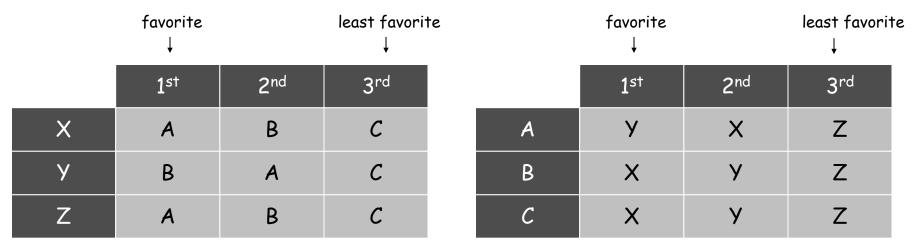
Goal. Given n companies and n applicants, find a "suitable" matching.

- Companies rate applicants, applicants rate companies.
- Each company lists applicants in order of preference from best to worst.



Company's Preference Profile

Applicant's Preference Profile

Perfect matching:

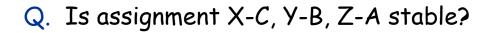
- Each company gets exactly one applicant.
- Each applicant gets exactly one company.

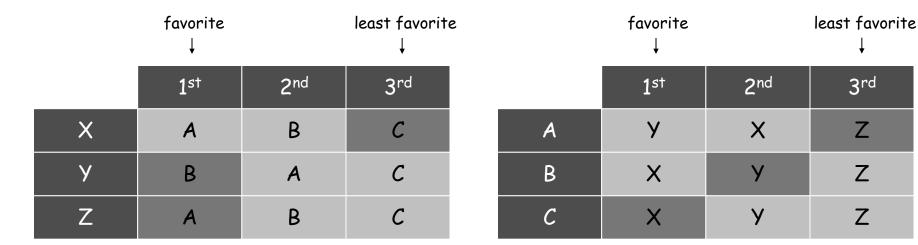
Stability: no incentive for some pair of participants to undermine assignment by joint action.

- In matching M, an unmatched pair c-a is unstable if company c and applicant a prefer each other to current matches.
- Unstable pair c-a could each improve by switching.

Stable matching: perfect matching with no unstable pairs.

Stable matching problem. Given the preference lists of n companies and n applicants, find a stable matching if one exists.

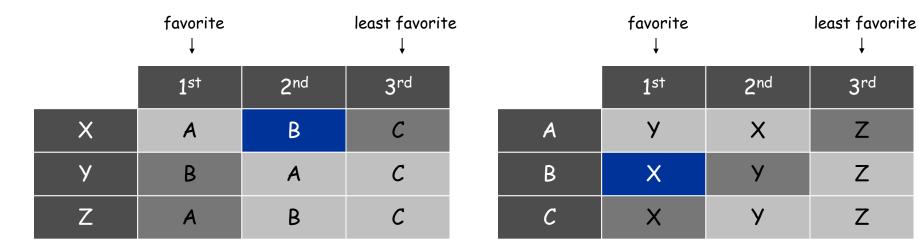




companies's Preference Profile

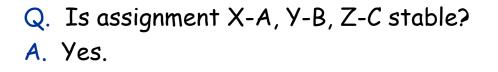
applicants's Preference Profile

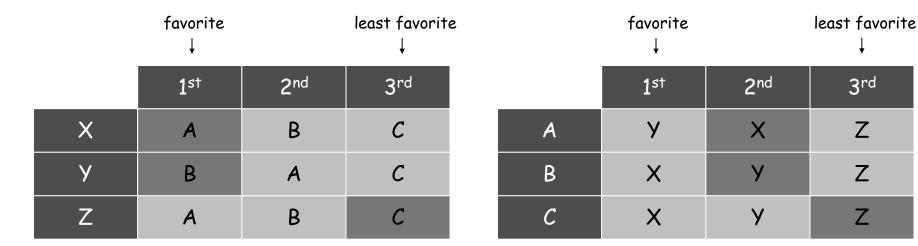
- Q. Is assignment X-C, Y-B, Z-A stable?
- A. No. B and X will defect.



companies's Preference Profile

applicants's Preference Profile





companies's Preference Profile

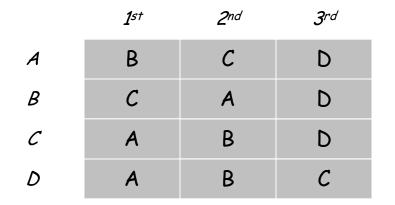
applicants's Preference Profile

Stable Roommate Problem

- Q. Do stable matchings always exist?
- A. Not obvious a priori.

Stable roommate problem.

- 2n people; each person ranks others from 1 to 2n-1.
- Assign roommate pairs so that no unstable pairs.



Observation. Stable matchings do not always exist for stable roommate problem.

Propose-And-Reject Algorithm

Propose-and-reject algorithm. [Gale-Shapley 1962] Intuitive method that guarantees to find a stable matching.

```
Initialize each person to be free.
while (some company is free and hasn't proposed to every
applicant) {
    Choose such a company x
    a = 1^{st} applicant on x's list to whom x has not yet
proposed
    if (a is free)
        assign x and a to each other
    else if (a prefers x to her current assignment y)
        assign a to x, and y to be free
    else
       a rejects x
```

Proof of Correctness: Termination

Observation 1. companies propose to applicants in decreasing order of preference.

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

Claim. Algorithm terminates after at most n² iterations of while loop. Pf. Each time through the while loop a company proposes to a new applicant. There are only n² possible proposals.

