

---

## References

- [1] E. H. L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. Wiley, 1989.
- [2] J. Abello, A. Buchsbaum, and J. Westbrook. A functional approach to external graph algorithms. *Algorithmica*, 32(3):437–458, 2002.
- [3] W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99:118–133, 1928.
- [4] G. M. Adel'son-Vel'skii and E. M. Landis. An algorithm for the organization of information. *Soviet Mathematics Doklady*, 3:1259–1263, 1962.
- [5] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.
- [6] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
- [7] A. V. Aho, B. W. Kernighan, and P. J. Weinberger. *The AWK Programming Language*. Addison-Wesley, 1988.
- [8] R. K. Ahuja, R. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.
- [9] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, 3(2):213–223, 1990.
- [10] N. Alon, M. Dietzfelbinger, P. B. Miltersen, E. Petrank, and E. Tardos. Linear hash functions. *Journal of the ACM*, 46(5):667–683, 1999.
- [11] A. Andersson, T. Hagerup, S. Nilsson, and R. Ramran. Sorting in linear time? *Journal of Computer and System Sciences*, 57(1):74–93, 1998.
- [12] F. Annexstein, M. Baumslag, and A. Rosenberg. Group action graphs and parallel architectures. *SIAM Journal on Computing*, 19(3):544–569, 1990.
- [13] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.
- [14] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.
- [15] H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.

- [16] R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173–189, 1972.
- [17] R. Beier and B. Vöcking. Random knapsack in expected polynomial time. *Journal of Computer and System Sciences*, 69(3):306–329, 2004.
- [18] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.
- [19] M. A. Bender, E. D. Demaine, and M. Farach-Colton. Cache-oblivious B-trees. In *41st Annual Symposium on Foundations of Computer Science*, pages 399–409, 2000.
- [20] J. L. Bentley and M. D. McIlroy. Engineering a sort function. *Software Practice and Experience*, 23(11):1249–1265, 1993.
- [21] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, pages 643–647, 1979.
- [22] J. L. Bentley and R. Sedgwick. Fast algorithms for sorting and searching strings. In *8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 360–369, 1997.
- [23] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.
- [24] G. E. Blelloch, C. E. Leiserson, B. M. Maggs, C. G. Plaxton, S. J. Smith, and M. Zagha. A comparison of sorting algorithms for the connection machine CM-2. In *3rd ACM Symposium on Parallel Algorithms and Architectures*, pages 3–16, 1991.
- [25] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448, 1972.
- [26] N. Blum and K. Mehlhorn. On the average number of rebalancing operations in weight-balanced trees. *Theoretical Computer Science*, 11:303–320, 1980.
- [27] Boost.org. Boost C++ Libraries. [www.boost.org](http://www.boost.org).
- [28] O. Boruvka. O jistém problému minimálním. *Práce, Moravské Přírodovědecké Společnosti*, pages 1–58, 1926.
- [29] F. C. Botelho, R. Pagh, and N. Ziviani. Simple and space-efficient minimal perfect hash functions. In *10th Workshop on Algorithms and Data Structures*, volume 4619 of Lecture Notes in Computer Science, pages 139–150. Springer, 2007.
- [30] G. S. Brodal. Worst-case efficient priority queues. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 52–58, 1996.
- [31] G. S. Brodal and J. Katajainen. Worst-case efficient external-memory priority queues. In *6th Scandinavian Workshop on Algorithm Theory*, volume 1432 of Lecture Notes in Computer Science, pages 107–118. Springer, 1998.
- [32] M. R. Brown and R. E. Tarjan. Design and analysis of a data structure for representing sorted lists. *SIAM Journal of Computing*, 9:594–614, 1980.
- [33] R. Brown. Calendar queues: A fast  $O(1)$  priority queue implementation for the simulation event set problem. *Communications of the ACM*, 31(10):1220–1227, 1988.
- [34] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, Apr. 1979.

- [35] B. Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the ACM*, 47:1028–1047, 2000.
- [36] B. Chazelle and L. J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986.
- [37] B. Chazelle and L. J. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1(2):163–191, 1986.
- [38] J.-C. Chen. Proportion extend sort. *SIAM Journal on Computing*, 31(1):323–330, 2001.
- [39] J. Cheriyan and K. Mehlhorn. Algorithms for dense graphs and networks. *Algorithmica*, 15(6):521–549, 1996.
- [40] B. V. Cherkassky, A. V. Goldberg, and T. Radzik. Shortest path algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73:129–174, 1996.
- [41] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS, 1997.
- [42] D. Cohen-Or, D. Levin, and O. Remez. Progressive compression of arbitrary triangular meshes. In *IEEE Conference on Visualization*, pages 67–72, 1999.
- [43] S. A. Cook. *On the Minimum Computation Time of Functions*. PhD thesis, Harvard University, 1966.
- [44] S. A. Cook. The complexity of theorem proving procedures. In *3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.
- [45] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. Wiley, 1951.
- [46] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry – Algorithms and Applications*. Springer, 2nd edition, 2000.
- [47] R. Dementiev, L. Kettner, J. Mehnert, and P. Sanders. Engineering a sorted list data structure for 32 bit keys. In *6th Workshop on Algorithm Engineering & Experiments*, New Orleans, 2004.
- [48] R. Dementiev, L. Kettner, and P. Sanders. STXXL: Standard Template Library for XXL data sets. *Software: Practice and Experience*, 2007. To appear, see also <http://stxxl.sourceforge.net/>.
- [49] R. Dementiev and P. Sanders. Asynchronous parallel disk sorting. In *15th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 138–148, San Diego, 2003.
- [50] R. Dementiev, P. Sanders, D. Schultes, and J. Sibeyn. Engineering an external memory minimum spanning tree algorithm. In *IFIP TCS*, Toulouse, 2004.
- [51] L. Devroye. A note on the height of binary search trees. *Journal of the ACM*, 33:289–498, 1986.
- [52] R. B. Dial. Shortest-path forest with topological ordering. *Communications of the ACM*, 12(11):632–633, 1969.
- [53] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A reliable randomized algorithm for the closest-pair problem. *Journal of Algorithms*, 1(25):19–51, 1997.

- [54] M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM Journal of Computing*, 23(4):738–761, 1994.
- [55] M. Dietzfelbinger and C. Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theoretical Computer Science*, 380(1–2):47–68, 2007.
- [56] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [57] E. A. Dinic. Economical algorithms for finding shortest paths in a network. In *Transportation Modeling Systems*, pages 36–44, 1978.
- [58] W. Domschke and A. Drexl. *Einführung in Operations Research*. Springer, 2007.
- [59] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989.
- [60] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near-linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006.
- [61] R. Fleischer. A tight lower bound for the worst case of Bottom-Up-Heapsort. *Algorithmica*, 11(2):104–115, 1994.
- [62] R. Floyd. Assigning meaning to programs. In J. Schwarz, editor, *Mathematical Aspects of Computer Science*, pages 19–32. AMS, 1967.
- [63] L. R. Ford. Network flow theory. Technical Report P-923, Rand Corporation, Santa Monica, California, 1956.
- [64] E. Fredkin. Trie memory. *Communications of the ACM*, 3:490–499, 1960.
- [65] M. L. Fredman. On the efficiency of pairing heaps and related data structures. *Journal of the ACM*, 46(4):473–501, 1999.
- [66] M. L. Fredman, J. Komlos, and E. Szemerédi. Storing a sparse table with  $O(1)$  worst case access time. *Journal of the ACM*, 31:538–544, 1984.
- [67] M. L. Fredman, R. Sedgwick, D. D. Sleator, and R. E. Tarjan. The pairing heap: A new form of self-adjusting heap. *Algorithmica*, 1:111–129, 1986.
- [68] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.
- [69] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *40th IEEE Symposium on Foundations of Computer Science*, pages 285–298, 1999.
- [70] H. N. Gabow. Path-based depth-first search for strong and biconnected components. *Information Processing Letters*, pages 107–114, 2000.
- [71] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
- [72] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [73] B. Gärtner and J. Matousek. *Understanding and Using Linear Programming*. Springer, 2006.

