*CS*06201*a*01: Network Computing and Efficient Algorithms

Distributed Sorting

Xiang-Yang Li and Xiaohua Xu

School of Computer Science and Technology
University of Science and Technology of China (USTC)

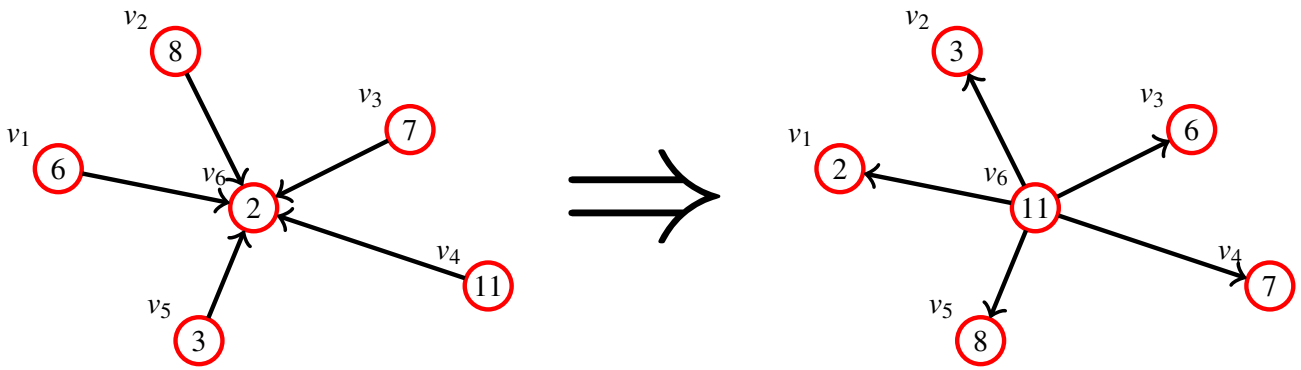
September 1, 2021

Model

Definition (sorting)

We choose a graph with n nodes $v_1, \cdots, v_n$. Initially, each node stores a value. After applying a sorting algorithm, node v_k stores the k-th smallest value.

- Simple Algorithm
  - Send to some node v, sorts it locally, redistributes.
  - With a star topology sorting finishes in O(1) time.

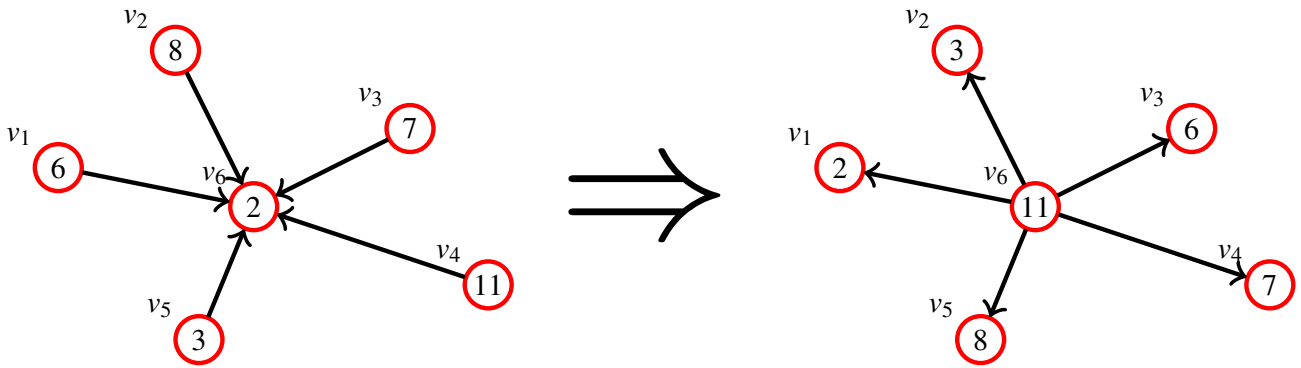


$\Omega(n)$ time, O(n) messages. Problem?

