

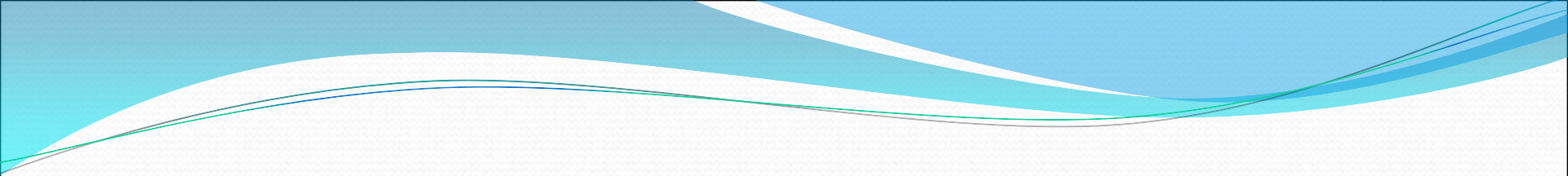
# Queues

- Queues:
  - Array and list representation,
  - Operations (traversal, insertion and deletion)
- Priority queues and Deques:
  - Array and List representations

# Definition of Queue

- 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 (the *rear* of the queue).
- The first element inserted into the queue is the first element to be removed. For this reason a queue is sometimes called a **FIFO** (first-in first-out) list as opposed to the stack, which is a **LIFO** (last-in first-out).



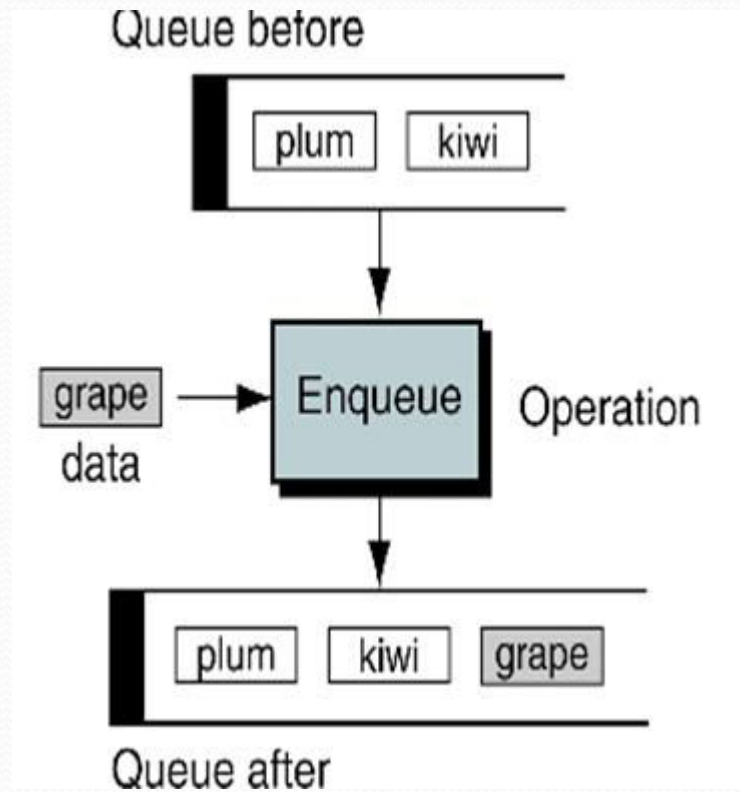
- 
- It is a linear data structure in which new elements are inserted at one end and elements are deleted from the other end.
  - This mechanism is called First-In-First-Out (FIFO).
  - Placing an item in a queue is called “insertion or **enqueue**”, which is done at the end of the queue called “rear”(or tail).
  - Removing an item from a queue is called “deletion or **dequeue**”, which is done at the other end of the queue called “front”(head).



