Announcements

- HW4 feedback out this afternoon.

CSE373: Data Structures & Algorithms
Course Victory Lap

Hunter Zahn

Summer 2016

Today

- Rest-of-course logistics: exam, etc.

- Review of main course themes

- Course evaluations (online)
  - Thoughtful and constructive feedback deeply appreciated
  - (Including what you liked)

Final Exam

As also indicated on the web page:

- Friday, in class
- Cumulative but topics post-midterm worth more points
- Not unlike the midterms in style, structure, etc.
- Tough-but-fair exams are the most equitable approach

Post Midterm Topics

- Sorting
- Graphs
- Minimum Spanning Trees
- Parallelism And Concurrency
- Problem Solving
- Perserving Abstractions

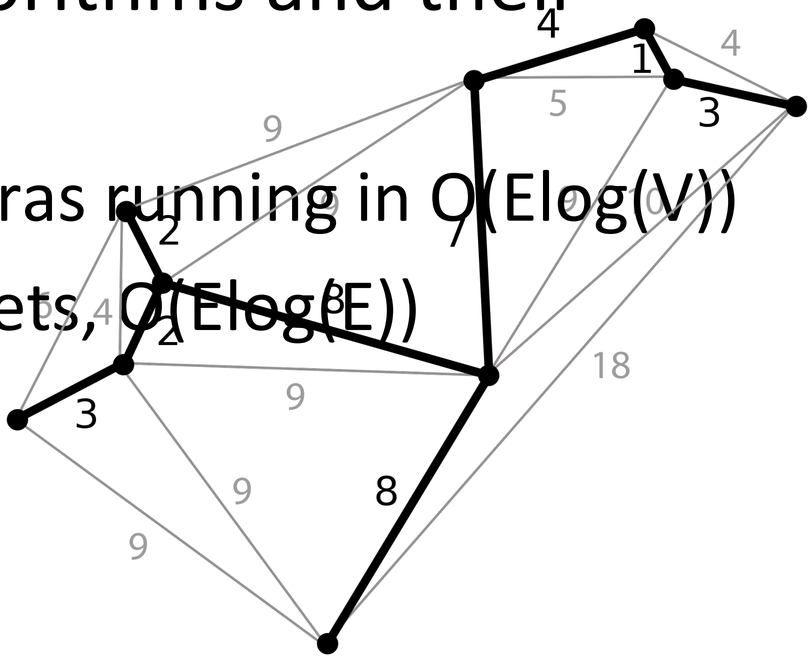
Sorting

- Properties and runtimes of varying sorting algorithms
  - How much extra space required? In place?
  - Partially sorted? Reverse order?
  - Homework 5 is great to help prepare
- Given a following block mystery psuedocode, determine which sorting algorithm it is
- Non-comparison based sorts (bucket sort and radix sort)

Minimum Spanning Trees

- How to build from a graph

- Kruscal’s vs Primm’s algorithms and their runtimes
  - **Primm’s**: Modified Dijkstras running in O(Elog(V))
  - **Kruscal’s**: Using Disjoin sets, O(Elog(E))



Parallelism and Concurrency

- What is the difference?

- Maps and Reductions
  - **Map**: Applying a function to all of the values in a collection, resulting in a identical length collection.
    - Map([1,2,3,4,5,6], +1) = [2,3,4,5,6,7]

  - **Reduction**: Applying a function to a collection to reduce it to a single value.
    - Reduce([1,2,3,4,5,6], leftmost even number) = 2

- **Work**: How long it takes **1 processor** to execute a sequentially execute a block of code
- **Span**: how long it takes **infinite processors** to execute a block of code

Preserving Abstraction

- Copy in vs. Copy out
  - To protect internal structures from being modified by clients

- Private and Final fields
  - How do they impact immutability?

- Hiding unnecessary information from Clients
  - **For example**: A client generally does not need to know that your Disjoint sets are being represented as Uptrees
  - **Another example**: Java's HashTable does not tell you what hashcode they are using. Why?

Problem Solving

- Lots of different metrics for determine which solution is "best"
- For example, solve the following problem first by optimizing for time, then by optimizing solely for space:

Given a list of Strings representing States of birth for students at a highschool. For the most common State, output all the students who were born there.

Victory Lap

A victory lap is an extra lap 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 TA's:

Alon

Dan

Lilian

Thank you!

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

- Great attitude
- *Great class attendance and questions from a smaller summer quarter class*
- Occasionally laughed at my bad jokes

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

Hunter'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

Where next?

Hopefully cse373 will not be your last exposure to computer science. There are lots of other awesome computer science courses for non-cse majors!

CSE 154: Web Programming

Developing Websites and client and server side software

CSE 374: Intermediate programming Concepts and Tools

Concepts of lower-level programming (C/C++) and explicit memory management

CSE 417: Algorithms and Computational Complexity

NP Complete problems, undecidable problems, graph theory and complexity

CSE 415: Introduction to Artificial Intelligence

Knowledge representation, logical and probabilistic reasoning, learning, language understanding, intro to game theory

So many other resources outside of UW

Coursera Python Coursehttps://www.coursera.org/specializations/python

Machine learning course:
https://www.coursera.org/learn/machine-learning

Computer Security:
https://www.coursera.org/course/security

Computational Neuroscience:
https://www.coursera.org/course/compneuro

Principles of Computing (Great next step for math lovers):
https://www.coursera.org/course/principlescomputing1

Learn a new Language!

- **Haskell**: http://learnyouahaskell.com/chapters
- **C++**: http://www.learncpp.com/
- **Scala**: http://www.scala-lang.org/documentation/
- **Ruby**: https://www.codecademy.com/learn/ruby
- **PHP**: https://www.codecademy.com/learn/php
- **Racket**: https://learnxinyminutes.com/docs/racket/

- There are resources of 100's of languages online. Pick one and mess with it!

Learn to code games!

- Using **Unity**: https://www.udemy.com/unitycourse/
- Using **ActionScript**: https://www.siteground.com/tutorials/actionscript/
- Make an **Android App** (using mostly Java):

http://developer.android.com/training/basics/firstapp/index.html

So much more!

- Create an account on StackOverflow
  - Ask and answer questions!
- Subscribe to the Programming subreddit (the people are only a little pretentious ☺)
- Fork peoples projects on github and read their code
- Contribute to open source projects
- Participate in a hackathon
- Learn how to write scripts to automate things you don’t like spending time on!

Finally, thanks for the opportunity to work with you all!