Curriculum Vitæ

Name	Gerth Stølting Brodal
Born	12 September 1969
	Sønderborg, Denmark
Citizenship	Danish
	Married, two children
Address	Department of Computer Science
	Aarhus University
	Åbogade 34
	8200 Aarhus N
	Denmark
Phone	+45 50 59 54 32
Email	gerth@cs.au.dk
W^3	http://www.cs.au.dk/~gerth
Bibliographies	ORCID: 0000-0001-9054-915X
	DBLP, Google Scholar (H-index: 41)

Brief Biography

Gerth Stølting Brodal is a Professor at the Department of Computer Science, Aarhus University, Denmark (since January 2016). He received his PhD in computer science in 1997 from Aarhus University for the thesis "Worst Case Efficient Data Structures". From 1997 to 1998 he was a PostDoc in the group of Kurt Mehlhorn at the Max-Planck-Institute for Computer Science in Saarbrücken, Germany. 1998–2005 he was affiliated with BRICS (Center for Basic Research in Computer Science) located at the Department of Computer Science, Aarhus University. 2004–2015 he was an Associate Professor (tenured) at the Department of Computer Science, Aarhus University. March 2007–December 2017 he was affiliated with MADALGO (Center for Massive Data Algorithmics), Aarhus University, founded by the Danish National Research Foundation.

His main research interests are the design and analysis of algorithms and data structures. He has done work on fundamental data structures, including dictionaries and priority queues, persistent data structures, computational geometry, graph algorithms, string algorithms, I/O-efficient and cache-oblivious algorithms and data structures, algorithm engineering, and computational biology.

Education

Feb 1993 – Apr 1997	PhD in Computer Science, Aarhus University, Denmark. Dissertation: Worst Case Efficient Data Structures. Advisor: Erik Meineche Schmidt. Committee: Mogens Nielsen (Aarhus), Arne Andersson (Lund), and J. Ian Munro (Waterloo).
Aug 1989 – Nov 1994	MSc (cand.scient.) in Computer Science and Mathematics, Aarhus University, Denmark.
Aug 1988 – May 1989	Military service, Jyske Telegrafregiment, Fredericia, Denmark.
Aug 1985 – Jun 1988	"Studentereksamen", Aabenraa Gymnasium og HF, Aabenraa, Denmark.

Positions

Jan 2016 –	Professor, Department of Computer Science, Aarhus University.
Apr 2009 – Dec 2015	Associate Professor (tenured, Lektor MSK), Department of Computer Science, Aarhus University.
Apr 2004 – Mar 2009	Associate Professor (tenured), Department of Computer Science, Aarhus University.
Aug 2001 – Jan 2005	Associate Professor, BRICS, Department of Computer Science, Aarhus University.

Aug 1999 – Jul 2001	Research Associate Professor, BRICS, Department of Computer Science, Aarhus University.
Aug 1998 – Jul 1999	Research Assistant Professor, BRICS PhD School, Department of Computer Science, Aarhus University.
Feb 1997 – Jul 1998	PostDoc at the Max-Planck-Institut for Computer Science, Saarbrücken, Germany.

Funding

Feb 2020 – Oct 2025	Independent Research Fund Denmark, Grant no. 9131-001113B (Algorithms Supporting Big Data Analysis, co-PI Lars Arge), 5.903.016 DKK.
Sep 2013 – Jun 2014	Aarhus University Research Foundation, Guest Researcher Grant, Seth Pettie (University of Michigan Ann Arbor), 250.000 DKK.
Jan 2011 – Dec 2012	Slovenian Research Agency, Project: Algorithms on Massive Geographical LiDAR Data-sets – AMAGELDA (Principal investigator: Andrej Brodnik, Ljubljana Slovenia), 3.000 Euro (22.500 DKK).
Jan 2008 – Dec 2009	Nordic Network on Algorithms from the Nordic Academy for Advanced Study (NORFA). Coordinator Fedor V. Fomin, University of Bergen, 600.000 NOK (510.000 DKK).
Jul 2005 – Jun 2006	The Danish Natural Science Research Council, *Graph Algorithms and Contraint Programming, PostDoc Irit Katriel*, 480.000 DKK.
Jan 2005 – Dec 2007	The Danish Natural Science Research Council, Grant no. 21-04-0389, *Algoritmer til rekonstruktion og sammenligning af træer og netværk*. Coordinator Christian N.S. Pedersen, Aarhus University, 360.000 DKK.
Jan 2005 – Dec 2007	Nordic Network on Algorithms from the Nordic Academy for Advanced Study (NORFA). Coordinator Fedor V. Fomin, University of Bergen, 900.000 NOK (825.000 DKK).
Feb 2002 – Jan 2005	Associate Professor grant from the Carlsberg Foundation, 1.350.000 DKK.
May – Jul 1998	Scholarship (PostDoc) from the Max-Planck-Institut für Informatik, Saarbrücken, Germany, 10.200 DM (39.200 DKK).
May 1997 – Apr 1998	Scholarship (PostDoc) from the Carlsberg Foundation, 300.000 DKK.
Feb – Apr 1997	Scholarship (PostDoc) from the Max-Planck-Institut für Informatik, Saarbrücken, Germany, 10.200 DM (39.200 DKK).
Feb 1995 – Jan 1997	Scholarship (PhD-stipendium) from the Danish Natural Science Research Council, 717.591 DKK.
Feb 1993 – Jan 1995	Scholarship (Scholarstipendium) from the Danish Research Academy, 156.000 DKK.

Awards

Sep 2022	Aarhus University Anniversary Foundation Teaching Price. Aarhus University.
May 2019	Lecturer of the Year. Department of Computer Science, Aarhus University.
May 2017	Lecturer of the Year. Department of Computer Science, Aarhus University.
May 2012	Lecturer of the Year. Department of Computer Science, Aarhus University.
Dec 2001	Best paper award 12th Annual International Symposium on Algorithms and Computation, for the paper "Computing the Quartet Distance Between Evolutionary Trees in Time $O(n \log^2 n)$", coauthored with Rolf Fagerberg and Christian N. S. Pedersen.

Publications

Conference publications appearing in journals and technical reports appearing elsewhere are numbered in parenthesis with the newer appearance.

Editor

1 *Scalable Data Structures (Dagstuhl Seminar 23211)*, *Dagstuhl, Germany*, *21–26 May 2023*, Gerth Stølting Brodal, John Iacono, László Kozma and Vijaya Ramachandran (Edt.), Volume 13 of Dagstuhl Reports(5), pages 114–135. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2023, doi: 10.4230/DagRep.13.5.114.

2 *Scalable Data Structures (Dagstuhl Seminar 21071)*, *Dagstuhl, Germany*, *14–19 February 2021*, Gerth Stølting Brodal, John Iacono, Markus E. Nebel and Vijaya Ramachandran (Edt.), Volume 11 of Dagstuhl Reports(1), pages 1–23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2021, doi: 10.4230/DagRep.11.1.1.

3 *Data Structures for the Cloud and External Memory Data (Dagstuhl Seminar 19051)*, *Dagstuhl, Germany*, *27 January – 1 February 2019*, Gerth Stølting Brodal, Ulrich Carsten Meyer, Markus E. Nebel and Robert Sedgewick (Edt.), Volume 9 of Dagstuhl Reports(1), pages 104–124. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2019, doi: 10.4230/DagRep.9.1.104.

4 *Algorithms - ESA 2005: 13th Annual European Symposium*, *Palma de Mallorca, Spain*, *3–6 October 2005*, Gerth Stølting Brodal and Stefano Leonardi (Edt.), Volume 3669 of Lecture Notes in Computer Science. Springer Verlag, Berlin, 2005, doi: 10.1007/11561071.

5 *Algorithm Engineering – 5th International Workshop (WAE 2001)*, *Aarhus, Denmark*, *28–31 August 2001*, Gerth Stølting Brodal, Daniele Frigioni and Alberto Marchetti-Spaccamela (Edt.), Volume 2141 of Lecture Notes in Computer Science. Springer Verlag, Berlin, 2001, doi: 10.1007/3-540-44688-5.

Book Chapters

6 Gerth Stølting Brodal, *Cache-Oblivious Sorting*. In *Encyclopedia of Algorithms*, Ming-Yang Kao (Edt.), pages 126–129. Springer Verlag, Berlin, 2008, doi: 10.1007/978-0-387-30162-4_63.

7 Lars Arge, Gerth Stølting Brodal and Rolf Fagerberg, *Cache-Oblivious Data Structures*. In *Handbook of Data Structures and Applications*, Dinesh Mehta and Sartaj Sahni (Edt.), Chapter 34, 27 pages. CRC Press, 2005, doi: 10.1201/9781420035179.ch34.

8 Gerth Stølting Brodal, *Finger Search Trees*. In *Handbook of Data Structures and Applications*, Dinesh Mehta and Sartaj Sahni (Edt.), Chapter 11, 11 pages. CRC Press, 2005, doi: 10.1201/9781420035179.ch11.

Journal Articles

9 Bruce Brewer, Gerth Stølting Brodal and Haitao Wang, *Dynamic Convex Hulls for Simple Paths*. In *Discrete & Computational Geometry*, 36 pages, 2025, doi: 10.1007/s00454-024-00715-0.

Abstract: We consider the planar dynamic convex hull problem. In the literature, solutions exist supporting the insertion and deletion of points in poly-logarithmic time and various queries on the convex hull of the current set of points in logarithmic time. If arbitrary insertion and deletion of points are allowed, constant time updates and fast queries are known to be impossible. This paper considers two restricted cases where worst-case constant time updates and logarithmic time queries are possible. We assume all updates are performed on a deque (double-ended queue) of points. The first case considers the monotonic path case, where all points are sorted in a given direction, say horizontally left-to-right, and only the leftmost and rightmost points can be inserted and deleted. The second case assumes that the points in the deque constitute a simple path. Note that the monotone case is a special case of the simple path case. For both cases, we present solutions supporting deque insertions and deletions in worst-case constant time and standard queries on the convex hull of the points in $O(\log n)$ time, where n is the number of points in the current point set. The convex hull of the current point set can be reported in $O(h + \log n)$ time, where h is the number of edges of the convex hull. For the one-sided monotone path case, where updates are only allowed on one side, the reporting time can be reduced to $O(h)$, and queries on the convex hull are supported in $O(\log h)$ time. All our time bounds are worst case. In addition, we prove lower bounds that match these time bounds, and thus our results are optimal. For a quick comparison, the previous best update bounds for the simple path problem were amortized $O(\log n)$ time by Friedman, Hershberger, and Snoeyink [SoCG 1989].

10 Gerth Stølting Brodal, George Lagogiannis and Robert E. Tarjan, *Strict Fibonacci Heaps*. In *Transactions on Algorithms*, Volume 21(2), Article no. 15, pages 1–18, January 2025, doi: 10.1145/3707692.

Abstract: We present the *strict Fibonacci heap*, the first pointer-based heap implementation with time bounds matching those of Fibonacci heaps in the worst case. Strict Fibonacci heaps support make-heap, insert, find-min, meld and decrease-key in worst-case $O(1)$ time, and delete and delete-min in worst-case $O(\lg n)$ time, where n is the size of the heap. The data structure uses linear space.

A previous solution achieving the same time bounds in the RAM model made essential use of arrays and extensive use of redundant counter schemes to maintain balance. Our solution uses neither. Our key simplification is to discard the structure of the smaller heap when doing a meld, and to use the pigeonhole principle in place of the redundant counter mechanism to maintain balance.

11 Gerth Stølting Brodal, *Priority Queues with Decreasing Keys*. In *Theoretical Computer Science*, Volume 1000, 114563 pages, 2024, doi: 10.1016/j.tcs.2024.114563.

Abstract: A priority queue stores a multiset of items, each item being a $\langle \text{key}, \text{value} \rangle$ pair, and supports the insertion of a new item and extraction of an item with minimum key. In applications like Dijkstra's single source shortest path algorithm and Prim-Jarník's minimum spanning tree algorithm, the key of an item can decrease over time. Usually this is handled by either using a priority queue supporting the deletion of an arbitrary item or a dedicated DecreaseKey operation, or by inserting the same item multiple times but with decreasing keys.

In this paper we study what happens if the keys associated with the items in a priority queue can decrease over time *without* informing the priority queue, and how such a priority queue can be used in Dijkstra's algorithm. We show that binary heaps with bottom-up insertions fail to report items with unchanged keys in correct order, while binary heaps with top-down insertions report items with unchanged keys in correct order. Furthermore, we show that skew heaps, leftist heaps, and priority queues based on linking the roots of heap-ordered trees, like pairing heaps, binomial queues and Fibonacci heaps, work correctly with decreasing keys without any modifications. Finally, we show that the post-order heap by Harvey and Zatloukal, a variant of a binary heap with amortized constant time insertions and amortized logarithmic time deletions, works correctly with decreasing keys and is a strong contender for an implicit priority queue supporting decreasing keys in practice.

12 Gerth Stølting Brodal and Konstantinos Mampentzidis, *Cache Oblivious Algorithms for Computing the Triplet Distance Between Trees*. In *ACM Journal of Experimental Algorithmics*, Volume 26, Article no. 1.2, pages 1–44, April 2021, doi: 10.1145/3433651.

Abstract: We consider the problem of computing the triplet distance between two rooted unordered trees with n labeled leaves. Introduced by Dobson in 1975, the triplet distance is the number of leaf triples that induce different topologies in the two trees. The current theoretically fastest algorithm is an $O(n \log n)$ algorithm by Brodal *et al.* (SODA 2013). Recently Jansson and Rajaby proposed a new algorithm that, while slower in theory, requiring $O(n \log^3 n)$ time, in practice it outperforms the theoretically faster $O(n \log n)$ algorithm. Both algorithms do not scale to external memory.

We present two cache oblivious algorithms that combine the best of both worlds. The first algorithm is for the case when the two input trees are binary trees, and the second is a generalized algorithm for two input trees of arbitrary degree. Analyzed in the RAM model, both algorithms require $O(n \log n)$ time, and in the cache oblivious model $O(\frac{n}{B} \log_2 \frac{n}{M})$ I/Os. Their relative simplicity and the fact that they scale to external memory makes them achieve the best practical performance. We note that these are the first algorithms that scale to external memory, both in theory and in practice, for this problem.

13 Gerth Stølting Brodal, *In Memoriam Lars Arge*. In *Bulletin of the EATCS*, Volume 133, pages 11–14, February 2021.

14 Gerth Stølting Brodal, Spyros Sioutas, Konstantinos Tsakalidis and Kostas Tsichlas, *Fully persistent B-trees*. In *Theoretical Computer Science*, Volume 841, pages 10–26, 2020, doi: 10.1016/j.

tcs.2020.06.027.

Abstract: We present efficient fully persistent B-trees in the I/O model with block size B that support range searches on t reported elements at any accessed version of size n in $O(\log_B nn+t/B)$ I/Os and updates at any accessed version in $O(\log_B n + \log_2 B)$ amortized I/Os, using $O(m/B)$ disk blocks after m updates. This improves both the query and update I/O-efficiency of the previous fully persistent B-trees of Lanka and Mays (ACM SIGMOD ICMD 1991). To achieve the result, we introduce an implementation for ephemeral B-trees that supports searches and updates in $O(\log_B n)$ I/Os, using $O(n/B)$ blocks, where moreover every update makes a worst-case constant number of modifications to the structure. We make these B-trees fully persistent using an I/O-efficient method for full persistence, inspired by the node-splitting method of Driscoll et al. (JCSS 1989). Interesting in its own right, the method is generic enough to be applied to any external memory pointer-based data structure with maximum in-degree din and out-degree $O(B)$, where every node occupies a constant number of blocks on disk. For a user-specified parameter $\pi = \Omega(d_{in})$, we achieve $O(\frac{\pi}{B} + \log_2 \pi)$ I/O-overhead per access to a field of an ephemeral block and amortized $O(\frac{\pi}{B} + \log_2 \pi + \frac{d_{in}}{\pi} \log_2 B)$ I/O-overhead and $O(1/B)$ block space-overhead per modification to the ephemeral structure.

15 Edvin Berglin and Gerth Stølting Brodal, *A Simple Greedy Algorithm for Dynamic Graph Orientation*. In *Algorithmica*, Volume 82, pages 245–259, 2020, doi: 10.1007/s00453-018-0528-0.

Abstract: *Graph orientations with low out-degree* are one of several ways to efficiently store sparse graphs. If the graphs allow for insertion and deletion of edges, one may have to *flip* the orientation of some edges to prevent blowing up the maximum out-degree. We use arboricity as our sparsity measure. With an immensely simple greedy algorithm, we get parametrized trade-off bounds between out-degree and worst case number of flips, which previously only existed for amortized number of flips. We match the previous best worst-case algorithm (in $O(\log n)$ flips) for almost all values of arboricity and beat it for either constant or super-logarithmic arboricity. We also match a previous best amortized result for at least logarithmic arboricity, and give the first results with worst-case $O(1)$ and $O(\sqrt{\log n})$ flips nearly matching out-degree bounds to their respective amortized solutions.

16 Gerth Stølting Brodal, Pooya Davoodi, Moshe Lewenstein, Rajeev Raman and S. Srinivasa Rao, *Two Dimensional Range Minimum Queries and Fibonacci Lattices*. In *Theoretical Computer Science*, Volume 638, pages 33–43, 2016, doi: 10.1016/j.tcs.2016.02.016.

Abstract: Given a matrix of size N, two dimensional range minimum queries (2D-RMQs) ask for the position of the minimum element in a rectangular range within the matrix. We study trade-offs between the query time and the additional space used by indexing data structures that support 2D-RMQs. Using a novel technique–the discrepancy properties of Fibonacci lattices–we give an indexing data structure for 2D-RMQs that uses $O(N/c)$ bits additional space with $O(c \log c(\log \log c)^2)$ query time, for any parameter c, $4 \leq c \leq N$. Also, when the entries of the input matrix are from $\{0, 1\}$, we show that the query time can be improved to $O(c \log c)$ with the same space usage.

17 Gerth Stølting Brodal, Spyros Sioutas, Kostas Tsichlas and Christos D. Zaroliagis, *D^2-Tree: A New Overlay with Deterministic Bounds*. In *Algorithmica*, Volume 72(3), pages 860–883, 2015, doi: 10.1007/s00453-014-9878-4.

Abstract: We present a new overlay, called the *Deterministic Decentralized tree* (D^2-tree). The D^2-tree compares favorably to other overlays for the following reasons: (a) it provides matching and better complexities, which are deterministic for the supported operations; (b) the management of nodes (peers) and elements are completely decoupled from each other; and (c) an efficient deterministic load-balancing mechanism is presented for the uniform distribution of elements into nodes, while at the same time probabilistic optimal bounds are provided for the congestion of operations at the nodes. The load-balancing scheme of elements into nodes is deterministic and general enough to be applied to other hierarchical tree-based overlays. This load-balancing mechanism is based on an innovative lazy weight-balancing mechanism, which is interesting in its own right.

18 Gerth Stølting Brodal, Gabriel Moruz and Andrei Negoescu, *OnlineMin: A Fast Strongly Competitive Randomized Paging Algorithm*. In *Theory of Computing Systems, Special issue of the 9th Workshop on Approximation and Online Algorithms*, Volume 56(1), pages 22–40, 2015, doi: 10.1007/s00224-012-9427-y.

Abstract: In the field of online algorithms paging is one of the most studied problems. For randomized paging algorithms a tight bound of H_k on the competitive ratio has been known for decades, yet existing algorithms matching this bound have high running times. We present a new randomized paging algorithm OnlineMin that has optimal competitiveness and allows fast implementations. In fact, if k pages fit in internal memory the best previous solution required $O(k^2)$ time per request and $O(k)$ space. We present two implementations of OnlineMin which use $O(k)$ space, but only $O(\log k)$ worst case time and $O(\log k/\log\log k)$ worst case time per page request respectively.

19 Andreas Sand, Morten Kragelund Holt, Jens Johansen, Gerth Stølting Brodal, Thomas Mailund and Christian Nørgaard Storm Pedersen, *tqDist: A Library for Computing the Quartet and Triplet Distances Between Binary or General Trees*. In *Bioinformatics*, Volume 30(14), pages 2079–2080, 2014, doi: 10.1093/bioinformatics/btu157.

Abstract: Summary: tqDist is a software package for computing the triplet and quartet distances between general rooted or unrooted trees, respectively. The program is based on algorithms with running time $O(n \log n)$ for the triplet distance calculation and $O(d{\cdot}n \log n)$ for the quartet distance calculation, where n is the number of leaves in the trees and d is the degree of the tree with minimum degree. These are currently the fastest algorithms both in theory and in practice.

Availability and implementation: tqDist can be installed on Windows, Linux and Mac OS X. Doing this will install a set of command-line tools together with a Python module and an R package for scripting in Python or R. The software package is freely available under the GNU LGPL licence at http://birc.au.dk/software/tqDist.

20 Gerth Stølting Brodal, Alexis Kaporis, Apostolos Papadopoulos, Spyros Sioutas, Konstantinos Tsakalidis and Kostas Tsichlas, *Dynamic 3-sided Planar Range Queries with Expected Doubly Logarithmic Time*. In *Theoretical Computer Science*, Volume 526, pages 58–74. Elsevier Science, 2014, doi: 10.1016/j.tcs.2014.01.014.

Abstract: The Priority Search Tree is the classic solution for the problem of dynamic 2-dimensional searching for the orthogonal query range of the form $[a, b] \times (-\infty, c]$ (3-sided rectangle). It supports all operations in logarithmic worst case complexity in both main and external memory. In this work we show that the update and query complexities can we improved to expected doubly-logarithmic, when the input coordinates are being continuously drawn from specific probability distributions. We present three pairs of linear space solutions for the problem, i.e. a RAM and a corresponding I/O model variant:

(1) First, we improve the update complexity to doubly-logarithmic expected with high probability, under the most general assumption that both the x- and y-coordinates of the input points are continuously being drawn from a distribution whose density function is unknown but fixed.

(2) Next, we improve both the query complexity to doubly-logarithmic expected with high probability and the update complexity to doubly-logarithmic amortized expected, by assuming that only the x-coordinates are being drawn from a class of *smooth* distributions, and that the deleted points are selected uniformly at random among the currently stored points. In fact, the y-coordinates are allowed to be arbitrarily distributed.

(3) Finally, we improve both the query and the update complexity to doubly-logarithmic expected with high probability by moreover assuming the y-coordinates to be continuously drawn from a more restricted class of realistic distributions.

All data structures are deterministic and their complexity's expectation is with respect to the assumed distributions. They comprise combinations of known data structures and of two new data structures introduced here, namely the *Weight Balanced Exponential Tree* and the *External Modified Priority Search Tree*.

21 Gerth Stølting Brodal, Mark Greve, Vineet Pandey and S. Srinivasa Rao, *Integer Representations*

towards Efficient Counting in the Bit Probe Model. In *Journal of Discrete Algorithms*, Volume 26, pages 34–44, 2014, doi: 10.1016/j.jda.2013.11.001.

Abstract: We consider the problem of representing integers in close to optimal number of bits to support increment and decrement operations efficiently. We study the problem in the bit probe model and analyse the number of bits read and written to perform the operations, both in the worst-case and in the average-case. We propose representations, called *counters*, with different trade-offs between the space used and the number of bits probed. A counter is *space-optimal* if it represents any integer in the range $[0, \ldots, 2^n - 1]$ using exactly n bits. We provide a *space-optimal counter* which supports increment and decrement operations by reading at most $n-1$ bits and writing at most 3 bits in the worst-case. This is the first space-optimal representation which supports these operations by always reading strictly less than n bits. For *redundant counters* where we only need to represent integers in the range $[0, \ldots, L-1]$ for some integer $L < 2^n$ using n bits, we define the *space-efficiency* of the counter as the ratio $L/2^n$. We provide representations that achieve different trade-offs between the read/write-complexity and the efficiency.

We also examine the problem of representing integers to support addition and subtraction operations. We propose a representation of integers using n bits and with space efficiency at least $1/n$, which supports addition and subtraction operations, improving the efficiency of an earlier representation by Munro and Rahman [Algorithmica, 2010]. We also show various trade-offs between the operation times and the space complexity.

22 Andreas Sand, Morten Kragelund Holt, Jens Johansen, Rolf Fagerberg, Gerth Stølting Brodal, Christian Nørgaard Storm Pedersen and Thomas Mailund, *Algorithms for Computing the Triplet and Quartet Distances for Binary and General Trees*. In *Biology - Special Issue on Developments in Bioinformatic Algorithms*, Volume 2(4), pages 1189–1209, 2013, doi: 10.3390/biology2041189.

Abstract: Distance measures between trees are useful for comparing trees in a systematic manner, and several different distance measures have been proposed. The triplet and quartet distances, for rooted and unrooted trees, respectively, are defined as the number of subsets of three or four leaves, respectively, where the topologies of the induced subtrees differ. These distances can trivially be computed by explicitly enumerating all sets of three or four leaves and testing if the topologies are different, but this leads to time complexities at least of the order n^3 or n^4 just for enumerating the sets. The different topologies can be counte dimplicitly, however, and in this paper, we review a series of algorithmic improvements that have been used during the last decade to develop more efficient algorithms by exploiting two different strategies for this; one based on dynamic programming and another based oncoloring leaves in one tree and updating a hierarchical decomposition of the other.

23 Andreas Sand, Gerth Stølting Brodal, Rolf Fagerberg, Christian Nørgaard Storm Pedersen and Thomas Mailund, *A practical $O(n \log^2 n)$ time algorithm for computing the triplet distance on binary trees*. In *BMC Bioinformatics*, Volume 14(Suppl 2), S18 pages, 2013, doi: 10.1186/1471-2105-14-S2-S18.

Abstract: The triplet distance is a distance measure that compares two rooted trees on the same set of leaves by enumerating all sub-sets of three leaves and counting how often the induced topologies of the tree are equal or different. We present an algorithm that computes the triplet distance between two rooted binary trees in time $O\left(n \log^2 n\right)$. The algorithm is related to an algorithm for computing the quartet distance between two unrooted binary trees in time $O\left(n \log n\right)$. While the quartet distance algorithm has a very severe overhead in the asymptotic time complexity that makes it impractical compared to $O\left(n^2\right)$ time algorithms, we show through experiments that the triplet distance algorithm can be implemented to give a competitive wall-time running time.

24 Lars Arge, Gerth Stølting Brodal and S. Srinivasa Rao, *External Memory Planar Point Location with Logarithmic Updates*. In *Algorithmica*, Volume 63(1), pages 457–475, 2012, doi: 10.1007/s00453-011-9541-2.

Abstract: Point location is an extremely well-studied problem both in internal memory models and recently also in the external memory model. In this paper, we present an I/O-efficient dynamic data structure for point location in general planar subdivisions. Our structure uses linear space

to store a subdivision with N segments. Insertions and deletions of segments can be performed in amortized $O(\log_B N)$ I/Os and queries can be answered in $O(\log_B^2 N)$ I/Os in the worst-case. The previous best known linear space dynamic structure also answers queries in $O(\log_B^2 N)$ I/Os, but only supports insertions in amortized $O(\log_B^2 N)$ I/Os. Our structure is also considerably simpler than previous structures.

25 Gerth Stølting Brodal, Pooya Davoodi and S. Srinivasa Rao, *On Space Efficient Two Dimensional Range Minimum Data Structures*. In *Algorithmica, Special issue on ESA 2010*, Volume 63(4), pages 815–830, 2012, doi: 10.1007/s00453-011-9499-0.

Abstract: The two dimensional range minimum query problem is to preprocess a static m by n matrix (two dimensional array) A of size $N = m \cdot n$, such that subsequent queries, asking for the position of the minimum element in a rectangular range within A, can be answered efficiently. We study the trade-off between the space and query time of the problem. We show that every algorithm enabled to access A during the query and using a data structure of size $O(N/c)$ bits requires $\Omega(c)$ query time, for any c where $1 \leq c \leq N$. This lower bound holds for arrays of any dimension. In particular, for the one dimensional version of the problem, the lower bound is tight up to a constant factor. In two dimensions, we complement the lower bound with an indexing data structure of size $O(N/c)$ bits which can be preprocessed in $O(N)$ time to support $O(c \log^2 c)$ query time. For $c = O(1)$, this is the first $O(1)$ query time algorithm using a data structure of optimal size $O(N)$ bits. For the case where queries can not probe A, we give a data structure of size $O(N \cdot \min\{m, \log n\})$ bits with $O(1)$ query time, assuming $m \leq n$. This leaves a gap to the space lower bound of $\Omega(N \log m)$ bits for this version of the problem.

26 Gerth Stølting Brodal, Beat Gfeller, Allan Grønlund Jørgensen and Peter Sanders, *Towards Optimal Range Median*. In *Theoretical Computer Science, Special issue of ICALP'09*, Volume 412(24), pages 2588–2601. Elsevier Science, 2011, doi: 10.1016/j.tcs.2010.05.003.

Abstract: We consider the following problem: Given an unsorted array of n elements, and a sequence of intervals in the array, compute the median in each of the subarrays defined by the intervals. We describe a simple algorithm which needs $O(n \log k + k \log n)$ time to answer k such median queries. This improves previous algorithms by a logarithmic factor and matches a comparison lower bound for $k = O(n)$. The space complexity of our simple algorithm is $O(n \log n)$ in the pointer-machine model, and $O(n)$ in the RAM model. In the latter model, a more involved $O(n)$ space data structure can be constructed in $O(n \log n)$ time where the time per query is reduced to $O(\log n / \log \log n)$. We also give efficient dynamic variants of both data structures, achieving $O(\log^2 n)$ query time using $O(n \log n)$ space in the comparison model and $O((\log n / \log \log n)^2)$ query time using $O(n \log n / \log \log n)$ space in the RAM model, and show that in the cell-probe model, any data structure which supports updates in $O(\log^{O(1)} n)$ time must have $\Omega(\log n / \log \log n)$ query time.

Our approach naturally generalizes to higher-dimensional range median problems, where element positions and query ranges are multidimensional — it reduces a range median query to a logarithmic number of range counting queries.

27 Martin Kutz, Gerth Stølting Brodal, Kanela Kaligosi and Irit Katriel, *Faster Algorithms for Computing Longest Common Increasing Subsequences*. In *Journal of Discrete Algorithms, Special Issue of CPM 2006*, Volume 9(4), pages 314–325. Elsevier Science, 2011, doi: 10.1016/j.jda.2011.03.013.

Abstract: We present algorithms for finding a longest common increasing subsequence of two or more input sequences. For two sequences of lengths n and m, where $m \geq n$, we present an algorithm with an output-dependent expected running time of $O((m + n\ell) \log \log \sigma + \mathit{Sort})$ and $O(m)$ space, where ℓ is the length of an LCIS, σ is the size of the alphabet, and *Sort* is the time to sort each input sequence. For $k \geq 3$ length-n sequences we present an algorithm which improves the previous best bound by more than a factor k for many inputs. In both cases, our algorithms are conceptually quite simple but rely on existing sophisticated data structures. Finally, we introduce the problem of longest common weakly-increasing (or non-decreasing) subsequences (LCWIS), for which we present an $O(\min\{m + n \log n, m \log \log m\})$-time algorithm for the 3-letter alphabet case.

For the extensively studied longest common subsequence problem, comparable speedups have not been achieved for small alphabets.

28 Michael A. Bender, Gerth Stølting Brodal, Rolf Fagerberg, Dongdong Ge, Simai He, Haodong Hu, John Iacono and Alejandro López-Ortiz, *The Cost of Cache-Oblivious Searching*. In *Algorithmica*, Volume 61(2), pages 463–505, 2011, doi: 10.1007/s00453-010-9394-0.

Abstract: This paper gives tight bounds on the cost of cache-oblivious searching. The paper shows that no cache-oblivious search structure can guarantee a search performance of fewer than $\lg e \log_B N$ memory transfers between any two levels of the memory hierarchy. This lower bound holds even if all of the block sizes are limited to be powers of 2. The paper gives modified versions of the van Emde Boas layout, where the expected number of memory transfers between any two levels of the memory hierarchy is arbitrarily close to $[\lg e + O(\lg \lg B / \lg B)] \log_B N + O(1)$. This factor approaches $\lg e \approx 1.443$ as B increases. The expectation is taken over the random placement in memory of the first element of the structure.

Because searching in the disk-access machine (DAM) model can be performed in $\log_B N + O(1)$ block transfers, this result establishes a separation between the (2-level) DAM model and cache-oblivious model. The DAM model naturally extends to k levels. The paper also shows that as k grows, the search costs of the optimal k-level DAM search structure and the optimal cache-oblivious search structure rapidly converge. This result demonstrates that for a multilevel memory hierarchy, a simple cache-oblivious structure almost replicates the performance of an optimal parameterized k-level DAM structure.

29 Michael A. Bender, Gerth Stølting Brodal, Rolf Fagerberg, Riko Jacob and Elias Vicari, *Optimal Sparse Matrix Dense Vector Multiplication in the I/O-Model*. In *Theory of Computing Systems, Special issue of SPAA'07*, Volume 47(4), pages 934–962. Springer Verlag, Berlin, 2010, doi: 10.1007/s00224-010-9285-4.

Abstract: We study the problem of sparse-matrix dense-vector multiplication (SpMV) in external memory. The task of SpMV is to compute $y := Ax$, where A is a sparse $N \times N$ matrix and x is a vector. We express sparsity by a parameter k, and for each choice of k consider the class of matrices where the number of nonzero entries is kN, i.e., where the average number of nonzero entries per column is k.

We investigate what is the external worst-case complexity, i.e., the best possible upper bound on the number of I/Os, as a function of k and N. We determine this complexity up to a constant factor for all meaningful choices of these parameters. Our model of computation for the lower bound is a combination of the I/O-models of Aggarwal and Vitter, and of Hong and Kung.

We study variants of the problem, differing in the memory layout of A. If A is stored in column major layout, we prove that SpMV has I/O complexity $\Theta(\min\{kN/B \cdot \max\{1, \log_{M/B}(N/\max\{k, M\})\}, kN\})$ for $k \leq N^{1-\varepsilon}$ and any constant $0 < \varepsilon < 1$. If the algorithm can choose the memory layout, the I/O complexity reduces to $\Theta(\min\{kN/B \cdot \max\{1, \log_{M/B}(N/(kM))\}, kN\})$ for $k \leq N^{1/3}$. In contrast, if the algorithm must be able to handle an arbitrary layout of the matrix, the I/O complexity is $\Theta(\min\{kN/B \cdot \max\{1, \log_{M/B}(N/M)\}, kN\})$ for $k \leq N/2$.

In the cache oblivious setting we prove that with tall cache assumption $M \geq B^{1+\varepsilon}$, the I/O complexity is $O(kN/B \cdot \max\{1, \log_{M/B}(N/\max\{k, M\})\})$ for A in column major layout.

30 Martin Stissing, Thomas Mailund, Christian Nørgaard Storm Pedersen, Gerth Stølting Brodal and Rolf Fagerberg, *Computing the All-Pairs Quartet Distance on a set of Evolutionary Trees*. In *Journal of Bioinformatics and Computational Biology*, Volume 6(1), pages 37–50, 2008, doi: 10.1142/S0219720008003266.

Abstract: We present two algorithms for calculating the quartet distance between all pairs of trees in a set of binary evolutionary trees on a common set of species. The algorithms exploit common substructure among the trees to speed up the pairwise distance calculations, thus performing significantly better on large sets of trees compared to performing distinct pairwise distance calculations, as we illustrate experimentally, where we see a speedup factor of around 130 in the best case.

31 Gerth Stølting Brodal, Rolf Fagerberg and Gabriel Moruz, *On the Adaptiveness of Quicksort*. In *ACM Journal of Experimental Algorithmics, Special Issue of 7th Workshop on Algorithm Engineering and Experiments*, Volume 12, Article no. 3.2, 19 pages, 2008, doi: 10.1145/1227161.1402294.

Abstract: Quicksort was first introduced in 1961 by Hoare. Many variants have been developed, the best of which are among the fastest generic sorting algorithms available, as testified by the choice of Quicksort as the default sorting algorithm in most programming libraries. Some sorting algorithms are adaptive, i.e. they have a complexity analysis that is better for inputs which are nearly sorted, according to some specified measure of presortedness. Quicksort is not among these, as it uses $\Omega(n \log n)$ comparisons even for sorted inputs. However, in this paper we demonstrate empirically that the actual running time of Quicksort *is* adaptive with respect to the presortedness measure Inv. Differences close to a factor of two are observed between instances with low and high Inv value. We then show that for the randomized version of Quicksort, the number of element *swaps* performed is *provably* adaptive with respect to the measure Inv. More precisely, we prove that randomized Quicksort performs expected $O(n(1 + \log(1 + \text{Inv}/n)))$ element swaps, where Inv denotes the number of inversions in the input sequence. This result provides a theoretical explanation for the observed behavior, and gives new insights on the behavior of Quicksort. We also give some empirical results on the adaptive behavior of Heapsort and Mergesort.

32 Gerth Stølting Brodal, Loukas Georgiadis and Irit Katriel, *An $O(n \log n)$ Version of the Averbakh-Berman Algorithm for the Robust Median of a Tree*. In *Operations Research Letters*, Volume 36(1), pages 14–18, 2008, doi: 10.1016/j.orl.2007.02.012.

Abstract: We show that the minmax regret median of a tree can be found in $O(n \log n)$ time. This is obtained by a modification of Averbakh and Berman's $O(n \log^2 n)$-time algorithm: We design a dynamic solution to their bottleneck subproblem of finding the middle of every root-leaf path in a tree.

33 Gerth Stølting Brodal, Rolf Fagerberg and Kristoffer Vinther, *Engineering a Cache-Oblivious Sorting Algorithm*. In *ACM Journal of Experimental Algorithmics, Special Issue of 6th Workshop on Algorithm Engineering and Experiments*, Volume 12, Article no. 2.2, 23 pages, 2007, doi: 10.1145/1227161.1227164.

Abstract: This paper is an algorithmic engineering study of cache-oblivious sorting. We investigate by empirical methods a number of implementation issues and parameter choices for the cache-oblivious sorting algorithm Lazy Funnelsort, and compare the final algorithm with Quicksort, the established standard for comparison-based sorting, as well as with recent cache-aware proposals.

The main result is a carefully implemented cache-oblivious sorting algorithm, which our experiments show can be faster than the best Quicksort implementation we are able to find, already for input sizes well within the limits of RAM. It is also at least as fast as the recent cache-aware implementations included in the test. On disk the difference is even more pronounced regarding Quicksort and the cache-aware algorithms, whereas the algorithm is slower than a careful implementation of multiway Mergesort such as TPIE.

Source code available at: Engineering Cache-Oblivious Sorting Algorithms, Kristoffer Vinther. Master's Thesis, Department of Computer Science, Aarhus University, June 2003.

34 Thomas Mailund, Gerth Stølting Brodal, Rolf Fagerberg, Christian Nørgaard Storm Pedersen and Derek Phillips, *Recrafting the Neighbor-Joining Method*. In *BMC Bioinformatics*, Volume 7(29), 2006, doi: 10.1186/1471-2105-7-29.

Abstract: **Background:** The neighbor-joining method by Saitou and Nei is a widely used method for constructing phylogenetic trees. The formulation of the method gives rise to a canonical $\Theta(n^3)$ algorithm upon which all existing implementations are based.

Results: In this paper we present techniques for speeding up the canonical neighbor-joining method. Our algorithms construct the same phylogenetic trees as the canonical neighbor-joining method. The best-case running time of our algorithms are $O(n^2)$ but the worst-case remains $O(n^3)$. We empirically evaluate the performance of our algoritms on distance matrices obtained from the

Pfam collection of alignments. The experiments indicate that the running time of our algorithms evolve as $\Theta(n^2)$ on the examined instance collection. We also compare the running time with that of the QuickTree tool, a widely used efficient implementation of the canonical neighbor-joining method.

Conclusions: The experiments show that our algorithms also yield a significant speed-up, already for medium sized instances.

BioMed Central Open Access

35 Gerth Stølting Brodal, Erik D. Demaine and J. Ian Munro, *Fast Allocation and Deallocation with an Improved Buddy System*. In *Acta Informatica*, Volume 41(4-5), pages 273–291, 2005, doi: 10.1007/s00236-004-0159-6.

Abstract: We propose several modifications to the binary buddy system for managing dynamic allocation of memory blocks whose sizes are powers of two. The standard buddy system allocates and deallocates blocks in $\Theta(\lg n)$ time in the worst case (and on an amortized basis), where n is the size of the memory. We present three schemes that improve the running time to $O(1)$ time, where the time bound for deallocation is amortized for the first two schemes. The first scheme uses just one more word of memory than the standard buddy system, but may result in greater fragmentation than necessary. The second and third schemes have essentially the same fragmentation as the standard buddy system, and use $O(2^{(1+\sqrt{\lg n})\lg\lg n})$ bits of auxiliary storage, which is $\omega(\lg^k n)$ but $o(n^\varepsilon)$ for all $k \geq 1$ and $\varepsilon > 0$. Finally, we present simulation results estimating the effect of the excess fragmentation in the first scheme.

36 Lars Arge, Gerth Stølting Brodal and Laura Toma, *On External-Memory MST, SSSP and Multi-way Planar Graph Separation*. In *Journal of Algorithms*, Volume 53(2), pages 186–206, 2004, doi: 10.1016/j.jalgor.2004.04.001.

Abstract: Recently external memory graph problems have received considerable attention because massive graphs arise naturally in many applications involving massive data sets. Even though a large number of I/O-efficient graph algorithms have been developed, a number of fundamental problems still remain open.

The results in this paper fall in two main classes: First we develop an improved algorithm for the problem of computing a minimum spanning tree (MST) of a general undirected graph. Second we show that on planar undirected graphs the problems of computing a multi-way graph separation and single source shortest paths (SSSP) can be reduced I/O-efficiently to planar breadth-first search (BFS). Since BFS can be trivially reduced to SSSP by assigning all edges weight one, it follows that in external memory planar BFS, SSSP, and multi-way separation are equivalent. That is, if any of these problems can be solved I/O-efficiently, then all of them can be solved I/O-efficiently in the same bound. Our planar graph results have subsequently been used to obtain I/O-efficient algorithms for all fundamental problems on planar undirected graphs.

37 Gerth Stølting Brodal, Rolf Fagerberg and Christian Nørgaard Storm Pedersen, *Computing the Quartet Distance Between Evolutionary Trees in Time $O(n \log n)$*. In *Algorithmica, Special issue on ISAAC 2001*, Volume 38(2), pages 377–395, 2004, doi: 10.1007/s00453-003-1065-y.

Abstract: Evolutionary trees describing the relationship for a set of species are central in evolutionary biology, and quantifying differences between evolutionary trees is therefore an important task. The quartet distance is a distance measure between trees previously proposed by Estabrook, McMorris and Meacham. The quartet distance between two unrooted evolutionary trees is the number of quartet topology differences between the two trees, where a quartet topology is the topological subtree induced by four species. In this paper, we present an algorithm for computing the quartet distance between two unrooted evolutionary trees of n species, where all internal nodes have degree three, in time $O(n \log n)$. The previous best algorithm for the problem uses time $O(n^2)$.

