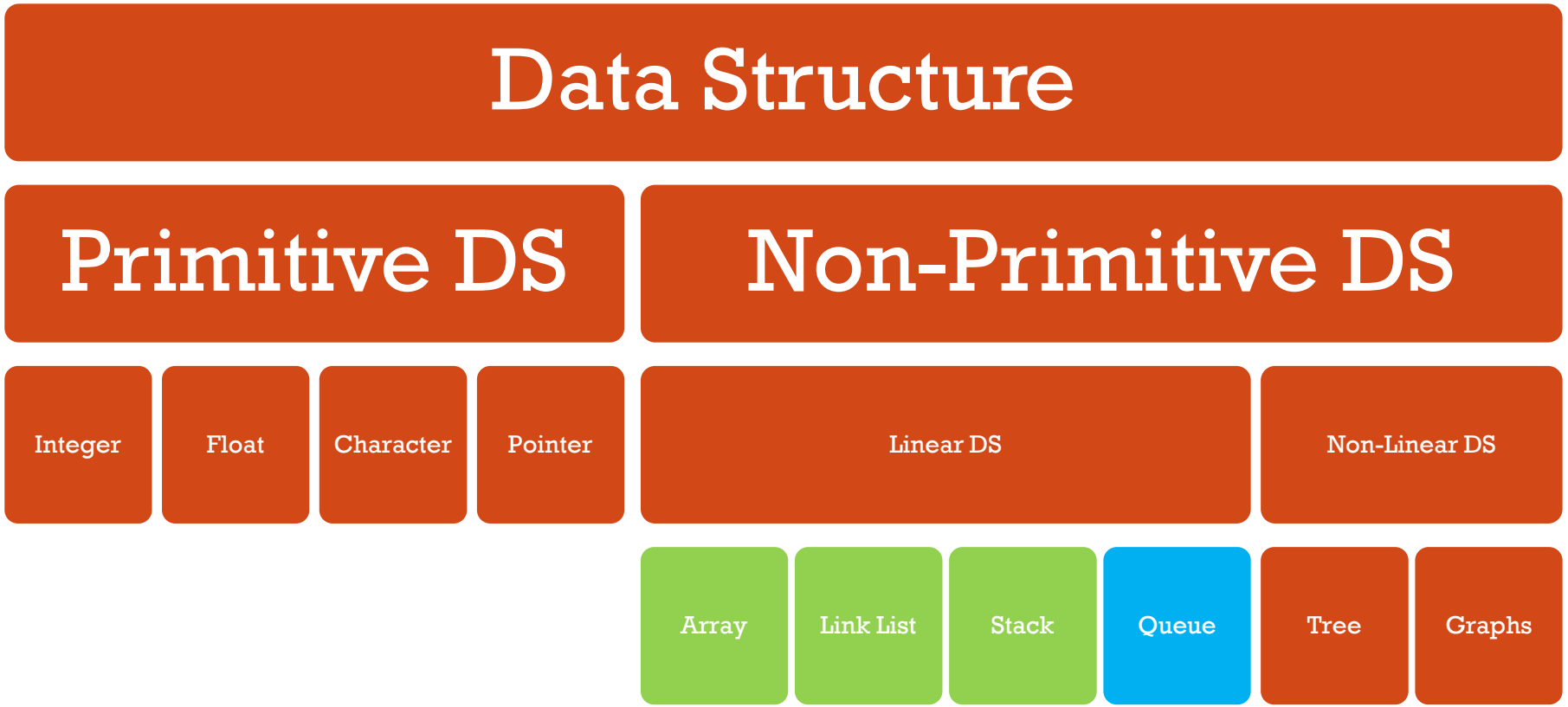
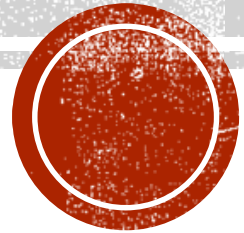
CLASSIFICATION OF DATA STRUCTURE



QUEUE



QUEUE

A list with the restriction that:

- Insertion can be performed only from one end, called the **rear** and
- Deletion can be performed at other end, called the **front**.

