ETH zürich

spcl.inf.ethz.ch
@spcl_eth

D INFK

TORSTEN HOEFLER

Parallel Programming

Parallel Sorting (A taste of parallel algorithms!)

On the usage of sorting networks to control greenhouse climatic factors

Blanca C López Ramírez, Giovanni Guzmán, Wadee Alhalabi, more... Show all authors

First Published February 10, 2018 | Research Article | Check for updates

Download PDF | Article information

Abstract

The goal of this article is the application of a non-adaptive classification algorithm to support the variable management process for internal climate control. The protected agriculture has given many advantages for the care and improvement in the production of almost any food. This work is focused on improving a control system for climate variables. The decision for activating an actuator for the correct care of the crop is very important. A sorting network technique with

Sorting networks and their applications - ACM Digital Library

https://dl.acm.org/citation.cfm?id=1468121

by KE Batcher - 1968 - Cited by 2919 - Related articles

David C. Van Voorhis, An economical construction for **sorting networks**, Proceedings of the May 6-10, 1974, national computer conference and exposition, May ...

You visited this page on 5/29/18.

Leadership Computing for Europe and the Path to Exascale Computing

May 29, 2018 by Rich Brueckner | Leave a Comment

In this video from the NVIDIA GPU Conference, Thomas Schulthess from CSCS presents: *Leadership Computing for Europe and the Path to Exascale Computing.*

Thomas Schulthess is the Director of CSCS in Switzerland.

"With over 5000 GPU-accelerated nodes, Piz Daint has been Europes leading supercomputing systems since 2013, and is currently one of the most performant and energy efficient supercomputers on the planet. It has been designed to optimize throughput of multiple applications, covering all aspects of the workflow, including data analysis and visualisation. We will discuss ongoing efforts to further integrate these extreme-scale compute and data services with infrastructure services of the cloud. As Tier-0 systems of PRACE, Piz Daint is accessible to all scientists in Europe and worldwide. It provides a baseline for future development of exascale computing. We will present a strategy for developing exascale computing technologies in domains such as weather and climate or materials science."

Optimal-depth sorting networks

Authors:
- Daniel Bundala — Department of Computer Science, University of Oxford, United Kingdom
- Michael Codish — Department of Computer Science, Ben-Gurion University of the Negev, Israel
- Luís Cruz-Filipe — Department of Mathematics and Computer Science, University of Southern Denmark, Denmark
- Peter Schneider-Kamp — Department of Mathematics and Computer Science, University of Southern Denmark, Denmark
- Jakub Závodný — Department of Computer Science, University of Oxford, United Kingdom

2017 Article

Bibliometrics
- Citation Count: 0
- Downloads (cumulative): n/a
- Downloads (12 Months): n/a
- Downloads (6 Weeks): n/a

Published in:
· Journal
Journal of Computer and System Sciences archive
Volume 84 Issue C, March 2017
Pages 185-204
Academic Press, Inc. Orlando, FL, USA
table of contents doi>10.1016/j.jcss.2016.09.004

Tools and Resources

TOC Service: Email RSS

Save to Binder

Export Formats: BibTeX EndNote ACM Ref

Share:

Author Tags

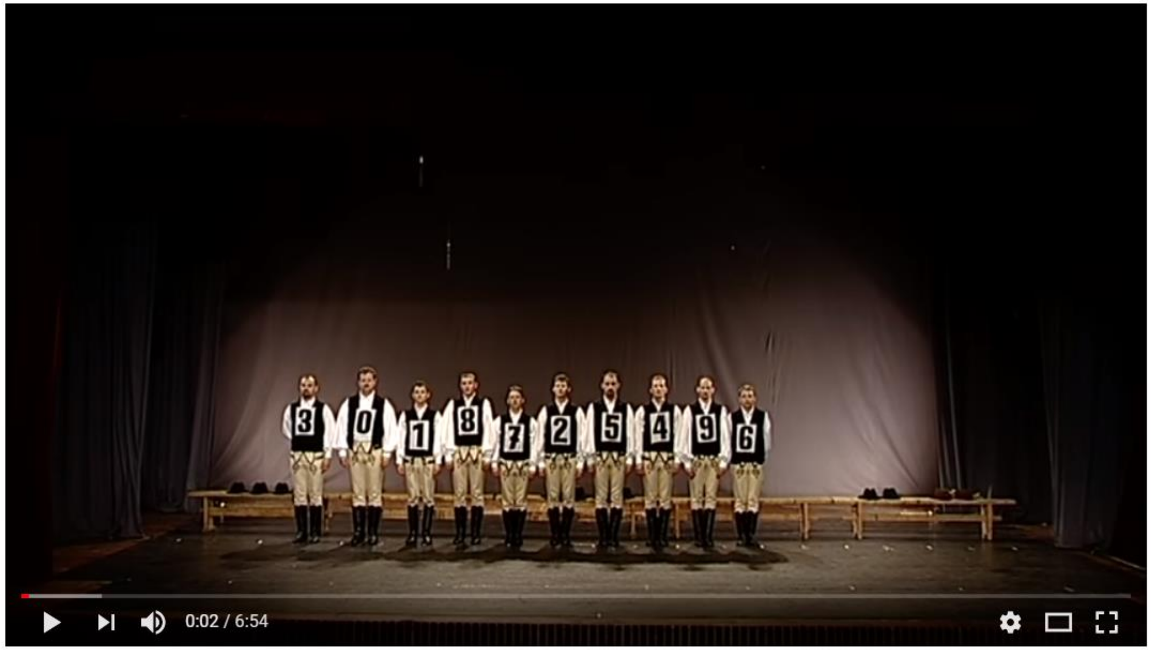
Contact Us | Switch to single page view (no tabs)

Abstract | Authors | References | Cited By | Index Terms | Publication | Reviews | Comments | Table of Contents

We prove depth optimality of sorting networks from "The Art of Computer Programming".Sorting networks posses symmetry that can be used to generate a few representatives.These representatives can be efficiently encoded using regular expressions.We construct SAT formulas whose unsatisfiability is sufficient to show optimality.Resulting algorithm is orders of magnitude faster than prior work on small instances. We solve a 40-year-old open problem on depth optimality of sorting networks. In 1973, Donald E. Knuth detailed sorting networks of the smallest depth known for n ź 16 inputs, quoting optimality for n ź 8 (Volume 3 of "The Art of Computer Programming"). In 1989, Parberry proved optimality of networks with 9 ź n ź 10 inputs. We present a general technique for obtaining such results, proving optimality of the remaining open cases of 11 ź n ź 16 inputs. Exploiting symmetry, we construct a small set R n of two-layer networks such that: if there is a depth-k sorting network on n inputs, then there is one whose first layers are in R n . For each network in R n , we construct a propositional formula whose satisfiability is necessary for the existence of a depth-k sorting network. Using an off-the-shelf SAT solver we prove optimality of the sorting networks listed by Knuth. For n ź 10 inputs, our algorithm is orders of magnitude faster than prior ones.

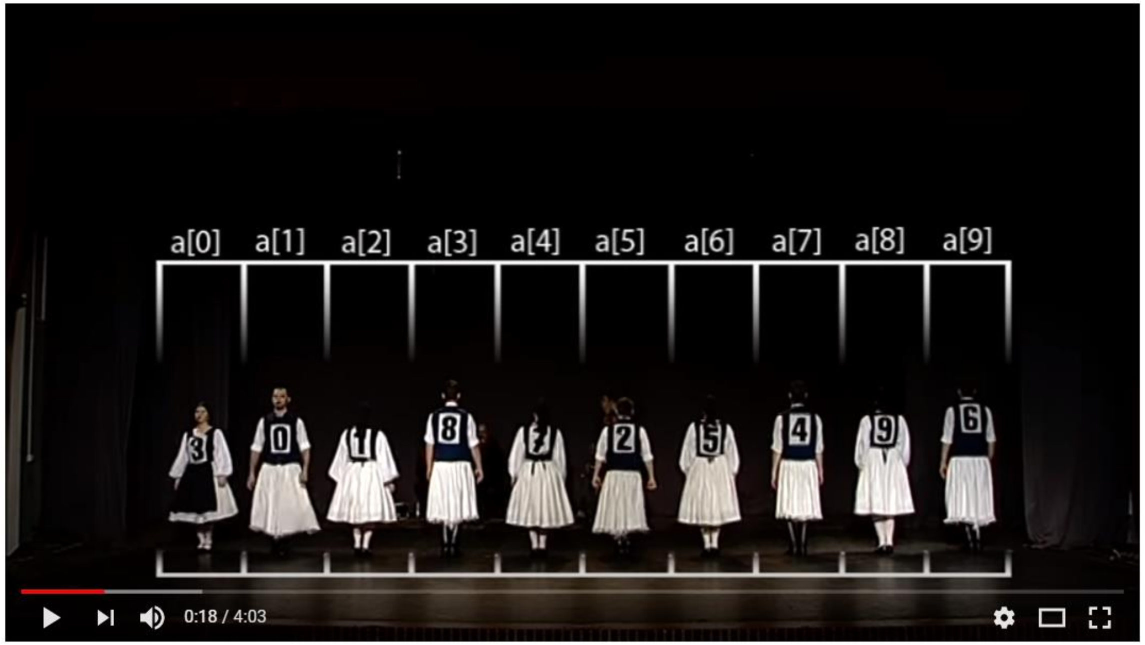
Today: Parallel Sorting

(one of the most fun problems in CS)



Quick-sort with Hungarian (Küküllőmenti legényes) folk dance

1,318,280 views 14K 186 SHARE



Insert-sort with Romanian folk dance

595,979 views 3.8K 57 SHARE

Literature

- D.E. Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Section 5.3.4: Networks for Sorting, pp. 219–247.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 1990. ISBN 0-262-03293-7. Chapter 27: Sorting Networks, pp.704–724.

google

"chapter 27 sorting networks"

How fast can we sort?

Heapsort & Mergesort have $O(n \log n)$ worst-case run time

Quicksort has $O(n \log n)$ average-case run time

These bounds are all tight, actually $\Theta(n \log n)$

So maybe we can dream up another algorithm with a lower asymptotic complexity, such as $O(n)$ or $O(n \log \log n)$

This is unfortunately ***IMPOSSIBLE!***

But why?

Permutations

Assume we have n elements to sort

For simplicity, also assume none are equal (i.e., no duplicates)

How many permutations of the elements (possible orderings)?

Example, n=3

a[0]<a[1]<a[2]	a[0]<a[2]<a[1]	a[1]<a[0]<a[2]
a[1]<a[2]<a[0]	a[2]<a[0]<a[1]	a[2]<a[1]<a[0]

In general, n choices for first, n-1 for next, n-2 for next, etc. → n(n-1)(n-2)…(1) = n! possible orderings

Representing every comparison sort

Algorithm must "find" the right answer among n! possible answers

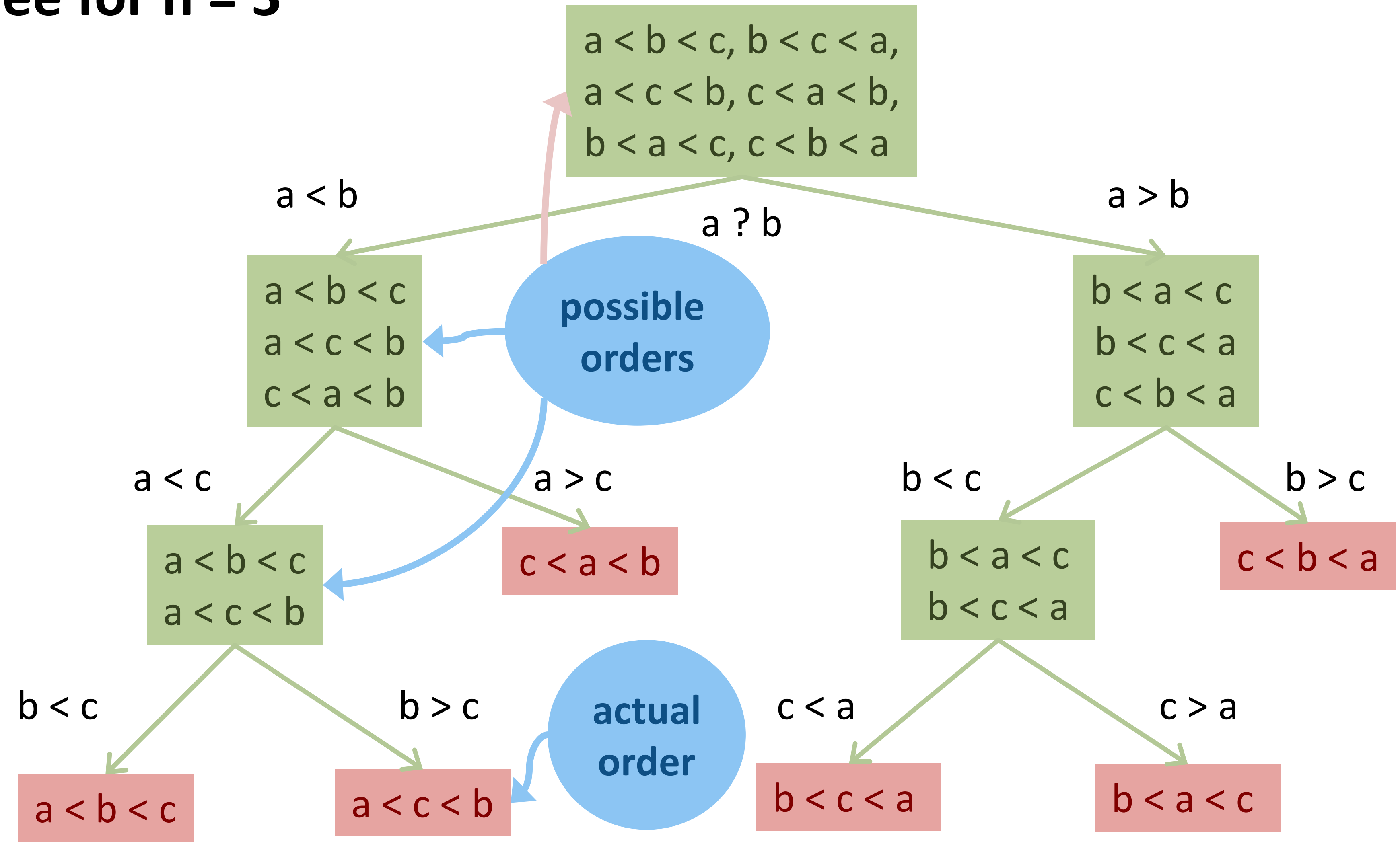
Starts "knowing nothing" and gains information with each comparison

Intuition is that each comparison can, at best,
eliminate half of the remaining possibilities

Can represent this process as a decision tree

- Nodes contain "remaining possibilities"
- Edges are "answers from a comparison"
- This is not a data structure but what our proof uses to represent "the most any algorithm could know"

Decision tree for n = 3



The leaves contain all possible orderings of a, b, c

What the decision tree tells us

Binary tree because

- Each comparison has binary outcome
- Assumes algorithm does not ask redundant questions

Because any data is possible, any algorithm needs to ask enough questions to decide among all n! answers

- Every answer is a leaf (no more questions to ask)
- So the tree must be big enough to have n! leaves
- Running any algorithm on any input will at best correspond to one root-to-leaf path in the decision tree

So no algorithm can have worst-case running time better than the height of the decision tree

Where are we

Proven: No comparison sort can have worst-case better than the height of a binary tree with n! leaves

- Turns out average-case is same asymptotically
- So how tall is a binary tree with n! leaves?

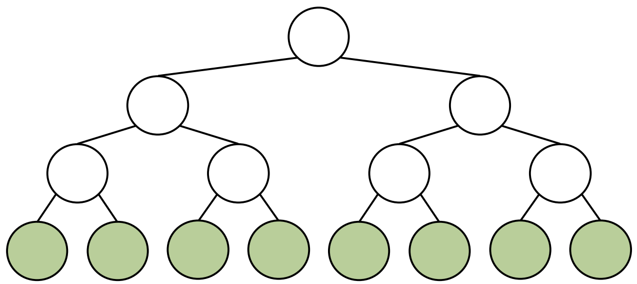
Now: Show a binary tree with n! leaves has height Ω(n log n)

- *n log n* is the lower bound, the height must be at least this
- It could be more (in other words, a comparison sorting algorithm could take longer but can not be faster)

Conclude that: (Comparison) Sorting is Ω(n log n)

Lower bound on height

The height of a binary tree with L leaves is at least $\log_2 L$



So the height of our decision tree, h:

$h \geq \log_2 (n!)$	property of binary trees
$= \log_2 (n*(n-1)*(n-2)\ldots(2)(1))$	definition of factorial
$= \log_2 n + \log_2 (n-1) + \ldots + \log_2 1$	property of logarithms
$\geq \log_2 n + \log_2 (n-1) + \ldots + \log_2 (n/2)$	keep first n/2 terms
$\geq (n/2) \log_2 (n/2)$	each of the n/2 terms left is $\geq \log_2 (n/2)$
$\geq (n/2)(\log_2 n - \log_2 2)$	property of logarithms
$\geq (1/2)n\log_2 n - (1/2)n$	arithmetic
"=" $\Omega(n \log n)$	

Breaking the lower bound on sorting

Assume 32/64-bit Integer:

2^{32} = 4294967296
13! = 6227020800

2^{64} = 18446744073709551616
21! = 51090942171709440000

Horrible algorithms: $\Omega(n^2)$

Bogo Sort (n!)
Stooge Sort ($n^{2.7}$)

Simple algorithms: $O(n^2)$

Insertion sort
Selection sort
Bubble Sort
Shell sort
…

Fancier algorithms: $O(n \log n)$

Heap sort
Merge sort
Quick sort (avg)
…

Comparison lower bound: $\Omega(n \log n)$

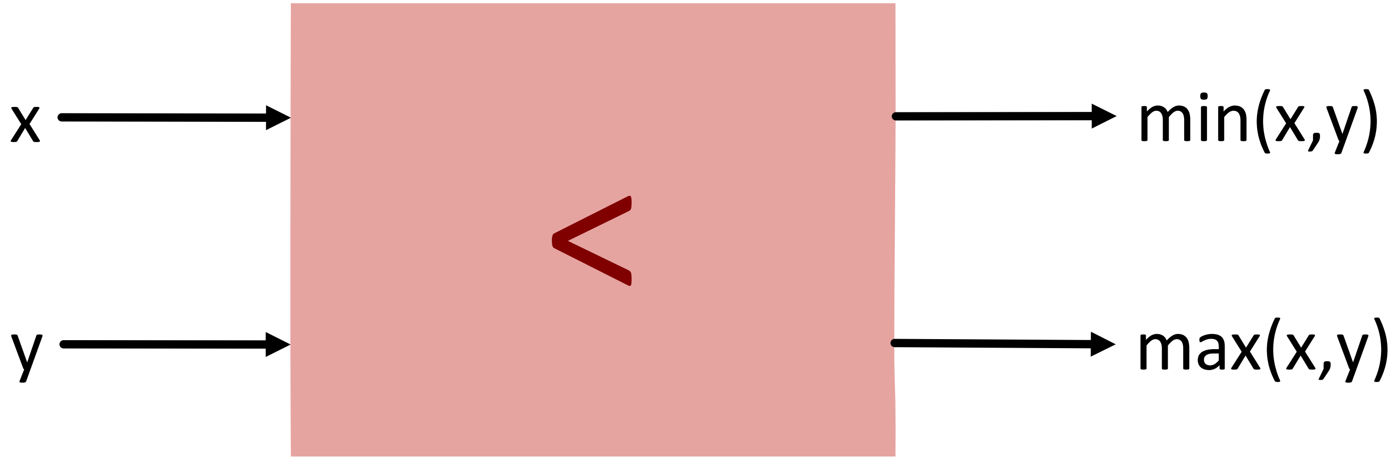
Nothing is ever straightforward in computer science…