Model

Definition (Node Contention)

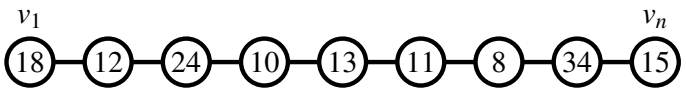
In each step of a synchronous algorithm, each node can only send and receive O(1) messages containing O(1) values, no matter how many neighbors the node has



Complexity to sort star graph? $\Omega(n)$ time! How to do it faster?

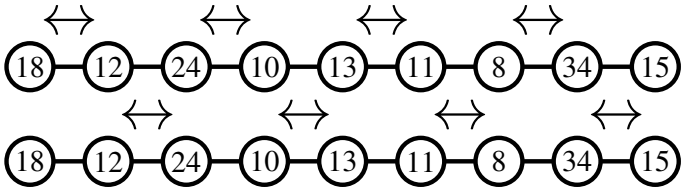
Array

- How to sort in an array?



ALGORITHM 4.3: ODD/EVEN SORT(

1:) Given an array of n nodes $(v_1, \cdots, v_n)$, each storing a value (not sorted).
2: **repeat**
3: Compare and exchange the values at nodes i and $i+1, i$ odd.
4: Compare and exchange the values at nodes i and $i+1, i$ even
5: **until** done



- Correctness?

0 – 1 Sorting Lemma

Lemma (0 – 1 Sorting Lemma)

If an oblivious comparison-exchange algorithm sorts all inputs of 0's and 1's, then it sorts arbitrary inputs.

Proof.

We prove the opposite direction (does not sort arbitrary inputs) does not sort 0's and 1's). Assume that there is an input $x = x_1, \cdots, x_n$ that is not sorted correctly. Then there is a smallest value k such that the value at node v_k after running the sorting algorithm is strictly larger than the k-th smallest value x(k). Define an input $x_i = 0 \leftrightarrow x_i \leq x(k), x_i = 1$ else. Whenever the algorithm compares a pair of 1's or 0's, it is not important whether it exchanges the values or not, so we may simply assume that it does the same as on the input x. On the other hand, whenever the algorithm exchanges some values $x_i = 0$ and $x_j = 1$, this means that $x_i \leq x(k) < x_j$. Therefore, in this case the respective compare-exchange operation will do the same on

Array

Theorem

Algorithm 4.3 sorts correctly in n steps.

Proof.

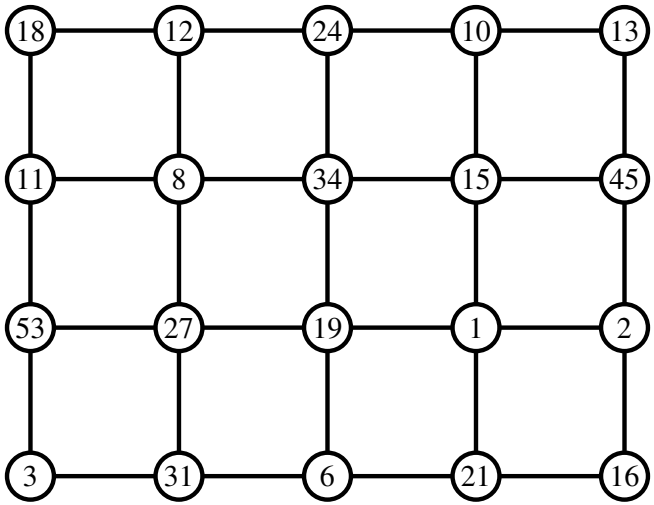
Thanks to Lemma 4.4 we only need to consider an array with 0's and 1's. Let j_1 be the node with the rightmost (highest index) 1. If j_1 is odd (even) it will move in the first (second) step. In any case it will move right in every following step until it reaches the rightmost node v_n. Let j_k be the node with the k-th rightmost 1. We show by induction that j_k is not "blocked" anymore (constantly moves until it reaches destination!) after step k. We have already anchored the induction at $k = 1$. Since j_{k-1} moves after step $k-1$, j_k gets a right 0-neighbor for each step after step k. (For matters of presentation we omitted a couple of simple details.) □

- Maybe we can do better by using a different topology?

