DATA STRUCTURE AND ALGORITHMS

LECTURE 4

Stack, Queue

DATA STRUCTURE AND ALGORITHMS

LECTURE 4b

Queue ADT

Reference links:

https://cs.nyu.edu/courses/fall17/CSCI-UA.0102-007/notes.php

https://www.comp.nus.edu.sg/~stevenha/cs2040.html

[M.Goodrich, chapter 6, sec 6.2]

Lecture outline

- Queue ADT
  - Introduction
  - Specification
  - Implementations
    - Array Based
    - Linked List Based
  - Applications
    - Palindrome checking

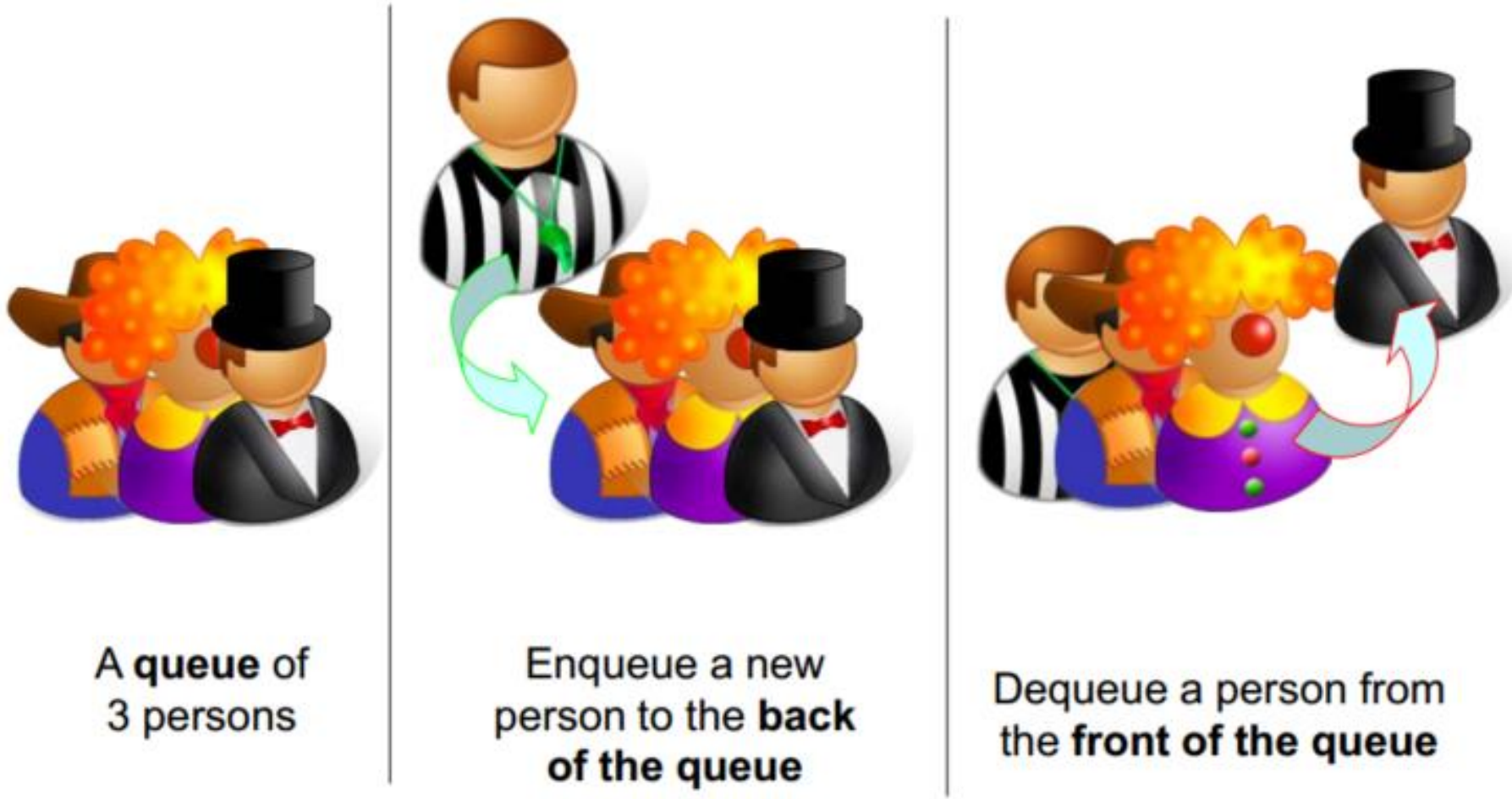
Queue ADT

Introduction

What is a queue

- Real life example:
  - A queue for movie tickets, Airline reservation queue, etc
- First item added will be the first item to be removed
  - Known as **F**irst **I**n **F**irst **O**ut (**FIFO**) property
- Major Operations:
  - enqueuer(): Items are added to the **back of the queue**
  - dequeue(): Items are removed from the **front of the queue**
  - first(): Take a look at the first item
  - size(): Returns the number of elements in the queue
  - isEmpty(): Check the queue is empty or not.

Queue: Illustration



Source: Visualgo.net

Queue ADT

Specification

Queue specification in Java

```
public interface Queue<E> {
        // Returns the number of elements in the queue
        int size();
        //Tests whether the queue is empty
        boolean isEmpty();
        // Inserts an element at last of the queue
        void enqueue(E e);
        //Returns, but not remove, the first element (null if empty)
        E first();
        //Removes and returns the top element from the stack
        E dequeue();
}
```

A simple version of the queue interface [M.Goodrich,239]

Queue specification in Java

Method	Return Value	first $\leftarrow Q \leftarrow$ last
enqueue(5)	–	(5)
enqueue(3)	–	(5, 3)
size( )	2	(5, 3)
dequeue( )	5	(3)
isEmpty( )	false	(3)
dequeue( )	3	( )
isEmpty( )	true	( )
dequeue( )	null	( )
enqueue(7)	–	(7)
enqueue(9)	–	(7, 9)
first( )	7	(7, 9)
enqueue(4)	–	(7, 9, 4)

Example queue operations on queue Q of intergers [M.Goodrich,240]

