Queues

Introduction

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

NOTE: The terms front and rear are used while describing queues 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 Queue as an array

- QUEUE → Linear Array
- FRONT → Pointer variable contain location of the front element of the queue.
- REAR → Pointer variable containing the location if rear element of the queue.
- If FRONT = NULL i.e. queue is empty

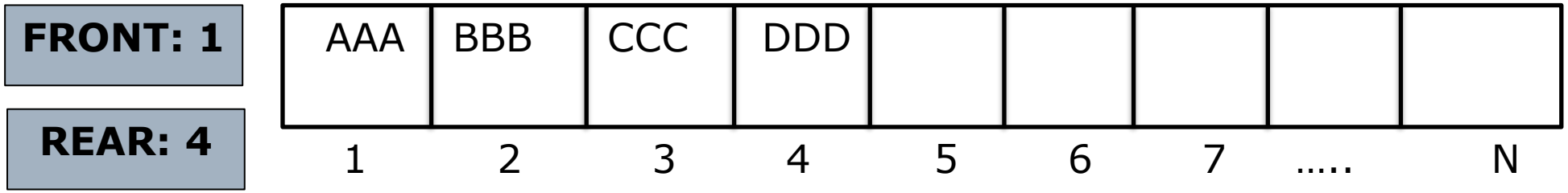


Figure (a): Queue

Representation of Queue as an array

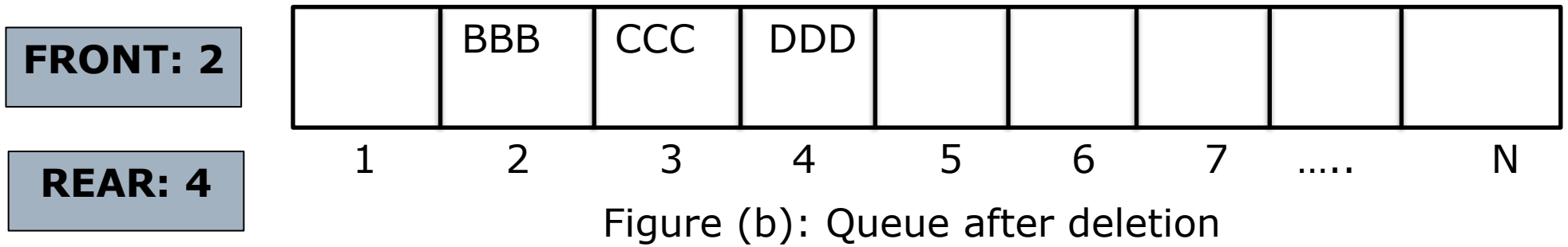


Figure (b): Queue after deletion

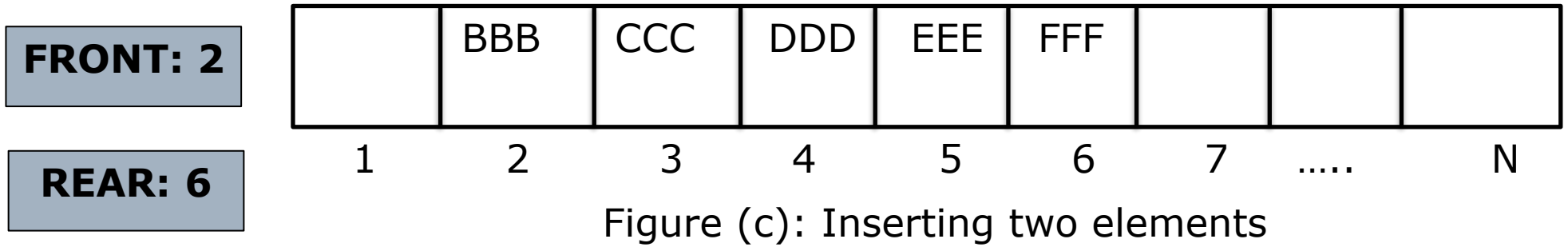


Figure (c): Inserting two elements

Representation of Queue as an array

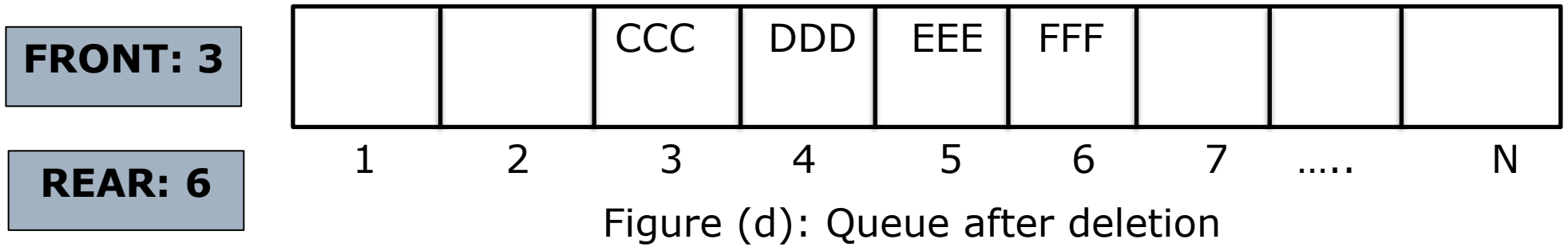


Figure (d): Queue after deletion

Special Case

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

  QUEUE[REAR]=ITEM

Special Case

- Similarly If FRONT=N and an element is deleted. Reset FRONT:=1
- If queue contains one element FRONT = REAR ≠ NULL
- If all the elements are deleted FRONT = REAR = NULL

Queue

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

This procedure inserts an element ITEM into a queue.

```
Step 1: [Queue already filled? ]
        if FRONT=1 and REAR=N or If FRONT=REAR+1, then
