**Exercise 4: Employee Management System**

**Understanding Array Representation:**

**Array Representation in Memory**

* **Contiguous Memory Allocation**: Arrays are stored in contiguous memory locations. This means that each element is placed next to the previous element in memory.
* **Fixed Size**: Arrays have a fixed size, which is defined at the time of creation. This size cannot be changed dynamically.
* **Direct Access**: Arrays allow direct access to elements using their index. The index acts as an offset from the base address of the array.

**Advantages of Arrays**

1. **Fast Access**: Direct access to elements using indices provides O(1) time complexity for accessing elements.
2. **Simple Data Structure**: Arrays are simple to use and understand, making them a fundamental data structure.
3. **Memory Efficiency**: Arrays are memory-efficient because they do not have additional overhead like linked lists.

**Analysis**

**Time Complexity**

* **Add Employee:** O(1) (if there is space available in the array)
* **Search Employee:** O(n) (linear search through the array)
* **Traverse Employees:** O(n) (iterate through the entire array)
* **Delete Employee:** O(n) (search and shift elements)

**Limitations of Arrays**

1. Fixed Size: Once defined, the size of the array cannot be changed. This can lead to wasted space if the array is too large or insufficient space if the array is too small.
2. Inefficient Insertions/Deletions: Inserting or deleting elements in the middle of the array requires shifting elements, leading to O(n) time complexity for these operations.
3. Linear Search: Searching for an element in an unsorted array takes O(n) time in the worst case.

**When to Use Arrays**

1. Static Data: When the size of the dataset is known in advance and doesn't change frequently.
2. Fast Access: When you need fast access to elements using an index.
3. Memory Constraints: When you need a simple and memory-efficient data structure without the overhead of pointers.