HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Periodic merging networks

Mirek Kutyłowski[ab],

Krzysztof Loryś[cb]

Brigitte Oesterdiekhoff[a]

7th ISAAC, Osaka'96

[a]Paderborn

[b]Wrocław

[c]Trier

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Merging problem:

Given two **sorted** sequences A and B.

Sort the set $A \cup B$.

Computational model:

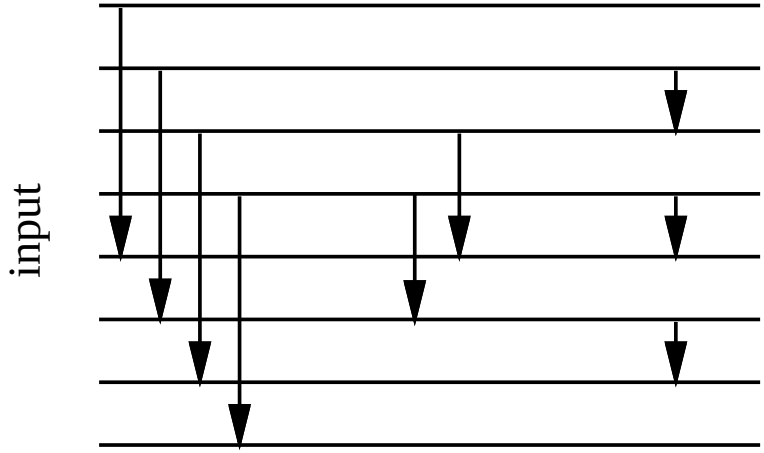
- only *compare-exchange* operations allowed,
- parallel setting: many disjoint compare-exchange operations may be executed simultaneously between nodes of a network

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

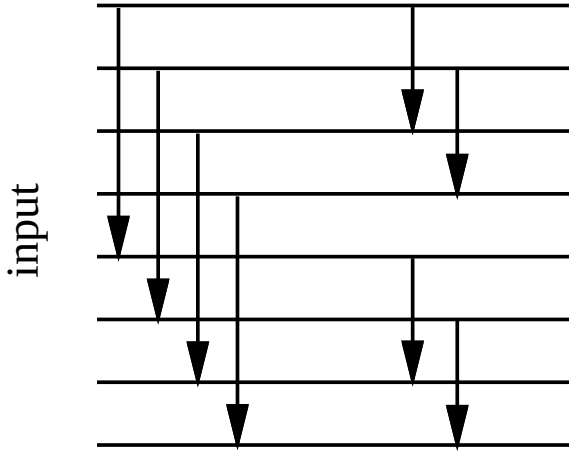
Classical results - Batcher Networks

Runtime: $\log(2n)$

Odd Even Merge, '68



Bitonic Merge Sort,

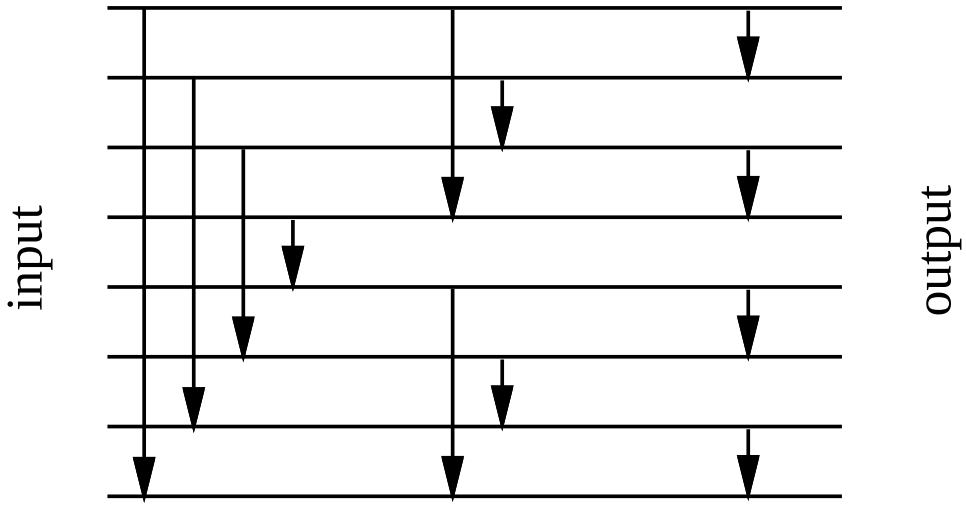


HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Classical results - Balanced Networks

