**Types of Linked Lists**

1. **Singly Linked List:**
   * **Definition:** A singly linked list consists of nodes where each node contains data and a reference (link) to the next node in the sequence.
   * **Structure:**
     + data: Stores the actual data (e.g., task details).
     + next: A reference to the next node in the list.
   * **Operations:**
     + Insertion: Can be done at the beginning, end, or middle.
     + Deletion: Requires traversal to find the node to be deleted.
     + Traversal: Involves iterating from the head node to the end.
2. **Doubly Linked List:**
   * **Definition:** A doubly linked list consists of nodes where each node has references to both the next and previous nodes.
   * **Structure:**
     + data: Stores the actual data.
     + next: A reference to the next node.
     + prev: A reference to the previous node.
   * **Operations:** Similar to singly linked lists but allows traversal in both directions.

**Advantages of Linked Lists**

* **Dynamic Size:** Linked lists can grow or shrink dynamically, unlike arrays with fixed sizes.
* **Efficient Insertions/Deletions:** Adding or removing nodes does not require shifting elements, as in arrays.

**Disadvantages of Linked Lists**

* **Extra Memory:** Requires additional memory for storing references (pointers).
* **Sequential Access:** Cannot access elements directly by index; traversal is necessary.

**Time Complexity Analysis**

1. **Add Task:**
   * Best Case: O(1) – When adding to an empty list or as the new tail node.
   * Worst Case: O(n) – When traversing the entire list to add at the end.
2. **Search Task by ID:**
   * Best Case: O(1) – When the task is the head node.
   * Worst Case: O(n) – When the task is at the end or not present.
3. **Traverse Tasks:**
   * Time Complexity: O(n) – You must visit each node in the list.
4. **Delete Task by ID:**
   * Best Case: O(1) – When the task is the head node.
   * Worst Case: O(n) – When the task is at the end or not present.

**When to Use Linked Lists**

* **Dynamic Size:** When the number of elements is not known in advance or changes frequently.
* **Frequent Insertions/Deletions:** Linked lists are suitable when insertions and deletions occur more frequently than access operations.