# Lecture 31: Stable ~~Marriage~~ Internship Algorithm

Heteronormativity Is Dumb.

# Announcements

End of course survey should now have all staff members!
Due by 11:59 PM tomorrow — pls fill it out!

# Problem Statement

4 students applying for internships
4 companies want 1 intern each

# Problem Statement

4 students applying for internships
4 companies want 1 intern each

Everyone has a preference:

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 1 > 2$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Who should work where?

# Bad Matching

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 1 > 2$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Should B work at 1?

# Bad Matching

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 1 > 2$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Should B work at 1?

B wants to work at 3, 3 wants B
Incentive for both to leave system

# Bad Matching

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 1 > 2$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Should B work at 1?

B wants to work at 3, 3 wants B
Incentive for both to leave system

Want to avoid this kind of problem

# Stability

**Rogue pair** is company + student that prefer each other over assigned counterpart

Matching **stable** if no rogue pairs

# Stability

**Rogue pair** is company + student that prefer each other over assigned counterpart

Matching **stable** if no rogue pairs

Goal: Given preference lists, find stable pairing

# Is It Stable?

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Is $(A, 4), (B, 3), (C, 1), (D, 2)$ stable?

# Is It Stable?

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Is $(A, 4), (B, 3), (C, 1), (D, 2)$ stable?
No — $(4, C)$ is rogue!

# Is It Stable?

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Is $(A, 4), (B, 3), (C, 1), (D, 2)$ stable?
No — $(4, C)$ is rogue!

What about $(A, 1), (B, 3), (C, 4), (D, 2)$?

# Is It Stable?

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Is $(A, 4), (B, 3), (C, 1), (D, 2)$ stable?
No — $(4, C)$ is rogue!

What about $(A, 1), (B, 3), (C, 4), (D, 2)$?
Yep!

# Is Stability Guaranteed?

Natural Q: is there always a stable matching?

# Is Stability Guaranteed?

Natural Q: is there always a stable matching?
Not immediately obvious!

# Is Stability Guaranteed?

Natural Q: is there always a stable matching?
Not immediately obvious!

Consider "Stable Roomates":

| Person | Preferences |
|---|---|
| $A$ | $B > C > D$ |
| $B$ | $C > A > D$ |
| $C$ | $A > B > D$ |
| $D$ | $A > B > C$ |

Possible pairings:

- $(A, B), (C, D)$
- $(A, C), (B, D)$
- $(A, D), (B, C)$

# Gale-Shapley Algorithm

Turns out, internships always has stable matching!
Prove by giving algorithm to find one

# Gale-Shapley Algorithm

Turns out, internships always has stable matching!
Prove by giving algorithm to find one

**Morning**: Students apply to top company on list

# Gale-Shapley Algorithm

Turns out, internships always has stable matching!
Prove by giving algorithm to find one

**Morning**: Students apply to top company on list
**Afternoon**: Companies reject all but top applicant

# Gale-Shapley Algorithm

Turns out, internships always has stable matching!
Prove by giving algorithm to find one

**Morning**: Students apply to top company on list
**Afternoon**: Companies reject all but top applicant
**Evening**: Rejected students cross off company

# Gale-Shapley Algorithm

Turns out, internships always has stable matching!
Prove by giving algorithm to find one

**Morning**: Students apply to top company on list
**Afternoon**: Companies reject all but top applicant
**Evening**: Rejected students cross off company

Algorithm stops once no rejections.

# Gale-Shapley Algorithm

Turns out, internships always has stable matching!
Prove by giving algorithm to find one

**Morning**: Students apply to top company on list
**Afternoon**: Companies reject all but top applicant
**Evening**: Rejected students cross off company

Algorithm stops once no rejections.

**Claim**: Algorithm always terminates
No more than $n^2$ rejections possible!

# Example Run Day 1

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

# Example Run Day 2

| Stdnt | Preferences |
|---|---|
| $A$ | ~~4~~ $> 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | ~~3~~ $> 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

# Example Run Day 3

| Stdnt | Preferences |
|---|---|
| $A$ | ~~4~~ $>$ ~~3~~ $> 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | ~~3~~ $> 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

# Example Run Day 3

| Stdnt | Preferences |
|---|---|
| $A$ | ~~4~~ $>$ ~~3~~ $>$ ~~2~~ $>$ 1 |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | ~~3~~ $> 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

# Example Run Day 3

| Stdnt | Preferences |
|---|---|
| $A$ | ~~4~~ $>$ ~~3~~ $>$ ~~2~~ $> 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | ~~3~~ $> 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

Algorithm terminates with matching
$(A, 1), (B, 3), (C, 4), (D, 2)$

Stable here — how do we know it always is?

