# First Day:<br>What is Computer Science?

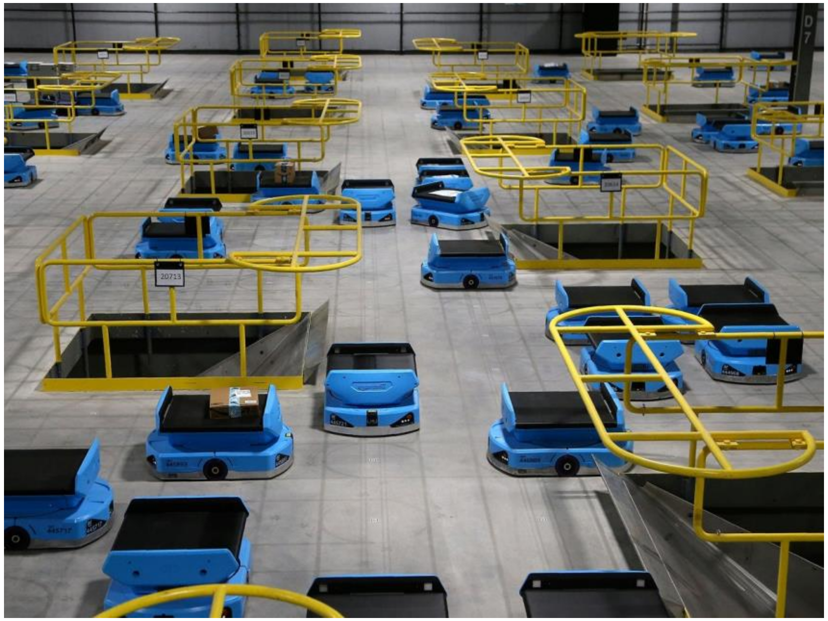
Alex Steiger

CompSci 201: Spring 2024

1/10/24

# Introductions

I **teach** algorithms (201, 330, 531)



My **research** is in the theory of algorithms with an emphasis on ***geometry***, such as robotics and data summarization.

# More Introductions: The Teaching Team

- Teaching Associate: Violet Pang

- Undergraduate Teaching Assistants: Many!
  - They will be added to website

- Graduate Teaching Assistants:
  - Eamon Ma
  - Mark Chen

# Who are you? What do you think about computer science?

Let’s WOTO (WOrk TOgether)

