Data Structures
and
Models of Computation

Gerth Stølting Brodal

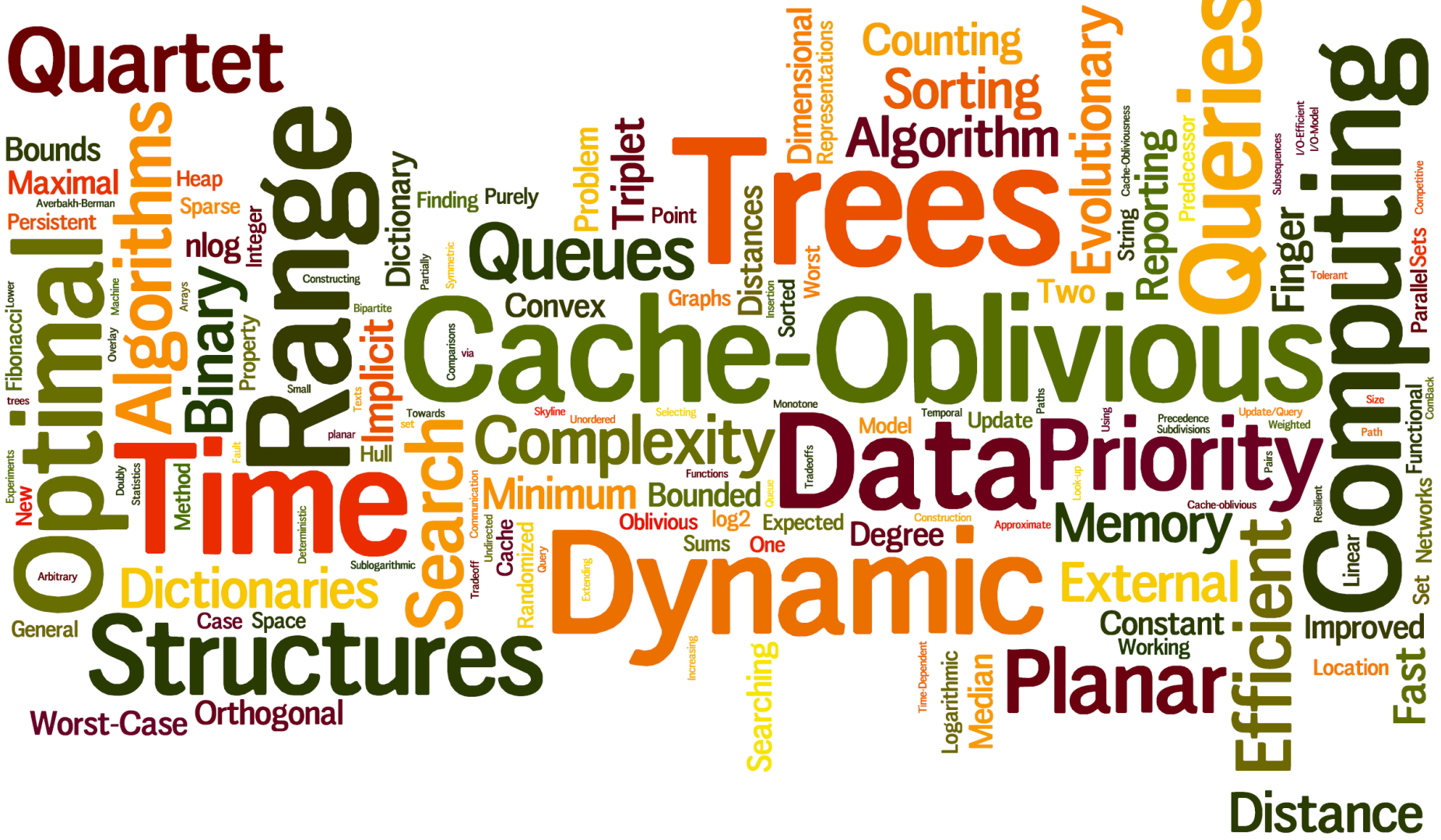
Inauguration talk, Department of Computer Science, Aarhus University, February 12, 2016

Looking back... publications since 1994

STACS12	APBC13	ISAAC02	ESA10	NJC96	SWAT96	SODA12
ESA13	IPPS97	SWAT00	JFP96	SWAT14	SODA03	SODA02
ISAAC10	SODA00	ESA06	WADS95	SODA98	WADS15	Algorithmica02
PhD97	CPM99	ORS08	STOC12	APBC07	ESA07	Handbook05
TAMC11	ISAAC09	FOCS00	ISAAC10	SODA13	ISAAC08	Encyclopedia08
ICALP01	ESA06	ISAAC12	STOC01	STACS97	FOCS02	STOC02
TCS11	CPM06	ESA12	ISAAC09	ICALP11	WADS09	BRICS-RS-96
FOCS06	ESA13	Biology13	SWAT96	SWAT04	WABI03	Handbook05
IPL00	SOCG05	SPAA07	ICALP02	ICALP02	CPM00	Algorithmica04
STACS16	ISAAC01	ICALP05	SODA11	MFCS07	ISAAC09	SODA99
SWAT98	ALENEX14	ISAAC09	APBC07	SODA96	ALENEX05	BRICS-RS-96
MFCS07	WADS99	SWAT02	SODA06	SWAT04	Bioinformatics14	
FOCS03	SWAT98	ATMOS03	Survey13	SOCG08	Progress-Report94	
SWAT00	ICATPN07	ALENEX04	CPM96	SODA10	Acta-Informatica05	
WADS11	SWAT14	WADS05	WAOA11	STOC03	BMC-Bioinformatics06	

