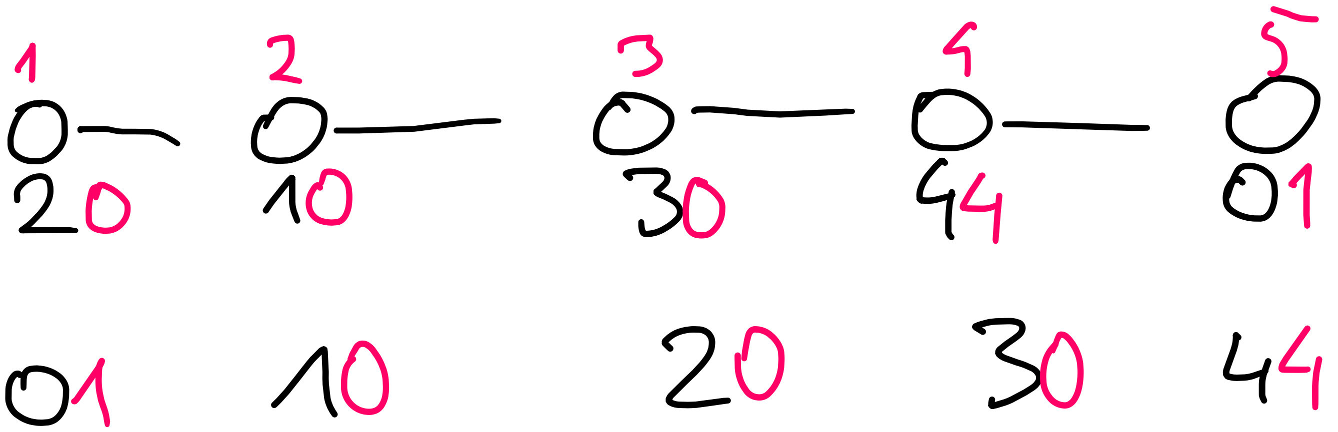
Sorting / Counters

n nodes: no ID's, or random ID's

output: nodes have ID's 1, ..., n, distinct



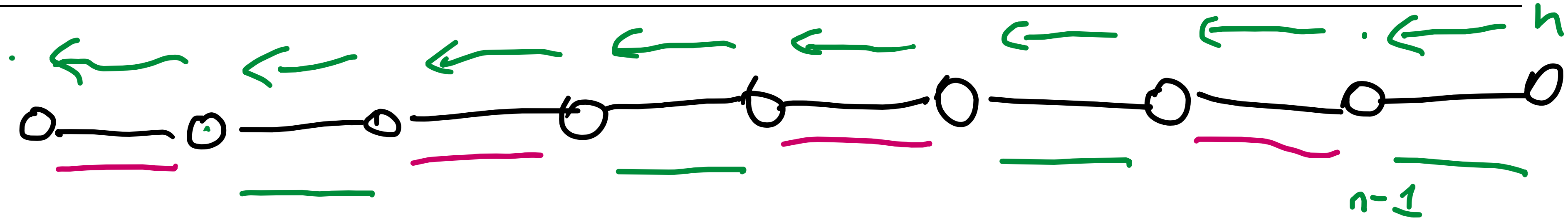
~~Sorting~~ / Counters

Events

Events	Counters
0 ||||	0 17
0 |||||||	0 17
0 |||||	
0 |||	0 16

Algorithm 4.3 Odd/Even Sort

Given an array of n nodes $(v_1, \ldots, v_n)$, each storing a value (not sorted).
repeat
    Compare and exchange the values at nodes i and $i + 1$, i odd
    Compare and exchange the values at nodes i and $i + 1$, i even
until done



Lemma 4.4 (0-1 Sorting Lemma). *If an oblivious comparison-exchange algorithm sorts all inputs of 0's and 1's, then it sorts arbitrary inputs.*

<u>Argument</u> x, x going to destination?

$n_i < x \Rightarrow 0$

$n_i \geq x \Rightarrow 1$

1 2 3 x 17 y . . .

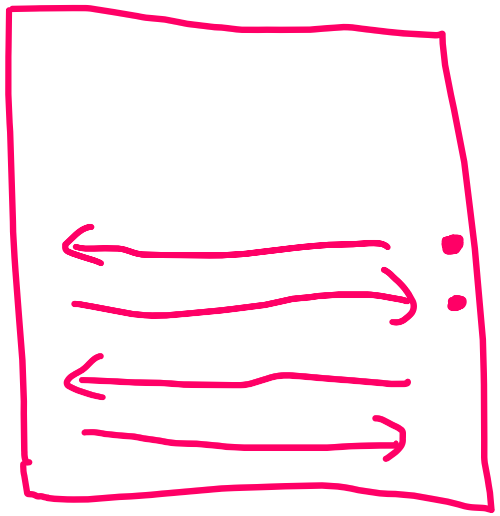
0 0 0 1 1 0 1 1 1 0 0 0

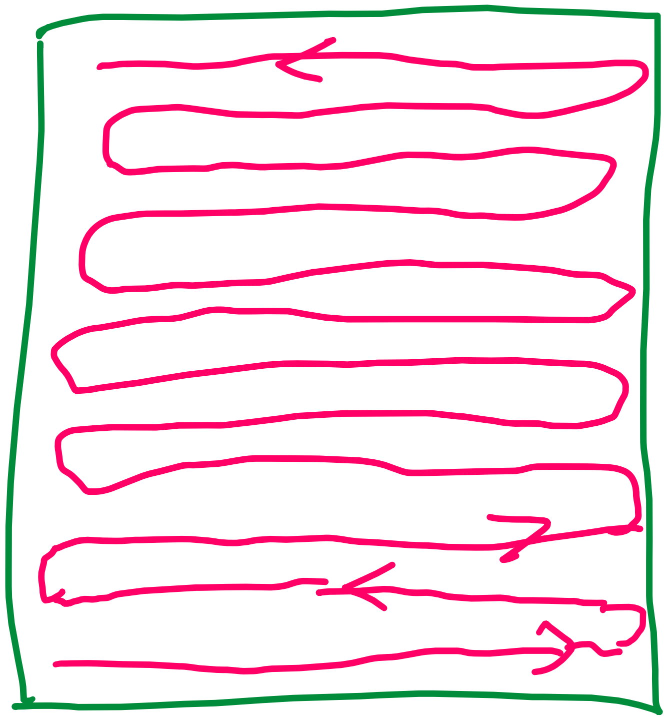
0 1 0 1 1 1 0 0 1 1 0 1 0 0 0

$\boxed{0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1}$

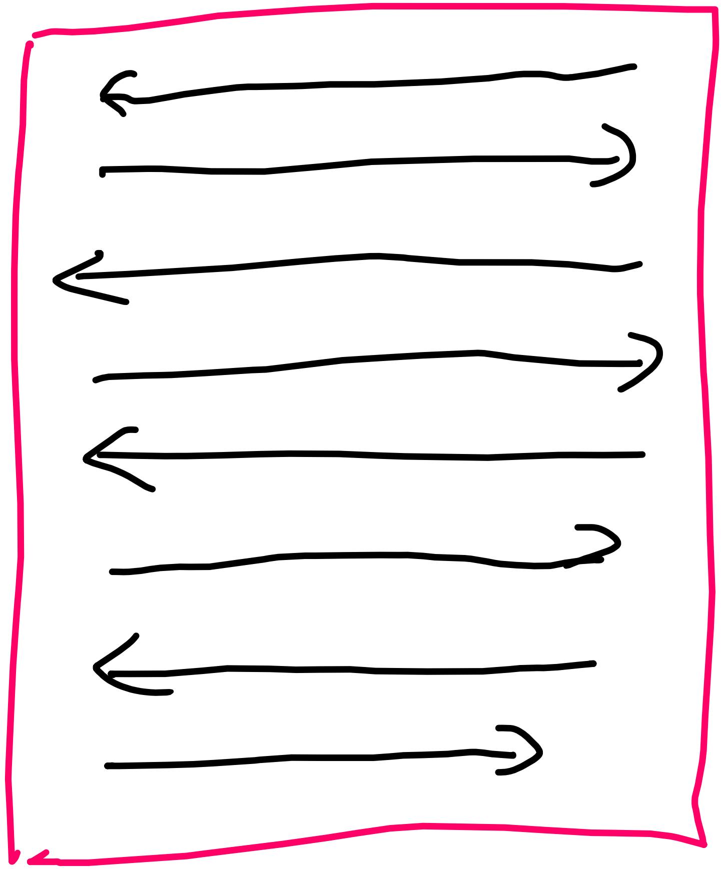
Algorithm 4.6 Shearsort

We are given a mesh with m rows and m columns, m even, $n = m^2$.
The sorting algorithm operates in phases, and uses the odd/even sort algorithm on rows or columns.
repeat
    In the odd phases $1, 3, \ldots$ we sort all the rows, in the even phases $2, 4, \ldots$ we sort all the columns, such that:
    Columns are sorted such that the small values move up.
    Odd rows $(1, 3, \ldots, m-1)$ are sorted such that small values move left.
    Even rows $(2, 4, \ldots, m)$ are sorted such that small values move right.
until done