Queue specification in Java

- Class Queue in java.util
  - https://docs.oracle.com/javase/9/docs/api/java/util/Queue.html

Queue ADT

Implementation

Queue: Implementations

- Two ways to implement Queue ADT:
  - Array based implementation
    - Removing item at the head is the worst case
    - Adding item at the back is the best case
  - Single Linked List implementation
    - Removing item at the head is the best case
    - Adding item at the back is the worst case

- Is it possible to have both efficient **enqueue()** and **dequeue()** operations?

Queue: Implementations

Using Array

[M. Goodrich, sub-section 6.2.2]

Queue Implementation using array

```
1  /** Implementation of the queue ADT using a fixed-length array. */
2  public class ArrayQueue<E> implements Queue<E> {
3    // instance variables
4    private E[ ] data;                          // generic array used for storage
5    private int f = 0;                          // index of the front element
6    private int sz = 0;                         // current number of elements
7
8    // constructors
9    public ArrayQueue( ) {this(CAPACITY);} // constructs queue with default capacity
10   public ArrayQueue(int capacity) {          // constructs queue with given capacity
11     data = (E[ ]) new Object[capacity];      // safe cast; compiler may give warning
12   }
13
14   // methods
15   /** Returns the number of elements in the queue. */
16   public int size( ) { return sz; }
17
18   /** Tests whether the queue is empty. */
19   public boolean isEmpty( ) { return (sz == 0); }
20
21   /** Inserts an element at the rear of the queue. */
22   public void enqueue(E e) throws IllegalStateException {
23     if (sz == data.length) throw new IllegalStateException("Queue is full");
24     int avail = (f + sz) % data.length;      // use modular arithmetic
25     data[avail] = e;
26     sz++;
27   }
```

Array-based implementation of the Queue interface [M.Goodrich, p243]