Specialized algorithms: $O(n)$

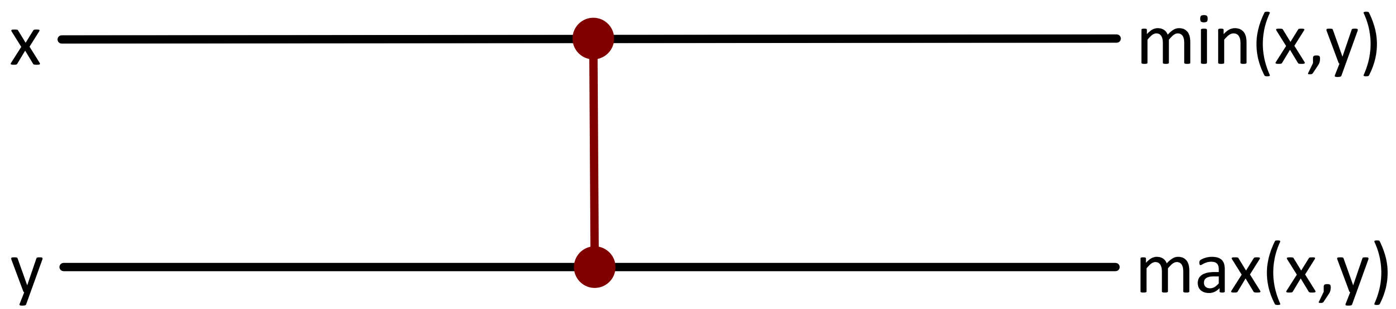
Radix sort

SORTING NETWORKS

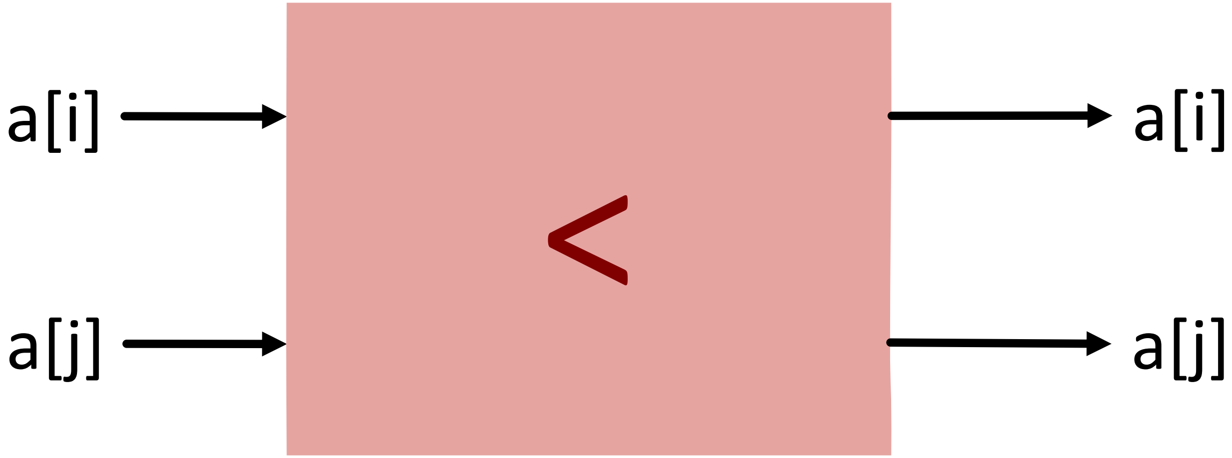
Comparator – the basic building block for sorting networks



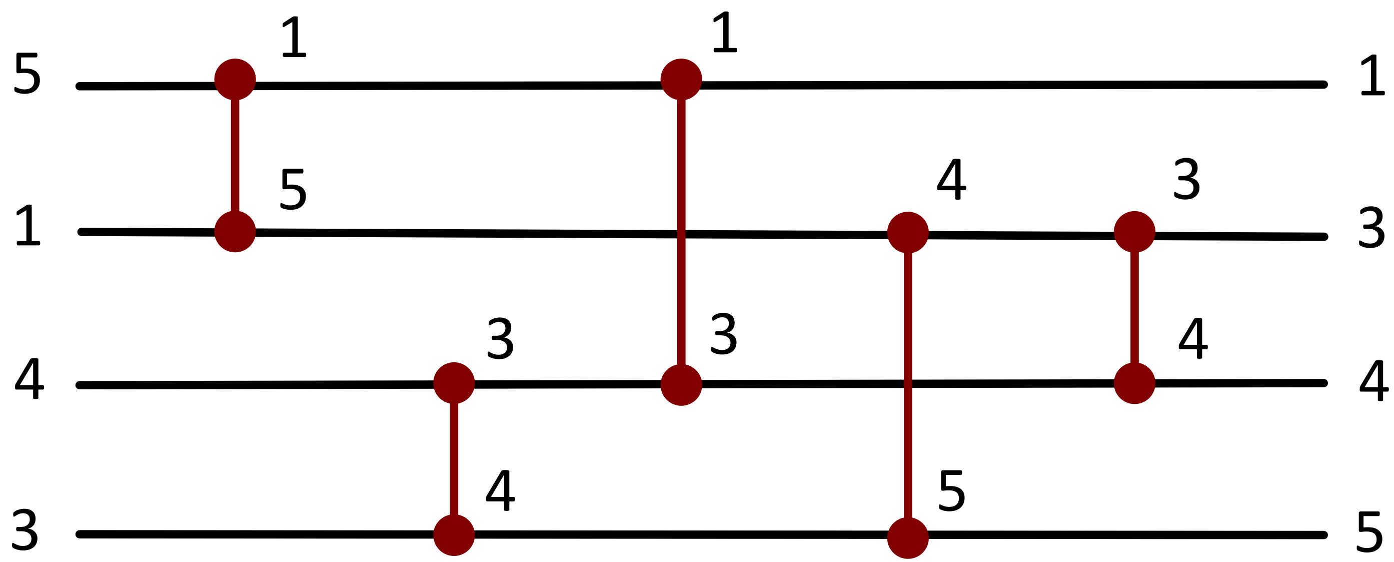
shorter notation:



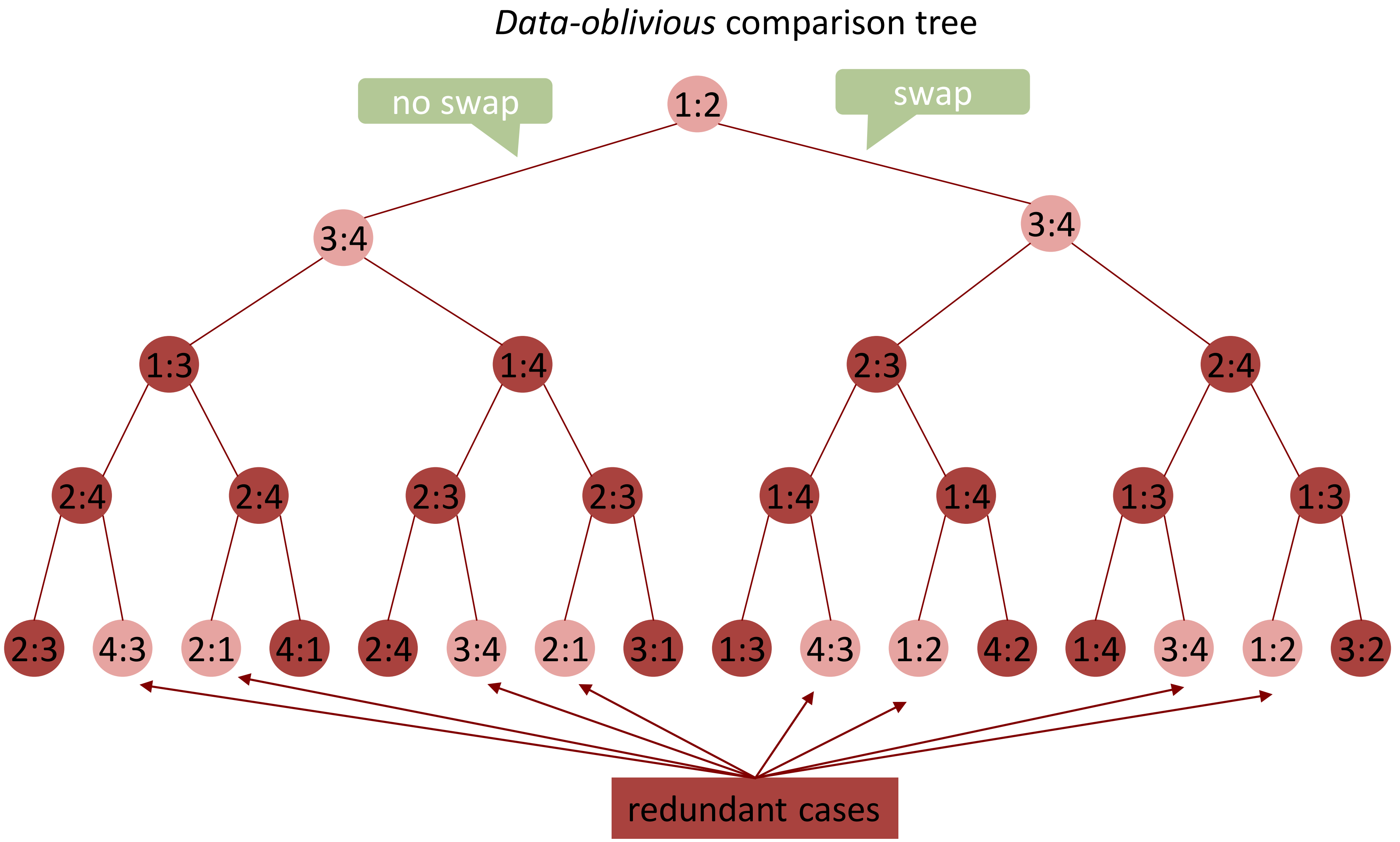
```
void compare(int[] a, int i, int j, boolean dir) {
   if (dir==(a[i]>a[j])){
      int t=a[i];
      a[i]=a[j];
      a[j]=t;
   }
}
```

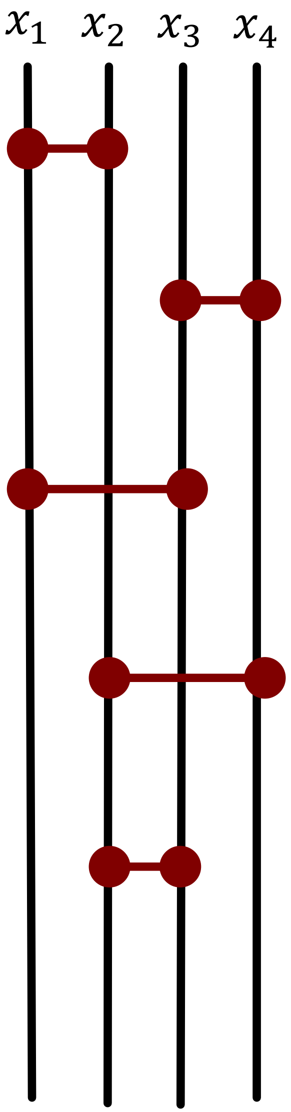


Sorting networks

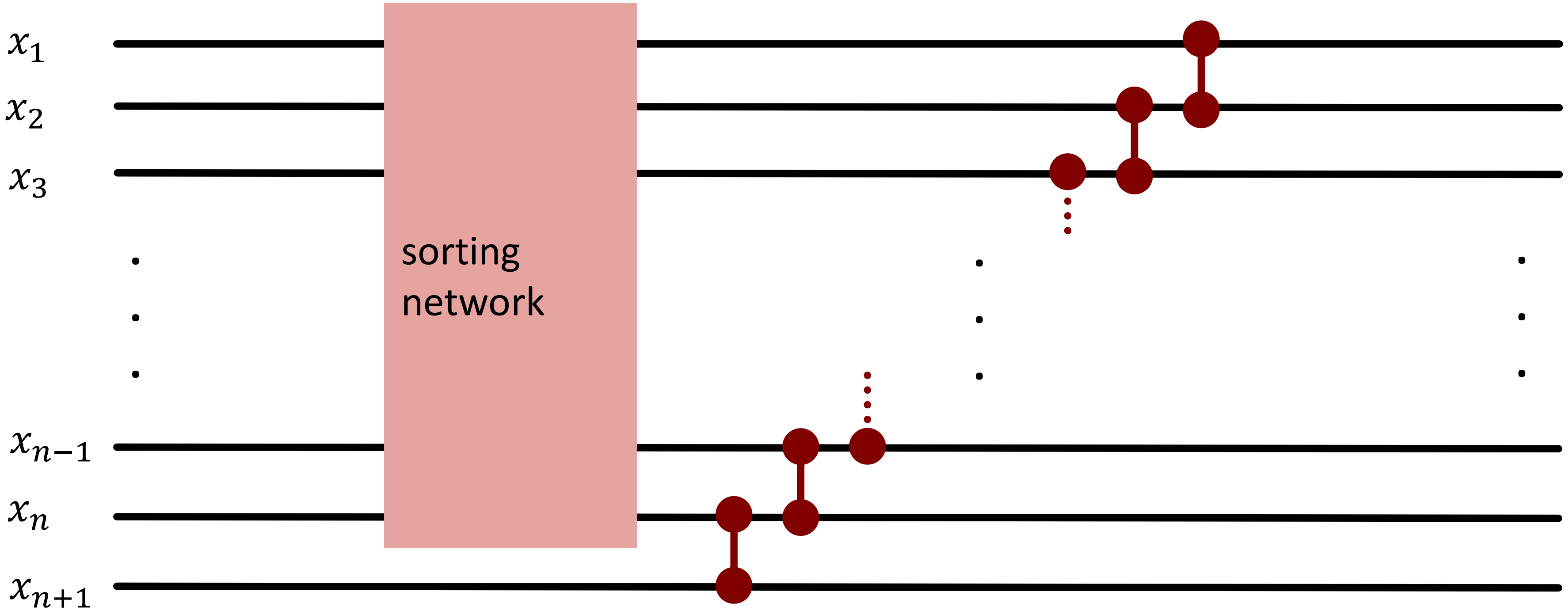


Sorting networks are *data-oblivious* (and redundant)

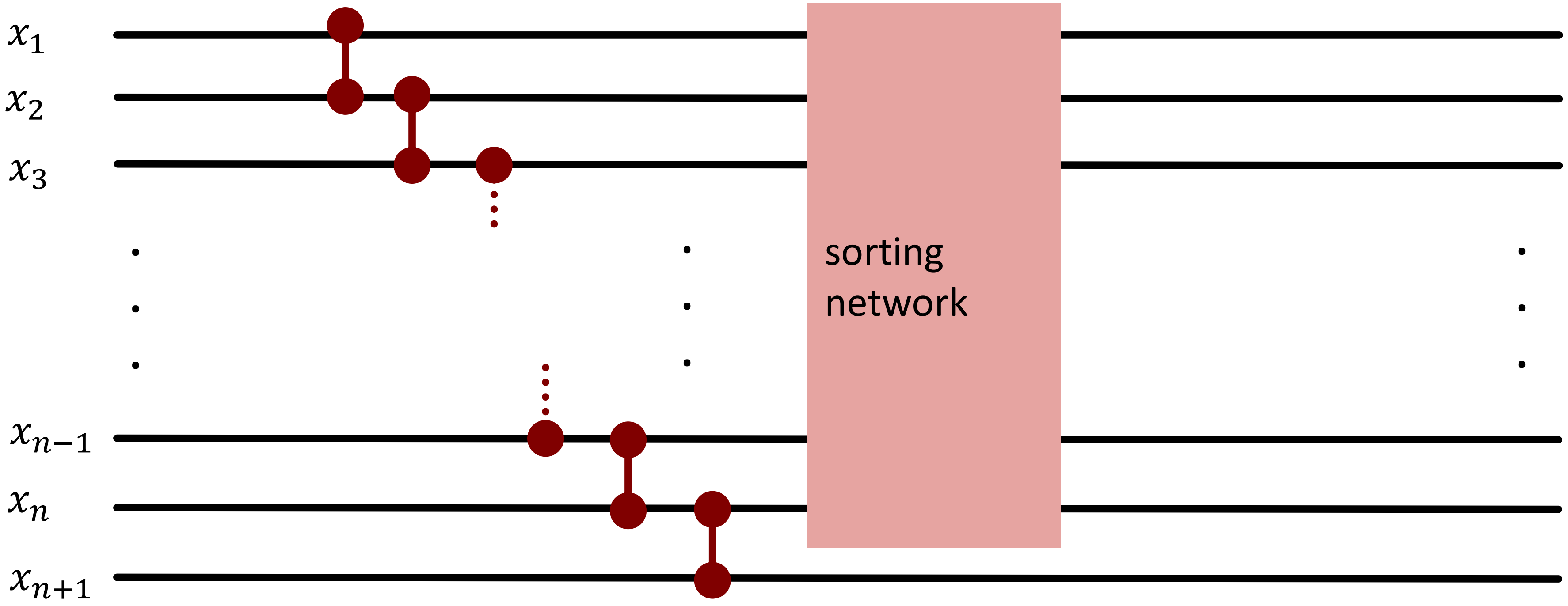




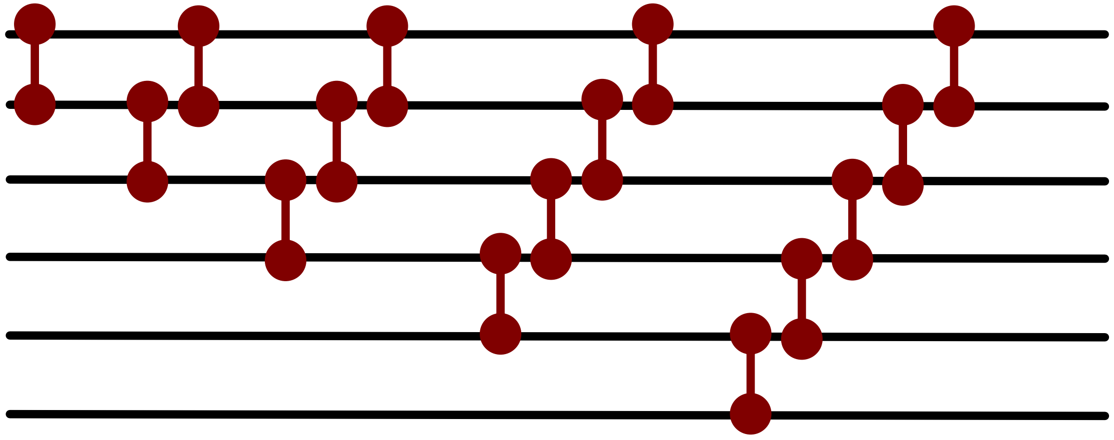
Recursive construction : Insertion



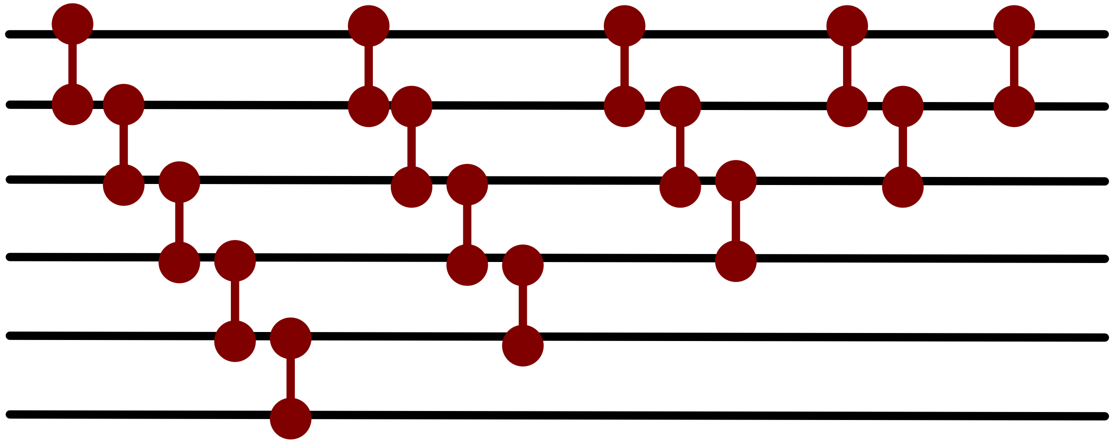
Recursive construction: Selection



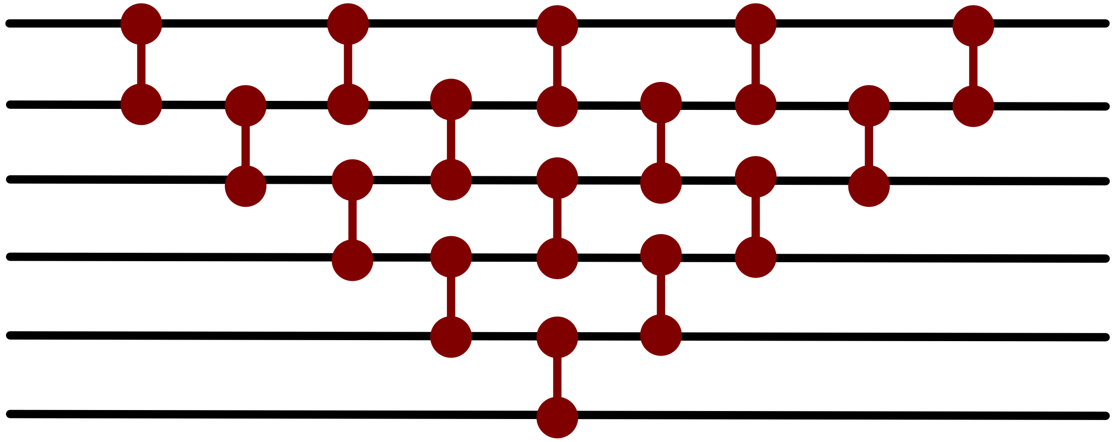
Applied recursively..



insertion sort



bubble sort



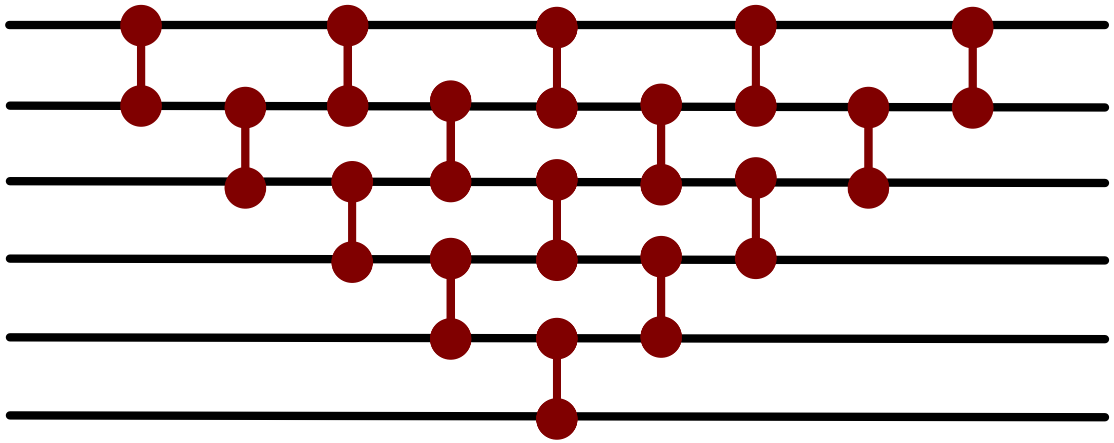
with parallelism: insertion sort = bubble sort !

