CSE 373: Final thoughts

Michael Lee
Friday, Mar 9, 2018

Logistics

Reminder:

- Project 4 due tomorrow
- Final review sessions:
  - Monday, Mar 12: EEB 125, 4:30 to 6:30
  - Tuesday, Mar 13: EEB 105, 4:30 to 6:30
- Final on Thursday, Mar 15, 2:30 to 4:30 in Gowen 301

Logistics

Shortened office hours next week:

- Monday: 2:30 to 4:30 in the 4th floor breakout
- Tuesday: 2:30 to 4:30 in CSE 216
- Wednesday: 2:30 to 4:30 in CSE 216
- No office hours Thursday and Friday

(Why CSE 216 on Tuesday and Wednesday? Because literally every other room is booked those two days.)

Victory lap

Hooray! We made it!

Today:

1. Looking back
2. Course goals
3. Now what?
4. Final questions?

Victory lap

But first, big thanks to your TAs!

Alan Tan
Andrew Li
Joseph Wunderlich
Justin Sievers
Kevin Du
Kimberly Bautista
Meredith Wu
Valerie Liao
Vivian Lappenbusch

Victory lap

And also... thank you!

This is a challenging course, with a lot of material, covered in a short span of time.

I'm really impressed with how much everybody has progressed and the work everybody put in.

And also, thank you for putting up with me!

This is my first time teaching, and I appreciate everybody's patience when things get messed up or delayed.

Victory lap

Relatedly, course evals!

https://uw.iasystem.org/survey/188762

There are definitely some things I would change about this course, and I'd also welcome any feedback or suggestions you have!

Looking back...

Here are the first few slides from the first day of the quarter...

CSE 143

Basic techniques for storing and manipulating data

- "Expanding arrays"
- Nodes and pointers/references
- Trees and recursion

How to use pre-made data structures

- Using standard Java collections
- (Lists, Stacks, Queues, Sets, Maps...)

Techniques for organizing code

- Refactoring, coding style
- Client vs implementer

CSE 373

Content

- Learn new techniques
- Learn how exactly data structures work
- How to precisely analyze algorithms

Core skills

- Design decisions, tradeoffs, and critical thinking
- Abstraction and implementation
- Communication: being able to justify your decisions

Incidental skills

- Debugging and testing
- Exposure to tools used in the industry

CSE 373

Big picture:

- Give you the tools to think critically about code and make decisions
- Give you a taste of how to approach and build large programs
- Give you a strong foundation for whatever tech-related thing you want to do next

Success?

Did we succeed?

I hope so – hopefully you agree.

(And if not, see course evals.)

Hopefully, this got you excited for more, and CSE 373 won't be your last exposure to computer science. If so...

What's next?

What's next?

13

Different pathways

If you wanted to...

- Want to learn about popular tools used in the industry (e.g. writing shell scripts in Linux, using version control and git)
- Understand how exactly your hardware and code interacts, on a low level

...then take **CSE 374, Intermediate Programming Concepts and Tools**

If you want to go even deeper, and...

- Learn more about things like locality and managing memory
- Learn how operating systems work, on a high level

...then take **CSE 410, Computer Systems**

14

Different pathways

If you wanted to...

- Learn new strategies for problem-solving and writing code
- Learn how programming languages actually work (e.g. how compilers and interpreters work)

...then take **CSE 413, Programming Languages**.

15

Different pathways

Also, try learning a new programming language!

Maybe another mainstream and popular one?

- Python ("executable pseudocode")
- C# ("Java done right")
- HTML/CSS/Javascript ("for making websites/webapps")
- C++ ("if you want full control")

Or maybe a more non-conventional one?

- Haskell
- Racket
- Rust
- Prolog

16

Different pathways

If you wanted to...

- Learn how to interact with and tune databases
- Learn about the core concepts of parallelism (how to sanely manage code where multiple programs are running at the same time)
- Learn how to manage large amounts of data (more data then you can fit in RAM? More data then you can fit on a single computer?)

...then take **CSE 414, Introduction to Database Systems**.

17

Different pathways

If you wanted to...

- Learn about artificial intelligence and machine learning
- Something something self-driving cars
- Learn how to write programs that take data and automatically learn how to make decisions based on that data
- Learn how to combine statistics and computer science

...then take:

- **CSE 415, Introduction to Artificial Intelligence**
- **CSE 416, Introduction to Machine Learning**

18

Different pathways

If you wanted to...

- Learn more about the underlying theory behind data structures and algorithms
- Learn about more complex algorithms, how to design algorithms
- Learn more about things like P vs NP

...then take **CSE 417, Algorithms and Computational Complexity**

19

Different pathways

If you wanted to learn how to build websites (both frontend and backend)...

...then take either CSE 154, INFO 343, or INFO 344.

Or try learning on your own! You can probably self-teach yourself enough to make a basic website over a weekend. (Writing something more complex will of course take longer.)

20

Different pathways

In fact, you can learn many different things by yourself!

- Learn how to use tools like Git
- Write an Android app (using Java)
- Write a video game (using Unity? using Java?)
- Buy a Raspberry Pi or an Arduino and start messing around with it
- Learn how to automate boring things! (Google "Automate the Boring Stuff")

21

Learning new things

...but how?

It's both easier and harder then it sounds:

1. Google and find resources on the topic you want to self-teach:
    - "Writing android apps in Java"
    - "Learning Python"
    - "How to write a simple compiler"
    - "Machine learning tutorial"
2. Look through a few resources and eliminate the ones that seem sloppy
3. Work through the tutorial and **practice as you go**
4. Try actively applying what you've learned to work on projects

22

Learning new things

General resources:

- You can find tutorials on almost any topic via google
- Coursera: lots of good online courses
- Google "open source CS curriculum" or "what every CS major should know" if you want a more curated list
- Try contributing to open source; try attending hackathons
- StackOverflow: pick interesting tags, sort by top, and read
- Sporadically browse Hacker News or /r/programming (Warning: people can often be opinionated, and sometimes a little pretentious, but you can pick up some interesting nuggests)

23

Learning new things

Core takeaways:

1. **Prioritize practice: apply and use what you're learning**
    - If you don't, you're only fooling yourself.
    - Prefer active learning, not passive learning.
    - Work on projects (personal, coursework, for clubs, research...)
2. **Take charge of your own education**
    - Don't wait for people to teach you things; go out and teach it to yourself.
3. **You can now learn anything in CS with enough effort**
    - This class was designed to give you the foundation to learn whatever you want to learn about in CS or tech.
    - Don't let yourself be intimidated or discouraged. Break it down, take it one step at a time, ask lots of questions, and don't give up.

24

Questions?

Any questions?