Telling the Story of the Stable (Queer) Marriage Problem

Edison Hauptman[1]

[1] Mathematics, The University of Pittsburgh, ERH129@pitt.edu

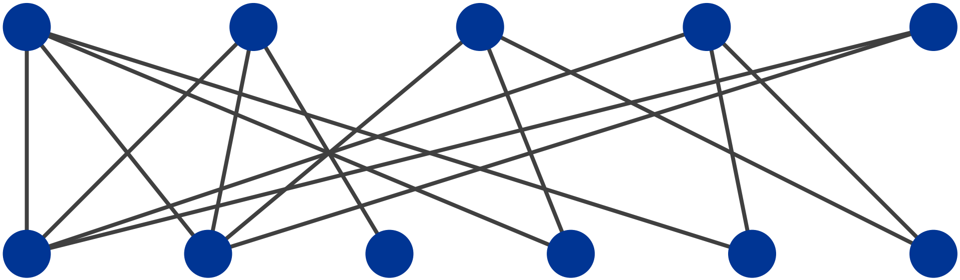
January 5, 2026

Outline

Matchings

Definition

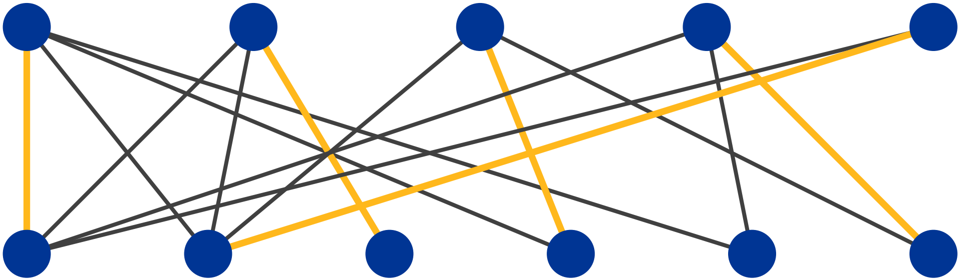
A **matching** in a graph $G = (V, E)$ is a subset of edges $E' \subset E$ such that each vertex belongs to at most one edge.



Matchings

Definition

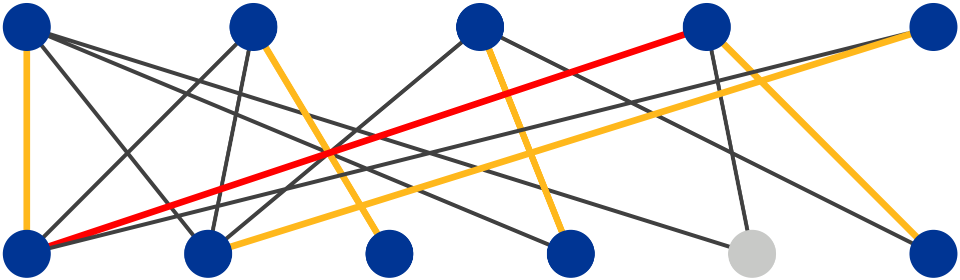
A **matching** in a graph $G = (V, E)$ is a subset of edges $E' \subset E$ such that each vertex belongs to at most one edge.



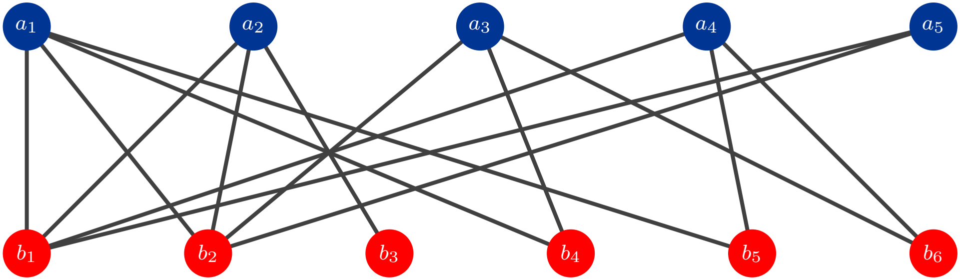
Matchings

Definition

A **matching** in a graph $G = (V, E)$ is a subset of edges $E' \subset E$ such that each vertex belongs to at most one edge.