Question

How many steps does a computer with infinite number of processors (comparators) require in order to sort using parallel bubble sort (depth)?

Answer: 2n – 3

Can this be improved ?



How many comparisons ?

Answer: (n-1) n/2

How many comparators are required (at a time)?

Answer: n/2

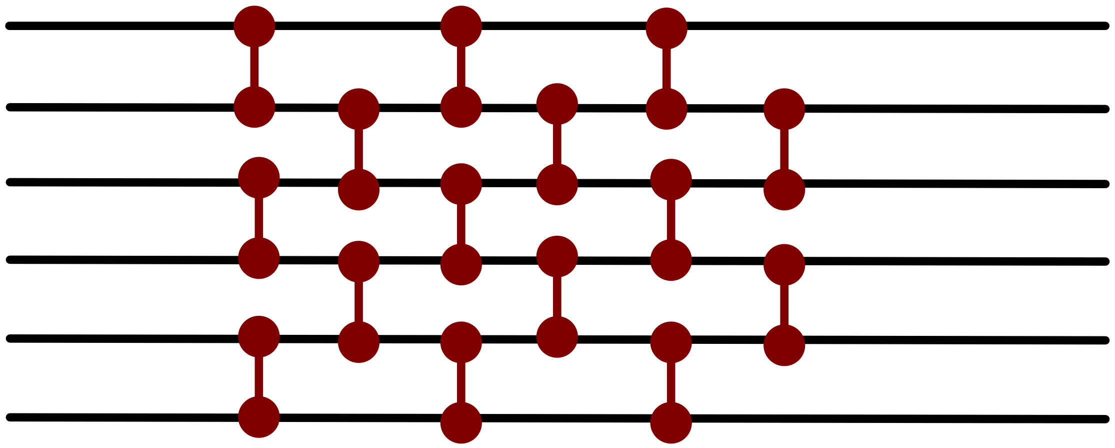
Reusable comparators: n-1

Improving parallel Bubble Sort

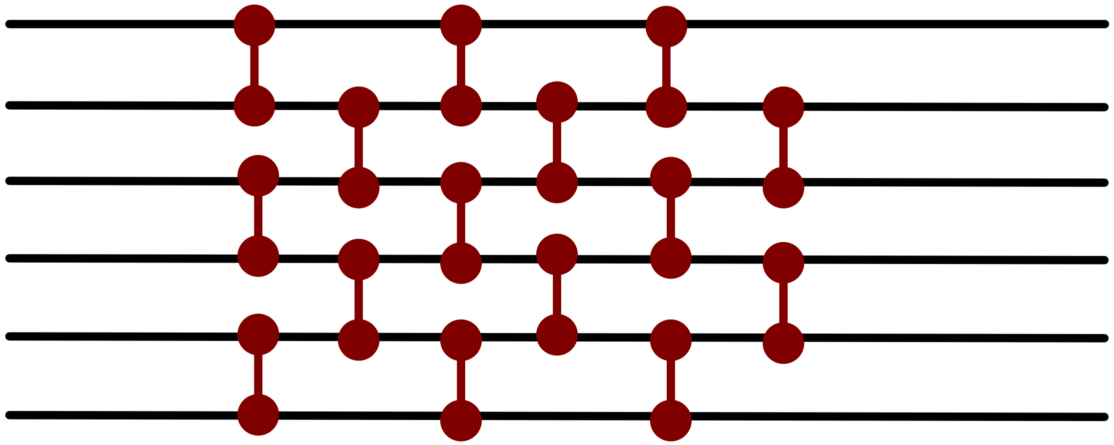
Odd-Even Transposition Sort:

0	9	↔	8		2	↔	7		3	↔	1		5	↔	6		4
1	8		9	↔	2		7	↔	1		3	↔	5		6	↔	4
2	8	↔	2		9	↔	1		7	↔	3		5	↔	4		6
3	2		8	↔	1		9	↔	3		7	↔	4		5	↔	6
4	2	↔	1		8	↔	3		9	↔	4		7	↔	5		6
5	1		2	↔	3		8	↔	4		9	↔	5		7	↔	6
6	1	↔	2		3	↔	4		8	↔	5		9	↔	6		7
7	1		2	↔	3		4	↔	5		8	↔	6		9	↔	7
8	1	↔	2		3	↔	4		5	↔	6		8	↔	7		9
	1		2		3		4		5		6		7		8		9

```
void oddEvenTranspositionSort(int[] a, boolean dir) {
   int n = a.length;
   for (int i = 0; i<n; ++i) {
      for (int j = i % 2; j+1<n; j+=2)
         compare(a,j,j+1,dir);
   }
}
```



Improvement?



Same number of comparators (at a time)

Same number of comparisons

But less parallel steps (depth): n

In a massively parallel setup, bubble sort is thus not too bad.

But it can go better...

How to get to a sorting network?

- **It's complicated ☺**
  - In fact, some structures are clear but there is a lot still to be discovered!
- **For example:**
  - What is the minimum number of comparators?
  - What is the minimum depth?
  - Tradeoffs between these two?

Source: wikipedia

Optimal sorting networks [edit]

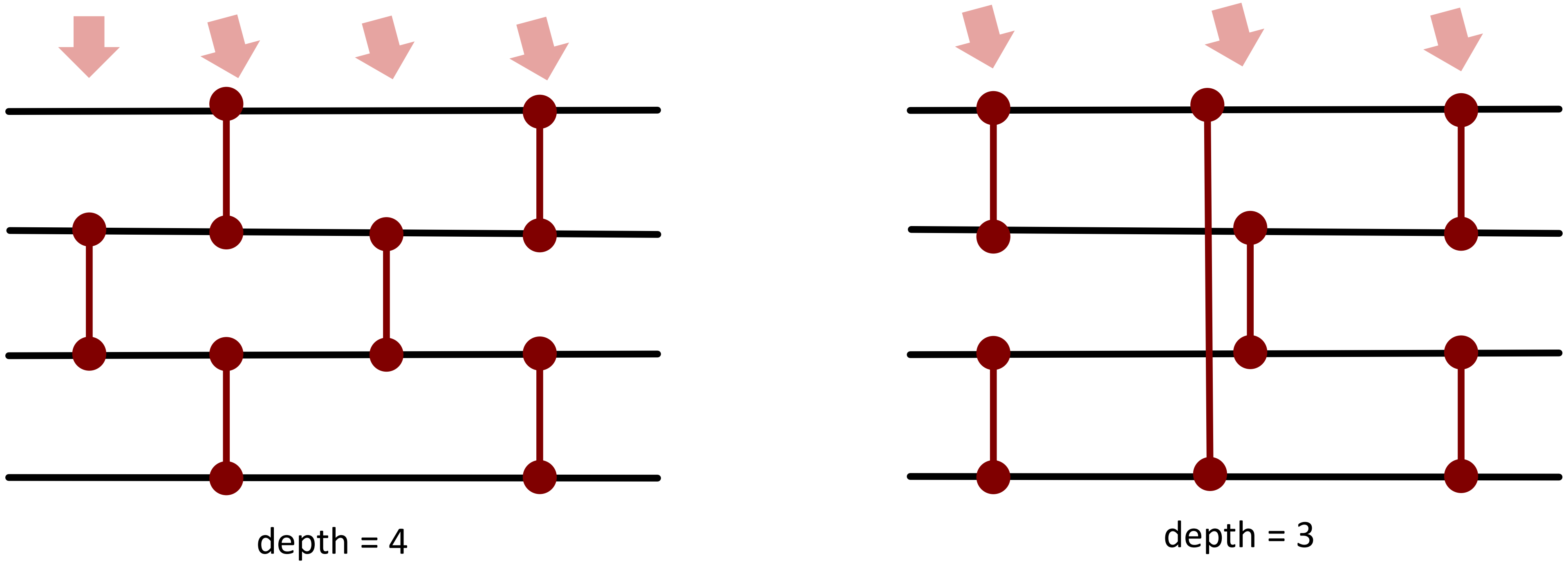
For small, fixed numbers of inputs n, *optimal* sorting networks can be constructed, with either minimal depth (for maximally parallel execution) or minimal size (number of comparators). These networks can be used to increase the performance of larger sorting networks resulting from the recursive constructions of, e.g., Batcher, by halting the recursion early and inserting optimal nets as base cases.[9] The following table summarizes the known optimality results:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Depth[10]	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9	10
Size, upper bound[11]	0	1	3	5	9	12	16	19	25	29	35	39	45	51	56	60	71
Size, lower bound (if different)[11]											33	37	41	45	49	53	58

The first sixteen depth-optimal networks are listed in Knuth's *Art of Computer Programming*,[1] and have been since the 1973 edition; however, while the optimality of the first eight was established by Floyd and Knuth in the 1960s, this property wasn't proven for the final six until 2014[12] (the cases nine and ten having been decided in 1991[9]).

For one to ten inputs, minimal (i.e. size-optimal) sorting networks are known, and for higher values, lower bounds on their sizes $S(n)$ can be derived inductively using a lemma due to Van Voorhis: $S(n+1) \geq S(n) + \lceil \log_2(n) \rceil$. All ten optimal networks have been known since 1969, with the first eight again being known as optimal since the work of Floyd and Knuth, but optimality of the cases $n = 9$ and $n = 10$ took until 2014 to be resolved.[11]

Parallel sorting



Prove that the two networks above sort four numbers. Easy?

Zero-one-principle

Theorem: If a network with n input lines sorts all 2^n sequences of 0s and 1s into non-decreasing order, it will sort any arbitrary sequence of n numbers in non-decreasing order.

Proof

Argue: If x is sorted by a network N then also any monotonic function of x.

1 | 8 | 20 | 30 | 5 | 9 → 1 | 5 | 8 | 9 | 20 | 30

e.g., floor(x/2) 0 | 4 | 10 | 15 | 2 | 4 → 0 | 2 | 4 | 4 | 10 | 15

Show: If x is **not** sorted by network N, **then** there is a monotonic function f that maps x to 0s and 1s and **f(x) is not sorted** by the network

1 | 8 | 20 | 30 | 5 | 9 → 1 | 5 | 9 | 8 | 20 | 30

0 | 0 | 1 | 1 | 0 | 1 → 0 | 0 | 1 | 0 | 1 | 1

$$x \text{ not sorted by } N \;\Rightarrow \text{ there is an } f(x) \in \{0,1\}^n \text{ not sorted by N}$$

$$\Leftrightarrow$$

$$f \text{ sorted by N for all } f \in \{0,1\}^n \Rightarrow x \text{ sorted by N for all x}$$

Proof

Assume a monotonic function $f(x)$ with $f(x) \leq f(y)$ whenever $x \leq y$ and a network N that sorts. Let N transform $(x_1, x_2, \ldots, x_n)$ into $(y_1, y_2, \ldots, y_n)$, then it also transforms $(f(x_1), f(x_2), \ldots, f(x_n))$ into $(f(y_1), f(y_2), \ldots, f(y_n))$.

All comparators must act in the same way for the $f(x_i)$ as they do for the x_i

Assume $y_i > y_{i+1}$ for some i, then consider the monotonic function

$$f(x) = \begin{cases} 0, if\ x < y_i \\ 1, if\ x \geq y_i \end{cases}$$

→N converts

$(f(x_1), f(x_2), \ldots, f(x_n))$ **into** $(f(y_1), f(y_2), \ldots f(y_i), f(y_{i+1}), \ldots, f(y_n))$

1

0

Bitonic sort

Bitonic (Merge) Sort is a parallel algorithm for sorting

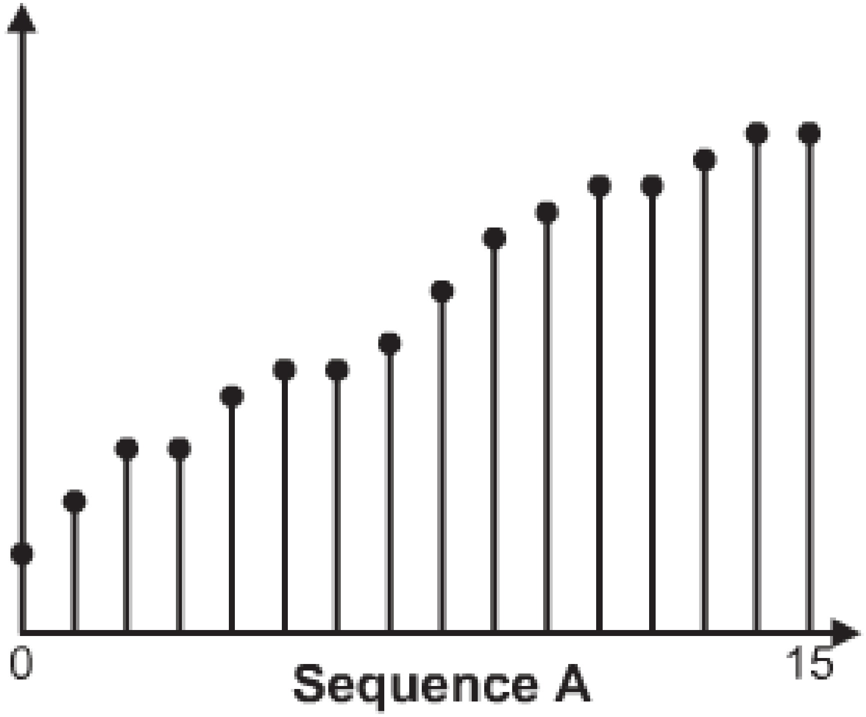
If enough processors are available, bitonic sort breaks the lower bound on sorting for comparison sort algorithm

Time complexity of $O(n \ \log^2 n)$ (sequential execution)

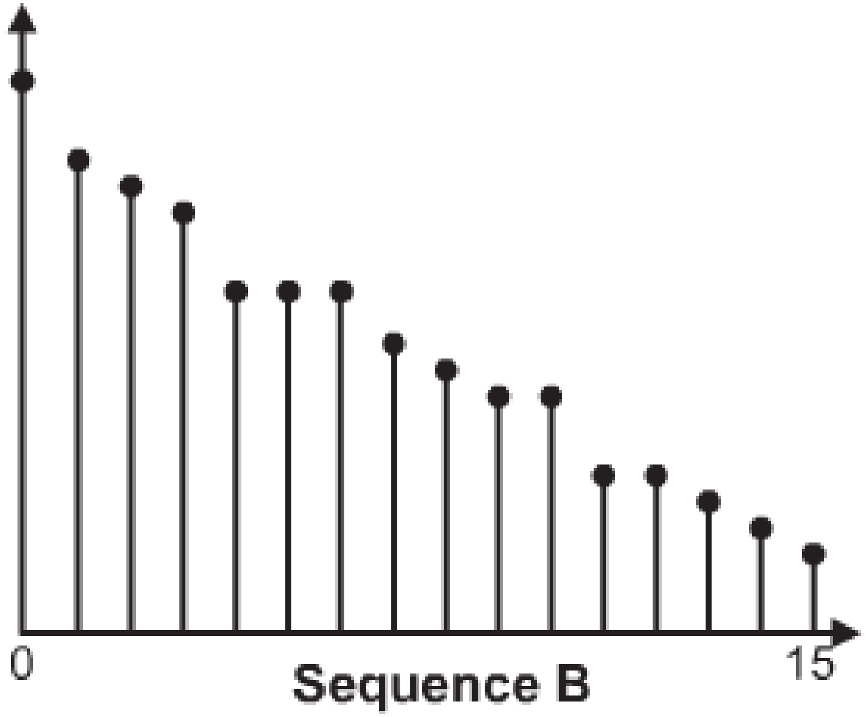
Time complexity of $O(\log^2 n)$ (parallel time)

Worst = Average = Best case

What is a Bitonic sequence?



Monotonic ascending sequence



Monotonic descending sequence

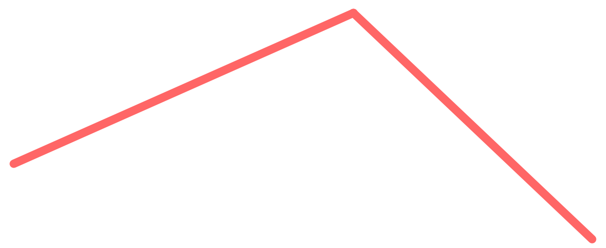
Bitonic Sequences Allow Wraparound

A *bitonic sequence* is defined as a list with no more than one **Local maximum** and no more than one **Local minimum**.

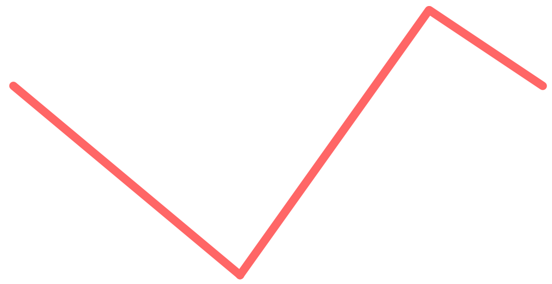
Bitonic (again)

Sequence $(x_1, x_2, \ldots, x_n)$ is bitonic, if it can be circularly shifted such that it is first monotonically increasing and then monontonically decreasing.

