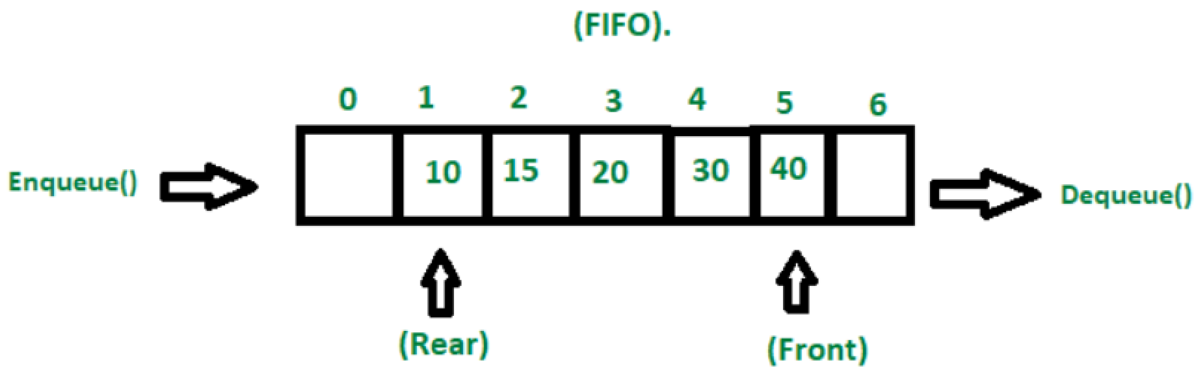
Queue Data Structure :

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. You can visualize it as a line of people waiting for a service: the person who arrives first is served first.

Key Operations :

1. **Enqueue**: Add an element to the end of the queue.
2. **Dequeue**: Remove the front element from the queue.
3. **Front**: Retrieve the front element without removing it.
4. **Back**: Retrieve the last element without removing it.
5. **Empty**: Check if the queue is empty.
6. **Size**: Get the number of elements in the queue.



Use cases for standard queues include:

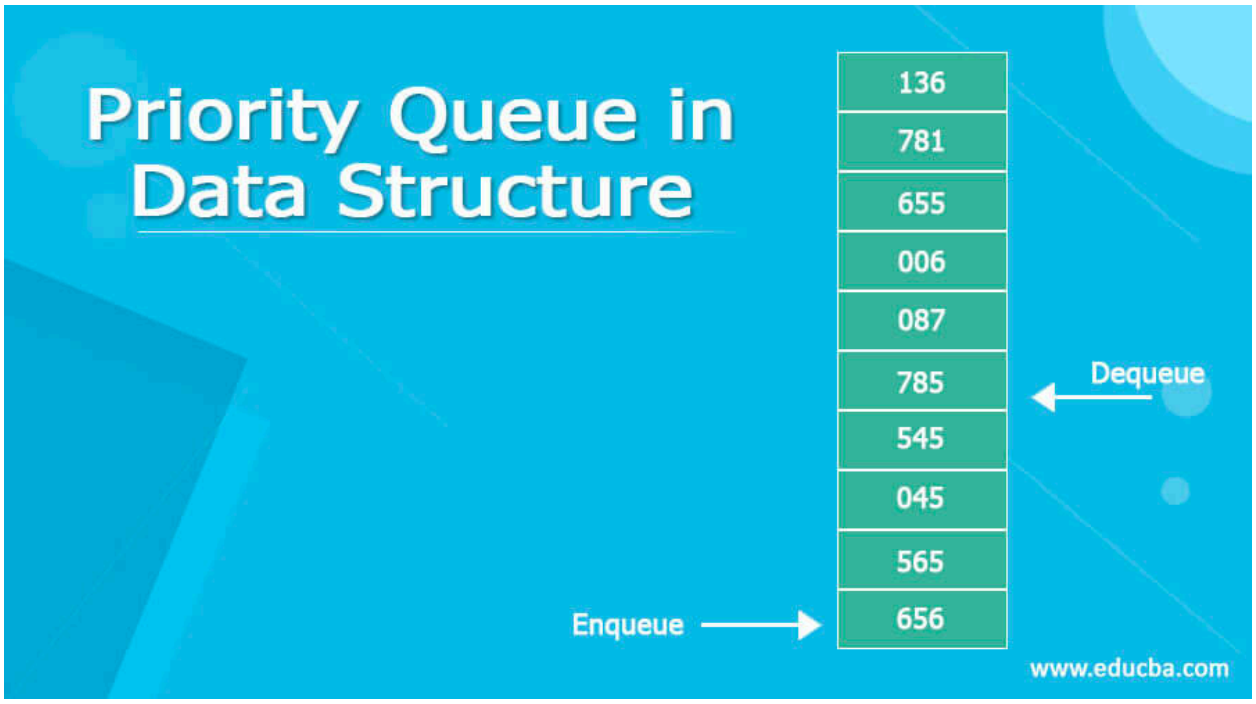
- **Task Scheduling**: Managing tasks in an operating system or a printing queue.
- **Breadth-First Search (BFS)**: Traversing graphs or trees level by level.
- **Buffer Management**: Storing data buffers in networking and multimedia applications.

