

CS 401: Computer Algorithms I

Stable Matching

Xiaorui Sun

Administrativa Stuffs

- Website: <http://www.cs.uic.edu/~xiaorui/cs401>
 - Lecture slides, homework:
- Piazza:
<https://piazza.com/uic/spring2024/cs40143452434534345743458>
 - Announcements, online discussion forum
 - TA will answer course related questions

Last Lecture (summary)

man-woman pairs s.t. everyone participates exact one pair

Stable matching problem: Given n men and n women, and their preferences, find a stable matching.

For a perfect matching M , a pair $m-w$ is **unstable** if they prefer each other to their match in M .

	favorite ↓ 1st	2nd	least favorite ↓ 3rd
Xavier	Amy	Brenda	Claire
Yuri	Brenda	Amy	Claire
Zoran	Amy	Brenda	Claire

Men's Preference Profile

	favorite ↓ 1st	2nd	least favorite ↓ 3rd
Amy	Yuri	Xavier	Zoran
Brenda	Xavier	Yuri	Zoran
Claire	Xavier	Yuri	Zoran

Women's Preference Profile

Q: Is Yuri-Brenda an unstable pair?

A: No, Yuri and Brenda get matched.

Last Lecture (summary)

Stable matching problem: Given n men and n women, and their preferences, find a stable matching.

For a perfect matching M , a pair $m-w$ is **unstable** if they prefer each other to their match in M .

	favorite ↓ 1st	2nd	least favorite ↓ 3rd
Xavier	Amy	Brenda	Claire
Yuri	Brenda	Amy	Claire
Zoran	Amy	Brenda	Claire

Men's Preference Profile

	favorite ↓ 1st	2nd	least favorite ↓ 3rd
Amy	Yuri	Xavier	Zoran
Brenda	Xavier	Yuri	Zoran
Claire	Xavier	Yuri	Zoran

Women's Preference Profile

Q: Is Yuri-Amy an unstable pair?

A: No. Yuri prefer Brenda to Amy

Last Lecture (summary)

Stable matching problem: Given n men and n women, and their preferences, find a stable matching.

For a perfect matching M , a pair $m-w$ is **unstable** if they prefer each other to their match in M .

	favorite ↓ 1 st	2 nd	least favorite ↓ 3 rd
Xavier	Amy	Brenda	Claire
Yuri	Brenda	Amy	Claire
Zoran	Amy	Brenda	Claire

Men's Preference Profile

	favorite ↓ 1 st	2 nd	least favorite ↓ 3 rd
Amy	Yuri	Xavier	Zoran
Brenda	Xavier	Yuri	Zoran
Claire	Xavier	Yuri	Zoran

Women's Preference Profile

Q: Is Xavier-Amy an unstable pair?

A: Yes.

Propose-And-Reject Algorithm [Gale-Shapley'62]

Initialize each person to be free.

```
while (some man is free and hasn't proposed to every woman) {  
    Choose such a man m  
    w = 1st woman on m's list to whom m has not yet proposed  
    if (w is free)  
        assign m and w to be engaged  
    else if (w prefers m to her current partner m')  
        assign m and w to be engaged, and m' to be free  
    else  
        w rejects m  
}
```

Questions

- Q: Why GS algorithm solves Stable Matching problem?
- Q: How to implement GS algorithm efficiently?

What do we need to prove?

Goal: prove Propose-And-Reject Algorithm always finds a stable matching.

- The algorithm ends.
- The output is correct.

Correctness proof is not required in the homework and exams.

But understanding the correctness helps you develop correct algorithms.

Propose-And-Reject Algorithm [Gale-Shapley'62]

Initialize each person to be free.

```
while (some man is free and hasn't proposed to every woman) {  
    Choose such a man m  
    w = 1st woman on m's list to whom m has not yet proposed  
    if (w is free)  
        assign m and w to be engaged  
    else if (w prefers m to her fiancé m')  
        assign m and w to be engaged, and m' to be free  
    else  
        w rejects m  
}
```

What do we need to prove?

Goal: prove Propose-And-Reject Algorithm always finds a stable matching.

- The algorithm ends.
How many iterations it takes?
- The output is correct.
It find a **perfect** matching that is **stable**.

Proof of Correctness: Termination

Each step, a man proposed to a new woman.

One strategy to bound # iterations is to find a measure of progress.

There are $n \times n = n^2$ possible man-to-woman proposals.

Therefore, it takes at most n^2 iterations.

