QUEUES

Intro

Movie Hall Ticketing

One Way Traffic

Intro

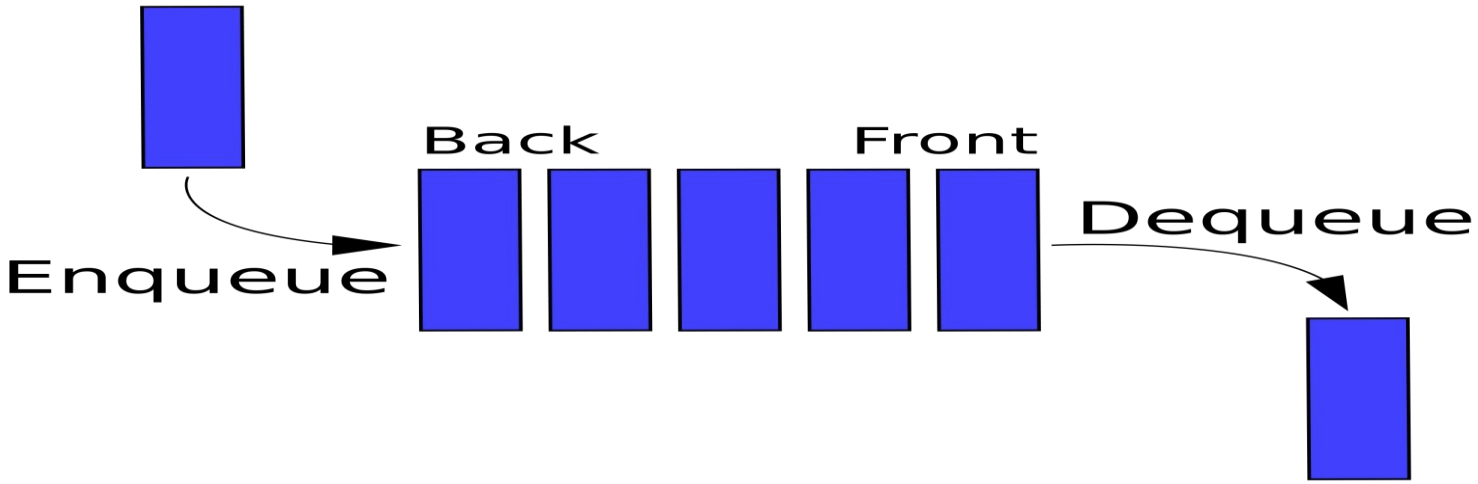
- Queue is a linear list of elements in which deletion of an element can take place at one end, called the **front** and insertion can take place at the other end, called the **rear**.
- Queue: هي قائمة خطية من العناصر التي يمكن أن يتم فيها حذف عنصر ما في أحد طرفيه ، وتسمى الواجهة الأمامية **(front)** ويمكن أن يحدث الإدراج في الطرف الآخر ، يسمى الجزء الخلفي **(rear)**.
- The first element in a queue will be the first one to be removed from the list.

سيكون العنصر الأول في قائمة الانتظار هو أول عنصر يتم إزالته من القائمة.

- Queues are also called **FIFO** (**First In First Out**) i.e. the data item stored first will be accessed first.
- تسمى قوائم الانتظار أيضًا FIFO (First In First Out)، أي سيتم الوصول إلى عنصر البيانات المخزن أولاً .. الوصول اليها أولاً

Queue Representation

- In queue, we access both ends for different reasons



Applications of queue تطبيقات الطابور

- Serving requests on a single shared resource, like a printer, CPU task scheduling etc.

- تقديم الطلبات على مورد واحد مشترك ، مثل الطابعة ، وجدولة مهام وحدة المعالجة المركزية ، وما إلى ذلك.

- In real life, **Call Center phone** systems will use Queues, to hold people calling them in an order, until a service representative is free.

- في الحياة الواقعية ، ستستخدم أنظمة الهاتف في مركز الاتصال قوائم الانتظار ، لإبقاء الأشخاص الذين يتصلون بهم بالترتيب ، حتى يصبح ممثل الخدمة مجانيًا.

- Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive, First come first served.

- معالجة المقاطعات في أنظمة الوقت الفعلي. يتم التعامل مع المقاطعات بنفس ترتيب وصولها ، من يأتي أولاً يخدم أولاً.