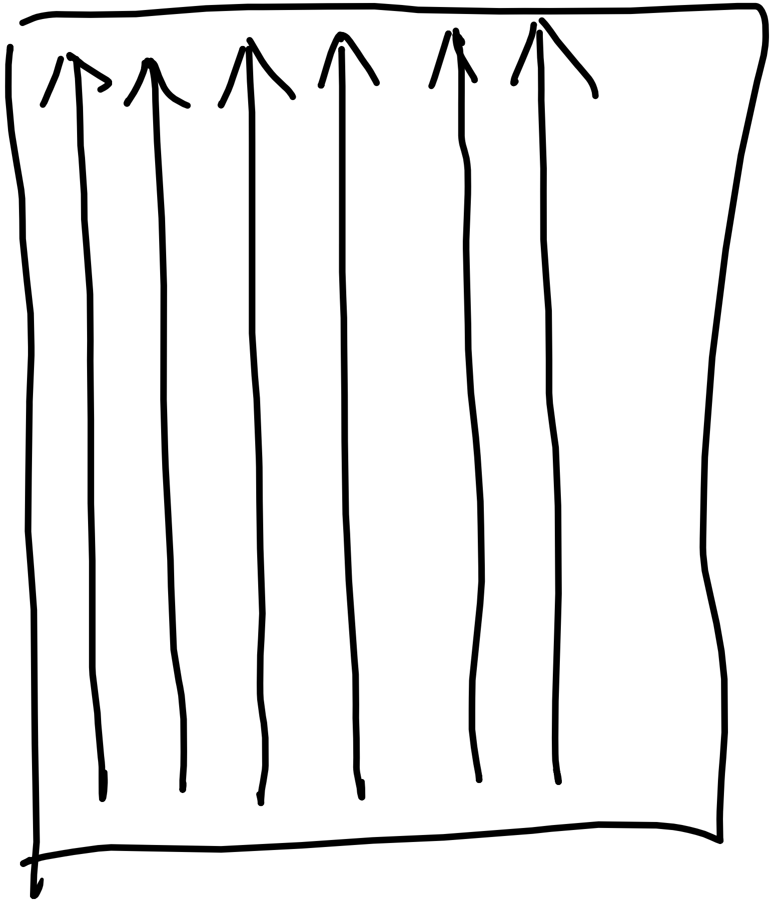


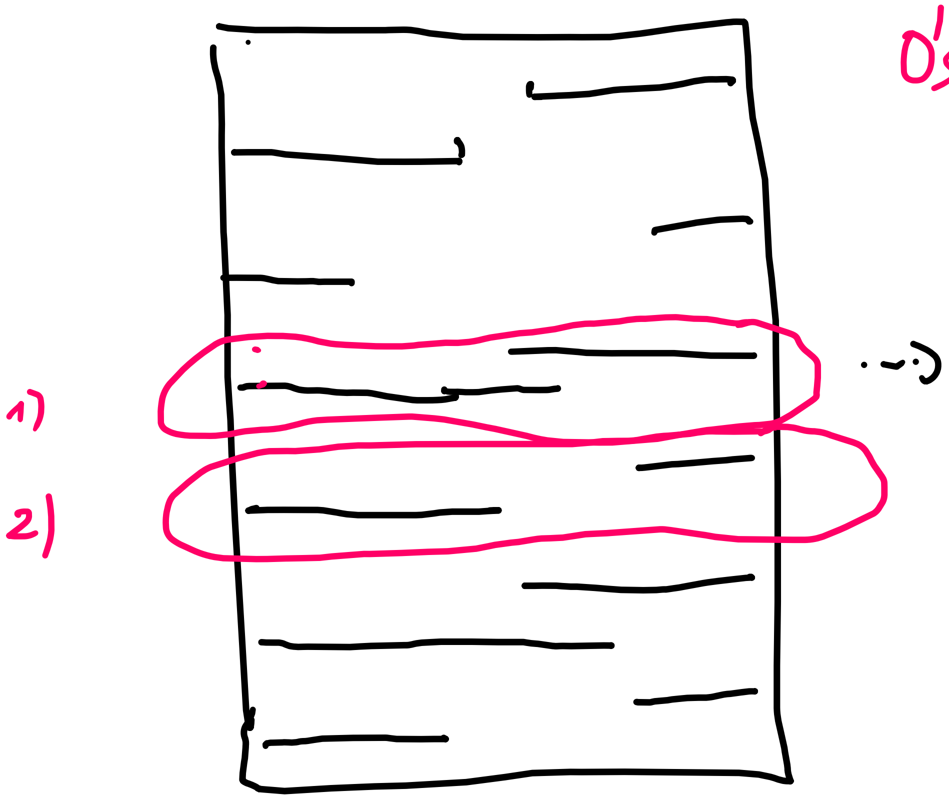


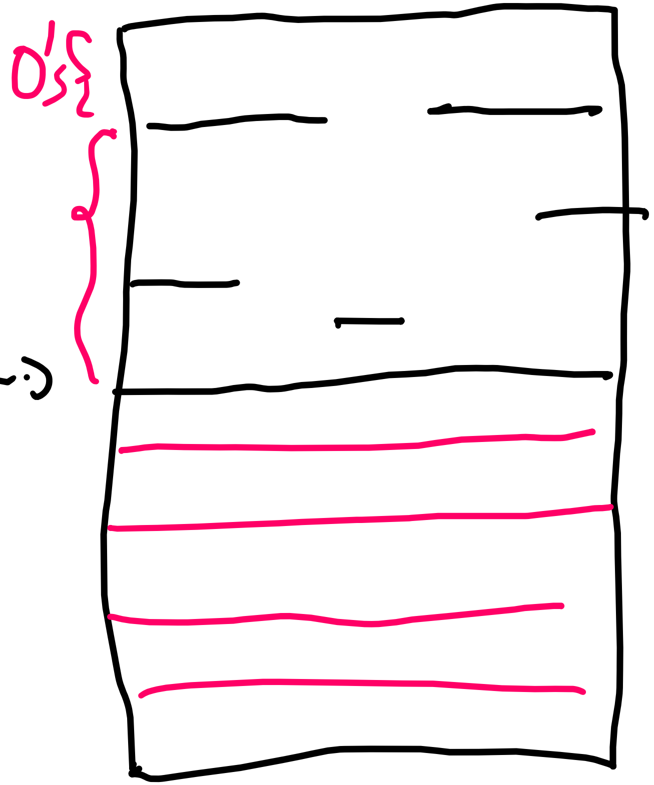
1)



2)

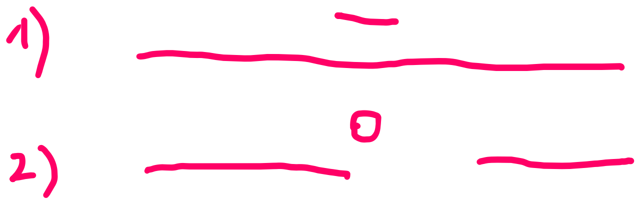


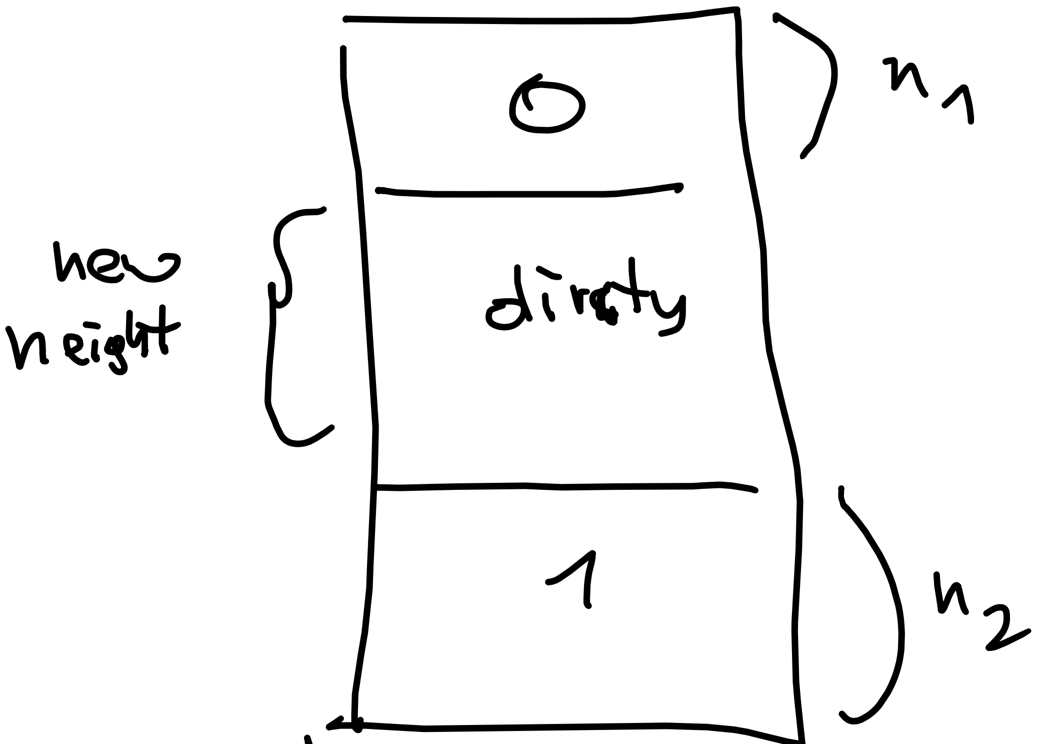




2 rows:
either
row of 0's
dirty row
or
dirty row
row of 1's

$\Rightarrow$ after sorting columns $\geq$ half row clean



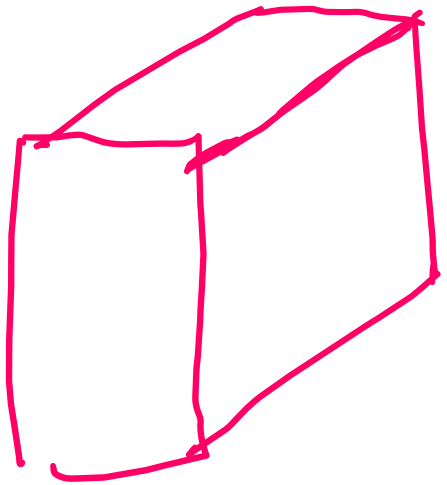


time: $O(\log n \sqrt{n})$

$n_1 + n_2 \geq \frac{1}{2}$ height

$\Rightarrow$ after $\log n$ rounds
dirty rows $= 1$

then 1 round terminates sorting



2 dim: $O(\sqrt{n} \cdot \log n)$

shearsort on 3-dim. cube

improves! $O(\sqrt[3]{n} \log n)$

4-dim., 5 dim....

Algorithm: periodic with constant period

period of length 3

repeat

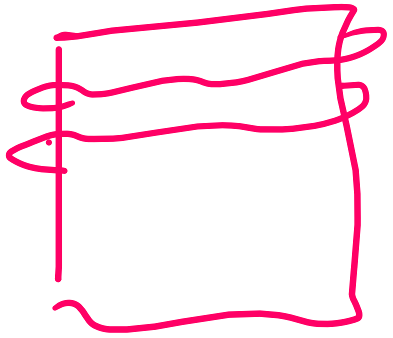
1. ...

2. ...

3. ...

possible:

time $\approx \sqrt{n} \cdot \log^2 n$



Proof: JACM

Theorem 4.7. *Algorithm 4.6 sorts n values in $\sqrt{n}(\log n+1)$ time in snake-like order.*

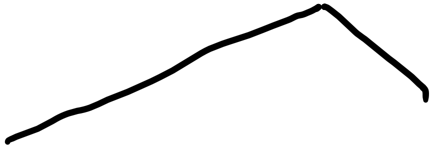
Algorithm 4.12 Half Cleaner

1: A half cleaner is a comparison network of depth 1, where we compare wire i with wire $i + n/2$ for $i = 1, \ldots, n/2$ (we assume n to be even).