- [74] GMP (GNU Multiple Precision Arithmetic Library). <http://gmplib.org/>.
- [75] A. V. Goldberg. Scaling algorithms for the shortest path problem. *SIAM Journal on Computing*, 24:494–504, 1995.
- [76] A. V. Goldberg. A simple shortest path algorithm with linear average time. In *9th European Symposium on Algorithms*, volume 2161 of Lecture Notes in Computer Science, pages 230–241. Springer, 2001.
- [77] A. V. Goldberg and C. Harrelson. Computing the shortest path:  $A^*$  meets graph theory. In *16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 156–165, 2005.
- [78] M. T. Goodrich and R. Tamassia. JDSL – the data structures library in Java. <http://www.jdsl.org/>.
- [79] G. Graefe and P.-A. Larson. B-tree indexes and CPU caches. In *17th International Conference on Data Engineering*, pages 349–358. IEEE, 2001.
- [80] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
- [81] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 2nd edition, 1994.
- [82] J. F. Grantham and C. Pomerance. Prime numbers. In K. H. Rosen, editor, *Handbook of Discrete and Combinatorial Mathematics*, chapter 4.4, pages 236–254. CRC Press, 2000.
- [83] R. Grossi and G. Italiano. Efficient techniques for maintaining multi-dimensional keys in linked data structures. In *26th International Colloquium on Automata, Languages and Programming*, volume 1644 of Lecture Notes in Computer Science, pages 372–381. Springer, 1999.
- [84] S. Halperin and U. Zwick. Optimal randomized EREW PRAM algorithms for finding spanning forests and for other basic graph connectivity problems. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 438–447, 1996.
- [85] Y. Han and M. Thorup. Integer sorting in  $O(n\sqrt{\log \log n})$  expected time and linear space. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 135–144, 2002.
- [86] G. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–309, 1980.
- [87] J. Hartmanis and J. Simon. On the power of multiplication in random access machines. In *5th IEEE Symposium on Foundations of Computer Science*, pages 13–23, 1974.
- [88] M. Held and R. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.
- [89] M. Held and R. Karp. The traveling-salesman problem and minimum spanning trees, part II. *Mathematical Programming*, 1:6–25, 1971.
- [90] P. V. Hentenryck and L. Michel. *Constraint-Based Local Search*. MIT Press, 2005.
- [91] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–585, 1969.

- [92] C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–281, 1972.
- [93] R. D. Hofstadter. Metamagical themas. *Scientific American*, pages 16–22, January 1983.
- [94] P. Høyer. A general technique for implementation of efficient priority queues. In *3rd Israeli Symposium on Theory of Computing and Systems*, pages 57–66, 1995.
- [95] S. Huddleston and K. Mehlhorn. A new data structure for representing sorted lists. *Acta Informatica*, 17:157–184, 1982.
- [96] J. Iacono. Improved upper bounds for pairing heaps. In *7th Scandinavian Workshop on Algorithm Theory*, volume 1851 of Lecture Notes in Computer Science, pages 32–45. Springer, 2000.
- [97] A. Itai, A. G. Konheim, and M. Rodeh. A sparse table implementation of priority queues. In *8th International Colloquium on Automata, Languages and Programming*, volume 115 of Lecture Notes in Computer Science, pages 417–431. Springer, 1981.
- [98] V. Jarník. O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti*, 6:57–63, 1930.
- [99] K. Jensen and N. Wirth. *Pascal User Manual and Report: ISO Pascal Standard*. Springer, 1991.
- [100] T. Jiang, M. Li, and P. Vitányi. Average-case complexity of shellsort. In *26th International Colloquium on Automata, Languages and Programming*, volume 1644 of Lecture Notes in Computer Science, pages 453–462. Springer, 1999.
- [101] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: Experimental evaluation, part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, 1991.
- [102] K. Kaligosi and P. Sanders. How branch mispredictions affect quicksort. In *14th European Symposium on Algorithms*, volume 4168 of Lecture Notes in Computer Science, pages 780–791. Springer, 2006.
- [103] H. Kaplan and R. E. Tarjan. New heap data structures. Technical Report TR-597-99, Princeton University, 1999.
- [104] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7(7):595–596, 1963.
- [105] D. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm for finding minimum spanning trees. *Journal of the ACM*, 42:321–329, 1995.
- [106] N. Karmakar. A new polynomial-time algorithm for linear programming. *Combinatorica*, pages 373–395, 1984.
- [107] J. Katajainen and B. B. Mortensen. Experiences with the design and implementation of space-efficient deque. In *Workshop on Algorithm Engineering*, volume 2141 of Lecture Notes in Computer Science, pages 39–50. Springer, 2001.
- [108] I. Katriel, P. Sanders, and J. L. Träff. A practical minimum spanning tree algorithm using the cycle property. Technical Report MPI-I-2002-1-003, MPI Informatik, Germany, October 2002.

- [109] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.
- [110] L. Khachiyan. A polynomial time algorithm in linear programming. *Soviet Mathematics Doklady*, 20(1):191–194, 1979.
- [111] V. King. A simpler minimum spanning tree verification algorithm. *Algorithmica*, 18:263–270, 1997.
- [112] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, 2nd edition, 1998.
- [113] D. E. Knuth. *MMIXware: A RISC Computer for the Third Millennium*, volume 1750 of Lecture Notes in Computer Science. Springer, 1999.
- [114] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.
- [115] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2000.
- [116] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.
- [117] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem*. Wiley, 1985.
- [118] LEDA (Library of Efficient Data Types and Algorithms). [www.algorithmic-solutions.com](http://www.algorithmic-solutions.com).
- [119] L. Q. Lee, A. Lumsdaine, and J. G. Siek. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2002.
- [120] L. Levin. Universal search problems. *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.
- [121] I. Lustig and J-F. Puget. Program does not equal program: Constraint programming and its relationship to mathematical programming. *Interfaces*, 31:29–53, 2001.
- [122] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.
- [123] C. Martínez and S. Roura. Optimal sampling strategies in Quicksort and Quickselect. *SIAM Journal on Computing*, 31(3):683–705, 2002.
- [124] C. McGeoch, P. Sanders, R. Fleischer, P. R. Cohen, and D. Precup. Using finite experiments to study asymptotic performance. In *Experimental Algorithmics — From Algorithm Design to Robust and Efficient Software*, volume 2547 of Lecture Notes in Computer Science, pages 1–23. Springer, 2002.
- [125] MCSTL: The Multi-Core Standard Template Library. <http://algo2.iti.uni-karlsruhe.de/singler/mcstl/>.
- [126] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, 27(3):125–128, Mar. 1988.
- [127] K. Mehlhorn. Amortisierte Analyse. In T. Ottmann, editor, *Prinzipien des Algorithmenentwurfs*, pages 91–102. Spektrum Lehrbuch, 1998.
- [128] K. Mehlhorn and U. Meyer. External memory breadth-first search with sublinear I/O. In *10th European Symposium on Algorithms*, volume 2461 of Lecture Notes in Computer Science, pages 723–735. Springer, 2002.

- [129] K. Mehlhorn and S. Näher. Bounded ordered dictionaries in  $O(\log \log N)$  time and  $O(n)$  space. *Information Processing Letters*, 35(4):183–189, 1990.
- [130] K. Mehlhorn and S. Näher. Dynamic fractional cascading. *Algorithmica*, 5:215–241, 1990.
- [131] K. Mehlhorn and S. Näher. *The LEDA Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
- [132] K. Mehlhorn, S. Näher, and P. Sanders. Engineering DFS-based graph algorithms. Submitted, 2007.
- [133] K. Mehlhorn, V. Priebe, G. Schäfer, and N. Sivadasan. All-pairs shortest-paths computation in the presence of negative cycles. *Information Processing Letters*, 81(6):341–343, 2002.
- [134] K. Mehlhorn and P. Sanders. Scanning multiple sequences via cache memory. *Algorithmica*, 35(1):75–93, 2003.
- [135] K. Mehlhorn and M. Ziegelmann. Resource constrained shortest paths. In *8th European Symposium on Algorithms*, volume 1879 of Lecture Notes in Computer Science, pages 326–337, 2000.
- [136] R. Mendelson, R. E. Tarjan, M. Thorup, and U. Zwick. Melding priority queues. In *9th Scandinavian Workshop on Algorithm Theory*, pages 223–235, 2004.
- [137] *Meyers Konversationslexikon*. Bibliographisches Institut, 1888.
- [138] B. Meyer. *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 1997.
- [139] U. Meyer. Average-case complexity of single-source shortest-path algorithms: Lower and upper bounds. *Journal of Algorithms*, 48(1):91–134, 2003.
- [140] U. Meyer and P. Sanders.  $\Delta$ -stepping: A parallel shortest path algorithm. In *6th European Symposium on Algorithms*, number 1461 in Lecture Notes in Computer Science, pages 393–404. Springer, 1998.
- [141] U. Meyer, P. Sanders, and J. Sibeyn, editors. *Algorithms for Memory Hierarchies*, volume 2625 of Lecture Notes in Computer Science. Springer, 2003.
- [142] B. M. E. Moret and H. D. Shapiro. An empirical analysis of algorithms for constructing a minimum spanning tree. In *2nd Workshop on Algorithms and Data Structures*, volume 519 of Lecture Notes in Computer Science, pages 400–411. Springer, 1991.
- [143] R. Morris. Scatter storage techniques. *Communications of the ACM*, 11(1):38–44, 1968.
- [144] S. S. Muchnick. *Advanced Compiler Design and Implementation*. Morgan Kaufmann, 1997.
- [145] S. Näher and O. Zlotowski. Design and implementation of efficient data types for static graphs. In *10th European Symposium on Algorithms*, volume 2461 of Lecture Notes in Computer Science, pages 748–759. Springer, 2002.
- [146] G. Nemhauser and Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.
- [147] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.



