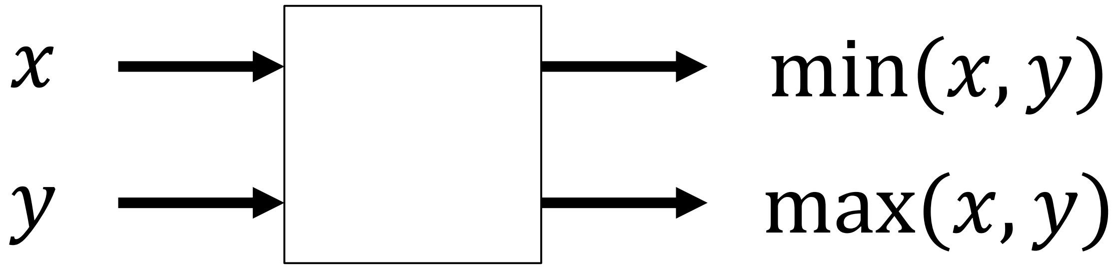
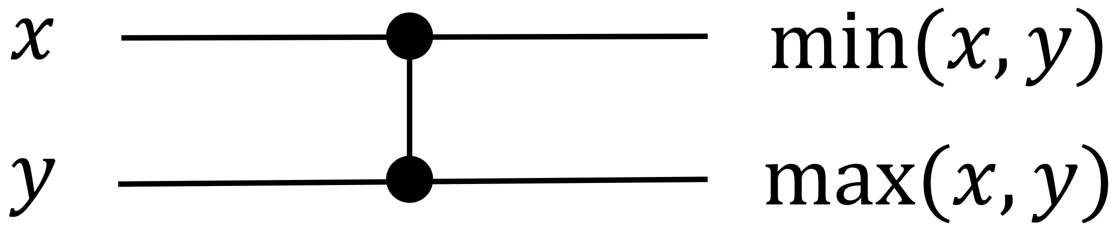
Sorting Networks

Uri Zwick
Tel Aviv University

May 2015

Comparators





Can we use comparators to
build *efficient* sorting networks?

Comparator networks

A comparator network is a
network composed of comparators.

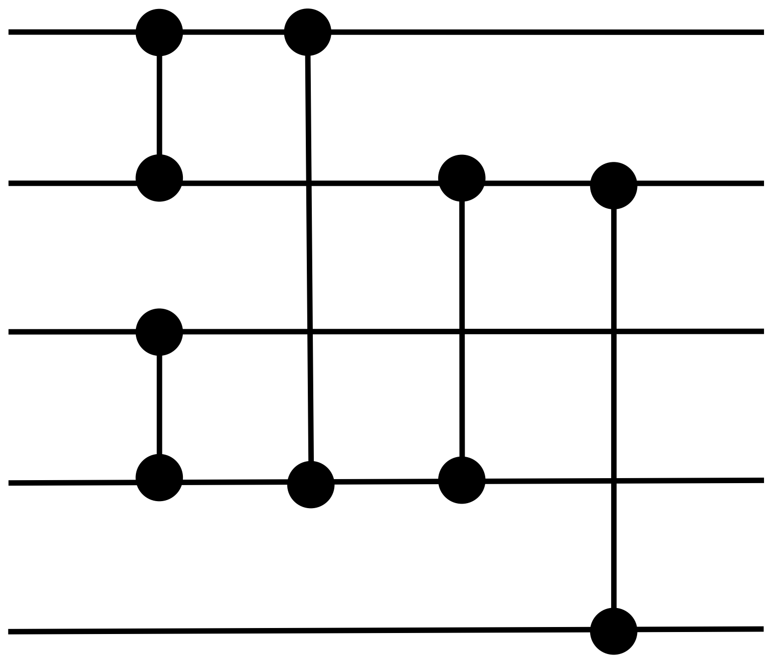
There are n input wires,
each feeding a single comparator.

Each output of a comparator is either
an output wire, or feeds a single comparator.

The network must be *acyclic*.

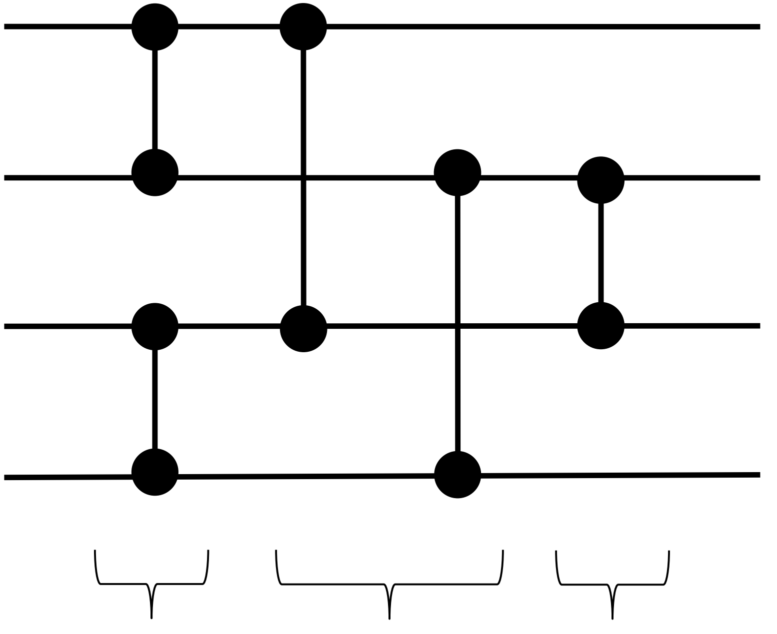
Number of output wires must also be n.

"Standard form"



Exercise: Show that any comparator network is equivalent to a network in standard form.

