Explaining the Math of Queer Relationship Dynamics

Edison Hauptman[1]

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

April 27, 2024

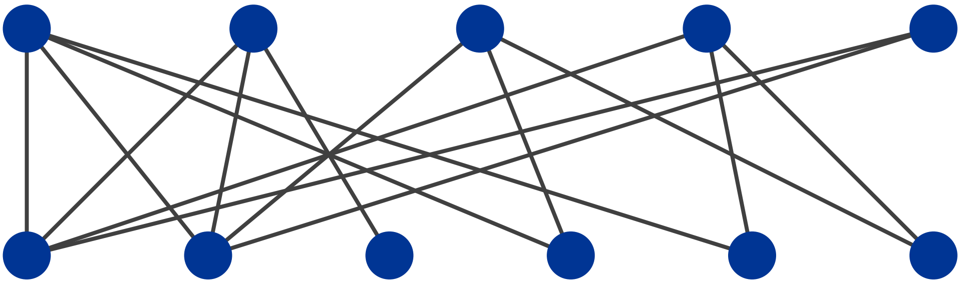
Outline

1. Preliminaries
2. The Stable Matching Problem
3. The (More Modern) Stable Marriage Problem
4. Further Investigations

Preliminaries

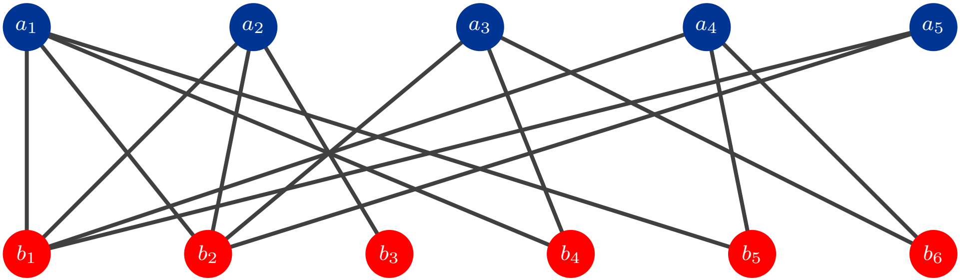
Bipartite Graphs

Consider a graph $G = (V, E)$:



Bipartite Graphs

Consider a graph $G = (V, E)$:



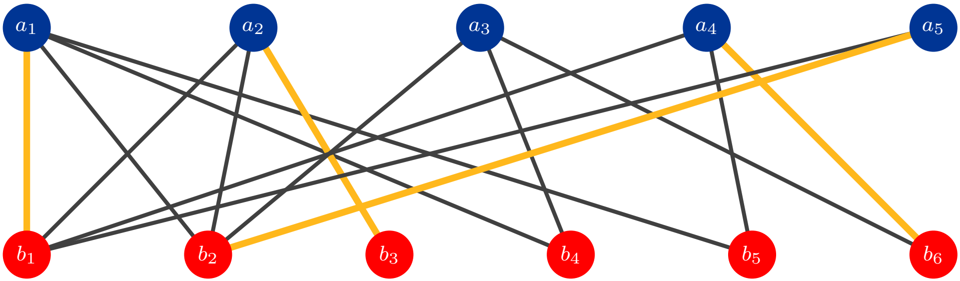
Definition

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 $b \in B$.

Matchings

Definition

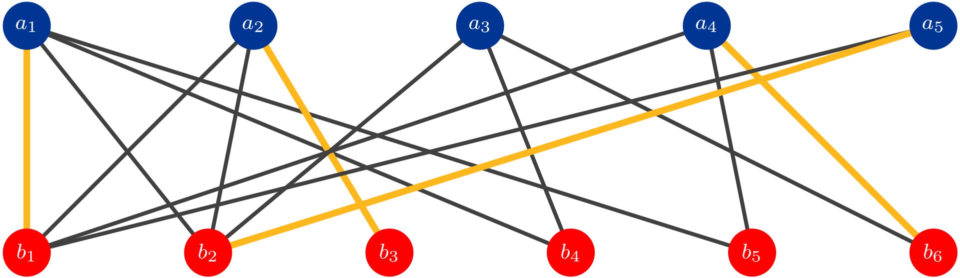
A **matching** in G is a subset of edges $E' \subset E$ such that each vertex belongs to at most one edge.



