Table of Contents

Introduction **1**

I The Interview **6**

1 Getting Ready **7**

2 Strategies For A Great Interview **13**

3 Conducting An Interview **19**

4 Problem Solving **22**

II Problems **39**

5 Primitive Types **40**

5.1 Computing the parity of a word 41
5.2 Swap bits 43
5.3 Reverse bits 44
5.4 Find a closest integer with the same weight 45
5.5 Compute $x \times y$ without arithmetical operators 46
5.6 Compute x/y 47
5.7 Compute x^y 48
5.8 Reverse digits 49
5.9 Check if a decimal integer is a palindrome 50
5.10 Generate uniform random numbers 51
5.11 Rectangle intersection 52

6 Arrays **54**

6.1 The Dutch national flag problem 56
6.2 Increment an arbitrary-precision integer 59
6.3 Multiply two arbitrary-precision integers 59
6.4 Advancing through an array 61
6.5 Delete duplicates from a sorted array 62

6.6 Buy and sell a stock once . . . 63
6.7 Buy and sell a stock twice . . . 63
6.8 Enumerate all primes to *n* . . . 64
6.9 Permute the elements of an array . . . 66
6.10 Compute the next permutation . . . 68
6.11 Sample offline data . . . 70
6.12 Sample online data . . . 71
6.13 Compute a random permutation . . . 72
6.14 Compute a random subset . . . 73
6.15 Generate nonuniform random numbers . . . 74
6.16 The Sudoku checker problem . . . 76
6.17 Compute the spiral ordering of a 2D array . . . 78
6.18 Rotate a 2D array . . . 81
6.19 Compute rows in Pascal's Triangle . . . 82

7 Strings 84
7.1 Interconvert strings and integers . . . 85
7.2 Base conversion . . . 86
7.3 Compute the spreadsheet column encoding . . . 87
7.4 Replace and remove . . . 88
7.5 Test palindromicity . . . 89
7.6 Reverse all the words in a sentence . . . 90
7.7 Compute all mnemonics for a phone number . . . 91
7.8 The look-and-say problem . . . 93
7.9 Convert from Roman to decimal . . . 94
7.10 Compute all valid IP addresses . . . 95
7.11 Write a string sinusoidally . . . 96
7.12 Implement run-length encoding . . . 97
7.13 Find the first occurrence of a substring . . . 98

8 Linked Lists 100
8.1 Merge two sorted lists . . . 102
8.2 Reverse a single sublist . . . 104
8.3 Test for cyclicity . . . 105
8.4 Test for overlapping lists—lists are cycle-free . . . 106
8.5 Test for overlapping lists—lists may have cycles . . . 108
8.6 Delete a node from a singly linked list . . . 110
8.7 Remove the *k*th last element from a list . . . 110
8.8 Remove duplicates from a sorted list . . . 111
8.9 Implement cyclic right shift for singly linked lists . . . 112
8.10 Implement even-odd merge . . . 113
8.11 Test whether a singly linked list is palindromic . . . 114
8.12 Implement list pivoting . . . 115
8.13 Add list-based integers . . . 116

9 Stacks and Queues 118
9.1 Implement a stack with max API . . . 119

9.2 Evaluate RPN expressions . . . 122
9.3 Test a string over “{,},(,),[,]” for well-formedness . . . 123
9.4 Normalize pathnames . . . 124
9.5 Search a postings list . . . 125
9.6 Compute buildings with a sunset view . . . 127
9.7 Compute binary tree nodes in order of increasing depth . . . 130
9.8 Implement a circular queue . . . 131
9.9 Implement a queue using stacks . . . 132
9.10 Implement a queue with max API . . . 133

10 Binary Trees **136**
10.1 Test if a binary tree is height-balanced . . . 138
10.2 Test if a binary tree is symmetric . . . 140
10.3 Compute the lowest common ancestor in a binary tree . . . 141
10.4 Compute the LCA when nodes have parent pointers . . . 142
10.5 Sum the root-to-leaf paths in a binary tree . . . 143
10.6 Find a root to leaf path with specified sum . . . 145
10.7 Implement an inorder traversal without recursion . . . 145
10.8 Implement a preorder traversal without recursion . . . 146
10.9 Compute the *k*th node in an inorder traversal . . . 147
10.10 Compute the successor . . . 148
10.11 Implement an inorder traversal with $O(1)$ space . . . 149
10.12 Reconstruct a binary tree from traversal data . . . 150
10.13 Reconstruct a binary tree from a preorder traversal with markers . . . 152
10.14 Form a linked list from the leaves of a binary tree . . . 153
10.15 Compute the exterior of a binary tree . . . 154
10.16 Compute the right sibling tree . . . 155
10.17 Implement locking in a binary tree . . . 157

