Bibliography

[1] T. Akenine-Möller et al. *Real-Time Rendering*. 4th ed. Boca Raton, FL, USA: A K Peters/CRC Press, 2018 (cit. on p. 86).

[2] J. Alber and R. Niedermeier. "On Multidimensional Curves with Hilbert Property". In: *Theory Comput. Systems* 33 (2000), pp. 295–312 (cit. on p. 86).

[3] M. H. Albert, R. J. Nowakowski, and D. Wolfe. *Lessons in Play. Introduction to Combinatorial Game Theory*. 2nd ed. CRC Press, 2019 (cit. on p. 147).

[4] K. Appel and W. Haken. "Every Planar Map is Four Colorable. Part I: Discharging". In: *Illinois J. Math.* 21(3) (1977), pp. 429–490 (cit. on p. 173).

[5] K. Appel, W. Haken, and J. Koch. "Every Planar Map is Four Colorable. Part II: Reducibility". In: *Illinois J. Math.* 21(3) (1977), pp. 491–567 (cit. on p. 173).

[6] J. Arndt and C. Haenel. *Pi Unleashed*. Springer, 2001 (cit. on p. 330).

[7] M. D. Atkinson. "The Cyclic Towers of Hanoi". In: *Information Processing Letters* 13(3) (Dec. 1981), pp. 118–119 (cit. on p. 63).

[8] M. Bader. *Space-Filling Curves. An Introduction with Applications in Scientific Computing*. Springer, 2013 (cit. on p. 86).

[9] D. H. Bailey et al. "The Quest for Pi". In: *Mathematical Intelligencer* 19(1) (Jan. 1997), pp. 50–56 (cit. on p. 330).

[10] M. Barnsley. *Fractals Everywhere*. 2nd ed. Academic Press, 2014 (cit. on p. 21).

[11] J. Beasley. *The Delights of Peg Solitaire*. 2021. URL: `http://www.jsbeasley.co.uk/delights.htm` (cit. on p. 146).

[12] G. I. Bell. *Notes on Solving and Playing Peg Solitaire on a Computer*. 2009. URL: `https://arxiv.org/abs/0903.3696` (cit. on p. 146).

[13] T. Bell and D. Kulp. "Longest-Match String Searching for Ziv-Lempel Compression". In: *Software: Practice and Experience* 23(7) (July 1993), pp. 757–771 (cit. on p. 273).

[14] A. Benjamin, G. Chartrand, and P. Zhang. *The Fascinating World of Graph Theory*. Princeton University Press, 2017 (cit. on p. 173).

[15] E. R. Berlekamp, J. R. Conway, and R. K. Guy. *Winning Ways for your Mathematical Plays*. 2nd ed. Vol. 4. CRC Press, 2004 (cit. on p. 146).

[16] J. F. Blinn. "What Is a Pixel?" In: *IEEE Comput. Graph. Appl.* 25(5) (Sept. 2005), pp. 82–87 (cit. on p. 44).

[17] J. Bloch. *Effective Java*. 3rd ed. Addison-Wesley Professional, Dec. 2017 (cit. on p. 339).

[18] P. Bourke. *Fractals, Chaos, Self-Similarity*. URL: `http://paulbourke.net/fractals/` (visited on 02/03/2026) (cit. on p. 43).

[19] C. L. Bouton. "Nim, a Game with a Complete Mathematical Theory". In: *Annals of Mathematics* 3(1/4) (1901), pp. 35–39 (cit. on p. 147).

[20] R. P. Brent and P. Zimmerman. *Modern Computer Arithmetic*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2010 (cit. on p. 328).

[21] C. Browne. "Bitboard Methods for Games". In: *ICGA Journal* 37(2) (June 2014), pp. 67–84 (cit. on p. 146).

[22] R. E. Bryant and D. R. O'Hallaron. *Computer Systems: A Programmer's Perspective*. 3rd ed. Pearson, 2015 (cit. on pp. 122, 246).

[23] T. C. Chen and I. T. Ho. "Storage-Efficient Representation of Decimal Data". In: *CACM* 18(1) (Jan. 1975), pp. 49–52 (cit. on p. 436).

[24] S. A. Cook. "On the Minimum Computation Time of Functions". PhD thesis. Harvard University, Dept. of Mathematics, 1966 (cit. on p. 328).

[25] T. H. Cormen et al. *Introduction to Algorithms*. 4th ed. MIT Press, 2022 (cit. on pp. 173, 208, 293, 320).

[26] M. Cowlishaw. "Densely Packed Decimal Encoding". In: *IEE Proceedings – Computers and Digital Techniques* 149(3) (May 2002), pp. 102–104 (cit. on p. 436).

[27] J. Culberson. *Sokoban is PSPACE-Complete*. Tech. rep. TR97-02. University of Alberta, 1997 (cit. on p. 209).

[28] L. P. Deutsch. *DEFLATE Compressed Data Format Specification*. RFC 1951. Version 1.3. May 1996. URL: `https://www.ietf.org/rfc/rfc1951.txt` (cit. on p. 273).

[29] A. K. Dewdney. "Computer Recreations. A Computer Microscope Zooms In for a Look at the Most Complex Object in Mathematics". In: *Scientific American* (Aug. 1985), pp. 16–24 (cit. on p. 43).

[30] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1 (1959), pp. 269–271 (cit. on p. 208).

[31] S. Draves and E. Reckase. *The Fractal Flame Algorithm*. 2008. URL: `https://flam3.com/flame_draves.pdf` (visited on 02/03/2026) (cit. on p. 21).

[32] J. Duda et al. "The Use of Asymmetric Numeral Systems as an Accurate Replacement for Huffman coding". In: *2015 Picture Coding Symposium (PCS)*. 2015, pp. 65–69 (cit. on p. 236).

[33] L. Euler. “Solutio problematis ad geometriam situs pertinentis”. In: *Commentarii Academiae Scientiarum Imperialis Petropolitanae* 8 (1741), pp. 128–140 (cit. on p. 172).

[34] G. Farin. “A History of Curves and Surfaces in CAGD”. In: *Handbook of Computer Aided Geometric Design*. Ed. by G. Farin, J. Hoschek, and M.-S. Kim. Elsevier, 2002, pp. 1–22 (cit. on p. 86).

[35] G. Farin. *Curves and Surfaces for CAGD. A Practical Guide*. 5th ed. Morgan Kaufmann, 2002 (cit. on p. 86).

[36] F. A. Farris. *Creating Symmetry. The Artful Mathematics of Wallpaper Patterns*. Princeton, NJ, USA: Princeton University Press, 2015 (cit. on p. 84).

[37] G. Flegg. *Numbers: Their History and Meaning*. Dover, 2002 (cit. on p. 328).

[38] E. Gamma et al. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994 (cit. on p. 183).

[39] M. Gardner. “Bouncing Balls in Polygons and Polyhedrons”. In: *Martin Gardner’s 6th Book of Mathematical Diversions*. University of Chicago Press, 1983. Chap. 4, pp. 29–38 (cit. on p. 174).

[40] M. Gardner. “Peg Solitaire”. In: *The Unexpected Hanging and Other Mathematical Diversions*. University of Chicago Press, 1991. Chap. 11, pp. 122–135 (cit. on pp. 125, 146).

[41] M. Gardner. *Penrose Tiles to Trapdoor Ciphers. …and the Return of Dr. Matrix*. Revised. The Mathematical Association of America, 1997 (cit. on p. 87).

[42] M. Gardner. “The Fantastic Combinations of John Conway’s New Solitaire Game “Life””. In: *Scientific American* 223(4) (Oct. 1970), pp. 120–123 (cit. on p. 44).

[43] C. Gotsman and M. Lindenbaum. “On the Metric Kroperties of Discrete Space-Filling Curves”. In: *IEEE Trans. Image Process.* 5(5) (May 1996), pp. 794–797 (cit. on p. 86).

[44] T. Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*. URL: `http://gmplib.org/` (cit. on p. 328).

[45] B. Grünbaum and G. C Shephard. *Tilings & Patterns*. 2nd ed. Dover, 2016 (cit. on p. 87).

[46] J. Hadley and D. Singmaster. “Problems to Sharpen the Young”. In: *The Mathematical Gazette* 76(475) (Mar. 1992), pp. 102–126 (cit. on p. 173).

[47] P. E. Hart, N. J. Nilsson, and B. Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Trans. Syst. Sci. Cybern.* 4(2) (July 1968), pp. 100–107 (cit. on p. 209).

[48] D. Harvey and J. Van Der Hoeven. “Integer Multiplication in Time O(n log n)”. In: *Ann. of Math.* 193(2) (Mar. 2021), pp. 563–617 (cit. on p. 329).

[49] R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. A K Peters, 2009 (cit. on p. 209).

[50] D. Hilbert. “Ueber die stetige Abbildung einer Linie auf ein Flächenstück”. In: *Mathematische Annalen* 38 (Sept. 1891), pp. 459–460 (cit. on p. 86).

[51] A. M. Hinz, S. Klavžar, and C. Petr. *The Tower of Hanoi – Myths and Maths*. 2nd ed. Cham: Birkhäuser, 2018 (cit. on p. 63).

[52] J. Hoppe. *Light Bulb Jokes*. 2012. URL: https://home.c-c-g.de/index.php/humor/120-light-bulb-jokes (visited on 02/04/2026) (cit. on p. 353).

[53] D. A. Huffman. "A Method for the Construction of Minimum-Redundancy Codes". In: *Proceedings of the IRE* 40(9) (Sept. 1952), pp. 1098–1101 (cit. on p. 236).

[54] J. F. Hughes et al. *Computer Graphics. Principles and Practice*. 3rd ed. Addison-Wesley, 2013 (cit. on p. 63).

[55] N. Johnston and D. Green. *Conway's Game of Life. Mathematics and Construction*. 2022. URL: https://conwaylife.com/book/ (visited on 02/05/2026) (cit. on p. 44).

[56] A. Junghanns. "Pushing the Limits: New Developments in Single-Agent Search". PhD thesis. University of Alberta, 1999 (cit. on p. 209).

[57] A. Karatsuba and Y. Ofman. "Multiplication of Many-Digital Numbers by Automatic Computers". In: *Dokl. Akad. Nauk SSSR* 145(2) (1962), pp. 293–294 (cit. on p. 328).

[58] B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. 2nd ed. Prentice Hall, 1988 (cit. on p. 381).

[59] E. Klarreich. "Multiplication Hits the Speed Limit". In: *Commun. ACM* 63(1) (Dec. 2019), pp. 11–13. URL: https://doi.org/10.1145/3371387 (cit. on p. 329).

[60] D. E. Knuth. *The Art of Computer Programming. Seminumerical Algorithms*. 3rd ed. Vol. 2. Addison-Wesley, 1997 (cit. on pp. 304, 328, 381).

[61] D. E. Knuth. *The Art of Computer Programming. Combinatorial Algorithms, Part 1*. Vol. 4A. Addison-Wesley, 2011 (cit. on p. 122).

[62] D. E. Knuth. *The Art of Computer Programming. Combinatorial Algorithms, Part 2*. Vol. 4B. Addison-Wesley, 2023 (cit. on p. 148).

[63] D. E. Knuth. *The Stanford GraphBase. A Platform for Combinatorial Computing*. Addison-Wesley, 1993 (cit. on p. 173).

[64] D. H. Larkin, S. Sen, and R. E. Tarjan. "A Back-to-Basics Empirical Study of Priority Queues". In: *Proceedings of the Meeting on Algorithm Engineering & Expermiments*. Portland, Oregon: Society for Industrial and Applied Mathematics, 2014, pp. 61–72 (cit. on p. 208).

