# A LOGARITHMIC TIME SORT FOR LINEAR SIZE NETWORKS

John H. Reif* and Leslie G. Valiant

Aiken Computation Laboratory
Division of Applied Sciences
Harvard University, Cambridge, Massachusetts

ABSTRACT. We give a randomized algorithm that sorts on an $N$ node network with constant valence in $O(\log N)$ time. More particularly the algorithm sorts $N$ items on an $N$ node cube-connected cycles graph and for some constant $k$ for all large enough $\alpha$ it terminates within $k\alpha \log N$ time with probability at least $1 - N^{-\alpha}$.

## 1. Introduction

This paper is concerned with the problem of sorting $N$ items in parallel on a fixed-connection graph $G$ having $N$ nodes labeled $\{0,1,\ldots,N-1\}$ and constant valence. Each node initially contains one key. The set $X$ of all $N$ keys is assumed to have a total ordering $<$. The network sorts by routing each key $x \in X$ to node $j = \text{rank}(x)$ where $\text{rank}(x)$ is defined as $|\{x' \in X | x' < x\}|$. This can be viewed as a distributed packet routing problem. Each $x \in X$ is considered to be an atomic packet that has to be routed from its initial node to the node corresponding to its rank. Both the rank computation and the packet routing have to be realized in a completely distributed manner.

We assume that each node contains a single sequential processor with local storage for $O(\log N)$ packets. The processors are regarded as synchronous for the purpose of step counting, but the algorithm itself does not require it. In a unit time interval a processor may transmit one of its packets along a departing edge and perform some elementary operation such as a comparison. The processors are capable of generating random bits of information and hence running randomized algorithms in the sense of Rabin [10] and Solovay and Strassen [12].

*This work was supported by the National Science Foundation Grant NSF-MCS79-21024 and the Office of Naval Research Contract N00014-80-C-0674.



Clearly the routing required to sort may require time at least the diameter of the graph. If $G$ has constant valence then the diameter is at least $\Omega(\log N)$. Hence the $O(\log N)$ time bound for our algorithm is asymptotically optimal. In this paper we restrict ourselves to demonstrating that this bound is achievable in principle and do not pursue the issue of the magnitude of the constant multipliers. We note, however, that it is within a large class of algorithms that is experimentally testable in the sense of [14].

The main components of the algorithm are the splitter directed routing procedure SDR and the splitter finding procedure SF which itself uses SDR. They are described and analyzed in Sections 5 and 7, respectively.

A summary of the algorithm for sorting on the $n$-dimensional cube-connected cycles network (CCC) of Preparata and Vuillemin [9] is as follows. Note that the number of nodes is $N = n2^n$ and hence $n < \log N$. (Logarithms are assumed to have base 2 throughout this paper.)

Step A: Call $SF(\lambda)$. This finds a set of $2^n/n^6$ elements called "splitters" that divide $X$, when regarded as an ordered set, into roughly equal intervals.

Step B: Route each packet to a random node and call $SDR(\lambda)$ with the splitters found in Step A. This will route the keys belonging to each interval to the $6 \log n$ dimensional subcube corresponding to it. In this way an approximate sort is achieved, but the keys are not spread completely uniformly around the network.

Step C: Compute the rank of each key.

Step D: Route each packet to the node corresponding to its rank.

The $O(\log N)$ behavior of each of the four steps A-D will be established respectively as follows: Theorem A (Section 7), Theorem B (Section 5), Algorithm C (Section 6) and Theorem D (Section 3). We note that Theorem B is invoked in Step B with $n - \ell = 6 \log n$, which is sufficient for the $O(\log N)$ bound. The following then follows immediately.

Main Theorem. *There is a randomized algorithm that for some $k$ and all $n$ and all sufficiently large*

$\alpha$ *sorts on an* n*-dimensional CCC network, and terminates within* $k\alpha n$ *steps with probability greater than* $1-2^{-\alpha n}$.

Previous algorithms for sorting N keys on constant valence fixed connection networks of N processors require time $\Omega(\log N)^2$. The bitonic sorter of Batcher [3] achieves this bound on such networks as the CCC [9].

For less realistic models of parallel computation faster algorithms have been known. For example, J. Wiedermann observed several years ago that the Quicksort of Hoare [6] takes time $O(\log N)$ with high likelihood on a parallel decision tree model. Reischuk [11] has a related result for a parallel random access model.

Our current algorithm follows the randomized routing ideas introduced in [14]. It can be viewed as a partially successful attempt at reducing the sorting problem to the apparently simpler problem of routing. In the analysis the critical path technique developed by Aleliunas [1] and Upfal [13] for analyzing routing in constant valence graphs plays an important part.

## 2. NETWORK DEFINITIONS

We define various constant valence networks derived from the n-dimensional binary hypercube. Consider some fixed $n \geq 1$. Let the *node set* be

$$V = \{(w,i) \mid w \in \{0,1\}^n,\ i \in \{0,\ldots,n-1\}\}$$

which has cardinality $N = n2^n$. For each $a \in V$ let $\underline{address}(a) = w$ and $\underline{stage}(a) = i$ if $a = (w,i)$. Let $w[i]$ be the i-th bit of $w$. Let $w' = EXT(w,i)$ be identical to $w$ except that $w'[i] \neq w[i]$. Also let $\tilde{w}$ be the integer of which $w$ is the binary representation.

