

# CSE 373: Final thoughts

---

Michael Lee

Friday, Mar 9, 2018

## 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

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

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.)

Hooray! We made it!

Hooray! We made it!

Today:

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

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

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!

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.

Relatedly, course evals!

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

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!

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

## Basic techniques for storing and manipulating data

- ▶ “Expanding arrays”
- ▶ Nodes and pointers/references
- ▶ Trees and recursion

## 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...)

## 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

## Content

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



## 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

## 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

Big picture:

- ▶ Give you the tools to think critically about code and make decisions

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

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?

# Success?

---

Did we succeed?

I hope so – hopefully you agree.

# Success?

Did we succeed?

I hope so – hopefully you agree.

(And if not, see course evals.)



# 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?

## 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)

## 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

## 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**

## 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

## 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

## 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**



If you wanted to...

- ▶ Learn new strategies for problem-solving and writing code

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)

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**.

## Different pathways

Also, try learning a new programming language!

## Different pathways

Also, try learning a new programming language!

Maybe another mainstream and popular one?

- ▶ Python (“executable pseudocode”)

## Different pathways

Also, try learning a new programming language!

Maybe another mainstream and popular one?

- ▶ Python (“executable pseudocode”)
- ▶ C# (“Java done right”)

## 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”)

## 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”)



## 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?

## 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

## Different pathways

If you wanted to...

- ▶ Learn how to interact with and tune databases

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)

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?)

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.**

## Different pathways

If you wanted to...

- ▶ Learn about artificial intelligence and machine learning

## Different pathways

If you wanted to...

- ▶ Learn about artificial intelligence and machine learning
- ▶ Something something self-driving cars



## 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

## 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

## 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**

If you wanted to...

- ▶ Learn more about the underlying theory behind data structures and algorithms

If you wanted to...

- ▶ Learn more about the underlying theory behind data structures and algorithms
- ▶ Learn about more complex algorithms, how to design algorithms

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

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**

## Different pathways

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



## 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.

## 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.)

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

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

- ▶ Learn how to use tools like Git

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

- ▶ Learn how to use tools like Git
- ▶ Write an Android app (using Java)

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?)

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

## 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”)



# Learning new things

...but how?

# Learning new things

...but how?

It's both easier and harder than it sounds:

# Learning new things

...but how?

It's both easier and harder than it sounds:

1. Google and find resources on the topic you want to self-teach:

# Learning new things

...but how?

It's both easier and harder than 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”

# Learning new things

...but how?

It's both easier and harder than 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

# Learning new things

...but how?

It's both easier and harder than 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**

# Learning new things

...but how?

It's both easier and harder than 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

General resources:

- ▶ You can find tutorials on almost any topic via google



General resources:

- ▶ You can find tutorials on almost any topic via google
- ▶ Coursera: lots of good online courses

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

### 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

### 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

# 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 nuggets)

# Learning new things

Core takeaways:

1. **Prioritize practice: apply and use what you're learning**

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...)

# 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**



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.

# 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**

# 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.

Any questions?