The Computational Complexity Column

by

Lance FORTNOW

NEC Research Institute
4 Independence Way, Princeton, NJ 08540, USA
fortnow@research.nj.nec.com

This has been quite an exciting summer for computational complexity. The 15th Conference on Computational Complexity held July 4-7 in Florence was a great success. We had many interesting talks and a wonderful invited talks and tutorials by Paul Beame, Luca Trevisan and Russell Impagliazzo. Preparations are underway for the 16th Conference next summer in Chicago.

The Institute of Advanced Studies in Princeton had a three-week summer session on computational complexity theory. This included a graduate school giving introductions to many of the current exciting research areas in complexity theory. Lecturers included Sanjeev Arora, Paul Beame, Michael Ben-Or, Oded Goldreich, Ran Raz, Steven Rudich, Luca Trevisan and Salil Vadhan. The summer session is part of a 2000-2001 special year on computational complexity at the institute.

Our column has a new web site: http://www.neci.nj.nec.com/homepages/fortnow/beatcs. Most of the previous columns are available there as well as pointers to interesting complexity sites such as described above.

This month we feature a survey on parallel sorting where we have a fixed number of rounds. If you only have one round than all comparisons are needed. Surprisingly even for two rounds computing the optimal number of rounds is a surprisingly difficult problem.

A Survey of Constant Time Parallel Sorting

William Gasarch[1] Evan Golub[2] Clyde Kruskal[3]

1 Introduction

It is well known that sorting can be done with $O(n \log n)$ comparisons. It is also known that (in the comparison decision tree model) sorting requires $\Omega(n \log n)$ comparisons.

What happens if you allow massive parallelism? In the extreme case you can sort n elements in one round by using $\binom{n}{2}$ processors to make all the comparisons at once. It is easy to show that sorting in one round *requires* $\binom{n}{2}$ processors. Can you sort in two rounds with a subquadratic number of processors? What about k rounds? We survey the known literature.

[1]Dept. of C.S. and Inst. for Adv. Comp. Stud., University of MD., College Park, MD 20742, U.S.A. Supported in part by NSF grant CCR–9732692. (Email: gasarch@cs.umd.edu.)

[2]Dept. of C.S., University of MD., College Park, MD 20742, U.S.A. (Email: egolub@acm.org.)

[3]Dept. of C.S., University of MD., College Park, MD 20742, U.S.A. Supported in part by NSF grant CCR–9732692. (Email: kruskal@cs.umd.edu.)

We use the parallel decision tree model introduced by Valiant [22]. If p processors are used then every node is a set of p comparisons and has $p!$ children corresponding to all possible answers. We think of a node as having information about how the comparisons that led to that node were answered (formally a dag on $\{x_1, \ldots, x_n\}$) and all information derivable from that information (formally the transitive closure of that dag). The model does not count the cost of communication between processors, nor does it count the cost of transitive closure. The model does capture how hard it is to gather the *information* needed to sort. Also, lower bounds in our model will apply to models that take these factors into account. For more realistic models of parallelism see any current textbook on parallel algorithms, e.g. [1, 15].

The first round of a p-processor algorithm takes $x_1, \ldots, x_n$ about which nothing is known and makes p comparisons. This can be represented as an undirected graph G on n vertices with p edges. Hence the search for parallel sorting algorithms will involve finding graphs G that have nice properties. Most of our algorithms depend on versions of the following two lemmas which we state informally:

1. Some undirected graphs G with property P exists and does not have too many edges.

2. Let $G = (V, E)$ be a graph with $V = \{x_1, \ldots, x_n\}$ that has property P. Let G' be any acyclic orientation of G and let H be the transitive closure of G'. The graph $\overline{H}$ is not too large.

2 Definitions and Notation

There are several types of sorting algorithms.

Definition 2.1

1. A *nonconstructive algorithm for sorting* n *elements in* k *rounds* is an algorithm that is proven to exist, but its existence proof does not reveal how to produce it. For example, the graph on n vertices that represents the first round may be proven to exist by the probabilistic method [5, 20].

2. A *constructive algorithm for sorting in* k *rounds* is a sequence of algorithms $\mathcal{A}_n$ with the following properties: (1) The algorithm $\mathcal{A}_n$ sorts n elements in k rounds. (2) There is a polynomial time algorithm that, given n (in unary), produces $\mathcal{A}_n$.

3. A *randomized algorithm* to sort n elements is one that flips coins. For these algorithms we could fix the number of processors and ask what the expected number of rounds is, or we could fix the number of rounds and ask what the expected number of processors is. We will do the latter. The former can be found in [15].

We noted above that the model assumes transitive closure is free. Some of our algorithms work with the weaker assumption that only a partial transitive closure is free.

Definition 2.2

1. Given a directed graph G the *2-step transitive closure* is the graph formed as follows: If in our original comparison graph we have (x, y) and (y, z) then we will add to that graph (x, z). Note that if we have (x, y), (y, z), and (z, w) we *do not* add (x, w).

2. Let $d \geq 2$. Given a directed graph G the *d-step transitive closure* is the graph defined inductively as follows: (1) the 2-step transitive closure is as above. (2) the d-step transitive closure is the 2-step transitive closure of the $(d-1)$-step transitive closure.

Definition 2.3

1. $\mathrm{sort}(k, n)$ is the number of processors needed to sort n elements in k steps. The algorithms may be nonconstructive.
2. $\mathrm{csort}(k, n)$ is the number of processors needed to sort n elements in k steps by means of a constructive algorithm.
3. $\mathrm{sort}(k, n, d)$ is the number of processors needed to sort n elements in k steps by means of an algorithm that only uses d-step transitive closure. The algorithm may be nonconstructive.
4. $\mathrm{csort}(k, n, d)$ is the number of processors needed to sort n elements in k steps by means of a constructive algorithm that only uses d-step transitive closure.
5. $\mathrm{rsort}(k, n)$ is the expected number of processors in the best randomized algorithm for sorting in k rounds.

Note 2.4 When we use order notation we take k to be a constant. Hence a statement like "$\mathrm{sort}(k, n) = O(n^{1+1/k}(\log n)^{2-2/k})$" means that the multiplicative constant might depend on k.

We survey all known upper and lower bounds on the quantities in Definition 2.3. We will not give full proofs; rather, we give proof sketches. We have two goals:

1. The reader should learn that there are many interesting constant-time parallel sorting algorithms in the literature and that they use mathematics of interest.
2. The reader should be inspired to look into the literature for more detail on some algorithms of interest.

A companion paper (in preparation) will discuss what happens if you try to code up these algorithms. This companion paper will be largely based on Evan Golub's thesis [12]. Bollobás and Hell [7] wrote a survey on graphs and orders which contained (among other things) information on constant round sorting as of 1985. Our paper can be considered an updated version of that part of their paper.

