L22

4102

11.12.2013

abhi shelat

max flow

Max flow

Min Cut

"Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other."

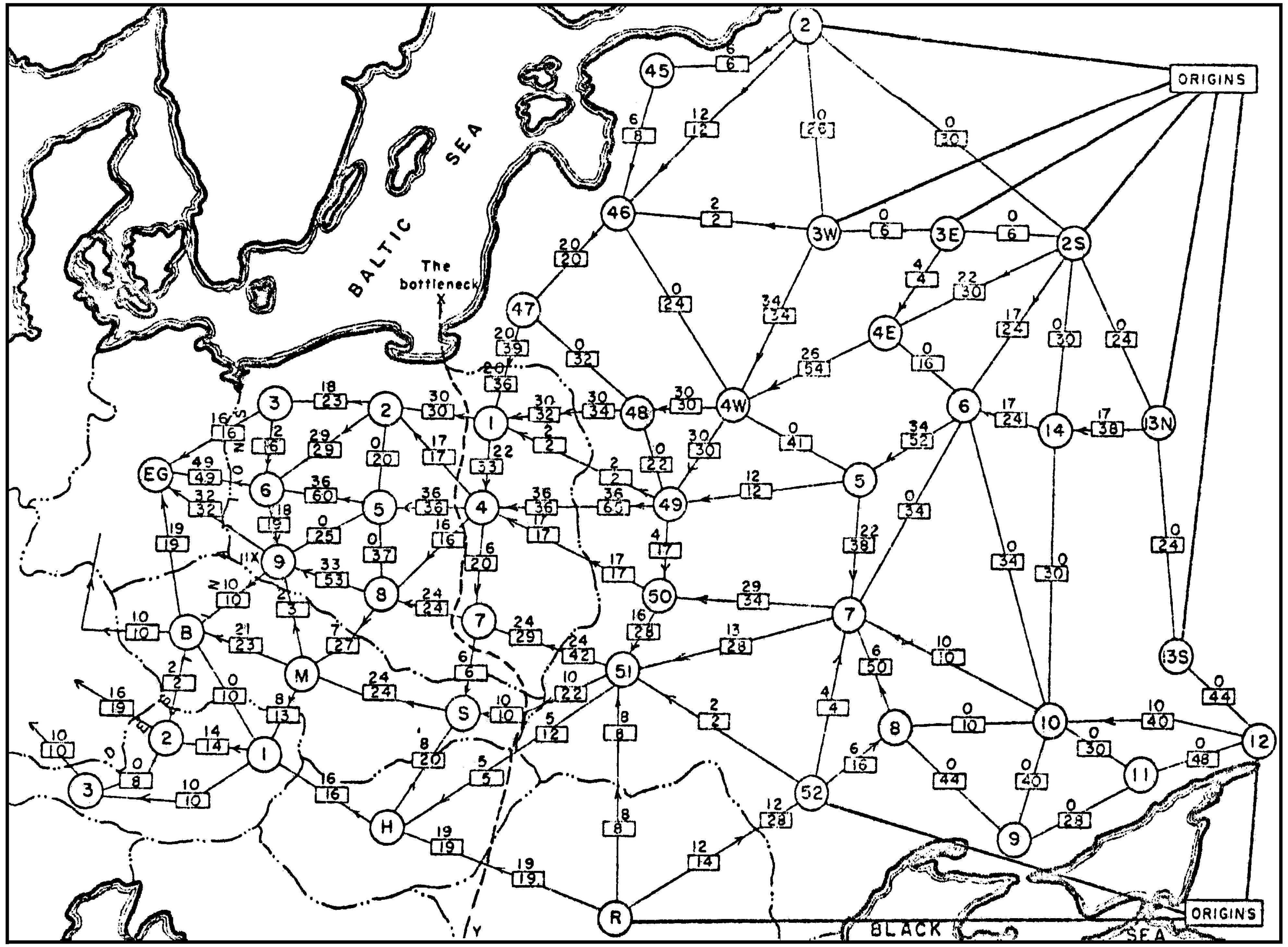


Figure 4 From Harris and Ross [3]: Schematic diagram of the railway network of the Western Soviet Union and East European countries, with a maximum flow of value 163,000 tons from Russia to Eastern Europe and a cut of capacity 163,000 tons indicated as 'The bottleneck'

courtesy Alexander Schrijver

FLOW NETWORKS

$$G = (V, E)$$

SOURCE + SINK: s t

CAPACITIES: $c : E \to \mathbb{Q}^+$ ⤴ rational positive numbers

FLOW

MAP FROM EDGES TO NUMBERS: $f: E \to Q^+$

CAPACITY CONSTRAINT: $f(e) \le c(e)$

FLOW CONSTRAINT: for every node $v \in V - \{s, t\}$

$IN(v) = OUT(v) \Rightarrow \sum_u f(u,v) = \sum_w f(v,w)$

$$|f| =$$

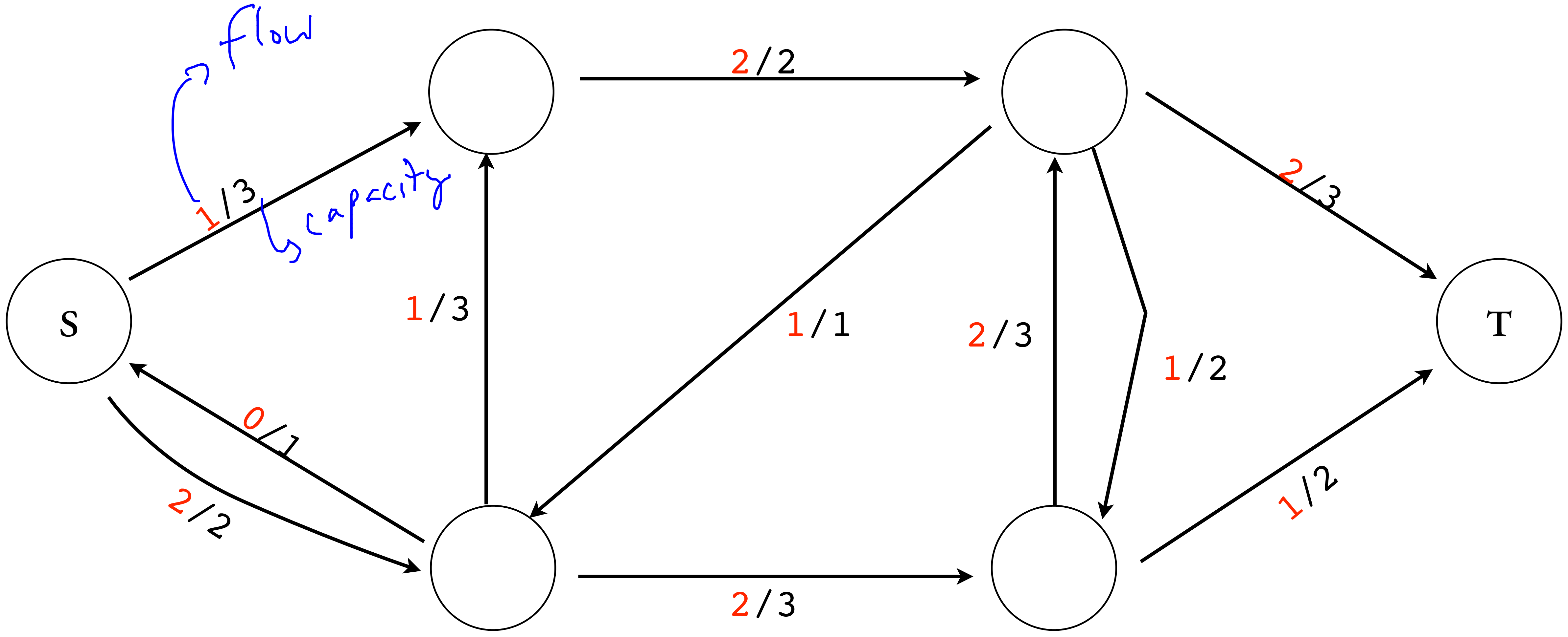
$OUT(s) - IN(s)$

MAX FLOW PROBLEM

GIVEN A GRAPH G, COMPUTE

$$\operatorname{ARGMAX}_f |f|$$

EXAMPLE of A FLOW



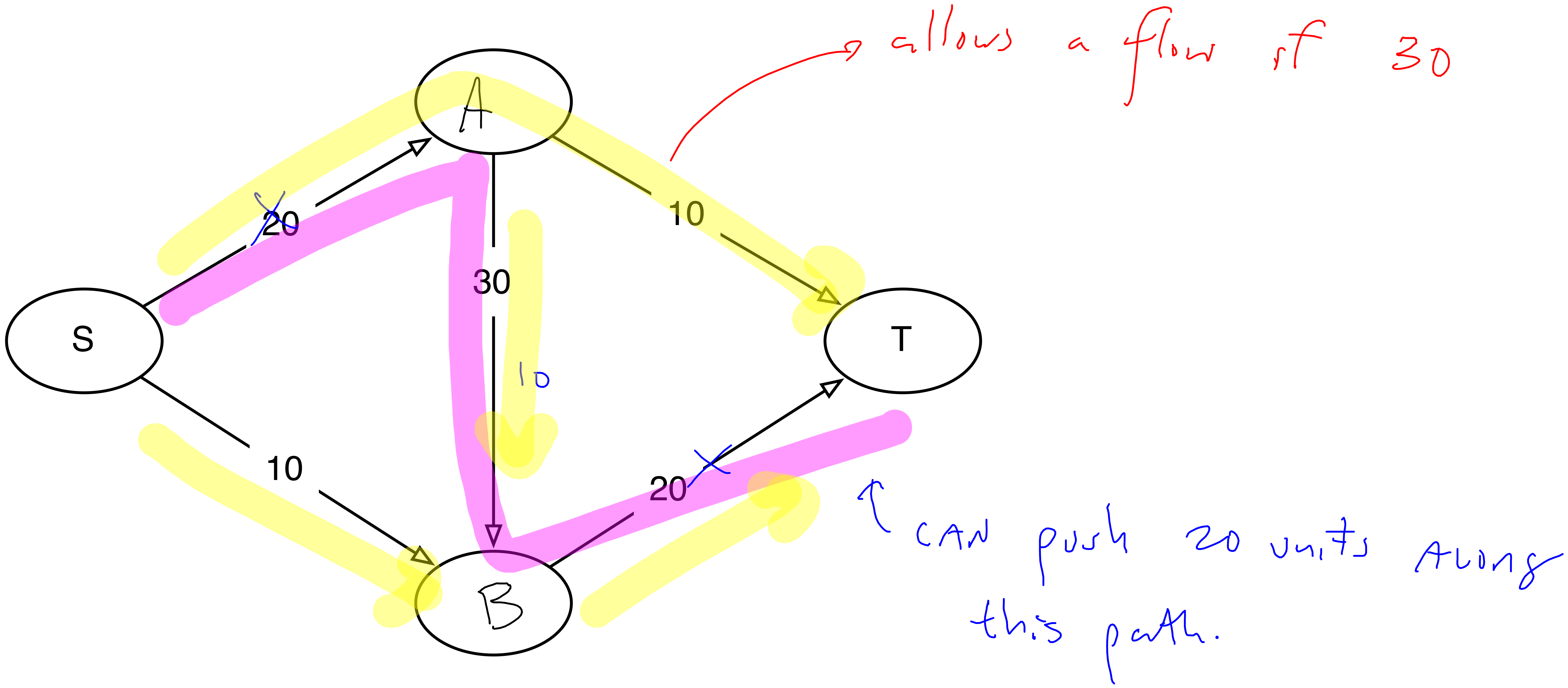
HUNDREDS OF APPLICATIONS

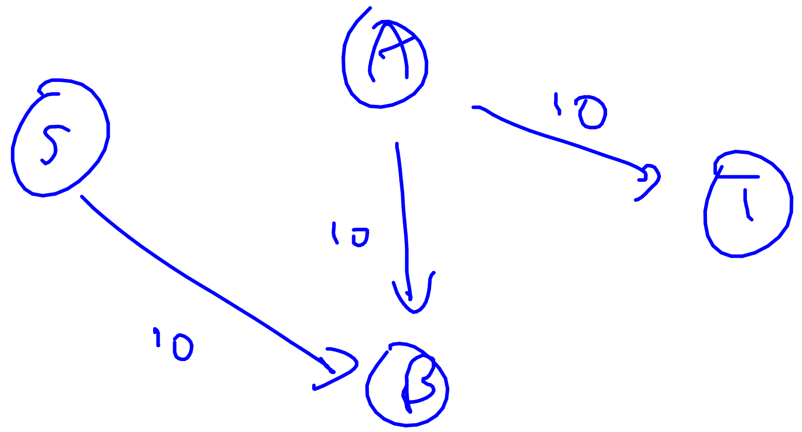
- bipartite matching
- edge-disjoint paths
- node-disjoint paths
- scheduling
- baseball elimination
- resource allocations

will discuss many of these applications in L22.

ALGORITHMS FOR MAX FLOW

GREEDY FAILS





RESIDUAL GRAPHS

$G_f = (V, E_f)$

given a flow f, one can construct a residual graph G_f using this rule

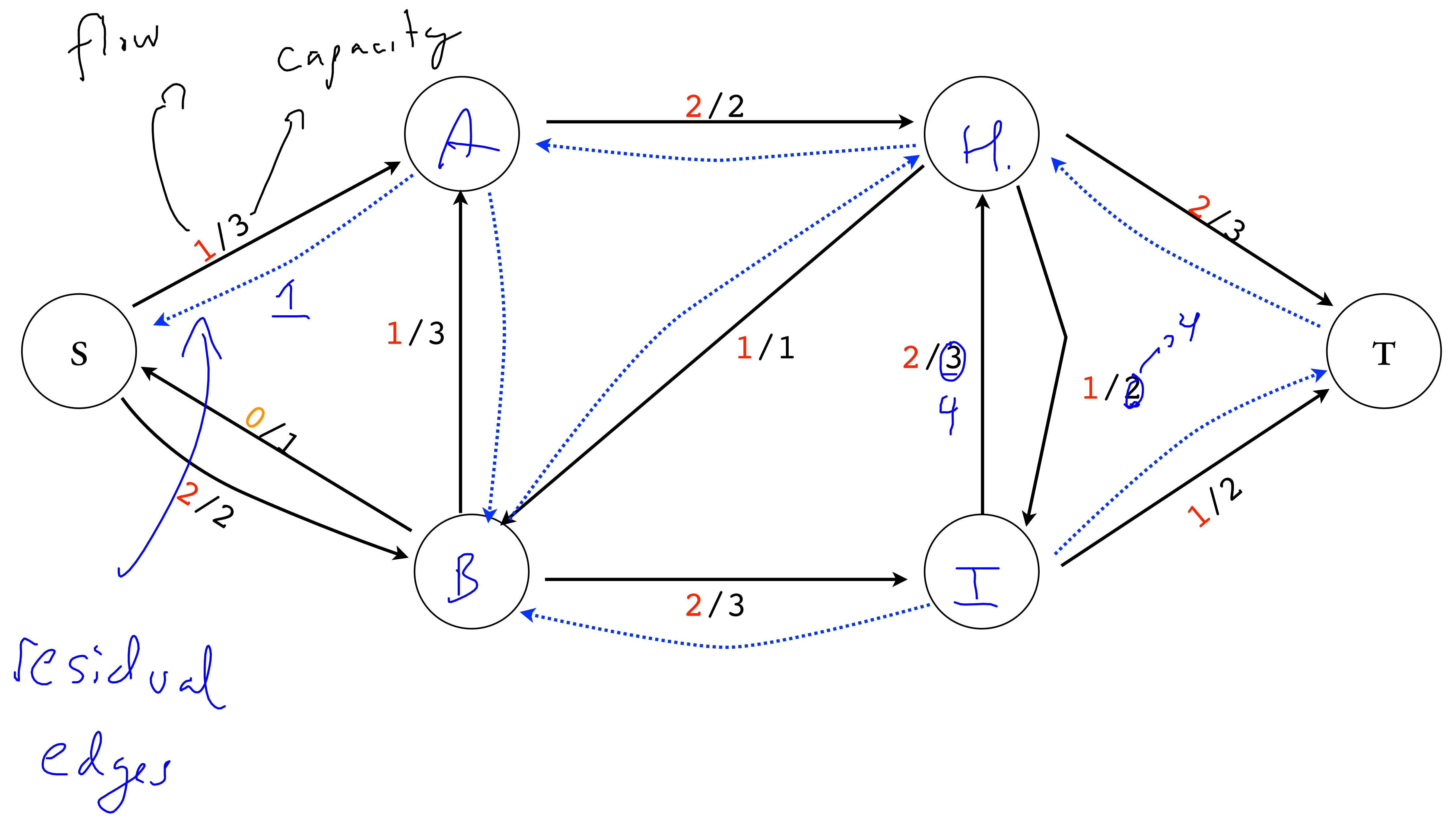
"whenever you push x units on edge (u,v), create a residual edge from (v,u) w/ capacity x."

