Here Early?

Here for CSE 417?

Welcome! You're early!

Want a copy of these slides to take notes?

You can download them from the webpage cs.uw.edu/417

Want to be ready for the end of the lecture?

Download the Activity slide from the same place

Go to pollev.com/cse417 and login with your at-uw email

You should see something about "presentation hasn't started yet"

Welcome

CSE 417 Winter 21
Lecture 1

Today

Logistics

What is this course?

Start of the content

Zoom Logistics

I'll pause every few minutes to check chat for questions.

If there are unanswered questions, I'll answer a couple on Ed later today (and can take more questions right after lecture as well)

If you're comfortable (and have the wifi) to turn on your video please do

Nodding/confused looks/glazed over eyes help me know if I said something super confusing.

We will put recordings of lecture on Panopto.

Usually takes until the evening to upload (zoom and panopto do some processing)

Staff

Instructor: Robbie Weber

Ph.D. from UW CSE in theory
First year as teaching faculty
Instructor for 373 two years ago

Office: CSE2 311
Email: rtweber2@cs.washington.edu

TAs

Josh Curtis

Alex Fang

Ruthvik Sai Mondreti

Ansh Nagda

William Nguyen

Howard Xiao

Joyce Zhou

Syllabus

It's all on the webpage:
https://courses.cs.washington.edu/courses/cse417/21wi/

We're supporting students who wish to take the course asynchronously.

If there's something we're not doing that we should be, please contact us!

Google "getting to know you" form is out – please fill out by Wednesday.

Textbook

Optional: Algorithm Design by Kleinberg & Tardos

It's a good introduction, and nice as a reference.

There are lots of other books:

Introduction to algorithms by Cormen, Leiserson, Rivest, Stein

One free reference: Algorithms by Jeff Erickson Algorithms.wtf

All are theoretical (expect more math background than 373).

Additional resource:

Lecture videos by Tim Roughgarden (Algorithms Illuminated)

Not a perfect match of topics, but math background matches.

Logistics – Work

Homework (70%): approximately 8. Most one-week (some two-weeks).

Mostly theory ("how would you solve this problem/why does it work/what's the big-O for an algorithm")

But also with some applied problems (actually write the code and debug it)

Three **"mini-projects"** (10%) on real-world effects of algorithms.

Read a blog post or paper that focuses on theory and think about how it might affect people in the real world.

Lecture activities (7.5%)

Final (given during finals week). (12.5%)

Exact rules TBA, but we won't be proctoring you for it, and you'll have at least 24 hours.

Logistics – Lecture Activities

I'm going to be experimenting with **active learning** in this course.

Why? Because it works.

https://www.pnas.org/content/111/23/8410 a meta-analysis of 225 studies.

Just listening to me isn't as good for you as listening to me then trying problems on your own and with each other.

To reward you for participating in those activities (beyond the increased learning) we're going to award points for **participating** in polleverywhere questions. (don't have to get them right)

The answers live help me adjust explanations.

If you miss a lecture or are participating asynchronously, there's a replacement on canvas.

All the replacements for a given week are due Sunday.

Logistics – where to go?

Slides, homework problems, etc. go up on the webpage

Homework submission on gradescope

Live lecture activities on polleverywhere

Lecture activity replacements on canvas

Questions on Ed discussion board

Don’t trust canvas – we won’t be updating frequently. We’ll tell you when we’re using it for specific purposes.

Late Policy

You have 4 late days to use during the quarter. One late day lets you submit a homework or mini-project up to 24 hours late.

No submissions will be taken more than 48 hours after the listed deadline (except in extenuating circumstances)

You won't be allowed to use late days on the final.

If you have any late days left over after homeworks, we will count each late day as a replacement for one lecture activity.

If you have an unexpected situation that interferes with your ability to meet a deadline, email Robbie as soon as possible and we'll discuss options.

Hey, We're in a pandemic

The staff is going to do our best to help you learn.

Real life is going to get in the way. If it does, tell us as soon as possible, and we'll work with you.

I don't need to know private details, just enough to know it's an emergency and how to help.

We will endeavor not to make any substantial changes to the syllabus.

But if something extremely unexpected happens we reserve the right to make changes.

Generally prefer individual accommodations, rather than course-wide ones.

What is this course?

Algorithms and Computational Complexity

themes:

“Design Techniques” – not just “here’s an algorithm” but “here’s a way of thinking about a class of algorithms”

“Modeling” – In the real world, no one will say “I need you to run Prim’s algorithm on this graph” they will say “I need you to choose where to build electrical wires so every town is connected to the power plant as cheaply as possible

“Set realistic expectations” – there are some things we (think/know) computers can’t do efficiently. How do you recognize these problems?

