**Exercise 5: Task Management System**

**Scenario:**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

**Steps:**

1. **Understand Linked Lists:**
   * Explain the different types of linked lists (Singly Linked List, Doubly Linked List).
2. **Setup:**
   * Create a class **Task** with attributes like **taskId**, **taskName**, and **status**.
3. **Implementation:**
   * Implement a singly linked list to manage tasks.
   * Implement methods to **add**, **search**, **traverse**, and **delete** tasks in the linked list.
4. **Analysis:**
   * Analyze the time complexity of each operation.
   * Discuss the advantages of linked lists over arrays for dynamic data.

**Explanation**

**Step 1: Understand Linked Lists**

**Types of Linked Lists:**

* **Singly Linked List:**
  + **Structure:** Each node contains data and a reference to the next node.
  + **Traversal:** Can only be traversed in one direction (forward).
  + **Memory Usage:** Requires less memory than doubly linked lists since each node only has one reference (to the next node).
  + **Operations:** Insertion and deletion are straightforward, especially at the beginning of the list.
* **Doubly Linked List:**
  + **Structure:** Each node contains data, a reference to the next node, and a reference to the previous node.
  + **Traversal:** Can be traversed in both directions (forward and backward).
  + **Memory Usage:** Requires more memory than singly linked lists since each node has two references (to the next and previous nodes).
  + **Operations:** Easier to insert and delete nodes at both ends and in the middle, since each node has a reference to its predecessor.

**Step 4: Analysis**

**Time Complexity of Each Operation:**

* **Add Task:** O(n) - Adding a task at the end of the list requires traversing the list to find the last node.
* **Search Task:** O(n) - Searching for a task involves traversing the list from the head to the tail until the task is found.
* **Traverse Tasks:** O(n) - Traversing the list to display all tasks involves visiting each node once.
* **Delete Task:** O(n) - Deleting a task involves searching for the task and then updating the references, requiring traversal of the list.

**Advantages of Linked Lists Over Arrays for Dynamic Data:**

* **Dynamic Size:** Linked lists can grow and shrink dynamically without needing to allocate or deallocate large blocks of memory.
* **Efficient Insertions/Deletions:** Inserting or deleting elements in a linked list is more efficient than in an array, especially when the operations are performed at the beginning or middle of the list. These operations do not require shifting elements, leading to better performance.
* **Memory Utilization:** Linked lists do not require contiguous memory allocation, which can help in situations where memory fragmentation is a concern.
* **Flexibility:** Linked lists are more flexible in terms of data structure modifications, allowing for easy implementation of other complex data structures like stacks, queues, and graphs.