



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 | 41 |
| 5 Primitive Types | 42 |
| 5.1 Compute parity | 42 |
| 5.2 Swap bits | 44 |
| 5.3 Reverse bits | 45 |
| 5.4 Find a closest integer with the same weight 🐙 | 46 |
| 5.5 Compute $x \times y$ without multiply or add | 46 |
| 5.6 Compute x/y 🐙 | 47 |
| 5.7 Compute x^y | 48 |
| 5.8 Convert base | 49 |
| 5.9 Compute the spreadsheet column encoding | 50 |
| 5.10 Reverse digits | 50 |
| 5.11 Check if a decimal integer is a palindrome | 51 |
| 5.12 Generate uniform random numbers | 52 |
| 5.13 Check if rectangles intersect | 53 |
| 5.14 The open doors problem | 54 |
| 5.15 Compute the greatest common divisor 🐙 | 55 |



















| | | |
|----------|--|-----------|
| 6 | Arrays | 57 |
| 6.1 | The Dutch national flag problem | 57 |
| 6.2 | Increment a BigInteger | 59 |
| 6.3 | Multiply two BigIntegers | 60 |
| 6.4 | Check if a board game is winnable | 61 |
| 6.5 | Delete a key from an array | 62 |
| 6.6 | Delete duplicates from a sorted array | 63 |
| 6.7 | Find the first missing positive entry | 64 |
| 6.8 | Compute the max difference | 65 |
| 6.9 | Generalizations of max difference 🐞 | 66 |
| 6.10 | Compute the maximum product of all but one entries 🐞 | 67 |
| 6.11 | Compute the longest contiguous increasing subarray 🐞 | 70 |
| 6.12 | Enumerate all primes to n 🐞 | 72 |
| 6.13 | Permute the elements of an array 🐞 | 73 |
| 6.14 | Compute the next permutation | 75 |
| 6.15 | Rotate an array 🐞 | 76 |
| 6.16 | Sample offline data | 78 |
| 6.17 | Compute a random permutation | 79 |
| 6.18 | Compute a random subset of $\{0, 1, \dots, n - 1\}$ | 79 |
| 6.19 | Sample online data | 81 |
| 6.20 | Generate nonuniform random numbers | 81 |
| 6.21 | The Sudoku checker problem | 83 |
| 6.22 | Compute the spiral ordering of a 2D array | 84 |
| 6.23 | Rotate a 2D array | 86 |
| 6.24 | Compute rows in Pascal's Triangle | 88 |
| 6.25 | Identify positions attacked by rooks 🐞 | 89 |
| 6.26 | Identify the celebrity 🐞 | 91 |
| 7 | Strings | 93 |
| 7.1 | Interconvert strings and integers | 93 |
| 7.2 | Replace and remove | 94 |
| 7.3 | Test palindromicity | 96 |
| 7.4 | Reverse all the words in a sentence | 97 |
| 7.5 | Compute all mnemonics for a phone number | 98 |
| 7.6 | The look-and-say problem | 99 |
| 7.7 | Convert from Roman to decimal | 100 |
| 7.8 | Compute all valid IP addresses | 101 |
| 7.9 | Write a string sinusoidally | 103 |
| 7.10 | Implement run-length encoding | 103 |
| 7.11 | Implement Elias gamma encoding | 104 |
| 7.12 | Implement the UNIX tail command | 106 |
| 7.13 | Justify text 🐞 | 107 |
| 7.14 | Find the first occurrence of a substring 🐞 | 108 |

