Article marked as read.

Dash

All

Articles

Videos

Problems

Quiz

Prev

Next

Introduction to Queues

Like

Stack

data structure,

Queue

is also a linear data structure that follows a particular order in which the operations are performed. The order is

F

irst

I

n

F

irst

O

ut (FIFO), which means that the element that is inserted first in the queue will be the first one to be removed from the queue. A good example of queue is any queue of consumers for a resource where the consumer who came first is served first.

Article marked as read.

Dash

All

Articles

Videos

Problems

Quiz

The difference between stacks and queues is in removing. In a stack, we remove the most recently added item; whereas, in a queue, we remove the least recently added item.

Operations on Queue:

Mainly the following four basic operations are performed on queue:

- **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
- **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
- **Front:** Get the front item from queue.
- **Rear:** Get the last item from queue.

Dash

All

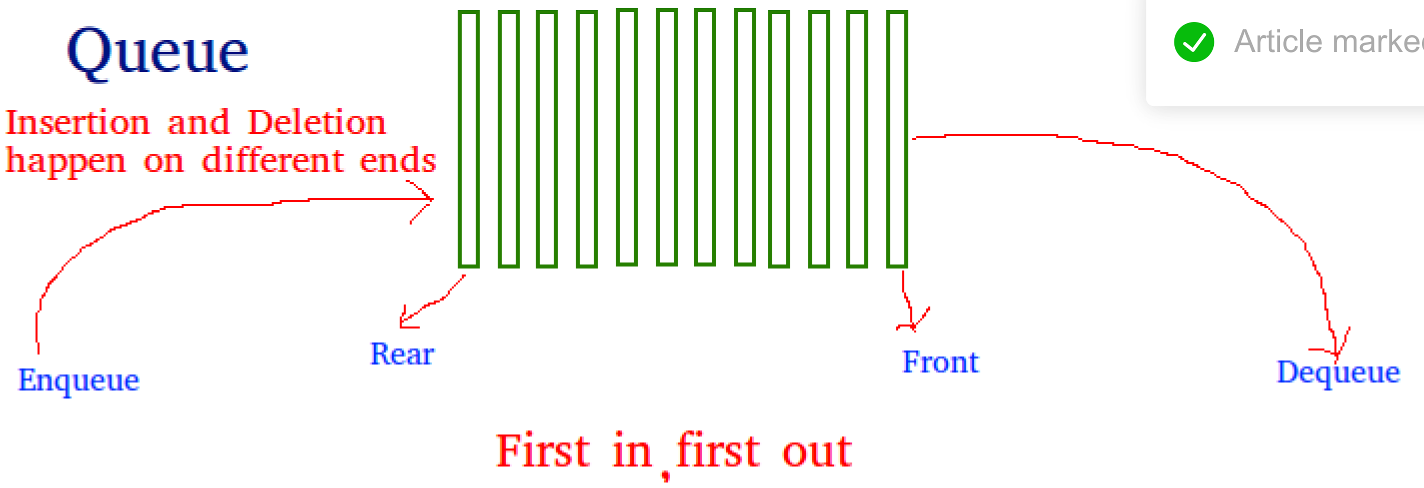
Articles

Videos

Problems

Quiz

Article marked as read.



Array implementation Of Queue

: For implementing a

queue

, we need to keep track of two indices - front and rear. We enqueue an item at the rear and dequeue an item from the front. If we simply increment front and rear indices, then there may be problems, the front may reach the end of the array. The solution to this problem is to increase front and rear in a circular manner.

Consider that an Array of size

N

is taken to implement a queue. Initially, the size of the queue will be zero(0). The total capacity of the queue will be the size of the array i.e. N. Now initially, the index

front

will be equal to 0, and

rear

will be equal to N-1. Every time an item is inserted, so the index

rear

will increment by one, hence increment it as:

rear = (rear + 1)%N

and everytime an item is removed, so the front index will shift to right by 1 place, hence increment it as:

front = (front + 1)%N

.

