DATA STRUCTURES AND ALGORITHM

Module

5

- To understand and apply the concept of queues
- To identify the application of three major variation of the queue

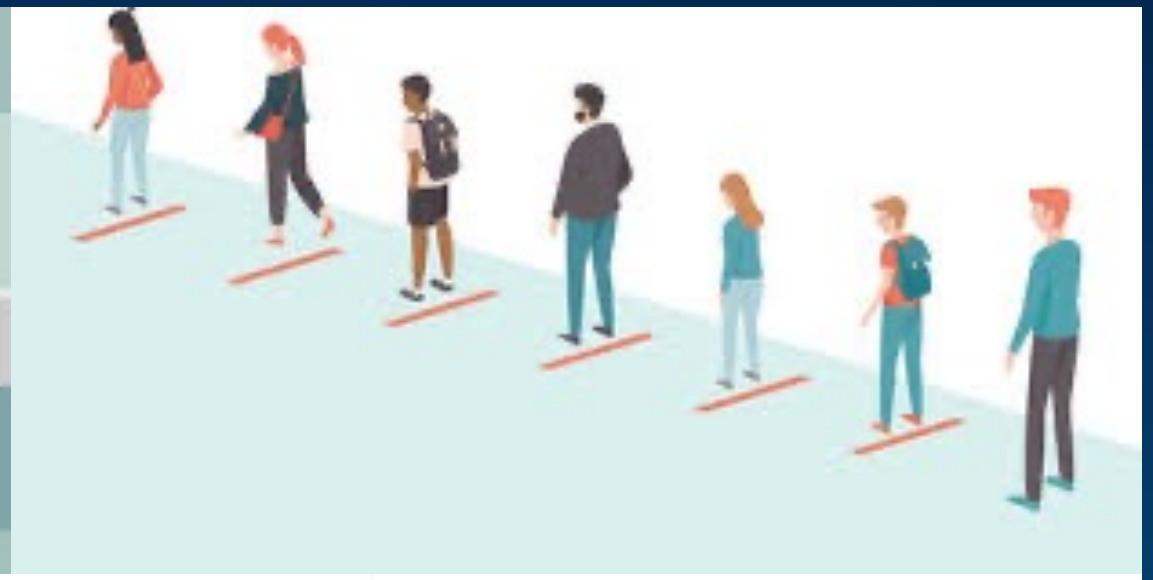
QUEUES

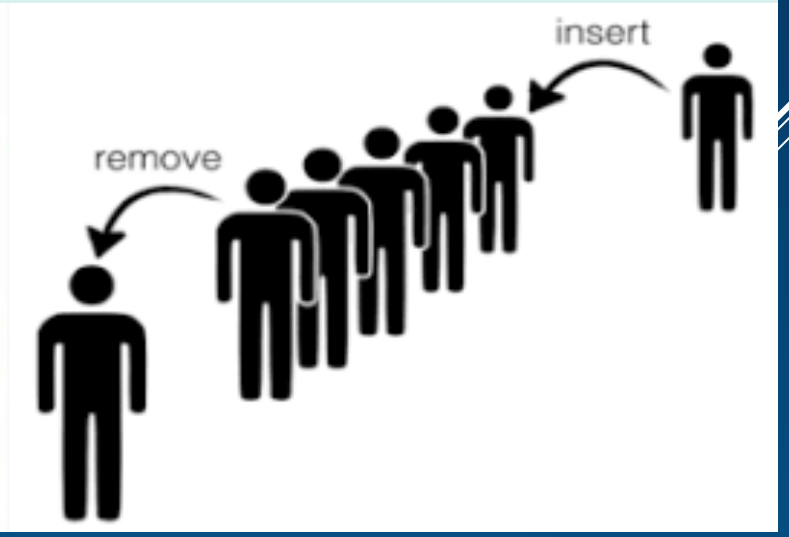
QUEUES

- A container of elements that are inserted and removed according to the ***first-in first-out*** (***FIFO*** |***LILO***) principle. Elements can be inserted in a queue at any time, but only the element that has been in the queue the longest can be removed at any time.
- Elements enter / add to the queue at the **rear** and are removed from the **front**.

