Introduction to CS 4104

T. M. Murali

January 17, 2024

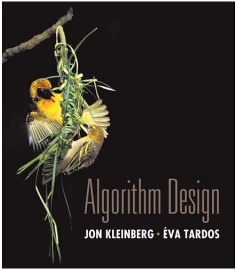
Course Information

- Instructor
  - T. M. Murali, D&DS 432, murali@cs.vt.edu
  - Office Hours: 10:30am–12pm, Mondays and Thursdays and by appointment
- Teaching assistants
  - Mehdi Esmaili (GTA), mesmaili@vt.edu
    Office Hours: 2:30pm–3:30pm, Mondays and Wednesdays, Room TBD.
  - Changqi (Charlie) Sun (GTA), changqi@vt.edu
    Office Hours: 9am–11am, Fridays, MCB 106
  - Michael Kim (UTA), michaelkim2002@vt.edu
    Office Hours: TBD
- Class meeting times: MW 4pm–5:15pm, Goodwin 115

Keeping in Touch

- Course web site: http://bioinformatics.cs.vt.edu/~murali/teaching/2024-spring-cs4104/ updated regularly through the semester
- Content on the course web site serves as the syllabus.
- Piazza: https://piazza.com/vt/spring2024/cs_4104_13402_202401/home announcements, including homeworks and exams.
- Canvas: homework/exam submissions and solutions, grades

Required Course Textbook



- Algorithm Design
- Jon Kleinberg and Éva Tardos
- Addison-Wesley
- 2006
- ISBN: 0-321-29535-8

Course Goals

- Learn methods and principles to construct algorithms.
- Learn techniques to analyze algorithms mathematically for correctness and efficiency (e.g., running time and space used).
- Schedule roughly follows the topics suggested in textbook
  - Stable matching
  - Measures of algorithm complexity
  - Graphs (may skip)
  - Greedy algorithms
  - Divide and conquer (briefly)
  - Dynamic programming
  - Network flow problems
  - NP-completeness
  - Coping with intractability
  - Approximation algorithms

Required Readings

- Reading assignment available on the website.
- Read **before** class.
- I strongly encourage you to keep up with the reading. Will make the class much easier.

Lecture Slides

- Will be available on class web site.
- Usually posted just before class.
- Class attendance is important.

Lecture Slides

- Will be available on class web site.
- Usually posted just before class.
- Class attendance is important. Lecture in class contains significant and substantial additions to material on the slides.
- I will not be taking attendance.

Homeworks

- Posted on the web site $\approx$ one week before due date.
- Announced via Piazza.
- Prepare solutions digitally and upload on Canvas.
  - Solution preparation recommended in LaTeX.
  - Do not submit handwritten solutions!

Homeworks

- Posted on the web site $\approx$ one week before due date.
- Announced via Piazza.
- Prepare solutions digitally and upload on Canvas.
  - Solution preparation recommended in LaTeX.
  - Do not submit handwritten solutions!
- Homework grading: lenient at beginning but gradually become stricter over the semester.
- Essential that you read posted homework solutions to learn how to describe algorithms and write proofs.

Examinations

- Take-home midterm.
- Take-home final (comprehensive).
- Prepare digital solutions (recommend LaTeX).

Grades

- Homeworks: 7–8, 60% of the grade.
- Take-home midterm: 15% of the grade.
- Take-home final: 25% of the grade.

Honor Code

- Virginia Tech Graduate Honor Code applies to this class.
- In particular, assistance from any source on the internet or anyone else is a violation of the Honor Code. *Do not use ChatGPT or any similar AI-based systems.*
- Your work and solutions to the examinations must be only your own.
- Special policy for homeworks:
  - Work on the homework in pairs. You can bounce ideas off your partner.
  - Prepare solutions individually. *Identical or similar solutions to any problem in a homework violate the Honor Code.*

What is an Algorithm?

What is an Algorithm?

Dictionary A set of prescribed computational procedures for solving a problem; a step-by-step method for solving a problem.

Knuth, TAOCP An algorithm is a finite, definite, effective procedure, with some input and some output.

What is an Algorithm?

Dictionary: A set of prescribed computational procedures for solving a problem; a step-by-step method for solving a problem.

Knuth, TAOCP: An algorithm is a finite, definite, effective procedure, with some input and some output.

Two other important aspects:

1. **Correct**: We will be able to rigorously prove that the algorithm does what it is supposed to do.
2. **Efficient**: We will also prove that the algorithm runs in polynomial time. We will try to make it as fast as we can.

Origin of the word “Algorithm”

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa.

Origin of the word “Algorithm”

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja’far Mohammed ben Musa.
2. From Al Gore, the former U.S. vice-president who “invented the internet”; joint winner of the Nobel Peace Prize in 2007.

