Analysis

Array Representation:

- **In Memory**: Arrays are stored in contiguous blocks of memory. This means each element in the array is stored next to the previous one, which allows for fast access using indices.
- **Advantages**:
  - **Fast Access**: Direct access to elements using indices (O(1) time complexity for retrieval).
  - **Efficient Memory Use**: No extra memory overhead compared to other data structures like linked lists.

Time Complexity:

- **Add Operation**: O(1) if there is space in the array.
- **Search Operation**: O(n) in the worst case because it may need to check every element.
- **Traverse Operation**: O(n) because it visits every element once.
- **Delete Operation**: O(n) because it may require shifting elements to fill the gap left by the deleted employee.

Limitations of Arrays:

- **Fixed Size**: Arrays have a fixed size, which means you must know the maximum number of elements in advance or handle resizing manually.
- **Inefficient Deletion and Insertion**: Operations like deletion or insertion are less efficient because they may require shifting elements.
- **Lack of Flexibility**: Arrays do not support dynamic resizing, unlike data structures like lists or dynamic arrays.

When to Use Arrays:

- **When Size is Known**: Arrays are suitable when you know the number of elements in advance and do not need to resize frequently.
- **For Simplicity**: When you need a simple and straightforward data structure with constant-time access and are working with a small to moderate number of elements.