The queue as an ADT

- A queue **q** of type **T** is a finite sequence of elements with the operations
- الطابور q من النوع T هو تسلسل محدود من العناصر مع العمليات
  - ***MakeEmpty(q):*** To make **q** as an empty queue
    - Make Empty (q) : لجعل q كقائمة انتظار فارغة
  - ***IsEmpty(q):*** To check whether the queue **q** is empty. Return true if **q** is empty, return false otherwise.
    - Is Empty (q) : للتحقق مما إذا كانت قائمة الانتظار q فارغة. إرجاع صحيح إذا كانت q فارغة ، وإرجاع خطأ خلاف ذلك.
  - ***IsFull(q):*** To check whether the queue **q** is full. Return true in **q** is full, return false otherwise.
    - Is Full (q) : للتحقق مما إذا كانت قائمة الانتظار q ممتلئة. إرجاع صحيح في q ممتلئ ، وإرجاع خطأ بخلاف ذلك.
  - ***Enqueue(q, x):*** To insert an item **x** at the rear of the queue, if and only if **q** is not full.

    Enqueue (q , x): لإدراج عنصر x في الجزء الخلفي من قائمة الانتظار ، إذا وفقط إذاف ليست ممتلئة.

- ***Dequeue(q):*** To delete an item from the front of the queue **q** if and only if **q** is not empty.

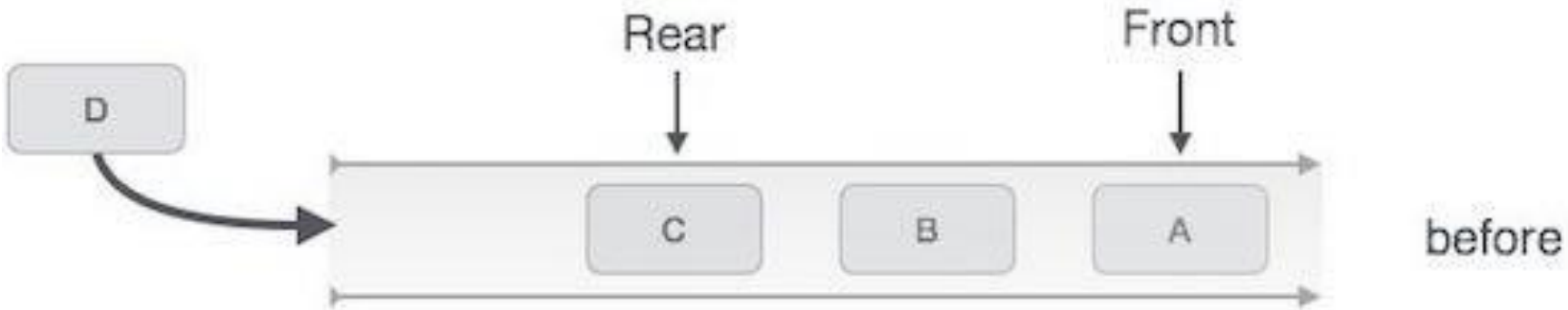
Dequeue (q): لحذف عنصر من مقدمة قائمة الانتظار q إذا وفقط إذا q ليس فارغًا.

- ***Traverse (q):*** To read entire queue that is display the content of the queue.

- ***Traverse (q)*** q): اجتياز (لقراءة قائمة الانتظار بأكملها التي تعرض محتوى قائمة الانتظار.

Enqueue Operation

- Queue maintains two data pointers, **front** and **rear**
- The following steps should be taken to **enqueue**(insert) data into queue –
  - Step 1 – Check if queue is full
  - Step 2 – if queue is full
    produce overflow error and exit
    else
    increment rear pointer to point next empty space
    and add data element to the queue location, where rear is pointing
  - Step 3 - return success



Queue Enqueue

Implementation of enqueue()

```
int enqueue(int data) {
        if(isfull())
                return 0;
        rear = rear + 1;
        queue[rear] = data;
        return 1;
}
```