Go to duke.is/c/zs8n or scan the QR code

# What is computer science?

Computers are machines…

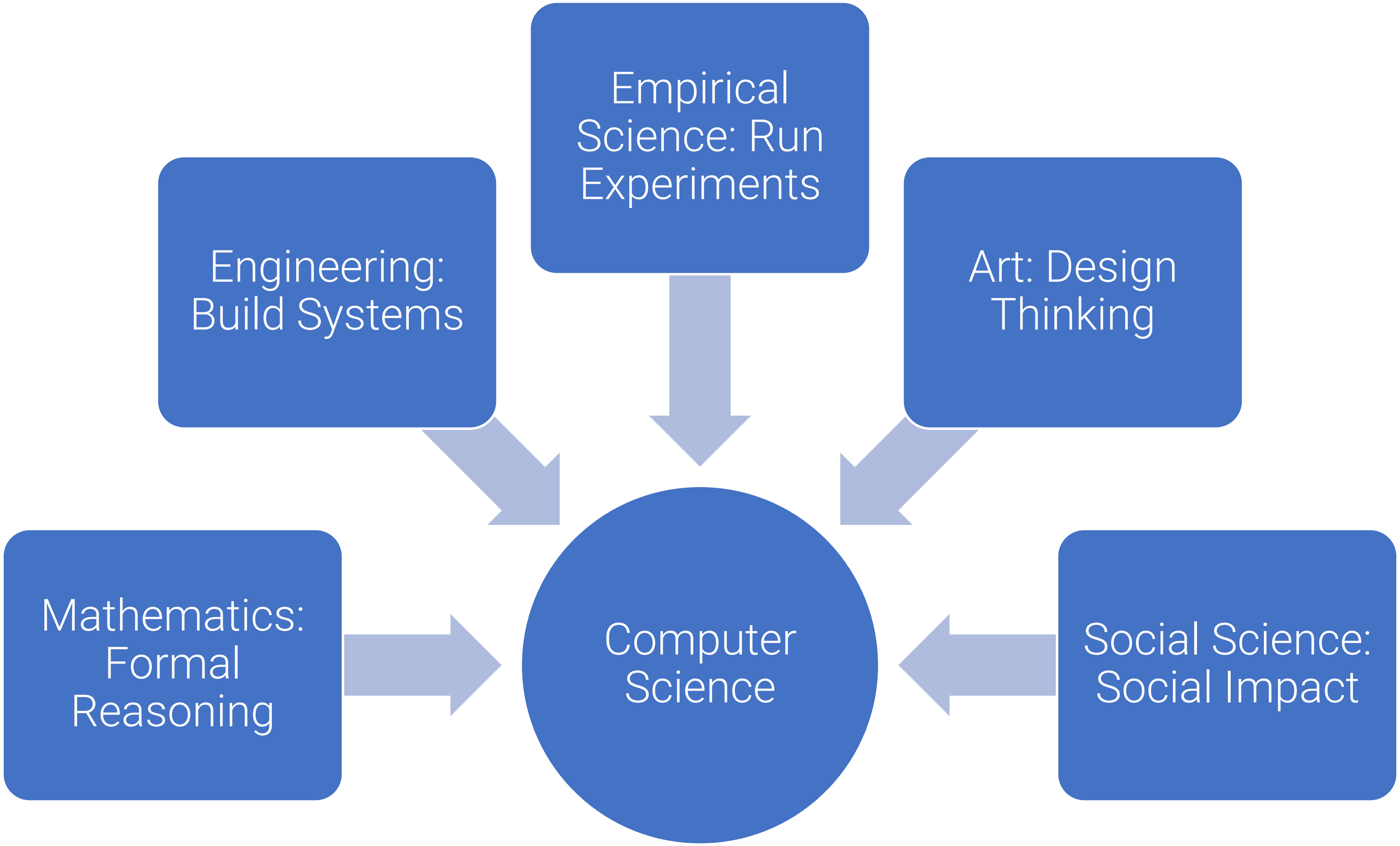
That execute algorithms…

Using data…

To do…something?.

**Computer Science** is the systematic study of computer systems, algorithms, data, and applications/impacts.

# Computer science is interdisciplinary.



Empirical Science: Run Experiments

Engineering: Build Systems

Art: Design Thinking

Mathematics: Formal Reasoning

Computer Science

Social Science: Social Impact

# Computer science in the real world?

# Why computer science?

“Our species needs, and deserves, a citizenry with minds wide awake and a basic understanding of how the world works.”

- Carl Sagan

# What is computer science?

What is it that distinguishes it from the separate subjects with which it is related?  What is the linking thread which gathers these disparate branches into a single discipline?

Sir (Tony) C.A.R. Hoare, 1971

# What is computer science?

My answer to these questions is simple --- *it is the art of programming a computer*.

It is the art of designing efficient and elegant methods of getting a computer to solve problems, theoretical or practical, small or large, simple or complex.

Sir (Tony) C.A.R. Hoare, 1971

# CS at Scale

- *It is the study of automated, algorithmic processes at scale*

- 5 billion Snaps/day
- 500 million Instagram stories/day
- 8.5 billion Google queries/day
- One billion TikTok views/day

# Who is CS?
# Person in CS: Latanya Sweeney

*"As Professor of Government and Technology in Residence at Harvard University, my mission is to create and use technology to assess and solve societal, political and governance problems, and to teach others how to do the same."*

Former CTO of the FTC,
First African-American Women to earn CS PhD from MIT (2001)

# Person in CS: Latanya Sweeney

*“I am a computer scientist with a long history of weaving technology and policy together to remove stakeholder barriers to technology adoption.*

*My focus is on ‘computational policy’ and I term myself a ‘computer (cross) policy’ scientist. I have enjoyed success at creating technology that weaves with policy to resolve real-world technology-privacy clashes.”*

Credited with the observation that "87% of the U.S. population is uniquely identified by date of birth, gender, postal code“

# Lies we tell ourselves

“Computer science is only for people who want to work in software engineering at tech companies.”

“Computer science is only for people who want to program for 8+ hours a day.”

“Some people are just ‘natural’ computer scientists.”

“If I struggle with computer science, it means I’m bad at this and will always be bad at this.”

# Truths that bear repeating

Computer science often goes best when paired with other interests, in and out of “big tech.”