Bipartite Graphs

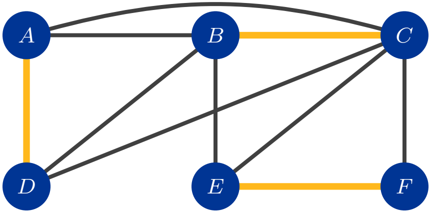


Definition

G is **bipartite** if there are sets A, B that partition V $[V = A \cup B,\ A \cap B = \emptyset]$ such that every edge can be written $e = ab \in E$, with $a \in A$ and $b \in B$.

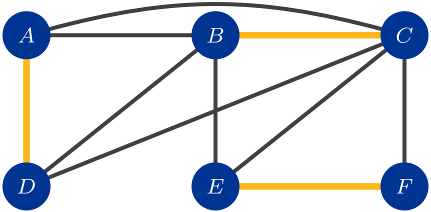
What Makes a Matching Stable?

Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



What Makes a Matching Stable?

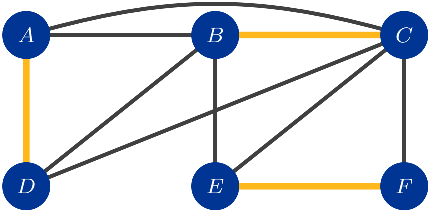
Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

What Makes a Matching Stable?

Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



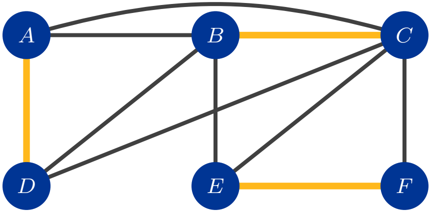
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

Definition

A matching is **unstable** if $\exists i, i', j, j' \in V$ such that (i, i') and (j, j') are matched but $P_i(j) > P_i(i')$ and $P_j(i) > P_j(j')$.

What Makes a Matching Stable?

Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

Definition

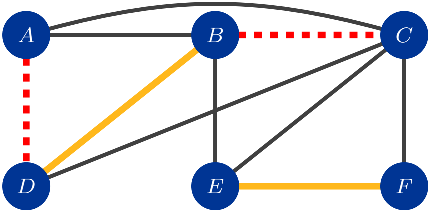
A matching is **unstable** if $\exists i, i', j, j' \in V$ such that (i, i') and (j, j') are matched but $P_i(j) > P_i(i')$ and $P_j(i) > P_j(j')$.

Definition

A matching is **stable** if no such unstable pair exists.

What Makes a Matching Stable?

Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



Node	Preference Order				
A	B	F	C	E	$\cancel{D}$
B	F	A	E	D	$\cancel{C}$
C	F	E	A	$\cancel{B}$	D
D	B	$\cancel{A}$	C		
E	F	D	C	B	
F	E	C			

Definition

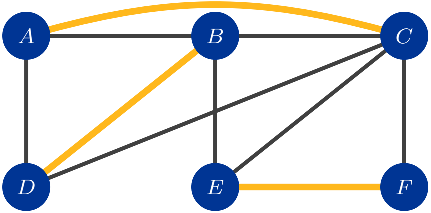
A matching is **unstable** if $\exists i, i', j, j' \in V$ such that (i, i') and (j, j') are matched but $P_i(j) > P_i(i')$ and $P_j(i) > P_j(j')$.

Definition

A matching is **stable** if no such unstable pair exists.

What Makes a Matching Stable?

Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

Definition

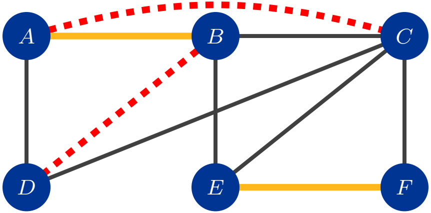
A matching is **unstable** if $\exists i, i', j, j' \in V$ such that (i, i') and (j, j') are matched but $P_i(j) > P_i(i')$ and $P_j(i) > P_j(j')$.

Definition

A matching is **stable** if no such unstable pair exists.

What Makes a Matching Stable?

Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



Node	Preference Order				
A	B	F	$\cancel{C}$	E	D
B	F	A	E	$\cancel{D}$	C
C	F	E	$\cancel{A}$	B	D
D	$\cancel{B}$	A	C		
E	F	D	C	B	
F	E	C			

Definition

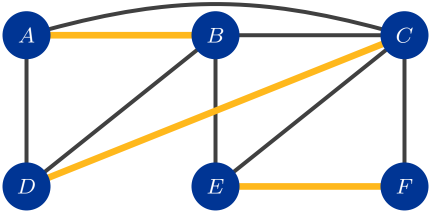
A matching is **unstable** if $\exists i, i', j, j' \in V$ such that (i, i') and (j, j') are matched but $P_i(j) > P_i(i')$ and $P_j(i) > P_j(j')$.

Definition

A matching is **stable** if no such unstable pair exists.

What Makes a Matching Stable?

Consider, for each $v \in V$, a preference order $(P_v(x), >)$. Note $x \in V$.



Definition

A matching is **unstable** if $\exists i, i', j, j' \in V$ such that (i, i') and (j, j') are matched but $P_i(j) > P_i(i')$ and $P_j(i) > P_j(j')$.

Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

Definition

A matching is **stable** if no such unstable pair exists.

Example: The National Resident Matching Program

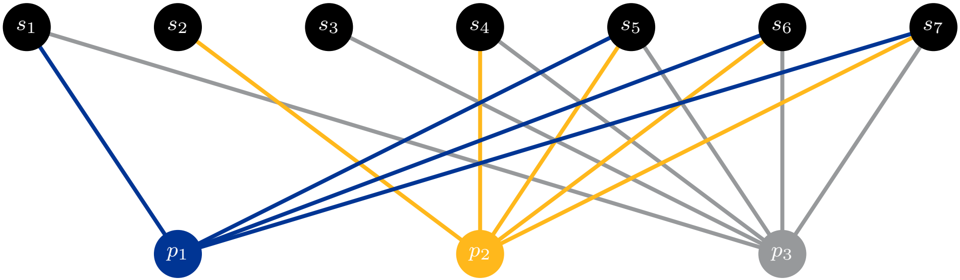
- Assigns medical students to residency programs.

Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- This can be represented with a bipartite graph

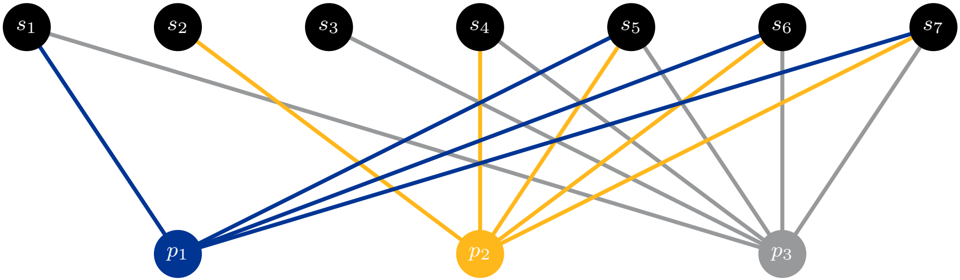
Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- This can be represented with a bipartite graph



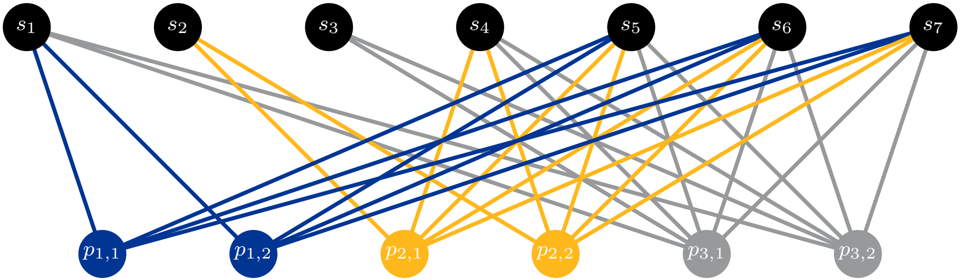
Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- This can be represented with a bipartite graph
- The programs can accept more than one student.



Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- This can be represented with a bipartite graph
- The programs can accept more than one student.



Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- This can be represented with a bipartite graph **and a preference order.**
- The programs can accept more than one student.

Student	Preference Order		
s_1	p_1	p_3	
s_2	p_3	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	p_2	p_3	p_1
s_7	p_1	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	s_7			
p_2	s_7	s_1	s_5	s_2	s_6	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	s_6	

Solution: The Gale-Shapley Algorithm (1962)

1. The students all submit a request to their top program that hasn't rejected them yet (they've already sent applications).

Student	Preference Order		
s_1	p_1	p_3	
s_2	p_3	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	p_2	p_3	p_1
s_7	p_1	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	s_7			
p_2	s_7	s_1	s_5	s_2	s_6	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	s_6	

Solution: The Gale-Shapley Algorithm (1962)

1. The students all submit a request to their top program that hasn't rejected them yet (they've already sent applications).
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	p_1	p_3	
s_2	p_3	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	p_2	p_3	p_1
s_7	p_1	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	s_7			
p_2	s_7	s_1	s_5	s_2	s_6	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	s_6	

Solution: The Gale-Shapley Algorithm (1962)

1. **The students all submit a request to their top program that hasn't rejected them yet.**
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	$\boxed{p_1}$	p_3	
s_2	$\boxed{p_3}$	p_2	p_1
s_3	$\boxed{p_3}$	p_1	
s_4	$\boxed{p_3}$	p_1	p_2
s_5	$\boxed{p_1}$	p_3	p_2
s_6	$\boxed{p_2}$	p_3	p_1
s_7	$\boxed{p_1}$	p_2	p_3

Program	Preference Order						
p_1	$\boxed{s_5}$	s_6	$\boxed{s_1}$	$\boxed{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\boxed{s_6}$	s_3	s_4
p_3	s_1	s_5	$\boxed{s_3}$	s_7	$\boxed{s_4}$	s_6	

The Stable Queer Marriage Problem

Edison Hauptman

Solution: The Gale-Shapley Algorithm (1962)

1. The students all submit a request to their top program that hasn't rejected them yet.
2. **Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.**

Student	Preference Order		
s_1	p_1	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	p_2	p_3	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	s_6	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	s_6	

Solution: The Gale-Shapley Algorithm (1962)

1. **Any students not currently in a program submit a request to their top program that hasn't rejected them yet.**
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	p_1	p_3	
s_2	$\cancel{p_3}$	$\boxed{p_2}$	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	p_2	p_3	p_1
s_7	$\cancel{p_1}$	$\boxed{p_2}$	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	$\cancel{s_7}$			
p_2	$\boxed{s_7}$	s_1	s_5	$\boxed{s_2}$	s_6	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	s_6	

Solution: The Gale-Shapley Algorithm (1962)

1. Any students not currently in a program submit a request to their top program that hasn't rejected them yet.
2. **Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.**

Student	Preference Order		
s_1	p_1	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	p_3	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	s_6	

Solution: The Gale-Shapley Algorithm (1962)

1. **Any students not currently in a program submit a request to their top program that hasn't rejected them yet.**
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	p_1	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\boxed{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	$\boxed{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. Any students not currently in a program submit a request to their top program that hasn't rejected them yet.
2. **Each program considers all of its requests to this point,** ***temporarily*** **accepts the top ones it has space for, and rejects the rest.**

Student	Preference Order		
s_1	p_1	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	s_1	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. **Any students not currently in a program submit a request to their top program that hasn't rejected them yet.**
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	p_1	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	$\boxed{p_1}$
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	$\boxed{s_6}$	s_1	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. Any students not currently in a program submit a request to their top program that hasn't rejected them yet.
2. **Each program considers all of its requests to this point,** ***temporarily*** **accepts the top ones it has space for, and rejects the rest.**

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	$\cancel{s_1}$	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	s_4	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. **Any students not currently in a program submit a request to their top program that hasn't rejected them yet.**
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	$\cancel{p_1}$	$\boxed{p_3}$	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	p_3	p_1	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	$\cancel{s_1}$	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	$\boxed{s_1}$	s_5	s_3	s_7	s_4	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. Any students not currently in a program submit a request to their top program that hasn't rejected them yet.
2. **Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.**

