

Sorting on hypercubic networks

Παράλληλοι Αλγόριθμοι

A.Τέντες

- Odd-Even Merge Sort
- Sorting Circuit
- Preparata's Algorithm

0-1 Lemma

If a comparison-exchange algorithm sorts input sets consisting solely of 0's and 1's, then it sorts all input sets of arbitrary values

Odd-Even Merge Sort (1/9)

- Works by recursively merging larger and larger sorted lists
- Firstly, we sort $N/2$ pairs and form $N/2$ lists of length 2. Then, these lists are merged to form $N/4$ lists of length 4, etc.
- Merging

We have two lists of length $M = 2^k$:

$$A = a_0, a_1, \dots, a_{M-1} \quad \text{and} \quad B = b_0, b_1, \dots, b_{M-1}$$

We partition them into odd and even index sublists:

$$\text{even}(A) = a_0, a_2, \dots, a_{M-2} \quad \text{and} \quad \text{odd}(A) = a_1, a_3, \dots, a_{M-1},$$

$$\text{even}(B) = b_0, b_2, \dots, b_{M-2} \quad \text{and} \quad \text{odd}(B) = b_1, b_3, \dots, b_{M-1}$$

Odd-Even Merge Sort (2/9)


- Because A,B are sorted, so are the odd and even sublists
- We use recursion to merge even(A) with odd(B) to form C, and odd(A) with even(B) to form D.
- Let $C = c_0, c_1, \dots, c_{M-1}$ and $D = d_0, d_1, \dots, d_{M-1}$
- To merge them we interleave them to form
$$L' = c_0 d_0, c_1 d_1, \dots, c_{M-1} d_{M-1}$$
- To sort this list we only have to compare each c_i with its following d_i

Odd-Even Merge Sort (3/9)

- For example:

A= 2,3,4,8 and B= 1,5,6,7

Then,

even(A)=2,4 and odd(B)= 5,7  C=2,4,5,7

odd(A)=3,8 and even(B)=1,6  D=1,3,6,8

so we have: L'=2,1, 4,3, 5,6, 7,8



L=1,2, 3,4, 5,6, 7,8

Odd-Even Merge Sort (4/9)

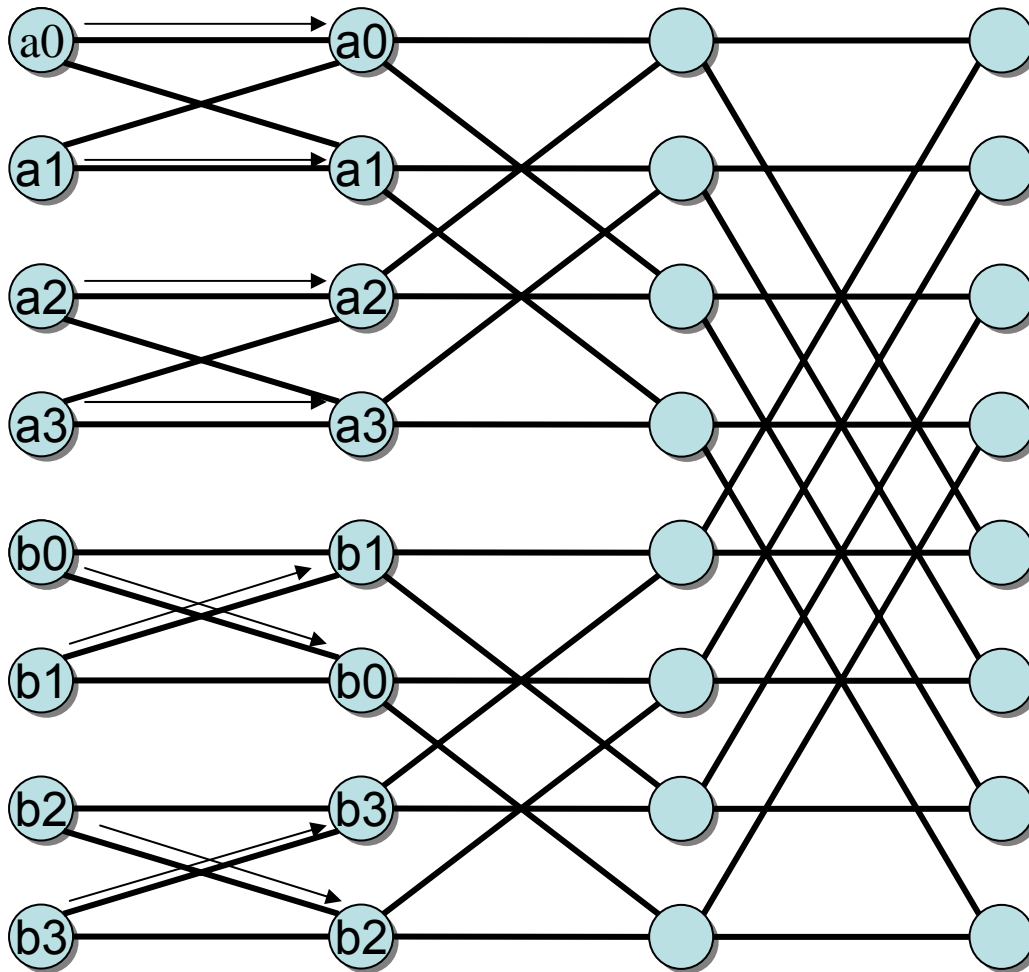
- Why does it work?
- Let A have 'a' zeros and 'M-a' ones and B have 'b' zeros and 'M-b' ones.
- Then $\text{even}(A)$ has $\text{ceil}(a/2)$ zeros and $\text{odd}(A)$ has $\text{floor}(a/2)$ zeros. The same for $\text{even}(B)$ and $\text{odd}(B)$.
- This means that C has $c = \text{ceil}(a/2) + \text{floor}(b/2)$ zeros and D has $d = \text{floor}(a/2) + \text{ceil}(b/2)$ zeros. So C and D differ by at most one.
- If $c=d$ or $c=d+1$

