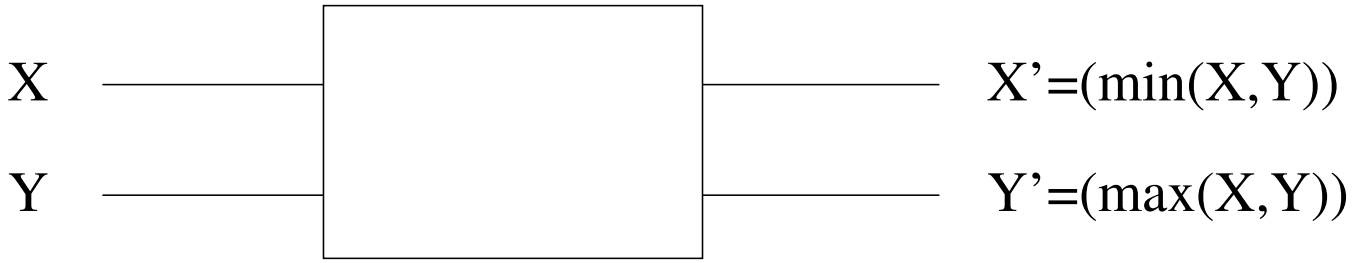
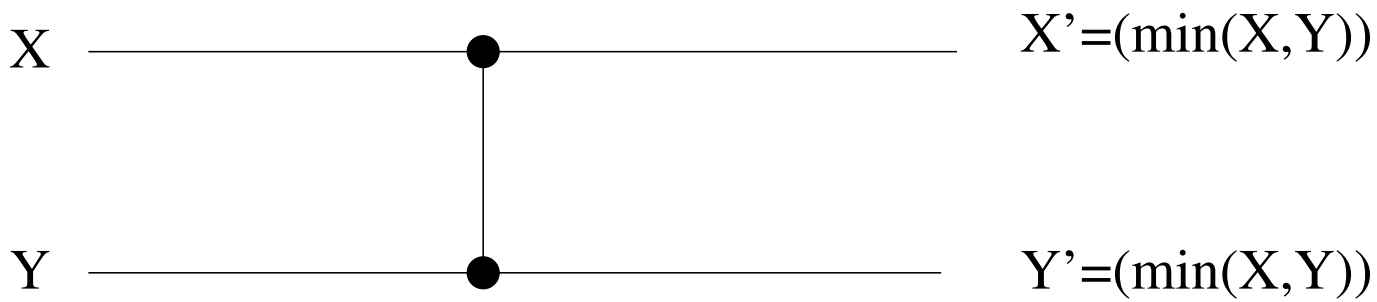
Sorting Network

- Previous mentioned sorting algorithms: one processor (one CPU), under RAM model, serial computation.
- What if there are more than one processor? What if there are many processors?
- *parallel computing*, many comparisons are performed simultaneously.
- What is the performance? If one man can build a house in 100 days. Is that correct that it takes only one day for 100 men to build the same house? Why? Why not?

Comparison Network, Sorting Network

- A network of processors, each processor can perform only comparison.
- Comparison Network: Allow comparisons occur in parallel.
- Sorting Network: A comparison network, sort its inputs.
- Only perform comparison, doesn't work for counting sort.





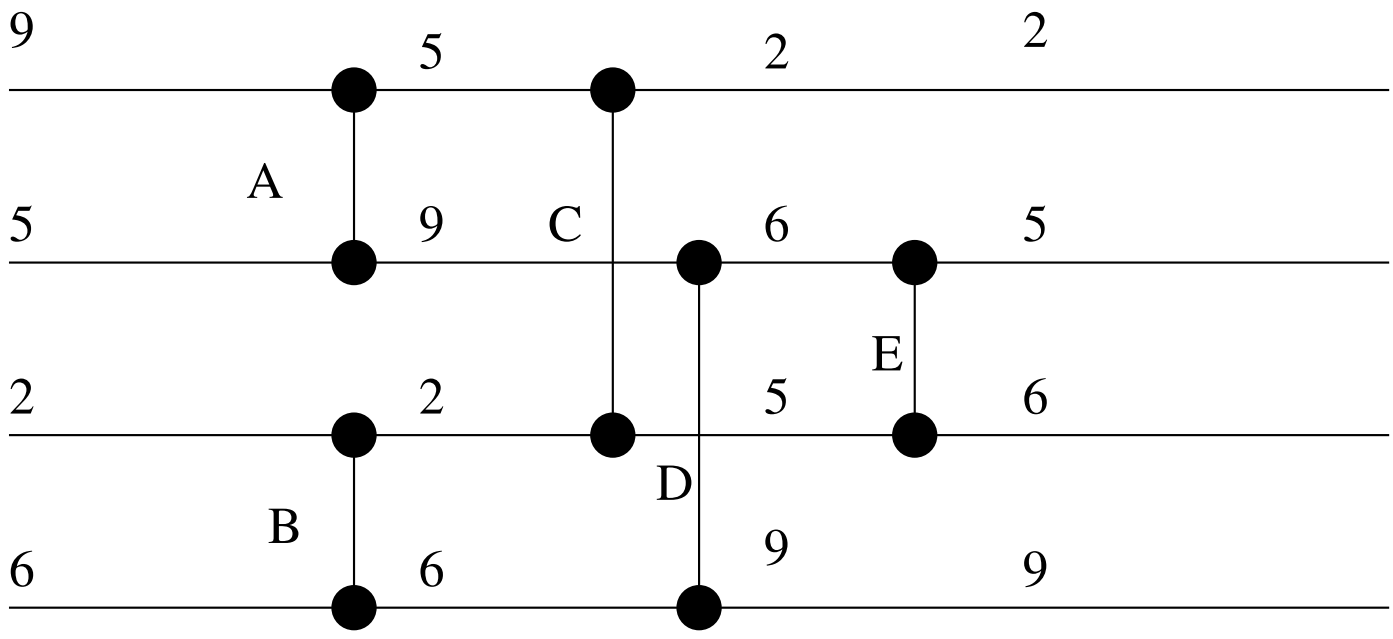
A comparator, a comparison is done in constant time.

