TECHNO INDIA UNIVERSITY
WESTBENGAL

Data Structures & Algorithms
(PCC-CS 301)

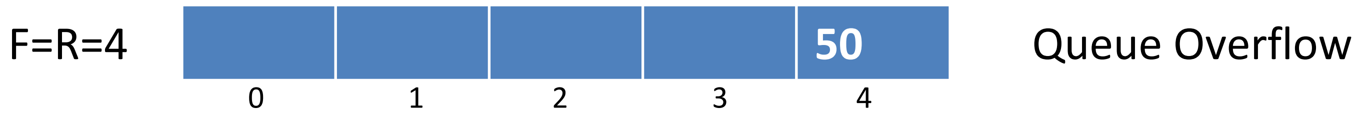
Dr. Debashis Das
Associate Professor
Department of CSE
Techno India University, Kolkata

Topics Covered

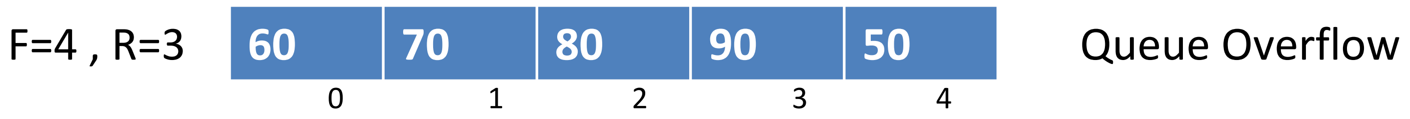
1. Linear Data Structure
    a. Circular Queue

Circular Queue

- Why necessary?
  - Problem in simple Queue implementation
    - Memory utilization is poor
      - Wastage of memory in following case where new data cannot be inserted although maximum cells are vacant



  - Solution in Circular Queue
    - Utilization of the unused spaces



After inserting new data: 60, 70, 80, 90

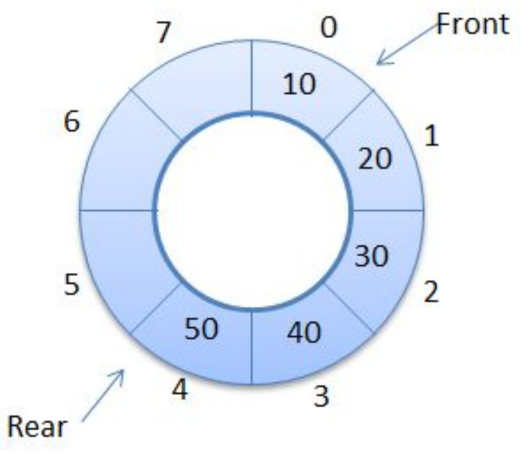
Department of CSE, Techno India University West Bengal

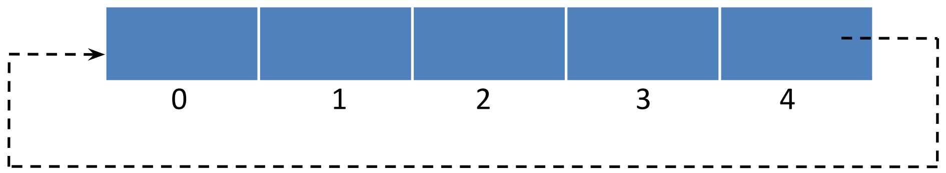
Circular Queue

- Circular Queue
  - Properties
    - It is defined as a **First In First Out (FIFO)** data structure
      - The first data inserted into the Queue to be deleted first
    - The **first element** of the Queue is pointed by **FRONT** pointer
    - The **last element** of the Queue is pointed by **REAR** pointer
    - New element is inserted through **REAR** pointer
    - An element is accessed or deleted through **FRONT** pointer
    - **REAR** pointer can rotate circularly to insert new element into the queue if the initial positions are found vacant

Circular Queue

- Circular Queue
  - Representation





Circular Queue

- Circular Queue
  - Different cases

	0	1	2	3	4	
F=R=null						Empty Queue
F=R=0	10					
F=0 , R=2	10	20	30			
F=0 , R=4	10	20	30	40	50	Queue Overflow
F=1 , R=4		20	30	40	50	
F=1 , R=0	60	20	30	40	50	Queue Overflow

F=FRONT
R=REAR

Circular Queue

- Circular Queue
  - Operations
    - ENQUEUE (data insertion into queue)
    - DEQUEUE (data deletion from queue)

Primary operation

- Front / Display (showing element of queue)
- QueueSize (returns the total element)
- IsFullQueue (checks if Queue is overflow)
- IsEmptyQueue (checks if Queue is underflow)

