Analysis

1. Analyze the time complexity of each operation.

Ans.

**Add Task**: Adding a new task to the end of the linked list.

Time Complexity:

Best Case: O(1) (if the list is empty)

Worst Case: O(n) (if the list has n elements)

**Search Task**: Searching for a task by taskId.

Time Complexity:

Best Case: O(1) (if the task is at the head)

Worst Case: O(n) (if the task is at the end or not present)

**Traverse Tasks**: Traversing and printing all tasks in the linked list.

Time Complexity: O(n) (as each node must be visited once)

**Delete Task:** Deleting a task by taskId.

Time Complexity:

Best Case: O(1) (if the task to delete is at the head)

Worst Case: O(n) (if the task is at the end or not present)

1. Discuss the advantages of linked lists over arrays for dynamic data.

Ans.

* 1. One of the most significant advantages of linked list over arrays is that linked lists can grow or shrink dynamically during runtime. This means that the size of a linked list can be adjusted to accommodate new elements or remove existing elements without having to allocate or deallocate a fixed-size block of memory, as is the case with arrays.
  2. Linked lists allow efficient insertion and deletion of elements at any position in the list, whereas arrays require shifting of elements when a new element is added or removed, which can be slow and inefficient for large arrays.

Thus linked list is more suitable than arrays when dealing with larger dataset which is handled dynamically.