3 Nonconstructive Methods

3.1 The First Nonconstructive Algorithm

The first k-round sorting algorithm that uses a subquadratic number of processors is due to Haggkvist and Hell [13]. They showed that $\mathrm{sort}(k, n) \leq O(n^{\alpha_k} \log_2 n)$ where $\alpha_k = \frac{3 \cdot 2^{k-1} - 1}{2^k - 1}$.

Lemma 3.1 ([13]) *Let* $\frac{3}{2} < \alpha < 2$. *Let* $p = \lfloor n^{2-\alpha} \rfloor$, $q = \lfloor n^{4-2\alpha} \rfloor$, *and* $r = \lfloor 2n^{4\alpha-6} \log_2 n \rfloor$. *There exists a graph* B *on* n *vertices with* pqr *edges such that* $\overline{G}$ *does not contain* $K_{p,q}$.

Proof sketch: Let

$A = |\{G : G \text{ has } n \text{ vertices and } pqr \text{ edges }\}|$,

$B = |\{G : G \text{ has } n \text{ vertices and } pqr \text{ edges and } \overline{G} \text{ has } K_{p,q}\}|$,

$C =$ the number of $K_{p,q}$ in K_n.

Some (non-trivial) algebra shows $\frac{AC}{B} > n^{p+q}$ and $C < n^{p+q}$. Hence $A > B$. Therefore there are graphs in $A - B$. (A more careful calculation shows that if you picked a graph at random from A then the probability of getting one in $A - B$ is close to 1.) ▮

Definition 3.2 Let $n, t \in \mathsf{N}$. We define the graph $P_n^t = (V, E)$ by

$V = \{1, \ldots, n\}$.

$E = \{(i, j) : |i - j| \leq t\}$.

Note that P_n^t has $\leq tn$ edges. Also note that all the edges are entirely contained in one of the following (non-disjoint) sets:

$\{jt + 1, \ldots, (j + 2)t\}$ as $j = 0, 1, \ldots, \lfloor \frac{n}{t} \rfloor - 2$

$\{n - 2t + 1, \ldots, n\}$.

Lemma 3.3 ([13]) *Let $n, p, q \in \mathsf{N}$ and G be such that $p^2 \leq q$, G is an undirected graph on n vertices, and $\overline{G}$ does not contain $K_{p,q}$. Let G' be any acyclic orientation of G. Let H be the transitive closure of G'. Then there exists a set of vertices W such that $|W| \leq \frac{2np}{q}$ and $\overline{G} - W$ is a subgraph of P_n^{7q}. Hence $\overline{G}$ has $O(\frac{p^2 n}{q} + qn)$ edges.*

The upper bound on $\mathrm{sort}(2, n)$ is simpler than the upper bound on $\mathrm{sort}(k, n)$ hence we present it separately.

Theorem 3.4 ([13]) $\mathrm{sort}(2, n) \leq O(n^{5/3} \log n)$.

Proof: Let α, p, q, r, n and G be as in Lemma 3.1. We will set α later.

ALGORITHM

Step 1: (Round 1) Compare $x_i : x_j$ iff (i, j) is an edge of G. (This takes $pqr = O(n^\alpha \log n)$ comparisons.) Let G' be the orientation of G obtained by directing i to j iff $x_i < x_j$. Let H be the transitive closure of G'.

Step 2: (Round 2) Compare all $x_i : x_j$ such that (i, j) is not an edge of H. (By Lemma 3.3 this takes $O(\frac{p^2 n}{q} + qn) = O(n^\alpha + n^{5-2\alpha})$ comparisons.)

END OF ALGORITHM

Set $\alpha = \frac{5}{3}$ to obtain the result. ▮

Theorem 3.5 ([13]) $\mathrm{sort}(k, n) \leq O(n^{\alpha_k} \log n)$ *where* $\alpha_k = \frac{3 \cdot 2^{k-1} - 1}{2^k - 1}$.

Proof sketch: We prove this by induction on k. The $k = 1$ case is obvious. The $k = 2$ case is Theorem 3.4.

Assume the theorem is true for $k - 1$. Let α, n, p, q, r and G be as in Lemma 3.1. We will pick α later.

ALGORITHM

Step 1: (Round 1) Compare $x_i : x_j$ iff (i,j) is an edge of G. (This takes $pqr = O(n^\alpha \log n)$ comparisons.) Let G' be the orientation of G obtained by directing i to j iff $x_i < x_j$. Let H be the transitive closure of G'.

Step 2: (Rounds $2,\ldots,k$) By Lemma 3.3 $\overline{G}$ is the union of W and P_n^{7q}. Recall that P_n^{7q} can be viewed as the (non-disjoint) union of $O(\frac{n}{q})$ graphs, each of $O(q)$ vertices. For each of these $O(\frac{n}{q}) = O(n^{2\alpha-3})$ graphs use (inductively) $O(q^{\alpha_{k-1}} \log q) = O(n^{(4-2\alpha)\alpha_{k-1}} \log n)$ processors to sort it in $k-1$ rounds. This requires $O(n^{2\alpha-3+(4-2\alpha)\alpha_{k-1}} \log n)$ processors. At the same time compare all (x,y) where $x \in W$ and $y \in \{x_1,\ldots,x_n\}$. This is $O(n|W|) = O(n^\alpha)$ comparisons. If you set $\alpha = \alpha_k$ then all the rounds use $O(n^{\alpha_k} \log n)$ processors.

END OF ALGORITHM ▮

3.2 An Improvement in the $k = 2$ Case

Bollobás and Thomason [9] improved upon Theorem 3.4 in the $k = 2$ case.

The following lemma is implicit in [9].

Lemma 3.6 *There exists a graph G such that (1) G has $O(n^{3/2} \log n)$ edges, and (2) If G' is any acyclic orientation of G and H is the transitive closure of G' then $\overline{H}$ has $O(n^{3/2} \log n)$ edges.*

Proof sketch: Assume you have a coin that has probability $p = \Theta(\frac{\log n}{\sqrt{n}})$ of being heads. Create a graph on n vertices as follows: for each $\{i,j\}$, flip the coin. Put the edge $\{i,j\}$ into the graph iff the coin is heads. The probability that the graph will be have the desired properties is nonzero (actually close to 1). Hence such a graph exists. ▮

Theorem 3.7 ([9]) $\text{sort}(2,n) \le O(n^{3/2} \log n)$.

Proof sketch: Let G be a graph shown to exist by Lemma 3.6. Use this graph and proceed similar to Theorem 3.4. ▮

Note 3.8 Some authors have credited [7] or [9] with the result $\text{sort}(k,n) = O(n^{1+1/k} \log n)$. This citation is incorrect and this result is not known to be true. However, Bollobás [6] later obtained $\text{sort}(k,n) = O(n^{1+1/k} \frac{(\log n)^{2-2/k}}{(\log \log n)^{1-1/k}})$ (see Section 3.4).

