UNIT-4:-

Linked Lists

A linked list is a sequence of data structures, which are connected together via links. Linked list is the second most-used data structure after array.

Important terms in Linked List:-

Link:- Each link of a linked list can store a data called an element.

Next:- Each link of a linked list contains a link to the next link called Next.

Linked List Representation:-

Linked list can be visualized as a chain of nodes.

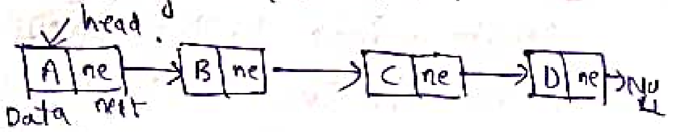
A linked list is represented by a pointer to the ~~linked list~~ .first node of the linked list. The first node is called the head. If the linked list is empty, then the value of the head is NULL.

Each node consist at least two parts.

(i) data (ii) Pointer.

We can represent a node using structures.

ex:-
```
struct node
  {
    int data;
    struct node* next;
  };
```



Types of Linked List:-

The various types of linked list are

(i) Single Linked list - Item navigation is forward only.

(ii) Doubly Linked list - Items can be navigated forward and backward.

(iii) Circular Linked list - Last item contains link to the first element and the first element has a link to the last element.

Basic Operations:-

Insertion :- Adds an element at the begining of the list.

Deletion :- Deletes an element at the begining of the list

Display :- Displays the complete list.

Search :- Searches an element using the given key.

Delete :- Deletes an element using the given key.

Why Linked List :-

Arrays can be used to store linear data of similar types. But arrays have the following limitations,

(i) The size of the array is fixed.

(ii) Inserting a new element in an array of elements is expensive.

Drawbacks of Single Linked List :-

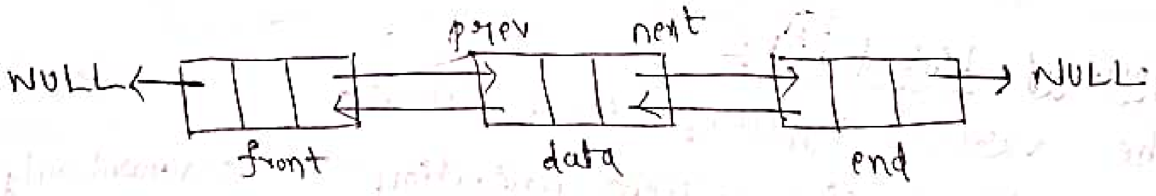
(i) Random access is not allowed. We have to access elements sequentially.

(ii) We can not do binary search with linked lists efficiently.

(iii) Extra a memory space for a pointer is required.

Doubly Linked List :-

In doubly Linked list each node has two links. The first points to the previous node and second link points to the next node. The first node of the list has it's previous link putting to NULL. The last node of the list has its next link putting to NULL.



The two links helps us to traverse the list in both backward and forward direction.

Basic Operations:-

add-front : Adds a new node in the beginning of th list.

add-after : Adds a new node after another node

add-before : Adds a new node before another node.

add-end : Adds new node in the end of the list.

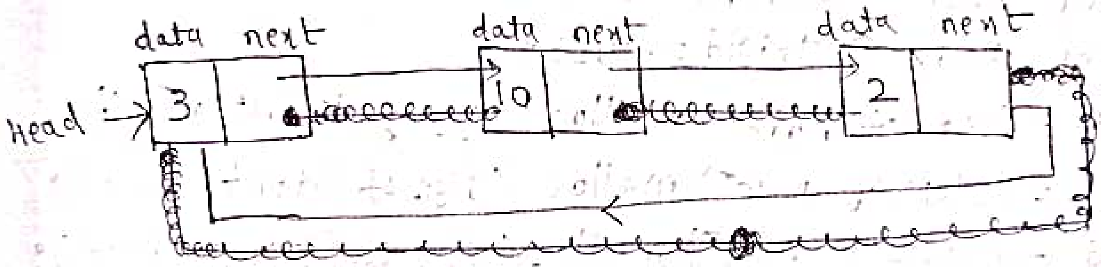
delete : Removes the node.

forward-traverse : Traverse the list in forward direction

backward-traverse : Traverse the list in backward direction

Circular Linked List:

Circular linked list is a little more complicated linked data structure. In the circular linked list we can insert elements anywhere in the list. In the circular linked list the previous element stores the address of the next element and the last element stores the address of the starting element.



Applications:-

(i) It is used in our personal computers, where multiple applications running.

(ii) Multiplayer Games like Ludo.

(iii) To create circular Queue.

Implementation:-

In circular linked list the last node will have it's next point to the Head of the list.

Circular Single linked list } two types of circular
Circular doubly linked list } linked lists

Drawbacks of Single Linked List:-

(i) We can navigate only in forward direction.
(ii) Searching is very difficult.
(iii) We require extra memory for pointer.
(iv) Random access is very difficult

Single Linked List Programme :-

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  struct node* next;
};
void main()
{
 struct node * first =NULL;
 struct node * second =NULL;
 struct node * third =NULL;
 first =(struct node*) malloc (size of(struct node));
 second = (struct node*) malloc (size of (struct node));
 third =(struct node*) malloc ( size of (struct node));
 first →data=10;
 first →next =second;
 second→data =20;
 second → next = third;
  third → data =30;
  third → next =NULL;
 struct node* temp;
  temp = first ;
printf(" Elements of Linked List are \n");
while(temp != NULL)
    {
        printf("%d", temp->data);
        temp = temp->next;
    }
    printf(" No elements to be displayed in Linked List\n");
}
```

Doubly Linked List Programe:-

```
#include<stdio.h>
#include<stdlib.h>
struct node
{ int data;
  struct node * prev;
  struct node * next;
};
void main()
{
  struct node * first = NULL;
  struct node * second = NULL;
  struct node * third = NULL;
   first = (struct node*) malloc (sizeof(struct node));
   second = (struct node*) malloc (sizeof(struct node));
   third = (struct node*) malloc (sizeof (struct node));
   first -> prev = NULL;
   first -> data = 10;
   first -> next = second -> prev;
   second -> prev = first -> next;
    second -> data = 20;
    second -> next = third -> prev;
    third -> prev = second -> next;
     third -> data = 30;
     third -> next = NULL;
```

```
struct node *temp;
temp = first;   printf(" the elements of doubly lin
while (temp != NULL)
  { printf("%d", temp->data);
    temp = temp -> next;
  }
}
```