**Understand Linked Lists**

**Linked Lists Overview:**

* **Singly Linked List:** Each node contains data and a reference to the next node in the sequence. The last node points to null.
* **Doubly Linked List:** Each node contains data, a reference to the next node, and a reference to the previous node. This allows traversal in both directions.

**Advantages of Linked Lists over Arrays:**

* **Dynamic Size:** Linked lists can grow and shrink dynamically as elements are added or removed.
* **Efficient Insertions/Deletions:** Insertions and deletions can be done in O(1) time if the node reference is known, unlike arrays that may require shifting elements.

**Analysis**

**Time Complexity Analysis:**

1. **Add Task:**
   * **Time Complexity:** O(n) (traverse to the end of the list to add the new task)
2. **Search Task:**
   * **Time Complexity:** O(n) (linear search through the list)
3. **Traverse Tasks:**
   * **Time Complexity:** O(n) (visit each node in the list)
4. **Delete Task:**
   * **Time Complexity:** O(n) (linear search to find the node and update links)

**Advantages of Linked Lists over Arrays:**

* **Dynamic Size:** Linked lists can grow and shrink dynamically as elements are added or removed.
* **Efficient Insertions/Deletions:** Insertions and deletions can be done efficiently without shifting elements.
* **Memory Utilization:** Memory is allocated as needed for each node, avoiding the need to allocate memory for a fixed-size array.