هياكل البيانات

المرحلة الثانية

محاضرة(5)

م.م فرح معاذ جاسم

Farahmaath86@uoanbar.edu.iq

1. QUEUE DATA STURCTURE

A queue is an ordered collection of items from which items may be deleted at one end (collect the front of the queue) and into which items may be inserted at the other end (called the nearer of the queue). Unlike stacks, a queue is open at both its ends.

The first element inserted into a queue is the first element to be removed. For this reason a queue is sometimes called a FIFO (first – in first – out) list.

Queue

The figure illustrates a queue containing there elements A, B and C. A is at the front of the queue and C is at the rear.

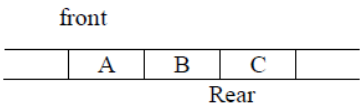


Figure -1-

In figure -2- an element has been deleted from the queue. Since may be deleted only from the front of the queue, A is removed and B is now at the front.

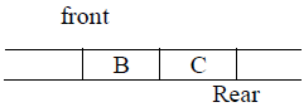


Figure -2-

In figure -3- , when items D and E are inserted, they may be inserted at the rear of the queue.

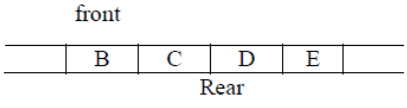


Figure -3-

Initially, the queue is empty. Then A, B, and C have been inserted, then two items have been deleted into two new items D and E have been inserted.

Although there are two spaces in the queue array but, we cannot insert a new element to this queue, because of the overflow case is notified when

rear = = maxqueue -1

When the queue is empty. Since rear is the index of last element of the queue, the condition

front = = rear=-1.

1.2BASIC OPERATIONS

Three primitive operations can be applied to a queue:.

1. The operation insert (q,x),inserts item x at the rear of the queue q. the operation

2.X = remove (a) deletes the front element from the queue q and sets x to its contents.

3.The third operation, empty (a), returns false or true depending on whether or not the queue contains any element.

```
begin procedure isfull

   if rear equals to MAXSIZE
      return true
   else
      return false
   endif

end procedure
```

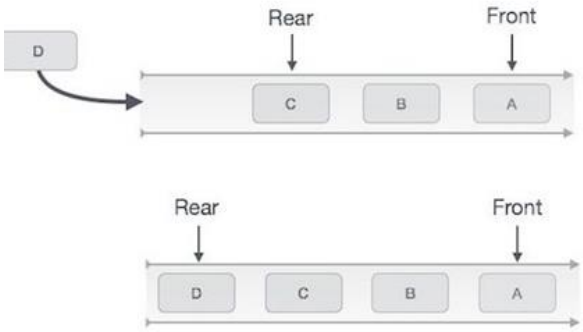
```
begin procedure isempty

   if front is less than MIN  OR front is greater than rear
      return true
   else
      return false
   endif

end procedure
```

1.3 INSERT IN QUEUE

Queues maintain two data pointers, **front** and **rear**. Therefore, its operations are comparatively difficult to implement than that of stacks.

The following steps should be taken to insert data into a queue −

- **Step 1** − Check if the queue is full.
- **Step 2** − If the queue is full, produce overflow error and exit.
- **Step 3** − If the queue is not full, increment **rear** pointer to point the next empty space.
- **Step 4** − Add data element to the queue location, where the rear is pointing.
- **Step 5** − return success.




```
procedure insert(data)

   if queue is full
      return overflow
   endif

   rear ← rear + 1
   queue[rear] ← data
   return true

end procedure
```

1.4 REMOVE FROM QUEUE

Accessing data from the queue is a process of two tasks − access the data where **front** is pointing and remove the data after access. The following steps are taken to perform **remove** operation −

- **Step 1** − Check if the queue is empty.
- **Step 2** − If the queue is empty, produce underflow error and exit.
- **Step 3** − If the queue is not empty, access the data where **front** is pointing.
- **Step 4** − Increment **front** pointer to point to the next available data element.
- **Step 5** − Return success.

```
procedure dequeue

   if queue is empty
      return underflow
   end if

   data = queue[front]
   front ← front + 1
   return true

end procedure
```