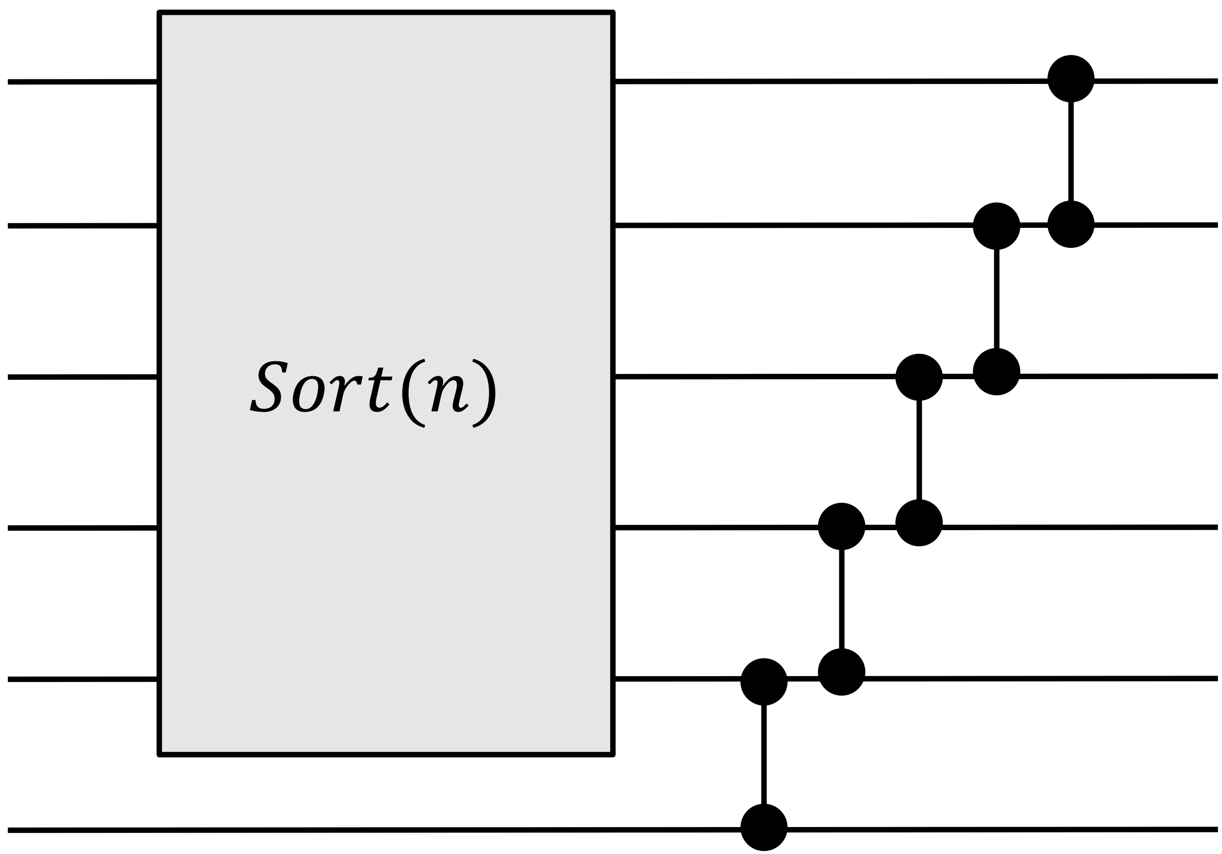
A simple sorting network



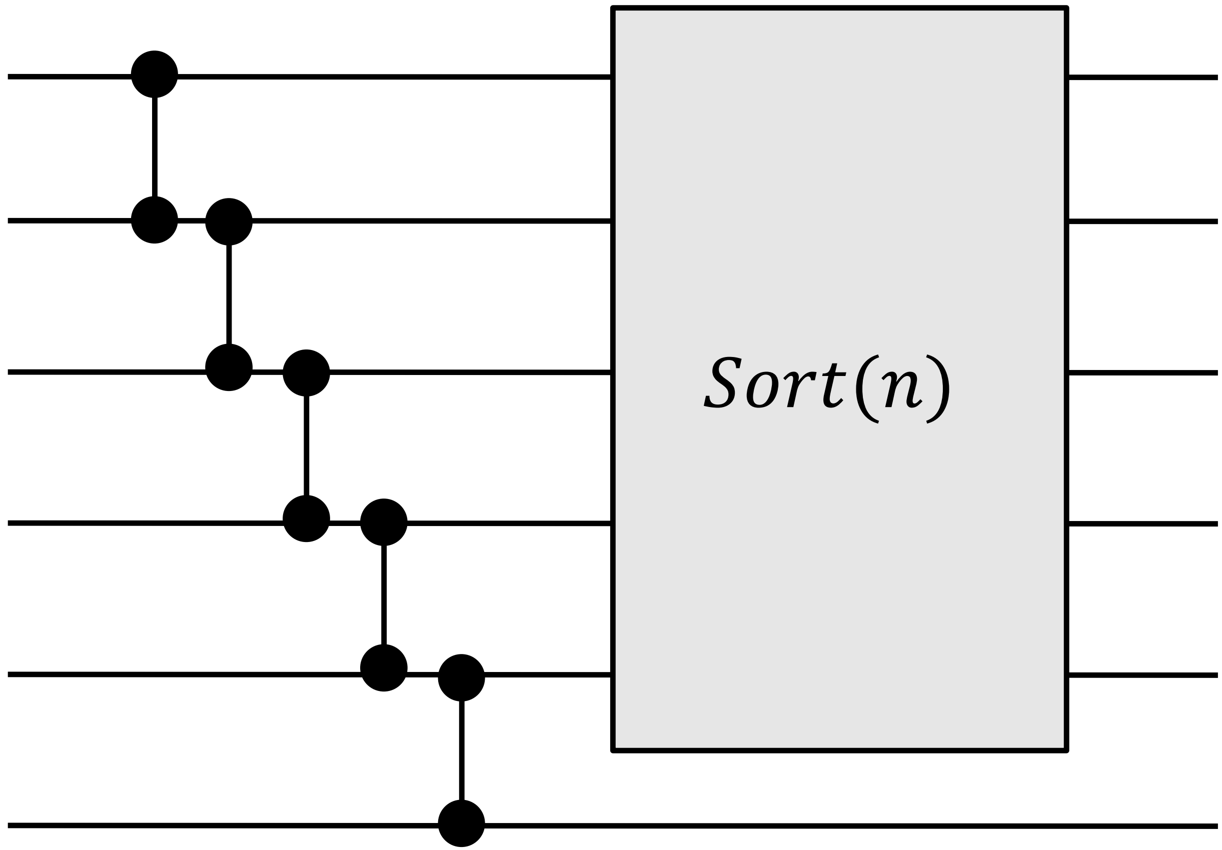
5 comparators

3 levels

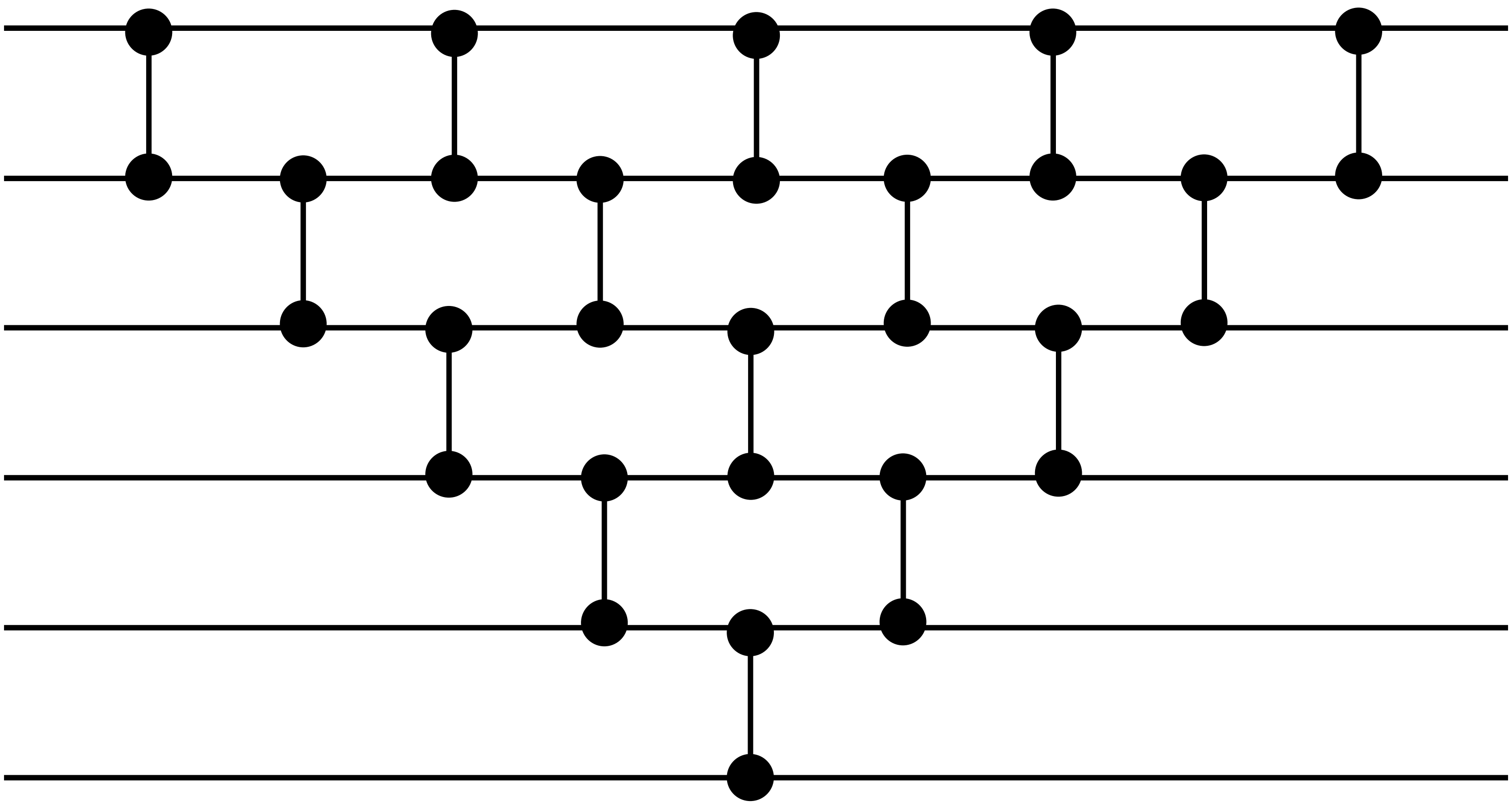
Insertion sort



Selection/bubble sort



Selection/bubble Sort

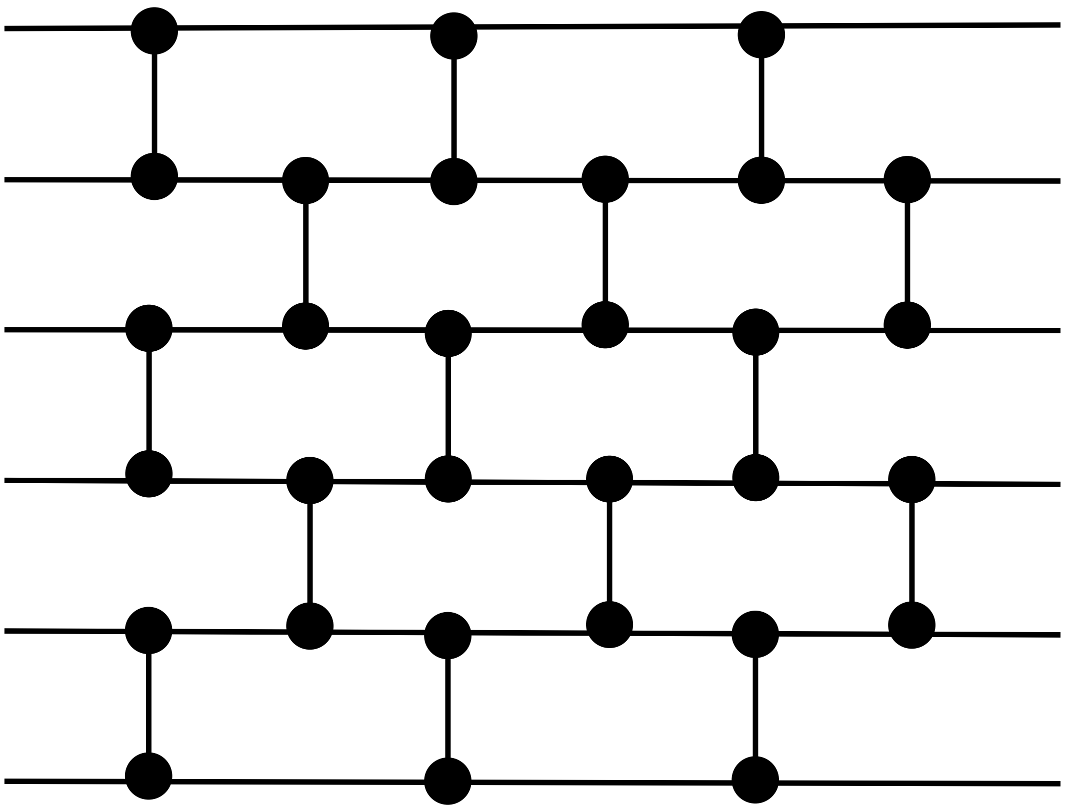


Size = $\frac{n(n-1)}{2}$ Depth = $2n-1$

Exercise: Any sorting network that only compares adjacent lines must be of size at least $\frac{n(n-1)}{2}$

Exercise: Prove that the network on the next slide, whose depth is n, is a sorting network

Odd-Even Transposition Sort



Size = $\frac{n(n-1)}{2}$ Depth = n

The 0-1 principle

Theorem:

If a network sort all 0-1 inputs,
then it sort all inputs

The 0-1 principle

Lemma:

Let f be a monotone non-decreasing function. Then, if a network maps $x_1, x_2, \ldots, x_n$ to $y_1, y_2, \ldots, y_n$, then it maps $f(x_1), f(x_2), \ldots, f(x_n)$ to $f(y_1), f(y_2), \ldots, f(y_n)$

Proof:

By induction on the number of comparisons using

$$f(\min(a,b)) = \min(f(a), f(b))$$
$$f(\max(a,b)) = \max(f(a), f(b))$$

The 0-1 principle

Proof:

Suppose that a network is *not* a sorting network.

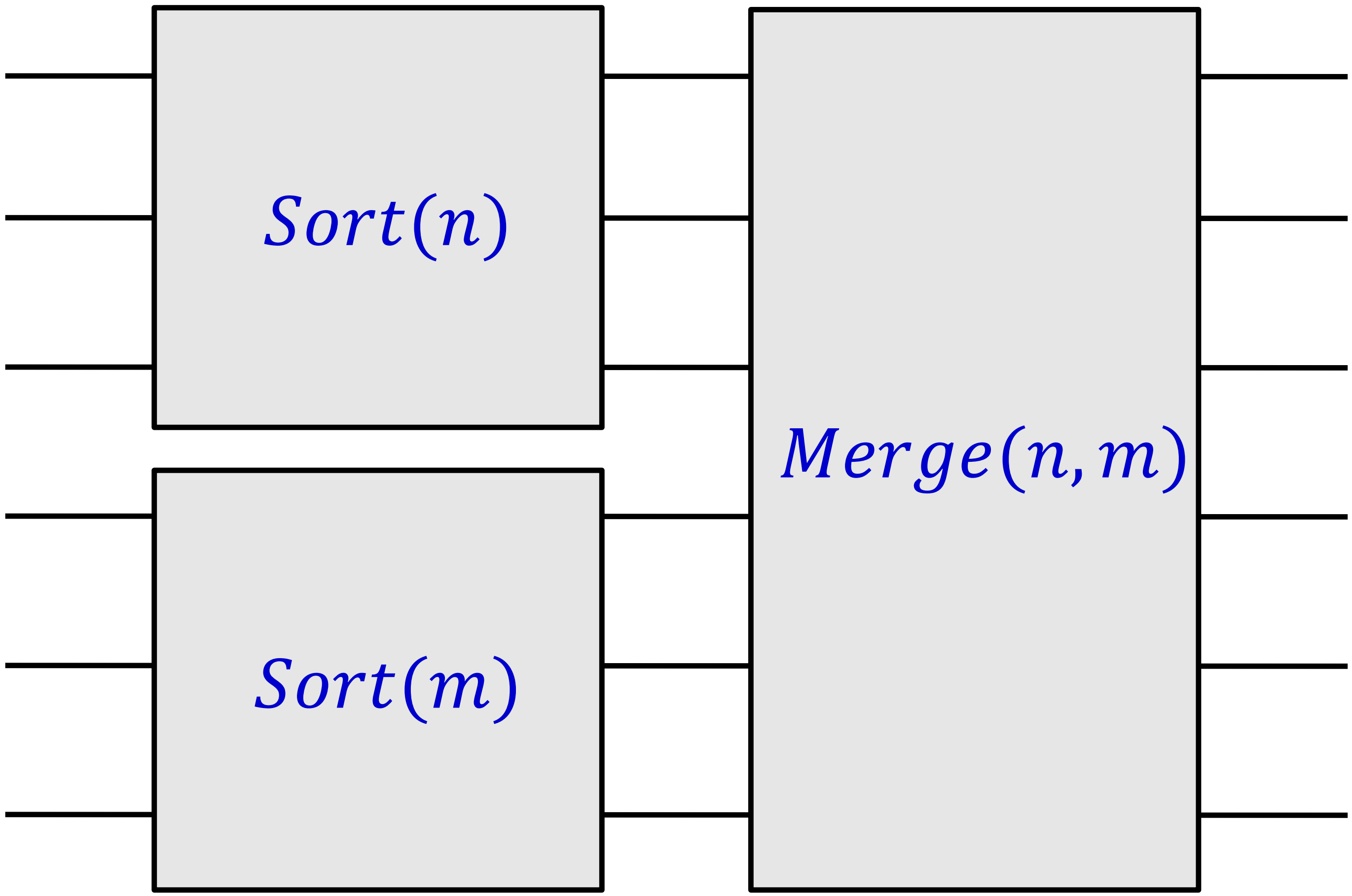
It then maps some $x_1, x_2, \ldots, x_n$ to $y_1, y_2, \ldots, y_n$, where $y_i > y_{i+1}$, for some $1 \le i < n$.

