INFORMATION AND CONTROL 60, 70–85 (1984)

The Complexity of Sorting on Distributed Systems

MICHAEL C. LOUI*

Department of Electrical and Computer Engineering and Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, Urbana, Illinois 61801

The sorting problem is to arrange N values in a distributed system of N processors into sorted order. Let the values be in $\{0,..., L\}$. Every sorting algorithm requires $\Omega(N^2 \lg(L/N)/\lg N)$ messages on a bidirectional ring with N processors. Every sorting algorithm requires $\Omega(N^{3/2} \lg(L/N)/\lg N)$ messages on a square mesh with N processors. A novel sorting algorithm for unidirectional rings achieves the first lower bound. © 1984 Academic Press, Inc.

1. INTRODUCTION

To cooperate to solve a problem, the processors in a distributed computing system must communicate among themselves. For both large computer networks and VLSI architectures, however, the inclusion of a shared memory to facilitate interprocessor communication is usually infeasible. The processors in these distributed systems can communicate only by sending messages via a network. Thus to exploit fully the potential efficiency of a distributed system, an effective algorithm should minimize the message traffic in order to minimize the computation time.

The problem of finding an extremum—also called electing a leader—in a distributed system is well solved (Dolev *et al.*, 1982; Matsushita, 1983; Peterson, 1982). Efficient distributed algorithms have also been proposed for determining medians (Frederickson, 1983; Matsushita, 1983; Rodeh, 1982; Santoro & Sidney, 1982), minimum spanning trees (Gallager *et al.*, 1983), shortest paths (Chandy & Misra, 1982), and maximum flows (Segall, 1982).

It is natural to ask whether these algorithms achieve the smallest possible message traffic for each problem. Let lg denote the logarithm taken to base 2. For the extrema-finding problem Burns (1980) established a lower bound of $0.25\, N \lg N$ messages in the worst case on a bidirectional ring. For

* Supported by the Joint Services Electronics Program (U.S. Air Force, U.S. Army, U.S. Navy) under Contract N00014-79-C-0424 and by the National Science Foundation under Grant MCS-8217445. A preliminary version of this paper was presented at the 1984 Conference on Information Sciences and Systems, Princeton, N. J.

0019-9958/84 $3.00

the computation of minimum spanning trees Santoro (1982) and Korach *et al.* (1984) obtained $\Omega(N \lg N)$ lower bounds on messages on various networks.

In this paper I determine the amount of communication required to arrange N values into sorted order. Let the values be in $\{0,..., L\}$. One might surmise that on a bidirectional ring of N processors, an algorithm that sorts N values transmits each value, represented by $\lg L$ bits, roughly $N/2$ times in the worst case. This intuitive argument suggests that every sorting algorithm has bit complexity $\Omega(N^2 \lg L)$. It is thus surprising that the actual bit complexity of sorting on a ring is $\Theta(N^2 \lg(L/N))$.

I prove that on a bidirectional ring with N processors every sorting algorithm requires $\Omega(N^2 \lg(L/N))$ bits among its messages. If every message has $O(\lg N)$ bits, then this lower bound implies that $\Omega(N^2 \lg(L/N)/\lg N)$ messages are necessary. Furthermore, on a square mesh with N processors every sorting algorithm requires $\Omega(N^{3/2} \lg(L/N)/\lg N)$ messages. I present a simple sorting algorithm on rings that uses $O(N^2 \lg(L/N))$ bits, achieving the first lower bound. Within a constant multiplicative factor, this algorithm is optimal.

The algorithm of Korach *et al.* (1982) uses $O(N^2)$ messages to rank the values in a network, but does not rearrange the values. Rotem *et al.* (1983) studied sorting problems on general networks without the a priori bound L.

Section 2 defines the computational model and the sorting problem. Section 3 establishes the lower bounds on bits and messages. Section 4 describes the optimal sorting algorithm for rings.

2. Definitions

2.1. *Computational Model*

This paper adopts the model of distributed computation developed by Santoro (1981). The model is asynchronous, requires decentralized control, admits no shared memory, and permits data transfers only on a communication network.

The distributed computing system comprises N identical processors connected via a communication network. A *link* is an ordered pair of processors, and a *network* is a set of links. A *message* is a nonempty binary string. Processor x can send a message directly to processor y if and only if link (x, y) is in the network.

Every processor runs the same program. Initially, each processor knows only the links that involve it and the overall topology of the network—for example, whether the network is a ring or a mesh.

Each of the processors has a distinct number representable with $O(\lg N)$ bits called its *initial value*. The processors exchange messages to compute a

function of these values. At the end of the computation, every processor has a *final value.*

The transmission of a message incurs an unpredictable but finite delay, and the state of a processor changes whenever it receives a message. At processor y every message is placed on a queue when it arrives. Messages that arrive simultaneously are queued arbitrarily. Messages sent on the same link (x, y) arrive at y in the same order as they were sent.

