Data Structures and Algorithms
CS-206

Queue

Instructor
Dr. Maria Anjum
Assistant Professor

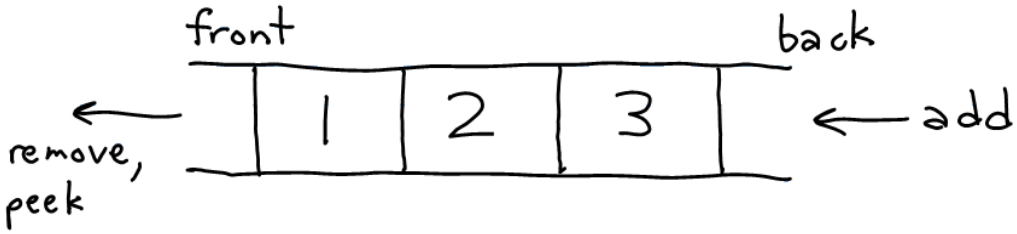
Department of Computer Science
Lahore College for Women University


Rear of Queue
Front of Queue
next...
PAY HERE

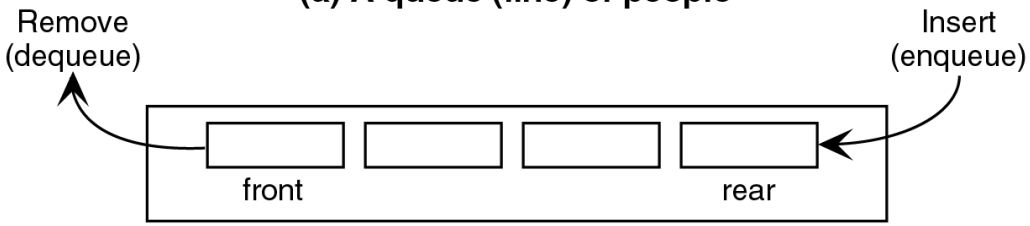
Queue Strategy

First-in, First-out (FIFO)

Objects enter the line at one end (rear)
Objects leave the line at the other end (front)

