TIGHT COMPLEXITY BOUNDS FOR PARALLEL COMPARISON SORTING

Noga Alon
Department of Mathematics
Tel Aviv University
and
Bell Communications Research

Yossi Azar
Department of Computer Science
School of Mathematical Sciences
Tel Aviv University

Uzi Vishkin
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
and
Tel Aviv University

ABSTRACT

The time complexity of sorting n elements using $p \geqslant n$ processors on Valiant's parallel comparison tree model is considered. The following results are obtained.

1. We show that this time complexity is $\Theta(\log n/\log(1+p/n))$.
This complements the AKS sorting network in settling the wider problem of comparison sort of n elements by p processors, where the problem for $p \leqslant n$ was resolved. To prove the lower bound, we show that to achieve time $k \leqslant \log n$, we need $\Omega(kn^{1+1/k})$ comparisons. Häggkvist and Hell proved a similar result only for fixed k.

2. For every fixed time k, we show that: (a) $\Omega(n^{1+1/k}\log n^{1/k})$ comparisons are required, ($O(n^{1+1/k}\log n)$ are known to be sufficient in this case), and (b) there exists a randomized algorithm for comparison sort in time k with an expected number of $O(n^{1+1/k})$ comparisons. This implies that for every fixed k, any deterministic comparison sort algorithm must be asymptotically worse than this randomized algorithm. The lower bound improves on Häggkvist-Hell's lower bound.

3. We show that "approximate sorting" in time 1 requires asymptotically more than $n\log n$ processors. This settles a problem raised by M. Rabin.

1. INTRODUCTION

Apparently, there is no problem in Computer Science which received more attention than sorting. [Kn-73], for instance, found that existing computers devote approximately a quarter of their time to sorting. The advent of parallel computers stimulated intensive research of sorting with respect to various models of parallel computation. Extensive lists of references which recorded this activity are given in [Ak-85], [BHe-86] and [Th-83].

Most of the fastest serial and parallel sorting algorithms are based on binary comparisons. In these algorithms the number of comparisons is typically the primary measure of time complexity. Any lower bound on the number of comparisons required for a problem, clearly implies a time lower bound for such algorithms. In the present paper, we restrict our attention to a parallel comparison model, introduced by Valiant [Va-75], where only comparisons are counted. In measuring time complexity within this model, we do not count steps in which communication among the processors, movement of data and memory addressing are performed. We also avoid counting steps in which consequences are deduced from comparisons that were performed. Note that our lower bounds apply to all algorithms, based on comparisons, in any parallel access machine (PRAM) including PRAMs which allow simultaneous access to the same common memory location for read and write purposes. See [BHo-82] for a discussion on hierarchy of models that implies this.

In a serial decision tree model, we wish to minimize the number of comparisons. The goal of an algorithm in a parallel comparison model is to minimize the number of comparison rounds as well as the total number of comparisons performed.

Let k stand for the number of comparison rounds (time) of an algorithm in the parallel comparison model. Let $c(k, n)$ denote the *minimum total* number of comparisons required to sort any n elements in k rounds (over all possible algorithms).
The known $\Omega(n\log n)$ comparisons lower bound for sorting in a serial decision tree model implies that, for any k, $c(k, n) = \Omega(n\log n)$. This lower bound can be matched by upper bounds as follows: For $k = c\log n$, the sorting network of [AKS-83] implies $c(k, n) = O(n\log n)$, where $c > 0$ is a constant which is implied by the network. For $k > c\log n$, the result $c(k, n) = O(n\log n)$ also holds. To see this, simply simulate the AKS network by slowing it down to work in k rounds.
For $k = 1$, $c(1, n) = \frac{1}{2}(n^2-n)$. This is since any sorting algorithm which works in one round must perform all comparisons. Otherwise, suppose that a dispensed comparison is between two successive elements in the sorted order; the algorithm will clearly fail to distinguish their order. On the other hand, observe that performing all comparisons simultaneously yields a one

0272-5428/86/0000/0502$01.00 © 1986 IEEE

round algorithm in the parallel comparison model that matches exactly this lower bound.

So, it remains to consider the situation for $1 < k \leqslant c \log n$.

RESULTS

We start with the **main result** of Section 2:

Result 1. $c(k, n) > k\left(\frac{n^{1+\frac{1}{k}}}{e} - n\right)$ for any $k, n \geqslant 1$, where e is the base of the natural logarithm.

Corollaries of Result 1:

Suppose we have p processors with the interpretation that each processor can perform at most one comparison at each round. Observe that $kp \geqslant c(k, n)$ or $p \geqslant c(k, n)/k$. Therefore,

Corollary 1. Any k-round ($k \geqslant 1$) parallel algorithm for sorting n elements needs $p > \frac{n^{1+\frac{1}{k}}}{e} - n$ processors. This yields $p = \Omega(n^{1+\frac{1}{k}})$ for $k \leqslant c \log n$ where c is any constant such that $0 < c < 1$.

Corollary 2. The number of rounds required to sort n elements using $p \geqslant n$ processors is $k = \Omega\left(\frac{\log n}{\log(1 + \frac{p}{n})}\right)$.

Proof: $p > \frac{n^{1+\frac{1}{k}}}{e} - n$ implies $1 + \frac{p}{n} > \frac{n^{\frac{1}{k}}}{e}$ and therefore $k > \frac{\log n}{1 + \log(1 + \frac{p}{n})}$. Hence, for $p \geqslant n$, $k = \Omega\left(\frac{\log n}{\log(1 + \frac{p}{n})}\right)$.