Let $f(x) = 1$, iff $x \ge y_i$, 0 otherwise.

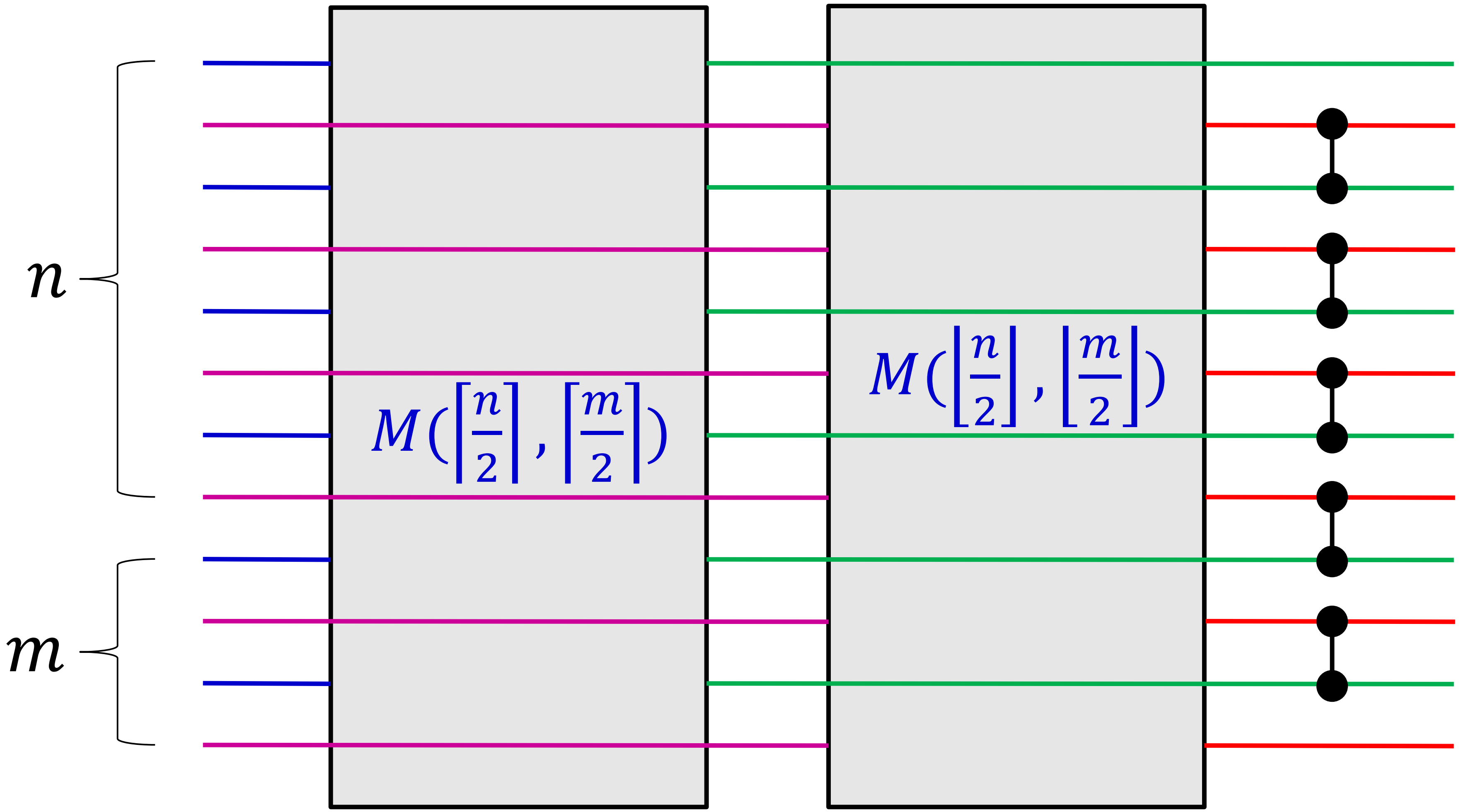
The network maps $f(x_1), f(x_2), \ldots, f(x_n)$ to $f(y_1), \ldots, f(y_i) = 1, f(y_{i+1}) = 0, \ldots, f(y_n)$

Thus, the network does *not* sort all 0-1 inputs.