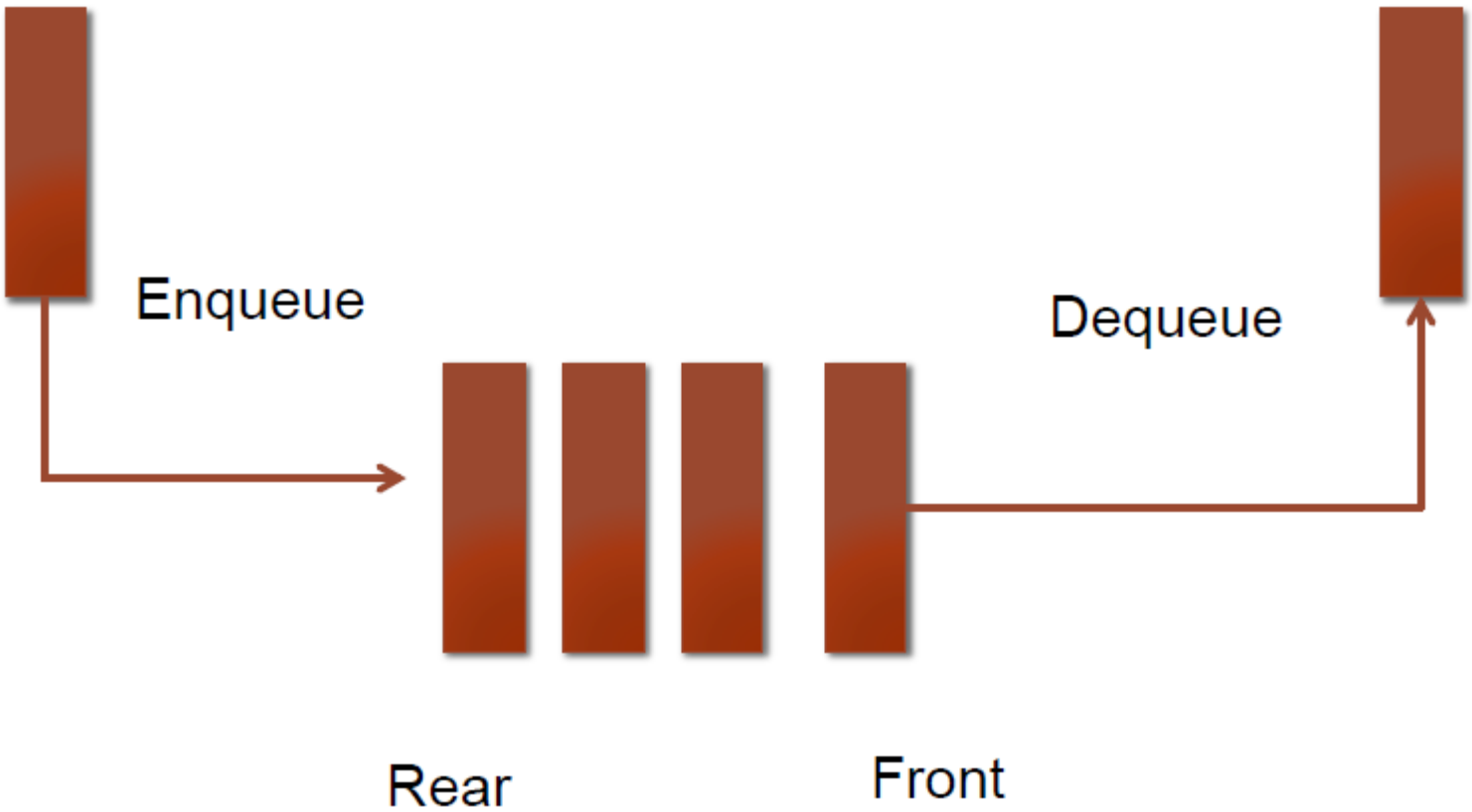


(a) A queue (line) of people



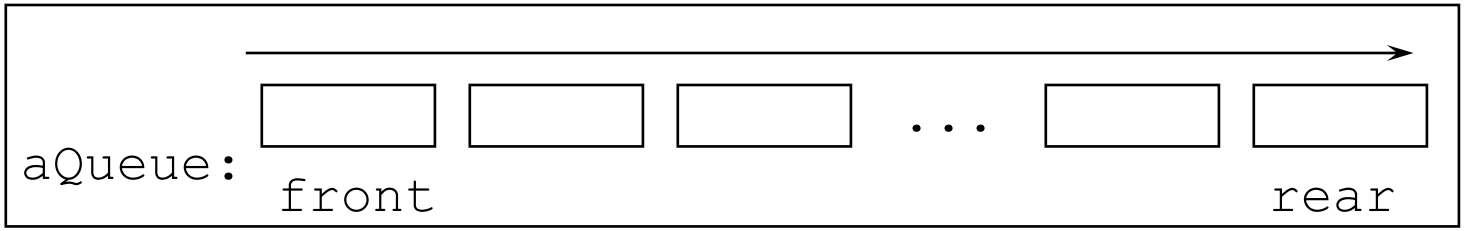
(b) A computer queue

Queue Strategy



Queue Definition

- Queue: Ordered collection, accessed only at the front (remove) and rear (insert)
  - Front: First element in queue
  - Rear: Last element of queue
- FIFO: First In, First Out
- Footnote: picture can be drawn in any direction



Queue Example

Operation	output	queue
• enqueue(5)	-	(5)
• enqueue(3)	-	(5, 3)
• dequeue()	5	(3)
• enqueue(7)	-	(3, 7)
• dequeue()	3	(7)
• front()	7	(7)
• dequeue()	7	()
• dequeue()	"error"	()
• isEmpty()	true	()
• enqueue(9)	-	(9)
• size()	1	(9)

Queues in Computer Science

- Operating systems:
  - queue of print jobs to send to the printer
  - queue of programs / processes to be run
  - queue of network data packets to send

- Programming:
  - modeling a line of customers or clients
  - storing a queue of computations to be performed in order

- Real world examples:
  - people on an escalator or waiting in a line
  - cars at a gas station (or on an assembly line)

Queue Operations

- Initialize the queue, *Q*, to be the empty queue.
- Determine whether or not if the queue *Q* is empty.
- Determine whether or not if the queue *Q* is full.
- Insert (enqueue) a new item onto the rear of the queue *Q*.
- Remove (dequeue) an item from the front of *Q*, provided *Q* is nonempty.