To each processor assign an integer p, $0 \leqslant p < N$. For simplicity, to obviate the phrase *mod* N, also assign the integers $p + N, p + 2N, \ldots$, to the same processor p. The assignment of integers to processors is used only for clarity of exposition; since the processors are identical, processor p does not actually have immediate access to the number p. If integer p is assigned to processor x and q is assigned to processor y, then the link (x, y) will be written (p, q). The phrase "processor p" also denotes processor x.

I consider several topologies for the communication network. In a *bidirectional ring*, processor p can send messages only to processors $p - 1$ and $p + 1$. Formally, the bidirectional ring has links $(p, p - 1)$ and $(p, p + 1)$ for every p. In a *unidirectional ring*, processor p can send messages only to processor $p + 1$.

The *discrete torus* is a square mesh with wrap-around connections. Let $N = M^2$. For each processor p, $0 \leqslant p < N$, write $p = i + jM$ such that $0 \leqslant i < M$ and $0 \leqslant j < M$. This equation defines a bijection between $\{0, \ldots, N - 1\}$ and pairs $\langle i, j \rangle$ in $\{0, \ldots, M - 1\}^2$. Processor p can also be called processor $\langle i, j \rangle$. In the discrete torus, for every i and j there are links $(\langle i, j \rangle, \langle i + 1, j \rangle)$, $(\langle i, j \rangle, \langle i - 1, j \rangle)$, $(\langle i, j \rangle, \langle i, j + 1 \rangle)$, $(\langle i, j \rangle, \langle i, j - 1 \rangle)$, where $i \pm 1$ and $j \pm 1$ are taken modulo M. For example, ILLIAC IV had the topology of a discrete torus with $N = 64$.

Each processor has $O(\lg N)$ bits of storage. This limit precludes trivial algorithms. For instance, on a fully interconnected network, if processor p had unbounded storage, then the other processors could ship their initial values to processor p, which could compute all the final values.

The limitation on storage implies that every message has $O(\lg N)$ bits. There are no other constraints on the form of messages; in particular, a message need not be one of the initial values. The limit on message length prohibits arbitrarily long messages. If messages of unbounded length were permitted, then for every solvable problem there would be an algorithm that used $O(N)$ messages after it elects a leader. For example, on a unidirectional ring N long messages would suffice to send all the initial values to the leader, which would perform the computation, and N more long messages would suffice to distribute the final values from the leader to the other processors.

To evaluate the performance of a distributed algorithm, I assume that the processing time within a processor is negligible. Indeed, because computation within a processor generally proceeds much faster than transmission of

messages, communication steps often dominate the running time of an algorithm (Lint & Agerwala, 1981). The two performance criteria used in this paper are expressed as functions of N. The *message complexity* of an algorithm assigns to each N the maximum number of messages used by the algorithm on distributed systems with N processors. The *bit complexity* of an algorithm assigns to each N the maximum number of bits among messages used by the algorithm on systems with N processors. Abelson (1980), Ja' Ja' and Prasanna Kumar (1984), Papadimitiou and Sipser (1984), and Yao (1979) studied this complexity measure for systems with only two processors.

2.2. *The Sorting Problem*

Initially, for every p, processor p has a distinct initial value $IV(p)$. A *sorting algorithm* rearranges these values so that at the end of the computation, processor p has a final value $FV(p)$ such that for some b,

$$FV(b+i) < FV(b+i+1) \qquad \text{for all} \quad 0 \leqslant i < N-2.$$

Call processor b the *base*. The base processor has the smallest final value.

For a sorting algorithm A and a distribution of initial values, the *destination* of a value v is the processor p such that at the end of the computation of A, the final value at processor p is $FV(p) = v$. The destination of a value depends on which processor becomes the base.

3. Lower Bounds

3.1. *Preliminaries*

Define SBS to be the set of all finite sequences of binary strings $(\beta_1, \ldots, \beta_k)$ in which every component β_i is a nonempty binary string.

LEMMA 1. *Fewer than* $4^{b+1}/6$ *sequences in SBS have at most b bits*

Proof. Let $n(b)$ be the number of sequences with a total of b bits. Evidently $n(1) = 2$. Furthermore, from a sequence s with b bits one can obtain a sequence with $b-1$ bits by deleting the leftmost bit of s; there are four situations:

(1) The leftmost string in s is 0.

(2) The leftmost string in s is 1.

(3) The leftmost string in s has two or more bits and begins with 0.

(4) The leftmost string in s has two or more bits and begins with 1.

