**Exercise 4: Employee Management System**

**1. Understand Array Representation**

**Array Representation:** Arrays are a fundamental data structure used to store a fixed-size sequence of elements of the same type. In memory, arrays are represented as contiguous blocks of memory. Each element in the array is stored at a specific index, and the memory address of each element is calculated based on the base address of the array and the size of each element.

**Advantages of Arrays:** Arrays offer constant time complexity O(1) for direct access to elements using their index, making them simple and efficient for storing and accessing sequences of elements. Their contiguous memory layout enhances cache performance and memory efficiency. However, resizing or inserting elements can be inefficient.

**4. Analysis**

**Time Complexity Analysis:**

* **Add Employee:** In the worst case, adding an employee is O(n) due to the need to resize the array, which involves copying all elements to a new array. Once resized, the operation is O(1). On average, if resizing is not required, adding an employee is O(1).
* **Search Employee by ID:** Searching for an employee by ID has a time complexity of O(n) because, in the worst case, you may need to scan through the entire array to locate the employee.
* **Traverse and Display Employees:** Traversing and displaying all employees requires O(n) time, as each employee in the array must be visited exactly once.
* **Delete Employee by ID:** Deleting an employee has a worst-case time complexity of O(n) because it involves both finding the employee and shifting subsequent elements to fill the gap. The average case remains O(n) due to the necessary element shifting.

**Limitations of Arrays:**

Arrays have a fixed size, so if more elements need to be added, resizing is required, which involves copying elements to a new array. Inserting or deleting elements, especially in the middle, is inefficient due to the need for shifting. Additionally, if the initial size is overestimated, memory can be wasted, while underestimation may lead to frequent resizing.

**When to Use Arrays:**

* Use arrays when you have a fixed number of elements or when the number of elements is known in advance.
* Arrays are suitable for scenarios where direct access and simple iteration are required, and resizing or dynamic operations are not frequent.