Queue Operations

- enqueue() Adds an element to rear of queue succeeds unless the queue is full (if implementation is bounded)

  Or insert()

- dequeue() Remove and return the front element of queue, Precondition: queue is not empty

  Or remove()

- Front() Return a copy of the front element of queue, precondition: queue is not empty
- Size()
- isEmpty()
- isFull()

Queue Example

- Draw a picture and show the changes to the queue in the following example:
    - Queue q; Object v1, v2;

    - q.insert(“chore”);
    - q.insert(“work”);
    - q.insert(“play”);
    - v1 = q.remove();
    - v2 = q.front();
    - q.insert(“job”);
    - q.insert(“fun”);

What is the result of:

- Queue q; Object v1,v2,v3,v4,v5,v6
- q.insert(“Sue”);
- q.insert(“Sam”);
- q.insert(“Sarah”);
- v1 = q.remove( );
- v2 = q. front( );
- q.insert(“Seymour”);
- v3 = q.remove( );
- v4 = q.front( );
- q.insert(“Sally”);
- v5 = q.remove( );
- v6 = q. front( );

Applications: Job Scheduling

front	rear	Q[0]	Q[1]	Q[2]	Q[3]	Comments
-1	-1					queue is empty
-1	0	J1				Job 1 is added
-1	1	J1	J2			Job 2 is added
-1	2	J1	J2	J3		Job 3 is added
0	2		J2	J3		Job 1 is deleted
1	2			J3		Job 2 is deleted

Implementing Queue ADT: Array Queue

- Keep track of the number of elements in the queue, `size`.
- Enqueue at the back of the array (`size`).
- Dequeue at the front of the array (index 0).

  – what is bad about this implementation?
  – what if we enqueue at 0 and dequeue at `size`?

Array Implementation

Any implementation of a queue requires:

storage for the data as well as

markers ("pointers") for the front and for the back of the queue.

An ***`array-based`*** implementation would need structures like

`items`, an array to store the elements of the queue

`Front`, an index to track the front queue element

`Rear`, an index to track the position *following* last queue element

`Additions` to the queue would result in incrementing ***Rear***.

`Deletions` from the queue would result in incrementing ***Front***.

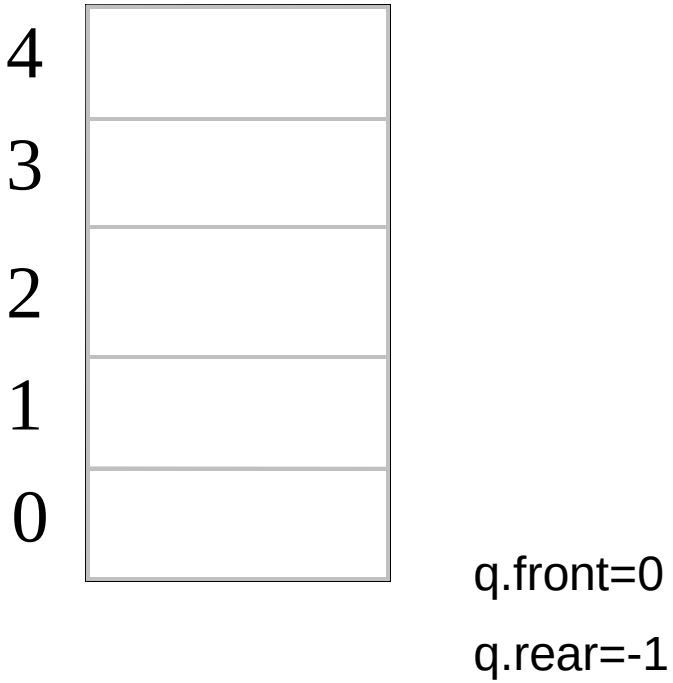
Clearly, we'd run out of space soon!

