TEMPUS: Activity 2
Parallel Algorithms
Sorting

T. Fadgyas, KFKI-MSZKI

January 2, 1997

Contents

1 SORTING 1

Sorting Generals 2

Sorting Networks 4

Bitonic Sort 5

Bitonic Sort on Hypercube 8

Bitonic Sort on Mesh 9

Bubble Sort and its Variants 10

Quicksort (± best!) 11

1 SORTING

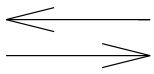
Sorting Generals

- **Important** (e.g. database applications)
- **Definition**

  $S = (a_1, a_2, a_3, \ldots, a_n)$

  $S' = (a'_1, a'_2, a'_3, \ldots, a'_n) \qquad a'_i \leq a'_j, 1 \leq i \leq j \leq n$

- **Types**
  - internal/external
  - comparison-based/non comparison-based
- **Computational complexity**
  - comparison based: $O(n \cdot \log n)$
  - non comparison based: $O(n)$
- **Comparison**

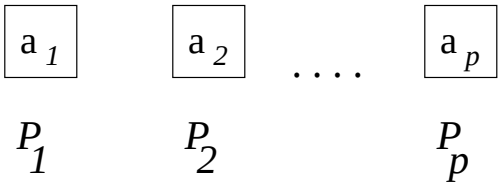
1st step 

$a_i, a_j \qquad a_i, a_j$

2nd step

$a'_i = \min(a_i, a_j) \qquad a'_j = \max(a_i, a_j)$

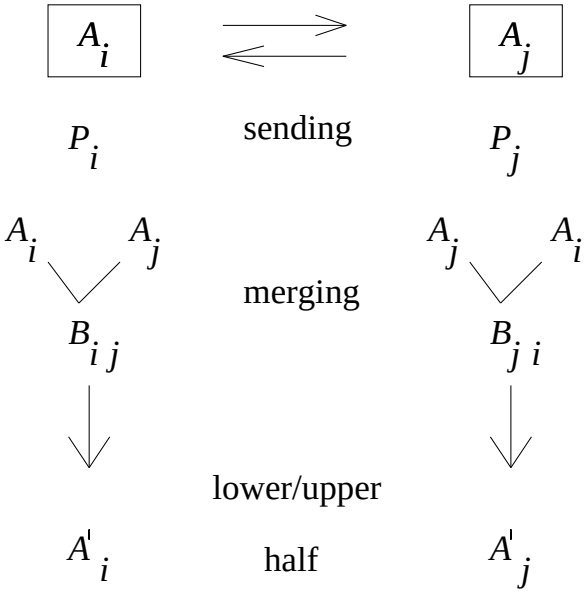
- **One element/processor** (do we have so much?)
  - communication:
    t_s (startup time) + t_w (per word transfer time)
  - compare (min *or* max): t_c (time of comparison)
  - In practice:
    $$t_c \ll t_s$$
- **More elements/processor**



$A_i \leq A_j$ if every element in A_i i is smaller than every element in A_j.

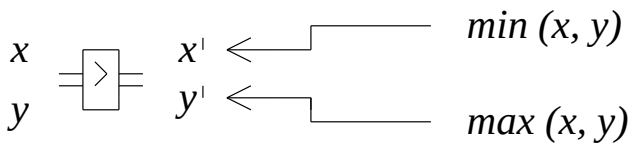
When sorting is finished $A'_1 \leq A'_2 \leq \ldots \leq A'_p$ and A'_i is sorted.

Comparison (A_i, A_j sorted!):