Wire

- Input wire
- Output wire
- Connecting output of a comparator to an input of another comparator.

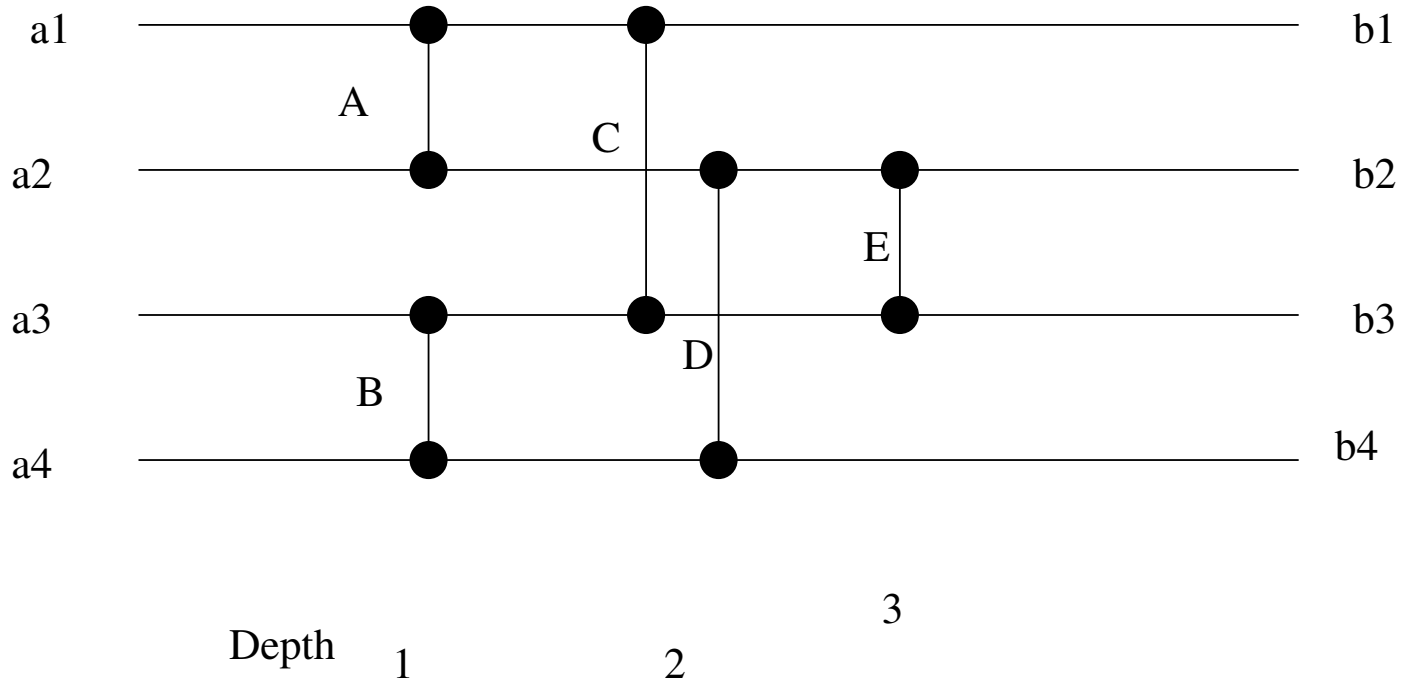
Comparison Network

- There are n input wires a_1, a_2, a_3, ..., a_n, and
- n output wires b_1, b_2, b_3, ..., b_n,
- Feed in from the input, data run through the network and reach the output.
- the interconnection network, an acyclic graph (no cycle).


9
5
2
2
A
5
9
C
6
5
E
2
2
5
6
D
B
9
6
6
9

Computing Time

- Defined by the depth of wires.
- Each comparator operates when all (two) inputs are available.
- The network finishes computation when all outputs receive results.
- Define the depth of a wire: inputs wires are depth 0, a comparator has input wire depth d_x and d_y, the output wires has depth $\max(d_x, d_y) + 1$


a1
a2
a3
a4
b1
b2
b3
b4
A
B
C
D
E
Depth
1
2
3

Computing Time

- Since there is no cycle, depth is well defined.
- the maximum depth of an output wire is the depth of the network.
- the computing time is the depth of the network.

