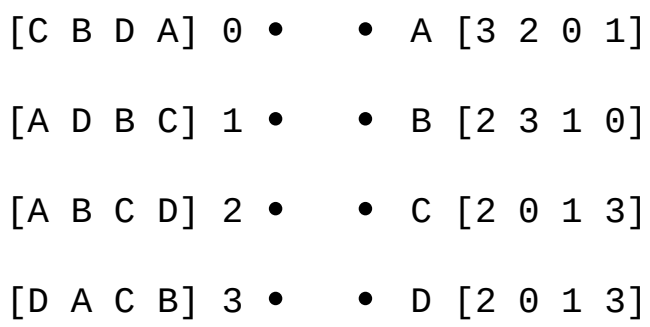
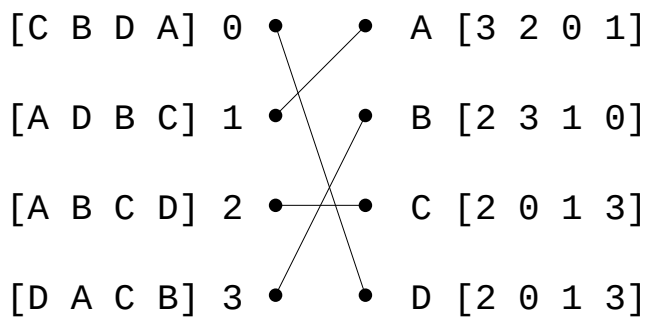
bipartite preference and stable matchings

- https://en.wikipedia.org/wiki/Stable_marriage_problem

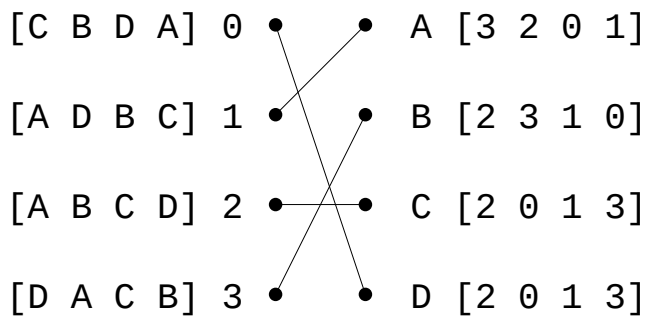


- ***bipartite preference system***
  - **hospitals** $H = \{0, 1, 2, 3\}$ **residents** $R = \{A, B, C, D\}$
  - **for each hospital, a *total preference order* of residents**
    e.g. hospital 0 **prefers** $C > B > D > A$
  - **for each resident, a TPO of hospitals**
    e.g. resident A **prefers** $3 > 2 > 0 > 1$
- **matching** $m : H \leftrightarrow R$

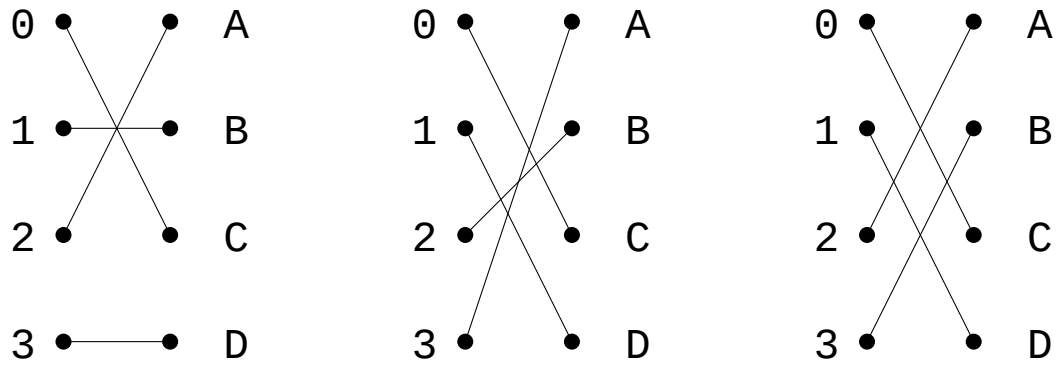


- **in** m**,** $\{2, A\}$ ***unhappy*: prefer each other to their matches**
  2 **prefers** A **to** $m(2) = C$ A **prefers** 2 **to** $m^{-1}(A) = 1$