Thanks to my co-authors

Peyman Afshani, Pankaj K. Agarwal, Stephen Alstrup, **Lars Arge**, Djamal Belazzougui
Michael A. Bender, Andrej Brodnik, Shiva Chaudhuri, Pooya Davoodi, Erik D. Demaine
Rolf Fagerberg, Jeremy T. Fineman, Irene Finocchi, Dongdong Ge, Loukas Georgiadis
Beat Gfeller, Fabrizio Grandoni, Mark Greve, Leszek Gasieniec, Inge Li Gørtz
Kristoffer A. Hansen, Simai He, Morten Kragelund Holt, Haodong Hu, Thore Husfeldt
John Iacono, Giuseppe Italiano, **Riko Jacob**, Jens Johansen, **Allan Grønlund Jørgensen**
Kanela Kaligosi, Alexis Kaporis, Alexis C. Kaporis, Jyrki Katajainen, Irit Katriel
Casper Kejlberg-Rasmussen, Lars Michael Kristensen, Martin Kutz
Alejandro López-Ortiz, George Lagogiannis, Stefan Langerman, Kasper Green Larsen
Morten Laustsen, Moshe Lewenstein, Rune Bang Lyngsø, **Thomas Mailund**
Christos Makris, Ulrich Meyer, **Gabriel Moruz**, J. Ian Munro, Thomas Mølhave
Andrei Negoescu, Jesper Sindahl Nielsen, Chris Okasaki, Vineet Pandey
Apostolos Papadopoulos, **Christian Nørgaard Storm Pedersen**, Derek Phillips
M. Cristina Pinotti, Jaikumar Radhakrishnan, Rajeev Raman, **S. Srinivasa Rao**
Theis Rauhe, Andreas Sand, Peter Sanders, Spyros Sioutas, Sven Skyum
Venkatesh Srinivasan, Martin Stissing, Jens Stoye, Robert E. Tarjan, Laura Toma
Jakob Truelsen, Jesper Larsson Träff, Athanasios Tsakalidis, Konstantinos Tsakalidis
Kostas Tsichlas, Constantinos Tsirogiannis, Elias Vicari, Kristoffer Vinther, Jeff Vitter
Michael Westergaard, Christos D. Zaroliagis, Norbert Zeh, Anna Östlin



Data Structures

Dictionaries

CPM96 STACS97 IPL00 SODA02 FOCS03
SODA06 ESA06 ESA07 WADS09 SODA10
ISAAC10 STACS12 SODA13
Finger search
SODA98 STOC02 Handbook05 ISAAC12

Computational Geometry

Point location SODA99 FOCS06 SOCG08
Range searching
FOCS00 STOC01 SOCG05 ISAAC09-09 SODA11
ICALP11 SWAT14 STACS16
Convex hull SWAT00 FOCS02
Distribution sweeping ICALP02

Priority Queues

WADS95 SODA96 SWAT96 STOC12 Survey13
JFP96 SWAT96 IPPS97 SWAT98 SWAT98
ISAAC02 WADS15

Sorting

STOC03 WADS05 ALENEX04 ALENEX05
ICALP05 Encyclopedia08 SWAT14

Phylogenetic Trees

Quartets/Triplets comparison ISAAC01 Algorithmica04 APBC07-07-13
Biology13 SODA13 ALENEX14 Bioinformatics14
Construction : experiments, Bunemann, neighbor-joining
ICALP01 WABI03 BMC-Bioinformatics06