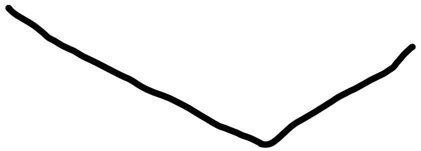
For bitonic inputs:

output: one half cleaned (same value)
one half bitonic

bitonic:



or

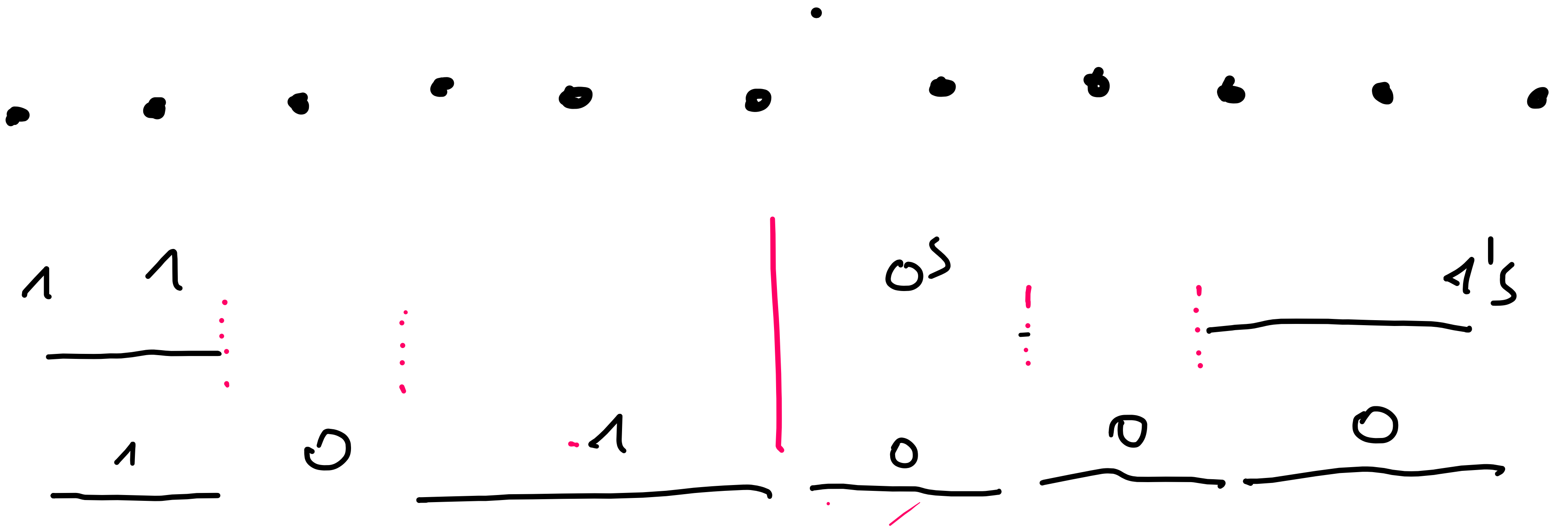


Algorithm 4.12 Half Cleaner

1: A half cleaner is a comparison network of depth 1, where we compare wire i with wire $i + n/2$ for $i = 1, \ldots, n/2$ (we assume n to be even).

For bitonic inputs:

output: one half cleaned (same value)
one half bitonic



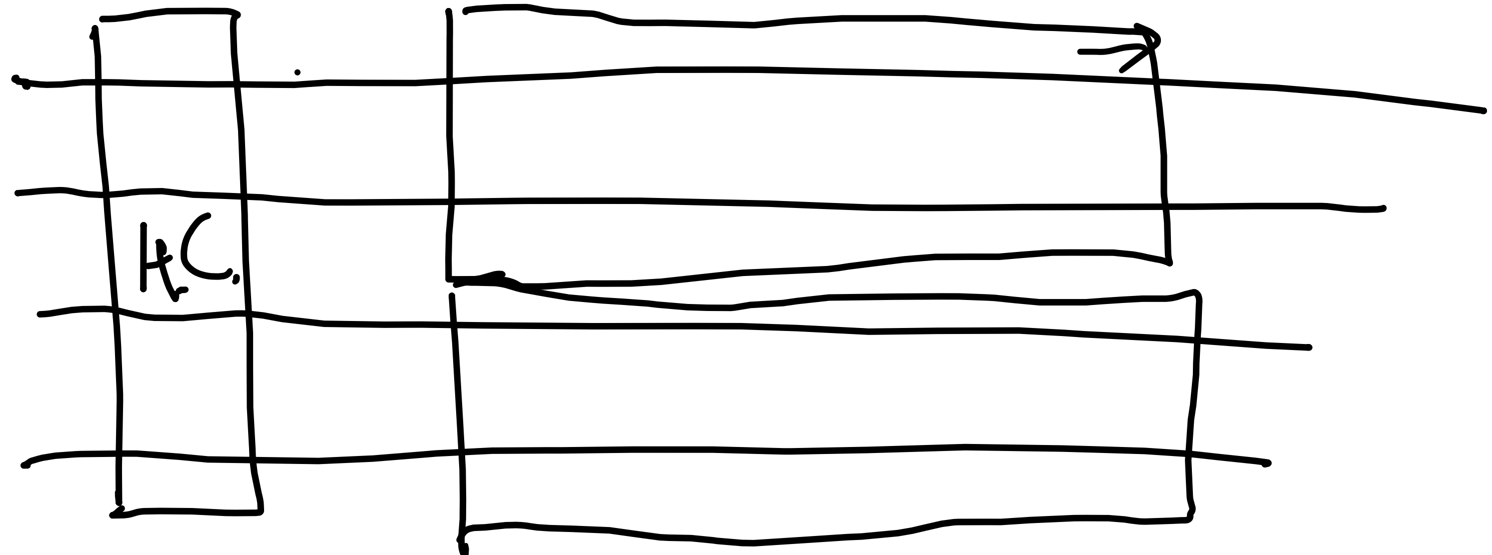
Algorithm 4.14 Bitonic Sequence Sorter

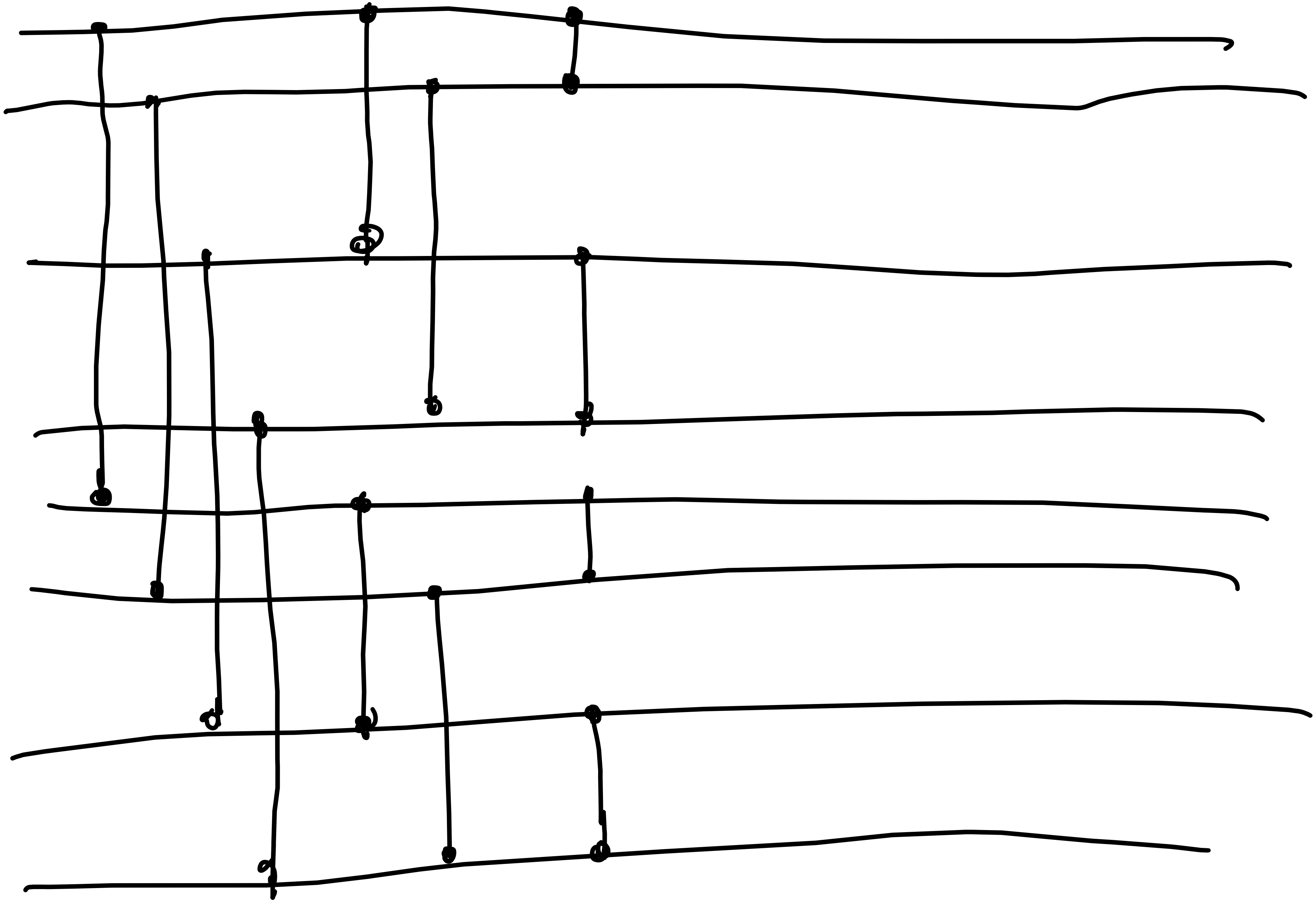
1: A bitonic sequence sorter of width n (n being a power of 2) consists of a half cleaner of width n, and then two bitonic sequence sorters of width $n/2$ each.

2: A bitonic sequence sorter of width 1 is empty.

