L20

4102

4.05.2016

abhi shelat

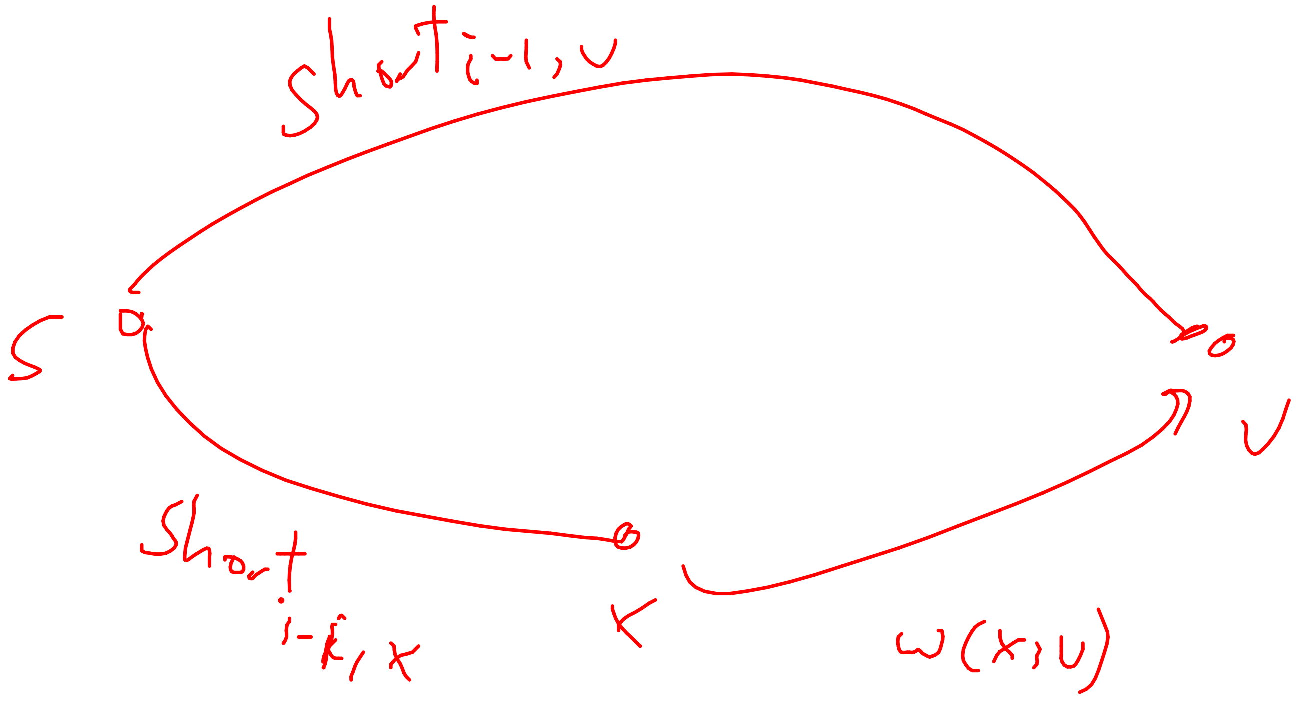
What about Negative edge weights?

sssp(G,s)

$\underline{\text{SHORT}_{i,v}}$ = length of the shortest path from s to v that uses ≤ i edges.

sssp(G,s)

$$\text{SHORT}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} & \left\{ \begin{array}{l} \text{SHORT}_{i-1,v} \\ \text{SHORT}_{i-1,x} + w(x,v) \end{array} \right\} \end{cases}$$



BELLMAN-FORD(G, s)

$\text{SHORT}_{0,s} \leftarrow 0$

$\forall v \in V - \{s\}$, $\text{SHORT}_{0,v} \leftarrow \infty$

for $i = 1, \ldots, V - 1$

  do for each $v \in V - \{s\}$

    do $\text{SHORT}_{i,v} = \min_{x \in Adj(v)} \begin{Bmatrix} \text{SHORT}_{i-1,v} \\ w(x,v) + \text{SHORT}_{i-1,x} \end{Bmatrix}$

loop over edges instead.

