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?

Grab a handout OR download a copy from the webpage if you want to save paper.

Welcome

CSE 417 Winter 24
Lecture 1

Today

Logistics

What is this course?

Start of the content

Staff

Instructor: Robbie Weber

Ph.D. from UW CSE in theory
Fourth year as teaching faculty

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

TAs

Ben Caffee
Rich Chen
Luksh Ganjoo
Yigao (Alex) Li
Alicia Stepin
Aria Tang

TODO List

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

Look at Homework 0 and 373 review materials.

Syllabus

It's all on the webpage:
https://courses.cs.washington.edu/courses/cse417/24wi/

In general, we're designing lecture to be synchronous (taking questions live, time for student discussions).

But we're trying to make sure if you need to stay home for a few days on short notice the effect will be minimal.

We won't have exams, and lectures are recorded, so there's no requirement to come in-person.

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

Your grade will be based (only) on homework assignments. We won't have any exams.

Logistics – Work

This quarter we're using "mastery grading" for assignments

Basic idea: Don't grade for "points"

Every assignment will have a few more problems on it than we expect you to do. You'll choose which to do.

Instead of assigning exact points, Every problem gets a score of:

Excellent	Main idea and edge cases are all correct. Would have gotten full credit (or extremely close) with points-based grading.
Satisfactory	Main idea is correct, but some edge cases or follow-up questions are wrong or missing. Would have gotten about 80-90% on points-based grading.
Not Yet	Some important error is made, but substantial progress toward a solution. Would have gotten above 50% on points-based grading.
Unassessable	Directions not followed (e.g., used a library that isn't permitted) or otherwise shows no substantial progress.

Logistics – Work

Minimums for grade breaks are already posted.

Why? We care if you understand the content by the end of the quarter, not right away. We'd like to give you more chances to show your understanding.

But we also need to make sure the TAs have enough time to grade everything.

Switching from points to E/S/N/U lets them grade faster, and therefore to regrade submissions. Every week you can submit up to two problems, and you'll get the better grade.

Logistics – Lecture Activities

I'm going to be teaching 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.

The answers live help me adjust explanations.

There aren't points associated with completing the activities.

Logistics – where to go?

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

Homework submission on gradescope

Live lecture activities on polleverywhere

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

The resubmit option is the late policy – it's intended to help with "normal" in-quarter difficulties (you might do one or two fewer problems the week you have a midterm in another course, for example).

If you have an unexpected situation that is going to interfere with your ability to work for an extended period of time (e.g., extended illness or family responsibilities) please send Robbie an email as soon as possible and we'll figure out what to do.

If something comes up…

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

Real life is going to get in the way for some of you in a class of this size. 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.

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?

A **mix** of theory and programming, but with a focus on theory.

Most of the problems in this course will be submitted with a mix of English and pseudocode.

But homeworks average between 1 and 2 programming questions per week available to you (out of 4 you're required to do).

This is a good course to get rid of cobwebs if it's been a while since you programmed, and you will be a better programmer when you leave it. But that's because you are gaining more **thinking** and **explaining** skills (much more than direct programming skills in this course).

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.

NOT: a list of all the most famous algorithms.

We'll cover many of the famous ones! But our focus will be more on helping you learn to solve new problems.

Course Topics (Tentative)

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

What's Coming Up

This week: An extremely useful algorithm.

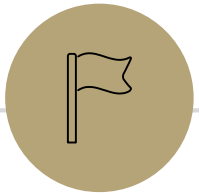
That has had lots of effect on the real world.

Along the way: what constitutes a convincing argument that my algorithm works?

We'll learn: direct arguments, proof by contradiction, and some fundamental logic.

Already know that? 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, and practice on the first assignment.



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.

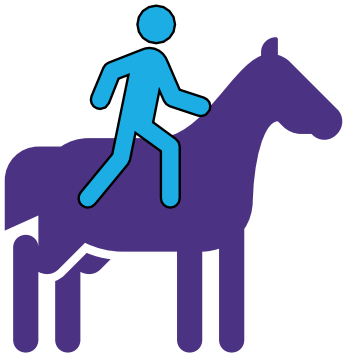
The core ideas have been adapted to all of these scenarios.

