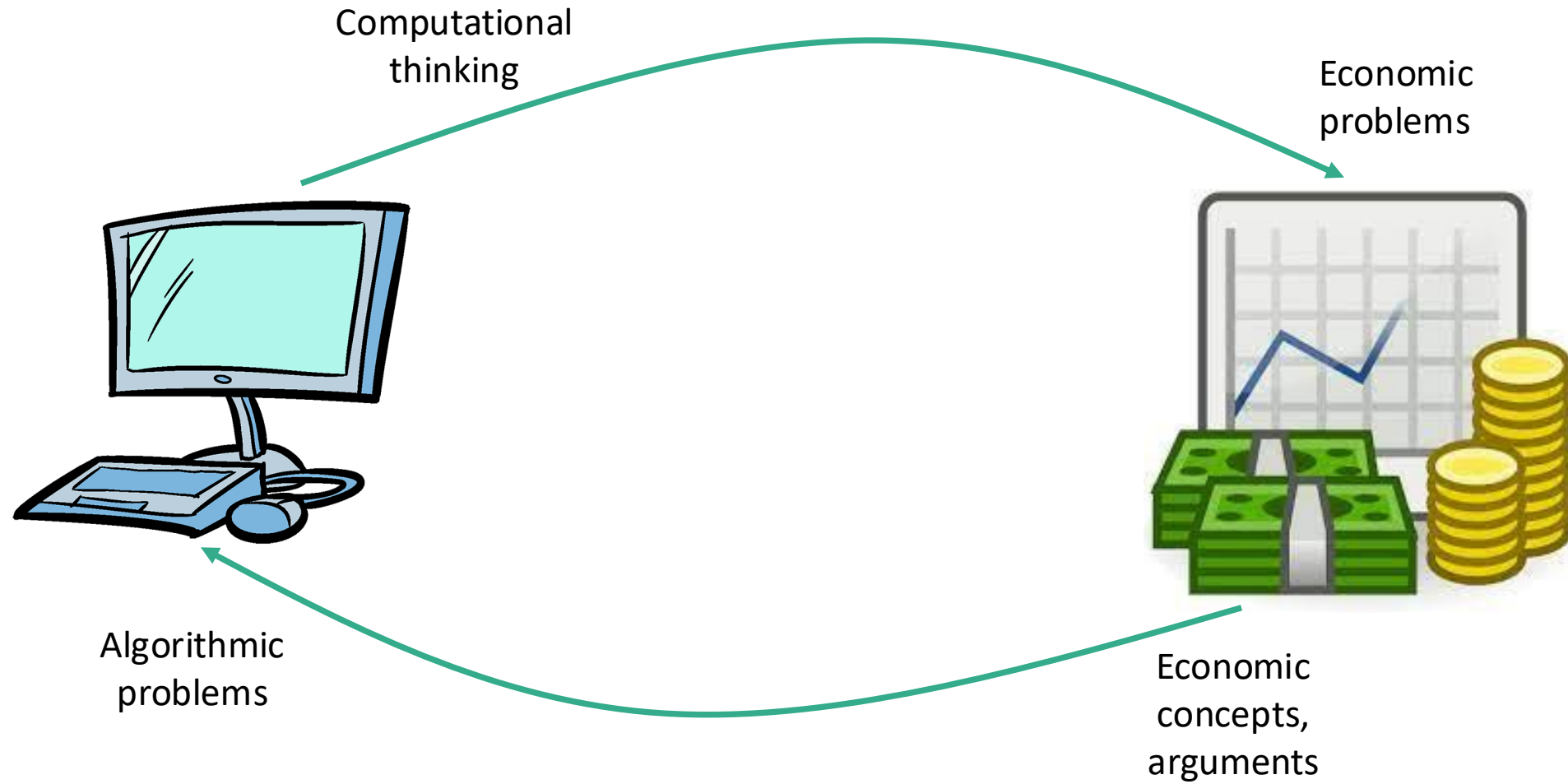


DS 574: Algorithmic Mechanism Design

PROFESSOR KIRA GOLDNER

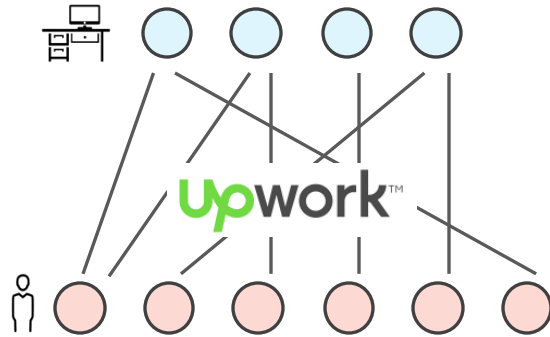
What is “EconCS”?

