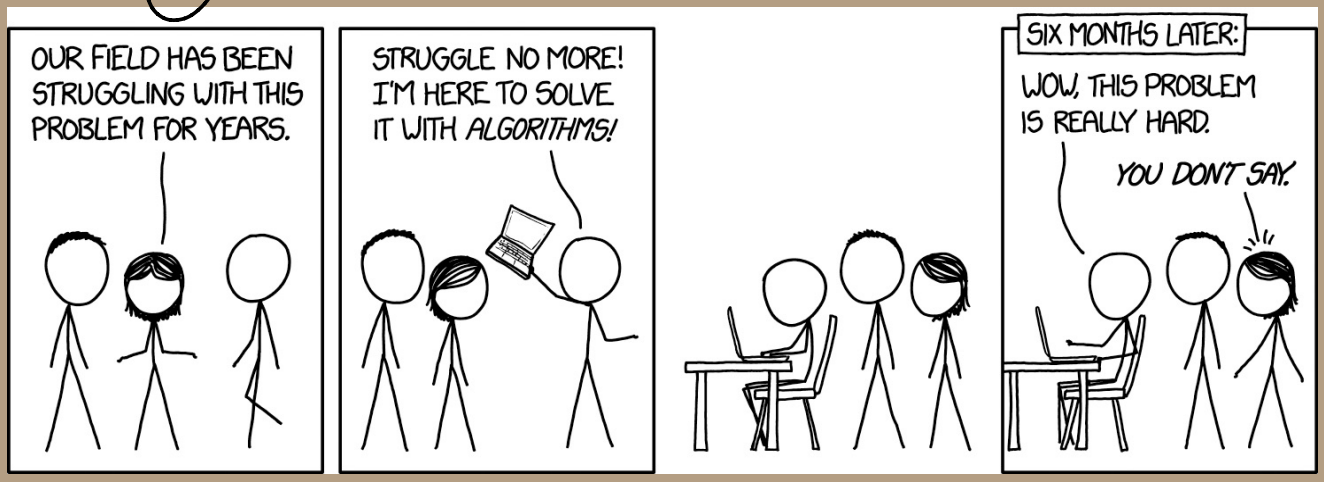
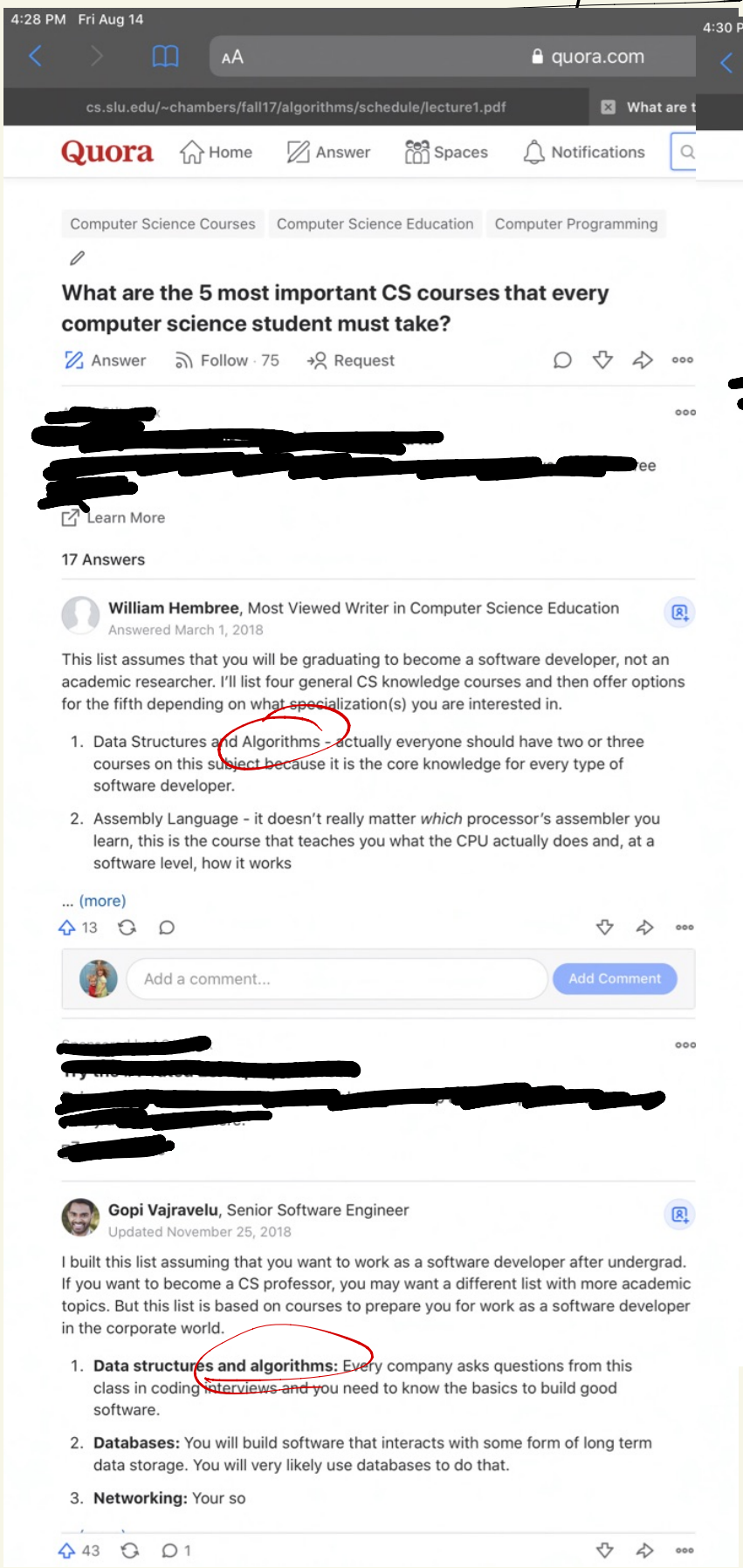
