Math and Your Love Life

Annie Raymond

University of Washington

March 21, 2016

The stable marriage problem

The stable marriage problem

Disclaimer: The following problem is very heteronormative and generally socially conservative in order to make the mathematics behind it easier. For this, I apologize.

The stable marriage problem

Disclaimer: The following problem is very heteronormative and generally socially conservative in order to make the mathematics behind it easier. For this, I apologize.

Problem: Given n girls and n boys and lists of their preferences, find a *stable matching*,

The stable marriage problem

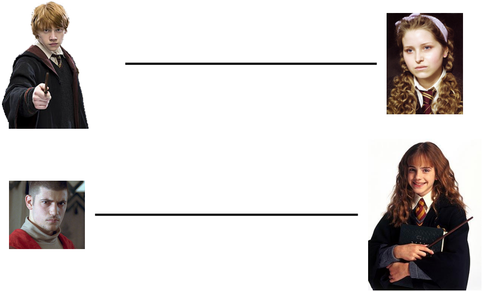
Disclaimer: The following problem is very heteronormative and generally socially conservative in order to make the mathematics behind it easier. For this, I apologize.

Problem: Given n girls and n boys and lists of their preferences, find a *stable matching*, i.e. we want everybody to be in a couple and we don't want a boy and a girl in two couples that would mutually prefer to be together.

The stable marriage problem

Disclaimer: The following problem is very heteronormative and generally socially conservative in order to make the mathematics behind it easier. For this, I apologize.

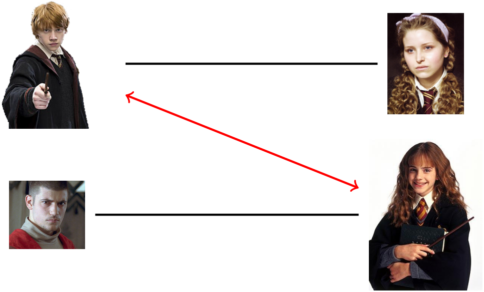
Problem: Given n girls and n boys and lists of their preferences, find a *stable matching*, i.e. we want everybody to be in a couple and we don't want a boy and a girl in two couples that would mutually prefer to be together.



The stable marriage problem

Disclaimer: The following problem is very heteronormative and generally socially conservative in order to make the mathematics behind it easier. For this, I apologize.

Problem: Given n girls and n boys and lists of their preferences, find a *stable matching*, i.e. we want everybody to be in a couple and we don't want a boy and a girl in two couples that would mutually prefer to be together.



Graph and matching

Graph and matching

Definition

We call a set of points, called *vertices*, and lines between those points, called *edges*, a *graph*.

Graph and matching

Definition

We call a set of points, called *vertices*, and lines between those points, called *edges*, a *graph*.

Definition

A subset of the edges of a graph such that each vertex is adjacent to at most one edge is called a *matching*.

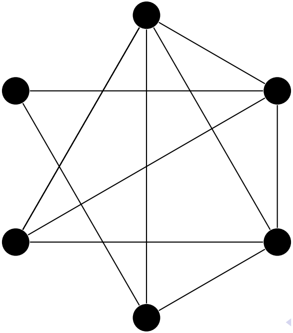
Graph and matching

Definition

We call a set of points, called *vertices*, and lines between those points, called *edges*, a *graph*.

Definition

A subset of the edges of a graph such that each vertex is adjacent to at most one edge is called a *matching*.



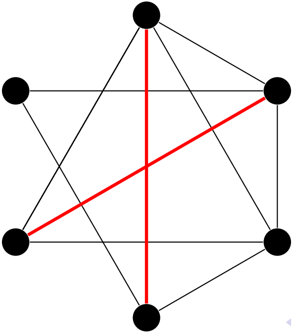
Graph and matching

Definition

We call a set of points, called *vertices*, and lines between those points, called *edges*, a *graph*.

Definition

A subset of the edges of a graph such that each vertex is adjacent to at most one edge is called a *matching*.



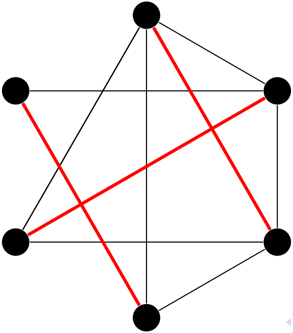
Graph and matching

Definition

We call a set of points, called *vertices*, and lines between those points, called *edges*, a *graph*.

Definition

A subset of the edges of a graph such that each vertex is adjacent to at most one edge is called a *matching*.



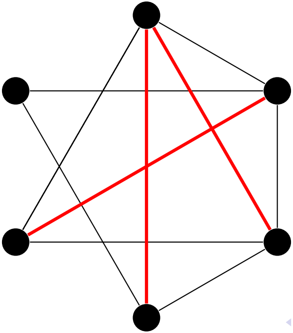
Graph and matching

Definition

We call a set of points, called *vertices*, and lines between those points, called *edges*, a *graph*.

Definition

A subset of the edges of a graph such that each vertex is adjacent to at most one edge is called a *matching*.



Bipartite graph and matching

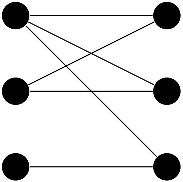
Definition

A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.

Bipartite graph and matching

Definition

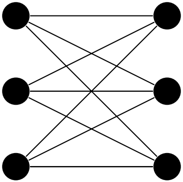
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

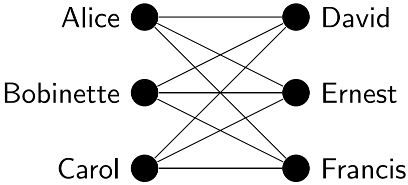
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

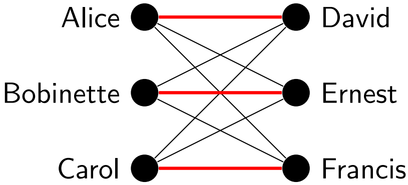
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

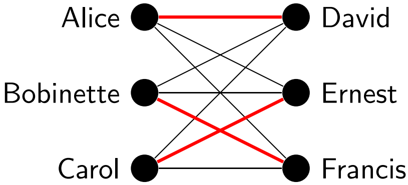
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

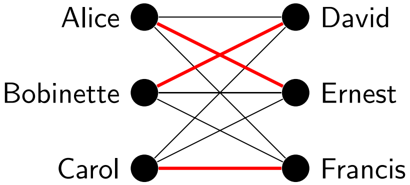
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

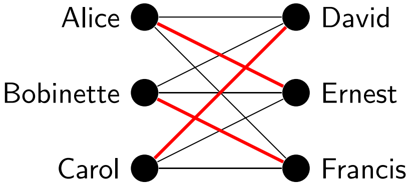
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

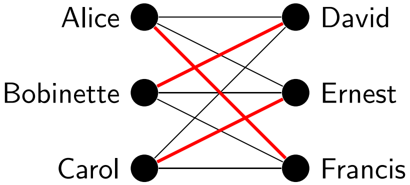
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

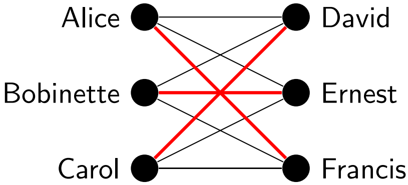
A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Bipartite graph and matching

Definition

A *bipartite graph* on vertex sets V_1 and V_2 is a graph for which no edge is going from a vertex in V_1 to another vertex in V_1 or from a vertex in V_2 to a vertex in V_2, i.e. any edge is going from a vertex in V_1 to a vertex in V_2.



