THE COMPLEXITY OF PARALLEL SORTING

Friedhelm Meyer auf der Heide

Avi Wigderson

IBM Research Laboratory, San Jose[1]

Abstract :
We consider PRAM's with arbitrary computational power for individual processors, infinitely large shared memory and "priority" write-conflict resolution.

The main result is that sorting n integers with n processors requires $\Omega(\sqrt{\log n})$ steps in this strong model.

We also show that computing any symmetric polynomial (e.g. the sum or product) of n integers requires exactly $\log_2 n$ steps, for any finite number of processors.

I. Introduction

In the last few years we have witnessed a multitude of upper bounds for parallel sorting on various models. These results culminated in the remarkable $O(n \log n)$ sorting network of Ajtai, Komlos and Szemeredi [*AKS*], and its beautiful adaptation to bounded degree n-node networks by Leighton [*L*].

Surprisingly, there are essentially no lower bound results for sorting. The reason seems to be that for many parallel models such bounds follow either from simple considerations or from parallel lower bounds on simpler functions. Here are a few examples.

1. Comparison trees and algebraic computation trees. The $\Omega(n \log n)$ sequential lower bound for sorting (Paul and Simon [*PS*]) implies an $\Omega(\log n)$ lower bound on the parallel versions of these trees with n processors.

2. Bounded degree n-node networks. An $\Omega(\log n)$ lower bound follows from diameter considerations.

3. Polynomial size, unbounded fan-in circuits. The $\Omega(\sqrt{\log n})$ on depth for computing the parity function, due to Ajtai from [*A*], yields a similar bound for sorting.

4. Exclusive-write PRAM. The $\Omega(\log n)$ lower bound for computing the Or function, due to Cook, Dwork and Reischuk from [*CDR*] implies a similar bound for sorting.

In all the parallel models mentioned above, there is a restriction on either the computational power of the individual processors, or on the nature of communication between them. Here we pose no such restrictions - we consider a concurrent-write PRAM, in which n processors of arbitrary computational power communicate via an infinite shared memory. Moreover, we use the strongest convention of resolving write conflicts -the processor with the largest index succeeds.

The only non-trivial lower bound on this model appears in [*FMRW*]. It is a (tight) $\Omega(\log \log n)$ bound on finding the maximum of n integers. As is almost traditional with lower bounds for models in which the computation and communication behavior depend on the input in a complicated way, Ramsey theory is used to "clear the smoke" and find some structure. Such an argument showed that, for finding the maximum, a

1 The first author is now at Johann Wolfgang Goethe Universitat, Fb Informatik, 6000 Frankfurt a.M., West-Germany. The second author is now at the Mathematical Sciences Research Institute, 1000 Centenial Dr., Berkeley, CA 94720.

parallel comparison tree is essentially (i.e. for a large input set) as good as a PRAM. Then Valiant's $\Omega(\log\log n)$ lower bound from [*V*] on such trees is applied.

The problem with the Ramsey theoretic argument in [*FMRW*] is that input variables are fixed to constants at a doubly exponential rate as the computation proceeds, and hence it can never yield a lower bound bigger than $\Omega(\log\log n)$.

We prove an $\Omega(\sqrt{\log n})$ on sorting n integers. We use a new Ramsey theoretic argument in which variables are not fixed, but instead some information about their ordering is given to the algorithm as the computation proceeds. With this argument we show that essentially all a processor can do in a step is to merge two ordered sets (chains) of variables. It is natural now to define a parallel (m,n)-merge tree, in which at every node n pairs of chains of size m each are merged. (Note that for $m = 1$ this tree is simply Valiant's parallel comparison tree.)

We prove a general lower bound, parameterized by m, for such trees. Then we combine it with the dependency of m on the time of the PRAM computation to obtain our $\Omega(\sqrt{\log n})$ lower bound. We believe that our lower bound on merge trees, and hence the lower bound on PRAM's can be improved to $\Omega(\log n)$.