Sorting Network

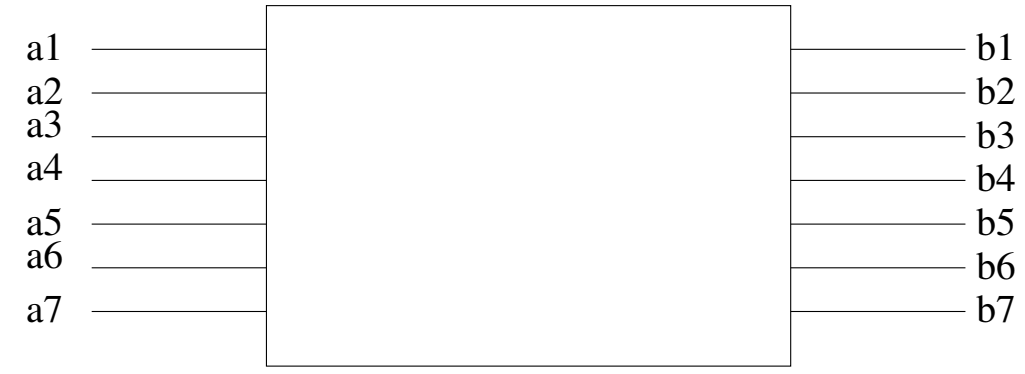
- A comparison network, output sequence is monotonically increasing
- $(b_1 \leq b_2 \leq b_3 \leq ... \leq b_n)$ for every input sequence. (Convert an algorithm to hardware.)
- The above comparison network is a sorting network.
- Depth is 3.
- Our goal in this chapter: Design Sorter[n]: a sorter (a sorting network) that sorts n numbers.

The Zero-One Principle

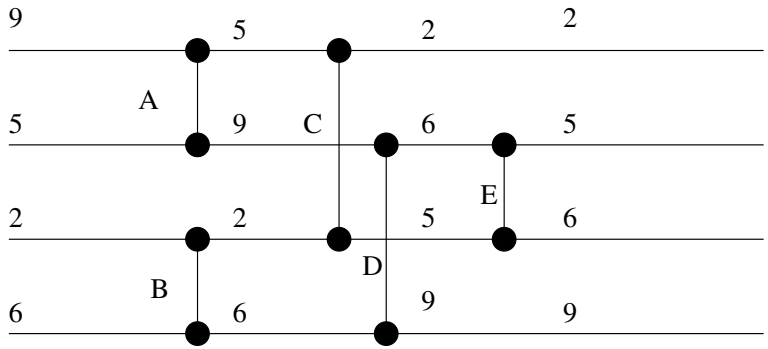
- If a sorting network works correctly when each input is drawn from the set $\{0,1\}$, then it works correctly on arbitrary input numbers (int, float, or any set of values from any linearly ordered set).
- By the zero-one principle, we can focus on their operation for input sequences consisting solely of 0's and 1's.

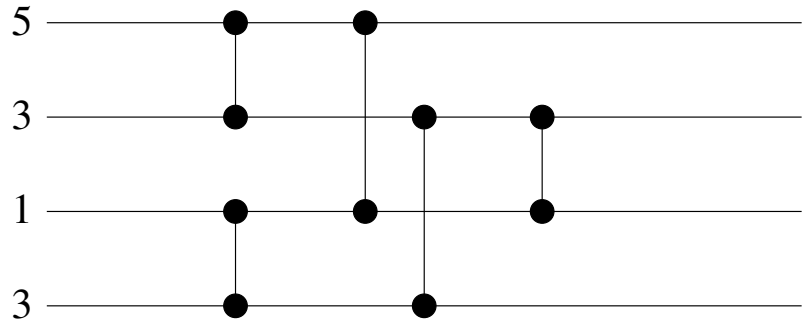
A monotonically increasing function, f, if $x \geq y$ then $f(x) \geq f(y)$.

Lemma If a comparison network transforms the input sequence $a = < a_1, a_2, \ldots, a_n >$ into output sequence $b = < b_1, b_2, \ldots, b_n >$, then for any *monotonically increasing* function f, the network transforms the input sequence
$f(a) = < f(a_1), f(a_2), \ldots, f(a_n) >$ into the output sequence $f(b) = < f(b_1), f(b_2), \ldots, f(b_n) >$.


a1
a2
a3
a4
a5
a6
a7
b1
b2
b3
b4
b5
b6
b7


f(a1)
f(a2)
f(a3)
f(a4)
f(a5)
f(a6)
f(a7)
f(b1)
f(b2)
f(b3)
f(b4)
f(b5)
f(b6)
f(b7)

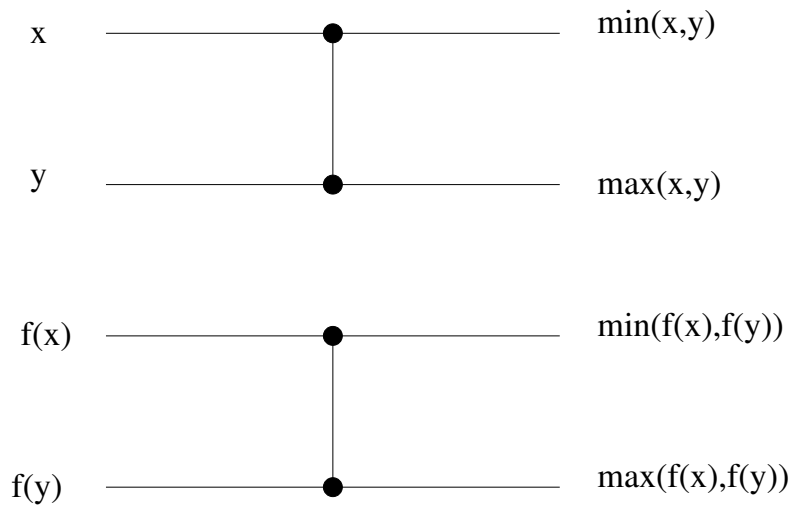




The function f is $\lceil \frac{x}{2} \rceil$

Proof of the Lemma

First show that if f is a monotonically increasing function, a single comparator with input $f(x)$ and $f(y)$ produces output $f(\min(x, y))$ and $f(\max(x, y))$.