Example

:

Dash

All

Articles

Videos

Problems

Quiz

```
Array = queue[N].
front = 0, rear = N-1.
N = 5.

Operation 1:
enque(5);
front = 0,
rear = (N-1 + 1)%N = 0.
Queue contains: [5].

Operation 2:
enque(10);
front = 0,
rear = (rear + 1)%N = (0 + 1)%N = 1.
Queue contains: [5, 10].

Operation 3:
enque(15);
front = 0,
rear = (rear + 1)%N = (1 + 1)%N = 2.
Queue contains: [5, 10, 15].

Operation 4:
deque();
print queue[front];
front = (front + 1)%N = (0 + 1)%N = 1.
Queue contains: [10, 15].
```

Article marked as read.

Dash
All
Articles
Videos
Problems
Quiz

Below is the Array implementation of queue in C++ and Java:

Article marked as read.

C++ Java

```
// Java program for array implementation of queue

// A class to represent a queue
class Queue
{
    int front, rear, size;
    int  capacity;
    int array[];

    public Queue(int capacity) {
         this.capacity = capacity;
         front = this.size = 0;
         rear = capacity - 1;
         array = new int[this.capacity];

    }

    // Queue is full when size becomes equal to
    // the capacity
    boolean isFull(Queue queue)
    {  return (queue.size == queue.capacity);
```

Dash

All

Articles

Videos

Problems

Quiz

```
    }

    // Queue is empty when size is 0
    boolean isEmpty(Queue queue)
    {  return (queue.size == 0); }

    // Method to add an item to the queue.
    // It changes rear and size
    void enqueue( int item)
    {
        if (isFull(this))
            return;
        this.rear = (this.rear + 1)%this.capacity;
        this.array[this.rear] = item;
        this.size = this.size + 1;
        System.out.println(item+ " enqueued to queue");
    }

    // Method to remove an item from queue.
    // It changes front and size
    int dequeue()
    {
        if (isEmpty(this))
            return Integer.MIN_VALUE;

        int item = this.array[this.front];
        this.front = (this.front + 1)%this.capacity;
        this.size = this.size - 1;
        return item;
```

Article marked as read.

Dash
All
Articles
Videos
Problems
Quiz

Article marked as read.

```
    }

    // Method to get front of queue
    int front()
    {
        if (isEmpty(this))
            return Integer.MIN_VALUE;

        return this.array[this.front];
    }

    // Method to get rear of queue
    int rear()
    {
        if (isEmpty(this))
            return Integer.MIN_VALUE;

        return this.array[this.rear];
    }
}

// Driver class
public class Test
{
    public static void main(String[] args)
    {
        Queue queue = new Queue(1000);

        queue.enqueue(10);
```

```
        queue.enqueue(20);
        queue.enqueue(30);
        queue.enqueue(40);

        System.out.println(queue.dequeue() +
                      " dequeued from queue\n");

        System.out.println("Front item is " +
                               queue.front());

        System.out.println("Rear item is " +
                                queue.rear());
    }
}
```

Article marked as read.

Dash
All
Articles
Videos
Problems
Quiz

Output:

```
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
10 dequeued from queue
Front item is 20
```

Get 90% Refund!
Courses Tutorials Jobs Practice Contests

P

Time Complexity:

Time complexity of all operations such as enqueue(), dequeue(), isFull(), isEmpt

no loop in any of the operations.

Article marked as read.

Dash

All

Articles

Videos

Problems

Quiz

Applications of Queue:

Queue is used when things don't have to be processed immediatly, but have to be processed in

F

irst

I

n

F

irst

O

ut order like

Breadth First Search

. This property of Queue makes it also useful in 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 same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

Article marked as read.

Dash

All

Articles

Videos

Problems

Quiz

Marked as Read

Report An Issue

If you are facing any issue on this page. Please let us know.