1}^{m} \sum_{\substack{i_1=1\\ i_k=1,2}} [(\ell-1)D_{i_1,\ldots,i_j} r + (\alpha_{\ell-1} D_{i_1,\ldots,i_j} r + \beta_{\ell-1} 2^{m-j})(\log r)m^{\ell-4}]$$

$$\le \sum_{j=1}^{m} [(\ell-1)r + \alpha_{\ell-1} r + 2^{j-1}\beta_{\ell-1} 2^{m-j}](\log r)m^{\ell-4}$$

$$\le [(\alpha_{\ell-1} + \ell-1)r + (\beta_{\ell-1}/2)2^m](\log r)m^{\ell-3}.$$

Hence by (4.9) we obtain that

$$C_\ell(r,2^m) \le \ell r + (\alpha_\ell r + \beta_\ell 2^m)(\log r)m^{\ell-3}.$$

where $\alpha_\ell = \alpha_{\ell-1} + (\ell-1)$ and $\beta_\ell = \beta_{\ell-1}/2$.

We have proven the theorem. ■

ACKNOWLEDGMENT

I want to thank M. Schkolnick for his comments on this paper.

REFERENCES

[1] F. Luccio and F. P. Preparata, On Finding the Maxima of a Set of Vectors, Instituto di Scienze dell'Informazione, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy, 1973.

[2] D. E. Knuth, The Art of Computer Programming, Vol. 3, Sorting and Searching, Addison-Wesley Publishing Company, Reading, Mass., 1973.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ON THE COMPUTATIONAL COMPLEXITY OF FINDING MAXIMA OF A SET OF VECTORS		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) H. T. Kung		8. CONTRACT OR GRANT NUMBER(s) N0014-67-A-0314-0010 NR 044-422
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Department of Computer Science Pittsburgh, Pennsylvania 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Washington, D. C. 20360		12. REPORT DATE April, 1974
		13. NUMBER OF PAGES 17
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Scientific Officer, Code 432 Mathematics Programs Office of Naval Research; Washington, D. C. 20360		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Let $U_1, U_2, \ldots, U_d$ be totally ordered sets and let V be a set of n d-dimensional vectors in $U_1 \times U_2 \times \ldots \times U_d$. A partial ordering is defined on V in a natural way. We consider the problem of finding all maximal elements of V with respect to the partial ordering. 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 $C_d(n) \leq 0(n^2)$ for $d \geq 2$.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 continued:

Previous results are $C_d(n) \leq O(n \log_2 n)$ for $d = 2,3$. In this paper, we show

1. $C_d(n) \leq (n(\log_2 n)^{d-2})$ for $d \geq 4$,
2. $C_d(n) \geq \lceil \log_2 n! \rceil$ for $d \geq 2$.