Stack - Last-In-First-Out
(LIFO

INTRODUCTION TO THE QUEUE ADT

- Like a stack, a queue (pronounced "cue") is a data structure that holds a sequence of elements.
- A queue, however, provides access to its elements in *first-in, first-out (FIFO)* order.
- The elements in a queue are processed like customers standing in a grocery check-out line: the first customer in line is the first one served.

QUEUE OPERATIONS

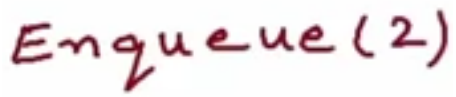
- Enqueue
  - causes a value to be stored in (pushed onto) the queue
- Dequeue
  - retrieves and removes a value from the queue

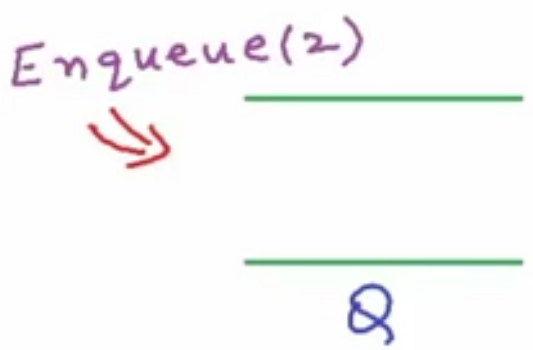
QUEUE OPERATIONS

- Enqueue(x)
- Dequeue( )
- IsEmpty( )
- IsFull( )

QUEUE OPERATIONS

- Enqueue(x)
- Dequeue( )
- IsEmpty( )
- IsFull( )



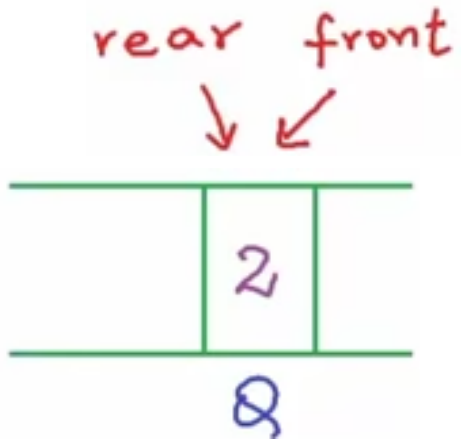


QUEUE OPERATIONS

- Enqueue(x)
- Dequeue( )
- IsEmpty( )
- IsFull( )

Enqueue(2)

Enqueue(5)



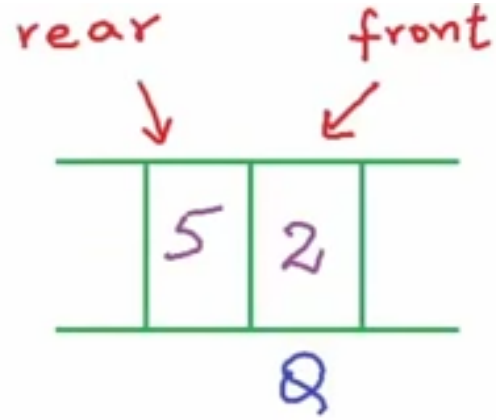
QUEUE OPERATIONS

- Enqueue(x)
- Dequeue( )
- IsEmpty( )
- IsFull( )

Enqueue(2)

Enqueue(5)

Enqueue(3)



QUEUE OPERATIONS

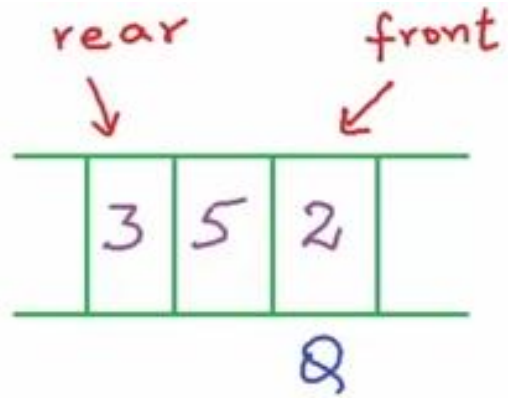
- Enqueue(x)
- Dequeue( )
- IsEmpty( )
- IsFull( )