(1, 2, 3, 4, 5, 3, 1, 0)



(4, 3, 2, 1, 2, 4, 6, 5)



Bitonic 0-1 Sequences

$$0^i 1^j 0^k$$

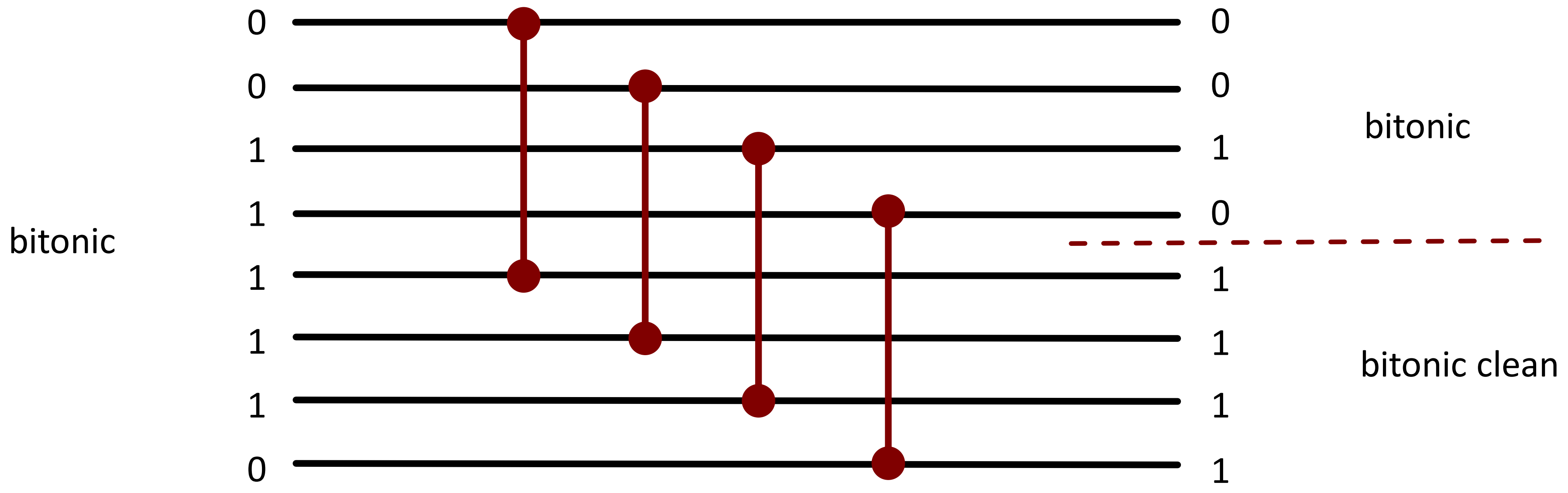
$$1^i 0^j 1^k$$

Properties

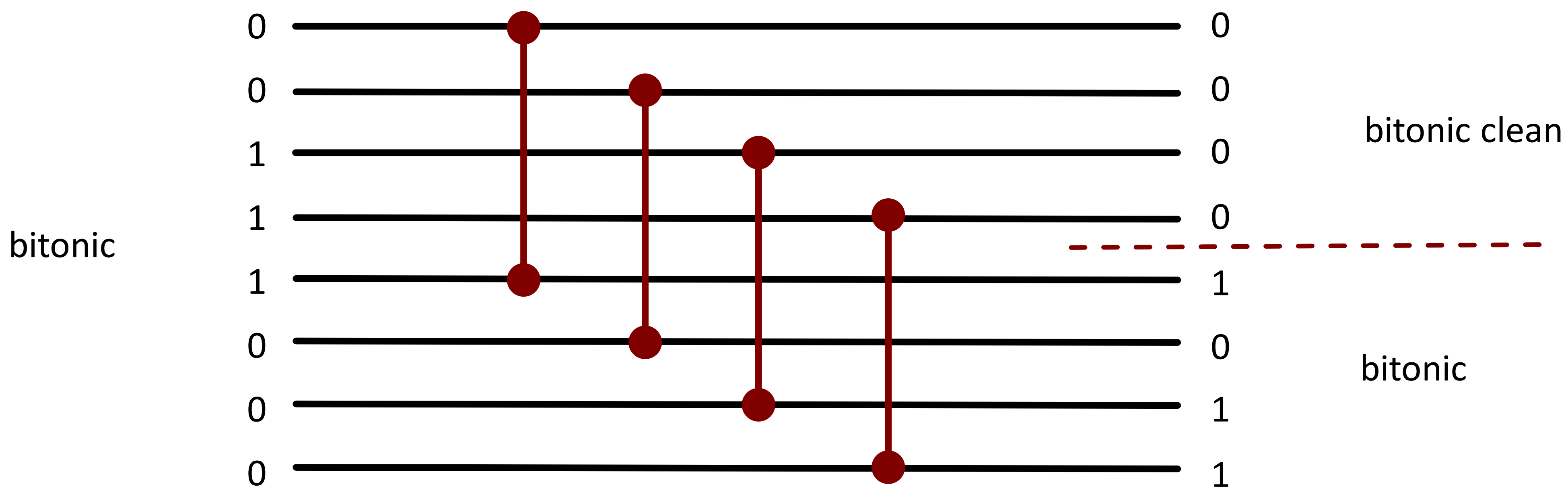
If $(x_1, x_2, \dots, x_n)$ is monotonically increasing (decreasing) and then monotonically decreasing (increasing), then it is bitonic

If $(x_1, x_2, \dots, x_n)$ is bitonic, then $(x_1, x_2, \dots, x_n)^R := (x_n, x_{n-1}, \dots, x_1)$ is also bitonic

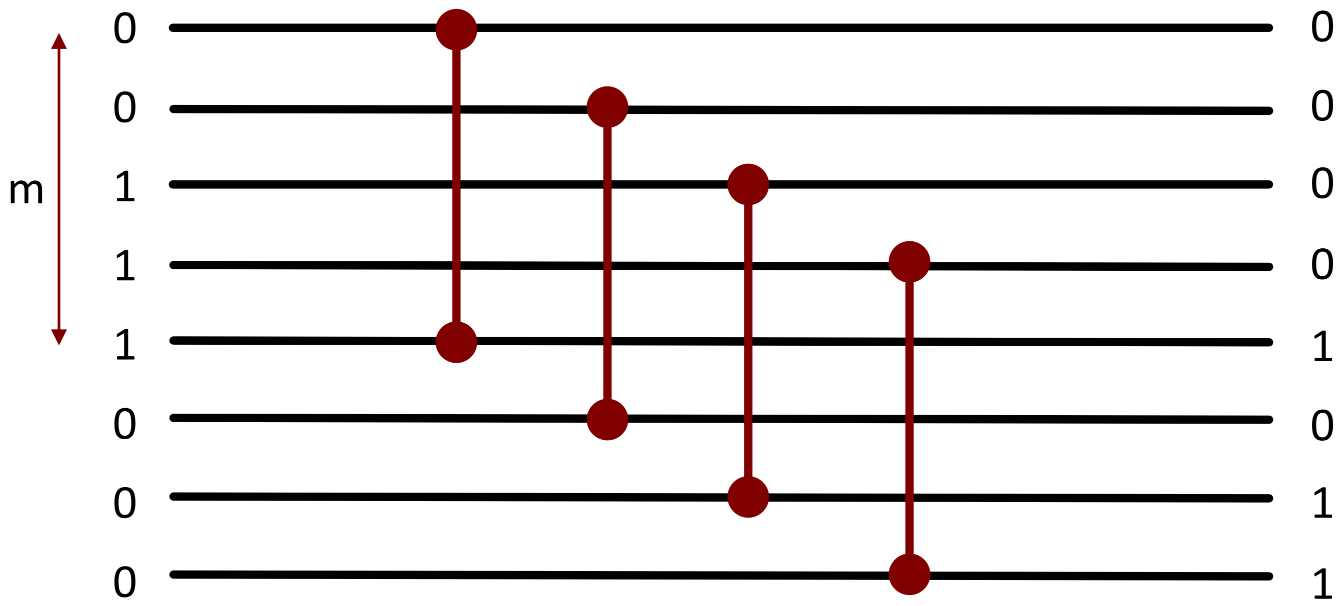
The Half-Cleaner



The Half-Cleaner

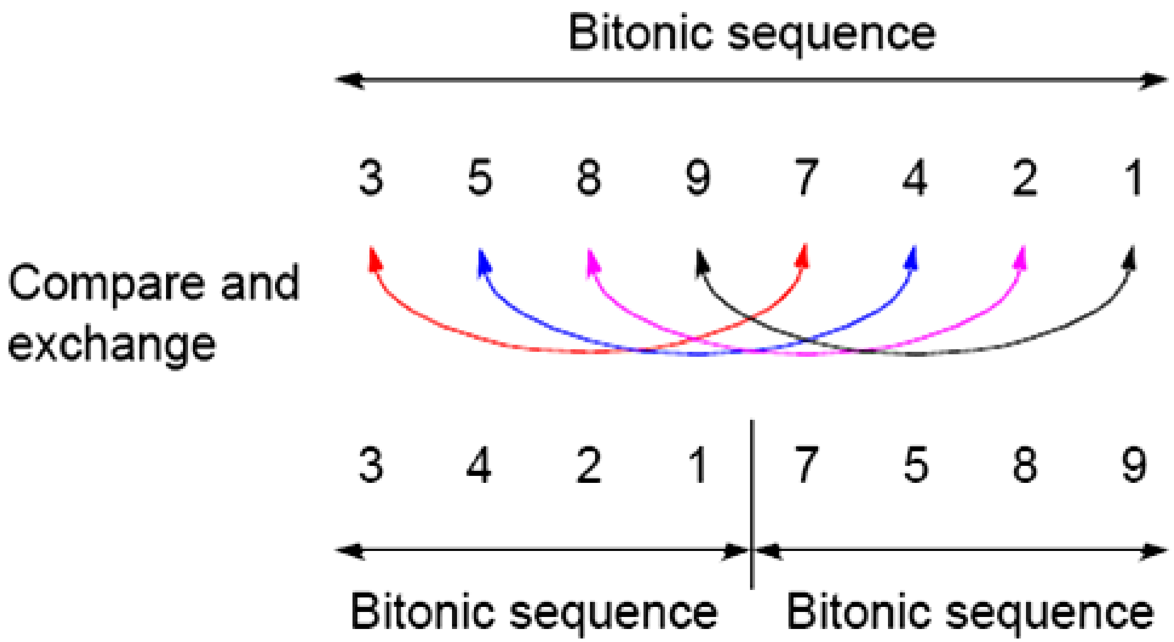


```
void halfClean(int[] a, int lo, int m, boolean dir)
{
   for (int i=lo; i<lo+m; i++)
      compare(a, i, i+m, dir);
}
```



Binary Split: Application of the Half-Cleaner

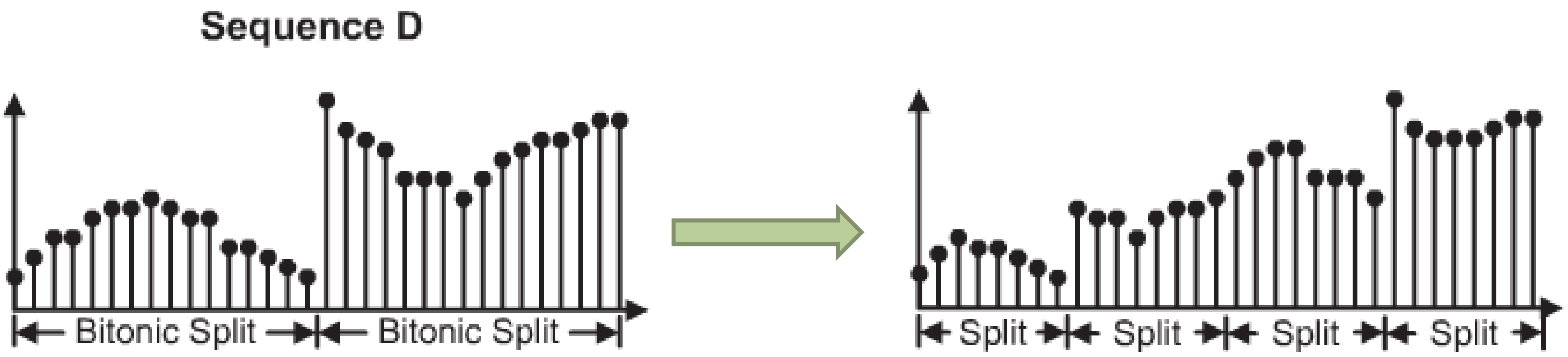
1. **Divide the bitonic list into two equal halves.**
2. **Compare-Exchange each item on the first half with the corresponding item in the second half.**



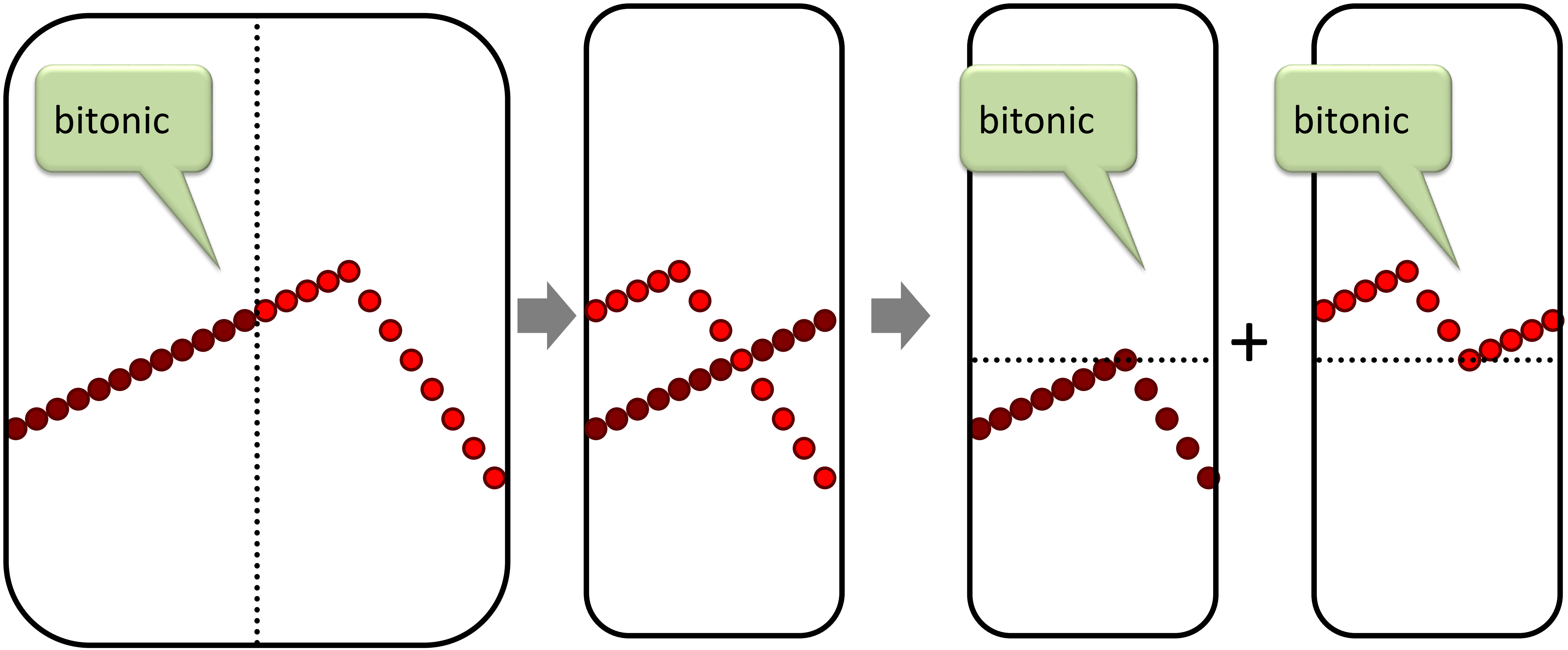
Binary Splits - Result

Two *bitonic* sequences where the numbers in one sequence are all less than the numbers in the other sequence.

Because the original sequence was *bitonic*, every element in the lower half of new sequence is less than or equal to the elements in its upper half.



Bitonic Split Example

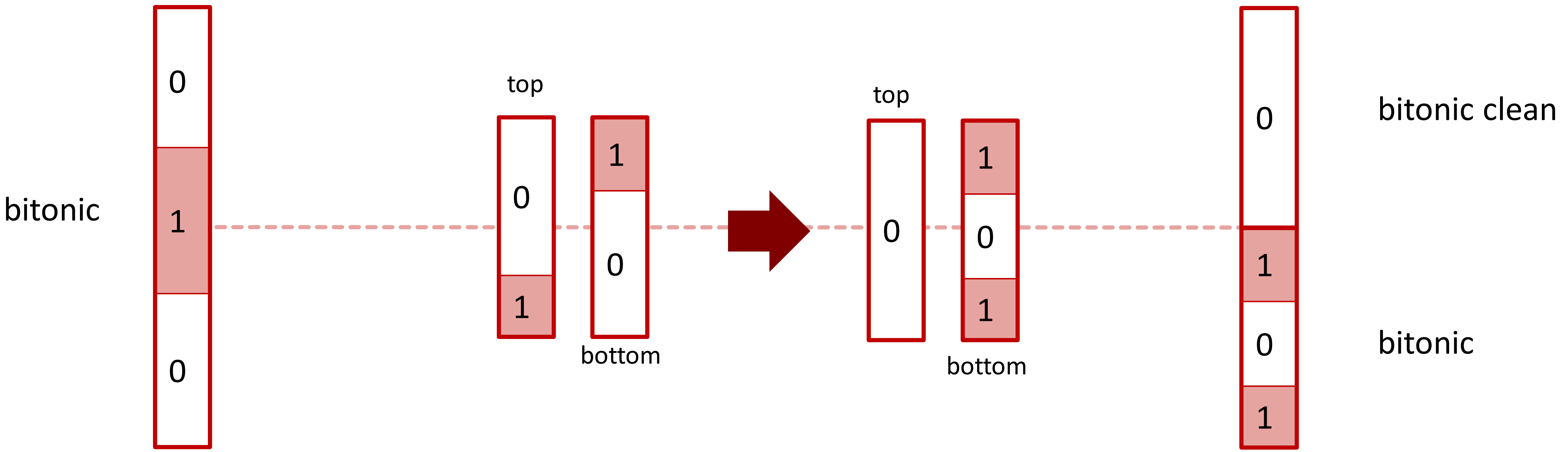


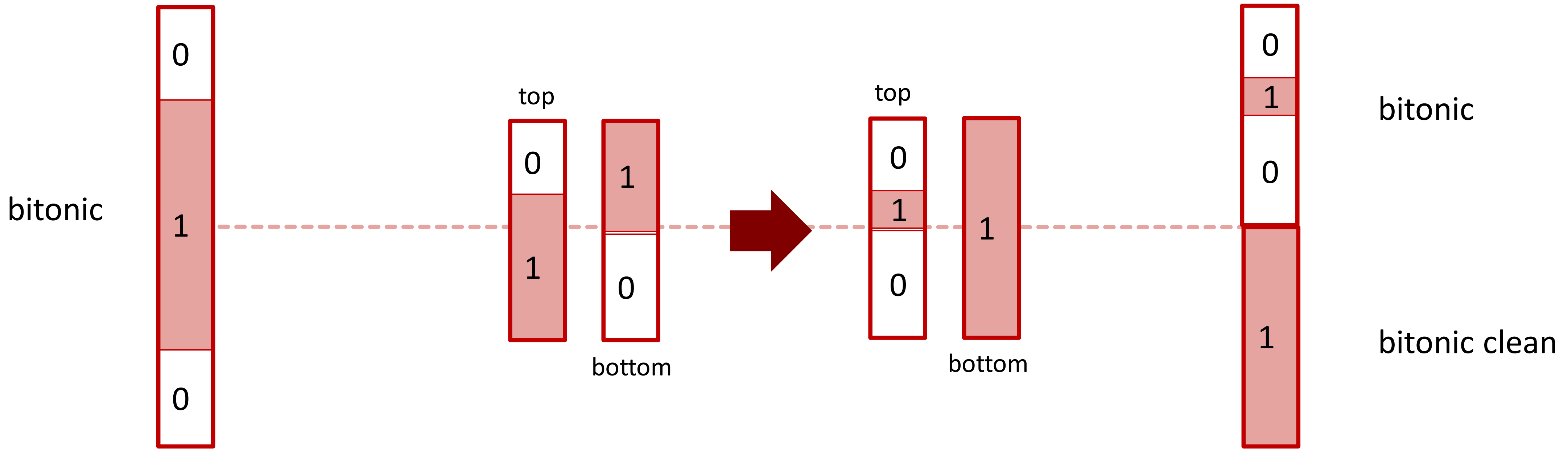
Lemma

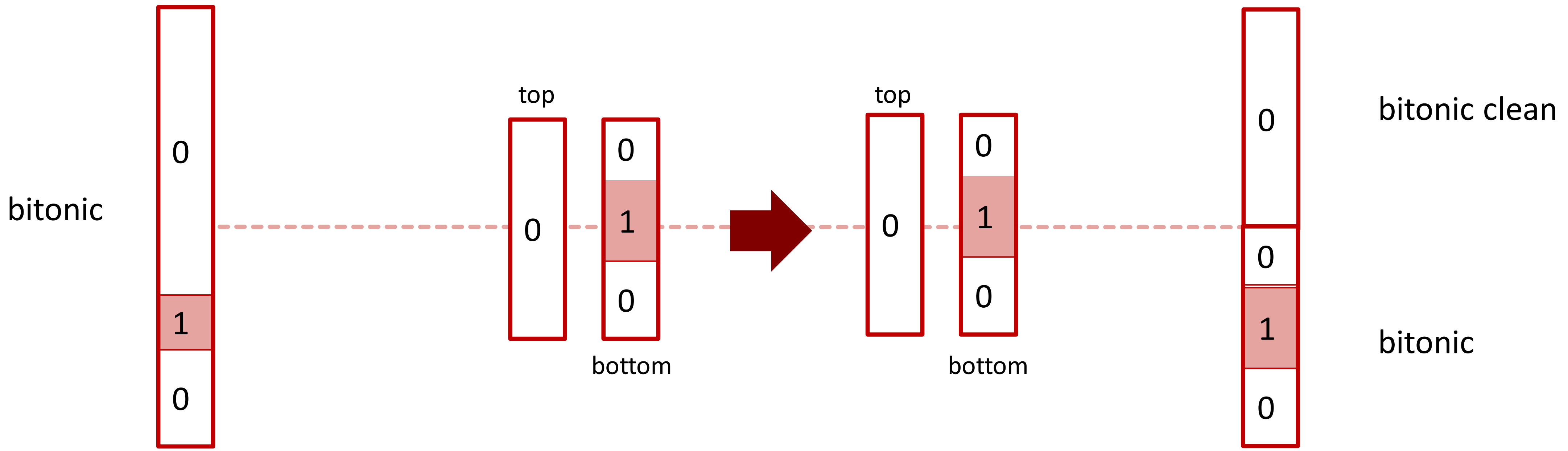
Input bitonic sequence of 0s and 1s, then for the output of the half-cleaner it holds that