Proof of the Lemma

Since f is monotonically increasing, by definition $x \leq y$ implies $f(x) \leq f(y)$. Consequently $\min(f(x), f(y)) = f(\min(x, y))$ and $\max(f(x), f(y)) = f(\max(x, y))$.

Proof of the Lemma

Secondly, we show that

if the wire assumes the value a_i when the input sequence a is applied to the network, then it assumes the value $f(a_i)$ when the input sequence $f(a)$ is applied. (Note output wires are included in the statement, prove the statement prove the lemma.)

By induction on the depth
For the basis, consider a wire at depth 0, input wire has a_i; if $f(a_i)$ is applied, input carries $f(a_i)$.

Inductive step, consider a wire at depth d, $d \geq 1$, the wire is the output of a comparator at depth d, the input wires to this comparator at a depth strictly less than d.
By induction hypothesis, if the input wires to the comparator carry a_i and a_j when the input sequence a applied, then they carry $f(a_i)$ $f(a_j)$ when input sequence $f(a)$ is applied.
By earlier claim, the output wire of the comparator then carry $f(\min(a_i, a_j))$ and $f(\max(a_i, a_j))$.
Since they carry $\min(a_i, a_j)$ and $\max(a_i, a_j)$ when the input sequence is a, the lemma proved.

Theorem (Zero-One Principle)

If a comparison network with n inputs sort all 2^n possible sequence of 0's and 1's, then it sorts all sequences of arbitrary number correctly.

Proof By contradiction
Suppose that the network sorts all zero-one sequence but there exists a sequence of arbitrary numbers that the network does not correctly sort them.
$\exists$ a sequence $< a_1, a_2, \ldots, a_n >$ containing elements a_i and a_j s.t. $a_i < a_j$ but the network places a_j before a_i in the output sequence.

Define a monotonically increasing function f

$$f(x) = \begin{cases} 0 & \text{if} \quad x \leq a_i \\ 1 & \text{if} \quad x > a_i \end{cases}$$

Since the network places a_j before a_i in the output when $< a_1, a_2, \ldots, a_n >$ is input.
Since $f(a_j) = 1$ and $f(a_i) = 0$, we obtain the contradiction that the network fails to sort the zero-one sequence $< f(a_1), f(a_2), \ldots, f(a_n) >$ correctly.

Bitonic Sorting Network

- A comparison network that can sort any bitonic sequence.
- Bitonic sequence:
  - A sequence monotonically increasing then monotonically decreasing,
  - or monotonically decreasing then monotonically increasing,
  - or monotonically increasing or decreasing sequences
  - For zero-one sequence $0^i1^j0^k$ or $1^i0^j1^k$.