Dequeue Operation

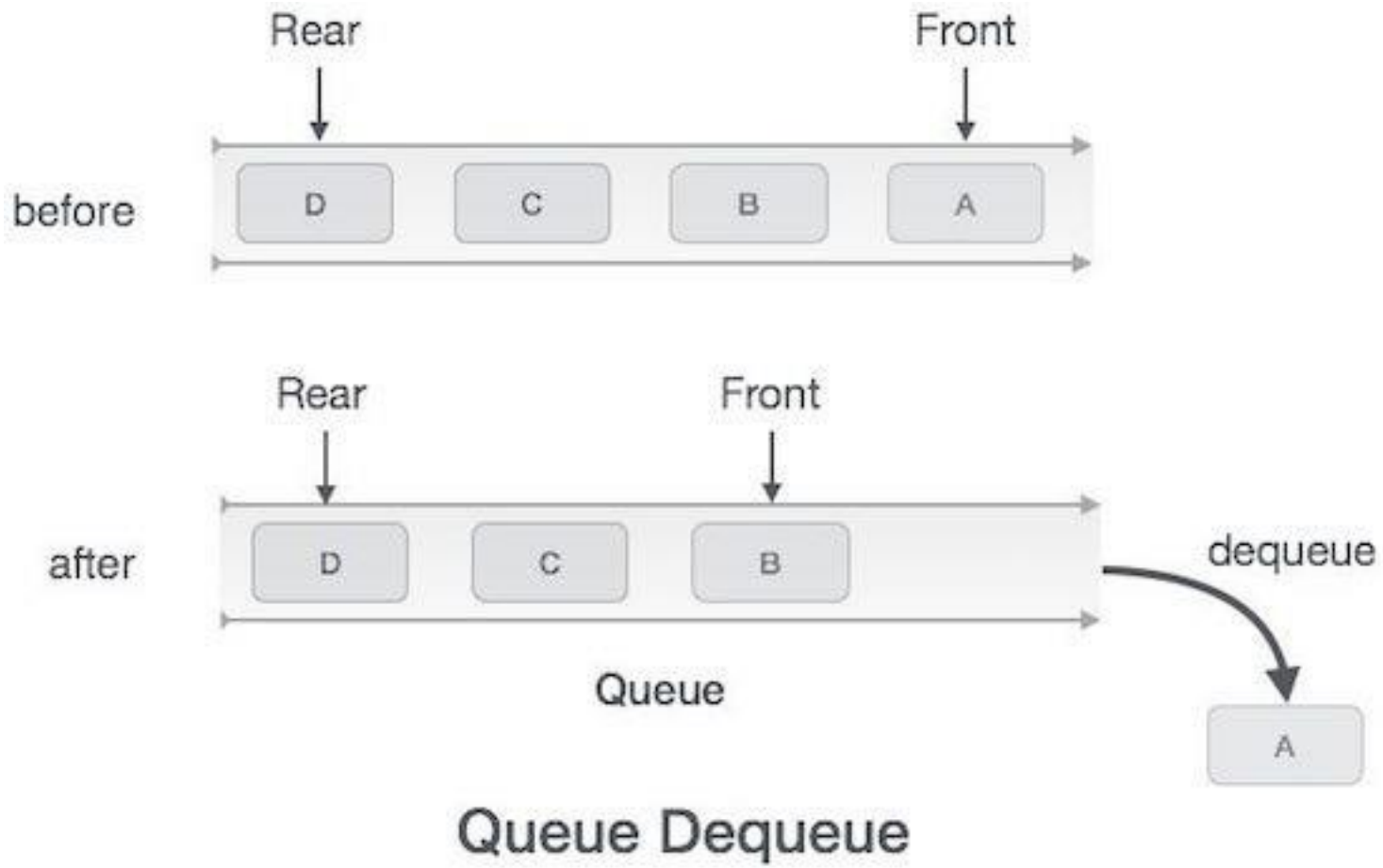
- Accessing data from queue is a process of two steps
  - Access the data from where **front** is pointing
  - And remove the data after access

- The following steps are taken to perform **dequeue** operation
  - Step 1 – Check if queue is empty
  - Step 2 – if queue is empty

    produce underflow error and exit

    else

    access data where front is pointing, increment front pointer to point next available data element
  - Step 3 – return success.



