Data Structures and its Applications
UE24CS252A

Dinesh Singh
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Dequeue - Implementation

Dinesh Singh
Department of Computer Science & Engineering

Dequeue(Double ended Queue) - definition

PES UNIVERSITY

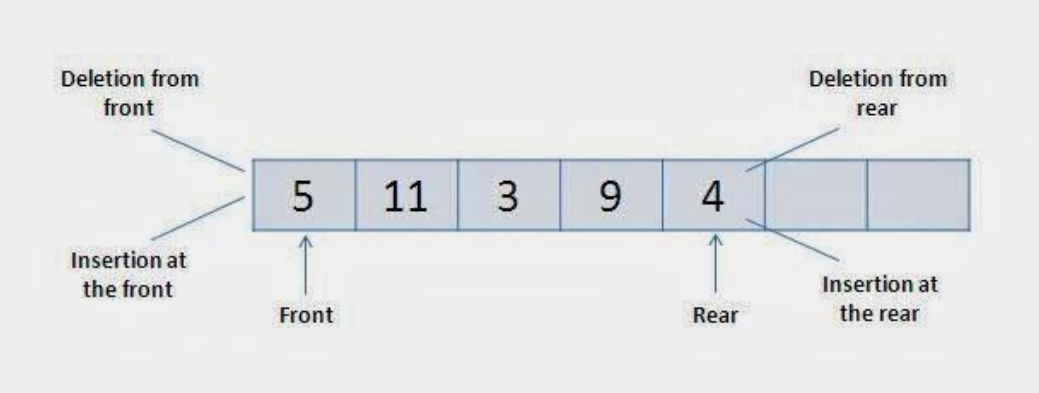
Double ended queue is a queue that allows insertion and deletion at both ends.

Data Structures and its Applications

Dequeue(Double ended Queue) - definition

The following four basic operations are performed on dequeue:

- *insertFront()*: Adds an item at the front of Deque.
- *insertRear()*: Adds an item at the rear of Deque.
- *deleteFront()*: Deletes an item from front of Deque.
- *deleteRear()*: Deletes an item from rear of Deque.



Data Structures and its Applications

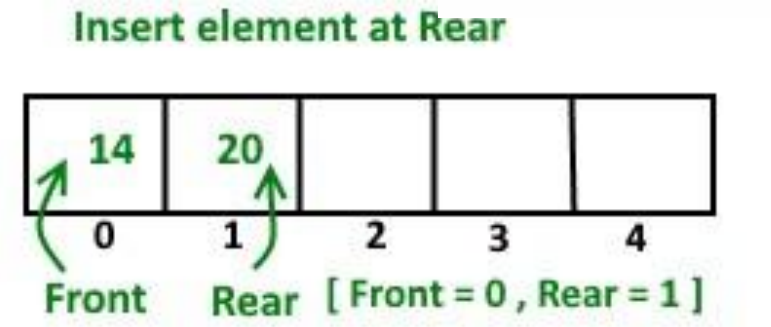
Dequeue(Double ended Queue) - Array Implementation

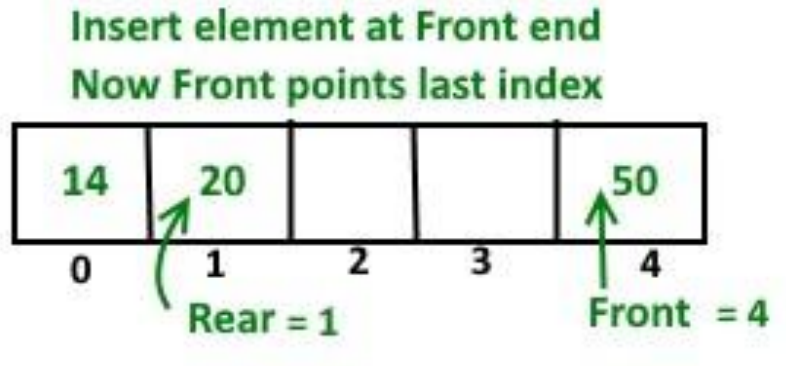
Insert Elements at Rear end :

Check whether the queue is full If rear = size-1
initialise rear to 0.
else
increment rear by 1
insert element at location rear

Insert element front end

Check if the queue is full If Front =0
move front to last location (size -1)
else
decrement front by 1 insert at location front





Data Structures and its Applications

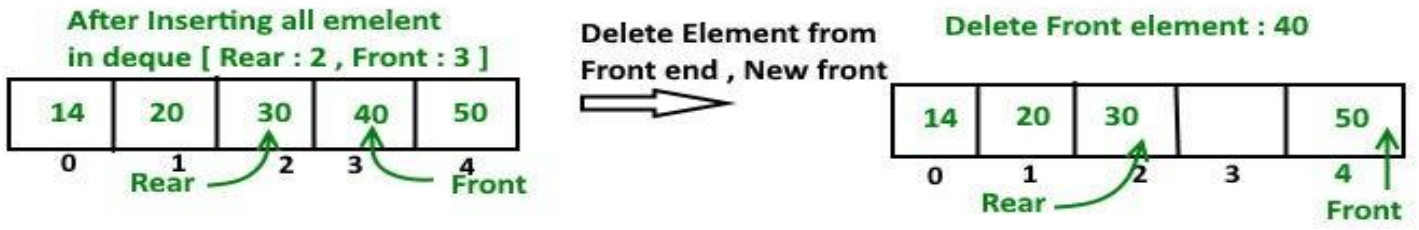
Dequeue(Double ended Queue) - Array Implementation

