L17

4800

11.4.2016

abhi shelat

userid:

What are the 2 restrictions on a flow f: (1) flow constraint $IN(x) = OUT(x)$

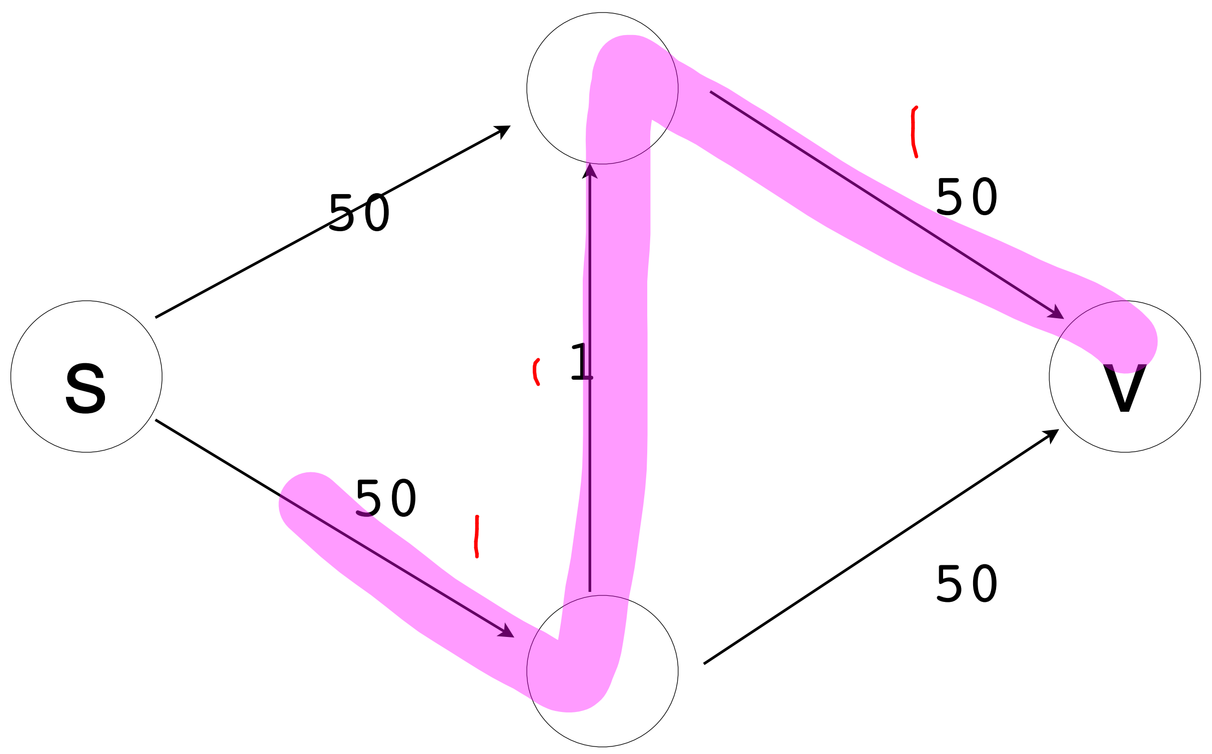
(2) capacity constraint $f(e) \leq c(e)$

What is the value of a flow |f| :

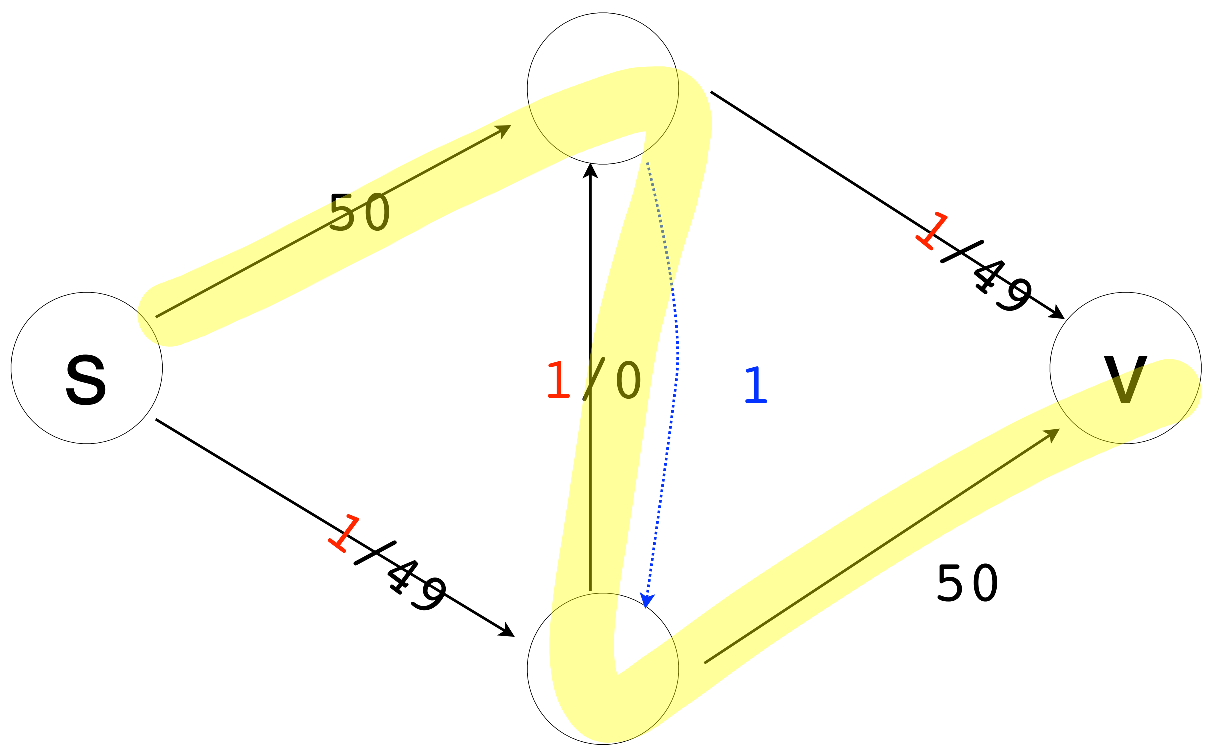
$$|f| = OUT(s) - IN(s)$$

$$\sum_{u \in V} f(s,u) - \sum_{u \in V} f(u,s)$$

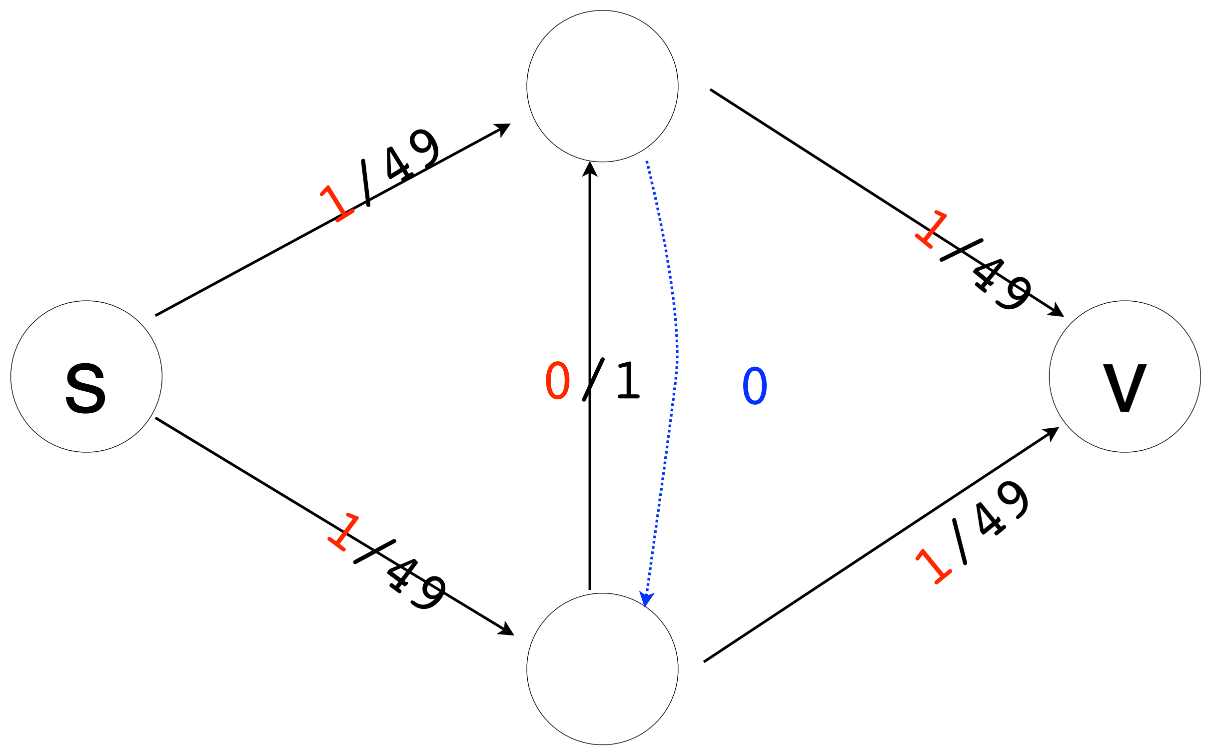
How does the Ford-Fulkerson algorithm work?



S
V
50
50
1
50
50

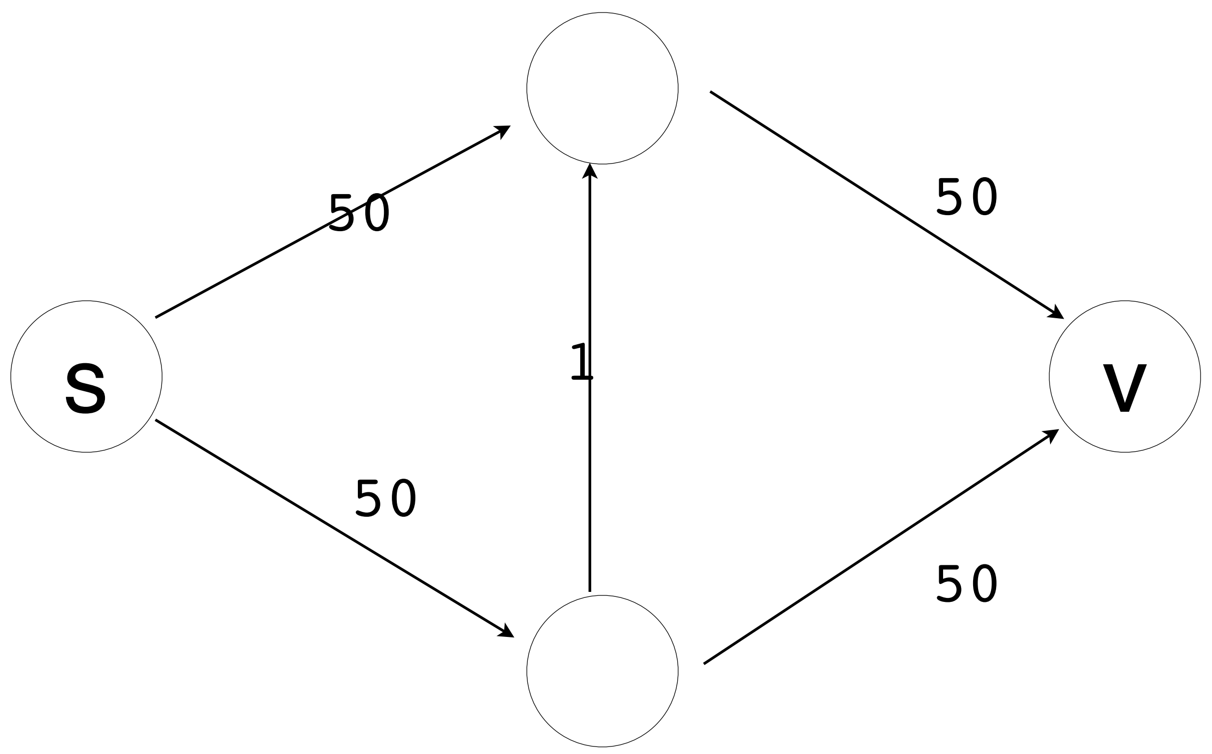


S
V
50
1/49
1/0
1
1/49
50



S
V
1/49
1/49
0/1
0
1/49
1/49

root of the problem



Edmonds-Karp 2

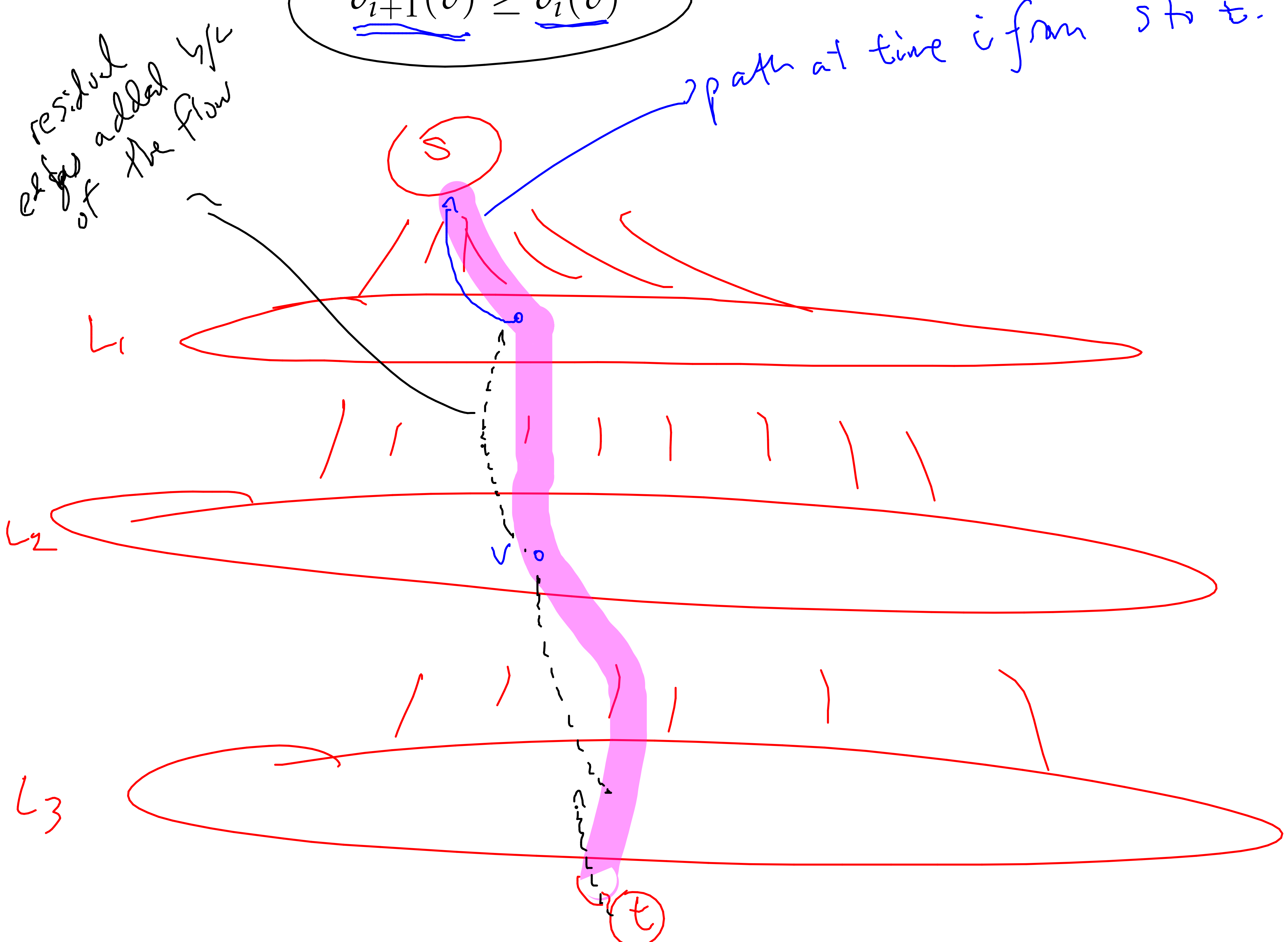
choose path with fewest edges first.

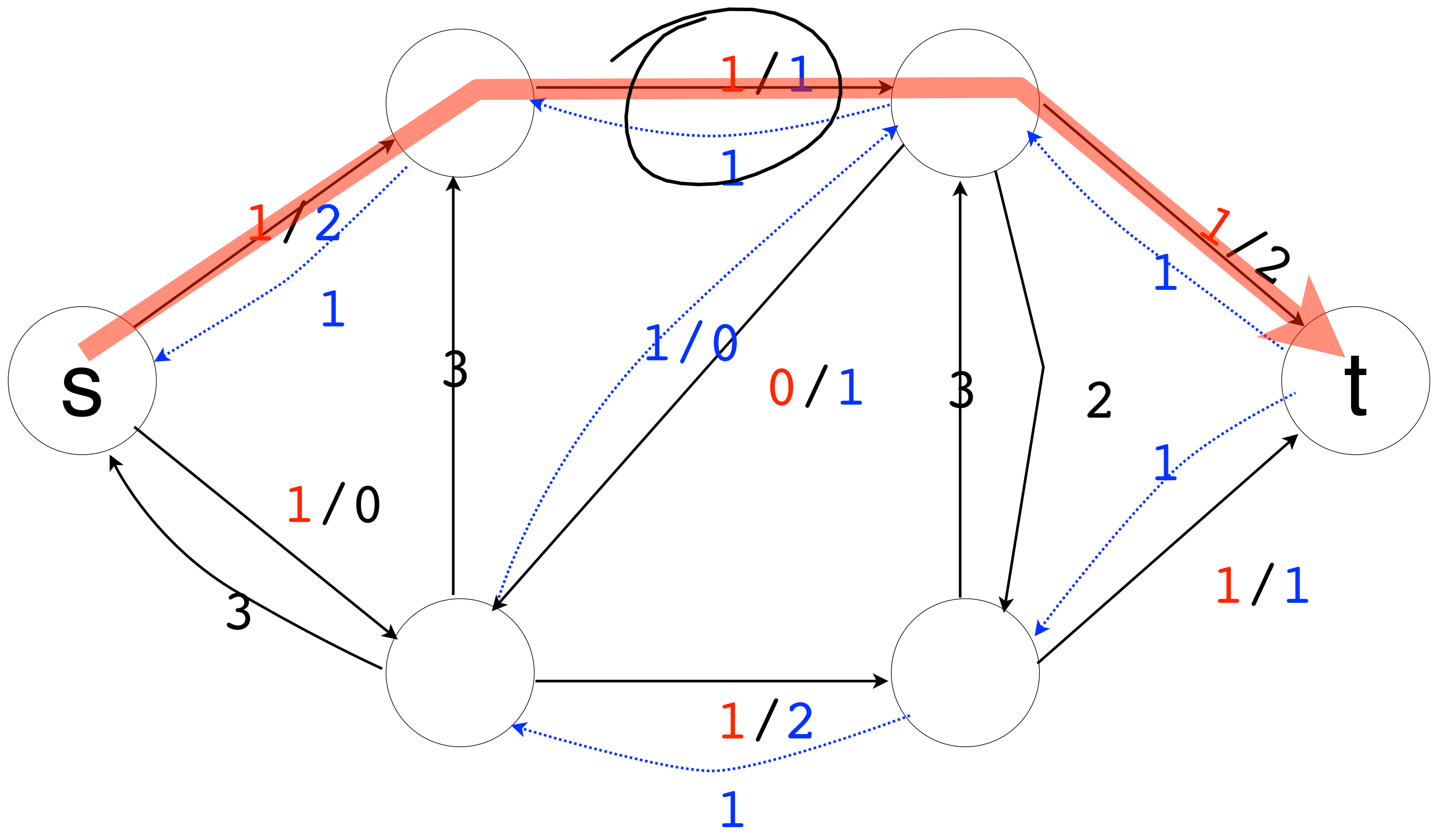
$\delta_f(s,v)$: In G_f, min # of edges in a path from s to v.