Queue Dequeue

Implementation of dequeue()

```
int dequeue() {
        if(isempty()) {
                return 0;
        }
        int data = queue[front];
        front = front + 1; return
        data;
}
```

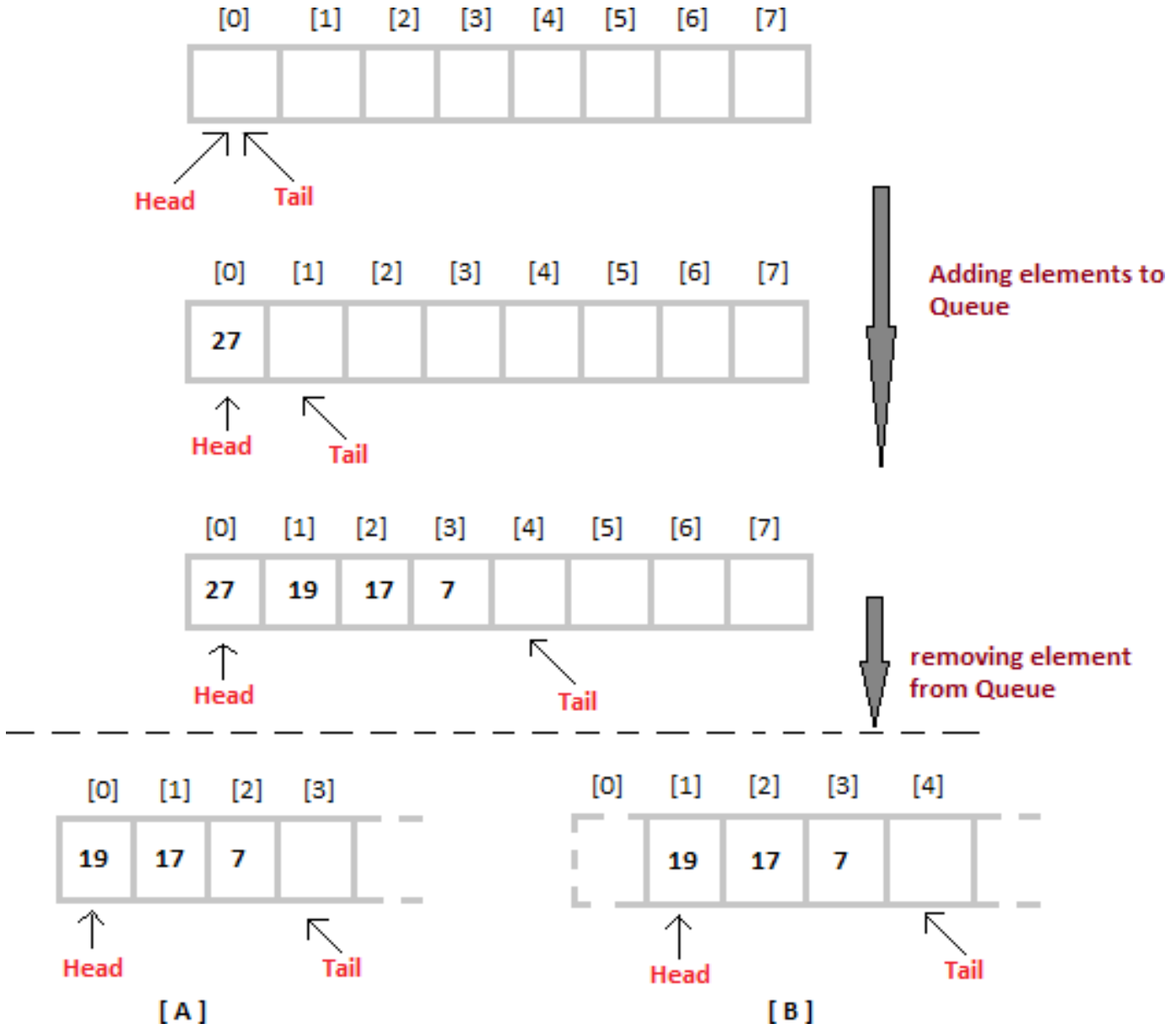
Implementation of queue

- There are two techniques for implementing the queue:
- هناك طريقتان لتنفيذ قائمة الانتظار:
  - **Array implementation of queue (static memory allocation)**
    - تنفيذ صفيف من قائمة الانتظار (تخصيص الذاكرة الثابتة)
    - ***Linear array implementation (Linear Queue)***
      - تنفيذ المصفوفة الخطية (قائمة الانتظار الخطية)
    - ***Circular arrayImplementation (Circular queue)***
      - تنفيذ مصفوفة دائرية (قائمة انتظار دائرية)
  - **Linked list implementation of queue (dynamic memory allocation)**
    - تنفيذ القائمة المرتبطة لقائمة الانتظار (تخصيص الذاكرة الديناميكي)