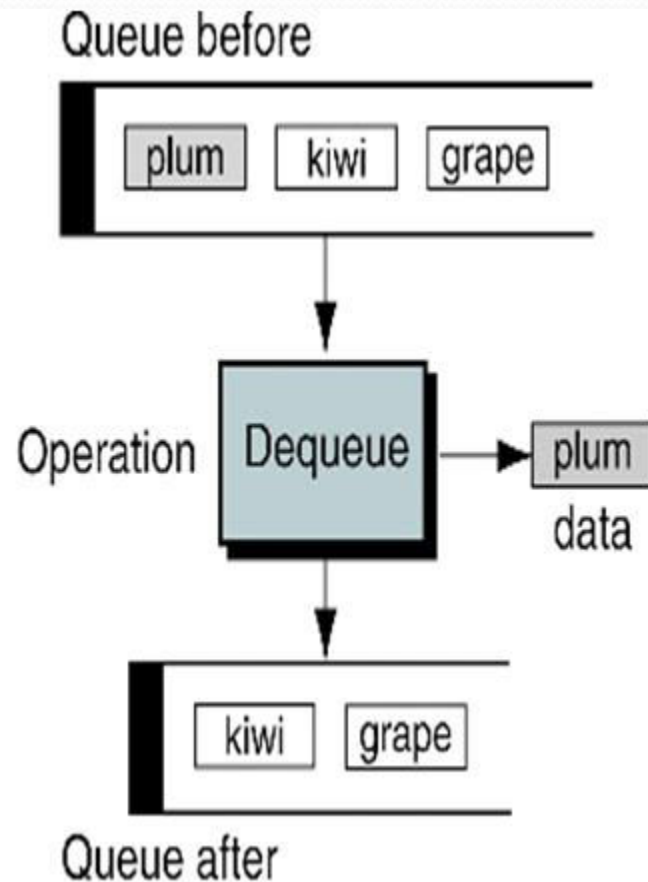
- **ENQUEUE:**The enqueue operations is used to insert a new element to the rear of the queue .After the enqueue operation, the element that has just been inserted becomes the rear, Each enqueue operations increases the number of elements in the queue by one.
- **DEQUEUE:**This is used to remove the existing element from the front of the queue.After removing the element from the front,the next older element becomes the front of the queue.Each dequeue operation reduces the number of element in the queue by one.