# Improvement Lemma

Say company "interviewing" student if student applies and not yet rejected

**Lemma**: If S applies to C on day $k$, C interviewing S or better on every subsequent day

# Improvement Lemma

Say company "interviewing" student if student applies and not yet rejected

**Lemma**: If S applies to C on day $k$, C interviewing S or better on every subsequent day

**Proof**:

- Base Case: S applies on day $k$, so best applicant S or better

# Improvement Lemma

Say company "interviewing" student if student applies and not yet rejected

**Lemma**: If S applies to C on day $k$, C interviewing S or better on every subsequent day

**Proof**:

- Base Case: S applies on day $k$, so best applicant S or better
- Suppose interviews S' $\geq$ S on day $j \geq k$

# Improvement Lemma

Say company "interviewing" student if student applies and not yet rejected

**Lemma**: If S applies to C on day $k$, C interviewing S or better on every subsequent day

**Proof**:

- Base Case: S applies on day $k$, so best applicant S or better
- Suppose interviews S' $\geq$ S on day $j \geq k$
- S' applies on day $j+1$, so best $\geq$ S' $\geq$ S

# Lemma Not Cool Enough To Have Name

**Lemma**: Applications on last day form a pairing

# Lemma Not Cool Enough To Have Name

**Lemma**: Applications on last day form a pairing

**Proof**:

- No rejections, so $\leq 1$ applicant per job
- Only poss issue if student rejected everywhere!

# Lemma Not Cool Enough To Have Name

**Lemma**: Applications on last day form a pairing

**Proof**:

- No rejections, so $\leq 1$ applicant per job
- Only poss issue if student rejected everywhere!
- That student applied everywhere
- Improvement Lemma: comps have better stdnt

# Lemma Not Cool Enough To Have Name

**Lemma**: Applications on last day form a pairing

**Proof**:

- No rejections, so $\leq 1$ applicant per job
- Only poss issue if student rejected everywhere!
- That student applied everywhere
- Improvement Lemma: comps have better stdnt
- Would need more students than companies!

# Lemma Not Cool Enough To Have Name

**Lemma**: Applications on last day form a pairing

**Proof**:

- No rejections, so $\leq 1$ applicant per job
- Only poss issue if student rejected everywhere!
- That student applied everywhere
- Improvement Lemma: comps have better stdnt
- Would need more students than companies!

Now just have to prove no rogue couples!

# Wrapping It Up

**Theorem**: Matching at end of algo is stable

# Wrapping It Up

**Theorem**: Matching at end of algo is stable

**Proof**:

- Suppose have (S, C) matched, (S, C*) rogue

# Wrapping It Up

**Theorem**: Matching at end of algo is stable

**Proof**:

- Suppose have (S, C) matched, (S, C*) rogue
- Def of rogue: S likes C* $>$ C
- So in algorithm S applied to C*

# Wrapping It Up

**Theorem**: Matching at end of algo is stable

**Proof**:

- Suppose have (S, C) matched, (S, C*) rogue
- Def of rogue: S likes C* $>$ C
- So in algorithm S applied to C*
- Improvement Lemma: C* has better than S
- So C* wouldn't go rogue — contradiction!

# Our Final Break :'(

Time to take a break!

Two options:

- Normal discussion question
- I can show you a magic trick

# Our Final Break :'(

Time to take a break!

Two options:

- Normal discussion question
- I can show you a magic trick

**Today's Discussion Question**:
Is a hot dog a taco?

# Magic Trick

# Optimality

Is the stable pairing we get good? What is “good”?

# Optimality

Is the stable pairing we get good? What is "good"?

**Def**: Optimal company for S is best they can get *in any stable pairing*

# Optimality

Is the stable pairing we get good? What is "good"?

**Def**: Optimal company for S is best they can get *in any stable pairing*

Not necessarily top of their list!