Observation

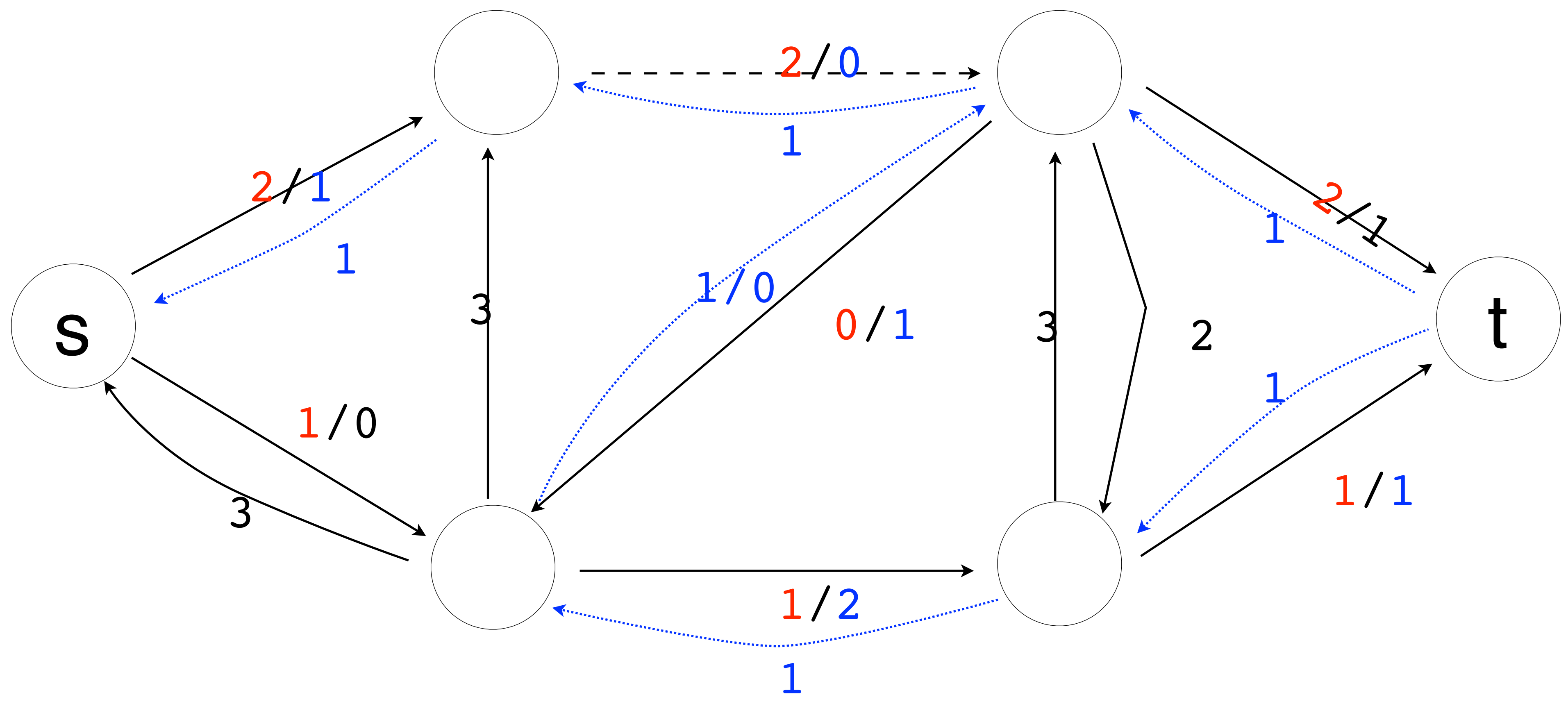
$\delta_f(s,v)$ increases monotonically thru exec

$$\delta_{i+1}(v) \geq \delta_i(v)$$

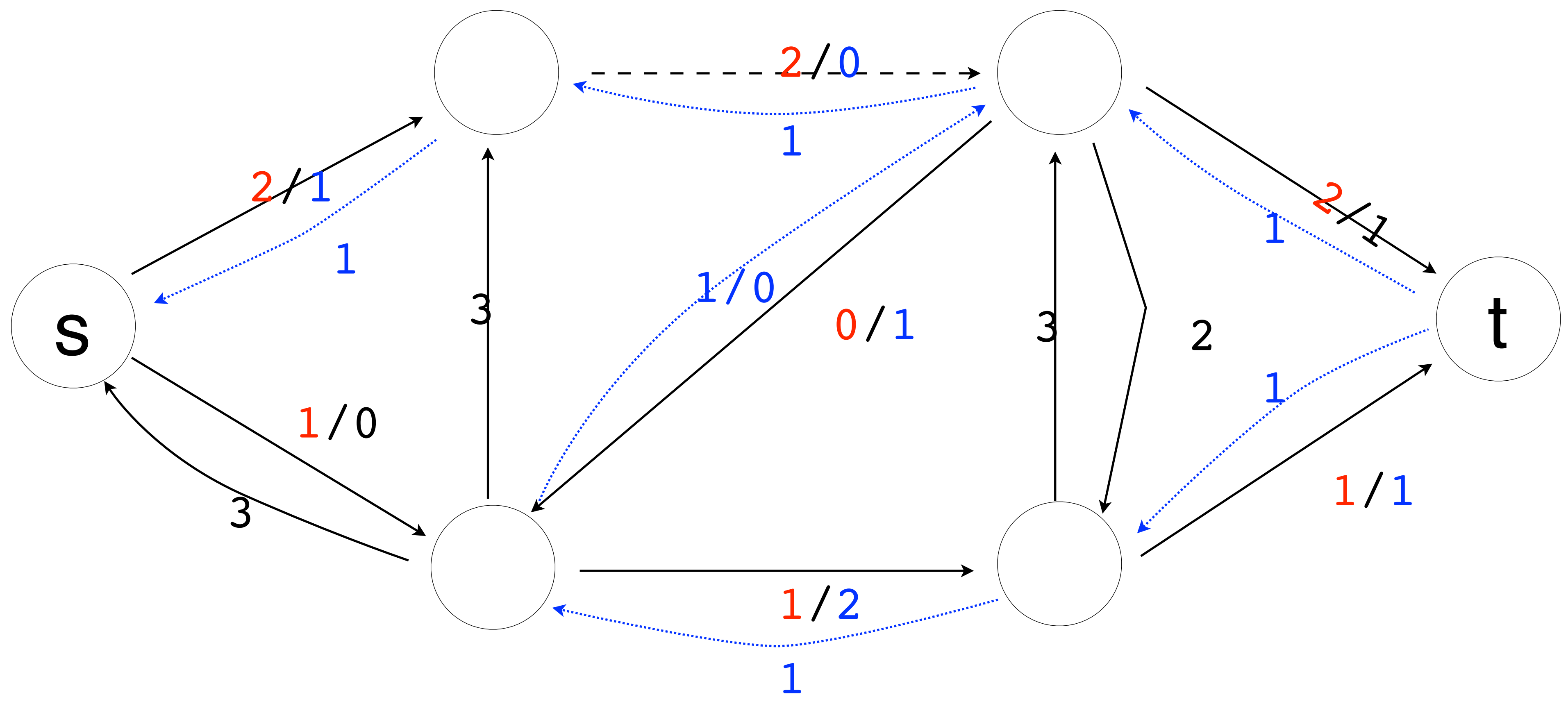




for every augmenting path, some edge is critical.

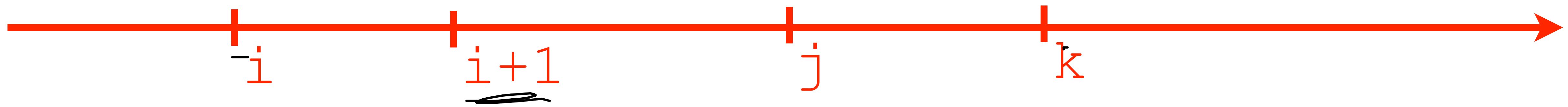


critical edges are removed in next residual graph.



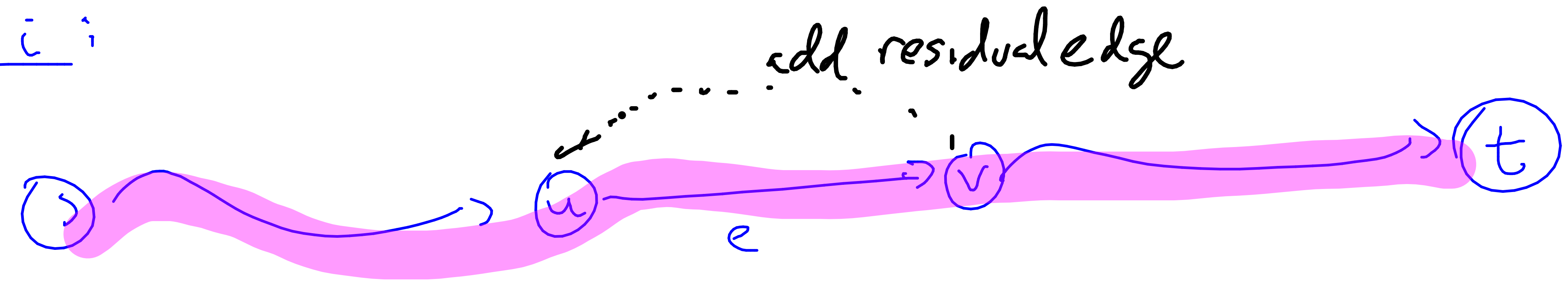
key idea: how many times can an edge be critical?

Time



first time edge $e=(u,v)$ becomes critical

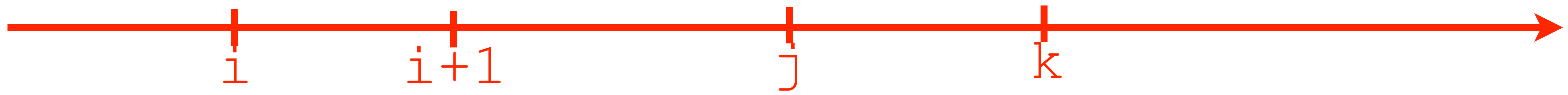
time i:



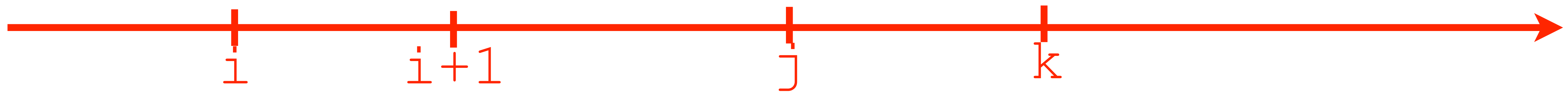
$$\delta_i(s,v) = \delta_i(s,u) + 1$$

time $i+1$

$$\boxed{\delta_{i+1}(s,v) \geq \delta_i(s,v) = \delta_i(s,u) + 1}$$



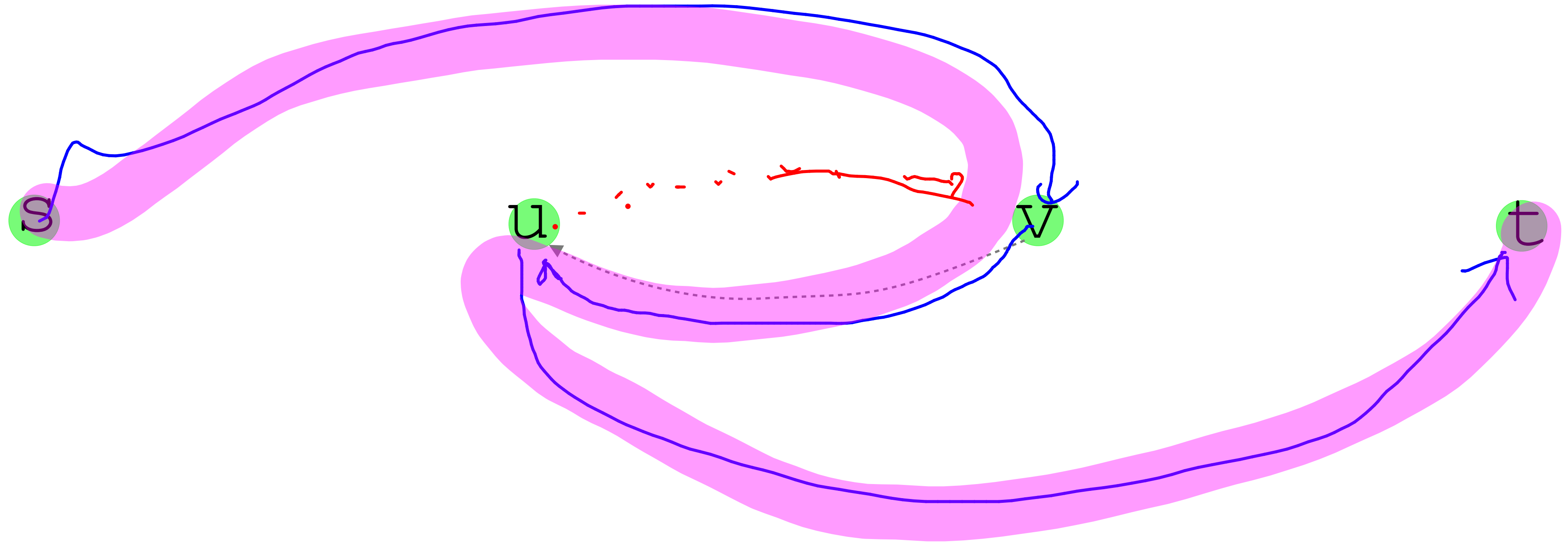
first time (u,v) is critical:



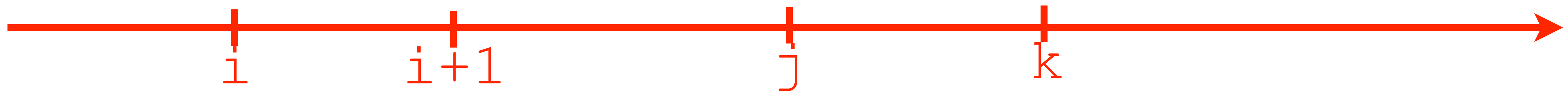
time i+1: (u,v) is critical: $\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$

time j: Edge (u,v) STRIKES BACK

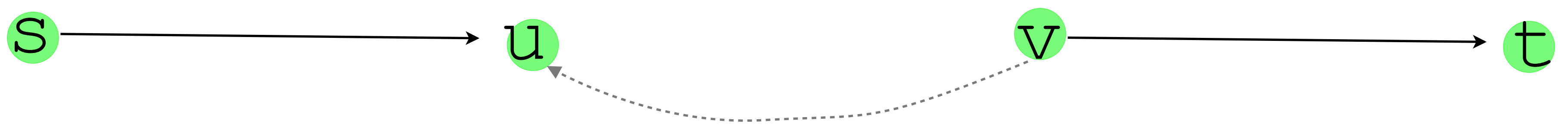
some path from $s \rightsquigarrow v \longrightarrow u \rightsquigarrow t$ must become the shortest augmenting path



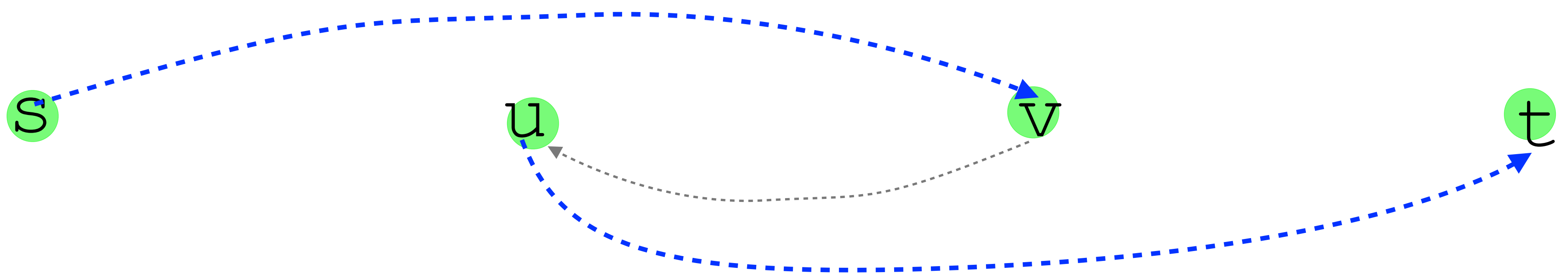
$$\delta_j(s,u) = \delta_j(s,v) + 1 \geq \delta_{i+1}(s,v) + 1 \geq \delta_i(s,u) \underbrace{+1+1}_{+2}$$



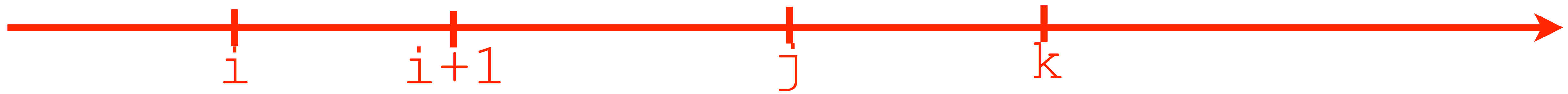
time i+1: (u,v) is critical: $\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$



time j: Edge (u,v) STRIKES BACK



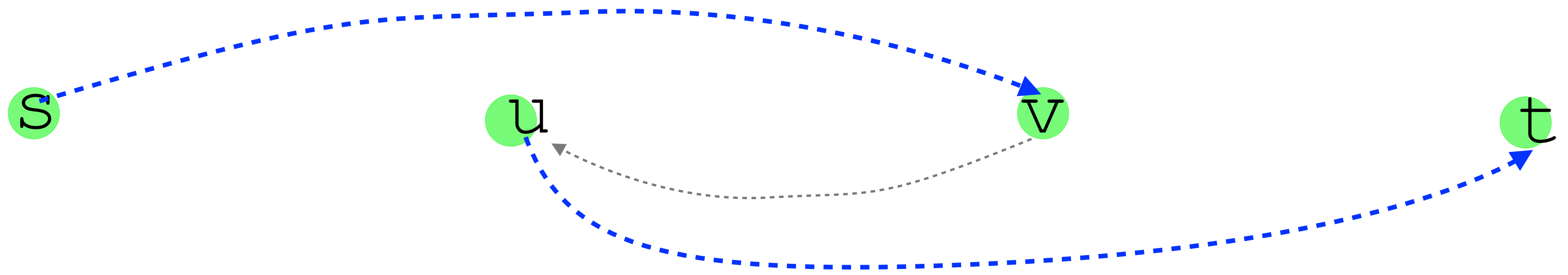
$$\delta_j(s,u) = \delta_j(s,v) + 1$$

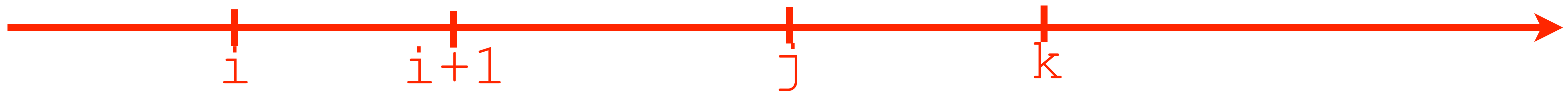


time j: Edge (u,v) STRIKES BACK

$$\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$$

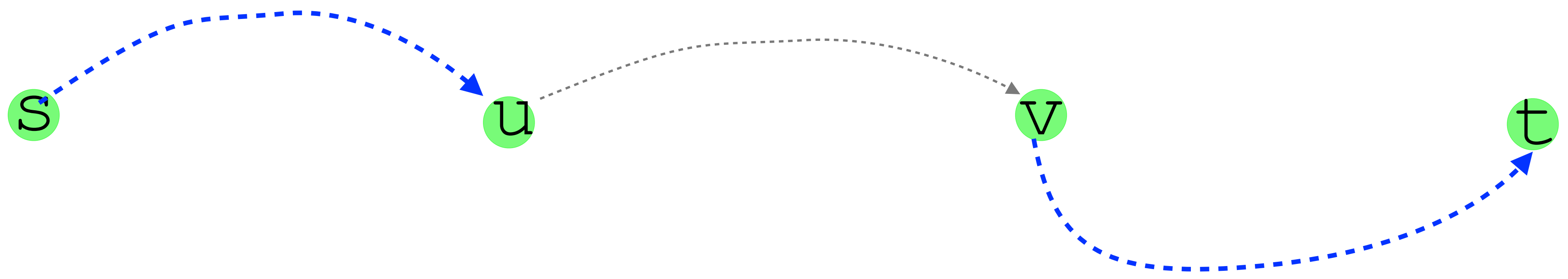
$$\delta_j(s,u) = \delta_j(s,v) + 1$$





time k: RETURN OF THE (u,v) critical

$$\delta_k(s,u) \geq \delta_i(s,u) + 2$$



QUESTION: How many times can (u,v) be critical?