Runtime: $\log(2n)$

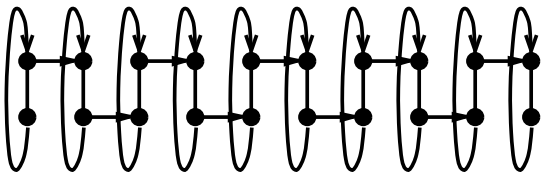
Balanced Merging Network
(Dowd, ..., '89)



HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Periodic networks

- the classical merging networks perform different operations at each step – inconvenient for VLSI implementations
- periodic networks: the same steps performed periodically, easy to implement on networks of constant degree



HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Known results on periodic networks with constant period

Sorting:

Period	Runtime	
2	$\Theta(n)$	Knuth
8	$O(\sqrt{n} \cdot \log n)$	Schwiegelshohn, 88
6	$O(\sqrt{n \cdot \log n})$	Krammer, 91
$O(k)$	$O(k^2 \cdot n^{1/k})$	Kik, Kutyłowski, Stachowiak, 92
3	$O(\log^2 n)$	Kutyłowski, Loryś, Oesterdiekhoff, Wanka, 94

Merging: no extra results for merging (only through sorting)

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

New results

A family of merging networks with **constant** periods and runtime $O(\log n)$:

Period	Runtime
3	$12 \cdot \log n$
4	$9 \cdot \log_3 n \approx 5.67 \log n$
$k+3$	$2.25 \cdot \frac{k+3}{k-1+\log 3} \cdot \log n \approx 2.25 \log n$

Novel features:

- no recursive structure
- completely different approach than the previous merging algorithms

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

01-Principle for merging

An algorithm A consisting of a fixed collection of compare-exchange operations solves the merging problem

iff

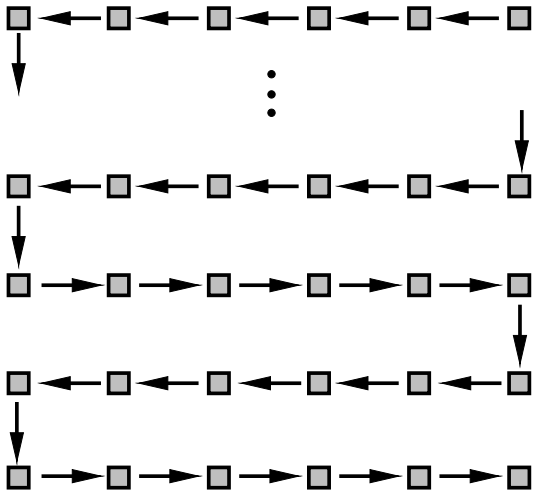
A merges input sequences consisting of zeroes and ones.