[65] A. Lempel and J. Ziv. "A Universal Algorithm for Sequential Data Compression". In: *IEEE Transactions on Information Theory* 23(3) (May 1977), pp. 337–343 (cit. on p. 273).

[66] A. Levitin and M. Levitin. *Algorithmic Puzzles*. Oxford University Press, 2011 (cit. on p. 173).

[67] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. URL: http://www.inference.org.uk/mackay/itila/ (visited on 02/05/2026) (cit. on pp. 235, 274).

[68] B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Co., 1983 (cit. on p. 43).

[69] S. Marschner and P. Shirley. *Fundamentals of Computer Graphics*. 5th ed. A K Peters/CRC Press, 2021 (cit. on p. 63).

[70] A. Moffat. "Huffman Coding". In: *ACM Comput. Surv.* 52(4) (2019), 85:1–85:35. URL: https://doi.org/10.1145/3342555 (cit. on p. 236).

[71] J.-M. Muller et al. *Handbook of Floating-Point Arithmetic*. 2nd ed. Birkhäuser Boston, 2018 (cit. on p. 328).

[72] P. Norvig. *Solving Every Sudoku Puzzle*. URL: http://www.norvig.com/sudoku.html (visited on 12/17/2022) (cit. on p. 148).

[73] Numberphile. *Why 381,654,729 is Awesome*. May 2013. URL: https://www.youtube.com/watch?v=gaVMrqzb91w (visited on 10/11/2021) (cit. on p. 452).

[74] M. L. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. Society for Industrial and Applied Mathematics, 2001 (cit. on p. 328).

[75] G. Peano. "Sur une courbe, qui remplit toute une aire plane". In: *Mathematische Annalen* 36(1) (1890), pp. 157–160 (cit. on p. 86).

[76] H.-O. Peitgen, H. Jürgens, and D. Saupe. *Chaos and Fractals. New Frontiers of Science*. 2nd ed. Springer, 2004 (cit. on pp. 21, 43).

[77] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. 4th ed. MIT Press, 2023. URL: https://www.pbr-book.org/ (visited on 02/05/2026) (cit. on p. 44).

[78] I. Pressman and D. Singmaster. ""The Jealous Husbands" and "The Missionaries and Cannibals"". In: *The Mathematical Gazette* 73(464) (June 1989), pp. 73–81 (cit. on pp. 173, 407).

[79] Quote Investigator. *Insanity Is Doing the Same Thing Over and Over Again and Expecting Different Results*. Mar. 2017. URL: https://quoteinvestigator.com/2017/03/23/same/ (visited on 10/11/2021) (cit. on p. 1).

[80] J. Rissanen and G. G. Langdon Jr. "Arithmetic Coding". In: *IBM J. Res. Develop.* 23(2) (Mar. 1979) (cit. on p. 236).

[81] L. Rougetet. "Machines Designed to Play Nim Games (1940–1970): A Possible (Re)Use in the Modern French Mathematics Curriculum?" In: *Teaching and Learning Discrete Mathematics Worldwide: Curriculum and Research*. Ed. by E. W. Hart and J. Sandefur. Cham: Springer International Publishing, 2018, pp. 229–250. URL: https://doi.org/10.1007/978-3-319-70308-4_15 (cit. on p. 147).

[82] K. Sadakane and H. Imai. "Improving the Speed of LZ77 Compression by Hashing and Suffix Sorting". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E83-A(12) (Dec. 2000), pp. 2689–2698 (cit. on p. 273).

[83] D. Salomon. *Data Compression: The Complete Reference*. 4th ed. With contributions by Giovanni Motta and David Bryant. Springer-Verlag, 2007 (cit. on p. 273).

[84] A. A. Sardinas and G. W. Patterson. “A Necessary and Sufficient Condition for Unique Decomposition of Coded Messages”. In: *Convention Record of the I.R.E., 1953 National Convention, Part 8: Information Theory* (1953), pp. 104–108 (cit. on p. 236).

[85] K. Sayood. *Introduction to Data Compression*. 5th ed. Morgan Kaufmann, 2018 (cit. on p. 273).

[86] P. Schneider and D. H. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, 2002 (cit. on p. 63).

[87] R. Sedgewick and K. Wayne. *Algorithms*. 4th ed. Addison-Wesley, 2011 (cit. on pp. 173, 208, 225, 293).

[88] C. E. Shannon. “A Mathematical Theory of Communication”. In: 27(3) (July 1948), pp. 379–423 (cit. on p. 235).

[89] C. E. Shannon. “A Mathematical Theory of Communication”. In: 27(4) (Oct. 1948), pp. 623–656 (cit. on p. 235).

[90] M. Sipser. *Introduction to the Theory of Computation*. 3rd ed. Cengage Learning, 2013. Boston, MA (cit. on p. 209).

[91] J. A. Storer and T. G. Szymanski. “Data Compression via Textual Substitution”. In: *Journal of the ACM* 29(4) (Oct. 1982), pp. 928–951 (cit. on p. 273).

[92] *The Sokoban Wiki*. 2024. URL: `http://sokobano.de/wiki/index.php?title=Main_Page` (visited on 02/09/2026) (cit. on p. 209).

[93] A. L. Toom. “The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers”. Russian. In: *Dokl. Akad. Nauk SSSR* 150 (1963), pp. 496–498 (cit. on p. 328).

[94] M. C. K. Tweedie. “A Graphical Method of Solving Tartaglian Measuring Puzzles”. In: *The Mathematical Gazette* 23(255) (July 1939), pp. 278–282 (cit. on p. 174).

[95] H. S. Warren Jr. *Hacker’s Delight*. 2nd ed. Addison-Wesley, 2013 (cit. on p. 122).

[96] P. Wegner. “A Technique for Counting Ones in a Binary Computer”. In: *Communications of the ACM* 3(5) (May 1960), p. 322 (cit. on p. 382).

[97] T. A. Welch. “A Technique for High-Performance Data Compression”. In: *Computer* 17(6) (June 1984), pp. 8–19 (cit. on p. 273).

[98] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002. URL: `https://www.wolframscience.com/nks/` (cit. on p. 44).

[99] A. J. Yee. *y-cruncher - A Multi-Threaded Pi-Program*. URL: `http://www.numberworld.org/y-cruncher/` (visited on 03/09/2026) (cit. on p. 330).

[100] J. Ziv and A. Lempel. “Compression of Individual Sequences via Variable-Rate Coding”. In: *IEEE Transactions on Information Theory* 24(5) (Sept. 1978), pp. 530–536 (cit. on p. 273).

Index of Identifiers

A

abs(), in BigInt, 314, **315**
abs(), in Duration, **455**, 458
AbstractCollection, 118
add(), in BitSet, 118, 182, **385**, 418
addAll(), in BitSet, 119, 184, **388**
addDigits(), in Duration, 456, **457**, 458
addWord(), in SimilarWords, 205, **422**
advance(), in GameOfLife, 362, **363**
advanceInput(), in LZTokenizer, 250, 255, **256**
Affine, **11**, 11–13, 15, 51, 53–58, 359
  Affine(), **11**, 13, 53, 55
  compose(), **55**, 55, 56
  IDENTITY, **53**, 55, 58
  rotation(), **53**, 53, 55, 56
  scaling(), **53**, 53, 55, 56
  times(), **55**, 55, 58
  transformPoint(), **12**, 12, 15
  transformPoints(), **12**, 12, 54, 57, 58
  translation(), **53**, 53, 55, 56
Affine(), in Affine, **11**, 13, 53, 55
AffineIFS, 12, **13**, 13–15, 359
  AffineIFS(), 12, **13**, 13
  BARNSLEY_FERN, **13**
  chooseIndex(), 14, **15**
  extendedChaosGame(), 14, **15**
  probabilities, 12, **13**
  transforms, 12, **13**
AffineIFS(), in AffineIFS, 12, **13**, 13
ALL_DIRECTIONS, in Board, **179**, 188, 196, 420, 427
allCratesOnGoals(), in GameState, 182, **183**, 183, 190, 203
angle(), in Point, 18, **356**
appendDigit(), in BigNat, **307**, 308
arctan1overX(), in PiMachin, **459**, 459
ArithmeticException, 106, 316, 451, 452, 457
ArrayDeque, 157, 158, 167, 254, 413, 415
ArrayList, 6, 12, 15, 58, 77, 112, 113, 115, 154, 161, 166, 188, 196, 254, 337, 343–345, 350, 351, 365, 371, 378, 397, 401, 409, 411, 418, 420–422, 425, 427, 440
Arrays, 279, 343, 385, 386, 395, 400, 405, 406, 456
Ascii, 428, 429
  encryptCaesar(), 428, **429**
  isDigit(), **428**
  isLower(), **428**
  isUpper(), **428**
  toLower(), **428**
  toUpper(), **428**

B

BARNSLEY_FERN, in AffineIFS, **13**
BASE, in BigNat, **282**, 282
BasicPath, **165**, 165–168, 185, 200, 203, 401
  length(), **166**, 166, 203
  node(), **165**, 165, 166
  predecessor(), **165**, 165, 166
  printNodes(), 167, 168, **401**, 401
  toNodeList(), **166**, 166, 185, 203, 401
BezierCurve, **72**, 72, 75–77, 371, 373
  closestPoint(), **373**
  draw(), **77**
  isStraight(), 76, **77**, 373
  length(), **371**
  mid(), **75**
  pointAt(), **72**, 72, 373
  split(), **75**, 75, 77, 371, 373
  subdivide(), 76, **77**, 77