1) ~~biton~~ half-cleaner of width n

2) recursively: 2 bitonic sequence sorter

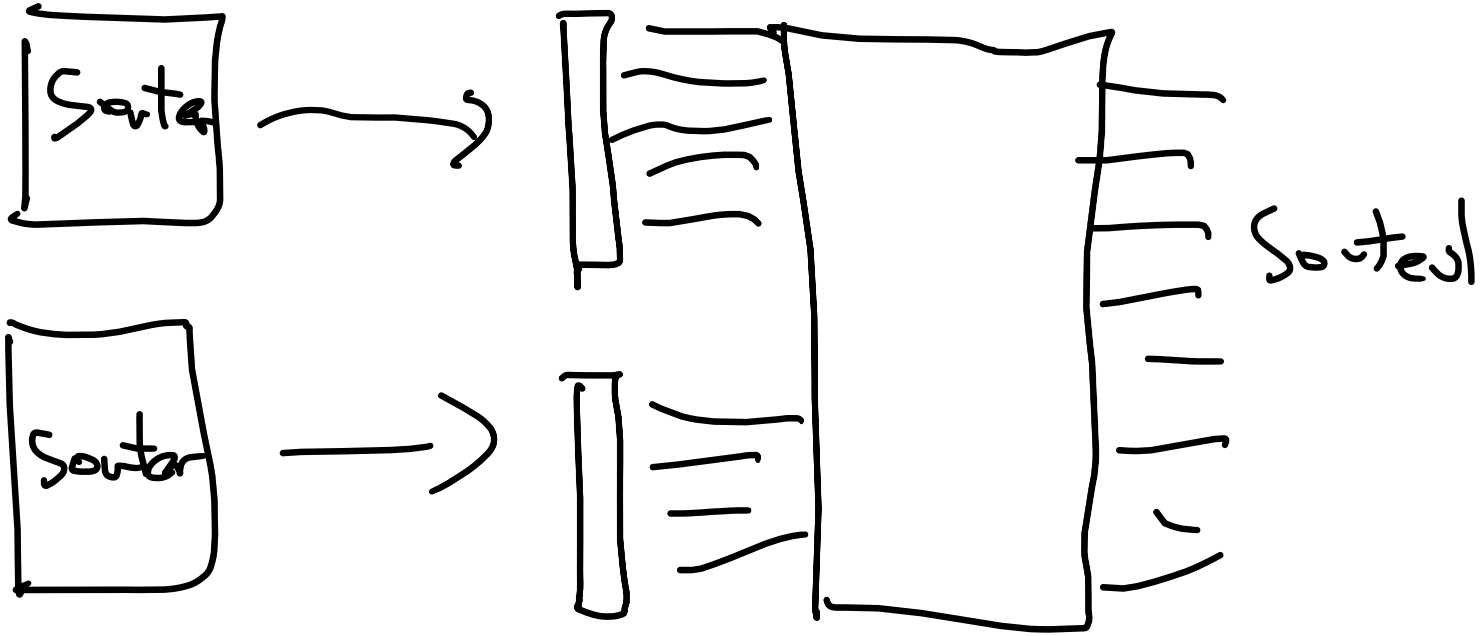




Algorithm 4.16 Merging Network

1: A merging network of width n is a merger of width n followed by two bitonic sequence sorters of width $n/2$. A merger is a depth-one network where we compare wire i with wire $n - i + 1$, for $i = 1, \ldots, n/2$.

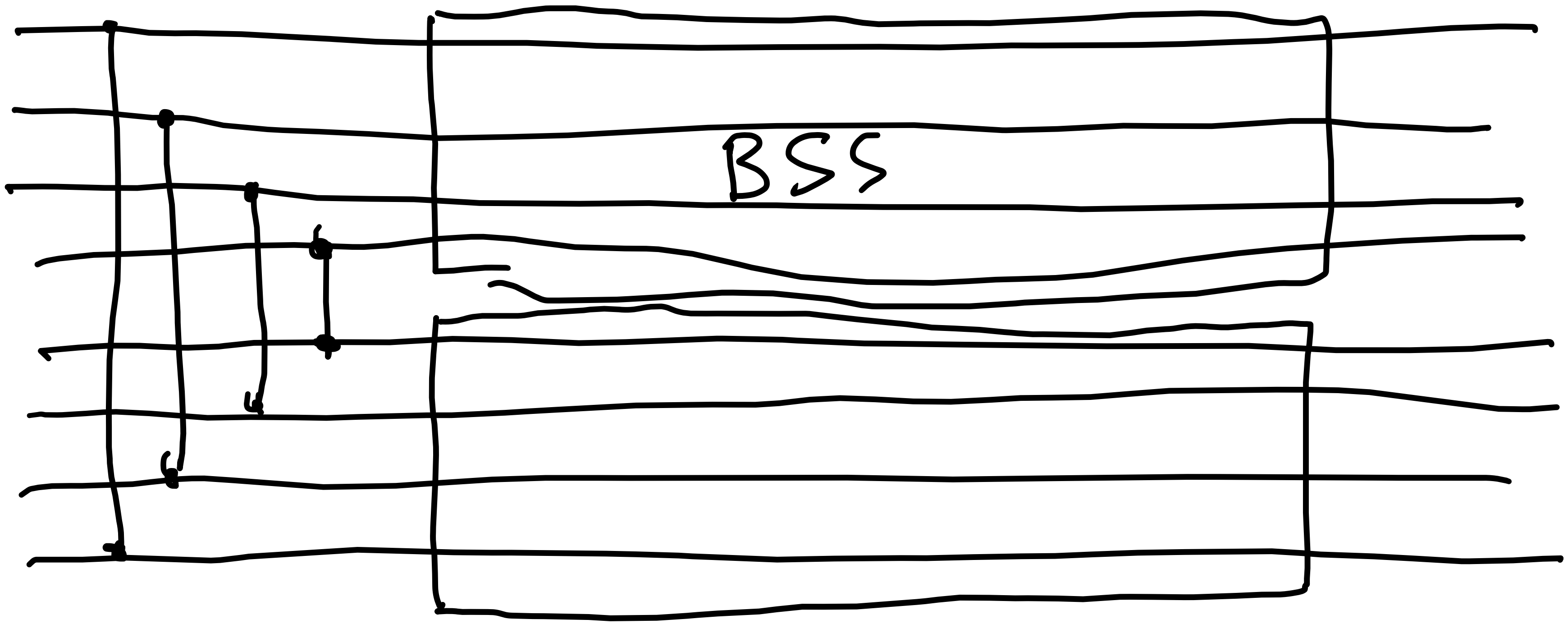
Lemma 4.17. *A merging network of width n (Algorithm 4.16) merges two sorted input sequences of length $n/2$ each into one sorted sequence of length n.*

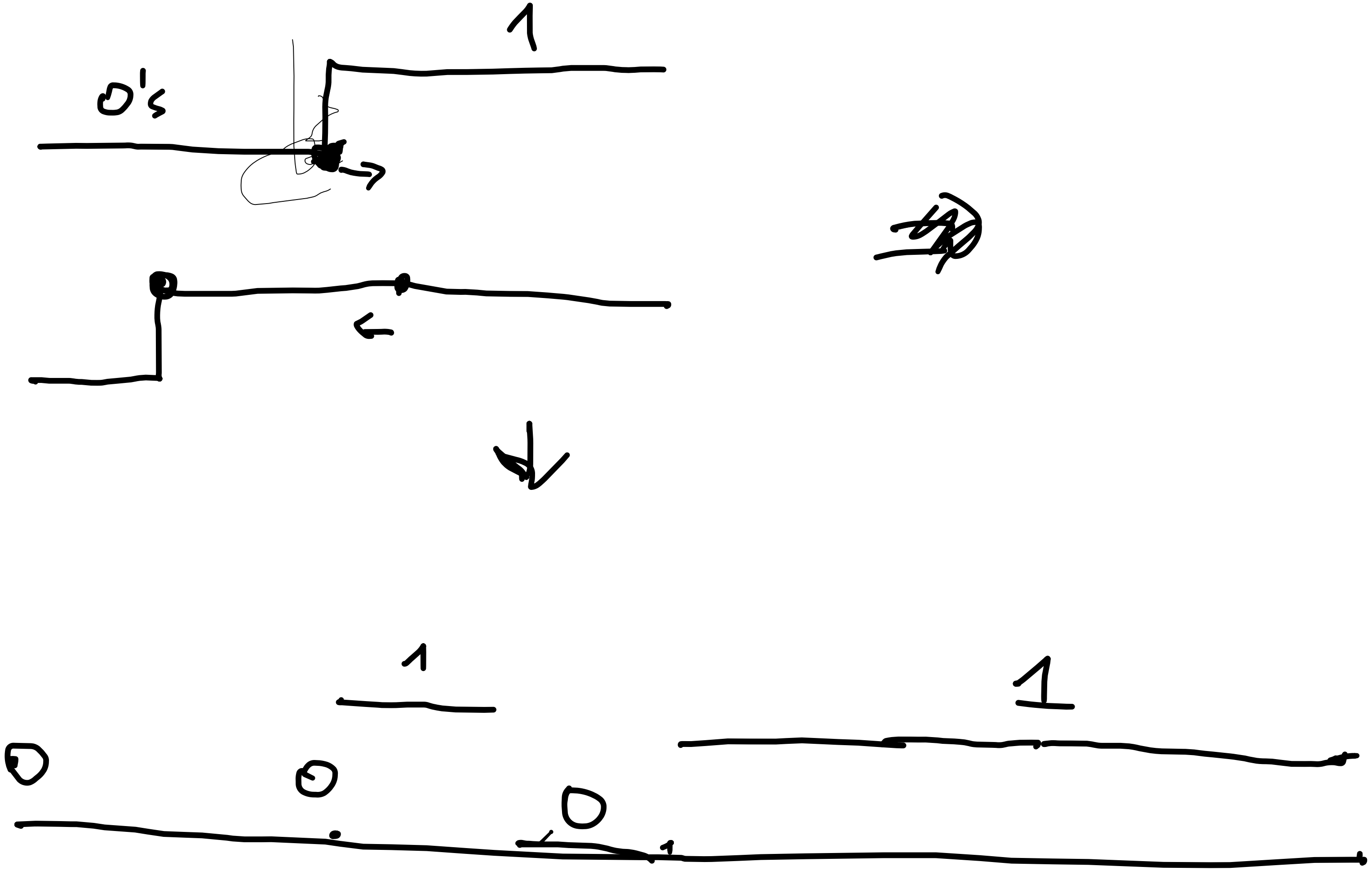


Algorithm 4.16 Merging Network

1: A merging network of width n is a merger of width n followed by two bitonic sequence sorters of width $n/2$. A merger is a depth-one network where we compare wire i with wire $n-i+1$, for $i = 1, \ldots, n/2$.