$$L=L'= \overbrace{0,0,\dots,0}^{c+d}, 1,1,\dots,1 \quad \text{and we are done}$$
- Else if $c=d-1$

$$L'= \overbrace{0,0,\dots,0}^{2c}, 1,0,1,1,\dots,1$$

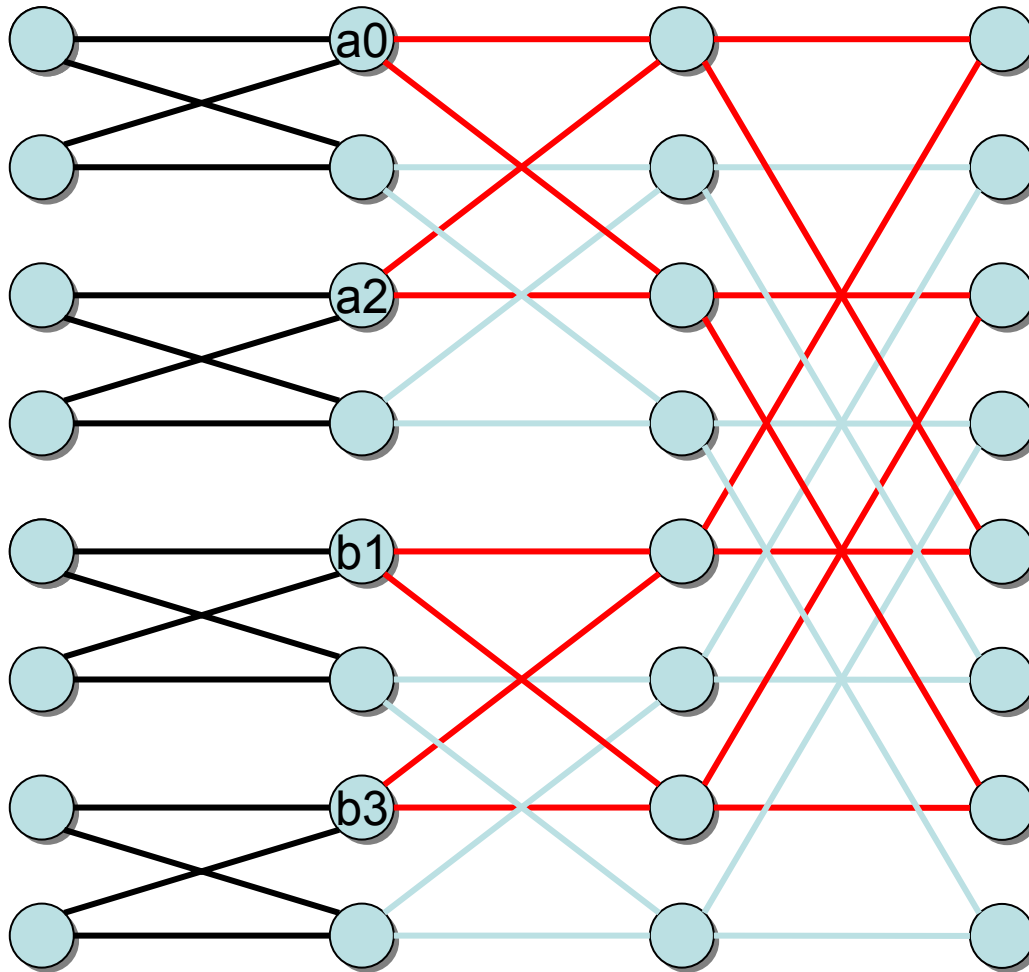
Odd-Even Merge Sort (5/9)

