Queues

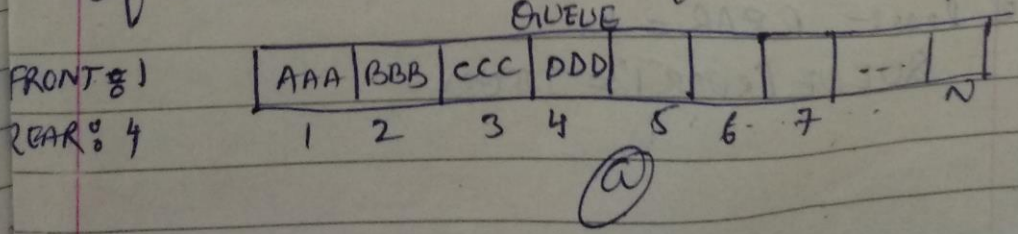
A queue is a linear list of elements in which deletions can take place only at one end called a front and insertions can take place only at the other end called the rear.

NOTE: The terms front and rear are used while describing queue in a linked list. Queues are also called FIFO lists since the first element in a queue will be the first element out of the queue.

Representation of Queues

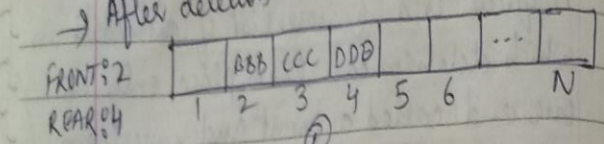
@ Representation of a Queue as an array

QUEUE → linear array
FRONT → ptr var. containing location of the front element of the queue.
REAR → ptr. var. containing the location of rear element of the queue.
If FRONT = NULL → queue is empty.



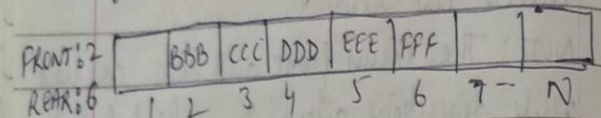
@

→ After deletion

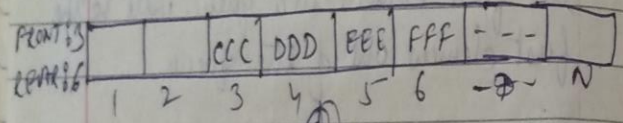


(b)

→ After inserting two elements



After deletion (c)



(d)

Fig: Array Representation of a Queue

Special Case

Suppose, we want to insert an ITEM into a queue at the time the queue does occupy its last part of the array i.e REAR = N. Instead of moving the FRONT to 1 & updating REAR we assume that queue is a circular queue as earlier which is an expensive one.

We reset REAR = 1

QUEUE [REAR] := ITEM

IIly If FRONT = N & an elements is deleted Reset FRONT := 1

If queue contains one element

FRONT = REAR ≠ NULL

If all the elements are deleted

FRONT = REAR = NULL

Procedure 1: QINSERT (QUEUE, N, FRONT, REAR, ITEM)

This procedure inserts an element ITEM into a queue

1 [Queue already filled?]
If FRONT = 1 and REAR = N or If FRONT = REAR + 1
then write "OVERFLOW" & Return
2. [Find new value of REAR]
If FRONT := NULL then [Queue initially empty]
Set FRONT := 1 and REAR := 1
Else if REAR = N then
Set REAR = 1
Else
Set REAR := REAR + 1
[End of If structure]
3. Set QUEUE [REAR] := ITEM
4 Return

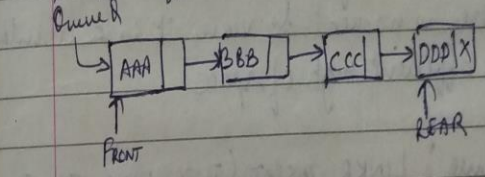
PROCEDURE 2 : QDELETE (QUEUE, N, FRONT, REAR, ITEM)

