CSE 332: Data Structures and Parallelism

Fall 2022

Richard Anderson

Lecture 1: Introduction, Stacks and Queues

Welcome!

Fundamental data structures and algorithms for organizing and processing information

- "Classic" data structures / algorithms and how to analyze rigorously their efficiency and when to use them
- Queues, dictionaries, graphs, sorting, etc.
- Parallelism and concurrency (!)
- NP-Completeness (!!)

CSE 332 Team

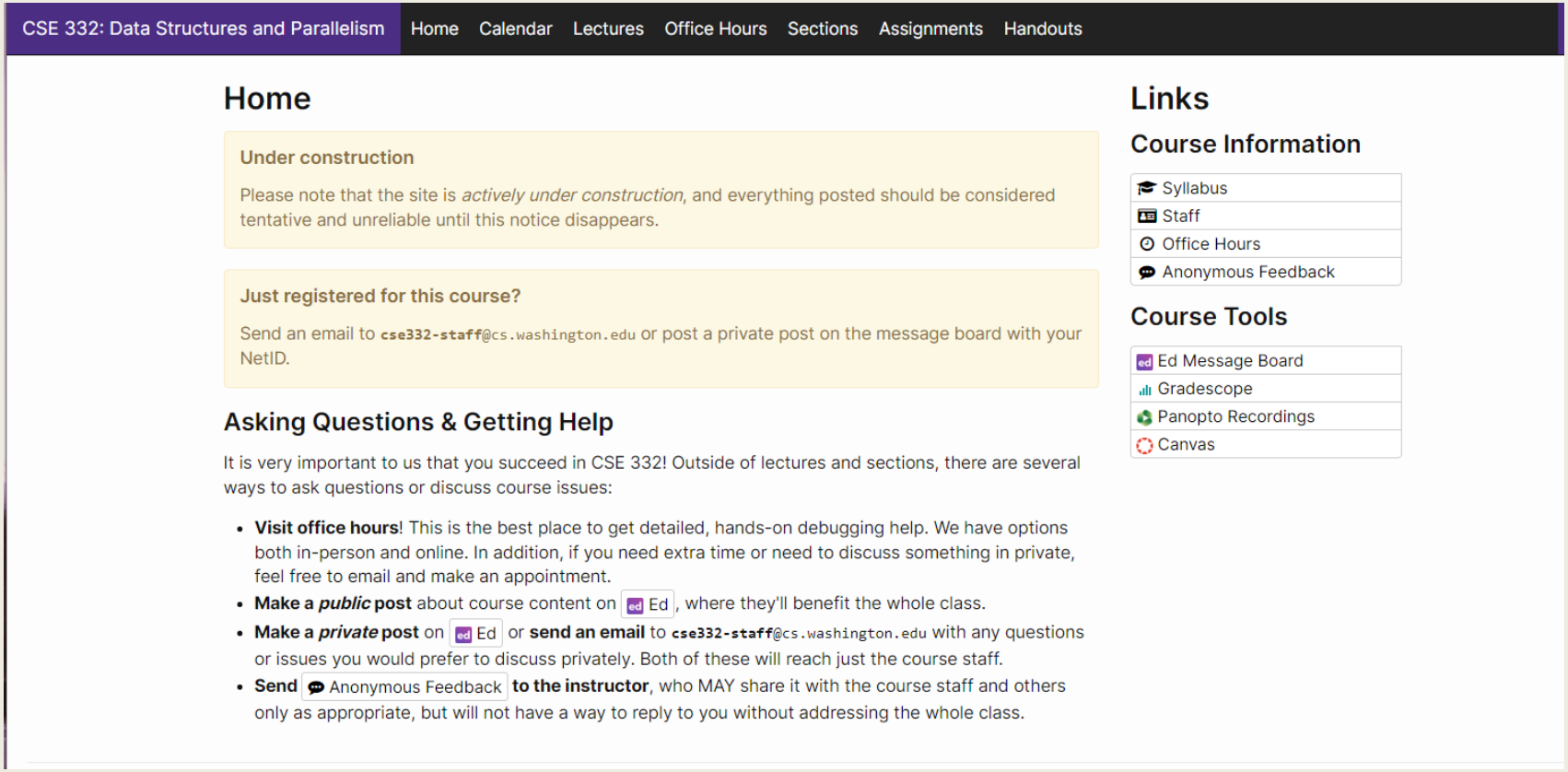
- Instructor: Richard Anderson, CSE2 344
- TAs:
  - Nathan Akkaraphab
  - Arya Krisna
  - Winston Jodjana
  - Sylvia Wang
  - Amanda Yuan
  - Allyson Mangus
  - Rahul Misal
  - Chandni Rajasekaran

