COMP 3270: Introduction to Algorithms

AUBURN UNIVERSITY

Hugh Kwon

What is an algorithm?

- An algorithm is a well-defined computation procedure (i.e., a finite number of computational steps specified at the required level of detail) that generally (but not always) takes some data as input and generally (but not always) produces some data as output in order to solve a problem.

Different views of algorithms

- They are magical, all powerful and taking over the world!
  - http://www.youtube.com/watch?v=TDaFwnOiKVE
- They are for socially inept nerds...
  - https://www.youtube.com/watch?v=k0xgjUhEG3U
- They are for everybody, and here is why in plain language...
  - https://www.youtube.com/watch?v=vSi6YoTPWLw
- But what we're concerned about is technical stuff, like this...
  - https://www.youtube.com/watch?v=kPRA0W1kECg

Why did I show these videos?

- To illustrate how pervasive algorithms have become…

- To emphasize that algorithms may be mysterious/magical, unreadable and uncontrollable to lay people, but you, as students of computing, must develop the skills of algorithmic thinking: being able to understand, design, implement and control them!

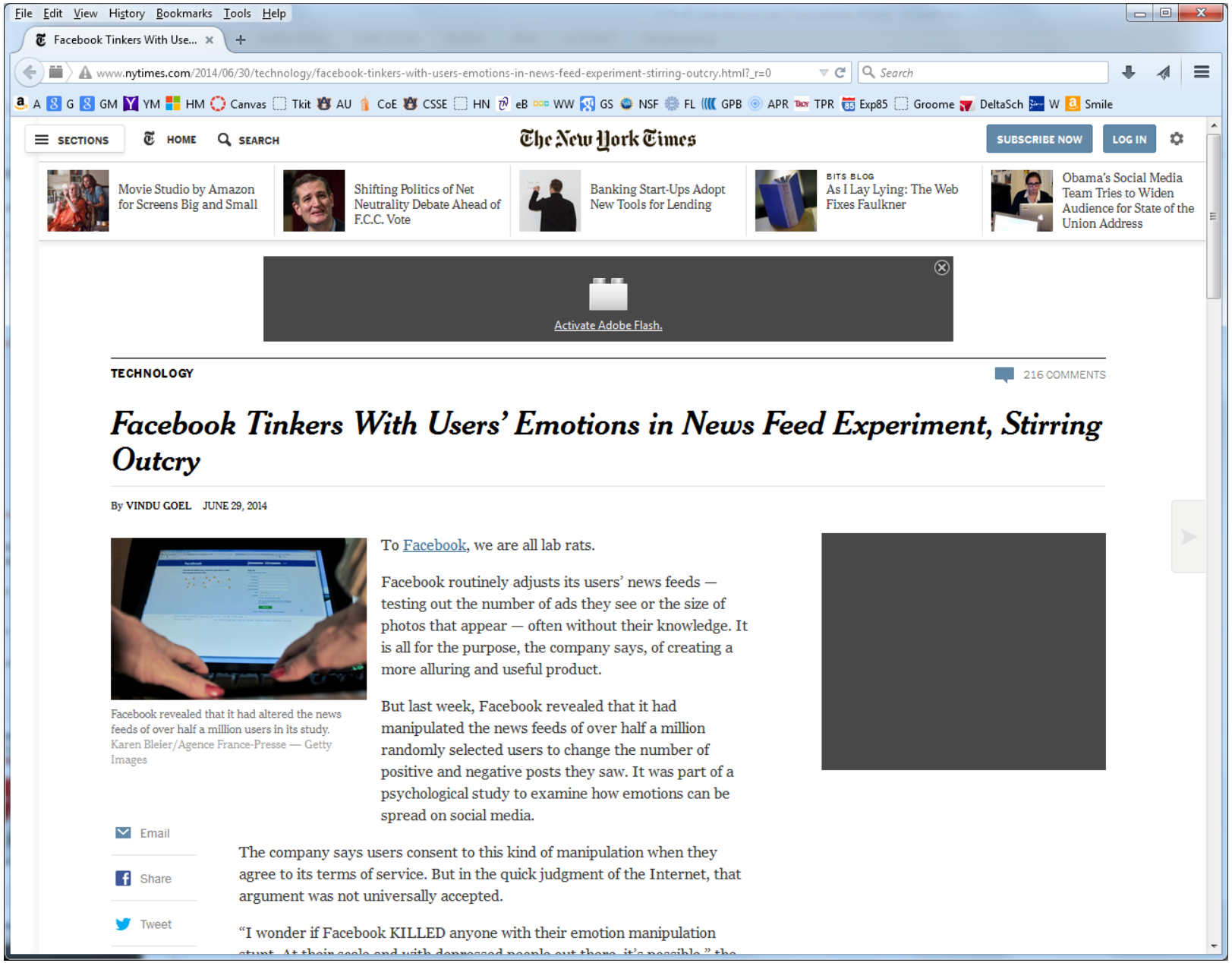
Why study algorithms?

