DSA (Data Structures & Algorithms) with C++

DSA Introduction

- DSA = Data Structures + Algorithms → Organize & process data efficiently.
- Importance: Efficient coding, problem-solving, interview prep, real-world apps.

What is Data?

- Raw facts/information that can be stored, processed, and analyzed.
- Types: Structured, Unstructured, Semi-structured.
- Sorted Data → arranged (ascending/descending) → faster search.
- Unsorted Data → no specific order → slower search.

Data Structures

- Array: Fixed-size, fast access, slow insert/delete.
- Linked List: Dynamic, easy insert/delete, slow access.
- Stack: LIFO, used in undo/redo, expression evaluation.
- Queue: FIFO, used in scheduling, printers.
- Tree: Hierarchical (Binary, BST), used in indexing, filesystems.
- Graph: Nodes + Edges, used in maps, networking.
- Hash Table: Key-value store, O(1) lookup, used in caches, credentials.

Algorithms

- Searching: Linear (O(n)), Binary (O(log n)) on sorted data.
- Sorting: Bubble ($O(n^2)$), Merge (O(n log n)), Quick (O(n log n)).
- Graph: BFS (level-wise), DFS (deep path), Dijkstra (shortest path).
- Dynamic Programming: Fibonacci, Knapsack → optimize by remembering.

Industry Applications

- E-commerce: Searching & Sorting in product catalogs.
- Search Engines & Social Media: Sorting feeds by relevance.
- Google Maps/Uber: Dijkstra for shortest path.
- Banking & Finance: Knapsack for portfolio optimization.
- Cloud Computing: DP for resource allocation.
- Games: Graphs & Trees for pathfinding.

DSA with C++

- Why C++: Fast execution, STL (vectors, stacks, queues, maps), memory control, widely used.
- Examples: Arrays, Linked Lists, Stack/Queue (STL), Binary Search, Graph (Adjacency List).
- Real Projects: Google Maps (Graphs), Amazon (Sorting/Searching), OS (Queues), Banking (Hash Tables).