Mesh

- How to sort in a mesh (aka grid)?

smallest



largest

Mesh

ALGORITHM 4.6: SHEAR SORT(

1:) We are given a mesh with m rows and m columns, m even, n = m2.
2: The sorting algorithm operates in phases, and uses the odd/even sort algorithm on rows or columns.
3: **repeat**
4: In the odd phases 1; 3; : : : we sort all the rows, in the even phases 2; 4; : : : we sort all the columns, such that:
5: Columns are sorted such that the small values move up.
6: Odd rows (1; 3; : : : ;m) are sorted such that small values move left.
7: Even rows (2; 4; : : : ;m) are sorted such that small values move right.
8: **until** done

Mesh

Theorem

Algorithm 4.6 sorts n values in p n(log n+1) time in snake-like order.

Proof.

Since the algorithm is oblivious, we can use Lemma 4.4. We show that after a row and a column phase, half of the previously unsorted rows will be sorted. More formally, let us call a row with only 0's (or only 1's) clean, a row with 0's and 1's is dirty. At any stage, the rows of the mesh can be divided into three regions. In the north we have a region of all-0 rows, in the south all-1 rows, in the middle a region of dirty rows. Initially all rows can be dirty. Since neither row nor column sort will touch already clean rows, we can concentrate on the dirty rows. □

Mesh (cont'd)

First we run an odd phase. Then, in the even phase, we run a peculiar column sorter: We group two consecutive dirty rows into pairs. Since odd and even rows are sorted in opposite directions, two consecutive dirty rows look as follows:

$$00000\cdots11111$$

$$11111\cdots00000$$

Such a pair can be in one of three states. Either we have more 0's than 1's, or more 1's than 0's, or an equal number of 0's and 1's. Column-sorting each pair will give us at least one clean row (and two clean rows if "$\|0\| = \|1\|$"). Then move the cleaned rows north/south and we will be left with half the dirty rows. At first glance it appears that we need such a peculiar column sorter. However, any column sorter sorts the columns in exactly the same way (we are very grateful to have Lemma 4.4!). All in all we need $2\log m = \log n$ phases to remain only with 1 dirty row in the middle which will be sorted (not

rst are called input wires of the comparison network, the second output wires. Given n values on the input wires, a sorting network ensures that the values are sorted on the output wires. We will also use the term width to indicate the number of wires in the sorting network.

Remarks

- Often we will draw all the wires on n horizontal lines (n being the ”width” of the network). Comparators are then vertically connecting two of these lines.
- Note that a sorting network is an oblivious comparison-exchange network. Consequently we can apply Lemma 4.4 throughout this section.

Sorting Networks

Definition (Depth)

The depth of an input wire is 0. The depth of a comparator is the maximum depth of its input wires plus one. The depth of an output wire of a comparator is the depth of the comparator. The depth of a comparison network is the maximum depth (of an output wire).

Bitonic Sequence

Definition (Bitonic Sequence)

A bitonic sequence is a sequence of numbers that first monotonically increases, and then monotonically decreases, or vice versa.

- $< 1,4,6,8,3,2 >$ or $< 5,3,2,1,4,8 >$ are bitonic sequences.
- $< 9,6,2,3,5,4 >$ or $< 7,4,2,5,9,8 >$ are not bitonic.
- Since we restrict ourselves to 0's and 1's (Lemma 4.4), bitonic sequences have the form $0^i1^j0^k$ or $1^i0^j1^k$ for $i,j,k \geq 0$.

Half Cleaner

ALGORITHM 4.12: HALF CLEARNER(

1:) A half cleaner is a comparison network of depth 1, where we compare wire i with wire $i+n=2$ for $i=1,\cdots,n/2$ (we assume n to be even).

