Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE

INSTITUTE OF TECHNOLOGY

West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8391

DATA STRUCTURES
(Common to CSE & IT)

UNIT NO 1

LINEAR DATA STRUCTURES - LISTS

1.4 CIRCULARLY LINKED LIST

1.4.2 DOUBLE CIRCULAR LINKED LIST

Version: 1.XX

4 QUALITY EDUCATION

NBA NATIONAL BOARD of ACCREDITATION

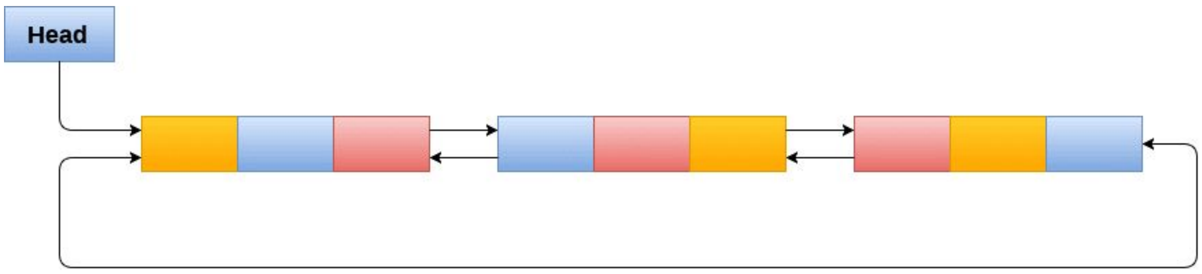
NAAC (A+)

nirf

Circular Doubly Linked List

Circular doubly linked list is a more complexed type of data structure in which a node contain pointers to its previous node as well as the next node. Circular doubly linked list doesn't contain NULL in any of the node. The last node of the list contains the address of the first node of the list. The first node of the list also contain address of the last node in its previous pointer.

A circular doubly linked list is shown in the following figure.

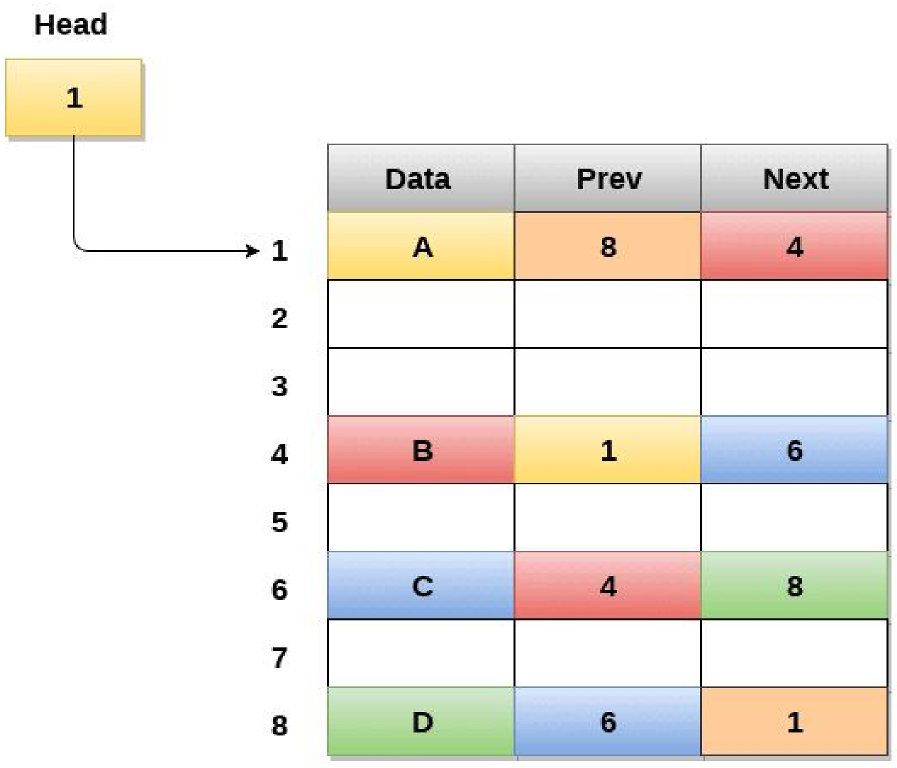


Circular Doubly Linked List

Due to the fact that a circular doubly linked list contains three parts in its structure therefore, it demands more space per node and more expensive basic operations. However, a circular doubly linked list provides easy manipulation of the pointers and the searching becomes twice as efficient.