BigInt, 275, **314**, 314–319, 450, 453, 459
  abs(), 314, **315**
  BigInt(), **314**, 314, 315
  compareTo(), 316, 319, **450**
  divideAndRemainder(), **319**
  dividedBy(), **319**, 459
  equals(), 314, **315**, 453, 459
  hashCode(), 314, **315**
  minus(), **318**, 459
  negate(), 314, **315**, 318, 459
  ONE, **315**
  plus(), **318**, 318, 453
  remainder(), **319**, 453
  signum(), 314, **315**
  TEN, **315**, 453
  times(), **319**, 453, 459
  toLong(), **316**, 316, 317
  valueOf(), **315**, 315–317, 453, 459
  ZERO, **315**, 318, 453, 459
BigInt(), in BigInt, **314**, 314, 315
BigInteger, 328
BigNat, 275, **278**, 278–286, 288, 290–292, 296–298, 301, 303, 306–308, 310–316, 318, 319, 322, 446–450, 452
  appendDigit(), **307**, 308
  BASE, **282**, 282
  BigNat(), **279**, 283, 285, 288, 291, 296–298, 303, 307, 447
  compareTo(), 281, **282**, 306, 307, 318, 450
  digit(), **278**, 278, 282, 288, 291, 303, 307, 447
  DIGIT_BITS, 281, **282**, 283, 285, 288, 291, 447
  DIGIT_MASK, 281, **282**, 283, 285, 288, 291, 447
  digitOr0(), **278**, 278, 283, 285, 288, 296, 307, 446
  divideAndRemainder(), **306**, 306, 319, 449
  divideByDigit(), 301, **303**, 306, 307
  dividedBy(), **306**, 306
  divideLong(), 306, **307**
  divideLongInternal(), 306, **307**
  equals(), 279, **280**, 291, 306, 314, 315, 449
  fromDecimal(), 308, 311, **312**
  fromUnsignedLong(), 282, **283**, 283, 303, 312, 315, 447
  hashCode(), 279, **280**, 315
  longFromDecimal(), 310, **311**, 311, 312, 448
  minus(), **288**, 290, 298, 307, 318
  multiplyBasePower(), 297, **298**, 298
  multiplySimple(), **291**, 297, 298
  ONE, **283**, 452
  plus(), 284, **285**, 286, 290, 298, 310, 312, 318
  pow(), 312, 322, 447, 449, **452**
  remainder(), **306**, 306
  slice(), 296, **297**, 297, 298, 307, 447
  TEN, **283**, 312, 447, 449
  times(), 297, **298**, 310, 312, 319, 447, 452
  timesDigit(), 307, **447**
  toString(), 313, 448, **449**
  toUnsignedLong(), **283**, 283, 316, 449
  ZERO, **283**, 283, 306, 314
BigNat(), in BigNat, **279**, 283, 285, 288, 291, 296–298, 303, 307, 447
bit(), in Bits, 107, **108**, 127, 132, 136, 140, 142, 399, 406, 407
BitCount, 382, 390
  bitCountKernighan(), **382**
  bitCountSubdivide(), **390**
bitCountKernighan(), in BitCount, **382**
bitCountSubdivide(), in BitCount, **390**
BitInputStream, 219–221, 223, 229–232, 259, 263, **434**, 434, 435, 441, 442
  BitInputStream(), **434**
  fillBuffer(), **434**, 434, 435
  readBit(), 220, 223, 259, **435**
  readBits(), 220, 221, 223, 232, 259, 263, 434, **435**, 441, 442
BitInputStream(), in BitInputStream, **434**
BitIterator, 113, **114**, 114, 115, 387
  BitIterator(), **114**, 115
  hasNext(), 113, **114**
  next(), 113, **114**
BitIterator(), in BitIterator, **114**, 115
bitLength(), in IntUtil, **222**, 222, 262, 263, 265, 266
BitmapFont, 90, 92, 93, 379
  digitsImage, **90**, 92, 93
  drawNumber(), **93**, 93
  printImage(), **92**
  writePbm(), **379**
BitOutputStream, 219, 220, 222, 228, 231, 232, 258, 262, 263, 267, 432, **433**, 433, 438
  BitOutputStream(), **433**
  flush(), 232, 432, **433**
  writeBit(), 220, 222, 232, 258, **433**
  writeBits(), 220, 222, 232, 258, 262, 263, 267, 432, **433**
BitOutputStream(), in BitOutputStream, **433**

Bits, 107–111, 114, 115, 127, 128, 132, 136, 140, 142, 391, 399, 400, 405–407
  bit(), 107, **108**, 127, 132, 136, 140, 142, 399, 406, 407
  bits(), 107, **108**, 109, 127, 128, 132
  contains(), 110, **111**, 407
  containsAll(), 110, **111**
  containsNone(), 110, **111**
  containsSome(), 110, **111**
  countTrailingZeros(), **391**
  iterate(), 114, **115**, 115, 132, 136, 140, 142, 400, 405–407
  range(), **108**, 108, 127, 405, 407
  toList(), **115**, 132
bits(), in Bits, 107, **108**, 109, 127, 128, 132
BitSet, 117, **118**, 118, 119, 181–185, 189, 190, 193, 385–389, 417, 418, 420
  add(), 118, 182, **385**, 418
  addAll(), 119, 184, **388**
  BitSet(), 181, 184, 193, **386**, 417, 418, 420
  contains(), 118, 385, **386**, 417, 418
  containsAll(), 119, 387, **388**
  equals(), 118, 119, **387**, 387
  hashCode(), 118, 119, **387**
  isEmpty(), 119, 387, **388**
  iterator(), 119, 193, **389**
  remove(), 118, 385, **386**
  removeAll(), 119, 193, **388**
  retainAll(), 119, **388**
  size(), 119, 387, **388**
BitSet(), in BitSet, 181, 184, 193, **386**, 417, 418, 420
BitSet.Iter, **389**, 389
  hasNext(), **389**
  Iter(), **389**
  next(), **389**
  remove(), **389**
blend(), in IFSBlend, **359**
blocked(), in GameState, **189**, 190
BLOCKER, in LaserPiece, 422, **423**
blockIndex(), in Sudoku, **399**
Board, 177, **178**, 178–182, 185, 188–190, 192, 193, 196, 417, 420, 423–427
  ALL_DIRECTIONS, **179**, 188, 196, 420, 427
  Board(), **178**
  DOWN, **179**, 179, 424
  getDirection(), **179**, 180, 185, 192, 193
  getX(), 177, **178**, 179, 190, 417
  getY(), 177, **178**, 179, 190, 417
  height(), **178**, 417
  index(), 177, **178**, 182, 185, 190, 417
  isInside(), **178**, 179
  LEFT, **179**, 179, 424
  moveIndex(), **179**, 179, 180, 188, 189, 193, 196, 420
  RIGHT, **179**, 179, 424
  UP, **179**, 179, 424
  width(), **178**, 417
Board(), in Board, **178**
board, in LaserBoard, **425**
Board.Direction, **179**, 179, 185, 192, 193, 421, 423–425, 427
  charCode(), **179**, 185, 192, 424
  xOff(), **179**, 179, 425
  yOff(), **179**, 179, 425
BOARD_MASK, in Connect4, **394**, 394
Boolean, 397
bottomLeft(), in Rectangle, 32, 37, **355**
BOUNDARY_LEFT, in PegBoard, **128**
BOUNDARY_RIGHT, in PegBoard, **128**
boundingBox(), in Rectangle, 354, **355**
branchL, in PythagorasTree, **56**, 57
branchR, in PythagorasTree, **56**, 57
Buddhabrot, **36**, 36–39, 360, 361
  Buddhabrot(), **36**, 36, 39
  computeHistogram(), **38**, 39, 360, 361
  drawBuddhabrot(), **39**
  drawNebulabrot(), **361**
  followTrajectory(), **37**, 38
  imageHeight, **36**
  imageWidth, **36**
  maximum(), **39**, 361
  region, **36**
Buddhabrot(), in Buddhabrot, **36**, 36, 39
BufferedImage, 19, 32, 39, 356, 357, 361, 412, 413, 415
BufferedReader, 421
buildSquares(), in PythagorasTree, **58**, 58
buildTree(), in PythagorasTree, **58**, 58
buildTreeSimple(), in PythagorasTree, **57**, 57
Byte, 229, 334
  toUnsignedInt(), 229
BYTE_TAG, in LZSS, **257**, 258

C

causesDeadlock(), in GameState, 189, **190**

ceilBinaryLog(), in IntMath, **106**, 106, 248
CellName, 313, 448, **450**, 450
  CellName(), **450**
  column(), **450**
  fromString(), 448, **450**
  row(), **450**
  toString(), 448, **450**
CellName(), in CellName, **450**
Cellular1D, 382, 385
  evolveCells(), **385**
  nextState(), 382, **385**
ChaosGame, 6, 7
  chaosGame(), **6**, 6, 7
  drawSierpinski(), 6, **7**
chaosGame(), in ChaosGame, **6**, 6, 7
Character, 93, 192, 311, 343, 427
charCode(), in Board.Direction, **179**, 185, 192, 424
checkTrip(), in JealousCouples, **406**, 406
checkWin(), in Connect4, **394**, 394
chooseIndex(), in AffineIFS, 14, **15**
CivBoard, 426, **427**, 427
  CivBoard(), **427**
  edges(), **427**
  getTerrain(), **427**
  Pos, **427**
CivBoard(), in CivBoard, **427**
CivBoard.Pos, 427
closestParam(), in LineSegment, 372, **373**, 373
closestPoint(), in BezierCurve, **373**
closestPoint(), in LineSegment, 372, **373**
Codeword, in PrefixCode, 220, 430
Collection, 154, 155, 343, 386, 406, 421–423
Collections, 166, 427
  reverse(), 166
Color, 360
column(), in CellName, **450**
Comparable, 227, 278, 281, 314
Comparator, 201
compare(), in Integer, 281, 282, 450
compareTo(), in BigInt, 316, 319, **450**
compareTo(), in BigNat, 281, **282**, 306, 307, 318, 450
compareTo(), in Duration, 455, **456**, 458
compareTo(), in HuffmanCode.Entry, **227**
Components, 418
  findComponents(), **418**
compose(), in Affine, **55**, 55, 56
compress(), in HuffmanPacker, **229**, 229
compress(), in LZHEncoder, 266, **267**
compress(), in LZSSEncoder, **258**, 258, 259
compressMoves(), in SokobanSolver, 203, 418, **419**
computeCodewords(), in PrefixCode, 218, **219**, 219
computeHistogram(), in Buddhabrot, **38**, 39, 360, 361
computePi(), in PiMachin, **459**, 459
computeSteps(), in SokobanSolver, **191**, 191, 203
Connect4, 394
  BOARD_MASK, **394**, 394
  checkWin(), **394**, 394
  fourBitsSet(), **394**, 394
  HEIGHT, **394**, 394
  WIDTH, **394**, 394
construct(), in HuffmanCode, **228**, 265
Consumer, 158, 159, 161, 350
contains(), in Bits, 110, **111**, 407
contains(), in BitSet, 118, 385, **386**, 417, 418
contains(), in Rectangle, 354, **355**
containsAll(), in Bits, 110, **111**
containsAll(), in BitSet, 119, 387, **388**
containsNone(), in Bits, 110, **111**
containsSome(), in Bits, 110, **111**
countLeadingZeros(), in LeadingZeros, **392**
countSolutions(), in PegCount, 141, **142**, 142
countTrailingZeros(), in Bits, **391**
crates(), in GameState, **183**, 188, 193, 196
crates(), in Level, **181**, 185
createFromTable(), in PrefixCodeFromTable, **430**, 430
Curve, 376, **377**, 377, 378
  pointAt(), 376, **377**, 378
  tMax(), 376, **377**, 378
  tMin(), 376, **377**, 378
CurvePlotter, 377, **378**, 378
  draw(), 377, **378**
  subdivide(), 377, **378**
CyclicBuffer, 247, **248**, 248–250, 255, 256, 441, 442
  CyclicBuffer(), **248**, 248, 249, 441
  get(), 247, **248**, 250, 255, 442
  set(), 247, **248**, 249, 256, 442
CyclicBuffer(), in CyclicBuffer, **248**, 248, 249, 441

