Queue

Data Structure

www.eshikshak.co.in

Introduction

- It is linear data structure
- It is collection of items – List
- Queue means line of items waiting for their turn
- Queue has two ends
  - Elements are added at one end.
  - Elements are removed from the other end.

Queue

- Removal of data item is restricted at one end known as FRONT
- Insertion of data item is restricted at other end known as REAR
- The FRONT and REAR are used in describing a linear list, when queue is implemented
- First element inserted in list, will be the first to be removed - FIFO

Queue as FIFO

- The element inserted first will be removed first from the Queue
- Thus, Queue is known as FIFO (**F**irst **I**n-**F**irst **O**ut) or FCFS (**F**irst **C**ome **F**irst **S**erve)

- **Examples of Queue**
  - People waiting in Queue to purchase tickets at railway station or cinema hall, where the first person in the queue will be served first

Representation of Queue

- It has two pointer varaibles
  - FRONT : Containing the location of the front element of the queue
  - REAR : Containing the location of the rear element of the queue
- When queue is empty

  FRONT = -1 and REAR = -1

Operations of Queue

- Insertion
  - Adding an element in queue will increased value of REAR by 1
    - REAR = REAR + 1
- Removal
  - Removing an element from queue will increased value of FRONT by 1
    - FRONT = FRONT + 1

Queue Empty	FRONT = -1, REAR = -1	
ADD(Q,10)	FRONT = 0 ,REAR = 0	10
ADD(Q,20)	FRONT = 0 ,REAR = 1	10 20
ADD(Q,30)	FRONT = 0 ,REAR = 2	10 20 30
ADD(Q,40)	FRONT = 0 ,REAR = 3	10 20 30 40
ADD(Q,50)	FRONT = 0 ,REAR = 4	10 20 30 40 50
Remove(Q)	FRONT = 1 ,REAR = 4	20 30 40 50
Remove(Q)	FRONT = 2 ,REAR = 4	30 40 50
Remove(Q)	FRONT = 3 ,REAR = 4	40 50
ADD(Q, 60)	FRONT = 3 ,REAR = 1	60 40 50

Types of Queue

- Queue as Array
- Circular Queue
- Priority Queue
- Input Restricted Queue
- Output Restricted Queue
- Dqueue

Queue as Array : Insert

Initially when the QUEUE is empty, set FRONT = NULL and REAR = 0

Step 1: start
Step2: [check for queue is over flow or not]
If (REAR >n) or (REAR==FRONT)
Print “queue is overflow”
else
go to step 3
Step 3: [enter the item]
QUEUE[REAR]=value
REAR=REAR+1
Step 4:[check condition]
If(FRONT==null)
FRONT=0
Step 5:end

Queue as Array : Delete

Step 1: start
Step 2: [check for queue is under flow or not]
If front>N or front==Null
Print ”queue is underflow”
else
goto step 3
Step 3: [check condition]
If front==rear
Front==null
Rear=0
else
goto step 4
Step 4: [delete element]
Queue[front]=null
Step 5: front=front+1
Step 6: end

Circular Queue

- To solve this problem, queues implement wrapping around. Such queues are called Circular Queues.
- Both the front and the rear pointers wrap around to the beginning of the array.
- It is also called as “Ring buffer”.
- Items can inserted and deleted from a queue in O(1) time.

Circular Queue

- When a new item is inserted at the rear, the pointer to rear moves upwards.
- Similarly, when an item is deleted from the queue the front arrow moves downwards.
- After a few insert and delete operations the rear might reach the end of the queue and no more items can be inserted although the items from the front of the queue have been deleted and there is space in the queue.

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

1 [Queue already filled ?]
if FRONT = 0 and REAR = N-1, or if FRONT=REAR + 1 then
write : Overflow and Return
2 [Find new value of REAR]
if FRONT=-1 then [Queue initially empty]
Set FRON T = 0 and REAR = 0
else if REAR = N-1 then
Set REAR = 1
else
Set REAR = REAR + 1
[End of If structure]
3 Set QUEUE[REAR] = ITEM [This inserts new element]
4 Return

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

1 [Queue already empty]
If FRONT = -1, then Write : Underflow and Return
2 Set ITEM = Queue[FRONT]
3 [Find new value of FRONT]
If FRONT = REAR, then [Queue has only one element to start]
Set FRONT = -1 and REAR = -1
Else if FRONT = N-1, then
Set FRONT = 0
Else
Set FRONT = FRONT + 1
[End of If Structure]
4 Return

Priority Queue

Suppose that you have a few assignments from different courses. Which assignment will you want to work on first?

Course	Priority	Due day
Database Systems	2	October 3
CONMS	4	October 10
Advance 'C' & Data Structure	1	September 29
Software Engineering	3	October 7
C++	5	October 15

You set your priority based on due days. Due days are calle
keys.

