Index

abstract data type (ADT), xiv, 8–12, 20–22, 51, 99–103, 140–147, 158, 171, 173, 206–208, 216, 218, 221, 227, 228, 292–297, 389, 394–395, 430, 446, 475, 476
abstraction, 11
accounting, 125, 134
Ackermann's function, 226
activation record, *see* compiler, activation record
aggregate type, 8
algorithm
  definition of, 18–19
algorithm analysis, xiii, 4, 57–96, 235
  amortized, *see* amortized analysis
  asymptotic, 4, 57, 58, 67–73, 99, 481
  empirical comparison, 57–58, 89, 237
  for program statements, 74–79
  multiple parameters, 83–84
  running time measures, 59
  space requirements, 58, 84–86
all-pairs shortest paths, 532–534, 551, 553
amortized analysis, 76, 119, 327, 481, 496–499, 502
approximation, 572
array
  dynamic, 118, 502
  implementation, 9, 22
artificial intelligence, 389
`assert`, xvii
asymptotic analysis, *see* algorithm analysis, asymptotic
ATM machine, 7
average-case analysis, 63–65
AVL tree, 198, 365, 447, 453–456, 475

back of the envelope, napkin, *see* estimating
backtracking, 571
bag, 26, 51
bank, 6–7
basic operation, 5, 6, 21, 22, 59, 61, 66
best fit, *see* memory management, best fit
best-case analysis, 63–65
big-Oh notation, *see* O notation
bin packing, 572–573
binary search, *see* search, binary
binary search tree, *see* BST
binary tree, 153–205
  BST, *see* BST
  complete, 154, 155, 170, 171, 181, 257

full, 154–157, 169, 170, 189, 198, 225
implementation, 153, 156, 198
node, 153, 157–158, 162–169
`null` pointers, 157
overhead, 169
parent pointer, 163
space requirements, 157, 163, 169–170
terminology, 153–155
threaded, 202
traversal, *see* traversal, binary tree
Binsort, 85, 86, 259–265, 268, 337, 557
bintree, 471, 476
birthday problem, 332, 353
block, 298, 303
Boolean expression, 565
clause, 565
Conjunctive Normal Form, 565
literal, 565
Boolean variable, 8, 30, 95
branch and bounds, 571
breadth-first search, 389, 401, 403, 404, 417
BST, xv, 171–180, 198, 200, 202, 230, 250, 256, 364–367, 369, 370, 375, 447, 452–459, 461, 470, 475, 476, 536, 541
efficiency, 179
insert, 175–176
remove, 176–179
search, 173–174
search tree property, 172
traversal, *see* traversal
B-tree, 318, 331, 358, 363, 372–382, 384, 473
analysis, 381
B^+-tree, 8, 10, 359, 363, 375–381, 384, 385, 448
B^*-tree, 379
Bubble Sort, 80, 240–243, 266, 272
buffer pool, xv, 12, 279, 289–297, 314–316, 325, 326, 365, 374, 438
ADT, 292–297
replacement schemes, 290–291, 326–328

Java
generic, 101
`new`, 115
private members, 106
cache, 282, 289–297, 325
CD-ROM, 8, 9
CD-ROM, 281, 284, 332
ceiling function, 30
city database, 151, 202, 465, 476
class, 100, *see* object-oriented programming, class
clique, 563, 568, 569, 571, 582
cluster problem, 230, 476
cluster, file, 285, 287, 288, 438
code tuning, 57–59, 87–90, 255–256
Collatz sequence, 72, 93, 94, 574, 581, 584
comparator, xiv, 16, 99
compiler, 89
activation record, 129
efficiency, 58
optimization, 89
complexity, 11
composite, *see* design pattern, composite
composite type, 8
computability, 20, 574, 581
computer graphics, 448, 459

connected component, *see* graph, connected component
contradiction, proof by, *see* proof, contradiction
cylinder, *see* disk drive, cylinder