D

DecimalCoder, 435, 437, 438

writeBCD(), 435, **437, 438**
writeDeclets(), **438**
decode(), in LZ4Decoder, **444**
decodeBlock(), in LZHDecoder, 441, **442**
DecodeException, **444**, 444, 445
decodeInteger(), in LZHDecoder, **442**
decodeLength(), in LZ4Decoder, 444, **445**, 445
decodeLiteral(), in LZ4Decoder, 444, **445**
decodeMatch(), in LZ4Decoder, 444, **445**
decodeSymbol(), in PrefixCode, 220, **221**, 230, 442
decompress(), in HuffmanPacker, **230**, 230
decompress(), in LZHDecoder, **441**, 441
decompress(), in LZSSDecoder, 258, **259**
deflateThick(), in Penrose, 375, **376**
deflateThin(), in Penrose, 375, **376**
Deque, 254, 256, 343, 344
diameter(), in GraphUtil, **163**, 163
dictionary, in LZHDecoder, **441**
DieHardGraph, **153**, 153–156, 159, 160, 162, 167
DieHardGraph(), **153**, 156, 159, 160, 162, 167
makeNode(), **153**, 153–155
neighbors(), 153, 154, **155**, 156
nodes(), 153, **154**
DieHardGraph(), in DieHardGraph, **153**, 156, 159, 160, 162, 167
digit(), in BigNat, **278**, 278, 282, 288, 291, 303, 307, 447
DIGIT_BITS, in BigNat, 281, **282**, 283, 285, 288, 291, 447
DIGIT_MASK, in BigNat, 281, **282**, 283, 285, 288, 291, 447
digitOr0(), in BigNat, **278**, 278, 283, 285, 288, 296, 307, 446
digitsImage, in BitmapFont, **90**, 92, 93
Dijkstra, **201**, 201–203
Dijkstra(), **201**, 201, 203
nextNode(), 201, **202**, 202, 203
shortestPath(), 202, **203**, 203
traverse(), 202, **203**
Dijkstra(), in Dijkstra, **201**, 201, 203
distance(), in GraphLayer, **160**, 163, 196
distance(), in Point, **354**, 371, 373
divideAndRemainder(), in BigInt, **319**
divideAndRemainder(), in BigNat, **306**, 306, 319, 449
divideByDigit(), in BigNat, 301, **303**, 306, 307
dividedBy(), in BigInt, **319**, 459
dividedBy(), in BigNat, **306**, 306
divideLong(), in BigNat, 306, **307**
divideLongInternal(), in BigNat, 306, **307**
Double, 334
down(), in PegBoard, **128**, 128, 130, 131
DOWN, in Board, **179**, 179, 424
draw(), in BezierCurve, **77**
draw(), in CurvePlotter, 377, **378**
drawBuddhabrot(), in Buddhabrot, **39**
drawMandelbrot(), in MandelbrotPainter, 31, **32**, 33
drawNebulabrot(), in Buddhabrot, **361**
drawNumber(), in BitmapFont, **93**, 93
drawPoints(), in IFSPainter, **357**
drawSierpinski(), in ChaosGame, 6, **7**
Duration, **455**, 455–458
abs(), **455**, 458
addDigits(), 456, **457**, 458
compareTo(), 455, **456**, 458
Duration(), **455**, 457
equals(), **456**
hashCode(), **456**
makeSeconds(), 456, **457**, 458
minus(), 456, **458**
negate(), **455**, 458
ofUnit(), **457**
plus(), 456, **458**
seconds(), 456, **457**
subDigits(), 456, **457**, 458
total(), **458**
totalWeeks(), 456, **458**
Duration(), in Duration, **455**, 457

E

EAST, in HilbertCurve, 68, **69**, 371
eccentricity(), in GraphUtil, **163**, 163
Edge(), in WeightedGraph.Edge, **195**
edges(), in CivBoard, **427**
edges(), in WeightedGraph, 194, **195**, 202
edges(), in WeightedPushGraph, **196**, 196
Ellipse2D, 7
emit(), in LaserPiece, 421, **423**, 425
Emitter, **423**
EMPTY, in LaserPiece, 422, **423**, 425, 426
EMPTY, in PrefixCodeFromTable, **430**
encryptCaesar(), in Ascii, 428, **429**
end(), in JealousCouples, **407**
ensure(), in LZ4Decoder, **444**, 445
Entry, 228

Entry(), in HuffmanCode.Entry, **227**, 228
Entry, in HuffmanCode, 227
equals(), in BigInt, 314, **315**, 453, 459
equals(), in BigNat, 279, **280**, 291, 306, 314, 315, 449
equals(), in BitSet, 118, 119, **387**, 387
equals(), in Duration, **456**
escapeTime(), in JuliaSet, 360, **362**
escapeTime(), in MandelbrotSet, **31**, 32, 38
everyone(), in JealousCouples, **407**
evolveCells(), in Cellular1D, **385**
Exception, 444
extendedChaosGame(), in AffineIFS, 14, **15**

F

farthestLayer(), in GraphUtil, 162, **163**, 163, 164
fillBuffer(), in BitInputStream, **434**, 434, 435
FINAL_PEGS, in PegBoard, **127**, 136, 142
findComponents(), in Components, **418**
findMatches(), in LZTokenizer, 250, 255, **256**
findPathTo(), in Graph, **167**, 167, 185, 190, 192
findSimilarWords(), in SimilarWords, 205, **422**, 423
FLIP_H, in PegCount, 138, **139**
flipDigit(), in WolfGoatCabbageGraph, **403**
flipH(), in PegCount, 138, **139**, 139
Float, 334, 336, 343
FloodFill, 413, 415
  floodFill_pixelBased, **413**
  floodFill_spanBased(), **415**
FloodFill.PixelPos, **413**, 415
floodFill_pixelBased, in FloodFill, **413**
floodFill_spanBased(), in FloodFill, **415**
floorBinaryLog(), in IntMath, **106**, 106
flush(), in BitOutputStream, 232, 432, **433**
followTrajectory(), in Buddhabrot, **37**, 38
fourBitsSet(), in Connect4, **394**, 394
fromDecimal(), in BigNat, 308, 311, **312**
fromDigits(), in Radix, **449**
fromString(), in CellName, 448, **450**
fromUnsignedLong(), in BigNat, 282, **283**, 283, 303, 312, 315, 447
front, in LZTokenizer, **249**, 250, 255, 256
FULL_BOARD, in PegBoard, **127**, 129, 131
Function, 142, 396

G

GameOfLife, 362, **363**, 363
  advance(), 362, **363**
  GameOfLife(), **363**
  get(), 362, **363**
  set(), 362, **363**
GameOfLife(), in GameOfLife, **363**
GameState, 182, **183**, 183, 185, 188–193, 196, 203, 416, 417
  allCratesOnGoals(), 182, **183**, 183, 190, 203
  blocked(), **189**, 190
  causesDeadlock(), 189, **190**
  crates(), **183**, 188, 193, 196
  GameState(), **183**, 203
  toString(), 416, **417**
  tryPush(), 188, **189**, 189, 196
  workerPos(), **183**, 188, 192, 193, 196
GameState(), in GameState, **183**, 203
get(), in CyclicBuffer, 247, **248**, 250, 255, 442
get(), in GameOfLife, 362, **363**
getByte(), in LZToken, **245**, 258, 265, 266
getDirection(), in Board, **179**, 180, 185, 192, 193
getLength(), in LZToken, **245**, 245, 258, 265, 266
getOffset(), in LZToken, **245**, 245, 258, 265, 266
getTerrain(), in CivBoard, **427**
getX(), in Board, 177, **178**, 179, 190, 417
getY(), in Board, 177, **178**, 179, 190, 417
goals(), in Level, **181**, 183, 190, 417, 420
Graph, **153**, 153–155, 157–163, 166–168, 183–185, 188, 190, 192, 194, 196, 403, 405, 409, 411, 417, 418, 420, 423–426
  findPathTo(), **167**, 167, 185, 190, 192
  neighbors(), **153**, 153, 155, 158, 161, 167, 194
  nodes(), **153**, 153, 154
  scan(), 157, **158**, 158–160, 417
  scanLayers(), 160, **161**, 161, 196
  scanPaths(), 166, **167**, 167
  visitLayers(), 160, **161**, 162, 163, 196
  visitNodes(), 158, **159**, 159, 188, 418, 420, 425, 426
Graphics2D, 6, 7, 356
GraphLayer, **160**, 160, 161, 163, 196
  distance(), **160**, 163, 196
  GraphLayer(), **160**, 161
  nodes(), **160**, 196
  start(), **160**
GraphLayer(), in GraphLayer, **160**, 161
GraphUtil, 162–164
  diameter(), **163**, 163
  eccentricity(), **163**, 163

farthestLayer(), 162, **163**, 163, 164
radius(), **163**, 163
greeting, in Hello, **xi**, xi

H

HanoiGraph, 410, **411**, 411
neighbors(), 410, **411**
hashCode(), in BigInt, 314, **315**
hashCode(), in BigNat, 279, **280**, 315
hashCode(), in BitSet, 118, 119, **387**
hashCode(), in Duration, **456**
hashFront(), in LZTokenizer, **255**, 255, 256
HashMap, 152, 201, 202, 227, 339, 343, 345, 396, 397
HashSet, 155, 157, 158, 161, 167, 345, 406, 422, 423, 426
hasNext(), in BitIterator, 113, **114**
hasNext(), in BitSet.Iter, **389**
hasRemaining(), in LZTokenizer, **249**, 251, 258, 267
height(), in Board, **178**, 417
height(), in Rectangle, 32, 354, **355**
HEIGHT, in Connect4, **394**, 394
Hello, **xi**, xi
greeting, **xi**, xi
printGreeting(), **xi**
HilbertCurve, 68, **69**, 69, 70, 370, 371
EAST, 68, **69**, 371
hilbertCurve(), 69, **70**, 70, 370, 371
NORTH, 68, **69**, 70, 371
printPoints(), **371**
SOUTH, 68, **69**, 371
turn(), 68, **69**, 70
WEST, 68, **69**, 371
hilbertCurve(), in HilbertCurve, 69, **70**, 70, 370, 371
HuffmanCode, 227, 228, 265
construct(), **228**, 265
Entry, 227
HuffmanCode.Entry, **227**, 227, 228
compareTo(), **227**
Entry(), **227**, 228
HuffmanPacker, 229, 230
compress(), **229**, 229
decompress(), **230**, 230

I

IDENTITY, in Affine, **53**, 55, 58
IFSBlend, 359
blend(), **359**
lerp(), **359**
IFSPainter, **357**, 357, 358
drawPoints(), **357**
IFSPainter(), **358**
IFSPainter(), in IFSPainter, **358**
IllegalArgumentException, 267, 297, 385
imageHeight, in Buddhabrot, **36**
imageWidth, in Buddhabrot, **36**
index(), in Board, 177, **178**, 182, 185, 190, 417
index, in LZTokenizer, **254**, 256
INDEX_SIZE, in LZTokenizer, **254**, 255
INIT_PEGS, in PegBoard, **127**, 136, 141
initBuffers(), in LZTokenizer, 248, **249**, 249
initIndex(), in LZTokenizer, 248, 249, **254**, 254
input, in LZHDecoder, **441**
input, in LZTokenizer, **249**
InputStream, 228, 229, 232, 248, 249, 256, 258, 267, 434, 438
insert(), in PrefixCodeFromTable, **430**, 430
Integer, 46, 47, 106, 113–115, 118, 120, 142, 155, 157, 161, 183, 184, 219, 222, 227, 281, 282, 334, 336, 343, 350, 351, 389, 397, 405, 409, 427, 450, 456, 457
compare(), 281, 282, 450
numberOfLeadingZeros(), 222
intersection(), in Rectangle, 354, **355**
IntMath, 106, 248, 451
ceilBinaryLog(), **106**, 106, 248
floorBinaryLog(), **106**, 106
pow(), **451**, 451
IntUtil, 222, 262, 263, 265, 266
bitLength(), **222**, 222, 262, 263, 265, 266
IOException, 249, 421, 433–435, 437, 438
isByteToken(), in LZToken, **245**, 245, 258, 265, 266
isDigit(), in Ascii, **428**
isEmpty(), in BitSet, 119, 387, **388**
isEmpty(), in Rectangle, 354, **355**
isInside(), in Board, **178**, 179
isLevelClosed(), in LevelClosed, **417**
isLower(), in Ascii, **428**
isStraight(), in BezierCurve, 76, **77**, 373
isStraight(), in QuadraticBezier, **374**, 374
isUpper(), in Ascii, **428**
isWinning(), in NimMemoize, 396, **397**
Iter, 119
Iter(), in BitSet.Iter, **389**
Iterable, 114, 115, 119, 153, 185, 343–345, 409

iterate(), in Bits, 114, **115**, 115, 132, 136, 140, 142, 400, 405–407
Iterator, 113, 114, 119, 344, 345, 387, 389
iterator(), in BitSet, 119, 193, **389**

