L16

4800

11.1.2016

abhi shelat

userid:

Explain how to find a minimum spanning tree:

(Try to just recall from memory, to test how much you understood. Then, look at your notes if you need to.)

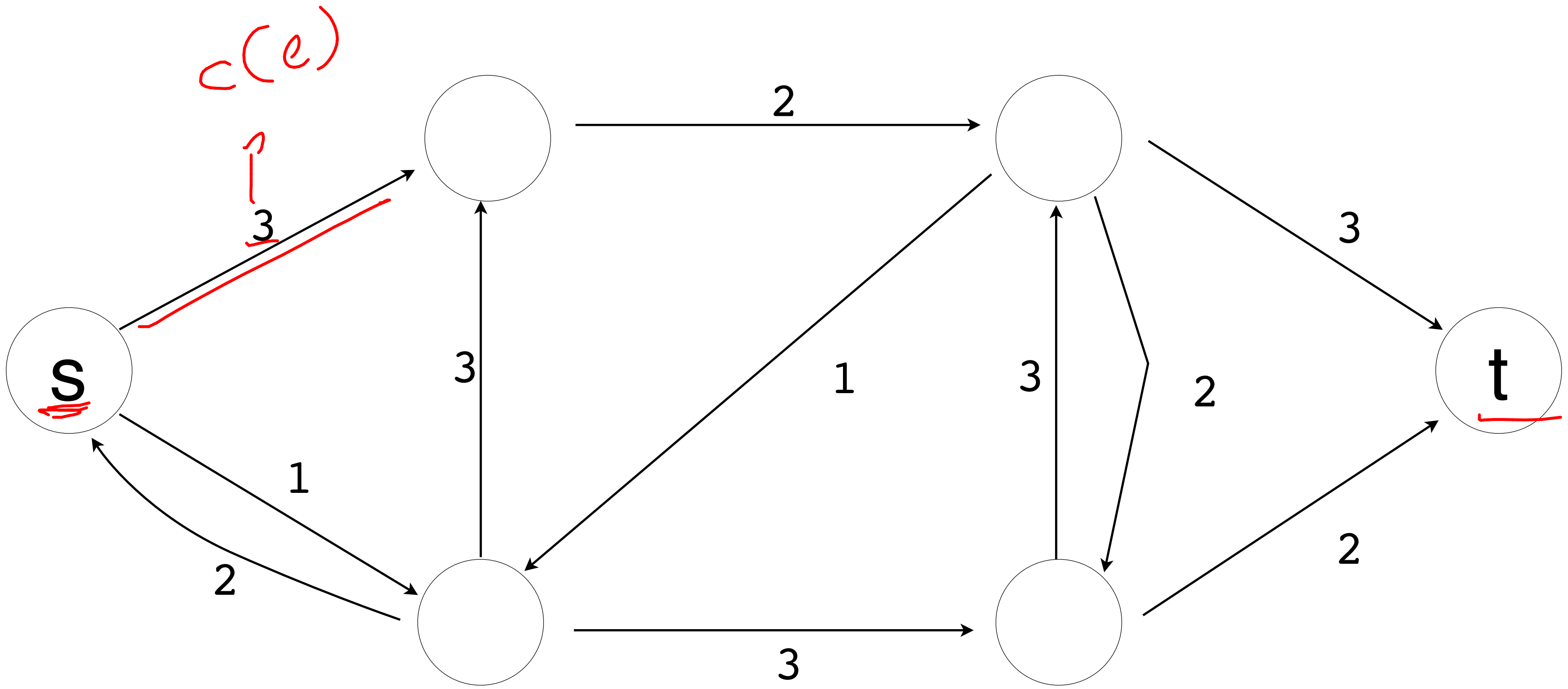
Explain why your method works:

Give 2-3 sentences explaining at a high level.

Max flow

Min Cut

example



flow

$G = (V, E), c$

map from edges to numbers: $f: E \rightarrow \mathbb{R}$

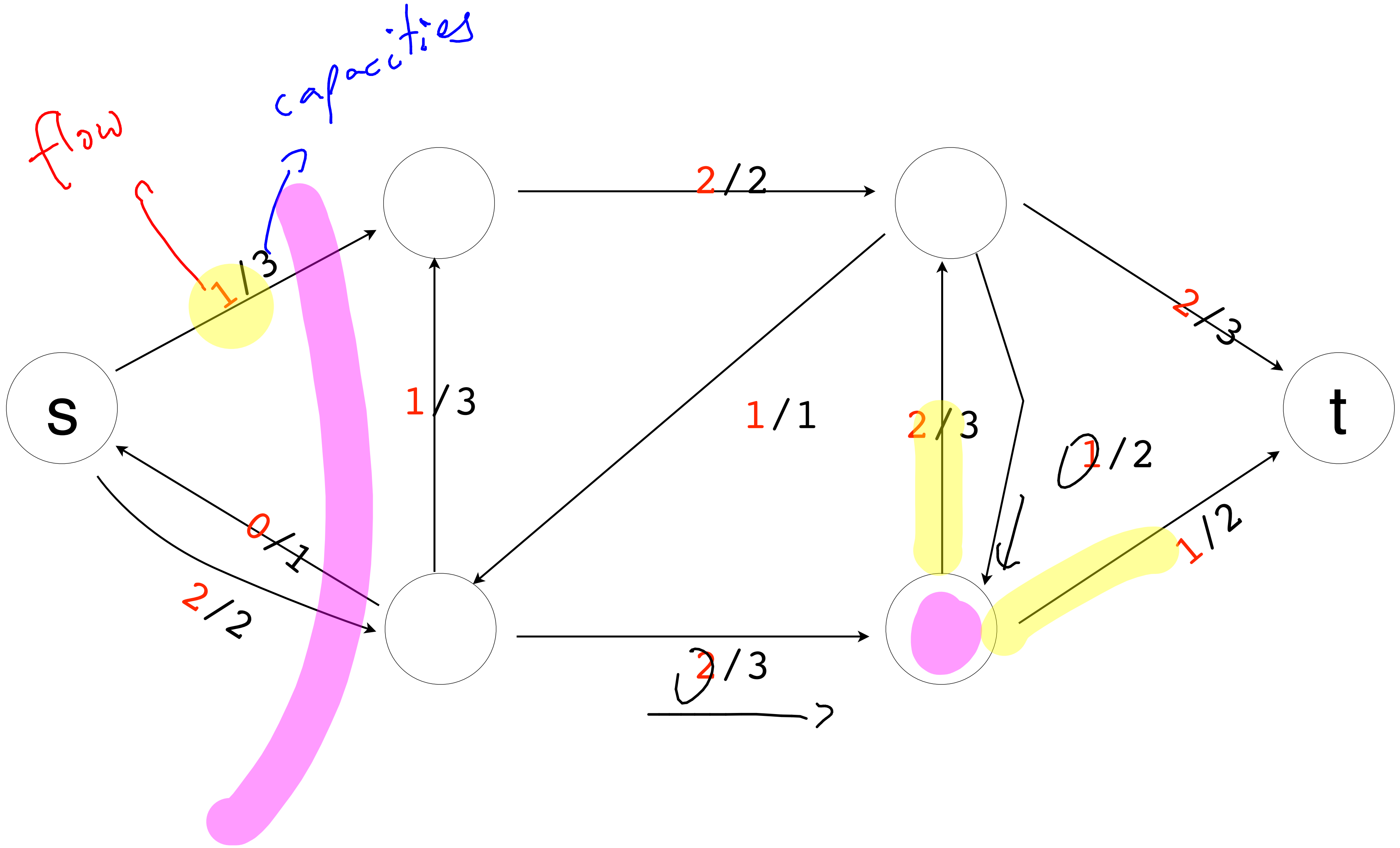
capacity constraint: 1 $f(e) \leq c(e)$

flow constraint: 2 for any $v \in V - \{s,t\}$, $\text{inflow}(v) = \text{outflow}(v)$

$$\sum_{u \in V} f(u,v) = \sum_{w \in V} f(v,w)$$

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

example



$|f| = 1 + 2 = \underline{\underline{3}}$

Residual graphs

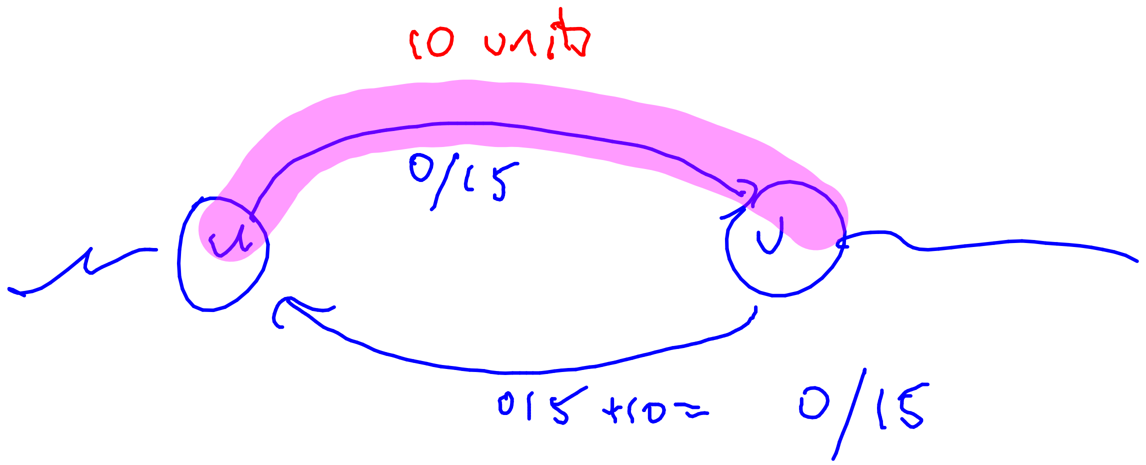
$G_f = (V, E_f)$ w.r.t a flow f

same set of verticies

If an edge $e=(u,v)$ has $f(e) > 0$

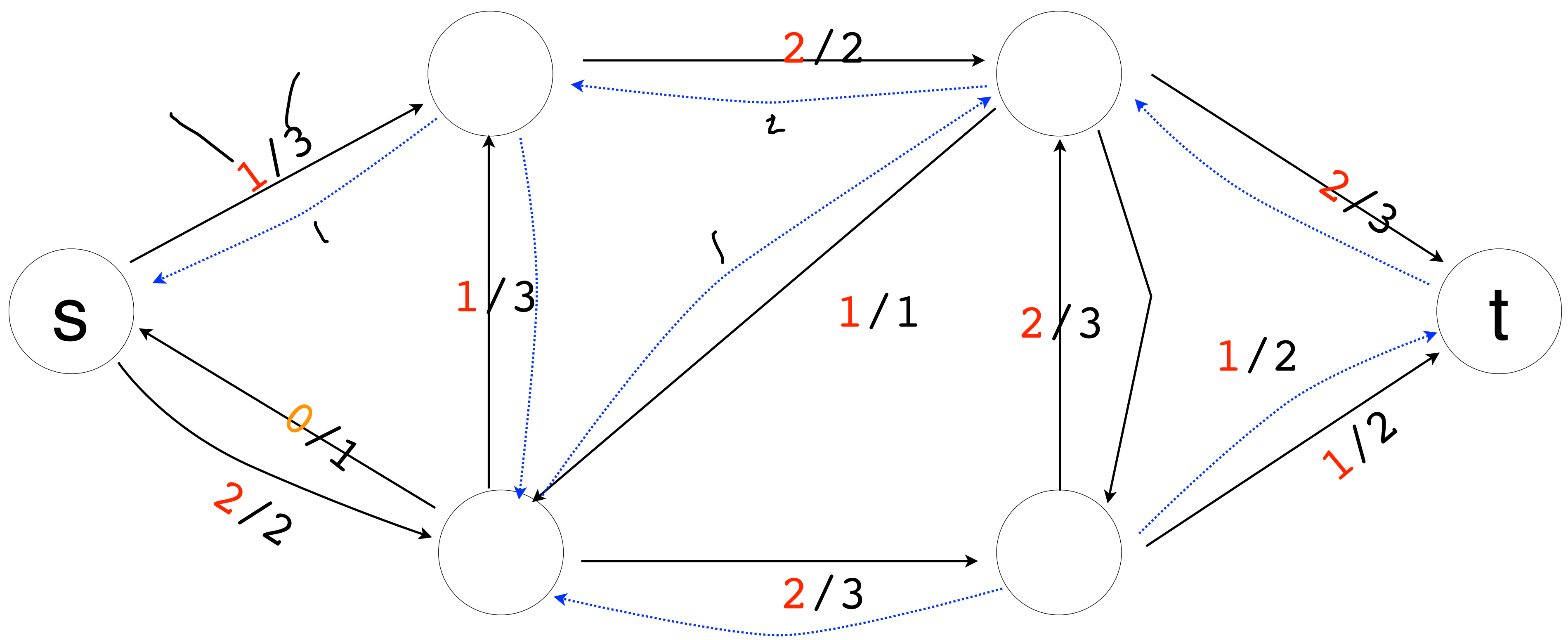
E_f contains the edge (u,v) with new capacity $c(e) - f(e)$

AND it contains the edge (v,u) with capacity $f(e) + c((v,u))$



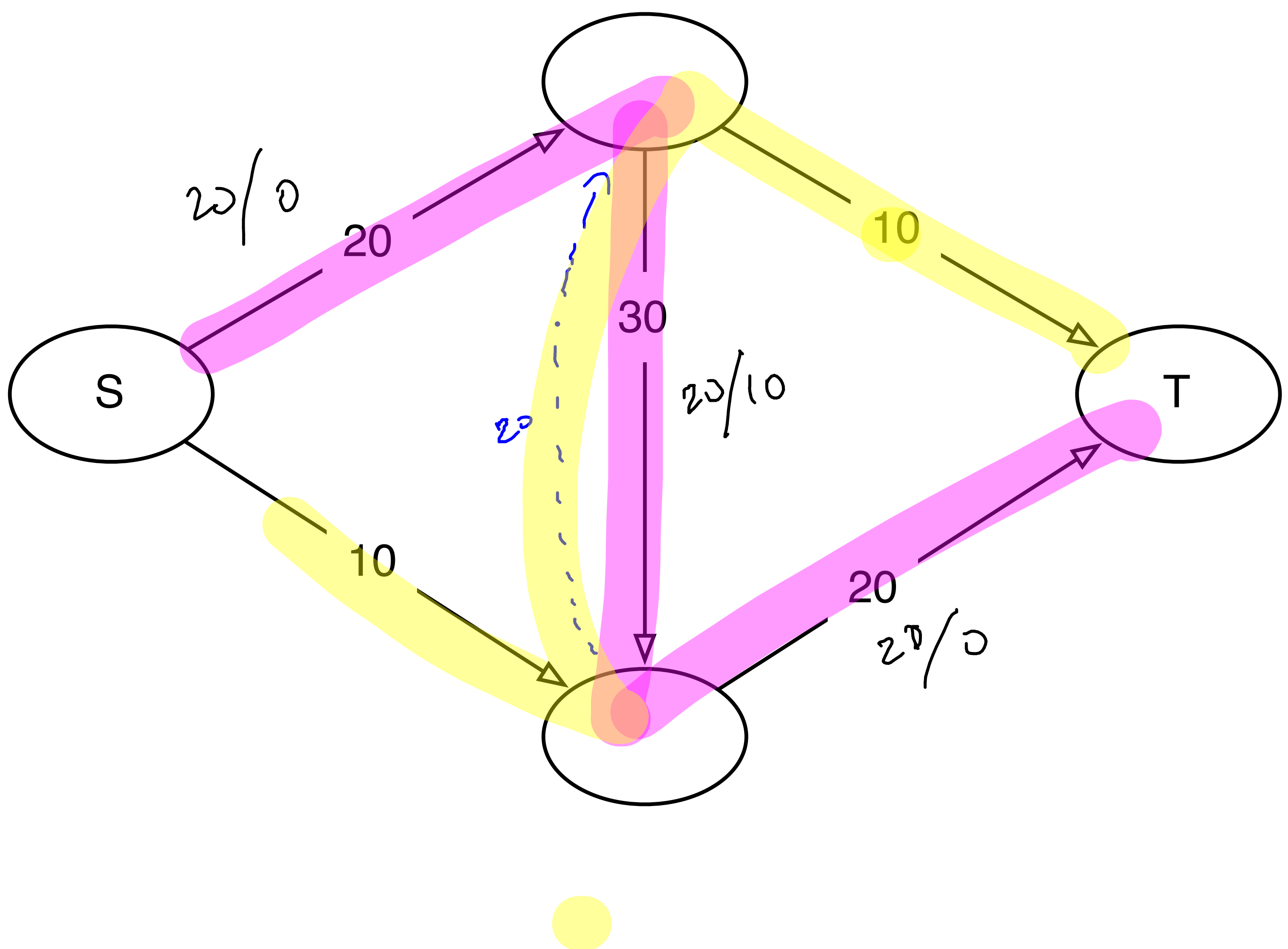
(* special cases)

example residual graph



why residual graphs ?