- [148] J. Nešetřil, H. Milková, and H. Nešetřilová. Otakar Boruvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1–3):3–36, 2001.
- [149] K. S. Neubert. The flashsort1 algorithm. *Dr. Dobbs's Journal*, pages 123–125, February 1998.
- [150] J. Nievergelt and E. Reingold. Binary search trees of bounded balance. *SIAM Journal of Computing*, 2:33–43, 1973.
- [151] K. Noshita. A theorem on the expected complexity of Dijkstra's shortest path algorithm. *Journal of Algorithms*, 6(3):400–408, 1985.
- [152] R. Pagh and F. Rodler. Cuckoo hashing. *Journal of Algorithms*, 51:122–144, 2004.
- [153] W. W. Peterson. Addressing for random access storage. *IBM Journal of Research and Development*, 1(2), Apr. 1957.
- [154] S. Pettie. Towards a final analysis of pairing heaps. In *46th IEEE Symposium on Foundations of Computer Science*, pages 174–183, 2005.
- [155] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. In *27th International Colloquium on Automata, Languages and Programming*, volume 1853 of Lecture Notes in Computer Science, pages 49–60. Springer, 2000.
- [156] J. Pinkerton. *Voyages and Travels*, volume 2. 1808.
- [157] P. J. Plauger, A. A. Stepanov, M. Lee, and D. R. Musser. *The C++ Standard Template Library*. Prentice Hall, 2000.
- [158] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, pages 1389–1401, Nov. 1957.
- [159] W. Pugh. Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.
- [160] A. Ranade, S. Kothari, and R. Udapa. Register efficient mergesorting. In *High Performance Computing*, volume 1970 of Lecture Notes in Computer Science, pages 96–103. Springer, 2000.
- [161] J. H. Reif. Depth-first search is inherently sequential. *Information Processing Letters*, 20(5):229–234, 1985.
- [162] N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas. Efficiently four-coloring planar graphs. In *28th ACM Symposium on Theory of Computing*, pages 571–575, 1996.
- [163] G. Robins and A. Zelikowsky. Improved Steiner tree approximation in graphs. In *11th ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.
- [164] P. Sanders. Fast priority queues for cached memory. *ACM Journal of Experimental Algorithmics*, 5(7), 2000.
- [165] P. Sanders and D. Schultes. Highway hierarchies hasten exact shortest path queries. In *13th European Symposium on Algorithms*, volume 3669 of Lecture Notes in Computer Science, pages 568–579. Springer, 2005.
- [166] P. Sanders and D. Schultes. Engineering fast route planning algorithms. In *6th Workshop on Experimental Algorithms*, volume 4525 of Lecture Notes in Computer Science, pages 23–36. Springer, 2007.

- [167] P. Sanders and S. Winkel. Super scalar sample sort. In *12th European Symposium on Algorithms*, volume 3221 of Lecture Notes in Computer Science, pages 784–796. Springer, 2004.
- [168] R. Santos and F. Seidel. A better upper bound on the number of triangulations of a planar point set. *Journal of Combinatorial Theory, Series A*, 102(1):186–193, 2003.
- [169] R. Schaffer and R. Sedgwick. The analysis of heapsort. *Journal of Algorithms*, 15:76–100, 1993.
- [170] A. Schönhage. Storage modification machines. *SIAM Journal on Computing*, 9:490–508, 1980.
- [171] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.
- [172] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.
- [173] D. Schultes. *Route Planning in Road Networks*. PhD thesis, 2008.
- [174] R. Sedgwick. Analysis of shellsort and related algorithms. In *4th European Symposium on Algorithms*, volume 1136 of Lecture Notes in Computer Science, pages 1–11. Springer, 1996.
- [175] R. Sedgwick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley, 1996.
- [176] R. Seidel and C. Aragon. Randomized search trees. *Algorithmica*, 16(4–5):464–497, 1996.
- [177] R. Seidel and M. Sharir. Top-down analysis of path compression. *SIAM Journal of Computing*, 34(3):515–525, 2005.
- [178] M. Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers and Mathematics with Applications*, 7(1):67–72, 1981.
- [179] J. C. Shepherdson and H. E. Sturgis. Computability of recursive functions. *Journal of the ACM*, 10(2):217–255, 1963.
- [180] J. Singler, P. Sanders, and F. Putze. MCSTL: The Multi-Core Standard Template Library. In *Euro-Par*, volume 4641 of Lecture Notes in Computer Science, pages 682–694. Springer, 2007.
- [181] M. Sipser. *Introduction to the Theory of Computation*. MIT Press, 1998.
- [182] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.
- [183] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686, 1985.
- [184] D. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.
- [185] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.
- [186] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.
- [187] R. E. Tarjan. Shortest paths. Technical report, AT&T Bell Laboratories, 1981.
- [188] R. E. Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods*, 6(2):306–318, 1985.

- [189] R. E. Tarjan and U. Vishkin. An efficient parallel biconnectivity algorithm. *SIAM Journal on Computing*, 14(4):862–874, 1985.
- [190] M. Thorup. Undirected single source shortest paths in linear time. *Journal of the ACM*, 46:362–394, 1999.
- [191] M. Thorup. Even strongly universal hashing is pretty fast. In *11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 496–497, 2000.
- [192] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004.
- [193] M. Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. In *35th ACM Symposium on Theory of Computing*, pages 149–158, 2004.
- [194] M. Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, 69(3):330–353, 2004.
- [195] M. Thorup and U. Zwick. Approximate distance oracles. In *33rd ACM Symposium on the Theory of Computing*, pages 183–192, 2001.
- [196] A. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 150(3):496–498, 1963.
- [197] Unknown. *Der Handlungsreisende – wie er sein soll und was er zu thun hat, um Auftraege zu erhalten und eines gluecklichen Erfolgs in seinen Geschaeften gewiss zu sein – Von einem alten Commis-Voyageur*. 1832.
- [198] P. van Emde Boas. Preserving order in a forest in less than logarithmic time. *Information Processing Letters*, 6(3):80–82, 1977.
- [199] R. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2001.
- [200] V. Vazirani. *Approximation Algorithms*. Springer, 2000.
- [201] J. von Neumann. First draft of a report on the EDVAC. Technical report, University of Pennsylvania, 1945.
- [202] J. Vuillemin. A data structure for manipulating priority queues. *Communications of the ACM*, 21:309–314, 1978.
- [203] L. Wall, T. Christiansen, and J. Orwant. *Programming Perl*. O’Reilly, 3rd edition, 2000.
- [204] 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(1):81–98, 1993.
- [205] I. Wegener. *Complexity Theory: Exploring the Limits of Efficient Algorithms*. Springer, 2005.
- [206] R. Wickremesinghe, L. Arge, J. S. Chase, and J. S. Vitter. Efficient sorting using registers and caches. *ACM Journal of Experimental Algorithmics*, 7(9), 2002.
- [207] R. Wilhelm and D. Maurer. *Compiler Design*. Addison-Wesley, 1995.
- [208] J. W. J. Williams. Algorithm 232: Heapsort. *Communications of the ACM*, 7:347–348, 1964.