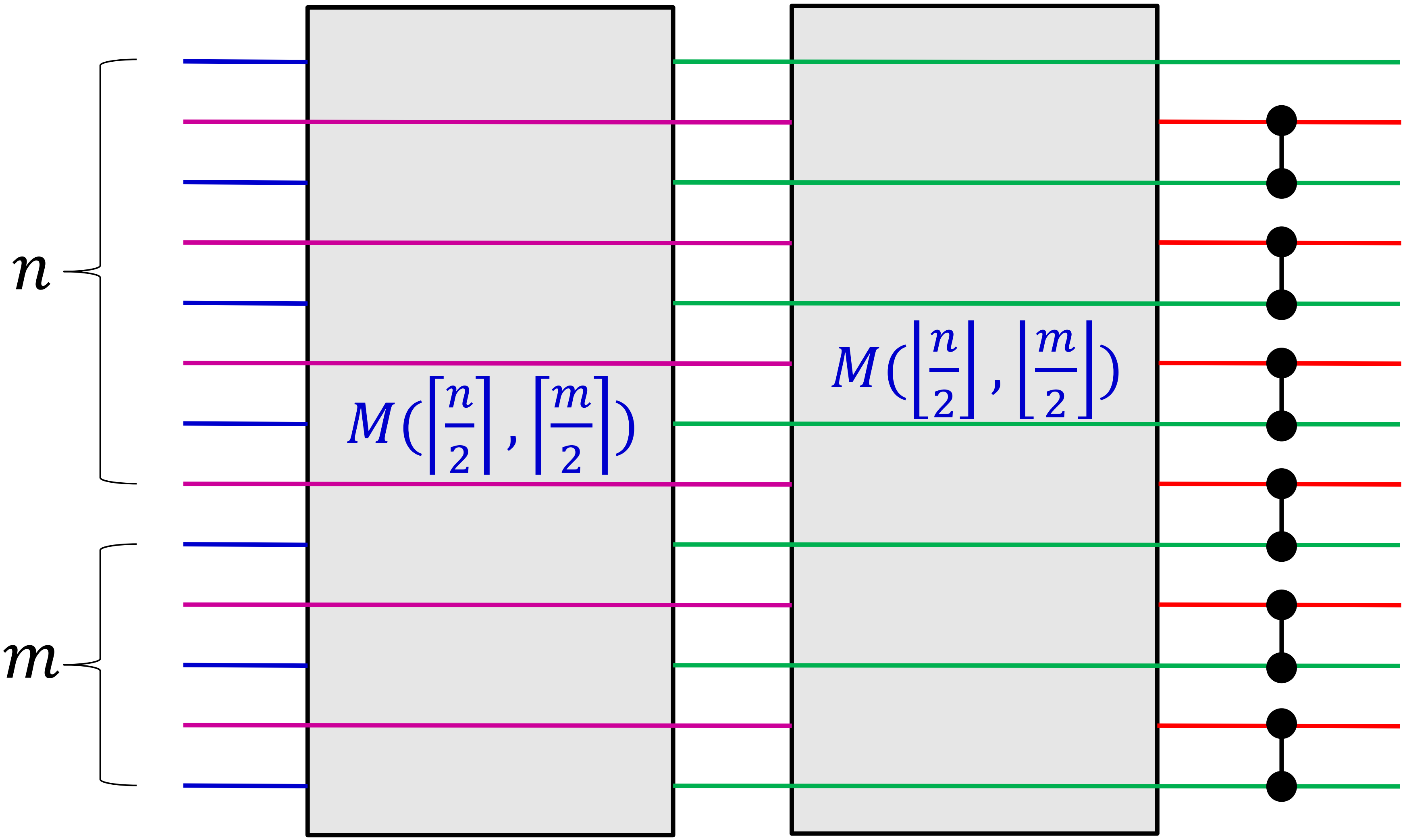
Sorting by merging



Batcher's odd-even merge



Batcher's odd-even merge



Batcher's odd-even merge

To merge $a_1, a_2, .., a_n$ with $b_1, b_2, .., b_m$:

Split $a_1, a_2, .., a_n$ into $a_1, a_3, \ldots$ and $a_2, a_4, \ldots$

Split $b_1, b_2, .., b_m$ into $b_1, b_3, \ldots$ and $b_2, b_4, \ldots$

Merge odd-indexed items to create $o_1, o_2, \ldots$

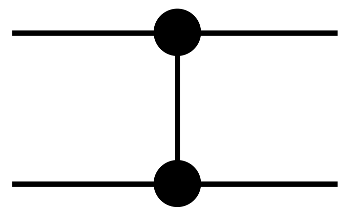
Merge even-indexed items to create $e_1, e_2, \ldots$

Compare/swap $(e_1, o_2), (e_2, o_3), \ldots$

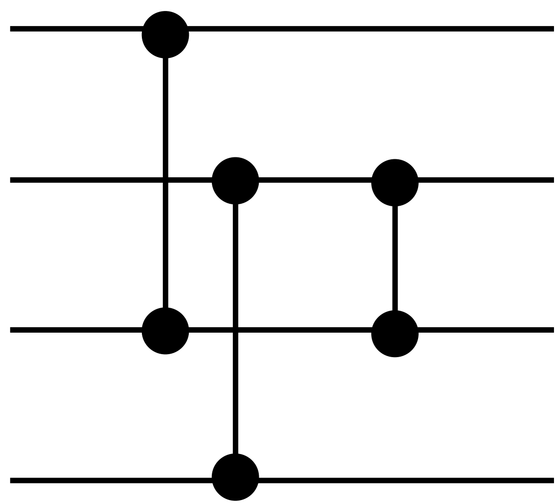
Does it really work?