Origin of the word "Algorithm"

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa.
2. From Al Gore, the former U.S. vice-president who "invented the internet"; joint winner of the Nobel Peace Prize in 2007.
3. From the Greek *algos* (meaning "pain," also a root of "analgesic") and *rythmos* (meaning "flow," also a root of "rhythm").

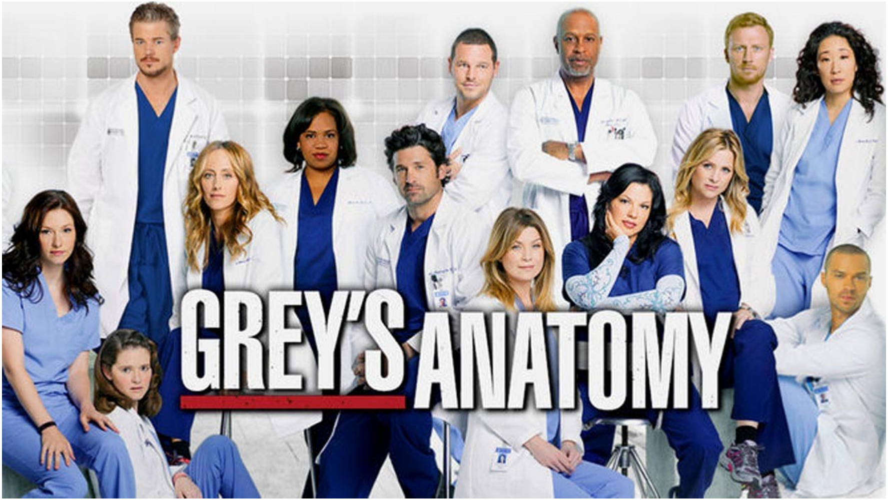
Origin of the word "Algorithm"

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa.
2. From Al Gore, the former U.S. vice-president who "invented the internet"; joint winner of the Nobel Peace Prize in 2007.
3. From the Greek *algos* (meaning "pain," also a root of "analgesic") and *rythmos* (meaning "flow," also a root of "rhythm"). "*Pain flowed through my body whenever I worked on CS 4104 homeworks.*" – student Thank-a-Teacher note.

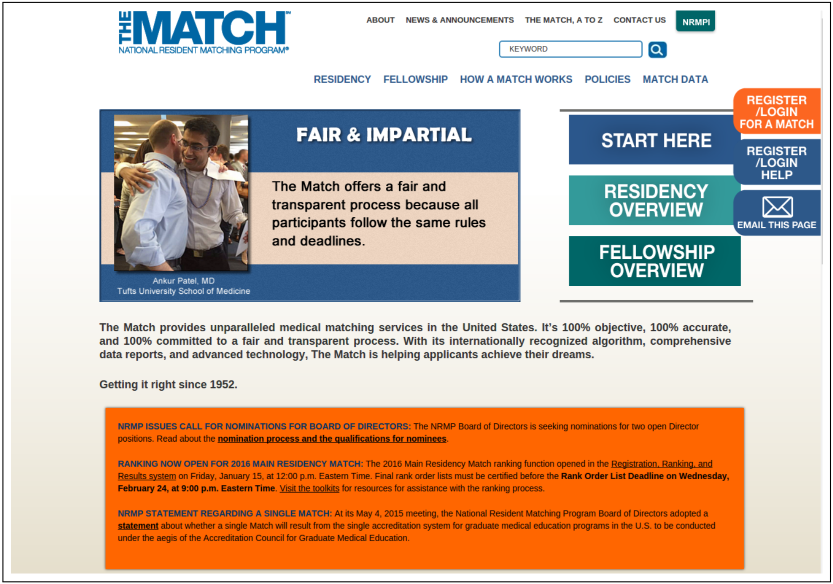
Modules: Lecture 1: Introduction: Algorithms

Origin of the word "Algorithm"

1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa. He wrote "Kitab al-jabr wa'l-muqabala," which evolved into today's high school algebra text.


GREY'S ANATOMY


THE
RESIDENT



THE MATCH
NATIONAL RESIDENT MATCHING PROGRAM®

ABOUT NEWS & ANNOUNCEMENTS THE MATCH, A TO Z CONTACT US NRMPI

KEYWORD

RESIDENCY FELLOWSHIP HOW A MATCH WORKS POLICIES MATCH DATA

FAIR & IMPARTIAL

The Match offers a fair and transparent process because all participants follow the same rules and deadlines.

Ankur Patel, MD
Tufts University School of Medicine

START HERE

RESIDENCY OVERVIEW

FELLOWSHIP OVERVIEW

REGISTER /LOGIN FOR A MATCH

REGISTER /LOGIN HELP

EMAIL THIS PAGE

The Match provides unparalleled medical matching services in the United States. It's 100% objective, 100% accurate, and 100% committed to a fair and transparent process. With its internationally recognized algorithm, comprehensive data reports, and advanced technology, The Match is helping applicants achieve their dreams.