3.3 Expander Graphs

Pippenger [18] showed that $\text{sort}(k,n) = O(n^{1+1/k} (\log n)^{2-2/k})$

Definition 3.9 [18] Let $1 \le a \le n/2$. An a-expanding graph is a graph in which for any two disjoint sets of vertices of size $a+1$, there is at least one edge between the two sets.

Lemma 3.10 ([18]) *For $1 \le a \le n/2$ there exists an a-expanding graph with $O(\frac{n^2 \log n}{a})$ edges.*

Proof sketch: Assume you have a coin that has probability $p = \frac{2\ln n}{a}$ of being heads. Create a graph on n vertices as follows: for each $\{i,j\}$, flip the coin. Put the edge $\{i,j\}$ into the graph iff the coin is heads. The probability that the graph will be an a-expander graph with $O(\frac{n^2\log n}{a})$ edges is nonzero (actually close to 1). Hence such a graph exists. ∎

Lemma 3.11 ([18]) *If n elements are compared according to the edges of an a-expander graph, then there will be at most $O(a\log n)$ candidates remaining for any given rank.*

From Lemma 3.11 it is easy to prove the following:

Lemma 3.12 ([18]) *If n elements are compared according to the edges of an a-expander graph, then they can be partitioned into $O(\frac{n}{a\log n})$ sets, each containing $O(a\log n)$ elements, such that the relationship between any pair of elements is known unless they both belong to a common set.*

Theorem 3.13 ([18]) $\mathrm{sort}(k,n) = O(n^{1+1/k}(\log n)^{2-2/k})$

Proof sketch: We prove this by induction on k. For $k=1$ this is trivial. Assume the theorem for $k-1$.

Let G be an a-expanding graph that is shown to exist by Lemma 3.10. We will pick the value of a later.

ALGORITHM

Step 1: (Round 1) Compare $x_i : x_j$ iff (i,j) is an edge of G. (This takes $O(\frac{n^2\log n}{a})$ comparisons.) Let G' be the orientation of G obtained by directing i to j iff $x_i < x_j$. Let H be the transitive closure of G'.

Step 2: (Rounds $2,\ldots,k$) Using Lemma 3.12 one can show that $\{x_1,\ldots,x_n\}$ can be partitioned into $O(\frac{n}{a\log n})$ groups of size $O(a\log n)$ such that all comparisons between different groups are known. Sort the groups inductively in $k-1$ rounds. This takes

$$O(\frac{n}{a\log n}(a\log n)^{1+\frac{1}{k-1}}(\log(a\log n))^{2-\frac{2}{k-1}})$$

processors.

END OF ALGORITHM

To achieve the result set $a = \Theta(\frac{n^{1-1/k}}{(\ln n)^{1-2/k}})$. ∎

3.4 Super Expander Graphs

Alon and Azar [3] showed that $\mathrm{sort}(2,n) = O(n^{3/2}\frac{\log n}{\sqrt{\log\log n}})$. Bollobás [6] extended this to show that $\mathrm{sort}(k,n) = O(n^{1+1/k}\frac{(\log n)^{2-2/k}}{(\log\log n)^{1-1/k}})$

All these results use graphs similar to the a-expander graphs discussed in Section 3.3. We sketch the algorithm of Alon and Azar and then make some brief comments about [6].

We define a subset of a-expander graphs that has additional expanding properties. The following definition is implicit in [3].

Definition 3.14 Let $a, n \in \mathsf{N}$ and $a = \Omega(\log n)$. A graph G on n vertices is an *a-super-expander* if the following hold.

1. If A and B are disjoint subsets of vertices with a vertices each then some $v \in B$ has at least $\log_2 n$ neighbors in A.

2. Let $x \le a/e^{\sqrt{\log_2 n}}$. If A and B are disjoint sets such that $|A| = x$ and $|B| = x(\log_2 n)^{1/4}$, each $v \in A$ that has at least $\log_2 n$ neighbors in B.

The following lemma asserts that there exists small a-super-expander graphs. It is similar to Lemma 3.10; however we will be using it with a different value of a to obtain a better upper bound on $\text{sort}(k, n)$.

Lemma 3.15 ([3]) *There exists an a-super-expanding graph with $O(\frac{n^2 \log n}{a})$ edges.*

Proof sketch: Assume you have a coin that has probability $p = \Theta(\frac{\log n}{a})$ of being heads. Create a graph on n vertices as follows: for each $\{i, j\}$, flip the coin. Put the edge $\{i, j\}$ into the graph iff the coin is heads. The probability that the graph will be an a-super-expander with $O(\frac{n^2 \log n}{a})$ edges is nonzero (actually close to 1). Hence such a graph exists. ∎

Lemma 3.16 ([3]) *If n elements are compared according to the edges of an a-super-expanding graph, then there will be at most $O(a \log n / \log \log n)$ candidates remaining for any given rank.*

Theorem 3.17 ([3]) $\text{sort}(2, n) = O(n^{3/2} \frac{\log n}{\sqrt{\log \log n}})$

Proof sketch: This is similar to the $k = 2$ case of Theorem 3.13. The value of a needed is $a = \Theta(\sqrt{n \log \log n})$. ∎

Theorem 3.18 ([6]) $\text{sort}(k, n) = O(n^{1+1/k} \frac{(\log n)^{2-2/k}}{(\log \log n)^{1-1/k}})$

Proof sketch: A rather complicated type of graph is defined which will, if used to guide comparisons, yield much information. Let

$$p = \Theta(\frac{n^{1/k}(\log n)^{2-2/k}}{n(\log \log n)^{1-1/k}}).$$

Assume you have a coin that has probability p of being heads. A graph on n vertices as follows: for each $\{i, j\}$, flip the coin. Put the edge $\{i, j\}$ into the graph iff the coin is heads. The probability that the graph will be of this type and have $O(n^{1+1/k} \frac{(\log n)^{2-2/k}}{(\log \log n)^{1-1/k}})$ edges is nonzero (actually close to 1). We use this type of graph in round 1 and then proceed inductively. ∎

4 Constructive Methods

4.1 Merging and Sort

Let merge(k,n) be the number of processors needed to merge two lists of n elements in k rounds.

Haggkvist and Hell [14] present constructive proofs for the following upper bounds : merge$(k,n) = \Theta(n^{1+1/(2^k-1)})$ and csort$(k,n) = O(n^{1+2/\sqrt{2k}})$. Their sorting algorithm uses parallel merging. The paper gives matching upper and lower bounds for merging. While all that was needed was an upper bound for merging, knowing the exact bound allows us to know that the sorting algorithm cannot be improved via an improvement to the bound on merging.

Lemma 4.1 ([14]) merge$(k,n) = O(n^{\frac{2^k}{2^k-1}})$

Proof sketch:

The algorithm given for merging two ordered lists of n elements is to partition each list into groups, and then do a pairwise comparison of the first element of each group in the first list with the first element of each group in the second list. After doing these comparisons, there will be a small number of groups whose members are still unordered relative to one another. To prove this they consider the following graph: V is the set of groups, and an edge is placed between A and B if there is an $x \in A$ and a $y \in B$ such that the ordering $x : y$ is not known. They show that this graph is planar and thus linear in size.

