SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : 19ITT201- DATA STRUCTURES

II YEAR / III SEMESTER

Unit 1- LINEAR STRUCTURES AND TREES

Topic 8 : Queue Based Implementation

Problem

- The Queue occurs when there are more customers than employees to serve them. This means that customers have to wait for their turn. Whether the waiting itself is an issue or not can only be determined by the customers.

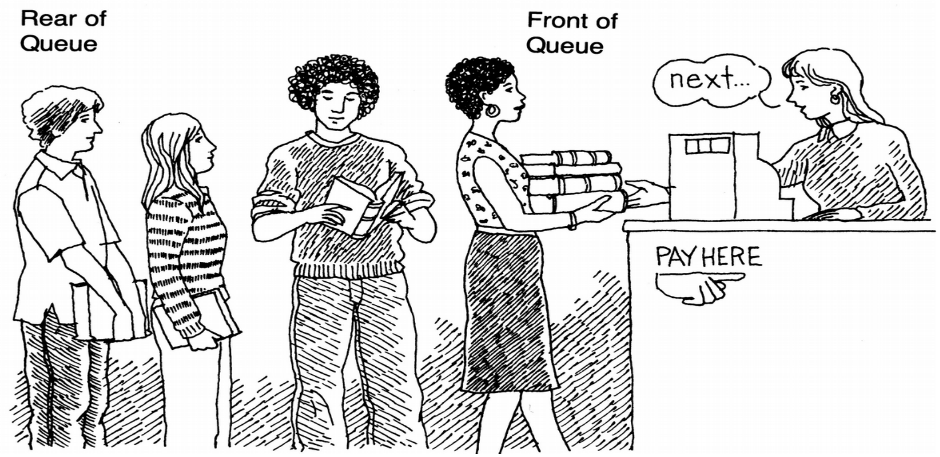
- In 1985, an American writer and expert on business management named David Maister published a seminal paper titled the Psychology of Waiting Lines. Here he outlines a simple formula to measure customer satisfaction.
- S = P – E

S = satisfaction

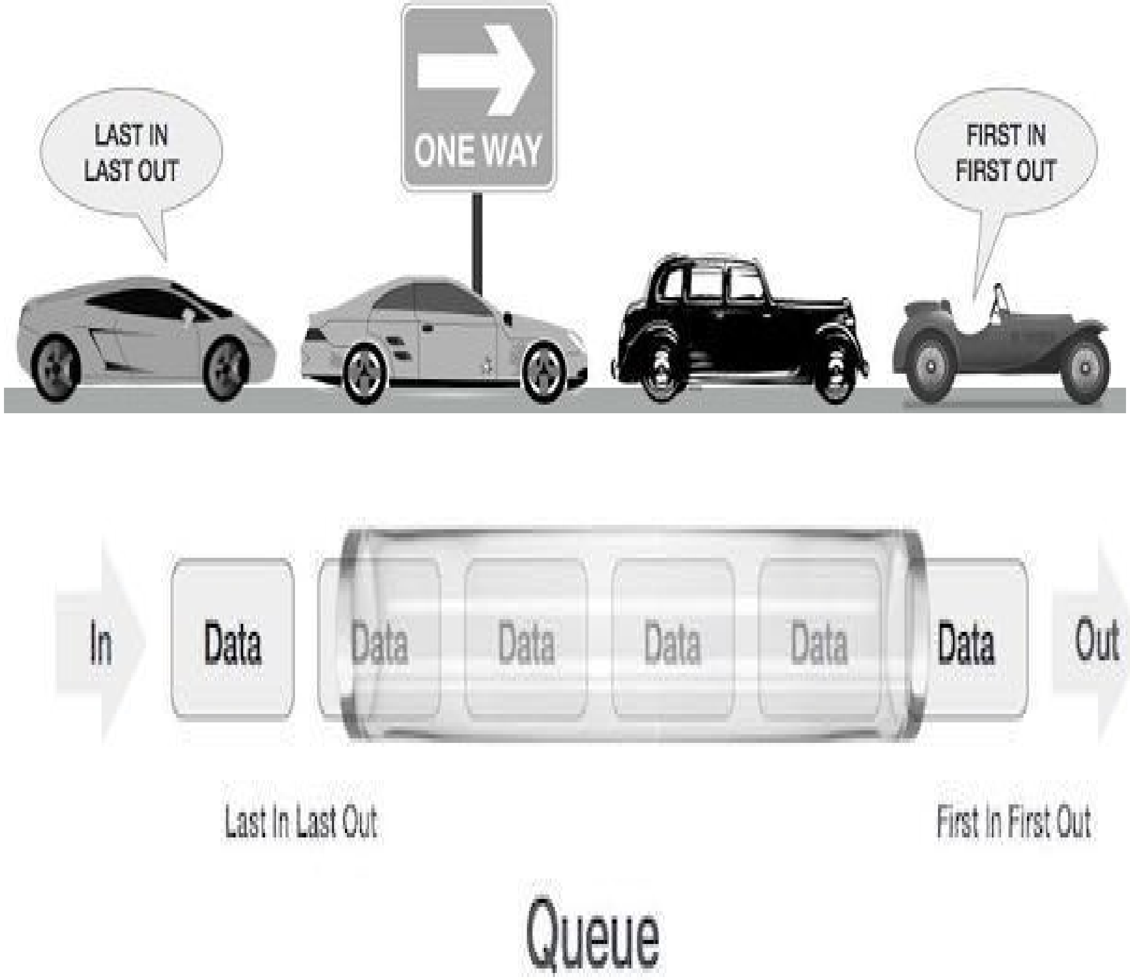
- P = perception of the service level
- E = expectation of the service level