Definition of a stable matching

Definition

A matching in a bipartite graph with n boys on one side and n girls on the other side is said to be *stable* if there doesn't exist a girl X who would rather be with Y than with her boyfriend and if boy Y would also rather be with X than with his girlfriend.

Definition of a stable matching

Definition

A matching in a bipartite graph with n boys on one side and n girls on the other side is said to be *stable* if there doesn't exist a girl X who would rather be with Y than with her boyfriend and if boy Y would also rather be with X than with his girlfriend.

Lists of preferences

Alice	Bobinette	Carol	David	Ernest	Francis
1. Francis	1. David	1. Francis	1. Carol	1. Alice	1. Carol
2. David	2. Ernest	2. Ernest	2. Alice	2. Bobinette	2. Bobinette
3. Ernest	3. Francis	3. David	3. Bobinette	3. Carol	3. Alice

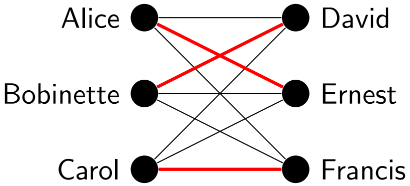
Definition of a stable matching

Definition

A matching in a bipartite graph with n boys on one side and n girls on the other side is said to be *stable* if there doesn't exist a girl X who would rather be with Y than with her boyfriend and if boy Y would also rather be with X than with his girlfriend.

Lists of preferences

Alice	Bobinette	Carol	David	Ernest	Francis
1. Francis	1. David	1. Francis	1. Carol	1. Alice	1. Carol
2. David	2. Ernest	2. Ernest	2. Alice	2. Bobinette	2. Bobinette
3. Ernest	3. Francis	3. David	3. Bobinette	3. Carol	3. Alice



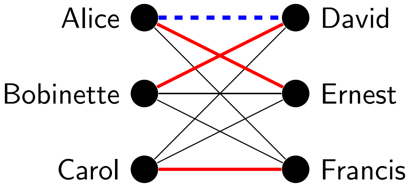
Definition of a stable matching

Definition

A matching in a bipartite graph with n boys on one side and n girls on the other side is said to be *stable* if there doesn't exist a girl X who would rather be with Y than with her boyfriend and if boy Y would also rather be with X than with his girlfriend.

Lists of preferences

Alice	Bobinette	Carol	David	Ernest	Francis
1. Francis	1. David	1. Francis	1. Carol	1. Alice	1. Carol
2. David	2. Ernest	2. Ernest	2. Alice	2. Bobinette	2. Bobinette
3. Ernest	3. Francis	3. David	3. Bobinette	3. Carol	3. Alice



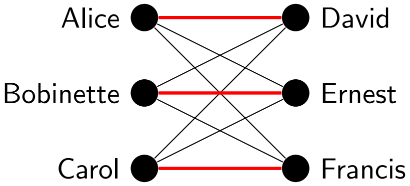
Definition of a stable matching

Definition

A matching in a bipartite graph with n boys on one side and n girls on the other side is said to be *stable* if there doesn't exist a girl X who would rather be with Y than with her boyfriend and if boy Y would also rather be with X than with his girlfriend.

Lists of preferences

Alice	Bobinette	Carol	David	Ernest	Francis
1. Francis	1. David	1. Francis	1. Carol	1. Alice	1. Carol
2. David	2. Ernest	2. Ernest	2. Alice	2. Bobinette	2. Bobinette
3. Ernest	3. Francis	3. David	3. Bobinette	3. Carol	3. Alice



Question: Does there always exists a stable matching no matter what n is and what the lists of preferences are?

Question: Does there always exists a stable matching no matter what n is and what the lists of preferences are?

Answer: YES! There exists an algorithm to find a stable matching. This algorithm was first described by Gale and Shapley in 1962.

Question: Does there always exists a stable matching no matter what n is and what the lists of preferences are?
Answer: YES! There exists an algorithm to find a stable matching. This algorithm was first described by Gale and Shapley in 1962.
It's assumed that

1. the number of girls is equal to the number of boys and everyone is heterosexual

Question: Does there always exists a stable matching no matter what n is and what the lists of preferences are?
Answer: YES! There exists an algorithm to find a stable matching. This algorithm was first described by Gale and Shapley in 1962.
It's assumed that

1. the number of girls is equal to the number of boys and everyone is heterosexual
2. each person has ranked all the members of the opposite sex in order of preference

Question: Does there always exists a stable matching no matter what n is and what the lists of preferences are?
Answer: YES! There exists an algorithm to find a stable matching. This algorithm was first described by Gale and Shapley in 1962.
It's assumed that

1. the number of girls is equal to the number of boys and everyone is heterosexual
2. each person has ranked all the members of the opposite sex in order of preference
3. everybody would rather be in a relationship with their worst choice than to be alone

Question: Does there always exists a stable matching no matter what n is and what the lists of preferences are?

Answer: YES! There exists an algorithm to find a stable matching. This algorithm was first described by Gale and Shapley in 1962.

It's assumed that

1. the number of girls is equal to the number of boys and everyone is heterosexual
2. each person has ranked all the members of the opposite sex in order of preference
3. everybody would rather be in a relationship with their worst choice than to be alone

We will now reenact the algorithm.

The stable marriage algorithm

While there exists a boy who is not in a relationship:

The stable marriage algorithm

While there exists a boy who is not in a relationship:

1. every boy who is not in a relationship asks out the girl who he ranks highest and who hasn't rejected him yet

The stable marriage algorithm

While there exists a boy who is not in a relationship:

1. every boy who is not in a relationship asks out the girl who he ranks highest and who hasn't rejected him yet
2. every girl who has more than one boy who wants to be with her rejects all of them but the one she ranks highest among them

The stable marriage algorithm

While there exists a boy who is not in a relationship:

1. every boy who is not in a relationship asks out the girl who he ranks highest and who hasn't rejected him yet
2. every girl who has more than one boy who wants to be with her rejects all of them but the one she ranks highest among them
3. every rejected boy is now not in a relationship

Analysis of the algorithm

Analysis of the algorithm

- **At the end, everybody has a partner.**

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

$\Rightarrow$ Every girl eventually gets asked out, and the algorithm ends.

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

$\Rightarrow$ Every girl eventually gets asked out, and the algorithm ends.

- **This set of couples is stable.**

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

  $\Rightarrow$ Every girl eventually gets asked out, and the algorithm ends.
- **This set of couples is stable.**
  Suppose not.

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

  $\Rightarrow$ Every girl eventually gets asked out, and the algorithm ends.
- **This set of couples is stable.**
  Suppose not.
  - Then there exists a boy and a girl, say Ron and Hermione, who are not together and who would prefer to be together than with their respective partners, say Lavender and Krum.

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

$\Rightarrow$ Every girl eventually gets asked out, and the algorithm ends.

