Report from Dagstuhl Seminar 16101

Data Structures and Advanced Models of Computation on Big Data

Edited by

Alejandro Lopez-Ortiz[1], Ulrich Carsten Meyer[2], Markus E. Nebel[3], and Robert Sedgewick[4]

1 **University of Waterloo, CA,** alopez-o@uwaterloo.ca
2 **Goethe-Universität – Frankfurt a. M., DE,** umeyer@cs.uni-frankfurt.de
3 **TU Kaiserslautern, DE,** nebel@techfak.uni-bielefeld.de
4 **Princeton University, US,** rs@cs.princeton.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 16101 "Data Structures and Advanced Models of Computation on Big Data". In today's computing environment vast amounts of data are processed, exchanged and analyzed. The manner in which information is stored profoundly influences the efficiency of these operations over the data. In spite of the maturity of the field many data structuring problems are still open, while new ones arise due to technological advances.

The seminar covered both recent advances in the "classical" data structuring topics as well as new models of computation adapted to modern architectures, scientific studies that reveal the need for such models, applications where large data sets play a central role, modern computing platforms for very large data, and new data structures for large data in modern architectures.

The extended abstracts included in this report contain both recent state of the art advances and lay the foundation for new directions within data structures research.

Seminar March 6–11, 2016 – http://www.dagstuhl.de/16101
1998 ACM Subject Classification E.1 Data Structures, F.1 Computation by Abstract Devices, F.2 Analysis of Algorithms and Problem Complexity, H.3 Information Storage and Retrieval
Keywords and phrases algorithms, big data, cloud services, data structures, external memory methods, information theory, large data sets, streaming, web-scale
Digital Object Identifier 10.4230/DagRep.6.3.1
Edited in cooperation with Sebastian Wild

1 Executive Summary

Alejandro Lopez-Ortiz (University of Waterloo, CA)
Ulrich Carsten Meyer (Goethe-Universität – Frankfurt a. M., DE)
Markus E. Nebel (TU Kaiserslautern, DE)
Robert Sedgewick (Princeton University, US)

About the Seminar

Data structures provide ways of storing and manipulating data and information that are appropriate for the computational model at hand. Every such model relies on assumptions that we have to keep questioning. The aim of this seminar was to exchange ideas for new algorithms and data structures, and to discuss our models of computations in light of recent technological advances. This Dagstuhl seminar was the 12th in a series of loosely related Dagstuhl seminars on data structures.

Topics

The presentations covered both advances in classic fields, as well as new models for recent trends in computing, in particular the appearance of big-data applications.

The talks by Brodal (§3.5), Penschuck (§3.19), Silvestri (§3.29), and Vahrenhold (§3.31) covered methods in the *external-memory model* that models the situation that data does no longer fit into internal memory. This limit can be pushed a bit further by using *succinct* data structures, which use only as much memory as absolutely necessary. Such methods were covered in the talks of Hagerup (§3.14), Raman (§3.25), and Gog (§3.12). If the task is to generate large random instances, Even (§3.9) showed that one can delay generation of large parts until they really become requested.

Big-data applications rely on *parallel* computation to speed up processing. Bingmann (§3.4) announced the creation of a new framework to simplify developing such applications. Brodnik (§3.6) presented a parallel string-searching algorithm. Since such methods are often used in a distributed setting, the cost of *communication* can become dominating. Sanders (§3.24) discussed several algorithms from this point of view.

Iacono (§3.15) and Mehlhorn (§3.18) reported on recent advances in the long-standing open problem of *dynamic optimality* of binary search trees (BSTs). The classic problem of finding *optimal* static *BSTs* was taken up by Munro (§3.21): it becomes significantly harder if the objective is to minimize the number of *binary* comparisons instead of the classic ternary comparisons.

Wild (§3.32) used the connection between BSTs and recursion trees of Quicksort to analyze Quicksort on inputs with equal keys, including multi-way partitioning Quicksort. The latter was discussed in detail by Aumüller (§3.3) who presented a novel analysis for comparison-optimal partitioning.

Neumann (§3.22) introduced a new randomized dictionary implementation based on jumplists. Kopelowitz (§3.17) showed a much simplified solution to the file-maintenance problem.

In the context of large sparse graphs, Andoni (§3.2), Fagerberg (§3.10), and Sun (§3.30) showed how to exploit special structure in the input for algorithmic applications. Pettie (§3.28) showed how to efficiently answer connectivity queries in graphs when vertices can be deleted.

The seminar also enjoyed contributions on new algorithms: two innovative applications of hashing were presented by Silvestri (§3.29) and Jacob (§3.16); Meyer auf der Heide (§3.20) applied the primal-dual approach for *online algorithms* to online leasing problems. Driemel (§3.7) reported on clustering methods for time series.

The theory-focused talks were complemented by broader perspectives from practice: Ajwani (§3.1) presented his vision for future communication tools that are supported by context-sensitive agents, and Sedgewick (§3.27) sketched his views on the future of higher education. Finally, Salinger (§3.26) summarized the approaches taken by SAP to include data-specific algorithms directly in their HANA database system.

New models of computation were also discussed. Owens (§3.23) explained how the architecture of graphic cards calls for different approaches to design data structures; Dütsch (§3.8) discussed the cost of virtual address translation in several algorithms. Finally, Farach-Colton (§3.11) and Graefe (§3.13) challenged the claim that data structures are independent of the application they are used in: they showed intriguing examples where the context a data structure was applied in entailed unforeseen additional requirements.

Final Thoughts

The organizers would like to thank the Dagstuhl team for their continuous support; the welcoming atmosphere made the seminar both highly productive and enjoyable. They also thank all participants for their contributions to this seminar.

2 Table of Contents

3 Overview of Talks

3.1 Towards fully-informed communication

Deepak Ajwani (Bell Labs – Dublin, IE)

I described a vision of a software cognitive layer that provides users with all the information they need at the time of communication. To realize such a vision, a communication platform should understand a user's communication, link it to other information available and mine the linked information in real-time. I presented a range of open problems related to graph algorithms and graph systems, to address these challenges.

