Algorithm Day in Copenhagen

Torben Hagerup and Jyrki Katajainen (Editors)

16 September 1997

This document contains the program of the Algorithm Day held at DIKU in Copenhagen on 16 September 1997 as well as abstracts of the presentations and email addresses of the participants.
## Final Program

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
<th>Speaker/Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:15–9:20</td>
<td>Opening</td>
<td></td>
</tr>
<tr>
<td>9:20–10:00</td>
<td>Dynamic Algorithms for Graphs of Bounded Treewidth</td>
<td>Torben Hagerup</td>
</tr>
<tr>
<td>10:00–10:30</td>
<td>Managing Large Scale Computational Markets</td>
<td>Arne Andersson</td>
</tr>
<tr>
<td>10:30–11:00</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>11:00–11:30</td>
<td>Top-Down Not-Up Heapsort</td>
<td>Tomi Pasanen</td>
</tr>
<tr>
<td>11:30–12:00</td>
<td>Improving Katajainen’s Ultimate Heapsort</td>
<td>Laurent Rosaz</td>
</tr>
<tr>
<td>12:00–12:30</td>
<td>Computing Minimal Structures on Geometric Inputs</td>
<td>Christos Levcopoulos</td>
</tr>
<tr>
<td>12:30–14:00</td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>14:00–14:30</td>
<td>Optimal Broadcasting in Tree-Like Networks</td>
<td>Andrzej Lingas</td>
</tr>
<tr>
<td>14:30–15:00</td>
<td>Undirected Single Source Shortest Paths in Linear Time</td>
<td>Mikkel Thorup</td>
</tr>
<tr>
<td>15:00–15:30</td>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>15:30–16:00</td>
<td>Worst-Case Efficient External-Memory Priority Queues</td>
<td>Jyrki Katajainen</td>
</tr>
<tr>
<td>16:00–16:30</td>
<td>Error Correcting Codes, Perfect Hashing Circuits, and Deterministic Dynamic Dictionaries</td>
<td>Peter Bro Miltersen</td>
</tr>
<tr>
<td>16:30–16:40</td>
<td>Closing</td>
<td></td>
</tr>
<tr>
<td>18:00–</td>
<td>Dinner</td>
<td>Baron von Dy, Frederiksborggade 5</td>
</tr>
</tbody>
</table>
Dynamic Algorithms for Graphs of Bounded Treewidth

Torben Hagerup

Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen East
Denmark

Abstract

The formalism of monadic second-order (MS) logic has been very successful in unifying a large number of algorithms for graphs of bounded treewidth. We extend the elegant framework of MS logic from static problems to dynamic problems, in which queries about MS properties of a graph of bounded treewidth are interspersed with updates of vertex and edge labels. This allows us to unify and occasionally strengthen a number of scattered previous results obtained in an ad-hoc manner and to enable solutions to a wide range of additional problems to be derived automatically.

As an auxiliary result of independent interest, we dynamize a data structure of Chazelle and Alon and Schieber for answering queries about sums of labels along paths in a tree with edges labeled by elements of a semigroup.

Related Literature


Managing Large Scale Computational Markets

Arne Andersson

Department of Computer Science and Numerical Analysis
Lund University
Box 118, S–221 00 Lund
Sweden

Fredrik Ygge

EnerSearch and
Department of Computer Science (IDE)
University of Karlskrona/Ronneby
S–372 25 Ronneby
Sweden

Abstract

The presentation aims at illustrating the use of algorithmic thinking in applied distributed computing. We are taking part in a project on distributed load management for the electric power industry. The ISES project (Information/Society/Energy/System) is an international project, sponsored by ABB Networks Partner AB, Electricité de France, IBM Utility & Energy Services Industry, IT Blekinge, Preussen Electra, and Sydkraft AB.