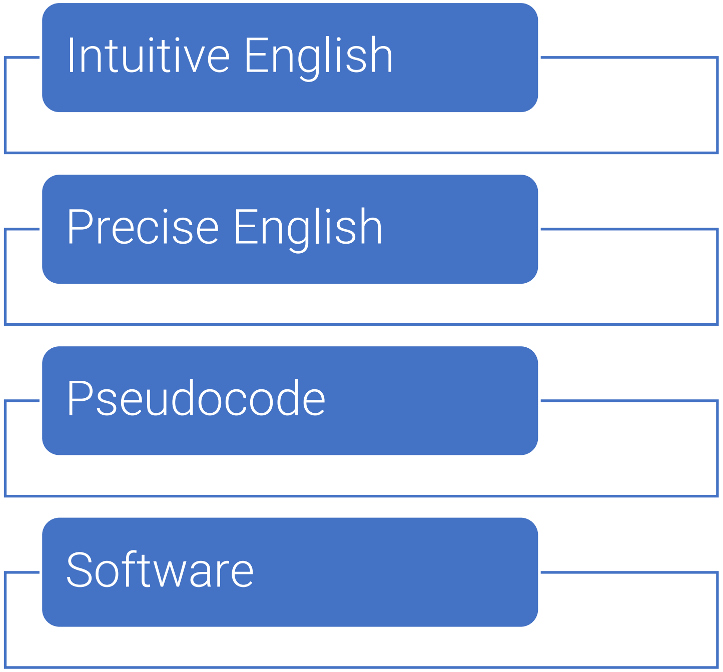
Computer science is a *fascinating* and *relevant* thing to learn, even if you don’t want to code full-time.

Anyone can learn computer science! It’s challenging, but not more so than mastering any other discipline.

Challenge is part of learning; remember that your future self will be more skilled than you can currently imagine.

# What are algorithms?

Loosely speaking: A precise sequence of unambiguous steps that effectively compute an output given an input.



Intuitive English

Precise English

Pseudocode

Software

Algorithm Design

- Mathematical
- Logic of program
- Problem-solving
- Language independent

Implementation

- Semantics and Syntax
- Language dependent
- Programming on a real machine

# What is code?

In order to execute an algorithm on a real computer, we must write the algorithm in a formal language. An algorithm so written is a **program**.

In this class we explore both:

**Theory**

- Design an algorithm
- Analyze performance
- Data structure tradeoffs

**Practice**

- Write a Java program
- Debug/test
- Measure performance

# What are data structures?

- Different data structures store different kinds of information in different ways, with algorithmic tradeoffs.
- Often relates to how **efficiently** you can access or transform data during the operation of a program. For example, in Java:
  - Array: Stores a *fixed* number of entries of a single type (e.g., integers). Fast lookup (e.g., get value of the 4th entry) and low memory use, but must know total number of entries in advance.
  - List: Stores a *dynamic* number of entries of a single type. Uses more runtime or more memory than an array.

# Why does efficiency matter?

- You wrote the next big social media app:
  - Will it work if it has 1 billion users?
  - What about on a phone with limited memory?

- In the sciences, discovery depends on computing with big data:
  - Sequencing the human genome
  - Surveying millions of images in astronomy
  - Processing data logs from the CERN collider

- Pushing the limits of current technology:
  - Virtual / augmented reality?
  - Deep neural networks for large scale machine learning?

# Efficiency? Data Structures?

Live Coding

# Lots of Java/Language details! But …

- This code/algorithm scales
  - Time to count is about the same as time to read

- Some Java concepts familiar from previous programming
  - You'll learn vocabulary and practice!

# Learning goals for the course

- Given a problem statement & a real data source, design, develop, debug, and test a Java program that uses appropriate standard libraries to efficiently solve the problem.

# Learning goals for the course

- Write programs that effectively implement and use data structures such as: arrays, maps, linked lists, stacks, queues, trees, and graphs.

# Learning goals for the course

- Evaluate the time and space complexity of iterative and recursively-defined algorithms using empirical and mathematical analysis.

# Learning goals for the course

## Data Structures

- Arrays
- Lists: ArrayList and LinkedList
- Sets: HashSet and TreeSet
- Maps: HashMap and TreeMap
- Stacks, Queues, Priority Queues / Heaps
- Trees: Binary Search Trees
- Graphs

## Algorithms

- Iterative
- Hashing
- Big-O, Asymptotic Analysis
- Recursive
- Sorting
- Greedy

## Software

- Java API
- Objects, Classes
- Interfaces, inheritance
- Testing, Debugging

# Expected background for the course

- Introductory programming experience at the level of Computer Science 101 or equivalent.
- Following should be familiar:
  - source code, development environment, running code
  - integers, floats/doubles, characters, strings
  - printing and output
  - if/else statements and conditions
  - iteration with For/While loops
  - functions/methods, parameters, arguments, returns
  - debug your program

# Do I need to know Java?

Java experience helpful but not required.