2 0 units



augmenting paths

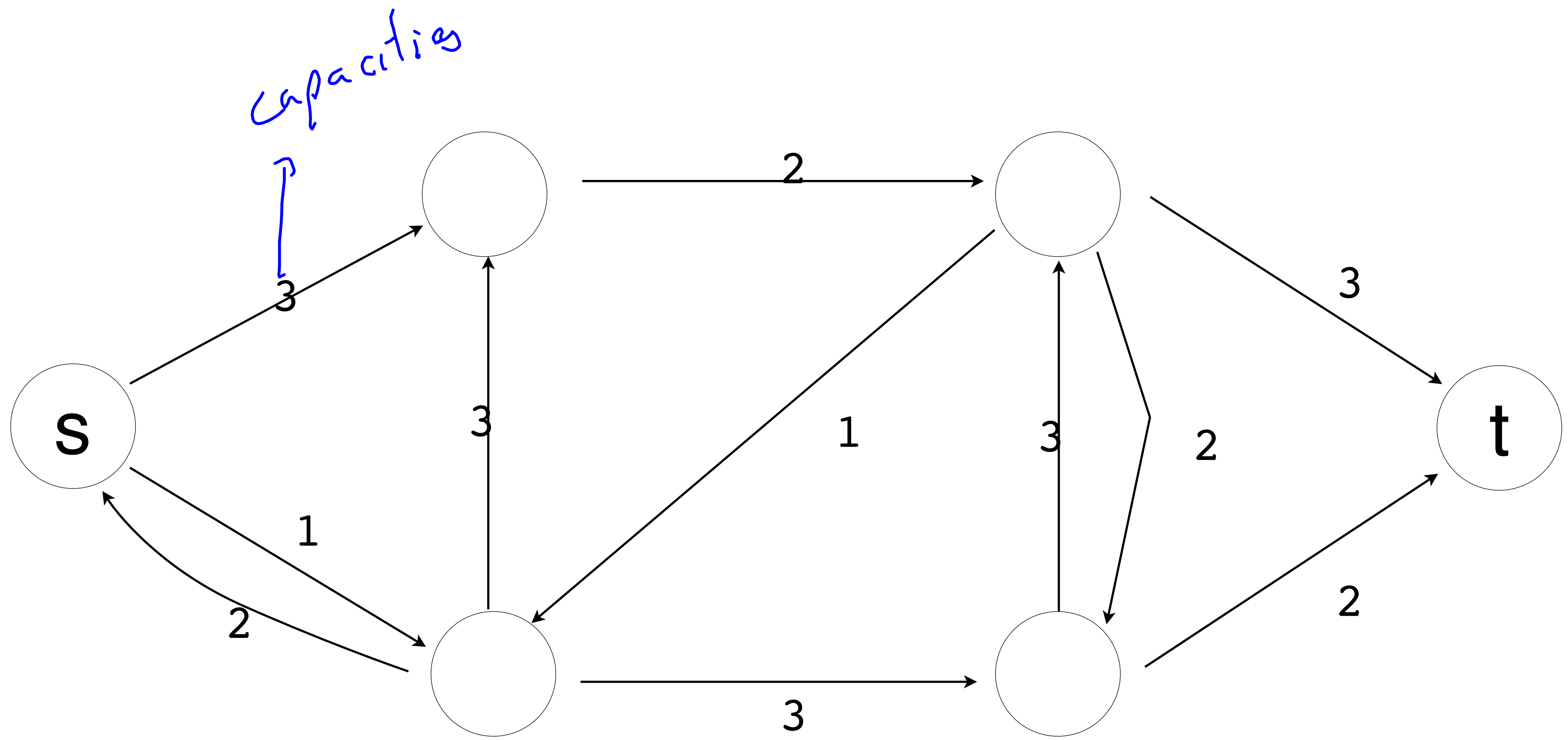
Def: A path from s to t in G_f.

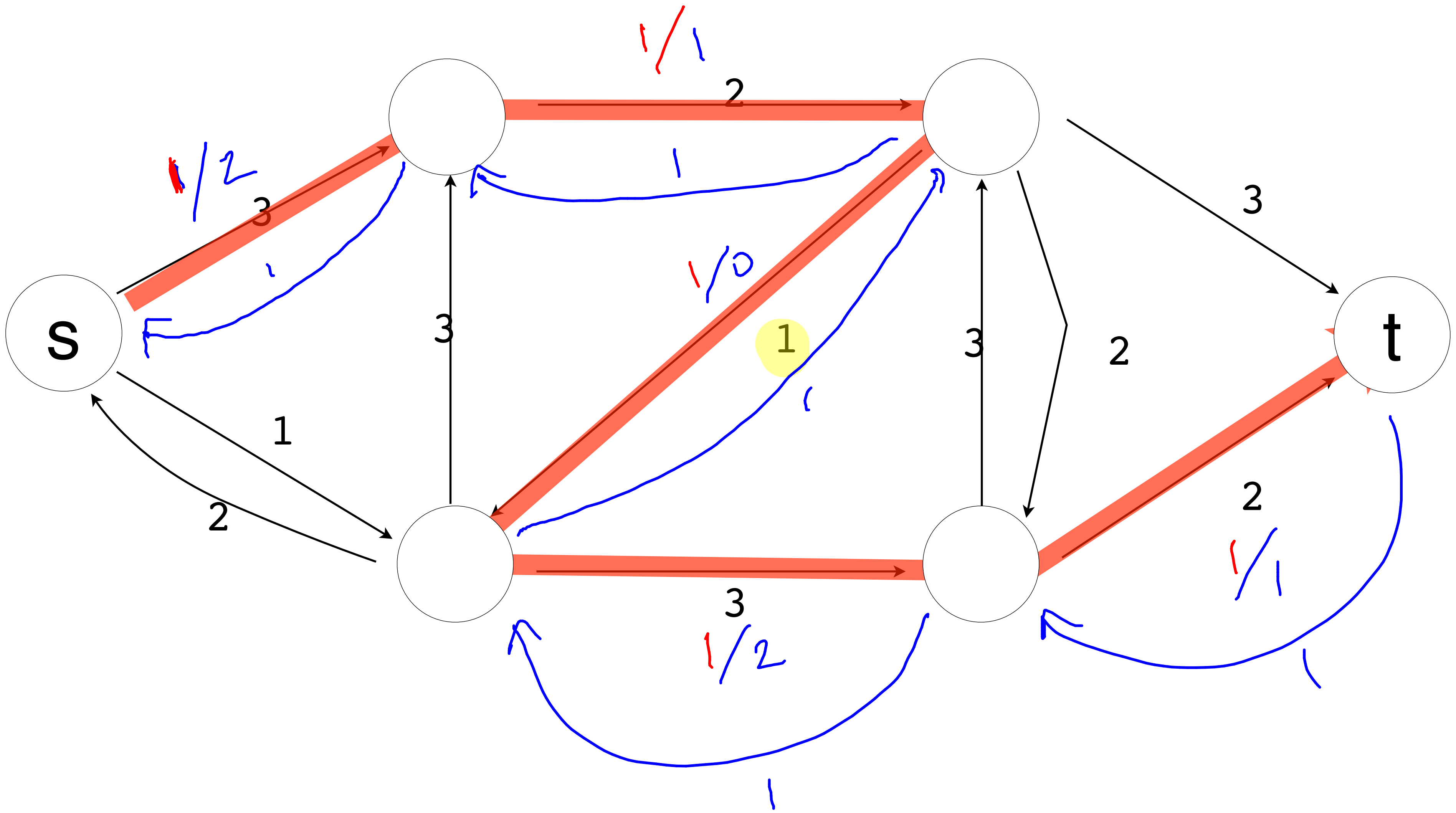
Ford-Fulkerson

initialize $f(u,v) \leftarrow 0 \; \forall u,v$

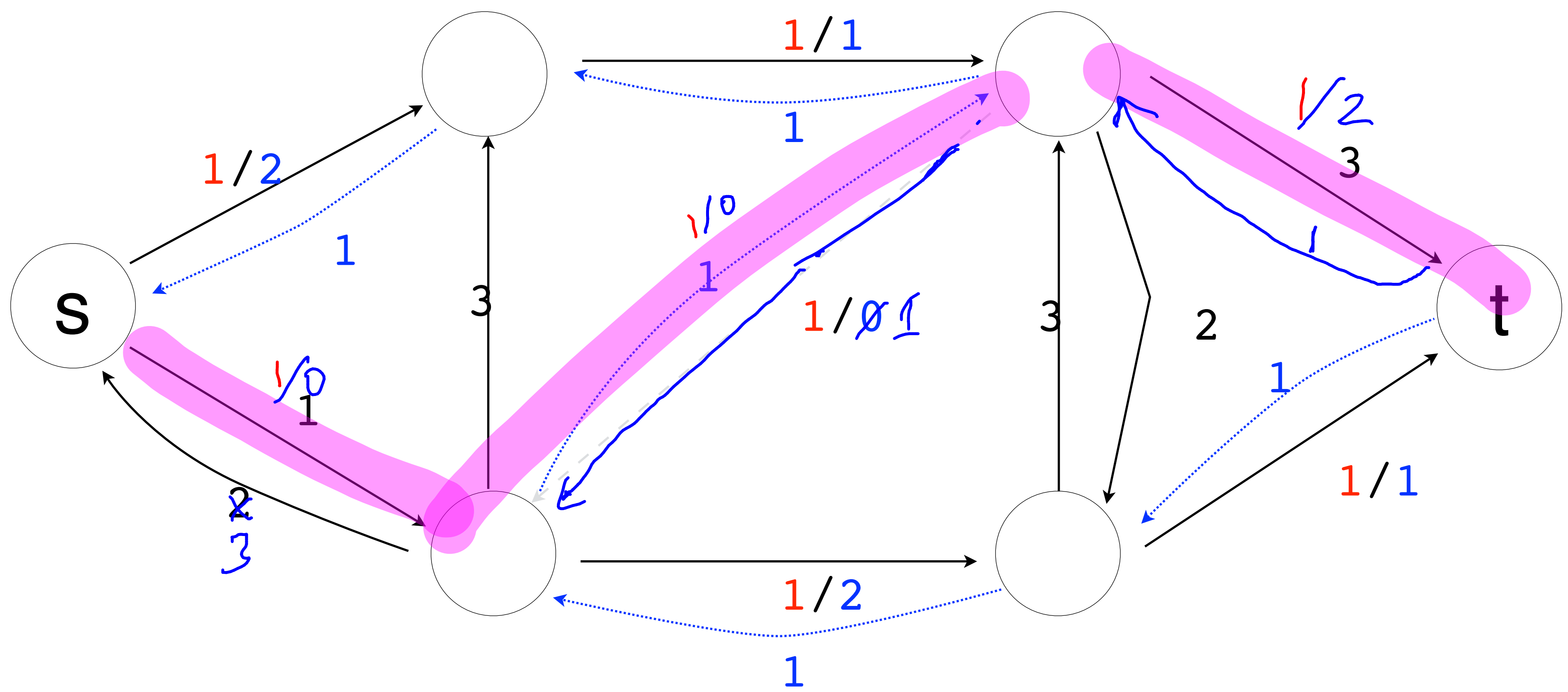
while exists an augmenting path p in G_f