Delete element at Rear end

check if the queue is empty
delete the element pointed by rear
If dequeue has one element
  front=-1 rear=-1;
If rear is at first index make
rear = size-1 else
  decrease rear by 1

Delete element at front end

check if the queue is empty
delete the element pointed by front
 If dequeue has one element
  front=-1 rear=-1;
If front is at last index make front = 0
else
  increase front by 1



Dequeue(Double ended Queue) - Doubly Linked list Implementation

Structure of Dequeue

```
struct dequeue
{
  struct node * front; struct node * rear;
};
struct node
{
   int data;
   struct node * prev, *next;
};
struct   dequeue dq;

dq.front=dq.rear = NULL
```

Dequeue(Double ended Queue)- Doubly Linked list Implementation


```
//insert in front of the queue
void qinsert_head(int x,struct dequeue *dq)
  {
    struct node *temp;

    temp=(struct node*)malloc(sizeof(struct node));
    temp->data=x;
    temp->prev=temp->next=NULL;

    if(dq->front==NULL) // first element
    dq->front=dq->rear=temp;
     else
      {
        temp->next=dq->front;  // insert in front
        dq->front->prev=temp; temp->prev=NULL;
        dq->front=temp;
      }
  }
```

Dequeue(Double ended Queue) - Doubly Linked list Implementation


```
//insert at the rear of the queue
void qinsert_tail(int x,struct dequeue* dq)
 {
  struct node *temp;

   temp=(struct node*)malloc(sizeof(struct node)); temp->data=x;
   temp->prev=temp->next=NULL;

   if(dq->front==NULL)
     dq->front=dq->rear=temp; else
     {
      dq->rear->next=temp; temp->prev=dq->rear; dq->rear=temp;
     }
 }
```

Data Structures and its Applications

Dequeue(Double ended Queue) - Doubly Linked list Implementation


```
//delete at the front of the queue
int qdelete_head(struct dequeue* dq)
 {
   struct node *q;
   int x;
   if(dq->front==NULL) return -1;

  q=dq->front;
   x=q->data;
  if(dq->front==dq->rear)//only one node
  dq->front=dq->rear=NULL;
  else
   {
     dq->front=dq->front->next;
     dq->front->prev=NULL;
   }
   free(q); return x;
  }
```

Dequeue(Double ended Queue)- Doubly Linked list Implementation


```
//delete at the rear of the queue
int  qdelete_tail(struct dequeue* dq)
 {
  struct node *q;
   int x;
   if(dq->front==NULL) return -1;
   q=dq->rear; x=q->data;
  if(dq->front==dq->rear)//only one node
    dq->front=dq->rear=NULL; else
   {
     dq->rear=dq->rear->prev;
     dq->rear->next=NULL;
    }
   free(q); return x;
 }
```

Multiple Choice Questions (MCQ's)

Question 1: What is the defining characteristic of a Dequeue (Double Ended Queue)?

a) It only allows insertions at the front.

b) It only allows deletions from the rear.

c) It allows insertion and deletion at both ends.

d) It is a type of priority queue.

Multiple Choice Questions (MCQ's)

Question 1: What is the defining characteristic of a Dequeue (Double Ended Queue)?

a) It only allows insertions at the front.

b) It only allows deletions from the rear.

c) It allows insertion and deletion at both ends.

d) It is a type of priority queue.

Multiple Choice Questions (MCQ's)

Question 2: Which of the following is NOT one of the four basic operations performed on a Dequeue as listed in the presentation?

a) insertFront()

b) deleteMiddle()

c) insertRear()

d) deleteFront()

Multiple Choice Questions (MCQ's)

Question 2: Which of the following is NOT one of the four basic operations performed on a Dequeue as listed in the presentation?

a) insertFront()

b) deleteMiddle()

c) insertRear()

d) deleteFront()

Question 3: When inserting an element at the rear end of an array-implemented Dequeue, if the rear is currently at size-1, what is the next step according to the presentation?

a) The queue is declared full, and insertion fails.

b) rear is initialized to 0.

c) front is incremented by 1.

d) The element is inserted directly at size-1.

Question 3: When inserting an element at the rear end of an array-implemented Dequeue, if the rear is currently at size-1, what is the next step according to the presentation?

a) The queue is declared full, and insertion fails.

b) rear is initialized to 0.

c) front is incremented by 1.

d) The element is inserted directly at size-1.

Question 4: In a doubly linked list implementation of a Dequeue, what pointers does each struct node contain in addition to its data?

a) Only next pointer.

b) Only prev pointer.

c) Both prev and next pointers.

d) A single pointer to the dequeue structure.

Question 4: In a doubly linked list implementation of a Dequeue, what pointers does each struct node contain in addition to its data?

a) Only next pointer.

b) Only prev pointer.

c) Both prev and next pointers.

d) A single pointer to the dequeue structure.

Question 5: When deleting an element from the front of a doubly linked list implemented Dequeue, if there was only one node in the queue, what happens to dq->front and dq->rear?

a) dq->front becomes NULL and dq->rear points to the new head.

b) dq->front and dq->rear both become NULL.

c) Only dq->front becomes NULL.

d) Both remain unchanged.

Multiple Choice Questions (MCQ's)

Question 5: When deleting an element from the front of a doubly linked list implemented Dequeue, if there was only one node in the queue, what happens to dq->front and dq->rear?

a) dq->front becomes NULL and dq->rear points to the new head.

b) dq->front and dq->rear both become NULL.

c) Only dq->front becomes NULL.

d) Both remain unchanged.

THANK YOU

Dinesh Singh

Department of Computer Science & Engineering

dineshs@pes.edu

+91 8088654402