J

JealousCouples, **405**, 405–407
  checkTrip(), **406**, 406
  end(), **407**
  everyone(), **407**
  makeState(), 406, **407**, 407
  MAX_COUPLES, **405**
  MEN_MASK, **405**
  noJealousy(), **405**, 406
  normalize(), **407**
  start(), **407**
  WOMEN_MASK, **405**
JealousCouples.State, 404, **405**, 405–407
JuliaSet, 360, 362
  escapeTime(), 360, **362**
jump(), in PegBoard, **127**, 136, 142

L

LaserBoard, 424, **425**, 425, 426
  board, **425**
  neighbors(), **425**
  Pos, **425**, 425, 426
  Ray, **425**, 425, 426
  simulateLight(), **426**
LaserPiece, 421, 422, **423**, 423–426
  BLOCKER, 422, **423**
  emit(), 421, **423**, 425
  EMPTY, 422, **423**, 425, 426
  reflect(), 421, 422, **423**, 425
LeadingZeros, 392
  countLeadingZeros(), **392**
left(), in PegBoard, 127, **128**, 128, 130, 131
LEFT, in Board, **179**, 179, 424
length(), in BasicPath, **166**, 166, 203
length(), in BezierCurve, **371**
LENGTH_BITS, in LZSS, **257**
lerp(), in IFSBlend, **359**
lerp(), in Point, **6**, 6, 75, 365, 372–374, 376
Level, **181**, 181–185, 188–190, 196, 203, 416–418, 420
  crates(), **181**, 185
  goals(), **181**, 183, 190, 417, 420
  Level(), **181**
  parse(), 181, **416**, 416
  resetSquare(), **182**
  setCode(), 181, **182**, 416
  walls(), **181**, 417, 420
  workerStart(), **181**, 417, 418, 420
Level(), in Level, **181**
level, in MoveGraph, **184**
level, in PushGraph, **188**
LevelClosed, 417
  isLevelClosed(), **417**
LineSegment, 372, 373
  closestParam(), 372, **373**, 373
  closestPoint(), 372, **373**
  LineSegment(), **373**, 373
  pointAt(), 372, **373**
LineSegment(), in LineSegment, **373**, 373
LinkedList, 343
List, 7, 12, 15, 56, 58, 70, 77, 115, 136, 152, 158, 161, 166, 167, 185, 188, 190, 191, 196, 254, 343, 344, 350, 351, 365, 374, 376, 378, 397, 403, 409, 411, 418, 420, 423–425, 427, 440
  of(), 158
Long, 112–114, 120, 315–317, 387–389
  MAX_VALUE, 316
  MIN_VALUE, 316
longFromDecimal(), in BigNat, 310, **311**, 311, 312, 448
lookaheadSize, in LZTokenizer, **249**, 250, 255, 256
LZ4Decoder, **272**, 272, 444, 445
  decode(), **444**
  decodeLength(), 444, **445**, 445
  decodeLiteral(), 444, **445**
  decodeMatch(), 444, **445**
  ensure(), **444**, 445
LZHDecoder, **441**, 441, 442
  decodeBlock(), 441, **442**
  decodeInteger(), **442**
  decompress(), **441**, 441
  dictionary, **441**
  input, **441**
  LZHDecoder(), **441**
  out, **441**
  outLength, **441**
  write(), 441, **442**
LZHDecoder(), in LZHDecoder, **441**
LZHEncoder, 263, **264**, 264–268, 439
  compress(), 266, **267**
  LZHEncoder(), **267**

maxLength, **264**, 265–267
maxOffset, **264**, 266, 267
minLength, **264**, 265–267
offsetCode, **264**, 264, 265, 267
prepareCodes(), **265**, 266, 267
tagCode, **264**, 264, 265, 267
writeToken(), **266**, 266, 267
LZHEncoder(), in LZHEncoder, **267**
LZSS, **257**, 257–259
BYTE_TAG, **257**, 258
LENGTH_BITS, **257**
MATCH_TAG, **257**, 258
MAX_LENGTH, **257**, 258
MAX_OFFSET, **257**, 258
MIN_LENGTH, **257**, 258
OFFSET_BITS, **257**, 258
LZSSDecoder, 258, 259
decompress(), 258, **259**
LZSSEncoder, 258, 259
compress(), **258**, 258, 259
LZToken, 245, 250, 258, 265, 266
getByte(), **245**, 258, 265, 266
getLength(), **245**, 245, 258, 265, 266
getOffset(), **245**, 245, 258, 265, 266
isByteToken(), **245**, 245, 258, 265, 266
makeByte(), **245**, 245, 250
makeMatch(), **245**, 245, 250
LZTokenizer, 248, **249**, 249–251, 254–256, 258, 267
advanceInput(), 250, 255, **256**
findMatches(), 250, 255, **256**
front, **249**, 250, 255, 256
hashFront(), **255**, 255, 256
hasRemaining(), **249**, 251, 258, 267
index, **254**, 256
INDEX_SIZE, **254**, 255
initBuffers(), 248, **249**, 249
initIndex(), 248, 249, **254**, 254
input, **249**
lookaheadSize, **249**, 250, 255, 256
LZTokenizer(), 248, **249**, 258, 267
maxLength, **249**, 249, 250
maxOffset, **249**, 249, 256
minLength, **249**, 250, 255
nextToken(), **250**, 251, 258
readTokens(), 250, **251**, 267
window, **249**, 250, 255, 256
LZTokenizer(), in LZTokenizer, 248, **249**, 258, 267

M

makeByte(), in LZToken, **245**, 245, 250
makeInner(), in PrefixCode, **218**, 218, 223, 228, 430
makeLeaf(), in PrefixCode, **218**, 218, 223, 228, 430
makeMatch(), in LZToken, **245**, 245, 250
makeNode(), in DieHardGraph, **153**, 153–155
makeSeconds(), in Duration, 456, **457**, 458
makeState(), in JealousCouples, 406, **407**, 407
MandelbrotPainter, 31–33, 36
drawMandelbrot(), 31, **32**, 33
zoomRect(), **33**, 33, 36
MandelbrotSet, 31, 32, 38
escapeTime(), **31**, 32, 38
Map, 201, 343, 344, 397, 424
MATCH_TAG, in LZSS, **257**, 258
Math, 7, 38, 53, 69, 72, 139, 155, 219, 255, 296, 297, 307, 317, 334, 335, 354, 356, 360, 361, 363, 368, 373, 416, 440, 457
min(), 139, 296
MAX_COUPLES, in JealousCouples, **405**
MAX_LENGTH, in LZSS, **257**, 258
MAX_OFFSET, in LZSS, **257**, 258
MAX_VALUE, in Long, 316
maximum(), in Buddhabrot, **39**, 361
maxLength, in LZHEncoder, **264**, 265–267
maxLength, in LZTokenizer, **249**, 249, 250
maxOffset, in LZHEncoder, **264**, 266, 267
maxOffset, in LZTokenizer, **249**, 249, 256
maxSymbol(), in PrefixCode, 218, **219**
Memoizer, 395
MEN_MASK, in JealousCouples, **405**
mid(), in BezierCurve, **75**
min(), in Math, 139, 296
MIN_LENGTH, in LZSS, **257**, 258
MIN_VALUE, in Long, 316
minLength, in LZHEncoder, **264**, 265–267
minLength, in LZTokenizer, **249**, 250, 255
minus(), in BigInt, **318**, 459
minus(), in BigNat, **288**, 290, 298, 307, 318
minus(), in Duration, 456, **458**
movablePegs(), in PegBoard, 130, **131**, 132, 136, 142
MoveGraph, 183, **184**, 184–186, 188, 192, 196, 417, 418, 420
level, **184**
MoveGraph(), 183, **184**, 185, 188, 192, 196, 417, 418, 420

neighbors(), 183, **184**, 185
obstacles, **184**
MoveGraph(), in MoveGraph, 183, **184**, 185, 188, 192, 196, 417, 418, 420
moveIndex(), in Board, **179**, 179, 180, 188, 189, 193, 196, 420
multiplyBasePower(), in BigNat, 297, **298**, 298
multiplySimple(), in BigNat, **291**, 297, 298
MysteryCurve, **377**, 377
pointAt(), **377**
tMax(), **377**
tMin(), **377**

N

NavigableMap, 344
NavigableSet, 344
negate(), in BigInt, 314, **315**, 318, 459
negate(), in Duration, **455**, 458
neighbors(), in DieHardGraph, 153, 154, **155**, 156
neighbors(), in Graph, **153**, 153, 155, 158, 161, 167, 194
neighbors(), in HanoiGraph, 410, **411**
neighbors(), in LaserBoard, **425**
neighbors(), in MoveGraph, 183, **184**, 185
neighbors(), in PushGraph, **188**, 188, 193
neighbors(), in ThreeJugProblem, 408, **409**
neighbors(), in WolfGoatCabbageGraph, **403**
neighbors(), in WordLadderGraph, **423**
nested(), in Whirl, **365**
next(), in BitIterator, 113, **114**
next(), in BitSet.Iter, **389**
nextNode(), in Dijkstra, 201, **202**, 202, 203
nextState(), in Cellular1D, 382, **385**
nextToken(), in LZTokenizer, **250**, 251, 258
Nim, 398
printWinningMoves(), **398**, 398
NimMemoize, 396, **397**, 397
isWinning(), 396, **397**
NimMemoize(), 396, **397**
normalize(), 396, **397**
NimMemoize(), in NimMemoize, 396, **397**
node(), in BasicPath, **165**, 165, 166
node(), in Path, 165, **166**, 167
node(), in WeightedPath, **200**, 202
Node, in PrefixCode, 221, 222, 227, 430
nodes(), in DieHardGraph, 153, **154**
nodes(), in Graph, **153**, 153, 154
nodes(), in GraphLayer, **160**, 196
nodes(), in ThreeJugProblem, 408, **409**
nodes(), in WolfGoatCabbageGraph, **403**
nodes(), in WordLadderGraph, **423**
noJealousy(), in JealousCouples, **405**, 406
normalize(), in JealousCouples, **407**
normalize(), in NimMemoize, 396, **397**
normalize(), in PegCount, 138, **139**, 142
NORTH, in HilbertCurve, 68, **69**, 70, 371
NumberFormatException, 449
numberOfLeadingZeros(), in Integer, 222
numSolutions, in PegCount, **142**, 395

O

Object, 163, 334, 337, 339, 342, 343
Objects, 339
obstacles, in MoveGraph, **184**
of(), in List, 158
OFFSET_BITS, in LZSS, **257**, 258
offsetCode, in LZHEncoder, **264**, 264, 265, 267
ofUnit(), in Duration, **457**
ONE, in BigInt, **315**
ONE, in BigNat, **283**, 452
out, in LZHDecoder, **441**
outLength, in LZHDecoder, **441**
OutputStream, 230, 232, 259, 433, 437, 441

P

parse(), in Level, 181, **416**, 416
parseDigit(), in Radix, 448, **449**, 449
Path, 165, **166**, 166–168
node(), 165, **166**, 167
Path(), **166**, 167, 168
predecessor(), 165, **166**
Path(), in Path, **166**, 167, 168
PegBoard, 126, **127**, 127–132, 136, 141, 142
BOUNDARY_LEFT, **128**
BOUNDARY_RIGHT, **128**
down(), **128**, 128, 130, 131
FINAL_PEGS, **127**, 136, 142
FULL_BOARD, **127**, 129, 131
INIT_PEGS, **127**, 136, 141
jump(), **127**, 136, 142
left(), 127, **128**, 128, 130, 131
movablePegs(), 130, **131**, 132, 136, 142
reachableHoles(), 130, **131**, 132, 136, 142
reachablePositions(), 130, **131**
right(), **128**, 128–131
up(), **128**, 128, 130, 131