38 Gerth Stølting Brodal, George Lagogiannis, Christos Makris, Athanasios Tsakalidis and Kostas Tsichlas, *Optimal Finger Search Trees in the Pointer Machine*. In *Journal of Computer and System Sciences, Special issue on 34th Annual ACM Symposium on Theory of Computing*, Volume 67(2), pages 381–418, 2003, doi: 10.1016/S0022-0000(03)00013-8.

Abstract: We develop a new finger search tree with worst case constant update time in the Pointer Machine (PM) model of computation. This was a major problem in the field of Data Structures and was tantalizingly open for over twenty years, while many attempts by researchers were made to solve it. The result comes as a consequence of the innovative mechanism that guides the rebalancing operations, combined with incremental multiple splitting and fusion techniques over nodes.

39 Gerth Stølting Brodal, Christos Makris, Spyros Sioutas, Athanasios Tsakalidis and Kostas Tsichlas, *Optimal Solutions for the Temporal Precedence Problem*. In *Algorithmica*, Volume 33(4), pages 494–510, 2002, doi: 10.1007/s00453-002-0935-z.

Abstract: In this paper we refer to the *Temporal Precedence Problem* on *Pure Pointer Machines*. This problem asks for the design of a data structure, maintaining a set of stored elements and supporting the following two operations: *insert* and *precedes*. The operation $insert(a)$ introduces a new element a in the structure, while the operation $precedes(a, b)$ returns true iff element a was inserted before element b temporally. In Ranjan et al. a solution was provided to the problem with worst-case time complexity $O(\log \log n)$ per operation and $O(n \log \log n)$ space, where n is the number of elements inserted. It was also demonstrated that the *precedes* operation has a lower bound of $\Omega(\log \log n)$ for the *Pure Pointer Machine* model of computation. In this paper we present two simple solutions with linear space and worst-case constant insertion time. In addition, we describe two algorithms that can handle the $precedes(a, b)$ operation in $O(\log \log d)$ time, where d is the temporal distance between the elements a and b.

40 Gerth Stølting Brodal and M. Cristina Pinotti, *Comparator Networks for Binary Heap Construction*. In *Theoretical Computer Science*, Volume 250(1-2), pages 235–245, 2001, doi: 10.1016/S0304-3975(99)00137-1.

Abstract: Comparator networks for constructing binary heaps of size n are presented which have size $O(n \log \log n)$ and depth $O(\log n)$. A lower bound of $n \log \log n - O(n)$ for the size of any heap construction network is also proven, implying that the networks presented are within a constant factor of optimal. We give a tight relation between the leading constants in the size of selection networks and in the size of heap construction networks.

41 Gerth Stølting Brodal and Venkatesh Srinivasan, *Improved Bounds for Dictionary Look-up with One Error*. In *Information Processing Letters*, Volume 75(1-2), pages 57–59, 2000, doi: 10.1016/S0020-0190(00)00079-X.

Abstract: Given a dictionary S of n binary strings each of length m, we consider the problem of designing a data structure for S that supports *d-queries*; given a binary query string q of length m, a d-query reports if there exists a string in S within Hamming distance d of q. We construct a data structure for the case $d = 1$, that requires space $O(n \log m)$ and has query time $O(1)$ in a cell probe model with word size m. This generalizes and improves the previous bounds of Yao and Yao for the problem in the bit probe model. The data structure can be constructed in randomized expected time $O(nm)$.

42 Gerth Stølting Brodal, Rune Bang Lyngsø, Christian Nørgaard Storm Pedersen and Jens Stoye, *Finding Maximal Pairs with Bounded Gap*. In *Journal of Discrete Algorithms, Special Issue of Matching Patterns*, Volume 1(1), pages 77–104. Hermes Science Publishing Ltd, Oxford 2000, 2000.

Abstract: A pair in a string is the occurrence of the same substring twice. A pair is maximal if the two occurrences of the substring cannot be extended to the left and right without making them different, and the gap of a pair is the number of characters between the two occurrences of the substring. In this paper we present methods for finding all maximal pairs under various constraints on the gap. In a string of length n we can find all maximal pairs with gap in an upper and lower bounded interval in time $O(n \log n + z)$, where z is the number of reported pairs. If the upper bound is removed the time reduces to $O(n + z)$. Since a tandem repeat is a pair with gap zero, our methods is a generalization of finding tandem repeats. The running time of our methods also equals the running time of well known methods for finding tandem repeats.

43 Gerth Stølting Brodal, *Priority Queues on Parallel Machines*. In *Parallel Computing*, Volume

25(8), pages 987–1011, 1999, doi: 10.1016/S0167-8191(99)00032-0.

Abstract: We present time and work optimal priority queues for the CREW PRAM, supporting FindMin in constant time with one processor and MakeQueue, Insert, Meld, FindMin, ExtractMin, Delete and DecreaseKey in constant time with $O(\log n)$ processors. A priority queue can be build in time $O(\log n)$ with $O(n/\log n)$ processors. A pipelined version of the priority queues adopt to a processor array of size $O(\log n)$, supporting the operations MakeQueue, Insert, Meld, FindMin, ExtractMin, Delete and DecreaseKey in constant time. By applying the k-bandwidth technique we get a data structure for the CREW PRAM which supports MultiInsert$_k$ operations in $O(\log k)$ time and MultiExtractMin$_k$ in $O(\log \log k)$ time.

44 Gerth Stølting Brodal, Jesper Larsson Träff and Christos D. Zaroliagis, *A Parallel Priority Queue with Constant Time Operations*. In *Journal of Parallel and Distributed Computing, Special Issue on Parallel Data Structures*, Volume 49(1), pages 4–21, 1998, doi: 10.1006/jpdc.1998.1425.

Abstract: We present a parallel priority queue that supports the following operations in constant time: *parallel insertion* of a sequence of elements ordered according to key, *parallel decrease key* for a sequence of elements ordered according to key, *deletion of the minimum key element*, as well as *deletion of an arbitrary element*. Our data structure is the first to support multi insertion and multi decrease key in constant time. The priority queue can be implemented on the EREW PRAM, and can perform any sequence of n operations in $O(n)$ time and $O(m \log n)$ work, m being the total number of keys inserted and/or updated. A main application is a parallel implementation of Dijkstra's algorithm for the single-source shortest path problem, which runs in $O(n)$ time and $O(m \log n)$ work on a CREW PRAM on graphs with n vertices and m edges. This is a logarithmic factor improvement in the running time compared with previous approaches.

45 Gerth Stølting Brodal, Shiva Chaudhuri and Jaikumar Radhakrishnan, *The Randomized Complexity of Maintaining the Minimum*. In *Nordic Journal of Computing, Selected Papers of the 5th Scandinavian Workshop on Algorithm Theory (SWAT'96)*, Volume 3(4), pages 337–351, 1996.

Abstract: The complexity of maintaining a set under the operations Insert, Delete and FindMin is considered. In the comparison model it is shown that any randomized algorithm with expected amortized cost t comparisons per Insert and Delete has expected cost at least $n/(e2^{2t}) - 1$ comparisons for FindMin. If FindMin is replaced by a weaker operation, FindAny, then it is shown that a randomized algorithm with constant expected cost per operation exists; in contrast, it is shown that no deterministic algorithm can have constant cost per operation. Finally, a deterministic algorithm with constant amortized cost per operation for an offline version of the problem is given.

46 Gerth Stølting Brodal, *Partially Persistent Data Structures of Bounded Degree with Constant Update Time*. In *Nordic Journal of Computing*, Volume 3(3), pages 238–255, 1996.

Abstract: The problem of making bounded in-degree and out-degree data structures partially persistent is considered. The node copying method of Driscoll *et al.* is extended so that updates can be performed in *worst-case* constant time on the pointer machine model. Previously it was only known to be possible in amortised constant time.

The result is presented in terms of a new strategy for Dietz and Raman's dynamic two player pebble game on graphs.

It is shown how to implement the strategy and the upper bound on the required number of pebbles is improved from $2b + 2d + O(\sqrt{b})$ to $d + 2b$, where b is the bound of the in-degree and d the bound of the out-degree. We also give a lower bound that shows that the number of pebbles depends on the out-degree d.

47 Gerth Stølting Brodal and Chris Okasaki, *Optimal Purely Functional Priority Queues*. In *Journal of Functional Programming*, Volume 6(6), pages 839–858, November 1996, doi: 10.1017/S095679680000201X.

Abstract: Brodal recently introduced the first implementation of imperative priority queues to support *findMin*, *insert*, and *meld* in $O(1)$ worst-case time, and *deleteMin* in $O(\log n)$ worst-case time. These bounds are asymptotically optimal among all comparison-based priority queues. In this paper, we adapt Brodal's data structure to a purely functional setting. In doing so, we both

simplify the data structure and clarify its relationship to the binomial queues of Vuillemin, which support all four operations in $O(\log n)$ time. Specifically, we derive our implementation from binomial queues in three steps: first, we reduce the running time of *insert* to $O(1)$ by eliminating the possibility of cascading links; second, we reduce the running time of *findMin* to $O(1)$ by adding a global root to hold the minimum element; and finally, we reduce the running time of *meld* to $O(1)$ by allowing priority queues to contain other priority queues. Each of these steps is expressed using ML-style functors. The last transformation, known as data-structural bootstrapping, is an interesting application of higher-order functors and recursive structures.

Conference Papers

48 Gerth Stølting Brodal, *A Simple Integer Successor-Delete Data Structure*. To appear in *Proc. 23rd International Symposium on Experimental Algorithms*, Leibniz International Proceedings in Informatics. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2025.

Abstract: We consider a simple decremental data structure for maintaining a set of integers, that supports initializing the set to $\{1, 2, \ldots, n\}$ followed by d deletions and s successor queries in arbitrary order in total $O(n+d+s\cdot(1+\log_{\max(2,s/n)}\min(s,n)))$ time. The data structure consists of a single array of n integers. A straightforward modification allows the data structure to also support p predecessor and r range queries, with a total output k, in total $O(n+d+k+q\cdot(1+\log_{\max(2,q/n)}\min(q,n)))$ time, where $q = s+p+r$. The data structure is essentially a special case of the classic union-find data structure with path compression but with unweighted linking (i.e., without linking by rank or size), that is known to achieve logarithmic amortized time bounds (Tarjan and van Leeuwen, 1984). In this paper we study the efficiency of this simple data structure, and compare it to other, theoretically superior, data structures.

49 Gerth Stølting Brodal and Casper Moldrup Rysgaard, *Pure Binary Finger Search Trees*. In *Proc. 8th SIAM Symposium on Simplicity in Algorithms*, pages 172–195, 2025, doi: 10.1137/1.9781611978315.14.

Abstract: We present dynamic binary search trees where each node only stores a value and pointers to its parent and its children. We denote such binary search trees *pure* binary search trees. Our structure supports finger searches in worst-case $O(\lg d)$ time, where d is the rank difference between the node given by the finger and the node found by the search. Inserting a new node with a successor or predecessor value for a node pointed to by a finger and deleting a node in the tree pointed to by a finger are supported in amortized $O(1)$ time and worst-case $O(\lg n)$ time, where n is the number of nodes in the tree. The temporary working space during the operations is $O(1)$ words. The result is obtained by an alternative representation of the red-black trees by Guibas and Sedgewick [FOCS 1978] that encodes bits of information in the tree structure, generalizing the encoding of 2-3-trees by Brown [IPL 1979], and rearranging the nodes in a red-black tree ("folding" left and right paths) such that the predecessor and successor of a node can always be found in worst-case constant time. The same time bounds can easily be obtained by, say, red-black trees and AVL trees augmented with pointers to the predecessor and successor of each node. The novelty of our result is that we store no extra information than the binary tree structure. The structure can be represented by two pointers per value, i.e., the same representation as a doubly linked list.

50 Gerth Stølting Brodal, Rolf Fagerberg and Casper Moldrup Rysgaard, *On Finding Longest Palindromic Subsequences using Longest Common Subsequences*. In *Proc. 32nd Annual European Symposium on Algorithms*, Volume 308 of Leibniz International Proceedings in Informatics, Article no. 35, pages 35:1–35:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2024, doi: 10.4230/LIPIcs.ESA.2024.35.

Abstract: Two standard textbook problems illustrating dynamic programming are to find the longest common subsequence (LCS) between two strings and to find the longest palindromic subsequence (LPS) of a single string. A popular claim is that the longest palindromic subsequence in a string can be computed as the longest common subsequence between the string and the reversed string. We prove that the correctness of this claim depends on how the longest common subse-

quence is computed. In particular, we prove that the classical dynamic programming solution by Wagner and Fischer [JACM 1974] for finding an LCS in fact does find an LPS, while a slightly different LCS backtracking strategy makes the algorithm fail to always report a palindrome.

51 Gerth Stølting Brodal, *Bottom-up Rebalancing Binary Search Trees by Flipping a Coin*. In *Proc. 12th International Conference on Fun With Algorithms*, Volume 291 of Leibniz International Proceedings in Informatics, Article no. 6, pages 6:1–6:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2024, doi: 10.4230/LIPIcs.FUN.2024.6 (presentation pdf, pptx).

Abstract: Rebalancing schemes for dynamic binary search trees are numerous in the literature, where the goal is to maintain trees of low height, either in the worst-case or expected sense. In this paper we study randomized rebalancing schemes for sequences of n insertions into an initially empty binary search tree, under the assumption that a tree only stores the elements and the tree structure without any additional balance information. Seidel (2009) presented a top-down randomized insertion algorithm, where insertions take expected $O(\lg^2 n)$ time, and the resulting trees have the same distribution as inserting a uniform random permutation into a binary search tree without rebalancing. Seidel states as an open problem if a similar result can be achieved with bottom-up insertions. In this paper we fail to answer this question.

We consider two simple canonical randomized bottom-up insertion algorithms on binary search trees, assuming that an insertion is given the position where to insert the next element. The subsequent rebalancing is performed bottom-up in expected $O(1)$ time, uses expected $O(1)$ random bits, performs at most two rotations, and the rotations appear with geometrically decreasing probability in the distance from the leaf. For some insertion sequences the expected depth of each node is proved to be $O(\lg n)$. On the negative side, we prove for both algorithms that there exist simple insertion sequences where the expected depth is $\Omega(n)$, i.e., the studied rebalancing schemes are *not* competitive with (most) other rebalancing schemes in the literature.

52 Gerth Stølting Brodal and Sebastian Wild, *Deterministic Cache-Oblivious Funnelselect*. In *Proc. 19th Scandinavian Workshop on Algorithm Theory*, Volume 294 of Leibniz International Proceedings in Informatics, Article no. 17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2024, doi: 10.4230/LIPIcs.SWAT.2024.17 (presentation pdf, pptx).

Abstract: In the multiple-selection problem one is given an unsorted array S of N elements and an array of q query ranks $r_1 < \cdots < r_q$, and the task is to return, in sorted order, the q elements in S of rank $r_1, \ldots, r_q$, respectively. The asymptotic deterministic comparison complexity of the problem was settled by Dobkin and Munro [JACM 1981]. In the I/O model an optimal I/O complexity was achieved by Hu *et al.* [SPAA 2014]. Recently [ESA 2023], we presented a *cache-oblivious* algorithm with matching I/O complexity, named *funnelselect*, since it heavily borrows ideas from the cache-oblivious sorting algorithm *funnelsort* from the seminal paper by Frigo, Leiserson, Prokop and Ramachandran [FOCS 1999]. Funnelselect is inherently randomized as it relies on sampling for cheaply finding many good pivots. In this paper we present *deterministic funnelselect*, achieving the same optimal I/O complexity cache-obliviously without randomization. Our new algorithm essentially replaces a single (in expectation) reversed-funnel computation using random pivots by a recursive algorithm using multiple reversed-funnel computations. To meet the I/O bound, this requires a carefully chosen subproblem size based on the entropy of the sequence of query ranks; deterministic funnelselect thus raises distinct technical challenges not met by randomized funnelselect. The resulting worst-case I/O bound is $O(\sum_{i=1}^{q+1} \frac{\Delta_i}{B} \cdot \log_{M/B} \frac{N}{\Delta_i} + \frac{N}{B})$, where B is the external memory block size, $M \geq B^{1+\epsilon}$ is the internal memory size, for some constant $\epsilon > 0$, and $\Delta_i = r_i - r_{i-1}$ (assuming $r_0 = 0$ and $r_{q+1} = N + 1$).

(9) Bruce Brewer, Gerth Stølting Brodal and Haitao Wang, *Dynamic Convex Hulls for Simple Paths*. In *Proc. 40th International Symposium on Computational Geometry*, Volume 293 of Leibniz International Proceedings in Informatics, Article no. 24, pages 24:1–24:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2024, doi: 10.4230/LIPIcs.SoCG.2024.24.

Abstract: We consider two restricted cases of the planar dynamic convex hull problem with point

insertions and deletions. We assume all updates are performed on a deque (double-ended queue) of points. The first case considers the monotonic path case, where all points are sorted in a given direction, say horizontally left-to-right, and only the leftmost and rightmost points can be inserted and deleted. The second case, which is more general, assumes that the points in the deque constitute a simple path. For both cases, we present solutions supporting deque insertions and deletions in worst-case constant time and standard queries on the convex hull of the points in $O(\log n)$ time, where n is the number of points in the current point set. The convex hull of the current point set can be reported in $O(h + \log n)$ time, where h is the number of edges of the convex hull. For the 1-sided monotone path case, where updates are only allowed on one side, the reporting time can be reduced to $O(h)$, and queries on the convex hull are supported in $O(\log h)$ time. All our time bounds are worst case. In addition, we prove lower bounds that match these time bounds, and thus our results are optimal.

53 Gerth Stølting Brodal and Sebastian Wild, *Funnelselect: Cache-Oblivious Multiple Selection*. In *Proc. 31st Annual European Symposium on Algorithms*, Volume 274 of Leibniz International Proceedings in Informatics, Article no. 25, pages 25:1—25:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2023, doi: 10.4230/LIPIcs.ESA.2023.25.

Abstract: We present the algorithm *funnelselect*, the first optimal randomized cache-oblivious algorithm for the multiple-selection problem. The algorithm takes as input an unsorted array of N elements and q query ranks $r_1 < \cdots < r_q$, and returns in sorted order the q input elements of rank $r_1, \ldots, r_q$, respectively. The algorithm uses expected and with high probability $O\left(\sum_{i=1}^{q+1} \frac{\Delta_i}{B} \cdot \log_{M/B} \frac{N}{\Delta_i} + \frac{N}{B}\right)$ I/Os, where B is the external memory block size, $M \geq B^{1+\varepsilon}$ is the internal memory size, for some constant $\varepsilon > 0$, and $\Delta_i = r_i - r_{i-1}$ (assuming $r_0 = 0$ and $r_{q+1} = N + 1$). This is the best possible I/O bound in the cache-oblivious and external memory models. The result is achieved by reversing the computation of the cache-oblivious sorting algorithm funnelsort by Frigo, Leiserson, Prokop and Ramachandran [FOCS 1999], using randomly selected pivots for distributing elements, and pruning computations that with high probability are not expected to contain any query ranks.

54 Gerth Stølting Brodal, Casper Moldrup Rysgaard, Jens Kristian Refsgaard Schou and Rolf Svenning, *Space Efficient Functional Off-line Partial Persistent Trees with Applications to Planar Point Location*. In *Proc. 18th International Workshop on Algorithms and Data Structures*, Volume 14079 of Lecture Notes in Computer Science, pages 644–659. Springer Verlag, Berlin, 2023, doi: 10.1007/978-3-031-38906-1_43.

Abstract: In 1989 Driscoll, Sarnak, Sleator, and Tarjan presented general space-efficient transformations for making ephemeral data structures persistent. The main contribution of this paper is to adapt this transformation to the functional model. We present a general transformation of an ephemeral, linked data structure into an offline, partially persistent, purely functional data structure with additive $O(n \log n)$ construction time and $O(n)$ space overhead; with n denoting the number of ephemeral updates. An application of our transformation allows the elegant slab-based algorithm for planar point location by Sarnak and Tarjan 1986 to be implemented space efficiently in the functional model using linear space.

55 Gerth Stølting Brodal, Casper Moldrup Rysgaard and Rolf Svenning, *External Memory Fully Persistent Search Trees*. *Proc. 55th Annual ACM Symposium on Theory of Computing*, In *55th Annual ACM Symposium on Theory of Computing*, pages 1410–1423, 2023, doi: 10.1145/3564246.3585140.

Abstract: We present the first fully-persistent external-memory search tree achieving amortized I/O bounds matching those of the classic (ephemeral) B-tree by Bayer and McCreight. The insertion and deletion of a value in any version requires amortized $O(\log_B N_v)$ I/Os and a range reporting query in any version requires worst-case $O(\log_B N_v + K/B)$ I/Os, where K is the number of values reported, N_v is the number of values in the version v of the tree queried or updated, and B is the external-memory block size. The data structure requires space linear in the total number of updates. Compared to the previous best bounds for fully persistent B-trees [Brodal, Sioutas, Tsakalidis, and Tsichlas, SODA 2012], this paper eliminates from the update bound an additive

term of $O(\log_2 B)$ I/Os. This result matches the previous best bounds for the restricted case of partial persistent B-trees [Arge, Danner and Teh, JEA 2003]. Central to our approach is to consider the problem as a dynamic set of two-dimensional rectangles that can be merged and split.

(11) Gerth Stølting Brodal, *Priority Queues with Decreasing Keys*. In *Proc. 11th International Conference on Fun With Algorithms*, Volume 226 of Leibniz International Proceedings in Informatics, Article no. 8, pages 8:1—8:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2022, doi: 10.4230/LIPIcs.FUN.2022.8 (presentation pdf, pptx).

Abstract: A priority queue stores a set of items with associated keys and supports the insertion of a new item and extraction of an item with minimum key. In applications like Dijkstra's single source shortest path algorithm and Prim-Jarník's minimum spanning tree algorithm, the key of an item can decrease over time. Usually this is handled by either using a priority queue supporting the deletion of an arbitrary item or a dedicated `DecreaseKey` operation, or by inserting the same item multiple times but with decreasing keys.

In this paper we study what happens if the keys associated with items in a priority queue can decrease over time *without* informing the priority queue, and how such a priority queue can be used in Dijkstra's algorithm. We show that binary heaps with bottom-up insertions fail to report items with unchanged keys in correct order, while binary heaps with top-down insertions report items with unchanged keys in correct order. Furthermore, we show that skew heaps, leftist heaps, and priority queues based on linking roots of heap-ordered trees, like pairing heaps, binomial queues and Fibonacci heaps, work correctly with decreasing keys without any modifications. Finally, we show that the post-order heap by Harvey and Zatloukal, a variant of a binary heap with amortized constant time insertions and amortized logarithmic time deletions, works correctly with decreasing keys and is a strong contender for an implicit priority queue supporting decreasing keys in practice.

56 Gerth Stølting Brodal, Rolf Fagerberg, David Hammer, Ulrich Meyer, Manuel Penschuck and Hung Tran, *An Experimental Study of External Memory Algorithms for Connected Components*. In *Proc. 19th International Symposium on Experimental Algorithms*, Volume 190 of Leibniz International Proceedings in Informatics, Article no. 23, pages 23:1—23:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2021, doi: 10.4230/LIPIcs.SEA.2021.23.

Abstract: We empirically investigate algorithms for solving Connected Components in the external memory model. In particular, we study whether the randomized $O(\mathrm{Sort}(E))$ algorithm by Karger, Klein, and Tarjan can be implemented to compete with practically promising and simpler algorithms having only slightly worse theoretical cost, namely Bor uvka's algorithm and the algorithm by Sibeyn and collaborators. For all algorithms, we develop and test a number of tuning options. Our experiments are executed on a large set of different graph classes including random graphs, grids, geometric graphs, and hyperbolic graphs. Among our findings are: The Sibeyn algorithm is a very strong contender due to its simplicity and due to an added degree of freedom in its internal workings when used in the Connected Components setting. With the right tunings, the Karger-Klein-Tarjan algorithm can be implemented to be competitive in many cases. Higher graph density seems to benefit Karger-Klein-Tarjan relative to Sibeyn. Bor uvka's algorithm is not competitive with the two others.

57 Gerth Stølting Brodal, *Soft Sequence Heaps*. In *Proc. 4th SIAM Symposium on Simplicity in Algorithms*, pages 14–24, 2021, doi: 10.1137/1.9781611976496.2 (presentation pdf, pptx, mp4).

Abstract: Chazelle [Journal of the ACM 2000] introduced the *soft heap* as a building block for efficient minimum spanning tree algorithms, and recently Kaplan *et al.* [SOSA 2019] showed how soft heaps can be applied to achieve simpler algorithms for various selection problems. A soft heap trades-off accuracy for efficiency, by allowing ϵN of the items in a heap to be *corrupted* after a total of N insertions, where a corrupted item is an item with artificially increased key and $0 < \epsilon \leq \frac{1}{2}$ is a fixed error parameter. Chazelle's soft heaps are based on binomial trees and support insertions in amortized $O(\lg \frac{1}{\epsilon})$ time and extract-min operations in amortized $O(1)$ time.

In this paper we explore the design space of soft heaps. The main contribution of this paper is an alternative soft heap implementation based on merging sorted sequences, with time bounds

matching those of Chazelle's soft heaps. We also discuss a variation of the soft heap by Kaplan *et al.* [SIAM Journal of Computing 2013], where we avoid performing insertions lazily. It is based on ternary trees instead of binary trees and matches the time bounds of Kaplan *et al.*, i.e. amortized $O(1)$ insertions and amortized $O(\lg \frac{1}{\epsilon})$ extract-min. Both our data structures only introduce corruptions after extract-min operations which return the set of items corrupted by the operation.

(15) Edvin Berglin and Gerth Stølting Brodal, *A Simple Greedy Algorithm for Dynamic Graph Orientation*. In *Proc. 28th Annual International Symposium on Algorithms and Computation*, Volume 92 of Leibniz International Proceedings in Informatics, pages 12:1–12:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2017, doi: 10.4230/LIPIcs.ISAAC.2017.12.

Abstract: *Graph orientations with low out-degree* are one of several ways to efficiently store sparse graphs. If the graphs allow for insertion and deletion of edges, one may have to *flip* the orientation of some edges to prevent blowing up the maximum out-degree. We use arboricity as our sparsity measure. With an immensely simple greedy algorithm, we get parametrized trade-off bounds between out-degree and worst case number of flips, which previously only existed for amortized number of flips. We match the previous best worst-case algorithm (in $O(\log n)$ flips) for general arboricity and beat it for either constant or super-logarithmic arboricity. We also match a previous best amortized result for at least logarithmic arboricity, and give the first results with worst-case $O(1)$ and $O(\sqrt{\log n})$ flips nearly matching degree bounds to their respective amortized solutions.

(12) Gerth Stølting Brodal and Konstantinos Mampentzidis, *Cache Oblivious Algorithms for Computing the Triplet Distance between Trees*. In *Proc. 25th Annual European Symposium on Algorithms*, Volume 87 of Leibniz International Proceedings in Informatics, pages 21:1–21:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2017, doi: 10.4230/LIPIcs.ESA.2017.21.

Abstract: We study the problem of computing the triplet distance between two rooted unordered trees with n labeled leafs. Introduced by Dobson 1975, the triplet distance is the number of leaf triples that induce different topologies in the two trees. The current theoretically best algorithm is an $O(n \log n)$ time algorithm by Brodal *et al.* (SODA 2013). Recently Jansson *et al.* proposed a new algorithm that, while slower in theory, requiring $O(n \log^3 n)$ time, in practice it outperforms the theoretically faster $O(n \log n)$ algorithm. Both algorithms do not scale to external memory.

We present two cache oblivious algorithms that combine the best of both worlds. The first algorithm is for the case when the two input trees are binary trees and the second a generalized algorithm for two input trees of arbitrary degree. Analyzed in the RAM model, both algorithms require $O(n \log n)$ time, and in the cache oblivious model $O(n/B \log_2(n/M))$ I/Os. Their relative simplicity and the fact that they scale to external memory makes them achieve the best practical performance. We note that these are the first algorithms that scale to external memory, both in theory and practice, for this problem.

58 Gerth Stølting Brodal, *External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates*. In *Proc. 33rd Annual Symposium on Theoretical Aspects of Computer Science*, Volume 47 of Leibniz International Proceedings in Informatics, pages 23:1–23:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2016, doi: 10.4230/LIPIcs.STACS.2016.23 (presentation pdf, pptx).

Abstract: An external memory data structure is presented for maintaining a dynamic set of N two-dimensional points under the insertion and deletion of points, and supporting unsorted 3-sided range reporting queries and top-k queries, where top-k queries report the k points with highest y-value within a given x-range. For any constant $0 < \varepsilon \leq \frac{1}{2}$, a data structure is constructed that supports updates in amortized $O(\frac{1}{\varepsilon B^{1-\varepsilon}} \log_B N)$ IOs and queries in amortized $O(\frac{1}{\varepsilon} \log_B N + K/B)$ IOs, where B is the external memory block size, and K is the size of the output to the query (for top-k queries K is the minimum of k and the number of points in the query interval). The data structure uses linear space. The update bound is a significant factor $B^{1-\varepsilon}$ improvement over the previous best update bounds for these two query problems, while staying within the same query and space bounds.

59 Gerth Stølting Brodal, Jesper Sindahl Nielsen and Jakob Truelsen, *Strictly Implicit Priority Queues: On the Number of Moves and Worst-Case Time*. In *Proc. 14th International Workshop on Algorithms and Data Structures*, Volume 9214 of Lecture Notes in Computer Science, pages 1–12. Springer Verlag, Berlin, 2015, doi: 10.1007/978-3-319-21840-3_8.

Abstract: The binary heap of Williams (1964) is a simple priority queue characterized by only storing an array containing the elements and the number of elements n – here denoted a *strictly implicit* priority queue. We introduce two new strictly implicit priority queues. The first structure supports amortized $O(1)$ time Insert and $O(\log n)$ time ExtractMin operations, where both operations require amortized $O(1)$ element moves. No previous implicit heap with $O(1)$ time Insert supports both operations with $O(1)$ moves. The second structure supports worst-case $O(1)$ time Insert and $O(\log n)$ time (and moves) ExtractMin operations. Previous results were either amortized or needed $O(\log n)$ bits of additional state information between operations.

60 Djamal Belazzougui, Gerth Stølting Brodal and Jesper Sindahl Nielsen, *Expected Linear Time Sorting for Word Size $\Omega(\log^2 n \log\log n)$*. In *Proc. 14th Scandinavian Workshop on Algorithm Theory*, Volume 8503 of Lecture Notes in Computer Science, pages 26–37. Springer Verlag, Berlin, 2014, doi: 10.1007/978-3-319-08404-6_3.

Abstract: Sorting n integers in the word-RAM model is a fundamental problem and a long-standing open problem is whether integer sorting is possible in linear time when the word size is $\omega(\log n)$. In this paper we give an algorithm for sorting integers in expected linear time when the word size is $\Omega(\log^2 n \log\log n)$. Previously expected linear time sorting was only possible for word size $\Omega(\log^{2+\varepsilon} n)$. Part of our construction is a new packed sorting algorithm that sorts n integers of w/b-bits packed in $O(n/b)$ words, where b is the number of integers packed in a word of size w bits. The packed sorting algorithm runs in expected $O(n/b \cdot (\log n + \log^2 b))$ time.

61 Gerth Stølting Brodal and Kasper Green Larsen, *Optimal Planar Orthogonal Skyline Counting Queries*. In *Proc. 14th Scandinavian Workshop on Algorithm Theory*, Volume 8503 of Lecture Notes in Computer Science, pages 98–109. Springer Verlag, Berlin, 2014, doi: 10.1007/978-3-319-08404-6_10 (presentation pdf, pptx).

Abstract: The skyline of a set of points in the plane is the subset of maximal points, where a point (x, y) is maximal if no other point (x', y') satisfies $x' \geq x$ and $y' \geq y$. We consider the problem of preprocessing a set P of n points into a space efficient static data structure supporting orthogonal skyline counting queries, i.e. given a query rectangle R to report the size of the skyline of $P \cap R$. We present a data structure for storing n points with integer coordinates having query time $O(\lg n / \lg\lg n)$ and space usage $O(n)$ words. The model of computation is a unit cost RAM with logarithmic word size. We prove that these bounds are the best possible by presenting a matching lower bound in the cell probe model with logarithmic word size: Space usage $n \lg^{O(1)} n$ implies worst case query time $\Omega(\lg n / \lg\lg n)$.

62 Morten Kragelund Holt, Jens Johansen and Gerth Stølting Brodal, *On the Scalability of Computing Triplet and Quartet Distances*. In *Proc. 16th Workshop on Algorithm Engineering and Experiments*, pages 9–19, 2014, doi: 10.1137/1.9781611973198.2 (presentation pdf, pptx).

Abstract: In this paper we present an experimental evaluation of the algorithms by Brodal *et al.* [SODA 2013] for computing the triplet and quartet distance measures between two leaf labelled rooted and unrooted trees of arbitrary degree, respectively. The algorithms count the number of rooted tree topologies over sets of three leaves (*triplets*) and unrooted tree topologies over four leaves (*quartets*), respectively, that have different topologies in the two trees.

The algorithms by Brodal *et al.* maintain a long sequence of variables (hundreds for quartets) for counting different cases to be considered by the algorithm, making it unclear if the algorithms would be of theoretical interest only. In our experimental evaluation of the algorithms the typical overhead per node is about 2 KB and 10 KB per node in the input trees for triplet and quartet computations, respectively. This allows us to compute the distance measures for trees with up to millions of nodes. The limiting factor is the amount of memory available. With 31 GB of memory all our input instances can be solved within a few minutes.

In the algorithm by Brodal *et al.* a few choices were made, where alternative solutions possibly could improve the algorithm, in particular for quartet distance computations. For quartet computations we expand the algorithm to also consider alternative computations, and make two observations: First we observe that the running time can be improved from $O(\max(d_1, d_2) \cdot n \cdot \lg n)$ to $O(\min(d_1, d_2) \cdot n \cdot \lg n)$, where n is the number of leaves in the two trees, and d_1 and d_2 are the maximum degrees of the nodes in the two trees, respectively. Secondly, by taking a different approach to counting the number of disagreeing quartets we can reduce the number of calculations needed to calculate the quartet distance, improving both the running time and the space requirement by our algorithm by a constant factor.

63 Lars Arge, Gerth Stølting Brodal, Jakob Truelsen and Constantinos Tsirogiannis, *An Optimal and Practical Cache-Oblivious Algorithm for Computing Multiresolution Rasters*. In *Proc. 21st Annual European Symposium on Algorithms*, Volume 8125 of Lecture Notes in Computer Science, pages 61–72. Springer Verlag, Berlin, 2013, doi: 10.1007/978-3-642-40450-4_6.

Abstract: In many scientific applications it is required to reconstruct a raster dataset many times, each time using a different resolution. This leads to the following problem; let G be a raster of $\sqrt{N} \times \sqrt{N}$ cells. We want to compute for every integer $2 \leq \mu \leq \sqrt{N}$ a raster G_μ of $\lceil \sqrt{N}/\mu \rceil \times \lceil \sqrt{N}/\mu \rceil$ cells where each cell of G_μ stores the average of the values of $\mu \times \mu$ cells of G. Here we consider the case where G is so large that it does not fit in the main memory of the computer.

We present a novel algorithm that solves this problem in $O(\text{scan}(N))$ data block transfers from/to the external memory, and in $\Theta(N)$ CPU operations; here $\text{scan}(N)$ is the number of block transfers that are needed to read the entire dataset from the external memory. Unlike previous results on this problem, our algorithm achieves this optimal performance without making any assumptions on the size of the main memory of the computer. Moreover, this algorithm is cache-oblivious; its performance does not depend on the data block size and the main memory size.

We have implemented the new algorithm and we evaluate its performance on datasets of various sizes; we show that it clearly outperforms previous approaches on this problem. In this way, we provide solid evidence that non-trivial cache-oblivious algorithms can be implemented so that they perform efficiently in practice.

64 Gerth Stølting Brodal, Andrej Brodnik and Pooya Davoodi, *The Encoding Complexity of Two Dimensional Range Minimum Data Structures*. In *Proc. 21st Annual European Symposium on Algorithms*, Volume 8125 of Lecture Notes in Computer Science, pages 229–240. Springer Verlag, Berlin, 2013, doi: 10.1007/978-3-642-40450-4_20 (presentation pdf, pptx).

Abstract: In the two-dimensional range minimum query problem an input matrix A of dimension $m \times n$, $m \leq n$, has to be preprocessed into a data structure such that given a query rectangle within the matrix, the position of a minimum element within the query range can be reported. We consider the space complexity of the encoding variant of the problem where queries have access to the constructed data structure but can not access the input matrix A, i.e. all information must be encoded in the data structure. Previously it was known how to solve the problem with space $O(mn \min\{m, \log n\})$ bits (and with constant query time), but the best lower bound was $\Omega(mn \log m)$ bits, i.e. leaving a gap between the upper and lower bounds for non-quadratic matrices. We show that this space lower bound is optimal by presenting an encoding scheme using $O(mn \log m)$ bits. We do not consider query time.

65 Gerth Stølting Brodal, *A Survey on Priority Queues*. In *Proc. Conference on Space Efficient Data Structures, Streams and Algorithms – Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*, Volume 8066 of Lecture Notes in Computer Science, pages 150–163. Springer Verlag, Berlin, 2013, doi: 10.1007/978-3-642-40273-9_11 (presentation pdf, pptx).

Abstract: Back in 1964 Williams introduced the binary heap as a basic priority queue data structure supporting the operations Insert and ExtractMin in logarithmic time. Since then numerous papers have been published on priority queues. This paper tries to list some of the directions research on priority queues has taken the last 50 years.

(23) Andreas Sand, Gerth Stølting Brodal, Rolf Fagerberg, Christian Nørgaard Storm Pedersen and Thomas Mailund, *A practical $O(n \log^2 n)$ time algorithm for computing the triplet distance on binary trees*. In *Proc. 11th Asia Pacific Bioinformatics Conference*, Advances in Bioinformatics & Computational. Tsinghua University Press, 2013.

Abstract: The triplet distance is a distance measure that compares two rooted trees on the same set of leaves by enumerating all sub-sets of three leaves and counting how often the induced topologies of the tree are equal or different. We present an algorithm that computes the triplet distance between two rooted binary trees in time $O(n \log^2 n)$. The algorithm is related to an algorithm for computing the quartet distance between two unrooted binary trees in time $O(n \log n)$. While the quartet distance algorithm has a very severe overhead in the asymptotic time complexity that makes it impractical compared to $O(n^2)$ time algorithms, we show through experiments that the triplet distance algorithm can be implemented to give a competitive wall-time running time.

66 Gerth Stølting Brodal, Rolf Fagerberg, Christian Nørgaard Storm Pedersen, Thomas Mailund and Andreas Sand, *Efficient Algorithms for Computing the Triplet and Quartet Distance Between Trees of Arbitrary Degree*. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1814–1832, 2013, doi: `knowledgecenter.siam.org/0236-000098` (presentation pdf, pptx).

Abstract: The triplet and quartet distances are distance measures to compare two rooted and two unrooted trees, respectively. The leaves of the two trees should have the same set of n labels. The distances are defined by enumerating all subsets of three labels (triplets) and four labels (quartets), respectively, and counting how often the induced topologies in the two input trees are different. In this paper we present efficient algorithms for computing these distances. We show how to compute the triplet distance in time $O(n \log n)$ and the quartet distance in time $O(dn \log n)$, where d is the maximal degree of any node in the two trees. Within the same time bounds, our framework also allows us to compute the parameterized triplet and quartet distances, where a parameter is introduced to weight resolved (binary) topologies against unresolved (non-binary) topologies. The previous best algorithm for computing the triplet and parameterized triplet distances have $O(n^2)$ running time, while the previous best algorithms for computing the quartet distance include an $O(d^9 n \log n)$ time algorithm and an $O(n^{2.688})$ time algorithm, where the latter can also compute the parameterized quartet distance. Since $d \leq n$, our algorithms improve on all these algorithms.

67 Gerth Stølting Brodal, Jesper Sindahl Nielsen and Jakob Truelsen, *Finger Search in the Implicit Model*. In *Proc. 23th Annual International Symposium on Algorithms and Computation*, Volume 7676 of Lecture Notes in Computer Science, pages 527–536. Springer Verlag, Berlin, 2012, doi: `10.1007/978-3-642-35261-4_55`.

Abstract: We address the problem of creating a dictionary with the finger search property in the strict implicit model, where no information is stored between operations, except the array of elements. We show that for any implicit dictionary supporting finger searches in $q(t) = \Omega(\log t)$ time, the time to move the finger to another element is $\Omega(q^{-1}(\log n))$, where t is the rank distance between the query element and the finger. We present an optimal implicit static structure matching this lower bound. We furthermore present a near optimal implicit dynamic structure supporting `search`, `change-finger`, `insert`, and `delete` in times $O(q(t))$, $O(q^{-1}(\log n) \log n)$, $O(\log n)$, and $O(\log n)$, respectively, for any $q(t) = \Omega(\log t)$. Finally we show that the `search` operation must take $\Omega(\log n)$ time for the special case where the finger is always changed to the element returned by the last query.

(16) Gerth Stølting Brodal, Pooya Davoodi, Moshe Lewenstein, Rajeev Raman and S. Srinivasa Rao, *Two Dimensional Range Minimum Queries and Fibonacci Lattices*. In *Proc. 20th Annual European Symposium on Algorithms*, Volume 7501 of Lecture Notes in Computer Science, pages 217–228. Springer Verlag, Berlin, 2012, doi: `10.1007/978-3-642-33090-2_20`.

Abstract: Given a matrix of size N, two dimensional range minimum queries (2D-RMQs) ask for the position of the minimum element in a rectangular range within the matrix. We study trade-offs between the query time and the additional space used by indexing data structures that support 2D-RMQs. Using a novel technique—the discrepancy properties of Fibonacci lattices—we give an indexing data structure for 2D-RMQs that uses $O(N/c)$ bits additional space with $O(c \log c (\log \log c)^2)$ query time, for any parameter c, $4 \leq c \leq N$. Also, when the entries of

the input matrix are from $\{0,1\}$, we show that the query time can be improved to $O(c \log c)$ with the same space usage.

(10) Gerth Stølting Brodal, George Lagogiannis and Robert E. Tarjan, *Strict Fibonacci Heaps*. In *Proc. 44th Annual ACM Symposium on Theory of Computing*, pages 1177–1184, 2012, doi: 10.1145/2213977.2214082 (presentation pdf, pptx).

Abstract: We present the first pointer-based heap implementation with time bounds matching those of Fibonacci heaps in the worst case. We support make-heap, insert, find-min, meld and decrease-key in worst-case $O(1)$ time, and delete and delete-min in worst-case $O(\lg n)$ time, where n is the size of the heap. The data structure uses linear space.

A previous, very complicated, solution achieving the same time bounds in the RAM model made essential use of arrays and extensive use of redundant counter schemes to maintain balance. Our solution uses neither. Our key simplification is to discard the structure of the smaller heap when doing a meld. We use the pigeonhole principle in place of the redundant counter mechanism.