Graphs

WADS99 SWAT00 ATMOS03
SWAT04 MFCS07

Selection

MFCS07 ISAAC08 ORS08 ISAAC09
TCS11

String Searching

CPM99 SODA00 CPM00 ICALP02
CPM06

System

SWAT02 Acta-Informatica05
ISAAC10 WAOA11

Range Minimum

Matrix ESA10 ESA12 ESA13
Trees WADS11

Miscellaneous

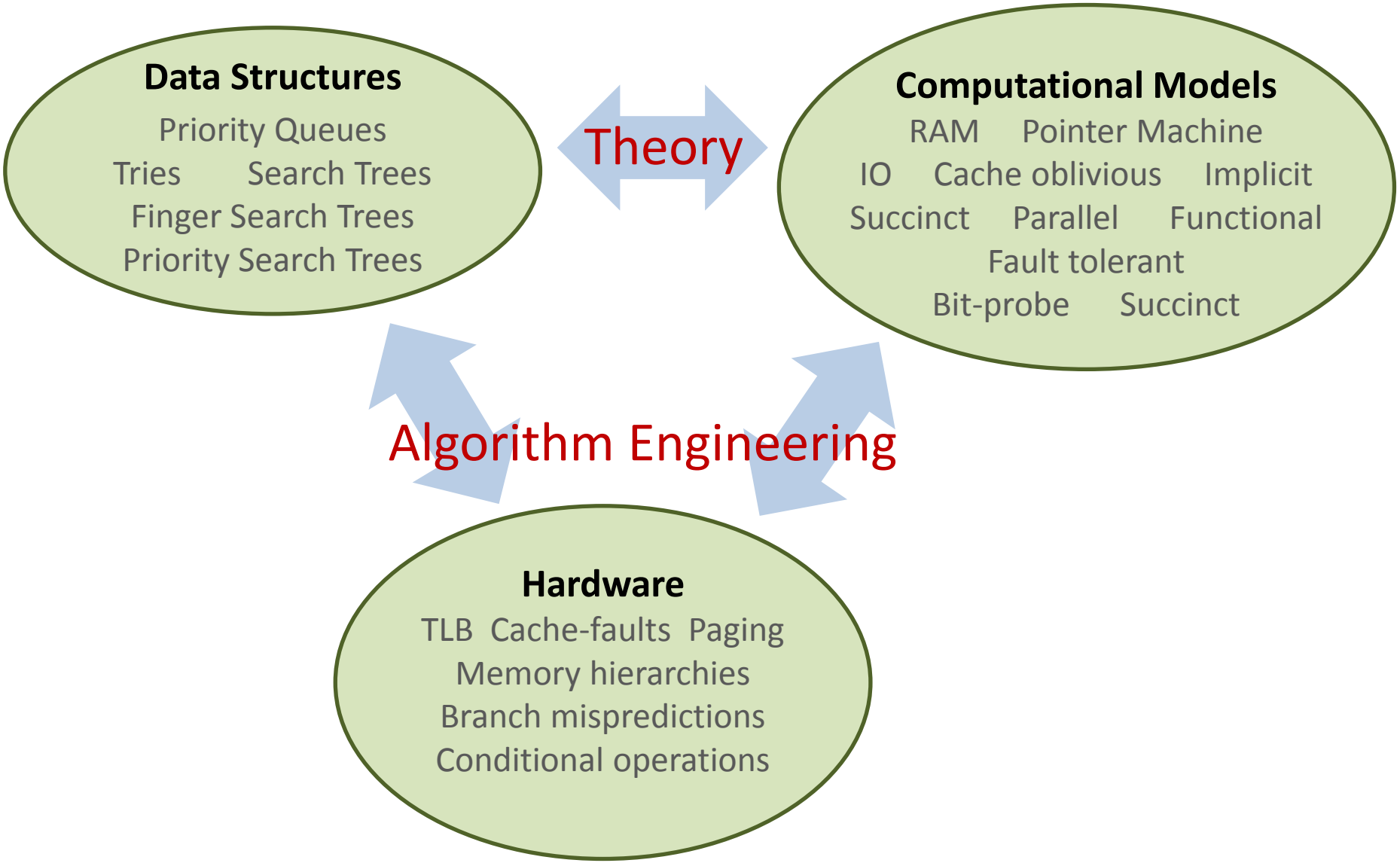
Persistence NJC96 SODA12
Lists ESA06 Algoritmica02
Counting ISAAC09 TAMC11

Progress-Report94 PhD97 SWAT04 Handbook05

BRICS-RS-96 BRICS-RS-96 ICATPN07 SPAA07 ESA13

IO Cache oblivious Implicit Parallel Functional Fault tolerant

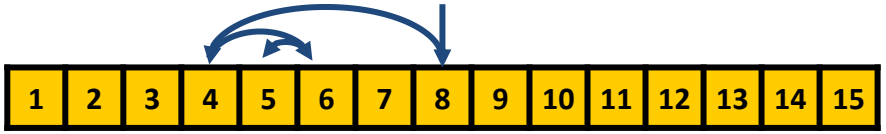
Research Context



Example 1 : Binary Search

BSc (Algorithms and Data Structures 1)

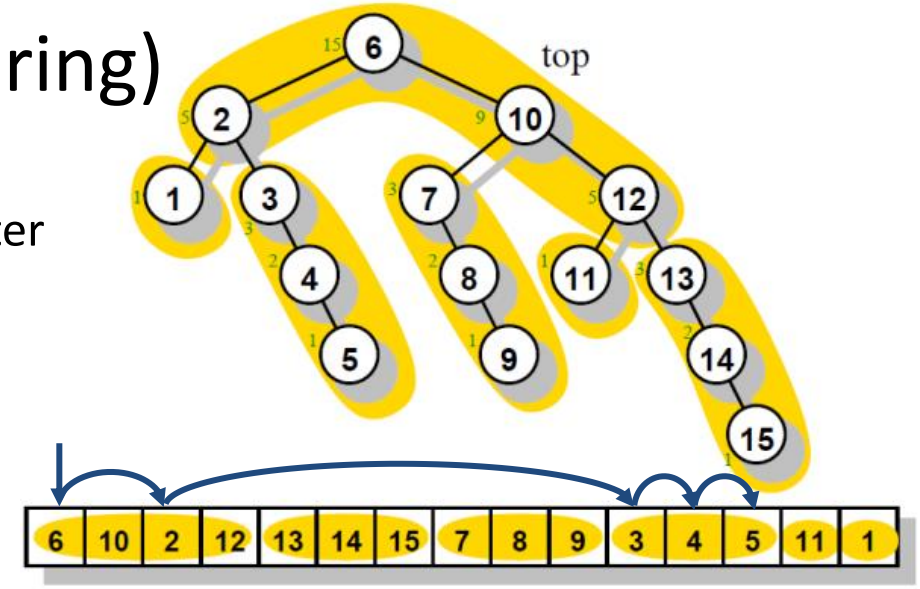
- Store data in sorted order
- Binary search from the center