- **This set of couples is stable.**
  Suppose not.
  - Then there exists a boy and a girl, say Ron and Hermione, who are not together and who would prefer to be together than with their respective partners, say Lavender and Krum.
  - Then Ron must have asked out Hermione before asking out Lavender since Hermione ranks higher than Lavender on his list.

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

  $\Rightarrow$ Every girl eventually gets asked out, and the algorithm ends.
- **This set of couples is stable.**
  Suppose not.
  - Then there exists a boy and a girl, say Ron and Hermione, who are not together and who would prefer to be together than with their respective partners, say Lavender and Krum.
  - Then Ron must have asked out Hermione before asking out Lavender since Hermione ranks higher than Lavender on his list.
  - Thus Hermione must have rejected him because she preferred to be with some other boy (Krum or someone else that she ranked lower than Krum but higher than Ron).

Analysis of the algorithm

- **At the end, everybody has a partner.**
  - As long as there is a girl who hasn't been asked out, there will be rejections and new asking out since n boys are fighting for at most $n-1$ girls.
  - No boy can ask out the same girl more than once.

  $\Rightarrow$ Every girl eventually gets asked out, and the algorithm ends.
- **This set of couples is stable.**
  Suppose not.
  - Then there exists a boy and a girl, say Ron and Hermione, who are not together and who would prefer to be together than with their respective partners, say Lavender and Krum.
  - Then Ron must have asked out Hermione before asking out Lavender since Hermione ranks higher than Lavender on his list.
  - Thus Hermione must have rejected him because she preferred to be with some other boy (Krum or someone else that she ranked lower than Krum but higher than Ron).

  $\Rightarrow$ Thus Hermione cannot prefer Ron to Krum and the set of couples is stable.

Switching up the algorithm

Observation: The algorithm is not symmetric for girls and boys.

Switching up the algorithm

Observation: The algorithm is not symmetric for girls and boys.

What happens if the roles of the girls and boys are switched?

Switching up the algorithm

Observation: The algorithm is not symmetric for girls and boys.

What happens if the roles of the girls and boys are switched?

While there exists a girl who is not in a relationship:

1. every girl who is not in a relationship asks out the boy she ranks highest and who hasn't rejected her yet
2. every boy who has more than one girl who wants to be with him rejects all of them but the one he ranks highest among them
3. every rejected girl is now not in a relationship

Best- and worst-stable

Definition

Consider all possible stable matchings. Look at the set S_X of the ranks of the persons that X gets paired with in the different stable matchings; the person that X rates highest in S_X is called his or her *best-stable* partner and the person that X rates lowest in S_x is called his or her *worst-stable* partner.

Best- and worst-stable

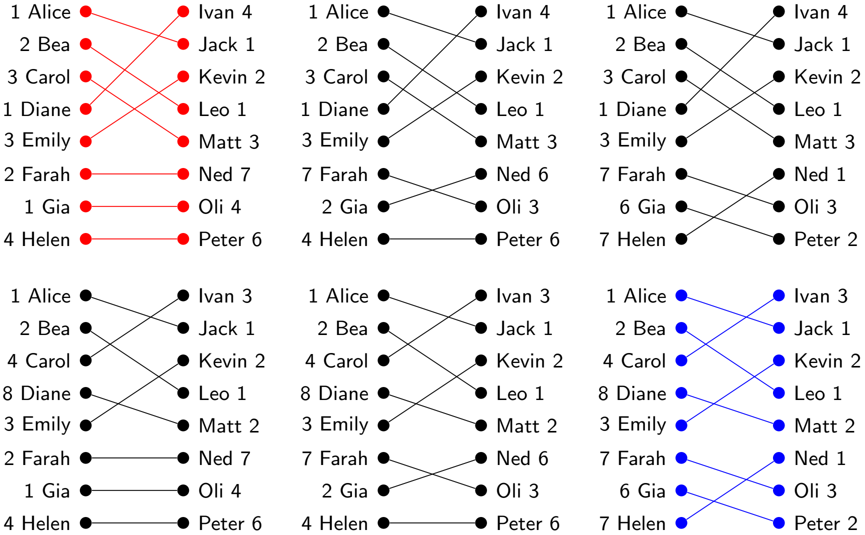
Definition

Consider all possible stable matchings. Look at the set S_X of the ranks of the persons that X gets paired with in the different stable matchings; the person that X rates highest in S_X is called his or her *best-stable* partner and the person that X rates lowest in S_x is called his or her *worst-stable* partner.

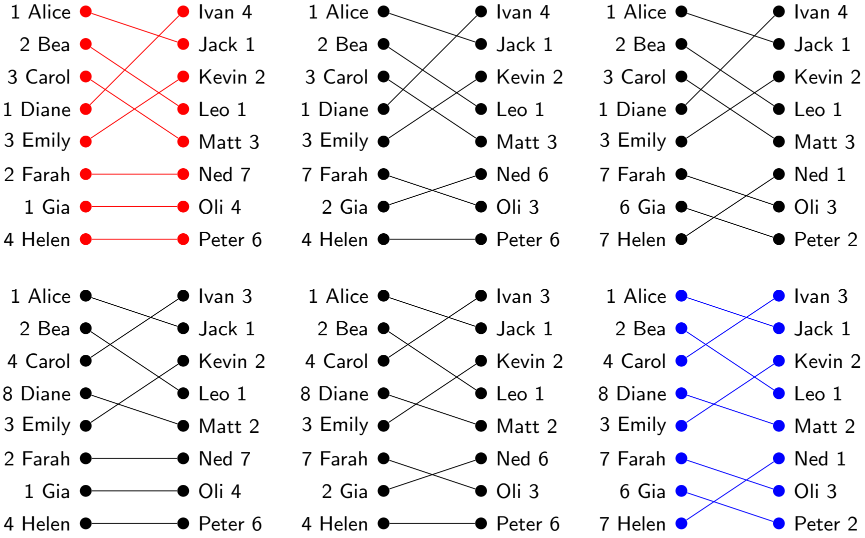
Proposition

In the algorithm, the members of the gender doing the 'asking out' get their best-stable partner, and the members of the other gender get their worst-stable partner.