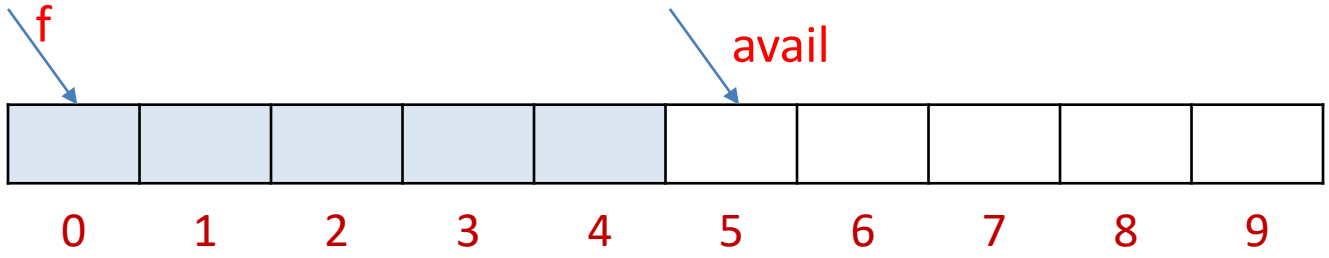
Queue Implementation using array

```
28    /** Returns, but does not remove, the first element of the queue (null if empty). */
29    public E first( ) {
30      if (isEmpty( )) return null;
31      return data[f];
32    }
33
34    /** Removes and returns the first element of the queue (null if empty). */
35    public E dequeue( ) {
36      if (isEmpty( )) return null;
37      E answer = data[f];
38      data[f] = null;                          // dereference to help garbage collection
39      f = (f + 1) % data.length;
40      sz--;
41      return answer;
42    }
```

- Three instance variables:
  - data : a reference to the inderlying array
  - f : the first element of the queue (assuming the queue is not empty)
  - sz : the current number of elements stored in the queue (not to be confused with the length of the array)

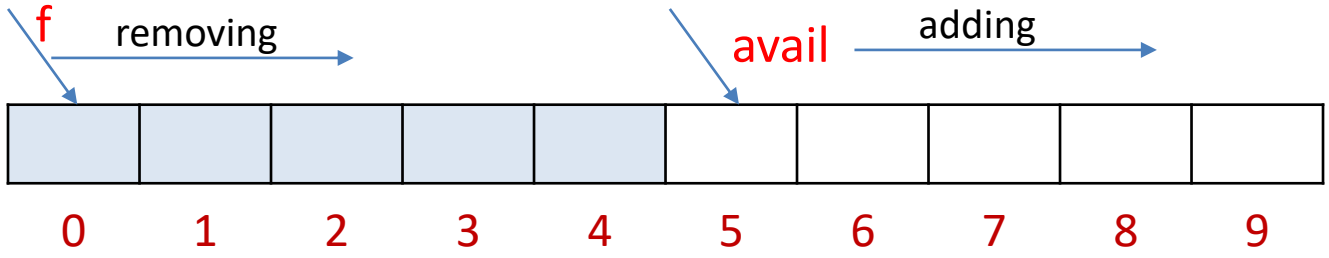
Queue Implementation using array

- Adding and Removing Elements
  - Adding elements with enqueue method
    - avail = (f + sz) % data.length;
  - Removing element with dequeue method
    - f = (f+1) % data.length



