DSA (Data Structures and Algorithms) Notes

1. Introduction to DSA

Data Structures and Algorithms (DSA) is a core area of computer science. Data Structures deal with the organization of data in memory, while Algorithms define a step-by-step procedure to solve a problem.

Importance of DSA:

- Helps in writing efficient code.

- Improves problem-solving skills.

- Essential for technical interviews.

DSA (Data Structures and Algorithms) Notes

2. Arrays

Definition: A collection of elements stored at contiguous memory locations.

Types: 1D Arrays, 2D Arrays, Multi-dimensional Arrays

Common Operations:

- Traversal: Visiting each element once.
- Insertion: Adding an element at a specific position.
- Deletion: Removing an element.
- Searching: Finding the index of an element.

Time Complexity:

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

DSA (Data Structures and Algorithms) Notes

3. Linked Lists

Definition: A linear data structure where each element (node) points to the next.

Types:

- Singly Linked List
- Doubly Linked List
- Circular Linked List

Operations:

- Insertion/Deletion at beginning, middle, end.
- Searching

Advantages:

- Dynamic size.
- Efficient insertions/deletions.

Disadvantages:

- No direct access by index.

DSA (Data Structures and Algorithms) Notes

4. Stacks

Definition: A LIFO (Last In First Out) data structure.

Operations:

- Push (Insert)
- Pop (Remove)
- Peek/Top (View top element)

Applications:

- Expression evaluation
- Backtracking (undo feature)
- Function call stack

Time Complexity: O(1) for all operations

DSA (Data Structures and Algorithms) Notes

5. Queues

Definition: A FIFO (First In First Out) data structure.

Types:

- Simple Queue
- Circular Queue
- Priority Queue
- Deque (Double-ended queue)

Operations:

- Enqueue (Insert)
- Dequeue (Remove)
- Peek (View front element)

Applications:

- CPU scheduling
- Printer queues

DSA (Data Structures and Algorithms) Notes

6. Trees

Definition: A hierarchical data structure with nodes connected by edges.

Types:

- Binary Tree
- Binary Search Tree (BST)
- AVL Tree (Self-balancing)
- Heap (Min-Heap, Max-Heap)

Traversals:

- Inorder (Left, Root, Right)
- Preorder (Root, Left, Right)
- Postorder (Left, Right, Root)
- Level-order (BFS)

Applications:

- File systems
- Databases
- Expression parsing

DSA (Data Structures and Algorithms) Notes

7. Graphs

Definition: A set of vertices (nodes) connected by edges.

Types:

- Directed vs Undirected
- Weighted vs Unweighted

Representations:

- Adjacency Matrix
- Adjacency List

Algorithms:

- BFS (Breadth-First Search)
- DFS (Depth-First Search)
- Dijkstras Algorithm
- Floyd-Warshall Algorithm
- Kruskals and Prims (for MST)

Applications:

- Social networks
- Web crawling
- GPS systems

8. Sorting Algorithms

Used to arrange data in a particular order (ascending/descending).

Types:

- Bubble Sort: O(n^2)
- Insertion Sort: O(n^2)
- Selection Sort: O(n^2)
- Merge Sort: O(n log n)
- Quick Sort: O(n log n)
- Heap Sort: O(n log n)
- Radix/Bucket Sort (Non-comparison based)

Applications:

- Searching
- Data analysis

DSA (Data Structures and Algorithms) Notes

9. Searching Algorithms

Used to find a specific element in a data structure.

Types:

- Linear Search: O(n)
- Binary Search (on sorted data): O(log n)
- Interpolation Search
- Exponential Search

Binary Search requires sorted data.

DSA (Data Structures and Algorithms) Notes

10. Recursion

A function calling itself to solve smaller instances of a problem.

Key Concepts:

- Base Case

- Recursive Case

Applications:

- Factorial

- Fibonacci

- Tower of Hanoi

- Tree Traversals

Time complexity depends on recursion depth.

11. Dynamic Programming (DP)

DP is used to optimize recursive problems by storing results of subproblems.

Approaches:

- Memoization (Top-Down)
- Tabulation (Bottom-Up)

Common Problems:

- Fibonacci
- Knapsack Problem
- Longest Common Subsequence
- Matrix Chain Multiplication

Time Complexity: Reduced to O(n^2) or better

12. Greedy Algorithms

Greedy algorithms make the locally optimal choice at each step.

Applications:

- Fractional Knapsack
- Activity Selection
- Huffman Encoding
- Kruskals & Prims MST

Not always optimal, but efficient.

DSA (Data Structures and Algorithms) Notes

13. Hashing

Hashing is used to map data to a fixed-size table using a hash function.

Key Concepts:

- Hash Functions
- Collision Handling (Chaining, Open Addressing)
- Load Factor

Applications:

- Hash Tables
- Caching
- Symbol Tables

14. Backtracking

Solves problems by trying out all possibilities and backtracking upon failure.

Applications:

- N-Queens Problem

- Sudoku Solver

- Maze Problems

- Subset Generation

DSA (Data Structures and Algorithms) Notes

15. Bit Manipulation

Working with binary representations using bitwise operators.

Common Operations:

- AND, OR, XOR, NOT
- Left/Right Shift

Applications:

- Checking power of 2
- Swapping using XOR
- Counting set bits