Corollary 3. If $p = n \log^{\beta} n$ for $\beta > 0$ then the number of rounds required to sort n elements is $k = \Omega\left(\frac{\log n}{\beta \log\log n}\right)$. This is an immediate corollary of Corollary 2.

A parallel algorithm is said to achieve *optimal speed up* if its running time is proportional to $\frac{Seq(n)}{p}$, where $Seq(n)$ is a lower bound on the serial running time, n is the size of the problem being considered and p is the number of processors used.

Corollary 4. If the number of processors is larger than n by an order of magnitude then it is impossible to design an optimal speed up comparison sorting algorithm. More formally, suppose that the number of processors p is not $O(n)$ (i.e., $n = o(p)$) then there is no (comparison) sorting algorithm which runs in time $O(\frac{n \log n}{p})$.

Result 2. Section 2 presents also upper bounds which match the lower bounds of Result 1. Specifically, we describe an explicit parallel comparison algorithm that sorts n elements in $O\left(\frac{\log n}{\log(1 + \frac{p}{n})}\right)$ rounds using $p \geqslant n$ processors. By *explicit* we mean that we actually describe such an algorithm, and not merely prove its existence using counting arguments.

To understand better the significance of these lower and upper bounds (results 1 and 2) we will use one more equivalent formulation of the results.

Corollary 5. Suppose we are given $p \geqslant n$ processors to sort n elements. The total number of comparisons performed by the fastest parallel sorting algorithm is

$$\Theta\left(\frac{p/n}{\log(1 + p/n)} n \log n\right).$$

The factor $n \log n$ represents the serial lower and upper bounds for sorting using comparisons. The other factor represents the deviation from optimal speed up.

All the remaining results, appearing in Section 3, apply to a fixed number of rounds k. Our **main result** in this part is that for every fixed k, there is an explicit randomized algorithm for sorting n elements in k rounds whose expected number of comparisons is smaller than any possible deterministic algorithm. This is an immediate corollary of results 3 and 4 below.

Result 3. We present a randomized algorithm whose expected number of comparisons is $O(n^{1+1/k})$.

Result 4. For every deterministic parallel sorting algorithms $c(k, n) = \Omega(n^{1+\frac{1}{k}} (\log n)^{1/k})$.

This improves on Häggkvist and Hell who showed that for every *fixed* k, $c(k, n) = \Omega(n^{1+1/k})$.

Notice that the only difference between our improved lower bound and the previously known one, is an extra factor of $(\log n)^{1/k}$. Nevertheless, this is precisely the factor that separates the asymptotic behavior of the best randomized algorithm from that of the best deterministic one.

Suppose we have to sort n elements and let A be a set of pairs of these elements. Denote $p = |A|$. The set A is an *approximate sorting in one round* if knowing the relative order of each pair in A, provides the relative order of $(1 - o(1))\binom{n}{2}$ out of the $\binom{n}{2}$ pairs without any further comparisons of pairs of elements.

Result 5. We show that here p must be asymptotically bigger than $n \log$ (i.e., $n \log n = o(p)$), thus settling a problem posed by Rabin (cf. [BHe-85]).

Using a similar technique we can show that for every fixed k, $\Omega(n^{1+1/(2^k-1)} \cdot (\log n)^{2/(2^k-1)})$ comparisons are needed to find the median of n elements in k rounds. (An upper bound of $O(n^{1+1/(2^k-1)} \cdot (\log n)^{2-2/(2^k-1)})$ was proved by Pippenger [Pi-86].) This improves by a factor of $(\log n)^{2/(2^k-1)}$ Häggkvist-Hell's lower bound [HH-80] and separates the asymptotic behavior of the best algorithm for selecting the maximum (which is $\Theta(n^{1+1/(2^k-1)})$, see [HH-80]) from that for selecting the median. The detailed proof of this last result will appear somewhere else.

More on the significance of the results. In studying the limit of parallel algorithms it is interesting to identify asymptotically the minimal time k that can be achieved by an optimal speed up algorithm. We call this minimal time the parallelism *break point* of the problem being considered. [Va-75] proved that $\Theta(\log\log n)$ is the break point for finding the maximum among n elements. [BHo-82] gave a lower bound and [Kr-83] an upper bound to prove that $\Theta(\log\log n)$ is the break point for merging two sorted lists, where n is the length of each list. The above two lower bounds were also obtained in a parallel comparison model (which is therefore often referred to as Valiant's model). The present paper enables us to add sorting to the list of problems for which the break point was identified. Specifically, Corollary 4 complements the sorting network of [AKS-83] in proving that $\Theta(\log n)$ is the break point for sorting n elements. It is interesting to compare the "pattern" in which the break point occurs in these three problems. The elegant lower bound proofs of Valiant and Borodin-Hopcroft show that $\Omega(\log\log n)$ rounds are required if n processors are used for the problems of finding the maximum and merging, respectively. The algorithms of Valiant and Kruskal run in $O(\log\log n)$ rounds using $\frac{n}{\log\log n}$ processors for each of these problems, respectively. This isolates distinctly the break points for these two problems since the asymptotic time bound can not be improved by increasing the number of processors from $\frac{n}{\log\log n}$ to n. On the other hand, such degenerate isolation does not occur in the sorting problem. Specifically, Corollary 5 implies that increasing the number of processors asymptotically always yields asymptotic decrease in the number of comparison rounds.

