CSE 332: Data Abstractions

Ruth Anderson

Winter Quarter 2011

Lecture 1

Today's Outline

- Introductions
- Administrative Info
- What is this course about?
- Review: Queues and stacks

CSE 332 Course Staff!!

Instructor:

Ruth Anderson

Teaching Assistants:

- Sandra Fan
- Nathan Armstrong
- Gloria Guo
- Tim Jang

UNIVERSITY of VIRGINIA

Me (Ruth Anderson)

- Grad Student at UW (Programming Languages, Compilers, Parallel Computing)
- Taught Computer Science at the University of Virginia for 5 years
- Grad Student at UW (Educational Technology, Pen Computing)
- Defended my PhD in fall 2006
- Computing and the Developing World
- Recently taught cse142, cse143, cse326, cse373, compilers, programming languages, architecture, cse capstone – tech for resource-constrained environments

Today's Outline

- Introductions
- Administrative Info
- What is this course about?
- Review: Queues and stacks

Course Information

- **Instructor**: Ruth Anderson, CSE 360

  Office Hours: M & W 3:30-4:20, and by appointment, (rea@cs.washington.edu)
- **Text**: *Data Structures & Algorithm Analysis in Java*, (Mark Allen Weiss), 2nd Edition, 2007
- **Course Web page**: `http://www.cs.washington.edu/332`

Communication (1)

Instructors

Do NOT use until Tuesday!

- cse332-staff@cs.washington.edu
- (or our individual addresses)

Announcements

- cse332a_wi11@u.washington.edu
- (you are automatically subscribed @u)
- You are responsible for traffic on this list
- Will be archived on the course web page

Communication (2)

Discussion

- Go-Post Discussion board linked off course webpage
- Use your real name and picture

Feedback Always Welcome!

- Positive or negative
- See anonymous link on webpage

Course Materials

- All lecture and section materials will be posted
  - But they are visual aids, not always a complete description!
  - If you have to miss class, find out what you missed
- Textbook: Weiss 2nd Edition in Java
  - Good reading, but only responsible for lecture/section/hw topics
  - Will assign homework problems from it
- Core Java: A good Java reference (others also o.k.)
  - Same book recommended for CSE331
- Weeks 8-10 not in either book
  - We will use a set of lecture notes (provided on-line)

Course Work

- 8 written/typed homeworks (25%)
  - Due at beginning of class each Friday (not this week)
  - No late homeworks accepted
  - Lowest homework grade dropped
- 3 programming projects (with phases) (25%)
  - First phase of first project due next week
  - Use Java and Eclipse (see this week's section)
  - One 24-hour late-day for the quarter
  - Projects 2 and 3 will allow partners
- Midterm - Friday Feb 4 (20%)
- Final Exam - Tuesday March 15 (25%)

Approximate Grading

25% - Written Homework Assignments

25% - Programming Projects

20% - Midterm Exam

25% - Final Exam

5% - Best of the four items above.

Project/Homework Guides

On the website - note especially:

- Gilligan's Island rule applies.

Homeworks: Use pseudocode, not code.

- A human being is reading your homeworks.
- See website for pseudocode example.

Projects: correctness of code is only 40% of your grade!

- Spend time commenting your code as you write - it will help you be a better programmer.

Section

What happens there?

- Answer questions about current homework
- Previous homeworks returned and discussed
- Discuss the project (getting started, getting through it, answering questions)
- Finer points of Java
- Reinforce lecture material
- Occasionally introduce new material

Homework for Today!!

0) **Review Java & install Eclipse**
1) **Project #1:** (released by Wednesday) bring questions to section on Thursday
2) **Preliminary Survey**: fill out by evening of Thurs January 6th
3) **Information Sheet**: bring to lecture on or before Friday January 7th
4) **Reading** in Weiss (see handout)

Reading

- Reading in *Data Structures and Algorithm Analysis in Java*, 2nd Ed., 2007 by Weiss
- For this week:
  - › Chapter 1 – (review) Mathematics and Java
  - › Chapter 3 – (Project #1) Lists, Stacks, & Queues
  - › Chapter 2 – (Topic for Wednesday) Algorithm Analysis

Bring to Class on Friday:

- Name
- Email address
- Year (1,2,3,4,5)
- Hometown
- Interesting Fact or what I did over summer/winter break.