MSc (Algorithm Engineering)

- **Don't** store data in sorted order
- **Don't** do binary search from the center

Why?
cache lines and branch prediction



Sorting **WADS05**
$O(n \cdot \log n)$ comparisons
$\rightarrow \Omega(n \cdot \log n)$ mispredictions

ESA06

Decimal	Binary	Reflected Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	Reads 4 bits Writes 4 bits	Reads 4 bits Writes 1 bit
10		
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000
0	0000	0000

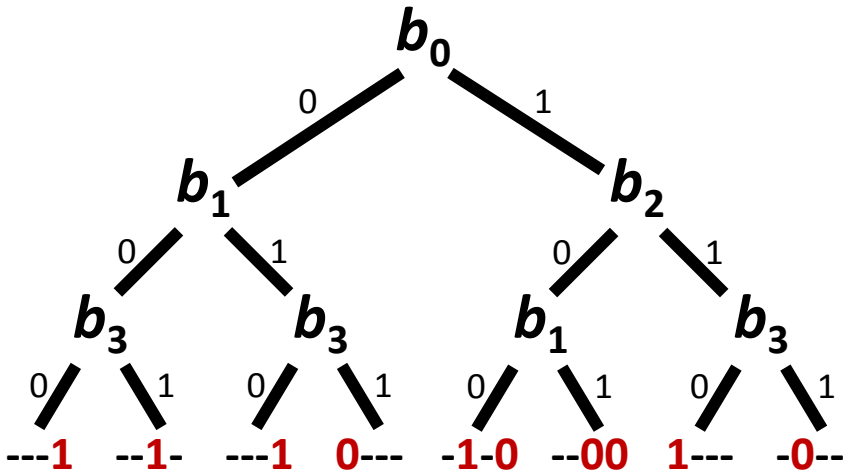
Example 2 : Counting [Bit-probe]

Question

Does there exist a counter where one never needs to read all bits to increment the counter ?

Example 2 : Counting [Bit-probe]

Decimal	$b_3b_2b_1b_0$
0	0000
1	0001
2	0100
3	0101
4	1101
5	1001
6	1100
7	1110
8	0110
9	0111
10	1111
11	1011
12	1000
13	1010
14	0010
15	0011
0	0000