More on extant work. Let us review works on sorting n elements in a parallel comparison model. Recall that Häggkvist and Hell [HH-81] proved that if k, the number of rounds, is *constant*, then $\Omega(n^{1+1/k})$ processors are required to sort n elements. Using random graphs, Bollobás and Thomason [BT-83] proved that there is an algorithm that uses $p=O(n^{3/2}\log n)$ processors and sorts n elements in two rounds. Bollobás and Hell [BH-85] (see also [Pi-86]) showed that n elements can be sorted in a *constant* number of rounds k using $O(n^{1+1/k}\log n)$ comparisons. This almost matches the Häggkvist-Hell lower bound.

Remark. Conversely, these results imply that for $p = O(n^{1+\epsilon})$ processors, it is impossible to sort in less than $k = 1/\epsilon$ rounds, but we can sort in $k = 1/\epsilon + 1$ rounds. So these upper and lower bounds are at most one round apart when k is constant.

However, a closer look at this lower bound of Häggkvist and Hell reveals the following. They actually proved that if k, the number of rounds, is a variable, then $p > \frac{n^{1+1/k}}{2^{k+1}k} - \frac{n}{2k}$ processors are required to sort n elements. For constant k, Result 4 provides an asymptotically better bound. Next, we compare Häggkvist-Hell's result with Corollary 1. Observe, that their proof implies that $p = \Omega(n^{1+1/k})$ *only* when k is constant and therefore for non-constant k Corollary 1 is stronger. Moreover, their result becomes trivial for $k \geq \sqrt{\log n}$. This is since for this range their result implies an asymptotic bound which is $O(n)$ for the number of processors p as can be readily verified. On the other hand, Corollary 1 states that $p > n^{1+1/k}/e - n$ for every k. As was indicated above, this implies that $p = \Omega(n^{1+1/k})$, for any $k < c\log n$, where $0 < c < 1$ is a constant.

We note a few additional papers whose titles are related to the title of the present paper. [Le-84] proposed an adaptation of AKS network to bounded degree n-node networks. [MW-85] gave a $\sqrt{\log n}$ lower bound for parallel sorting by n processors in some variant of PRAM (see also [Be-86] for a stronger result). Their model is not comparable to the parallel comparison model considered here. The trivial $\log n$ lower bound for parallel sorting by n processors in the parallel comparison model does not allow non comparison algorithms like bucket sort. On the other hand, ranking an element among n other elements can be done in one round of comparisons using n processors in the parallel comparison model, while their PRAM seems to require non constant time using n processors.

Results 3 and 4 separate deterministic and randomized complexity for sorting in a fixed number of rounds. A result of a similar flavor for the problem of selecting the l-th out of n elements is known. Specifically, Reischuk [Re-81] gave a randomized comparison parallel algorithm for selection whose expected running time is bounded by a constant, using n processors. Together with the lower bound of [Va-75] for finding the maximum among n elements, we conclude that there exists a randomized algorithm for selection that performs better than any of its deterministic counterparts.

2. TIGHT LOWER AND UPPER BOUNDS FOR NOT NECESSARILY CONSTANT NUMBER OF ROUNDS

2.1 The parallel computation model

Let V be a set of n elements taken from a totally ordered domain. The *parallel comparison model of computation* allows algorithms that work as follows. The algorithm consists of time steps called *rounds*. In each round binary comparisons are performed simultaneously. The input for each comparison are two elements of V. The output of each comparison is one of the following two: $<$ or $>$. Note that we do not allow equality between two elements of V. This can be done without loss of generality, since we define the order between two equal input elements to be the order of their indices. Each item may take part in several comparisons during the same round.

Remark. Our discussion uses the following correspondence between each round and a graph. The elements are the vertices. Each comparison to be performed is an undirected edge which connects its input elements. Each computation results in orienting this edge from the largest element to the smallest. Thus in each round we get an acyclic orientation of the corresponding graph, and the transitive closure of the union of the r oriented graphs obtained until round r represents the set of all pairs of elements whose relative order is known at the end of round r.

Suppose we performed r rounds where $r > 0$ is some integer. Consider any function of V that can be computed using the comparisons performed in these r rounds without any further comparisons of elements in V. Our model defines such a function to be *computable following round r*. Note that this definition suppresses all computational steps that do not involve comparisons of elements in V. Which comparisons to perform at round $r+1$ and the input for each such comparison should be functions which are computable following round r. We are interested in sorting the elements in V from the smallest to the largest in k rounds, where the integer k can be either constant or a function of n.

Recall that $c(k, n)$ denotes the minimum *total* number of comparisons required to sort any n elements in k rounds (over all possible algorithms).

2.2 The lower bound

Let us restate the main theorem of this section.

The Lower Bound Theorem:

$c(k, n) > k\left(\frac{n^{1+\frac{1}{k}}}{e} - n\right)$ for any $k, n \geqslant 1$, where e is the base of the natural logarithm.

Proof. By induction on k and n.

The base of the induction. For $k = 1$ and every $n \geqslant 1$, clearly $c(1, n) = \frac{n^2 - n}{2} > \frac{n^2}{e} - n$. For $n = 1, 2$ and every $k \geqslant 1$ $c(k, 1) \geqslant 0 > k\left(\frac{1}{e} - 1\right)$, $c(k, 2) > 0 > k\left(\frac{4}{e} - 2\right) \geqslant k\left(\frac{2^{1+1/k}}{e} - 2\right)$.