PegCount, 138–142, 395
countSolutions(), 141, **142**, 142
FLIP_H, 138, **139**
flipH(), 138, **139**, 139
normalize(), 138, **139**, 142
numSolutions, **142**, 395
ROTATE, 138, **139**
rotate(), 138, **139**, 139
shuffleBits(), 138, 139, **140**
PegSolve, 136
printSolutions(), **136**, 136
Penrose, 375, 376
deflateThick(), 375, **376**
deflateThin(), 375, **376**
PiMachin, 459
arctan1overX(), **459**, 459
computePi(), **459**, 459
planMovement(), in SokobanSolver, 191, **192**, 192
plus(), in BigInt, **318**, 318, 453
plus(), in BigNat, 284, **285**, 286, 290, 298, 310, 312, 318
plus(), in Duration, 456, **458**
Point, 4, **5**, 5–7, 12, 15, 18, 33, 37, 54, 56, 58, 72, 75, 77, 354, 356, 365, 368, 371–374, 376, 378
angle(), 18, **356**
distance(), **354**, 371, 373
lerp(), **6**, 6, 75, 365, 372–374, 376
Point(), **5**, 6, 7, 33, 54, 56, 356, 368, 371, 372
polar(), **356**
radius(), 18, **356**
x(), **5**, 33
y(), **5**, 33
Point(), in Point, **5**, 6, 7, 33, 54, 56, 356, 368, 371, 372
pointAt(), in BezierCurve, **72**, 72, 373
pointAt(), in Curve, 376, **377**, 378
pointAt(), in LineSegment, 372, **373**
pointAt(), in MysteryCurve, **377**
polar(), in Point, **356**
Pos, in CivBoard, **427**
Pos, in LaserBoard, **425**, 425, 426
pow(), in BigNat, 312, 322, 447, 449, **452**
pow(), in IntMath, **451**, 451
predecessor(), in BasicPath, **165**, 165, 166
predecessor(), in Path, 165, **166**
predecessor(), in WeightedPath, **200**
Predicate, 157, 158, 161, 167, 202, 203, 351
PrefixCode, 217, **218**, 218–223, 227–230, 263, 266, 267, 430, 442
Codeword, 220, 430
computeCodewords(), 218, **219**, 219
decodeSymbol(), 220, **221**, 230, 442
makeInner(), **218**, 218, 223, 228, 430
makeLeaf(), **218**, 218, 223, 228, 430
maxSymbol(), 218, **219**
Node, 221, 222, 227, 430
PrefixCode(), 218, **219**, 223, 228
readCode(), **223**, 230, 442
readTree(), 222, **223**, 223
writeCode(), **222**, 222, 229, 267
writeSymbol(), **220**, 220, 229, 263, 266, 267
writeTree(), 221, **222**, 222
PrefixCode(), in PrefixCode, 218, **219**, 223, 228
PrefixCode.Codeword, 217, **218**, 219
PrefixCode.Node, 217, **218**, 218, 219, 223, 228, 430
PrefixCodeFromTable, 430
createFromTable(), **430**, 430
EMPTY, **430**
insert(), **430**, 430
prepareCodes(), in LZHEncoder, **265**, 266, 267
printGreeting(), in Hello, **xi**
printImage(), in BitmapFont, **92**
printNodes(), in BasicPath, 167, 168, **401**, 401
printPoints(), in HilbertCurve, **371**
printSolutions(), in PegSolve, **136**, 136
printWinningMoves(), in Nim, **398**, 398
PrintWriter, 379
PriorityQueue, 201, 202, 208, 228
probabilities, in AffineIFS, 12, **13**
PushGraph, **188**, 188, 190, 193
level, **188**
neighbors(), **188**, 188, 193
PushGraph(), **188**, 190
PushGraph(), in PushGraph, **188**, 190
putDigit(), in Sudoku, 398, **399**, 400
PythagorasTree, **56**, 56–58
branchL, **56**, 57
branchR, **56**, 57
buildSquares(), **58**, 58
buildTree(), **58**, 58
buildTreeSimple(), **57**, 57
PythagorasTree(), **56**
UNIT_SQUARE, **56**, 57, 58
PythagorasTree(), in PythagorasTree, **56**

Q

QuadraticBezier, **374**, 374
  isStraight(), **374**, 374
  subdivide(), **374**
Queue, 344

R

radius(), in GraphUtil, **163**, 163
radius(), in Point, 18, **356**
Radix, 448, 449
  fromDigits(), **449**
  parseDigit(), 448, **449**, 449
Random, 6, 350, 351
range(), in Bits, **108**, 108, 127, 405, 407
Ray, in LaserBoard, **425**, 425, 426
reachableHoles(), in PegBoard, 130, **131**, 132, 136, 142
reachablePositions(), in PegBoard, 130, **131**
readBit(), in BitInputStream, 220, 223, 259, **435**
readBits(), in BitInputStream, 220, 221, 223, 232, 259, 263, 434, **435**, 441, 442
readCode(), in PrefixCode, **223**, 230, 442
readTokens(), in LZTokenizer, 250, **251**, 267
readTree(), in PrefixCode, 222, **223**, 223
readWords(), in WordList, 420, **421**
Rectangle, 16, 31–33, 36, 37, 354, **355**, 355
  bottomLeft(), 32, 37, **355**
  boundingBox(), 354, **355**
  contains(), 354, **355**
  height(), 32, 354, **355**
  intersection(), 354, **355**
  isEmpty(), 354, **355**
  Rectangle(), 33, **355**
  topRight(), **355**
  width(), 32, 37, 354, **355**
Rectangle(), in Rectangle, 33, **355**
reflect(), in LaserPiece, 421, 422, **423**, 425
reflect(), in TransparentMirror, 422, **424**
region, in Buddhabrot, **36**
remainder(), in BigInt, **319**, 453
remainder(), in BigNat, **306**, 306
remove(), in BitSet, 118, 385, **386**
remove(), in BitSet.Iter, **389**
removeAll(), in BitSet, 119, 193, **388**
removeDigit(), in Sudoku, 398, **399**, 400
resetSquare(), in Level, **182**
retainAll(), in BitSet, 119, **388**
reverse(), in Collections, 166
reverse(), in ReverseList, **365**
ReverseList, 365
  reverse(), **365**
right(), in PegBoard, **128**, 128–131
RIGHT, in Board, **179**, 179, 424
rotate(), in PegCount, 138, **139**, 139
ROTATE, in PegCount, 138, **139**
rotation(), in Affine, **53**, 53, 55, 56
row(), in CellName, **450**
RuntimeException, 311, 427

S

scaling(), in Affine, **53**, 53, 55, 56
scan(), in Graph, 157, **158**, 158–160, 417
scanLayers(), in Graph, 160, **161**, 161, 196
scanPaths(), in Graph, 166, **167**, 167
seconds(), in Duration, 456, **457**
Set, 343, 344, 423, 426
set(), in CyclicBuffer, 247, **248**, 249, 256, 442
set(), in GameOfLife, 362, **363**
setCode(), in Level, 181, **182**, 416
shortestPath(), in Dijkstra, 202, **203**, 203
shuffleBits(), in PegCount, 138, 139, **140**
signum(), in BigInt, 314, **315**
SimilarWords, 205, **206**, 206, 422, 423
  addWord(), 205, **422**
  findSimilarWords(), 205, **422**, 423
  SimilarWords(), **206**, 423
SimilarWords(), in SimilarWords, **206**, 423
SimpleToken, **243**, 244, 246, 440
SimpleToken.Byte, **243**, 440
SimpleToken.Match, **243**, 440
simulateLight(), in LaserBoard, **426**
size(), in BitSet, 119, 387, **388**
slice(), in BigNat, 296, **297**, 297, 298, 307, 447
SokobanSolver, 191–193, 203, 418, 419
  compressMoves(), 203, 418, **419**
  computeSteps(), **191**, 191, 203
  planMovement(), 191, **192**, 192
  workerBeforePush(), 192, **193**
solve(), in Sudoku, **400**, 400
SOUTH, in HilbertCurve, 68, **69**, 371
split(), in BezierCurve, **75**, 75, 77, 371, 373
start(), in GraphLayer, **160**
start(), in JealousCouples, **407**
String, xi, 93, 142, 153–155, 157, 159, 167, 181, 204, 205, 254, 308, 311, 312, 334, 342, 343,

379, 401, 403, 411, 416, 417, 419, 421–423, 449
StringBuilder, 191, 192, 400, 417, 419, 422, 450
Strings, 400
subDigits(), in Duration, 456, **457**, 458
subdivide(), in BezierCurve, 76, **77**, 77
subdivide(), in CurvePlotter, 377, **378**
subdivide(), in QuadraticBezier, **374**
Sudoku, 398, **399**, 399, 400
  blockIndex(), **399**
  putDigit(), 398, **399**, 400
  removeDigit(), 398, **399**, 400
  solve(), **400**, 400
  Sudoku(), **399**, 400
Sudoku(), in Sudoku, **399**, 400
Supplier, 350
System, xi, 92, 132, 136, 159, 167, 168, 272, 298, 307, 334, 350, 368, 371, 398, 400, 444, 445

T

tagCode, in LZHEncoder, **264**, 264, 265, 267
target(), in WeightedGraph.Edge, **195**, 202
TEN, in BigInt, **315**, 453
TEN, in BigNat, **283**, 312, 447, 449
ThreeJugProblem, 408, 409
  neighbors(), 408, **409**
  nodes(), 408, **409**
times(), in Affine, **55**, 55, 58
times(), in BigInt, **319**, 453, 459
times(), in BigNat, 297, **298**, 310, 312, 319, 447, 452
timesDigit(), in BigNat, 307, **447**
tMax(), in Curve, 376, **377**, 378
tMax(), in MysteryCurve, **377**
tMin(), in Curve, 376, **377**, 378
tMin(), in MysteryCurve, **377**
toList(), in Bits, **115**, 132
toLong(), in BigInt, **316**, 316, 317
toLower(), in Ascii, **428**
toNodeList(), in BasicPath, **166**, 166, 185, 203, 401
topRight(), in Rectangle, **355**
toString(), in BigNat, 313, 448, **449**
toString(), in CellName, 448, **450**
toString(), in GameState, 416, **417**
total(), in Duration, **458**
totalWeeks(), in Duration, 456, **458**
totalWeight(), in WeightedPath, **200**, 201–203
toUnsignedInt(), in Byte, 229
toUnsignedLong(), in BigNat, **283**, 283, 316, 449
toUpper(), in Ascii, **428**
transformPoint(), in Affine, **12**, 12, 15
transformPoints(), in Affine, **12**, 12, 54, 57, 58
transforms, in AffineIFS, 12, **13**
translation(), in Affine, **53**, 53, 55, 56
TransparentMirror, 422, **424**, 424
  reflect(), 422, **424**
traverse(), in Dijkstra, 202, **203**
tryPush(), in GameState, 188, **189**, 189, 196
turn(), in HilbertCurve, 68, **69**, 70

U

UNIT_SQUARE, in PythagorasTree, **56**, 57, 58
up(), in PegBoard, **128**, 128, 130, 131
UP, in Board, **179**, 179, 424

V

valueOf(), in BigInt, **315**, 315–317, 453, 459
visitLayers(), in Graph, 160, **161**, 162, 163, 196
visitNodes(), in Graph, 158, **159**, 159, 188, 418, 420, 425, 426