	1 ^{s†}	2 nd	3 rd	4 th	5 th		1 ^{s†}	2 nd	3 rd	4 th	5 th
V	A	В	С	D	E	A	W	Х	У	Z	V
W	В	С	D	А	E	В	X	У	Z	V	W
Х	С	D	А	В	E	С	У	Z	V	W	Х
У	D	А	В	С	E	D	Z	V	W	Х	У
Z	A	В	С	D	E	E	V	W	Х	У	Z

n(n-1) + 1 proposals required

Proof of Correctness: Perfection

- Claim. All companies and applicants get matched.
- Pf. (by contradiction)
 - Suppose, for sake of contradiction, that Z is not matched upon termination of algorithm.
 - Then some applicant, say A, is not matched upon termination.
 - By Observation 2, A was never proposed to.
 - But, Z proposes to everyone, since Z ends up unmatched.

Proof of Correctness: Stability

Claim. No unstable pairs.

- Pf. (by contradiction)
 - Suppose A-Z is an unstable pair: each prefers each other to partner in Gale-Shapley matching S*.

Case 1: Z never proposed to A.	order of preference	^{ng} S*
\Rightarrow Z prefers GS applicant to A.	×	A-Y
\Rightarrow A-Z is stable.		B-Z

- Case 2: Z proposed to A.
 - \Rightarrow A rejected Z (right away or later)
 - \Rightarrow A prefers her GS company to Z. \leftarrow applicants only trade up
 - \Rightarrow A-Z is stable.
- In either case A-Z is stable, a contradiction.

Summary

Stable matching problem. Given n companies and n applicants, and their preferences, find a stable matching if one exists.

Gale-Shapley algorithm. Guarantees to find a stable matching for any problem instance.

- Q. How to implement GS algorithm efficiently?
- Q. If there are multiple stable matchings, which one does GS find?

Efficient Implementation

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

Note: this is linear in the size of the input.

Representing companies and applicants.

- Assume companies are named 1, ..., n.
- Assume applicants are named 1', ..., n'.

Queues.

- Maintain a list of free companies, e.g., in a queue.
- Maintain two arrays applicant[c], and company[a].
 - set entry to \circ if unmatched
 - if c matched to a then applicant[c]=a and company[a]=c

companies proposing.

- For each company, maintain a list of applicants, ordered by preference.
- Maintain an array count[c] that counts the number of proposals made by company c.

Efficient Implementation

applicants rejecting/accepting.

- Does applicant $\tt a$ prefer company $\tt c$ to company $\tt c$ '?
- For each applicant, create inverse of preference list of companies.
- Constant time access for each query after O(n) preprocessing.

A	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
A	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

A prefers company 3 to 6 since inverse[3] < inverse[6]

2 7

Understanding the Solution

Q. For a given problem instance, there may be several stable matchings. Do all executions of Gale-Shapley yield the same stable matching? If so, which one?

An instance with two stable matchings.

- A-X, B-Y, C-Z.
- A-Y, B-X, C-Z.

	1 st	2 nd	3 rd
X	A	В	С
У	В	А	С
Z	А	В	С

	1 st	2 nd	3 rd
A	У	Х	Z
В	Х	У	Ζ
С	Х	У	Z

Understanding the Solution

Q. Do all executions of Gale-Shapley yield the same stable matching?

Def. company m is a valid partner of applicant w if there exists some stable matching in which they are matched.

1st

У

У

У

2nd

Ζ

Ζ

Х

3rd

Х

Х

Ζ

	1st	2 nd	3 rd	
X	А	С	В	A
У	А	В	С	В
Z	А	В	С	С

Q. Are X-A valid partners?

Understanding the Solution

Q. Do all executions of Gale-Shapley yield same stable matching?

Def. company c is valid partner of applicant a if exists some stable matching in which they are matched.

company-optimal assignment. Each company receives best valid partner.

Claim. All executions of GS yield company-optimal assignment, which is a stable matching!

- No reason a priori to believe that company-optimal assignment is a matching, let alone stable.
- Simultaneously best for every company.

company Optimality

Claim. GS matching S* is company-optimal.

- Pf. (by contradiction)
 - Suppose some company is paired with someone other than best partner. companies propose in decreasing order of preference ⇒ some company is rejected by valid partner.
 - Let Y be first such company, and let A be first valid applicant that rejects it.
 - Let S be a stable matching where A and Y are matched.
 - When Y is rejected, A forms (or reaffirms) engagement with a company, say Z, whom she prefers to Y.
 - Let B be Z's partner in S. B is a valid partner of Z.
 - Z matched to A and not yet rejected by any valid partner at the point when Y is rejected by A. Thus, Z prefers A to B.¹
 - But A prefers Z to Y.
 - Thus A-Z is unstable in S.

since this is first rejection by a valid partner of anyone

A-Y

B-Z

. . .

Stable Matching Summary

Stable matching problem. Given preference profiles of n companies and n applicants, find a stable matching.

no company and applicant prefer to be with each other than assigned partner

Gale-Shapley algorithm. Finds a stable matching in $O(n^2)$ time.

company-optimality. In version of GS where companies propose, each company receives best valid partner.

w is a valid partner of m if there exist some stable matching where m and w are paired

Q. Does company-optimality come at the expense of the applicants?

applicant Pessimality

applicant-pessimal assigncompaniest. Each applicant receives worst valid partner.

Claim. GS finds applicant-pessimal stable matching S*.

Pf.

- Suppose A-Z matched in S*, but Z is not worst valid partner for A.
- There exists stable matching S in which A is paired with a company, say Y, whom she likes less than Z.
- Let B be Z's partner in S. company-optimality
- Z prefers A to B.
- Thus, A-Z is an unstable in S. •

5	
A-Y	
B-Z	
•••	

Lessons Learned

Powerful ideas

- Isolate underlying structure of problem.
- Create useful and efficient algorithms.

Potentially deep social ramifications.

Moral: Be the one doing the proposing!