Also referred to as:
Algorithmic Game Theory (AGT)

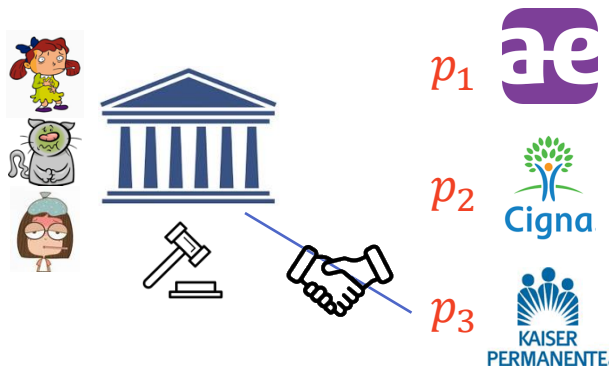


Econ → CS

Online Labor Markets



Health Insurance



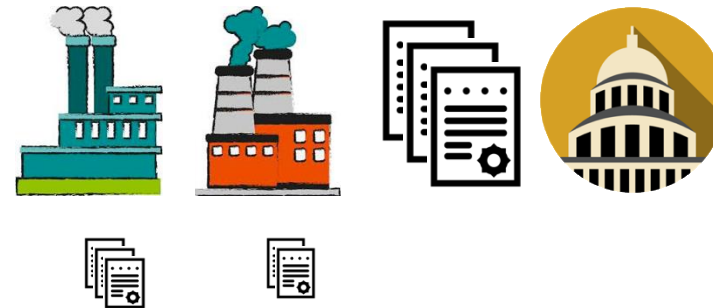
Algorithmic
problems



Economic concepts,
arguments

- The systems interact with **strategic individuals**.
- We must **design** them to be **robust** to **strategic behavior**.

Carbon Emissions



Econ→CS

Input:

Data reported by
strategic agents.



20



15



10

Objective: Maximize
buyer's value

Mechanism

Output:

-who gets what
-who pays (gets
paid) what



15

Use **game theory** to reason about
incentives within the **algorithm**
so that we can **guarantee**
(approximate) optimality.



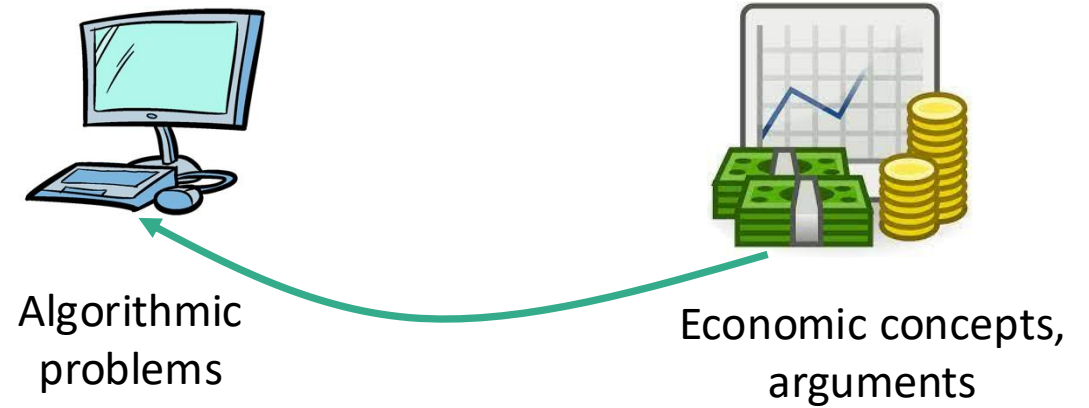
Algorithmic
problems



Economic concepts,
arguments

Econ→CS

Elegant **proofs** using an economic lens:



Maximum weight matching [Demange Gale Sotomayor '86]

- LHS runs ascending auction “bidding” on RHS until perfecting matching achieved.