The Half Cleaner

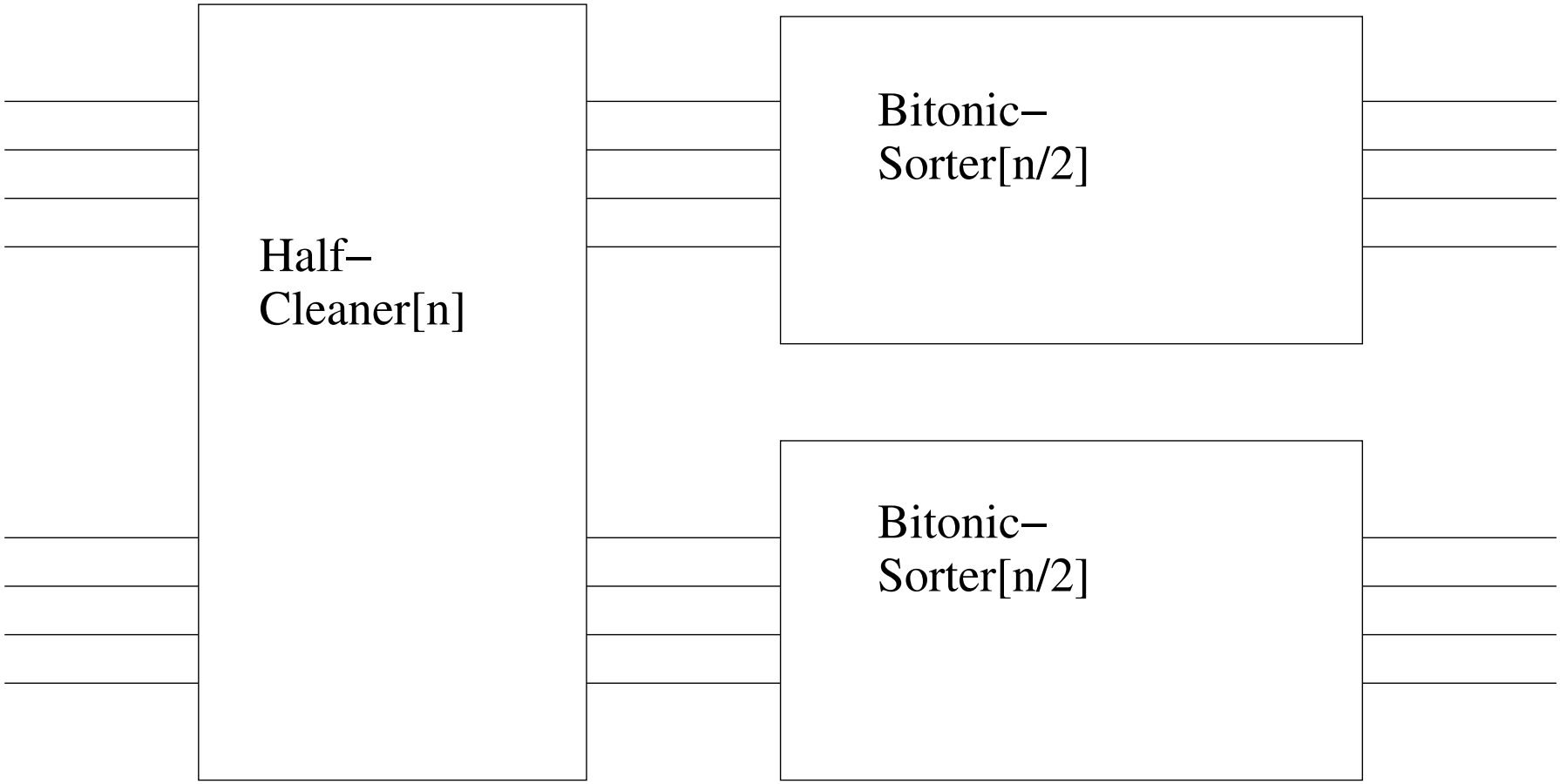
- Several stages in a bitonic sorter, each of which is a half-cleaner.
- A half-cleaner, a comparison network of depth 1. There are n input wires, i is compared with $i = \frac{n}{2} + i$, for $i = 1, \ldots, \frac{n}{2} - 1$.
- Input sequence is bitonic, output sequence is (1) smaller values are in the top half, (2) larger values are in the bottom half, (3) Both halves are bitonic, (4) one of the halves is clean (only 0 or 1).

An example

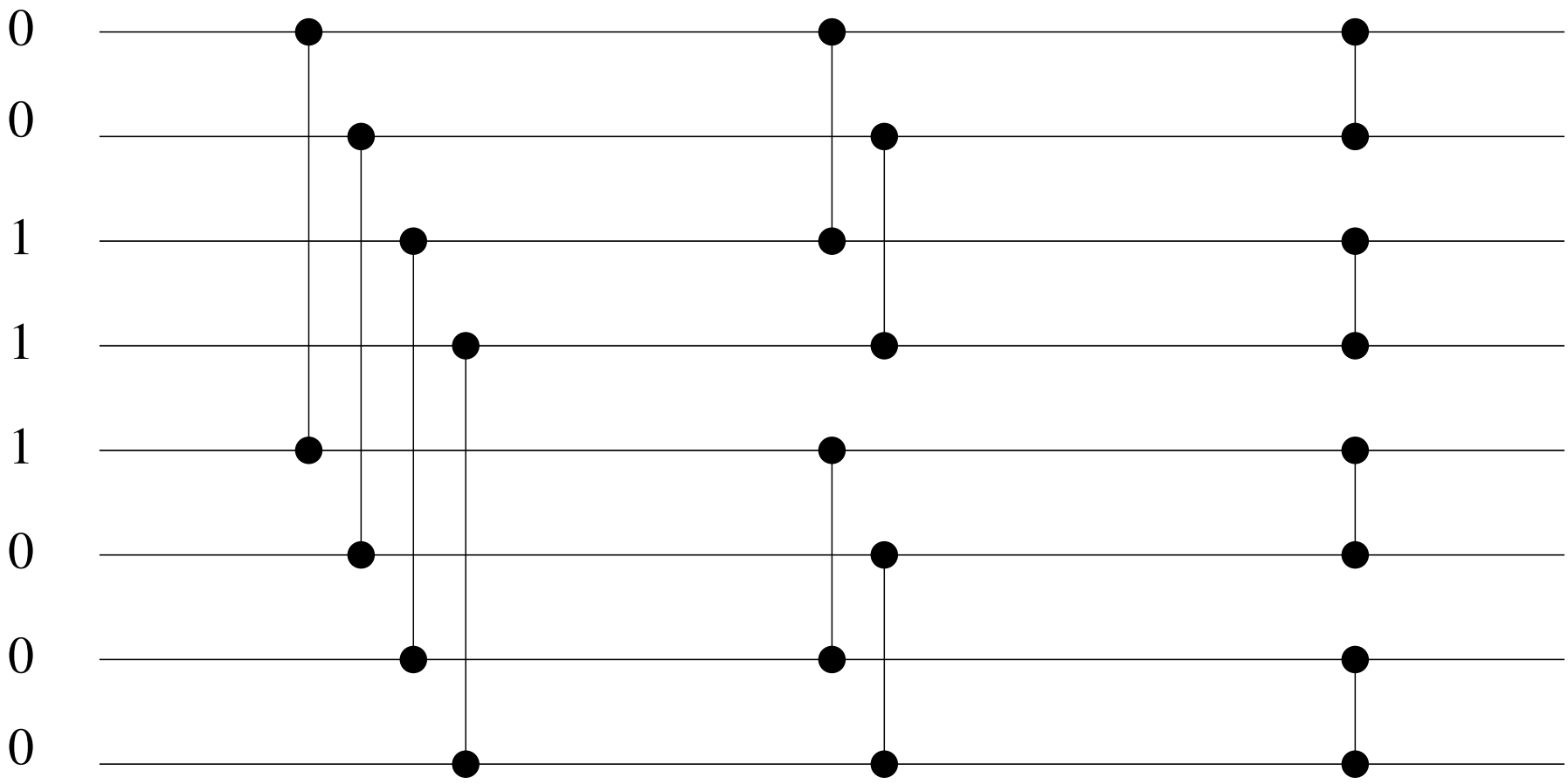
To prove the above statement, assume the input if 00...011...10...00. The other case is symmetric. Cases depend on where $\frac{n}{2}$ is.