- Algorithms affect ALL FACETS of your life!
  - your romantic life
  - your social life
  - your financial transactions
  - how you entertain yourself
  - your privacy (or lack of it)

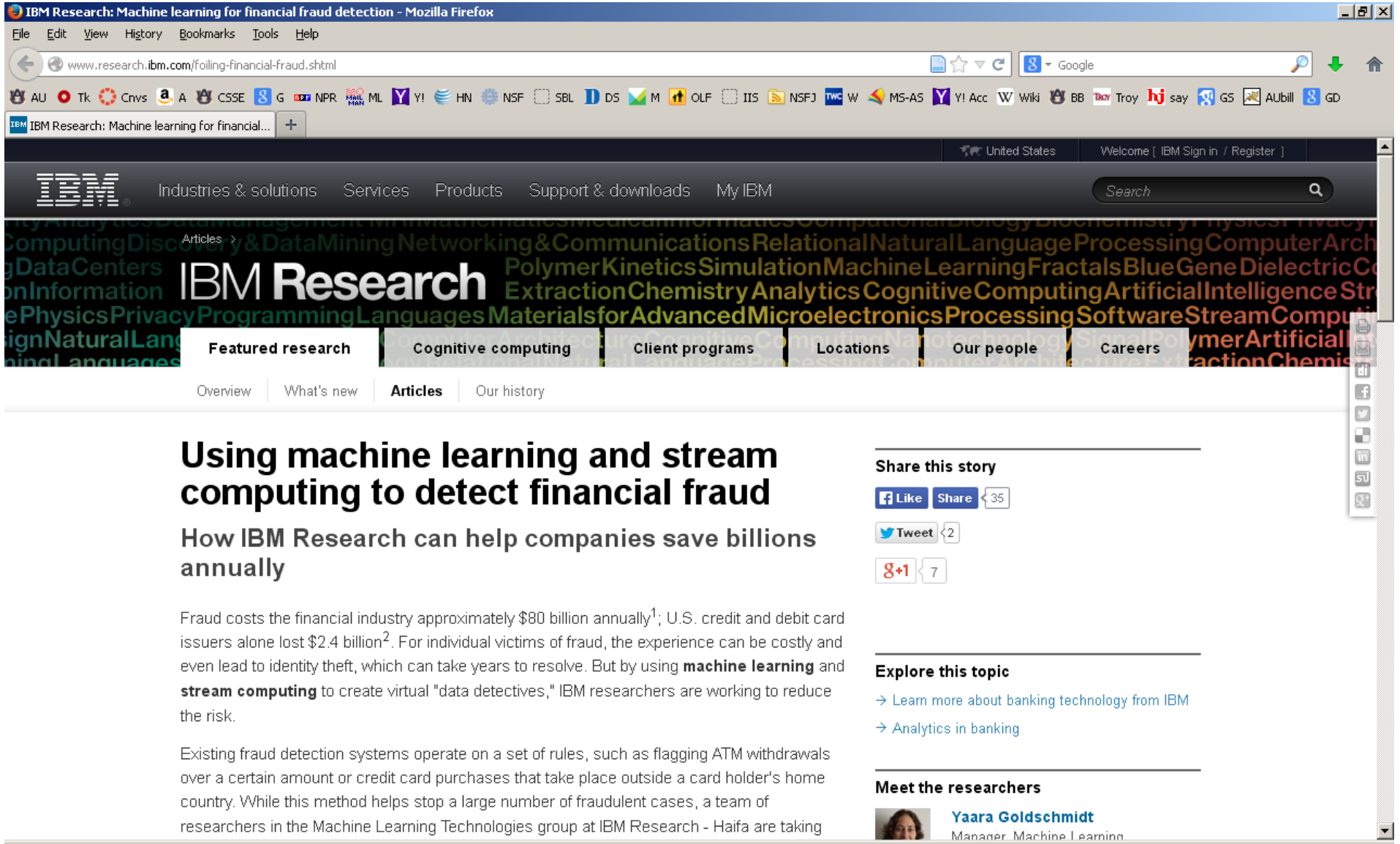
your romantic life

http://en.wikipedia.org/wiki/Stable_marriage_problem#Algorithm

