CSE 332: Data Abstractions

Course Victory Lap

Ruth Anderson

Winter 2015

Today

- Rest-of-course logistics: exam, etc.
- Review of main course themes
- Course evaluations – Please don’t forget to fill these out!
  - Thoughtful and constructive feedback deeply appreciated
  - (Including what you liked)

Final Exam

- Next Wednesday, March 18, **12:30-2:20pm**
- In Kane 220

- Intention is to test a subset of the topics in sorting, graphs, parallelism, concurrency, amortization, NP-complete
  - In other words, “stuff not covered by the midterm”
  - But as always the course topics build on earlier ones, especially algorithm analysis
- May need to read and write Java, among other things
- Topics and Sample exams listed on course web site

Victory Lap

A victory lap is an extra trip around the track

- By the exhausted victors (that's us) ☺

Review course goals

- Slides from Lecture 1
- What makes CSE332 special

Thank you!

Big thank-you to your TAs!

- Matthew Gillette
- Daphna Khen
- Conrad Nied
- Nicholas Shahan
- Ian Turner
- Jack Warren

Now a few slides from Lecture 1

- Hopefully they make more sense now
- Hopefully we succeeded

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

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.

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

Parallelism and Concurrency

Hopefully this seemed to “fit”, because:

- Work, span, Amdahl’s Law are about asymptotics
- Fork-join is great for divide-and-conquer
- Sequential cutoffs are like quicksort/insertion-sort cutoffs
- ADTs shared by multiple threads need critical sections
- … (several more examples)

Other main thesis: emphasize parallelism vs. concurrency distinction

  – Not always widely appreciated
  – Often mixed in practice

Last slide

What do you think was good about 332?

What could be improved?

Hopefully I will see you in other courses in CSE but if not, please stay in touch!