Task Management System

1. Explain the different types of linked lists (Singly Linked List, Doubly Linked List)?

Ans. A singly linked list stores each node with data and a pointer to the next node, allowing one-way traversal from head to end. It uses less memory and is efficient for simple, sequential operations.

A doubly linked list stores each node with data, a pointer to the next node, and a pointer to the previous node, allowing two-way traversal. It uses more memory but makes insertions, deletions, and reverse traversals easier and more efficient.

1. Analyze the time complexity of each operation?

Ans. Adding a task at the end of a linked list takes O(n) time because it requires traversing to the last node, though this can be optimized to O(1) if a tail pointer is used.

Searching for a task involves a linear search through the list, resulting in O(n) time complexity.

Traversing all tasks takes O(n) as each node is visited once.

Deleting a task also requires a linear search to locate the task and then adjusting the pointers, making the operation O(n) as well.

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

Ans. Arrays have fixed size defined at creation, while linked lists offer dynamic sizing, growing or shrinking as needed. Insertions and deletions in arrays are inefficient (O(n)) due to the need for shifting elements, whereas linked lists handle insertions and deletions efficiently, typically O(1) at the head and O(n) at the end. In terms of memory, arrays use a contiguous block of memory, while linked lists allocate memory non-contiguously, linking scattered nodes dynamically. Linked lists are ideal when frequent insertions and deletions are required or when the exact size of the collection is unknown at runtime.