Announcement:

① Quiz this Friday (in class)
② Midterm next Wednesday
③ See policy email from Andy.

The Gale-Shapley Algorithm continued.

Recall the Algorithm:

Input: Preference rankings by all children and puppies

Iteration: Each puppy chooses the highest child on its list who hasn't rejected it.

```
    if each puppy chooses a different child:
        Stop and use the resulting matching
    else each child says maybe to the most attractive proposal
    received and rejects all the others. Repeat iteration.
```

Thm 3.2.18: The Gale-Shapley Algorithm always produce a stable matching.

(No unstable pairs (x,y) st y is higher than x's match on x's list and vice versa)

pf: Need 3 things: ① Algorithm terminates
② result is a matching
③ no unstable pairs

①: There are n^2 total possible pairs (child, puppy), each iteration of Algorithm decreases that number since either puppies get rejected or a matching is formed.

The Algorithm stops when the # becomes 0.

② : Suppose not, we would have a child who has rejected all remaining unmatched puppies.

This is impossible since if a child has rejected a puppy, it means the child says "maybe" to some other puppy. After that, the child will keep saying "maybe" to the same puppy until either

a) the child says "yes" to the "maybe" puppy

or

b) some other puppy higher on the list chooses the child, in which case the child says "maybe" to new puppy.

Either way, the child says "maybe" or "yes" to a puppy. Impossible to reject all.

③. Suppose there's an unstable pair

$$\begin{tikzcd} & \text{Child} & \text{puppy} & \\ (b > a) & x \arrow[r, leftrightarrow] & a & \\ & y \arrow[r, leftrightarrow] & b & (x > y) \end{tikzcd}$$

Q: Has b chosen x ?

If yes, then x either said "yes" to b (didn't happen)
or said "maybe" to some puppy higher than b.

Notice that the "maybe" puppy will only rise in x's preference list and thus impossible for x to match w/ a

If No, then a must have got to x first and matched. In this scenario, b chooses y, which won't happen before choosing x Impossible again.

□

Example of Algorithm

Child Puppy

$(d>b>c>a)$	w	a	$(w>x>y>z)$
$(c>a>b>d)$	x	b	$(w>y>x>z)$
$(a>b>c>d)$	y	c	$(y>z>w>x)$
$(b>c>d>a)$	z	d	$(y>x>z>w)$

First iteration: a, b choose w; c, d choose y.

w rejects a ; maybe b ; y rejects d. maybe c

Second iteration: b choose w ; a, d choose x , c choose y

x rejects d maybe a

Third iteration: $b \leftrightarrow w$; $a \leftrightarrow x$, $c \leftrightarrow y$, $d \leftrightarrow z$ matched !

If children choose puppies, then

$$w \leftrightarrow \begin{array}{c} d \\ \vee \\ b \end{array} \qquad x \leftrightarrow \begin{array}{c} c \\ \vee \\ a \end{array} \qquad y \leftrightarrow \begin{array}{c} a \\ \vee \\ c \end{array} \qquad z \leftrightarrow \begin{array}{c} b \\ \vee \\ d \end{array}$$

Children ends up with pupples higher on their lists !

What does this tell us ?

Go after your most preferable choice is better than waiting to be chosen.

3.3 Matchings in General Graphs.

Recall Def: A perfect matching of G is a spanning subgraph such that each vertex has $\deg = 1$.

Q: when does a graph G have a perfect matching ?

Need one more definition:

Def: For $S \subseteq \underset{\uparrow \atop \text{vertices}}{V(G)}$, define $o(G-S)$ to be the # of odd components of $G-S$.

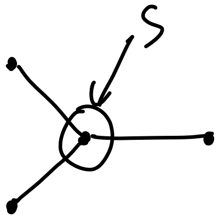
First Observation: If $|V(G)|$ is odd, impossible to have a perfect matching.

Thm 3.3.3 (Tutte, 1947)

G has a perfect matching $\Leftrightarrow$ $o(G-S) \leq |S|$ for all $S \subseteq V(G)$

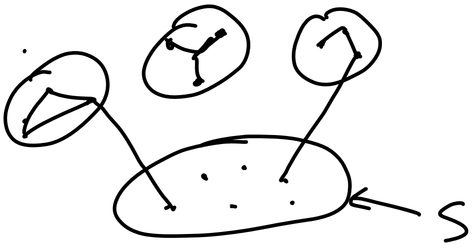
side note: First observation is the case $S = \phi$.

