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

COURSE NAME : 19ITT201- DATA STRUCTURES

II YEAR / III SEMESTER

Unit 1- LINEAR STRUCTURES AND TREES

Topic 5 : Doubly Linked List Based Implementation

Problem

- All operations require an extra pointer previous to be maintained
- It uses extra memory when compared to the array and singly linked list.
- Since elements in memory are stored randomly, therefore the elements are accessed sequentially no direct access is allowed.

Doubly Linked List Based Implementation

➢Define Doubly Linked List

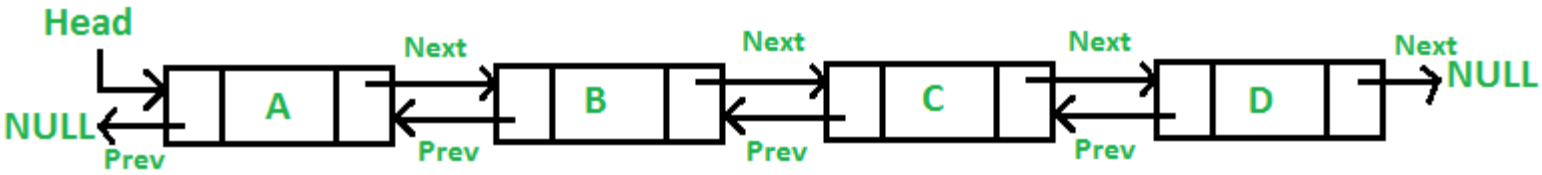
•Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly linked list.

•**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.

•**Prev** − Each link of a linked list contains a link to the previous link called Prev.

•**LinkedList** − A Linked List contains the connection link to the first link called First and to the last link called Last.



Doubly Linked List Based Implementation **–Cont..**

Basic Operations

Following are the basic operations supported by a list.

Insertion – Adds an element at the beginning of the list.

Deletion – Deletes an element at the beginning of the list.

Insert Last – Adds an element at the end of the list.

Delete Last – Deletes an element from the end of the list.

Insert After – Adds an element after an item of the list.

Delete – Deletes an element from the list using the key.

Display forward – Displays the complete list in a forward manner.

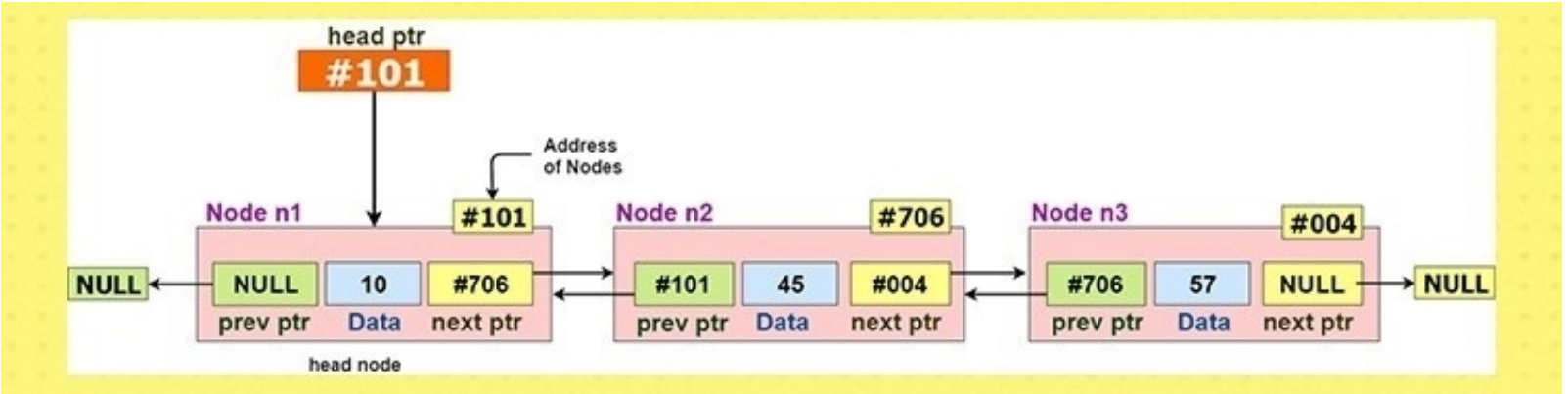
Display backward – Displays the complete list in a backward manner.