Haggkvist and Hell establish that a group size of $O(n^{1/3})$ is optimal for two round parallel merging, giving merge$(n,2) = O(n^{4/3})$. By applying induction on the merging of the groups whose orientation was not previously determined by the comparison of the first elements of each group, they derive the generalization merge$(k,n) = \Theta(n^{\frac{2^k}{2^k-1}})$ ▮

Note 4.2 Haggkvist and Hell also showed that merge$(k,n) = \Omega(n^{\frac{2^k}{2^k-1}})$.

Theorem 4.3 ([14])

1. csort$(3,n) = O(n^{8/5})$.
2. csort$(4,n) = O(n^{20/13})$.
3. csort$(5,n) = O(n^{28/19})$.
4. csort$(k,n) = O(n^{1+2/\sqrt{2k}})$.

Proof sketch: The algorithm to sort a list of values in k rounds is based on using some number of rounds j to partition the list and sort each partition, and then use the remaining $k-j$ rounds to do a pairwise merge of those partitions. In the 3 round case, the list is partitioned into groups of size $O(n^{2/5})$ and each partition is then sorted in one round using $O(n^{6/5})$ processors per partition, or a total of $O(n^{8/5})$ processors. Then in the two remaining rounds, a pairwise merging of the $O(n^{2/5})$ groups would produce all information required to fully order the original n values.

The other results are similar. In each case the calculation of the optimal value of j is nontrivial. Let s_k denote the smallest value such that a j exists that allows one to sort n numbers in k-rounds with $O(n^{s_k})$ processors. The following recurrence allows one to find s_k for any particular k; however, it has no closed form.

$$s_{k+1} = \min\{\frac{2(2^j-1)s_{k+1-j} - 2^j}{(2^j-1)s_{k+1-j} - 1} : j > 0 \wedge s_{k+1-j} \geq \frac{2^j}{2^j-1}\}$$

From this one can derive the approximation $\text{csort}(k,n) = O(n^{1+2/\sqrt{2k}})$. The calculation is not straightforward. ▮

4.2 Attempts at the $k = 2$ case

Theorem 4.3 did not produce a constructive 2-round subquadratic sorting algorithm. This was eventually solved by Pippenger (see Section 4.3); however, before it was solved there were some interesting results that broke the $\binom{n}{2}$ barrier.

1. Haggkvist and Hell [13] showed $\text{csort}(2,n) \leq \frac{39}{45}\binom{n}{2}$. Their proof used the Peterson graph and balanced incomplete block designs.

2. Bollobás and Rosenfeld [8] showed $\text{csort}(2,n) \leq \frac{4}{5}\binom{n}{2}$. Their proof used the Erdos-Renyi graph [11] based on projective geometry.

4.3 The First Constructive Subquadratic Algorithm for $k = 2$

In 1984 the first constructive 2-round subquadratic sorting algorithm was discovered by Pippenger [19] who showed $\text{sort}(2,n) = O(n^{1.95})$. He never wrote it up; however, several references to it exist including one in [7]. A year later he improved this result and generalized to k rounds by developing the framework of expander graphs for sorting (see Section 3.3) and showing that the graphs constructed by Lubotzky, Phillips, and Sarnak [16] were a-expander graphs. These can be used to obtain sorting algorithms that are constructive, though not as good as the nonconstructive ones in Theorem 3.13.

Lemma 8 of [18] proves two things. We separate them out into two separate lemmas.

Lemma 4.4 ([18]) *Let G be a graph on n vertices. Let λ_i be the ith largest eigenvalue of the adjacency matrix. If $\lambda_1 = p+1$ and $(\forall i \geq 2)[\lambda_i \leq 2\sqrt{p}]$ then G is an $O(\frac{n}{\sqrt{p}})$-expanding graph.*

Lemma 4.5 ([16, 18]) *Let p, q be primes that are congruent to 1 mod 4. Assume $p < q$. There exists an explicitly constructed $O(\frac{q}{\sqrt{p}})$-expanding graph with $q+1$ vertices and $O(pq)$ edges.*

Proof sketch: Lubotzky, Phillips, and Sarnak [16] constructed a $p+1$-regular graph G on $q+1$ vertices with the following properties: (1) the largest eigenvalue of the adjacency matrix, $p+1$, has multiplicity 1, and (2) all other eigenvalues have magnitude at most $2\sqrt{p}$. Clearly this graph is on $q+1$ vertices and has $O(pq)$ edges. By Lemma 4.4 this graph is an $O(\frac{q}{\sqrt{p}})$-expanding graph. ▮

Note 4.6 The graphs constructed by Lubotzky, Phillips, and Sarnak are somewhat complicated. They use graphs associated to certain groups.

Lemma 4.7 *Let $1 \le a \le n$. Let $\frac{n}{a}$ be sufficiently large. There is an explicitly constructed a-expanding graph G on n vertices of size $O(n^3/a^2)$.*

Proof sketch: We need to find primes p, q such that $(\frac{n}{a})^2 \le p \le 2(\frac{n}{a})^2$, $n \le q \le 2n$, and both p, q are congruent to 1 mod 4. Such exists for $\frac{n}{a}$ large by the Prime number theorem for arithmetic progressions (see [10] for example). Apply Lemma 4.5 to obtain a graph on $\Theta(n)$ vertices that is a $O(\frac{q}{\sqrt{p}})$-expanding, hence $O(\frac{n^3}{a^2})$-expanding. ∎

Theorem 4.8 $\mathrm{csort}(k,n) \le O(n^{1+\frac{2}{(k+1)}}(\log n)^{2-\frac{4}{(k+1)}})$.

Proof sketch: This proof is similar to that of Theorem 3.13 except that we use Lemma 4.7 with

$$a = \Theta(\frac{n^{1-1/(2^k-1)}}{(\log n)^{2/(2^k-1)}}).$$

∎

4.4 Two Simple Constructive Algorithm

Alon [2] showed that $\mathrm{csort}(2,n,2) = O(n^{7/4})$. His algorithm is simpler than that of Theorem 4.8. Since Alon's result is about limited closure sorting we will discuss it in Section 7; however by combining it with the recurrence in Theorem 4.3 he obtained improvements over Theorem 4.3. We give the first few improvements. More numbers can be generated; however, the asymptotic values do not improve.

Theorem 4.9 ([2])

1. $\mathrm{csort}(2,n) = O(n^{7/4})$.
2. $\mathrm{csort}(3,n) = O(n^{8/5})$.
3. $\mathrm{csort}(4,n) = O(n^{26/17})$.
4. $\mathrm{csort}(5,n) = O(n^{22/15})$.