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

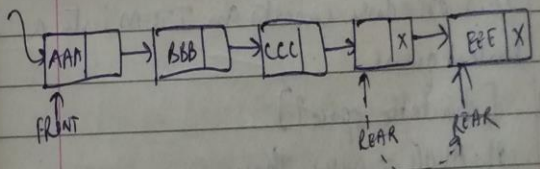
1. [Queue already empty?]
   If FRONT = NULL then
   Write "UNDERFLOW" and Return.
2. Set ITEM := QUEUE[FRONT]
3. [Find new value of FRONT]
   If FRONT = REAR then
   Set FRONT := NULL & REAR := NULL
   Else if FRONT = N then
   Set FRONT := 1
   Else
   Set FRONT := FRONT + 1
   [End of if structure]
4. Return

(b) LINKED REPRESENTATION OF QUEUES

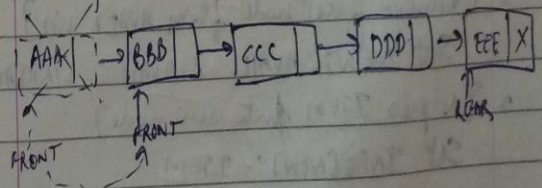
A linked queue is a queue implemented as a linked list with two ptr. variables FRONT and REAR pointing to the nodes which is in the FRONT & REAR of the queue. The INFO fields of the list hold the elements of the queue & the LINK fields hold pointers to the neighbouring elements in the queue.



Insert EEE into Queue :-



Delete from Queue



Comparison of array representation of a queue with linked queue

(1) Array representation of a queue suffers from the drawback of limited queue capacity whereas linked queue is not limited in capacity.

(2) Data movement is expensive whereas linked queue functions as a linear queue & there is no need to view it as a circular for efficient mgt. of space.

Procedure: LINKQ_INSERT (INFO, LINK, FRONT, REAR, AVAIL, ITEM)

This procedure inserts an ITEM into a linked queue.

1. [Available space?]
   If AVAIL = NULL then
   Write "OVERFLOW"
   Exit.
2. [Remove first node from AVAIL list]
   Set NEW := AVAIL & AVAIL := LINK[AVAIL]
3. [Copies ITEM into new node]
   Set INFO[NEW] := ITEM
   LINK[NEW] := NULL
4. If (FRONT = NULL) then FRONT = REAR = NEW
   else
   set LINK[REAR] := NEW & REAR := NEW
5. Exit

Procedure: LINKQ_DELETE (INFO, LINK, FRONT, REAR, AVAIL, ITEM)

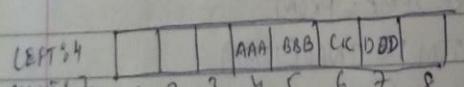
This procedure deletes front element of the linked queue & stores it in ITEM.

1. [Linked queue Empty?]
   If FRONT = NULL then
   Write "UNDERFLOW"
   Exit
2. Set TEMP = FRONT
3. ITEM := INFO[TEMP]
4. FRONT := LINK[TEMP]
5. LINK[TEMP] = AVAIL & AVAIL = TEMP
6. Exit

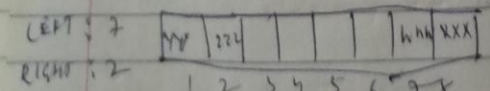
DEQUES (Double ended Queue)

It is a linear list in which elements can be added or removed at either end but not in the middle.

NOTE: These are various ways of representing a deque in a computer. Unless it is otherwise stated or implied, we will assume our deque is maintained by a circular array DEQUE with ptrs LEFT & RIGHT.
Circular means DEQUE[1] comes after DEQUE[N] in the array.
Ex:- 4 elements N=8



(a)



Types of variations of DEQUE:

(1) Input-restricted deque: It is a deque which allows insertions at only one end of the list but deletions at both ends of the list.

(2) Output-restricted deque: It is a deque which allows deletions at only one end of the list but allows insertions at both ends of the list.