Our example

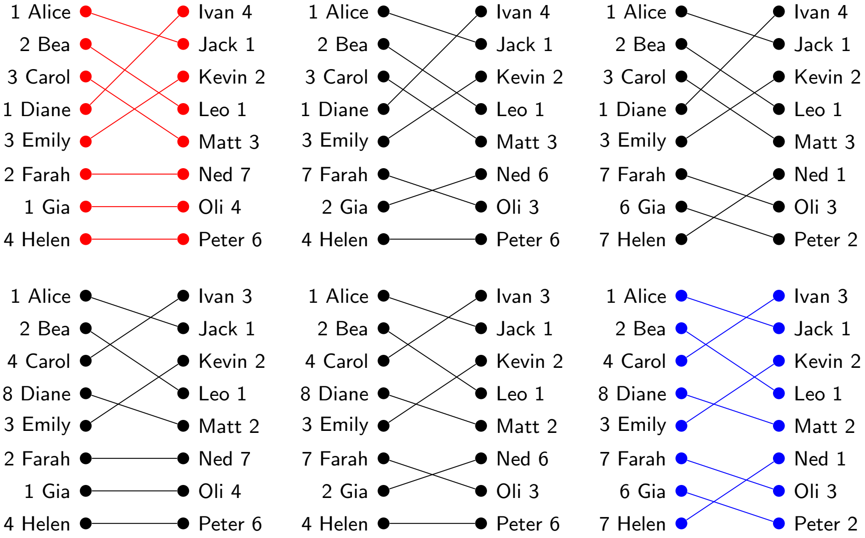

1 Alice
2 Bea
3 Carol
1 Diane
3 Emily
2 Farah
1 Gia
4 Helen
Ivan 4
Jack 1
Kevin 2
Leo 1
Matt 3
Ned 7
Oli 4
Peter 6
1 Alice
2 Bea
3 Carol
1 Diane
3 Emily
7 Farah
2 Gia
4 Helen
Ivan 4
Jack 1
Kevin 2
Leo 1
Matt 3
Ned 6
Oli 3
Peter 6
1 Alice
2 Bea
3 Carol
1 Diane
3 Emily
7 Farah
6 Gia
7 Helen
Ivan 4
Jack 1
Kevin 2
Leo 1
Matt 3
Ned 1
Oli 3
Peter 2
1 Alice
2 Bea
4 Carol
8 Diane
3 Emily
2 Farah
1 Gia
4 Helen
Ivan 3
Jack 1
Kevin 2
Leo 1
Matt 2
Ned 7
Oli 4
Peter 6
1 Alice
2 Bea
4 Carol
8 Diane
3 Emily
7 Farah
2 Gia
4 Helen
Ivan 3
Jack 1
Kevin 2
Leo 1
Matt 2
Ned 6
Oli 3
Peter 6
1 Alice
2 Bea
4 Carol
8 Diane
3 Emily
7 Farah
6 Gia
7 Helen
Ivan 3
Jack 1
Kevin 2
Leo 1
Matt 2
Ned 1
Oli 3
Peter 2

Our example



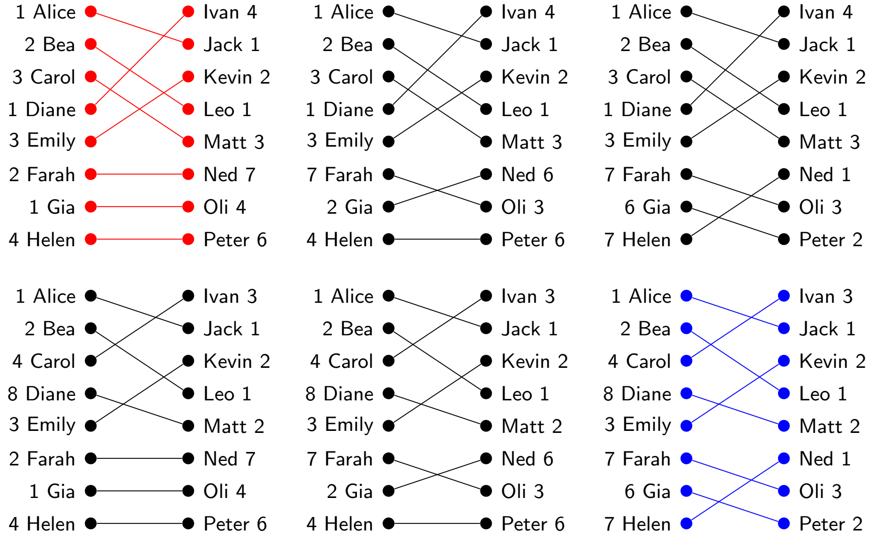
$S_A = \{1\}$

Our example



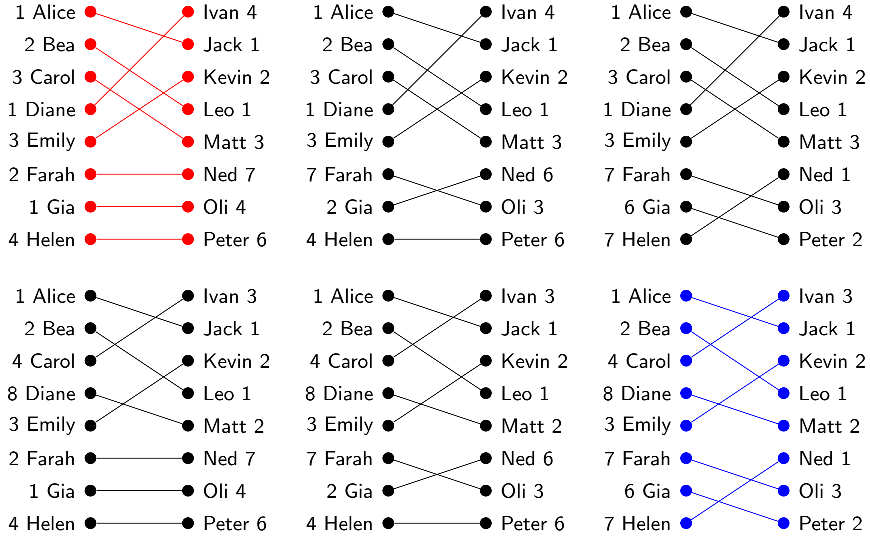
$S_A = \{1\}$, $S_B = \{2\}$

Our example



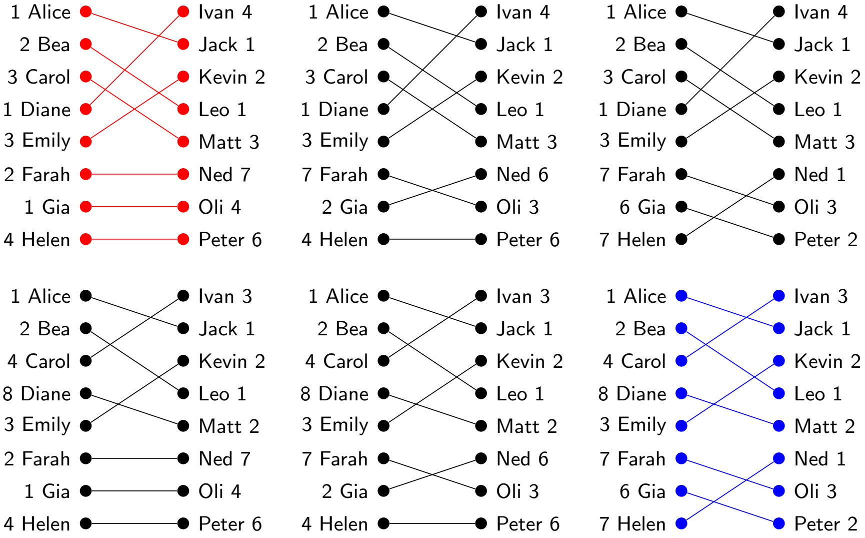
$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3, 4\}$

