**INVENTORY MANAGEMENT SYSTEM:**

1. The importance of Data Structures and Algorithms in handling large inventories is for efficient data storage and retrieval to ensure quick access to product information, fast updates, and minimal downtime, for scalability as the system should handle the increased load, and for optimization to optimize the system for memory and speed.

The type of data structures suitable for this problem are ArrayList(good for dynamic arrays) and HashMap(for quick insertions and deletions).

1. Chosen Data Structure -> HashMap
2. Time Complexity Add Product -> O(1) Update Product(1) -> O(1) Delete Product -> O(1) Get Product -> O(1)

The optimization for these operations may include to use good hash function to maintain O(1) performance, to monitor and adjust the load factor of the HashMap to balance memory usage, considering the addition of secondly indexes for fields like productName if frequent searches are to be performed on this attribute.

**E-COMMERCE PLATFORM SEARCH FUNCTION:**

1. Big O notation is used to describe the upper bound of an algorithm’s time complexity, giving an idea of the worst-case scenarios for the algorithm’s execution time as the input size grows. It helps in analysing the efficiency of an algorithm.

Best Case: Finding the element on first try.

Average Case: Representing all expected number of operations an algorithm will perform.

Worst Case: Scenario representing an algorithm performing the maximum number of operations.

1. Time Complexity:

Linear Search -> O(n) as it needs to check each element in an array.

Binary Search -> O(log n) as it halves the search interval in each step. But the only catch here is that, the array should be sorted.

Considering the array to be sorted already the suitable algorithm would be Binary Search with the time complexity of O(log n).

**SORTING CUSTOMER ORDERS:**

1. Bubble Sort -> Repeated sorting that compares adjacent elements and swaps them if they are unsorted. O(n2) for average and worst case and O(n) for best case.

Insertion Sort -> Builds the final sorted array one time at a time. O(n2) for average and worst case and O(n) for best case.

Quick Sort -> In-place sorting algorithm that works by selecting a pivot element and segregating elements towards its left or right based on the order. O(n log n) for average O(n2) worst case.

Merge Sort -> It is a divide and conquer algorithm. O(n log n) for all best, average and worst cases.

1. Bubble Sort:

Time complexity -> O(n2) for average and worst case.

Space complexity -> O(1)

Quick Sort:

Time complexity -> O(n log n) for average case.

Space complexity -> O(log n) due to recursive stack.

Quick sort is generally preferred over Bubble sort due to its lesser time complexity, efficiency, scalability and adaptability. It can work over large datasets efficiently when compared to Bubble sort.

**EMPLOYEE MANAGEMENT SYSTEM:**

1. In memory arrays are stored in contiguous memory location allowing efficient access via their index. They store data of fixed size and same type. It is a O(1) operation.

Advantages:

Fast access

Efficient use of memory

Easy to use for a collection of fixed data and same type.

1. Add -> O(1)

Search -> O(n)

Traverse -> O(n)

Delete -> O(n)

Limitations:

Arrays only work with fixed size making it allergic to dynamic situation and deletion of elements require the shifting of remaining elements.

Should only be used when the data are of same type, length is fixed, for quick access and when memory usage needs to be minimised.

**TASK MANAGEMENT SYSTEM:**

1. Singly Linked Lists -> Each node contains a data and a reference pointing to the next node in the sequence. The last node points to null.

Doubly Linked List -> Each node contains a data, a reference to the next node, and a reference to the previous node allowing the traversal in both directions.

1. Add -> O(n) for worst case, if maintained a trail reference the complexity can be reduced to O(1).

Search -> O(n)

Traverse -> O(n)

Delete -> O(n)

Advantages of Linked List over arrays for dynamic data:

Dynamic size makes Linked List flexible as they can grow and shrink dynamically. Insertion or deletion of nodes is more efficient when position is known without shifting of remaining elements.

**LIBRARY MANAGEMENT SYSTEM:**

1. Linear Search:

It is a sequential search for finding a target value within a list.

Time Complexity -> O(n)

Binary Search:

Finds a target value within a sorted list. Repeatedly divides the list in half.

Time Complexity -> O(n)

1. Time Complexity:

Linear Search -> O(n) as it may require checking each element in a list.

Binary Search -> O(log n) as it repeatedly divides the list in half.

When to Use?

Linear Search -> Small and unsorted datasets.

Binary Search -> Large and sorted datasets.

**FINANCIAL FORECASTING:**

1. Recursion is a method of solving problems where a function calls itself repeatedly. Every recursive function has at least one base case, which terminates the recursion.

Ex – Factorial calculation of N.

1. Time Complexity -> O(n). Function calls itself n times.

Optimization -> Memoization. It stores the results of expensive function calls and returns the temporarily stored result when the same inputs occur again.