



Practice for Cracking Any Coding Interview

[Read](#)[Discuss](#)[Courses](#)[Practice](#)[Video](#)

Coding questions in this article are **difficulty wise ordered**. The idea of this post is to target two types of people.

1. **Competitive Programming Preparation (For 1st and 2nd Year Students)** : It is recommended to finish all questions from all categories except possibly Linked List, Tree and BST. However at least 10 questions from these categories should also be covered. If you have never done competitive programming before, it is strongly recommended to see [How to Begin with Competitive Programming](#) first. If you wish to get yourself prepared with a language first, you may first begin [C++ Track](#) or [Java Track](#)
2. **Interview preparation** It is recommended to cover all topics. In every topic, you can start from questions according to your comfort level.

The [practice system](#) tells you exactly the test case where your code failed. In case you need more clarity about a question, you may use the expected output button to see output for your given input. You can also view successful submissions of others in case you are stuck. **To see solutions of others**, please click the “All Submissions” button at the bottom of the problem statement.

Topics

- [Mathematical](#)
- [Puzzles](#)
- [Arrays](#)
- [String](#)
- [Searching](#)
- [Sorting](#)
- [Hashing](#)
- [Matrix](#)
- [Recursion](#)
- [Divide & Conquer](#)
- [Linked List](#)
- [Queue](#)
- [Prefix Sum and Sliding Window](#)
- [Bit Magic](#)
- [Tree](#)
- [Binary Search Tree](#)
- [Heap](#)
- [Graph](#)
- [Greedy Algorithms](#)
- [Dynamic Programming](#)
- [Backtracking](#)
- [Trie](#)



- [Doubly and Circular Linked Lists](#)
- [Stack](#)
- [Misc Topics](#)
- [Important Links](#)

[Complete Interview Preparation](#)

Get fulfilled all your interview preparation and coding practice needs at a single place with the [Complete Interview Preparation Course](#) that provides you all the required stuff to prepare for any product-based, service-based, or start-up company at the most affordable prices. Practice 200+ coding interview questions with the help of this course and get yourself interview prepared for your dream company.

Mathematical

1. [Print the pattern](#) (You only need to write function here)
2. [Print table](#) (This is a full code problem. Please see sample codes [here](#) before attempting the problem)
3. [Series AP](#)
4. [Series GP](#)
5. [Closest Number](#)
6. [Armstrong Numbers](#)
7. [Sum of digits of a number](#)
8. [Reverse digits](#)
9. [Print the Kth Digit](#)
10. [Binary number to decimal number](#)
11. [Jumping Numbers](#)
12. [GCD of two numbers](#)
13. [LCM of two numbers](#)
14. [Add two fractions](#)
15. [GCD of array](#)
16. [Factorial of a number](#)
17. [Compute nPr](#)
18. [Compute nCr](#)
19. [Largest prime factor](#)
20. [Perfect Numbers](#)
21. [Pair cube count](#)

22. [Find Nth root of M](#)
23. [Prime Number](#)
24. [Sieve of Eratosthenes](#)
25. [Sum of all prime numbers between 1 and N.](#)
26. [Pairs of prime numbers](#)

Related Learning Resources : [Mathematical Algorithms](#) and [Number Theory](#)

Puzzles

1. [Count Squares](#)
2. [3 Divisors](#)
3. [Check if four points form a square](#)
4. [Check for power](#)
5. [Overlapping rectangles](#)
6. [Trailing zeroes in factorial](#)
7. [Angle between hour and minute hand](#)
8. [Number Of Open Doors](#)
9. [Triangular Numbers](#)
10. [Nth Even Fibonacci Number](#)
11. [Last two digit Fibonacci](#)
12. [Squares in a Matrix](#)
13. [Day of the week](#)

Related Learning Resources : [Puzzles](#)

Arrays

1. [Array operations \(Search, insert, delete\)](#)
2. [Array alternate printing](#)
3. [Maximum and minimum in an array](#)
4. [Second largest in array](#)
5. [Sum of array elements](#)
6. [Reverse an Array](#)
7. [Rotate Array](#)
8. [Count of smaller elements](#)
9. [Remove duplicate elements from sorted Array](#)