Our example



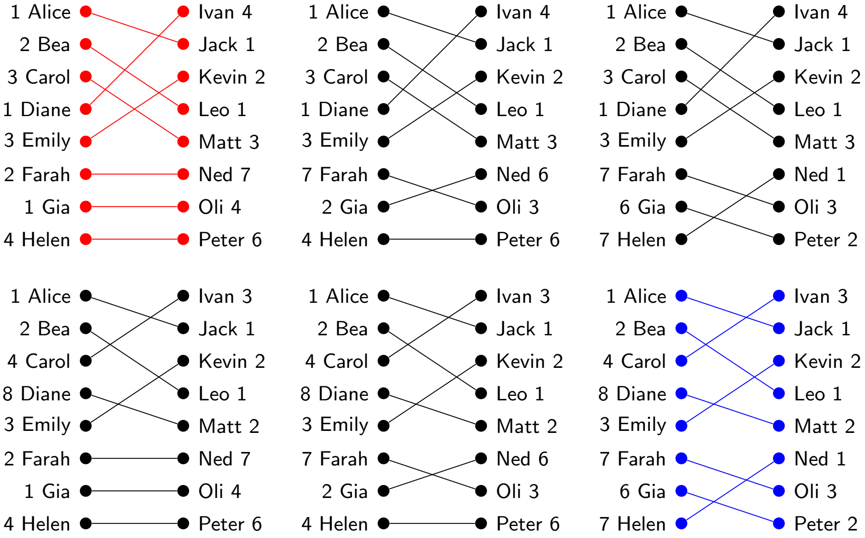
$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3, 4\}$, $S_D = \{1, 8\}$

Our example



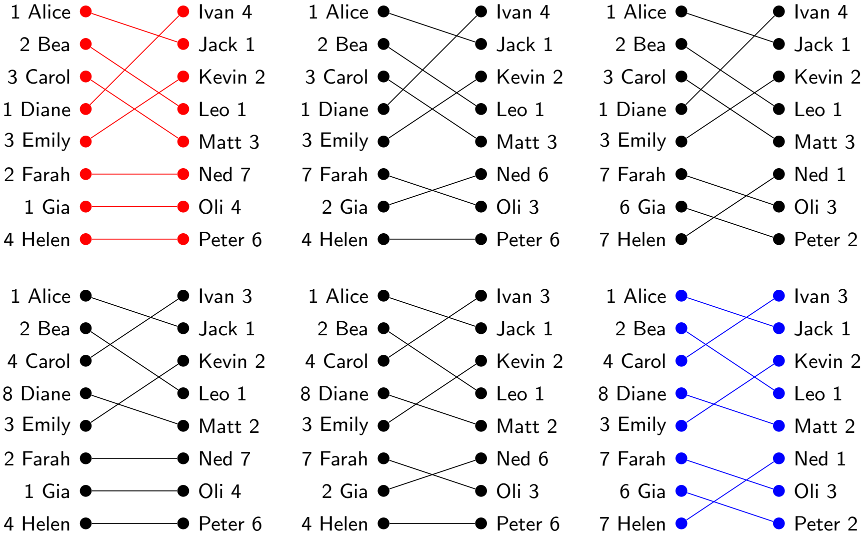
$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$

Our example



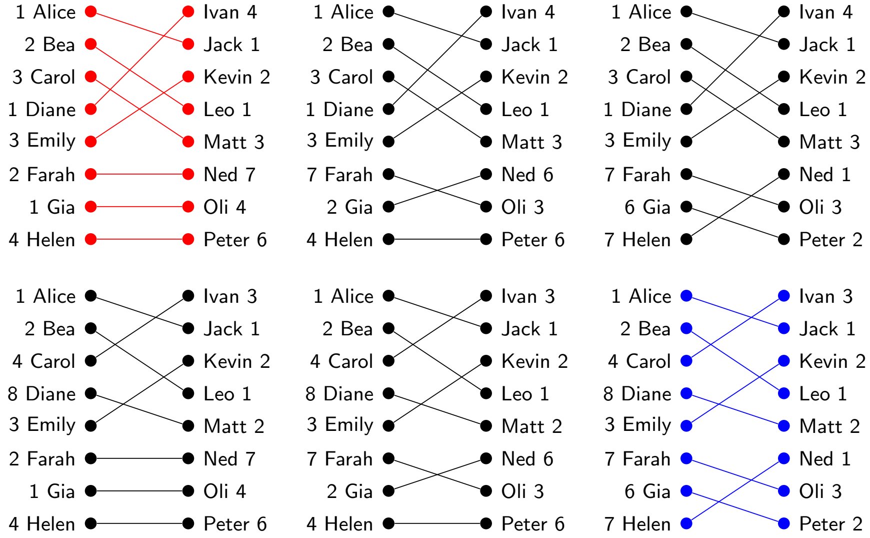
$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3, 4\}$, $S_D = \{1, 8\}$, $S_E = \{3\}$, $S_F = \{2, 7\}$,

Our example



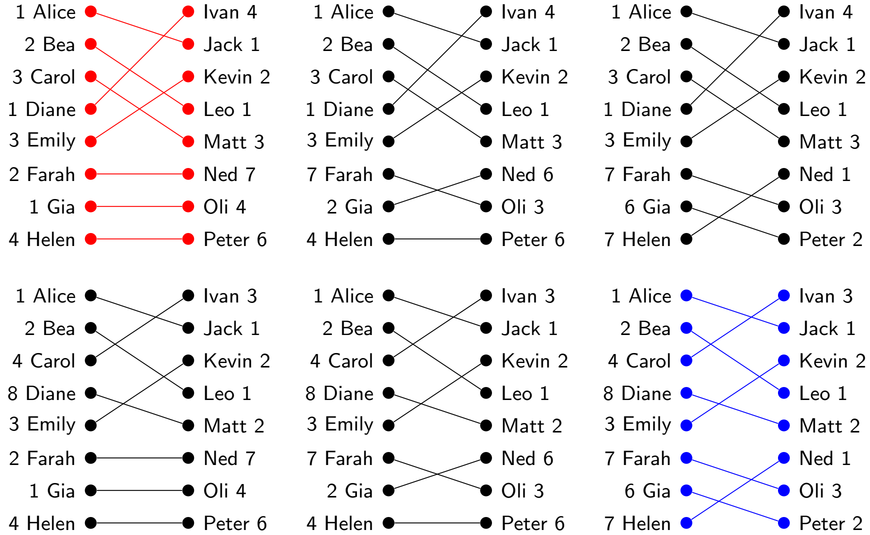
$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3, 4\}$, $S_D = \{1, 8\}$, $S_E = \{3\}$, $S_F = \{2, 7\}$, $S_G = \{1, 2, 6\}$,