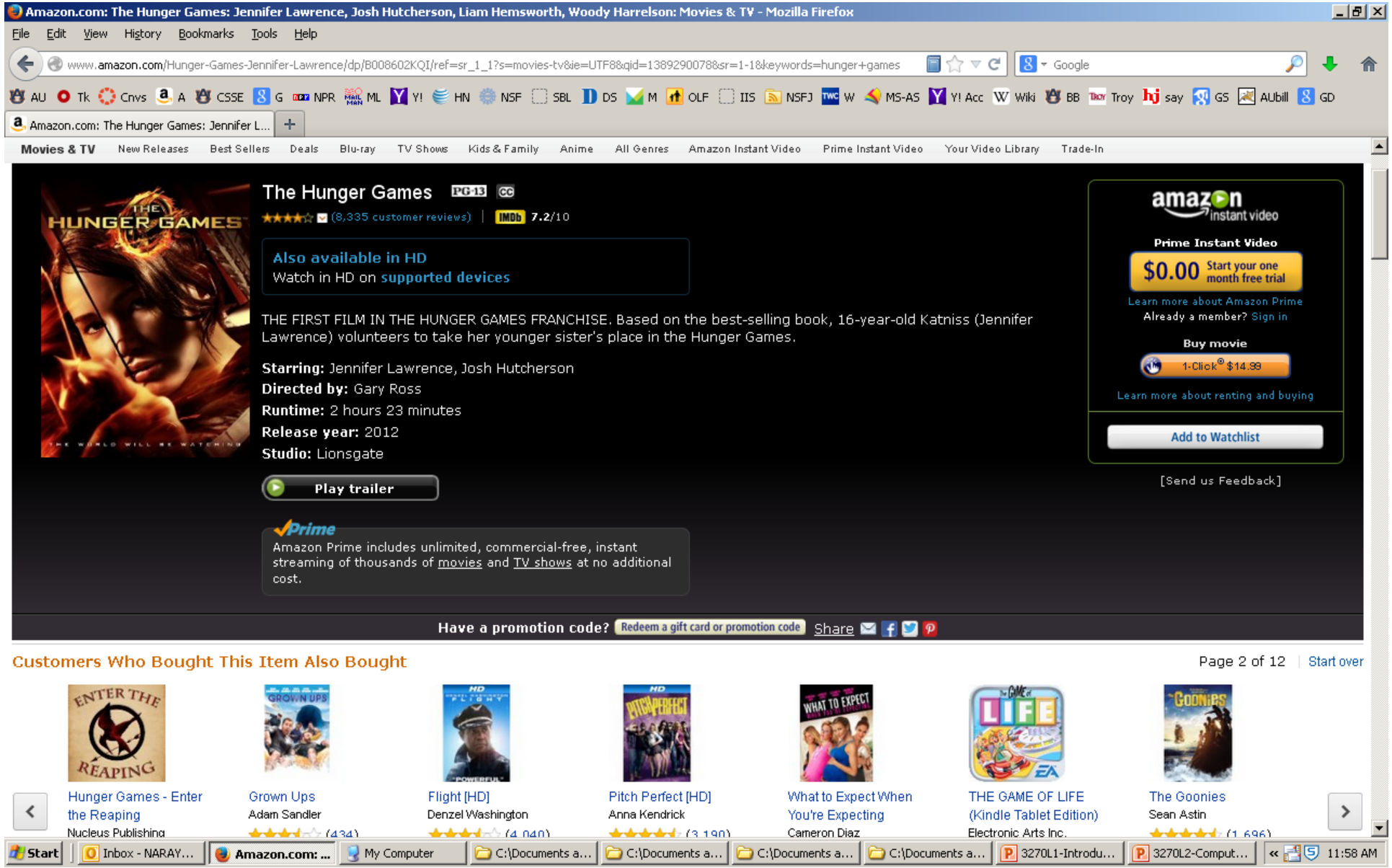
your online social life



your financial transactions



your entertainment



www.cs.carleton.edu/cs_comps/0607/recommend/recommender/index.html

your privacy (actually, lack of it)

http://en.wikipedia.org/wiki/Cryptography

Why study algorithms?

