# Index

% operator, **24**, 242
== operator, 25, 138
^ operator, 412

Abelson, Hal, 223
abstract class, **80–81**, 313–314, 323
abstract data type, 62
  deque, 248–249
  graph, 612–618
  map, 402–404
  partition, 672–675
  positional list, 272–275
  priority queue, 361
  queue, 239–240
  sorted map, 428
  stack, 227–228
  string, 17–18
  tree, 312–314
abstract methods, 80
**abstract** modifier, 11, 81
AbstractBinaryTree class, **319–320**, 323, 325, 330, 339, 341, 342
AbstractHashMap class, 406, 422–424
abstraction, 62
AbstractMap class, 384, **406–407**, 408, 422
AbstractPriorityQueue class, **364–365**, 366
AbstractSortedMap class, 406, **430**, 466
AbstractTree class, **313–316**, 323, 330, 339–342
$(a,b)$ tree, 702–704
access frequency, 294
accessor method, 5
activation record, ***see*** frame
acyclic graph, 615
adaptability, 60, 61
adaptable priority queue, 390–392, 658, 659
adapter design pattern, 233, 245
Adel'son-Vel'skii, Georgii, 479, 530
adjacency list, 619, 622–623
adjacency map, 619, 624, 626
adjacency matrix, 619, 625
Aggarwal, Alok, 709
Aho, Alfred, 256, 305, 530, 610
Ahuja, Ravindra, 686
algorithm analysis, 164–181
alphabet, 17, 575
amortization, 205, **266–269**, 376, 672–675
ancestor, 310
antisymmetric property, 363
Apache Commons, 448
API, 76, 228
arithmetic operators, 24
arithmetic progression, 71, 268
Arnold, Ken, 57
array, 20–21, 104–119
  dynamic, 263–269
array list, 260–265
ArrayDeque class, 251
ArrayIndexOutOfBounds exception, 20, 33, 84, 87
ArrayList class, **260–261**, 263–265, 283–285, 290
ArrayQueue class, **242–244**, 302
Arrays class, **112**, 114, 139, 175
ArrayStack class, 230–232, 300
associative array, 402
asymptotic notation, 164–177
  big-Oh, 164–167
  big-Omega, 167, 265
  big-Theta, 167
autoboxing, 19, 92
AVL tree, 479–486

back edge, 639, 680
Baeza-Yates, Ricardo, 530, 572, 709
BalanceableBinaryTree class, 476–478
Barůvka, Otakar, 683, 686
base class, 64
base type, 4
Bayer, Rudolf, 530, 709
Bellman, Richard, 610
Bentley, Jon, 223, 400, 572
best-fit algorithm, 692
BFS, ***see*** breadth-first search
biconnected graph, 681
big-Oh notation, 164–167
big-Omega notation, 167, 265
big-Theta notation, 167
binary heap, 370–384
binary search, **196–197**, 203–204, 429–432, 563
binary search tree, 338, **460–478**
  rotation, 472
  trinode restructuring, 473
binary tree, 317–330, 533
  array-based representation, 331–332
  complete, 370
  improper, 317
  level, 321
  linked structure, 323–330
  proper, 317
BinaryTree interface, 319
bipartite graph, 681
bit vector, 456

Copyright © 2014. Wiley. All rights reserved.

Booch, Grady, 101, 305
bootstrapping, 424, 502
Boyer, Robert, 610
Boyer-Moore algorithm, 578–581
Brassard, Gilles, 188
breadth-first search, 640–642
breadth-first tree traversal, 336, 341–342
**break** statement, 32, 37
brute force, 576
B-tree, 704
bubble-sort, 304
bucket-sort, 558–559, 562
Budd, Timothy, 101, 305
buffer overflow attack, 20
Burger, Doug, 709