W

walls(), in Level, **181**, 417, 420
weight(), in WeightedGraph.Edge, **195**, 202
WeightedGraph, 194, **195**, 195, 196, 201–203, 426, 427
  edges(), 194, **195**, 202
WeightedGraph.Edge, **195**, 195, 196, 202, 427
  Edge(), **195**
  target(), **195**, 202
  weight(), **195**, 202
WeightedPath, **200**, 200–203
  node(), **200**, 202
  predecessor(), **200**
  totalWeight(), **200**, 201–203
  WeightedPath(), **200**, 202
WeightedPath(), in WeightedPath, **200**, 202
WeightedPushGraph, **196**, 196, 203
  edges(), **196**, 196
  WeightedPushGraph(), **196**, 203
WeightedPushGraph(), in WeightedPushGraph, **196**, 203
WEST, in HilbertCurve, 68, **69**, 371
Whirl, 365
  nested(), **365**

whirl(), **365**
whirl(), in Whirl, **365**
width(), in Board, **178**, 417
width(), in Rectangle, 32, 37, 354, **355**
WIDTH, in Connect4, **394**, 394
window, in LZTokenizer, **249**, 250, 255, 256
WolfGoatCabbageGraph, **403**, 403
flipDigit(), **403**
neighbors(), **403**
nodes(), **403**
WOMEN_MASK, in JealousCouples, **405**
WordLadderGraph, **423**, 423
neighbors(), **423**
nodes(), **423**
WordLadderGraph(), **423**
WordLadderGraph(), in WordLadderGraph, **423**
WordList, 420, 421
readWords(), 420, **421**
workerBeforePush(), in SokobanSolver, 192, **193**
workerPos(), in GameState, **183**, 188, 192, 193, 196
workerStart(), in Level, **181**, 417, 418, 420
write(), in LZHDecoder, 441, **442**
writeBCD(), in DecimalCoder, 435, **437, 438**
writeBit(), in BitOutputStream, 220, 222, 232, 258, **433**
writeBits(), in BitOutputStream, 220, 222, 232, 258, 262, 263, 267, 432, **433**
writeCode(), in PrefixCode, **222**, 222, 229, 267
writeDeclets(), in DecimalCoder, **438**
writePbm(), in BitmapFont, **379**
writeSymbol(), in PrefixCode, **220**, 220, 229, 263, 266, 267
writeToken(), in LZHEncoder, **266**, 266, 267
writeTree(), in PrefixCode, 221, **222**, 222

X

x(), in Point, **5**, 33
xOff(), in Board.Direction, **179**, 179, 425

Y

y(), in Point, **5**, 33
yOff(), in Board.Direction, **179**, 179, 425

Z

ZERO, in BigInt, **315**, 318, 453, 459
ZERO, in BigNat, **283**, 283, 306, 314
zoomRect(), in MandelbrotPainter, **33**, 33, 36

Index

A

A* algorithm, 208, 209
access modifiers, 335, 338, 341
accessor methods, 165, 338
adaptive subdivision, 75–78, 85
ADD, 284, 288, 289, 291
addition, 283–285
adjacency lists, 152, 153
affine iterated function systems, 358
  blending, 19–21
  chaos game, 13
  choosing probabilities, 17
  examples, 20
  implementation, 12
affine transformations, 9, 12, 14, 56, 59, 63
  blending, 19
  composing, 53–54, 58
  elementary, 51–53
  implementation, 10–12, 53–55
aliasing, 44
Alice in Wonderland, 149, 260, 269
ambiguous codes, 214, 217
AND operator, 92, 109, 247, 288
animation of Julia sets, 39
anonymous classes, 347
anonymous functions, *see* lambda expressions
anonymous objects, 162, 349
arbitrary precision integers, *see* big integers
arbitrary-precision fractions, 327
Archimedes' recurrence formula, 325
arctan, power series, 325
arithmetic coding, 236
arithmetic shift, 101, 102, 288
ASCII, 212, 213, 216, 227, 428
  case conversion, 216
  control characters, 214
  relation to Unicode, 232
  relation to UTF-8, 233
aspect ratio, 33
asymmetric numeral systems, 236
Avogadro constant, 292
axis-aligned rectangles, 16

B

backtracking, 133–137, 145, 148, 400
Barnsley fern, 8–10, 12, 17, 19, 20, 22
base, *see* radix
BCD, *see* binary-coded decimals
Bézier curves, 86
  coordinate transformation, 81
  cubic, 70–73
  length of, 78
  parametric form, 71
  quadratic, 80–81
  recursive subdivision, 74–78
  straightness, 75
BFS, *see* breadth-first search
big integers
  addition, 284–285
  comparison, 279, 281
  conversion, 282, 283
  division, 301–308
  hashing, 279
  implementations of, 328
  multiplication, 290–292, 296–298
  representation, 276–279, 281–283
  signed, 314–319
  subtraction, 286–288

three-way comparison, 281
big-endian order, 277
big-Oh notation, 293
big-Omega notation, 293
big-Theta notation, 293
bignums, *see* big integers
binary-coded decimals, 234
binary codes, 212–217
binary heaps, 208
binary logarithm, 105, 106, 248
binary numbers, 90, 91, 95, 100, 113, 276
binary search, 121, 293
binary trees, 215–219, 223, 227, 228, 231
inserting into, 430
representation, 217
traversing, 218, 221
bit count, *see* population count
bit length, 221, 261–263, 268, 441, 451
bit masks, 92–93, 121, 281, 282, 393, 394
bit representation of numbers, *see* two's-complement representation
bit rotation, 94
bit sets, 112, 177, 180–183, 398, 404, 407
complement, 110
constructing, 107–109
large, 117–119
membership tests, 110
overview, 107
set operations, 109–112
bit shifts, 92–94, 99–101, 107–109, 127, 128, 248, 385, 394
bit streams, 228, 230
implementation, 231–232
padding, 229, 232
bit tricks, 122
bit width, 97, 221
bitboards, 126–129, 133, 138, 146, 180
bitlen function, *see* bit length
bitmap images, 90–94, 122
bits
iterating over, 92, 112–115
reading, 90–93
blinker life form, 41
Boolean formula, 148
borrow term, 286–288, 456
bounding box, 16
boxing of primitive data types, 337
breadth-first search, 151, 162, 164, 166, 170, 185, 188, 193, 200, 417, 425
implementation, 157–158
memory consumption, 186
overview, 156–157
relation to Dijkstra's algorithm, 197–198
visiting all layers, 160
visiting all nodes, 158, 159
BreadthFirstSearch, 157
Buddhabrot, 35, 36, 38
byte order, 277
bzip2 program, 268

C

Caesar cipher, 216
call stack, 48, 62, 63
carriage return, 214
carry term, 284–286, 288
Cartesian coordinates, 4, 18, 37
ceiling operator, 102, 103, 106, 331
cellular automata, 44, 117
elementary, 116–117
center of a graph, 162, 163
chaos game
for iterated function systems, 7–8, 13
implementation, 4–6, 14
overview, 2–4
theory of, 21–22
variations, 7, 17–19
visualizing point sets, 6–7, 19
ChaosGame, 3, 4
Chen-Ho encoding, 436
chess, use of bitboards, 146
ChooseIndex, 13–15
Chudnovsky formula, 330
Chudnovsky, David, 330
Chudnovsky, Gregory, 330
Civilization, computer game, 207
classes in Java, 333
code point in Unicode, 233
codewords, 211–215, 217–219, 229
ambiguous, 214
representation, 217
coding theory, 235, 274
collection classes, 343–344
collision in a hash table, 254
combinatorial explosion, 146
combinatorial games, 147

combinatorial problems, 146
comparators, 201, 352
complement, *see* set operations
complement in a positional number system, 287
complex numbers, 24–25, 30, 33
complexity classes, 209
complexity theory, 209
components of a graph, 186
compressing large files, 229, 266
compression algorithms, 268, 273
  see also Huffman coding, LZ4, LZ77, LZH, LZSS
computer arithmetic, 327, 328
  see also big integers, integer overflow, sign-magnitude representation, two's-complement representation
computer graphics, 70
computer-aided design, 86
Connect 4, 133
constraint propagation, 148
constraint satisfaction problem, 148
consumers, 349–350
contraction factor, 22
contractive functions, 21, 22
control characters, 214
control points, 71, 72, 81
convex hull, 72
cosine transform, 273
covariant return types, 185
cyclic buffers, 246–248, 441

D

Daft Punk, 237
data classes, 338, 339
data compression, 212, 235, 272
de Casteljau's algorithm, 74–76, 78, 80, 81
decanting problems, *see* water measurement problems
decimal expansion, 276, 322, 323
decimal numbers, 276, 277
decision problems, 209
declets, 235
Deflate algorithm, 273
deflation of Penrose tilings, 82
densely packed decimal encoding, 436
depth first search
  see also backtracking
depth of a node, 223
depth-first search, 134
design patterns, 183
  see also flyweight design pattern, iterators
determinants, 17
diameter of a graph, 162, 163, 169
dictionary region, 238, 248, 251, 255, 267
  indexing, 251–254
  searching, 250
  typical size, 251
dictionary-based compression, 237
die-hard graph, 159, 162, 169
  implementation, 153–154
  interior nodes, 156
  layers, 160
  overview, 151–152
  properties, 163–164
  visualization, 152
die-hard problem
  see also water measuring problems
  algorithmic solution, 167
  manual solution, 150
  overview, 149–151
  variations, 156, 160, 164
DIJKSTRA, 198
Dijkstra's algorithm, 197, 202
  generalization, 208
  implementation, 200–202
  overview, 197–200
  performance, 208
  visualization, 199
directed edges, 151
distance
  between points, 7
  between points and geometric objects, 79–80
distributive laws, 130, 132
DIVIDE, 300, 301, 304, 308, 448
divide-and-conquer method, 45–48, 120, 294, 320
DIVIDE-DIGIT, 301, 303, 306, 308
DIVIDE-LONG, 304–306, 308
divisibility, 322
division, 298–302
  by zero, 306
  of big integers, 301–308
  of signed integers, 318–319
double-ended queue, 254, 343, 344
dragon curve, 86
durations, computing with, 323–324

E

eccentricity of a node, 162–164
Eddington's number, 277
edges in a graph, 151
elementary cellular automata, 116, 117
Elias delta code, 274
Elias gamma code, 268, 274
emoticons, 233, 234
endianness, 277
entropy, 235
entropy codes, 235, 236, 273
equilateral triangle, 6
escape radius, 26–28, 31, 35, 36
escape time, 26, 28, 30, 31, 39, 360
EscapeTime, 30
Euler's theorem, 172
Eulerian cycle, 172
ExtendedChaosGame, 14

F

factorial function, 140
fast Fourier transform, 328, 329
Fibonacci heaps, 208
Fibonacci numbers, 59–61, 142
Fibonacci trees, 59–61
finite state entropy, 273
first-order functions, 346
fixed-length codes, 212–214
fixed-point numbers, 327
fixed-width integers, 275, 327
floating-point numbers, 327, 328
flood fill, 171–172
floor division, 101, 102, 331
floor operator, 101, 103, 331
flyweight design pattern, 183
four color problem, 173
Fourier transform, 273
fractal flame algorithm, 21
fractals, 2, 8, 13, 39, 42, 43, 78
  see also Hilbert curves, iterated function systems, Julia sets, Koch snowflage, Mandelbrot set
fractions, 327
From-Radix, 309–311, 450
From-Radix', 309, 310
functional interfaces, 158, 346, 347, 349–352
functional programming, 345–352
functional programming languages, 63