Array implementation of queue

- The easies way of implementing a queue is by using an Array.
  - الطريقة الأسهل لتطبيق قائمة انتظار هي باستخدام صفيف.
- Initially head(FRONT) and the tail(REAR) of the queue points at the first index of the array.
  - في البداية رأس (أمامي) وذيل (خلفي) لنقاط الانتظار عند الفهرس الأول للصفيف.
- As we add elements to the queue, we can either move tail before adding another item or we can move the tail after adding the item, while the head remains at the first index.
  - عندما نضيف عناصر إلى قائمة الانتظار ، يمكننا إما تحريك الذيل قبل إضافة عنصر آخر أو يمكننا تحريك الذيل بعد إضافة العنصر ، بينما يظل الرأس في الفهرس الأول.

Linear Queue

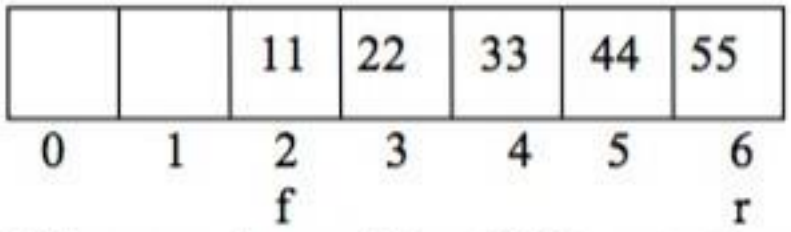


Linear Queue

Insertion of an item in queue	Deletion of an item from queue
1. *Initialize front=0 and rear=-1* *if rear>=MAXSIZE-1* *print "queue overflow" and return* *else* *set rear=rear+1* *queue[rear]=item* *2. end*	*1. if rear<front* *print "queue is empty" and return* *else* *item=queue[front++]* *2. end*

Problems with Linear queue implementation

- Both rear and front indices are **increased** **but never decreased.**
  - زاد كل من المؤشرين الخلفي والأمامي ولكنهما لم ينخفضا أبدًا.
- As items are removed from the queue, the storage space at the beginning of the array is discarded and never used again.
  - عند إزالة العناصر من قائمة الانتظار ، يتم تجاهل مساحة التخزين في بداية المصفوفة وعدم استخدامها مرة أخرى.
- Wastage of the space is the main problem with linear queue which is illustrated by the following example.
  - الهدر في الفضاء هو المشكلة الرئيسية في الطابور الخطي الذي يوضحه المثال التالي.



This queue is considered full, even though the space at beginning is vacant.

Circular queue

- A queue where the start and end of the queue are joined together.
  قائمة انتظار حيث يتم ربط بداية قائمة الانتظار ونهايتها معًا.
- A circular queue is one in which the insertion of a new element is done at very first location of the queue if the last location of the queue is full.
  - قائمة الانتظار الدائرية هي تلك التي يتم فيها إدخال عنصر جديد في أول موقع لقائمة الانتظار إذا كان آخر موقع لقائمة الانتظار ممتلئًا.

