06091 Abstracts Collection
Data Structures
— Dagstuhl Seminar —

Lars Arge[1], Robert Sedgewick[2] and Dorothea Wagner[3]

[1] BRICS - Aarhus, DK
large@daimi.au.dk
[2] Princeton Univ., US
rs@cs.princeton.edu
[3] Univ. Karlsruhe, DE
dwagner@ira.uka.de

Abstract. From 26.02.06 to 03.03.06, the Dagstuhl Seminar 06091 "Data Structures" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Algorithms, data structures

06091 Executive Summary – Data Structures

The Dagstuhl Seminar on Data Structures in 2006 reported on ongoing research on data structures, including randomized, cache-oblivious and succinct data structures. There was some shift of interest away from purely theoretical issues towards scientific studies that are directly relevant to practical applications. The participants were asked to think about the direction that research on data structure should take. Several presentations were provocative responses to this question. Interest in the topic remains high: another attendance record was set.

Keywords: Algorithms, data structures

Joint work of: Arge, Lars; Sedgewick, Robert; Wagner, Dorothea

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2006/841

I/O-Efficient Batched Union-Find and Its Applications to Terrain Analysis

Pankaj Kumar Agarwal (Duke University, USA)

Despite extensive study over the last four decades and numerous applications, no I/O-efficient algorithm is known for the union-find problem. In this paper we present an I/O-efficient algorithm for the batched (off-line) version of the union-find problem. Given any sequence of N union and find operations, where each union operation joins two distinct sets, our algorithm uses $O(\text{SORT}(N)) = O(\frac{N}{B}\log_{M/B}\frac{N}{B})$ I/Os, where M is the memory size and B is the disk block size. This bound is asymptotically optimal in the worst case. If there are union operations that join a set with itself, our algorithm uses $O(\text{SORT}(N) + \text{MST}(N))$ I/Os, where $\text{MST}(N)$ is the number of I/Os needed to compute the minimum spanning tree of a graph with N edges. We also describe a simple and practical $O(\text{SORT}(N)\log(\frac{N}{M}))$-I/O algorithm for this problem, which we have implemented.

We are interested in the union-find problem because of its applications in terrain analysis. A terrain can be abstracted as a height function defined over $\mathbb{R}^2$, and many problems that deal with such functions require a union-find data structure. With the emergence of modern mapping technologies, huge amount of elevation data is being generated that is too large to fit in memory, thus I/O-efficient algorithms are needed to process this data efficiently. In this paper, we study two terrain analysis problems that benefit from a union-find data structure: (i) computing topological persistence and (ii) constructing the contour tree. We give the first $O(\text{SORT}(N))$-I/O algorithms for these two problems, assuming that the input terrain is represented as a triangular mesh with N vertices.

Finally, we report some preliminary experimental results, showing that our algorithms give order-of-magnitude improvement over previous methods on large data sets that do not fit in memory.

Keywords: Terrain modeling, data structures, I/O-efficient algorithms

Joint work of: Agarwal, Pankaj Kumar; Arge, Lars; Yi, Ke

See also: P. K. Agarwal, L. Arge, and K. Yi, I/O-Efficient Batched Union-Find and Its Applications to Terrain Analysis, Proc. 22nd ACM Symp. on Computational Geometry (SoCG), 2006.

An Adaptive Packed-Memory Array

Michael A. Bender (SUNY at Stony Brook, USA)

The packed-memory array (PMA) is a data structure that maintains a dynamic set of N elements in sorted order in a $\Theta(N)$-sized array.

The idea is to intersperse $\Theta(N)$ empty spaces or gaps among the elements so that only a small number of elements need to be shifted around on an insert or delete. Because the elements are stored physically in sorted order in memory or on disk, the PMA can be used to support extremely efficient range queries. Specifically, the cost to scan L consecutive elements is $O(1 + L/B)$ memory transfers.

This paper gives the first adaptive packed-memory array (APMA), which automatically adjusts to the input pattern. Like the original PMA, any pattern of updates costs only $O(\log^2 N)$ amortized element moves and $O(1+(\log^2 N)/B)$ amortized memory transfers per update.