Today's Outline

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

Course Information

http://www.cs.washington.edu/332



Course Information

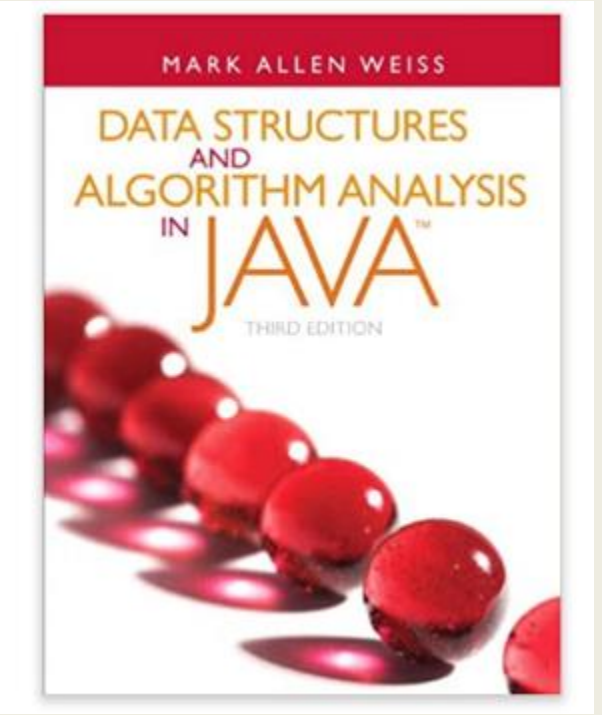
- Course Website

  http://www.cs.washington.edu/332

- Ed Message Board
- Gradescope
- Panopto Recordings

Text Book

- Data Structures and Algorithm Analysis in Java, 3rd Edition. Mark Weiss
  - UW Bookstore, $184.25
  - Amazon, $132.98 ($61.91 Used)
  - eTextbook, $74.99
  - Paperback, $58.00
  - eBay, $4.73 (2nd Edition)



Course meetings

- Lecture
  - Materials posted (sometimes afterwards), but take notes
  - Ask questions, focus on key ideas (rarely coding details)
- Section
  - Practice problems!
  - Answer Java/project/homework questions, etc.
  - Occasionally may introduce new material
  - An important part of the course (not optional)
- Office hours
  - Use them: *please visit us!*

Course Work

- ~20 Weekly individual homework exercises
- 3 programming projects (with phases)
  - Use Java, Gitlab
  - Single person projects (this is a change from previous quarters)
- Midterm and final exam
  - In class
    - *Midterm: Friday, November 4*
    - *Final: Thursday, December 15, 8:30 AM.*

Overall grading

Grading

- 25% - Written Homework Assignments
- 35% - Programming Assignments
- 15% - Midterm Exam (Nov 4)
- 25% - Final Exam (Dec 15, 8:30 AM)

Section

Meet on Thursdays

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, etc.
- Reinforce lecture material

Homework for Today!!

1. **Project #1:** Checkpoint 1, Oct 6 at 11:59 pm
2. **Exercise #1:** Due Monday, Oct 3 at 11:59 pm
3. **Review Java**
4. **Reading** in Weiss (see course web page)
   - (Topic for Project #1) Weiss 3.1-3.7 – Lists, Stacks, & Queues
   - (Friday) Weiss 2.1-2.4 –Algorithm Analysis
   - (Useful) Weiss 1.1-1.6 –Mathematics and Java

Today's Outline

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

Common tasks

- Many possible solutions
  - Choice of algorithm, data structures matters
  - What properties do we want?

Why should we care?

- Computers are getting faster
  - No need to optimize

- Libraries: experts have done it for you

Program Abstraction

Problem defn:

Algorithm:

Implementation:

Data Abstraction

Abstract Data Type (**ADT**):

Data Structure:

Implementation:

Trade offs: storing a set

Terminology

- Abstract Data Type (ADT)
  - Mathematical description of an object with set of operations on the object. Useful building block.
- Algorithm
  - A high level, language-independent, description of a step-by-step process.
- Data structure
  - A specific organization of the data to accompany algorithms for an abstract data type.
- Implementation of data structure
  - A specific implementation in a specific language.

Today's Outline

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

First Example: Queue ADT

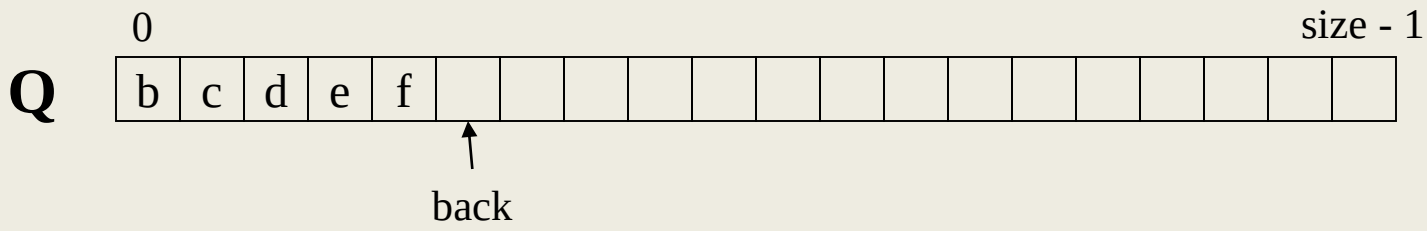
- FIFO: First In First Out
- Queue operations
  - create
  - destroy
  - enqueue
  - dequeue
  - is_empty

Queues in practice

- Print jobs
- File serving
- Phone calls and operators

(Later, we will consider “priority queues.”)