$$c_f(u,v) =$$

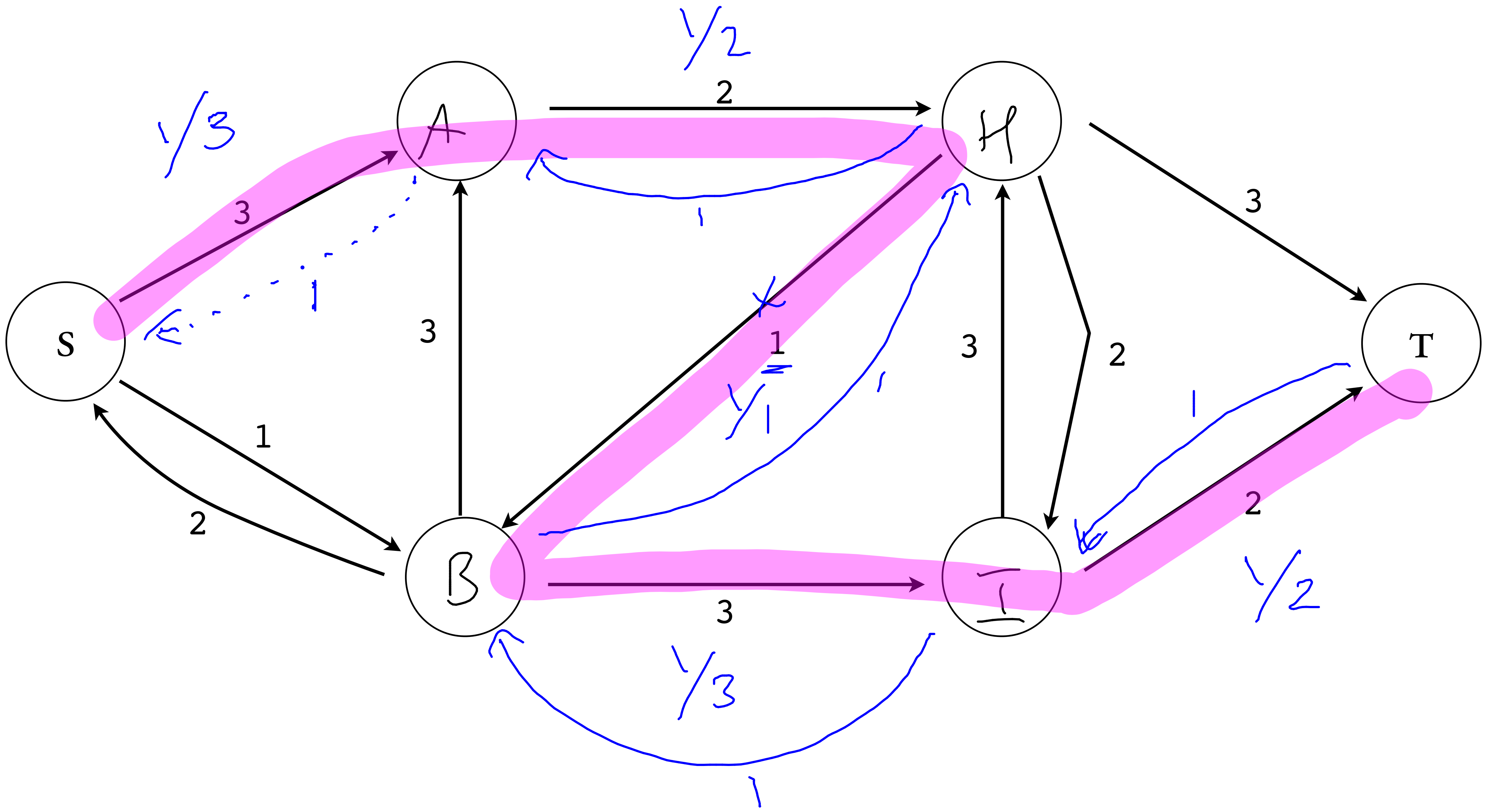
AUGMENTING PATHS

DEF: Any path from s to t in a residual graph G_f.

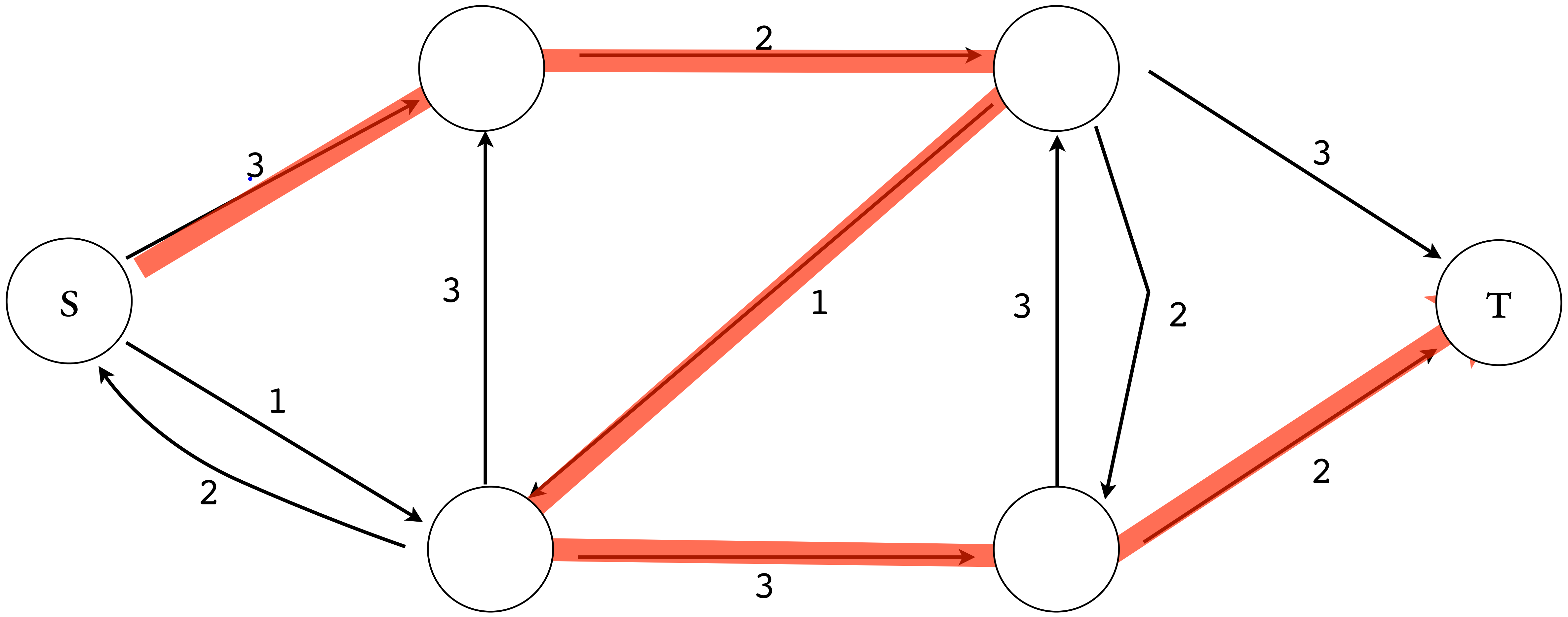
EXAMPLE RESIDUAL GRAPH

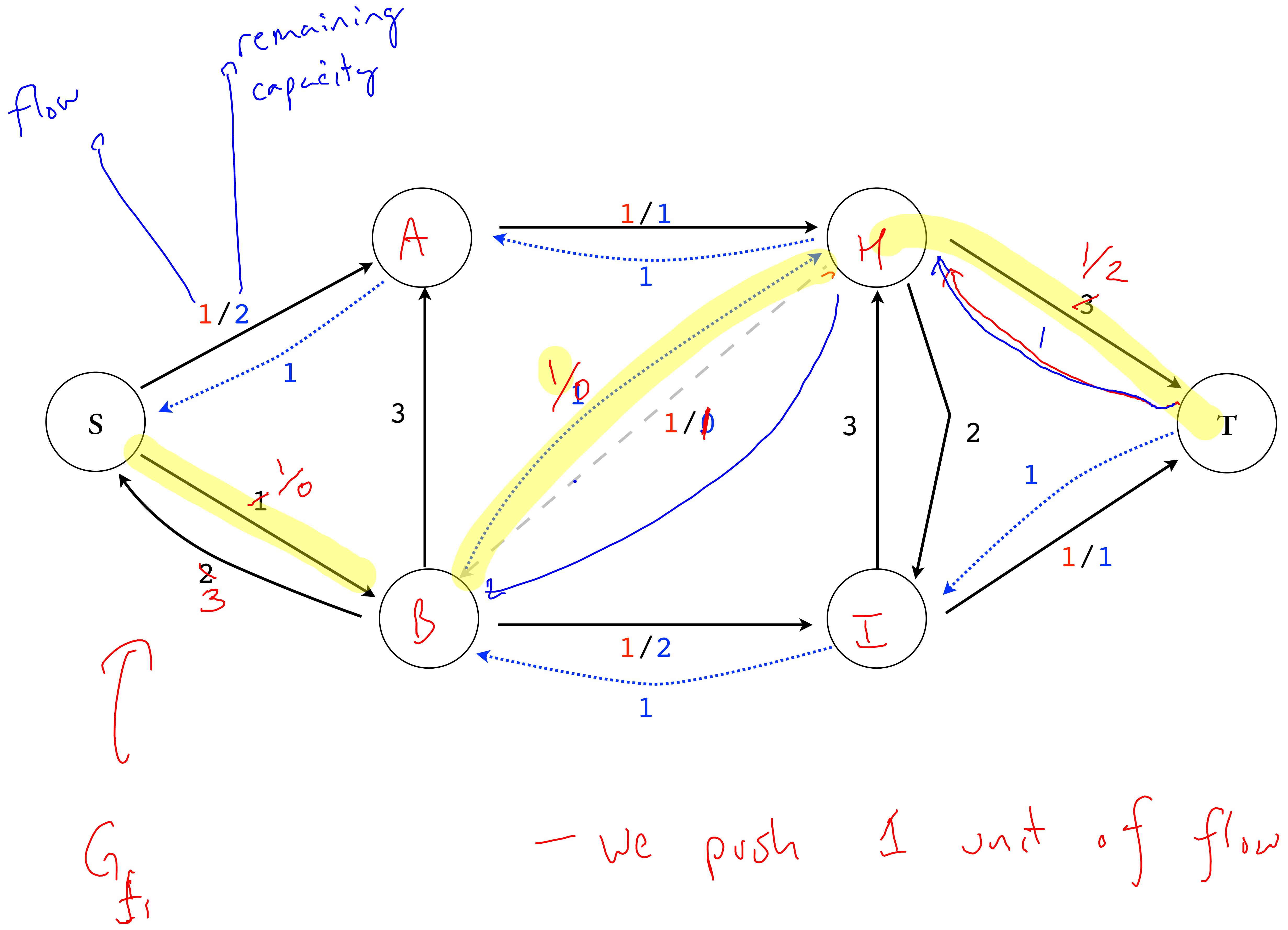


$G = (V, E)$ c

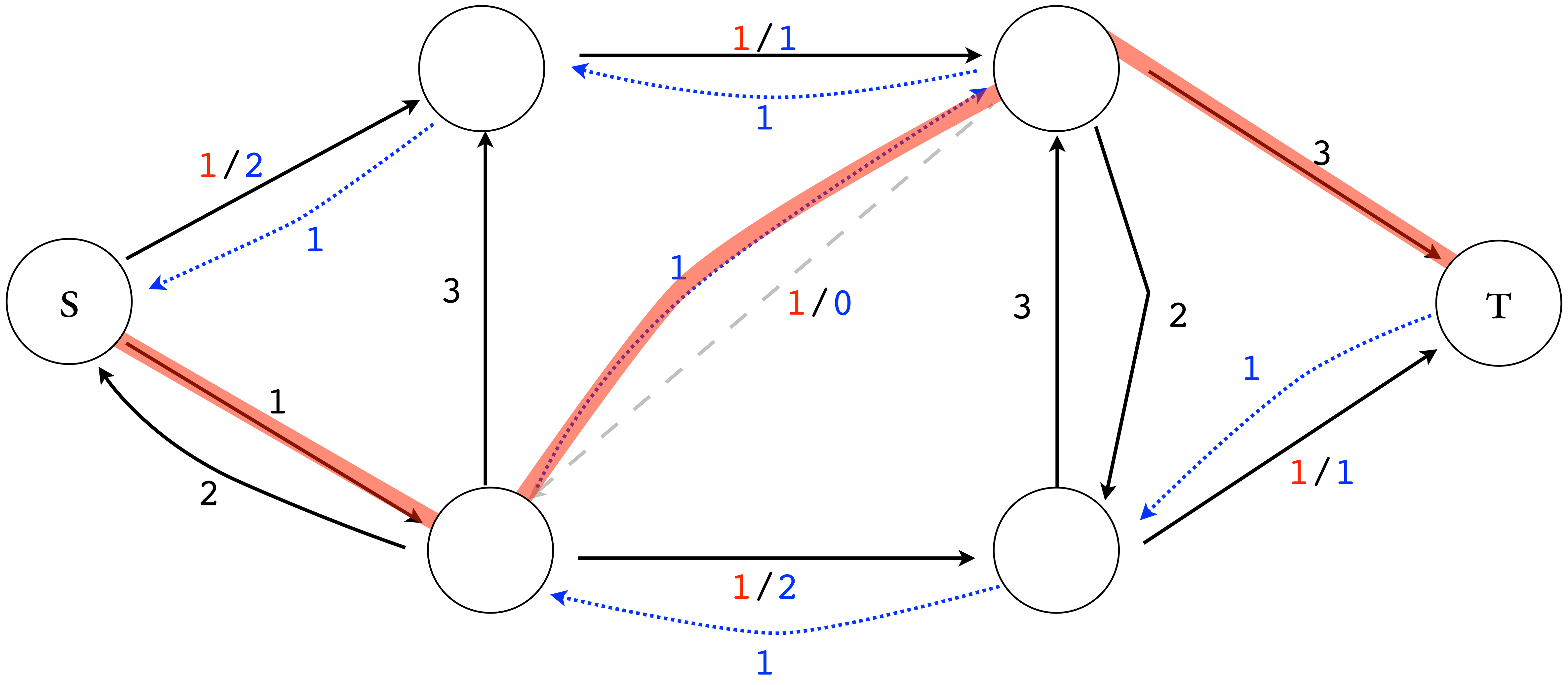


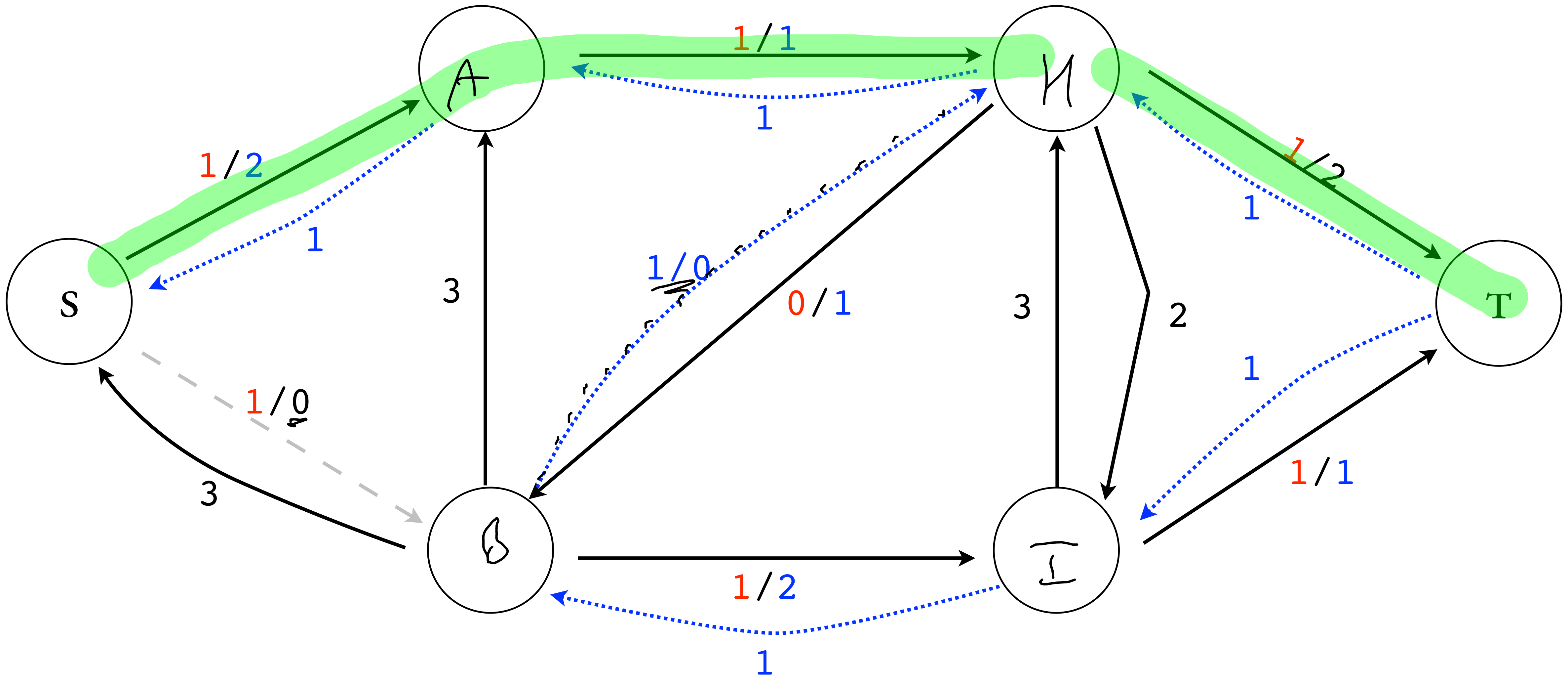
- 1st augmenting path
- we can push 1 unit


