**Exercise 5: Task Management System**

**Linked Lists** are data structures where each element (node) points to the next. They are used to implement dynamic memory allocation. The two main types are:

* **Singly Linked List**: Each node points to the next node, and the last node points to null.
* **Doubly Linked List**: Each node points to both the next and previous nodes.

**Advantages**:

* Dynamic Size: Easily grows or shrinks as needed.
* Efficient Insertion/Deletion: Operations are more efficient compared to arrays when adding or removing elements.

**Disadvantages**:

* Extra Memory: Each node requires additional memory for the pointer.
* No Random Access: Accessing elements is slower compared to arrays.

**Implementation:**

Please refer the code.

**Analysis of Time Complexity**

1. **Add Operation**:
   * **Time Complexity**: O(n)O(n)O(n) - Finding the end of the list requires traversing it, which is linear in complexity.
2. **Search Operation**:
   * **Time Complexity**: O(n)O(n)O(n) - Requires a linear search through the list.
3. **Traverse Operation**:
   * **Time Complexity**: O(n)O(n)O(n) - Requires visiting each node in the list.
4. **Delete Operation**:
   * **Time Complexity**: O(n)O(n)O(n) - Finding the task requires linear search; removing it involves bypassing the node.

**Advantages of Linked Lists Over Arrays**

1. **Dynamic Size**: Linked lists can grow and shrink dynamically, unlike arrays that require a fixed size.
2. **Efficient Insertion/Deletion**: Insertion and deletion are more efficient in linked lists as they do not require shifting elements.
3. **Memory Utilization**: Linked lists use memory more efficiently for applications where the size of data changes frequently.

**When to Use Linked Lists**:

* **Dynamic Data**: When the number of elements is unknown or changes frequently.
* **Frequent Insertions/Deletions**: When you need to add or remove elements frequently, especially in the middle of the collection.

Linked lists offer advantages in specific scenarios, but for applications where random access or fixed-size collections are required, arrays or other data structures like ‘ArrayList’ might be preferable.

Top of Form

Bottom of Form