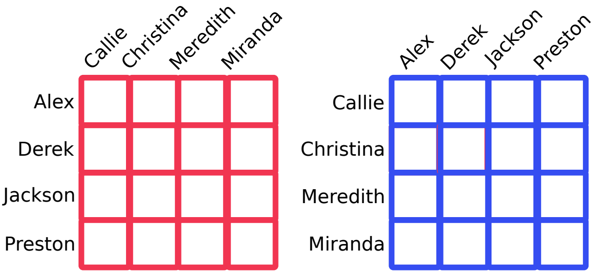
Getting it right since 1952.

NRMP ISSUES CALL FOR NOMINATIONS FOR BOARD OF DIRECTORS: The NRMP Board of Directors is seeking nominations for two open Director positions. Read about the **nomination process and the qualifications for nominees**.

RANKING NOW OPEN FOR 2016 MAIN RESIDENCY MATCH: The 2016 Main Residency Match ranking function opened in the Registration, Ranking, and Results system on Friday, January 15, at 12:00 p.m. Eastern Time. Final rank order lists must be certified before the **Rank Order List Deadline on Wednesday, February 24, at 9:00 p.m. Eastern Time**. Visit the toolkits for resources for assistance with the ranking process.

NRMP STATEMENT REGARDING A SINGLE MATCH: At its May 4, 2015 meeting, the National Resident Matching Program Board of Directors adopted a **statement** about whether a single Match will result from the single accreditation system for graduate medical education programs in the U.S. to be conducted under the aegis of the Accreditation Council for Graduate Medical Education.

Stable Matching Problem: Input



There are n men and n women.

Stable Matching Problem: Input

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

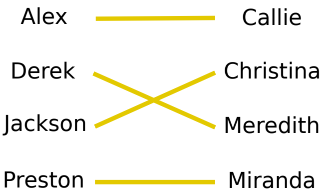
	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

Each man ranks all the women in order of preference.
Each woman ranks all the men in order of preference.
Each person uses all ranks from 1 to n, i.e., no ties, no incomplete lists.

Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	**1**	2	3	4
Derek	4	3	**1**	2
Jackson	4	**3**	1	2
Preston	3	1	4	**2**

	Alex	Derek	Jackson	Preston
Callie	**1**	2	3	4
Christina	4	1	**3**	2
Meredith	4	**1**	2	3
Miranda	3	1	2	**4**

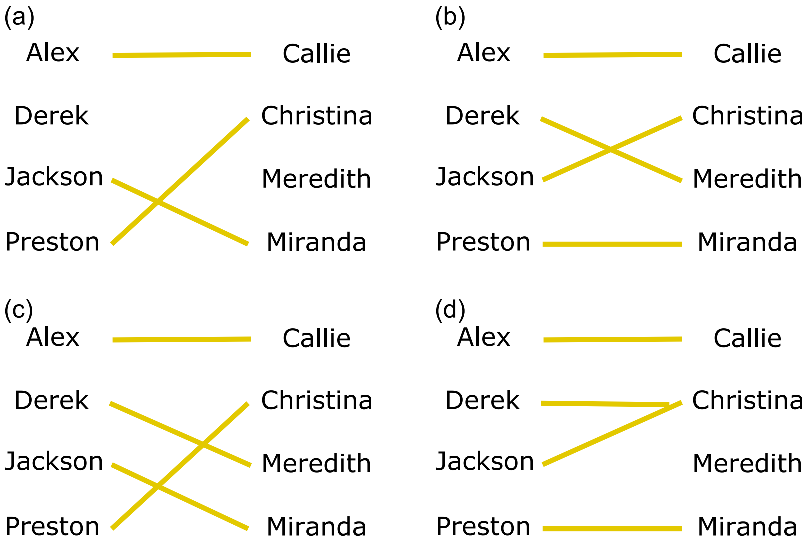


Matching: each man is paired with ≤ 1 one woman and vice versa.

Perfect matching: each man is paired with exactly one woman and vice versa.

"Perfect": only means one-one mapping, not that people are happy with matches.