Lemma 4.17. *A merging network of width n (Algorithm 4.16) merges two sorted input sequences of length $n/2$ each into one sorted sequence of length n.*




1
0's
1
1
0
0
0

Algorithm 4.18 Batcher's "Bitonic" Sorting Network

A batcher sorting network of width n consists of two batcher sorting networks of width $n/2$ followed by a merging network of width n. (See Figure 4.19.)

A batcher sorting network of width 1 is empty.

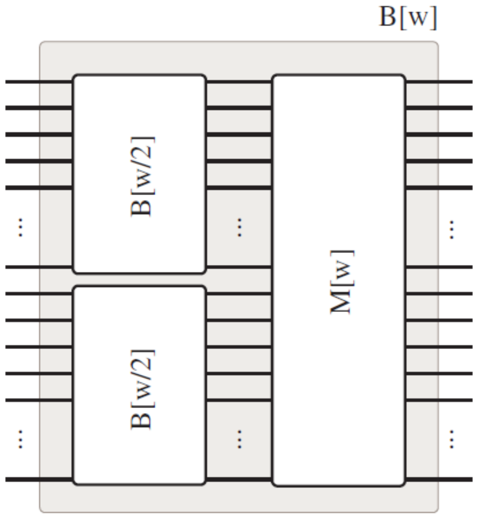
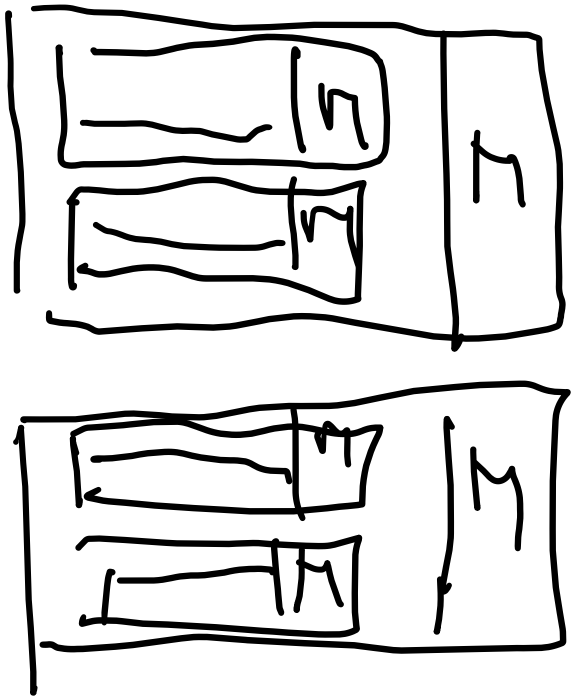
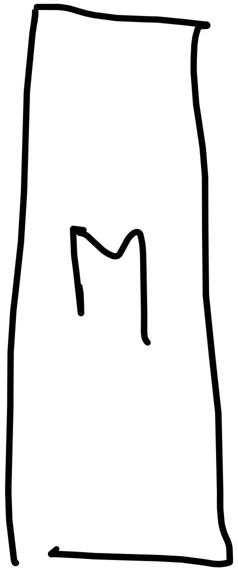


Figure 4.19: A batcher sorting network





number of „layers" is $\approx \log^2 n$

Merger – $O(\log n)$

$\log n$ – depth of recursion

Intersting theoretical result:

$\exists$ sorting network of depth $O(\log n)$

AKS network

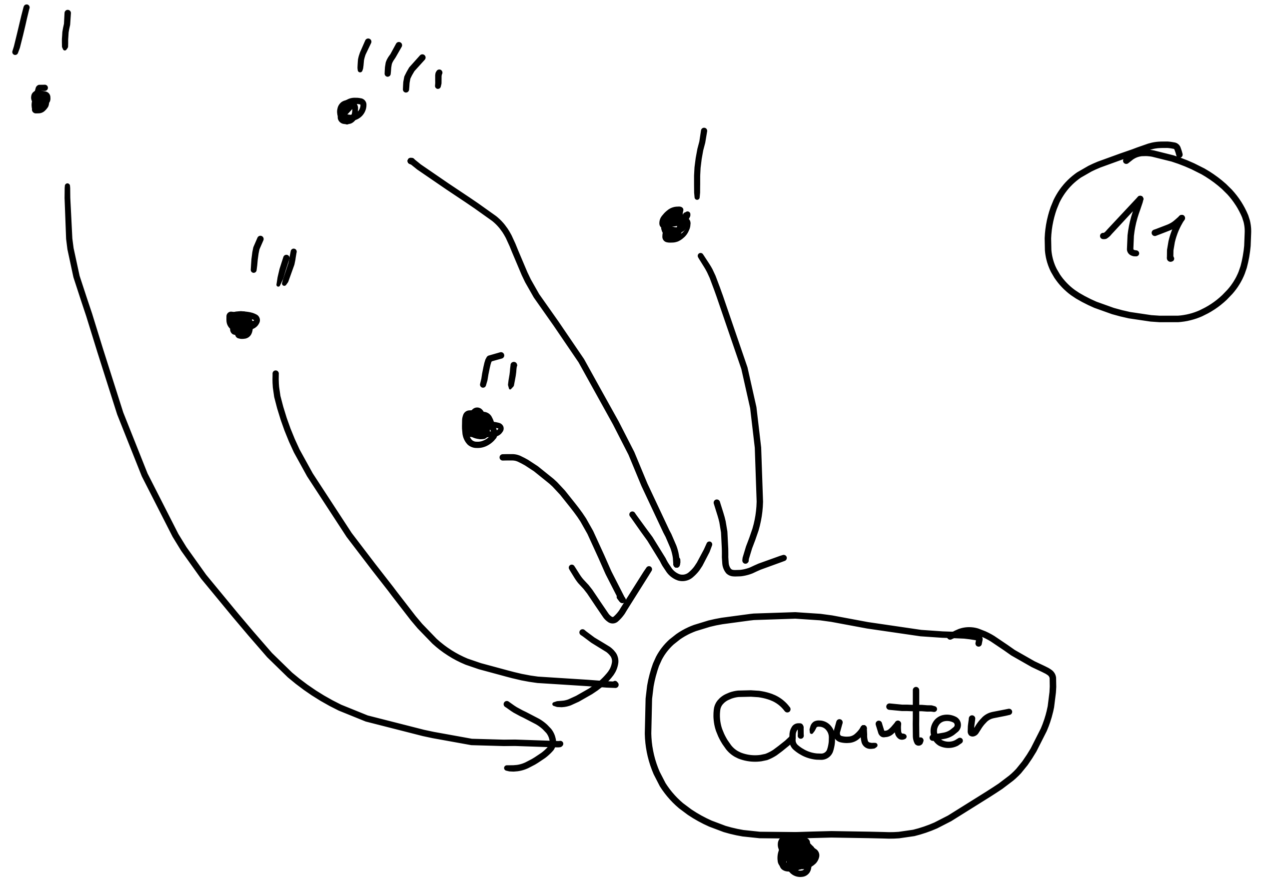
Ajtai, Komlos, Szemeredi

nice: $O(\log n) \approx$ thousands of n

$c \cdot \log n \geq \log^2 n$, for $\log n < c$, $n \leq 2^{\overset{1000}{c}} \approx \infty$