QUEUES

QUEUES





QUEUES

APPLICATIONS OF A LINEAR QUEUE

- Time-sharing system
- A CPU endowed with memory resources
- Round robin techniques for processor scheduling is implemented using queue.
- Printer server routines (in drivers) are designed using queues.
- All types of customer service software (like Railway/Air ticket reservation) are designed using queue to give proper service to the customers.

QUEUES

THE QUEUES ABSTRACT DATA TYPE

- **enqueue(e) :** Insert element e at the rear of the queue.
- **dequeue():** Remove element at the front of the queue; an error occurs if the queue is empty.
- **front():** Return, but do not remove, a reference to the front element in the queue; an error occurs if the queue is empty.
- **size():** Return the number of elements in the queue
- **empty():** Return true if the queue is empty and false otherwise.

QUEUES

INSERTING AN ELEMENT INTO THE QUEUE

1. Initialize front= -1 rear = –1
2. Input the value to be inserted and assign to variable "data"
3. If (rear >= SIZE)
   (*a*) Display "Queue overflow"
   (*b*) Exit
4. Else
   (*a*) Rear = rear +1
5. Q[rear] = data
6. Exit

QUEUES

INSERTING AN ELEMENT INTO THE QUEUE

1. Initialize front= -1 rear = –1
2. Input the value to be inserted and assign to variable "data"
3. If (rear >= SIZE)
   (*a*) Display "Queue overflow"
   (*b*) Exit
4. Else
   (*a*) Rear = Rear +1
5. Q[1] = data
6. Exit

20	10	1		
0	1	2	3	4

QUEUES

DELETING AN ELEMENT INTO THE QUEUE

1. If (rear< front)
   (*a*) Front = 0, rear = –1
   (*b*) Display "The queue is empty"
   (c) Exit
2. Else
   (*a*) Data = Q[1]
3. Front = front +1
4. Exit

	10	1		
0	1	2	3	4

QUEUES

Size: 3

OPERATION	OUTPUT	QUEUES		
		0	1	2
enqueue(5)		5		
enqueue(3)		5	3	
front()	5	5	3	
size()	2	5	3	
dequeue()			3	
enqueue(7)			3	7
dequeue()				7
front()	7			7
dequeue()				
dequeue()	underflow			
empty()	true			

QUEUES

Size: 2

OPERATION	OUTPUT	QUEUES (0 1)
enqueue(5)		
enqueue(3)		
front()		
size()		
dequeue()		
enqueue(7)		
dequeue()		
front()		
dequeue()		
dequeue()		
empty()		

CIRCULAR QUEUES

- In circular queues the elements Q[0],Q[1],Q[2] …. Q[*n – 1] is represented in a circular* fashion with Q[1] following Q[n].

- The insertion of a new element is done at the very first location of the queue if the last location at the queue is full

CIRCULAR QUEUES

ALGORITHM OF CIRCULAR QUEUES

Let Q (Queue) be the array of some specified size **SIZE**.

- **FRONT** the element is deleted.
- **REAR** the inserted was made.
- **DATA** is the element to be inserted.

CIRCULAR QUEUES

INSERTING AN ELEMENT TO CIRCULAR QUEUE

1. Initialize FRONT = -1; REAR = -1
2. REAR = (REAR + 1) % SIZE
3. If (FRONT is equal to REAR)
   (a) Display "Queue is full" (b) Exit
4. Else
   Input the value to be inserted and assign to variable "DATA"
5. If (FRONT is equal to – 1)
   FRONT = 0 and REAR = 0
6. Q[REAR] = DATA
7. Repeat steps 2 to 5 , to add elements
8. Exit

CIRCULAR QUEUES

DELETING AN ELEMENT FROM A CIRCULAR QUEUE

```
If (FRONT == – 1)
        Display "Queue is empty"
        Then Exit
Else
        DATA = Queue[FRONT]
If (REAR is equal to FRONT)
        FRONT = –1 ;
        REAR = –1
Else
        FRONT = (FRONT +1) % SIZE
```

QUEUES

MS WORD

QUEUES

There are **THREE MAJOR VARIATIONS** in a simple queue.

	 Lynda.com	 Dreamstime.com
Circular Queue	Double Ended Queue	Priority Queue

DOUBLE-ENDED QUEUES

Insertion and deletion at both the front and the rear of the queue.