ERHEBUNG  
KOPF

---

## Index

- “folklore” (result), 79  
15-puzzle, 248
- Aarts, E. H. L., 255  
( $a, b$ )-tree, *see under* sorted sequence  
Abello, J., 232  
Ackermann, W., 224  
Ackermann function (inverse), 224  
addition, **2**  
address, 24, 27  
Adel’son-Vel’skii, G. M., 165  
adjacency array, *see under* graph  
adjacency list, *see under* graph  
adjacency matrix, *see under* graph  
adjacent, **49**  
Aggarwal, A., 120  
Aho, A. V., 165  
Ahuja, R. K., 201  
al-Khwarizmi, Muhammad ibn Musa, **1**, 6  
ALD, *see under* shortest path  
algorithm, **1**  
algorithm analysis, 36, *see also*  
    running time, **36**  
amortized, 60, 135, 158, 203  
    accounting method, **68**  
    binary counter, 70  
    deamortization, 70  
    general definition, **71**  
    operation sequence, **71**  
    potential method, **68**  
    token, **68**  
    unbounded array, 66  
    universality of potential method, **73**
- approximation algorithm, 241  
average case, **41**, 84, 103, 107, 109, 115,  
    117, 124, 148, 199, 205, 245  
global, **41**  
master theorem, **37**, 104  
randomized, **45**, 107, 109, 115, 121  
recursion, 9, 16, 37, 104  
recursive, 9, 12  
smoothed analysis, 262  
sum, 4, **36**  
worst case, 109  
algorithm design, 1  
    “make the common case fast”, 66  
algebraic, 9, 86, 87, 89, 101, 171, 174  
black-box solvers, 234, 248, 261  
certificate, **33**, 36, 51, 187  
deterministic, **46**, 100  
divide-and-conquer, **7**, 34, 37, 103  
    building a heap, 131  
    mergesort, 103  
    MSD radix sort, 117  
    multiplication, 7  
    multiway mergesort, 119  
    quicksort, 108, 114  
dynamic programming, **243**, 261  
    Bellman–Ford algorithm, 206  
    changing money, 245  
    knapsack, 243, 245  
    matrix products, chained, 245  
    minimum edit distance, 245  
    principle of optimality, **243**, 246  
    shortest paths, 193  
evolutionary algorithm, **259**, 262

- greedy, 101, **239**, 257, 261
    - changing money, 245
    - cycle detection, 51
    - Dijkstra's algorithm, 196
    - Jarník–Prim algorithm, 219
    - knapsack, 239, 240
    - Kruskal's algorithm, 221
    - machine scheduling, 241
  - local search, **249**, 262
    - hill climbing, 250
    - relaxing constraints, 256
    - restarts, 259
    - simplex algorithm, 250
    - simulated annealing, 252
    - tabu search, 258
    - threshold acceptance, 258
  - lookup table, 203
  - preprocessing, 34, 100
  - random sampling, 120, 232
  - randomized, **45**, 92, 125, 165, **226**, 262
    - Las Vegas, **48**, 85, 108, 114
    - Monte Carlo, **48**, 101
  - recursion, 7, 9, 53, 104, 108, 113, 114, 117, 131, 178, 246
  - result checking, 6, **33**, 101, 198
  - systematic search, **246**, 248, 261
    - constraint programming, 248, **262**
    - ILP solving, 248
    - iterative deepening, 248
    - knapsack, 246
    - use of sorting, 34, 99–101, 125, 172, 239
  - algorithm engineering, **1**, 5, 10, 11, 92, 95, 111, 120, 123, 125, 163, 199, 209, 257, 261
  - alignment, 8, 163
  - all-pairs shortest path, *see under* shortest path
  - allocate**, **27**
  - Alon, N., 97
  - amortized, *see under* algorithm analysis
  - analysis, *see also* algorithm analysis
  - ancestor, **52**
  - AND, **24**
  - Andersson, A, 125
  - antisymmetric, **264**
  - Applegate, D. L., 230
  - approximation algorithm, 217, **240**
  - approximation ratio, **240**
  - Aragon, C. R., 257
  - Aragon, S. R., 165
  - arbitrage, 207
  - Arge, L., 123
  - arithmetic, 26
  - arithmetics, 24
  - array, 26, **26**, 59
    - access `[.]`, **66**
    - associative, 81
    - find*, **82**
    - forall*, **82**
    - insert*, **82**
    - remove*, **82**
  - circular, **75**, 201
  - growing, 66
  - popBack, **66**
  - pushBack, **66**
  - reallocate*, 66
  - shrinking, 66
  - size, **66**
  - sorting, 111
  - unbounded, 170
- assertion, **32**
  - assignment, 28
  - asymptotic, 11, 20, **21**, 25
  - Ausiello, G., 54
  - average case, *see under* running time
  - AVL tree, *see under* sorted sequence
  - AWK, 81
  - B* (block size), 25
  - B-tree, 163
  - bandwidth, 25
  - base, 1
  - Bast, H., 212
  - Bayer, R., 163
  - Beier, R., 245
  - Bellman, R., 206
  - Bellman–Ford algorithm, *see under* shortest path
  - Bender, M. A., 165
  - Bentley, J. L., 124
  - Bertsekas, D. P., 262
  - best case, *see under* running time
  - best-first branch-and-bound, 128
  - bin packing, 146, 242
  - binary heap, *see under* priority queue
  - binary operation, 24
  - binary search, *see under* searching

- binary search tree, *see under*
  - sorted sequence
- binomial coefficient, **270**
- binomial heap, *see under* priority queue
- binomial tree, 137
- bisection method, **35**
- bit operation, **24**
- Bixby, E. E., 230
- Blelloch, G. E., 125
- block, *see* memory block
- Blum, N., 124, 165
- Boolean formula, 242
- Boolean value, **26**
- Boost, 57
  - Bellman–Ford algorithm, 214
  - Dijkstra’s algorithm, 214
  - graph, 173
  - graph traversal, 189
  - union–find, 231
- Boruvka, O., 231
- Botelho, F., 97
- bottleneck shortest path, 217
- bottom-up heap operation, 131
- bounded array, **59**
- branch, 24
- branch prediction, 125, 162
- branch-and-bound, 128, 246
- branch-and-cut, 249
- Bro Miltersen, P., 97
- Brodal, G., 141, 143
- Brown, M. R., 79
- Brown, R., 143
- Buchsbaum, A., 232
- bucket, 121
- bucket sort, *see under* sorting
  
- C, 26
- C++, 17, 26, 31, 57, 78, 96, 123, 142, 164, 173, 214, 231
- cache, 24
  - limited associativity, 123
- cache-oblivious, 142, 165
- calendar queue, *see under* priority queue
- call by reference, **29**
- call by value, **29**
- carry, 1, 2
- Carter, J., 97
- cascading cut, 138
- casting out nines, 6
  
- Cayley, A., 174
- census, 99
- certificate, *see* algorithm design
- certifying algorithm, **33**
- changing money, 245
- characteristic function, 54
- Chase, S., 123
- Chazelle, B., 166, 232
- checksum, 6
- Cheriyán, J., 189
- Cherkassky, B., 214
- Chernoff bound, 122, **269**
- chess, 81
- child, **52**
- Chvátal, V., 230
- class, 26, 27, **31**
- clique, *see under* graph
- clock cycle, 25
- clustering, 217
- Coffman, E. G., 146
- Cohen-Or, D., 174
- collision, 82
- combinatorial search, 81
- comparison, **24**
  - three-way, 34, 108, 109
  - two-way, 35
- comparison-based algorithm, 34, 106
- competitive ratio, **242**
- compiler, 3, 26, 58, 81, 123
  - symbol table, 81
- complex number, 31, 100
- complexity, 24, *see also* running time
- complexity theory, 54
- composite data structure, 27
- composite type, 26
- computation, model of, 24
- concave function, 200, **265**
- conditional branch instruction, 125
- conditional statement, **28**
- cone, 251
- congruent, 264
- constant, **24**
- constant factor, 21, 25
- constraint, 235
- constraint programming, *see under*
  - algorithm design, systematic search
- contract, **32**
- convex, **265**
- convex polytope, 251

- Cook, W. J., 18, 230  
 cooling schedule, 354  
 coprocessor, 25  
 core, 25  
 correctness, 31  
 cost vector, 235  
 crossover operation, 260  
 C#, 26  
 cuneiform script, 59  
 cycle, 50  
   Hamiltonian, 50, 54  
   simple, 50  
   testing for, 51
- DAG, *see* graph, directed, acyclic  
 Dantzig, G. B., 235  
 data dependency, 24  
 data struct. inv., *see under* invariant  
 data structure, VII  
 data type, *see* type  
 database, 147, 163  
 database join, 81  
 decision problem, 54  
 declaration, 26, 29  
   implicit, 29  
 decrement (--), 28  
 degree, 49  
 Delaunay triangulation, 232  
 Demaine, E. D., 165  
 Dementiev, R., 124, 125, 166, 225  
 deque, 75, 79  
   *first*, 75  
   *last*, 75  
   *popBack*, 75  
   *pushFront*, 75  
   *pushBack*, 75  
   *pushFront*, 75  
 dereference, 27  
 descendant, 52  
 design by contract, 32  
 deterministic algorithm, *see under*  
   algorithm design  
 Devroye, L., 148  
 dictionary, 81, 99  
 diet problem, 235  
 Dietzfelbinger, M., 97  
 digit, 1  
 digraph, *see* graph, directed
- Dijkstra's algorithm, *see under*  
   shortest path  
 Dijkstra, E., 196, 219  
 discrete-event simulation, 128  
 disk, *see* hard disk  
**dispose**, 27  
 distributed system, 25  
**div**, 24  
 division (integer), 6  
 Driscoll, J., 166  
 dynamic programming, *see under*  
   algorithm design  
 dynamic tree, 222
- edge, 49  
   associated information, 167  
   backward, 175, 179  
   contraction, 189  
   cost, 50  
   cross, 175, 179, 181  
   crossing, 51  
   forward, 175, 179  
   parallel, 167, 173  
   reduced cost, 207, *see also*  
     node potential  
     tree, 175, 179  
   weight, 50, 167  
 edge contraction, 226  
 edge query, 168, 171  
*edgeArray*, 168  
 efficiency, *see* running time  
 Eiffel, 56  
 eight queens problem, 248, 256  
 element, 26, 99  
 empty sequence {}, 27  
 equals (=), 24  
 equivalence relation, 265  
 Eratosthenes, 31  
 event, 266  
 evolutionary algorithm, *see under*  
   algorithm design  
 exchange argument, 219, 239  
 exclusive OR ( $\oplus$ ), 24  
 execution time, *see* running time  
 existence problem, 233  
 expected value, 41, 266  
 exponential search, 35  
 external memory, *see also* machine model  
   building heap, 132