68 Gerth Stølting Brodal and Casper Kejlberg-Rasmussen, *Cache-Oblivious Implicit Predecessor Dictionaries with the Working Set Property*. In *Proc. 29th Annual Symposium on Theoretical Aspects of Computer Science*, Volume 14 of Leibniz International Proceedings in Informatics, pages 112–123. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2012, doi: 10.4230/LIPIcs.STACS.2012.112.

Abstract: In this paper we present an implicit dynamic dictionary with the working-set property, supporting insert(e) and delete(e) in $O(\log n)$ time, predecessor(e) in $O(\log \ell_{p(e)})$ time, successor(e) in $O(\log \ell_{s(e)})$ time and search(e) in $O(\log \min(\ell_{p(e)}, \ell_e, \ell_{s(e)}))$ time, where n is the number of elements stored in the dictionary, ℓ_e is the number of distinct elements searched for since element e was last searched for and $p(e)$ and $s(e)$ are the predecessor and successor of e, respectively. The time-bounds are all worst-case. The dictionary stores the elements in an array of size n using *no* additional space. In the cache-oblivious model the log is base B and the cache-obliviousness is due to our black box use of an existing cache-oblivious implicit dictionary. This is the first implicit dictionary supporting predecessor and successor searches in the working-set bound. Previous implicit structures required $O(\log n)$ time.

(14) Gerth Stølting Brodal, Spyros Sioutas, Konstantinos Tsakalidis and Kostas Tsichlas, *Fully Persistent B-trees*. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 602–614, 2012, doi: 10.1137/1.9781611973099.51 (presentation pdf, pptx).

Abstract: We present I/O-efficient fully persistent B-Trees that support range searches at any version in $O(\log_B n + t/B)$ I/Os and updates at any version in $O(\log_B n + \log_2 B)$ amortized I/Os, using space $O(m/B)$ disk blocks. By n we denote the number of elements in the accessed version, by m the total number of updates, by t the size of the query's output, and by B the disk block size. The result improves the previous fully persistent B-Trees of Lanka and Mays by a factor of $O(\log_B m)$ for the range query complexity and $O(\log_B n)$ for the update complexity. To achieve the result, we first present a new B-Tree implementation that supports searches and updates in $O(\log_B n)$ I/Os, using $O(n/B)$ blocks of space. Moreover, every update makes in the worst case a constant number of modifications to the data structure. We make these B-Trees fully persistent using an I/O-efficient method for full persistence that is inspired by the *node-splitting* method of Driscoll et al. The method we present is interesting in its own right and can be applied to any external memory pointer based data structure with maximum in-degree d_{in} bounded by a constant and out-degree bounded by $O(B)$, where every node occupies a constant number of blocks on disk. The I/O-overhead per modification to the ephemeral structure is $O(d_{in} \log_2 B)$ amortized I/Os, and the space overhead is $O(d_{in}/B)$ amortized blocks. Access to a field of an ephemeral block is supported in $O(\log_2 d_{in})$ worst case I/Os.

(18) Gerth Stølting Brodal, Gabriel Moruz and Andrei Negoescu, *OnlineMin: A Fast Strongly Competitive Randomized Paging Algorithm*. In *Proc. 9th Workshop on Approximation and Online Algorithms*, Volume 7164 of Lecture Notes in Computer Science, pages 164–175. Springer Verlag,

Berlin, 2011, doi: 10.1007/978-3-642-29116-6_14.

Abstract: In the field of online algorithms paging is one of the most studied problems. For randomized paging algorithms a tight bound of H_k on the competitive ratio has been known for decades, yet existing algorithms matching this bound have high running times. We present the first randomized paging approach that both has optimal competitiveness and selects victim pages in subquadratic time. In fact, if k pages fit in internal memory the best previous solution required $O(k^2)$ time per request and $O(k)$ space, whereas our approach takes also $O(k)$ space, but only $O(\log k)$ time in the worst case per page request.

69 Gerth Stølting Brodal, Pooya Davoodi and S. Srinivasa Rao, *Path Minima Queries in Dynamic Weighted Trees*. In *Proc. 12th International Workshop on Algorithms and Data Structures*, Volume 6844 of Lecture Notes in Computer Science, pages 290–301. Springer Verlag, Berlin, 2011, doi: 10.1007/978-3-642-22300-6_25.

Abstract: In the path minima problem on trees each tree edge is assigned a weight and a query asks for the edge with minimum weight on a path between two nodes. For the dynamic version of the problem on a tree, where the edge-weights can be updated, we give comparison-based and RAM data structures that achieve optimal query time. These structures support inserting a node on an edge, inserting a leaf, and contracting edges. When only insertion and deletion of leaves in a tree are needed, we give two data structures that achieve optimal and significantly lower query times than when updating the edge-weights is allowed. One is a semigroup structure for which the edge-weights are from an arbitrary semigroup and queries ask for the semigroup-sum of the edge-weights on a given path. For the other structure the edge-weights are given in the word RAM. We complement these upper bounds with lower bounds for different variants of the problem.

70 Gerth Stølting Brodal and Konstantinos Tsakalidis, *Dynamic Planar Range Maxima Queries*. In *Proc. 38th International Colloquium on Automata, Languages, and Programming*, Volume 6755 of Lecture Notes in Computer Science, pages 256–267. Springer Verlag, Berlin, 2011, doi: 10.1007/978-3-642-22006-7_22.

Abstract: We consider the dynamic two-dimensional maxima query problem. Let P be a set of n points in the plane. A point is *maximal* if it is not dominated by any other point in P. We describe two data structures that support the reporting of the t maximal points that dominate a given query point, and allow for insertions and deletions of points in P. In the pointer machine model we present a linear space data structure with $O(\log n+t)$ worst case query time and $O(\log n)$ worst case update time. This is the first dynamic data structure for the planar maxima dominance query problem that achieves these bounds in the worst case. The data structure also supports the more general query of reporting the maximal points among the points that lie in a given 3-sided orthogonal range unbounded from above in the same complexity. We can support 4-sided queries in $O(\log^2 n + t)$ worst case time, and $O(\log^2 n)$ worst case update time, using $O(n \log n)$ space, where t is the size of the output. This improves the worst case deletion time of the dynamic rectangular visibility query problem from $O(\log^3 n)$ to $O(\log^2 n)$. We adapt the data structure to the RAM model with word size w, where the coordinates of the points are integers in the range $U = \{0, \ldots, 2^w - 1\}$. We present a linear space data structure that supports 3-sided range maxima queries in $O(\frac{\log n}{\log \log n} + t)$ worst case time and updates in $O(\frac{\log n}{\log \log n})$ worst case time. These are the first sublogarithmic worst case bounds for all operations in the RAM model.

(21) Gerth Stølting Brodal, Mark Greve, Vineet Pandey and S. Srinivasa Rao, *Integer Representations towards Efficient Counting in the Bit Probe Model*. In *Proc. 8th Annual Conference on Theory and Applications of Models of Computation*, Volume 6648 of Lecture Notes in Computer Science, pages 206–217. Springer Verlag, Berlin, 2011, doi: 10.1007/978-3-642-20877-5_22.

Abstract: We consider the problem of representing numbers in close to optimal space and supporting increment, decrement, addition and subtraction operations efficiently. We study the problem in the bit probe model and analyse the number of bits read and written to perform the operations, both in the worst-case and in the average-case. A counter is *space-optimal* if it represents any number in the range $[0, \ldots, 2^n - 1]$ using exactly n bits. We provide a *space-optimal counter* which supports increment and decrement operations by reading at most $n - 1$ bits and writing at most

3 bits in the worst-case. To the best of our knowledge, this is the first such representation which supports these operations by always reading strictly less than n bits. For *redundant counters* where we only need to represent numbers in the range $[0, \ldots, L]$ for some integer $L < 2^n - 1$ using n bits, we define the efficiency of the counter as the ratio between $L+1$ and 2^n. We present various representations that achieve different trade-offs between the read and write complexities and the efficiency. We also give another representation of integers that uses $n + O(\log n)$ bits to represent integers in the range $[0, \ldots, 2^n - 1]$ that supports efficient addition and subtraction operations, improving the space complexity of an earlier representation by Munro and Rahman [Algorithmica, 2010].

71 Peyman Afshani, Gerth Stølting Brodal and Norbert Zeh, *Ordered and Unordered Top-K Range Reporting in Large Data Sets*. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 390–400, 2011, doi: 10.1137/1.9781611973082.31.

Abstract: We study the following problem: Given an array A storing N real numbers, preprocess it to allow fast reporting of the K smallest elements in the subarray $A[i, j]$ in sorted order, for any triple (i, j, K) with $1 \leq i \leq j \leq N$ and $1 \leq K \leq j - i + 1$. We are interested in scenarios where the array A is large, necessitating an I/O-efficient solution.

For a parameter f with $1 \leq f \leq \log_m n$, we construct a data structure that uses $O((N/f) \log_m n)$ space and achieves a query bound of $O(\log_B N + fK/B)$ I/Os, where B is the block size, M is the size of the main memory, $n := N/B$, and $m := M/B$. Our main contribution is to show that this solution is nearly optimal. To be precise, we show that achieving a query bound of $O(\log^\alpha n + fK/B)$ I/Os, for any constant α, requires $\Omega(N(f^{-1} \log_M n)(\log(f^{-1} \log_M n)))$ space, assuming $B = \Omega(\log N)$. For $M \geq B^{1+\varepsilon}$, this is within a $\log \log_m n$ factor of the upper bound. The lower bound assumes indivisibility of records and holds even if we assume K is always set to $j - 1 + 1$.

We also show that it is the requirement that the K smallest elements be reported in sorted order which makes the problem hard. If the K smallest elements in the query range can be reported in any order, then we can obtain a linear-size data structure with a query bound of $O(\log_B N + K/B)$ I/Os.

(17) Gerth Stølting Brodal, Spyros Sioutas, Kostas Tsichlas and Christos D. Zaroliagis, *D^2-Tree: A New Overlay with Deterministic Bounds*. In *Proc. 21th Annual International Symposium on Algorithms and Computation, Part II*, Volume 6507 of Lecture Notes in Computer Science, pages 1–12. Springer Verlag, Berlin, 2010, doi: 10.1007/978-3-642-17514-5_1.

Abstract: We present a new overlay, called the *Deterministic Decentralized tree* (D^2-tree). The D^2-tree compares favourably to other overlays for the following reasons: (a) it provides matching and better complexities, which are deterministic for the supported operations (b) the management of nodes (peers) and elements are completely decoupled from each other; and (c) an efficient deterministic load-balancing mechanism is presented for the uniform distribution of elements into nodes, while at the same time probabilistic optimal bounds are provided for the congestion of operations at the nodes.

72 Gerth Stølting Brodal, Casper Kejlberg-Rasmussen and Jakob Truelsen, *A Cache-Oblivious Implicit Dictionary with the Working Set Property*. In *Proc. ISAAC10, Part II*, Volume 6507 of Lecture Notes in Computer Science, pages 37–48. Springer Verlag, Berlin, 2010, doi: 10.1007/978-3-642-17514-5_4.

Abstract: In this paper we present an implicit dictionary with the working set property i.e. a dictionary supporting insert(e), delete(x) and predecessor(x) in $O(\log n)$ time and search(x) in $O(\log \ell)$ time, where n is the number of elements stored in the dictionary and ℓ is the number of distinct elements searched for since the element with key x was last searched for. The dictionary stores the elements in an array of size n using *no* additional space. In the cache-oblivious model the operations insert(e), delete(x) and predecessor(x) cause $O(\log_B n)$ cache-misses and search(x) causes $O(\log_B \ell)$ cache-misses.

(25) Gerth Stølting Brodal, Pooya Davoodi and S. Srinivasa Rao, *On Space Efficient Two Dimensional Range Minimum Data Structures*. In *Proc. 18th Annual European Symposium on Algorithms*, Volume 6347 of Lecture Notes in Computer Science, pages 171–182. Springer Verlag, Berlin, 2010, doi: 10.1007/978-3-642-15781-3_15 (presentation pdf, pptx).

Abstract: The two dimensional range minimum query problem is to preprocess a static two dimensional m by n array A of size $N = m \cdot n$, such that subsequent queries, asking for the position of the minimum element in a rectangular range within A, can be answered efficiently. We study the trade-off between the space and query time of the problem. We show that every algorithm enabled to access A during the query and using $O(N/c)$ bits additional space requires $\Omega(c)$ query time, for any c where $1 \leq c \leq N$. This lower bound holds for any dimension. In particular, for the one dimensional version of the problem, the lower bound is tight up to a constant factor. In two dimensions, we complement the lower bound with an indexing data structure of size $O(N/c)$ bits additional space which can be preprocessed in $O(N)$ time and achieves $O(c \log^2 c)$ query time. For $c = O(1)$, this is the first $O(1)$ query time algorithm using optimal $O(N)$ bits additional space. For the case where queries can not probe A, we give a data structure of size $O(N \cdot \min\{m, \log n\})$ bits with $O(1)$ query time, assuming $m \leq n$. This leaves a gap to the lower bound of $\Omega(N \log m)$ bits for this version of the problem.

73 Gerth Stølting Brodal, Erik D. Demaine, Jeremy T. Fineman, John Iacono, Stefan Langerman and J. Ian Munro, *Cache-Oblivious Dynamic Dictionaries with Optimal Update/Query Tradeoff*. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1448–1456, 2010, doi: 10.1137/1.9781611973075.117.

Abstract: Several existing cache-oblivious dynamic dictionaries achieve $O(\log_B N)$ (or slightly better $O(\log_B(N/M))$) memory transfers per operation, where N is the number of items stored, M is the memory size, and B is the block size, which matches the classic B-tree data structure. One recent structure achieves the same query bound and a sometimes-better amortized update bound of $O(1/B^{\Theta(1/(\log\log B)^2)} \cdot \log_B N + 1/B \cdot \log^2 N)$ memory transfers. This paper presents a new data structure, the *xDict*, implementing predecessor queries in $O(1/\epsilon \cdot \log_B(N/M))$ worst-case memory transfers and insertions and deletions in $O(1/(\epsilon B^{1-\epsilon}) \cdot \log_B(N/M))$ amortized memory transfers, for any constant ϵ with $0 < \epsilon < 1$. For example, the xDict achieves subconstant amortized update cost when $N = M\, B^{o(B^{1-\epsilon})}$, whereas the B-tree's $\Theta(\log_B(N/M))$ is subconstant only when $N = o(MB)$, and the previously obtained $\Theta((1/B^{\Theta(1/(\log\log B)^2)}) \log_B N + 1/B \cdot \log^2 N)$ is subconstant only when $N = o(2^{\sqrt{B}})$. The xDict attains the optimal tradeoff between insertions and queries, even in the broader external-memory model, for the range where inserts cost between $\Omega((1/B)\log^{1+\epsilon} N)$ and $O(1/\log^3 N)$ memory transfers.

(26) Gerth Stølting Brodal and Allan Grønlund Jørgensen, *Data Structures for Range Median Queries*. In *Proc. 20th Annual International Symposium on Algorithms and Computation*, Volume 5878 of Lecture Notes in Computer Science, pages 822–831. Springer Verlag, Berlin, 2009 2009, doi: 10.1007/978-3-642-10631-6_83.

Abstract: In this paper we design data structures supporting *range median* queries, i.e. report the median element in a sub-range of an array. We consider static and dynamic data structures and batched queries. Our data structures support *range selection* queries, which are more general, and dominance queries (*range rank*). In the static case our data structure uses linear space and queries are supported in $O(\log n/\log\log n)$ time. Our dynamic data structure uses $O(n \log n/\log\log n)$ space and supports queries and updates in $O((\log n/\log\log n)^2)$ time.

74 Gerth Stølting Brodal, Rolf Fagerberg, Mark Greve and Alejandro López-Ortiz, *Online Sorted Range Reporting*. In *Proc. 20th Annual International Symposium on Algorithms and Computation*, Volume 5878 of Lecture Notes in Computer Science, pages 173–182. Springer Verlag, Berlin, 2009 2009, doi: 10.1007/978-3-642-10631-6_19.

Abstract: We study the following one-dimensional range reporting problem: On an array A of n elements, support queries that given two indices $i \leq j$ and an integer k report the k smallest elements in the subarray $A[i..j]$ in sorted order. We present a data structure in the RAM model

supporting such queries in optimal $O(k)$ time. The structure uses $O(n)$ words of space and can be constructed in $O(n \log n)$ time. The data structure can be extended to solve the online version of the problem, where the elements in $A[i..j]$ are reported one-by-one in sorted order, in $O(1)$ worst-case time per element. The problem is motivated by (and is a generalization of) a problem with applications in search engines: On a tree where leaves have associated rank values, report the highest ranked leaves in a given subtree. Finally, the problem studied generalizes the classic range minimum query (RMQ) problem on arrays.

(20) Gerth Stølting Brodal, Alexis C. Kaporis, Spyros Sioutas, Konstantinos Tsakalidis and Kostas Tsichlas, *Dynamic 3-sided Planar Range Queries with Expected Doubly Logarithmic Time*. In *Proc. 20th Annual International Symposium on Algorithms and Computation*, Volume 5878 of Lecture Notes in Computer Science, pages 193–202. Springer Verlag, Berlin, 2009 2009, doi: 10.1007/978-3-642-10631-6_21.

Abstract: We consider the problem of maintaining dynamically a set of points in the plane and supporting range queries of the type $[a, b] \times (-\infty, c]$. We assume that the inserted points have their x-coordinates drawn from a class of *smooth* distributions, whereas the y-coordinates are arbitrarily distributed. The points to be deleted are selected uniformly at random among the inserted points. For the RAM model, we present a linear space data structure that supports queries in $O(\log \log n+t)$ expected time with high probability and updates in $O(\log \log n)$ expected amortized time, where n is the number of points stored and t is the size of the output of the query. For the I/O model we support queries in $O(\log \log_B n + t/B)$ expected I/Os with high probability and updates in $O(\log_B \log n)$ expected amortized I/Os using linear space, where B is the disk block size. The data structures are deterministic and the expectation is with respect to the input distribution.

75 Gerth Stølting Brodal, Allan Grønlund Jørgensen, Gabriel Moruz and Thomas Mølhave, *Counting in the Presence of Memory Faults*. In *Proc. 20th Annual International Symposium on Algorithms and Computation*, Volume 5878 of Lecture Notes in Computer Science, pages 842–851. Springer Verlag, Berlin, 2009 2009, doi: 10.1007/978-3-642-10631-6_85.

Abstract: The faulty memory RAM presented by Finocchi and Italiano is a variant of the RAM model where the content of any memory cell can get corrupted at any time, and corrupted cells cannot be distinguished from uncorrupted cells. An upper bound, δ, on the number of corruptions and $O(1)$ reliable memory cells are provided.

In this paper we investigate the fundamental problem of counting in faulty memory. Keeping many reliable counters in the faulty memory is easily done by replicating the value of each counter $\Theta(\delta)$ times and paying $\Theta(\delta)$ time every time a counter is queried or incremented. In this paper we decrease the expensive increment cost to $o(\delta)$ and present upper and lower bound tradeoffs decreasing the increment time at the cost of the accuracy of the counters.

76 Gerth Stølting Brodal, Allan Grønlund Jørgensen and Thomas Mølhave, *Fault Tolerant External Memory Algorithms*. In *Proc. 11th International Workshop on Algorithms and Data Structures*, Volume 5664 of Lecture Notes in Computer Science, pages 411–422. Springer Verlag, Berlin, 2009, doi: 10.1007/978-3-642-03367-4_36.

Abstract: Algorithms dealing with massive data sets are usually designed for I/O-efficiency, often captured by the I/O model by Aggarwal and Vitter. Another aspect of dealing with massive data is how to deal with memory faults, *e.g.* captured by the adversary based faulty memory RAM by Finocchi and Italiano. However, current fault tolerant algorithms do not scale beyond the internal memory. In this paper we investigate for the first time the connection between I/O-efficiency in the I/O model and fault tolerance in the faulty memory RAM, and we assume that both memory and disk are unreliable. We show a lower bound on the number of I/Os required for any deterministic dictionary that is resilient to memory faults. We design a static and a dynamic deterministic dictionary with optimal query performance as well as an optimal sorting algorithm and an optimal priority queue. Finally, we consider scenarios where only cells in memory or only cells on disk are corruptible and separate randomized and deterministic dictionaries in the latter.

77 Gerth Stølting Brodal and Allan Grønlund Jørgensen, *Selecting Sums in Arrays*. In *Proc. 19th*

Annual International Symposium on Algorithms and Computation, Volume 5369 of Lecture Notes in Computer Science, pages 100–111. Springer Verlag, Berlin, 2008, doi: 10.1007/978-3-540-92182-0_12.

Abstract: In an array of n numbers each of the $\binom{n}{2}+n$ contiguous subarrays define a sum. In this paper we focus on algorithms for selecting and reporting maximal sums from an array of numbers. First, we consider the problem of reporting k subarrays inducing the k largest sums among all subarrays of length at least l and at most u. For this problem we design an optimal $O(n+k)$ time algorithm. Secondly, we consider the problem of selecting a subarray storing the k'th largest sum. For this problem we prove a time bound of $\Theta(n \cdot \max\{1, \log(k/n)\})$ by describing an algorithm with this running time and by proving a matching lower bound. Finally, we combine the ideas and obtain an $O(n \cdot \max\{1, \log(k/n)\})$ time algorithm that selects a subarray storing the k'th largest sum among all subarrays of length at least l and at most u.

(24) Lars Arge, Gerth Stølting Brodal and S. Srinivasa Rao, *External memory planar point location with logarithmic updates*. In *Proc. 24st Annual ACM Symposium on Computational Geometry*, pages 139–147, 2008, doi: 10.1145/1377676.1377699.

Abstract: Point location is an extremely well-studied problem both in internal memory models and recently also in the external memory model. In this paper, we present an I/O-efficient dynamic data structure for point location in general planar subdivisions. Our structure uses linear space to store a subdivision with N segments. Insertions and deletions of segments can be performed in amortized $O(\log_B N)$ I/Os and queries can be answered in $O(\log_B^2 N)$ I/Os in the worst-case. The previous best known linear space dynamic structure also answers queries in $O(\log_B^2 N)$ I/Os, but only supports insertions in amortized $O(\log_B^2 N)$ I/Os. Our structure is also considerably simpler than previous structures.

78 Michael Westergaard, Lars Michael Kristensen, Gerth Stølting Brodal and Lars Arge, *The ComBack Method - Extending Hash Compaction with Backtracking*. In *Proc. 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007*, Volume 4546 of Lecture Notes in Computer Science, pages 445–464. Springer Verlag, Berlin, 2007 2007, doi: 10.1007/978-3-540-73094-1_26.

Abstract: This paper presents the ComBack method for explicit state space exploration. The ComBack method extends the well-known hash compaction method such that full coverage of the state space is guaranteed. Each encountered state is mapped into a compressed state descriptor (hash value) as in hash compaction. The method additionally stores for each state an integer representing the identity of the state and a backedge to a predecessor state. This allows hash collisions to be resolved on-the-fly during state space exploration using backtracking to reconstruct the full state descriptors when required for comparison with newly encountered states. A prototype implementation of the ComBack method is used to evaluate the method on several example systems and compare its performance to related methods. The results show a reduction in memory usage at an acceptable cost in exploration time.

79 Gerth Stølting Brodal, Rolf Fagerberg, Irene Finocchi, Fabrizio Grandoni, Giuseppe Italiano, Allan Grønlund Jørgensen, Gabriel Moruz and Thomas Mølhave, *Optimal Resilient Dynamic Dictionaries*. In *Proc. 15th Annual European Symposium on Algorithms*, Volume 4708 of Lecture Notes in Computer Science, pages 347–358. Springer Verlag, Berlin, 2007, doi: 10.1007/978-3-540-75520-3_32.

Abstract: We investigate the problem of computing in the presence of faults that may arbitrarily (i.e., adversarially) corrupt memory locations. In the faulty memory model, any memory cell can get corrupted at any time, and corrupted cells cannot be distinguished from uncorrupted ones. An upper bound δ on the number of corruptions and $O(1)$ reliable memory cells are provided. In this model, we focus on the design of resilient dictionaries, i.e., dictionaries which are able to operate correctly (at least) on the set of uncorrupted keys. We first present a simple resilient dynamic search tree, based on random sampling, with $O(\log n + \delta)$ expected amortized cost per operation, and $O(n)$ space complexity. We then propose an optimal deterministic static dictionary supporting searches in $\Theta(\log n + \delta)$ time in the worst case, and we show how to use it in a dynamic setting in

order to support updates in $O(\log n + \delta)$ amortized time. Our dynamic dictionary also supports range queries in $O(\log n + \delta + t)$ worst case time, where t is the size of the output. Finally, we show that every resilient search tree (with some reasonable properties) must take $\Omega(\log n + \delta)$ worst-case time per search.

80 Gerth Stølting Brodal, Loukas Georgiadis, Kristoffer A. Hansen and Irit Katriel, *Dynamic Matchings in Convex Bipartite Graphs*. In *Proc. 32nd International Symposium on Mathematical Foundations of Computer Science*, Volume 4708 of Lecture Notes in Computer Science, pages 406–417. Springer Verlag, Berlin, 2007, doi: 10.1007/978-3-540-74456-6_37 (presentation pdf, ppt).

Abstract: We consider the problem of maintaining a maximum matching in a convex bipartite graph $G = (V, E)$ under a set of update operations which includes insertions and deletions of vertices and edges. It is not hard to show that it is impossible to maintain an explicit representation of a maximum matching in sub-linear time per operation, even in the amortized sense. Despite this difficulty, we develop a data structure which maintains the set of vertices that participate in a maximum matching in $O(\log^2 |V|)$ amortized time per update and reports the status of a vertex (matched or unmatched) in constant worst-case time. Our structure can report the mate of a matched vertex in the maximum matching in worst-case $O(\min\{k \log^2 |V| + \log |V|, |V| \log |V|\})$ time, where k is the number of update operations since the last query for the same pair of vertices was made. In addition, we give an $O(\sqrt{|V|} \log^2 |V|)$-time amortized bound for this pair query.

81 Gerth Stølting Brodal and Allan Grønlund Jørgensen, *A Linear Time Algorithm for the k Maximal Sums Problem*. In *Proc. 32nd International Symposium on Mathematical Foundations of Computer Science*, Volume 4708 of Lecture Notes in Computer Science, pages 442–453. Springer Verlag, Berlin, 2007, doi: 10.1007/978-3-540-74456-6_40 (presentation pdf, ppt).

Abstract: Finding the sub-vector with the largest sum in a sequence of n numbers is known as the maximum sum problem. Finding the k sub-vectors with the largest sums is a natural extension of this, and is known as the k maximal sums problem. In this paper we design an optimal $O(n + k)$ time algorithm for the k maximal sums problem. We use this algorithm to obtain algorithms solving the two-dimensional k maximal sums problem in $O(m^2 \cdot n + k)$ time, where the input is an $m \times n$ matrix with $m \leq n$. We generalize this algorithm to solve the d-dimensional problem in $O(n^{2d-1} + k)$ time. The space usage of all the algorithms can be reduced to $O(n^{d-1} + k)$. This leads to the first algorithm for the k maximal sums problem in one dimension using $O(n + k)$ time and $O(k)$ space.

(29) Michael A. Bender, Gerth Stølting Brodal, Rolf Fagerberg, Riko Jacob and Elias Vicari, *Optimal Sparse Matrix Dense Vector Multiplication in the I/O-Model*. In *Proc. 19th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 61–70, 2007, doi: 10.1145/1248377.1248391.

Abstract: We analyze the problem of sparse-matrix dense-vector multiplication (SpMV) in the I/O-model. The task of SpMV is to compute $y := Ax$, where A is a sparse $N \times N$ matrix and x and y are vectors. Here, sparsity is expressed by the parameter k that states that A has a total of at most kN nonzeros, i.e., an average number of k nonzeros per column. The extreme choices for parameter k are well studied special cases, namely for $k = 1$ permuting and for $k = N$ dense matrix-vector multiplication.

We study the worst-case complexity of this computational task, i.e., what is the best possible upper bound on the number of I/Os depending on k and N only. We determine this complexity up to a constant factor for large ranges of the parameters. By our arguments, we find that most matrices with kN nonzeros require this number of I/Os, even if the program may depend on the structure of the matrix. The model of computation for the lower bound is a combination of the I/O-models of Aggarwal and Vitter, and of Hong and Kung.

We study two variants of the problem, depending on the memory layout of A. If A is stored in column major layout, SpMV has I/O complexity $\Theta(\min\{kN/B \cdot (1+\log_{M/B}(N/\max\{M,k\})), kN\})$ for $k \leq N^{1-\varepsilon}$ and any constant $1 > \varepsilon > 0$. If the algorithm can choose the memory layout, the I/O complexity of SpMV is $\Theta(\min\{kN/B \cdot (1 + \log_{M/B}(N/(kM))), kN\})$ for $k \leq N^{1/3}$.

In the cache oblivious setting with tall cache assumption $M \geq B^{1+\varepsilon}$, the I/O complexity is $OkN/B \cdot (1 + \log_{M/B}(N/k))$ for A in column major layout.

82 Martin Stissing, Christian Nørgaard Storm Pedersen, Thomas Mailund, Gerth Stølting Brodal and Rolf Fagerberg, *Computing the Quartet Distance Between Evolutionary Trees of Bounded Degree*. In *Proc. 5th Asia Pacific Bioinformatics Conference*, Volume 5 of Advances in Bioinformatics & Computational Biology, pages 101–110. Imperial College Press, 2007, doi: 10.1142/9781860947995_0013.

Abstract: We present an algorithm for calculating the quartet distance between two evolutionary trees of bounded degree on a common set of n species. The previous best algorithm has running time $O(d^2n^2)$ when considering trees, where no node is of more than degree d. The algorithm developed herein has running time $O(d^9n\log n))$ which makes it the first algorithm for computing the quartet distance between non-binary trees which has a sub-quadratic worst case running time.

(30) Martin Stissing, Thomas Mailund, Christian Nørgaard Storm Pedersen, Gerth Stølting Brodal and Rolf Fagerberg, *Computing the All-Pairs Quartet Distance on a set of Evolutionary Trees*. In *Proc. 5th Asia Pacific Bioinformatics Conference*, Advances in Bioinformatics & Computational Biology, pages 91–100. Imperial College Press, 2007, doi: 10.1142/9781860947995_0012.

Abstract: We present two algorithms for calculating the quartet distance between all pairs of trees in a set of binary evolutionary trees on a common set of species. The algorithms exploit common substructure among the trees to speed up the pairwise distance calculations thus performing significantly better on large sets of trees compared to performing distinct pairwise distance calculations, as we illustrate experimentally, where we see a speedup factor of around 130 in the best case.

83 Lars Arge, Gerth Stølting Brodal and Loukas Georgiadis, *Improved Dynamic Planar Point Location*. In *Proc. 47th Annual Symposium on Foundations of Computer Science*, pages 305–314, 2006, doi: 10.1109/FOCS.2006.40.

Abstract: We develop the first linear-space data structures for dynamic planar point location in general subdivisions that achieve logarithmic query time and poly-logarithmic update time.

84 Gerth Stølting Brodal, Christos Makris and Kostas Tsichlas, *Purely Functional Worst Case Constant Time Catenable Sorted Lists*. In *Proc. 14th Annual European Symposium on Algorithms*, Volume 4168 of Lecture Notes in Computer Science, pages 172–183. Springer Verlag, Berlin, 2006, doi: 10.1007/11841036_18 (presentation pdf, ppt).

Abstract: We present a purely functional implementation of search trees that requires $O(\log n)$ time for search and update operations and supports the join of two trees in worst case constant time. Hence, we solve an open problem posed by Kaplan and Tarjan as to whether it is possible to envisage a data structure supporting simultaneously the join operation in $O(1)$ time and the search and update operations in $O(\log n)$ time.

85 Gerth Stølting Brodal and Gabriel Moruz, *Skewed Binary Search Trees*. In *Proc. 14th Annual European Symposium on Algorithms*, Volume 4168 of Lecture Notes in Computer Science, pages 708–719. Springer Verlag, Berlin, 2006, doi: 10.1007/11841036_63 (presentation pdf, zip).

Abstract: It is well-known that to minimize the number of comparisons a binary search tree should be perfectly balanced. Previous work has shown that a dominating factor over the running time for a search is the number of cache faults performed, and that an appropriate memory layout of a binary search tree can reduce the number of cache faults by several hundred percent. Motivated by the fact that during a search branching to the left or right at a node does not necessarily have the same cost, e.g. because of branch prediction schemes, we in this paper study the class of skewed binary search trees. For all nodes in a skewed binary search tree the ratio between the size of the left subtree and the size of the tree is a fixed constant (a ratio of 1/2 gives perfect balanced trees). In this paper we present an experimental study of various memory layouts of static skewed binary search trees, where each element in the tree is accessed with a uniform probability. Our results show that for many of the memory layouts we consider skewed binary search trees can perform better than perfect balanced search trees. The improvements in the running time are on the order of 15%.

(27) Gerth Stølting Brodal, Kanela Kaligosi, Irit Katriel and Martin Kutz, *Faster Algorithms for Computing Longest Common Increasing Subsequences*. In *Proc. 17th Annual Symposium on Combinatorial Pattern Matching*, Volume 4009 of Lecture Notes in Computer Science, pages 330–341. Springer Verlag, Berlin, 2006, doi: 10.1007/11780441_30.

Abstract: We present algorithms for finding a longest common increasing subsequence of two or more input sequences. For two sequences of lengths m and n, where $m \geq n$, we present an algorithm with an output-dependent expected running time of $O((m + n\ell) \log\log \sigma + Sort)$ and $O(m)$ space, where ℓ is the length of an LCIS, σ is the size of the alphabet, and *Sort* is the time to sort each input sequence. For $k \geq 3$ length-n sequences we present an algorithm which improves the previous best bound by more than a factor k for many inputs. In both cases, our algorithms are conceptually quite simple but rely on existing sophisticated data structures. Finally, we introduce the problem of longest common weakly-increasing (or non-decreasing) subsequences (LCWIS), for which we present an $O(m + n \log n)$-time algorithm for the 3-letter alphabet case. For the extensively studied longest common subsequence problem, comparable speedups have not been achieved for small alphabets.

86 Gerth Stølting Brodal and Rolf Fagerberg, *Cache-oblivious String Dictionaries*. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 581–590, 2006, doi: 10.1145/1109557.1109621 (presentation pdf, zip).

Abstract: We present static cache-oblivious dictionary structures for strings which provide analogues of tries and suffix trees in the cache-oblivious model. Our construction takes as input either a set of strings to store, a single string for which all suffixes are to be stored, a trie, a compressed trie, or a suffix tree, and creates a cache-oblivious data structure which performs prefix queries in $O(\log_B n + |P|/B)$ I/Os, where n is the number of leaves in the trie, P is the query string, and B is the block size. This query cost is optimal for unbounded alphabets. The data structure uses linear space.

87 Gerth Stølting Brodal and Gabriel Moruz, *Tradeoffs Between Branch Mispredictions and Comparisons for Sorting Algorithms*. In *Proc. 9th International Workshop on Algorithms and Data Structures, Waterloo, Canada, 15–17 August 2005*, Volume 3608 of Lecture Notes in Computer Science, pages 385–395. Springer Verlag, Berlin, 2005, doi: 10.1007/11534273_34.

Abstract: Branch mispredictions is an important factor affecting the running time in practice. In this paper we consider tradeoffs between the number of branch mispredictions and the number of comparisons for sorting algorithms in the comparison model. We prove that a sorting algorithm using $O(dn \log n)$ comparisons performs $\Omega(n \log_d n)$ branch mispredictions. We show that Multiway MergeSort achieves this tradeoff by adopting a multiway merger with a low number of branch mispredictions. For adaptive sorting algorithms we similarly obtain that an algorithm performing $O(dn(1 + \log(1 + \mathrm{Inv}/n)))$ comparisons must perform $\Omega(n \log_d(1 + \mathrm{Inv}/n))$ branch mispredictions, where Inv is the number of inversions in the input. This tradeoff can be achieved by GenericSort by Estivill-Castro and Wood by adopting a multiway division protocol and a multiway merging algorithm with a low number of branch mispredictions.

88 Gerth Stølting Brodal, Rolf Fagerberg and Gabriel Moruz, *Cache-Aware and Cache-Oblivious Adaptive Sorting*. In *Proc. 32nd International Colloquium on Automata, Languages, and Programming*, Volume 3580 of Lecture Notes in Computer Science, pages 576–588. Springer Verlag, Berlin, 2005, doi: 10.1007/11523468_47.

Abstract: Two new adaptive sorting algorithms are introduced which perform an optimal number of comparisons with respect to the number of inversions in the input. The first algorithm is based on a new linear time reduction to (non-adaptive) sorting. The second algorithm is based on a new division protocol for the GenericSort algorithm by Estivill-Castro and Wood. From both algorithms we derive I/O-optimal cache-aware and cache-oblivious adaptive sorting algorithms. These are the first I/O-optimal adaptive sorting algorithms.

89 Lars Arge, Gerth Stølting Brodal, Rolf Fagerberg and Morten Laustsen, *Cache-Oblivious Planar Orthogonal Range Searching and Counting*. In *Proc. 21st Annual ACM Symposium on Computational Geometry*, pages 160–169, 2005, doi: 10.1145/1064092.1064119.

Abstract: We present the first cache-oblivious data structure for planar orthogonal range counting, and improve on previous results for cache-oblivious planar orthogonal range searching.

Our range counting structure uses $O(N \log_2 N)$ space and answers queries using $O(\log_B N)$ memory transfers, where B is the block size of any memory level in a multilevel memory hierarchy. Using bit manipulation techniques, the space can be further reduced to $O(N)$. The structure can also be modified to support more general semigroup range sum queries in $O(\log_B N)$ memory transfers, using $O(N \log_2 N)$ space for three-sided queries and $O(N \log_2^2 N / \log_2 \log_2 N)$ space for four-sided queries.

Based on the $O(N \log N)$ space range counting structure, we develop a data structure that uses $O(N \log_2 N)$ space and answers three-sided range queries in $O(\log_B N + T/B)$ memory transfers, where T is the number of reported points. Based on this structure, we present a general four-sided range searching structure that uses $O(N \log_2^2 N / \log_2 \log_2 N)$ space and answers queries in $O(\log_B N + T/B)$ memory transfers.

(31) Gerth Stølting Brodal, Rolf Fagerberg and Gabriel Moruz, *On the Adaptiveness of Quicksort*. In *Proc. 7th Workshop on Algorithm Engineering and Experiments*, pages 130–140, 2005, doi: www.siam.org/meetings/alenex05/papers/12gbrodal.pdf (presentation pdf).

Abstract: Quicksort was first introduced in 1961 by Hoare. Many variants have been developed, the best of which are among the fastest generic sorting algorithms available, as testified by the choice of Quicksort as the default sorting algorithm in most programming libraries. Some sorting algorithms are adaptive, i.e. they have a complexity analysis which is better for inputs which are nearly sorted, according to some specified measure of presortedness. Quicksort is not among these, as it uses $\Omega(n \log n)$ comparisons even when the input is already sorted. However, in this paper we demonstrate empirically that the actual running time of Quicksort *is* adaptive with respect to the presortedness measure Inv. Differences close to a factor of two are observed between instances with low and high Inv value. We then show that for the randomized version of Quicksort, the number of element *swaps* performed is *provably* adaptive with respect to the measure Inv. More precisely, we prove that randomized Quicksort performs expected $O(n(1+\log(1+\mathrm{Inv}/n)))$ element swaps, where Inv denotes the number of inversions in the input sequence. This result provides a theoretical explanation for the observed behavior, and gives new insights on the behavior of the Quicksort algorithm. We also give some empirical results on the adaptive behavior of Heapsort and Mergesort.

90 Gerth Stølting Brodal, *Cache-Oblivious Algorithms and Data Structures*. In *Proc. 9th Scandinavian Workshop on Algorithm Theory*, Volume 3111 of Lecture Notes in Computer Science, pages 3–13. Springer Verlag, Berlin, 2004, doi: 10.1007/978-3-540-27810-8_2 (presentation pdf, zip).

Abstract: Frigo, Leiserson, Prokop and Ramachandran in 1999 introduced the ideal-cache model as a formal model of computation for developing algorithms in environments with multiple levels of caching, and coined the terminology of *cache-oblivious algorithms*. Cache-oblivious algorithms are described as standard RAM algorithms with only one memory level, i.e. without any knowledge about memory hierarchies, but are analyzed in the two-level I/O model of Aggarwal and Vitter for an arbitrary memory and block size and an optimal off-line cache replacement strategy. The result are algorithms that automatically apply to multi-level memory hierarchies. This paper gives an overview of the results achieved on cache-oblivious algorithms and data structures since the seminal paper by Frigo *et al.*

91 Gerth Stølting Brodal, Rolf Fagerberg, Ulrich Meyer and Norbert Zeh, *Cache-Oblivious Data Structures and Algorithms for Undirected Breadth-First Search and Shortest Paths*. In *Proc. 9th Scandinavian Workshop on Algorithm Theory*, Volume 3111 of Lecture Notes in Computer Science, pages 480–492. Springer Verlag, Berlin, 2004, doi: 10.1007/978-3-540-27810-8_41.

Abstract: We present improved cache-oblivious data structures and algorithms for breadth-first search and the single-source shortest path problem on undirected graphs with non-negative edge

weights. Our results removes the performance gap between the currently best *cache-aware* algorithms for these problems and their *cache-oblivious* counterparts. Our shortest-path algorithm relies on a new data structure, called *bucket heap*, which is the first cache-oblivious priority queue to efficiently support a weak DecreaseKey operation.

(33) Gerth Stølting Brodal, Rolf Fagerberg and Kristoffer Vinther, *Engineering a Cache-Oblivious Sorting Algorithm*. In *Proc. 6th Workshop on Algorithm Engineering and Experiments*, pages 4–17, 2004, doi: www.siam.org/meetings/alenex04/abstacts/alenex04.pdf (presentation pdf, zip).

Abstract: This paper is an algorithmic engineering study of cache-oblivious sorting. We investigate a number of implementation issues and parameter choices for the cache-oblivious sorting algorithm Lazy Funnelsort by empirical methods, and compare the final algorithm with Quicksort, the established standard for comparison based sorting, as well as with recent cache-aware proposals.

The main result is a carefully implemented cache-oblivious sorting algorithm, which our experiments show can be faster than the best Quicksort implementation we can find, already for input sizes well within the limits of RAM. It is also at least as fast as the recent cache-aware implementations included in the test. On disk the difference is even more pronounced regarding Quicksort and the cache-aware algorithms, whereas the algorithm is slower than a careful implementation of multiway Mergesort such as TPIE.

Source code available at: Engineering Cache-Oblivious Sorting Algorithms, Kristoffer Vinther. Master's Thesis, Department of Computer Science, Aarhus University, June 2003.

92 Gerth Stølting Brodal and Riko Jacob, *Time-Dependent Networks as Models to Achieve Fast Exact Time-Table Queries*. In *Proc. Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003)*, Volume 92 of Electronic Notes in Theoretical Computer Science(1), 12 pages. Elsevier Science, 2003, doi: 10.1016/j.entcs.2003.12.019.

