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

9:15– 9:20	Jyrki Katajainen	Opening
9:20–10:00	Torben Hagerup	Dynamic Algorithms for Graphs of Bounded Treewidth
10:00–10:30	Arne Andersson	Managing Large Scale Computational Markets
10:30–11:00	Break	
11:00–11:30	Tomi Pasanen	Top-Down Not-Up Heapsort
11:30–12:00	Laurent Rosaz	Improving Katajainen's Ultimate Heapsort
12:00–12:30	Christos Levcopoulos	Computing Minimal Structures on Geometric Inputs
12:30–14:00	Lunch	Kantinen, Fysioterapeutskolen, Universitetsparken 4
14:00–14:30	Andrzej Lingas	Optimal Broadcasting in Tree-Like Networks
14:30–15:00	Mikkel Thorup	Undirected Single Source Shortest Paths in Linear Time
15:00–15:30	Break	
15:30–16:00	Jyrki Katajainen	Worst-Case Efficient External-Memory Priority Queues
16:00–16:30	Peter Bro Miltersen	Error Correcting Codes, Perfect Hashing Circuits, and Deterministic Dynamic Dictionaries
16:30–16:40	Torben Hagerup	Closing
18:00–	Dinner	Baron von Dy, Frederiksborggade 5

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

1. Alon, N., and Schieber, B. (1987), Optimal preprocessing for answering on-line product queries, Tech. Rep. No. 71/87, Tel Aviv University.

2. Arikati, S. R., Chaudhuri, S., and Zaroliagis, C. D. (1995), All-pairs min-cut in sparse networks, in Proc. 15th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS), Springer Lecture Notes in Computer Science, Vol. 1026, pp. 363–376.

3. Arnborg, S., Lagergren, J., and Seese, D. (1991), Easy problems for tree-decomposable graphs, *J. Algorithms* **12**, pp. 308–340.

4. Bern, M. W., Lawler, E. L., and Wong, A. L. (1987), Linear-time computation of optimal subgraphs of decomposable graphs, *J. Algorithms* **8**, pp. 216–235.

5. Bodlaender, H. L. (1993a), On reduction algorithms for graphs with small treewidth, in Proc. 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), Springer Lecture Notes in Computer Science, Vol. 790, pp. 45–56.

6. Bodlaender, H. L. (1993b), Dynamic algorithms for graphs with treewidth 2, in Proc. 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), Springer Lecture Notes in Computer Science, Vol. 790, pp. 112–124.

7. Bodlaender, H. L. (1996a), A linear-time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.* **25**, pp. 1305–1317.

8. Bodlaender, H. L. (1996b), A partial k-arboretum of graphs with bounded treewidth, Tech. Rep. No. UU–CS–1996–02, Dept. of Computer Science, Utrecht University, Utrecht University, The Netherlands.

9. Borie, R. B., Parker, R. G., and Tovey, C. A. (1992), Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families, *Algorithmica* **7**, pp. 555–581.

10. Chaudhuri, S., and Zaroliagis, C. D. (1995), Shortest path queries in digraphs of small treewidth, Proc. 22nd International Colloquium on Automata, Languages and Programming (ICALP), Springer Lecture Notes in Computer Science, Vol. 944, pp. 244–255.

11. Chazelle, B. (1987), Computing on a free tree via complexity-preserving mappings, *Algorithmica* **2**, pp. 337–361.

12. Courcelle, B. (1990a), Graph rewriting: An algebraic and logic approach, in *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics* (J. van Leeuwen, ed.), Chap. 5, pp. 193–242, Elsevier, Amsterdam.

13. Courcelle, B. (1990b), The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, *Inform. and Comput.* **85**, pp. 12–75.

14. Courcelle, B., and Mosbah, M. (1993), Monadic second-order evaluations on tree-decomposable graphs, *Theor. Comput. Sci.* **109**, pp. 49–82.