As mentioned, our Ramsey theoretic argument gives information about the ordering of input variables to the algorithm. For sorting, we had to take care to give as little information of this type as possible. For symmetric functions, however, applying this argument becomes much simpler, since we can assume that the input variables are sorted to begin with. Using this idea, combined with known techniques to making the "degree" of the computation finite (albeit the infinite address space), we get a simple proof of the following result. The computation of any symmetric polynomial of n integers (e.g. the sum or the product) requires exactly $\log_2 n$ steps.

This result was independently obtained (at least for addition) by Israeli and Moran in [*IM*]. It unifies and generalizes similar bounds on models that restrict either the computational power of processors (Meyer auf der Heide and Reischuk [*MR*]) or the interprocessor communication (Parberry [*P*]).

II. Formal Definitions and Statements of Results

The two important models in this paper are the PRAM and the parallel merge tree. We start with the (standard) definition of the PRAM.

A (PRIORITY) PRAM M consists of n processors (RAM's) $P_1,P_2,\ldots,P_n$, and infinitely many shared memory cells (which we abbreviate 'cells'), indexed by the integers N. We say that M computes a function $f{:}N^n \rightarrow N^m$ in T steps, if initially P_i has the value a_i of the ith input variable in its local memory, all cells are initialized to '0', and after T steps the first m cells contain the m values of $f(a_1,a_2,\ldots,a_n)$.

Every step of the computation consists of two phases, synchronously performed by all processors. In the write phase, each processor writes some value into some cell. In the read phase, each processor reads some cell to its local memory. (These addresses and this write value depend on the processor's local memory, but we place no restriction on the complexity of computing them.) If a cell is written into by several processors at one step, its contents will be the value written by the highest index processor among them.

For the definition of the merge tree, we need some notation for partial orders.

Let V be a set of elements $\{x_1,x_2,\ldots,x_n\}$. A directed acyclic graph $Q = (V,E)$ defines a partial order $<_Q$ in the natural way. $\overline{Q} = (V,\overline{E})$ is the *transitive closure* of Q. A set $U \subset V$ is a *chain* (or a *total order*) in Q if the elements of U belong to a directed path in Q. A set U is an *antichain* in Q if U is an independent set in $\bar{Q}$.

Let $U_1, U_2 \subset V$ and $Q_1 = (U_1, E_1)$, $Q_2 = (U_2, E_2)$ be partial orders. Q_1 and Q_2 are *consistent* if there is no pair $a,b \in V$ such that $(a,b) \in \bar{E}_1$ and $(b,a) \in \bar{E}_2$. A family of partial orders on subsets of V, $Q_1, Q_2, \ldots, Q_k$ is *consistent* if every pair is consistent. The *union* of two consistent orders Q_1 and Q_2 is $Q_1 \cup Q_2 = (V, E_1 \cup E_2)$. Similarly we define $\bigcup_{i=1}^{k} Q_i$.

We say that $Q' = (V,E')$ is an *extension* of $Q = (V,E)$ iff $\bar{E} \subset \bar{E}'$. If Q' is a total order, we say that it is a *linear extension* of Q.

Let $C_1, C_2 \subset V$ be two chains in $Q = (V,E)$. A *merge* of C_1 and C_2 in Q is any total order $Q' = (C_1 \cup C_2, E')$, consistent with Q.

Now we are ready to define the model. An (m,n) *– merge tree* (for sorting) is a rooted tree with labels on the nodes and on the edges. The label of an internal node v is a partial order Q on V, together with n pairs of chains in Q, (C_{i1}, C_{i2}), $i = 1,2,\ldots,n$, satisfying $|C_{ij}| \le m$ for all $i = 1,2,\ldots,n$, $j = 1,2$. Every branch e out of v is labeled with a different consistent family of merges C_i of C_{i1} and C_{i2}, $i = 1,2,\ldots,n$. If u is the child of v connected by e, then the partial order in u is $Q \cup \bigcup_{i=1}^{n} C_i$. The root is labeled with the empty partial order on V.

The cost of an (m,n)-merge tree H, denoted $c(H)$ is the length of a longest root-leaf path in H. H sorts n numbers, if each leaf is labeled with a total order. The cost of sorting on (m,n)-merge trees is $c(m,n)$, the minimum cost $c(H)$ of an (m,n)-merge tree H which sorts n numbers.