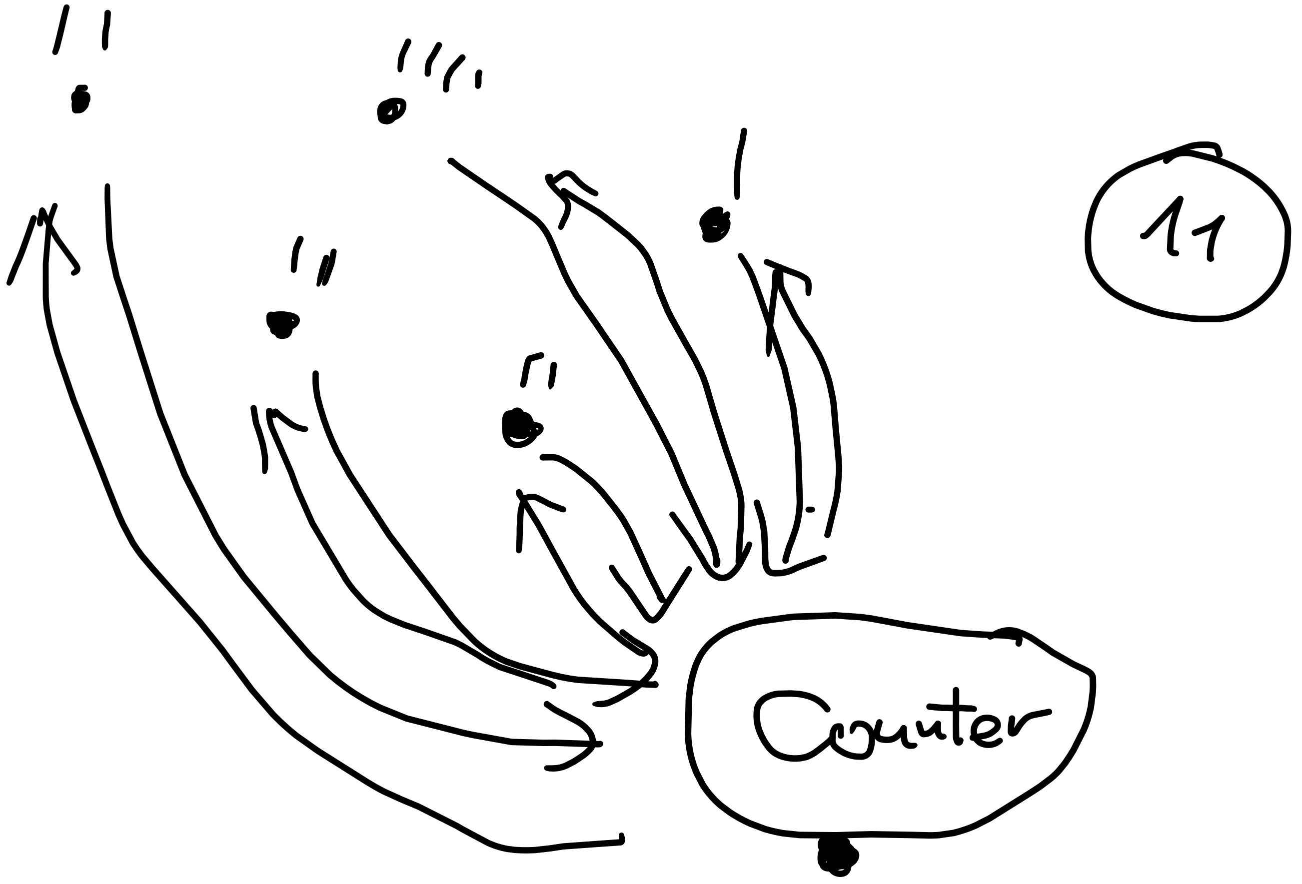
Distributed counting

naive: central counter



Distributed counting

naive: central counter



state

c_1	c_2	c_3	c_4	c_5
0	0	0	0	0

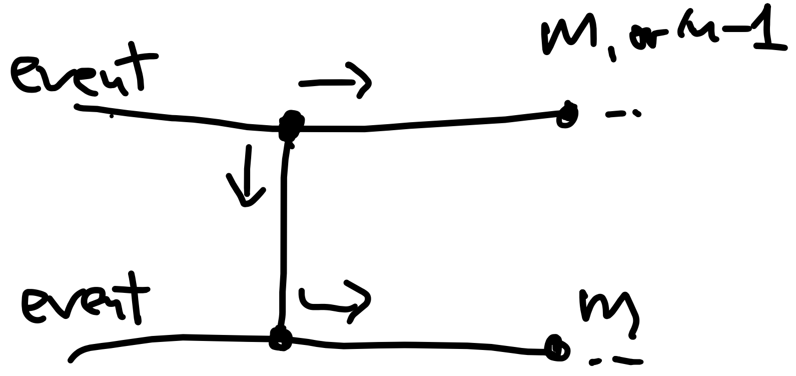
$c_1 \approx c_2 \approx c_3 \approx c_4 \approx c_5 \approx$ number of events

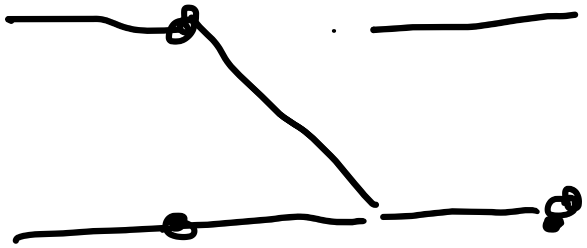
number of events is m

of nodes: n

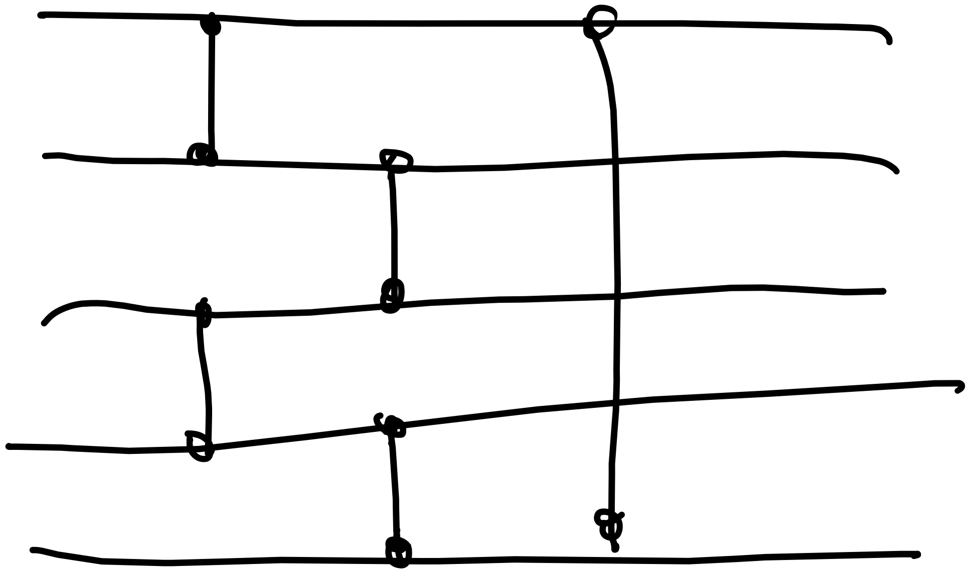
counter values: $\approx \frac{m}{n}$ $\qquad \lfloor \frac{m}{n} \rfloor, \lceil \frac{m}{n} \rceil$

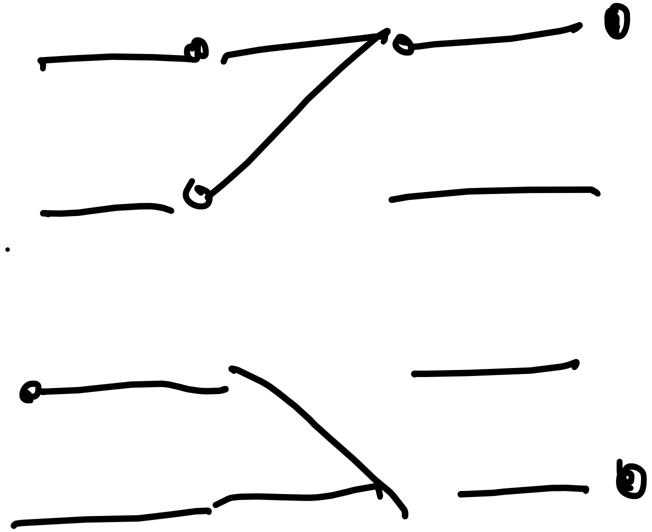
balancer:





counter network:



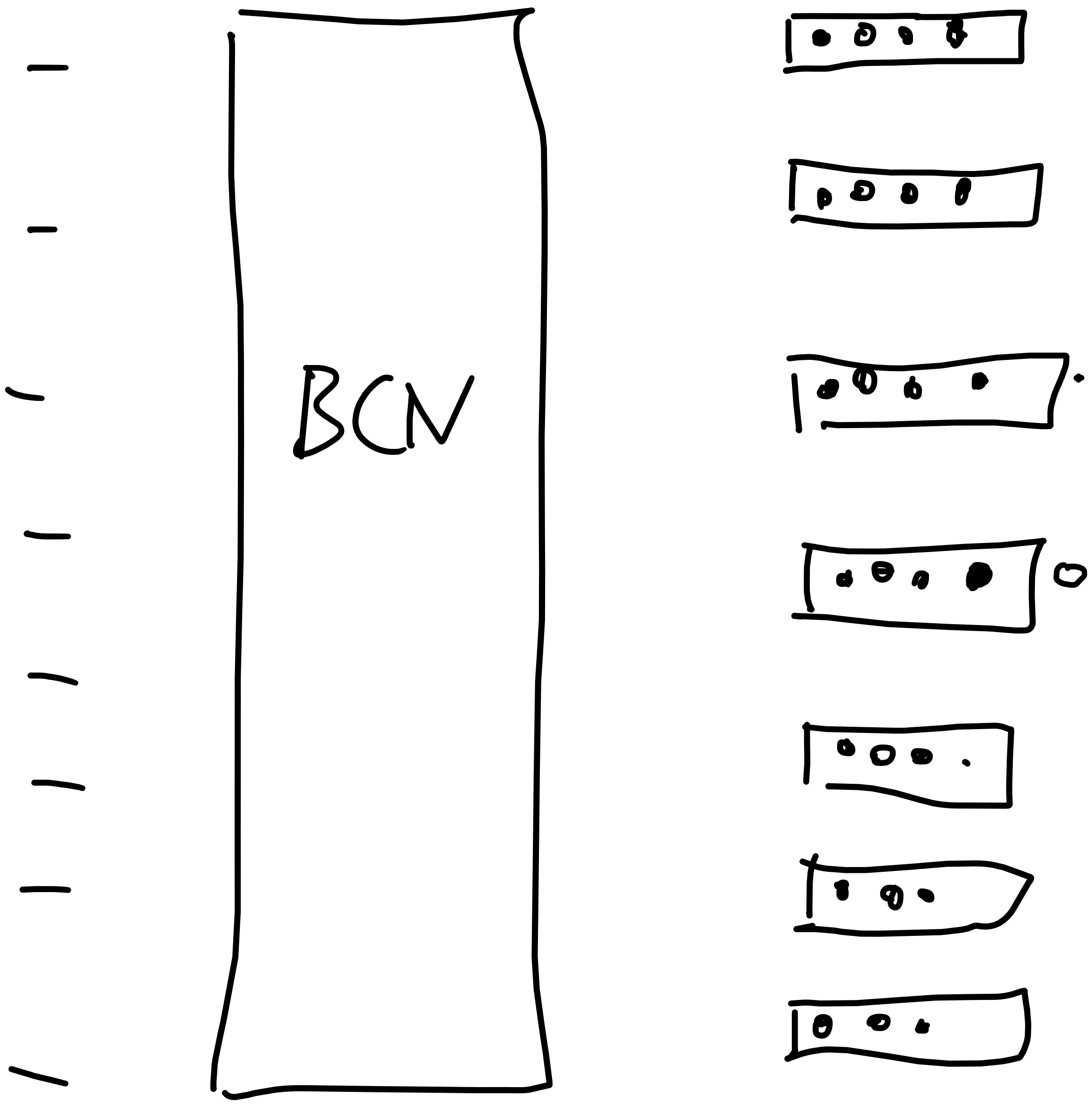


Algorithm 4.23 Bitonic Counting Network.

1: Take Batcher's bitonic sorting network of width w and replace all the comparators with balancers.
2: When a node wants to count, it sends a message to an arbitrary input wire.
3: The message is then routed through the network, following the rules of the asynchronous balancers.
4: Each output wire is completed with a "mini-counter."
5: The mini-counter of wire k replies the value "$k + i \cdot w$" to the initiator of the i^{th} message it receives.