We have developed a new algorithm for distributed resource allocation and for finding equilibrium prices in a distributed computational market. Our algorithm, CoTree, is well suited for large distributed systems. It is communication sparse, it adapts fast to changes in agent’s objective functions, and it is easy to implement.
Top-Down Not-Up Heapsort\textsuperscript{1}

Jyrki Katajainen
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK–2100 Copenhagen East
Denmark

Jukka Teuhola
Department of Computer Science
University of Turku
Lemminkäisenkatu 14A
FIN–20520 Turku
Finland

Tomi Pasanen
Turku Centre for Computer Science
Lemminkäisenkatu 14A
FIN–20520 Turku
Finland

Abstract

Assume that we are given $n$ elements each consisting of a key and some information associated with this key. In the worst case, the original Heapsort requires $2n \log_2 n + \Theta(n)$ key comparisons and $n \log_2 n + \Theta(n)$ element moves when sorting these $n$ elements. Moreover, the sorting is carried out in-place, i.e., only a constant amount of extra storage is utilized during the sorting process. Recently, several in-place and non-in-place variants of Heapsort have been introduced. The main ambition has been to devise a variant of Heapsort requiring only $n \log_2 n + \Theta(n)$ key comparisons and element moves.

If more than a constant amount of storage is used or only the average case is considered, we must admit that the problem has been in principle solved. In this work, we present the first variant of Heapsort—called Top-Down Not-Up Heapsort—that sorts $n$ elements in-place by performing only $n \log_2 n + \Theta(n)$ key comparisons and element moves.

\textsuperscript{1}The results of this work were presented in preliminary form in [13].
key comparisons and element moves in the worst case. The best previous variant requires $n \log_2 n + n \log^* n + \Theta(n)$ key comparisons in the worst case. However, due to the constant in the linear term $n$ must be astronomical before the new variant actually beats the old one.

Related Literature


16. I. Wegener, The worst case complexity of McDiarmid and Reed’s variant of BOTTOM-UP HEAPSORT is less than $n \log n + 1.1n$, *Information and Computation* 97 (1992) 86–96.

17. I. Wegener, BOTTOM-UP-HEAPSORT, a new variant of HEAPSORT beating, on an average, QUICKSORT (if $n$ is not very small), *Theoretical Computer Science* 118 (1993) 81–98.


Improving Katajainen’s Ultimate Heapsort

Laurent Rosaz
Laboratoire de Recherche en Informatique
Batiment 490
Université Paris-Sud
F-91405 Orsay Cedex
France

Abstract
In [9], Jyrki Katajainen gave an in-place heapsort version with a worst-case complexity in numbers of key comparisons less than $n \log_2 n + C n + o(n)$ and in numbers of element moves less than $n \log_2 n + M n + o(n)$, where $C = 32$ and $M = 62$. In this paper, I improve his algorithm and obtain complexities of the same kind, but with $C = M = 8$.

Related Literature


14. I. Wegener, The worst case complexity of McDiarmid and Reed’s variant of BOTTOM-UP HEAPSORT is less than n\log n + 1.1n, Information and Computation 97 (1992) 86–96.

15. I. Wegener, BOTTOM-UP-HEAPSORT, a new variant of HEAPSORT beating, on an average, QUICKSORT (if n is not very small), Theoretical Computer Science 118 (1993) 81–98.


Abstract

We survey some recent results concerning computation of minimal structures on geometric inputs. These include, for example:

- The complete linkage clustering of \( n \) points in the plane can be computed in \( O(n \log^2 n) \) time and linear space. If the points lie in \( \mathbb{R}^d \), the complete linkage clustering can be computed in optimal \( O(n \log n) \) time, under the \( L_1 \) and \( L_{\infty} \)-metrics. We also design efficient algorithms for approximating the complete linkage clustering.

- A minimum spanning tree of \( n \) points in \( \mathbb{R}^d \) can be obtained in optimal \( O(T_d(n, m)) \) time, where \( T_d(n, m) \) denotes the time to find a closest bicromatic pair between \( n \) red points and \( m \) blue points.

- The greedy triangulation of \( n \) points in the plane has length at most \( O(\sqrt{n}) \) times that of a minimum weight triangulation, and can be computed in linear time, given the Delaunay triangulation.

