

Index

- Abstract data type (ADT), 127-195
 - abstract classes, 163
 - classes, 129-136
 - collections of items, 137-139
 - creating, 157-164
 - defined, 128
 - duplicate items, 173-176
 - equivalence-relations, 159-162
 - FIFO queues, 165-171
 - first-class, 177-186
 - generic operations, 273
 - index items, 177
 - insert/remove* operations, 138-139
 - modular programming, 135
 - polynomial, 188-192
 - priority queues, 375-376
 - pushdown stack, 138-156
 - stubs, 135
 - symbol table, 497-506
- ADT interfaces
 - array (*myArray*), 274
 - complex number (*Complex*), 181
 - existence table (*ET*), 663
 - full priority queue (*PQfull*), 397
 - indirect priority queue (*PQi*), 403
 - item (*myItem*), 273, 498
 - key (*myKey*), 498
 - polynomial (*Poly*), 189
 - point (*Point*), 134
 - priority queue (*PQ*), 375
 - queue of int (*intQueue*), 166
 - stack of int (*intStack*), 140
 - symbol table (*ST*), 503
 - text index (*TI*), 525
 - union-find (*UF*), 159
- Abstract in-place merging, 351-353
- Abstract operation, 10
- Access control state, 131
- Actual data, 31
- Adapter class, 155-157
- Adaptive sort, 268
- Address, 84-85
- Adjacency list, 120-123
 - depth-first search, 251-256
- Adjacency matrix, 120-122
- Ajtai, M., 464
- Algorithm, 4-6, 27-64
 - abstract operations, 10, 31, 34-35
 - analysis of, 6
 - average-/worst-case performance, 35, 60-62
 - big-Oh notation, 44-47
 - binary search, 56-59
 - computational complexity, 62-64
 - efficiency, 6, 30, 32
 - empirical analysis, 30-32, 58
 - exponential-time, 219
 - implementation, 28-30
 - logarithm function, 40-43
 - mathematical analysis, 33-36, 58
 - primary parameter, 36
 - probabilistic, 331
 - recurrences, 49-52, 57
 - recursive, 198
 - running time, 34-40
 - search, 53-56, 498
 - steps in, 22-23
 - See also* Randomized algorithm
- Amortization approach, 557, 627
- Arithmetic operator, 177-179, 188, 191
- Array, 12, 83
 - binary search, 57
 - dynamic allocation, 87
 - and linked lists, 92, 94-95
 - merging, 349-350
 - multidimensional, 117-118
 - references, 86-87, 89
 - sorting, 265-267, 273-276
 - and strings, 119
 - two-dimensional, 117-118, 120-124
 - vectors, 87
 - visualizations, 295
 - See also* Index, array
- Array representation
 - binary tree, 381
 - FIFO queue, 168-169
 - linked lists, 110
 - polynomial ADT, 191-192
 - priority queue, 377-378, 403, 406
 - pushdown stack, 148-150
 - random queue, 170
 - symbol table, 508, 511-512, 521
- Asymptotic expression, 45-46
- Average deviation, 80-81
- Average-case performance, 35, 60-61
- AVL tree, 583
- B tree, 584, 692-704
 - external/internal pages, 695
 - 4-5-6-7-8 tree, 693-704
 - Markov chain, 701
 - remove*, 701-703
 - search/insert*, 697-701
 - select/sort*, 701
- Balanced tree, 238, 555-598
 - B tree, 584
 - bottom-up, 576, 584-585
 - height-balanced, 583
 - indexed sequential access, 690-692
 - performance, 575-576, 581-582, 595-598
 - randomized, 559-564
 - red-black, 577-585
 - skip lists, 587-594
 - splay, 566-571