We call an edge from node $a$ to node $b$ *internal* if $address(a) = address(b)$ and *external* if $address(b) = EXT(address(a), stage(a) + 1 \bmod n)$. It is forward if $stage(b) = stage(a) + 1 \bmod n$, *static* if $stage(b) = stage(a)$, and *reverse* if $stage(b) = stage(a) - 1 \bmod n$. The $\underline{CCC_n}$ network of Preparata and Vuillemin [9] has node set $V$ and exactly all forward internal edges, reverse internal edges and static external edges. For each of description this paper will assume a network more similar to that of Upfal [13] which we call $\underline{CCC_n^+}$. It contains node set $V$ and all forward and reverse internal edges and all forward and reverse external edges. Clearly any algorithm for $CCC_n^+$ can be simulated on $CCC_n$ with at most a factor of two time increase. Finally, we define $\underline{CCC_n^*}$ to be the network obtained by taking a $CCC_n^+$ and removing all edges that join pairs of nodes with respective stages 0 and n-1. The significance of $CCC_m^*$ is that numerous copies of it can be found in $CCC_n^*$ if $n > m$. In particular, for any $w_1$, $w_2$ such that $|w_1| + |w_2| = n-m$ the subgraph of $CCC_n^*$ spanned by the nodes $\{(w_1 w w_2, i) \mid w \in \{0,1\}^m$ and $|w_1| \leq i < |w_1| + m\}$ is isomorphic to $CCC_n^*$.

Note that $CCC_n$, $CCC_n^+$ and $CCC_n^*$ are all naturally related to the n-dimensional hypercube $H_n$. Intuitively, for each $w \in \{0,1\}^n$ the set of nodes $\{a \in V \mid address(a) = w\}$ can be considered to be a "supernode" of $H_n$. Each such supernode of $H_n$ is connected by external edges to n other supernodes $\{b \in V \mid address(b) = EXT(w,i)$ for $i=0,1,\ldots, n-1\}$.

For any m let $\{0,1\}^{<m>}$ be the set of binary strings of length not more than m-1. We define a subdivision of the node set $V$ that indexes the subsets by binary strings from $\{0,1\}^{<n+1>}$. For each $w \in \{0,1\}^n$ let $V[w] = \{b \in V \mid address(b) = w$ and $stage(b) = 0\}$. For each $w \in \{0,1\}^{<n>}$ let $V[w] = \{b \in V \mid w$ is a prefix of $address(b)$ and $|w| = stage(b)\}$. Thus $V[\lambda]$ is the set of nodes of stage zero where $\lambda$ is the empty string. Let *root* $v[w]$ of $V[w]$ be the node with address $w0^{n-|w|}$ and stage $|w|$. Note that for $|w| \leq n-1$, $v[w]$ has a departing forward internal edge entering $v[w0]$ and a departing forward external edge entering $v[w1]$.

## 3. PACKET ROUTING ON THE $CCC_n^+$

This section briefly describes the probabilistic packet routing algorithm of Valiant and Brebner [15] as applied to the $CCC_n^+$ by Upfal [13].

We require that each node $a \in V$ contain for each departing edge $e$ a queue $Q_e$ for the packets that are to be transmitted across edge $e$. Each node also contains its address and stage posted as local variables.

Let $X$ be the set of $cN$ packets to be routed, where each packet $x \in X$ is initially at a given node $I_x \in V$ and we wish $x$ to be routed to given destination node $D_x \in V$. The algorithm has two phases:

A. (Random Routing) Route $x$ from $I_x$ to a node $R_x \in V$ with random address.

B. (Fixed Destination Routing) Route $x$ from $R_x$ to $D_x$.

The random routing of $x$ in Phase A is accomplished by repeating for n stages the transmission of $x$ across a randomly chosen departing forward edge (i.e., transmit $x$ across the forward internal edge or forward external edge with equal probability). Phase B repeats for n stages the following: if $x$ is currently at node $a \neq D_x$ with $j = stage(a) + 1$ and $address(a)[j] = address(D_x)[j]$, then $x$ is transmitted across the forward internal edge departing $v$ and otherwise $x$ is transmitted across the forward external edge departing $v$. This takes the packets to nodes with the correct addresses. Finally, the packets are pipelined to the nodes with correct stage by traversing internal edges.

We have not yet specified the management of the queues of packets at each node. Suppose the *priority* of packet $x \in X$ is assigned to be the number of stages of phases A and B so far accomplished, and we allow packet $x$ to be transmitted from each node $a \in V$ only after all packets of lower priority have been transmitted from $a$. Let $T_A$, $T_B$ be the total execution times of phases A

and B respectively. The techniques of Aleliunas [1] and Upfal [13] show the following under assumption that there are initially c packets at each node.