- A triangulation of length at most a constant times that of a minimum weight triangulation can be computed in polynomial time (in fact, \( O(n \log n) \) time suffices). If the points are corners of their convex hull, we show that linear time suffices to find a triangulation of length at most \( 1 + \epsilon \) times that of a minimum weight triangulation, where \( \epsilon \) is an arbitrarily small positive constant.

- We can compute a rectangular covering of any hole-free polygon in optimal time, i.e. linear with respect to the minimum rectangular covering plus the number of vertices of the polygon. Analogous results hold for covering hole-free polygons with squares.
Related Literature


Optimal Broadcasting in Tree-Like Networks

Anders Dessmark
Department of Computer Science
Lund University
Box 118, S-22100 Lund
Sweden

Krzysztof Diks
Instytut Informatyki
Uniwersytet Warszawski
Banacha 2
PL-02-097 Warszawa
Poland

Andrzej Lingas
Department of Computer Science
Lund University
Box 118, S-22100 Lund
Sweden

Hans Olsson
Department of Computer Science
Lund University
Box 118, S-22100 Lund
Sweden

Andrzej Pelc
Département d’Informatique
Université du Québec à Hull
Hull, Québec J8X 3X7
Canada

Abstract
We consider message broadcasting in networks that have tree-like topology. The source node of the input network has $k$ messages which have to be
broadcasted to all nodes of the network. In every time unit each node can
send one of already obtained messages to one of its neighbors. A broad-
casting scheme prescribes in which time unit a given node should send a
given message to which neighbor. It is minimum if it achieves the smallest
possible time for broadcasting the messages from the source to all nodes.
We give an algorithm to construct an optimal broadcasting scheme for an
arbitrary n-node tree. The time complexity of our algorithm is \( O(nk) \), i.e.,
the best possible. We also give the following algorithms to construct a mini-
mum single-message broadcasting scheme for different types of weakly cyclic
networks:

A linear-time algorithm for networks whose cycles are node-disjoint and in
which any simple path intersects at most \( O(1) \) cycles.

An \( O(n \log n) \)-time algorithm for networks whose cycles are edge-disjoint
and in which a node can belong to at most \( O(1) \) cycles.

An \( O(n^k \log n) \)-time algorithm for networks whose each edge-biconnected
component is convertible to a tree by removal of at most \( k \) edges.

Finally, we present an \( O(n^{4k+5}) \)-time algorithm for constructing a minimum
single-message broadcasting scheme for partial \( k \)-trees.

**Related Literature**

1. S. Arnborg, A. Proskurowski, Linear time algorithms for NP-hard
    problems on graphs embedded in \( k \)-trees. Discrete Applied Mathe-

2. A. Bar-Noy and S. Kipnis, Designing broadcasting algorithms in the
    postal model for message passing systems, Proc. 5th Ann. ACM
    Symp. on Par. Alg. and Arch. (1992), 11-22.

3. A. Bar-Noy and S. Kipnis, Broadcasting multiple messages in simulta-

4. D.P. Bertsekas and J.N. Tsitsiklis, Parallel and Distributed Compu-

5. H. Bodlaender. A linear time algorithm for finding tree-decomposi-


Undirected Single Source Shortest Paths in Linear Time

Mikkel Thorup
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK–2100 Copenhagen East
Denmark

Abstract
The single source shortest paths problem (SSSP) is one of the classic problems in algorithmic graph theory: given a weighted graph $G$ with a source vertex $s$, find the shortest path from $s$ to all other vertices in the graph. Since 1959 all theoretical developments in SSSP have been based on Dijkstra’s algorithm, visiting the vertices in order of increasing distance from $s$. Thus, any implementation of Dijkstra’s algorithm sorts the vertices according to their distances from $s$. However, we do not know how to sort in linear time.

Here, a deterministic linear time and linear space algorithm is presented for the undirected single source shortest paths problem with integer weights. The algorithm avoids the sorting bottleneck by building a hierarchical bucketing structure, identifying vertex pairs that may be visited in any order.