Consequently,

$$n(b) = 4n(b-1),$$
$$n(1) = 2,$$

hence

$$n(b) = 4^b/2.$$

It follows that

$$n(1) + \cdots + n(b) = \frac{1}{2}(4 + \cdots + 4^b) = \frac{1}{2}\frac{4^{b+1} - 4}{4-1} < 4^{b+1}/6. \quad \blacksquare$$

LEMMA 2. *Let S be a set of σ different sequences in SBS. The total number of bits among the sequences in S is at least $\frac{1}{6}\sigma \lg \sigma$.*

Proof. Set

$$b = 1 + \lfloor \tfrac{1}{2} \lg \sigma \rfloor.$$

By definition, $\frac{2\sigma}{3} \geqslant \frac{4^b}{6}$. Lemma 1 implies that at least $\frac{1}{3}$ of the sequences in S have at least b bits each. The total number of bits among these sequences is at least

$$\tfrac{1}{3}\sigma b \geqslant \tfrac{1}{6}\sigma \lg \sigma. \quad \blacksquare$$

In a distributed computing system call the function $p \to IV(p)$ a *distribution*. If P is a set of processors in the system, then the restriction of a distribution d to P is the *distribution for P induced by d*. Distributions d_1 and d_2 *agree* on P if their values on P are the same; equivalently, the restriction of d_1 to P is identical to the restriction of d_2 to P.

Consider a partition of the processors in a system into two sets P_1 and P_2. The *cut* c induced by this partition is the set of all links (x, y) for which either $x \in P_1$ and $y \in P_2$ or $x \in P_2$ and $y \in P_1$.

Let A be a distributed algorithm and let c be a cut. During the computation of A for a distribution d consider the sequence of messages transmitted on links in c in the order in which they were sent. To each message μ of this sequence append a string of $\lceil \lg |c| \rceil$ bits that identifies on which of the $|c|$ links in c the message μ was sent. Call the resulting sequence of binary strings the *signature* of A for d on c. A signature is a sequence in SBS. Note that the signature is not unique: Since the system is asynchronous, one distribution might have several signatures that correspond to different orderings of messages on c. For the validity of the subsequent proofs it suffices to choose one signature for each distribution.

LEMMA 3. *Let c be a cut induced by a partition of the processors into sets P_1 and P_2. Let D be a collection of distributions that agree on all processors in P_2. If algorithm A has fewer than $|D|$ different signatures on c for the distributions in D, then for two different distributions in D, algorithm A produces the same set of final values in P_2.*

Proof. By hypothesis, there are different distributions d_1 and d_2 in D for which A has the same signature on c. For both d_1 and d_2 the computation of A sends the same messages on the same links in c in the same order. From the viewpoint of the processors in P_2 the computation of A for d_1 is the same as its computation for d_2. Consequently, at the end of both computations the final values in P_2 are the same. ∎

LEMMA 4. *In a signature with b bits on cut c the number of message bits is at least*

$$\frac{b}{1 + \lceil \lg |c| \rceil}.$$

Proof. Let b' be the number of message bits in the signature and m be the number of messages. Since $m \leqslant b'$,

$$b = b' + m\lceil \lg |c| \rceil \leqslant (1 + \lceil \lg |c| \rceil)\, b'. \quad \blacksquare$$

3.2. *The Ring*

This subsection establishes lower bounds on the complexity of sorting in a bidirectional ring. These lower bounds apply a fortiori to unidirectional rings too.

THEOREM 1. *On a bidirectional ring of N processors with initial values in $\{0,\ldots,L\}$, every sorting algorithm has bit complexity $\Omega(N^2 \lg(L/N))$.*

In particular, if $L = 2N$, then $\Omega(N^2)$ bits among messages are necessary. If $L = N \lg N$, then $\Omega(N^2 \lg\lg N)$ bits are necessary. If $L = N^\alpha$ for a constant $\alpha > 1$, then $\Omega(N^2 \lg N)$ bits are necessary.

Proof. Consider an algorithm A that arranges values into sorted order. The main idea is the following: For some distribution of initial values, approximately $N/4$ initial values must migrate at least distance $N/16$ to their destinations. But the destination of a value depends on the processor that becomes the base, which in turn depends on the initial values. The bulk of this proof overcomes this circularity.

Define $R = L/N$. Without loss of generality, assume that R is an integer and that $N - 1$ is divisible by 16. Define a collection of R^N distributions of

initial values as follows. For $p = 0,..., N-1$ the initial value at processor p satisfies

$$\begin{aligned} (p/2)\,R &\leqslant IV(p) < (p/2+1)\,R && \text{if } p \text{ is even,} \\ ((N+p)/2)\,R &\leqslant IV(p) < ((N+p)/2+1)\,R && \text{if } p \text{ is odd.} \end{aligned} \tag{1}$$

EXAMPLE. $(N = 17, R = 3)$.