Batcher's odd-even merge

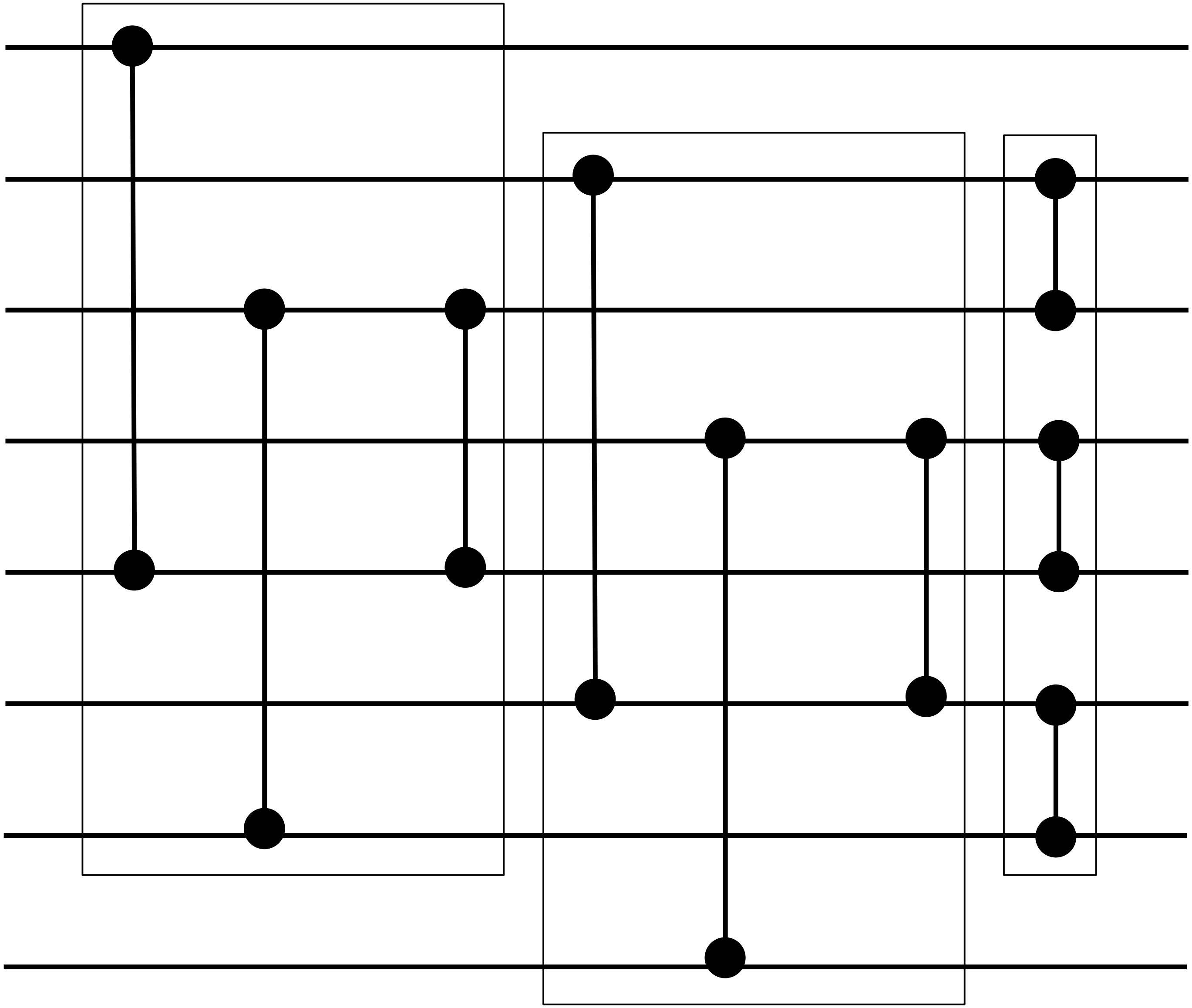
$M(1,1)$



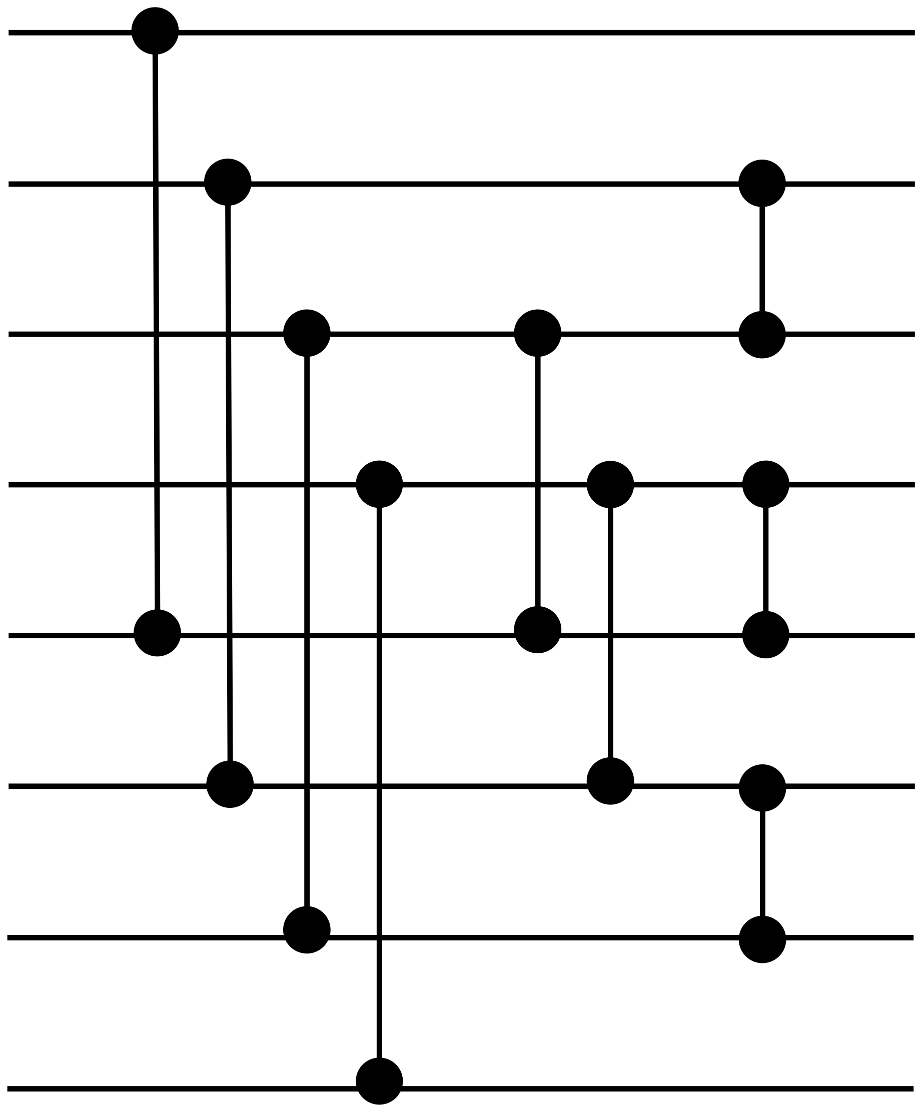
$M(2,2)$



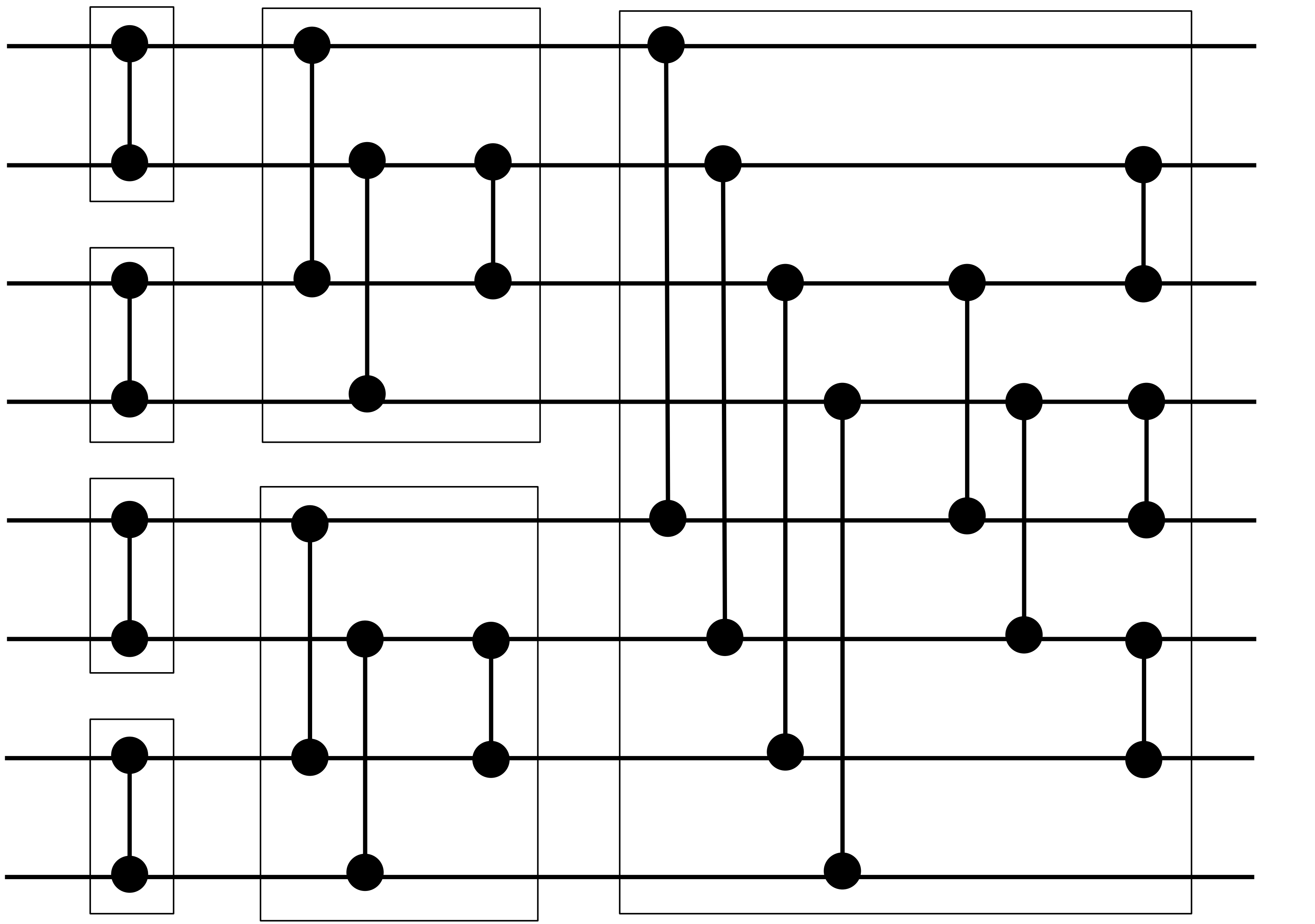
Batcher's odd-even merge - $M(4,4)$



Batcher's odd-even merge - $M(4,4)$



Odd-even merge → Odd-even sort



Batcher's odd-even merge

Proof using the 0-1 principle.

