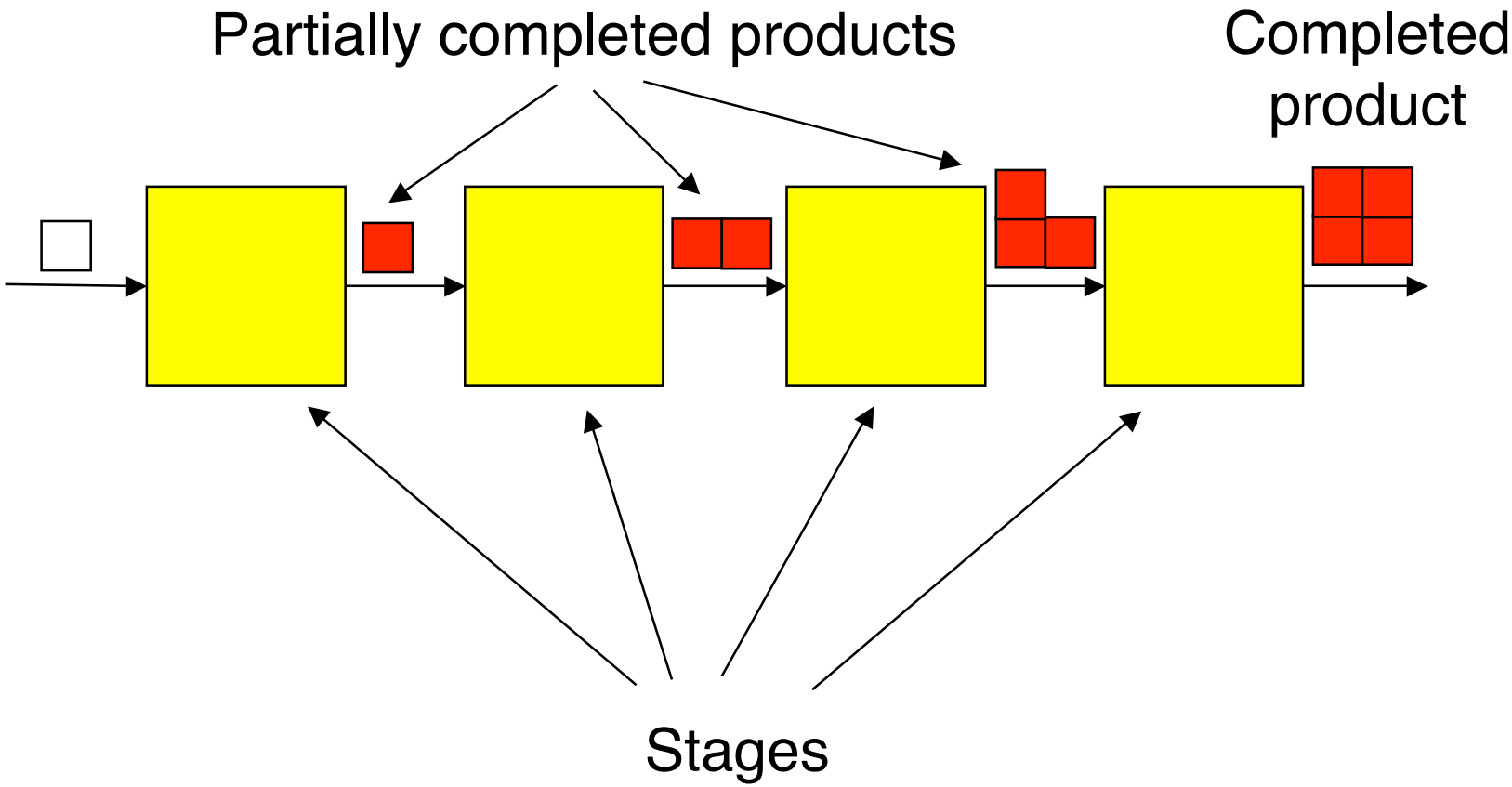
Pipelining

Pipelining Defined

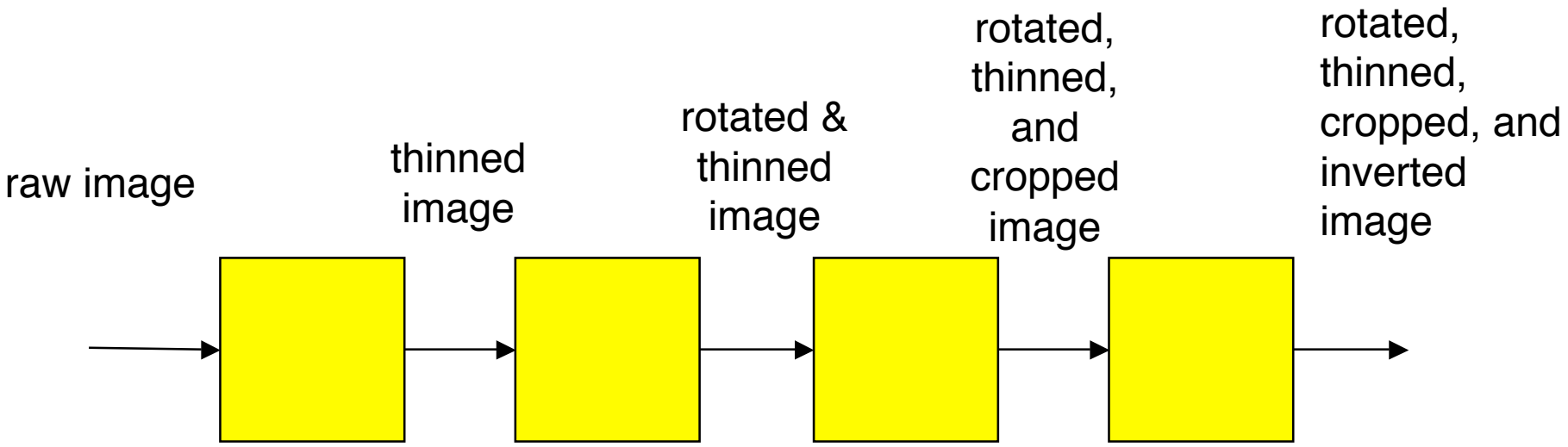
- A “product” in a stage of partial completion, is moved along through a series of “stations”, becoming more complete at each station.
- The stations operate in parallel on multiple products, each working on a product in its respective stage of completion.
- There may or may not be parallelism **within** a given station.