# Enqueue

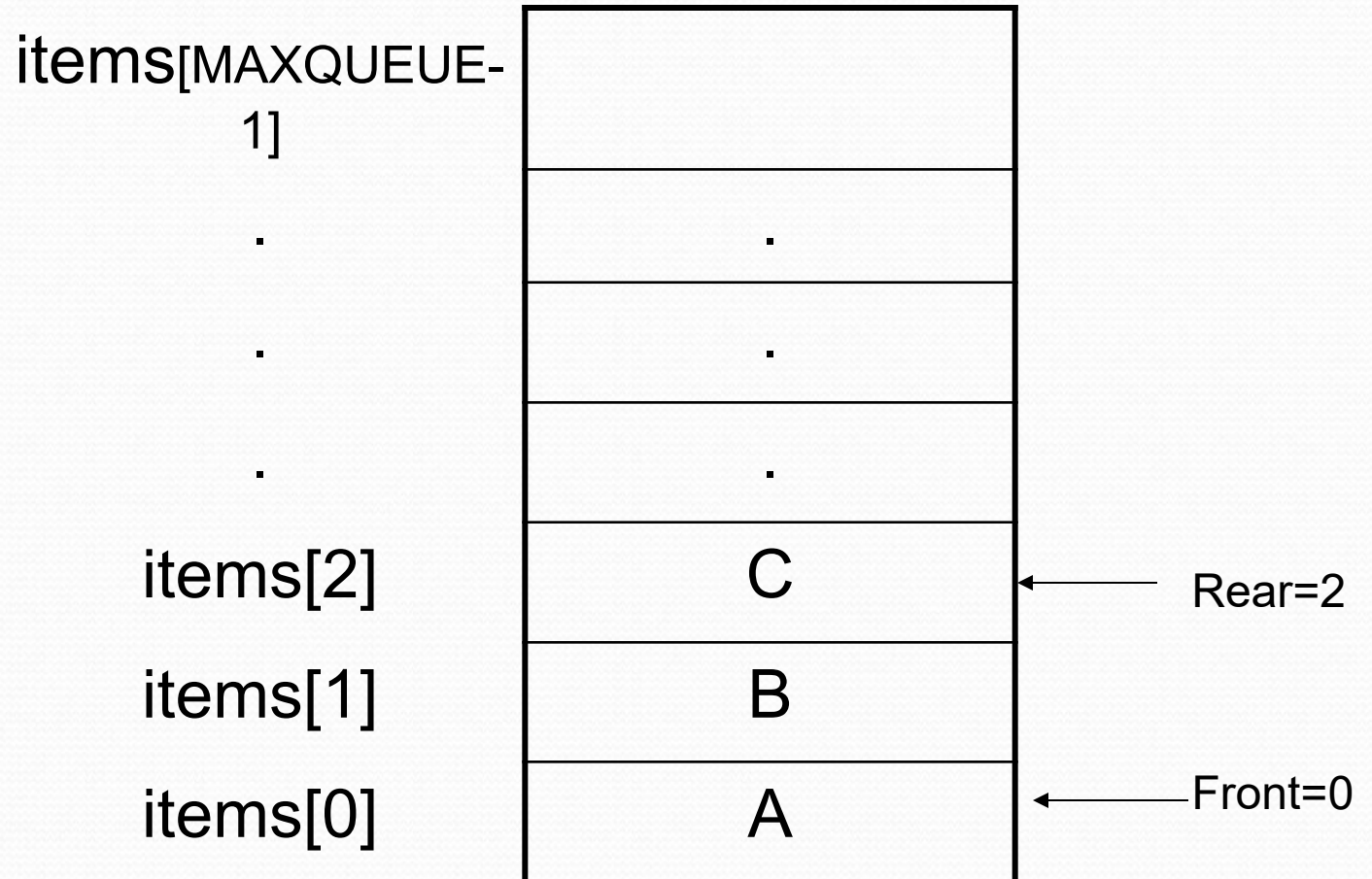


# Deque



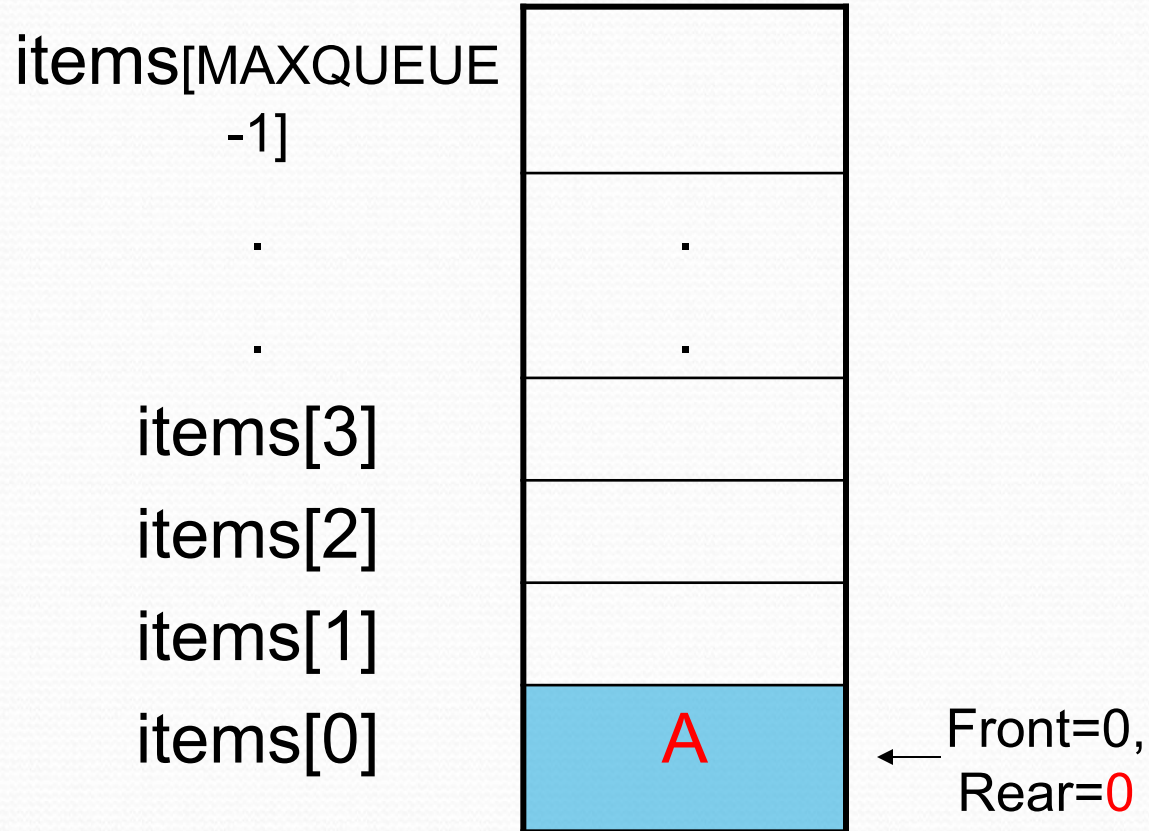


