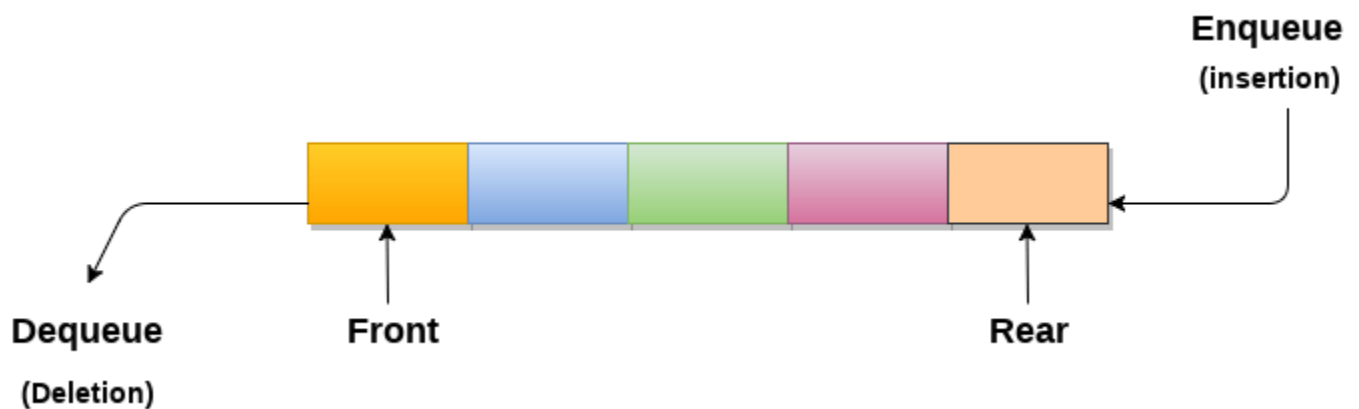


Queue ADT

1. A queue can be defined as an ordered list which enables insert operations to be performed at one end called **REAR** and delete operations to be performed at another end called **FRONT**.
2. Queue is referred to be as First In First Out list.
3. For example, people waiting in line for a rail ticket form a queue.



Applications of Queue

Due to the fact that queue performs actions on first in first out basis which is quite fair for the ordering of actions. There are various applications of queues discussed as below.

1. Queues are widely used as waiting lists for a single shared resource like printer, disk, CPU.
2. Queues are used in asynchronous transfer of data (where data is not being transferred at the same rate between two processes) for eg. pipes, file IO, sockets.
3. Queues are used as buffers in most of the applications like MP3 media player, CD player, etc.
4. Queue are used to maintain the play list in media players in order to add and remove the songs from the play-list.
5. Queues are used in operating systems for handling interrupts.

Complexity

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Queue	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$

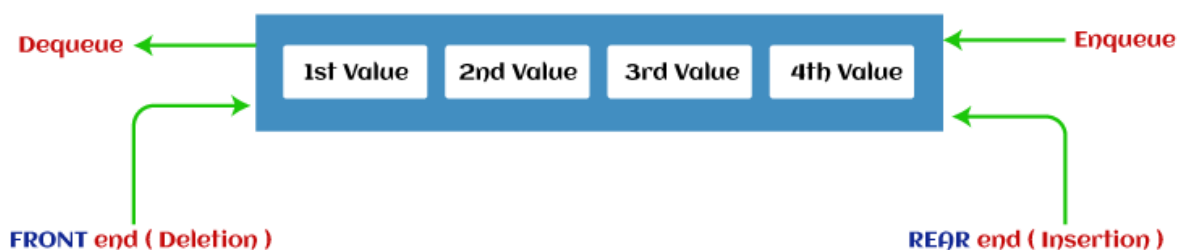
What is a Queue?

Queue is the data structure that is similar to the queue in the real world. A queue is a data structure in which whatever comes first will go out first, and it follows the FIFO (First-In-First-Out) policy.

Queue can also be defined as the list or collection in which the insertion is done from one end known as the **rear end** or the **tail** of the queue, whereas the deletion is done from another end known as the **front end** or the **head** of the queue.

