**Week 1:DSA (Rahul Chakraborty)**

Exercise 1:

Question 1: Data Structures are used to store and organize the data ,about our inventory, efficiently and effectively. Since we will be storing a large amount of different products we cannot store them in just primitive variables.

Well, we can use Arrays, ArrayList, LinkedList or HashMaps. However, Hashmap would be the better choice since product Id would be unique which in this case can be our key in HashMap.

Question 4: Add has an average time complexity of O(1).

Update has an average time complexity of O(1).

Delete has an average time complexity of O(1). The methods provided are already optimized.

Exercise 2:

Question 1: Big O notation is a mathematical notation used to describe the worst-case time complexity or the worst-case space complexity of a data structure. It provides a way to compare the performance of different algorithms and data structures, and to predict how they will behave as the input size increases.

Best Case for searching: O(1) When the element we want to search is available as the first element and we get it as soon as we traverse the data structure. And there is no need to traverse the left-over data structure.

Average Case for searching: O(n+1/2) When the element we want to search is either present in the first index or in the last index or it is present in middle. Thus, average of all the possibilities (1+2+3…..n)/n.

Worst Case for searching: O(n) When the element we want to search is present in the last index of our data structure. Traversing all the elements and thus it leads to the worst case.

Question 4: Linear search: O(n) and Binary Search: O(log n)

Both Linear Search and Binary Search have been implemented here. Since a sorted array is already been used, there is no overhead of first sorting the array and the perform binary search. Thus, binary search would be perfect for this platform since it takes less time.

Exercise 3:

Question1: Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Insertion sortis a simple sorting algorithm that works by iteratively inserting each element of an unsorted list into its correct position in a sorted portion of the list.

Quick Sortis a sorting algorithm based on the Divide and Conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array.

Merge sortis a sorting algorithm that follows the divide-and-conquerapproach. It works by recursively dividing the input array into smaller subarrays and sorting those subarrays then merging them back together to obtain the sorted array.

Question4: Bubble Sort: O(n^2) and Quick Sort: O(nlogn)

Quick Sort is much faster on average compared to Bubble Sort, especially for large datasets. It efficiently reduces the problem size by partitioning the array, making it a better choice for larger datasets.

Exercise 4:

Question1: Arrays, in java, are stored in the heap memory. In memory, arrays are stored in contiguous locations. Each element is stored in adjacent memory locations. The memory representation of an array is like a long tape of bytes, with each element taking up a certain number of bytes.

Advantages:

* Arrays provide direct and efficient access to any element in the collection.
* Arrays allow for fast data retrieval because the data is stored in contiguous memory locations.
* Arrays are easy to implement and understand.

Question4: Add: O(n) ,Search: O(n), Traverse: O(n), Delete: O(n)

Limitations:

* Arrays have a fixed size that is determined at the time of creation. This means that if the size of the array needs to be increased, a new array must be created and the data must be copied from the old array to the new array, which can be time-consuming and memory-intensive.
* Inserting or deleting an element from an array can be inefficient and time-consuming because all the elements after the insertion or deletion point must be shifted to accommodate the change.
* If an array is not fully populated, there can be wasted space in the memory allocated for the array.

Arrays are suitable when the number of elements is known in advance and does not change frequently.

Exercise 5:

Question1: Singly linked list is a linear data structure in which the elements are not stored in contiguous memory locations and each element is connected only to its next element. It can only move in forward direction.

Doubly linked list is a data structure that has reference to both the previous and next nodes in the list. It provides simplicity to traverse, insert and delete the nodes in both directions in a list.

In a doubly linked list, each node contains three data members:

* data:The data stored in the node
* next:It refers to the reference to the next node
* prev: It refers to the reference to the previous node

Question4: Add: O(1), Search: O(n), Traverse: O(n), Delete: O(n)

There are several advantages of linked list over arrays, such as dynamic size, efficient insertion and deletion, memory efficiency and many more.

Exercise 6:

Question1: The linear search algorithm is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found; otherwise, the search continues till the end of the dataset.

Binary search is one of the searching techniques applied when the input is sorted here we are focusing on finding the middle element that acts as a reference frame whether to go left or right to it as the elements are already sorted.

Question 4: Linear search: O(n) and Binary Search: O(log n)

Linear search can be used on both sorted and unsorted data. It doesn't require any preprocessing or sorting of the data set. For small data sets, the simplicity of linear search can make it an appropriate choice.

Binary search requires the data set to be sorted. It is only applicable when the data is in a specific order (ascending or descending). For large data sets, binary search significantly outperforms linear search due to its logarithmic time complexity.

Exercise 7:

Question1: Recursion is a technique in programming where a function calls itself to solve smaller instances of the same problem until a base condition is met. It is particularly useful for problems that can be broken down into smaller, similar sub-problems. It simplifies code for problems that have a naturally recursive structure (for e.g. factorial).

Question4: The formula for calculating the final value is p\*(1+r)^n, where p= present value, r=rate and n= no. of periods. We can see that we have a (1+r) which runs n times. Thus calling the recursive methods n times would be enough. If n=0, then the final value becomes p\*(1+r)^0 which is p\*1=p. Thus, we have used this as our base case. Therefore, the time complexity is O(n).

We can simply not use Recursion here to optimize our code. For both recursive and iterative approach, the time complexity is O(n). However, the space complexity is O(1) in case of iterative approach. This improves efficiency.