Table of Contents

Introduction 1

I The Interview 6

1 Getting Ready 7

2 Strategies For A Great Interview 12

3 Conducting An Interview 19

4 Problem Solving 23

II Problems 42

5 Primitive Types 43
- 5.1 Computing the parity of a word . . . 43
- 5.2 Swap bits . . . 45
- 5.3 Reverse bits . . . 46
- 5.4 Find a closest integer with the same weight . . . 47
- 5.5 Compute $x \times y$ without arithmetical operators . . . 48
- 5.6 Compute x/y . . . 50
- 5.7 Compute x^y . . . 51
- 5.8 Reverse digits . . . 52
- 5.9 Check if a decimal integer is a palindrome . . . 52
- 5.10 Generate uniform random numbers . . . 54
- 5.11 Rectangle intersection . . . 55

6 Arrays 57
- 6.1 The Dutch national flag problem . . . 57
- 6.2 Increment an arbitrary-precision integer . . . 61

6.3 Multiply two arbitrary-precision integers . . . 62
6.4 Advancing through an array . . . 63
6.5 Delete a key from an array . . . 64
6.6 Delete duplicates from a sorted array . . . 65
6.7 Buy and sell a stock once . . . 66
6.8 Buy and sell a stock twice . . . 67
6.9 Enumerate all primes to n . . . 68
6.10 Permute the elements of an array . . . 70
6.11 Compute the next permutation . . . 73
6.12 Sample offline data . . . 75
6.13 Sample online data . . . 76
6.14 Compute a random permutation . . . 77
6.15 Compute a random subset . . . 79
6.16 Generate nonuniform random numbers . . . 80
6.17 The Sudoku checker problem . . . 82
6.18 Compute the spiral ordering of a 2D array . . . 84
6.19 Rotate a 2D array . . . 87
6.20 Compute rows in Pascal's Triangle . . . 89

7 Strings **91**
7.1 Interconvert strings and integers . . . 91
7.2 Base conversion . . . 93
7.3 Compute the spreadsheet column encoding . . . 94
7.4 Replace and remove . . . 95
7.5 Test palindromicity . . . 96
7.6 Reverse all the words in a sentence . . . 97
7.7 Compute all mnemonics for a phone number . . . 98
7.8 The look-and-say problem . . . 100
7.9 Convert from Roman to decimal . . . 101
7.10 Compute all valid IP addresses . . . 102
7.11 Write a string sinusoidally . . . 103
7.12 Implement run-length encoding . . . 104
7.13 Implement the UNIX `tail` command . . . 105
7.14 Find the first occurrence of a substring . . . 106

8 Linked Lists **109**
8.1 Merge two sorted lists . . . 110
8.2 Reverse a singly linked list . . . 111
8.3 Reverse a single sublist . . . 112
8.4 Test for cyclicity . . . 113
8.5 Test for overlapping lists—lists are cycle-free . . . 115
8.6 Test for overlapping lists—lists may have cycles . . . 116
8.7 Delete a node from a singly linked list . . . 118
8.8 Remove the kth last element from a list . . . 118
8.9 Remove duplicates from a sorted list . . . 119

8.10 Implement cyclic right shift for singly linked lists . . . 120
8.11 Implement even-odd merge . . . 121
8.12 Test whether a singly linked list is palindromic . . . 123
8.13 Implement list pivoting . . . 124
8.14 Add list-based integers . . . 125

9 Stacks and Queues **127**
9.1 Implement a stack with max API . . . 127
9.2 Evaluate RPN expressions . . . 130
9.3 Test a string over "{,},(,),[,]" for well-formedness . . . 131
9.4 Normalize pathnames . . . 132
9.5 BST keys in sort order . . . 134
9.6 Search a postings list . . . 135
9.7 Compute buildings with a sunset view . . . 136
9.8 Sort a stack . . . 138
9.9 Compute binary tree nodes in order of increasing depth . . . 139
9.10 Implement a circular queue . . . 141
9.11 Implement a queue using stacks . . . 142
9.12 Implement a queue with max API . . . 143

10 Binary Trees **146**
10.1 Test if a binary tree is balanced . . . 148
10.2 Test if a binary tree is symmetric . . . 149
10.3 Compute the lowest common ancestor in a binary tree . . . 150
10.4 Compute the LCA when nodes have parent pointers . . . 152
10.5 Sum the root-to-leaf paths in a binary tree . . . 153
10.6 Find a root to leaf path with specified sum . . . 154
10.7 Compute the kth node in an inorder traversal . . . 155
10.8 Compute the successor . . . 156
10.9 Implement an inorder traversal with $O(1)$ space . . . 157
10.10 Reconstruct a binary tree from traversal data . . . 159
10.11 Reconstruct a binary tree from a preorder traversal with markers . . . 161
10.12 Form a linked list from the leaves of a binary tree . . . 162
10.13 Compute the exterior of a binary tree . . . 163
10.14 Compute the right sibling tree . . . 165
10.15 Implement locking in a binary tree . . . 166