Related Literature


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 such that in every disjoint interval of $B$ consecutive priority-queue operations at most $c \log_{M/B} \frac{N}{M}$ I/Os are performed, for some positive constant $c$. These I/Os are divided evenly among the operations: if $B \geq c \log_{M/B} \frac{N}{M}$, one I/O is necessary for every $B/(c \log_{M/B} \frac{N}{M})$th operation and if $B < c \log_{M/B} \frac{N}{M}$, $\frac{c}{2} \log_{M/B} \frac{N}{M}$ 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 $O(\sum_{i=1}^{S} \frac{1}{N_i} \log_{M/B} \frac{N_i}{M})$ I/Os, where $N_i$ denotes the number of elements stored in the data structure prior to the $i$th operation, without giving any guarantee for the performance of the individual operations.
Related Literature


Australasian Theory Symposium, Australian Computer Science Communications, Volume 19, Number 2, pages 72–78, 1997.


Error Correcting Codes, Perfect Hashing Circuits, and Deterministic Dynamic Dictionaries

Peter Bro Miltersen
BRICS
University of Aarhus
Ny Munkegade, Building 540
DK–8000 Århus C
Denmark

Abstract

We consider dictionaries of size \( n \) over the finite universe \( U = \{0, 1\}^w \) and introduce a new technique for their implementation: error correcting codes. The use of such codes makes it possible to replace the use of strong forms of hashing, such as universal hashing, with much weaker forms, such as clustering.

We use our approach to construct, for any \( \epsilon > 0 \), a deterministic solution to the dynamic dictionary problem using linear space, with worst case time \( O(n^\epsilon) \) for insertions and deletions, and worst case time \( O(1) \) for lookups. This is the first deterministic solution to the dynamic dictionary problem with linear space, constant query time, and non-trivial update time. In particular, we get a solution to the static dictionary problem with \( O(n) \) space, worst case query time \( O(1) \), and deterministic initialization time \( O(n^{1+\epsilon}) \). The best previous deterministic initialization time for such dictionaries, due to Andersson, is \( O(n^{2+\epsilon}) \).

The model of computation for these bounds is a unit cost RAM with word size \( w \) (i.e. matching the universe), and a standard instruction set. The constants in the big-O’s are independent upon \( w \). The solutions are weakly non-uniform in \( w \), i.e. the code of the algorithm contains word sized constants, depending on \( w \), which must be computed at compile-time, rather than at run-time, for the stated run-time bounds to hold.

An ingredient of our proofs, which may be interesting in its own right, is the following observation: A good error correcting code for a bit vector fitting into a word can be computed in \( O(1) \) time on a RAM with unit cost multiplication.

As another application of our technique in a different model of computation, we give a new construction of perfect hashing circuits, improving a construc-
tion by Goldreich and Wigderson. In particular, we show that for any set 
$S \subseteq \{0, 1\}^w$ of size $n$, there is a Boolean circuit $C$ of size $O(w \log w)$ with $w$ 
inputs and $2 \log n$ outputs so that the function defined by $C$ is 1-1 on $S$. The 
best previous bound on the size of such a circuit was $O(w \log w \log \log w)$.

Related Literature

1. A. Andersson. Faster deterministic sorting and searching in linear 
space In 37th IEEE Symposium on Foundations of Computer Science, 

2. A. Andersson, P.B. Miltersen, S. Riis, and M. Thorup. Static dic-
tionaries on $\text{AC}^0$ RAMs: Query time $\Theta(\sqrt{\log n/ \log \log n})$ is necessary 
and sufficient. In 36th IEEE Symposium on Foundations of Computer 

3. A.M. Ben-Amram and Z. Galil. When can we sort in $o(n \log n)$ time? 
In Proc. 34th IEEE Symposium on Foundations of Computer Science, 

4. J. Bierbrauer, I. Johansson, G. Kabatianskii, and B. Smeets. On fam-


6. M. Dietzfelbinger and F. Meyer auf der Heide, Dynamic hashing in real 
time, in: J. Buchmann, H. Ganzinger, W. J. Paul (Eds.): Informatik - 
Festschrift zum 60. Geburtstag von Günter Hotz, Teubner-Texte zur 
Informatik, Band 1, B. G. Teubner, 1992, pp. 95–119. (A prelimi-
nary version appeared under the title “A New Universal Class of Hash 
Functions and Dynamic Hashing in Real Time” in ICALP’90.)

