CSE 421
Introduction to Algorithms

Lecture 2: Stable Matching Properties, Overview

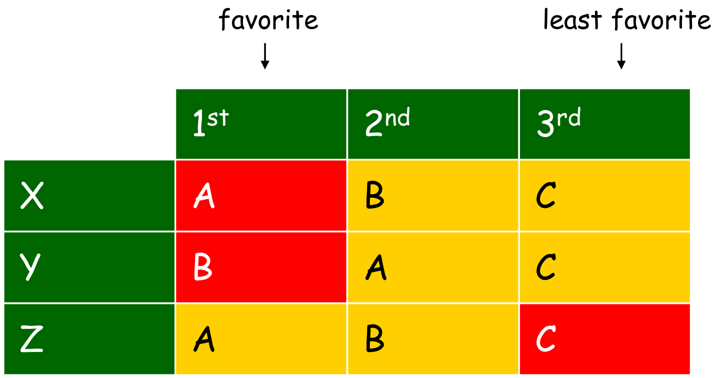
Administrative

- Homework 1
  - Posted on course website
  - Due next Wednesday
  - Available for Gradescope submission starting tomorrow
- Edstem discussion:
  - Public: Discuss everything course-related except *solutions* to current homework or anything about on-going exams
  - Use private posts to staff for other kinds of questions:
- Office hours start today (except for one of them)

Stable Matching Problem

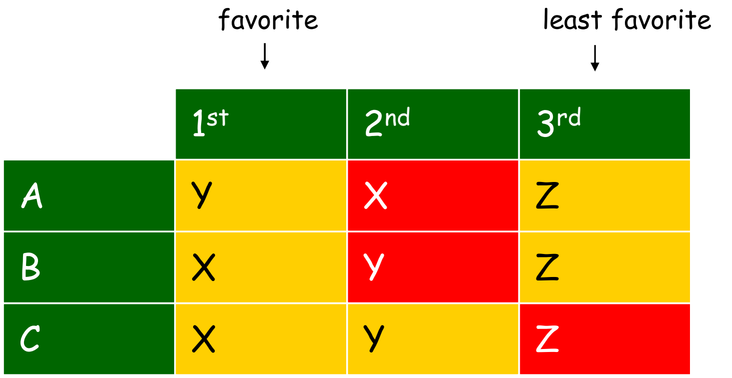
Goal: Given two groups of n people each, find a “stable matching”.

- Participants rate members from opposite group.
- Each person lists members from the other group in order of preference from best to worst.
- Matching: Each person matched with exactly one person in the other group.
- Stable: No pair of people both prefer to be with each other rather than with their assigned partner



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

*Group **P** Preference Profile*



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

*Group **R** Preference Profile*

Propose-And-Reject Algorithm

Propose-and-reject algorithm: [Gale-Shapley 1962]

Intuitive method that guarantees to find a stable matching.

- Members of one group ***P*** make *proposals*, the other group ***R*** *receives* proposals

```
Initialize each person to be free.
while (some p in P is free) {
    Choose some free p in P
    r = 1st person on p's preference list to whom p has not yet proposed
    if (r is free)
        tentatively match (p,r)   //p and r both engaged, no longer free
    else if (r prefers p to current tentative match p’)
        replace (p’,r) by (p,r)   //p now engaged, p’ now free
    else
        r rejects p
}
```

Gale-Shapley Algorithm

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

⇒ a stable matching always exists!

Q: How do we implement the Gale-Shapley algorithm efficiently? ←

Q: If there are multiple stable matchings, which one(s) does it find?

Implementation for Stable Matching

- Input size
  - $N = 2n^2$ words

    table entries
    - $2n$ people each with a preference list of length n
  - $2n^2 \log n$ bits
    - specifying an ordering for each preference list takes $n \log n$ bits

- Brute force algorithm
  - Try all $n!$ possible matchings

- Gale-Shapley Algorithm
  - n^2 iterations. Can have constant time per iteration as follows …

Efficient Implementation

How do we get the $O(n^2)$ time implementation?

Input: Representing members of the two groups P and R and their preferences:

- Assume elements of P (proposers) are numbered $1, \ldots, n$.
- Assume elements of R (receivers) are numbered $1', \ldots, n'$.
- For each proposer, a list/array of the n receivers, ordered by preference.
- For each receiver, a list/array of the n proposers, ordered by preference.

(i, j')

The matching:

- Maintain two arrays **match**$[p]$, and **match'**$[r]$.
  - set entry to **0** if free
  - if p matched to r then **match**$[p]=r$ and **match'**$[r]=p$

$\text{match}[i]=j$

$\text{match}'(j)=i$

Making proposals:

- Maintain a list of free proposers, e.g., in a queue.
- Maintain an array **count**$[p]$ that counts the number of proposals already made by proposer p.

Efficient Implementation

Rejecting/accepting proposals:

- Does receiver r prefer proposer p to proposer p'?
  - Using original preference list would be slow
- For each receiver, create *inverse* of preference list of proposers.
- Constant time access for each query after $O(n)$ preprocessing per receiver. $O(n^2)$ total preprocessing cost.

Creating all inverse lists $O(n^2)$

r	1st	2nd	3rd	4th	5th	6th	7th	8th
pref	8	3	7	1	4	5	6	2

Proposer 3 or proposer 6 ?

Total $O(n^2)$

r	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
```

r prefers proposer 3 to 6

since `inverse`[3] = 2 < 7 = `inverse`[6]

Understanding the Solution

Q: For a given problem input, 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 (see section for more).
  - (A,X), (B,Y), (C,Z).
  - (A,Y), (B,X), (C,Z).

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

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

Understanding the Solution

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

Def: p in P and r in R are valid partners iff there is some stable matching containing (p, r)

Def: Proposer-optimal assignment: Each proposer is matched with their **best** valid partner
(their most preferred among all of their valid partners)

Claim: All executions of Gale-Shapley yield a proposer-optimal assignment!

- Not obvious that proposer-optimal assignment is perfect, let alone stable
- Simultaneously best for each and every proposer

Proposer Optimality

Claim: Any Gale-Shapley matching M is proposer-optimal.

Proof: (By contradiction)

Want to prove that these rejections never happen

Suppose that there are some proposers in M not matched to their best valid partners

$\Rightarrow$ Each must have been rejected by a valid partner, since they propose in decreasing preference order.

- Among all of these, choose the first time a proposer p is rejected by a valid partner.
- Call that rejecting valid partner r and let p' be the proposer who r prefers to p s.t either r was engaged to p' or p' replaced p when that rejection happened.

Let M' be a stable matching containing (p, r).

Must exist since (p, r) is valid

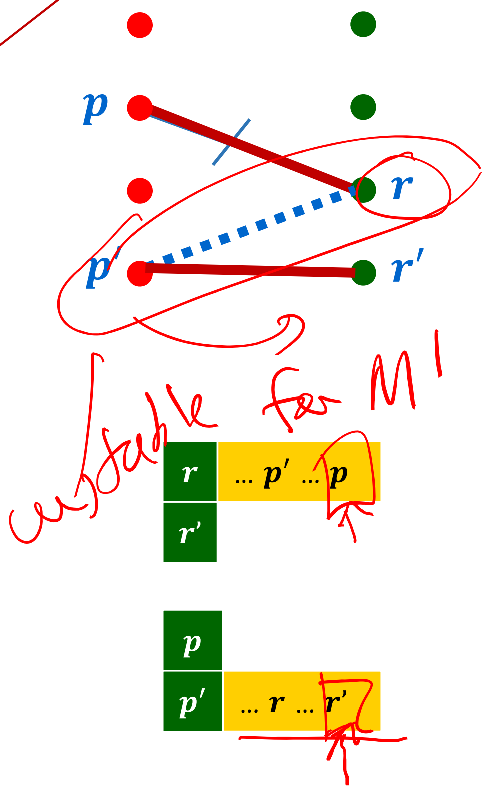
Let r' be the partner of p' in M'. This $\Rightarrow$ (p', r') are valid partners.

- Since r rejecting p was the first rejection by a valid partner, when that happened, r' had not rejected p' since (p', r') are valid partners, so p' hadn't proposed to r'.
- $\Rightarrow$ p' prefers r to r'

But we already said that r prefers p' to p.

$\Rightarrow$ p'-r is unstable in M'.

$\Rightarrow$ M' is not stable. Contradiction ■



unstable for M'

r	… p' … p
r'	

p	
p'	… r … r'

Non-obvious consequence of proposer optimality

- That proof worked no matter which free proposer was selected in each step!

- There is just one proposer-optimal stable matching

So all the orders of selecting free proposers in the Gale-Shapley algorithn yield the same stable matching!

Stable Matching: Summary so far

Stable matching problem: Given preference profiles of two groups of n people, find a stable matching between them.

No pair of people both prefer to be with each rather than with their assigned partner

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

Proposer-optimality: In Gale-Shapley matching, each proposer gets best partner possible among all stable matchings