Queue Based Implementation

- What is a Queue?
- It is an ordered group of homogeneous items of elements.
- Queues have two ends:
  - Elements are added at one end.
  - Elements are removed from the other end.
- The element added first is also removed first (**FIFO**: First In, First Out).



Queue Based Implementation –Cont..



Queue Based Implementation - Cont..

Basic Operations

- Queue operations may involve initializing or defining the queue, utilizing it, and then completely erasing it from the memory.
- Here we shall try to understand the basic operations associated with queues −
- **enqueue()** − add (store) an item to the queue.
- **dequeue()** − remove (access) an item from the queue.
- Few more functions are required to make the above-mentioned queue operation efficient. These are −
- **peek()** − Gets the element at the front of the queue without removing it.
- **isfull()** − Checks if the queue is full.

Queue Based Implementation - Cont..

➢Queue operations work as follows:

- Two pointers called FRONT and REAR are used to keep track of the first and last elements in the queue.
- When initializing the queue, we set the value of FRONT and REAR to -1.
- On enqueing an element, we increase the value of REAR index and place the new element in the position pointed to by REAR.
- On dequeueing an element, we return the value pointed to by FRONT and increase the FRONT index.
- Before enqueing, we check if queue is already full.
- Before dequeuing, we check if queue is already empty.
- When enqueing the first element, we set the value of FRONT to 0.
- When dequeing the last element, we reset the values of FRONT and REAR to -1.

Queue Based Implementation - Cont..

Enqueue Operation

- Queues maintain two data pointers, **front** and **rear**. Therefore, its operations are comparatively difficult to implement than that of stacks.
- The following steps should be taken to enqueue (insert) data into a
queue −
- **Step 1** − Check if the queue is full.
- **Step 2** − If the queue is full, produce overflow error and exit.
- **Step 3** − If the queue is not full, increment **rear** pointer to point the next empty space.
- **Step 4** − Add data element to the queue location, where the rear is
pointing.
- **Step 5** − return success.

Queue Based Implementation - Cont..