Always reads 3 bits
Always writes ≤ 2 bits

General case

n-bit counter ***n*-1 reads** and 3 writes

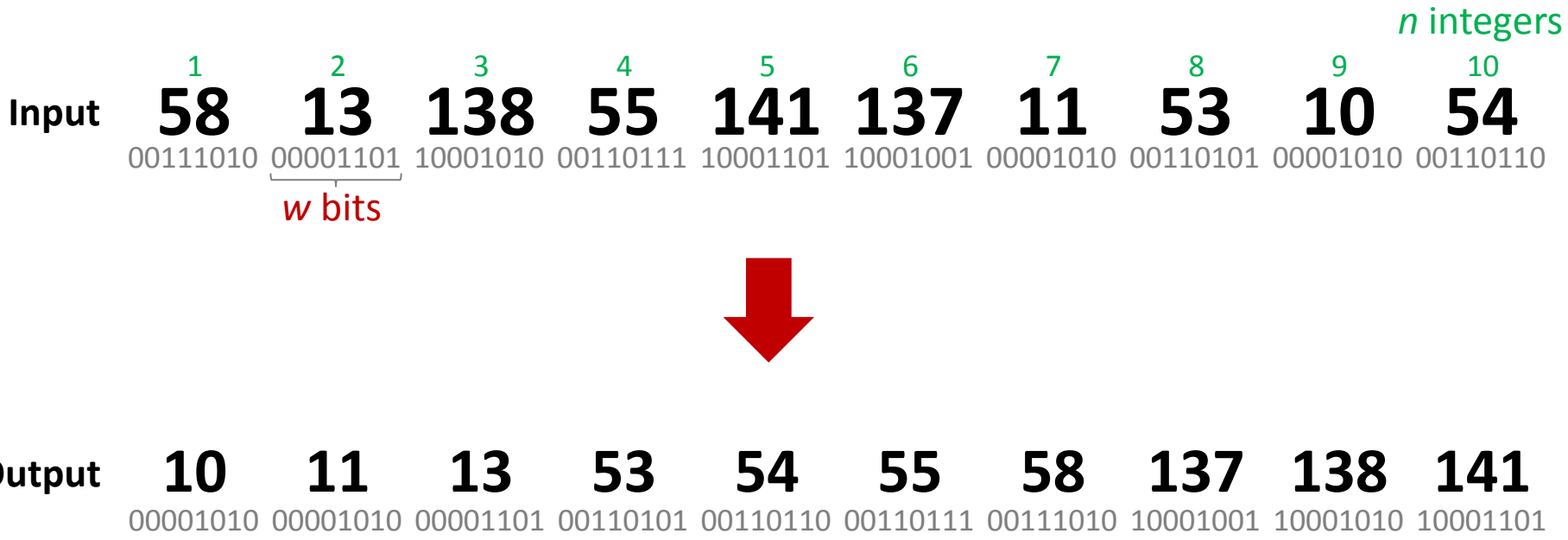
lower bound $\log_2 n$ reads

Ridiculous

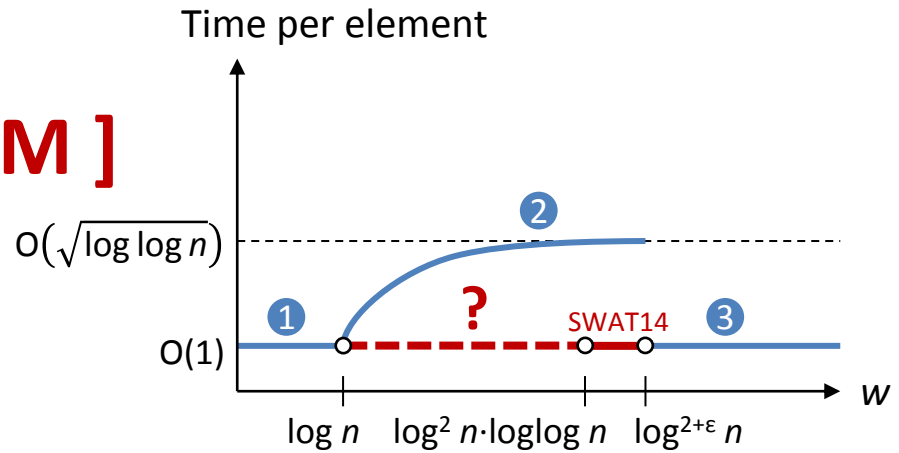
Redundant counters

use 1 extra bit → Θ(log *n*) reads sufficient

Example 3 : Integer Sorting [RAM]

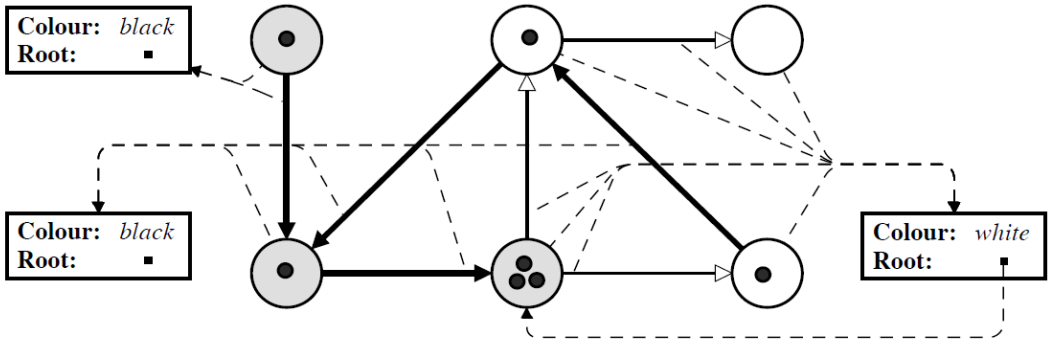


Example 3 : Integer Sorting [RAM]



1	Bucket sort	$O(n+2^w)$	
	Radix sort; Hollerith 1887	$O\left(n\frac{w}{\log n}\right)$	
	van Emde Boas 1975 Willard 1983	$O(n \log w)$	superlinear space expected
	Kirkpatrick and Reicsh 1983	$O\left(n \log \frac{w}{\log n}\right)$	
	Merge sort: von Neumann 1945	$O(n \log n)$	comparison based optimal
2	Thorup and Han 2002	$O\left(n\sqrt{\log (w/\log n)}\right)$ $O\left(n\sqrt{\log\log n}\right)$	expected
3	Andersson et al. 1998	$O(n)$	expected, $w \geq \Omega\left(\log^{2+\varepsilon} n\right)$
	SWAT14	$O(n)$	expected, $w \geq \Omega\left(\log^2 n \cdot \log\log n\right)$

Open $w \leq \log^2 n$




```
procedure BREAK(v)
    if v.cr.colour = black then
        r ← v
    else
        r ← v.cr.root
        v.cr.colour ← black
        v.cr.root ← ⊥
    endif
    if r.Q ≠ ∅ then
        u ← ROTATE(r.Q)
        if u.cr.colour = black then
            u.cr ← new-component-record(white, u)
        endif
        r.cr ← (u, r).cr ← u.cr
    endif
    ZERO(r)
end.
```

Progress-Report94

Example 4 : Persistence

[Pointer Machine]

Driscoll et al. STOC86

General techniques to make data structures of constant degree **partial** and **fully persistent** with constant **amortized** overhead.