The inductive assumption: Given k, n, if $k' \leqslant k$ and $n' < n$, or $k' < k$ and $n' \leqslant n$, then $c(k', n') > k'\left(\frac{n'^{1+\frac{1}{k'}}}{e} - n'\right)$.

Take any k-round algorithm for sorting a set V of n elements. The first round of the algorithm consists of some set E of comparisons. Recall that we look at them as edges in the graph $G = (V, E)$. An *independent set* in G is a subset of vertices from V such that no two vertices are adjacent by an edge in E. An independent set is *maximal* if it is not a proper subset of another independent set. Consider the graph of the first round of comparisons. Let S be a maximal independent set in this graph and denote $x = |S|$. Each of the $n - x$ elements of $\bar{S}$ must share an edge with an element of S, or otherwise S is not maximal. For our lower bound proof, we restrict our attention to linear orders on V, in which each element of S is greater than each element of $\bar{S}$. For any of these orders it is impossible to obtain any information regarding the relation between two elements of S or two elements of $\bar{S}$ using comparisons between an element of S and an element of $\bar{S}$. Therefore, aside from these $n - x$ comparisons, there must be at least $c(k-1, x)$ comparisons to sort S and at least $c(k, n-x)$ comparisons to sort $\bar{S}$. This implies the following recursion,

$$c(k, n) \geqslant c(k, n-x) + n - x + c(k-1, x),$$

by the inductive assumption

$$> k\left(\frac{(n-x)^{1+1/k}}{e} - (n-x)\right) + (n-x) + (k-1)\left(\frac{x^{1+1/(k-1)}}{e} - x\right).$$

By opening parentheses and permuting terms we get

$$\frac{k}{e}(n-x)^{1+1/k} + \frac{k-1}{e}x^{1+1/(k-1)} + n - kn = \frac{k}{e}n^{1+1/k}\left[\left(1 - \frac{x}{n}\right)^{1+1/k} + \left(1 - \frac{1}{k}\right)\frac{x^{1+1/(k-1)}}{n^{1+1/k}} + \frac{1}{k}\frac{e}{n^{1/k}}\right] - kn.$$

Recall the Geometric-Arithmetic Mean Inequality: $\alpha a + \beta b \geqslant a^{\alpha} b^{\beta}$, where $\alpha + \beta = 1$ $\alpha, \beta, a, b \geqslant 0$. By taking

$$\alpha = 1 - \frac{1}{k}, \beta = \frac{1}{k}, a = \frac{x^{1+1/(k-1)}}{n^{1+1/k}}, b = \frac{e}{n^{1/k}},$$

we get that the last expression is

$$\geqslant \frac{k}{e} n^{1+1/k} \left[\left(1 - \frac{x}{n} \right)^{1+1/k} + \frac{x}{n^{1-1/k^2}} \frac{e^{1/k}}{n^{1/k^2}} \right] - kn .$$

Recall that the increasing sequence $\left(1 + \frac{1}{k}\right)^k$ converges to e and therefore, $e^{1/k} > 1 + \frac{1}{k}$. This implies

$$\geqslant \frac{k}{e} n^{1+1/k} \left[\left(1 - \frac{x}{n} \right)^{1+1/k} + \frac{x}{n} \left(1 + \frac{1}{k} \right) \right] - kn .$$

Recall Bernoulli's Inequality: $(1 - \alpha)^t \geqslant 1 - \alpha t$ for $t \geqslant 1$, $\alpha \leqslant 1$. This implies,

$$\geqslant \frac{k}{e} n^{1+1/k} \left[1 - \frac{x}{n} \left(1 + \frac{1}{k} \right) + \frac{x}{n} \left(1 + \frac{1}{k} \right) \right] - kn$$

$$= \frac{k}{e} n^{1+1/k} - kn = k \left(\frac{n^{1+1/k}}{e} - n \right) .$$

This completes the proof of the Lower Bound Theorem.

2.3 The upper bound

Theorem. Given n elements from a totally ordered domain, there is an explicit algorithm in a parallel comparison model for sorting these elements in $O\left(\frac{\log n}{\log(1 + p/n)}\right)$ rounds using $p \geqslant n$ processors.

Proof: First recall the AKS comparison network. It sorts n elements in $O(\log n)$ rounds using $p = n/2$ processors (i.e., $n/2$ comparisons in each round). We give an algorithm in a parallel comparison model. Each round of the new algorithm is called *superround*. The algorithm is derived from AKS network by simply shrinking $\delta = 0.5\log(1+p/n)$ rounds of this network into one superround.

The construction of the algorithm is based on the following idea. We aim that the following Assertion will hold.

Assertion. After superround r, the following things are available: (1) The pair of input elements for each comparison performed in the first δr rounds of AKS network. (2) The result of each such comparison.

This Assertion implies that after $O(\frac{\log n}{\log(1+p/n)})$ superrounds the results of all comparisons of AKS network are available and the sorting is completed (since it is computable).

We show how to satisfy the Assertion for any superround r. For $r = 0$ the Assertion trivially holds. We show how to satisfy the Assertion for superround r assuming that it is satisfied for any superround $< r$.

