Data Structures and Algorithms

Data Structures and Algorithms (DSA) are fundamental concepts in computer science.

KEY DATA STRUCTURES:

1. Arrays
   - Fixed-size sequential collection
   - O(1) access time by index
   - Example: int arr[5] = {1,2,3,4,5};
2. Linked Lists
   - Dynamic size, nodes with pointers
   - Types: Singly, Doubly, Circular
   - Operations: Insert O(1), Delete O(1), Search O(n)
3. Stacks
   - LIFO (Last In First Out) principle
   - Operations: Push, Pop, Peek
   - Applications: Expression evaluation, backtracking
4. Queues
   - FIFO (First In First Out) principle
   - Types: Simple, Circular, Priority, Deque
   - Applications: CPU scheduling, BFS algorithm
5. Trees
   - Binary Tree: Each node has max 2 children
   - Binary Search Tree (BST): Left < Root < Right
   - Operations: Insert, Delete, Search - O(log n) average
6. Graphs
   - Vertices and Edges
   - Representations: Adjacency Matrix, Adjacency List
   - Algorithms: BFS, DFS, Dijkstra, Kruskal

SORTING ALGORITHMS:

1. Bubble Sort - $O(n^2)$ - Simple but inefficient
2. Selection Sort - $O(n^2)$ - Finds minimum repeatedly
3. Insertion Sort - $O(n^2)$ - Good for small datasets
4. Merge Sort - O(n log n) - Divide and conquer
5. Quick Sort - O(n log n) average - Most used
6. Heap Sort - O(n log n) - Uses heap data structure

SEARCHING ALGORITHMS:

1. Linear Search - O(n) - Sequential search
2. Binary Search - O(log n) - Requires sorted array
3. Depth First Search (DFS) - For graphs/trees
4. Breadth First Search (BFS) - Level-order traversal

TIME COMPLEXITY BASICS:

- O(1): Constant time
- O(log n): Logarithmic (Binary Search)

- O(n): Linear (Loop through array)
- O(n log n): Efficient sorting
- O(n^2): Nested loops (Bubble Sort)
- O(2^n): Exponential (Fibonacci recursive)

SPACE COMPLEXITY:

- Memory used by algorithm
- Important for embedded systems
- Trade-off with time complexity