CS 473ug: Algorithms

Chandra Chekuri
chekuri@cs.uiuc.edu
3228 Siebel Center

University of Illinois, Urbana-Champaign

Fall 2007

Part I

Administrivia

Instructional Staff

- Instructor: Chandra Chekuri (`chekuri`)
  - Office Hours: 11–12am on Monday, 1-2pm Tuesday, and by appointment.
- Teaching Assistants: Charles Blatti (`blatti`), Tracy Graumann (`tgrauman2`), Chia-chi Lin (`lin36`)
  - Office Hours: To be determined; see course webpage

Electronic Bulletin Boards

- Webpage: `www.cs.uiuc.edu/class/fa07/cs473ug`
- Newsgroup: `uiuc.class.cs473ug`

Textbooks

- Prerequisites: All material in CS 173, CS 225 and CS 273
- Text-book: Algorithm Design by Jon Kleinberg and Éva Tardos
- Lecture Notes: Available on the web-page after every class
- Additional References
  - Introduction to Algorithms: Cormen, Leiserson, Rivest, Stein
  - Algorithms: Dasgupta, Papadimitriou, and Vazirani.
  - Computers and Intractibility: Garey and Johnson
  - Previous class notes of Jeff Erickson, Sariel Har-Peled and Mahesh Viswanathan.

Grading Policy: Overview

Total Grade and Weight

- Homeworks: 25%
- Midterms: 40% (2×20)
- Finals: 35%

Additional Component

- Extra Credit: "Head Banging Sessions"
- Extra Credit: "Special problems on homeworks"

Homeworks

- One homework every week: Assigned on Friday and due the following Friday.
- Homeworks can be worked on in groups of up to 3 and each group submits *one* written solution (except Homework 0).
- Groups can be changed a *few* times in the course of the semester.
- Homeworks will be "turned in" orally every third week; the rest of the times you will turn in a written homework
  - Oral: Explain (verbally) to TA the solution to problems that are asked for.
  - Written: Write solutions to every problem and turn in the written solutions.

Head Banging Sessions

- 1 hour sessions, led by TAs, where you solve problems in groups
- Attendence optional, but (small) extra credit for those who attend
- When and where?

Head Banging Sessions

- 1 hour sessions, led by TAs, where you solve problems in groups
- Attendence optional, but (small) extra credit for those who attend
- When and where? Fill Survey! Details soon.

User Manual to Performing Badly

- Skip lectures
- Skip homework or copy homework solutions
- Skip reading assignments
- Fail to clarify doubts immediately
- Study right before exams

Part II

Course Goals and Overview

Topics

- Some useful basic algorithms and data structures
- Broadly applicable techniques in algorithm design
  - Greedy methods
  - Divide and Conquer
  - Dynamic Programming
  - Randomization and Approximation
- Analysis techniques
  - Recurrences
  - Amortization and elementary potential functions
- Reductions, NP-Completeness, Heuristics

Goals

- Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...)
- Algorithmic thinking.
- Understand/appreciate limits of computation.
- Learn/remember some basic tricks, algorithms, problems.

Goals

- Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...)
- Algorithmic thinking.
- Understand/appreciate limits of computation.
- Learn/remember some basic tricks, algorithms, problems.
- Have fun!!!

Part III

Primality Testing

Problem

Given an integer $n > 0$, is n a prime?

Problem

Given an integer $n > 0$, is n a prime?

SimpleAlgorithm:

```
for i = 2 to ⌊√n⌋ do
    if i divides n then
        return ‘‘COMPOSITE’’
endfor
return ‘‘PRIME’’
```

Problem

Given an integer $n > 0$, is n a prime?

SimpleAlgorithm:

```
for i = 2 to ⌊√n⌋ do
    if i divides n then
        return ‘‘COMPOSITE’’
endfor
return ‘‘PRIME’’
```

Correctness?

Problem

Given an integer $n > 0$, is n a prime?

SimpleAlgorithm:

```
for i = 2 to ⌊√n⌋ do
    if i divides n then
        return ‘‘COMPOSITE’’
endfor
return ‘‘PRIME’’
```

Correctness? If n is composite, at least one factor in $\{2, \ldots, \sqrt{n}\}$

Problem

Given an integer $n > 0$, is n a prime?

SimpleAlgorithm:

```
for i = 2 to ⌊√n⌋ do
    if i divides n then
        return ''COMPOSITE''
endfor
return ''PRIME''
```

Correctness? If n is composite, at least one factor in $\{2, \ldots, \sqrt{n}\}$
Running time?

Problem

Given an integer $n > 0$, is n a prime?

SimpleAlgorithm:

```
for i = 2 to ⌊√n⌋ do
    if i divides n then
        return ''COMPOSITE''
endfor
return ''PRIME''
```

Correctness? If n is composite, at least one factor in $\{2, \ldots, \sqrt{n}\}$
Running time? $O(\sqrt{n})$ divisions? Sub-linear in input size!