7. M. Dietzfelbinger, J. Gil, Y. Matias and N. Pippenger. Polynomial 
hash functions are reliable. In Proc. 19th Int'l. Colloq. on Automata, 
Languages and Programming, Lecture Notes in Computer Science Vol. 

8. M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A 
reliable randomized algorithm for the closest-pair problem. Technical 
Report 513, Fachbereich Informatik, Universität Dortmund, Dort-
mund, Germany, 1993.


List of Participants

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stephen Alstrup</td>
<td><a href="mailto:stephen@diku.dk">stephen@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Arne Andersson</td>
<td><a href="mailto:Arne.Andersson@dna.lth.se">Arne.Andersson@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
<tr>
<td>John Anker Corneliusen</td>
<td><a href="mailto:anker@diku.dk">anker@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Anders Dessmark</td>
<td><a href="mailto:andersd@dna.lth.se">andersd@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
<tr>
<td>Joakim Gudmundsson</td>
<td><a href="mailto:Joakim.Gudmundsson@dna.lth.se">Joakim.Gudmundsson@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
<tr>
<td>Torben Hagerup</td>
<td><a href="mailto:torben@mpi-sb.mpg.de">torben@mpi-sb.mpg.de</a></td>
<td>Department of Computer Science, University of Copenhagen and Max-Planck-Institut, Saarbrücken</td>
</tr>
<tr>
<td>Jacob Holm</td>
<td><a href="mailto:samson@diku.dk">samson@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Jyrki Katajainen</td>
<td><a href="mailto:jyrki@diku.dk">jyrki@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>László B. Kovács</td>
<td><a href="mailto:klbzyx@diku.dk">klbzyx@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Jesper Larsson</td>
<td><a href="mailto:Jesper.Larsson@dna.lth.se">Jesper.Larsson@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
<tr>
<td>Christos Levcopoulos</td>
<td><a href="mailto:Christos.Levcopoulos@dna.lth.se">Christos.Levcopoulos@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
<tr>
<td>Andrzej Lingas</td>
<td><a href="mailto:Andrzej.Lingas@dna.lth.se">Andrzej.Lingas@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
<tr>
<td>Peter Bro Miltersen</td>
<td><a href="mailto:bromille@brics.dk">bromille@brics.dk</a></td>
<td>BRICS, University of Aarhus</td>
</tr>
<tr>
<td>Tomi Pasanen</td>
<td><a href="mailto:tpasanen@cs.utu.fi">tpasanen@cs.utu.fi</a></td>
<td>Turku Centre for Computer Science</td>
</tr>
<tr>
<td>David Pisinger</td>
<td><a href="mailto:pisinger@diku.dk">pisinger@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Peter Riber</td>
<td><a href="mailto:riber@diku.dk">riber@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Laurent Rosaz</td>
<td><a href="mailto:Laurent.Rosaz@lri.fr">Laurent.Rosaz@lri.fr</a></td>
<td>Département d’Informatique, Université Paris Sud</td>
</tr>
<tr>
<td>Kurt Swanson</td>
<td><a href="mailto:kurt@dna.lth.se">kurt@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
<tr>
<td>Name</td>
<td>Email</td>
<td>Affiliation</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Mikkel Thorup</td>
<td><a href="mailto:thorup@diku.dk">thorup@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Pawel Winter</td>
<td><a href="mailto:pawel@diku.dk">pawel@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Martin Zachariasen</td>
<td><a href="mailto:martinz@diku.dk">martinz@diku.dk</a></td>
<td>Department of Computer Science, University of Copenhagen</td>
</tr>
<tr>
<td>Anna Östlin</td>
<td><a href="mailto:Anna.Ostlin@dna.lth.se">Anna.Ostlin@dna.lth.se</a></td>
<td>Department of Computer Science, University of Lund</td>
</tr>
</tbody>
</table>