Suppose that $a_1, a_2, .., a_n$ starts with n_0 0's and that $b_1, b_2, .., b_m$ starts with m_0 0's.

Then $o_1, o_2, \ldots$ starts with $\left\lceil \frac{n_0}{2} \right\rceil + \left\lceil \frac{m_0}{2} \right\rceil$ 0's, and $e_1, e_2, \ldots$ starts $\left\lfloor \frac{n_0}{2} \right\rfloor + \left\lfloor \frac{m_0}{2} \right\rfloor$ 0's.

The difference is either 0,1 or 2!

There is a problem only if difference is 2 and last level of comparators fixes it.

Batcher's odd-even merge

e		0	0	0	0	0	1	1	1
o	0	0	0	0	0	1	1	1	

e		0	0	0	0	1	1	1	1
o	0	0	0	0	0	1	1	1	

e		0	0	0	1	1	1	1	1
o	0	0	0	0	0	1	1	1	

Batcher’s odd-even merge

Exercise: Justify the use of the 0-1 principle.

Exercise: Prove the correctness of the odd-even merging network directly without relying on the 0-1 principle.

Batcher's odd-even merge

Size – number of comparators

$$M(n,0) = M(0,m) = 0 \qquad\qquad M(1,1) = 1$$

$$M(n,m) = M\left(\left\lceil\frac{n}{2}\right\rceil, \left\lceil\frac{m}{2}\right\rceil\right) + M\left(\left\lfloor\frac{n}{2}\right\rfloor, \left\lfloor\frac{m}{2}\right\rfloor\right) + \left\lfloor\frac{n+m-1}{2}\right\rfloor$$

$$M(n,n) = 2M\left(\frac{n}{2}, \frac{n}{2}\right) + (n-1)$$

$$M\left(2^k, 2^k\right) = k2^k + 1$$

$$M(n,n) = n \lg n + O(n)$$

No better merging networks are known for any n, m!

Are the odd-even merge networks optimal?

Bitonic sequences

A sequence is *strict* bitonic iff it is a concatenation of a *decreasing* sequence and an *increasing* sequence

A sequence is bitonic iff it is a *cyclic* shift of a *strict* bitonic sequence

8 6 4 1 2 5 7 9 

4 1 2 5 7 9 8 6 

1 4 2 3 

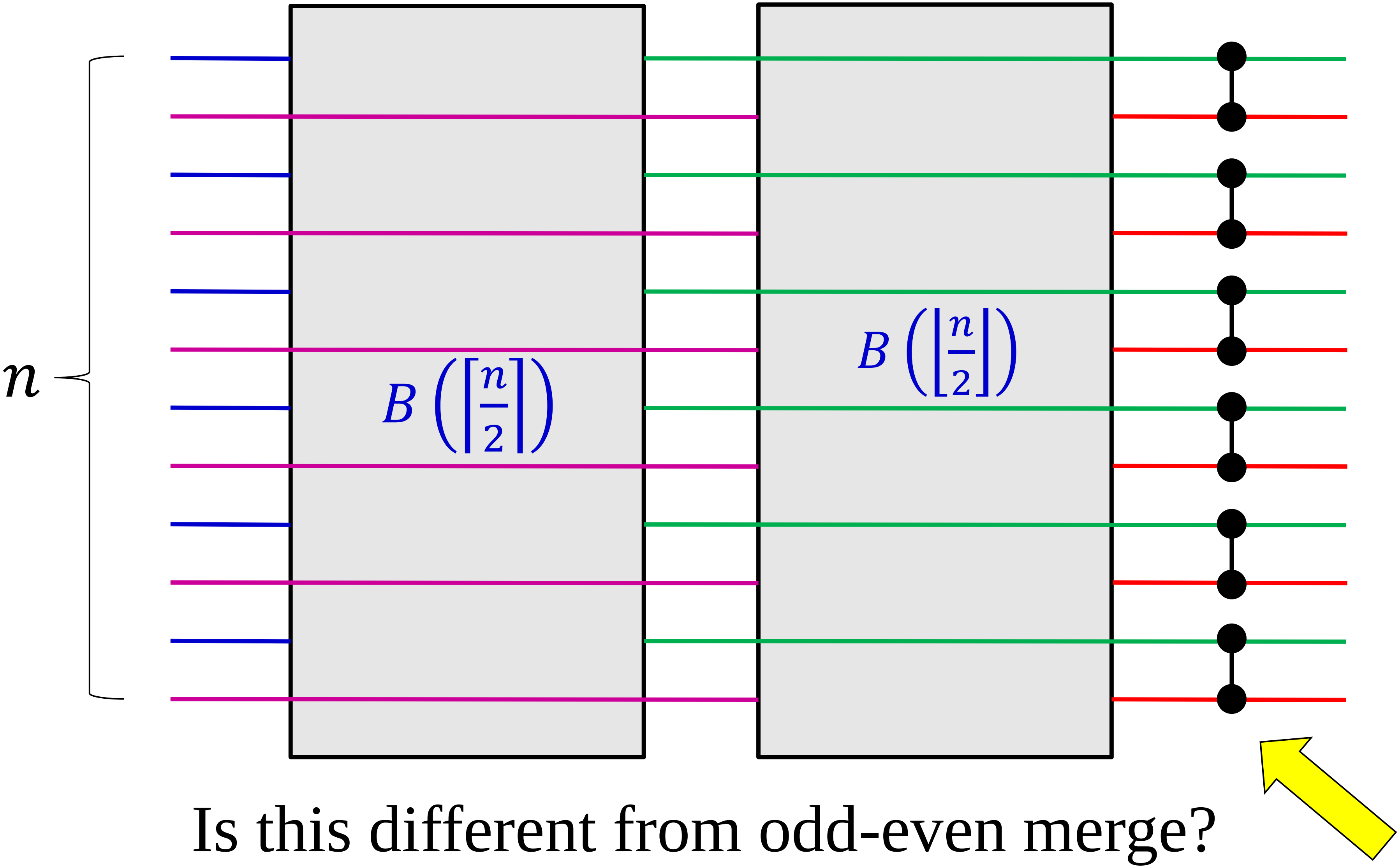
A Bitonic sorter

A (strict) bitonic sorter is a network that sorts every (strict) bitonic sequence.

If $a_1, a_2, .., a_n$ and $b_1, b_2, .., b_m$ are sorted, then $a_n, a_{n-1}, .., a_1, b_1, b_2, .., b_m$, is bitonic.

Thus, a strict bitonic sorter can serve as a merging network

Batcher's bitonic sorter



Is this different from odd-even merge?

Batcher's bitonic sorter

Simple proof using the 0-1 principle.

A strict binary bitonic sequence – $1^k 0^\ell 1^{n-k-\ell}$

The odd and even subsequences are also strict binary bitonic sequences.

By induction they are sorted correctly.

The difference between the number of 0's in the two sorted sequences is one of $-1,0,1$.

Final level of comparators fixes the problem.