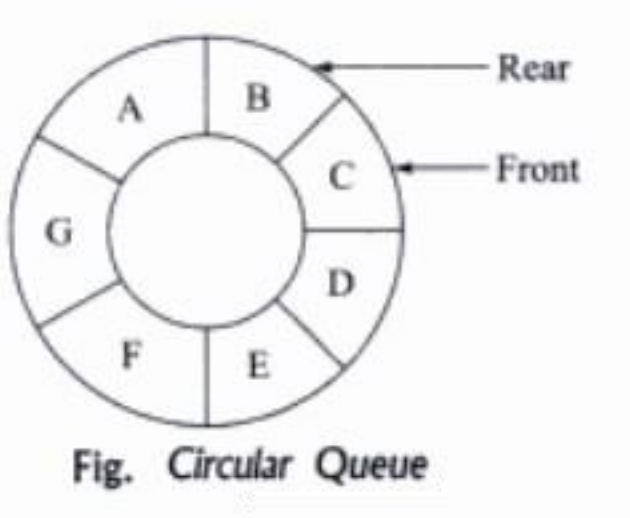
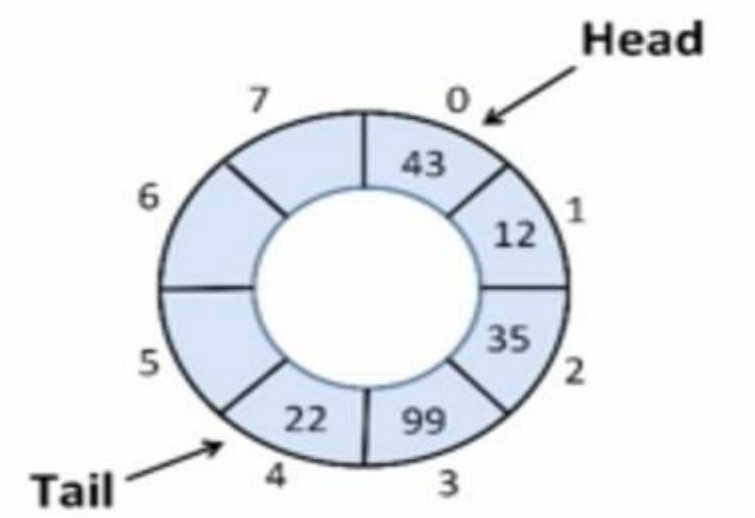
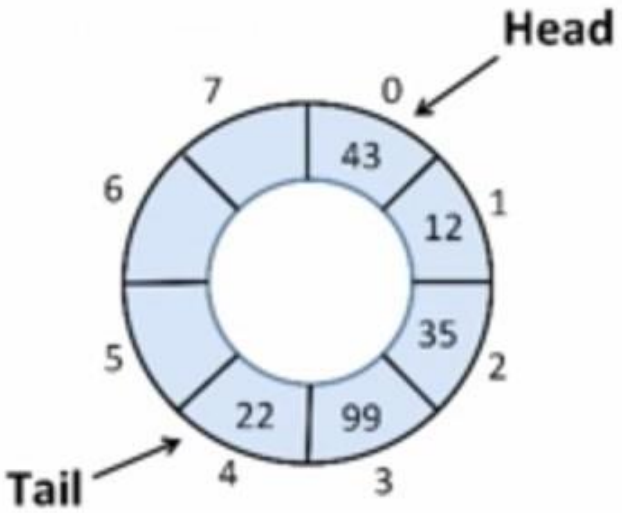
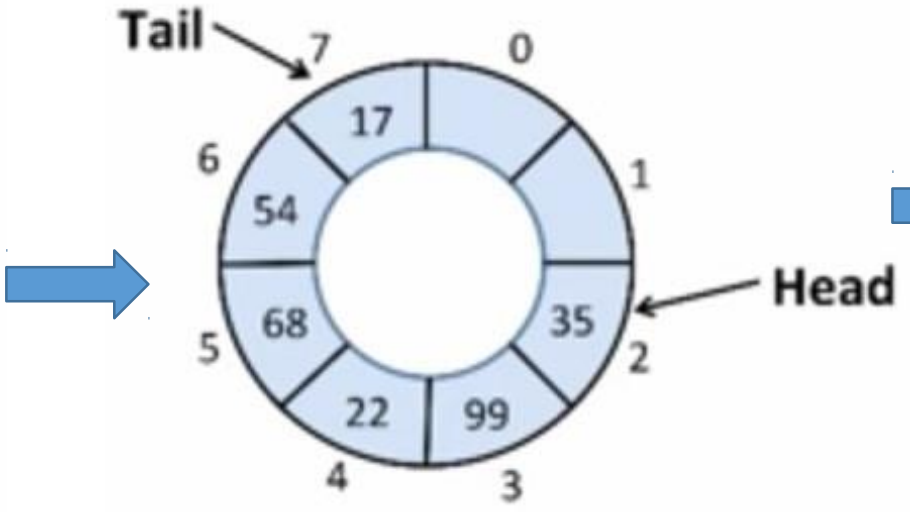
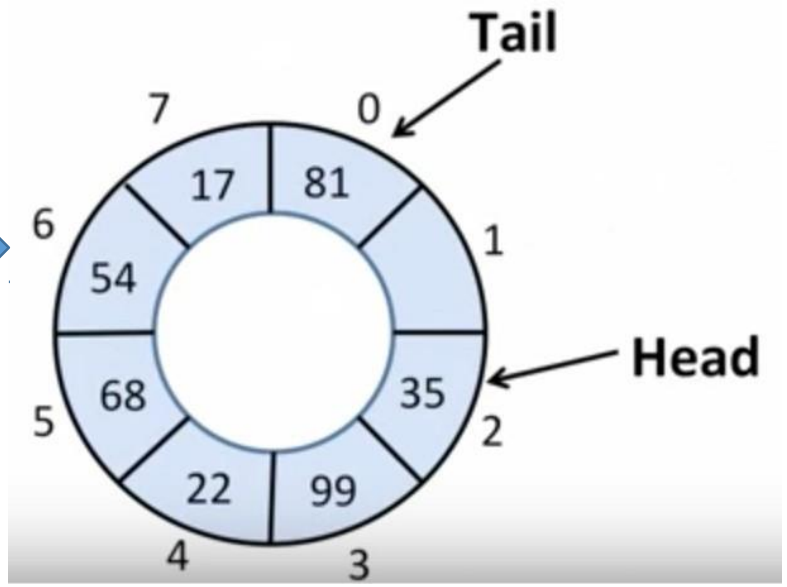