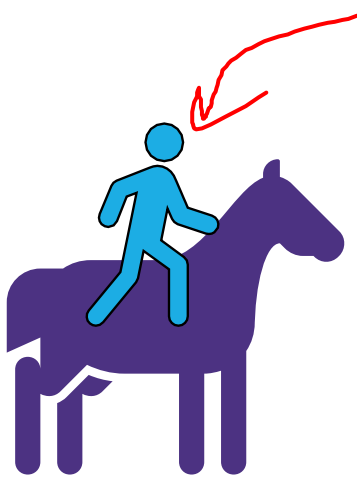
Stable Matching Problem

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

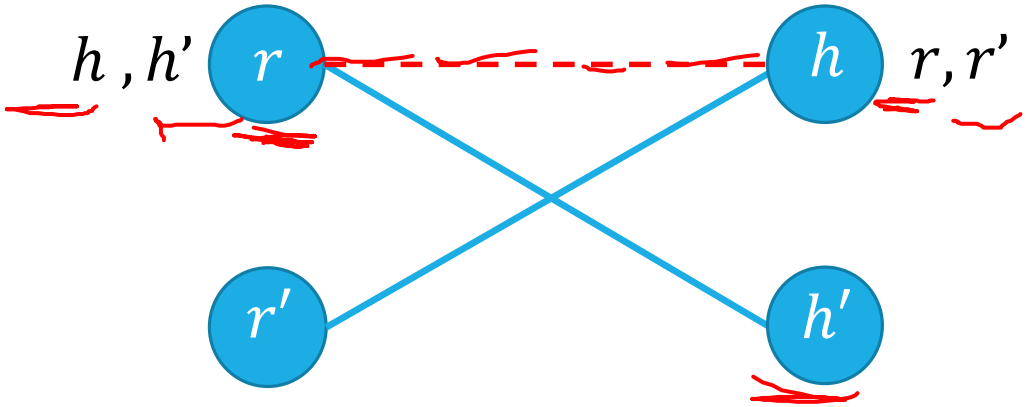
Given two sets $\mathrm{R} = \{r_1, \ldots, r_n\}, H = \{h_1, \ldots, h_n\}$

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.

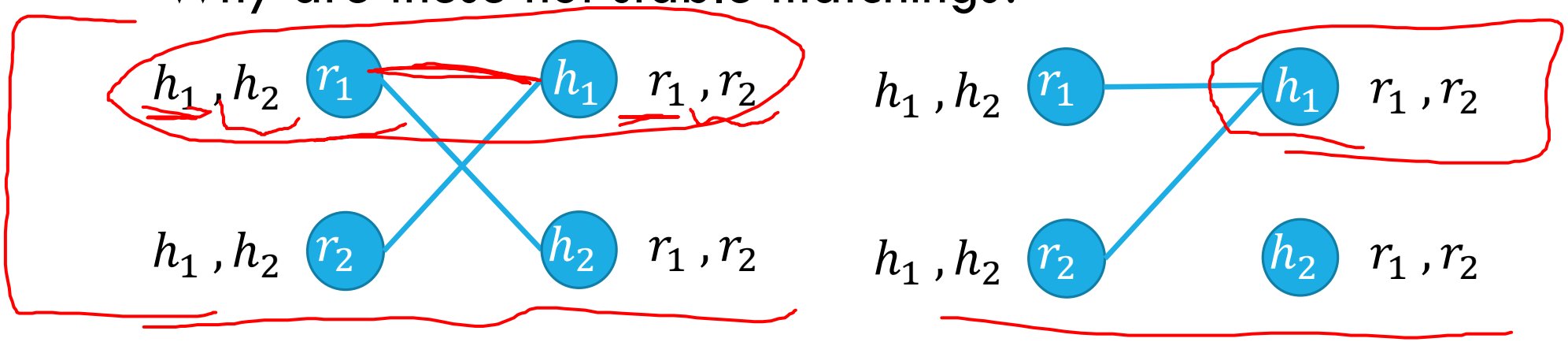
Lecture Activity

To make sure you've got the definition:

1. Download the activity pdf from the webpage (it's just the next slide in this slide deck) or look at the physical handout.
2. Introduce yourself to those around you.
3. Try the problem.
4. Fill out the polleverywhere

Try it!

Why are these not stable matchings?



