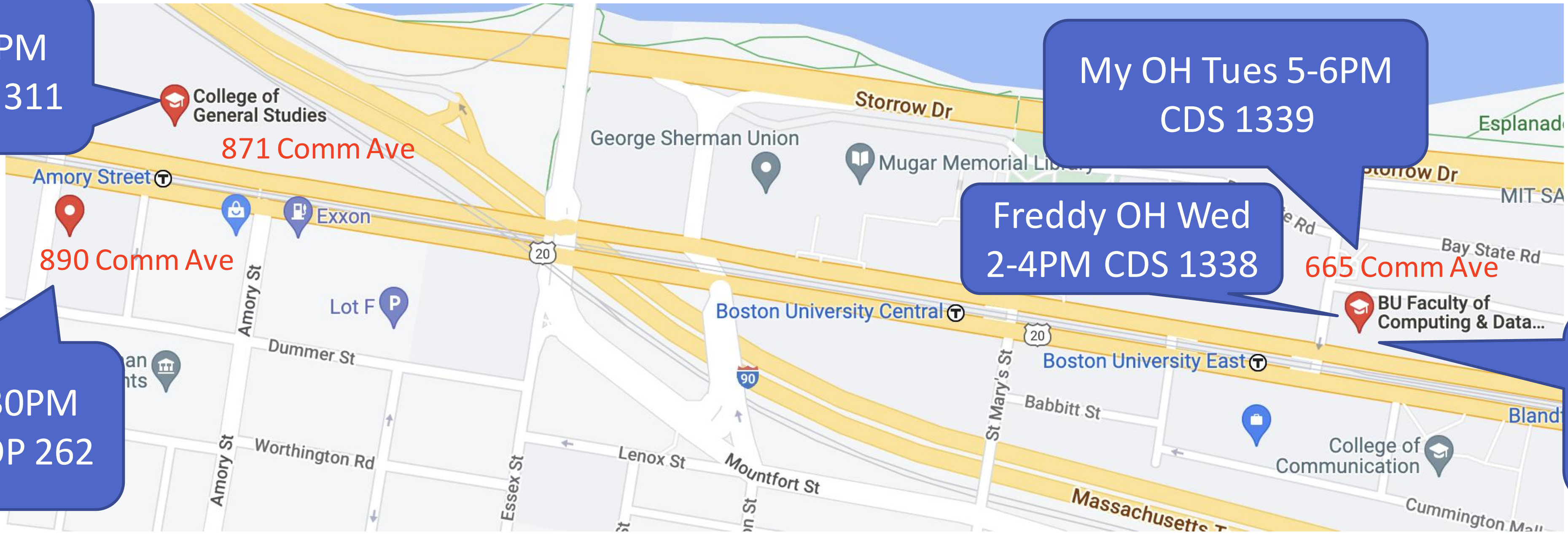
DS 320: Algorithms for Data Science

PROFESSOR KIRA GOLDNER

Teaching Staff

Instructor: Prof. Kira Goldner
Email: goldner@bu.edu
OH: Tuesday 5-6PM and by appointment
Office Location: CDS 1339, 665 Comm Ave

TF: Freddy Reiber
Email: freddyr@bu.edu
OH: Wednesday 2-4PM
Location: CDS 1338



Mon 3:30-4:25PM
Discussion, CGS 311

Mon 4:40-5:30PM
Discussion, EOP 262

My OH Tues 5-6PM
CDS 1339

Freddy OH Wed
2-4PM CDS 1338

Lecture T/TH
3:30-4:45PM
CDS 164

Today

syllabus

DS 320: Algorithms for Data Science — Spring 2023

Instructor: Prof. Kira Goldner
Email: goldner@bu.edu
Office Hours: Tuesday 5-6PM and by appointment
Office Location: CDS 1339, 665 Commonwealth Ave

Lectures: Tuesday and Thursday 3:30-4:45PM, CDS 164
Discussion Section A2: Monday 3:30-4:25PM, CGS 311
Discussion Section A3: Monday 4:40-5:30PM, EOP 262

Teaching Fellow: Freddy Reiber
Email: freddyr@bu.edu
Office Hours: Wednesday 2-4PM
OH Location: CDS 1338, 665 Commonwealth Ave

Course Description: This course covers the fundamental principles underlying the design and analysis of algorithms. We will walk through classical design methods, such as greedy algorithms, design and conquer, and dynamic programming, focusing on applications in data science. We will also study algorithmic methods more specific to data science and machine learning. The course places a particular emphasis on algorithmic efficiency, crucial with large and/or streaming data sets, for which multiple scans of data are infeasible, including the use of approximation and randomized algorithms.

