**DAILY DSA | DAY-11 | Linked lists – Part 2|**
**-GOPALKRISHNA A**

We will continue linked lists to calculate memory usage, operations on linked lists, advantages & disadvantages.

**Calculating memory usage**
The memory usage of a linked list is slightly more complex than an array. In addition to the data we are storing in each node, we are also storing the pointers between nodes: a singly-linked list has one pointer for every node and a doubly-linked list has two pointers. Typically a memory pointer today is an 8-byte memory address (for software running on 6-bit machines)

Calculation on the memory usage of a doubly-linked list that contains one million 32-bit integers?
10^6 * (32 bits + 64 bits * 2) = 10^6 * 20 bytes = 20MB

**Linked list operations (45 - 50 mins) - Must read article:**

- I have referred the blog from linked lists this article is pretty much well explained and on point

  https://www.geeksforgeeks.org/python-linked-list/

**Advantages of linked lists:**

1. Because of the chain-like system of linked lists, we **can add and remove elements quickly.** This also doesn't require reorganizing the data structures, unlike arrays or lists. Linear data structures are often easier to implement using linked lists.
2. Linked lists don't need a fixed size or initial size for the chain-like structure.

**Disadvantages of a linked list:**

1. **More memory** is required when compared to an array. This is because we need a pointer (which takes up its own memory) to point to the next element.
2. **Search operations on a linked list are very slow.** Unlike an array, we don't have the option of random access

**When to use a linked list:**

1. When we need constant time insertion/deletion from the list (unlike an array, we don't have to shift every other item in the list first)
2. When we need to insert items in the middle of the list
3. When we don't need random access to any elements (unlike an array, we cannot access an element at a particular index in a linked list)