# Queue



# Insert an item A

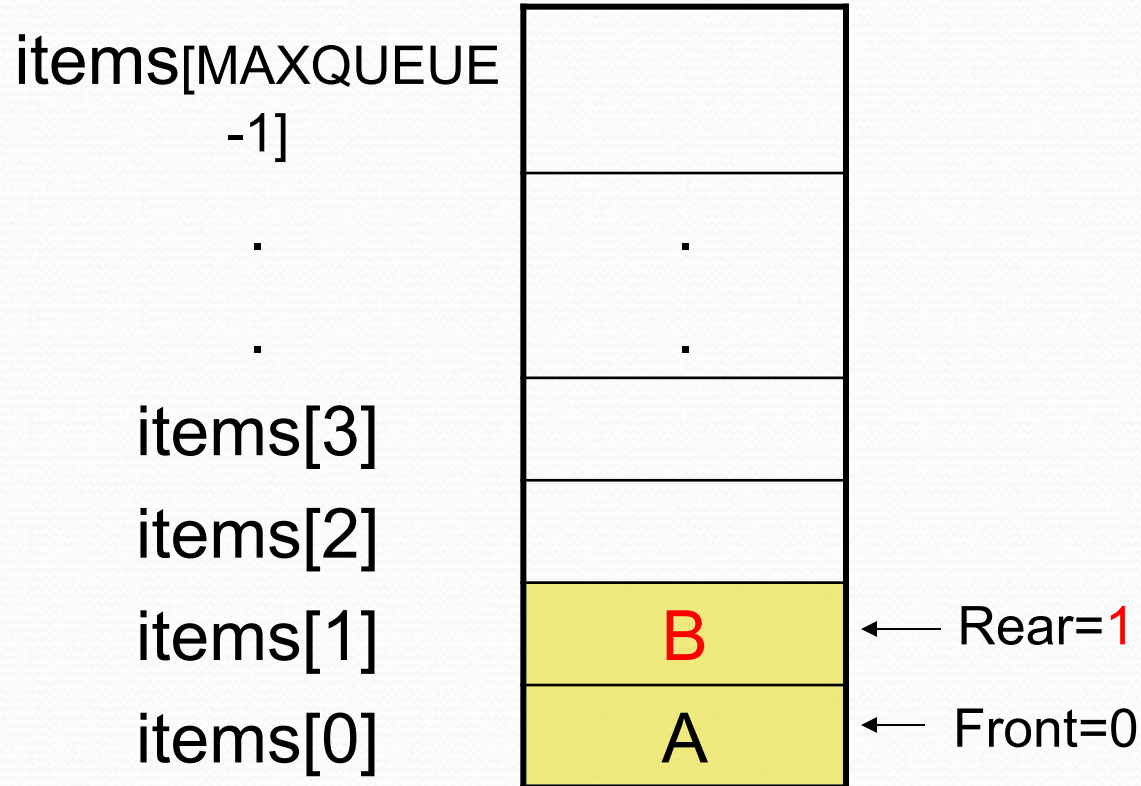
- A new item (*A*) is *inserted* at the *Rear* of the queue