Priority Queue :

A priority queue is a type of queue where each element has a priority associated with it. Elements with higher priorities are dequeued before those with lower priorities, regardless of their order in the queue. If two elements have the same priority, they are dequeued based on their order of insertion.

Key operations include:

- **Insert with Priority**: Adds an element with a specific priority.
- **Extract-Max/Min**: Removes and returns the element with the highest or lowest priority.
- **Peek**: Retrieves the highest or lowest priority element without removing it.



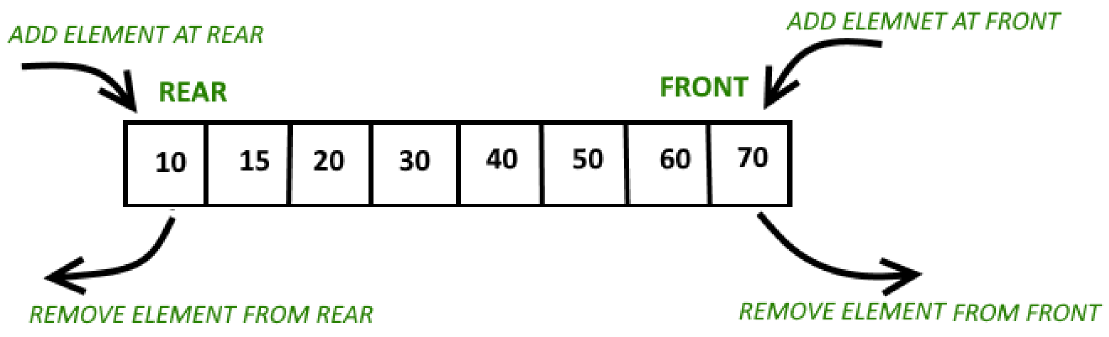
Priority queues are often implemented using binary heaps or other efficient data structures. Common applications include:

- **Task Scheduling**: Where tasks are prioritized based on urgency.
- **Graph Algorithms**: Like Dijkstra's shortest path algorithm.
- **Event-Driven Simulations**: Managing events based on their scheduled time.

Deque (Double-Ended Queue)

A deque allows insertion and deletion of elements from both the front and the back, offering greater flexibility than a standard queue. The operations supported include:

- **Push Front**: Adds an element to the front of the deque.
- **Push Back**: Adds an element to the back of the deque.
- **Pop Front**: Removes the front element.
- **Pop Back**: Removes the back element.
- **Front**: Retrieves the front element.
- **Back**: Retrieves the back element.
- **Empty**: Checks if the deque is empty.
- **Size**: Returns the number of elements in the deque.