10. [Count possible triangles](#)
11. [Leaders in an array](#)
12. [Minimum distance between two numbers](#)
13. [Sorted subsequence of size 3](#)
14. [Maximum Sub Array](#)
15. [Majority Element](#)
16. [Wave Array](#)
17. [Maximum Index](#)
18. [Max sum path in two arrays](#)
19. [Product array puzzle](#)
20. [Find duplicates in a small ranged array](#)
21. [Find Missing And Repeating](#)
22. [Stock buy and sell](#)
23. [Trapping Rain Water](#)
24. [Pair with given sum in a sorted array](#)
25. [Chocolate Distribution Problem](#)
26. [Longest Consecutive subsequence](#)
27. [Three way partitioning](#)

Related Learning Resources : [Array Data Structure](#)

String

1. [Check for palindrome](#)
2. [Check for anagram](#)
3. [Anagram Palindrome](#)
4. [Title case conversion](#)
5. [Sort the string](#)
6. [Merge two strings](#)
7. [Save Ironman](#)
8. [Good or Bad string](#)
9. [URLify a given string](#)
10. [Extract Maximum](#)
11. [Reverse words in a given string](#)
12. [Implement strstr](#)
13. [Check for subsequence](#)
14. [Check for rotation](#)
15. [Check if two strings are k-anagrams](#)
16. [Uncommon characters](#)
17. [Anagram Search](#)
18. [First repeating character](#)

19. [First non-repeating character](#)
20. [Longest Distinct characters in string](#)
21. [Longest Palindromic Substring](#)
22. [Find k-th character in string](#)
23. [Smallest window in a string containing all characters of another string](#)
24. [Add Binary Strings](#)
25. [Multiply two Strings](#)
26. [Nearest multiple of 10](#)

Related Learning Resources : [String Data Structure](#)

Searching

1. [Linear Search](#)
2. [Facing the sun](#)
3. [Magnet Array Problem](#)
4. [Binary Search](#)
5. [Floor in a Sorted Array](#)
6. [Count occurrences in a sorted array](#)
7. [Search in a sorted and rotated](#)
8. [Find the missing number](#)
9. [Missing element of AP](#)
10. [Square root of a number](#)
11. [Find Transition Point in a Sorted Binary Array](#)
12. [Last index of One](#)
13. [Peak element](#)
14. [Allocate minimum number of pages](#)
15. [Common elements in three sorted](#)
16. [Smallest Positive missing number](#)

Related Learning Resources : [Searching Algorithms](#)

Sorting

1. [Check if array is sorted](#)
2. [Sort a binary array](#)
3. [Sort an array of 0s, 1s and 2s](#)
4. [Bubble Sort](#)
5. [Insertion Sort](#)
6. [Selection Sort](#)

7. [Quick Sort](#)
8. [Merge Sort](#)
9. [Sort an array when two halves are sorted](#)
10. [Relative Sorting](#)
11. [Triplet Sum in Array](#)
12. [Minimum Swaps to Sort](#)
13. [Sorting elements by frequency](#)
14. [Triplet Family](#)
15. [Count the triplets](#)

Related Learning Resources : [Sorting Algorithms](#)

Hashing

1. [Count distinct elements](#)
2. [Array Subset of another array](#)
3. [Nuts and Bolts Problem](#)
4. [Count frequencies of elements](#)
5. [Check if two arrays are equal or not](#)
6. [First element to occur k times](#)
7. [In First But Second](#)
8. [Non-Repeating Element](#)
9. [Group Anagrams Together](#)
10. [Winner of an election](#)
11. [Check for a pair with given sum](#)
12. [Count distinct pairs with difference k](#)
13. [Count pairs with given sum](#)
14. [Find all four sum numbers](#)
15. [A Simple Fraction](#)
16. [Largest Fibonacci Subsequence](#)

Related Learning Resources : [Hashing Data Structure](#)

Matrix

1. [Transpose of Matrix](#)
2. [Print Matrix in snake Pattern](#)
3. [Print a given matrix in spiral form](#)
4. [Is Sudoku Valid](#)
5. [Count zeros in a sorted matrix](#)

