UNIT – II LINEAR DATA STRUCTURES

Lecture N0. 8

Queue

Queue is that type of list in which insertion can take place from one end called (Rear) and deletion can take place from another end called (Front).

Queue works on the principle of first in first out (FIFO) i.e

List come First Serve and Last in last out (LILO).

Daily Life Example

i) The number of cars at police check

ii) People purchase tickets from cinema make a queue.

iii) People deposits their bills in bank make a queue.

Computer Based Example

i) When number of instructions are given for printing to computer, then that instructions will be executed first that was first assign to computer.

Operations on Queue

Different operations can be performed at queue as

1) Create Queue

By using this operation, the structure of the queue is generated.

2) Enqueue

The insertion of element in the queue is called enqueue.

3) Dequeue

The deletion of element from the queue is called dequeue.

4) Empty Queue

It returns Boolean value true, if queue is empty.

5) Full Queue

It returns Boolean value false, if queue is full.

6) Destroy Queue

This operation deletes all the elements of the queue and also deletes the structure of queue from the computer memory.

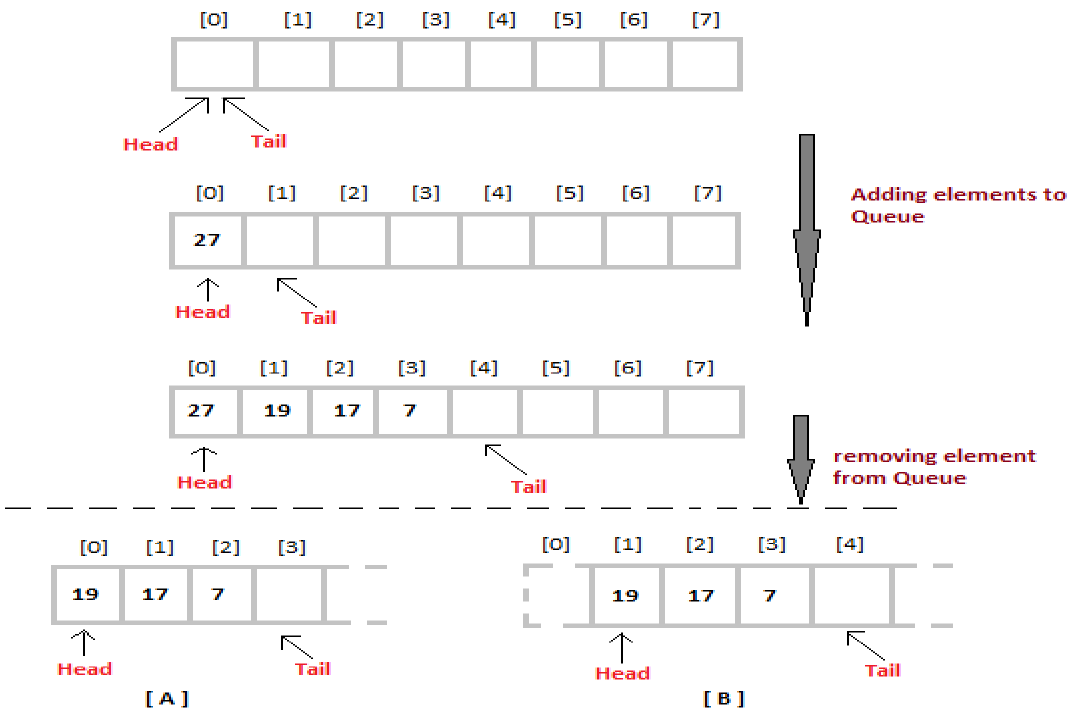
Major Operation on Queue

1) Overflow Condition

When no more elements can be inserted in the queue then this condition is called overflow condition.

2) Underflow Condition

When queue is empty, and no more elements can be deleted from the queue then this condition is called underflow condition.