                Print “OVERFLOW”
                Return
Step 2: [Find new value of REAR]
         If FRONT=NULL the
                Set FRONT:=1 and REAR:=1
        else if REAR=N then
                 Set REAR:=1
        else
                Set REAR:=REAR+1
        [End of If structure]
Step 3: QUEUE[REAR]:=ITEM
Step 4: Return
```

Queue

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

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

```
Step 1: [Queue already empty? ]
        if FRONT=NULL, then
                Print “UNDERFLOW”
                Return
Step 2: Set ITEM:=QUEUE[FRONT]
Step 3: [Find new value of FRONT]
         If FRONT=REAR the
                Set FRONT:=NULL and REAR:=NULL
        else if FRONT=N then
                 Set FRONT:=1
        else
                Set FRONT:=FRONT+1
        [End of If structure]
```

Linked Representation of Queues

- A linked queue is a queue implemented as a linked list with two pointer variables.
- FRONT and REAR pointing to the nodes which is in the front and rear of the queue.
- The INFO field holds the elements of the queue and the LINK field hold the pointer to the neighboring elements in the queue.

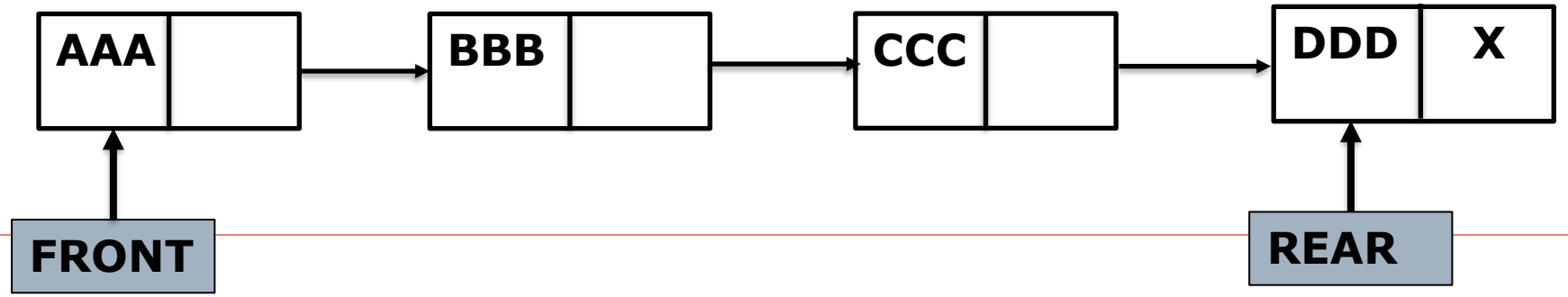
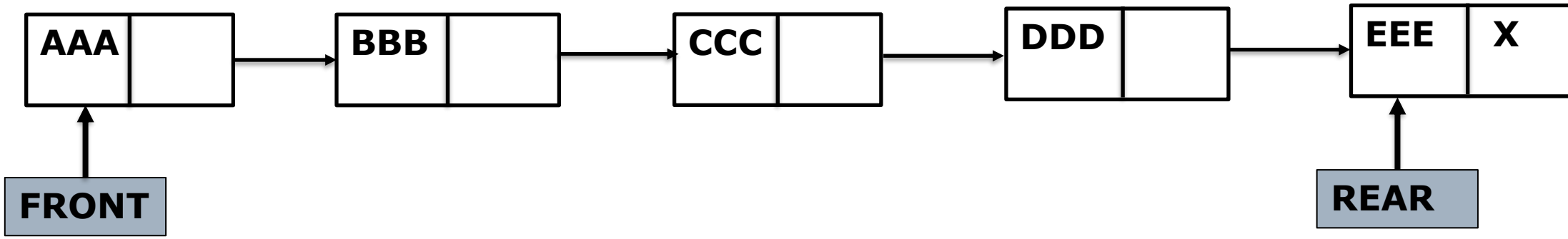