3.2 Parallel Algorithms for Geometric Graph Problems

Alex Andoni, Grisha Yaroslavtsev, Krzysiek Onak, and Sasho Nikolov

Main reference A. Andoni, A. Nikolov, K. Onak, G. Yaroslavtsev, "Parallel algorithms for geometric graph problems", in Proc. of the 46th Annual ACM Symp. on Theory of Computing (STOC'14), pp. 574–583, ACM, 2014.
URL http://dx.doi.org/10.1145/2591796.2591805

Motivated by modern parallel computing models (think MapReduce), we give a new algorithmic framework for geometric graph problems. Our framework applies to problems such as the Minimum Spanning Tree (MST) problem over a set of points in a low-dimensional space, which is useful for hierarchical agglomerative clustering. Our algorithm computes a $(1+\epsilon)$-approximate MST in a *constant* number of rounds of communication, while using total space and communication proportional to the size of the data only.

Our framework proves to have implications beyond the parallel models as well. For example, we consider the Earth-Mover Distance (EMD) problem, for which we obtain a new near-linear time algorithm as well as a first streaming algorithm (assuming we can pre-sort the points). Technically, our framework for EMD shows how to effectively break up a "big Linear Programming problem" into a number of "small Linear Programming problems," which can be later recombined using a dynamic programming approach.

3.3 News on Multi-Pivot Quicksort

Martin Aumüller (IT University of Copenhagen, DK)

Joint work of Martin Aumüller; Martin Dietzfelbinger; Clemens Heuberger; Daniel Krenn; Helmut Prodinger
Main reference M. Aumüller, M. Dietzfelbinger, C. Heuberger, D. Krenn, H. Prodinger, "Counting Zeros in Random Walks on the Integers and Analysis of Optimal Dual-Pivot Quicksort", arXiv:1602.04031 [math.CO], 2016.
URL http://arxiv.org/abs/1602.04031

We discuss two results with respect to multi-pivot quicksort.

In the first part of this talk, we present an exact average case analysis of two variants of dual-pivot quicksort, one with a non-algorithmic comparison-optimal partitioning strategy, the other with a closely related algorithmic strategy. For both we calculate the expected number of comparisons exactly as well as asymptotically, in particular, we provide exact expressions for the linear, logarithmic, and constant terms. An essential step is the analysis of zeros of lattice paths in a certain probability model. Furthermore, we show that the closely related algorithmic strategy yields a comparison-optimal dual-pivot algorithm.

In the second part of this talk, I will talk about rearranging elements to produce a partition. A substantial part of the partitioning cost is caused by rearranging elements. A rigorous analysis of an algorithm for rearranging elements in the partitioning step is carried out, observing mainly how often array cells are accessed during partitioning. The algorithm behaves best if 3 or 5 pivots are used. Experiments show that this translates into good cache behavior and is closest to predicting observed running times of multi-pivot quicksort algorithms.

3.4 Thrill: Distributed Big Data Batch Processing in C++

Timo Bingmann (KIT – Karlsruher Institut für Technologie, DE)

We present on-going work on a new distributed Big Data processing framework called Thrill. It is a C++ framework consisting of a set of basic scalable algorithmic primitives like mapping, reducing, sorting, merging, joining, and additional MPI-like collectives. This set of primitives goes beyond traditional Map/Reduce and can be combined into larger more complex algorithms, such as WordCount, PageRank, k-means clustering, and suffix sorting. These complex algorithms can then be run on very large inputs using a distributed computing cluster. Among the main design goals of Thrill is to lose very little performance when composing primitives such that small data types are well supported. Thrill thus raises the questions of a) how to design algorithms using the scalable primitives, b) whether additional primitives should be added, and c) if one can improve the existing ones using new ideas to reduce communication volume and latency.

3.5 External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates

Gerth Stølting Brodal (Aarhus University, DK)

Main reference G. Stølting Brodal, "External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates", in Proc. of the 33rd Annual Symp. on Theoretical Aspects of Computer Science (STACS'16), LIPIcs, Vol. 47, pp. 23:1-23:14, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.
URL http://dx.doi.org/10.4230/LIPIcs.STACS.2016.23

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.

3.6 Parallel Queries

Andrej Brodnik (University of Primorska, SI)

Joint work of Andrej Brodnik; Tanja Štular; Matevž Jekovec
Main reference M. Jekovec, A. Brodnik, "Parallel Query in the Suffix Tree", arXiv:1509.06167 [cs.DS], 2015.
URL http://arxiv.org/abs/1509.06167

When considering parallel queries, the researchers in past were mostly concerned with parallel execution of several queries on the same data structure. In this contribution we ask ourselves how p processors can be employed to jointly perform a single search under CREW PRAM model.

The query we study is a search of pattern P in a text T, where $|P| = m$ and $|T| = n$. Our presentation starts with employing automaton based approach first and later evolve it into an index based. For the later we show, that one can perform search in time $O(m/p + \log p)$, deploying $O(m + m \log p)$ work and using $O(n^2)$ space. We are also able to change the solution to $O(m/p \log p)$ time, $O(m \log p)$ work and $O(n \log p)$ space.

The trivial lower is, of course, $\Omega(m/p)$ time, $\Omega(m)$ work and $\Omega(n)$ space. It remains an open question whether it is achievable. Mind though, that in our solution we pay a log factor for communication among the processors. Another interesting open question is how the scheme can be used for an approximate searching.

3.7 Clustering time series under the Frechet distance

Anne Driemel (TU Eindhoven, NL)

Joint work of Anne Driemel; Amer Krivosija; Christian Sohler
Main reference A. Driemel, A. Krivosija, C. Sohler, "Clustering time series under the Fréchet distance", in Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'16), pp. 766–785, SIAM, 2016.
URL http://dx.doi.org/10.1137/1.9781611974331.ch55

The Frechet distance is a popular distance measure for curves. We study the problem of clustering time series under the Frechet distance. In particular, we give $(1+\varepsilon)$-approximation algorithms for variations of the following problem with parameters k and l. Given n univariate time series P, each of complexity at most m, we find k time series, not necessarily from P, which we call cluster centers and which each have complexity at most l, such that (a) the maximum distance of an element of P to its nearest cluster center or (b) the sum of these distances is minimized. Our algorithms have running time near-linear in the input size. To the best of our knowledge, our algorithms are the first clustering algorithms for the Frechet distance which achieve an approximation factor of $(1+\varepsilon)$ or better.

