STABLE MATCHING PROBLEM

The Problem

Informally

Consider the problem of optimally match a set of applicants to a set of open positions

- ~~Application~~ Applicants to summer internships
- Applicants to graduate school
- Medical school graduate applicants to residency programs
- Eligible males wanting to marry eligible females

Seems like an easy problem. Why is this a hard problem in practice?

e.g. of students & jobs

The process is not self-enforcing - everyone is allowed to act in their self interest, then it risks breaking down.

Unstable Pair

Applicant x & hospital y are unstable if

- x prefers y to its assigned hospital
- y prefers x to one of its admitted students.

Stable Assignment

A stable assignment is one with no unstable pairs.

- This is a natural and desirable condition.
- Individual self-interest will prevent any applicant/hospital deal being made

Formulating the Problem

Formally,

Consider a set $M = \{m_1, \dots, m_n\}$ of n men and a set $W = \{w_1, \dots, w_n\}$ of n women.

- A matching S is a set of ordered pairs each from $M \times W$, such that each member of M and each member from W appears in at most one pair in S.
- A perfect matching S' is a matching such that each member of ~~W~~ M and each member of W appears in exactly one pair in S'.

(contd.)

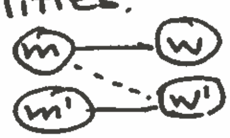
- Each man $m \in M$ ranks all of the women; m prefers w to w' if m ranks w higher than w'. We refer to the ordered ranking of m as his preference list.

- Each woman ranks all of the men in the same way

- An instability results when a perfect matching S' contains two pairs (m, w) and (m', w'), such that m prefers w' to w and w' prefers m to m'.

GOAL: A perfect set of marriages with no instabilities.

Example ①



m prefers w' to w
w' prefers m' to m
(m, w') is an instability w.r.t S

② Men's preference profile

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

Women's preference profile

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

Is the assignment X−C, Y−B, Z−A stable?

X−B would abandon their matches and get together.

KEY QUESTIONS ABOUT STABLE MARRIAGE

- Does there exist a stable matching for every set of preference list?
- Given a set of preference lists, can we efficiently construct a stable matching if there is one?

DESIGNING THE ALGORITHM

Initially, everyone is unmarried.

- Suppose an unmarried man n chooses the women w who ranks highest on his preference list and proposes to her.
- w can wait for someone higher on the her list may propose to her in the future. But, w may never receive a proposal from someone as highly as m.
- Idea, have pair (m, w) enter an ~~[illegible]~~ intermediate state - engagement.
- The next available man m chooses the highest-ranked woman w to whom he has not proposed yet. and proposes to her. If she's free, they get engaged If she's engaged to some other man m', she determines which of m, m' rank higher on her list; she becomes engaged to this man & the other becomes free
- Repeat till no one is free; All engagements are considered final & the perfect matching is returned.

GALE-SHAPLEY ALGORITHM

```
Gale-Shapley()
  Initially all m ∈ M and w ∈ W are free
  while (∃m who is free and hasn't
            proposed to every w ∈ W)
      do choose such a man m
          Let w be the highest ranked in
              m's preference list to whom m
              has not proposed yet
          if w is free
             then (m, w) become engaged
             else w is current engaged to m'
                  if w prefers m' to m
                     then m remains free
                     else w prefers m to m'
                          (m, w) become engaged
                          m' becomes free
  return the set S of engaged pairs
```

Analysing the algorithm

~~Some~~

Some Axioms

- w remains engaged from the point at which she receives her first proposal.
- The sequence of partners w/ which w is engaged to gets increasingly better in terms of her preference list.
- The sequence of women to whom m proposes get increasingly worse in terms of his preference list.

Some Observations

- Men propose in decreasing order of preference
- Once Women is match, she never becomes unmatched. She only trades up.

TERMINATION

THEOREM

The G-S algorithm terminates after at most n^2 iterations of the while loop.

Side Note: To find the upper bound of the run time (or # of iterations), we need some notion of progress.
We need a ~~measure~~ way to say at each iteration we move closer to termination?
For G-S algorithm what is a good measure of progress?
Hint: need some metric that increases (or decreases) at every iteration
- # of free individuals?
- the # of engaged couples?
- the # of proposals made?

Proof: Each iteration consists of one man proposing to a woman he has never proposed to before.
Let $P(t)$ denote the set of pairs (m, w) such that m has proposed to w by the end of iteration t.
$\forall t \quad P(t+1) > P(t)$ $\because$ @ each iteration a proposal is made
There are only n^2 possible pairs of men & women (n^2 possible proposals) in total.
So $P(t)$ can increase at most n^2 times.
$\therefore$ there can be at most n^2 iterations

Towards a perfect matching

Theorem: If m is free at some point in the execution of the algorithm, then there is a woman to whom he has not yet proposed.

<u>Proof:</u> Suppose, at some point, m is free but has already proposed to every woman.