augment f with $c_f(p) = \min_{(u,v)\in p} c_f(u,v)$

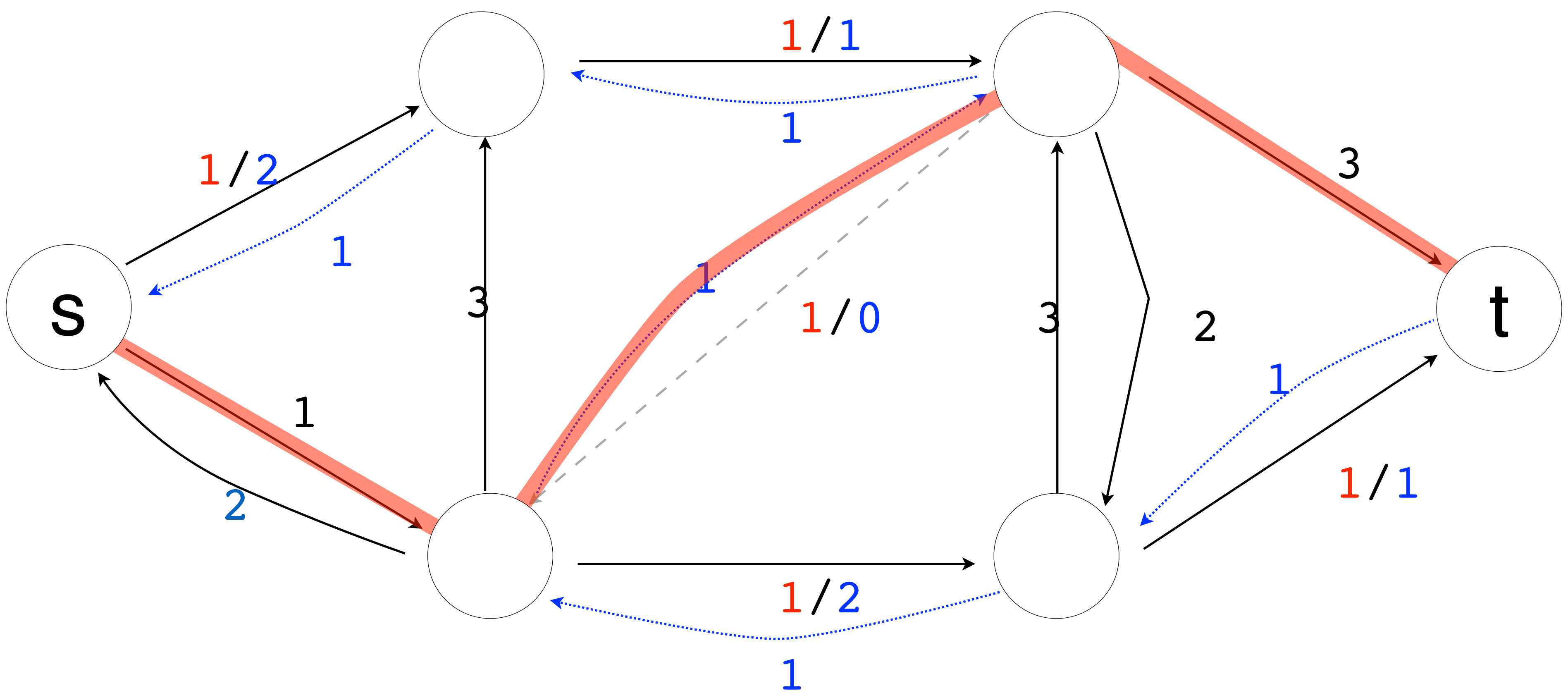

capacities
s
t
3
2
3
1
3
3
2
2
1
2
3
2


s
t
3
2
3
1
2
3
3
2
2
3
1
1/2
1/1
1/0
1/2
1/1
1
1
1
1
1

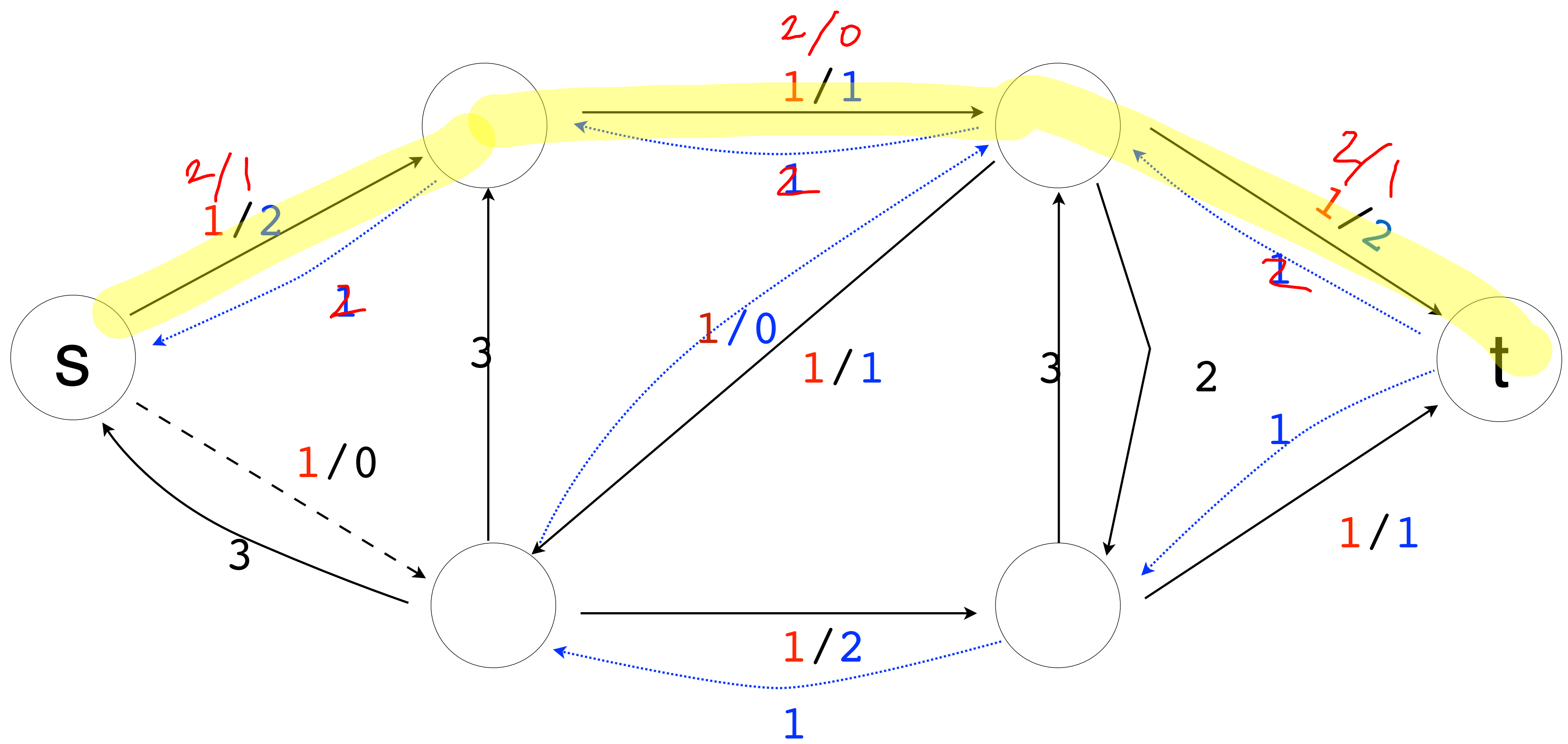
current flow/remaining capacity



current flow/remaining capacity



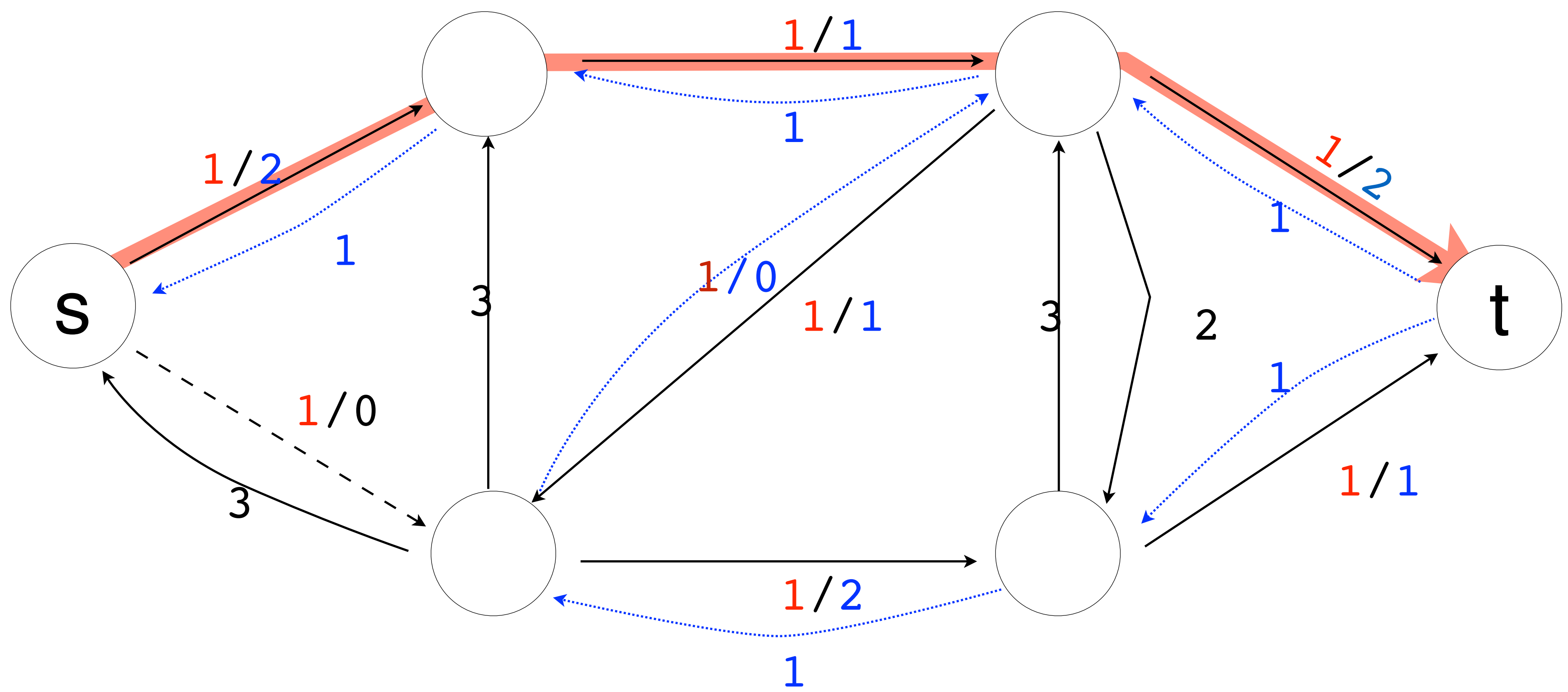
current flow/remaining capacity



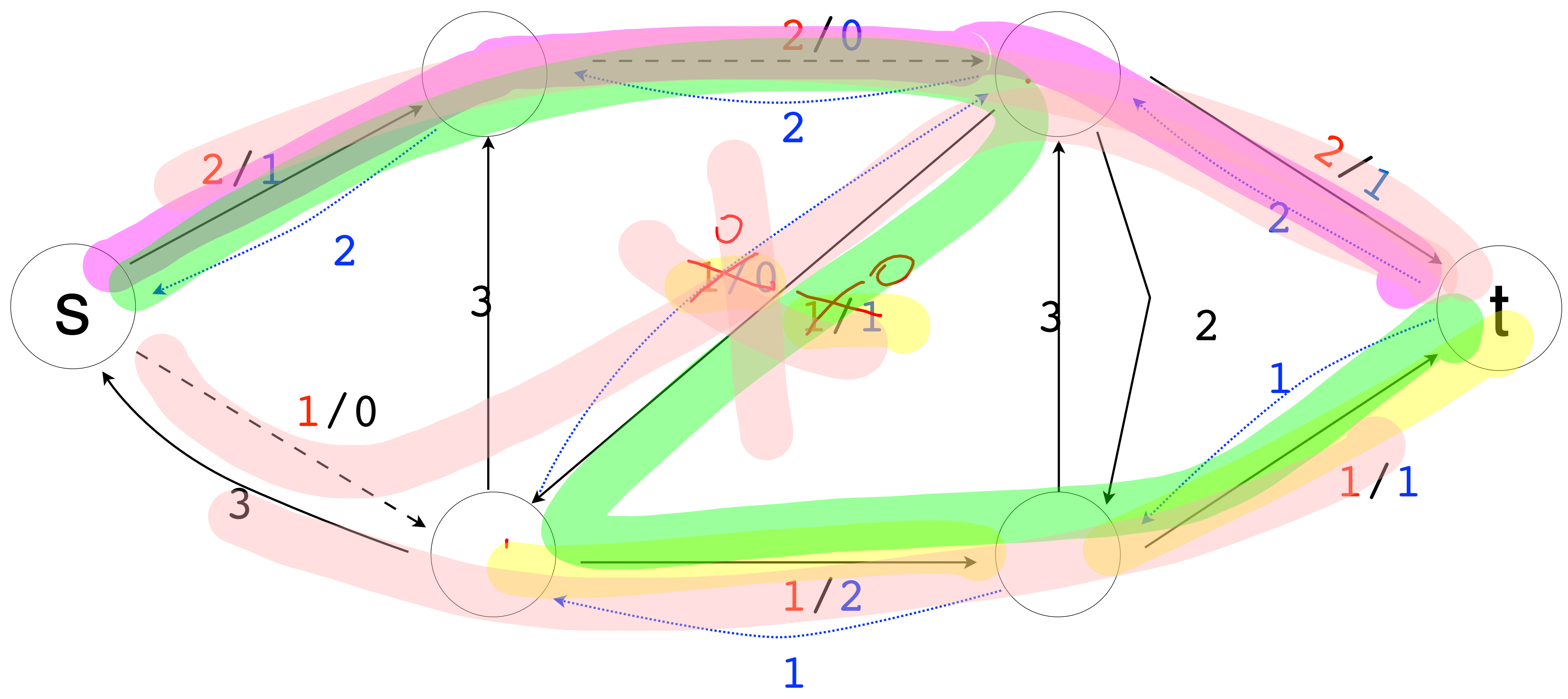
D – E log V

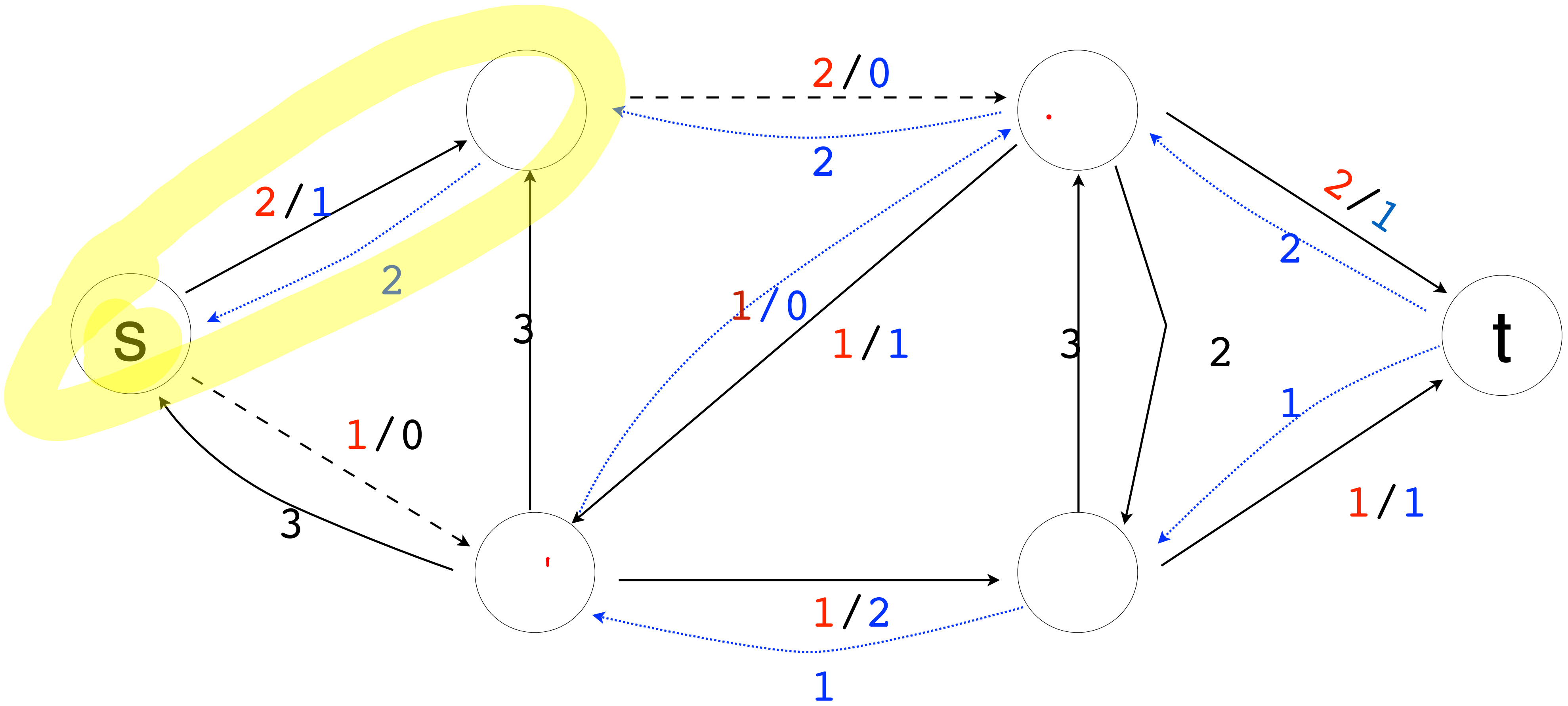
BFS → (E+V)

current flow/remaining capacity

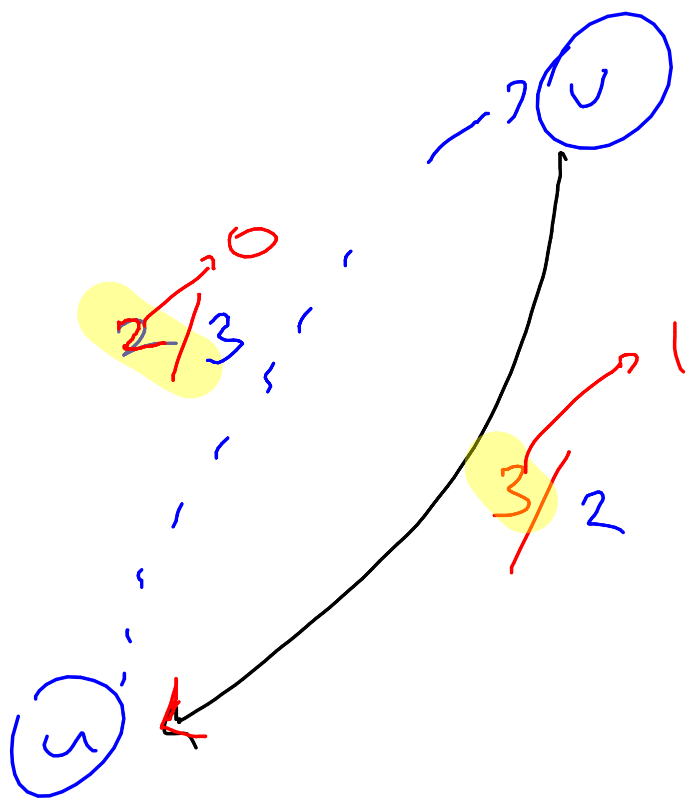


Gf





s
t
2 / 0
2
2 / 1
2
1 / 0
1 / 1
3
3
2
2 / 1
2
1
1 / 0
3
1 / 2
1
1 / 1



v
u
2/3
0
3/2
1

Ford-Fulkerson

initialize $f(u,v) \leftarrow 0 \ \forall u,v$
while exists an augmenting path *p* in G_f
    augment *f* with $c_f(p) = \min_{(u,v)\in p} c_f(u,v)$

$O(E|f|)$

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

number of iterations of while loop: $|f|$

Why does the algorithm work correctly??

Cuts

Def of a cut: Partition of V into S, T s.t.

