Different Types of Queues and its Applications

- A Queue is a linear data structure which follows a particular order in which the operations are performed.

- The order is **First in First out** (FIFO).

- A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first.
- The queue is used when things don't have to be processed immediately, but have to be processed in **First In First Out order** like Breadth First Search.

- This property of Queue makes it also useful in the following kind of scenarios.

1. When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.

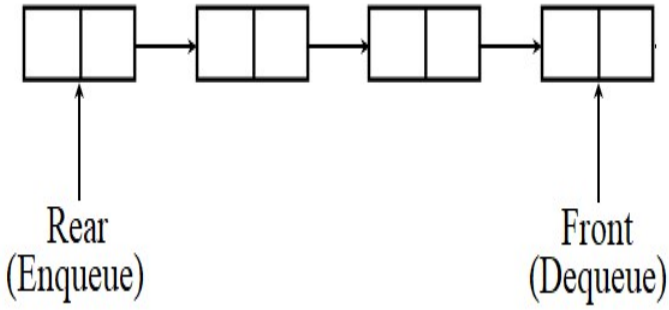
2. When data is transferred asynchronously (data not necessarily received at the same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

There are five different types of queues which are used in different scenarios. They are:

Types of Queues

1. Simple Queue

A simple queue is the most basic queue. In this queue, **the enqueue operation takes place at the rear, while the dequeue operation takes place at the front:**



Its applications are process scheduling, disk scheduling, memory management, IO buffer, pipes, call center phone systems, and interrupt handling.

2. Circular Queue:

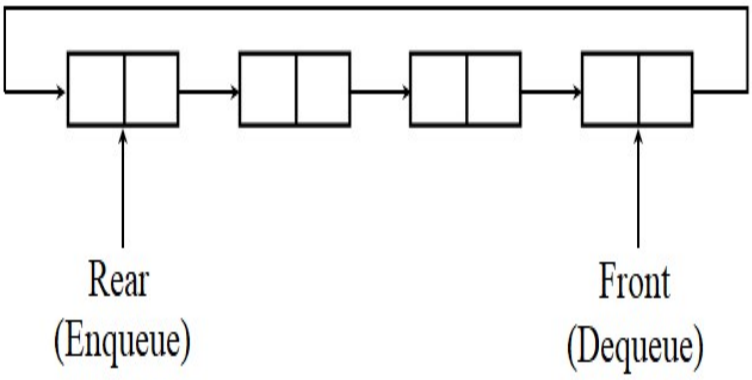
- Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle.
- It is also called **'Ring Buffer'**.
- This queue is primarily used in the following cases:

a. **Memory Management:** The unused memory locations in the case of ordinary queues can be utilized in circular queues.
b. **Traffic system:** In a computer-controlled traffic system, circular queues are used to switch on the traffic lights one by one repeatedly as per the time set.

c. **CPU Scheduling:** Operating systems often maintain a queue of processes that are ready to execute or that are waiting for a particular event to occur.

A circular queue **permits better memory utilization than a simple queue** when the queue has a fixed size.

- In this queue, the last node points to the first node and creates a circular connection.
- Thus, it allows us to insert an item at the first node of the queue when the last node is full and the first node is free.
- It's also called a ring buffer:
-



- It's used to switch on and off the lights of the traffic signal systems.
- Apart from that, it can be also used in place of a simple queue in all the applications mentioned above.

3. Priority Queue

A priority queue is a special kind of queue in which each item has a predefined priority of service.

- 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:

    1. **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.

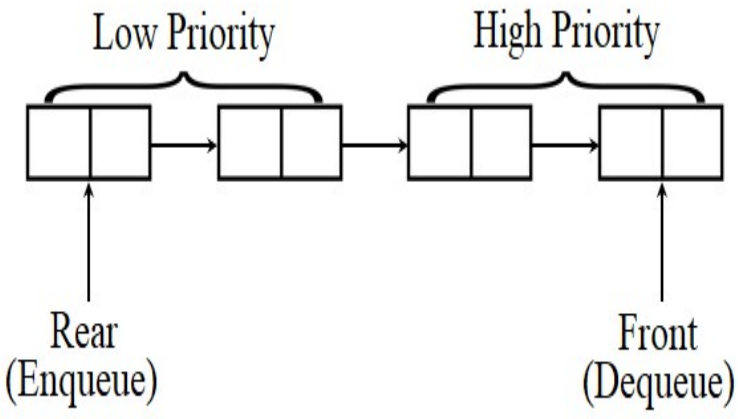
    2. **Descending priority Queue:**

        - 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.

- In this queue, the enqueue operation takes place at the rear in the order of arrival of the items, while the dequeue operation takes place at the front based on the priority of the items.
- That is to say that **an item with a high priority will be dequeued before an item with a low priority**.
- In the case, when two or more items have the same priority, then they'll be dequeued in the order of their arrival.

