

# CSE 326: Data Structures

Dave Bacon

Winter Quarter 2007

Lecture 1

# Who?

---

## Dave Bacon

- › Research Assistant Professor in CSE



- › Sorry, no relation to Kevin Bacon

# CSE 326 Course Staff

---

**Section A Instructor:** Dave Bacon

**Section B Instructor:** Ruth Anderson

## **Teaching Assistants:**

- Ethan Phelps-Goodman (Section A)
- Jonah Cohen (Section B)
- David Wu (all over the place)

# Sections for Lecture A

---

- AA, Th 9:30-10:20 am MGH 251
- AB, Th 12:30-1:20 pm MEB 246
- There **WILL** be section this week!

# Today's Outline

---

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

# Course Information

---

- **Instructor:** Dave Bacon, CSE 460  
[dabacon@cs.washington.edu](mailto:dabacon@cs.washington.edu)
- **Text:** *Data Structures & Algorithm Analysis in Java*, (Mark Allen Weiss), 2<sup>nd</sup> Edition, 2007
- **Course Web page:**  
<http://www.cs.washington.edu/326>
- **Mailing List for course announcements:**
  - › [cse326-announce@cs.washington.edu](mailto:cse326-announce@cs.washington.edu)
  - › Subscribe *yourself* using web interface, see course web page

# Course Mechanics

---

- Written Homeworks (7 total)
  - › Due at the **start** of class on due date (Fridays)
  - › No late homeworks accepted
  - › Lowest homework grade dropped
- Programming Projects (3 total)
  - › In Java
  - › Turned in electronically (Wed eve) and on paper
  - › Once per quarter: use your “late day” for extra 24 hours – **Must email TA**
- Work in teams only on explicit team projects
  - › Appropriate *discussions* encouraged – see website

# Course Mechanics(2)

---

- Approximate Grading
  - 25% - Written Homework Assignments
  - 25% - Programming Projects
  - 20% - Midterm Exam (in class)
  - 25% - Final Exam (common – different time than listed in UW exam schedule, more coming on this)
  - 5% - Best of the four items above.

# Office Hours

---

- Dave Bacon, Tu 4:00-5:00, CSE 460
- Ruth Anderson, M 3:30-4:30, CSE 360
  
- Ethan Phelps-Goodman, Th 10:30-11:30, CSE 218
- Jonah Cohen, W 1:30-2:30, TBA
  
- David Wu, W 4:00-5:00 (in lab CSE 002/003), Th 3:30-4:30 (in CSE 218)

# Homework for Today!!

---

- 1) **Sign up for mailing list (immediately)**
- 2) **Project #1:** (read before section tomorrow)
- 3) **Preliminary Survey:** fill out by evening of Friday January 5<sup>th</sup> (link from webpage)
- 4) **Information Sheet:** bring to lecture on Friday January 5<sup>th</sup>
- 5) **Reading in Weiss** (see next slide)

# Reading

---

- Reading in *Data Structures and Algorithm Analysis in Java*, 2<sup>nd</sup> Ed., 2007 by Weiss
- For this week (on handout)
  - › Chapter 1 – (review) Mathematics and Java (pp. 1-25)
  - › Chapter 3 – (Project #1) Lists, Stacks, & Queues
    - Lists (pp. 57-81, heavy on Java, much of this should be review)
    - Stacks (pp. 82-83)
    - Applications of Stacks (pp. 83-91, sections on “Postfix Expressions” and “Infix to Postfix Conversion” can be skipped, but read “Method Calls”)
    - Queues (pp. 91-95)
  - › Chapter 2 – (Topic for Friday) Algorithm Analysis (pp. 29-50)

# Bring to Class on Friday:

---

- Name **Dave Bacon**
- Email address **dabacon@**
- Year (1,2,3,4) **00 CS.**
- Major **Physics & Literature**
- Hometown **Yreka, CA**
- Interesting Fact or what I did over winter break. **Planned wedding**



# Today's Outline

---

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

# Class Overview

---

Introduction to many of the basic data structures used in computer software

- › Be exposed to a variety of data structures
- › Know when to use them
- › Practice mathematical techniques for analyzing the algorithms that use them
- › Practice implementing and using them by writing programs

Goal:

be able to make good design choices as a developer, project manager, or system customer

# Goals

---

## Get a job at Google?

- › Interview question: Design and describe a system/application that will most efficiently produce a report of the top 1 million Google search requests. These are the particulars.
  - \* You are given 12 servers to work with. They are all dual-processor machines with 4Gb of RAM, 4x400GB hard drives and networked together. (Basically, nothing more than high-end PC's)
  - \* The log data has already been cleaned for you. It consists of 100 Billion log lines, broken down into 12 320 GB files of 40-byte search terms per line.