15. Hagerup, T. (1997), Dynamic algorithms for graphs of bounded tree-width. Proc., 24th International Colloquium on Automata, Languages and Programming (ICALP), Springer Lecture Notes in Computer Science, Vol. 1256, pp. 292–302.

16. Radhakrishnan, V., Hunt, H B., III, and Stearns, R. E. (1992), Efficient algorithms for solving systems of linear equations and path problems, Proc. 9th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Springer Lecture Notes in Computer Science, Vol. 577, pp. 109–119.

17. Robertson, N., and Seymour, P. D. (1986), Graph Minors. II. Algorithmic aspects of tree-width, *J. Algorithms* **7**, pp. 309–322.

18. Stearns, R. E., and Hunt, H. B., III (1996), An algebraic model for combinatorial problems, *SIAM J. Comput.* **25**, pp. 448–476.

19. van Leeuwen, J. (1990), Graph algorithms, in *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity* (J. van Leeuwen, ed.), Chap. 10, pp. 525–631, Elsevier, Amsterdam.

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[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 $n \log_2 n + \Theta(n)$

[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

1. B. Bollobás, T. I. Fenner, and A. M. Frieze, On the best case of Heapsort, *Journal of Algorithms* **20** (1996) 205–217.

2. S. Carlsson, Average-case result on Heapsort, *BIT* **27** (1987) 2–17.

3. S. Carlsson, A variant of Heapsort with almost optimal number of comparisons, *Information Processing Letters* **24** (1987) 247–250.

4. S. Carlsson, A note on Heapsort, *The Computer Journal* **35** (1992) 410–411.

5. S. Carlsson, J. Chen, and C. Mattsson, Heaps with bits, *Theoretical Computer Science* **164** (1996) 1–12.

6. J. Chen, A framework for constructing heap-like structures in-place, *Proceedings of the 4th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science **762**, Springer-Verlag, Berlin/Heidelberg (1993) 118–127.

7. R. D. Dutton, Weak-heap sort, *BIT* **33** (1993) 372–381.

8. R. Fleischer, A tight lower bound for the worst case of Bottom-Up-Heapsort, *Proceedings of the 2nd International Symposium on Algorithms*, Lecture Notes in Computer Science **557**, Springer-Verlag, Berlin/Heidelberg (1991) 251–262.

9. R.W. Floyd, Algorithm 113: TREESORT, *Communications of the ACM* **5** (1962) 434.

10. R. W. Floyd, Algorithm 245: TREESORT 3, *Communications of the ACM* **7** (1964) 701.

11. G. H. Gonnet and J. I. Munro, Heaps on heaps, *SIAM Journal on Computing* **15** (1986) 964–971.

12. X. Gu and Y. Zhu, Optimal heapsort algorithm, *Theoretical Computer Science* **163** (1996) 239–243.

13. J. Katajainen, *The Ultimate Heapsort*, Report 96/42, Department of Computer Science, University of Copenhagen, Copenhagen (1996).

14. J. Katajainen, T. Pasanen, and J. Teuhola, Practical in-place mergesort, *Nordic Journal of Computing* **3** (1996) 27–40.

15. R. Schaffer and R. Sedgewick, The analysis of Heapsort, *Journal of Algorithms* **15** (1993) 76–100.

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.

18. I. Wegener, A simple modification of Xunrang and Yuzhang's HEAPSORT variant improving its complexity significantly, *The Computer Journal* **36** (1993) 286–288.

19. J. W. J. Williams, Algorithm 232: HEAPSORT, *Communications of the ACM* **7** (1964) 347–348.

20. G. Xunrang and Z. Yuzhang, A new HEAPSORT algorithm and the analysis of its complexity, *The Computer Journal* **33** (1990) 281–282.

21. G. Xunrang and Z. Yuzhang, Asymptotic optimal HEAPSORT algorithm, *Theoretical Computer Science* **134** (1994) 559–565.