6. [Squares in a Matrix](#)
7. [A Boolean Matrix Question](#)
8. [Search in row-wise and column-wise sorted](#)
9. [Find the row with maximum number of 1s](#)
10. [Count pairs Sum in matrices](#)
11. [Median In a Row-Wise sorted Matrix](#)

Related Learning Resources : [Matrix Data Structure](#)

Recursion

1. [Print Pattern](#)
2. [Handshakes](#)
3. [Tower of Hanoi](#)
4. [Josephus problem](#)
5. [Recursively remove all adjacent duplicates](#)
6. [Possible words from Phone digits](#)
7. [Flood fill Algorithm](#)
8. [Permutations of a string](#)

Related Learning Resources : [Recursion](#)

Divide & Conquer

1. [Write your own power function](#)
2. [Program for n-th Fibonacci Number](#)
3. [K-th element of two sorted Arrays](#)
4. [Median of two sorted arrays](#)
5. [Karatsuba Algorithm](#)
6. [The Painter's Partition Problem](#)
7. [Convex Hull](#)
8. [Counting inversions](#)

Related Learning Resources : [Divide and Conquer Algorithms](#)

Linked List

1. [Print a Linked List](#)
2. [Length of a linked list](#)
3. [Node at a given index in linked list](#)

4. [Middle of a linked list](#)
5. [n-th node from end of a linked list](#)
6. [Delete a node](#)
7. [Remove every k'th node](#)
8. [Delete N nodes after M nodes of a linked list](#)
9. [Delete without head pointer](#)
10. [Rearrange a linked list](#)
11. [Segregate even and odd \(Using only one traversal\)](#)
12. [Reorder List](#)
13. [Polynomial Addition](#)
14. [Insert in a Sorted List](#)
15. [Swap nodes in pairs](#)
16. [Reverse a linked list](#)
17. [Reverse a Linked List in groups of given size.](#)
18. [Check for palindrome](#)
19. [Flattening a linked list](#)
20. [Get intersection point](#)
21. [Remove duplicates from sorted list](#)
22. [Remove duplicates from unsorted lists](#)
23. [Sort a linked list of 0s, 1s and 2s.](#)
24. [Circular Linked List](#)
25. [Detect loop in a linked list](#)
26. [Find length of Loop](#)
27. [Remove loop in a linked list](#)
28. [Add two numbers represented by linked lists](#)
29. [Clone a linked list with random pointers](#)
30. [Add 1 to a number represented as linked list](#)
31. [Add two numbers represented as linked list](#)
32. [Multiply two linked lists](#)
33. [Merge two sorted linked lists](#)
34. [Merge Sort on Linked List](#)
35. [Intersection of Two Linked Lists](#)
36. [Union of Two Linked Lists](#)

Related Learning Resources : [Linked List Data Structure](#)

Doubly and Circular Linked Lists

1. [Insert a node in Doubly linked list](#)
2. [Delete node in Doubly Linked List](#)
3. [Circular Linked List Traversal](#)

4. [Split a Circular Linked List into two halves](#)
5. [Insert in Sorted way in a Sorted DLL](#)
6. [QuickSort on Doubly Linked List](#)
7. [Merge Sort on Doubly Linked List](#)
8. [Rotate doubly Linked List by P nodes](#)
9. [XOR Linked List](#)

Related Learning Resources : [Doubly Linked List](#) and [Circular Linked List](#).

Stack

1. [Implement Stack using Array](#)
2. [Implement Stack using Linked List](#)
3. [Check for balanced parenthesis](#)
4. [Reverse a stack](#)
5. [Implement two stacks in an array](#)
6. [Design a stack with getMin](#)
7. [The celebrity problem](#)
8. [Stock Span Problem](#)
9. [Next Greater Element](#)
10. [Next Smaller Element](#)
11. [Longest valid Parentheses](#)

Related Learning Resources : [Stack Data Structure](#)

Queue and Dequeue

1. [Implement Queue using Linked List](#)
2. [Implement Queue using Array](#)
3. [Implement Stack using Queue](#)
4. [Implement Queue using Stack](#)
5. [Reversing a Queue](#)
6. [Circular tour](#)
7. [First non-repeating character in a stream](#)

Related Learning Resources : [Queue Data Structure](#)