S
T
3
2
3
1
1
2
3
3
2
3
2

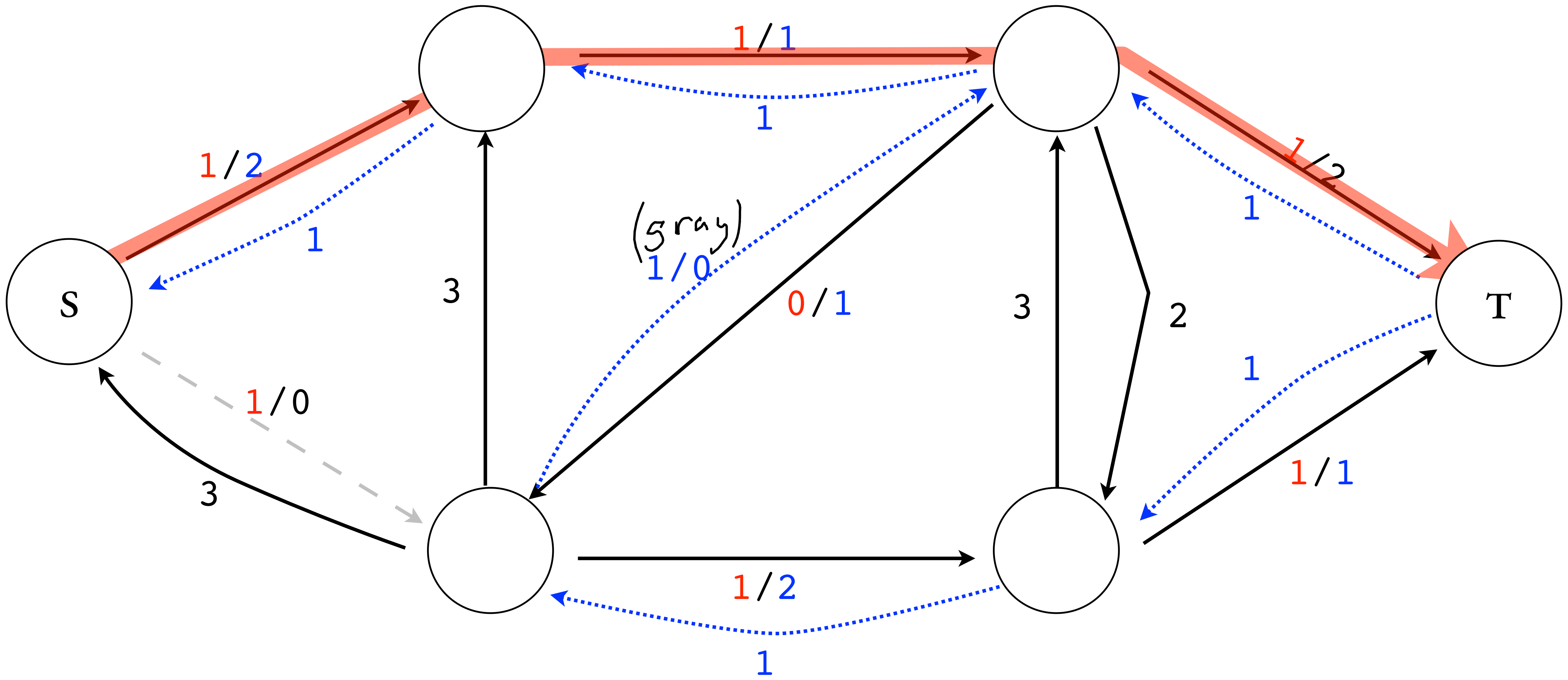


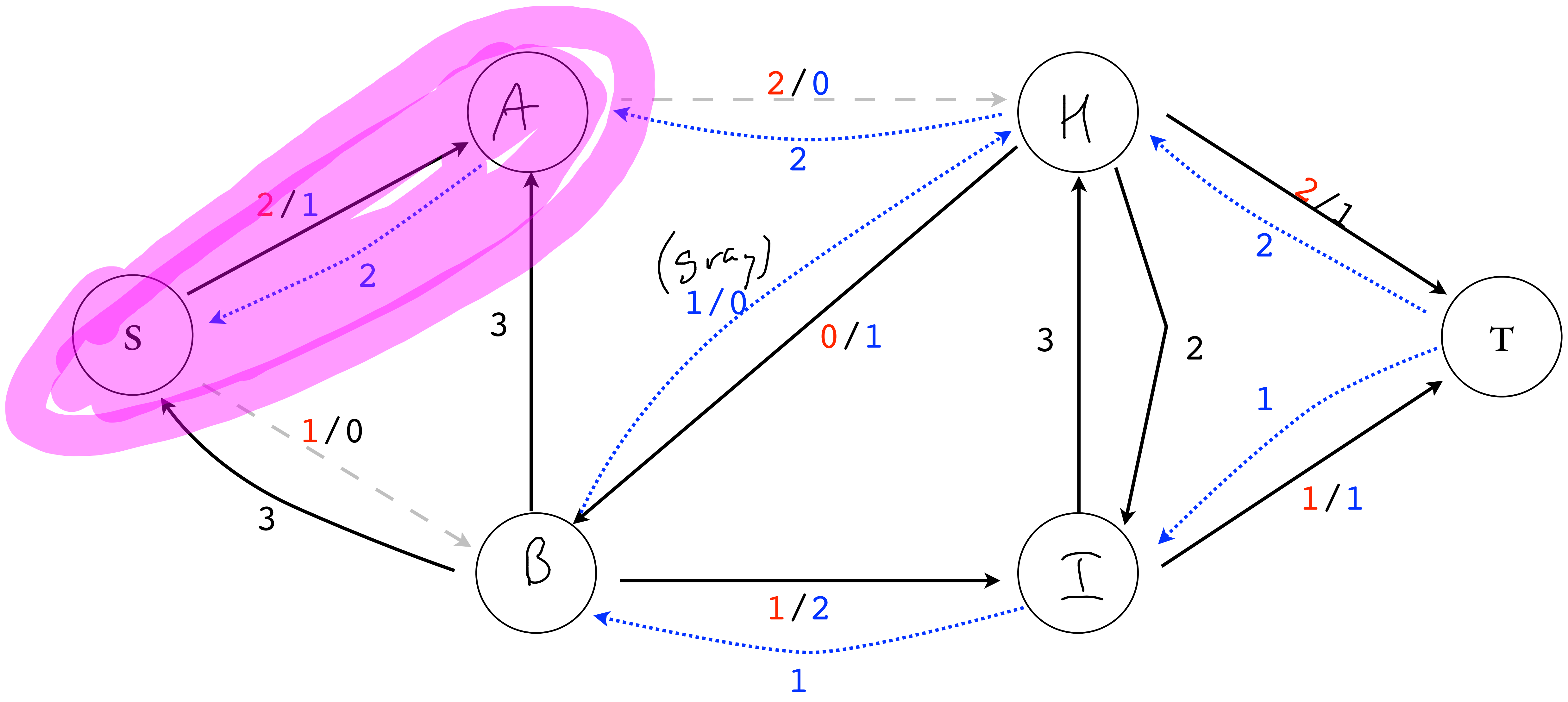
- we push 1 unit of flow S → B → H → T


S
T
1 / 2
1
1 / 1
1
3
1
1 / 0
3
3
2
1
1 / 1
1
2
1 / 2
1



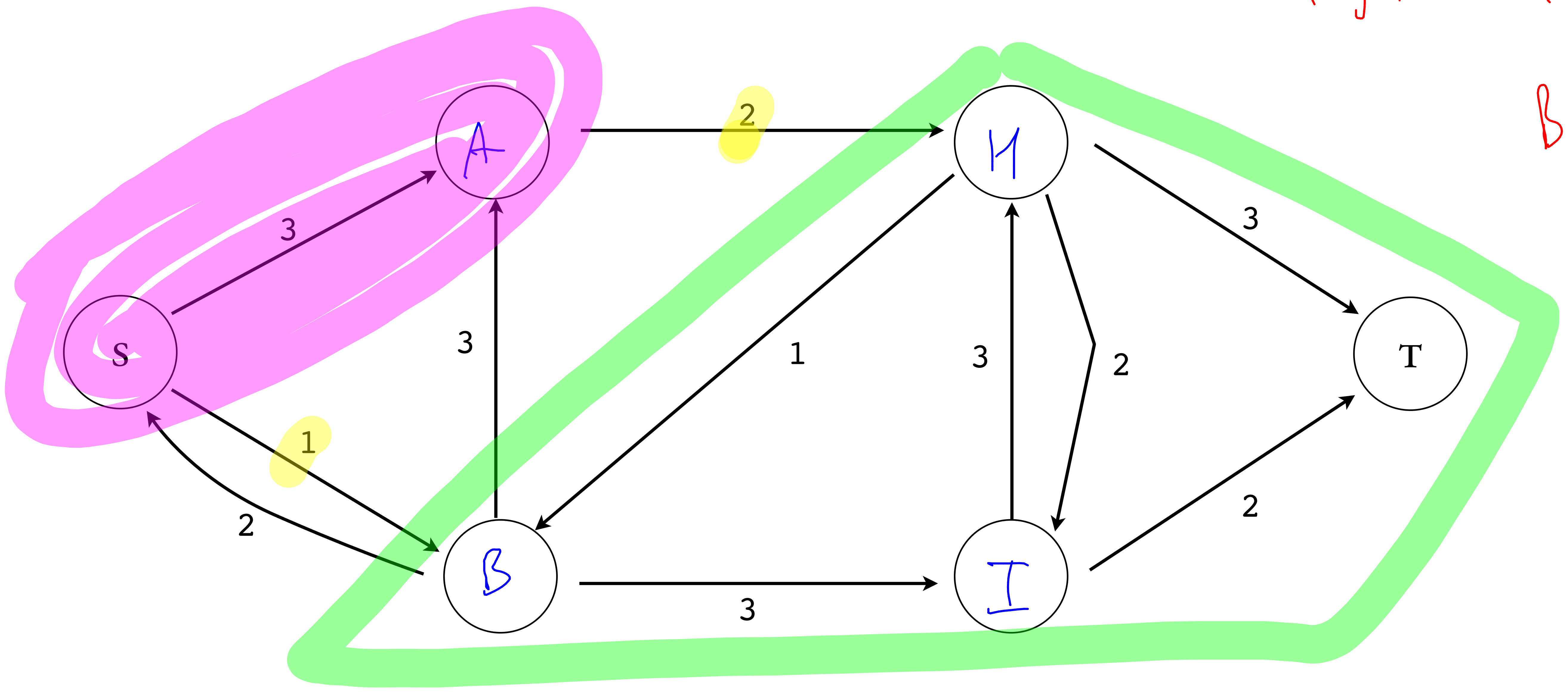
S
A
H
T
B
I
1/2
1
1/1
1
1/2
1
3
1/0
0/1
3
2
1
1/0
3
1/1
1/2
1


S
T
1/2
1
1/1
1
1/2
1
(gray)
1/0
0/1
3
3
2
1
1/1
1/0
3
1/2
1



No more augmenting paths from s to t!!

⇒ done.



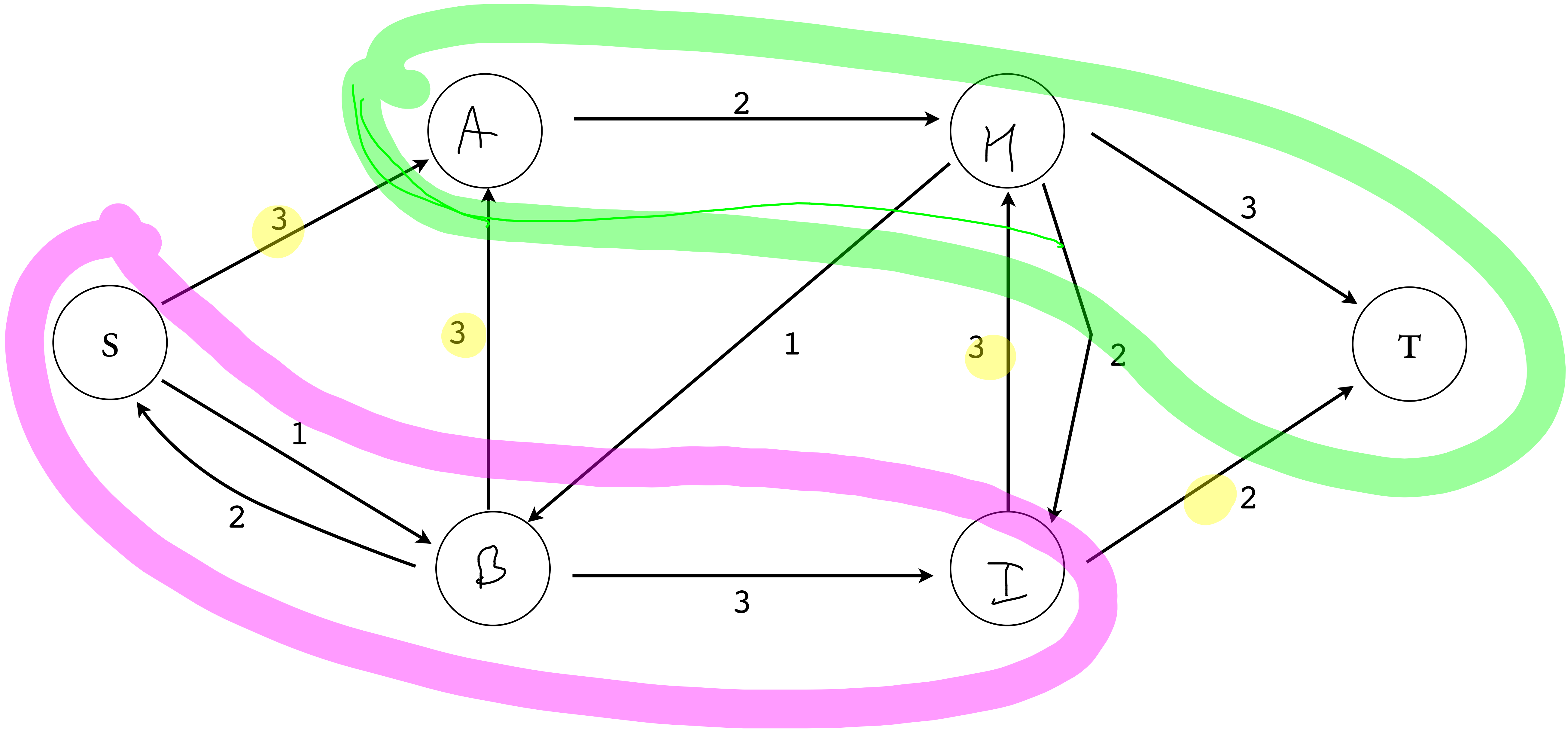
$|f| = \| \{S,A\}, \{B,H,I,T\} \|$

By lemma, there cannot be a larger flow.