Doubly Linked List Based Implementation–**Cont..**

1. Creation of list

A linked list node

```
* Node of a doubly linked list */
struct Node {
    int data;
    struct Node* next; // Pointer to next node in DLL
    struct Node* prev; // Pointer to previous node in DLL
};
```



Doubly Linked List Based Implementation

In this post, methods to insert a new node in linked list are discussed.

A node can be added

Insertion

A node can be added in four ways

1) At the front of the DLL

2) After a given node.

3) At the end of the DLL

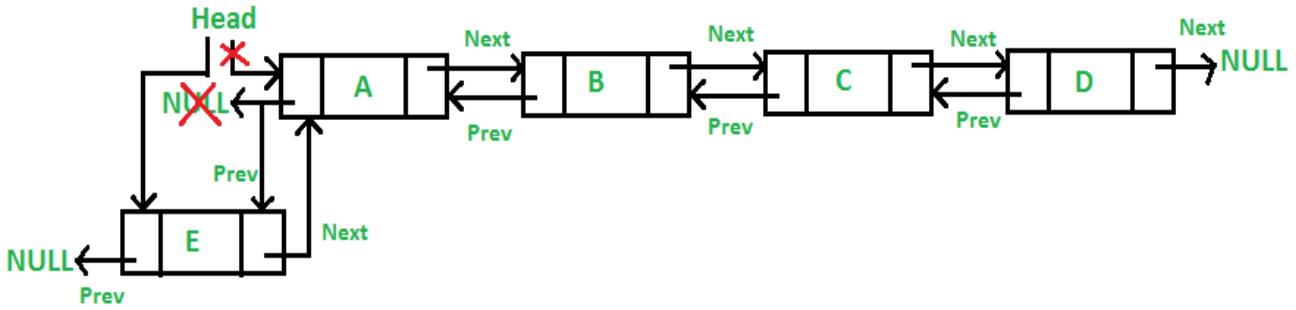
Doubly Linked List Based Implementation

1) **Add a node at the front.**

   For example if the given Linked List is 10152025 and we add an item 5 at the front, then the Linked List becomes 510152025.

2) Let us call the function that adds at the front of the list is push().

3) The push() must receive a pointer to the head pointer, because push must change the head pointer to point to the new node .



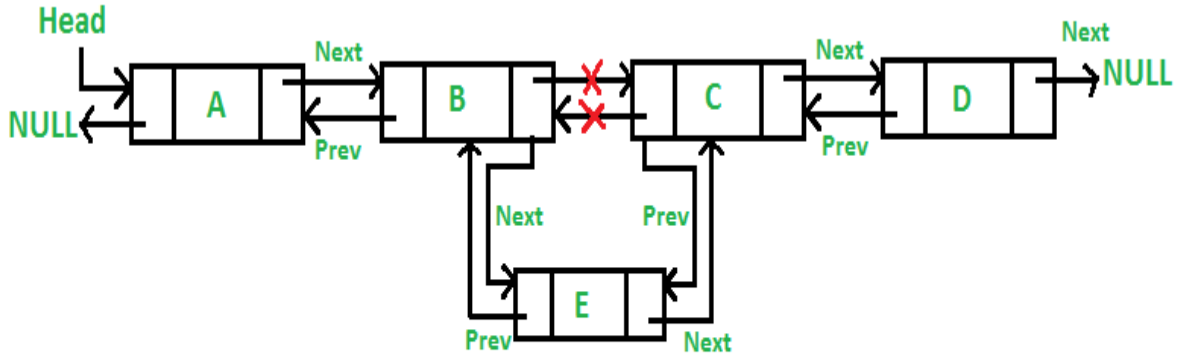
Doubly Linked List Based Implementation –Cont..

```
void push(struct Node** head_ref, int new_data)
{
   /* 1. allocate node */
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

   /* 2. put in the data  */
   new_node->data = new_data;

   /* 3. Make next of new node as head and previous as NULL */
   new_node->next = (*head_ref);
   new_node->prev = NULL;

   /* 4. change prev of head node to new node */
   if ((*head_ref) != NULL)
      (*head_ref)->prev = new_node;

   /* 5. move the head to point to the new node */
   (*head_ref) = new_node;
}
```

Doubly Linked List Based Implementation –Cont..

2) Add a node after a given node.

We are given pointer to a node as prev_node, and the new node is inserted after the given node.



Doubly Linked List Based Implementation –Cont..