$IV(0) = 0$	$IV(5) = 33$	$IV(9) = 39$	$IV(13) = 45$
$IV(1) = 27$	$IV(6) = 9$	$IV(10) = 15$	$IV(14) = 21$
$IV(2) = 3$	$IV(7) = 36$	$IV(11) = 42$	$IV(15) = 48$
$IV(3) = 30$	$IV(8) = 12$	$IV(12) = 18$	$IV(16) = 24$
$IV(4) = 6$			

Since the ring has N processors, there are only N possible bases. Therefore there is a base b such that for at least R^N/N of the distributions defined by (1), processor b becomes the base during the computation of algorithm A. Let D be this collection of R^N/N distributions.

Put

$$\begin{aligned} q &= 6(N-1)/16 + 1 + 2b \\ r &= 10(N-1)/16 + 2b \\ s &= 11(N-1)/16 + 1 + 2b \\ t &= 5(N-1)/16 + 2b. \end{aligned} \tag{2}$$

Let P_1 be the set of $(N-1)/4$ processors $q, q+1,..., r$. Let P_2 be the set of $5(N-1)/8$ processors $s, s+1,..., t$. See Fig. 1.

For the distributions in D processor b becomes the base. Definition (1) implies that for every p the destination of $IV(p)$ is

$$\begin{aligned} &\text{processor } b + p/2 && \text{if } p \text{ is even,} \\ &\text{processor } b + (N+p)/2 && \text{if } p \text{ is odd.} \end{aligned}$$

It follows that for $p = q, q+1,..., r$, the destination of $IV(p)$ is among processors

$$b + (N+q)/2 = s, s+1,..., b + r/2 = t.$$

Thus each of the initial values in P_1 must travel at least distance $1 + (N-1)/16$ to its destination in P_2.

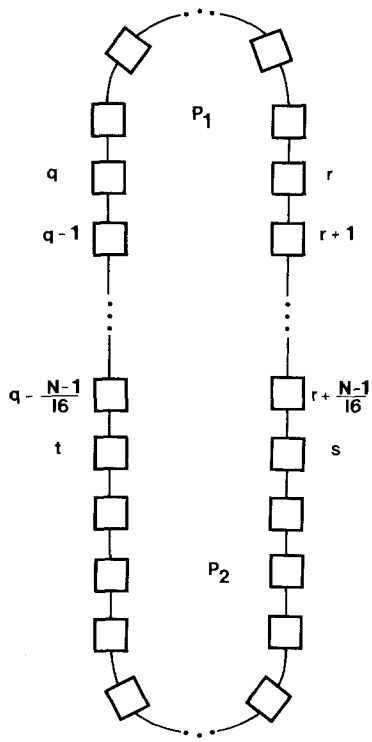

FIGURE 1

Let P_1' be the set of $(3N+1)/4$ processors that are not in P_1. There are $R^{(3N+1)/4}$ distributions for P_1' consistent with (1). Consequently there is a distribution d_0 for P_1' such that d_0 is induced by at least

$$\frac{R^N/N}{R^{(3N+1)/4}} = \frac{R^{(N-1)/4}}{N}$$

of the distributions in D. Let D' be a subset of $R^{(N-1)/4}/N$ of these distributions. The distributions in D' agree on P_1'. Let $\sigma = |D'| = R^{(N-1)/4}/N$.

Let C be the following set of $1 + (N-1)/16$ pairwise disjoint cuts, which separate P_1 from P_2:

$$\{(q, q-1), (q-1, q), (r, r+1), (r+1, r)\},$$

$$\{(q-1, q-2), (q-2, q-1), (r+1, r+2), (r+2, r+1)\},\ldots,$$

$$\left\{\left(q - \frac{(N-1)}{16}, t\right), \left(t, q - \frac{(N-1)}{16}\right), \left(r + \frac{(N-1)}{16}, s\right), \left(s, r + \frac{(N-1)}{16}\right)\right\}.$$

For each of the $1 + (N-1)/16$ cuts c in C the number of different signatures of A on c must be at least $|D'| = \sigma$ because otherwise, by Lemma 3, there would be two different distributions in D' that would yield the same set of final values in P_2. Let $n(c, d)$ be the total number of bits in messages used by A on links in c for the initial distribution d. By Lemmas 2, 3, and 4, for each of the cuts c in C,

$$\sum_{d \in D'} n(c, d) \geqslant \frac{\sigma \lg \sigma}{6(1 + \lceil \lg |c| \rceil)} = \frac{1}{18} \sigma \lg \sigma,$$

since $|c| = 4$. It follows that

$$\sum_{c \in C} \sum_{d \in D'} n(c, d) \geqslant \left(1 + \frac{N-1}{16}\right) \frac{1}{18} \sigma \lg \sigma.$$