Online bipartite matching [Karp Vazirani Vazirani '90]

- Algorithm: Randomly permute RHS. LHS arrives and takes first available item in LHS according to permutation.
- Prove this using elegant random price argument. [Eden Feldman Fiat Segal '21]

CS→Econ



1 item



- Simple.
- Easy to compute.
- Only one real option.

[Myerson '81]

\$5: $\Pr[\text{apple}] = 1$

Computational
thinking



Economic
problems



2 items



- Uncountably infinite options.
[Manelli Vincent '07, Daskalakis
Deckelbaum Tzamos '15]
- Intractable to compute. [Daskalakis
Deckelbaum Tzamos '13]
- We still know very little about
how to do this.



\$5.89: $(\Pr[\text{apple}] = .60, \Pr[\text{orange}] = .29)$

CS→Econ

2 items



- Uncountably infinite options. [Manelli Vincent '07, Daskalakis Deckelbaum Tzamos '15]
- Intractable to compute. [Daskalakis Deckelbaum Tzamos '13]
- We still know very little about how to do this.

\$5.89: (Pr[]=.60, Pr[]=.29)

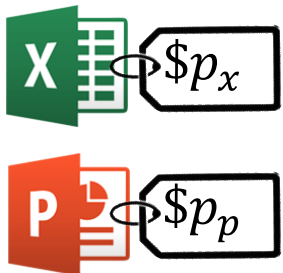
Computational
thinking



Economic
problems



Simple Mechanisms



(Lack of) information

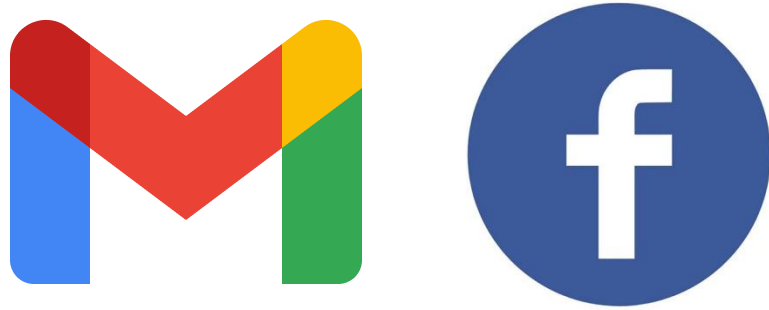
???



Robustness

Why is this important to learn about?

Mechanism Design and Society



Computationally Efficient:

- To design.
- To run.
- To strategize within.



Mechanism Design and Society

Settings where:

- Allocations are a mess.
- There are perverse incentives.

Computationally Efficient:

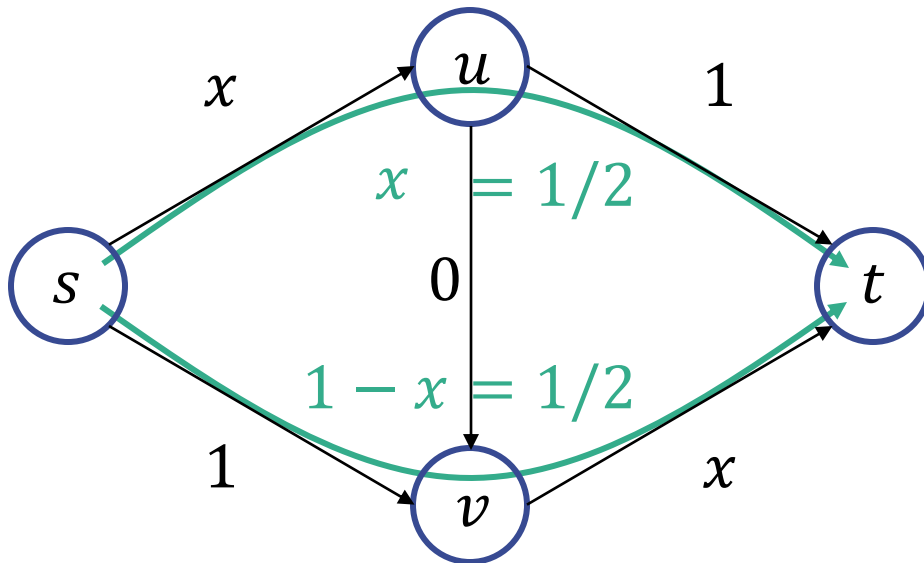
- To design.
- To run.
- To strategize within.