The fact that we relate to a comparison network implies that each element, which is compared in round $\delta(r - 1) + i$, where $1 \leqslant i \leqslant \delta$, is one of at most 2^{i-1} elements which are outputs of comparisons of the first $\delta(r - 1)$ rounds (or input elements). By the inductive assumption, each of these outputs is available following superround $r - 1$. Therefore, each comparison in round $\delta(r - 1) + i$, is actually one of $(2^{i-1})^2$ possible pairs. All we do is perform all these possible comparisons simultaneously (for $1 \leqslant i \leqslant \delta$). These comparisons clearly include the actual comparisons performed by AKS network in these rounds. It remains to show that this construction also yields the pairs of input elements to each comparison which was actually performed in each of these rounds. For this we show by simple induction that the actual pair of each comparison, as well as its result are available, for all rounds $\leqslant \delta(r - 1) + i$, $0 \leqslant i \leqslant \delta$. For $i = 0$, this follows from the inductive assumption of the Assertion for $r - 1$. Suppose that for all rounds $< \delta(r - 1) + i$, the actual pairs compared, as well as their result are available. Each element participating in round $\delta(r - 1) + i$ is an outcome of the actual comparisons of preceding rounds and their results. They are known by the inductive assumption. Therefore, the input pair for each such comparison is known. We already argued that the result of each such comparison was found by our algorithm. This completes the proof of the induction for i. Taking $i = \delta$, we complete the inductive proof of the Assertion.

The number of comparisons that the algorithm has to perform in each superround is:

$$\frac{n}{2} \sum_{i=1}^{\delta} (2^{i-1})^2 < \frac{n}{2} \frac{4^{\delta}}{3} < \frac{n}{2} \cdot 2^{2\delta} .$$

But $\delta = 0.5\log(1 + \frac{p}{n})$, and therefore, this number of comparisons is not more than $\frac{n}{2} 2^{\log(1+\frac{p}{n})} \leqslant \frac{n}{2} \left(1 + \frac{p}{n}\right) = \frac{n + p}{2} \leqslant p$. So, there are enough processors to perform all these comparisons.

3. SORTING IN A FIXED NUMBER OF ROUNDS

3.1 Randomized algorithms

Theorem 3.1 For any $k \geqslant 1$, there is an explicit randomized algorithm for sorting n elements in k rounds, whose expected number of comparisons $E(n, k)$ is at most $c(k) \cdot n^{1+1/k}$, where $c(k)$ is some constant depending on k only.

Proof. By induction on k. For $k = 1$ the result is trivial. Assuming it holds for $k - 1$ and every n, we prove it for k. Put $t = \lceil n^{1/k} \rceil$. In the first round our algorithm chooses randomly a set T of $t - 1$ elements from the set V of n elements we have to sort and compares each of them to every $v \in V$ (including the other elements of T). After this round, the set $V - T$ will be broken into t blocks $A_1, A_2, \ldots, A_t$, such that for each $i < j$ and $a_i \in A_i$, $a_j \in A_j$ a_i is smaller than a_j.

We now apply, recursively, our randomized algorithm for sorting in $k-1$ rounds, to each A_i, in parallel. We claim that

$$E(n,k) \leqslant (t-1)\cdot n + t \sum_{i=1}^{n-t+2} \left(\frac{\binom{n-i}{t-2}}{\binom{n}{t-1}} \right)$$

$$E(i-1,k-1) \leqslant n^{1+1/k} + \frac{t(t-1)}{n} \sum_{i=1}^{n-t+2} \left(\frac{n-i}{n-1} \right)^{t-2} c(k-1)\cdot (i-1)^{1+\frac{1}{k-1}} .$$

Indeed, there are $\binom{n}{t-1}$ ways to choose the set T, and for each fixed j, $1 \leqslant j \leqslant t$, the number of these choices with $|A_j| = i-1$ (for $1 \leqslant i \leqslant n-t+2$) is precisely the number of ways to write $n-t-i+2$ as an ordered sum of $t-1$ non-negative integers (representing the cardinalities of the blocks A_s besides A_j), which is $\binom{n-i}{t-2}$.

To estimate the right hand side of the last inequality we break the sum into consecutive blocks, each of size $\sim (n-1)/(t-2) \approx n^{1-\frac{1}{k}}$, and notice that

$$\sum_{(j-1)\frac{n-1}{t-2} < i \leqslant j\frac{n-1}{t-2}} \left(\frac{n-i}{n-1} \right)^{t-2} \cdot (i-1)^{1+\frac{1}{k-1}}$$

$$\leqslant (1+o(1))\, n^{1-\frac{1}{k}} \cdot e^{-(j-1)} \cdot j^{1+\frac{1}{k-1}} \cdot n .$$

Since the sum $\sum_{j \geqslant 1} j^{1+\frac{1}{k-1}} \cdot e^{-j}$ converges, this implies that $E(n,k) \leqslant c(k) \cdot n^{1+\frac{1}{k}}$ for a properly defined constant $c(k)$. This completes the proof.

Remark 3.2

We can show that Theorem 3.1 is sharp for $k=2$ in the sense that for every randomized algorithm for sorting n elements in two rounds there is an input for which the expected number of comparisons of the algorithm is $\Omega(n^{3/2})$. We do not know if the theorem is sharp for larger values of k.

