**EX 5**

**Understanding 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.
   * **Advantages**: Simplicity, less memory per node compared to doubly linked lists.
   * **Disadvantages**: Cannot traverse backward, requires additional traversal to find the previous node for deletions.
2. **Doubly Linked List**
   * **Structure**: Each node contains data, a reference to the next node, and a reference to the previous node.
   * **Advantages**: Can be traversed in both directions, easier deletions and insertions at both ends.
   * **Disadvantages**: More memory per node due to extra reference, more complex to implement.

**Analysis**

**Time Complexity of Operations**

1. **Add Task**
   * **Time Complexity**: O(n) (worst case)
   * **Explanation**: Traversing the entire list to add a new task at the end. In the best case, adding to an empty list is O(1).
2. **Search Task**
   * **Time Complexity**: O(n)
   * **Explanation**: In the worst case, we need to traverse the entire list to find the task.
3. **Traverse Tasks**
   * **Time Complexity**: O(n)
   * **Explanation**: Traversing the list involves visiting each node once.
4. **Delete Task**
   * **Time Complexity**: O(n)
   * **Explanation**: In the worst case, we need to traverse the entire list to find the task to be deleted and update links.

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

1. **Dynamic Size**: Linked lists can grow and shrink dynamically without the need for resizing, unlike arrays which have a fixed size.
2. **Efficient Insertions/Deletions**: Insertions and deletions (especially at the beginning or end) are more efficient and don't require shifting elements like in arrays.
3. **Memory Utilization**: Linked lists can utilize memory more efficiently as they do not need to allocate a large contiguous block of memory.
4. **Flexibility**: Linked lists provide more flexibility in terms of memory allocation and deallocation, making them suitable for dynamic datasets.

However, linked lists also have some disadvantages compared to arrays:

1. **Memory Overhead**: Each node in a linked list requires additional memory for storing pointers.
2. **Cache Performance**: Due to non-contiguous memory allocation, linked lists can have poorer cache performance compared to arrays.
3. **Access Time**: Linked lists have linear access time (O(n)) compared to constant time (O(1) for arrays when accessing elements by index.