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	p_1	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	$\cancel{s_1}$	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	$\cancel{s_4}$	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. **Any students not currently in a program submit a request to their top program that hasn't rejected them yet.**
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	$\boxed{p_1}$	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	$\cancel{s_1}$	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	$\cancel{s_4}$	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. Any students not currently in a program submit a request to their top program that hasn't rejected them yet.
2. **Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.**

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	$\cancel{p_1}$	p_2
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	$\cancel{s_1}$	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	s_4
p_3	s_1	s_5	s_3	s_7	$\cancel{s_4}$	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. **Any students not currently in a program submit a request to their top program that hasn't rejected them yet.**
2. Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	$\cancel{p_1}$	$\boxed{p_2}$
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	$\cancel{s_1}$	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	$\boxed{s_4}$
p_3	s_1	s_5	s_3	s_7	$\cancel{s_4}$	$\cancel{s_6}$	

Solution: The Gale-Shapley Algorithm (1962)

1. Any students not currently in a program submit a request to their top program that hasn't rejected them yet.
2. **Each program considers all of its requests to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.**

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	$\cancel{p_1}$	$\cancel{p_2}$
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Program	Preference Order						
p_1	s_5	s_6	$\cancel{s_1}$	$\cancel{s_7}$			
p_2	s_7	s_1	s_5	s_2	$\cancel{s_6}$	s_3	$\cancel{s_4}$
p_3	s_1	s_5	s_3	s_7	$\cancel{s_4}$	$\cancel{s_6}$	

How well does this work?

- This algorithm always terminates.

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	$\cancel{p_1}$	$\cancel{p_2}$
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

How well does this work?

- This algorithm always terminates.
- The output will be a stable matching.

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	$\cancel{p_1}$	$\cancel{p_2}$
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

How well does this work?

- This algorithm always terminates.
- The output will be a stable matching.
- Since we focused on the students' requests, they got the best stable matching possible!

Student	Preference Order		
s_1	$\cancel{p_1}$	p_3	
s_2	$\cancel{p_3}$	p_2	p_1
s_3	p_3	p_1	
s_4	$\cancel{p_3}$	$\cancel{p_1}$	$\cancel{p_2}$
s_5	p_1	p_3	p_2
s_6	$\cancel{p_2}$	$\cancel{p_3}$	p_1
s_7	$\cancel{p_1}$	p_2	p_3

Can We Apply This Elsewhere?

- In some countries, students are matched to high schools using the Gale-Shapley algorithm.

Can We Apply This Elsewhere?

- In some countries, students are matched to high schools using the Gale-Shapley algorithm.
- Directly applying the Gale-Shapley algorithm to course registration can cause some snags (Tilley/Reid, 2020)

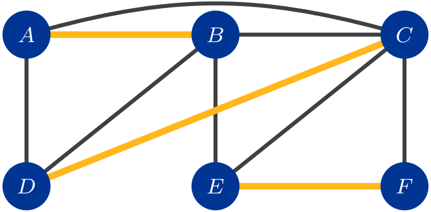
Can We Apply This Elsewhere?

- In some countries, students are matched to high schools using the Gale-Shapley algorithm.
- Directly applying the Gale-Shapley algorithm to course registration can cause some snags (Tilley/Reid, 2020)
- What if preferences can change, or if they are conditional?

Can We Apply This Elsewhere?

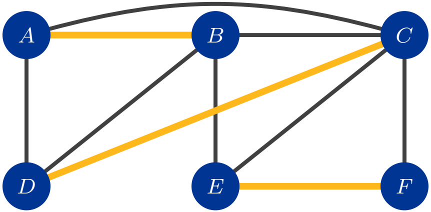
- In some countries, students are matched to high schools using the Gale-Shapley algorithm.
- Directly applying the Gale-Shapley algorithm to course registration can cause some snags (Tilley/Reid, 2020)
- What if preferences can change, or if they are conditional?
- Can we play matchmaker with a population of people?