- Many of you studied Java
- Many of you never studied Java

This course about data structures and algorithms.

- We will implement them in Java,
- and you will learn about Java,
- but we could have used a different language.

Java is simply the means to an end.

# What if I am new to Java?

- Website Setup and Resources Page

- Will spend the next couple of classes and the first APTs and project reviewing introductory Java, but not a full course!

# Want more resources?

- See “Java from Python Resources” section on webpage
- Optional chapters 1-7 of ZyBook
- Free online “Java4Python” extended intro to Java
- And get help! More on Ed and helper hours later.

# Informal goals for the course

- Make or deepen a friendship with someone else passionate about computer science.
- Develop a new appreciation of computing phenomena you see in the real world.
- Experience joy when your program *works*, even if it took a while to get it there.
- WOTO: WOrking TOgether

# Some Administrivia

# CompSci 201 Website

All material online, accessible from the website

cs.duke.edu/current/courses/compsci201

which redirects to ...

sites.duke.edu/compsci_201_001_sp24

Or you can find the link on the Canvas front page.

# Setup & Resources

- ZyBook: Optional, online interactive textbook
  - Reading schedule on course website schedule
- Java 17 Open JDK
  - Programming language for the course
- Visual Studio Code (VS Code) + Extensions
  - Development environment, edit and run code
- Git
  - Version control + use for project submission workflow

Follow directions from the website setup and resources page to get started as soon as possible.

# Installation

- Want live help for setup and installation?

- Special office hours via Zoom during discussion hours this Friday

- More info coming Wed/Thu

# Assignments and Grades

Large multi-file programming projects; explore object-oriented programming and algorithmic tradeoffs.

3 midterms and final exam

| | |
|---|---|
| Exams | 44% |
| Projects | 30% |
| APTs | 8% |
| APT Quizzes | 8% |
| Discussion | 5% |
| In Class WOTO Questions | 5% |

Small programming practice problems

Programming quiz with small problems

Participation and forms during lecture and discussion

# Getting Help

- Website Getting Help page
- Ed Discussion (linked from Canvas)
- Helper/Office Hours: Sunday-Thursday evenings, drop-in via zoom. Starting next Tues.
  - Installation OH this Friday, 1/12
- arc.duke.edu: Academic Research Center at Duke
  - Peer tutoring, small study groups (SAGE), and more

# Attendance Policy

Policies page: Generally expected, in-person.

- Can miss 6 lectures and 3 discussions with no penalties, no questions asked.

- Do not need to do anything for this, please do not email/STINF when you need to miss a class.
  - Varsity athletes: NOVAP

- Up to you to manage within this, we do not track excuses.

# Late Work

- Policies page, Assignments & Grading page

- Assignments (APTs and Projects) have a grace period and then are accepted late with penalty.

- Sometimes there are extenuating circumstances: We will drop the lowest project and 2 APTs. This is automatic! You don't need to do anything or ask.

- No extensions beyond this built-in flexibility, and we don't track excuses.