Matchings

Definition

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

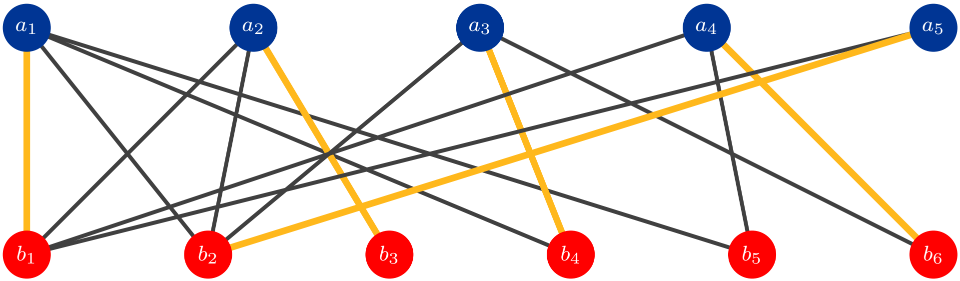


We will look at maximal matchings, which are not a subset of any other matching.

Matchings

Definition

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

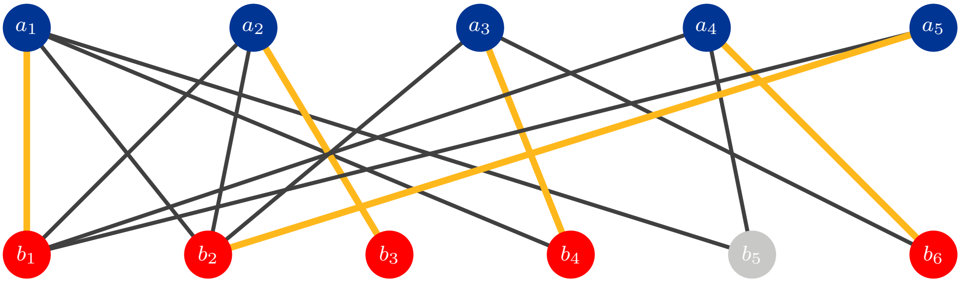


We will look at maximal matchings, which are not a subset of any other matching.

Matchings

Definition

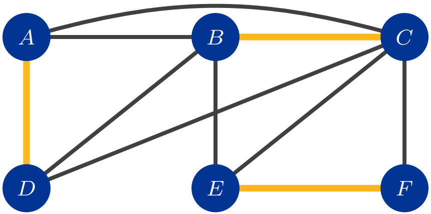
A **matching** in G is a subset of edges $E' \subset E$ such that each vertex belongs to at most one edge.



We will look at maximal matchings, which are not a subset of any other matching.

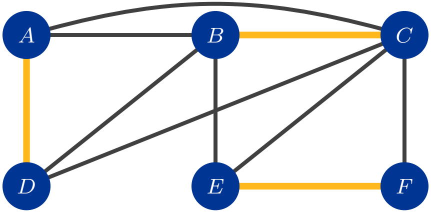
What Makes a Matching Stable?

Each vertex ranks its interest in being matched with another vertex:



What Makes a Matching Stable?

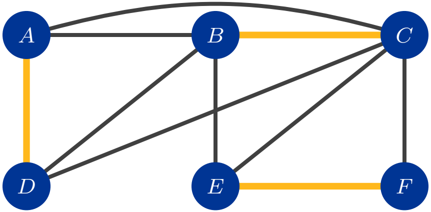
Each vertex ranks its interest in being matched with another vertex:



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?

Each vertex ranks its interest in being matched with another vertex:



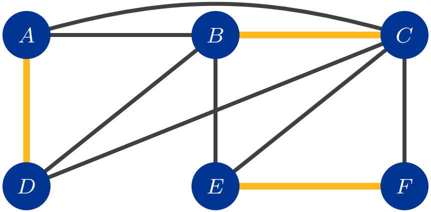
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	F	D	C	B	
F	E	C			

Definition

A matching is **unstable** if there are $i, j \in V$ such that i and j "prefer each other" to their current partners.

What Makes a Matching Stable?

Each vertex ranks its interest in being matched with another vertex:



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

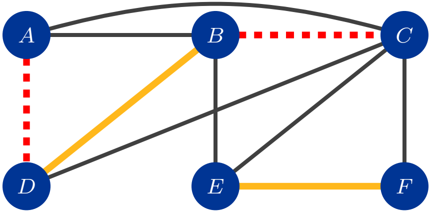
A matching is **unstable** if there are $i, j \in V$ such that i and j "prefer each other" to their current partners.

