Advanced Algorithms & Data Structures

AIM OF THE SESSION

To familiarize students with the basic concept of Queue Using Singly Linked List

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Demonstrate the Queue Using Singly Linked List
2. List out the operations possible on the Queue Using Singly Linked List.
3. Describe the each operation
4. List out the advantages and applications of Queue Using Singly Linked List.

.

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define Queue Using Singly Linked List
2. Describe the insertion and deletion operations of Queue Using Singly Linked List
3. Summarize definition and operations of Queue Using Singly Linked List

Queue INTRODUCTION

- A queue is an ordered list in which all insertions take place at one end, the rear, while all deletions take place at the other end, the front.

- The most common occurrence of a queue in computer applications is for the scheduling of jobs.

- In batch processing the jobs are "queued-up" as they are read-in and executed, one after another in the order they were received.

Queue INTRODUCTION Contd….

- When we talk of queues we talk about two distinct ends: the front and the rear.
- Additions to the queue take place at the rear. Deletions are made from the front.
- So, if a job is submitted for execution, it joins at the rear of the job queue.
- The job at the front of the queue is the next one to be executed.

Types of Queue

- Simple Queue or Linear Queue
- Circular Queue
- Priority Queue
- Double Ended Queue (or Deque)

Operations performed on queue

The fundamental operations that can be performed on queue are listed as follows -

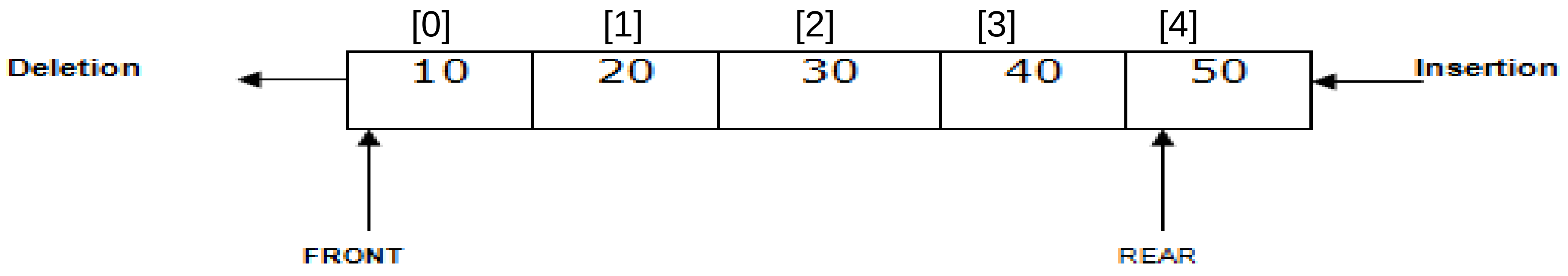
- **Enqueue:** The Enqueue operation is used to insert the element at the rear end of the queue. It returns void.
- **Dequeue:** It performs the deletion from the front-end of the queue. It also returns the element which has been removed from the front-end. It returns an integer value.
- **Peek:** This is the third operation that returns the element, which is pointed by the front pointer in the queue but does not delete it.
- **Queue overflow (isfull):** It shows the overflow condition when the queue is completely full.
- **Queue underflow (isempty):** It shows the underflow condition when the Queue is empty, i.e., no elements are in the Queue.

Linear Queue:

- A ***queue*** is an ordered list in which items may be added only at one end called the ***rear*** and items may be removed only at the other end called ***front.***
- **Examples:**

1. A line of passengers waiting to buy tickets in a reservation counter. Each new passenger gets in line at the rear; the passenger at the front of the line is served.
2. A queue can be found in a time-sharing computer system where many users share the system simultaneously.
3. A line of cars waiting to proceed in some fixed direction at an intersection of streets.

- The first element, which is added into the queue, will be the first one to be removed. Thus queues are also called ***First-In-Fist-Out lists (FIFO)*** or ***Last-In-Last-Out lists (LILO).***



Array Representation of a Queue:

- Queues may be represented in the computer memory by means of linear arrays. There are two pointer variables namely FRONT and REAR, FRONT denotes the location of the first element of the queue. The REAR describes the location of the rear element of the queue.
- ***FRONT==-1*** will indicate that the queue is empty. Whenever an item is added to the queue, the value of ***REAR*** is increased by ***1,*** that is ***REAR=REAR+1,*** whenever an 'item' is deleted
- from the queue, the value of ***FRONT*** is increment by ***1.*** That is ***FRONT=FRONT+1.*** ***FRONT==REAR*** if there are no elements in the queue. Note that the ***FRONT*** of the queue will be the first location of the array.

