1. Inventory Management System

Q.Explain why data structures and algorithms are essential in handling large inventories.

ans: Effiecient data structures and algorithms are crucial in handling large inventories for reasons like,

Effiecient Storage:Memory usage is optimized and data redundancy is minimised.

Fast Retreival:Quick retreival of information.

Scalablity:As the inventory grows, the system should be able to handle the increased load without perfommance degradation.

Q.Discuss the type of data structures suitable for this problem.

ans:

1. Arrays - When inventory items are fixed.

2. LinkedList - Efficient insertion and deletion, inventory size can increase and it accounts that.

3. Hash Tables - Fast retreival, deletion and addition of inventory items. Requires good hash function to reduce collisions.

Q. Analyze the time complexity of each operation in your chosen data structure.

ans:

1. add() - O(1). Worst-Case: O(N)

2. delete() - O(1). Worst-Case: O(N)

3. update() - O(1).

4. display() - O(N): iterate over entire map.

Q. Discuss how you can optimize these operations.

ans: HashMap itself is a very optimised data structure. These steps can be ensured for its optimisation:

Choosing a good hash function, minimises collisions, ensures proper usage of memory-space.

Choosing an appropriate initial capacity, if approximate number of elements to be stored is known. This reduces the need for resizing which can be costly.

2. E-Commerce Platform Search function

Q. Explain Big O notation and how it helps in analyzing algorithms.

ans: Big O notation is a mathematical concept to describe the upper bound of an algorithm's run time or space complexity. It focuses on the dominant term as the input size grows, ignoring constant factors and lower-order terms. It provides a high level understanding of algorithm's efficiency and scalability.

It helps in providing insight into how an algorithm performs as the input size grows. It helps in understanding the worst-case perfommance.

Q. Describe the best, average, and worst-case scenarios for search operations.

ans:

Best Case Scenario : O(1) denoting constant time. The condition in which algorithm performs minimum number of operations.

Average Case Scenario : The expected perfommance of the algorithm when considering all inputs.

Worst Case Scenario : The condition under which the algorithm performs the most number of operations.

Q. Compare the time complexity of linear and binary search algorithms.

ans:

1. Linear Search - Best-Case: O(1), Average-Case: O(N), Worst-Case: O(N)

2. Binary Search - Best-Case: O(1), Average-Case: O(logN), Worst-Case: O(logN)

Q. Discuss which algorithm is more suitable for your platform and why ?

ans: Ecommerce platform has data that grows immensely with passing time, so an algorithm which shall give constantly the same time complexity for searching is mmore suitable. Thus, Binary Search is more suitable for my platform.

3. Sorting Customer Orders

Q.Explain different sorting algorithms(Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).

ans:

1. Bubble Sort : It is a comparison-based sorting algorithm. It repeteadly steps through the list, compares adjacent elements and swaps them if they are not at the correct place. This process continues till the list is sorted. Best Case:O(n) Average and Worst-Case:O(n2).

2. Insertion Sort : It builds the array one item at a time. Starts from the second element, assuming the first element is already sorted. Compare it with the element in sorted part of the list. Shift the sorted elements to the right to insert the new element at it's correct position. Repeated for all the elements. Best Case:O(n) Average and Worst-Case:O(n2).

3. Quick Sort : It is a divide-and-conquer algorithm that selects a pivot element and partitions the array into two sub-arrays, elements less than pivot are moved left and vice versa. Recursively applied to all sub-arrays. Finally, sorted sub-arrays and pivot is combined. Best Case:O(nlogn) Average-Case:O(nlogn) Worst-Case:O(n2).

4. Merge Sort : Divide and conquer algorithm that divides the array into two halves, sorts each half, and then merges the sorted halves to produce a sorted array. Each half is sorted recursively. Best Case, Average-Case and Worst-Case:O(nlogn).

