## linear vs circular queue

| Basis of comparison | Linear Queue | Circular Queue |
|---|---|---|
| **Meaning** | The linear queue is a type of linear data structure that contains the elements in a sequential manner. | The circular queue is also a linear data structure in which the last element of the Queue is connected to the first element, thus creating a circle. |
| **Insertion and Deletion** | In linear queue, insertion is done from the rear end, and deletion is done from the front end. | In circular queue, the insertion and deletion can take place from any end. |
| **Memory space** | The memory space occupied by the linear queue is more than the circular queue. | It requires less memory as compared to linear queue. |
| **Memory utilization** | The usage of memory is inefficient. | The memory can be more efficiently utilized. |
| **Order of execution** | It follows the FIFO principle in order to perform the tasks. | It has no specific order for execution. |

## Priority queue:

A priority queue is a collection of elements such that each element has been assigned a priority and the order in which elements are deleted and processed comes from the following rules:.

 An element of higher priority is processed before any element of lower priority.

 If two elements has same priority then they are processed according to the order in which they were added to the queue.

The best application of priority queue is observed in CPU scheduling.

✓ The jobs which have higher priority are processed first.

✓ If the priority of two jobs is same this jobs are processed according to their position in queue.

✓ A short job is given higher priority over the longer one.

**Types of priority queues:**

 ***Ascending priority queue(min priority queue):***

An ***ascending priority queue*** is a collection of items into which items can be inserted arbitrarily but from which only the smallest item can be removed.

 ***Descending priority queue(max priority queue)***

An **descending priority queue** is a collection of items into which items can be inserted

arbitrarily but from which only the largest item can be removed.

***Priority QUEUE Operations:***

**Insertion :**

The insertion in Priority queues is **the same as** in non-priority queues.

**Deletion :**

Deletion requires a search for the element of highest priority and deletes the element with highest priority. The following methods can be used for deletion/removal from a given Priority Queue:

✓ An empty indicator replaces deleted elements.

✓ After each deletion elements can be moved up in the array decrementing the rear.

✓ The array in the queue can be maintained as an ordered circular array.

## The Queue as a ADT

The queue abstract data type is defined by the following structure and operations. A queue is structured, as described above, as an ordered collection of items which are added at one end, called the "rear," and removed from the other end, called the "front." Queues maintain a FIFO ordering property. The queue operations are given below.

- A queue **q** of type **T** is a finite sequence of elements with the operations
  - ***MakeEmpty(q):*** To make **q** as an empty queue
  - ***IsEmpty(q):*** To check whether the queue **q** is empty. Return true if **q** is empty, return false otherwise.
  - ***IsFull(q):*** To check whether the queue **q** is full. Return true in **q** is full, return false otherwise.
  - ***Enqueue(q, x):*** To insert an item **x** at the rear of the queue, if and only if **q** is not full.
  - ***Dequeue(q):*** To delete an item from the front of the queue **q** if and only if **q** is not empty.
  - ***Traverse (q):*** To read entire queue that is display the content of the queue.

### Applications of queue:

- Task waiting for the printing
- Time sharing system for use of CPU
- For access to disk storage
- Task scheduling in operating system