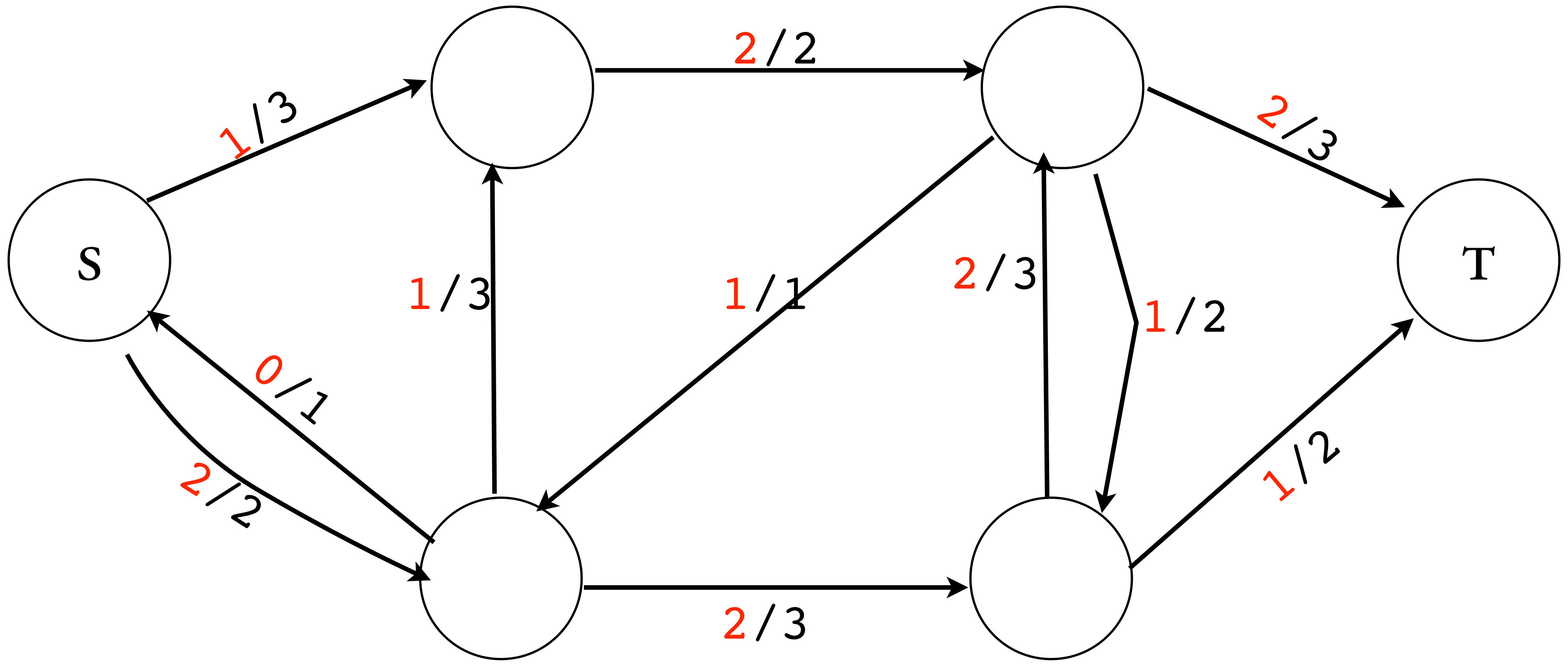
{S,A} {BHIT} is a graph cut.

This cut has value 3.

OTHER CUTS ARE LARGER



FOR ANY $f, (S,T)$ IT HOLDS THAT $|f| \leq ||S,T||$



EXAMPLE:

CUTS

DEF OF A CUT:

COST OF A CUT:

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

LEMMA: [MIN CUT] FOR ANY $f, (S, T)$, $|f| \le \|S, T\|$

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.

THM: MAX FLOW = MIN CUT

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

FORD-FULKERSON

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

WHILE EXISTS AN AUGMENTING PATH p IN G_f

AUGMENT f WITH $c_f(p) = \min_{(u,v) \in p} c_f(u,v)$

WHY DOES FF WORK? (HIGH LEVEL)

We simultaneously construct a flow f and cut (S, T)

s.t. $|f| = \|S, T\|$

FORD-FULKERSON

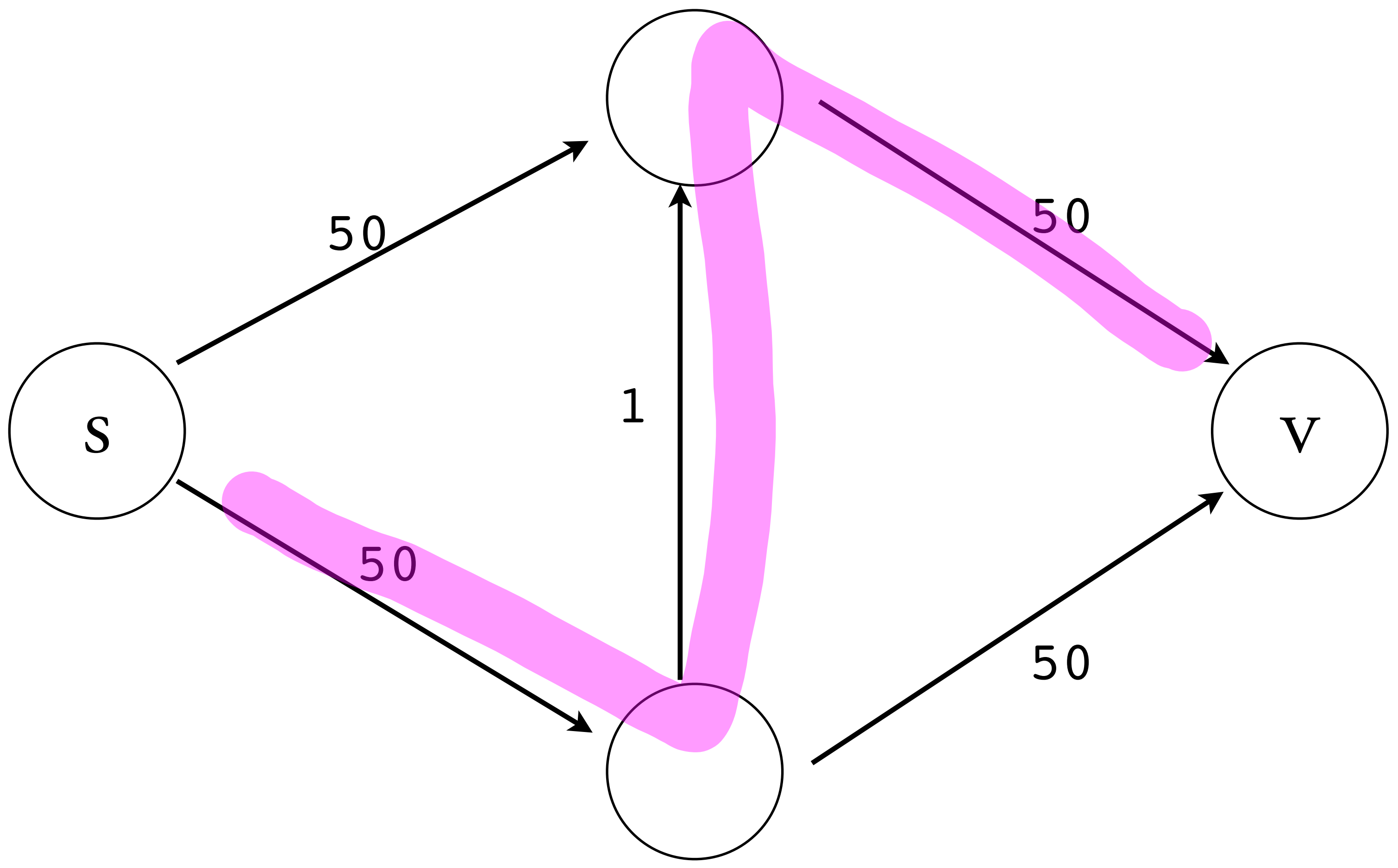
INITIALIZE $f(u,v) \leftarrow 0 \; \forall u, v$

WHILE EXISTS AN AUGMENTING PATH p IN G_f

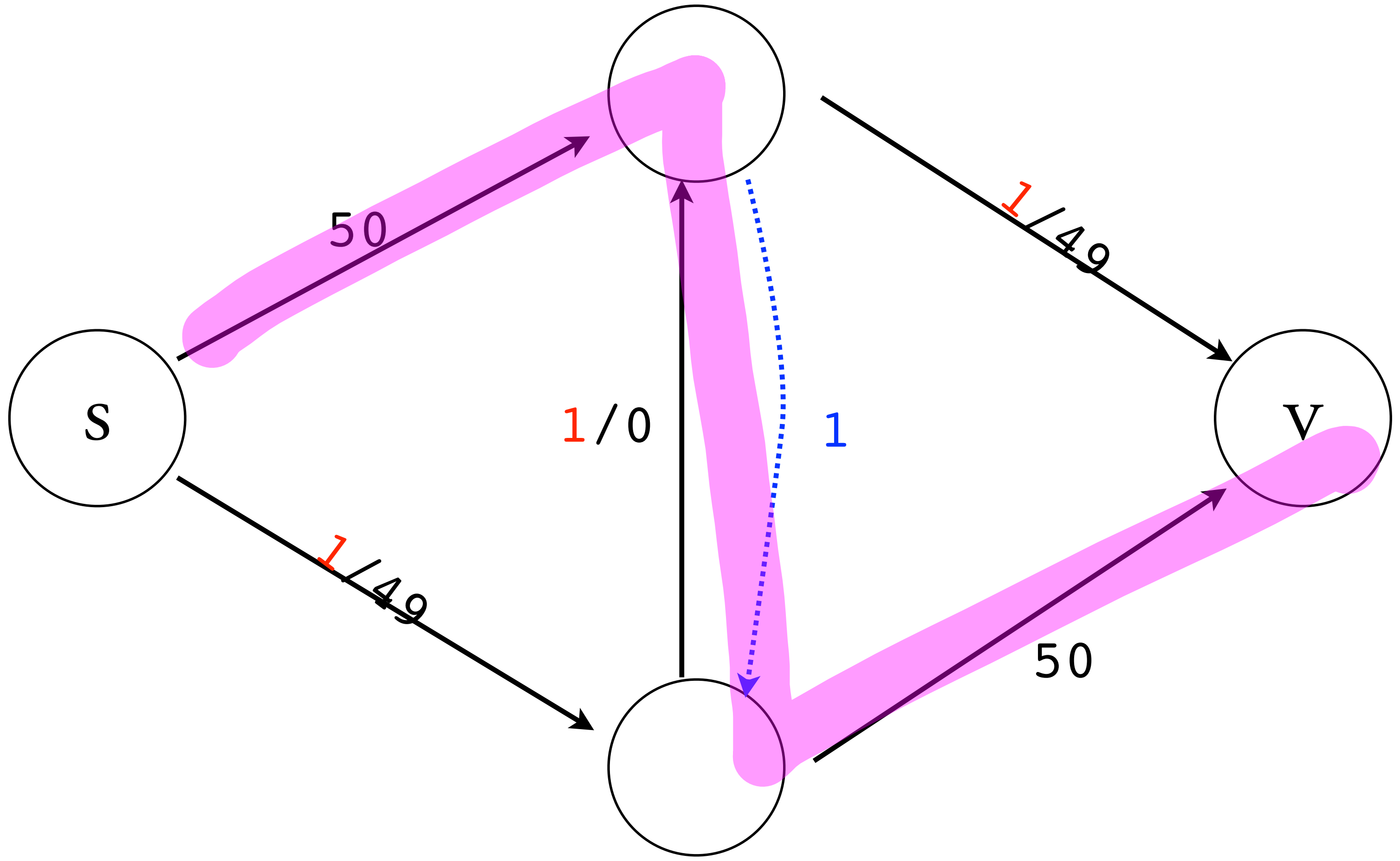
AUGMENT f WITH $c_f(p) = \min_{(u,v) \in p} c_f(u,v)$

TIME TO FIND AN AUGMENTING PATH:

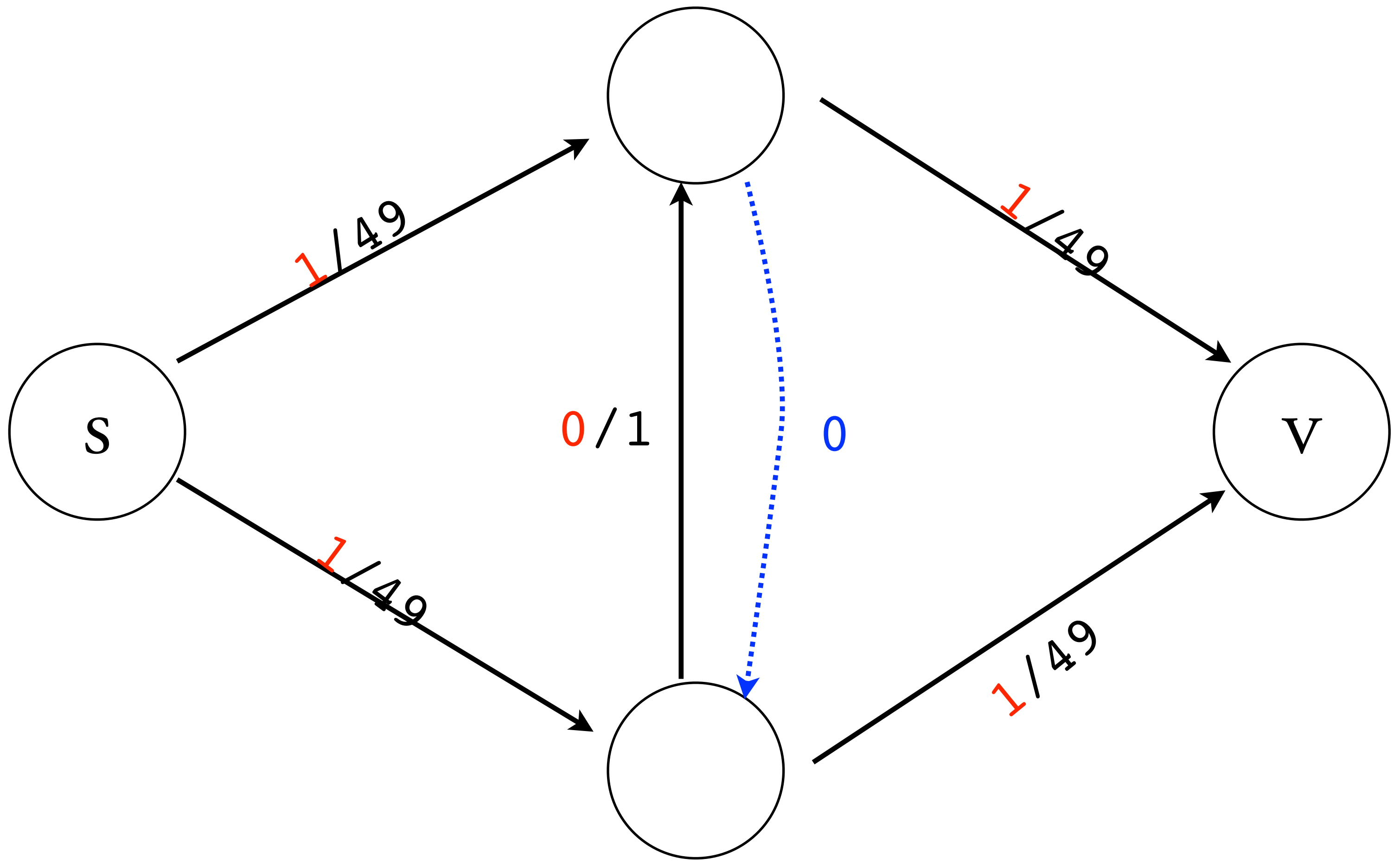
NUMBER OF ITERATIONS OF WHILE LOOP: $|f^*|$ — max flow