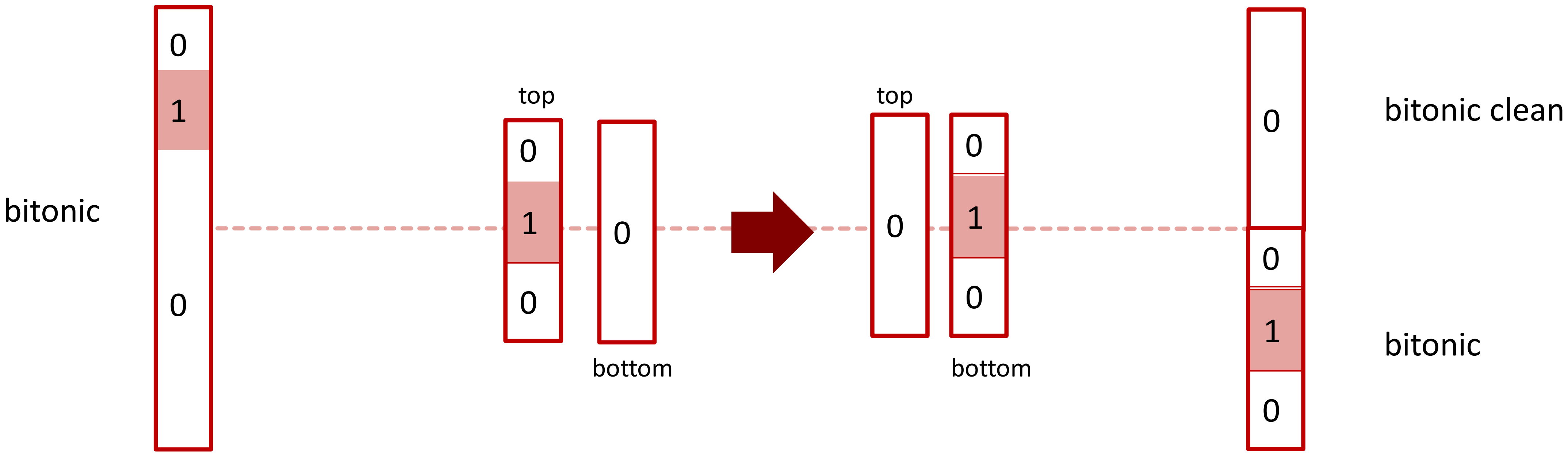
- Upper and lower half is bitonic
- One of the two halfs is bitonic clean
- Every number in upper half $\leq$ every number in the lower half

Proof: All cases

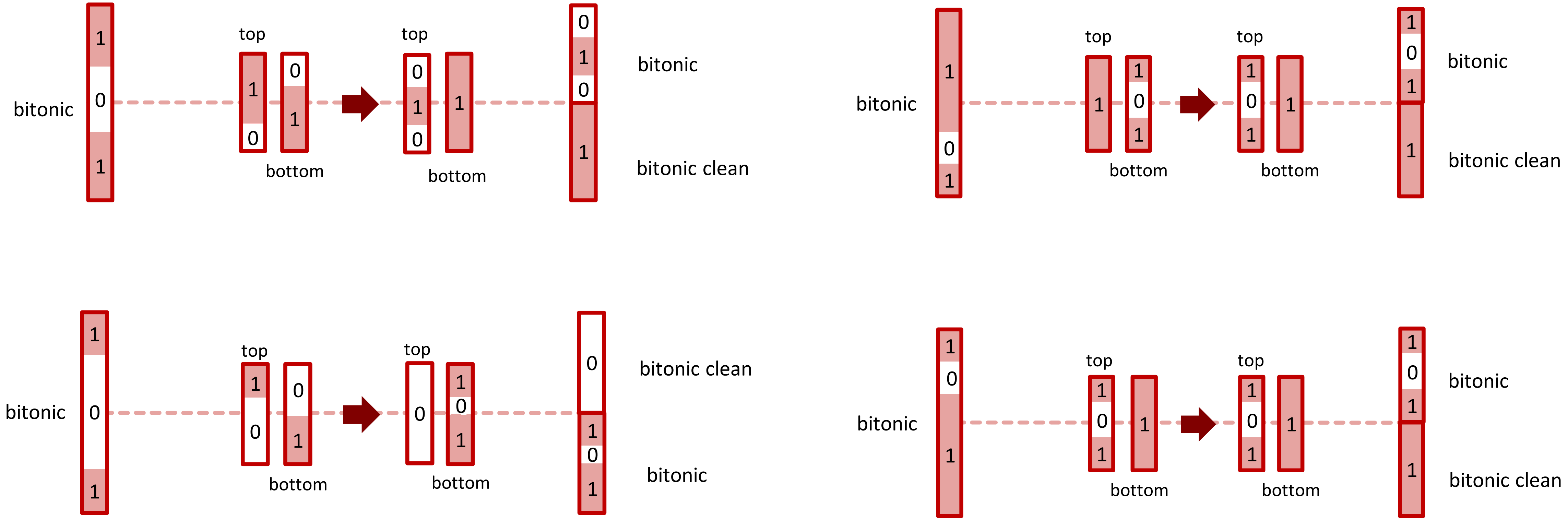



bitonic
0
1
0
top
0
1
1
0
bottom
top
0
1
0
1
bottom
0
1
0
1
bitonic
bitonic clean


bitonic
0
1
0
top
0
0
1
0
bottom
top
0
0
1
0
bottom
0
0
1
0
bitonic clean
bitonic

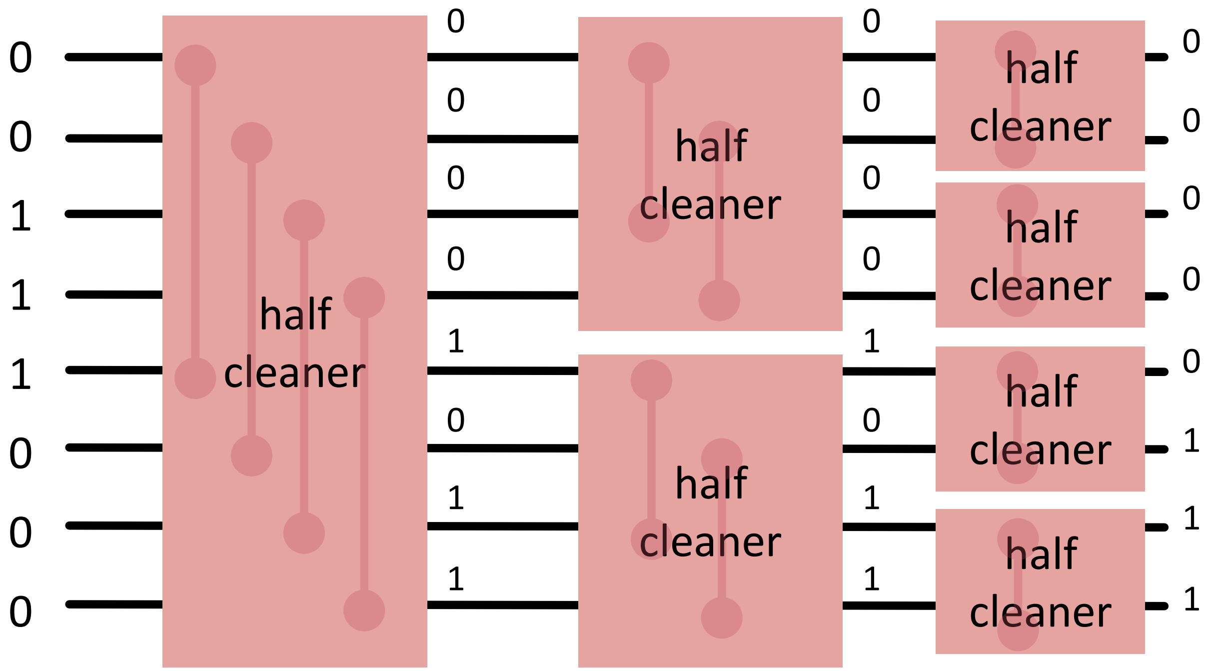

bitonic
0
1
0
top
0
1
0
0
bottom
top
0
0
1
0
bottom
0
bitonic clean
0
1
0
bitonic

The four remaining cases (010 → 101)



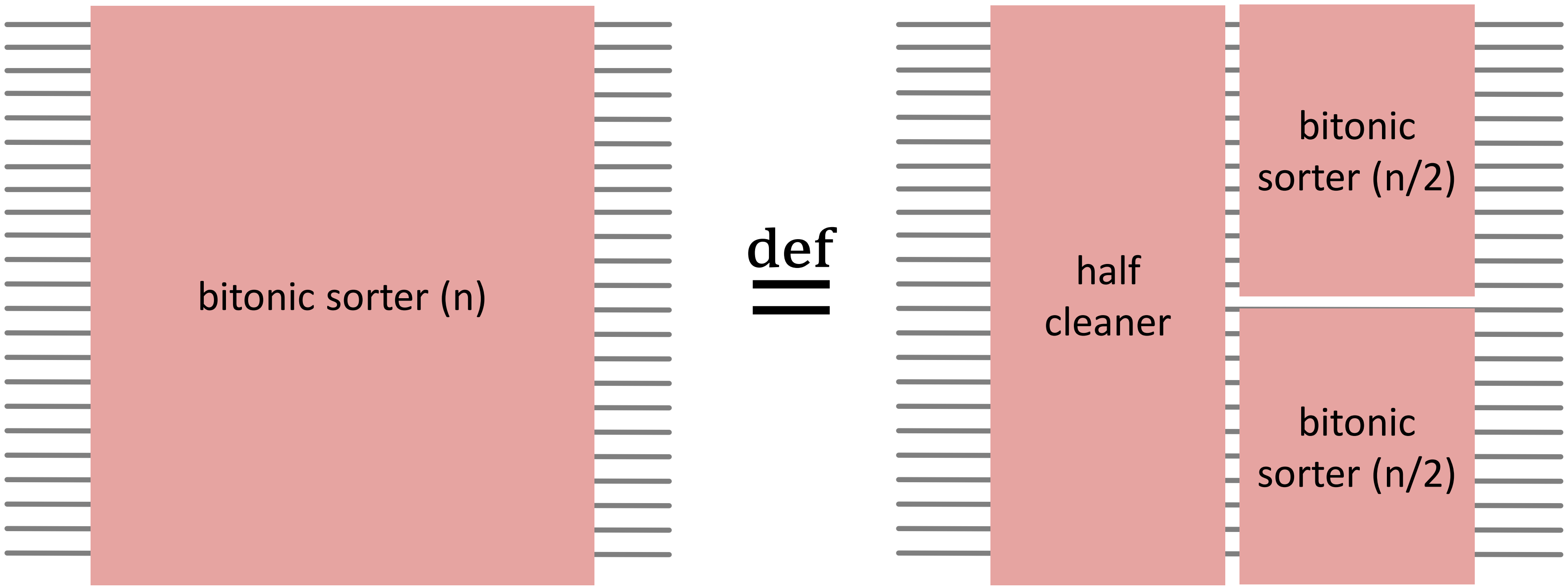
Construction of a Bitonic Sorting Network

bitonic

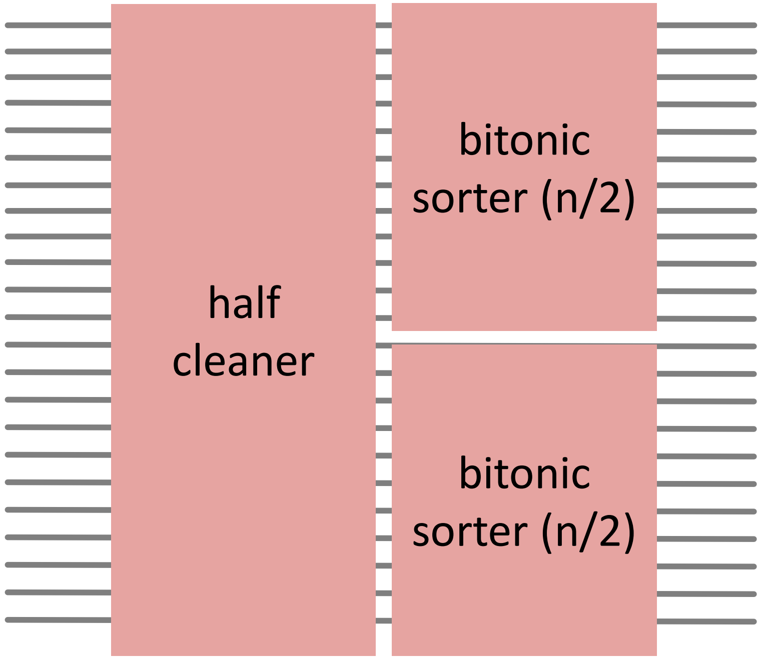


sorted

Recursive Construction

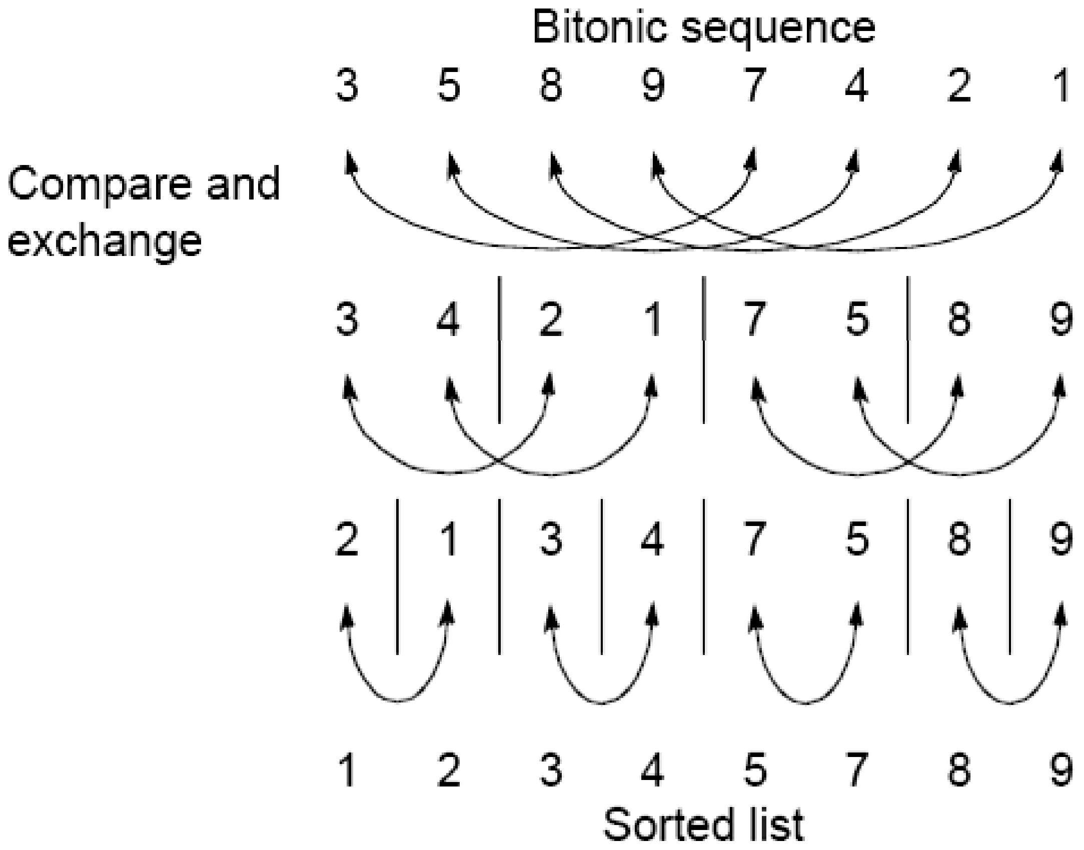


```
void bitonicMerge(int[] a, int lo, int n, boolean dir)
{
   if (n>1) {
      int m=n/2;
      halfClean(a, lo, m, dir);
      bitonicMerge(a, lo, m, dir);
      bitonicMerge(a, lo+m, m, dir);
   }
}
```

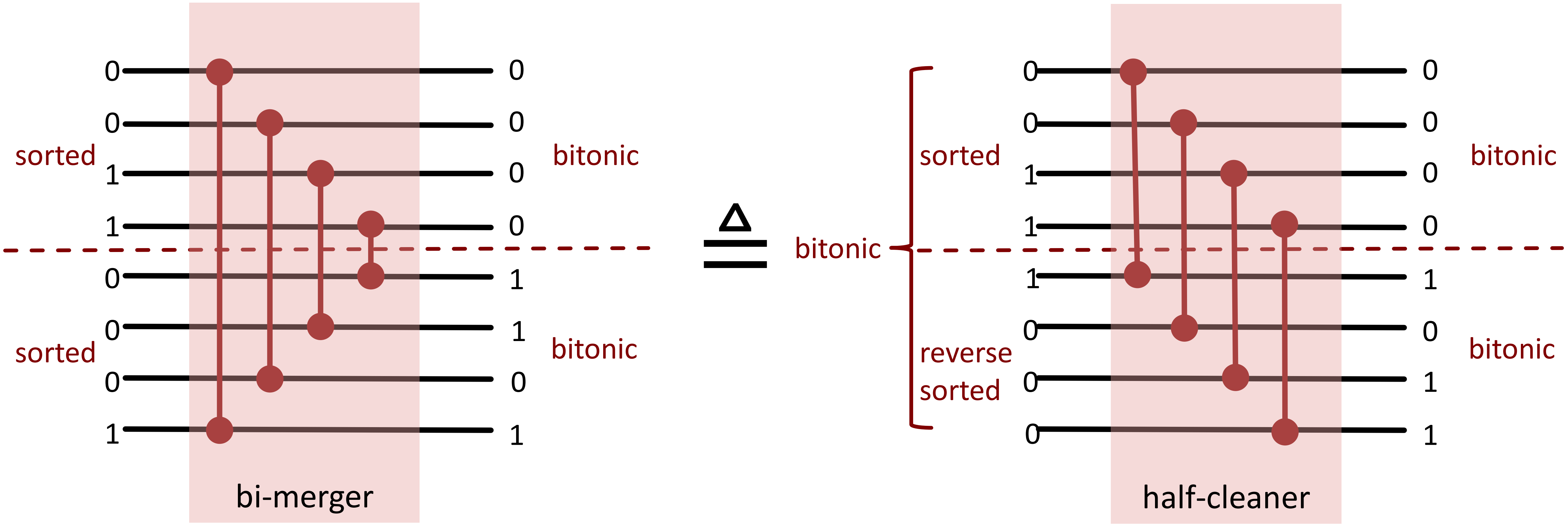


Bitonic Merge

- Compare-and-exchange moves smaller numbers of each pair to left and larger numbers of pair to right.
- Given a *bitonic* sequence, recursively performing '*binary split*' will sort the list.