BELLMAN-FORD(G, s)

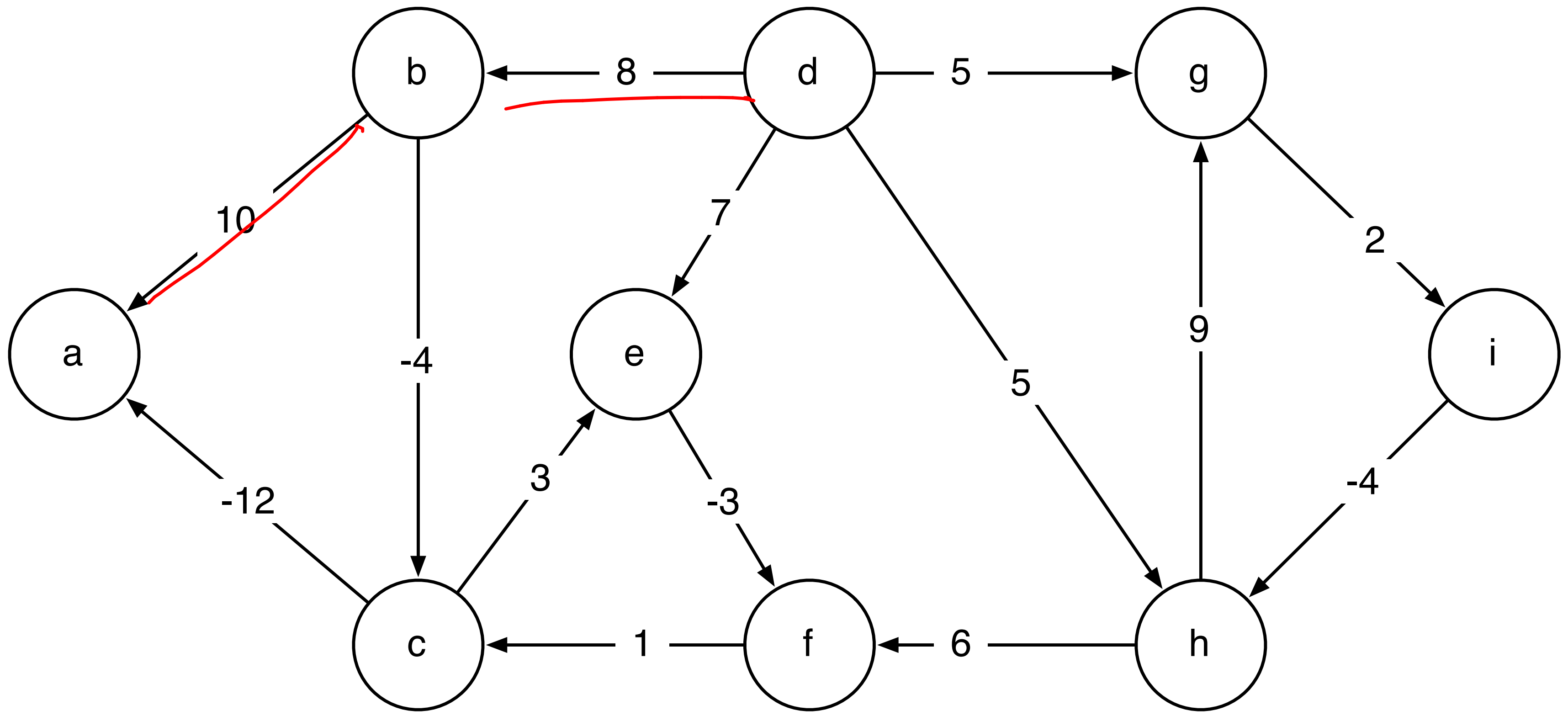
$\textsc{short}_{0,s} \leftarrow 0$

$\forall v \in V - \{s\}$, $\textsc{short}_{0,v} \leftarrow \infty$

for $i = 1, \ldots, V - 1$

  do for each $e = (x, y) \in E$

    do $\textsc{short}_{i,y} = \min \left\{ \begin{array}{l} \textsc{short}_{i-1,y} \\ \textsc{short}_{i,y} \\ w(x, y) + \textsc{short}_{i-1,x} \end{array} \right\}$



(G, s)