Theorem D. *For some* $c \geq 1$ *for all sufficiently large* $\alpha$

$$\text{Prob}(T_A > c\alpha n) < N^{-\alpha}, \text{ and } \text{Prob}(T_B > c\alpha n) < N^{-\alpha}.$$

We note that since the first phase sends packets to random addresses the probability that, at its completion, there are more than $c_1\alpha n$ packets at any one node or $c_2\alpha n$ packets at any address, can be similarly bounded by $N^{-\alpha}$ (for suitable constants $c_1$ and $c_2$).

## 4. SOME COMBINATORIAL IDENTITIES

We shall use the following inequalities. Let exp denote exponentiation of Euler's constant e.

Fact 4.1. For all $x > 0$ $(1+x^{-1})^x < e$ and $(1-x^{-1})^x < e^{-1}$. Since for all large enough x $(1+x^{-1})^x(1+(4x)^{-1}) < e < (1+x^{-1})^x(1+(2x)^{-1})$, it follows that $(1-x^{-1})^{-x} < e(1+4(5(x-1))^{-1})$ and $(1+x^{-1})^{-x} < e^{-1}(1+(2x)^{-1})$.

Let $B(m,N,p)$ be the probability that in N independent Bernoulli trials with probability p of success there are at least m successes.

Fact 4.2. (Chernoff [4])

$$B(m,N,p) \leq \left(\frac{Np}{m}\right)^m \left(\frac{N-Np}{N-m}\right)^{N-m} \leq \exp(-m-Np) \text{ if } m > Npe^2.$$

Fact 4.3. ([2]) If $m = Np(1+\beta)$ where $0 \leq \beta \leq 1$ then

$$B(m,N,p) \leq \exp(-\beta^2 Np/2) .$$

Fact 4.4. (Hoeffding [7]) If we have N independent Poisson trials with respective probabilities $p_1,\ldots,p_N$ where $\Sigma p_i = Np$ and if $m \geq Np+1$ is an integer then the probability of at least m successes is at most $B(m,N,p)$.

Fact 4.5. ([5], p. 18) if $n = o(N^{2/3})$ then

$$\binom{N}{n} = (1+o(1))\frac{N^n}{n!}\exp(-n^2/2N) .$$

Fact 4.6. Suppose $x < a$, $x < X < A$ are all functions of n such that $Xx = o(A)$ and $X = o(A^{2/3})$. Let $x = aP + G$, $X = AP + G$ where $P = (X+x)/(A+a)$, $G = o(aP)$ and $G = o(AP)$. Then for all large enough n

$$\binom{a}{x}\binom{A}{X}\Big/\binom{a+A}{x+X} \leq (1+o(1))\exp(-G^2/5aP) .$$

Proof. Applying Fact 4.5 gives

$$\binom{A}{X} = (1+o(1))\frac{A^X}{X!}\exp(-X^2/2A)$$

and

$$\binom{A+a}{X+x} \geq (1+o(1))\frac{(A+a)^{X+x}}{(X+x)!}\exp(-(X^2+2xX+x^2)/2A).$$

Using $Xx = o(A)$ and applying Stirling's formula to $X!$, $(X+x)!$ and $x!$ gives

$$\binom{a}{x}\binom{A}{X}\Big/\binom{a+A}{x+X} < \left(\frac{a}{x}\right)^x\left(\frac{A}{X}\right)^X\left(\frac{X+x}{A+a}\right)^{X+x}(1+o(1)) .$$

Substituting $x = aP+G$ and $X = AP-G$ (or $x = aP-G$ and $X = AP+G$) and using Fact 4.1 gives the claimed bound. □

We shall denote by $\omega(1)$ any function that tends to infinity as $n \to \infty$. We shall assume that ratios take integral values whenever this is convenient and otherwise insubstantial.

## 5. SPLITTER DIRECTED ROUTING

Let X be a set of cN keys that are totally ordered by the relation $<$. We assume that each key $x \in X$ is initially located at a random node in $V[\lambda]$ chosen independently of any other key in $X - \{x\}$. Suppose that we are given a set of splitters $\Sigma \subseteq X$ of size $|\Sigma| = 2^\ell - 1$. We index each splitter $\sigma[w]$ EC by a distinct binary string $w \in \{0,1\}^{<\ell>}$ of length less than $\ell$. Let $<\bullet$ denote the ordering defined as follows: For all $w,u,v \in \{0,1\}^{<\ell>}$ $w0u <\bullet w <\bullet w1v$. We require that for all $w_1,w_2 \in \{0,1\}^{<\ell>}$ $\sigma[w_1] < \sigma[w_2]$ iff $w_1 <\bullet w_2$. We assume that a copy of each splitter $\sigma[w]$ is already available in each node of $V[w]$.

Let $X[\lambda] = X$ where $\lambda$ is the empty string. Initially we assume that the keys of $X[\lambda]$ are located at $V[\lambda]$, that is the nodes of V having stage zero. The splitter directed routing is executed in $\ell$ temporally overlapping stages $i = 0,1,\ldots,\ell-1$. For each $w \in \{0,1\}^i$ the set of keys $X[w]$ are all eventually routed through $V[w]$. The splitter $\sigma[w]$ partitions $X[w] - \sigma[w]$ into disjoint subsets

$$X[w0] = \{x \in X[w] \mid x < \sigma[w]\}$$

and

$$X[w1] = \{x \in X[w] \mid \sigma[w] < x\}$$

which are subsequently routed through $V[w0]$ and $V[w1]$ respectively.