Memory Management of Circular Doubly linked list

The following figure shows the way in which the memory is allocated for a circular doubly linked list. The variable head contains the address of the first element of the list i.e. 1 hence the starting node of the list contains data A is stored at address 1. Since, each node of the list is supposed to have three parts therefore, the starting node of the list contains address of the last node i.e. 8 and the next node i.e. 4. The last node of the list that is stored at address 8 and containing data as 6, contains address of the first node of the list as shown in the image i.e. 1. In circular doubly linked list, the last node is identified by the address of the first node which is stored in the next part of the last node therefore the node which contains the address of the first node, is actually the last node of the list.



	Data	Prev	Next
1	A	8	4
2			
3			
4	B	1	6
5			
6	C	4	8
7			
8	D	6	1

Memory Representation of a Circular Doubly linked list

Operations on circular doubly linked list :

There are various operations which can be performed on circular doubly linked list. The node structure of a circular doubly linked list is similar to doubly linked list. However, the operations on circular doubly linked list is described in the following table.

SN	Operation	Description
1	Insertion at beginning	Adding a node in circular doubly linked list at the beginning.
2	Insertion at end	Adding a node in circular doubly linked list at the end.
3	Deletion at beginning	Removing a node in circular doubly linked list from beginning.
4	Deletion at end	Removing a node in circular doubly linked list at the end.

Traversing and searching in circular doubly linked list is similar to that in the circular singly 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 and the last node points to first node by next pointer and also the previous pointer of the head node points to the tail node. This list has eliminated all the short comings of all previous lists discussed in the previous sections. Node traversal from any direction is possible and also jumping from head to tail or from tail to head is only one operation

Insertion in circular doubly linked list at beginning

There are two scenario of inserting a node in circular doubly linked list at beginning. Either the list is empty or it contains more than one element in the list.

Allocate the memory space for the new node **ptr** by using the following statement.

```
ptr = (struct node *)malloc(sizeof(struct node));
```

In the first case, the condition **head == NULL** becomes true therefore, the node will be added as the first node in the list. The next and the previous pointer of this newly added node will point to itself only. This can be done by using the following statement.

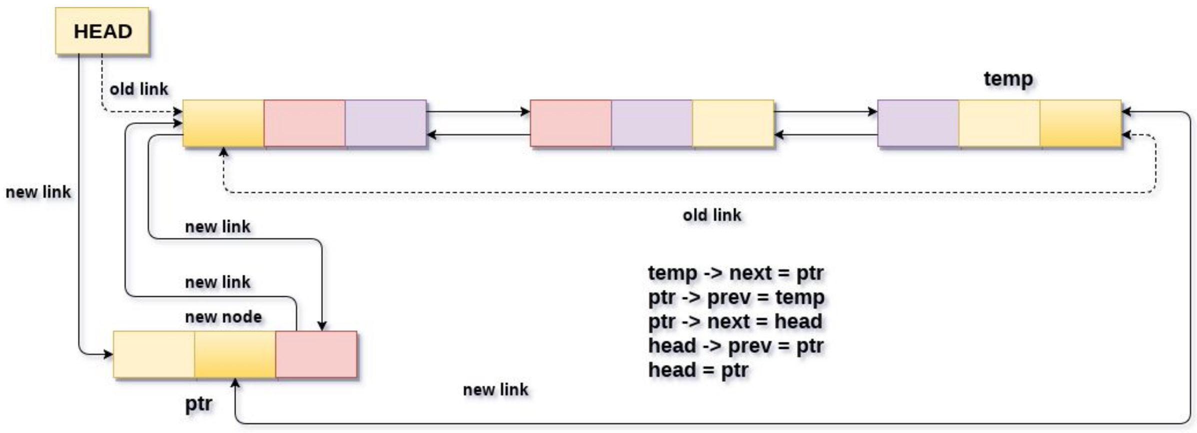
```
head = ptr;
     ptr -> next = head;
     ptr -> prev = head;
```

