**Understand Linked Lists**

**Types of Linked Lists**

**Singly Linked List**:

* Each node contains a data part and a reference to the next node in the sequence.
* Operations like insertion and deletion can be efficiently performed, especially at the beginning of the list.

**Doubly Linked List**:

* Each node contains a data part, a reference to the next node, and a reference to the previous node.
* Allows traversal in both forward and backward directions, making it more flexible than singly linked lists.
* Insertion and deletion operations can be performed more efficiently, especially in the middle of the list.

**Analysis**

**Time Complexity of Each Operation**

* **Add Task**:
  + **Time Complexity**: O(n)O(n)O(n) - Adding a task at the end requires traversing the list to find the last node. If adding at the beginning, it is O(1)O(1)O(1).
* **Search Task**:
  + **Time Complexity**: O(n)O(n)O(n) - Linear search through the list is required to find the task.
* **Traverse Tasks**:
  + **Time Complexity**: O(n)O(n)O(n) - Each node is visited once during traversal.
* **Delete Task**:
  + **Time Complexity**: O(n)O(n)O(n) - Linear search through the list to find the task and then rearrange the pointers.

**Advantages of Linked Lists Over Arrays for Dynamic Data**

* **Dynamic Size**: Linked lists can grow or shrink dynamically, which is not possible with arrays that have a fixed size.
* **Efficient Insertions/Deletions**: Insertions and deletions in linked lists are more efficient, especially when operations are performed at the beginning or middle of the list, as they do not require shifting elements like arrays.
* **Memory Utilization**: Linked lists utilize memory more efficiently as they do not require pre-allocation of memory like arrays.

**Disadvantages of Linked Lists**

* **Access Time**: Linked lists have slower access time compared to arrays due to the need for sequential traversal.
* **Memory Overhead**: Linked lists use extra memory for storing references/pointers, which can be significant for large lists.