- 2-3, 584
- 2-3-4, 572-576, 584, 593-594
- Batcher's odd-even mergesort, 455-459
 - networks, 461-466
 - shuffling, 464-466
- Bayer, R., 692-693
- Bentley, J. L., 336
- Bernoulli trial, 87-88
- Big-Oh notation. *See* O-notation
- Bin, 428
- Binary logarithm, 40-42
- Binary representation, 50-51, 602, 636
- Binary search, 56-59, 682
 - as divide-and-conquer algorithm, 216-217
 - index implementations, 527-530
 - interpolation search, 522-523
 - symbol tables, 519-523, 527-530
 - text-string index, 682
- Binary search tree (BST), 249, 531-536
 - count field, 547, 551-552, 596
 - defined, 531
 - digital, 637-640
 - duplicate keys, 536
 - insertion, 542-546, 560, 573-575, 580-582
 - join, 550-552, 563-564
 - memory, 552-553
 - nonrecursive, 535-536
 - partitioning, 547-548
 - path length, 538
 - perfectly balanced, 555-557
 - performance, 537-541
 - and quicksort, 536
 - randomized, 559-564, 595-598
 - red-black, 577-585
 - remove*, 548-551, 563-564
 - rotation, 543-544, 566-569
 - search/insert*, 531, 533-539, 545, 561-562
 - select*, 547-548
 - sort*, 534-535
 - splay, 566-571, 595-598
 - text-string index, 681-682
 - and TSTs, 676
 - 2-3, 584
 - 2-3-4, 572-576, 584, 593-594
 - visit, 535
 - worst-case, 539-540, 555
 - See also* Balanced tree; Symbol table
- Binary tree, 230-233
 - binary search comparisons, 522-523
 - complete, 239, 381-382
 - and hashing, 633
 - heap-ordered, 381, 393, 406-408
 - join operation, 407-408
 - links in, 230-231
 - mathematical properties, 236-239
 - operations, 232
 - path length, 237
 - printing, 247
 - recursive algorithms, 246-251
 - tournament construction, 248-249
 - See also* Binary search tree; Trie
- Binomial distribution, 43
- Binomial queue, 406-415
 - defined, 408
 - insert*, 409-410
 - join, 411-413
 - performance, 413-415
 - remove the maximum*, 413-415
- Binomial tree, 407
- Bin-span heuristic, 433
- Birthday problem, 612
- Bit, 71, 420
 - binary quicksort, 424-426
 - bottom-up mergesort, 362
- bit method, 635-636, 661
- Bitonic sequence, 352
- Block sorting, 488-489
- Boolean data type, 71
- Bounding a function, 47, 48
 - upper/lower bound, 62-64
- Breadth-first search, 253, 256-257
- Brent, R., 631
- Bubble sort, 288-289, 290-294
- Bucket, 428
- Bug. *See* Error
- Butterfly network, 465
- Byte, 417, 419-420
- Bytecode, 34
- Cache, 688, 719
- Cast, type, 72, 155
- Ceiling function, 43
- Character data type, 71-72
- Child node, 229
- Circular list, 92, 102-104
 - head and tail conventions, 102
 - Josephus election, 95-96
 - memory allocation, 109
 - mergesort, 367-368
- Class, 70, 74-83, 129-136
 - abstract, 163
 - adapter, 155-157
 - concrete data types, 134-135
 - extended, 82
 - instantiation, 75
 - members, 76-77
 - public/private members, 129
 - sort methods, 272
- Class paths, 161-162
- Client, 81, 108-110
 - implementations, 128-129, 162-164
 - interface, 141, 158
- Cloning, 182-185
- Closest-point computation, 90
- Cluster, 617-620, 623
- Coin-flipping simulation, 87-88
- Combine-and-conquer approach, 215
 - mergesort, 361-363
- Comparator, 460-461, 488
- `compareTo` method, 530
- Complex (complex number) ADT
 - interface, 181
- Complex number, 178-182
- Complex roots of unity, 179-180
- Computational complexity, 62-64
- Connectivity problem, 7-11, 158-161

- connected components, 10
- quick-find algorithm, 11-13, 15
- quick-union algorithm, 13-17
- union-find operations, 10-11
- Conquer-and-divide algorithm, 370-371
- Constant running time, 37
- Constructor, 77-78, 163
 - copy, 138
 - linked lists, 93
- Cost. *See* Performance; Empirical analysis; Running time
- Count field, 547, 551-552, 596
- Coupon collector problem, 612
- Cubic running time, 38
- Cyclic structure, 92

- Data, 31
 - page, 687-689
 - perverse, 35
 - random, 35
- Data structure, 4, 69-70, 83
 - compound, 89, 116-125
 - cyclic, 92
 - self-referent, 91
 - See also* Array; Linked list; String
- Data type, 70-78
 - character, 71-72
 - concrete, 134-135
 - defined, 72
 - first-class, 177
 - floating-point, 71-72
 - integer, 71-72
 - interface, 81
 - item, 495
 - point, 129-132, 134
 - See also* Abstract data type
- Database, 686-687
 - index, 688-692
- de la Briandais, P., 641
- delete operator, 108
- Depth-first search, 251-257
- Deque ADT, 170
- Dictionary, 496, 631-633
- digit method, 635-636, 661
- Digital search tree (DST), 637-640

- Dijkstra, E., 336
- Directory, 691
 - pages, 9, 687-689, 695, 707-712
 - size, 716-717
 - split, 710-711
- Disk device, 686, 691
- Divide-and-conquer algorithm, 206-217
 - Batcher's merge, 457
 - binary search, 216-217
 - bottom-up implementation, 214-215
 - drawing a ruler, 211-215
 - finding the maximum, 206-209
 - fractals, 215-216
 - internal stack, 207-208
 - mergesort, 216-217, 347, 353
 - quicksort, 216-217
 - running time, 208
 - towers of Hanoi, 209-214
- Divide-and-conquer tree, 208-209, 354-355
- Doubling, 46
- Doubly linked list, 105, 398-399, 401
- Driver
 - array sort, 266
 - complex numbers, 180
 - sortable arrays, 274-275
- Dummy node, 92, 99-103, 106
- Duplicate item, 173-176
- Duplicate key, 336-337, 502-504
 - binary search tree, 536
 - extendible hashing, 715
 - hashing, 613
 - red-black trees, 583
 - splay BST, 570
 - symbol tables, 521
- Dutch National Flag problem, 336
- Dynamic allocation, 87
 - linked lists, 107-110
 - two-dimensional arrays, 118
- Dynamic programming, 219-225
 - bottom-up, 220, 222-223
 - Fibonacci numbers, 219-221
 - knapsack problem, 220-225
 - running time, 222
 - top-down, 220, 224