$s \in S \quad t \in T$

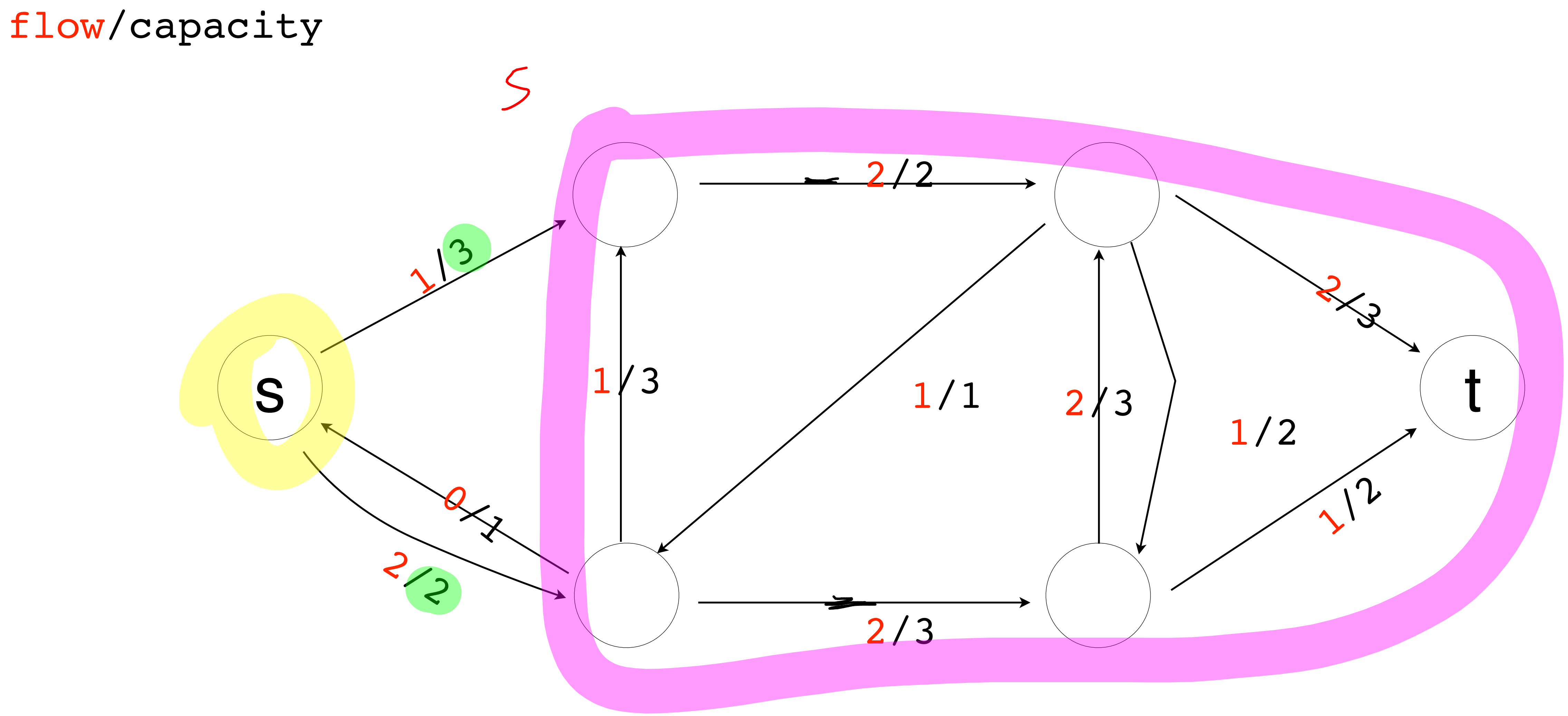
cost of a cut:

$$||S,T|| = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

lemma: [MinCut] for any $f, (S,T)$

$$|f| \leq \|S,T\|$$

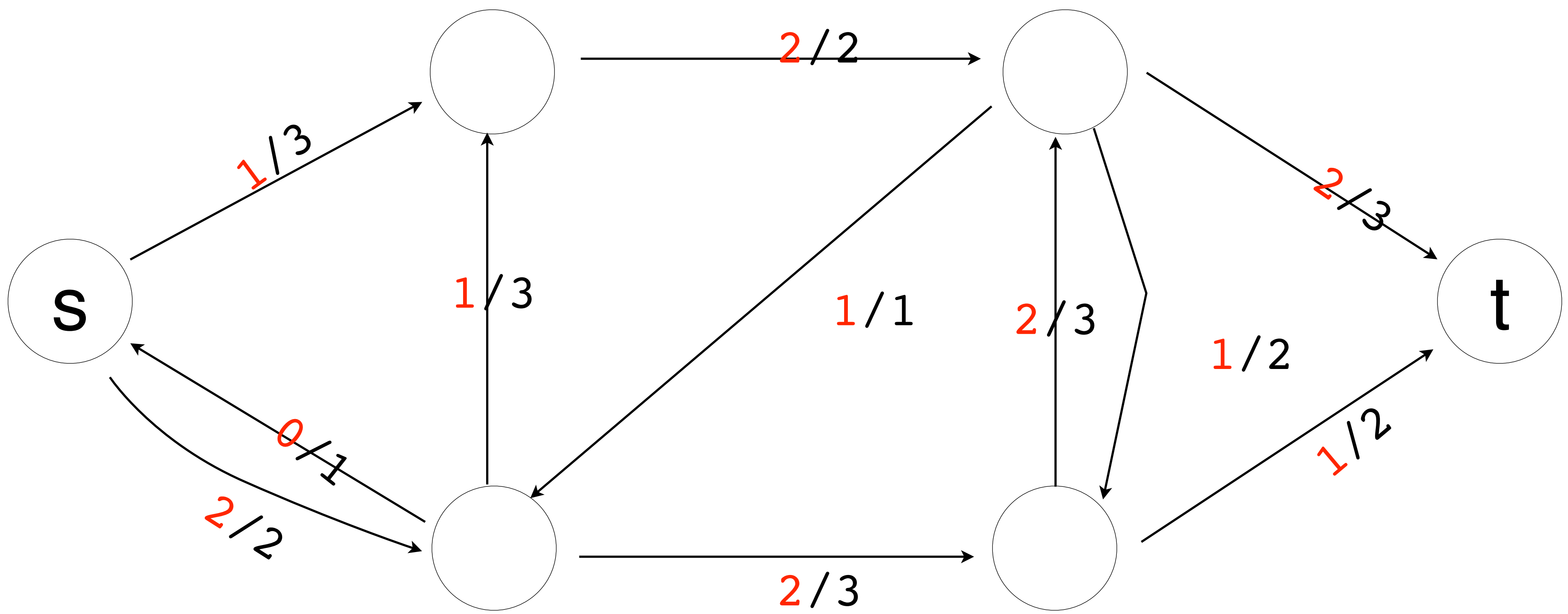
for any $f, (S, T)$ it holds that $|f| \leq ||S, T||$



$||S, T|| = 5$

for any $f, (S,T)$ it holds that $|f| \leq ||S,T||$

flow/capacity



Main point

for any $f, (S,T)$ it holds that $|f| \le ||S,T||$

Proof: Consider any flow f.

$\rightarrow \quad |f| = \sum_{v \in V} f(s,v) - \sum_{w \in V} f(w,s)$

Step 1: Add 0.

for all the nodes $u \in S - \{s\}$, we have

$$\sum_{v \in V} f(u,v) - \sum_{w \in V} f(w,u) = 0$$

by conservation of flow.

$\Rightarrow$

$$|f| = \sum_{u \in S} \left[\sum_{v \in V} f(u,v) - \sum_{w \in V} f(w,u) \right]$$

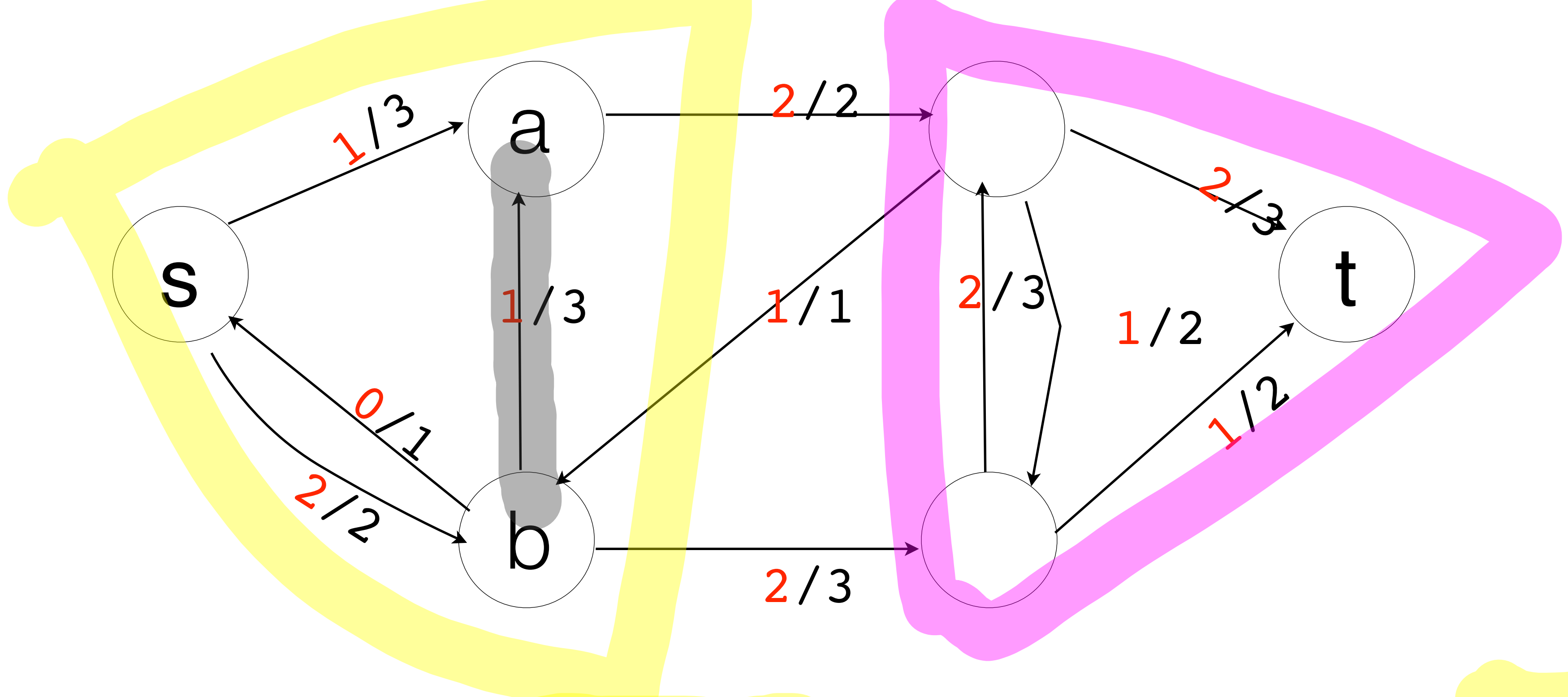
Step 2: Split it up.

$$= \sum_{u \in S} \left[\sum_{v \in T} f(u,v) + \sum_{v \in S} f(u,v) - \sum_{w \in S} f(w,u) - \sum_{w \in T} f(w,u) \right]$$

A property to remember

For any $f, (S,T)$ it holds that $|f| \leq ||S,T||$

proof:



$$\sum_{u\in S}\left[\sum_{v\in T} f(u,v) + \sum_{v\in S} \underset{\text{out}}{f(u,v)} - \sum_{w\in T} f(w,u) - \sum_{w\in S} \underbrace{f(w,u)}_{\text{in}}\right]$$

$u=b$ $\qquad$ $+ \quad f(b,a)$

$u=a$ $\qquad$ $- \quad f(b,a)$

Consider an edge (b,a) whose endpoints are entirely in S.

Edges in S contribute 0 to |f|.

for any $f, (S,T)$ it holds that $|f| \le ||S,T||$

(finishing proof)

$$\sum_{u\in S}\left[\sum_{v\in T} f(u,v) + \cancel{\sum_{v\in S} f(u,v)} - \sum_{w\in T} f(w,u) - \cancel{\sum_{w\in S} f(w,u)}\right]$$

$$= \sum_{u\in S}\left[\sum_{v\in T} f(u,v) - \sum_{w\in T} f(w,u)\right]$$

$$\le \sum_{u\in S}\sum_{v\in T} f(u,v) \le \sum_{u\in S}\sum_{v\in T} c(u,v) = ||S,T||$$

Thm: max flow = min cut

$$\max_f |f| = \min_{S,T} ||S,T||$$

If f is a max flow, then G_f has no augmenting paths.

Define the set $S = \{ u \mid \exists$ a path p in G_f from $s \rightsquigarrow u$ such that $c_f(p) > 0 \}$

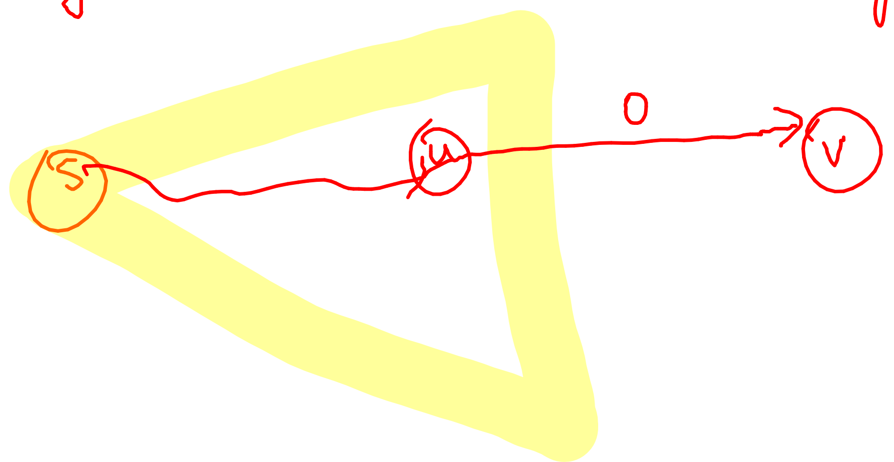
"all nodes that one can still "reach" from s"

① $(S, T = V - S)$ is a cut. why ?? ⓐ $s \in S$ and ⓑ $t \in V - S$ b/c o.w there would still be an augmenting path in G_f.

Thm: max flow = min cut

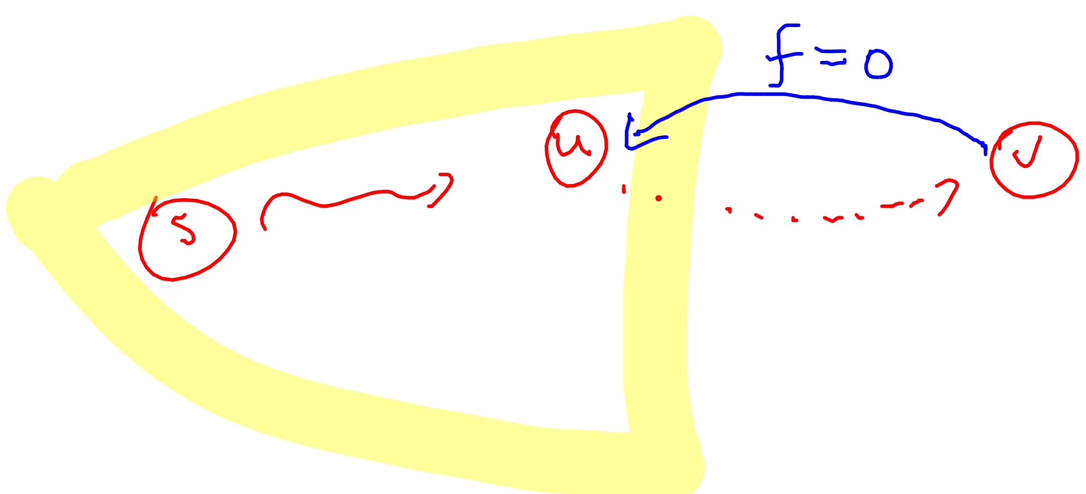
$$\max_f |f| = \min_{S,T} ||S,T||$$ (continued)

① $c_f(u,v) = 0$ for any $u \in S$ and $v \in T$. $\Rightarrow f(u,v) = c(u,v)$



If the capacity on this edge was > 0, then v would be in S!! b/c we could reach it from $\underline{s}$

② $f(v,u) = 0$ for any $u \in S$ and $v \in T$



If $f(v,u) > 0$ then there would be a residual edge from u to v with positive capacity $\Rightarrow$ v would be in S

