Objectives

- Introduction to problem solving
  - Our process, through an example

Matching Residents to Hospitals

- **Goal**: Given a set of preferences among hospitals and medical school students, design a *self-reinforcing* admissions process.
- Applicant *x* and hospital *y* are ***unstable*** if
  - *x* prefers *y* to its assigned hospital
  - *y* prefers *x* to one of its admitted students
- **Stable assignment**: Assignment with no unstable pairs
  - Natural and desirable condition
  - Individual self-interest will prevent unstable applicant/hospital deal from being made

What details make this problem tricky?

Stable Matching Problem

Simplified version of resident-matching problem

- Goal: Given *n* men and *n* women, find a "suitable" matching
  - Participants rank members of opposite sex
  - Each man lists women in order of preference from best to worst
  - Each woman lists men in order of preference from best to worst

favorite → 1st; least favorite → 3rd

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Matching Goals

- **Perfect matching**: everyone is matched monogamously
  - Each man is paired with exactly one woman
  - Each woman is paired with exactly one man
- **Stability**: no incentive for some pair of participants to undermine assignment by joint action
  - An *unmatched* pair *m-w* is **unstable** if man *m* and woman *w* prefer each other to current partners
  - Unstable pair *m-w* could each improve by eloping
- **Stable matching**: perfect matching with no unstable pairs

Stable matching problem:
Given the preference lists of *n* men and *n* women, find a stable matching if one exists.

Analyzing Stability

- Is pairing X-C, Y-B, Z-A stable?
  - Instable: m prefers w to his woman; w prefers m to her man

favorite → 1st; least favorite → 3rd

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Matching Problem

- Is pairing X-C, Y-B, Z-A stable?
- No. Bertha and Xavier prefer each other

favorite → 1st; least favorite → 3rd

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Stable Matching Problem

- Is pairing X-A, Y-B, Z-C stable?
- Yes.

favorite → 1st; least favorite → 3rd

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

Any Questions?

- What are you wondering about this problem/ its solution at this point?

Any Questions?

- What are you wondering about this problem/ its solution at this point?
- Hopefully:
  - Is there a stable matching for every pair of preference lists?
  - If so, is there an algorithm to find the stable matching?
  - Can we be fair in the matching? (preferences)
  - Will the matching always be the same?

Propose-And-Reject Algorithm

[Gale-Shapley 1962]

- Intuitive method that guarantees finding a stable matching

```
Initialize each person to be free
while (some man is free and hasn't proposed to every woman)
    Choose such a man m
    w = 1st woman on m's list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged and m' to be free
    else
        w rejects m
```


Applying the Algorithm

favorite → 1st; least favorite → 3rd

	1st	2nd	3rd
Xavier	Amy	Bertha	Clare
Yancey	Bertha	Amy	Clare
Zeus	Amy	Bertha	Clare

Men's Preference Profile

	1st	2nd	3rd
Amy	Yancey	Xavier	Zeus
Bertha	Xavier	Yancey	Zeus
Clare	Xavier	Yancey	Zeus

Women's Preference Profile

```
Initialize each person to be free
while (some man is free and hasn't proposed to every woman)
    Choose such a man m
    w = 1st woman on m's list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged and m' to be free
    else
        w rejects m
```

Observations about the Algorithm

- What can we say about any woman's partner during the execution of the algorithm?
- How does a woman's state change over the execution of the algorithm?
- What can we say about a man's partner?

Observations about the Algorithm

- What can we say about any woman's partner during the execution of the algorithm?
  - Observation 1. He gets "better" → she prefers him over her last partner
- How does a woman's state change over the execution of the algorithm?
  - Observation 2. Once a woman is matched, she never becomes unmatched; she only "trades up"
- What can we say about a man's partner?
  - Observation 3. She gets "worse"

Jan 13, 2010 CS211 13

Proof of Correctness: Termination

- Claim. Algorithm terminates after at most n^2 iterations of while loop.
  - Hint: How wouldn't the algorithm terminate?

Jan 13, 2010 CS211 14

Proof of Correctness: Termination

- Claim. Algorithm terminates after at most n^2 iterations of while loop.
- Pf. Each time through the while loop, a man proposes to a new woman. There are only n^2 possible proposals.

