QUEUE

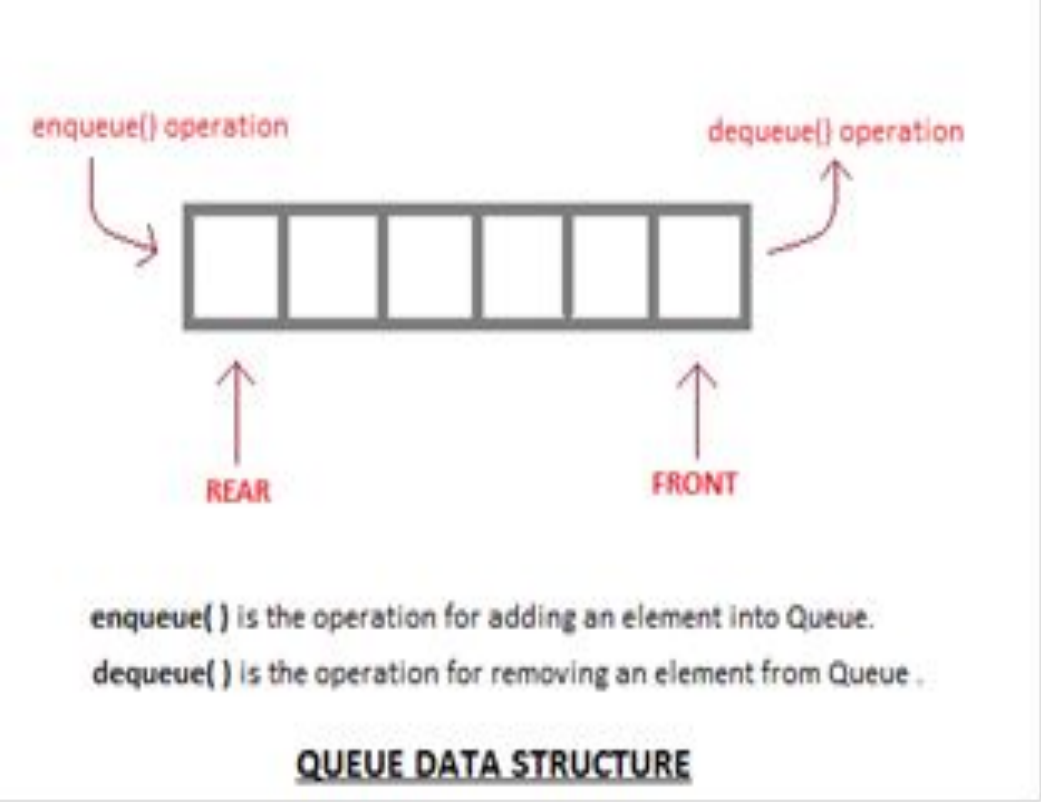
Visit for more Learning Resources

Queue

- Ordered collection of homogeneous elements
- Non-primitive linear data structure.
- A new element is added at one end called rear end and the existing elements are deleted from the other end called front end.
- This mechanism is called First-In-First-Out (FIFO).

e.g. People standing in Queue for Movie Ticket

Fig: Model of a Queue



Queue as ADT(abstract data type.)

- Queue is a data structure which allows a programmer to insert an element at one end known as “Rear” and to delete an element at other end known as “Front”.
- Queue is an abstract data type because it not only allows storing a elements but also allows to perform certain operation on these elements.

Queue as ADT(abstract data type.)

- These operations are as follows.
  - Initialize()
  - enqueue()
  - dequeue()
  - Isempty()
  - Isfull()
  - Display()

