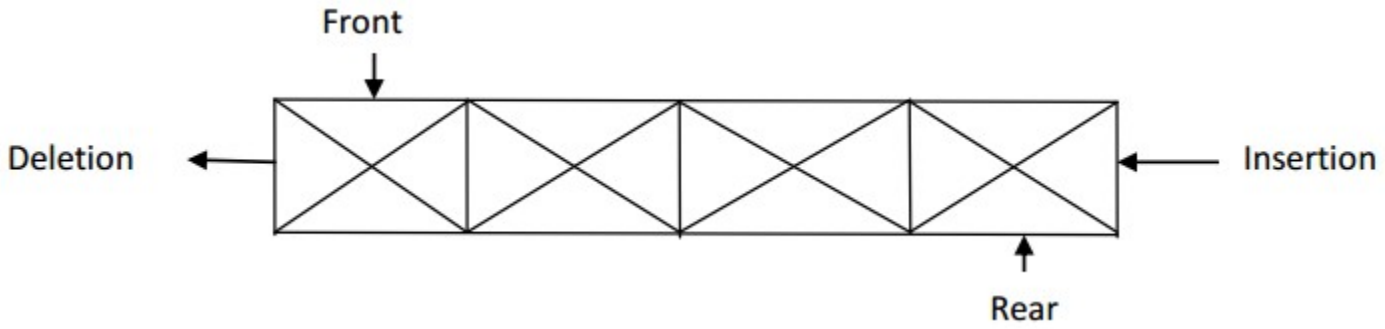
QUEUE

Concepts and Operations

Prof. Siddharth Shah

What is Queue?

- A linear list which permits deletion to be performed at one end of the list and insertion at the other end is called queue.
- The information in such a list is processed in FIFO (first in first out) or FCFS (first come first served) manner.
- The process to add an element into queue is called **EnQueue** performed at **rear / tail** end.
- The process of removal of an element from queue is called **DeQueue** performed at **front / head** end.
- **Example:** Checkout line at Supermarket Cash Register where the first person in line is (usually) the first to be checked out.



EnQueue

Algorithm : QINSERT (Q, F, R, N,Y)

Given F and R, pointers to the front and rear elements of a queue, a queue Q consisting of N elements, and an element Y, this procedure inserts Y at the rear of the queue. Prior to the first invocation of the procedure, F and R have been set to zero.

1. [Overflow]

If R >= N Then

write ('OVERFLOW')

Return

2. [Increment REAR pointer]

R ← R + 1

3. [Insert element]

Q[R] ← Y

4. [Is front pointer properly set]

If F=0 Then

F ← 1

Return

DeQueue

Algorithm : QDELETE (Q, F, R)

Given the pointers, F and R, to the front and rear elements of a queue Q respectively. This function deletes and returns the oldest element of the queue. Y is a temporary variable.

1. [Underflow]

If F= 0 Then

write ('UNDERFLOW')

Return (0)

2. [Delete element]

Y ← Q[F]

3. [Queue empty?]

IF F = R Then

F ← R ← 0

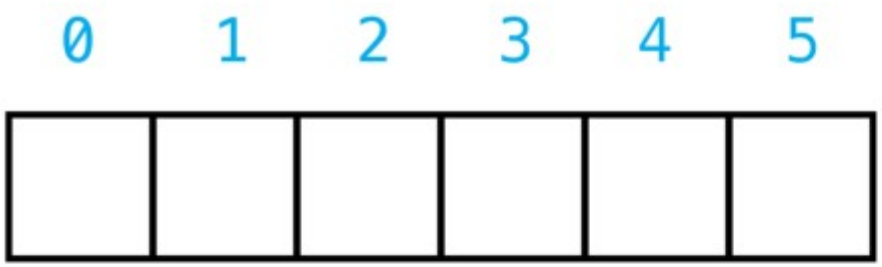
Else

F ← F + 1

4. [Return element]

Return (Y)

