Queue

By Mani Abedii

A clear, compact guide covering essential data structures and algorithms,

designed for easy understanding and quick learning.

A **queue** is a linear data structure that follows the First In, First Out (FIFO) principle. This means that the first element added to the queue will be the first one to be removed, similar to a line at a ticket counter or a checkout counter, where the person who arrives first is served first.

Queue can be implemented using an array, a linked list or two stacks.

A queue operates with specific restrictions on where elements can be inserted and removed:

1. **Enqueue:** This is the operation of adding an element to the queue. Insertion occurs at the **rear** (or **tail**) of the queue.
2. **Dequeue:** This operation removes an element from the queue. Deletion occurs from the **front** (or **head**) of the queue.

This structure ensures that elements are processed in the exact order in which they arrive, maintaining a FIFO sequence.
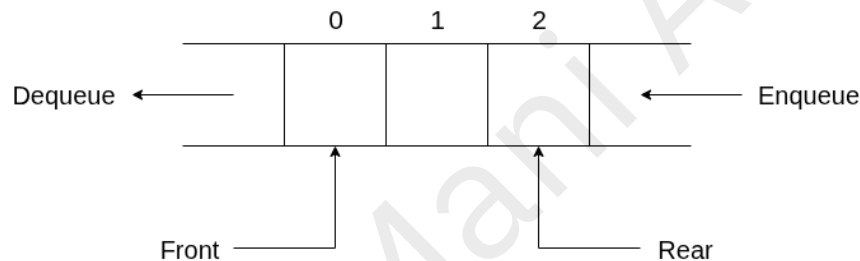
In a simple (linear) queue, the queue can become inefficient due to unused space at the front after several dequeue operations. Once rear reaches the end of the array, no further insertions can be performed, even if there are vacant positions at the beginning of the array (due to previous deletions). This situation is called the **false overflow problem** of a linear array-based queue. If we try other ways to implement using simple arrays, we may have to make Enqueue operation O(n) by shifting all elements one position ahead.

A circular queue overcomes this limitation by connecting the rear of the queue back to the front, forming a circle.
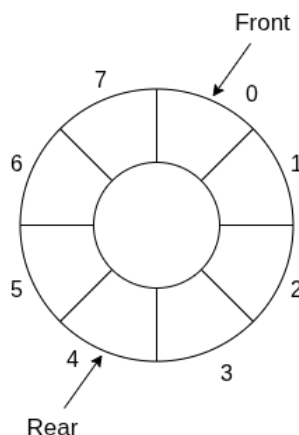
In a circular queue, the rear of the queue connects back to the front, forming a circle. This allows the queue to reuse spaces freed by previous dequeue operations, making it more efficient.

- **Enqueue:** Adds an element at the rear. If the rear reaches the end of the array, it wraps around to the beginning if space is available.
- **Dequeue:** Removes an element from the front. The front also wraps around when it reaches the end of the array.

Circular queues maintain the FIFO order while using memory efficiently, avoiding the wasted space problem of linear queues.