caching, 695–700
Caesar cipher, 115–117
Carlsonn, Svante, 400
casting, 28–29, 88–90
  implicit, 29
**catch**, 82
catching an exception, 82–84
ceiling function, **163**
central processing unit (CPU), 151
ChainHashMap class, 406, 424–425
character, 17
checked exception, 86
Chen, Wen-Chin, 458
Chernoff bound, 570
child class, ***see*** subclass
circular queue, 246–247
circularly linked list, **128–131**, 246
Clarkson, Kenneth, 572
class, 2, 5–22, 60, 62
  abstract, **80–81**, 313–314
  base, 64
  child, 64
  inner, 96, 284
  nested, 96
  outer, 96
  parent, 64
  sub, 64
  super, 64
class diagram, 47
ClassCastException, 87, **89**
clone method, 141–144
Cloneable interface, 79, 141, 144, 302, 303, 353
cloning, 141–144
clustering, 419
coding, 46
Cole, Richard, 610
Collection interface, 288
collections, ***see*** Java collections framework
collision resolution, 411, 417–419
Comer, Douglas, 709
comparability property, 363
Comparable interface, 79, 363
Comparator interface, 363, 538
complete binary tree, 370
complete graph, 678
composition design pattern, 91, 295
compression function, 411, 416
concatenation, 17, 24
concrete methods, 80
ConcurrentSkipListMap class, 436
connected components, 615, 635, 638
constructor, 14
**continue** statement, 37
contradiction, 178
contrapositive, 178
control flow, 30–37
core memory, 695
Cormen, Thomas, 530, 686
Cornell, Gary, 57
CPU, 151
CRC cards, 47
CreditCard class, **41–43**, 47, 50–51, 65–68, 88–89
Crochemore, Maxime, 610
Crosby, Scott, 458
cryptography, 115–117
cubic function, 160
cuckoo hashing, 456
currentTimeMillis method, 113, 151
cyber-dollar, 266–267, 495–498, 673
cycle, 615
  directed, 615
cyclic-shift hash code, 413–414

DAG, ***see*** directed acyclic graph
data packets, 304
de Morgan's law, 178
debugging, 46
decision tree, 317, 461, 556
decrease-and-conquer, 563–565
decryption, 115
default constructor, 6, 14
degree of a vertex, 613
delimiter, 40, 235
Demurjian, Steven, 101, 256
denial-of-service attack, 421
depth of a tree, 314–316
depth-first search (DFS), 631–639
deque, 248–251
  abstract data type, 248–249
  linked-list implementation, 250
Deque interface, 288
descendant, 310
design patterns, 49, 63
  adapter, 233, 245
  amortization, 266–269
  brute force, 576
  composition, 91, 295, 362
  divide-and-conquer, 532–536, 544–545
  dynamic programming, 598–604
  factory method, 325, 477
  greedy method, 597
  iterator, 282–286
  position, 272–275
  prune-and-search, 563–565
  template method, 81, 446, 475
DFS, ***see*** depth-first search
Di Battista, Giuseppe, 358, 686
diameter, 355
dictionary, ***see*** map
Dijkstra's algorithm, 653–661
Dijkstra, Edsger, 223, 686

Copyright © 2014. Wiley. All rights reserved.

directed acyclic graph, 647–649
disk usage, 198–201, 204–205, 345
divide-and-conquer, 532–536, 544–545
division method for hash codes, 416
dot operator, 7
double hashing, 419
double-ended queue, ***see*** deque
doubly linked list, 125, 132–137
DoublyLinkedList class, **135–137**, 250, 271, 276
down-heap bubbling, 374
dynamic array, 263–269
  shrinking, 269
dynamic dispatch, 68
dynamic programming, 598–604

Eades, Peter, 358, 686
edge, 310
  destination, 613
  endpoint, 613
  incident, 613
  multiple, 614
  origin, 613
  outgoing, 613
  parallel, 614
  self-loop, 614
edge list, 619–621
edge of a graph, 612
edge relaxation, 653
edit distance, 608
element uniqueness problem, 174–175, 215
encapsulation, 62
encryption, 115
endpoints, 613
enum, 22
equals method, 25, 138–140
equivalence relation, 138
equivalence testing, 138–140
erasure, 140
Error class, 86, 87
Euclidean norm, 56
Euler tour of a graph, 677, 681
Euler tour tree traversal, 348–349, 358

evolvability, 61
exception, 82–87
  catching, 82–84
  checked, 86
  throwing, 85–86
  unchecked, 86
Exception class, 86, 87
exponential function, 161–162, 209–210
expression, 23–29
expression tree, 318
external memory, 695–707, 709
external-memory algorithm, 695–707
external-memory sorting, 705–707