Prerequisites: Required prerequisites are DS-110 and DS-122 or equivalent. Equivalents include for DS-110: CS-111, and for DS-122: CS-131 or MA-293. This is a **theoretical problem-solving** and **proof-writing** course. Additional useful background may include Discrete Math, Combinatorics, Linear Algebra, Data Structures and related algorithms, and Probability (none required, but the more the better).

Course website: `https://www.kiragoldner.com/teaching/DS320/`. There will also be a Piazza website for the course: `http://piazza.com/bu/spring2023/ds320/home` (access code: algs).

BU Hub: This course satisfies **Quantitative Reasoning II** and **Toolkit Critical Thinking**.

Quantitative Reasoning II: Throughout this course, we will build a toolkit of quantitative tool for solving algorithmic problems and proving correctness. Students will face complex problems from a breadth of applications and determine when each tool is appropriate and how to use it to design and analyze algorithms. They will learn to communicate clear and logically correct arguments in proofs. This will build on prior math and proof skills from discrete math and combinatorics (DS-120, DS-121, and DS-122 or equivalent).

FAQ

What should you expect to learn from DS 320?

Skills: Skills will be demonstrated in lectures and suggested readings, and will be practiced in homeworks, exams, and discussion sections.

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

Knowledge: Lectures will cover all methods and proof techniques that you are expected to know; suggested readings will go into further details.

5. Specific algorithmic methods.
6. Specific proof techniques.

How is this course different from CS 330? How is it Algorithms *for Data Science*? This course will parallel a typical introduction to algorithms course—the skills and basic methods covered in one—while focusing more on methods and applications that are most relevant in data science. We will cover basics from a typical algorithms course, such as sorting, greedy, divide and conquer, dynamic programming, max flow. Within the more standard topics, some of our applications will be focused more on methods and applications relevant in Data Science, i.e. Fast Fourier Transform. The more "standard" topics will be somewhat abbreviated, and in the second-half of the course, we will foray more into concepts imperative for handling data such as multiplicative weights, linear programming, etc.

Which is most important? In my opinion, it is significantly more importantly to develop skills than to learn specific knowledge. This means, in my opinion, that your time is much better spent engaging with homework problems than on reading additional material. In my opinion, the skills are listed in decreasing order of importance, so $1 > 2 > 3 > 4$.[1] In this course, you should develop as a problem-solver. **The goal of the course is not for you to learn specific solutions to interesting problems, but to learn *how* to solve interesting problems.** This is a slight oversimplification, but hopefully makes the distinction clear.

Will I need to know lots of math? No, but you'll need to engage deeply with formal arguments and sometimes basic probability. This isn't a math class, and the goal isn't to teach you math. Some problems will require you to be creative with math, but nothing too advanced.

On the flip side, some problems may be challenging just to phrase as a math question, and you should expect to spend a little bit of effort just to figure out exactly what some problems are asking.

[1] But this isn't universal: for instance, if you're a math major taking four classes a semester that grill 1 and 3, probably you should hope to learn most about 2 and 4.

worksheet #1

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[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?

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

Class Resources

Course website: https://www.kiragoldner.com/teaching/DS320/

- Lecture notes, links to everything

Kira Goldner | About | Research | Teaching | Diversity & Service | Resources | Blog

DS 320: Algorithms for Data Science
Spring 2023
Boston University

Course Details

Instructor: Professor Kira Goldner (goldner@).
Office Hours: Tuesday 5–6pm and by appointment.
Office Location: CDS 1339, 665 Comm Ave.

Teaching Fellow: Freddy Reiber (freddyr@).
Office Hours: CDS 1338.
OH Location: Wednesday 2–4pm.