Definition

A matching is **stable** if there does not exist an unstable pair of vertices.

What Makes a Matching Stable?

Each vertex ranks its interest in being matched with another vertex:



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

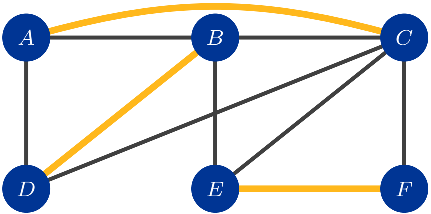
A matching is **unstable** if there are $i, j \in V$ such that i and j "prefer each other" to their current partners.

Definition

A matching is **stable** if there does not exist an unstable pair of vertices.

What Makes a Matching Stable?

Each vertex ranks its interest in being matched with another vertex:



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

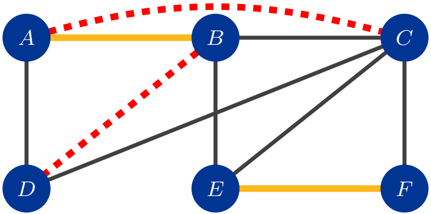
A matching is **unstable** if there are $i, j \in V$ such that i and j "prefer each other" to their current partners.

Definition

A matching is **stable** if there does not exist an unstable pair of vertices.

What Makes a Matching Stable?

Each vertex ranks its interest in being matched with another vertex:



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

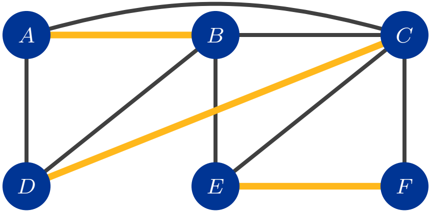
A matching is **unstable** if there are $i, j \in V$ such that i and j "prefer each other" to their current partners.

Definition

A matching is **stable** if there does not exist an unstable pair of vertices.

What Makes a Matching Stable?

Each vertex ranks its interest in being matched with another vertex:



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 there are $i, j \in V$ such that i and j "prefer each other" to their current partners.

Definition

A matching is **stable** if there does not exist an unstable pair of vertices.

The Stable Matching Problem

Example: The National Resident Matching Program

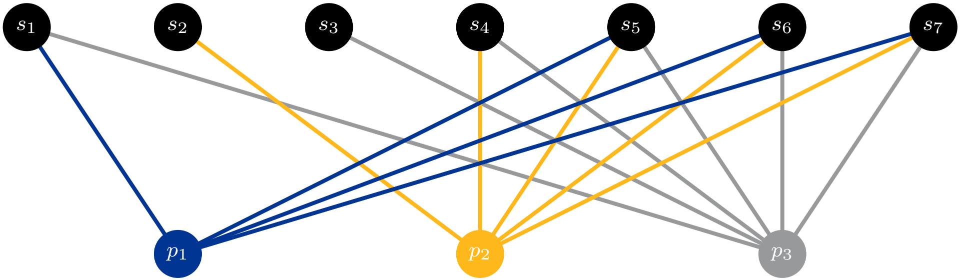
- Assigns medical students to residency programs.

Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- Note that this is a bipartite graph.

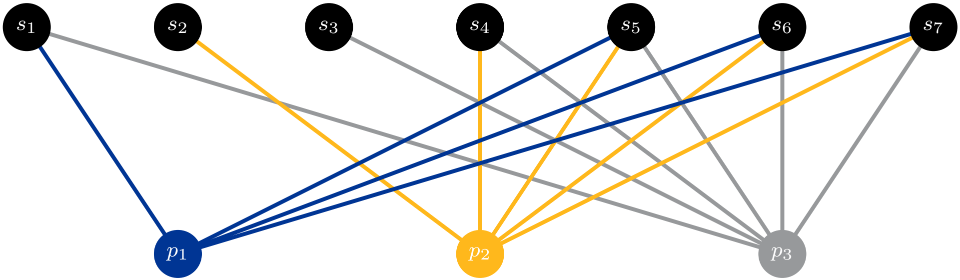
Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- Note that this is a bipartite graph.