Therefore there exists a d^* in D' such that

$$\begin{aligned} \sum_{c \in C} n(c, d^*) &\geqslant \left(\sum_{d \in D'} \sum_{c \in C} n(c, d) \right) \Big/ \sigma \\ &\geqslant \frac{1}{18} \left(1 + \frac{N-1}{16}\right) \lg \sigma \\ &= \frac{1}{18} \left(\frac{N+15}{16}\right) \left(\frac{N-1}{4} \lg R - \lg N\right). \\ &= \Omega(N^2 \lg R). \end{aligned}$$

Ergo, the bit complexity of A is $\Omega(N^2 \lg (L/N))$. ∎

The proof of Theorem 1 resembles the proofs of Thompson (1979), who established time–space tradeoffs in VLSI. As Lipton and Sedgewick (1981) have observed, Thompson's technique is analogous to a crossing sequence argument for Turing machine complexity (Hopcroft & Ullman, 1979).

Since this proof does not require that the system be asynchronous, the lower bound applies to synchronous rings too. Also, the lower bound holds even when the processors have unbounded storage.

Since every message has $O(\lg N)$ bits, Theorem 1 implies the following lower bound on message complexity.

THEOREM 2. *On a bidirectional ring of N processors with initial values in $\{0,..., L\}$, every sorting algorithm has message complexity $\Omega(N^2 \lg(L/N)/\lg N)$.*

3.3. *The Discrete Torus*

A modification of the proof of Subsection 3.2 would yield an $\Omega(N^{3/2} \lg(L/N)/\lg N)$ lower bound on the bit complexity of sorting on the

discrete torus. This section establishes a stronger result: Every sorting algorithm has message complexity $\Omega(N^{3/2} \lg(L/N)/\lg N)$.

Let $SBS(Q)$ denote the set of finite sequences of binary strings $(\beta_1,\ldots,\beta_k)$ in which every component β_i is a string of at most $\lg Q$ bits.

LEMMA 5. *Fewer than $2(2Q)^k$ sequences in $SBS(Q)$ have at most k components.*

Proof. Since each component in a sequence in SBS(Q) has at most $\lg Q$ bits, the number of possible components is

$$2 + \cdots + 2^{\lg Q - 1} + 2^{\lg Q} < 2Q.$$

It follows that the number of sequences in SBS(Q) with at most k components is smaller than

$$2Q + (2Q)^2 + \cdots + (2Q)^k < 2(2Q)^k. \quad \blacksquare$$

LEMMA 6. *Let S be a set of σ different sequences in $SBS(Q)$. When each occurrence of a string is counted, the total number of strings among the sequences in S is at least*

$$\frac{4\sigma \lg(\sigma/10)}{5 \lg(2Q)}.$$

Proof. Set

$$k = 1 + \left\lfloor \frac{\lg(\sigma/10)}{\lg(2Q)} \right\rfloor.$$

By definition,

$$\lg(\sigma/10) \geqslant (k-1)\lg(2Q),$$
$$\sigma/5 \geqslant 2(2Q)^{k-1}.$$

Lemma 5 implies that at least $\frac{4}{5}$ of the sequences in S have at least k components each. The total number of strings among these sequences is at least

$$\frac{4}{5}\sigma k \geqslant \frac{4\sigma \lg(\sigma/10)}{5 \lg(2Q)}. \quad \blacksquare$$

Consider a discrete torus of N processors with initial values in $\{0,\ldots,L\}$. Let A be a sorting algorithm on the discrete torus that uses messages with at most $\alpha \lg N$ bits each. Let $M = N^{1/2}$ and $R = L/N$. Suppose the initial value

at every processor p satisfies (1) in Subsection 3.2. For the q, r, s, and t defined by (2), the values among processors $q, q+1,\ldots, r$ must migrate to their destinations at processors $s, s+1,\ldots, t$. Let P_1 be the set of processors $q, q+1,\ldots, r$. Let P_2 be the set of processors $s, s+1,\ldots, t$. Let D' be the set of distributions defined in the proof of Theorem 1, and let $\sigma = |D'| = R^{(N-1)/4}/N$.