$n(n-1) + 1$ proposals required

Jan 13, 2010 CS211 15

Algorithm Analysis

Prove that final matching is a *perfect* matching

- Hint: in algorithm, we know if *m* is free at some point in the execution of the algorithm, then there is a woman to whom he has not yet proposed.

Jan 13, 2010 CS211 16

Proof of Correctness: Perfection

- Claim. All men and women get matched.
- Pf. (by contradiction)
  - Where should we start?

Jan 13, 2010 CS211 17

Proof of Correctness: Perfection

- Claim. All men and women get matched.
- Pf. (by contradiction)
  - Suppose that *m* is not matched upon termination of algorithm
  - Then some woman, say *w*, is not matched upon termination.
  - By Observation 2, *w* was never proposed to.
  - But, last man proposes to everyone, since he ends up unmatched
    - (by the while loop's condition)
  - Contradiction ▪

Jan 13, 2010 CS211 18

Proof of Correctness: Stability

- Claim. No unstable pairs.

How will we prove this?

S*

Amy-Yancey
Bertha-Zeus
...

Proof of Correctness: Stability

S*

Amy-Yancey
Bertha-Zeus
...

- Claim. No unstable pairs.
- Pf. (by contradiction)
  - Suppose A-Z is an unstable pair: each prefers each other to partner in Gale-Shapley matching S*.

How could that happen?
What are the possibilities that lead to this?

Proof of Correctness: Stability

S*

Amy-Yancey
Bertha-Zeus
...

- Claim. No unstable pairs.
- Pf. (by contradiction)
  - Suppose A-Z is an unstable pair: each prefers each other to partner in Gale-Shapley matching S*.
  - Case 1: Z never proposed to A.
    - ⇒ Z prefers his GS partner to A. ← men propose in decreasing order of preference
    - ⇒ A-Z is stable.
  - Case 2: Z proposed to A.
    - ⇒ A rejected Z (right away or later) ← women only trade up
    - ⇒ A prefers her GS partner to Z.
    - ⇒ A-Z is stable.
  - In either case A-Z is stable, a contradiction. ▪

Summary So Far...

- **Stable matching problem.** Given *n* men and *n* women and their preferences, find a stable matching if one exists.
- **Gale-Shapley algorithm.** Guarantees to find a stable matching for *any* input

Remaining Questions:
- How to implement GS algorithm efficiently?
- If there are multiple stable matchings, which one does GS find? *(see book)*

How Can We Implement The Algorithm Efficiently?

- What is our goal for the implementation's runtime?

Efficient Implementation

- Representing men and women
  - Assume men are named 1, …, n
  - Assume women are named 1', …, n'
- Preferences
  - Two nxn arrays
- Engagements
  - Maintain a list of free men, e.g., in a queue.
  - Maintain two arrays `wife[m]`, and `husband[w]`.
    - Set entry to `0` if unmatched
    - If m matched to w then `wife[m]=w` and `husband[w]=m`
- Men proposing
  - For each man, maintain a list of women, ordered by preference.
  - Maintain an array `count[m]` that counts the number of proposals made by man `m`.

Efficient Implementation

- Women rejecting/accepting: determine does woman `w` prefer man `m` to man `m'`?
  - For each woman, create *inverse* of preference list of men
  - Constant time access for each query after O(n) preprocessing

Amy	1st	2nd	3rd	4th	5th	6th	7th	8th
Pref	8	3	7	1	4	5	6	2

Amy	1	2	3	4	5	6	7	8
Inverse	4th	8th	2nd	5th	6th	7th	3rd	1st

Amy prefers man 3 to 6
since inverse[3] < inverse[6]

2 7

```
for i = 1 to n
   inverse[ pref[i] ] = i
```


Review: Our Process

1. Understand/identify problem
   - Simplify as appropriate
2. Design a solution
3. Analyze
   - Correctness, efficiency
   - May need to go back to step 2 and try again
4. Implement
   - Within bounds shown in analysis

Stable Matching Summary

- **Stable matching problem.** Given preference profiles of n men and n women, find a *stable* matching.

no man and woman prefer to be with each other than assigned partner

- **Gale-Shapley algorithm.** Finds a stable matching in $O(n^2)$ time.
  - Can implement algorithm *efficiently*

Assignments

- Review Chapter 1
- Read Chapter 2