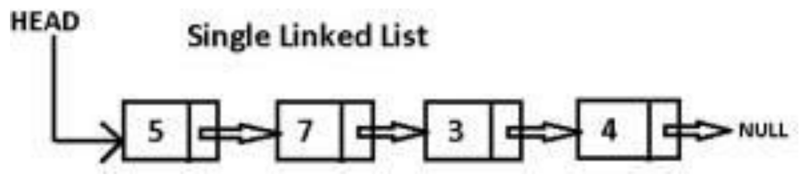
LINKED LISTS

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.



Representation of structure

```
struct node
{
 int info;
 struct node *ptr;
};
```

While declaring the structure for a linked list

Declare a structure with two members is there i.e data member and next pointer member. The data member can be character or integer or depends upon the type of the information that the linked list is having. The link member contains the address of next node.

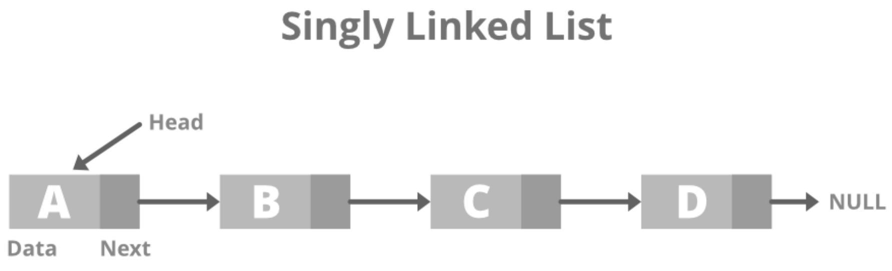
Types of linked list

1. Singly linked list (SLL)
2. Circular singly linked list (CSLL)
3. Doubly linked list (DLL)
4. Circular doubly linked list (CDLL)

Singly linked list (SLL)

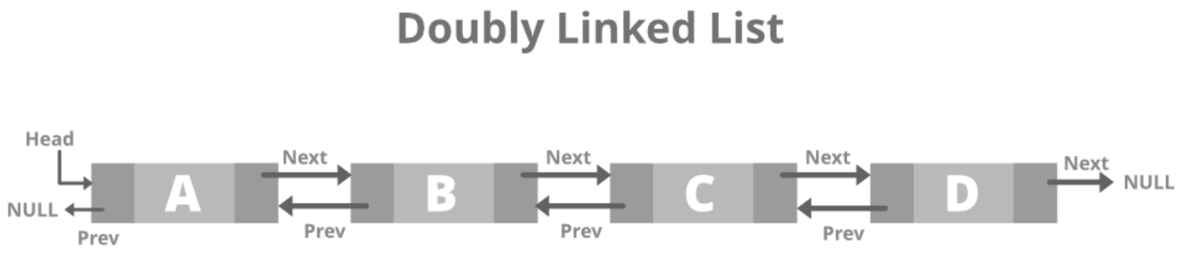
It is the simplest type of linked list in which every node contains some data and a pointer to the next node of the same data type.

The node contains a pointer to the next node means that the node stores the address of the next node in the sequence. A single linked list allows the traversal of data only in one way. Below is the image for the same:



Doubly linked list (SLL)

A doubly linked list or a two-way linked list is a more complex type of linked list that contains a pointer to the next as well as the previous node in sequence.
Therefore, it contains three parts of data, a pointer to the next node, and a pointer to the previous node. This would enable us to traverse the list in the backward direction as well. Below is the image for the same:

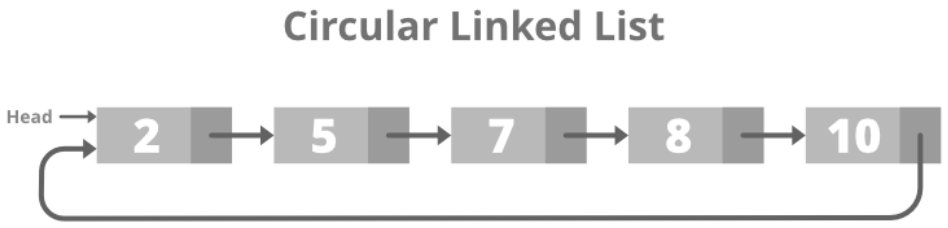


Advantages of DLL over the singly linked list:

- A DLL can be traversed in both forward and backward directions.
- The delete operation in DLL is more efficient if a pointer to the node to be deleted is given.
- We can quickly insert a new node before a given node.
- In a singly linked list, to delete a node, a pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using the previous pointer.

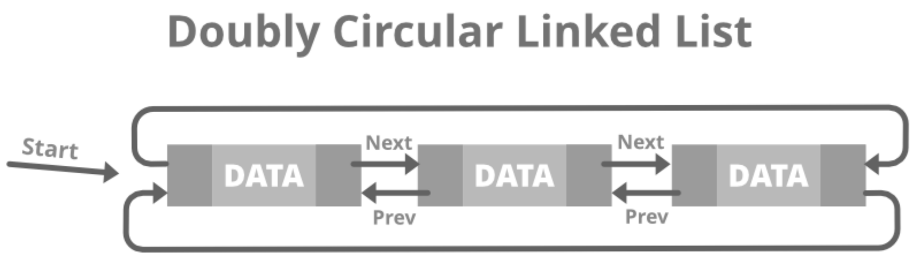
Circular Singly linked list (CSLL)