Health Insurance



Shortest Path—Braess's Paradox

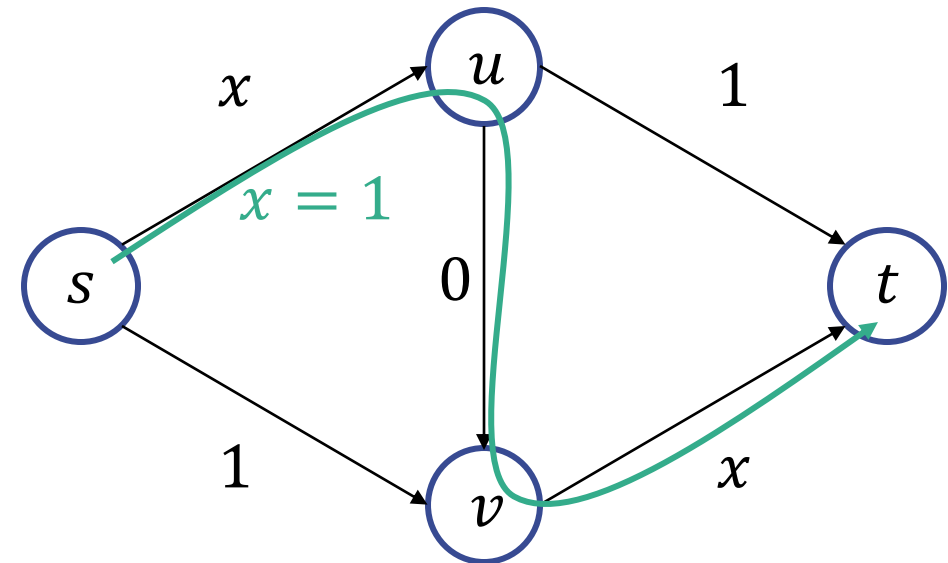
Centralized OPT



Fraction of population on route

Cost (think: time to travel with traffic)

Price of Anarchy (PoA)



Takeaways:

Agents don't choose what's best for them!

Adding a 0-cost road doesn't always help!

What should you expect to learn?

- Mechanism Design basics (welfare, revenue, environments)
 - Similar to other MD/EconCS courses. Probably the only part that is.
- Mechanism Design for Social Good
- Robustness
- New frontiers (two-sided markets, interdependent values, fairness)
- LP Duality applied to mechanism design

Where can you go after this course?

Research in related fields:

- EconCS (from CS)
- Operations Research (IE or Business)
- Microeconomic theory
- Some interdisciplinary split!

Add incentives or an economics perspective to your research:

- Privacy for strategic agents
- Learning with strategic agents

Related industries:

- Platform economics
- Allocation systems in welfare or industry
- Legal regulation (when is regulation better than markets?)