G

Game of Life, 40–42, 44, 116
generics, 337, 342–344
geometric mean, 325
geometric transformations, 50, 51, 53
  see also affine transformations
getter methods, 338
GIF file format, 273
glider gun, 42
glider life form, 41
golden ratio, 83
Gosper curve, 86
graph algorithms, 173
  see also breadth-first search, Dijkstra's algorithm
graph theory, 151, 172, 173
graphs, 151, 169
  see also die-hard graph, Hanoi graph, move graph, push graph, wolf-goat-cabbage graph
  representation, 152–153
  visualization, 151
gzip program, 273

H

Hamiltonian cycles, 156, 170, 412
Hamiltonian paths, 170, 412
Hanoi, 368
Hanoi graphs, 169–170
harmonic mean, 325
Harvey-van der Hoeven multiplication, 329
hash codes, 119, 253, 339
hash functions, 253–255, 279
hash tables, 152, 339, 454
hexadecimal numbers, 91, 276
higher-order functions, 159, 346
Hilbert curves, 65–70, 86
histograms, 36, 38, 360
Horner's scheme, 255, 309, 311
Huffman codes, 226, 260, 268, 269, 272, 273
  compared to other codes, 230, 231, 235, 236
  for byte sequences, 228–230
  for Lempel-Ziv tokens, 263–266
  limitations, 237
  overview, 223–225
Huffman trees, 225–227, 441, 443
  large, 260
  maximal height, 230
Huffman's algorithm, 225–227, 237, 431, 441
  deterministic implementation, 227

implementation, 226–228
worst case, 431
HuffmanTree, 225, 226, 431

I

identity matrix, 51
identity transformation, 51, 53, 55
IFS, *see* iterated function systems
imaginary part, 24
imaginary unit, 24, 25
immutable classes, 280, 283
importing classes, 334
importing static members, 335
induction, 170, 288, 291, 353, 369, 410
infinite sequences, 23, 25, 26
information theory, 235
integer overflow, 97, 285, 316–317, 380, 381
integer powers, 320–322, 381
integer types, 90, 96
integers modulo m, 327
interfaces, 340–342
interpolation, *see* linear interpolation
intersection
see also set operations
of axis-aligned rectangles, 16
iterated function systems, 7, 8, 21
see also affine iterated function systems
iteration, 1, 45–48, 62–63
iteration limit, 28, 31, 38, 360
iterators, 113–114, 344–345

J

Java, 333–352
jealous couples problem, 168–169
Julia sets, 39–40, 43

K

Karatsuba, 295
Karatsuba multiplication, 294–298, 320, 328
Kernighan's algorithm, 381
Koch snowflake, 78

L

lambda expressions, 115, 162, 347–349, 351
modifying local variables, 348
layers of a graph, 160, 196, 401
leading zeros, *see* number of leading zeros
least significant 1-bit, 112, 115
Lempel-Ziv compression, 238
Lempel-Ziv-Markov chain algorithm, 273
lerp function, 5, 7, 19, 21, 73, 75, 81
Life, *see* Game of Life
life forms, 40–41
line feed, 214
linear interpolation, 5, 19–21, 359
see also lerp function
linear transformations, 10
effect on area, 17
lists, reversing, 48
little-endian order, 277, 278
locality-preserving mapping, 86
logarithm, binary, *see* binary logarithm
longest shortest path, *see* diameter
lookahead region, 238, 248, 250
lossless compression, 212, 273
lossy compression, 273
LWSS life form, 41
LZ4 compression, 270–272
LZ77, 238, 273
bit encoding, 256–268
compression, 239–240, 243
decompression, 241–243
finding the longest match, 251–256
generating tokens, 248–250
token representation, 244
tokens, 239, 246
LZ77-Compress, 241, 243, 439
LZ77-Decompress, 243, 439
LZ78 compression, 273
LZH encoding, 263–268
LZMA compression, 273
LZSS encoding, 256–259, 268, 269
LZW compression, 273

M

Machin's formula, 325, 329, 330
Mandelbrot iteration, 26, 27, 30, 33, 35, 360
Mandelbrot set, 23–24, 26, 29, 35, 43
see also Buddhabrot, Julia sets
coloring, 26–29, 33
computing, 24–26
drawing, 30–33
landmarks, 29
Mandelbrot-BW, 26
Mandelbrot-Color, 30
master theorem, 320

memoization, 137, 140–144
memory partitioning, 104
method overriding, 339–341
method references, 347, 351
midpoint, 5, 374, 377
minimum in a list, 46
mixed-radix system, 323
modulo operator, 68, 247, 255, 362, 429
Morse code, 211, 212
move graph, 183–186, 188, 192, 196, 417
multigraphs, 155
multiplication
  algorithms compared, 328–329
  Karatsuba's algorithm, 294–298
  of big integers, 290–292
  standard algorithm, 288–292
MULTIPLY, 290, 291, 294, 295
MULTIPLYBYDIGIT, 289, 291, 446

N

Nebulabrot, 38, 361
Nim, 143–145, 147
Nim sum, 144, 398
Nimatron, 147
nlz function, *see* number of leading zeros
nodes in a graph, 151
normalization, 137, 138, 144, 404, 406
NOT operator, 110
NP-hard problems, 148, 162
ntz function, *see* number of trailing zeros
number of leading zeros, 103, 105, 106, 122, 222
number of trailing zeros, 103, 112, 113, 121

O

object-oriented design, 183
OR operator, 93, 107, 109
overflow, *see* integer overflow

P

P (complexity class), 209
packages in Java, 333
pandigital numbers, 452
parametric curves, 71, 79
  examples, 84
  plotting, 84–85
path finding, 185, 190–192, 202–204, 207, 208
  see also breadth-first search, Dijkstra's algorithm, A* algorithm
path in a graph, 152, 186, 190
  length, 152, 166
  longest shortest, 162
  representation, 164–166
  shortest, 164–167, 174, 175, 185, 192, 418
path in a weighted graph
  representation, 200
  shortest, 194, 197–202
  weighted length, 194
PBM file format, 94
Peano curve, 86
peg solitaire, 125, 142, 146
  computing moves, 129–132
  counting solutions, 137, 141
  difficulty of, 141, 142, 395
  representation of boards, 126–127
  solving, 134, 136
  symmetries, 137, 138
  variations, 132
Penrose tilings, 81–83, 87
period of decimal expansion, 322, 323
periodic boundary conditions, 41, 116, 117
pi, computing, 324–327, 329–330
plants, modeling of, 59
PNG file format, 273
points vs. vectors, 4
polar coordinates, 18, 37, 59
polydivisible numbers, 322
polynomial time, 209
population count, 115, 119–121, 451
position vectors, 4
positional number systems, 276
powers of 2, 105
  bit patterns, 99
  bit patterns of negative, 99
  computing with, 100–107
  dividing by, 101–102, 385
  divisibility, 102–103
  multiplying by, 100
  partitioning memory, 104
  testing for, 115
predecessor on a path, 164, 166, 198, 200
predicates, 157–160, 166, 167, 202, 351–352
prefix codes, 214–217, 219, 223, 231, 234
  see also Huffman codes
  bit encoding, 221–222
  converting representations, 218, 219
  decoding symbols, 215, 220

encoding symbols, 220
implementation, 217–222
optimal, 223
tree representation, 215, 216, 223
prefix of a string, 214, 215
prefix-free codes, *see* prefix codes
primitive types, 246, 335–337
priority queues, 198, 201, 227, 228, 344
data structures for implementing, 208
PSPACE-completeness, 209
pure function, 141
push graph, 186–189, 191, 194, 418
visualization, 187
Pythagoras trees, 48–51, 56–59
Pythagorean theorem, 49, 353

Q

queues, 156, 157, 197
quotient, 299

R

R-pentomino life form, 41
radius of a graph, 162, 163
radix, 276, 281, 283, 299, 303, 305
conversion, 308–313
for big integers, 276–277
multiplying by power of, 294, 297
radix conversion, 313, 448
random number generators, 6, 14, 37, 351
random numbers, 4, 13–15
random point inside a circle, 37
rasterization, 43
rational numbers, 322
real part, 24
records, 339–340
recursion, 45–48, 57–63, 78, 80, 133, 143, 368, 369, 400, 410
see also backtracking, divide-and-conquer method, memoization
eliminating, 122
implementation of, 62
infinite, 377, 447
optimization of, 63, 137, 142
relation to trees, 65
runtime analysis, 320
reflection of points, 353
regular polygons, 18
remainder, 101, 299
reversing bytes, 94
Rhind Papyrus, 329
river-crossing puzzles, 173
ROT13 cipher, 429
rotation, 10, 53
rotation matrix, 53
rule of a cellular automaton, 116
runtime complexity, 291, 295, 320
Russian peasant multiplication, 328

S

Sardinas-Patterson algorithm, 236
satisfiability problem, 148
scaling, 10, 53
scene graphs, 63
Schönhage-Strassen multiplication, 328, 329
self-similarity, 3, 67, 356
set operations, 109–112
setter methods, 338
seven bridges of Königsberg, 172
Shannon-Fano codes, 230–231
shearing, 10
Sierpiński triangle, 2, 3, 6, 8, 116, 354, 410
sign extension, 97, 101
sign-magnitude representation, 314–319, 454
simple graphs, 155
singleton sets, 107
snowflake, *see* Koch snowflake
Sokoban, 175–204
computational complexity, 209
deadlocks, 189–190
difficulty of, 176, 209
Level 1, 176
moving crates, 186–190
moving the player, 182–185
overview, 175–176
representing levels, 177–182
solving, 190–194, 202–204
text encoding of levels, 180–181
source coding theorem, 235
sources in a graph, 401
space-filling curves, 68, 86
spaceships in Life, 41, 362
spreadsheet, cell names, 313
squared length, 31
stack overflow, 63
subdivision surfaces, 86
SUBTRACT, 286–288, 291

subtraction, 286–288
Sudoku, 145, 148
suppliers, 350–351
symmetric difference, 381
systems programming, 122

T

tabulator, 214
tail-call optimization, 63
telegraphy, 211
tessellations, 81
three-way comparison, 281, 319, 352
tilings, 81, 87
TO-RADIX, 313, 450
Toom-Cook algorithm, 328
Towers of Hanoi, 61–63, 169, 170
trailing zeros, *see* number of trailing zeros
translation, 10, 51, 53
Turing completeness, 41
two's-complement representation, 95–100, 105, 112, 115, 287, 380, 381
  asymmetry, 96, 315
  changing bit width, 97
  common bit patterns, 97–99
  negation, 96
  overflow, 316–317
  trailing zeros, 103
type casts, 283, 336
type inference, 347
type promotion, 335

U

unambiguous codes, 217, 236
unary code, 217, 261, 268, 274
unboxing of primitive data types, 337
undirected edges, 151
Unicode, 232–234
union, *see* set operations
unit circle, 37
universal codes, 274
universe, age of the known, 369
UTF-8, 232–234

V

variable-length codes, 214, 233
  see also variable-length integers
variable-length integers, 261, 263, 268, 274, 441
vertices in a graph, *see* nodes in a graph
visibility of fields and methods, 335

W

water measuring problems, 169, 173
  see also die-hard problem
wavelet transforms, 273
Weierstrass function, 42
weighted average, 5
weighted graphs, 194, 197, 198
  see also Dijkstra's algorithm, paths in a weighted graph
weighted push graph, 194–196, 202
whirl patterns, 59
winning strategy, 143, 147
wolf-goat-cabbage problem, 168
word ladders, 204–205
word-ladder graph, 204
wrapper classes, 336–337

X

XOR operator, 112, 144, 147
xz program, 268, 273

Z

z-order curve, 86
ZIP file format, 273
zstandard compression, 273