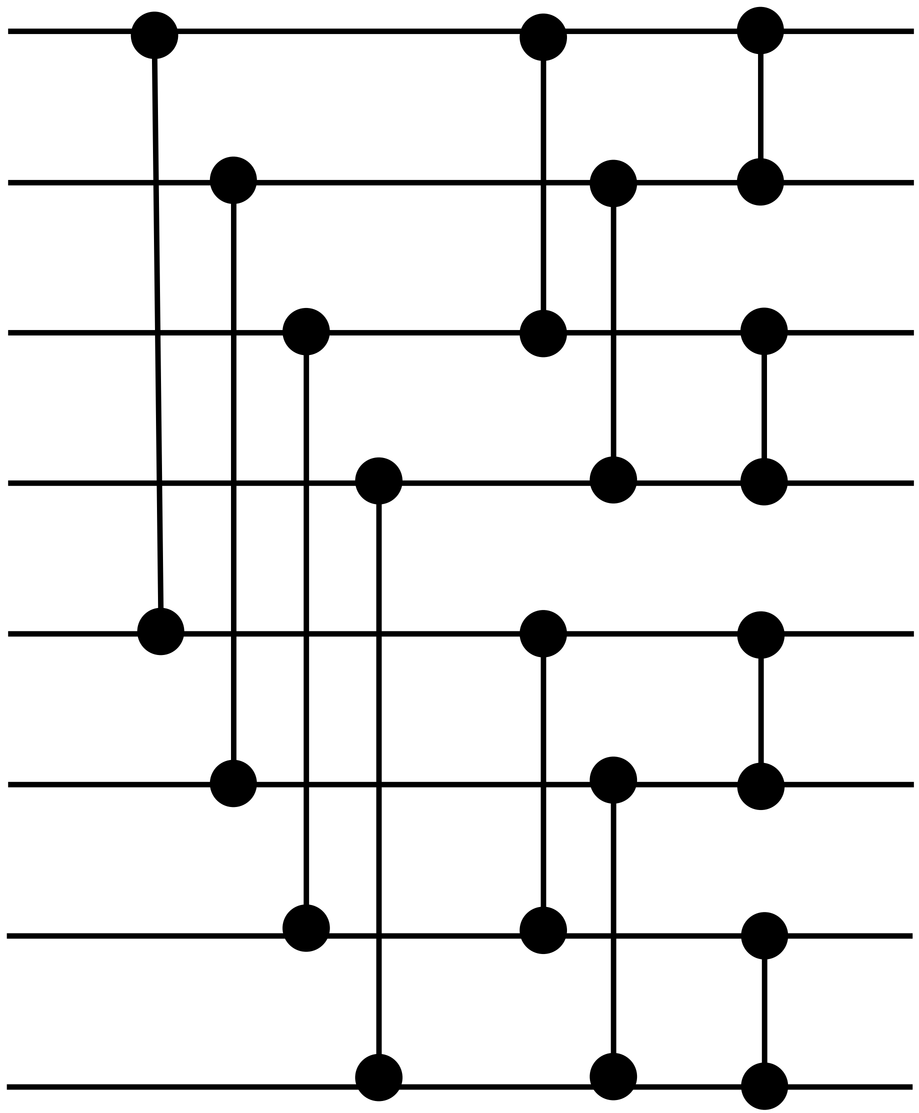
Batcher’s bitonic sorter

Exercise: Show that the difference between the number of 0’s in the odd and even subsequences of $1^k 0^\ell 1^{n-k-\ell}$ is

$$\left(\left\lceil\frac{k+\ell}{2}\right\rceil - \left\lfloor\frac{k+\ell}{2}\right\rfloor\right) - \left(\left\lceil\frac{k}{2}\right\rceil - \left\lfloor\frac{k}{2}\right\rfloor\right)$$

Note: We do not need this exact formula in the proof given in the previous slide. Seeing that the difference is $-1,0,1$ is immediate.

Batcher's bitonic sorter for n=2^k

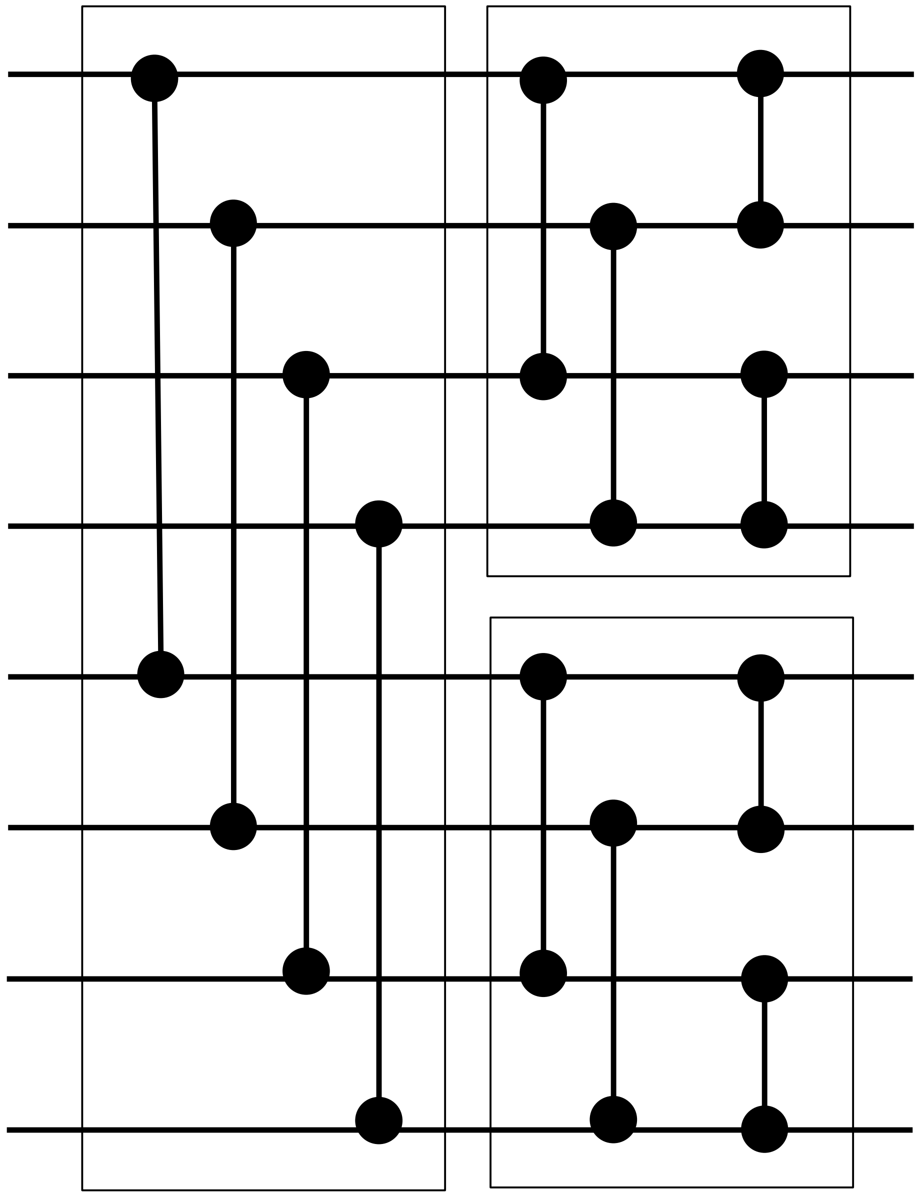


Very regular structure!

When $n = 2^k$, there are k levels with $\frac{n}{2}$ comparators each.

Lines i and j are compared at level ℓ iff they differ *only* in the ℓ-th most significant bit

Batcher's bitonic sorter for n=2^k



Alternative recursive definition for $n = 2^k$

Sorts general bitonic sequences

The AKS sorting networks

[Ajtai-Komlós-Szemerédi (1983)]

There are sorting networks of $O(\log n)$ depth, and hence $O(n \log n)$ size.

The construction is fairly complicated.

The constant factors are very large.