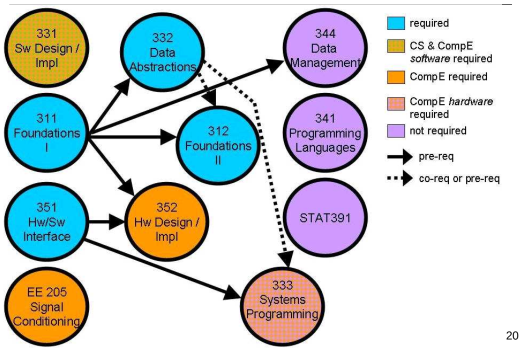
Today’s Outline

- Introductions
- Administrative Info
- What is this course about?
- Review: Queues and stacks

Last word about CSE 326

- CSE 332 is about *70%* of the material from CSE 326
  - › First 7 weeks or so, obviously cutting out some topics
    - and a little moving to CSE 312
  - › Timeless, essential stuff
- Biggest new topic: a serious treatment of programming with *multiple threads*
  - › For *parallelism*: To use multiple processors to finish sooner
  - › For *concurrency*: Allow properly synchronized access to shared resources

Where CSE 332 fits



Okay, so what is 332 about?

- Introduction to many of the basic data structures used in computer software:
  - › Understand the data structures and the **trade-offs** they make
  - › Rigorously **analyze** the algorithms that use them (math!)
  - › Learn how to **pick "the right data structure for the job"**
  - › More thorough and rigorous take on topics introduced in CSE 143 (plus more new topics)
- Practice design and analysis of data structures/algorithms
- Practice implementing and using these data structures by writing programs
- Experience the purposes (and headaches) of multithreading

Goals

- You will understand:
  - › what the tools are for storing and processing common data types
  - › which tools are appropriate for which need
- So that you will be able to:
  - › make good design choices as a developer, project manager, or system customer
  - › justify and communicate your design decisions

Data structures?

"Clever" ways to organize information in order to enable *efficient* computation over that information.

Data structures!

A data structure supports certain *operations*, each with a:

- › **Meaning**: what does the operation do/return?
- › **Performance**: how efficient is the operation?

Examples:

- › ***List*** with operations **`insert`** and **`delete`**
- › ***Stack*** with operations **`push`** and **`pop`**

Trade-offs

A data structure strives to provide many useful, efficient operations

But there are unavoidable trade-offs:

- Time vs. space
- One operation more efficient if another less efficient
- Generality vs. simplicity vs. performance

That is why there are many data structures and educated CSEers internalize their main trade-offs and techniques

- And recognize logarithmic < linear < quadratic < exponential

Terminology

- Abstract Data Type (ADT)
  - Mathematical description of a “thing” with set of operations on that “thing”
- Algorithm
  - A high level, language-independent description of a step-by-step process
- Data structure
  - A specific *organization of data* and family of algorithms for implementing an ADT
- Implementation of a data structure
  - A specific implementation in a specific language

Example: Stacks

- The ***Stack*** ADT supports operations:
  - **`isEmpty`**: initially true, later have there been same number of pops as pushes
  - **`push`**: takes an item
  - **`pop`**: raises an error if isEmpty, else returns most-recently pushed item not yet returned by a pop
  - … (Often some more operations)
- A Stack data structure could use a linked-list or an array or something else, and associated algorithms for the operations
- One implementation is in the library **`java.util.Stack`**

Why useful

The ***Stack*** ADT is a useful abstraction because:

- It arises all the time in programming (see text for more)
  - Recursive function calls
  - Balancing symbols (parentheses)
  - Evaluating postfix notation: 3 4 + 5 *
  - Clever: Infix ((3+4) * 5) to postfix conversion (see text)
- We can code up a reusable library
- We can communicate in high-level terms
  - “Use a stack and push numbers, popping for operators…”
  - Rather than, “create a linked list and add a node when…”