Pipelining



Examples of Computational Products

Product	Partially Complete Product
Number	Approximation to a number
Sorted array	Partially sorted array
Image	Transformed image



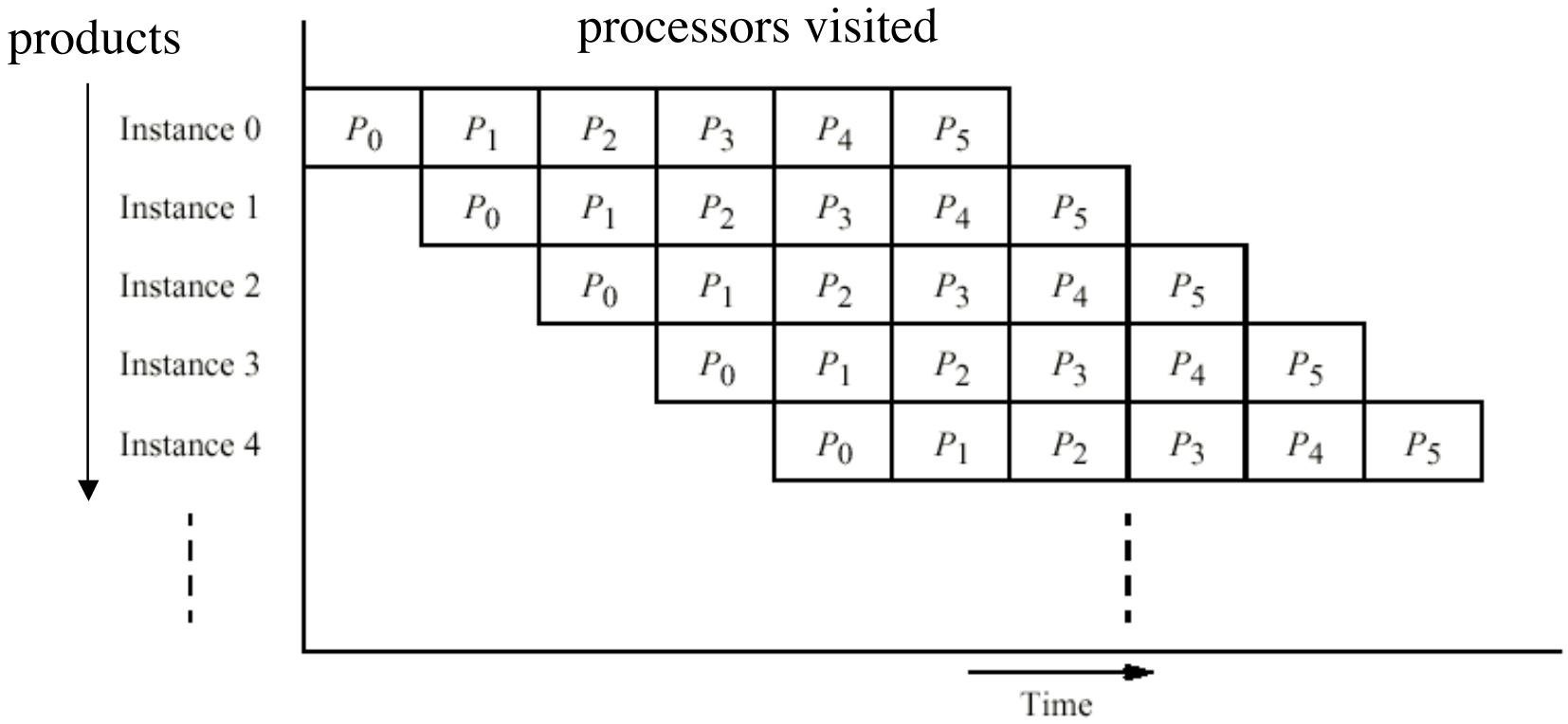
Utility of Pipelines

- Pipelines can provide a modular approach to constructing functions, rather than trying to invent or manage one single comprehensive function.

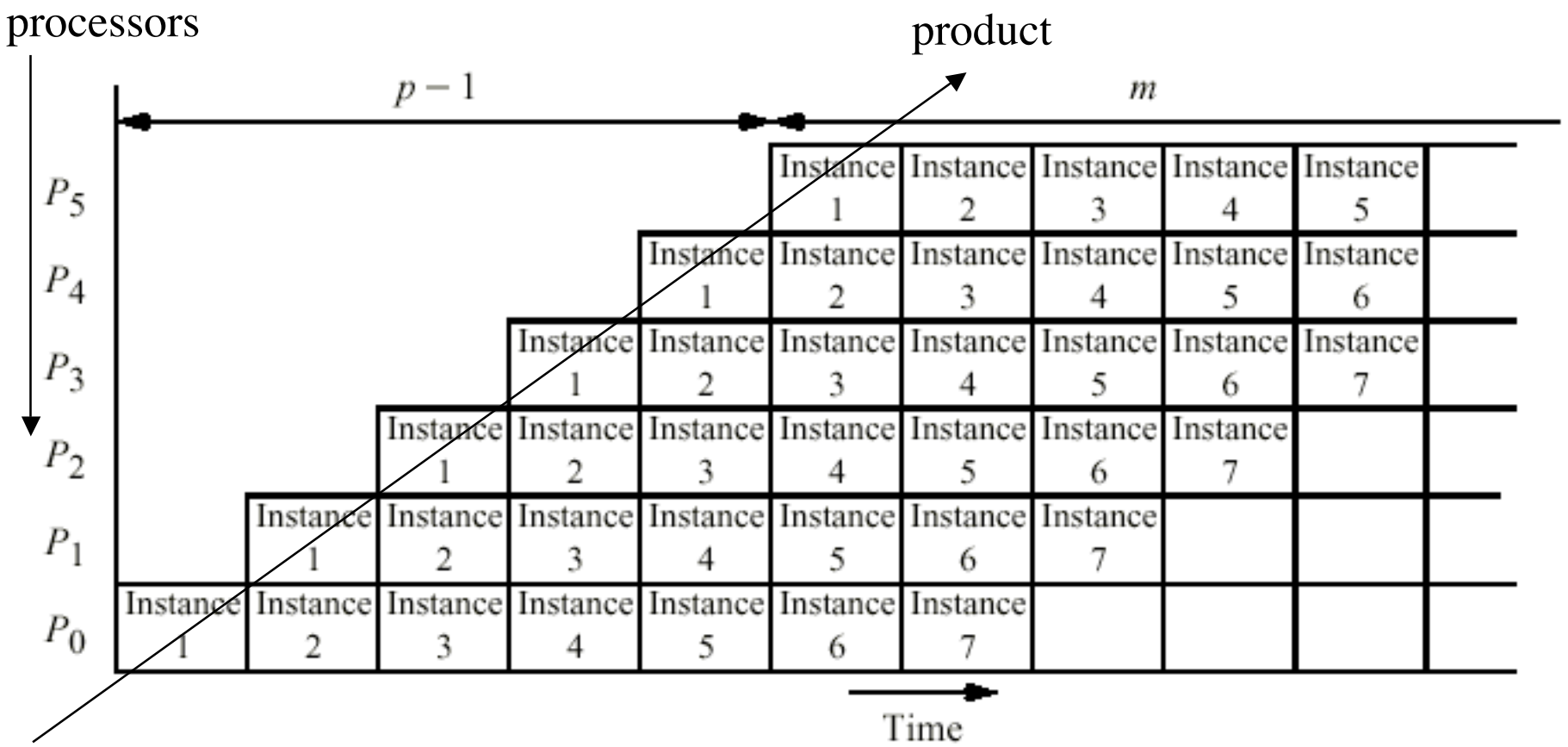
- Unix users are used to this kind of thing:

(thin < image) | rotate | crop | invert

Pipeline Timing from the products' view

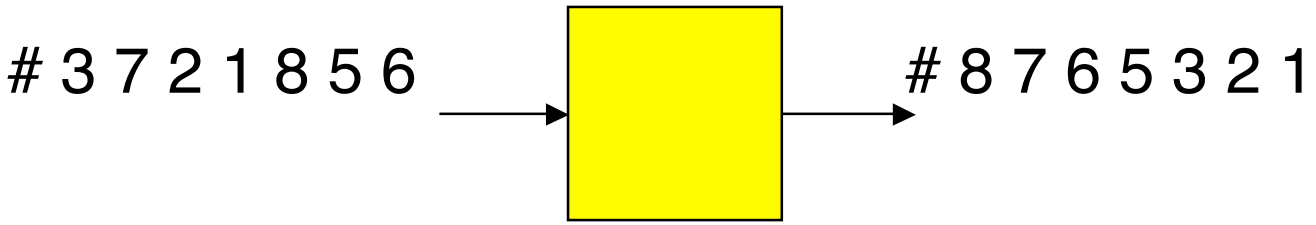


Pipeline Timing from the processors' view

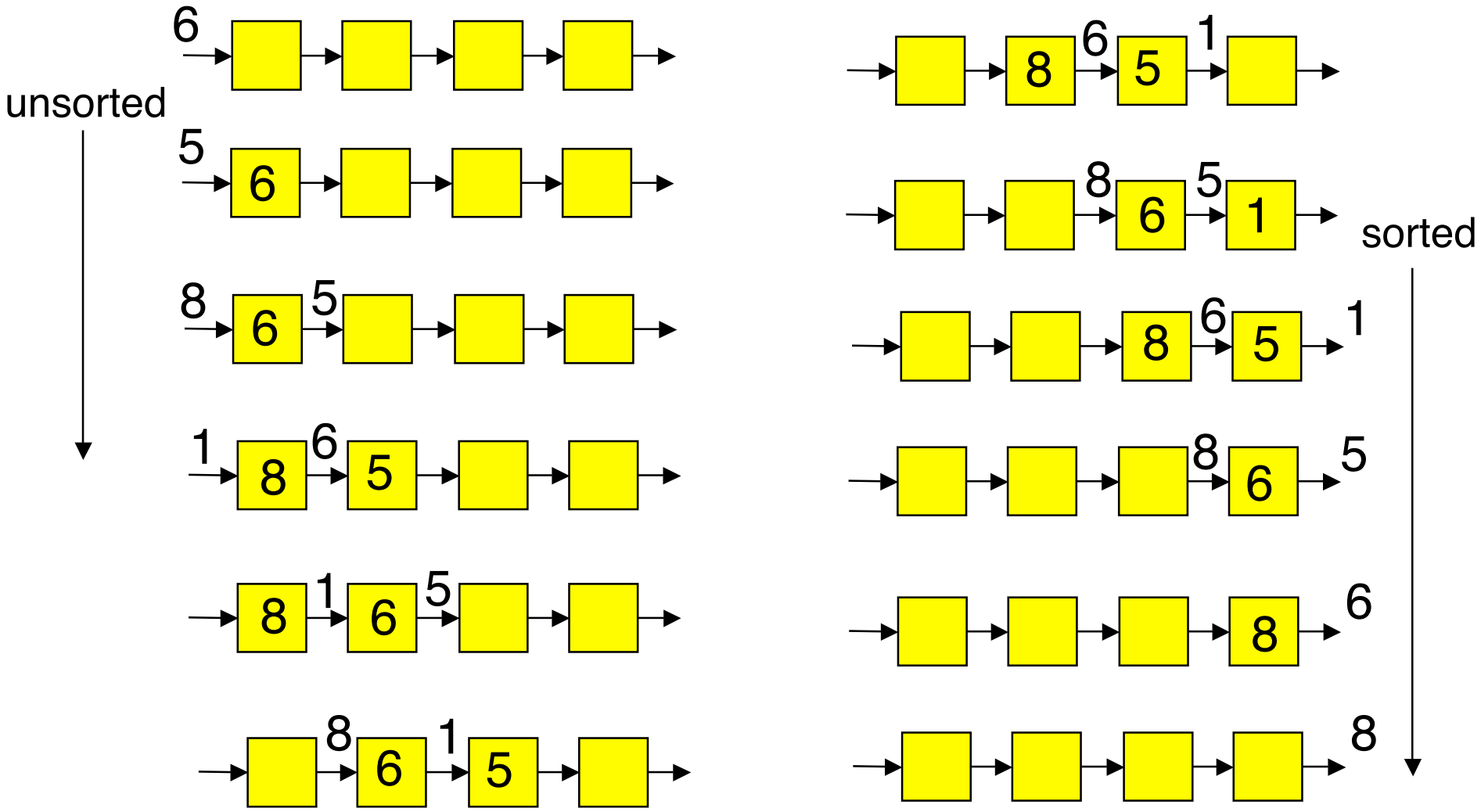


Distributed Pipelining
(my definition)

- In *distributed* pipelining, the result is not what comes out the last stage in one step, but rather the **collective sequence** of things coming out.