factorial function, 191–192, 202, 690
factory method pattern, 325, 477
fail-fast iterator, 284, 304
favorites list, 294–299
FavoritesList class, 295–296
FavoritesListMTF class, 298, 399
Fibonacci heap, 659
Fibonacci series, 73, 180, 186, 216–217, 480
field, 5
FIFO, ***see*** first-in, first-out
File class, 200
file system, 198–201, 310, 345
**final** modifier, 11
first-fit algorithm, 692
first-in, first-out (FIFO) protocol, **238**, 255, 336, 360, 699–700
Flajolet, Philippe, 188
Flanagan, David, 57
floor function, **163**, 209
flowchart, 31
Floyd, Robert, 400, 686
Floyd-Warshall algorithm, 644–646, 686
for-each loop, 36, 283
forest, 615
fractal, 193
fragmentation of memory, 692
frame, 192, 688

free list, 692

game tree, 336, 358
Gamma, Erich, 101
garbage collection, 232, 693–694
  mark-sweep, 693
Gauss, Carl, 159
generics, 91–95, 126, 228
geometric progression, 72, 267
geometric sum, 162
Gonnet, Gaston, 400, 530, 572, 709
Goodrich, Michael, 709
Gosling, James, 57
Graham, Ronald, 686
graph, 612–686
  abstract data type, 612–618
  acyclic, 615, 647–649
  breadth-first search, 640–642
  connected, 615, 630
  data structures, 619–629
    adjacency list, 619, 622–623
    adjacency map, 619, 624, 626
    adjacency matrix, 619, 625
    edge list, 619–621
  depth-first search, 631–639
  directed, 612, 613, 647–649
  mixed, 613
  reachability, 643–646
  shortest paths, 651–661
  simple, 614
  strongly connected, 615
  traversal, 630–642
  undirected, 612, 613
  weighted, 651–686
greedy method, 597, 652, 653
Guava library, 448
Guibas, Leonidas, 530
Guttag, John, 101, 256, 305

Harmonic number, 171, 221
hash code, 411–415
  cyclic-shift, 413–414
  polynomial, 413, 609
hash table, 410–427
  clustering, 419
  collision, 411

Copyright © 2014. Wiley. All rights reserved.

collision resolution, 417–419
double hashing, 419
linear probing, 418
quadratic probing, 419
hashing
cuckoo, 456
power-of-two-choices, 457
header sentinel, 132
heap, 370–384
bottom-up construction, 380–384
heap-sort, 388–389, 561
HeapAdaptablePriorityQueue class, 392–394
HeapAdaptablePriorityQueue class, 659
HeapPriorityQueue class, 377–378, 382
height of a tree, 315–316, 471
Hell, Pavol, 686
Hennessy, John, 709
heuristic, 297
hierarchy, 64
Hoare, C. A. R., 572
Holmes, David, 57
hook, 466, 475
Hopcroft, John, 256, 305, 530, 686
Horner's method, 187
Horstman, Cay, 57
HTML, 235–237, 253, 574
Huang, Bing-Chao, 572
Huffman coding, 595–596

I/O complexity, 701
identifier, 2
IllegalArgumentException, 85, 87
immutable, 18
implicit cast, 29
**import** statement, 45
in-degree, 613
in-place algorithm, 389, 553
incoming edges, 613
index, 17, 20
IndexOutOfBoundsException, 259
induction, **179–180**, 203
infix notation, 356
inheritance, 64–74
multiple, 79
single, 66
inner class, 96, 284
inorder tree traversal, 337, 341, 473
insertion-sort, 110–111, 293–294, 387, 561
instance, 5, 60
instance variable, 5, 60
**instanceof** operator, 68, 89
integrated development environment (IDE), 16, 49
interface, 62, 76–79, 90, 228
internal memory, 695
Internet, 304
inversion, 387, 561, 569
inverted file, 456
isomorphism, 352
Iterable interface, 36, 283
iterator, 282–286
fail-fast, 284, 304

JáJá, Joseph, 358
Jarník, Vojtěch, 686
Java, 2–57, 60–96
arrays, 20–21, 104–119
casting, 88–90
control flow, 30–37
exceptions, 82–87
expressions, 23–29
input, 38–40
method stack, 688–690
methods, 12–13
output, 38–40
packages, 44–45
Java collections framework, 251, **288–292**, 384, 445–448
Java Virtual Machine (JVM), 688–693
javadoc, 50
Jones, Richard, 709
Josephus problem, 246