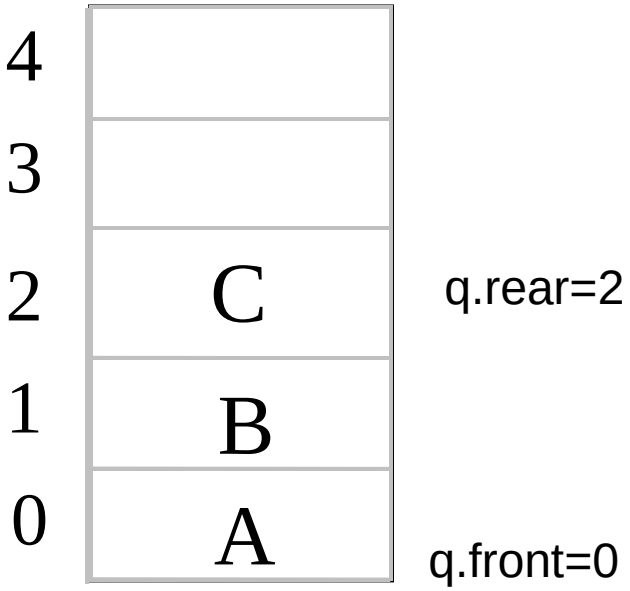
Queue using Arrays

```
#define length 10
Struct queue
{
    int items[length];
    int front, rear;
}
```

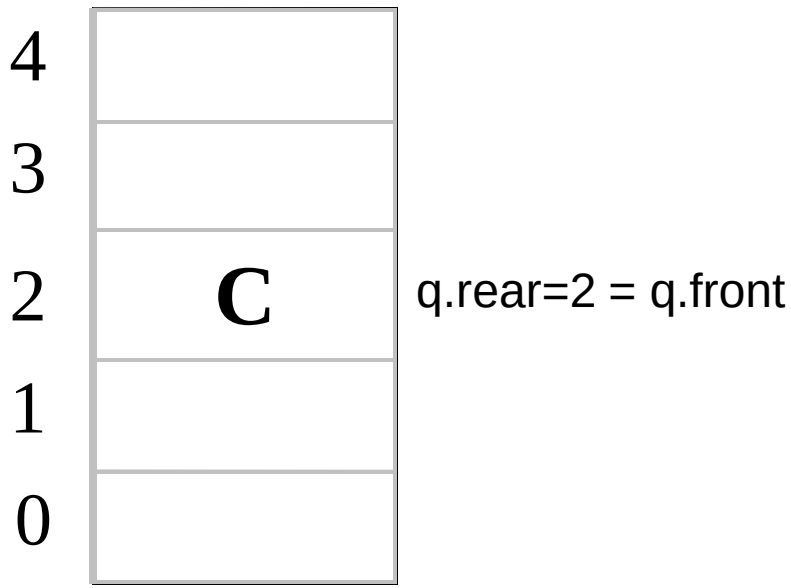
- Insert(q,x)
  q.items[++q.rear]=x;

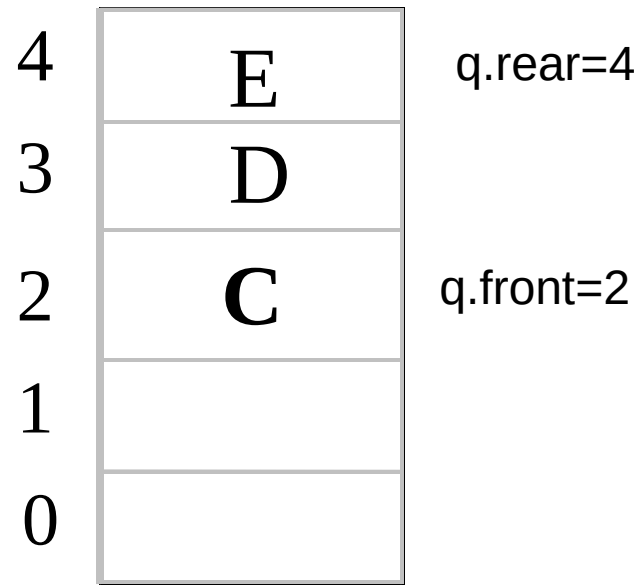
- X=remove(q)
  x=q.items[q.front++];





- The queue is empty whenever q.rear<q.front
- The number of elements in the queue at any time is equal to the value of ***q.rear –q.front +1***





•Now there are 3 elements the queue but there is room for 5

•If to insert *F* in the queue the q.rear must be increased by 1 to 5 and q. items[5] must be set to the value F. but q.items is an array of only 5 elements so this insertion cannot be made