Enqueue(2)

Enqueue(5)

Enqueue(3)

Dequeue( )



QUEUE OPERATIONS

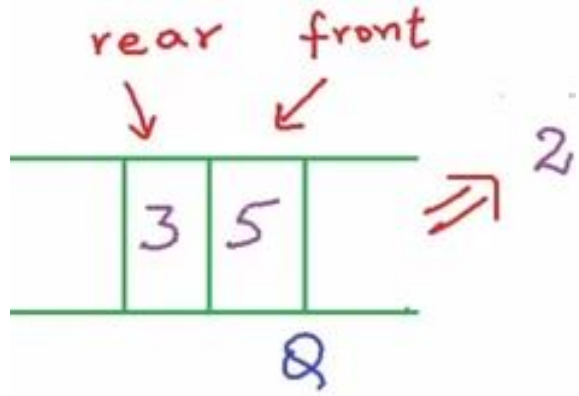
- Enqueue(x)
- Dequeue( )
- IsEmpty( )
- IsFull( )

Enqueue(2)

Enqueue(5)

Enqueue(3)

Dequeue( )



QUEUE OPERATIONS

- Enqueue(x)
- Dequeue( )
- IsEmpty( )
- IsFull( )

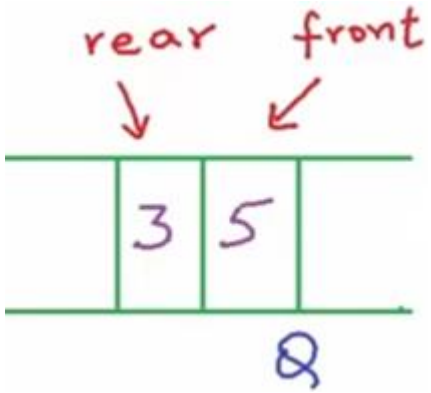
Enqueue(2)

Enqueue(5)

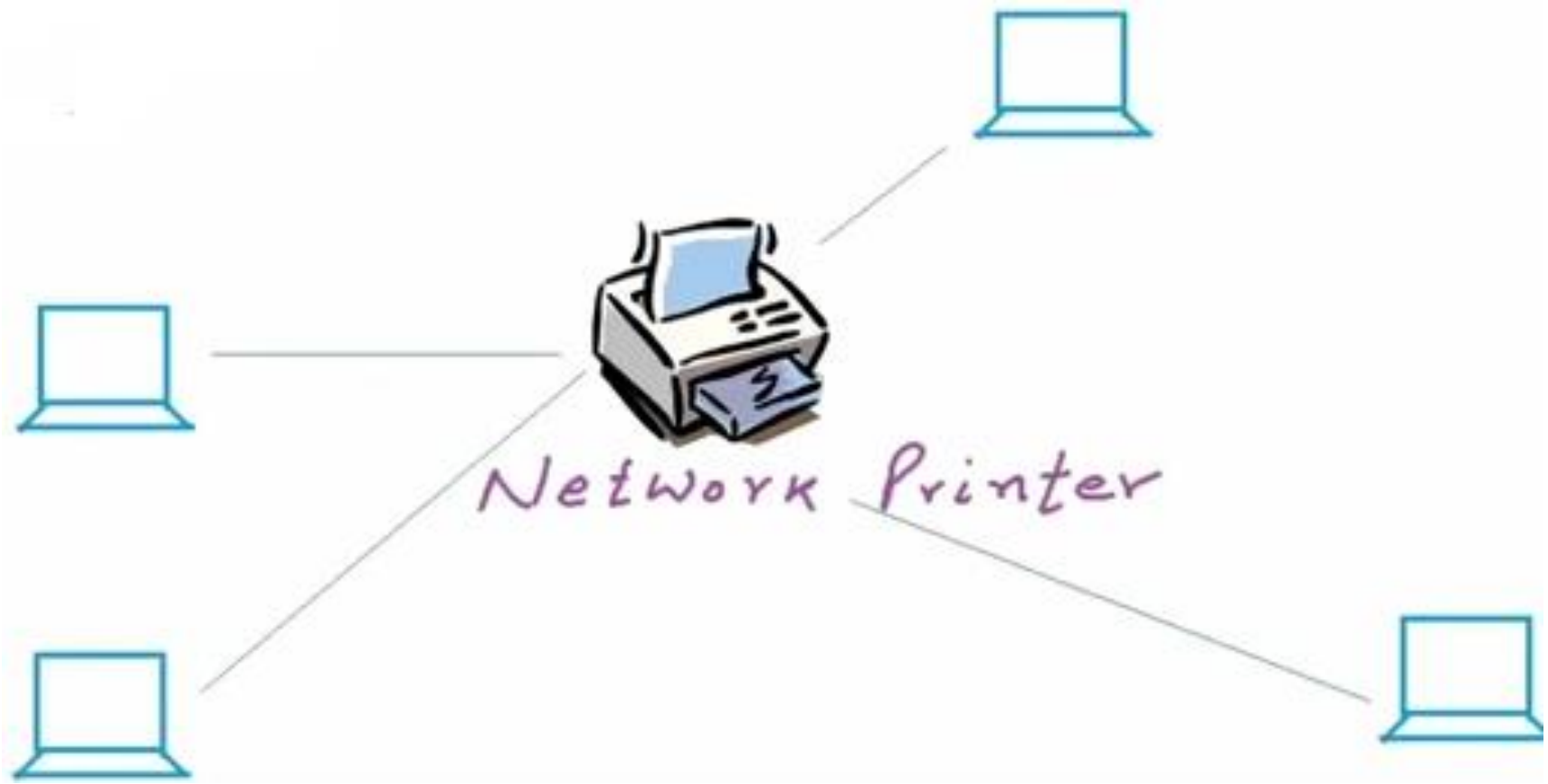
Enqueue(3)

