DSA

(Data structure and Algorithm)

Short Notes based on ***<u>Exit Exam</u>***

Zemzem labs

1. Introduction

Algorithm

A step-by-step procedure to solve a problem.

Characteristics of Good Algorithm

- Correctness
- Efficiency
- Finiteness
- Definiteness
- Output

Asymptotic Notation

Used to describe algorithm complexity:

- **Big-O:** upper bound (worst case)
- **Ω:** lower bound (best case)
- **Θ:** tight bound (average case)

2. Time & Space Complexity

Common Time Complexities

- O(1) – constant
- O(log n) – logarithmic
- O(n) – linear
- O(n log n) – divide & conquer
- $O(n^2)$ – quadratic
- $O(2^n)$ – exponential
- O(n!) – factorial (TSP)

Space Complexity

Includes:

- Auxiliary space
- Total memory used

3. Arrays

Properties

- Fixed-size
- Contiguous memory
- Random access (O(1))

Operations

- Access: O(1)
- Insert: O(n)
- Delete: O(n)

Multi-dimensional arrays

Used for matrices, DP tables.

4. Linked Lists

Types

- Singly linked list
- Doubly linked list
- Circular linked list

Operations

- Insertion: O(1) at head, O(n) at position
- Deletion: O(1) if pointer known
- Traversal: O(n)

Compared to arrays:

- No random access
- Efficient insert/delete

5. Stacks

LIFO Structure

Operations:

- push
- pop
- peek

Applications:

- Expression evaluation
- Function call stack
- Balanced parentheses
- DFS

6. Queues

FIFO Structure

Types:

- Simple queue
- Circular queue
- Priority queue
- Deque

Applications:

- Scheduling
- Buffering
- BFS

7. Trees

Terminology

- Root
- Parent/child
- Leaf
- Depth/Height
- Subtree

Binary Tree Traversals

- Inorder (LNR)
- Preorder (NLR)
- Postorder (LRN)
- Level order (BFS)

8. Binary Search Tree (BST)

Properties

- Left subtree < root
- Right subtree > root

Complexity

- Search: O(h)
- Insert: O(h)
- Delete: O(h)
  Where h = height
  Worst case h = n (unbalanced tree)

9. Balanced Trees

AVL Tree

- Height-balanced
- Balance factor = -1, 0, 1
- Rotations: LL, RR, LR, RL

Red-Black Tree

- At most 2x height of balanced tree
- Faster insert/delete than AVL
- Used in maps, sets

10. Heaps

Types

- Min Heap

- Max Heap

Properties

- Complete binary tree
- Root contains min or max element

Heap Operations

- Insert → $O(\log n)$
- Delete/Extract → $O(\log n)$
- Build heap → $O(n)$

Used in Priority Queue and Heap Sort.

11. Hashing

Hash Table

Stores key-value pairs.

Collision Handling

- Chaining (linked lists)
- Open addressing (linear probing, quadratic, double hashing)

Good hash function

- Uniform distribution
- Minimizes collisions

12. Graphs

Representation

- Adjacency matrix ($O(n^2)$)
- Adjacency list ($O(V + E)$) — preferred

Traversal

- **BFS** (queue)
- **DFS** (stack/recursion)

Graph Types

- Directed, Undirected
- Weighted, Unweighted
- Cyclic, Acyclic (DAG)

13. Shortest Path Algorithms

Dijkstra

- Non-negative weights
- Time: O((V+E) log V)

Bellman-Ford

- Handles negative weights
- Detects negative cycles

Floyd–Warshall

- All-pairs shortest path
- Time: $O(n^3)$

14. Minimum Spanning Tree (MST)

Kruskal's Algorithm

- Greedy
- Uses Sorting + Disjoint Set (Union-Find)
- Best for sparse graphs

Prim's Algorithm

- Greedy
- Grows MST from a starting node
- Best for dense graphs

15. Sorting Algorithms

Comparison-based Sorting

Algorithm	Best	Average	Worst

Bubble	O(n)	O(n^2)	O(n^2)
Insertion	O(n)	O(n^2)	O(n^2)
Selection	O(n^2)	O(n^2)	O(n^2)
Merge	O(n log n)	O(n log n)	O(n log n)
Quick	O(n log n)	O(n log n)	O(n^2)
Heap	O(n log n)	O(n log n)	O(n log n)

Linear-Time Sorting

- Counting sort
- Radix sort
- Bucket sort

16. Divide and Conquer

Used in:

- Merge Sort
- Quick Sort
- Binary Search
- Closest pair of points
- FFT

17. Dynamic Programming (DP)

Characteristics

- Overlapping subproblems
- Optimal substructure

Examples

- Fibonacci
- Knapsack
- Longest Common Subsequence
- Matrix Chain Multiplication
- Coin change

18. Greedy Algorithms

Global optimal decisions built from local optimal choices.

Examples

- Activity Selection
- Huffman Coding
- MST (Kruskal, Prim)
- Dijkstra

⭐ Ultra-Condensed Exam Summary

- Big-O for complexity → worst-case measure
- Trees: BST, AVL, Red-Black
- Sorting: Merge/Heap = guaranteed O(n log n)
- Graphs: BFS (shortest in unweighted), DFS (cycle detection)
- Dijkstra no negatives; Bellman-Ford handles negatives
- DP = optimal substructure + overlapping subproblems
- Greedy = local optimal → global optimal
- Hashing = collisions handled via chaining or probing