3.2 Lower and upper deterministic bounds

Even the first nontrivial case, that of sorting n elements in two rounds, received considerable attention. Häggkvist and Hell [HH-81] showed that

$$\tfrac{1}{8}n^{3/2} - \tfrac{1}{2}n \leqslant c(2,n) = O(n^{5/3}\log n) .$$

Bollobás and Thomason [BT-83] improved it and showed that

$$c_1 \cdot n^{3/2} \leqslant c(2,n) = O(n^{3/2}\log n)$$

for any $c_1 < \sqrt{2/3}$, if $n > n(c_1)$.

Explicit algorithms for sorting in two rounds with $o(n^2)$ comparisons are given in [Pi-85], [Al-85] and [Pi-86].

Here we slightly improve both bounds and show

Theorem 3.3

$$\Omega(n^{3/2}\sqrt{\log n}) \leqslant c(2,n) \leqslant O\left(n^{3/2} \frac{\log n}{\sqrt{\log\log n}} \right) .$$

We also prove:

Theorem 3.4

For every fixed $k \geqslant 2$

$$c(k,n) = \Omega(n^{1+1/k}\ (\log n)^{1/k}) .$$

An upper bound of $O(n^{1+1/k}\log n)$ for $c(k,n)$ is known, as indicated in the introduction.

Our methods also enable us to improve the known bounds for approximate sorting in one round. An algorithm that approximately sorts n elements in one round with p comparisons is a set of p pairs of elements $(a_i, b_i)_{i=1}^{p}$ from the set V of n elements we have to sort, such that for each possible set of answers for the p questions "is $a_i < b_i$" the relative order of all but $o(n^2)$ of the pairs of elements of V will be known. Let $a(n)$ denote the minimum p such that an approximate sorting algorithm in one round with p comparisons exists. Bollobás and Rosenfeld studied these algorithms in [BR-81] (also see [BHe-85]) and their results imply that for every fixed $\epsilon > 0$ $a(n) = o(n^{1+\epsilon})$.

M. Rabin (cf. [BHe-85]) asked whether $a(n) = O(n\log n)$. The next proposition shows that this is false.

Proposition 3.5

(i) $\lim_{n\to\infty} a(n)/n\log n \to \infty$. More precisely; for every $\epsilon > 0$, any two rounds sorting algorithm that uses at most ϵn^2 comparisons in the second round must use $\Omega(\frac{1}{\epsilon} n\log n)$ comparisons in the first round.

(ii) For any function $\omega(n) \to \infty$,

$$a(n) \leqslant n \cdot \log n \cdot \log\log n \cdot \omega(n) .$$

The upper bounds in Theorem 3.3 and in Proposition 3.5 are proved by combining certain probabilistic arguments with some of the ideas of [BT-83] and [Pi-86]. The details will appear somewhere else. Here we

present the proofs of the lower bounds in Theorems 3.3, 3.4 and Proposition 3.5. A crucial lemma here is the following result.

Lemma 3.6

Every graph with n vertices and at most dn edges, contains an induced subgraph with $\lfloor n/4 \rfloor$ vertices and maximum degree at most $4d$ which has a $4d$ proper vertex coloring with color classes $V_1, V_2, \ldots, V_{4d}$ such that for each $1 \leq i < i + j \leq 4d$ and each $v \in V_i$, v has at most 2^{j+1} neighbors in V_{i+j}.

Proof. Let $G = (V, E)$ be a graph with n vertices and at most dn edges. Since the sum of the degrees of all vertices of G is at most $2dn$, not more than half of the vertices have degrees $\geq 4d$, and thus G contains an induced subgraph K on a set U of at least $n/2$ vertices with maximum degree smaller than $4d$. By a standard result from extremal graph theory, K has a proper $4d$ vertex coloring. Let $U_1, U_2, \ldots, U_{4d}$ be the color classes. For every vertex u of K, let $N(u)$ denote the set of all its neighbors in K. For a permutation π of $1, 2, \ldots, 4d$ and any vertex u of K, define the π-degree $d(\pi, u)$ of u as follows; let i satisfy $u \in U_{\pi(i)}$ then

$$d(\pi, u) = \sum_{j=0}^{4d-i} |N(u) \cap U_{\pi(i+j)}| / 2^j .$$

We claim that the expected value of $d(\pi, u)$ over all permutations π of $\{1, \ldots, 4d\}$ is at most 1. Indeed, for a random permutation π the probability that a fixed neighbor v of u will contribute $1/2^r$ to $d(\pi, u)$ is at most $1/4d$ for all $r > 0$. Hence each neighbor contributes to this expected value at most $\frac{1}{4d} \sum_{r>0} 1/2^r = 1/4d$, and the desired result follows since $|N(u)| \leq 4d$.

Consider now the sum $\sum_{u \in U} d(\pi, u)$. The expected value of this sum (over all π's) is at most $|U|$, by the preceding paragraph. Hence there is a fixed permutation σ such that $\sum_{u \in U} d(\sigma, u) \leq |U|$. It follows that $d(\sigma, u) \leq 2$ for at least $|U|/2 \geq n/4$ vertices u of K. Let W be a set of $\lfloor n/4 \rfloor$ of these vertices, let H be the induced subgraph of G on W and define $V_i = U_{\sigma(i)} \cap W$ $(1 \leq i \leq 4d)$. Clearly, for every $1 \leq i < 4d$ and every $v \in V_i$

$$\sum_{j>0} |N(u) \cap V_{i+j}| / 2^j \leq 2$$

and thus v has at most 2^{j+1} neighbors in V_{i+j}. This completes the proof.