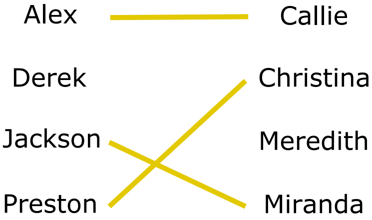
Examples



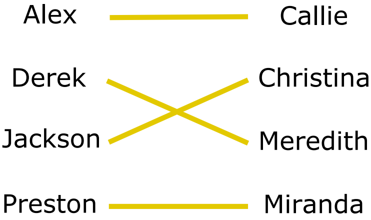
▸ Modules: Lecture 1: Stable Matching: Perfect or not

Examples

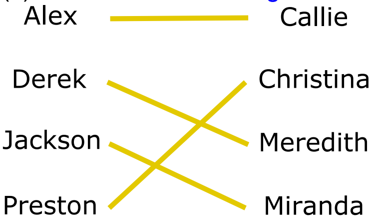
(a) Matching, not perfect



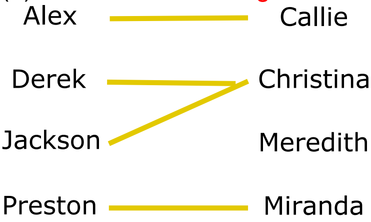
(b) Perfect matching



(c) Perfect matching



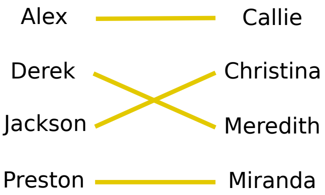
(d) Not a matching



Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

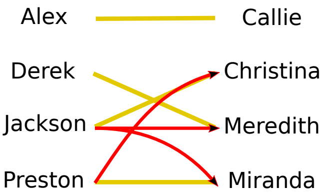


Are there problems with this matching?

Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

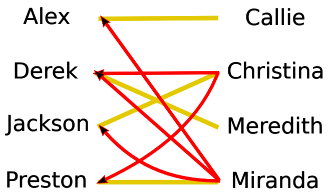


Are there problems with this matching?

Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

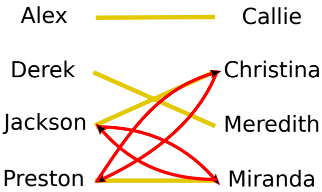


Are there problems with this matching?

Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

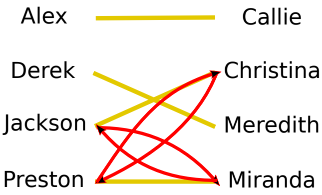


Rogue couple: a man and a woman who are not matched but prefer each other to their current partners.

Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

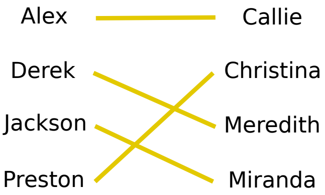


Stable matching: A perfect matching without any rogue couples.

Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	1	2	3	4
Derek	4	3	1	2
Jackson	4	3	1	2
Preston	3	1	4	2

	Alex	Derek	Jackson	Preston
Callie	1	2	3	4
Christina	4	1	3	2
Meredith	4	1	2	3
Miranda	3	1	2	4

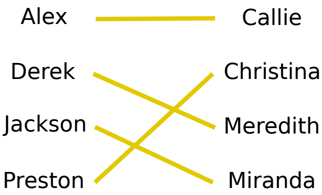


Stable matching: A perfect matching without any rogue couples.

Stable Matching Problem: Output

	Callie	Christina	Meredith	Miranda
Alex	**1**	2	3	4
Derek	4	3	**1**	2
Jackson	4	3	1	**2**
Preston	3	**1**	4	2

	Alex	Derek	Jackson	Preston
Callie	**1**	2	3	4
Christina	4	1	3	**2**
Meredith	4	**1**	2	3
Miranda	3	1	**2**	4



Questions

1. Given preferences for every woman and every man, does a stable matching exist?
2. If it does, can we compute it? How fast?

Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

Stable matching

Unrequited love

Example

	w1	w2
m1	1	2
m2	1	2

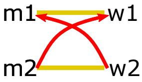
	m1	m2
w1	1	2
w2	1	2

Stable matching

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

Unrequited love



Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

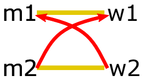
	m1	m2
w1	2	1
w2	2	1

Stable matching

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

Unrequited love



Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

Stable matching

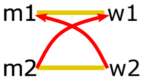
	w1	w2
m1	1	2
m2	1	2

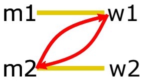
	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

Unrequited love





Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

Stable matching

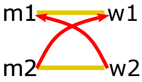
	w1	w2
m1	1	2
m2	1	2

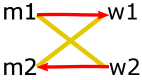
	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

Unrequited love





Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

	w1	w2
m1	1	2
m2	2	1

	m1	m2
w1	2	1
w2	1	2

Stable matching

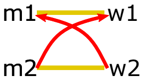
	w1	w2
m1	1	2
m2	1	2

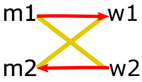
	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