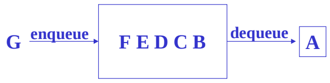
Today’s Outline

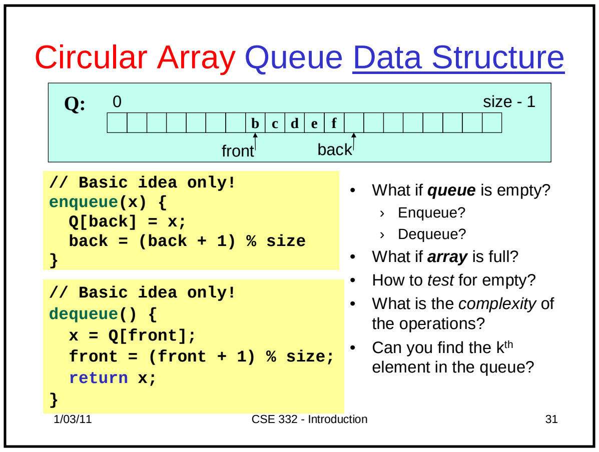
- Introductions
- Administrative Info
- What is this course about?
- Review: Queues and stacks

The Queue ADT

Queue Operations:

`create`
`destroy`
`enqueue`
`dequeue`
`is_empty`





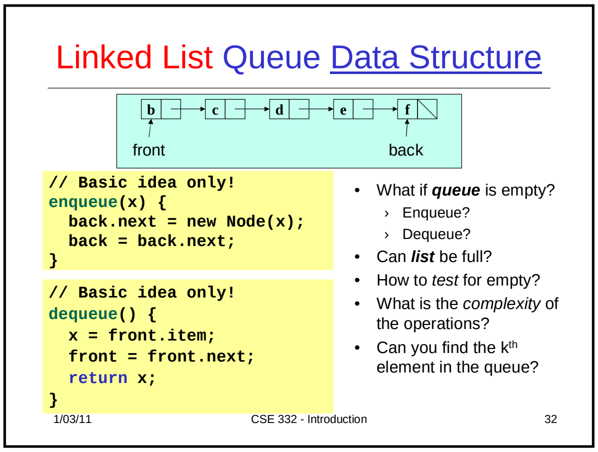
Circular Array Queue Data Structure

Q:
0
size - 1
b
c
d
e
f
front
back

```
// Basic idea only!
enqueue(x) {
  Q[back] = x;
  back = (back + 1) % size
}
```

```
// Basic idea only!
dequeue() {
  x = Q[front];
  front = (front + 1) % size;
  return x;
}
```

- What if ***queue*** is empty?
  - › Enqueue?
  - › Dequeue?
- What if ***array*** is full?
- How to *test* for empty?
- What is the *complexity* of the operations?
- Can you find the kth element in the queue?