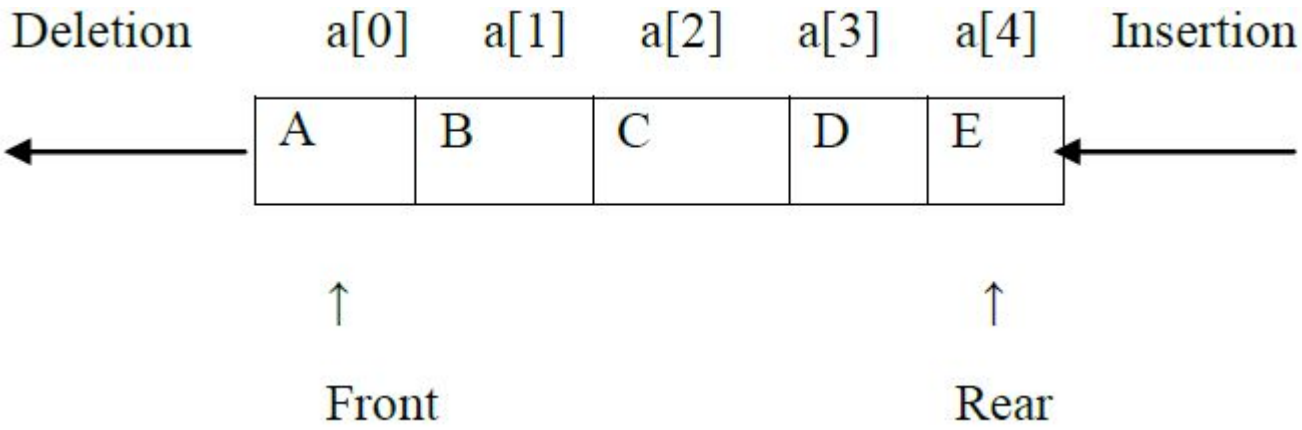
- Elements of queue:-
  - Front
  - Rear
  - array

Elements of queue:-

- **Front: -**
  - This end is used for deleting an element from a queue. Initially front end is set to -1. Front end is incremented by one when a new element has to be deleted from queue.

Rear: -

This end is used for inserting an element in a queue. Initially rear end is set to -1. rear end is incremented by one when a new element has to be inserted in queue.



Algorithm to insert element (enqueue Operation)

- **Step 1:** [check queue full condition]
  if rear = max -1 then write “queue is full”
  otherwise go to step 2
- **Step 2:** [increment rear point]
  rear = rear + 1
- **Step 3:** [insert element]
  q [rear] = Data
- **Step 4:** [check front pointer]
  if front = -1 then assign front =0
- **Step 5:** End

Algorithm to delete element (dequeue Operation)

- **Step 1:** [check queue empty condition]
  if front = -1 then write "queue is empty"
  otherwise go to step 2
- **Step 2:** [copy data]
  Data = q[front]
- Step 3: [check front and rear pointer]
  if front = rear then
  front = rear = -1
  otherwise
  front = front + 1
- **Step 4:** end

1

Front=-1 Queue is empty

0 1 2 3 4

Rear=-1

2

Front=0 Insert A

0 1 2 3 4

Rear=0

3

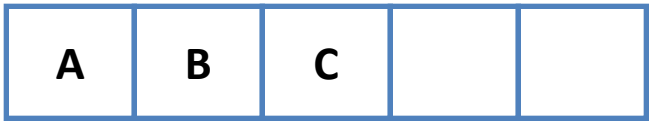
Front=0 Insert B

0 1 2 3 4

Rear=1

4

Front=0 Insert C

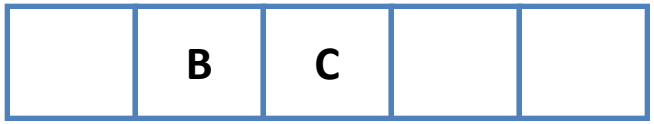


0 1 2 3 4

Rear=2

5

Front=1 Delete

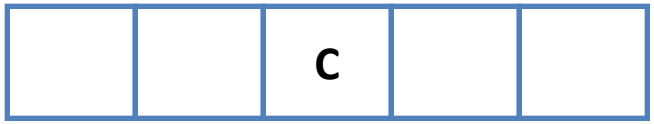


0 1 2 3 4

Rear=2

6

Front=2 Delete



0 1 2 3 4

Rear=2

7

Front=3 Delete

0 1 2 3 4

Rear=2

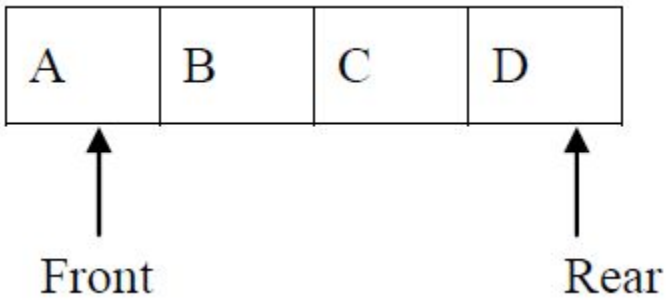
Queue is empty

Entry point is called Rear &
Exit point is called Front

‘Queue Full(Overflow)’ Condition

- **Queue Full(Overflow):**
  - Inserting an element in a queue which is already full is known as Queue Full condition (Rear = Max-1).
  - When the queue is fully occupied and enqueue() operation is called queue overflow occurs.