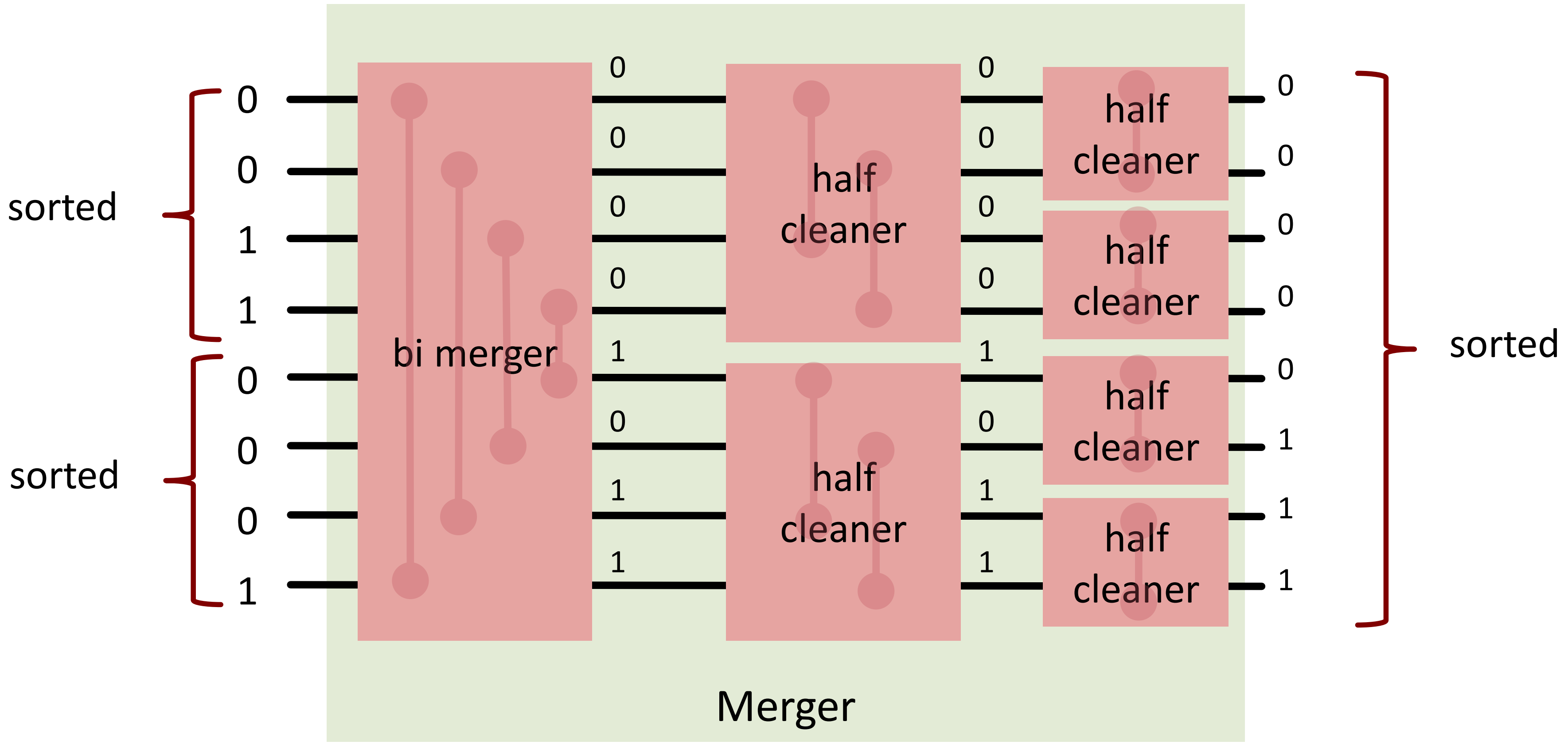


Bi-Merger

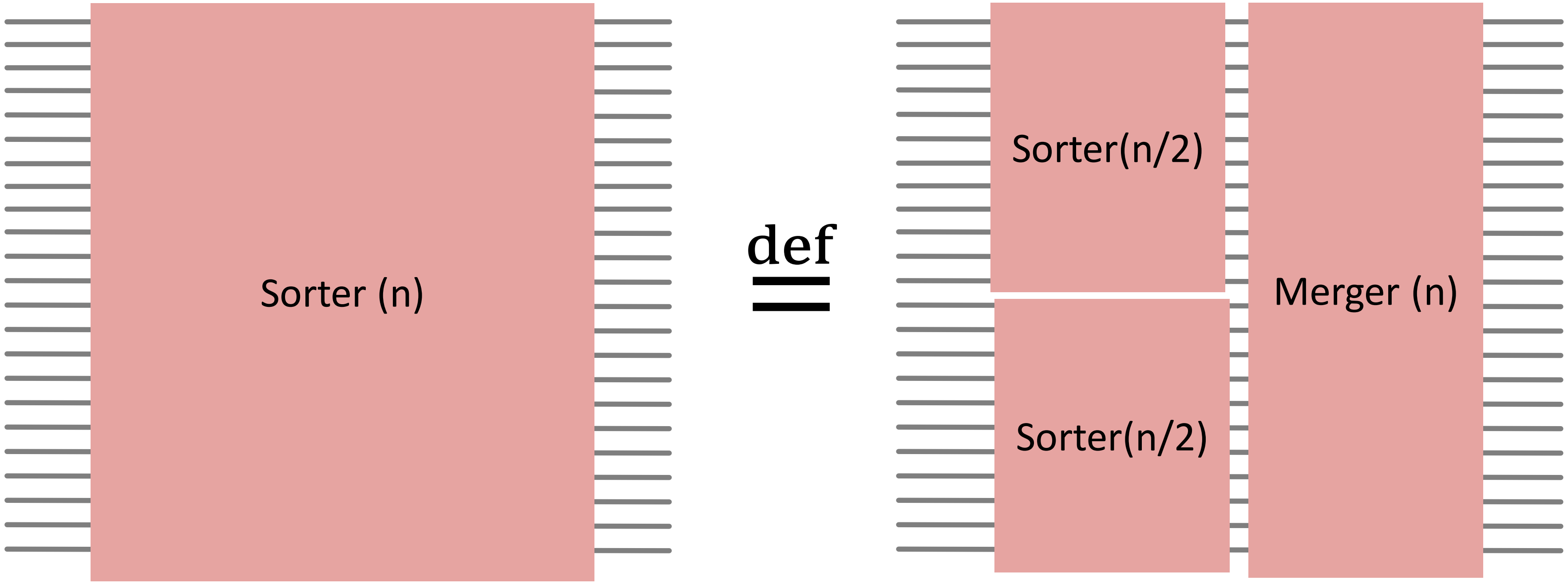


Bi-Merger on two sorted sequences acts like a half-cleaner on a bitonic sequence (when one of the sequences is reversed)

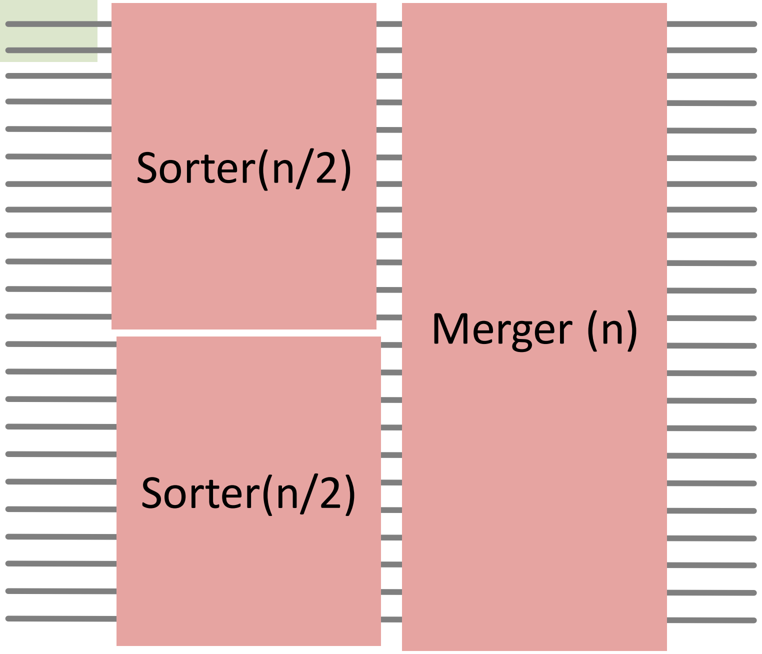
Merger


sorted
sorted
0
0
1
1
0
0
0
1
bi merger
0
0
0
0
1
0
1
1
half cleaner
half cleaner
0
0
0
0
1
0
1
1
half cleaner
half cleaner
half cleaner
half cleaner
0
0
0
0
0
1
1
1
sorted
Merger

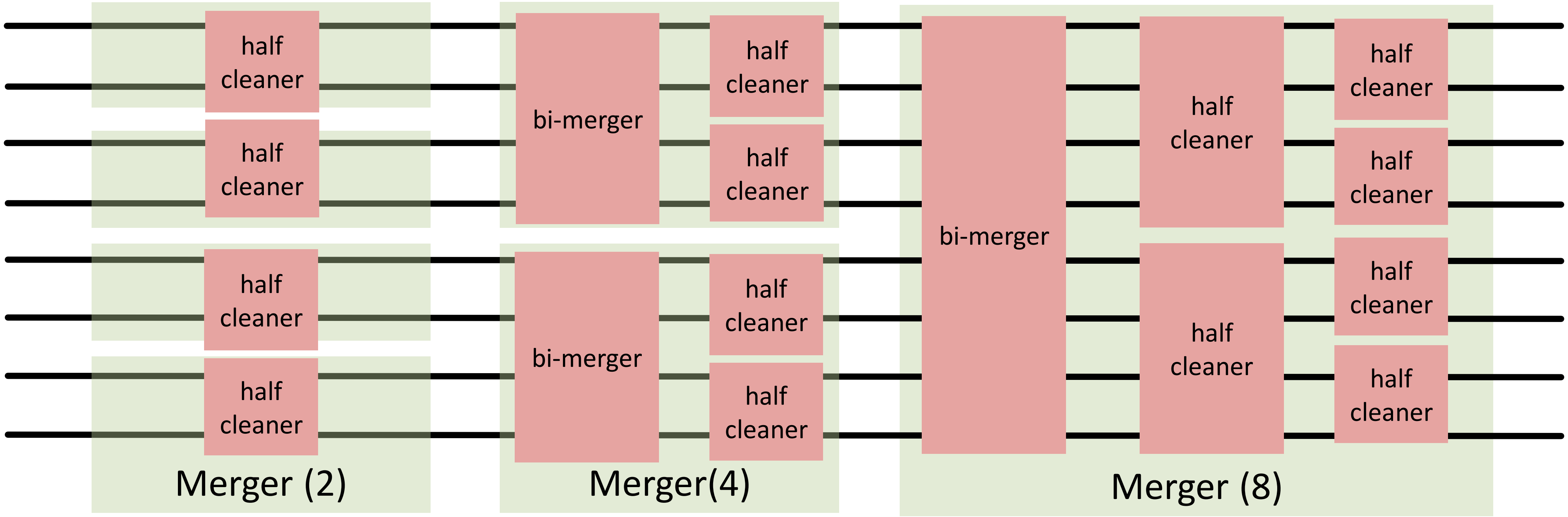
Recursive Construction of a Sorter



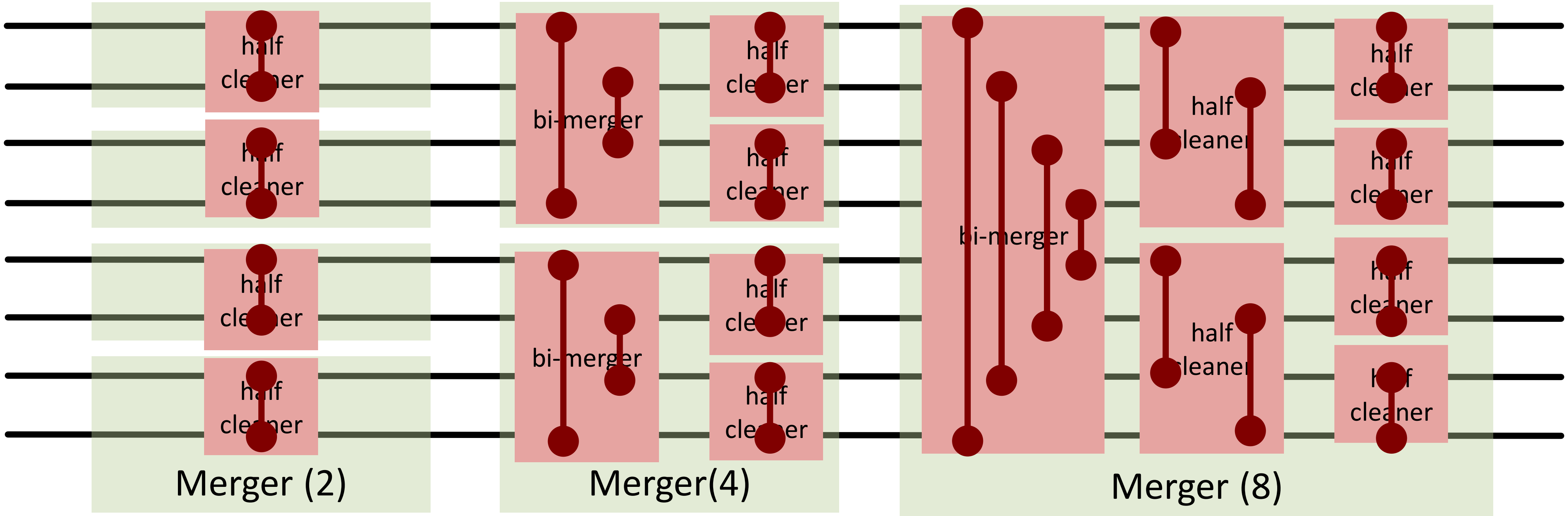
```
private void bitonicSort(int a[], int lo, int n, boolean dir) {
   if (n>1){
      int m=n/2;
      bitonicSort(a, lo, m, ASCENDING);
      bitonicSort(a, lo+m, n, DESCENDING);
      bitonicMerge(a, lo, n, dir);
   }
}
```



Example


half cleaner
half cleaner
half cleaner
half cleaner
Merger (2)
bi-merger
half cleaner
half cleaner
bi-merger
half cleaner
half cleaner
Merger(4)
bi-merger
half cleaner
half cleaner
half cleaner
half cleaner
half cleaner
half cleaner
Merger (8)

Example


half cleaner
half cleaner
half cleaner
half cleaner
Merger (2)
bi-merger
half cleaner
half cleaner
bi-merger
half cleaner
half cleaner
Merger(4)
bi-merger
half cleaner
half cleaner
half cleaner
half cleaner
half cleaner
half cleaner
Merger (8)

Bitonic Merge Sort

How many steps?

#mergers

$$\sum_{i=1}^{\log n} \log 2^i = \sum_{i=1}^{\log n} i \log 2 = \frac{\log n \cdot (\log n + 1)}{2} = O(\log^2 n)$$

#steps / merger

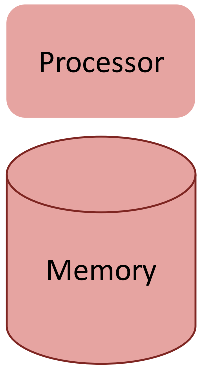
Interlude: Machine Models

RAM : Random Access Machine

- Unbounded local memory
- Each memory has unbounded capacity
- Simple operations: data, comparison, branches
- All operations take unit time

Time complexity: number of steps executed

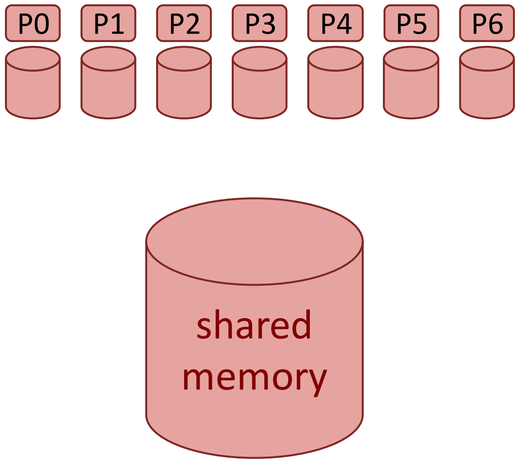
Space complexity: (maximum) number of memory cells used



Machine Models

PRAM : *Parallel* Random Access Machine

- Abstract machine for designing algorithms applicable for parallel computers
- Unbounded collection of RAM processors $P_0, P_1, \ldots$
- Each processor has unbounded registers
- Unbounded shared memory
- All processors can access all memory in unit time
- All communication via shared memory



Shared Memory Access Model

ER: processors can simultaneously read from distinct memory locations

EW: processors can simultaneously write to distinct memory locations

CR: processors can simultanously read from any memory location

CW: processors can simultaneously write to any memory location

Specification of the machine model as one of EREW, CREW, CRCW

Example: Why the machine model can be important

Find maximum of n elements in an array A

Assume $O(n^2)$ processors and the **CRCW model**

For all $i \in \{0,1,\ldots,n-1\}$ in parallel do

$$P_{i0}: m_i \leftarrow true$$

For all $i,j \in \{0,1,\ldots,n-1\}, i \neq j$ in parallel do

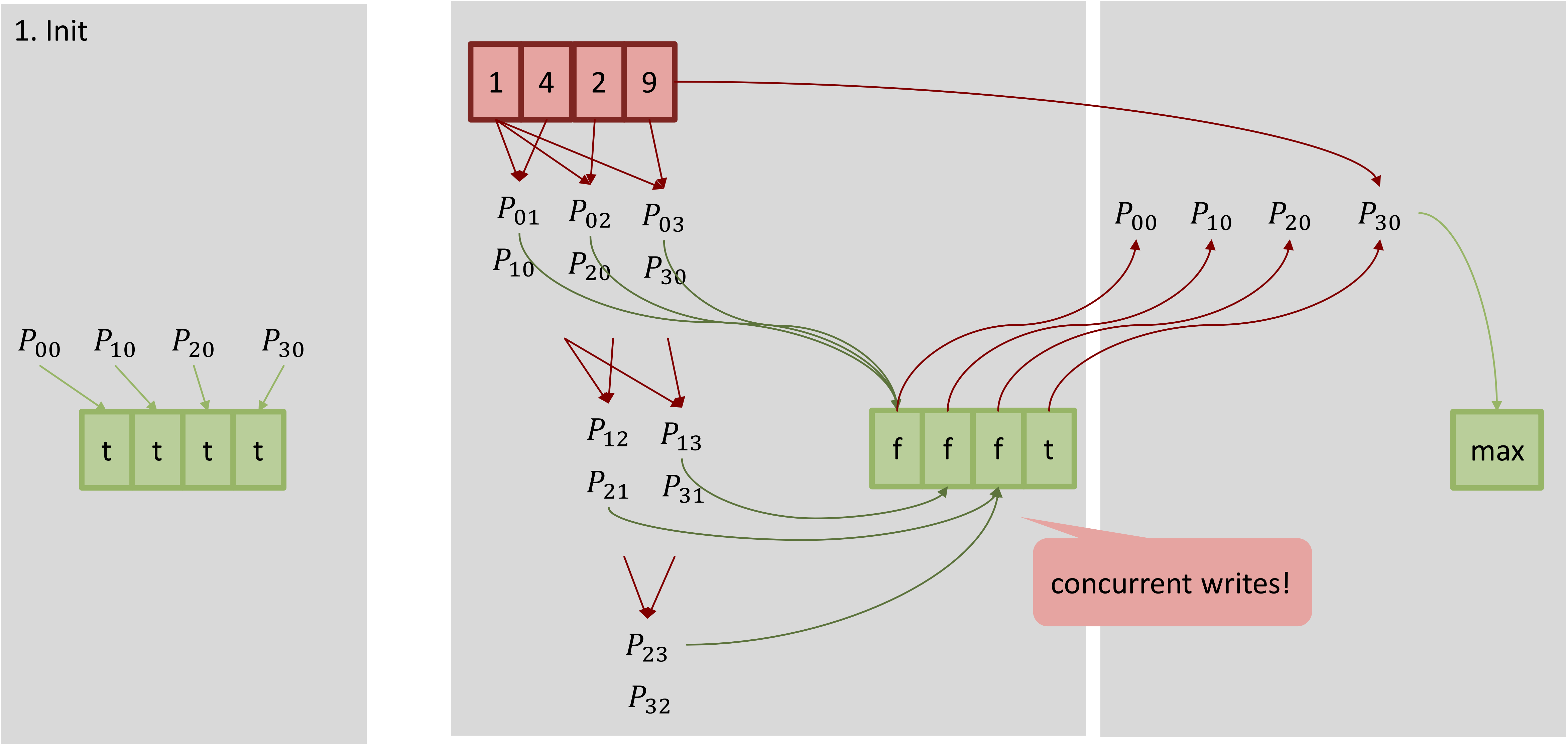
$$P_{ij}: if\ A_i < A_j\ then\ m_i \leftarrow false$$

For all $i \in \{0,1,\ldots,n-1\}$ in parallel do

$$P_{i0}: if\ m_i = true\ \ then\ max \leftarrow A_i$$

O(1) time complexity!