Dequeue( ) ⇒ 2

IsEmpty() ⇒ false



QUEUE APPLICATIONS



EXAMPLE APPLICATIONS OF QUEUES

- In a multi-user system, a queue is used to hold print jobs submitted by users , while the printer services those jobs one at a time.
- Communications software also uses queues to hold information received over networks and dial-up connections. Sometimes information is transmitted to a system faster than it can be processed, so it is placed in a queue when it is received.

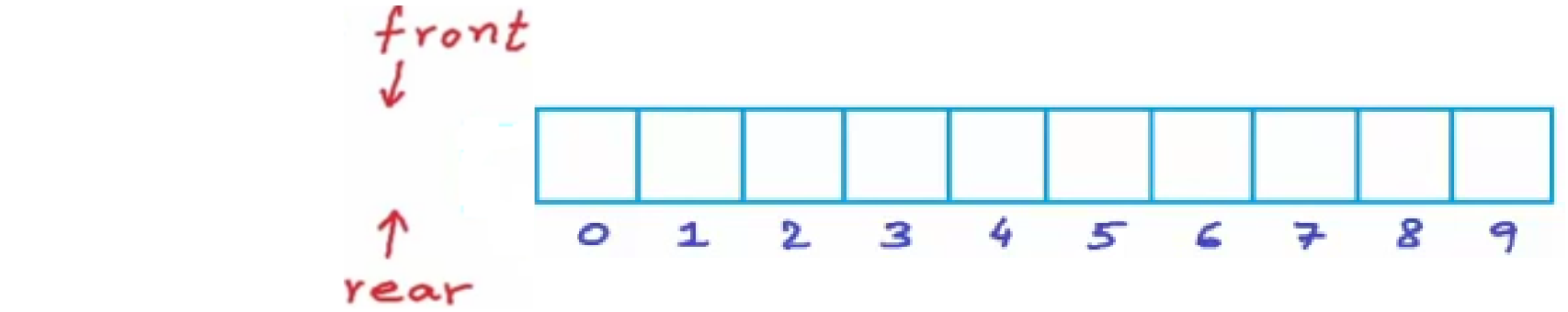
STATIC AND DYNAMIC QUEUES

- Static Queues
  - Fixed size
  - Can be implemented with an array
- Dynamic Queues
  - Grow in size as needed
  - Can be implemented with a linked list