Abstract: We consider efficient algorithms for exact time-table queries, i.e. algorithms that find optimal itineraries for travelers using a train system. We propose to use time-dependent networks as a model and show advantages of this approach over space-time networks as models.

(28) Michael A. Bender, Gerth Stølting Brodal, Rolf Fagerberg, Dongdong Ge, Simai He, Haodong Hu, John Iacono and Alejandro López-Ortiz, *The Cost of Cache-Oblivious Searching*. In *Proc. 44th Annual Symposium on Foundations of Computer Science*, pages 271–282, 2003, doi: 10.1109/SFCS.2003.1238201.

Abstract: Tight bounds on the cost of cache-oblivious searching are proved. It is shown that no cache-oblivious search structure can guarantee that a search performs fewer than $\lg e \log_B N$ block transfers between any two levels of the memory hierarchy. This lower bound holds even if all of the block sizes are limited to be powers of 2. A modified version of the van Emde Boas layout is proposed, whose expected block transfers between any two levels of the memory hierarchy arbitrarily close to $[\lg e + O(\lg\lg B/\lg B)]\log_B N + O(1)$. This factor approaches $\lg e \approx 1.443$ as B increases. The expectation is taken over the random placement of the first element of the structure in memory.

As searching in the Disk Access Model (DAM) can be performed in $\log_B N + 1$ block transfers, this result shows a separation between the 2-level DAM and cache-oblivious memory-hierarchy models. By extending the DAM model to k levels, multilevel memory hierarchies can be modelled. It is shown that as k grows, the search costs of the optimal k-level DAM search structure and of the optimal cache-oblivious search structure rapidly converge. This demonstrates that for a multilevel memory hierarchy, a simple cache-oblivious structure almost replicates the performance of an optimal parameterized k-level DAM structure.

93 Gerth Stølting Brodal, Rolf Fagerberg, Anna Östlin, Christian Nørgaard Storm Pedersen and S. Srinivasa Rao, *Computing Refined Buneman Trees in Cubic Time*. In *Proc. 3rd Workshop on Algorithms in BioInformatics*, Volume 2812 of Lecture Notes in Computer Science, pages 259–270. Springer Verlag, Berlin, 2003, doi: 10.1007/978-3-540-39763-2_20.

Abstract: Reconstructing the evolutionary tree for a set of n species based on pairwise distances

between the species is a fundamental problem in bioinformatics. Neighbor joining is a popular distance based tree reconstruction method. It always proposes fully resolved binary trees despite missing evidence in the underlying distance data. Distance based methods based on the theory of Buneman trees and refined Buneman trees avoid this problem by only proposing evolutionary trees whose edges satisfy a number of constraints. These trees might not be fully resolved but there is strong combinatorial evidence for each proposed edge. The currently best algorithm for computing the refined Buneman tree from a given distance measure has a running time of $O(n^5)$ and a space consumption of $O(n^4)$. In this paper, we present an algorithm with running time $O(n^3)$ and space consumption $O(n^2)$. The improved complexity of our algorithm makes the method of refined Buneman trees computational competitive to methods based on neighbor joining.

94 Gerth Stølting Brodal and Rolf Fagerberg, *On the Limits of Cache-Obliviousness*. In *Proc. 35th Annual ACM Symposium on Theory of Computing*, pages 307–315, 2003, doi: 10.1145/780542.780589.

Abstract: In this paper, we present lower bounds for permuting and sorting in the cache-oblivious model. We prove that (1) I/O optimal cache-oblivious comparison based sorting is not possible without a tall cache assumption, and (2) there does not exist an I/O optimal cache-oblivious algorithm for permuting, not even in the presence of a tall cache assumption.

Our results for sorting show the existence of an inherent trade-off in the cache-oblivious model between the strength of the tall cache assumption and the overhead for the case $M \gg B$, and show that Funnelsort and recursive binary mergesort are optimal algorithms in the sense that they attain this trade-off.

95 Gerth Stølting Brodal and Rolf Fagerberg, *Lower Bounds for External Memory Dictionaries*. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 546–554, 2003, doi: doi.acm.org/10.1145/644108.644201 (presentation pdf, zip).

Abstract: We study trade-offs between the update time and the query time for comparison based external memory dictionaries. The main contributions of this paper are two lower bound trade-offs between the I/O complexity of member queries and insertions: If $N > M$ insertions perform at most $\delta \cdot N/B$ I/Os, then (1) there exists a query requiring $N/(M \cdot (M/B)^{O(\delta)})$ I/Os, and (2) there exists a query requiring $\Omega(\log_{\delta \log^2 N}(N/M))$ I/Os when δ is $O(B/\log^3 N)$ and N is at least M^2. For both lower bounds we describe data structures which give matching upper bounds for a wide range of parameters, thereby showing the lower bounds to be tight within these ranges.

96 Gerth Stølting Brodal and Rolf Fagerberg, *Funnel Heap - A Cache Oblivious Priority Queue*. In *Proc. 13th Annual International Symposium on Algorithms and Computation*, Volume 2518 of Lecture Notes in Computer Science, pages 219–228. Springer Verlag, Berlin, 2002 2002, doi: 10.1007/3-540-36136-7_20 (presentation pdf, zip).

Abstract: The cache oblivious model of computation is a two-level memory model with the assumption that the parameters of the model are unknown to the algorithms. A consequence of this assumption is that an algorithm efficient in the cache oblivious model is automatically efficient in a multi-level memory model. Arge et al. recently presented the first optimal cache oblivious priority queue, and demonstrated the importance of this result by providing the first cache oblivious algorithms for graph problems. Their structure uses cache oblivious sorting and selection as subroutines. In this paper, we devise an alternative optimal cache oblivious priority queue based only on binary merging. We also show that our structure can be made adaptive to different usage profiles.

97 Gerth Stølting Brodal and Riko Jacob, *Dynamic Planar Convex Hull*. In *Proc. 43rd Annual Symposium on Foundations of Computer Science*, pages 617–626, 2002, doi: 10.1109/SFCS.2002.1181985.

Abstract: In this paper we determine the computational complexity of the dynamic convex hull problem in the planar case. We present a data structure that maintains a finite set of n points in the plane under insertion and deletion of points in amortized O(log n) time per operation. The

space usage of the data structure is O(n). The data structure supports extreme point queries in a given direction, tangent queries through a given point, and queries for the neighboring points on the convex hull in O(log n) time. The extreme point queries can be used to decide whether or not a given line intersects the convex hull, and the tangent queries to determine whether a given point is inside the convex hull. We give a lower bound on the amortized asymptotic time complexity that matches the performance of this data structure.

98 Stephen Alstrup, Gerth Stølting Brodal, Inge Li Gørtz and Theis Rauhe, *Time and Space Efficient Multi-Method Dispatching*. In *Proc. 8th Scandinavian Workshop on Algorithm Theory*, Volume 2368 of Lecture Notes in Computer Science, pages 20–29. Springer Verlag, Berlin, 2002, doi: 10.1007/3-540-45471-3_3.

Abstract: The *dispatching problem* for object oriented languages is the problem of determining the most specialized method to invoke for calls at run-time. This can be a critical component of execution performance. A number of recent results, including [Muthukrishnan and Müller SODA'96, Ferragina and Muthukrishnan ESA'96, Alstrup et al. FOCS'98], have studied this problem and in particular provided various efficient data structures for the *mono-method* dispatching problem. A recent paper of Ferragina, Muthukrishnan and de Berg [STOC'99] addresses the *multi-method* dispatching problem.

Our main result is a linear space data structure for *binary* dispatching that supports dispatching in logarithmic time. Using the same query time as Ferragina et al., this result improves the space bound with a logarithmic factor.

99 Gerth Stølting Brodal and Rolf Fagerberg, *Cache Oblivious Distribution Sweeping*. In *Proc. 29th International Colloquium on Automata, Languages, and Programming*, Volume 2380 of Lecture Notes in Computer Science, pages 426–438. Springer Verlag, Berlin, 2002, doi: 10.1007/3-540-45465-9_37.

Abstract: We adapt the distribution sweeping method to the cache oblivious model. Distribution sweeping is the name used for a general approach for divide-and-conquer algorithms where the combination of solved subproblems can be viewed as a merging process of streams. We demonstrate by a series of algorithms for specific problems the feasibility of the method in a cache oblivious setting. The problems all come from computational geometry, and are: orthogonal line segment intersection reporting, the all nearest neighbors problem, the 3D maxima problem, computing the measure of a set of axis-parallel rectangles, computing the visibility of a set of line segments from a point, batched orthogonal range queries, and reporting pairwise intersections of axis-parallel rectangles. Our basic building block is a simplified version of the cache oblivious sorting algorithm Funnelsort of Frigo et al., which is of independent interest.

100 Gerth Stølting Brodal, Rune Bang Lyngsø, Anna Östlin and Christian Nørgaard Storm Pedersen, *Solving the String Statistics Problem in Time $O(n \log n)$*. In *Proc. 29th International Colloquium on Automata, Languages, and Programming*, Volume 2380 of Lecture Notes in Computer Science, pages 728–739. Springer Verlag, Berlin, 2002, doi: 10.1007/3-540-45465-9_62.

Abstract: The string statistics problem consists of preprocessing a string of length n such that given a query pattern of length m, the maximum number of non-overlapping occurrences of the query pattern in the string can be reported efficiently. Apostolico and Preparata introduced the minimal augmented suffix tree (MAST) as a data structure for the string statistics problem, and showed how to construct the MAST in time $O(n \log^2 n)$ and how it supports queries in time $O(m)$ for constant sized alphabets. A subsequent theorem by Fraenkel and Simpson stating that a string has at most a linear number of distinct squares implies that the MAST requires space $O(n)$. In this paper we improve the construction time for the MAST to $O(n \log n)$ by extending the algorithm of Apostolico and Preparata to exploit properties of efficient joining and splitting of search trees together with a refined analysis.

(38) Gerth Stølting Brodal, George Lagogiannis, Christos Makris, Athanasios Tsakalidis and Kostas Tsichlas, *Optimal Finger Search Trees in the Pointer Machine*. In *Proc. 34th Annual ACM Symposium on Theory of Computing*, pages 583–591, 2002, doi: 10.1145/509907.509991.

Abstract: We develop a new finger search tree with worst-case constant update time in the Pointer Machine (PM) model of computation. This was a major problem in the field of Data Structures and was tantalizingly open for over twenty years while many attempts by researchers were made to solve it. The result comes as a consequence of the innovative mechanism that guides the rebalancing operations combined with incremental multiple splitting and fusion techniques over nodes.

101 Gerth Stølting Brodal, Rolf Fagerberg and Riko Jacob, *Cache-Oblivious Search Trees via Binary Trees of Small Height*. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 39–48, 2002, doi: doi.acm.org/10.1145/545381.545386.

Abstract: We propose a version of cache oblivious search trees which is simpler than the previous proposal of Bender, Demaine and Farach-Colton and has the same complexity bounds. In particular, our data structure avoids the use of weight balanced B-trees, and can be implemented as just a single array of data elements, without the use of pointers. The structure also improves space utilization.

For storing n elements, our proposal uses $(1+\varepsilon)n$ times the element size of memory, and performs searches in worst case $O(\log_B n)$ memory transfers, updates in amortized $O((\log^2 n)/(\varepsilon B))$ memory transfers, and range queries in worst case $O(\log_B n + k/B)$ memory transfers, where k is the size of the output.

The basic idea of our data structure is to maintain a dynamic binary tree of height $\log n + O(1)$ using existing methods, embed this tree in a static binary tree, which in turn is embedded in an array in a cache oblivious fashion, using the van Emde Boas layout of Prokop.

We also investigate the practicality of cache obliviousness in the area of search trees, by providing an empirical comparison of different methods for laying out a search tree in memory.

102 Gerth Stølting Brodal, Rolf Fagerberg, Christian Nørgaard Storm Pedersen and Anna Östlin, *The Complexity of Constructing Evolutionary Trees Using Experiments*. In *Proc. 28th International Colloquium on Automata, Languages, and Programming, Hersonissos, Crete, Greece, 8–12 July 2001*, Volume 2076 of Lecture Notes in Computer Science, pages 140–151. Springer Verlag, Berlin, 2001, doi: 10.1007/3-540-48224-5_12.

Abstract: We present tight upper and lower bounds for the problem of constructing evolutionary trees in the experiment model. We describe an algorithm which constructs an evolutionary tree of n species in time $O(nd \log_d n)$ using at most $n\lceil d/2\rceil(\log_{2\lceil d/2\rceil-1} n + O(1))$ experiments for $d > 2$, and at most $n(\log n + O(1))$ experiments for $d = 2$, where d is the degree of the tree. This improves the previous best upper bound by a factor $\Theta(\log d)$. For $d = 2$ the previously best algorithm with running time $O(n \log n)$ had a bound of $4n \log n$ on the number of experiments. By an explicit adversary argument, we show an $\Omega(nd \log_d n)$ lower bound, matching our upper bounds and improving the previous best lower bound by a factor $\Theta(\log_d n)$. Central to our algorithm is the construction and maintenance of separator trees of small height, which may be of independent interest.

(37) Gerth Stølting Brodal, Rolf Fagerberg and Christian Nørgaard Storm Pedersen, *Computing the Quartet Distance Between Evolutionary Trees in Time $O(n \log^2 n)$*. In *Proc. 12th Annual International Symposium on Algorithms and Computation*, Volume 2223 of Lecture Notes in Computer Science, pages 731–742. Springer Verlag, Berlin, 2001 2001, doi: 10.1007/3-540-45678-3_62.

Abstract: Evolutionary trees describing the relationship for a set of species are central in evolutionary biology, and quantifying differences between evolutionary trees is an important task. One previously proposed measure for this is the quartet distance. The quartet distance between two unrooted evolutionary trees is the number of quartet topology differences between the two trees, where a quartet topology is the topological subtree induced by four species. In this paper, we present an algorithm for computing the quartet distance between two unrooted evolutionary trees of n species in time $O(n \log^2 n)$. The previous best algorithm runs in time $O(n^2)$.

103 Stephen Alstrup, Gerth Stølting Brodal and Theis Rauhe, *Optimal Static Range Reporting in One*

Dimension. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, pages 476–482, 2001, doi: 10.1145/380752.380842 (presentation pdf, zip).

Abstract: We consider static one dimensional range searching problems. These problems are to build static data structures for an integer set $S \subseteq U$, where $U = \{0, 1, \ldots, 2^w - 1\}$, which support various queries for integer intervals of U. For the query of reporting all integers in S contained within a query interval, we present an optimal data structure with linear space cost and with query time linear in the number of integers reported. This result holds in the unit cost RAM model with word size w and a standard instruction set. We also present a linear space data structure for approximate range counting. A range counting query for an interval returns the number of integers in S contained within the interval. For any constant $\varepsilon > 0$, our range counting data structure returns in constant time an approximate answer which is within a factor of at most $1 + \varepsilon$ of the correct answer.

104 Gerth Stølting Brodal and Riko Jacob, *Dynamic Planar Convex Hull with Optimal Query Time and $O(\log n \cdot \log\log n)$ Update Time*. In *Proc. 7th Scandinavian Workshop on Algorithm Theory, Bergen, Norway, 5–7 July 2000*, Volume 1851 of Lecture Notes in Computer Science, pages 57–70. Springer Verlag, Berlin, 2000, doi: 10.1007/3-540-44985-X_7.

Abstract: The dynamic maintenance of the convex hull of a set of points in the plane is one of the most important problems in computational geometry. We present a data structure supporting point insertions in amortized $O(\log n \cdot \log\log\log n)$ time, point deletions in amortized $O(\log n \cdot \log\log n)$ time, and various queries about the convex hull in optimal $O(\log n)$ worst-case time. The data structure requires $O(n)$ space. Applications of the new dynamic convex hull data structure are improved deterministic algorithms for the k-level problem and the red–blue segment intersection problem where all red and all blue segments are connected.

(36) Lars Arge, Gerth Stølting Brodal and Laura Toma, *On External Memory MST, SSSP and Multi-way Planar Graph Separation*. In *Proc. 7th Scandinavian Workshop on Algorithm Theory, Bergen, Norway, 5–7 July 2000*, Volume 1851 of Lecture Notes in Computer Science, pages 433–447. Springer Verlag, Berlin, 2000, doi: 10.1007/3-540-44985-X_37.

Abstract: Recently external memory graph algorithms have received considerable attention because massive graphs arise naturally in many applications involving massive data sets. Even though a large number of I/O-efficient graph algorithms have been developed, a number of fundamental problems still remain open. In this paper we develop an improved algorithm for the problem of computing a minimum spanning tree of a general graph, as well as new algorithms for the single source shortest paths and the multi-way graph separation problems on planar graphs.

105 Gerth Stølting Brodal and Christian Nørgaard Storm Pedersen, *Finding Maximal Quasiperiodicities in Strings*. In *Proc. 11th Annual Symposium on Combinatorial Pattern Matching, Montreal, Quebec, Canada, 21–23 June 2000*, Volume 1848 of Lecture Notes in Computer Science, pages 397–411. Springer Verlag, Berlin, 2000, doi: 10.1007/3-540-45123-4_33.

Abstract: Apostolico and Ehrenfeucht defined the notion of a maximal quasiperiodic substring and gave an algorithm that finds all maximal quasiperiodic substrings in a string of length n in time $O(n \log^2 n)$. In this paper we give an algorithm that finds all maximal quasiperiodic substrings in a string of length n in time $O(n \log n)$ and space $O(n)$. Our algorithm uses the suffix tree as the fundamental data structure combined with efficient methods for merging and performing multiple searches in search trees. Besides finding all maximal quasiperiodic substrings, our algorithm also marks the nodes in the suffix tree that have a superprimitive path-label.

106 Stephen Alstrup, Gerth Stølting Brodal and Theis Rauhe, *New Data Structures for Orthogonal Range Searching*. In *Proc. 41st Annual Symposium on Foundations of Computer Science*, pages 198–207, 2000, doi: 10.1109/SFCS.2000.892088.

Abstract: We present new general techniques for static orthogonal range searching problems in two and higher dimensions. For the general range reporting problem in $\mathbb{R}^3$, we achieve query time $O(\log n + k)$ using space $O(n \log^{1+\epsilon} n)$, where n denotes the number of stored points and k the number of points to be reported. For the range reporting problem on an $n \times n$ grid, we achieve

query time $O(\log\log n + k)$ using space $O(n \log^{\epsilon} n)$. For the two-dimensional semi-group range sum problem we achieve query time $O(\log n)$ using space $O(n \log n)$.

107 Stephen Alstrup, Gerth Stølting Brodal and Theis Rauhe, *Pattern Matching in Dynamic Texts*. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 819–828, 2000, doi: doi.acm.org/10.1145/338219.338645.

Abstract: Pattern matching is the problem of finding all occurrences of a pattern in a text. In a dynamic setting the problem is to support pattern matching in a text which can be manipulated on-line, *i.e.* the usual situation in text editing.

We present a data structure that supports insertions and deletions of characters and movements of arbitrary large blocks within a text in $O(\log^2 n \log\log n \log^* n)$ time per operation. Furthermore a search for a pattern P in the text is supported in time $O(\log n \log\log n + occ + |P|)$, where occ is the number of occurrences to be reported. An ingredient in our solution to the above main result is a data structure for the *dynamic string equality* problem introduced by Mehlhorn, Sundar and Uhrig. As a secondary result we give almost quadratic better time bounds for this problem which in addition to keeping polylogarithmic factors low for our main result also improves the complexity for several other problems.

108 Gerth Stølting Brodal and Rolf Fagerberg, *Dynamic Representations of Sparse Graphs*. In *Proc. 6th International Workshop on Algorithms and Data Structures, Vancouver, Canada, 11–14 August 1999*, Volume 1663 of Lecture Notes in Computer Science, pages 342–351. Springer Verlag, Berlin, 1999, doi: 10.1007/3-540-48447-7_34.

Abstract: We present a linear space data structure for maintaining graphs with bounded arboricity—a large class of sparse graphs containing e.g. planar graphs and graphs of bounded treewidth—under edge insertions, edge deletions, and adjacency queries.

The data structure supports adjacency queries in worst case $O(c)$ time, and edge insertions and edge deletions in amortized $O(1)$ and $O(c + \log n)$ time, respectively, where n is the number of nodes in the graph, and c is the bound on the arboricity.

(42) Gerth Stølting Brodal, Rune Bang Lyngsø, Christian Nørgaard Storm Pedersen and Jens Stoye, *Finding Maximal Pairs with Bounded Gap*. In *Proc. 10th Annual Symposium on Combinatorial Pattern Matching, Warwick, UK, 22–24 June 1999*, Volume 1645 of Lecture Notes in Computer Science, pages 134–149. Springer Verlag, Berlin, 1999, doi: 10.1007/3-540-48452-3_11.

Abstract: A pair in a string is the occurrence of the same substring twice. A pair is maximal if the two occurrences of the substring cannot be extended to the left and right without making them different. The gap of a pair is the number of characters between the two occurrences of the substring. In this paper we present methods for finding all maximal pairs under various constraints on the gap. In a string of length n we can find all maximal pairs with gap in an upper and lower bounded interval in time $O(n \log n + z)$ where z is the number of reported pairs. If the upper bound is removed the time reduces to $O(n + z)$. Since a tandem repeat is a pair where the gap is zero, our methods can be seen as a generalization of finding tandem repeats. The running time of our methods equals the running time of well known methods for finding tandem repeats.

109 Pankaj K. Agarwal, Lars Arge, Gerth Stølting Brodal and Jeff Vitter, *I/O-Efficient Dynamic Point Location in Monotone Subdivisions*. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 11–20, 1999, doi: doi.acm.org/10.1145/314500.314525.

Abstract: We present an efficient external-memory dynamic data structure for point location in monotone planar subdivisions. Our data structure uses $O(N/B)$ disk blocks to store a monotone subdivision of size N, where B is the size of a disk block. It supports queries in $O(\log_B^2 N)$ I/Os (worst-case) and updates in $O(\log_B^2 N)$ I/Os (amortized).

We also propose a new variant of B-trees, called *level-balanced B-trees*, which allow insert, delete, merge, and split operations in $O((1 + (b/B)\log_{M/B}(N/B))\log_b N)$ I/Os (amortized), $2 \le b \le B/2$, even if each node stores a pointer to its parent. Here M is the size of main memory. Besides

being essential to our point-location data structure, we believe that *level-balanced B-trees* are of significant independent interest. They can, for example, be used to dynamically maintain a planar st-graph using $O((1+(b/B)\log_{M/B}(N/B))\log_b N) = O(\log_B^2 N)$ I/Os (amortized) per update, so that reachability queries can be answered in $O(\log_B N)$ I/Os (worst case).

110 Gerth Stølting Brodal and Jyrki Katajainen, *Worst-Case Efficient External-Memory Priority Queues*. In *Proc. 6th Scandinavian Workshop on Algorithm Theory, Stockholm, Sweden, 8–10 July 1998*, Volume 1432 of Lecture Notes in Computer Science, pages 107–118. Springer Verlag, Berlin, 1998, doi: 10.1007/BFb0054359 (presentation pdf, zip).

Abstract: A priority queue Q is a data structure that maintains a collection of elements, each element having an associated priority drawn from a totally ordered universe, under the operations Insert, which inserts an element into Q, and DeleteMin, which deletes an element with the minimum priority from Q. In this paper a priority-queue implementation is given which is efficient with respect to the number of block transfers or I/Os performed between the internal and external memories of a computer. Let B and M denote the respective capacity of a block and the internal memory measured in elements. The developed data structure handles any intermixed sequence of Insert and DeleteMin operations such that in every disjoint interval of B consecutive priority-queue operations at most $c\log_{M/B}(N/M)$ I/Os are performed, for some positive constant c. These I/Os are divided evenly among the operations: if $B \geq c\log_{M/B}(N/M)$, one I/O is necessary for every $B/(c\log_{M/B}(N/M))$th operation and if $B < c\log_{M/B}(N/M)$, $(c/B)\log_{M/B}(N/M)$ I/Os are performed per every operation. Moreover, every operation requires $O(\log_2 N)$ comparisons in the worst case. The best earlier solutions can only handle a sequence of S operations with $O(\sum_{i=1}^{S}(1/B)\log_{M/B}(N_i/M))$ I/Os, where N_i denotes the number of elements stored in the data structure prior to the ith operation, without giving any guarantee for the performance of the individual operations.

(40) Gerth Stølting Brodal and M. Cristina Pinotti, *Comparator Networks for Binary Heap Construction*. In *Proc. 6th Scandinavian Workshop on Algorithm Theory, Stockholm, Sweden, 8–10 July 1998*, Volume 1432 of Lecture Notes in Computer Science, pages 158–168. Springer Verlag, Berlin, 1998, doi: 10.1007/BFb0054364 (presentation pdf, zip).

Abstract: Comparator networks for constructing binary heaps of size n are presented which have size $O(n\log\log n)$ and depth $O(\log n)$. A lower bound of $n\log\log n - O(n)$ for the size of any heap construction network is also proven, implying that the networks presented are within a constant factor of optimal. We give a tight relation between the leading constants in the size of selection networks and in the size of heap construction networks.

111 Gerth Stølting Brodal, *Finger Search Trees with Constant Insertion Time*. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 540–549, 1998, doi: doi.acm.org/10.1145/314613.314842 (presentation pdf, zip).

Abstract: We consider the problem of implementing finger search trees on the pointer machine, *i.e.*, how to maintain a sorted list such that searching for an element x, starting the search at any arbitrary element f in the list, only requires logarithmic time in the distance between x and f in the list.

We present the first pointer-based implementation of finger search trees allowing new elements to be inserted at any arbitrary position in the list in worst case constant time. Previously, the best known insertion time on the pointer machine was $O(\log^* n)$, where n is the total length of the list. On a unit-cost RAM, a constant insertion time has been achieved by Dietz and Raman by using standard techniques of packing small problem sizes into a constant number of machine words.

Deletion of a list element is supported in $O(\log^* n)$ time, which matches the previous best bounds. Our data structure requires linear space.

(44) Gerth Stølting Brodal, Jesper Larsson Träff and Christos D. Zaroliagis, *A Parallel Priority Data Structure with Applications*. In *Proc. 11th International Parallel Processing Symposium*, Dror G.

Feitelson and Larry Rudolph (Edt.), pages 689–693. IEEE Comput. Soc. Press, Los Alamitos, USA, 1997, doi: 10.1109/IPPS.1997.580979.

Abstract: We present a parallel priority data structure that improves the running time of certain algorithms for problems that lack a fast and work-efficient parallel solution. As a main application, we give a parallel implementation of Dijkstra's algorithm which runs in $O(n)$ time while performing $O(m \log n)$ work on a CREW PRAM. This is a logarithmic factor improvement for the running time compared with previous approaches. The main feature of our data structure is that the operations needed in each iteration of Dijkstra's algorithm can be supported in $O(1)$ time.

112 Gerth Stølting Brodal, *Predecessor Queries in Dynamic Integer Sets*. In *Proc. 14th Annual Symposium on Theoretical Aspects of Computer Science*, Volume 1200 of Lecture Notes in Computer Science, pages 21–32. Springer Verlag, Berlin, 1997, doi: 10.1007/BFb0023445 (presentation pdf).

Abstract: We consider the problem of maintaining a set of n integers in the range $0..2^w-1$ under the operations of insertion, deletion, predecessor queries, minimum queries and maximum queries on a unit cost RAM with word size w bits. Let $f(n)$ be an arbitrary nondecreasing smooth function satisfying $\log\log n \leq f(n) \leq \sqrt{\log n}$. A data structure is presented supporting insertions and deletions in worst case $O(f(n))$ time, predecessor queries in worst case $O((\log n)/f(n))$ time and minimum and maximum queries in worst case constant time. The required space is $O(n2^{\epsilon w})$ for an arbitrary constant $\epsilon > 0$. The RAM operations used are addition, arbitrary left and right bit shifts and bit-wise boolean operations. The data structure is the first supporting predecessor queries in worst case $O(\log n/\log\log n)$ time while having worst case $O(\log\log n)$ update time.

(45) Gerth Stølting Brodal, Shiva Chaudhuri and Jaikumar Radhakrishnan, *The Randomized Complexity of Maintaining the Minimum*. In *Proc. 5th Scandinavian Workshop on Algorithm Theory, Reykjavik, Iceland, 3–5 July 1996*, Volume 1097 of Lecture Notes in Computer Science, pages 4–15. Springer Verlag, Berlin, 1996, doi: 10.1007/3-540-61422-2_116 (presentation pdf, zip).

Abstract: The complexity of maintaining a set under the operations Insert, Delete and FindMin is considered. In the comparison model it is shown that any randomized algorithm with expected amortized cost t comparisons per Insert and Delete has expected cost at least $n/(e2^{2t}) - 1$ comparisons for FindMin. If FindMin is replaced by a weaker operation, FindAny, then it is shown that a randomized algorithm with constant expected cost per operation exists, but no deterministic algorithm. Finally, a deterministic algorithm with constant amortized cost per operation for an offline version of the problem is given.

(43) Gerth Stølting Brodal, *Priority Queues on Parallel Machines*. In *Proc. 5th Scandinavian Workshop on Algorithm Theory, Reykjavik, Iceland, 3–5 July 1996*, Volume 1097 of Lecture Notes in Computer Science, pages 416–427. Springer Verlag, Berlin, 1996, doi: 10.1007/3-540-61422-2_150 (presentation pdf, zip).

Abstract: We present time and work optimal priority queues for the CREW PRAM, supporting FindMin in constant time with one processor and MakeQueue, Insert, Meld, FindMin, ExtractMin, Delete and DecreaseKey in constant time with $O(\log n)$ processors. A priority queue can be build in time $O(\log n)$ with $O(n/\log n)$ processors and k elements can be inserted into a priority queue in time $O(\log k)$ with $O((\log n + k)/\log k)$ processors. With a slowdown of $O(\log\log n)$ in time the priority queues adopt to the EREW PRAM by only increasing the required work by a constant factor. A pipelined version of the priority queues adopt to a processor array of size $O(\log n)$, supporting the operations MakeQueue, Insert, Meld, FindMin, ExtractMin, Delete and DecreaseKey in constant time.

113 Gerth Stølting Brodal and Leszek Gąsieniec, *Approximate Dictionary Queries*. In *Proc. 7th Annual Symposium on Combinatorial Pattern Matching, Laguna Beach, CA, 10–12 June 1996*, Volume 1075 of Lecture Notes in Computer Science, pages 65–74. Springer Verlag, Berlin, 1996, doi: 10.1007/3-540-61258-0_6.

Abstract: Given a set of n binary strings of length m each. We consider the problem of answering d–queries. Given a binary query string α of length m, a d–query is to report if there exists a string in the set within Hamming distance d of α.

We present a data structure of size $O(nm)$ supporting 1–queries in time $O(m)$ and the reporting of all strings within Hamming distance 1 of α in time $O(m)$. The data structure can be constructed in time $O(nm)$. A slightly modified version of the data structure supports the insertion of new strings in amortized time $O(m)$.

114 Gerth Stølting Brodal, *Worst-Case Efficient Priority Queues*. In *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 52–58, 1996, doi: doi.acm.org/10.1145/313852.313883 (presentation pdf, zip).

Abstract: An implementation of priority queues is presented that supports the operations MakeQueue, FindMin, Insert, Meld and DecreaseKey in worst case time $O(1)$ and DeleteMin and Delete in worst case time $O(\log n)$. The space requirement is linear. The data structure presented is the first achieving this worst case performance.

115 Gerth Stølting Brodal, *Fast Meldable Priority Queues*. In *Proc. 4th International Workshop on Algorithms and Data Structures*, Volume 955 of Lecture Notes in Computer Science, pages 282–290. Springer Verlag, Berlin, 1995, doi: 10.1007/3-540-60220-8_70 (presentation pdf, zip).

Abstract: We present priority queues that support the operations FindMin, Insert, MakeQueue and Meld in worst case time $O(1)$ and Delete and DeleteMin in worst case time $O(\log n)$. They can be implemented on the pointer machine and require linear space. The time bounds are optimal for all implementations where Meld takes worst case time $o(n)$.

To our knowledge this is the first priority queue implementation that supports Meld in worst case constant time and DeleteMin in logarithmic time.

Technical Reports

116 Gerth Stølting Brodal, Casper Moldrup Rysgaard and Rolf Svenning, *Buffered Partially-Persistent External-Memory Search Trees*. Technical report 2503.08211, 20 pages. ArXiv.org, March 2025, doi: arxiv.org/abs/2503.08211.

Abstract: We present an optimal partially-persistent external-memory search tree with amortized I/O bounds matching those achieved by the non-persistent B^ε-tree by Brodal and Fagerberg [SODA 2003]. In a partially-persistent data structure each update creates a new version of the data structure, where all past versions can be queried, but only the current version can be updated. All operations should be efficient with respect to the size N_v of the accessed version v. For any parameter $0 < \varepsilon < 1$, our data structure supports insertions and deletions in amortized $\mathcal{O}\big(\frac{1}{\varepsilon B^{1-\varepsilon}} \log_B N_v\big)$ I/Os, where B is the external-memory block size. It also supports successor and range reporting queries in amortized $\mathcal{O}\big(\frac{1}{\varepsilon} \log_B N_v + K/B\big)$ I/Os, where K is the number of values reported. The space usage of the data structure is linear in the total number of updates. We make the standard and minimal assumption that the internal memory has size $M \geq 2B$. The previous state-of-the-art external-memory partially-persistent search tree by Arge, Danner and Teh [JEA 2003] supports all operations in worst-case $\mathcal{O}(\log_B N_v + K/B)$ I/Os, matching the bounds achieved by the classical B-tree by Bayer and McCreight [Acta Informatica 1972]. Our data structure successfully combines buffering updates with partial persistence. The I/O bounds can also be achieved in the worst-case sense, by slightly modifying our data structure and under the requirement that the memory size $M = \Omega\big(B^{1-\varepsilon} \log_2(\max_v N_v)\big)$. For updates, where the I/O bound is $o(1)$, we assume that the I/Os are performed evenly spread out among the updates (by performing buffer-overflows incrementally). The worst-case result slightly improves the memory requirement over the previous ephemeral external-memory dictionary by Das, Iacono, and Nekrich (ISAAC 2022), who achieved matching worst-case I/O bounds but required $M = \Omega(B \log_B N)$.

(9) Bruce Brewer, Gerth Stølting Brodal and Haitao Wang, *Dynamic Convex Hulls for Simple Paths*. Technical report 2403.05697, 32 pages. ArXiv.org, March 2024, doi: arxiv.org/abs/2403.05697.

Abstract: We consider the planar dynamic convex hull problem. In the literature, solutions exist supporting the insertion and deletion of points in poly-logarithmic time and various queries on the convex hull of the current set of points in logarithmic time. If arbitrary insertion and deletion of

points are allowed, constant time updates and fast queries are known to be impossible. This paper considers two restricted cases where worst-case constant time updates and logarithmic time queries are possible. We assume all updates are performed on a deque (double-ended queue) of points. The first case considers the monotonic path case, where all points are sorted in a given direction, say horizontally left-to-right, and only the leftmost and rightmost points can be inserted and deleted. The second case assumes that the points in the deque constitute a simple path. Note that the monotone case is a special case of the simple path case. For both cases, we present solutions supporting deque insertions and deletions in worst-case constant time and standard queries on the convex hull of the points in $O(\log n)$ time, where n is the number of points in the current point set. The convex hull of the current point set can be reported in $O(h + \log n)$ time, where h is the number of edges of the convex hull. For the 1-sided monotone path case, where updates are only allowed on one side, the reporting time can be reduced to $O(h)$, and queries on the convex hull are supported in $O(\log h)$ time. All our time bounds are worst case. In addition, we prove lower bounds that match these time bounds, and thus our results are optimal. For a quick comparison, the previous best update bounds for the simple path problem were amortized $O(\log n)$ time by Friedman, Hershberger, and Snoeyink [SoCG 1989].

(51) Gerth Stølting Brodal, *Bottom-up Rebalancing Binary Search Trees by Flipping a Coin*. Technical report 2404.08287, 30 pages. ArXiv.org, April 2024, doi: arxiv.org/abs/2404.08287.

Abstract: Rebalancing schemes for dynamic binary search trees are numerous in the literature, where the goal is to maintain trees of low height, either in the worst-case or expected sense. In this paper we study randomized rebalancing schemes for sequences of n insertions into an initially empty binary search tree, under the assumption that a tree only stores the elements and the tree structure without any additional balance information. Seidel (2009) presented a top-down randomized insertion algorithm, where insertions take expected $O\big(\lg^2 n\big)$ time, and the resulting trees have the same distribution as inserting a uniform random permutation into a binary search tree without rebalancing. Seidel states as an open problem if a similar result can be achieved with bottom-up insertions. In this paper we fail to answer this question.

We consider two simple canonical randomized bottom-up insertion algorithms on binary search trees, assuming that an insertion is given the position where to insert the next element. The subsequent rebalancing is performed bottom-up in expected $O(1)$ time, uses expected $O(1)$ random bits, performs at most two rotations, and the rotations appear with geometrically decreasing probability in the distance from the leaf. For some insertion sequences the expected depth of each node is proved to be $O(\lg n)$. On the negative side, we prove for both algorithms that there exist simple insertion sequences where the expected depth is $\Omega(n)$, i.e., the studied rebalancing schemes are *not* competitive with (most) other rebalancing schemes in the literature.

(52) Gerth Stølting Brodal and Sebastian Wild, *Deterministic Cache-Oblivious Funnelselect*. Technical report 2402.17631, 12 pages. ArXiv.org, February 2024, doi: arxiv.org/abs/2402.17631.

Abstract: In the multiple-selection problem one is given an unsorted array S of N elements and an array of q query ranks $r_1 < \cdots < r_q$, and the task is to return, in sorted order, the q elements in S of rank $r_1, \ldots, r_q$, respectively. The asymptotic deterministic comparison complexity of the problem was settled by Dobkin and Munro [JACM 1981]. In the I/O model an optimal I/O complexity was achieved by Hu *et al.* [SPAA 2014]. Recently [ESA 2023], we presented a *cache-oblivious* algorithm with matching I/O complexity, named *funnelselect*, since it heavily borrows ideas from the cache-oblivious sorting algorithm *funnelsort* from the seminal paper by Frigo, Leiserson, Prokop and Ramachandran [FOCS 1999]. FUNNELSELECT is inherently randomized as it relies on sampling for cheaply finding many good pivots. In this paper we present *deterministic funnelselect*, achieving the same optional I/O complexity cache-obliviously without randomization. Our new algorithm essentially replaces a single (in expectation) reversed-funnel computation using random pivots by a recursive algorithm using multiple reversed-funnel computations. To meet the I/O bound, this requires a carefully chosen subproblem size based on the entropy of the sequence of query ranks; DETERMINISTIC FUNNELSELECT thus raises distinct technical challenges not met by randomized FUNNELSELECT. The resulting worst-case I/O bound is $O(\sum_{i=1}^{q+1} \frac{\Delta_i}{B} \cdot \log_{M/B} \frac{N}{\Delta_i} + \frac{N}{B})$, where B is the external memory block size, $M \geq B^{1+\epsilon}$ is the internal memory size, for some constant $\epsilon > 0$, and $\Delta_i = r_i - r_{i-1}$ (assuming $r_0 = 0$ and $r_{q+1} = N + 1$).

(57) Gerth Stølting Brodal, *Soft Sequence Heaps*. Technical report 2008.05398, 16 pages. ArXiv.org, August 2020, doi: arxiv.org/abs/2008.05398.

Abstract: Chazelle [JACM2000] introduced the *soft heap* as a building block for efficient minimum spanning tree algorithms, and recently Kaplan *et al.* [SOSA2019] showed how soft heaps can be applied to achieve simpler algorithms for various selection problems. A soft heap trades-off accuracy for efficiency, by allowing ϵN of the items in a heap to be *corrupted* after a total of N insertions, where a corrupted item is an item with artificially increased key and $0 < \epsilon \leq \frac{1}{2}$ is a fixed error parameter. Chazelle's soft heaps are based on binomial trees and support insertions in amortized $O(\lg \frac{1}{\epsilon})$ time and extract-min operations in amortized $O(1)$ time.

In this paper we explore the design space of soft heaps. The main contribution of this paper is an alternative soft heap implementation based on merging sorted sequences, with time bounds matching those of Chazelle's soft heaps. We also discuss a variation of the soft heap by Kaplan *et al.* [SICOMP13], where we avoid performing insertions lazily. It is based on ternary trees instead of binary trees and matches the time bounds of Kaplan *et al.*, i.e. amortized $O(1)$ insertions and amortized $O(\lg \frac{1}{\epsilon})$ extract-min. Both our data structures only introduce corruptions after extract-min operations which return the set of items corrupted by the operation.

(97) Riko Jacob and Gerth Stølting Brodal, *Dynamic Planar Convex Hull*. Technical report 1902.11169, 87 pages. ArXiv.org, February 2019, doi: arxiv.org/abs/1902.11169.

Abstract: In this article, we determine the amortized computational complexity of the planar dynamic convex hull problem by querying. We present a data structure that maintains a set of n points in the plane under the insertion and deletion of points in amortized $O(\log n)$ time per operation. The space usage of the data structure is $O(n)$. The data structure supports extreme point queries in a given direction, tangent queries through a given point, and queries for the neighboring points on the convex hull in $O(\log n)$ time. The extreme point queries can be used to decide whether or not a given line intersects the convex hull, and the tangent queries to determine whether a given point is inside the convex hull. We give a lower bound on the amortized asymptotic time complexity that matches the performance of this data structure.

(12) Gerth Stølting Brodal and Konstantinos Mampentzidis, *Cache Oblivious Algorithms for Computing the Triplet Distance Between Trees*. Technical report 1706.10284, 16 pages. ArXiv.org, June 2017, doi: arxiv.org/abs/1706.10284.

Abstract: We study the problem of computing the triplet distance between two rooted unordered trees with n labeled leafs. Introduced by Dobson 1975, the triplet distance is the number of leaf triples that induce different topologies in the two trees. The current theoretically best algorithm is an $\mathrm{O}(n \log n)$ time algorithm by Brodal *et al.* (SODA 2013). Recently Jansson *et al.* proposed a new algorithm that, while slower in theory, requiring $\mathrm{O}(n \log^3 n)$ time, in practice it outperforms the theoretically faster $\mathrm{O}(n \log n)$ algorithm. Both algorithms do not scale to external memory.

We present two cache oblivious algorithms that combine the best of both worlds. The first algorithm is for the case when the two input trees are binary trees and the second a generalized algorithm for two input trees of arbitrary degree. Analyzed in the RAM model, both algorithms require $\mathrm{O}(n \log n)$ time, and in the cache oblivious model $\mathrm{O}(n/B \log_2(N/M))$ I/Os. Their relative simplicity and the fact that they scale to external memory makes them achieve the best practical performance. We note that these are the first algorithms that scale to external memory, both in theory and practice, for this problem.

(58) Gerth Stølting Brodal, *External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates*. Technical report 1509.08240, 16 pages. ArXiv.org, September 2015, doi: arxiv.org/abs/1509.08240.

