Analysis

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

Ans.

**Add Employee**

The addEmployee method adds an employee to the end of the array.

Time Complexity: 𝑂(1) - This operation involves assigning the new employee to the next available index in the array, which takes constant time.

**Search Employee**

The searchEmployee method searches for an employee by employeeId.

Time Complexity: 𝑂(𝑛) - In the worst case, the method may have to check all employees in the array (if the employee is not found or is the last one in the array), resulting in linear time complexity.

**Traverse Employees**

The traverseEmployees method prints all employees in the array.

Time Complexity: 𝑂(𝑛) - This operation involves iterating through all employees in the array, which takes linear time.

**Delete Employee**

The deleteEmployee method searches for an employee by employeeId and deletes the employee if found.

Time Complexity: 𝑂(𝑛) - Searching for the employee takes 𝑂(𝑛) time in the worst case. Deleting the employee involves shifting all subsequent employees one position to the left, which also takes O(n) time. Therefore, the overall time complexity is linear.

1. Discuss the limitations of arrays and when to use them.

Ans.

**Disadvantages of array data structure**

* Fixed size: Arrays have a fixed size that is determined at the time of creation. This means that if the size of the array needs to be increased, a new array must be created and the data must be copied from the old array to the new array, which can be time-consuming and memory-intensive.
* Memory allocation issues: Allocating a large array can be problematic, particularly in systems with limited memory. If the size of the array is too large, the system may run out of memory, which can cause the program to crash.
* Insertion and deletion issues: Inserting or deleting an element from an array can be inefficient and time-consuming because all the elements after the insertion or deletion point must be shifted to accommodate the change.
* Wasted space: If an array is not fully populated, there can be wasted space in the memory allocated for the array. This can be a concern if memory is limited.
* Limited data type support: Arrays have limited support for complex data types such as objects and structures, as the elements of an array must all be of the same data type.
* Lack of flexibility: The fixed size and limited support for complex data types can make arrays inflexible compared to other data structures such as linked lists and trees.

**Need for Arrays**

When dealing with a large number of similar data items, such as roll numbers of multiple students, using individual variables for each item becomes impractical. This is where arrays come into play.

* Arrays allow us to store multiple values under a single variable name.
* They provide a way to efficiently manage and process large amounts of data.