- lower bound, 120
- merging, 119
- MST, 225
- parallel disks, 120, 125
- priority queue, **139**
- queue, 76
- scanning, 119
- semiexternal algorithm, 226
- sorting, 118, 120, 124
- stack, 76
  
- Fakcharoenphol, J., 215
- false*, **24**
- Farach-Colton, M., 165
- fast memory, 25
- ferry connections, 217
- Fibonacci, L., 135
- Fibonacci heap, *see under* priority queue
- field (algebraic), **86, 265**
- field (of variable), 27
- FIFO queue, **74, 177**
  - external-memory, 76
  - first*, **74**
  - popFront*, **74**
  - pushBack*, **74**
  - using circular array, 75
  - using two stacks, 75
- file, 27
- filig card, 145
- Flajolet, P., 40
- Fleischer, R., 142
- floating-point, **24, 56, 203**
- flow, 237
- Floyd, R. W., 58, 124
- for, **28**
- Ford, L. R., Jr., 206
- forest, **51**
- Fredkin, E., 166
- Fredman, M. L., 97, 135, 143
- frequency allocation, 258
- Frigo, M., 142
- function object, 96
- function pointer, 123
- Funke, S., 212
  
- Gabow, H., 189
- Gärtner, B., 262
- garbage collection, **57**
- Garey, M. R., 54, 146
- generic methods, 233
- generic programming, **31, 173**
- genome, 259
- geometric series, *see under* sum
- geometry, 252
- GMP, 17
- Goldberg, A., 205, 212, 214
- Goodrich, M. T., 174
- Graefe, G., 163
- Graham, R. L., 40, 58, 241
- graph, **49**
  - 2-edge-connected components, 187
  - adjacency array, **168**
  - adjacency list, **170**
  - adjacency matrix, **171**
    - undirected, 171
  - average degree, 228
  - BFS, **176, 192**
    - implementation, 188
  - biconnected components, 188, 189
  - bidirected, **49, 167, 170**
  - bipartite, **34, 174**
  - breadth-first search, *see* BFS
  - Cayley, 174
  - citation network, 167
  - clique, **54, 55**
  - coloring, **34, 54, 55, 255, 257**
    - fixed-*K* annealing, 258
    - Kempe chain annealing, 255
    - penalty function annealing, 256
    - XRLF greedy algorithm, 257
  - communication network, 175
  - complete, 54
  - component, **50**
  - compression, 174
  - connected components, **50, 177**
  - construction, 168
  - conversion, 168, 169
  - counting paths, 171
  - cut, 172, 218
  - cycle detection, 170
  - DAG, *see* graph, directed, acyclic (DAG)
  - dense, 171
  - depth-first search, *see* DFS
  - DFS, 175, **178, 206**
    - backtrack*, 178
    - init*, 178
    - root*, 178
    - traverseNonTreeEdge*, 178

- traverseTreeEdge*, 178
- diameter, **209**
- directed, **49**
  - acyclic (DAG), **50**, 51, 52, 180
- dynamic, 168, 170
- ear decomposition, 189
- edge, *see under* edge
- edge sequence, 168, 221
- exploration, *see* graph traversal
- face, 174
- grid, 172
- hypergraph, 174
- input, 168
- interval graph, 172
- interval-, 100
- Kempe chain, 255
- layer, 176
- linked edge objects, 170
- minimum spanning tree, *see* MST
- MST, *see* MST
- multigraph, 167, 173
- navigation, 168
- negative cycle, *see under* shortest path
- network design, 217
- node, *see* node
- output, 168
- planar, **51**, 174
  - 4-coloring, 255
  - 5-coloring, 256
  - embedding, 189
  - testing planarity, 189
- random, 208, 257
- random geometric graph, 257
- representation, **167**
- reversal information, 168
- SCC, *see* graph, strongly connected component
- shortest path, *see* shortest path
- shrunk graph, 182
- sparse, 170
- static, 168
- Steiner tree, 228
  - 2-approximation, 228
- street network, 51
- strongly connected component
  - certificate, 187
  - open, 182
  - strongly connected components, **50**, 175, **181**
    - closed, 183
    - implementation, 188
    - invariant, 182
    - more algorithms, 189
    - open, 183
    - subgraph (induced), **50**
    - topological sorting, 180, 195
    - transitive closure, 177
  - traversal, **175**
  - triconnected components, 189
  - undirected, **49**
  - vertex, *see* node
  - visitor, 189
- graphics processor, 25
- greater than ( $>$ ), **24**
- greedy algorithm, *see under* algorithm design
- Grossi, R., 166
- group, 174
- grouping, 100
- growth rate, 20
- Guibas, L. J., 166
- Hagerup, T., 125
- half-space, 251
- Halperin, S., 232
- Hamilton, W. R., 50
- Han, Y., 125, 143
- handle, **26**, 60, 128, 146
- Handler, G., 215
- hard disk, 25
- harmonic sum, *see under* sum
- Harrelson, C., 212
- hash function, 82
- hash table, *see* hashing
- hashing, **81**, 100
  - closed, **90**
  - large elements, 96
  - large keys, 96
  - linear probing, 83, 90
    - cyclic, 91
    - find*, **90**
    - insert*, **90**
    - remove*, **90**
    - unbounded, 91
  - open, **90**
  - perfect, 92

- perfect (dynamic), 95
- realistic analysis, 86
- universal, 85
  - bit strings, 86
  - by integer multiplication, 89
  - by shifting, 89
  - by table lookup, 89
  - simple linear, 89
  - using bit matrix multiplication, 88
  - using scalar products, 87
- universal family, 86
- unrealistic analysis, 84
- use of, 100, 101, 108, 168
- with chaining, 82, **83**
  - average case, 85
  - fat, **95**
  - find*, **83**
  - implementation, 95
  - insert*, **83**
  - remove*, **83**
  - slim, **95**
  - unbounded, 85
- heap property, **130**, *see also*
  - priority queue
- heapsort, *see under* sorting
- Held, M., 230
- Held–Karp lower bound, *see under*
  - MST
- heuristic, 44
- high-performance computing, 25
- hill climbing, *see under* algorithm design,
  - local search
- $H_n$ , *see* sum, harmonic
- Hoare, C. A. R., 58
- Hollerith, H., 99
- Hopcroft, J., 165
- Huddleston, S., 79, 165
- hyperplane, 250
- Høyer, P., 143
  
- I/O step, 25
- Iacono, J., 143
- IBM, 99
- IEEE floating-point, 56
- if**, **28**
- iff, **265**
- ILP, *see* linear program, integer
- imperative programming, 26
- implementation note, 25
  
- incident, **49**
- increment (++) , **28**
- incumbent, 246
- indentation, 28
- independent random variable, **268**
- index, 26, 59
- indicator random variable, 41, 110
- inequality
  - Chernoff, **269**
  - Jensen’s, **270**
  - Markov’s, 48, **268**
- infinity ( $\infty$ ), **26**, 56
- initialization, **26**
- inlining, **29**
- input, 24
- input size, 20, 23
- inserting into a sequence, 60
- insertion sort, *see under* sorting
- instance, 20
- instruction, 24, **24**
- integer, **26**
- integer arithmetics, **1**
- internal memory, 25
- invariant, **32**, 182
  - data structure invariant, 32, 33, 60, 129, 133, 149, 159, 165, 202, 222
  - loop invariant, **32**, 34, 90, 102
- inverse, **265**
- Itai, A., 165
- Italiano, G., 166
- item, 60
- iteration, 28
- iterative deepening search, 248
- iterator, *see under* STL
  