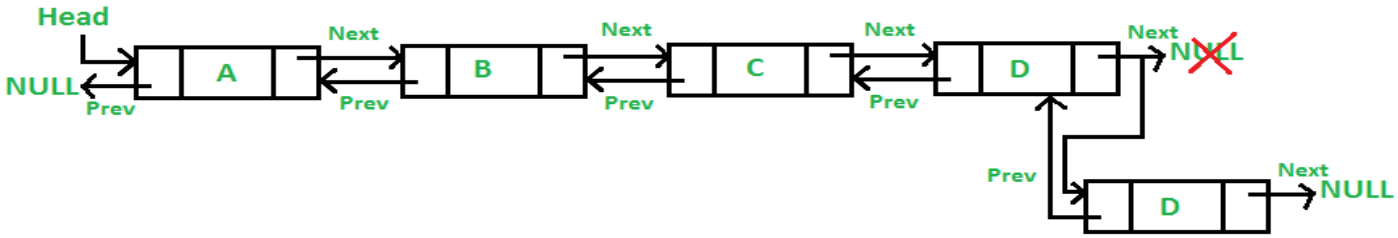
```
void insertAfter(struct Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL) {
       printf("the given previous node cannot be NULL");
       return;
    }
     /* 2. allocate new node */
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
     /* 3. put in the data  */
    new_node->data = new_data;
     /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;
     /* 5. Make the next of prev_node as new_node */
    prev_node->next = new_node;
/* 6. Make prev_node as previous of new_node */
    new_node->prev = prev_node;

    /* 7. Change previous of new_node's next node */
    if (new_node->next != NULL)
       new_node->next->prev = new_node;
}
```

Doubly Linked List Based Implementation –Cont..

- **3) Add a node at the end.**

  The new node is always added after the last node of the given Linked List. For example if the given DLL is 5101520 and we add an item 25 at the end, then the DLL becomes 510152025.

  Since a Linked List is typically represented by the head of it, we have to traverse the list till end and then change the next of last node to new node.



Doubly Linked List Based Implementation –Cont..

```
void append(struct Node** head_ref, int new_data)
{
   /* 1. allocate node */
   struct Node* new_node = (struct
Node*)malloc(sizeof(struct Node));

   struct Node* last = *head_ref; /* used in step 5*/

   /* 2. put in the data  */
   new_node->data = new_data;

   /* 3. This new node is going to be the last node, so
        make next of it as NULL*/
   new_node->next = NULL;

   /* 4. If the Linked List is empty, then make the new
        node as head */
   if (*head_ref == NULL) {
      new_node->prev = NULL;
      *head_ref = new_node;
      return;
   }

   /* 5. Else traverse till the last node */
   while (last->next != NULL)
      last = last->next;

   /* 6. Change the next of last node */
   last->next = new_node;

   /* 7. Make last node as previous of new
node */
   new_node->prev = last;

   return;
}
```

Activity

Advantages

1. A DLL can be traversed in both forward and backward direction.
2. The delete operation in DLL is more efficient if pointer to the node to be deleted is given.
3. We can quickly insert a new node before a given node.

   In singly linked list, to delete a node, 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 previous pointer.
4. Other data structures like stacks, Hash Tables, Binary trees can also be constructed or programmed using a doubly-linked list.

Disadvantages

1. Every node of DLL Require extra space for an previous pointer. It is possible to implement DLL with single pointer though .
2. All operations require an extra pointer previous to be maintained. For example, in insertion, we need to modify previous pointers together with next pointers. For example in following functions for insertions at different positions, we need 1 or 2 extra steps to set previous pointer.

Assessment 1

1. List out the advantages of doubly linked list based implementation

a)_____

b)_____

c)_____

d)_____

2. Identify the disadvantages of doubly linked list based implementation

a)_____

b)_____

c)_____

d)_____

REFERENCES

1. M. A. Weiss, “Data Structures and Algorithm Analysis in C”, Pearson Education, 8th Edition, 2007. [Unit I, II, III, IV,V]

2. A. V. Aho, J. E. Hopcroft and J. D. Ullman, “Data Structures and Algorithms”, Pearson Education, 2nd Edition, 2007 [Unit IV].

3. A.M.Tenenbaum, Y. Langsam and M. J. Augenstein, “Data Structures using C”,PearsonEducation, 1st Edition, 2003.(UNIT I,II,V)

4. https://www.youtube.com/watch?v=jkbS-bKXTEE

THANK YOU