Auxiliary operation

Circular Queue

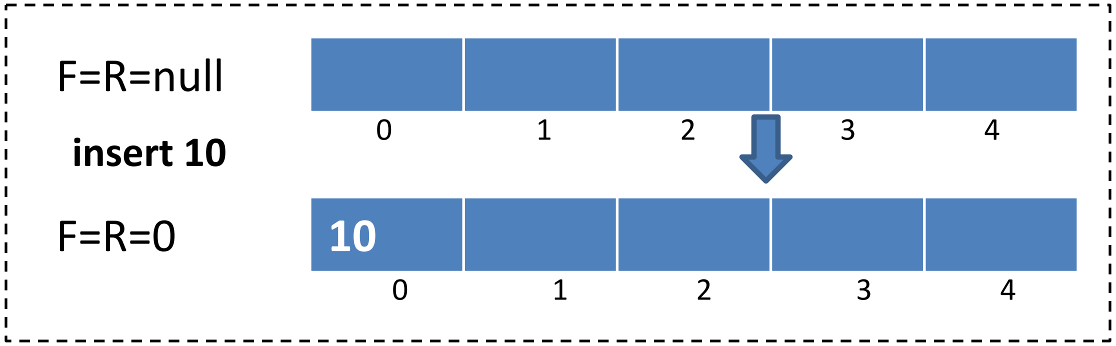
- Operation
  - ENQUEUE
    - This function inserts one element at the REAR position of the Queue if it is not full

```
void ENQUEUE(data)
{
  if  IsFullQueue = TRUE
     print  Q is full
  else
     if   IsEmptyQueue = TRUE
          F := 0 and R:= 0
     else
         if F > 0 and R = Max_Size
            R:= 0
         else
            R:=R+1
     Q(R) := data
}
```

```
void ENQUEUE(data)
{
  if  IsFullQueue = TRUE
     print  Q is full
  else
     if   IsEmptyQueue = TRUE
          F := 0 and R:= 0
     else
        R:= (R+1) mod (Max_Size)
   Q(R) := data
}
```

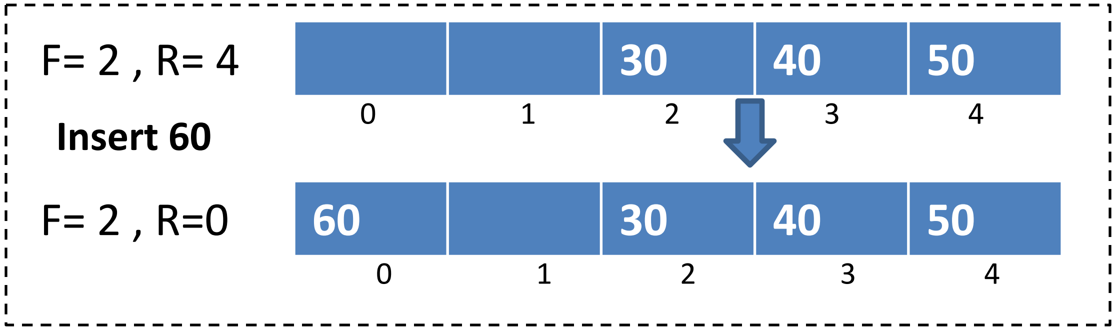
Circular Queue

- Operation
  - ENQUEUE (example)



insert 20 20

R= (0+1) mod (Max_Size)
= (0+1) mod (5)
= 1 mod 5
= 1



R= (R+1) mod (Max_Size)
= (4+1) mod (5)
= 5 mod 5
= 0

Circular Queue

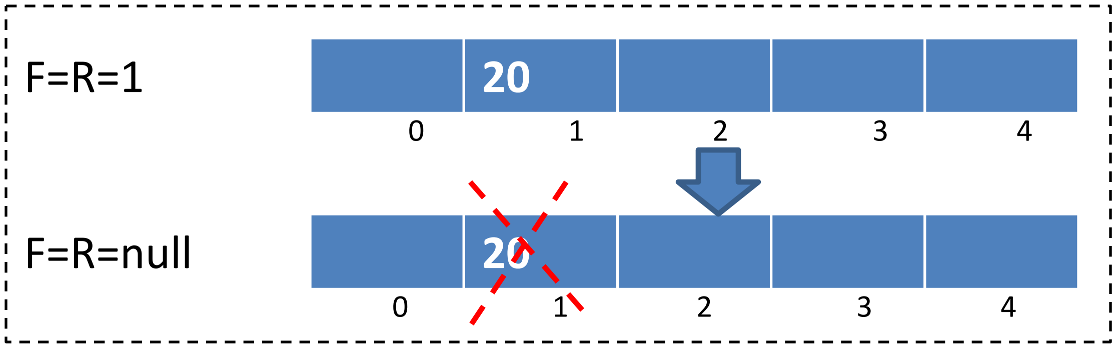
- Operation
  - DEQUEUE
    - This operation deletes the front element of the Queue if it is not empty