3.8 Algorithm Design Paradigms in the VAT-Model

Fabian Dütsch (Universität Münster, DE)

Recently, Jurkiewicz and Mehlhorn [ALENEX '13] observed that the cost of virtual address translation affects the practical runtime behavior of several fundamental algorithms on modern computers. In this talk, we extend their results to two dimensions and investigate the translation cost of some algorithm design paradigms. For this purpose, we analyze closest pair algorithms representing the divide and conquer, plane-sweep and randomized incremental construction paradigms in the VAT-model. Furthermore, we investigate the VAT-complexities of hashing and comparison-based searching. Finally, we verify the theoretical analyses by experimental results.

3.9 Sublinear Random Access Generators for Preferential Attachment Graphs

Guy Even (Tel Aviv University, IL)

Joint work of Reut Elvi; Guy Even; Moti Medina; Adi Rosen
Main reference G. Even, R. Levi, M. Medina, A. Rosen, "Sublinear Random Access Generators for Preferential Attachment Graphs", arXiv:1602.06159 [cs.DS], 2016.
URL http://arxiv.org/abs/1602.06159

We consider the problem of generating random graphs in evolving random graph models. In the standard approach, the whole graph is chosen randomly according to the distribution of the model before answering queries to the adjacency lists of the graph. Instead, we propose to answer queries by generating the graphs on-the-fly while respecting the probability space of the random graph model.

We focus on two random graph models: the Barabási-Albert Preferential Attachment model (BA-graphs) and the random recursive tree model. We present sublinear randomized generating algorithms for both models. Per query, the running time, the increase in space, and the number of random bits consumed are $\text{poly}\log(n)$ with probability $1 - 1/\text{poly}(n)$, where n denotes the number of vertices.

This result shows that, although the BA random graph model is defined sequentially, random access is possible without chronological evolution. In addition to a conceptual contribution, on-the-fly generation of random graphs can serve as a tool for simulating sublinear algorithms over large BA-graphs.

3.10 On Routing in Geometric Spanners

Rolf Fagerberg (University of Southern Denmark – Odense, DK)

Joint work of Rolf Fagerberg; Jit Bose; André van Renssen; Sander Verdonschot
Main reference P. Bose, R. Fagerberg, A. van Renssen, S. Verdonschot, "Optimal Local Routing on Delaunay Triangulations Defined by Empty Equilateral Triangles", SIAM Journal on Computing, 44(6):1626–1649, 2015.
URL http://dx.doi.org/10.1137/140988103

A geometric spanner on a point set in the Euclidean plane is a straight-line graph on the point set in which any pair u, v of points has a path between them which is not more than a constant factor longer than the direct distance between u and v. The constant factor is called the spanning ratio. Such spanners have practical applications in e.g. add-hoc wireless networks, and can be seen as static data structures for approximate route finding in geometric graphs. Two classical constructions of geometric spanners are Yao$_k$-graphs and θ_k-graphs.

Recently, bounds on the spanning ratio of these graphs have evolved substantially. We give an overview of the current knowledge, and then focus on a result on the half-θ_6-graph which shows that while its spanning ratio is 2, actually following such a short path between u and v using local routing is only feasible in one direction, whereas in the other direction the factor becomes $5/\sqrt{3} = 2.886\ldots$. We do this by giving a lower bound and a routing algorithm matching this bound.

3.11 Migrating a data structure from one system to another

Martin Farach-Colton (Rutgers University – Piscataway, US)

Joint work of Michael Bender; Martin Farach-Colton; Bradley Kuszmaul; Don Porter; Rob Johnson

Data structures are typically design independent of any particular use case. Sometimes, data structures get deployed in a system, during which they are optimized for the specific needs of the system. In this case, I discuss the experience of migrating a data structure from one system to another. Specifically, I discuss deploying an external-memory dictionary that has been optimized for a data base key-value store in a file system.

3.12 Practical Compact Indexes for Top-k Document Retrieval

Simon Gog (KIT – Karlsruher Institut für Technologie, DE), Gonzalo Navarro, and Roberto Konow

Joint work of Simon Gog; Gonazlo Navarro; Roberto Konow

In this talk we present a fast and compact index for top-k document retrieval on general string collections. For a given string pattern P, the index returns the k documents where P occurs most often, i.e. we score by pattern frequency. We adapt a linear-space and optimal-time theoretical solution of Navarro and Nekrich [1], whose implementation poses various algorithm engineering challenges. While a naive implementation of the optimal solution is estimated to require around $80n$ bytes for a text collection of n symbols, we show how this can be improved to $2.5n - 3.0n$ bytes (including the text). The resulting index can still answer queries within microseconds and outperforms all previous work.

References

1 Gonzalo Navarro, Yakov Nekrich. *Top-k document retrieval in optimal time and linear space.* SODA 2012: 1066-1077

3.13 Lock-free data structures

Goetz Graefe (HP Labs – Madison, US)

I am trying to understand lock-free data structures, their rules and their limitations, and I appreciate the other attendees' help in grasping the fundamentals and the subtleties. This is part of a larger effort to understand optimistic and pessimistic concurrency control as well as transaction isolation levels as known in SQL databases.

3.14 Succinct Choice Dictionaries

Torben Hagerup (Universität Augsburg, DE) and Frank Kammer

The choice dictionary is introduced as a data structure that can be initialized with a parameter n in $\{1, 2, \ldots\}$ and subsequently maintains an initially empty subset S of $\{1, \ldots, n\}$ under insertion, deletion, membership queries and an operation choice that returns an arbitrary element of S. The choice dictionary appears to be fundamental in space-efficient computing. We show that there is a choice dictionary that can be initialized with n and an additional parameter t in $\{1, 2, \ldots\}$ and subsequently occupies $n + O(n(t/w)^t + \log n)$ bits of memory and executes each of the four operations insert, delete, contains (i.e., a membership query) and choice in $O(t)$ time on a word RAM with a word length of $w = \Omega(\log n)$ bits. In particular, with $w = \Theta(\log n)$, we can support insert, delete, contains and choice in constant time using $n + O(n/(\log n)^t)$ bits for arbitrary fixed t. We extend our results to maintaining several pairwise disjoint subsets of $\{1, \ldots, n\}$.