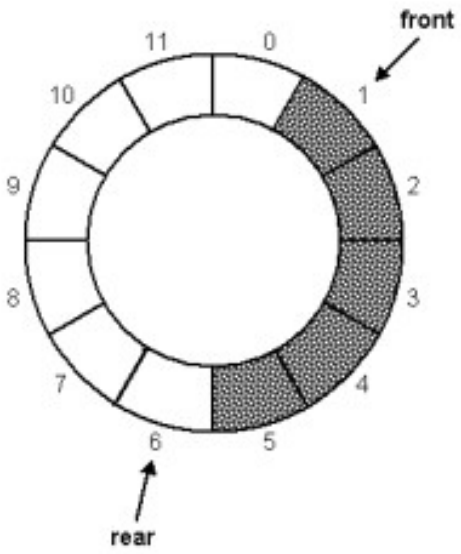
Queue : Example




```
Q = Queue(6)
Q.Enqueue(5)
Q.Enqueue(10)
Q.Enqueue(15)
Q.Enqueue(20)
Q.Enqueue(25)
Q.Enqueue(30)
Q.Dequeue()
Q.Dequeue()
Q.Enqueue(35)
Q.Enqueue(40)
```

Circular Queue

- A more suitable method of representing simple queue which prevents an excessive use of memory is to arrange the elements **Q[1], Q[2] ….,Q[n]** in a circular fashion with **Q[1]** following **Q[n]**, this is called circular queue
- In circular queue the last node is connected back to the first node to make a circle.
- Both the front and the rear pointers points to the beginning of the array.
- It is also called as “Ring buffer”.



EnQueue

Algorithm : CQINSERT (F, R, Q, N,Y)

Given pointers to the front and rear of a circular queue, F and R, a vector Q consisting of N elements and an element Y, this procedure inserts Y at the rear of the queue. Initially, F and R are set to zero.

1. **[Reset rear pointer]**
   If R = N Then
   R ← 1
   Else
   R ← R + 1
2. **[Overflow?]**
   If F = R Then
   write ('OVERFLOW')
   If R = 1 Then
   R ← N
   Else
   R ← R - 1
   Return
3. **[Insert element]**
   Q[R] ← Y
4. **[Is front pointer properly set]**
   If F=0 Then
   F ← 1
   Return

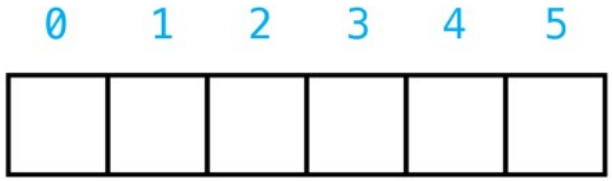
DeQueue

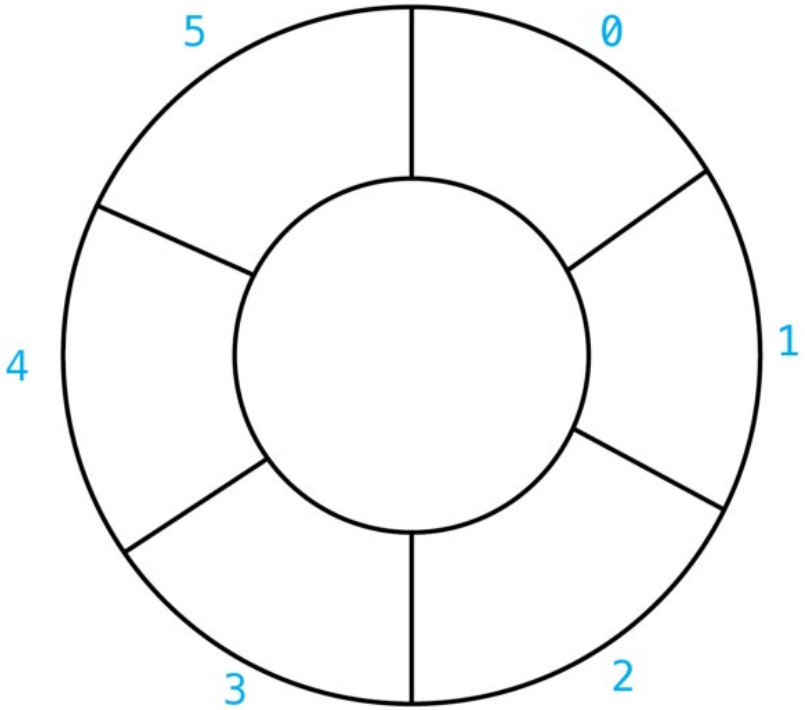
Algorithm : CQDELETE (F, R, Q, N)

Given F and R, pointers to the front and rear of a circular queue, respectively, and a vector Q consisting of N elements, this function deletes and returns the oldest element of the queue. Y is a temporary variable.