Logistics

Teaching Staff

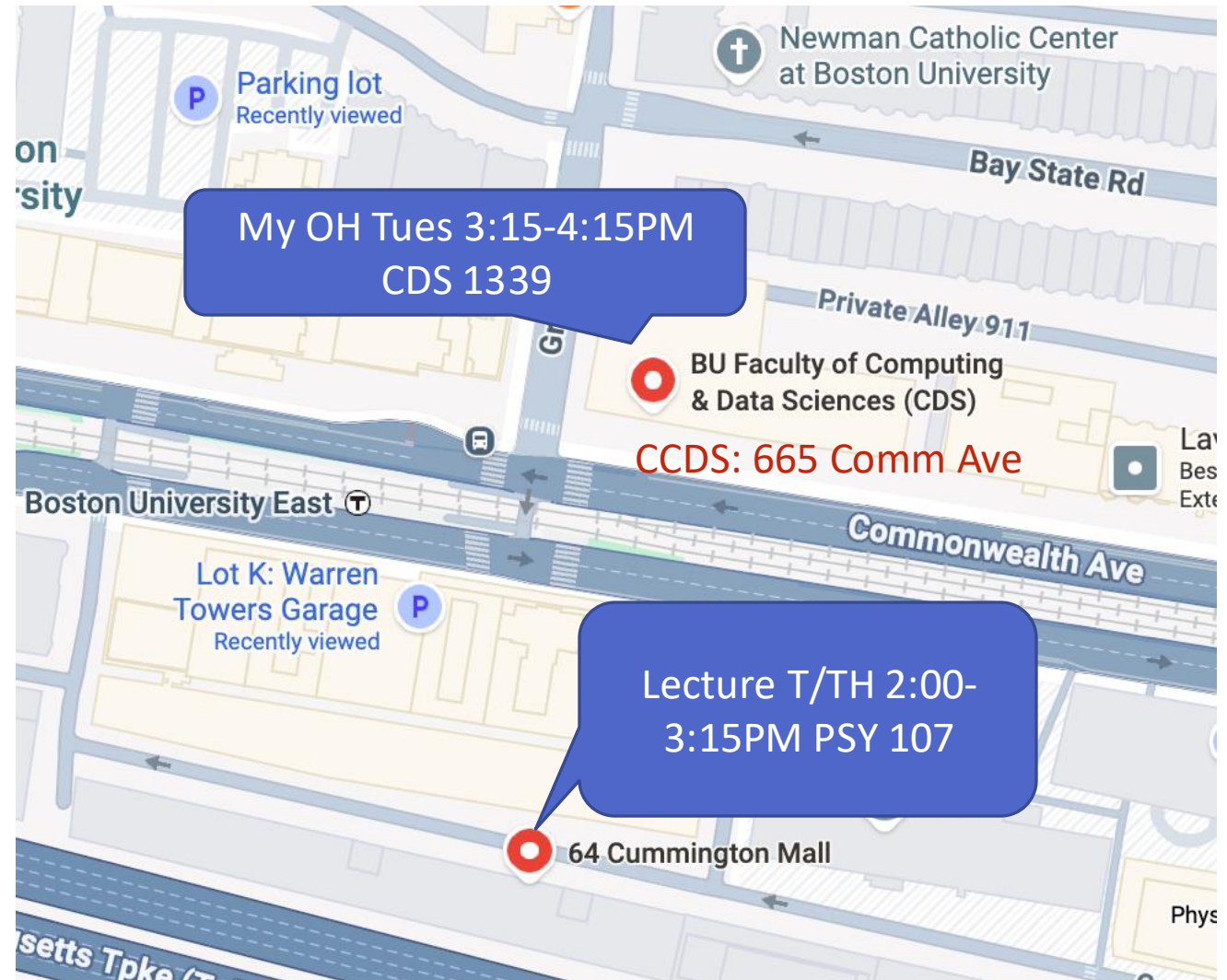
Instructor: Prof. Kira Goldner

Email: goldner@bu.edu

OH: Tues 3:15-4:15PM

& by appointment

Office Location: CCDS 1339



Class Resources

Course website: <https://www.kiragoldner.com/teaching/DS574/>

- Lecture notes, links to everything

Piazza (code “AMD”):

- Class announcements, Q+A, **assignments + solutions**,
use **instead of email (won't respond to email)**
- I am a human who does not live inside the computer!

Gradescope (code “X2RX4Z”):

- Turn in assignments and view grades

Sign up for these if you have not already! (Links on... the course website!)



Website

This is a theoretical problem-solving class

No programming assignments! Evaluation based on problem sets and project.

Prerequisites:

- A first proofs class that's Discrete-Math-esque (DS 121, CS 131, MA 293, ...)
- Undergrad algorithms (DS 320, CS 330, ...)—algorithmic reasoning, runtime and complexity notions
- Intro probability (solid in DS120, 121, 122, and preferably MA 581)—know r.v.s and compute their moments
- Mathematical maturity

Not expected:

- Any background in game theory/incentives/economics.

Evaluation

Homework (30%)

- Collaborative problem sets ~every other week.