Find a stable matching for this instance.

h_1, h_2, h_3	r_1	h_1	r_1, r_2, r_3
h_2, h_1, h_3	r_2	h_2	r_1, r_2, r_3
h_1, h_2, h_3	r_3	h_3	r_1, r_2, r_3

Pollev.com/robbie

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.

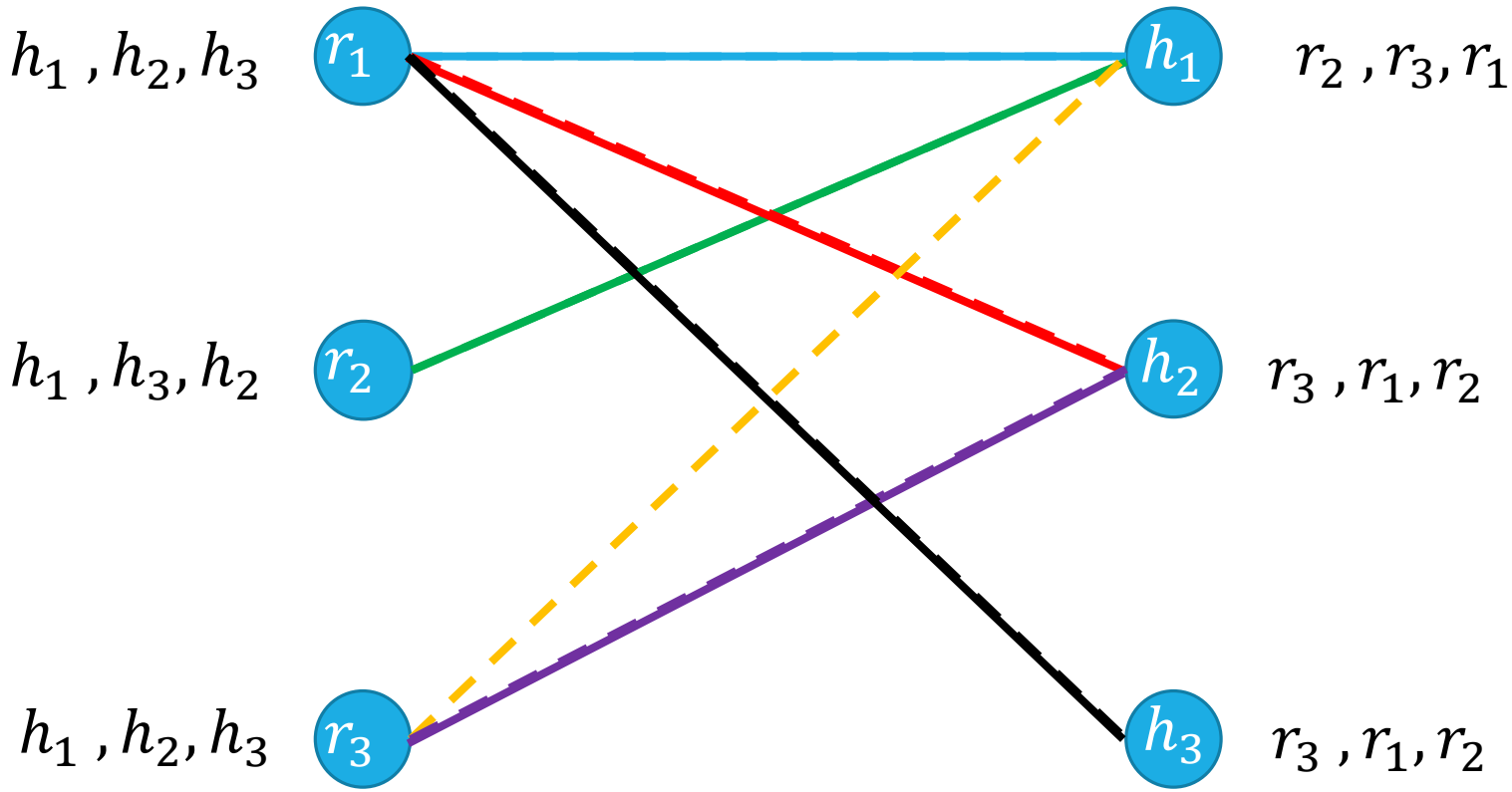
Rejected riders go back outside.

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).

Look at Homework 0 and 373 review materials.