Stable Matching
Overview of Greedy Algorithms

The Stable Marriage Problem (input)

Goal. Given n men, n women, and their preference lists, find a "suitable" matching.

- Participants rate members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference from best to worst.

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's Preference lists

Note!

- The stable matching problem was historically stated in terms of men/women (that's why it's sometimes called the stable marriage problem).

- The original problem statement is old and outdated, may raise gender stereotype concerns. But the general matching problem has broad applications:
  - Matching Medical students and Hospital residency places in US
  - Auction mechanisms for sponsored internet search
  - JUPAS
  - ……

The Stable Matching Problem

Problem. Finding a stable matching between two equally sized of elements given an ordering of preferences for each element.

A matching is a bijection from the elements of one set to the elements of the other set.

Later on, we will use the job applicants and employers as example to introduce the stable matching problem and the greedy algorithm that solves it.

The Stable Matching Problem (input)

Goal. Given n job applicants, n employers, and their preference lists, find a "suitable" matching.

- Participants rate members of the other type.
- Each job applicant lists employers in order of preference from best to worst.
- Each employer lists job applicants in order of preference from best to worst.

The Stable Matching Problem (output)

A set of n e-a pairs that constitute a perfect and stable matching

Perfect matching: bijection.

- Each employer is matched to exactly one applicant.
- Each applicant is matched to exactly one employer.

Stability: no incentive for any unmatched pair to undermine assignment by joint action.

- In matching M, an unmatched pair e-a is unstable if employer e and applicant a prefer each other to their current matched partners.
- Unstable pair e-a would prefer to leave the system and be matched to each other.

Stable matching: perfect matching with no unstable pairs.

> Stable matching problem.
> Given the preference lists of n employers and n applicants, find a stable matching (if one exists).

Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Employers' Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Applicants' Preference Lists

Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?

No. X-B is an unstable pair.

X and B prefer each other to their current matches (C and Y, respectively).

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Employers' Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Applicants' Preference Lists

Stable Matching Problem

Q. Is assignment X-A, Y-B, Z-C stable?

A. Yes.

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Employers' Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Applicants' Preference Lists

Stable Matching Problem

Q. Do stable matchings always exist?

A. Yes. (This is not obvious; we will prove later)

Q. Is the stable matching unique?

A. No. It is possible that there are several stable matchings

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Employers' Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Applicants' Preference Lists

Shaded boxes are stable matching X-B, Y-A, Z-C

Previous page had stable matching X-A, Y-B, Z-C for same lists!

Gale-Shapley Algorithm

Gale-Shapley algorithm. [Gale-Shapley 1962]
Intuitive algorithm that guarantees to find a stable matching.
Also known as the **deferred acceptance algorithm** or **propose-and-reject algorithm**.

```
Initialize each participant (employers and applicants) to be free.
while (some employer is free and has an applicant to send an offer
to) {
    Choose such an employer e
    a = 1st applicant on e's list to whom e has not yet sent an offer
    if (a is free)
        assign e and a to be matched
    else if (a prefers e to the current employer e*)
        assign e and a to be matched, and e* to be free
    else
        a rejects e
}
```

Shapley won Nobel Prize in Economics (partially) for this in 2012.
Gale was not eligible because he had died in 2008.

Proof of Correctness: Termination

Claim. Algorithm terminates after at most n^2 iterations of while loop.

Pf. An employer starts from the first applicant in the list and then continues in decreasing order of preference, without sending offer to the same applicant again. There are only n^2 possible offers. ▪

Proof of Correctness: Perfection

Perfection means that in the end of the algorithm each employer and applicant gets matched to exactly one partner.

Observation 1. Employers send offers to applicants in decreasing order of preference.

Observation 2. Once an applicant is matched, this applicant never becomes unmatched; matched applicant only "trades up."

Claim. All employers and applicants get matched.

Pf. (by contradiction)

- Suppose, for sake of contradiction, that employer Z is not matched upon termination of algorithm.
- Then some applicant, say A, is not matched upon termination.
- By Observation 2, A never received an offer.
- But, Z must have sent offer to everyone, since Z ends up unmatched.
- Contradiction! ▪

Proof of Correctness: Stability

Claim. No unstable pairs.

Pf. (by contradiction)

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

- Case 1: Z never sent an offer to A.
  - ⇒ Z prefers the current partner to A.
  - ⇒ A-Z is stable.

employers send offers in decreasing order of preference

S*

A-Y
B-Z
. . .

- Case 2: Z sent an offer to A.
  - ⇒ A rejected Z (right away or later)
  - ⇒ A prefers the current partner to Z. ← applicants only trade up
  - ⇒ A-Z is stable.

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

Efficient Implementation

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

Representing employers and applicants.

- Assume employers are named 1, …, n.
- Assume applicants are named 1', …, n'.

Contracts.

- Maintain a list of free employers, e.g., in a queue or stack.
- Maintain two arrays `employee[e]`, and `employer[a]`.
  - if e is matched to a then `employee[e]=a` and `employer[a]=e`
  - if e or a is unmatched, set `employee[e]` or `employer[a] = 0` correspondingly.