Pippenger [18] noticed that a variant of Alon's algorithm actually yields $\mathrm{csort}(2,n) \le O(n^{5/3}\log n)$. (We will discuss this algorithm when discussing Alon's algorithm.) Golub [12] noticed that this could be combined with an easy modification of the recurrence in Theorem 4.3 to obtain a *simple* constructive algorithm which is better than that of Theorem 4.9. We give the first few improvements. More numbers can be generated; however, the asymptotic values do not improve.

Theorem 4.10

1. $\mathrm{csort}(2,n) = O(n^{5/3}\log n)$ *(use Pippenger's modification of Alon).*
2. $\mathrm{csort}(3,n) = O(n^{8/5})$ *(use Theorem 4.9 or Theorem 4.3).*
3. $\mathrm{csort}(4,n) = O(n^{3/2})$.
4. $\mathrm{csort}(5,n) = O(n^{23/16})$.

4.5 A Constructive Algorithm via Pseudo-Random Generators

Wigderson and Zuckerman [23] present a constructive proof that $\mathrm{sort}(n,k) = O(n^{1+1/k+o(1)})$. Their algorithm is based upon Pippenger's non-constructive sorting algorithm (See Section 3.3). Recall that Pippenger showed that small a-expander graphs were useful for sorting, and then showed that small a-expander graphs exist. The value of a taken for k round sorting was $a = \frac{n^{1-\frac{1}{k}}}{(\ln n)^{1-\frac{2}{k}}}$. Wigderson and Zuckerman present a constructive proof of the existence of a a-expander graphs with slightly worse values of a.

Definition 4.11 [24] A distribution D on $\{0,1\}^n$ is a δ-source if for all $x \in \{0,1\}^n$, $D(x) \le 2^{-\delta n}$.

We intend D to be an approximation to the uniform distribution. If $\delta = 1$ then D is uniform. The smaller δ is the worse an approximation D is.

Definition 4.12 [17] Let $n, m, t \in \mathsf{N}$ and $0 < \epsilon, \delta < 1$. A function $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is an (n,m,t,δ,ϵ)-*extractor* if for every δ-source D, the distribution of $E(x,y)y$ induced by choosing x from D and y uniformly in $\{0,1\}^t$ is within statistical distance ϵ of the uniform distribution.

We think of $t << n$ and $m = \Omega(n)$. Hence we think of having a weak random source D for strings in $\{0,1\}n$ and a uniform random source for $\{0,1\}^t$, and producing from these a (stronger) random source.

We first state that there are P-time, $DSPACE(O(n))$, computable extractors, and then state that such can be used to construct n^δ-expanding graphs for an appropriate δ.

Lemma 4.13 *[17] Let δ, ϵ be functions of n such that $1/n \le \delta \le 1/2$ and $2^{-\delta n} \le \epsilon \le 1/n$. Let $t = O(\frac{\log \epsilon^{-1} \log^2 n \log \delta^{-1}}{\delta})$. There exists polynomial-time linear-space computable (n,m,t,δ,ϵ)-extractors.*

Lemma 4.14 *[23] If there is an $(n,m,t,\delta,1/4)$-extractor computable in linear space then there is an N^δ-expanding graph on $N = 2^n$ nodes with maximum degree $N2^{1+2t-m}$ constructible in* DSPACE$(\log n)$*, hence in P.*

Proof sketch: Let $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ be the extractor. First define a bipartite graph $H = (V, W, E)$ by $V = \{0,1\}^n$, $W = \{0,1\}^m$, and $(x,z) \in E$ iff $(\exists y)[E(x,y) = z]$. We obtain H' from H by removing all vertices in W that have degree larger than the average degree. Now form a graph G as follows: the vertex set is V and we connect x to x' if there exists z such that both (x,z) and (z,x) are edges in H'. Using the properties of extractors one can show that G is an N^δ-expander. ∎

By combining Lemma 4.13, 4.14, and the techniques of Theorem 3.13 one can obtain the following.

Theorem 4.15 ([23]) $\mathrm{csort}(k,n) \le O(n^{1+1/k+o(1)})$.

Note 4.16 Most of the algorithms discussed in this paper only work if n is "large." The algorithm in Theorem 4.15 needs much larger n than usual.

5 A Randomized Algorithm

Alon, Azar, and Vishkin showed $\mathrm{rsort}(k,n) = O(n^{1+1/k})$. Their algorithm is fairly simple; however, the analysis requires care.

Theorem 5.1 ([4]) $\mathrm{rsort}(k,n) = O(n^{1+1/k})$.

Proof sketch:

In the first round of this algorithm, $n^{1/k} - 1$ values are chosen at random and then compared to all $n-1$ other values requiring processors. Between rounds, the n values are partitioned into $O(n^{1/k})$ blocks $(A_1, \ldots, A_{n^{1/k}})$ based on the now ordered list of $O(n^{1/k})$ values such that if $i < j$ then all members of A_i are less than members of A_j.

In the remaining $k-1$ rounds, each A_i is sorted. A careful analysis shows that the expected number of processors required to do this will be $O(n^{1+1/k})$. ∎

6 A Nonconstructive Algorithm for $\mathrm{sort}(2,n,d)$

Bollobás and Thomason [9] were the first ones to look at sorting with limited transitive closure. They used nonconstructive means, similar (though more complicated) to those we have seen in Theorems 3.7, 3.4, 3.13, 3.17 and 3.18. Hence we omit even a sketch here.

Theorem 6.1 ([9]) $\mathrm{sort}(2,n,d) \le O(\frac{1}{2d} n^{1+\frac{d}{2d-1}} (\log n)^{1/2d-1})$.

7 A Constructive Algorithm for $\mathrm{sort}(2,n,2)$

Alon [2] showed that $\mathrm{csort}(2,n,2) = O(n^{7/4})$. He used techniques in projective geometry over finite fields to construct graphs. He used the eigenvalue methods of [21] to prove his graphs had the relevant properties. Pippenger used a variation of Alon's graphs to obtain $\mathrm{csort}(2,n) = O(n^{5/3}\log n)$.

Notation 7.1 If q is a prime power than F_q is the finite field on q elements.

Definition 7.2 [2] Let $d, q \in \mathsf{N}$ and let q be a prime power. The number d will be referred to as the *dimension*. The *Geometric Expander over F_q of dimension d* is the bipartite graph that we construct below:

1. Create a set of $d+1$ tuples of the following form

$$\begin{array}{ll}
(1, a_1, a_2, \ldots, a_{d+1}) & a_1, \ldots, a_{d+1} \in \{0,1,\ldots,q-1\} \\
(0, 1, a_2, \ldots, a_{d+1}) & a_2, \ldots, a_{d+1} \in \{0,1,\ldots,q-1\} \\
(0, 0, 1, a_3, \ldots, a_{d+1}) & a_3, \ldots, a_{d+1} \in \{0,1,\ldots,q-1\} \\
\quad\vdots & \\
(0, 0, 0, \ldots, 0, 1, a_{d+1}) & a_{d+1} \in \{0,1,\ldots,q-1\}
\end{array}$$

   Note that each tuple represents a hyperplane in $d+1$ space over F_q. (Alternatively we could have allowed all tuples that were not $(0,\ldots,0)$ and then identify any two that differ by a constant multiple in F_q.)