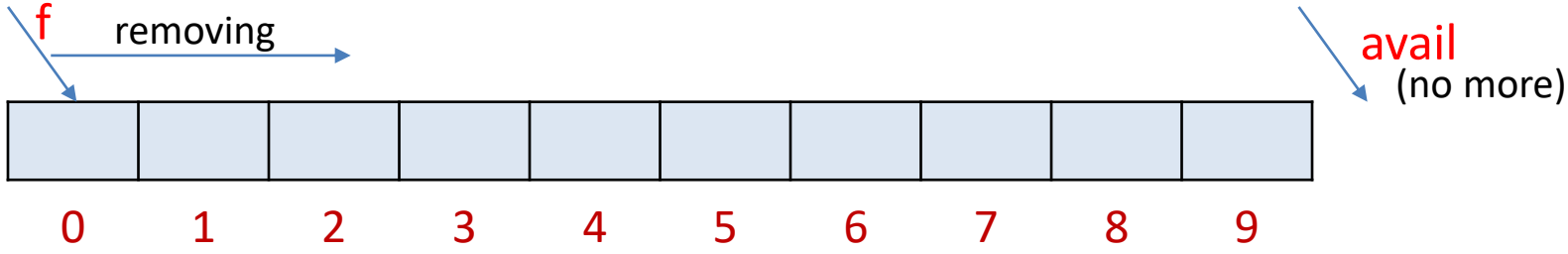
Queue Implementation using array

- Adding and Removing Elements
  - Adding elements with enqueue method
    - avail = (f + sz) % data.length;
  - Removing element with dequeue method
    - f = (f+1) % data.length



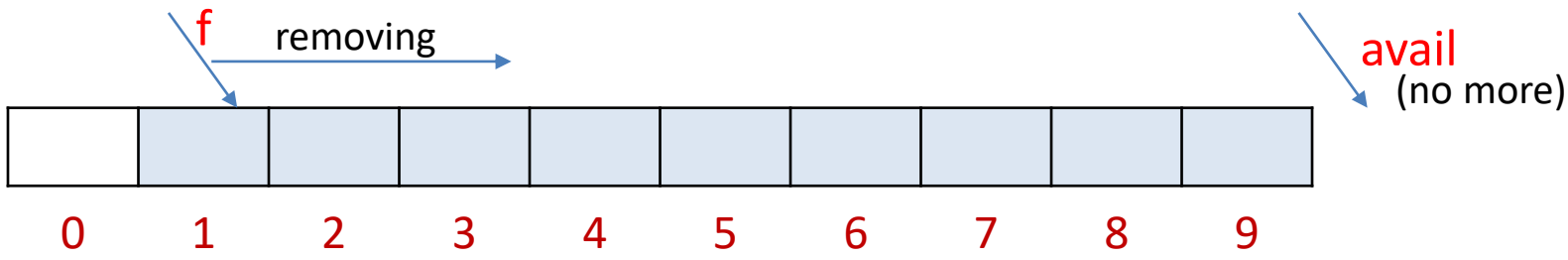
Queue Implementation using array

- Adding and Removing Elements
  - Adding elements with enqueue method
    - avail = (f + sz) % data.length;
  - Removing element with dequeue method
    - f = (f+1) % data.length



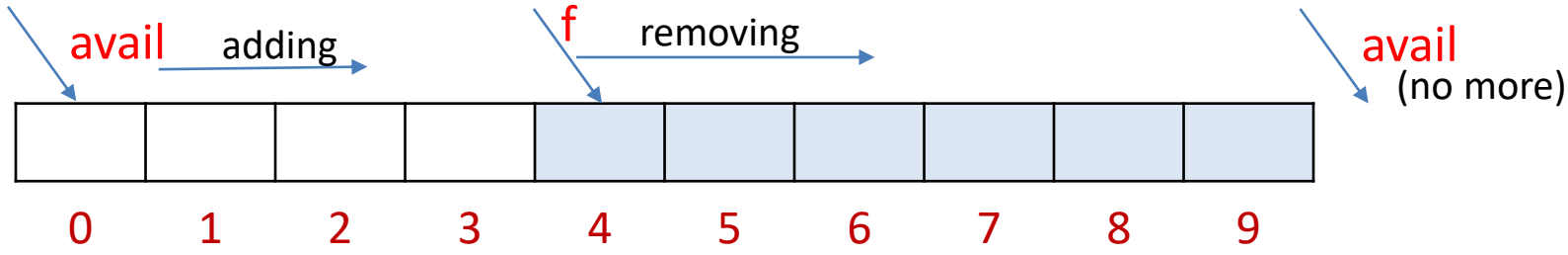
Queue Implementation using array

- Adding and Removing Elements
  - Adding elements with enqueue method
    - avail = (f + sz) % data.length;
  - Removing element with dequeue method
    - f = (f+1) % data.length