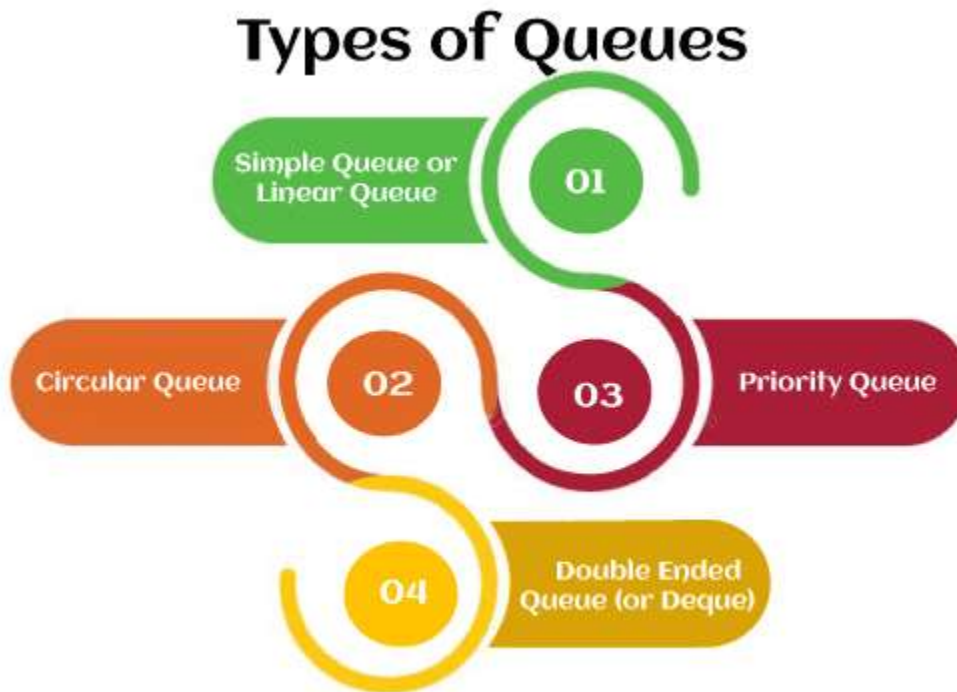
The real-world example of a queue is the ticket queue outside a cinema hall, where the person who enters first in the queue gets the ticket first, and the last person enters in the queue gets the ticket at last. Similar approach is followed in the queue in data structure.

The representation of the queue is shown in the below image -



Types of Queue

There are four different types of queue that are listed as follows -



- Simple Queue or Linear Queue
- Circular Queue
- Priority Queue
- Double Ended Queue (or Deque)

1. Simple Queue or Linear Queue

In Linear Queue, an insertion takes place from one end while the deletion occurs from another end. The end at which the insertion takes place is known as the rear end, and the end at which the deletion takes place is known as front end. It strictly follows the FIFO rule.



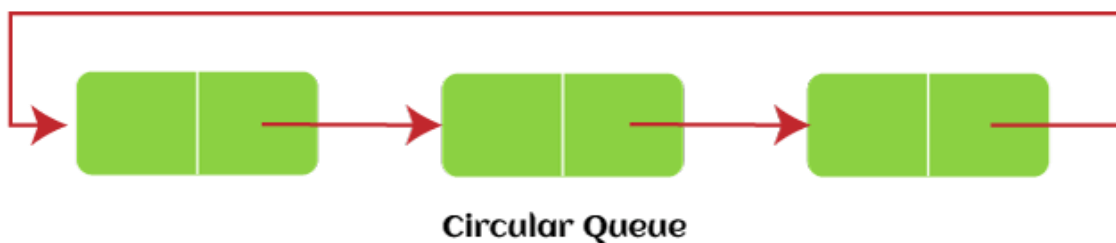
The major drawback of using a linear Queue is that insertion is done only from the rear end. If the first three elements are deleted from the Queue, we cannot insert more

elements even though the space is available in a Linear Queue. In this case, the linear Queue shows the overflow condition as the rear is pointing to the last element of the Queue.

2. Circular Queue

In Circular Queue, all the nodes are represented as circular. It is similar to the linear Queue except that the last element of the queue is connected to the first element. It is also known as Ring Buffer, as all the ends are connected to another end.