- Algorithms form the backbone of all computations.

- And, as students majoring in a computing-related discipline, you must develop skills to understand and analyze computation, and know how to design computations to solve problems!

- So really, this course is actually about **Computational Problem Solving** – Algorithms are the products of computational problem solving.

Computational Problem Solving

- **Computational Problem Solving** is the art of coming up with a correct AND efficient computational solution (i.e., one or more algorithms) to a given problem.

Problems and Problem Instances

- An instance of a problem consists of one set of valid inputs from which a solution to the problem can be computed.
- “How to sort a set of numbers in the ascending order” is a problem.
- “Sorting the set of numbers 4,5,7,1,2,0 in the ascending order” is a problem instance.

Steps of Computational Problem Solving

- Problem specification
- Designing solution strategies
- Developing corresponding algorithms
  - Understand existing algorithms and modify/reuse, or
  - Design new algorithms
- Understanding the algorithms by simulating their operation on inputs
- Ensuring/proving correctness of the algorithms
- Analyzing and comparing the performance/efficiency of algorithms
  - Theoretically: Using a variety of mathematical tools
  - Empirically: Code, run and collect performance data
- Choosing the best algorithm to code

More on why study algorithms?

- You could make a good living (or even become one of the richest persons on earth) if you know how to design algorithms!
- Or get hired by big tech companies like Google.
- Or convince some investors to fund your startup.

What have we covered so far?

- Different perspectives on algorithms
  - And what perspective we should take as serious students of computing.
- Discussion of why algorithms are important
- Problems and Problem Instances
- The process of Computational Problem Solving
- We will begin to delve into the technical material next
- Meanwhile, a warm-up reading/watching assignment
  - "How algorithms rule the world": https://www.theguardian.com/science/2013/jul/01/howalgorithms-rule-world-nsa
  - "How Google's algorithm rules the web": https://www.wired.com/2010/02/ff_google_algorithm/
  - Another TED talk: https://www.youtube.com/watch?v=H_aLU-NOdHM

DISCLAIMER

- We will now cover the course logistics.
- Although rare, some extraordinary situations may occur that warrant changes to the plans and so the syllabus will change accordingly.
- I will communicate any major edits to the syllabus to you in class, via Canvas, and via Discord.
- However, this slide set will not be updated anymore (except to fix a typo) this semester and may become out-of-date.

What exactly will we study?

- In particular, we will study:
  - How to specify problems so that one can start to think about computational solutions
  - How to come up with solution strategies
  - How to turn those strategies into algorithms
  - How to write algorithms
  - How to understand algorithms written by others
  - How to analyze algorithms
  - How to modify and reuse algorithms
  - How to design new algorithms
- We will also look at algorithms and data structures designed by others to solve fundamental computational problems

This course is divided into three parts

- Basics of Computational Problem Solving
  - Specifying problems, coming up with computational strategies to solve problems, turning those strategies into algorithms, understanding, designing and writing algorithms.

- Tools for analyzing algorithms
  - The tools you need to understand, analyze, and guarantee the performance of algorithms. This part is mathematical, abstract and harder than other parts.

- Algorithms that solve important computational problems
  - Application of the tools to understand and analyze existing algorithms that solve fundamental problems of computing.

