Data Structures and itsApplications

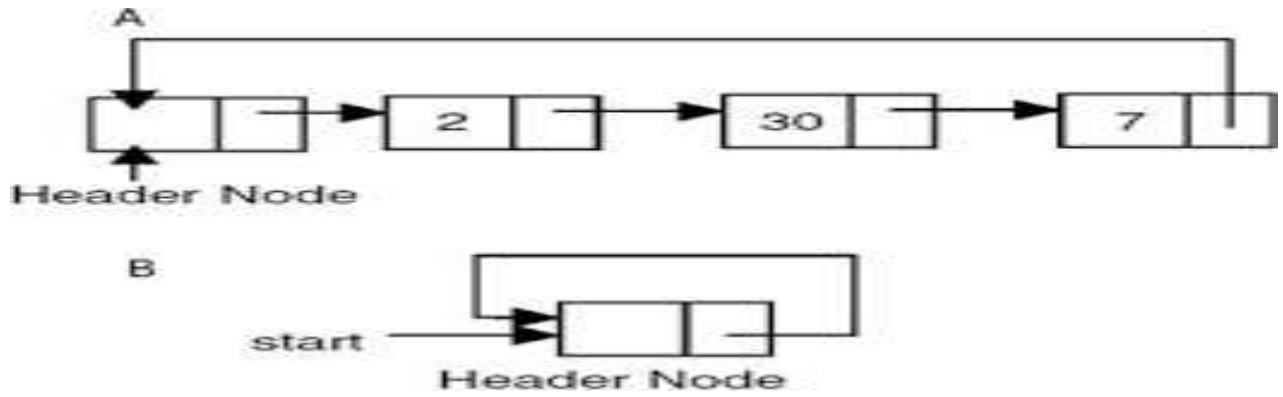
Dinesh Singh
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Circular Linked Lists

Dinesh Singh
Department of Computer Science & Engineering

Circular Linked Lists



- The first node in the list is the header node.
- The address part of the last node points to the header node
- Circular list does not have a natural first or the last node
- An External pointer points to the header node and the one following this node is the first node.

Operations on Circular Linked Lists

Implementation of some operations on Circular linked Lists with header node

- Insert at the head of a list
- Insert at the end of the List
- Delete a Node given its value

Note: head is a pointer to the header node and the following node is the first node

Creating Header node

```
struct node *create_head()
 {
   struct node *temp;
   temp=(struct node*)malloc(sizeof(struct node));
   temp->data=0; // keeps the count of nodes in the list
   temp->next=temp;
   return temp;
  }
```

Operations on Circular Linked Lists

Algorithm to insert a node at the head of the list
insert_head(p,x)
//p pointer to header node, x element to be inserted
//x gets inserted after the header node

allocate memory for the node
initialise the node
//insert the new node after the header node
Copy the value of the next part of the header node into the next part of the new node
Copy the address of the new node into next part of the header node

Operations on Circular Linked Lists with header node


```
void insert_head(struct node *p,int x)
//p points to the header node, x element to be inserted
 {
  struct node *temp;
  //create node and initialise
   temp=(struct node*)malloc(sizeof(struct node));
   temp->data=x;
// next part of new node points to the node after the header node
   temp->next=p->next;
   p->next=temp; //next part of header node points to the new node
   p->data++;
  }
```

Operations on Circular Linked Lists with header node

Algorithm to insert a node at the end of the list
insert_tail(p,x)
//p pointer to header node, x element to be inserted
allocate memory for the node
initialise the node

move to the last node
//insert the new node after the last node
Copy the address of the header node into next of new node
Copy the address of the new node into the next of last node

Operations on Circular Linked Lists with header node

Algorithm to insert a node at the end of the list

```
void insert_tail(struct node *p,int x)
 {
  struct node *temp,*q;
   temp=(struct node*)malloc(sizeof(struct node));
   temp->data=x;

   q=p->next; // go the first node

   while(q->next!=p) // move to the last node
    q=q->next;

 temp->next=p;// copy address of header node into next of new node
 q->next=temp; // copy the address of new node into next of the last node
 p->data++; // increment the count of nodes in the list
 }
```

Operations on Circular Linked Lists with header node

Algorithm to delete a node given its value
delete_node(p,x)
//p pointer to header node, x element to be deleted

move forward until the node to be deleted is found
or header node is reaches
If(node found)
  delete the node by adjusting the pointers
else
 node not found // if header node is reached

Operations on Circular Linked Lists with header node

```
void delete_node(struct node *p, int x)
 {
  //p points to the header node, x is element to be inserted
   struct node *prev,*q;

   q=p->next; // go to the first node
   prev=p;
  //move forward until the data is found or header node is reached
   while((q!=p)&&(q->data!=x))
   {
    prev=q; // keep track of the previous node
    q=q->next;
   }
```

Operations on Circular Linked Lists with header node


```
if(q==p) // header node reached
    printf("Node not found..\n");
  else
  {
   prev->next=q->next; //delete the node
   free(q);
   p->data--; // decrement the count of nodes in the list
  }
}
```

THANK YOU

Dinesh Singh
Department of Computer Science & Engineering

dineshs@pes.edu

+91 8088654402