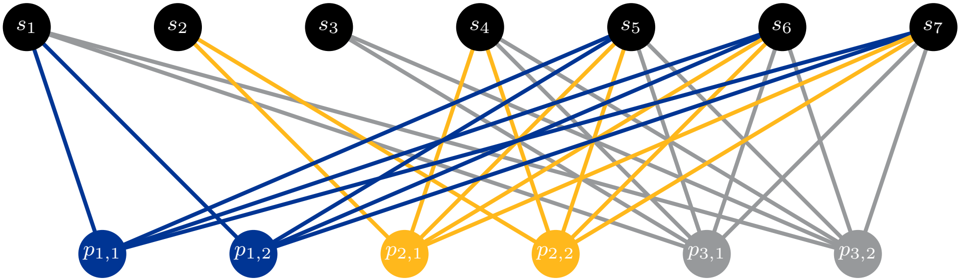
Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- Note that this is a bipartite graph.
- The programs can accept more than one student.



Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- Note that this is a bipartite graph.
- The programs can accept more than one student.



Example: The National Resident Matching Program

- Assigns medical students to residency programs.
- Note that this is a bipartite graph.
- 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 proposal 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 proposal to their top program that hasn't rejected them yet (they've already sent applications).
2. Each program considers all of its proposals 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 proposal to their top program that hasn't rejected them yet.**
2. Each program considers all of its proposals to this point, *temporarily* accepts the top ones it has space for, and rejects the rest.

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

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

Solution: The Gale-Shapley Algorithm (1962)

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

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

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

Solution: The Gale-Shapley Algorithm (1962)

1. Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.
2. **Each program considers all of its proposals 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. **Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.**
2. Each program considers all of its proposals 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. Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.
2. **Each program considers all of its proposals 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. **Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.**
2. Each program considers all of its proposals 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. Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.
2. **Each program considers all of its proposals 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. **Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.**
2. Each program considers all of its proposals 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. Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.
2. **Each program considers all of its proposals 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. **Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.**
2. Each program considers all of its proposals 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. Any students not currently in a program submit a proposal to their top program that hasn't rejected them yet.
2. **Each program considers all of its proposals 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~~	

How well does this work?

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

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

What Assumptions Did We Make?

- Consistently ordered preferences

What Assumptions Did We Make?

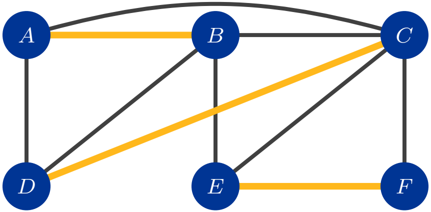
- Consistently ordered preferences *(This is a non-trivial assumption!)*

What Assumptions Did We Make?

- Consistently ordered preferences *(This is a non-trivial assumption!)*
- Bipartite graph

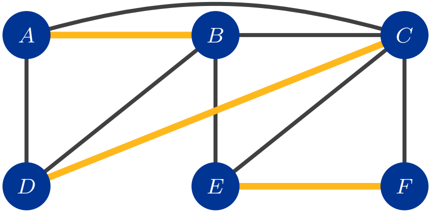
The (More Modern) Stable Marriage Problem

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			

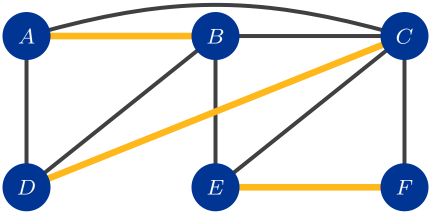
Consider our graph from earlier:



Here we found a stable matching. But can we always find one?

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:

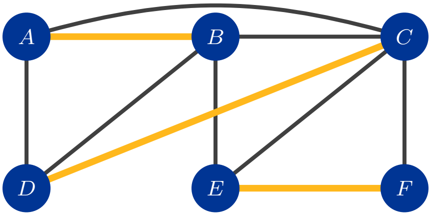


Here we found a stable matching. But can we always find one?

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			

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

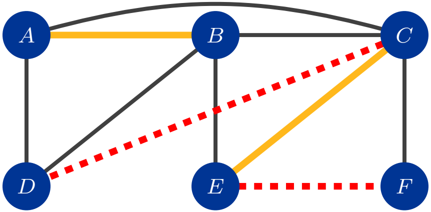
Consider our graph from earlier:



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			

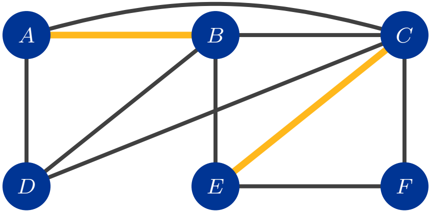
In a slightly different world, maybe E prefers to be matched with C instead of F. Now what happens?