# Goals

---

## Master of the programming universe?

- › “I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships.”

-Linus Torvalds, 2006

# Data Structures

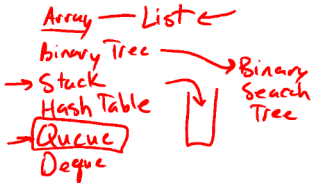
---

“Clever” ways to organize information in order to enable efficient computation

- › What do we mean by clever?
- › What do we mean by efficient?

# Data Structures?

---



# Picking the best Data Structure for the job

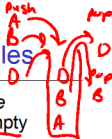
---


- The data structure you pick needs to *support* the operations you need
- Ideally it supports the operations you will use most often in an efficient manner
- Examples of operations:
  - › List ADT with operations insert and delete
  - › Stack ADT with operations push and pop

# Terminology

- Abstract Data Type (ADT) *Queue 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 family of algorithms for implementing an abstract data type. *array implement Queue*
- Implementation of data structure *JAVA*
  - › A specific implementation in a specific language

# Terminology examples



- A stack is an *abstract data type* supporting push, pop and isEmpty operations
- A stack *data structure* could use an array, a linked list, or anything that can hold data 
- One stack *implementation* is found in java.util.Stack



# Concepts vs. Mechanisms

- Abstract
- Pseudocode
- Algorithm
  - › A sequence of high-level, language independent operations, which may act upon an abstracted view of data.
- Abstract Data Type (ADT)
  - › A mathematical description of an object and the set of operations on the object.
- Concrete
- Specific programming language
- Program
  - › A sequence of operations in a specific programming language, which may act upon real data in the form of numbers, images, sound, etc.
- Data structure
  - › A specific way in which a program's data is represented, which reflects the programmer's design choices/goals.

# Why So Many Data Structures?

---

Ideal data structure:

“fast”, “elegant”, memory efficient

Generates tensions:

- › time vs. space
- › performance vs. elegance
- › generality vs. simplicity
- › one operation's performance vs. another's

*The study of data structures is the study of tradeoffs. That's why we have so many of them!*

# Today's Outline

---

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

# Priority

## First Example: FIFO Queue ADT

- Queue operations

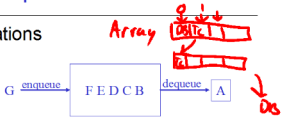
create ←

destroy ←

enqueue

dequeue

is\_empty

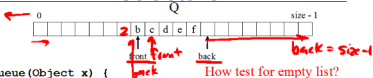


FIFO: First in, first out



# Circular Array Queue Data Structure

Array size



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

move back over

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

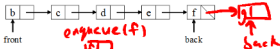
How test for empty list?

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->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

---

space    array    fixed size  
          LL     variable size ←

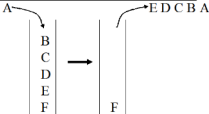


allocation time

# Second Example: Stack ADT

- Stack operations

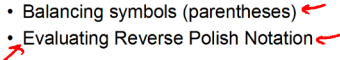
- › create
- › destroy
- › push ←
- › pop →
- › top ←
- › is\_empty



LI  
S  
+  
FI  
S  
+

# Stacks in Practice

---

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

# Homework for Today!!

---

- 1) **Sign up for mailing list (immediately)**
- 2) **Project #1:** (read before section tomorrow)
- 3) **Preliminary Survey**: fill out by evening of Friday January 5<sup>th</sup>
- 4) **Information Sheet**: bring to lecture on Friday January 5<sup>th</sup>
- 5) **Reading in Weiss** (see next slide)