Data Structures and Algorithms

QUEUE

PRASHANT HEMRAJANI

Assistant Professor

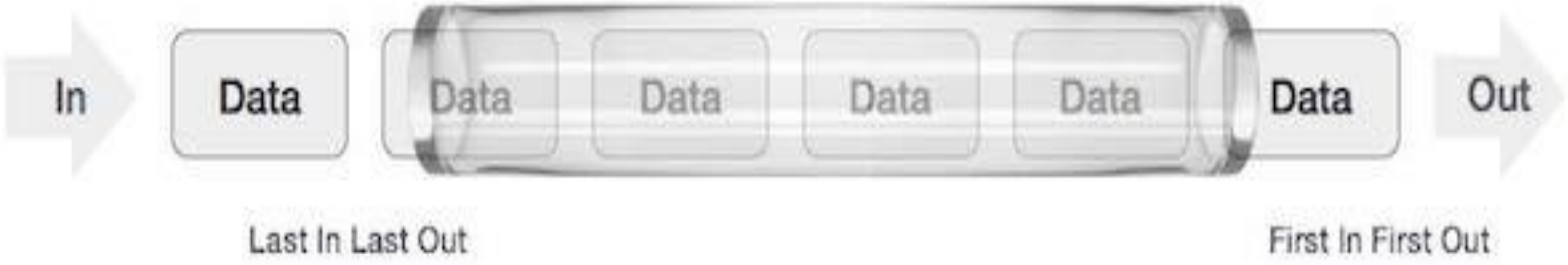
(Computer and Communication Engineering)

Introduction

- A queue is a linear list in which insertions can take place at one end on the list called the rear of the list.
- Deletions can take place only at the other end called the front of the list.
- The behavior of a queue is like a First-In-First-Out (FIFO) system.

Representation

LAST IN
LAST OUT
ONE WAY
FIRST IN
FIRST OUT



Queue

Operations

1. Addition
   - To insert elements in the end of a queue.
2. Deletion
   - To access and remove front element of queue.

Applications

1. Serving requests on a single shared resource, like a printer, CPU task scheduling etc.
2. In real life, Call Center phone systems will use 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, First come first served.
4. Breadth First Search in Tree.

Types of queue

1. Linear Queue
2. Circular Queue
3. Double ended Queue
4. Multi Queue
5. Priority Queue

Insertion in Linear Queue

ADDQ (QUEUE, MAXQ, FRONT, REAR, ITEM)

This procedure inserts an element ITEM into a queue.

```
1.      [Check overflow condition. ]
        If REAR= MAXQ, then: Write: OVERFLOW and Exit.
2.      [Increment REAR]
        If FRONT : = 0, then: [Queue initially empty.]
                Set FRONT= 1 and REAR= 1.
        Else:
                Set REAR=REAR +1
        [End of If structure]
3.      Set QUEUE [REAR]= ITEM. [This inserts new element.]
4.      Exit
```

Deletion in Linear Queue

DELQ (QUEUE, FRONT, REAR, ITEM)

This procedure deletes an element from a queue and assigns it to the variable ITEM.

```
1.      [Queue already empty?]
        If FRONT=0, then: Write: UNDERFLOW, and Exit.
2.      [Remove an element.]
        Set ITEM= QUEUE [FRONT]
3.      [If Removing the last element.]
                If FRONT= REAR then:
                Set FRONT=0, REAR=0
        Else:
                Set FRONT= FRONT +1
        [End of If Structure]
4.      Exit.
```

Insertion in Linear Queue

(Linked Representation)

ADDQ (FRONT, REAR, AVAIL, LINK, INFO, ITEM)

This procedure adds a new element in the REAR of a QUEUE using the link list.

```
Step I: -  [OVERFLOW?]
           IF AVAIL = NULL, then: Write OVERFLOW and Exit
Step II: - Set NEW: = AVAIL, AVAIL: = LINK [AVAIL]
Step III: - Set INFO [NEW]:= ITEM, LINK [NEW]: = NULL
Step IV: -If REAR: = NULL, then:
                     Set FRONT: =  NEW, REAR: = NEW
Step V: - Else:
                     Set LINK [REAR]:= NEW
                     Set REAR: =  NEW.
           [End of If structure.]
Step VI:- Exit
```

Deletion in Linear Queue

(Linked Representation)

DELQ (FRONT, REAR, LINK, INFO, ITEM)

This procedure removes the FRONT element of a QUEUE using the linked list.

```
Step I: -  [UNDERFLOW?]
           If REAR= NULL, then: Write: UNDERFLOW and Exit.
Step II: - Set PTR:= FRONT.
Step III: -If FRONT=REAR Then
                     Set FRONT:= NULL, REAR: = NULL
Step IV: -Else:
                     Set FRONT: = LINK [FRONT]
           [End of If Structure]
Step V: - Set ITEM: = INFO [PTR]
Step VI: -Set LINK [PTR]=AVAIL
Step VII: - Set AVAIL:= PTR
Step VIII: - Exit.
```

Insertion in Circular Queue

ADDCIRQ(QUEUE, MAXQ, FRONT, REAR, ITEM)

This procedure inserts an element ITEM into circular queue.

```
1.      [Check for the Overflow.]
        If FRONT =1 and REAR=MAXQ, or FRONT=REAR+ 1, then:
                Write: OVERFLOW, and Exit.
        [End of If structure]
2.      [Find new value of REAR.]
        If FRONT=0, then: [Queue initially empty.]
                Set FRONT: = 1 and REAR=1
        Else if REAR=MAXQ, then:
                Set REAR=1.
        Else
                Set REAR=REAR+ 1.
        [End of if structure.]
3.      Set QUEUE [REAR]=ITEM. [This inserts new element.]
4.      Exit.
```