Lemma 3.7

Every graph $G = (V, E)$ with n vertices and at most $n \cdot d$ edges, where $d = o(n)$ and $d = \Omega(\log n)$, has an acyclic orientation whose transitive closure has at most $\binom{n}{2} - \Omega\left(\frac{n^2}{d} \log \frac{n}{d}\right)$ edges.

Proof. By Lemma 3.6 there is a subset W of cardinality $\lfloor n/4 \rfloor$ of V and a proper $4d$ vertex coloring of the induced subgraph of G on W with color classes $V_1, \ldots, V_{4d}$ satisfying the conclusions of the lemma. Put $V_0 = V - (V_1 \cup \ldots \cup V_{4d})$ and orient each edge (u, v) of G with $u \in V_i$, $v \in V_j$ and $0 \leq i < j \leq 4d$ from u to v. The other edges of V (that join two members of V_0) will be oriented in an arbitrary acyclic order. Let T be the transitive closure of this oriented graph. For $v \in V$, let $N_T(v)$ denote the set of neighbors of v in T. Suppose $v \in V_i$, $1 \leq i < i + j \leq 4d$. We claim that the number of directed paths in our oriented G that start at v and end at some member of $\bigcup_{r=1}^{j} V_{i+r}$ is at most 2^{3j}. Indeed, each such path must be of the form $v, v_{i_1} v_{i_2} \cdots v_{i_r}$, where

$$i < i_1 < i_2 < \ldots < i_r \leq i + j , \; v_{i_1} \in V_{i_1}, \ldots, v_{i_r} \in V_{i_r} .$$

There are 2^j possibilities for choosing $i_1, i_2, \ldots, i_r$, and since each vertex of the path is a neighbor of the previous one, there are at most $2^{i_1 - i + 1}$ choices for v_{i_1}, $2^{i_2 - i_1 + 1}$ choices for v_{i_2}, etc. Hence the total number of paths is smaller than 2^{3j} and thus if $v \in V_i$ then

$$|N_T(v) \cap (\bigcup_{r=1}^{j} V_{i+r})| < 2^{3j} .$$

Put $r = \lfloor \frac{1}{6} \log_2 \left(\frac{n}{4d}\right) \rfloor$, and partition the set of blocks $V_1, V_2, \ldots, V_{4d}$ into $s = \lceil \frac{4d}{r} \rceil$ blocks $W_1, \ldots, W_s$ of consecutive V_{i-s}, each containing at most r blocks. By the preceding paragraph, the total number of edges of T in each block W_i is not bigger than $|W_i| \cdot 2^{3r} \leq |W_i| \sqrt{\frac{n}{4d}}$. Thus there are at least

$$\sum_{i=1}^{s} \binom{|W_i|}{2} - \frac{n}{4} \left(\frac{n}{4d}\right)^{1/2}$$

pairs of elements that are not adjacent in T. By the convexity of the function $g(x) = \binom{x}{2}$

$$\sum_{i=1}^{s} \binom{|W_i|}{2} \geq s \binom{\frac{n}{4s}}{2} = \Omega\left(\frac{n^2 \log (\frac{n}{d})}{d}\right)$$

$$= \Omega\left(\frac{n^2 \log \frac{n}{d}}{d}\right)$$

and thus T does not contain at least

$$\Omega\left(\frac{n^2}{d}\log\frac{n}{d}\right) - \frac{n}{4}\left(\frac{n}{4d}\right)^{1/2} = \Omega\left(\frac{n^2}{d}\log\frac{n}{d}\right)$$

edges. This completes the proof.

We can now prove the lower bounds in Theorems 3.3, 3.4 and in Proposition 3.5.

To prove the lower bound in Theorem 3.3, consider any two rounds algorithm that sorts a set V of n elements. The first round of the algorithm consists of some set E of comparisons. Define d by $|E| = n \cdot d$. Clearly we may assume that $d = o(n^{2/3})$ and $d = \Omega(n^{1/3})$. By Lemma 3.7 the graph $G = (V, E)$ has an acyclic orientation whose transitive closure misses $\Omega\left(\frac{n^2}{d}\log n\right)$ edges. If the answers in the first round correspond to this orientation then clearly in the second round the algorithm has to compare all these $\Omega\left(\frac{n^2}{d}\log n\right)$ pairs. Thus, by the trivial inequality $a + b \geqslant 2\sqrt{ab}$

$$c(2, n) \geqslant nd + \Omega\left(\frac{n^2}{d}\log n\right)$$

$$\geqslant \Omega(n^{3/2}(\log n)^{1/2}) \text{ , as needed .}$$

The proof of Proposition 3.5 part (i) is analogous. If a two rounds sorting algorithm uses $c \cdot n \log n$ comparisons in the first round, then by Lemma 3.7, it must use $\Omega\left(\frac{n^2}{c}\right)$ comparisons in the second round.