- Edge, 120
- Empirical analysis, 30-32
 - balanced trees, 596-597
 - elementary sorting, 294
 - hashing, 632
 - heapsort, 395
 - mergesort, 364
 - quicksort, 333, 340
 - radix sorting, 450-451
 - search algorithms, 59
 - shellsort, 306
 - symbol tables, 541
 - union-find algorithms, 20
 - See also* Performance; Running time
- equals method, 501, 503
- equals operator, 682
- Eratosthenes, sieve of, 85-86, 95
- Error
 - inheritance, 155
 - performance bug, 114
 - private methods, 131-133
 - pushdown stack, 153
 - quicksort, 319
 - references, 98
- ET (existence table) ADT interface, 663
- Euclid's algorithm, 201
- Euler's constant, 42
- Exception dictionary, 631-633
- Existence table, 507
- Existence trie, 662-669
- Exponential running time, 38, 219
- Extended class, 82
- Extendible hashing, 706-717
 - directory page indices, 707-710
 - hash table, 710-713
 - search/insert*, 709-716
- External searching, 685-719
 - B trees, 692-704
 - cache, 688, 719
 - database, 686-690
 - directory pages, 687-689, 695, 707-712

- disk device, 686, 691
- extendible hashing, 706-717
- indexed sequential access, 690-692
- probes, 687
- search and insertion, 691-692
- storage devices, 689-690
- virtual memory, 686, 688

- Factorial function, 42-43, 199
- Fagin, R., 706
- Fibonacci number, 42-43, 219-221
 - generalized, 483
- FIFO queue, 138, 165-171, 183-184
 - array implementation, 168-169
 - duplicate items, 174-175
 - get/put, 166, 168
 - linked list implementation, 167
 - performance, 168-169
- File
 - disk pages, 691
 - h*-sorted, 300, 303
 - reading/writing, 475-476
 - sorting, 265
- First-class ADT, 177-186
- Floating-point data type, 71-72
- Floating-point number, 41
 - hash function, 599-602
- Floor function, 43
- Floyd, R., 392-393
- Forest, 228, 232
- Fractal, 215-216
- Fredkin, E., 641
- Free list, 108-110
- Free tree, 228, 234-235
- Function
 - bounding, 47, 48, 62-64
 - ceiling/floor, 43
 - hash, 599-608
 - Josephus, 95
 - logarithm, 40-43
 - recursive, 223
- Function call, 102
- Functional programming, 102

- Garbage collection, 110, 186, 191

- Generalized queue, 138, 169-170, 373
- Generic pushdown stack, 154-156
- Golden ratio, 42
- Grandparent node, 229
- Graph, 10, 120-125
 - defined, 234
 - dense/sparse, 121
 - traversal, 251-257
 - undirected, 120
 - union-find solutions, 254
- See also* Tree
- Guibas, L., 622, 623

- Halving, 19, 208
- Handle, 397-398, 405
 - binary search trees, 550
 - symbol tables, 502-503
- See also* Reference
- Harmonic number, 42
- Hashing, 509, 599-634
 - and binary trees, 633
 - clusters, 617-620, 623
 - collision-resolution, 599-600
 - double, 620-625, 630
 - dynamic hash tables, 625-629
 - errors, 607-608
 - extendible, 706-717
 - hash functions, 599-608
 - linear probing, 615-619, 623-624, 626, 630
 - load factor, 615
 - memory, 613
 - modular, 602-603
 - multiplicative, 600
 - open-addressing, 615, 624
 - ordered, 631
 - patricia tries, 658-659
 - performance, 607-608, 629-631
 - prime table size, 602-603
 - random, 622-623
 - rehashing, 618-619
 - remove*, 624, 628
 - search/insert*, 611-614, 627-629, 633
 - separate chaining, 610-614
 - select/sort*, 614, 618
- string keys, 604-605
- time-space tradeoff, 600
- universal, 605-607
- Head node, 99-103
- Heap data structure, 381-389
 - bottom-up heapify, 383-384
 - defined, 381
 - index heap, 402-404
 - insert*, 385-386
 - power-of-2 heap, 407-409
 - sortdown process, 389, 390
 - sorting, 388-389
 - top-down heapify, 384-385
- See also* Binomial queue; Priority queue
- Heapsort, 347, 389-395
 - bottom-up construction, 390-391
 - running time, 391-393
- Hoare, C. A. R., 315, 341
- Hopcroft, J., 584
- Horner's algorithm, 191
- h*-sorted file, 300, 303

