CSE332: Data Abstractions

Lecture 24: Course Wrap-Up

Ruth Anderson
Winter 2011

Final Exam

Tuesday, March 15, 2:30-4:20

- Intention is to test topics in sorting, graphs, parallelism, concurrency
  - stuff not covered by the midterm"
  - But as always the course topics build on earlier ones
- You will need to read and write Java, among other things

Four slides from Lecture 1

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 (new!)

Four slides from Lecture 1

- Introduction to many (not all) of the basic data structures used in computer software
  - Understand the data structures and the trade-offs they make
  - 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 143
    - And more
- Practice design and analysis of data structures / algorithms
- Practice implementing and using these data structures by writing programs
- Experience the purposes and headaches of multithreading

Four slides from Lecture 1

- To 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
- To be able to justify and communicate your design decisions

This course is key!

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 "a *computer scientist* knows"

Four slides from Lecture 1

(Often highly *non-obvious*) ways to organize information in order 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

Topics: Data structures + Threads

326 & 332

Big-Oh, Algorithm Analysis
Binary Heaps (Priority Qs)
AVL Trees
B Trees
Hashing
Sorting
Graph Traversals
Topological Sort
Shortest Paths
Minimum Spanning Trees

Topics: Data structures + Threads

326 & 332

Big-Oh, Algorithm Analysis
Binary Heaps (Priority Qs)
AVL Trees
B Trees
Hashing
Sorting
Graph Traversals
Topological Sort
Shortest Paths
Minimum Spanning Trees

Removed from 326

D-heaps
Leftist heaps
Skew heaps
Binomial queues
Splay trees
Disjoint sets

Topics: Data structures + Threads

326 & 332

Big-Oh, Algorithm Analysis
Binary Heaps (Priority Qs)
AVL Trees
B Trees
Hashing
Sorting
Graph Traversals
Topological Sort
Shortest Paths
Minimum Spanning Trees

Added to 332

Multithreading Basics (1)
Fork-Join Parallelism (3)
- Using Java library
- Analysis: **T1** and **T∞**
- Amdahl's Law
- Reductions, Prefix, Sorting

Concurrency (4)
- Races, deadlocks
- Locks (mostly)
- Condition variables (a bit)
- Programming guidelines (!)

I want feedback on what worked/did not!

- "Quiz" on wednesday
- Course Evals