Bitonic Sorter

- A network that sorts bitonic sequence.
- Bitonic-Sorter[n], input is bitonic sequence of length n, output is a sorted list.



a Bitonic-Sorter[n]



a Bitonic-Sorter[8]

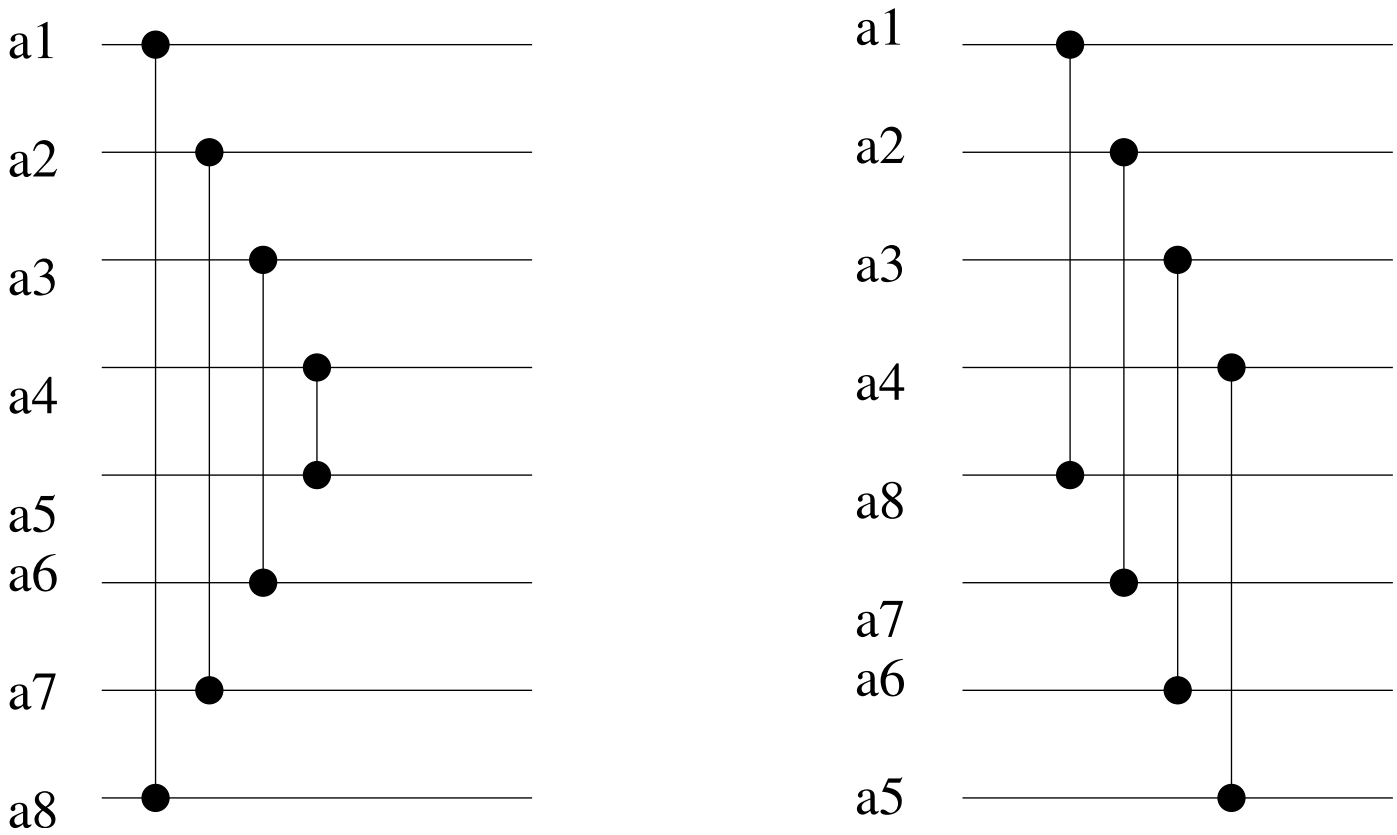
Depth of Bitonic-Sorter[n]