Karger, David, 686
Karp, Richard, 358, 609
Klein, Philip, 686
Kleinberg, Jon, 572
Klink, Alexander, 458
Knuth, Donald, 148, 188, 305, 358, 400, 458, 530, 572, 610, 686, 709
Knuth-Morris-Pratt algorithm, 582–585
Kosaraju, S. Rao, 686
Kruskal's algorithm, 667–675
Kruskal, Joseph, 686

Landis, Evgenii, 479, 530
Langston, Michael, 572
last-in, first-out (LIFO) protocol, **226**, 228
lazy iterator, 284
LCS, ***see*** longest common subsequence
leaf of a tree, 310
least recently used (LRU) protocol, 699–700
Lecroq, Thierry, 610
Leiserson, Charles, 530, 686
Lesuisse, R., 223
level in a tree, 321
level numbering, 331, 371
Leveson, Nancy, 101
lexicographic order, 363, 559
LIFO, ***see*** last-in, first-out
linear function, 158
linear probing, 418
linearity of expectation, 565
linked list, 122–137, 233, 245
circularly linked, 128–131, 246
doubly linked, 125, 132–137, 250, 276–280
singly linked, 122–127, 233, 245
LinkedBinaryTree class, **325–330**, 466, 476–477
LinkedHashMap class, 454
LinkedList class, 251, 288, **289**, 290
LinkedPositionalList class, **276–280**, 286–287, 620
LinkedQueue class, **245**, 341, 541, 549
Lins, Rafael, 709
Liotta, Giuseppe, 358, 686
Liskov substitution principle, 68
Liskov, Barbara, 68, 101, 256, 305
list
of favorites, 294–299

Copyright © 2014. Wiley. All rights reserved.

positional, 270–281
List interface, **258–259**, 284, 288
literal, 23
Littman, Michael, 572
live objects, 693
load factor, 417, 420–421
locality of reference, 297, 697
log-star function, 675
logarithm function, **156–157**
longest common subsequence, 601–604
looking-glass heuristic, 578
lookup table, 410
loop invariant, 181
lowest common ancestor, 355

Magnanti, Thomas, 686
main memory, 695
map, **402–444**
abstract data type, 402–404
binary search tree, 460–478
hash table, 410–427
skip list, 436–444
sorted, **428–435**, 460
Map interface, 406
mark-sweep algorithm, 693
matrix, 118
matrix chain-product, 598–600
maximal independent set, 682
McCreight, Edward, 610, 709
McDiarmid, Colin, 400
McIlroy, Douglas, 572
median, 196, 555, 563, 571
Megiddo, Nimrod, 572
Mehlhorn, Kurt, 530, 686, 709
member of a class, 5
memory address, 688
memory allocation, 692
memory heap, 691
memory hierarchy, 695
memory management, 688–694
merge-sort, 532–544, 562
multiway, 705–707
mergeable heap, 530
method, 2, 12–13, 60
abstract, 80
concrete, 80
signature, 12
minimum spanning tree, 662–675
Kruskal's algorithm, 667–675
Prim-Jarnik algorithm, 664–666
mixin, 79
modularity, 62
modulo operator, **24**, 116, 242
Moore, J. Strother, 610
Morris, James, 610
Morrison, Donald, 610
Motwani, Rajeev, 458, 572
move-to-front heuristic, 297–299
MST, ***see*** minimum spanning tree
multimap, 445, 448–450
multiple inheritance, 79
Multiply-Add-and-Divide (MAD), 416
multiset, 445, 447–448
multiway merge-sort, 705–707
multiway search tree, 500–502
Munro, J. Ian, 400

*n*-log-*n* function, 158
narrowing conversion, 88
natural join, 304
natural ordering, 363
nested class, 96
nested loops, 159
next-fit algorithm, 692
node, 309
ancestor, 310
child, 309
descendant, 310
external, 310
internal, 310
leaf, 310
parent, 309
root, 309
sibling, 310
node of a graph, 612
NoSuchElementException, 86, 87, 240, 251, 282
**null** value, 6, 7, 21, 23
NullPointerException, **7**, 87
Number class, 89
NumberFormatException, 28, 84, 85, 87
object, 5–22, 60
Object class, **66**, 91, 138, 141
object-oriented design, 60–101
open addressing, 418
operand stack, 690
order statistic, 563
Orlin, James, 686
out-degree, 613
outer class, 96
outgoing edge, 613
override, 64

