Data Structures

Queue

Rutvij H. Jhaveri
Computer Science & Engineering

Outline

- What is Queue?
  - Applications
- Circular Queue
- Priority Queue
- Double-Ended Queue

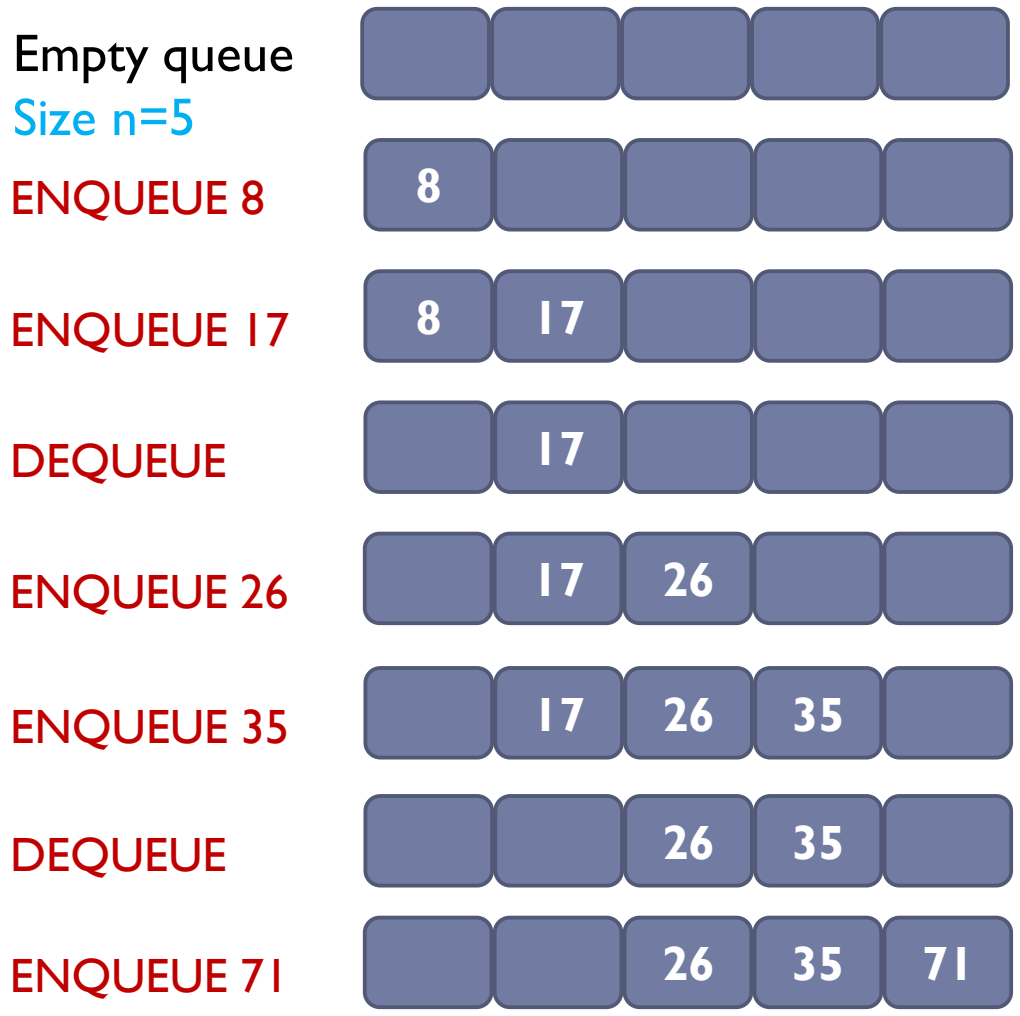
What is a Queue?

- A linear data structure containing ordered elements
- Insertion of an element is performed at **rear** end
- Deletion of an element is performed at **front** end
- Least recently inserted item is removed first
- Follows First In First Out (FIFO) principle

Basic Queue Operations

- ENQUEUE: Insert an element at **Rear**
- DEQUEUE: Remove an element from **Front**
- PEEK: Display the element at the front of the queue
- isEmpty: Check whether queue is empty
- isFull: Check whether queue is full
- Traverse: Print all the elements of the queue from front to rear.