Priority Queue

- It is collection of elements where elements are stored according to the their priority levels
- Inserting and removing of elements from queue is decided by the priority of the elements
- The two fundamental methods of a priority queue *P*:
  - insertItem(k,e): Insert an element *e* with key *k* into *P*.
  - removeMin(): Return and remove from *P* an element with the smallest key.

Priority Queue

- When you want to determine the priority for your assignments, you need a value for each assignment, that you can compare with each other.

- **key**: An object that is assigned to an element as a specific attribute for that element, which can be used to identify, rank, or weight that element.

Example: Student records

Any of t Score can be u

Note: K

Student Name	ID	Final Score (out of 450)
Ashwin	09BCA08	310
Payal	09BCA80	311
Darshika	09BCA24	380
Nilkamal	09BCA75	400
Nikunj	09BCA74	440
Mori	09BCA102	400

Rules to maintain a Priority Queue

- The elements with the higher priority will be processed before any element of lower priority
- If there are elements with the same priority, then the element added first in the queue would get processed

Priority Queue Insert

PQInsert (M, Item)
Step 1 Find the Row M
Step2 [Reset the Rear Pointer]
If Rear[M] = N-1 then Rear[M] = 0
Else
Rear[M] = Rear[M]+1
Step 3 [Overflow]
If Front[M] = Rear[M] then Write (“This Priority Queue is full”)
Return
Step 4 [Insert Element]
Q[M] [Rear[M]] = then
Step 5 [Is Front Pointer Properly Set]
If Front[M] = -1 then Front[m] = 0
Return
Step 6 Exit

Priority Queue Delete

PQDelete (K, Item)
Step 1 Initialize K = 0
Step 2 while (Front[K] = -1)
K = K+1
[To find the first non empty queue]
Step 3 [Delete Element]
Item = Q[K] [Front[K]
Step 4 [Queue Empty]
If Front[K] = N-1 then Front[K] = 0
Else
Front[K] = Front[K]+1
Return Item
Step 6 Exit

Deque

- Deque stands for double-end queue
- A data structure in which elements can be added or deleted at either the front or rear
- But no changes can be made in the list
- Deque is generalization of both stack and queue

Removing Element from Deque

- There are two variations of a deque. These are
  - Input Restricted Deque
    - An input restricted deque restricts the insertion of elements at one end only, but the deletion of elements can perform at both the ends.
  - Output Restricted Deque
    - An output restricted queue, restricts the deletion of elements at one end only, and allows insertion to be done at both the ends of deque

Possibilities

- The two possibilities that must be considered while inserting or deleting elements into the queue are :
  - When an attempt is made to insert an element into a deque which is already full, an overflow occurs.
  - When an attempt is made to delete an element from a deque which is empty, underflow occurs.

