What is a Queue Data Structure?

Queue is also an abstract data type or a linear data structure, just like [stack data structure](), in which the first element is inserted from one end called the **REAR**(also called **tail**), and the removal of existing element takes place from the other end called as **FRONT**(also called **head**).

Before you continue reading about queue data structure, check these topics before to understand it clearly:

- [Data Structures and Algorithms]()
- [Stack Data Structure]()

This makes queue as **FIFO**(First in First Out) data structure, which me[ ] Index element inserted first will be removed first.

Which is exactly how queue system works in real world. If you go to a ticket counter to buy movie tickets, and are first in the queue, then you will be the first one to get the tickets. Right? Same is the case with Queue data structure. Data inserted first, will leave the queue first.

The process to add an element into queue is called **Enqueue** and the process of removal of an element from queue is called **Dequeue**.
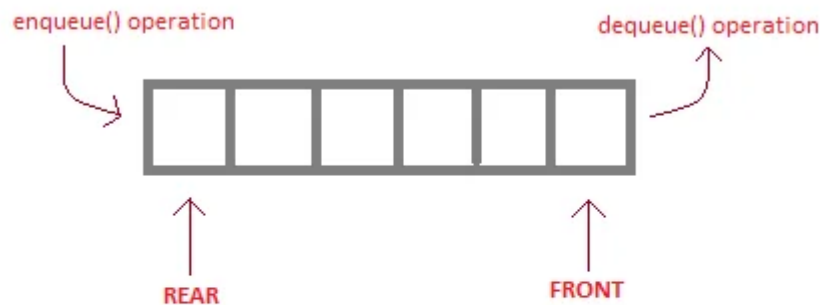
Home   Code   Library   Test   Account

enqueue( ) is the operation for adding an element into Queue.

dequeue( ) is the operation for removing an element from Queue .

QUEUE DATA STRUCTURE

Index

Basic features of Queue

1. Like stack, queue is also an ordered list of elements of similar data types.

2. Queue is a FIFO(First in First Out) structure.

3. Once a new element is inserted into the Queue, all the elements inserted before the new element in the queue must be removed, to remove the

Home   Code   Library   Test   Account

dequeuing it.

Applications of Queue

Queue, as the name suggests is used whenever we need to manage any group of objects in an order in which the first one coming in, also gets out first while the others wait for their turn, like in the following scenarios:

1. Serving requests on a single shared resource, like a printer, CPU task scheduling etc.

2. In real life scenario, Call Center phone systems uses Queues to hold people calling them in an order, until a service representative is free.

3. Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive i.e First come first served.

Implementation of Queue Data Structure

Queue can be implemented using an Array, Stack or Linked List. The way of implementing a queue is by using an Array.

Index

Initially the **head**(FRONT) and the **tail**(REAR) of the queue points at the first index of the array (starting the index of array from `0`). As we add elements to the queue, the **tail** keeps on moving ahead, always pointing to the position where the next element will be inserted, while the **head** remains at the first index.