11 Heaps **169**
11.1 Merge sorted files . . . 170
11.2 Sort an increasing-decreasing array . . . 172
11.3 Sort an almost-sorted array . . . 173
11.4 Compute the k closest stars . . . 174
11.5 Compute the median of online data . . . 176
11.6 Compute the k largest elements in a max-heap . . . 177
11.7 Implement a stack API using a heap . . . 179

12 Searching **181**
12.1 Search a sorted array for first occurrence of k . . . 183
12.2 Search a sorted array for the first element greater than k . . . 184
12.3 Search a sorted array for entry equal to its index . . . 186
12.4 Search a cyclically sorted array . . . 186
12.5 Compute the integer square root . . . 188
12.6 Compute the real square root . . . 189
12.7 Search in a 2D sorted array . . . 191
12.8 Find the min and max simultaneously . . . 192
12.9 Find the kth largest element . . . 194
12.10 Compute the optimum mailbox placement . . . 196
12.11 Find the missing IP address . . . 197
12.12 Find the duplicate and missing elements . . . 199

13 Hash Tables **202**
13.1 Partition into anagrams . . . 203
13.2 Test for palindromic permutations . . . 204
13.3 Is an anonymous letter constructible? . . . 205
13.4 Implement an ISBN cache . . . 207
13.5 Compute the LCA, optimizing for close ancestors . . . 209
13.6 Compute the k most frequent queries . . . 210
13.7 Find the nearest repeated entries in an array . . . 210
13.8 Find the smallest subarray covering all values . . . 211
13.9 Find smallest subarray sequentially covering all values . . . 215
13.10 Find the longest subarray with distinct entries . . . 217
13.11 Find the length of a longest contained range . . . 218
13.12 Compute the average of the top three scores . . . 220
13.13 Compute all string decompositions . . . 221
13.14 Find a highest affinity pair . . . 223
13.15 Test the Collatz conjecture . . . 224
13.16 Implement a hash function for chess . . . 226

14 Sorting **228**
14.1 Compute the intersection of two sorted arrays . . . 229
14.2 Implement mergesort in-place . . . 231
14.3 Count the frequencies of characters in a sentence . . . 232
14.4 Find unique elements . . . 233
14.5 Render a calendar . . . 233
14.6 Sets of disjoint intervals . . . 235
14.7 Compute the union of intervals . . . 237
14.8 Partitioning and sorting an array with many repeated entries . . . 239
14.9 Team photo day—1 . . . 241
14.10 Implement a fast sorting algorithm for lists . . . 243
14.11 Compute a salary threshold . . . 244

15 Binary Search Trees **247**
15.1 Test if a binary tree satisfies the BST property . . . 247
15.2 Find the first occurrence of a key in a BST . . . 250
15.3 Find the first key larger than a given value in a BST . . . 252
15.4 Find the k largest elements in a BST . . . 253
15.5 Compute the LCA in a BST . . . 254
15.6 Reconstruct a BST from traversal data . . . 255
15.7 Find the closest entries in three sorted arrays . . . 258
15.8 Enumerate numbers of the form $a + b\sqrt{2}$ . . . 260
15.9 The most visited pages problem . . . 262
15.10 Build a minimum height BST from a sorted array . . . 264
15.11 Insertion and deletion in a BST . . . 265
15.12 Test if three BST nodes are totally ordered . . . 267
15.13 The range lookup problem . . . 269
15.14 Add credits . . . 271
15.15 Count the number of entries in an interval . . . 273

16 Recursion **275**
16.1 The Tower of Hanoi problem . . . 275
16.2 Generate all nonattacking placements of n-Queens . . . 278
16.3 Generate permutations . . . 280
16.4 Generate the power set . . . 282
16.5 Generate all subsets of size k . . . 284
16.6 Generate strings of matched parens . . . 285
16.7 Generate palindromic decompositions . . . 286
16.8 Generate binary trees . . . 288
16.9 Implement a Sudoku solver . . . 289
16.10 Compute a Gray code . . . 291
16.11 Compute the diameter of a tree . . . 293

17 Dynamic Programming **296**
17.1 Count the number of score combinations . . . 298
17.2 Compute the Levenshtein distance . . . 300
17.3 Count the number of ways to traverse a 2D array . . . 302
17.4 Plan a fishing trip . . . 305
17.5 Search for a sequence in a 2D array . . . 305
17.6 The knapsack problem . . . 307
17.7 Divide the spoils fairly . . . 309
17.8 The bedbathandbeyond.com problem . . . 310
17.9 Find the minimum weight path in a triangle . . . 312
17.10 Pick up coins for maximum gain . . . 313
17.11 Count the number of moves to climb stairs . . . 315
17.12 Compute the probability of a Republican majority . . . 316
17.13 The pretty printing problem . . . 318
17.14 Find the longest nondecreasing subsequence . . . 319

18 Greedy Algorithms and Invariants **322**
18.1 Implement Huffman coding . . . 323
18.2 Compute an optimum assignment of tasks . . . 326
18.3 Implement a schedule which minimizes waiting time . . . 327
18.4 The interval covering problem . . . 328
18.5 The 3-sum problem . . . 331
18.6 Find the majority element . . . 332
18.7 The gasup problem . . . 334
18.8 Compute the maximum water trapped by a pair of vertical lines . . . 335
18.9 Compute the largest rectangle under the skyline . . . 337