A static representation of a subset S of $\{1, \ldots, n\}$ that consists of $n + s$ bits $b_1, \ldots, b_{n+s}$ is called systematic if $b_l = 1 \iff l$ is in S for $l = 1, \ldots, n$ and is said to have redundancy s. We extend the former definition to dynamic data structures and prove that the minimum redundancy of a systematic choice dictionary with parameter n that executes every operation in $O(t)$ time on a w-bit word RAM is $\Theta(n/(tw))$. Allowing a redundancy of $\Theta(n \log(t \log n)/(t \log n) + n^\varepsilon)$ for arbitrary fixed $\varepsilon > 0$, we can support additional $O(t)$-time operations p-rank and p-select that realize a bijection from S to $\{1, \ldots, |S|\}$ and its inverse. The bijection may be chosen arbitrarily by the data structure, but must remain fixed as long as S is not changed. In particular, an element of S can be drawn uniformly at random in constant time with a redundancy of $O(n \log \log n/ \log n)$.

We study additional space-efficient data structures for subsets S of $\{1,\ldots,n\}$, including one that supports only insertion and an operation extract-choice that returns and deletes an arbitrary element of S. All our main data structures can be initialized in constant time and support efficient iteration over the set S, and we can allow changes to S while an iteration over S is in progress. We use these abilities crucially in designing the most space-efficient algorithms known for solving a number of graph and other combinatorial problems in linear time. In particular, given an undirected graph G with n vertices and m edges, we can output a spanning forest of G in $O(n+m)$ time with at most $(1+\varepsilon)n$ bits for arbitrary fixed $\varepsilon > 0$, and if G is connected, we can output a shortest-path spanning tree of G rooted at a designated vertex in $O(n+m)$ time with $n\log_2 3 + O(n/(\log n)^t)$ bits for arbitrary fixed t in $\{1,2,\ldots\}$.

3.15 Weighted dynamic finger in binary search trees

John Iacono (New York University, US)

License Creative Commons BY 3.0 Unported license
© John Iacono
Joint work of John Iacono; Stefan Langerman
Main reference J. Iacono, S. Langerman, "Weighted dynamic finger in binary search trees", in Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'16), pp. 672–691, SIAM, 2016.
URL http://dx.doi.org/10.1137/1.9781611974331.ch49

It is shown that the online binary search tree data structure GreedyASS performs asymptotically as well on a sufficiently long sequence of searches as any static binary search tree where each search begins from the previous search (rather than the root). This bound is known to be equivalent to assigning each item i in the search tree a positive weight w_i and bounding the search cost of an item in the search sequence $s_1,\ldots,s_m$ by

$$O\left(1+\log\frac{\sum_{\min(s_{i-1},s_i)\le x\le\max(s_{i-1},s_i)} w_x}{\min(w_{s_i},w_{s_{i-1}})}\right)$$

amortized. This result is the strongest finger-type bound to be proven for binary search trees. By setting the weights to be equal, one observes that our bound implies the dynamic finger bound. Compared to the previous proof of the dynamic finger bound for Splay trees, our result is significantly shorter, stronger, simpler, and has reasonable constants.

3.16 Fast Output-Sensitive Matrix Multiplication

Riko Jacob (IT University of Copenhagen, DK)

License Creative Commons BY 3.0 Unported license
© Riko Jacob
Joint work of Riko Jacob; Mortten Stöckel
Main reference R. Jacob, M. Stöckel, "Fast Output-sensitive Matrix Multiplication", in Proc. of the 23rd Annual European Symp. on Algorithms (ESA'15), LNCS, Vol. 9294, pp. 766–778, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-662-48350-3_64

We consider the problem of multiplying two $U\times U$ matrices A and C of elements from a field $\mathbb{F}$. We present a new randomized algorithm that can use the known fast square matrix multiplication algorithms to perform fewer arithmetic operations than the current state of the art for output matrices that are sparse.

In particular, let ω be the best known constant such that two dense $U \times U$ matrices can be multiplied with $O(U^{\omega})$ arithmetic operations. Further denote by N the number of nonzero entries in the input matrices while Z is the number of nonzero entries of matrix product AC. We present a new Monte Carlo algorithm that uses $\tilde{O}\left(U^2\left(\frac{Z}{U}\right)^{\omega-2}+N\right)$ arithmetic operations and outputs the nonzero entries of AC with high probability. For dense input, i.e., $N=U^2$, if Z is asymptotically larger than U, this improves over state of the art methods, and it is always at most $O(U^{\omega})$. For general input density we improve upon state of the art when N is asymptotically larger than $U^{4-\omega}Z^{\omega-5/2}$.

The algorithm is based on dividing the input into "balanced" subproblems which are then compressed and computed. The new subroutine that computes a matrix product with balanced rows and columns in its output uses time $\tilde{O}\left(UZ^{(\omega-1)/2}+N\right)$ which is better than the current state of the art for balanced matrices when N is asymptotically larger than $UZ^{\omega/2-1}$, which always holds when $N=U^2$.

In the I/O model – where M is the memory size and B is the block size – our algorithm is the first nontrivial result that exploits cancellations and sparsity of the output. The I/O complexity of our algorithm is $\tilde{O}\left(U^2(Z/U)^{\omega-2}/(M^{\omega/2-1}B)+Z/B+N/B\right)$, which is asymptotically faster than the state of the art unless M is large.

