**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.

Arrays are a data structure that store elements of the same type in contiguous memory locations. This allows for efficient indexing and access since the memory address of each element can be calculated using its index. Given a base address A of the array and an index i, the address of the element can be calculated as:

Address(A[i])=Base Address(A)+i× Size of each element

1. **Setup:**
   * Create a class Employee with attributes like **employeeId**, **name**, **position**, and **salary**.
2. **Implementation:**
   * Use an array to store employee records.
   * Implement methods to **add**, **search**, **traverse**, and **delete** employees in the array.
3. **Analysis:**
   * Analyze the time complexity of each operation (add, search, traverse, delete).

**Time Complexity of Operations**

* **Add**:
  + Best Case: **O(1)** (adding at the end)
  + Average Case: **O(1)**
  + Worst Case: **O(1)** (if there is capacity)
* **Search**:
  + Best Case: **O(1)** (if the element is the first one)
  + Average Case: **O(n)**
  + Worst Case: **O(n)**
* **Traverse**:
  + Best Case: **O(n)**
  + Average Case: **O(n)**
  + Worst Case: **O(n)**
* **Delete**:
  + Best Case: **O(1)** (if the element is the last one)
  + Average Case: **O(n)**
  + Worst Case: **O(n)** (shifting elements)
* Discuss the limitations of arrays and when to use them.
  +  Fixed Size: Arrays have a fixed size, and resizing requires creating a new array and copying elements, which is inefficient.
  +  Inefficient Deletion/Insertion: Deleting or inserting elements requires shifting elements, which can be time-consuming.
  +  Memory Waste: If the array has many unused slots, it wastes memory.

**When to Use Arrays**

* **Static Data**: When the size of the dataset is known in advance and does not change.
* **Fast Access**: When you need fast access to elements by index.
* **Memory Contiguity**: When you need contiguous memory allocation for performance reasons.