The main result of this paper is the following.

Main Theorem : Any PRAM needs $\Omega(\sqrt{\log n})$ steps to sort n integers.

The main theorem follows directly from the theorems 1 and 2 below.

Theorem 1 : Let M be a PRAM sorting n integers in T steps. Then there is a $(2^T, n)$-merge tree which sorts n numbers in T steps.

Theorem 2 : $c(m,n) = \Omega(\frac{\log n}{\log(m \log n)})$.

As a corollary to the proof method of theorem 1 we get a general tight bound for a family of symmetric functions. For $S \subset N$ let $S_*^n = \{(a_1,\ldots,a_n) \in S^n \mid a_i \neq a_j \text{ for } i \neq j\}$. A function $f: N^n \to N$ is *strongly non-constant*, if for each infinite $S \subset N$, f restricted to S_*^n is non-constant.

Theorem 3 : A PRAM needs exactly $\log_2 n$ steps to compute a symmetric, strongly non-constant function.

Example : Every non-constant, symmetric polynomial on n variables (e.g. the sum or the product) is a symmetric, strongly non-constant function.

III. Simulating PRAM's by Merge Trees

Theorem 1 : Let M be a PRAM sorting n integers in T steps. Then there is a $(2^T, n)$-merge tree which sorts n elements in T steps.

We simulate M by a computation tree. We want to maintain at each node v of this tree in depth t the following property :

Let I be the set of inputs arriving at v. Then, for inputs from I, each processor P_i only knows variables with indices from some set $X_i \subset [n]$ at time t. ($[n] = \{1,2,\ldots,n\}$)

The exact meaning of this is that for inputs from I, the configuration of P_i after t steps (considered as a function in I) only depends on the variables with indices from X_i. In the sequel we shall refer to them as the variables from X_i for short.

Clearly, because of its arbitrary computational power, P_i can now sort the variables it knows in one step and can proceed dependent on their order. We shall see that this is essentially the only property of the input which can influence the behavior of P_i, if we restrict the input set suitably. Namely, if the orderings of the sets of variables the processors know are fixed, the next "communication pattern" is fixed, too, in the following sense.

For each $i\epsilon[n]$, P_i reads the value that was written by some fixed processor P_{j_i} at some fixed time t_i.

Thus it is fixed which new variables P_i gets to know in this step, namely just those which P_{j_i} knew before. As they also were already sorted before, the behavior of P_i in the next step only depends on the outcome of the merging of the two ordered sets of variables it now knows. Thus M behaves like a merge tree.

In the sequel we only consider inputs which consist of distinct numbers. Recall that for a set $S \subset N$, $S^n_* = \{(a_1,...,a_n)\epsilon S^n \mid a_i \neq a_j \text{ for } i \neq j\}$. For $i\epsilon[n]$ let $X_i \subset [n]$ and π_i be a total order on X_i. Let $X = (X_1,..,X_n)$, $\pi = \bigcup_{i=1}^{n} \pi_i$. We always assume that the π_i's are consistent. Then, following the above intuition, the set of inputs arriving at a node of the merge tree should be of the form $I(X,\pi,S) = \{(a_1,...,a_n)\epsilon S^n_* \mid a_p < a_q \text{ for all } p,q \text{ such that } p <_\pi q\}$.

The following lemma is the heart of the proof of the theorem.

Main Lemma I: Suppose that at time t, for inputs from $I = I(X,\pi,S)$, P_i only knows variables from X_i. Then there are $j_1,...j_n\epsilon[n]$ and an infinite $S' \subset S$, such that after step t, for inputs from $I' = I(X,\pi,S')$, P_i only knows variables from $X_i \cup X_{j_i}$.

Before we prove this lemma we first conclude theorem 1 from it. For this purpose we define inductively a $(2^t,n)$-merge tree H of depth t. We shall show that finally, when H has depth T, it sorts n numbers. For the inductive construction to work, we keep extra information at the nodes.