Lemma

Feeding a bitonic sequence into a half cleaner (Algorithm 4.12), the half cleaner cleans (makes all-0 or all-1) either the upper or the lower half of the n wires. The other half is bitonic.

Bitonic Sequence Sorter

ALGORITHM 4.14: BITONIC SEQUENCE SORTER(

1:) A bitonic sequence sorter of width n (n being a power of 2) consists of a half cleaner of width n, and then two bitonic sequence sorters of width n=2 each.
2: A bitonic sequence sorter of width 1 is empty.

Lemma

A bitonic sequence sorter (Algorithm 4.14*) of width n sorts bitonic sequences. It has depth* $\log n$.

Merging Networks

ALGORITHM 4.16: MERGING NETWORK(

1:) A merging network of width n is a merger of width n followed by two bitonic sequence sorters of width $n = 2$. A merger is a depth-one network where we compare wire i with wire $n - i + 1$, for $i = 1, \cdots, n/2$.

- If two sorted sub-sequences are input to Merger, then output two sub-sequences: one clean, other bitonic.
- Note that a merging network is a bitonic sequence sorter where we replace the (first) half-cleaner by a merger.

Merging Networks

Lemma

A merging network of width n (Algorithm 4.16) merges two sorted input sequences of length n=2 each into one sorted sequence of length n.

Proof.

We have two sorted input sequences. Essentially, a merger does to two sorted sequences what a half cleaner does to a bitonic sequence, since the lower part of the input is reversed. In other words, we can use the same argument as in Theorem 4.7 and Lemma 4.13: Again, after the merger step either the upper or the lower half is clean, the other is bitonic. The bitonic sequence sorters complete sorting. □

Batcher's "Bitonic" Sorting Network

ALGORITHM 4.18: BATCHER'S "BITONIC" SORTING NETWORK(

1:) A batcher sorting network of width n consists of two batcher sorting networks of width $n = 2$ followed by a merging network of width n. (See Figure 4.19.)

2: A batcher sorting network of width 1 is empty.

Batcher's "Bitonic" Sorting Network

Theorem

A sorting network (Algorithm 4.18*) sorts an arbitrary sequence of n values. It has depth* $O(\log^2 n)$.

Proof.

Correctness is immediate: at recursive stage $k(k = 1, \cdots, \log n)$, we merge $2k$) sorted sequences into $2k-1$ sorted sequences. The depth $d(n)$ of the sorting network of level n is the depth of a sorting network of level $n = 2$ plus the depth $m(n)$ of a merging network with width n. The depth of a sorter of level 1 is 0 since the sorter is empty. Since a merging network of width n has the same depth as a bitonic sequence sorter of width n, we know by Lemma 4.15 that $m(n) = \log n$. This gives a recursive formula for $d(n)$ which solves to $d(n) = 1/2\log^2 n + 1/2\log n$. □

References

- Shearsort for meshes with higher dimension:
  - Isaac D. Scherson and Sandeep Sen. Parallel sorting in two dimensional VLSI models of computation. Computers, IEEE Transactions on, 38(2):238-249, Feb 1989.
- Asymptotically optimal algorithms for grid network:
  - Clark David Thompson and Hsiang Tsung Kung. Sorting on a meshconnected parallel computer. Commun. ACM, 20(4):263-271, April 1977.
  - Claus Peter Schnorr and Adi Shamir. An optimal sorting algorithm for mesh connected computers. In Proceedings of the eighteenth annual ACM symposium on Theory of computing, STOC 86, pages 255-263, New York, NY, USA, 1986. ACM.
- A sorting network with asymptotically optimal depth O(log n) (but with a large constant hidden in the big-O):
  - Miklos Ajtai, Janos Komlos, and Endre Szemeredi. An O(n log n) sorting network. In Proceedings of the fifteenth annual ACM symposium on Theory of computing, STOC 83, pages 1-9, New York, NY, USA, 1983. ACM.