“Reductions” – if you’ve already solved a problem, don’t solve it again (reuse ideas) and if you know you can’t solve a problem, what else can’t you solve.

What is this course **not**

NOT: A list of the fastest-known algorithms for common problems.

I'm not concerned with which library is best.

The best library changes over time and by language.

I'm not qualified to keep a list.

I want you to find this course useful 5 years from now.

And the best theoretical algorithms probably aren't practical…

and when they are, it's often clever combinations/complicated variants of big ideas that we'll see.

Course Topics (Tentative)

Stable Matchings
Graph Algorithms from 373 (BFS/DFS, MST algorithms, shortest paths)
Greedy algorithms
Divide & Conquer
Dynamic Programming
Network Flow
Linear Programming
P/NP
Approximation Algorithms (applications of all the prior big ideas)
Undecidable Problems

What's Coming Up

This week: An extremely useful algorithm.

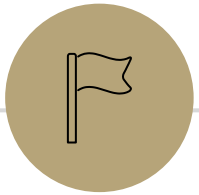
That has had lots of effect on the real world.

Along the way: how to argue that your code works!

We'll learn: proof by induction, proof by contradiction, and proving an implication.

Already know them? You'll get to learn a fun algorithm along the way!

Don't know them yet? The introduction will be fast! More resources on the webpage.



Stable Matchings

Stable Matchings

Motivation:

You have to assign TAs to instructors.

Two groups of people you need to pair off, with preferences about their matches.
You can't make everyone happy…so at least ensure that everyone listens to you.

There are lots of other similar applications

- Assign doctors to the hospitals where they do residency.
- Assign high schoolers to magnet schools.
- Among many, many other applications.

Motivation

The real world is complicated.
Students shouldn't TA a course they haven't taken.
Instructors need varying numbers of TAs.
There are more TA applicants than positions.
Doctors might want to be in the same city as their partner.

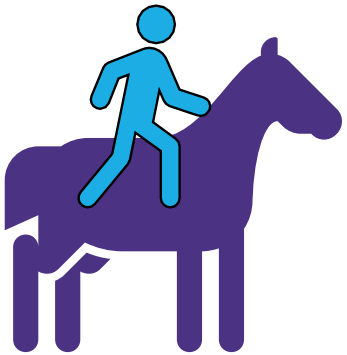
We're going to simplify away the real world constraints.
The core ideas have been adapted to all of these scenarios.

Stable Matching Problem

To simplify. We have two sets:
A set of n horses, and a set of n riders.

Every rider can ride any horse, and vice versa.

We just need to pair them off. What could go wrong?



Stable Matching Problem

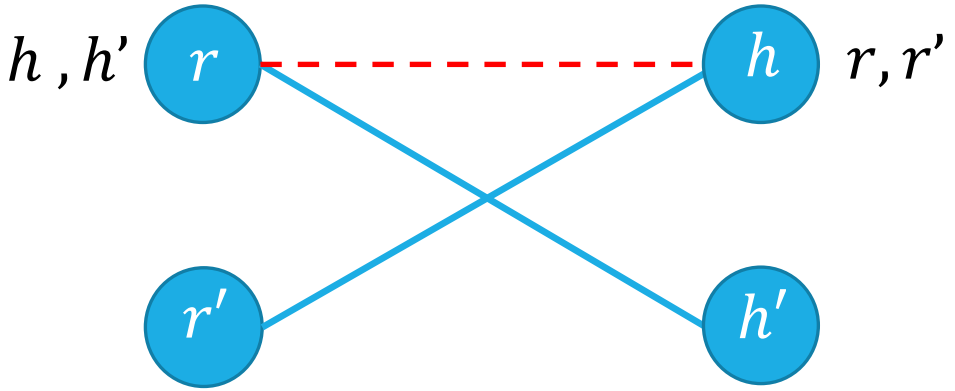
Given two sets $\mathrm{R} = \{r_1, \ldots, r_n\}, H = \{h_1, \ldots, h_n\}$

each agent ranks **every** agent in the other set.

Goal: Match each agent to **exactly one** agent in the other set, respecting their preferences.

How do we "respect preferences"?

Avoid blocking pairs: unmatched pairs (r, h) where r prefers h to their match, and h prefers r to its match.



Stable Matching, More Formally

Perfect matching:

- Each rider is paired with exactly one horse.
- Each horse is paired with exactly one rider.

Stability: no ability to exchange

an unmatched pair r-h is blocking if they both prefer each other to current matches.

