# SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

**An Autonomous Institution**

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IoT Including CS & BCT**

COURSE NAME :23ITB201-DATA STRUCTURES & ALGORITHMS

II YEAR / III SEMESTER

Unit I- **LIST ADT**

Topic :Doubly Linked Lists

## Doubly linked list

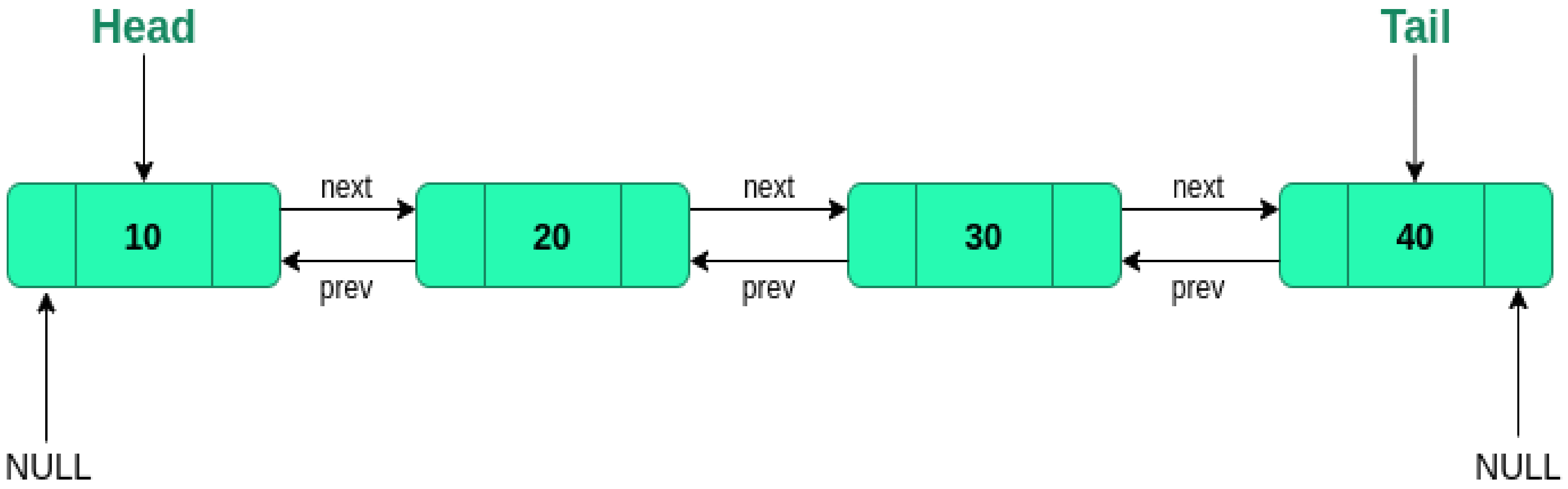
Doubly linked list is used where each node contains a link to the next as well as the previous node.

**Problem in SLL**

In a singly linked list, each node has only one link which points to the node next to it.

For this reason, we can traverse it from one side only and we cannot traverse back and forth.

# Doubly Linked List



## Create a Doubly Linked List

A doubly linked list is a type of linked list in which each node has two links.

The first link points to the previous node in the list while the second link points to the next node in the list.

In other words, navigation in a doubly linked list is possible from both sides, from the front as well as the back.

```
typedef struct node
{
    int data;
    struct node* next;
    struct node* prev;
} Node;
```

Here,

data: data field of the node.
next: pointer to the next node.
prev: pointer to the prev node.

## Basic Operations

The basic operations that are performed on the double linked list in this program are as follows:-

| Operation | Description | Time Complexity | Space Complexity |
|---|---|---|---|
| **Insert_at_head** | Adding an element to the beginning of the doubly linked list. | O(1) | O(1) |
| **Delete_at_head** | Deleting an element from the beginning of the doubly linked list. | O(1) | O(1) |
| **Insert_at_tail** | To add an element to the end of the doubly linked list. | O(1), if we have pointer to tail, otherwise O(N). | O(1) |

|  |  |  |  |
|---|---|---|---|
| **Delete_at_tail** | Delete an element from the end of the doubly linked list. | O(1), if we have pointer to tail, otherwise O(N). | O(1) |
| **Display forward** | To display the entire doubly linked list in a forward manner. | O(N) | O(1) |
| **Display backwards** | To display the entire doubly linked list in a backward manner. | O(N) | O(1) |

## Insertion:

Adding a new node at the beginning, end, or a specific position within the list.

```
void insert_at_head(int data)
{
    struct node* new_node = create_node(data);
    if (head == NULL) {
        head = new_node;
        tail = new_node;
    }
    else {
        new_node->next = head;
        head->prev = new_node;
        head = new_node;
    }
}
```

## Deletion:

Removing a node from the list, whether it be the head, tail, or a specific node.

```
void delete_at_head()
{
    if (head == NULL) {
        return;
    }
    struct node* temp = head;
    if (head == tail) {
        head = NULL;
        tail = NULL;
}
    else {
        head = head->next;
        head->prev = NULL;
    }
    free(temp);
}
```

## Traversal:

Visiting each node of the list in either forward or backward direction.

```
void display_forward()
{
    struct node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
```

## Advantages of Doubly Linked List

1. Efficient traversal in both forward and backward directions
2. Dynamic size adjustment
3. Easy to implement
4. Efficient memory utilization

## Disadvantages of Doubly Linked List

1. More memory usage
2. Slower access and search times
3. Complex implementation
4. Higher overhead for updates
5. Pointer management

## Real-Time Applications of Doubly Linked List

Doubly linked lists are used in web page navigation in both forward and backward directions.

It can be used in games like a deck of cards.

Any Query????

Thank you……