However, the APMA performs even better on many common input distributions achieving only $O(\log N)$ amortized element moves and $O(1 + (\log N)/B)$ amortized memory transfers. The paper analyzes sequential inserts, where the insertions are to the front of the APMA, hammer inserts, where the insertions "hammer" on one part of the APMA, random inserts, where the insertions are after random elements in the APMA, and bulk inserts, where for constant c $(0 \leq c \leq 1)N^c$ elements are inserted after random elements in the APMA. The paper then gives simulation results that are consistent with the asymptotic bounds. For sequential insertions of roughly 1.4 million elements, the APMA has four times fewer element moves per insertion than the traditional PMA and running times that are more than seven times faster.

Keywords: Adaptive Packed-Memory Array, Cache Oblivious, Locality Preserving, Packed-Memory Array, Range Query, Rebalance, Sequential File Maintenance, Sequential Scan, Sparse Array

Joint work of: Bender, Michael A.; Hu, Haodong

See also: Michael A. Bender and Haodong Hu. An Adaptive Packed-Memory Array. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), pages 20-29, 2006.

On the power of data structures for parsing *LR* languages

Norbert Blum (Universität Bonn, D)

Usually, a parser for an $LR(k)$ grammar $G = (V, \Sigma, P, S)$ is a deterministic pushdown transducer which produces backwards a rightmost derivation for a given input string $x \in L(G)$. The best known upper bound for the size of such a parser is $O(2^{|G||\Sigma|^{k+1}})$.

More then twenty years ago, Ukkonen has given a family of $LR(0)$ grammars proving that every such a parser for these grammars has size $\geq 2^{c\sqrt{|G|}}$ for some constant $c > 0$. If we add to a parser the possibility to manipulate a directed graph of size $O(|G|n)$ where n is the length of the input we obtain an extended parser. Given an arbitrary $LR(k)$ grammar G, we show how to construct an extended parser of polynomial size.

Keywords: $LR(k)$ grammar, $LR(k)$ parser, size of parser, extended $LR(k)$ parser

Joint work of: Blum, Norbert; Leinen, Hans-Hermann

Skewed Binary Search Trees

Gerth Stølting Brodal (BRICS - Aarhus, DK)

We consider in this talk the class of balanced binary search trees, where the left subtree of a node v stores a fraction $\alpha \in [0, 1]$ of the nodes in the subtree rooted at v. Surprisingly, we show that skewed balanced search trees with $\alpha \approx 0.3$ outperform perfectly balanced search trees for uniformly distributed searches on a set of experiments involving 20 000 elements.

This observation is explained by the fact that searching left and rigth of a node can have different costs because of various CPU features, including branch prediction schemes and cache replacement strategies.

We show experiments for various layouts of skewed balanced trees.

The most efficient layout in the experiments is a variation of the van Emde Boas layout adapted to skewed balanced search trees.

Joint work of: Brodal, Gerth Stølting; Moruz, Gabriel

When RAMBO goes Cuckoo

Andrej Brodnik (University of Primorska, SLO)

A new data structure is presented that dynamically maintains a collection of n w-bit word-size integers and supports insertions, deletions, and predecessor queries. Our data structure runs on a model of computation that has an AC^0 CPU, a memory of bits, and memory control circuitry of size $w^{O(1)}$ and depth $O(\log w / \log \log w)$. Our data structure executes all three operations using $O(1)$ CPU operations and memory accesses. The runtime of our data structure is different in different computational models. In the extended-circuit-RAM, the operations on our data structure cost $\Theta(\log w / \log \log w)$ which matches the known lower bound for the simpler static dictionary problem when the memory is limited to size $2^{\mathrm{polylog}(n)}$. In the it-probe model, there is a lower bound and matching data structure that uses time $\Theta(\sqrt{\log n / \log \log n})$ for predecessor queries.