The representation of circular queue is shown in the below image -

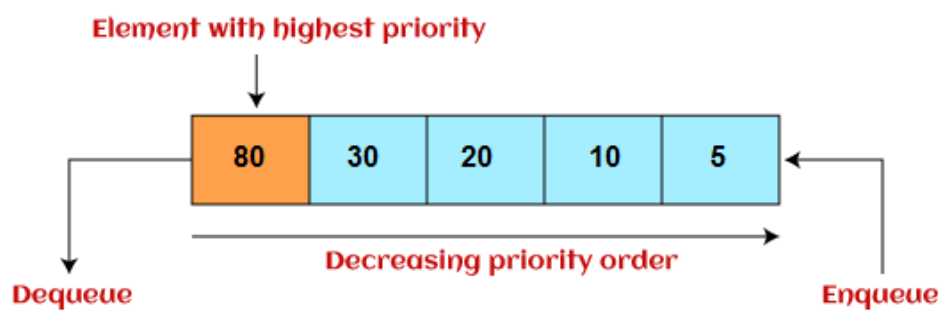


The drawback that occurs in a linear queue is overcome by using the circular queue. If the empty space is available in a circular queue, the new element can be added in an empty space by simply incrementing the value of rear. The main advantage of using the circular queue is better memory utilization.

3. Priority Queue

It is a special type of queue in which the elements are arranged based on the priority. It is a special type of queue data structure in which every element has a priority associated with it.

Suppose some elements occur with the same priority, they will be arranged according to the FIFO principle. The representation of priority queue is shown in the below image -



Insertion in priority queue takes place based on the arrival, while deletion in the priority queue occurs based on the priority. Priority queue is mainly used to implement the CPU scheduling algorithms.

There are two types of priority queue that are discussed as follows -

- **Ascending priority queue** - In ascending priority queue, elements can be inserted in arbitrary order, but only smallest can be deleted first. Suppose an array with elements 7, 5, and 3 in the same order, so, insertion can be done with the same sequence, but the order of deleting the elements is 3, 5, 7.
- **Descending priority queue** - In descending priority queue, elements can be inserted in arbitrary order, but only the largest element can be deleted first. Suppose an array with elements 7, 3, and 5 in the same order, so, insertion can be done with the same sequence, but the order of deleting the elements is 7, 5, 3.

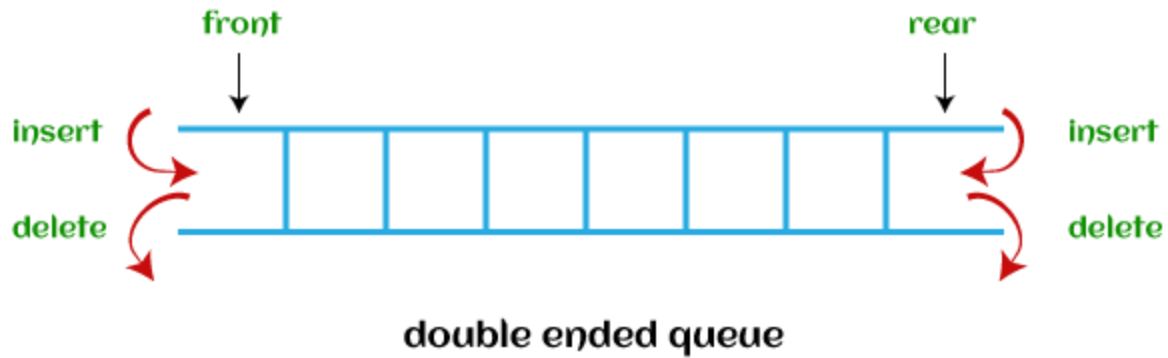
4. Deque (or, Double Ended Queue)

In Deque or Double Ended Queue, insertion and deletion can be done from both ends of the queue either from the front or rear. It means that we can insert and delete elements from both front and rear ends of the queue.

Deque can be used as a palindrome (a word, phrase, or sequence that reads the same backwards as forwards, e.g. *madam*) checker means that if we read the string from both ends, then the string would be the same.

Deque can be used both as stack and queue as it allows the insertion and deletion operations on both ends. Deque can be considered as stack because stack follows the LIFO (Last In First Out) principle in which insertion and deletion both can be performed only from one end. And in deque, it is possible to perform both insertion and deletion from one end, and Deque does not follow the FIFO principle.