$\frac{V}{2}$ because $\delta_n(s,u)$ is always at most $V-1$.

edge critical only $\frac{V}{2}$ times.

there are only E edges.

ergo, total # of augmenting paths: $O(EV)$

time to find an augmenting path: $O(E+V)$

total running time of E-K algorithm: $O(E^2V)$

FF $O(E|f^*|)$

EK2 $O(E^2V)$ DINIC $O(EV^2)$

PUSH-RELABEL

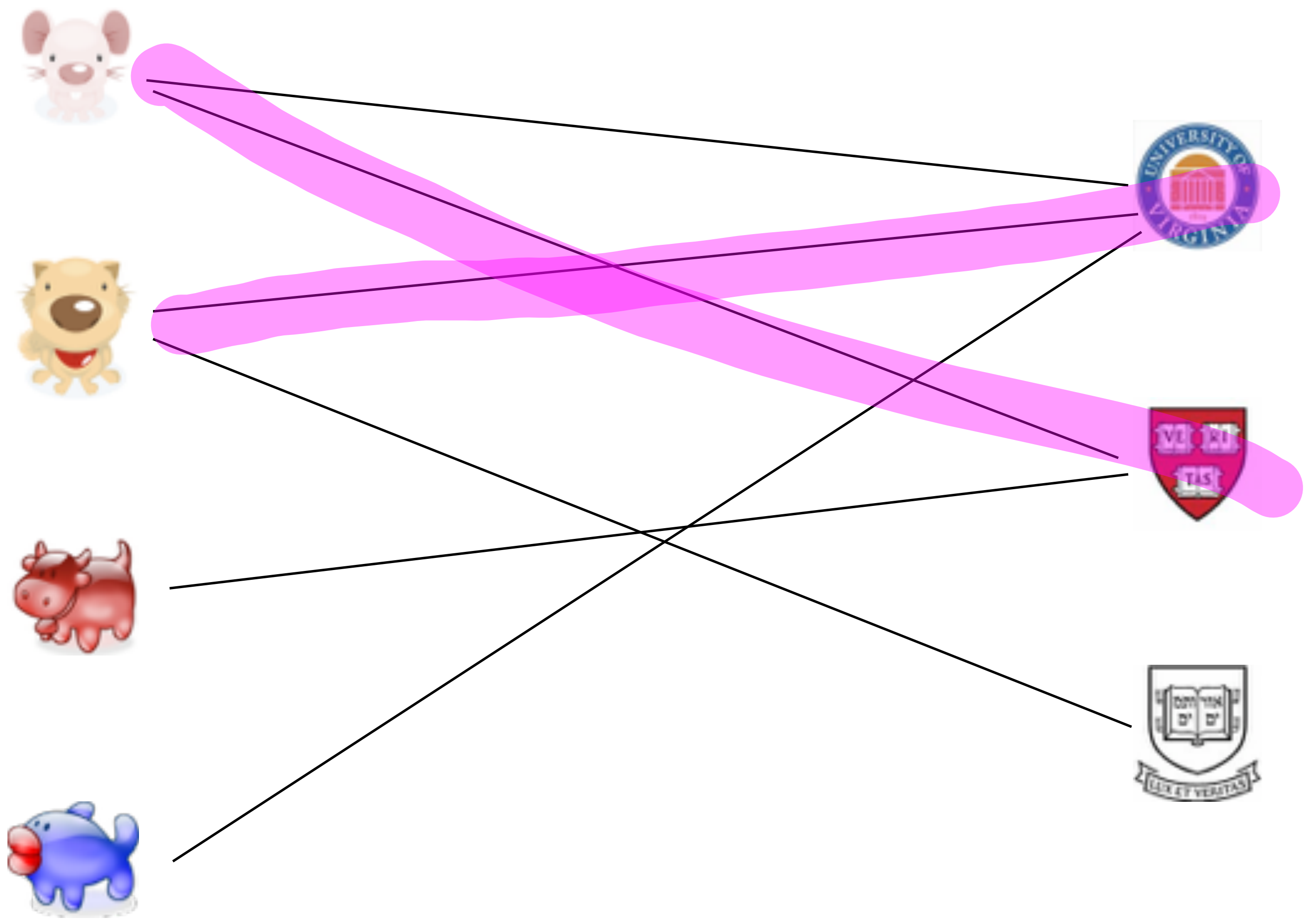
FASTER PUSH-RELABEL $O(V^3)$

Golberg-Rao: $O\left(E \min\{V^{2/3}, E^{1/2}\} \log\left(\frac{V^2}{E}\right) \cdot \log(U)\right)$

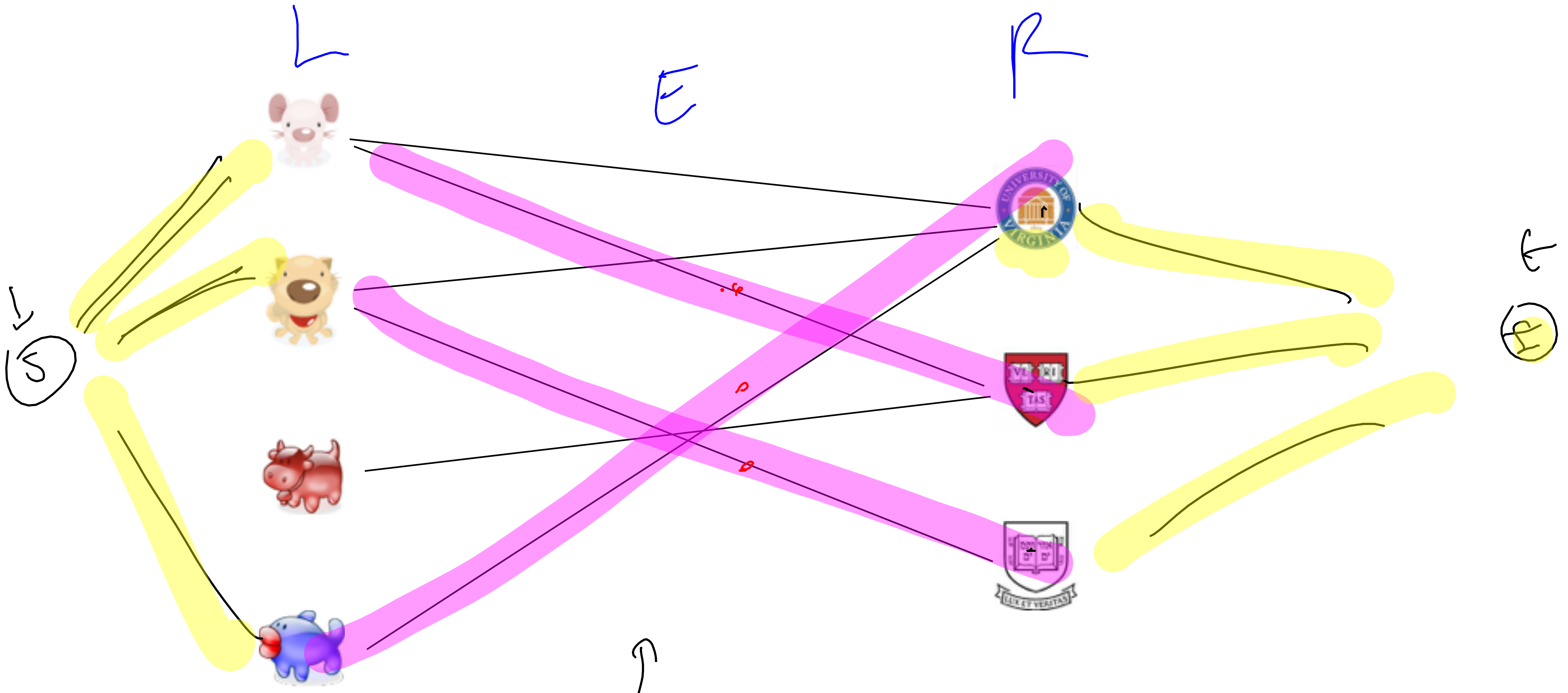
max of edge capacities

Bipartite

maximum bipartite matching



maximum bipartite matching



$\forall e \in M$, add 1 unit of flow

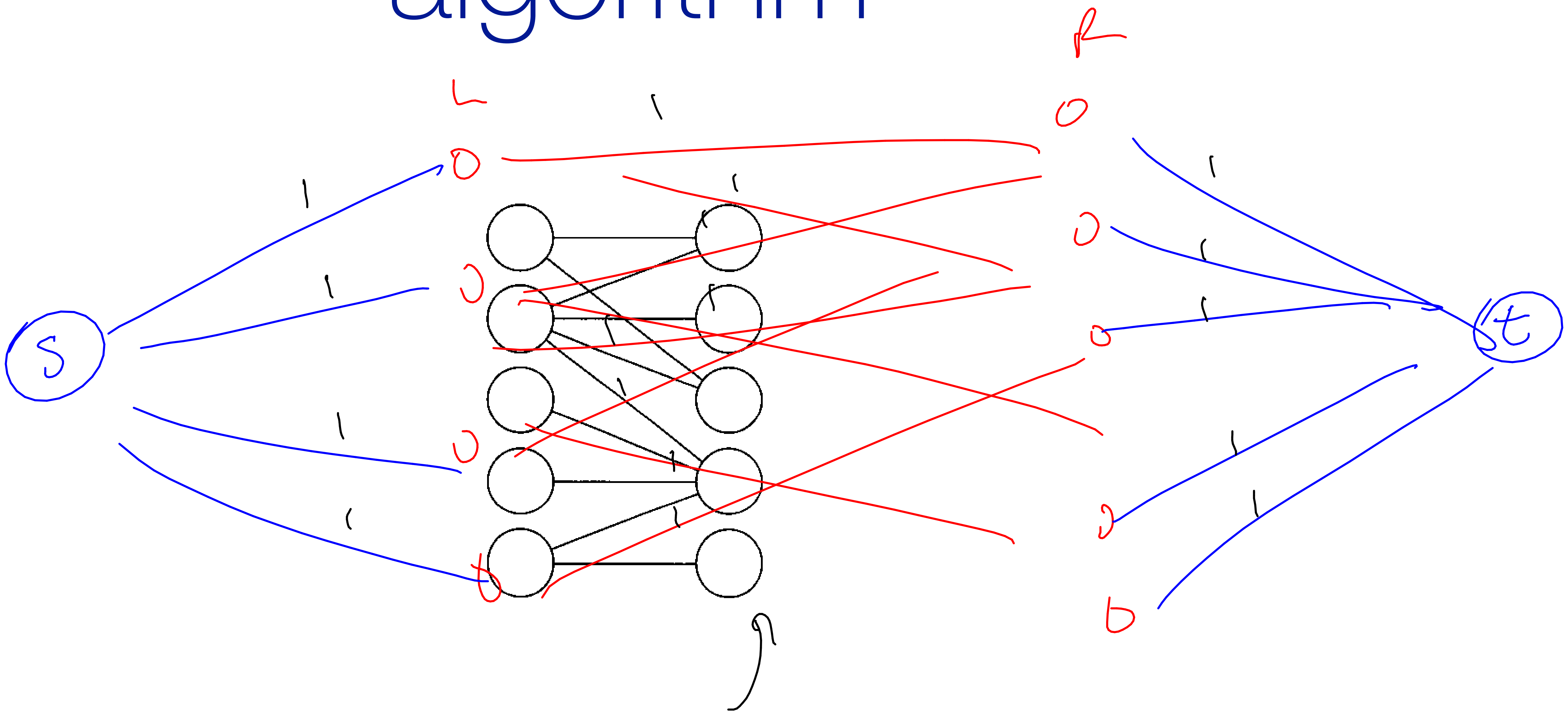
bipartite matching

problem: given a bipartite graph, $G = (L, R, E)$, all E are between L and R

Find a "largest" subset $M \subseteq E$ such that each node occurs at most once in M,

Further find the largest such M.

algorithm



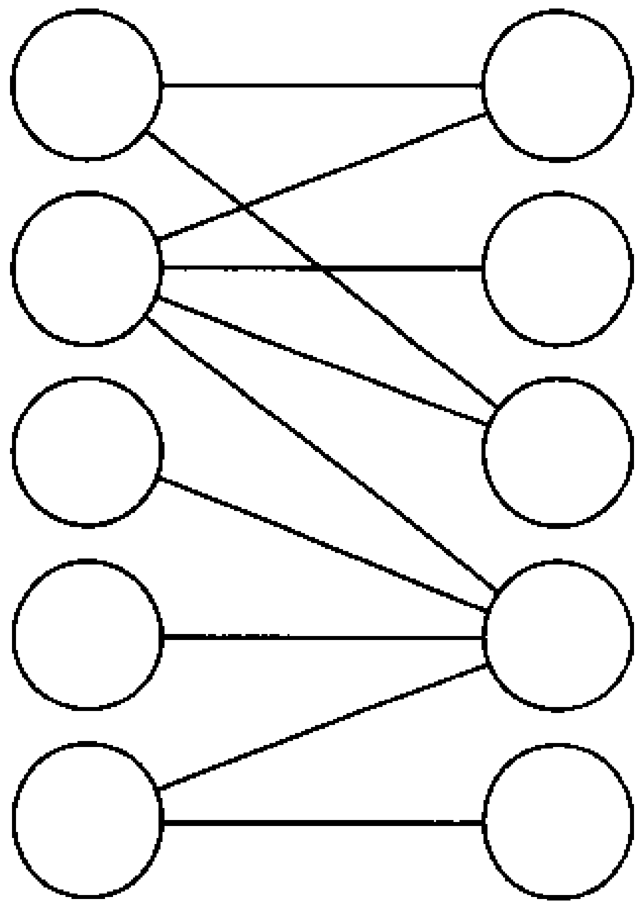
$G' = (V = \{s,t\} \cup L \cup R,\ E \text{ as above})$ w

algorithm

1. MAKE NEW G'
FROM INPUT G.

2. RUN FF ON G'

3. OUTPUT ALL MIDDLE EDGES
WITH FLOW F(E)=1.



correctness

IF G HAS A MATCHING OF SIZE K, THEN G' has a max flow of K.

Proof: Given a matching M, of size K, construct a flow f which assigns 1 unit of flow to each $e \in M$, and 1 unit of flow from $(s \rightsquigarrow u)$ and $(v \rightsquigarrow t)$ for every $e = (u,v) \in M$.

① show that f is a valid flow.

ⓐ capacity constraint

ⓑ flow constraint (conservation)

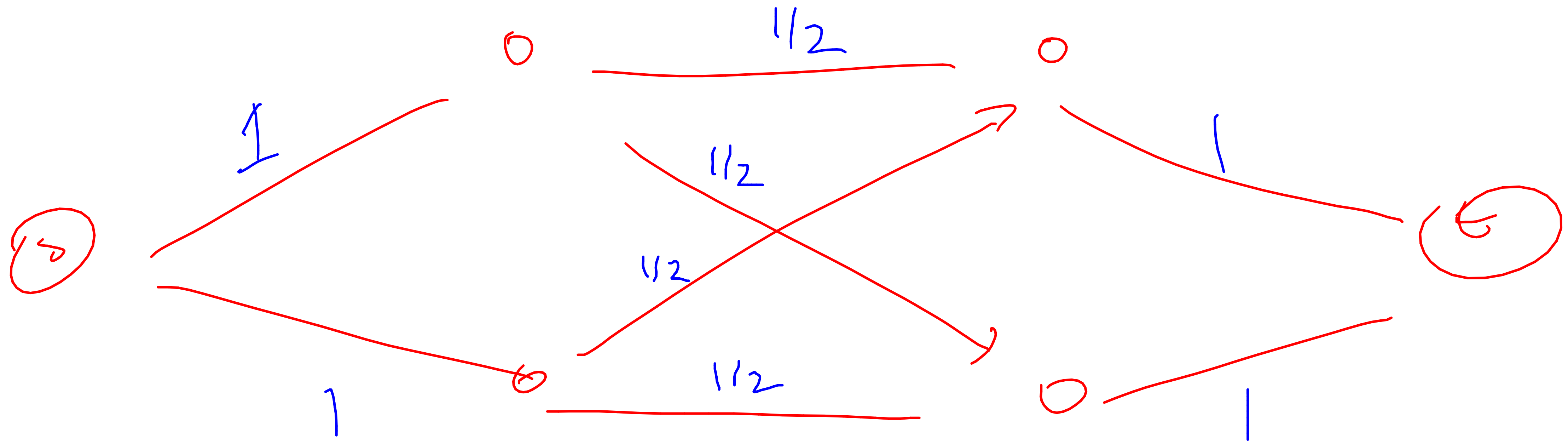
correctness

IF G' HAS A FLOW OF K, THEN G has a matching of size K.

Consider all edges in G' between L and R of $f(e) = 1$.

Add e to M. $\Rightarrow |M| = K$, so G has a K-matching.

G':



G' has a flow of 2.

integrality theorem

IF CAPACITIES ARE ALL INTEGRAL, THEN FF returns an integral flow.

Proof: By induction.

Base case: @ start, FF has an integral flow (0)

Spse true after i iterations.

On iteration i, flow is integral, so residual capacities on all edges are integral. FF finds an augmenting path P, and the min capacity edge will therefore be integral. ⇒ flow remains integral on iteration $i+1$.

correctness

' HAS A FLOW OF K, THEN G HAS K-MATCHING.

B/c G' has integral capacities, the FF returns an integral flow.

Every edge has either $f(e)=0$ or $f(e)=1$.

Set M to be all edges b/w L and R w $f(e)=1$. Can be atmost K by MIN-CUT theorem.

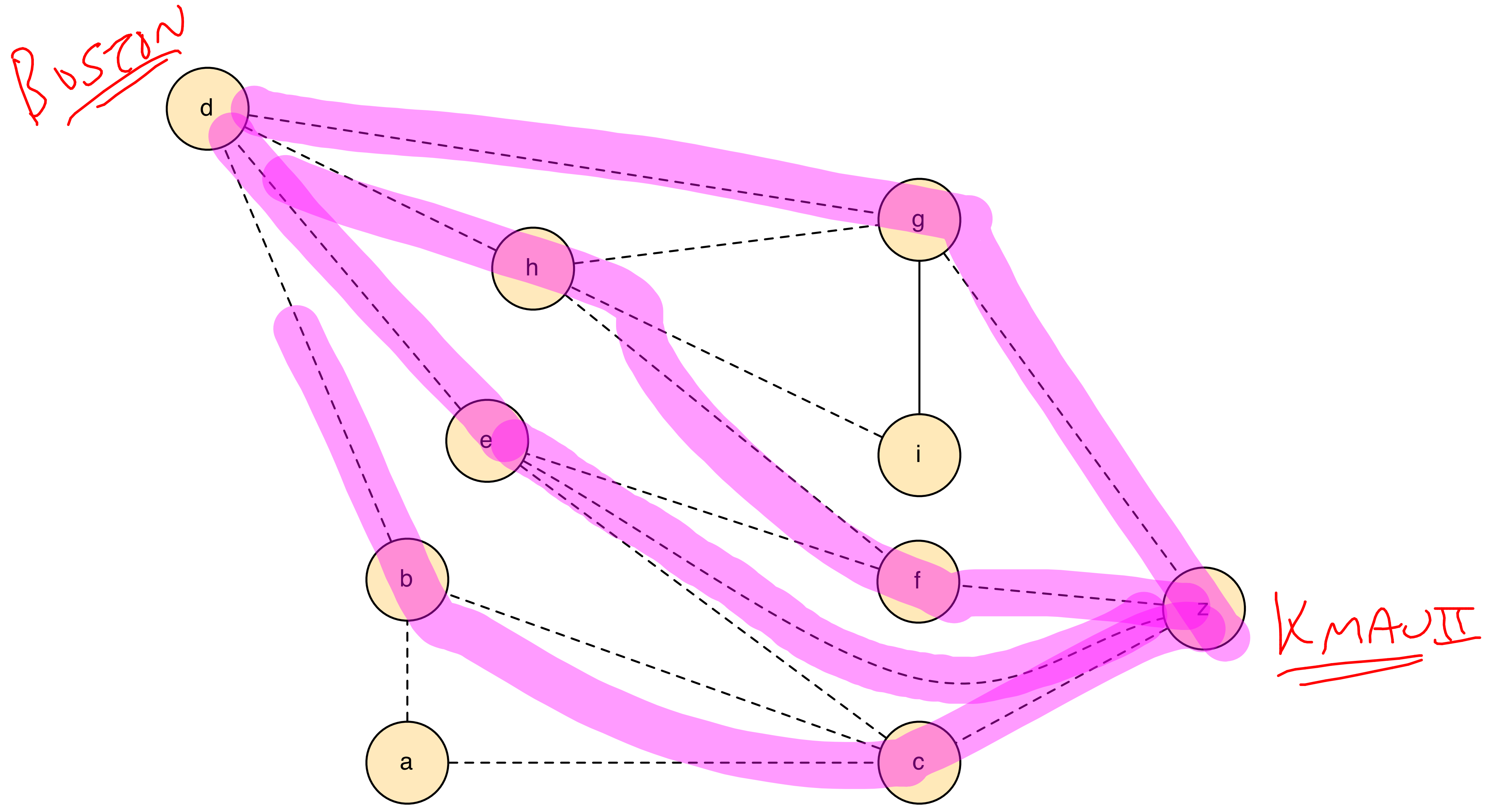
Each node appears at most once in M by the conservation constraint.

running time

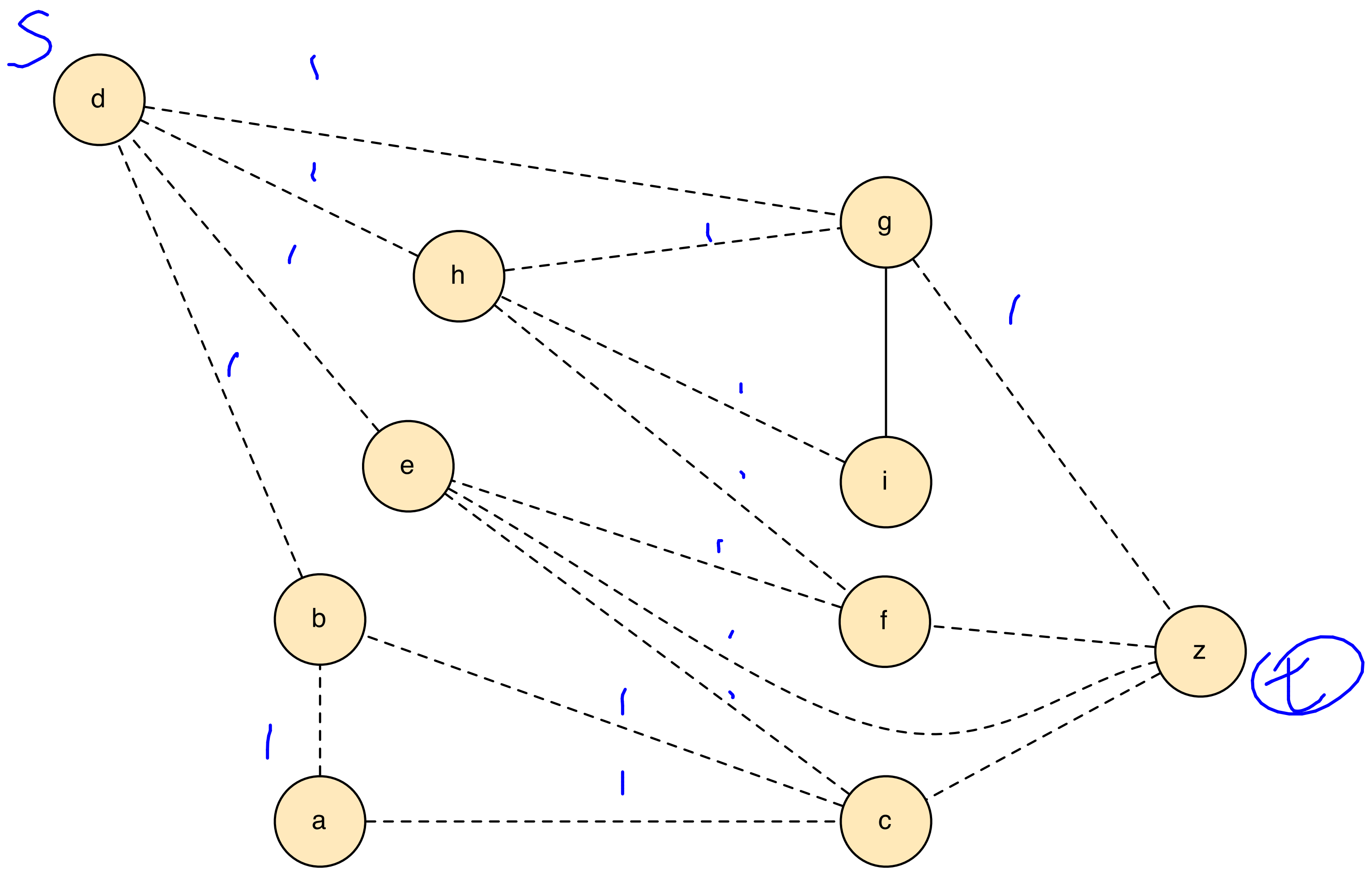
$O(E \cdot |f|) \sim O(\underline{E \cdot V})$