Mechanism Design for Social Good problem formulation (15%)

- Formulate a problem and defend why the question is important both for the domain and within mechanism design. Identify a domain expert for potential collaboration.

Class participation (10%)

- In class (participation cards every two weeks) and via Piazza (asking and answering questions—Piazza stats).

Final Project (45%)

- Investigate a research question not covered in class—read papers and write a survey OR do original research. Write up and presentation.

Homework Policies

- Expect to spend at least 10 hours per week on homework.
- **Late policy:** You have 4 late days, max 2 per assignment (integer numbers used only). No exceptions. You don't get extra later if you're sick!
- Type up homework with **LaTeX**.
- Turn in via **gradescope**. Due at 11:59pm on Wednesdays (typically).
- **Regrades:** Requests within 7 days, only via gradescope, with explanation/argument. Only for **incorrect** grading (not insufficient credit). If you request a regrade, the whole assignment/exam may be regraded, and your score may go up or down. **Do not use these to ask for feedback.**

Collaboration Policy

Collaboration is encouraged!!!

- You may work with up to **three** classmates on an assignment. **List your collaborators' names on your assignment.** (E.g., Collaborators: None.)
- Good rule: **Nobody should leave the room with anything written down.** If you really understand, you should be able to reconstruct it on your own.
- You may **not** use the internet or ChatGPT on homework problems. You may use course materials and the recommended readings from textbooks.

I believe **strongly** in learning over evaluation, learning via collaboration, and academic integrity. **Please adhere to BU's academic conduct policy.**

Generative AI Policy

First, I do not use LLMs in the design or preparation of course materials.

You are **not permitted** to use LLMs to solve homework or generate writing for class assignments.

You **may** use them **like a search engine**: to point you toward other resources (papers, LaTeX commands, etc.).

If you are unsure whether LLM use is permitted in your use case, **ask**, otherwise, the assumption is that it is not permitted. **Don't understand why, have other opinions, want to talk about? Happy to!**

Violation of this policy is a violation of BU's academic integrity policy.

Class Etiquette

We strive toward an accessible and equitable classroom for all students.

- Raise your hand.
- Be conscious of how often you participate (in class and in collaboration).
 - Don't talk over others, leave room for other voices if you speak up a lot, and speak up more if you do not.
- Use your participation card to estimate.

But also

- Ask questions!!!

Best advice I ever got was to just ask and not wait to fill in gaps myself later.

Class Time

Date	Topic	Resources
Sep 6	Overview and Policies, Intro to AGT	Slides , Worksheet , Notes , R1.1-2
Sep 8	Incentive Compatibility	Worksheet , Notes , R1.3
Sep 13	The Revelation Principle	Worksheet , Notes , R1.4, H2

DS 320 Algorithms for Data Science
Spring 2023

Lecture #1 Worksheet
Prof. Kira Goldner

Covered in introduction slides:

- Course policies (also in syllabus).
- What to expect in this class (also in FAQ).
- Sample of content we'll cover.

Runtime Review

In runtime analysis we do an informal accounting. We count basic operations (algebra, array assignment, etc) as constant time.¹

Analyze the runtime of the following algorithm:

Algorithm 1 FindMinIndex($B[t+1, n]$).

```
Let MinIndex =  $t + 1$ .
for  $i = t + 1$  to  $n$  do
  if  $B[i] < B[\text{MinIndex}]$  then
    MinIndex =  $i$ .
  end if
end for
return MinIndex.
```

Which operations are constant-time?

Are there any loops? How many times do they run?

How does everything come together?

Which factors dominate asymptotically?

¹This isn't quite right—for example, multiplication of large numbers should scale with the bit complexity—but is a good approximation for us.

- Worksheet listed in advance on website
- **Bring worksheet to class** (on iPad, printed, etc)
- Lecture + exercises
- Notes posted after class

DS 320 Algorithms for Data Science
Spring 2023

Lecture #1
Prof. Kira Goldner

Covered in introduction slides:

- Course policies (also in syllabus).
- What to expect in this class (also in FAQ).
- Sample of content we'll cover.