Lectures and Discussion Sections:
Tuesday/Thursday 3:30–4:45pm, CDS 164.
Discussion Section A2: Monday 3:30-4:25pm, CGS 311.
Discussion Section A3: Monday 4:40-5:30pm, EOP 262.

Important Links:

- Course Policies: Syllabus
- For communication: Piazza (access code: algs)
- For submitting homework: Gradescope (entry code: V5PJW4)
- What to expect from this course: FAQ

Website

Class Resources

Website

Course website: https://www.kiragoldner.com/teaching/DS320/

- Lecture notes, links to everything

Piazza (code "algs"):

- Class announcements, Q+A, **assignments + solutions**, basically **instead of email (I am terrible at email)**
- I am a human who does not live inside the computer!

Piazza

Gradescope

Gradescope (code "V5PJW4"):

- Turn in assignments and view grades

Sign up for these if you have not already!

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)

- Focus more on DS-relevant applications (i.e. Huffman Codes)

- 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 22-27 and March 29-April 3)

Final Exam (20%)
- Take-home from last day of classes until our scheduled time. (Back up: closed-book in-class.)

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

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 **LaTeX**.
- Turn in via **gradescope**. Due at 11:59pm on the due date, typically Wed.

- **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.

Type up homework with LaTeX

- Slight learning curve! May want to use Overleaf (overleaf.com).

Asymptotic Notation

Definition 1 (Upper bound $O(\cdot)$)**.** For a pair of functions $f, g : \mathbb{N} \to \mathbb{R}$, 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.

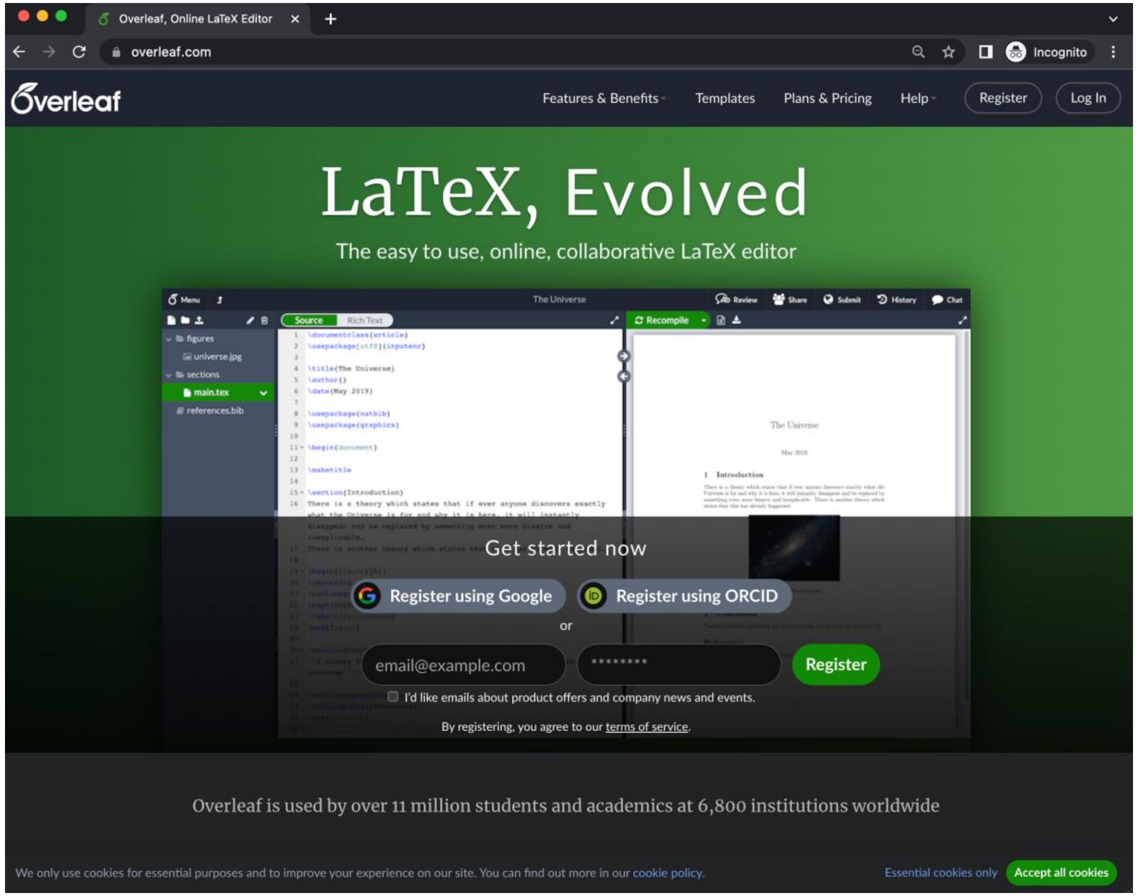
Translation: For large n (at least some c_1), the function $g(n)$ dominates $f(n)$ up to a constant factor.