Employers sending offers.

- For each employer, maintain a list (linked list or array) of applicants, ordered by preference.

Efficient Implementation

Applicants accepting/rejecting.

- Does applicant `a` prefer employer `e` to employer `e*`?
- Naïve implementation requires O(n) time for this comparison.
- For each applicant, create inverse of preference list of employers.
- Constant time access for each query after O(n) preprocessing.

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

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

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

Amy prefers employer 3 to employer 6 since `inverse[3]` < `inverse[6]`

2 7

Understanding the Solution

Q. For a given problem instance, several stable matchings might exist.
Recall that Gale-Shapely gives us freedom to decide *which* employer sends offer.
Do all executions of Gale-Shapley (for different orders in which employers send offer) yield the same stable matching?
If so, which one?

A. Yes, it always returns the (unique) matching that is optimal for the employers (proof omitted).

> Optimal for the employers means: each employer gets the best possible partner in any possible stable matching.

Observation. Employers and applicants are not equivalent in the algorithm.
Employers send offers, applicants accept/reject.

> To get an applicant optimal algorithm, let applicants apply and employers accept/reject.

Several variants of the problem exist with multiple applications.

Stable Matching: Questions

Consider the group e1, e2, a1, and a2 with preference lists:

e1: a1, a2	e2: a2, a1
a1: e2, e1	a2: e1, e2

1. What is an employer-optimal stable matching in the above example:
(e1,a1), (e2,a2)

2. What is an applicant-optimal stable matching in the above example:
(e1,a2), (e2,a1)

3.In every instance of the Stable Matching Problem, there is a stable matching containing a pair (e, a) such that e is ranked first on the preference list of a and a is ranked first on the preference list of e.
True or False?
False. See above example

4. Consider that there exists an employer e and an applicant a such that e is ranked first on the preference list of a and a is ranked first on the preference list of e. Then, the pair (e, a) belongs to every stable matching. True or False?
True. Otherwise the pair (e, a) would be unstable.

Stable Matching: Questions

1. Suppose that each employer has a different most favorite applicant. How many steps does it take for the algorithm to converge?

A single round (n offers). Each employer sends an offer to a different applicant who accepts the offer. No applicant will receive offer again.

2. Suppose that the employers have identical preferences. How many rounds does it take for the algorithm to converge?

At round 1, all employers send offer to the same applicant. That applicant selects the favorite. At the second round, all but that favorite employer send offer to the second-best applicant. The second-best applicant selects the favorite. The process continues for n rounds until one employer is left to send offer to the least desirable applicant.

Roommates Problem

$2n$ people must be paired up to be roommates in n rooms. For each person we have the list of preferences, but the graph is not bi-partite (a person can be matched with any of the remaining $2n-1$). Is it always possible to find a stable match?

Assume 4 people A,B,C,D with preference lists:
A: B C D
B: C A D
C: A B D
D: any
How many different matchings exist?
3, the number of choices for a single person – the other two people must be matched together.
If we match (A,B), (C,D):
then (B,C) is unstable.
If we match (A,C), (B,D):
then (A,B) is unstable.
If we match (A,D), (B,C):
then (A,C) is unstable.
Any matching will contain an unstable pair.

JUPAS Admission Scheme

Applicants correspond to applicants that order their programme choices (which correspond to employers).
Differences w.r.t. to the basic version of the problem:

1. Many more applicants than programmmes (places)
2. A programme is matched to many (instead of one) applicants.
   # it's matched to is its *intake*

The JUPAS computer system will match (the "iteration process") the order of preference you have assigned to your programme choice list with the position you have been placed in each merit order list of these programmes.

You will then be made an offer of the highest priority on your programme choice list for which you have the required rating.

That is, the matching is optimal (i.e., fair) for the applicants!

Greedy Overview

1. Greedy algorithms are often the simplest possible algorithms to design
   - They build solutions, one step at a time.
   - Sometimes the solution is optimal, sometimes not.

2. Proving that the greedy solution is optimal is usually the hard part
   - Greedy is not always optimal (often isn't)
   - There is not only one way to prove optimality
     - We saw several standard approaches
     - Note that not every method can work for every problem.

Three Different Proof Techniques

1. Modifying an Optimal Solution into Greedy (cost common)
   - Did this for **Interval Scheduling** & **Fractional Knapsack** and several exercises
   - Start (conceptually) with different ***G***(greedy) and ***O***(optimal) solutions
     - Show how ***O*** can be modified to ***O'*** that is still optimal but closer to ***G***
     - Repeat, until have created optimal solution that is exactly ***G***

2. Lower Bound Technique
   - Did this for **Interval Partitioning**
   - For problems trying to find **minimum** solution (can be modified for max)
     - Define value **L** that is a lower bound on ANY feasible solution
     - Show that Greedy solution has value **L**

3. Algorithm-specific Techniques
   - Inductive Proof for **Huffman Coding**
   - Proof by Contradiction for **Stable Matching**