data item, 8
data member, 9
data structure, 5, 9
  costs and benefits, xiii, 3, 6–8
  definition, 9
  philosophy, 4–6
  physical vs. logical form, xvi, 9, 12–13, 99, 181, 282, 293, 423
  selecting, 5–6
  spatial, *see* spatial data structure
data type, 9
decision problem, 561, 565, 566, 582
decision tree, 268–271
decomposition
  image space, 448
  key space, 448
  object space, 447
`delete`, *see* **C**++, **`delete`**
depth-first search, 389, 400–402, 417, 442, 502
deque, 150
dequeue, *see* queue, dequeue
design pattern, xiv, 12–16, 21
  composite, 15–16, 167, 471
  flyweight, 13–14, 167, 202, 470–471
  strategy, 16
  visitor, 14–15, 161, 401
Deutsch-Schorr-Waite algorithm, 443, 446
dictionary, 173, 345–449
  ADT, 140–146, 318, 356, 385, 529
Dijkstra's algorithm, 408–412, 419, 420, 532, 533
Diminishing Increment Sort, *see* Shellsort
directed acyclic graph (DAG), 391, 405, 419, 424, 442
discrete mathematics, xv, 49
disjoint set, *see* equivalence class
disk drive, 9, 279, 282–313
  access cost, 286–288, 311
  cylinder, 283, 363
  organization, 282–286
disk processing, *see* file processing
divide and conquer, 250, 252, 255, 320, 487, 492–494
document retrieval, 330, 352
double buffering, 290, 302, 304
dynamic array, *see* array, dynamic
dynamic memory allocation, 106
dynamic programming, 571

efficiency, xiii, 3–5
80/20 rule, 325, 350
element, 25
  homogeneity, 100, 119
  implementation, 119–120
Emacs text editor, 441, 443
encapsulation, 9
enqueue, *see* queue, enqueue
entry-sequenced file, 357
enumeration, *see* traversal
equation, representation, 164
equivalence, 27–28
  class, 27, 205, 210–216, 226–228, 230, 415, 416, 419, 421, 476
  relation, 27, 50
estimating, 25, 47–50, 54, 55, 57–59, 68

exact-match query, *see* search, exact-match query
exponential growth rate, *see* growth rate, exponential
expression tree, 163–169
extent, 285
external sorting, *see* sorting, external

factorial function, 29, 34, 37, 47, 51, 76, 85, 91, 131, 268, 271, 581
  Stirling's approximation, 29, 271
Fibonacci sequence, 34, 51–53, 96
FIFO list, 134
file manager, 282, 285, 288, 431, 432, 438
file processing, 86, 236, 310
file structure, 9, 282, 357, 382
first fit, *see* memory management, first fit
floor function, 30
floppy disk drive, 280, 281, 284
Floyd's algorithm, 532–534, 551, 553
flyweight, *see* design pattern, flyweight
fragmentation, 286, 288, 432, 437–439
  external, 432
  internal, 286, 432
free list, 151
free store, 115
free tree, 391, 411, 416
freelist, 125, 128
full binary tree theorem, 156–157, 169, 198, 224
function, mathematical, 17

garbage collection, 110
general tree, 205–230
  ADT, 206–207, 227
  converting to binary tree, 220, 221, 228
  dynamic implementations, 228
  implementation, 216–221
  left-child/right-sibling, 218, 228
  list of children, 217–218, 228, 391
  parent pointer implementation, 208–216, 456
  terminology, 205–206
  traversal, *see* traversal
Geographic Information System, 8
geometric distribution, 325, 333, 538, 541
gigabyte, 29
graph, xv, 23, 389–421, 425
  adjacency matrix, 427
  adjacency list, 389, 391, 392, 398, 417
  adjacency matrix, 389, 391, 392, 395–397, 416
  ADT, 389, 394–395
  connected component, 391, 421, 502
  edge, 390
  implementation, 389, 394–397
  modeling of problems, 389, 397, 405, 407, 408, 411
  representation, 391–394
  terminology, 389–391
  traversal, *see* traversal, graph
  undirected, 390, 427
  vertex, 390
greatest common divisor, *see* largest common factor
greedy algorithm, 192, 413, 415
growth rate, 57, 60–63, 91
  asymptotic, 67–68
  constant, 60, 69
  exponential, 61, 66, 67
  linear, 61, 65–67, 86

