CSE332: Data Abstractions
Lecture 26: Course Victory Lap

Dan Grossman
Spring 2012

Today

• Rest-of-course logistics: exam, etc.
• Review of main course themes
• Some thoughts on “data structures and threads” together
• Some time for questions and discussion
• Course evaluations
  – Thoughtful and constructive feedback deeply appreciated
  – (Including what you liked)

Final Exam

As also indicated on the web page:

• Next Tuesday, 2:30-4:20
• Intention is to test a subset of the topics in sorting, graphs, parallelism, concurrency, amortization
  – 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

Grading Schedule

Needs grading:
  – Homework 8 [Stanley]
  – Project 3 [Tyler]
  – Final exam [Dan]

Will let you know when grading is done
  – Encourage you to pick up your homework, exam
  – But Dan will be out of town June 9-20
    • Exams at department front desk during this time?

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
  – Section covers essential software topics and complements lecture: indispensable in my opinion
  – Lots of grading in CSE332: “free response” and “open design” better for students, harder for TAs
Thank you!

And huge thank you to all of you
– Great attitude
– Extraordinarily good class attendance and questions
– Occasionally laughed at stuff 😊

Now five slides, completely unedited, 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 is 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

• Be able to make good design choices as a developer, project manager, etc.
  – Reason in terms of the general abstractions that come up in all non-trivial software (and many non-software) systems
• Be able to justify and communicate your design decisions

Dan’s take:
3 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

Data structures

(Often highly non-obvious) ways to organize information to enable efficient computation over that information

– Key goal over the next week is introducing asymptotic analysis to precisely and generally describe efficient use of time and space

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

Now thoughts on teaching parallelism and concurrency in this class:

- Something I have vigorously advocated personally
- And it seems to be working

Background

- “Old” data structures course taught more data structures and algorithms
  - Splay trees, leftist heaps, skew heaps, disjoint-set, network flow, …
- Threads are way more important than they used to be
- “Data structures” is not what most faculty would think of for the “best place to fit it”…

The fit

Hopefully it did not seem to odd to you, 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 need critical sections
- Queues motivate passive waiting
- … (several more examples)

Other main thesis: emphasize parallelism vs. concurrency distinction

- Not always widely appreciated
- Often mixed in practice

Seems to work

- CSE332 instructors at UW
- Parts of materials picked up by 8 other schools so far
  - So please keep reporting typos, especially in reading notes
- A paper at SIGCSE2012 (main CS Education Conference)

Last slide

What do you think was good about 332?

What could be improved?

And:

Don’t be a stranger: let me know how the rest of your time in CSE (and beyond!) goes… I really do like to know.