3.17 File Maintenance: When in Doubt, Change the Layout!

Tsvi Kopelowitz (University of Michigan – Ann Arbor, US)

Joint work of Michael Bender; Jeremy Fineman; Seth Gilbert; Tsvi Kopelowitz; Pablo Montas

In this talk I will describe a new deamortized solution to the sequential-file-maintenance problem. The data structure uses several new tools, for solving this historically complicated problem. These tools include an unbalanced ternary-tree layout embedded in the sparse table, a level-based approach for triggering, and one-way rebalancing.

3.18 Self-Organizing Binary Search Trees: Recent Results

Kurt Mehlhorn

Joint work of Parinya Chalermsook; Mayank Goswami; Lazlo Kosma; Kurt Mehlhorn; Thatchaphol Saranurak
Main reference P. Chalermsook, M. Goswami, L. Kozma, K. Mehlhorn, T. Saranurak, "Pattern-Avoiding Access in Binary Search Trees", in Proc. of the IEEE 56th Annual Symp. on Foundations of Computer Science (FOCS'15), pp. 410–423, IEEE CS, 2015.
URL http://dx.doi.org/10.1109/FOCS.2015.32

The dynamic optimality conjecture is perhaps the most fundamental open question about binary search trees (BST). It postulates the existence of an asymptotically optimal online BST, i.e. one that is constant factor competitive with any BST on any input access sequence. The two main candidates for dynamic optimality in the literature are splay trees [Sleator and Tarjan, 1985], and Greedy [Lucas, 1988; Munro, 2000; Demaine et al. 2009]. Despite BSTs being among the simplest data structures in computer science, and despite extensive effort over the past three decades, the conjecture remains elusive. Dynamic optimality is trivial for almost all sequences: the optimum access cost of most length-n sequences is $\Theta(n \log n)$, achievable by any balanced BST.

Thus, the obvious missing step towards the conjecture is an understanding of the "easy" access sequences. Preorder sequences (the access sequence arises from a preorder traversal of a tree) can easily be served in linear time by an off-line algorithms. No online BST is known to serve them in linear time.

We prove (FOCS 2015) two different relaxations of the traversal conjecture for Greedy: (i) Greedy with an arbitrary initial tree is almost linear for preorder sequences. (ii) Greedy with a fixed initial tree is in fact linear. These statements are corollaries of our more general results that express the complexity of access sequences in terms of a pattern avoidance.

Splay trees satisfy the so-called *access lemma*. Many of the nice properties of splay trees follow from it. *What makes self-adjusting binary search trees (BSTs) satisfy the access lemma?* In our ESA 2015 paper, we give sufficient conditions for the access lemma to hold and give strong hints of their necessity.

3.19 Generating Massive Scale-Free Networks under Resource Constraints

Ulrich Carsten Meyer (Goethe-Universität – Frankfurt a. M., DE) and Manuel Penschuck (Goethe-Universität – Frankfurt a. M., DE)

Main reference U. Meyer, M. Penschuck, "Generating Massive Scale-Free Networks under Resource Constraints", in Proc. of the 18th Workshop on Algorithm Engineering and Experiments (ALENEX'16), pp. 39–52, SIAM, 2016.
URL http://dx.doi.org/10.1137/1.9781611974317.4

Random graphs as mathematical models of massive scale-free networks have recently become very popular. For experimental evaluation and in order to provide artificial data sets, huge instances of such networks actually need to be generated. We consider generation methods for random graph models based on linear preferential attachment under limited computational resources and investigate our techniques using the well-known Barabási-Albert (BA) graph model. We present the two I/O-efficient BA generators, MP-BA and TFP-BA, for the external-memory (EM) model and then extend MP-BA to massive parallelism based on but not limited to GPGPU.

3.20 Online Resource Leasing

Friedhelm Meyer auf der Heide (Universität Paderborn, DE)

Main reference S. Abshoff , P. Kling, C. Markarian, F. Meyer auf der Heide, P. Pietrzyk, "Towards the price of leasing online", Journal of Combinatorial Optimisation, pp. 1–20, 2015.
URL http://dx.doi.org/10.1007/s10878-015-9915-5

We consider online leasing problems in which demands arrive over time and need to be served by leased resources. Each resource can be leased for K different durations, each incurring a different cost (longer leases cost less per time unit). This model is a natural generalization of Meyerson's Parking Permit Problem (FOCS 2005). In the talk, I review Meyerson's result and present it using the primal-dual approach. In addition, I present new online leasing variants of classical problems like facility location and set cover, and present primal-dual-based online algorithms for them together with their competitive analysis.

3.21 Optimal search trees with 2-way comparisons

Ian Munro (University of Waterloo, CA) and Mordecai Golin (HKUST – Kowloon, HK)

Joint work of Marek Chrobak; Mordecai Golin; J. Ian Munro; Neal E. Young

This talk is about finding a polynomial time algorithm that you probably thought was known almost a half century ago, but it wasn't. The polynomial time algorithm is still rather slow and requires a lot of space to solve, so we also look at extremely good and fast approximate solutions.

In 1971, Knuth gave an $O(n^2)$-time algorithm for the now classic problem of finding an optimal binary search tree. Knuth's algorithm works only for search trees based on 3-way comparisons, but most modern programming languages and computers support only 2-way comparisons ($<$, $=$ and $>$). Until this work, the problem of finding an optimal search tree using 2-way comparisons remained open – polynomial time algorithms were known only for restricted variants. We solve the general case, giving (i) an $O(n^4)$-time algorithm and (ii) a linear time algorithm that gives a tree with expected search cost within 2 comparisons of the optimal.

3.22 Randomized k-Jumplists

Elisabeth Neumann (TU Kaiserslautern, DE)

Joint work of Elisabeth Neumann; Markus Nebel; Sebastian Wild

In this talk, I presented an extension of randomized jumplists, introduced by Brönnimann, Cazals and Durand. To improve search costs, the number of jump-pointers per node have been increased from one to k, ($k > 1$), to allow faster navigation through the list. Pointer targets are chosen at random such that the pointer structure is (strongly) nested and no two jump-pointers point to the same node. I presented algorithms for search, construction and insertion, extending algorithms for regular jumplists, all of which run in expected logarithmic time. Finally, I analysed the expected costs of searching and came to the conclusion that k-jumplists ($k > 1$) outperform regular jumplists if binary search is used to determine which pointer to follow during the search.