0

$\}, \text{SHORT}_{0,v} \leftarrow \infty$

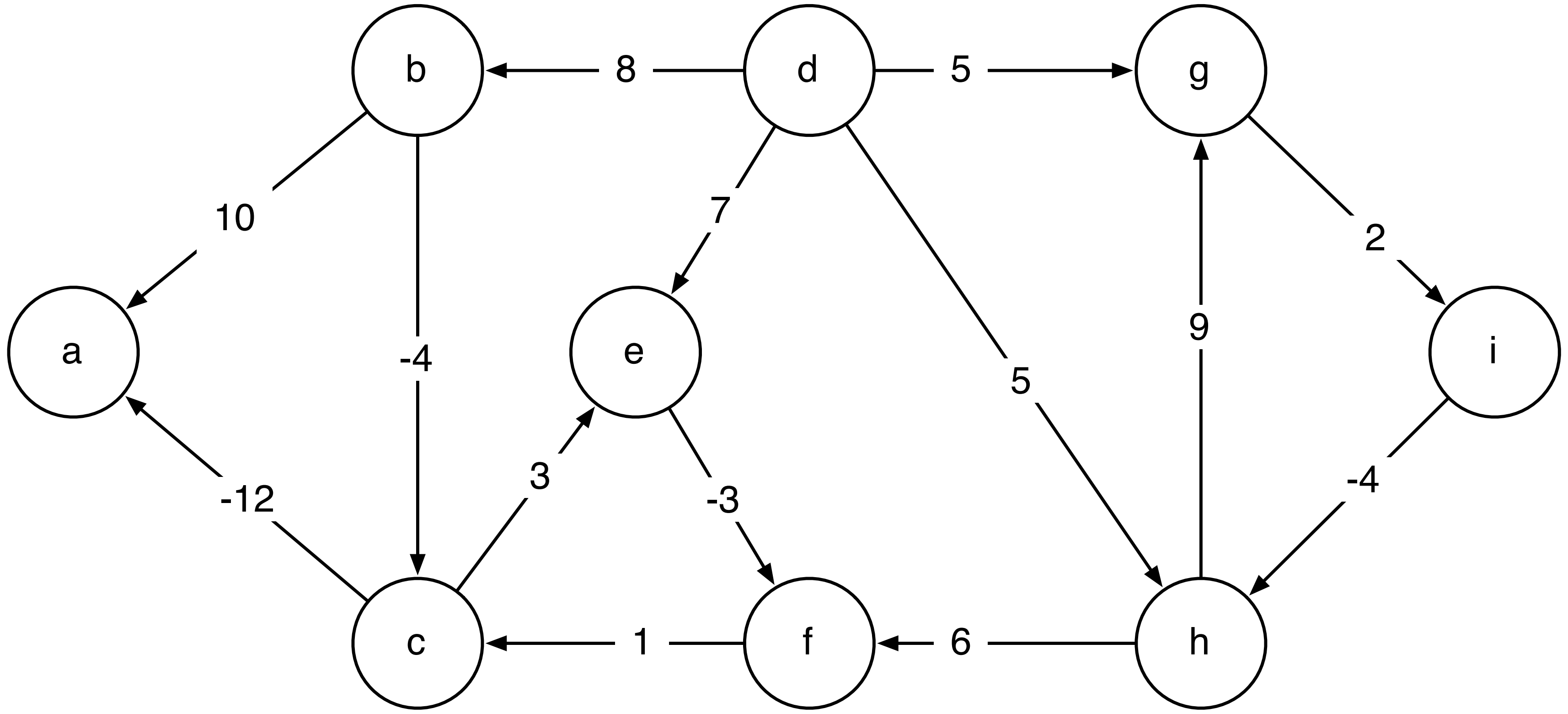
$, V - 1$

each $e = (x, y) \in E$

do $\text{SHORT}_{i,y} = \min \left\{ \begin{array}{l} \text{SHORT}_{i-1,y} \\ \text{SHORT}_{i,y} \\ w(x,y) + \text{SHORT}_{i-1,x} \end{array} \right\}$

bf(G,d)

	0	1	2	3	4	5	6	7
a	∞							
b	∞							
c	∞							
d	0							
e	∞							
f	∞							
g	∞							
h	∞							
i	∞							



(G, s)

