**Exercise 4: Employee Management System**

**Scenario:**

You are developing an employee management system for a company. Efficiently managing employee records is crucial.

**Steps:**

1. **Understand Array Representation:**
   * Explain how arrays are represented in memory and their advantages.
2. **Setup:**
   * Create a class Employee with attributes like **employeeId**, **name**, **position**, and **salary**.
3. **Implementation:**
   * Use an array to store employee records.
   * Implement methods to **add**, **search**, **traverse**, and **delete** employees in the array.
4. **Analysis:**
   * Analyze the time complexity of each operation (add, search, traverse, delete).
   * Discuss the limitations of arrays and when to use them.

**Explanation**

**Step 1: Understand Array Representation**

**How Arrays are Represented in Memory:**

* **Contiguous Memory Allocation:** Arrays are stored in contiguous memory locations. Each element in the array is placed next to the previous element.
* **Indexing:** Arrays provide direct access to elements using their index. The index starts from 0 for the first element.
* **Fixed Size:** Arrays have a fixed size, which is determined at the time of their creation. The size cannot be changed during runtime.

**Advantages of Arrays:**

* **Fast Access:** The primary advantage of arrays is fast access to elements. Accessing an element by its index takes constant time, O(1).
* **Memory Efficiency:** Arrays are memory-efficient because they have a low overhead. There is no additional storage required for pointers or metadata.
* **Predictability:** The memory layout of arrays is predictable, which can lead to better cache performance.

**Step 4: Analysis**

**Time Complexity of Each Operation:**

* **Add Employee:** O(1)
  + Adding an employee to the array is a constant time operation if there is space available.
* **Search Employee:** O(n)
  + Searching for an employee requires a linear scan through the array, leading to a linear time complexity.
* **Traverse Employees:** O(n)
  + Traversing the array to display all employees involves visiting each element, resulting in linear time complexity.
* **Delete Employee:** O(n)
  + Deleting an employee involves finding the employee (O(n)) and shifting elements to fill the gap (O(n)), resulting in a linear time complexity.

**Limitations of Arrays and When to Use Them:**

* **Fixed Size:** The primary limitation of arrays is their fixed size. Once an array is created, its size cannot be changed, which can lead to wasted space or insufficient capacity.
* **Insertion and Deletion:** Inserting or deleting elements in an array can be inefficient, especially if the operation needs to occur at the beginning or middle of the array. These operations can require shifting elements, leading to O(n) time complexity.
* **When to Use:** Arrays are best used when the size of the dataset is known in advance and does not change frequently. They are ideal for scenarios requiring fast access to elements by index and where memory efficiency is a concern.