Behavior of our algorithms is understandable only for 01-sequences

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

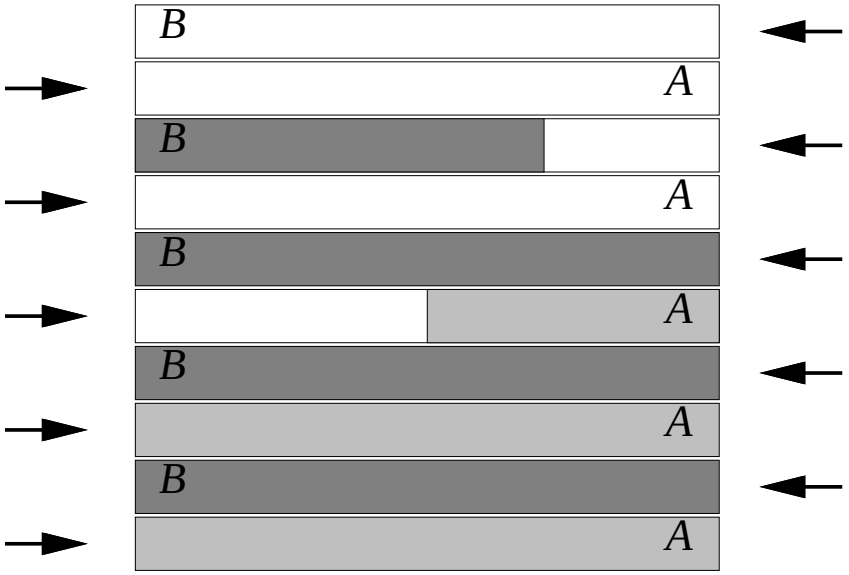
Architecture of our network

- 2-dimensional structure
- snake-like ordering



HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Input allocation

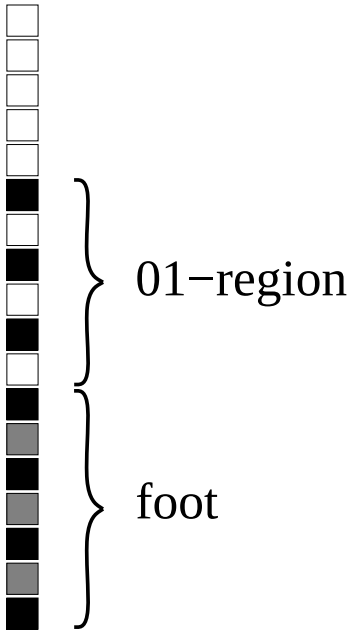


HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Crucial properties of the algorithms

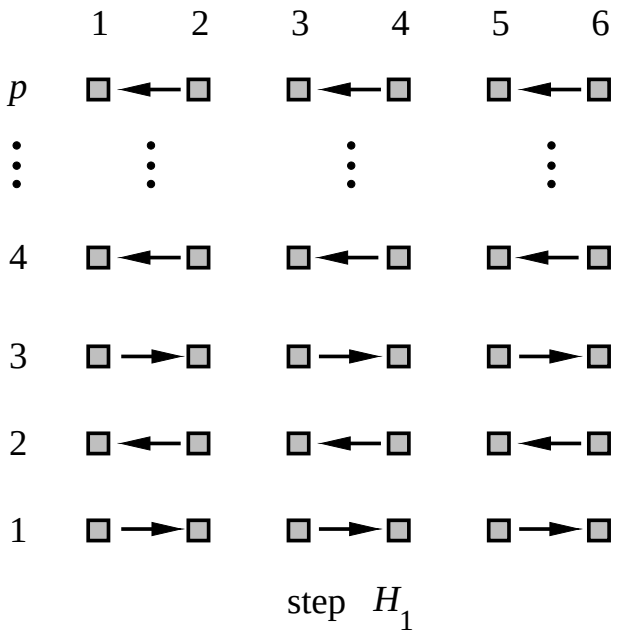
During the whole computation:

- the number of ones in two columns may differ by at most 1
- each column has the form $1^d(01)^l0^*$

