Types of Linked Lists:

Singly Linked List:

Description: A linked list where each node points to the next node in the sequence. The last node points to null.

Structure:

Node -> Node -> Node -> null

Operations: Traversal is only possible in one direction (forward).

Doubly Linked List:

Description: A linked list where each node points to both the next and the previous node in the sequence.

Structure:

null <- Node <-> Node <-> Node -> null

Operations: Traversal is possible in both directions (forward and backward).

Analysis

Time Complexity of Operations:

Add Operation: O(n) - Adding an element to the end of the list requires traversing the list to find the last node.

Search Operation: O(n) - In the worst case, we need to search through the entire list.

Traverse Operation: O(n) - We need to visit each node in the list.

Delete Operation: O(n) - In the worst case, we need to traverse the list to find the node to be deleted.

Advantages of Linked Lists Over Arrays for Dynamic Data

Dynamic Size: Linked lists can grow and shrink in size dynamically. Unlike arrays, there is no need to define the size upfront, making linked lists more flexible for dynamic datasets.

Efficient Insertions/Deletions: Insertions and deletions can be done more efficiently, especially in the middle of the list. While adding or removing elements in arrays requires shifting elements, linked lists simply involve adjusting pointers.

Memory Usage: Linked lists can be more memory efficient if the number of elements is unpredictable. They do not have the overhead of reserved but unused space, as arrays do.

No Fixed Size Limit: Linked lists do not have a fixed size, so they can accommodate an arbitrary number of elements without the need to resize, as long as memory is available.

While arrays offer fast access times due to direct indexing, linked lists provide flexibility and efficiency in handling dynamic datasets with frequent insertions and deletions. For a task management system where tasks need to be added, deleted, and traversed efficiently, a singly linked list is a suitable choice.