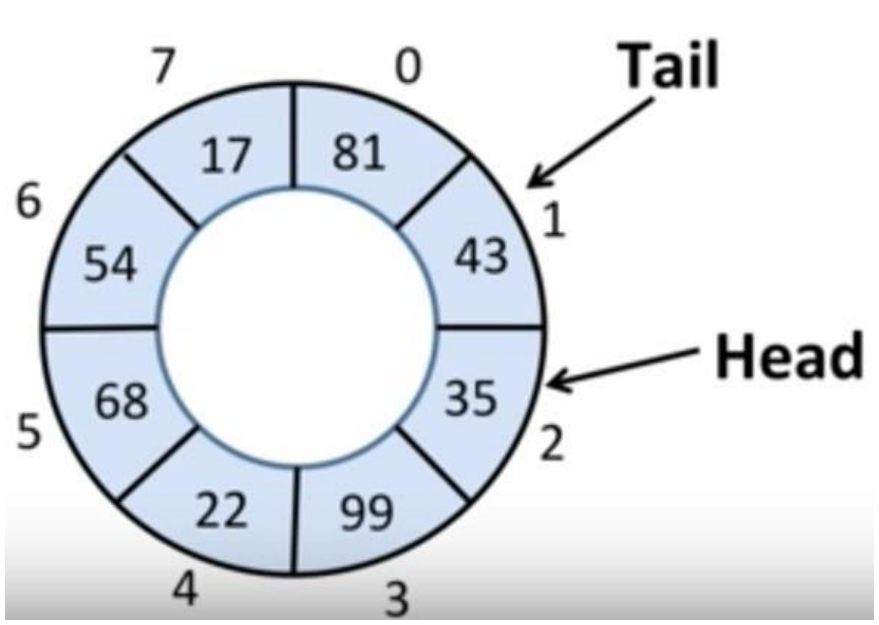
Fig. Circular Queue




Head
Tail
0
1
2
3
4
5
6
7
43
12
35
22
99


Tail
Head
0
1
2
3
4
5
6
7
17
54
68
22
99
35


Tail
Head
0
1
2
3
4
5
6
7
17
81
54
68
22
99
35


Tail
Head
0
1
2
3
4
5
6
7
17
81
54
68
22
99
43
35

Circular queue

isFull()	IsEmpty
- If HEAD == (Tail % MAX) + 1 Then **Full** <- True; Else **Full** <- False; - If they have caught up to each other, then it's full	- If Head ==Tail Then **Full** <- True; Else **Full** <- False;

Circular queue

Enqueue operation	Dequeue operation

Step 1. start
Step 2. if (front == (rear+1)%max)
Print error “circular queue overflow “
Step 3. else{
rear = (rear+1)%max
Q[rear] = element;
}
Step 4. stop
	Step 1. start
	Step 2. if isEmpty() == True
	Print error “Queue is Empty“
	Step 3. else{
	element = Q[front]
	front = (front + 1) % max
	}
	Step 4. stop