- Imaginary number, 179
- Implementation, 81, 128-129
 - interface, 141
 - separate, 162-164
- In class, 724
- Increment sequence, 300-302, 304-306
- Index, array, 85, 87, 97
 - database, 688-692
 - directory, 691, 707-710
 - items, 175-176, 402-405
 - key as, 312-314
 - key-indexed search, 507-510
 - sorting, 470
 - string, 679-683
 - symbol tables, 524-530
- See also* Key-indexed counting
- Index heap, 402-404
- Indexed sequential access, 690-692
- Indirect sorting, 269-270
- Inductive proof, 200-203
- Infix notation, 143, 146-147

- Inheritance, 82, 154-155, 162-164
- Inner loop, 268
 - insertion sort, 285-286
 - quicksort, 318
- Inorder traversal, 240-241
- In-place merging, 350-353
- In-place rearrangement, 472
- Input
 - duplicate keys, 336-337
 - empirical analysis, 31
 - external sorting, 475
 - list, 100, 102
 - mergesort, 356
 - running time, 60-62
 - selection sort, 290
- Insertion sort, 268, 285-287
 - performance, 290, 292-294
 - and quicksort, 329
 - See also* Shellsort
- Instantiation, 75, 93
- intQueue (queue of int) ADT
 - interface, 166
- intStack (stack of int) ADT interface, 140
- Integer
 - and ADTs, 136
 - binary representation, 50-51
 - conversion, 41
 - hash function, 601-602
 - sorting, 276-277
- Integer data type, 71-72
- Interface (ADT), 81, 141
 - freezing, 158
 - item, 271, 500
 - opaque, 128-129
 - See also* ADT interfaces
- interface (Java), 163
- Interpolation search, 522-523
- Inversion, 292
- Item
 - collections of, 137-139
 - duplicate, 173-176
 - interface, 271, 500
 - search keys, 495
 - symbol tables, 496, 498-502, 511, 527
- Java
 - animations, 295-297
 - arithmetic operators, 177-179
 - byte access, 421
 - class paths, 161-162
 - classes, 70, 131-133
 - compound structures, 118-119
 - constructors, 77-78
 - memory allocation, 108-110
 - method definition, 73
 - object references, 76, 279-282
 - sorting items, 271-272
 - strings, 111-112, 436
 - VM implementation, 34, 363
- Josephus problem, 95-96, 102-104
- Karp, R., 561
- Key, 265
 - bitonic sequence, 352
 - database, 688-689
 - hash function, 600-608
 - heap-ordered tree, 381
 - indexing, 312-314
 - inversion, 292
 - multiple, 269
 - octal, 690-691
 - partitioning, 319-320
 - radix search, 635-637
 - radix sorts, 417-422
 - random, 538-539, 559-564
 - search, 495
 - sentinel, 285-287, 318, 351-352, 511-513
 - string, 339, 604-605, 670-671, 679-683
 - symbol tables, 496, 498-502, 511, 527
 - trie, 642-648, 652, 661-662
 - vector as, 341, 439-440
 - See also* Duplicate key
- Key-indexed counting, 312-314
 - radix sorting, 422, 429-431, 442, 444-449
- Key-indexed search, 507-510
 - See also* Hashing
- Knapsack problem, 220-225
- Knuth, D. E., 301-302, 446, 484, 618
- Koch star, 215, 218
- Komlos, J., 464
- Language, programming, 144-146
- Leading term, 38
- Leaf, 229, 642
- Least-significant-digit (LSD) radix sorting, 419, 441-443, 448-450
- less method, 271-272, 501, 503, 527
- Level-order traversal, 243-245, 360, 361
- Lexicographic order, 112
- Library method, 113-114
- Library program, 5
- LIFO queue, 140
- Linear probing, 615-619, 623-624, 626, 630
- Linear running time, 37
- Linked list, 91-96
 - adjacency lists, 120-123
 - and arrays, 92, 94-95, 110
 - constructors, 93
 - defined, 91, 98
 - doubly, 105, 398-399, 401
 - dynamic allocation, 107-110
 - find operation, 94
 - hashing, 610
 - head and tail conventions, 103
 - insert/remove*, 94
 - random queues, 170
 - recursive methods, 203-205
 - sequential, 92
 - skip list, 587-594
 - sorting, 99-101, 308-311
 - two-dimensional array, 120-125
- Linked list representation
 - binary search, 520
 - clonable queues, 186
 - FIFO queues, 167, 183-184
 - mergesort, 366-369
 - MSD radix sorting, 431

- priority queue, 398-399, 401, 407-408
- pushdown stack, 150-152
- symbol table, 513-514, 516
- trees, 381-382
- List, 83
 - adjacency, 120-123, 251-256
 - free, 108-110
 - multilist, 119
 - skip, 587-594
 - See also* Circular list; Linked list
- List processing, 97-106
 - free list, 108
 - reversal, 99
 - sorting, 99-101
 - traversal, 99
- Logarithm function, 40-43
- Logarithmic running time, 37
- Loop
 - inner, 268, 285-286
 - simple, 199
- Lower bound, 62-64