Suppose that a key $x \in X[w]$ is located at a node $a \in V[w]$ with address $ww'$ and stage i. Let B be the first bit of the address suffix $w'$. Then x is transmitted from node a across the departing forward internal edge if $B \equiv (\sigma[w] < x)$, and x is transmitted across the departing forward external edge otherwise. Thus if $x < \sigma[w]$ then x is transmitted to a node with address prefix $w0$, and if $\sigma[w] < x$ then x is transmitted to a node with prefix $w1$.

Note that at any one time distinct keys may be at distinct stages. When all the keys have completed stage R-1 the keys x-c are partitioned into $2^\ell$ disjoint subsets of the form $X[w]$ where $w \in \{0,1\}^\ell$, and the keys $X[w]$ are then at addresses prefixed by w. The sets $X[w]$ are thus uniquely defined by the choice of $\Sigma$. The following follows

directly from the assumption that $\sigma[w_1] < \sigma[w_2]$ if $w_1 <^{\bullet} w_2$:

Lemma 5.1. *For any* $w_1, w_2 \in \{0,1\}^{\ell}$ *if* $w_1 <^{\bullet} w_2$ *then* $x_1 < x_2$ *for all* $x_1 \in X[w_1]$ *and* $x_2 \in X[w_2]$.

Also, since each packet is assumed initially to be at a random node and since the above described splitter directed routing (SDR) procedure does not modify the last $n-\ell$ bits in the address of a packet, we can deduce that:

Lemma 5.2. *For each* $w \in \{0,1\}^{\ell}$ *and each* $x \in X[w]$ SDR *takes* $x$ *to a random node in* $V[w]$ *chosen independently of any other packet.*

The lemma above can be used to speed up the overall algorithm by avoiding repeated randomization. We shall not invoke it, however, as it does not change the asymptotic runtime.

The SDR procedure can be viewed as a generalization of Phase B of the routing procedure described in Section 3. It routes packets from random source nodes to specified destinations such that the number of packets destined for each region is about the same. The analysis used in the proof is an extension of the techniques introduced by Aleliunas [1] and Upfal [13] for establishing good bounds for such constant degree graphs as the CCC and d-way shuffle.

Theorem B. *Suppose we have a network* $CCC_n^*$ *with a set* $X$ *of* $cn2^n$ *packets and a set* $\Sigma$ *of* $2^{\ell}-1$ *splitters where* $n \geq \ell \geq n/2$ *such that for all* $w \in \{0,1\}^{\ell}$ $|X[w]| \leq 2cn2^{n-\ell}$. *Suppose that all the remaining packets are at independently chosen random nodes of* $V[\lambda]$. *If* $T$ *is the total time for execution of* SDR *then for some* $c_2, k > 0$, *for all* $n$, $c \geq 1$ *and all sufficiently large* $\alpha$

$$\text{Prob}(T > c_2 c\alpha n) < 2^{-c\alpha n} + \exp(-k \cdot 2^{n-\ell}) \cdot 2^{2\alpha n}$$

Proof. First we observe that since the packets are randomly distributed initially, the probability that some $a \in V[\lambda]$ initially contains more than $c(\alpha+1)n$ keys is less than $2^{-c(\alpha+1)n}$ if $\alpha > e^2$. This follows immediately from Fact 4.2.

Let $\beta = \alpha + 1$. To each packet we assign a random integer from the set $1,\ldots,\beta n$ as its *priority*. Each packet has probability $(\beta n)^{-1}$ independently of being assigned any particular such number. In SDR we will insist that no key be forwarded from a node before all keys of higher priority that ever visit it have been forwarded. [In practice we simply forward the packets currently at any node in order of their priority. This will be at least as fast, clearly, as the hypothesized algorithm that prophesies about future arrivals.]

For each node $a$ and priority $\pi \in \{1,\ldots,\beta n\}$ let *task* $\tau = (a,\pi)$ be the job of forwarding all keys of priority $\pi$ that ever visit node $a$. Let a *delay* be any pair of tasks $(\tau_1,\tau_2) = ((a,\pi),(b,\rho))$ where *either* $a = b$ *and* $\rho = \pi + 1$ *or* $(a,b)$ is an edge of the network and $\rho = \pi$. The two cases correspond to the two ways in which the execution of a task $\tau_2$ may depend on the completion of task $\tau_1$. In the first case $\tau_2$ has to wait for packets of lower priority to be processed at its node. In the second case $\tau_2$ has to wait for the arrival of a packet from an adjacent node.