THE DEQUE ABSTRACT DATA TYPE

insertFront(e):

Insert a new element **e** at the beginning of the deque.

eraseFront():

Remove the first element of the deque; an error occurs if the deque is empty.

insertBack(e):

Insert a new element **e** at the end of the deque.

eraseBack():

Remove the last element of the deque; an error occurs if the deque is empty.

DOUBLE-ENDED QUEUES

THE DEQUE ABSTRACT DATA TYPE

front():

- Return the first element of the deque;

back():

- Return the last element of the deque;

size()

- Return the number of elements of the deque

empty():

- Return true if the deque is empty and false otherwise.

DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

insertFront(31)

DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

insertFront(10)

DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

front()

Output:
10

DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

eraseFront()

DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

insertBack(15)

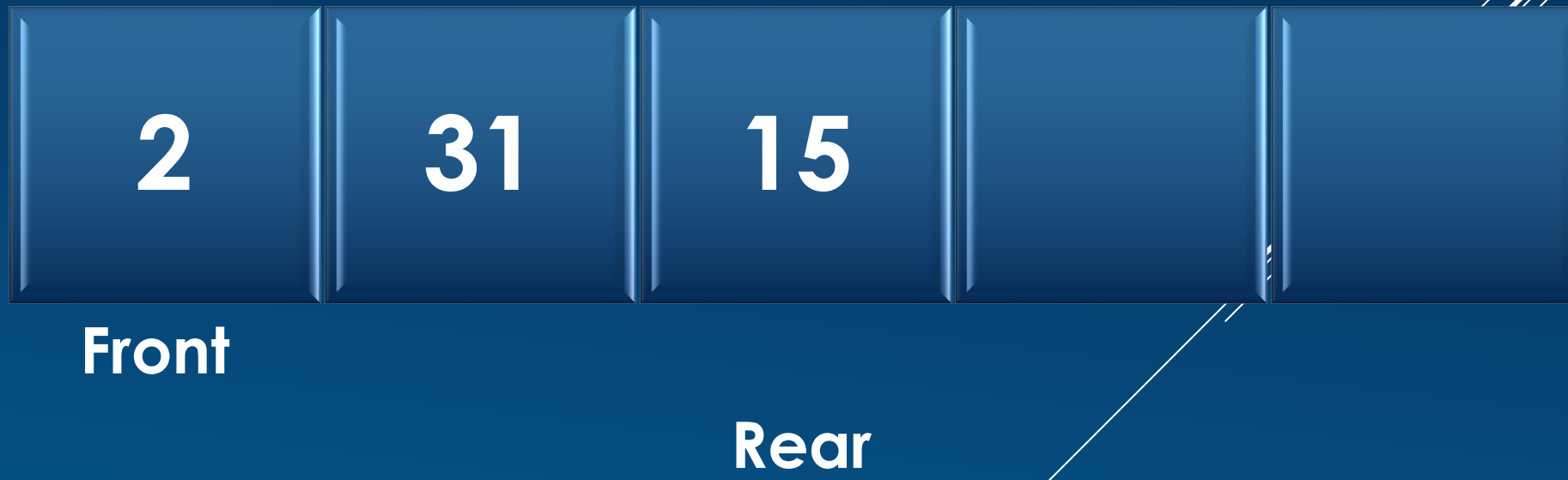
DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

insertFront(2)



