**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:**

**(i) Explain the different types of linked lists (Singly Linked List, Doubly Linked List).**

**Answer:**

**Singly Linked List**

A Singly Linked List is a data structure in which each node has a reference to the next node in the sequence. This means that each node only points to the next node, and not to the previous node. This structure allows for efficient insertion and deletion of nodes at the beginning of the list, but can be less efficient for operations in the middle or end of the list. Singly Linked Lists are simple to implement and use less memory than other types of linked lists, but may not be suitable for applications that require frequent insertion or deletion of nodes in the middle of the list.

**Doubly Linked List**

A Doubly Linked List is a data structure in which each node has references to both the next node and the previous node in the sequence. This means that each node points to both the node before it and the node after it, allowing for efficient insertion and deletion of nodes at any position in the list. Doubly Linked Lists are more flexible than Singly Linked Lists and can be traversed in both forward and reverse directions, but they use more memory and are more complex to implement. They are suitable for applications that require frequent insertion or deletion of nodes in the middle of the list, or that need to traverse the list in both directions.

**4. Analysis:**

**(i) Analyze the time complexity of each operation.**

**Answer:**

* **addTask()**: O(n), where n is the number of tasks in the linked list, because we need to traverse the list to add a new task at the end.
* **searchTask()**: O(n), where n is the number of tasks in the linked list, because we need to traverse the list to find a task by taskId.
* **traverseTasks()**: O(n), where n is the number of tasks in the linked list, because we need to traverse the list to print all tasks.
* **deleteTask()**: O(n), where n is the number of tasks in the linked list, because we need to traverse the list to find and delete a task by taskId.

**(ii) Discuss the advantages of linked lists over arrays for dynamic data.**

**Answer:**

* Dynamic memory allocation: Linked lists can grow or shrink dynamically as tasks are added or deleted, whereas arrays have a fixed size.
* Efficient insertion and deletion: Linked lists can insert or delete tasks at any position in O(1) time, whereas arrays require shifting elements, which can be O(n) in the worst case.
* Good for sparse data: Linked lists are suitable for sparse data, where most tasks are empty or null, because they only allocate memory for non-null tasks.