The depth is denoted $D(n)$

$$D(n) = \begin{cases} 0 & \text{if} \quad n = 1, \\ D(\frac{n}{2}) + 1 & \text{if} \quad n = 2^k, \quad k \geq 1, \end{cases}$$

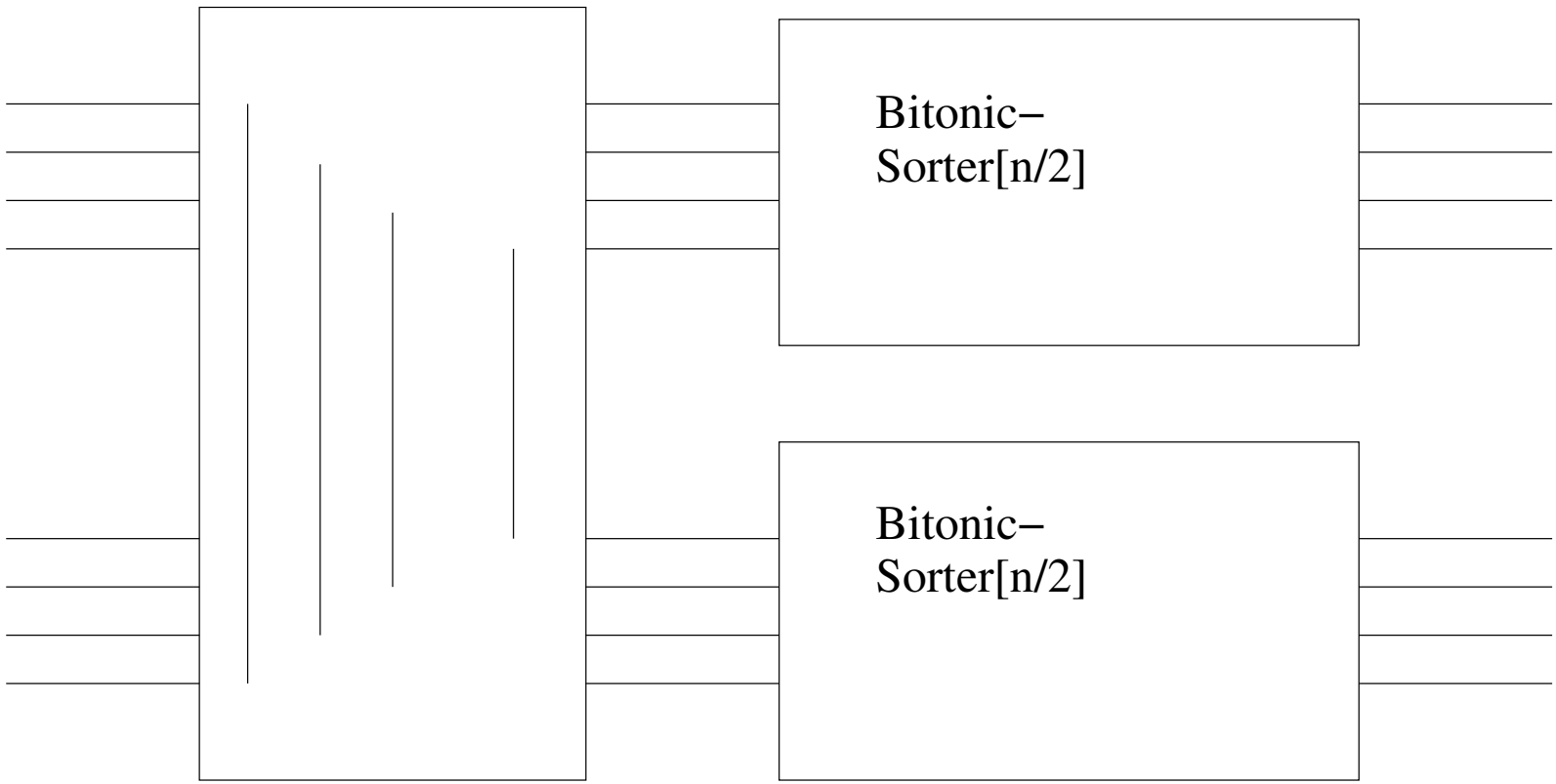
Merging Network

- Merge[n]: Merge two sorted sequences to get the whole sorted sequences.
- Modify from Bitonic Sorter, Bitonic-Sorter[n].
- Given two sorted zero-one sequences X and Y, XY^R is Bitonic where Y^R is reverse sequence of Y.
- Thus Bitonic Sorter sorts XY^R.

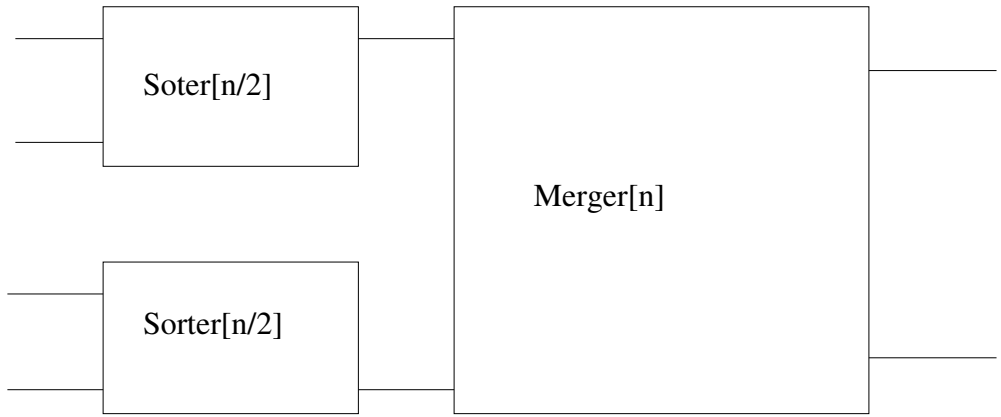


Comparing the first stage of the Merge[n] (left) and Bitonic-Sorter[n] (right)

