**Exercise 4: Employee Management System**

**Step 1: Understand Array Representation**

**Array Representation in Memory**

* **Contiguous Memory Allocation:** Arrays are stored in contiguous memory locations, meaning all elements are stored one after another. This allows for constant time access to elements using their index.
* **Advantages:**
  + **Fast Access:** Direct access to any element using its index (O(1) time complexity).
  + **Memory Efficiency:** No overhead for storing pointers or additional structure information.
  + **Cache Friendly:** Due to their contiguous nature, arrays are cache-friendly, which can lead to performance improvements.

**Time Complexity of Operations**

* **Add Employee:**
  + **Best-case:** O(1) - Adding an employee at the end of the array.
  + **Worst-case:** O(1) - Adding an employee at the end of the array.
* **Search Employee:**
  + **Best-case:** O(1) - Finding the employee at the first position.
  + **Average-case:** O(n) - On average, the employee will be somewhere in the middle.
  + **Worst-case:** O(n) - The employee is at the last position or not present.
* **Traverse Employees:**
  + **Time complexity:** O(n) - All employees need to be visited.
* **Delete Employee:**
  + **Best-case:** O(1) - Deleting the last employee.
  + **Average-case:** O(n) - On average, shifting half the elements.
  + **Worst-case:** O(n) - Deleting the first employee and shifting all elements.

**Limitations of Arrays**

* **Fixed Size:** Arrays have a fixed size, making it difficult to add more elements once the capacity is reached.
* **Inefficient Deletion:** Deleting an element requires shifting subsequent elements, leading to O(n) time complexity.
* **Inefficient Insertion:** Inserting elements at positions other than the end requires shifting elements, also leading to O(n) time complexity.

**When to Use Arrays**

* **Fast Access:** When you need constant-time access to elements by index.
* **Static Size:** When the number of elements is known in advance and does not change frequently.
* **Memory Efficiency:** When memory overhead is a concern, as arrays do not require extra memory for pointers or additional structures.