Deletion in Circular Queue

DELCIRQ (QUEUE, MAXQ, FRONT, REAR, ITEM)

This procedure deletes an element from a circular queue and assigns it to the variable ITEM.

```
1.      [Queue already empty?]
        If FRONT =0, then: Write UNDERFLOW, and Return
2.      Set ITEM=QUEUE [FRONT]
3.      [Find new value of FRONT.]
        If FRONT=REAR, then : [Queue has only one element to remove.]
                Set FRONT=0, and REAR=0.
        Else if FRONT=MAXQ, then.
                Set FRONT=1.
        Else
                Set FRONT=FRONT + 1.
        [End of if structure]
4.      Return
```

Insertion in Double Ended (Left)

LEFTADD_DEQ (QUEUE, MAXQ, LEFT, RIGHT, ITEM)

This procedure inserts an element ITEM from left into a DE Queue.

```
1.      [Check for the Overflow.]
        If LEFT= (RIGHT + 1) or LEFT =1 and RIGHT= MAXQ then:
                Write: OVERFLOW, and Exit.
        [End of If structure]
2.      [Find new value of LEFT.]
        If LEFT =0, then:
                Set LEFT= MAXQ, RIGHT= MAXQ.
        Else If LEFT=1, then:
                Set LEFT= MAXQ.
        Else
                Set LEFT=LEFT- 1.
        [End of if structure]
3.      Set QUEUE [LEFT]=ITEM [This inserts new element.]
4.      Exit.
```

Insertion in Double Ended (Right)

RIGHTADD_DEQ (QUEUE, MAXQ, LEFT, RIGHT, ITEM)

This procedure inserts an element Item from right into a DE queue.

```
1.    [Check for the Overflow.]
      If LEFT= (RIGHT + 1) or LEFT =1 and RIGHT =MAXQ then:
            Write: OVERFLOW, and Exit.
2.    [Find new value of RIGHT.]
      If RIGHT =0 then:
            Set LEFT= 1, RIGHT=1.
      Else If RIGHT=MAXQ, then
            Set RIGHT=1.
      Else
            Set RIGHT=RIGHT+ 1.
      [End of If structure]
3.    Set QUEUE [RIGHT= ITEM. [This inserts new element]
4.    Exit.
```

Deletion in Double Ended (Left)

LEFTDEL_DEQ (QUEUE, MAXQ, LEFT, RIGHT, ITEM)

This procedure deletes and element from a DE queue from left and assigns it to the variable ITEM.

```
1.      [Queue already empty?]
        If LEFT =0, then: Write UNDERFLOW, and Exit.
2.      Set ITEM=QUEUE [LEFT]
3.      [Find new value of LEFT]
        If LEFT=RIGHT, then:
                Set RIGHT=0 and LEFT=0.
        Else if LEFT=MAXQ, then:
                Set LEFT=1.
        Else
                Set LEFT=Left + 1.
        [End of if structure.]
4.      Exit.
```

Deletion in Double Ended (Right)

RIGHTDEL_DEQ (QUEUE, MAXQ, LKEFT, RIGHT, ITEM)

This procedure deletes an element from a DE queue from right and assigns it to the variable ITEM.

```
1.      [Queue already empty?]
        If LEFT =0, then: Write UNDERFLOW, and Exit.
2.      Set ITEM= QUEUE [RIGHT]
3.      [Find new value of RIGHT]
        If RIGHT=LEFT, then:
                Set RIGHT=0 and LEFT=0.
        Else If RIGHT=1, then:
                Set RIGHT=MAXQ.
        Else
                Set RIGHT=RIGHT-1
        [End of if structure]
4.      Exit.
```

Insertion in Priority Queue

ADDPQ (PQUEUE, MAXQ, FRONT, REAR, PRIORITY, ITEM, MAXP)

This procedure inserts an element in a priority queue.

```
1.    If PRIORITY > MAXP, then: Write “Such a priority queue does not exist”.
      And Return [Queue already full?]
2.    If FRONT [PRIORITY]= (REAR [PRIORITY] + 1) then Write.
      “This priority queue is full” and Exit.
3.    [Find new value of REAR.]
      If FRONT [PRIORITY] =0, then:
            Set FRONT [PRIORITY]= 1, REAR [PRIORITY]=1
      Else if REAR [PRIORITY]= MAXQ, then,
            Set REAR [PRIORITY]=1
      Else :
            Set REAR [PRIORITY]= REAR [PRIORITY] + 1
      [End of if structure.]
4.    Set PQUEUE [PRIORITY][REAR [PRIORITY]]=ITEM.
5.    Exit
```

Deletion in Priority Queue

DELPQ (PQUEUE, MAXQ, FRONT, REAR, ITEM, N)

This procedure deletes and element in a priority queue.

```
1.      [Finding the smallest queue which is not empty?]
        Set PRIORITY= 1.
        Repeat steps while REAR[PRIORITY]=0 AND PRIORITY< =N
                Set PRIORITY=PRIORITY + 1.
        [End of loop]
2.      If PRIORITY > N then Write: “Priority queue is empty” and Exit.
3.      Set ITEM= PQUEUE [PRIORITY][FRONT[PRIORITY]]
4.      [Find new value of FRONT and REAR]
        If FRONT [PRIORITY] = REAR [PRIORITY], then:
                Set FRONT [PRIORITY]=0, REAR [PRIORITY]=0.
        Else If FRONT [PRIORITY]= MAXQ:
                Set FRONT [PRIORITY] =1.
        Else:
                Set FRONT [PRIORITY] = Set FRONT [PRIORITY] + 1.
        [End of If structure.]
5.      Exit.
```

Any Queries
????