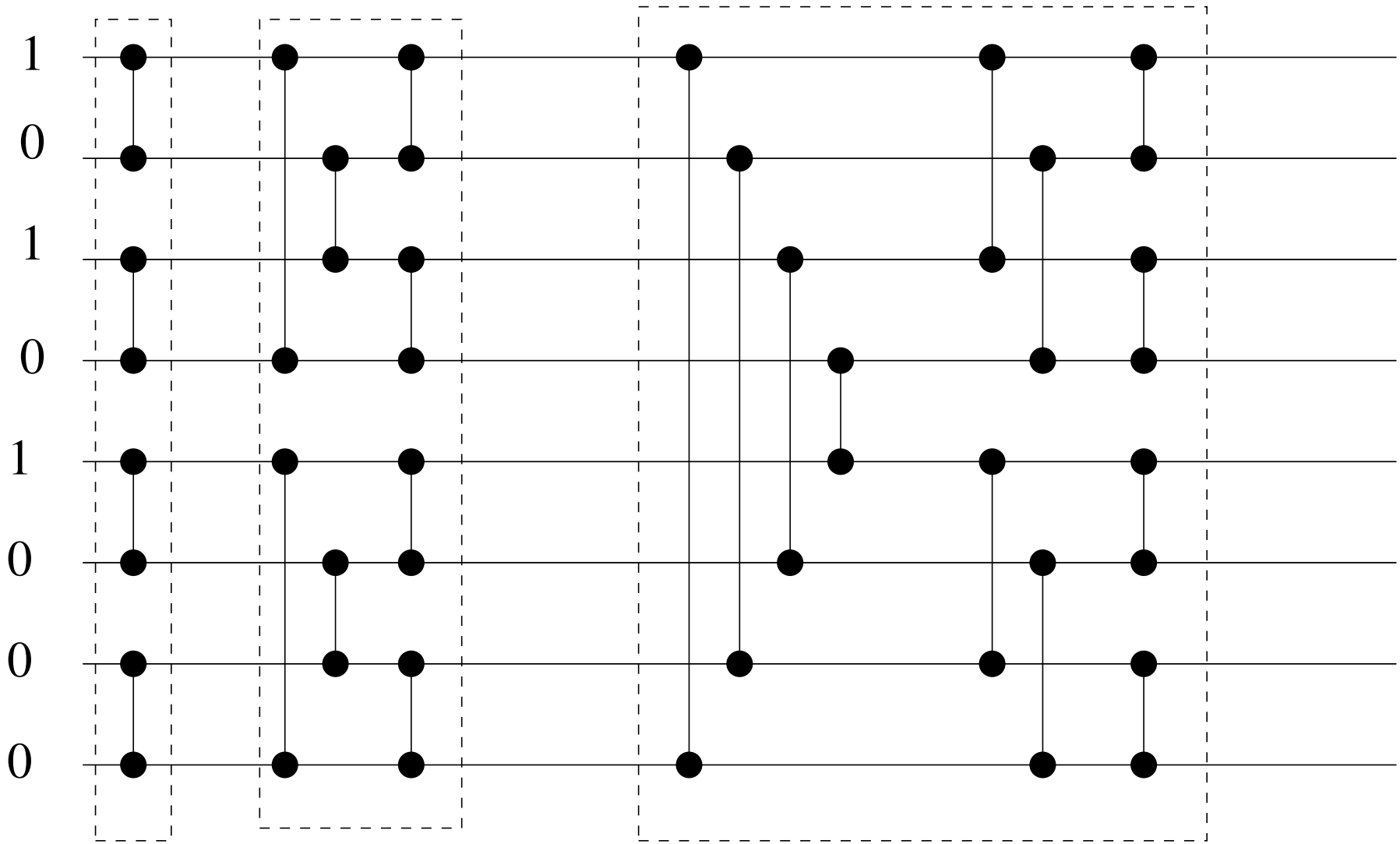
Merger[n]



Sorter[n]



A Sorter[n], to merge results from two Sorter[$\frac{n}{2}$].
Sorter[$\frac{n}{2}$] (recursively) consists of two Sorter[$\frac{n}{4}$] follow by a Merger[$\frac{n}{2}$].
Sorter[n] is actually a sequence of Merger[.].


1
0
1
0
1
0
0
0

Depth of Sorter[n]

Depth of Sorter[n] denoted $D(n)$
Depth of parallel Sorter[$\frac{n}{2}$] plus the depth of $\lg n$, the depth of Merger[n].

$$D(n) = \begin{cases} 0 & \text{if} \quad n = 1 \\ D(\frac{n}{2}) + \lg n & \text{if} \quad n = 2^k, k > 1 \end{cases}$$