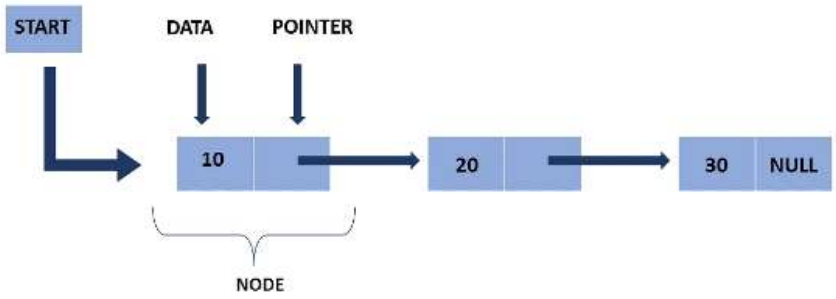
LINKED LIST

Linked list is a linear data structure in which elements are stored at contiguous memory location. Linked lists are mainly used when we need to deal with dynamic data elements. The linked list consists of data elements that are linked together. These data elements are called as nodes. Each node consist of a data and pointer field. The pointer field stores the address of the next node.

It is possible to grow or shrink the size of the linked list based on requirement. Unlike array linked list does not waste the memory space.

Linked list representation

Each node in linked list consists of a data and pointer field. The pointer field points to the next node in the list.



Here, the pointer field of the last node is NULL which indicated the end of the list.

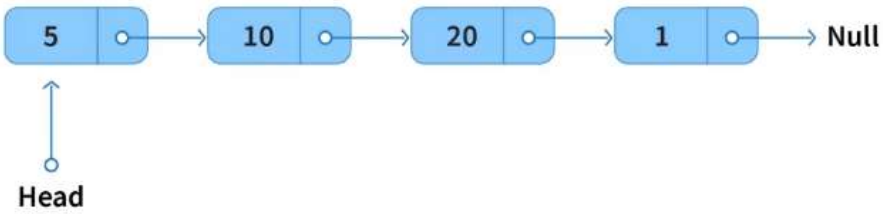
Types of linked list

Linked list can be of three types namely

- Singly linked list
- Doubly linked list
- Circular linked list

SINGLY LINKED LIST

The most common type of linked list is the singly linked list which is unidirectional, i.e. traversal can be done only in one direction.



The first node in the list is called as the *Header* node which points to the next node in the list. The pointer field of the last node is always *Null* indicating the end of the list.

The following operations can be performed in a singly linked list

- Insertion
- Deletion

- Display
- Search

Insertion

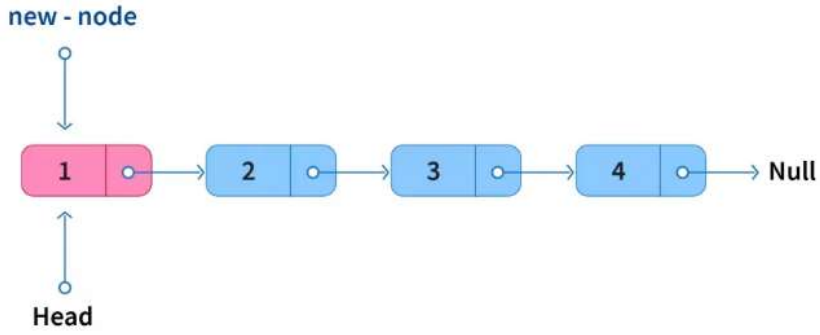
Insertion operation in a singly linked list can be done in three ways as follows

- Insert at the beginning
- Insert in the middle
- Insert at the end

1. *Insert at the beginning*

To insert at the beginning of the list, the below mentioned steps are to be followed

- Create a new node
- Make it as the header node by pointing to the first node in the list