A structure to represent a queue

```
struct Queue {
    int front, rear, size;
    unsigned capacity;
    int* array;
};
```

- Algorithm for enqueue operation
- procedure enqueue(data)
- if queue is full
- return overflow
- endif
- rear ← rear + 1 queue[rear] ← data return

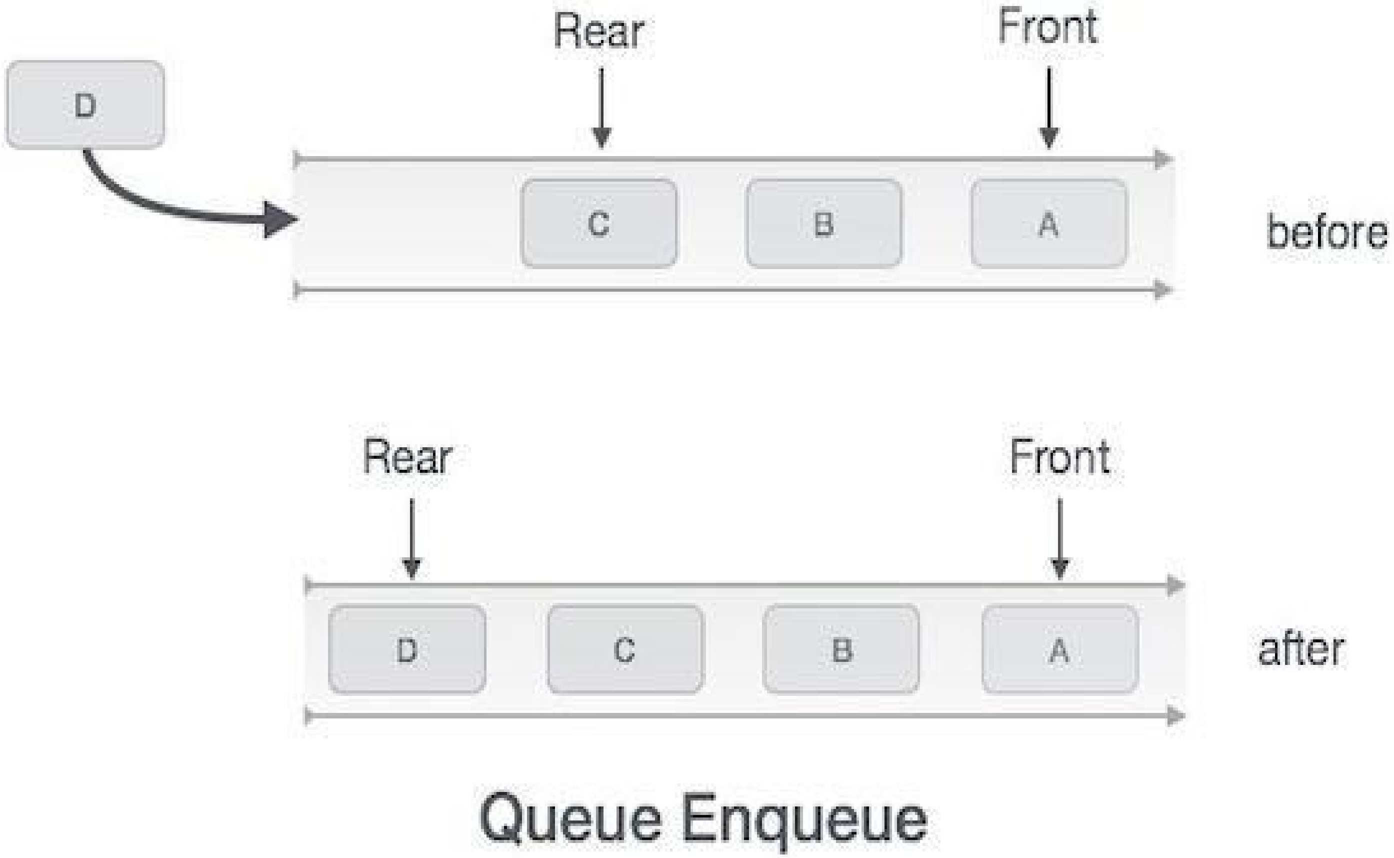
true

- end procedure

Example – enqueue operation

- int enqueue(int data)
- if(isfull())
- return 0;
- rear = rear + 1;
- queue[rear] = data;
- return 1;
- end procedure

Queue Based Implementation - Cont..