	1 st	2 nd	3 rd	4 th	5 th
Victor	A	B	C	D	E
Walter	B	C	D	A	E
Xavier	C	D	A	B	E
Yuri	D	A	B	C	E
Zoran	A	B	C	D	E

	1 st	2 nd	3 rd	4 th	5 th
Amy	W	X	Y	Z	V
Brenda	X	Y	Z	V	W
Claire	Y	Z	V	W	X
Diane	Z	V	W	X	Y
Erika	V	W	X	Y	Z

$n(n-1) + 1$ proposals required

Main Properties of the algorithm

Observation 1: Men propose to women in decreasing order of preference.

Observation 2: Once a woman is matched, she never becomes unmatched; she only "trades up."

```
Initialize each person to be free.
```

```
while (some man is free and hasn't proposed to every woman) {  
    Choose such a man m  
    w = 1st woman on m's list to whom m has not yet proposed  
    if (w is free)  
        assign m and w to be engaged  
    else if (w prefers m to her fiancé m')  
        assign m and w to be engaged, and m' to be free  
    else  
        w rejects m  
}
```

Proof of Correctness: Perfection

Claim. All men and women get matched.

Proof. (by contradiction)

Suppose, for sake of contradiction, that **Zoran** is not matched upon termination of algorithm.

Then some woman, say **Amy**, is not matched upon termination.

(Observation 2: once women matched, they never becoming unmatched.) **Amy** was never proposed to.

But, **Zoran** proposes to everyone, since he ends up unmatched.



Proof of Correctness: Stability

Claim. No unstable pairs.

Proof. (by contradiction)

Suppose **A-Z** is an unstable pair: each prefers each other to the partner in Gale-Shapley matching.

Case 1: **Z** never proposed to **A**.

⇒ **Z** prefers his GS partner to **A**.

⇒ **A-Z** is stable.

men propose in decreasing order of preference

Case 2: **Z** proposed to **A**.

⇒ **A** rejected **Z** (right away or later)

⇒ **A** prefers her GS partner to **Z**.


⇒ **A-Z** is stable.

women only trade up

In either case **A-Z** is stable, a contradiction.



Questions

- Q: Why GS algorithm solves Stable Matching problem? 
- Q: How to implement GS algorithm efficiently?
 - Different implementations may have different running time

Propose-And-Reject Algorithm [Gale-Shapley'62]

Can be improved to $O(n)$ using the fact that if a man is free, then he hasn't proposed to every woman yet (any free man is fine)

Initialize each person to be free.

```
while (some man is free and hasn't proposed to every woman) {  
    Choose such a man  $m$   
     $w = 1^{\text{st}}$  woman on  $m$ 's list to whom  $m$  has not yet proposed  
    if ( $w$  is free)  
        assign  $m$  and  $w$  to be engaged  
    else if ( $w$  prefers  $m$  to her fiancé  $m'$ )  
        assign  $m$  and  $w$  to be engaged, and  $m'$  to be free  
    else  
         $w$  rejects  $m$   
}
```

$O(n^2)$
time

$O(n)$
time

$O(n)$
time

- Maintain two arrays $wife[m]$, and $husband[w]$.
 - set entry to 0 if unmatched
 - if m matched

Can be improved to $O(n^3)$ $wife[w]=m$

Overall: $O(n^4)$
time

- Maintain a matrix $proposed[m, w]$.
 - Set entry to 1 if m has proposed to w , otherwise set to 0

Efficient Implementation

We describe $O(n^2)$ time implementation.

Representing men and women:

Assume men are named **1, ..., n**.

Assume women are named **n+1, ..., 2n**.

Free men:

Maintain two arrays **wife[m]**, and **husband[w]**.

- set entry to **0** if unmatched
- if **m** matched to **w** then **wife[m]=w** and **husband[w]=m**

Maintain a list of free men, e.g., in a queue.

Men proposing:

For each man, maintain a list of women, ordered by preference **pref[m/w, i]**.

Maintain an array **count[m]** that counts the number of proposals made by man **m**.

Propose-And-Reject Algorithm [Gale-Shapley'62]

Initialize each person to be free.

```
while (some man is free and hasn't proposed to every woman) {  
    Choose such a man m  
    w = 1st woman on m's list to whom m has not yet proposed  
    if (w is free)  
        assign m and w to be engaged  
    else if (w prefers m to her fiancé m')  
        assign m and w to be engaged, and m' to be free  
    else  
        w rejects m  
}
```

O(1)
time

O(1)
time

A Preprocessing Idea

Women rejecting/accepting.

Does woman **w** prefer man **m** to man **m'**?

For each woman, create **inverse** of preference list of men.