Unrequited love





Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

	w1	w2
m1	1	2
m2	2	1

	m1	m2
w1	2	1
w2	1	2

Stable matching

	w1	w2
m1	**1**	2
m2	1	**2**

	m1	m2
w1	**1**	2
w2	1	**2**

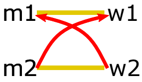
	w1	w2
m1	1	**2**
m2	**1**	2

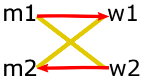
	m1	m2
w1	2	**1**
w2	**2**	1

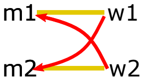
	w1	w2
m1	**1**	2
m2	2	**1**

	m1	m2
w1	**2**	1
w2	1	**2**

Unrequited love







Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

	w1	w2
m1	1	2
m2	2	1

	m1	m2
w1	2	1
w2	1	2

Stable matching

	w1	w2
m1	**1**	2
m2	1	**2**

	m1	m2
w1	**1**	2
w2	1	**2**

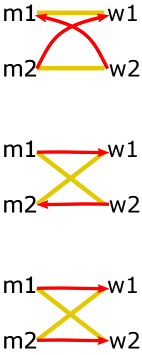
	w1	w2
m1	1	**2**
m2	**1**	2

	m1	m2
w1	2	**1**
w2	**2**	1

	w1	w2
m1	1	**2**
m2	**2**	1

	m1	m2
w1	2	**1**
w2	**1**	2

Unrequited love



Example

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	1	2
w2	1	2

	w1	w2
m1	1	2
m2	1	2

	m1	m2
w1	2	1
w2	2	1

	w1	w2
m1	1	2
m2	2	1

	m1	m2
w1	2	1
w2	1	2

Stable matching

	w1	w2
m1	**1**	2
m2	1	**2**

	m1	m2
w1	**1**	2
w2	1	**2**

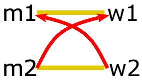
	w1	w2
m1	1	**2**
m2	**1**	2

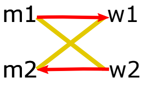
	m1	m2
w1	2	**1**
w2	**2**	1

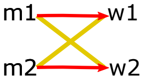
	w1	w2
m1	1	**2**
m2	**2**	1

	m1	m2
w1	2	**1**
w2	**1**	2

Unrequited love







Challenge: Can you create an example that does not have a stable matching?

▸ Modules: Lecture 1: Stable Matching: Existence

Gale-Shapley Algorithm

Each man proposes to each woman, in decreasing order of preference.
Woman accepts if she is free or prefers new prospect to current fiance.

```
Initially, all men and all women are free
Set S of matched pairs is empty
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed
    If w is free, then
         she becomes engaged to m Add (m, w) to S
    else if w is engaged to m' and she prefers m to m'
         she becomes engaged to m Add (m, w) to S
         m' becomes free Delete (m', w) from S
    Otherwise, m remains free
Return set S of engaged pairs
```

Gale-Shapley Algorithm

Each man proposes to each woman, in decreasing order of preference.
Woman accepts if she is free or prefers new prospect to current fiance.

```
Initially, all men and all women are free
Set S of matched pairs is empty
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed
    If w is free, then
         she becomes engaged to m Add (m, w) to S
    else if w is engaged to m' and she prefers m to m'
         she becomes engaged to m Add (m, w) to S
         m' becomes free Delete (m', w) from S
    Otherwise, m remains free
Return set S of engaged pairs
```

Modules: Lecture 1: Stable matching: Termination

Questions about the Algorithm

What can go wrong?

Questions about the Algorithm

What can go wrong?

- Does the algorithm even terminate?
- If it does, how long does the algorithm take to run?
- If it does, is S a perfect matching? A stable matching?

Some Simple Observations

- Does the Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa?

Some Simple Observations

- Does the Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa? Yes, no one will ever be paired with two or more people.

Some Simple Observations

- Does the Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa? Yes, no one will ever be paired with two or more people.
- Man's status:

Some Simple Observations

- Does the Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa? Yes, no one will ever be paired with two or more people.
- Man's status: Can alternate between being free and being engaged.
- Woman's status:

Some Simple Observations

- Does the Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa? Yes, no one will ever be paired with two or more people.
- Man's status: Can alternate between being free and being engaged.
- Woman's status: Remains engaged after first proposal.
- Ranking of a man's partner:

Some Simple Observations

- Does the Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa? Yes, no one will ever be paired with two or more people.
- Man's status: Can alternate between being free and being engaged.
- Woman's status: Remains engaged after first proposal.
- Ranking of a man's partner: Remains the same or goes down.
- Ranking of a woman's partner:

Some Simple Observations

- Does the Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa? Yes, no one will ever be paired with two or more people.
- Man's status: Can alternate between being free and being engaged.
- Woman's status: Remains engaged after first proposal.
- Ranking of a man's partner: Remains the same or goes down.
- Ranking of a woman's partner: Can never go down.

Proof: Algorithm Terminates

- Is there some quantity that we can use to measure the progress of the algorithm in each iteration?

