- Welcome!

- OH tomorrow (Fri) : 10-11am
  - Zoom link will be available on Canvas & on Piazza

- Hws 0 and 1 will be released today.

- Course policies

Stable Matching

Input:
- n people, P
- n pets, T

- each person ranks every pet in T in decreasing order of their preferences (no ties)

- each pet ranks every person in P in increasing order of their preferences (no ties)
Objective: To design an algorithm that outputs a stable matching.

Example: $(t_1, t_1, t_3) P_1 \rightarrow (P_1, P_2, P_3)$

$(t_3, t_2, t_1) P_2 \rightarrow (P_2, P_1, P_3)$

$(t_1, t_1, t_3) P_3 \rightarrow (P_1, P_2, P_3)$

Definition: A matching $M$ in the given instance is stable iff it is a perfect matching (every person is paired with exactly one partner and every partner is paired with exactly one person) and there is no pair $(j, t)$, where $j \in P$ and $t \in T$ s.t. $j$ and $t$
prefer each other over their existing partners.

Q: Does every instance always have a stable matching?
Ans: Yes ✓

Q: Can an instance have more than one stable matching?
Ans: No ✓ Yes ✓

\[ (t, t') \quad \text{p} \quad \text{t} \quad (p, p') \quad \text{Yes} \]
\[ \quad \rightarrow \]
\[ (t, t') \quad \text{p'} \quad \text{t'} \quad (p, p') \quad \text{No} \]

\[ (t, t') \quad \text{p} \quad \text{t} \quad (p', p) \quad \text{Yes} \]
(t', t) \quad p' \quad \text{match} \quad t' \quad (p, p') \quad \text{YES}

Alg.

1. Fix an ordering of the people in $P$.
2. for each permutation $\pi$ of people in $T$ do
3. \quad match $p_i$ with $\pi(i)$
4. \quad if there is no instability then
5. \quad \quad output the matching
6. output "No stable matching".

\[
p_1, p_2, \ldots, p_n
t_1, t' \text{ etc.}
\]

The above alg. takes $n!$ iterations in
the worst case. For example, if $n = 30$, our
diy will take $> 10^{25}$ years.

$$P_1 \sim \sim \sim \sim \sim \sim t_1$$

$$P_2 \sim \sim \sim \sim \sim t_2$$

$$P_3 \sim \sim \sim \sim \sim t_3 \ (\ldots P_3, \ldots P_5)$$

$$P_4 \sim \sim \sim \sim \sim t_4$$

$$P_5 \sim \sim \sim \sim \sim t_5 \ (t_3, t_1, t_5)$$

Gale-Shapley (GS)

1. Initially all people & pets are free.
2. While there is a free person $p$ who has
not yet proposed to every pet do
3. \( t \leftarrow \text{highest ranked pet on } p\text{'}s \text{ list; whom } p \text{ has not yet proposed to.} \)
4. if \( t \) is free then
   \((p, t)\) becomes a pair.
5. else if \((p', t)\) exists then
6. if \( t \) prefers \( p \) over \( p' \) then
7. \((p, t)\) becomes a pair
8. \( p' \) becomes free
9. Output all pairs

Q: Does the above alg. always terminate?
Ans: The alg. may go into an infinite loop.
\[ n^2 \text{ iterations} \]

- each person makes \( \leq n \) proposals
  & there are \( n \) people in total.

**Lemma 1**: Once a pit receives their first proposal, they will always remain "engaged" and as the alg. progresses, their partners can only get better.

**Lemma 2**: In the GS alg. each person is paired with at most one pit and each pit is paired with at most one person.

**Lemma**: GS alg outputs a perfect matching. (Every pit & every person are matched)

**Proof**: Assume for contradiction that
Some person say $p$ is free when the alg. ended. $p$ must have proposed to all pets (because the alg. has ended).

No free pets. Why? Because since every pet is proposed to, they are engaged by Lemma 1. Since $n$ pets are matched to $\leq n-1$ people, by PHP, some person must be matched to two pets, contradicting Lemma 2.

Thus: GS alg outputs a stable matching.

Proof: Assume for contradiction that in the output of the GS alg, there is an
unstable pair, say \((p, t')\).

\[ (\ldots t', \ldots, t, \ldots) \quad \text{unstable} \quad t \]

\[ p \quad \text{unstable} \quad t' \]

\[ p' \quad (\ldots p, \ldots, p', \ldots) \]

- \(p\) will propose to \(t'\) before proposing \(t\).

- Since \(t'\)'s partners can only get better (Lemma 1), it must be that \(p'\) is ranked higher than \(p\) on \(t'\)'s list, a contradiction.

Q. Does different order of proposals yield different stable matchings?
Ans: No? Yes?

**Defn:**

\[
\text{valid}(p) = \{ t \in T \mid (p, t) \text{ is a pair in some stable matching} \}
\]

\[
\text{Best}(p) = t \quad \text{iff}
\]

- \( t \in \text{valid}(p) \)

- no pet \( t' \) ranked higher than \( t \) on \( p's \) list belongs to \( \text{valid}(p) \).

In other words, \( t \) is the highest ranked valid partner of \( p \).

\[
S^* = \{(p, \text{Best}(p)) \mid p \in P \}
\]

Thus: All executions of the GS alg.
result in $S^*$. 

**Proof:** Assume for contradiction that some execution $E$ of the GS algorithm some people are rejected by their best valid partners. Among all such people, let $p$ have the honor of being the first person to be rejected by $\text{Best}(p) \neq t$. Why did $t$ reject $p$?

Because of $p'.

Since $t \in \text{valid}(p)$, by defn., there must be a stable matching $S$ that contains $(p,t)$ as a pair.
- \( p' \) proposes to \( t' \) before proposing to \( t \).
- \( p' \) gets rejected by \( t' \).
- Note that \( t' \notin \text{valid}(p') \).
- \( p \) is not the first process to be
reinitialized by valid(•).

**Def:**

valid(t) = \{ p \mid (p, t) \text{ is a pair in some stable matching} \}.

\text{Worst}(t) = \exists p \text{ s.t.}
- p \in \text{valid}(t)
- no person ranked lower than p in t’s list \in \text{valid}(t).

\[ t \leftarrow (\ldots, [p], \ldots) \]

\text{not rank 1 valid}
Thus: As always, always outputs $(\text{worst}(t), t)$. 