11 Heaps **159**
11.1 Merge sorted files . . . 160
11.2 Sort an increasing-decreasing array . . . 163
11.3 Sort an almost-sorted array . . . 164
11.4 Compute the k closest stars . . . 165
11.5 Compute the median of online data . . . 166
11.6 Compute the k largest elements in a max-heap . . . 167
11.7 Implement a stack API using a heap . . . 169

12 Searching **171**
12.1 Search a sorted array for first occurrence of k . . . 174
12.2 Search a sorted array for entry equal to its index . . . 175
12.3 Search a cyclically sorted array . . . 176
12.4 Compute the integer square root . . . 178
12.5 Compute the real square root . . . 179
12.6 Search in a 2D sorted array . . . 180
12.7 Find the min and max simultaneously . . . 182
12.8 Find the *k*th largest element . . . 183

12.9 Find the missing IP address . . . 185
12.10 Find the duplicate and missing elements . . . 187

13 Hash Tables **190**
13.1 Test for palindromic permutations . . . 194
13.2 Is an anonymous letter constructible? . . . 195
13.3 Implement an ISBN cache . . . 196
13.4 Compute the LCA, optimizing for close ancestors . . . 198
13.5 Compute the *k* most frequent queries . . . 199
13.6 Find the nearest repeated entries in an array . . . 199
13.7 Find the smallest subarray covering all values . . . 200
13.8 Find smallest subarray sequentially covering all values . . . 204
13.9 Find the longest subarray with distinct entries . . . 206
13.10 Find the length of a longest contained interval . . . 207
13.11 Compute the average of the top three scores . . . 208
13.12 Compute all string decompositions . . . 210
13.13 Test the Collatz conjecture . . . 211
13.14 Implement a hash function for chess . . . 213

14 Sorting **215**
14.1 Compute the intersection of two sorted arrays . . . 217
14.2 Merge two sorted arrays . . . 219
14.3 Remove first-name duplicates . . . 219
14.4 Render a calendar . . . 220
14.5 Merging intervals . . . 222
14.6 Compute the union of intervals . . . 224
14.7 Partitioning and sorting an array with many repeated entries . . . 226
14.8 Team photo day—1 . . . 228
14.9 Implement a fast sorting algorithm for lists . . . 230
14.10 Compute a salary threshold . . . 231

15 Binary Search Trees **233**
15.1 Test if a binary tree satisfies the BST property . . . 235
15.2 Find the first key greater than a given value in a BST . . . 237
15.3 Find the *k* largest elements in a BST . . . 238
15.4 Compute the LCA in a BST . . . 239
15.5 Reconstruct a BST from traversal data . . . 240
15.6 Find the closest entries in three sorted arrays . . . 243
15.7 Enumerate numbers of the form $a + b\sqrt{2}$ . . . 245
15.8 The most visited pages problem . . . 247
15.9 Build a minimum height BST from a sorted array . . . 249
15.10 Insertion and deletion in a BST . . . 250
15.11 Test if three BST nodes are totally ordered . . . 252
15.12 The range lookup problem . . . 254
15.13 Add credits . . . 256

16 Recursion **259**
16.1 The Towers of Hanoi problem 260
16.2 Generate all nonattacking placements of n-Queens 262
16.3 Generate permutations 264
16.4 Generate the power set 265
16.5 Generate all subsets of size k 267
16.6 Generate strings of matched parens 268
16.7 Generate palindromic decompositions 270
16.8 Generate binary trees 271
16.9 Implement a Sudoku solver 272
16.10 Compute a Gray code 274
16.11 Compute the diameter of a tree 276

17 Dynamic Programming **279**
17.1 Count the number of score combinations 282
17.2 Compute the Levenshtein distance 284
17.3 Count the number of ways to traverse a 2D array 287
17.4 Compute the binomial coefficients 289
17.5 Search for a sequence in a 2D array 290
17.6 The knapsack problem 292
17.7 The bedbathandbeyond.com problem 294
17.8 Find the minimum weight path in a triangle 297
17.9 Pick up coins for maximum gain 298
17.10 Count the number of moves to climb stairs 300
17.11 The pretty printing problem 301
17.12 Find the longest nondecreasing subsequence 303

18 Greedy Algorithms and Invariants **306**
18.1 Compute an optimum assignment of tasks 307
18.2 Schedule to minimize waiting time 308
18.3 The interval covering problem 309
18.4 The 3-sum problem 312
18.5 Find the majority element 313
18.6 The gasup problem 315
18.7 Compute the maximum water trapped by a pair of vertical lines 317
18.8 Compute the largest rectangle under the skyline 318