Problem

Given an integer $n > 0$, is n a prime?

SimpleAlgorithm:

```
for i = 2 to ⌊√n⌋ do
    if i divides n then
        return ''COMPOSITE''
endfor
return ''PRIME''
```

Correctness? If n is composite, at least one factor in $\{2, \ldots, \sqrt{n}\}$
Running time? $O(\sqrt{n})$ divisions? Sub-linear in input size! Wrong!

How many bits does it take to represent n in decimal or binary?
$O(\log n)$ bits.

Simple Algorithm takes *exponential time* in the input size.
Modern cryptography deals with binary numbers with 128, 256, 512 bits.
Simple Algorithm will take 2^{64}, 2^{128}, 2^{256} steps!
Fastest computer today: about 3 petaFlops $\simeq 3 \times 2^{50}$ ops/sec.

How many bits does it take to represent n in decimal or binary?
$O(\log n)$ bits.

Simple Algorithm takes *exponential time* in the input size.
Modern cryptography deals with binary numbers with 128, 256, 512 bits.
Simple Algorithm will take 2^{64}, 2^{128}, 2^{256} steps!
Fastest computer today: about 3 petaFlops $\simeq 3 \times 2^{50}$ ops/sec.

Lesson

Representation size very important, especially in *number* problems.

So, is there an *efficient/good/effective* algorithm for primality?

So, is there an *efficient/good/effective* algorithm for primality?

Question

What does efficiency mean?

So, is there an *efficient/good/effective* algorithm for primality?

Question

What does efficiency mean?

In this class *efficiency* is broadly equated to *polynomial time*. $O(n), O(n \log n), O(n^2), O(n^3), O(n^{100}), \ldots$ where n is size of the input.

So, is there an *efficient/good/effective* algorithm for primality?

Question

What does efficiency mean?

In this class *efficiency* is broadly equated to *polynomial time*. $O(n), O(n \log n), O(n^2), O(n^3), O(n^{100}), \ldots$ where n is size of the input.

Why? Is n^{100} really efficent/practical? Etc.

So, is there an *efficient/good/effective* algorithm for primality?

Question

What does efficiency mean?

In this class *efficiency* is broadly equated to *polynomial time*. $O(n), O(n \log n), O(n^2), O(n^3), O(n^{100}), \ldots$ where n is size of the input.

Why? Is n^{100} really efficent/practical? Etc.

Short answer: polynomial time is the single, robust, mathematically sound way to define efficiency.

Primes is in P!

Theorem (Agrawal-Kayal-Saxena'02)

There is a polynomial time algorithm for primality.

First polynomial time algorithm for testing primality. Running time is $O(\log^{12} n)$ further improved to about $O(\log^{6} n)$ by others.

Breakthrough announced in August 2002. Three days later announced in New York Times. Only 9 pages!

Primes is in P!

Theorem (Agrawal-Kayal-Saxena'02)

There is a polynomial time algorithm for primality.

First polynomial time algorithm for testing primality. Running time is $O(\log^{12} n)$ further improved to about $O(\log^{6} n)$ by others.

Breakthrough announced in August 2002. Three days later announced in New York Times. Only 9 pages!

Neeraj Kayal and Nitin Saxena were undergraduates at IIT-Kanpur!

What about before 2002?

Primality testing a key part of cryptography. What was the algorithm being used before 2002?
Miller-Rabin *randomized* algorithm:

- runs in polynomial time: $O(\log^2 n \log\log n 2^{O(\log^* n)})$.
- if n is prime correctly says "yes".
- if n is composite it says "yes" with probability at most $1/2^{100}$ (can be reduced further at the expense of more running time).

Based on Fermat's little theorem and some basic number theory.
More practical (the $O(\log^3 n)$ version) and faster than the deterministic algorithm.

Factoring

Modern public-key cryptography based on RSA (Rivest-Shamir-Adelman) system.
Relies on the difficulty of factoring a composite number into its prime factors.
There is a polynomial time algorithm that decides whether a given number n is prime or not (hence composite or not) but no known polynomial time algorithm to factor a given number.

Lesson

Intractability can be useful!

Quantum Computing

Theorem (Shor'1994)

There is a polynomial time algorithm for factoring on a quantum *computer.*

RSA and current commercial cryptographic systems can be broken if a quantum computer can be built!

Quantum Computing

Theorem (Shor'1994)

There is a polynomial time algorithm for factoring on a quantum *computer.*

RSA and current commercial cryptographic systems can be broken if a quantum computer can be built!

Lesson

Pay attention to the model of computation.

RAM Model

Random Access Memory model.

