- **Dynamic Programming**:
  - Fibonacci Series
  - Longest Common Subsequence (LCS)
  - Knapsack Problem
  - Matrix Chain Multiplication
  - Shortest Paths in DAG
- **String Algorithms**:
  - Pattern Searching (Naive, KMP, Rabin-Karp)
  - Longest Common Substring
  - Longest Palindromic Subsequence
  - Edit Distance
- **Mathematical Algorithms**:
  - Prime Number Generation
  - Factorial Calculation
  - GCD (Euclidean Algorithm)
  - Sieve of Eratosthenes
  - Fast Exponentiation (Modular Exponentiation)
- **Miscellaneous Algorithms**:
  - Backtracking
  - Greedy Algorithms
  - Divide and Conquer
  - Randomized Algorithms

2. ALGORITHMS

- **Sorting Algorithms:**
  - Bubble Sort
  - Selection Sort
  - Insertion Sort
  - Merge Sort
  - Quick Sort
  - Heap Sort
- **Searching Algorithms:**
  - Linear Search
  - Binary Search (iterative and recursive)
- **Graph Algorithms:**
  - Breadth-First Search (BFS)
  - Depth-First Search (DFS)
  - Shortest Path Algorithms (Dijkstra's, Bellman-Ford)
  - Minimum Spanning Tree Algorithms (Prim's, Kruskal's)
  - Topological Sorting
  - Strongly Connected Components (SCC)
- **Tree Algorithms:**
  - Tree Traversal (Inorder, Preorder, Postorder)
  - Binary Search Tree (BST) Operations (Insertion, Deletion, Search)
  - Lowest Common Ancestor (LCA)

@code._learning

3. UTILITY CLASSES

- **Helper functions**: Generic functions commonly used across different algorithms.
- **Input/Output** handling: Utility functions for reading input and printing output.

4. TESTING

- **Unit Tests**: Tests for each data structure and algorithm to ensure correctness.
- **Performance Tests**: Benchmarking to evaluate time and space complexity.

5. DOCUMENTATION

- **Javadoc**: Documentation for each class, method, and data structure.

Each data structure and algorithm would typically be implemented in its own class or module, with well-defined interfaces for ease of use and reusability. Additionally, efficient implementations, along with considerations for edge cases and optimizations, are crucial for a robust DSA library.

1. DATA STRUCTURES

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

Linked Lists: A linear collection of elements where each element points to the next.

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

Queues: A First-In-First-Out (FIFO) data structure.

Trees: Hierarchical data structures with nodes connected by edges.

Graphs: Non-linear data structures consisting of vertices and edges.

Hash Tables: Data structures that implement associative arrays or mappings of keys to values.

Heaps: Tree-based data structures where the parent node is either greater or smaller than its child nodes.

Trie: An efficient information retrieval data structure.

Disjoint Set (Union-Find): A data structure that keeps track of a set of elements partitioned into disjoint subsets.