However, the lower bounds require the model to limit the size of a memory request to be $\log M \leq w$ even though the memory responds with w bits. By allowing word sized memory requests and responses in the bit probe model, the existing lower bounds for predecessor queries do not hold. Our data structure, through this slight modification of the model, surpasses the known lower bounds by executing all operations in time $O(1)$. All runtimes for insertions and deletions are amortized expected.

Keywords: Data structures, predecessor, RAMBO

Joint work of: Brodnik, Andrej; Iacono, John

Split and share: Simulating full randomness

Martin Dietzfelbinger (TU Ilmenau, D)

In many situations in which a hash function $h\colon U \to R$ or several hash functions $h_1, \ldots, h_k$ are used, one makes the "full randomness assumption": On the set S of keys that appear in the application h is fully random (or: each h_i is fully random and the h_i are independent). Previous work has shown that full randomness can be simulated at the space cost of $O(n)$ words from R, where $n = |S|$.

We describe a general technique that allows one to assume full randomness without making assumptions about the input or idealizing on the behaviour of technically weaker hash functions, using $o(n)$ space.

An old idea is to use weaker hash functions to *split* S into disjoint chunks $S_1, \ldots, S_{\sqrt{n}}$ of size $\sqrt{n}$ each, and apply the algorithm in question to each chunk separately. It is easy to provide a data structure D that provides full randomness on each chunk, using space $O(n^{3/4})$.

The new idea is to use the same structure D for all chunks ("*share*").

The approach was sketched in a paper on "blocked cuckoo hashing" at ICALP 2005 (with Ch. Weidling), but it is assumed to have other applications.
(Work in progress.)

Keywords: Randomized algorithms, hashing, Bloom filters

See also: Martin Dietzfelbinger and Christoph Weidling: Balanced Allocation and Dictionaries with Tightly Packed Constant Size Bins. In *Proc. Int. Coll. on Algorithms, Logic and Programming (ICALP 2005)*, LNCS 3580, pages 166–178, 2005.

Cache-oblivious String Dictionaries

Rolf Fagerberg (Univ. of Southern Denmark - Odense, DK)

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.

Joint work of: Brodal, Gerth; Fagerberg, Rolf

Fully-Adaptive Strong Renaming

Faith Ellen Fich (University of Toronto & Technion Haifa, CA)

Renaming allows processes to get distinct names from a small name space and release these names.

In strong renaming, the name a process gets is at most the number of processes currently occupying, getting or releasing a name. Adaptive renaming performs both GetName and ReleaseName in time bounded above by a function of this number of processes. Fully-adaptive renaming performs these operations in time bounded above by a function of the number of processes currently performing GetName and ReleaseName, but does not depend on the number of processes currently occupying names. A fully-adaptive strong renaming data structure is presented for a synchronous distributed system in which processes communicate via shared registers and counters.

The number of steps taken by a process to perform GetName or ReleaseName is polylogarithmic in the number of processes concurrently performing these operations. The implementation tolerates any number of process crashes.

Joint work of: Fich, Faith Ellen; Brodsky, Alex; Woelfel, Philipp

Die Another Day

Rudolf Fleischer (Fudan University - Shanghai, PRC)

The hydra was a many-headed monster from Greek mythology that could grow one or two new heads when one of its heads got cut off. It was the second task of Hercules to kill this monster.

In an abstract sense a hydra can be modeled by a tree where the leaves are the heads, and when a head is cut off some subtrees get duplicated. Different hydra species differ by the location of subtrees to be duplicated and by the number of new subtrees grown in each step. Using some deep mathematics, it had been shown that two classes of rather restricted hydra species must always die, independent of the order in which heads are cut off.

In this paper we provide an elementary proof which actually gives a complete classification of all hydra species as immortal or doomed.

Now, if Hercules had known this...

Keywords: Hydra, Koenig's Lemma, Peano Arithmetic

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/765

Small Stretch (α, β)-spanners on Dynamic Graphs

Paolo Franciosa (Università di Roma I, I)

We present fully dynamic algorithms for maintaining (2,1)- and (3,2)-spanners of unweighted undirected graphs.