Let a *delay sequence* $D$ be a sequence of delays $(\tau_0,\tau_1), (\tau_1,\tau_2),\ldots(\tau_{d-2},\tau_{d-1}),(\tau_{d-1},\tau_d)$. Note that $d \leq \ell + \beta n$ since in each delay in any such sequence either the stage of the node increases by one or the priority increases by one. Since there are just two possible forward edges of transmission and just one way of increasing the priority, the total number of delay sequences starting at any one node is at most $3^{\ell+\beta n}$. Hence their total number is at most $2^n \cdot 3^{\ell+\beta n} \leq 2^{5n+2\alpha n}$

Let $T(D)$ be the number of time units (i.e., packet transmissions) involved in $D$ (i.e., in $\tau_0,\tau_1,\ldots,\tau_d$). It remains to prove that for some $c_4$ for all $D$ and all sufficiently large $c$ and $\alpha$,

$$\text{Prob}(T(D) > c_4 c\alpha n) < 2^{-3c\alpha n - 6n}$$

for then the probability that the worst sequence suffers that much delay is at most

$$2^{-3c\alpha n-6n} \cdot 2^{2\alpha n+5n} < 2^{-\alpha n - n}$$

This is proved under the assumption that there are at most $c(\alpha+1)n$ packets initially at any node. Since, as has been observed, this event is equally unlikely the result follows.

To establish the time bound on $T(D)$ consider any particular $D$ and let $\tau_j = (a,\pi)$ where stage$(a) = i$ be a task in $D$. Let $P_j$ be the set of keys that have nonzero probability of being routed through $\tau_j$ (i.e., if their priority and initial position are suitably chosen) but would then certainly depart from $D$ at $\tau_j$. Departure from $D$ is forced *either* because $(\tau_j,\tau_{j+1}) = ((a,\pi),(a,\pi+1))$ (since the priority of a packet cannot change) *or* because $(\tau_j,\tau_{j+1}) = ((a,\pi),(b,\pi+1))$ but $(a,b)$ is not the edge along which the packet leaves node $a$. Note that in the latter case the $i$-th bit of the destination address of packets that depart from $D$ at $\tau_j$ is different from those that depart at later points. It is easily dedeuced that once the priorities are fixed, the sets $P_1,P_2,\ldots P_j,\ldots P_d$ are pairwise disjoint.

Now $P_j$ is just the union of $X[w]$ for various $w \in \{0,1\}^{\ell}$ such that $w$ and $a$ agree in the first $i$ bits. By the assumption about the size of $X[w]$ it follows that $|P_j| \leq 2cn2^{n-i}$.

Let $R_j$ be the set of keys that have nonzero probability of being routed through $\tau_j$ once the priorities have been decided. Since the priorities are determined by Bernoulli trials with probability $(\beta n)^{-1}$, Fact 4.2 can be used to give the following bound

$$\text{Prob}(|R_j| > 4cn2^{n-i}(\beta n)^{-1}) \leq \exp(-k \cdot 2^{n-\ell})$$

for an appropriate constant $k > 0$. The second term in the theorem follows from multiplying the above bound by the number of choices of $D$ and $j$.

Finally, let $K_j$ be the actual set of keys that do depart from $D$ at $\tau_j$ because both the priority and the initial positions were appropriately chosen. For each such packet the initial position must agree with $a$ in the last $n-i$ bits. Hence $K_j$ is determined by $R_j$ Bernoulli trials each with probability $2^{i-n}$ of success. Hence assuming $|R_j| < 4cn2^{n-i}(\beta n)^{-1}$ for each $j$ we have Bernoulli trials with expectation $\leq 4c/\beta$. To upper bound

$$\sum_{j=1}^{d} |K_j|$$

we appeal to Hoeffding's Theorem (Fact 4.4). We have at most $cn2^n$ trials with mean at most $(4c/\beta)(\ell+\beta n) \leq 5cn$ if $\beta \geq 4$. Using Fact 4.2 it follows that

$$\text{Prob}(\Sigma|K_j| > c_3 c\alpha n) < 2^{-c_3 c\alpha n} \qquad (1)$$

if $c_3\alpha > 5e^2$.

Finally, we have to consider the case of packets being involved in more than one task of $D$. This can be done by considering any *fixed* assignment of keys to departure points in $D$ and considering the probabilities of repeated earlier involvement in $D$. If a key was involved in $D$ at $\tau_j$ then the probability of a previous involvement at $\tau_{j-1}$ is at most one half independent of subsequent involvements. Hence if a key was involved in $D$ at $\tau_j$ then the probability of $t$ previous involvements (i.e., with $\tau_{j-1},\ldots,\tau_{j-t}$) is at most $2^{-t}$. It follows that

$$\text{Prob}(T(D) \geq K+s \text{ and } \Sigma|K_j| = K) \leq 2^{-s} \cdot \text{Prob}(\Sigma|K_j| \geq K). \qquad (2)$$

From (1) and (2) it follows that if $c_3\alpha > 5e^2$.

$$\text{Prob}(T(D) \geq 2c_3 c\alpha n) < 2^{1-c_3 c\alpha n}. \qquad \square$$

## 6. DETERMINISTIC SORTING AND RANKING

We use as subroutines some known deterministic algorithms. A crucial step in splitter finding is sorting a sparse subset of elements. For this we can use the algorithm of Nassimi and Sahni [8].

Theorem NS. *For any* $\varepsilon > 0$ $N^{1-\varepsilon}$ *keys can be sorted on a* $CCC_n$ *when* $N = n2^n$ *in time* $O(n)$.