Prefix Sum and Sliding Window

1. [Equilibrium Point](#)
2. [Check if there is a subarray with 0 sum](#)
3. [Longest Sub-Array with Sum K](#)
4. [Longest subarray with sum divisible by K](#)
5. [Largest subarray with equal 1s and 0s](#)
6. [Longest common span with same number of 1s and 0s among two arrays](#)
7. [Find maximum sum in any subarray of size k](#)
8. [Count distinct elements in every window of size k](#)
9. [Check for subarray with given sum](#)

Related Learning Resources : [Prefix Sum](#) and [Sliding Window](#)

Bit Magic

1. [Check if a number is even or odd.](#)
2. [Number of bit flips](#)
3. [Game of XOR](#)
4. [Find bit at a position](#)
5. [Swap odd and even bits](#)
6. [Power of 2](#)
7. [Odd occurring element](#)
8. [Missing number in array](#)
9. [Index Of an Extra Element](#)
10. [Reverse Bits](#)
11. [Count set bits](#)
12. [Power Set](#)

Related Learning Resources : [Bit Magic](#)

Tree

1. [Inorder Traversal](#)
2. [Preorder Traversal](#)
3. [Postorder Traversal](#)
4. [Level order traversal](#)
5. [Find height of Binary Tree](#)
6. [Count Leaves in Binary Tree](#)
7. [Check for Children Sum Property](#)
8. [Mirror Tree](#)
9. [Check for Balanced Tree](#)
10. [Lowest Common Ancestor in a Binary Tree](#)

11. [Diameter of Binary Tree](#)
12. [Left View of Binary Tree](#)
13. [Right View of Binary Tree](#)
14. [Maximum path sum](#)
15. [Level order traversal line by line](#)
16. [Tree from Postorder and Inorder](#)
17. [Tree from Preorder and Inorder](#)
18. [Connect Nodes at Same Level](#)
19. [Zig-Zag level order traversal](#)
20. [Serialize and Deserialize a Binary Tree](#)
21. [Leaves to DLL](#)
22. [Binary Tree to Doubly Linked List](#)
23. [Binary Tree to Circular Doubly Linked List](#)

Related Learning Resources : [Tree Data Structure](#)

Binary Search Tree

1. [BST Search](#)
2. [BST Insert](#)
3. [BST Delete](#)
4. [Minimum in BST](#)
5. [Inorder Traversal and BST](#)
6. [Count BST nodes that lie in a given range](#)
7. [Add all greater values](#)
8. [Predecessor and Successor in BST](#)
9. [Closest Neighbor in BST](#)
10. [Lowest Common Ancestor in a BST](#)
11. [Convert Level Order Traversal to BST](#)
12. [Normal BST to Balanced BST](#)
13. [Pair with given sum in BST](#)
14. [Check for BST](#)
15. [Correct BST with two nodes swapped](#)
16. [Median of BST](#)
17. [k-th smallest element in BST](#)
18. [Unique BST's](#)
19. [Array to BST](#)
20. [Preorder Traversal and BST](#)
21. [Preorder to Postorder](#)
22. [Leaf nodes from preorder traversal](#)
23. [Triplet with 0 sum in BST](#)

24. [Merge two BST 's](#)
25. [Largest BST Subtree](#)

Related Learning Resources : [Binary Search Tree](#)

Heap

1. [Binary Heap Operations](#)
2. [Height of Heap](#)
3. [Heap Sort](#)
4. [Sort a Nearly Sorted Array](#)
5. [K Largest Elements](#)
6. [K-th largest element in a stream](#)
7. [Median of stream](#)
8. [Merge k sorted arrays](#)

Related Learning Resources : [Heap Data Structure](#)

Graph

1. [Print adjacency list](#)
2. [Breadth First Search](#)
3. [Depth First Search](#)
4. [Find whether path exist](#)
5. [Knight Walk](#)
6. [Snake and Ladder Problem](#)
7. [Bipartite Graph](#)
8. [Detect Cycle in an undirected graph](#)
9. [Detect Cycle in a directed graph](#)
10. [Find first n numbers with given set of digits](#)
11. [Rotten oranges](#)
12. [Topological sort](#)
13. [Shortest Source to Destination Path](#)
14. [Transitive closure of a Graph](#)
15. [Strongly Connected Components](#)