quadratic, 61, 65–67, 86, 87

halting problem, 573–580
Hamiltonian cycle, 582
handle, 439
Harmonic Series, 34, 325, 496
hashing, 7, 10, 31, 64, 318, 330–351, 357, 358, 372, 428, 473, 502
analysis of, 346–350
bucket, 338–339, 356
closed, 336–345, 354
collision resolution, 331, 337–345, 350, 352
deletion, 350–351, 355
double, 345, 354
dynamic, 352
hash function, 331–336, 354
home position, 337
insert, 339
linear probing, 339–342, 345, 349, 350, 354, 356
load factor, 346
open, 336–338
perfect, 331, 352
primary clustering, 342–345
probe function, 340–345
probe sequence, 339, 341–346, 348, 350, 351
pseudo-random probing, 343, 345
quadratic probing, 344, 345, 354
search, 340
table, 330
tombstone, 350
header node, 128, 137
heap, 153, 155, 171, 180–187, 189, 198, 200, 201, 203, 256–259, 277, 409, 416
building, 184–186
for memory management, 431
insert, 184
max-heap, 181
min-heap, 181, 304
partial ordering property, 181
remove, 187
siftdown, 185–187, 277
heapsort, 277
Heapsort, 181, 256–259, 266, 304
heuristic, 572–573
hidden obligations, *see* obligations, hidden
Huffman coding tree, 153, 155, 163, 187–198, 201–204, 229, 448
prefix property, 196

independent set, 582
index, 12, 273, 357–385
file, 300, 358
inverted list, 361, 383
linear, 8, 359–361, 383, 384
tree, 358, 364–381
induction, 156–157, 185, 193–195, 198, 199, 228, 272, 416, 487, 488, 501
induction, proof by, *see* proof, induction
inheritance, 101, 104, 109, 146, 165, 166, 168, 170, *see* object-oriented programming, inheritance
inorder traversal, *see* traversal, inorder
input size, 59, 63
Insertion Sort, 80, 238–239, 241, 243–246, 256, 266, 268–270, 272, 274–276
integer representation, 5, 8–10, 21, 151
inversion, 239, 243
inverted list, *see* index, inverted list
ISAM, 358, 361–364, 383

Java
    generic, 13
Java
    file access, 297–298
    new, 114–122
Java, xv–xvii, 20, 21
    inheritance, xvi
Java
    new, 431

k-d tree, 459, 461–466, 471, 473, 476
K-ary tree, 221–223, 226, 228, 466
key, 140–142
kilobyte, 29
knapsack problem, 570–571, 584
Kruskal's algorithm, xv, 259, 415–416, 419, 420

largest common factor, 52
latency, 285, 287, 288
least frequently used (LFU), 291, 314, 326
least recently used (LRU), 291, 314, 316, 326, 374
LIFO list, 125
linear growth, *see* growth rate, linear
linear index, *see* index, linear
linear search, *see* search, sequential
link, *see* list, link class
linked list, *see* list, linked
LISP, 49, 426, 440, 441, 443
list, 24, 99–153, 189, 358, 423, 425, 502
    ADT, 10, 99–103, 147
    append, 106, 121, 122
    array-based, 9, 99, 103–106, 117–119, 125, 150
    basic operations, 100
    circular, 148
    comparison of space requirements, 149
    current position, 100, 101, 108–110, 113, 118
    doubly linked, 120–125, 148, 151, 163
        space, 124–125
    element, 100, 119–120
    freelist, 114–117, 431–442
    head, 100, 106, 109
    implementations compared, 117–119
    initialization, 100, 104, 105
    insert, 100, 106–110, 113, 118, 121–123, 153
    link class, 106–107
    linked, 9, 99, 103, 106–113, 117–119, 361, 423, 447, 536, 541
    node, 106–109, 121
    notation, 100
    ordered by frequency, 324–329, 482
    orthogonal, 428
    remove, 100, 106, 110, 113, 114, 118, 121, 123, 124
    search, 153, 318–329
    self-organizing, xv, 65, 326–329, 350, 351, 353, 355, 456, 498–499
    singly linked, 107, 120
    sorted, 5, 100, 146
    space requirements, 117–118, 148, 149
    tail, 100
    terminology, 100
    unsorted, 100

locality of reference, 285, 289, 349, 365, 372
logarithm, 31–32, 51
  log*, 216, 226
logical representation, *see* data structure, physical vs. logical form
lookup table, 85
lower bound, 57, 70–72, 348
  sorting, 267–271, 556