The set of inputs at the root is $N^n_* = I((\{1\},...,\{n\}),(\Phi,..,\Phi),N)$, where Φ denotes the trivial order on one element. At this time P_i only knows x_i. So the assumption in the lemma holds for t=0.

Now let $t\epsilon[T]$ and assume that we have constructed a $(2^t,n)$-merge tree of depth t for inputs from S^n_*, such that the set of inputs arriving at a node v in depth t is of the form $I = I(X,\pi,S)$, and for inputs from I, P_i only knows variables from X_i at time t. Furthermore we maintain that although X and π depend on v, S is the same set for all nodes in depth t.

For each node v in depth t successively perform the construction below.

Let $I = I(X,\pi,S)$ be the set of inputs arriving at v, and S', $j_1,...j_n$ be as in the main lemma I. Now we define a son v' of v for each consistent tuple of mergings $\pi' = \bigcup_{i=1}^{n} \pi'_i$ of the sets in $X' = (X'_1,...,X'_n)$ with $X'_i = X_i \cup X_{j_i}$. Then the set of inputs arriving at v' is $I' = (X',\pi',S')$. As $I' \subset I(X,\pi,S')$, we know by main lemma I that, for inputs from I', P_i only knows variables from X'_i at time t+1. Now for the already defined sets of inputs arriving at nodes replace S by S'. Finally, after having performed this construction for all nodes in depth t, we get a $(2^{t+1},n)$-merge tree of depth t+1.

Now suppose that we have constructed H up to depth T. We finally have to verify that H sorts n numbers, i.e. to prove that for each set of inputs $I = I(X,\pi,S)$ arriving at a leaf of H the order on $[n]$ induced by π is total. Suppose it is not. Then, as for inputs from I each P_i only knows variables from X_i at time T, M can not distinguish between the different possible total orders and would compute the wrong output for some inputs. Thus H is a merge tree as demanded in the theorem.

Before we prove main lemma I, we introduce some notation and useful Ramsey theoretic results.

For an infinite set $S \subset N$ and a total order Q on $[n]$ let $S^n_Q = \{(a_1,\ldots,a_n)\varepsilon S^n \mid a_i < a_j \text{ if } i <_Q j\}$. Such a set is called to be of fixed order type. If Q is the natural order on $[n]$, we write $S^n_<$ for S^n_Q.

We apply the following "canonical" Ramsey theorem due to Erdos and Rado from [ER] (see also [GRS]).

Theorem [ER] (Erdos-Rado Theorem) : Let $f:S^n_Q \to N$. Then there is $\tilde{S} \subset S$, $\tilde{S}$ infinite, such that $f' = f|_{\tilde{S}^n_Q}$ is 1-1 on the variables it depends on. Precisely, there is a $J \subset [n]$ such that $f'(a_1,\ldots,a_n) \neq f'(b_1,\ldots,b_n)$ if and only if $a_i \neq b_i$ for some $i\varepsilon J$. In particular, if f has a finite range, f' is constant.

Lemma 1 : Let $f:S^n_< \to N$ and $g:S^{n'}_< \to N$ be 1-1 functions. Then there is $\tilde{S} \subset S$, $\tilde{S}$ infinite, such that, restricted to $\tilde{S}^n_<$, f and g either have disjoint ranges, or they are identical. Precisely, either $f(\tilde{S}^n_<) \cap g(\tilde{S}^{n'}_<) = \Phi$ or $n = n'$ and $f|_{\tilde{S}^n_<} \equiv g|_{\tilde{S}^n_<}$.