Abstract: An external memory data structure is presented for maintaining a dynamic set of N two-dimensional points under the insertion and deletion of points, and supporting 3-sided range reporting queries and top-k queries, where top-k queries report the k points with highest y-value within a given x-range. For any constant $0 < \varepsilon \leq \frac{1}{2}$, a data structure is constructed that supports updates in amortized $O(\frac{1}{\varepsilon B^{1-\varepsilon}} \log_B N)$ IOs and queries in amortized $O(\frac{1}{\varepsilon} \log_B N + K/B)$ IOs, where B is the external memory block size, and K is the size of the output to the query (for top-k queries K is the minimum of k and the number of points in the query interval). The data structure uses linear space. The update bound is a significant factor $B^{1-\varepsilon}$ improvement over the previous best update bounds for the two query problems, while staying within the same query and space bounds.

(59) Gerth Stølting Brodal, Jesper Sindahl Nielsen and Jakob Truelsen, *Strictly Implicit Priority Queues: On the Number of Moves and Worst-Case Time*. Technical report 1505.00147, 15 pages. ArXiv.org, May 2015, doi: arxiv.org/abs/1505.00147.

Abstract: The binary heap of Williams (1964) is a simple priority queue characterized by only storing an array containing the elements and the number of elements n – here denoted a *strictly implicit* priority queue. We introduce two new strictly implicit priority queues. The first structure supports amortized $O(1)$ time Insert and $O(\log n)$ time ExtractMin operations, where both operations require amortized $O(1)$ element moves. No previous implicit heap with $O(1)$ time Insert supports both operations with $O(1)$ moves. The second structure supports worst-case $O(1)$ time Insert and $O(\log n)$ time (and moves) ExtractMin operations. Previous results were either amortized or needed $O(\log n)$ bits of additional state information between operations.

(61) Gerth Stølting Brodal and Kasper Green Larsen, *Optimal Planar Orthogonal Skyline Counting Queries*. Technical report 1304.7959, 17 pages. ArXiv.org, April 2013, doi: arxiv.org/abs/1304.7959.

Abstract: The skyline of a set of points in the plane is the subset of maximal points, where a point (x, y) is maximal if no other point (x', y') satisfies $x' \geq x$ and $y' \geq x$. We consider the problem of preprocessing a set P of n points into a space efficient static data structure supporting orthogonal skyline counting queries, i.e. given a query rectangle R to report the size of the skyline of $P \cap R$. We present a data structure for storing n points with integer coordinates having query time $O(\lg n/\lg\lg n)$ and space usage $O(n)$. The model of computation is a unit cost RAM with logarithmic word size. We prove that these bounds are the best possible by presenting a lower bound in the cell probe model with logarithmic word size: Space usage $n \lg^{O(1)} n$ implies worst case query time $\Omega(\lg n/\lg\lg n)$.

(20) Gerth Stølting Brodal, Alexis C. Kaporis, Apostolos Papadopoulos, Spyros Sioutas, Konstantinos Tsakalidis and Kostas Tsichlas, *Dynamic 3-sided Planar Range Queries with Expected Doubly Logarithmic Time*. Technical report 1201.2702, 29 pages. ArXiv.org, January 2012, doi: arxiv.org/abs/1201.2702.

Abstract: This work studies the problem of 2-dimensional searching for the 3-sided range query of the form $[a, b] \times (-\infty, c]$ in both main and external memory, by considering a variety of input distributions. We present three sets of solutions each of which examines the 3-sided problem in both RAM and I/O model respectively. The presented data structures are deterministic and the expectation is with respect to the input distribution:

(1) Under continuous μ-random distributions of the x and y coordinates, we present a dynamic linear main memory solution, which answers 3-sided queries in $O(\log n + t)$ worst case time and scales with $O(\log\log n)$ expected with high probability update time, where n is the current num ber of stored points and t is the size of the query output. We externalize this solution, gaining $O(\log_B n + t/B)$ worst case and $O(\log_B \log n)$ amortized expected with high probability I/Os for query and update operations respectively, where B is the disk block size.

(2) Then, we assume that the inserted points have their x-coordinates drawn from a class of smooth distributions, whereas the y-coordinates are arbitrarily distributed. The points to be deleted are selected uniformly at random among the inserted points. In this case we present a dynamic linear main memory solution that supports queries in $O(\log\log n+t)$ expected time with high probability and updates in $O(\log\log n)$ expected amortized time, where n is the number of points stored and t is the size of the output of the query. We externalize this solution, gaining $O(\log\log_B n + t/B)$ expected I/Os with high probability for query operations and $O(\log_B \log n)$ expected amortized I/Os for update operations, where B is the disk block size. The space remains linear $O(n/B)$.

(3) Finally, we assume that the x-coordinates are continuously drawn from a smooth distribution and the y-coordinates are continuously drawn from a more restricted class of realistic distributions. In this case and by combining the Modified Priority Search Tree with the Priority Search Tree, we present a dynamic linear main memory solution that supports queries in $O(\log\log n + t)$ expected time with high probability and updates in $O(\log\log n)$ expected time with high probability. We externalize this solution, obtaining a dynamic data structure that answers 3-sided queries in $O(\log_B \log n + t/B)$ I/Os expected with high probability, and it can be updated in $O(\log_B \log n)$ I/Os amortized expected with high probability. The space remains linear $O(n/B)$.

(68) Gerth Stølting Brodal and Casper Kejlberg-Rasmussen, *Cache-Oblivious Implicit Predecessor Dictionaries with the Working Set Property*. Technical report 1112.5472, 16 pages. ArXiv.org, December 2011, doi: arxiv.org/abs/1112.5472.

Abstract: In this paper we present an implicit dynamic dictionary with the working-set property, supporting insert(e) and delete(e) in $O(\log n)$ time, predecessor(e) in $O(\log \ell_{p(e)})$ time, successor(e) in

$O(\log \ell_{s(e)})$ time and search(e) in $O(\log \min(\ell_{p(e)}, \ell_e, \ell_{s(e)}))$ time, where n is the number of elements stored in the dictionary, ℓ_e is the number of distinct elements searched for since element e was last searched for and $p(e)$ and $s(e)$ are the predecessor and successor of e, respectively. The time-bounds are all worst-case. The dictionary stores the elements in an array of size n using *no* additional space. In the cache-oblivious model the log is base B and the cache-obliviousness is due to our black box use of an existing cache-oblivious implicit dictionary. This is the first implicit dictionary supporting predecessor and successor searches in the working-set bound. Previous implicit structures required $O(\log n)$ time.

(17) Gerth Stølting Brodal, Spyros Sioutas, Kostas Tsichlas and Christos D. Zaroliagis, *D^2-Tree: A New Overlay with Deterministic Bounds*. Technical report 1009.3134, 21 pages. ArXiv.org, September 2010, doi: arxiv.org/abs/1009.3134.

Abstract: We present a new overlay, called the *Deterministic Decentralized tree* (D^2-tree). The D^2-tree compares favourably to other overlays for the following reasons: (a) it provides matching and better complexities, which are deterministic for the supported operations; (b) the management of nodes (peers) and elements are completely decoupled from each other; and (c) an efficient deterministic load-balancing mechanism is presented for the uniform distribution of elements into nodes, while at the same time probabilistic optimal bounds are provided for the congestion of operations at the nodes. The load-balancing scheme of elements into nodes is deterministic and general enough to be applied to other hierarchical tree-based overlays. This load-balancing mechanism is based on an innovative lazy weight-balancing mechanism, which is interesting in its own right.

(99) Gerth Stølting Brodal and Rolf Fagerberg, *Cache Oblivious Distribution Sweeping*. Technical report BRICS-RS-02-18, 21 pages. BRICS, Department of Computer Science, Aarhus University, 2009.

Abstract: We adapt the distribution sweeping method to the cache oblivious model. Distribution sweeping is the name used for a general approach for divide-and-conquer algorithms where the combination of solved subproblems can be viewed as a merging process of streams. We demonstrate by a series of algorithms for specific problems the feasibility of the method in a cache oblivious setting. The problems all come from computational geometry, and are: orthogonal line segment intersection reporting, the all nearest neighbors problem, the 3D maxima problem, computing the measure of a set of axis-parallel rectangles, computing the visibility of a set of line segments from a point, batched orthogonal range queries, and reporting pairwise intersections of axis-parallel rectangles. Our basic building block is a simplified version of the cache oblivious sorting algorithm Funnelsort of Frigo et al., which is of independent interest.

(79) Gerth Stølting Brodal, Rolf Fagerberg, Allan Grønlund Jørgensen, Gabriel Moruz and Thomas Mølhave, *Optimal Resilient Dynamic Dictionaries*. Technical report DAIMI PB-585, 14 pages. Department of Computer Science, Aarhus University, November 2007.

Abstract: In the resilient memory model any memory cell can get corrupted at any time, and corrupted cells cannot be distinguished from uncorrupted cells. An upper bound, δ, on the number of corruptions and $O(1)$ reliable memory cells are provided. In this model, a data structure is denoted resilient if it gives the correct output on the set of uncor- rupted elements. We propose two optimal resilient static dictionaries, a randomized one and a deterministic one. The randomized dictionary supports searches in $O(\log n + \delta)$ expected time using $O(\log \delta)$ random bits in the worst case, under the assumption that corruptions are not performed by an adaptive adversary. The deterministic static dictionary supports searches in $O(\log n + \delta)$ time in the worst case. We also introduce a deterministic dynamic resilient dictionary supporting searches in $O(\log n + \delta)$ time in the worst case, which is optimal, and updates in $O(\log n + \delta)$ amortized time. Our dynamic dictionary supports range queries in $O(\log n + \delta + k)$ worst case time, where k is the size of the output.

(27) Gerth Stølting Brodal, Kanela Kaligosi, Irit Katriel and Martin Kutz, *Faster Algorithms for Computing Longest Common Increasing Subsequences*. Technical report BRICS-RS-05-37, 16 pages. BRICS, Department of Computer Science, Aarhus University, December 2005.

Abstract: We present algorithms for finding a longest common increasing subsequence of two or more input sequences. For two sequences of lengths m and n, where $m \geq n$, we present an algorithm with an output-dependent expected running time of $O((m + n\ell) \log \log \sigma + \mathit{Sort})$ and $O(m)$ space, where ℓ is the length of a LCIS, σ is the size of the alphabet, and *Sort* is the time to sort each input sequence.

For $k \geq 3$ length-n sequences we present an algorithm with running time $O(\min\{kr^2, r\log^{k-1} r\} + Sort)$, which improves the previous best bound by more than a factor k for many inputs. In both cases, our algorithms are conceptually quite simple but rely on existing sophisticated data structures.

Finally, we introduce the problem of longest common weakly-increasing (or non-decreasing) subsequences (LCWIS), for which we present an $O(m + n\log n)$ time algorithm for the 3-letter alphabet case. For the extensively studied Longest Common Subsequence problem, comparable speedups have not been achieved for small alphabets.

(31) Gerth Stølting Brodal, Rolf Fagerberg and Gabriel Moruz, *On the Adaptiveness of Quicksort*. Technical report BRICS-RS-04-27, 23 pages. BRICS, Department of Computer Science, Aarhus University, December 2004.

Abstract: Quicksort was first introduced in 1961 by Hoare. Many variants have been developed, the best of which are among the fastest generic sorting algorithms available, as testified by the choice of Quicksort as the default sorting algorithm in most programming libraries. Some sorting algorithms are adaptive, i.e. they have a complexity analysis which is better for inputs which are nearly sorted, according to some specified measure of presortedness. Quicksort is not among these, as it uses $\Omega(n\log n)$ comparisons even when the input is already sorted. However, in this paper we demonstrate empirically that the actual running time of Quicksort *is* adaptive with respect to the presortedness measure Inv. Differences close to a factor of two are observed between instances with low and high Inv value. We then show that for the randomized version of Quicksort, the number of element *swaps* performed is *provably* adaptive with respect to the measure Inv. More precisely, we prove that randomized Quicksort performs expected $O(n(1+\log(1+\mathrm{Inv}/n)))$ element swaps, where Inv denotes the number of inversions in the input sequence. This result provides a theoretical explanation for the observed behavior, and gives new insights on the behavior of the Quicksort algorithm. We also give some empirical results on the adaptive behavior of Heapsort and Mergesort.

(91) Gerth Stølting Brodal, Rolf Fagerberg, Ulrich Meyer and Norbert Zeh, *Cache-Oblivious Data Structures and Algorithms for Undirected Breadth-First Search and Shortest Paths*. Technical report BRICS-RS-04-2, 19 pages. BRICS, Department of Computer Science, Aarhus University, January 2004.

Abstract: We present improved cache-oblivious data structures and algorithms for breadth-first search (BFS) on undirected graphs and the single-source shortest path (SSSP) problem on undirected graphs with non-negative edge weights. For the SSSP problem, our result closes the performance gap between the currently best *cache-aware* algorithm and the *cache-oblivious* counterpart. Our cache-oblivious SSSP-algorithm takes nearly full advantage of block transfers for *dense* graphs. The algorithm relies on a new data structure, called *bucket heap*, which is the first cache-oblivious priority queue to efficiently support a weak DecreaseKey operation. For the BFS problem, we reduce the number of I/Os for *sparse* graphs by a factor of nearly $\sqrt{B}$, where B is the cache-block size, nearly closing the performance gap between the currently best *cache-aware* and *cache-oblivious* algorithms.

(34) Gerth Stølting Brodal, Rolf Fagerberg, Thomas Mailund, Christian Nørgaard Storm Pedersen and Derek Phillips, *Speeding Up Neighbour-Joining Tree Construction*. Technical report ALCOMFT-TR-03-102, 9 pages. ALCOM-FT, November 2003.

Abstract: A widely used method for constructing phylogenetic trees is the neighbour-joining method of Saitou and Nei. We develope heuristics for speeding up the neighbour-joining method which generate the same phylogenetic trees as the original method.

All heuristics are based on using a quad-tree to guide the search for the next pair of nodes to join, but differ in the information stored in quad-tree nodes, the way the search is performed, and in the way the quad-tree is updated after a join.

We empirically evaluate the performance of the heuristics on distance matrices obtained from the Pfam collection of alignments, and compare the running time with that of the QuickTree tool, a well-known and widely used implementation of the standard neighbour-joining method.

The results show that the presented heuristics can give a significant speed-up over the standard neighbour-joining method, already for medium sized instances.

(33) Gerth Stølting Brodal, Rolf Fagerberg and Kristoffer Vinther, *Engineering a Cache-Oblivious Sorting Algorithm*. Technical report ALCOMFT-TR-03-101, 16 pages. ALCOM-FT, November 2003.

Abstract: The cache-oblivious model of computation is a two-level memory model with the assumption that the parameters of the model are unknown to the algorithms. A consequence of this assumption is that an algorithm efficient in the cache oblivious model is automatically efficient in a multi-level memory model.

Since the introduction of the cache-oblivious model by Frigo et al. in 1999, a number of algorithms and data structures in the model has been proposed and analyzed. However, less attention has been given to whether the nice theoretical proporities of cache-oblivious algorithms carry over into practice.

This paper is an algorithmic engineering study of cache-oblivious sorting. We investigate a number of implementation issues and parameters choices for the cache-oblivious sorting algorithm Lazy Funnelsort by empirical methods, and compare the final algorithm with Quicksort, the established standard for comparison based sorting, as well as with recent cache-aware proposals.

The main result is a carefully implemented cache-oblivious sorting algorithm, which we compare to the best implementation of Quicksort we can find, and find that it competes very well for input residing in RAM, and outperforms Quicksort for input on disk.

(93) Gerth Stølting Brodal, Rolf Fagerberg, Anna Östlin, Christian Nørgaard Storm Pedersen and S. Srinivasa Rao, *Computing Refined Buneman Trees in Cubic Time*. Technical report ALCOMFT-TR-03-73, 11 pages. ALCOM-FT, November 2003.

Abstract: Reconstructing the evolutionary tree for a set of n species based on pairwise distances between the species is a fundamental problem in bioinformatics. Neighbor joining is a popular distance based tree reconstruction method. It always proposes fully resolved binary trees despite missing evidence in the underlying distance data. Distance based methods based on the theory of Buneman trees and refined Buneman trees avoid this problem by only proposing evolutionary trees whose edges satisfy a number of constraints. These trees might not be fully resolved but there is strong combinatorial evidence for each proposed edge. The currently best algorithm for computing the refined Buneman tree from a given distance measure has a running time of $O(n^5)$ and a space consumption of $O(n^4)$. In this paper, we present an algorithm with running time $O(n^3)$ and space consumption $O(n^2)$. The improved complexity of our algorithm makes the method of refined Buneman trees computational competitive to methods based on neighbor joining.

(94) Gerth Stølting Brodal and Rolf Fagerberg, *On the Limits of Cache-Obliviousness*. Technical report ALCOMFT-TR-03-74, 17 pages. ALCOM-FT, November 2003.

Abstract: In this paper, we present lower bounds for permuting and sorting in the cache-oblivious model. We prove that (1) I/O optimal cache-oblivious comparison based sorting is not possible without a tall cache assumption, and (2) there does not exist an I/O optimal cache-oblivious algorithm for permuting, not even in the presence of a tall cache assumption.

Our results for sorting show the existence of an inherent trade-off in the cache-oblivious model between the strength of the tall cache assumption and the overhead for the case $M \gg B$, and show that Funnelsort and recursive binary mergesort are optimal algorithms in the sense that they attain this trade-off.

(95) Gerth Stølting Brodal and Rolf Fagerberg, *Lower Bounds for External Memory Dictionaries*. Technical report ALCOMFT-TR-03-75, 13 pages. ALCOM-FT, November 2003.

Abstract: We study trade-offs between the update time and the query time for comparison based external memory dictionaries. The main contributions of this paper are two lower bound trade-offs between the I/O complexity of member queries and insertions: If $N > M$ insertions perform at most $\delta \cdot N/B$ I/Os, then (1) there exists a query requiring $N/(M \cdot (M/B)^{O(\delta)})$ I/Os, and (2) there exists a query requiring $\Omega(\log_{\delta \log^2 N}(N/M))$ I/Os when δ is $O(B/\log^3 N)$ and N is at least M^2. For both lower bounds we describe data structures which give matching upper bounds for a wide range of parameters, thereby showing the lower bounds to be tight within these ranges.

(28) Michael A. Bender, Gerth Stølting Brodal, Rolf Fagerberg, Dongdong Ge, Simai He, Haodong Hu, John Iacono and Alejandro López-Ortiz, *The Cost of Cache-Oblivious Searching*. Technical report ALCOMFT-TR-03-76, 18 pages. ALCOM-FT, November 2003.

Abstract: Tight bounds on the cost of cache-oblivious searching are proved. It is shown that no cache-oblivious search structure can guarantee that a search performs fewer than $\lg e \log_B N$ block transfers between any two levels of the memory hierarchy. This lower bound holds even if all of the block sizes are limited to be powers of 2. A modified version of the van Emde Boas layout is proposed, whose expected block transfers between any two levels of the memory hierarchy arbitrarily close to $[\lg e + O(\lg \lg B / \lg B)] \log_B N + O(1)$. This factor approaches $\lg e \approx 1.443$ as B increases. The expectation is taken over the random placement of the first element of the structure in memory.

As searching in the Disk Access Model (DAM) can be performed in $\log_B N + 1$ block transfers, this result shows a separation between the 2-level DAM and cache-oblivious memory-hierarchy models. By extending the DAM model to k levels, multilevel memory hierarchies can be modelled. It is shown that as k grows, the search costs of the optimal k-level DAM search structure and of the optimal cache-oblivious search structure rapidly converge. This demonstrates that for a multilevel memory hierarchy, a simple cache-oblivious structure almost replicates the performance of an optimal parameterized k-level DAM structure.

(35) Gerth Stølting Brodal, Erik D. Demaine and J. Ian Munro, *Fast Allocation and Deallocation with an Improved Buddy System*. Technical report ALCOMFT-TR-03-3, 15 pages. ALCOM-FT, May 2003.

Abstract: We propose several modifications to the binary buddy system for managing dynamic allocation of memory blocks whose sizes are powers of two. The standard buddy system allocates and deallocates blocks in $\Theta(\lg n)$ time in the worst case (and on an amortized basis), where n is the size of the memory. We present three schemes that improve the running time to $O(1)$ time, where the time bound for deallocation is amortized for the first two schemes. The first scheme uses just one more word of memory than the standard buddy system, but may result in greater fragmentation than necessary. The second and third schemes have essentially the same fragmentation as the standard buddy system, and use $O(2^{(1+\sqrt{\lg n})\lg\lg n})$ bits of auxiliary storage, which is $\omega(\lg^k n)$ but $o(n^\varepsilon)$ for all $k \geq 1$ and $\varepsilon > 0$. Finally, we present simulation results estimating the effect of the excess fragmentation in the first scheme.

(93) Gerth Stølting Brodal, Rolf Fagerberg, Anna Östlin, Christian Nørgaard Storm Pedersen and S. Srinivasa Rao, *Computing Refined Buneman Trees in Cubic Time*. Technical report BRICS-RS-02-51, 14 pages. BRICS, Department of Computer Science, Aarhus University, December 2002.

Abstract: Reconstructing the evolutionary tree for a set of n species based on pairwise distances between the species is a fundamental problem in bioinformatics. Neighbor joining is a popular distance based tree reconstruction method. It always proposes fully resolved binary trees despite missing evidence in the underlying distance data. Distance based methods based on the theory of Buneman trees and refined Buneman trees avoid this problem by only proposing evolutionary trees whose edges satisfy a number of constraints. These trees might not be fully resolved but there is strong combinatorial evidence for each proposed edge. The currently best algorithm for computing the refined Buneman tree from a given distance measure has a running time of $O(n^5)$ and a space consumption of $O(n^4)$. In this paper, we present an algorithm with running time $O(n^3)$ and space consumption $O(n^2)$.

(96) Gerth Stølting Brodal and Rolf Fagerberg, *Funnel Heap - A Cache Oblivious Priority Queue*. Technical report ALCOMFT-TR-02-136, 11 pages. ALCOM-FT, June 2002.

Abstract: The cache oblivious model of computation is a two-level memory model with the assumption that the parameters of the model are unknown to the algorithms. A consequence of this assumption is that an algorithm efficient in the cache oblivious model is automatically efficient in a multi-level memory model. Arge et al. recently presented the first optimal cache oblivious priority queue, and demonstrated the importance of this result by providing the first cache oblivious algorithms for graph problems. Their structure uses cache oblivious sorting and selection as subroutines. In this paper, we devise an alternative optimal cache oblivious priority queue based only on binary merging. We also show that our structure can be made adaptive to different usage profiles.

(38) Gerth Stølting Brodal, George Lagogiannis, Christos Makris, Athanasios Tsakalidis and Kostas Tsichlas, *Optimal Finger Search Trees in the Pointer Machine*. Technical report ALCOMFT-TR-

02-77, 17 pages. ALCOM-FT, May 2002.

Abstract: We develop a new finger search tree with worst-case constant update time in the Pointer Machine (PM) model of computation. This was a major problem in the field of Data Structures and was tantalizingly open for over twenty years while many attempts by researchers were made to solve it. The result comes as a consequence of the innovative mechanism that guides the rebalancing operations combined with incremental multiple splitting and fusion techniques over nodes.

(98) Stephen Alstrup, Gerth Stølting Brodal, Inge Li Gørtz and Theis Rauhe, *Time and Space Efficient Multi-Method Dispatching*. Technical report ALCOMFT-TR-02-76, 9 pages. ALCOM-FT, May 2002.

Abstract: The *dispatching problem* for object oriented languages is the problem of determining the most specialized method to invoke for calls at run-time. This can be a critical component of execution performance. A number of recent results, including [Muthukrishnan and Müller SODA'96, Ferragina and Muthukrishnan ESA'96, Alstrup et al. FOCS'98], have studied this problem and in particular provided various efficient data structures for the *mono-method* dispatching problem. A recent paper of Ferragina, Muthukrishnan and de Berg [STOC'99] addresses the *multi-method* dispatching problem.

Our main result is a linear space data structure for *binary* dispatching that supports dispatching in logarithmic time. Using the same query time as Ferragina et al., this result improves the space bound with a logarithmic factor.

(100) Gerth Stølting Brodal, Rune Bang Lyngsø, Anna Östlin and Christian Nørgaard Storm Pedersen, *Solving the String Statistics Problem in Time $O(n \log n)$*. Technical report BRICS-RS-02-13, 28 pages. BRICS, Department of Computer Science, Aarhus University, October 2002.

Abstract: The string statistics problem consists of preprocessing a string of length n such that given a query pattern of length m, the maximum number of non-overlapping occurrences of the query pattern in the string can be reported efficiently. Apostolico and Preparata introduced the minimal augmented suffix tree (MAST) as a data structure for the string statistics problem, and showed how to construct the MAST in time $O(n \log^2 n)$ and how it supports queries in time $O(m)$ for constant sized alphabets. A subsequent theorem by Fraenkel and Simpson stating that a string has at most a linear number of distinct squares implies that the MAST requires space $O(n)$. In this paper we improve the construction time for the MAST to $O(n \log n)$ by extending the algorithm of Apostolico and Preparata to exploit properties of efficient joining and splitting of search trees together with a refined analysis.

(100) Gerth Stølting Brodal, Rune Bang Lyngsø, Anna Östlin and Christian Nørgaard Storm Pedersen, *Solving the String Statistics Problem in Time $O(n \log n)$*. Technical report ALCOMFT-TR-02-55, 12 pages. ALCOM-FT, May 2002.

Abstract: The string statistics problem consists of preprocessing a string of length n such that given a query pattern of length m, the maximum number of non-overlapping occurrences of the query pattern in the string can be reported efficiently. Apostolico and Preparata introduced the minimal augmented suffix tree (MAST) as a data structure for the string statistics problem, and showed how to construct the MAST in time $O(n \log^2 n)$ and how it supports queries in time $O(m)$ for constant sized alphabets. A subsequent theorem by Fraenkel and Simpson stating that a string has at most a linear number of distinct squares implies that the MAST requires space $O(n)$. In this paper we improve the construction time for the MAST to $O(n \log n)$ by extending the algorithm of Apostolico and Preparata to exploit properties of efficient joining and splitting of search trees together with a refined analysis.

(37) Gerth Stølting Brodal, Rolf Fagerberg and Christian Nørgaard Storm Pedersen, *Computing the Quartet Distance Between Evolutionary Trees in Time $O(n \log n)$*. Technical report ALCOMFT-TR-02-54, 15 pages. ALCOM-FT, May 2002.

Abstract: Evolutionary trees describing the relationship for a set of species are central in evolutionary biology, and quantifying differences between evolutionary trees is an important task. The quartet distance is a distance measure between trees previously proposed by Estabrook, McMorris and Meacham. The quartet distance between two unrooted evolutionary trees is the number of quartet topology differences between the two trees, where a quartet topology is the topological subtree induced by four species. In this paper, we present an algorithm for computing the quartet

distance between two unrooted evolutionary trees of n species in time $O(n \log n)$. The previous best algorithm for the problem uses time $O(n \log^2 n)$.

(101) Gerth Stølting Brodal, Rolf Fagerberg and Riko Jacob, *Cache-Oblivious Search Trees via Trees of Small Height*. Technical report ALCOMFT-TR-02-53, 20 pages. ALCOM-FT, May 2002.

Abstract: We propose a version of cache oblivious search trees which is simpler than the previous proposal of Bender, Demaine and Farach-Colton and has the same complexity bounds. In particular, our data structure avoids the use of weight balanced B-trees, and can be implemented as just a single array of data elements, without the use of pointers. The structure also improves space utilization.

For storing n elements, our proposal uses $(1+\varepsilon)n$ times the element size of memory, and performs searches in worst case $O(\log_B n)$ memory transfers, updates in amortized $O((\log^2 n)/(\varepsilon B))$ memory transfers, and range queries in worst case $O(\log_B n + k/B)$ memory transfers, where k is the size of the output.

The basic idea of our data structure is to maintain a dynamic binary tree of height $\log n + O(1)$ using existing methods, embed this tree in a static binary tree, which in turn is embedded in an array in a cache oblivious fashion, using the van Emde Boas layout of Prokop.

We also investigate the practicality of cache obliviousness in the area of search trees, by providing an empirical comparison of different methods for laying out a search tree in memory.

(99) Gerth Stølting Brodal and Rolf Fagerberg, *Cache Oblivious Distribution Sweeping*. Technical report ALCOMFT-TR-02-52, 22 pages. ALCOM-FT, May 2002.

Abstract: We adapt the distribution sweeping method to the cache oblivious model. Distribution sweeping is the name used for a general approach for divide-and-conquer algorithms where the combination of solved subproblems can be viewed as a merging process of streams. We demonstrate by a series of algorithms for specific problems the feasibility of the method in a cache oblivious setting. The problems all come from computational geometry, and are: orthogonal line segment intersection reporting, the all nearest neighbors problem, the 3D maxima problem, computing the measure of a set of axis-parallel rectangles, computing the visibility of a set of line segments from a point, batched orthogonal range queries, and reporting pairwise intersections of axis-parallel rectangles. Our basic building block is a simplified version of the cache oblivious sorting algorithm Funnelsort of Frigo et al., which is of independent interest.

(101) Gerth Stølting Brodal, Rolf Fagerberg and Riko Jacob, *Cache Oblivious Search Trees via Binary Trees of Small Height*. Technical report BRICS-RS-01-36, 20 pages. BRICS, Department of Computer Science, Aarhus University, October 2001.

Abstract: We propose a version of cache oblivious search trees which is simpler than the previous proposal of Bender, Demaine and Farach-Colton and has the same complexity bounds. In particular, our data structure avoids the use of weight balanced B-trees, and can be implemented as just a single array of data elements, without the use of pointers. The structure also improves space utilization.

For storing n elements, our proposal uses $(1+\varepsilon)n$ times the element size of memory, and performs searches in worst case $O(\log_B n)$ memory transfers, updates in amortized $O((\log^2 n)/(\varepsilon B))$ memory transfers, and range queries in worst case $O(\log_B n + k/B)$ memory transfers, where k is the size of the output.

The basic idea of our data structure is to maintain a dynamic binary tree of height $\log n + O(1)$ using existing methods, embed this tree in a static binary tree, which in turn is embedded in an array in a cache oblivious fashion, using the van Emde Boas layout of Prokop.

We also investigate the practicality of cache obliviousness in the area of search trees, by providing an empirical comparison of different methods for laying out a search tree in memory.

(102) Gerth Stølting Brodal, Rolf Fagerberg, Christian Nørgaard Storm Pedersen and Anna Östlin, *The Complexity of Constructing Evolutionary Trees Using Experiments*. Technical report BRICS-RS-01-1, 27 pages. BRICS, Department of Computer Science, Aarhus University, July 2001.

Abstract: We present tight upper and lower bounds for the problem of constructing evolutionary trees in the experiment model. We describe an algorithm which constructs an evolutionary tree of n species in time $O(nd \log_d n)$ using at most $n\lceil d/2 \rceil (\log_{2\lceil d/2 \rceil - 1} n + O(1))$ experiments for $d > 2$,

and at most $n(\log n + O(1))$ experiments for $d = 2$, where d is the degree of the tree. This improves the previous best upper bound by a factor $\Theta(\log d)$. For $d = 2$ the previously best algorithm with running time $O(n \log n)$ had a bound of $4n \log n$ on the number of experiments. By an explicit adversary argument, we show an $\Omega(nd \log_d n)$ lower bound, matching our upper bounds and improving the previous best lower bound by a factor $\Theta(\log_d n)$. Central to our algorithm is the construction and maintenance of separator trees of small height. We present how to maintain separator trees with height $\log n + O(1)$ under the insertion of new nodes in amortized time $O(\log n)$. Part of our dynamic algorithm is an algorithm for computing a centroid tree in optimal time $O(n)$.

(92) Gerth Stølting Brodal and Riko Jacob, *Time-dependent networks as models to achieve fast exact time-table queries*. Technical report ALCOMFT-TR-01-176, 12 pages. ALCOM-FT, September 2001.

Abstract: We consider efficient algorithms for exact time-table queries, i.e. algorithms that find optimal itineraries. We propose to use time-dependent networks as a model and show advantages of this approach over space-time networks as models

(37) Gerth Stølting Brodal, Rolf Fagerberg and Christian Nørgaard Storm Pedersen, *Computing the Quartet Distance Between Evolutionary Trees in Time $O(n \log^2 n)$*. Technical report ALCOMFT-TR-01-131, 12 pages. ALCOM-FT, May 2001.

Abstract: Evolutionary trees describing the relationship for a set of species are central in evolutionary biology. Comparing evolutionary trees to quantify differences arising when estimating trees using different methods or data is a fundamental problem. In this paper we present an algorithm for computing the quartet distance between two unrooted evolutionary trees of n species in time $O(n \log^2 n)$. The previous best algorithm runs in time $O(n^2)$. The quartet distance between two unrooted evolutionary trees is the number of quartet topology differences between the two trees, where a quartet topology is the topological subtree induced by four species.

(102) Gerth Stølting Brodal, Rolf Fagerberg, Christian Nørgaard Storm Pedersen and Anna Östlin, *The Complexity of Constructing Evolutionary Trees Using Experiments*. Technical report ALCOMFT-TR-01-130, 25 pages. ALCOM-FT, May 2001.

Abstract: We present tight upper and lower bounds for the problem of constructing evolutionary trees in the experiment model. We describe an algorithm which constructs an evolutionary tree of n species in time $O(nd \log_d n)$ using at most $n\lceil d/2 \rceil (\log_{2\lceil d/2 \rceil - 1} n + O(1))$ experiments for $d > 2$, and at most $n(\log n + O(1))$ experiments for $d = 2$, where d is the degree of the tree. This improves the previous best upper bound by a factor $\Theta(\log d)$. For $d = 2$ the previously best algorithm with running time $O(n \log n)$ had a bound of $4n \log n$ on the number of experiments. By an explicit adversary argument, we show an $\Omega(nd \log_d n)$ lower bound, matching our upper bounds and improving the previous best lower bound by a factor $\Theta(\log_d n)$. Central to our algorithm is the construction and maintenance of separator trees of small height. We present how to maintain separator trees with height $\log n + O(1)$ under the insertion of new nodes in amortized time $O(\log n)$. Part of our dynamic algorithm is an algorithm for computing a centroid tree in optimal time $O(n)$.

(103) Stephen Alstrup, Gerth Stølting Brodal and Theis Rauhe, *Optimal Static Range Reporting in One Dimension*. Technical report ALCOMFT-TR-01-53, 13 pages. ALCOM-FT, May 2001.

Abstract: We consider static one dimensional range searching problems. These problems are to build static data structures for an integer set $S \subseteq U$, where $U = \{0, 1, \ldots, 2^w - 1\}$, which support various queries for integer intervals of U. For the query of reporting all integers in S contained within a query interval, we present an optimal data structure with linear space cost and with query time linear in the number of integers reported. This result holds in the unit cost RAM model with word size w and a standard instruction set. We also present a linear space data structure for approximate range counting. A range counting query for an interval returns the number of integers in S contained within the interval. For any constant $\varepsilon > 0$, our range counting data structure returns in constant time an approximate answer which is within a factor of at most $1 + \varepsilon$ of the correct answer.

(36) Lars Arge, Gerth Stølting Brodal and Laura Toma, *On External Memory MST, SSSP and Multi-way Planar Graph Separation*. Technical report ALCOMFT-TR-01-38, 14 pages. ALCOM-FT, May 2001.

Abstract: Recently external memory graph algorithms have received considerable attention because

massive graphs arise naturally in many applications involving massive data sets. Even though a large number of I/O-efficient graph algorithms have been developed, a number of fundamental problems still remain open. In this paper we develop an improved algorithm for the problem of computing a minimum spanning tree of a general graph, as well as new algorithms for the single source shortest paths and the multi-way graph separation problems on planar graphs.

(106) Stephen Alstrup, Gerth Stølting Brodal and Theis Rauhe, *New Data Structures for Orthogonal Range Searching*. Technical report ALCOMFT-TR-01-35, 17 pages. ALCOM-FT, May 2001.

Abstract: We present new general techniques for static orthogonal range searching problems in two and higher dimensions. For the general range reporting problem in $I\!R3$, we achieve query time $O(\log n + k)$ using space $O(n \log^{1+\varepsilon} n)$, where n denotes the number of stored points and k the number of points to be reported. For the range reporting problem on an $n \times n$ grid, we achieve query time $O(\log\log n + k)$ using space $O(n \log^{\varepsilon} n)$. For the two-dimensional semi-group range sum problem we achieve query time $O(\log n)$ using space $O(n \log n)$.

(104) Gerth Stølting Brodal and Riko Jacob, *Dynamic Planar Convex Hull with Optimal Query Time and $O(\log n \cdot \log\log n)$ Update Time*. Technical report ALCOMFT-TR-01-34, 14 pages. ALCOM-FT, May 2001.

Abstract: The dynamic maintenance of the convex hull of a set of points in the plane is one of the most important problems in computational geometry. We present a data structure supporting point insertions in amortized $O(\log n \cdot \log\log\log n)$ time, point deletions in amortized $O(\log n \cdot \log\log n)$ time, and various queries about the convex hull in optimal $O(\log n)$ worst-case time. The data structure requires $O(n)$ space. Applications of the new dynamic convex hull data structure are improved deterministic algorithms for the k-level problem and the red–blue segment intersection problem where all red and all blue segments are connected.

(98) Stephen Alstrup, Gerth Stølting Brodal, Inge Li Gørtz and Theis Rauhe, *Time and Space Efficient Multi-Method Dispatching*. Technical report ITU-TR-2001-8, 13 pages. The IT University of Copenhagen, October 2001.

Abstract: The *dispatching problem* for object oriented languages is the problem of determining the most specialized method to invoke for calls at run-time. This can be a critical component of execution performance. A number of recent results, including [Muthukrishnan and Müller SODA'96, Ferragina and Muthukrishnan ESA'96, Alstrup et al. FOCS'98], have studied this problem and in particular provided various efficient data structures for the *mono-method* dispatching problem. A recent paper of Ferragina, Muthukrishnan and de Berg [STOC'99] addresses the *multi-method* dispatching problem.

Our main result is a linear space data structure for *binary* dispatching that supports dispatching in logarithmic time. Using the same query time as Ferragina et al., this result improves the space bound with a logarithmic factor.

(103) Stephen Alstrup, Gerth Stølting Brodal and Theis Rauhe, *Optimal Static Range Reporting in One Dimension*. Technical report ITU-TR-2000-3, 12 pages. The IT University of Copenhagen, November 2000.

Abstract: We consider static one dimensional range searching problems. These problems are to build static data structures for an integer set $S \subseteq U$, where $U = \{0, 1, \ldots, 2^w - 1\}$, which support various queries for integer intervals of U. For the query of reporting all integers in S contained within a query interval, we present an optimal data structure with linear space cost and with query time linear in the number of integers reported. This result holds in the unit cost RAM model with word size w and a standard instruction set. We also present a linear space data structure for approximate range counting. A range counting query for an interval returns the number of integers in S contained within the interval. For any constant $\varepsilon > 0$, our range counting data structure returns in constant time an approximate answer which is within a factor of at most $1 + \varepsilon$ of the correct answer.

(41) Gerth Stølting Brodal and Venkatesh Srinivasan, *Improved Bounds for Dictionary Look-up with One Error*. Technical report BRICS-RS-99-50, 5 pages. BRICS, Department of Computer Science, Aarhus University, December 1999.

Abstract: Given a dictionary S of n binary strings each of length m, we consider the problem of

designing a data structure for S that supports *d-queries*; given a binary query string q of length m, a d-query reports if there exists a string in S within Hamming distance d of q. We construct a data structure for the case $d = 1$, that requires space $O(n \log m)$ and has query time $O(1)$ in a cell probe model with word size m. This generalizes and improves the previous bounds of Yao and Yao for the problem in the bit probe model.

(105) Gerth Stølting Brodal and Christian Nørgaard Storm Pedersen, *Finding Maximal Quasiperiodicities in Strings*. Technical report BRICS-RS-99-25, 20 pages. BRICS, Department of Computer Science, Aarhus University, September 1999.

Abstract: Apostolico and Ehrenfeucht defined the notion of a maximal quasiperiodic substring and gave an algorithm that finds all maximal quasiperiodic substrings in a string of length n in time $O(n \log^2 n)$. In this paper we give an algorithm that finds all maximal quasiperiodic substrings in a string of length n in time $O(n \log n)$ and space $O(n)$. Our algorithm uses the suffix tree as the fundamental data structure combined with efficient methods for merging and performing multiple searches in search trees. Besides finding all maximal quasiperiodic substrings, our algorithm also marks the nodes in the suffix tree that have a superprimitive path-label

(42) Gerth Stølting Brodal, Rune Bang Lyngsø, Christian Nørgaard Storm Pedersen and Jens Stoye, *Finding Maximal Pairs with Bounded Gap*. Technical report BRICS-RS-99-12, 31 pages. BRICS, Department of Computer Science, Aarhus University, April 1999.

Abstract: A pair in a string is the occurrence of the same substring twice. A pair is maximal if the two occurrences of the substring cannot be extended to the left and right without making them different. The gap of a pair is the number of characters between the two occurrences of the substring. In this paper we present methods for finding all maximal pairs under various constraints on the gap. In a string of length n we can find all maximal pairs with gap in an upper and lower bounded interval in time $O(n \log n + z)$ where z is the number of reported pairs. If the upper bound is removed the time reduces to $O(n + z)$. Since a tandem repeat is a pair where the gap is zero, our methods can be seen as a generalization of finding tandem repeats. The running time of our methods equals the running time of well known methods for finding tandem repeats

(107) Stephen Alstrup, Gerth Stølting Brodal and Theis Rauhe, *Dynamic Pattern Matching*. Technical report DIKU Report 98/27, 16 pages. Department of Computer Science, University of Copenhagen, 1998.

Abstract: Pattern matching is the problem of finding all occurrences of a pattern in a text. For a long period of time significant progress has been made in solving increasingly more generalized and dynamic versions of this problem. In this paper we introduce a fully dynamic generalization of the pattern matching problem. We show how to maintain a family of strings under split and concatenation operations. Given a string in the family, all occurrences of it in the family are reported within time $O(\log n \log \log n + occ)$ time, where n is the total size of the strings and *occ* is the number of occurrences. Updates are performed in $O(\log^2 n \log \log n \log^* n)$ time. These bounds are competitive or improve former results for less generalized versions of the problem. As an intermediate result of independent interest, we provide an almost quadratic improvement of the time bounds for the *dynamic string equality* problem due to Mehlhorn, Sundar and Uhrig.

(40) Gerth Stølting Brodal and M. Cristina Pinotti, *Comparator Networks for Binary Heap Construction. Im Stadtwald, D-66123 Saarbrücken, Germany*, technical report MPI-I-98-1-002, 11 pages. Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, January 1998.

Abstract: Comparator networks for constructing binary heaps of size n are presented which have size $O(n \log \log n)$ and depth $O(\log n)$. A lower bound of $n \log \log n - O(n)$ for the size of any heap construction network is also proven, implying that the networks presented are within a constant factor of optimal. We give a tight relation between the leading constants in the size of selection networks and in the size of heap construction networks.