Example of Distributed Pipeline: Pipeline Sorting



Pipeline Sorting

- The method shown is O(N) and requires N processors (so is not cost-optimal).
- Also, the number of items to be sorted is limited by the number of processors.
- As soon as a sequence clears the first processor, the next sequence can be started, so N-1 processors can be kept busy.

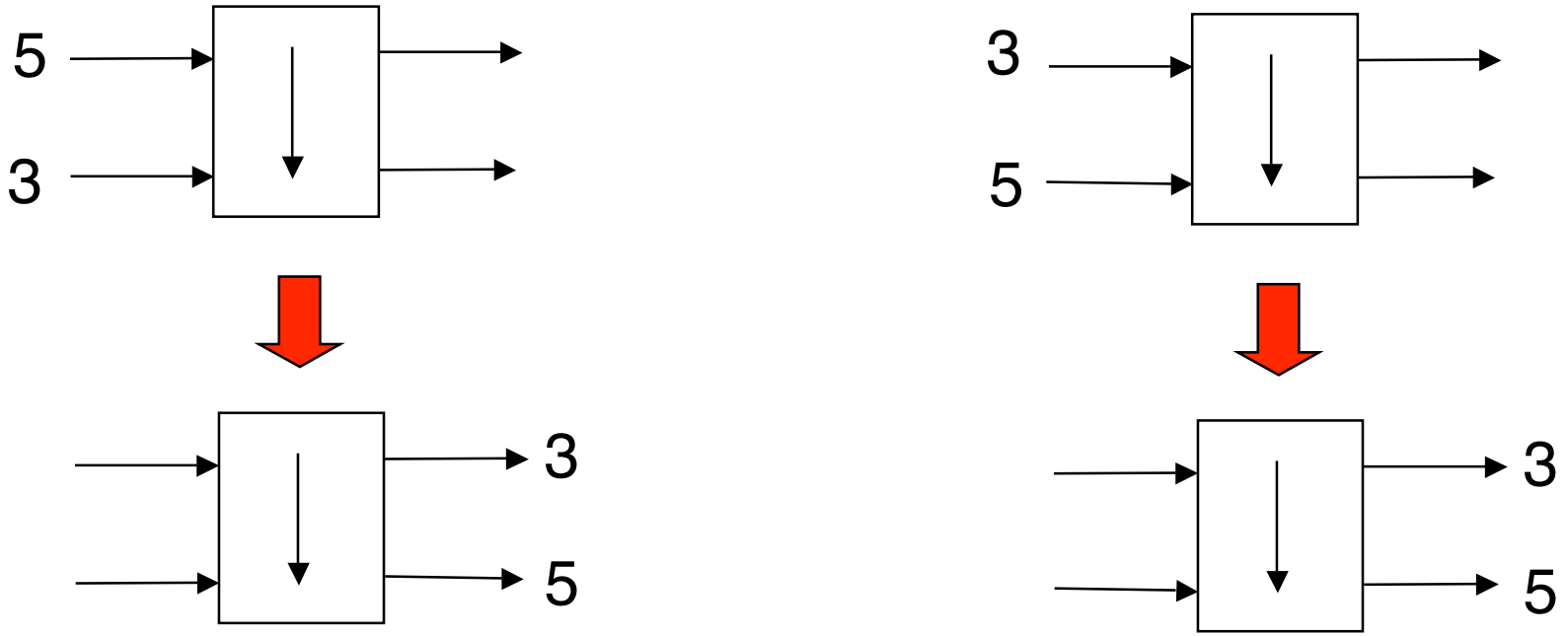
Pipelinable + Parallel Sorting Methods

- **Sorting networks** were first proposed by Kenneth Batcher in the 1960's.

- They are networks constructed from a number of simple devices, called **comparators**.

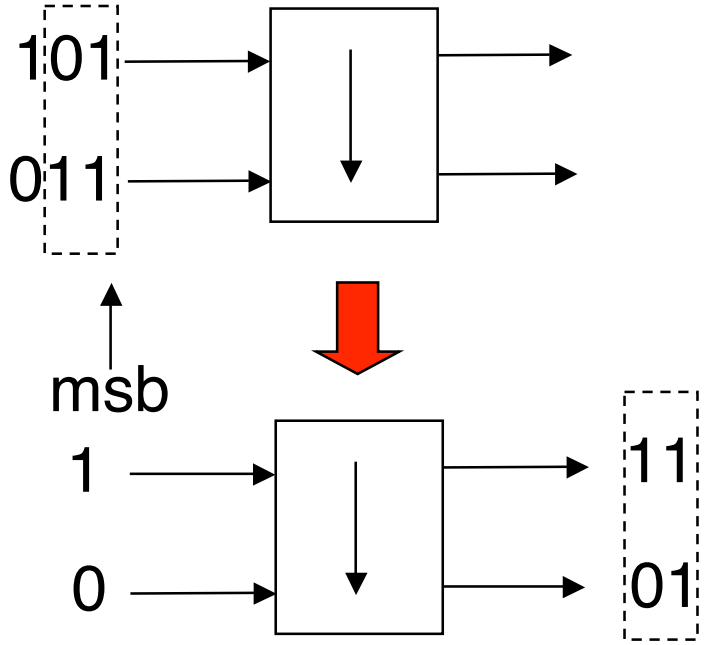
Sorting Network Comparator

- A single comparator inputs two numbers and outputs them in sorted order.