- **Example: Queue Full:**
  - Before inserting an element in queue 1st check whether space is available for new element in queue. This can be done by checking position of rear end. Array begins with 0th index position & ends with max-1 position. If numbers of elements in queue are equal to size of queue i.e. if rear end position is equal to max-1 then queue is said to be full. **Size of queue = 4**



‘Queue Empty(Underflow)’ Condition

- **Queue Empty:**
  - Deleting an element from queue which is already empty is known as Queue Empty condition (Front = Rear = -1)
  - When the queue is fully empty and dequeue() operation is called queue underflow occurs.
  -
- **Queue Empty:**
  - Before deleting any element from queue check whether there is an element in the queue. If no element is present inside a queue & front & rear is set to -1 then queue is said to be empty.
  - **Size of queue = 4**
  - Front = Rear = -1

Disadvantages of linear queue

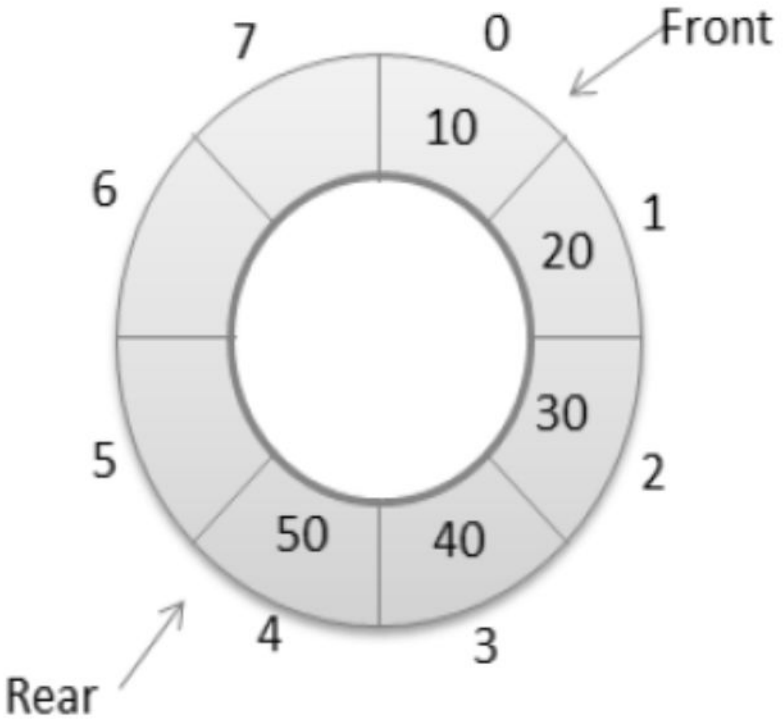
- On deletion of an element from existing queue, front pointer is shifted to next position.
- This results into virtual deletion of an element.
- By doing so memory space which was occupied by deleted element is wasted and hence inefficient memory utilization is occur.

Overcome disadvantage of linear queue:

- To overcome disadvantage of linear queue, **circular queue** is use.
- We can solve this problem by joining the front and rear end of a queue to make the queue as a circular queue .
- Circular queue is a linear data structure. It follows FIFO principle.
- In circular queue the last node is connected back to the first node to make a circle.

Overcome disadvantage of linear queue:

- It is also called as “Ring buffer”.
- Items can inserted and deleted from a queue in O(1) time.



Representation Of Queues

1. Using an array
2. Using linked list

Types Of Queue

1. Circular Queue

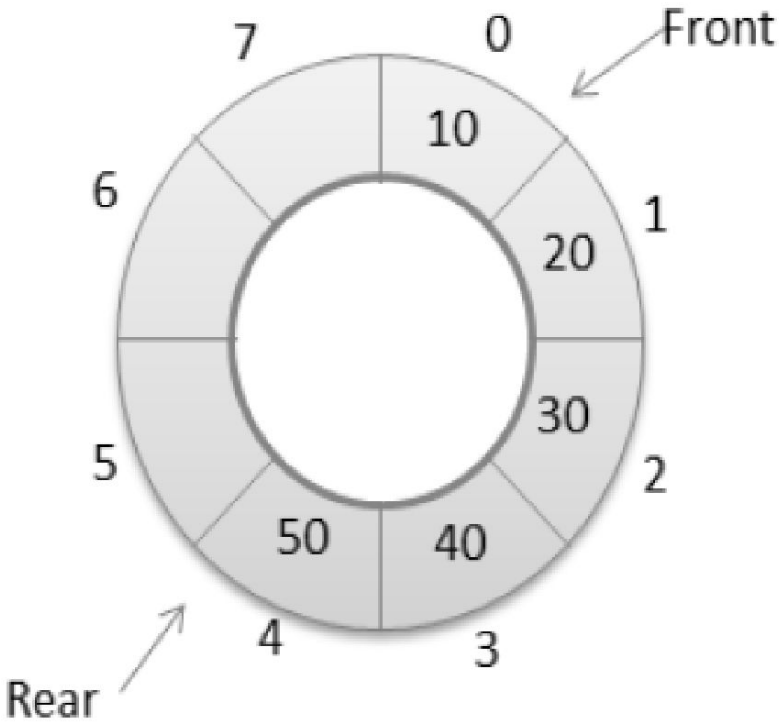
2. Dequeue (Double Ended Queue)