s
v
50
50
1
50
50

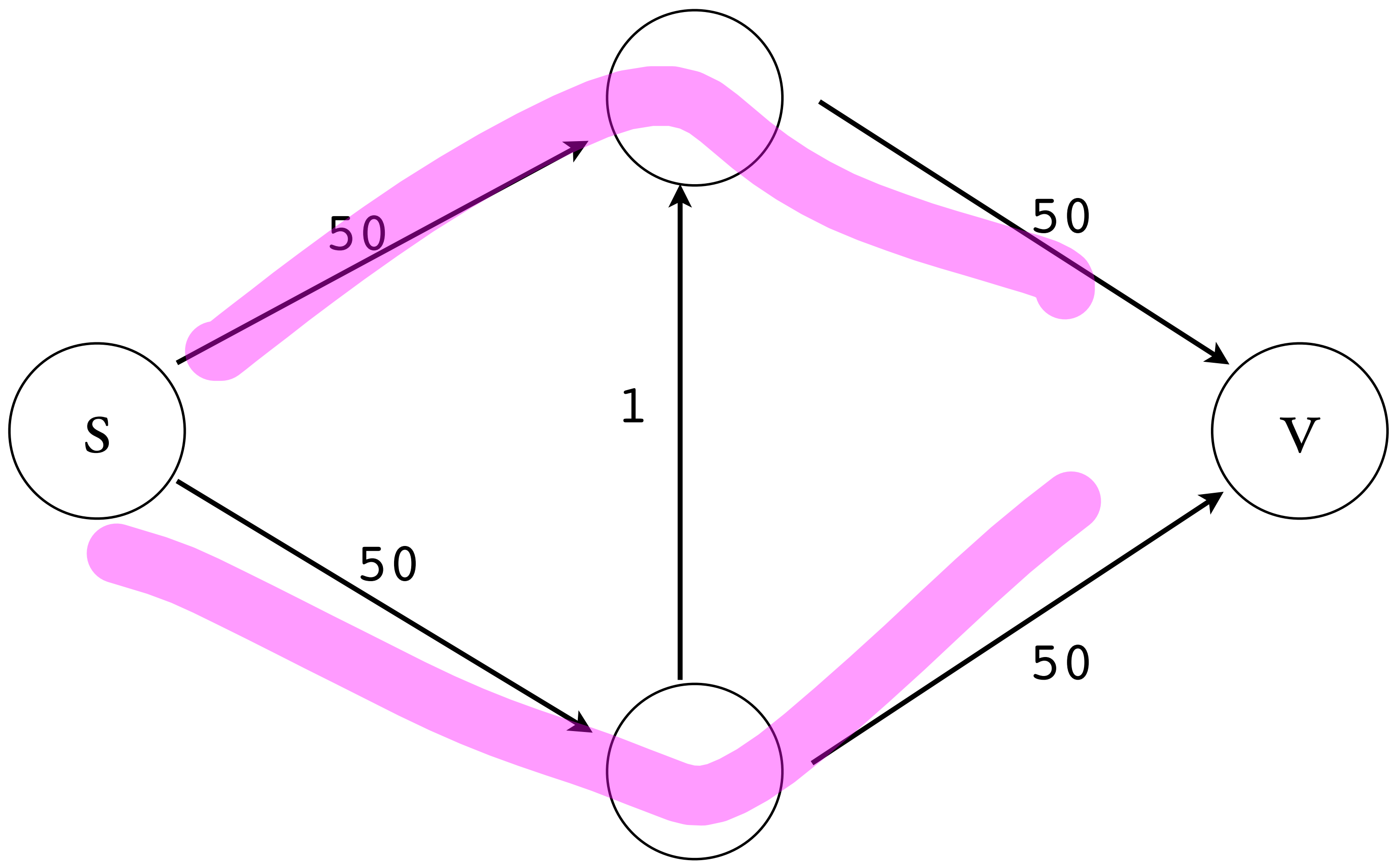


S
V
50
1/49
1/0
1
1/49
50



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

ROOT OF THE PROBLEM



picking bad paths.

EDMONDS-KARP 2

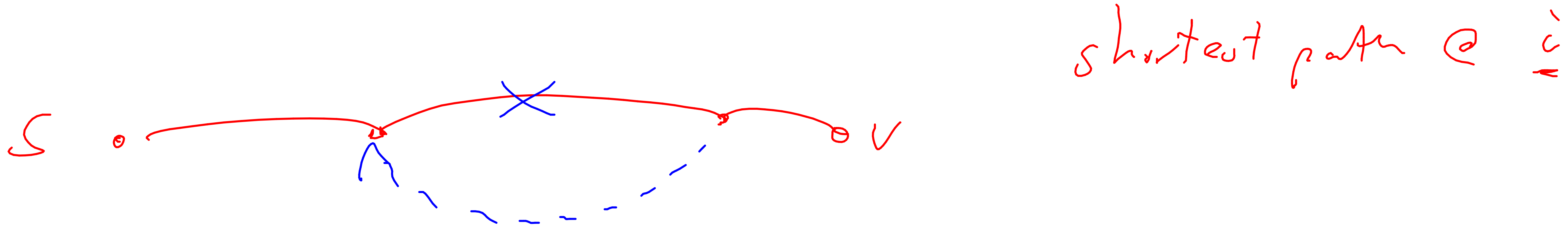
CHOOSE PATH WITH FEWEST EDGES FIRST.

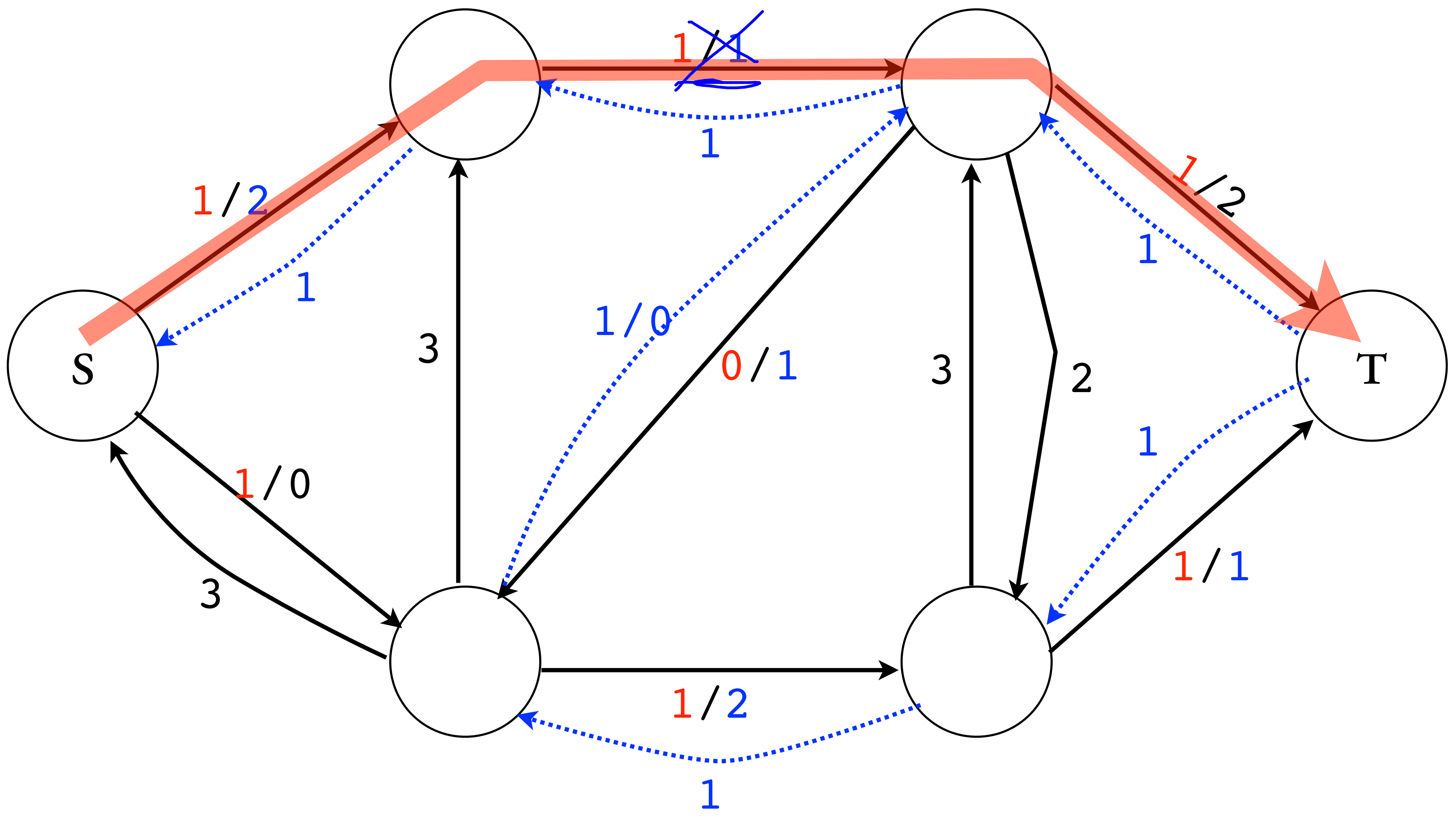
$\delta_f(s,v)$: # of hops from s to v along the shortest path in residual graph G_f.

LEMMA:

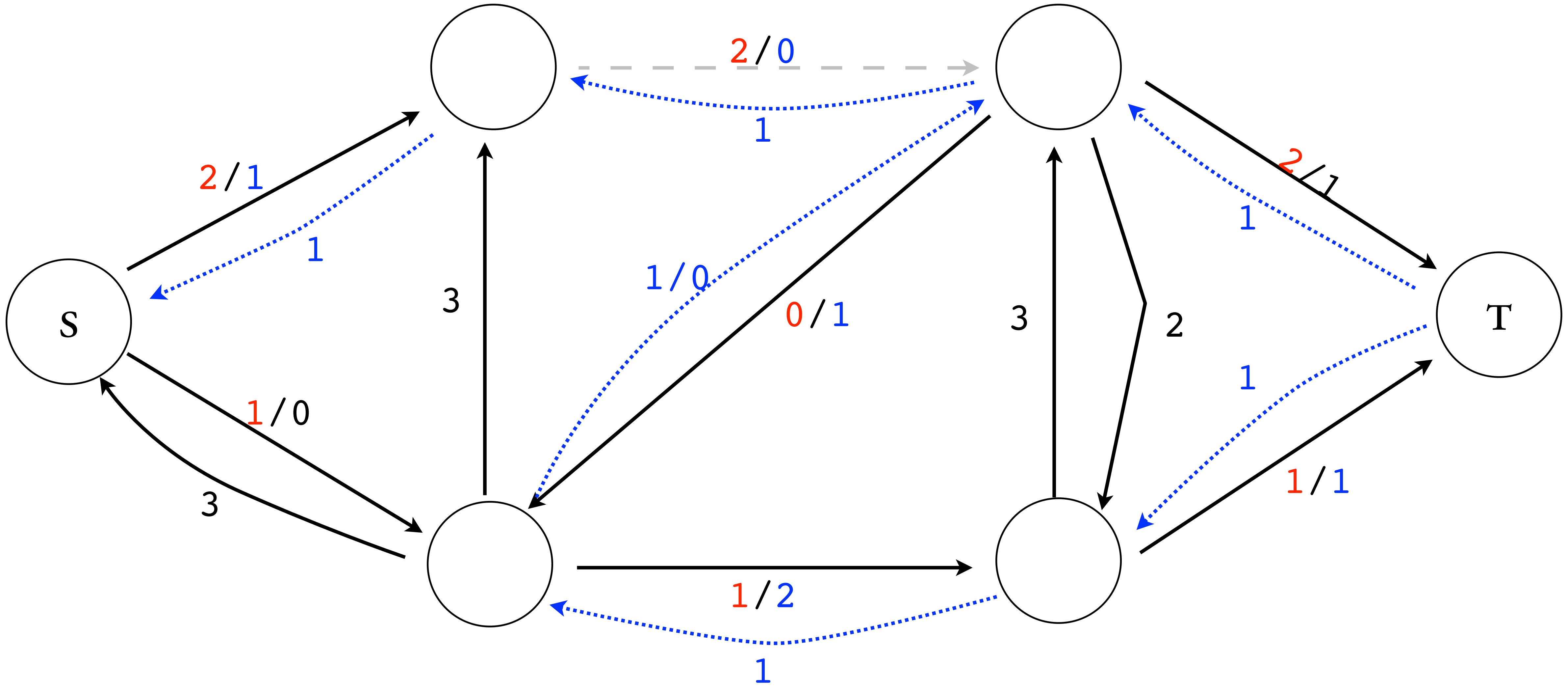
$\delta_f(s,v)$ INCREASES MONOTONICALLY THRU EXEC

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

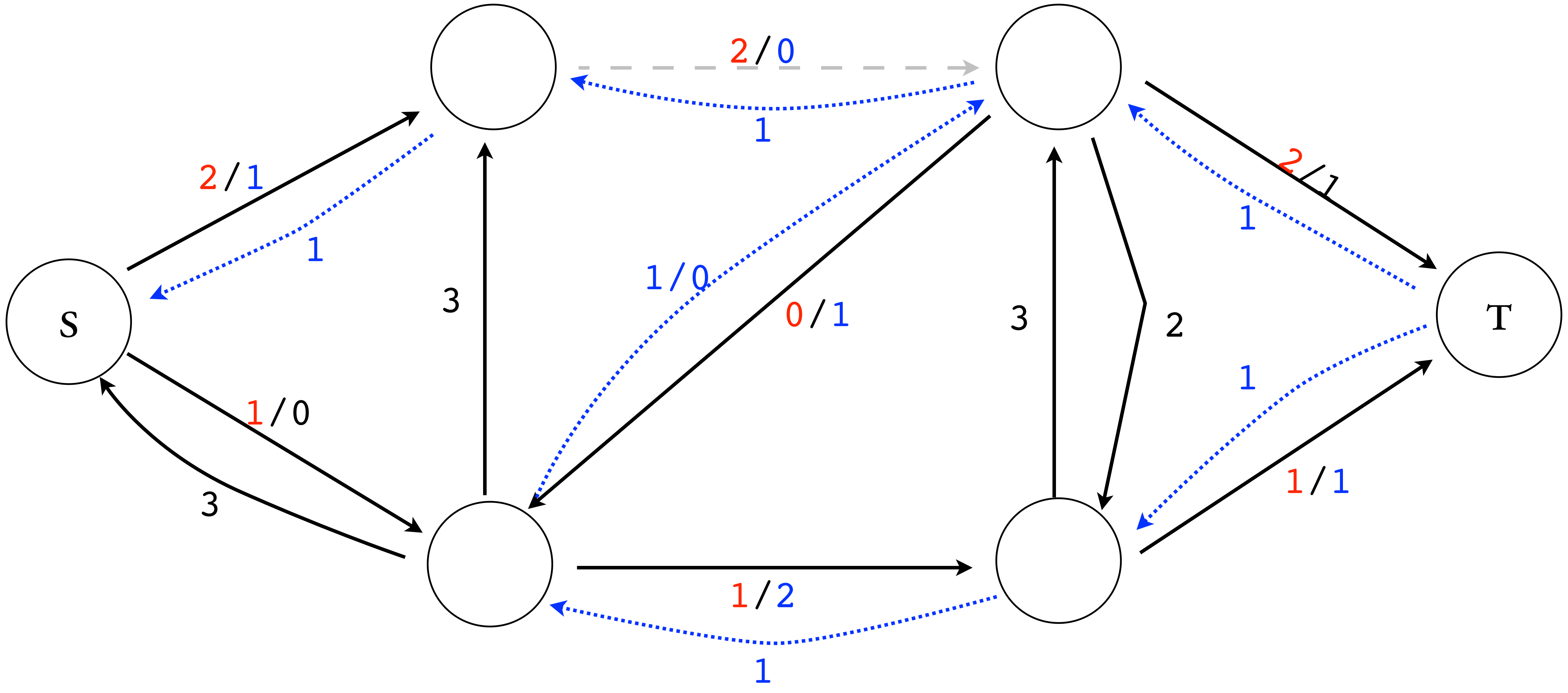




FOR EVERY AUGMENTING PATH, SOME EDGE IS CRITICAL.

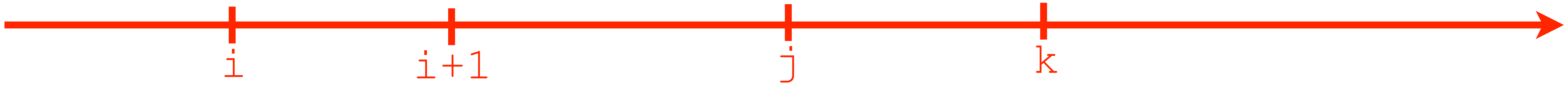


CRITICAL EDGES ARE REMOVED IN NEXT RESIDUAL GRAPH.