map, 389, 407
matching, 572
matrix, 427–430
  multiplication, 558, 559
  sparse, xv, 9, 423, 427–430, 444
  triangular, 427
megabyte, 29
member, *see* object-oriented programming, member
member function, *see* object-oriented programming, member function
memory management, 12, 423, 430–443, 445
  ADT, 430, 446
  best fit, 435, 573
  buddy method, 433, 436–437, 446
  failure policy, 432, 438–443
  first fit, 435, 572
  garbage collection, 439–443
  memory allocation, 430
  memory pool, 430
  sequential fit, 433–436, 445
  worst fit, 436
Mergesort, 131, 246–249, 262, 266, 275, 487, 493, 494
  external, 301–304
  multiway merging, 307–310, 314, 316
metaphor, 11, 20
Microsoft Windows, 285, 311
millisecond, 29
minimum-cost spanning tree, 235, 259, 389, 411–416, 419, 420, 553
modulus function, 28, 30
move-to-front, 326–329, 353, 498–499
multilist, 26, 423–426, 444
multiway merging, *see* Mergesort, multiway merging

nested parentheses, 23, 150
networks, 389, 408
`new`, *see* **C**++, **`new`**
$\mathcal{NP}$-complete, *see* problem, $\mathcal{NP}$-complete
null pointer, 107

O notation, 68–73, 92
object-oriented programming, xvi, 9, 12–16, 21
  class, 9
  class hierarchy, 15–16, 163–169, 468–471
  members and objects, 8, 9
obligations, hidden, 160, 296
octree, 471
Ω notation, 70–73, 92
one-way list, 107
operating system, 19, 180, 282, 285, 287, 289–291, 301, 304, 431, 438, 439, 443
operator overloading, *see* **C**++, operator overloading
overhead, 84, 117–118
  binary tree, 199
  matrix, 429

stack, 129

pairing, 555–557
palindrome, 149
partial order, 28, 50, 181
  poset, 28
partition, 582
path compression, 215–216, 227, 228, 504
permutation, 30, 51, 52, 85, 86, 259, 268–271, 346
physical representation, *see* data structure, physical vs. logical form
Pigeonhole Principle, 53, 137
point quadtree, 471, 476
pop, *see* stack, pop
postorder traversal, *see* traversal, postorder
powerset, *see* set, powerset
PR quadtree, 14, 163, 222, 459, 466–468, 471, 473, 474, 476
preorder traversal, *see* traversal, preorder
prerequisite problem, 389
Prim's algorithm, 412–415, 419, 420
primary index, 358
primary key, 358
priority queue, 153, 171, 189, 203, 409, 413
probabilistic data structure, 529, 536–541
problem, 6, 17, 19, 554
  analysis of, 57, 79–80, 236, 267–271
  hard, 559–573
  impossible, 573–580
  $\mathcal{NP}$-complete, 560–573, 581
problem solving, 20
program, 3, 19
  running time, 58–59
programming style, 20
proof
  contradiction, 40–41, 53
  direct, 40
  induction, 34, 40–46, 49, 53
proof by contradiction, 413, 556, 578, 579
pseudo-polynomial time algorithm, 570
pseudocode, xvii, 19
push, *see* stack, push

quadratic growth, *see* growth rate, quadratic
queue, 99, 107, 133–140, 149, 401, 404–406
  array-based, 134–137
  circular, 135–137, 149
  dequeue, 134, 140
  empty vs. full, 136–137
  enqueue, 133, 135
  implementations compared, 140
  linked, 137, 139, 140
  priority, *see* priority queue
  terminology, 133
Quicksort, 131, 239, 249–257, 266, 273, 274, 276, 299, 301, 304, 352, 481
  analysis, 495–496

Radix Sort, 260–266, 268, 276
RAM, 280, 281
`Random`, 30
range query, 358, *see* search, range query
real-time applications, 64, 65
recurrence relation, 34–36, 53, 254, 481, 487–496, 499, 501

divide and conquer, 492–494
estimating, 487–490
expanding, 491, 492, 501
solution, 36
recursion, xiv, 34–39, 41, 42, 51, 52, 76, 130, 159–161, 174, 175, 199, 202, 247, 249, 273, 275, 276, 442
implemented by stack, 129–133, 256
replaced by iteration, 51, 131
reduction, 268, 554–559, 579, 581, 583
relation, 27–29, 50
replacement selection, 181, 304–307, 309, 314, 316, 481
resource constraints, 5, 6, 17, 57, 58
run (in sorting), 301
run file, 301, 302
running-time equation, 60