Why FF works

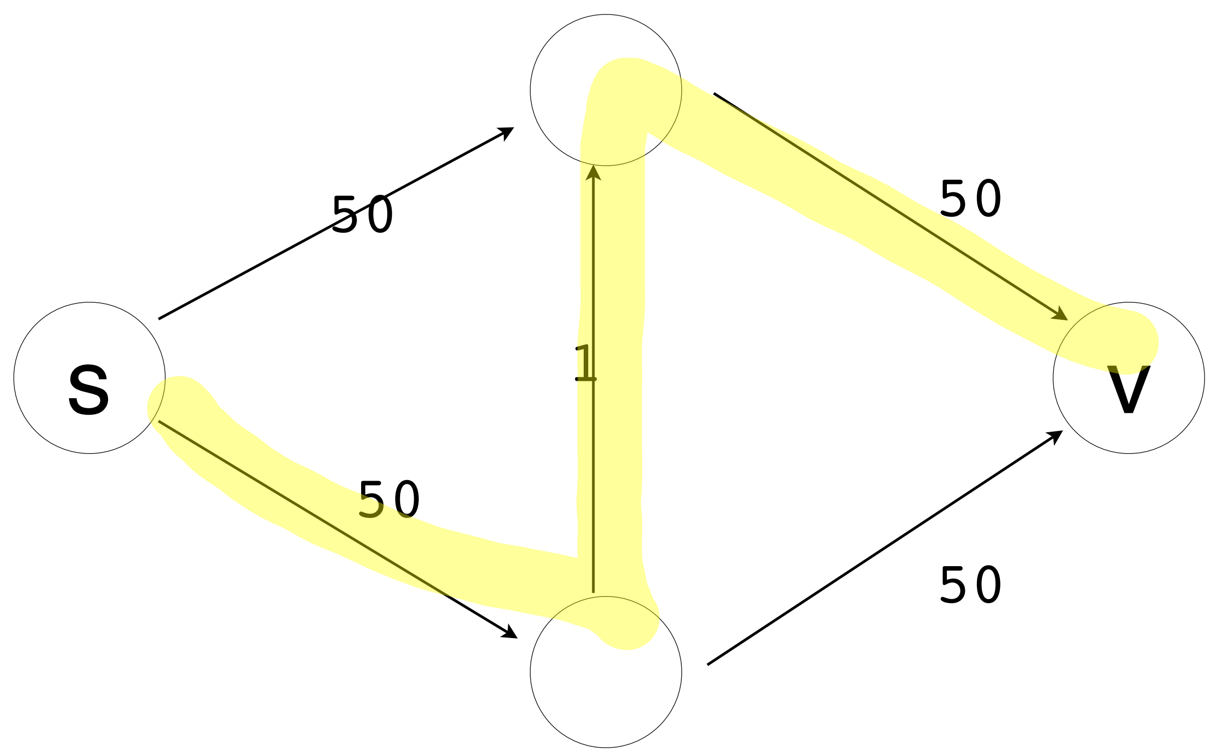
$$|f| = \sum_{u \in S} \left[\sum_{v \in T} f(u,v) - \sum_{w \in T} f(w,u) \right]$$

$\downarrow$ by point ②

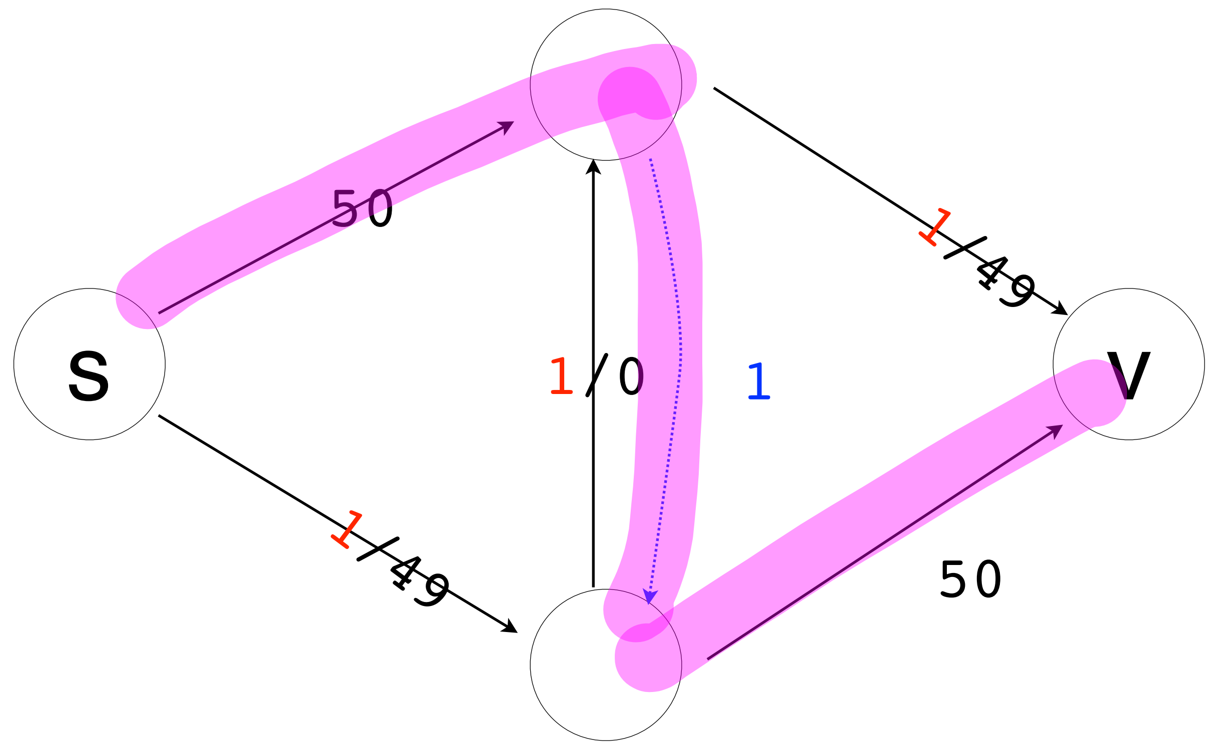
$$= \sum_{u \in S} \left[\sum_{v \in T} f(u,v) - 0 \right]$$

$\downarrow$ by point ①

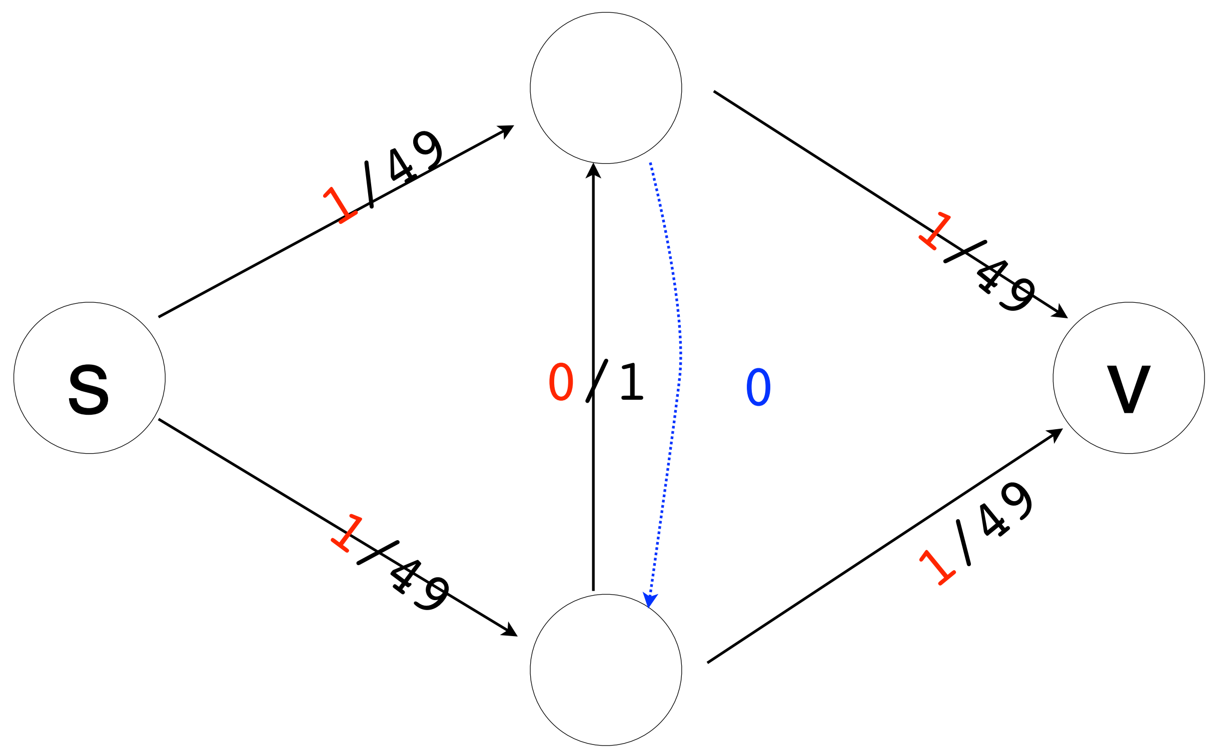
$$= \sum_{u \in S} \sum_{v \in T} c(u,v) = \|S,T\|$$



S
V
50
50
1
50
50

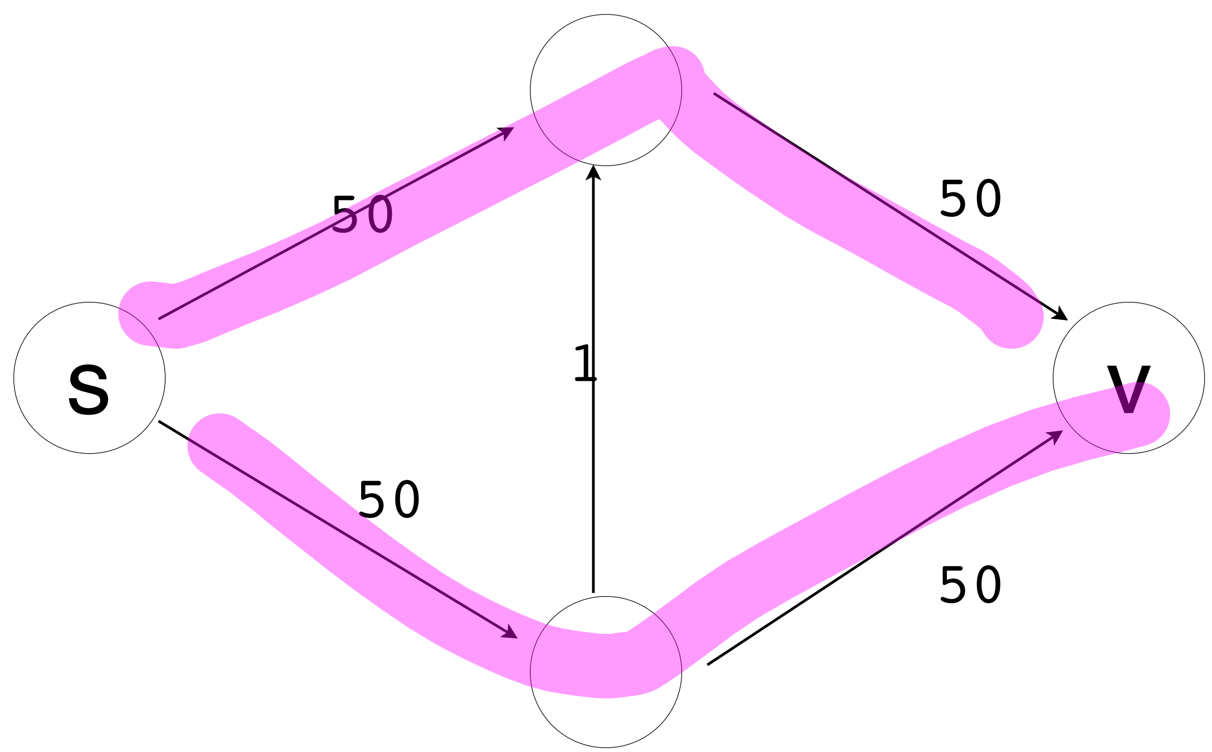


50
1/49
S
1/0
1
V
1/49
50



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

root of the problem

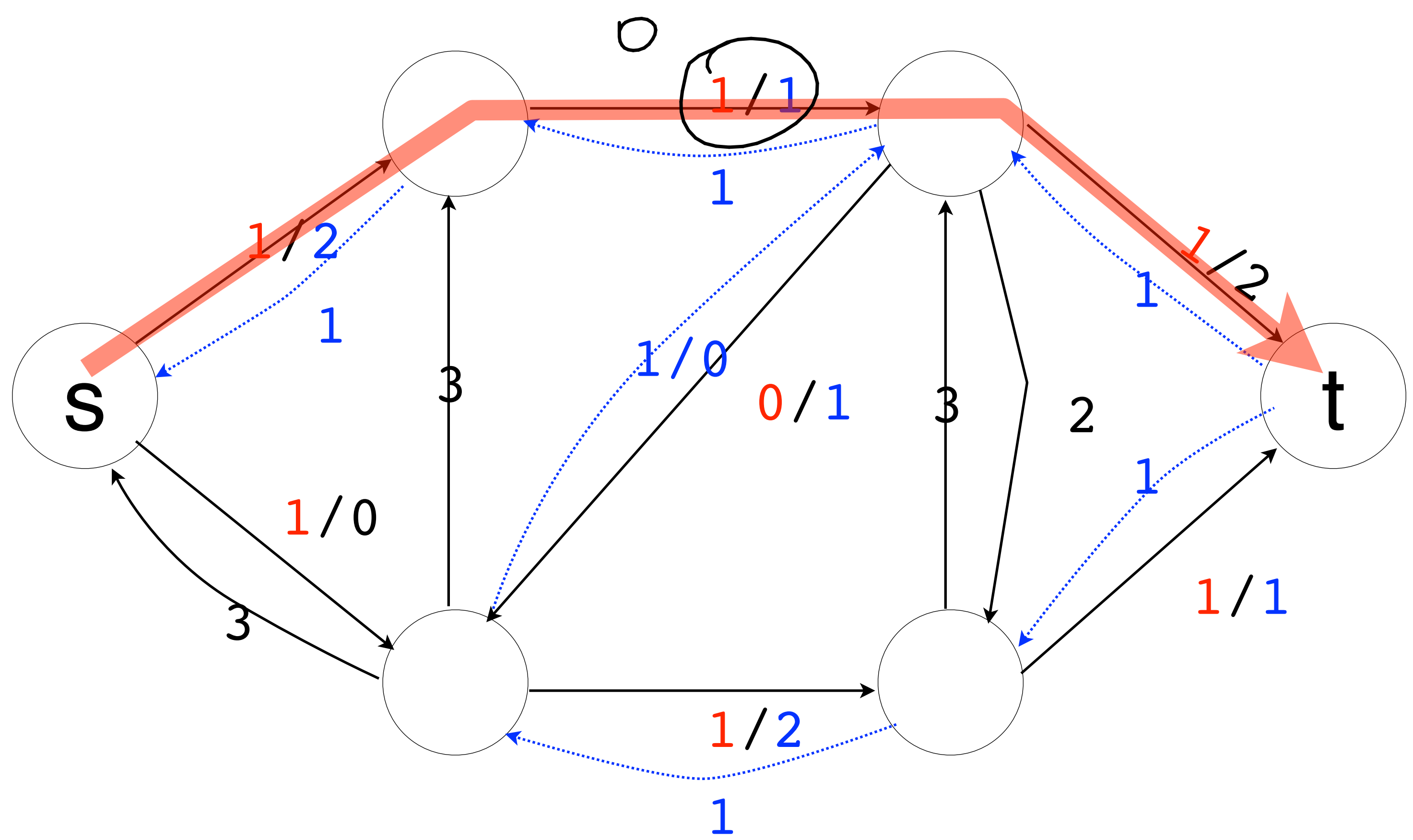


Edmonds-Karp 2

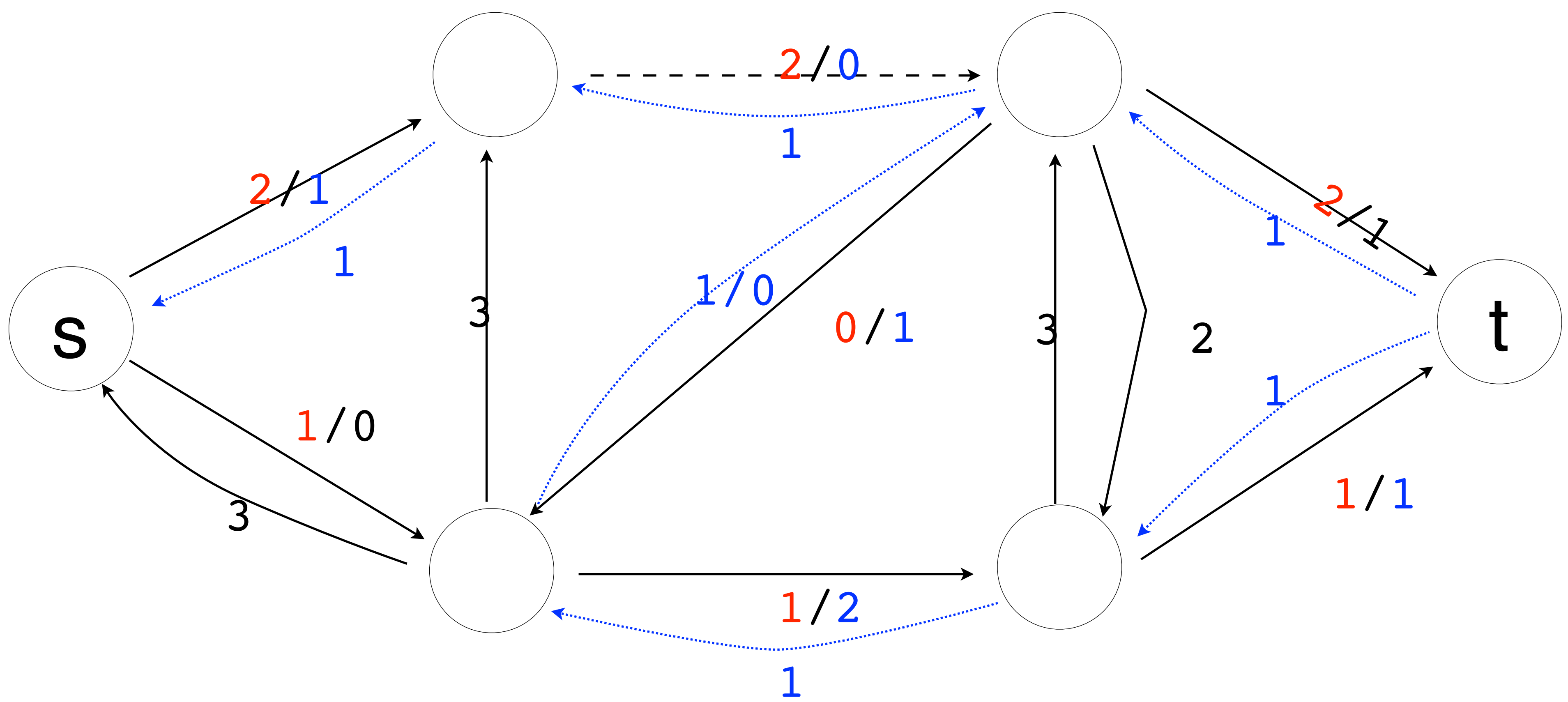
choose path with fewest edges first. (Use BFS)

$\delta_f(s, v)$: smallest number of edges in a path from s to v in the residual graph G_f.