Solution to Problem

Clearly, we'have run out of space

Solutions include:

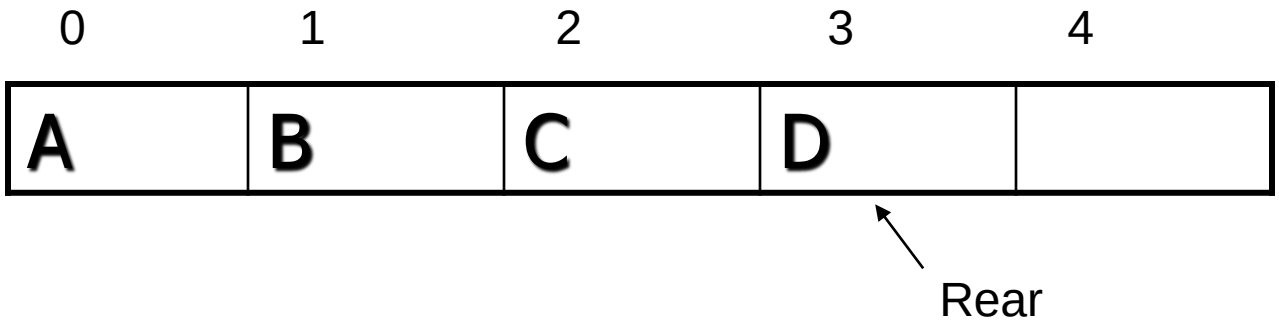
- Shifting the elements downward with each deletion

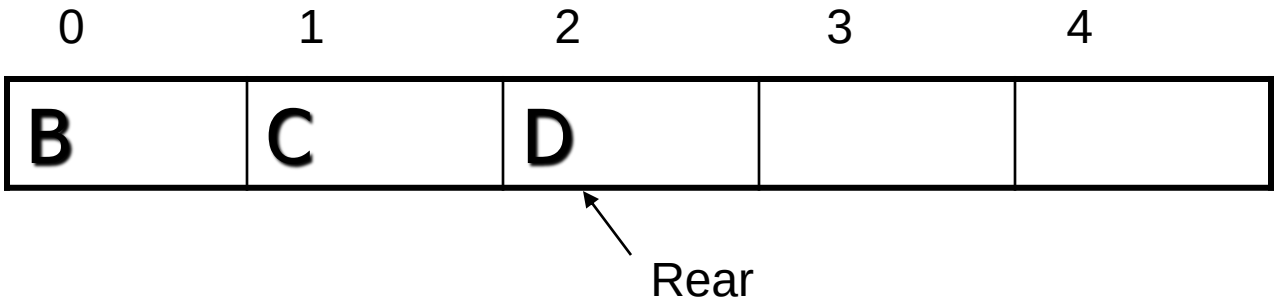
- Viewing array as a circular buffer, i.e. wrapping the end to the front

Solution 1

- For arrays there are two methods
- First is do it as we do in real world
  - Check if array is not empty
  - Simply dequeue from the first location of array say array[0] i.e. the zeroth index
  - After dequeue shift all the elements of the array from array[index] to array[index-1]

Dequeue





DE-QUEUE

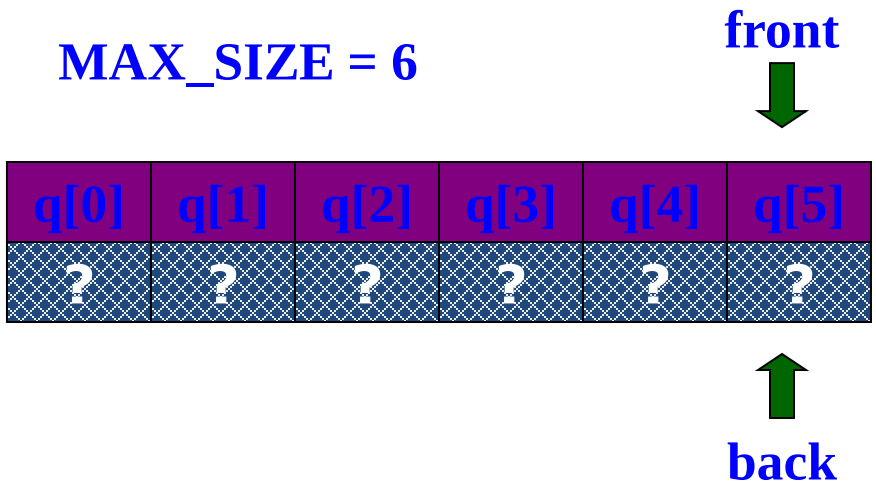
Dequeue Operation

```
x=q.items[0];
for(i=0; i<q.rear; i++)
    q.items[i]=q.items[i+1]
q.rear--;
```