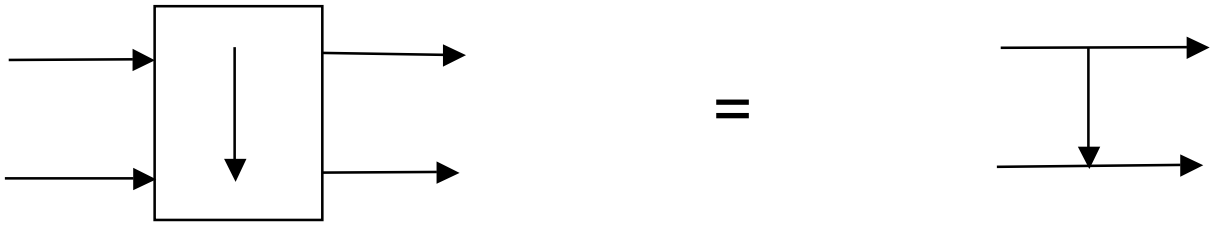
Sorting Network Comparator

- If desired, a comparator *can* be **pipelined** at the *bit-level*, as a finite-state machine accepting inputs MSB first (assuming the same number of bits in both numbers:

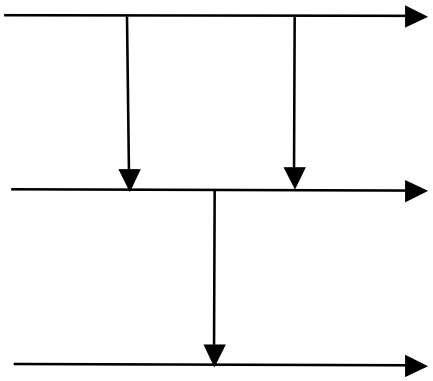


Sorting Networks

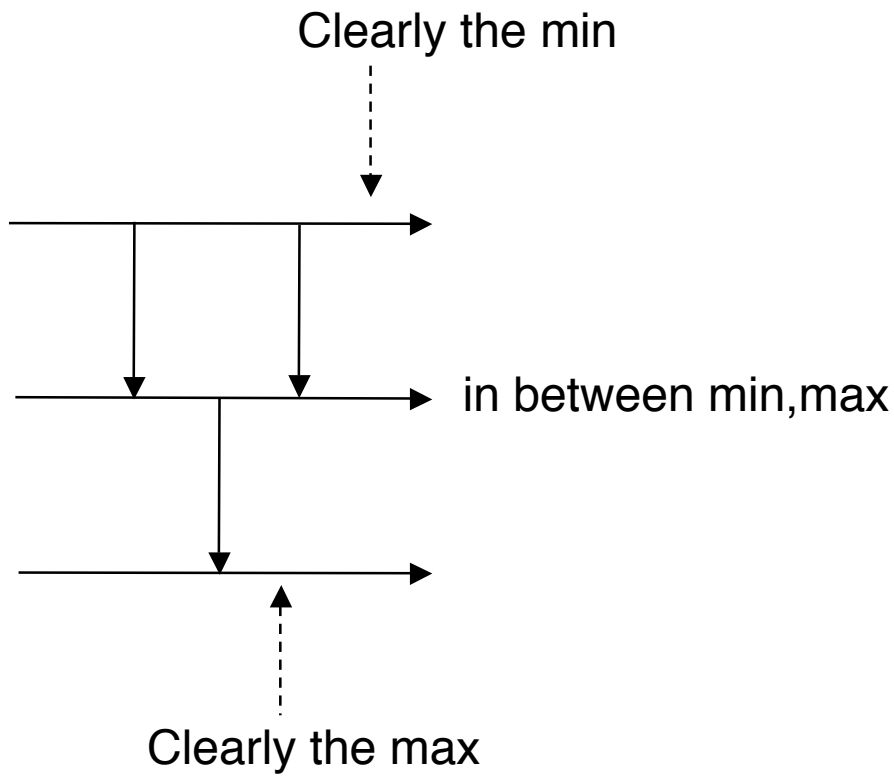
- Comparator abstraction (for drawing networks)



- A network that sorts 3 numbers



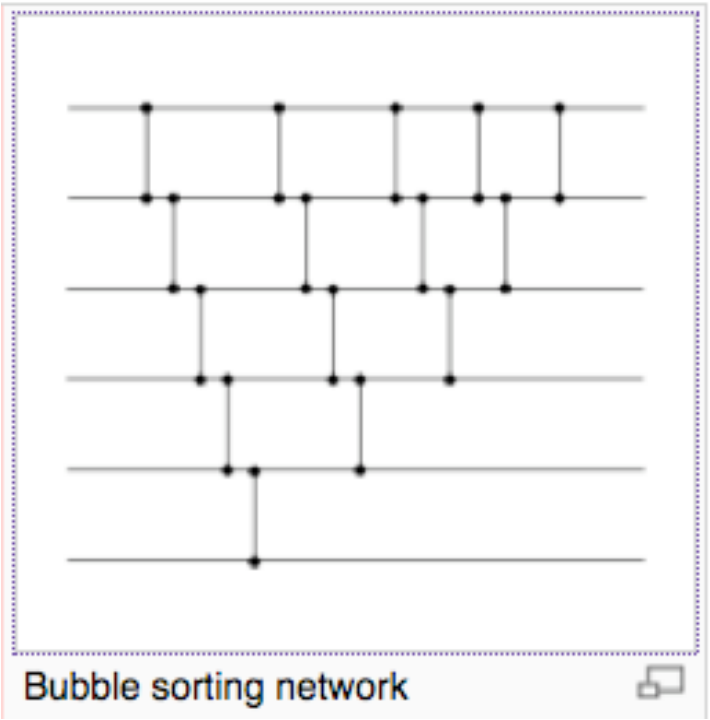
Analysis



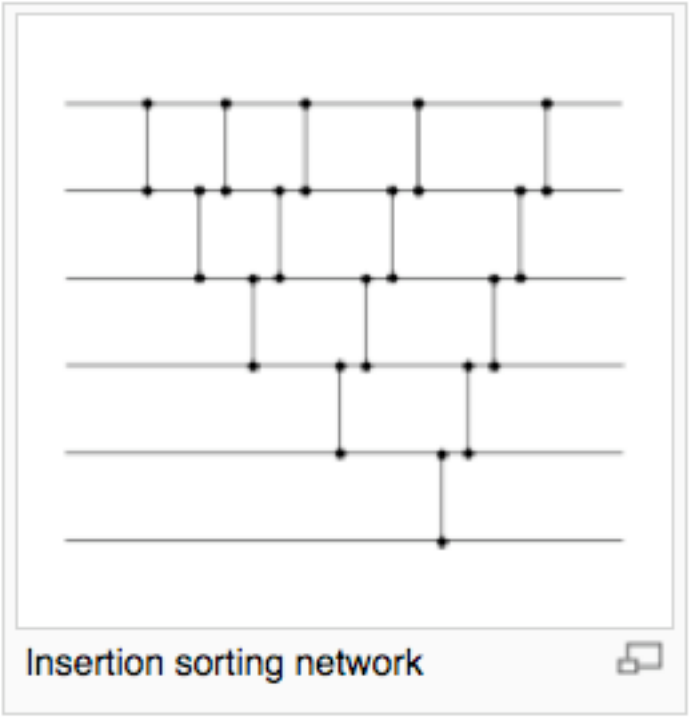
Exercises

- Construct a sorting network for 4 numbers
- Construct a sorting network for 8 numbers