Queue Implementation using array

- Adding and Removing Elements
  - Adding elements with enqueue method
    - avail = (f + sz) % data.length;
  - Removing element with dequeue method
    - f = (f+1) % data.length



Queue Implementation using array

- Analyzing the efficiency
  - Running time

Method	Running Time
size	O(1)
isEmpty	O(1)
first	O(1)
enqueue	O(1)
dequeue	O(1)

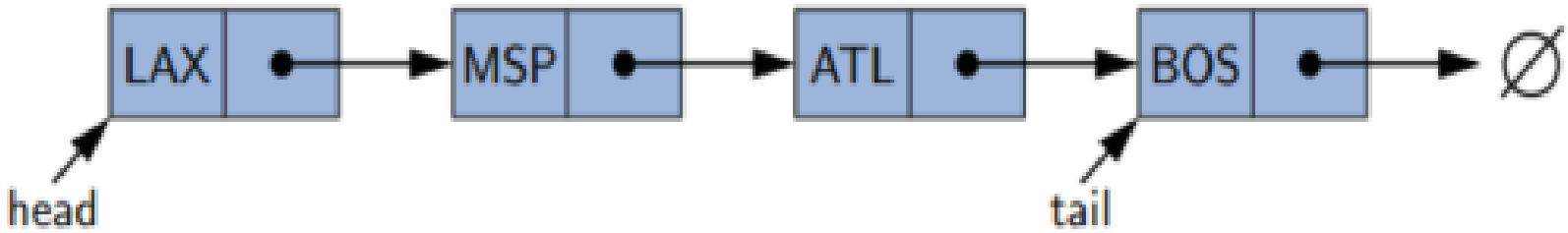
  - Storage: O(*N*) - *N* là kích thước mảng được khai báo

Queue: Implementations

Using Singly Linked List

[M. Goodrich, subsection 6.2.3]

Queue Implementation using linked list



- Characteristics of singly linked list
  - Efficient manipulation of head, tail node:
    - No need to traverse the list – không cần duyệt danh sách
  - Without an arbitrary capacity limit – không giới hạn số phần tử
- Hence, use singly linked list for storing of queue
  - Use head node as the front of the queue and tail node as the back of the queue

Queue Implementation using linked list

- The corresponding methods from SinglyLinkedList:

Queue Method	Singly Linked Method
size()	List.size()
isEmpty()	list.isEmpty)
enqueue(e)	list.addLast()
first()	list.first()
dequeue()	list.removeFirst()

- Implements:

```
public class LinkedQueue<E> implements Queue<E> {
  private SinglyLinkedList<E> list = new SinglyLinkedList<>();    // an empty list
  public LinkedQueue( ) { }                  // new queue relies on the initially empty list
  public int size( ) { return list.size( ); }
  public boolean isEmpty( ) { return list.isEmpty( ); }
  public void enqueue(E element) { list.addLast(element); }
  public E first( ) { return list.first( ); }
  public E dequeue( ) { return list.removeFirst( ); }
}
```

Implementation of the Queue using SinglyLinkedList [M.Goodrich, p245]

Queue Implementation using linked list

- Analyzing the implementation
  - Running time – same as using array

Method	Running Time
size	O(1)
isEmpty	O(1)
enqueue(e)	O(1)
first()	O(1)
dequeue()	O(1)

  - Storage: O(*M*) - *M* là số nhiều nhất phần tử được lưu và xử lí trong queue

Queue ADT

Queue applications

Fist In First Out!

Queue applications: a fun!

```
void fun(int n) {
        IntQueue q = new IntQueue();
        q.enqueue(0);
        q.enqueue(1);
        for (int i = 0; i < n; i++) {
                        int a = q.dequeue();
                        int b = q.dequeue();
                        print(a);
                        q.enqueue(b);
                        q.enqueue(a + b);
        }
}
```

Giả sử `IntQueue` là hàng đợi số nguyên (integer queue).

Hàm fun thực hiện việc gì?