3.23 Dynamic Data Structures for the GPU

John D. Owens (University of California, Davis, US)

Joint work of Martin Farach-Colton; John D. Owens

Today's GPU programming environments feature few general-purpose data structures. Only a handful of those can be *constructed* on the GPU, and to first order, none of them can be *updated* on the GPU. We aim to develop a family of GPU data structures that permit dynamic updates without rebuilding, and identify cross-cutting issues – e.g., modeling the memory hierarchy, leveraging task parallelism vs. cooperative parallelism, and choosing the right granularity of GPU parallelism for data-structure operations – that will affect their design.

3.24 Communication efficient algorithms

Peter Sanders (KIT – Karlsruher Institut für Technologie, DE)

I proposed to have a closer look at algorithms that have sublinear bottleneck communication volume. Examples were given for duplicate detection, distributed Bloom filters and various top-k problems. The talk is based on the two papers [1, 2].

References

1 Peter Sanders and Sebastian Schlag and Ingo Müller. *Communication Efficient Algorithms for Fundamental Big Data Problems.* IEEE Int. Conf. on Big Data. 2013.

2 Lorenz Hübschle-Schneider and Peter Sanders. *Communication Efficient Algorithms for Top-k Selection Problems.* IPDPS 2016.

3.25 Encoding Data Structures

Rajeev Raman (University of Leicester, GB)

Main reference R. Raman, "Encoding Data Structures," in Proc. of the 9th Int'l Workshop on Algorithms and Computation (WALCOM'15), LNCS, Vol. 8973, pp. 1-7, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-15612-5_1

Note: Survey talk based on several papers.

Driven by the increasing need to analyze and search for complex patterns in very large data sets, the area of compressed and succinct data structures has grown rapidly in the last 10–15 years. Such data structures have very low memory requirements, allowing them to fit into the main memory of a computer, which in turn avoids expensive computation on hard disks.

This talk will focus on a topic that has become popular recently: encoding "the data structure" itself. Some data structuring problems involve supporting queries on data, but the queries that need to be supported do not allow the original data to be deduced from the queries. This presents opportunities to obtain space savings even when the data is incompressible, by pre-processing the data, extracting only the information needed to answer the queries, and then deleting the data. The minimum information needed to answer the queries is called the effective entropy of the problem: precisely determining the effective entropy can involve interesting combinatorics.

3.26 Towards a Web-scale Data Management Ecosystem Demonstrated by SAP HANA

Alejandro Salinger (SAP SE – Walldorf, DE)

Joint work of Franz Faerber; Jonathan Dees; Martin Weidner; Stefan Baeuerle; Wolfgang Lehner
Main reference F. Faerber, J. Dees, M. Weidner, S. Bäuerle, W. Lehner, "Towards a web-scale data management ecosystem demonstrated by SAP HANA", in Proc. of the IEEE 31st Int'l Conf. on Data Engineering (ICDE'15), pp. 1259–1267, 2015.
URL http://dx.doi.org/10.1109/ICDE.2015.7113374

The requirements for modern data management systems have changed in the last years mainly due to the growth in application space with different usage patterns, changes in underlying hardware, and growing data volumes. In this scenario, a solution must deal with a multidimensional problem space with multiple domain-specific data types, data consumption models, consistence notions, and query languages, among others. As no single engine can handle all the different dimensions, it is natural to tackle and optimize each dimension with specialized approaches. However, we argue for a deep integration of individual engines into a single coherent and consistent data management ecosystem that provides a common understanding of the overall business semantics.

We describe SAP HANA as an example of what such a holistic but also flexible data management ecosystem could look like. We describe the system's in-memory column store engine as well as its specialized engines that allow for data processing beyond relational data (e.g., time series, text search, graph), and we argue about the advantages of bringing data processing closer to the data itself.

We then describe HANA's Scale-Out Extension (SAP HANA Vora) with its low footprint, highly scalable processing engines as well as the system's integration with the Hadoop ecosystem. We give an example of the techniques to store and process large amounts of time series data such as compression based on the combination of several approximation methods with varying accuracy in the representation for different data warmness levels.

3.27 A 21st Century Model for Disseminating Knowledge

Robert Sedgewick (Princeton University, US)

In this talk, we describe a scalable model for teaching and learning based on a combination of studio-produced video lectures, a web repository of associated materials, and an authoritative classic textbook. The approach has already proven effective for teaching algorithms and data structures, the analysis of algorithms, and analytic combinatorics, and will be further tested in the coming year with a new computer science textbook and associated materials that can serve as a basis for a first course in computer science that can stand alongside traditional first courses in physics, chemistry, economics, and other disciplines.

3.28 Connectivity Oracles

Seth Pettie (University of Michigan, Ann Arbor, MI, US)

Joint work of Ran Duan; Seth Pettie
Main reference R. Duan and S. Pettie, "Connectivity oracles for failure prone graphs", in Proc. of the 42nd ACM Symp. on Theory of Computing (STOC'10), pp. 465–474, 2010.
URL http://dx.doi.org/10.1145/1806689.1806754

A d-failure connectivity oracle is a data structure for undirected graphs that can answer connectivity queries after any d vertices have been deleted. The best d-failure connectivity oracles that have fast query time either have exorbitant preprocessing or linear deletion time. In this talk I'll discuss a simplified variant of the Duan-Pettie (2010) d-failure connectivity oracle that has polynomial (in n) preprocessing, polynomial (in d) time for vertex deletion, and $O(d)$ time to answer a connectivity query. A new type of graph decomposition is used, which is inspired by the Fürer-Raghavachari algorithm for approximating the minimum-degree spanning tree.