Step C of the overall algorithm determines the rank of every element given that it is "almost" sorted. Suppose that for some $v$ we have that all elements are in nodes at stage $i$ and for all $w_1 <\bullet w_2$, $|w_1| = |w_2| = i$ the keys in $V[w_1]$ are smaller than the keys in $V[w_2]$. If $i = n$ then we have a complete sort except that the elements may not be uniformly distributed among the stage 0 nodes. In this situation the rank of each key can be determined by first sorting the keys at each node locally. The global rank computation is performed on the binary tree that has these nodes as leaves and consists of all forward internal edges, and just those forward external edges along which some address bit changes from 0 to 1. The number of keys in each subcube can be determined recursively by sending these sums from the leaves toward the root and accumulating at each internal tree node. Finally in a reverse information flow from the root to the leaves, the range of the ranks in each subcube can be determined, and hence the ranks of the individual keys. This all takes $O(n)$ parallel transfers of tokens that contain only binary numbers of $O(n)$ digits.

In Step C of the actual algorithm we start with only a partial sort (i.e., for all $w_1 <\bullet w_2$ with $|w_1| = |w_2| = n-s$ where $s = 6\log_2 n$, for all $x \in V[w_1]$ and $y \in V[w_2]$, $x < y$). To find ranks in this situation we determine the rank range for each $X[w_1]$, sort each $X[w_1]$, and finally deduce the rank of each element. The determination of the rank ranges and final rank is as described in the above paragraph. With overwhelming probability each $X[w_1]$ will have at most $2n2^s$ packets. For sorting $X[w_1]$ we assign a separate $CCC_t^*$ to it where $t = s + \log n - \log s$. At least if $t$ divides $n$, one can find $n2^n/(t2^2)$ disjoint copies of $CCC_t^*$ in $CCC_n^*$. The packets are routed to their appropriate copy of $CCC_t^*$ (Theorem D) and then sorted there by some $o(n)$ method such as Batcher's (see Preparata and Vuillemin [9]) which takes $O(\log n)^2$. The above described algorithm for ranking the elements given a partial sort will be called *Algorithm C*.

## 7. SPLITTER FINDING

We describe a procedure SF that given a $CCC_n^*$ with $c$ packets at each node finds a subset $U$ of $2^n/n^\delta$ packets called "splitters" that divide the ordered sequence of the $cn2^n$ total packets into intervals that are, with large probability, all of length smaller than $2cn^{\delta+1}$. The procedure is recursive, nested recursive calls corresponding to nested subcubes. At the i-th level of recursion the splitters found divide the ordered sequence into $2^{n(1-1/2^i)}$ roughly equal intervals. The subcubes at the i-th level are $CCC_r^*$ where $r = n/2^i$ $[i = 0,\ldots,\log n - \log(2\delta \log n)]$. At the i-th level a fraction of about $2^{-i}$ of the packets are considered "active". The choice of splitters at lower levels is restricted to these active elements. In this way the average density of active packets in each $CCC_r^*$ is kept a constant $c$ independent of the cube size. This is necessary for the recursive procedure to succeed. Any integer greater than or equal to six suffices as a value of $\delta$.

The set $U$ of all splitters found in a run of $SF[\lambda]$ will be used in Step B of the overall sorting procedure.

The procedure SF applied to the subcube with root $(w,n-m)$, where $|w| = n-m$, is as follows. When the procedure is called initially with $w = \lambda$ all the packets are considered active.

Procedure SF(w)

(1) Let $Y[w]$ be the active packets in $V[w]$. For each $x \in Y[w]$ route $x$ to a random node in $V[w]$.

(2) For each $w_1$, $|w_1| = m/2 + 2\log n$, choose at random an active element from $V[ww_1]$. Sort this

set $S^*$ of $n^2 2^{m/2}$ chosen elements using Theorem NS. Route the j-th largest to the address that is the binary representation of $\tilde{w} + j2^{m/2}/n^2$. Let $S$ the newly created set of splitters be the packets at addresses $\tilde{w} + j2^{m/2}$ for $j = 1,\ldots,2^{m/2} - 1$. If the splitter is found at address $ww_1$ and $w_1 = w_2 1 w_3$ where $w_3 \in 0^*$ then the splitter is denoted by $\sigma[ww_2]$ and routed deterministically to every node in $V[ww_2]$.

(3) For each $x \in Y[w] - S$ decide according to a Bernoulli trial with probability one half whether it is to remain active. Let the active subset of $Y[w]$ be $Z[w]$.

(4) Apply SDR with the newly found splitters to $Z[w]$.

(5) For each $w'$ with $|w'| = m/2$ let $Y[ww']$ be the subset of $Z[w]$ routed to subcube $V[ww']$ by (4). For each such $w'$ call in parallel $SF(ww')$ for $Y[ww']$ as active elements, unless $m = 2\delta \log n$.

We have seen that SDR for $CCC_r$ takes time $O(r)$ with overwhelming probability. Theorem A will establish that if SF is run, with the recursive calls of SF being allowed to be asynchronous, then the overall algorithm runs in time $O(n)$ with large probability. The main fact which has to be established (Theorem 7.2) is that with overwhelming probability, at every call of SDR the hypotheses of Theorem B are satisfied. We leave it to the reader to verify that all the other operations performed in a call of $SF(w)$ with $|w| = n-m$ can be achieved deterministically, by pipelining if necessary, in time $O(m)$.