3. Priority Queue

CIRCULAR QUEUE

- A queue, in which the last node is connected back to the first node to form a cycle, is called as circular queue.
- Circular queue are the queues implemented in circular form rather than in a straight line.
- Circular queues overcome the problem of unutilized space in linear queue implemented as an array.
- The main disadvantage of linear queue using array is that when elements are deleted from the queue, new elements cannot be added in their place in the queue, i.e. the position cannot be reused.

CIRCULAR QUEUE



CIRCULAR QUEUE IMPLEMENTATION

- After rear reaches the last position, i.e. MAX-1 in order to reuse the vacant positions, we can bring rear back to the 0th position, if it is empty, and continue incrementing rear in same manner as earlier.
- Thus rear will have to be incremented circularly.
- For deletion, front will also have to be incremented circularly..

Enqueue(Insert) operation on Circular Queue:

- **Step 1:** Check for queue full
- If rear=max–1 and front=0 or if front=rear+1 then circular queue is full and insertion operation is not possible. otherwise go to step 2
- **Step 2:** Check position of rear pointer

  If rear=max–1

  then set rear=0 otherwise increment rear by 1.

  rear=(rear+1)%MAX
- **Step 3:** Insert element at the position pointer by rear pointer.

  q[rear]=Data
- **Step 4:** Check the position of front pointer

  If front=–1 then set front as 0.

Dequeue (Delete) operation on Circular Queue:

- **Step 1:** Check for queue empty **if (front = -1)**
  then circular queue is empty and deletion operation is not possible. otherwise go to step 2
- **Step 2:** Check for position of front and rear pointers.
  if front = rear then
  Data = q[front];
  set front=rear=-1
- **Step 3:** **C**heck position of front
  if front = Max-1
  Data = q[front];
  then set front=0;
  otherwise
  Data = q[front]; front = (front+1)%MAX

PRIORITY QUEUE

- A priority Queue is a collection of elements where each element is assigned a priority and the order in which elements are deleted and processed is determined from the 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 are added to the queue.

The priority queue implementation

- The priority queue is again implemented in two way

1. array/sequential representation.
2. Dynamic/ linked representation.

In priority queue node is divided into three parts

Structure of priority queue Node

PRIORITY QUEUE

- An example where priority queue are used is in operating systems.
- The operating system has to handle a large number of jobs.
- These jobs have to be properly scheduled.
- The operating system assigns priorities to each type of job.
- The jobs are placed in a queue and the job with the highest priority will be executed first.

PRIORITY QUEUE

- **Advantages:-**
  - Preferences to the higher priority process are added at the beginning.
  - Keep the list sorted in *increasing* order.

Applications of Queue

- Queue, as the name suggests is used whenever we need to have any group of objects in an order in which the first one coming in, also gets out first while the others wait for there turn, like in the following scenarios :
  - Serving requests on a single shared resource, like a printer, CPU task scheduling etc.
  - In real life, Call Center phone systems will use Queues, to hold people calling them in an order, until a service representative is free.
  - Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive, First come first served.

Distinguish between stack and queue

Sr.No	STACK	QUEUE
1	It is LIFO(Last In First Out) data structure	It is FIFO (First In First Out) data structure.
2	Insertion and deletion take place at only one end called top	Insertion takes place at rear and deletion takes place at front.
3	It has only one pointer variable	It has two pointer variables.
4	No memory wastage	Memory wastage in linear queue
5	Operations: 1.push() 2.pop()	Operations: 1.enqueue() 2.dequeue()
6	In computer system it is used in procedure calls	In computer system it is used time/resource sharing
7.	Plate counter at marriage reception is an example of stack	Student standing in a line at fee counter is an example of queue.