



Data Structures and its Applications

UE24CS252A

Dinesh Singh

Department of Computer Science &
Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Basic Structure of a Queue , Implementation using an Array

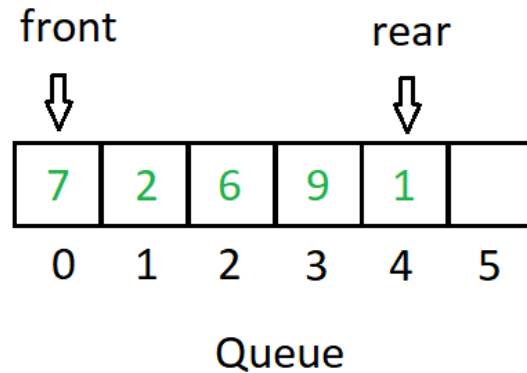
Dinesh Singh

Department of Computer Science & Engineering

Data Structures and its Applications

Queue Data Structure - definition

- A Queue is an ordered collection of items from which items may be deleted at one end (called the front of the queue)and into which items may be inserted at the other end (called the rear of the queue).



- Different Types of Queues
 - Simple Queue
 - Circular Queue
 - Priority Queue
 - Dequeue
- Implementation
 - Sequential Representation (Arrays)
 - Linked Representation (Linked Lists)

- Three primitive operations can be applied to the queue
- $\text{Insert}(q, x)$: inserts x at the rear of the queue q
- $x = \text{remove}(q)$: deletes front element from the queue and set x to its contents
- $\text{empty}(q)$: returns true or false depending on whether the queue contains any elements.
- Insert operation cannot be performed if the queue has reached the maximum size.
- The result of an illegal attempt to insert an element into a queue which has reached its maximum size is called overflow.
- The remove operation can be applied only if the queue is non empty.
- The result of an illegal attempt to remove an element from an empty queue is called underflow.
- The empty operation is always applicable.

Data Structures and its Applications

Structure of a Simple Queue – Sequential Representation



```
#define MAXQUEUE 100 struct queue
{
    int items [MAXQUEUE]; int front, rear;
};
```

```
struct queue q; q.rear = q.front = -1;
```

Functions to implement the operations

- insert (x, &q)
- remove (&q)
- empty(&q)

Inserting into a queue

1. Check the queue for overflow condition
2. If the queue is not in overflow condition, increment the rear index and insert the element at a location indicated by the rear index.
3. If this is the first element in the queue, initialise front to 0.
4. Return 1

Data Structures and its Applications

Simple Queue – Implementation of Insert



```
int insert(int x, struct queue *q)
{
    //check queue overflow if(q->rear==MAXQUEUE - 1)
    {
        printf("Queue overflow..\n"); return -1;
    }
    (q->rear)++;
    q->items[q->rear]=x;
    if(q->front==-1)//if first element
        q->front=0;
    return 1;
}
```


Removing element from the queue

1. Check the queue for underflow condition
2. If the queue is not in underflow condition, remove the element pointed by the front index into x.
3. If this was the only element in the queue, initialise front and rear to -1.
4. Return x

Data Structures and its Applications

Simple Queue – Implementation of remove



```
int remove(struct queue *q)
{
    int x;

    if(q->front==-1)
    {
        printf("Queue empty..\n");//underflow
        return -1;
    }
    x=q->items[q->front];
    if(q->front==q->rear)//only one element in queue
        q->front=q->rear=-1;
    else
        (q->front)++; return x;
}
```

Data Structures and its Applications

Simple Queue – Implementation of empty and display



```
int empty ( struct queue * q)
{
    if (q->front ==-1)
        return 1;
    return -1;
}
display (struct queue * q)
{
    int i;
    if(q->front== -1) printf("Empty queue..\n") else
    {
        for ( i = q->front; i<=q->rear;i++)
            printf("%d",q->items[i]);
    }
}
```

Data Structures and its Applications

Simple Queue – Another Implementation



Implementation where queue represented as an array , front and rear are separate variables

```
#define MAXQUEUE 100 int q[MAXQUEUE]
```

```
int front, rear
```

```
front = rear = -1
```

Function calls to implement the operations

- insert(x, q, &front, &rear) ;
- remove (q , &front, &rear);
- empty(q, &front)

Data Structures and its Applications

Simple Queue – Another Implementation



Implementation where queue represented as an array ,
front and rear are separate variables

```
int insert(int x,int *q,int *f,int *r)
{
    //check queue overflow if(*r==MAXQUEUE - 1)
    {
        printf("Queue overflow..\n");
        return -1;
    } (*r)++;
    q[*r]=x;
    if(*f==-1)//if first element
        *f=0; return 1;
}
```

Data Structures and its Applications

Simple Queue – Another Implementation



Implementation where queue represented as an array ,
front and rear are separate variables

```
int remove(int *q,int *f,int *r)
{
    int x;
    if(*f==-1)
    {
        printf("Queue empty..\n"); return -1;
    }
    x=q[*f];
    if(*f==*r)//only one element in queue
        *f=*r=-1; else (*f)++;
    return x;
}
```

Data Structures and its Applications

Simple Queue – Another Implementation



Implementation where queue represented as an array ,
front and rear are separate variables

```
display (int *q, int f, int r)
{
    int i;
    if(f==0)
    {
        printf("Empty queue..\n")
    }
    else
    {
        for ( i = f; i<= r; i++)
            printf("%d",q[i]);
    }
}
```

Question 1: At which end of a queue are items typically inserted?

- a) The front of the queue.
- b) The rear of the queue.
- c) Anywhere in the middle.
- d) Both ends simultaneously.

Question 1: At which end of a queue are items typically inserted?

- a) The front of the queue.
- b) The rear of the queue.**
- c) Anywhere in the middle.
- d) Both ends simultaneously.

Question 2: What is the term for an illegal attempt to insert an element into a queue that has reached its maximum size?

- a) Underflow
- b) Deletion error
- c) Overflow
- d) Segmentation fault

Question 2: What is the term for an illegal attempt to insert an element into a queue that has reached its maximum size?

- a) Underflow
- b) Deletion error
- c) Overflow**
- d) Segmentation fault

Question 3: Which of the following is NOT listed as a type of queue in the presentation?

- a) Simple Queue
- b) Circular Queue
- c) Double-ended Queue (Deque)
- d) Stack Queue

Question 3: Which of the following is NOT listed as a type of queue in the presentation?

- a) Simple Queue
- b) Circular Queue
- c) Double-ended Queue (Deque)
- d) Stack Queue

Question 4: In the sequential (array) representation of a simple queue, what are the initial values of front and rear indices?

- a) front = 0, rear = 0
- b) front = -1, rear = -1
- c) front = 0, rear = MAXQUEUE - 1
- d) front = -1, rear = 0

Question 4: In the sequential (array) representation of a simple queue, what are the initial values of front and rear indices?

a) front = 0, rear = 0

b) front = -1, rear = -1

c) front = 0, rear = MAXQUEUE - 1

d) front = -1, rear = 0

Question 5: When implementing the insert operation for a simple queue using an array, if it's the first element being inserted, what value is front initialized to?

- a) -1
- b) 1
- c) 0
- d) MAXQUEUE - 1

Question 5: When implementing the insert operation for a simple queue using an array, if it's the first element being inserted, what value is front initialized to?

a) -1

b) 1

c) 0

d) MAXQUEUE - 1



THANK YOU

Dinesh Singh

Department of Computer Science &
Engineering

dineshs@pes.edu

+91 8088654402