DOUBLE-ENDED QUEUES

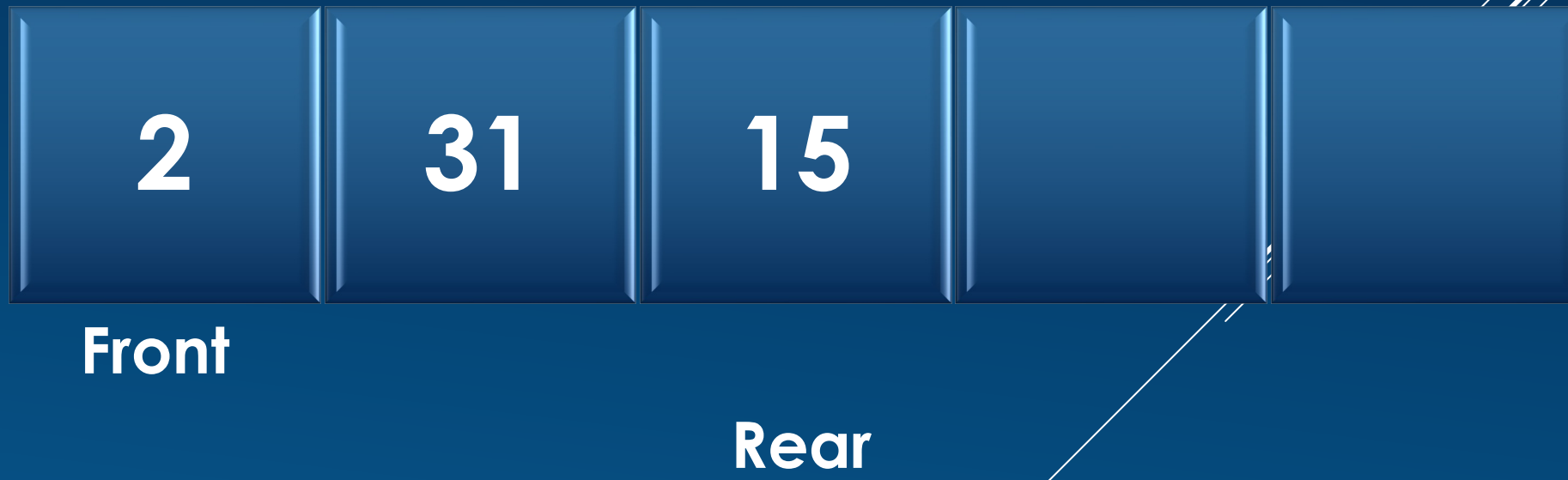
Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

back()

Output:
15



DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

eraseFront()

DOUBLE-ENDED QUEUES

Example

Operations

insertFront(31)
insertFront(10)
front()
eraseFront()
insertBack(15)
insertFront(2)
back()
eraseFront()
eraseBack()

eraseBack()

DOUBLE-ENDED QUEUES

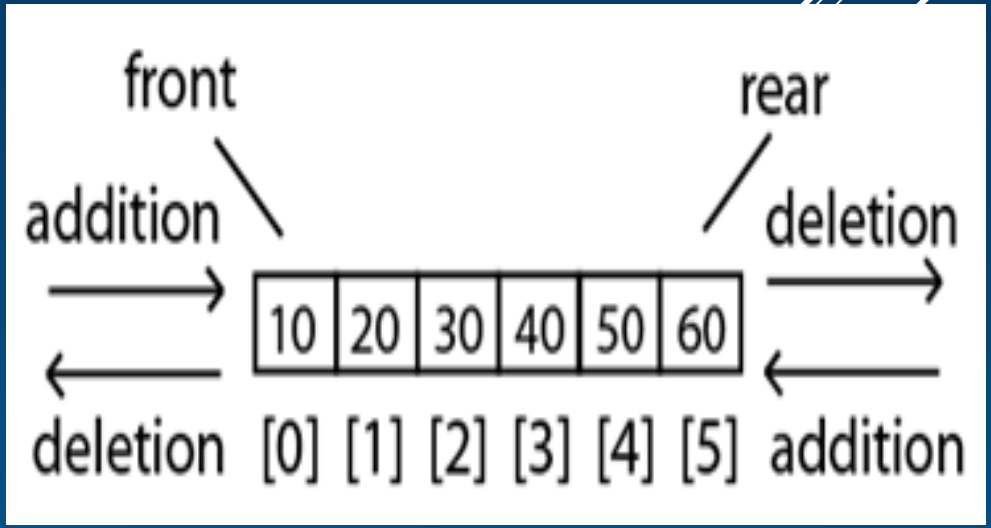
TWO VARIANTS OF DEQUE

Input restricted deque

Insertions are allowed at one end only while the deletions are allowed at both ends.

Output restricted deque

Allows insertions at both ends of the deque but permits deletions only at one end.



PRIORITY QUEUES

Priority Queue is a queue where each element is assigned a priority.

The elements are deleted and processed by following rules.

1. An element of higher priority is processed before any element of lower priority.
2. Two elements with the same priority are processed according to the order in which they were inserted to the queue.