- Jarník, V., 219
- Jarník–Prim algorithm, *see under*
  - MST
- Java, 18, 26, 31, 57, 79, 96, 124, 164, 214, 231
  - deque, 79
  - hashCode*, 96
  - HashMap*, 96
  - linked list, 79
  - memory management, 79
  - PriorityQueue*, 142
  - SortedMap*, 164
  - SortedSet*, 164
  - sorting, 124

- stack, 79
- TreeMap*, 164
- TreeSet*, 164
- vector, 79
- JDSL, 57
  - Dijkstra's algorithm, 214
  - graph, 174
  - graph traversal, 189
  - MST, 231
  - PriorityQueue*, 142
- Jiang, T., 125
- Johnson, D. S., 54, 146, 257
- jump, 24
  
- Kaligosi, K., 125
- Kaplan, H., 143
- Karatsuba, A., 9
- Karger, D., 232
- Karlin, A., 97
- Karmakar, N., 237
- Karp, R., 230
- Katajainen, J., 79, 141
- Katriel, I., 232
- Kellerer, H., 233
- Kempe, A. B., 256
- Kempe chain, *see under* graph
- Kettner, L., 124, 166
- key, 82, 99, 127
- Khachian, L., 237
- King, V., 232
- Klein, P., 232
- knapsack, 54, 191
- knapsack problem, 233
  - 2-approximation (*round*), 240
  - as an ILP, 238
  - average case, 245
  - branch-and-bound algorithm, 246
  - dynamic programming, 243
    - by profit, 245
  - evolutionary algorithm, 260
  - fractional, 238, 239, 247
  - fractional solver, 239
  - greedy algorithm, 240
  - local search, 250
  - simulated annealing, 255
  - use of, 233
- knot, 59
- Knuth, D., 40, 58, 97, 125
- Komlos, J., 97
  
- Konheim, A. G., 165
- Korf, R. E., 248
- Korst, J., 255
- Korte, B., 232
- Kosaraju, S. R., 189
- Kothari, S., 123
- Kruskal, J., 221
  
- Landis, E. M., 165
- Larsen, P.-A., 163
- Las Vegas algorithm, *see under*
  - algorithm design, randomized
- latency, 25
- Lawler, E. L., 230
- leading term, 22
- leaf, 52
- LEDA, 17, 57
  - Bellman–Ford algorithm, 214
  - bounded stack, 78
  - Dijkstra's algorithm, 214
  - graph, 173
  - graph traversal, 188
  - h\_array*, 96
  - list, 78
  - map, 96
  - MST, 231
  - node\_pq*, 214
  - priority queue, 142
  - queue, 78
  - sortseq*, 164
  - stack, 78
  - static graph, 173
  - union–find, 231
- Lee, L. W., 173
- left-to-right maximum, 42, 45, 110, 200
- Leiserson, C. E., 125, 142
- Lenstra, J. K., 230
- less than (<), 24
- Levenshtein distance, 245
- Levin, D., 174
- lexicographic order, 100, 265
- Li, M., 125
- linear algebra, 171, 252
- linear order, 99, 215, 265
- linear program (LP), 234
  - fractional solution, 238
  - integer (ILP), 236, 238
    - 0–1 ILP, 238, 248
    - branch-and-cut, 249

- knapsack, 238
- pigeonhole principle, 242
- set covering, 239
- maximum flow, 237
- minimum-cost flow, 237
- mixed integer (MILP), **238**
- relaxation of ILP, 238
- rounding, 238
- shortest path, 236
- simplex algorithm, 250
- smoothed analysis, 262
- solver, 262
- strict inequality, 251
- tight inequality, 251
- linearity of expectations, 41, 85, 86, 110, 228, **267**
- list, 27, 59, 83, 170
  - blocked, 76, 106, 118
  - bulk insert, **105**
  - circular, 136, 170
  - concat*, 64, 65
  - concatenate, 60, 65
  - doubly linked, **60**, 145
  - dummy item, **61**, 170
  - empty, 61
  - find*, 63, 65
  - findNext*, 64, 65
  - first*, 64, 65
  - head*, 64, 65
  - insert*, **62**, 64, 65
  - interference between ops., 64
  - invariant, 60
  - isEmpty*, 64, 65
  - last*, 64, 65
  - linked, 60
  - makeEmpty*, 64, 65
  - memory management, 61, 64
  - move item, **61**
  - popBack*, 64
  - popFront*, 64, 65
  - pushBack*, 64, 65
  - pushFront*, 64, 65
  - remove*, 61, 64, 65
  - rotation, 64
  - singly linked, **65**, 95
  - size, **64**
  - sorting, 105
  - splice*, **61**, 65
  - swapping sublists, **64**
- load instruction, **24**
- local search, *see under* algorithm design
- locate*, *see under* sorted sequence
- logarithm, **264**
- logical operations, **24**
- loop, **28**, 36
- loop fusion, 3
- loop invariant, *see under* invariant
- lower bound, 241
  - “breaking”, 116
  - element uniqueness, **108**
  - external sorting, 120
  - minimum, **107**
  - pairing heap priority queue, 143
  - sorting, **106**
- lower-order term, **22**
- LP, *see* linear program
- Lucas, E., 75
- Lumsdaine, A., 173
- Lustig, I. J., 262
- M* (size of fast memory), 25
- machine instruction, *see* instruction
- machine model, 21, **23**
  - accurate, 25
  - complex, 25
  - external memory, 25
  - parallel, 24, **25**
  - RAM, **23**, 26
  - real, 24
  - sequential, 23
  - simple, 25
  - von Neumann, 23
  - word, 125
- machine program, **24**, 26
- machine scheduling, **241**
  - decreasing-size algorithm, 242
  - online algorithm, **241**
  - shortest-queue algorithm, **241**
- machine word, 23, 24
- Maggs, B. M., 125
- makespan, 241
- map coloring, 255
- Markov, A., 48
- Markov’s inequality, *see under* inequality
- Martello, S., 233
- Martinez, C., 124

- master theorem, *see under* algorithm  
     analysis  
 mating, 260  
 Matousek, J., 262  
 matrix, 171  
 matrix products, chained, 245  
 Mauer, D., 58  
 maximization problem, **233**  
 maximum flow, 237  
 McCreight, E. M., 163  
 McGeoch, L. A., 257  
 McIlroy, M. D., 124  
 median, 114, *see also* selection, **265**  
 Mehlhorn, K., 79, 97, 165, 166, 189, 201, 209, 215, 229  
 Mehnert, J., 166  
 member variable, **31**  
*memcpy*, 78  
 memory access, 24  
 memory block, **25**  
 memory cell, 23, *see also* machine word  
 memory management, **27**  
 memory size, 24  
 Mendelson, R., 143  
 mergesort, *see under* sorting  
 merging, 103, 244  
     external, 119  
     multiway, **119**  
 Meyer auf der Heide, F., 97  
 Meyer, B., 56  
 Meyer, U., 189, 205, 214  
 Michel, L., 262  
 minimization problem, **233**  
 minimum edit distance, 245  
 minimum spanning forest, *see*  
     MST  
 minimum spanning tree, *see*  
     MST  
 mobile device, 25  
**mod**, **24**  
 modulo, 7, **264**  
 Monte Carlo algorithm, *see under*  
     algorithm design, randomized  
 Moret, B., 231  
 Morris, R., 97  
 most significant distinguishing index, 202  
 move-to-front, 44  
 msd, *see* most significant distinguishing  
     index  
 MST, **217**  
     2-approximation of TSP, 230  
     Boruvka's algorithm, 231  
     clustering, 217, 232  
     cut property, **218**, 221  
     cycle property, 219, 221, 232  
     Euclidean, 232  
     external memory, 225  
     Held–Karp lower bound, 230  
     Jarník–Prim algorithm, **219**  
     maximum-cost spanning tree, 218  
     parallel, 232  
     semiexternal Kruskal algorithm, 226  
     streaming algorithm, 222  
     uniqueness conditions, 219  
     use of, 217, 228, 232  
 multicore processor, **25**  
 multigraph, 167, 173  
 multikey quicksort, 113  
 multiplication (integer)  
     Karatsuba, **9**  
     refined, 12  
     recursive, **7**  
     school method, 1, **3**  
     use of, 1  
 multithreading, **25**  
 mutation, 259  
  