Our example



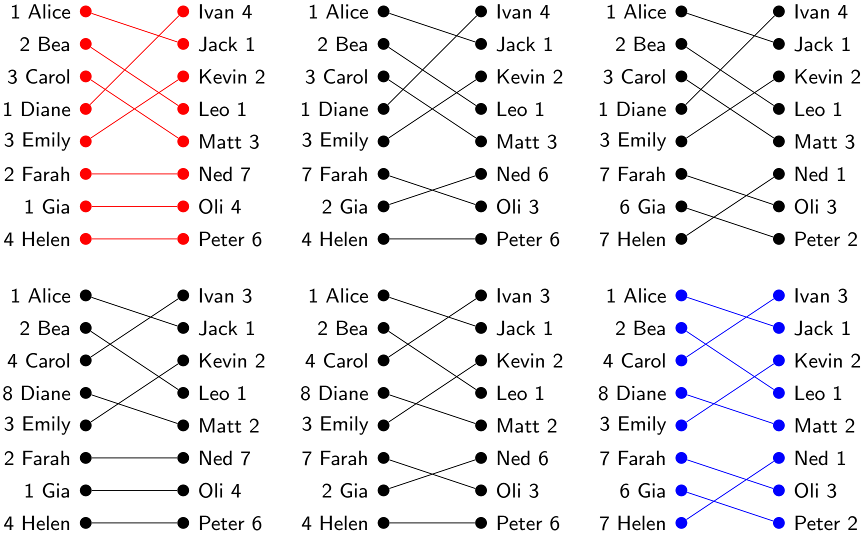
$S_A = \{1\},\ S_B = \{2\},\ S_C = \{3,4\},\ S_D = \{1,8\},\ S_E = \{3\},\ S_F = \{2,7\},\ S_G = \{1,2,6\},\ S_H = \{4,7\}$

Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$, $S_F = \{2,7\}$, $S_G = \{1,2,6\}$, $S_H = \{4,7\}$
$S_I = \{3,4\}$

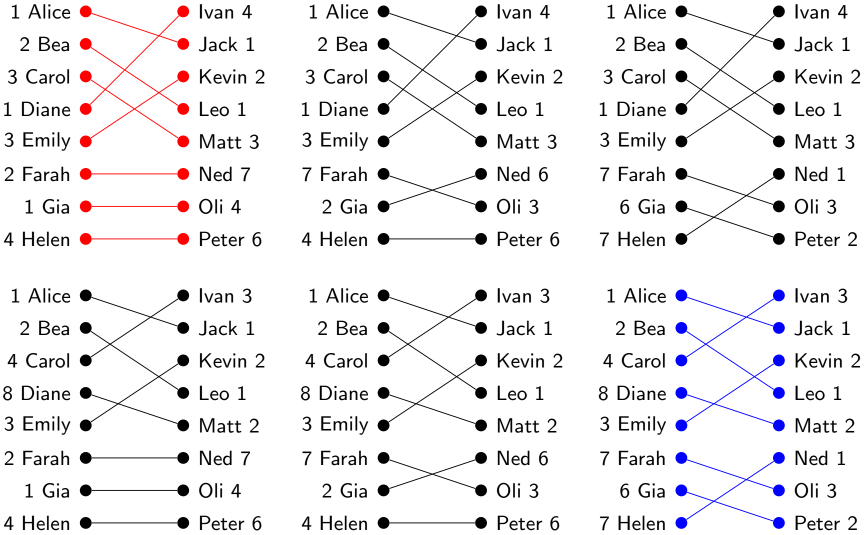
Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$, $S_F = \{2,7\}$, $S_G = \{1,2,6\}$, $S_H = \{4,7\}$

$S_I = \{3,4\}$, $S_J = \{1\}$

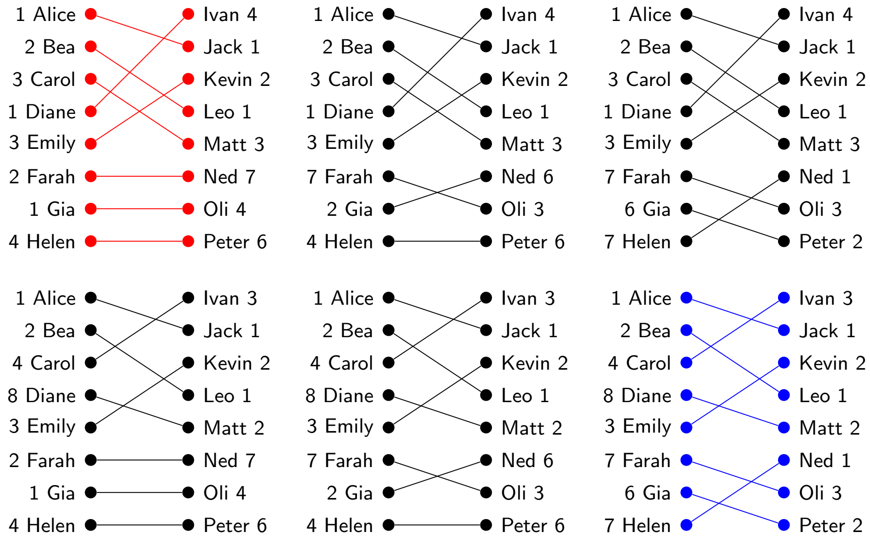
Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$, $S_F = \{2,7\}$, $S_G = \{1,2,6\}$, $S_H = \{4,7\}$

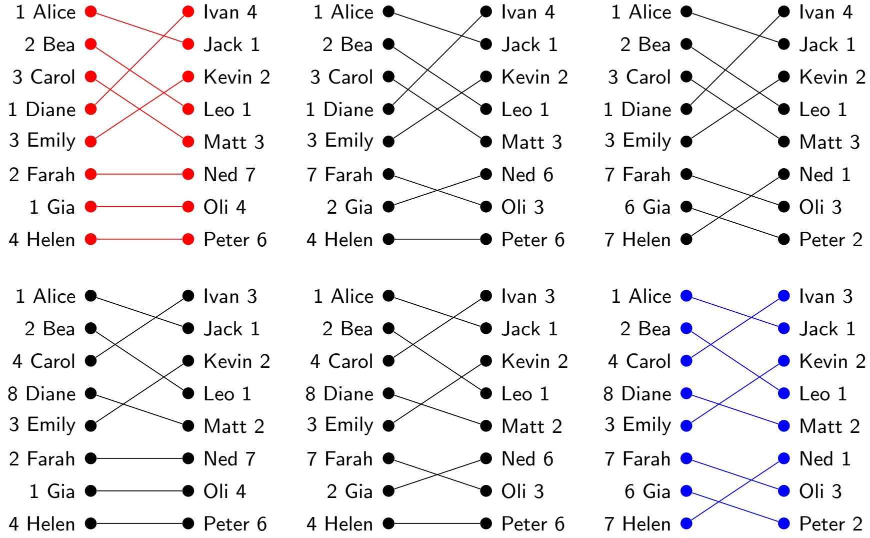
$S_I = \{3,4\}$, $S_J = \{1\}$, $S_K = \{2\}$

Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$, $S_F = \{2,7\}$, $S_G = \{1,2,6\}$, $S_H = \{4,7\}$
$S_I = \{3,4\}$, $S_J = \{1\}$, $S_K = \{2\}$, $S_L = \{1\}$

