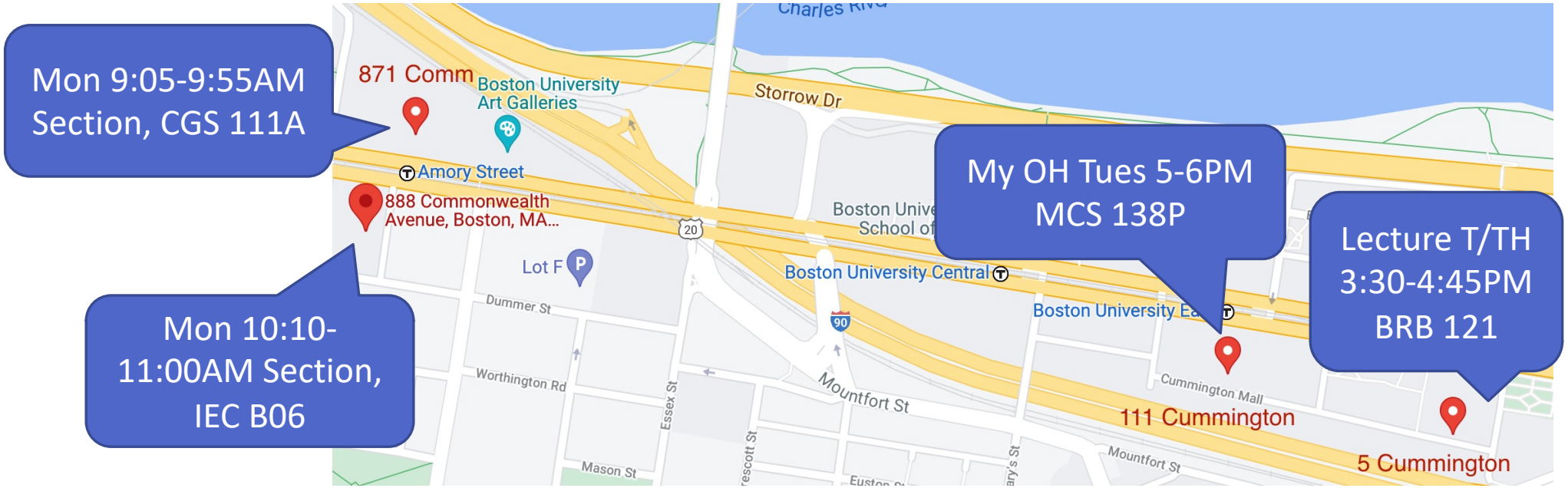
# DS 320: Algorithms for Data Science

PROFESSOR KIRA GOLDNER

# Teaching Staff

Instructor: Prof. Kira Goldner
Email: goldner@bu.edu
OH: Tuesday 5-6PM and by appointment
Office Location: 111 Cummington Mall, 138P

TF: Habeen Chang
Email: ahc98@bu.edu
OH: TBD



# Class Resources

Course website: https://www.kiragoldner.com/teaching/DS320/
- Lecture notes, links to everything

Piazza:
- Questions and answers
- I am a human who does not live inside the computer!

Gradescope:
- Turn in assignments and view grades

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

Also! I am open to suggestions on how to best utilize things like Piazza!

# This is a theoretical problem-solving class

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

Prerequisites:

- Intro programming (DS 110, CS 111, …)
- A first proofs class that's Discrete-Math-esque (DS 122, CS 131, MA 293, …)

Not required but might make you more comfortable:

- Data structures and algorithms (DS 210, CS 112, …)
- More proof classes

# How is this Algorithms for *Data Science*?

- Still the same skills and basic methods and typical algorithms course (sorting, greedy, divide and conquer, dynamic programming, max flow)
- Focus more on DS-relevant applications (i.e. Fast Fourier Transform)
- Focus more on methods and applications relevant in data science (multiplicative weights, linear programming)

# Evaluation

Homework (45%)
- ~Weekly problem sets

Midterm Exams (30%)
- Two midterm exams, worth 15% each. (Approx Feb 24-March 1 and March 31-April 5)

Final Exam (15%)
- Closed-book during Finals period at our scheduled time.

Class participation (5%)
- In class and via Piazza (asking and answering questions) gets 100% here.

Peer Grading (5%)
- One session grading homework per person during the semester.

# 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.
- Lowest homework will be dropped at the end of the semester.
- Type up homework with L**aTeX**.
- Turn in via **gradescope**. Due at 11:59pm on the date assigned.

- **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.

# Collaboration Policy

Collaboration is encouraged!!!

- You may work with up to two classmates on an assignment. **List your collaborators' names on your assignment. (E.g., Collaborators: None.)**
- Good rough 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 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.

# Midterms

Two midterm exams, worth 15% each.
Tentative dates: Feb 24-March 1 and March 31-April 5

Essentially the same format as homework, but no collaboration allowed and covering slightly more material.

Think of them as solo problem sets to prove you can do them by yourself.

# Peer Grading (5%)

Everyone will help grade homeworks once during the semester.

It is a valuable opportunity to learn how others write proofs and what we look for in arguments!

# Class Etiquette

I 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.
- I’m always open to new strategies here.