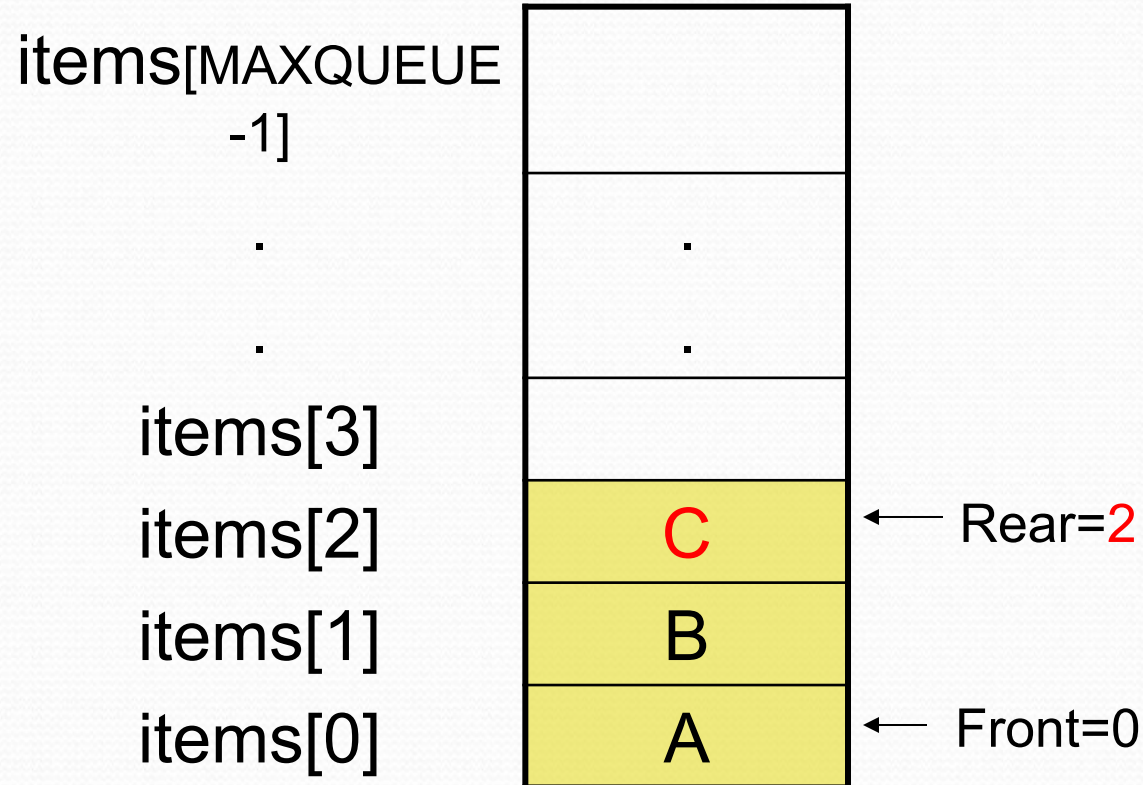
# Insert an item B

- A new item (*B*) is *inserted* at the *Rear* of the queue



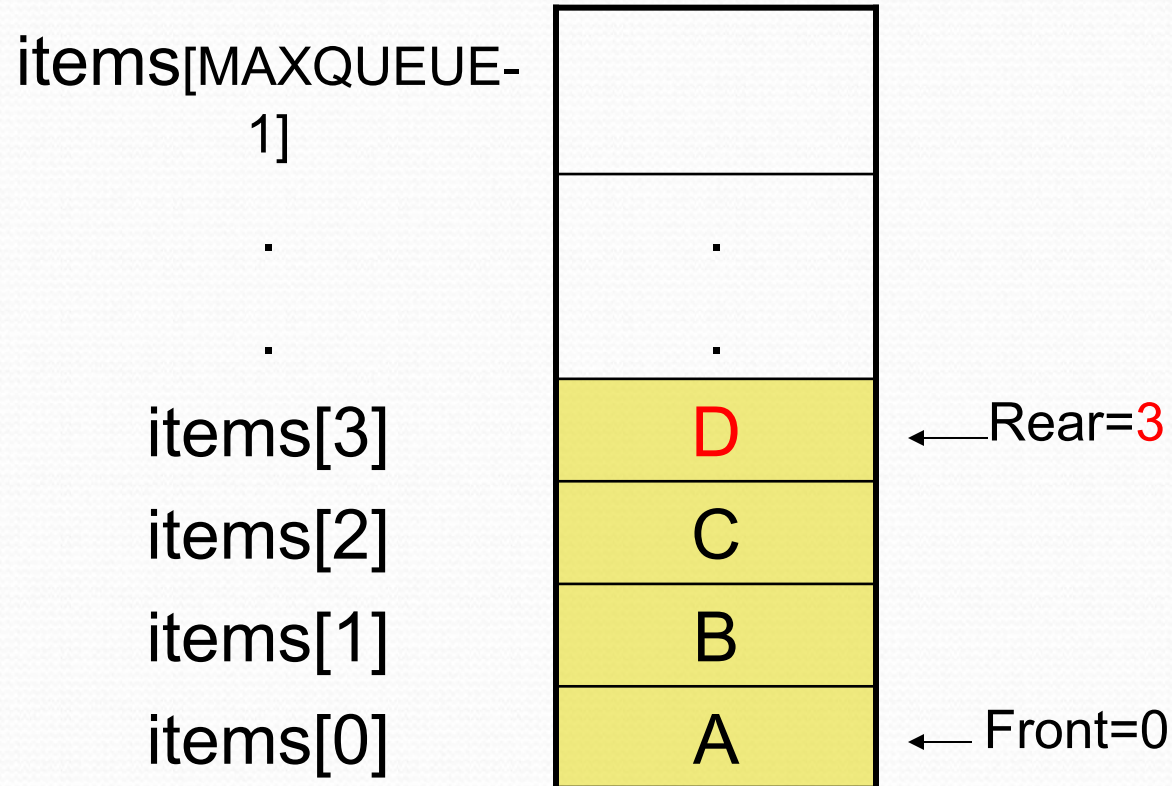
# Insert an item C

- A new item (*C*) is *inserted* at the *Rear* of the queue



# Insert an item D

- A new item (*D*) is *inserted* at the *Rear* of the queue





# Insert Operation(Array)

QINSERT(Queue, N, FRONT, REAR, ITEM)

1. If  $\text{FRONT} = 1$  and  $\text{REAR} = N$ , or  $\text{FRONT} = \text{REAR} + 1$ , then :  
write OVERFLOW, and Return.
2. If  $\text{FRONT} = \text{NULL}$ , then:  
Set  $\text{FRONT} := 1$  and  $\text{REAR} := 1$ .  
Else if  $\text{REAR} = N$ , then:  
