Stacks & Queues and Asymptotic Analysis

Yang Li
University of Washington
Autumn 2007

September 28, 2007

Today's Outline

- Admin: Office hours, etc.
- A Quick Review
- Assignment #1
- Stacks & Queues
- Asymptotic analysis

Office Hours, etc.

Yang Li	MW 2:00-3:00	CSE 212
Cam Thach Nguyen	TTh 9:30-10:20	CSE 218
Sean Shih-Yen Liu	Th 1:30-3:00	CSE 218
Sierra Michels-Slettvet	Th 3:30-4:20	CSE 3rd Floor Breakout

Or by appointment.

TODO : *Important!*

Subscribe to mailing lists if you haven't

A Quick Review

1. The Problem to solve

2. Abstract Data Types (ADT)
   - Objects + operations
   - E.g., a **stack** that allows push and pop
   - Use your intuition

3. Data Structures
   - A step-by-step description of how an ADT is realized in **pseudo code**
   - E.g., using an array or a list
   - **Proof by Induction** & **Asymptotic Analysis**

4. Programs
   - An actual implementation of an ADT based on particular data structures
   - E.g., java.util.Stack
   - Test the program with real data!

Project 1 – Sound Blaster!

Play your favorite song in reverse!

Aim:

1. Implement stack ADT two different ways
2. Use to reverse a sound file

Due: Mon, Oct 8,

Electronic: 11:00AM

Hardcopy: in lecture

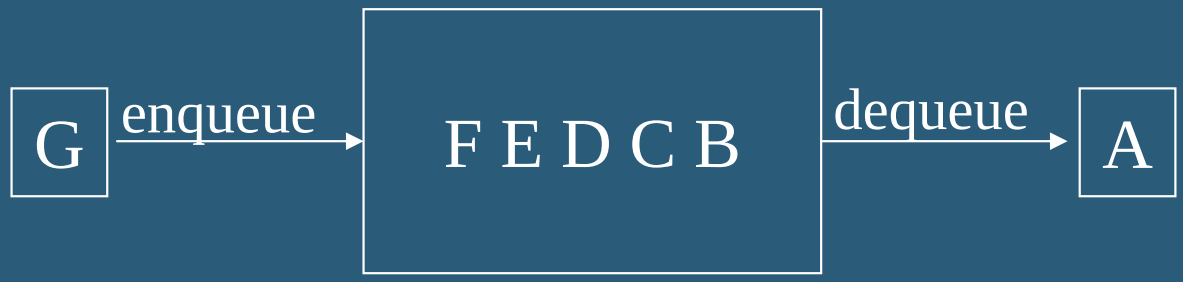
Today's Outline

- Admin: Office hours, etc.
- A Quick Review
- Assignment #1
- **Stacks & Queues**
- Asymptotic analysis

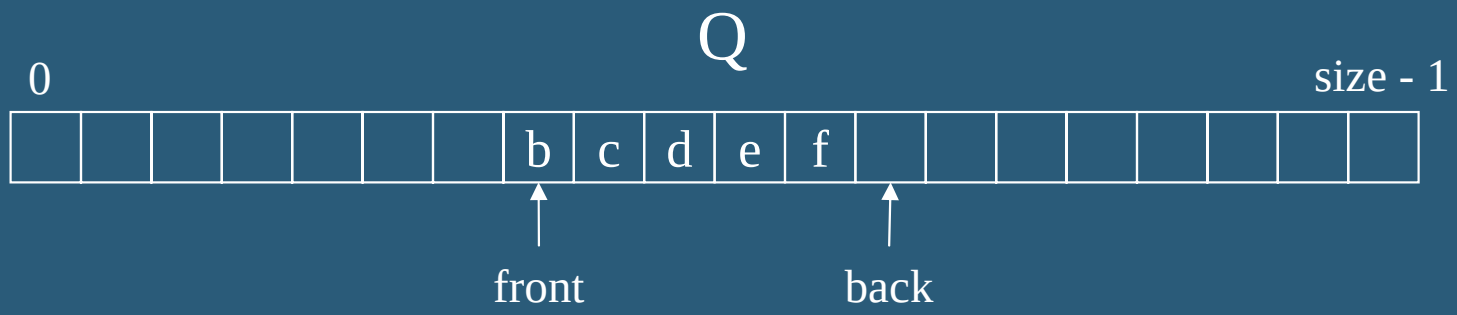
First Example: Queue ADT

Queue operations

- create
- destroy
- enqueue
- dequeue
- is_empty



Circular Array Queue Data Structure




```
enqueue(Object x) {
    Q[back] = x ;
    back = (back + 1) % size
    }
dequeue() {
    x = Q[front] ;
    front = (front + 1) % size;
    return x ;
    }
```

