**Understand Array Representation**

**Q1. Explain how arrays are represented in memory and their advantages.**

Answer: Arrays are a fundamental data structure in programming, and understanding how they are represented in memory is essential.

**Array Representation in Memory:**

In memory, an array is a contiguous block of memory locations, each of which can store a value of a specific data type. The memory locations are indexed, meaning each location has a unique address or index. The array's size is fixed, and it is allocated at compile-time.

**Advantages of Arrays:**

1. **Efficient memory usage**: Arrays store elements in contiguous memory locations, making it efficient in terms of memory usage.
2. **Fast access**: Arrays provide fast access to elements using their index.
3. **Cache-friendly**: Arrays are cache-friendly, which means that the CPU can prefetch elements, reducing memory access latency.

**Analysis:**

Q1. Analyze the time complexity of each operation (add, search, traverse, delete).

Answer: Here's a brief explanation of the time complexity of each operation:

* **addEmployee**: O(1) because we are simply adding an element to the end of the array.
* **searchEmployee**: O(n) because we are iterating through the entire array to find a specific employee.
* **traverseEmployees**: O(n) because we are iterating through the entire array to print all employee records.
* **deleteEmployee**: O(n) because we are iterating through the entire array to find the employee to delete, and then shifting all subsequent elements to the left.

**Q2. Discuss the limitations of arrays and when to use them.**

Answer: The limitations of arrays include:

* Fixed size: Arrays have a fixed size that is determined at the time of creation. This can be a limitation if the number of employees is unknown or dynamic.
* Slow search and deletion: Searching for a specific employee or deleting an employee from the middle of the array can be slow because it requires iterating through the entire array.
* Slow insertion: Inserting a new employee at a specific position in the array can be slow because it requires shifting all subsequent elements to the right.

Arrays are suitable when:

* The size of the data is fixed and known in advance.
* Fast access to elements by index is required.
* Memory efficiency is important.

However, in a real-world scenario, a more suitable data structure for an Employee Management System would be a dynamic data structure like a linked list or a database, which can handle dynamic additions and deletions of employees more efficiently.