| Home | Code | Library | Test | Account |
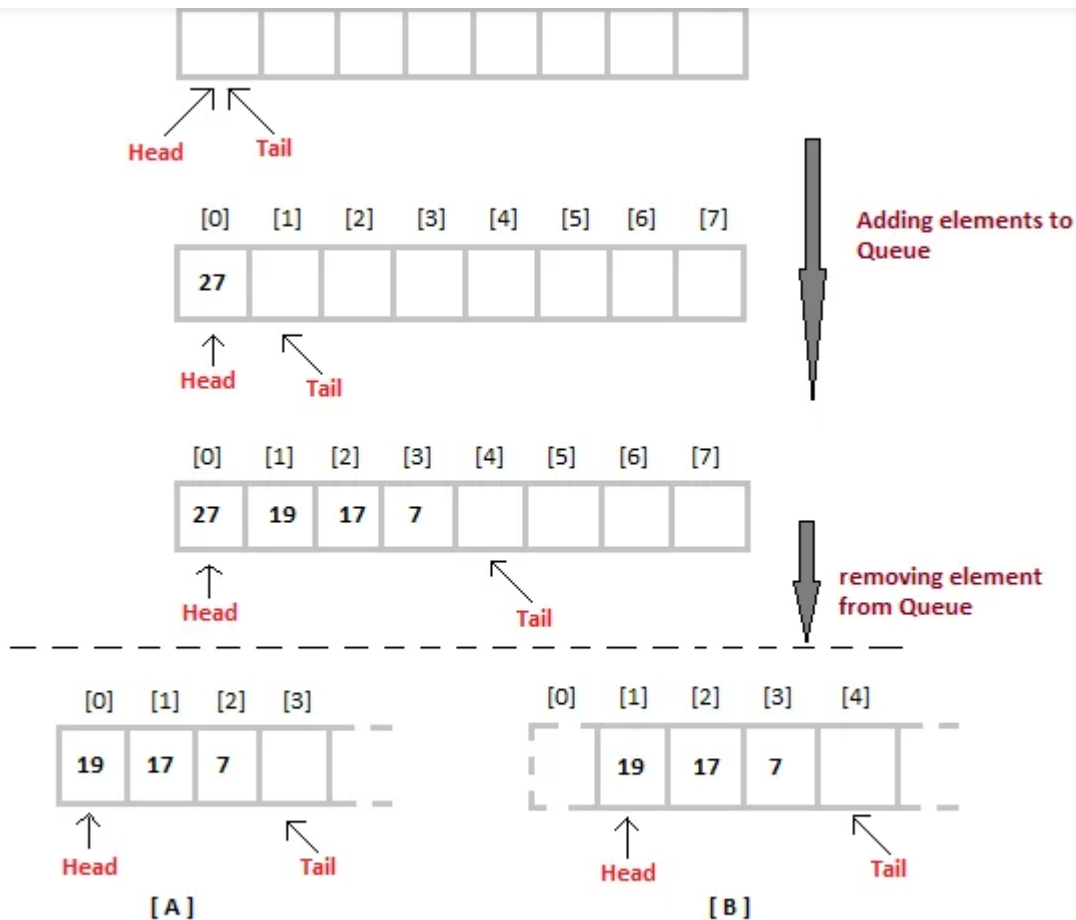
When we remove an element from Queue, we can follow two possibl[e] approaches (mentioned [A] and [B] in above diagram). In [A] approach, we remove the element at **head** position, and then one by one shift all the other elements in forward position.

In approach [B] we remove the element from **head** position and then move **head** to the next position.

In approach [A] there is an **overhead of shifting the elements one position forward** every time we remove the first element.

by one space each time.

Algorithm for ENQUEUE operation

1. Check if the queue is full or not.

2. If the queue is full, then print overflow error and exit the program.

3. If the queue is not full, then increment the tail and add the element.

Algorithm for DEQUEUE operation

1. Check if the queue is empty or not.

2. If the queue is empty, then print underflow error and exit the program.

3. If the queue is not empty, then print the element at the head and increment the head.

```cpp
/* Below program is written in C++ language */

#include<iostream>

using namespace std;

#define SIZE 10

class Queue
```

Index

Home   Code   Library   Test   Account

```
public:
Queue()
{
    rear = front = -1;
}
```

To implement approach [A], you simply need to change the `dequeue` method, and include a `for` loop which will shift all the remaining elements by one position.

```
return a[0];      //returning first element
for (i = 0; i < tail-1; i++)      //shifting all other elements
{
    a[i] = a[i+1];
    tail--;
}
```

Index

Complexity Analysis of Queue Operations

Just like Stack, in case of a Queue too, we know exactly, on which position new element will be added and from where an element will be removed, hence both these operations requires a single step.

- Enqueue: **O(1)**

Home   Code   Library   Test   Account

ADVERTISEMENT

🔲 DS & Algo MCQ Tests

Crack top Job Interviews.

Explore

Index

🔗 Advanced DS

Learn Advanced Data Structures.

Explore

🔷 C++ STL Tutorial

C++ Standard Template Library is best suited for Algorithms.

🏠 Home	</> Code	📖 Library	🔲 Test	👤 Account

studytonight.com

About Us

Testimonials

Authors

Collaborate

Privacy Policy

Terms

Contact Us

Suggest

© 2022 Studytonight Technologies Pvt. Ltd.

>_ **Learn Coding** (for beginners)

                                                    Index

📖 **Tutorial Library**

🔲 **Interview Tests**

\# **Curious**

>_ **Practice Coding**

🏠   </>   📖   🔲   👤
Home   Code   Library   Test   Account

Learn HTML

Learn CSS

Learn JavaScript

Resources

C Language

C++/STL

Java

DBMS

Python

PHP

Android

Game Development

Data Structure & Alog.

Operating System

Computer Network

Computer Architecture

More...

Interview Tests

Index

Java Interview Tests

Python Interview Tests

DBMS Interview Tests

Linux Interview Tests

Aptitude Tests

GATE 2022 Tests

More...

ADVERTISEMENT

| Home | Code | Library | Test | Account |

Index

Home

Code

Library

Test

Account