- McCreight, E. M., 692-693
- Machine word, 419-420
 - key as, 635
- McIlroy, M. D., 336
- Markov chain, 701
- M-ary tree, 229-230
 - defined, 232
- Matrix, 117-118
 - adjacency, 120-122
 - sparse, 119
- Median-of-three partitioning, 331-335
- Member, class, 129
- Memorization, 220
- Memory
 - associative, 497
 - binary search tree, 552-553
 - cache, 688
 - dynamic allocation, 87
 - FIFO queues, 168
 - generic sorting, 280-281
 - hashing, 613
 - initialization, 93
 - in-place sorting, 468-470
 - linked lists, 93-94, 107-110
 - mergesort, 355-356
 - radix sorting, 429
 - sorting, 269, 474-479
 - sort-merge, 484-485
 - stacks, 151-152
 - strings, 115-116
 - two-dimensional arrays, 118
 - virtual, 686, 688
- Mergesort, 353-371
 - Batcher's odd-even mergesort, 455-459
 - bottom-up, 359-369
 - divide-and-conquer tree, 354-355
 - and heapsort, 392-393
 - improvements to, 357-358
 - linked list representation, 366-369
 - memory, 355-356
 - natural, 368
 - performance, 363-366
 - and quicksort, 357, 363-365, 370-371
 - running time, 347-348
 - sequential access, 348
 - stability, 356
 - top-down, 353-356, 363, 366-368, 457
- Merging, 347-353
 - arrays, 349-350
 - comparator, 488
 - in-place, 350-353
 - multiway, 476-482
 - polyphase, 482-484
 - and selection, 347
 - sentinels, 351-352
 - split-interleave, 465
 - stable, 352-353
 - two-way, 348-350
 - See also* Sort-merge method
- Mersenne prime, 603
- Method, 70
 - accessor, 131-133
 - definition, 73
 - library, 113-114
 - recursive, 198, 207
- Modular programming, 135
- Morrison, D., 651
- Most-significant-digit (MSD) radix sorting, 419, 427-434, 446-447
- Multidimensional array, 117-118
- Multikey quicksort, 440
- Multilist, 119
- Multiway trie, 432, 664-665
- myArray (array) ADT interface, 274
- myItem (item) ADT interface, 273, 498
- myKey (key) ADT interface, 498

- $N!$, 42-43, 199
- $N^{3/2}$ algorithm, 40
- Name-equivalence problem, 8
- Natural logarithm, 40-42, 46
- Network, sorting, 460-466
 - AKS, 464-465
 - Batcher's merge, 462, 464-465
 - Bose-Nelson problem, 464-465
 - butterfly, 465
 - comparators, 460-461
 - parallel stages, 461
 - new operator, 75, 87, 110
- Nievergelt, J., 706
- $N \log^2 N$ algorithm, 40
- $N \log N$ running time, 37-38
- Node
 - B tree, 695-700
 - dummy, 92, 99-102, 106
 - fixed-size, 107
 - head, 99-102
 - heap-ordered tree, 381
 - leaf, 229, 642
 - level of, 237
 - linked list, 91-94
 - marked, 549
 - parent/child, 229
 - priority of, 405
 - red-black trees, 577-581
 - sorting, 309
 - tail, 102, 103
 - terminal/nonterminal, 229
 - of a tree, 228-229