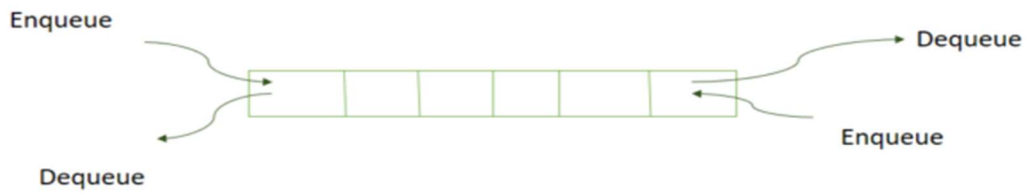
- Hence, it may or may not strictly follow the FIFO rule:



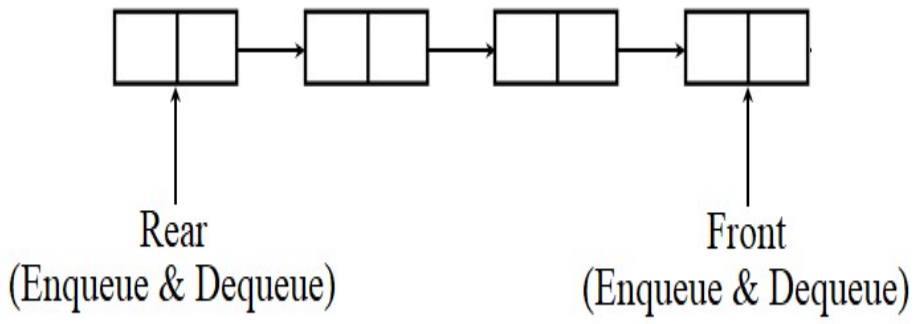
- It's used in interrupt handling, Prim's algorithm, Dijkstra's algorithm, A* search algorithm, heap sort, and Huffman code generation.

4. Double-Ended Queue (Deque)

- A deque is also a special type of queue. In this queue, the **enqueue and dequeue operations take place at both front and rear**.
- That means, we can insert an item at both the ends and can remove an item from both the ends. Thus, it may or may not adhere to the FIFO order.
- 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.

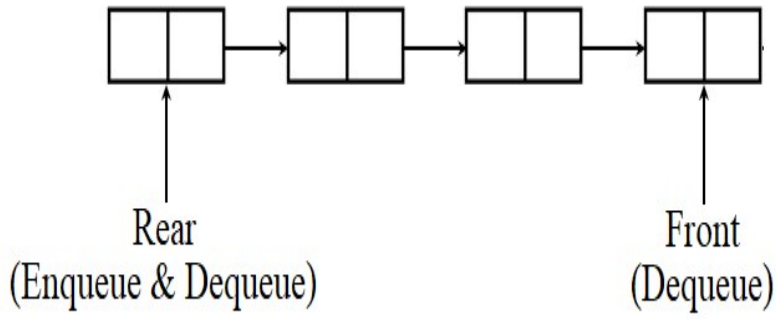


- It's used to save browsing history, perform undo operations, implement A-Steal job scheduling algorithm, or implement a stack or implement a simple queue.

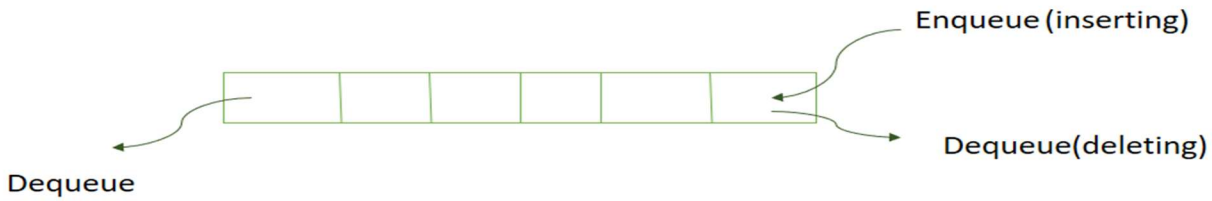
- Further, it has two special cases: **input-restricted deque** and **output-restricted deque**.
- **Input restricted Queue:**
- In this type of Queue, the input can be taken from one side only (rear) and deletion of element can be done from both side(front and rear).
- This kind of Queue does not follow 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.

- In the first case, the enqueue operation takes place only at the rear, but the dequeue operation takes place at both rear and front:

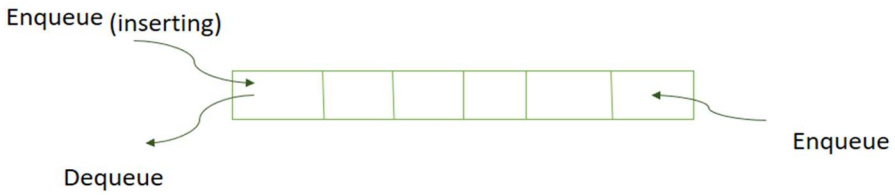


- An input-restricted queue is useful when we need to remove an item from the rear of the queue.

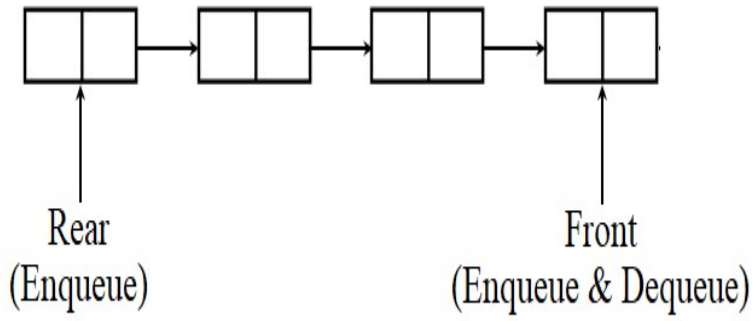


Output restricted Queue:

- 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.
-
- The enqueue operation takes place at both rear and front, but the dequeue operation takes place only at the front:



- An output-restricted queue is handy when we need to insert an item at the front of the queue.