Stable matching: perfect matching with no blocking pairs.

Stable Matching Problem

Given: the preference lists of n riders and n horses.
Find: a stable matching.

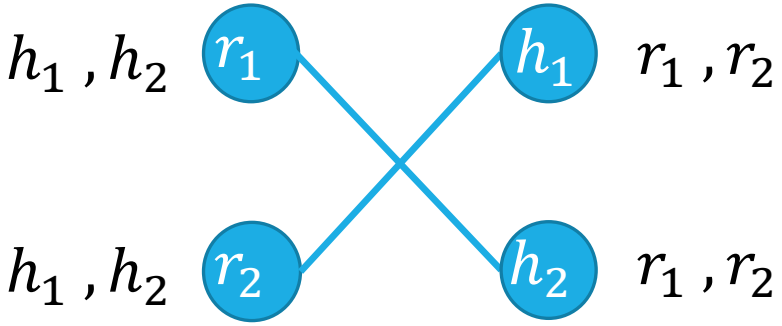
Lecture Activity

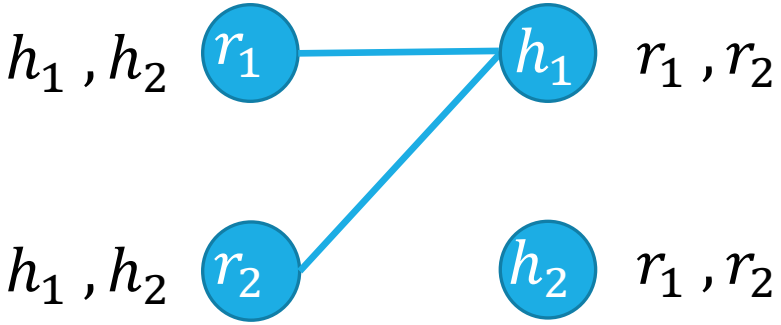
To make sure you've got the definition, and practice breakout rooms:

1. Download the activity pdf from the webpage (it's just the next slide in this slide deck)
2. Introduce yourselves; if possible have someone share that slide.
3. Try the problem.
4. Fill out the polleverywhere

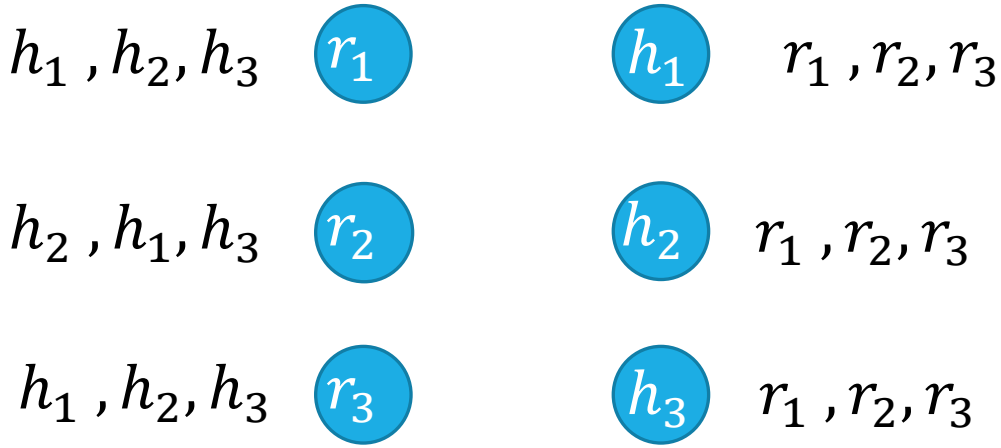
Try it!

Why are these not stable matchings?





Find a stable matching for this instance.



Questions

Does a stable matching always exist?

Can we find a stable matching efficiently?

We'll answer both of those questions in the next few lectures.

Let's start with the second one.

Idea for an Algorithm

Key idea

Unmatched riders “propose” to the highest horse on their preference list that they have not already proposed to.

Send in a rider to walk up to their favorite horse.

Everyone in front of a different horse? Done!

If more than one rider is at the same horse, let the horse decide its favorite.

