RESEARCH PAPER ON QUEUES

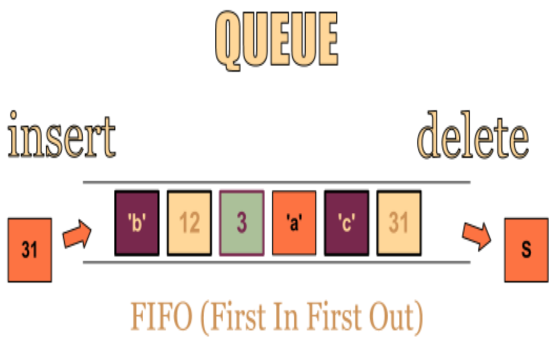
A. Jain, U. Kumar
Computer Science & Information Technology, Dronacharya College of Engineering
Farrukh Nagar, Gurgaon, India

Abstract- **In this research paper we're discussing about the topics based on queues its applications in daily life and its types like queue introduction, priority queue, double ended queue & circular queue and some important topics related to queue like the queue interface, queue implementation, properties of queues e.g. insertion and deletion in a queue(enqueue and dequeue),applications of queue in data structure.**

Index Terms- **Types of queues, queue implementation, properties of queue, insertion and deletion in a queue and applications of queues.**

I. INTRODUCTION

A queue is a collection of linearly ordered elements in which elements are added at one end and retrieved at the other end. First element entering the queue is known as FRONT and last element entering the queue is known as REAR. The first item entering the queue is also the first to be retrieved and removed from the queue and this is why a queue is also called a first-in-first-out (FIFO) structure i.e. the item first put into the queue will be the first served, the second item added to the queue will be the second to be served and so on.



II. PRIORITY QUEUE

A priority queue stores objects, and on request releases the object with greatest value.

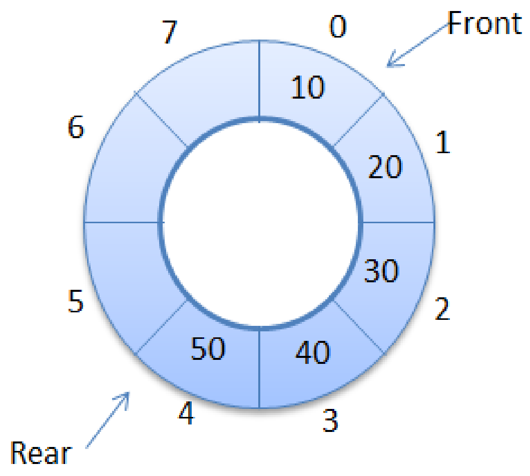
Example: Scheduling jobs in a multi-tasking operating system.
There are two types of priority queue.
• Ascending Priority Queue
• Descending Priority Queue
An ascending priority queue is a collection of items into which items can be inserted arbitrarily and from which only the smallest item can be removed.
A descending priority queue is similar but allows deletion of only the largest item.

Priority queue-unordered array implementation:-

```
public class UnorderedPQ<Item extends Comparable>
{
private Item[] pq;          // pq[i] = ith element on PQ
private int N;              // number of elements on PQ
public UnorderedPQ(int maxN)
{ pq = (Item[]) new Comparable[maxN]; }
public boolean isEmpty()
{ return N == 0; }
public void insert(Item x)
{
pq[N++] = x;
}
public Item delMax()
{
int max = 0;
for (int i = 1; i < N; i++)
if (less(max, i)) max = i;
exch(max, N-1);
return pq[--N];
}}
```

III. DOUBLE ENDED QUEUE

A double-ended priority queue *(DEPQ)* is a collection of zero or more elements. Each element has a priority or value. The operations performed on a double-ended priority queue are:
1. getMin() ... return element with minimum priority
2. getMax() ... return element with maximum priority
3. put(*x*) ... insert the element *x* into the DEPQ
4. removeMin() ... remove an element with minimum priority and return this element
5. removeMax() ... remove an element with maximum priority and return this element.



One application of a DEPQ is to the adaptation of quick sort, which has the the best
expected run time of all known internal sorting methods, to external sorting.

IV. CIRCULAR QUEUE

Although there is space in the following queue, we may not be able to add a new item. An attempt will cause an overflow. A solution of this problem is circular queue.
Imagine a linear queue is wrapped around a cylinder such that the first and last elements of the array are next to each other. When the queue gets apparently full, it can continue to store elements in the empty spaces in the beginning of the array. This makes efficient use of the array slots.

V. QUEUE INTERFACE

A queue is a data structure where we add elements at the back and remove elements from the front. In that way a queue is like "waiting in line": the first one to be added to the queue will be the first one to be removed from the queue. This is also called a FIFO (First In First Out) data structure.

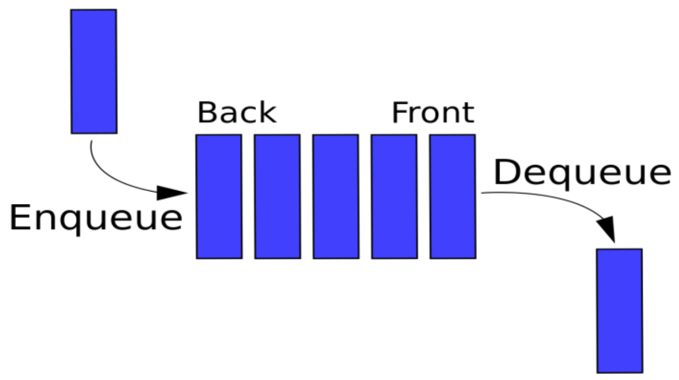
VI. QUEUE IMPLEMENTATION

The array to implement the queue would need two variables (indices) called *front* and *rear* to point to the first and the last elements of the queue. For each enqueue operation *rear* is incremented by one, and for each dequeue operation, front is incremented by one.
While the enqueue and dequeue operations are easy to implement, there is a big disadvantage in this set up. The size of the array needs to be huge, as the number of slots would go on increasing as long as there are items to be added to the list (irrespective of how many items are deleted, as these two are independent operations.

```
struct queue
{
list front;
list back;
};
```

VII. PROPERTIES OF QUEUE

Enqueue and Dequeue-



Enqueue means adding a new item to the rear of the queue and Dequeue refers to removing the front item from queue and returning it.

Enqueue and Dequeue algorithms:
Here are the enqueue and the dequeue algorithms. It is assumed here that the

variables, rear, front and capacity are globally visible.

```
initially,
rear = 0;
front = 0;
void enqueue(int Q[], int d)
{
rearplus1 = (rear+1)% capacity;
        if (rearplus1 == front)
        print “Q full”;
        else
        {
        Q[rear]= d;
        rear = rearplus1;
        } }
void dequeue( int Q[ ], *dd )
{
        if(front == rear)
        {
        *dd = -9999 ;
print “Q empty”
        }
        else
        {
        *dd = Q[front];
        fron            =(front
+1)%capacity;
        } }
```

VIII. APPLICATIONS OF QUEUE

Queues have many applications in computer systems :-

− Handling jobs in a single processor computer
− print spooling
− transmitting information packets in computer networks.
-The queue of processes to be scheduled on the CPU i.e. the process at front is dequeued and processed. New processes are added at the end of the queue.

REFRENCE

[1]http://faculty.mu.edu.sa
[2]http://www.cs.ucf.edu
[3]http://www.bowdoin.edu
[4]http://scanftree.com
[5]http://home.deib.polimi.it
[6]http://www.cs.cmu.edu
[7]http://web.onda.br