ENQUEUE, DEQUEUE, isEmpty, isFull Operations



Disadvantage
There are 2 empty slots in spite of queue is Full.
Thus, linear queue wastes memory

Conditions: isEmpty() -> Front=-1 or Front>Rear isFull() -> Rear=n-1

PEEK and TRAVERSE Operations

Front=2 Rear=4

PEEK: Display 26

TRAVERSE: Display 26, 35, 71

Psuedocode

- Initialize(Queue, Front, Rear)
  - Front=Rear=-1
- IsEmpty(Queue, Front, Rear)
  - IF Front==-1 OR Front>Rear:
    - Return True
  - ELSE:
    - Return False
- IsFull (Queue, Rear, MAX)
  - IF Rear==MAX-1:
    - Return True
  - Else:
    - Return False

…Psuedocode

- Enqueue (Queue, Rear, Item)
  - IF (!IsFull()):
    - Rear++
    - Queue[Rear]=Item
    - IF(Front=-1)
      - Front++
- Dequeue (Queue, Front)
  - IF (!IsEmpty()):
    - Remove Queue[Front]
    - Front++
- Peek (Queue, Front)
  - IF (!IsEmpty()):
    - Display Queue[Front]
- Traverse (Queue, Front, Rear)
  - IF (!IsEmpty()):
    - For i=Front to Rear:
      - Display Queue[i]

Applications

- CPU scheduling
- Disk scheduling
- Job scheduling
- Resource management
- Implementing printer spooler
- …

Types of Queues

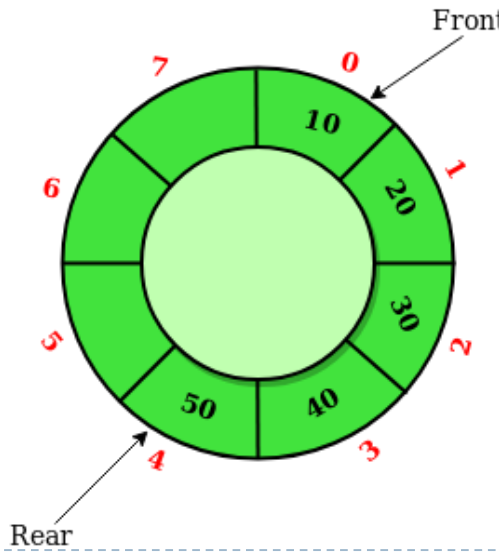
- Linear Queue (Simple Queue) [studied so far]
- Circular Queue
- Priority Queue
- Double Ended Queue (DE-Queue)

Circular Queue Vs Linear Queue

- Linear queue wastes memory
- Circular queue was devised to limit the memory wastage of the linear queue.
- In circular queue, last element is connected with the first element.
- In circular queue, after Rear reaches to n-1, a new element can be inserted at 0^{th} position (if empty)

Circular queue implementation

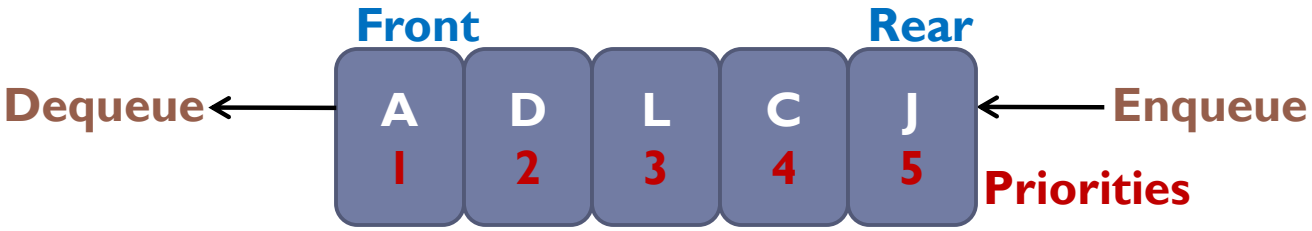
circular_queue_op.c



Priority Queue

- Priority Queue is an extension of queue in which:

1. Every item has a priority associated with it.
2. An element with higher priority is dequeued before an element with lower priority.
3. If two elements have the same priority, they are served according to their order (FIFO) in the queue.