- Implementation on a butterfly. First step:



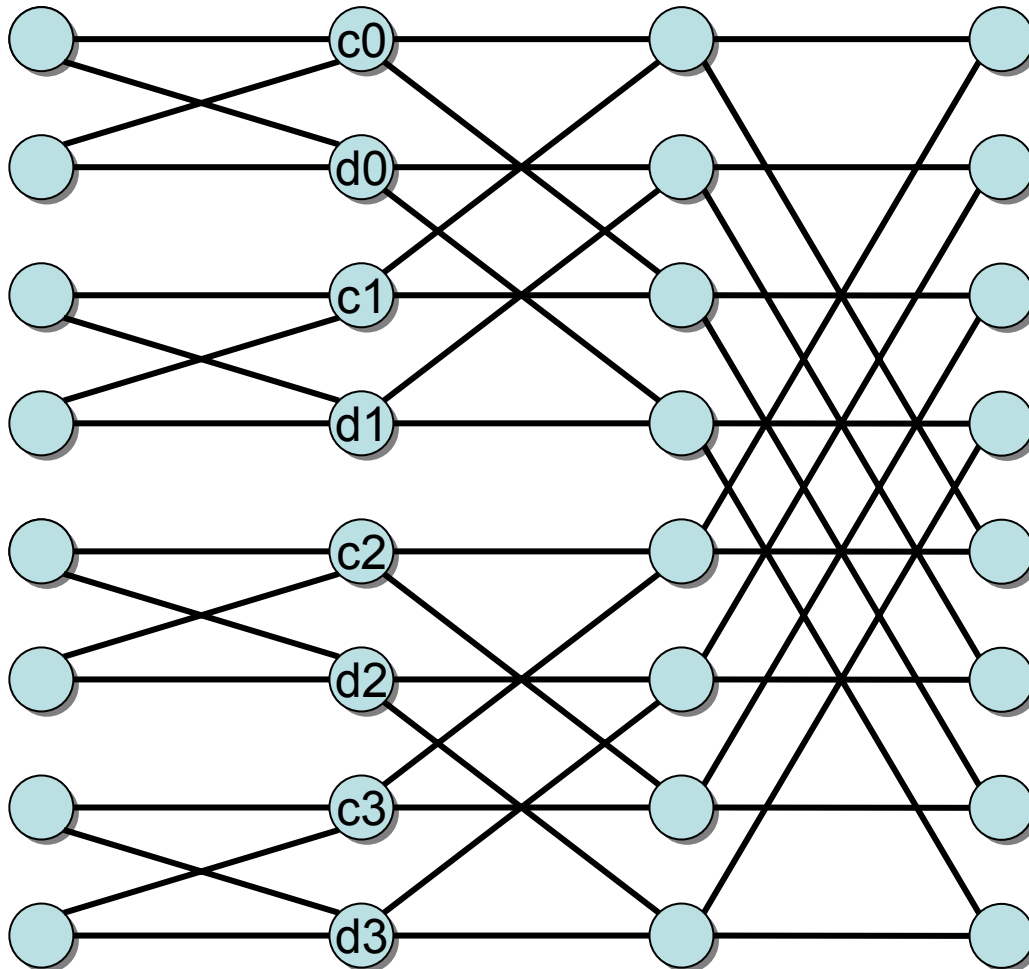
Odd-Even Merge Sort (6/9)