A circular linked list is that in which the last node contains the pointer to the first node of the list. While traversing a circular linked list, we can begin at any node and traverse the list in any direction forward and backward until we reach the same node we started. Thus, a circular linked list has no beginning and no end. Below is the image for the same:



Circular Doubly linked list (CDLL)

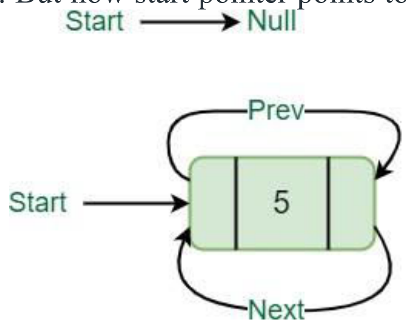
A Doubly Circular linked list or a circular two-way linked list is a more complex type of linked list that contains a pointer to the next as well as the previous node in the sequence. The difference between the doubly linked and circular doubly list is the same as that between a singly linked list and a circular linked list. The circular doubly linked list does not contain null in the previous field of the first node. Below is the image for the same:



Insertion in Circular Doubly Linked List:

1. Insertion at the end of the list or in an empty list:

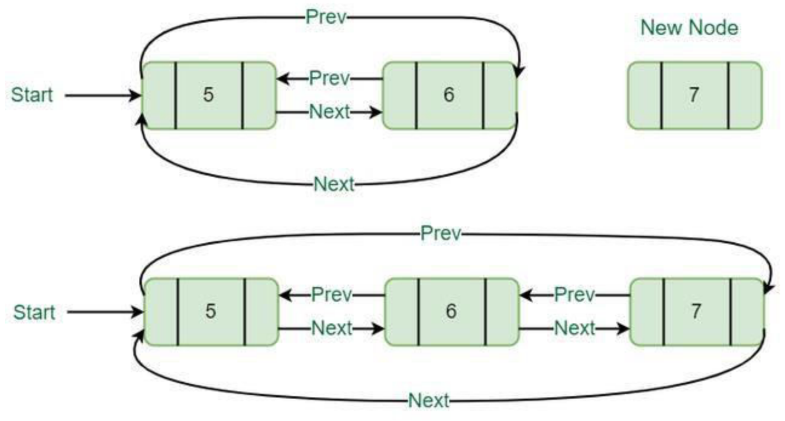
A node(Say **N**) is inserted with **data = 5**. So, the previous pointer of N points to N and the next pointer of N also points to N. But now start pointer points to the first node of the list.



Insertion in an empty list

2. List initially contains some nodes, start points to the first node of the List:

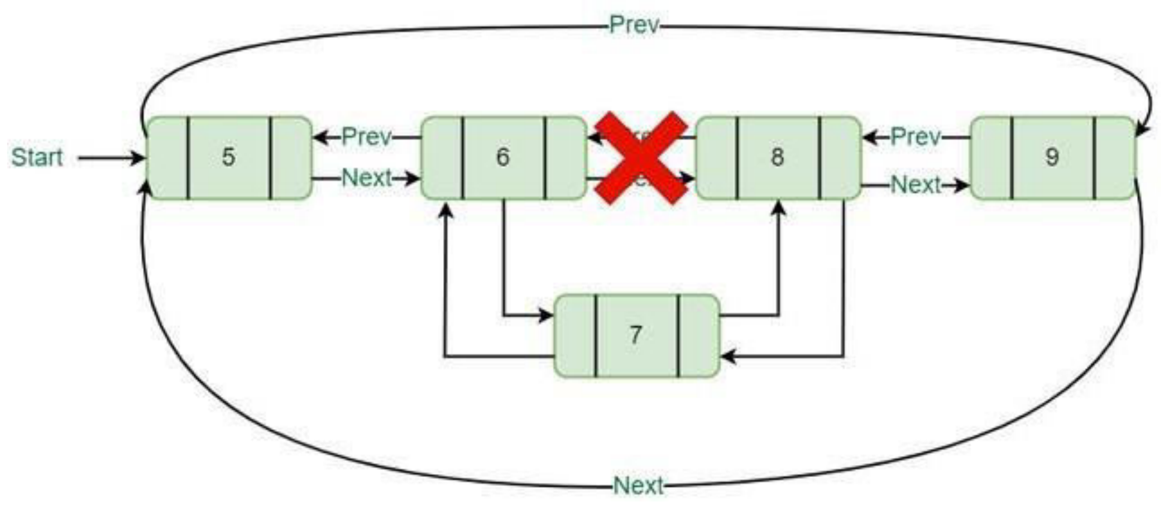
A node(Say M) is inserted with data = 7, so the previous pointer of M points to the last node, the next pointer of M points to the first node and the last node's next pointer points to this M node, and first node's previous pointer points to this M node.



Insertion at the end of list

4. Insertion in between the nodes of the list:

To insert a node in between the list, two data values are required one after which new node will be inserted and another is the data of the new node.



Advantages of circular doubly linked list:

- The list can be traversed from both directions i.e. from head to tail or from tail to head.
- Jumping from head to tail or from tail to head is done in constant time O(1).
- Circular Doubly Linked Lists are used for the implementation of advanced data structures like the Fibonacci Heap.

Common Operations on a linked list:

1. Insert to node at the end of linked list
2. Insert to node at the front of linked list
3. Insert to node at the specified position in the linked list
4. Delete the node from the end of linked list
5. Delete the node from the beginning of linked list
6. Delete the node at the specified position from the linkedlist
7. 7. Search the node in linked list
8. Implement a linked list based on stack principle
9. Implement a linked list based on queue principle
10. Count the number of nodes in the linked list.