It is easy to find a set C' of $\lceil (N-1)/(16M) \rceil$ pairwise disjoint cuts that separate P_1 and P_2. See Fig. 2. For every c in C', since $|c| = 4M = 4N^{1/2}$, the length of each binary string in a signature on c is at most

$$\alpha \lg N + \lceil \lg |c| \rceil \leqslant \lg(4N^{\alpha+1}),$$

hence every signature on c is in $\mathrm{SBS}(4N^{\alpha+1})$. By Lemma 3, for each of the cuts c in C' the number of different signatures of A on c must be at least $|D'| = \sigma$. Let $m(c, d)$ be the number of messages used by A on links in c for the initial distribution d. By Lemma 6, for each c in C',

$$\sum_{d \in D'} m(c, d) \geqslant \frac{4\sigma \lg(\sigma/10)}{5 \lg(8N^{\alpha+1})}.$$

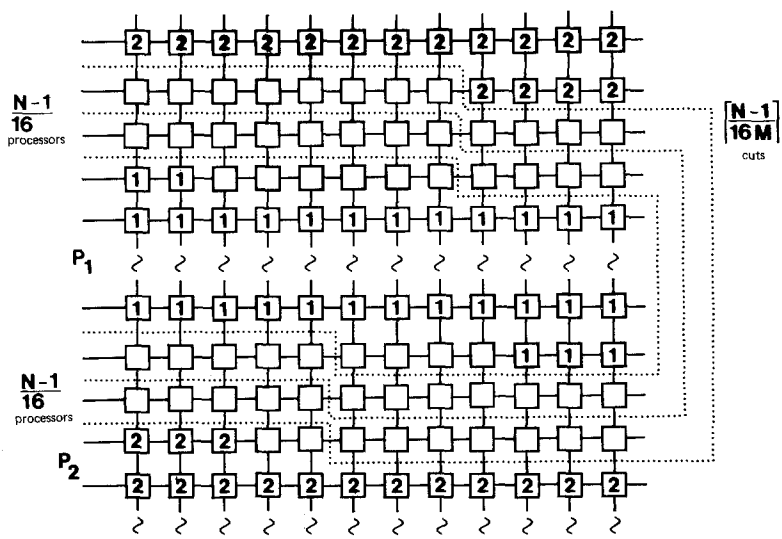

FIG. 2. Processors labeled "1" are in P_1, processors labeled "2" are in P_2.

Therefore there exists a d^* in D' such that

$$\begin{aligned}\sum_{c\in C'} m(c,d^*) &\geqslant \left(\sum_{d\in D'}\sum_{c\in C'} m(c,d)\right)\Big/\sigma \\ &\geqslant \left(\frac{N-1}{16M}\right)\frac{4\lg(\sigma/10)}{5\lg(8N^{\alpha+1})} \\ &= \left(\frac{N-1}{16M}\right)\frac{(N-1)\lg R - 4\lg(10N)}{5\lg(8N^{\alpha+1})} \\ &\geqslant \frac{(N^2-2N)\lg R}{80M\lg(8N^{\alpha+1})} - \frac{(N-1)\lg(10N)}{20M\lg(8N^{\alpha+1})} \\ &= \Omega\left(\frac{N^{3/2}\lg R}{\lg N}\right).\end{aligned}$$

THEOREM 3. *On a discrete torus of N processors with initial values in $\{0,...,L\}$, every sorting algorithm has message complexity $\Omega(N^{3/2}\lg(L/N)/\lg N)$.*

Since each message has at least one bit, Theorem 3 implies the following lower bound on bit complexity.

THEOREM 4. *On a discrete torus of N processors with initial values in $\{0,...,L\}$, every sorting algorithm has bit complexity $\Omega(N^{3/2}\lg(L/N)/\lg N)$.*

4. OPTIMAL SORTING

4.1. *Representing a Sorted Subset*

Let $S=\{a_1,...,a_k\}$ be a nonempty subset of $\{0,...,L\}$. Index the elements of S so that $a_1<a_2<\cdots<a_k$. Let $a_0=0$. The set S can be represented by the sequence $(a_1-a_0, a_2-a_1,..., a_k-a_{k-1})$. Encode this sequence as follows. Write each a_j-a_{j-1} in binary; then replace simultaneously

0 by 00 1 by 01

, by 10 (and) by 11.

Call this encoded result E(S). The length of E(S) is

$$\sum_{j=1}^{k}(2\lg(a_j-a_{j-1})+O(1)) \qquad \text{bits.}$$

By Jensen's inequality,

$$\frac{1}{k}\sum_{j=1}^{k}\lg(a_j - a_{j-1}) \leqslant \lg\left(\frac{1}{k}\sum_{j=1}^{k}(a_j - a_{j-1})\right).$$

Thus the length of $E(S)$ is at most

$$\begin{aligned} 2\sum_{j=1}^{k}\lg(a_j - a_{j-1}) + O(k) &\leqslant 2k\lg\left(\frac{1}{k}\sum_{j=1}^{k}(a_j - a_{j-1})\right) + O(k) \\ &= 2k\lg(a_k/k) + O(k) \\ &\leqslant 2k\lg(L/k) + O(k). \end{aligned}$$

If every a_j were written out in binary, then S would be encoded with $k\lg L + O(k)$ bits. When k is large, E(S) has fewer bits.