a



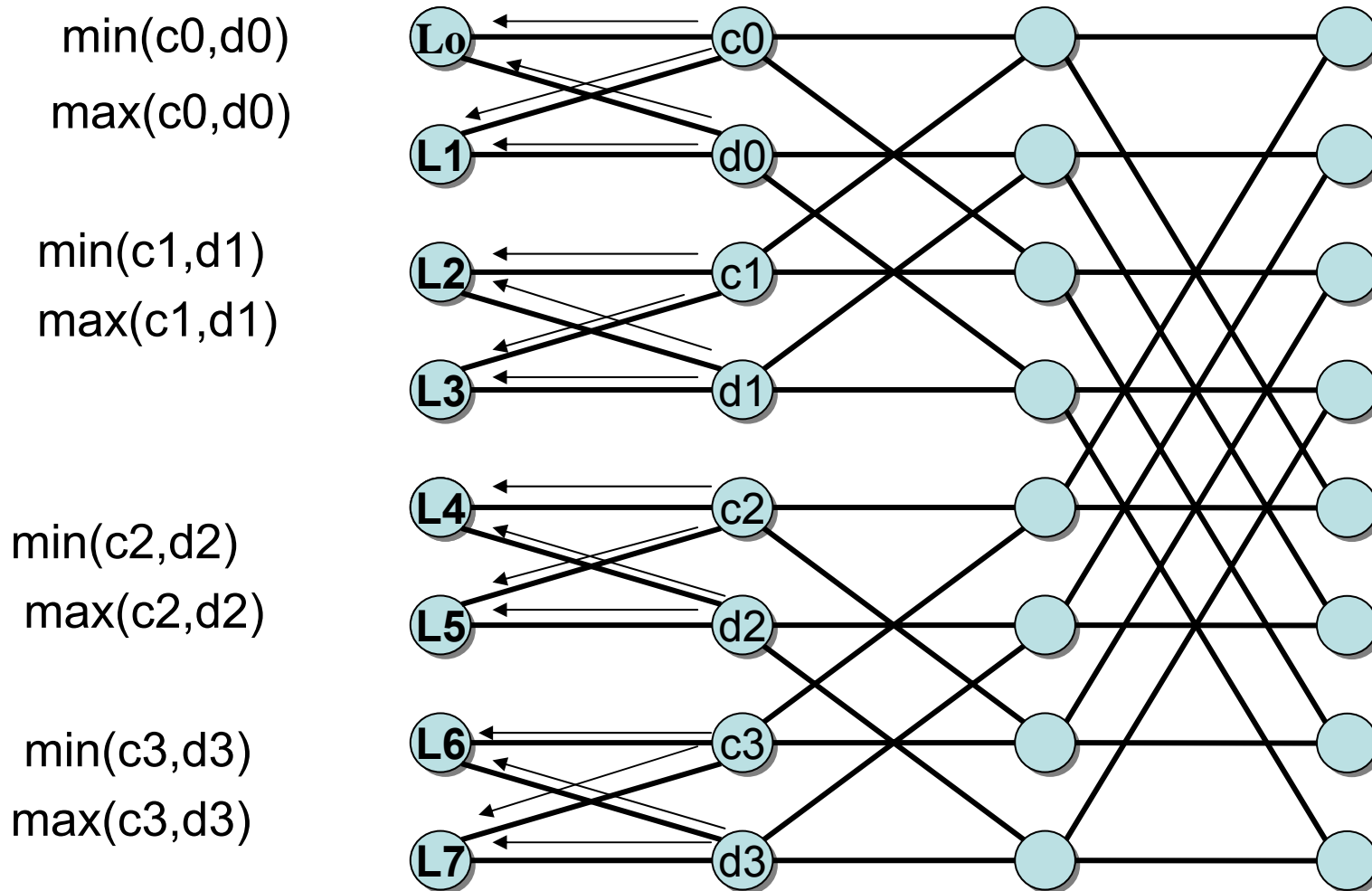
Odd-Even Merge Sort (7/9)

- Location after recursion:



Odd-Even Merge Sort (8/9)

- Last step:



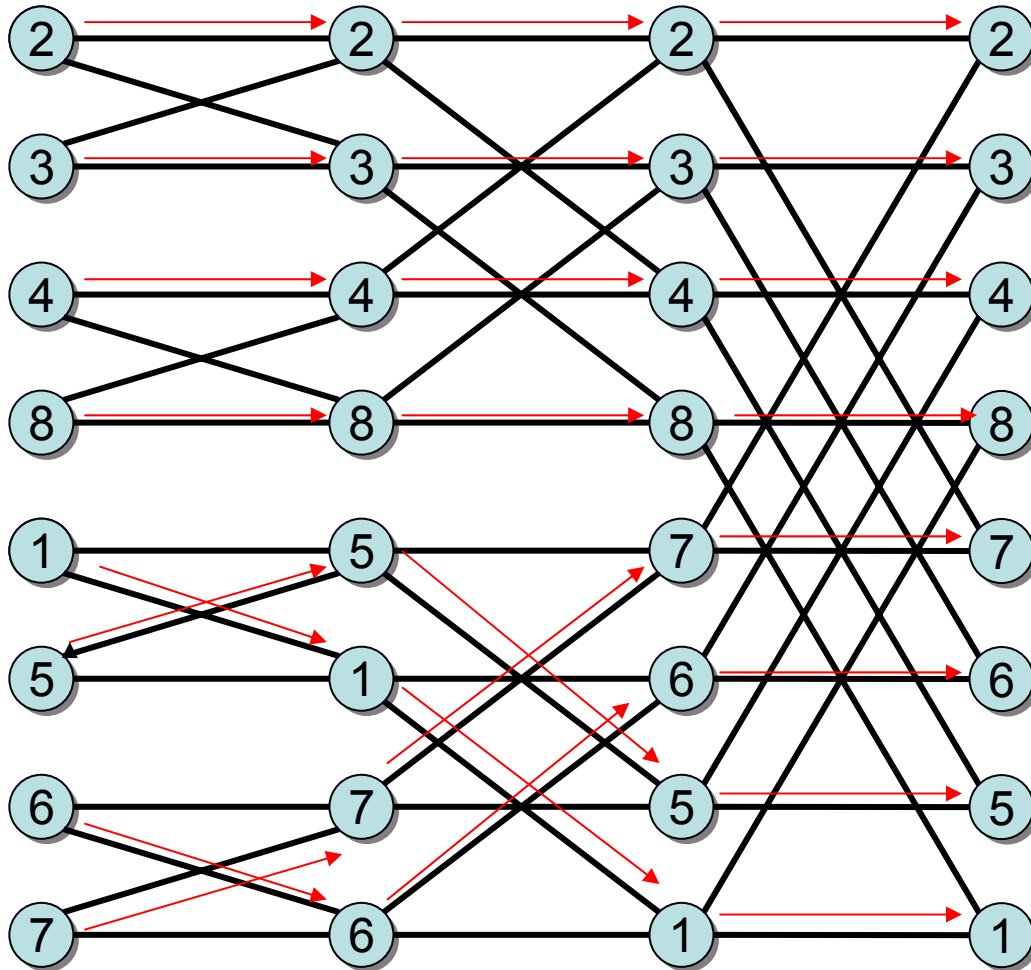
Odd-Even Merge Sort (9/9)

