**Singly Linked List:**

* Each node contains a data element and a reference (or pointer) to the next node in the sequence. The list starts with a head node and ends with a node that points to null.
* Easy to add and remove nodes at the beginning or in the middle but requires traversal to access or modify nodes at the end.
* Simple structure, uses less memory per node compared to a doubly linked list.

**Doubly Linked List:**

* Each node contains a data element and two references: one to the next node and one to the previous node. It allows traversal in both directions.
* Efficient to add and remove nodes from both ends or in the middle, as nodes can be accessed in both directions.
* Allows bidirectional traversal and easier deletion of nodes if you have a reference to the node.

**Time Complexity Analysis:**

* **Add Task: O(1)**
* **Search Task by ID: O(n)**
* **Traverse Tasks: O(n)**
* **Delete Task : O(n)**

**Advantages of Linked Lists Over Arrays:**

* Linked lists can grow and shrink dynamically, unlike arrays that require predefined sizes.
* Adding or removing elements does not require shifting other elements, making operations more efficient, especially in the middle of the list.
* No need to allocate extra space as in arrays; linked lists use only as much memory as needed for storing elements.