TASK MANAGEMENT SYSTEM

**1. Understanding Linked Lists**

**Types of Linked Lists**:

* **Singly Linked List**: Each node contains data and a reference (or link) to the next node in the sequence. Operations like traversal are straightforward, but accessing previous nodes is inefficient as there are no back links.
* **Doubly Linked List**: Each node contains data, a reference to the next node, and a reference to the previous node. This allows for efficient traversal in both directions and easier deletion of nodes, but it requires more memory for the additional references.

**4. Analysis**

**Time Complexity of Operations**:

* **Add**: O(n) - Adding a task requires traversing to the end of the list.
* **Search**: O(n) - Searching requires traversing the list until the task is found.
* **Traverse**: O(n) - Traversing involves visiting each node once.
* **Delete**: O(n) - Deleting a node requires traversing to find the node to delete and then updating the links.

**Advantages of Linked Lists Over Arrays**:

* **Dynamic Size**: Linked lists can grow and shrink dynamically, making them more flexible for managing collections of data where the size is not known in advance.
* **Efficient Insertions/Deletions**: Inserting or deleting nodes in a linked list (especially at the beginning) is more efficient compared to arrays, as it involves just updating pointers rather than shifting elements.
* **Memory Usage**: Linked lists use memory proportionally to the number of elements, whereas arrays may allocate more memory than needed if they are not fully utilized.

**Disadvantages**:

* **Access Time**: Linked lists have slower access times (O(n)) compared to arrays (O(1)), as elements must be accessed sequentially.
* **Memory Overhead**: Each node in a linked list requires extra memory for the reference(s), which can add up in large lists.
* **Cache Locality**: Arrays provide better cache performance due to contiguous memory allocation, whereas linked lists do not.

Linked lists are preferable when the collection size is dynamic and insertions/deletions are frequent. For static-size collections with frequent access operations, arrays might be a better choice.