STATIC(ARRAY) QUEUE IMPLEMENTATION

```
int A[10]

front ← -1
rear ← -1
```

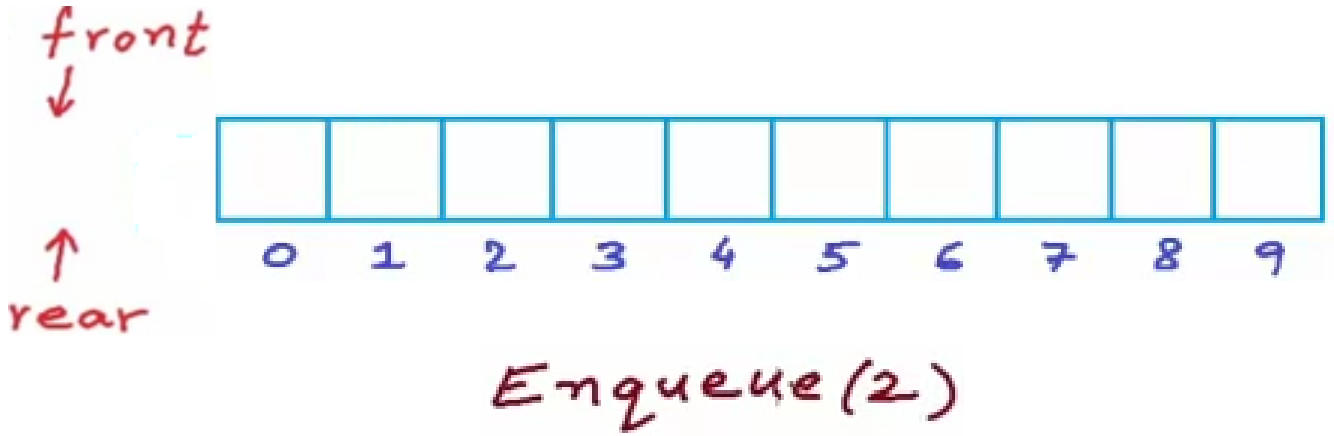



```
IsEmpty()
{ if front==-1 && rear==-1
        return true
  else
        return false
}
```

STATIC(ARRAY) QUEUE IMPLEMENTATION

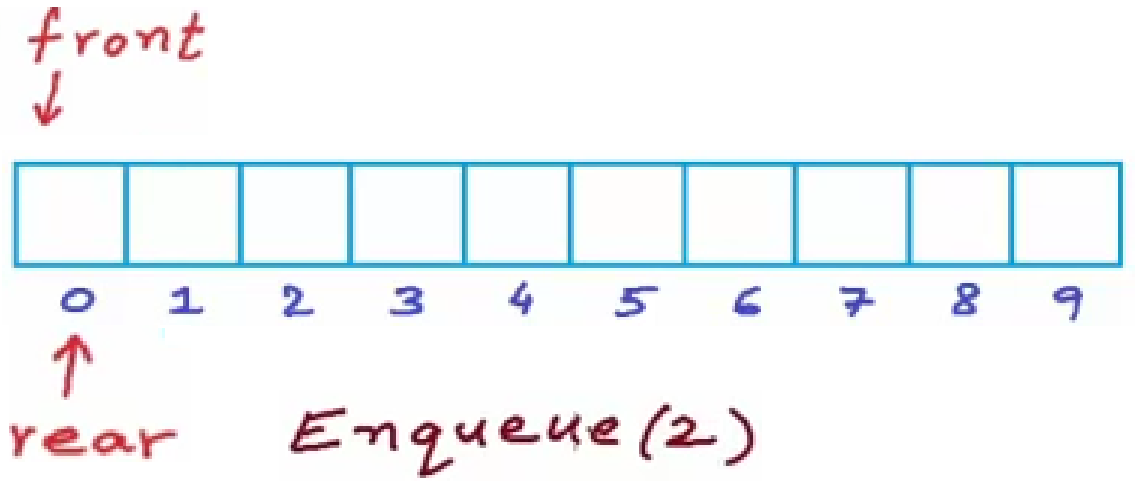
```
Enqueue(x)
{  if  IsFull()
         return
   else if IsEmpty()
   {
⇒    front ← rear ← 0
   }
   else
   {
     rear ← rear+1
   }

   A[rear] ← x
}
```