- The procedure to sort 2 lists of length $M/2$ each requires $2\log M$ steps.
- As we have already said, we merge larger and larger lists. At the beginning we merge lists of length 1, then lists of length 2, then lists of length 4, etc, until we merge two lists of length $N/2$. Thus, the total complexity is:

$$2\log 2 + 2\log 4 + 2\log 8 + \dots + 2\log N = \log N(1 + \log N)$$

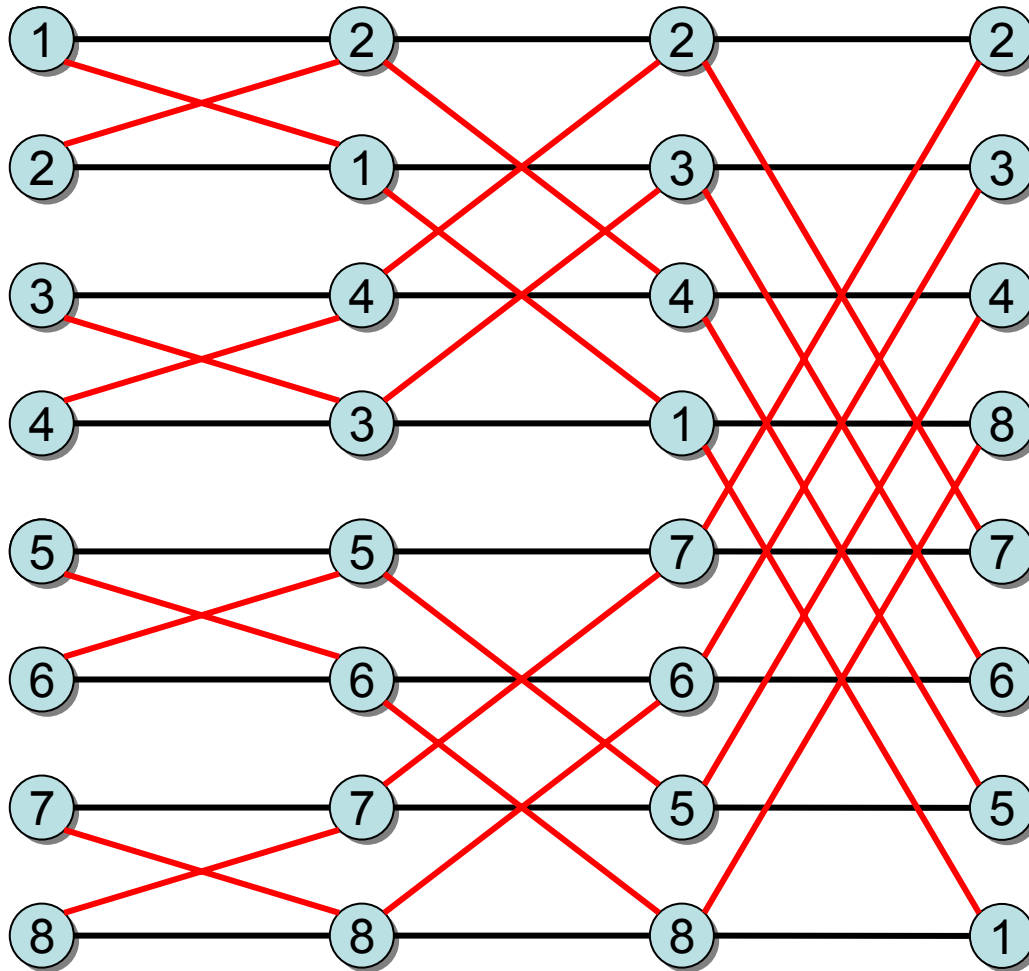
Example (1/2)