↑ b/c the max matching has size L or R ~ V.

edge-disjoint paths



algorithm



Argue that this is correct

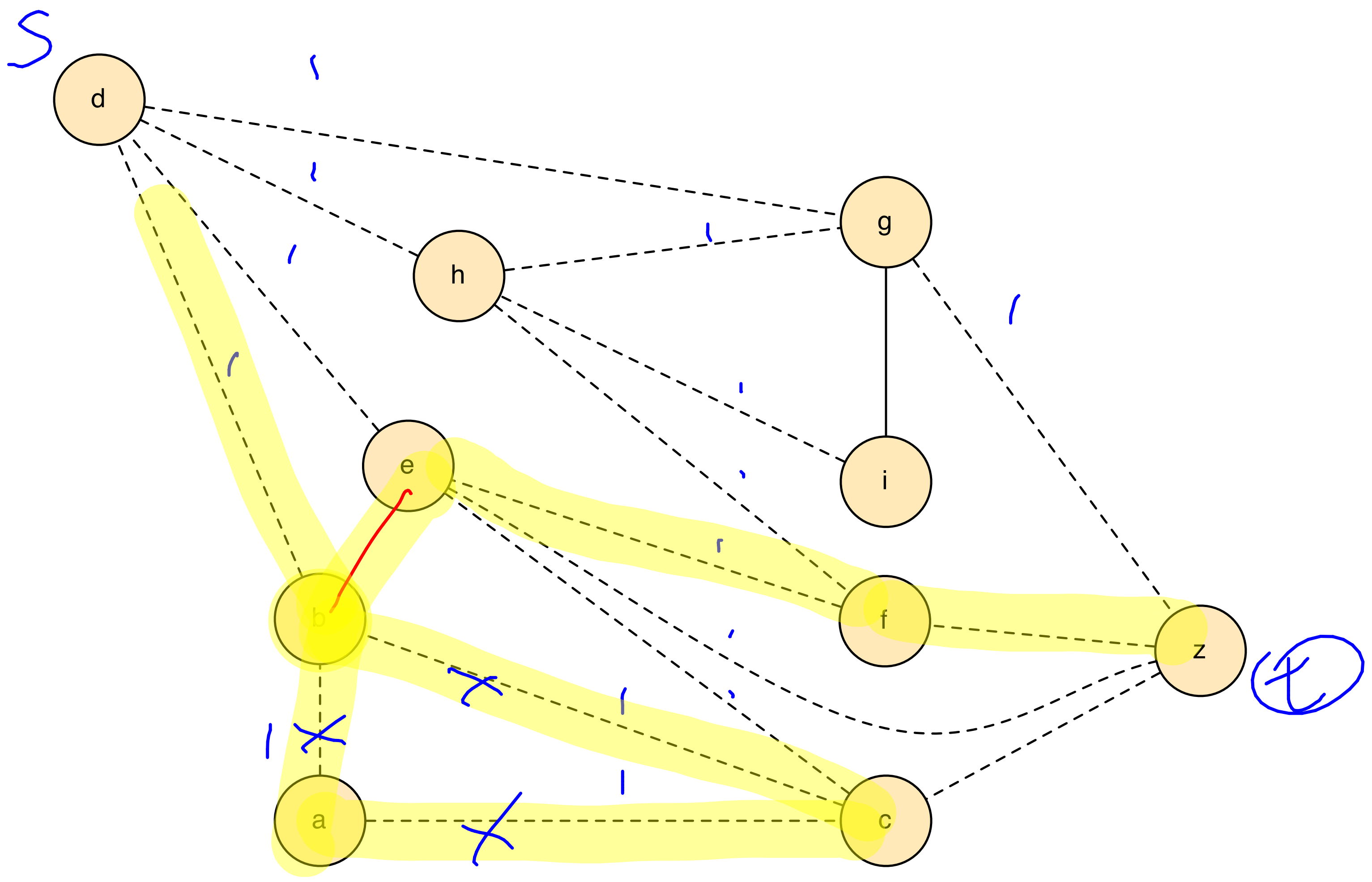
① If G has k edge disjoint paths $\Rightarrow$ G has a k max flow

② If G has a k-maxflow $\Rightarrow$ G has k edge disjoint paths

① By integrality, $f(e) = \{0 \text{ or } 1\}$

② $\exists$ k edge disjoint paths among all the edges with $f(e) = 1$.

algorithm



Argue that this is correct

① If G has k edge disjoint paths

⇒ G has a k max flow

② If G has a k-max flow

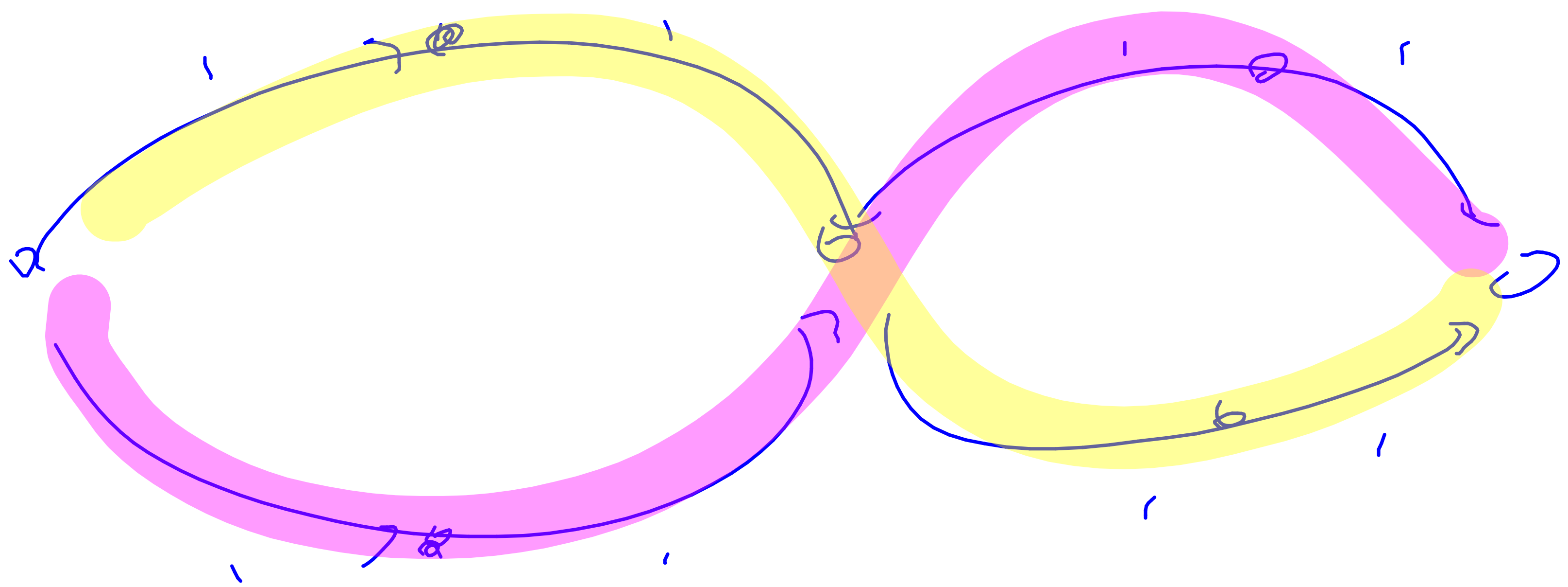
⇒ G has k edge disjoint paths.

① By Integrality, $f(e) = \{0 \text{ or } 1\}$

② $\exists$ k edge disjoint paths among all the edges with $f(e) = 1$.

1. Compute max flow
2. Remove all edges with f(e) = 0.
3. Walk from s.
    1. If you reach a node you have visited before, erase flow along path
    2. If you reach t, add this path to your set, erase flow along path.

Works by Induction on the # of edges with $f(e) = 1$



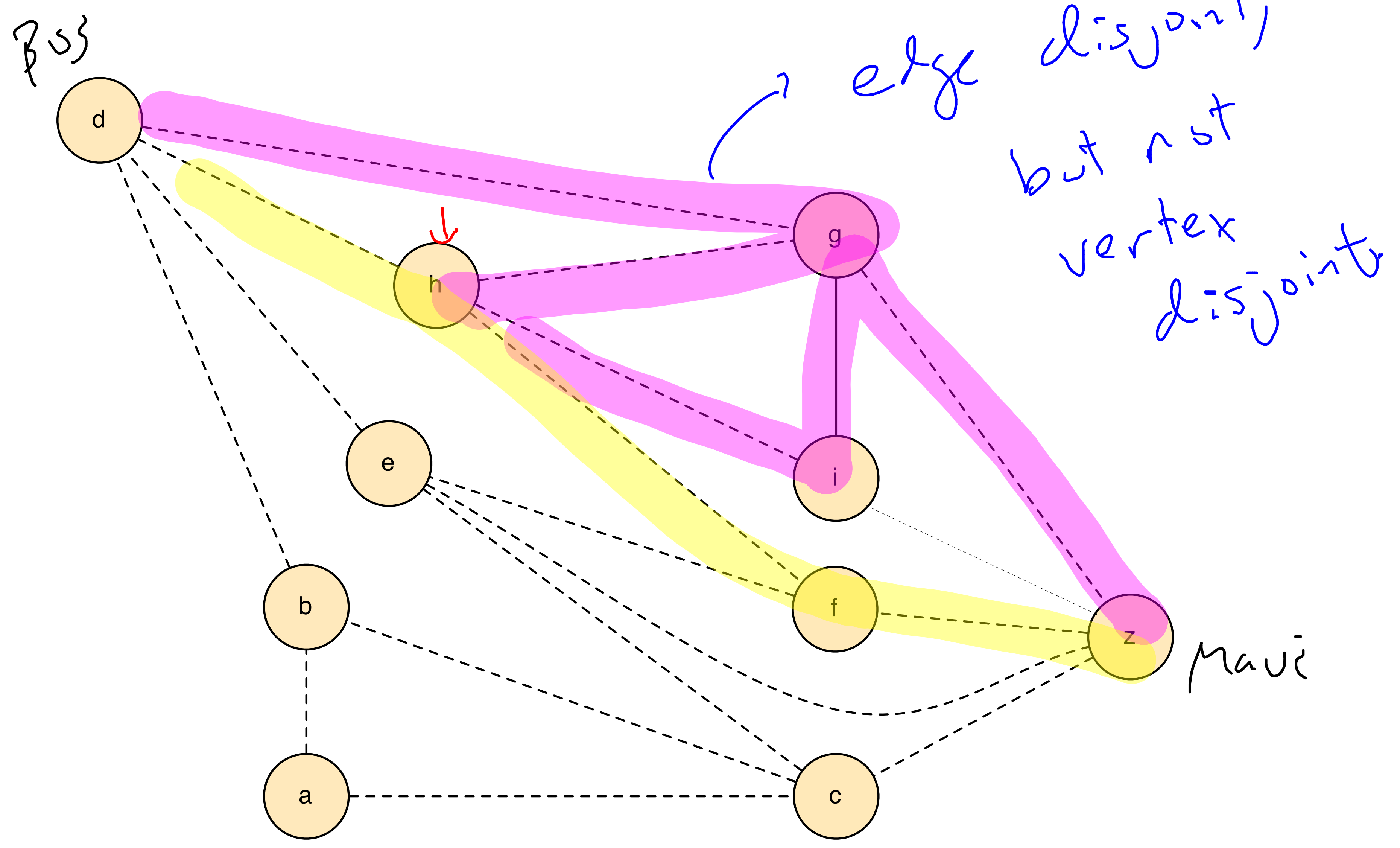
analysis

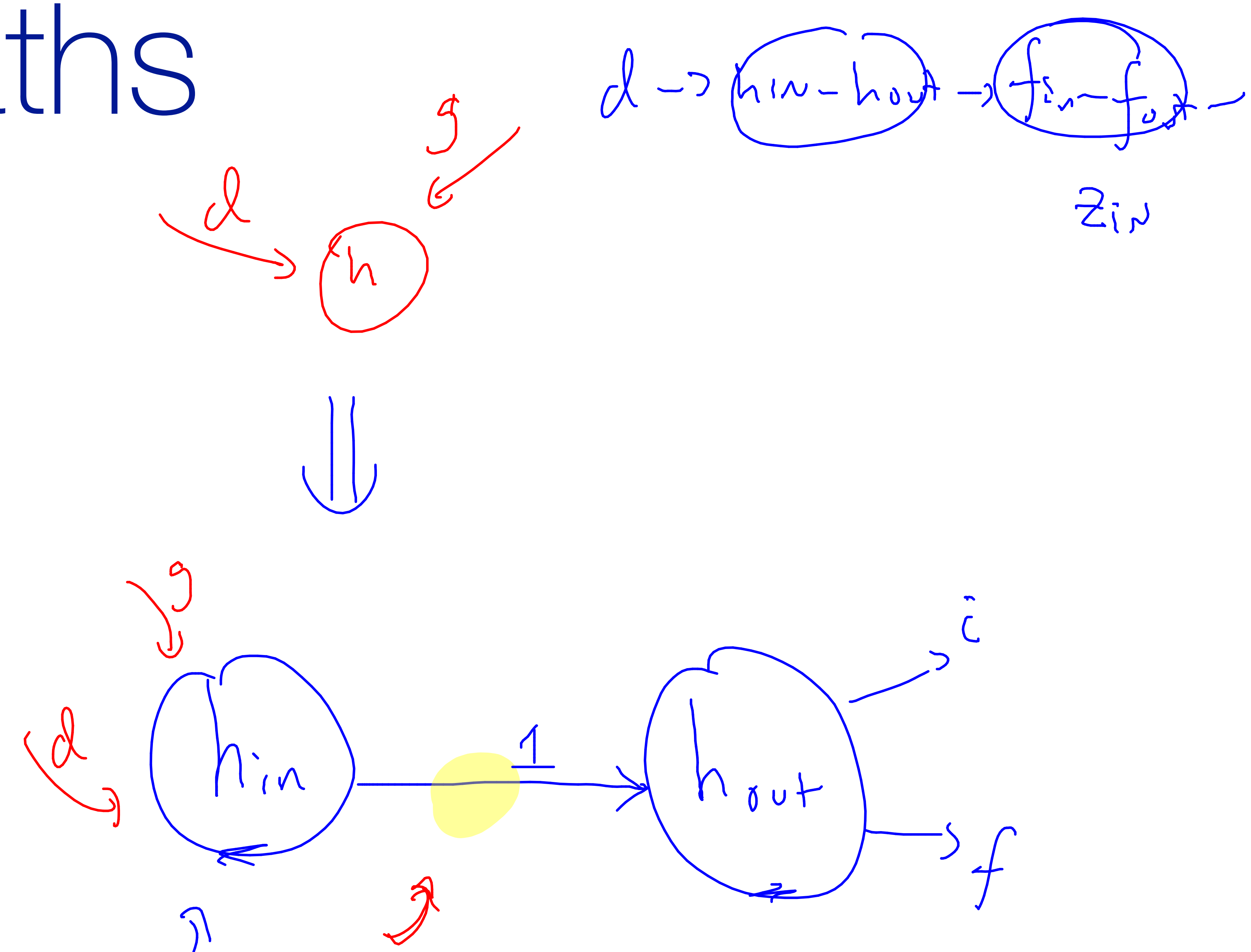
IF G HAS K DISJOINT PATHS, THEN

analysis

G' HAS A FLOW OF K, THEN

vertex-disjoint paths

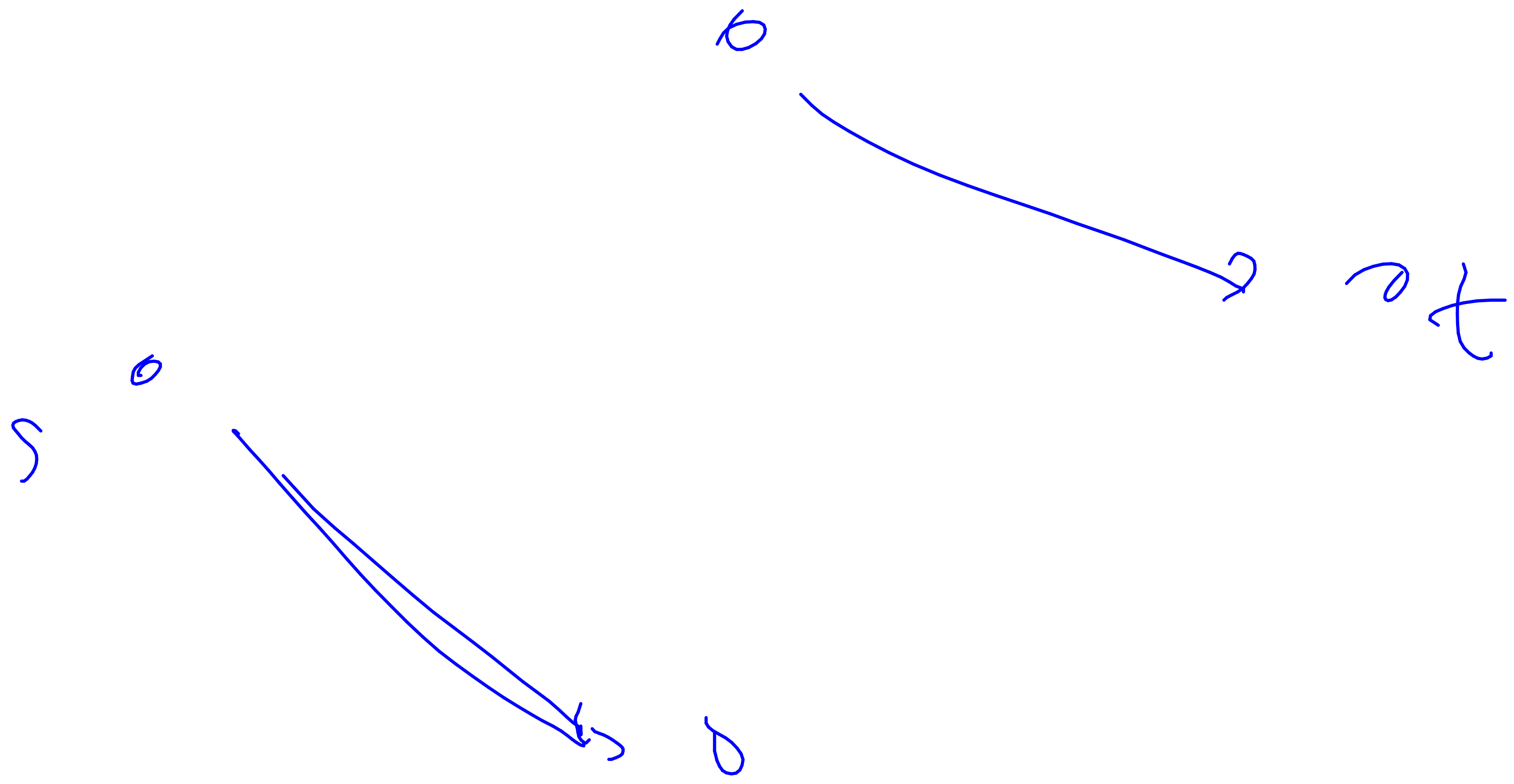




① Make G' by substituting gadget for every node in G,

② Compute edge disjoint paths in G'.