- m ***unstable*: has unhappy couple**



- **which matchings here are unstable?**



(use above preferences. answers at bottom)

- **problem: find stable matching, if one exists**
- **input: preference system, n hospitals, n residents**

m0 unstable (1,D) **m1 stable** **m2 unstable (3,A)**

- **algorithm 0: brute force**
- **for each possible matching (there are n!):**
  - **check whether matching stable in $O(n^3 \lg n)$ time (how?)**
    - **for each hospital h (there are n),**
    - **for each resident r (n),**
    - **check: h prefers r to $m(h)$?**
    - **time $O(n \lg n)$**
    - **check: r prefers h to $m^{-1}(h)$?**
    - **time $O(n \lg n)$**
- **worstcase runtime $O(n!\ n^3 \lg n)$**

- **this algorithm: can we do better?**
- **this pseudocode: can we implement more efficiently?**
- **yes: we can use a preference lookup table**
- **worstcase runtime $O(n!\ n^2 \lg n)$**

- **brute force stop-when-stable WC runtime in $\Omega(n!)$**
- **why?**
- **exist instances with only 1 stable matching, e.g.**
- **0-A are each other's most preferred**

  1-B " "

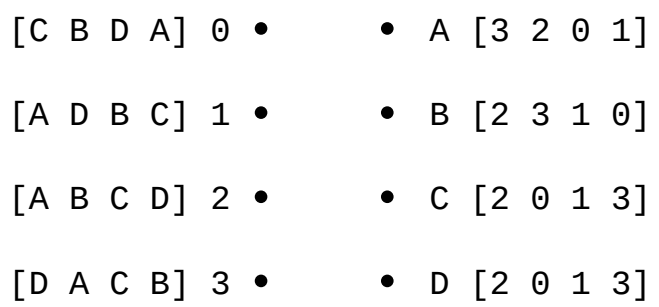
  2-C " "

  ...

  exercise: prove this system has only 1 stable matching
- **brute force stop-when-stable WC looks at $n!$ matchings**

- **can we do better?**
- **can we do a lot better, i.e. polytime?**
- **yes: Gayle-Shapley propose-maybe-reject**

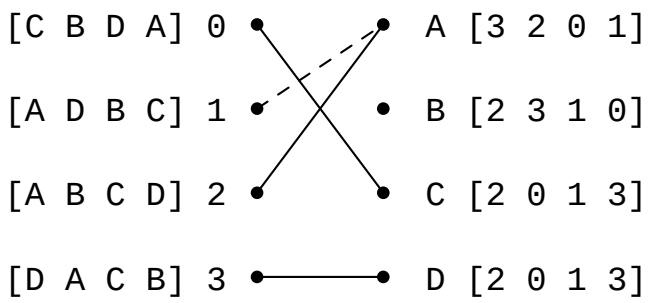
Gayle-Shapley propose-maybe-reject alg'm



	A	B	C	D
0				
1				
2				
3				

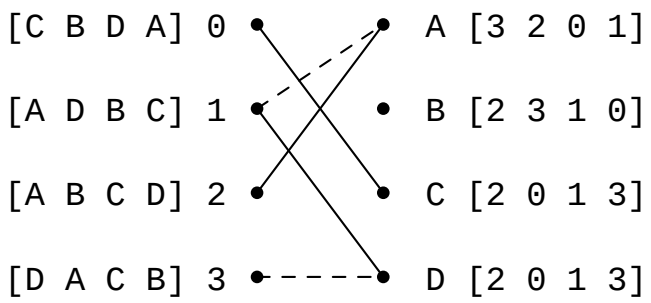
- **each round: propose phase**
  - **each hospital proposes to its most-preferred resident**
- **each round: maybe-reject phase**
  - **each resident says maybe to favorite proposer**
  - **each resident says no to all other proposers**
  - **each rejected proposer erases resident from its list**
- **stop once each resident has a proposal**