Theorem

Pointer based data structures of constant degree can be made **partial persistent** with **worst-case** constant overhead.

NJC96

Open if worst-case full persistence result exist ?

Fully persistent B-tree **SODA12**

Chapter 3

A Counter

The result of this chapter is a new counter where it is possible to increase and decrease an arbitrary "bit". More precisely we implement a data type over Z that supports the following three operations.

- ADD(i) adds 2^i to the counter,
- SUB(i) subtracts 2^i from the counter,
- ZERO tests if the counter is equal to zero.

We assume that the length (i.e. the number of bits in the binary representation) of the counter is bounded by N. Our result is:

Theorem 2 *A data structure exists that implements* ADD(i), SUB(i) *and* ZERO *in worst case time* $O(1)$.

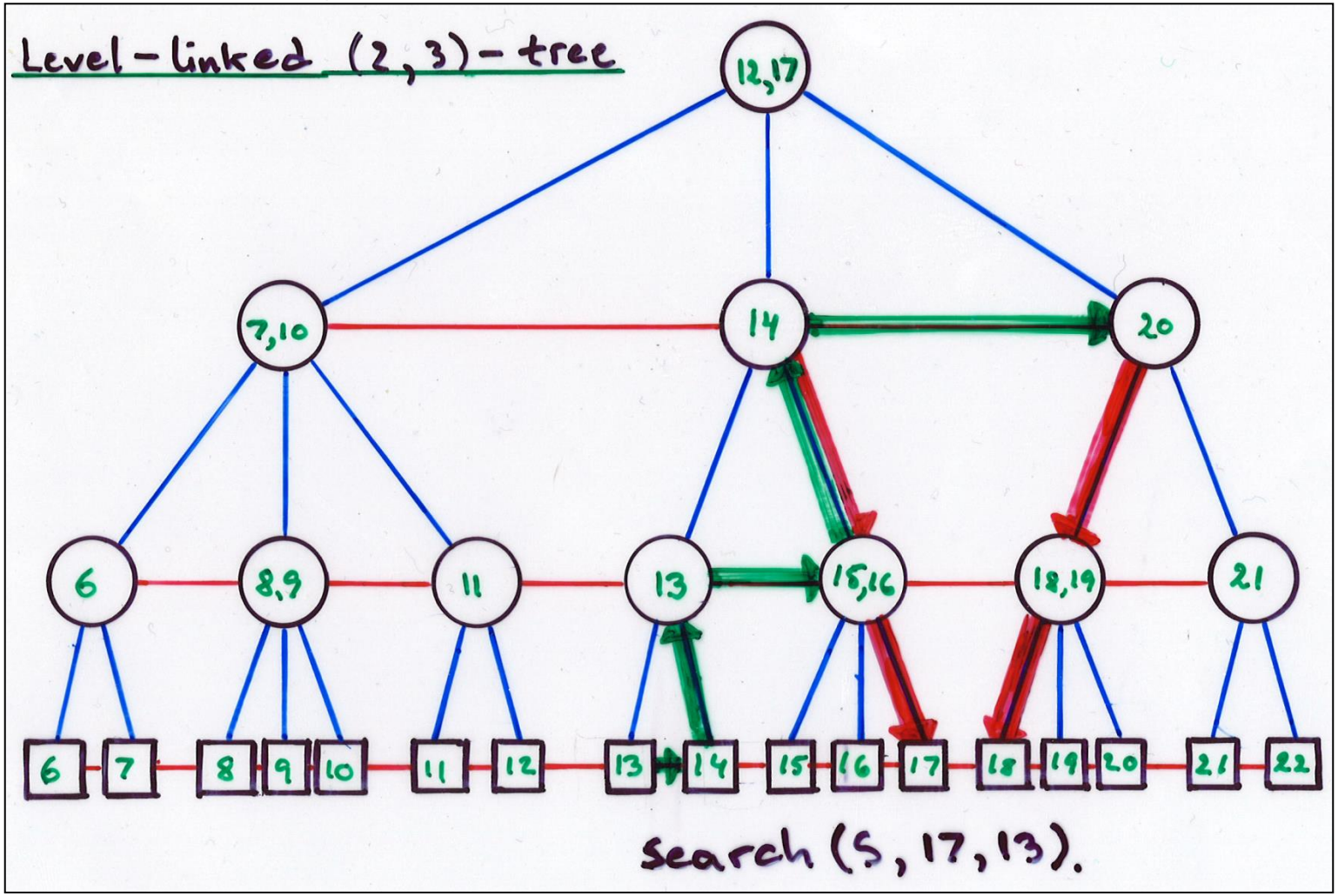
Example 5 : Finger Search Trees

[Pointer Machine]

- Failed to solve cascaded splittings in full persistence technique

- Turned towards the **simpler** (but related) problem of cascaded splittings in (a,b)-trees with finger search

Finger Search Trees



Brown and Tarjan 1980

A two player game related to finger search trees

The game:

Player A: Can insert/delete an element from one of the leaves of a tree.