*p*-norm, 56
package, 10, 44–45
palindrome, 222, 606
parameter passing, 13
parent class, 64
parent node, 309
parenthetic string representation, 346
partition, 670, 672–675
path, 310, 615
compression, 675
directed, 615
length, 352, 652
simple, 615
pattern matching, 576–585
Boyer-Moore algorithm, 578–581
brute force, 576–577
Knuth-Morris-Pratt algorithm, 582–585
Rabin-Karp algorithm, 609
Patterson, David, 709
permutation, 191
Peters, Tim, 562
polymorphism, 68
polynomial function, 160, 187
polynomial hash code, 413, 609
portability, 61
position, 272–275, 312, 437
Position interface, **274**, 313, 325
positional list, 270–281
abstract data type, 272–280
PositionalList interface, **275**, 293, 295
postfix notation, 253, 356
postorder tree traversal, 335
power function, 209

Copyright © 2014. Wiley. All rights reserved.

power-of-two-choices hashing, 457
Pratt, Vaughan, 610
PredatoryCreditCard, 65–68, 88–89
prefix average, 175–177
prefix code, 595
prefix of a string, 575
preorder tree traversal, 334
Prim, Robert, 686
Prim-Jarnik algorithm, 664–666
primitive operations, 154
primitive type, 4
priority queue, 360–400
  adaptable, 390–392, 658
  ADT, 361
  heap implementation, 372–379
  sorted list implementation, 368–369
  unsorted list implementation, 366–367
priority search tree, 400
**private** modifier, 10
ProbeHashMap class, 406, 426–427
program counter, 689
progression
  arithmetic, 71, 268
  Fibonacci, 73
  geometric, 72, 267
**protected** modifier, 10, 67
prune-and-search, 563–565
pseudocode, 48
pseudorandom number generator, 113–114, 437
**public** modifier, 9
Pugh, William, 458
puzzle solver, 212–213

quadratic function, 158
quadratic probing, 419
queue, 238–247
  abstract data type, 239–240
  array implementation, 241–244
  circular, 246–247
  linked-list implementation, 245
Queue interface, 239, 240, 288
java.util.Queue interface, 384
quick-sort, 544–555, 562

Rabin, Michael, 609
Rabin-Karp algorithm, 609
radix-sort, 559–560, 562
Raghavan, Prabhakar, 458, 572
Ramachandran, Vijaya, 358
Random class, 53, **113**, 437
randomization, 421, 437, 442–444, 551–552, 564–565
randomized quick-select, 564
randomized quick-sort, 551
reachability, 615, 630
recurrence equation, 203, **540**, 565, 705
recursion, **190–220**, 314–316, 334–335, 344–349, 461–462, 532, 540, 563, 690
  binary, 211
  depth limit, 218, 525
  linear, 206–210
  multiple, 212–213
  tail, 219–220
  trace, 192, 202, 690
red-black tree, 510–524
Reed, Bruce, 400
reference type, 6
reference variable, 6
reflexive property, 363
rehashing, 420
reusability, 60, 61
Rivest, Ronald, 530, 686
robustness, 60
root objects, 693
root of a tree, 309
round-robin scheduling, 128
running time, 150
RuntimeException, 87

Samet, Hanan, 709
Scanner class, **39–40**, 45, 86
Schaffer, Russel, 400
scheduling, 399
Scoreboard class, 105–109
search engine, 594
search table, 429–432
search tree, 460–530
Sedgewick, Robert, 400, 530
seed, 113, 437
selection problem, 563–565
selection-sort, 386
self-loop, 614
sentinel, 132–133
separate chaining, 417
sequential search, 196
set ADT, 445–447
Sharir, Micha, 358
short-circuit evaluation, 33
shortest path, 651–661
  Dijkstra's algorithm, 653–661
  tree, 661
sieve algorithm, 453
signature, 7, 12, 14
single inheritance, 66
singly linked list, 122–127, 233, 245
SinglyLinkedList class, **126–127**, 140, 144
skip list, 436–444
Sleator, Daniel, 530
snapshot iterator, 284, 320, 340
sort method, 175
sorted map, 428–435, 460
  abstract data type, 428
  search table, 429–432
SortedMap interface, 406
SortedPriorityQueue class, 368–369
SortedTableMap class, 406, 429–432
sorting, 110, 385–389, **532–560**
  bucket-sort, 558–559
  external-memory, 705–707
  heap-sort, 388–389
  in-place, 389, 553
  insertion-sort, 110–111, 293, 387
  lower bound, 556–557
  merge-sort, 532–544
  priority-queue, 385–389
  quick-sort, 544–555
  radix-sort, 559–560
  selection-sort, 386
  stable, 559
  Tim-sort, 562
space usage, 150