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 $n \log_2 n + Cn + o(n)$ and in numbers of element moves less than $n \log_2 n + Mn + o(n)$, 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

1. B. Bollobás, T. I. Fenner, and A. M. Frieze, On the best case of Heapsort, *Journal of Algorithms* **20** (1996) 205–217.

2. S. Carlsson, A variant of Heapsort with almost optimal number of comparisons, *Information Processing Letters* **24** (1987) 247–250.

3. S. Carlsson, A note on Heapsort, *The Computer Journal* **35** (1992) 410–411.

4. S. Carlsson, J. Chen, and C. Mattsson, Heaps with bits, *Theoretical Computer Science* **164** (1996) 1–12.

5. J. Chen, A framework for constructing heap-like structures in-place, *Proceedings of the 4th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science **762**, Springer-Verlag, Berlin/Heidelberg (1993) 118–127.

6. R. Fleischer, A tight lower bound for the worst case of Bottom-Up-Heapsort, *Proceedings of the 2nd International Symposium on Algorithms*, Lecture Notes in Computer Science **557**, Springer-Verlag, Berlin/Heidelberg (1991) 251–262.

7. R. W. Floyd, Algorithm 245: TREESORT 3, *Communications of the ACM* **7** (1964) 701.

8. G. H. Gonnet and J. I. Munro, Heaps on heaps, *SIAM Journal on Computing* **15** (1986) 964–971.

9. J. Katajainen, *The Ultimate Heapsort*, Report 96/42, Department of Computer Science, University of Copenhagen, Copenhagen (1996).

10. D. E. Knuth, *The Art of Computer Programming*, Volume 3/ *Sorting and Searching*, Addison-Wesley Publishing Company, Reading (1973).

11. T. W. Lai and D. Wood, *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science **318**, Springer-Verlag, Berlin/Heidelberg (1988) 14–23.

12. C. L. McMaster, An analysis of algorithms for the Dutch national flag problem, *Communications of the ACM* **21** (1978) 842–846.

13. R. Schaffer and R. Sedgewick, The analysis of Heapsort, *Journal of Algorithms* **15** (1993) 76–100.

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.

16. I. Wegener, A simple modification of Xunrang and Yuzhang's HEAPSORT variant improving its complexity significantly, *The Computer Journal* **36** (1993) 286–288.

17. J. W. J. Williams, Algorithm 232: HEAPSORT, *Communications of the ACM* **7** (1964) 347–348.

18. G. Xunrang and Z. Yuzhang, A new HEAPSORT algorithm and the analysis of its complexity, *The Computer Journal* **33** (1990) 281–282.

19. G. Xunrang and Z. Yuzhang, Asymptotic optimal HEAPSORT algorithm, *Theoretical Computer Science* **134** (1994) 559–565.

Computing Minimal Structures on Geometric Inputs

Christos Levcopoulos

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

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 $I\!R^d$, the complete linkage clustering can be computed in optimal $O(n \log n)$ time, under the L_1 and L_∞-metrics. We also design efficient algorithms for approximating the complete linkage clustering.

- A minimum spanning tree of n points in $I\!R^d$ can be obtained in optimal $O(T_d(n, m))$ time, where $T_d(n, m)$ denotes the time to find a closest bichromatic 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 ϵ 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

1. D. Krznaric and C. Levcopoulos. *The First Subquadratic Algorithm for Complete Linkage Clustering.* In Proc. of 6th International Symposium on Algorithms and Computation (ISAAC '95), Lecture Notes in Computer Science 1004, pp. 392–401, Springer-Verlag, 1995.

2. C. Levcopolos and D. Krznaric. *Quasi-Greedy Triangulations Approximating the Minimum Weight Triangulation.* In Proc. of 7th ACM-SIAM Symposium on Discrete Algorithms (SODA'96), 1996.

3. C. Levcopolos and D. Krznaric. *A Fast Heuristic for Approximating the Minimum Weight Triangulation.* In Proc. SWAT'96, Lecture Notes in Computer Science, Vol. 1097, pp. 296-308, Springer Verlag, 1996.