- Method is costly as we have to move all the elements of the array
- Do we need Front Index in this case?
  - No, Because we are always dequeue(ing) from the first index

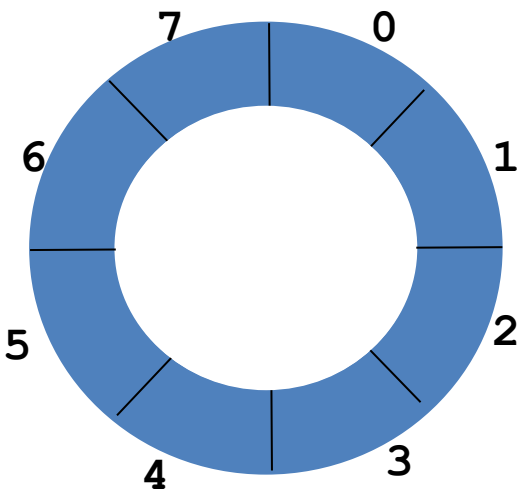
Solution2: Circular Queue

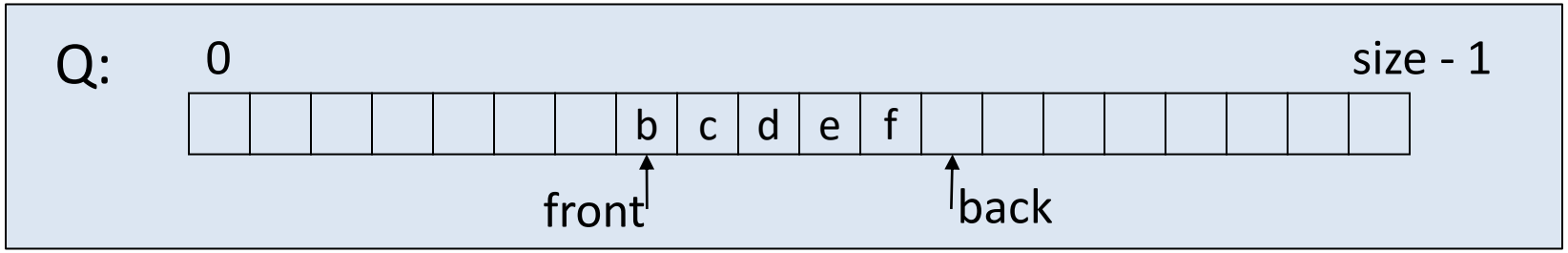
- To avoid the costly operation of coping all the elements again we employ another method called circular queue
- Simply dequeue the element and move the front pointer to next index
- If q.rear==q.front queue is empty
- q.front=q.rear=max_size-1;



Implementing Queue ADT: Circular Array Queue

- **Neat trick**: use a ***circular array*** to insert and remove items from a queue in constant time.
- The idea of a circular array is that the end of the array "wraps around" to the start of the array.





Summary

- Stacks and Queues
  - Specialized list data structures for particular applications
- Stack
  - LIFO (Last in, first out)
  - Operations: push(Object), top( ), and pop( )

- Queue
  - FIFO (First in, first out)
  - Operations: insert(Object), getFront( ), and remove( )
- Implementations: arrays or lists are possibilities for each

Assignment

Palindromes are words which can be read same from forward and revers. Few examples are:

- Radar
- Mom
- Dad
- Stats
- Madam
- Wassamassaw

How we may use Stack and Queue to determine a given word is palindrome?

Credit

Data Structures and Algorithms in C++ Goodrich, Tamassia and Mount (Wiley, 2004)

University of Washington