Constant time access for each query after $O(n)$ preprocessing per woman.
 $O(n^2)$ total preprocessing cost.

Amy	1 st	2 nd	3 rd	4 th	5 th	6 th	7 th	8 th
Pref	8	3	7	1	4	5	6	2

Amy	1	2	3	4	5	6	7	8
Inverse	4 th	8 th	2 nd	5 th	6 th	7 th	3 rd	1 st

```
for i = 1 to n
  inverse[amy, pref[amy, i]] = i
```

Amy prefers man **3** to **6**, since
 $\text{inverse}[\text{amy}, 3] = 2 < 7 = \text{inverse}[\text{amy}, 6]$

Propose-And-Reject Algorithm [Gale-Shapley'62]

Initialize each person to be free.

```
while (some man is free and hasn't proposed to every woman) {  
    Choose such a man m  
    w = 1st woman on m's list to whom m has not yet proposed  
    if (w is free)  
        assign m and w to be engaged  
    else if (w prefers m to her fiancé m')  
        assign m and w to be engaged, and m' to be free  
    else  
        w rejects m  
}
```

O(1)
time

O(1)
time

O(1)
time

Overall: O(n²)
time

Implementation Summary

We can implement GS algorithm in $O(n^2)$ time.

- Problem size: $N=2n^2$ words
 - $2n$ people each with a preference list of length n

Q. Why do we care?

A. Usually, the running time is lower-bounded by input length.

- GS is the best we can hope for the stable matching problem ($O(N)$ time).

Different implementations of same algorithm may have different running time.

Stable Matching Summary

- **Stable matching problem:** Given n men and n women, and their preferences, find a stable matching.
- **Gale-Shapley algorithm:** Guarantees to find a stable matching for **any** problem instance.
- **Q:** Why GS algorithm solves Stable Matching problem?
- **Q:** How to implement GS algorithm efficiently?
Different implementations may have different running time

Why this problem is important?

In 1962, Gale and Shapley published the paper
“College Admissions and the Stability of Marriage”
To
“The American Mathematical Monthly”

COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only q . Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the q best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept. Accordingly, in order for a college to receive q acceptances, it will generally have to offer to admit more than q applicants. The problem of determining how many and which ones to admit requires some rather involved guesswork. It may not be known (a) whether a given applicant has also applied elsewhere; if this is known it may not be known (b) how he ranks the colleges to which he has applied; even if this is known it will not be known (c) which of the other colleges will offer to admit him. A result of all this uncertainty is that colleges can expect only that the entering class will come reasonably close in numbers to the desired quota, and be reasonably close to the attainable optimum in quality.



David Gale (1921-2008)
PROFESSOR, UC BERKELEY



Lloyd Shapley
PROFESSOR EMERITUS, UCLA

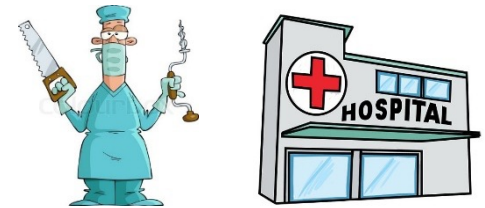
Why this problem is important?



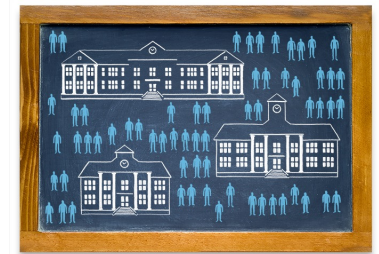
Alvin Roth
PROFESSOR, STANFORD

Alvin Roth modified the Gale-Shapley algorithm and apply it to

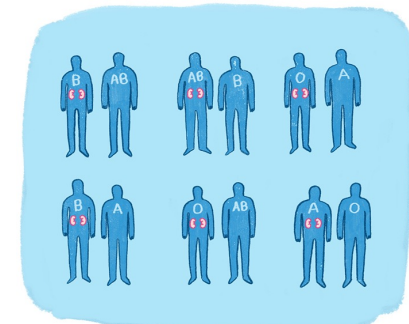
- National Residency Match Program (NRMP), a system that assigns new doctors to hospitals around the country. (90s)



- Public high school assignment process (00s)



- Helping transplant patients find a match (2004)
(Saved >1,000 people every year!)



Blood types: A, B, AB and O

Why this problem is important?



Some of the problems in this course may seem obscure or even pointless.

But their abstraction allows for variety of applications.

Shapley and Roth got the Nobel Prize (Economic) in 2012.
(David Gale passed away in 2008.)