4. C. Levcopolos and D. Krznaric. *A Near-Optimal Heuristic for Minimum Weight Triangulation of Convex Polygons.* In Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA'97), pp. 518-527, New Orleans, Louisiana, January 1997.

5. C. Levcopolos and J. Gudmundsson. *A Linear-Time Heuristic for Minimum Rectangular Covering.* Technical Report LU-CS-TR:96-170, Dept. of Computer Science, Lund University. Accepted for publication in the Proceedings of FCT'97 (Foundations of Computation Theory), LNCS, Vol. 1279, pp. 305–316, Springer Verlag, 1997.

6. D. Krznaric and C. Levcopoulos. *Optimal Algorithms for Complete Linkage Clustering in d Dimensions.* Technical Report LU-CS-TR:96-180 Lund University. Accepted for publication in the Proceedings of MFCS'97 (Mathematical Foundations of Computer Science), LNCS, Springer Verlag. Also selected to the special MFCS issue of Theoretical Computer Science.

7. D. Krznaric, C. Levcopoulos, and B. Nilsson. *Minimum Spanning Trees in d Dimensions.* Technical Report LU-CS-TR:96-183, Lund University. Accepted for publication in the Proceedings of ESA'97 (European Symposium on Algorithms), LNCS, Springer Verlag.

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 broadcasting 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 minimum 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 Mathematics 23 (1989), pp. 11-24.

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 simultaneous send/receive systems, Discr. Appl. Math. 53 (1994), 95-105.

4. D.P. Bertsekas and J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, (1989).

5. H. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. Proc. 33rd ACM STOC, pp. 226-234.

6. G. Fox, M. Johnsson, G. Lyzenga, S. Otto, J. Salmon and D. Walker, Solving Problems on Concurrent Processors, Volume I, Prentice Hall, (1988).

7. H.N. Gabow and R.E. Tarjan, A linear time algorithm for a special case of disjoint set union, J. Comput. System Sc. 30 (1985), 209-221.

8. M. Garey and D. Johnson, Computers and Intractability: a guide to the theory of NP-completeness, Freeman and Co., San Francisco (1979).

9. Y. Gurevich, L. Stockmeyer and U. Vishkin, Solving NP-Hard Problems on Graphs That Are Almost Trees and an Application to Facility Location Problems, Journal of the Association for Computing Machinery, Vol. 31, No. 3, July 1984, pp. 459-473.

10. S.M. Hedetniemi, S.T. Hedetniemi and A.L. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1988), 319-349.

11. A. Jacoby, R. Reischuk, Ch. Schindelhaner, The complexity of broadcasting in planar and decomposable graphs, Proc. 20th International Workshop WG'94, June 1994, LNCS 903, 219-231.

12. S.L. Johnsson and C.T. Ho, Matrix multiplication on Boolean cubes using generic communication primitives, in: Parallel Processing and Medium-Scale Multiprocessors, A. Wouk (Ed.), SIAM, (1989), 108-156.

13. G. Kortsarz and D. Peleg, Approximation Algorithms for Minimum Time Broadcast, Proc. of the Israel Symposium on Theoretical Computer Science, LNCS 601, 1992.

14. O.-H. Kwon and K.-Y. Chwa, Multiple Message Broadcasting in Communication Networks, Networks, vol. 26 (1995), 253-261.

15. P.J. Slater, E. Cockayne and S.T. Hedetniemi, Information dissemination in trees, SIAM J. Comput. 10 (1981), 692-701.

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 bottle-neck by building a hierechical bucketing structure, identifying vertex pairs that may be visited in any order.

Related Literature

1. R.K. Ahuja, K. Mehlhorn, J.B. Orlin, and R.E. Tarjan, Faster algorithms for the shortest path problem, *J. ACM* **37** (1990) 213–223.

