Introduction to CS 4104

T. M. Murali

January 19, 2016

Course Information

- Instructor
  - T. M. Murali, 2160B Torgerson, 231-8534, murali@cs.vt.edu
  - Office Hours: 9:30am-11:30am, Mondays and by appointment
- Graduate teaching assistants
  - Ahmed Attia, attia@vt.edu
  - Office Hours: McBryde 106, 12pm-1:30pm, Tuesdays, starting January 26, 2016
  - Mahesh Narayanamurthi, maheshnm@vt.edu
  - Office Hours: McBryde 106, 12pm-1:30pm, Thursdays, starting January 28, 2016

Course Information

- Instructor
  - T. M. Murali, 2160B Torgerson, 231-8534, murali@cs.vt.edu
  - Office Hours: 9:30am-11:30am, Mondays and by appointment
- Graduate teaching assistants
  - Ahmed Attia, attia@vt.edu
  - Office Hours: McBryde 106, 12pm-1:30pm, Tuesdays, starting January 26, 2016
  - Mahesh Narayanamurthi, maheshnm@vt.edu
  - Office Hours: McBryde 106, 12pm-1:30pm, Thursdays, starting January 28, 2016
- Class meeting time
  - TR 2pm–3:15pm, GYM 124

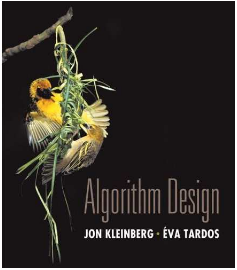
Course Information

- Instructor
  - T. M. Murali, 2160B Torgerson, 231-8534, murali@cs.vt.edu
  - Office Hours: 9:30am-11:30am, Mondays and by appointment
- Graduate teaching assistants
  - Ahmed Attia, attia@vt.edu
  - Office Hours: McBryde 106, 12pm-1:30pm, Tuesdays, starting January 26, 2016
  - Mahesh Narayanamurthi, maheshnm@vt.edu
  - Office Hours: McBryde 106, 12pm-1:30pm, Thursdays, starting January 28, 2016
- Class meeting time
  - TR 2pm–3:15pm, GYM 124
- Prerequisite: Grade of C or better in CS 3114; P or better in MATH 3034 or MATH 3134
- Force-add: Visit `https://www.cs.vt.edu/S16Force-Adds`

Keeping in Touch

- Course web site
  `http://courses.cs.vt.edu/~cs4104/murali/spring2016`,
  updated regularly through the semester
- Canvas: grades and homework/exam solutions
- Piazza: announcements

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).
- Course roughly follows the topics suggested in textbook
  - Stable matching
  - Measures of algorithm complexity
  - Graphs
  - Greedy algorithms
  - Divide and conquer
  - 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 extremely important.

Lecture Slides

- Will be available on class web site.
- Usually posted just before class.
- Class attendance is extremely important. Lecture in class contains significant and substantial additions to material on the slides.

Homeworks

- Posted on the web site $\approx$ one week before due date.
- Announced on the class mailing list.

Homeworks

- Posted on the web site $\approx$ one week before due date.
- Announced on the class mailing list.
- Work on the homework in pairs but prepare solutions individually. *TAs will check for identical or very similar solutions.*
- Prepare solutions digitally and upload on Canvas.

Homeworks

- Posted on the web site $\approx$ one week before due date.
- Announced on the class mailing list.
- Work on the homework in pairs but prepare solutions individually. *TAs will check for identical or very similar solutions.*
- 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 on the class mailing list.
- Work on the homework in pairs but prepare solutions individually. *TAs will check for identical or very similar solutions.*
- 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: ≈ 8, 60% of the grade.
- Take-home midterm: 15% of the grade.
- Take-home final: 25% of the grade.

What is an Algorithm?

What is an Algorithm?

Chamber's
: 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?

Chamber's 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 rigourously 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.

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.
3. From the Greek *algos* (meaning "pain," also a root of "analgesic") and *rythmos* (meaning "flow," also a root of "rhythm").

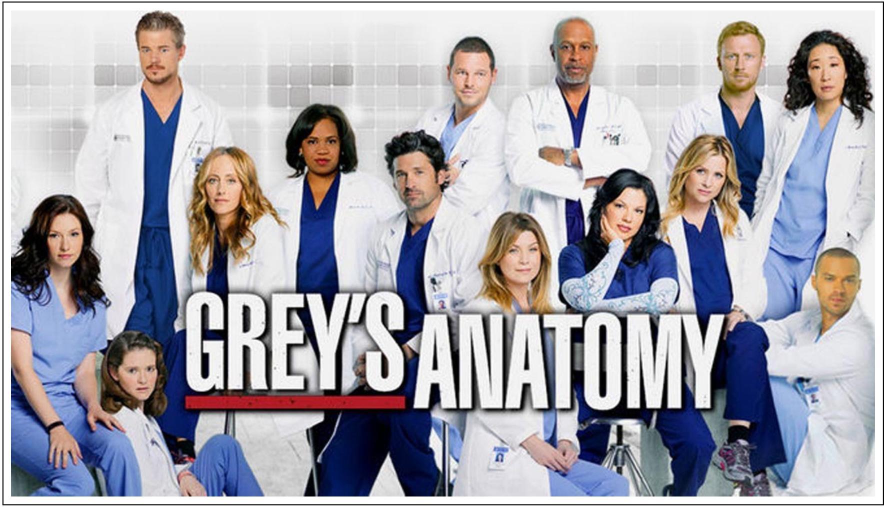
Origin of the word "Algorithm"

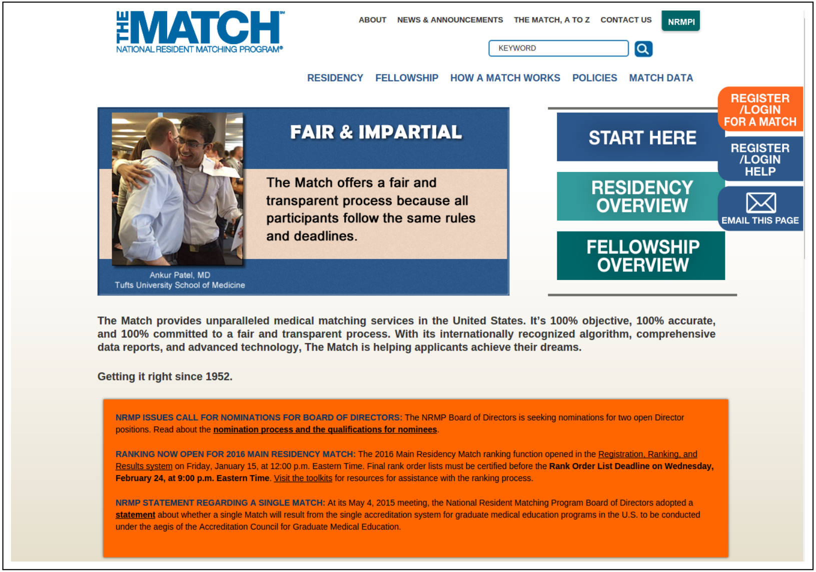
1. From the Arabic *al-Khwarizmi*, a native of Khwarazm, a name for the 9th century mathematician, Abu Ja'far Mohammed ben Musa.

2. From Al Gore, the former U.S. vice-president who invented the internet.
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 endorsement.

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 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
REGISTER /LOGIN FOR A MATCH
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
REGISTER /LOGIN HELP
RESIDENCY OVERVIEW
EMAIL THIS PAGE
FELLOWSHIP OVERVIEW
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.
- *Perfect Matching*: each man is paired with exactly one woman and vice-versa.
- *Rogue Couple*: a man and a woman who are not matched but prefer each other to their current partners.
- Goal: *Stable Matching*: a perfect matching without any rogue couples.

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.
- *Perfect Matching*: each man is paired with exactly one woman and vice-versa.
- *Rogue Couple*: a man and a woman who are not matched but prefer each other to their current partners.
- Goal: *Stable Matching*: a perfect matching without any rogue couples.
- Given preferences for every woman and every man, does a stable matching exist?
- If it does, can we compute it? How fast?

Examples

Example

$$\frac{\begin{array}{c} m \text{ prefers } w \text{ to } w' \\ m' \text{ prefers } w \text{ to } w' \end{array}}{\begin{array}{c} w \text{ prefers } m \text{ to } m' \\ w' \text{ prefers } m \text{ to } m' \end{array}}$$

Stable Matching

Examples

Example

m prefers w to w'
m' prefers w to w'

---

w prefers m to m'
w' prefers m to m'

Stable Matching

(m, w) and (m', w')

Examples

Example

$$\frac{\begin{array}{c} m \text{ prefers } w \text{ to } w' \\ m' \text{ prefers } w \text{ to } w' \end{array}}{\begin{array}{c} w \text{ prefers } m \text{ to } m' \\ w' \text{ prefers } m \text{ to } m' \end{array}}$$

$$\frac{\begin{array}{c} m \text{ prefers } w \text{ to } w' \\ m' \text{ prefers } w \text{ to } w' \end{array}}{\begin{array}{c} w \text{ prefers } m' \text{ to } m \\ w' \text{ prefers } m' \text{ to } m \end{array}}$$

Stable Matching

(m, w) and (m', w')

Examples

Example	**Stable Matching**
$\dfrac{m \text{ prefers } w \text{ to } w' \quad m' \text{ prefers } w \text{ to } w'}{w \text{ prefers } m \text{ to } m' \quad w' \text{ prefers } m \text{ to } m'}$	(m, w) and (m', w')
$\dfrac{m \text{ prefers } w \text{ to } w' \quad m' \text{ prefers } w \text{ to } w'}{w \text{ prefers } m' \text{ to } m \quad w' \text{ prefers } m' \text{ to } m}$	(m, w') and (m', w)

Examples

Example	Stable Matching
m prefers w to w' m' prefers w to w' ——— w prefers m to m' w' prefers m to m'	(m, w) and (m', w')
m prefers w to w' m' prefers w to w' ——— w prefers m' to m w' prefers m' to m	(m, w') and (m', w)
m prefers w to w' m' prefers w' to w ——— w prefers m' to m w' prefers m to m'	

Examples

Example	Stable Matching
m prefers w to w' m' prefers w to w' ——— w prefers m to m' w' prefers m to m'	(m, w) and (m', w')
m prefers w to w' m' prefers w to w' ——— w prefers m' to m w' prefers m' to m	(m, w') and (m', w)
m prefers w to w' m' prefers w' to w ——— w prefers m' to m w' prefers m to m'	(m, w) and (m', w') or (m, w') and (m', w)

Gale-Shapley Algorithm

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

```
Gale-Shapley Algorithm:
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 prefers m to m'
         she becomes engaged to m
         m' becomes free
    Otherwise, m remains free
EndWhile
```

Questions about the Algorithm

- 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

- Man's status:

Some Simple Observations

- Man's status: Can alternate between being free and being engaged.
- Woman's status:

Some Simple Observations

- 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

- 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

- 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: Termination

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

Proof: Termination

- 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?

Proof: Termination

- 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.

Proof: Termination

- 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?

Proof: Termination

- 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?

Proof: Termination

- 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: Termination

- 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!

Formal proof: Let $p(k), k \geq 1$ be the number of proposals made after k iterations. Clearly, $p(k) \leq n^2$ since there are n^2 man-woman pairs. Moreover, at least one proposal is made in every iteration. Hence, the algorithm terminates after n^2 iterations.

Running Time

Impelement 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
EndWhile
Return set S of engaged pairs
```

Running Time

Impelement 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
EndWhile
Return set S of engaged pairs
```

Running Time

Impelement 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
EndWhile
Return set S of engaged pairs
```

Running Time

Impelement 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
EndWhile
Return set S of engaged pairs
```

Running Time

Impelement 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
EndWhile
Return set S of engaged pairs
```

Running Time

Impelement 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
EndWhile
Return set S of engaged pairs
```

Running Time

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

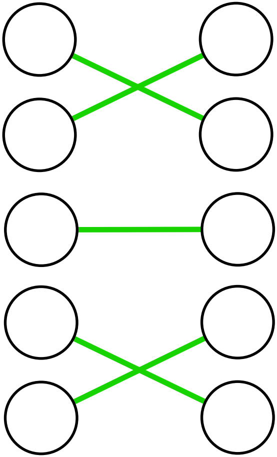
Initially, all men and all women are free
While there is at least one free man who has not
  proposed to every woman
  Choose such a man m Linked list
  m proposes to the highest-ranked woman w on his list
  to whom he has not yet proposed Next[m] = index of next
  If w is free, then woman m can propose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

Proof: Perfect Matching

- 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.