Example:



$o(G-S) = 3 > 1 = |S|$

pf: $\Rightarrow$:



Each odd component must be matched to a diffecent vertex in S.

$\Leftarrow$: When we add an edge to G, $o(G-S)$ does not increase.

Moreover, if $G' = G + e$, G' has no perfect matching $\Rightarrow$ G has no perfect matching.

Therefore if the statement fails, can find G s.t

① $o(G-S) \le |S|$ for all $S \subseteq V(G)$

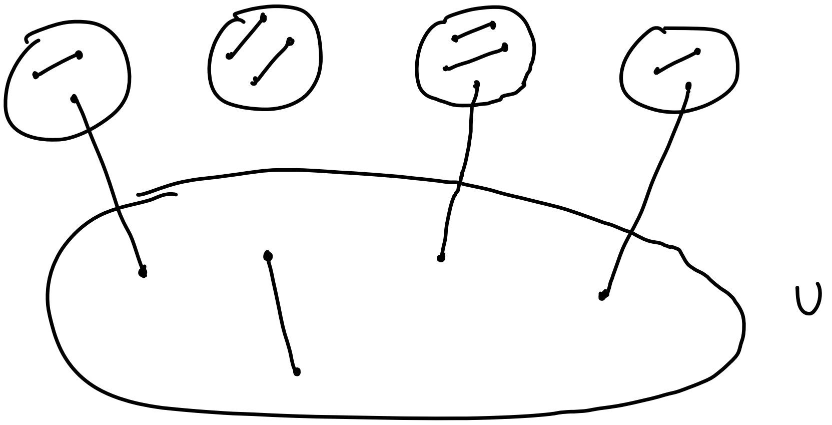
② G has no perfect matching

③ Adding any edge would have a perfect matching.

We'll show that G actually has a perfect matching.

Let $U \subseteq V(G)$ be the set of vertices w/ deg $|V(G)|-1$.

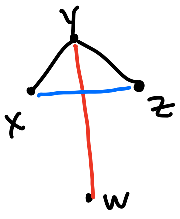
Case 1: $G-U$ consists of disjoint union of complete graphs.



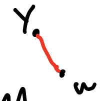
A matching is guaranteed since each component of $G-U$ is complete graph. We can get a matching of all / all but 1 vertices in the component. The single
(even / odd)
vertex can be matched with anyone in U.

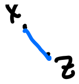
Case 2: $G-U$ is not disjoint union of complete graphs.

In a component, we can find x, z not connected but both connected to y.



Since $y \notin U$, can find w s.t. $(w,y) \notin E(G)$

Let $G' = G +$ 

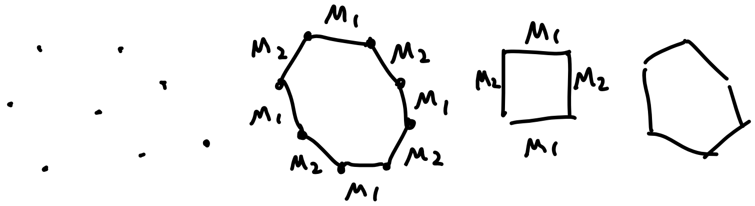
, $G'' = G +$ 

, by hypthesis ③, G', G'' has perfect matchings M_1 , M_2 .

Let $F = M_1 \triangle M_2$ be the symmetric difference. In particular, xz and $yw \in F$.

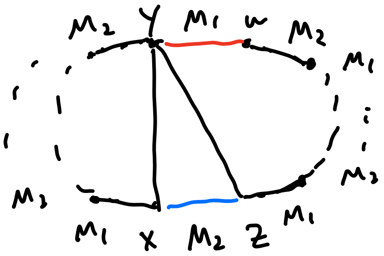
$$F = \{e \in M_1 : e \notin M_2\} \cup \{e \in M_2 : e \notin M_1\}.$$

Since every vertep in M_1 or M_2 has deg 1, every vertex in F has deg 0 or 2.



If xz, yw in different cycles, choose M_1 in C_1 and M_2 in others.
C_1 C_2

If xz, yw in same cycle.



we use yz and M_2's in the right half and M_1's in the left half to get a perfect matching without using xz or yw

For the other cycles use M_1 or M_2

□