2. Let U and V be the set of tuples above. An edge will exist between $u \in U$ and $v \in V$ iff $u \cdot v = 0$ in the field F_q. (This is equivalent to saying that the planes which represent u and v are orthogonal to one another.)

Note 7.3 Note that the number of vertices in the graph in Definition 7.2 is $\Theta(q^{d+1})$ and the number of edges is $\Theta(q^{2d+1})$. If we denote the number of vertices by n then the number of edges is $\Theta(n^{2-\frac{1}{d}})$.

The following definition is implicit in [2].

Definition 7.4 An $(\alpha n^a, \beta n^b, \chi n^c, \delta n^d)$-expander is a bipartite graph $G = (U, V, E)$ such that $|U| = |V| = n$ and the following two properties hold.

1. $(\forall Z \subseteq V)[|Z| \geq \alpha n^a \Rightarrow |\{x \in U : |N(x) \cap Z| \leq \beta n^b\}| \leq \chi n^c]$
2. $(\forall Y \subseteq V)[|Y| \geq \beta n^b \Rightarrow |N(Y) \geq n - \delta n^d]$

Alon proved the following theorem using the eigenvalues methods of [21].

Lemma 7.5 ([2]) *Let $G = (U, V, E)$ be the Geometric expander of dimension 4 over F_q. Let n be the number of vertices in U (also V). Then G is an $(3n^{3/4}, n^{1/2}, n^{1/2}, n^{3/4})$-expanding graph with $\Theta(n^{7/4})$ edges.*

The following lemma is implicitly in [2].

Lemma 7.6 *If there exist $(\alpha n^a, \beta n^b, \chi n^c, \delta n^d)$-expanders of size $O(n^e)$ then we can sort in 2 rounds using only 2-step transitive closure in $O(n^{\max\{e,d+1,c+2-a,a+1\}})$ processors.*

Proof sketch: Assume that we have a set of n values that we want to order and an $(\alpha n^a, \beta n^b, \chi n^c, \delta n^d)$-expanding graph of size $O(n^e)$.

1. (Round 1) Do the comparisons as designated by the graph. This requires $O(n^e)$ processors. Take the 1-step transitive closure.
2. (Round 2) For all x, y such that we do not know how they compare, compare $x : y$.

We need to know how many comparisons remain for Round 2. Since we know that there exists an ordering of the n values, we can imagine having the vertices correctly ordered. Note that the graph connecting these vertices is still an $(\alpha n^a, \beta n^b, \chi n^c, \delta n^d)$-expander. We can then divide the vertices into $1/\alpha$ blocks $(A_1, \ldots, A_{n^{1-a}/\alpha})$ each of αn^a vertices. By using the properties of being an $(\alpha n^a, \beta n^b, \chi n^c, \delta n^d)$ expander one can show that there are at most $O(n^{\max\{d+1,n+2-a\}})$ comparisons left between elements in the same block, and at most $O(n^{a+1})$ comparisons left between elements in different blocks. The result follows. ∎

Theorem 7.7 ([2]) $\mathrm{sort}(2, n, 2) = O(n^{7/4})$.

Proof: This follows from Lemma 7.5 and Lemma 7.6. ∎

We now discuss Pippenger's variant on Alon's algorithm. As stated above Alon used eigenvalue methods. In particular he showed the following.

Lemma 7.8 ([2]) *Let $H = (V, V, E)$ be a geometric expander of degree d over a field of q elements. Let $G = (V, E')$ be the graph where $(x, y) \in E'$ iff $(x, y) \in E$ and $x \neq y$. Then G has $\Theta(q^d)$ vertices and $O(q^{2-1/d})$ edges. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of the matrix for G in decreasing order. Then $\lambda_1 = \Theta(q^{2d-2})$ and $\lambda_2 = \cdots = \lambda_n = O(q^{d-1})$. The constants work out so that if $d = 3$ then $\lambda_2 \leq 2\sqrt{\lambda_1}$.*

The following lemma is implicit in [18]. It follows from Lemmas 4.4 and 7.8.

Lemma 7.9 *Let $d = 3$. Let G be as in Lemma 7.8. Let n be the number of vertices in G. Then G is an $O(n^{1/3})$-expander with $O(n^{5/3})$ edges.*

From Lemmas 7.9 and 3.12 one can easily prove Theorem 4.10.

7.1 Using Merging for $\mathrm{sort}(k, n, 2)$ with k odd

Bollobás and Thomason [9] show that $\mathrm{csort}(n, k, 1) = O(n^{\frac{3}{2}+\frac{1}{2(2^{k+1/2}-1)}})$ for k odd.

Their algorithm is similar to the approach of Haggkvist and Hell (see Section 4.1). In that algorithm you first partition the original list into sublists, recursively sort those sublists (in j rounds), and merge them back into a single ordered list (in $k-j$) rounds, where j is picked cleverly. By contrast the algorithm of Bollobás and Thomason uses $k - 2$ rounds to recursively sort the sublists and then only 2 rounds to accomplish the merging of the sublists. These last 2 rounds are done in a clever way. This uses more processors than Haggkvist-Hell; however, rather than computing the full transitive closure of the relationships learned, only 2-step transitive closure.

Theorem 7.10 ([9]) *For k odd,* $\mathrm{sort}(n, k, 1) = O(n^{\frac{3}{2}+\frac{1}{2(2^{k+1/2}-1)}})$.

Proof sketch: We show this by induction. If $k = 1$ then this is trivial. Assume $k > 1$ and k is odd.

ALGORITHM

Step 1: (Preprocessing. Does not count.) Partition the n values into m sublists (each of size n/m) where $m = n^{1/2^{(k-1/2)}}$.

Step 2: (Rounds 1 to $k - 2$) Sort these sublists recursively in $k - 2$ rounds.

Step 3: (Round $k - 1$) For all sublists $X = \{x_1, x_2, \ldots, x_{n/m}\}$ and all $v \notin X$ compare v to all elements in $\{x_{\sqrt{n/m}}, x_{2\sqrt{n/m}}, \ldots, x_{n/m}\}$ simultaneously. We view each sublist as having been partitioned into sublocks (e.g. the elements between $x_{4\sqrt{n/m}}$ and $x_{5\sqrt{n/m}}$ form a sublock). At the end of this round we know, for each v, which sublock it belongs to.

Step 4: (Round k) For each v and each sublist X we know which sublock of X, v belongs. Compare v to the elements in that sublock.