③ Turn ∫ into the node disjoint paths of G.


s
t

baseball elimination

	W	L	Left	Against A	P	N	M
ATL	83	71	8	-	1	6	1
PHL	80	79	3	1	-	0	2
NY	78	78	6	6	0	-	0
MONT	77	82	3	1	2	0	-

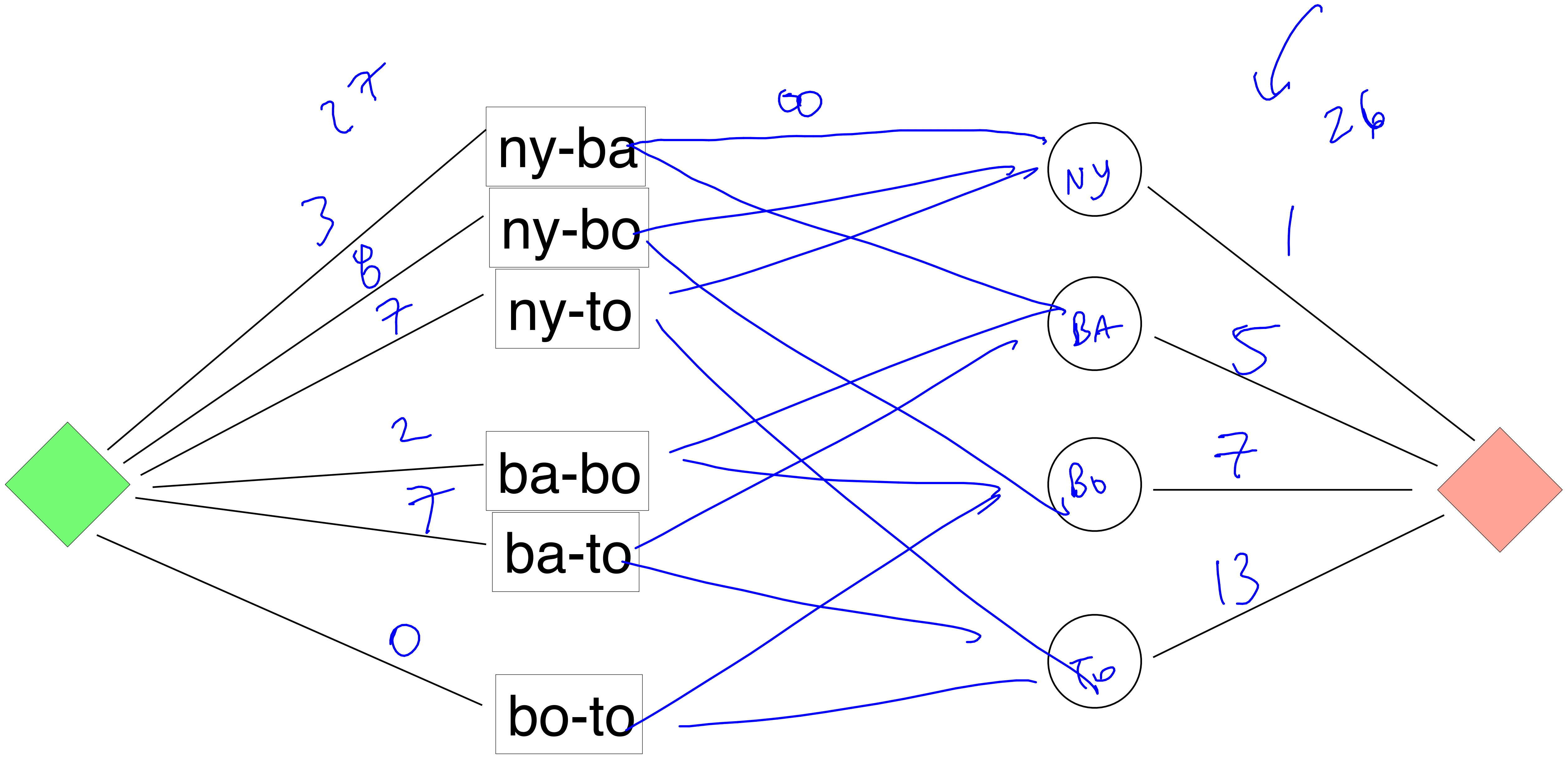
baseball elimination

77

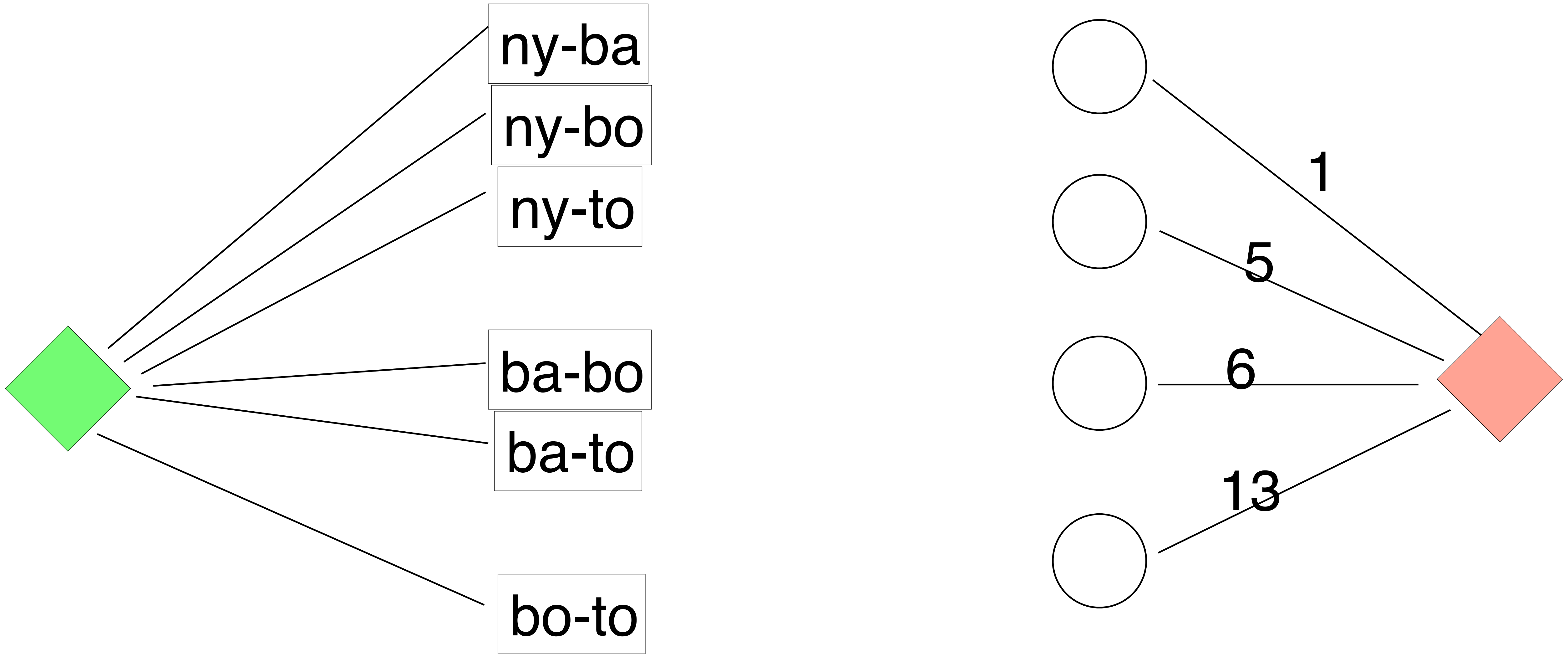
	W	L	Left	Against N	B	Bo	T	D
NY	75	59	28		3	~~8~~ 7	~~7~~	3
BAL	71	63	28	~~3~~		~~2~~	7	4
BOS	69	66	27	8 7 1	~~2~~			
TOR	63	72	27	7	7			
~~DET~~	49	86	27	3	4			

75
71
69
63
278
27
305

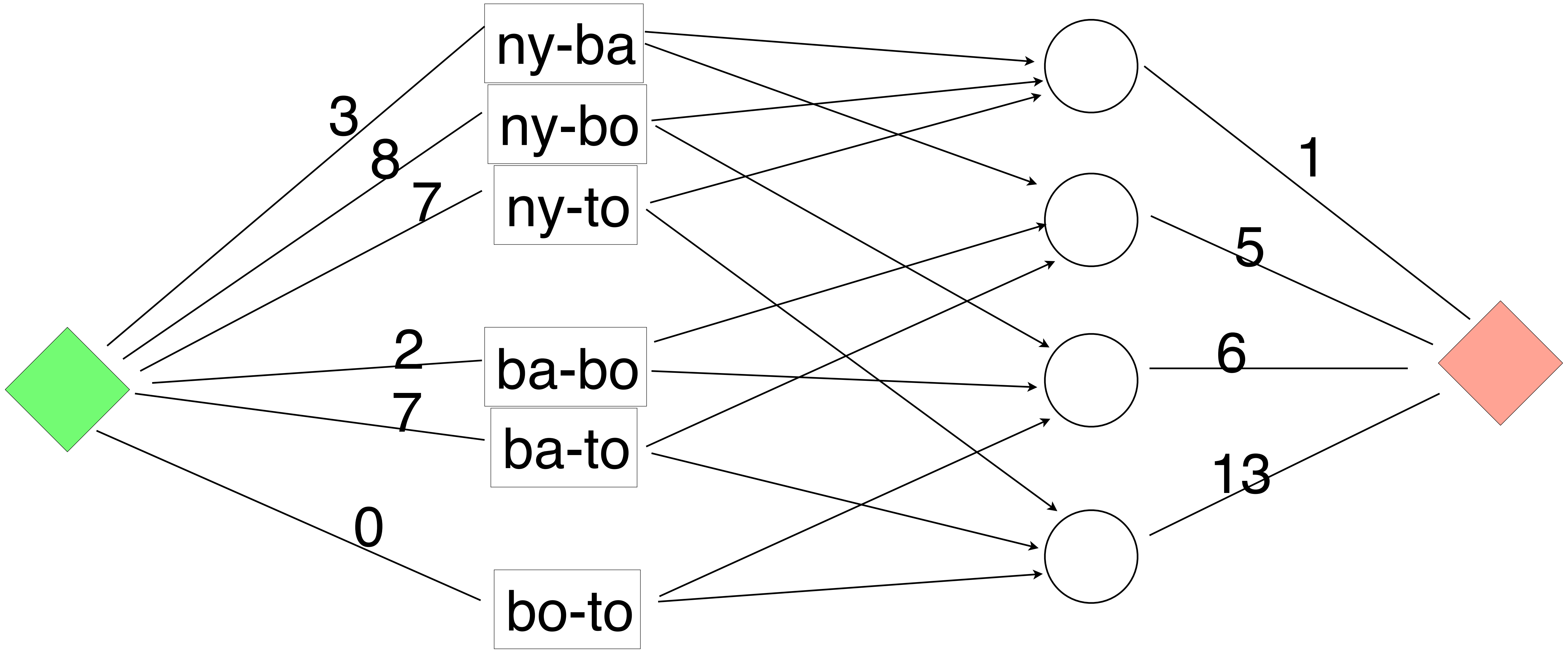
76.25
4 | 305
28
25



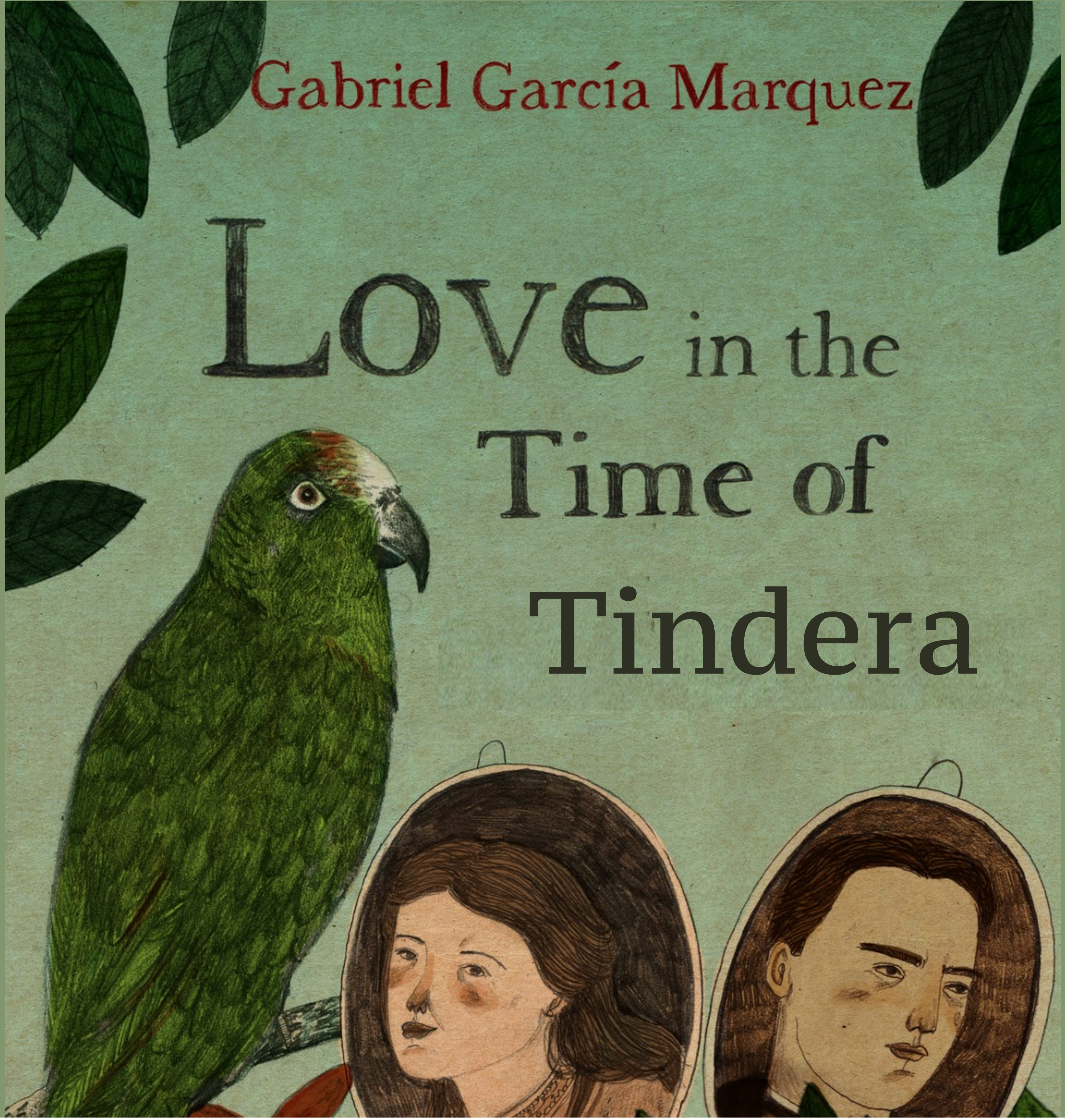
	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			



	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			



	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			


Gabriel García Marquez
Love in the
Time of
Tindera

We have a group of suitors and reviewers

2>1>3

2>3>1

1>3>2

Each has preferences over the other group

1>3>2

1>2>2

3>2>1

2>1>3

2>3>1

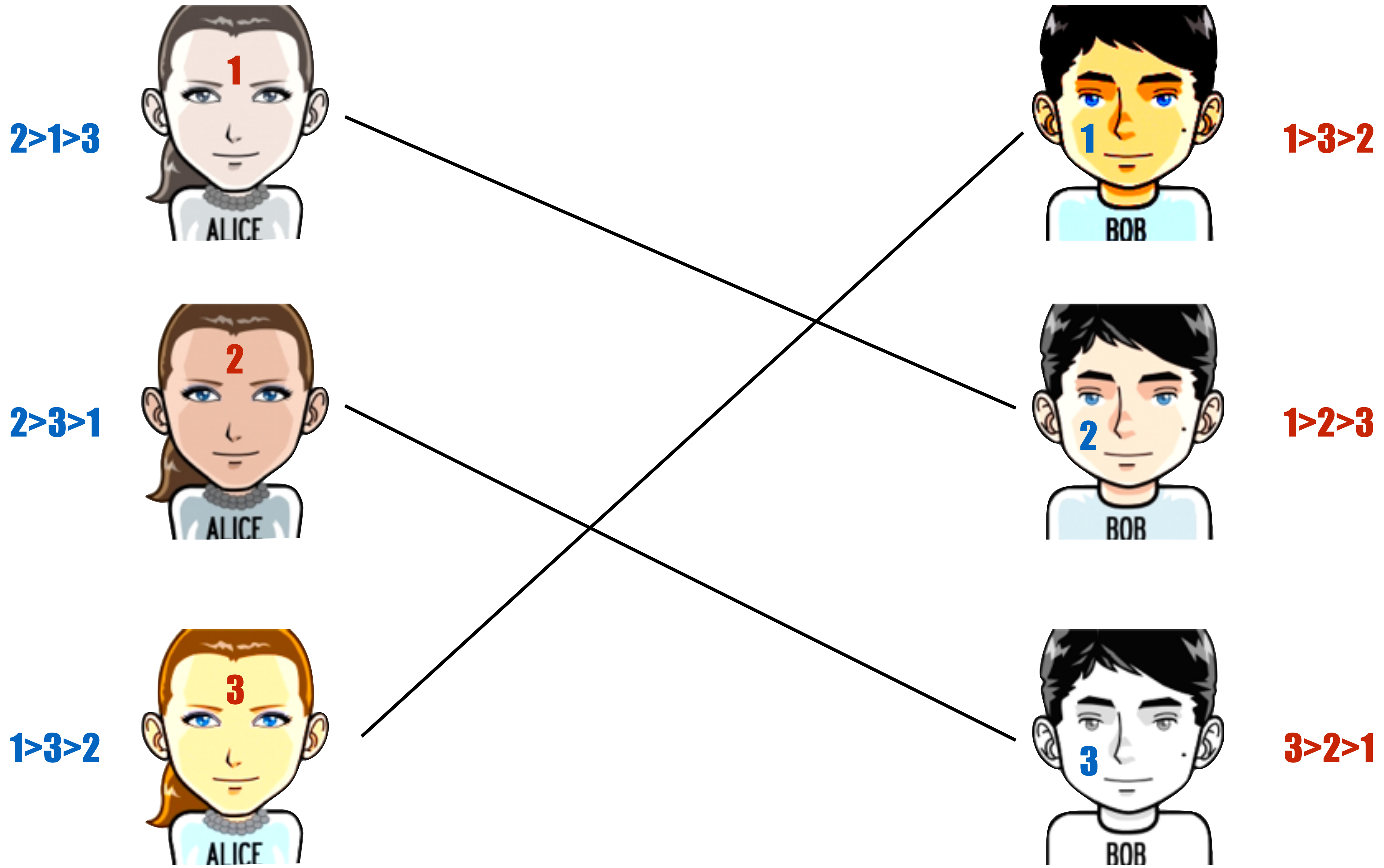
1>3>2

We seek a **stable matching** between the two

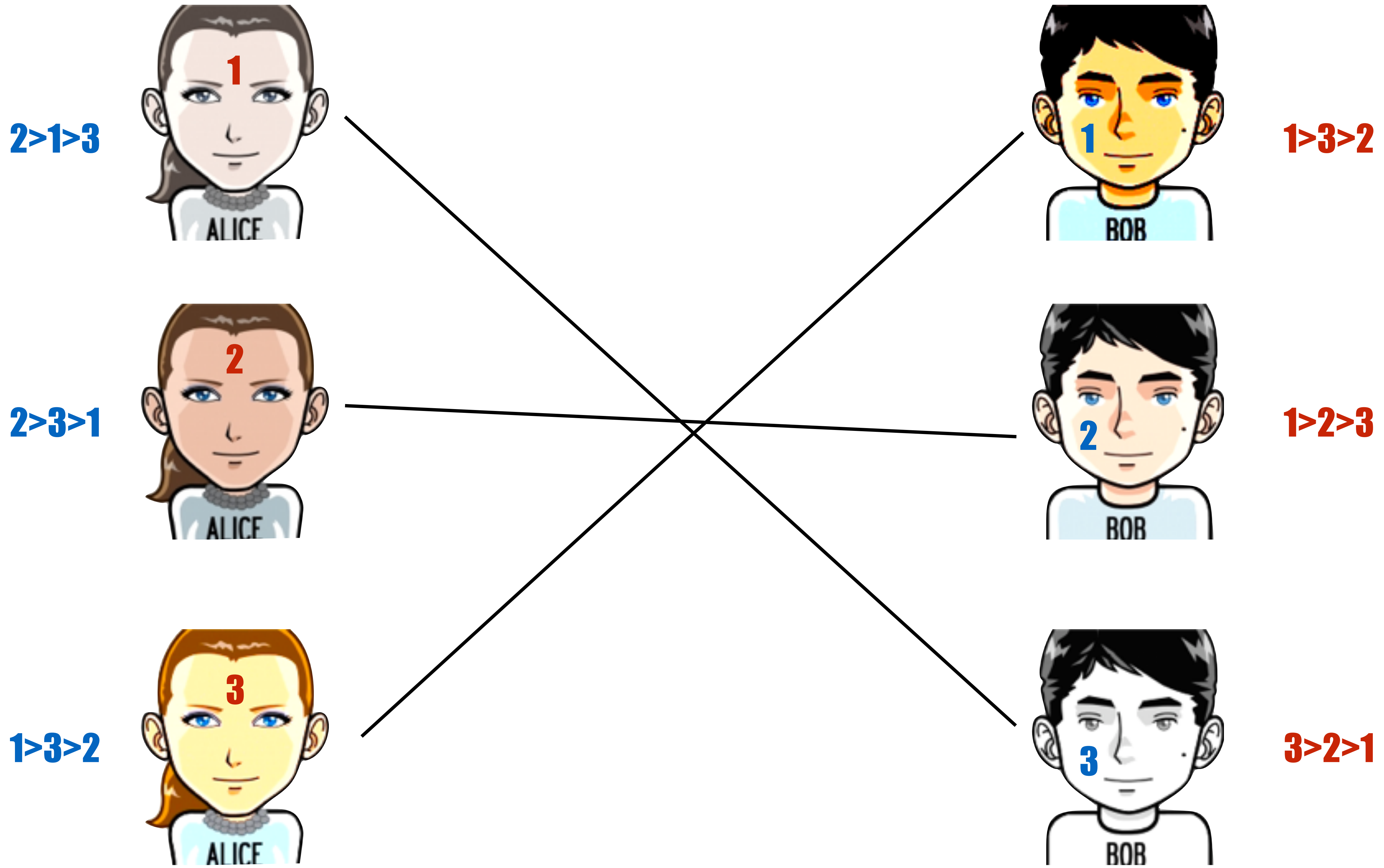
1>3>2

1>2>2

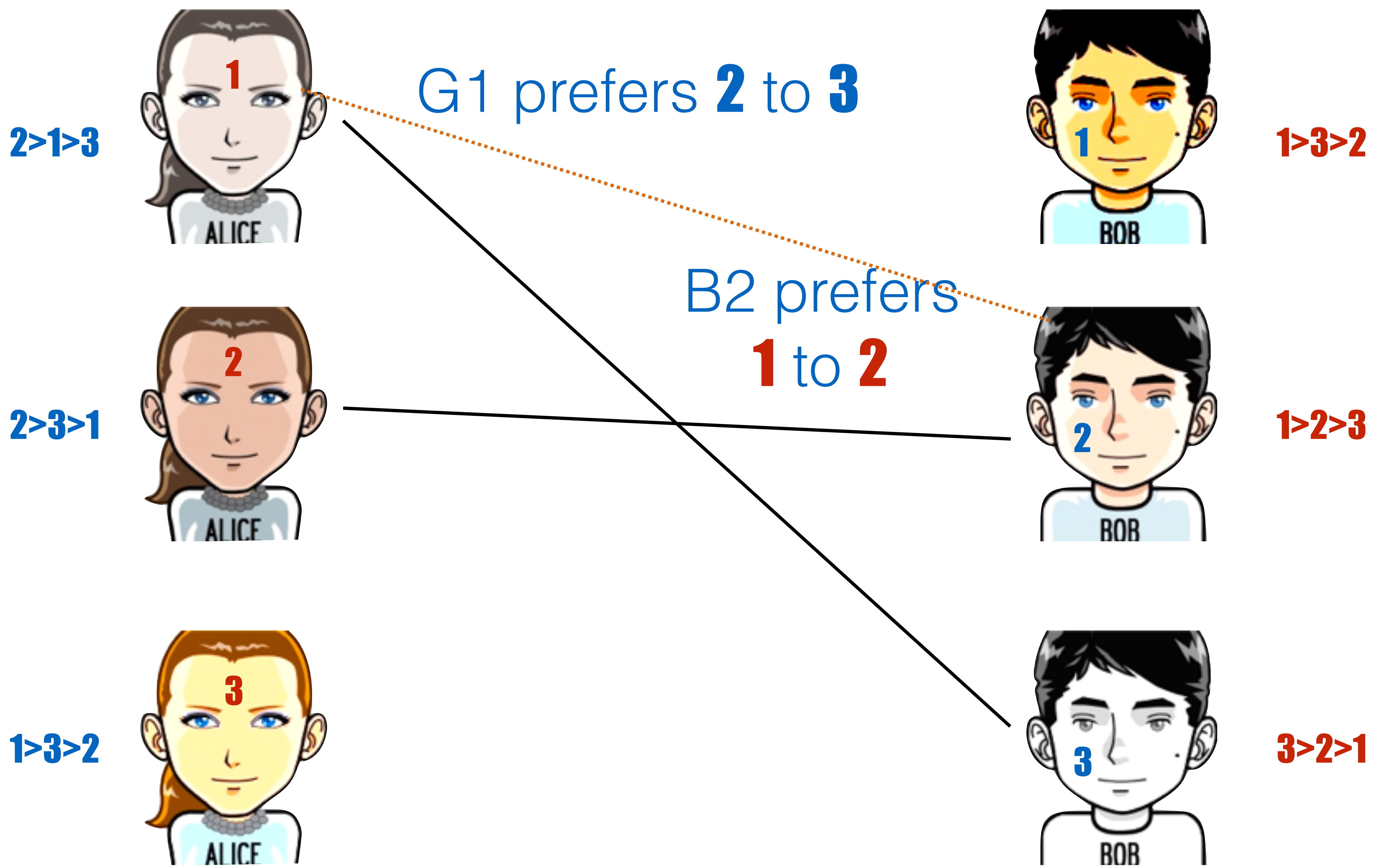
3>2>1



2>1>3
1
ALICE
2>3>1
2
ALICE
1>3>2
3
ALICE
1
BOB
1>3>2
2
BOB
1>2>3
3
BOB
3>2>1



Unstable Matching

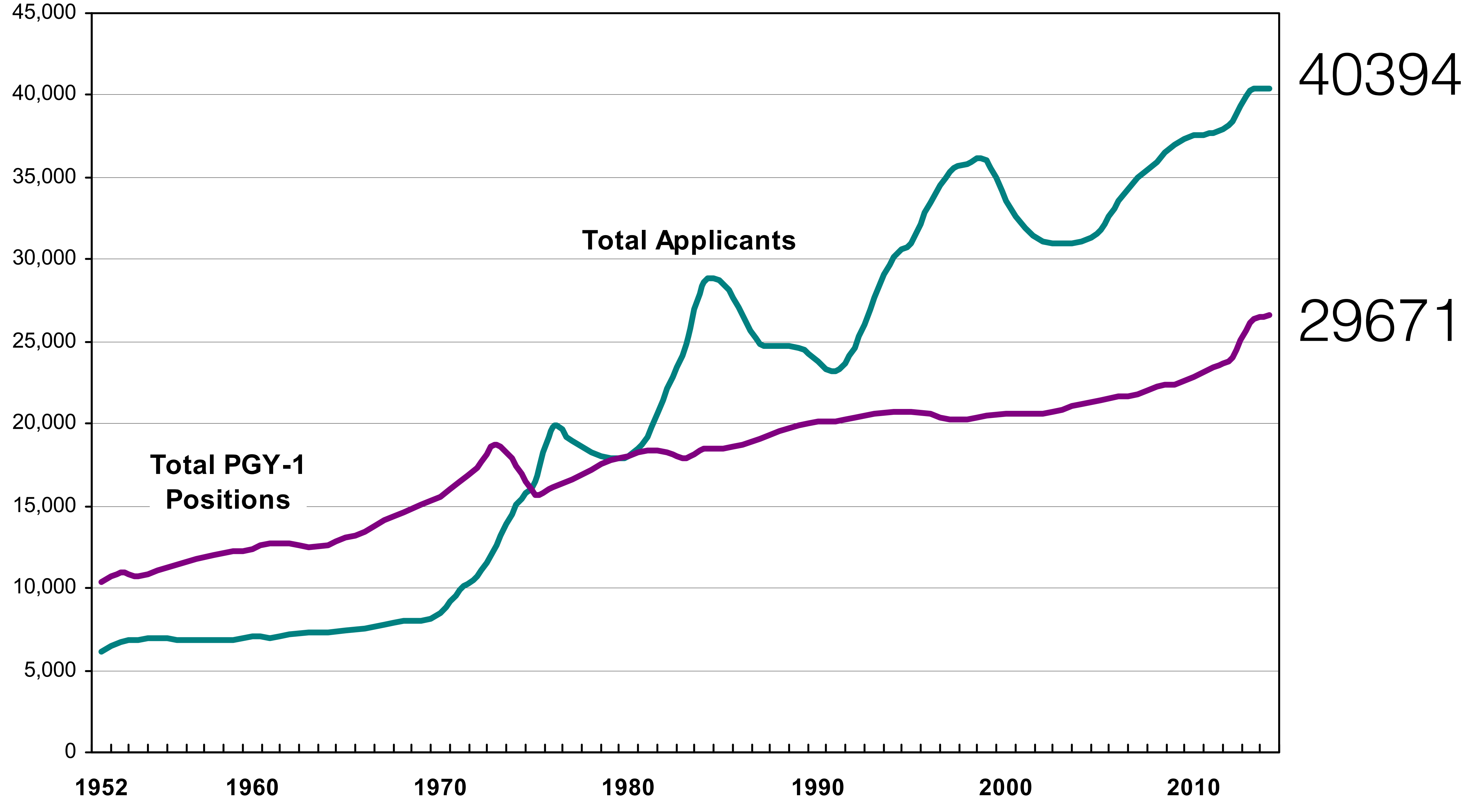


Unstable Matching

Stable Matching

Stable
matching has
many practical
applications

Figure 1 Applicants and 1st Year Positions in The Match, 1952 - 2014



Applicant Type	Matched		
	2013 Graduates	Prior Year Graduates[1]	Total
CMG	2571	74	2645
IMG	146	353	499
USMG	23	2	25
TOTAL	**2740**	**429**	**3169**

UNIVERSITAS OSLOENSIS
MDCCCXI

NUS
National University
of Singapore

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Established in collaboration with MIT

NANYANG
TECHNOLOGICAL
UNIVERSITY


GreekYearbook
www.GreekYearbook.com
University of Virginia
Chi Omega Bid Day 2012

Definition: matchings

M=

W=

S=

Definition: matchings

$$M = \{m_1, \ldots, m_n\}$$
$$W = \{w_1, \ldots, w_n\}$$

$$S = \{(m_{i_1}, w_{j_1}), \ldots, (m_{i_k}, w_{i_k})\}$$

Each $\mathtt{m_i}$ $(\mathtt{w_i})$ appears only one in a pairing.

A matching is perfect if every $\mathtt{m_i}$ appears.

Image credits: Julia Nikolaeva

Definition: preferences

$$M = \{m_1, \ldots, m_n\}$$

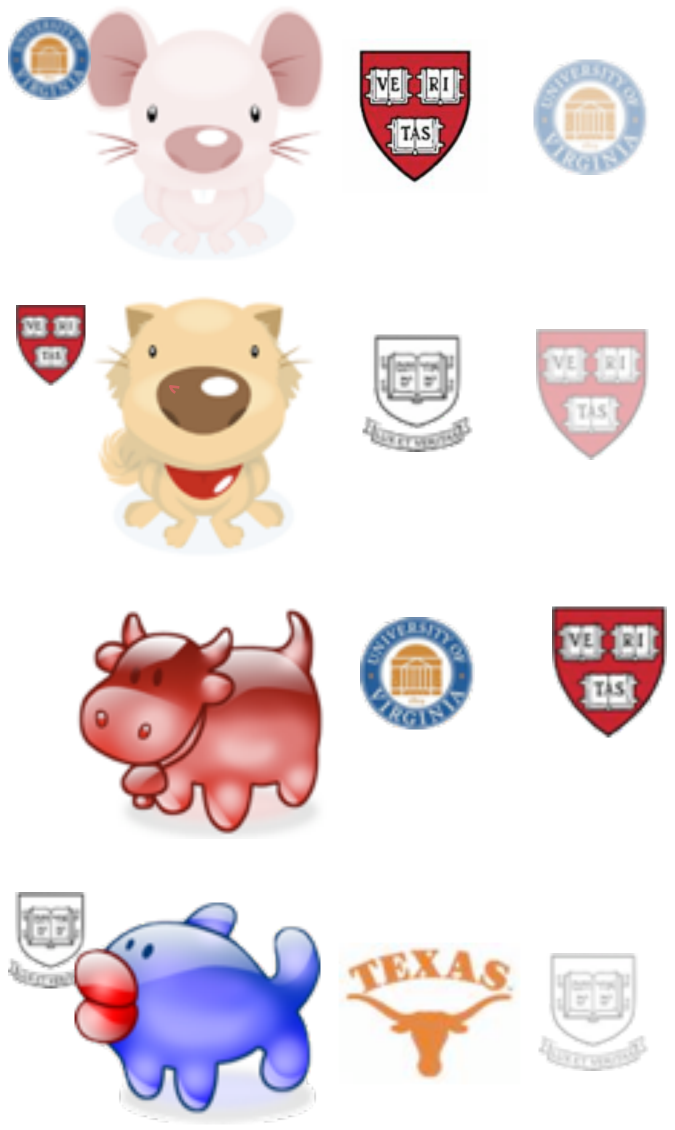


Image credits: Julia Nikolaeva

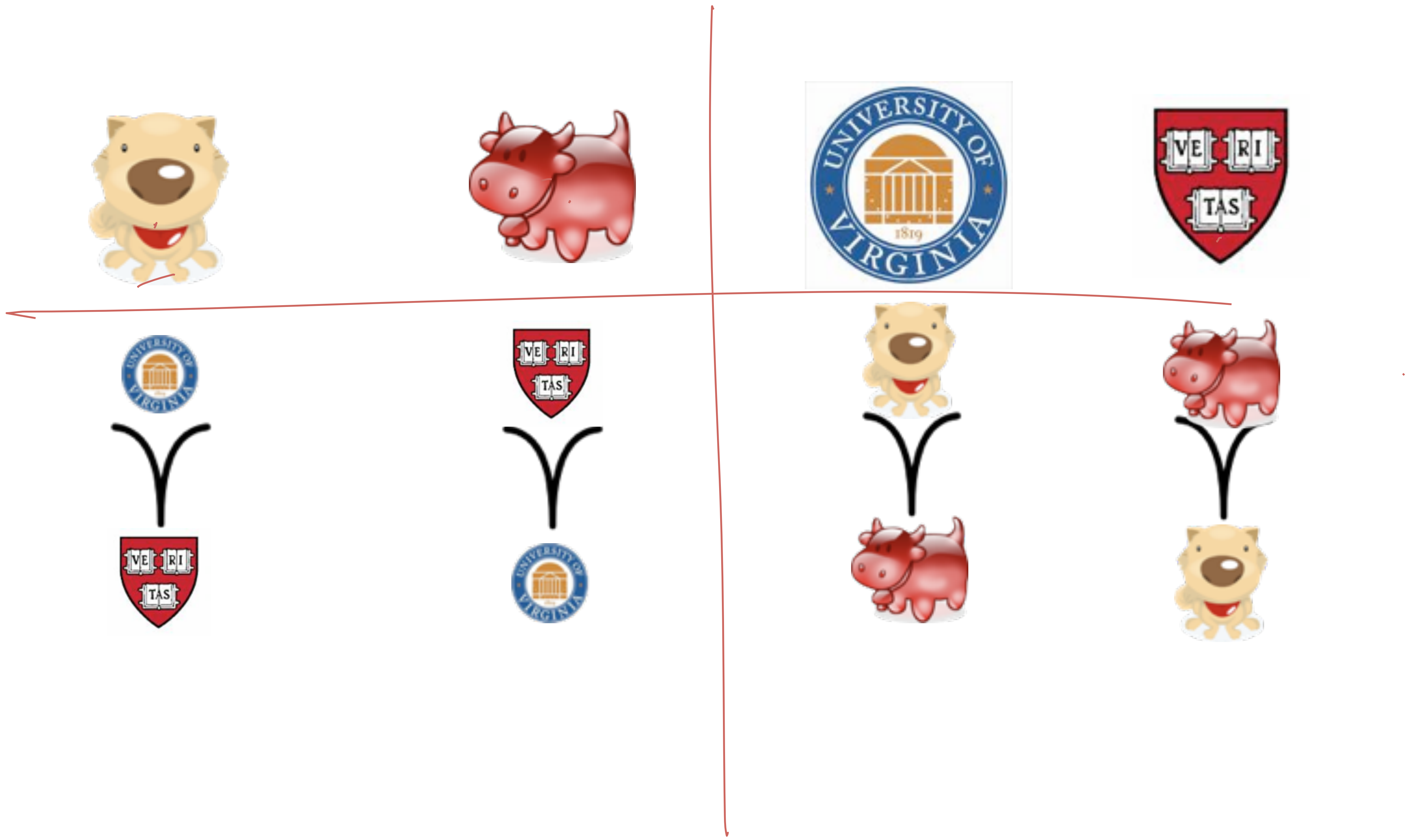
Example: preferences

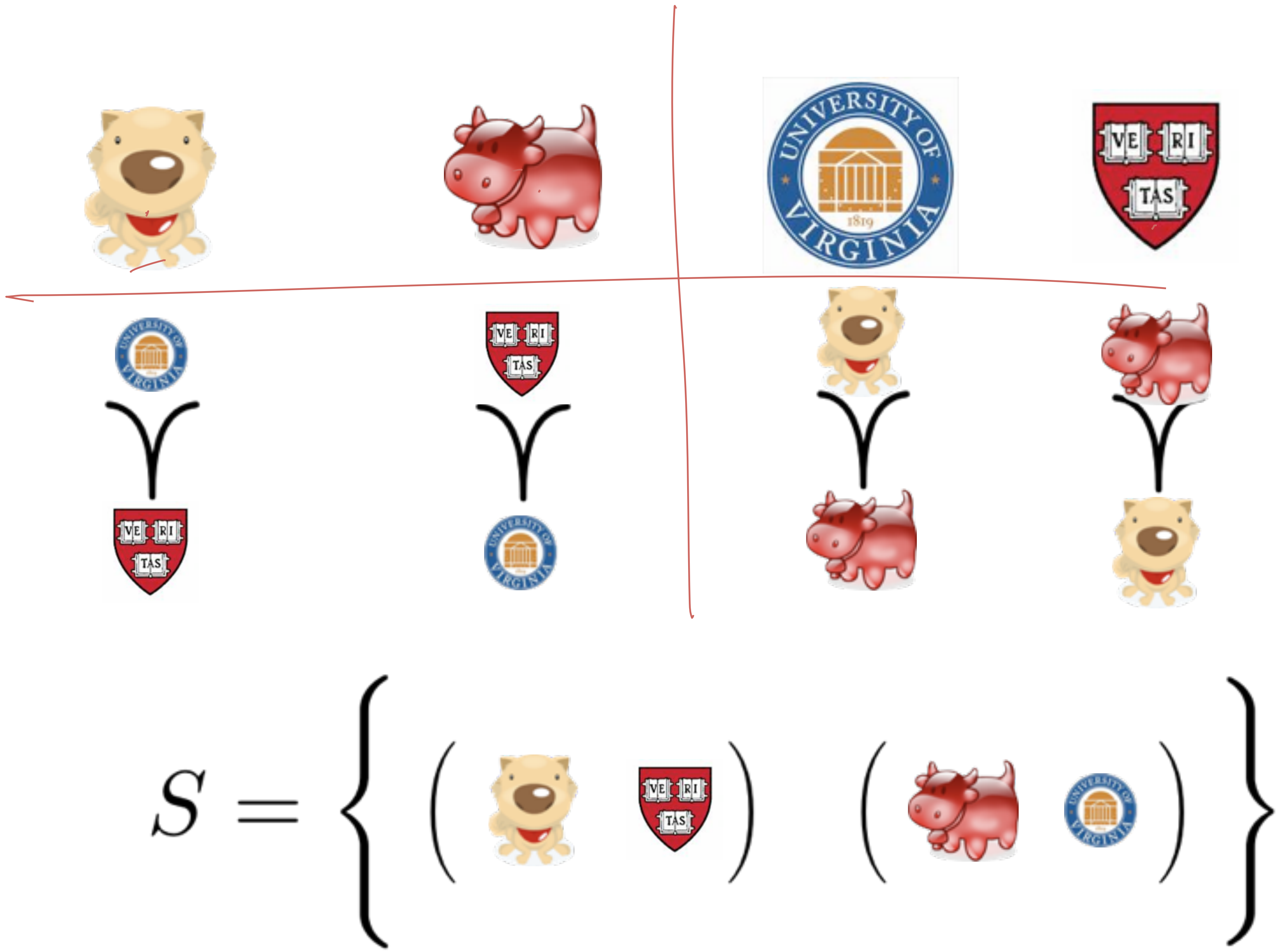
$$M = \{m_1, \ldots, m_n\}$$

m_i has a preference relation $\prec_{m_i}$ on the set W

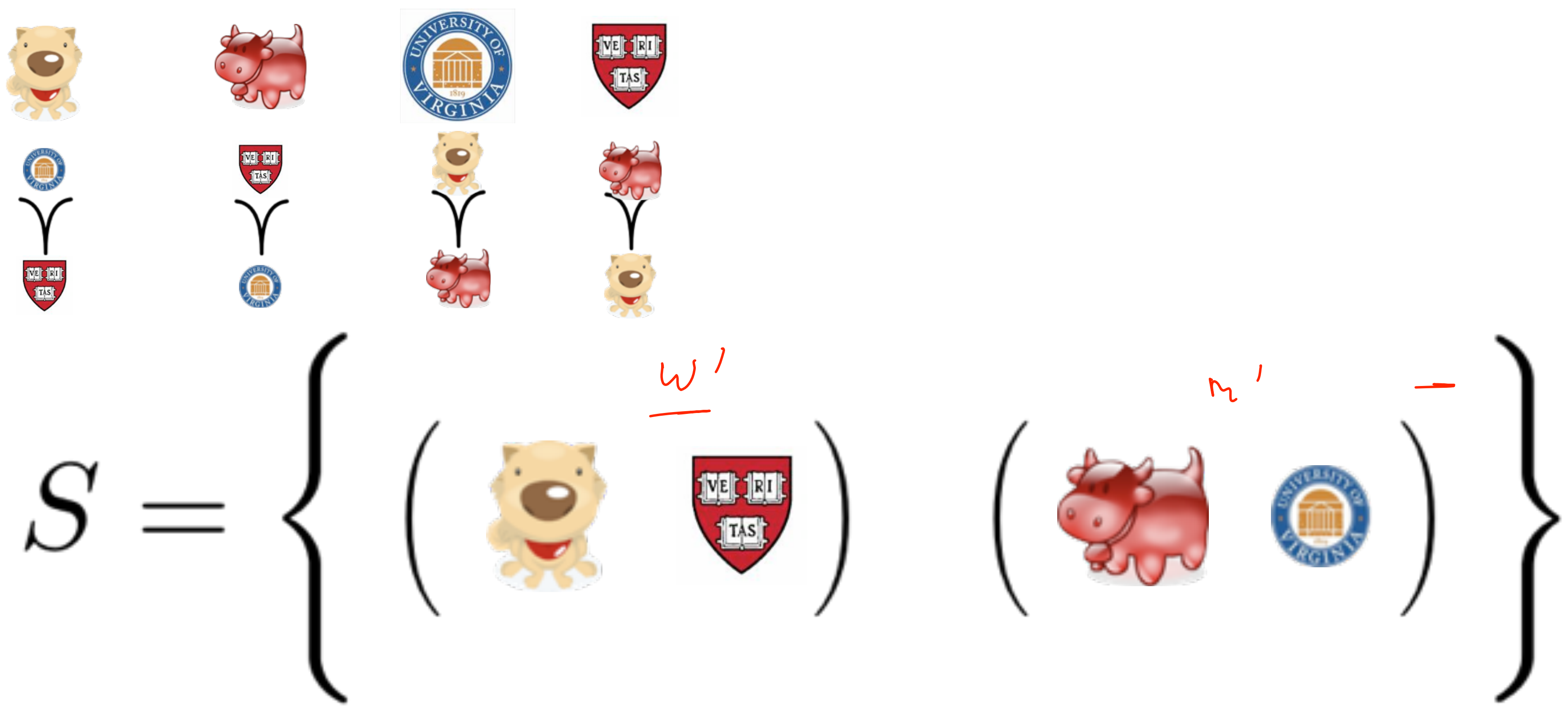
$$w_1 \prec_{m_i} w_4 \prec_{m_i} w_2 \prec_{m_i} w_8 \cdots w_n$$