- tries, 654-656
- 2-3-4, 572-576
- Nonadaptive sort, 268, 285, 455-458
- Normal approximation, 88-89
- Null link, 92, 409-411, 531
- Object, 70
 - array as, 86-87
 - cloning, 182-185
 - complex, 180-181
 - handles, 397-398
 - immutable, 131
 - linked lists, 92
 - See also* Reference
- Object-oriented programming (OOP), 78, 82-83
- Occupancy problem, 612
- Octal key, 690-691
- O-notation (O), 44-47
 - asymptotic expression, 45-46
 - bounding a function, 47, 48
 - leading terms, 45-47
 - natural logarithm, 46
 - upper/lower bound, 62-64
- Opaque interface, 128-129
- Open-addressing hashing, 615, 624
- Optimization approach, 557-558
- Order statistics, 341
- Ordered tree, 229, 232-233
- Out class, 724
- Output
 - external sorting, 475
 - list, 100, 102
- Overloading, 78
- Page, 687-689
 - external/internal, 695
 - indices, 707-708
 - split, 710-712
- Parameter, 73
- Parent node, 229
- Parse tree, 250-251
- Partitioning, 316-320, 327
 - binary quicksort, 425-427
 - binary search tree, 547-548
- LSD radix sorting, 442-443
- median-of-three, 331-335
- MSD radix sort, 429
- three-way, 336-341
- three-way radix quicksort, 435-440
- Path
 - cycle, 234
 - length, 237, 538
 - simple, 234
 - in tree, 228
- Path compression, 18-19, 21, 63
- Patricia trie, 650-659, 681-682
- Perfect shuffle, 456-457, 465-466
- Performance
 - average-/worst-case, 35, 60-62
 - balanced tree, 575-576, 581-582, 595-598
 - binary search tree, 537-541
 - binomial queue, 413-415
 - bug, 114
 - FIFO queue, 168-169
 - hashing, 607-608, 629-631
 - insertion sort, 290, 292-294
 - mergesort, 363-366
 - priority queue, 376, 406
 - pushdown stack, 151-152
 - quicksort, 321-324
 - radix sort, 431, 444-452
 - sorting, 289-294, 488-490
 - symbol tables, 505-506
 - tries, 657-660, 665-666
 - TSTs, 676-677
 - See also* Empirical analysis; Running time
- Permutation, 470, 472
- Perverse data, 31, 35
- Pippenger, N., 706
- Point (point) ADT interface, 134
- Point data type, 77, 129-132, 134
- Pointer, 76
 - duplicate keys, 336-337
 - head/tail, 167
 - partitioning, 320
- Pointer sorting, 279-280
- Poisson approximation, 43
- Polar coordinate, 183
- Polish notation, 143
- Poly (polynomial) ADT interface, 189
- Polynomial ADT, 188-192
- Polyphase merging, 482-484
- Postfix notation, 142-147
- Postorder traversal, 240-241
- PostScript, 144-146, 295-296
- Power-of-2 heap, 407-409
- PQ (priority queue) ADT interface, 375
- PQfull (full priority queue) ADT interface, 397
- PQi (indirect priority queue) ADT interface, 403
- Pratt, V., 305
- Prefix notation, 143, 202
- Preorder traversal, 240-241, 243
- Primary parameter, 36
- Prime, Mersenne, 603
- Prime number, 85
- Primitive data type, 177-178
- Priority queue, 170-171, 373-379, 396-405
 - applications of, 373
 - array representation, 377-378, 403, 406
 - change priority, 387
 - defined, 373
 - find/remove the maximum*, 377-380, 386-387
 - handles, 397-398
 - heap-based, 385-389
 - index items, 402-405
 - insert*, 377
 - linked list, 398-399, 401, 407-408
 - multiway merging, 478, 480-481
 - operations, 374-376
 - performance, 376, 406
 - sorting, 388-389
 - worst-case costs, 378-379
 - See also* Binomial queue; Heap data structure
- Private method, 131-133
- Probabilistic algorithm, 331

- Probe, 615, 687
- Problem specification, 9
- Public method, 131-133
- Pugh, W., 587
- Pushdown stack, 138-156
 - array implementation, 148-150
 - depth-first search, 254-256
 - duplicate items, 173-176
 - and FIFO queue, 167
 - generic, 154-156
 - infix-to-postfix conversion, 146-147
 - linked list implementation, 150-152
 - performance, 151-152
 - postfix-expression evaluation, 142-144
 - push/pop operations, 140, 150-152
 - quicksort, 325-328
 - tree traversal, 241-244
- Quadratic running time, 38
- Queue
 - binomial, 406-415
 - FIFO, 138, 183-184
 - generalized, 138, 169-170, 373
 - LIFO, 140
 - random, 170, 183, 185
 - simulation, 184
 - tree traversal, 244
 - See also* Priority queue
- Quick-find algorithm, 11-13, 15
- Quicksort, 315-344
 - and binary search trees, 536
 - binary trie, 645
 - duplicate keys, 336-337
 - and heapsort, 392, 393
 - median-of-three partitioning, 331-341
 - and mergesort, 357, 363-365, 370-371
 - multikey, 440
 - nonrecursive, 325-327, 332-333
 - partitioning, 316-320
 - performance, 321-324
 - recurrence formula, 322-324
 - running time, 438-439
 - selection of median, 341-344
 - small subfiles, 328-330
 - stack size, 325-328, 333
 - tail-recursion removal, 326-327
 - three-way partitioning, 336-341
 - See also* Radix sorting
- Quick-union algorithm, 13-17
- Radix search, 635-683
 - digital search trees, 637-640
 - search/insert*, 638-640
 - See also* Radix sorting; Trie
- Radix sorting, 417-452
 - binary quicksort, 423-427
 - bins/buckets, 427
 - bin-span heuristic, 433
 - key-indexed counting, 422, 429-431, 442, 448
 - keys in, 417-422
 - linear, 444
 - LSD, 419, 441-443, 448-450
 - memory, 429
 - MSD, 419, 427-434, 446-447
 - performance, 431, 444-452
 - radix-exchange sort, 423
 - sublinear, 446, 448-452
 - three-way radix quicksort, 435-440, 448
 - See also* Quicksort
- Random data, 31, 35
- Random number, 562
- Random queue, 170, 183, 185
- Randomized algorithm, 62, 556-557
 - BSTs, 559-564, 595-598
 - extendible hashing, 706
 - hashing, 605, 608
- Range-search problem, 521
- Real number
 - conversion, 41
 - and floating-point numbers, 71
- Records, sorting, 277-280
- Recursion, 49-52, 197-226
 - Batcher's merge, 458
 - binary-tree algorithms, 246-251, 533, 544
 - bottom-up implementation, 214-215
 - depth of, 203
 - divide-and-conquer, 206-217
 - drawing a ruler, 211-215
 - dynamic programming, 219-225
 - Euclid's algorithm, 201
 - factorial function, 199
 - Fibonacci numbers, 219-221
 - finding the maximum, 206-209
 - fractals, 215-216
 - graph traversal, 251-257
 - inductive proof, 200-203
 - knapsack problem, 220-225
 - linked lists, 203-205
 - MSD radix sorting, 432-434
 - multiple recursive calls, 201-202
 - prefix expression evaluation, 202
 - quicksort, 317, 326-327
 - recurrence relations, 199, 219
 - recursive descent parser, 202
 - recursive function, 223
 - recursive methods, 198, 207
 - sorting, 370-371
 - tail, 204
 - termination condition, 197
 - three-way radix quicksort, 439
 - towers of Hanoi, 209-214
 - tree traversal, 241
 - See also* Tree
- Red-black tree, 577-585, 595-598
- Reference, 71, 75-76
 - arithmetic operators, 178
 - arrays, 86-87, 89
 - bugs, 98
 - extendible hashing, 709
 - linked lists, 92
 - sorting, 279-282
 - strings, 116, 118-119
 - See also* Handle
- Rehashing, 618-619
- Repainting, 297
- Replacement selection, 480-481
- Return value, 73
- Root
 - insertion, 542-546, 560