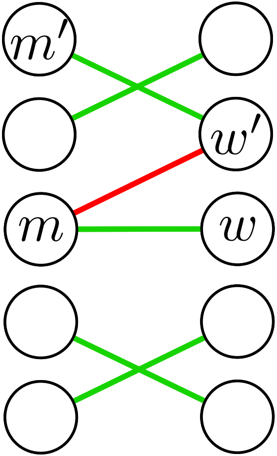
Proof: Stable Matching

Perfect matching S returned by algorithm



Proof: Stable Matching

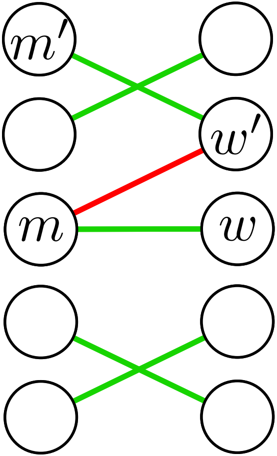
Not stable: m prefers w' to w; w prefers m' to m



Proof: Stable Matching

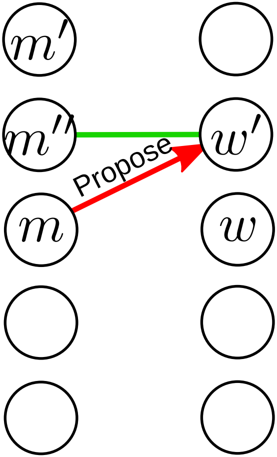
Not stable: *m* prefers w' to *w*; *w* prefers m' to *m*

$\Rightarrow$ *m* proposes to w' before *w*



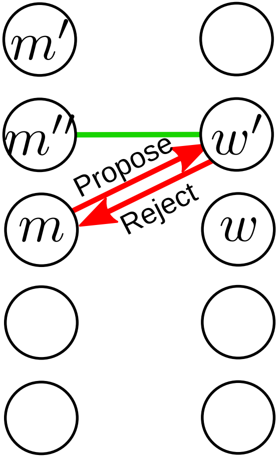
Proof: Stable Matching

What happened when m proposed to w'?



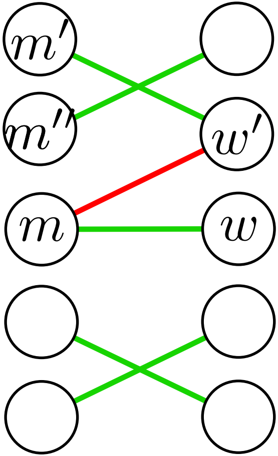
Proof: Stable Matching

Case 1: w' rejected m because she preferred current partner m''.



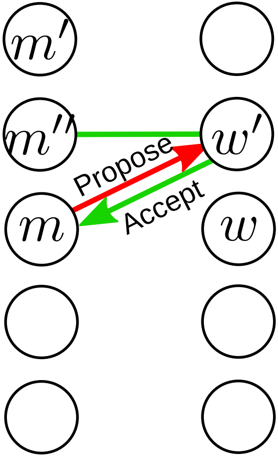
Proof: Stable Matching

Case 1: At termination, w' must like her final partner m' at least as much as m''. Contradicts instability: w prefers m to m'



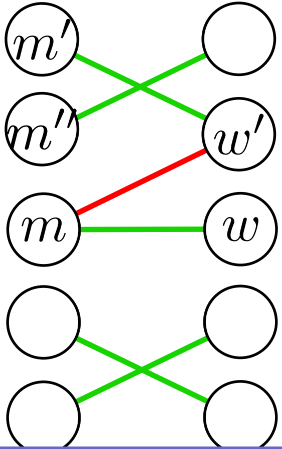
Proof: Stable Matching

Case 2: w' accepted m because she had no partner or preferred m to current partner m''.



Proof: Stable Matching

Case 2: By instability, we know w prefers m to m'. But at termination, w' is matched with m', 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, w) and (m', w') in S such that m prefers w' to w and w' prefers m to m'.
- m must have proposed to w' before w.
- At that stage w' must have rejected m; otherwise, (m, w') would be paired preventing formation of (m', w') later.
- When w' rejected m, she must have been paired with m'' whom she prefers to m.
- Since (m', w') is in S, w' must prefer to m' to m'' or $m' = m''$, which contradicts our assumption that w' prefers m to m'.

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 "gender".
- 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 "gender".
- 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 "gender".
- 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 "gender". 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 "gender". 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.