(110) Gerth Stølting Brodal and Jyrki Katajainen, *Worst-Case Efficient External-Memory Priority Queues*. Technical report DIKU Report 97/25, 16 pages. Department of Computer Science, University of Copenhagen, October 1997.

Abstract: A priority queue Q is a data structure that maintains a collection of elements, each

element having an associated priority drawn from a totally ordered universe, under the operations INSERT, which inserts an element into Q, and DELETEMIN, which deletes an element with the minimum priority from Q. In this paper a priority-queue implementation is given which is efficient with respect to the number of block transfers or I/Os performed between the internal and external memories of a computer. Let B and M denote the respective capacity of a block and the internal memory measured in elements. The developed data structure handles any intermixed sequence of INSERT and DELETEMIN operations such that in every disjoint interval of B consecutive priority-queue operations at most $c \log_{M/B}(N/M)$ I/Os are performed, for some positive constant c. These I/Os are divided evenly among the operations: if $B \geq c \log_{M/B}(N/M)$, one I/O is necessary for every $B/(c \log_{M/B}(N/M))$th operation and if $B < c \log_{M/B}(N/M)$, $c/B \cdot \log_{M/B}(N/M)$ I/Os are performed per every operation. Moreover, every operation requires $O(\log_2 N)$ comparisons in the worst case. The best earlier solutions can only handle a sequence of S operations with $O(\sum_{i=1}^{S}(1/B)\log_{M/B}(N_i/M))$ I/Os, where N_i denotes the number of elements stored in the data structure prior to the ith operation, without giving any guarantee for the performance of the individual operations.

(111) Gerth Stølting Brodal, *Finger Search Trees with Constant Insertion Time*. *Im Stadtwald, D-66123 Saarbrücken, Germany*, technical report MPI-I-97-1-020, 17 pages. Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, September 1997.

Abstract: We consider the problem of implementing finger search trees on the pointer machine, *i.e.*, how to maintain a sorted list such that searching for an element x, starting the search at any arbitrary element f in the list, only requires logarithmic time in the distance between x and f in the list.

We present the first pointer-based implementation of finger search trees allowing new elements to be inserted at any arbitrary position in the list in worst case constant time. Previously, the best known insertion time on the pointer machine was $O(\log^* n)$, where n is the total length of the list. On a unit-cost RAM, a constant insertion time has been achieved by Dietz and Raman by using standard techniques of packing small problem sizes into a constant number of machine words.

Deletion of a list element is supported in $O(\log^* n)$ time, which matches the previous best bounds. Our data structure requires linear space.

(44) Gerth Stølting Brodal, Jesper Larsson Träff and Christos D. Zaroliagis, *A Parallel Priority Queue with Constant Time Operations*. *Im Stadtwald, D-66123 Saarbrücken, Germany*, technical report MPI-I-97-1-011, 19 pages. Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, May 1997.

Abstract: We present a parallel priority queue that supports the following operations in constant time: *parallel insertion* of a sequence of elements ordered according to key, *parallel decrease key* for a sequence of elements ordered according to key, *deletion of the minimum key element*, as well as *deletion of an arbitrary element*. Our data structure is the first to support multi insertion and multi decrease key in constant time. The priority queue can be implemented on the EREW PRAM, and can perform any sequence of n operations in $O(n)$ time and $O(m \log n)$ work, m being the total number of keys inserted and/or updated. A main application is a parallel implementation of Dijkstra's algorithm for the single-source shortest path problem, which runs in $O(n)$ time and $O(m \log n)$ work on a CREW PRAM on graphs with n vertices and m edges. This is a logarithmic factor improvement in the running time compared with previous approaches.

117 Gerth Stølting Brodal and Sven Skyum, *The Complexity of Computing the k-ary Composition of a Binary Associative Operator*. Technical report BRICS-RS-96-42, 15 pages. BRICS, Department of Computer Science, Aarhus University, November 1996.

Abstract: We show that the problem of computing all contiguous k-ary compositions of a sequence of n values under an associative and commutative operator requires $3(k-1)/(k+1)n - O(k)$ operations.

For the operator max we show in contrast that in the decision tree model the complexity is $(1 + \Theta(1/\sqrt{k}))n - O(k)$. Finally we show that the complexity of the corresponding on-line problem for the operator max is $(2 - 1/(k-1))n - O(k)$.

(45) Gerth Stølting Brodal, Shiva Chaudhuri and Jaikumar Radhakrishnan, *The Randomized Complexity of Maintaining the Minimum*. Technical report BRICS-RS-96-40, 20 pages. BRICS, Depart-

ment of Computer Science, Aarhus University, November 1996.

Abstract: The complexity of maintaining a set under the operations Insert, Delete and FindMin is considered. In the comparison model it is shown that any randomized algorithm with expected amortized cost t comparisons per Insert and Delete has expected cost at least $n/(e2^{2t}) - 1$ comparisons for FindMin. If FindMin is replaced by a weaker operation, FindAny, then it is shown that a randomized algorithm with constant expected cost per operation exists; in contrast, it is shown that no deterministic algorithm can have constant cost per operation. Finally, a deterministic algorithm with constant amortized cost per operation for an offline version of the problem is given.

(45) Gerth Stølting Brodal, Shiva Chaudhuri and Jaikumar Radhakrishnan, *The Randomized Complexity of Maintaining the Minimum*. Technical report MPI-I-96-1-014. Max-Planck-Institut für Informatik, May 1996.

Abstract: The complexity of maintaining a set under the operations Insert, Delete and Findmin is considered. In the comparison model it is shown that any randomized algorithm with expected amortized cost t comparisons per Insert and Delete has expected cost at least $n/(e2^{2t}) - 1$ comparisons for Findmin. If FindMin is replaced by a weaker operation, FindAny, then it is shown that a randomized algorithm with constant expected cost per operation exists, but no deterministic algorithm. Finally, a deterministic algorithm with constant amortized cost per operation for an offline version of the problem is given.

118 Gerth Stølting Brodal and Thore Husfeldt, *A Communication Complexity Proof that Symmetric Functions have Logarithmic Depth*. Technical report BRICS-RS-96-1, 3 pages. BRICS, Department of Computer Science, Aarhus University, January 1996.

Abstract: We present a direct protocol with logarithmic communication that finds an element in the symmetric difference of two sets of different size. This yields a simple proof that symmetric functions have logarithmic circuit depth.

(47) Gerth Stølting Brodal and Chris Okasaki, *Optimal Purely Functional Priority Queues*. Technical report BRICS-RS-96-37, 27 pages. BRICS, Department of Computer Science, Aarhus University, October 1996.

Abstract: Brodal recently introduced the first implementation of imperative priority queues to support *findMin*, *insert*, and *meld* $O(1)$ worst-case time, and *deleteMin* in $O(\log n)$ worst-case time. These bounds are asymptotically optimal among all comparison-based priority queues. In this paper, we adapt Brodal's data structure to a purely functional setting. In doing so, we both simplify the data structure and clarify its relationship to the binomial queues of Vuillemin, which support all four operations in $O(\log n)$ time. Specifically, we derive our implementation from binomial queues in three steps: first, we reduce the running time of *insert* to $O(1)$ by eliminating the possibility of cascading links; second, we reduce the running time of *findMin* to $O(1)$ by adding a global root to hold the minimum element; and finally, we reduce the running time of *meld* to $O(1)$ by allowing priority queues to contain other priority queues. Each of these steps is expressed using ML-style functors. The last transformation, known as data-structural bootstrapping, is an interesting application of higher-order functors and recursive structures.

(115) Gerth Stølting Brodal, *Fast Meldable Priority Queues*. Technical report BRICS-RS-95-12, 12 pages. BRICS, Department of Computer Science, Aarhus University, February 1995.

Abstract: We present priority queues that support the operations MakeQueue, FindMin, Insert and Meld in worst case time $O(1)$ and Delete and DeleteMin in worst case time $O(\log n)$. They can be implemented on the pointer machine and require linear space. The time bounds are optimal for all implementations where Meld takes worst case time $o(n)$.

To our knowledge this is the first priority queue implementation that supports Meld in worst case constant time and DeleteMin in logarithmic time.

(46) Gerth Stølting Brodal, *Partially Persistent Data Structures of Bounded Degree with Constant Update Time*. Technical report BRICS-RS-94-35, 24 pages. BRICS, Department of Computer Science, Aarhus University, November 1994.

Abstract: The problem of making bounded in-degree and out-degree data structures partially persistent is considered. The node copying method of Driscoll *et al.* is extended so that updates can be performed in *worst-case* constant time on the pointer machine model. Previously it was only known to be possible in amortised constant time.

The result is presented in terms of a new strategy for Dietz and Raman's dynamic two player pebble game on graphs.

It is shown how to implement the strategy and the upper bound on the required number of pebbles is improved from $2b+2d+O(\sqrt{b})$ to $d+2b$, where b is the bound of the in-degree and d the bound of the out-degree. We also give a lower bound that shows that the number of pebbles depends on the out-degree d.

Theses

119 Gerth Stølting Brodal, *Worst Case Efficient Data Structures*. PhD Thesis, technical report BRICS-DS-97-1, x+121 pages. Department of Computer Science, Aarhus University, Denmark, January 1997 (presentation pdf, zip).

Abstract: We study the design of efficient data structures. In particular we focus on the design of data structures where each operation has a worst case efficient implementations. The concrete problems we consider are *partial persistence*, implementation of *priority queues*, and implementation of *dictionaries*.

The first problem we consider is how to make bounded in-degree and out-degree data structures partially persistent, *i.e.*, how to remember old versions of a data structure for later access. A *node copying* technique of Driscoll *et al.* supports update steps in amortized constant time and access steps in worst case constant time. The worst case time for an update step can be linear in the size of the structure. We show how to extend the technique of Driscoll *et al.* such that update steps can be performed in worst case constant time on the pointer machine model.

We present two new comparison based priority queue implementations, with the following properties. The first implementation supports the operations FindMin, Insert and Meld in worst case constant time and Delete and DeleteMin in worst case time $O(\log n)$. The priority queues can be implemented on the pointer machine and require linear space. The second implementation achieves the same worst case performance, but furthermore supports DecreaseKey in worst case constant time. The space requirement is again linear, but the implementation requires auxiliary arrays of size $O(\log n)$. Our bounds match the best known amortized bounds (achieved by respectively binomial queues and Fibonacci heaps). The data structures presented are the first achieving these worst case bounds, in particular supporting Meld in worst case constant time. We show that these time bounds are optimal for all implementations supporting Meld in worst case time $o(n)$. We also present a tradeoff between the update time and the query time of comparison based priority queue implementations. Finally we show that any randomized implementation with expected amortized cost t comparisons per Insert and Delete operation has expected cost at least $n/2^{O(t)}$ comparisons for FindMin.

Next we consider how to implement priority queues on parallel (comparison based) models. We present time and work optimal priority queues for the CREW PRAM, supporting FindMin, Insert, Meld, DeleteMin, Delete and DecreaseKey in constant time with $O(\log n)$ processors. Our implementation is the first supporting all of the listed operations in constant time. To be able to speed up Dijkstra's algorithm for the single-source shortest path problem we present a different parallel priority data structure. With this specialized data structure we give a parallel implementation of Dijkstra's algorithm which runs in $O(n)$ time and performs $O(m \log n)$ work on a CREW PRAM. This represents a logarithmic factor improvement for the running time compared with previous approaches.

We also consider priority queues on a RAM model which is stronger than the comparison model. The specific problem is the maintenance of a set of n integers in the range $0..2^w-1$ under the operations Insert, Delete, FindMin, FindMax and Pred (predecessor query) on a unit cost RAM with word size w bits. The RAM operations used are addition, left and right bit shifts, and bit-wise boolean operations. For any function $f(n)$ satisfying $\log\log n \leq f(n) \leq \sqrt{\log n}$, we present a data structure supporting FindMin and FindMax in worst case constant time, Insert

and DELETE in worst case $O(f(n))$ time, and PRED in worst case $O((\log n)/f(n))$ time. This represents the first priority queue implementation for a RAM which supports INSERT, DELETE and FINDMIN in worst case time $O(\log \log n)$ — previous bounds were only amortized. The data structure is also the first dictionary implementation for a RAM which supports PRED in worst case $O(\log n/\log \log n)$ time while having worst case $O(\log \log n)$ update time. Previous sublogarithmic dictionary implementations do not provide for updates that are significantly faster than queries. The best solutions known support both updates and queries in worst case time $O(\sqrt{\log n})$.

The last problem consider is the following dictionary problem over binary strings. Given a set of n binary strings of length m each, we want to answer d–queries, *i.e.*, given a binary query string of length m to report if there exists a string in the set within Hamming distance d of the query string. We present a data structure of size $O(nm)$ supporting 1–queries in time $O(m)$ and the reporting of all strings within Hamming distance 1 of the query string in time $O(m)$. The data structure can be constructed in time $O(nm)$. The implementation presented is the first achieving these optimal time bounds for the preprocessing of the dictionary and for 1–queries. The data structure can be extended to support the insertion of new strings in amortized time $O(m)$.

120 Gerth Stølting Brodal, *Complexity of Data Structures*. Progress Report, 29 pages. Department of Computer Science, Aarhus University, Denmark, November 1994 (presentation pdf, zip).

Abstract: This progress report presents the work accomplished by the author during part A of the Ph.D. programme at the University of Aarhus.

We consider the complexity of different data structures, and introduce the distinction between *query* and *restructuring* complexity of data structures. In the light of three different computational frameworks we argue that data structures should be designed to have minimal restructuring complexity.

The main result is the result of [3] where we show how to make bounded degree data structures partially persistent with worst case slowdown in $O(1)$. We also give a restricted result for the case of fully persistent data structures.

Reference [3] is appended at the end of the report.

Submissions

121 Gerth Stølting Brodal, Casper Moldrup Rysgaard and Rolf Svenning, *Buffered Partially-Persistent External-Memory Search Trees*. Submitted to *33rd Annual European Symposium on Algorithms, 24 April 2025*.

122 Gerth Stølting Brodal, Michael T. Goodrich, John Iacono, Jared Lo, Ulrich Meyer, Victor Pagan, Nodari Sitchinava and Rolf Svenning, *External-Memory Priority Queues with Optimal Insertions*. Submitted to *33rd Annual European Symposium on Algorithms, 24 April 2025*.

123 Gerth Stølting Brodal, Michale T. Goodrich, Ryuto Kitagawa, Nodari Sitchinava and Rolf Svenning, *Orgainic Mergesort and Finger Buffer-Tree Sort: Adaptive Sorting Algorithms with External-Memory or Parallel Implementations*. Submitted to *33rd Annual European Symposium on Algorithms, 24 April 2025*.

124 Peyman Afshani, Gerth Stølting Brodal and Nodari Sitchinava, *The Impossibility of Simultaneous Time and I/O Optimality for The Planar Maxima and Convex Hull Problems*. Submitted to *65th Annual Symposium on Foundations of Computer Science (FOCS), 3 April 2025*.

125 Gerth Stølting Brodal, *Bottom-up Rebalancing Binary Search Trees by Flipping a Coin*. Submitted to *Theoretical Computer Science, Special issue on 12th International Conference on Fun With Algorithms, 15 February 2025*.

Coauthors (102)

Peyman Afshani, Pankaj K. Agarwal, Stephen Alstrup, Lars Arge, Djamal Belazzougui, Michael A. Bender, Edvin Berglin, Bruce Brewer, Andrej Brodnik, Shiva Chaudhuri, Pooya Davoodi, Erik D. Demaine, Rolf Fagerberg, Jeremy T. Fineman, Irene Finocchi, Dongdong Ge, Loukas Georgiadis, Beat Gfeller, Michael T. Goodrich, Michale T. Goodrich, Fabrizio Grandoni, Mark Greve, Leszek Gąsieniec, Inge Li Gørtz, David Hammer, 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, Ryuto Kitagawa, Lars Michael Kristensen, Martin Kutz, George Lagogiannis, Stefan Langerman, Kasper Green Larsen, Morten Laustsen, Moshe Lewenstein, Jared Lo, Rune Bang Lyngsø, Alejandro López-Ortiz, Thomas Mailund, Christos Makris, Konstantinos Mampentzidis, Ulrich Meyer, Gabriel Moruz, J. Ian Munro, Thomas Mølhave, Andrei Negoescu, Jesper Sindahl Nielsen, Chris Okasaki, Victor Pagan, Vineet Pandey, Apostolos Papadopoulos, Christian Nørgaard Storm Pedersen, Manuel Penschuck, Derek Phillips, M. Cristina Pinotti, Jaikumar Radhakrishnan, Rajeev Raman, S. Srinivasa Rao, Theis Rauhe, Casper Moldrup Rysgaard, Andreas Sand, Peter Sanders, Jens Kristian Refsgaard Schou, Spyros Sioutas, Nodari Sitchinava, Sven Skyum, Venkatesh Srinivasan, Martin Stissing, Jens Stoye, Rolf Svenning, Robert E. Tarjan, Laura Toma, Hung Tran, Jakob Truelsen, Jesper Larsson Träff, Athanasios Tsakalidis, Konstantinos Tsakalidis, Kostas Tsichlas, Constantinos Tsirogiannis, Elias Vicari, Kristoffer Vinther, Jeff Vitter, Haitao Wang, Michael Westergaard, Sebastian Wild, Christos D. Zaroliagis, Norbert Zeh, Anna Östlin.

Program Committees

2025 *50th International Symposium on Mathematical Foundations of Computer Science* (*MFCS*). Warsaw, Poland, 25–29 Aug 2025.

20th International Workshop on Algorithms and Data Structures (*WADS*). Toronto, Canada, 11–13 Aug 2025.

41st International Symposium on Computational Geometry (*SoCG*). Kanazawa, Japan, 23–27 Jun 2025.

50th International Conference on Current Trends in Theory and Practice of Computer Science (*SOFSEM*). Bratislava, Slovak Republic, 20–23 Jan 2025.

2024 *32nd Annual European Symposium on Algorithms* (*ESA*). Royal Holloway, University of London in Egham, United Kingdom, 22–4 Sep 2024.

26th Workshop on Algorithm Engineering and Experiments (*ALENEX*). Alexandria, Virginia, USA, 7–8 Jan 2024.

2023 *21st International Symposium on Experimental Algorithms* (*SEA*). Barcelona, Spain, 24–26 Jul 2023.

2022 *24th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Alexandria, Virginia, USA, 9–10 Jan 2022.

2021 *29th Annual European Symposium on Algorithms (Track B - Algorithm Engineering)* (*ESA*). Lisbon, Portugal, 6–8 Sep 2021.

2019 *30th International Workshop on Combinatorial Algorithms* (*IWOCA*). Pisa, Italy, 23–25 Jul 2019.

46th International Colloquium on Automata, Languages, and Programming (*ICALP*). Patras, Greece, 8–12 Jul 2019.

36th Annual Symposium on Theoretical Aspects of Computer Science (*STACS*). Berlin, Germany, 13–16 Mar 2019.

2017 *43rd International Conference on Current Trends in Theory and Practice of Computer Science* (*SOFSEM*). Limerick, Ireland, 16–20 Jan 2017.

28th Annual ACM-SIAM Symposium on Discrete Algorithms (*SODA*). Barcelona, Spain, 16–19 Jan 2017.

2016 *18th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Arlington, Virginia, USA, 10 Jan 2016.

2015 *26th International Workshop on Combinatorial Algorithms* (*IWOCA*). Verona, Italy, 5–7 Oct 2015.

31st European Workshop on Computational Geometry (*EuroCG*). Ljubljana, Slovenia, 16–18 Mar 2015.

2014 *25th Annual International Symposium on Algorithms and Computation* (*ISAAC*). Jeonju, Korea, 15–17 Dec 2014.

25th International Workshop on Combinatorial Algorithms (*IWOCA*). Duluth, Minnesota, USA, 15–17 Oct 2014.

6th Workshop on Massive Data Algorithmics (*MASSIVE*). Wroclaw, Poland, 11–11 Sep 2014.

39th International Symposium on Mathematical Foundations of Computer Science (*MFCS*). Budapest, Hungary, 25–29 Aug 2014.

41st International Colloquium on Automata, Languages, and Programming (*ICALP*). IT University of Copenhagen, Copenhagen, Denmark, 8–11 Jul 2014.

25th Annual ACM-SIAM Symposium on Discrete Algorithms (*SODA*). Portland, Oregon, USA, 5–7 Jan 2014.

2013 *5th Workshop on Massive Data Algorithmics* (*MASSIVE*). Sophia Antipolis, France, 5–5 Sep 2013.

19th International Symposium on Fundamentals of Computation Theory (*FCT*). Liverpool, United Kingdom, 19–23 Aug 2013.

24rd International Workshop on Combinatorial Algorithms (*IWOCA*). Rouen, Normandy, France, 10–12 Jul 2013.

7th International Conference on Language and Automata Theory and Applications (*LATA*). Bilbao, Spain, 2–5 Apr 2013.

2012 *4th Workshop on Massive Data Algorithmics* (*MASSIVE*). Ljulbjana, Slovenia, 13–13 Sep 2012.

23rd International Workshop on Combinatorial Algorithms (*IWOCA*). Kalasalingam University, Tamil Nadu, India, 19–21 Jul 2012.

14th Workshop on Algorithm Engineering and Experiments (*ALENEX*). Kyoto, Japan, 16 Jan 2012.

2011 *22nd International Workshop on Combinatorial Algorithms* (*IWOCA*). University of Victoria, Victoria, British Columbia, Canada, 20–22 Jun 2011.

3rd Workshop on Massive Data Algorithmics (*MASSIVE*). Paris, France, 16–16 Jun 2011.

2010 *18th Annual European Symposium on Algorithms* (*ESA*). Liverpool, UK, 6–8 Sep 2010.

21st International Workshop on Combinatorial Algorithms (*IWOCA*). King's College London, UK, 26–28 Jul 2010.

2nd Workshop on Massive Data Algorithmics (*MASSIVE*). Snowbird, Utah, 17–17 Jun 2010.

12th Scandinavian Workshop on Algorithm Theory (*SWAT*). Bergen, Norway, Jun 2010.

9th Latin American Symposium on Theoretical Informatics (*LATIN*). Oaxaca, Mexico, 19–23 Apr 2010.

2009 *11th International Workshop on Algorithms and Data Structures* (*WADS*). Banff, Alberta, Canada, 21–23 Aug 2009.

21st ACM Symposium on Parallelism in Algorithms and Architectures (*SPAA*). Calgary, Canada, Aug 2009.

36th International Colloquium on Automata, Languages and Programming (*ICALP*). Rhodes, Greece, 5–12 Jul 2009.

Co-chair of program committee. *1st Workshop on Massive Data Algorithmics* (*MASSIVE*). Aarhus, Denmark, 11–11 Jun 2009.

8th International Symposium on Experimental Algorithms (*SEA*). Dortmund, Germany, 3–6 Jun 2009.

2008 *7th International Workshop on Experimental Algorithms* (*WEA*). Provincetown, Cape Cod, Massachusetts, USA, 30 May – 2008.

2007 *IEEE 2007 International Symposium on Parallel and Distributed Processing with Applications* (*ISPA*). Niagara Falls, Ontario, Canada, 21–24 Aug 2007.

10th International Workshop on Algorithms and Data Structures (*WADS*). Halifax, Canada, 15–17 Aug 2007.

International Workshop on Algorithmic Topics in Constraint Programming (cancelled) (*ATCP*). Wroclaw, Poland, 8 Jul 2007.

24th Annual Symposium on Theoretical Aspects of Computer Science (*STACS*). Aachen, Germany, 22–24 Feb 2007.

18th Annual ACM-SIAM Symposium on Discrete Algorithms (*SODA*). New Orleans, Louisiana, USA, 7–9 Jan 2007.

Co-chair of program committee. *9th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). New Orleans, Louisiana, USA, 6 Jan 2007.

2006 *13th Symposium on String Processing and Information Retrieval (SPIRE)* (*SPIRE*). Glasgow, Scotland, 11–13 Oct 2006.

9th Scandinavian Workshop on Algorithm Theory (*SWAT*). Riga, Latvia, 6–8 Jul 2006.

2005 *37th Annual ACM Symposium on Theory of Computing* (*STOC*). Baltimore, Maryland, USA, 22–24 May 2005.

4th International Workshop on Efficient and Experimental Algorithms (*WEA*). Santorini Island, Greece, 10–13 May 2005.

2004 *24th Conference on Foundations of Software Technology and Theoretical Computer Science* (*FSTTCS*). Chennai, India, 16–18 Dec 2004.

Co-chair of program committee. *13th Annual European Symposium on Algorithms – Engineering and Application Track* (*ESA*). Mallorca, Spain, Oct 2004.

31st International Colloquium on Automata, Languages and Programming (*ICALP*). Turku, Finland, 12–16 Jul 2004.

15th Annual Symposium on Combinatorial Pattern Matching (*CPM*). Istanbul, Turkey, 5–7 Jul 2004.

3rd International Conference on Fun With Algorithms (*FUN*). Isola d'Elba, Tuscany, Italy, 26–28 May 2004.

6th Latin American Symposium on Theoretical Informatics (*LATIN*). Buenos Aires, Argentina, 5–9 Apr 2004.

15th Annual ACM-SIAM Symposium on Discrete Algorithms (*SODA*). New Orleans, Louisiana, USA, 11–13 Jan 2004.

2003 *8th International Workshop on Algorithms and Data Structures* (*WADS*). Ottawa, Canada, 30 Jul – 1 Aug 2003.

Co-chair of program committee. *Workshop on Algorithms for Massive Data Sets (cancelled)* (*MASSIVE*). Eindhoven, The Netherlands, 29 Jun 2003.

11th Euromicro Conference on Parallel Distributed and Networking based Processing, Special session on Memory Hierachies (*PDP*). Genoa, Italy, 5–7 Feb 2003.

5th Workshop on Algorithm Engineering and Experiments (*ALENEX*). Baltimore, MD, USA, 10–11 Jan 2003.

2001 *9th Annual European Symposium on Algorithms* (*ESA*). Aarhus, Denmark, 28–31 Aug 2001.

1999 *Workshop on Algorithmic Aspects of Advanced Programming Languages* (*WAAAPL*). Paris, France, 30 Sep 1999.

Invited Speaker

Sep 2023 Algorithm Engineering the Theory. *Summer School on Algorithm Engineering for Network Problems* (*ADYN*). Hasso Plattner Institute, Potsdam, Germany, 19–22 Sep 2023 (presentation pdf, pptx).

Jan 2023 Data Structure Design: Theory and Practice. *48th International Conference on Current Trends in Theory and Practice of Computer Science* (*SOFSEM*). Nový Smokovec, Slovakia, 15–18 Jan 2023 (presentation pdf, pptx).

Oct 2021 In Memoriam - Lars Arge. *Lars Arge Memorial Symposium.* Aarhus University, Aarhus, Denmark, 6 Oct 2021 (presentation pdf, pptx).

Jan 2021 In Memoriam - Lars Arge. *23rd Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Virtual conference. Originally scheduled in Alexandria, Virginia, USA, 11 Jan 2021 (presentation pdf, pptx, mp4).

Aug 2008 Lectures on lower bounds and string algorithms. *MADALGO Summer School on Cache Oblivious Algorithms.* MADALGO, Aarhus University, Denmark, 17–21 Aug 2008.

Jun 2008 Word RAM algorithms. *International PhD School in Algorithms for Advanced Processor Architectures* (*AFAPA*). IT University of Copenhagen, Denmark, 9–12 Jun 2008 (presentation

pdf, pptx).

Jul 2004 Cache-Oblivious Algorithms and Data Structures. *9th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Louisiana Museum of Modern Art, Humlebæk, Denmark, 8–10 Jul 2004 (presentation pdf, zip).

Dec 1999 Regularities in Sequences. *Workshop on Advances in Data Structures, preworkshop of the 19th Annual Conference on Foundations of Software Technology and Theoretical Computer Science* (*FSTTCS*). Chennai, India, 11–12 Dec 1999 (presentation pdf, zip).

Scientific visits

Jun 2023 Leszek Gąsieniec and Sebastian Wild. University of Liverpool, Liverpool, UK.

Jan 2023 Leszek Gąsieniec and Sebastian Wild. University of Liverpool, Liverpool, UK.

Jun 2013 Riko Jacob. ETH Zürich, Zürich, Switzerland.

Nov 2012 Riko Jacob. ETH Zürich, Zürich, Switzerland.

Nov 2011 Andy Brodknik. University of Primorska and University of Ljubljana, Koper and Ljubljana, Slovenia.

Nov 2009 Riko Jacob. Technische Universität München, Munich, Germany.

May 2002 Michiel Smid. School of Computer Science, Carleton University, Ottawa, Ontario, Canada.

Oct 2001 Athanasios K. Tsakalidis. Computer Technology Institute, Patras, Greece.

Jan 1999 Lars Arge. Department of Computer Science, Duke University, Durham, North Carolina, USA.

Jan 1999 Jyrki Katajainen. Department of Computer Science, University of Copenhagen, Copenhagen, Denmark.

May 1998 Ian Munro. Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.

May 1998 Lars Arge. Department of Computer Science, Duke University, Durham, North Carolina, USA.

Nov 1997 M. Cristina Pinotti. Instituto di Elaborazione della Informazione, CNR, Pisa, Pisa, Italy.

Jan 1996 Haim Kaplan. Princeton University/DIMACS, Princeton, NJ, USA.

Sep 1995 – May 1996 Kurt Mehlhorn. Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Aug 1995 Peter Bro Miltersen. Toronto University, Toronto, Ontario, Canada.

Conference and Workshop Participation

May 2025 *Dagstuhl Seminar on "Adaptive and Scalable Data Structures"*. Dagstuhl, Germany, 4–9 May 2025 (presentation pdf, pptx).

Apr – May 2025 *Dagstuhl Seminar on "Learned Predictions for Data Structures and Running Time"*. Dagstuhl, Germany, 27 Apr – 2 May 2025.

Mar 2025 *3rd Hawaii Workshop on Parallel Algorithms and Data Structures* (*ALGOPARC*). Honolulu, Hawaii, USA, 17–21 Mar 2025.

Jan 2025 *36th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). New Orleans, Louisiana, USA, 12–15 Jan 2025.

Sep 2024 *32nd Annual European Symposium on Algorithms (ALGO 2024)* (*ESA*). Royal Holloway, University of London, Egham, United Kingdom, 2–4 Sep 2024.

Jun 2024 *19th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Helsinki, Finland, 13–14 Jun 2024 (presentation pdf, pptx).

Jun 2024 *12th International Conference on Fun With Algorithms* (*FUN*). Island of La Madallena, Sardinia, Italy, 4–8 Jun 2024 (presentation pdf, pptx).

Apr 2024 *31st ARCO Workshop (Algorithmic Research-Cooperation around Oresound)* (*ARCO*). Royal Danish Academy of Sciences and Letters, Copenhagen, Denmark, 12 Apr 2024.

Jan 2024 *35th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Alexandria, Virginia, USA, 7–10 Jan 2024.

Nov – Dec 2023 *Scientific Symposium - 30 Years Max Planck Institute for Informatics.* Max Planck Institute for Informatik, Saarbrücken, Germany, 30 Nov – 1 Dec 2023.

Nov 2023 *30th ARCO Workshop (Algorithmic Research-Cooperation around Oresound)* (*ARCO*). University of Southern Denmark, Odense, Denmark, 24 Nov 2023.

Sep 2023 Algorithm Engineering the Theory. *Summer School on Algorithm Engineering for Network Problems* (*ADYN*). Hasso Plattner Institute, Potsdam, Germany, 19–22 Sep 2023 (presentation pdf, pptx).

Sep 2023 *31st Annual European Symposium on Algorithms (ALGO 2023)* (*ESA*). Centrum Wiskunde & Informatica (CWI), Amsterdam, the Netherlands, 4–6 Sep 2023.

Jul – Aug 2023 *18th International Workshop on Algorithms and Data Structures* (*WADS*). Montreal, QC, Canada, 31 Jul – 2 Aug 2023.

May 2023 *Dagstuhl Seminar on "Scalable Data Structures" (co-organizer).* Dagstuhl, Germany, 21–26 May 2023.

Jan 2023 *34th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Florenze, Italy, 22–25 Jan 2023.

Jan 2023 Data Structure Design: Theory and Practice. *48th International Conference on Current Trends in Theory and Practice of Computer Science* (*SOFSEM*). Nový Smokovec, Slovakia, 15–18 Jan 2023 (presentation pdf, pptx).

Jun 2022 *Final meeting DFG Priority Progamme "Algorithms for Big Data".* Goethe University, Frankfurt am Main, Germany, 9–10 Jun 2022.

May – Jun 2022 *10th and 11th International Conference on Fun With Algorithms* (*FUN*). Island of Favignana, Sicily, Italy, 30 May – 3 Jun 2022 (presentation pdf, pptx).

Jan 2022 *33rd Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Virtual conference. Originally scheduled in Alexandria, Virginia, USA, 9–12 Jan 2022.

Oct 2021 In Memoriam - Lars Arge. *Lars Arge Memorial Symposium.* Aarhus University, Aarhus, Denmark, 6 Oct 2021 (presentation pdf, pptx).

Sep 2021 *29th Annual European Symposium on Algorithms (ALGO 2021)* (*ESA*). Virtual conference. Originally scheduled in Lisbon, Portugal, 6–10 Sep 2021.

Jun 2021 *19th International Symposium on Experimental Algorithms* (*SEA*). Virtual conference. Originally scheduled in Université Côte d'Azur, Valrose, France, 7–9 Jun 2021.

Jan 2021 In Memoriam - Lars Arge. *23rd Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Virtual conference. Originally scheduled in Alexandria, Virginia, USA, 11 Jan 2021 (presentation pdf, pptx, mp4).

Jan 2021 *32nd Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Virtual conference. Originally scheduled in Alexandria, Virginia, USA, 10–13 Jan 2021.

Jan 2020 *31st Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Salt Lake City, Utah, USA, 5–8 Jan 2020.

Dec 2019 *2nd Hawaii Workshop on Parallel Algorithms and Data Structures* (*ALGOPARC*). Honolulu, Hawaii, USA, 9–13 Dec 2019.

Jan – Feb 2019 *Dagstuhl Seminar on "Data Structures for the Cloud and External Memory Data" (co-organizer).* Dagstuhl, Germany, 27 Jan – 1 Feb 2019.

Jan 2019 *30th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). San Diego, California, USA, 6–9 Jan 2019.

Aug 2018 *26th Annual European Symposium on Algorithms (ALGO 2018)* (*ESA*). Helsinki, Finland, 19–24 Aug 2018.

Jan 2018 *29th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). New Orleans, Louisiana, USA, 7–10 Jan 2018.

Nov 2017 *Searching in Trees. Workshop on The Art of Data Structures in honor of Prof. Athana-*

sios Tsakalidis (picture of participants). University of Patras, Patras, Greece, 7 Nov 2017 (presentation pdf, pptx).

Sep 2017 *25th Annual European Symposium on Algorithms (ALGO 2017)* (*ESA*). Vienna, Austria, 3–8 Sep 2017.

Jan 2017 *28th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Barcelona, Spain, 16–19 Jan 2017.

Aug 2016 *8th Workshop on Massive Data Algorithmics* (*MASSIVE*). Aarhus, Denmark, 23 Aug 2016 (presentation pdf, pptx).

Feb 2016 *33rd Annual Symposium on Theoretical Aspects of Computer Science* (*STACS*). Orléans, France, 17–20 Feb 2016 (presentation pdf, pptx).

Jan 2016 *27th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Washington, DC, USA, 10–12 Jan 2016.

Sep – Oct 2014 *Algorithms for Big Data*. Goethe University, Frankfurt am Main, Germany, 29 Sep – 1 Oct 2014.

Sep 2014 *6th Workshop on Massive Data Algorithmics* (*MASSIVE*). Wroclaw, Poland, 11 Sep 2014.

Sep 2014 *22nd Annual European Symposium on Algorithms* (*ESA*). Wroclaw, Poland, 8–12 Sep 2014.

Jul 2014 *14th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Copenhagen, Denmark, 2–4 Jul 2014 (presentation pdf, pptx).

Jan 2014 *25th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Portland, Oregon, USA, 4–6 Jan 2014.

Sep 2013 *Dagstuhl Seminar on "Algorithm Engineering"*. Dagstuhl, Germany, 22–27 Sep 2013.

Sep 2013 *5th Workshop on Massive Data Algorithmics* (*MASSIVE*). Inria, Sophia Antipolis, France, 5 Sep 2013.

Sep 2013 *21st Annual European Symposium on Algorithms* (*ESA*). Inria, Sophia Antipolis, France, 2–4 Sep 2013 (presentation pdf, pptx).

Aug 2013 *(A Survey on) Priority Queues*. *Conference on Space Efficient Data Structures, Streams and Algorithms – In Honor of J. Ian Munro on the Occasion of His 66th Birthday* (*IanFest*). University of Waterloo, Waterloo, Ontario, Canada, 15–16 Aug 2013 (presentation pdf, pptx).

Aug 2013 *13th International Workshop on Algorithms and Data Structures* (*WADS*). University of Western Ontario, London, Ontario, Canada, 12–14 Aug 2013.

Jan 2013 *24th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). New Orleans, Louisiana, USA, 6–8 Jan 2013 (presentation pdf, pptx).

Sep 2012 *4th Workshop on Massive Data Algorithmics* (*MASSIVE*). Ljubljana, Slovenia, 13 Sep 2012 (presentation pdf, pptx).

Sep 2012 *20th Annual European Symposium on Algorithms* (*ESA*). Ljubljana, Slovenia, 10–12 Sep 2012.

May 2012 *44th Annual ACM Symposium on Theory of Computing* (*STOC*). New York, New York, USA, 19–22 May 2012 (presentation pdf, pptx).

Jan 2012 *23rd Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Kyoto, Japan, 17–19 Jan 2012 (presentation pdf, pptx).

Jan 2012 *14th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Kyoto, Japan, 16 Jan 2012.

Aug 2011 *12th International Workshop on Algorithms and Data Structures* (*WADS*). Polytechnic Institute of New York University, Brooklyn, NY, USA, 15–17 Aug 2011.

Jun 2011 *3rd Workshop on Massive Data Algorithmics* (*MASSIVE*). Paris, France, 16 Jun 2011.

Jan 2011 *22nd Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). San Francisco, CA, USA, 23–25 Jan 2011.

Jan 2011 *13th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). San Francisco, CA, USA, 22 Jan 2011.

Sep 2010 *18th Annual European Symposium on Algorithms* (*ESA*). Liverpool, United Kingdom, 6–8

Sep 2010 (presentation pdf, pptx).

Jun 2010 *2nd Workshop on Massive Data Algorithmics* (*MASSIVE*). Snowbird, Utah, USA, 17 Jun 2010 (presentation pdf, pptx).

Jan 2010 *21st Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Austin, TX, USA, 17–19 Jan 2010.

Jan 2010 *11th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Austin, TX, USA, 16–16 Jan 2010.

Sep 2009 *17th Annual European Symposium on Algorithms* (*ESA*). Copenhagen, Denmark, 7–9 Sep 2009.

Jun 2009 *Workshop on Massive Data Algorithmics* (*MASSIVE*). Aarhus, Denmark, 11 Jun 2009.

Jun 2009 *25th Annual ACM Symposium on Computational Geometry* (*SoCG*). Aarhus, Denmark, 8–10 Jun 2009.

Mar 2009 *Cache-Oblivious Algorithms A Unified Approach to Hierarchical Memory Algorithms. Current Trends in Algorithms, Complexity Teory, and Cryptography* (*CTACC*). Tsinghua University, Beijing, China, 22–22 Mar 2009 (presentation pdf, pptx).

Aug 2008 Lectures on lower bounds and string algorithms. *MADALGO Summer School on Cache Oblivious Algorithms*. MADALGO, Aarhus University, Denmark, 17–21 Aug 2008.

Jun 2008 Word RAM algorithms. *International PhD School in Algorithms for Advanced Processor Architectures* (*AFAPA*). IT University of Copenhagen, Denmark, 9–12 Jun 2008 (presentation pdf, pptx).

Jan 2008 *19th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). San Francisko, CA, USA, 20–22 Jan 2008.

Jan 2008 *10th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). San Francisko, CA, USA, 19–19 Jan 2008.

Sep – Oct 2007 *3rd Bertinoro Workshop on Algorithms and Data Structures* (*ADS*). Bertinoro, Forlì, Italy, 30 Sep – 5 Oct 2007 (presentation pdf, ppt).

Aug 2007 *32nd International Symposium on Mathematical Foundations of Computer Science* (*MFCS*). Cesky Krumlov, Czech Republic, 26–31 Aug 2007 (presentation pdf, ppt).

Jan 2007 *18th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). New Orleans, Louisiana, USA, 7–7 Jan 2007.

Jan 2007 *9th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). New Orleans, Louisiana, USA, 6 Jan 2007.

Sep 2006 *14th Annual European Symposium on Algorithms* (*ESA*). Zürich, Switzerland, 11–13 Sep 2006 (presentation pdf, ppt).

Jul 2006 *10th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Riga, Latvia, 6–8 Jul 2006.

Jun 2006 *Workshop on Space-Conscious Algorithms* (*Bertinoro06*). Bertinoro, Forlì, Italy, 10–15 Jun 2006 (presentation pdf, zip).

Jan 2006 *17th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Miami, Florida, USA, 22–24 Jan 2006 (presentation pdf, zip).

Jan 2006 *8th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Miami, Florida, USA, 21 Jan 2006.

Oct 2005 *13th Annual European Symposium on Algorithms (ALGO 2005)* (*ESA*). Palma de Mallorca, Mallorca, Spain, 3–6 Oct 2005.

May – Jun 2005 *Algorithms and Data Structures* (*ADS*). Bertinoro, Forlì, Italy, 29 May – 4 Jun 2005 (presentation pdf, zip).

Jan 2005 *16th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Vancouer, British Columbia, Canada, 23–25 Jan 2005.

Sep 2004 *12th Annual European Symposium on Algorithms (ALGO 2004)* (*ESA*). Bergen, Norway, 14–17 Sep 2004.

Jul 2004 *Dagstuhl Seminar on "Cache-Oblivious and Cache-Aware Algorithms"*. Dagstuhl, Germany, 18–23 Jul 2004.

Jul 2004 Cache-Oblivious Algorithms and Data Structures. *9th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Louisiana Museum of Modern Art, Humlebæk, Denmark, 8–10 Jul 2004 (presentation pdf, zip).

Jan 2004 *15th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). New Orleans, Louisiana, USA, 11–13 Jan 2004.

Jan 2004 *6th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). New Orleans, Louisiana, USA, 10–10 Jan 2004 (presentation pdf, zip).

Jun 2003 *Algorithms and Data Structures* (*ADS*). Bertinoro, Forlì, Italy, 23–27 Jun 2003 (presentation pdf, zip).

Jan 2003 *14th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Baltimore, Maryland, USA, 12–14 Jan 2003 (presentation pdf, zip).

Jan 2003 *5th Workshop on Algorithm Engineering and Experiments* (*ALENEX*). Baltimore, Maryland, USA, 11–11 Jan 2003.