Illustration



CREW

Q: How many steps does max-find require with CREW?

Using CREW only two values can be merged into a single value by one processor at a time step: number of values that need to be merged can be halved at each step → Requires $\Omega(\log n)$ steps

There is a lot of interesting theoretical results for PRAM machine models (e.g., CRCW simulatable with EREW) and for PRAM based algorithms (e.g., cost optimality / time optimality proofs etc). We will not go into more details here.

In the following we assume a CREW PRAM model -- and receive in retrospect a justification for the results stated above on parallel bubble sorting.

How to compute fast?

March 2015

Last lecture -- basic exam tips

- **First of all, read all instructions**
- **Then, read the whole exam paper through**
- **Look at the number of points for each question**
  - This shows how long we think it will take to answer!
- **Find one you know you can answer, and answer it**
  - This will make you feel better early on.
- **Watch the clock!**
  - If you are taking too long on a question, consider dropping it and moving on to another one.
- **Always show your working**
- **You should be able to explain most of the slides**
  - Tip: form learning groups and present the slides to each other
  - If something is unclear:

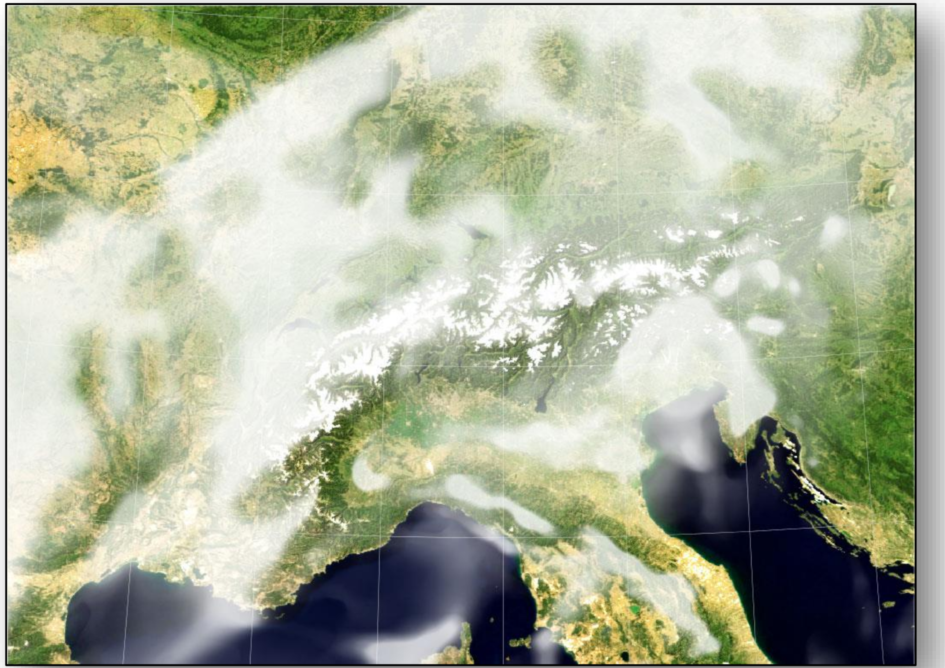
    Ask your friends

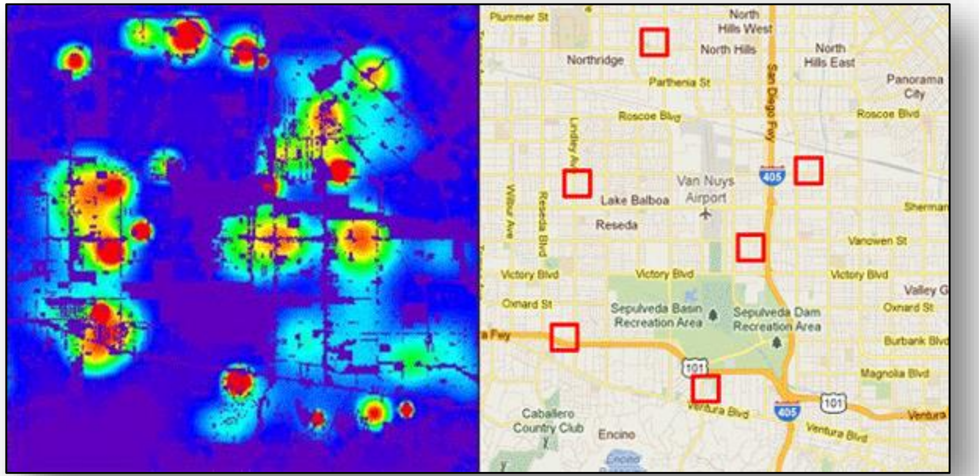
    Read the book (Herlihy and Shavit for the second part)

    Ask your TAs

Why computing fast?

- **Computation is the third pillar of science**





But why do I care!!?? Maybe you like the weather forecast?

Swiss Weather Forecasting Achieves 1.1km Resolution on 'Piz Kesch'

By John Russell

April 1, 2016

After six months of tweaking – producing a 20 percent reduction in time-to-solution for weather forecasting – MeteoSwiss, the Federal Office of Meteorology and Climatology, today reported its next generation COSMO-1 forecasting system is now operational. COSMO-1 requires 20 times the computing power of COSMO-2 and runs on the hybrid CPU-GPU supercomputer, Piz Kesch, operated by the Swiss National Supercomputing Centre (CSCS) and custom built in collaboration with Cray and NVIDIA.

COSMO-1 was put into service last September (see, Today's Outlook: GPU-accelerated Weather Forecasting, *HPCwire*) and improves resolution from 2.2 km to 1.1 km over COSMO-2, an important advance, particularly for Alpine topography forecasts where high spatial resolution is required to accurately predict local weather events such as thunderstorms and thermally induced mountain and valley wind systems.

Swiss First to Tap GPUs to Improve National Weather Forecasts

September 15, 2015 by ROY KIM

Ten years ago, Hurricane Katrina devastated New Orleans. Three years ago, Hurricane Sandy battered New York City. Hundreds lost their lives. Damages were in the billions.

Wherever you live, predicting the weather is a high-stakes game.

Now, thanks to GPU-accelerated computing, the Swiss have made significant advancements in their ability to predict storms and other weather hazards with higher levels of accuracy.

The Swiss Federal Office of Meteorology and Climatology, MeteoSwiss, is the first major national weather service to deploy a GPU-accelerated supercomputer to improve its daily weather forecasts.

Tobias Gysi,
PhD Student @SPCL

MeteoSchweiz und das CSCS gewinnen den Swiss ICT Award

Date of publication	15 November 2016
Topics	About us
Type	Press release

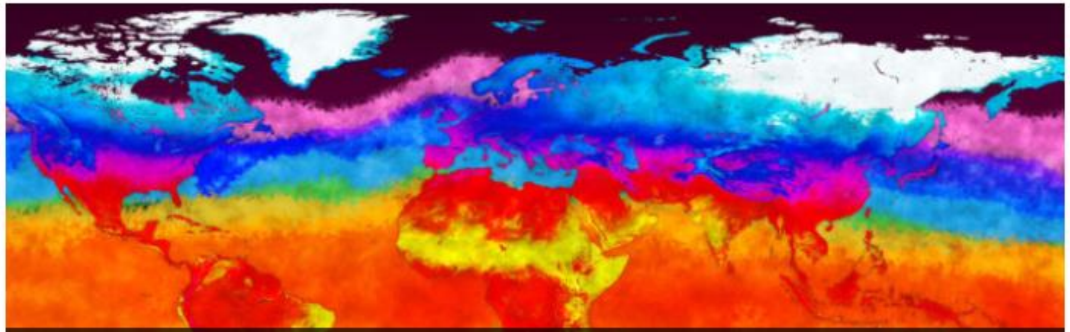
Die diesjährige Auszeichnung für ein besonderes IT-basiertes Produkt oder Service der Schweizer Informatikbranche geht an das Nationale Hochleistungszentrum der Schweiz (CSCS) und das Bundesamt für Meteorologie und Klimatologie MeteoSchweiz für ihr gemeinsames Projekt „Super-Wetterrechner". Nach Ansicht der Jury ist die neue Art der Berechnung riesiger Wetter-Datenmengen auf einem Supercomputer richtungsweisend.

MeteoSwiss and CSCS pave the way for more detailed weather forecasts

Date of publication	15 September 2015
Topics	Hazards Weather
Type	Press release

At the Swiss National Supercomputing Centre (CSCS) in Lugano the new "super weather computer" of the Swiss Federal Office of Meteorology and Climatology MeteoSwiss has started its operation. MeteoSwiss is the first meteorological service which has switched to a new GPU based computer architecture. Thus, the new supercomputer is able to calculate weather models with a resolution twice as high more efficiently and quicker than before.

Or you wonder about the future of the earth?



Researchers Scale COSMO Climate Code to 4888 GPUs on Piz Daint

By John Russell

October 17, 2017

Effective global climate simulation, sorely needed to anticipate and cope with global warming, has long been computationally challenging. Two of the major obstacles are the needed resolution and prolonged time to compute. This month a group of researchers from ETH Zurich, MeteoSwiss, and the Swiss National Supercomputing Center (CSCS) report scaling popular COSMOS code to run on all 4888 GPUs of CSCS's Piz Daint supercomputer and achieving ultra-high resolution.

In their paper, 'Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0', posted on the open access site, Geoscientific Model Development Discussion, authors present their rather extensive efforts necessary to port the code. Previously COSMO had only been scaled to 1000 GPUs on Piz Daint.

Near-global climate simulation at 1 km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0

Oliver Fuhrer[1], Tarun Chadha[2], Torsten Hoefler[3], Grzegorz Kwasniewski[3], Xavier Lapillonne[1], David Leutwyler[4], Daniel Lüthi[4], Carlos Osuna[1], Christoph Schär[4], Thomas C. Schulthess[5,6], and Hannes Vogt[6]

[1]Federal Institute of Meteorology and Climatology, MeteoSwiss, Zurich, Switzerland
[2]ITS Research Informatics, ETH Zurich, Switzerland
[3]Scalable Parallel Computing Lab, ETH Zurich, Switzerland
[4]Institute for Atmospheric and Climate Science, ETH Zurich, Switzerland
[5]Institute for Theoretical Physics, ETH Zurich, Switzerland
[6]Swiss National Supercomputing Centre, CSCS, Lugano, Switzerland

Correspondence: Oliver Fuhrer (oliver.fuhrer@meteoswiss.ch)

Received: 16 September 2017 – Discussion started: 5 October 2017
Revised: 7 February 2018 – Accepted: 8 February 2018 – Published: 2 May 2018

Abstract. The best hope for reducing long-standing global climate model biases is by increasing resolution to the kilometer scale. Here we present results from an ultrahigh-resolution non-hydrostatic climate model for a near-global setup running on the full Piz Daint supercomputer on 4888 GPUs (graphics processing units). The dynamical core of the model has been completely rewritten using a domain-specific language (DSL) for performance portability across different hardware architectures. Physical parameterizations and diagnostics have been ported using compiler directives. To our knowledge this represents the first complete atmospheric model being run entirely on accelerators on this scale. At a grid spacing of 930 m (1.9 km), we achieve a simulation throughput of 0.043 (0.23) simulated years per day and an energy consumption of 596 MWh per simulated year. Furthermore, we propose a new memory usage efficiency (MUE) metric that considers how efficiently the memory bandwidth – the dominant bottleneck of climate codes – is being used.

in the availability of water resources and the occurrence of droughts (Pachauri and Meyer, 2014).

Current climate projections are mostly based on global climate models (GCMs). These models represent the coupled atmosphere–ocean–land system and integrate the governing equations, for instance, for a set of prescribed emissions scenarios. Despite significant progress during the last decades, uncertainties are still large. For example, current estimates of the equilibrium global mean surface warming for doubled greenhouse gas concentrations range between 1.5 and 4.5 °C (Pachauri and Meyer, 2014). On regional scales and in terms of the hydrological cycle, the uncertainties are even larger. Reducing the uncertainties of climate change projections, in order to make optimal mitigation and adaptation decisions, is thus urgent and has a tremendous economic value (Hope, 2015).

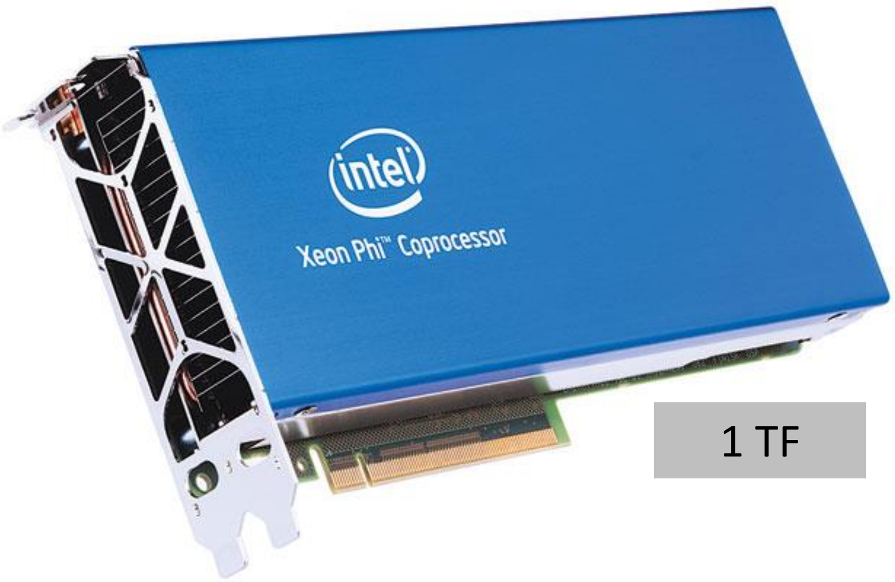
How can the uncertainties of climate projections be reduced? There is overwhelming evidence from the literature that the leading cause of uncertainty is the representation of clouds, largely due to their influence upon the reflection of

1 Teraflop in 1997

$67 Million

1 Teraflop 17 years later (2014)

Want to play with any of these?



1 TF

"Amazon.com by Intel even has the co-processor selling for just $142 (plus $12 shipping) though they seem to be now out of stock until early December." (Nov. 11, 2014)

[Update 2018]
7.8 Tflop/s double precision
15.7 Tflop/s single precision
125 Tflop/s half precision

1 Teraflop 20 years later (2017)

TECHNOLOGY

Intel's new chip puts a teraflop in your desktop. Here's what that means

It's as fast as a turn-of-the-century supercomputer.

By Rob Verger June 1, 2017

1 Teraflop 25 years later (2022)

1 Petaflop 35 years later (2032???)

Not so fast …

(or: performance became interesting again)

Changing hardware constraints and the physics of computing

How to address locality challenges on standard architectures and programming?

D. Unat et al.: “**Trends in Data Locality Abstractions for HPC Systems**”

IEEE Transactions on Parallel and Distributed Systems (TPDS). Vol 28, Nr. 10, IEEE, Oct. 2017

Three Ls of modern computing:

Spatial Locality

Temporal Locality

Control Locality

[1]: Marc Horowitz, Computing’s Energy Problem (and what we can do about it), ISSC 2014, plenary

[2]: Moore: Landauer Limit Demonstrated, IEEE Spectrum 2012

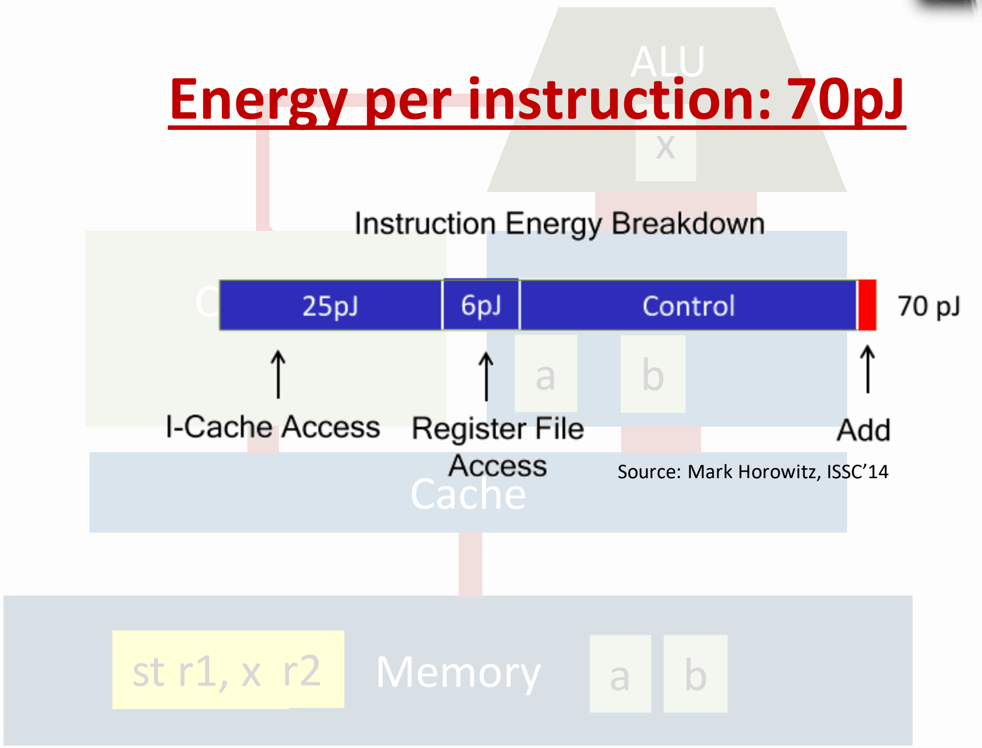
Load-store vs. Dataflow architectures

Turing Award 1977 (Backus): "Surely there must be a less primitive way of making big changes in the store than pushing vast numbers of words back and forth through the von Neumann bottleneck."