Runtime Review

When we analyze runtime, we'll do an informal accounting. We'll count basic operations (algebra, array assignment, etc) as constant time.¹

We will analyze the runtime of the following algorithm:

Algorithm 1 FindMinIndex($B[t+1, n]$).

```
Let MinIndex =  $t + 1$ .
for  $i = t + 1$  to  $n$  do
  if  $B[i] < B[\text{MinIndex}]$  then
    MinIndex =  $i$ .
  end if
end for
return MinIndex.
```

Each of the following lines is a unit (constant-time) operation:

- **Let** MinIndex = $t + 1$.
- **if** $B[i] < B[\text{MinIndex}]$ **then**
- MinIndex = i .

The for-loop runs $n - t$ times (notice that both n and t are variables as they are in our input). Thus the runtime of this algorithm is $O(n - t)$.

Asymptotic Notation

Definition 1 (Upper bound $O(\cdot)$). For a pair of functions $f, g : \mathbb{N} \rightarrow \mathbb{R}$, we write $f \in O(g(n))$ if there exist (\exists) constants c_1, c_2 such that for all (s.t. \forall) $n \geq c_1$,

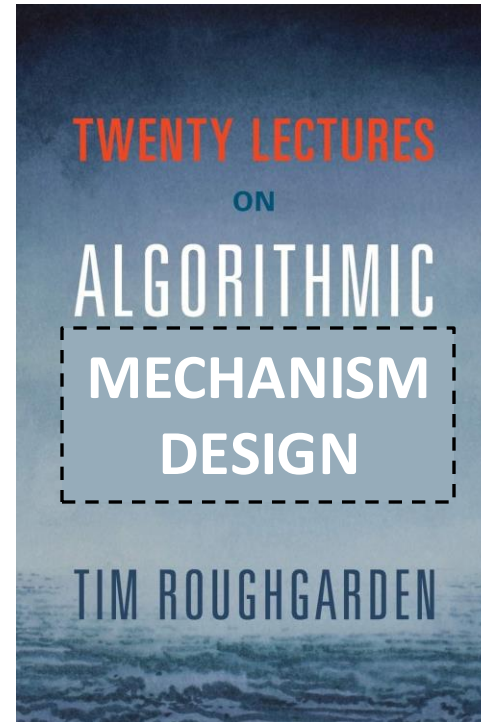
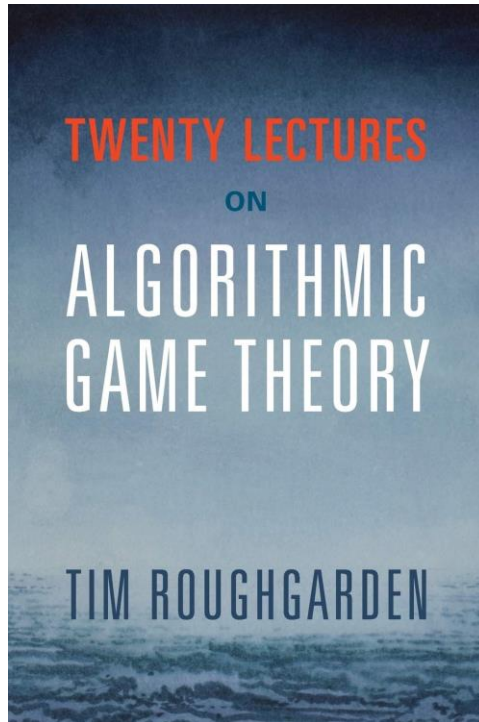
$$f(n) \leq c_2 g(n).$$

We'll often write $f(n) = O(g(n))$ because we are sloppy.

¹This isn't quite right—for example, multiplication of large numbers should scale with the bit complexity—but is a good approximation for us. We will analyze runtime by counting these operations.

Book

There is no required textbook, and the lecture notes will be self contained. But many of the topics we are covering are well covered in standard algorithms textbooks; some lectures are adapted from Tim Roughgarden's lecture notes.



Let's get started!

Game Theory Basics

Player 2:
Column Player Collin

	Rock	Paper	Scissors
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissors	-1,1	1,-1	0,0

Player 1:
Row Player Ron

Actions = {Rock, Paper, Scissors}

Pure strategy: pick 1 action

Mixed strategy: probability distribution over actions

Algorithms with Incentives