| | | |
|-----------|--|------------|
| 8 | Linked Lists | 111 |
| 8.1 | Merge two sorted lists | 112 |
| 8.2 | Reverse a singly linked list | 113 |
| 8.3 | Reverse a single sublist | 114 |
| 8.4 | Reverse sublists k at a time | 115 |
| 8.5 | Test for cyclicity | 116 |
| 8.6 | Test for overlapping lists—lists are cycle-free | 118 |
| 8.7 | Test for overlapping lists—lists may have cycles | 120 |
| 8.8 | Delete a node from a singly linked list | 121 |
| 8.9 | Remove the k -th last element from a list | 122 |
| 8.10 | Remove duplicates from a sorted list | 122 |
| 8.11 | Implement cyclic right shift for singly linked lists | 123 |
| 8.12 | Implement even-odd merge | 125 |
| 8.13 | Implement list zipping 🧐 | 126 |
| 8.14 | Copy a postings list 🧐 | 127 |
| 8.15 | Test whether a singly linked list is palindromic | 129 |
| 8.16 | Compute the median of a sorted circular linked list | 130 |
| 8.17 | Implement list pivoting | 131 |
| 8.18 | Sort a list | 132 |
| 8.19 | Add list-based integers | 133 |
| 9 | Stacks and Queues | 135 |
| 9.1 | Implement a stack with max API | 135 |
| 9.2 | Evaluate RPN expressions | 138 |
| 9.3 | Test a string over "{,},(,),[,]" for well-formedness | 139 |
| 9.4 | Compute the longest substring with matching parens 🧐 | 140 |
| 9.5 | Normalize pathnames | 142 |
| 9.6 | BST keys in sort order | 143 |
| 9.7 | Search a postings list | 144 |
| 9.8 | Compute buildings with a sunset view | 145 |
| 9.9 | Sort a stack | 146 |
| 9.10 | Compute binary tree nodes in order of increasing depth | 148 |
| 9.11 | Implement a circular queue | 149 |
| 9.12 | Implement a queue using stacks | 150 |
| 9.13 | Implement a queue with max API 🧐 | 151 |
| 9.14 | Compute the maximum of a sliding window 🧐 | 152 |
| 9.15 | The shortest straight-line program for x^n | 154 |
| 10 | Binary Trees | 156 |
| 10.1 | Test if a binary tree is balanced | 158 |
| 10.2 | Find k -unbalanced nodes | 159 |
| 10.3 | Test if a binary tree is symmetric | 161 |
| 10.4 | Compute the lowest common ancestor in a binary tree | 162 |
| 10.5 | Compute the LCA when nodes have parent pointers | 163 |
| 10.6 | Sum the root-to-leaf paths in a binary tree | 164 |

| | | |
|-----------|--|------------|
| 10.7 | Find a root to leaf path with specified sum | 165 |
| 10.8 | Compute the k -th node in an inorder traversal | 166 |
| 10.9 | Implement an inorder traversal with $O(1)$ space | 167 |
| 10.10 | Implement preorder and postorder traversals without recursion 🧐 | 169 |
| 10.11 | Compute the successor | 172 |
| 10.12 | Reconstruct a binary tree from traversal data | 173 |
| 10.13 | Reconstruct a binary tree from a preorder traversal with markers | 175 |
| 10.14 | Form a linked list from the leaves of a binary tree | 176 |
| 10.15 | Compute the exterior of a binary tree 🧐 | 177 |
| 10.16 | Compute the right sibling tree | 179 |
| 10.17 | Implement locking in a binary tree | 180 |
| 11 | Heaps | 183 |
| 11.1 | Merge sorted files | 184 |
| 11.2 | Sort an increasing-decreasing array | 185 |
| 11.3 | Sort an almost-sorted array | 186 |
| 11.4 | Compute the k closest stars | 187 |
| 11.5 | Compute the median of online data | 189 |
| 11.6 | Compute the k largest elements in a max-heap | 190 |
| 11.7 | Compute fair bonuses 🧐 | 191 |
| 11.8 | Find k elements closest to the median 🧐 | 194 |
| 11.9 | The k -th largest element in a max-heap 🧐 | 195 |
| 11.10 | Implement a stack API using a heap | 196 |
| 12 | Searching | 198 |
| 12.1 | Search a sorted array for first occurrence of k | 200 |
| 12.2 | Search a sorted array for the first element greater than k | 201 |
| 12.3 | Search a sorted array for entry equal to its index | 203 |
| 12.4 | Search a cyclically sorted array | 203 |
| 12.5 | Search a sorted array of unknown length 🧐 | 205 |
| 12.6 | Compute the integer square root | 207 |
| 12.7 | Compute the real square root | 208 |
| 12.8 | Search in two sorted arrays 🧐 | 209 |
| 12.9 | Search in a 2D sorted array 🧐 | 210 |
| 12.10 | Find the min and max simultaneously | 212 |
| 12.11 | Find the k -th largest element | 213 |
| 12.12 | Compute the optimum mailbox placement | 215 |
| 12.13 | Find the k -th largest element—large n , small k 🧐 | 216 |
| 12.14 | Find the missing IP address | 217 |
| 12.15 | Find the duplicate and missing elements | 219 |
| 12.16 | Find an element that appears only once 🧐 | 221 |
| 13 | Hash Tables | 224 |
| 13.1 | Partition into anagrams | 225 |
| 13.2 | Test for palindromic permutations | 226 |