2. S. Albers and T. Hagerup, Improved parallel integer sorting without concurrent writing, *Information and Control* **136** (1997) 25–51.

3. A. Andersson, T. Hagerup, S. Nilsson, and R. Raman. Sorting in linear time? In *Proc. 27th ACM Symposium on Theory of Computing (STOC)*, pages 427–436, 1995.

4. A. Andersson, P.B. Miltersen, and M. Thorup. Fusion trees can be implemented with AC^0 instructions only. BRICS-TR-96-30, Aarhus, 1996. To appear in *Theor. Comp. Sc.*

5. E.W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1** (1959), 269–271.

6. E.A. Dinic, Economical Algorithms for Finding Shortest Paths in a Network, In *Transportation Modeling Systems*, Y.S. Popkov and B.L. Shmulyian (eds), Institute for System Studies, Moscow, 1978, 36–44.

7. B.V. Cherkassky, A.V. Goldberg, and C. Silverstein, Buckets, heaps, lists, and monotone priority queues. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 83–92, 1997.

8. H.N. Gabow and R.E. Tarjan, A linear-time algorithm for a special case of disjoint set union. *J. Comp. Syst. Sc.* 30:209–221, 1985.

9. M.L. Fredman and R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. ACM* **34** (1987) 596–615.

10. M.L. Fredman and D.E. Willard, Surpassing the information theoretic bound with fusion trees. *J. Comp. Syst. Sc.* 47:424–436, 1993.

11. M.L. Fredman and D.E. Willard, Trans-dichotomous algorithms for minimum spanning trees and shortest paths, *J. Comp. Syst. Sc.* **48** (1994) 533–551.

12. M.R. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster Shortest-Path Algorithms for Planar Graphs. *J. Comp. Syst. Sc.* **53** (1997) 2–23. See also STOC'94.

13. J.B. Kruskal, On the shortest spanning subtree of a graph and the traviling salesman problem. *Proc. AMS* **7** (1956), 48–50.

14. K. Mehlhorn and S. Näher, Bounded ordered dictionaries in $O(\log\log N)$ time and $O(n)$ space, *Inf. Proc. Lett.* **35,** 4 (1990), 183–189.

15. R. Raman. Priority queues: small monotone, and trans-dichotomous. *Proc. ESA'96, LNCS 1136*, 1996, 121–137.

16. R. Raman. Recent results on the single-source shortest paths problem. *SICACT News* **28**, 2 (1997), 81–87.

17. R.E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM* 22:215-225, 1975.

18. M. Thorup. On RAM priority queues. In *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 59–67, 1996.

19. M. Thorup. Floats, Integers, and Single Source Shortest Paths. Technical Report DIKU-TR-97/19, Dept. Computer Science, University of Copenhagen, 1997.

20. P. van Emde Boas, Preserving order in a forest in less than logarithmic time and linear space, *Inf. Proc. Lett.* **6** (1977), 80–82.

21. P. van Emde Boas, R. Kaas, and E. Zijlstra, Design and implementation of an efficient priority queue, *Math. Syst. Th.* **10** (1977), 99–127.

22. J.W.J. Williams, Heapsort, *Comm. ACM* **7**, 5 (1964), 347–348.

Worst-Case Efficient External-Memory Priority Queues

Gerth Stølting Brodal

Max-Planck-Institut für Informatik
Im Stadtwald
D–66123 Saarbrücken
Germany

Jyrki Katajainen

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

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 $B < c \log_{M/B} \frac{N}{M}$, $\frac{c}{B} \log_{M/B} \frac{N}{M}$ 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 $O(\sum_{i=1}^{S} \frac{1}{B} \log_{M/B} \frac{N_i}{M})$ I/Os, where N_i denotes the number of elements stored in the data structure prior to the ith operation, without giving any guarantee for the performance of the individual operations.

