CSE373: Data Structures & Algorithms
Lecture 23: Course Victory Lap

Dan Grossman

Fall 2013

Today

- Finish parallel-program analysis
- Rest-of-course logistics: exam, etc.
- Review of main course themes
- 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
- Cumulative but topics post-midterm-2 worth about 2/3 of the points
- See information on course web-page
- Not unlike the midterms in style, structure, etc.
- Tough-but-fair exams are the most equitable approach
  - And/but 110 minutes will make a big difference

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 CSE373 special

Thank you!

Big thank-you to your TAs

- Amazingly cohesive “big team”
- Prompt grading and question-answering
- Optional TA sessions weren’t optional for them!





Thank you!

And huge thank you to all of **you**

- Great attitude
- *Good class attendance and questions for the largest-ever (?) CSE373*
  - *Thoughts on how to "make it feel smaller" appreciated*
- Occasionally laughed at stuff ☺

Now three slides, completely unedited, from Lecture 1

- Hopefully they make more sense now
- Hopefully we succeeded

Data Structures

- Introduction to Algorithm Analysis
- Lists, Stacks, Queues
- Trees, Hashing, Dictionaries
- Heaps, Priority Queues
- Sorting
- Disjoint Sets
- Graph Algorithms
- *May have time for other brief exposure to topics, maybe parallelism*

What 373 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"
  - More thorough and rigorous take on topics introduced in CSE143 (plus more new topics)

- Practice design, analysis, and implementation
  - The elegant interplay of "theory" and "engineering" at the core of computer science

- More programming experience (as a way to learn)

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:

- Key abstractions used almost every day in just about anything related to computing and software
- It is a vocabulary you are likely to internalize permanently

Last slide

What do you think was good about 373?

What could be improved?

Advice:

- ***Make the most of your time at UW and beyond***
- ***You have learned the key ideas for organizing data, a skill that far transcends computer science***