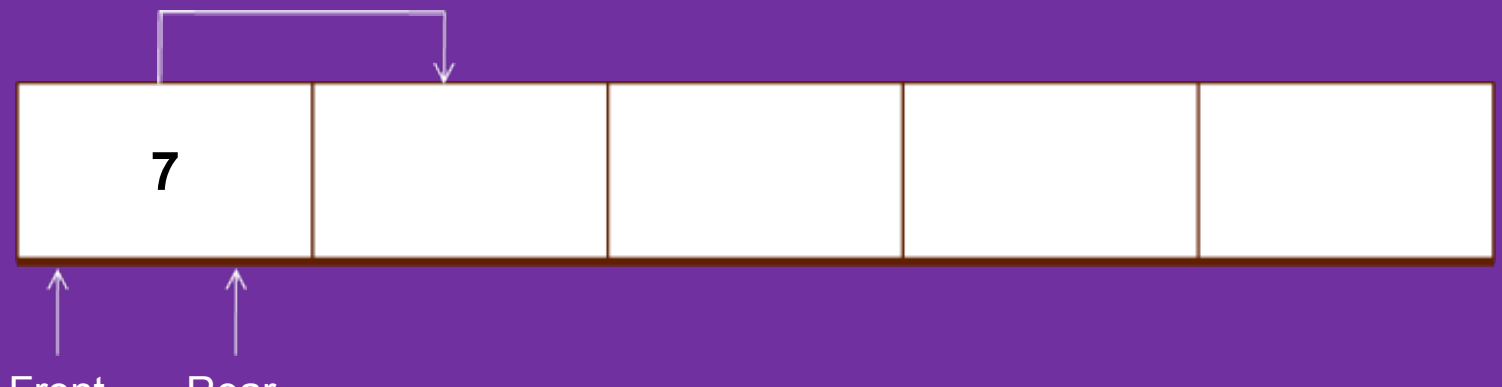
Representation of Deque

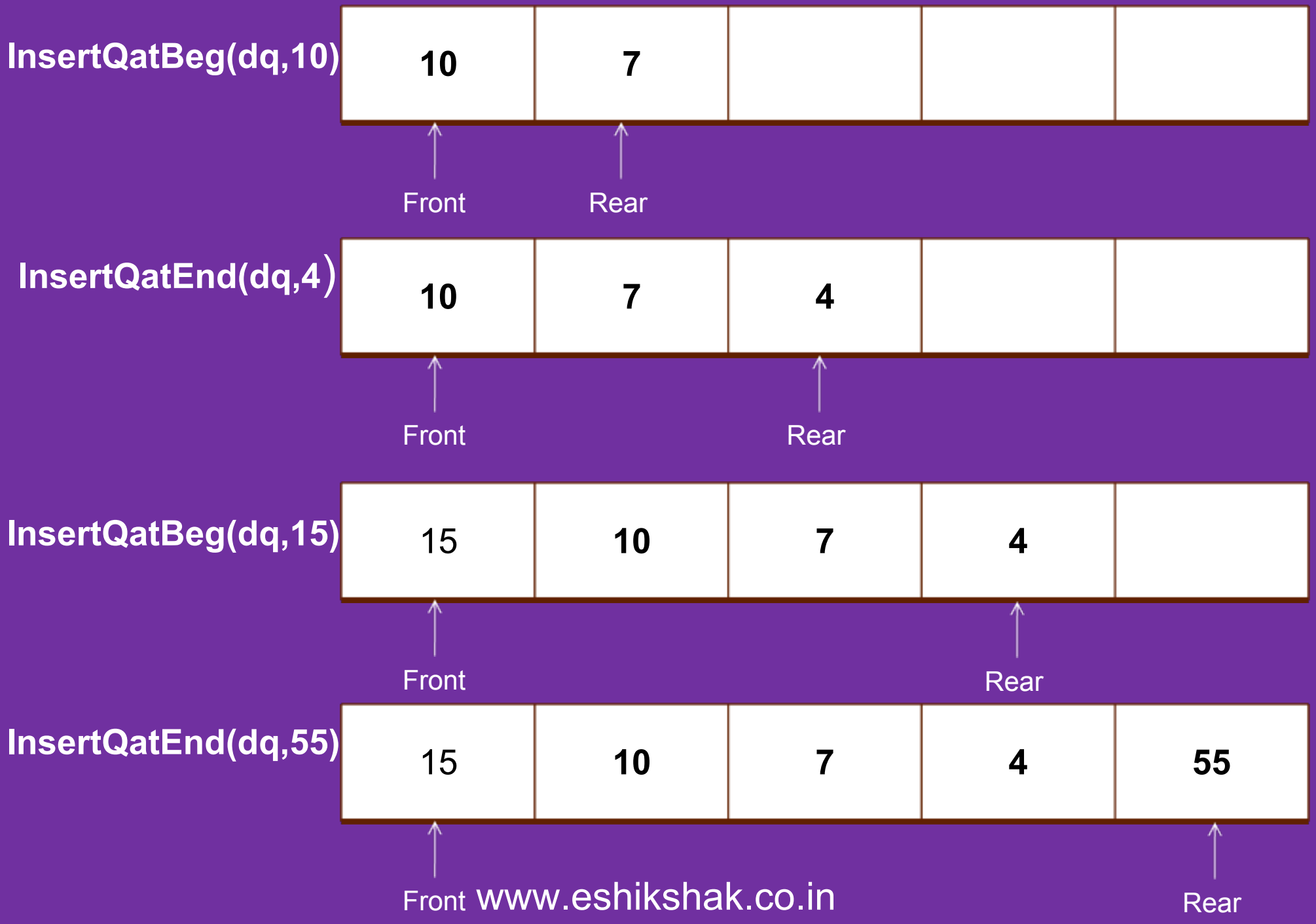
Inserting Element in Deque

InsertQatEnd(dq,7)