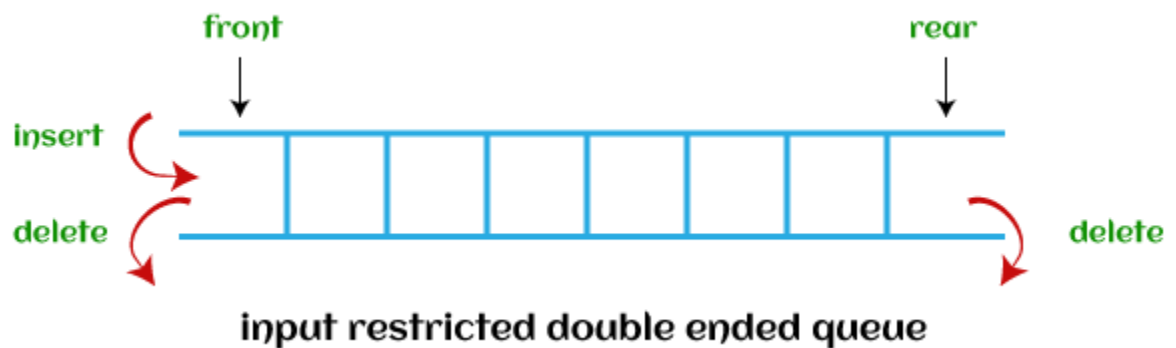
The representation of the deque is shown in the below image -



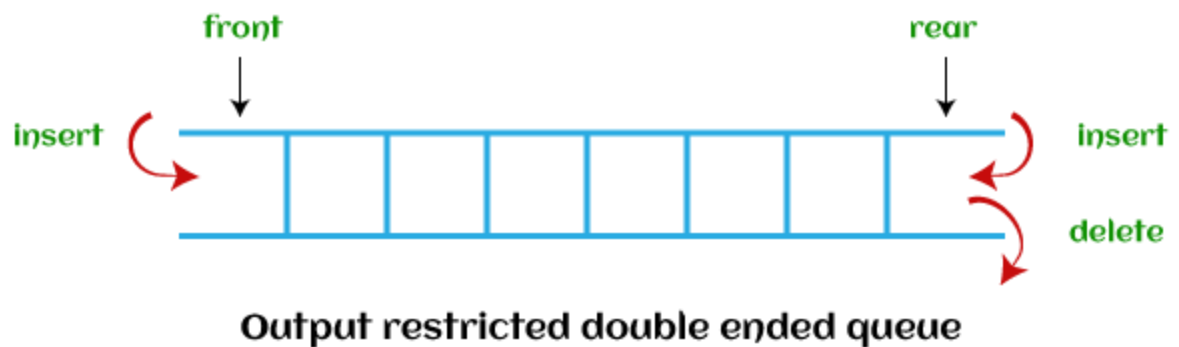
To know more about the deque, you can click the link - <https://www.javatpoint.com/ds-deque>

There are two types of deque that are discussed as follows -

- **Input restricted deque** - As the name implies, in input restricted queue, insertion operation can be performed at only one end, while deletion can be performed from both ends.



- **Output restricted deque** - As the name implies, in output restricted queue, deletion operation can be performed at only one end, while insertion can be performed from both ends.



Operations performed on queue

The fundamental operations that can be performed on queue are listed as follows -

- **Enqueue:** The Enqueue operation is used to insert the element at the rear end of the queue. It returns void.
- **Dequeue:** It performs the deletion from the front-end of the queue. It also returns the element which has been removed from the front-end. It returns an integer value.
- **Peek:** This is the third operation that returns the element, which is pointed by the front pointer in the queue but does not delete it.
- **Queue overflow (isfull):** It shows the overflow condition when the queue is completely full.
- **Queue underflow (isempty):** It shows the underflow condition when the Queue is empty, i.e., no elements are in the Queue.

Ways to implement the queue

There are two ways of implementing the Queue:

- **Implementation using array:** The sequential allocation in a Queue can be implemented using an array.
- **Implementation using Linked list:** The linked list allocation in a Queue can be implemented using a linked list.