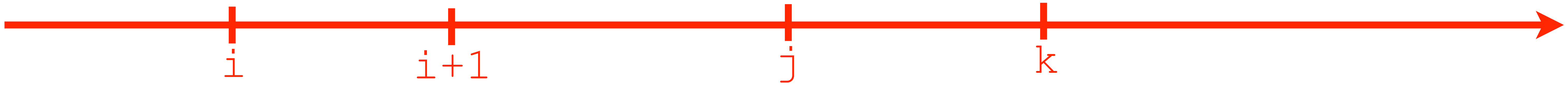


KEY IDEA: HOW MANY TIMES CAN AN EDGE BE CRITICAL?

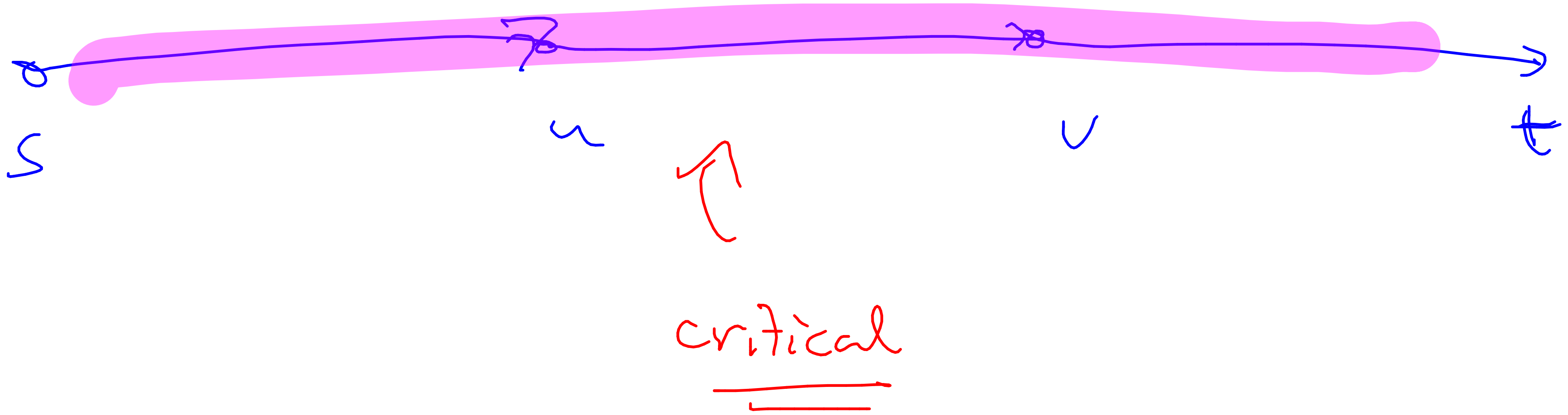
$\frac{V}{2}$ times

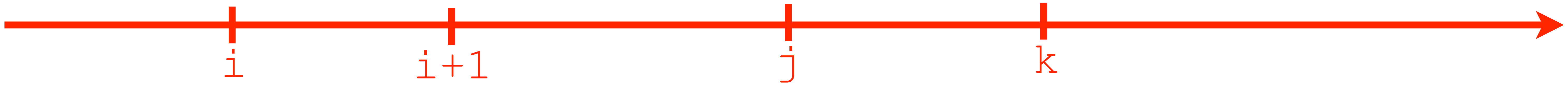


Outline of the argument



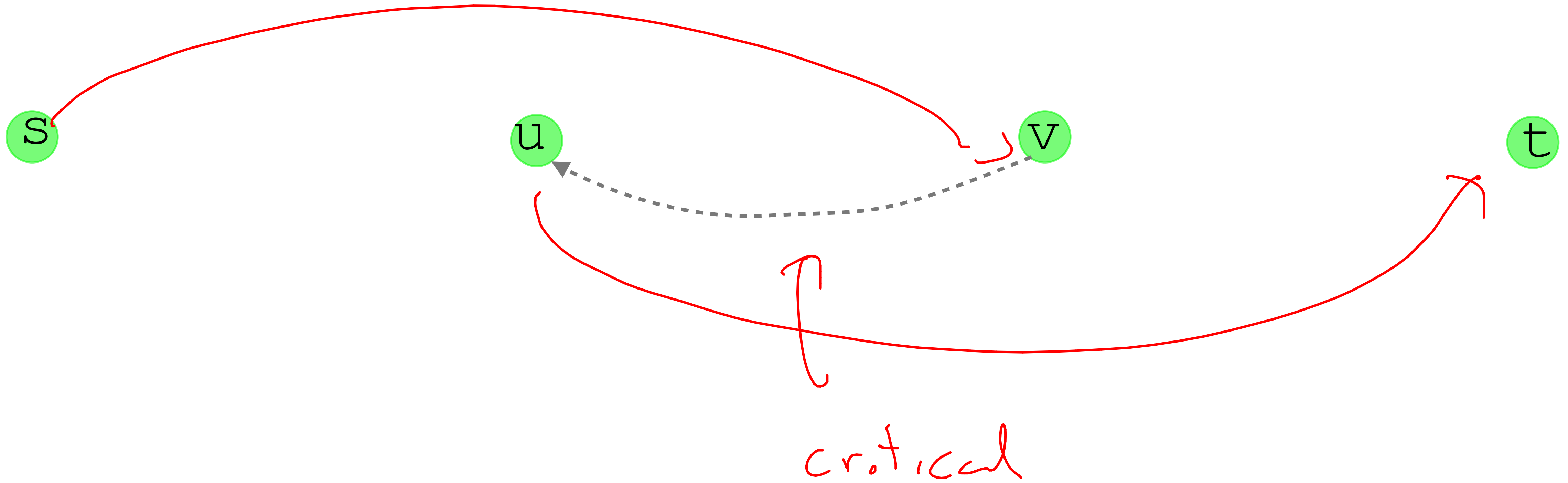
first time (u,v) is critical:

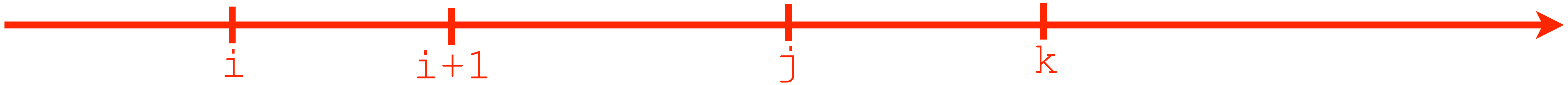




time i+1: (u,v) is critical: $$\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$$

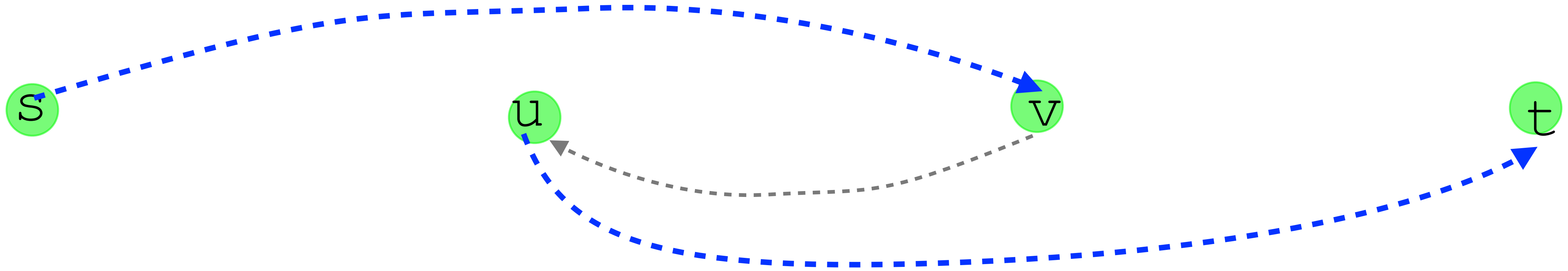
time j: Edge (u,v) STRIKES BACK



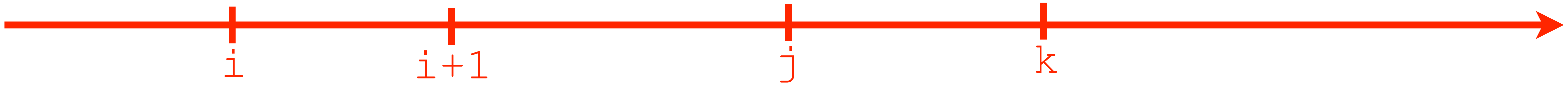


time i+1: (u,v) is critical: $\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$

time j: Edge (u,v) STRIKES BACK



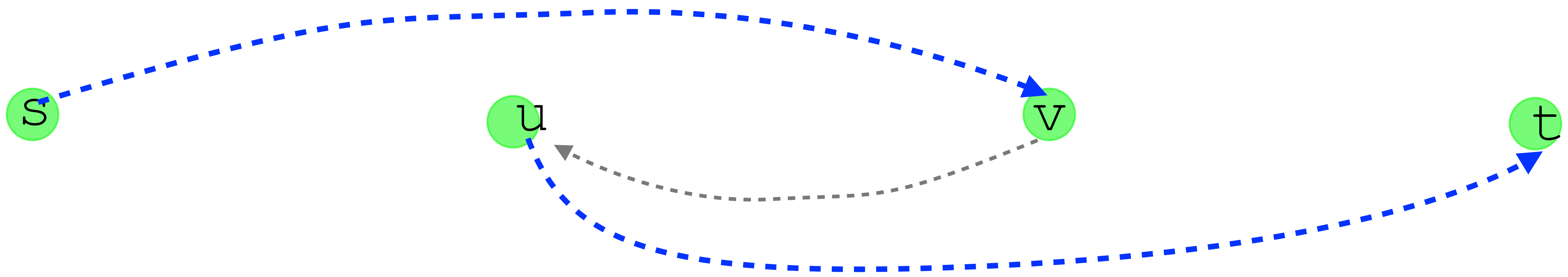
$$\delta_j(s,u) = \delta_j(s,v) + 1$$

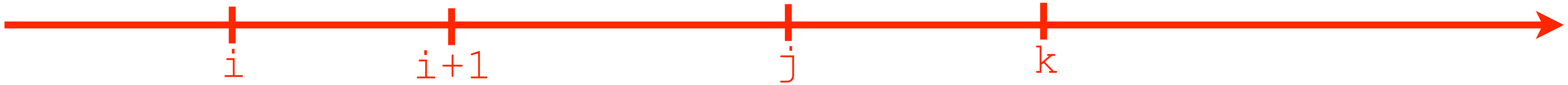


time j: Edge (u,v) STRIKES BACK

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

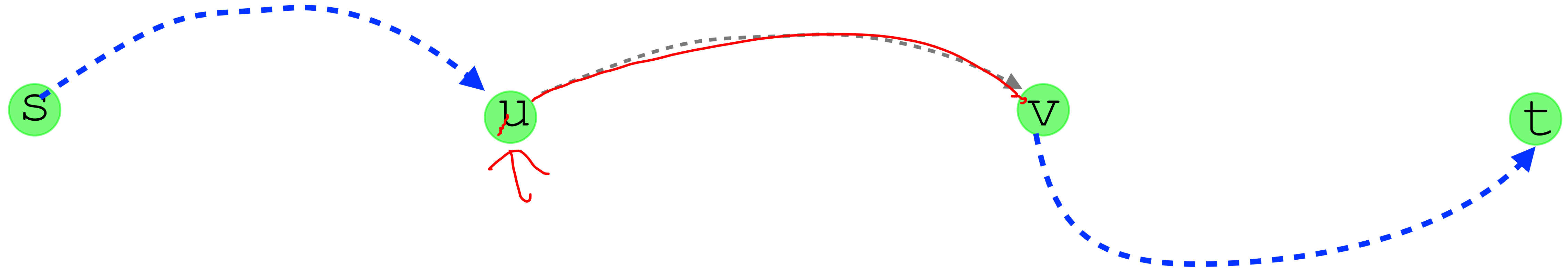
$$\delta_j(s,u) = \delta_j(s,v) + 1$$





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?

$E \geq V$

- edge critical only $\frac{V}{2}$ times.
- there are only E edges.

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

time to find an augmenting path: $\Theta(E+V)$ (BFS)

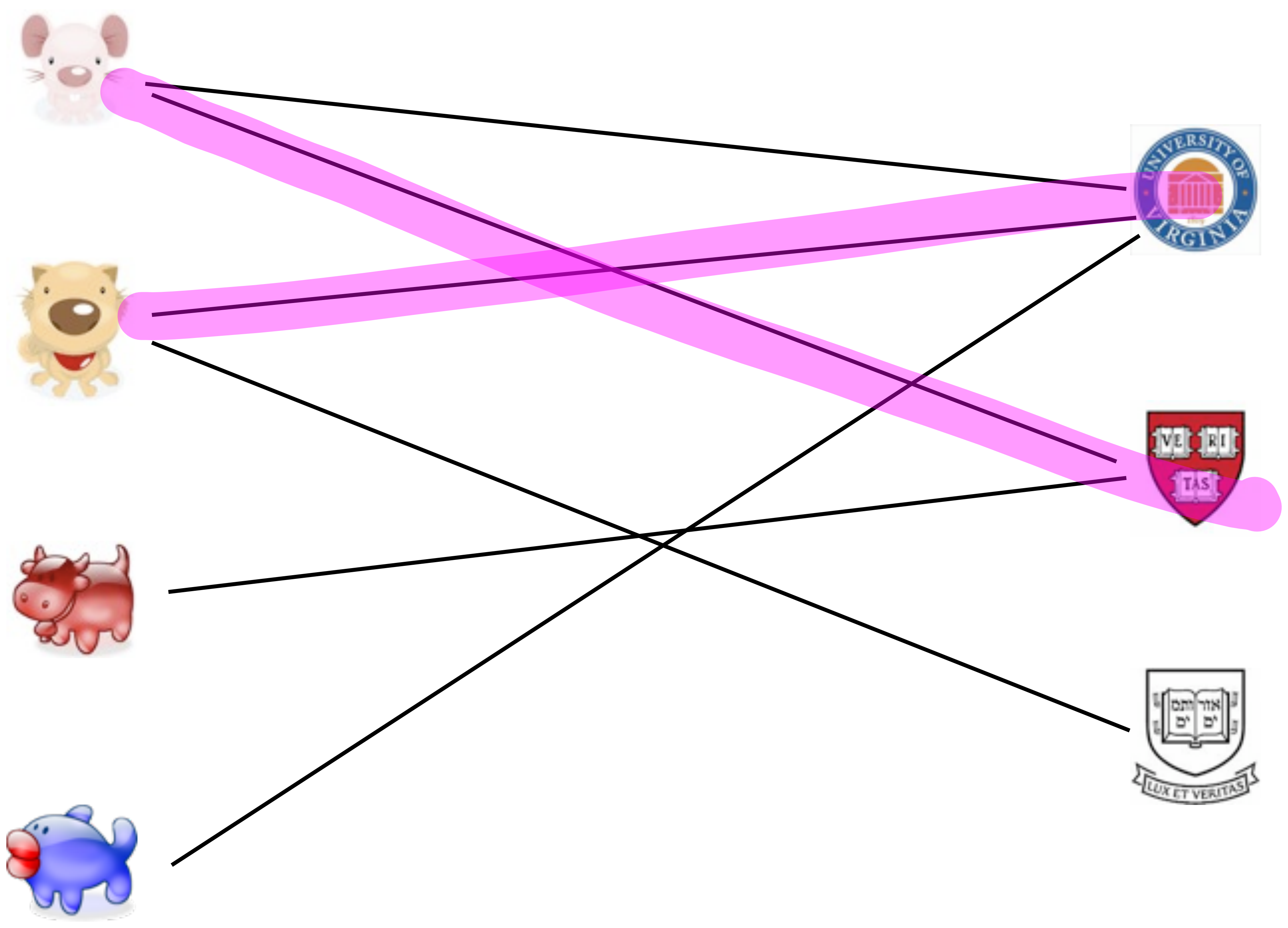
total running time of E-K algorithm:

$\Theta(E^2 V)$

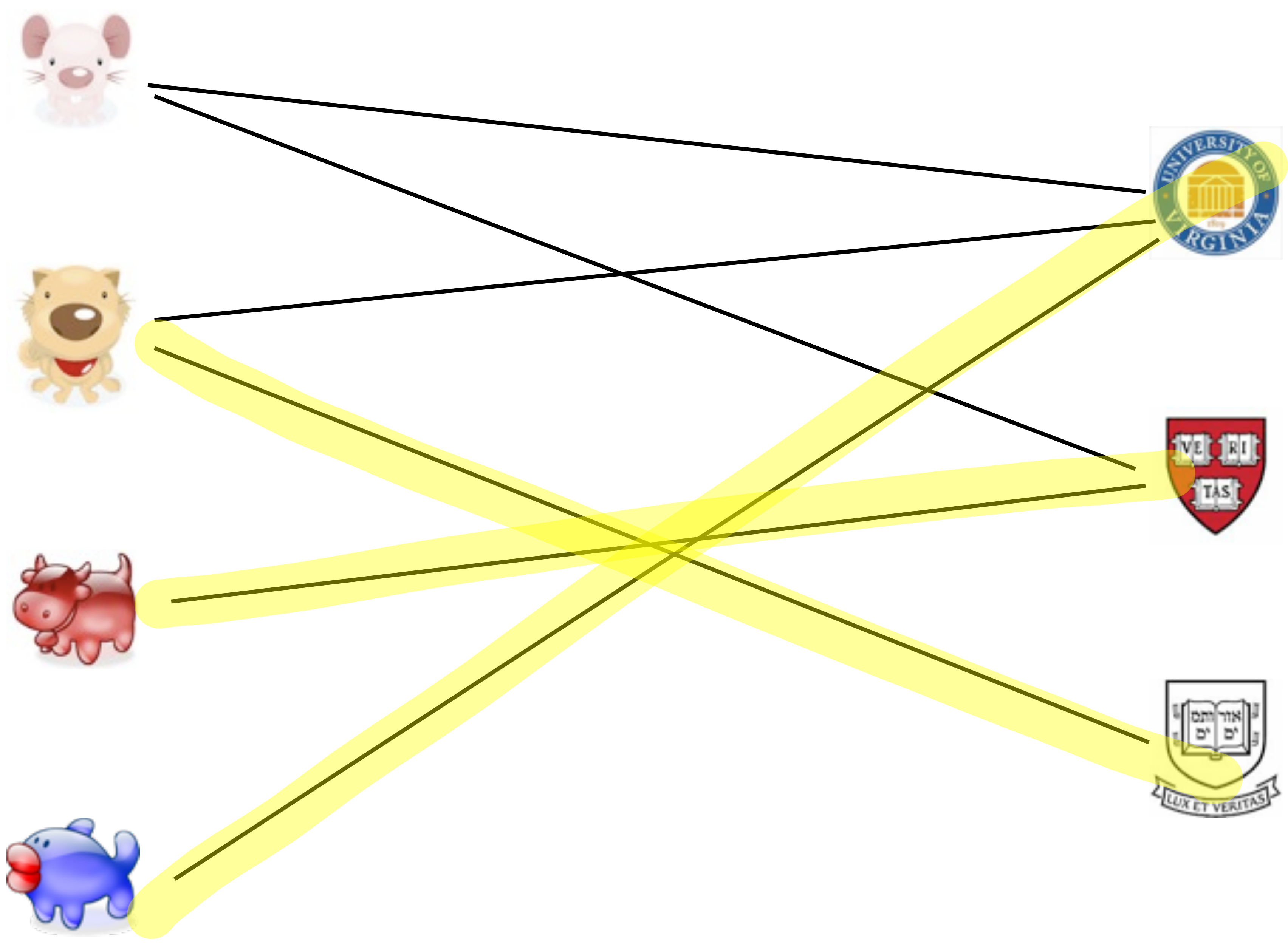
ff	$O(E\|f^*\|)$
ek2	$\Theta(E^2V)$
Tarjan push-relabel	$\Theta(EV^2)$
faster push-relabel	$\Theta(V^3)$

APPLICATIONS OF MAX FLOW

MAXIMUM BIPARTITE MATCHING



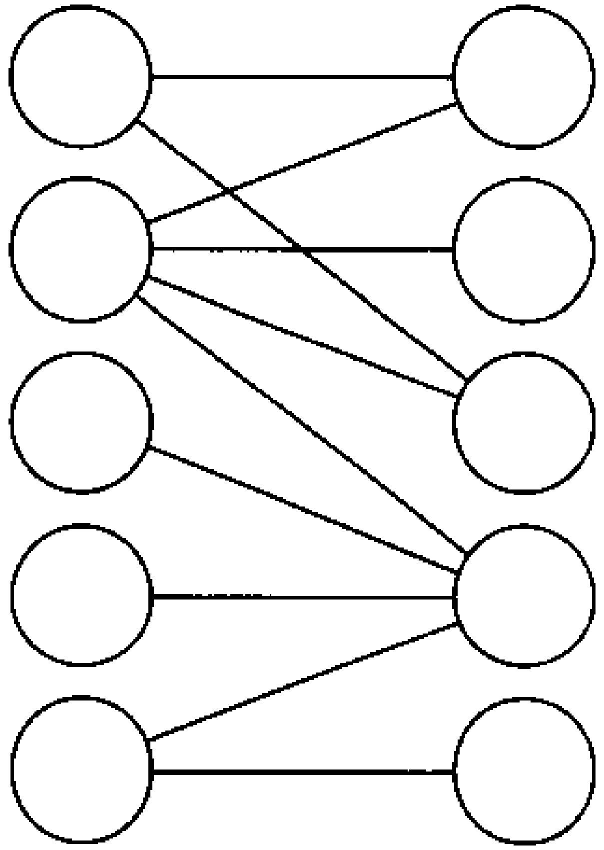
MAXIMUM BIPARTITE MATCHING



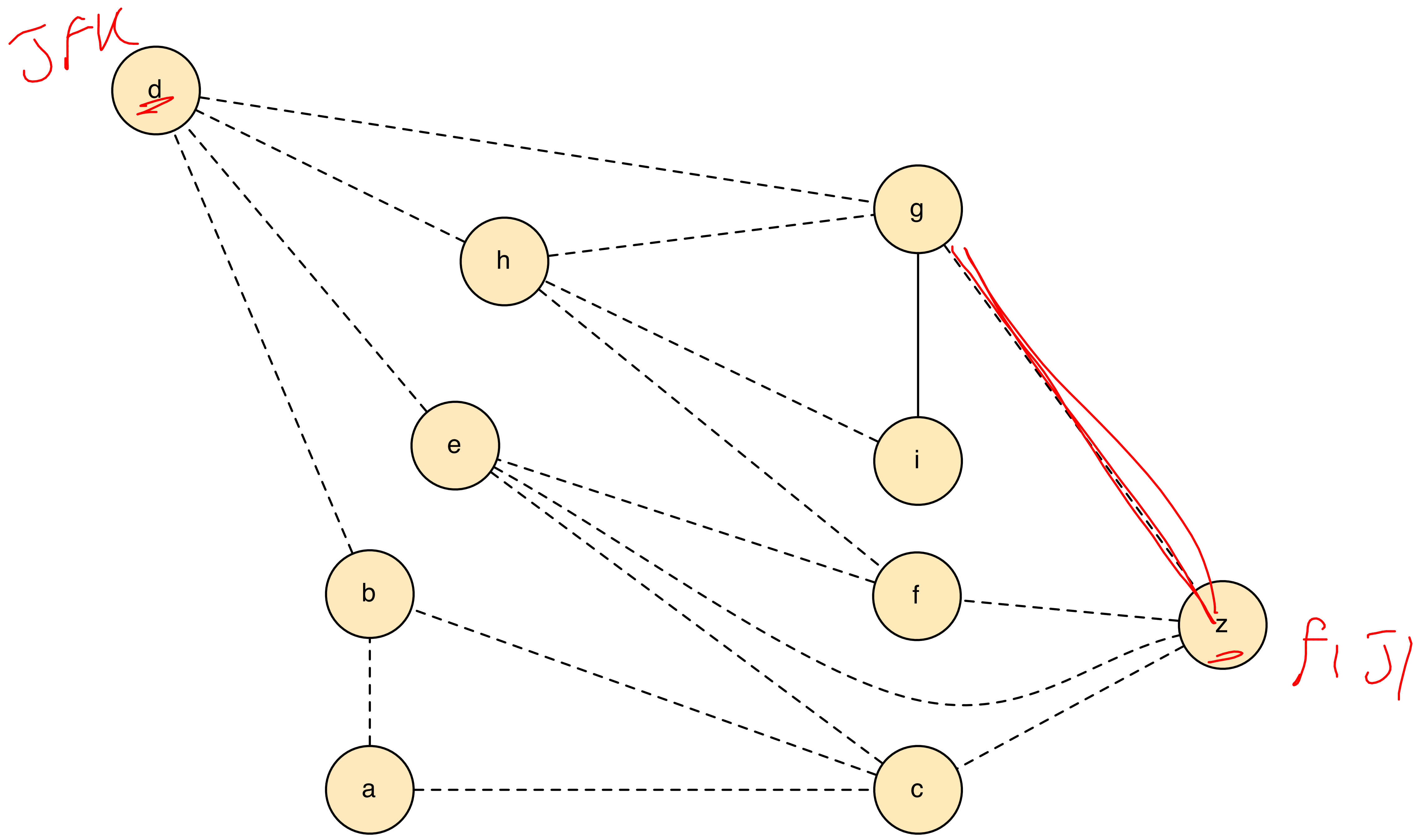
BIPARTITE MATCHING

PROBLEM:

