Linked List – Introduction

1. Introduction

A **Linked List** is a **linear data structure** in which elements are stored in **non-contiguous memory locations**.

Each element of a linked list is called a **node**, and every node contains:

- **Data**
- **A reference (link) to the next node**

Linked lists overcome many limitations of arrays, especially **fixed size** and **memory wastage**.

---

2. What is a Linked List?

A linked list is a collection of nodes where:

- Each node stores data and a pointer
- Nodes are connected using pointers
- Elements are accessed sequentially

Unlike arrays, linked lists **do not require contiguous memory allocation**.

---

3. Structure of a Node

Each node has two parts:

- **Data** → Stores the value
- **Next** → Stores address of the next node

---

4. Head Pointer

- The **Head** is a reference to the **first node** of the linked list
- If the list is empty, head points to **NULL**

```
Head → [Data | Next] → [Data | Next] → NULL
```

---

5. How Linked List Works (Plain English)

1. Memory is allocated dynamically for each node
2. Each node stores data and link to the next node
3. Nodes are connected through pointers
4. Traversal starts from the head
5. Access is sequential, not random

---

6. Why Linked List is Needed?

Problems with arrays:

- Fixed size
- Memory wastage
- Insertion and deletion are costly

Linked list solves these problems by:

- Allowing dynamic memory allocation
- Making insertion and deletion efficient
- Using memory as required

---

7. Types of Linked Lists

1. **Singly Linked List**
2. **Doubly Linked List**

3. **Circular Linked List**
4. **Circular Doubly Linked List**

(Each type differs in how nodes are connected.)

---

8. Comparison: Array vs Linked List

Feature	Array	Linked List
Memory Allocation	Contiguous	Non-contiguous
Size	Fixed	Dynamic
Insertion/Deletion	Costly	Efficient
Access	Random	Sequential
Memory Usage	May waste space	Efficient

---

9. Advantages of Linked List

- Dynamic size
- Efficient insertion and deletion
- No memory wastage
- Easy memory utilization
- Flexible structure

---

10. Limitations of Linked List

- Extra memory for pointers
- No random access
- Slower traversal
- More complex implementation than arrays

---

11. Applications of Linked List

- Implementation of stacks and queues
- Dynamic memory allocation
- Polynomial manipulation
- Undo/redo operations
- Browser navigation
- Music playlist management

---

12. Real-World Example

A **train**:

- Each coach is a node
- Coaches are linked to the next coach
- Coaches can be added or removed easily

---

13. Summary

- Linked list is a dynamic linear data structure
- Nodes store data and links
- Memory is allocated dynamically
- Efficient for insertions and deletions
- Traversal is sequential