Queue Based Implementation - Cont..

Dequeue Operation

- Accessing data from the queue is a process of two tasks − access the data where **front** is pointing and remove the data after access. The following steps are taken to perform **dequeue** operation −
- **Step 1** − Check if the queue is empty.
- **Step 2** − If the queue is empty, produce underflow error and exit.
- **Step 3** − If the queue is not empty, access the data where **front** is pointing.
- **Step 4** − Increment **front** pointer to point to the next available data element.
- **Step 5** − Return success.

Queue Based Implementation - Cont..

Algorithm for dequeue operation

- procedure dequeue
- if queue is empty
- return underflow
- end if data = queue[front] front ← front + 1

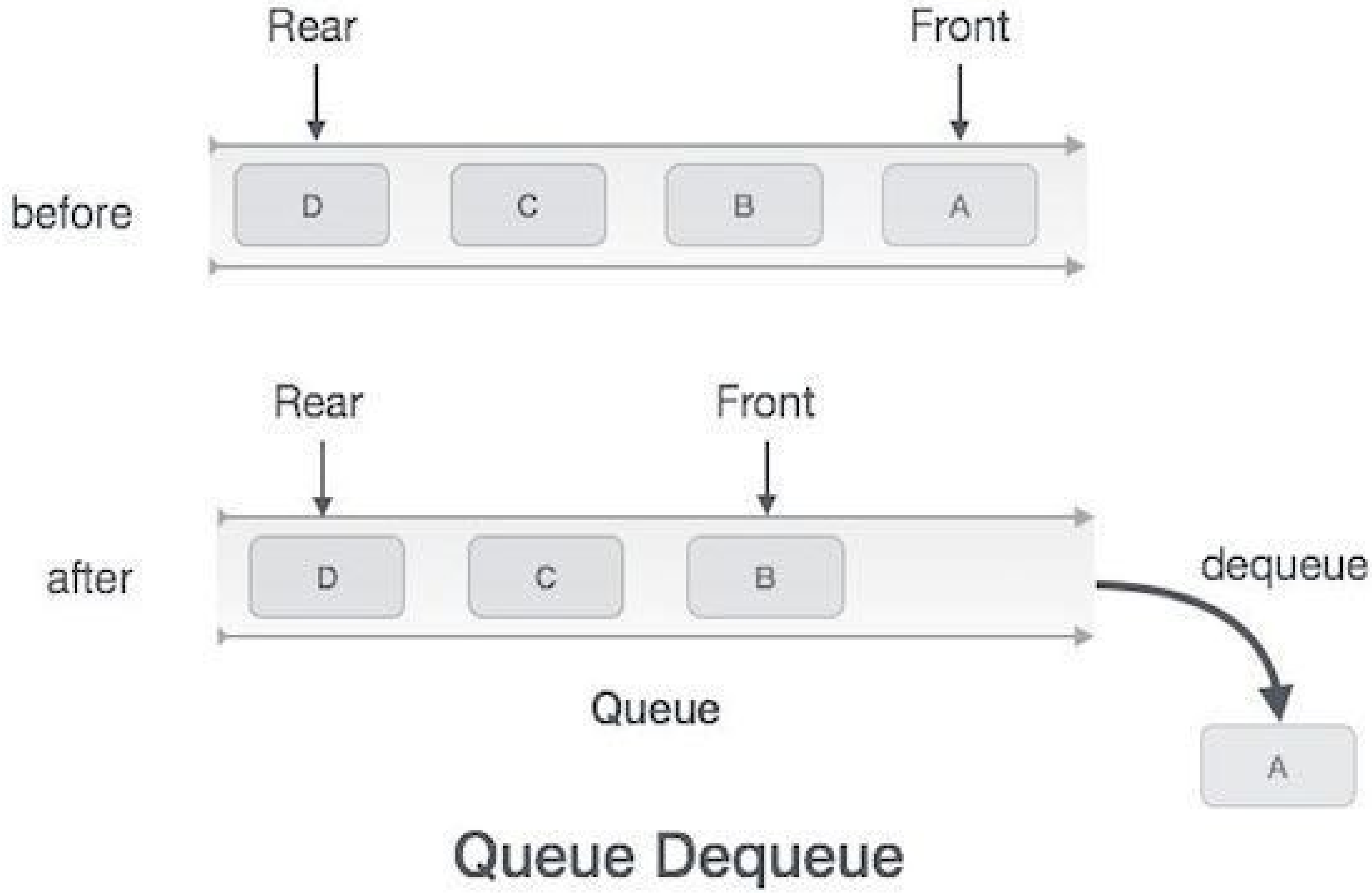
return true

- end procedure

Dequeue operation

- int dequeue()
- {
- if(isempty())
- return 0;
- int data = queue[front];
- front = front + 1;
- return data;
- }

Queue Based Implementation –Cont..



Queue Based Implementation –Cont..

- Let's first learn about supportive functions of a queue
- peek()
- This function helps to see the data at the **front** of the queue.
- The algorithm of peek() function is as follows −
- **Algorithm**

1. begin procedure peek
2. return queue[front]
3. end procedure

Queue Based Implementation –Cont..

- Implementation of peek() function in C programming language
- **Example**
- int peek()
- {
- return queue[front];
- }

Queue Based Implementation –Cont..

isfull()

- **Algorithm**
- begin procedure
- isfull
- if rear equals to MAXSIZE
- return true
- else return false
- endif
- end procedure

- bool isfull()
- {
- if(rear == MAXSIZE - 1)
- return true;
- else
- return false;
- }