Copyright © 2014. Wiley. All rights reserved.

spanning tree, 615, 630, 634, 635, 662
sparse array, 303
splay tree, 475, 488–499
stable sorting, 559
stack, 226–237
  abstract data type, 227–228
  array implementation, 230–232
  linked-list implementation, 233
Stack interface, 228–229
**static** modifier, 10
Stein, Clifford, 530, 686
Stephen, Graham, 610
stop words, 588, 609
string
  mutable, 18
  prefix, 575
  suffix, 575
String class, 17–18
StringBuilder class, 18, 152, 269
strong typing, 76
strongly connected components, 638
strongly connected graph, 615
subclass, 10, 64
subgraph, 615
subsequence, 601
subtree, 310
suffix of a string, 575
summation, 161
  geometric, 162
**super** keyword, **67**, 81
superclass, 64
Sussman, Gerald, 223
Sussman, Julie, 223

Tamassia, Roberto, 358, 686
Tardos, Éva, 572
Tarjan, Robert, 358, 530, 686
template method pattern, 81, 446, 475
testing, 46
text compression, 595–596
**this** keyword, **15**, 67, 96
three-way set disjointness, 173–174
**throw** statement, 85
Throwable class, 86, 87
throwing an exception, 85–86
Tic-Tac-Toe, 119, 336, 358
Tim-sort, 562
Tollis, Ioannis, 358, 686
topological ordering, 647–649
total order, 363
tower-of-twos, 675
Towers of Hanoi, 222
trailer sentinel, 132
transitive closure, 643–646
transitive property, 363
tree, 205, **307–358**, 615
  abstract data type, 312–314
  binary, ***see*** binary tree
  binary search, ***see*** binary search tree
  binary tree representation, 354
  child node, 309
  decision, 317
  depth, 314–316
  edge, 310
  expression, 318
  external node, 310
  height, 315–316
  internal node, 310
  leaf, 310
  level, 321
  linked structure, 333
  multiway, 500–502
  node, 309
  ordered, 311
  parent node, 309
  path, 310
  red-black, ***see*** red-black tree
  root node, 309
  splay, ***see*** splay tree
  traversal, 205, 334–349
    breadth-first, 336, 341–342
    Euler tour, 348–349
    inorder, 337, 341, 473
    postorder, 335, 341
    preorder, 334, 340
  $(2,4)$, ***see*** $(2,4)$ tree
TreeMap class, 406
triangulation, 608
trie, 586–594
  compressed, 590
trinode restructuring, **473**, 482, 513
**try-catch** statement, 82
Tsakalidis, Athanasios, 530
Turner, Clark, 101
two-dimensional array, 118
$(2,4)$ tree, 500–509
type, 5
type conversion, 28–29
type inference, 93

Ullman, Jeffrey, 256, 305, 530
unboxing, 19, 93
unchecked exception, 86
Unicode, 115, 575
union-find, 670, 672–675
unit testing, 54
UnsortedPriorityQueue class, 366–367
UnsortedTableMap class, 406, 408–409, 424
up-heap bubbling, 372
update method, 5

van Leeuwen, Jan, 686
vertex of a graph, 612
virtual memory, 697
Vishkin, Uzi, 358
visibility, 9
Vitter, Jeffrey, 188, 458, 709

Wälde, Julian, 458
Wallach, Dan, 458
Warshall, Stephen, 686
widening conversion, 88
Williams, J. W. J., 400
Wood, Derick, 305
worst-fit algorithm, 692
wrapper type, 19, 91, 93, 232

XML, 236, 574

Copyright © 2014. Wiley. All rights reserved.