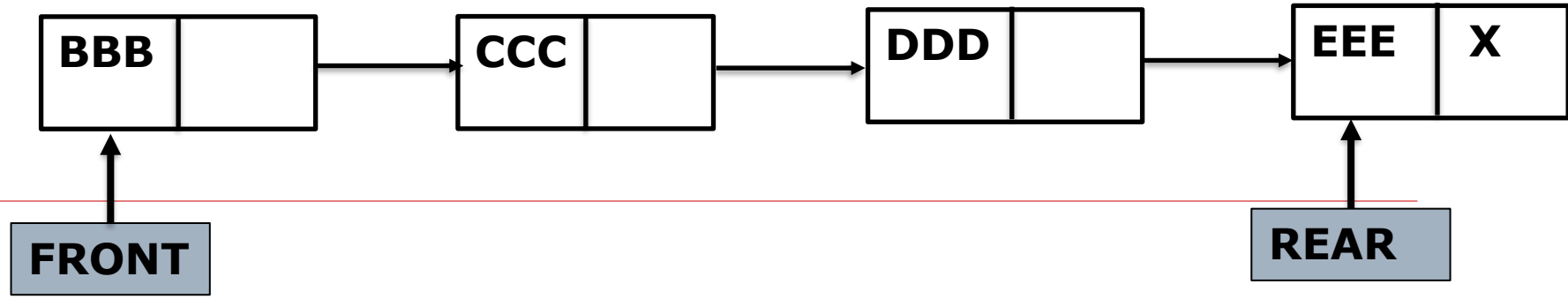
Fig.: Queue Q

Linked Representation of Queues

- Insert EEE into Queue Q:



- Delete from Queue Q:



Comparison of array representation of a queue with linked queue

- Array representation of a queue suffers from the drawback of limited queue capacity whereas linked queue is not limited in capacity.
- Data movement is expressive whereas linked queue functions as a linear queue and there is no need to view it as a circular for efficient management of space.

Linked Queue

Procedure 3: LINKQ_INSERT(INFO, LINK, FRONT, REAR, AVAIL, ITEM)

This procedure inserts an ITEM into a linked queue.

```
Step 1: [Available Space? ]
        if AVAIL=NULL, then
                Print "OVERFLOW"
                Return
Step 2:[Remove first node from AVAIL list]
        Set NEW:=AVAIL and AVAIL:=LINK[AVAIL]
Step 3:[Copies ITEM into new node]
        Set INFO[NEW]:=ITEM
             LINK[NEW]:=NULL
Step 4: [If Queue is empty]
        If (FRONT=NULL) then
                FRONT=REAR=NEW
        else
                Set LINK[REAR]:=NEW and REAR:=NEW
Step 5: Return
```

Linked Queue

Procedure 4: LINKQ_DELETE(INFO, LINK, FRONT, REAR, AVAIL, ITEM)