0

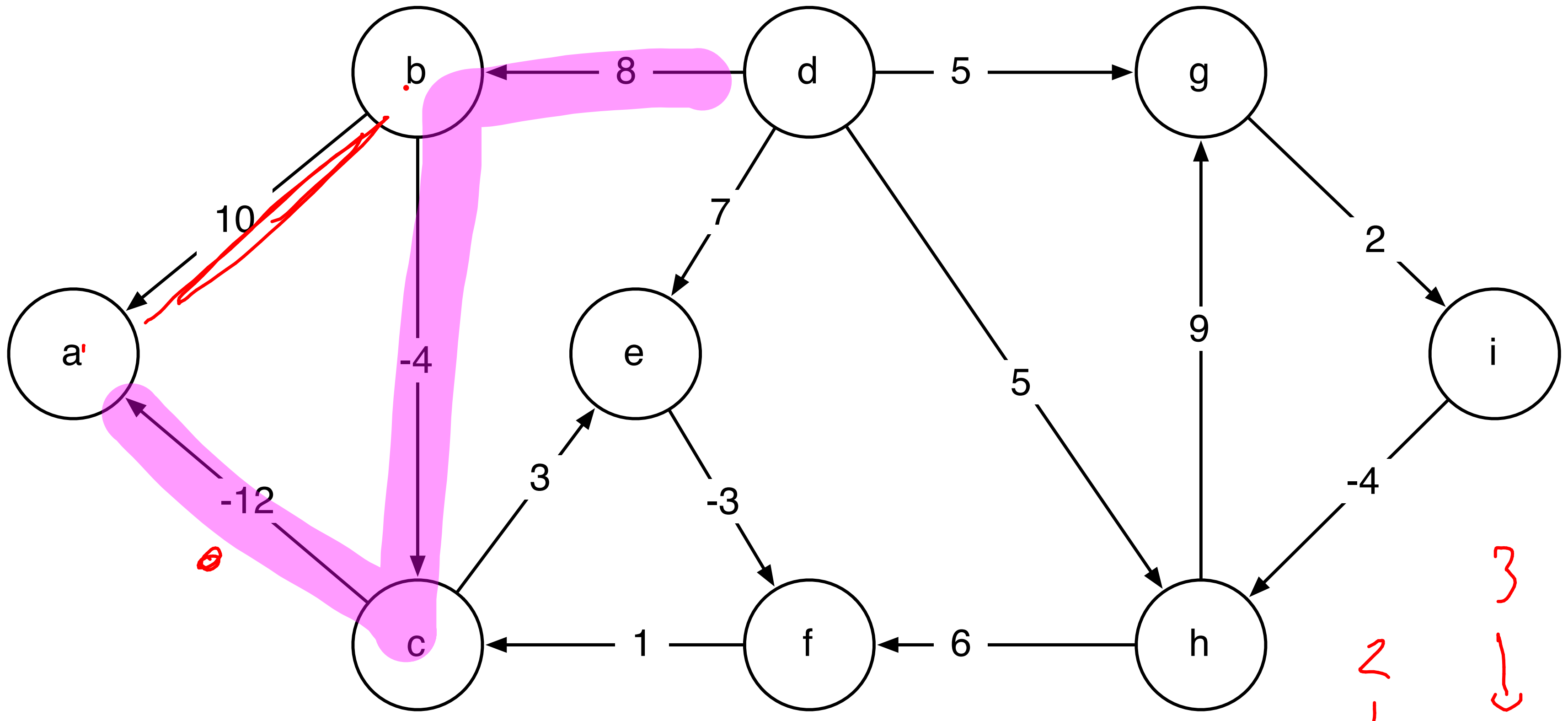
$\}$, $\text{SHORT}_{0,v} \leftarrow \infty$

$, V - 1$

each $e = (x, y) \in E$

$$\textbf{do } \text{SHORT}_{i,y} = \min \left\{ \begin{array}{l} \text{SHORT}_{i-1,y} \\ \text{SHORT}_{i,y} \\ w(x,y) + \text{SHORT}_{i-1,x} \end{array} \right\}$$

	8						
0	0						
	7						
	5						
	5						



(G, s)