Related Literature

1. A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, Volume 31, pages 1116–1127, 1988.

2. T. O. Alanko, H. H. A. Erkiö, and I. J. Haikala. Virtual memory behavior of some sorting algorithms. *IEEE Transactions on Software Engineering*, Volume SE-10, pages 422–431, 1984.

3. L. Arge. The buffer tree: A new technique for optimal I/O-algorithms. In *Proceedings of the 4th Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science 955, pages 334–345, Springer, Berlin/Heidelberg, 1995.

4. L. Arge. Efficient external-memory data structures and applications. BRICS Dissertation DS-96-3, Department of Computer Science, University of Aarhus, Århus, 1996.

5. R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, Volume 1, pages 173–189, 1972.

6. M. R. Brown. Implementation and analysis of binimial queue algorithms. *SIAM Journal on Computing*, Volume 7, pages 298–319, 1978.

7. S. Carlsson, J. I. Munro, and P. V. Poblete. An implicit binomial queue with constant insertion time. In *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science 318, pages 1–13, Springer-Verlag, Berlin/Heidelberg, 1988.

8. D. Comer. The ubiquitous B-tree. *ACM Computing Surveys*, Volume 11, pages 121–137, 1979.

9. C. A. Crane. Linear lists and priority queues as balanced trees. Technical Report STAN-CS-72-259, Computer Science Department, Stanford University, Stanford, 1972.

10. R. Fadel, K. V. Jakobsen, J. Katajainen, and J. Teuhola. Heaps and heapsort on secondary storage. Submitted to *Theoretical Computer Science*. A preliminary version appeared as "External heaps combined with effective buffering" in *Proceedings of the Computing: The*

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

11. M. J. Fischer and M. S. Paterson. Fishspear: A priority queue algorithm. *Journal of the ACM*, Volume 41, pages 3–30, 1994.

12. D. Naor, C. U. Martel, and N. S. Matloff. Performance of priority queue structures in a virtual memory environment. *The Computer Journal*, Volume 34, pages 428–437, 1991.

13. M. Thorup. On RAM priority queues. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 59–67, ACM, New York and SIAM, Philadelphia, 1996.

14. J. Vuillemin. A data structure for manipulating priority queues. *Communications of the ACM*, Volume 21, pages 309–315, 1978.

15. L. M. Wegner and J. I. Teuhola. The external heapsort. *IEEE Transactions on Software Engineering*, Volume 15, pages 917–925, 1989.

16. J. W. J. Williams. Algorithm 232, Heapsort. *Communications of the ACM*, Volume 7, pages 347–348, 1964.

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*, pages 135–141, Burlington, Vermont, 1996.

2. A. Andersson, P.B. Miltersen, S. Riis, and M. Thorup. Static dictionaries on AC^0 RAMs: Query time $\Theta(\sqrt{\log n/\log\log n})$ is necessary and sufficient. In *36th IEEE Symposium on Foundations of Computer Science*, pages 538–546, Burlington, Vermont, 1996.

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*, pages 538–546, Palo Alto, California, 1993.

4. J. Bierbrauer, I. Johansson, G. Kabatianskii, and B. Smeets. On families of hash functions via geometric codes and concatenation. In *Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science* vol. 773, pp. 331–342, Springer, 1993.

5. J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, April 1979.

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 preliminary 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. 623, pages 235-246, Springer-Verlag, July 1992.

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, Dortmund, Germany, 1993.

9. M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer Auf Der Heide, H. Rohnert, R .E. Tarjan, Dynamic perfect hashing: upper and lower bounds, *SIAM J. Comput.* **23** (1994) 738–761.

10. M. Dietzfelbinger. Universal Hashing and k-wise Independent Random Variables via Integer Arithmetic without Primes. In *Proc. 13th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science* vol. 1046, pp. 569-580, Springer, 1996.

11. S.I. Gelfand, R.L. Dobrushin, and M.S. Pinsker. On the complexity of coding. In *Second International Symposium on Information Theory*, pages 177–184, Akademiai Kiado, Budapest, 1973.

