**Company-wise DSA Qs List ByHimanshu Gupta**

| **Company Name** | **DSA Problem Link** | **Tricks for Solving Patterns** |
| --- | --- | --- |
| **Google** | [**Google DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on backtracking and dynamic programming.** |
| **Microsoft** | [**Microsoft DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Practice binary search, greedy algorithms.** |
| **Amazon** | [**Amazon DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on sliding window, heap, and DFS/BFS.** |
| **Facebook** | [**Facebook DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Master recursion, graphs, and bit manipulation.** |
| **Apple** | [**Apple DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Prioritize dynamic programming and divide and conquer.** |
| **Adobe** | [**Adobe DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Practice sorting and searching techniques.** |
| **Goldman Sachs** | [**Goldman Sachs DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on arrays, hashing, and string manipulation.** |
| **Uber** | [**Uber DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on graphs, BFS/DFS, and Dijkstra's algorithm.** |
| **LinkedIn** | [**LinkedIn DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on two-pointer and sliding window techniques.** |
| **Netflix** | [**Netflix DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Prioritize dynamic programming and DP on trees.** |
| **Twitter** | [**Twitter DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on greedy algorithms and interval problems.** |
| **Dropbox** | [**Dropbox DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Master recursion and binary search on sorted arrays.** |
| **Airbnb** | [**Airbnb DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Practice graph traversal and backtracking.** |
| **Salesforce** | [**Salesforce DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on greedy algorithms and dynamic programming.** |
| **Oracle** | [**Oracle DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on matrix traversal and dynamic programming.** |
| **PayPal** | [**PayPal DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Prioritize binary search and sorting problems.** |
| **Walmart** | [**Walmart DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on hashing, prefix sums, and arrays.** |
| **Expedia** | [**Expedia DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Practice stack-based problems and recursion.** |
| **Snap** | [**Snap DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on graphs and dynamic programming.** |
| **Yahoo** | [**Yahoo DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on linked lists and recursion.** |
| **DoorDash** | [**DoorDash DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Master binary trees and backtracking.** |
| **Stripe** | [**Stripe DSA Problems**](about:blank) | **Practice greedy algorithms and string manipulation.** |
| **Lyft** | [**Lyft DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on two-pointer and sliding window techniques.** |
| **Intuit** | [**Intuit DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on backtracking and dynamic programming.** |
| **IBM** | [**IBM DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Master dynamic programming and recursion problems.** |
| **Atlassian** | [**Atlassian DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on graph traversal and dynamic programming.** |
| **Reddit** | [**Reddit DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on hashing and bit manipulation.** |
| **Pinterest** | [**Pinterest DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Master recursion and divide and conquer techniques.** |
| **Spotify** | [**Spotify DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on sorting, searching, and heaps.** |
| **Bloomberg** | [**Bloomberg DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on arrays, dynamic programming, and graphs.** |
| **Cisco** | [**Cisco DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Focus on linked lists and dynamic programming.** |
| **ByteDance** | [**ByteDance DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Master sorting algorithms and binary search.** |
| **Tesla** | [**Tesla DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Work on graph traversal and dynamic programming.** |
| **TikTok** | [**TikTok DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Prioritize dynamic programming and recursion.** |
| **Nvidia** | [**Nvidia DSA Problems**](https://drive.google.com/drive/u/1/folders/1zCjvu1WLqGoM5VrnGRMFpyNsoA7sofzM) | **Practice bit manipulation and backtracking.** |

**50 tricks to identify DSA Patterns Link and other guides and cheat sheets required for DSA:**

[**https://drive.google.com/drive/folders/1GbYapJnWJZtFlf2mSC1HqvgYLLL\_BATJ?usp=drive\_link**](https://drive.google.com/drive/folders/1GbYapJnWJZtFlf2mSC1HqvgYLLL_BATJ?usp=drive_link)

