Complete Algorithm Preparation List for Google SDE-3

# I. Core Algorithms (Fundamentals You Must Know Cold)

Sorting: Merge Sort (recursive + iterative), Quick Sort, Counting Sort, Radix Sort, Heap Sort, Bucket Sort, Cyclic Sort

Searching: Binary Search (recursive + with boundaries), Ternary Search (unimodal functions)

Hashing: HashMap, HashSet, Custom Hash Functions, Rolling Hash

Greedy Algorithms: Activity Selection, Job Sequencing, Fractional Knapsack, Huffman Coding

Bit Manipulation: Bitmasking, Set bits, XOR tricks

Basic Recursion: Tower of Hanoi, Permutations, Subsets, Recursive backtracking

# II. Dynamic Programming (DP)

1D DP: Fibonacci, Climbing stairs, LIS (in-depth), Max subarray

2D DP: Grid unique paths, LCS, Edit Distance, Matrix Chain Multiplication

Knapsack: 0/1 Knapsack, Unbounded, Bounded, Multidimensional

Bitmask DP: TSP (Traveling Salesman), Subset DP

Tree DP: DP on post-order, Rerooting

Memoization: With state tuples or bitmask

DP Optimization: Sliding window, Monotonic queue, Divide and Conquer optimization

# III. Tree Algorithms

Traversals: Preorder, Inorder, Postorder, Level-order (BFS), Morris Traversal

LCA (Lowest Common Ancestor): Binary Lifting, Tarjan’s Offline Algorithm

Tree Types / Properties: Height-balanced, Complete Binary Tree, Perfect BT, Full BT

AVL Tree: Rotations, Balance Factor

Red-Black Tree: Properties, Insertion/Deletion logic

B / B+ Trees: Indexing, Leaf-level data structure, Disk optimization

Diameter & Path Sum: DFS aggregation

BST: Inorder properties, Range queries, Validation

Segment Tree: Range Query + Update, Lazy Propagation

Fenwick Tree (BIT): Prefix Sum, Point Update, Range Query

Merge Sort Tree / Wavelet Tree

Persistent Segment Tree

Dynamic K-th Order Queries

Heavy-Light Decomposition

Rerooting DP

# IV. Graph Algorithms

Traversals: BFS, DFS (recursive + iterative), 0-1 BFS

Shortest Paths: Dijkstra, A\* Search, Bellman-Ford, Floyd-Warshall

Minimum Spanning Tree: Kruskal’s, Prim’s (Min Heap + Adjacency List)

Topological Sort: DFS-based, Kahn’s Algorithm, Conceptual prerequisites

Union Find: Path compression + union by rank, Cycle detection

Strong Components: Kosaraju’s, Tarjan’s

Biconnected Components: Tarjan’s Articulation Point algorithm

Graph Properties: Hamiltonian Path, Euler Circuit, Bipartite Check

Multistage Graphs: Shortest path using layered DP

Branch and Bound: Used for TSP, N-Queens, Job Scheduling

Flow Algorithms (Max Flow, Edmonds-Karp)

# V. Advanced Techniques & Patterns

Trie: Prefix matching, Bitwise XOR, Autocomplete

Monotonic Stack/Queue: NGE, Histogram, Sliding window max/min

Sliding Window: Longest substring, Max sum subarray

Two Pointers: Sorted array operations, Merge-like problems

Meet-in-the-middle: Subset sum with optimization

Rolling Hash: Rabin-Karp, Plagiarism detection

Reservoir Sampling: Random element from stream

Heap: Heapify, Top-K, Min/Max Heap operations

Recurrence Relation Analysis: Master’s Theorem

Strassen’s Matrix Multiplication: Divide and conquer for O(n^2.81) multiplication

Row/Col Major Order: Array mapping in memory (2D to 1D index)

# VI. String Matching & Pattern Algorithms

KMP Algorithm: Pattern finding in O(n + m)

Rabin-Karp: Rolling hash-based match

Z-Algorithm: String search, periodic substrings

Aho-Corasick: Multi-pattern matching using Trie + BFS

Suffix Arrays / Trees

# VII. System Design-Relevant Algorithmic Tools

Consistent Hashing: Sharded distributed systems

LRU Cache: LinkedHashMap / Doubly Linked List + HashMap

Token Bucket / Leaky Bucket: Rate limiting

Message Queues: Pub-sub systems

Observer / Strategy Patterns: Clean extensible design

Bloom Filter: Approximate membership testing

# VIII. Time & Space Complexity Analysis

Big O (O): Upper bound runtime

Big Ω (Omega): Lower bound runtime

Big Θ (Theta): Tight bound

Amortized Analysis: Stack resizing, Union-Find

Worst/Avg/Best Case: For sort/search/tree balancing

Space Complexity: Recursion stack, dynamic memory, auxiliary arrays

For the above complete categorized algorithm preparation list (keep the order same for everything), for every section and for every category and for every concept/usage create an in depth information on them. Tell me what's that section, what's that category, what's that concept (explain this in detail the info should be enough so that I wont have to look on the internet to understand), their usage, whats the algorithm, the codes required for every concept and their every approach (like codes for both recursive & iterative), their pros and cons, their applications, the time and space BigO (also include theta & omega), the diagrams and charts for visualization (generate it or take it from internet) and understanding. This should contain EVERYTHING there is on that topic, so that before interview I just only refer this and can go ahead. Remove system Design section for now.

Give me the full document so that I can print it

For the above topics, for every category and for every concept/usage create an in depth information on them. Tell me what's that section, what's that category, what's that concept (explain this in detail the info should be enough so that I wont have to look on the internet to understand), their usage, whats the algorithm, the codes required for every concept and their every approach (like codes for both recursive & iterative), their pros and cons, their applications, the time and space BigO (also include theta & omega). This should contain EVERYTHING there is on that topic, so that before interview I just only refer this and can go ahead.