Gayle-Shapley: after round 1



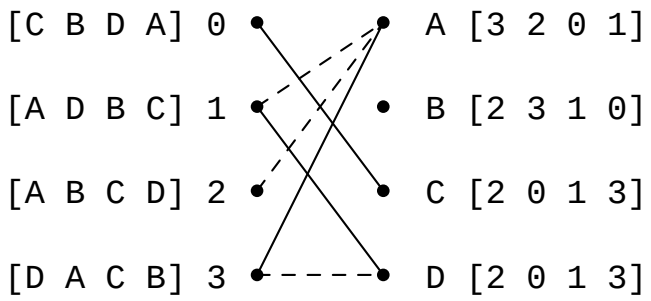
	A	B	C	D
0			**?**	
1	**x**			
2	**?**			
3				**?**

Gayle-Shapley: after round 2



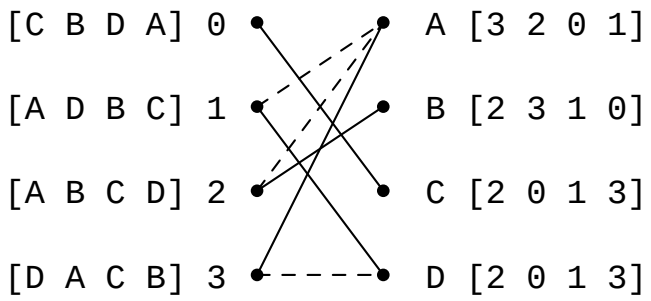
	A	B	C	D
0			**?**	
1	**x**			**?**
2	**?**			
3				**x**

Gayle-Shapley: after round 3



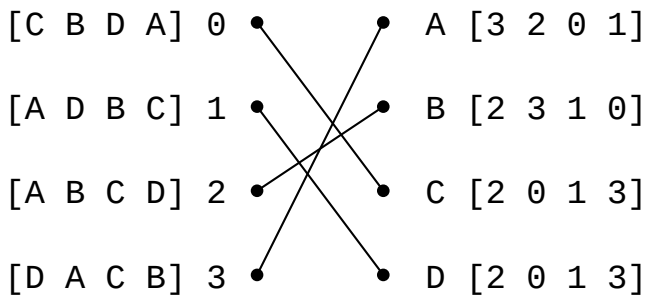
	A	B	C	D
0			**?**	
1	**x**			**?**
2	**x**			
3	**?**			**x**

Gayle-Shapley: after round 4



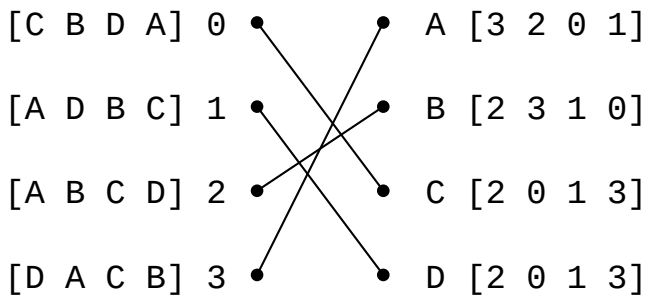
	A	B	C	D
0			**?**	
1	**x**			**?**
2	**x**	**?**		
3	**?**			**x**

Gayle-Shapley: matching



	A	B	C	D
0			?	
1				?
2		?		
3	?			

switch: proposals from R



	0	**1**	**2**	**3**
A				**?**
B			**?**	
C	**?**			
D		**?**		

Gayle-Shapley correctness: termination

- **once maybe-rejecter has proposal, always has proposal**
- **after proposal phase, a rejection or done**
- **each proposer rejected at most $n-1$ times**

  (assume p rejected in rounds 1 to $n-1$.

  assume p proposes to q in round n.

  all rejected by p have proposal, q has proposal)

Gayle-Shapley correctness: stable matching

claim. after each round, matching-so-far is stable

- **proof. assume q said maybe to proposer p**
- **can q be in unhappy couple?**
- **must have q prefers p' to p**
- **can (p', q) be unhappy?**
- **if p' has no maybe, then (p', q) not unhappy, done**
- **so assume p' has maybe from some q'**
- **q never received proposal from p' (q prefers p')**
- **so p' proposed to q' instead of q**
- **so p' prefers q' to q**
- **so (p', q) not unhappy**
- **QED**