An (α, β)-spanner S of a graph G is a subgraph of G such that for any pair of vertices x, y we have $d_S(x, y) \leq \alpha \cdot d_G(x, y) + \beta$, where $d_S(x, y)$ (resp., $d_G(x, y)$) is the distance between x and y in S (resp., in G).

We show how to maintain a (2,1)- or (3,2)-spanner under an intermixed sequence of edge insertions and edge deletions in $O(\Delta)$ amortized time per operation over a sequence of $\Omega(n)$ updates, where Δ is the maximum degree of the vertices and n is the number of vertices in the graph. The maintained (2,1)-spanner (resp., (3,2)-spanner) has $O(n^{3/2})$ edges (resp., $O(n^{4/3})$ edges), which is known to be optimal in the worst case.

All our algorithms are deterministic and are substantially faster than recomputing a spanner from scratch after each update.

These results derive from a more careful analysis of the algorithms presented in [Ausiello, Franciosa, Italiano, ESA 05].

Keywords: Graph algorithms, dynamic algorithms, graph spanners

Joint work of: Ausiello, Giorgio; Franciosa, Paolo; Italiano, Giuseppe Francesco

See also: Proc. European Symposium on Algorithms (ESA 2005), LNCS 3669, pages 532–543, 2005

Dynamic Matrix Rank

Gudmund S. Frandsen (Univ. of Aarhus, DK)

We consider maintaining information about the rank of a matrix under changes of the entries. For $n \times n$ matrices, we show an upper bound of $O(n^{1.575})$ arithmetic operations and a lower bound of $\Omega(n)$ arithmetic operations per change. The upper bound is valid when changing up to $O(n^{0.575})$ entries in a single column of the matrix. Both bounds appear to be the first non-trivial bounds for the problem. The upper bound is valid for arbitrary fields, whereas the lower bound is valid for algebraically closed fields. The upper bound uses fast rectangular matrix multiplication, and the lower bound involves further development of an earlier technique for proving lower bounds for dynamic computation of rational functions.

Joint work of: Frandsen, Gudmund S.; Frandsen, Peter F.

On Quadtrees for Point Sets

Martin Fürer (Pennsylvania State University, USA)

Quadtrees describe a given point set contained in a square as follows. If there is just one point, then the tree is trivial, otherwise the root has 4 children which are the quadtrees of the 4 smaller squares obtained by partitioning in the middle.

Compressed quadtrees are obtained from regular quadtrees by replacing branchless paths by single edges.

We use compressed quadtrees to produce spanners in (unit) disk graphs with hereditary small separators. We show how to construct compressed quadtrees in time linear plus the time for sorting.

Keywords: Compressed quadtrees, unit disk graphs, spanners

Joint work of: Fürer, Martin; Kasiviswanathan, Shiva

Design of Data Structures for Mergeable Trees

Loukas Georgiadis (Univ. of Aarhus, DK)

Motivated by an application in computational topology, we consider a novel variant of the problem of efficiently maintaining dynamic rooted trees.

This variant allows an operation that merges two tree paths. In contrast to the standard problem, in which only one tree arc at a time changes, a single merge operation can change many arcs. In spite of this, we develop a data structure that supports merges and all other standard tree operations in $O(\log^2 n)$ amortized time on an n-node forest. For the special case that occurs in the motivating application, in which arbitrary arc deletions are not allowed, we give a data structure with an $O(\log n)$ amortized time bound per operation, which is asymptotically optimal. The analysis of both algorithms is not straightforward and requires ideas not previously used in the study of dynamic trees. We explore the design space of algorithms for the problem and also consider lower bounds for it.

Keywords: Data structures, dynamic trees, heap-ordered forest, computational topology

Joint work of: Georgiadis, Loukas; Tarjan, Robert; Werneck, Renato

See also: Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 394-403, 2006.

Online and Offline Access to Short Lists

Torben Hagerup (Universität Augsburg, D)

We consider the list-update problem introduced by Sleator and Tarjan, specializing it to the case of accesses only and focusing on short lists.

We describe a new optimal offline algorithm, faster than the best previous algorithm when the number of accesses is sufficiently large relative to the number ℓ of items.

We also give a simple optimal offline algorithm for $\ell = 3$.

