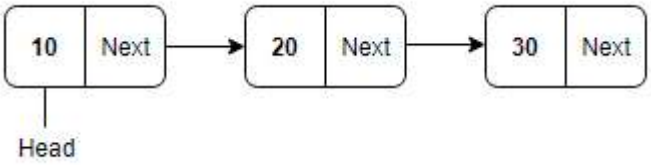
Lecture 3 -Linked List (1)

Linked List is a linear data structure which consists of a group of nodes in a sequence. Each node contains two parts.

- Data− Each node of a linked list can store a data.
- Address − Each node of a linked list contains an address to the next node, called "Next".

The first node of a Linked List is referenced by a pointer called Head

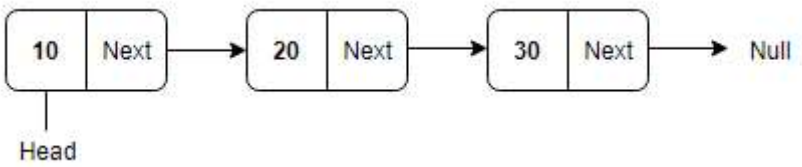


Advantages of Linked List

- They are dynamic in nature and allocate memory as and when required.
- Insertion and deletion is easy to implement.
- Other data structures such as Stack and Queue can also be implemented easily using Linked List.
- It has faster access time and can be expanded in constant time without memory overhead.
- Since there is no need to define an initial size for a linked list, hence memory utilization is effective.
- Backtracking is possible in doubly linked lists.

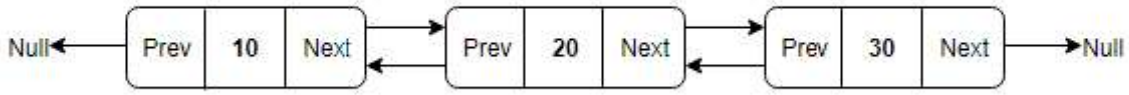
Types of Linked List

- **Singly Linked List:** Singly linked lists contain nodes which have a data part and an address part, i.e., Next, which points to the next node in the sequence of nodes. The next pointer of the last node will point to null.

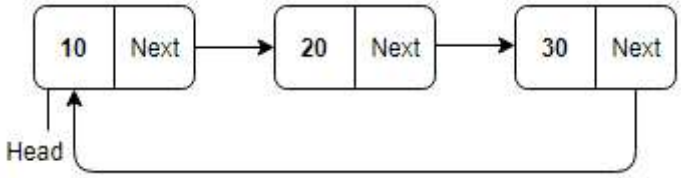


- **Doubly Linked List:** In a doubly linked list, each node contains two links - the first link points to the previous node and the next link points to the next

node in the sequence. The previous pointer of the first node and next pointer of the last node will point to null.



- **Circular Linked List:** In the circular linked list, the next of the last node will point to the first node, thus forming a circular chain.



- **Doubly Circular Linked List:** In this type of linked list, the next of the last node will point to the first node and the previous pointer of the first node will point to the last node.

➢ Linked List vs Array

ARRAY	LINKED LISTS
1. Arrays are stored in contiguous location.	1. Linked lists are not stored in contiguous location.
2. Fixed in size.	2. Dynamic in size.
3. Memory is allocated at compile time.	3. Memory is allocated at run time.
4. Uses less memory than linked lists.	4. Uses more memory because it stores both data and the address of next node.
5. Elements can be accessed easily.	5. Element accessing requires the traversal of whole linked list.
6. Insertion and deletion operation takes time.	6. Insertion and deletion operation is faster.

➢ *Below is the structure of the singly linked list*

```
// Node of a doubly linked list
class Node {
public:
        int data;

        // Pointer to next node in LL
        Node* next;
};
```

Structure of Doubly Linked List:

```
// Node of a doubly linked list
struct Node {
        int data;

        // Pointer to next node in DLL
        struct Node* next;
```

```
    // Pointer to the previous node in DLL
    struct Node* prev;
};
```

- ***Below is the structure of the Circular Linked List:***

```
// Structure for a node
class Node {
public:
    int data;

    // Pointer to next node in CLL
    Node* next;
};
```

- ***Below is the structure of the Doubly Circular Linked List:***

```
// Node of doubly circular linked list
struct Node {

    int data;

    // Pointer to next node in DCLL
    struct Node* next;

    // Pointer to the previous node in DCLL
    struct Node* prev;
};
```