Algorithm: Insertion into a Queue

```
Step.1:[ Check whether Queue is full]
    If Rear>= Max-1 then
        Print 'Queue Overflow'
    Return

Step.2:[Increment the Rear index]
    Rear ← Rear +1

Step.3:[Insert the new ITEM at rear end of queue */
    Q[Rear] ← ITEM

Step.4:[ if initially the queue is empty, then  adjust Front index]
    If Front  = -1 then
        Front =0

Step.5: return.
```

Algorithm: Deletion from a Queue

```
Step.1: [Check whether the queue is empty]
    If Front=-1 then
        Print ‘Queue Underflow’
    Return
Step.2: [Delete the queue element at front end and store into a
         temporary variable]
    K ← Q[Front]
Step.3: [if queue is empty after deletion, then set front and rear
         indexes to –1, else adjust the Front index]
    If Front = Rear then
    Begin
        Front ← -1
        Rear ← -1
    End
    Else
        Front ← Front + 1
Step.4: return(k).
```

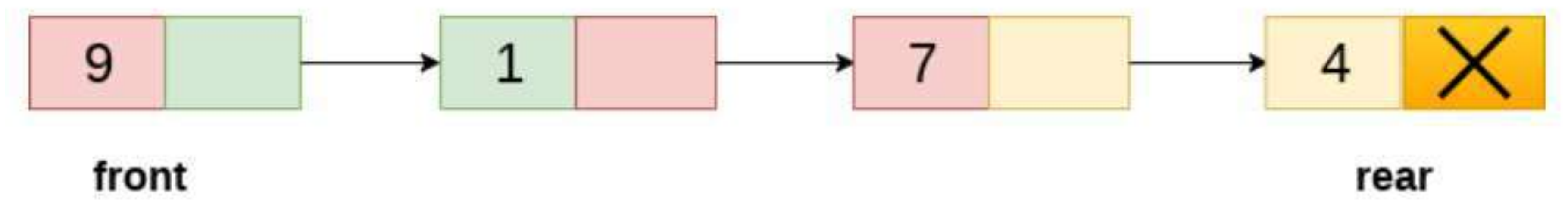
Applications of Queue data structure:

1. Queue is useful in CPU scheduling, Disk Scheduling. When multiple processes require CPU at the same time, various CPU scheduling algorithms are used which are implemented using Queue data structure.
2. When data is transferred asynchronously between two processes. Queue is used for synchronization. Examples: IO Buffers, pipes, file IO, etc.
3. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate. Spooling also lets you place a number of print jobs on a queue instead of waiting for each one to finish before specifying the next one.

4. Breadth First search in a Graph .It is an algorithm for traversing or searching graph data structures. It starts at some arbitrary node of a graph and explores the neighbour nodes first, before moving to the next level neighbours.
5. Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive, First come first served.
6. In real life, Call Center phone systems will use Queues, to hold people calling them in an order, until a service representative is free.

Linked Queue
INTRODUCTION Contd….

- The linked representation of queue is shown in the following figure.



Linked Queue

Operations on Linked Queue

➢ There are two basic operations which can be implemented on the linked queues.

1. Insertion Operation and
2. Deletion Operation.

➢ The insert operation append the queue by adding an element to the end of the queue.

➢ Deletion operation removes the element that is first inserted among all the queue elements.

SUMMARY

- A queue is an ordered list in which all insertions take place at one end, the rear, while all deletions take place at the other end, the front.
- The array implementation can not be used for the large scale applications where the queues are implemented.
- One of the alternative of array implementation is linked list implementation of queue.
- In a queue using singly linked list, we perform the following operations...
  i) Insertion ii) Deletion iii) Display iv) Search
- The insert operation append the queue by adding an element to the end of the queue.
- Deletion operation removes the element that is first inserted among all the queue elements.

SELF-ASSESSMENT QUESTIONS

1. In linked list implementation of a queue, where does a new element be inserted?

a) At the head of link list

b) At the centre position in the link list

c) At the tail of the link list

d) At any position in the linked list

2. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into EMPTY queue?

a) Only front pointer
b) Only rear pointer
c) Both front and rear pointer
d) No pointer will be changed

TERMINAL QUESTIONS

1. Describe About Queue using Singly Linked List?

2. List out different operations can perform on Queue using Singly Linked Lists?

3. Analyze the time complexity of the Queue using Singly Linked List for the operations such as insertion, deletion, and search.

4. Write a C language program that implement the Queue using Singly Linked List.

5. Summarize What is Queue using Singly Linked List and operations possible on it?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. Mark Allen Weiss, Data Structures and Algorithm Analysis in C, 2010 , Second Edition, PearsonEducation.
2. 2. Ellis Horowitz, Fundamentals of Data Structures in C: Second Edition, 2015
3. A.V.Aho, J. E. Hopcroft, and J. D. Ullman, “Data Structures And Algorithms”, Pearson Education, First Edition Reprint2003.

Sites and Web links:

1. https://nptel.ac.in/courses/106102064
2. https://in.udacity.com/course/intro-to-algorithms--cs215
3. https://www.coursera.org/learn/data-structures?action=enroll

DEEMED TO BE UNIVERSITY

THANK YOU