12. A. Fiat, M. Naor, J.P. Schmidt and A. Siegel, Non-Oblivious Hashing. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 367–376, 1988.

13. M.L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the ACM*, 31(3):538–544, July 1984.

14. M.L. Fredman and D.E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47:424–436, 1993.

15. M.L. Fredman and D.E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences*, 48(3):533–551, June 1994.

16. O. Goldreich and A. Wigderson, On the Circuit Complexity of Perfect Hashing, *Electronic Colloquium on Computational Complexity*, TR96-04, 1996.

17. R. Raman. Priority queues: Small, monotone, and trans-dichotomus. In *Proceedings 4th European Symposium on Algorithms*, volume 1136 of *Lecture Notes in Computer Science*, pages 121–137. Springer-Verlag, 1996.

18. D.A. Spielman. Linear-time encodable and decodable error-correcting codes. In *Proceedings 27th annual ACM symposium on the theory of computing*, pages 388–397, Las Vegas, Nevada, 1994.

19. D.R. Stinton. On the connections between universal hashing, combinatorial designs and error-correcting codes. *Electronic Colloquium on Computational Complexity*, TR95-052, 1995.

20. S.C. Sahinalp and U. Vishkin. Efficient approximate and dynamic matching of patterns using a labelling paradigm. In *36th IEEE Symposium on Foundations of Computer Science*, pages 320–328, Burlington, Vermont, 1996.

21. R.E. Tarjan and A.C. Yao. Storing a sparse table. *Communications of the ACM*, 22(11):606–611.

22. M. Thorup. On RAM priority queues. In *7th ACM-SIAM Symposium on Discrete Algorithms*, pages 59–67, Atlanta, Georgia, 1996.

List of Participants

Stephen Alstrup	stephen@diku.dk	Department of Computer Science, University of Copenhagen
Arne Andersson	Arne.Andersson@dna.lth.se	Department of Computer Science University of Lund
John Anker Corneliussen	anker@diku.dk	Department of Computer Science, University of Copenhagen
Anders Dessmark	andersd@dna.lth.se	Department of Computer Science University of Lund
Joakim Gudmundsson	Joakim.Gudmundsson@dna.lth.se	Department of Computer Science University of Lund
Torben Hagerup	torben@mpi-sb.mpg.de	Department of Computer Science, University of Copenhagen and Max-Planck-Institut, Saarbrücken
Jacob Holm	samson@diku.dk	Department of Computer Science, University of Copenhagen
Jyrki Katajainen	jyrki@diku.dk	Department of Computer Science, University of Copenhagen
László B. Kovács	klbzyx@diku.dk	Department of Computer Science, University of Copenhagen
Jesper Larsson	Jesper.Larsson@dna.lth.se	Department of Computer Science University of Lund
Christos Levcopoulos	Christos.Levcopoulos@dna.lth.se	Department of Computer Science University of Lund
Andrzej Lingas	Andrzej.Lingas@dna.lth.se	Department of Computer Science University of Lund
Peter Bro Miltersen	bromille@brics.dk	BRICS, University of Aarhus
Tomi Pasanen	tpasanen@cs.utu.fi	Turku Centre for Computer Science
David Pisinger	pisinger@diku.dk	Department of Computer Science, University of Copenhagen
Peter Riber	riber@diku.dk	Department of Computer Science, University of Copenhagen
Laurent Rosaz	Laurent.Rosaz@lri.fr	Département d'Informatique, Université Paris Sud
Kurt Swanson	kurt@dna.lth.se	Department of Computer Science University of Lund

Mikkel Thorup	thorup@diku.dk	Department of Computer Science, University of Copenhagen
Pawel Winter	pawel@diku.dk	Department of Computer Science, University of Copenhagen
Martin Zachariasen	martinz@diku.dk	Department of Computer Science, University of Copenhagen
Anna Östlin	Anna.Ostlin@dna.lth.se	Department of Computer Science University of Lund