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

Step 1: [Linked Queue empty?]
    if FRONT=NULL, then
        Print “UNDERFLOW”
        Return
Step 2: Set TEMP:=FRONT
Step 3: ITEM:=INFO[TEMP]
Step 4: FRONT:=LINK[TEMP]
Step 5: LINK[TEMP]:=AVAIL and AVAIL:=TEMP
Step 5: Return

DEQUES
(Double Ended Queues)

DEQUES

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

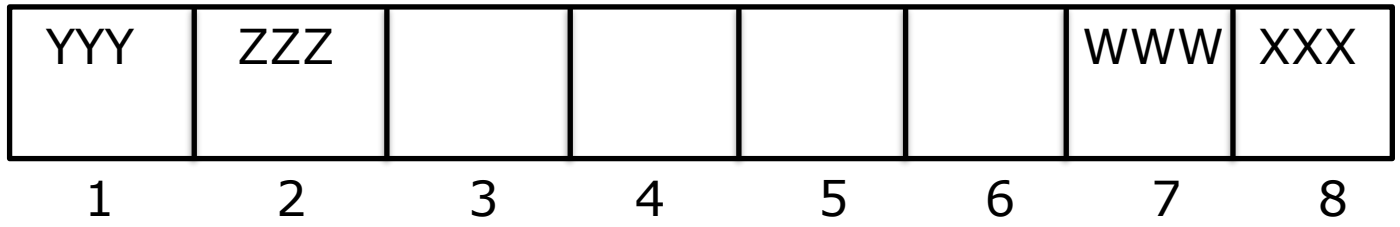
NOTE:

- There are various ways of representing a deque in a computer. Assume our deque is maintained by a circular array DEQUE with the pointers LEFT and RIGHT.
- Circular means DEQUE[1] comes after DEQUE[N] in the array.
- Example: N=8

DEQUES

- Example: N=8

LEFT: 7
RIGHT: 2



Types of variations of DEQUE

1. **Input-Restricted Deque:** It is a deque which allows insertions at only one end of the list but deletion at both ends of the list.
2. **Output-Restricted Deque:** It is a deque which allows deletion at only one end of the list but insertion at both ends of the list.

DEQUE 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 the deque

(b) Two letter on the right are deleted

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

(d) One letter on the left is deleted

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

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

(g) T is added to the right of the deque.

DEQUE Example

Solution:

LEFT=2, RIGHT=4

DEQUE: __, A, C, D, __, __

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

DEQUE: __, A, C, D, F, __

LEFT=2, RIGHT=5

(b) Two letter on the right are deleted

DEQUE: __, A, C, __, __, __

LEFT=2, RIGHT=3

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

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

LEFT=5, RIGHT=3

DEQUE Example

Solution:

(d) One letter on the left is deleted
DEQUE: K, A, C, __, __, L
LEFT=6, RIGHT=3
(e) R is added to the left of the deque
DEQUE: K, A, C, __, R, L
LEFT=5, RIGHT=3
(f) S is added to the right of the deque
DEQUE: K, A, C, S, M, L
LEFT=5, RIGHT=4
(g)T is added to the right of the deque.
Since LEFT=RIGHT+1, the array is full and hence T can not be added to the deque i.e. overflow has occurred.

Priority Queue

A priority queue is a collection of elements such that order in which elements are deleted and 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 are added to the queue.

A prototype a priority queue is a timesharing system.

One Way List Representationof a Priority Queue

A priority queue can be maintained in memory by means one-way list as follows:

(a) Each node in the list contain three items of information: An info field INFO, a priority no. PRN and 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.

One Way List Representationof a Priority Queue

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

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

One Way List Representation of a Priority Queue

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.

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.

Types of Simulation:

- **Physical Simulation**
- **Mathematical simulation**
- **Computer Simulation**

Applications of Queues

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

In computer simulation

-Objects being studied are represented as Data Structures.

-Actions being studies are represented as operations on data structures.

-Rules describing these actions are translated into computer algorithms.

Example: Simulation of an airport

Airport wit only one runway

Two queues of landing and takeoff. Landing queue has higher priority than takeoff.