(Non-Optimal) Sorters for Any Number of Lines



Bubble sorting network



Insertion sorting network

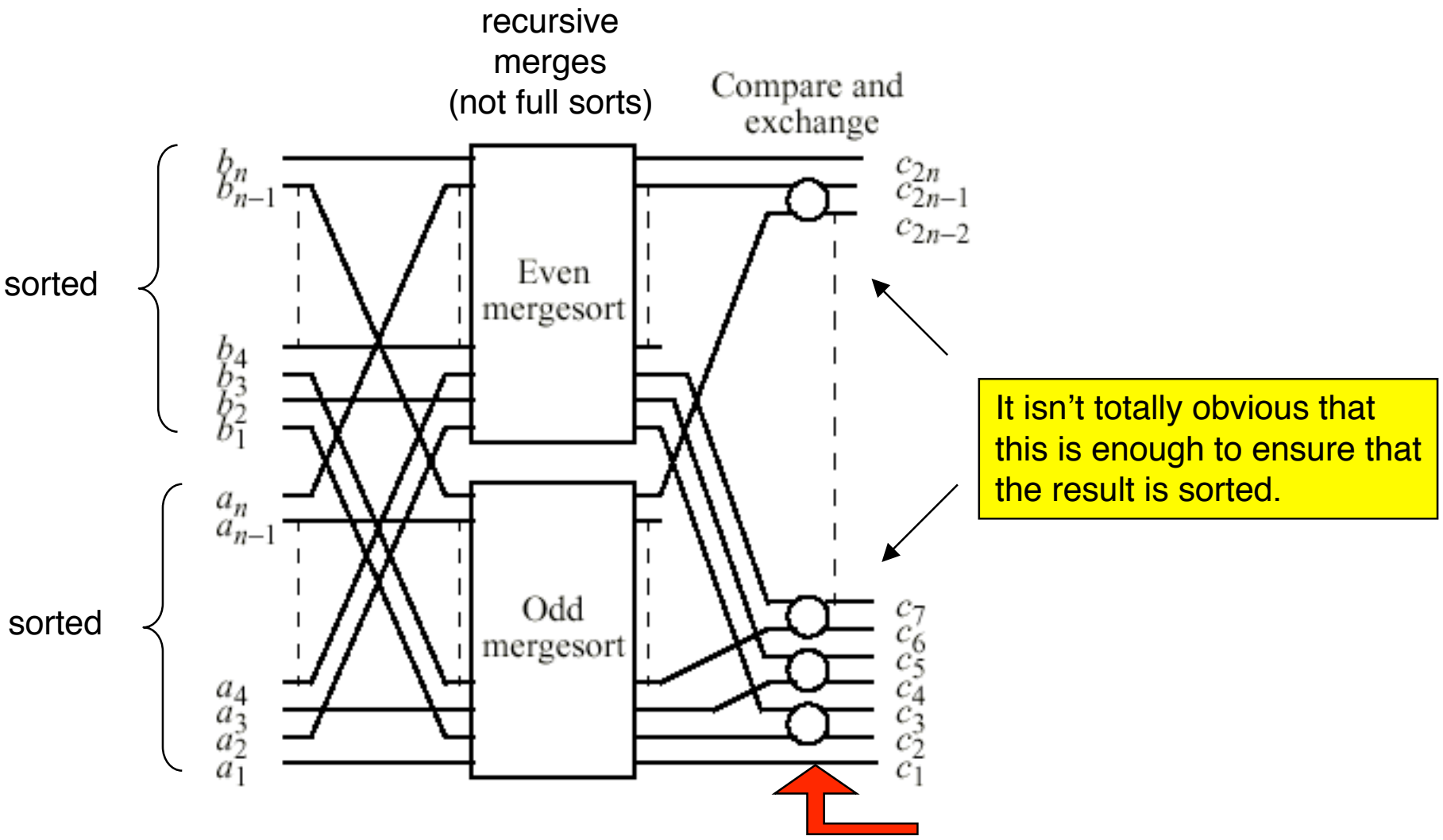
Superior Constructions

- Odd-Even Merging
  - Assume 2n elements to be sorted.
  - Split the elements into two groups.
  - Sort each half recursively.
  - Merge the odd-indexed components of the result.
  - Merge the even-indexed components of the result.

  (parallel)
  - Merge the output of the merges.

  (can be done in 1 stage)

Merging


recursive
merges
(not full sorts)
Compare and
exchange
sorted
sorted
b_n
b_{n-1}
b_4
b_3
b_2
b_1
a_n
a_{n-1}
a_4
a_3
a_2
a_1
Even
mergesort
Odd
mergesort
c_{2n}
c_{2n-1}
c_{2n-2}
c_7
c_6
c_5
c_4
c_3
c_2
c_1
It isn't totally obvious that this is enough to ensure that the result is sorted.

Timing Analysis of Sorting by Odd-Even Merging

- Let S(n) = time to sort n elements
- Let M(n) = time to merge n/2 with n/2 elements
- Recurrences:
  - S(1) = 0
  - S(2n) = S(n) + M(2n)
  - M(2) = 1
  - M(2n) = M(n) + 1
- assuming we don't charge for splitting into 2 groups