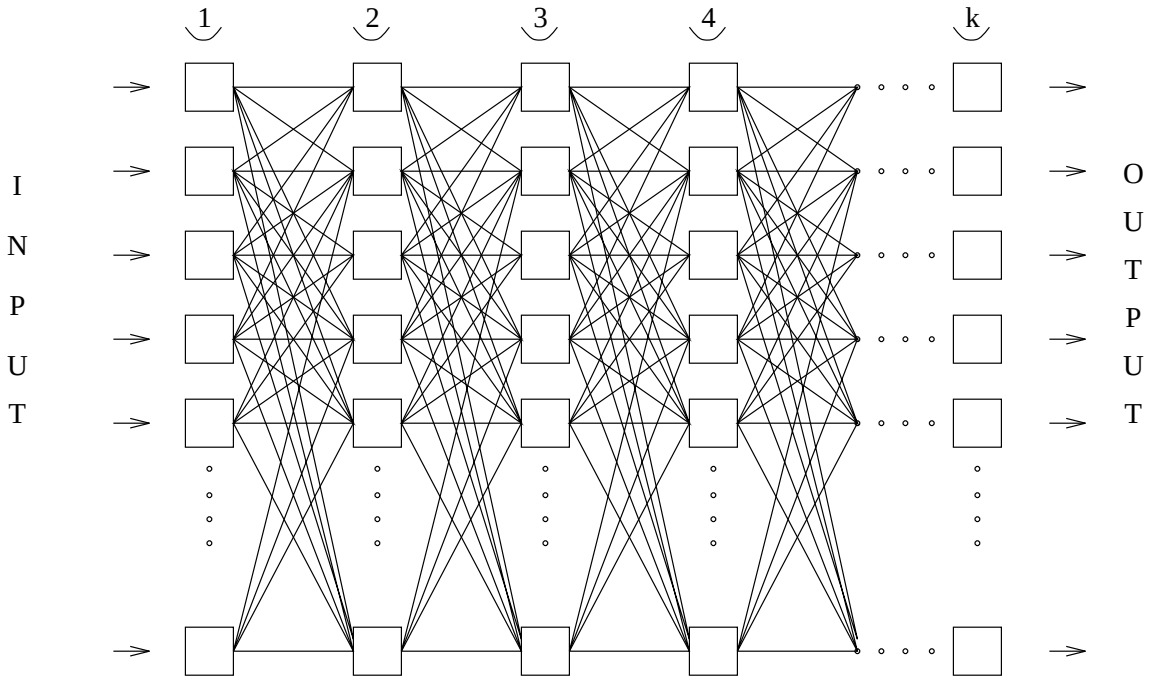
Sorting Networks

- **Comparators**



  - increasing comp. $\oplus$
  - decreasing comp. $\ominus$

- **Depth**



One column executed once, execution time proportional to depth.

Bitonic Sort

- **Time:** $O(\log^2 n)$
- **Bitonic sequence**

  Exists an $1 \leq i \leq n-1$ index:

  $(a_0, a_1, a_2, \ldots, a_i, a_{i+1}, \ldots, a_{n-1})$

  $\leq \leq \leq \ldots \leq \Uparrow \geq \geq \ldots \geq$

  Or, exists a circular shift

  $(a_k, a_{k+1}, a_{k+2}, \ldots, a_0, a_1, \ldots, a_{k-1})$

  such that the resulting array satisfies the previous condition.

- **Bitonic split** (assume $n = 2^k$; index$=\frac{n}{2}$)

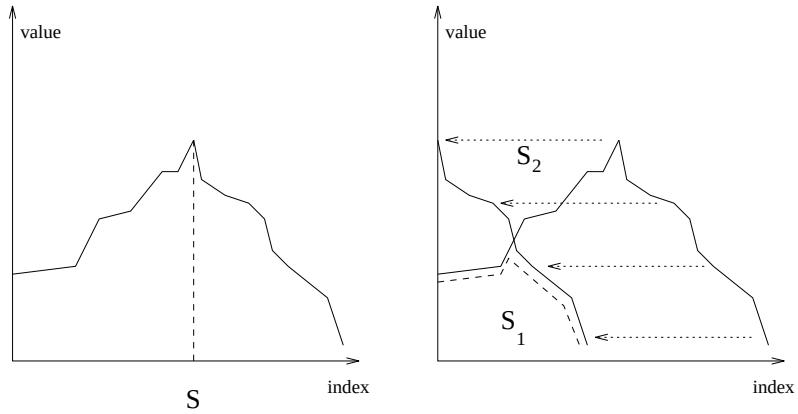
  $S = (a_0, a_1, \ldots, a_{n-1})$

  $S_1 = (\min(a_0, a_{\frac{n}{2}}, \min(a_1, a_{\frac{n}{2}+1}), \ldots \min(a_{\frac{n}{2}} - 1, a_{n-1})$