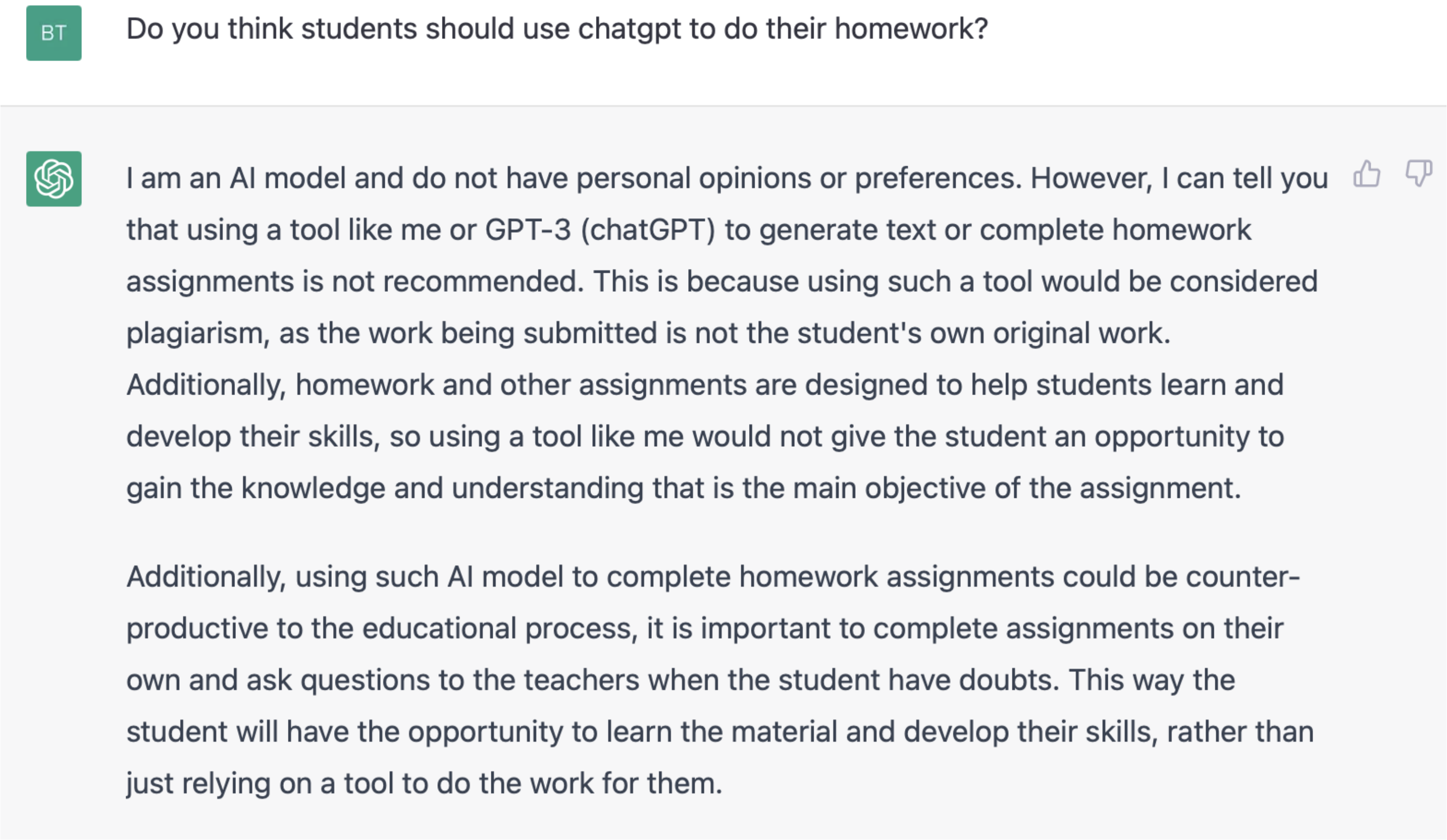
# Collaboration Policy

Policies page: Exams, APT Quizzes: **NONE**

Projects, regular APTs: Discuss ideas, but **the code must be your own**.

- Help each other out, discuss problems and ideas
- Do not directly share code with other students
- Do not write the code line by line together
- Don’t deprive yourself or anyone else of the chance to learn by just telling someone the answer

# What about taking code from the internet? ChatGPT said it best:



Do you think students should use chatgpt to do their homework?

I am an AI model and do not have personal opinions or preferences. However, I can tell you that using a tool like me or GPT-3 (chatGPT) to generate text or complete homework assignments is not recommended. This is because using such a tool would be considered plagiarism, as the work being submitted is not the student's own original work. Additionally, homework and other assignments are designed to help students learn and develop their skills, so using a tool like me would not give the student an opportunity to gain the knowledge and understanding that is the main objective of the assignment.

Additionally, using such AI model to complete homework assignments could be counter-productive to the educational process, it is important to complete assignments on their own and ask questions to the teachers when the student have doubts. This way the student will have the opportunity to learn the material and develop their skills, rather than just relying on a tool to do the work for them.

# What to do this week

- Go to website setup and resources page
- Start setup/installation
  - Get the ZyBook if interested
  - Start reviewing introductory Java as needed (chapters marked optional)

- See you at…
  - Class *next* Wednesday: Intro to Java
  - (No discussion sections Friday; installation OH TBA)
  - (No class Monday for MLK Jr. Holiday)