19 Graphs **340**
19.1 Identify the celebrity . . . 342
19.2 Search a maze . . . 343
19.3 Paint a Boolean matrix . . . 345
19.4 Compute enclosed regions . . . 347
19.5 Degrees of connectedness—1 . . . 349
19.6 Clone a graph . . . 351
19.7 Making wired connections . . . 352
19.8 Transform one string to another . . . 354
19.9 The shortest straight-line program for x^n . . . 355
19.10 Team photo day—2 . . . 357
19.11 Compute a shortest path with fewest edges . . . 358

20 Parallel Computing **361**
20.1 Implement caching for a multithreaded dictionary . . . 362
20.2 Analyze two unsynchronized interleaved threads . . . 364
20.3 Implement synchronization for two interleaving threads . . . 365
20.4 Implement a thread pool . . . 367
20.5 Implement asynchronous callbacks . . . 368
20.6 Implement a Timer class . . . 369
20.7 The readers-writers problem . . . 370
20.8 The readers-writers problem with write preference . . . 372
20.9 Test the Collatz conjecture in parallel . . . 372
20.10 Design TeraSort and PetaSort . . . 374
20.11 Implement distributed throttling . . . 375

21 Design Problems **376**
21.1 Design a spell checker . . . 377
21.2 Design a solution to the stemming problem . . . 378
21.3 Plagiarism detector . . . 379
21.4 Pair users by attributes . . . 380
21.5 Design a system for detecting copyright infringement . . . 381
21.6 Design TEX . . . 382
21.7 Design a search engine . . . 382

21.8 Implement PageRank . . . 384
21.9 Design a scalable priority system . . . 385
21.10 Create photomosaics . . . 385
21.11 Implement Mileage Run . . . 386
21.12 Implement Connexus . . . 387
21.13 Design an online advertising system . . . 388
21.14 Design a recommendation system . . . 389
21.15 Design an optimized way of distributing large files . . . 390
21.16 Design the World Wide Web . . . 391
21.17 Estimate the hardware cost of a photo sharing app . . . 392

22 Honors Class **393**
22.1 Compute the greatest common divisor . . . 393
22.2 Find the first missing positive entry . . . 395
22.3 Buy and sell a stock *k* times . . . 396
22.4 Compute the maximum product of all entries but one . . . 397
22.5 Compute the longest contiguous increasing subarray . . . 399
22.6 Rotate an array . . . 401
22.7 Identify positions attacked by rooks . . . 402
22.8 Justify text . . . 404
22.9 Reverse sublists *k* at a time . . . 406
22.10 Implement list zipping . . . 407
22.11 Copy a postings list . . . 408
22.12 Compute the median of a sorted circular linked list . . . 409
22.13 Compute the longest substring with matching parens . . . 410
22.14 Compute the maximum of a sliding window . . . 412
22.15 Implement preorder and postorder traversals without recursion . . . 413
22.16 Compute fair bonuses . . . 416
22.17 Find *k* elements closest to the median . . . 419
22.18 Search a sorted array of unknown length . . . 420
22.19 Search in two sorted arrays . . . 422
22.20 Find the *k*th largest element—large *n*, small *k* . . . 423
22.21 Find an element that appears only once . . . 424
22.22 Find the line through the most points . . . 426
22.23 Find the shortest unique prefix . . . 428
22.24 Compute the smallest nonconstructible change . . . 430
22.25 Find the most visited pages in a window . . . 432
22.26 Convert a sorted doubly linked list into a BST . . . 433
22.27 Convert a BST to a sorted doubly linked list . . . 434
22.28 Merge two BSTs . . . 436
22.29 Test if a binary tree is an almost BST . . . 437
22.30 The view from above . . . 439
22.31 Searching a min-first BST . . . 442
22.32 Implement regular expression matching . . . 443

22.33 Synthesize an expression . . . 446
22.34 Count inversions . . . 449
22.35 Draw the skyline . . . 451
22.36 Find the two closest points . . . 454
22.37 Measure with defective jugs . . . 457
22.38 Compute the maximum subarray sum in a circular array . . . 459
22.39 Determine the critical height . . . 460
22.40 Voltage selection in a logic circuit . . . 462
22.41 Find the maximum 2D subarray . . . 463
22.42 Trapping water . . . 466
22.43 Load balancing . . . 467
22.44 Search for a pair-sum in an abs-sorted array . . . 469
22.45 The heavy hitter problem . . . 472
22.46 Find the longest subarray whose sum $\leq k$ . . . 473
22.47 Degrees of connectedness—2 . . . 475
22.48 Compute a minimum delay schedule, unlimited resources . . . 477
22.49 Road network . . . 477
22.50 Test if arbitrage is possible . . . 479
22.51 The readers-writers problem with fairness . . . 481
22.52 Implement a producer-consumer queue . . . 481

III Notation, Language, and Index **483**

Notation **484**

C++ best practices, C++11, and C++ for Java developers **486**

Index of Terms **488**