Data Structures & Algorithms — Ultra Detailed Guide

Arrays

Arrays store elements in contiguous memory. They allow O(1) access by index but cost O(n) for inserts in the middle. Used in dynamic programming, buffers, and fixed collections.

Vectors (Dynamic Arrays)

Vectors resize automatically and support O(1) amortized insertion at the end. They allow random access but suffer O(n) insertion in the middle. Common in C++ STL and Java ArrayList.

Linked Lists

Linked lists consist of nodes linked by pointers. They allow O(1) insertion when the pointer is known but lack random access. Types include singly, doubly, and circular linked lists.

Stacks

Stacks operate on LIFO order. Common uses include function call stacks, expression evaluation, and backtracking.

Queues & Priority Queues

Queues operate on FIFO order. Priority queues use heaps to always extract the highest-priority element in O(log n).

Trees

Trees represent hierarchical structures. Variants include BST, AVL, Red-Black, Segment Trees, and Fenwick Trees.

Graphs

Graphs represent relationships using nodes and edges. They are used in networks, transport systems, and social platforms.

Trie

A prefix tree used for fast string search, autocomplete, and dictionary structures.

Algorithms — Each Explained Individually

Sorting Algorithms

Bubble Sort

Compares adjacent elements and swaps them if out of order. • Time: $O(n^2)$ • Space: $O(1)$ • Notes: Educational, rarely used in real systems.

Selection Sort

Repeatedly finds the minimum element and places it at the front. • Time: $O(n^2)$ • Space: $O(1)$ • Notes: Stable? No. Good for minimal swaps.

Insertion Sort

Builds a sorted array one element at a time. • Time: $O(n^2)$, but $O(n)$ on nearly sorted arrays • Notes: Used in real-world hybrid sorting algorithms (TimSort).

Merge Sort

Divide-and-conquer algorithm that splits, sorts, and merges. • Time: $O(n \log n)$ • Space: $O(n)$ • Notes: Stable, excellent for linked lists.

Quick Sort

Partition-based divide-and-conquer sorting. • Avg: $O(n \log n)$, Worst: $O(n^2)$ • Notes: Fastest practical sort, used in many standard libraries.

Heap Sort

Uses a binary heap to repeatedly extract the max/min. • Time: $O(n \log n)$ • Notes: In-place, but not stable.

Counting Sort

Sorts numbers by counting frequency. • Time: $O(n + k)$ • Notes: Only works for limited ranges.

Radix Sort

Sorts numbers digit-by-digit using counting sort as a subroutine. • Time: $O((n + k) * d)$ • Notes: Great for integers and strings.

Searching Algorithms

Linear Search

Scans each element sequentially. • Time: $O(n)$ • Use case: Unsorted collections.

Binary Search

Divides the search space by half each time. • Time: $O(\log n)$ • Requirement: Sorted arrays.

Graph Algorithms

BFS (Breadth-First Search)

Explores graph layer by layer. • Time: $O(V + E)$ • Uses: Shortest path in unweighted graphs, level-order traversal.

DFS (Deep-First Search)

Explores deep into a branch before backtracking. • Time: $O(V + E)$ • Uses: Cycle detection, connected components, topological sorting.

Dijkstra’s Algorithm

Finds shortest path with non-negative weights. • Time: $O((V + E) \log V)$ using a priority queue • Uses: Maps, routing, networks.

Bellman-Ford

Handles negative edges and detects negative cycles. • Time: $O(V \times E)$ • Uses: Currency arbitrage detection.

Floyd–Warshall

Computes all-pairs shortest paths using DP. • Time: $O(V^3)$ • Uses: Dense graphs, routing tables.

Kruskal's MST Algorithm

Builds a minimum spanning tree by always choosing the smallest edge. • Time: $O(E \log E)$ • Needs: Disjoint Set Union (Union-Find).

Prim’s MST Algorithm

Builds a spanning tree starting from any node using a priority queue. • Time: $O(E \log V)$

Dynamic Programming (DP)

Knapsack DP

Solves 0/1 knapsack by transforming the problem into subproblems. • Time: $O(n \times W)$ • Idea: maximize value without exceeding weight.

LCS (Longest Common Subsequence)

Finds longest sequence appearing in both strings. • Time: $O(n \times m)$ • Uses: diff tools, DNA analysis.

LIS (Longest Increasing Subsequence)

Finds longest strictly increasing sequence. • Time: $O(n \log n)$ using binary search.

Matrix Chain Multiplication

Finds best order to multiply matrices. • Time: $O(n^3)$ • Concept: DP to minimize operations.

Greedy Algorithms

Activity Selection

Selects maximum non-overlapping activities. • Always pick earliest finishing activity.

Huffman Coding

Builds optimal prefix-free codes for compression. • Used in ZIP, PNG, GZIP.

Fractional Knapsack

Greedy version of knapsack allowing fractions. • Time: $O(n \log n)$.

Divide & Conquer

Binary Search

Classic example: divide problem by half each step.

Merge Sort

Splits -> sorts -> merges.

Quick Sort

Partitions -> sorts partitions recursively.

Backtracking

N-Queens

Places queens so they don't attack each other. • Uses recursion + backtracking.

Permutations Generation

Generates all permutations of a list.

Subset Generation

Generates all subsets (power set).

String Algorithms

KMP (Knuth–Morris–Pratt)

Efficient pattern matching using LPS table. • Time: O(n + m).

Rabin–Karp

Uses rolling hash for fast substring search. • Great for multiple pattern search.

Z-Algorithm

Computes Z-array for linear-time pattern matching.

Trie

Used for prefix queries and dictionary storage.

Bit Manipulation

Bit Masks

Represent subsets or flags compactly using bits.

Brian Kernighan’s Algorithm

Counts set bits efficiently by turning off each lowest set bit.

Two’s Complement

Represents negative integers in binary systems.

End of Ultra-Detailed Guide.