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

Singly Linked List: This is the simplest type of linked list where each node contains data and a reference (or pointer) to the next node in the sequence. The last node points to null, indicating the end of the list. Singly linked lists allow traversal in only one direction1**.**

Doubly Linked List: In this type, each node contains data and two references: one to the next node and one to the previous node. This allows traversal in both directions (forward and backward). The first node's previous reference and the last node's next reference point to null1.

Analyze the time complexity of each operation.

- **Add Task**: O(n) - In the worst case, we need to traverse to the end of the list to add a new task.
- **Search Task**: O(n) - In the worst case, we need to traverse the entire list to find the task.
- **Traverse Tasks**: O(n) - We need to visit each node once.
- **Delete Task**: O(n) - In the worst case, we need to traverse the entire list to find and delete the task.

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

- **Dynamic Size**: Linked lists can grow and shrink dynamically, making them more flexible than arrays, which have a fixed size.
- **Efficient Insertions/Deletions**: Inserting or deleting elements in a linked list is more efficient than in an array, especially for large datasets, as it does not require shifting elements.
- **Memory Utilization**: Linked lists do not require contiguous memory allocation, which can be advantageous when dealing with large datasets or limited memory.