| | | |
|-----------|--|------------|
| 13.3 | Is an anonymous letter constructible? | 227 |
| 13.4 | Implement an ISBN cache | 228 |
| 13.5 | Compute the LCA, optimizing for close ancestors | 230 |
| 13.6 | Compute the k most frequent queries | 231 |
| 13.7 | Find the line through the most points 🐼 | 232 |
| 13.8 | Find the nearest repeated entries in an array | 235 |
| 13.9 | Find the smallest subarray covering all values 🐼 | 235 |
| 13.10 | Find smallest subarray that sequentially covering all values 🐼 | 239 |
| 13.11 | Find the longest subarray with distinct entries 🐼 | 241 |
| 13.12 | Find the length of a longest contained range 🐼 | 242 |
| 13.13 | Compute all string decompositions | 243 |
| 13.14 | Find a highest affinity pair | 245 |
| 13.15 | Pair users by attributes | 246 |
| 13.16 | Test the Collatz conjecture | 247 |
| 13.17 | Implement a hash function for chess | 248 |
| 13.18 | Find the shortest unique prefix 🐼 | 250 |
| 14 | Sorting | 253 |
| 14.1 | Compute the intersection of two sorted arrays | 254 |
| 14.2 | Implement mergesort in-place | 256 |
| 14.3 | Count the frequencies of characters in a sentence | 257 |
| 14.4 | Find unique elements | 258 |
| 14.5 | Render a calendar | 258 |
| 14.6 | Sets of disjoint intervals | 260 |
| 14.7 | Compute the union of intervals | 262 |
| 14.8 | The interval covering problem | 264 |
| 14.9 | Compute an optimum assignment of tasks 🐼 | 266 |
| 14.10 | Implement counting sort 🐼 | 267 |
| 14.11 | Team photo day—1 | 269 |
| 14.12 | Implement a fast sorting algorithm for lists | 271 |
| 14.13 | Compute the smallest nonconstructible change 🐼 | 272 |
| 14.14 | Compute a salary threshold | 273 |
| 14.15 | Implement a variable-length sort | 275 |
| 14.16 | Schedule time trials | 275 |
| 14.17 | Find the winner and runner-up | 276 |
| 15 | Binary Search Trees | 278 |
| 15.1 | Test if a binary tree satisfies the BST property | 278 |
| 15.2 | Find the first occurrence of a key in a BST | 281 |
| 15.3 | Find the first key larger than a given value in a BST | 283 |
| 15.4 | Find the k largest elements in a BST | 284 |
| 15.5 | Compute the LCA in a BST | 285 |
| 15.6 | Reconstruct a BST from traversal data | 286 |
| 15.7 | Find the closest entries in three sorted arrays | 289 |
| 15.8 | The most visited pages problem | 291 |

| | | |
|-----------|--|------------|
| 15.9 | Find the most visited pages in a window 🧐 | 292 |
| 15.10 | Build a BST from a sorted array | 293 |
| 15.11 | Convert a sorted doubly linked list into a BST 🧐 | 294 |
| 15.12 | Convert a BST to a sorted doubly linked list 🧐 | 296 |
| 15.13 | Merge two BSTs 🧐 | 297 |
| 15.14 | Insertion and deletion in a BST 🧐 | 298 |
| 15.15 | Test if three BST nodes are totally ordered | 302 |
| 15.16 | Test if a binary tree is an almost BST 🧐 | 304 |
| 15.17 | Compute the average of the top three scores | 305 |
| 15.18 | The range lookup problem | 307 |
| 15.19 | The view from above 🧐 | 309 |
| 15.20 | Searching a min-first BST | 311 |
| 15.21 | Add credits | 313 |
| 15.22 | Count the number of entries in an interval | 315 |
| 16 | Recursion | 317 |
| 16.1 | The Towers of Hanoi problem | 317 |
| 16.2 | Enumerate all nonattacking placements of n -Queens | 319 |
| 16.3 | Enumerate permutations | 321 |
| 16.4 | Enumerate the power set | 322 |
| 16.5 | Enumerate all subsets of size k | 324 |
| 16.6 | Enumerate strings of balanced parens | 325 |
| 16.7 | Enumerate palindromic decompositions | 326 |
| 16.8 | Enumerate binary trees | 327 |
| 16.9 | Implement a Sudoku solver | 328 |
| 16.10 | Compute a Gray code | 330 |
| 16.11 | Implement regular expression matching 🧐 | 331 |
| 16.12 | Synthesize an expression | 334 |
| 16.13 | Count inversions 🧐 | 337 |
| 16.14 | Compute the diameter of a tree | 338 |
| 16.15 | Draw the skyline 🧐 | 340 |
| 16.16 | Find the two closest points 🧐 | 342 |
| 17 | Dynamic Programming | 346 |
| 17.1 | Count the number of score combinations | 348 |
| 17.2 | Compute the Levenshtein distance | 349 |
| 17.3 | Compute the binomial coefficients | 352 |
| 17.4 | Count the number of ways to traverse a 2D array | 353 |
| 17.5 | Plan a fishing trip | 355 |
| 17.6 | Search for a sequence in a 2D array | 355 |
| 17.7 | The knapsack problem | 357 |
| 17.8 | Measure with defective jugs 🧐 | 358 |
| 17.9 | Test if a tie is possible | 360 |
| 17.10 | Divide the spoils fairly | 361 |
| 17.11 | Compute the maximum subarray sum in a circular array 🧐 | 362 |