$$\prec \quad \prec$$

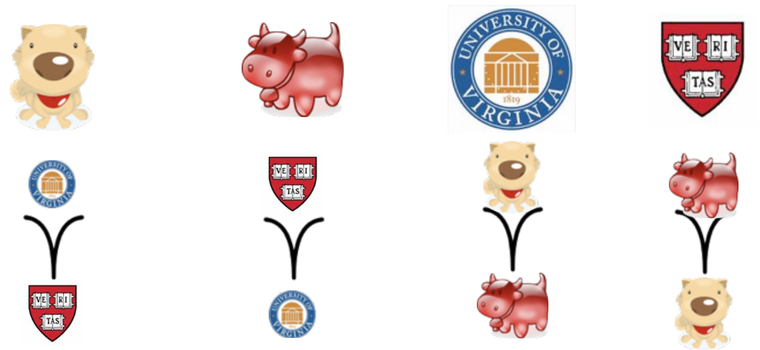

UNIVERSITY OF VIRGINIA
1819
VE RI
TAS

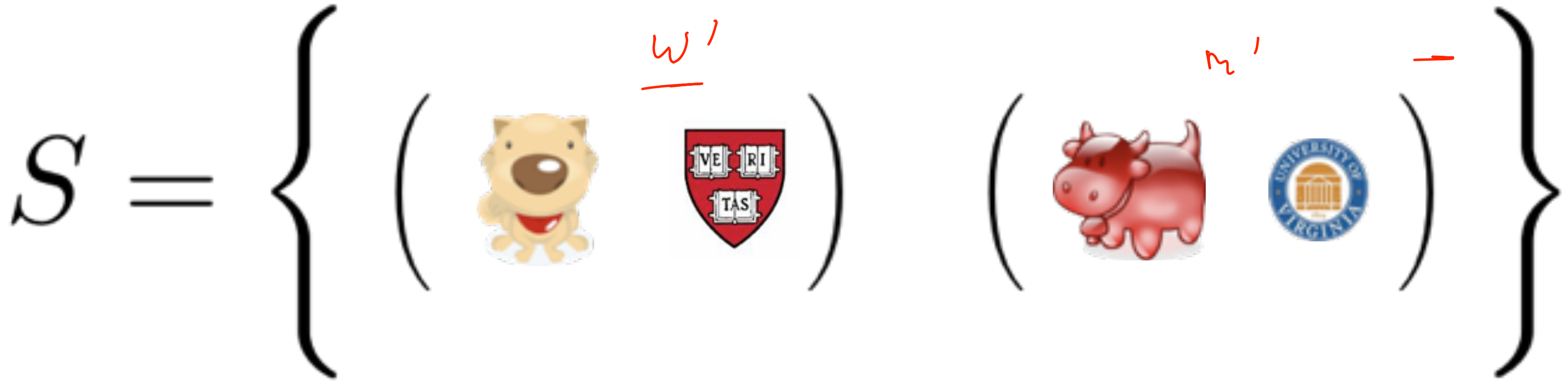

S =

Def: instability



Def: instability





$(m^*, w^*) \notin S$

$w' \prec_{m^*} w^*$

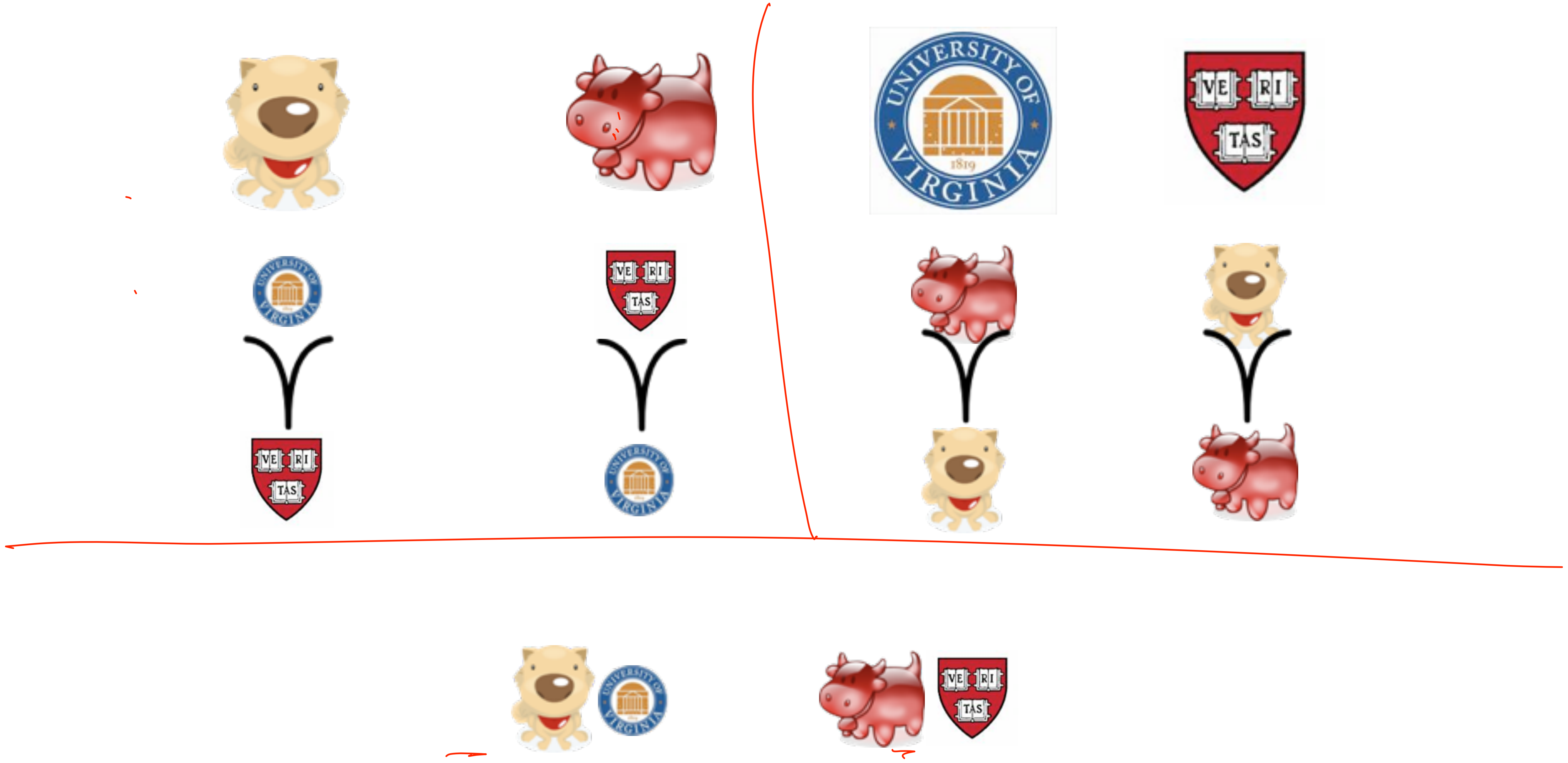
$m' \prec_{w^*} m^*$

M $= \{ (s_1,r_1), (s_2,r_2), \ldots (s_n,r_n) \}$

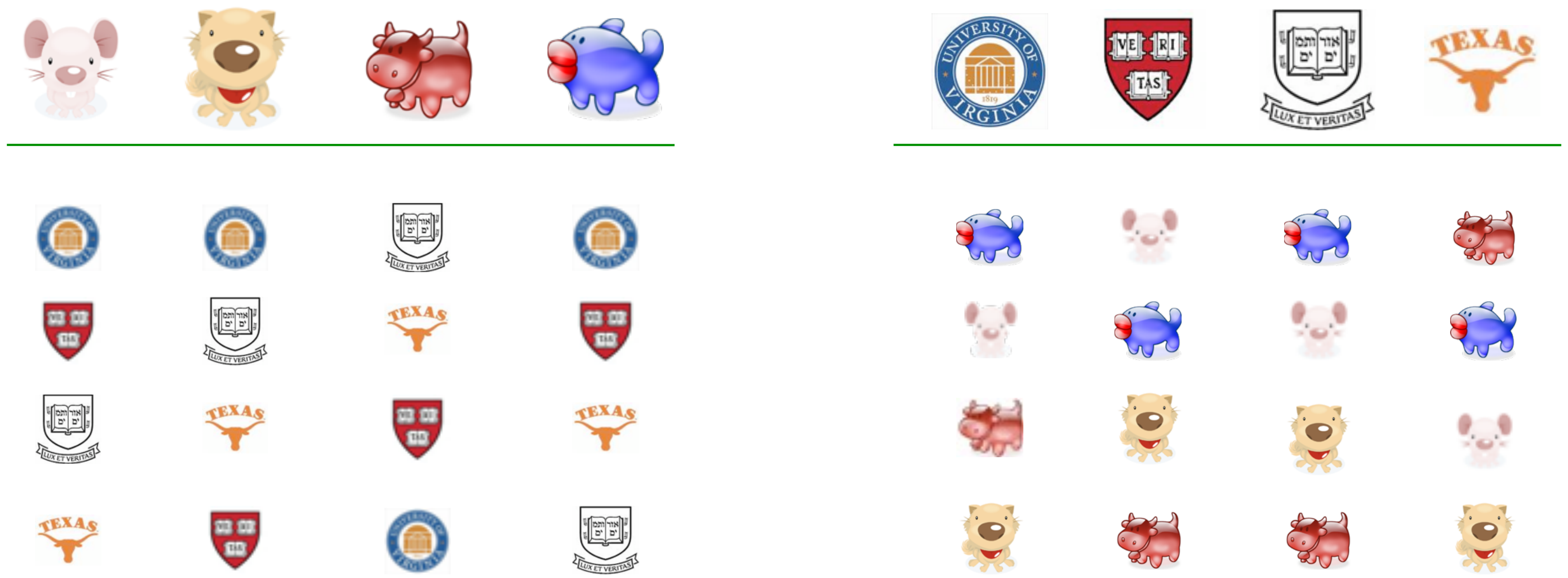
is a stable matching if

No unmatched pair (s^*,r^*) prefer each other to their partners in M

Example 2



Prove: for every input



there exists a stable matching.

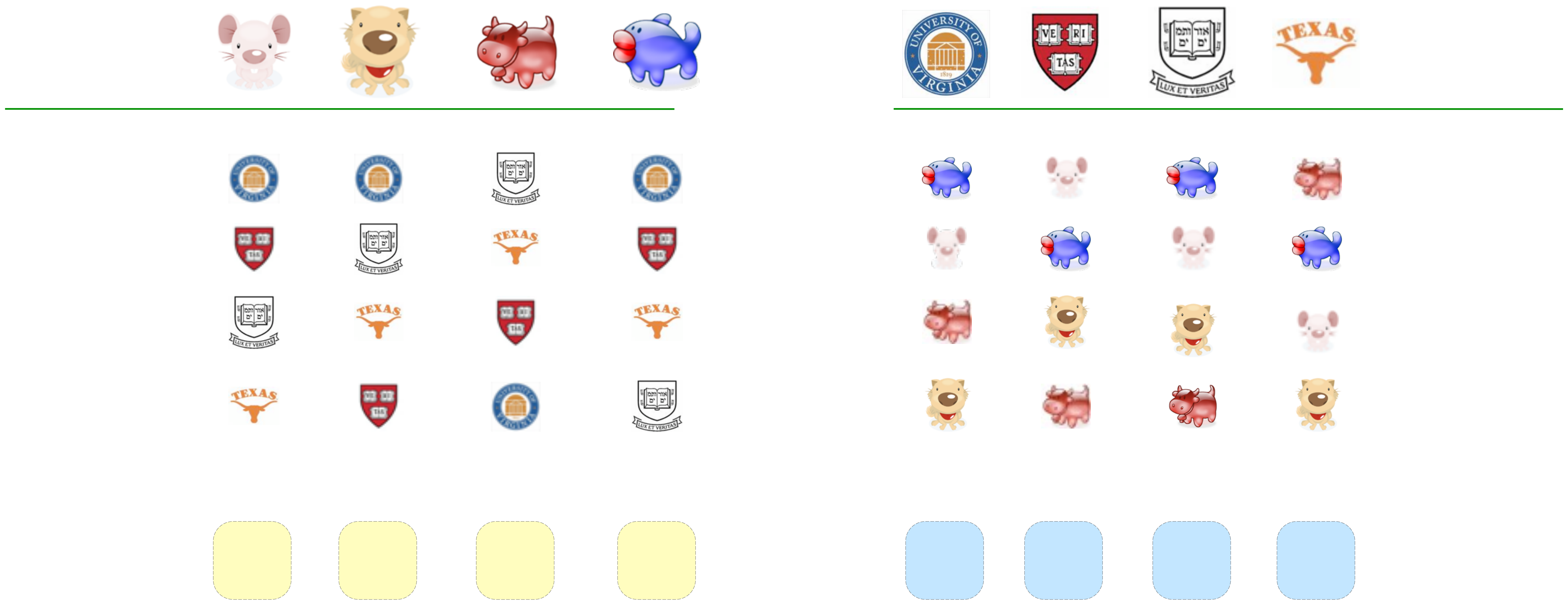
proposal algorithm

StableMatch($M, W, \prec_m, \prec_w$)

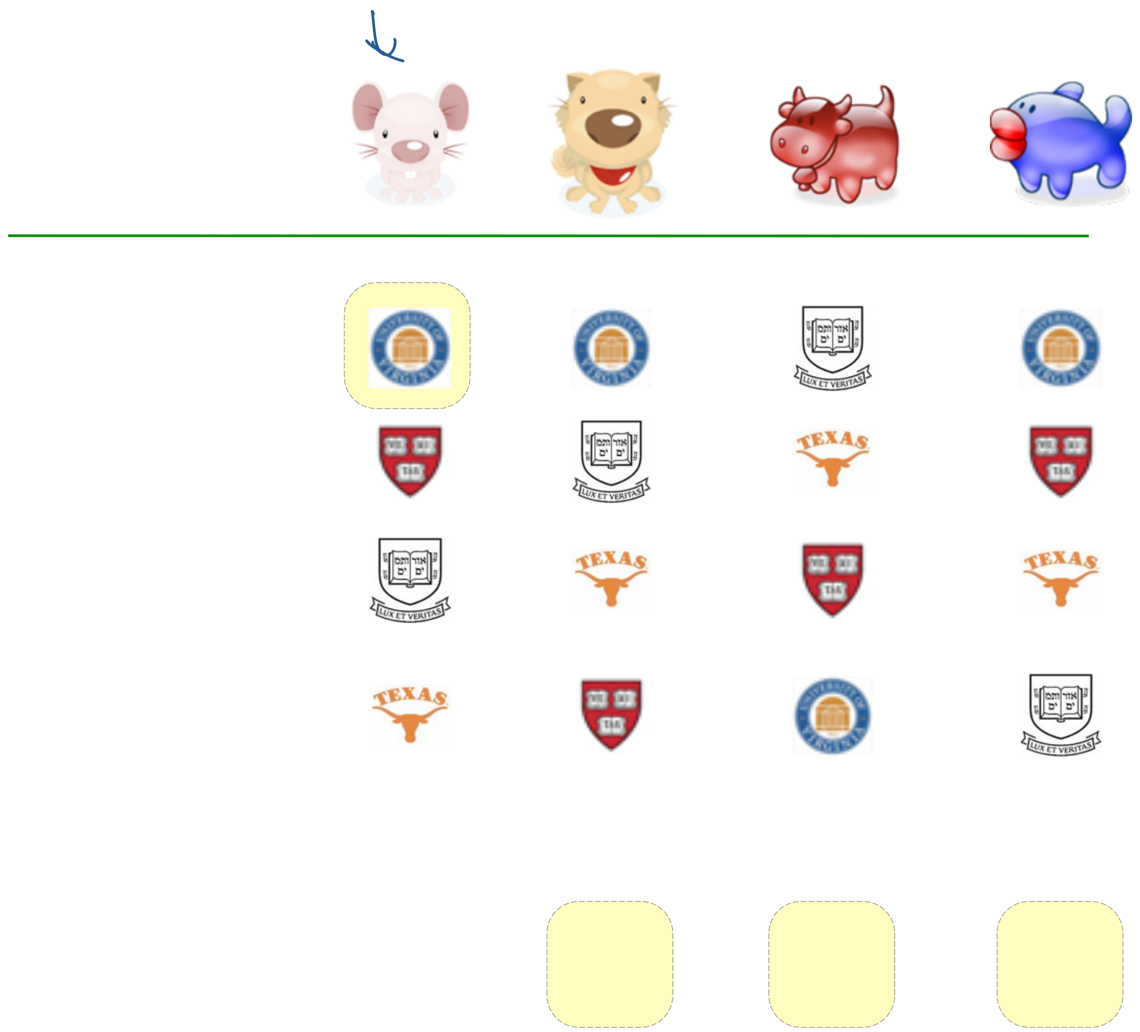
Initialize all m, w to be free
while $\exists$free(m) and hasn't proposed to all W
    do Pick such an m
        Let $w \in W$ be highest-ranked to whom m has not yet proposed
        if free(w)
          then Make a new pair (m, w)
        elseif (m', w) is paired and $m' \prec_w m$
            do Break pair (m', w) and make m' free
                Make pair (m, w)
return Set of pairs