3.29 I/O-Efficient Similarity Join

Francesco Silvestri (IT University of Copenhagen, DK)

Joint work of Rasmus Pagh; Ninh Pham; Francesco Silvestri; Morten Stöckel
Main reference R. Pagh, N. Pham, F. Silvestri, M. Stöckel, "I/O-Efficient Similarity Join", in Proc. of the 23rd Annual European Symp. on Algorithms (ESA'15), LNCS, Vol. 9294, pp. 941–952, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-662-48350-3_78

We present an I/O-efficient algorithm for computing similarity joins based on locality-sensitive hashing (LSH). In contrast to the filtering methods commonly suggested our method has provable sub-quadratic dependency on the data size. Further, in contrast to straightforward implementations of known LSH-based algorithms on external memory, our approach is able to take significant advantage of the available internal memory: Whereas the time complexity of classical algorithms includes a factor of M^ρ , where ρ is a parameter of the LSH used, the I/O complexity of our algorithm merely includes a factor $(N/M)^\rho$, where N is the data size and M is the size of internal memory. Our algorithm is randomized and outputs the correct result with high probability. It is a simple, recursive, cache-oblivious procedure, and we believe that it will be useful also in other computational settings such as parallel computation.

3.30 Fast construction of graph sparsification: graphs, ellipsoids, and balls-into-bins

He Sun (University of Bristol, GB)

Joint work of Yin Tat Lee; He Sun
Main reference Y. Tat Lee, H. Sun, "Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time", in Proc. of the IEEE 56th Annual Symp. on Foundations of Computer Science (FOCS'15), pp. 250–269, IEEE CS, 2015; pre-print available as arXiv:1508.03261 [cs.DS], 2015.
URL http://dx.doi.org/10.1109/FOCS.2015.24
URL http://arxiv.org/abs/1508.03261

Spectral sparsification is the procedure of approximating a graph by a sparse graph such that many properties between these two graphs are preserved. Over the past decade, spectral sparsification has become a standard tool in speeding up runtimes of algorithms for various combinatorial and learning problems.

In this talk I will present our recent work on constructing a linear-sized spectral sparsification in almost-linear time. In particular, I will discuss some interesting connections among graphs, ellipsoids, and balls-into-bins processes.

3.31 Revisiting the Construction of SSPDs in the Presence of Memory Hierarchies

Jan Vahrenhold (Universität Münster, DE)

Joint work of Sylvie Temme; Jan Vahrenhold

We revisit the randomized internal-memory algorithm of Abam and Har-Peled [SoCG 2010] for constructing a semi-separated pair decomposition (SSPD) for N points in $\mathbb{R}^d$ in the context of the cache-oblivious model of computation. Their algorithm spends $O(n\varepsilon^{-d}\log_2 N)$ time (assuming that the floor function can be evaluated in constant time, $O(n\varepsilon^{-d}\log_2^2 N)$ time otherwise) in expectation and produces an SSPD of linear size in which each point participates in only a logarithmic number of pairs with high probability.

Using a modified analysis of their algorithm and several cache-oblivious techniques for tree construction, labeling, and traversal, we obtain a cache-oblivious algorithm that spends an expected number of $O(\text{sort}(n\varepsilon^{-d})\log_2 N)$ memory transfers.

3.32 Quicksort with Equal Keys

Sebastian Wild (TU Kaiserslautern, DE)

Joint work of Martin Aumüller; Martin Dietzfelbinger; Conrado Martínez; Markus Nebel; Sebastian Wild

In this talk, I present the first analysis of generalized Quicksort on inputs with equal keys, confirming in part a conjecture of Sedgewick and Bentley.

I consider Quicksort variants which partition inputs into s segments, around $s-1$ pivots chosen as order statistics from a sample, generalizing on Quicksort variants used in practice.

The input model is random u-ary words, i. e., n elements chosen i. i. d. uniformly from $\{1, \ldots, u\}$. Generalized Quicksort needs on average $\frac{\bar{a}}{\mathcal{H}} n \ln(u) + O(n)$ ternary comparisons to sort a random u-ary word of length n, provided that $u = \omega(1)$ and $u = O(n^{1/3-\varepsilon})$. The corresponding number for the classic model of random permutations is $\frac{\bar{a}}{\mathcal{H}} n \ln(n) + O(n)$, so the same relative speedup from sampling and multi-way partitioning is attained in both input models. Here, $\frac{\bar{a}}{\mathcal{H}}$ is a (known) constant that depends only on the used partitioning algorithm and the pivot-sampling scheme.

The analysis relies on the connection of Quicksort and search trees: I reduce the analysis of Quicksort to determining the path length in search trees built from inserting elements drawn i.i.d. from $\{1, \ldots, u\}$.

4 Open problems

The seminar included an open problem session during which the following problems were discussed.

4.1 Open Problem 1

Deepak Ajwani (Bell Labs – Dublin, IE)

The problem that I posed was the following: Given a directed graph G, find an acyclic subgraph D, such that TransitiveClosure(D) is as close to TransitiveClosure(G) as possible.

Since the transitive closure of D can only be a subset of transitive closure of G and it is a 0-1 matrix, the problem can also be formulated as "Given a directed graph G, find an acyclic subgraph D, such that TransitiveClosure(D) has as many ones as possible."

This is an interesting theoretical problem (particularly as the related problem of minimum feedback arc set is APX-hard). But my motivation for this problem came from a practical consideration, namely cleaning up crowd-sourced taxonomies. These taxonomies capture the specificity or generality of semantic concepts/categories and should logically not have directed cycles. Unfortunately, because they are created in a crowd-sourced way, they usually have thousands of cycles. So, the question is "can we remove the cycles while preserving the logical structure of the taxonomy as much as possible?"

The discussion in the session revolved around the correct formulation of this problem, particularly in going from "preserving the logical structure" to "maximising the transitive closure." In addition, I was asked if additional input related to number of different users creating an edge is available and I replied in negative. Also, I clarified that I am interested in good approximation solutions as well as heuristics that can deal with taxonomies that have hundreds of millions of edges.

4.2 Open Problem 2

Alejandro Lopez-Ortiz (University of Waterloo, CA)

My meta-problem is as follows:

Nowadays large data sets live in distributed NoSQL databases, often in main memory. A typical computation locks a view of the data (but not necessarily the data itself) that is at most a handful of operations old, until the transaction completes and the lock on the view is released.

