QUEUE

=========================================================

Queue can be implemented **using an Array,**

**Stack**

**Or**

**Linked List**.

The easiest way of implementing a queue is by using an Array. 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 ).

=========================================================

Like [Stack](https://www.geeksforgeeks.org/stack-data-structure-introduction-program/), [Queue](http://en.wikipedia.org/wiki/Queue_%28data_structure%29)is a linear structure which follows a particular order in which the operations are performed. The order is **F**irst **I**n **F**irst **O**ut (FIFO).  A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.   
The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

=========================================================

**Algorithm for Queue Implementation using Array**

=========================================================

**Operations on Queue:**

Mainly the following four basic operations are performed on queue:

**1)Enqueue:**Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.

***Enqueue(Element)***

***-if queue is full then stop***

***-Make space at rear for new element***

***-Store new element and make it the new element***

**if** (**this**.IsFull()) {

System.***out***.println("Queue is full.");

}

queue[++rear] = element;

**if** (front == -1) {

front = 0;

}

=========================================================

**2)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

***Dequeue()***

***-If queue is empty then we stop***

***-Move t the front towards rear.***

**if** (**this**.IsEmpty()) {

System.***out***.println("Queue is empty.");

}

**int** element = queue[front];

queue[front++] = 0;

**return** element;

.

=========================================================

**3)Front:**Get the front item from queue.

**if** (**this**.IsEmpty()) {

System.***out***.println("Queue is empty.");

}

**return** queue[front];

}

=========================================================

**4)Rear:** Get the last item from queue.

**if** (**this**.IsEmpty()) {

System.***out***.println("Queue is empty.");

}

**return** queue[rear];

}

=========================================================

***5)IsEmpty()***

***-If no elements stored in queue then return true***

***Else return false.***

**public** **boolean** IsEmpty() {

**return** (front == -1);

}

=========================================================

***6)IsFull()***

***-If no space left for new element to be stored then return true***

***Else return false***

**public** **boolean** IsFull() {

**return** rear == (size - 1);

}

=========================================================



=========================================================

**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](http://en.wikipedia.org/wiki/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.  
See [this](http://introcs.cs.princeton.edu/43stack/)for more detailed applications of Queue and Stack.

=========================================================

The data structure required for Breadth First Traversal on a graph is?  
a) Stack  
b) Array  
c) Queue  
d) Tree

=========================================================