Nov 2002 *13th Annual International Symposium on Algorithms and Computation* (*ISAAC*). Vancouver, British Columbia, Canada, 21–23 Nov 2002 (presentation pdf, zip).

Nov 2002 *43rd Annual Symposium on Foundations of Computer Science* (*FOCS*). Vancouver, British Columbia, Canada, 16–19 Nov 2002.

Nov 2002 *Workshop on Algorithms and Models for the Web-Graph* (*WAW*). Vancouver, British Columbia, Canada, 16 Nov 2002.

May 2002 *34th Annual ACM Symposium on Theory of Computing* (*STOC*). Montréal, Québec, Canada, 19–21 May 2002.

Aug 2001 *ALGO 2001 (9th Annual European Symposium on Algorithms, 5th Workshop on Algorithm Engineering, and 1st Workshop on Algorithms in BioInformatics)* (*ALGO*). Aarhus, Denmark, 27–31 Aug 2001.

Jul 2001 *28th International Colloquium on Automata, Languages, and Programming* (*ICALP*). Hersonissos, Crete, Greece, 8–12 Jul 2001.

Jul 2001 *33rd Annual ACM Symposium on Theory of Computing* (*STOC*). Hersonissos, Crete, Greece, 6–8 Jul 2001 (presentation pdf, zip).

Sep 2000 *Dagstuhl Seminar on "Experimental Algorithmics"*. Dagstuhl, Germany, 10–15 Sep 2000.

Sep 2000 *Alcom-FT Meeting*. Saarbrücken, Germany, 9 Sep 2000.

Jul 2000 *7th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Bergen, Norway, 5–7 Jul 2000.

Dec 1999 *19th Annual Conference on Foundations of Software Technology and Theoretical Computer Science* (*FSTTCS*). Chennai, India, 13–15 Dec 1999.

Dec 1999 Regularities in Sequences. *Workshop on Advances in Data Structures, preworkshop of the 19th Annual Conference on Foundations of Software Technology and Theoretical Computer Science* (*FSTTCS*). Chennai, India, 11–12 Dec 1999 (presentation pdf, zip).

Jan 1999 *10th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Baltimore, Maryland, USA, 17–19 Jan 1999.

Jan 1999 *1st Workshop on Algorithm Engineering and Experimentation* (*ALENEX*). Baltimore, Maryland, USA, 15–16 Jan 1999.

Jul 1998 *6th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Stockholm, Sweden, 8–10 Jul 1998 (presentation pdf, zip, pdf, zip).

May 1998 *DIMACS Workshop on External Memory Algorithms and Visualization* (*DIMACS Workshop*). Piscataway, New Jersey, USA, 20–22 May 1998.

Jan 1998 *9th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). San Francisko, California, USA, 25–27 Jan 1998 (presentation pdf, zip).

Jan 1998 *ALCOM-IT Review Meeting* (*ALCOM-IT*). Saarbrücken, Germany, 15–17 Jan 1998.

Sep 1997 *11th International Workshop on Distributed Algorithms* (*WDAG*). Saarbrücken, Germany, 24–26 Sep 1997.

Sep 1997 *International School On Distributed Computing and Systems - ALCOM-IT SODICS* (*SODICS*). Saarbrücken, Germany, 21–23 Sep 1997.

May 1997 *Algorithms for Future Technologies - ALTEC'97* (*ALTEC*). Saarbrücken, Germany, 21–24 May 1997.

Feb – Mar 1997 *14th Annual Symposium on Theoretical Aspects of Computer Science* (*STACS*). Lübeck, Germany, 27 Feb – 1 Mar 1997 (presentation pdf).

Jul 1996 *5th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Reykjavik, Iceland, 3–5 Jul 1996 (presentation pdf, zip, pdf, zip).

Jun 1996 *BRICS Strategy Workshop*. BRICS, Department of Computer Science, Aarhus University, 5–6 Jun 1996.

Jan 1996 *7th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*). Atlanta, Georgia, USA, 28–30 Jan 1996 (presentation pdf, zip).

Aug 1995 *4th International Workshop on Algorithms and Data Structures* (*WADS*). Kingston, Ontario, Canada, 16–18 Aug 1995 (presentation pdf, zip).

Mar 1995 *12th Annual Symposium on Theoretical Aspects of Computer Science* (*STACS*). München, Germany, 2–4 Mar 1995.

Sep 1994 *2nd Annual European Symposium on Algorithms* (*ESA*). Utrecht, The Netherlands, 28–28 Sep 1994.

Aug 1994 *Complexity Theory: Present and Future*. Aarhus, Denmark, 15–18 Aug 1994.

Jul 1994 *4th Scandinavian Workshop on Algorithm Theory* (*SWAT*). Aarhus, Denmark, 6–8 Jul 1994.

Sep – Oct 1993 *1st Annual European Symposium on Algorithms* (*ESA*). Bad Honnef, Germany, 30 Sep – 2 Oct 1993.

Talks

Feb 2025 *Algoritmer*. U-days, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Aug 2024 *Burrows-Wheeler transformation*. Lecture for new computer science students. Aarhus, Denmark (presentation pdf, pptx).

May 2024 *Deterministic Cache-Oblivious Funnelselect*. Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Feb 2024 *Algoritmer*. U-days, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Feb 2023 *Algoritmer*. U-days, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Jan 2023 *The challenges of implementing Dijkstra's algorithm*. University of Liverpool, Liverpool, UK (presentation pdf, pptx).

Aug 2022 *Something on Teaching*. PhD and Postdoc retreat, Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Aug 2022 *Burrows-Wheeler transformation*. Lecture for new computer science students. Aarhus, Denmark (presentation pdf, pptx).

May 2022 *Priority Queues with Decreasing Keys*. Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Feb 2022 *Algoritmer*. U-days, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Nov 2021 *About the course Introduction to Programming with Scientific Applications*. DSAU (student association for computer science students). Department of Computer Science, Aarhus Universiy, Aarhus, Denmark (presentation pdf, pptx).

Aug 2021 *An Optimal and Practical Cache-Oblivious Algorithm for Computer Multiresolution Rasters*. Lecture for new computer science students. Aarhus, Denmark (presentation pdf, pptx).

Feb 2021 *Algoritmer*. U-days, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Nov 2020 *Soft Sequence Heaps*. Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Jun 2020 *25 Years of Teaching*. Teaching@Nat-Tech. Aarhus, Denmark (presentation pdf, pptx).

Mar 2020 *Autogenerating Algorithms and Data Structures Exams*. Algorithms and Data Structures Retreat. Aarhus University, Sandbjerg, Denmark.

Feb 2020 *Commodore 64 - PETSCII*. Verdens Kedeligste Foredrag (The World's Most Boring Lec-

ture). Selected "the most boring"/Winner lecture among three lectures. Tågekammeret, Aarhus University, Aarhus, Denmark.

Nov 2019 *Exams – behind the scenes.* DSAU (student association for computer science students). Department of Computer Science, Aarhus Universiy, Aarhus, Denmark (presentation pdf, pptx).

Apr 2019 *Introduction to Programming with Scientific Applications.* Course on Computational Thinking for High School Teachers. Center for Computational Thinking and Design, Aarhus University, Odense, Denmark (presentation pdf, pptx).

Feb 2019 *Recursion.* Verdens Kedeligste Foredrag (The World's Most Boring Lecture). Selected "the most boring"/Winner lecture among three lectures. Tågekammeret, Aarhus University, Aarhus, Denmark.

Nov 2018 *Primtalssætningen skalerer billeder.* Eulers Venner. Department of Mathematics, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Nov 2018 *Julehjerter (X-mas hearts).* DSAU (student association for computer science students). Department of Computer Science, Aarhus Universiy, Aarhus, Denmark (presentation pdf, pptx).

May 2018 *A Short Report on the Course: Introduction to Programming with Scientific Applications.* Computational Math / Science. Center for Computational Thinking and Design, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Feb 2018 *An introduction to PowerPoint - A Dogma talk.* Verdens Kedeligste Foredrag (The World's Most Boring Lecture). Selected "the most boring"/Winner lecture among three lectures. Tågekammeret, Aarhus University, Aarhus, Denmark.

Aug 2017 *An Optimal and Practical Cache-Oblivious Algorithm for Computer Multiresolution Rasters.* Lecture for new computer science students. Aarhus, Denmark (presentation pdf, pptx).

Feb 2017 *Pong - The Multiplayer Game.* Verdens Kedeligste Foredrag (The World's Most Boring Lecture). Tågekammeret, Aarhus University, Aarhus, Denmark.

Mar 2016 *External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates.* Dagstuhl Seminar on "Data Structures and Advanced Models of Computation on Big Data". Dagstuhl, Germany (presentation pdf, pptx).

Feb 2016 *Professor inauguration talk: Data Structures and Models of Computation.* Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Nov 2015 *External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates.* Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Odense, Denmark (presentation pdf, pptx).

Nov 2015 *External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates.* MADALGO, Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Aug 2015 *Blackboard Introduction to TAs @ CS.* Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Aug 2015 *Håndtering af øvelseshold og gruppeafleveringer.* Blackboard Brugermøde @ ST. Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Mar 2015 *Prime Time.* Verdens Kedeligste Foredrag (The World's Most Boring Lecture). Tågekammeret, Aarhus University, Aarhus, Denmark (presentation pdf).

Nov 2014 *The Algorithms and Data Strutures Group at Aarhus University.* Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Sep 2014 *Computing Triplet and Quartet Distances Between Trees.* Computer Science Institute, Charles University, Prague, Czech Republic (presentation pdf, pptx).

Sep 2014 *Simplicity in Computational Geometry – Sven Skyum's Algorithm for Computing the Smallest Enclosing Circle.* Sven Skyum - farewell celebration. Department of Computer Science, Aarhus, Denmark (presentation pdf, pptx).

Aug 2014 *Voronoi Diagrammer.* Lecture for new computer science students. Aarhus, Denmark (presentation pdf, pptx).

Apr 2014 *Sorting Integers in the RAM Model.* Annual MADALGO Review Meeting. Aarhus, Denmark (presentation pdf, pptx).

Apr 2014 *Algoritmer.* Master Class in Mathematics, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Apr 2014 *Writing and Defending your Thesis.* PhD retreat, Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Feb 2014 *Range Minimum Queries (Part II).* Dagstuhl Seminar on "Data Structures and Advanced Models of Computation on Big Data". Dagstuhl, Germany (presentation pdf, pptx).

Jan 2014 *Computing Triplet and Quartet Distances Between Trees.* Department of Computer Science, University of Copenhagen, Copenhagen, Denmark (presentation pdf, pptx).

Oct 2013 *The Encoding Complexity of Two Dimensional Range Minimum Data Structures.* MADALGO, Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Aug 2013 *Voronoi Diagrammer.* Lecture for new computer science students. Aarhus, Denmark (presentation pdf, pptx).

May 2013 *Algorithms and Data Structures.* Computer Science Day. Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Apr 2013 *Algoritmer.* Master Class in Mathematics, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Feb – Mar 2013 *Algoritmer.* U-days, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Dec 2012 *Julehjerter.* Open Space Aarhus, Aarhus, Denmark (presentation pdf, pptx).

Nov 2012 *Triplet and Quartet Distances Between Trees of Arbitrary Degree.* ETH Zürich, Zürich, Switzerland (presentation pdf, pptx).

May 2012 *Algorithms and Data Structures – Strict Fibonacci Heaps.* Computer Science Day. Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Feb 2012 *Clossing a Classical Data Structure Problem: Strict Fibonacci Heaps.* Annual MADALGO Review Meeting. Aarhus, Denmark (presentation pdf, pptx).

Nov 2011 *Dynamic Planar Range Maxima Queries.* LIAFA, Université Paris Diderot, Paris, Paris, France (presentation pdf, pptx).

Oct 2011 *Integer Representations towards Efficient Counting in the Bit Probe Model.* University of Ljubljana, Ljubljana, Slovenia (presentation pdf, pptx).

Oct 2011 *Dynamic Planar Range Maxima Queries.* University of Primorska, Koper, Slovenia (presentation pdf, pptx).

Aug 2011 *Sådan virker Google.* Ungdommens Naturvidenskabelige Forening i Aarhus (UNF). Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Jun 2011 *Integer Representations towards Efficient Counting in the Bit Probe Model.* MADALGO, Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Apr 2011 *Sådan virker Google.* Ungdommens Naturvidenskabelige Forening i København (UNF). IT University of Copenhagen, Copenhagen, Denmark (presentation pdf, pptx).

Mar 2011 *Binære Tællere.* Verdens Kedeligste Foredrag (The World's Most Boring Lecture). Tågekammeret, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Nov 2010 *Udfordringer ved håndtering af massive datamængder: Forskingen ved Grundforskningscenteret for Massive Data Algortihmics Data Algorithmics.* Møde i Universitets-Samvirket Aarhus. Aarhus University, Statsbiblioteket, Aarhus (presentation pdf, pptx).

Nov 2010 *Massive Data Algorithmics.* Forskningsdag for Datamatikerlærere. Erhvervsakademiet Lillebælt, Vejle, Denmark (presentation pdf, pptx).

Oct 2010 *External Memory Indexing Structures.* Dansk Selskab for Datalogi. Copenhagen Business School, Frederiksbjerg, Copenhagen, Denmark (presentation pdf, pptx).

Feb – Mar 2010 *Time-Space Trade-Offs for 2D Range Minimum Queries*. Dagstuhl Seminar on "Data Structures". Dagstuhl, Germany (presentation pdf, pptx).

Oct 2009 *Algorithms: Matrices and Graphs.* MasterClass in Mathematics. ScienceTalenter, Mærsk Mc-Kinney Møller Videncenter, Sorø, Denmark (presentation pdf, pptx).

Sep 2009 *Internetsøgemaskiner.* Ungdommens Naturvidenskabelige Forening i Aalborg (UNF). Aalborg University, Ålborg, Denmark (presentation pdf, zip).

Sep 2009 *MADALGO.* MasterClass Teacher Meeting. ScienceCenter, Mærsk Mc-Kinney Møller Videncenter, Sorø, Denmark (presentation pdf, pptx).

Jun 2008 *Algorithms and Data Structures for Faulty Memory.* Computer Science Day. Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, pptx).

Jan 2008 *Massive Data Algorithmics.* Danske Bank, Faglig Dag. Danske Bank, Aarhus, Denmark (presentation pdf, ppt).

May 2007 *Cache-Oblivious and External Memory Algorithms: Theory and Experiments.* Oberwolfach Seminar on "Algorithm Engineering". Oberwolfach, Germany (presentation pdf, ppt).

Feb 2007 *Internetsøgemaskiner.* Ungdommens Naturvidenskabelige Forening i Ålborg (UNF). Aalborg University, Ålborg, Denmark (presentation pdf, zip).

Oct 2006 *Skewed Binary Search Trees.* Department of Computer Science, University of Copenhagen, Copenhagen, Denmark (presentation pdf).

Feb – Mar 2006 *Skewed Binary Search Trees*. Dagstuhl Seminar on "Data Structures". Dagstuhl, Germany (presentation pdf, zip).

Dec 2004 *Internetsøgemaskiner.* Ungdommens Naturvidenskabelige Forening i Aarhus (UNF). Aarhus University, Aarhus, Denmark (presentation pdf, zip).

Nov 2004 *Søgemaskiner.* Udviklerkonference. Danske Bank, Brabrand, Denmark (presentation pdf, zip).

May 2004 *Algorithms and Data Structures for Hierarchical Memory.* Opfølgningsmøde med Danmarks Grundforskningsfond. BRICS, Department of Computer Science, Aarhus University (presentation pdf, zip).

Apr 2003 *Cache Oblivious Searching and Sorting.* IT University of Copenhagen, Copenhagen, Denmark (presentation pdf, zip).

Jan 2003 *Søgemaskiner på Internettet (with Rolf Fagerberg).* Datalogforeningen. Department of Computer Science, Aarhus University (presentation pdf, zip).

Oct 2002 *BRICS Research Activities - Algorithms.* BRICS Retreat. Sandbjerg, Denmark (presentation pdf, zip).

Feb – Mar 2002 *Optimal Finger Search Trees in the Pointer Machine*. Dagstuhl Seminar on "Data Structures". Dagstuhl, Germany (presentation pdf, zip).

Oct 2001 *Cache Oblivious Search Trees via Trees of Small Height.* Computer Technology Institute, Patras, Greece (presentation pdf, zip).

Sep 2001 *Cache Oblivious Search Trees via Trees of Small Height.* ALCOM-FT Annual Review Meeting. Rome, Italy (presentation pdf, zip).

Jan 2001 *Udvikling og Implementering af Ombrydningsalgoritmer - Et projekt med CCI Europe.* Opfølgningsmøde med Danmarks Grundforskningsfond. BRICS, Department of Computer Science, Aarhus University (presentation pdf, zip).

Nov 2000 *Optimal Static Range-Reporting in One Dimension.* BRICS, Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf, zip).

Oct 2000 *BRICS Research Activities - Algorithms.* BRICS Retreat. Sandbjerg, Denmark (presentation pdf, zip).

Feb – Mar 2000 *Dynamic Convex Hull*. Dagstuhl Seminar on "Data Structures". Dagstuhl, Germany (presentation pdf, zip).

Jan 1999 *Level-Balanced B-Trees.* Department of Computer Science, Duke University, Durham, North Carolina, USA (presentation pdf, zip).

Aug 1998 *Level-Rebuilt B-Trees.* Theory and Practice of Algorithms for Problems Involving Massive Data Sets. BRICS, Department of Computer Science, Aarhus University (presentation pdf, zip).

Mar 1998 *Worst-Case Efficient External-Memory Priority Queues*. Dagstuhl Seminar on "Data Struc-

tures". Dagstuhl, Germany (presentation pdf).

Nov 1997 *Finger Search Trees with Constant Insertion Time.* Instituto di Elaborazione della Informazione, CNR, Pisa, Pisa, Italy (presentation pdf).

Aug 1997 *Finger Search Trees with Constant Insertion Time.* Oberwolfach Seminar on "Effiziente Algoritmen". Oberwolfach, Germany (presentation pdf).

Feb 1997 *Predecessor Queries in Dynamic Integer Sets.* Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Sep 1996 *Approximate dictionary queries.* BRICS, Department of Computer Science, Aarhus University, Aarhus, Denmark.

Sep 1996 *Predecessor Queries in Dynamic Integer Sets.* BRICS, Department of Computer Science, Aarhus University, Aarhus, Denmark.

Mar 1996 *Priority Queues on Parallel Machines.* BRICS, Department of Computer Science, Aarhus University, Aarhus, Denmark (presentation pdf).

Feb – Mar 1996 *Priority Queues on Parallel Machines.* Dagstuhl Seminar on "Data Structures". Dagstuhl, Germany (presentation pdf).

Sep 1995 *Fast Meldable Priority Queues.* Max-Planck-Institut für Informatik, Saarbrücken, Germany (presentation pdf, zip).

Aug 1995 *Priority Queues with Good Worst Case Performance.* Toronto University, Toronto, Ontario, Canada (presentation pdf, zip).

May 1994 *Finger Search Trees.* BRICS Strategy Workshop. BRICS, Department of Computer Science, Aarhus University, Hjarnø, Denmark (presentation pdf).

Teaching

Fall 2024 Lecturer, *Algorithms and Data Structures (213 students)*. Department of Computer Science, Aarhus University.

Spring 2024 Lecturer, *Introduction to Programming with Scientific Applications (168 students)*. Department of Computer Science, Aarhus University.

Oct 2023 Lecturer, *Lecture at the IT-Camp 2023 (woman in CS initiative)*. Department of Computer Science, Aarhus University.

Fall 2023 Lecturer, *Algorithms and Data Structures (172 students)*. Department of Computer Science, Aarhus University.

Spring 2023 Lecturer, *Introduction to Programming with Scientific Applications (168 students)*. Department of Computer Science, Aarhus University.

Oct 2022 Lecturer, *Lecture on algorithms for high-school students (Studiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2022 Lecturer, *Algorithms and Data Structures (197 students)*. Department of Computer Science, Aarhus University.

Spring 2022 Lecturer, *Introduction to Programming with Scientific Applications (198 students)*. Department of Computer Science, Aarhus University.

Oct 2021 Lecturer, *Lecture on algorithms for high-school students (Studiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2021 Lecturer, *Algorithms and Data Structures (223 students)*. Department of Computer Science, Aarhus University.

Spring 2021 Lecturer, *Introduction to Programming with Scientific Applications (177 students)*. Department of Computer Science, Aarhus University.

Nov 2020 Lecturer, *Python Crash Course (3 lectures)*. Steno Diabetes Center Aarhus, Aarhus University Hospital (presentation pdf, pptx, zip).

Fall 2020 Lecturer, *Algorithms and Data Structures (235 students)*. Department of Computer Science, Aarhus University.

Spring 2020 Lecturer, *Introduction to Programming with Scientific Applications (153 students)*. Depart-

ment of Computer Science, Aarhus University.

Spring 2020 Lecturer, *Pre-talent track - 1 lecture on Backwards Analysis*. Department of Computer Science, Aarhus University.

Fall 2019 Lecturer, *Computational Geometry: Theory and Applications (5 lectures)*. Department of Computer Science, Aarhus University.

Fall 2019 Lecturer, *Algorithms and Data Structures (185 students)*. Department of Computer Science, Aarhus University.

Spring 2019 Lecturer, *Pre-talent track - 1 lecture on Backwards Analysis (6 students)*. Department of Computer Science, Aarhus University.

Spring 2019 Lecturer, *Introduction to Programming with Scientific Applications (151 students)*. Department of Computer Science, Aarhus University.

Fall 2018 Lecturer, *Foundations of Algorithms and Data Structures (200 students)*. Department of Computer Science, Aarhus University.

Spring 2018 Lecturer, *Pre-talent track - 1 lecture on Backwards Analysis (6 students)*. Department of Computer Science, Aarhus University.

Spring 2018 Lecturer, *Introduction to Programming with Scientific Applications (93 students)*. Department of Computer Science, Aarhus University.

Oct 2017 Lecturer, *Lecture on algorithms for high-school students (Studiepraktik)*. Department of Computer Science, Aarhus University.

Oct 2017 Lecturer, *Lecture and exercise class on algorithms at the IT-Camp 2017 (woman in CS initiative)*. Department of Computer Science, Aarhus University.

Spring 2017 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 120 students)*. Department of Computer Science, Aarhus University.

Spring 2017 Lecturer, *Algorithm Engineering (Quarter 3, 29 students)*. Department of Computer Science, Aarhus University.

Spring 2017 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 164 students)*. Department of Computer Science, Aarhus University.

Fall 2016 Lecturer, *Computer Science in Perspective (topic Classic Algoritms, and Internet Algorithms, 2 weeks)*. Department of Computer Science, Aarhus University.

Spring 2016 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 145 students)*. Department of Computer Science, Aarhus University.

Spring 2016 Lecturer, *Algorithm Engineering (Quarter 3, 29 students)*. Department of Computer Science, Aarhus University.

Spring 2016 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 171 students)*. Department of Computer Science, Aarhus University.

Fall 2015 Lecturer, *Computer Science in Perspective (topic Classic Algoritms, and Internet Algorithms, 2 weeks)*. Department of Computer Science, Aarhus University.

Fall 2015 Lecturer, *Advanced Algorithms: Data Structures (Quarters 1+2, 18 students)*. Department of Computer Science, Aarhus University.

Spring 2015 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 122 students)*. Department of Computer Science, Aarhus University.

Spring 2015 Lecturer, *Algorithm Engineering (Quarter 3, 38 students)*. Department of Computer Science, Aarhus University.

Spring 2015 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 170 students)*. Department of Computer Science, Aarhus University.

Oct 2014 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2014 Lecturer, *Computer Science in Perspective (topic Classic Algoritms, and Internet Algorithms, 2 weeks)*. Department of Computer Science, Aarhus University.

Spring 2014 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 136 students)*. Department of Computer Science, Aarhus University.

Spring 2014 Lecturer, *Algorithm Engineering (Quarter 3, 21 students)*. Department of Computer Science, Aarhus University.

Spring 2014 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 190 students)*. Department of Computer Science, Aarhus University.

Oct 2013 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Oct 2013 Lecturer, *Exercise class on algorithms at the IT-Camp 2013 (woman in CS initiative)*. Department of Computer Science, Aarhus University.

Fall 2013 Lecturer, *Advanced Algorithms: Data Structures (Quarters 1+2, 47 students)*. Department of Computer Science, Aarhus University.

Fall 2013 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, and Internet Algorithms, 2 weeks)*. Department of Computer Science, Aarhus University.

Spring 2013 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 123 students)*. Department of Computer Science, Aarhus University.

Spring 2013 Lecturer, *Algorithm Engineering (Quarter 3, 18 students)*. Department of Computer Science, Aarhus University.

Spring 2013 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 186 students)*. Department of Computer Science, Aarhus University.

Oct 2012 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Oct 2012 Lecturer, *Exercise class on algorithms at the IT-Camp 2012 (woman in CS initiative)*. Department of Computer Science, Aarhus University.

May 2012 Lecturer, *Lecture on Google and exercises on algorithms for high-school students (from Silkeborg Gymnasium and Rødkilde Gymnasium, Vejle)*. Department of Computer Science, Aarhus University.

Fall 2012 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, and Internet Algorithms, 2 weeks)*. Department of Computer Science, Aarhus University.

Spring 2012 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 129 students)*. Department of Computer Science, Aarhus University.

Spring 2012 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 171 students)*. Department of Computer Science, Aarhus University.

Oct 2011 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2011 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Fall 2011 Lecturer, *Advanced Algorithms: Data Structures (Quarters 1+2, 29 students)*. Department of Computer Science, Aarhus University.

Spring 2011 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 99 students)*. Department of Computer Science, Aarhus University.

Spring 2011 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 146 students)*. Department of Computer Science, Aarhus University.

Oct 2010 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2010 Lecturer, *Computational Geometry (Quarters 1+2, 22 students)*. Department of Computer Science, Aarhus University.

Fall 2010 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Spring 2010 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 84 students)*. Department of Computer Science, Aarhus University.

Spring 2010 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 134 students)*. Department of Computer Science, Aarhus University.

Oct 2009 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2009 Lecturer, *Advanced Algorithms: Data Structures (Quarters 1+2, 54 students)*. Department of Computer Science, Aarhus University.

Fall 2009 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Spring 2009 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 98 students)*. Department of Computer Science, Aarhus University.

Spring 2009 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 140 students)*. Department of Computer Science, Aarhus University.

Oct 2008 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2008 Lecturer, *Computational Geometry (Quarters 1+2, 24 students)*. Department of Computer Science, Aarhus University.

Fall 2008 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Spring 2008 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 89 students)*. Department of Computer Science, Aarhus University.

Spring 2008 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 97 students)*. Department of Computer Science, Aarhus University.

Nov 2007 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2007 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Fall 2007 Lecturer, *Advanced Algorithms: Data Structures (Quarters 1+2, 39 students)*. Department of Computer Science, Aarhus University.

Spring 2007 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 92 students)*. Department of Computer Science, Aarhus University.

Spring 2007 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 105 students)*. Department of Computer Science, Aarhus University.

Nov 2006 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2006 Lecturer, *Computational Geometry (Quarters 1+2, 27 students)* (with Lars Arge). Department of Computer Science, Aarhus University.

Fall 2006 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Spring 2006 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 91 students)*. Department of Computer Science, Aarhus University.

Spring 2006 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 111 students)*. Department of Computer Science, Aarhus University.

Nov 2005 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Feb 2005 Lecturer, *Exercise class on algorithms for high-school students (Gymnasiepraktik)*. Department of Computer Science, Aarhus University.

Fall 2005 Lecturer, *Advanced Algorithms: Data Structures (Quarter 1+2, 22 students)* (with Lars Arge). Department of Computer Science, Aarhus University.

Fall 2005 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Spring 2005 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 92 students)*. Department of Computer Science, Aarhus University.

Spring 2005 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 101 students)*. Department of

Computer Science, Aarhus University.

Fall 2004 Lecturer, *Computer Science in Perspective (topic Algoritms and Complexity, 1 week)*. Department of Computer Science, Aarhus University.

Fall 2004 Lecturer, *Computational Geometry (Quarter 1+2, 29 students)* (with Lars Arge). Department of Computer Science, Aarhus University.

Spring 2004 Lecturer, *Algorithms and Data Structures 2 (Quarter 4, 124 students)*. Department of Computer Science, Aarhus University.

Spring 2004 Lecturer, *Algorithms and Data Structures 1 (Quarter 3, 136 students)*. Department of Computer Science, Aarhus University.

Fall 2003 Lecturer, *External Memory Algorithms and Data Structures (26 students)* (with Rolf Fagerberg). Department of Computer Science, Aarhus University.

Spring 2003 Lecturer, *Algorithms and Data Structures (176 students)* (with Erik Meineche Schmidt). Department of Computer Science, Aarhus University.

Fall 2002 Lecturer, *Algorithms and Data Structures* (with Rolf Fagerberg). Department of Computer Science, Aarhus University.

Fall 2002 Lecturer, *Algorithms for Web Indexing and Searching* (with Rolf Fagerberg). Department of Computer Science, Aarhus University.

Spring 2002 Lecturer, *Algorithms and Data Structures (139 students)* (with Erik Meineche Schmidt). Department of Computer Science, Aarhus University.

Fall 2001 Lecturer, *External Memory Algorithms and Data Structures* (with Rolf Fagerberg). Department of Computer Science, Aarhus University.

Spring 2001 Three lectures on I/O-algorithms, *Advanced Algorithms (Stephen Alstrup and Theis Rauhe)*. IT University of Copenhagen.

Spring 2001 Lecturer, *Algorithms study group*. BRICS International PhD School, Department of Computer Science, Aarhus University.

Fall 2000 Lecturer, *External Memory Algorithms and Data Structures* (with Rolf Fagerberg). Department of Computer Science, Aarhus University.

Fall 1999 Lecturer, *External Memory Algorithms and Data Structures (17 students)* (with Rolf Fagerberg). Department of Computer Science, Aarhus University.

Fall 1999 Lecturer, *Algorithms (6 students)* (with Rolf Fagerberg). BRICS International PhD School, Department of Computer Science, Aarhus University.

May 1998 Lecturer, *MPII Advanced Mini Course: Functional Data Structures*. Max-Planck-Institut für Informatik.

Fall 1998 Lecturer, *Algorithms (6 students)*. BRICS International PhD School, Department of Computer Science, Aarhus University.

Fall 1996 Lecturer, *Algorithms and Data Structures: A course for students at the Engineering College of Aarhus*. Department of Computer Science, Aarhus University.

Spring 1995 Administrator and Teaching Assistant (Instruktor), *dADS: Algorithms and Data Structures*. Department of Computer Science, Aarhus University.

Spring 1994 Teaching Assistant (Instruktor), *dAlg: Algorithmic*. Department of Computer Science, Aarhus University.

Fall 1993 Teaching Assistant (Instruktor), 2 classes, *dOvs: Compiler Construction*. Department of Computer Science, Aarhus University.

Spring 1993 Teaching Assistant (Instruktor), *dAlg: Algorithmic*. Department of Computer Science, Aarhus University.

Fall 1992 Teaching Assistant (Instruktor), *dProg2: Object Oriented Programming*. Department of Computer Science, Aarhus University.

Advising

PostDoc

Nov 2024 – Oct 2025	PostDoc host for Justin Dallant.
Jul 2005 – Jun 2006	PostDoc host for Irit Katriel.

PhD

Feb 2022 – Jul 2025	PhD advisor for Rolf Svenning.
Aug 2021 – Jul 2025	PhD advisor for Jens Kristian Refsgaard Schou.
Aug 2021 – Jul 2025	PhD advisor for Casper Moldrup Rysgaard.
Jan – Jul 2021	PhD co-advisor for Svend Christian Svendsen (main advisor Lars Arge), *Algorithms for Massive Terrains and Graphs* (presentation).
Aug 2020 – Jun 2021	PhD advisor for Jesper Steensgaard (first year of PhD).
Aug 2014 – Oct 2019	PhD advisor for Konstantinos Mampentzidis, *Comparison and Construction of Phylogenetic Trees and Networks* (presentation).
Apr 2014 – Nov 2017	PhD advisor for Edvin Berglin, *Geometric covers, graph orientations, counter games* (presentation).
Apr 2014 – Sep 2017	PhD co-advisor for Ingo van Duijn (main advisor Peyman Afshani).
Feb 2011 – Sep 2015	PhD advisor for Jesper Sindahl Nielsen, *Implicit Data Structures, Sorting, and Text Indexing* (presentation).
Feb 2009 – Nov 2013	PhD advisor for Casper Kejlberg-Rasmussen (Danske Commodities), *Dynamic Data Structures: The Interplay of Invariants and Algorithm Design* (presentation).
Aug 2008 – Jan 2015	PhD advisor for Jakob Truelsen (SCALGO), *Space Efficient Data Structures and External Terrain Algorithms* (presentation).
May 2008 – Jul 2011	PhD advisor for Pooya Davoodi (Polytechnic Institute of New York University), *Data Structures: Range Queries and Space Efficiency* (presentation).
Feb 2008 – Jan 2013	PhD Part A advisor for Mark Greve (Octoshape), *Online Sorted Range Reporting and Approximating the Mode (Progress report)*.
Aug 2007 – Sep 2011	PhD advisor for Kostantinos Tsakalidis (The Chinese University of Hong Kong), *Dynamic Data Structures: Orthogonal Range Queries and Update Efficiency* (presentation).
Feb 2006 – Feb 2010	PhD advisor for Allan Grønlund Jørgensen (Siemens Wind Power A/S), *Data Structures: Sequence Problems, Range Queries, and Fault Tolerance* (presentation).
Sep 2005 – Nov 2009	PhD advisor for Martin Olsen (Aarhus University, Institute of Business and Technology), *Link building* (presentation).
Aug 2004 – Oct 2007	PhD advisor for Johan Nilsson (Octoshape), *Combinatorial algorithms for graphs and partially ordered sets* (presentation).
Aug 2003 – Sep 2007	PhD advisor for Gabriel Moruz (Johann Wolfgang Goether-Universität Frankfurt), *Hardware Aware Algorithms and Data Structures* (presentation).
Feb 2001 – Feb 2002	PhD advisor for Riko Jacob (Technische Universität München), *Dynamic Planar Convex Hull* (presentation).

MSc

Feb – Jun 2024	MSc advisor for Jonas Skøtt Dam and Andreas Østergaard Jakobsen, *Optimizing Contour Lines on Bathymetric Charts*.
Feb – Jun 2023	MSc advisor for Andreas Nikolaj Valkær, *Splay Top Trees and 2-Edge Connectivity*.
Feb – Jun 2022	MSc advisor (Math) for Johannes Jensen, *Computing Voronoi Diagrams Using*

	Fortune's Algorithm.
Feb – Jun 2020	MSc advisor for Peter Ellerup Frank og Lars Gunnar Stjernholm Lundqvist, *Soft Sequence Heaps — Theory and Experimentation*.
Feb – Jun 2018	MSc advisor for Nick Bakkegaard og Peter Burgaard, *Shortest Path Problem in the Plane with Polygonal Obstacle Violations*.
Sep 2017 – Apr 2018	MSc advisor for Martin Jacobsen, *Cache Oblivious Dynamic Dictionaries with Insert/Query Tradeoffs*.
Feb – Jun 2016	MSc advisor for Peter Gabrielsen og Christoffer Holbæk Hansen (formal advisor; project advisor Kasper Green Larsen), *Three-sided Range Reporting in External Memory*.
Feb – Jun 2016	MSc advisor for Jonas Nicolai Hovmand and Morten Houmøller Nygård (formal advisor; project advisor Kasper Green Larsen), *Estimating Frequencies and Finding Heavy Hitters*.
Feb – Jun 2016	MSc advisor for Thor Bagge and Kent Grigo (formal advisor; project advisor Kasper Green Larsen), *Getting to Know the Captain's Mistress with Reinforcement Learning*.
Feb – Jun 2016	MSc advisor for Troels Thorsen (formal advisor; project advisor Kasper Green Larsen), *Categorical Range Searching*.
Feb – Jun 2016	MSc advisor for Kenn Daniel and Casper Færgemand, *Selection in a Heap*.
Sep 2015 – Jan 2016	MSc advisor for Henrik Knakkegaard Christensen, *Algorithms for Finding Dominators in Directed Graphs*.
Sep 2015 – Jan 2016	MSc advisor for Jakob Peter Landbo and Casper Green (formal advisor; project advisor Kasper Green Larsen), *Range Mode Queries in Arrays*.
Sep – Dec 2015	MSc advisor for Jens Christian Christensen Jensen, *Event Detection in Soccer using Spatio-Temporal Data*.
Feb – Aug 2015	MSc advisor for Lukas Walther (formal advisor; project advisor Peyman Afshani), *Intersection of Convex Objects in the Plane*.
Feb – Jul 2015	MSc advisor for Simon Nordved Madsen and Rasmus Hallenberg-Larsen (formal advisor; project advisor Peyman Afshani), *Computing Set Operations on Simple Polygons Using Binary Space Partition Trees*.
Feb – Jun 2015	MSc advisor for Mathies Boile Christensen and Thomas Sandholt (formal advisor; project advisor Peyman Afshani), *Geometric Measures of Depth*.
Feb – Jun 2015	MSc advisor for Mads Ravn (formal advisor; project advisor Kasper Green Larsen), *Orthogonal Range Searching in 2D with Ball Inheritance*.
Feb – Jun 2015	MSc advisor for Anders Strand-Holm Vinther and Magnus Strand-Holm Vinther (formal advisor; project advisor Peyman Afshani), *Pathfinding in Two-dimensional Worlds*.
Dec 2014 – Jun 2015	MSc advisor for Jan Hessellund Knudsen and Roland Larsen Pedersen, *Engineering Rank and Select Queries on Wavelet Trees*.
Nov 2014 – Apr 2015	MSc advisor for Kris Vestergaard Ebbesen, *On the Practicality of Data-Oblivious Sorting*.
Aug 2014 – Feb 2015	MSc advisor for Bo Mortensen (project advisor Peyman Afshani), *Algorithms for Computing Convex Hulls Using Linear Programming*.
May 2014 – Mar 2015	MSc advisor for Claus Jespersen, *Monte Carlo Evaluation of Financial Options Using a GPU*.
May 2013 – Jun 2014	MSc advisor for Daniel Winther Petersen (Nykredit), *Orthogonal Range Skyline Counting Queries*.
May 2013 – Jan 2014	MSc advisor for Jakob Mark Friis (Lind Capital) and Steffen Beier Olesen (Lind Capital), *An Experimental Comparison of Max Flow Algorithms*.
Apr 2013 – Apr 2014	MSc advisor for Jana Kunert, *Hashing and Random Graphs*.
Nov 2012 – Aug 2013	MSc advisor for Jørgen Fogh, *Engineering a Fast Fourier Transform*.

Oct 2012 – Jun 2013	MSc advisor for Morten Holt and Jens Johansen (thesis awarded the best Danish MSc thesis in Computer Science in 2013, by the Danish Society for Computer Science), *Computing Triplet and Quartet Distances*.
Aug 2012 – Oct 2013	MSc advisor for Jeppe Schou, *Range Minimum Data Structures*.
Apr 2012 – Jan 2015	MSc advisor for Mikkel Engelbrecht Hougaard, *On the Complexity of Red-Black Trees for Higher Dimensions*.
Sep 2011 – Mar 2012	MSc advisor for Andreas Koefoed-Hansen (Aarhus University), *Representations for Path Finding in Planar Environments*.
Feb – Sep 2010	MSc advisor for David Kjær (Milestone Systems), *Range Median Algorithms*.
Sep 2009 – Sep 2010	MSc advisor (joint with Mohammad Ali Abam) for Jonas Suhr Christensen, *Experimental Study of Kinetic Geometric t-Spanner Algorithms*.
Apr 2008 – Apr 2009	MSc advisor for Henrik Bitsch Kirk (Statsbiblioteket), *Searching with Dynamic Optimality: In Theory and Practice*.
Feb – Dec 2008	MSc advisor for Claus Andersen (Translucent), *An optimal minimum spanning tree algorithm*.
Jan 2008 – Feb 2009	MSc advisor for Krzysztof Piatkowski (Peopleway), *Implementering og udvikling af maksimum delsum algoritmer*.
Sep 2007 – Mar 2008	MSc advisor for Jonas Maturana Larsen (Trifork) and Michael Nielsen (Plushost), *En undersøgelse af algoritmer til løsning af generalized movers problem i 3D*.
Sep 2006 – Aug 2007	MSc advisor for Thomas Rasmussen, *Evaluering af en skæringsalgoritme for Bezier kurver i planen*.
Sep 2006 – Mar 2007	MSc advisor for Bjørn Casper Torndahl and Bo Søndergaard Carstensen, *Cache Oblivious String Dictionaries*.
Mar 2006 – Aug 2007	MSc advisor for Lasse Østerlund Gram (Marcantec), *Robusthed af netværk - med focus på scale-free grafer*.
Jul 2005 – Jan 2007	MSc advisor for Kristian Dorph-Petersen (Danske Bank), *Praktisk brug af dynamisk sampling i data streams*.
Feb 2005 – Jun 2006	MSc advisor for Dennis Søgaard (Accenture), *Minimising the Number of Collision Tests in Probabilistic Road Maps using Approximations in a Binary Connection Strategy*.
Jan 2005 – Jan 2006	MSc advisor for Jesper Buch Hansen (Danske Bank), *Computing the Visibility Graph of Points Within a Polygon*.
Feb 2004 – May 2006	MSc advisor for Morten Laustsen (Mjølner Informatics), *Orthogonal Range Counting in The Cache Oblivious Model*.
Aug 2002 – Jan 2004	MSc advisor for Louise Skouboe Bjerg (Systematic Software Engineering A/S) and Lone Asferg Laursen (Mjølner Informatics), *Approksimative afstande i planare grafer*.
Aug 2002 – Jun 2003	MSc advisor (joint with Rolf Fagerberg) for Kristoffer Vinther (Bang & Olufsen), *Engineering Cache-Oblivious Sorting Algorithms*.
Feb 2000 – Jan 2001	MSc advisor for Kristian Høgsberg Kristensen (Intel), *Automated Layout of Classified Ads*.

Project

Sep 2018 – Jun 2025	Project advisor for 96 computer science bachelor students on their bachelor project (15 ECTS) and 4 data science bachelor students on their bachelor project (10 ECTS), 1 mathematics bachelor student on the bachelor project (10 ECTS).
Sep 2018 – Jun 2024	Project advisor for 13 bachelor students on the talent track (5 ECTS).
Sep – Dec 2017	Project advisor for Nick Bakkegaard (5 ECTS), *Survey on Range Minimum Query*.
Feb – Sep 2000	Project advisor for Jakob Skyberg (5 ECTS), *Implementation of three Convex Hull algorithms in Java*.