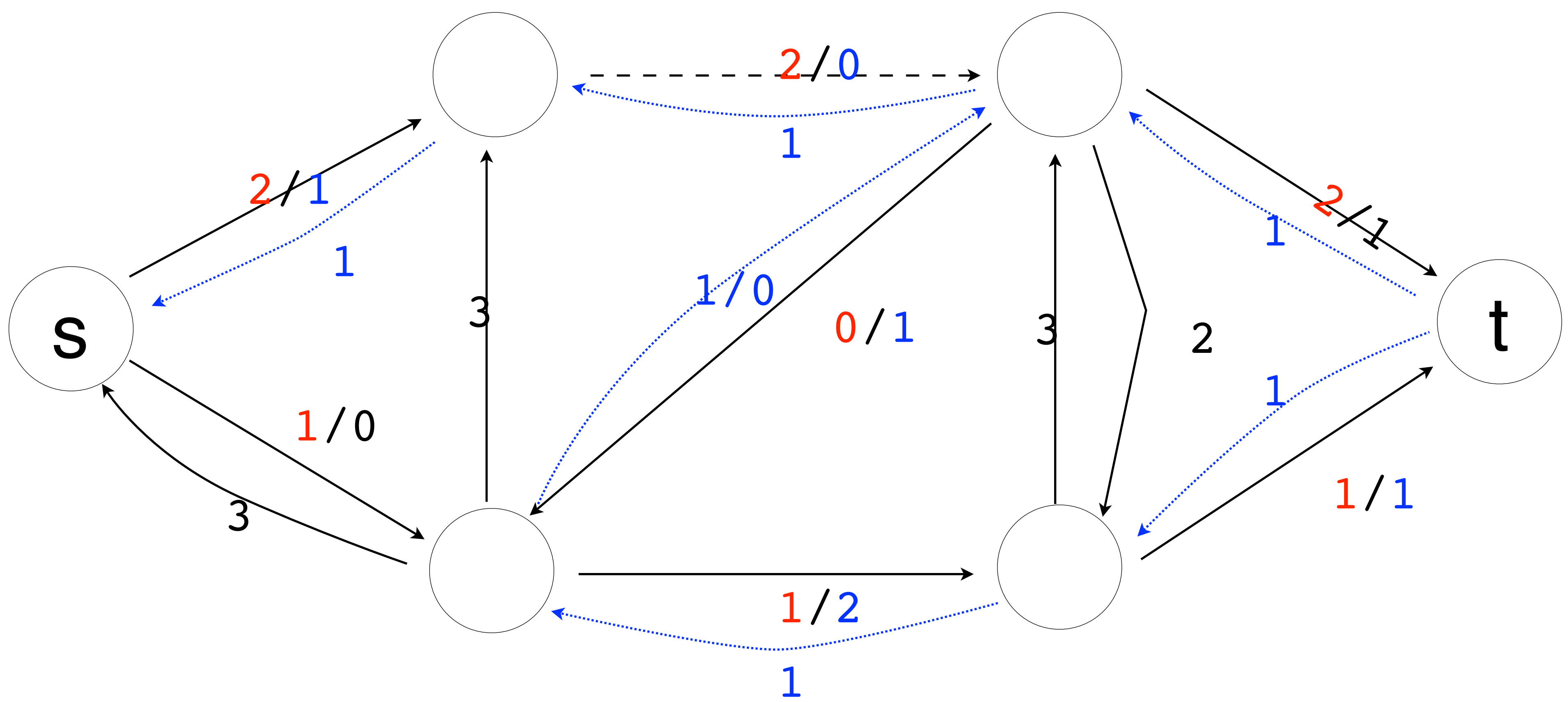
$$\underbrace{\delta_f(s,v) \text{ increases monotonically thru exec}}_{\delta_{i+1}(v) \geq \delta_i(v)}$$



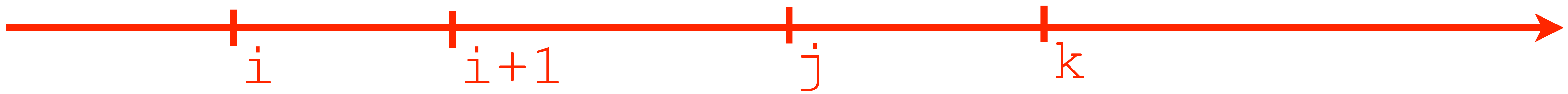
for every augmenting path, some edge is critical.

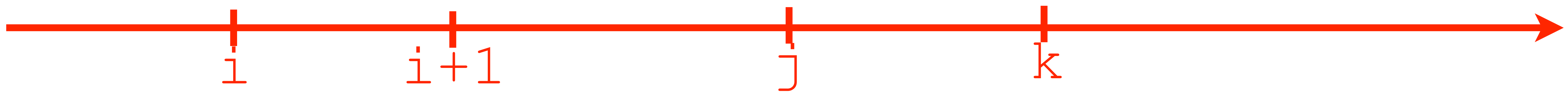


critical edges are removed in next residual graph.

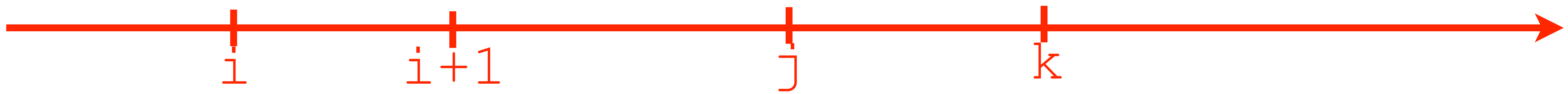


key idea: how many times can an edge be critical? $\frac{V}{2}$

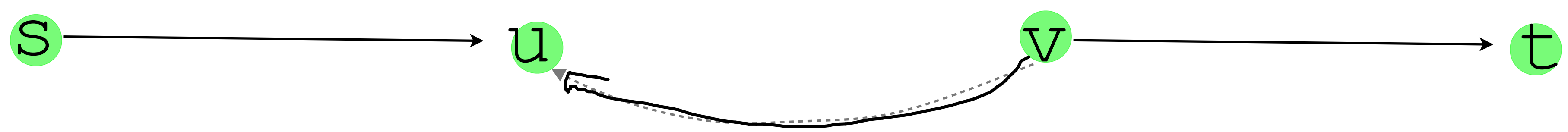

i
i+1
j
k



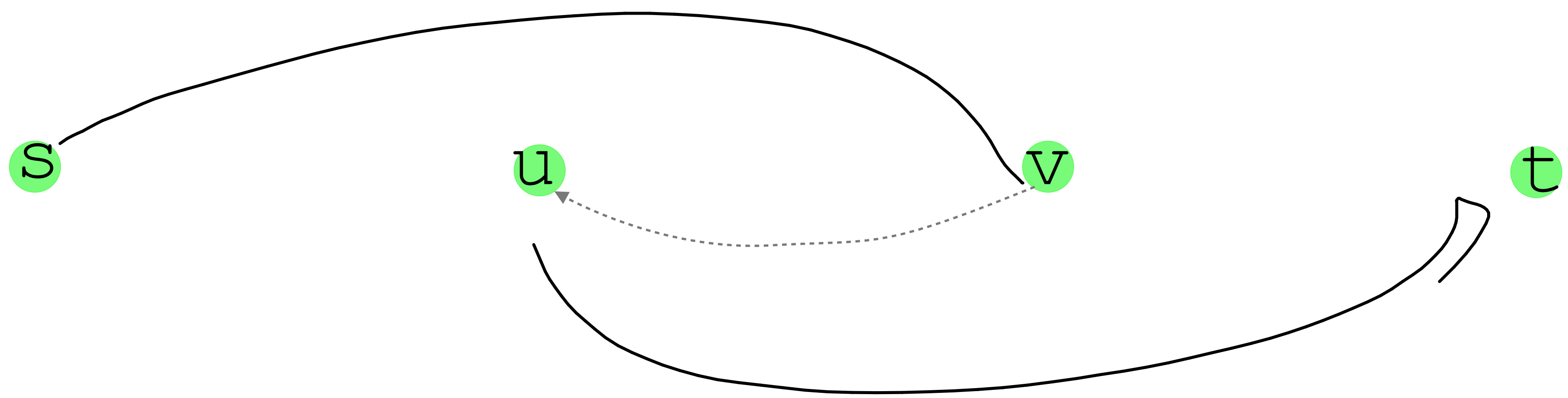
first time (u,v) is critical:

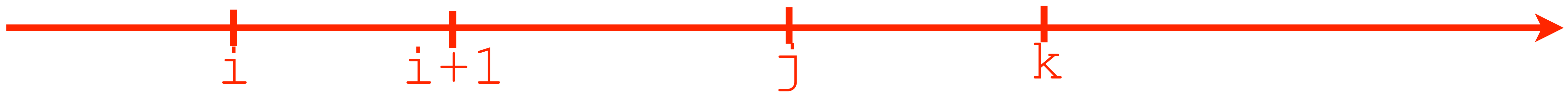


time i+1: (u,v) is critical: $\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$

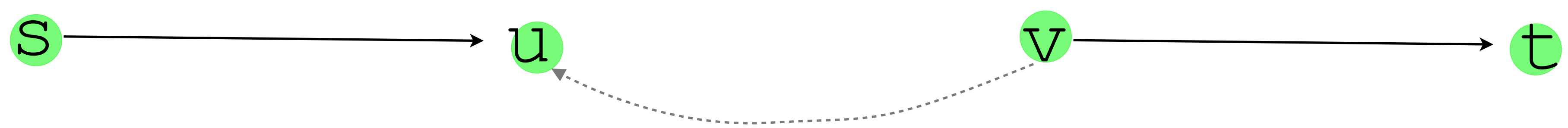


time j: Edge (u,v) STRIKES BACK

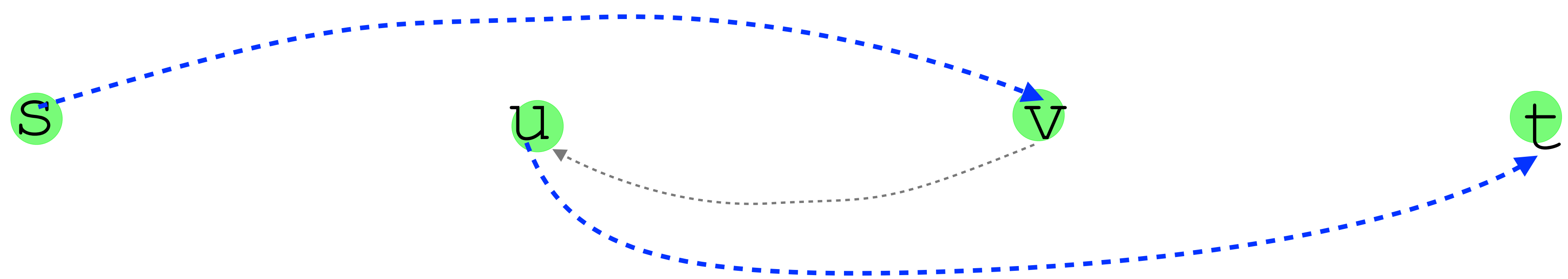




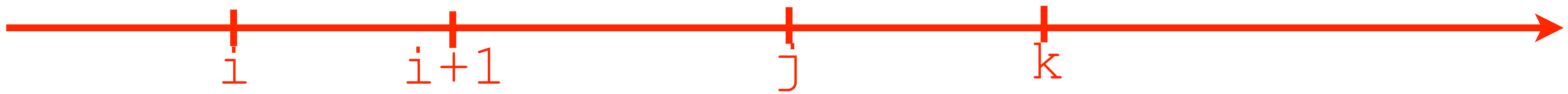
time i+1: (u,v) is critical: $\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$



time j: Edge (u,v) STRIKES BACK



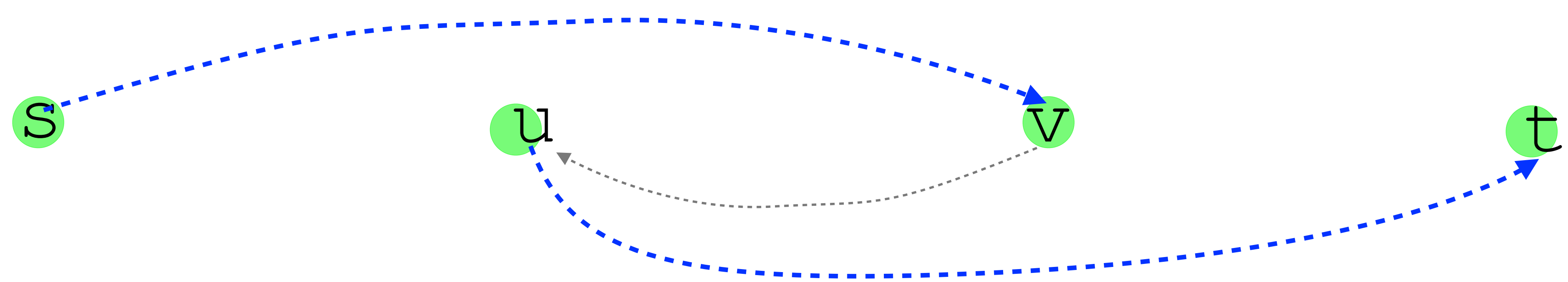
$$\delta_j(s,u) = \delta_j(s,v) + 1$$

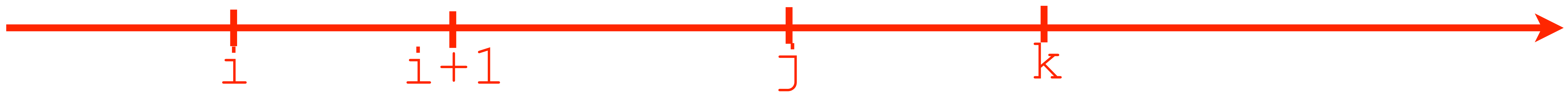


time j: Edge (u,v) STRIKES BACK

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

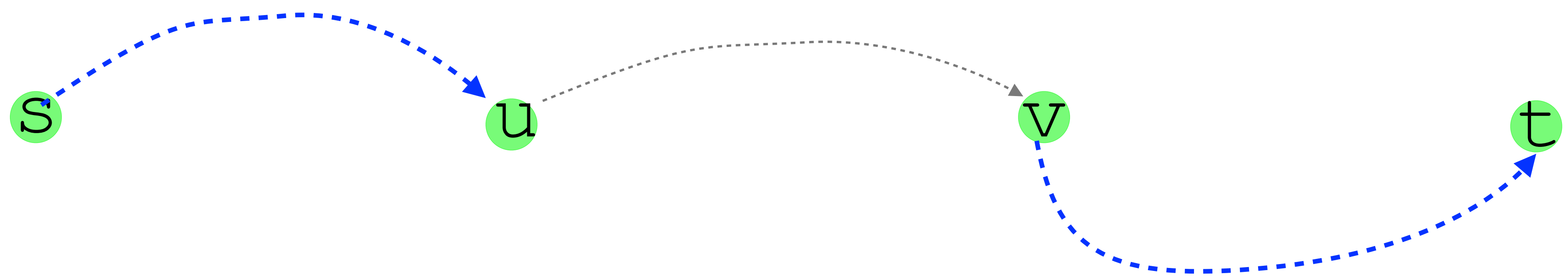
$$\underline{\delta_j(s,u)} = \delta_j(s,v) + 1$$





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?