- splitting, 574, 695-700
 - of tree, 14
 - of unity, 179-180
- Rooted tree, 228, 233-234
- Rotation, 543-544, 566-569
 - red-black trees, 577-583
- Row-major order, 118
- Running time, 34-40
 - abstract classes, 163
 - animations, 297
 - binary search, 57-59, 537-541
 - constant, 37
 - cubic, 38
 - divide-and-conquer, 208
 - doubling, 46
 - dynamic programming, 222
 - elementary sorts, 264-265, 268, 289-294
 - exponential, 38, 219
 - heapsort, 391-393
 - in-place sort, 471
 - input, 60-62
 - linear, 37
 - mergesort, 347-348
 - $N \log N$, 37-38
 - priority queues, 379
 - quadratic, 38
 - queue simulation, 185-186
 - quicksort, 438-439
 - radix sort, 444-452
 - recurrence relations, 49
 - reference approach, 281-282
 - sequential search, 54-56
 - sorting, 488-490
 - symbol table, 509-510, 520
 - See also* Empirical analysis; O-notation (O); Performance
- Search, 495
 - breadth-first, 253, 256-257
 - depth-first, 251-257
 - hit/miss, 515-516
 - interpolation, 522-523
 - key-indexed, 507-510
 - sequential, 511-517
 - string, 113
- See also* Binary search; External searching; Radix search; Symbol table
- Seconds conversion, 39
- Selection of median, 341-344
- Selection sort, 283-284, 289-291
 - in-place, 470
 - linked lists, 309-310
 - running time, 294
- Self-referent structure, 91
- Sentinel key, 286-287, 318
 - merging, 351-352
 - sequential search, 511-513
- Sequential access, 687
 - indexed, 690-692
 - mergesort, 348
- Sequential search, 53-56, 475
 - symbol table, 511-517
- Shell, D., 302
- Shellsort, 300-307
 - h*-sorting, 300
 - increment sequences, 300-302, 304-306
 - properties, 303-305
- Shuffling, 456-457, 464-466
- Sibling node, 229
- Sieve of Eratosthenes, 85-86, 95
- Signature, method, 73
- Skip list, 587-594, 595-597
 - defined, 588
 - insert*, 590-591
 - remove*, 593
 - search*, 588-589
 - 2-3-4 tree, 593-594
- Sleator, D., 569
- Slow union, 12
- Sortable array, 274-276
- Sortdown process, 389, 390
- Sorting, 263-314
 - adaptive/nonadaptive, 268, 285, 455-458
 - animations, 295-299
 - array sorts, 265-267
 - binary search tree, 534-535
 - block, 488-489
 - bubble, 288-294
 - external, 265, 474-479
 - generic implementations, 270-282
 - heap, 388-389
 - in place, 468-473
 - indirect, 269-270
 - insertion, 268, 285-287, 290-294
 - internal, 265
 - key-indexed counting, 312-314
 - linked lists, 99-101, 308-311
 - memory, 269, 474-479
 - networks, 460-466
 - perfect shuffle/unshuffle, 456-457
 - performance, 289-294, 488-490
 - pointer, 279-280
 - priority queue, 388-389
 - read/write, 474
 - recursive, 370-371
 - running time, 264-265, 268
 - shellsort, 300-307
 - stable, 269-270
 - strings, 118, 119, 280
 - vectors, 439-440
 - See also* Heapsort; Mergesort; Quicksort; Selection sort
- Sort-merge method, 476-490
 - multiway merging, 476-479, 481-482
 - parallel, 484-485
 - polyphase merging, 482-484
 - replacement selection, 480-481
 - virtual memory, 484-485
- Spanning tree, 10
- Sparse matrix, 119
- Specification
 - problem, 9
 - program, 141-142
- Splay BST, 566-571, 595-598
- ST (symbol table) ADT interface, 503
- Stable sort, 269-270
- Stack. *See* Pushdown stack
- Standard deviation, 80-81
- static designation, 78
- Stirling's formula, 42
- Straight-line program, 455-456