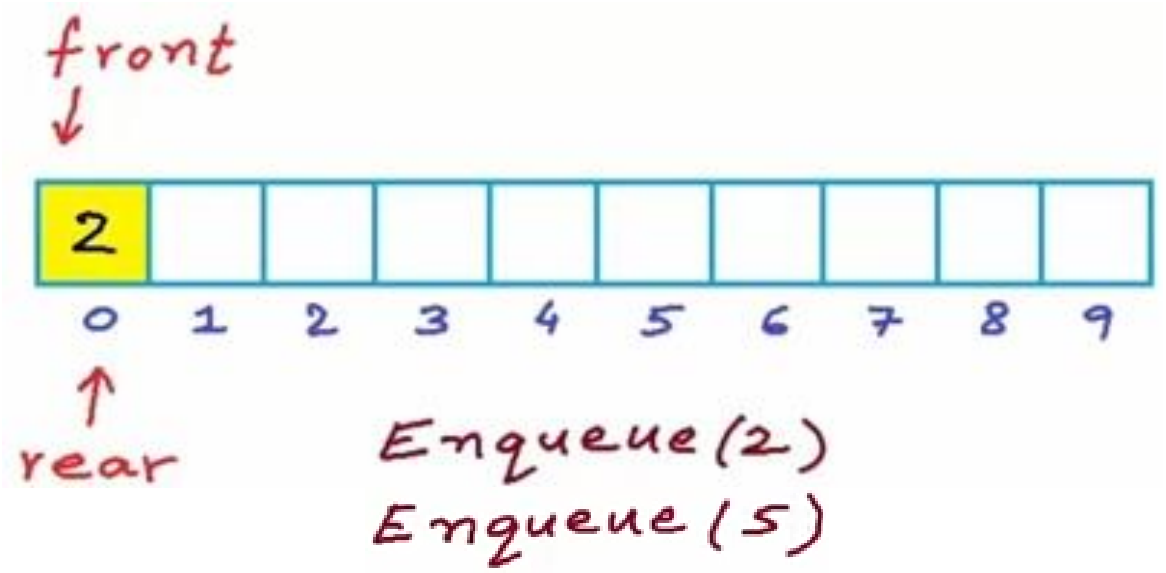
STATIC(ARRAY) QUEUE IMPLEMENTATION

```
Enqueue(x)
{  if  IsFull()
         return
   else if IsEmpty()
   {
⇒    front ← rear ← 0
   }
   else
   {
     rear ← rear+1
   }
⇒  A[rear] ← x
}
```