BN

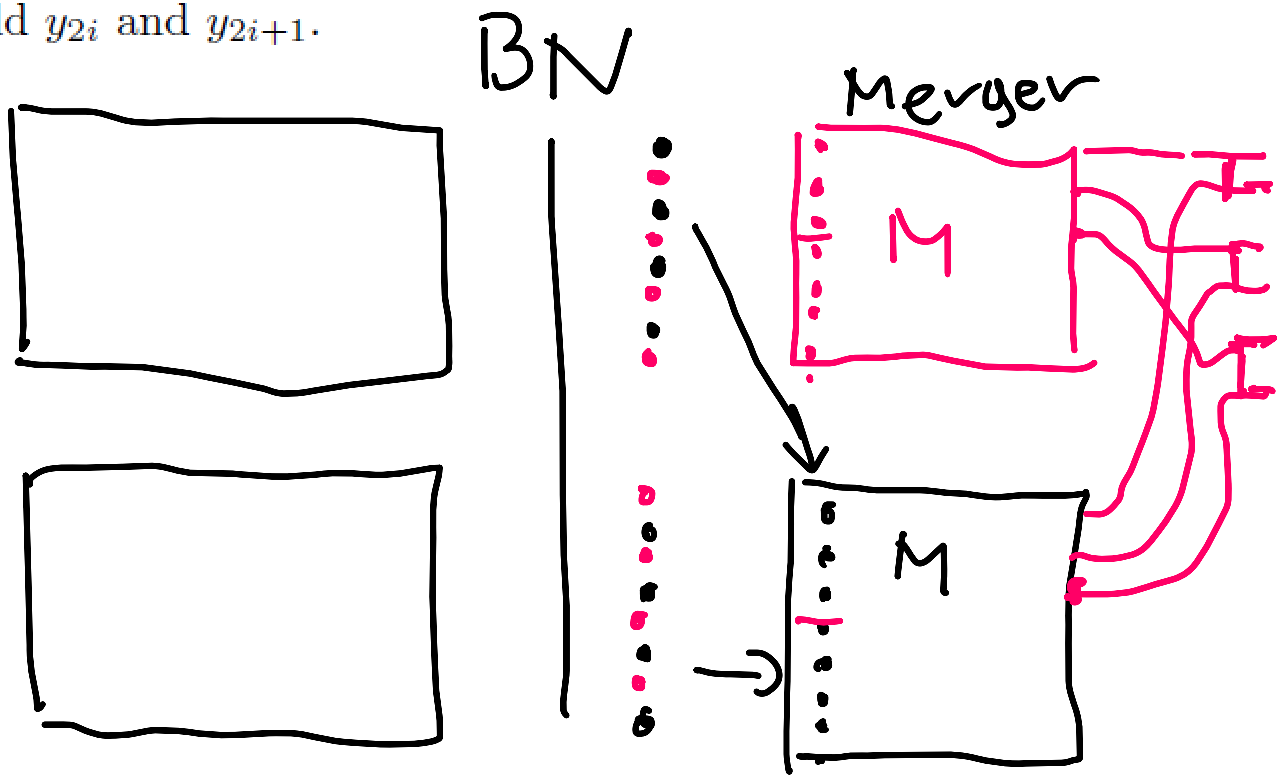
output, mini-counter for $\frac{m}{5}$

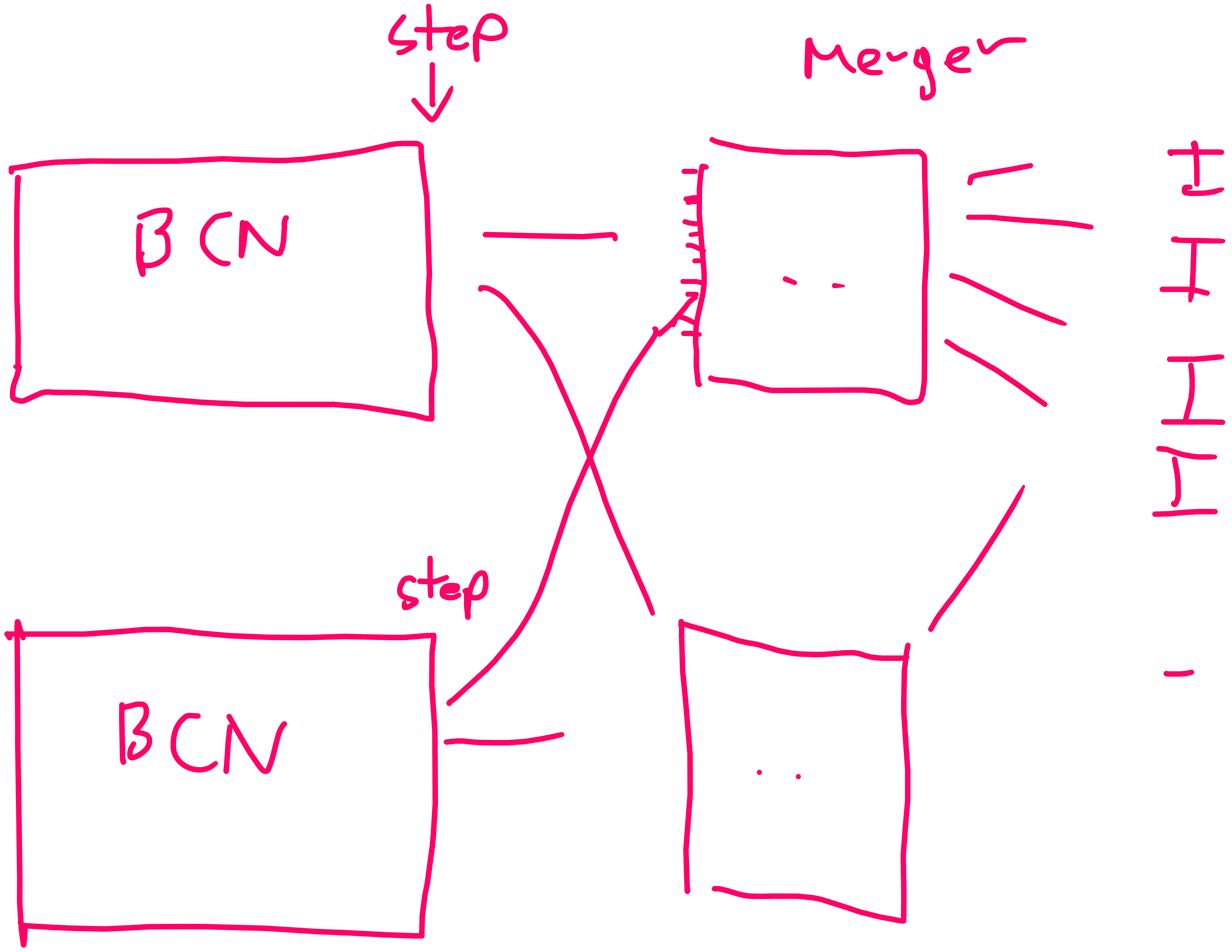


e<

$$3 \cdot n + 4 \leq 4n + 3$$

- An alternative representation of Batcher's network has been introduced in [AHS94]. It is isomorphic to Batcher's network, and relies on a Merger Network $M[w]$ which is defined inductively: $M[w]$ consists of two $M[w/2]$ networks (an upper and a lower one) whose output is fed to $w/2$ balancers. The upper balancer merges the even subsequence $x_0, x_2, \ldots, x_{w-2}$, while the lower balancer merges the odd subsequence $x_1, x_3, \ldots, x_{w-1}$. Call the outputs of these two $M[w/2]$, z and z' respectively. The final stage of the network combines z and z' by sending each pair of wires z_i and z'_i into a balancer whose outputs yield y_{2i} and y_{2i+1}.




step
Merger
BCN
step
BCN

a
a
a
a
⋮
a+1
a+1
⋮
a+1

b
⋮
b
b+1
⋮
b+1

a
a
a
a+1
a+1
⋮
a+1

b
b
b+1
b+1
b+1

a
a
a
a+1
a+1
⋮
a+1

b
b
b+1
b+1
b+1

a
a
a
a
a
a+1
a+1

b
b
b
b
b
b+1
b+1

A=
B+

a
a
a
a+1
b
b
b+1
b+1

A+
B=

a
a
a+1
a+1

b
b
b
b+1

Definition 4.24 (Step Property). *A sequence $y_0, y_1, \ldots, y_{w-1}$ is said to have the* step property, *if $0 \leq y_i - y_j \leq 1$, for any $i < j$.*

