**Exercise 1: Inventory Management System**

Data structures and algorithms are essential in handling large inventories for efficient data storage, retrieval, and manipulation. Suitable data structures include Array List and Hash Map.

**Analysis :**

 Adding a product: O(1)

 Updating a product: O(1)

 Deleting a product: O(1)

**Optimization :**

Using HashMap already provides efficient average-case time complexities for add, update, and delete operations. For large-scale optimizations, consider indexing strategies or more advanced data structures based on specific use cases.

**Exercise 2: E-commerce Platform Search Function**

 **Big O Notation**: Describes the upper limit of the time complexity.

 **Best, Average, and Worst-case Scenarios**: Important for understanding performance under different conditions.

**Analysis :**

 **Linear Search**: O(n)

 **Binary Search**: O(log n)

**Optimization :**

 For unsorted data, linear search is the only option.

 For sorted data, binary search is significantly faster.

**Exercise 3: Sorting Customer Orders**

Different sorting algorithms include Bubble Sort, Insertion Sort, Quick Sort, and Merge Sort.

**Analysis :**

 **Bubble Sort**: O(n^2)

 **Quick Sort**: O(n log n) (average case), O(n^2) (worst case)

**Optimization :**

Quick Sort is generally preferred due to its better average-case performance.

**Exercise 4: Employee Management System**

Arrays are contiguous blocks of memory, allowing for fast access but limited in dynamic resizing.

**Analysis :**

 **Add**: O(1)

 **Search**: O(n)

 **Traverse**: O(n)

 **Delete**: O(n)

**Limitations:**

Arrays are not dynamic; for dynamic resizing, consider using ArrayList.

**Exercise 5: Task Management System**

Linked lists include Singly Linked List and Doubly Linked List. They allow dynamic memory allocation and efficient insertions/deletions.

**Analysis :**

 **Add**: O(n) (to add at the end)

 **Search**: O(n)

 **Traverse**: O(n)

 **Delete**: O(n)

**Optimization :**

For frequent insertions/deletions at the beginning, consider using a Doubly Linked List or a different data structure like Array List if random access is also needed.

**Exercise 6: Library Management System**

**Linear Search:**

* **Algorithm**: Sequentially checks each element until the target element is found or the list ends.
* **Time Complexity**: O(n)

**Binary Search:**

* **Algorithm**: Repeatedly divides a sorted array in half to locate the target element.
* **Time Complexity**: O(log n)

**Analysis :**

* **Linear Search**: O(n)
* **Binary Search**: O(log n)

**When to use:**

* **Linear Search**: Small or unsorted datasets.
* **Binary Search**: Large and sorted datasets.

**Exercise 7: Financial Forecasting**

**Recursion:**

* **Concept**: A function calls itself to solve a smaller instance of the same problem.
* **Advantages**: Simplifies the code for problems that have a natural recursive structure.
* **Disadvantages**: Can be less efficient due to repeated calculations and higher memory usage.

#### Analysis:

**Time Complexity**: O(n) (where n is the number of years)

**Optimizing Recursion:**

* **Memoization**: Store results of subproblems to avoid repeated calculations.
* **Dynamic Programming**: Use iterative approaches to improve efficiency.