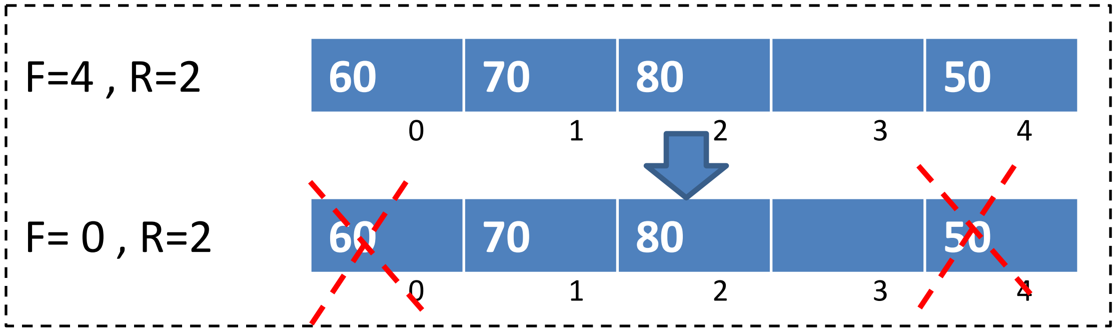
```
int DEQUEUE()
{
 if  IsEmptyQueue = TRUE
   return NULL
 else
     data := Q(F)
     if   F = R
        F := null and R:= null
     else
          if  F = Max_Size
             F := 0
          else
             F := F+1
   return data
}
```

```
int DEQUEUE()
{
 if  IsEmptyQueue = TRUE
   return NULL
 else
     data := Q(F)
     if   F = R
        F := null and R:= null
     else
        F := (F+1) mod (Max_Size)
   return data
}
```

Circular Queue

- Operation
  - DEQUEUE (example)





F= (F+1) mod (Max_Size)
= (4+1) mod (5)
= 5 mod 5
= 0

Dequeue()

F= (F+1) mod (Max_Size)
= (0+1) mod (5)
= 1

F=1 , R=2

Circular Queue

- Operation
  - Front / Display
    - Front function displays the front element of the Queue
    - All elements can also be displayed through an auxiliary pointer without shifting FRONT or REAR

```
int Front()
{
  if  IsEmptyQueue = TRUE
    return NULL
  else
    return Q(F)
}
```

```
void Display()
{
  if  IsEmptyQueue = TRUE
    print  Q is empty
  else
     for i= F to R
        print  Q(i)
}
```

Circular Queue

- Operation
  - QueueSize
    - This function returns the counting of elements present in the current queue

```
int QueueSize()
{
 if  F = null and R = NULL
    return 0
 else
     for i = F to R
        count := count +1
     return count
}
```

Circular Queue

- Operation
  - IsFullQueue
    - This function checks whether the Queue is full or not
    - We cannot insert data into Queue if it is full

```
Boolean IsFullQueue()
{
 if  R = Max_Size or F = R+1
   return TRUE
 else
   return FALSE
}
```

F=0 , R=4

10	20	30	40	50
0	1	2	3	4

Queue Overflow

F=1 , R=0

60	20	30	40	50
0	1	2	3	4

Circular Queue

- Operation
  - IsEmptyQueue
    - This function checks whether the Queue is empty or not
    - We cannot delete or display the Queue if it is empty

```
Boolean IsEmptyQueue()
{
 if  F = null and R = null
   return TRUE
 else
   return FALSE
}
```

TECHNO INDIA UNIVERSITY
WESTBENGAL

Circular Queue

- Operation: complexity

Operation	Time Complexity
Enqueue()	O(1)
DeQueue()	O(1)
Display()	O(n)
QueueSize()	O(n)
IsFullQueue()	O(1)
IsEmptyQueue()	O(1)

Circular Queue

- Circular Queue
  - Applications
    - Memory management
      - To maintain the list of unused memory. As soon as any memory gets free by any process, it is added at rear end in the circular queue
    - Computer controlled traffic system
      - Circular queue is used to switch on the traffic lights one-by-one repeatedly
    - CPU scheduling (in operating system)
      - Operating system maintains a circular queue to store the ready (or waiting for some event to occur) processes to be executed

Queries?