1. **[Underflow]**
   If F= 0 Then
   write ('UNDERFLOW')
   Return (0)
2. **[Delete element]**
   Y ← Q[F]
3. **[Queue empty?]**
   If F = R Then
   F ← R ←0
   Return (Y)
4. **[Increment front pointer]**
   If F = N Then
   F ← 1
   Else
   F ← F + 1
   Return (Y)

Circular Queue : Example

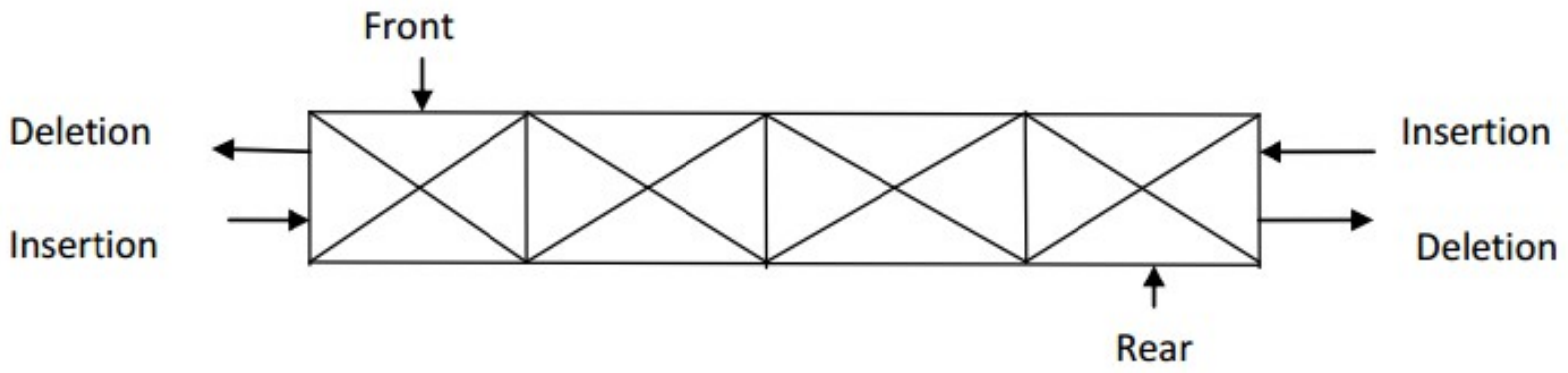





```
Q = circularQueue(6)
Q.Enqueue(5)
Q.Enqueue(10)
Q.Enqueue(15)
Q.Enqueue(20)
Q.Enqueue(25)
Q.Enqueue(30)
Q.Dequeue()
Q.Dequeue()
Q.Enqueue(35)
Q.Enqueue(40)
```

Deque

- A **Deque** (**double ended queue**) is a linear list in which insertion and deletion are performed from the either end of the structure.
- There are two variations of Deque
  - Input restricted Deque allows insertion at only one end
  - Output restricted Deque allows deletion from only one end
- Such a structure can be represented by following fig.



Operation on Deque

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

Applications of Deque

- Since Deque supports both stack and queue operations, it can be used as both.

- **Applications:**

    - Pallindrome checker

    - Undo-redo operations in software applications

Priority Queue

- A priority queue is an abstract data type that behaves similarly to the normal queue except that each element has some priority.

- The element with the highest priority moved to the front of the queue and removed first.

- The priority queue supports only comparable elements, which means that the elements are either arranged in an ascending or descending order.

Characterstics of a Priority Queue

- A priority queue is an extension of a queue that has the following characteristics:

1. Every element in a priority queue has some priority associated with it.

2. An element with the higher priority will be deleted before the deletion of the lesser priority.

3. If two elements in a priority queue have the same priority, they will be arranged using the FIFO principle.

Types of Priority Queue

- There are two types of priority queue:

1. Ascending order priority queue: lower the value higher the priority

2. Descending order priority queue: higher the value higher priority

Applications of Priority Queue

- It is used in the Dijkstra's shortest path algorithm.
- It is used in prim's algorithm.
- It is used in data compression techniques like Huffman code.
- It is used in heap sort.
- It is also used in operating system like priority scheduling, load balancing and interrupt handling.