Welcome!

We have 10 weeks to learn 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 (!)

Today's Outline

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

CSE 332 Course Staff!!

Instructor:
Ruth Anderson

Teaching Assistants:
• Daniel Jones
• Hye In Kim
• Jacob Gile
• David Swanson

Me (Ruth Anderson)

• Grad Student at UW in Programming Languages, Compilers, Parallel Computing
• Taught Computer Science at the University of Virginia for 5 years
• Grad Student at UW: PhD in Educational Technology, Pen Computing
• Current Research: Computing and the Developing World
• Recently Taught: majors and non-majors data structures, architecture, compilers, programming languages, cse143, Designing Technology 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 3:30-4:30pm, Tu 11-11:50am, and by appointment, (rea@cs.washington.edu)
• **Text:** *Data Structures & Algorithm Analysis in Java*, (Mark Allen Weiss), 3rd edition, 2012
• **Course Web page:** [http://www.cs.washington.edu/332](http://www.cs.washington.edu/332)

Communication

• Course email list: cse332a_wi13@u
  • Students and staff already subscribed
  • You must get announcements sent there
  • Fairly low traffic
• Course staff: cse332-staff@cs plus individual emails
• Discussion board
  • For appropriate discussions; staff will monitor
  • Optional, won’t use for important announcements
• Anonymous feedback link
  • For good and bad; if you don’t tell me, I don’t know

Course meetings

• Lecture (Ruth)
  • Materials posted (sometimes afterwards), but take notes
  • Ask questions, focus on key ideas (rarely coding details)
• Section (Hye In and Daniel)
  • Often focus on software (Java features, tools, project issues)
  • Reinforce key issues from lecture
  • Occasionally introduce new material
  • Answer homework questions, etc.
  • An important part of the course (not optional)
• Office hours
  • Use them: please visit me
  • Ideally not just for homework questions (but that’s great too)

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, find out what you missed
• Textbook: Weiss 3rd Edition in Java
  • Good read, but only responsible for lecture/section/hw topics
  • Will assign homework problems from it
  • 3rd edition improves on 2nd, but we’ll support the 2nd
• Core Java book: A good Java reference (there may be others)
  • Don’t struggle Googling for features you don’t understand
  • Same book recommended for CSE331
• Parallelism / concurrency units in separate free resources designed for 332

Course Work

• 8 written/typed homeworks (25%)
  • Due at beginning of class each Friday (not this week)
  • No late homeworks accepted
• 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 - (20%)
• Final Exam - Tuesday March 19 (25%)

Collaboration & Academic Integrity

• Read the course policy very carefully
  • Explains quite clearly how you can and cannot get/provide help on homework and projects
  • Gilligan’s Island rule applies.
• Always proactively explain any unconventional action on your part
  • When it happens, (not when asked)
• I offer great trust but with little sympathy for violations
• Honest work is the most important feature of a university
Unsolicited advice

- Get to class on time!
- Learn this stuff
  - You need it for so many later classes/jobs anyway
  - Falling behind only makes more work for you
- Have fun
  - So much easier to be motivated and learn

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 10th
3) Information Sheet: bring to lecture on or before Friday January 11th
4) Reading in Weiss (see handout)

Reading

- Reading in Data Structures and Algorithm Analysis in Java, 3rd Ed., 2012 by Weiss
- For this week:
  - (Topic for Project #1) Weiss 3.1-3.7 –Lists, Stacks, & Queues
  - (Wed) Weiss 1.1-1.6 –Mathematics and Java
  - (Fri) Weiss 2.1-2.4 –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

Data Structures + Threads

- About 70% of the course is a “classic data-structures course”
  - Timeless, essential stuff
  - Core data structures and algorithms that underlie most software
  - How to analyze algorithms
- Plus a serious first treatment of programming with multiple threads
  - For parallelism: Use multiple processors to finish sooner
  - For concurrency: Correct access to shared resources
  - Will make many connections to the classic material
Where 332 fits

- Most common pre-req for 400-level courses
- Essential stuff for many internships too!

What 332 is about

- Deeply understand the basic structures used in all software
  - Understand the data structures and their trade-offs
  - Rigorously analyze the algorithms that use them (math!)
  - Learn how to pick “the right thing for the job”
- Experience the purposes and headaches of multithreading
- Practice design, analysis, and implementation
  - The elegant interplay of “theory” and “engineering” at the core of computer science

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

Views on this course

- Prof. Steve Seitz (graphics):
  - 100-level and some 300-level courses teach how to do stuff
  - 332 teaches really cool ways to do stuff
  - 400 level courses teach how to do really cool stuff
- Prof. James Fogarty (HCI):
  - Computers are fricking insane
  - Raw power can enable bad solutions to many problems
  - This course is about how to attack non-trivial problems
  - Problems where it actually matters how you do it

Views on this course

- Prof. Dan Grossman (prog. langs.):
  - Three years from now this course will seem like it was a waste of your time because you can’t imagine not “just knowing” every main concept in it
  - Key abstractions computer scientists and engineers use almost every day
  - A big piece of what separates us from others

Views on this course

- This is the class where you begin to think like a computer scientist
  - You stop thinking in Java or C++ code
  - You start thinking that this is a hashtable problem, a stack problem, etc.
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]
front back

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

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

Linked List Queue Data Structure

// 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;
}

Circular Array vs. Linked List
Array:
– May waste unneeded 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

- 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 10th
3) Information Sheet: bring to lecture on or before Friday January 11th
4) Reading in Weiss (see handout)