First we need a technical lemma:

Lemma 7.1. *Given an ordered set $T$ suppose that a set $S^*$ of $n^2 2^{m/2}$ elements are then chosen from $T$ at random and $S^*$ is then sorted. Let $S \subset S^*$ be the subset of elements having positions $n^2, 2n^2, \ldots (2^{m/2}-1)n^2$ in the ordered set.*

*Suppose $t_0,\ldots,t_{f+1}$ is the longest ordered subsequence of $T$ such that $t_0, t_{f+1} \in S$ but $t_1,\ldots,t_f \notin S$. Then*

(i) $\text{Prob}(f > (1+n^{-1/3})|T|/2^{m/2}) = N^{-\omega(1)}$

(ii) $\text{Prob}(f < (1-n^{-1/3})|T|/2^{m/2}) = N^{-\omega(1)}$

*Suppose that a subset $Y \subseteq T - S$ is chosen by performing independent Bernoulli trials with probability 1/2. Let $y_0,\ldots,y_{h+1}$ be the longest ordered subsequence of $Y \cup S$ such that $y_0, y_{h+1} \in S$ but $y_1,\ldots,y_h \notin S$. Then*

(iii) $\text{Prob}(h > (1+2n^{-1/3})|Y|/(2\cdot 2^{m/2})) = N^{-\omega(1)}$

(iv) $\text{Prob}(h < (1-2n^{-1/3})|Y|/(2\cdot 2^{m/2})) = N^{-\omega(1)}$

*These claims assume that $n^4 2^{m/2} = o(|T|)$ and $n^2 2^{m/2} = o(|T|^{2/3})$.*

Proof. All choices of $S^*$ are equally likely. To prove (i) and (ii) consider any sequence $t_0,\ldots,t_{f+1}$ with $f = (1 \pm n^{-1/3})|T|/2^{m/2}$. Then the probability that of the $n^2 2^{m/2}$ members of $S^*$ exactly $n^2$ lie in the above range and the rest outside is

$$\binom{|T|-f}{n^2 2^{m/2}-n^2}\binom{f}{n^2}\Big/\binom{|T|}{n^2 2^{m/2}}$$

Applying Fact 4.6 with $A = |T|-f$, $a = f$, $X = n^2 2^{m/2} - n^2$, $x = n^2$ gives $G = n^{5/3}$ and an upper bound of

$$\exp(-n^{4/3}/6)$$

provided $n^4 2^{m/2} = o(|T|)$ and $n^2 2^{m/2} = o(|T|^{2/3})$. This establishes (i) and (ii) since there are at most $2^n$ choices of $t_0$, $t_{f+1}$ and $f$, respectively.

To show (iii) and (iv) it is sufficient to prove that in a sequence of $(1 \pm n^{-1/3})|T|/2^{m/2}$ ordered elements of $T$ the probability that the number of elements chosen to be in $Y$ is outside the range $(1 \pm 2n^{-1/3})|T|/2\cdot 2^{m/2}$ is negligible. In fact Lemma 4.3 upper bounds this probability by

$$\exp(-n^{-2/3}|T|/(4\cdot 2^{m/2}))$$

which is bounded above by $\exp(-n^{4/3})$ if $2^{m/2}\cdot n^2 = o(T)$. □

Theorem 7.2. *In a run of $SF(\lambda)$ the probability of each of the following events for each recursive call of $SF(w)$ is bounded above by $N^{-\omega(1)}$ provided $m \geq 12 \log n$.*

(a) *Step (ii) fails because $V[ww_1]$ has no active packets.*

(b) *In the call SDR for subcube $w$ with $|w| = n-m$, it happens that $|Z[w]| > 2cm2^m$ or $|Z[w]| < cm2^m/2$.*

(c) *In the call SF for subcube $w$ with $|w| = n-2m$ two neighboring splitters are created that in the total ordering of the $cn2^n$ elements have more than $2cn2^m$ elements between them.*

*Since in a run of $SF[\lambda]$ there are at most $3N$ such events altogether the probability that any such event ever occurs in a run is therefore also bounded by $N^{-\omega(1)}$.*

Proof. The proof proceeds by induction on the depth of recursion. We assume that the Theorem holds down to the current level of recursion and argue that the probability of "going wrong" at the current call is less than $N^{-\omega(1)}$.

(a) Since the active elements $Y[w]$ have been sent to random nodes in $V[w]$ the probability that all of at least $cm2^m/2$ elements miss $V[ww_1]$ is at most

$$(1 - 1/(2^{m/2}n^2))^{cm2^m/2}$$

By Fact 4.1 this is bounded above by

$$\exp(-cm2^{m/2}/(2n^2)) = N^{-\omega(1)}$$

if $m \geq 12 \log n$.

(b) We assume inductively that in the call of SDR at the i-th level of recursion the set of active elements denoted again by $T$ in the subcube corresponding to $w$ is in the range $(1 \pm 2n^{-1/3})^i cn2^{m-i}$.