Example

Consider the following deque of characters where DEQUE is a circular array which is allocated six memory cells
LEFT = 2, RIGHT = 4.
DEQUE: _, A, C, D, _, _

Describe the deque while the following operations take place:

(a) F is added to the right of deque.

Sol:-

LEFT=2, RIGHT=5, DEQUE: -, A, C, D, F, -

(b) Two letters on the right are deleted

LEFT=2, RIGHT=3, DEQUE: -, A, C, -, -, -

(c) K, L, M are added to the left of the deque.

LEFT=5, RIGHT=3

DEQUE: K, A, C, -, -, M, L

(d) One letter on the left is deleted

LEFT=6, RIGHT=3

DEQUE: K, A, C, -, -, -, L

(e) R is added to the left of the deque

LEFT=5, RIGHT=3

DEQUE: K, A, C, -, -, R, L

(f) S is added to the right of the deque

LEFT=5, RIGHT=4

DEQUE: K, A, C, S, R, L

(g) T is added to the right of the deque. Since LEFT = RIGHT+1, the array is full & hence T cannot be added to the deque i.e. overflow has occurred.

Example (6.26)

A linked queue Q and an AVAIL list is maintained as a linked stack as shown in fig. Trace the contents of the memory after the execution of the following operations on a linked queue Q.

	INFO	LINK
23	56	NULL
24	8	29
25	12	34
26	5	NULL
27	76	30
28	123	31
29	09	33
30	45	23
31	23	26
32	56	25
33	78	24
34	123	24

After operations Updated

	INFO	LINK
23	56	32
24	8	29
25	12	34
26	5	NULL
27	76	26
28	123	31
29	09	33
30	67	NULL
31	23	26
32	567	30
33	78	24
34	123	24

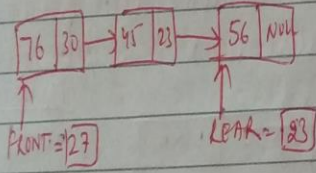
AVAIL := 32

LINKED QUEUE Q : FRONT : 27 , REAR : 23

① Insert 567
② Delete
③ Delete
④ Insert 67

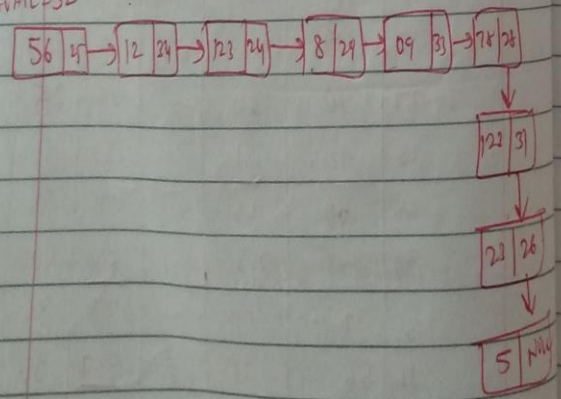
Sol

Linked Queue



AVAIL LIST :

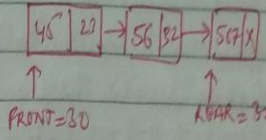
AVAIL = 32



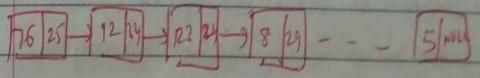
After

(I) & (II)

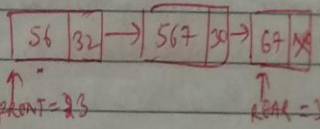
Q



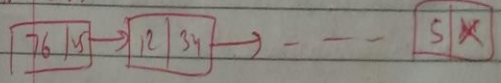
AVAIL



(III) & (IV)



AVAIL = 27



PRIORITY QUEUES

A priority queue is a collection of elements such that each element has been assigned a priority and such that order in which elements are deleted & processed comes from the following rules:

(1) An element of higher priority is processed before any element of lower priority.

