**Week-I**

**Algorithms & DSA**

**Exercise-4:**

**Array Representation in Memory:**

* Contiguous Memory Allocation: Arrays are stored in contiguous memory locations i.e each element in the array is located next to the other in memory.
* Indexing: Arrays allow constant time O(1) access to elements using indices, which makes them very efficient for read operations.
* Fixed Size: The size of an array is fixed at the time of creation. You cannot change its size dynamically.

**Advantages of Arrays:**

* Fast Access: Direct access to elements via indices.
* Memory Efficiency: Arrays are memory efficient due to their contiguous storage.
* Predictable Performance: Access time is constant, making performance predictable.

**Time Complexity Analysis:**

* Add Operation: O(1) - Adding an employee to the end of the array is constant time.
* Search Operation: O(n) - Searching requires scanning the array, which takes linear time.
* Traverse Operation: O(n) - Traversing all elements in the array takes linear time.
* Delete Operation: O(n) - Deleting requires searching for the element and shifting subsequent elements, which takes linear time.

**Limitations of Arrays:**

* **Fixed Size:** The size of the array is fixed upon creation, which means it cannot dynamically grow or shrink based on the number of employees.
* **Inefficient Deletion and Insertion:** Inserting or deleting elements in the middle of the array requires shifting elements, which can be inefficient for large arrays.
* **Wasted Memory:** If the array is larger than the number of employees, there will be wasted memory.

**When to Use Arrays:**

* Arrays are suitable when the number of elements is known in advance and does not change frequently.
* They are efficient for read-heavy operations where fast access to elements is required.
* Arrays are preferable when the overhead of dynamic resizing is not acceptable or necessary.