Definition 2 (Lower bound $\Omega(\cdot)$)**.** For a pair of functions $f, g : \mathbb{N} \to \mathbb{R}$, we write $f \in \Omega(g(n))$ if there exist constants c_1, c_2 such that for all $n \geq c_1$,

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

Definition 3 (Tight bound $\Theta(\cdot)$)**.** For a pair of functions $f, g : \mathbb{N} \to \mathbb{R}$, we write $f \in \Theta(g(n))$ if $f \in O(g(n))$ and $f \in \Omega(g(n))$.

Exercise: True or False?



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 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 22-27 and March 29-April 3

Essentially: **the same format as homework, but no collaboration allowed and cumulative material.**

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

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.

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 **Lecture #1 Worksheet**
Spring 2023 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?

[1] This 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 **Lecture #1**
Spring 2023 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 $f, g : \mathbb{N} \to \mathbb{R}$, we write $f \in O(g(n))$ if there exist ($\exists$) constants c_1, c_2 such that for all (s.t. $\forall$) $n \geq c_1$,

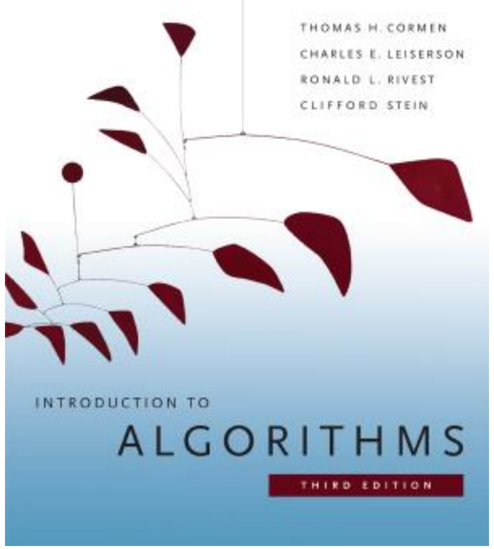
$$f(n) \leq c_2 g(n).$$

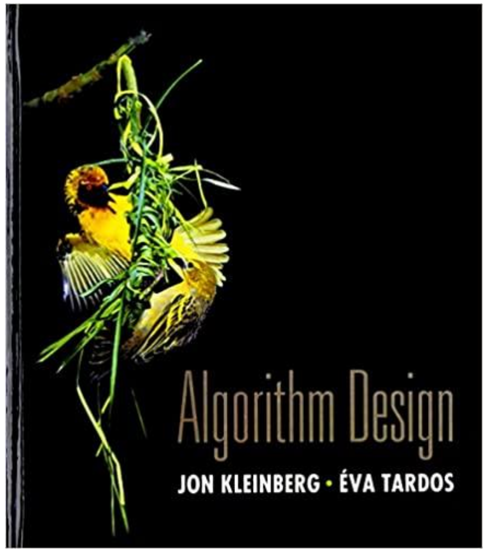
We'll often write $f(n) = O(g(n))$ because we are sloppy.

[1] This 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 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** than course knowledge, so your **time is much better spent engaging with homework problems** than on reading additional material.

The Study of Efficient Algorithms

ALWAYS INCLUDE RUNTIME AND CORRECTNESS

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.

Randomized Algorithms

- When and how randomization can improve upon deterministic guarantees.

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
  - which solution to apply when and ***why*** it's better
- Better formal thinking and writing
- More advanced toolkits (e.g., streaming, algorithmic game theory)