Set  $\text{REAR} := 1$ .  
Else :  
Set  $\text{REAR} := \text{REAR} + 1$ .
3. Set  $\text{Queue}[\text{REAR}] := \text{ITEM}$ .
4. Exit.

# Delete A

- An item (A) is deleted from the *Front* of the queue

items[MAXQUEUE-1]

.

.

.

items[3]

D

Rear=3

items[2]

C

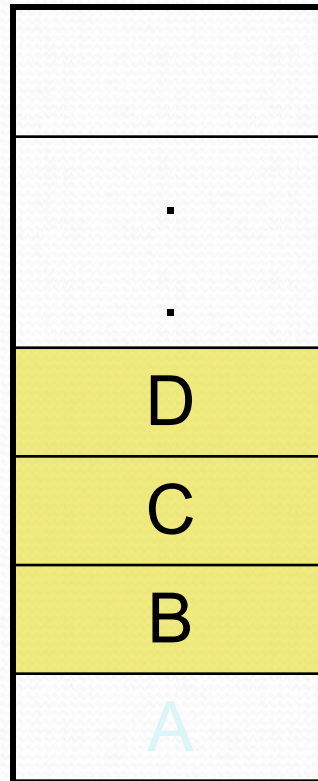
items[1]

B

Front=1

items[0]

A



# Delete B

- An item (*B*) is deleted from the *Front* of the queue.

items[MAXQUEUE-1]

.

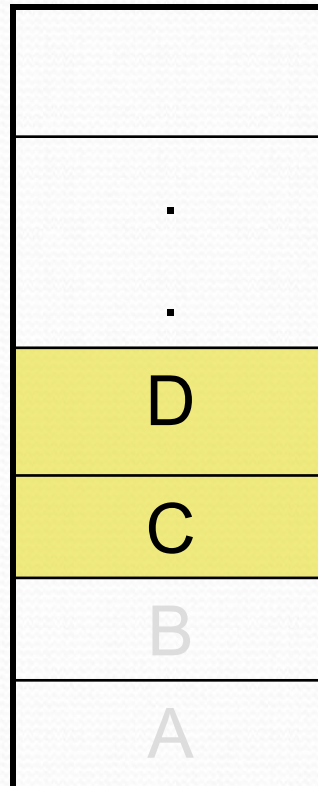
.

items[3]

items[2]

items[1]

items[0]



Rear=3

Front=2



# Delete C

- An item (C) is deleted from the *Front* of the queue

items[MAXQUEUE-1]

.

.