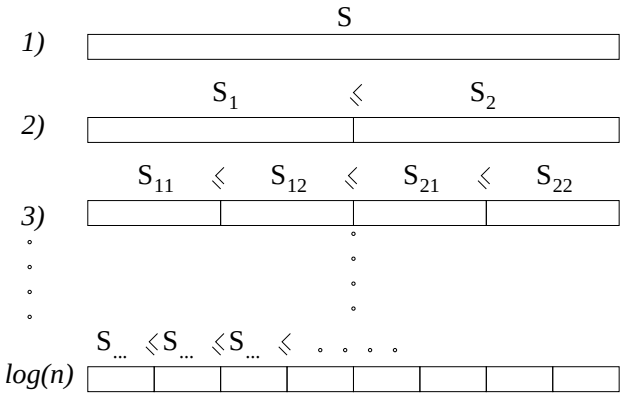
  $S_2 = (\max(a_0, a_{\frac{n}{2}}), \max(a_1, a_{\frac{n}{2}+1}), \ldots \max(a_{\frac{n}{2}-1}, a_{n-1})$

  Then the result, S_1 and S_2 are bitonic sequences and

  $$S_1 \leq S_2$$

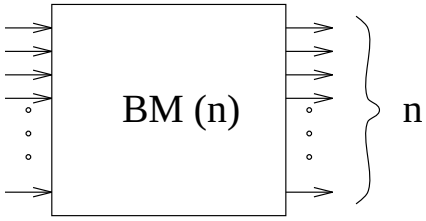


- **Bitonic merge**

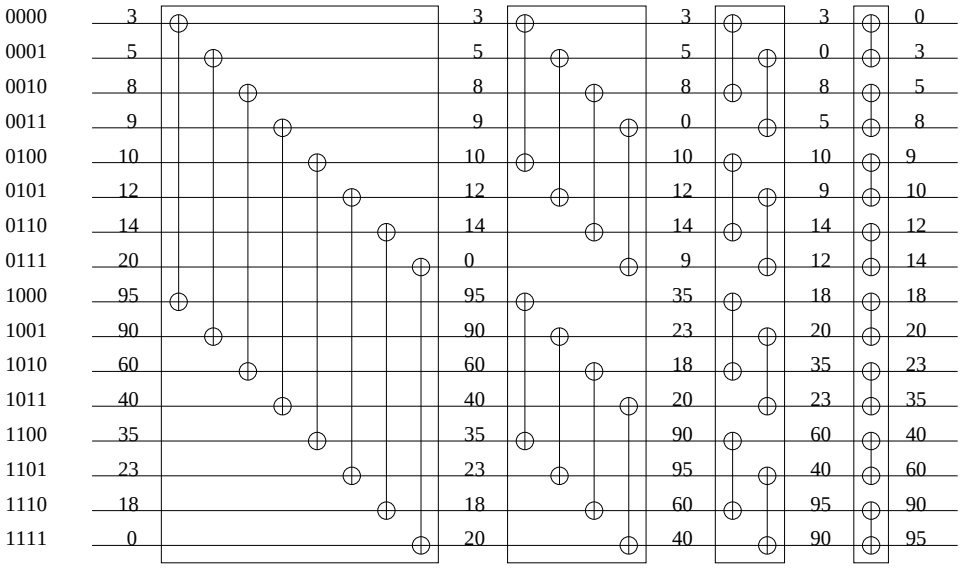


- **Bitonic merging network**

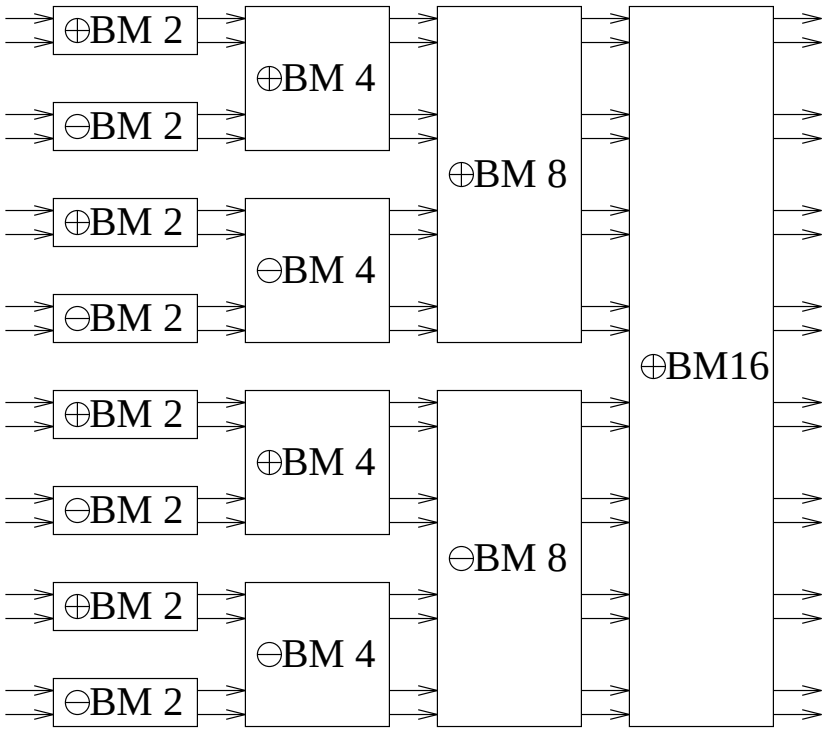
Increasing : $\oplus BM(n)$
Decreasing : $\ominus BM(n)$



- **Example:** $\oplus BM(16)$



- **Bitonic sorting**



A pair of elemets (x, y) is always bitonic.

- **Complexity**

Time (depth):

$d(n) = d(\frac{n}{2}) + \log(n)$