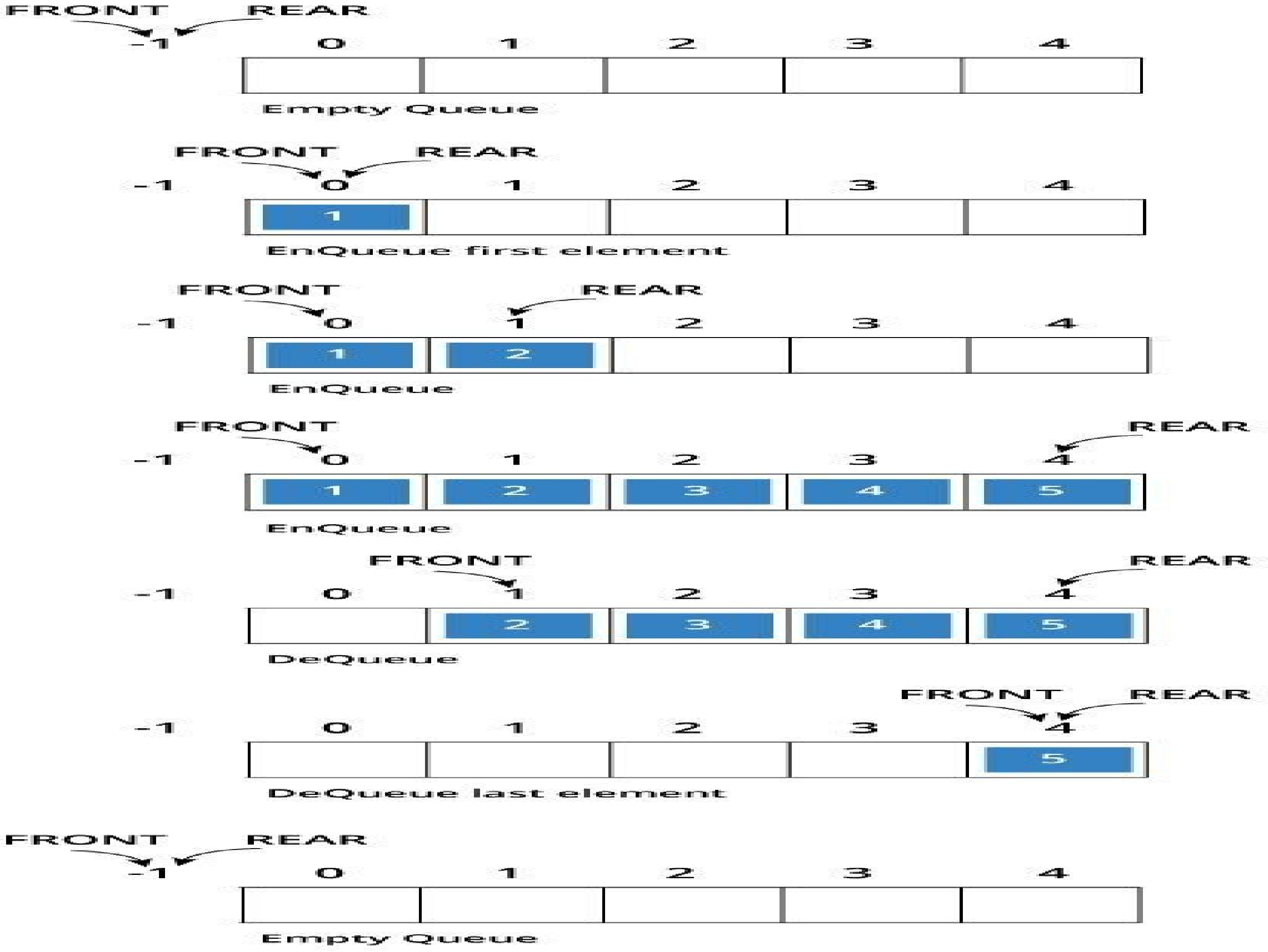
Queue Based Implementation –Cont..

isempty()

- isempty()
- Algorithm of isempty() function
- **Algorithm**
- begin procedure
- isempty
- if front is less than MIN OR front is greater than
- rear return true
- else return false
- Endif
- end procedure

- **Example**
- bool isempty()
- {
- if(front < 0 || front > rear)
- return true;
- Else
- return false;
- }

Queue Based Implementation –Cont..



Activity

MCQ

1. Which of the following properties is associated with a queue?
   a) First In Last Out
   b) First In First Out
   c) Last In First Out
   d) None of the mentioned
2. What is the term for inserting into a full queue known as?
   a) overflow
   b) underflow
   c) null pointer exception
   d) all of the mentioned

Advantages

1. Ease of supervision/operation by skilled workers
2. High utilization of expensive machines
3. Flexibility – broad scope of products

Disadvantages

1. High WIP and long flow times
2. Conflict between resource utilization and customer service
3. Difficult to meet promise dates
4. Cost of variety: reduced learning, difficult scheduling

Assessment 1

1. List out the advantages of queue based implementation

   a)________________________________

   b)________________________________

   c)________________________________

   d)________________________________

2. Identify the disadvantages of queue based implementation

   a)________________________________

   b)________________________________

   c)________________________________

   d)________________________________

REFERENCES

1. M. A. Weiss, “Data Structures and Algorithm Analysis in C”, Pearson Education, 8th Edition, 2007. [Unit I, II, III, IV,V]

2. A. V. Aho, J. E. Hopcroft and J. D. Ullman, “Data Structures and Algorithms”, Pearson Education, 2nd Edition, 2007 [Unit IV].

3. A.M.Tenenbaum, Y. Langsam and M. J. Augenstein, “Data Structures using C”,PearsonEducation, 1st Edition, 2003.(UNIT I,II,V)

4.https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/

THANK YOU