ALGORITHM FOR THE INSERTION OF ELEMENT IN THE QUEUE

Algorithm enqueue (Q, N, item, Rear, Front)

[This algorithm is used to insert the element in the queue where, Q = Queue Array, N = Total number of elements, Item = which is to be inserted, Front, Rear = Deletion and Insertion pointers]

```
Step-1      [Check for overflow]
            If (Rear >= N) then
            Write (" Over Flow")
            Goto step-5
            End if
Step-2      [Increment the Rear Pointer]
            Rear = Rear + 1
Step-3      [Insert the element at Rear position]
            Q [Rear] = item
Step-4      [Reset Pointer]
            If (Front = 0) then
            Set Front = 1
            End if
            Goto step-1
Step-5      [Finished]
             Exit
```

ALGORITHM FOR THE DELETION OF ELEMENT FROM THE QUEUE

Algorithm Dequeue (Q, item, Front, Rear)

[This algorithm is used for the deletion of element from the queue where Q = Array, Item = which is to be deleted]

```
Step-1      [Check for under flow]
             If ((Front = 0) or (Front > Rear)) then
            Write ("Under Flow")
             Goto step-4
            End if
```

```
Step-2      [Delete the item from queue]
            Item= Q [Front]
Step-3      [Set Front Pointer]
            If (Front = Rear) then
            Front = Rear=0
            Else
                Front = Front + 1
            End if
                Goto step-1
Step-4      [Finished]
              Exit
```

Circular Queue

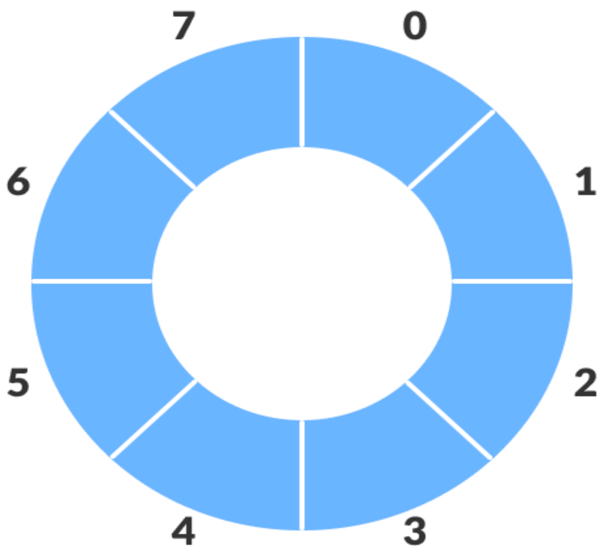
Circular queue provides flexible way for the insertion of element in a circular way.

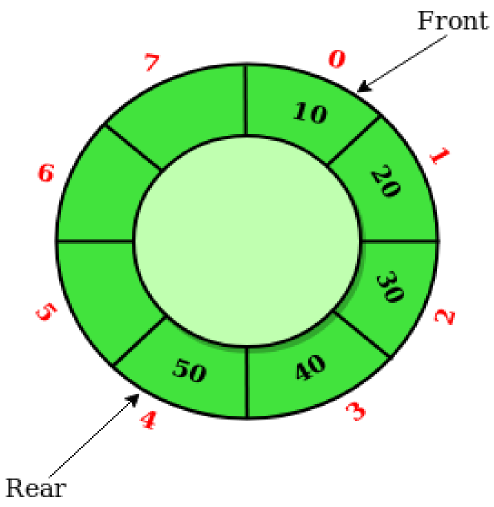
In queue we are nor sure that when the rear pointer goes to an end i.e Rear = N , overflow condition occur, may be the same element from the front are deleted, it is the major drawback of queue.