This means that our current data structures and algorithms need to (1) be adapted to run in a distributed fashion (2) with as low as possible amounts of communication between nodes and (3) supporting limited, bounded persistence in the most efficient manner.

For some data structures and algorithms this can be achieved in a straightforward manner (e.g. BSTs, embarrassingly parallelizable algorithms), some others an efficient implementation requires new tools and last but not least, in some cases this might not be achievable. In this case a lower bound for the data structure/algorithm would be desirable, e.g. how expensive it is to recompute Dijkstra in a distributed, persistent setting with a small number of communication rounds.

4.3 Open Problem 3

Sebastian Wild (TU Kaiserslautern, DE)

I posed the problem to compute or approximate the expected costs of the optimal alphabetic search tree for *random weights* on the leaves.

More precisely, assume that we draw $U_1, \dots, U_{n-1}$ i. i. d. uniformly in $(0, 1)$ and denote by $D_1, \dots, D_n$ the *spacings* between the sorted numbers, i. e., D_j is the difference of the jth smallest and the $(j-1)$st smallest of the U_i, where we add $U_0 = 0$ and $U_n = 1$. The resulting vector $(D_1, \dots, D_n)$ is a stochastic vector, drawn uniformly from the closed $n-1$ dimensional simplex; it thus represents a uniformly chosen random probability distribution over the numbers $\{1, \dots, n\}$. The vector $(D_1, \dots, D_{n-1})$ is also said to have a *Dirichlet-distribution* with parameter $(1, \dots, 1)$. We now construct the optimal binary search tree with leaf weights $D_1, \dots, D_n$, and consider as cost of the tree $C = \sum_i D_i \cdot \mathrm{depth}(i\text{th leaf})$, i. e., the average leaf depth.

This problem is related to the analysis of comparison-optimal partitioning methods in Quicksort, which have been proposed by Aumüller and Dietzfelbinger [1]: The expected costs of the optimal alphabetic search tree are the leading-term coefficient of the expected number of comparisons of comparison-optimal partitioning.

The problem is also a natural information-theoretic question: How much can an alphabet with random letter access probabilities be compressed *on average* using an alphabetic prefix code (i. e., one that retains the order of symbols among code words; such codes are also known as *Hu-Tucker codes*). If we subtract from the average code word length C the (binary) entropy of $(D_1, \dots, D_n)$, we obtain the *redundancy* of the code, $R = C - \mathcal{H}(D_1, \dots, D_n) \geq 0$.

The problem is thus to determine the expected redundancy $\mathbb{E}[R]$, where the expectation is taken over the weights $D_1, \ldots, D_n$.

From the information-theoretic perspective, one could also ask for other coding schemes, such as *Huffman codes* or *Shannon codes*; results for Huffman codes yield an upper bound on Hu-Tucker codes. For Huffman codes, worst-case bounds on the redundancy are known for given symbol weights (see, e. g., [2] and the references therein). The bound depends only on the probability of the most probable symbol, but a precise computation of the expected value is still challenging.

References

1 M. Aumüller and M. Dietzfelbinger. "Optimal Partitioning for Dual-Pivot Quicksort", *ACM Transactions on Algorithms* 12:2 article 18, 2016. http://dx.doi.org/10.1145/2743020

2 Chunxuan Ye and R. W. Yeung. "A simple upper bound on the redundancy of Huffman codes", *IEEE Transactions on Information Theory* 48:7, 2132–2138, 2002. http://dx.doi.org/10.1109/TIT.2002.1013158

Participants

- Deepak Ajwani
Bell Labs – Dublin, IE
- Helmut Alt
FU Berlin, DE
- Alexandr Andoni
Columbia Univ. – New York, US
- Martin Aumüller
IT Univ. of Copenhagen, DK
- Timo Bingmann
KIT – Karlsruher Institut für Technologie, DE
- Gerth Stølting Brodal
Aarhus University, DK
- Andrej Brodnik
University of Primorska, SI
- Martin Dietzfelbinger
TU Ilmenau, DE
- Anne Driemel
TU Eindhoven, NL
- Fabian Dütsch
Universität Münster, DE
- Guy Even
Tel Aviv University, IL
- Rolf Fagerberg
University of Southern Denmark – Odense, DK
- Martin Farach-Colton
Rutgers Univ. – Piscataway, US
- Simon Gog
KIT – Karlsruher Institut für Technologie, DE
- Mordecai Golin
HKUST – Kowloon, HK
- Goetz Graefe
HP Labs – Madison, US
- Torben Hagerup
Universität Augsburg, DE
- Herman J. Haverkort
TU Eindhoven, NL
- John Iacono
New York University, US
- Riko Jacob
IT Univ. of Copenhagen, DK
- Tsvi Kopelowitz
University of Michigan – Ann Arbor, US
- Moshe Lewenstein
Bar-Ilan University – Ramat Gan, IL
- Alejandro Lopez-Ortiz
University of Waterloo, CA
- Jérémie Lumbroso
Princeton University, US
- Conrado Martinez
UPC – Barcelona, ES
- Kurt Mehlhorn
MPI für Informatik – Saarbrücken, DE
- Ulrich Carsten Meyer
Goethe-Universität – Frankfurt a. M., DE
- Friedhelm Meyer auf der Heide
Universität Paderborn, DE
- Ian Munro
University of Waterloo, CA
- Markus E. Nebel
TU Kaiserslautern, DE
- Elisabeth Neumann
TU Kaiserslautern, DE
- John D. Owens
Univ. of California, Davis, US
- Manuel Penschuck
Goethe-Universität – Frankfurt a. M., DE
- Seth Pettie
University of Michigan – Ann Arbor, US
- Rajeev Raman
University of Leicester, GB
- Alejandro Salinger
SAP SE – Walldorf, DE
- Peter Sanders
KIT – Karlsruher Institut für Technologie, DE
- Robert Sedgewick
Princeton University, US
- Francesco Silvestri
IT Univ. of Copenhagen, DK
- He Sun
University of Bristol, GB
- Jan Vahrenhold
Universität Münster, DE
- Sebastian Wild
TU Kaiserslautern, DE