- data in memory can be accessed in one time step using an address
- simple arithmetic operations such as addition/multiplication of numbers can be done in one step

RAM model not valid for all problems, especially numerical problems (example: primality, multiplication, ...).

Why not Turing machines?

- RAM model better reflects the capabilities of a real computer for many common problems.
- TMs are too cumbersome

Part IV

The Mathematics of Dating

Warning!!

This lecture is intended for a mature audience

Warning!!

This lecture is intended for a mathematically mature audience

Warning!!

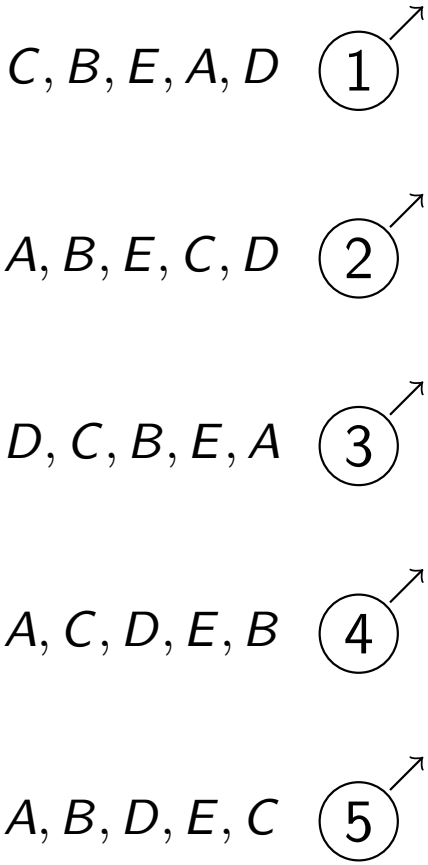
This lecture is intended for a mathematically mature audience
This lecture contains material that may be shocking to some students

Dating Scenario

- There are n boys and n girls
- Each girl has her own ranked preference list of all the boys
- Each boy has his own ranked preference list of all the girls
- The lists have no ties

Question: How do we pair the boys and girls?

Dating Scenario: An Example



Stable Pairing

Definition

Suppose we pair off all the boys and girls. Now if there is some boy b and some girl g who prefer each other to whom they are paired. Then (b, g) will be called a unhappy pair.

Stable Pairing

Definition

Suppose we pair off all the boys and girls. Now if there is some boy b and some girl g who prefer each other to whom they are paired. Then (b, g) will be called a unhappy pair.

- Unhappy pair motto: Why be with our partners when we can be with each other?

Stable Pairing

Definition

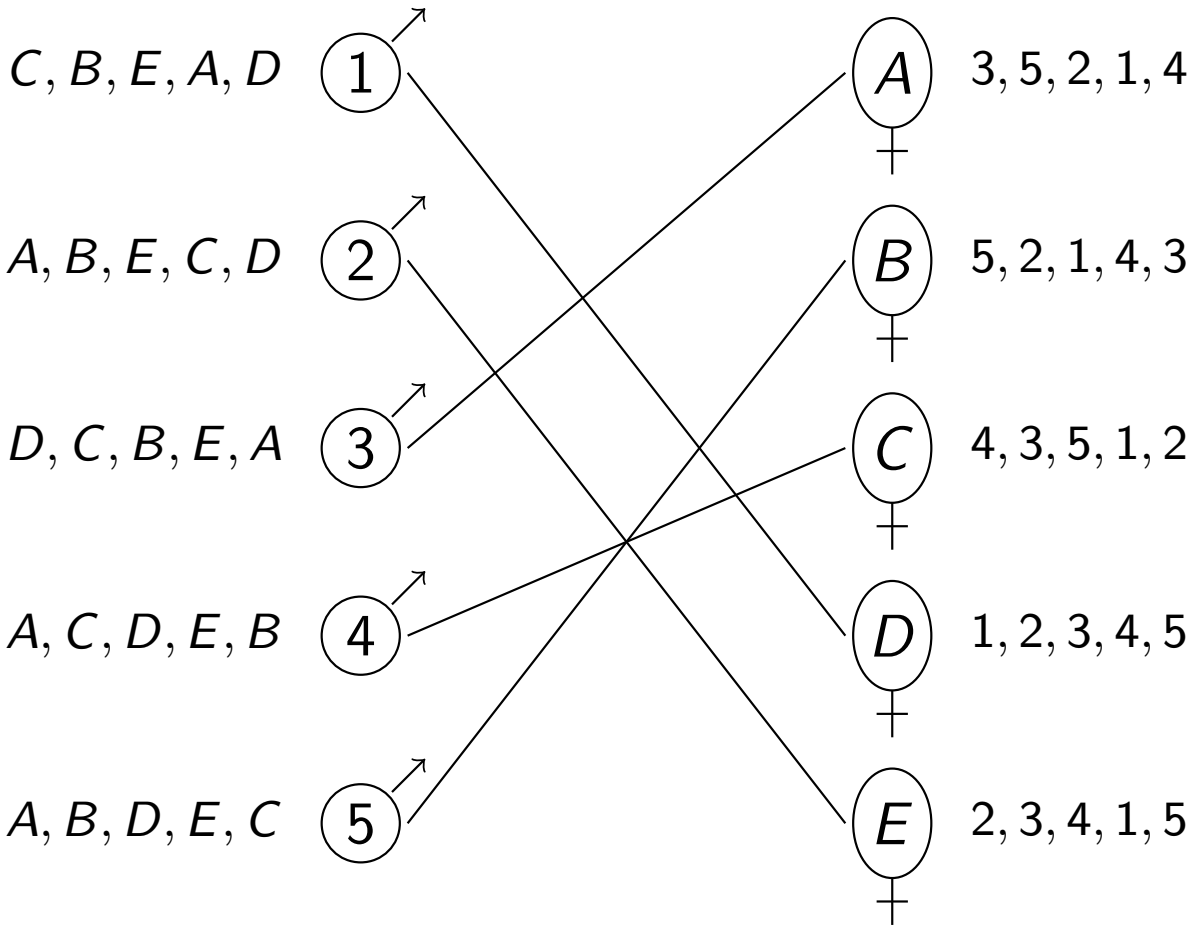
Suppose we pair off all the boys and girls. Now if there is some boy b and some girl g who prefer each other to whom they are paired. Then (b, g) will be called a unhappy pair.