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

# Optimal Pairing

**Theorem**: Pairing from algorithm gives all students their optimal company

# Optimal Pairing

**Theorem**: Pairing from algorithm gives all students their optimal company

**Proof**:
Show: no student rejected by opt company on day $k$

- Base Case: Day 1
- Suppose S rejected by C* in favor of S'

# Optimal Pairing

**Theorem**: Pairing from algorithm gives all students their optimal company

**Proof**:
Show: no student rejected by opt company on day $k$

- Base Case: Day 1
- Suppose S rejected by C* in favor of S'
- C* opt for S, so have stable pairing w/(S, C*)
- S' has company C' in that pairing

# Optimal Pairing

**Theorem**: Pairing from algorithm gives all students their optimal company

**Proof**:
Show: no student rejected by opt company on day $k$

- Base Case: Day 1
- Suppose S rejected by C* in favor of S'
- C* opt for S, so have stable pairing w/(S, C*)
- S' has company C' in that pairing
- S' applies to C* on first day, so C* $\geq$ C'
- C* rejects S, so S' $\geq$ S

# Optimal Pairing

**Theorem**: Pairing from algorithm gives all students their optimal company

**Proof**:
Show: no student rejected by opt company on day $k$

- Base Case: Day 1
- Suppose S rejected by C* in favor of S'
- C* opt for S, so have stable pairing w/(S, C*)
- S' has company C' in that pairing
- S' applies to C* on first day, so C* $\geq$ C'
- C* rejects S, so S' $\geq$ S
- (S', C*) rogue — contradiction!

# Optimality Inductive Step

Just need (strong) inductive step:
If no student rejected by opt company day $k$ or earlier, none on day $k+1$

# Optimality Inductive Step

Just need (strong) inductive step:
If no student rejected by opt company day $k$ or earlier, none on day $k+1$

**Proof**:

- Suppose S rejected by C* in favor of S'

# Optimality Inductive Step

Just need (strong) inductive step:
If no student rejected by opt company day $k$ or earlier, none on day $k+1$

**Proof**:

- Suppose S rejected by C* in favor of S'
- C* opt for S, so have stable pairing w/(S, C*)
- S' has company C' in pairing; opt company C'*

# Optimality Inductive Step

Just need (strong) inductive step:
If no student rejected by opt company day $k$ or earlier, none on day $k+1$

**Proof**:

- Suppose S rejected by C* in favor of S'
- C* opt for S, so have stable pairing w/(S, C*)
- S' has company C' in pairing; opt company C'*
- Ind Hypothesis: S' not rejected by C'*
- So for S', C* $\geq$ C'* $\geq$ C'
- C* rejects S, so S' $\geq$ S

# Optimality Inductive Step

Just need (strong) inductive step:
If no student rejected by opt company day $k$ or earlier, none on day $k+1$

**Proof**:

- Suppose S rejected by C* in favor of S'
- C* opt for S, so have stable pairing w/(S, C*)
- S' has company C' in pairing; opt company C'*
- Ind Hypothesis: S' not rejected by C'*
- So for S', C* $\geq$ C'* $\geq$ C'
- C* rejects S, so S' $\geq$ S
- (S', C*) rogue — contradiction!

# Pessimality

What is opposite of optimal?

# Pessimality

What is opposite of optimal?

**Def**: Pessimal student for C is worst they get *in any stable pairing*

# Pessimality

What is opposite of optimal?

**Def**: Pessimal student for C is worst they get *in any stable pairing*

Not necessarily bottom of their list!

| Stdnt | Preferences |
|---|---|
| $A$ | $4 > 3 > 2 > 1$ |
| $B$ | $3 > 4 > 1 > 2$ |
| $C$ | $4 > 1 > 3 > 2$ |
| $D$ | $3 > 2 > 1 > 4$ |

| Comp | Preferences |
|---|---|
| 1 | $B > D > C > A$ |
| 2 | $B > D > A > C$ |
| 3 | $B > A > D > C$ |
| 4 | $B > C > A > D$ |

# Perfectly Balanced, As All Things Should Be

**Thm**: Algorithm output pessimal for companies

# Perfectly Balanced, As All Things Should Be

**Thm**: Algorithm output pessimal for companies

**Proof**:

- Let output pair S with C
- Suppose $\exists$ stable pairing with (S', C), S' $\leq$ S
- Let C' be S' company in that pairing

# Perfectly Balanced, As All Things Should Be

**Thm**: Algorithm output pessimal for companies

**Proof**:

- Let output pair S with C
- Suppose $\exists$ stable pairing with (S', C), S' $\leq$ S
- Let C' be S' company in that pairing
- C optimal for S, so C' $\leq$ C

# Perfectly Balanced, As All Things Should Be

**Thm**: Algorithm output pessimal for companies

**Proof**:

- Let output pair S with C
- Suppose $\exists$ stable pairing with (S', C), S' $\leq$ S
- Let C' be S' company in that pairing
- C optimal for S, so C' $\leq$ C
- Then (C, S) is rogue — contradiction!

# Fin

Good luck on the final!