Timing Analysis of Sorting by Odd-Even Merging

- $M(2) = 1$
- $M(2n) = M(n) + 1$

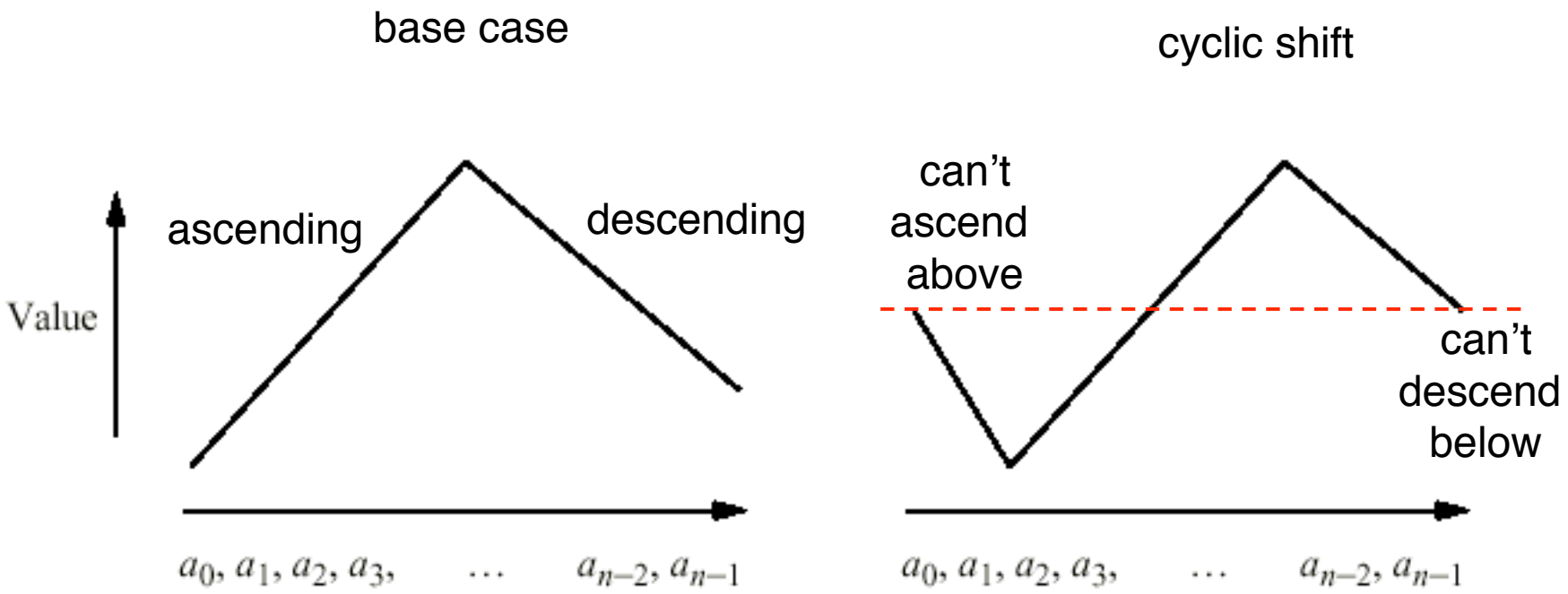
- Therefore
  - $M(2^n) = n-1$
  - $S(2^n) = S(2^{n-1}) + n-1$
- Giving $S(2^n) = (n-1) + (n-2) + \ldots + 1$

or $\boxed{S(N) = O(\log^2 N)}$

A Different Construction

- Bitonic sorting
- Define a *bitonic* sequence to be one in which either:
  - There is a strictly ascending portion, followed by a strictly descending portion, OR
  - is a **cyclic shift** of a sequence with the above property.

Bitonicity


base case
cyclic shift
Value
ascending
descending
can't ascend above
can't descend below
$a_0, a_1, a_2, a_3, \ldots a_{n-2}, a_{n-1}$
$a_0, a_1, a_2, a_3, \ldots a_{n-2}, a_{n-1}$

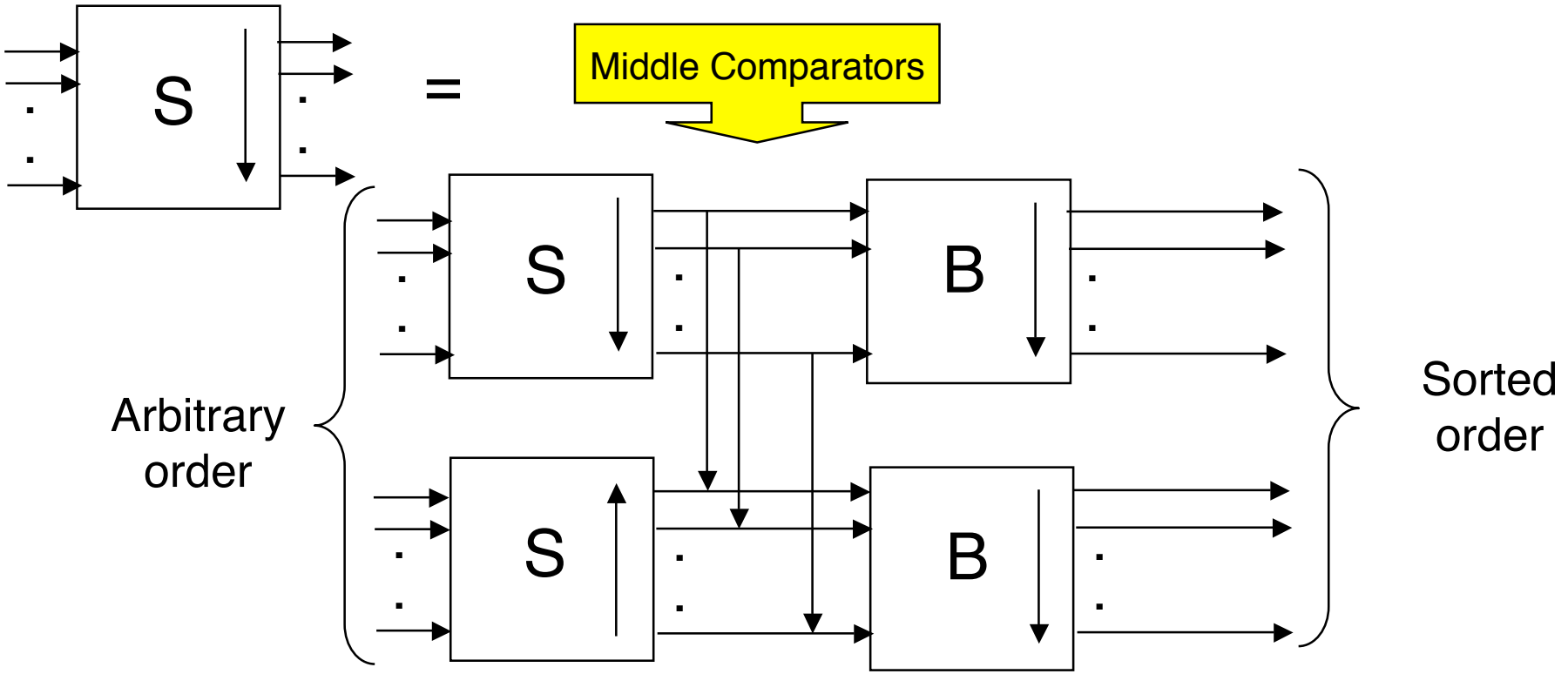
Bitonic Sorting

- Two *sorted* sequences can be made into a bitonic sequence by reversing the second one and abutting them.