Elements in queue arranged according to the priority (e.g. 1 the highest, 5 the lowest)

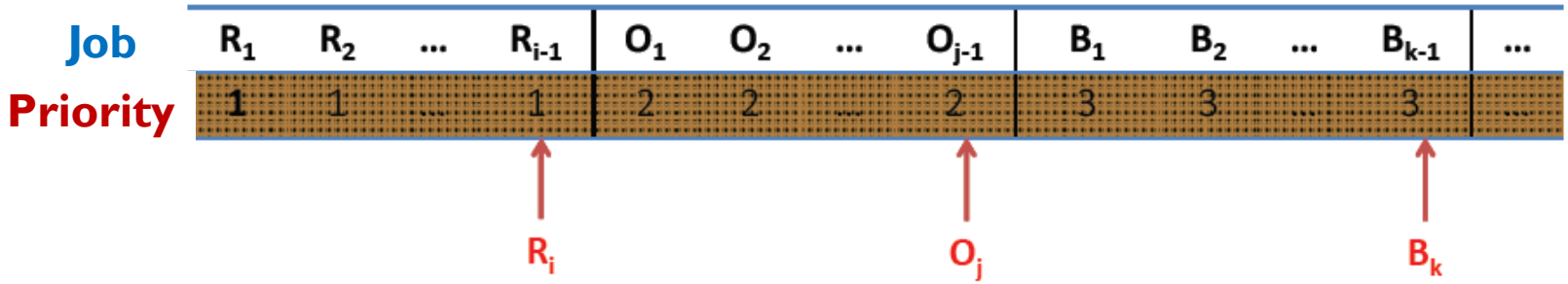
```
struct priority_queue
{
    int ele[MAX][1];
    int front, rear;
};
```

…Priority Queue

Example: Job scheduling

- **Priority 1 for Real Time Jobs (R_i)**
- **Priority 2 for Online Jobs (O_i)**
- **Priority 3 for Batch Processing Jobs (B_i)**

- The queue is divided into 3 lists: first with Priority 1 (Highest), Second with Priority 2 (Next highest) and third with Priority 3 (Lowest).
- If a job is initiated with priority *i*, it is inserted immediately at the end of list of other jobs with same priority.
- Here jobs are always removed from the front of the queue.



Priority Queue viewed as a single queue with insertion allowed at any position

Applications of Priority Queue

- Job scheduling
- For load balancing and interrupt handling in an operating system
- For data compression in Huffman code
- Bandwidth management
- Discrete event simulation
- All queue applications where priority is involved

Double-Ended Queue (De-Queue)

De-Queue has the following properties:

1. Has two ends.
2. Elements can be enqueued from both the front and the rear.

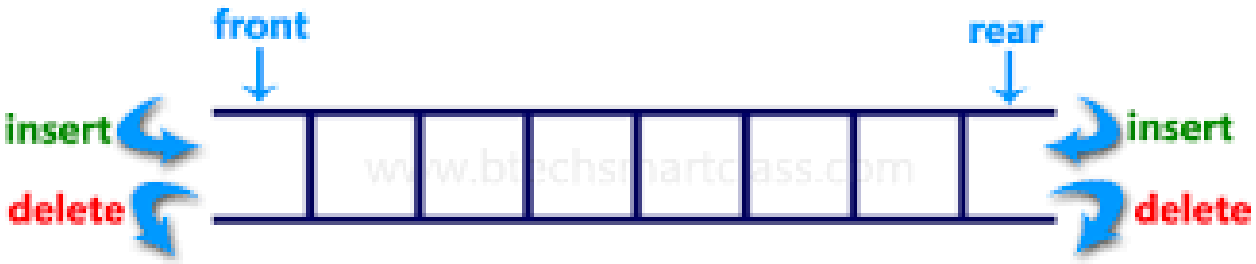
Applications

- Palindrome checker
- Steal job scheduling for multiprocessor scheduling
- Undo-Redo operations

…Double-Ended Queue (De-Queue)

De-Queue can be implemented with circular queue with these major operations:

- **Enqueue_Front()**: Adds an item at the front.
- **Enqueue_Rear()**: Adds an item at the rear.
- **Dequeue_Front()**: Deletes an item from front.
- **Dequeue_Rear()**: Deletes an item from rear of Deque.



Summary

- Queue >> FIFO
- Applications
- Types of queues
  - Linear
  - Circular
  - Priority
  - De-Queue

Assignments

- Psuedocode for Linear Queue
- Psuedocode for Double-Ended Queue