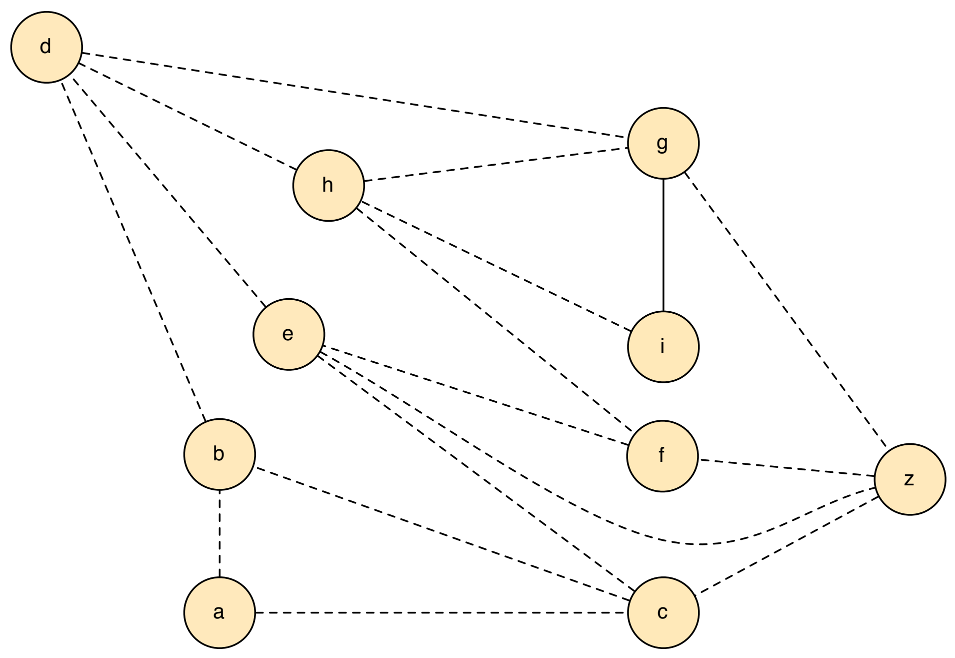
ALGORITHM



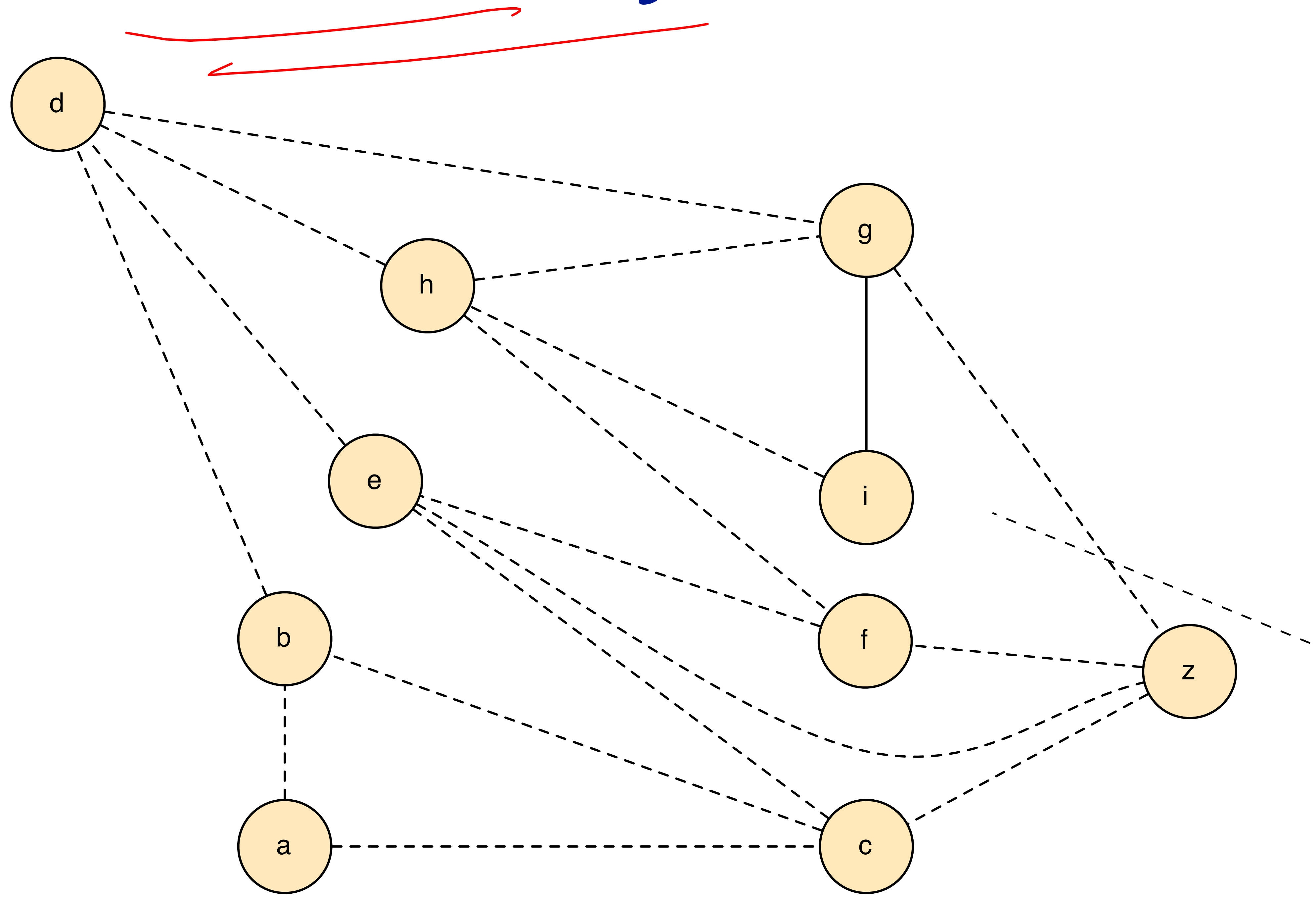
EDGE-DISJOINT PATHS



ALGORITHM



VERTEX-DISJOINT PATHS

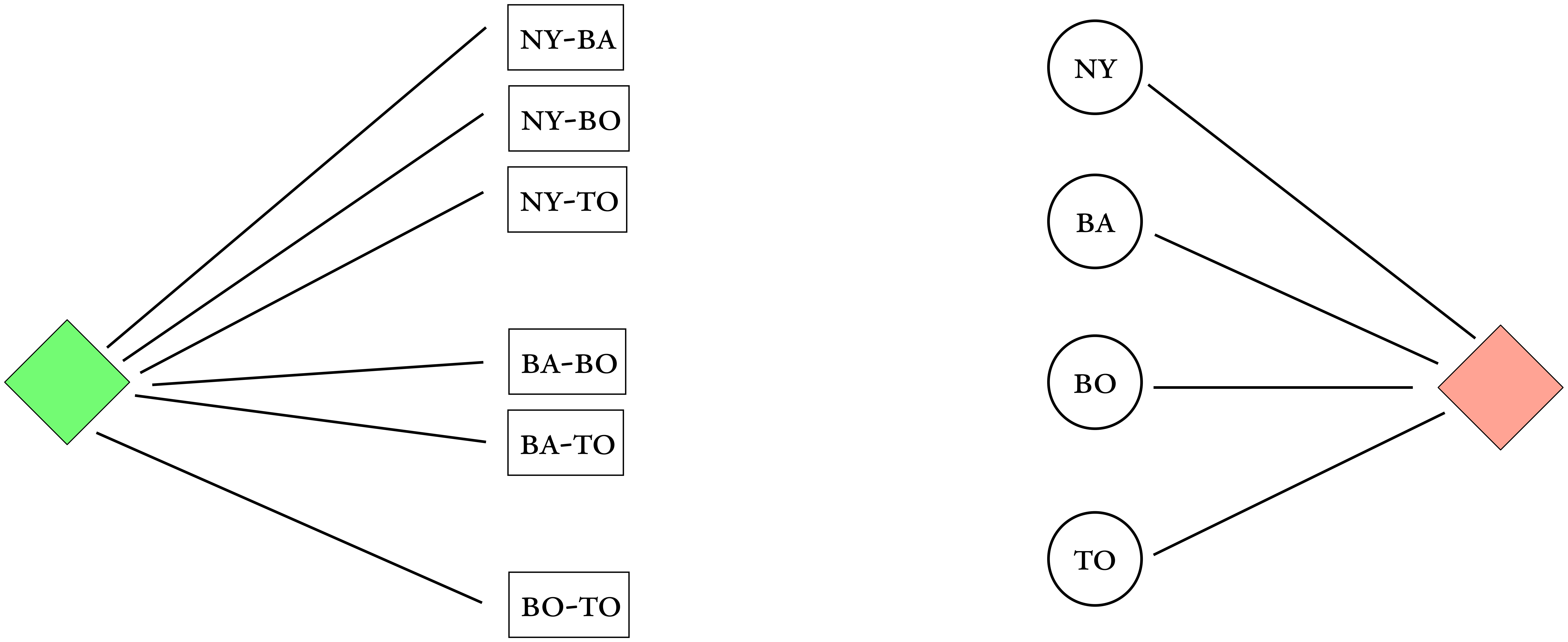


BASEBALL ELIMINATION

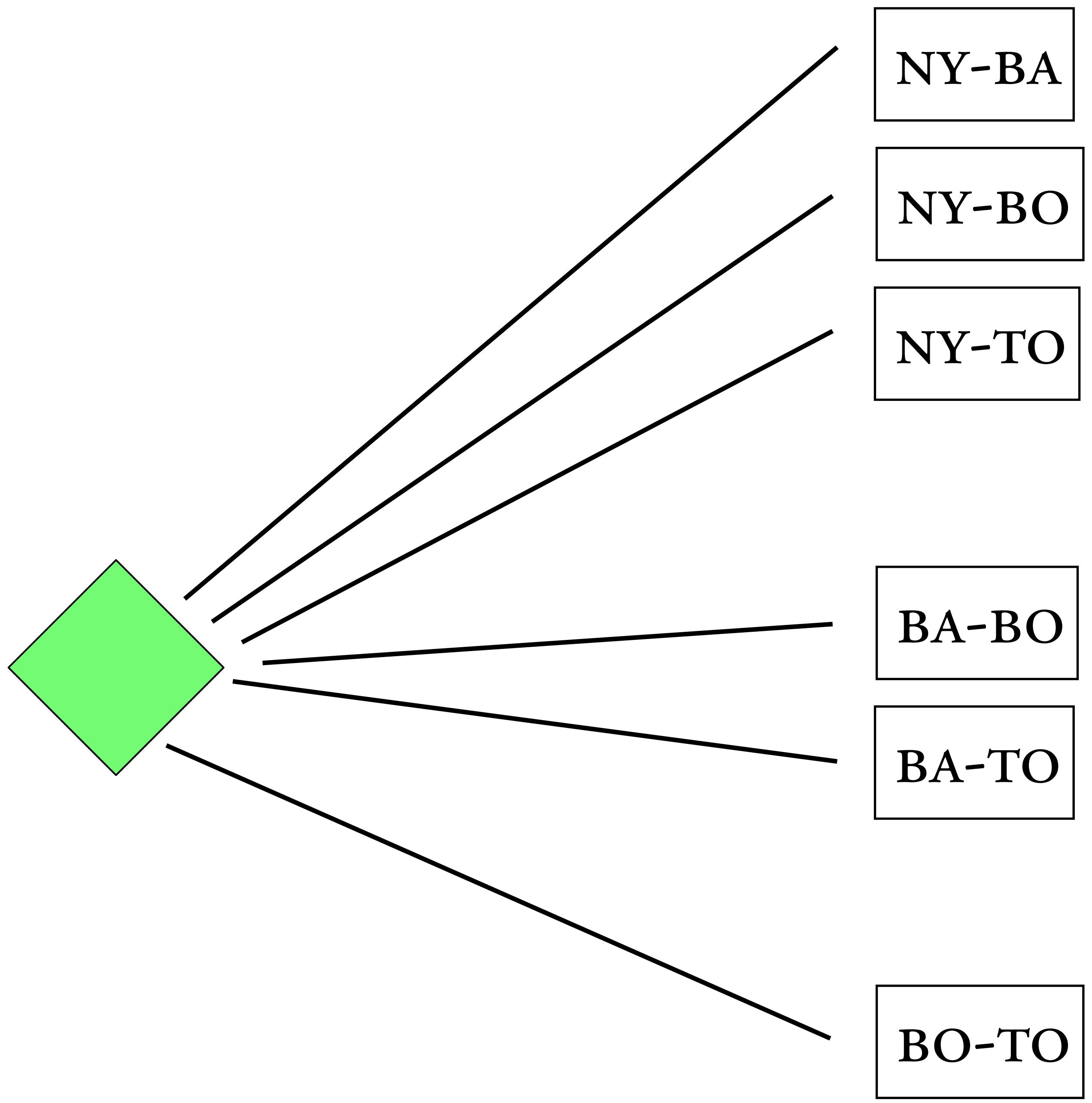
	W	L	Left	Against: A	P	N	M
ATL	83	71	8	-	1	6	1
PHL	80	79	3	1	-	0	2
NY	78	78	6	6	0	-	0
MONT	77	82	3	1	2	0	-

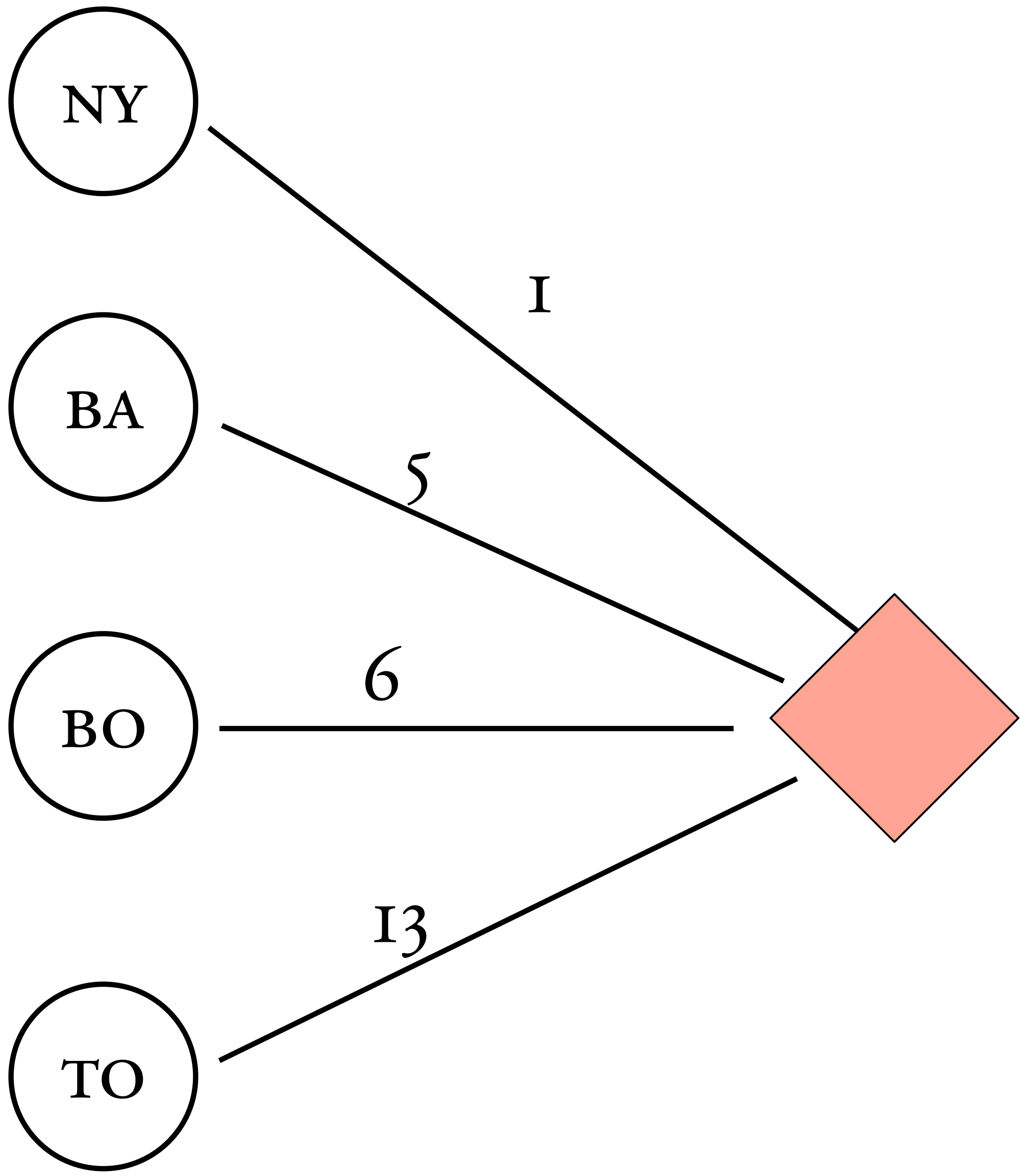
BASEBALL ELIMINATION

	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			

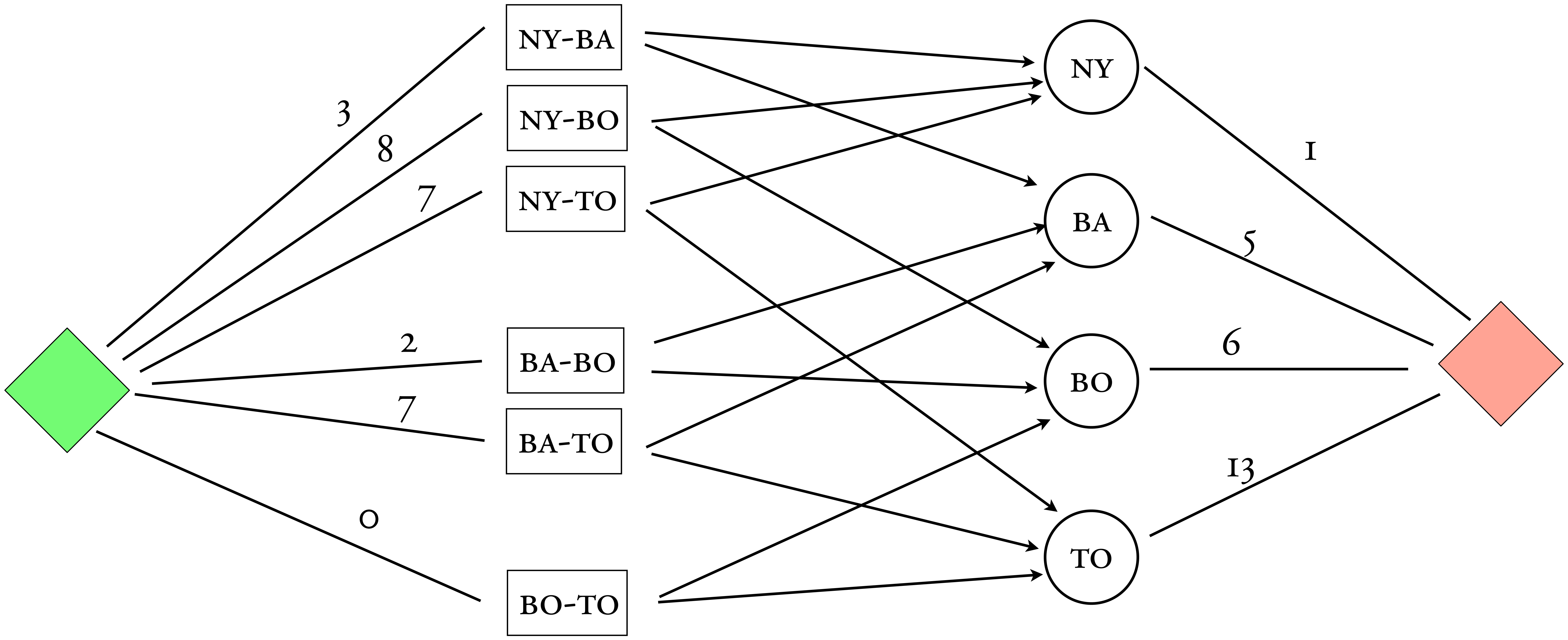


	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			





	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			

ALGORITHMS FOR MAX FLOW

CUTS

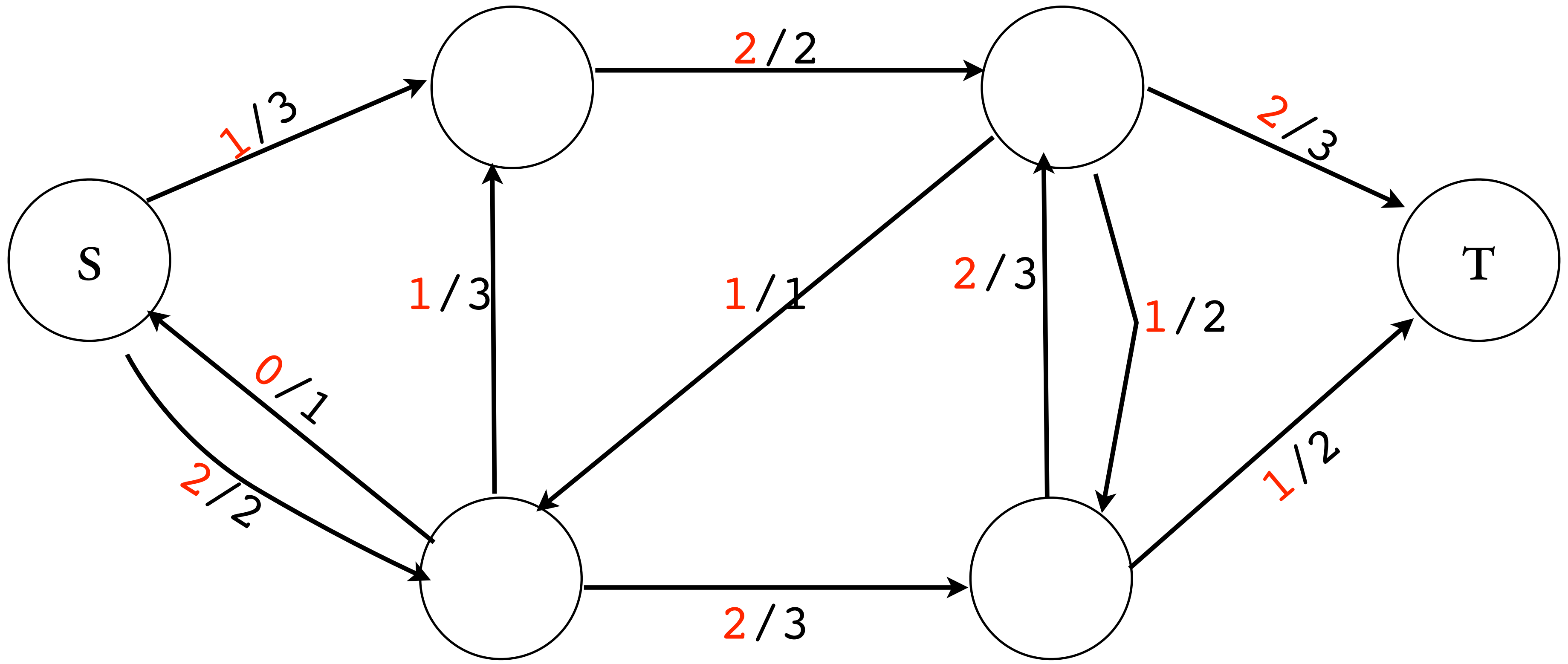
DEF OF A CUT:

COST OF A CUT:

$$||S, T|| =$$

LEMMA: [MIN CUT] FOR ANY $f, (S, T)$

FOR ANY $f, (S,T)$ IT HOLDS THAT $|f| \leq ||S,T||$



EXAMPLE: