Please wear a mask at all times in class!

Be committed. Be well.
Introduction to CS 4104

T. M. Murali

August 24, 2021
Course Information

Instructor

- T. M. Murali, murali@cs.vt.edu
- Office Hours: 4pm–5:30pm, Tuesdays and Thursdays, Zoom (link on Canvas)
Course Information

- **Instructor**
  - T. M. Murali, murali@cs.vt.edu
  - Office Hours: 4pm–5:30pm, Tuesdays and Thursdays, Zoom (link on Canvas)

- **Class meeting times**
  - TR 9:30am–10:45am, NCB 250
Keeping in Touch

- Piazza: [https://piazza.com/class/krkusfyxgba6da](https://piazza.com/class/krkusfyxgba6da) announcements, including homeworks and exams.
- Canvas: homework/exam submissions and solutions, grades
- TopHat: [tophat.com](http://tophat.com), join code 756658, for in-class polls.
About the Course

Data and Algorithm Analysis

Stable Matching

Required Course Textbook

- Algorithm Design
- Jon Kleinberg and Éva Tardos
- Addison-Wesley
- 2006
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).
- Course 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
  - Randomized algorithms (if there is time)
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 the internet or anyone else is a violation of the Honor Code.

- 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?

Dictionary

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.
What is an Algorithm?


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


**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.

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.


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.


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.
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.
About the Course Data and Algorithm Analysis Stable Matching

T. M. Murali August 24, 2021 Introduction to CS 4104
The Match offers a fair and transparent process because all participants follow the same rules and deadlines.

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

There are $n$ men and $n$ women.
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

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

(a) Alex — Callie
   Derek — Christina
   Jackson — Meredith
   Preston — Miranda

(b) Alex — Callie
   Derek — Christina
   Jackson — Meredith
   Preston — Miranda

(c) Alex — Callie
   Derek — Christina
   Jackson — Meredith
   Preston — Miranda

(d) Alex — Callie
   Derek — Christina
   Jackson — Meredith
   Preston — Miranda
Examples

(a) Matching, not perfect
Alex — Callie
Derek — Christina
Jackson — Meredith
Preston — Miranda

(b) Perfect matching
Alex — Callie
Derek — Christina
Jackson — Meredith
Preston — Miranda

(c) Perfect matching
Alex — Callie
Derek — Christina
Jackson — Meredith
Preston — Miranda

(d) Not a matching
Alex — Callie
Derek — Christina
Jackson — Meredith
Preston — Miranda
Stable Matching Problem

Are there problems with this matching?
Stable Matching Problem

Are there problems with this matching?
Stable Matching Problem

Are there problems with this matching?


Rogue couple: a man and a woman who are not matched but prefer each other to their current partners.
Stable matching: A perfect matching without any rogue couples.
Stable Matching Problem

Stable matching: A perfect matching without any rogue couples.
# Stable Matching Problem

Given preferences for every person, does a stable matching exist? If it does, can we compute it? How fast?

<table>
<thead>
<tr>
<th></th>
<th>Alex</th>
<th>Derek</th>
<th>Jackson</th>
<th>Preston</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callie</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Christina</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Meredith</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Miranda</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Alex</th>
<th>Derek</th>
<th>Jackson</th>
<th>Preston</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callie</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Christina</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Meredith</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Miranda</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

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?
Stable Matching

Example

<table>
<thead>
<tr>
<th>m1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Stable matching

<table>
<thead>
<tr>
<th>m1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Unrequited love

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

T. M. Murali August 24, 2021 Introduction to CS 4104
### Example

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>m2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Stable matching

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>m2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Unrequited love

- m1 \(\rightarrow\) w1
- m2 \(\rightarrow\) w2

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

Data and Algorithm Analysis

Stable Matching

**Example**

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1 2</td>
</tr>
<tr>
<td>m2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>2 1</td>
</tr>
<tr>
<td>m2</td>
<td>2 1</td>
</tr>
</tbody>
</table>

**Stable matching**

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1 2</td>
</tr>
<tr>
<td>m2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

**Unrequited love**

m1 \(\xrightarrow{w1}\) w1
m2 \(\xrightarrow{w2}\) w2

T. M. Murali August 24, 2021

Introduction to CS 4104
Stable Matching

Example

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>w2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>w2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Stable matching

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>w2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>w2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Unrequited love

m1 → w1
m2 → w2

T. M. Murali August 24, 2021 Introduction to CS 4104
Example

Stable matching

Unrequited love

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

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>m1</strong></td>
<td>1 2</td>
<td>2 1</td>
</tr>
<tr>
<td><strong>m2</strong></td>
<td>1 2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>w1</strong></td>
<td>1 2</td>
<td>2 1</td>
</tr>
<tr>
<td><strong>w2</strong></td>
<td>1 2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

### Stable matching

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>m1</strong></td>
<td>1 2</td>
<td>2 1</td>
</tr>
<tr>
<td><strong>m2</strong></td>
<td>1 2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>w1</strong></td>
<td>1 2</td>
<td>1 2</td>
</tr>
<tr>
<td><strong>w2</strong></td>
<td>1 2</td>
<td>2 1</td>
</tr>
</tbody>
</table>

### Unrequited love

- **m1** → w1
- **m2** → w2
- **w1** → **m1**
- **w2** → **m2**
### Example

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>m2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>m2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

### Stable matching

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>m2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>m2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>m2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

### Unrequited love

- m1 \(\rightarrow\) w1
- m2 \(\rightarrow\) w2
- m1 \(\rightarrow\) w1
- m2 \(\rightarrow\) w2
- m1 \(\rightarrow\) w1
- m2 \(\rightarrow\) w2
### Example

<table>
<thead>
<tr>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>w2</td>
</tr>
<tr>
<td>1 2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

### Stable matching

<table>
<thead>
<tr>
<th>m1</th>
<th>m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>w2</td>
</tr>
<tr>
<td>1 2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>1 2</td>
</tr>
</tbody>
</table>

### Unrequited love

- m1 → w1
- m2 → w2

T. M. Murali
August 24, 2021

Introduction to CS 4104
Challenge: Can you create an example that does not have a stable matching?
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
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

- Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa.
Some Simple Observations

- Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa.
- Man’s status:
Some Simple Observations

- Gale-Shapley algorithm computes a matching, i.e., each woman paired with at most one man and vice versa.
- Man’s status: Can alternate between being free and being engaged.
- Woman’s status:
Some Simple Observations

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

- 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

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

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

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: 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?

- 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: Algorithm Terminates

- Is there some quantity that we can use the 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: 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 the 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 \( m_1 \)
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 $w_2$?

![Diagram showing the proposed connections between men and women with arrows indicating proposed connections.](image-url)
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 $m_3$. 

\[ \begin{array}{c}
\text{Propose} \\
\text{Accept}
\end{array} \]
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 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 woman \( m \) can propose to

If \( w \) is free, then

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 $\text{Next}[m] = \text{index of next}$

If $w$ is free, then she becomes engaged to $m$

else if $w$ is engaged to $m'$ and she prefers $m$ to $m'$ she becomes engaged to $m$ $\text{Rank}[w,m] = \text{rank of } m \text{ in } w$’s list $m'$ becomes free

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.*