a



Example (2/2)

s

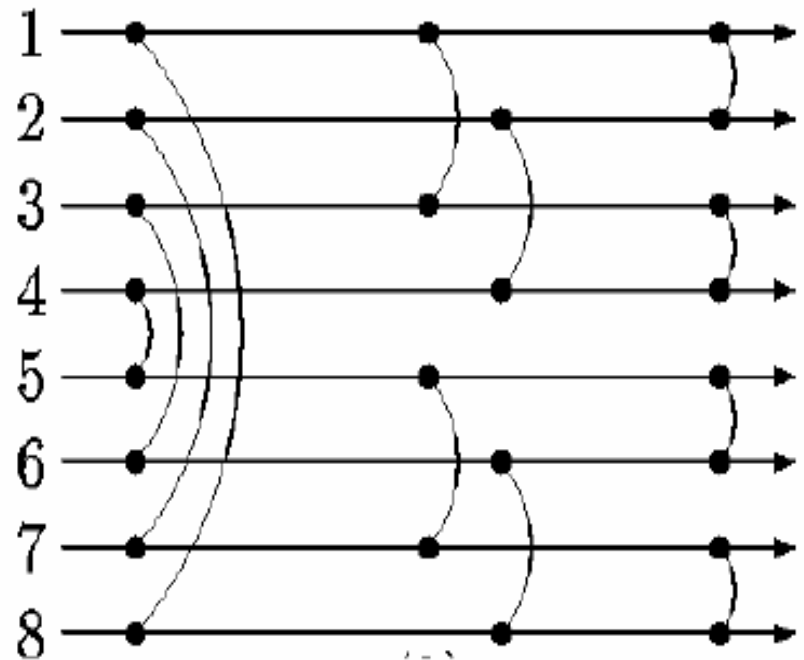
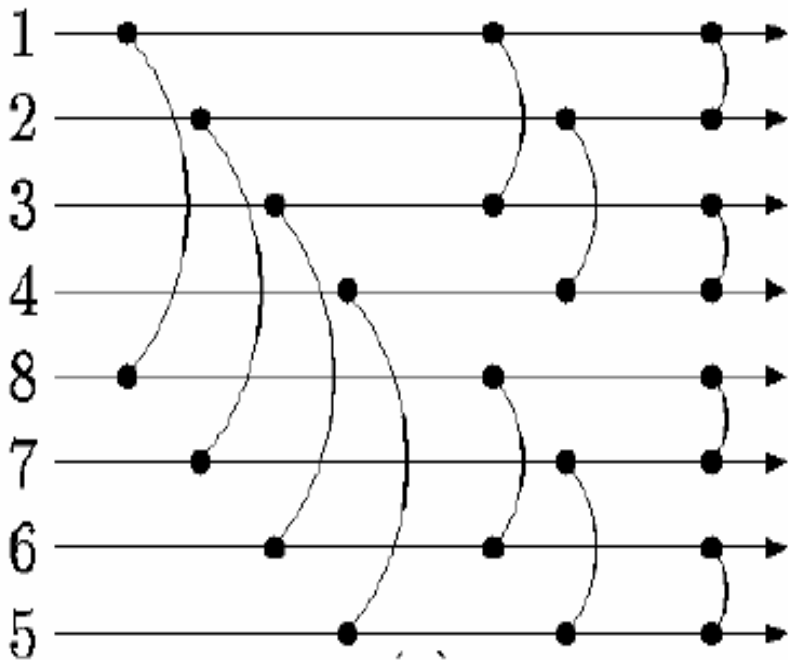


Sorting Circuit (1/4)

- If we take a closer look to the procedure of merging two lists, we observe that after the first $\log M$ steps, the first $m/2$ values remain in the same level, while the rest go to level $3M/2-i$.
- This means that we can build a Sorting Circuit, in which we can avoid the first $\log M$ steps. Consequently, we have the following known Circuit, with depth $\log N(\log N + 1)/2$:

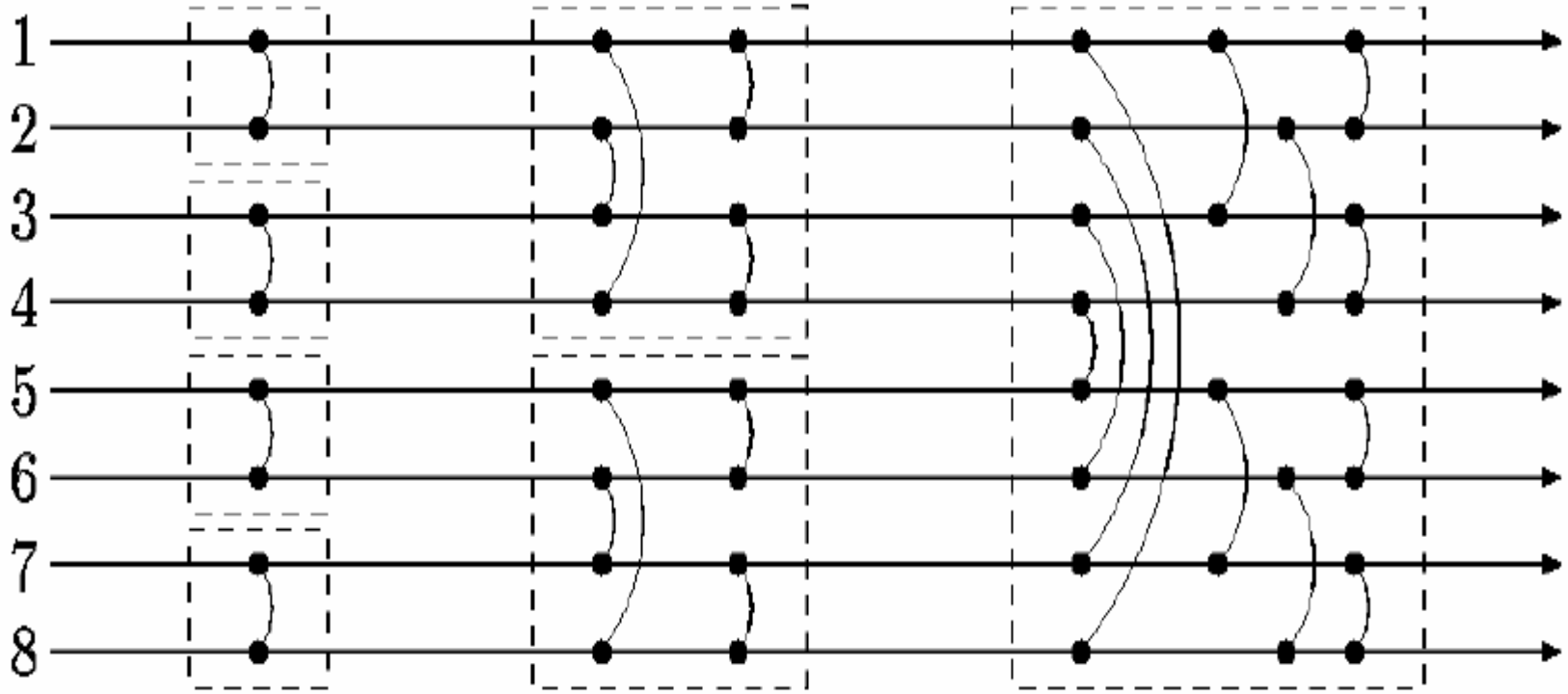
Sorting Circuit (2/4)

- Merger of two sorted lists:



Sorting Circuit (3/4)

- Sorter:



Sorting Circuit (4/4)

The last merge stage requires depth $\log N$, because the first register is compared with the $n/2$ -th register, then with the $N/4$ -th register, etc. The merge stage before the last requires depth $\log(N/2)$, because there are two pairs of lists to be merged in parallel. Thus the total Circuit requires:

$$\log N + \log(N/2) + \log(N/4) + \dots + \log 2 = \log N(\log N + 1)/2$$

Preparata's Algorithm

- Sorts M elements using N processors, for any $M \leq N/4$.
- Requires $O(\log M \log N / (\log(N/M)))$ steps
- If $M \leq N^{1-\epsilon}$ for some constant $\epsilon > 0$, the algorithm runs in $O(\log M)$ steps, which is optimal.

Preparata's Algorithm (1/9)

- The key for this algorithm is a procedure for merging s lists of size r in $O(\log rs)$ steps in a hypercube with $2rs^2$ nodes.
- In the merging procedure we merge every list with another. So we will be able to compute the rank of each item within each list. Then, by summing these partial ranks, we are able to compute the overall rank of each item.

Preparata's Algorithm (2/9)

- The procedure consists of 9 phases.
- The a -th item of the b -th list, x_{ba} , is located in node:

$$y \quad \underbrace{\text{bin}(b-1)}_{\text{logs bits}} | \underbrace{\text{bin}(a-1)}_{\text{logr bits}} | \underbrace{0 \dots 0}_{\text{logs bits}} | \underbrace{0}_{\text{1bit}}$$

- We define H_{bk} to be the 2^r -node subhypercube with nodes:

$$\underbrace{\text{bin}(b-1)}_{\text{logs bits}} | \underbrace{* \dots *}_{\text{logr bits}} | \underbrace{\text{bin}(k-1)}_{\text{logs bits}} | \underbrace{*}_{\text{1bit}}$$

with $H_{bk}(1) : \text{bin}(b-1) | * \dots * | \text{bin}(k-1) | 1$

and $H_{bk}(0) : \text{bin}(b-1) | * \dots * | \text{bin}(k-1) | 0$

Preparata's Algorithm (3/9)