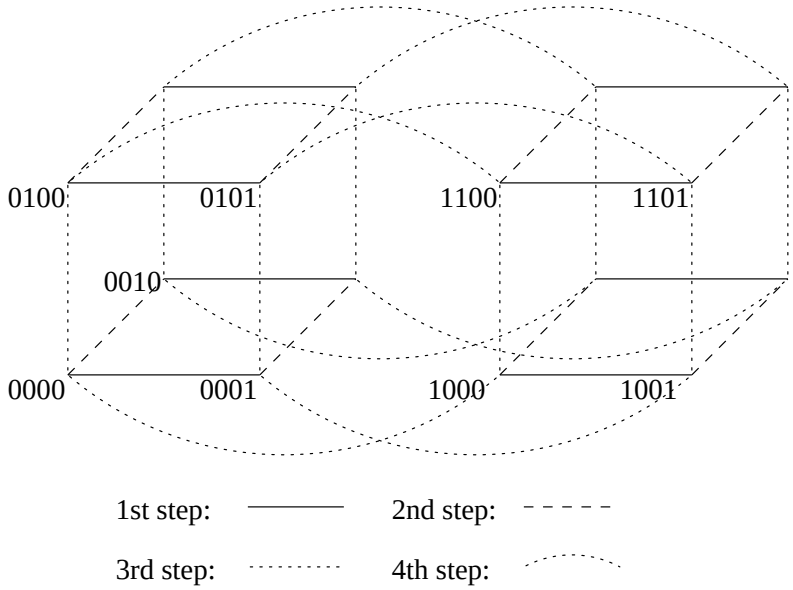
$d(n) = \Sigma_{i=1}^{\log(n)} i = (\log^2(n) + \log(n)) = O(\log^2(n))$

Bitonic Sort on Hypercube

- **Mapping**

  Bitonic processor index (binary form) is the coordinate on the hypercube, e.g. [0100].

- **Executing**



  In the ith step those processors communicate which have indexes differring only in the ith coordinate.

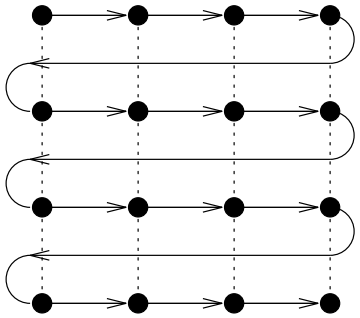
- **Complexity**

  Time: $O(\log^2(n))$ optimal

  Cost: $O(n \cdot \log^2(n))$ non optimal

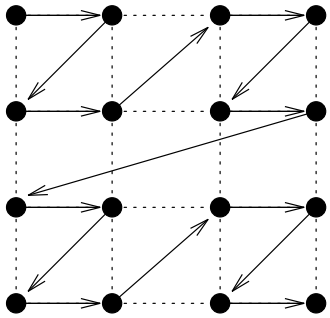
Bitonic Sort on Mesh

- **Mapping**



row-major mapping

(column-major mapping)



