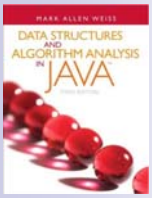
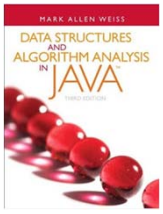


FIU FLORIDA INTERNATIONAL UNIVERSITY

Lists, Stacks, and Queues (V)

Dr. Antonio L. Bajuelos

FIU School of Computing & Information Sciences



COP-3530 - Data Structures

Module #2: Lists, Stacks, and Queues (part V)

Outline:

- The Queue model (FIFO list).
- Array implementation of a Queue.
- Circular array implementation of a Queue.
- Complexity analysis.
- Applications of Queues.

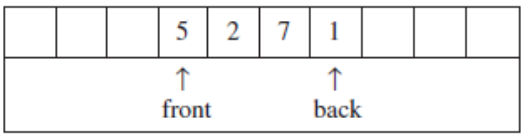
The Queue Model

- **Definition:** A **queue** is a list of elements such that:
  - elements are **inserted** at the end of the list (the rear/back) and
  - **deleted** at the front of the list
- **Queues** are sometimes known as
  - **FIFO** - **(F)irst (I)n, (F)irst (O)ut**) lists.
- **Operations:**
  - **enqueue** – insert at the end
  - **dequeue** - delete the most recently inserted element
- **Important point:**
  - For queues, both the **linked list** and **array implementations** give fast *O*(1) running times for every operation.

3

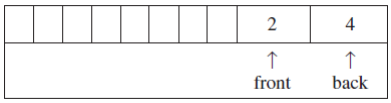
Array implementation of Queues

- For each **queue**, we keep:
  - an array - **theArray**;
  - the positions **front** and **back**, which represent the ends of the queue;
  - the number of elements that are actually in the queue, **currentSize**.



4

Array implementation of Queues



- **Operations:**
  - **init**

    front = 0; back = -1; currentSize = 0

    qMaxSize //size of queue
  - To **enqueue** (insert) an element x:

    currentSize +=1;

    back +=1;

    theArray[back]=x;
  - To **dequeue** (delete) an element:

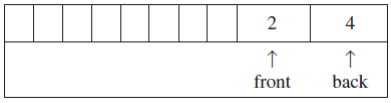
    the return value is theArray[front];

    currentSize -=1;

    front +=1;

5

Array implementation of Queues



- **Potential problem:**
  - After 10 **enqueues**, the queue appears to be full, since back is now at the last array index, and the next **enqueue** would be in a nonexistent position!!!
- **Simple solution:**
  - Whenever front or back gets to the end of the array, it is wrapped around to the beginning - **circular array implementation**

6

Circular Array Implementation of Queues

- **Operations:**
  - **init**

    front = 0; back = -1; currentSize = 0

    qMaxSize //size of queue

  - To **enqueue** (insert) an element x:

    if currentSize == qMaxSize

    Overflow Error!

    else

    currentSize +=1;

    back = (back + 1) mod qMaxSize;

    theArray[back]=x;

  - To **dequeue** (delete) an element:

    if currentSize == 0

    Underflow Error!

    else

    the return value is theArray[front];

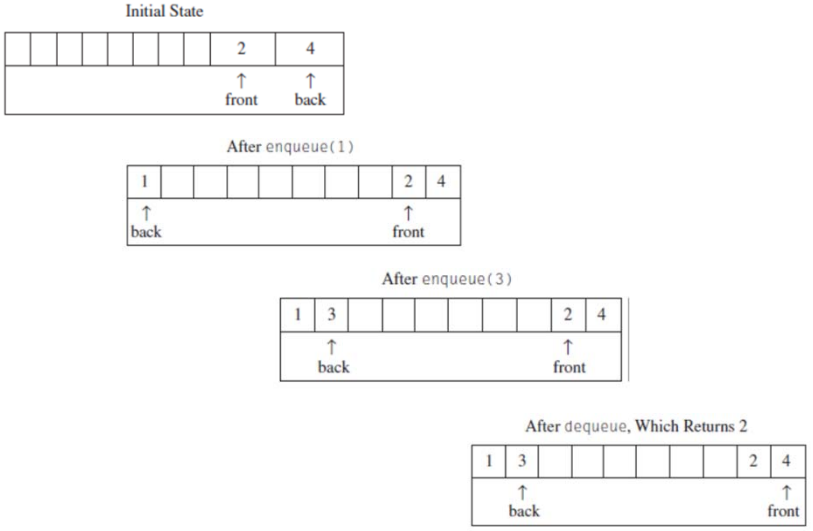
    currentSize -=1;

    front = (front +1) mod qMaxSize;

7

Circular Array Implementation of Queues

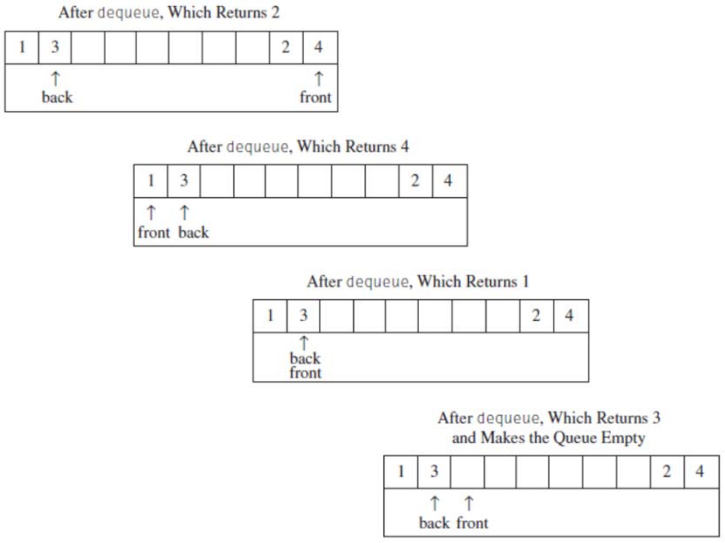
- Example



8

Circular Array Implementation of Queues

- Example (cont…)



9

Implementation of Queues

- See **Java** implementation of **Queues** using a **Linked-list**:

http://introcs.cs.princeton.edu/java/43stack/Queue.java.html

- See **Java** implementation of **Queues** using a **Circular Array**:

http://www.cs.utsa.edu/~wagner/CS2213/queue/queue.html

10

Applications of the Queues

- Some simple examples of queue usage:
  - When jobs are submitted to a **printer**, they are arranged in order of arrival.
  - Virtually every real-life line is (supposed to be) a queue: **first-come first-served**.
  - In **computer networks**, **users** on other computers (file server) are given access to files on a first-come first-served basis.
  - **Calls** to large companies are generally placed on a **queue** when all operators are busy.
  - Etc...

11

12