S

R



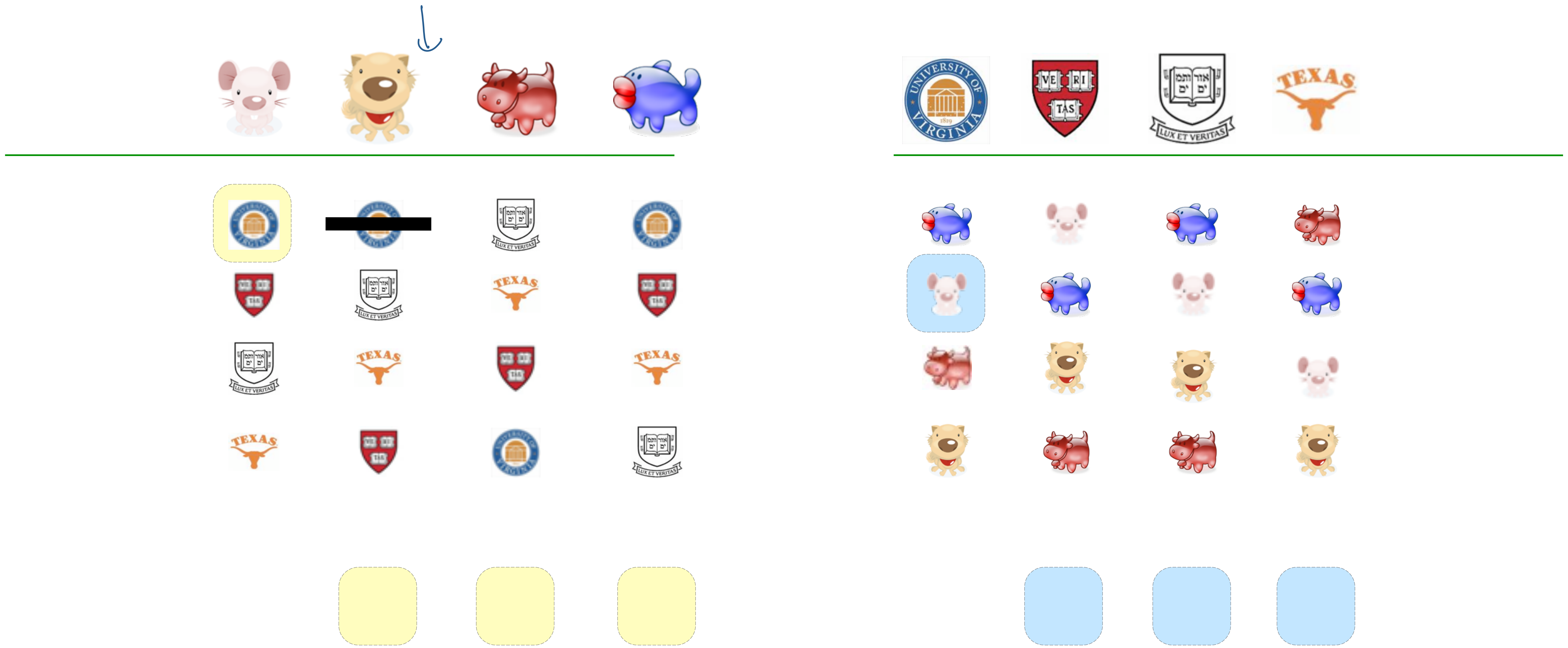
S



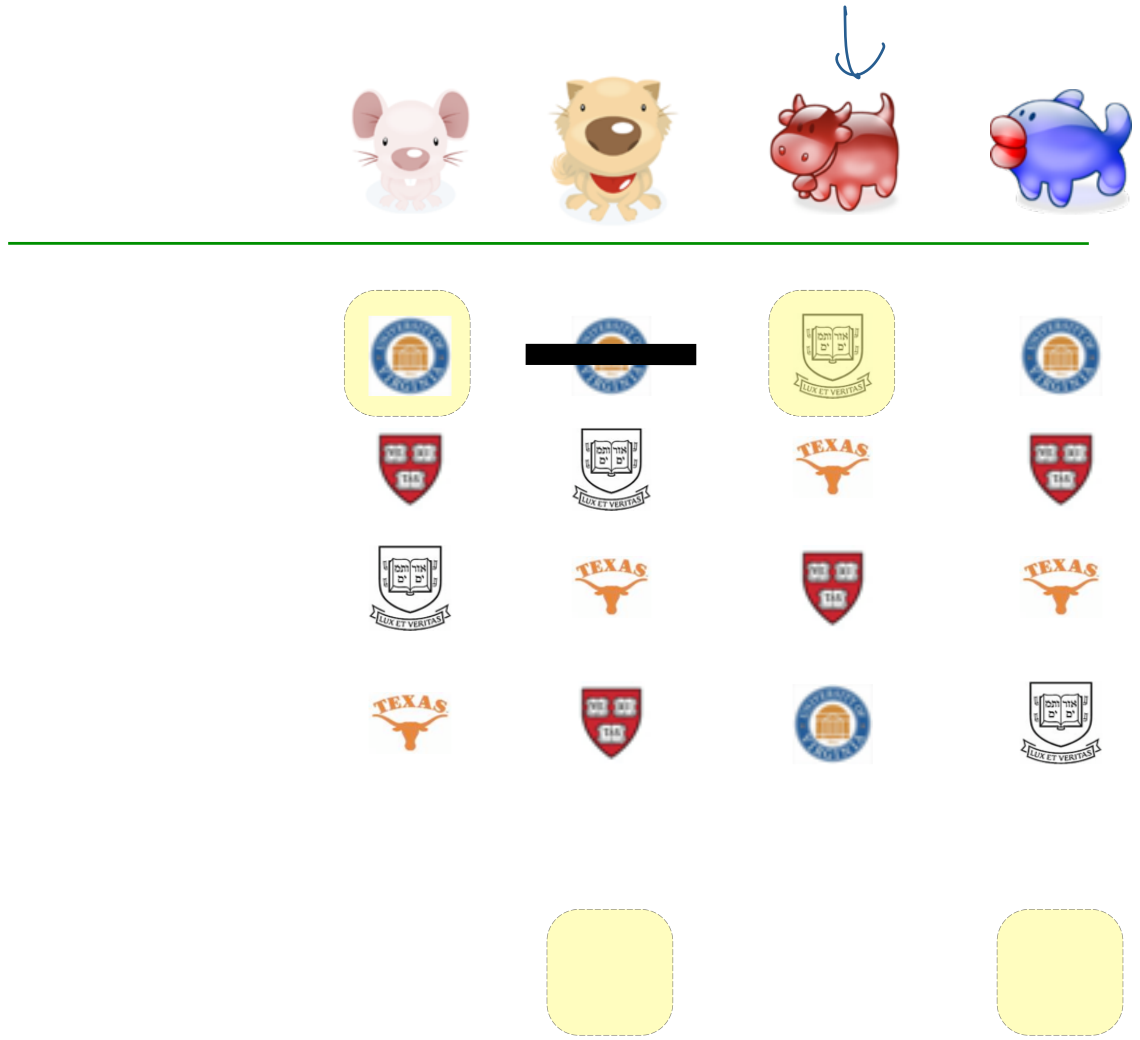
R

S

R



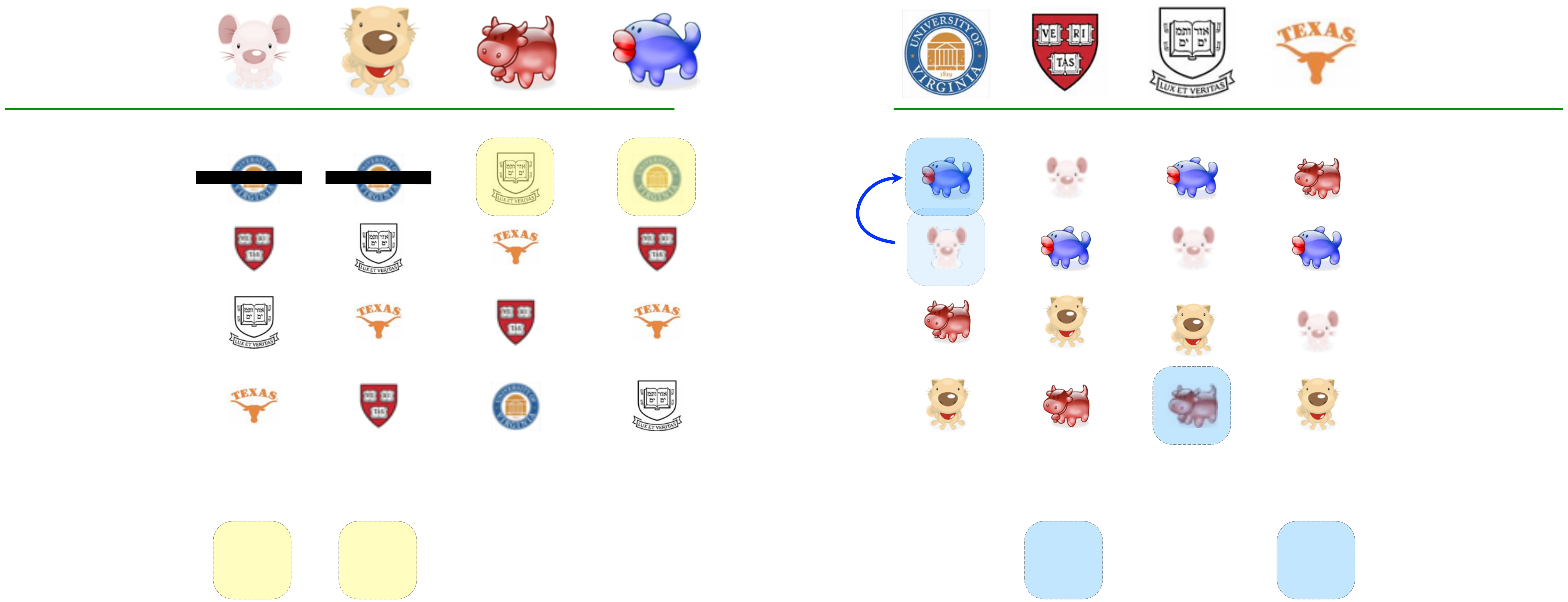
S



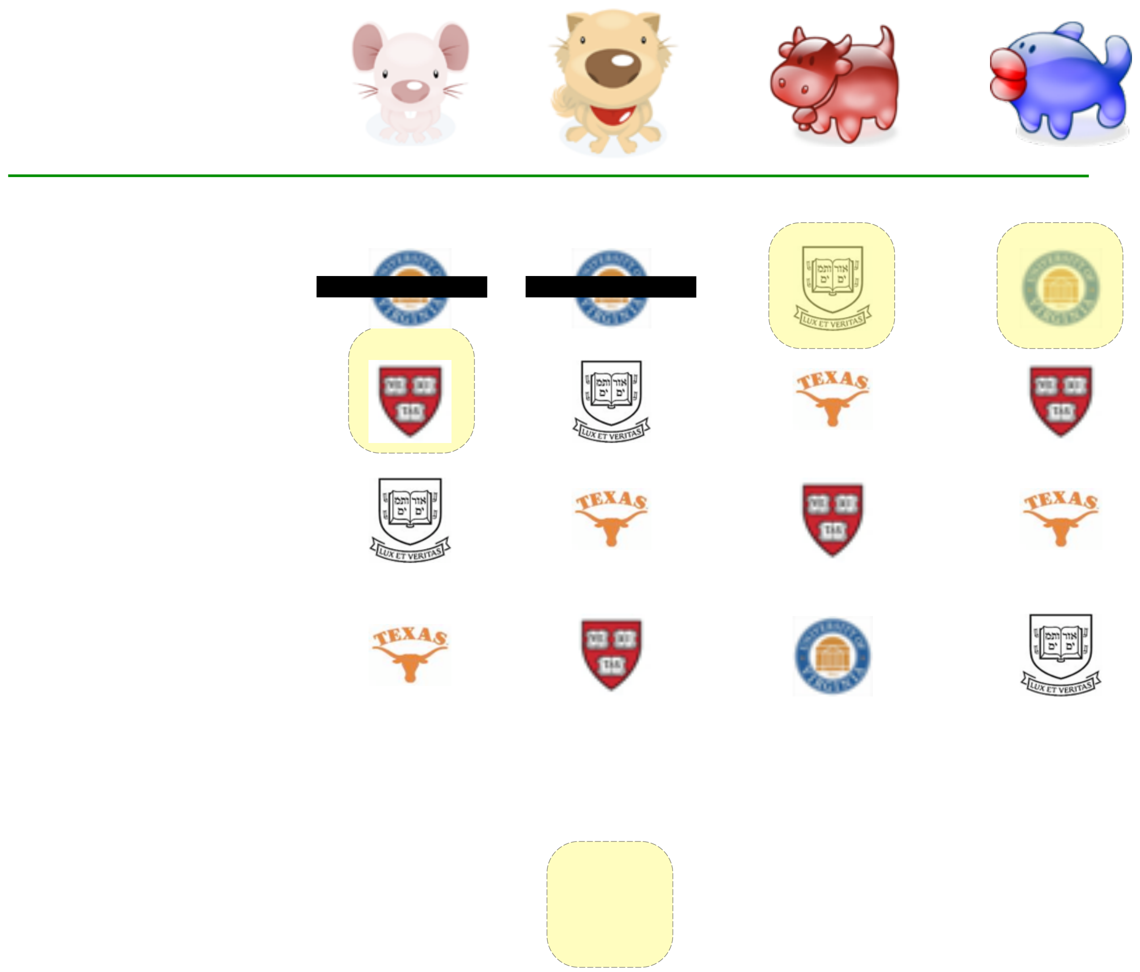
R

S

R

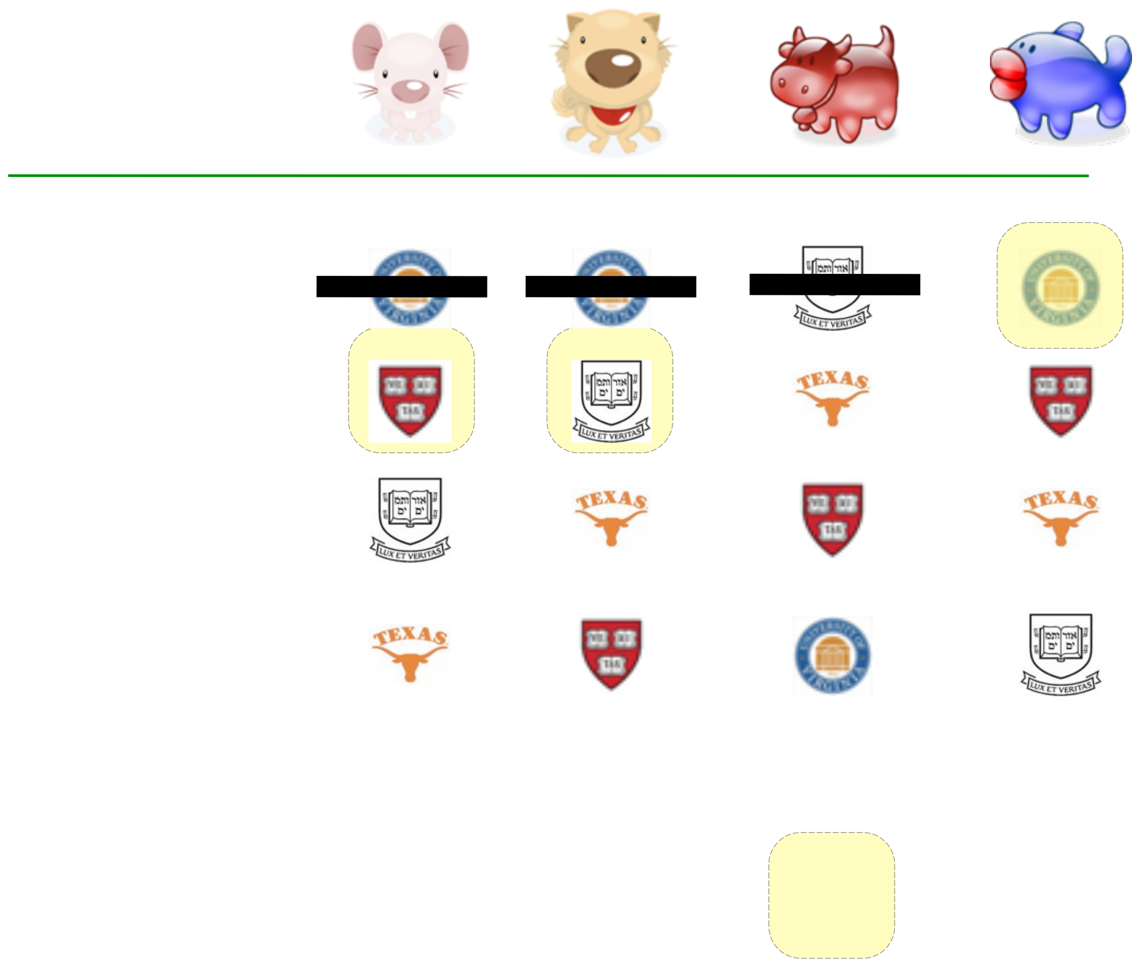


S



R

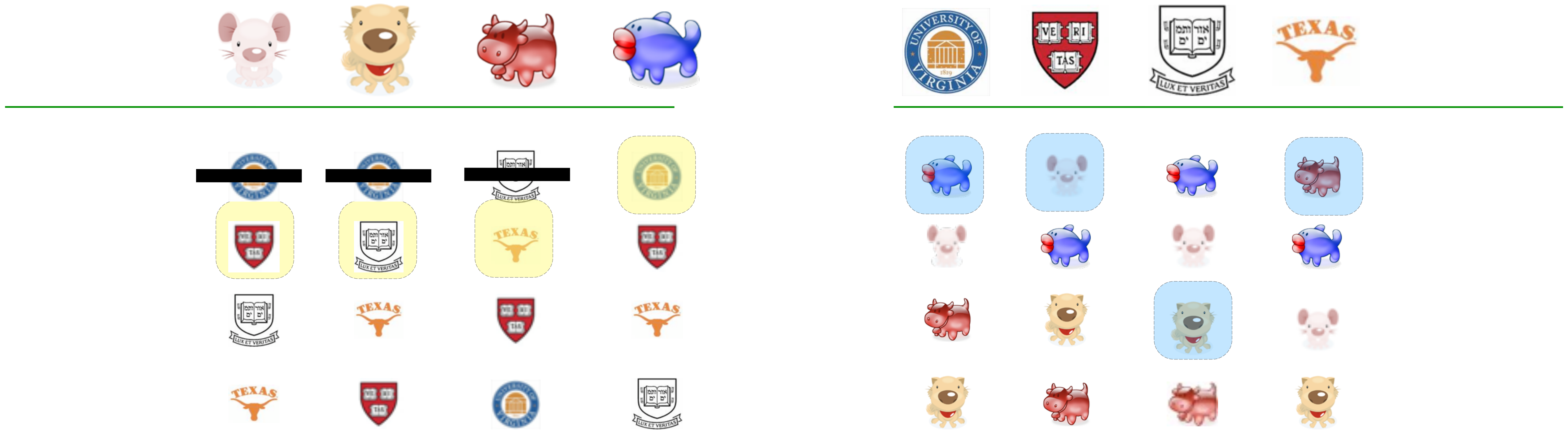
S



R

S

R



Proposal algorithm ends

proposal algorithm ends

$O(n^2)$ steps

each m proposes at most once to each w.

each m proposes at most n times.

size of M is n.

output is a matching

$\textsc{StableMatch}(M, W, \prec_m, \prec_w)$

Initialize all m, w to be FREE
while $\exists\textsc{free}(m)$ and hasn't proposed to all W
    do Pick such an m
        Let $w \in W$ be highest-ranked to whom m has not yet proposed
        if $\textsc{free}(w)$
            then Make a new pair (m, w)
        elseif (m', w) is paired and $m' \prec_w m$
                do Break pair (m', w) and make m' free
                    Make pair (m, w)
return Set of pairs

StableMatch$(M, W, \prec_m, \prec_w)$

Initialize all m, w to be FREE
while $\exists$FREE(m) and hasn't proposed to all W
    do Pick such an m
        Let $w \in W$ be highest-ranked to whom m has not yet proposed
        if FREE(w)
          then Make a new pair (m, w)
        elseif (m', w) is paired and $m' \prec_w m$
            do Break pair (m', w) and make m' free
                Make pair (m, w)
return Set of pairs

output is perfect

output is perfect

if $\exists m$ who is free, then

$\exists w$ who has not been asked

output is stable

output is stable

spse not. $\exists (m^*, w), (m, w^*) \in S \quad w \prec_{m^*} w^* \quad m \prec_{w^*} m^*$

output is stable

spse not. $\exists (m^*, w), (m, w^*) \in S$ $w \prec_{m^*} w^*$ $m \prec_{w^*} m^*$

m* last proposal was to w

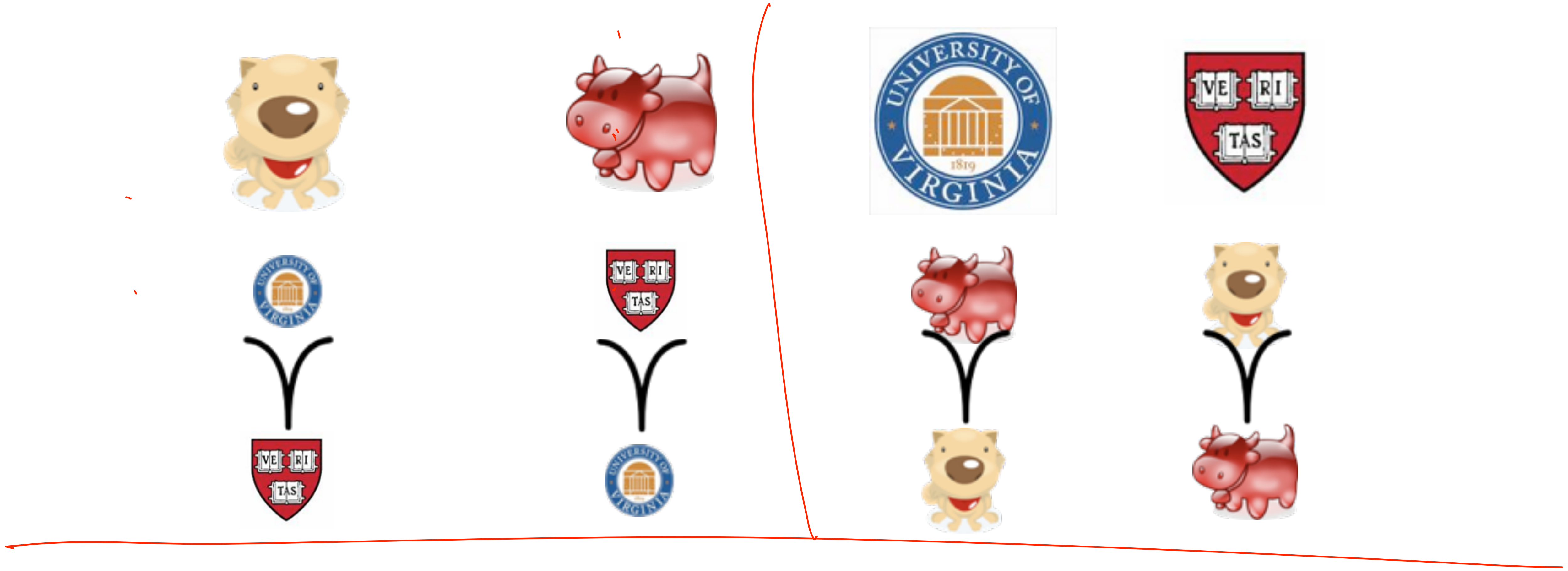
but $w \prec_{m^*} w^*$ and so m* must have already asked w*

and must have been rejected by $m^* \prec_{w^*} m'$

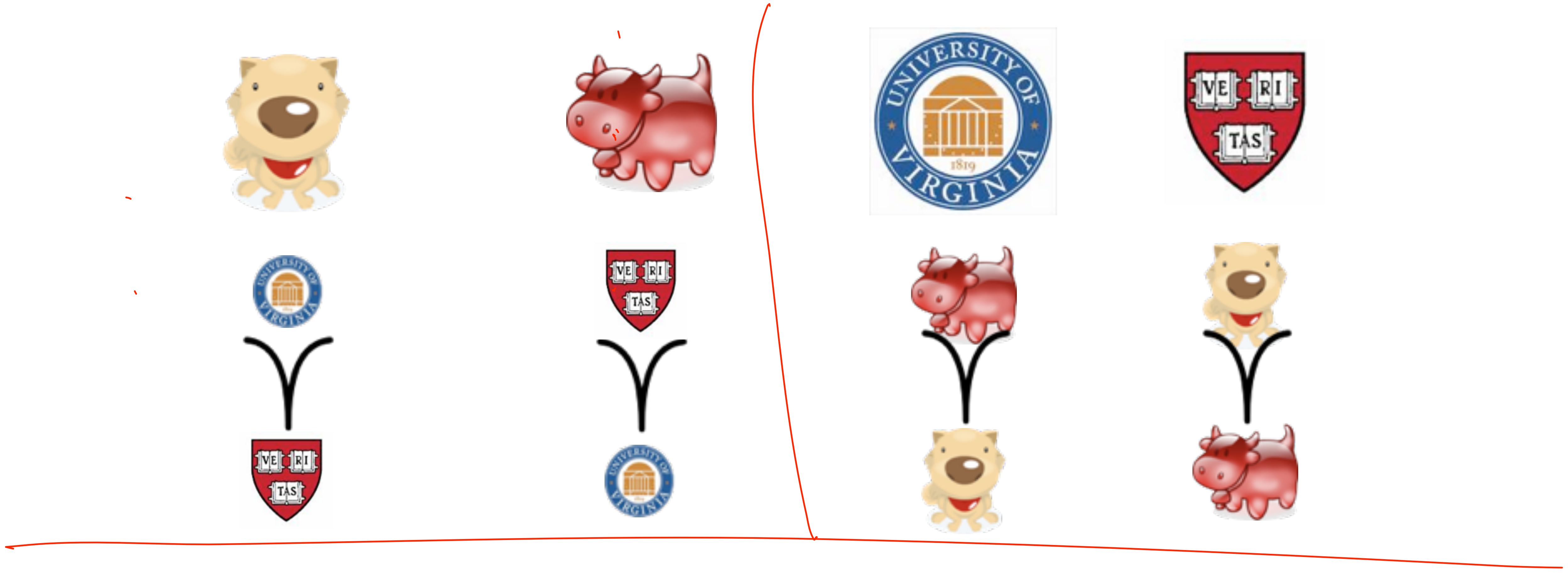
then either $m' \prec_{w^*} m$ or m'=m

which contradicts assumption $m \prec_{w^*} m^*$

Proposer wins



Proposer wins



Remarkable theorem

w is valid for m:

best(m):

GS is Suitor-optimal.

GS matching vs R-opt

S1	S2	S3	S4

R1	R2	R3	R4
S1	S1	S1	S1
S2	S2	S2	S2
S3	S3	S3	S3
S4	S4	S4	S4

S1	S2	S3	S4
R1	R1	R1	R1
R2	R2	R2	R2
R3	R3	R3	R3
R4	R4	R4	R4

R1	R2	R3	R4
S1	S1	S1	S1
S2	S2	S2	S2
S3	S3	S3	S3
S4	S4	S4	S4

S1	S2	S3	S4
R1	R1	R1	R1
R2	**R2**	R2	R2
R3	R3	**R3**	R3
R4	R4	R4	**R4**

R1	R2	R3	R4
S1	S1	S1	S1
S2	**S2**	S2	S2
S3	S3	**S3**	S3
S4	S4	S4	**S4**