CSE 326b: Data Structures

Hal Perkins
Winter 2008
Lecture 1

CSE 326 Crew

- Hal Perkins
- Kathleen Tuite
- Ray Smith

CSE 326 Crew

- Hal Perkins
- Kathleen Tuite
- Ray Smith

- And *You*!

Today's Outline

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

Course Information

- **Instructor**: Hal Perkins, CSE 548
  perkins@cs.washington.edu
- **Text**: *Data Structures & Algorithm Analysis in Java*, (Mark Allen Weiss), 1999
- **Web page**: **`www.cs.washington.edu/education/...courses/326/08wi/b`**
- **Mailing Lists:** You are automatically subscribed via your *UW netid* – be sure you read mail sent there
- **Discussion list: link on course home page**

CSE 326 A & B

- There are two sections of CSE 326 this quarter
  - Lecture A: 12:30 MWF, Ladner
  - Lecture B: 2:30 MWF, Perkins
- Both versions of the course will cover the same basic material, but detailed content, assignments, and exams are likely to be different
  - Corollary: You must attend one lecture and one of the sections associated with that lecture; you can't mix-n-match

Course Mechanics

- Written homeworks (6-7 total)
  - Due at the start of class on due date (typically Friday)
  - No late homeworks accepted
- Programming homeworks (3-4 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

  20% - Written Homework Assignments

  25% - Programming Assignments

  20% - Midterm Exam (in class)

  25% - Final Exam

  10% - Best of the four items above.

Homework for Today!!

1) **Information Sheet**: bring to lecture on Wednesday, Jan. 9
2) **Reading** in Weiss

(Details follow)

Bring to Class on Wednesday:

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

Reading

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

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; understand tradeoffs
- 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

"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

Goals

"Show me your flowcharts and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowcharts; they'll be obvious."

Fred Brooks, 1975

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?

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)
  - 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.
- Implementation of data structure
  - 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 java.util.Stack; another is java.util.LinkedList

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!

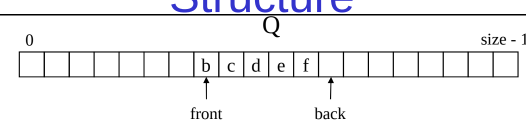
Today's Outline

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

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 ;
   }
```

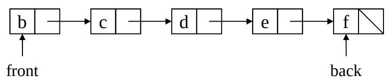
How test for empty list?

How to find K-th element in the queue?

What is complexity of these operations?

Limitations of this structure?

Linked List Queue Data Structure



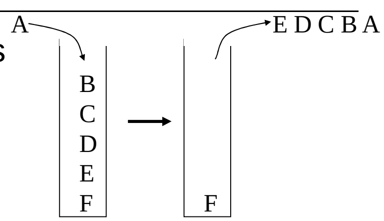

```
void enqueue(Object x) {
  if (is_empty())
      front = back = new Node(x)
  else
      back->next = new Node(x)
      back = back->next
}
bool is_empty() {
  return front == null
}
```

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


Circular Array vs. Linked List

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

Homework for Today!!

1) **Information Sheet**: bring to lecture on Wednesday
2) **Reading** in Weiss