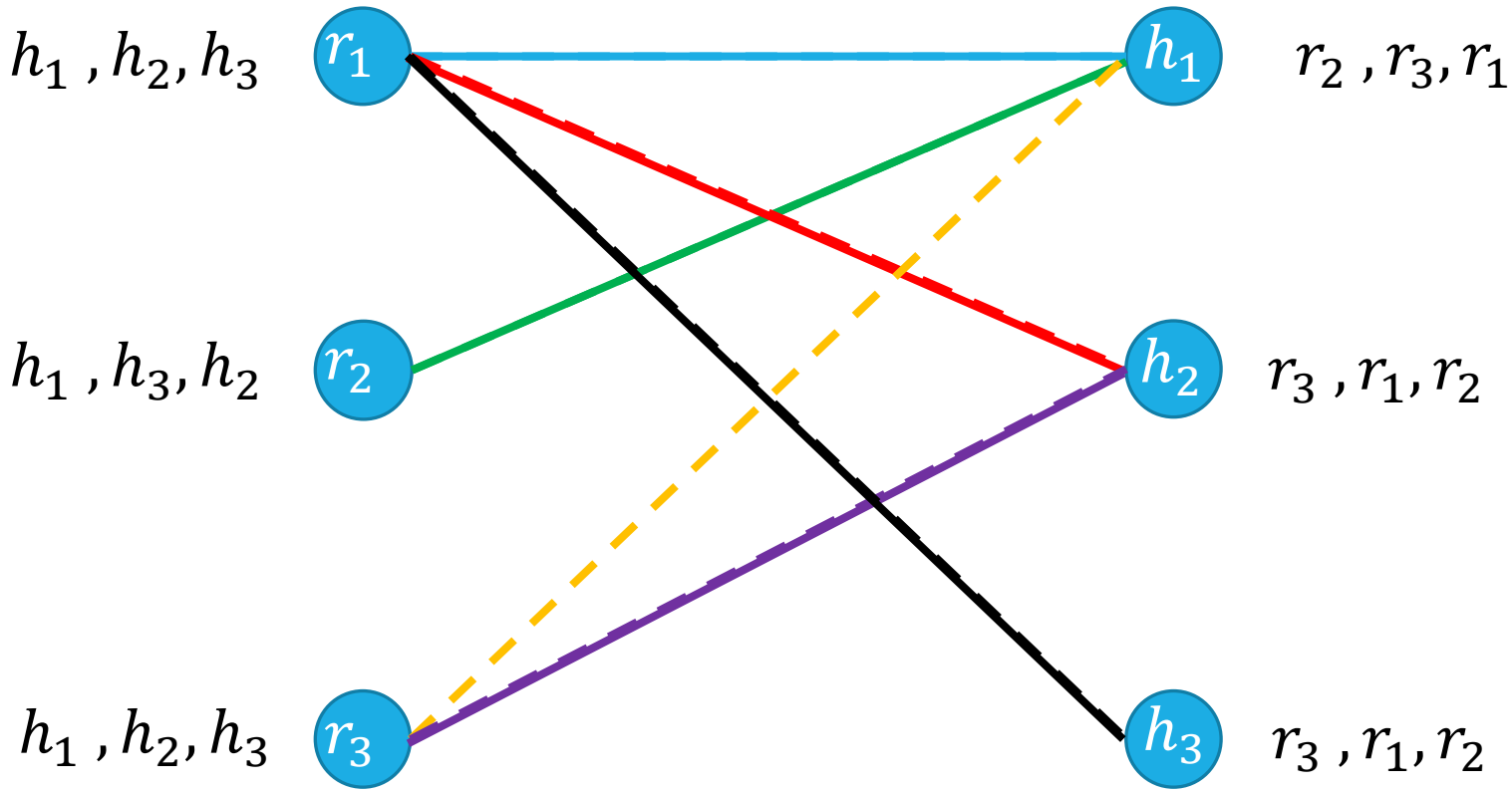
Rejected riders go back outside.

Repeat until you have a perfect matching.

Gale-Shapley Algorithm

Initially all r in R and h in H are free
While there is a free r
    Let h be highest on r's list that r has not proposed to
    if h is free, then match (r, h)
    else //h is not free
    suppose (r', h) are matched
        if h prefers r to r′
            unmatch (r', h)
            match (r, h)

Algorithm Example



Proposals: $r_1, r_2, r_1, r_3, r_3, r_1$

Does this algorithm work?

Does it run in a reasonable amount of time?

Is the result correct (i.e. a stable matching)?

Begin by identifying invariants and measures of progress

- Observation A: r's proposals get worse for them.
- Observation B: Once h is matched, h stays matched.
- Observation C: h's partners get better.

How do we justify these? A one-sentence explanation would suffice for each of these on the homework.

How did we know these were the right observations? Practice. And editing – we wouldn't have found these the first time, but after reading through early proof attempts.

TODO List

Make sure you're on Ed! (check your spam folder for an invite, if not there send an email to Robbie).

Fill out the course background survey.

Look at Homework 1 and start looking at 373 review materials.