Proof: Assume w.l.o.g. that $n \geq n'$. Add dummy variables such that also g is defined on $S^n_<$, but only depends on the first n' variables. We first consider the 2-coloring c on $S^n_<$ with $c(\bar{a}) = 1$ if $f(\bar{a}) = g(\bar{a})$, and $c(\bar{a}) = 0$ otherwise. By the Erdos-Rado Theorem there is $\bar{S} \subset S$, $\bar{S}$ infinite, such that c is constant on $\bar{S}^n_<$. If $c \equiv 1$, then $f|_{\bar{S}^n_<} \equiv g|_{\bar{S}^n_<}$ and we are done. If $c \equiv 0$, then let G be the directed graph on $\bar{S}^n_<$ with $(\bar{a},\bar{b})\varepsilon E(G)$ if $f(\bar{a}) = g(\bar{b})$. G has no self-loops because $c \equiv 0$ on $\bar{S}^n_<$. G has indegree 1, because f is 1-1. Therefore it is easy to see that the underlying undirected graph is 3-colorable. Color it with 3 colors. By the Erdos-Rado Theorem there is an infinite $\tilde{S} \subset \bar{S}$ such that $\tilde{S}^n_<$ is monocromatic. Thus, $f(\tilde{S}^n_<) \cap g(\tilde{S}^n_<) = \Phi$. q.e.d.

Proof of main lemma I : Let $I = I(X,\pi,S)$ be such that P_i only knows variables from X_i at time t. Consider what P_i has done until time t. At each time $d \varepsilon [t]$, it wrote some value v^d_i to some cell w^d_i. Furthermore P_i read cell r_i at time t. These values are functions of the input set N^n. But, as P_i only knows variables from X_i, they only depend on the variables from X_i. Our goal now is to restrict the input set I to a set I' in such a way that, for inputs from I', P_i reads what some P_j wrote at some time d, only if r_i and w^d_j are the "same functions", and are applied to the "same arguments".

We refer to the functions w^d_i and r_i as address functions. The *clean form* of such a function f is derived by fixing all variables f does not depend on to arbitrary constants. Thus a clean form of a function always depends on all its variables. Let f' be the clean form of some address function f. If f was used by P_i (i.e. is r_i or w^d_i for some d), then it only depends on some (not necessary all) variables from X_i. As they are totally ordered according to π_i, f' is defined on a set of fixed order type. Thus we may apply the Erdos-Rado Theorem successively to the clean forms of all address functions. The result is an infinite set $\tilde{S} \subset S$ such that, on

$\tilde{I} = I \cap \tilde{S}^n$, all address functions are 1-1 on the variables they depend on.

From now on we assume that the domain of all address functions is $\tilde{I}$. Note that the clean form of the address functions can now depend on fewer variables than before we applied the Erdos-Rado Theorem.

We know that the clean form of an address function is defined on a set of fixed order type. The function derived from it by reordering these variables such that it is defined on $\tilde{S}^n_<$ is called the *standard form* of f.

As we now know that the standard forms of address functions are 1-1 functions and are defined on $S^n_<$, we may apply lemma 1 successively to all pairs of them. As a result we get an infinite set $\bar{S} \subset \tilde{S}$ with the following property (*). Assume from now on that the domain of all address functions is $I' = \tilde{I} \cap \bar{S}^n$.

(*) *Two address functions either have disjoint ranges or have the same standard form.*

Now let f and g be two address functions. Recall that they are defined on I'.

Lemma 2 : f and g are either identical or $f(\bar{a}) \neq g(\bar{a})$ for all $\bar{a} \in I'$.

*Proof :*Suppose $f(\bar{a}) = g(\bar{a})$ for some $\bar{a} \in I'$. Then, by (*), they have the same standard form h. Thus they are identical if they depend on the same variables. Assume that f and g depend on different sets of variables. Let $\bar{a}'$ and $\bar{a}''$ be the subvectors of the above $\bar{a}$ of those variables f and g depend on, increasingly ordered. These vectors are different, because they contain values of different variables, and the a_i's are distinct. On the other hand, we know that $h(\bar{a}') = f(\bar{a})$, $h(\bar{a}'') = g(\bar{a})$. But as h is 1-1, $h(\bar{a}') \neq h(\bar{a}'')$, which contradicts the supposition we started with. q.e.d.