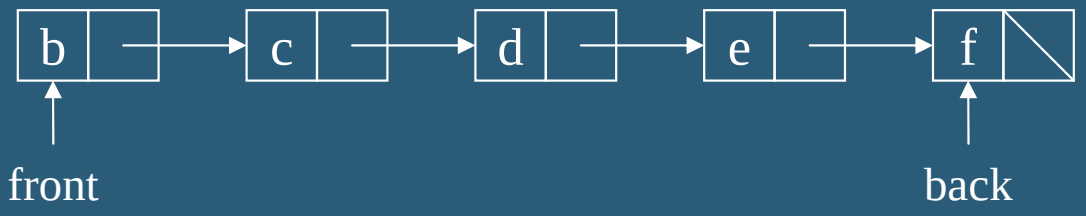
How test for empty list?

How to find K-th element in the queue?

What is complexity of these operations?

Limitations of this structure?

Linked List Queue Data Structure



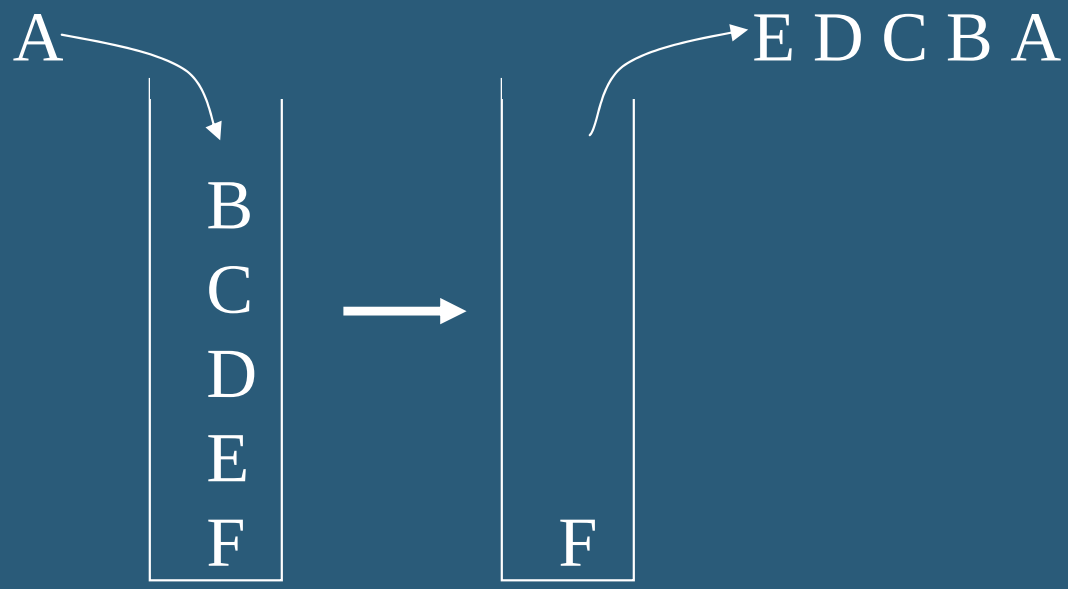

```
void enqueue(Object x) {
   if (is_empty())
        front = back = new Node(x)
   else
        back's next = new Node(x)
        back = back's next
}
bool is_empty() {
   return front == null
}
```

```
Object dequeue() {
   assert(!is_empty)
   return_data = front->data
   temp = front
   front = front->next
   delete temp
   return return_data
}
```

Circular Array vs. Linked List

Second Example: Stack ADT

- Stack operations
  - create
  - destroy
  - push
  - pop
  - top
  - is_empty



Stacks in Practice

- Function call stack
- Removing recursion
- Balancing symbols (parentheses)
- Evaluating Reverse Polish Notation

Comparing Two Algorithms

- Actual time & space used
  - Hours, minutes, seconds?
  - KB, MB, GB?
- Problems
  - You have to implement it
  - Hard & expensive to predict effectiveness as input changes

Today's Outline

- Admin: Office hours, etc.
- A Quick Review
- Assignment #1
- Stacks & Queues
- **Asymptotic analysis**

Big-O Analysis

Ignores “details”

Analysis of Algorithms

- Efficiency measure
  - › how long the program runs — time complexity
  - › how much memory it uses — space complexity
    - For today, we'll focus on time complexity only

- *Why analyze at all?*

Asymptotic Analysis

- Complexity as a function of input size n

  $T(n) = 4n + 5$

  $T(n) = 0.5\, n \log n - 2n + 7$

  $T(n) = 2^n + n^3 + 3n$

- *What happens as n grows?*

To Do

- Get working on **Project 1**
  - Due Wed, Oct 8
  - Ask questions!
- Sign up for 373 mailing list
- Mark errata in your textbook
- Continue reading chapters 1, 2 and 3