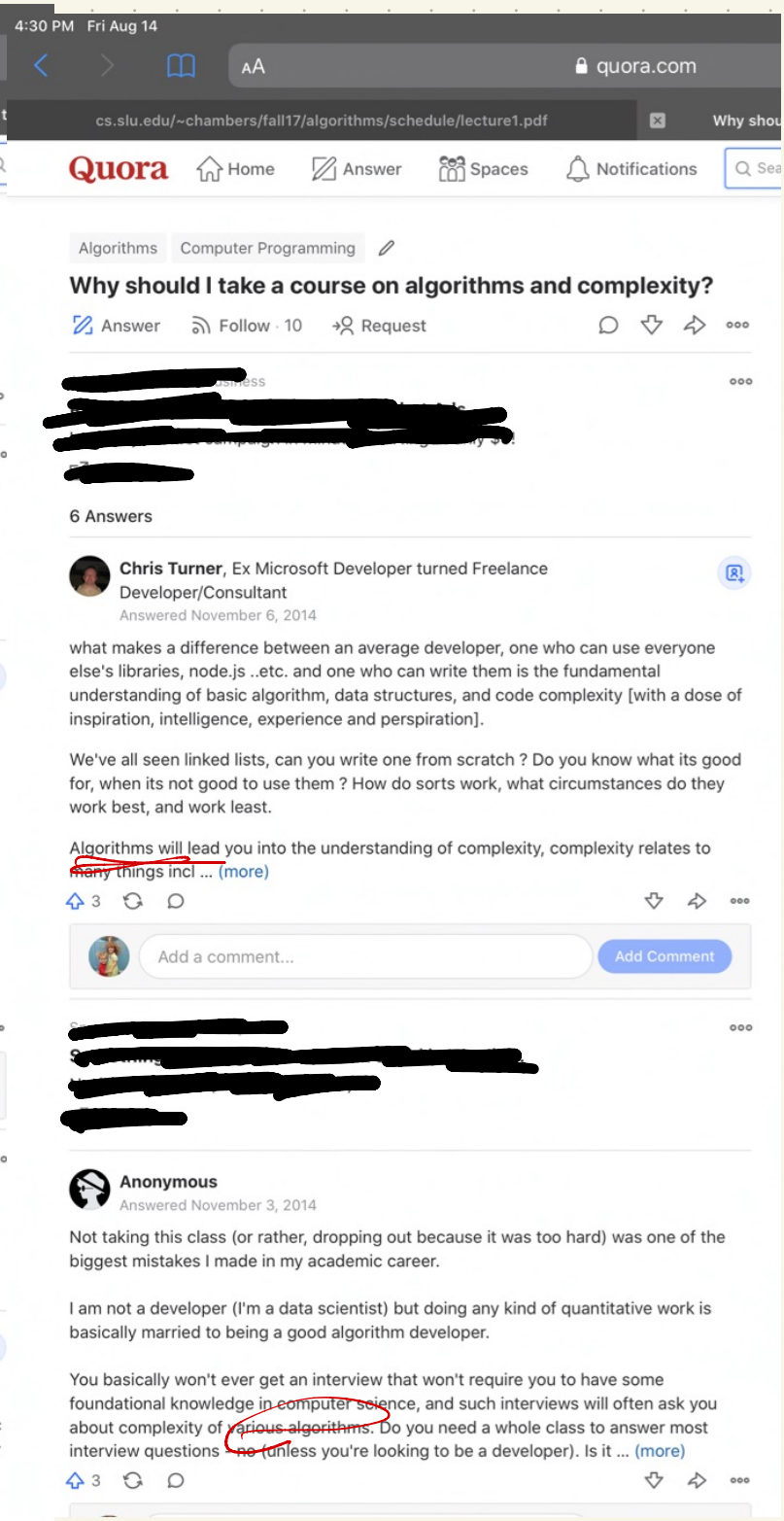
# Algorithms – Spring '25