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

there are only E edges.

ergo, total # of augmenting paths: $E\frac{V}{2}$

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

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

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

EK2 $O(E^2V)$

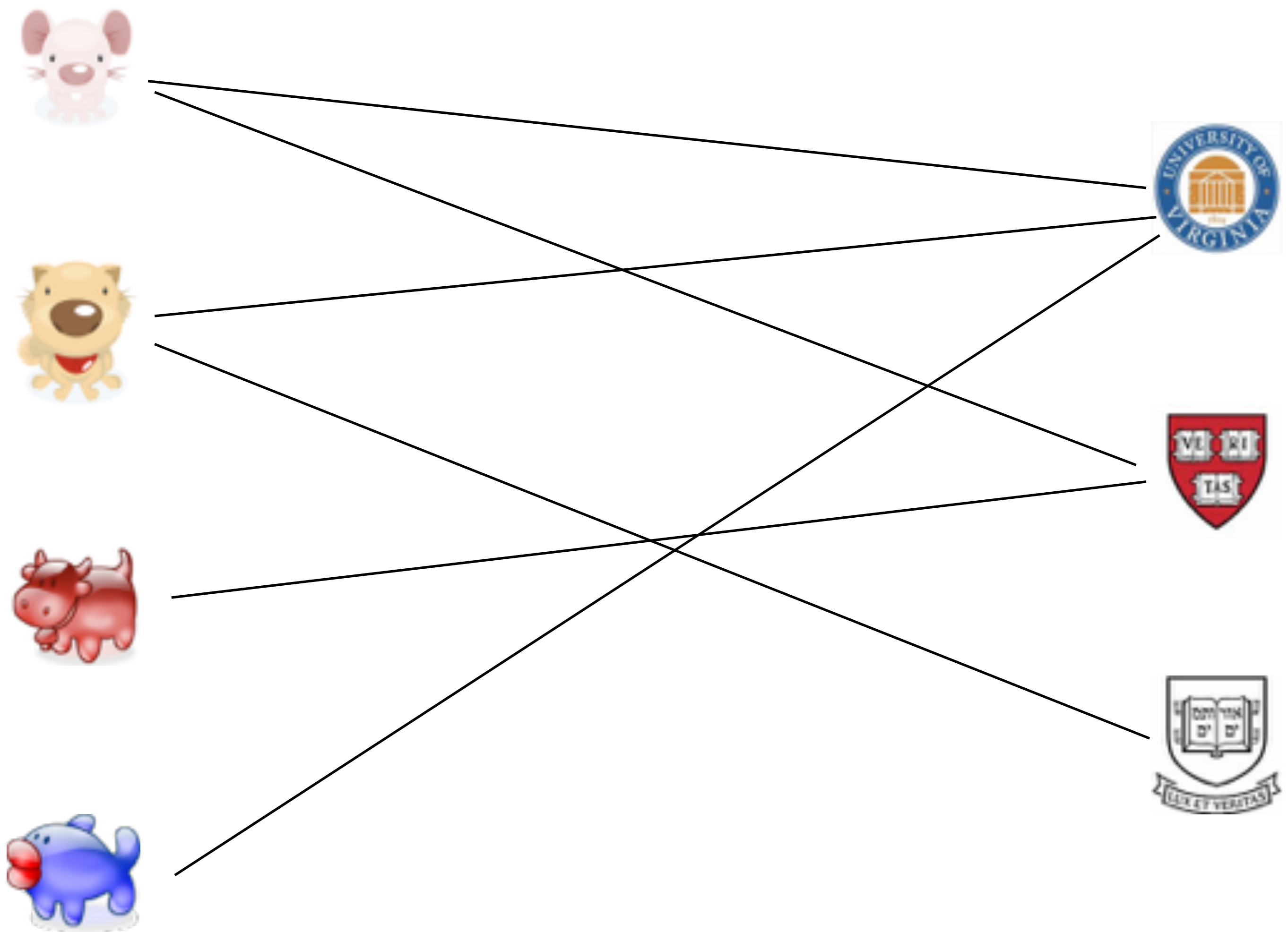
Tarjan

PUSH-RELABEL $\rightarrow$ $O(EV^2)$

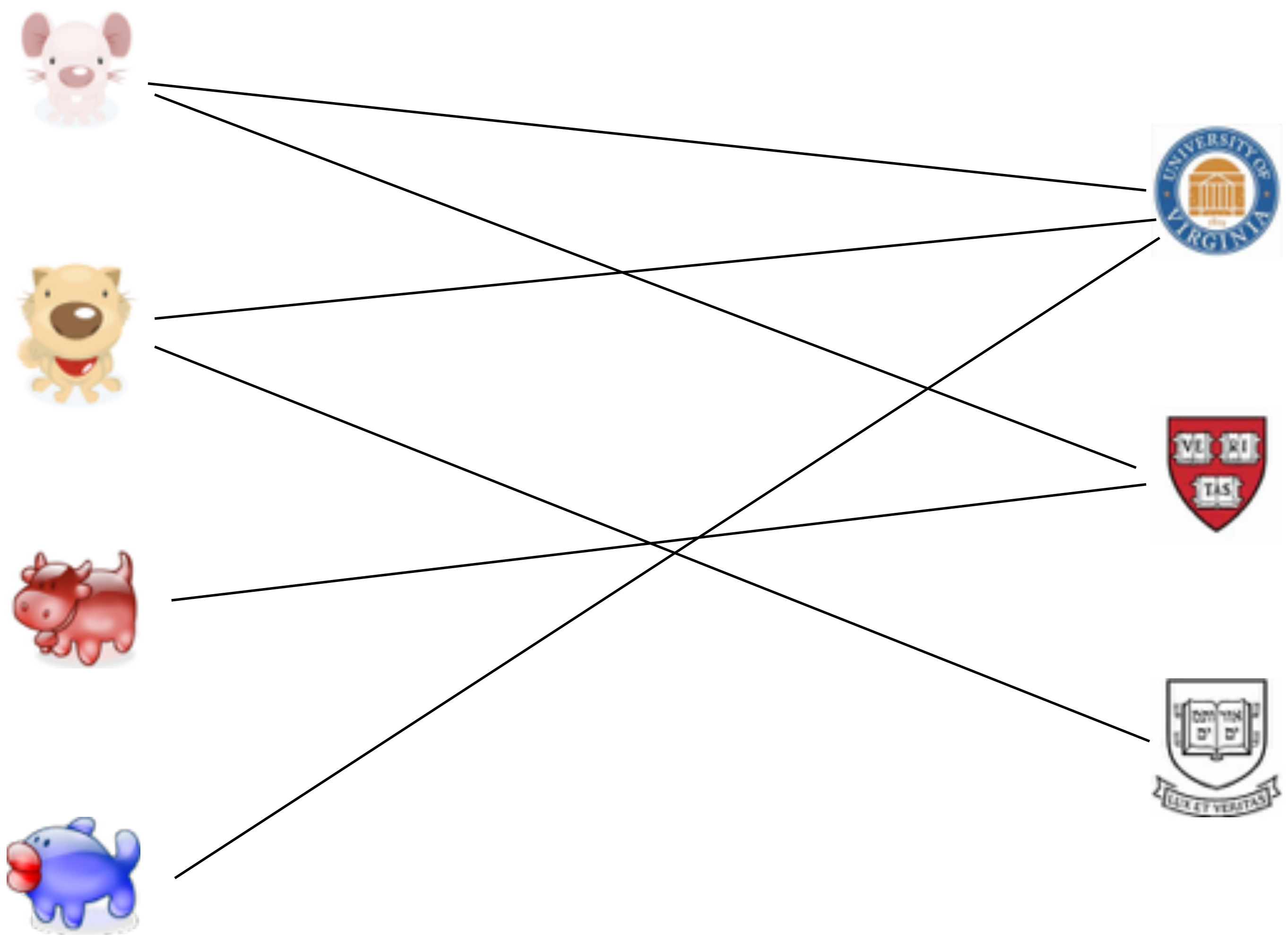
FASTER PUSH-RELABEL $O(V^3)$

Bipartite

maximum bipartite matching



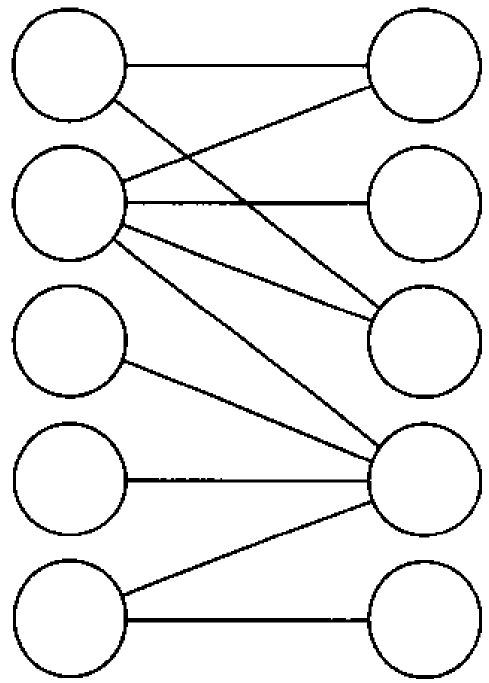
maximum bipartite matching



bipartite matching

problem:

algorithm

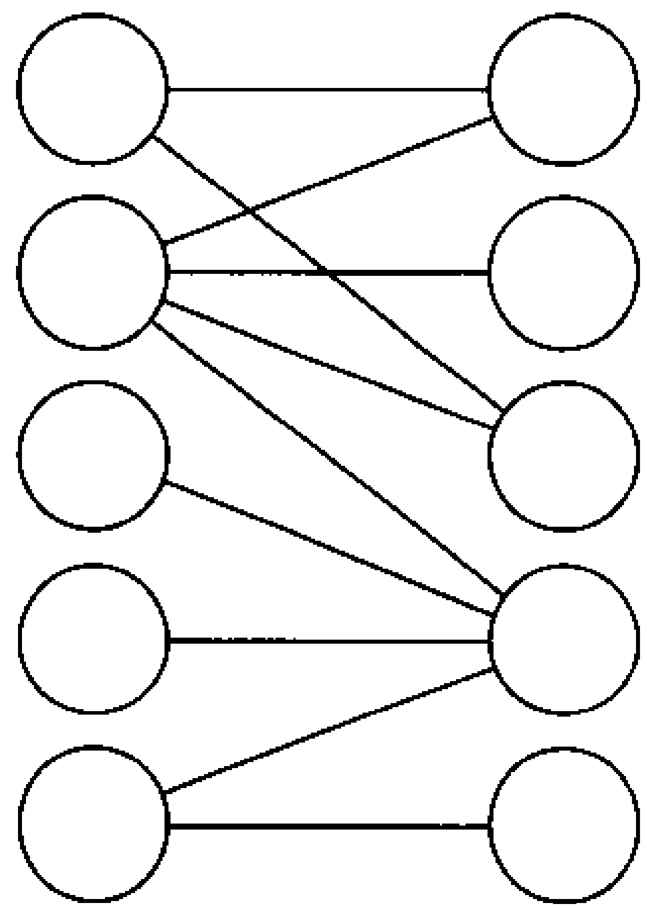


algorithm

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

correctness

IF G' HAS A FLOW OF K, THEN

integrality theorem

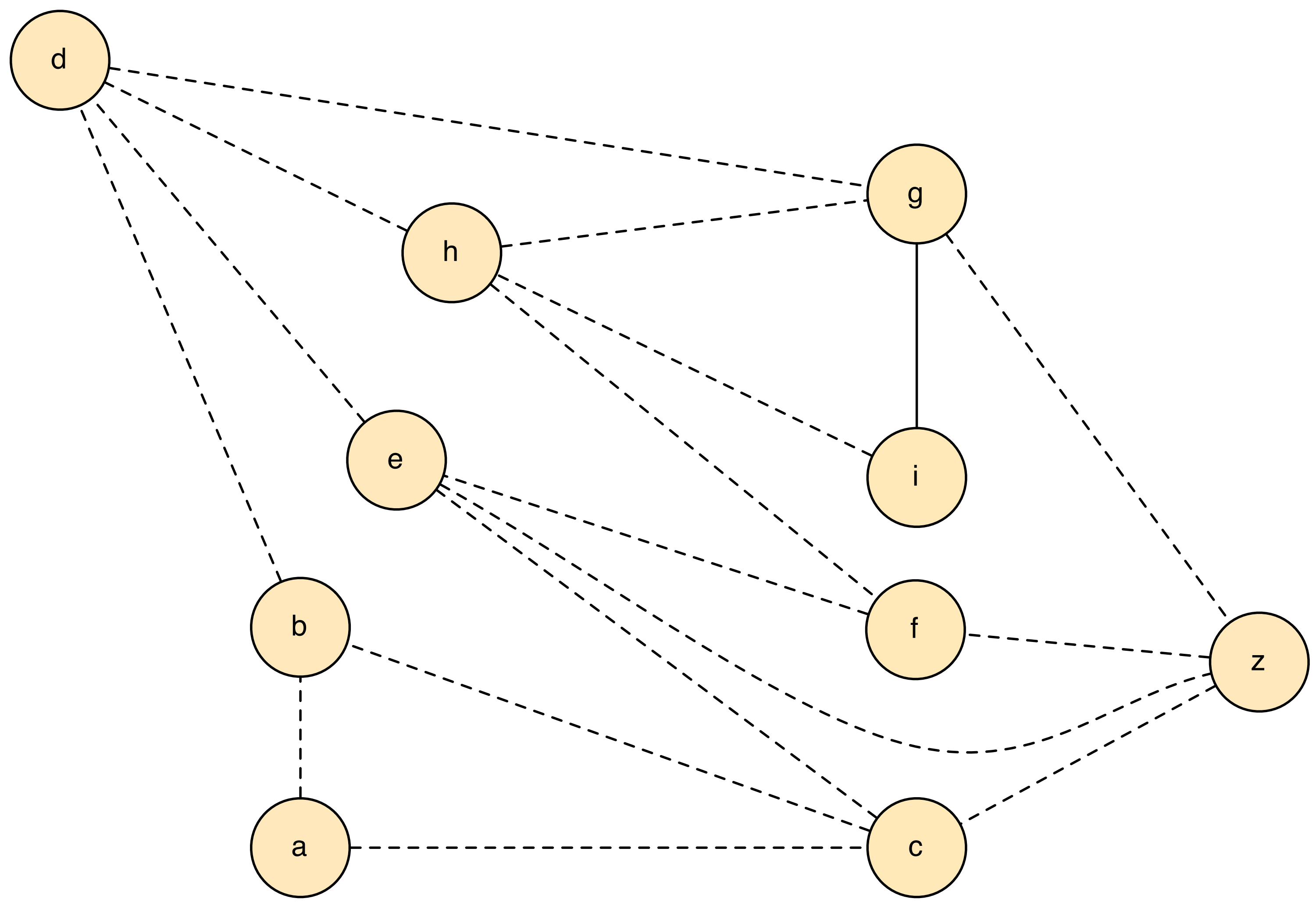
IF CAPACITIES ARE ALL INTEGRAL, THEN

correctness

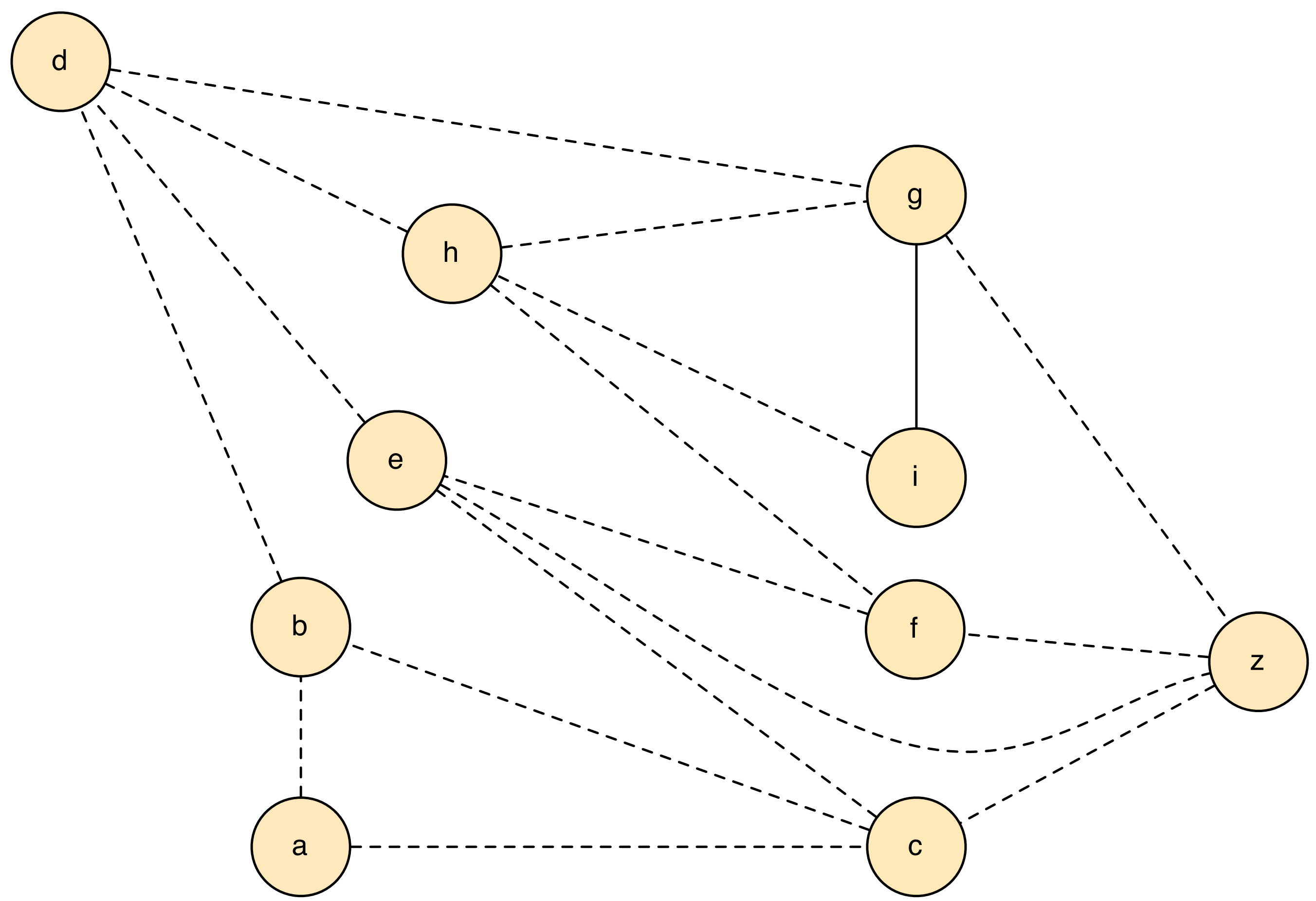
HAS A FLOW OF K, THEN G HAS K-MATCHING.