Queue applications

- An other fun problem:
  - Checking for palindrome (kiểm tra tính đối xứng của xâu)

- Palindrome is a string which reads the same either left to right, or right to left. Example:
  - “radar”
  - “a man, a plan, a canal - panama!”
  - “Able I was ere, I saw Elba.”
  - “Won ton? Not now!”
  - “Madam, I’m Adam.”
  - “Eve.”

Palindrome: Problem Description

- Many solutions:
  - We use the two newly learned ADTs
  - Highlight the difference of LIFO and FIFO properties
- Idea with stack and queue:
  - Use stack to reverse the input
  - Use queue to preserve the input
  - The two sequence should be the same for palindrome

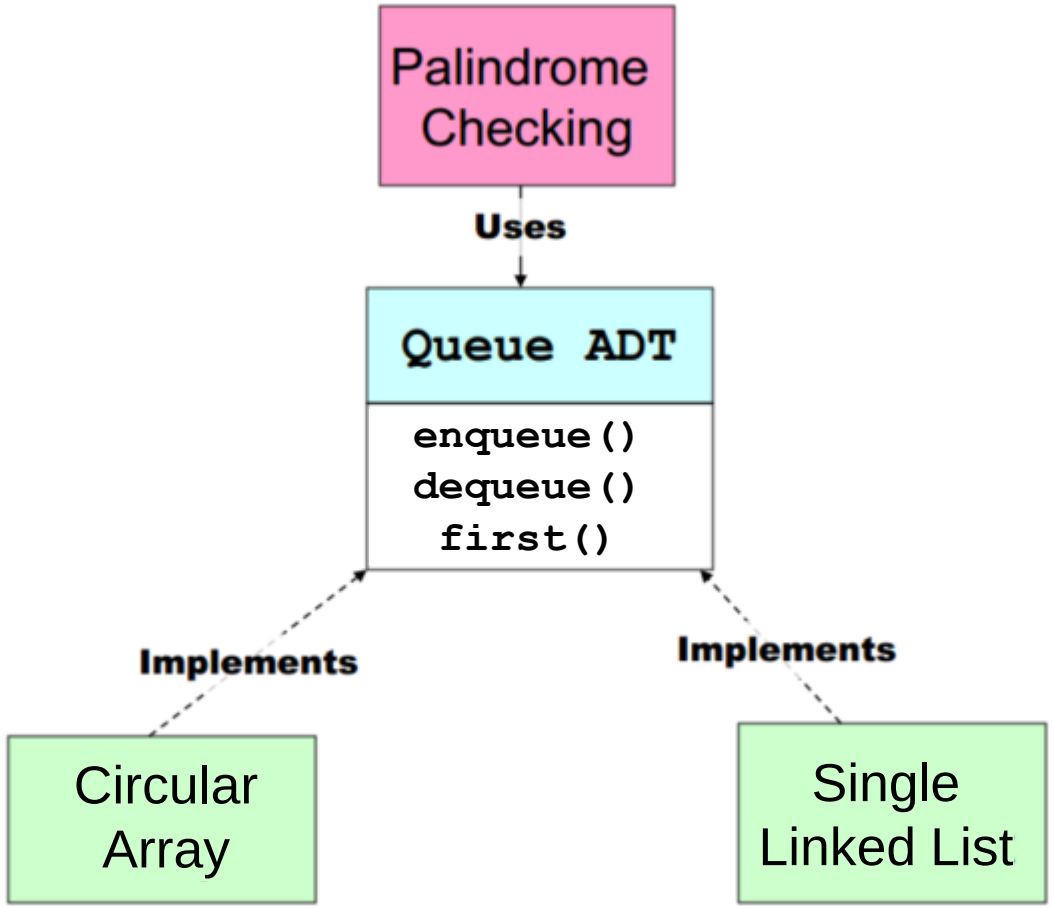
Queue applications

- Queue is used when things don’t have to be processed immediately, but have to be processed in FIFO order like.
  - When a resource is shared among multiple consumers: Disk scheduling, Printer Scheduling, Call Center phone systems ...
  - When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes: IO Buffers, pipes, file IO...
  - ... (*find yourself*)

Queue ADT

Summary

Summary



Applications

Queue API

Implementations