Load-store ("von Neumann")

x=a+b

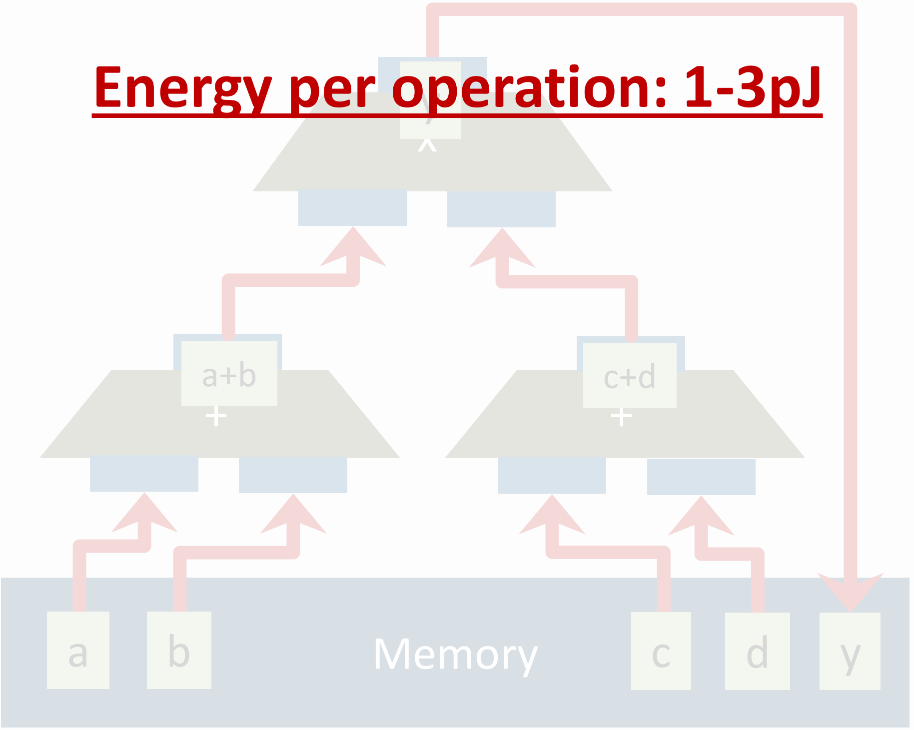
Energy per instruction: 70pJ



Static Dataflow ("non von Neumann")

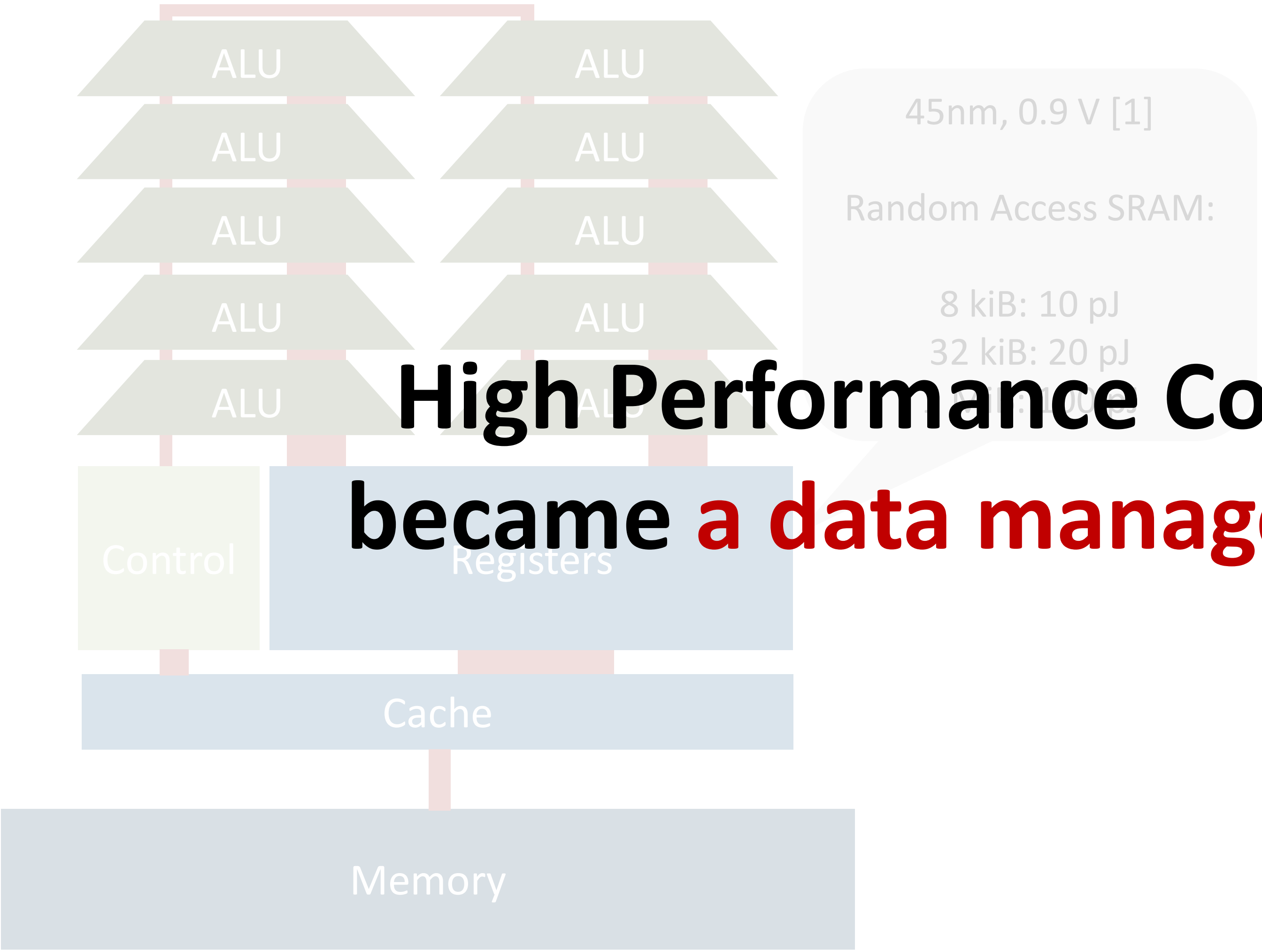
y=(a+b)*(c+d)

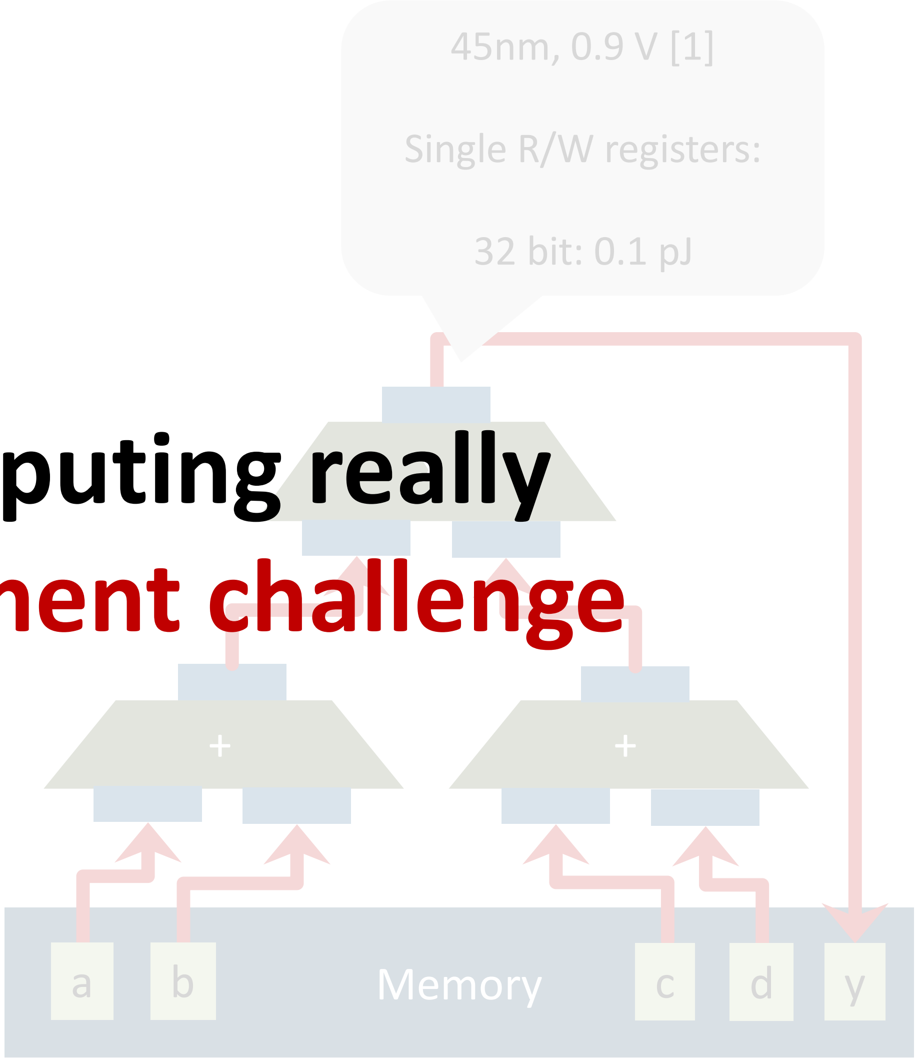
Energy per operation: 1-3pJ



Control Locality

Single Instruction Multiple Data/Threads (SIMD - Vector CPU, SIMT - GPU)





High Performance Computing really became a data management challenge

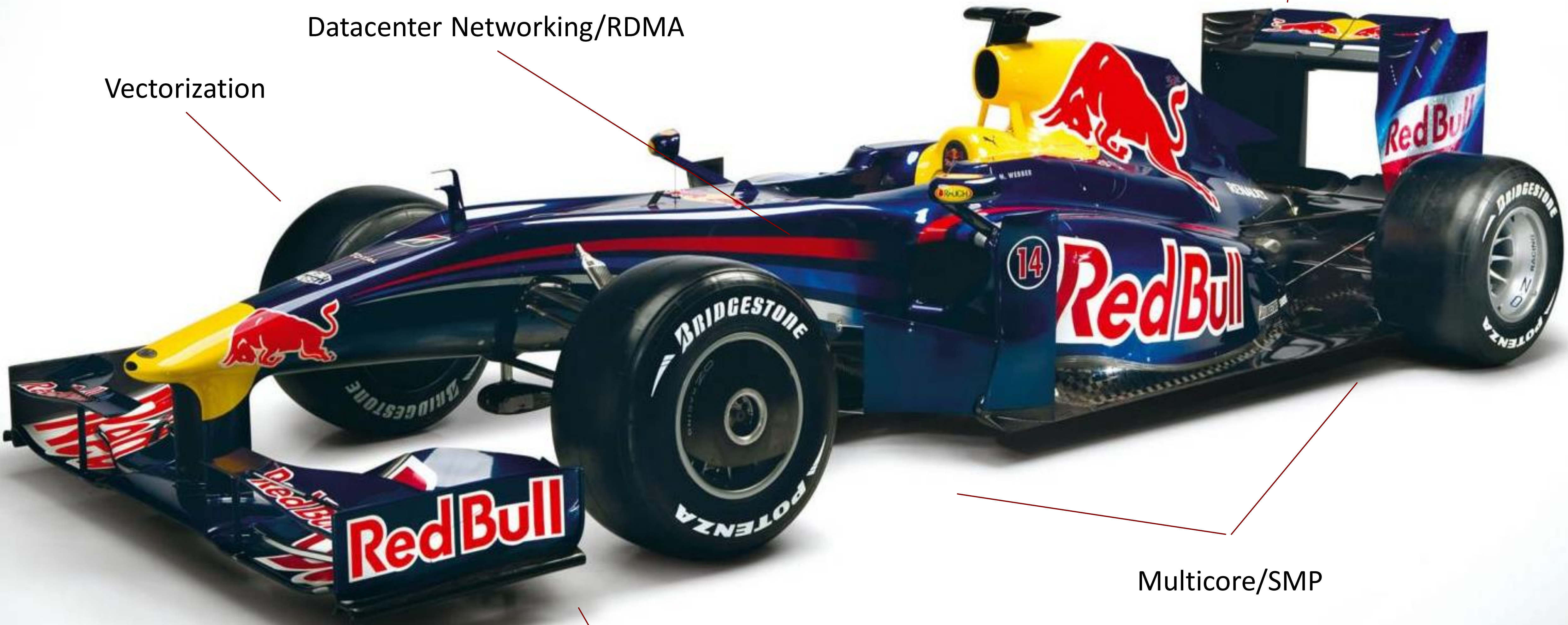
[1]: Marc Horowitz, Computing's Energy Problem (and what we can do about it), ISSC 2014, plenary

High-performance Computing (Supercomputing)

GPU/FPGA Computing

Datacenter Networking/RDMA

Vectorization



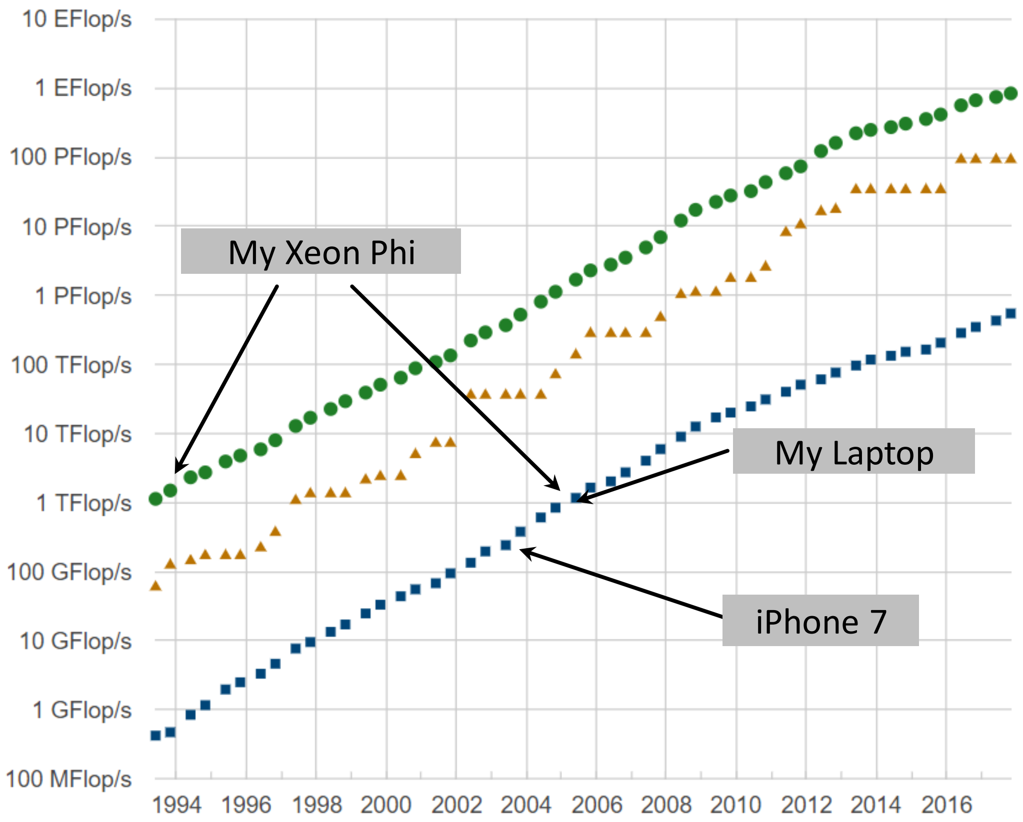
Multicore/SMP

IEEE Floating Point

…. next: specialized & reconfigurable computing

Top 500

- **A benchmark, solve Ax=b**
  - As fast as possible! → as big as possible ☺
  - Reflects **some** applications, not all, not even many
  - Very good historic data!
- **Speed comparison for computing centers, states, countries, nations, continents ☹**
  - Politicized (sometimes good, sometimes bad)
  - Yet, fun to watch



The November 2018 List

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384
6	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States	979,072	20,158.7	41,461.2	7,578
7	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649

DPHPC class of 2015

Want to run on that system?

www.top500.org

Computing Pi on a supercomputer!

```
int main( int argc, char *argv[] ) {
  // definitions …
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &myid);

  double t = -MPI_Wtime();
  for (j=0; j<n; ++j) {
    h   = 1.0 / (double) n;
    sum = 0.0;
    for (i = myid + 1; i <= n; i += numprocs) { x = h * ((double)i - 0.5);  sum += (4.0 /
    mypi = h * sum;
    MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD
  }
  t+=MPI_Wtime();

  if (!myid) {
    printf("pi is approximately %.16f, Error is %.16f\n", pi, fabs(pi - PI25DT));
    printf("time: %f\n", t);
  }

  MPI_Finalize();
}
```

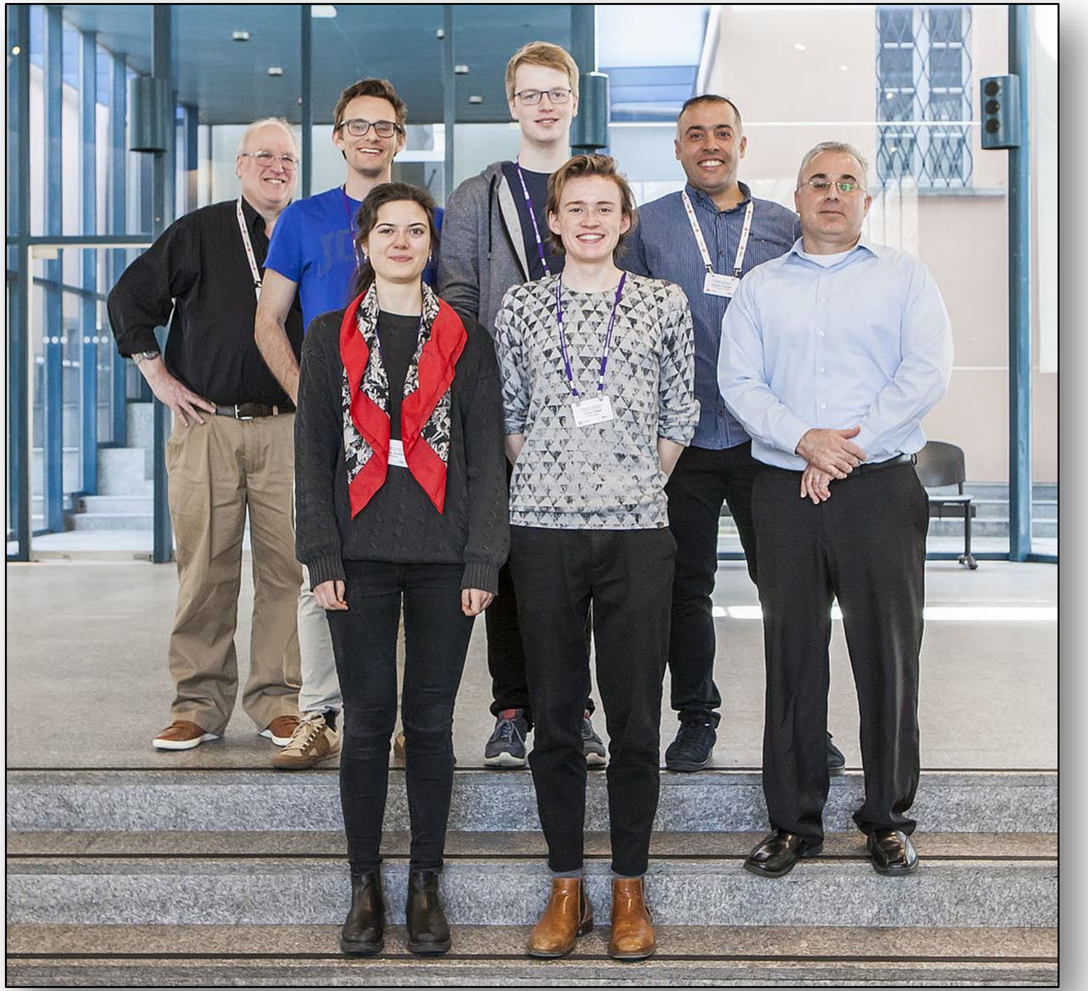
htor@hassi:~

```
htor@daint104:~> salloc --partition debug -N 4 -C mc -t 10
salloc: Pending job allocation 7815988
salloc: job 7815988 queued and waiting for resources
salloc: job 7815988 has been allocated resources
salloc: Granted job allocation 7815988
salloc: Waiting for resource configuration
salloc: Nodes nid000[08-11] are ready for job
htor@daint104:~> srun -n 1 ./a.out
srun: Warning: can't run 1 processes on 4 nodes, setting nnodes to 1
pi is approximately 3.1415926535981167, Error is 0.0000000000083236
time: 13.022794
htor@daint104:~> srun -n 4 ./a.out
pi is approximately 3.1415926535981260, Error is 0.0000000000083329
time: 3.598728
htor@daint104:~> srun -n 8 ./a.out
pi is approximately 3.1415926535981251, Error is 0.0000000000083320
time: 2.120363
htor@daint104:~> srun -n 16 ./a.out
pi is approximately 3.1415926535981269, Error is 0.0000000000083338
time: 1.366739
htor@daint104:~> srun -n 32 ./a.out
pi is approximately 3.1415926535981265, Error is 0.0000000000083333
time: 1.034170
htor@daint104:~> srun -n 64 ./a.out
pi is approximately 3.1415926535981269, Error is 0.0000000000083338
time: 0.859992
htor@daint104:~> srun -n 128 ./a.out
pi is approximately 3.1415926535981269, Error is 0.0000000000083338
time: 0.740548
htor@daint104:~> srun -n 256 ./a.out
pi is approximately 3.1415926535981269, Error is 0.0000000000083338
time: 0.953909
htor@daint104:~>
```

Student Cluster Competition

Want to become an expert in HPC?

- **6 undergrads, 1 advisor, 1 cluster, 2x13 amps**
  - 20 teams, most continents @SC or @ISC
  - 48 hours, five applications, non-stop!
  - top-class conference (>13,000 attendees)
- **Lots of fun**
  - Even more experience!
- **Introducing team Racklette**
  - https://racklette.ethz.ch/
  - Search for "Student Cluster Challenge"
  - HPC-CH/CSCS is helping
- **Let me know, my assistants are happy to help!**
  - If we have a full team



Finito

- **Thanks for being such fun to teach ☺**
  - Comments (also anonymous) are always appreciated!
- **If you are interested in parallel computing research, talk to me or my assistants!**
  - Large-scale (datacenter) systems
  - Next-generation parallel programming (e.g., FPGAs)
  - Parallel computing (SMP and MPI)
  - GPUs (CUDA), FPGAs, Manycore …
  - … spcl-friends mailing list (subscribe on webpage)
  - … on twitter: @spcl_eth ☺

  - Hope to see you again! *Maybe in Design of Parallel and High-Performance Computing in the Masters ☺*
  - Or for theses/research projects: *http://spcl.inf.ethz.ch/SeMa/*