| | | |
|-----------|---|------------|
| 17.12 | The bedbathandbeyond.com problem | 364 |
| 17.13 | Determine the critical height  | 365 |
| 17.14 | Find the maximum weight path in a triangle | 367 |
| 17.15 | Pick up coins for maximum gain | 367 |
| 17.16 | Decompose into palindromic strings | 369 |
| 17.17 | Test if s is an interleaving of s_1 and s_2 | 370 |
| 17.18 | Count the number of steps in a board game | 371 |
| 17.19 | Compute the probability of a Republican majority | 372 |
| 17.20 | The pretty printing problem | 374 |
| 17.21 | Find the longest nondecreasing subsequence  | 375 |
| 17.22 | Voltage selection in a logic circuit  | 378 |
| 17.23 | Find the maximum 2D subarray  | 379 |
| 18 | Greedy Algorithms and Invariants | 383 |
| 18.1 | Implement Huffman coding  | 384 |
| 18.2 | Implement a schedule which minimizes waiting time | 387 |
| 18.3 | Trapping water  | 388 |
| 18.4 | Load balancing  | 389 |
| 18.5 | Pack for USPS priority mail  | 391 |
| 18.6 | The 3-sum problem | 393 |
| 18.7 | The gasup problem | 395 |
| 18.8 | Enumerate numbers of the form $a + b\sqrt{2}$  | 396 |
| 18.9 | Find the majority element | 398 |
| 18.10 | Search for a pair-sum in an abs-sorted array  | 399 |
| 18.11 | Compute the maximum water trapped by a pair of vertical lines | 402 |
| 18.12 | The heavy hitter problem  | 403 |
| 18.13 | Find the longest subarray whose sum $\leq k$  | 405 |
| 18.14 | Compute the largest rectangle under the skyline  | 407 |
| 19 | Graphs | 410 |
| 19.1 | Search a maze | 413 |
| 19.2 | Paint a Boolean matrix | 414 |
| 19.3 | Compute enclosed regions | 417 |
| 19.4 | Clone a graph | 419 |
| 19.5 | Transform one string to another  | 419 |
| 19.6 | Making wired connections | 421 |
| 19.7 | Test degrees of connectedness  | 422 |
| 19.8 | Team photo day—2 | 426 |
| 19.9 | Compute a minimum delay schedule, unlimited resources  | 427 |
| 19.10 | Compute a shortest path with fewest edges | 428 |
| 19.11 | Road network  | 430 |
| 19.12 | Test if arbitrage is possible  | 431 |
| 20 | Parallel Computing | 434 |
| 20.1 | Implement caching for a multithreaded dictionary | 435 |

| | | |
|------------|--|------------|
| 20.2 | Analyze two unsynchronized interleaved threads | 437 |
| 20.3 | Implement synchronization for two interleaving threads | 438 |
| 20.4 | Implement a thread pool | 440 |
| 20.5 | Implement asynchronous callbacks | 441 |
| 20.6 | Implement a Timer class | 442 |
| 20.7 | The readers-writers problem | 443 |
| 20.8 | The readers-writers problem with write preference | 444 |
| 20.9 | The readers-writers problem with fairness | 445 |
| 20.10 | Implement a producer-consumer queue | 445 |
| 20.11 | Test the Collatz conjecture in parallel | 446 |
| 20.12 | Implement broadcast in a tree-structured network 🕸 | 448 |
| 20.13 | Design TeraSort and PetaSort | 448 |
| 20.14 | Implement distributed throttling | 449 |
| 21 | Design Problems | 451 |
| 21.1 | Create photomosaics | 453 |
| 21.2 | Design a spell checker | 453 |
| 21.3 | Design a solution to the stemming problem | 454 |
| 21.4 | Plagiarism detector | 455 |
| 21.5 | Design a system for detecting copyright infringement | 456 |
| 21.6 | Design TeX | 456 |
| 21.7 | Design a search engine | 457 |
| 21.8 | Implement PageRank | 458 |
| 21.9 | Design a scalable priority system | 459 |
| 21.10 | Implement Mileage Run | 460 |
| 21.11 | Implement Connexus | 462 |
| 21.12 | Design an online advertising system | 463 |
| 21.13 | Design a recommendation system | 464 |
| 21.14 | Design an optimized way of distributing large files | 464 |
| 21.15 | Design the World Wide Web | 465 |
| 21.16 | Estimate the hardware cost of a photo sharing app | 466 |
| III | Notation, Language, and Index | 468 |
| | Notation | 469 |
| | C++11, and C++ for Java developers | 471 |
| | Index of Terms | 473 |