Q. Compare the perfommance of Bubble Sort and Quick Sort.

ans:

1. Bubble Sort : Best-Case: O(N), Average-Case: O(N^2), Worst-Case: O(N^2)

2. Quick Sort : Best-Case: O(NlogN), Average-Case: O(NlogN), Worst-Case: O(N^2)

Q. Discuss why Quick Sort is generally preferred over Bubble Sort.

ans:Quick Sort can give O(NlogN) in average, which is also it's best case. While Bubble Sort, Average Time-Complexity is O(N^2). When the data set is huge, the difference is very prominient. This also proves that Quick Sort gives a better perfommance constantly.

4. Employee Management System

Q. Explain how arrays are represented in memory and their advantages.

ans: Arrays are stored in contiguous memory locations, which means each element of the array is placed next to the previous element in memory.

Their Advantages: Array elements can be accessed directly using its index. This provides constant time complexity O(1). As arrays are stored in contiguous memory location, there is minimal memory overhead.

Q. Analyze the time complexity of each operation.

ans:

1. add() - O(1).

2. search() - O(n). The entire list needs to be traversed to find an element.

3. traverse() - O(n). The entire list is traversed.

4. delete() - O(n). Element is first searched, then O(1) is the time complexity to delete that element.

Q. Discuss the limitations of array and when to use them.

ans:The size of array cannot be dynamically modified. Thus it has to be used only when data is static.

5. Task Management System

Q. Explain the different types of linked lists(Singly Linked List, Doubly Linked List)

ans: Single Linked List : Each node contains two parts: data and a pointer to the next node. The last node points to null. Insertion is simple. Deletion is effecient if node to be deleted is known. Access the elements by traversal.

Double Linked List : Each node contains data, a pointer to the previous node, a pointer to the next node. The first node's previous pointer and last nodes next pointer points to null. Allows bidirectional travel. Easier deletion of node as the previous node is directly accessed.

Q. Analyze the time complexity of each operation.

ans:

1. add() - O(1). A node is added at beginning. O(n). A node is added elsewhere.

2. search() - O(n). The entire list needs to be traversed to find an element.

3. traverse() - O(n). The entire list is traversed.

4. delete() - O(1). The beginning node to be deleted. O(n). Any other node to be deleted.

Q. Discuss the advantages of linked lists over arrays for dynamic data.

ans: As data is dynamic, it is not possible to know the final size of data. Thus, an array can't be use. Even if an array is used, when the limit of array cross, it must be moved to a different array, this shall cost enough overhead. While linked lists, a node can be added any time without any large overhead. Arrays occupy contiguous memory locations, a larger block of memory needs to be asssigned to it. While linked list can do better memory usage.

6. Library Management System

Q. Explain linear search and binary search algorithms.

ans: Linear Search : It is a straightforward searching algorithm that checks each element in a list sequentially until the desired element is found. Works on both sorted and unsorted lists. Time Complexity : O(n)

Binary Search : It repeteadly divides the list in half to locate the target value. It works only on sorted list. Time Complexity : O(logn).

Q. Compare the time complexity of linear and binary search.

ans:

Linear Search :

Best-Case : O(1)

Average-Case : O(N)

Worst-Case : O(N)

Binary Search :

Best-Case : O(1)

Average-Case : O(logn)

Worst-Case : O(logn)

Q. Discuss when to use each algorithm based on the data set size and order.

ans:

1) When list is ordered. Use Binary Search as it will give a time complexity of O(logn) much better than Linear Search.

2) When list is unsorted, linear search has to be used. Binary search worls only on sorted list.

3) When list size is small, linear search can be used.

4) When list size is immense, it is necessary for list to be sorted. Then, Binary Sort is the best option.

7. Financial Forecasting

Q. Explain the concept of recursion and how it can simplify certain problems.

ans: Recursion is a programming technique where a function calls itself in order to solve a problem. This breaks down the complex problem into simpler, more manageable sub-problems. They make the code more intuitive and easier to understand in some cases. They can simplify problems by breaking them down to simpler sub-problems that are easier to solve.

Q. Discuss the time complexity of your recursive algorithm.

ans: The algorithm depends on the variable periods. So, O(N) is time complexity.

Q. Explain how to optimize the recursive solution to avoid excess computation.

ans: This recursion is very straight forwawrd. Memoization can't be used as there are no overlapping sub-problems.

We can use iterative aproach this will avoid stack overflow for deep recursion problems.