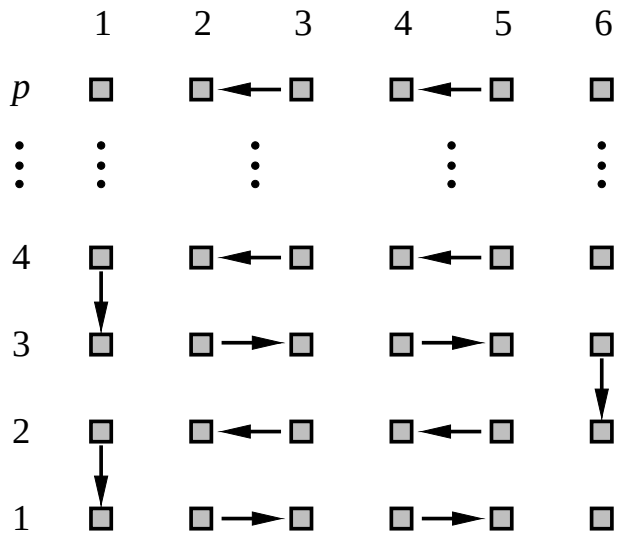


HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Horizontal steps



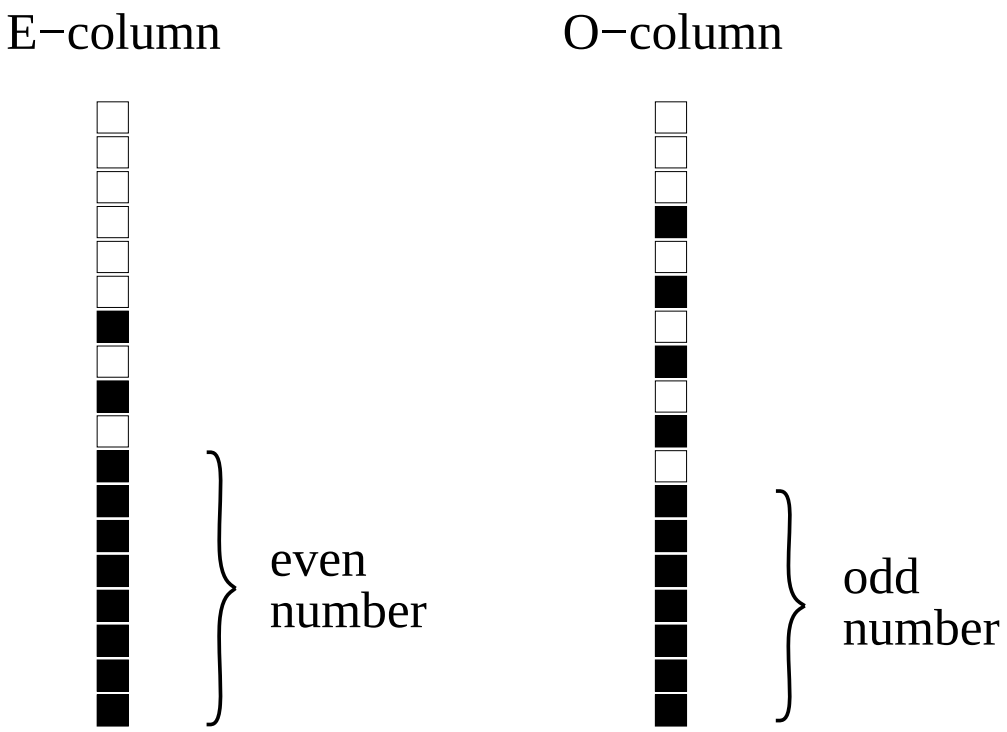
step H_1



step H_2

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

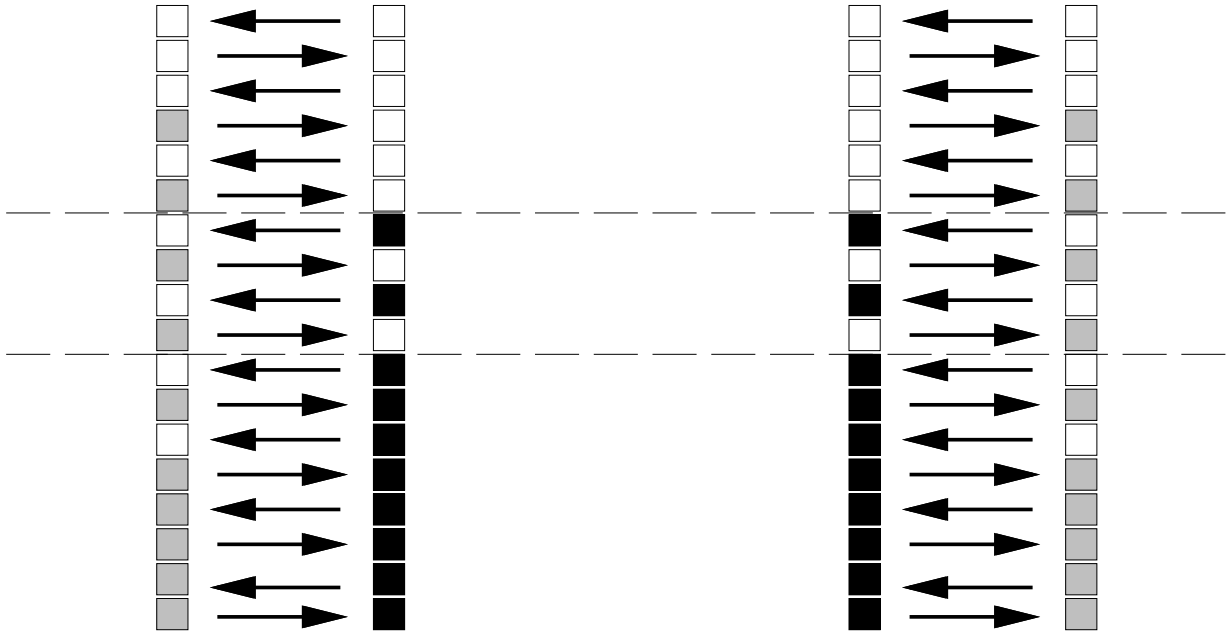
Classification of columns



HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Moving columns

an O-column and E-column compared at a horizontal step exchange their positions!



HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

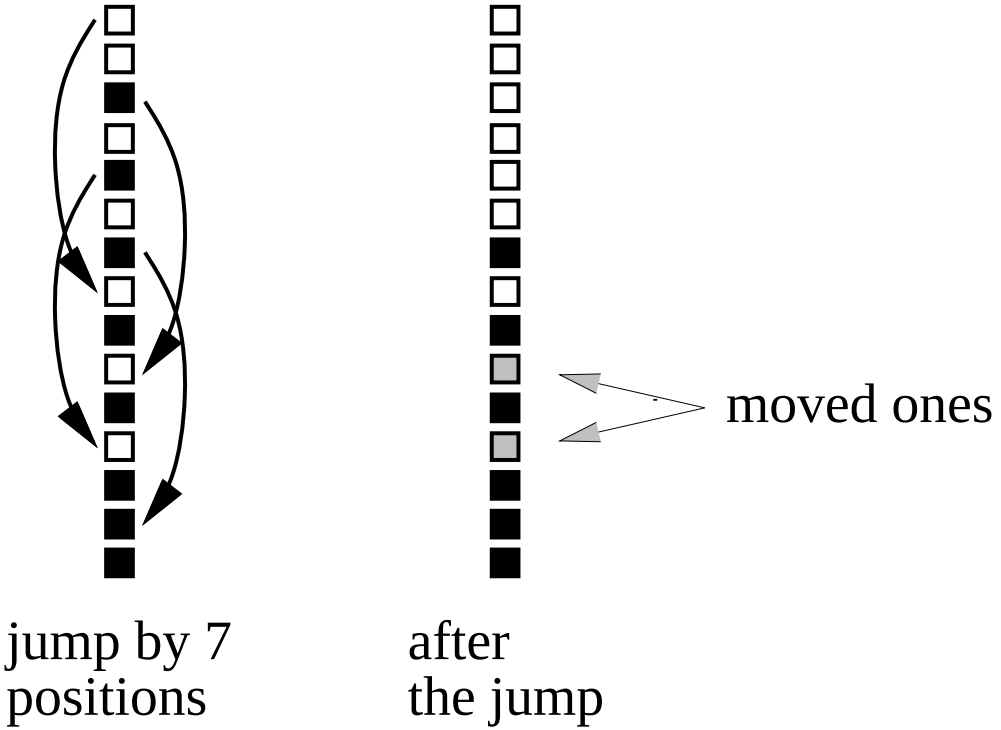
The way made by the moving columns

- During execution of the algorithm each *moving column* travels through the network.
- At different positions different *jumps* reorder a moving column.
- The number of different jumps is *not* a constant.

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Jump steps

Goal: reduce the size of the 01-region



HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Runtime of the algorithms

Sorting the columns :

- time needed to reach a *starting position* by the moving column — $O(q)$
- time for applying the jumps — $O(q)$
  $\Omega(\log n)$ jumps $\quad\Rightarrow\quad$ $q = \Omega(\log n)$

Relocating the sorted columns — $O(q)$

$\Rightarrow$ q should be as small as possible
$q = \Omega(\log n)$

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Getting q small $\Rightarrow$ 1/3-jumps