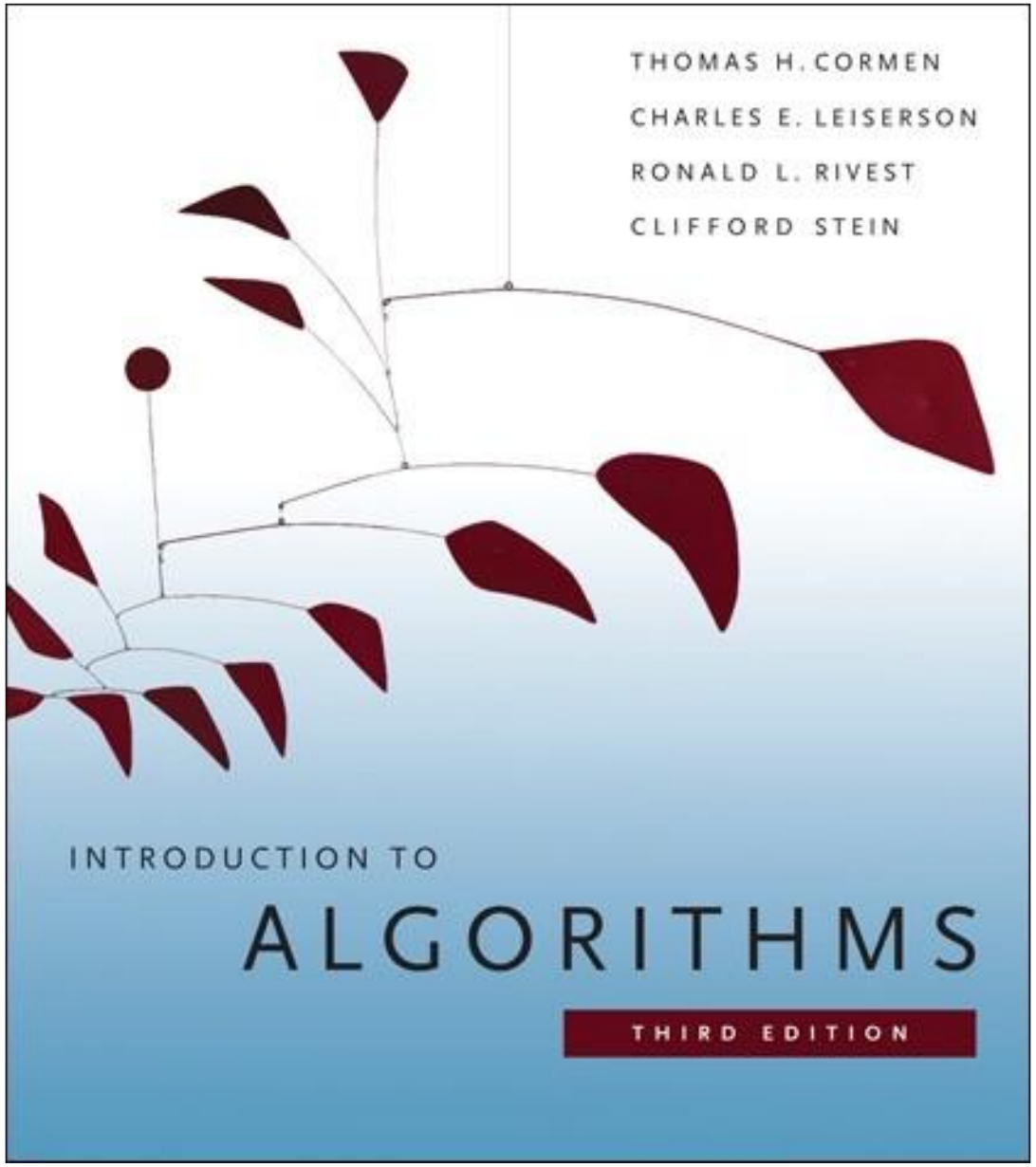
What will you learn?

- Given a problem to solve, be able to specify it at a level that allows you to develop a computational solution
- Given a problem specification, be able to come up with computational strategies for solving it
- Be able to turn those strategies into algorithms, by either reusing existing algorithms with appropriate modifications, or designing a new algorithm
- Given an algorithm, understand how it works
- Check if it is correct
  - If not, fix it
- Calculate its efficiency
  - If inefficient, be able to modify it to make it more efficient
- Several fundamental computational problems and algorithms to solve them

What should you expect?