END OF ALGORITHM

A straightforward analysis shows that this algorithm uses the number of processors specified. A careful look at the last two rounds shows that it only uses 2-step transitive closure. ∎

8 Lower Bounds

Haggkvist and Hell [13] showed that $\mathrm{sort}(k,n) \geq \Omega(n^{1+1/k})$. Bollobas and Thomason [9] improved the constant. Alon and Azar [3] improved this to $\mathrm{sort}(k,n) \geq \Omega(n^{1+1/k}(\log n)^{\frac{1}{k}})$ (for $k \geq 2$). Note that this is quite close to the upper bound in Section 3.4. Of more importance, this bound is larger than $\mathrm{rsort}(k,n)$ (see Section 5); hence sorting-in-rounds is a domain where randomized algorithms are provably better than deterministic.

Theorem 8.1 ([13]) $\mathrm{sort}(k,n) > \frac{n^{1+1/k}}{2^{k+1}} - n/2$.

Proof sketch: By induction on k. For $k = 1$ this is trivial. Assume true for $k-1$. Assume, by way of contradiction, that the first round uses $\leq 2^{-k}n^{1+1/k} - n/2$ processors. Let G be the graph of comparisons made.

By looking at the average degrees of vertices one can show that there is a set of $n/2$ nodes such that the the induced subgraph G' on them is s-colorable where $s = 2^{-k-2}n^{1/k}$. Color G' with s colors. Let $V_1, \ldots, V_s$ be the color classes. Orient G' as follows: For all $1 \leq i < j \leq s$, for all $v \in V_i$, for all $u \in V_j$, if (v,u) is an edge then set $v < u$. Orient G so that you use this orientation on G' and all the vertices not in G' are less than all the vertices in G'. Once the transitive closure of G is taken one still needs to sort each V_i in $k-1$ rounds (one also needs to sort those elements not in G' but this is not needed for the lower bound). One can show the lower bound by using the inductive lower bound on each V_i and some algebra. ∎

Theorem 8.2 ([9]) *For all $c < \sqrt{3/2}$, $\mathrm{sort}(k,n) \geq cn^{1+1/k}$*

Proof sketch: This proof is similar to that of Theorem 8.1 except that the $V_1, \ldots, V_s$ are obtained by a greedy coloring and more care is taken in showing the largest value of $\sum_{i=1}^{s} \mathrm{sort}(k-1, |V_i|)$. Lagrange multipliers are used. ∎

The key to the proofs of Theorems 8.1 and 8.2 is that we still need to sort each V_i. The proof does not use the fact that you might have to make some comparisons between vertices in different V_i's. To improve this lower bound Alon and Azar showed that you will have to make such comparisons.

Lemma 8.3 *Let G be a graph with n vertices and dn edges. There exists an induced subgraph on $\frac{n}{4}$ vertices such that (1) G' has degree $< 4d$, and (2) there is a $4d$-coloring of G' with color classes $V_1, \ldots, V_{4d}$ such that for all $1 \leq i, j \leq 4d$, for all $x \in V_i$, there are at most $2^{|i-j|+1}$ neighbors in V_j.*

Proof sketch: Remove successively the highest degree vertex $\frac{n}{2}$ times. Let G' be the induced subgraph on the remaining $\frac{n}{2}$ vertices. One can show that G' has degree $< 4d$. Clearly G' is $4d$-colorable. Let $U_1, \ldots, U_{4d}$ be the color classes. A probabilistic argument shows that there exists a permutation of $\{1, \ldots, 4d\}$ that satisfies the properties needed. ∎

Lemma 8.4 *Let d be such that $d = o(n)$ and $d = \Omega(\log n)$. Let G be a graph with n vertices and dn edges. There exists an orientation of G such that the complement of its transitive closure has at least $\Omega(\frac{n^2}{d}\log(\frac{n}{d}))$ edges.*

Proof sketch: Use Lemma 8.3 to obtain G' and $V_1, \ldots, V_{4d}$ as specified there. Orient G' as follows: For all $1 \leq i < j \leq m$, for all $v \in V_i$, for all $u \in V_j$, if (v,u) is an edge then set $v < u$. Orient G so that you use this orientation on G' and all the vertices not in G' are less than all the vertices in G'. The complement of the transitive closure will not contain any edges within an V_i. In addition, because of the limit on how many edges can go from an V_i to an V_j, one can estimate additional lower bounds on the number of edges in the complement of the transitive closure. (This is highly non-trivial.) ∎

Theorem 8.5 ([3]) *For $k \geq 2$, $\mathrm{sort}(k,n) \geq \Omega(n^{1+1/k}(\log n)^{1/k})$.*

Proof sketch:

We prove this by induction. Note that $k = 2$ is the base case and is non-trivial. Assume that there is an algorithm that sorts n elements in 2 rounds and takes dn processors. We know that $d = \Omega(n^{1/3})$ by Theorem 8.1. We can assume $d = o(n^{2/3})$ since if it is not then the theorem for $k = 2$ is already true. Let G be the graph representing the first round. Because of the bounds on d we can apply Lemma 8.4 to the graph to obtain an orientation such that the compliment of the transitive closure has $\Omega(\frac{n^2}{d}\log(\frac{n}{d}))$ edges. Hence the second round needs $\Omega(\frac{n^2}{d}\log(\frac{n}{d}))$ processors. Algebra shows that $d = \Omega(\sqrt{n \log n})$.

We now sketch the induction step. It will be *easier* than the $k = 2$ case since it does not use Lemma 8.4. Assume the lower bound for $k-1$ where $k \geq 3$. Assume there is an algorithm for sorting in k rounds. Let G be the graph representing the first round of the algorithm. Assume G has dn edges. By Turan's theorem (see [5] for a nice probabilistic proof) a graph with dn edges has an independent set of size $\frac{n}{2d+1}$. By repeated application of Turan's theorem we can find $s = \Omega(d)$ pairwise disjoint independent sets of size $\Omega(\frac{n}{1+d})$ which we denote $V_1, \ldots, V_s$. Let V_0 be all the other vertices. Orient G as follows: For all $1 \leq i < j \leq s$, for all $v \in V_i$, for all $u \in V_j$, if (v,u) is an edge then set $v < u$. The remaining $k-1$ rounds need to sort each V_i, $i \geq 1$. Algebra and the induction hypothesis suffice to prove the result. ∎

9 Open Problems

The next section has tables of known results, both upper and lower bounds. The tables yield many open questions about closing these gaps.

There are no lower bounds for constructive sorting except those bounds that come from general sorting. Hence another open question would be to either obtain lower bounds for $\mathrm{csort}(k,n)$ that use the fact that the algorithm is constructive, or show that any sorting algorithm can be turned into a constructive one.

Another open problem is to obtain simpler proofs of the known upper bounds, especially the constructive ones.

10 Tables of Results

In this section we put all the known results into tables. We leave out the big-O's and big-Ω's unless there is an interesting point to be made about the constants.