Then by Lemma 7.1 (iii) the probability that the number of active elements in a subcube at the $(i+1)$-st level call is in the range $(1 \pm 2n^{-1/3})\, 2^{-m/2}.2^{-1}$ times this quantity, which is $(1 \pm 2n^{-1/3})^{i+1} cn2^{m/2-(i+1)}$, is bounded by $N^{-\omega(1)}$.

(c) We assume inductively that in the call of SDR at the i-th level of recursion the set $T$ of elements in the corresponding subcube had size at most $(1+n^{-1/3})^i cn2^m$ (where $m = n/2^i$). Applying Lemma 7.1 (i) gives that, at the next level of recursion, the probability of a subcube having more than $(1+n^{-1/3})2^{-m/2}$ times as many packets is bounded above by $N^{-\omega(1)}$. □

Theorem A. *For all* $c \geq 1$ *there is a* $c_5$ *such that for all sufficiently large* $\beta$ *if* $SF(\lambda)$ *is run on* $CCC_n^*$ *with* $c$ *packets per node then*

$$\text{Prob}(T > cc_5\beta n) < N^{-c\beta}.$$

Proof. In a run of SF a critical path is a sequence of nested calls of $SF(\lambda)$, $SF(w_1)$, $SF(w_1w_2)$, $\ldots SF(w_1w_2\ldots w_i),\ldots$ where $|w_j| = n2^{-j}$. The deterministic components of each take time proportional to $|w_i|$. When summed for $i = 1,\ldots,\log n - \log(12\log n)$ this gives an upper bound of $O(n)$ as required. Hence it remains only to analyze the cumulative probabilistic effects of such a chain of calls of SDR. Note that these calls are probabilistically independent.

Theorems B and 7.2(b) say that for sufficiently large $\alpha$ a call of SDR on the subcube with address prefix $w_1\ldots w_i$ exceeds runtime $2cc_2\alpha n/2^i$ with probability less than

$$2^{-c\alpha n/2^i} + 2^{-\Omega(n^6)}$$

Hence it exceeds runtime $2cc_2n/2^i + 2c(\alpha-1)c_2n/2^i = 2cc_2n/2^i + t_i$ (say) with probability less than

$$2^{-t_i/(3c_2)}$$

Hence the probability that such a sequence of nested calls takes time more than $c_2n + t$ is less than

$$\sum_{\Sigma t_i = t} \prod_i 2^{-t_i/(3c_2)} \leq \sum_{\Sigma t_i = t} 2^{-t/(3c_2)}$$

$$\leq 2^{-c(\alpha-1)n/3+o(n)}$$

if $t = 2c_2c(\alpha-1)n$. The result follows for $\beta = \alpha/2$ for $\alpha > 6$ with $c_5 = 4c_2$. □

In conclusion we note that Theorem 7.2(c) ensures that the hypotheses of Theorem B hold when Step B of the overall algorithm is invoked. In other words, when SDR is called with the $2^n/n^6$ splitters found in Step A then the number of elements destined for each $6\log n$ dimensional subcube is never more than twice the average.

In the unlikely event that Step (ii) of SF fails in any call of SF the sorting algorithm is restarted from the beginning.

Acknowledgment. We are grateful to G.H. Gonnet for pointing out an error in an earlier version of this paper.

## 8. REFERENCES


[1] R. Aleliunas. Randomized parallel communication. Proc. of ACM Symp. on Principles of Distributed Computing, Ottawa, Canada (1982), 60-72.

[2] D. Angluin and L.G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. of Comp. and Syst. Sci.* (1979), 155-193.

[3] K. Batcher. Sorting networks and their applications. AFIPS Spring Joint Comp. Conf. 32 (1968), 307-314.

[4] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. of Math. Stat.* 23 (1952), 493-507.

[5] P. Erdös and J. Spencer. *Probabilistic Methods in Combinatorics.* Academic Press (1974).

[6] C.A.R. Hoare. QUICKSORT. *Computer J.* 5(1), (1962), 10-15.

[7] W. Hoeffding. On the distribution of the number of successes in independent trials. *Ann.of Math. Stat.* 27 (1956), 713-721.

[8] D. Nassimi and S. Sahni. Parallel permutation algorithms and a new generalized connection network. *JACM* 29:3 (1982), 642-667.

[9] F.P. Preparata and J. Vuillemin. The cube-connected cycles: A versatile network for parallel computation. *CACM* 24 (1981),300-310.

[10] M.O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity*. J.F. Traub (ed.), Academic Press, New York, 1976.

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

[12] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM J. on Computing* 6, (1977), 84-85.

[13] E. Upfal. Efficient schemes for parallel communication. Proc. of ACM Symp. on Principles of Distributed Computing, Ottawa, Canada (1982), 55-59.

[14] L.G. Valiant. A scheme for fast parallel communication. *SIAM J. on Computing* 11:2 (1982), 350-361.

[15] L.G. Valiant and G.J. Brebner. Universal schemes for parallel communication. Proc. of the Thirteenth Annual ACM Symposium on Theory of Computing (1981), 263-277.