Taking c_ℓ to denote the best competitivity ratio of a randomized online algorithm for the list-access problem with ℓ items, we demonstrate that $c_2 = 9/8$ and $c_3 = 6/5$.

Keywords: Online algorithms, list-update problem, competitive analysis

Small maps and few probes suffice to find a path

Riko Jacob (ETH Zürich, CH)

We consider the problem of finding a short path between two vertices in an unweighted undirected graph G by accessing only a small part of the graph.

This is made possible by a preprocessing step that creates a small, condensed representation of the graph that we call a map. To gain information about G that is not stored in the map, it is possible to probe the graph, i.e., to ask for the neighbors of a particular vertex, at a cost proportional to the number of returned neighbours. Our focus is on sub-linear size maps and sub-linear probing cost, which is not achievable for arbitrary graphs. Still, for graphs with at most ε vertices of degree higher than δ, we present a map creation function $\mathcal{M}$ and a path finding algorithm such that if M is the size limit of the maps created by $\mathcal{M}$ and the probing cost is limited by C times the true distance between the two end-nodes of the path, then $M \cdot C = O(\delta n \log^2 n)$.

For the same class of graphs we give a lower bound $M \cdot C = \Omega(\delta\varepsilon \log n)$.

Joint work of: Derungs, Jörg; Jacob, Riko; Widmayer, Peter

The Density of Iterated Crossing Points and a Gap Result on Triangulations of Finite Point Sets

Rolf Klein (Universität Bonn, D)

Let S denote a finite point set in the plane that is not contained in the vertex set of a dilation-free triangulation. We show that iterately drawing the complete graph on S, and adding its crossing points to S, results in an infinite point set that lies dense in some part of the plane. We conclude that there exists a threshold $\eta > 1$ such that each triangulation containing S in its vertex set is of dilation at least η.

Keywords: Dilation, geometric networks, plane graph, spanning ratio, stretch factor, triangulation

Joint work of: Klein, Rolf; Kutz, Martin

See also: Proc. ACM Symp. on Computational Geometry (SoCG), 2006

New(ish) Research Directions in Data Structures and Algorithms Research

Alejandro Lopez-Ortiz (University of Waterloo, CA)

We present several new and upcomings models of computation that are of practical relevance. These are models which are under various degrees of incipient research, yet given their relevance it is safe to say that they deserve even more attention from the algorithms and data structures community.

The first model introduced is the new world of "small parallelism" consisting of a small number (i.e. at most $\log n$) of processing units. These units are usually pipelines or multi-core processors in modern CPU architectures. We suggest the study of thread level parallelism with a small degree of parallelism. We also considered the setting in distributed computation in which a large number of processors collaborate towards a solution (in PRAM style) but the cost of communication is much greater, which makes some of the shared memory PRAM algorithms unfeasible. The next model considered is derived from the large volumes of data being generated by embedded systems (e.g. purchasing data from a chain of grocery stores, sensor networks, scientific measurement equipment). In this case the volume of data is such that at most one (or a constant) number of passes can be done on the data. This is the actvie field of data streams research, which we predict it will become even more active. A similar problem is sample computing and sublinear computations as introduced by Chazelle et al. The last considerations of large sets is lossy compression, in which an external agent gives an acceptable degradation function and we wish to compress the data in a way that the degradation function is minimized. The remaining two computing models are Monte-Carlo/Las Vegas like solutions to non-computable problems (yes/no/don't know answers). This is particularly important in the case of verification of programs and code obfuscation, which are needed in practice and at least partially achievable, but generally known to be uncomputable.