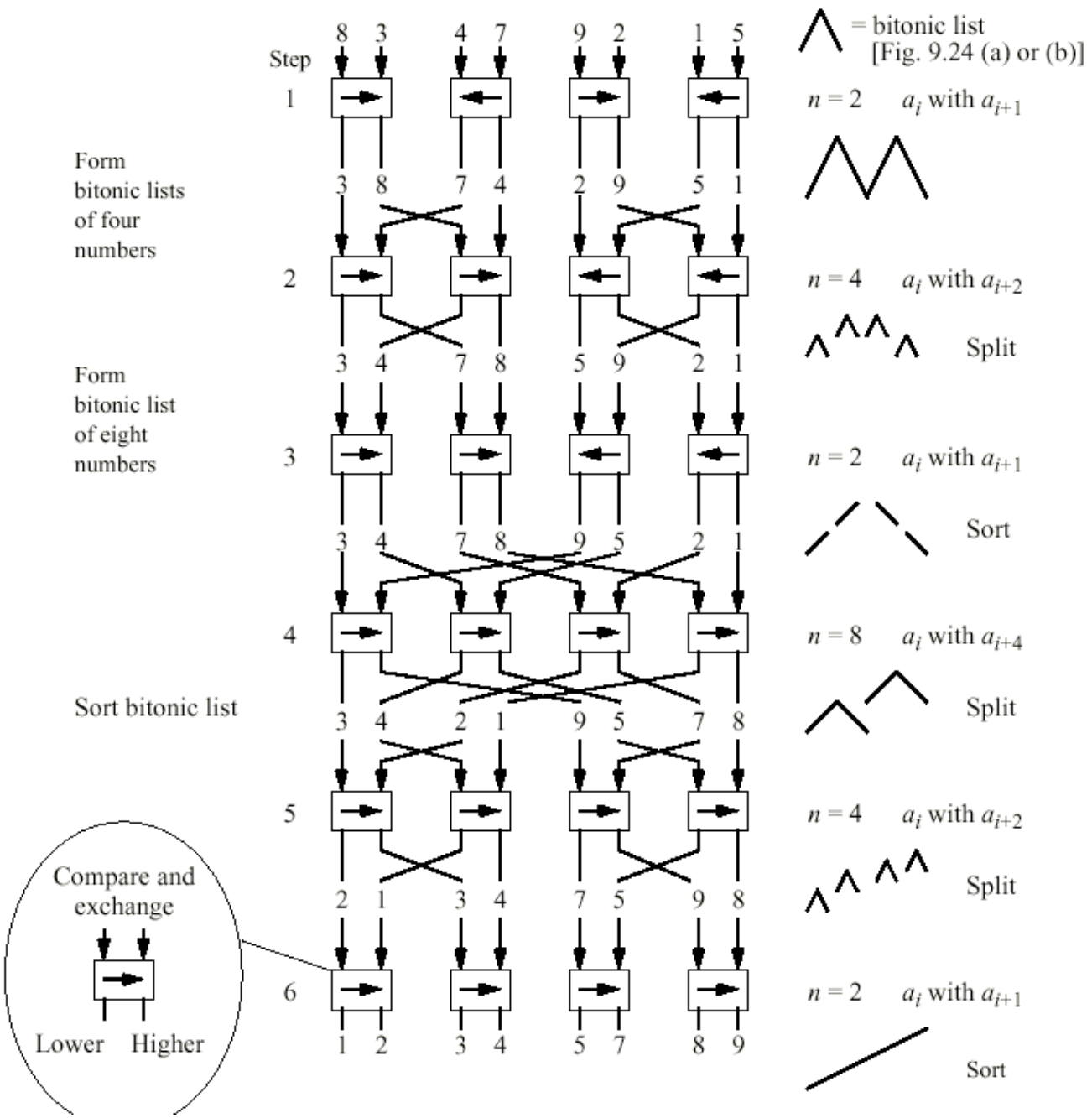
Bitonic Sorting

- A bitonic sequence can be sorted by the following scheme:
  - Do compare-exchanges between elements j and j+n/2 for j = 1, 2, …, n/2
  - This gives two sequences that:
    - are both bitonic
    - all elements of one are $\leq$ all elements of the other
  - Repeating this scheme recursively with the resulting two bitonic sequences gives a sorted sequence

Bitonic Sort Recursion



S = general sorter B = sorter for bitonic sequence


Step
1
2
3
4
5
6
8 3 4 7 9 2 1 5
3 8 7 4 2 9 5 1
3 4 7 8 5 9 2 1
3 4 7 8 9 5 2 1
3 4 2 1 9 5 7 8
2 1 3 4 7 5 9 8
1 2 3 4 5 7 8 9
Form bitonic lists of four numbers
Form bitonic list of eight numbers
Sort bitonic list
Compare and exchange
Lower
Higher
= bitonic list [Fig. 9.24 (a) or (b)]
n = 2 a_i with a_{i+1}
n = 4 a_i with a_{i+2}
Split
n = 2 a_i with a_{i+1}
Sort
n = 8 a_i with a_{i+4}
Split
n = 4 a_i with a_{i+2}
Split
n = 2 a_i with a_{i+1}
Sort

Exercise

- Comment on the pipelinability of bitonic sorting.
- Analyze the time taken for bitonic sorting.

Other facets of sorting networks

- The **number** of comparators for odd-even merging is $O(n \log^2 n)$, which is typical of the most accessible constructions.
- An upper bound of $O(n \log n)$ comparators and $O(\log n)$ time has been shown, but the constants are large (in the 1000's) [AKS network after its discoverers Ajtai, Komlós, and Szemerédi].
- See http://en.wikipedia.org/wiki/Sorting_network