 Näher, S., 166, 171  
 Nemhauser, G., 244, 248  
 network, **25**, *see also* graph  
     communication network, 49  
     design, 217  
 Neubert, K. S., 125  
 Nilsson, S., 125  
 node, **49**  
     active, 178  
     associated info., 167  
     depth, 52, 176  
     *dfsNum*, 178  
     finishing time, 178  
     interior, **52**  
     marked, 178  
     numbering, 167  
     ordering relation ( $\prec$ ), 179  
     potential, **207**, 211, 230  
     reached, 176, 197  
     representative, 177, 182  
     scanned, 196

- NodeArray*, **168**, 173  
 Noshita, K., 200  
 NOT, **24**  
 NP, **53**  
 NP-complete, **54**  
 NP-hard, **55**, 238  
 numeric type, 26
- $O(\cdot)$ , **21**  
 $o(\cdot)$ , **21**  
 object, 26  
 object-oriented, **31**  
 objective function, **233**  
**of** (in type declaration), 26, 27, 31  
 Ofman, Y., 9  
 $\Omega(\cdot)$ , **21**  
 $\omega(\cdot)$ , **21**  
 online algorithm, 44, **241**  
 optimization, **233**  
 optimization problem, 56, **233**  
 OR, **24**  
 Orlin, J., 201  
 oversampling, 121
- P**, **53**  
 Pagh, R., 97  
 pair, **27**  
 pairing heap, *see under* priority queue  
 parallel assignment, 28  
 parallel processing, 24, **25**, 121, 214, 232, 259, 262  
 parameter, **29**  
   actual, 29  
   formal, 29  
 parameterized class, **31**  
 parent, **52**  
 Pareto, V., 244  
 Pareto-optimal, 244, 261  
 parser, 53  
 partition, 222  
 Pascal, 26  
 Patashnik, O., 40, 58  
 path, **50**  
   simple, **50**  
 Perl, 81  
 permutation, 42, 100, 101, 106  
   random, 42, 45  
 persistent data structure, 166  
 Peru, 59
- Peterson, W. W., 90  
 Petrank, E., 97  
 Pettie, S., 143, 232  
 Pferschy, U., 233  
 pigeonhole principle, 242  
 pipelining, 4  
 Pisinger, D., 233  
 pivot, **108**, 121  
   selection, 111, 124  
 Plaxton, C. G., 125  
 pointer, **26**  
 polynomial, 22, 101, *see also under*  
   running time  
 polytope, 251  
 population, 259  
 postcondition, **32**  
 potential function, *see* node, potential  
 powers (of numbers), 32  
 Pratt, V. R., 124  
 precedence relation, 49  
 precondition, **32**  
 predecessor, 60, **60**  
 Priebe, V., 209  
 Prim, R. C., 219  
 Prim's algorithm, *see* MST, Jarník–Prim  
   algorithm  
 prime number, 31, 86, 101, **265**  
   abundance, 88  
 primitive operation  
   full adder, 1  
   product, 2  
 principle of optimality, **243**, 246  
 priority queue, **127**  
   addressable, **128**, 133, 198  
   binary heap, **129**, 199  
   addressable, 129, 133  
   bottom-up *deleteMin*, 142  
   building, 131  
   bulk insertion, 133  
   *deleteMin*, **131**  
   *insert*, **130**  
   invariant, 129  
   *siftDown*, **131**  
   *siftUp*, **130**  
   binomial heap, 137  
   bounded, 129  
   bucket, 143  
   bucket queue, **201**  
   invariant, 202

- calendar queue, 143
- decrease key, **128**, 199
- deleteMin*, **127**
- double-ended, 156
- external, 139
- fat heap, 143
- Fibonacci heap, **135**, 199, *see also*
  - priority queue, heap-ordered forest
  - decreaseKey*, **138**
  - deleteMin*, **136**
  - item, **136**
  - rank, **136**
- heap-ordered forest, **133**
  - cut, **133**
  - decreaseKey*, **133**
  - deleteMin*, **133**
  - insert*, **133**
  - invariant, 133
  - link, **133**
  - merge*, **135**
  - new tree, **133**
  - remove*, **135**
- insert*, **128**
- integer, 142, 143, **201**
- item, 133
- memory management, 141
- merge*, **128**
- minimum, **127**, 130, 133
- monotone, 128, 143, 198, **201**
- naive, 129, 199
- pairing heap, **135**, *see also*
  - priority q., heap-ordered forest
  - complexity, 143
  - three-pointer items, 135
  - two-pointer items, 135
- radix heap, **201**
  - base *b*, 204
  - remove*, **128**
- thin heap, 143
- unbounded, 129
- use of, 102, 120, 125, 128, 198, 226
- probability, **266**
- probability space, 41, **266**
- problem instance, 20
- procedure, **29**
- profit vector, *see* cost vector
- program, **24**
- program analysis, *see* algorithm analysis
- programming language, 26, 28, 58
  - functional, 105
  - logical, 105
- programming model, *see* machine model
- Prokop, H., 142
- pseudo-polynomial algorithm, 245
- pseudocode, **26**, 56
- Puget, J.-F., 262
- Pugh, W., 165
- quartile, 114, *see also* selection
- queue, 27, 170, *see also* FIFO
- quickselect, *see under* selection
- quicksort, *see under* sorting
- quipu, **59**
- radix sort, *see under* sorting
- Radzik, T., 214
- RAM model, *see under* machine model
- Ramachandran, S., 142
- Ramachandran, V., 232
- Raman, R., 125
- Ranade, A., 123
- random experiment, **266**
- random number, **46**
- random source, 57
- random variable, 41, **266**
  - independent, **268**
  - indicator, **266**
  - product, **268**
- randomized algorithm, *see under*
  - algorithm design; algorithm analysis
- rank, 103, **265**
- Rao, S., 215
- realloc*, 78
- recombination, 259, 260
- record, *see* composite type
- recurrence relation, 9, 16, 35, **37**, 58
- recursion, **29**, *see also under*
  - algorithm design; algorithm analysis
  - elimination, 113, 141
- red-black tree, *see under* sorted sequence
- reduction, **55**
- reflexive, **265**
- register, 24, **24**, 25
- Reif, J., 189
- relation, **265**
  - antisymmetric, **264**
  - equivalence, **265**



- reflexive, **265**
- symmetric, **265**
- transitive, **265**
- weakly antisymmetric, **265**
- relaxation, 256, *see also under*
  - linear program
- remainder, **24**
- Remez, O., 174
- removing from a sequence, 60
- repeat, 28**
- result checking, *see under* algorithm design
- return, 29**
- Rivest, R. L., 124
- road map, 49
- Robertson, N., 255
- Robins, G., 229
- Rodeh, M., 165
- root, *see under* tree
- Roura, S., 124
- run, *see under* sorting
- running time, 20, **24**, 28, 36, *see also*
  - algorithm analysis
  - average case, **20**, 41
  - best case, **20**, 24
  - polynomial, **53**
  - worst case, **20**
- sample space, **266**
- Sanders, D. P., 255
- Sanders, P., 124, 125, 141, 142, 166, 212, 214, 215, 225, 232
- Santos, R., 174
- Sarnak, N., 166
- SAT solver, 242
- satisfiability problem, **54**
- satisfiable, 242
- Schäfer, G., 209
- Schaffer, R., 142
- scheduling, 128, 191, **241**
- Schevon, C., 257
- Schönhage, A., 18
- Schrijver, A., 262
- Schultes, D., 212, 225
- search tree, *see* sorted sequence
- searching, **145**, *see also* sorted sequence
  - binary search, **34**, 56, 100, 121, 151
  - dynamic, **43**
  - exponential, 35, 56
  - linear, 43
  - range, 100
  - shortest path, *see under* shortest path
- Sedgewick, R., 40, 125, 142, 143
- Seidel, R., 165, 174, 224
- selection, **114**
  - deterministic, 124
  - quickselect, **114**
  - streaming, **115**
- self-loop, **49**
- semicolon (in pseudocode), 28
- sentinel, **63**, 95, 102, 106, 141
- sequence, 27, **27**, 59, 100
  - overview of operations, 77
  - space efficiency, 77
- series, *see* sum
- server, 25
- set, **27**
- set covering, 239
- Seymour, P., 255
- Shapiro, H. D., 231
- shared memory, **25**
- Sharir, M., 189
- Shell sort, *see under* sorting
- Sheperdson, J., 23
- shift, **24**
- Shmoys, D. B., 230
- shortest path, **191**
  - acyclic, 192
  - ALD (average linear Dijkstra), **205**, 214
  - all-pairs, 191
  - all-pairs with negative costs, 207
  - arbitrary edge costs, 206
  - as a linear program, 236
  - A\*-search, 211
  - Bellman–Ford algorithm, 206
    - refined, 214
  - bidirectional search, 209
  - bottleneck, 217, 232
  - by table lookup, 212
  - constrained, 215, 246
  - correctness criterion, 194
  - DAG, 195
  - Dijkstra’s algorithm, **196**
    - invariant, 201
  - edge relaxation, **194**
  - geometric, 215
  - goal-directed search, 211
  - hierarchical search, 212
  - integer edge cost, 192