Array Queue Data Structure

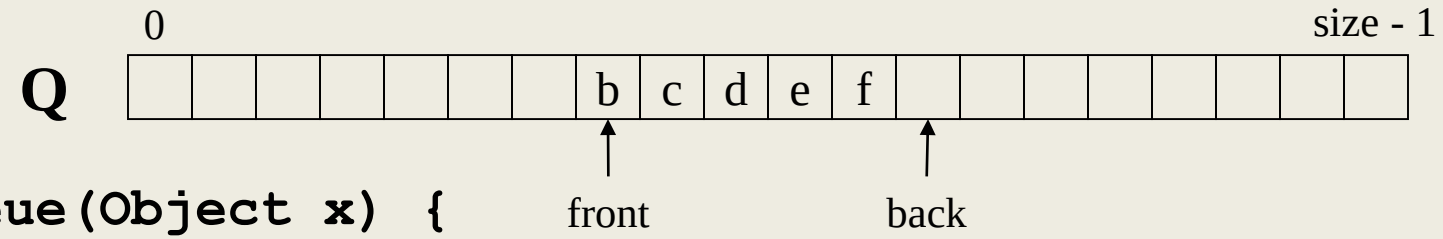



```
enqueue(Object x) {
    Q[back] = x
    back = (back + 1)
}

dequeue() {
    x = Q[0]
    shiftLeftOne()
    back = (back - 1)
    return x
}
```

What's missing in these functions?

How to find K-th element in the queue?

Circular Array Queue Data Structure



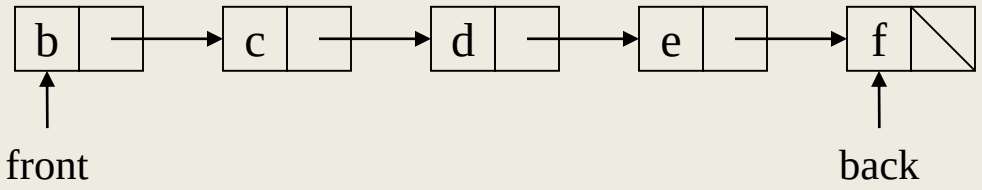

```
enqueue(Object x) {
    assert(!is_full())
    Q[back] = x
    back = (back + 1)
}

dequeue() {
    assert(!is_empty())
    x = Q[front]
    front = (front + 1)
    return x
}
```

How test for empty/full list?

How to find K-th element in the queue?

What to do when full?

Linked List Queue Data Structure




```
void enqueue(Object x) {
   if (is_empty())
        front = back = new Node(x)
   else {
        back->next = new Node(x)
        back = back->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
}
```

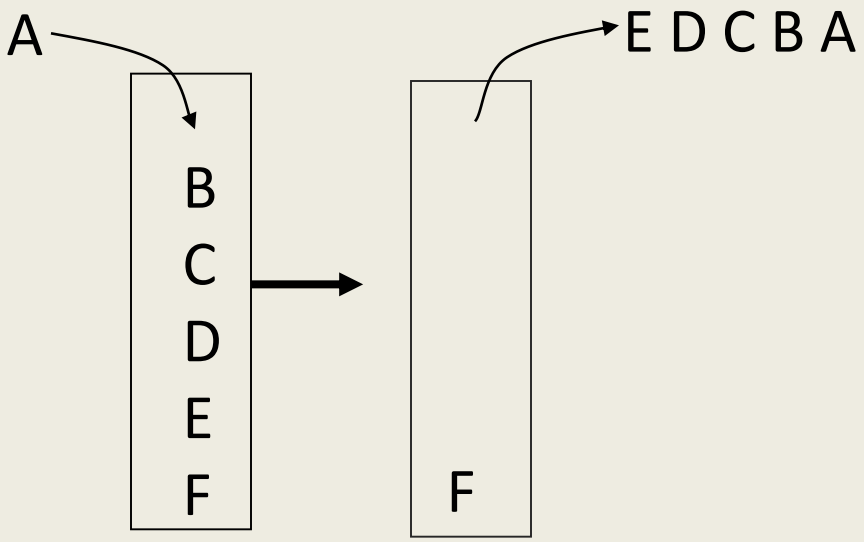
Circular Array vs. Linked List

- Advantages of circular array?

- Advantages of linked list?

Second Example: Stack ADT

- LIFO: Last In First Out
- Stack operations
  - create
  - destroy
  - push
  - pop
  - top
  - is_empty



Stacks in Practice

- Function call stack
- Removing recursion
- Balancing symbols (parentheses)
- Evaluating postfix or “reverse Polish” notation

Assigned readings

Reading in Weiss

- Chapter 1 – (Review) Mathematics and Java
- Chapter 2 – (Next lecture) Algorithm Analysis
- Chapter 3 – (Project #1) Lists, Stacks, & Queues