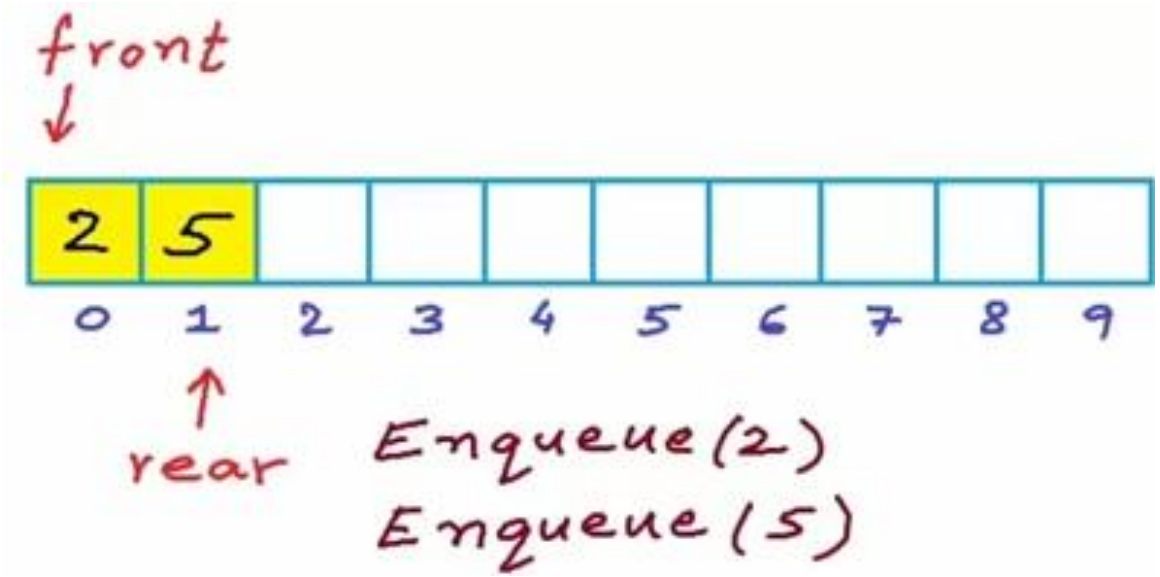
STATIC(ARRAY) QUEUE IMPLEMENTATION

```
Enqueue(x)
{  if  IsFull()
         return
   else if IsEmpty()
   {
      front ← rear ← 0
   }
   else
   {
      rear ← rear+1
   }
   A[rear] ← x
}
```



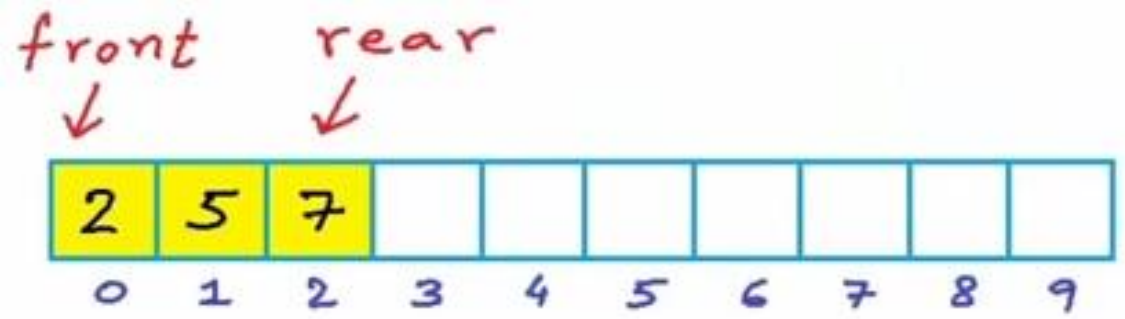
STATIC(ARRAY) QUEUE IMPLEMENTATION

```
Enqueue(x)
{  if  IsFull()
         return
   else if IsEmpty()
   {
      front ← rear ← 0
   }
   else
   {
      rear ← rear+1
   }
   A[rear] ← x
}
```



STATIC(ARRAY) QUEUE IMPLEMENTATION

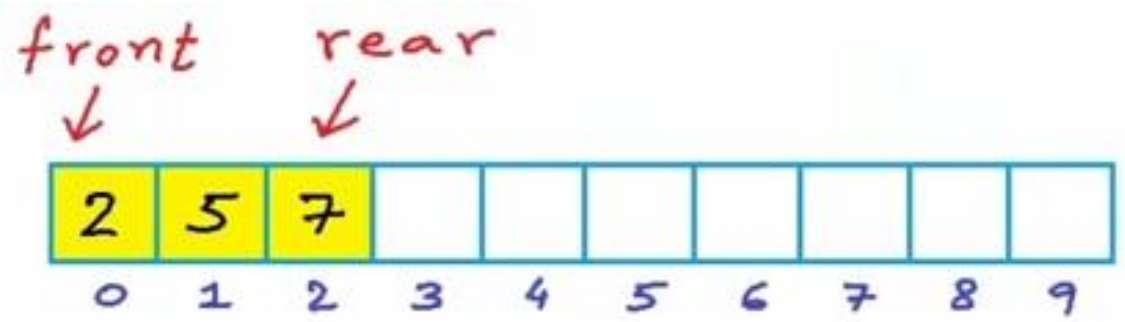
```
Enqueue(x)
{  if  IsFull()
         return
   else if IsEmpty()
   {
      front ← rear ← 0
   }
   else
   {
      rear ← rear+1
   }
   A[rear] ← x
}
```



