**Employee Management System**

**1. Understand Array Representation**

Array Representation: The array is one of the simplest data structures which can be used to store the fixed-size sequential collection of items of the same data type. When it comes to memory, arrays are considered as a linear chunk of memory which in most cases they actually are. In an array, each item is a member of the structured element, and each element uses the base address of the array and the size of every element to create its memory address.

Advantages of Arrays: Arrays have constant time complexity O (1) for direct access to its elements by index that is why it is easy and efficient to store and search sequences of elements. They have better localization in cache and memory thus improving on the performance. However, when it comes to resizing or inserting elements the approach can be less efficient.

**4. Analysis**

**Time Complexity Analysis:**

• Add Employee: Worst case is employed, so that adding an employee is O(n) because of resizing which means copying all the elements to a new array. After it is resized, the specifics are O(1). Based on the analyses above, it can be claimed that in the worst-case scenario, if no resizing is needed, the complexity of adding an employee to the system is O(1).

• Search Employee by ID: To search for an employee using an ID, the time complexity is O(n), mainly because you may have to search through the array to get the index of the employee in question.

• Traverse and Display Employees: To traverse and display all employees, at most O(n) time complexity is used since every employee in the array will be processed once.

• Delete Employee by ID: Worst case time analysis of deleting an employee = O(n) as finding the employee and shifting other elements to overwrite his position are involved. The average case still is O(n) because of the required shifting of elements.

**Limitations of Arrays:**

Essential point Arrays: have a fixed size If more elements need to be added they have to resize that is bad because involves copying of elements to the new data array. Adding or removing one or more elements, especially to or from the middle of the list becomes time-consuming due to shifting. Furthermore, if the initial size is too large, memory will be unnecessarily used, but if the size is too small then the container may be resized very often.

**When to Use Arrays:**

• Arrays should be used when you know the number of elements that is going to be inserted in an array is limited or fixed in advance.

• Arrays can be used where one needs to get direct access and basic traversal and the resizing or dynamic operations are not often needed.