This encoding permits efficient insertion of a new value into S and efficient deletion of the smallest value from S. To insert a value b such that $a_i < b < a_{i+1}$, replace the encoding of $a_{i+1} - a_i$ by the encoding of the subsequence $b - a_i, a_{i+1} - b$. To delete the smallest value a_1, replace the encoding of the subsequence $a_1, a_2 - a_1$ at the beginning of E(S) by the encoding of a_2.

The following bound is used in Subsection 4.2.

LEMMA 7. *If $k \leqslant N$, then $k\lg(L/k) \leqslant N\lg(L/N) + O(N)$.*

Proof. Let $e = 2.71828...$. The function f defined by

$$f(k) = k\lg(L/k)$$

increases monotonically for $k < L/e$ and attains its maximum value at $k = L/e$. Thus if $N < L/e$, then

$$f(k) \leqslant f(N) = N\lg(L/N),$$

and if $N \geqslant L/e$, then

$$f(k) \leqslant f(L/e) = (L/e)\lg e = O(N).$$

The lemma follows. ∎

4.2. *The Unidirectional Ring*

Consider a unidirectional ring of N processors with distinct initial values in $\{0,..., L\}$. By the hypothesis of Subsection 2.1, each initial value is representable with $O(\lg N)$ bits, hence

$$\lceil \lg L \rceil \leqslant \alpha \lg N$$

for some constant α. This section presents an algorithm that sorts these values by successive insertions with $O(N^2 \lg (L/N)/\lg N)$ messages, each with at most $\alpha \lg N$ bits. This algorithm achieves the optimal bit complexity of Theorem 1 and the optimal message complexity of Theorem 2.

The algorithm employs the encoding E defined in Subsection 4.1. Let $1 \leqslant k \leqslant N$, and let $S \subseteq \{0,...,L\}$ have k values. The encoding $\mathrm{E}(S)$ is transmitted as a sequence of messages. By Lemma 7, the number of messages used to transmit $\mathrm{E}(S)$ is

$$\left\lceil \frac{2k \lg(L/k) + O(k)}{\alpha \lg N} \right\rceil \leqslant \frac{2N \lg(L/N) + O(N)}{\alpha \lg N}.$$

The algorithm comprises three phases. During the first phase, the processors use the algorithm of Peterson (1982) to elect a leader with $O(N \lg N)$ messages. Since each message represents an initial value, $\alpha \lg N$ bits suffice for each message. Without loss of generality, assume that processor 0 is the leader.

To initiate the second phase, the leader sends $\mathrm{E}\{IV(0)\}$ to processor 1. For $p = 0,..., N-1$, define $S(p) = \{IV(0),..., IV(p-1)\}$. In general, during the second phase, processor p receives $\mathrm{E}(S(p))$ from processor $p-1$ and sends $\mathrm{E}(S(p+1))$ to processor $p+1$. Since $\mathrm{E}(S(p))$ is an encoding of a sorted set, processor p need not store all of $\mathrm{E}(S(p))$. Rather, processor p inserts $IV(p)$ into $S(p)$ at the appropriate point, as described at the end of Subsection 4.1. At the end of the second phase the leader begins to receive $\mathrm{E}(S(N))$.

During the third phase, the processors successively remove the smallest value from $S(N)$. Define $T(0) = S(N)$. For $p = 0,..., N-2$, processor p receives $\mathrm{E}(T(p))$ from processor $p-1$. It defines $FV(p)$ to be the smallest value in $T(p)$. Let $T(p+1) = T(p) - FV(p)$. Processor p sends $\mathrm{E}(T(p+1))$ to processor $p+1$. Subsection 4.1 shows that the encoding E supports efficient deletion of the smallest value in the set. Processor $N-1$ receives the largest value.

THEOREM 5. *On a unidirectional ring of N processors with initial values in $\{0,..., L\}$, the sorting problem can be solved with $O(N^2 \lg(L/N)/\lg N)$ messages, each of $O(\lg N)$ bits.*

Proof. The first phase involves $O(N \lg N)$ messages, each of $\alpha \lg N$ bits. Every processor transmits an encoding of a set during the second phase and another encoding during the third phase. Therefore the algorithm uses at most

$$2N \frac{2N \lg(L/N) + O(N)}{\alpha \lg N} = O(N^2 \lg(L/N)/\lg N)$$

messages after it elects a leader. ∎

4.3. *The Discrete Torus*

On the discrete torus, the algorithm of Nassimi and Sahni (1979), when implemented asynchronously, uses $O(N^{3/2})$ messages because each of the N processors sends $O(N^{1/2})$ messages. Each message is an initial value. By Theorem 3, when the initial values are in $\{0,..., N^{\alpha}\}$ for some constant $\alpha > 1$, this algorithm is optimal within a constant multiplicative factor. The construction of an optimal algorithm for small L remains an open problem.

