**1.Understand Array Representation:**

***Q1.Explain how arrays are represented in memory and their advantages.***

**Array Representation in Memory:**

Contiguous Memory Allocation: All the elements of the arrays are stored in contiguous memory locations. That is, every element is addended right after the previous one in memory.

Fixed Size: The size of an array is declared at the time it is declared and is unchangeable. This enables efficient indexing and thus random access to elements.

Indexing: An array is accessed through an index that allows for access to any element by its index in constant time, O(1).

**Arrays Advantages:**

Fast Access: This provides direct access to the elements through an index, and hence is used to provide access time complexity of O(1).

Memory Efficiency: Contiguous memory allocation may be memory efficient in some cases.

Simplicity: The simplicity of an array makes it easy to understand and use; it is one of the basic data structures in programming.

**2.Analysis:**

***Q. Analyze the time complexity of each operation (add, search, traverse, delete).***

**Time Complexity:**

Add:

Best Case: O(1) (adding to the end of the array if there is space)

Average Case: O(1)

Worst Case: O(1)

Search:

Best Case: O(1) (if the element is at the beginning)

Average Case: O(n)

Worst Case: O(n) (if the element is at the end or not found)

Traverse:

Best Case: O(n)

Average Case: O(n)

Worst Case: O(n)

Delete:

Best Case: O(1) (if the element is at the end)

Average Case: O(n)

Worst Case: O(n) (if the element is at the beginning and requires shifting).

***Q.Discuss the limitations of arrays and when to use them.***

Limitations of Arrays:

Fixed Size: Arrays have a fixed size, so once they are full, no more elements can be added without creating a new, larger array and copying the elements over.

Inefficient Insertions and Deletions: Inserting or deleting elements can be inefficient, especially if done frequently, as it may require shifting elements.

Contiguous Memory Allocation: Requires a contiguous block of memory, which might be problematic for large arrays.

When to Use Arrays:

Static Data: When the number of elements is known in advance and doesn't change frequently.

Fast Access: When fast access to elements by index is required.

Memory Efficiency: When memory layout and cache performance are critical.