- Phase 1: The b -th list is replicated and stored in $H_{bk}(0)$ for $1 \leq k \leq s$. In other words x_{ba} is copied in all nodes of the form

$$\text{bin}(b-1) | \text{bin}(a-1) | \text{bin}(k-1) | 0 \quad , \text{ for all } k$$

- This can be accomplished in $\log s$ steps. In each step, every node which receives x_{ba} sends it to a neighbour of his. After t steps the value will have been copied into all nodes of the form

$$\text{bin}(b-1) | \text{bin}(a-1) | \overbrace{ * \dots * }^t \overbrace{ 0 \dots 0 }^{\log s - t} | 0$$

Preparata's Algorithm (4/9)

- Phase 2: The list contained in $H_{bb}(0)$ is copied into $H_{bb}(1)$ for all $1 \leq b \leq s$.
- Phase 3: The k -th list is copied into $H_{bk}(1)$ for all $1 \leq b \leq s$. After this phase subhypercube H_{bk} contains the b -th and k -th lists in sorted order.
- Phase 4: These lists are merged with Odd-Even Merge Algorithm.
- Phase 5: For each x_{ba} we compute the number of elements, which are smaller in the k -th list for all a, b, k . This is denoted by $y_k(x_{ba})$ and can be done with parallel prefix computation.

Preparata's Algorithm (5/9)

- Phase 6: We route the value $y_k(x_{ba})$ to node:

$$\text{bin}(b-1)|\text{bin}(a-1)|\text{bin}(k-1)|0 \text{ in } H_{bk}$$

- Phase 7: We compute the total rank of the value by adding all $y_k(x_{ba})$:

$$z_{ba} = y_1(x_{ba}) + y_2(x_{ba}) + \dots + y_k(x_{ba})$$

All the values needed for x_{ba} are contained in subhypercube $\text{bin}(b-1)|\text{bin}(a-1)|^* \dots |0$. So all we have to do is to add all these values.

The addition is done so that the rank of x_{ba} as well as the value itself are contained in $\text{bin}(b-1)|\text{bin}(a-1)|\text{bin}(b-1)|0$.

Preparata's Algorithm (6/9)

It remains only to route the values to the correct positions. This is done in the last two phases:

- Phase 8: We send value x_{ba} from

$$\text{bin}(b-1)|\text{bin}(a-1)|\text{bin}(b-1)|0$$

to

$$\overbrace{\text{bin}(z_{ba-1})}^{\text{logrs bits}}|\text{bin}(b-1)|0$$

- Phase 9: We send value x_{ba} to $\text{bin}(z_{ba-1})|0\dots0|0$

Preparata's Algorithm (6/9)

- This procedure requires $O(\log r s)$ steps.
- If $M \leq \sqrt{N/2}$, the the sorting can be accomplished with a single run of the previous procedure with $r=1$ and $s=M$. Then the algorithm requires $O(\log r s)=O(\log M)$
- If $M > \sqrt{N/2}$ we will use the procedure more than once.

Preparata's Algorithm (7/9)

- We input the a -th item ($1 \leq a \leq M$) to node

$$\overbrace{\text{bin}(a-1)}^{\log M \text{ bits}} | \overbrace{0 \dots 0}^{\log s \text{ bits}} | 0 \text{ where } s = N/2M$$

- Phase 1: We partition the M items into M/s groups of s items each. Each item is treated like a list of length 1, so we use the previous procedure in each such group. We run the procedure in each subhypercube of the form:

$$\text{bin}(w-1) | \overbrace{[* \dots *]}^{\log s} | \overbrace{[* \dots *]}^{\log s} | *$$

Preparata's Algorithm (8/9)

After the first phase we are left with M/s sorted lists of length s . These lists are stored in nodes of the form

$$* \overbrace{\dots}^{\log M} * | \overbrace{0 \dots 0}^{\log s} | 0$$

- Phase 2: We partition the M/s lists into M/s^2 groups of s lists each and run the merge procedure with $r=s$. This can be accomplished by running the procedure in subhypercube of the form

$$\text{bin}(w-1) | \overbrace{* \dots *}^{\log s^2} | \overbrace{* \dots *}^{\log s} | *$$

Preparata's Algorithm (9/9)

- We continue in this fashion. During the i -th phase we partition the $M/s^{(i-1)}$ lists of length $s^{(i-1)}$ into M/s^i groups of s lists each. The i -th phase takes $O(\log s) = O(\log s^i) = O(\log N)$ steps.
- The total number of phases is $O(\log M / (\log(N/M)))$
- The items will be sorted after a total $O(\log M \log N / (\log(N/M)))$ steps