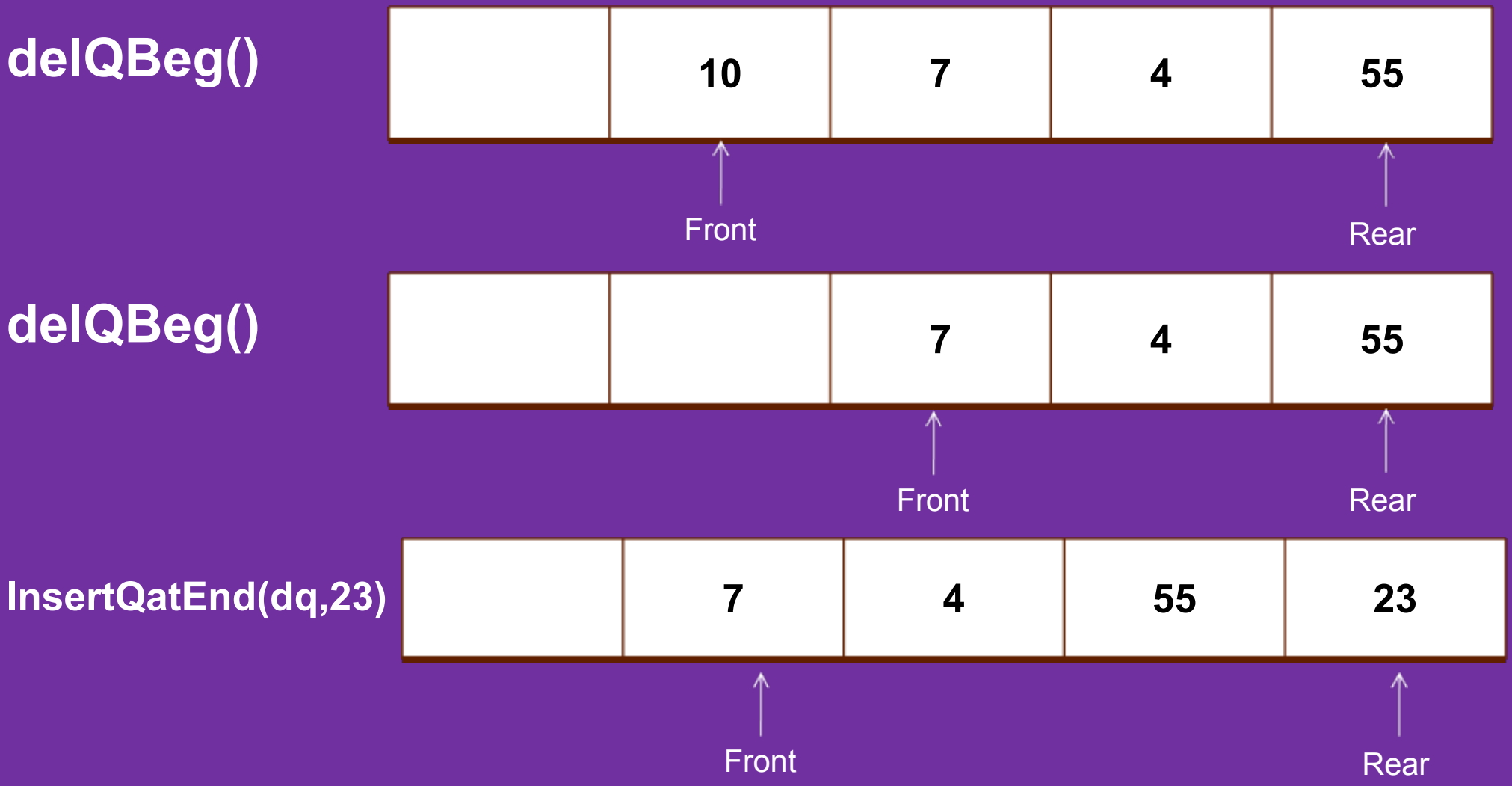
InsertQatBeg(dq,10)

InsertQatBeg(dq, 10)





Removing Element from Deque



Deque : Insert

- There are two variations of a deque. These are
  - Input Restricted Deque
    - An input restricted deque restricts the insertion of elements at one end only, but the deletion of elements can perform at both the ends.
  - Output Restricted Deque
    - An output restricted queue, restricts the deletion of elements at one end only, and allows insertion to be done at both the ends of deque

Input Restricted queue : Insert Element

Step 1: start
Step 2:[check condition for overflow]
If(rear==N-1 && front==0 or front=rear+1)
Print “over flow”
else
goto step 3
Step 3: [check condition]
If(front==null)
front = 0 and rear=-1
else
goto step 4
Step 4: [check condition and value]
If (rear== N -1)
rear=0
Else
rear=rear+1
Step 5: [Add value]
dq[rear]=value
Step 6:end

Input Restricted queue : Delete Beginning

Step 1: start
Step 2: [check condition for underflow]
If(rear==null & front==null)
Print”underflow”
else
goto step 3
Step 3: [delete]
dq[rear]=null
Step 4: [check condition]
If(rear==front)
front=rear=null
else
rear--;
Step 5: end

Input Restricted queue : Delete End

Step 1: start
Step2 : [check condition for under flow)
If(rear==null & front==null)
Print “under flow”
else
goto step 3
Step 3: [delete]
dq[front]=null
Step 4: [check condition]
If(rear==front)
rear=-1 and front=null
else
front=front+1
Step 5: end