Consider our graph from earlier:



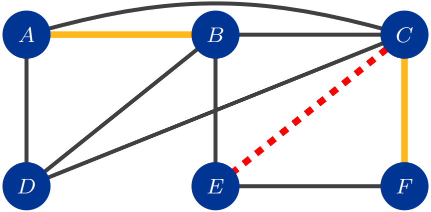
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



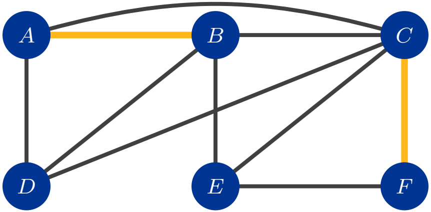
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



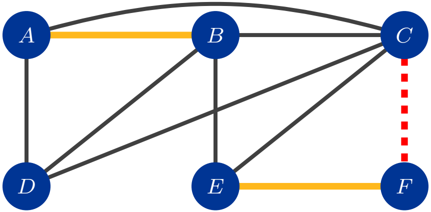
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



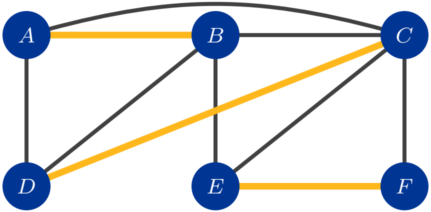
Node	Preference Order				
A	B	F	C	E	D
B	F	A	E	D	C
C	F	E	A	B	D
D	B	A	C		
E	C	D	F	B	
F	E	C			

Consider our graph from earlier:



Node	Preference Order				
A	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			

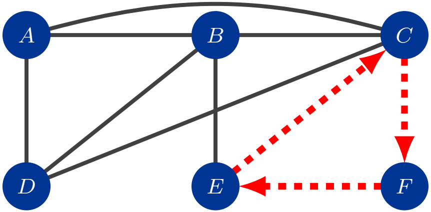
Consider our graph from earlier:



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			

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

Consider our graph from earlier:

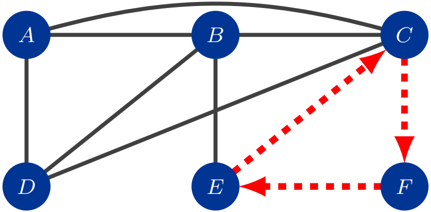


And we're back to where we started...

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	$\boldsymbol{E}$	$\boldsymbol{C}$			

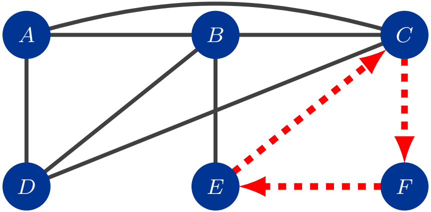
This cycle caused a problem. And since none of C, E, F prefer someone else more, there can be no stable matching.

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	$\boldsymbol{E}$	$\boldsymbol{C}$			

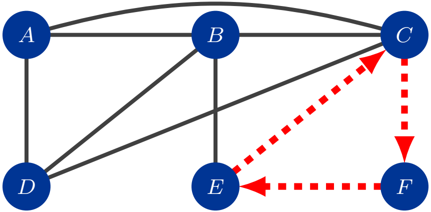
The Stable “Roommates” Problem



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	$\mathbf{F}$	$\mathbf{E}$	A	B	D
D	B	A	C		
E	$\mathbf{C}$	D	$\mathbf{F}$	B	
F	$\mathbf{E}$	$\mathbf{C}$			

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 remove the condition that everyone has to be straight, relationships become *mathematically* harder!

Thank You!

- MAA Maryland/DC/Virginia Region
- James Madison University Math Department
- University of Pittsburgh Math Department (especially Prof. Jeff Wheeler)
- Prof. Elizabeth Reid (Marist College)
- ...and you all, for your time and attention!

Further Investigations

Further Investigations

- In the Stable Marriage Problem (the boring one), 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.