running time

edge-disjoint paths



algorithm



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.

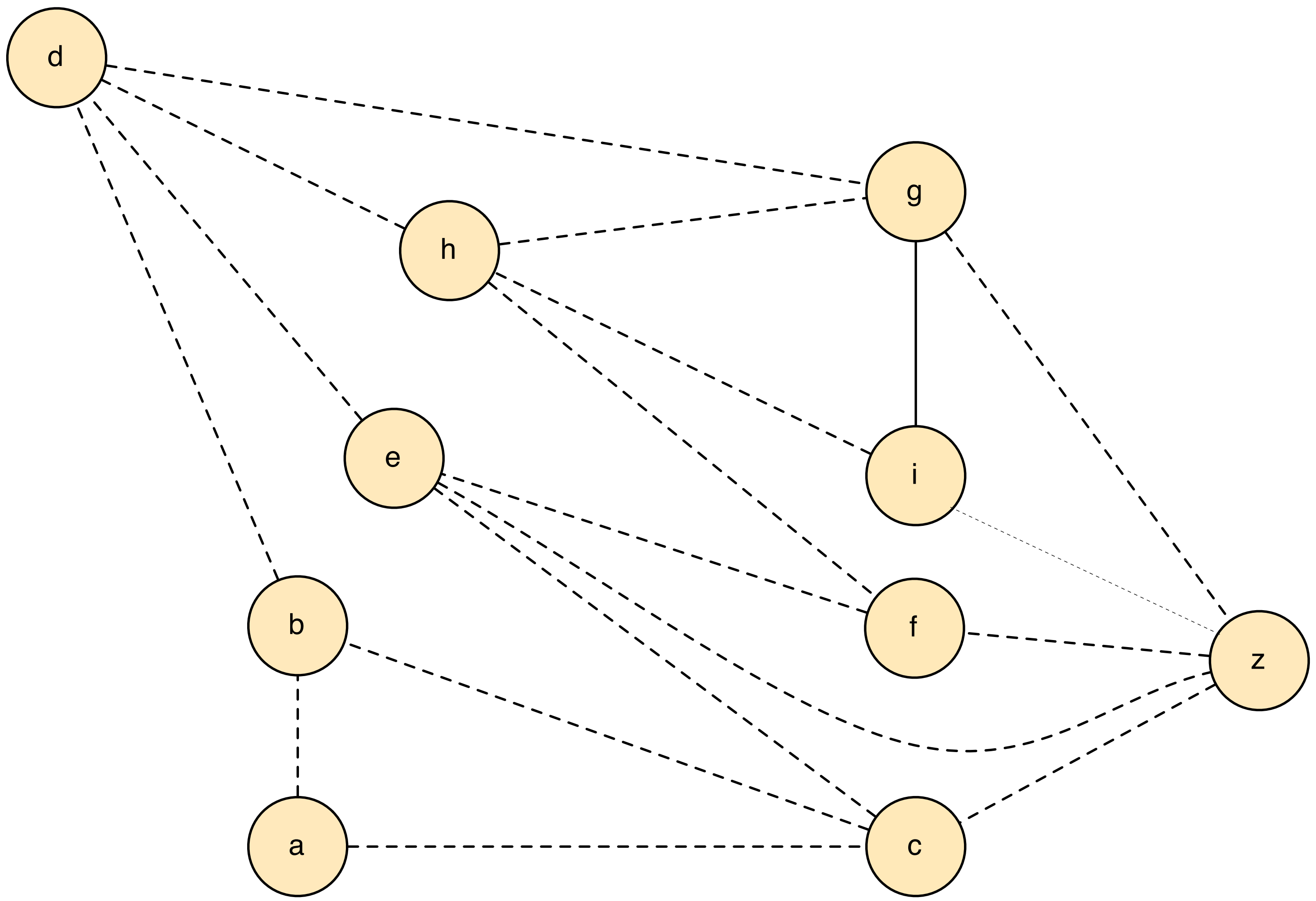
analysis

IF G HAS k DISJOINT PATHS, THEN

analysis

G’ HAS A FLOW OF K, THEN

vertex-disjoint paths

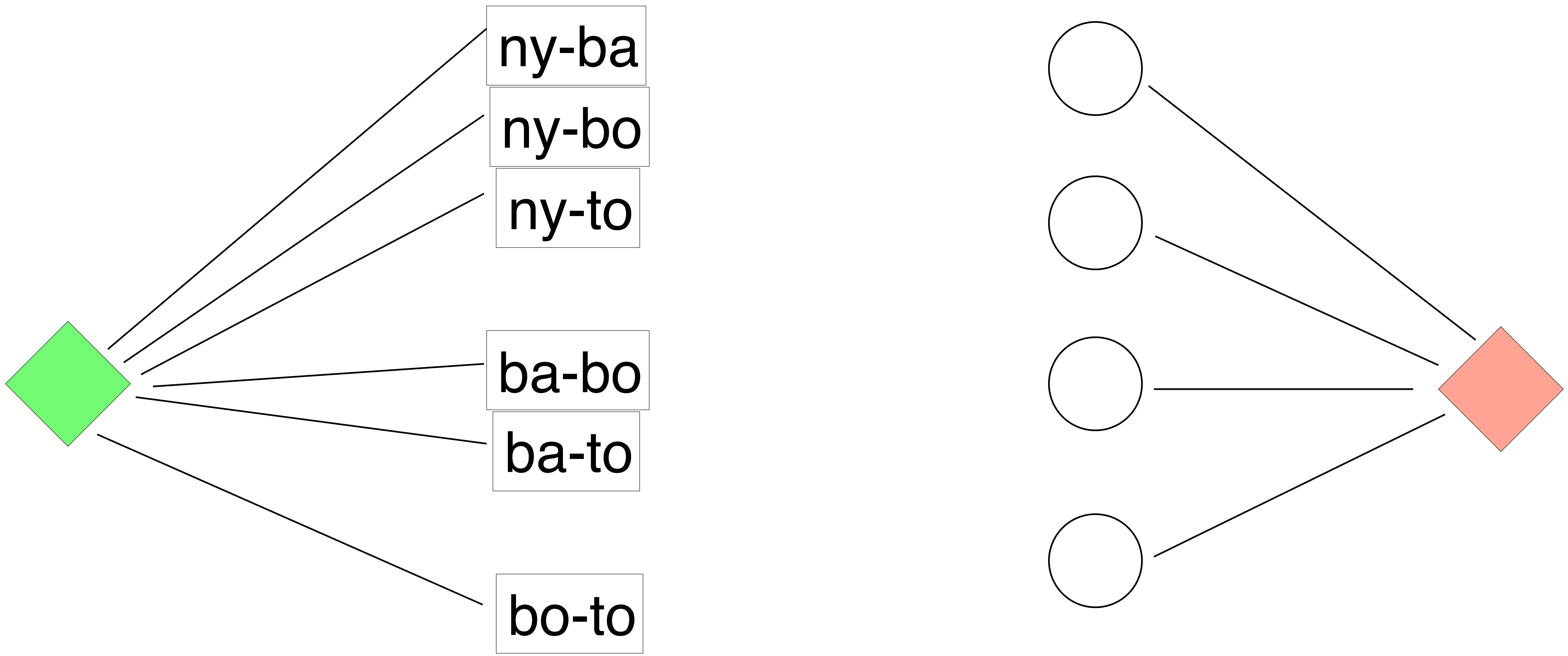


baseball elimination

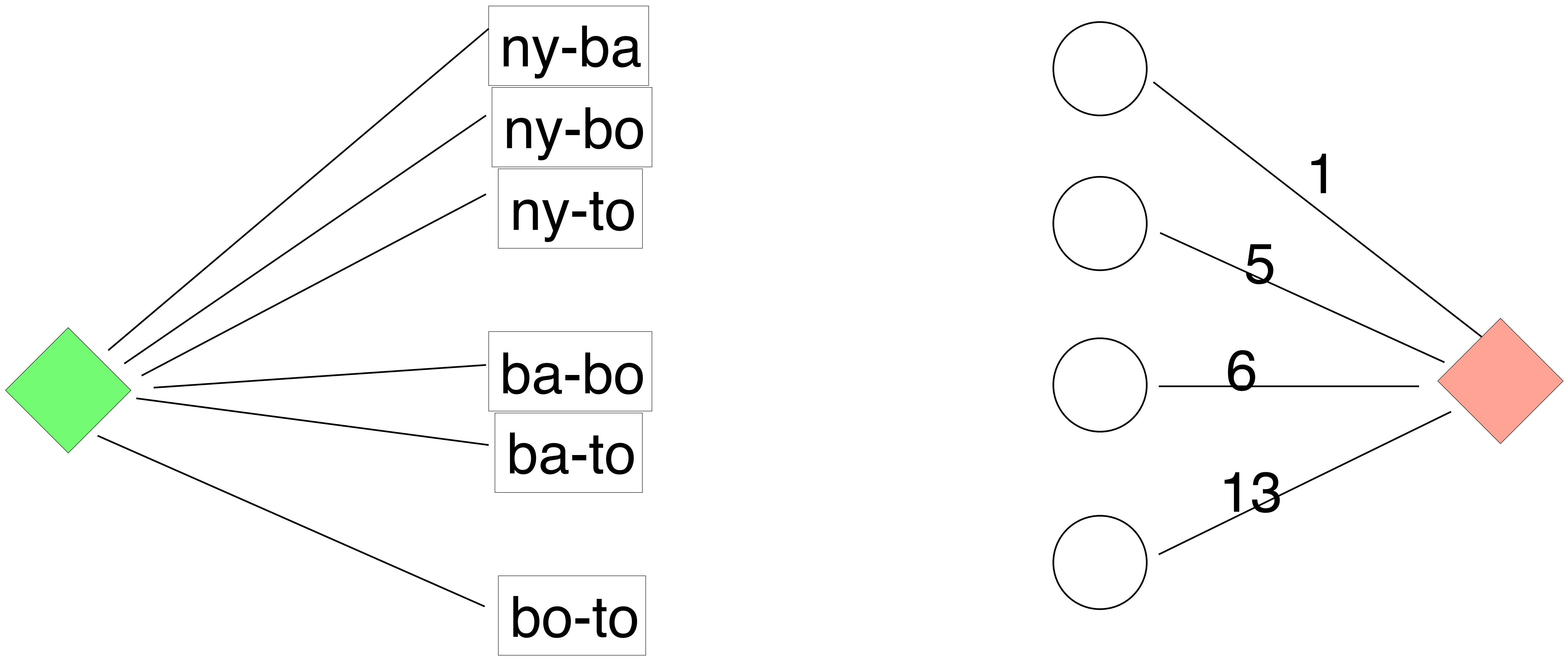
				Against			
	W	L	Left	A	P	N	M
ATL	83	71	8	-	1	6	1
PHL	80	79	3	1	-	0	2
NY	78	78	6	6	0	-	0
MONT	77	82	3	1	2	0	-

baseball elimination

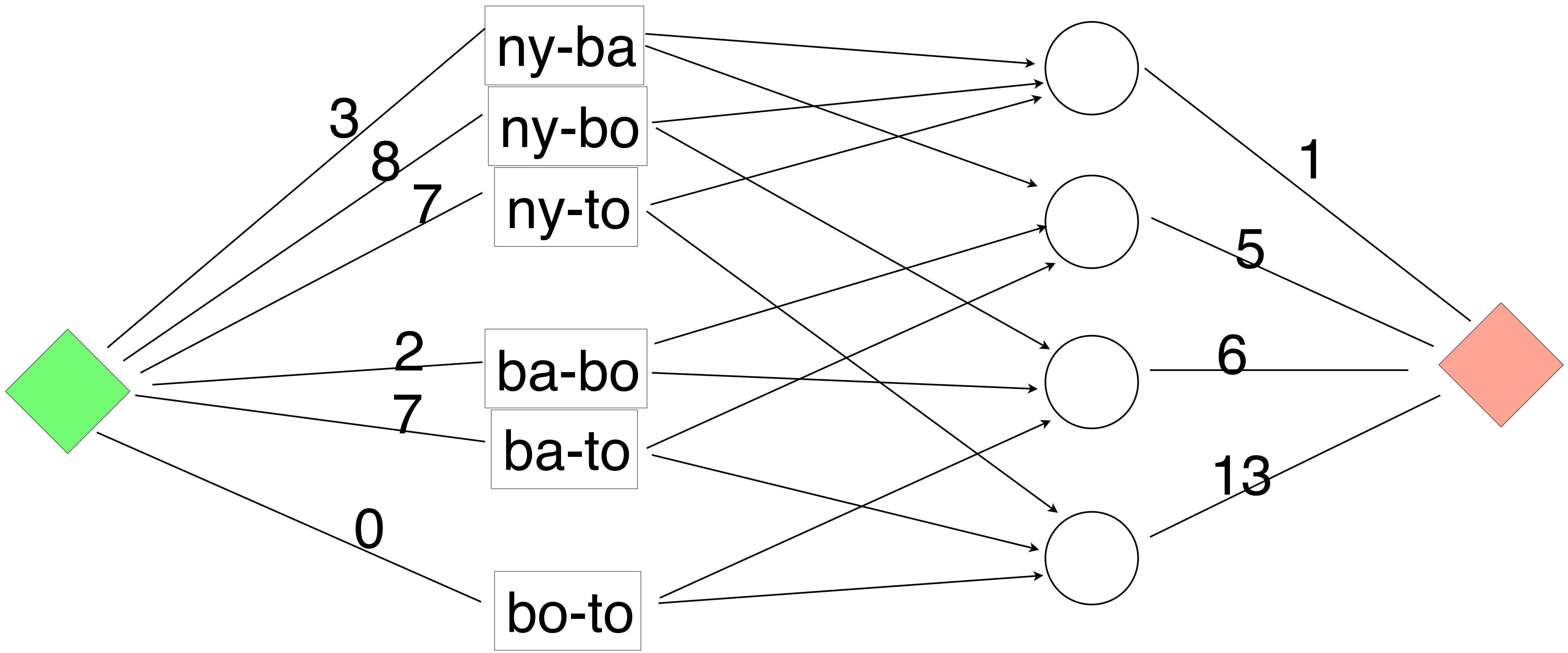
	W	L	Left	Against N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			



	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			



	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			