.

items[3]

D

Front=Rear=3

items[2]

C

items[1]

B

items[0]

A

# Delete D

- An item (*A*) is deleted from the *Front* of the queue.

items[MAXQUEUE-1]

.

items[3]

items[2]

items[1]

items[0]

.
D
C
B
A

Front=Rear=-1

# Delete Operation(Array)

QDELETE(Queue, N, FRONT, REAR, ITEM)

1. If  $FRONT = NULL$  then :

    write: UNDERFLOW, and Return.

2. Set  $ITEM := Queue[FRONT]$ .

3. If  $FRONT = REAR$ ,

    Set  $FRONT := NULL$  and  $REAR := NULL$

Else if  $FRONT = N$ , then:

    Set  $FRONT := 1$ .

Else :

    Set  $FRONT := FRONT + 1$ .

4. Return.



# Insert Operation(LL)

LINKQ\_INSERT(INFO, LINK, FRONT, REAR, AVAIL, ITEM)

1. If  $AVAIL = NULL$ , then :

write **OVERFLOW**, and Exit.

2. Set  $NEW := AVAIL$  and  $AVAIL := LINK[AVAIL]$

3. Set  $INFO[NEW] := ITEM$  and  $LINK[NEW] = NULL$

4. If  $FRONT = NULL$ ,  $FRONT := REAR := NEW$

Else :

set  $LINK[REAR] := NEW$  and  $REAR := NEW$

5. Exit

# Delete Operation(LL)

LINKQ\_DELETE(INFO, LINK, FRONT, REAR, AVAIL, ITEM)

1. If  $FRONT = NULL$ , then :  
    write UNDERFLOW, and Exit.
2. Set  $TEMP := FRONT$
3. Set  $ITEM := INFO[FRONT]$
4.  $FRONT := LINK[FRONT]$
5.  $LINK[TEMP] = AVAIL$  and  $AVAIL = TEMP$
6. Exit

# Priority Queue

- More specialized data structure.
- Similar to Queue, having front and rear.
- Items are removed from the front.
- Items are ordered by key value so that the item with the lowest key (or highest) is always at the front.
- Items are inserted in proper position to maintain the order.



# DEQueue

- It is a double-ended queue.
- Items can be inserted and deleted from either ends.
- More versatile data structure than stack or queue.
- E.g. policy-based application (e.g. low priority go to the end, high go to the front)

# DEQUE

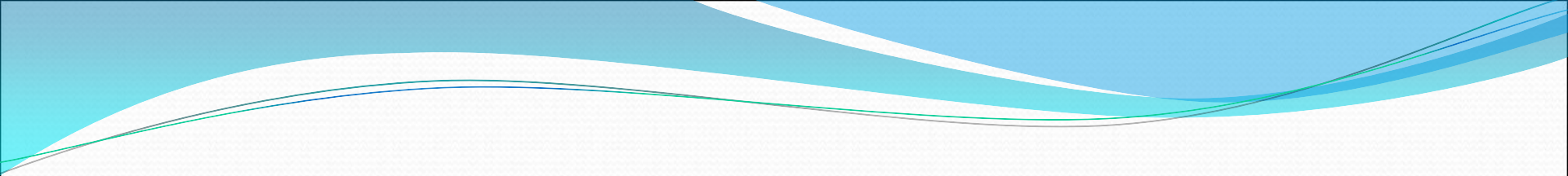
- A deque(double ended queue) is one of variation of a queue.
- A deque is a linear link that allows insertion and deletion at either end,i.e. at the front or rear of the queue but not in the middle.
- Generally the leftmost element represents the front and the rightmost elements represent the rear.



- The four basic operations that can be performed on deque are **insertLeft**, **insertRight**, **deleteLeft** and **deleteRight**.
- The **insertLeft** and **deleteLeft** inserts and deletes an element to/from the left of the deque respectively
- The **insertRight** and **deleteRight** inserts and deletes an element to/from the right of the deque respectively.
- There are multiple ways of implementing a deque, but most commonly used implementations are circular array and doubly linked list.



- Deque can be classified as  
input restricted deque.  
output restricted deque.
- In **input restricted deque**, the insertion of an element is restricted to one end only but deletion can take place at both the ends.
- The operations that are possible on input restricted deque are insertRight, deleteRight and deleteLeft.

- 
- An **output restricted deque** is a deque which allows deletion at only one end of the list but allow insertion at both ends of the list.
  - The operations that are possible on output restricted deque are deleteLeft, insertRight and insertLeft.



# Thank You