- linear average time, 205
- multicriteria, 215
- negative cycle, 192
- nonnegative edge cost, 192
- parallel, 214
- parent pointer, 193
- public transportation, 196
- query, 209
- relaxing of edges, **194**
- single-source, 191
- subpath, 193
- tentative distance, 194
- transit node routing, 212
- tree, 193
- uniqueness, 193
- unit edge cost, 192
- use of, 191, 207
- shortest-queue algorithm, 241
- shrunk graph, 182
- Sibeyn, J., 225
- sibling, **52**
- sibling pointer, 136
- Siek, J. G., 173
- sieve of Eratosthenes, 31
- SIMD, **25**, 95
- simplex algorithm, *see under*
  - linear programming
- simulated annealing, *see under*
  - algorithm design, local search
- Singler, J., 124
- Sipser, M., 54
- Sivadasan, N., 209
- Sleator, D., 79, 143, 165, 166, 222
- slow memory, 25
- Smith, S. J., 125
- snow plow heuristic, 125
- solution
  - feasible, **233**
  - potential, **233**
- sorted sequence, 34, **145**
  - $(a, b)$ -tree, **149**
    - split* (node), **152**
  - amortized update cost, **158**
  - augmentation, **160**
  - balance*, **153**
  - build/rebuild*, **156**
  - concatenation, **157**
  - fusing, **153**
  - height, **150**
  - insert*, **151**
  - invariant, **149**
  - item, **150**
  - locate*, **150**
  - parent pointer, **161**
  - reduction, **161**
  - remove*, **153**
  - removing a range, **158**
  - splitter, **149**
  - splitting, **157**
- adaptable, 165
- AVL tree, 165
- binary search tree, **147**
  - degenerate, 148
  - expected height, 148
  - implicit, 149
  - insert*, 147
  - locate*, **147**
  - perfect balance, 147
  - rotation, 149
  - selection, 161
- cache-oblivious, 165
- finger search, 161
- first*, **146**, 156
- insert*, **145**
- integer, 166
- last*, **146**, 156
- locate*, **145**, 146
- merging, 161
- navigation, 145
- persistent, 166
- pred*, **146**
- randomized search tree, 165
- range searching, 156
- red-black tree, 155, 164
- remove*, **145**
- skip list, 165
- sparse table, 165
- splay tree, 165
- strings, 166
- succ*, **146**
- trie, 166
- use of, **146**, **147**
- weight-balanced tree, 160, 165
- sorting, **99**
  - almost sorted inputs, 103
  - bottom-up heapsort, 142
  - bucket, **116**
  - comparison-based, 116

- dynamic, 102
- external, **118**
- flash, 125
- heapsort, 128, **132**
- in-place, **101**, 111
- insertion, 36, **102**, 105
- large elements, 123
- list, 105
- lower bound, 116
- mechanical, 99
- mergesort, **103**, 124
- multiway merge, **119**
- numbers, 116, **116**, 122, 170
- parallel, 121, 125
- parallel disks, 125
- quicksort, **108**, 123, 124, 148
- radix, **116**
  - LSD, **116**
  - MSD, **117**, 123
- random numbers, **117**
- run formation, 119, 125
- sample, **120**, 125
- selection, **101**, 128
- Shell sort, 125
- small inputs, 102, 108
- small subproblems, 111
- stable algorithm, **116**
- strings, 113, 116
  - use of, 34, 99–101, 125, 172, 226, 239
  - word model, 125
- source node, **49**
- spellchecking, 125
- Spielmann, D., 262
- splitter, 121, 147
- stack, 27, 29, 74, 75
  - bounded, 75
  - external-memory, 76
  - pop, **74**
  - push, **74**
  - top, **74**
  - unbounded, 75
- statement, 28
- static array, **27**, 59
- statistics, 114
- Stirling's approximation, 107, 118, **270**
- STL, 13, 57, 164
  - deque*, 78
  - hash\_map*, 96
  - hash\_multimap*, 96
  - hash\_multiset*, 96
  - hash\_set*, 96
  - iterator, 78, 123
  - list*, 78
  - map*, 164
  - multimap*, 164
  - multiset*, 164
  - priority\_queue*, 142
  - set*, 164
  - sort*, 123
  - stack*, 78
- store instruction, **24**
- Strassen, V., 18
- streaming algorithm, 115, 222
- string, 27, 59, 100
- striping, 125
- struct, *see* composite type
- Sturgis, H., 23
- STXXL, 124, 141, 142
- subroutine, **29**
- successor, 60, **60**
- succinct data structure, 97
- Sudoku, 255
- sum, 58, *see also under* algorithm analysis
  - estimation by integral, 271
  - geometric, 38, **270**
  - harmonic, 43, 45, 88, 110, 200, 228, 264, **270**
- Sumerian, 59
- survival of the fittest, 259
- swap, 28
- sweep-line algorithm, 146
- symmetric, **265**
- syntax, 26
- Szemerédi, E., 97
- table, 59
- tablet, 59
- tabu list, *see* tabu search
- tabu search, *see under* algorithm design,
  - local search
- tail bound, **269**
- tail recursion, *see* recursion, elimination
- Tamassia, R., 174
- Tardos, E., 97
- target node, **49**
- Tarjan, R. E., 79, 97, 124, 135, 143, 165, 166, 189, 201, 214, 222, 224, 232
- telephone book, 99

- template programming, 31, 123
- Teng, S. H., 262
- termination, 33, 34
- $\Theta(\cdot)$ , 21
- Thomas, R., 255
- Thompson, K., 246
- Thorup, M., 95, 125, 143, 214
- thread, 25
- threshold acceptance, *see under*
  - algorithm design, local search
- time, *see* running time
- time step, 24
- Toom, A., 18
- total order, 99, 265, 265
- Toth, P., 233
- tournament tree, 125
- Tower of Hanoi, 75
- Träff, J. L., 232
- transitive, 265
- translation, 27–30
- traveling salesman problem, 54, 55, 56, 230
  - 2-exchange, 250
  - 3-exchange, 250
  - Held–Karp lower bound, 230
  - hill climbing, 250
- tree, 51, 147
  - depth, 52
  - dynamic, 222
  - expression tree, 53
  - height, 52
  - implicitly defined, 129
  - interior node, 52
  - ordered, 53
  - representation, 136
  - root, 52
  - sorting tree, 106
  - traversal, 53
- triangle inequality, 230, 250
- trie, *see under* sorted sequence
- triple, 27
- true*, 24
- truth value, 24
- Tsitsiklis, J. N., 262
- TSP, *see* traveling salesman problem
- tuple, 27, 100
- type, 26
- Udupa, R., 123
- Ullman, J., 165
- Ullmann, Z., 244
- unary operation, 24
- unbounded array, 60, 66
- undefined value ( $\perp$ ), 26
- uniform memory, 23
- union–find, 222
  - path compression, 223
  - union by rank, 223
- universe ( $\mathcal{U}$ ), 233
- upper bound, *see* worst case
- Vöcking, B., 245
- van Emde Boas layout, 165
- van Emde Boas, P., 166
- Van Hentenryck, P., 262
- Vanderbei, R. J., 262
- variable, 26, 235
- Vazirani, V., 232
- vector (in C++), 78
- verification, 32, 103
- vertex, *see* node
- Vishkin, U., 189
- visitor, *see under* graph
- Vitányi, P., 125
- Vitter, J. S., 120, 123
- von Neumann, J., 23
- von Neumann machine, *see under*
  - machine model
- Vuillemin, J., 137
- Vygen, J., 232
- weakly antisymmetric, 265
- Wegener, I., 54, 142
- Wegman, M., 97
- Weidling, C., 97
- Westbrook, J., 232
- while**, 28
- Wickremsinghe, R., 123
- Wilhelm, R., 58
- Williams, J. W. J., 129
- Winkel, S., 125, 142
- witness, *see* algorithm design, certificate
- Wolsey, L., 248
- word, *see* machine word
- worst case, *see under* running time
- XOR ( $\oplus$ ), 24, 203
- Zagha, M., 125

Zang, I., 215  
Zelikowski, A., 229  
Ziegelmann, M., 215  
Ziviani, N., 97

Zlotowski, O., 171  
Zwick, U., 143, 232

PRELIMINARY COPY