Now consider what P_i reads in step t. If r_i was never used for writing (i.e. $r_i \neq w_j^d$ for all j and d), then we know by lemma 2 that P_i reads 0. Otherwise, because of the PRIORITY write conflict resolution and lemma 2, P_i reads v_j^d, the value P_j wrote at time d, where (d,j) is lexicographically maximal such that $w_j^d \equiv r_i$. Thus, as v_j^d only depends on variables from X_j, after this step P_i only knows variables from $X_i \cup X_j$ for inputs from I', which proves the lemma. q.e.d.

IV. A Lower Bound for Sorting on Merge Trees

In this section we complete the proof of our main theorem by showing the demanded lower bound on (m,n)-merge trees.

Theorem 2 : $c(m,n) = \Omega(\frac{\log n}{\log(m \log n)})$.

Proof of theorem 2: It is sufficient to prove that $c(m,n) = \Omega(\frac{\log n}{\log m})$ for $m \geq 9 \log n$. Let H be any (m,n)-merge tree which sorts n numbers. We shall inductively exhibit a long path from the root in H. Intuitively, the argument is as follows. Suppose that we constructed the last node in the path, v, s.t. the partial order in v has many linear extensions. We shall look for a child u of v with the same property. To do that, we must show that there is a way of merging the n pairs of chains given at v, s.t. as few as possible transitive implications are added to the partial order.

To this end, we shall define a "nice" class of partial orders, in which we can get a handle on the quantities mentioned above. We will show that for each node in the path we construct, its partial order has an extension which is a partial order in the nice class.

We now describe the "nice" partial orders. Let $n = kl + a$ for positive integers k,l,a. A $(k,l,a)-graph$ is obtained as follows. Take a chain on $k + a$ vertices. Then choose some k of the vertices, and replace each by l copies of itself (each having the same in and out neighbors).

A *(k,l,a)-partial order* is a partial order Q s.t. $\overline{Q}$ is isomorphic to the transitive closure $\overline{G}$ of some (k,l,a) graph G.

Let $Q = (V,E)$ be a (k,l,a)-partial order. It has k antichains, each of size l. Call them $S_1,S_2,...,S_k$. Clearly, to "sort" Q, it is necessary and sufficient to sort all the $S_i's$. Consider a merge operation on Q. It involves two chains, C_1 and C_2 $(|C_1|, |C_2| \le m)$. Clearly, we may assume that $C_1,C_2 \subset \bigcup_{i=1}^{k} S_i$ (as the rank of the elements outside those antichains is known). Moreover, by the structure of Q, $|C_j \cap S_i| \le 1$ for all $j = 1,2$ and $i = 1,2,...,k$. Hence, merging C_1 and C_2 reduces to performing at most m comparisons, at most one in each S_i.

Now we are ready for the main lemma of this section.

Main Lemma II : Let v be a node in H whose partial order Q_v is contained in a (k,l,a)-partial order, with $a \le \frac{n}{2}$ (hence $kl \ge \frac{n}{2}$). Then there is a child u of v whose partial order Q_u is contained in an (k',l',a') partial order, with $k' \le km^4$ and $a' \le a + \frac{3n}{m} + km^4$.

Before we prove main lemma II, let us see how it implies theorem 2. Clearly, the empty partial order at the root of H is a $(1,n,0)$ partial order. Apply the lemma t times with $t = \frac{\log n}{5 \log m}$. We reach a node whose partial order is contained in a (k,l,a)-partial order with $k \le m^{4t}$ and $a \le \frac{3tn}{m} + m^{4t}$. (Note that since $m \ge 9 \log n$, and because of the choice of t, the assumption in the lemma holds at every step). But for this choice of t, $k \le \frac{n}{4}$, $a \le \frac{n}{2}$, and therefore $l \ge 2$. So this partial order cannot be a total order, and this node at depth t cannot be a leaf. Hence, $c(H) = \Omega(\frac{\log n}{\log m})$.

Before we proceed to prove lemma 2, we need the following technical lemma.

Lemma 3 : Let $G = (V,E)$ be an *undirected* graph, and b a positive number. Then one can remove a set $V' \subset V$ of vertices, with $|V'| \le \frac{2|V|}{b} + \frac{2|E|b}{|V|}$, s.t. the remaining graph on $V - V'$ can be colored with $\frac{2|E|b}{|V|}$ colors, with each color class of the same size.