10.1 Nonconstructive Methods for Sorting

k	$k=2$	$k=3$	ref
$n^{(3\cdot 2^{k-1}-1)/(2^k-1)}\log n$	$n^{5/3}\log n$	$n^{11/3}\log n$	[14]
	$n^{3/2}\log n$		[9]
$n^{1+1/k}(\log n)^{2-2/k}$	$n^{3/2}\log n$	$n^{4/3}(\log n)^{4/3}$	[18]
	$n^{3/2}\frac{\log n}{\sqrt{\log\log n}}$		[3]
$n^{1+1/k}\frac{(\log n)^{2-2/k}}{(\log\log n)^{1-1/k}}$	$n^{3/2}\frac{\log n}{\sqrt{\log\log n}}$	$n^{4/3}\frac{(\log n)^{4/3}}{(\log\log n)^{2/3}}$	[6]

1. All of the above results use the Probabilistic method.
2. In [14] a graph is picked at random from the set of all graphs with n vertices and n^{α_k} edges where $\alpha_k = \frac{3\cdot 2^{k-1}-1)}{(2^k-1)}$.
3. In all of the other algorithms a graph was picked by assigning to each edge a probability.

10.2 Constructive Methods for Sorting

k	$k=2$	$k=4$	ref	Math Used
	$\frac{39}{45}\binom{n}{2}$		[13]	Peterson graph and block designs
	$\frac{4}{5}\binom{n}{2}$		[8]	Erdos-Renyi graph and proj. geom.
$n^{1+(2/\sqrt{2k})}$	n^2	$n^{20/13}$	[14]	Merging and Graph Theory
$n^{1+(2/\sqrt{2k})}$	$n^{7/4}$	$n^{26/17}$	[2]	Proj. geom. over finite fields
$n^{1+(2/\sqrt{2k})}$	$n^{7/4}$	$n^{3/2}$	[18, 12]	Proj. geom. over finite fields
$n^{1+2/(k+1)}(\log n)^{2-4/(k+1)}$	$n^{5/3}(\log n)^{2/3}$	$n^{7/5}(\log n)^{6/5}$	[18]	Graphs of Groups from [16]
$n^{1+1/k+o(1)}$	$n^{3/2+o(1)}$	$n^{5/4+o(1)}$	[23]	Extractors

The first two results listed for general k are both approximations for a general recurrence. For all results calculated the first algorithm seems better; however, the asymptotic seem to be the same.

10.3 Limited Closure Sorting

The only result here are for 2-round sorting.

d	$d=2$	ref	Constructive?	Math Used
$\frac{1}{2d}n^{1+\frac{d}{2d-1}}(\log n)^{1/2d-1}$	$\frac{1}{4}n^{5/3}(\log n)^{1/3}$	[9]	No	Prob. Method
	$n^{7/4}$	[2]	Yes	Proj. Geom. over finite fields

10.4 Lower Bounds

All the lower bounds use graph theory. Under "Math Used" we point out other math that was used.

Problem	Bound	ref	Math Used
sort(k,n)	$\frac{n^{1+1/k}}{2^{k+1}} - n/2$	[13]	
sort(k,n)	$(\sqrt{3/2}-\epsilon)n^{1+1/k}$	[9]	Lagrange Mult.
sort(k,n)	$n^{1+1/k}(\log n)^{\frac{1}{k}}$	[3]	Prob. Meth.
sort$(2,n,2)$	$n^{5/3}$	[9]	
sort$(2,n,d)$	$n^{1+d/2d-1}$	[9]	
Merging	$n^{\frac{2^k}{2^k-1}}$	[14]	

11 Acknowledgment

We want to thank Nick Pippenger for intellectual inspiration and helpful discussions.

References

[1] S. Akl. *Parallel computation: models and methods.* Prentice Hall, 1997.

[2] N. Alon. Eigenvalues, geometric expanders, sorting in rounds, and Ramsey theory. *Combinatorica*, 6, 1986.

[3] N. Alon and Y. Azar. Sorting, approximate sorting, and searching in rounds. *SIAM Journal on Discrete Mathematics*, 1:269–280, 1988.

[4] N. Alon, Y. Azar, and U. Vishkin. Tight bounds for parallel comparison sorting. In *Proc. of the 29th IEEE Sym. on Found. of Comp. Sci.*, pages 502–510, 1988.

[5] N. Alon and J. Spencer. *The Probabilistic Method.* Wiley, 1992.

[6] B. Bollobás. Sorting in rounds. *Discrete Mathematics*, 72, 1988.

[7] B. Bollobás and P. Hell. Sorting and graphs. In I. Rival, editor, *Graphs and Orders*, pages 169–184. D. Reidel Publishing Company, 1985.

[8] B. Bollobás and M. Rosenfeld. Sorting in one round. *Israel Journal of Mathematics*, 38, 1981.

[9] B. Bollobás and A. Thomason. Parallel sorting. *Discrete Applied Mathematics*, 6, 1983.

[10] H. Davenport. *Multiplicative number theory.* Springer, 1980.

[11] P. Erdos and A. Renyi. On a problem in the theory of graphs (in Hungarian). *Publ. Math. Inst. Hungar. Acad. Sci.*, 7, 1962.

[12] E. Golub. *Empirical Studies in Parallel Sorting,1999.* PhD thesis, University of Maryland,College Park. `http://www.cs.umd.edu/~egolub/dissert.pdf`.

[13] R. Haggkvist and P. Hell. Parallel sorting with constant time for comparisons. *SIAM J. Comput.*, 10(3):465–472, 1981.

[14] R. Haggkvist and P. Hell. Sorting and merging in rounds. *SIAM Journal on Algebraic and Discrete Methods*, 3, 1982.

[15] F. T. Leighton. *Introduction to parallel algorithms and architectures: arrays, trees, and hypercubes.* Morgan Kaufman, 1992.

[16] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 1988.

[17] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and Systems Sciences*, 52(1):43–52, Feb. 1996. Earlier version in FOCS93. The title in FOCS93 was *More deterministic simultations in Logspace.*

[18] N. Pippenger. Sorting and selecting in rounds. *SIAM J. Comput.*, 16:1032–1038, 1987.

[19] N. Pippenger. Peronal communication, 2000.

[20] J. Spencer. *Ten Lectures on the Probabilistic Method.* Conf. Board of the Math. Sciences, Regional Conf. Series, AMS and MAA, 1987.

[21] M. Tanner. Explicit construction of concentrators from generalized n-gons. *SIAM Journal on Algebraic and Discrete Methods*, 5, 1984. Despite the title the main result relates eigenvalues to expanders.

[22] L. G. Valiant. Parallelism in comparison problems. *SIAM J. Comput.*, 4(3):348–355, Sept. 1975.

[23] A. Wigderson and D. Zuckerman. Expanders that beat the eigenvalue bound: explicit construction and applications. *Combinatorica*, 19:125–138, 1999. Earlier version appeared in FOCS93.

[24] D. Zuckerman. General weak random sources. In *Proc. of the 31st IEEE Sym. on Found. of Comp. Sci.*, pages 534–543, 1990.