**🛤️ 3-Month Logic-Building & Problem-Solving Roadmap by *HIMANSHU GUPTA (codeprime.io)***

| **Week** | **Focus Area** | **Daily Breakdown** | **Checkpoints** |
| --- | --- | --- | --- |
| **Week 1** | **Foundation: Basics of Programming** | **Day 1-2:** Learn variables, data types, loops (for, while). **Day 3-4:** Conditionals, functions. **Day 5-6:** Arrays & Strings basics. **Day 7:** Solve 10 simple pattern-building questions (stars, triangles). | ✅ Understand basic syntax and concepts. ✅ Solve **10 pattern-building problems**. |
| **Week 2** | **Introduction to Problem Solving** | **Day 1-2:** Learn dry-run and pseudocode. **Day 3-5:** Solve **10 simple DSA problems** (easy level, e.g., reverse array, Fibonacci). **Day 6-7:** Build **Mini Project 1: Calculator App**. | ✅ Solve 10 DSA problems. ✅ Complete **Mini Project 1: Calculator App**. |
| **Week 3** | **Level Up: Arrays & Logic Thinking** | **Day 1-2:** Advanced array techniques (sliding window, prefix sum). **Day 3-6:** Solve **15 array-based problems** (medium level). **Day 7:** Build **Mini Project 2: Tic-Tac-Toe Game**. | ✅ Solve 25 total DSA problems (15 new). ✅ Complete **Mini Project 2: Tic-Tac-Toe Game**. |
| **Week 4** | **Strings & Problem-Solving Practice** | **Day 1-2:** String manipulation (reversal, palindromes, substrings). **Day 3-5:** Solve **15 string-based problems** (easy-medium). **Day 6-7:** Build **Mini Project 3: Text Manipulation Tool (e.g., Uppercase, Reverse)**. | ✅ Solve 40 total DSA problems (15 new). ✅ Complete **Mini Project 3: Text Tool**. |
| **Week 5** | **Recursion & Logic Expansion** | **Day 1-2:** Learn recursion basics (factorial, Fibonacci). **Day 3-5:** Solve **10 recursion-based problems** (medium level). **Day 6-7:** Build **Mini Project 4: Recursive Maze Solver**. | ✅ Solve 50 total DSA problems (10 new). ✅ Complete **Mini Project 4: Recursive Maze Solver**. |
| **Week 6** | **Data Structures: Stacks & Queues** | **Day 1-2:** Learn stacks and queues basics. **Day 3-6:** Solve **15 problems** (balanced parentheses, queue reversal). **Day 7:** Work on **Mini Project 5: Browser History Tracker (using stack)**. | ✅ Solve 65 total DSA problems (15 new). ✅ Complete **Mini Project 5: Browser History Tracker**. |
| **Week 7** | **Dynamic Programming Introduction** | **Day 1-3:** Learn DP basics (knapsack, Fibonacci with memoization). **Day 4-6:** Solve **10 DP problems** (easy-medium). **Day 7:** Revise all past concepts/projects. | ✅ Solve 75 total DSA problems (10 new). ✅ Master basic DP problems. |
| **Week 8** | **Advanced Problem Solving (Graph)** | **Day 1-2:** Learn graph representation (adjacency list/matrix). **Day 3-5:** Solve **10 graph-based problems** (BFS/DFS). **Day 6-7:** Build a **Mini Project: Path Finder Visualizer**. | ✅ Solve 85 total DSA problems (10 new). ✅ Complete Graph-Based Mini Project. |
| **Week 9** | **Advanced DSA (Sorting/Greedy)** | **Day 1-2:** Learn sorting algorithms (merge sort, quicksort). **Day 3-5:** Solve **15 problems** on sorting/greedy (e.g., activity selection, job scheduling). | ✅ Solve 100 DSA problems (15 new). |
| **Week 10-11** | **Integration: Complex Projects** | **Day 1-7:** Brainstorm and build **Project 1: Expense Tracker with Charts**. **Next 7 days:** Build **Project 2: Multiplayer Rock-Paper-Scissors Game** (WebSockets, if possible). | ✅ Integrate past knowledge into real-world projects. ✅ Master real-world application-building. |
| **Week 12** | **Final Touches and Mock Practice** | **Day 1-3:** Revise all concepts learned. **Day 4-6:** Solve 10 problems from past mistakes or blindspots. **Day 7:** Build a final project of your choice integrating at least one complex DSA concept. | ✅ Complete final project. ✅ Feel confident in solving beginner-intermediate DSA problems. |

**💡 How to Think of Logic**

1. **Break Down the Problem**: Read the question twice. Identify inputs, outputs, and constraints.
2. **Start with Examples**: Create test cases manually and simulate the solution step-by-step.
3. **Write Pseudocode**: Draft a high-level plan before coding.
4. **Ask “Why?” at Every Step**: Understand each operation; don’t memorize solutions.
5. **Visualize**: Use diagrams or dry-run tables to debug.
6. **Optimize Gradually**: Start with brute force; iterate to optimize for efficiency.