satisfiability, 565–569, 571
searc
sequential, 22
search, 23, 87, 317–357
binary, 31, 77–79, 94–96, 272, 320, 352, 359, 360, 374, 493, 502
defined, 317
exact-match query, 7–8, 10, 317, 318, 357
in a dictionary, 320
interpolation, 320–323, 352
jump, 319–320
methods, 317
multi-dimensional, 459
range query, 8, 10, 317, 330, 357, 364
sequential, 59–60, 63–64, 69, 77–79, 94–96, 318–319, 328, 352, 496
sets, 329–330
successful, 317
unsuccessful, 317, 346
search trees, 64, 180, 358, 363, 365, 372, 452, 456, 459
secondary index, 358
secondary key, 358
secondary storage, 279–288, 311–313
sector, 283, 286, 288, 299
seek, 285, 286
Selection Sort, 241–243, 256, 266, 272
self-organizing lists, *see* list, self-organizing
sequence, 27, 30, 51, 100, 318, 329, 359, 554, 555
sequential search, *see* search, sequential
sequential tree implementations, 223–226, 228, 229
serialization, 223
set, 25–29, 51, 330
powerset, 26, 29
search, 318, 329–330
subset, superset, 26
terminology, 25–26
union, intersection, difference, 26, 329, 353
Shellsort, 239, 244–246, 266, 274
shortest paths, 389, 407–411, 419
simulation, 89–90
skip list, xv
Skip List, 529, 536–541, 551, 552
slide rule, 32, 547
software engineering, xiii, 4, 20, 554
sorting, 18, 22–24, 59, 64, 65, 80, 83, 87, 235–277, 319, 328, 554–557
adaptive, 271

comparing algorithms, 237, 265–267, 309, 310
exchange sorting, 243
external, 171, 236, 257, 279, 298–311, 314–316
lower bound, 236, 267–271
small data sets, 237, 255, 271, 275
stable algorithms, 236, 272, 273
terminology, 236–237
spatial data structure, 447, 459–473
splay tree, 180, 198, 365, 447, 453, 455–459, 473–476, 482, 536
stable sorting alorithms, *see* sorting, stable algorithms
stack, 99, 107, 125–133, 149, 199, 202, 256, 272, 273, 276, 400–402, 497–498
array-based, 125–126
constructor, 125
implementations compared, 129
insert, 125
linked, 128, 129
pop, 125, 126, 128, 151
push, 125, 126, 128, 151
remove, 125
terminology, 125
top, 125–126, 128
two in one array, 129, 149
variable-size elements, 151
Strassen's algorithm, 543, 552
strategy, *see* design pattern, strategy
subclass, *see* object-oriented programming, class hierarchy
subset, *see* set, subset
suffix tree, 475
summation, 33–34, 42–44, 53, 54, 75, 76, 95, 180, 186, 254, 324, 325, 427, 481–486, 491, 493–497, 500
guess and test, 500
list of solutions, 33, 34
notation, 33
shifting method, 483–486, 495, 500
`swap`, 30

table, 318
tape drive, 282, 283, 298
template, *see* **C**++, template
text compression, 153, 187–198, 282, 328–329, 351, 355
Θ notation, 71–73, 94
topological sort, 389, 405–406, 418
total order, 29, 51, 181
Towers of Hanoi, 37–39, 131, 553, 559
tradeoff, xiv, 3, 13, 79, 286, 298
disk-based space/time principle, 86, 282, 349
space/time principle, 85–86, 101, 124, 187, 282, 349
transportation network, 389, 407
transpose, 327, 328, 353
traveling salesman, 561–563, 570–572, 582, 584
traversal
binary tree, 131, 153, 158–162, 167, 172, 180, 199, 397
enumeration, 158, 172, 223
general tree, 207–208, 227
graph, 389, 397–406
tree
height balanced, 371, 372, 374, 536
terminology, 153
trie, 163, 265, 447–452, 473, 475
alphabet, 449
binary, 448

PATRICIA, 451–452, 473
tuple, 27
Turing machine, 566
two-coloring, 45
2-3 tree, 180, 358, 366–371, 374, 378, 383, 384, 452, 502, 536
type, 8

uncountability, 574–577
UNION/FIND, xv, 209, 416, 421, 482, 504
units of measure, 29, 49
UNIX, 250, 285, 311, 441
upper bound, 57, 68–72

variable-length record, 151, 359, 384, 423, 425, 430
sorting, 237
vector, 27
`Vector`, 118
vertex cover, 568, 571, 572, 582, 584
virtual function, 170
virtual memory, 291–293, 301, 314
visitor, *see* design pattern, visitor

weighted union rule, 214–215, 227–228, 504
worst fit, *see* memory management, worst fit
worst-case analysis, 63–65, 68

Zipf distribution, 325, 333, 353
Ziv-Lempel coding, 329, 352