Q: Does proposer-optimality come at the expense of the other side?

Receiver Pessimality

Receiver-pessimal assignment: Each receiver is gets their worst valid partner.

Claim: Gale-Shapley produces a receiver-pessimal stable matching M.

Proof: (By contradiction again)

Suppose (p, r) matched in M, but p is not worst valid partner for r.

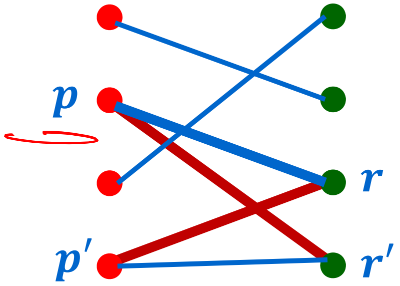
$\Rightarrow$ there exists some other stable matching M' in which r is paired with a proposer, say p', whom r likes less than p.

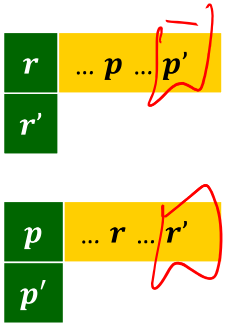
Let r' be the partner of p in M'.

Since M is proposer-optimal, p prefers r to r'

$\Rightarrow$ p-r is an unstable in M'

$\Rightarrow$ M' is not stable. ■





Extensions: Matching Residents to Hospitals

Original: Proposers ≈ hospitals, Receivers ≈ med school residents.

Variant 1: Some participants declare others as unacceptable.

e.g. resident r unwilling to work in Cleveland

Variant 2: Unequal number of proposers and receivers.

e.g. hospital h wants to hire 3 residents

Variant 3: Limit on # of pairs person participates in can be >1.

Def: Matching M is unstable if there is a hospital h and resident r such that:

- h and r are acceptable to each other; and
- either r is unmatched, or r prefers h to her assigned hospital; and
- either h does not have all its places filled, or h prefers r to at least one of its assigned residents.

Application: Matching Residents to Hospitals

NRMP: (National Resident Matching Program)

- Original use just after WWII
- Ides of March: 23,000+ residents legally bound by the outcome
- Pre-1995 NRMP favored hospitals (they proposed)
- Changed in 1995 to favor residents (after a lawsuit)

Rural hospital dilemma:

- Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.
- Rural hospitals were under-subscribed in NRMP matching.
- Q: Find stable matching that benefits "rural hospitals"?

Rural hospital theorem: Rural hospitals get exactly same residents in every stable matching!

The original paper

The title of the 1962 Gale-Shapley paper was “College Admissions and the Stability of Marriage”

- The propose-and-reject algorithm was clearly inspired by Western traditions of proposals
- The fact that the result is much more advantageous to the proposing side even in this non-binding scenario took some time to be appreciated

Though Gale had died by then, Shapley and Roth shared the 2012 Nobel Prize in Economic Sciences for their work on stable assignments.

Lessons Learned

- Powerful ideas learned in course.
  - Isolate underlying structure of problem.
  - Create useful and efficient algorithms.

- Potentially deep social ramifications. [legal disclaimer]

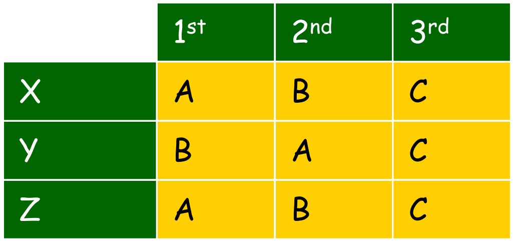
- Technique: sometimes useful to consider the first time something bad might happen for an algorithm in order to rule it out.

Deceit: Machiavelli Meets Gale-Shapley

