**Linked Lists**

**Types of Linked Lists**

1. **Singly Linked List**:
   * **Structure**: Each node contains data and a reference (or pointer) to the next node in the sequence.
   * **Traversal**: Can only be traversed in one direction, from the head to the last node.
   * **Operations**: Easy to add and remove nodes, but searching requires traversing from the head to the desired node.
2. **Doubly Linked List**:
   * **Structure**: Each node contains data, a reference to the next node, and a reference to the previous node.
   * **Traversal**: Can be traversed in both directions, forward and backward.
   * **Operations**: More flexible and efficient for certain operations like deletion but requires more memory due to the additional pointer.

**Analysis**

**Time Complexity**

* **Add Task:**
  + Time Complexity: O(n) (since we need to traverse to the end of the list).
* **Search Task:**
  + Time Complexity: O(n) (since we may need to traverse the entire list to find the task).
* **Traverse Tasks:**
  + Time Complexity: O(n) (since we need to visit each node).
* **Delete Task:**
  + Time Complexity: O(n) (since we may need to traverse the list to find the task).

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

* *Dynamic Size*: Linked lists can easily grow and shrink in size by adding or removing nodes, whereas arrays have a fixed size.
* *Efficient Insertions/Deletions*: Insertions and deletions in linked lists are more efficient compared to arrays, especially when dealing with large data, as they don't require shifting elements.
* *Memory Usage*: Linked lists use memory for pointers, but they do not need to allocate memory for unused elements as arrays do.