Explain how arrays are represented in memory and their advantages.

Arrays are a linear data structure that stores elements in contiguous memory locations. This means that each element in the array is stored next to its neighboring elements in memory. The memory address of each element can be calculated using the base address of the array and the index of the element.

For example, if the base address of an integer array arr is 1000 and each integer takes 4 bytes, the address of arr[i] can be calculated as:

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

So, the address of arr[3] would be 1000 + (3 \times 4) = 1012.

Advantages of Arrays:

- **Random Access**: Arrays provide constant-time access to elements based on their index, making retrieval very fast.
- **Cache Locality**: Due to contiguous memory allocation, arrays have better cache locality, which improves performance.
- **Simple Structure**: Arrays are simple and easy to understand, making them beginner-friendly.
- **Efficient Memory Usage:** Arrays use contiguous memory locations, which reduces memory overhead compared to linked data structure

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

- **Add Employee**: O(1) - Adding an employee to the end of the array is a constant-time operation.
- **Search Employee**: O(n) - In the worst case, we may need to search through all employees.
- **Traverse Employees**: O(n) - We need to visit each employee once.
- **Delete Employee**: O(n) - In the worst case, we may need to shift all elements after the deleted employee.

Discuss the limitations of arrays and when to use them.

Limitations of Arrays

- **Fixed Size**: Arrays have a fixed size, which means you need to know the maximum number of elements in advance. This can lead to wasted memory if the array is not fully utilized or insufficient memory if the array is too small.
- **Inefficient Insertions and Deletions**: Inserting or deleting elements in the middle of the array requires shifting elements, which can be inefficient.
- **Contiguous Memory Allocation**: Arrays require contiguous memory allocation, which can be a limitation if large blocks of memory are not available.

When to Use Arrays

- When you need fast access to elements using an index.
- When the number of elements is known in advance and does not change frequently.
- When memory overhead needs to be minimized.