This lemma is based on the following result, due to Hanjal & Szemeredi from [*HS*].

***Theorem* [*HS*] :** Any graph $G = (V,E)$ with maximum degree Δ can be colored with $\Delta + 1$ colors s.t. each color class has size $\frac{|V|}{\Delta + 1}$ or $\frac{|V|}{\Delta + 1} + 1$.

Proof of lemma 3: Remove the $\frac{2|V|}{b}$ vertices of highest degree from G. Then the maximum degree is at most $\frac{2|E|b}{|V|} - 1$. Apply the previous theorem, and then remove one vertex from each of the larger color classes (at most $\frac{2|E|b}{|V|}$), to make them all of equal size.

Proof of main lemma II : Assume w.l.o.g that Q_v is a (k,l,a)-partial order, and let $S_1,S_2,...,S_k$ be its antichains of size l. Consider the n pairs of chains, (C_{i_1},C_{i_2}), $i = 1,2,...,n$, that are merged at v. Each such pair gives rise to at most m comparisons. So we have a total of nm comparisons, distributed among the $S_j's$. By averaging, at most $\frac{k}{m}$ of the $S_j's$ have more than $\frac{nm^2}{k}$ comparisons. For each such S_j, arbitrarily choose a total order, (hence resolving all comparisons within it). We obtain an extension Q_1 of Q_v which is a (k_1,l,a_1)-partial order, with $k_1 \leq k$ and $a_1 \leq a + \frac{k}{m} l \leq a + \frac{n}{m}$.

So each of the remaining $S_j's$ has at most $\frac{nm^2}{k}$ comparisons. Think of these comparisons as edges in an undirected graph with vertex set S_j. Now we use lemma 3 on each of those graphs. From each S_j remove $\leq \frac{2l}{m} + \frac{2\frac{nm^2}{k}m}{l}$ vertices as in the lemma, make them all (say) bigger than the rest of S_j, and arbitrarily totally order them. This resolves all comparisons involving these vertices. The result is an extension Q_2 of Q_1, which is a (k_2,l,a_2) partial order, with $k_2 = k_1 \leq k$ and

$$a_2 \leq a_1 + k(\frac{2l}{m} + \frac{2nm^3}{kl})$$
$$\leq a_1 + \frac{2n}{m} + 4km^3 \leq a + \frac{3n}{m} + km^4.$$

Finally, the remaining graph on each S_j can be colored with $\leq \frac{2nm^3}{kl} \leq 4m^3 \leq m^4$, each color class of equal size. Totally order the *color classes* arbitrarily, thus resolving the remaining comparisons in each S_j. The result is a (k',l',a') partial order Q', with $k' \leq km^4$ and $a' \leq a + \frac{3n}{m} + km^4$. Since we resolved all comparisons in a consistent way, there must be a child u of v s.t. Q' is an extension of Q_u.

V. A general lower bound for a family of symmetric functions

In this section we prove theorem 3. Recall that for $S \subset N$ we defined $S^n_{\ne} = \{(a_1,...,a_n) \in S^n \mid a_i \neq a_j \text{ for } i \neq j\}$. A function $f: N^n \to N$ is *strongly non-constant*, if for each infinite $S \subset N$, f restricted to $S^n_{\ne}$ is non-constant.

Theorem 3 : A PRAM M with any finite number p of processors needs exactly $\log_2 n$ steps to compute a symmetric, strongly non-constant function.

Proof : The upper bound is obvious because of the computational power of the processors : in $\log_2 n$ steps one processor can get to know all variables and then compute the function in one step.

In order to show the lower bound, we shall again apply the Erdos-Rado Theorem to find an infinite $S \subset N$ such that M is oblivious to inputs from $S^n_<$.

We first describe what we mean by oblivious. Let $\bar{x} \in N^n$ be an input for M. Then it is well defined which processor writes to or reads from which memory cell at a given time, if M is started with $\bar{x}$. Thus we can define the following *communication pattern for* $\bar{x}$:

For each $t \in [T]$, $i \in [p]$, P_i reads at time t what P_j has written at time d, where $d \in [t]$ and $j \in [p]$ depend only on i and t.