19 Graphs **322**
19.1 Search a maze 326
19.2 Paint a Boolean matrix 328
19.3 Compute enclosed regions 330
19.4 Deadlock detection 332
19.5 Clone a graph 334
19.6 Making wired connections 335
19.7 Transform one string to another 336
19.8 Team photo day—2 339
19.9 Compute a shortest path with fewest edges 340

20 Parallel Computing **344**
20.1 Implement caching for a multithreaded dictionary 345
20.2 Analyze two unsynchronized interleaved threads 347
20.3 Implement synchronization for two interleaving threads 348
20.4 Implement a thread pool 350
20.5 Deadlock 351
20.6 The readers-writers problem 352
20.7 The readers-writers problem with write preference 354
20.8 Implement a Timer class 354
20.9 Test the Collatz conjecture in parallel 355

III Domain Specific Problems **357**

21 Design Problems **358**
21.1 Design a spell checker 359
21.2 Design a solution to the stemming problem 360
21.3 Plagiarism detector 361
21.4 Pair users by attributes 361
21.5 Design a system for detecting copyright infringement 363
21.6 Design $\TeX$ 363
21.7 Design a search engine 364
21.8 Implement PageRank 365
21.9 Design TeraSort and PetaSort 366
21.10 Implement distributed throttling 367
21.11 Design a scalable priority system 367
21.12 Create photomosaics 368
21.13 Implement Mileage Run 369
21.14 Implement Connexus 370
21.15 Design an online advertising system 370
21.16 Design a recommendation system 371
21.17 Design an optimized way of distributing large files 372
21.18 Design the World Wide Web 373
21.19 Estimate the hardware cost of a photo sharing app 373

22 Language Questions **375**
22.1 The JVM 375
22.2 `throw` vs. `throws` 375
22.3 `final`, `finally`, and `finalizer` 376
22.4 `equals()` vs. `==` 376
22.5 `equals()` and `hashCode()` 376
22.6 `List`, `ArrayList`, and `LinkedList` 377
22.7 `String` vs. `StringBuilder` 377
22.8 Autoboxing 378
22.9 Static initialization 379

23 Object-Oriented Design **380**
23.1 Template Method vs. Strategy 380
23.2 Observer pattern 381
23.3 Push vs. pull observer pattern 381
23.4 Singletons and Flyweights 382
23.5 Adapters 382
23.6 Creational Patterns 383
23.7 Libraries and design patterns 385

24 Common Tools **386**
24.1 Merging in a version control system 386
24.2 Hooks 388
24.3 Is scripting is more efficient? 389
24.4 Polymorphism with a scripting language 390
24.5 Dependency analysis 390
24.6 ANT vs. Maven 391
24.7 SQL vs. NoSQL 392
24.8 Normalization 392
24.9 SQL design 393
24.10 IP, TCP, and HTTP 393
24.11 HTTPS 394
24.12 DNS 395

IV The Honors Class **396**

25 Honors Class **397**
25.1 Compute the greatest common divisor 397
25.2 Find the first missing positive entry 398
25.3 Buy and sell a stock *k* times 400
25.4 Compute the maximum product of all entries but one 400
25.5 Compute the longest contiguous increasing subarray 403
25.6 Rotate an array 404
25.7 Identify positions attacked by rooks 406
25.8 Justify text 408
25.9 Implement list zipping 409
25.10 Copy a postings list 410
25.11 Compute the longest substring with matching parens 412
25.12 Compute the maximum of a sliding window 413
25.13 Implement a postorder traversal without recursion 415
25.14 Compute fair bonuses 417
25.15 Search a sorted array of unknown length 420
25.16 Search in two sorted arrays 421
25.17 Find the *k*th largest element—large *n*, small *k* 422
25.18 Find an element that appears only once 423
25.19 Find the line through the most points 424
25.20 Find the shortest unique prefix 427

25.21 Find the most visited pages in a window . . . 429
25.22 Convert a sorted doubly linked list into a BST . . . 430
25.23 Convert a BST to a sorted doubly linked list . . . 432
25.24 Merge two BSTs . . . 433
25.25 The view from above . . . 434
25.26 Implement regular expression matching . . . 437
25.27 Synthesize an expression . . . 440
25.28 Count inversions . . . 442
25.29 Draw the skyline . . . 444
25.30 Measure with defective jugs . . . 446
25.31 Compute the maximum subarray sum in a circular array . . . 448
25.32 Determine the critical height . . . 450
25.33 Find the maximum 2D subarray . . . 452
25.34 Implement Huffman coding . . . 455
25.35 Trapping water . . . 459
25.36 Search for a pair-sum in an abs-sorted array . . . 460
25.37 The heavy hitter problem . . . 463
25.38 Find the longest subarray whose sum $\leq k$ . . . 465
25.39 Road network . . . 466
25.40 Test if arbitrage is possible . . . 468

V Notation, and Index **471**

Notation **472**

Index of Terms **474**