Related Learning Resources : [Graph Data Structure](#)

Greedy Algorithms

1. [Fractional Knapsack](#)
2. [Largest number with given sum](#)
3. [Activity Selection](#)
4. [N meetings in one room](#)
5. [Minimum Platforms](#)
6. [Minimum number of Coins](#)
7. [Job Sequencing Problem](#)
8. [Minimize the heights](#)
9. [Huffman Coding](#)
10. [Huffman Decoding](#)
11. [Minimum Spanning Tree](#)
12. [Dijkstra for Adjacency Matrix](#)

Related Learning Resources : [Greedy Algorithms](#)

Dynamic Programming

1. [Print first n Fibonacci Numbers.](#)
2. [Count ways to reach the n'th stair](#)
3. [Cutted Segments](#)
4. [Kadane's Algorithm](#)
5. [Stickler Thief](#)
6. [Minimum number of jumps](#)
7. [Total Decoding Messages](#)
8. [Min Cost Path](#)
9. [Coin Change](#)
10. [Longest Common Subsequence](#)
11. [Consecutive 1's not allowed](#)
12. [Edit Distance](#)
13. [Rod Cutting](#)
14. [Water Overflow](#)
15. [Maximum Tip Calculator](#)
16. [Longest Increasing Subsequence](#)
17. [Maximum sum increasing subsequence](#)
18. [Max length chain](#)
19. [0 – 1 Knapsack Problem](#)
20. [Interleaved string](#)
21. [Longest Palindromic Subsequence](#)
22. [Wildcard Pattern Matching](#)
23. [Box Stacking](#)
24. [Longest Bitonic subsequence](#)

25. [Minimum sum partition](#)
26. [Largest square formed in a matrix](#)
27. [Word Break](#)
28. [Matrix Chain Multiplication](#)
29. [Special Keyboard](#)
30. [Egg Dropping Puzzle](#)
31. [Optimal Strategy for a Game](#)

Related Learning Resources : [Dynamic Programming](#)

Backtracking

1. [Rat Maze With Multiple Jumps](#)
2. [Coins and Game](#)
3. [Hamiltonian Path](#)
4. [Solve the Sudoku](#)
5. [Combination Sum – Part 2](#)
6. [Combination Sum](#)
7. [Subsets](#)
8. [Largest number in K swaps](#)
9. [M-Coloring Problem](#)
10. [Black and White](#)

Related Learning Resources : [Backtracking](#)

Trie

1. [Trie Search and Insert](#)
2. [Trie Delete](#)
3. [Unique rows in a binary matrix](#)
4. [Count of distinct substrings](#)
5. [Word Boggle](#)

Related Learning Resources : [Trie Data Structure](#)

Misc Questions to test your overall learning

1. [Longest common prefix](#)
2. [Implement Atoi](#)
3. [Two numbers with sum closest to zero](#)

4. [Smallest greater elements in whole array](#)
5. [Max rectangle](#)
6. [Find triplets with zero sum](#)
7. [Counting elements in two arrays](#)
8. [Merge K sorted linked lists](#)
9. [Maximum Difference](#)
10. [Circle of strings](#)
11. [All possible Word Breaks](#)
12. [Alien Dictionary](#)
13. [Design a tiny URL or URL shortener](#)
14. [Implement LRU Cache](#)

Important Links

1. [Sudo Placement](#) : For companies like Amazon, Microsoft, Adobe, .., etc
2. [Sudo Placement 2](#) : For companies like TCS, Infosys, Wipro, Cognizant, .. etc
3. Aptitude questions asked in round 1 : [Placements Course](#) designed for this purpose.
4. MCQs asked from different computer science subjects : [Subject-Wise Quizzes](#)
5. Interview theory and coding questions of all companies : [Company wise all practice questions](#).
6. Interview experiences of all companies : [Interview corner](#).
7. [Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe, ...](#)
8. [Must Do Coding Questions Company-wise](#)

Don't forget to check out the courses mentioned below:


**Complete
Interview
Preparation**



• Online Course

Sign Up For Free

**Competitive
Programming**
Live



• Online Course

Sign Up For Free