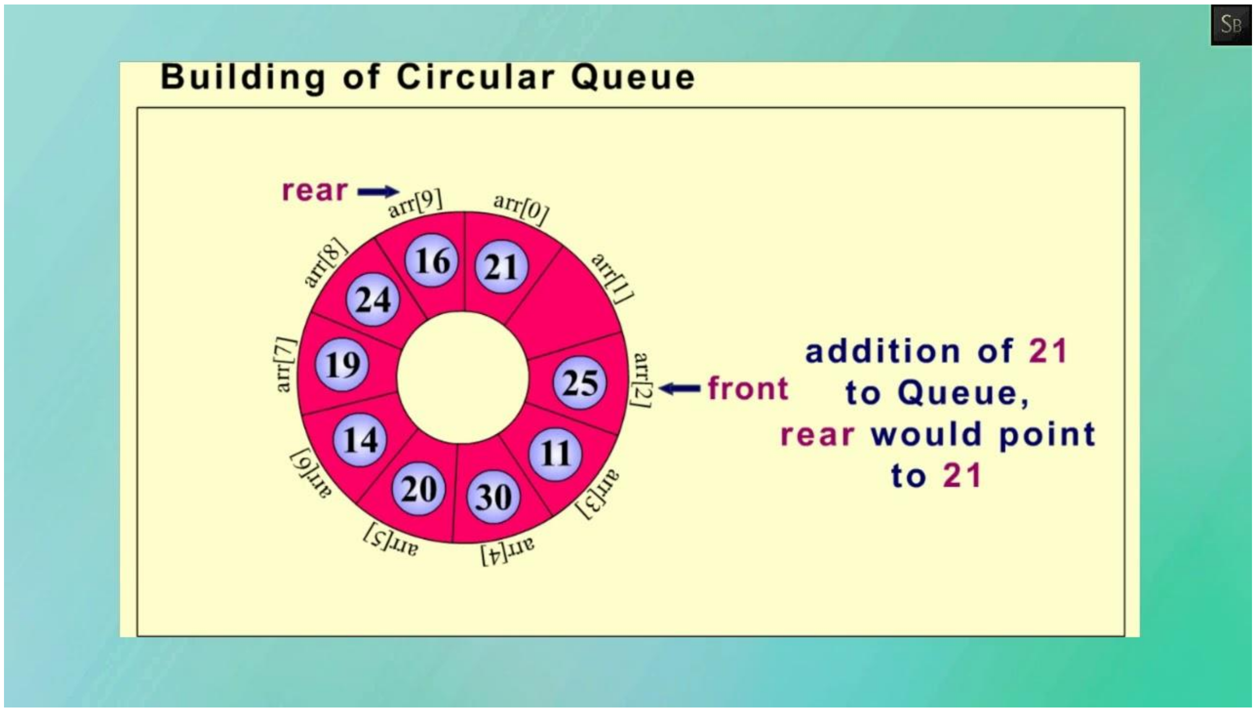
This drawback can be overcome with the help of circular queue.

In circular queue the insertion can be take place by using this sequence

1,2,3, …… N







Building of Circular Queue

addition of 21 to Queue, rear would point to 21

Check for Underflow

When the front and rear pointer become equal that is

Front = Rear = 0

Underflow condition may arise.

Check for Overflow

When Front = 1 and Rear = N then overflow condition may arise. OR

When Front Rear + 1 then overflow condition may arise.

ALGORITHM FOR THE INSERTION OF ELEMENT IN CIRCULAR QUEUE

```
Algorithm CircularInsertion (CQ, N, item, Rear, Front)

        [This algorithm is used to insert the element in CQ, where

          CQ = Array

            N  = Final Limit

         Item  = which is to be inserted

 Front, Rear =  Deletion and Insertion  pointers ]


Step-1          [Check for overflow]

                 If ((Front = 1) && (Rear = N) or (Front = Rear + 1) then

                      Write ("Over Flow")

                      Goto step-4

                End if


Step-2          [Reset Rear Pointer]

                If (Front = Rear =0) then

                        Rear = Front = 1

                Else if (Rear = N) then

                        Rear = 1

                Else

                        Rear = Rear + 1

                End if



Step-3          [Insert the element]

                        CQ [Rear] = item

                Goto step-1
```

```
Step-4          [Finished]
                 Exit
```

ALGORITHM FOR THE DELETION OF ELEMENT FROM CIRCULAR QUEUE

Algorithm DeleteCQ (CQ, item, N, Front, Rear)

[This algorithm is used for the deletion of element from Circular queue where

CQ = Array

Item = which is to be deleted

N = Total number of element

```
Step-1          [Check for under flow]
                 If ((Front = Rear = 0)) then
                    Write ("Under Flow")
                    Goto step-4
                End if

Step-2          [Delete the item from Circular queue]
                        Item  CQ [Front]

Step-3          [Set Front Pointer]
                If (Front = Rear) then
                        Front  Rear 0
                Else if (Front = N) then
                        Front =1
                Else
                        Front  Front + 1
                End if
                        Goto step-1
```

Step-4 [Finished]

Exit