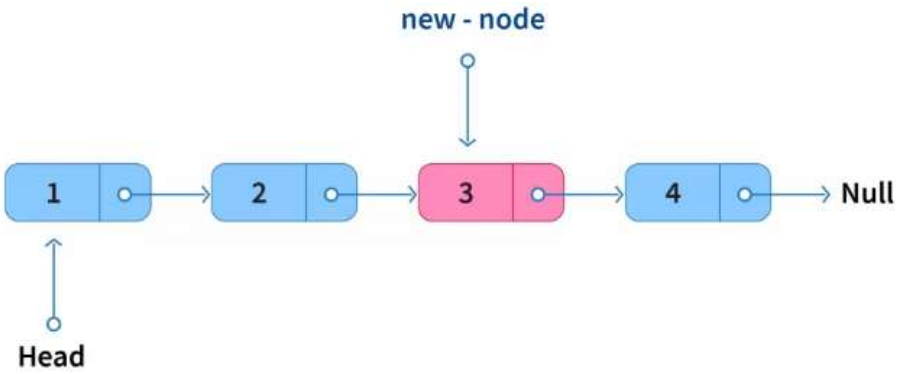
Here is the pseudo code

Newnode->data = 5;
Newnode->next = Head;
Head->next=Newnode;

2. Insert in the middle

Inserting a new node after some node in the list is done in the following ways

- Reach to the node after which you want to insert the new node
- Make the new node point to the next node of the current node that you want to insert after
- Now make the current node point to the new node



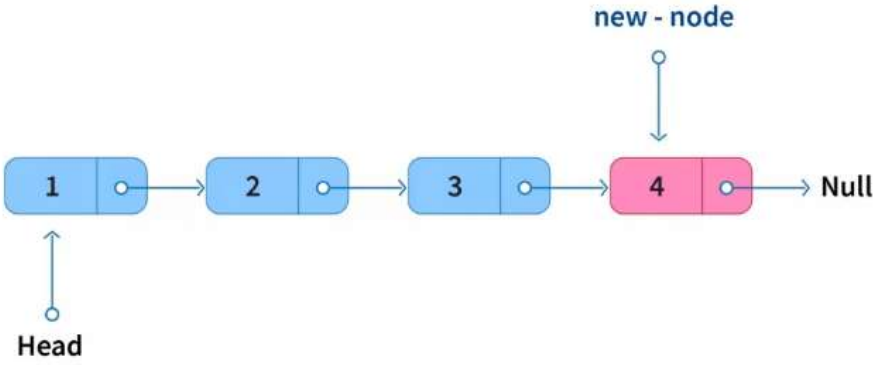
Here is the pseudo code

```
Newnode->data = 19;
Node temp = head;
While(temp->data!=value_mentioned)
{
temp=temp->next;
}
Newnode->next=temp->next;
Temp->next=Newnode;
```

3. Insert at the end

Inserting a node at the end can be done in following ways

- Create a new node
- Traverse to the last node
- Make the last node point to the new node
- Make the pointer field of new node as Null



Here is the pseudo code

```
Newnode->data=33;
Node temp = head
While(temp->next!=Null)
{
temp=temp->next;
}
temp->next=newnode;
newnode->next=Null;
}
```

Deletion

Like insertion deletion in a singly linked list can be done in the following ways

- Deletion at the beginning
- Deletion at the middle
- Deletion at the end

1. Deletion at the beginning

In a singly linked list, the first node can be deleted by making the Head point to the next node in the list.

The pseudo code to delete the first node is as follows

```
Head=Head->next;
```

2. Deletion at the middle

Deleting a node from the middle of the singly linked list can be done in the following way,

- Move to the node after which the required node is to be deleted.
- Make the current node point to the pointer field (next) of next node

The pseudo code to delete the node at the middle is as follows

```
Node temp = head;
While(temp->data!=value_mentioned)
{
Temp=Temp->next;
}
Temp->next=Temp->next->next;
```

3. Deleting at the end

The last node in a singly linked list can be deleted in the following way

- Move to the node before the last node
- Make the pointer field of this node null

The pseudo code to delete a node at the end of singly linked list is as follows

```
Node temp=Head;
While(temp->next->next!=Null)
{
temp=temp->next;
}
Temp->next=Null;
```

Search

It is essential to traverse through the singly linked list from the beginning of the list in order to search for an element. While searching, it the target value is found, the position of the node is returned.