In the second scenario, the condition **head == NULL** becomes false. In this case, we need to make a few pointer adjustments at the end of the list. For this purpose, we need to reach the last node of the list through traversing the list. Traversing the list can be done by using the following statements.

```
temp = head;
while(temp -> next != head)
{
    temp = temp -> next;
}
```

At the end of loop, the pointer temp would point to the last node of the list. Since the node which is to be inserted will be the first node of the list therefore, temp must contain the address of the new node ptr into its next part. All the pointer adjustments can be done by using the following statements.

```
temp -> next = ptr;
    ptr -> prev = temp;
    head -> prev = ptr;
    ptr -> next = head;
    head = ptr;
```



Insertion into circular doubly linked list at beginning

Deletion in Circular doubly linked list at beginning

There can be two scenario of deleting the first node in a circular doubly linked list.

The node which is to be deleted can be the only node present in the linked list. In this case, the condition head → next == head will become true, therefore the list needs to be completely deleted.

It can be simply done by assigning head pointer of the list to null and free the head pointer.

```
1.  head = NULL;
2.      free(head);
```

in the second scenario, the list contains more than one element in the list, therefore the condition head → next == head will become false. Now, reach the last node of the list and make a few pointer adjustments there. Run a while loop for this purpose

```
1.  temp = head;
2.      while(temp -> next != head)
3.      {
4.          temp = temp -> next;
5.      }
```

Now, temp will point to the last node of the list. The first node of the list i.e. pointed by head pointer, will need to be deleted. Therefore the last node must contain the address of the node that is pointed by the next pointer of the existing head node. Use the following statement for this purpose.

1. temp -> next = head -> next;

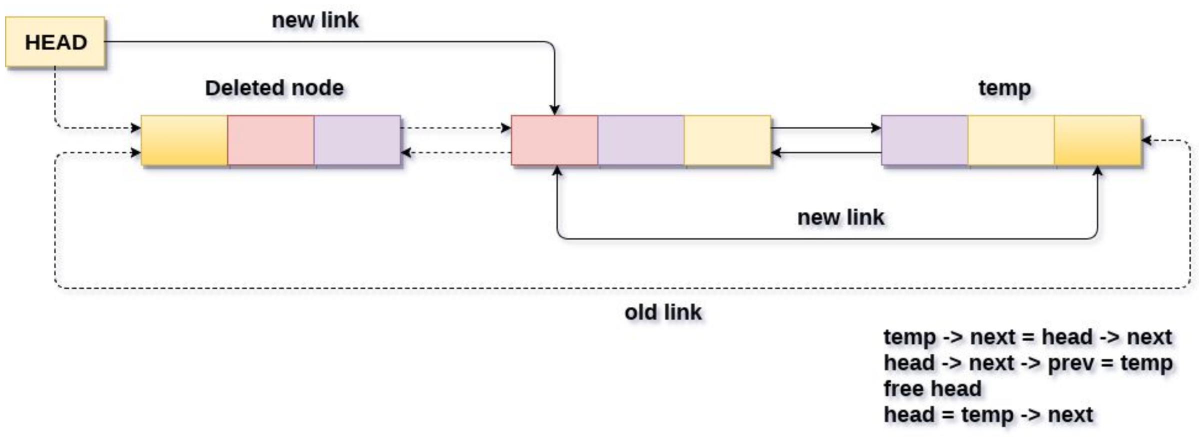
The new head node i.e. next of existing head node must also point to the last node of the list through its previous pointer. Use the following statement for this purpose.

1. head -> next -> prev = temp;

Now, free the head pointer and the make its next pointer, the new head node of the list.

1. free(head);
2. head = temp -> next;

in this way, a node is deleted at the beginning from a circular doubly linked list.



Deletion in circular doubly linked list at beginning