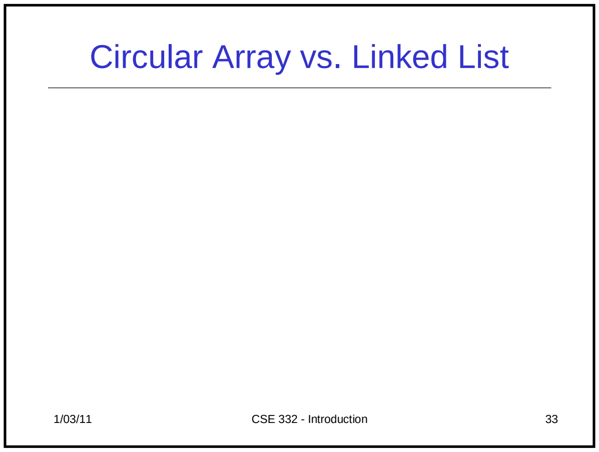
Linked List Queue Data Structure

b
c
d
e
f
front
back

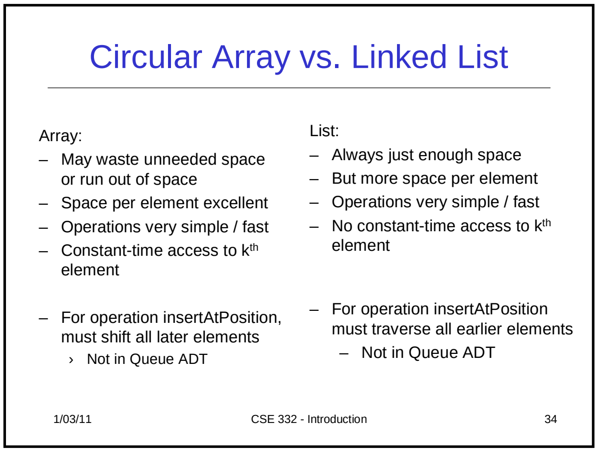
```
// Basic idea only!
enqueue(x) {
  back.next = new Node(x);
  back = back.next;
}
```

```
// Basic idea only!
dequeue() {
  x = front.item;
  front = front.next;
  return x;
}
```

- What if ***queue*** is empty?
  - › Enqueue?
  - › Dequeue?
- Can ***list*** be full?
- How to *test* for empty?
- What is the *complexity* of the operations?
- Can you find the kth element in the queue?



Circular Array vs. Linked List



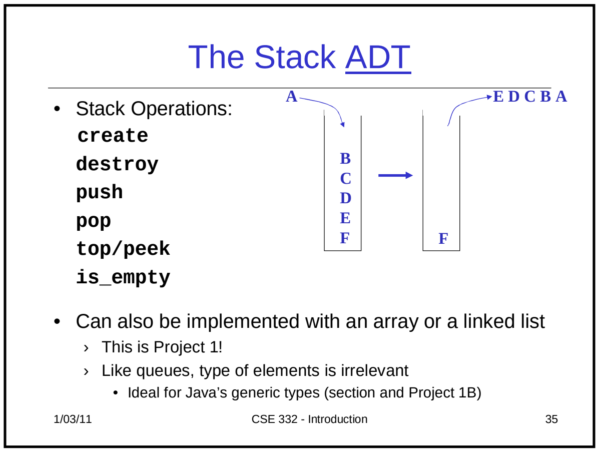
Circular Array vs. Linked List

Array:
- May waste unneeded space or run out of space
- Space per element excellent
- Operations very simple / fast
- Constant-time access to kth element

- For operation insertAtPosition, must shift all later elements
  - › Not in Queue ADT

List:
- Always just enough space
- But more space per element
- Operations very simple / fast
- No constant-time access to kth element

- For operation insertAtPosition must traverse all earlier elements
  - Not in Queue ADT

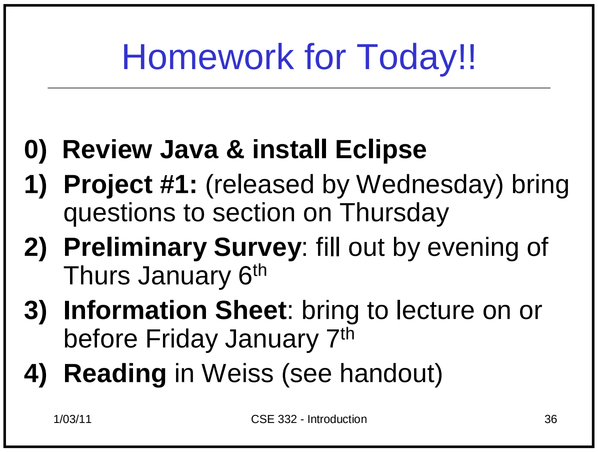


The Stack ADT

- Stack Operations:
  `create`
  `destroy`
  `push`
  `pop`
  `top/peek`
  `is_empty`

A
B
C
D
E
F
E D C B A
F

- Can also be implemented with an array or a linked list
  - › This is Project 1!
  - › Like queues, type of elements is irrelevant
    - Ideal for Java's generic types (section and Project 1B)



Homework for Today!!

0) **Review Java & install Eclipse**
1) **Project #1:** (released by Wednesday) bring questions to section on Thursday
2) **Preliminary Survey**: fill out by evening of Thurs January 6th
3) **Information Sheet**: bring to lecture on or before Friday January 7th
4) **Reading** in Weiss (see handout)