Keywords: Low-degree thread-level parallelism, data streams, code obfuscation

Adaptive Sampling for Selection

Conrado Martinez (TU of Catalonia - Barcelona, E)

Using samples to select the pivots in Hoare's Find algorithm (a.k.a. quickselect) yields substantial improvements in the algorithm's expected performance. It reduces the variance as well. We study here proportion-from-s, a "natural" sampling strategy where the pivot of each recursive stage is an element in a sample of s elements, whose relative rank within the sample is the same as the relative rank of the element we are looking for. Wefind analytic solutions for proportion-from-s with $s = 2$ and $s = 3$ and we are also able to compute the variance of median-of-three, which was not previously known. Besides our results

for small samples, we can use our general results to analyze the case $s \to \infty$. It turns out that using variable-size samples and proportion-from-s sampling we can obtain optimal expected performance (i.e., the average number of comparisons is then $n + \min(m, n - m)$ + lower order terms, where m is the rank of the sought element; we also show that the variance is subquadratic, implying that large deviations from the mean performance are highly unlikely. Finally, we show that the optimal sample size is $\Theta(\sqrt{n})$.

Keywords: Selection, sampling, quickselect

Joint work of: Martinez, Conrado; Panario, Daniel; Viola, Alfredo; Daligault, Jean

Controlled Perturbation for Reliable and Efficient Geometric Computing

Kurt Mehlhorn (MPI für Informatik - Saarbrücken, D)

Most algorithms of computational geometry are designed for the Real-RAM and non-degenerate inputs input. We call such algorithms idealistic. Executing an idealistic algorithm with floating point arithmetic may fail. Controlled perturbation replaces an input x by a random nearby $\tilde{x}$ in the δ-neighborhood of x and then runs the floating point version of the idealistic algorithm on $\tilde{x}$. The hope is that this will produce the correct result for $\tilde{x}$ with constant probability provided that δ is small and the precision L of the floating point system is large enough. We turn this hope into a theorem for a large class of geometric algorithms and describe a general methodology for deriving a relation between δ and L. We exemplify the usefulness of the methodology by examples.

Full Paper:
http://www.mpi-inf.mpg.de/~mehlhorn/ftp/ControlledPerturbationGeneralStrategy.pdf

Adaptive Searching in Succinctly Encoded Binary Relations and Tree-Structured Documents

Ian Munro (University of Waterloo, CA)

The most heavily used methods to answer conjunctive queries on binary relations (such as the one associating keywords with web pages) are based on inverted lists stored in sorted arrays and use variants of binary search. We show that a succinct representation of the binary relation permits much better results, while using space within a lower order term of the optimal. We apply our results not only to conjunctive queries on binary relations, but also to queries on semi-structured documents such as XML documents or file-system indexes, using a variant of an adaptive algorithm used to solve conjunctive queries on binary relations.

Keywords: Conjunctive queries, intersection problem, succinctdata structures, labeled trees, multi-labeled trees

Joint work of: Munro, Ian; Barbay, Jeremy; Golynski, Alex; Rao, Srinivasa

An Infinite Number of Open Problems Related to Data Structures

Rasmus Pagh (IT University of Copenhagen, DK)

Classical data structures are asymmetric in the sense that the data stored is much larger than the amount of data involved in a query. We argue, and give examples pointing out, that data structures techniques can also be useful in symmetric settings in which we may preprocess parts of data. Our main example is computation of the intersection of two sets (joint work with Anna Pagh). We give examples of the endless number of questions of this form that could be addressed.

Keywords: Preprocessing, sets

Towards a Final Analysis of Pairing Heaps

Seth Pettie (MPI für Informatik - Saarbrücken, D)

Fredman, Sedgewick, Sleator, and Tarjan proposed the pairing heap as a self-adjusting, streamlined version of the Fibonacci heap. It provably supports all priority queue operations in logarithmic time and is known to be extremely efficient in practice.

However, despite its simplicity and empirical superiority, the pairing heap is one of the few popular data structures whose basic complexity remains open. In this paper we prove that pairing heaps support the deletemin operation in optimal logarithmic time and all other operations (insert, meld, and decreasekey) in time $O(2^{2\sqrt{\log\log n}})$. This result gives the *first* sub-logarithmic time bound for decreasekey and comes close to the lower bound of $\Omega(\log\log n)$ established by Fredman.

Pairing heaps have a well known but poorly understood relationship to splay trees and, to date, the transfer of ideas has flowed in one direction: from splaying to pairing. One contribution of this paper is a new analysis that reasons explicitly with information-theoretic measures. Whether these ideas could contribute to the analysis of splay trees is an open question.