Consider our graph from earlier:



Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

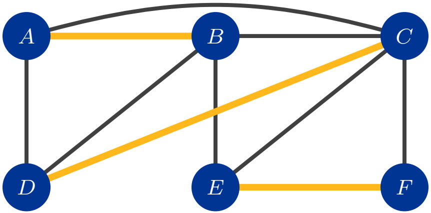
Consider our graph from earlier:



Here we found a stable matching. Is that always possible?

Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

Consider our graph from earlier:

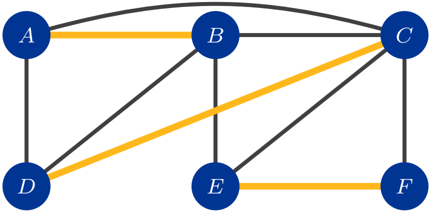


Here we found a stable matching. Is that always possible?

Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
$\boldsymbol{E}$	$\boxed{C}$	D	$\boxed{F}$	B	
F	E	C			

In a different world, maybe E prefers to be matched with C instead of F. Now what happens?

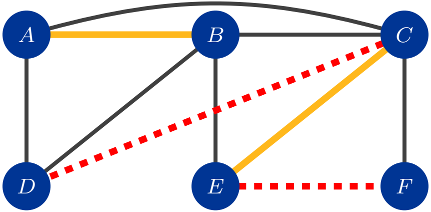
Consider our graph from earlier:



Node	Preference Order				
A	$\mathbf{B}$	F	C	E	D
B	F	$\mathbf{A}$	E	D	C
C	F	E	A	B	$\mathbf{D}$
D	B	A	$\mathbf{C}$		
E	C	D	$\mathbf{F}$	B	
F	$\mathbf{E}$	C			

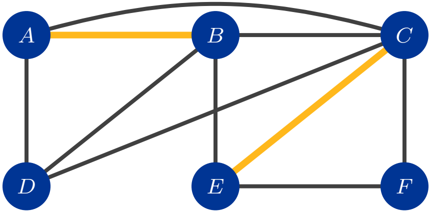
In a different world, maybe E prefers to be matched with C instead of F. Now what happens?

Consider our graph from earlier:



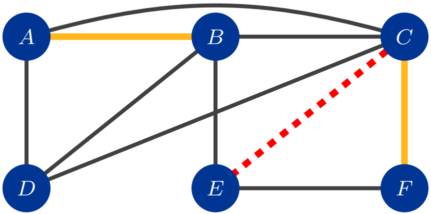
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



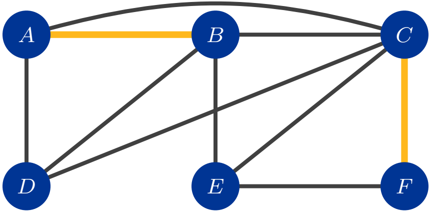
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



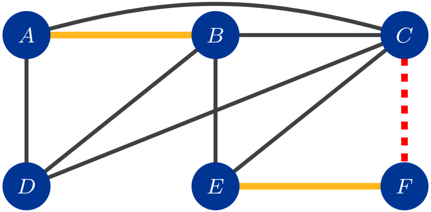
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



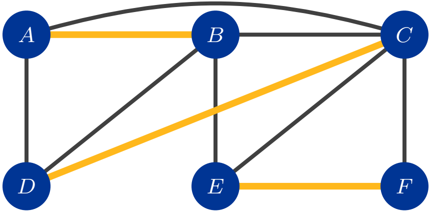
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



Node	Preference Order				
A	$\mathbf{B}$	F	C	E	D
B	F	$\mathbf{A}$	E	D	C
C	$\mathbf{F}$	E	A	B	D
D	B	A	C		
E	C	D	$\mathbf{F}$	B	
F	$\mathbf{E}$	$\mathbf{C}$			

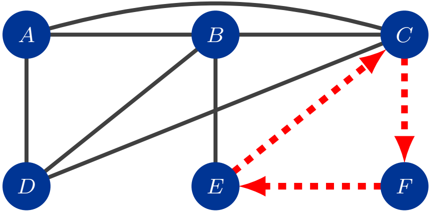
Consider our graph from earlier:



Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

And we're back to where we started...