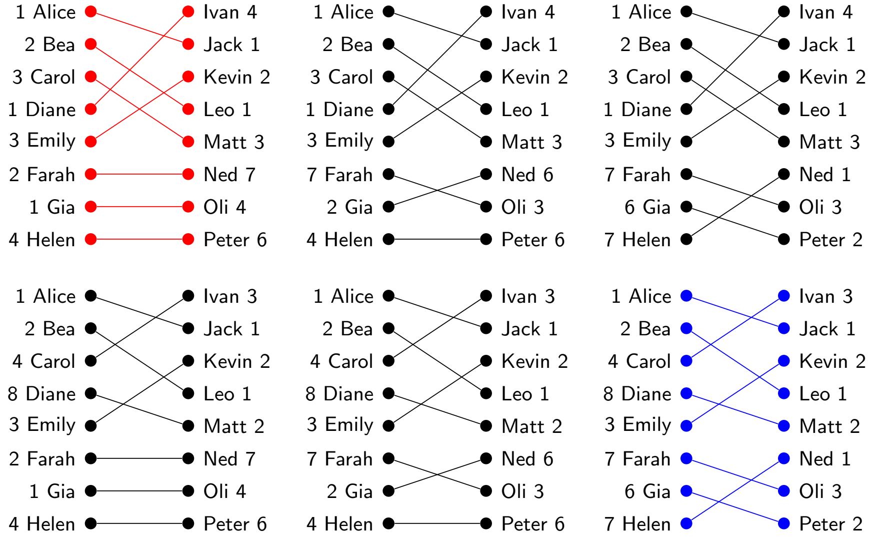
Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3, 4\}$, $S_D = \{1, 8\}$, $S_E = \{3\}$, $S_F = \{2, 7\}$, $S_G = \{1, 2, 6\}$, $S_H = \{4, 7\}$

$S_I = \{3, 4\}$, $S_J = \{1\}$, $S_K = \{2\}$, $S_L = \{1\}$, $S_M = \{2, 3\}$

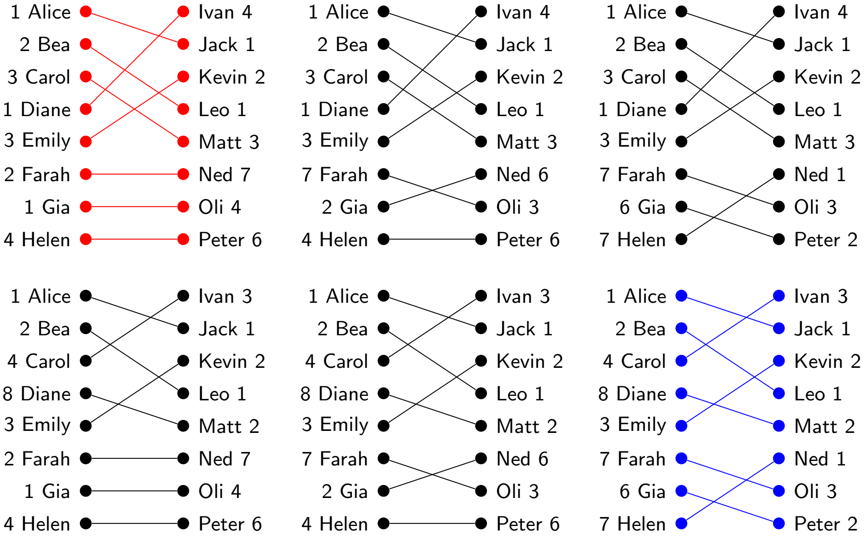
Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$, $S_F = \{2,7\}$, $S_G = \{1,2,6\}$, $S_H = \{4,7\}$

$S_I = \{3,4\}$, $S_J = \{1\}$, $S_K = \{2\}$, $S_L = \{1\}$, $S_M = \{2,3\}$, $S_N = \{1,6,7\}$

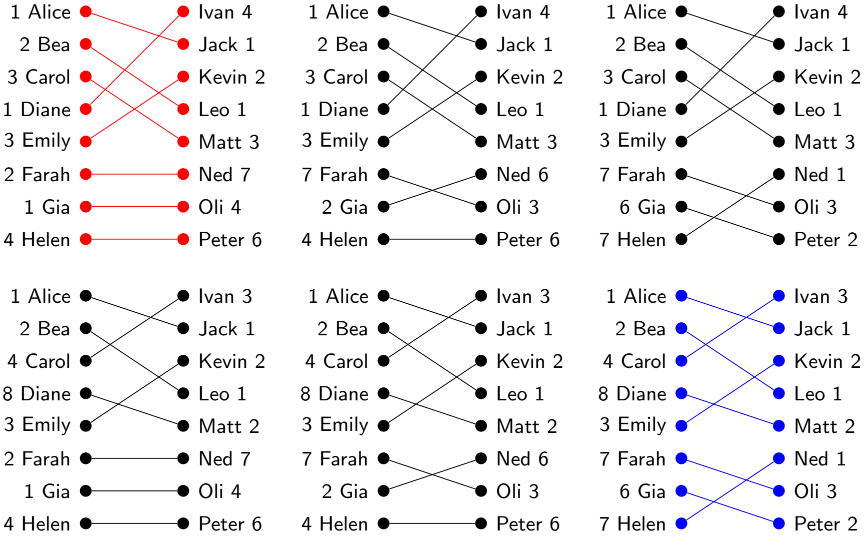
Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$, $S_F = \{2,7\}$, $S_G = \{1,2,6\}$, $S_H = \{4,7\}$

$S_I = \{3,4\}$, $S_J = \{1\}$, $S_K = \{2\}$, $S_L = \{1\}$, $S_M = \{2,3\}$, $S_N = \{1,6,7\}$, $S_O = \{3,4\}$

Our example



$S_A = \{1\}$, $S_B = \{2\}$, $S_C = \{3,4\}$, $S_D = \{1,8\}$, $S_E = \{3\}$, $S_F = \{2,7\}$, $S_G = \{1,2,6\}$, $S_H = \{4,7\}$

$S_I = \{3,4\}$, $S_J = \{1\}$, $S_K = \{2\}$, $S_L = \{1\}$, $S_M = \{2,3\}$, $S_N = \{1,6,7\}$, $S_O = \{3,4\}$, $S_P = \{2,6\}$

Boys get their best-stable girlfriend

Proposition

In the algorithm where boys ask girls out, each boy gets his best-stable girlfriend.

Boys get their best-stable girlfriend

Proposition

In the algorithm where boys ask girls out, each boy gets his best-stable girlfriend.

Proof.

- Suppose not: suppose that some boy is rejected by his best-stable girlfriend in the algorithm.

Boys get their best-stable girlfriend

Proposition

In the algorithm where boys ask girls out, each boy gets his best-stable girlfriend.

Proof.

- Suppose not: suppose that some boy is rejected by his best-stable girlfriend in the algorithm.
- Let i be the earliest round in which a boy, say Ron, gets rejected by his best-stable girlfriend, say Hermione

Boys get their best-stable girlfriend

Proposition

In the algorithm where boys ask girls out, each boy gets his best-stable girlfriend.

Proof.

- Suppose not: suppose that some boy is rejected by his best-stable girlfriend in the algorithm.
- Let i be the earliest round in which a boy, say Ron, gets rejected by his best-stable girlfriend, say Hermione
- Hermione rejected Ron because she preferred some other man, say Krum

Boys get their best-stable girlfriend

Proposition

In the algorithm where boys ask girls out, each boy gets his best-stable girlfriend.

Proof.

- Suppose not: suppose that some boy is rejected by his best-stable girlfriend in the algorithm.
- Let i be the earliest round in which a boy, say Ron, gets rejected by his best-stable girlfriend, say Hermione
- Hermione rejected Ron because she preferred some other man, say Krum
- Krum hasn't been rejected by his best-stable girl (by the definition of i)

$\Rightarrow$ either Hermione is the best-stable woman of Krum or she is better than his best-stable woman.

