Exercise-4

1. Understanding Array Representation

Array Representation:

- **Memory Representation:** Arrays are contiguous blocks of memory where each element is stored in a consecutive location. Each element can be accessed in constant time, O(1), by computing its address using an index.
- **Advantages:**
  - **Direct Access:** Provides O(1) time complexity for access and modification by index.
  - **Simplicity:** Arrays are straightforward to use and implement.
  - **Efficient Use of Space:** Minimal overhead compared to other data structures with additional metadata.

4. Analysis

Time Complexity:

- **Add Employee:**
  - **Best Case:** O(1) if there is space available in the array.
  - **Worst Case:** O(1), but may involve checking for array size limits.
- **Search Employee:**
  - **Best Case:** O(1) if the employee is found at the beginning of the array.
  - **Worst Case:** O(n) if the employee is at the end or not found at all.
- **Traverse Employees:**
  - **Time Complexity:** O(n) where n is the number of employees, as each employee must be visited.
- **Delete Employee:**
  - **Best Case:** O(1) if the employee is at the end of the array.
  - **Worst Case:** O(n) if the employee is at the beginning or middle of the array, as elements must be shifted to fill the gap.

Limitations of Arrays:

- **Fixed Size:** Arrays have a fixed size once created. Adding more elements than the initial capacity requires creating a new array and copying elements, which is not efficient.
- **Shifting Elements:** Insertion and deletion operations are less efficient compared to data structures like `ArrayList` or linked lists, as elements may need to be shifted.

When to Use Arrays:

- Arrays are suitable when the number of elements is known ahead of time or when dynamic resizing is not needed. They are ideal for simple data structures where fixed-size storage and fast access are required.