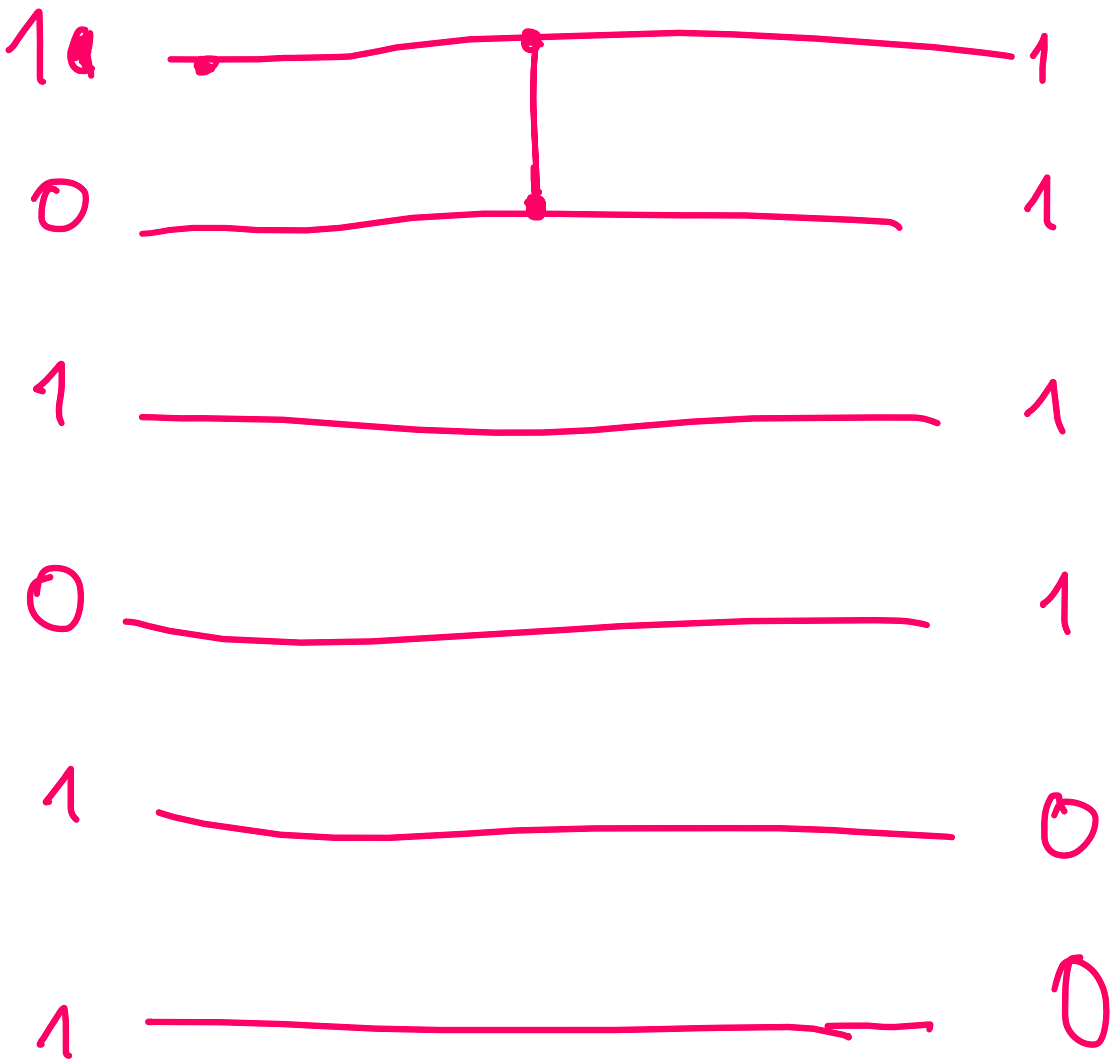
- y_1 a
- y_2 a
- y_3 a
- y_4 $a+1$

$a+1$

$\vdots$

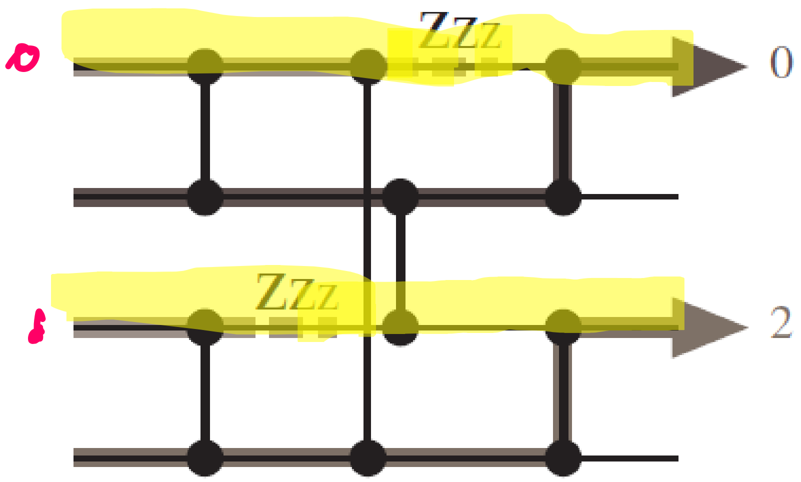
Lemma 4.28. *Let $M[w]$ be a merger network of width w. In a quiescent state (no message in transit), if the inputs $x_0, x_1, \ldots, x_{w/2-1}$ resp. $x_{w/2}, x_{w/2+1}, \ldots, x_{w-1}$ have the step property, then the output $y_0, y_1, \ldots, y_{w-1}$ has the step property.*

Counting network $\overset{\not\leftarrow}{\longrightarrow}$ sorting network

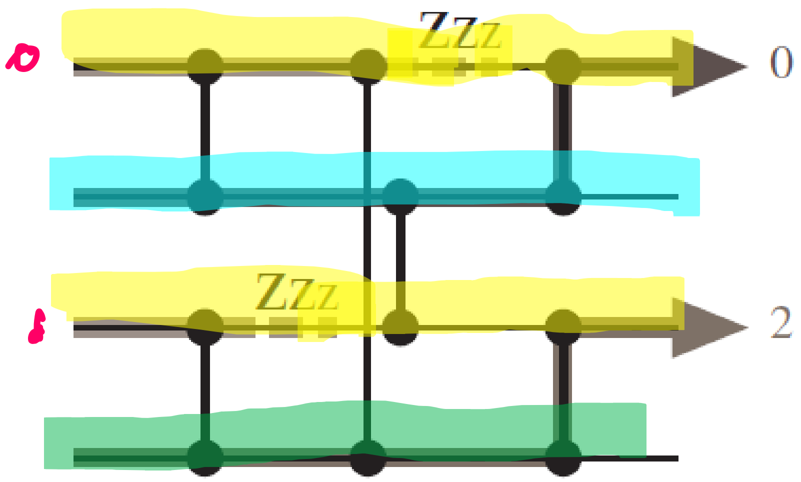


Example:
Odd-even sorter is not counter

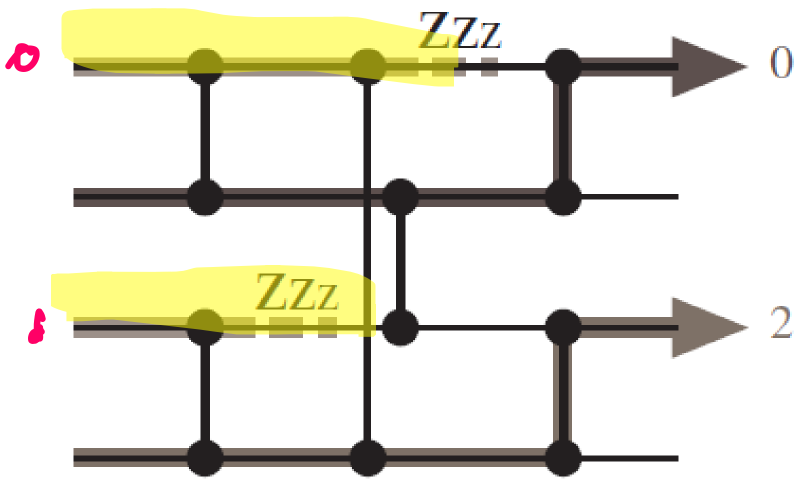
asynchronous!



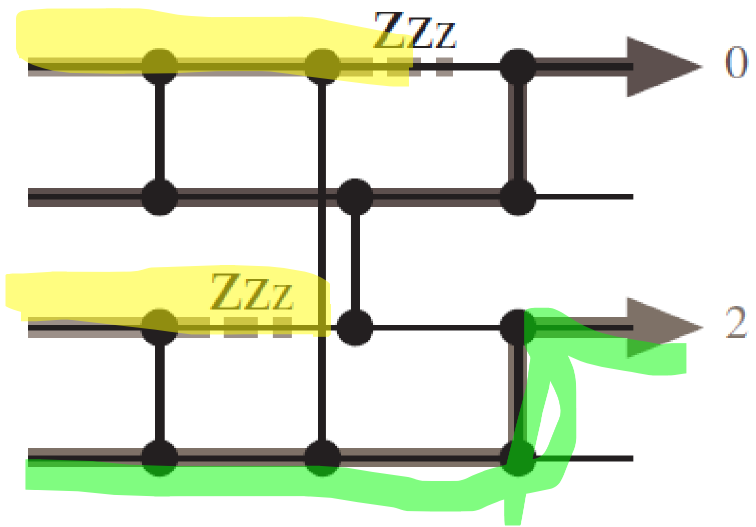
0
Zzz
0
Zzz
2

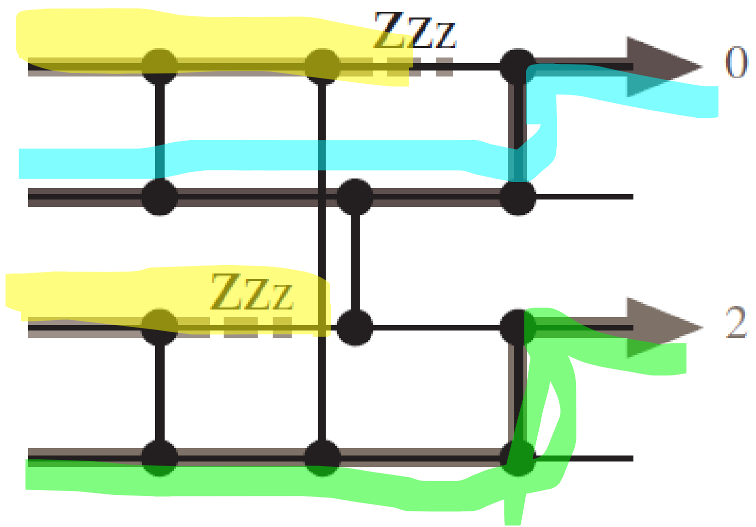


Zzz
0
Zzz
2

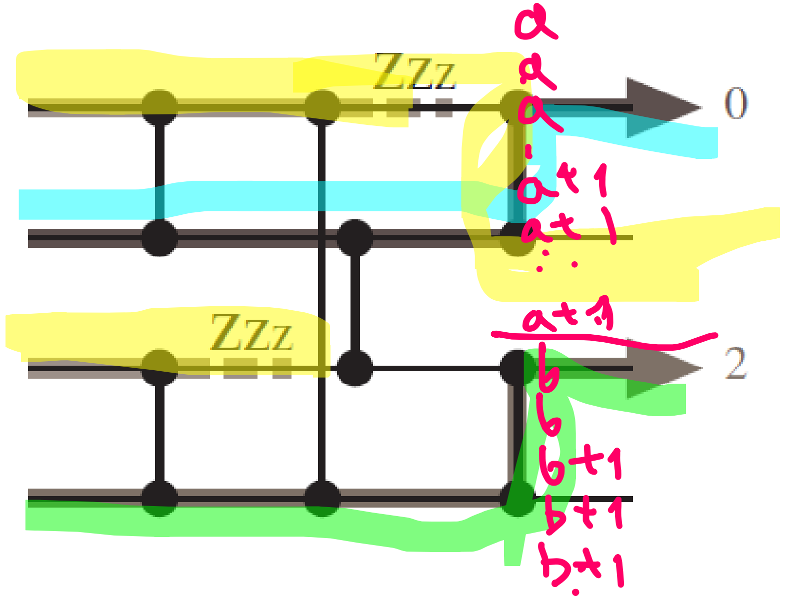


ZZz
0
ZZz
2


Zzz
0
Zzz
2



Zzz
0
Zzz
2


Zzz
Zzz
0
2
a
a
a
a+1
a+1
a+1
b
b
b+1
b+1
b+1