RECEIVED: September 15, 1983; ACCEPTED: February 27, 1984

REFERENCES

ABELSON, H. (1980), Lower bounds on information transfer in distributed systems, *J. Assoc. Comput. Mach.* **27**, 384–392.

BURNS, J. E. (1980), "A Formal Model for Message Passing Systems," Tech. Rep. No. 91, Comput. Sci. Dept., Indiana Univ. at Bloomington, May 1980.

CHANDY, K. M., AND MISRA, J. (1982), Distributed computation on graphs: Shortest path algorithms, Comm. *ACM* **25**, 833–837.

DOLEV, D., KLAWE, M., AND RODEH, M. (1982), An $O(n \log n)$ unidirectional distributed algorithm for extrema finding in circles, *J. Algorithms* **3**, 245–260.

FREDERICKSON, G. N. (1983), Tradeoffs for selection in distributed networks, *in* "Proc. 2nd ACM Sympos. on Principles of Distributed Computing," pp. 154–160, Assoc. Comput. Mach., New York.

GALLAGER, R. G., HUMBLET, P. A., AND SPIRA, P. M. (1983), A distributed algorithm for minimum-weight spanning trees, *ACM Trans. Prog. Lang. Systems* **5**, 66–77.

HOPCROFT, J. E., AND ULLMAN, J. D. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, Mass.

JA' JA', J., AND PRASANNA KUMAR, V. K. (1984), Information transfer in distributed computing with applications to VLSI, *J. Assoc. Comput. Mach.* **31**, 150–162.

KORACH, E., ROTEM, D., AND SANTORO, N. (1982), Distributed algorithms for ranking the nodes of a network, *in* "Proc. 13th Southeast Conf. on Combinatorics, Graph Theory, and Computing," pp. 235–246.

KORACH, E., MORAN, S., AND ZAKS, S. (1984), Tight lower and upper bounds for some distributed algorithms for a complete network of processors, *in* "Proc. 3rd Ann. ACM Sympos. on Principles of Distributed Computing," Assoc. Comput. Mach., New York, in press.

LINT, B., AND AGERWALA, T. (1981), Communication issues in the design and analysis of parallel algorithms, *IEEE Trans. Software Engrg.* **SE-7**, 174–188.

LIPTON, R. J., AND SEDGEWICK, R. (1981), Lower bounds for VLSI, *in* "Proc. 13th Ann. ACM Sympos. on Theory of Computing," pp. 300–307, Assoc. Comput. Mach., New York.

MATSUSHITA, T. A. (1983), "Distributed Algorithms for Selection," Tech. Rep. No. T-127 (ACT-37), Coordinated Science Lab., Univ. Illinois at Urbana–Champaign, July 1983.

NASSIMI, D., AND SAHNI, S. K. (1979), Bitonic sort on a mesh-connected parallel computer, *IEEE Trans. Comput.* **C-28**, 2–7.

PAPADIMITRIOU, C. H., AND SIPSER, M. (1984), Communication complexity, *J. Comput. System Sci.* **28**, 260–269.

PETERSON, G. L. (1982), An $O(n \log n)$ unidirectional algorithm for the circular extrema problem, *ACM Trans. Prog. Lang. Systems* **4**, 758–762.

RODEH, M. (1982), Finding the median distributively, *J. Comput. System Sci.* **24**, 162–166.

ROTEM, D., SANTORO, N., AND SIDNEY, J. B. (1983), "Distributed Sorting," Tech. Rep. No. SCS-TR-34, School of Comput. Sci., Carleton Univ., Dec. 1983.

SANTORO, N. (1981), Distributed algorithms for very large distributed environments: New results and research directions, *in* "Proc. Canad. Inform. Processing Soc.," 1.4.1–1.4.5, Waterloo, Ontario.

SANTORO, N. (1982), "On the Message Complexity of Distributed Problems," Tech. Rep. No. SCS-TR-13, School of Computer Science, Carleton Univ., Dec. 1982.

SANTORO, N., AND SIDNEY, J. B. (1982), Order statistics on distributed sets, *in* "Proc. 20th Ann. Allerton Conf. on Communication, Control, and Computing," pp. 251–256, Univ. Illinois at Urbana–Champaign.

SEGALL, A. (1982), Decentralized maximum-flow protocols, *Networks* **12**, 213–220.

THOMPSON, C. D. (1979), Area–time complexity for VLSI, *in* "Proc. 11th Ann. ACM Sympos. on Theory of Computing," pp. 81–88, Association for Computing Machinery, New York.

YAO, A. C. C. (1979), Some complexity questions related to distributive computing, *in* "Proc. 11th Ann. ACM Sympos. on Theory of Computing," pp. 209–213, Association for Computing Machinery, New York.