Linked List

By Mani Abedii

View Full DSA Repository

A clear, compact guide covering essential data structures and algorithms, designed for easy understanding and quick learning.

A **linked list** is a linear data structure where each element (called a node) contains data and a reference (or pointer) to the next node in the sequence. Unlike arrays, linked lists do not store elements in contiguous memory locations.

Linked lists are sometimes preferred over arrays because:

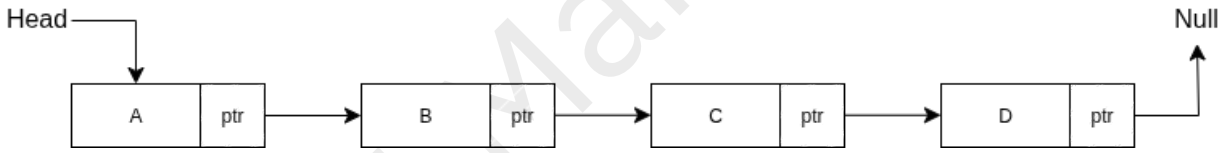
- **Dynamic Size:** They can grow or shrink at runtime without reallocating memory.
- **Efficient Insertions/Deletions:** Adding or removing elements in the middle or beginning does not require shifting elements.
- **Memory Utilization:** You only allocate memory for what you need, instead of pre-allocating a large array.

However:

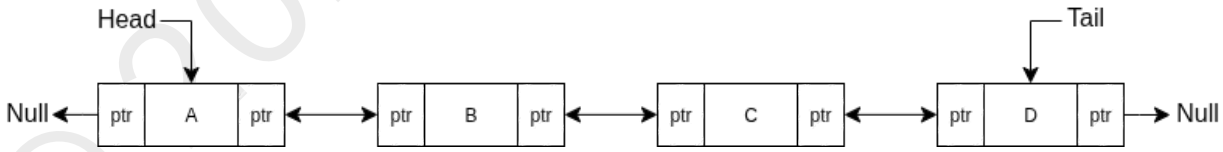
- Accessing an element by index is slower O(n), because you have to traverse the list.
- They require extra memory to store pointers alongside data.

Types of Linked Lists

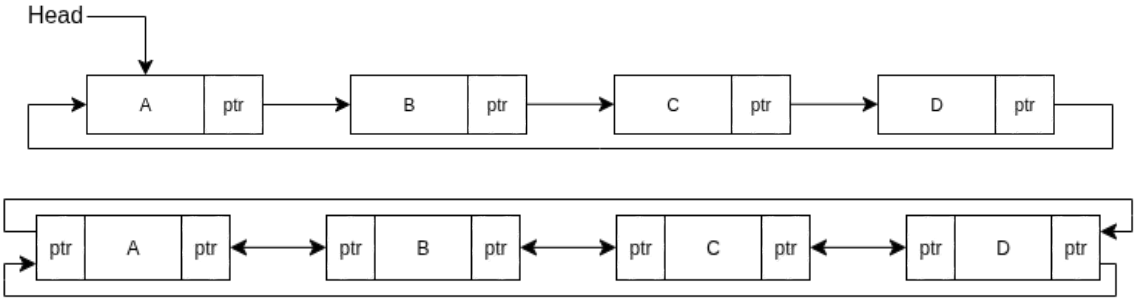
1. **Singly Linked List (LL):** Each node points to the next node only. Traversal is forward only. It is simple and memory-efficient.



2. **Doubly Linked List (DLL):** Each node points to both the next and previous nodes. Traversal can be forward and backward. Insertion and deletion are easier compared to singly linked lists at any position.



3. **Circular Linked List (CLL):** The last node points back to the head. It can be singly circular or doubly circular.



We use linked lists when:

- The number of elements is unknown or changes frequently.
- Frequent insertions and deletions are required.
- Memory fragmentation is not a major concern.