Q: Can there be an incentive to misrepresent your preference profile?

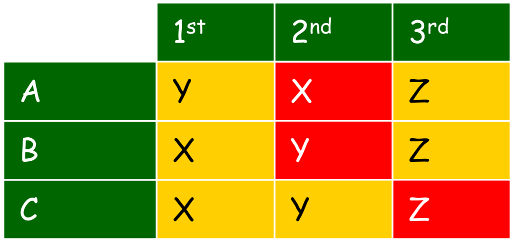
- Assuming you know that propose-and-reject algorithm will be run and who will be proposers.
- And assuming that you know the preference profiles of all other participants.

Fact: No, for proposers. Yes, for some receivers. No mechanism can guarantee a stable matching and be cheatproof.



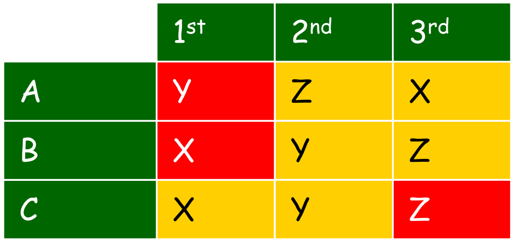
	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

*Group **P** Preference List*



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

*Group **R** True Preference List*



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

A pretends to prefer Z to X

Introduction to Algorithms

- Overview

Measuring efficiency: The RAM model

- RAM = Random Access Machine

- Time ≈ # of instructions executed in an ideal assembly language
  - each simple operation (+,*,-,=,if,call) takes one time step
  - each memory access takes one time step

Complexity analysis

- Problem size n
  - **Worst-case complexity**:

    maximum # steps algorithm takes on any input of size n

  - **Best-case complexity:**

    minimum # steps algorithm takes on any input of size n

  - **Average-case complexity**:

    average # steps algorithm takes on inputs of size n

Complexity

- The complexity of an algorithm associates a number $T(n)$, the worst/average-case/best time the algorithm takes, with each problem size **n**.

- Mathematically,
  - T is a function that maps positive integers giving problem size to positive real numbers giving number of steps.

- Sometimes we have more than one size parameter
  - e.g. n=# of vertices, m=# of edges in a graph.

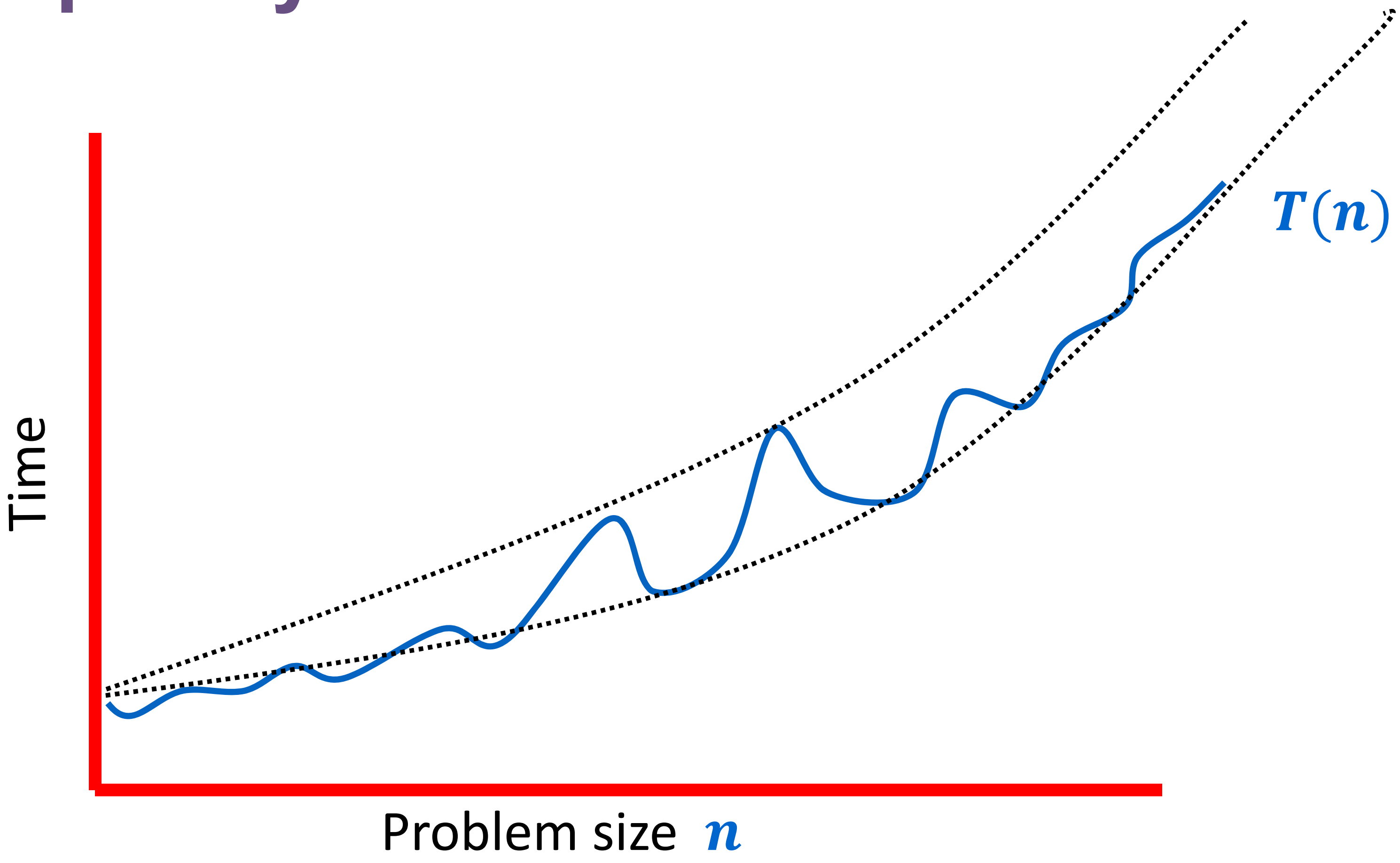
Efficient = Polynomial Time

- Polynomial time
  - Running time $T(n) \le \underline{cn^k} + d$ for some $c, d, k \ge 0$

- Why polynomial time?
  - If problem size grows by at most a constant factor then so does the running time
    - e.g. $T(2n) \le c\,(2n)^{\boxed{k}} + d = 2^k cn^k + d \le 2^k\left(cn^k + d\right) = \underline{2^k}\, T(n)$
    - polynomial-time is exactly the set of running times that have this property

  - Typical running times are small degree polynomials, mostly less than n^3, at worst n^6, not n^{100}

Complexity



O-notation etc

- Given two positive functions f and g ≤
  - $f(n)$ is $O(g(n))$ iff there is a constant $c > 0$ so that $f(n)$ is eventually always $\leq c \cdot g(n)$ ✓
  - $f(n)$ is $o(g(n))$ iff the ratio $f(n)/g(n)$ goes to 0 as n gets large
  - $f(n)$ is $\Omega(g(n))$ iff there is a constant $\varepsilon > 0$ so that $f(n) \geq \varepsilon \cdot g(n)$ for infinitely many values of n ≥
  - $f(n)$ is $\Theta(g(n))$ iff $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$ =

Note: The definition of $f(n)$ is $\Omega(g(n))$ is the same as "$f(n)$ is **not** $o(g(n))$ "