Definition

A pairing of boys and girls is called stable if it contains no unhappy pairs.

Moral Code: Sustainability must be prerequisite for "goodness".

Stable Pairing: An Example



Stable Marriage Problem

Problem

Given a set of preference lists (one for each boy and girl), how do we find a stable pairing?

Stable Marriage Problem

Problem

Given a set of preference lists (one for each boy and girl), how do we find a stable pairing?

- But wait, how do we know that a stable pairing even exists!!

Stable Marriage Problem

Problem

Given a set of preference lists (one for each boy and girl), how do we find a stable pairing?

- But wait, how do we know that a stable pairing even exists!!
- Break up problem as follows

Stable Marriage Problem

Problem

Given a set of preference lists (one for each boy and girl), how do we find a stable pairing?

- But wait, how do we know that a stable pairing even exists!!
- Break up problem as follows
  1. Given a set of preference lists, does a stable pairing exist?
  2. Given a set of preference lists for which a stable pairing exists, design an algorithm to find a stable pairing.

A Simpler Problem

- Does every set of preference lists have a stable pairing?

A Simpler Problem

- Does every set of preference lists have a stable pairing?
- Idea: Start with some pairing. Allow pairs to now keep breaking up and reforming until they become stable.

A Simpler Problem

- Does every set of preference lists have a stable pairing?
- Idea: Start with some pairing. Allow pairs to now keep breaking up and reforming until they become stable.
- Challenge 1: Why will couples not continually break up and reform forever?

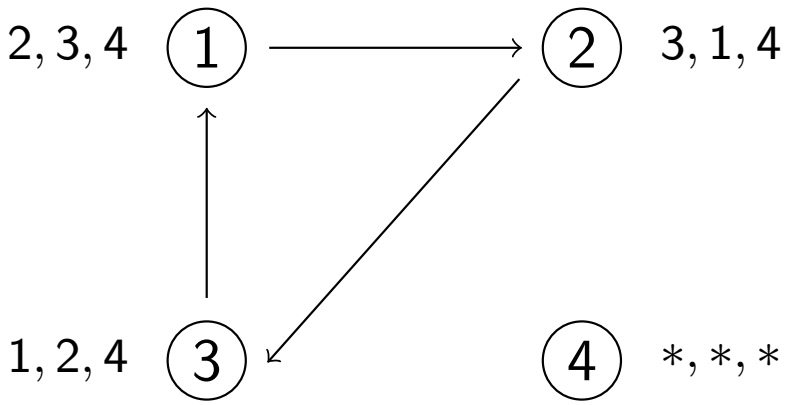
A Simpler Problem

- Does every set of preference lists have a stable pairing?
- Idea: Start with some pairing. Allow pairs to now keep breaking up and reforming until they become stable.
- Challenge 1: Why will couples not continually break up and reform forever?
  - Response 1: There are only finitely many pairings . . .