Proof: Algorithm Terminates

- Is there some quantity that we can use to measure the progress of the algorithm in each iteration?
- Number of free men? Number of free women?

Proof: Algorithm Terminates

- Is there some quantity that we can use to measure the progress of the algorithm in each iteration?
- Number of free men? Number of free women? No, since both can remain unchanged in an iteration.

Proof: Algorithm Terminates

- Is there some quantity that we can use to measure the progress of the algorithm in each iteration?
- Number of free men? Number of free women? No, since both can remain unchanged in an iteration.
- Number of proposals made after k iterations?

Proof: Algorithm Terminates

- Is there some quantity that we can use to measure the progress of the algorithm in each iteration?
- Number of free men? Number of free women? No, since both can remain unchanged in an iteration.
- Number of proposals made after k iterations? Must increase by one in each iteration.
- How many total proposals can be made?

Proof: Algorithm Terminates

- Is there some quantity that we can use to measure the progress of the algorithm in each iteration?
- Number of free men? Number of free women? No, since both can remain unchanged in an iteration.
- Number of proposals made after k iterations? Must increase by one in each iteration.
- How many total proposals can be made? n^2. Therefore, the algorithm must terminate in n^2 iterations!

Proof: Matching Computed is Perfect

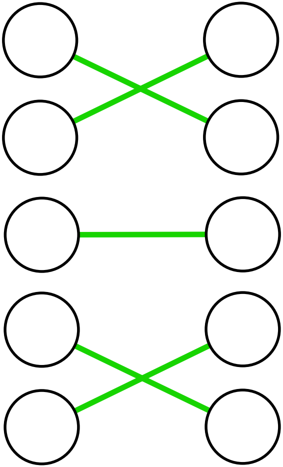
- Suppose the set S of pairs returned by the Gale-Shapley algorithm is not perfect.
- S is a matching. Therefore, there must be at least one free man m.
- m has proposed to all the women (since algorithm terminated).
- Therefore, each woman must be engaged (since she remains engaged after the first proposal to her).
- Therefore, all men must be engaged, contradicting the assumption that m is free.

Proof: Matching Computed is Perfect

- Suppose the set S of pairs returned by the Gale-Shapley algorithm is not perfect.
- S is a matching. Therefore, there must be at least one free man m.
- m has proposed to all the women (since algorithm terminated).
- Therefore, each woman must be engaged (since she remains engaged after the first proposal to her).
- Therefore, all men must be engaged, contradicting the assumption that m is free.
- Proof that matching is perfect relies on
  - proof that the algorithm terminated and
  - the very specific termination condition.

Proof: Matching Computed is Stable

Perfect matching S returned by algorithm

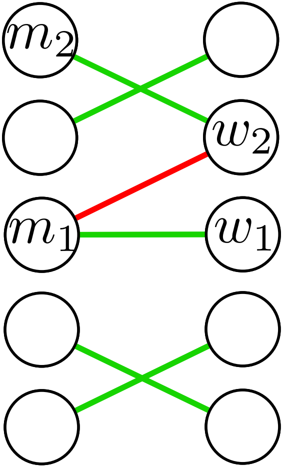


Proof: Matching Computed is Stable

Not stable: m_1 paired with w_1 but prefers w_2;

w_2 paired with m_2 but prefers m_1

▸ Modules: Lecture 1: Stable Matching: Rogue couple at termination

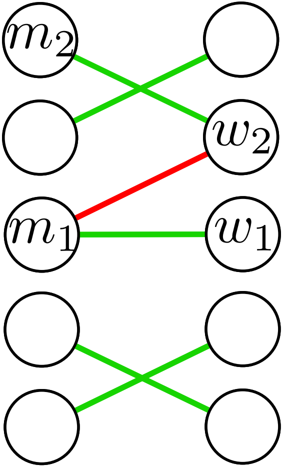


Proof: Matching Computed is Stable

Not stable: m_1 paired with w_1 but prefers w_2;