- String, 111-116
 - array of, 119
 - compare operation, 112
 - hash functions, 604-605
 - index, 525-530
 - key, 339, 604-605, 635, 670-671, 679-683
 - library methods, 113-114
 - memory allocation, 115-116
 - operations, 112
 - radix sorting, 420, 432-433, 443
 - search, 113
 - sort, 118, 280
 - Unicode, 527, 666
- String data type, 177
- Strong, H. R., 706
- Structure, data, 4, 69-70, 83
- Stub, 135
- Sublinear sorting, 446, 448-452
- Subtree, 228
- Suffix tree, 680
- Symbol table, 171, 175, 495-530
 - associative memory, 497
 - binary search, 519-523
 - client, 504-505
 - costs, 517
 - count, 509
 - defined, 495
 - existence table, 507
 - index implementations, 524-530
 - item/key implementations, 498-502, 511, 527
 - key-indexed search, 507-510
 - operations, 506
 - ordered/unordered, 515-516
 - performance, 505-506
 - range-search problem, 521
 - running time, 509-510, 520
 - search/insert*, 515-517, 531-533
 - select/sort*, 503, 515
 - sequential search, 511-517
 - TST, 673-675
 - See also* Balanced tree; External searching; Hashing
- Symbolic mathematics, 188
- Szemerédi, E., 464, 622, 623
- Tail node, 102, 103
- Tail recursion, 204, 326-327
- Tarjan, R. E., 63, 569
- Terminal node, 229
- Termination condition, 197
- Ternary search trie (TST), 666-677
 - and BSTs, 676
 - existence, 666-669
 - partial-match, 672
 - and patricia tries, 672
 - performance, 676-677
 - string keys, 670-671
 - symbol table, 673-675
 - worst case, 668-670
- Text-string index, 679-683
- Thrashing, 485
- TI (text index) ADT interface, 525
- Time-space tradeoff, 665
- Tournament construction, 248-249
- Towers of Hanoi problem, 209-214
- Traversal, tree, 240-245
 - orders of, 241-245, 360, 361
 - recursive, 241
 - stack contents, 241-244
- Tree, 197, 227-235
 - AVL, 583
 - B tree, 584, 692-704
 - binomial, 407
 - digital search, 637-640
 - divide-and-conquer, 208-209, 354-355
 - drawing a ruler, 212
 - free, 228, 234-235
 - graph-traversal, 257
 - height of, 237
 - knapsack algorithm, 224
 - M-ary, 229-230, 232
 - ordered/unordered, 229, 232-234
 - parse, 250-251
 - path length, 237
 - quick find, 13
 - quick union, 14, 16, 17
 - quicksort, 327
- red-black, 577-585, 595-598
- rooted, 228, 233-234
- spanning, 10
- subtree, 228
- suffix, 680
- terminology, 228-230
- traversal, 240-245
- 2-3 tree, 584
- 2-3-4 tree, 572-576, 584, 593-594
- types of, 228
- See also* Balanced tree; Binary search tree; Binary tree; Graph
- Trie, 641-677
 - binary, 426
 - defined, 642
 - existence, 662-664, 666-669
 - multiway, 432, 664-665
 - patricia, 650-659, 681-682
 - performance, 657-660, 665-666
 - search/insert*, 643-646, 651-655, 663-670
 - text-string index, 679-683
 - worst case, 646-647
 - See also* Ternary search trie (TST)
- Two-dimensional array, 117-118
 - adjacency matrix/list, 120-124
 - dynamic allocation, 118
 - linked lists, 120-125
- 2-3 tree, 584
- 2-3-4 tree, 572-576, 584, 593-594
- Two-way merging, 348-350
- Type conversion, 72, 155
- UF (union-find) ADT interface, 159
- Undirected graph, 120
- Unicode string, 527, 666
- Union
 - connectivity problem, 10-11, 159-161
 - quick-union algorithm, 13-17
 - slow, 12
- Universal hashing, 605-607
- Unordered tree, 233-234

Upper bound, 62-64

Variable-name-equivalence problem, 8

Vector, 87

as key, 341, 440

Vertex, 120, 228

Virtual machine (VM), 34

Virtual memory, 484-485, 686, 688

visit operation, 535

Vuillemin, J., 407, 408

Weighted quick union, 16-17, 63
equivalence relations ADT, 161

Word, machine, 419-420

key as, 635

Worst-case performance, 35, 60, 62

Yao, A., 701