Keywords: Data structure, heap, self-adjusting, amortized analysis

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/764

Full Paper:
http://www.mpi-inf.mpg.de/∼pettie/papers/focs05.pdf

See also: Proceedings 46th Annual Symposium on Foundations of Computer Science (FOCS), pp. 174–183, 2005

Engineering the LOUDS succinct tree representation

Rajeev Raman (University of Leicester, GB)

Ordinal trees are arbitrary rooted trees where the children of each node are ordered. We consider *succinct*, or highly space-efficient, representations of (static) ordinal trees with n nodes.

There are a number of representations that use $2n + o(n)$ bits of space to represent ordinal trees; each representation supports a different set of tree operations in $O(1)$ time on the RAM computation model.

In this paper we focus on the practical performance of a fundamental representation: the Level-Order Unary Degree Sequence (LOUDS) representation [Jacobson, *Proc. 30th FOCS*, 549–554, 1989].

Due to its conceptual simplicity, LOUDS would appear to be a good practical candidate for succinct tree representations.

A complementary succinct representation of a tree as a balanced parentheses sequ ence [Munro and Raman, *SIAM J. Comput.* **31** (2001), 762–776; Jacobson, *op. cit.*], as simplified and engineered by Geary et al. [*Proc. 15th CPM Symp.*, LNCS 3109, pp. 159–172, 2004]. In essence, the two representations have only the basic navigational operations in common (parent, first_child, last_child, previous_sibling, next_sibling).

Unfortunately, a direct implementation of LOUDS is not competitive with the above parenthesis implementation. We propose several variants of LOUDS, of which one in particular, called *LOUDS++*, is competitive with the parenthesis representation on the common set of operations.

Our primary motivation is the succinct representation of large static XML documents which are accessed through the DOM (Document Object Model) interface, and our tests involve traversing these documents in various canonical orders.

Keywords: Succinct data structures, data compression, algorithm engineering, semistructured data.

Joint work of: Delpratt, O'Neil; Rahman, Naila; Raman, Rajeev

Approximate Shortest Path Queries on Weighted Polyhedral Surfaces

Jörg-Rüdiger Sack (Carleton University - Ottawa, CA)

We consider the classical geometric problem of determining shortest paths between pairs of points lying on a weighted polyhedral surface P consisting of n triangular faces. We present query algorithms that compute approximate distances and/or approximate (weighted) shortest paths. Our algorithm takes as input an approximation parameter $\varepsilon \in (0, 1)$ and a query time parameter q and builds a data structure $\mathrm{APQ}(P, \varepsilon; q)$ which is then used for answering ε-approximate

distance queries in $O(q)$ time. This algorithm is source point independent and improves significantly on the previous solution.

For the case where one of the query points is fixed we build a data structure $\mathrm{SSQ}(P, \varepsilon; a)$ that can answer ε-approximate distance queries from a to any query point in P in $O(\log \frac{1}{\varepsilon})$ time. This is an improvement upon the previously known solution for the Euclidean fixed source query problem. Our algorithm also generalizes the setting from previously studied unweighted polyhedra to weighted polyhedral surfaces of arbitrary genus.

Our shortest path algorithms are based on a novel separator algorithm which we introduce here and which extends and generalizes previously known separator algorithms.

Joint work of: Aleksandrov, Lyudmil; Djidjev, Hristo N.; Guo, Hua ; Maheshwari, Anil ; Nussbaum, Doron; Sack, Jörg-Rüdiger

Algorithm Engineering - An Attempt at a Definition

Peter Sanders (Universität Karlsruhe, D)

We explain our view of algorithm engineering as a wider view of algorithmics that encompasses models, design, analysis, implemenation of algorithms, experiments, algorithm libraries, and benchmarks.

These activities form a tightly interacting methodology that encompasses but is more powerful than classical algorithm theory.

