**Step 1: Understand Linked Lists**

1. Explain the different types of linked lists (Singly Linked List, Doubly Linked List).

A **linked list** is a dynamic data structure made up of **nodes**, where each node contains:

* The data (e.g., a task)
* A reference (link) to the next node

Types of Linked Lists:

| Type | Structure | Use Case |
| --- | --- | --- |
| **Singly Linked List** | Each node points to the **next node only** | Simple insertion, traversal |
| **Doubly Linked List** | Each node points to both **next and previous nodes** | Bidirectional traversal, deletion from both ends |

### Linked Lists are

* **Dynamic size**: No need to define max capacity
* **Efficient insert/delete**: No shifting like arrays
* **Flexible structure**: Great for apps where size changes often

**Step 4: Analysis**

1. Analyze the time complexity of each operation.

Time Complexity (Singly Linked List):

| Operation | Description | Time |
| --- | --- | --- |
| Add | Traverse to end, then add | O(n) |
| Search | Linear scan | O(n) |
| Traverse | Print all tasks | O(n) |
| Delete | Find and unlink node | O(n) |

II. Discuss the advantages of linked lists over arrays for dynamic data.

Advantages of Linked Lists Over Arrays:

| Feature | Array | Linked List |
| --- | --- | --- |
| Size | Fixed | Dynamic (grows/shrinks freely) |
| Add/Delete (middle) | Slow (shifting needed) | Fast (just update links) |
| Memory | Pre-allocated | Uses only needed memory |
| Access by index | Fast (O(1)) | Slow (O(n)) |

**When to use linked lists:**

* Data changes frequently
* One don’t know size in advance
* Frequent insertions/deletions