~~Then m is free~~

Then every woman must be engaged.
($\because$ once a woman becomes engage, she remains engaged or trades up)

The set of engaged pairs forms a matching, there must be n engaged men. But there are a total of n men and m is not engaged.
This a ~~contration~~ contradiction.

Theorem: The set S returned at termination is a perfect matching.

Side Note { Recall definition of perfect matching

Proof: Suppose the algorithm terminates with a free man m.
Then m must have proposed to every woman (other wise the loop would still be active, and we wouldn't be at termination.) But this contradicts the previous theorem which stated that there are cannot be a free man that has proposed to every woman.

CORRECTNESS

Theorem: Consider an execution of the G-S algorithm that returns a set of Pairs S. The set S is a stable matching

Proof:

Assume there is an instability.
Then there exist two pairs (m, w) and (m', w') in S such that m prefers w' to w and w' prefers m to m'.
During execution m's last proposal must have been to w.
Had m proposed to w' at an earlier time?
If no, then w ~~occurred~~ must occur higher on m's preference list than w', contradicting our assumption that m prefers w' to w.
If yes, then he was rejected by w' in favor for some other man m'' whom w' prefers to m. m' is the final partners of w', so either m'' = m' or w' prefers her final partner m' to m''; eiter way this contradicts our assumption the w' prefers m to m'.

It follows that S is a stable matching

Exercise

Prove that if all men have the same list of preferences, and all the women have the same list of preferences, then only one stable matching exists.

(Solution in slides)

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

- How do we implement the Gale-Shapley algorithm efficiently?
- If there are multiple stable matchings, which one does the algorithm find?

Implementation

We can describe an $O(n^2)$ implementation

Representing Men & Women

- Assume men are named $1 \ldots n$
  women are named $1' \ldots n'$

Engagements

- Maintain a list of free men in a queue
- Maintain ~~a~~ two arrays of length n
  wife[m] and husband[w]
  - Set entry to 0 if unmatched
  - if m is matched to w, then
    wife[m] = w
    husband[w] = m

Proposals

For each man, maintain a list of women ~~sorted~~ ordered by preference

- Maintain an array count[m], the number of proposals made by m.

Women Rejecting / Accepting

- For each woman, create an inverse preference list.
- Allows constant time queries:

  A woman prefers m to m' if inverse[m] < inverse[m']

Proposal Process

- The first free man m ~~is~~ in the queue proposes to the women at the front of his preference list, w.
- He increments counts[m] and removes w from his preference list
- w accepts the proposal if she is unengaged or prefers m to her current match
- if w accepts, her former match goes back on the queue of men; otherwise m proposes to his next favorite.

Understanding the Solution

Multiple stable matchings

m prefers w to w'

m' prefers w' to w

w prefers m' to m

w' prefers m to m'

In any execution of G-S algorithm m becomes engaged to w & m' becomes engaged to w'.

Another possible stable matching (m', w), (m, w').
This possibility is not attaineble in the version of G-S algorithm where men propose

w is a valid partner of a man m if there is a stable matching that contains the pair (m, w)

w is the best valid partner of a man m if w is a valid partner of m and no woman whom m ranks higher than w is a valid partner of his.

A man-optimal ~~solution~~ assignment is one in which every man receives the best valid partner.

Claim 1: All executions of G-S yield man-optimal assignments, which is a stable matching

Claim 2: All executions of GS yield woman-pessimal assignment, which is a stable matching. (i.e. each woman receives the worst possible valid partner).

Claim 1: All executions of GS yield man-optimal assignment, which is a stable matching.

Proof: • Suppose a man is matched with someone other than best valid partner.

• Men propose in decreasing order of preference
⇒ Some man is rejected by valid partner during GS.

• Let m be such a man and let w be the first valid ~~w~~ women that rejects him.

• Let S ~~be~~ be a stable matching where (m, w) are matched

• ~~Let~~ When ~~w~~ m is rejected by w in GS, w forms or continues engagement with m'
⇒ w prefers m' to m

• Let w' be a partner ~~in~~ of m' in S

because this is a first rejection by a valid partner
• m' has not been rejected by any valid partner at the point when m is rejected by w.

• Thus ~~m~~ m' has not yet proposed to ~~w~~ w' when he proposes to ~~w~~ w
⇒ m' prefers w to w'

Thus ~~w~~ (m', w) is unstable in S, a contradiction.

VARIATIONS

- Consider the stable roommate problem 2n people, each rank the others from 1 to 2n-1. The goal is to assign a roommate pair so that ~~is not~~ none are unstable.

Final thoughts

Steps in algorithm design

- Formulate the problem precisely.
- Design an algorithm for the problem
- Prove the algorithm to be correct
- Give a bound on the algorithm's running time

Design Techniques

We'll explore algorithm design by enumerating a set of design techniques And learning to recognize problems that likely belong to one class or another.