Geometric spanners with few edges and degree five

Michiel Smid (Carleton Univ. - Ottawa, CA)

An $O(n \log n)$–time algorithm is presented that, when given a set S of n points in $\mathbb{R}^d$ and an integer k with $0 \leq k \leq n$, computes a graph with vertex set S, that contains at most $n - 1 + k$ edges, has stretch factor $O(n/(k + 1))$, and whose degree is at most five. This generalizes a recent result of Aronov *et al.*, who obtained this result for two-dimensional point sets.

Keywords: Computational geometry, spanners, minimum spanning trees

Robust Ceometric Computation Through Digital Picture Approximation

Kokichi Sugihara (University of Tokyo, J)

We combine the topology-oriented method with the digital picture technique, and thus propose an easier version of the topology-oriented method.

The topology-oriented method is an approach to robust geometric computation, in which we place the highest priority on keeping the topological consistency, and use numerical values only when they are consistent with the topological structure. This method has many good properties, but is not necessarily easy to use because we need deep insight to the individual problems in order to extract purely topological properties on which our algorithm relies. We overcome this shortcoming by the digital picture technique. That is, we first solve the problem in a digital picture approximately, then extract the topological structure from the digital picture, and finally use it in the topology-oriented method. The validity of the proposed method is shown in the context of disk packing.

Keywords: Topology-oriented approach, numerical error, digital topology, disk packing

Joint work of: Sugihara, Kokichi; Matuura, Shiro

In-Place Randomized Slope-Selection

Jan Vahrenhold (Universität Münster, D)

Slope selection is a well-known algorithmic tool used in the context of computing robust estimators for fitting a line to a collection $\mathcal{P}$ of n points in the plane. We demonstrate that it is possible to perform slope selection in expected $\mathcal{O}(n \log n)$ time using only constant extra space in addition to the space needed for representing the input.

Our solution is based upon a space-efficient variant of Matoušek's randomized interpolation search, and we believe that the techniques developed in this paper will prove helpful in the design of space-efficient randomized algorithms using samples. To underline this, we also sketch how to compute the repeated median line estimator in an in-place setting.

Keywords: In-Place Algorithms, Slope Selection

Joint work of: Blunck, Henrik; Vahrenhold, Jan

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/839

Simulated Annealing Beats Metropolis in Combinatorial Optimization

Ingo Wegener (Universität Dortmund, D)

The Metropolis algorithm is simulated annealing with a fixed temperature.

Surprisingly enough, many problems cannot be solved more efficiently by simulated annealing than by the Metropolis algorithm with the best temperature.

The problem of finding a natural example (artificial examples are known) where simulated annealing outperforms the Metropolis algorithm for all temperatures has been discussed by Jerrum and Sinclair (1996) as "an outstanding open problem." This problem is solved here. The examples are instances of the well- known minimum spanning tree problem. Moreover, it is investigated which instances of the minimum spanning tree problem can be solved efficiently by simulated annealing. This is motivated by the aim to develop further methods to analyze the simulated annealing process.

Cache-Oblivious Planar Orthogonal Range Searching Simplified

Norbert Zeh (Dalhousie University, CA)

We present a simple static cache-oblivious structure for planar 2-sided range searching. Our structure uses linear space and supports queries using $O(\log_B N + T/B)$ memory transfers. Using this structure as a building block, we obtain two additional structures. The first one is a static cache-oblivious structure that supports planar 3-sided range queries in $O(\log_B N + T/B)$ memory transfers. The structure uses $O(N \log N)$ space, thereby matching the space and query bounds of the best previous structure for this problem by Arge et al.; but our structure is much simpler. The second structure is a semi-dynamic linear-space structure for 2-sided range queries. This structure supports insertions in $O(\log_2 N \cdot \log_B N)$ memory transfers and queries in $O(\log_2 N + T/B)$ memory transfers.

Keywords: Data structures, planar range searching, cache-obliviousness, memory hierarchies

Joint work of: Arge, Lars; Zeh, Norbert

See also: Lars Arge and Norbert Zeh. Simple and Semi-Dynamic Structures for Cache-Oblivious Planar Orthogonal Range Searching. In Proceedings of the 22nd ACM Symposium on Computational Geometry, 2006.