Exercise 1: Inventory Management System

Understand the Problem:

Importance of Data Structures and Algorithms: Efficient data storage and retrieval are crucial in handling large inventories to ensure quick access, updates, and management of data.

Suitable Data Structures:

ArrayList: Good for simple operations and when the number of elements is dynamic.

HashMap: Ideal for fast lookups, insertions, and deletions based on keys.

Analysis:

Time Complexity:

Add: O(1)

Update: O(1)

Delete: O(1)

Optimization: Use HashMap for constant time complexity operations.

Exercise 2: E-commerce Platform Search Function

Understand Asymptotic Notation:

Big O Notation: Represents the upper bound of an algorithm's time complexity. It helps analyze how the algorithm scales with input size.

Search Operations:

Best Case: O(1) (Immediate find)

Average Case: O(n) for linear search, O(log n) for binary search

Worst Case: O(n) for linear search, O(log n) for binary search

Analysis:

Time Complexity Comparison:

Linear Search: O(n)

Binary Search: O(log n)

Suitability: Binary search is preferred for sorted data due to better time complexity.

Exercise 3: Sorting Customer Orders

Understand Sorting Algorithms:

Bubble Sort: O(n²) - Simple but inefficient for large data.

Insertion Sort: O(n²) - Better for small or nearly sorted datasets.

Quick Sort: O(n log n) - Efficient for large datasets.

Merge Sort: O(n log n) - Stable and efficient for large datasets.

Analysis:

Performance Comparison:

Bubble Sort: O(n²)

Quick Sort: O(n log n)

Preference: Quick Sort is generally preferred due to better performance with large datasets.

Exercise 4: Employee Management System

Understand Array Representation:

Array Representation: Arrays are stored in contiguous memory locations, allowing O(1) time complexity for accessing elements but requiring fixed size and costly resizing.

Analysis:

Time Complexity:

Add: O(1) (if capacity is sufficient)

Search: O(n)

Traverse: O(n)

Delete: O(n)

Limitations: Fixed size and costly resizing. Arrays are best used for static or small datasets.

Exercise 5: Task Management System

Understand Linked Lists:

Singly Linked List: Each node points to the next node. Operations are efficient for adding/removing nodes.

Doubly Linked List: Each node points to both the next and previous nodes. Allows bi-directional traversal.

Analysis:

Time Complexity:

Add: O(1) (at the head)

Search: O(n)

Traverse: O(n)

Delete: O(n)

Advantages: Dynamic size, efficient insertions/deletions compared to arrays.

Exercise 6: Library Management System

Understand Search Algorithms:

Linear Search: O(n) - Simple but inefficient for large datasets.

Binary Search: O(log n) - Efficient for sorted datasets.

Analysis:

Time Complexity Comparison:

Linear Search: O(n)

Binary Search: O(log n)

Usage: Binary search is more efficient with sorted data.

Exercise 7: Financial Forecasting

Understand Recursive Algorithms:

Recursion: A technique where a function calls itself to solve smaller instances of the problem. Useful for problems that can be divided into similar sub-problems.

Analysis:

Time Complexity: O(n) due to the number of recursive calls.

Optimization: Use memoization or iterative approaches to reduce redundant calculations.