A Simpler Problem

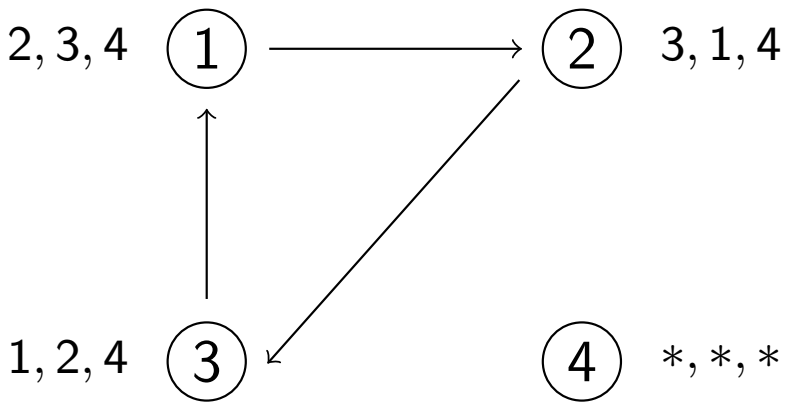
- Does every set of preference lists have a stable pairing?
- Idea: Start with some pairing. Allow pairs to now keep breaking up and reforming until they become stable.
- Challenge 1: Why will couples not continually break up and reform forever?
  - Response 1: There are only finitely many pairings ...
- Challenge 2: Why can't we be cycling through the same pairings?

Bisexual Dating (aka Stable Roommates Problem)



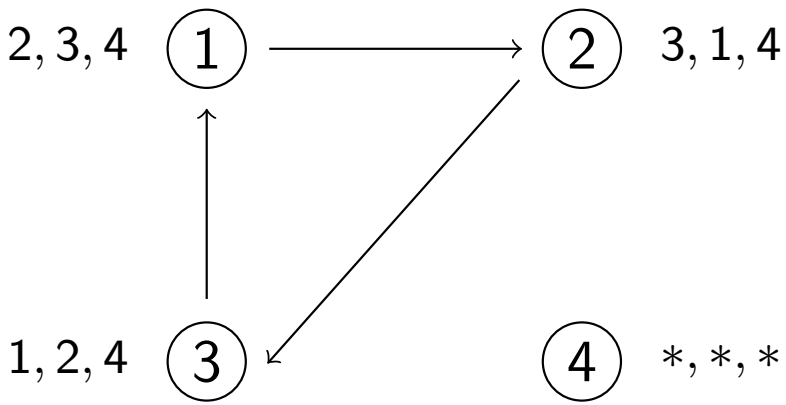
- Let x and y be paired with one other and let z be paired with 4

Bisexual Dating (aka Stable Roommates Problem)



- Let x and y be paired with one other and let z be paired with 4
- Observe, z prefers x and y to 4

Bisexual Dating (aka Stable Roommates Problem)



- Let x and y be paired with one other and let z be paired with 4
- Observe, z prefers x and y to 4
- One among x and y (say x) prefers z
- x and z form an unhappy pair; this process keeps repeating!

Existence of Stable Pairings

Does the break up/reform algorithm work for heterosexual case?

Existence of Stable Pairings

Does the break up/reform algorithm work for heterosexual case?

Not clear and not easy to analyze.

Instead, will show existence via a constructive algorithm.

Observation: multiple stable pairings possible

- b prefers g to g' and b' prefers g' to g.
- g prefers b' to b and g' prefers b to b'.
- $\{(b, g), (b', g')\}$ and $\{(b, g'), (b', g)\}$ are both stable pairings.

Traditional Marriage Algorithm (TMA)

The Gale-Shapley Algorithm (1962)

```
repeat
    Morning
        Each girl stands on her balcony
        Each boy proposes to the best girl whom he has not yet
            crossed off his list
        (A boy with an empty list stays home doing homework)
    Afternoon (for those girls with at least one suitor)
        To today's best suitor:  ``Maybe I will marry you.
            Come back tomorrow''
        To all other suitors that day:  ``I will never marry you''
    Evening
        Any rejected boy crosses the girl off his list
until no boy is rejected
Each girl marries the boy to whom she last said ``maybe''
```

Examining TMA

Questions:

- Does the algorithm terminate?
- Does the algorithm match each boy and girl (a perfect matching)?
- Does it produce a stable pairing?

Termination Proof

Lemma

The algorithm terminates in at most n^2 days.

Proof.

Termination Proof

Lemma

The algorithm terminates in at most n^2 days.

Proof.

If the algorithm does not terminate on day t then at least one girl got 2 or more suitors that day and one of them is rejected.

Termination Proof

Lemma

The algorithm terminates in at most n^2 days.

Proof.

If the algorithm does not terminate on day t then at least one girl got 2 or more suitors that day and one of them is rejected.

If g rejects b then b crosses g off his list and will never propose to her again.

Termination Proof

Lemma

The algorithm terminates in at most n^2 days.

Proof.

If the algorithm does not terminate on day t then at least one girl got 2 or more suitors that day and one of them is rejected.

If g rejects b then b crosses g off his list and will never propose to her again.

Total number of possible proposals is n^2 (each boy to each girl) and thus algorithm has to terminate after at most n^2 days □

Improvement Lemma

Lemma

If a girl g has boy b on a string (i.e., g has said maybe to b), then she will always have someone at least as good on a string, or for a husband.

Informal Proof.

g will let go of b only for a better suitor.

Maybe b will let go for a higher ranked (in his list) girl??

Improvement Lemma

Lemma

If a girl g has boy b on a string (i.e., g has said maybe to b), then she will always have someone at least as good on a string, or for a husband.

Informal Proof.

g will let go of b only for a better suitor.

Maybe b will let go for a higher ranked (in his list) girl??

Observation: If b courts g on day t, then b has already been rejected by day $t-1$ by all girls that he prefers to g. Continues to court g until she rejects (for a better boy) or marries him. □

Improvement Lemma: Formal Proof

Proof.

- Let q be the first day when she gets b on a string.

Improvement Lemma: Formal Proof

Proof.

- Let q be the first day when she gets b on a string.
- Suppose the lemma is false. Then there is a smallest $k \geq 1$ such that g has some b^* inferior to b on day $q + k$

Improvement Lemma: Formal Proof

Proof.

- Let q be the first day when she gets b on a string.
- Suppose the lemma is false. Then there is a smallest $k \geq 1$ such that g has some b^* inferior to b on day $q + k$
- Since k is the smallest number, on day $q + k - 1$, g has b_1 who is at least as good as b.

Improvement Lemma: Formal Proof

Proof.

- Let q be the first day when she gets b on a string.
- Suppose the lemma is false. Then there is a smallest $k \geq 1$ such that g has some b^* inferior to b on day $q + k$
- Since k is the smallest number, on day $q + k - 1$, g has b_1 who is at least as good as b.
- On day $q + k$, b_1 also appears (by observation).

Improvement Lemma: Formal Proof

Proof.

- Let q be the first day when she gets b on a string.
- Suppose the lemma is false. Then there is a smallest $k \geq 1$ such that g has some b^* inferior to b on day $q + k$
- Since k is the smallest number, on day $q + k - 1$, g has b_1 who is at least as good as b.
- On day $q + k$, b_1 also appears (by observation).
- Since b_1 is better than b^*, there is at least one suitor on day $q + k$ who is better than b^*.

Improvement Lemma: Formal Proof

Proof.

- Let q be the first day when she gets b on a string.
- Suppose the lemma is false. Then there is a smallest $k \geq 1$ such that g has some b^* inferior to b on day $q + k$
- Since k is the smallest number, on day $q + k - 1$, g has b_1 who is at least as good as b.
- On day $q + k$, b_1 also appears (by observation).
- Since b_1 is better than b^*, there is at least one suitor on day $q + k$ who is better than b^*.
- g will therefore not pick b^* on day $q + k$, which contradicts the assumption that b^* is picked on day $q + k$. □

Consequences of Improvement Lemma

Corollary

Each girl will marry her absolute favorite among all the boys who visit her during the TMA.

All boys paired!

Lemma

No boy can be rejected by all the girls.

Proof.

All boys paired!

Lemma

No boy can be rejected by all the girls.

Proof.

Proof by Contradiction

- Suppose b is rejected by all the girls

All boys paired!

Lemma

No boy can be rejected by all the girls.

Proof.

Proof by Contradiction

- Suppose b is rejected by all the girls
- Each girl that rejects g has a suitor when rejecting b. By Improvement Lemma, she eventually marries some boy.

All boys paired!

Lemma

No boy can be rejected by all the girls.

Proof.

Proof by Contradiction

- Suppose b is rejected by all the girls
- Each girl that rejects g has a suitor when rejecting b. By Improvement Lemma, she eventually marries some boy.
- That means, since we have n girls, we have at least n suitors not counting b. Thus, there are at least $n + 1$ boys!

Stability of TMA's pairing

Theorem

Let T be the pairing produced by TMA. T is stable.

Proof.

Stability of TMA's pairing

Theorem

Let T be the pairing produced by TMA. T is stable.

Proof.

Suppose (b, g) and (b^*, g^*) are two pairs in the output such that (b, g^*) is unhappy: b prefers g^* to g and g^* prefers b to b^*.

Stability of TMA's pairing

Theorem

Let T be the pairing produced by TMA. T is stable.

Proof.

Suppose (b, g) and (b^*, g^*) are two pairs in the output such that (b, g^*) is unhappy: b prefers g^* to g and g^* prefers b to b^*.

- During TMA b proposed to g^* before he proposed to g

Stability of TMA's pairing

Theorem

Let T be the pairing produced by TMA. T is stable.

Proof.

Suppose (b, g) and (b^*, g^*) are two pairs in the output such that (b, g^*) is unhappy: b prefers g^* to g and g^* prefers b to b^*.

- During TMA b proposed to g^* before he proposed to g
- At some point, g^* rejected b for someone better. By the Improvement Lemma, g^* marries b^* who she prefers to b

Stability of TMA's pairing

Theorem

Let T be the pairing produced by TMA. T is stable.

Proof.

Suppose (b, g) and (b^*, g^*) are two pairs in the output such that (b, g^*) is unhappy: b prefers g^* to g and g^* prefers b to b^*.

- During TMA b proposed to g^* before he proposed to g
- At some point, g^* rejected b for someone better. By the Improvement Lemma, g^* marries b^* who she prefers to b
- Hence (b, g^*) is not an unhappy couple!

Implementation

Lemma

The algorithm can be implemnted in $O(n^2)$ time.

Note: input size is $O(n^2)$ and hence time is linear in input size.

The algorithm in the book is essentially the same:

- maybe $\equiv$ engaged
- each day only one unengaged boy proposes

Opinion Poll

Who is better off in traditional dating, the boys or girls?

Optimal Girl

Question

How should we define the notion of "the optimal girl for a boy b"?

Optimal Girl

Question

How should we define the notion of "the optimal girl for a boy b"?

Flawed Definition

The girl at the top of b's list

Optimal Girl

Question

How should we define the notion of "the optimal girl for a boy b"?

Flawed Definition

The girl at the top of b's list

Definition

g is the optimal girl for b, if she is the highest ranked girl for whom there is *some* stable pairing in which b gets g.

Optimal Girl

Question

How should we define the notion of "the optimal girl for a boy b"?

Definition

g is the optimal girl for b, if she is the highest ranked girl for whom there is *some* stable pairing in which b gets g.

- Optimal Girl: g is the best girl b can conceivably get in a stable world.

Optimal Girl

Question

How should we define the notion of "the optimal girl for a boy b"?

Definition

g is the optimal girl for b, if she is the highest ranked girl for whom there is *some* stable pairing in which b gets g.

- Optimal Girl: g is the best girl b can conceivably get in a stable world.
- The optimal girl for b maybe better than the girl b gets by TMA.

Pessimal Girl

Definition

g is the pessimal girl for b, if she is the lowest ranked girl for whom there is *some* stable pairing in which b gets g.

- Pessimal Girl is the worst girl b can conceivably get in a stable world.

Optimal boy for a girl g and pessimal boy for a girl g can be similarly defined.

Dating Heaven and Hell: Boy's Perspective

Definition

A pairing is male optimal if every boy gets paired with his optimal girl.

- The best of all possible stable worlds for every boy simutaneously.

Definition

A pairing is male pessimal if every boy gets paired with his pessimal girl.

- The worst of all stable worlds for every boy simultaneously.

Dating Heaven and Hell: Girl's Perspective

Definition

A pairing is female optimal if every girl gets paired with her optimal boy.

- The best of all possible stable worlds for every girl simutaneously.

Definition

A pairing is female pessimal if every girl gets paired with her pessimal boy.

- The worst of all stable worlds for every girl simultaneously.

Naked Truth!

The Traditional Marriage Algorithm always produces a male-optimal, and female-pessimal pairing.

Male Optimality of TMA

Theorem

TMA produces a male-optimal pairing.

Proof.

By contradiction. Suppose some boy b gets rejected by his optimal girl g during TMA. Let t be the earliest time when a boy gets rejected by his optimal girl.

Male Optimality of TMA

Theorem

TMA produces a male-optimal pairing.

Proof.

By contradiction. Suppose some boy b gets rejected by his optimal girl g during TMA. Let t be the earliest time when a boy gets rejected by his optimal girl.

- At time t, b got rejected by g because she had on a string b^* whom she prefered.

Male Optimality of TMA

Theorem

TMA produces a male-optimal pairing.

Proof.

By contradiction. Suppose some boy b gets rejected by his optimal girl g during TMA. Let t be the earliest time when a boy gets rejected by his optimal girl.

- At time t, b got rejected by g because she had on a string b^* whom she prefered.
- Since t is the earliest time, b^* at time t has not been rejected by his optimal girl.

Male Optimality of TMA

Theorem

TMA produces a male-optimal pairing.

Proof.

By contradiction. Suppose some boy b gets rejected by his optimal girl g during TMA. Let t be the earliest time when a boy gets rejected by his optimal girl.

- At time t, b got rejected by g because she had on a string b^* whom she prefered.
- Since t is the earliest time, b^* at time t has not been rejected by his optimal girl.
- Thus b^* likes g at least as much as his optimal girl. □

Male Optimality of TMA (contd)

Proof.

b^* likes g at least as much as his optimal girl. And b got rejected by his optimal girl g because of b^*.

Male Optimality of TMA (contd)

Proof.

b^* likes g at least as much as his optimal girl. And b got rejected by his optimal girl g because of b^*.

- Let S be a stable pairing where b gets paired with g

Male Optimality of TMA (contd)

Proof.

b^* likes g at least as much as his optimal girl. And b got rejected by his optimal girl g because of b^*.

- Let S be a stable pairing where b gets paired with g (exists, because g is b's optimal girl).
- b^* would like to be with g over his wife in S, say g^*

Male Optimality of TMA (contd)

Proof.

b^* likes g at least as much as his optimal girl. And b got rejected by his optimal girl g because of b^*.

- Let S be a stable pairing where b gets paired with g (exists, because g is b's optimal girl).
- b^* would like to be with g over his wife in S, say g^*
  - g is at least as good as b^*'s optimal girl and b^* is not with g in S.

Male Optimality of TMA (contd)

Proof.

b^* likes g at least as much as his optimal girl. And b got rejected by his optimal girl g because of b^*.

- Let S be a stable pairing where b gets paired with g (exists, because g is b's optimal girl).
- b^* would like to be with g over his wife in S, say g^*
  - g is at least as good as b^*'s optimal girl and b^* is not with g in S.
- g likes b^* more than b

Male Optimality of TMA (contd)

Proof.

b^* likes g at least as much as his optimal girl. And b got rejected by his optimal girl g because of b^*.

- Let S be a stable pairing where b gets paired with g (exists, because g is b's optimal girl).
- b^* would like to be with g over his wife in S, say g^*
  - g is at least as good as b^*'s optimal girl and b^* is not with g in S.
- g likes b^* more than b
  - g rejected b because she had b^* on a string.

Male Optimality of TMA (contd)

Proof.

b^* likes g at least as much as his optimal girl. And b got rejected by his optimal girl g because of b^*.

- Let S be a stable pairing where b gets paired with g (exists, because g is b's optimal girl).
- b^* would like to be with g over his wife in S, say g^*
  - g is at least as good as b^*'s optimal girl and b^* is not with g in S.
- g likes b^* more than b
  - g rejected b because she had b^* on a string.
- S is not stable since (b^*, g) are an unhappy pair. □

Male Optimality of TMA

optimal(b): optimal girl for b

Corollary

The pairs $(b, \mathit{optimal}(b))$ *is a perfect matching.*

Female Pessimality of TMA

Theorem

TMA produces a female-pessimal pairing.

Proof.

Let T be produced by TMA. We know T is male-optimal. The proof proceeds by contradiction.

Female Pessimality of TMA

Theorem

TMA produces a female-pessimal pairing.

Proof.

Let T be produced by TMA. We know T is male-optimal. The proof proceeds by contradiction.

- Suppose there is a stable pairing S in which some girl g does worse than T

Female Pessimality of TMA

Theorem

TMA produces a female-pessimal pairing.

Proof.

Let T be produced by TMA. We know T is male-optimal. The proof proceeds by contradiction.

- Suppose there is a stable pairing S in which some girl g does worse than T
- Let b be g's husband in T and let b^* be g's husband in S.
- S is not stable

Female Pessimality of TMA

Theorem

TMA produces a female-pessimal pairing.

Proof.

Let T be produced by TMA. We know T is male-optimal. The proof proceeds by contradiction.

- Suppose there is a stable pairing S in which some girl g does worse than T
- Let b be g's husband in T and let b^* be g's husband in S.
- S is not stable
  - By assumption, g likes b more than b^*

Female Pessimality of TMA

Theorem

TMA produces a female-pessimal pairing.

Proof.

Let T be produced by TMA. We know T is male-optimal. The proof proceeds by contradiction.

- Suppose there is a stable pairing S in which some girl g does worse than T
- Let b be g's husband in T and let b^* be g's husband in S.
- S is not stable
  - By assumption, g likes b more than b^*
  - b likes g more than his mate in S, because g is his optimal girl.

Female Pessimality of TMA

Theorem

TMA produces a female-pessimal pairing.

Proof.

Let T be produced by TMA. We know T is male-optimal. The proof proceeds by contradiction.

- Suppose there is a stable pairing S in which some girl g does worse than T
- Let b be g's husband in T and let b^* be g's husband in S.
- S is not stable
  - By assumption, g likes b more than b^*
  - b likes g more than his mate in S, because g is his optimal girl.
  - (b, g) are an unhappy pair in S.

□

$pessimal(g)$: pessimal boy for g.

Corollary

$pessimal(optimal(b)) = b$

Stable Marriage in Practice: Medical Residencies

- World's most successful dating service: Matching doctors to Medical residencies
  - Students submit a ranked list of hospitals
  - Hospitals submit a ranked list of new doctors
  - Computer runs TMA (extended to handle Mormon marriages)
  - Until recently, it was hospital optimal
- Assigning traffic to web-servers: Akamai Inc.

Reading Assignment

Chapters 1-3 from the textbook. At least half is prereq material.