Theorem 3.4 is derived from the lower bound of Theorem 3.3 proved above by induction on k, starting with $k = 2$. For $k = 2$, the result is just the statement of Theorem 3.3. Suppose, by induction, that $c(k, n) \geqslant c_k\, n^{1+\frac{1}{k}} (\log n)^{1/k}$, where $c_k > 0$ is a constant, depending only on k. Consider an algorithm for sorting a set V of n elements in $k + 1$ rounds. Let E be the set of comparisons between pairs of elements of V made in the first round. As before, E corresponds to a set of edges of a graph $G = (V, E)$. Define d by $|E| = d \cdot n$. By a standard result from extremal graph theory (that follows, e.g., from the trivial part of Lemma 3.6), any graph with m vertices and average degree f, contains an independent set of size $\Omega(m/f)$. By a repeated application of this, we conclude that G contains $\Omega(d)$ independent sets, each of size $\Omega(n/d)$. Denote these sets by $V_1, \ldots, V_s$ $(s = \Omega(d))$ and define $V_0 = V - \bigcup_{i=1}^{s} V_i$. Restrict our attention now only to linear orders on V for which each $v_i \in V_i$ is smaller than each $v_j \in V_j$, for all $0 \leqslant i < j \leqslant s$. Clearly, if $0 < i \leqslant s$, and $u, v \in V_i$ we do not have any information about the relative order of u and v from the results of the first round, and such an information can be obtained only from comparisons between elements of V_i. Thus, in the next k rounds, all the sets $V_1, \ldots, V_s$ have to be sorted. By the induction hypothesis the number of comparisons for this task is at least

$$\sum_{i=1}^{s} c_k\, |V_i|^{1+1/k} (\log |V_i|)^{1/k}$$

$$\geqslant \Omega_k\left(d \cdot \left(\frac{n}{d}\right)^{1+1/k} \cdot \left(\log\left(\frac{n}{d}\right)\right)^{1/k}\right) .$$

The total number of comparisons is thus at least

$$nd + \Omega(n^{1+1/k}(\log\frac{n}{d})^{1/k}/d^{1/k}) .$$

One can easily check that this number is $\geqslant \Omega(n^{1+1/(k+1)} \cdot (\log n)^{1/(k+1)})$. (Indeed, at least one of the two summands must be that big). This completes the induction and Theorem 3.4 follows.

Acknowledgments.

We would like to thank A. Borodin, M. Dubiner and M. Paterson for helpful comments.

REFERENCES

[Ak-85] S. Akl, "Parallel Sorting Algorithms", Academic Press, 1985.

[Al-85] N. Alon, Expanders, sorting in rounds and superconcentrators of limited depth, Proc. 17th ACM Symposium on Theory of Computing (1985), 98-102.

[AKS-83] M. Ajtai, J. Komlós and E. Szémerédi, An $O(n\log n)$ sorting network, Proc. 15th ACM Symposium on Theory of Computing (1983), 1-9. Also, M. Ajtai, J. Komlós and E. Szémerédi, Sorting in $c\log n$ parallel steps, Combinatorica 3(1983), 1-19.

[AV-86] Y. Azar and U. Vishkin, Tight comparison bounds on the complexity of parallel sorting, SIAM J. Comput., to appear.

[Be-86] P. Beame, Limits on the power of concurrent-write parallel machines, Proc. 18th ACM Symposium on Theory of Computing (1986), 169-176.

[BR-82] B. Bollobás and M. Rosenfeld, Sorting in one round, Israel J. Math. 38(1981) 154-160.

[BT-83] B. Bollobás and A. Thomason, Parallel sorting, Discrete Applied Math. 6(1983) 1-11.

[BHe-85] B. Bollobás and P. Hell, Sorting and Graphs, *in: Graphs* and orders, I. Rival ed., D. Reidel (1985), 169-184.

[BO-86] B. Bollobás, Random Graphs, Academic Press (1986), Chapter 15 (Sorting algorithms).

[BHo-82] A. Borodin and J.E. Hopcroft, Routing, merging and sorting on parallel models of computation, Proc. 14th ACM Symposium on Theory of Computing (1982), 338-344.

[HH-80] R. Häggkvist and P. Hell, Graphs and parallel comparison algorithms, Congr. Num. 29(1980) 497-509.

[HH-81] R. Häggkvist and P. Hell, Parallel sorting with constant time for comparisons, SIAM J. Comput. 10(1981) 465-472.

[HH-82] R. Häggkvist and P. Hell, Sorting and merging in rounds, SIAM J. Alg. and Disc. Math. 3(1982) 465-473.

[Kn-73] D.E. Knuth, "The Art of Computer Programming, Vol. 3: Sorting and Searching", Addison Wesley 1973.

[Kr-83] C.P. Kruskal, Searching, merging and sorting in parallel computation, IEEE Trans. Computers c-32(1983) 942-946.

[Le-84] F.T. Leighton, Tight bounds on the complexity of parallel sorting, Proc. 16th ACM Symposium on Theory of Computing (1984) 71-80.

[Pi-85] N. Pippenger, Explicit construction of highly expanding graphs, preprint (1985).

[Pi-86] N. Pippenger, Sorting and selecting in rounds, preprint (1986).

[Th-83] C. Thompson, The VLSI complexity of sorting, IEEE Trans. Computers C-32, 12(1983).

[Re-81] R. Reischuk, A fast probabilistic sorting algorithm, Proc. 22nd IEEE Symp. on Foundations of Computer Science (1981), 212-219.

[RV-83] J. Reif and L.G. Valiant, A logarithmic time sort for linear size network, Proc. 15th ACM Symposium on Theory of Computing (1983) 10-16.

[SV-81] Y. Shiloach and U. Vishkin, Finding the maximum, merging and sorting in a parallel model of computation, J. Algorithms 2,1 (1981) 88-102.

[Va-75] L.G. Valiant, Parallelism in comparison problems, SIAM J. Comp. 4(1975) 348-355.