The pseudo code to search a value in the singly linked list is as follows

```
Node temp=head;
While(temp!=Null && temp->data!=target_value)
{
temp=temp->next;
}
return temp;
```

Application of singly linked list

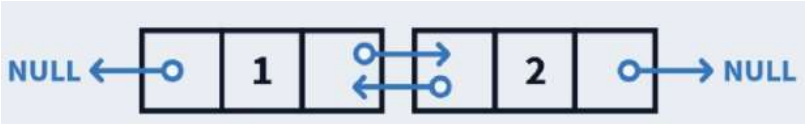
- Implementation of stack and queues
- Implementation of hash tables
- To perform Backspace operation
- To perform undo and redo operation
- Polynomial representation
- Dynamic memory representation

DOUBLY LINKED LIST

Doubly linked list can be considered as the enhancement of singly linked list where each node consists of a data field and two pointer fields. Unlike singly linked list, traversal is possible in both the directions.

Representation of doubly linked list

A node in a doubly linked list consists of three fields namely data, previous pointer and next pointer.



- The previous pointer of the first node will be null which marks the beginning of the list.
- The next pointer of the last node will be null which marks the end of the list.

The pseudo code to represent doubly linked list is as follows

```
Struct Node
{
Node previous; //pointer to the previous node
int data; //value stored in the particular node
Node next; //pointer to next node
}
```

Memory representation of doubly linked list

Linear array is used to represent doubly linked list in memory space where each memory address consists of three parts – data, memory of previous node and memory of next node.

Memory	Data	Previous pointer	Next pointer
1000	2	NULL	1001

1001	5	1000	1001
1002	9	1001	1003
1003	7	1002	NULL

The following are the operations that can be performed in a doubly linked list.

- Insertion
- Deletion
- Search
- Display

Insertion

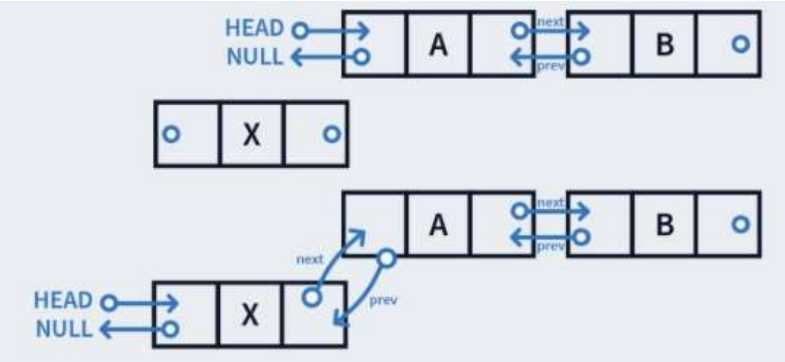
Compared to singly linked list, inserting in a doubly linked list is faster. Insertion in a doubly linked list can be done in a constant time.

1. Insert at the beginning

Consider a linked list $A \rightleftarrows B \rightleftarrows C$

Now, we want to insert the node X in the beginning of the list. After insertion at the beginning, the new list will be like $X \rightleftarrows A \rightleftarrows B \rightleftarrows C$

From the above it is seen that while inserting a new node the previous pointer of the first node in the list is to be changed.