Enqueue(2)
Enqueue(5)
Enqueue(7)

STATIC(ARRAY) QUEUE IMPLEMENTATION

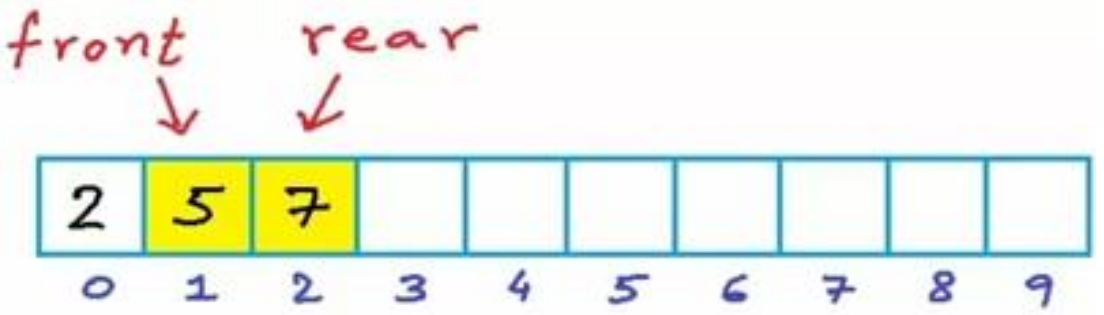
```
Dequeue()
{
  if IsEmpty()
      return
  else if front == rear
      front ← rear ← -1
  else
      front ← front +1
}
```



Enqueue(2)
Enqueue(5)
Enqueue(7)
Dequeue()

STATIC(ARRAY) QUEUE IMPLEMENTATION

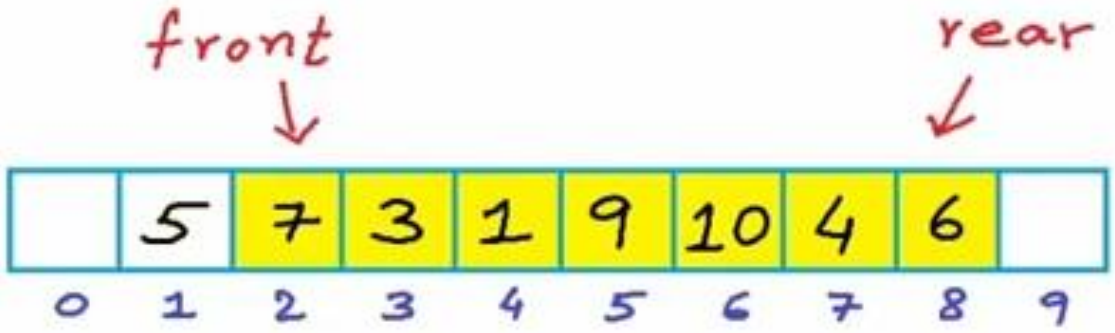
```
Dequeue()
{
  if IsEmpty()
      return
  else if front == rear
     front← rear← -1
  else
     front← front+1
}
```



Enqueue(2)
Enqueue(5)
Enqueue(7)
Dequeue()

STATIC(ARRAY) QUEUE IMPLEMENTATION

```
Dequeue()
{
  if IsEmpty()
      return
  else if front == rear
      front ← rear ← -1
  else
      front ← front +1
}
```



Enqueue(2)
Enqueue(5)
Enqueue(7)
Dequeue()
Enqueue(3)
Enqueue(1)

Enqueue(9)
Enqueue(10)
Enqueue(4)
Enqueue(6)
Dequeue()

STATIC(ARRAY) QUEUE IMPLEMENTATION

```
Dequeue()
{
  if IsEmpty()
      return
  else if front == rear
      front ← rear ← -1
  else
      front ← front +1
}
```