BML MUNJAL UNIVERSITY™

# Data Structures & Algorithms

## Linked List

## Variations on Linked Lists

# Drawback of singly linked list

**Limitation of Singly linked list**

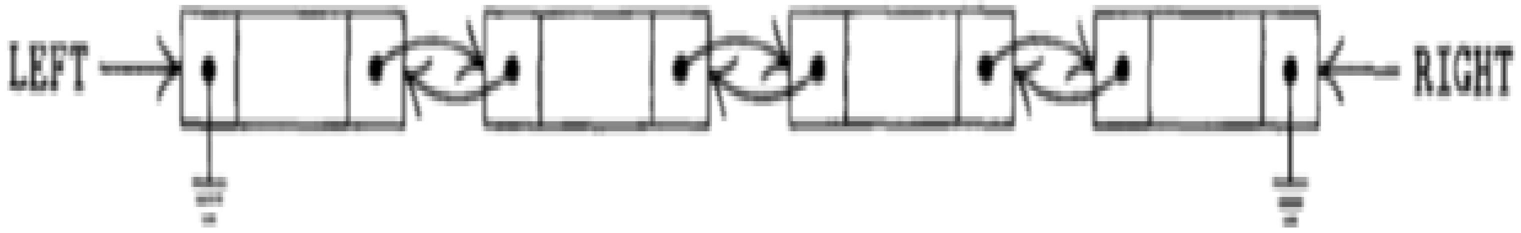
In a singly linked list, we could traverse only in one direction.

**How to overcome the limitation of singly linked list?**

Use doubly linked list.

Solves the problem of traversing backwards in a singly linked list.

The list can be traversed either forward or backward.



Each node contains a value, a link to its successor (if any), and a link to its predecessor (if any).

**Disadvantages of doubly linked list over singly linked list**

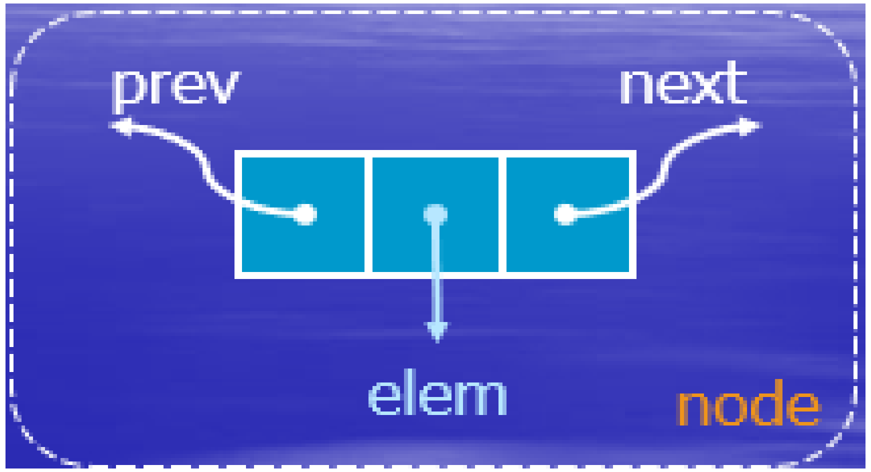
Doubly linked list occupy more space and often more operations are required for the similar tasks as compared to singly linked lists.

# Doubly-linked list

- Each node points to not only successor but the predecessor
- There are two NULL: at the first and last nodes in the list
- Advantage: given a node, it is easy to visit its predecessor. Convenient to traverse lists backwards
- Elements are still accessed sequentially.

A node in a doubly-linked list store two references:

- A **next** link; that points to the next node in the list, and
- A **prev** link; that points to the previous node in the list.



## Creating a Doubly-linked list

```
typedef struct node  {
  int data; // Data field
  struct node *next; // Address of next node
  struct node *prev; // Address of previous node
} DL_Node;

 DL_Node *left, *right;
 left=right=NULL;        // Initializing an empty doubly linked list
```

## Doubly-linked list

Write algorithms for all the operations described for singly linked list, using doubly linked list.

A doubly linked list has O(1) insertion and deletion at both ends.

# Circular linked list

Instead of 2 pointers, left and right, singly linked circular list use only one pointer.

In the circular list, the pointer of the last node points not NULL but points to the first node.

There is no NULL link in the list.

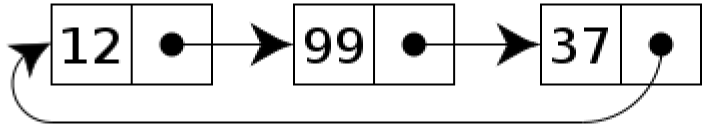
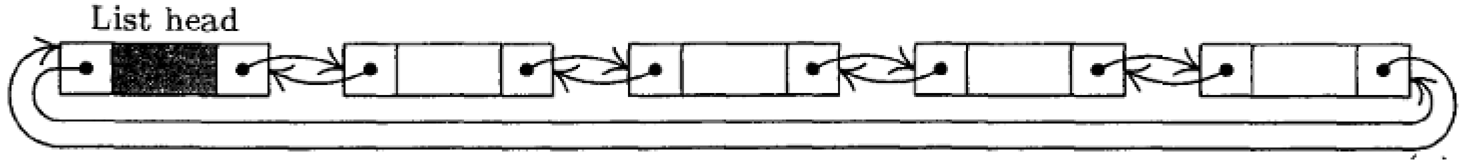


Fig: Singly circular linked list

Write an algorithm to traverse (or display) the given circular linked list.

# Circular-doubly linked list

Doubly Circular linked list has both the properties of doubly linked list and circular linked list.



Two consecutive elements are linked by previous and next pointer.

The last node points to first node by next pointer and also the previous pointer of the first node points to the last node.

In a circular linked list, the last pointer points to the first node, whereas in the case of a circular doubly linked list, the first node also points to the last node of the list.

Node traversal from any direction is possible and also jumping from left to right or from right to left is only one operation: left pointer previous is right and also right pointer next is left.