## Introduction + Recap

First & Why?


4:28 PM Fri Aug 14
AA
quora.com
cs.slu.edu/~chambers/fall17/algorithms/schedule/lecture1.pdf
What are t
Quora
Home
Answer
Spaces
Notifications
Computer Science Courses
Computer Science Education
Computer Programming
What are the 5 most important CS courses that every computer science student must take?
Answer
Follow · 75
Request
Learn More
17 Answers
William Hembree, Most Viewed Writer in Computer Science Education
Answered March 1, 2018
This list assumes that you will be graduating to become a software developer, not an academic researcher. I'll list four general CS knowledge courses and then offer options for the fifth depending on what specialization(s) you are interested in.
1. Data Structures and Algorithms - actually everyone should have two or three courses on this subject because it is the core knowledge for every type of software developer.
2. Assembly Language - it doesn't really matter which processor's assembler you learn, this is the course that teaches you what the CPU actually does and, at a software level, how it works
... (more)
13
Add a comment...
Add Comment
Gopi Vajravelu, Senior Software Engineer
Updated November 25, 2018
I built this list assuming that you want to work as a software developer after undergrad. If you want to become a CS professor, you may want a different list with more academic topics. But this list is based on courses to prepare you for work as a software developer in the corporate world.
1. Data structures and algorithms: Every company asks questions from this class in coding interviews and you need to know the basics to build good software.
2. Databases: You will build software that interacts with some form of long term data storage. You will very likely use databases to do that.
3. Networking: Your so
43
1


4:30 PM Fri Aug 14
AA
quora.com
cs.slu.edu/~chambers/fall17/algorithms/schedule/lecture1.pdf
Why shou
Quora
Home
Answer
Spaces
Notifications
Algorithms
Computer Programming
Why should I take a course on algorithms and complexity?
Answer
Follow · 10
Request
6 Answers
Chris Turner, Ex Microsoft Developer turned Freelance Developer/Consultant
Answered November 6, 2014
what makes a difference between an average developer, one who can use everyone else's libraries, node.js ..etc. and one who can write them is the fundamental understanding of basic algorithm, data structures, and code complexity [with a dose of inspiration, intelligence, experience and perspiration].
We've all seen linked lists, can you write one from scratch ? Do you know what its good for, when its not good to use them ? How do sorts work, what circumstances do they work best, and work least.
Algorithms will lead you into the understanding of complexity, complexity relates to many things incl ... (more)
3
Add a comment...
Add Comment
Anonymous
Answered November 3, 2014
Not taking this class (or rather, dropping out because it was too hard) was one of the biggest mistakes I made in my academic career.
I am not a developer (I'm a data scientist) but doing any kind of quantitative work is basically married to being a good algorithm developer.
You basically won't ever get an interview that won't require you to have some foundational knowledge in computer science, and such interviews will often ask you about complexity of various algorithms. Do you need a whole class to answer most interview questions - no (unless you're looking to be a developer). Is it ... (more)
3

# Mindset (& main goals):

- Interview: how to concisely explain your answer to a technical question, complete with trade-offs
- Design meeting: same!
- Developer: What known approaches / issues are there with regards to your problem / code?
- Modeling: If already developed, which solution fits your domain area best.

(Note: all of these are relevant for data structures, too!)

# Q: What IS an algorithm?

- way to solve a problem
- template or set of procedures to solve a problem

**algorithm**
*noun*

Word used by programmers when they do not want to explain what they did.

Related: What is a program?
(+how is it different?)

- program solves specific problem

programs generally fix
- language
- compiles
- set of data structure
:

Goal: Give high level idea of your solution.

- Not code!
  Why?
- Not English prose! ←
  Use data structures, loops, functions, etc.

In here, 3 parts to every algorithm:

1. Pseudocode
2. Runtime (with justification) → vs. O
3. Proofs of correctness

Preregs:

① Discrete math:

Why?

- proofs!
- big-O
- recurrence + induction

② Data structures:

Why?

use them!

understand tradeoffs:

- linked
- array
- tree
- hash
- heap

# Goal of HW0:

- Re-locate references to these 2 classes.
- Re-familiarize your self with core concepts.
- Practice pseudo code. ← (It will be hard!)
- Meet some fellow students & form working relationships
- Figure out Canvas, while tackling known topics! (Hopefully, you'll already have seen everything on HW0...)

# Course logistics:

Canvas site for ~~everything~~ Homework

## Outline:

- Canvas: HW submission, lecture notes
- Perusall: every Sunday + Thurs.
- Homework: Oral + written

## Policies to note:

- Integrity
- Group work
- Missing work policies

Next time: same review for HW0!