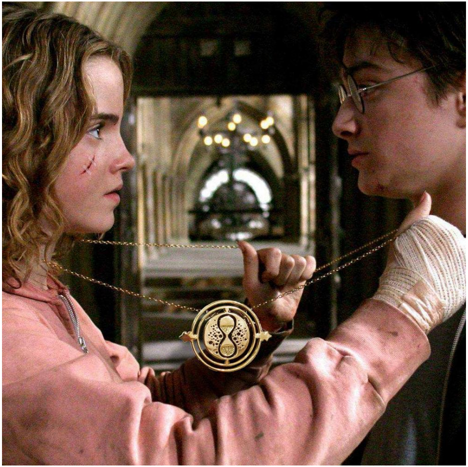
w_2 paired with m_2 but prefers m_1

$\Rightarrow$ m_1 proposed to w_2 before proposing to w_1



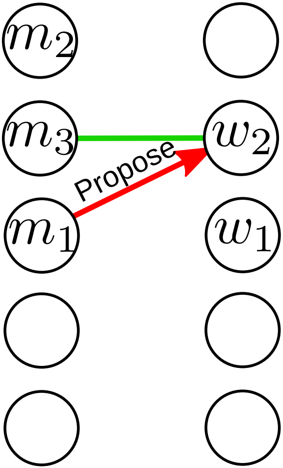
Proof: Matching Computed is Stable

Proof: Matching Computed is Stable



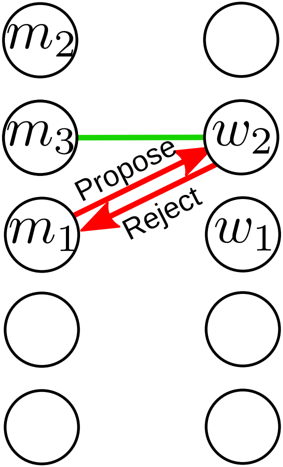
Proof: Matching Computed is Stable

Rewind: What happened when m_1 proposed to w_2?



Proof: Matching Computed is Stable

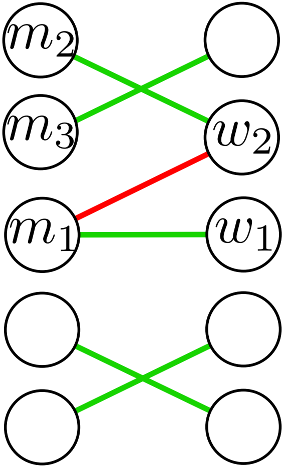
Case 1: w_2 rejected m_1 because she preferred current partner m_3.



Proof: Matching Computed is Stable

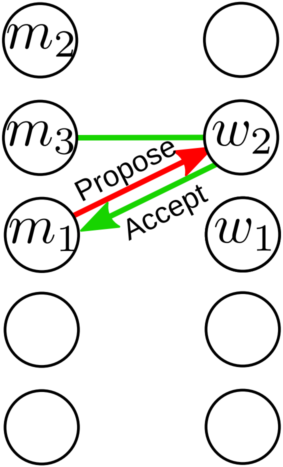
Case 1: At termination, w_2 must prefer her final partner m_2 to m_3.

Contradicts consequence of instability: w_2 prefers m_1 to m_2



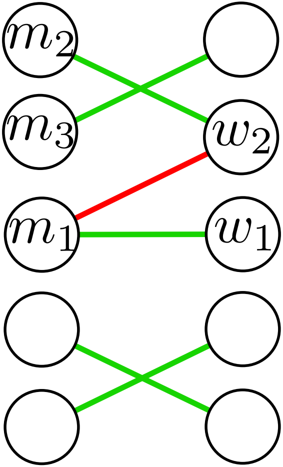
Proof: Matching Computed is Stable

Case 2: w_2 accepted m_1 because she had no partner or preferred m_1 to current partner m_3.



Proof: Matching Computed is Stable

Case 2: By instability, we know w_2 prefers m_1 to m_2. But at termination, w_2 is matched with m_2, which contradicts property that a woman switches only to a better match.



Proof: Stable Matching (in Words)

- Suppose S is not stable, i.e., there are two pairs (m_1, w_1) and (m_2, w_2) in S such that m_1 prefers w_2 to w_1 and w_2 prefers m_1 to m_2.
- m_1 must have proposed to w_2 before w_1 because ....
- At that stage w_2 must have rejected m_1; otherwise, the algorithm would pair m_1 and w_2, which would prevent the pairing of m_2 and w_2 in a later iteration of the algorithm. (Why?)
- When w_2 rejected m_1, she must have been paired with some man, say m_3, whom she prefers to m_1.
- Since m_2 is paired with w_2 at termination, w_2 must prefer to m_2 to m_3 or $m_2 = m_3$ (Why?), which contradicts our conclusion (from instability) that w_2 prefers m_1 to m_2.