Player B: Can do one of the actions:

i) Split an internal node

ii) Fusion an internal node with one of its brothers.

iii) Move a number of sons from an internal node to its neighbour brother.

Theorem:

A strategy for player B exists that results in trees with internal nodes of degrees between a and b where $2 \leq a < b$ and a, b are constants.

Problem:

How to find the place to do the update in O(1) time!

13

May 1994, *Finger Search Trees*. BRICS Strategy Workshop. Hjarnø, Denmark

Conclusion and Open Problems

Thm A pointer based implementation of finger search trees exists, supporting:

Insert in worst-case O(1) time, and
Search –"– O(log S) time.
The data structure requires linear space.
Delete can be supported in worst-case $O(\log^* n)$ time.

Open problems

- Delete in O(1) too.
- Make other splitting based data structures worst-case.
  Ex. Full Persistence.
  Dynamic Fractional Cascading.

14.

August 1997, *Finger Search Trees with Constant Insertion Time*. Oberwolfach Seminar on ``Effiziente Algoritmen''

History

	Insert	Delete	Search
AVL-trees, (2,3)-trees [a]		$\log n$	$\log n$
Red-black-trees, (2,4)-trees [a]		1 [b]	$\log n$
Levcopoulos, Overmars '88 [a]		1	$\log n$
Guibas et al. '77, Tsakalidis '85 [a]		1 [c]	$\log d$
Harel, Lucker '79 [a]		$\log^* n$	$\log d$
Huddleston, Mehlhorn '82 [a]		1 [b]	$\log d$
Brodal '98 [a]	1	$\log^* n$	$\log d$
This talk [a]		1	$\log d$
Dietz, Raman '94 [d]		1	$\log d$
Andersson, Thorup '00 [e]		1	$\sqrt{\frac{\log d}{\log \log d}}$

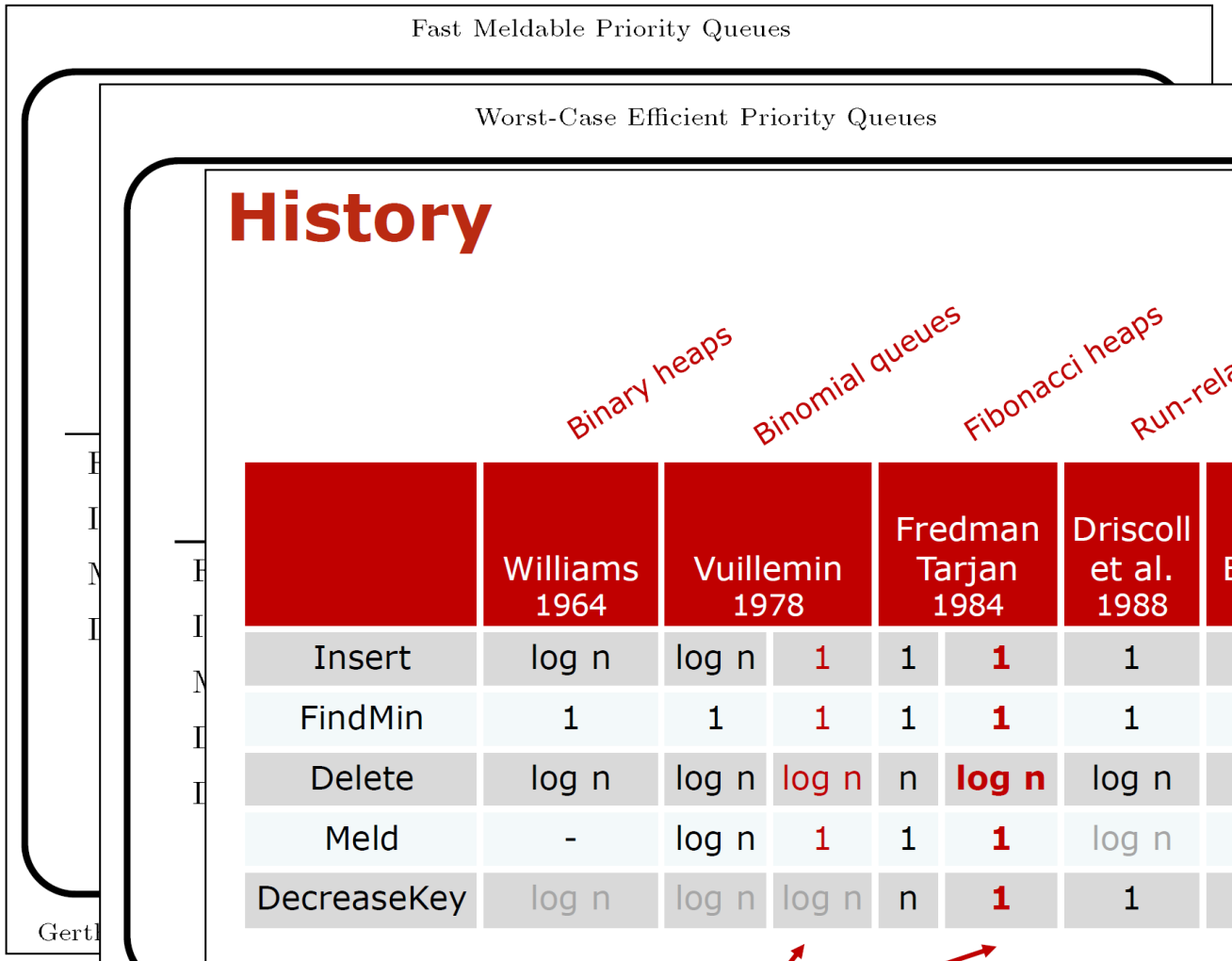
[a] Pointer machine [b] Amortized [c] $O(1)$ movable fingers
[d] Comparison RAM [e] Word RAM