Boys get their best-stable girlfriend (continued)

Proof continued.

Ron gets rejected by Hermione because she prefers Krum, who likes her at least as much as his best-stable girlfriend.

Boys get their best-stable girlfriend (continued)

Proof continued.

Ron gets rejected by Hermione because she prefers Krum, who likes her at least as much as his best-stable girlfriend.

We now show that any matching M where Ron and Hermione are together is not stable, a contradiction to the fact that Hermione is Ron's best-stable girlfriend:

Boys get their best-stable girlfriend (continued)

Proof continued.

Ron gets rejected by Hermione because she prefers Krum, who likes her at least as much as his best-stable girlfriend.

We now show that any matching M where Ron and Hermione are together is not stable, a contradiction to the fact that Hermione is Ron's best-stable girlfriend:

- Krum ranks Hermione higher than his girlfriend in M, since Hermione is at least as good as his best-stable girlfriend.

Boys get their best-stable girlfriend (continued)

Proof continued.

Ron gets rejected by Hermione because she prefers Krum, who likes her at least as much as his best-stable girlfriend.

We now show that any matching M where Ron and Hermione are together is not stable, a contradiction to the fact that Hermione is Ron's best-stable girlfriend:

- Krum ranks Hermione higher than his girlfriend in M, since Hermione is at least as good as his best-stable girlfriend.
- Hermione ranks Krum higher than Ron since she rejects Ron for him in the algorithm.

Boys get their best-stable girlfriend (continued)

Proof continued.

Ron gets rejected by Hermione because she prefers Krum, who likes her at least as much as his best-stable girlfriend.

We now show that any matching M where Ron and Hermione are together is not stable, a contradiction to the fact that Hermione is Ron's best-stable girlfriend:

- Krum ranks Hermione higher than his girlfriend in M, since Hermione is at least as good as his best-stable girlfriend.
- Hermione ranks Krum higher than Ron since she rejects Ron for him in the algorithm.

We reached a contradiction, and so our first assumption that some boy is rejected by his best-stable girlfriend in the algorithm is wrong

Boys get their best-stable girlfriend (continued)

Proof continued.

Ron gets rejected by Hermione because she prefers Krum, who likes her at least as much as his best-stable girlfriend.

We now show that any matching M where Ron and Hermione are together is not stable, a contradiction to the fact that Hermione is Ron's best-stable girlfriend:

- Krum ranks Hermione higher than his girlfriend in M, since Hermione is at least as good as his best-stable girlfriend.
- Hermione ranks Krum higher than Ron since she rejects Ron for him in the algorithm.

We reached a contradiction, and so our first assumption that some boy is rejected by his best-stable girlfriend in the algorithm is wrong $\Rightarrow$ every boy in the algorithm gets matched to his best-stable girlfriend.

$\square$

Girls get their worst-stable boyfriend

Proposition

In the algorithm where boys ask girls out, each girl gets her worst-stable boyfriend.

Girls get their worst-stable boyfriend

Proposition

In the algorithm where boys ask girls out, each girl gets her worst-stable boyfriend.

Proof.

- Suppose there exists a stable matching M where some girl, say Hermione, gets a worse boy, say Krum, than in the algorithm, say Ron.

Girls get their worst-stable boyfriend

Proposition

In the algorithm where boys ask girls out, each girl gets her worst-stable boyfriend.

Proof.

- Suppose there exists a stable matching M where some girl, say Hermione, gets a worse boy, say Krum, than in the algorithm, say Ron.
- We know Hermione likes Ron better than Krum by assumption.

Girls get their worst-stable boyfriend

Proposition

In the algorithm where boys ask girls out, each girl gets her worst-stable boyfriend.

Proof.

- Suppose there exists a stable matching M where some girl, say Hermione, gets a worse boy, say Krum, than in the algorithm, say Ron.
- We know Hermione likes Ron better than Krum by assumption.
- Ron likes Hermione better than his girlfriend in M since the algorithm gives him his best-stable girlfriend.

Girls get their worst-stable boyfriend

Proposition

In the algorithm where boys ask girls out, each girl gets her worst-stable boyfriend.

Proof.

- Suppose there exists a stable matching M where some girl, say Hermione, gets a worse boy, say Krum, than in the algorithm, say Ron.
- We know Hermione likes Ron better than Krum by assumption.
- Ron likes Hermione better than his girlfriend in M since the algorithm gives him his best-stable girlfriend.
- M is not stable, a contradiction

Girls get their worst-stable boyfriend

Proposition

In the algorithm where boys ask girls out, each girl gets her worst-stable boyfriend.

Proof.

- Suppose there exists a stable matching M where some girl, say Hermione, gets a worse boy, say Krum, than in the algorithm, say Ron.
- We know Hermione likes Ron better than Krum by assumption.
- Ron likes Hermione better than his girlfriend in M since the algorithm gives him his best-stable girlfriend.
- M is not stable, a contradiction $\Rightarrow$ every girl gets her worst-stable boyfriend in the algorithm

□

Girls get their worst-stable boyfriend

Proposition

In the algorithm where boys ask girls out, each girl gets her worst-stable boyfriend.

Proof.

- Suppose there exists a stable matching M where some girl, say Hermione, gets a worse boy, say Krum, than in the algorithm, say Ron.
- We know Hermione likes Ron better than Krum by assumption.
- Ron likes Hermione better than his girlfriend in M since the algorithm gives him his best-stable girlfriend.
- M is not stable, a contradiction $\Rightarrow$ every girl gets her worst-stable boyfriend in the algorithm

□

Conclusion: Girls should ask boys out!

Variants of the problem

- Not same number of people on both sides

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences
  $\Rightarrow$ Easy

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences
  $\Rightarrow$ Easy
- Not complete list of preferences

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences
  $\Rightarrow$ Easy
- Not complete list of preferences
  $\Rightarrow$ Easy

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences
  $\Rightarrow$ Easy
- Not complete list of preferences
  $\Rightarrow$ Easy
- Not always strict preferences and not complete lists of preferences

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences
  $\Rightarrow$ Easy
- Not complete list of preferences
  $\Rightarrow$ Easy
- Not always strict preferences and not complete lists of preferences
  $\Rightarrow$ hard, 2-approximation

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences
  $\Rightarrow$ Easy
- Not complete list of preferences
  $\Rightarrow$ Easy
- Not always strict preferences and not complete lists of preferences
  $\Rightarrow$ hard, 2-approximation
- Gay/lesbian/bisexual stable marriage problem (like stable roommate problem)

Variants of the problem

- Not same number of people on both sides
  $\Rightarrow$ Easy, application to hospitals
- Not always strict preferences
  $\Rightarrow$ Easy
- Not complete list of preferences
  $\Rightarrow$ Easy
- Not always strict preferences and not complete lists of preferences
  $\Rightarrow$ hard, 2-approximation
- Gay/lesbian/bisexual stable marriage problem (like stable roommate problem)
  $\Rightarrow$ no guarantee of finding a stable matching!

Thank you!