Services

Department of Computer Science, Aarhus University

Aug 2016	Organizing chair of ALGO 2016 (319 participants) that covered the joinly conferences and workshops: ESA, WABI, IPEC, WAOA, ALGOCLOUD, ALGOSENSORS, ATMOS, MASSIVE.
Feb 2015 – Sep 2018	Member of the study board of the Aarhus School of Science (ASOS), Aarhus University.
Sep 2014 – Jan 2019	Member of the departments research committee.
Sep 2014 – Jan 2019	Chair of the departments education committee.
Oct 2013 –	Member of the departments PostDoc committee.
May 2013 – Feb 2016	Member of the departments PhD committee.
Nov – Dec 2009	Chair of assessment committee for associate professor position.
Dec 2008 –	Member of the departments office-committee.
Jun 2006	Organizing chair of the Summer School on Game Theory in Computer Science.
Jul 2003	Member of the organizing committee of the 18th IEEE Conference on Computational Complexity.
Jun – Jul 2002	Organizing chair of the EEF Summer School on Massive Data Sets (55 participants), BRICS, Aarhus University, Denmark.
Feb 2002 – Nov 2009	Member of the departments web-committee.
Aug 2001	Organizing chair of ALGO 2001 (178 participants) that covered the joinly conferences and workshops: 9th Annual European Symposium on Algorithms, 5th Workshop on Algorithm Engineering, 1st Workshop on Algorithms in BioInformatics, and 2nd International Workshop on Approximation and Randomized Algorithms in Communication Networks.
May 1999 – May 2000	Member of the departments web-committee.
Jan 1999 – Dec 2005	Co-organizer of the BRICS mini-courses.
Aug 1998 –	Co-organizer of the algorithms and complexity theory seminars.

To the profession

May 2023	Co-organizer Dagstuhl Seminar on "Scalable Data Structures". Dagstuhl, Germany.
Apr 2022 – Mar 2026	Member of the Danish Censor list for Mathematics.
Apr 2022 – Mar 2026	Member of the Danish Censor list for Bachelor of Science (BSc), Master of Science (MSc) in Engineering programmes and master programmes (continuing education) within the field of Electronics, IT and Energy.
Apr 2022 – Mar 2026	Member of the Danish Censor list for Bachelor of Engineering (BEng) programmes and Technical Diploma programmes (field Diplom - Electronics, IT and Energy).
Feb 2021	Co-organizer Dagstuhl Seminar on "Scalable Data Structures". Dagstuhl, Germany.
Jan – Feb 2019	Co-organizer Dagstuhl Seminar on "Data Structures for the Cloud and External Memory Data". Dagstuhl, Germany.
May 2014 – Dec 2018	Co-Editor-in-Chief Journal of Discrete Algorithms.
Apr 2014 – Mar 2022	Member of the Danish Censor list for Engineering (Mathematics, Physics, and Social Sciences).
Apr 2010 – Dec 2016	Member of the Computer Science Group (Group 38) of the Danish Bibliometric Research Indicator, chair January 2014-December 2016.
Mar – Oct 2010	Member of the Scientific Panel for e-Science, the Danish National Roadmap for Research Infrastructures.
Jan 2010 – Feb 2013	Member of the Scientific Advising Group for the ESS Data Management Centre.

Jan 2007 – Jan 2016	Member of the Steering Committee, Meeting on Algorithm Engineering and Experiments (ALENEX).
Apr 2006 – Mar 2026	Member of the Danish Censor list for Computer Science.
Mar 2006	Responsible for the electronic submission server and the eletronic server for the program committee of the 10th Scandinavian Workshop on Algorithm Theory, Riga, Latvia.
Sep 2004 – Sep 2007	Member of the Steering Committee, European Symposium on Algorithms (ESA).
Sep 2000 – Jan 2004	Responsible for the electronic server handling ALCOM-FT technical reports.
Jul 1999	Co-responsible for the electronic submission server and the eletronic server for the program committee of the 3rd Workshop on Algorithm Engineering, London, UK.
Aug 1997	Editor of the Oberwolfach report "Tagungsbericht 29/1997 – Effiziente Algorithmen".

Max-Planck-Institut für Informatik

Aug 1998	Member of the organizing committee of the 2nd Workshop on Algorithm Engineering.
Jan – Jul 1998	Member of the travel committee.

Examiner

Jan 2025	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Methods for Computer Science (DM549), Discrete Methods for Data Science (DS820, DSK820), Introduction to Mathemathical Methods (MM537), Discrete Mathematics (DM547), Kevin Schewior (156 students).
Dec 2024	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2024 Fall, 10 ECTS) Peyman Afshani (13 students).
Dec 2024	Course examiner, Department of Computer Science, Aarhus University. Machine Learning (2024 Fall, 10 ECTS), Kasper Green Larsen (3 students).
Dec 2024	Course examiner, Department of Computer Science, Aarhus University. Optimization (2024 Spring, 10 ECTS), Kristoffer Arnsfelt Hansen (4 students, reexam).
Aug 2024	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507/DM578/DS814, 8 ECTS) and Discrete Mathematics, Algorithms and Data Structures (IMADA SE4-DMAD, 10 ECTS), Rolf Fagerberg and Lene Monrad Favrholdt (28 students, reexam).
Aug 2024	Course co-examiner, IT University of Copenhagen. Algorithms and Data Structures (BSALDAS1KU/KSALDAS1KU/1408001U, 7.5 ECTS), Thore Husfeldt og Riko Jacob (49 students).
Aug 2024	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Mathematics (DM549/DS820), Kevin Schewior (1 student, reexam).
Jun 2024	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507/DM578/DS814, 8 ECTS) and Discrete Mathematics, Algorithms and Data Structures (IMADA SE4-DMAD, 10 ECTS), Rolf Fagerberg and Lene Monrad Favrholdt (180 students).
Jun 2024	Course examiner, Department of Computer Science, University of Copenhagen. BSc projects, advisor Jacob Holm (4 students).
Jun 2024	Course examiner, Department of Computer Science, Aarhus University. Project work in Computer Science (2024 Spring, 10 ECTS) Kasper Green Larsen (1 student).

Jun 2024	Course examiner, Department of Computer Science, Aarhus University. Optimization (2024 Spring, 10 ECTS) Kristoffer Arnsfelt Hansen (30 students).
Jun 2024	Course examiner, IT University of Copenhagen. First-year Project: Map of Denmark. Visualization, Navigation, Searching, and Route Planning (15 ECTS), Nutan Limaye (32 students).
Jun 2024	Bachelor project examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense (Spring 2024). Rolf Fagerberg and Lene Monrad Favrholdt (4 students).
May 2024	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2023 Fall, 10 ECTS) Peyman Afshani (2 students, reexam).
May 2024	Course co-examiner, IT University of Copenhagen. Algorithms and Data Structures (BSALDAS1KU/KSALDAS1KU/1408001U, 7.5 ECTS), Thore Husfeldt og Riko Jacob (335 students).
Apr 2024	Course examiner, Department of Computer Science, Aarhus University. Scalable Microservices (2024 Spring, 5 ECTS). Henrik Bærbak Christensen (7 students).
Mar 2024	Course examiner, IT University of Copenhagen. Applied Algorithms (2023 Fall, 7.5 ECTS) Riko Jacob (3 students, reexam).
Mar 2024	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Methods for Computer Science (DM549), Discrete Methods for Data Science (DS820), Introduction to Mathemathical Methods (MM537), Kevin Schewior (26 students, reexam).
Jan 2024	Project examiner, Department of Computer Science, Aarhus University. Project work in Computer Science (2023 Fall, 10 ECTS) Jaco van de Pol and Steffan Christ Sølvsten Jørgensen (1 student).
Jan 2024	Course examiner, IT University of Copenhagen. Applied Algorithms (2023 Fall, 7.5 ECTS) Riko Jacob (15 students).
Jan 2024	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Methods for Computer Science (DM549), Discrete Methods for Data Science (DS820), Introduction to Mathemathical Methods (MM537), Discrete Mathematics (DM547), Kevin Schewior (102 students).
Jan 2024	Course examiner, Department of Computer Science, Aarhus University. Software Engineering and Software Architecture (2023 Fall, 10 ECTS) Henrik Bærbak Christensen (16 students).
Jan 2024	Bachelor project examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense (Fall 2023). Rolf Fagerberg and Lene Monrad Favrholdt (2 students).
Dec 2023	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2023 Fall, 10 ECTS) Peyman Afshani (12 students).
Dec 2023	Course examiner, Department of Computer Science, Aarhus University. Algorithms and Data Structures, Master Degree Programme in Informatics Teaching (2022 Fall, 5 ECTS) Mathias Rav (11 students).
Aug 2023	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Methods for Computer Science (DM549), Introduction to Mathemathical Methods (MM537), Lene Monrad Favrholdt (4 students, reexam).
Aug 2023	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507/DM578/DS814, 8 ECTS) and Discrete Mathematics, Algorithms and Data Structures (IMADA SE4-DMAD, 10 ECTS), Rolf Fagerberg and Lene Monrad Favrholdt (13 students, reexam).
Jun 2023	Course examiner, Department of Mathematics and Computer Science, Univer-

	sity of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507/DM578/DS814, 8 ECTS) and Discrete Mathematics, Algorithms and Data Structures (IMADA SE4-DMAD, 10 ECTS), Rolf Fagerberg and Lene Monrad Favrholdt (186 students).
Jun 2023	Course examiner, Department of Computer Science, University of Copenhagen. BSc projects, advisor Jacob Holm (6 students).
Jun 2023	Examiner, Department of Computer Science, Aarhus University. BSc talent track project in Algorithms (2023 Spring, 5 ECTS), Kasper Green Larsen (1 student).
Jun 2023	MSc thesis examiner, Laura Kristensen, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Joan Boyar.
Jun 2023	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2022 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (1 student, reexam).
Jun 2023	Bachelor project examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense (Summer 2023). Rolf Fagerberg (1 student).
Mar 2023	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2022 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (1 student, reexam).
Feb 2023	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Methods for Computer Science (DM549), Discrete Methods for Data Science (DS820), Introduction to Mathemathical Methods (MM537), Discrete Mathematics (DM547), Lene Monrad Favrholdt (9 students, reexam).
Feb 2023	Course examiner, Department of Computer Science, Aarhus University. Algorithms and Data Structures, Master Degree Programme in Informatics Teaching (2022 Fall, 5 ECTS) Mathias Rav (3 students).
Jan 2023	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Methods for Computer Science (DM549), Discrete Methods for Data Science (DS820), Introduction to Mathemathical Methods (MM537), Discrete Mathematics (DM547), Lene Monrad Favrholdt (87 students).
Jan 2023	Course examiner, Department of Computer Science, Aarhus University. Introduction to Programming (2022 Fall, 10 ECTS) Kurt Jensen (31 students).
Jan 2023	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. BSc projects, advisors Rolf Fagerberg, Joan Boyar, Lene Monrad Favrholdt (3 students).
Dec 2022	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2022 Fall, 10 ECTS) Peyman Afshani (13 students).
Dec 2022	Course examiner, Department of Computer Science, Aarhus University. Algorithms and Data Structures, Master Degree Programme in Informatics Teaching (2022 Fall, 5 ECTS) Mathias Rav (12 students).
Aug 2022	Course examiner, IT University of Copenhagen. Algorithms and Data Structures (BSALDAS2KU/KSALDAS2KU, 7.5 ECTS), Riko Jacob (5 students).
Aug 2022	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507/DS814, 8 ECTS) and Discrete Mathematics and Algorithms and Data Structures (IMADA SE4-DMAD, 10 ECTS), Rolf Fagerberg and Lene Monrad Favrholdt (21 students, reexam).
Aug 2022	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Complexity and Computability (IMADA DM553/MM850, 10 ECTS), Joan Boyar (5 students, reexam).

Aug 2022	Course examiner, IT University of Copenhagen. Algorithms and Data Structures (BSALDAS1KU/KSALDAS1KU/1408001U, 7.5 ECTS), Riko Jacob (29 students).
Jul 2022	Course examiner, Department of Computer Science, University of Copenhagen. Algorithms and Data Structures (7.5 ECTS), Rasmus Pagh (15 students).
Jun 2022	Course examiner, Department of Computer Science, University of Copenhagen. BSc projects, advisors Danupon Na Nongkai and Jacob Holm (6 students).
Jun 2022	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Complexity and Computability (IMADA DM553/MM850, 10 ECTS), Joan Boyar (38 students).
Jun 2022	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507/DS814, 8 ECTS) and Discrete Mathematics and Algorithms and Data Structures (IMADA SE4-DMAD, 10 ECTS), Rolf Fagerberg and Lene Monrad Favrholdt (170 students).
Jun 2022	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. BSc projects and MSc theses, advisors Rolf Fagerberg, Joan Boyar, Lene Monrad Favrholdt and Kim Skak Larsen (7 students).
May 2022	Course examiner, IT University of Copenhagen. Algorithms and Data Structures (BSALDAS1KU/KSALDAS1KU/1408001U, 7.5 ECTS), Thore Husfeldt og Riko Jacob (150 students).
Apr 2022	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2021 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (1 student, reexam).
Jan 2022	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2021 Fall, 10 ECTS) Peyman Afshani (16 students).
Dec 2021	Course examiner, Department of Computer Science, Aarhus University. Algorithms and Data Structures, Master Degree Programme in Informatics Teaching (2021 Fall, 5 ECTS) Troels Bjerre Lund (10 students).
Aug 2021 – Aug 2020	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (7 students, reexam) and Discrete Algorithms, Algorithms and Data Structures (IMADA SE4-DMAD, 10 ECTS), Rolf Fagerberg and Lene Monrad Favrholdt (11 students, reexam).
Jun 2021	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2020 Fall, 10 ECTS) Peyman Afshani (1 student, reexam).
Jun 2021	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. BSc projects, advisors Rolf Fagerberg, Joan Boyar, Lene Monrad Favrholdt (5 students).
Jun 2021	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Heuristics and Approximation Algorithms (IMADA DM865, 10 ECTS), Lene Monrad Favrholdt and Marco Chiarandini (2 students).
Apr – May 2021	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2020 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (2 students, reexam).
Mar 2021	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Probability (IMADA DM551, 10 ECTS), Joan Boyar (10 students).
Jan 2021	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Probability (IMADA DM551/MM851, 10 ECTS), Joan Boyar (48 students).

Jan 2021	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. BSc project, advisors Rolf Fagerberg (1 student).
Jan 2021	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2020 Fall, 10 ECTS) Peyman Afshani (15 students).
Jan 2021	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2020 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (12 students).
Dec – 2020	Course examiner, Department of Computer Science, Aarhus University. Algorithms and Data Structures, Master Degree Programme in Informatics Teaching (2020 Fall, 5 ECTS) Erik Meineche Schmidt (23 students).
Aug 2020	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (7 students).
Aug 2020	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Complexity and Computability (IMADA DM553, 10 ECTS), Joan Boyar (7 students).
Jun 2020	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Complexity and Computability (IMADA DM553, 10 ECTS), Joan Boyar (13 students).
Jun 2020	Course examiner, Department of Computer Science, Aarhus University. Optimization (2020 Spring, 10 ECTS) Kristoffer Arnsfelt Hansen (23 students).
Jun 2020	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. BSc projects, advisors Rolf Fagerberg, Joan Boyar, Lene Monrad Favrholdt (8 students).
Jun 2020	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Heuristics and Approximation Algorithms (IMADA DM865, 10 ECTS), Lene Monrad Favrholdt and Marco Chiarandini (12 students).
Jan 2020	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2019 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (18 students).
Jan 2020	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2019 Fall, 10 ECTS) Peyman Afshani (9 students).
Dec 2019	Course examiner, Department of Computer Science, Aarhus University. Introduction to Programming (2019 Fall, 10 ECTS) Kurt Jensen (15 students).
Aug 2019	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (18 students).
Aug 2019	Course examiner, Department of Computer Science, Aarhus University. Optimization (2019 Spring, 10 ECTS) Kristoffer Arnsfelt Hansen (7 students).
Jun 2019	MSc thesis examiner, Nicolai Aarestrup Jørgensen, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg.
Jun 2019	Course examiner, Department of Computer Science, Aarhus University. Optimization (2019 Spring, 10 ECTS) Kristoffer Arnsfelt Hansen (19 student).
Jun 2019	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (154 students).
Jun 2019	Bachelor project examiner, Department of Computer Science, Aalborg University. Ingo van Duijn (4 students).
Jan 2019	Course examiner, Department of Computer Science, Aarhus University. Soft-

	ware Engineering and Software Architecture (2018 Fall, 10 ECTS) Henrik Bærbak Christensen (54 students).
Dec 2018 – Jan 2019	Course examiner, Department of Computer Science, Aarhus University. Introduction to Programming (2018 Fall, 10 ECTS) Kurt Jensen (53 students).
Aug 2018	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Heuristics and Approximation Algorithms (IMADA DM865, 10 ECTS), Lene Monrad Favrholt and Marco Chiarandini (1 student), Computer Game Programming (IMADA DM842, 10 ECTS), Rolf Fagerberg and Christian Kudahl (1 student), Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (19 students).
Jun 2018	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (192 students).
May 2018	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2017 Fall, 10 ECTS) Peyman Afshani (4 students).
May 2018	Course examiner, Department of Computer Science, Aarhus University. Introduction to Programming (2017 Fall, 10 ECTS) Kurt Jensen (5 students).
May 2018	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2017 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (1 student).
Jan 2018	Course examiner, Department of Computer Science, Aarhus University. Theory of Algorithms and Computational Complexity (2017 Fall, 10 ECTS) Kristoffer Arnsfelt Hansen (14 students).
Jan 2018	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry: Theory and Experimentation (2017 Fall, 10 ECTS) Peyman Afshani (25 students).
Jan 2018	Course examiner, Department of Computer Science, Aarhus University. Introduction to Programming (2017 Fall, 10 ECTS) Kurt Jensen (36 students).
Jan 2018	Course examiner, Department of Computer Science, Aarhus University. Software Engineering and Software Architecture (2017 Fall, 10 ECTS) Henrik Bærbak Christensen (39 students).
Aug 2017	Course examiner, Department of Computer Science, Aarhus University. Algorithmic Gems (2017 Q4, 5 ECTS) Kasper Green Larsen (2 students).
Jun 2017	MSc thesis examiner, Kristine Vitting Klinkby Knudsen, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Jørgen Bang-Jensen.
Jun 2017	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. DM553 Complexity and Computability (10 ECTS, 23 student) and DM508 Algorithms and Complexity (5 ECTS, 1 student), Jørgen Bang-Jensen.
Jun 2017	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Approximation Algorithms (IMADA DM833, 5 ECTS), Lene Monrad Favrholdt (1 student).
Jun 2017	Course examiner, Department of Computer Science, Aarhus University. Distribuerede systemer (2017 Q3+Q4, 5 ECTS) Claudio Orlandi (17 students).
Jun 2017	Bachelor project examiner, Department of Computer Science, Aalborg University. Stefan Schmid and Klaus-Tycho Förster (7 students).
Jun 2017	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (130 students).
Mar 2017	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense, DM543/DM557/VK-DKM-

	E1v16/DM534/DM558 (9 students, reexam).
Jan 2017	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2016, 10 ECTS), Lars Arge (12 students).
Jan 2017	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2016, 5 ECTS, 40 students), Gudmund Skovbjerg Frandsen.
Oct 2016	Course examiner, Department of Computer Science, Aarhus University. Computability and Logic (14 students).
Aug 2016	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense, BSc project/DM507/DM553/DM860/DM546 (14 students, reexam).
Aug 2016	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense, DM508/DM553/DM817/DM556 (9 students, reexam).
Jun 2016	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. DM553 Complexity and Computability (10 ECTS, 21 students).
Jun 2016	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (118 students).
Feb 2016	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense, DM557/I-ITS5-El/VK-DKM-E1, DM819, DM551, DM207, DM847 (14 students, reexam).
Jan 2016	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. I/O-Efficient Algoritms and Data Structures (IMADA DM207, 10 ECTS), Rolf Fagerberg (10 students).
Jan 2016	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry (Q1+Q2 2015, 10 ECTS), Peyman Afshani (6 students).
Jan 2016	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2015, 5 ECTS), Gudmund Skovbjerg Frandsen (24 students).
Jan 2016	Course examiner, Department of Computer Science, Aarhus University. Visualisering og projektkommunikation (2016 Q2, 5 ECTS) Majken Kirkegård Rasmussen (13 students).
Nov 2015	MSc thesis examiner, Michael Nørskov, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Jørgen Bang-Jensen.
Aug 2015	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (8 students).
Aug 2015	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Complexity (IMADA DM508, 5 ECTS), Joan Boyar and Bjarne Toft (1 student).
Jun 2015	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. DM538 Algorithms and Probability (5 ECTS, 1 student) Lene Monrad Favrholdt, DM553 Complexity and Computability (10 ECTS, 1 student) and DM508 Algorithms and Complexity (5 ECTS, 6 students) Joan Boyar.
Jun 2015	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (106 students).
Mar 2015	Course examiner, Department of Computer Science, Aarhus University. Fysisk Design (2015 Q3, 5 ECTS), Peter Krogh (22 students).
Mar 2015	Course examiner, Department of Computer Science, Aarhus University. Visualisering og projektkommunikation (ITvap, 2014 Q2, 5 ECTS), Peter Krogh (4

	students, reexam).
Jan 2015	Course examiner, Department of Computer Science, Aarhus University. Visualisering og projektkommunikation (ITvap, 2014 Q2, 5 ECTS), Aviaja Borup (13 students).
Jan 2015	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2014, 5 ECTS), Gudmund Skovbjerg Frandsen (32 students).
Oct 2014	Course examiner, Department of Computer Science, Aarhus University. Pervasive Computing (OS, Q1 2014, 5 ECTS), Niels Olof Bouvin (14 students).
Sep 2014	Bachelor project examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense (Summer 2014). Rolf Fagerberg (1 student).
Jun 2014	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (73 students).
Jun 2014	Course examiner, Department of Computer Science, Aarhus University. Advanced Realtime Graphics Effects (Q4 2014, 5 ECTS), Toshiya Hachisuka (15 students).
Jun 2014	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Complexity (IMADA DM508, 5 ECTS), Joan Boyar (33 students).
Jun 2014	Course examiner, Department of Computer Science, Aarhus University. Discrete Computaional Geometry (Q3+Q4 2014, 10 ECTS), Peyman Afshani (5 students).
Jun 2014	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Approximation Algorithms (IMADA DM833, 5 ECTS), Lene Monrad Favrholdt (10 students).
May 2014	Course examiner, Department of Computer Science, Aarhus University. Dynamic Algorithms (Q4 2013, 5 ECTS), Gudmund Skovbjerg Frandsen (1 student).
May 2014	Study group examiner, Department of Computer Science, Aarhus University. Advisor Toshiya Hachisuka (1 student).
Jan 2014	MSc thesis examiner, Thomas Nørbo Jensen, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg.
Jan 2014	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. String Algorithms (IMADA DM823, 5 ECTS), Rolf Fagerberg (4 students).
Jan 2014	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2013, 10 ECTS), Lars Arge (12 students).
Jan 2014	MSc thesis examiner, Søren Erling Lynnerup, Department of Computer Science, University of Copenhagen. Advisor Pawel Winter.
Jan 2014	BSc project examiner, Niklas Thiemann and Claus Vium, Department of Computer Science, University of Copenhagen. Advisor Pawel Winter.
Jan 2014	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2013, 5 ECTS), Gudmund Skovbjerg Frandsen (55 students).
Dec 2013	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Computer Game Programming IV: Projects (IMADA DM816, 5 ECTS), Rolf Fagerberg (2 students).
Sep 2013	MSc thesis examiner, Thomas Palludan Hargreaves, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg.
Jun 2013	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Rolf Fagerberg (47 students).
Jun 2013	Course examiner, Department of Mathematics and Computer Science, University

	of Southern Denmark, Odense. Algorithms and Complexity (IMADA DM508, 5 ECTS), Joan Boyar (28 students).
Jun 2013	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Computer Game Programming III: Physics (IMADA DM815, 5 ECTS), Rolf Fagerberg (6 students).
Jun 2013	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Approximation Algorithms (IMADA DM833, 5 ECTS), Lene Monrad Favrholdt (22 students).
Jun 2013	Course examiner, Department of Computer Science, Aarhus University. Dynamic Algorithms (Q4 2013, 5 ECTS), Gudmund Skovbjerg Frandsen (1 student).
Jan 2013	Course examiner, Department of Computer Science, Aarhus University. Computational Geometry (Q1+Q2 2012, 10 ECTS), Peyman Afshani (20 students).
Jan 2013	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2012, 5 ECTS), Gudmund Skovbjerg Frandsen (59 students).
Jan 2013	Course examiner, Department of Computer Science, Aarhus University. Operativ Systemer (dOpSys, Q2 2012, 5 ECTS), Erik Ernst (6 students).
Dec 2012	MSc thesis examiner, Stoyan Ivanov Kamburoy, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg.
Aug 2012	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2011, 5 ECTS), Gudmund Skovbjerg Frandsen (13 students).
Jun 2012	Bachelor projects examiner, DTU Informatik, Technical University of Denmark (Spring 2012). Phillip Bille and Inge Li Gørtz (5 students).
Jun 2012	Project examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Rolf Fagerberg (1 student).
Jun 2012	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2012, 10 ECTS), Lars Arge (9 students).
Mar 2012	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Complexity (IMADA DM508, 5 ECTS), Joan Boyar (27 students).
Jan 2012	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. I/O-Efficient Algoritms and Data Structures (IMADA DM207, 10 ECTS), Rolf Fagerberg (4 students).
Jan 2012	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2011, 5 ECTS), Gudmund Skovbjerg Frandsen (42 students).
Sep 2011	MSc thesis examiner, Jens Henrik Hertz and Martin Ancher Müller Neiiendam, DTU Informatik, Technical University of Denmark. Advisor Philip Bille and Inge Li Gørtz.
Aug 2011	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algoritmer og Datastrukturer (IMADA DM507, 37 ECTS), Lene Monrad Favrholdt (4 students).
Aug 2011	MSc thesis examiner, Hjalte Wedel Vildhøj and Søren Vind, DTU Informatik, Technical University of Denmark. Advisor Philip Bille and Inge Li Gørtz.
Jun 2011	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2011, 10 ECTS), Lars Arge (21 students).
Jun 2011	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algoritmer og Datastrukturer (IMADA DM507, 37 ECTS), Lene Monrad Favrholdt (11 students).
Apr 2011	MSc thesis examiner, Jakob Lund, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg and Kim Skak Larsen.
Mar 2011	Course examiner, Department of Computer Science, Aarhus University.

	Functional-Programming Techniques (dTFP, Q3 2011, 5 ECTS), Olivier Danvy (10 students).
Mar 2011	Course examiner, Department of Computer Science, Aarhus University. Introduction to Functional Programming (dIFP Q1 2010, 5 ECTS), Olivier Danvy (1 student, reexam).
Jan 2011	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Computer Game Programming III: Physics (IMADA DM815, 5 ECTS), Rolf Fagerberg (11 students).
Jan 2011	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2010, 5 ECTS), Gudmund Skovbjerg Frandsen (35 students).
Jan 2011	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Mathematics (IMADA MM524-DM527, 5 ECTS), Daniel Merkle (5 students).
Oct 2010	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Discrete Mathematics (IMADA MM524-DM527, 5 ECTS), Daniel Merkle (70 students).
Oct 2010	Course examiner, Department of Computer Science, Aarhus University. Introduction to Functional Programming (dIFP Q1 2010, 5 ECTS), Olivier Danvy (20 students).
May 2010	MSc thesis examiner, Nikolaj Bytsø, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg.
Apr 2010	Course examiner, Department of Computer Science, Aarhus University. Reliable Software Architetures (5 ECTS), Henrik Bærbak Christensen (4 students).
Jan 2010	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. I/O-Efficient Algoritms and Data Structures (IMADA DM207, 10 ECTS), Rolf Fagerberg (3 students).
Jan 2010	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2009, 5 ECTS), Gudmund Skovbjerg Frandsen (28 students).
Oct 2009	Course examiner, Department of Computer Science, Aarhus University. Introduction to Programming (dIntProg, Q1 2009, 5 ECTS), Michael Caspersen.
Aug 2009	MSc thesis examiner, Thomas Nordahl Pedersen, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Lene Monrad Favrholdt.
Aug 2009	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms and Data Structures (IMADA DM507, 10 ECTS), Lene Monrad Favrholdt (5 students).
Jan 2009	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2008, 5 ECTS), Gudmund Skovbjerg Frandsen (78 students).
Oct 2008	Course examiner, Department of Computer Science, Aarhus University. Introduction to Programming (dIntProg, Q1 2008, 5 ECTS), Michael Caspersen.
Jun 2008	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2008, 10 ECTS), Lars Arge (17 students).
Jun 2008	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. I/O-Efficient Algoritms and Data Structures (IMADA DM808, 10 ECTS), Rolf Fagerberg (4 students).
Jun 2008	MSc thesis examiner, Torsten Bonde Christiansen, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg.
Jan 2008	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Algorithms for Web Indexing and Searching (IMADA DM79, 10 ECTS), Rolf Fagerberg (7 students).
Jan 2008	Course examiner, Department of Computer Science, Aarhus University. Programming 2 (dProg 2, Q2 2007, 5 ECTS), Gudmund Skovbjerg Frandsen (37 students).

Jun 2007	Course examiner, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Topics in Algorithmics (IMADA DM69, 10 ECTS), Lene Monrad Favrholdt (4 students).
Jun 2007	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2007, 10 ECTS), Lars Arge (19 students).
Oct 2006	Course examiner, Department of Computer Science, University of Copenhagen. The 6th STL Workshop, Jyrki Katajainen (6 students).
Oct 2006	MSc thesis examiner, Martin Ehmsen, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Kim Skak Larsen.
Jul 2006	MSc thesis examiner, Jacob Allerelli, Department of Mathematics and Computer Science, University of Southern Denmark, Odense. Advisor Rolf Fagerberg.
Jun 2006	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2006, 10 ECTS), Lars Arge (11 students).
Jul 2005	Course examiner, Department of Computer Science, Aarhus University. Dynamic Algorithms (Q4 2005, 5 ECTS), Gudmund Skovbjerg Frandsen (21 students).
Jun 2005	Course examiner, Department of Computer Science, Aarhus University. IO Algorithms (Q3+Q4 2005, 10 ECTS), Lars Arge (12 students).
Jul 2004	Course examiner, Department of Computer Science, Aarhus University. Dynamic Algorithms (Q4 2004, 5 ECTS), Gudmund Skovbjerg Frandsen (17 students).

Department of Mathematics and Computer Science, University of Southern Denmark, Odense

Mar 2024	Member of assessment committee for Assistant/Associate Professor Positions in Algorithms.

PhD Committee

Feb 2025	PhD committeee member, Marko Grgurovic, University of Primorska, Kopper, Slovenia.
Aug 2023	PhD committee member, Kaiyu Wu, University of Waterloo, Ontario, Canada.
Apr 2018	PhD committee member, Hicham El Zein, University of Waterloo, Ontario, Canada.
Sep 2014	PhD opponent, Jan Bulanek, Charles University, Prague, Czech Republic.
Nov 2011	PhD committee member, Djamal Belazzougui, LIAFA, Université Paris Diderot–Paris 7, Paris.
Oct 2011	PhD committee member, Andrea Campagna, IT University of Copenhagen.
Dec 2008	PhD committee member, Deepak Ajwani, Max Planck Institute for Computer Science, Saarbrücken, Germany.
Aug 2008	PhD committee member, Karim Douieb, Université Libre de Bruxelles, Belgium.
Sep 2006	PhD committee member, Anders Gidestam, Chalmers Technical University, Goteborg, Sweden.

Grant Reviewer

Aug 2025	Review panel of the Swedish Research Council, panel Natural and Engineering Sciences (NT-Q).
Jul 2020	Swiss National Science Foundation.
Mar 2019	United States - Israel Binational Science Foundation.
May 2017	German Research Foundation (DFG), Excellence Strategy by the German Federal and State Governments to Promote Science and Research at German Universities, Research panel on Engineering Sciences.

Dec 2016 German Research Foundation (DFG), Priority Programme "Algorithms for Big Data" (SPP 1736/2).

Oct 2014 The Research Council of Norway, ICT Panel 1, FRINATEK Applications.

Dec 2013 German Research Foundation (DFG), Priority Programme "Algorithms for Big Data" (SPP 1736).

Oct 2013 The Research Council of Norway, ICT Panel 2, FRINATEK Applications.

Oct 2012 The Research Council of Norway, ICT Panel 1, FRINATEK Applications.

Journal Reviewer

- ACM Journal of Experimental Algorithmics 2016, 2015, 2012, 2000, 1998
- ARS MATHEMATICA CONTEMPORANEA 2021, 2020
- Algorithmica 2022, 2020, 2017, 2015, 2014, 2013, 2012, 2011, 2010, 2008, 2005, 2004, 2002, 2001, 2000, 1999, 1998
- Applicable Algebra in Engineering, Communication and Computing 2003
- Applied Computing and Informatics 2015
- ETRI Journal 2004
- Fundamenta Informaticae 2021, 2014
- Higher-Order and Symbolic Computation 2003, 1999
- Information Processing Letters 2021, 2016, 2014, 2009, 2008, 2003, 2002, 2000
- Information and Computation 1998
- International Journal of Computational Geometry and Applications 2012, 2011
- Journal of Algorithms 1998
- Journal of Algorithms - Algorithms in Cognition, Informatics and Logic 2008
- Journal of Automata, Languages and Combinatorics 2003
- Journal of Combinatorial Optimization 2013
- Journal of Computational Biology 2016
- Journal of Computational Geometry 2020
- Journal of Discrete Algorithms 2011, 2000
- Journal of Functional Programming 1998
- Journal of Parallel and Distributed Computing 1998, 1997
- Journal of Systems and Software 2003
- Journal of the Association for Computing Machinery 1997
- Nordic Journal of Computing 2014, 1999
- PLOS ONE 2020
- SIAM Journal of Computing 2012, 2007, 2004
- Software: Practice and Experience 2006
- The Computer Journal 2005
- Theoretical Computer Science 2025, 2006, 2004, 2000
- Transactions on Algorithms 2017, 2013
- Transportation Science 2013

Conference Reviewer

2025 20th International Workshop on Algorithms and Data Structures (WADS25)

41st International Symposium on Computational Geometry (SOCG25)

50th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM25)

27th Workshop on Algorithm Engineering and Experiments (ALENEX)

2024 49th International Symposium on Mathematical Foundations of Computer Science (MFCS)

16th Latin American Symposium on Theoretical Informatics (LATIN)

26th Workshop on Algorithm Engineering and Experiments (ALENEX)

32nd Annual European Symposium on Algorithms (ESA)

2022 24th Workshop on Algorithm Engineering and Experiments (ALENEX)

5th SIAM Symposium on Simplicity in Algorithms (SOSA)

33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

2021 32th Annual International Symposium on Algorithms and Computation (ISAAC)

23rd International Symposium on Fundamentals of Computation Theory (FCT)

53rd Annual ACM Symposium on Theory of Computing (STOC)

29th Annual European Symposium on Algorithms (ESA)

2020 28th Annual European Symposium on Algorithms (ESA)

17th Scandinavian Workshop on Algorithm Theory (SWAT)

37th Annual Symposium on Theoretical Aspects of Computer Science (STACS)

31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

52nd Annual ACM Symposium on Theory of Computing (STOC)

2019 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

46th International Colloquium on Automata, Languages, and Programming (ICALP)

30th International Workshop on Combinatorial Algorithms (IWOCA)

36th Annual Symposium on Theoretical Aspects of Computer Science (STACS)

2018 45th International Colloquium on Automata, Languages, and Programming (ICALP)

29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

2017 28th Annual International Symposium on Algorithms and Computation (ISAAC)

44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)

28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

43rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)

2016 57th Annual Symposium on Foundations of Computer Science (FOCS)

35th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)

48th Annual ACM Symposium on Theory of Computing (STOC)

33rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)

27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

18th Workshop on Algorithm Engineering and Experiments (ALENEX)

2015 23rd Annual European Symposium on Algorithms (ESA)

56th Annual Symposium on Foundations of Computer Science (FOCS)

9th International Frontiers of Algorithmics Workshop (FAW)

17th Workshop on Algorithm Engineering and Experiments (ALENEX)

31st European Workshop on Computational Geometry (EuroCG)

26th International Workshop on Combinatorial Algorithms (IWOCA)

2014 14th Scandinavian Workshop on Algorithm Theory (SWAT14)

41st International Colloquium on Automata, Languages, and Programming (ICALP)

25th Annual International Symposium on Algorithms and Computation (ISAAC)

25th International Workshop on Combinatorial Algorithms (IWOCA)

39th International Symposium on Mathematical Foundations of Computer Science (MFCS)

25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

2013 21st Annual European Symposium on Algorithms (ESA)

54th Annual Symposium on Foundations of Computer Science (FOCS)

13th International Workshop on Algorithms and Data Structures (WADS)

40th International Colloquium on Automata, Languages, and Programming (ICALP)

32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)

27th IEEE International Parallel & Distributed Processing Symposium (IPDPS)

24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

19th International Symposium on Fundamentals of Computation Theory (FCT)

24rd International Workshop on Combinatorial Algorithms (IWOCA)

7th International Conference on Language and Automata Theory and Applications (LATA)

2012 23th Annual International Symposium on Algorithms and Computation (ISAAC)

20th Annual European Symposium on Algorithms (ESA)

53rd Annual Symposium on Foundations of Computer Science (FOCS)

39th International Colloquium on Automata, Languages, and Programming (ICALP)

31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)

10th Latin American Symposium on Theoretical Informatics (LATIN)

23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

14th Workshop on Algorithm Engineering and Experiments (ALENEX)

23rd International Workshop on Combinatorial Algorithms (IWOCA)

2011 38th International Colloquium on Automata, Languages, and Programming (ICALP)

30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)

10th International Symposium on Experimental Algorithms (SEA)

6th International Computer Science Symposium in Russia (CSR)

43rd Annual ACM Symposium on Theory of Computing (STOC)

22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

22nd International Workshop on Combinatorial Algorithms (IWOCA)

3rd Workshop on Massive Data Algorithmics (MASSIVE)

2010 21th Annual International Symposium on Algorithms and Computation (ISAAC)

12th Workshop on Algorithm Engineering and Experiments (ALENEX)

21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

9th Latin American Symposium on Theoretical Informatics (LATIN)

12th Scandinavian Workshop on Algorithm Theory (SWAT)

2009 41st Annual ACM Symposium on Theory of Computing (STOC)

50th Annual Symposium on Foundations of Computer Science (FOCS)

17th Annual European Symposium on Algorithms (ESA)

20th Annual International Symposium on Algorithms and Computation (ISAAC)

11th Workshop on Algorithm Engineering and Experiments (ALENEX)

20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

36th International Colloquium on Automata, Languages, and Programming (ICALP)

1st Workshop on Massive Data Algorithmics (MASSIVE)

8th International Symposium on Experimental Algorithms (SEA)

21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)

11th International Workshop on Algorithms and Data Structures (WADS)

2008 15th Scandinavian Workshop on Algorithm Theory (SWAT16)

11th Scandinavian Workshop on Algorithm Theory (SWAT08)

19th Annual Symposium on Combinatorial Pattern Matching (CPM08)

Computability in Europe 2008 - Logic and Theory of Algorithms (CiE)

40th Annual ACM Symposium on Theory of Computing (STOC)

IPDPS 2008 - IEEE International Parallel & Distributed Processing Symposium (IPDPS)

7th International Workshop on Experimental Algorithms (WEA)

2007 15th Annual European Symposium on Algorithms (ESA)

34th International Colloquium on Automata, Languages, and Programming (ICALP)

6th International Workshop on Experimental Algorithms (WEA)

27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)

18th Annual International Symposium on Algorithms and Computation (ISAAC)

International Workshop on Algorithmic Topics in Constraint Programming (cancelled) (ATCP)

18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

24th Annual Symposium on Theoretical Aspects of Computer Science (STACS)

10th International Workshop on Algorithms and Data Structures (WADS)

2006 47th Annual Symposium on Foundations of Computer Science (FOCS)

14th Annual European Symposium on Algorithms (ESA)

31st International Symposium on Mathematical Foundations of Computer Science (MFCS)

33rd International Colloquium on Automata, Languages, and Programming (ICALP)

17th Annual Symposium on Combinatorial Pattern Matching (CPM)

25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)

17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

10th Scandinavian Workshop on Algorithm Theory (SWAT)

2005 32nd International Colloquium on Automata, Languages, and Programming (ICALP)

9th International Workshop on Algorithms and Data Structures (WADS)

20th IEEE Conference on Computational Complexity (COMPLEXITY05)

7th Workshop on Algorithm Engineering and Experiments (ALENEX)

13th Annual European Symposium on Algorithms (ESA05)

16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

37th Annual ACM Symposium on Theory of Computing (STOC)

4th International Workshop on Efficient and Experimental Algorithms (WEA)

2004 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)

15th Annual Symposium on Combinatorial Pattern Matching (CPM)

12th Annual European Symposium on Algorithms (ESA)

24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)

3rd International Conference on Fun With Algorithms (FUN)

31st International Colloquium on Automata, Languages, and Programming (ICALP)

6th Latin American Symposium on Theoretical Informatics (LATIN)

15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)

9th Scandinavian Workshop on Algorithm Theory (SWAT)

2003 11th Annual European Symposium on Algorithms (ESA)

International Conference on Software Engineering and Formal Methods (SEFM)

20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)

5th Workshop on Algorithm Engineering and Experiments (ALENEX)

8th International Workshop on Algorithms and Data Structures (WADS)

2002 43rd Annual Symposium on Foundations of Computer Science (FOCS)

10th Annual European Symposium on Algorithms (ESA)

8th Scandinavian Workshop on Algorithm Theory (SWAT)

13th Annual Symposium on Combinatorial Pattern Matching (CPM)

34th Annual ACM Symposium on Theory of Computing (STOC)

11th Euromicro Conference on Parallel Distributed and Networking based Processing, Special session on Memory Hierachies (PDP)

2001 42nd Annual Symposium on Foundations of Computer Science (FOCS)

28th International Colloquium on Automata, Languages and Programming (ICALP)

9th Annual European Symposium on Algorithms (ESA)

2000 8th Annual European Symposium on Algorithms (ESA)

7th Scandinavian Workshop on Algorithm Theory (SWAT)

1999 Workshop on Algorithmic Aspects of Advanced Programming Languages (WAAAPL)

19th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)

3rd Workshop on Algorithm Engineering (WAE)

16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)

1998 2nd Workshop on Algorithm Engineering (WAE)

18th International Conference on Foundations of Software Technology & Theoretical Computer Science (FSTTCS)

6th Annual European Symposium on Algorithms (ESA)

25th International Colloquium on Automata, Languages, and Programming (ICALP)

15th Annual Symposium on Theoretical Aspects of Computer Science (STACS)

1996 4th Annual European Symposium on Algorithms (ESA)

5th Scandinavian Workshop on Algorithm Theory (SWAT)

ACM SIGPLAN International Conference on Functional Programming (ICFP)

Theory and Practice of Informatics – 23rd Seminar on Current Trends in Theory and Practice of Informatics (SOFSEM)

1995 ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM)

Theory and Practice of Software Development. 6th International Joint Conference CAAP/FASE (TAPSOFT)

1994 4th Scandinavian Workshop on Algorithm Theory (SWAT)

21st International Colloquium on Automata, Languages and Programming (ICALP)