- Somewhat abstract material that requires you to think about problems and how to solve them in new ways...
- Mathematical…but not rocket science!
- NOT a programming course
- **Not easy** (so plan to spend a lot of time and keep up with the materials; if not, you won't be able to catch up later)

Required Textbook



Attendance Policy

- I know the syllabus says that attendance is required…

- I will not take attendance this semester.

- Again, this is not the easiest course there is.
- If miss a class or classes, you are risking falling behind on the course topics and it will be very difficult to catch up.

Lecture Slides

- Lecture slides will be posted to Canvas.

- Lecture slides will contain:
  - Reading assignments (mostly from the textbook)
  - Optional thinking assignments

- I will *try* to post lecture slides ahead of time so you can go through them before coming to the class
  - This will allow you to come to the class prepared and ask meaningful questions

- If I am unable to post lecture slides earlier, then I will reserve some time during the next lecture for Q&A.

Graded Items

- 5 Assignments (8% each) - Tentative schedule is in the syllabus
- 1 Algorithm Design, Implementation, and Analysis Project (10%) – in Java
- Midterm 1 (15%): Thursday, September 22 (tentative)
- Midterm 2 (15%): Thursday, October 20 (tentative)
- Comprehensive Final Exam (20%): Monday, December 5, 8:00 a.m. - 10:30 a.m.

Course grade

- Based on the accumulated points from all graded items in the previous slides

Total	Letter Grade
[90, 100]	A
[80, 90)	B
[70, 80)	C
[70, 80)	D
[0, 60)	F

- Actual setting in the Canvas is a little more lenient: grades within 0.5 pt boundary will be bumped up to the next letter grade.
  - E.g. 89.5 -> A, 89.4 -> B

- Grades will not be rounded or curved in any other way!
  - Please don’t ask your grade to be rounded at the end of the semester.

Grade appeals

- The first point of contact for all re-grading requests should be the TA.
  - Always cc me so I can be informed, but I will never step in before the TA.

- If you cannot resolve the re-grading with the TA, then an appeal may be made to the instructor.
  - In this case, cc the TA.

- Initial request for re-grading must be made **within a week the grades are posted**.

Make-up

- Except in emergencies, you must inform me beforehand and request Excused Absence Memo from Engineering Student Services (https://aub.ie/EngAbsence).

- In case of emergencies, you must contact me within 48 hours from the time exam was given or an assignment was due. Otherwise, the grade for the missed activity will remain zero.
  - You will still need to obtain Excused Absence Memo.

Classroom policy and Academic Honesty

- Phones should be turned off or in silence/vibrate mode.
  - Calls should be answered outside the classroom.

- Food/drinks are okay as long as it is not disruptive to other students.

- You may use laptops or tablets to take notes or to review class materials; most other uses (e.g. entertainment, social networking) are not condoned.

- In short, we will strictly follow the university policy on classroom behavior. Refer to the student handbook.

- Cheating of any kind is unacceptable!
  - There will be serious consequences per the Academic Honesty Code.

How to make an A

- Attend all classes
- Review the slides, any supplemental materials if provided, your notes and the textbook after each class and before the next class
- Complete ALL reading assignments after each class and before the next class
- Attempt as many thinking assignments as you can after each class and before the next class
- Ask questions in class
- Study with friends
- Practice solving problems – make up and solve problems similar to those in assignments, or problems from the textbook that are listed in the slides
- Do all assignments yourself
- Work regularly, not at the last minute!

I just want a D or F; is that easy?

- Do not keep up with the course on a regular basis
- Study the night before for an exam
- Not take assignments seriously by doing a sloppy job or not doing them at all or cheating

How to Get Help

- Instructor (me)
  - Office hours: MW 9:00 a.m. – 10:30 a.m. via Zoom
    - Other times and in-person meetings are available by appointment
  - Zoom: https://auburn.zoom.us/my/hughkwon
  - Email: hkwon@auburn.edu
    - Do NOT use Canvas messaging, only send from/to AU email

- TA
  - To be determined… TA info will be updated on the syllabus soon.

- Discord
  - I invite all of you to chime in when someone asks questions on something you can answer or want to discuss.
  - https://discord.gg/fDKXhDxAxF

- Engineering Student Services Tutoring
  - COMP 3270 is one of the courses that ESS provides tutoring services for!
  - https://eng.auburn.edu/current-students/tutoring/index

AUBURN UNIVERSITY