row-major snake like mapping

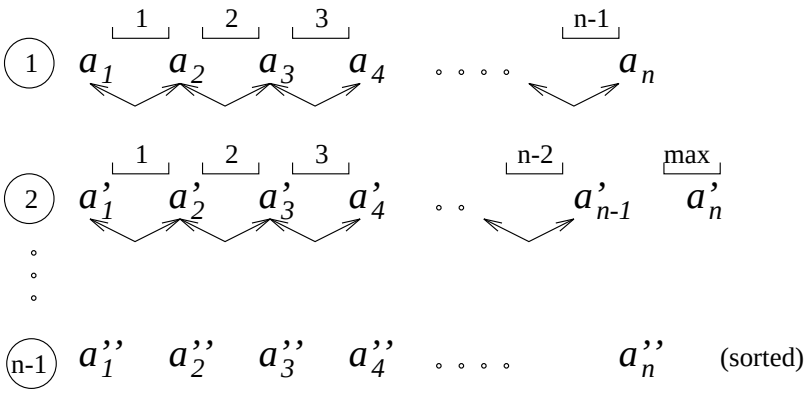
- **Complexity**

Time: $O(\log^2(n)) + O(\sqrt{n})$

comparison communication

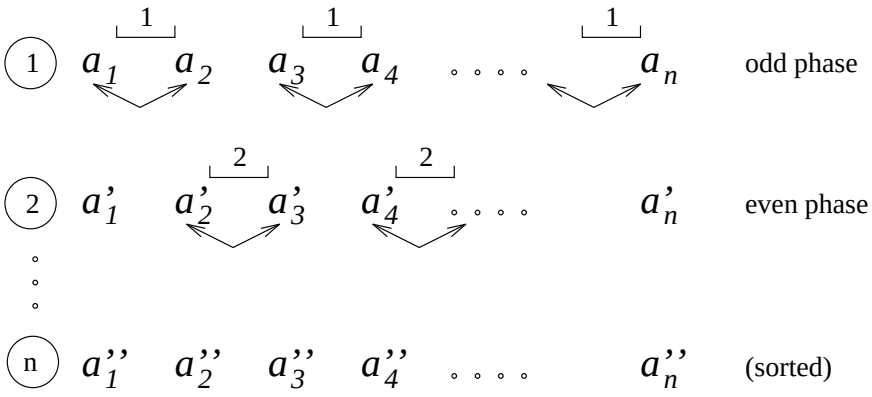
Bubble Sort and its Variants

- **Bubble sort** (sequential $O(n^2)$)



Not suitable for parallelizing.

- **Odd-Even transposition** (sequential $O(n^2)$)



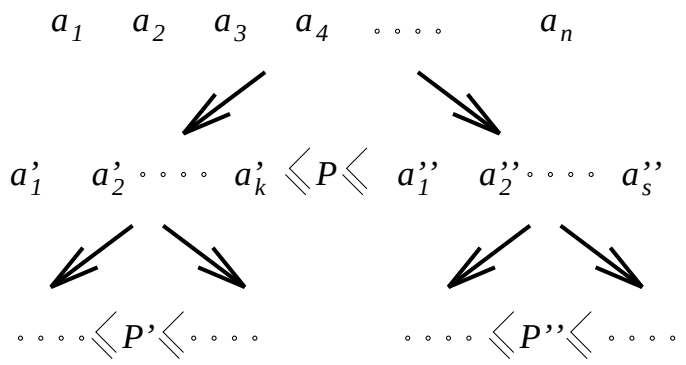
Parallelizing on a ring interconnection network.

  - One element/processor
    - Cost: $O(n^2)$
    - Time: $O(n)$
  - More element/processor

$$\text{time: } \underbrace{O(\tfrac{n}{p} \cdot \log \tfrac{n}{p})}_{\text{local sort}} + \underbrace{O(n)}_{\text{comp.}} + \underbrace{O(n)}_{\text{communic.}}$$

Quicksort (± the best!)