February 2002, *Optimal Finger Search Trees in the Pointer Machine*. Dagstuhl Seminar on ``Data Structures''

Dagstuhl

Example 6 : Meldable Priority Queues

[RAM & Pointer Machine & Functional]



JFP96 [Functional]

History

Binary heaps | Binomial queues | Fibonacci heaps | Run-relaxed heaps | Strict Fibonacci heaps

	Williams 1964	Vuillemin 1978		Fredman Tarjan 1984		Driscoll et al. 1988	Brodal 1995	Brodal 1996	Brodal Lagogianis Tarjan STOC 2012
Insert	log n	log n	1	1	**1**	1	1	1	**1**
FindMin	1	1	1	1	**1**	1	1	1	**1**
Delete	log n	log n	log n	n	**log n**	log n	log n	log n	**log n**
Meld	-	log n	1	1	**1**	log n	1	1	**1**
DecreaseKey	log n	log n	log n	n	**1**	1	log n	1	**1**

Arrays Complicated | Pointer Based

Amortized complexity (Tarjan 1983)

WADS95 [Pointer] SODA96 [RAM] STOC12 [Pointer]

Example 7 : Functional Catenable Sorted Lists

[Functional]

- Join((1,3,9,11),(14,19,21)) → (1,3,9,11,14,19,21)

	Search trees	Kaplan Tarjan'96	This talk
Search	log n	log n	log n
Insert/Delete	log n	log n	log n
Join	log n 1*	loglog n_s	1
Split	log n	log n_s	-

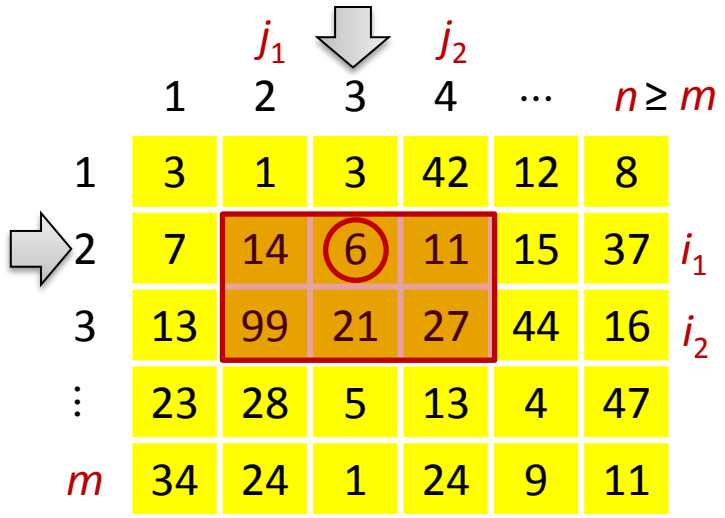
Extend to Finger Search ?

Open

$n = |T|$ $n_s = \min(|T_1|,|T_2|)$ * amortized

Example 8 : Range Minimum Queries [Succinct]

RMQ(i_1, i_2, j_1, j_2) = (2,3)
= **position** of min



	Indexing Model (input accessible)	**Encoding Model** (input not accessable)
$m = 1$ 1D	$2n+o(n)$ bits, $O(1)$ time [FH07] n/c bits $\Rightarrow$ $\Theta(c)$ time ESA10	$\geq 2n - O(\log n)$ bits $2n+o(n)$ bits, $O(1)$ time [F10]
$1<m<n$	$O(mn \cdot \log n)$ bits, $O(1)$ time [AY10] $O(mn)$ bits, $O(1)$ time ESA10 mn/c bits $\Rightarrow$ $\Omega(c)$ time ESA10	$\Omega(mn \cdot \log m)$ bits ESA10 $O(mn \cdot \log n)$ bits, $O(1)$ time ESA10 $O(mn \cdot \log m)$ bits, $O(mn)$ time ESA13
$m = n$ squared	$O(c \cdot \log^2 c)$ time ESA10 $O(c \cdot \log c \cdot (\text{loglog } c)^2)$ time ESA12	$\Theta(mn \cdot \log n)$ bits, $O(1)$ t… [DLW09,AY10]

better upper or lower bound ?

Completely Ridiculous

Summary

- Classic data structures
  - still a lot of open problems
- Lesson learned
  - some problems take long time to solve
- Future
  - new technologies

    $\rightarrow$ new models of computation ?

    $\rightarrow$ intrinsic trade-offs ?