**Types of Linked Lists:**

A **linked list** is a linear data structure where elements (called nodes) are connected using pointers. Each node typically contains data and a reference to the next (or previous) node. Unlike arrays, linked lists don’t store elements in continuous memory, which makes them more flexible in certain situations.

There are mainly two types:

* **Singly Linked List:** Each node has two parts—data and a pointer to the next node. It moves in one direction only, from the head (first node) to the end (null). It is simple and uses less memory than a doubly linked list.
* **Doubly Linked List:** Each node contains three parts—data, a pointer to the next node, and a pointer to the previous node. It can be traversed both forwards and backwards, which is useful for certain operations but takes more memory.

**Time Complexity of Linked List Operations:**

* **Insertion:** O(1) at the beginning (for both singly and doubly linked lists). Inserting at the end or at a specific position is O(n) since we may need to traverse the list.
* **Deletion:** O(1) if the node is known (in doubly linked list), but O(n) if we need to search first.
* **Search:** O(n) since we may need to look through each node until we find the target.
* **Traversal:** O(n), as we must visit every node one by one.

So while some operations like adding/removing at the beginning are fast, others like searching are slower than arrays.

**Advantages of Linked Lists Over Arrays:**

One major advantage of linked lists over arrays is **dynamic memory allocation**. Arrays have a fixed size, so we need to know in advance how much space we’ll need. In contrast, linked lists can **grow or shrink in size** during runtime, which makes them ideal for situations where the number of elements keeps changing.

Also, **inserting or deleting elements** in a linked list is more efficient, especially at the beginning or middle, because we don't have to shift elements like in arrays. Linked lists are also better suited for implementing complex data structures like stacks, queues, and graphs.

However, since linked lists use extra memory for pointers and don’t allow direct access using indexes, they’re not always the best choice—especially when fast access or simple memory layout is more important.