(2) Two elements with the same priority are processed according to the order in which they were added to the queue.

A prototype of a priority queue is a timesharing system.

ONE WAY LIST REPRESENTATION OF A PRIORITY QUEUE

Priority queue can be maintained in memory by means of one-way list as follows:

(a) Each node in the list contains three items of info: an info field INFO, a priority no PRN & a link no. LINK

(b) A node X precedes a node Y in the list (i) when X has higher priority than Y or (ii) when both have the same priority but X was added to list before Y.

This means that the order in the one-way list corresponds to the order of the priority queue

Algorithm: To delete and process the first element in a priority queue which appears in memory as a one-way list.

1. Set ITEM := INFO[START]
2. Delete first node from the list
3. Process ITEM
4. Exit

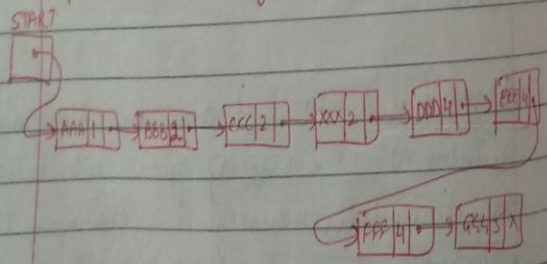
Algorithm: To add an ITEM with priority number N to a priority queue which is maintained in memory as a one-way list.

(a) Traverse the one-way list until finding a node X whose priority no exceeds N. Insert ITEM in front of node X.

(b) If no such node is found, insert ITEM as the last element of the list.

ARRAY REPRESENTATION OF A PRIORITY QUEUE

Another way to maintain a priority queue in memory is to use a separate queue for each level of priority.
Each such queue will appear in its own circular array and must have its own pair of pointers, FRONT & REAR. If each queue is allocated the same amount of space, a 2-D array QUEUE can be used instead of the linear array. Suppose the priority queue is as follows



Step1 Observe the FRONT[K] & REAR[K] of K^th row.

	Given FRONT	REAR
1	2	2
2	1	3
3	0	0
4	5	1
5	4	4

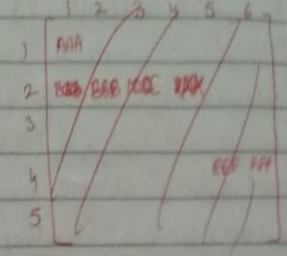


Fig 1

Example

Consider the priority queue in Fig 1 which is maintained by a 2-D array QUEUE

(a) Describe the structure after (RRR, 3), (SSS, 4), (TTT, 1), (UUU, 4) and (VVV, 2) are added to the queue.

(b) Describe the structure of, after the preceding insertions, three elements are deleted.

Sol

(a)

	FRONT	REAR
1	2	3
2	1	4
3	1	1
4	5	3
5	4	4

	1	2	3	4	5	6
1		AAA	TTT			
2	BBB	CCC	XXX	VVV		
3	RRR					
4	FFF	SSS	UUU		DDD	EEE
5				GGG		

(b)

	FRONT	REAR
1	0	0
2	2	4
3	1	1
4	5	3
5	4	4

	1	2	3	4	5	6
1						
2		CCC	XXX	VVV		
3	RRR					
4	FFF	SSS	UUU		DDD	EEE
5				GGG		

Applications of Queues

Simulation : It is the use of one system to imitate the behavior of another system. Simulation are often used when it would be too expensive or dangerous to experiment with the real system.

- Physical Simulation such as wind tunnels used to experiment with designs for car bodies & flight simulators used to train airline pilots.
- Mathematical simulations are systems of equations used to describe some systems
- Computer simulations use the steps of a program to imitate the behaviour of the system under study

In Comp. simulation

- objects being studied are represented as DS
- Actions " " are " " as operations and
- Rules describing these actions are translated into computer algorithms.

Eg: Simulation of an airport

Airport with only one runway

Two queues of landing & takeoff

landing queue has higher priority than takeoff.