**📚 100 DSA Questions for Logic Building**

| **Category** | **Question Name** | **Platform** | **Link** |
| --- | --- | --- | --- |
| **Basics & Warm-Up** | Print a pattern of stars (triangle, pyramid) | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/right-angle-triangle-pattern/0) |
|  | Reverse a number | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/reverse-digit0316/1) |
|  | Check if a number is palindrome | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/palindrome0746/1) |
|  | Count digits in a number | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/count-digits5716/1) |
|  | Find factorial of a number | HackerRank | [Link](https://www.hackerrank.com/challenges/extra-long-factorials/problem) |
| **Arrays** | Reverse an array | LeetCode | [Link](https://leetcode.com/problems/reverse-string/) |
|  | Find the maximum and minimum of an array | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/find-minimum-and-maximum-element-in-an-array4428/1) |
|  | Rotate an array by K steps | LeetCode | [Link](https://leetcode.com/problems/rotate-array/) |
|  | Move all zeroes to the end | LeetCode | [Link](https://leetcode.com/problems/move-zeroes/) |
|  | Kadane's Algorithm (Maximum Subarray Sum) | LeetCode | [Link](https://leetcode.com/problems/maximum-subarray/) |
| **Strings** | Reverse a string | LeetCode | [Link](https://leetcode.com/problems/reverse-string/) |
|  | Check if two strings are anagrams | LeetCode | [Link](https://leetcode.com/problems/valid-anagram/) |
|  | Longest Common Prefix | LeetCode | [Link](https://leetcode.com/problems/longest-common-prefix/) |
|  | Check if a string is a palindrome | LeetCode | [Link](https://leetcode.com/problems/valid-palindrome/) |
|  | Count and say | LeetCode | [Link](https://leetcode.com/problems/count-and-say/) |
| **Recursion** | Fibonacci series using recursion | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/print-first-n-fibonacci-numbers1002/1) |
|  | Tower of Hanoi | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/tower-of-hanoi-1587115621/1) |
|  | Factorial using recursion | LeetCode | [Link](https://leetcode.com/problems/factorial-trailing-zeroes/) |
|  | Reverse a linked list using recursion | LeetCode | [Link](https://leetcode.com/problems/reverse-linked-list/) |
|  | Permutations of a string | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-string/0) |
| **Sorting** | Bubble sort | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/bubble-sort/1) |
|  | Selection sort | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/selection-sort/1) |
|  | Merge sort | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/merge-sort/1) |
|  | Quick sort | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/quick-sort/1) |
|  | Insertion sort | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/insertion-sort/1) |
| **Searching** | Binary search | LeetCode | [Link](https://leetcode.com/problems/binary-search/) |
|  | Linear search | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/search-an-element-in-an-array/0) |
|  | Search in a rotated sorted array | LeetCode | [Link](https://leetcode.com/problems/search-in-rotated-sorted-array/) |
|  | First and last position in a sorted array | LeetCode | [Link](https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/) |
|  | Square root of a number (using binary search) | LeetCode | [Link](https://leetcode.com/problems/sqrtx/) |
| **Linked List** | Reverse a linked list | LeetCode | [Link](https://leetcode.com/problems/reverse-linked-list/) |
|  | Detect a cycle in a linked list | LeetCode | [Link](https://leetcode.com/problems/linked-list-cycle/) |
|  | Merge two sorted linked lists | LeetCode | [Link](https://leetcode.com/problems/merge-two-sorted-lists/) |
|  | Remove Nth node from the end | LeetCode | [Link](https://leetcode.com/problems/remove-nth-node-from-end-of-list/) |
|  | Find the middle of a linked list | LeetCode | [Link](https://leetcode.com/problems/middle-of-the-linked-list/) |
| **Stacks & Queues** | Implement a stack using arrays | LeetCode | [Link](https://leetcode.com/problems/implement-stack-using-queues/) |
|  | Evaluate postfix expression | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/evaluation-of-postfix-expression/0) |
|  | Balanced parentheses | LeetCode | [Link](https://leetcode.com/problems/valid-parentheses/) |
|  | Next greater element | LeetCode | [Link](https://leetcode.com/problems/next-greater-element-i/) |
|  | Implement a queue using stacks | LeetCode | [Link](https://leetcode.com/problems/implement-queue-using-stacks/) |
| **Dynamic Programming** | 0/1 Knapsack problem | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/0-1-knapsack-problem/0) |
|  | Fibonacci using dynamic programming | LeetCode | [Link](https://leetcode.com/problems/fibonacci-number/) |
|  | Longest common subsequence | LeetCode | [Link](https://leetcode.com/problems/longest-common-subsequence/) |
|  | Longest increasing subsequence | LeetCode | [Link](https://leetcode.com/problems/longest-increasing-subsequence/) |
|  | Minimum steps to reach the end | LeetCode | [Link](https://leetcode.com/problems/jump-game-ii/) |
| **Graphs** | BFS traversal | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/bfs-traversal-of-graph/1) |
|  | DFS traversal | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/depth-first-traversal-for-a-graph/1) |
|  | Detect cycle in an undirected graph | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/detect-cycle-in-an-undirected-graph/1) |
|  | Shortest path in a graph (Dijkstra’s) | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/implementing-dijkstra-set-1-adjacency-matrix/1) |
|  | Topological sort | GeeksforGeeks | [Link](https://practice.geeksforgeeks.org/problems/topological-sort/1) |

***Follow me on Insta @***[***codeprime.io***](https://www.instagram.com/codeprime.io/)

***Subscribe to my YouTube channel @***[***itshimanshu2001***](https://www.youtube.com/@itshimanshu2001)