Running Time

Implement each iteration in constant time to get running time $\propto n^2$

```
Initially, all men and all women are free
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed
    If w is free, then
         she becomes engaged to m
    else if w is engaged to m' and
         she becomes engaged to m
         m' becomes free
    Otherwise, m remains free
Return set S of engaged pairs
```

Running Time

Implement each iteration in constant time to get running time $\propto n^2$

```
Initially, all men and all women are free
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed
    If w is free, then
         she becomes engaged to m
    else if w is engaged to m' and
         she becomes engaged to m
         m' becomes free
    Otherwise, m remains free
Return set S of engaged pairs
```

Running Time

Implement each iteration in constant time to get running time $\propto n^2$

```
Initially, all men and all women are free
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m Linked list
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed
    If w is free, then
         she becomes engaged to m
    else if w is engaged to m' and
         she becomes engaged to m
         m' becomes free
    Otherwise, m remains free
Return set S of engaged pairs
```

Running Time

Implement each iteration in constant time to get running time $\propto n^2$

```
Initially, all men and all women are free
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m Linked list
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed
    If w is free, then
         she becomes engaged to m
    else if w is engaged to m′ and
         she becomes engaged to m
         m′ becomes free
    Otherwise, m remains free
Return set S of engaged pairs
```

Running Time

Implement each iteration in constant time to get running time $\propto n^2$

```
Initially, all men and all women are free
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m Linked list
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed Next[m] = index of next
    If w is free, then                    woman m can propose to
         she becomes engaged to m
    else if w is engaged to m' and
         she becomes engaged to m
         m' becomes free
    Otherwise, m remains free
Return set S of engaged pairs
```

Running Time

Implement each iteration in constant time to get running time $\propto n^2$

```
Initially, all men and all women are free
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m Linked list
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed Next[m] = index of next
    If w is free, then                  woman m can propose to
         she becomes engaged to m
    else if w is engaged to m' and she prefers m to m'
         she becomes engaged to m
         m' becomes free
    Otherwise, m remains free
Return set S of engaged pairs
```

Running Time

Implement each iteration in constant time to get running time $\propto n^2$

```
Initially, all men and all women are free
While there is at least one free man who has not
   proposed to every woman
    Choose such a man m Linked list
    m proposes to the highest-ranked woman w on his list
    to whom he has not yet proposed Next[m] = index of next
    If w is free, then                       woman m can propose to
         she becomes engaged to m
    else if w is engaged to m′ and she prefers m to m′
         she becomes engaged to m Rank[w,m] = rank of m in
         m′ becomes free                          w's list
    Otherwise, m remains free
Return set S of engaged pairs
```

Further Reading and Viewing

- Gail-Shapley algorithm always produces the same matching in which each man is paired with his best valid partner but each woman is paired with her worst valid partner. Read pages 9–12 of the textbook.
- Video describing matching algorithm used by the National Resident Matching Program
- Description of research to Roth and Shapley that led to 2012 Nobel Prize in Economics

Variants of Stable Matching

- Hospitals and residents: Each hospital can take multiple residents.
- Hospitals and residents with couples: Each hospital can take multiple residents. A couple must be assigned together, either to the same hospital or to a specific pair of hospitals chosen by the couple.
- Stable roommates problem: there is only one pool of people.
- Preferences may be incomplete or have ties or people may lie.

Variants of Stable Matching

- Hospitals and residents: Each hospital can take multiple residents. *Modification of Gale-Shapley algorithm works. Some residents may not be matched. Some hospitals may not fill quota.*
- Hospitals and residents with couples: Each hospital can take multiple residents. A couple must be assigned together, either to the same hospital or to a specific pair of hospitals chosen by the couple.
- Stable roommates problem: there is only one pool of people.
- Preferences may be incomplete or have ties or people may lie.

Variants of Stable Matching

- Hospitals and residents: Each hospital can take multiple residents. *Modification of Gale-Shapley algorithm works. Some residents may not be matched. Some hospitals may not fill quota.*
- Hospitals and residents with couples: Each hospital can take multiple residents. A couple must be assigned together, either to the same hospital or to a specific pair of hospitals chosen by the couple. *NP-complete*
- Stable roommates problem: there is only one pool of people.
- Preferences may be incomplete or have ties or people may lie.

Variants of Stable Matching

- Hospitals and residents: Each hospital can take multiple residents. *Modification of Gale-Shapley algorithm works. Some residents may not be matched. Some hospitals may not fill quota.*
- Hospitals and residents with couples: Each hospital can take multiple residents. A couple must be assigned together, either to the same hospital or to a specific pair of hospitals chosen by the couple. *NP-complete*
- Stable roommates problem: there is only one pool of people. *Stable matching may not exist. Irving's algorithm; more complex than Gale-Shapley.*
- Preferences may be incomplete or have ties or people may lie.

Variants of Stable Matching

- Hospitals and residents: Each hospital can take multiple residents. *Modification of Gale-Shapley algorithm works. Some residents may not be matched. Some hospitals may not fill quota.*
- Hospitals and residents with couples: Each hospital can take multiple residents. A couple must be assigned together, either to the same hospital or to a specific pair of hospitals chosen by the couple. *NP-complete*
- Stable roommates problem: there is only one pool of people. *Stable matching may not exist. Irving's algorithm; more complex than Gale-Shapley.*
- Preferences may be incomplete or have ties or people may lie. *Several variants are NP-hard, even to approximate.*