M is oblivious to inputs from $I \subset N^n$, if the communication patterns of all inputs from I are the same.

The following lemma is often proved in similar forms in literature, e.g. in [*MR,CDR*].

Lemma 4 :Let $I \subset N^n$ such that $f: I \to N$ depends on all its variables. If the PRAM M computing f is oblivious to inputs from I, then M needs at least $\log_2 n$ steps to compute f.

Lemma 5 : If M computes $f: N^n \to N$ then there is an infinite $S \subset N$, such that M is oblivious to inputs from $S^n_<$.

The lower bound in theorem 3 now follows directly from the above lemmas and the definition of strongly non-constant functions. As we only deal with symmetric functions, if such a function is non-constant on S_*^n then it is so even on $S_<^n$.

Proof of lemma 5 : We restrict f to inputs from $N_<^n$. As the number of processors and the number of steps M executes are finite, there are only finitely many different communication patterns. Thus, by the Erdos-Rado Theorem, we find an infinite $S \subset N$ such that all inputs from $S_<^n$ have the same communication pattern, i.e. M is oblivious for inputs from $S_<^n$. q.e.d.

VI. Conclusions

We do not believe that our $\Omega(\sqrt{\log n})$ lower bound for sorting on PRAM's is tight. We believe the right bound is $\Theta(\log n)$.

To prove this bound via our definition of a merge tree is hopeless for the following reason. It would require a result like $c(n^\epsilon,n) = \Omega(\log n)$ for some fixed $\epsilon > 0$. But as Mike Saks showed us, $c(n^\epsilon,n) = O(\log \log n)$!

Note, however, that our merge tree is stronger than what the simulation in theorem 1 produces. There, sets that are merged in depth t can have size at most 2^t. We conjecture that for these trees, and hence for PRAMs, an $\Omega(\log n)$ bound holds.

References

[*A*] *M. Ajtai :* Σ_1^1- Formulae on finite structures, Annals of Pure and Applied Logic 24 (1983) pp. 1-48

[*AKS*] *M. Ajtai, J. Komlos, E. Szemeredi :* Sorting in $c \log n$ parallel steps, Combinatorica 3, (1983), pp. 1-19.

[*CDR*] *S. Cook, C. Dwork, R. Reischuk :* Upper and lower time bounds for parallel random access machines without simultaneous writes, preprint, 1983.

[*ER*] *P. Erdos, R. Rado:* A Combinatorial Theorem, J. London Math Society 25 (1950), pp. 249-255.

[*FMRW*] *F.Fich, F. Meyer auf der Heide, P. Ragde, A. Wigderson :* One, two, three...infinity: Lower bounds for parallel computation, to appear: 16th ACM STOC, Providence, 1985.

[*GRS*] *R. L. Graham, B. L. Rothschild, J. H. Spencer:* Ramsey Theory, Wiley and Sons, 1980.

[*HS*] *A. Hanjal, E. Szemeredi :* Proof of a Conjecture of P. Erdos, Combinatorial Theory and its Applications 2, (1970), North Holland, pp. 601-623.

[*IM*] *A. Israeli, S. Moran :* Private Communication.

[*L*] *T. Leighton :* Tight bounds on the complexity of parallel sorting, 16th ACM STOC, Washington D. C. (1984), pp. 71-80.

[*MR*] *F. Meyer auf der Heide, R. Reischuk :* On the limits to speed up parallel machines by large hardware and unbounded communication, 25th IEEE FOCS, Miami (1984), pp. 56-64.

[*P*] *I. Parberry :* A complexity theory of parallel computation, Phil. D. Thesis, Warwick 1984.

[*PS*] *W. J. Paul, J. Simon :* Decision trees and random access machines, Symposium ueber Logik und Algorithmik, Zuerich 1980, pp. 331-339.

[*V*] *L. Valiant :* Parallelism in comparison problems, SIAM J. on Comp. 4(3) (1975), pp. 348-355