But also
- Ask questions!!!!!!

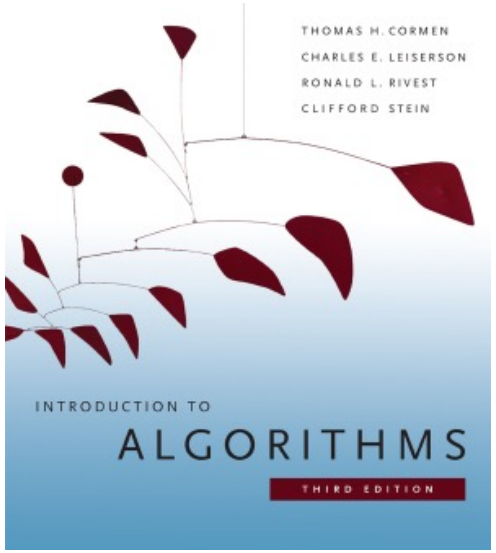
Best advice I ever got was to just ask and not wait to fill in gaps myself later.

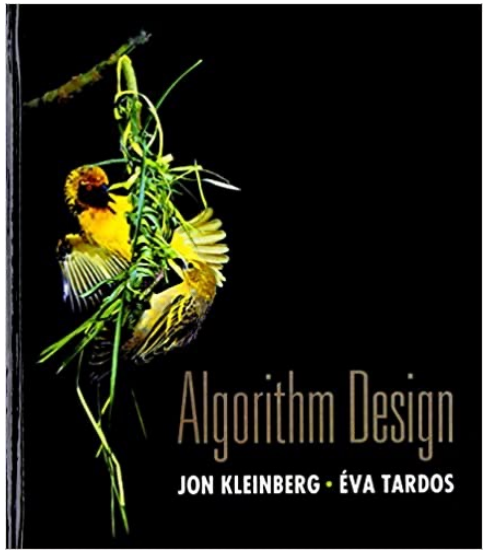
# Pandemic Etiquette

- Green-badge
- N95/surgical-mask at all times
- Distance as possible
- Do not come if sick!!
  - I will make the course accessible if you need to miss for COVID.

# 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 Kleinberg and Tardos.



# What should you expect to learn?

Skills:

- Getting comfortable understanding and writing formal definitions and statements.
- Creative problem solving and thinking algorithmically.
- Writing clear and convincing arguments.
- Domain-specific skills: Identifying algorithmic problems within applications; determining when to apply which technique; analyzing runtime.

IMO, skills are more important and course knowledge, so your time is much better spent engaging with homework problems than on reading additional material.

# Runtime Analysis

Analyze in the worst-case, for the biggest instances.

| | $n$ | $n \log_2 n$ | $n^2$ | $n^3$ | $1.5^n$ | $2^n$ | $n!$ |
|---|---|---|---|---|---|---|---|
| $n = 10$ | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | 4 sec |
| $n = 30$ | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | 18 min | $10^{25}$ years |
| $n = 50$ | < 1 sec | < 1 sec | < 1 sec | < 1 sec | 11 min | 36 years | very long |
| $n = 100$ | < 1 sec | < 1 sec | < 1 sec | 1 sec | 12,892 years | $10^{17}$ years | very long |
| $n = 1{,}000$ | < 1 sec | < 1 sec | 1 sec | 18 min | very long | very long | very long |
| $n = 10{,}000$ | < 1 sec | < 1 sec | 2 min | 12 days | very long | very long | very long |
| $n = 100{,}000$ | < 1 sec | 2 sec | 3 hours | 32 years | very long | very long | very long |
| $n = 1{,}000{,}000$ | 1 sec | 20 sec | 12 days | 31,710 years | very long | very long | very long |

# An Arsenal of Algorithmic Techniques

Greedy Algorithms
- Make myopic choices. Very fast. Works when optimal solutions satisfy a certain "exchange" property.

Divide and Conquer
- Figure out how to quickly stitch together two (or more) optimal solutions to sub-problems. Recursively solve the sub-problems.

"Dynamic Programming" (actually Divide and Conquer++)
- The naïve recursion might have exponential size, but if we have only polynomially many *distinct* sub-problems, we can just cache the solutions to avoid wasted effort.

# + Continuous Optimization (“ML”)

Linear Programming

- Powerful framework for optimizing linear functions subject to linear constraints. Closely related to online optimization and zero sum games.

Multiplicative Weights

- For online optimization—obtains guarantees for adversarial sequences of loss functions

Gradient Descent

- For optimizing continuous, differential functions. Quickly converges to the optimal solution for convex problems, and to stationary points for nonconvex problems.

# Impossibilities & Approximation

Formal statements that you can do no better with a solution.

- E.g., the knapsack problem is NP-complete.
- If you could find a polynomial-time algorithm for it, then you could solve all these other algorithms in poly-time.

Approximation algorithms

- E.g. an algorithm that is fast and provably always get at least 1/2 as good as the optimal.

# Where can you go after algorithms?

- Coding interviews
- Better problem solver in general, whether in code or puzzle hunts
- Better formal thinking and writing
- More advanced toolkits (e.g., streaming, algorithmic game theory)