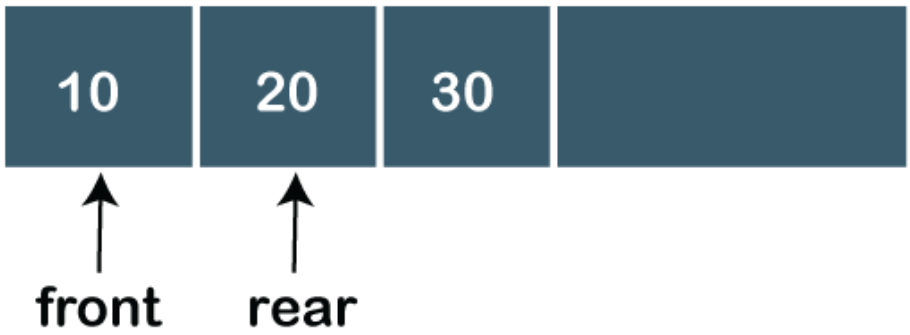
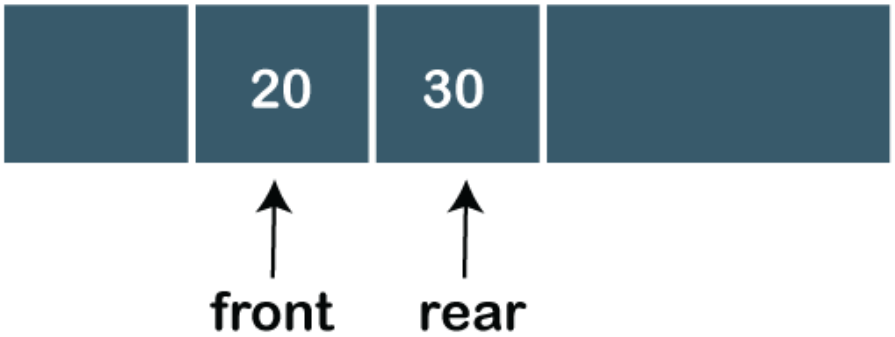
There are four types of Queues:

- **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 linear Queue can be represented, as shown in the below figure:



The above figure shows that the elements are inserted from the rear end, and if we insert more elements in a Queue, then the rear value gets incremented on every insertion. If we want to show the deletion, then it can be represented as:

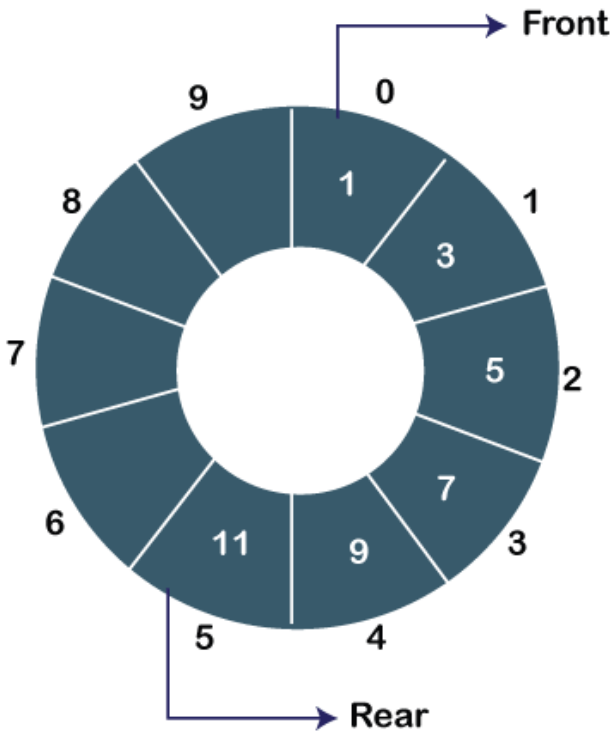


In the above figure, we can observe that the front pointer points to the next element, and the element which was previously pointed by the front pointer was deleted.

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.

- **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 circular queue can be represented as:



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.