Intuitive: a jump halves the maximal size of an 01-region

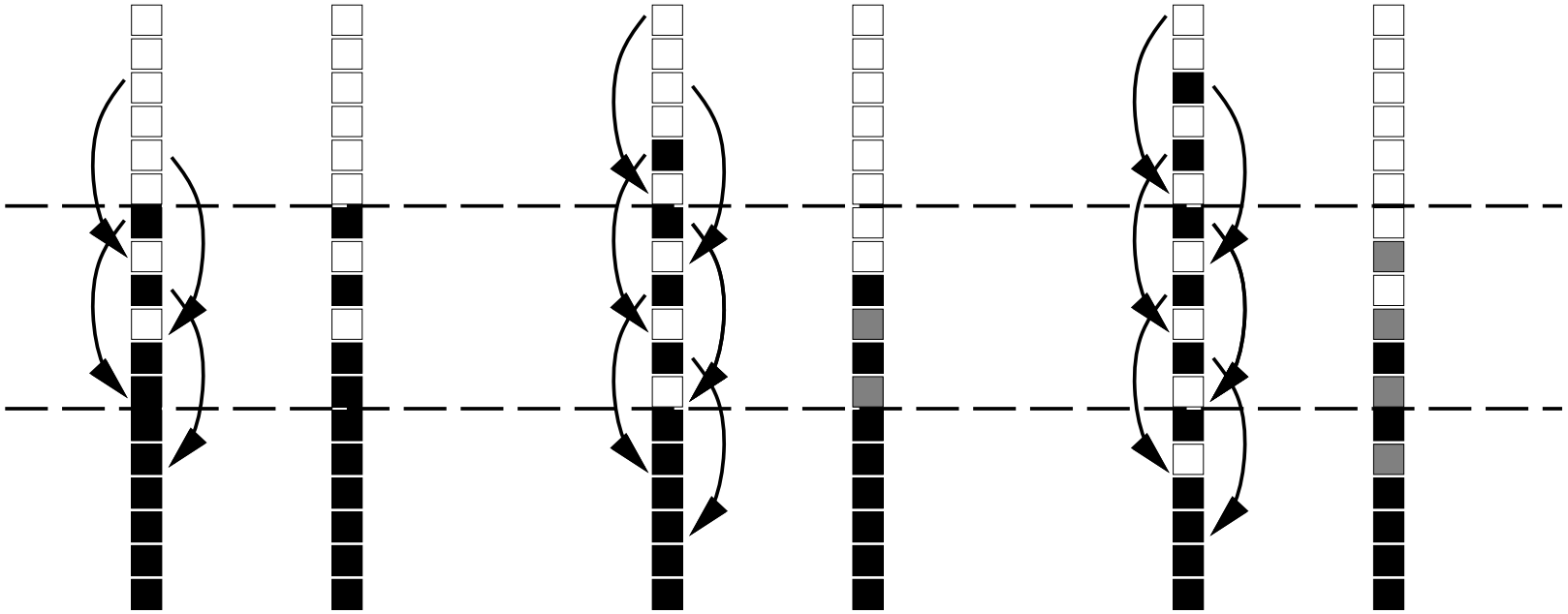
We get more: a jump reduces the maximal size of an 01-region to 1/3 of the original size.

l = the jump size
b = actual size of the 01-region
$3l$ = maximal size of the 01-region

HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

1/3-jumps: how it works

case 1: $b < l$ case 2: $l <= b < 2l$ case 3: $2l <= b <= 3l$



nothing happens

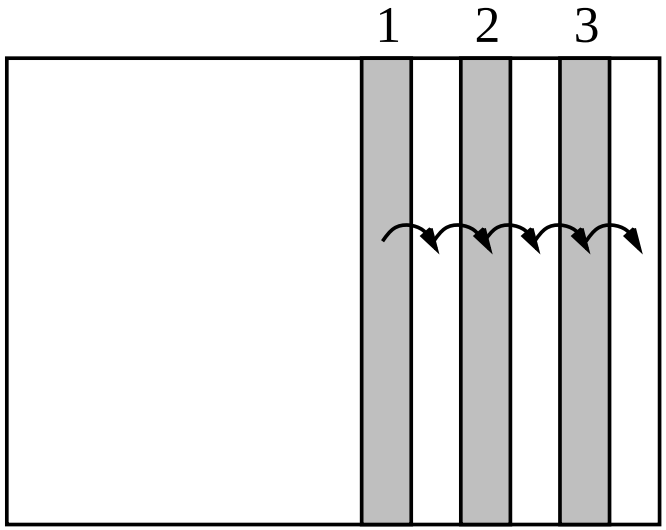
reduced below l

reduced below l

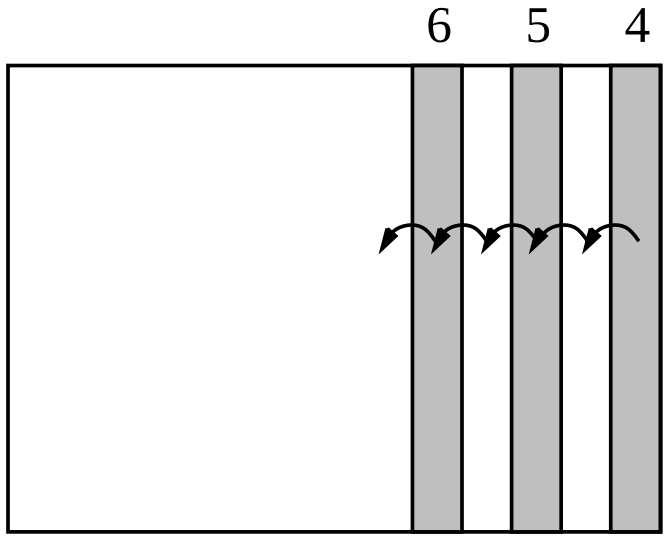
HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Getting q small $\Rightarrow$ wrapping around

- two horizontal steps necessary
- anyway: jumps at each position



while moving to the right



while moving to the left

$\Rightarrow$ $q =$ the number of jumps necessary

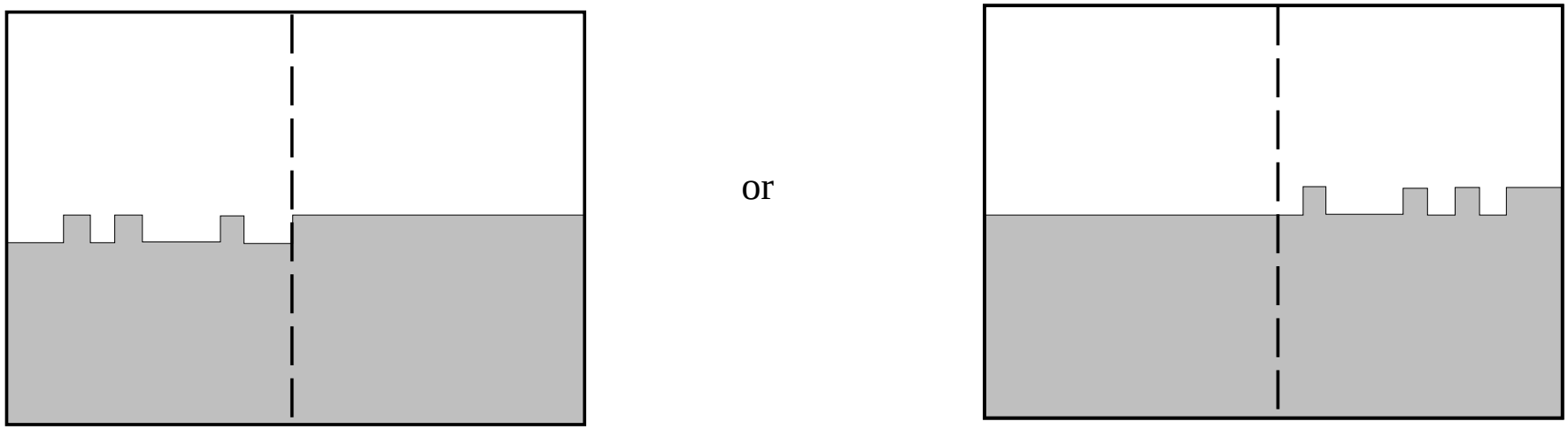
HEINZ NIXDORF INSTITUT
Universität-GH Paderborn
Brigitte Oesterdiekhoff
brigitte@uni-paderborn.de

Trick: starting and ending in the middle

Idea: last jump required for sorting executed in **the middle column**

$\Rightarrow$ sorted columns emerge in the middle!

Situation after sorting the last column



$\Rightarrow$ only a region of width $q/2$ needs to be sorted

($q/2$ steps of Odd-Even Transposition Sort suffice)