Deques are useful in scenarios where elements need to be processed from both ends, such as:

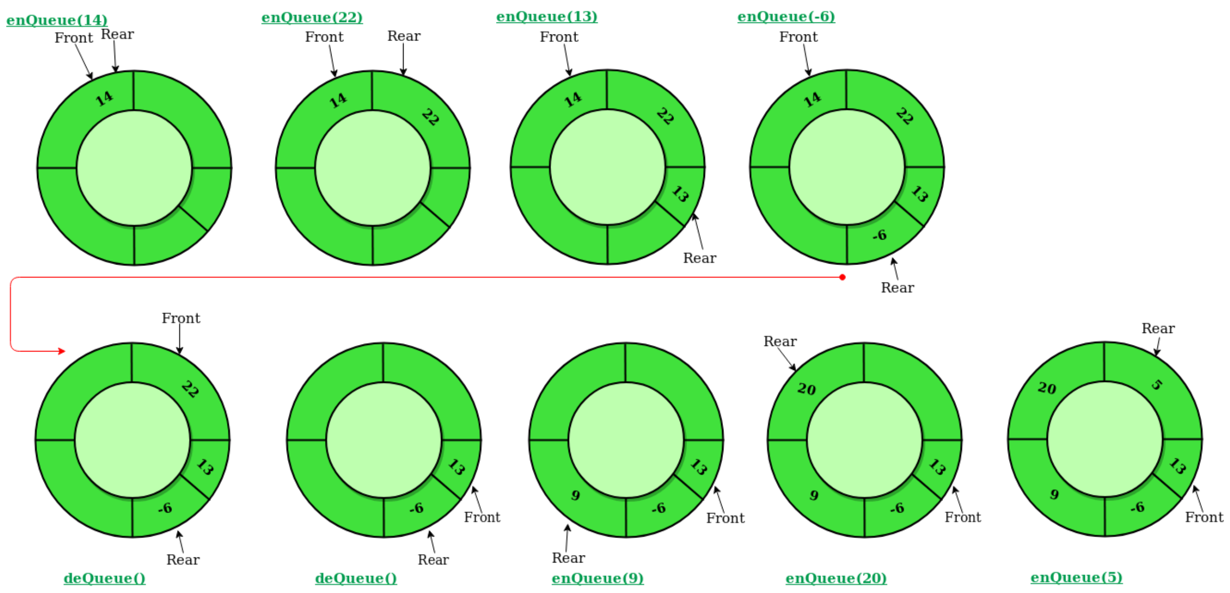
- **Palindrome Checking**: Comparing characters from both ends of a string.
- **Sliding Window Algorithms**: Managing a range of elements in array processing.
- **Job Scheduling**: Where tasks can be added or removed from either end of the queue.

Circular Queue

A circular queue, also known as a ring buffer, connects the end of the queue back to the front, forming a circle. This structure allows for efficient use of memory by reusing the space of dequeued elements. The circular queue supports the same operations as a standard queue but with a fixed size.

Key properties include:

- **Fixed Size**: The maximum number of elements it can hold is predefined.
- **Wrap Around**: When the end of the queue is reached, it wraps around to the beginning if there is space.



Circular queues are particularly useful in:

- **Buffer Management**: Such as in network buffers or audio/video streaming.
- **Round-Robin Scheduling**: Where tasks are cyclically processed.
- **Producer-Consumer Problems**: Managing data between producers and consumers in a fixed buffer.

LInks :

1) **STL Queue :**
   Queue in C++ Standard Template Library (STL) - GeeksforGeeks

2) **Priority Queue :**
   Intro : What is Priority Queue | GeeksforGeeks
   STL : Priority Queue in C++ STL | GeeksforGeeks

3) **Deque :**
   Working : How Deque Works Internally in C++? - GeeksforGeeks
   STL : Deque STL - GeeksforGeeks

Next, let's move on to some solved problems on Queues:

You could go through this code once, its great to understand reversal o queues with help of stacks.
https://leetcode.com/discuss/interview-question/2959311/Reversing-a-Queue

Code 360 - First Negative in every window
Lecture 61: Queue FAANG Interview Questions || Placement Series by Lov…

LeetCode 622 : Circular Queue
Design Circular Queue - Leetcode 622 - Python

Problems on Queue :

Top 50 Problems on Queue Data Structure asked in SDE Interviews - GeeksforGeeks