- **Priority Queue**

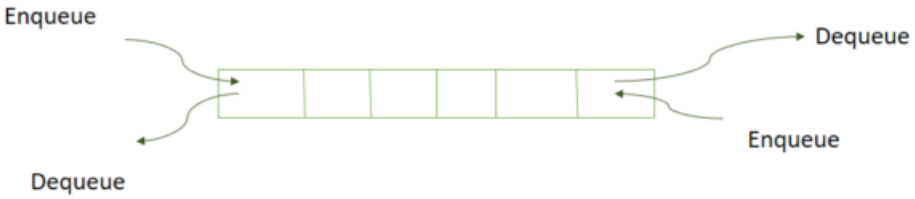
A priority queue is a special type of queue in which each element is associated with a priority and is served according to its priority. There are two types of Priority Queues. They are:

- **Ascending Priority Queue:** Element can be inserted arbitrarily but only smallest element can be removed. For example, suppose there is an array having elements 4, 2, 8 in the same order. So, while inserting the elements, the insertion will be in the same sequence but while deleting, the order will be 2, 4, 8.
- **Descending priority Queue:** Element can be inserted arbitrarily but only the largest element can be removed first from the given Queue. For example, suppose there is an array having elements 4, 2, 8 in the same order. So, while inserting the elements, the insertion will be in the same sequence but while deleting, the order will be 8, 4, 2.

The priority queue is primarily used to implement the CPU scheduling algorithms.

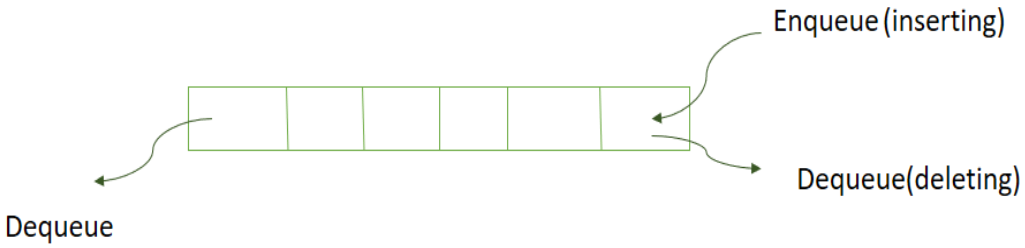
- **Deque (Double ended Queue)**

  Double Ended Queue is also a Queue data structure in which the insertion and deletion operations are performed at both the ends (front and rear). That means, we can insert at both front and rear positions and can delete from both front and rear positions.



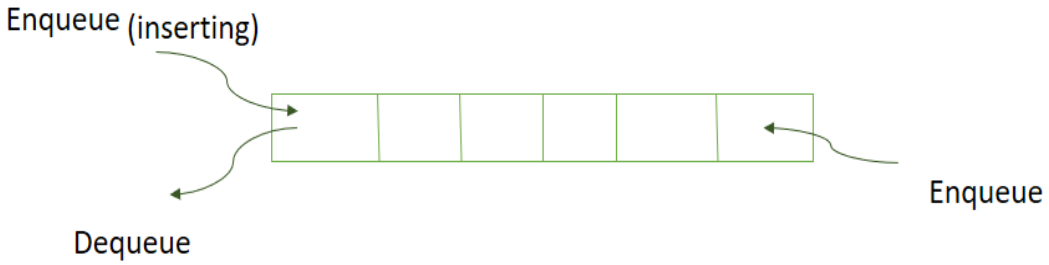
Since Deque supports both stack and queue operations, it can be used as both. The Deque data structure supports clockwise and anticlockwise rotations in O(1) time which can be useful in certain applications. Also, the problems where elements need to be removed and or added both ends can be efficiently solved using Deque.

1. **Input restricted Queue:** In this type of Queue, the input can be taken from one side only(rear) and deletion of element can be done from both side(front and rear). This kind of Queue does not follow FIFO(first in first out).



This queue is used in the cases where the consumption of the data needs to be in FIFO order but and if there is a need to remove the recently inserted data for some reasons and one such case can be irrelevant data, performance issue, etc.

2. **Output restricted Queue:** In this type of Queue, the input can be taken from both sides(rear and front) and the deletion of the element can be done from only one side(front).



This queue is used in the case where the inputs have some priority order to be executed and the input can be placed even in the first place so that it is executed first.