The pseudo code to insert a node at the beginning of the list is as follows

```
Node newnode = new Node(data);
newnode->next = head;
newnode->prev = null;
if(head!=null)
{
head->prev = newnode;
head=newnode;
}
```

2. Insert at the middle

Doubly linked list before inserting a newnode X is $A \rightleftarrows B \rightleftarrows C$

Doubly linked list after inserting a newnode X is $A \rightleftarrows B \rightleftarrows X \rightleftarrows C$

From the above it is identified that the next pointer of B points to the new node and the previous pointer of C points to the new node. That is,

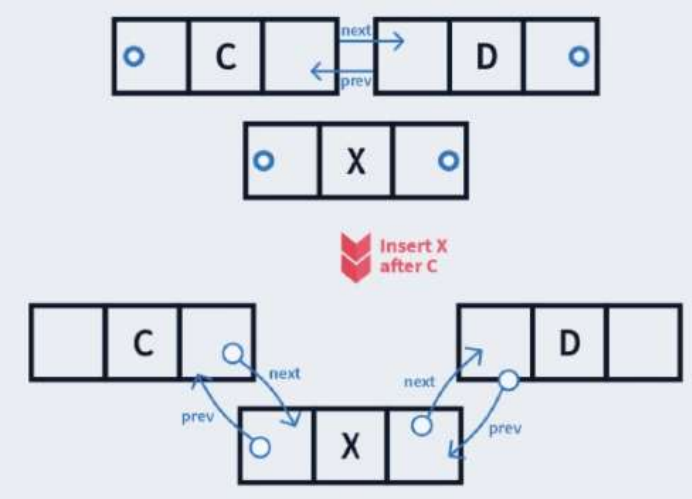
B->next = X

C->prev = X

Also,

X->prev = B

X ->next = C



3. Insert at the end

Doubly linked list before inserting a newnode X is $A \rightleftarrows B \rightleftarrows C$

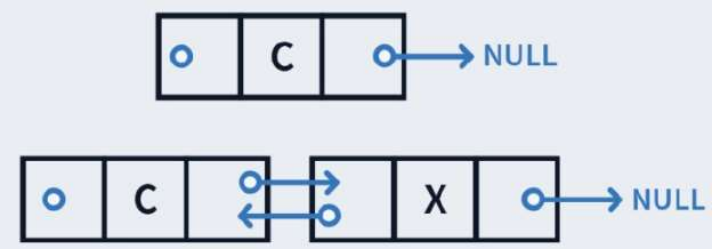
Doubly linked list after inserting a newnode X is $A \rightleftarrows B \rightleftarrows C \rightleftarrows X$

From the above it is identified that C->next points to the node X and the previous pointer of X points to C.

C->next = X

X->prev = C

X->next = Null



The pseudo code for doubly linked list is as follows

Node newnode = new Node(data);

```
newnode->next = Null;
//if the linked list is empty follow the below code
if(head==null)
{
newnode->prev = null;
head=newnode;
                    }
//if the list is not empty
Node temp = head;
    While(Temp->next!=null)
        Temp=temp->next
      temp->next=newnode;
      Newnode->prev=temp;
```

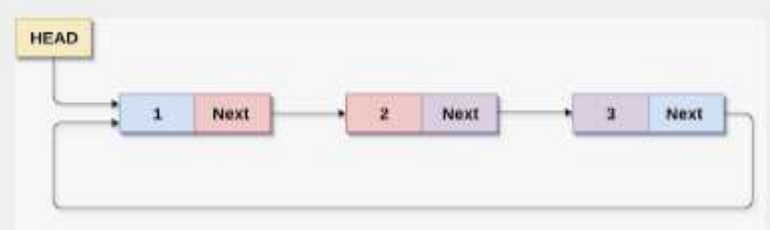
Applications of doubly linked list

1. Tabs in windows is a best example for doubly linked list with which you can switch between applications in both directions. In other words it is used where both forward and backward traversal is required.

2. It is used in performing undo and redo operation

3. Real – time example for doubly linked list is it is used in web page navigation.

CIRCULAR LINKED LIST



In circular linked list the nodes are connected together to form a cycle