Exercise 4: Employee Management System

1. **Analysis:**
   - Analyze the time complexity of each operation (add, search, traverse, delete).
   - Discuss the limitations of arrays and when to use them.

Time Complexity:

Operation	**Time complexity**	**Explanation**
Add	O(1) or O(n)	Inserting at the end is fast if there's space; inserting at the beginning/middle requires shifting.
Search	O(n)	Must check each element to find a match (e.g., by employee ID or name).
Traverse	O(n)	Must visit each employee one by one.
Delete	O(n)	Need to search and shift elements after deletion to maintain order.

Limitations of Arrays:

Limitation	**Explanation**
Fixed size	Arrays have a predefined size; resizing requires creating a new array and copying elements.
Inefficient insertion/deletion	Inserting or deleting from the middle requires shifting elements (O(n)).
Slow search	If data is unsorted, search is linear (O(n)).
Wasted memory or overflow	Pre-allocating large arrays can waste memory; too small can lead to overflow.
No dynamic memory handling	Arrays don't automatically grow/shrink like dynamic data structures (e.g., lists, hash maps).

When to Use Arrays:

When?	**Why?**
You know the number of employees in advance	Fixed-size is fine and avoids resizing overhead.
You need fast indexed access (e.g., by position)	Arrays offer O(1) access by index.
Memory is constrained and simple structure is needed	Arrays are memory-efficient and cache-friendly.
You don't perform frequent insertions/deletions	Arrays perform poorly with frequent structural changes.

Conclusion

- Arrays are simple and fast for static, fixed-size data with minimal changes.
- For real-world employee management systems, dynamic structures like hash maps or databases are preferred due to better performance for search and modifications.