Consider our graph from earlier:

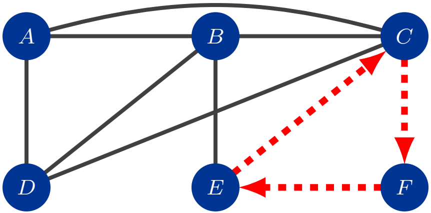


And we're back to where we started...

Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	$\boldsymbol{F}$	$\boldsymbol{E}$	A	B	D
D	B	A	C		
E	$\boldsymbol{C}$	D	$\boldsymbol{F}$	B	
F	$\boldsymbol{E}$	$\boldsymbol{C}$			

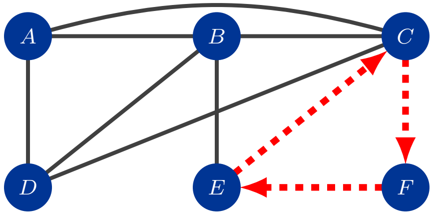
This cycle caused a problem. And since none of C, E, F prefer someone else more, there can be no stable matching.

The Stable "Roommates" Problem



Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	$\boldsymbol{F}$	$\boldsymbol{E}$	A	B	D
D	B	A	C		
E	$\boldsymbol{C}$	D	$\boldsymbol{F}$	B	
F	$\boldsymbol{E}$	$\boldsymbol{C}$			

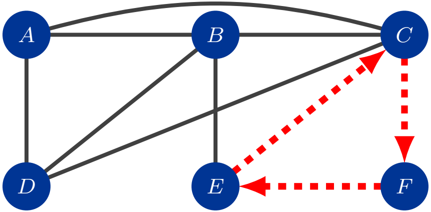
The Stable "Roommates" Problem



Bipartite graphs have no triangles, so this kind of cycle can never happen there.

Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	$\boldsymbol{F}$	$\boldsymbol{E}$	A	B	D
D	B	A	C		
E	$\boldsymbol{C}$	D	$\boldsymbol{F}$	B	
F	$\boldsymbol{E}$	$\boldsymbol{C}$			

The Stable "Roommates" Problem



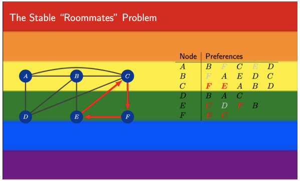
Bipartite graphs have no triangles, so this kind of cycle can never happen there.

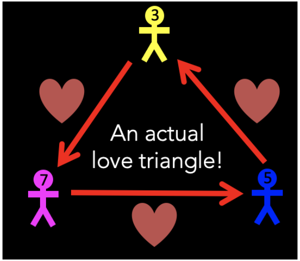
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	$\boldsymbol{F}$	$\boldsymbol{E}$	A	B	D
D	B	A	C		
E	$\boldsymbol{C}$	D	$\boldsymbol{F}$	B	
F	$\boldsymbol{E}$	$\boldsymbol{C}$			

Hence, if we don't assume that everyone is straight, relationships become *mathematically* harder!

Ways I've Told this Story

- A big twist at the end
- A way to connect parts of my (and others') identities
- A lecture on an application of graph theory, with a personal touch
- A case study in how our values shape the math we do
  - What choices are made in the setup and the application?
  - Wondering helps discover/create new math!
- A puzzle for my students
- An discussion on why certain reality TV shows should have a largely queer cast





Thank You!

- JMM 2026
- Spectra (especially Prof. Apoorva Mate)
- Prof. Jeffrey Wheeler (University of Pittsburgh)
- Prof. Elizabeth Reid (Marist University)
- ...and you all, for your time and attention!

Further Questions

- In the bipartite graph case, can we find a stable matching that balances out each groups' preferences?
  - (Viet/Lee/Trang/Chung 2016) use simulations to approximate optimality.
- (Irving 1985): Algorithm shows when the Stable "Roommates" Problem has a stable matching on $2n$ vertices, and finds one if it exists.
  - (Cseh/Manlove 2016) show that allowing for "unacceptable partners" (as we've done throughout) makes the problem NP-hard.
- Can we extend this problem (and the idea of stability) to allow for "matches" of 3 or more people?
  - (Biró 2007) defines a "stable fractional matching" in a hypergraph, and proves that one always exists.