- **Algorithm**



  - Pivot element selection

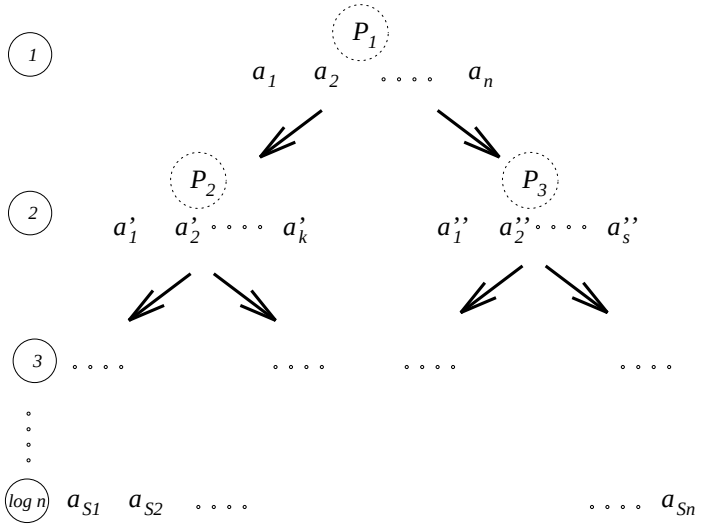
    when bad $O(n^2)$

    when good $O(n \cdot \log n)$

  - Randomly selected pivot

    $1.4n \cdot \log n$

- **Parallelization I. (novice attempt)**



Complexity (time): at least n !!

- **Parallelization II. (. . .)**
  - drawbacks
    a.) biggest comparison made by a single
    b.) comparisons performed sequentially
  - to achieve $O(n \cdot \log n)$:
    a.) steps $O(\log n)$
    b.) comparisons $O(n)$
    c.) with n processor $O(1)$!

- **Model**

  CRCW PRAM *(good example!)*

  Arbitrary selected pivot

  One element/processor

- **Steps**

  a.) all try to write root

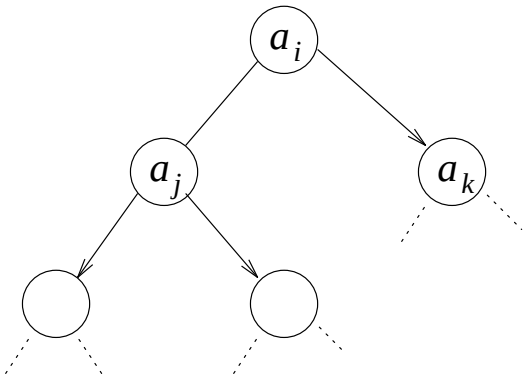
  b.) read the root

  c.) compare (the winner exits)

  a.) all try to write left/right child

  b.) read the left/right child

  c.) compare (the two succesful exists)



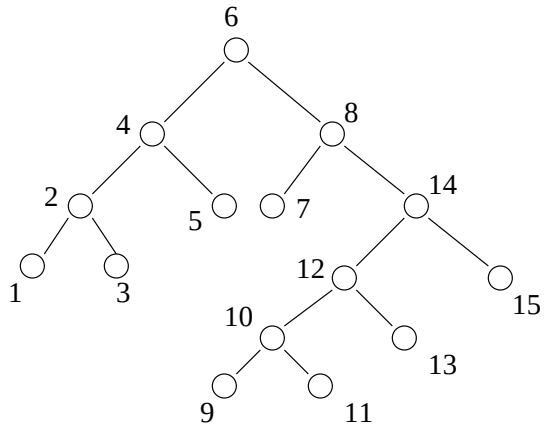
Child

Nr.	left	right
1	.....	.....
2	.....	.....
⋮		
i	j	k
⋮		

Processors store their parents in a variable $parent_i$.

The next tries to write the $left(parent_i)$ or $right(parent_i)$ based on the comparison with a_i.

- **Finishing**



Traversing the tree: $O(\log n)$ steps.

Placing the elements: $O(1)$ steps.

- **Complexity**

  Time: $O(\log n)$ (with optimal pivot)

  Cost: $O(n \cdot \log n)$

  Performance issue (a little bit more practical):

$O(\log^2 n)$	$\longleftrightarrow$	$O(n \log n)$
Bitonic		*Quick*
!	$n < 10^{22}$	

- **Mapping onto the Hypercube**

  Time: $O(\frac{n}{p} \log \frac{n}{p}) + O(\frac{n}{p} \cdot \log p) + O(\log^2 p)$