0

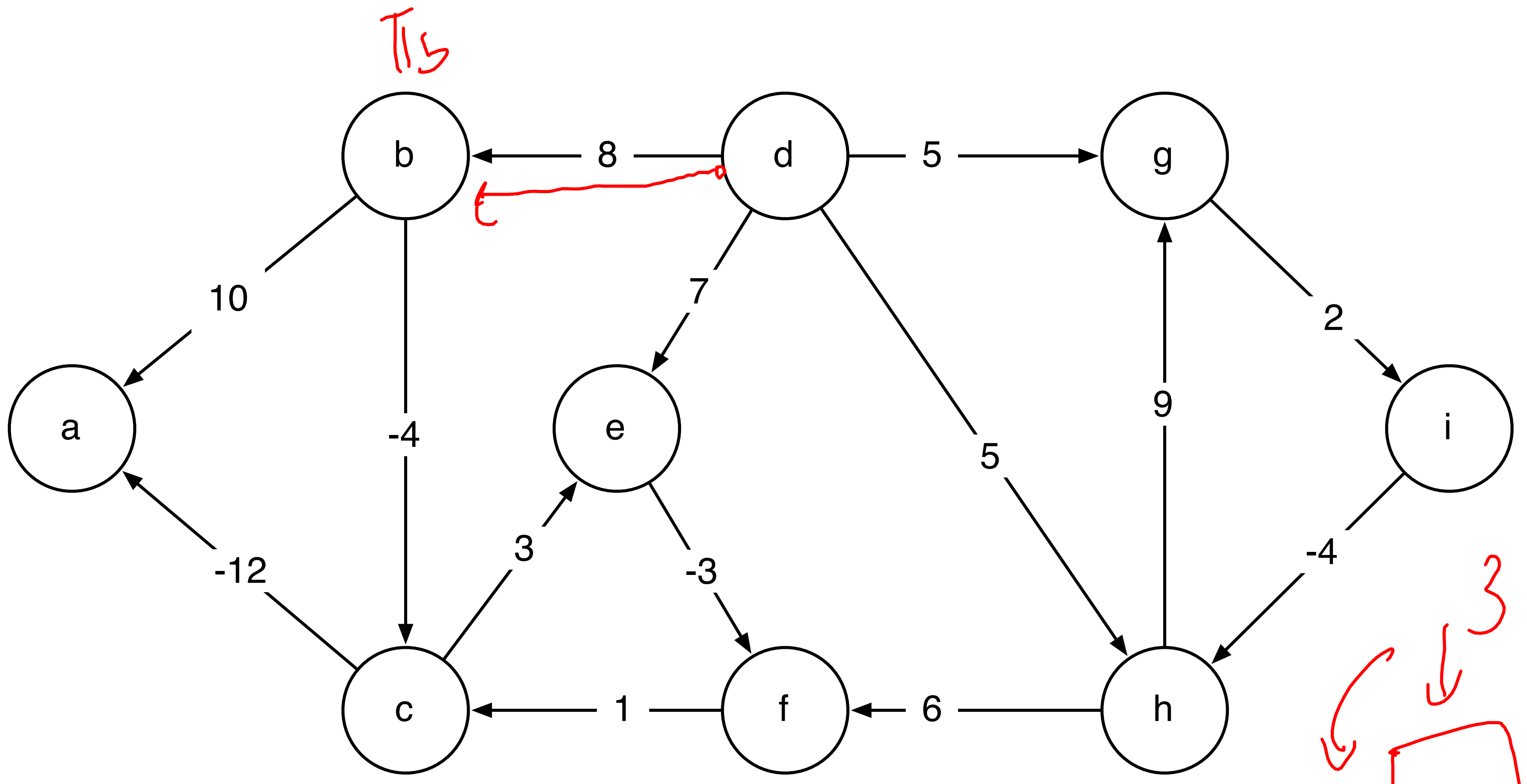
$\}$, $\text{SHORT}_{0,v} \leftarrow \infty$

$, V - 1$

each $e = (x, y) \in E$

$$\textbf{do } \text{SHORT}_{i,y} = \min \left\{ \begin{array}{l} \text{SHORT}_{i-1,y} \\ \text{SHORT}_{i,y} \\ w(x,y) + \text{SHORT}_{i-1,x} \end{array} \right\}$$

			2	3				
a			18	8				
b		8	8					
c			4					
d	0	0	0					
e		7	7					
f			4					
g		5	5					
h		5	5					
			7					



(G, s)

0

$\}$, $\text{SHORT}_{0,v} \leftarrow \infty$

$, V - 1$

each $e = (x, y) \in E$

$$\textbf{do } \text{SHORT}_{i,y} = \min \left\{ \begin{array}{l} \text{SHORT}_{i-1,y} \\ \text{SHORT}_{i,y} \\ w(x,y) + \text{SHORT}_{i-1,x} \end{array} \right\}$$

O(V²) space

		18	-8				
	8	8	8				
		4	4				
0	0	0	0				
	7	7	7				
		4	4				
	5	5	5				
	5	5	3				
		7	7				

0 1 2 3

V−1 V

same, or smaller

Optimization to save Space

BELLMAN-FORD(G, s)

$\text{SHORT}_{0,s} \leftarrow 0$

$\forall v \in V - \{s\}$, $\text{SHORT}_{0,v} \leftarrow \infty$

for $i = 1, \ldots, V-1$

  do for each $e = (x,y) \in E$

    do $\text{SHORT}_{i,y} = \min \begin{cases} \text{SHORT}_{i-1,y} \\ \text{SHORT}_{i,y} \\ w(x,y) + \text{SHORT}_{i-1,x} \end{cases}$

$O(V^2)$ space

BELLMAN-FORD(G, s)

$d_s \leftarrow 0$

$\forall v \in V - \{s\}$, $d_v \leftarrow \infty$

for $i = 1, \ldots, V-1$

  do for each $e = (x,y) \in E$

    do $d_y \leftarrow \min\{\, d_y, w(x,y) + d_x \,\}$

$\Theta(V)$ space

running time

```
BELLMAN-FORD(G, s)
  d_s ← 0
  ∀v ∈ V − {s}, d_v ← ∞
  for i = 1, . . . , V − 1
        do for each e = (x, y) ∈ E
              do d_y ← min{ d_y, w(x, y) + d_x }
```

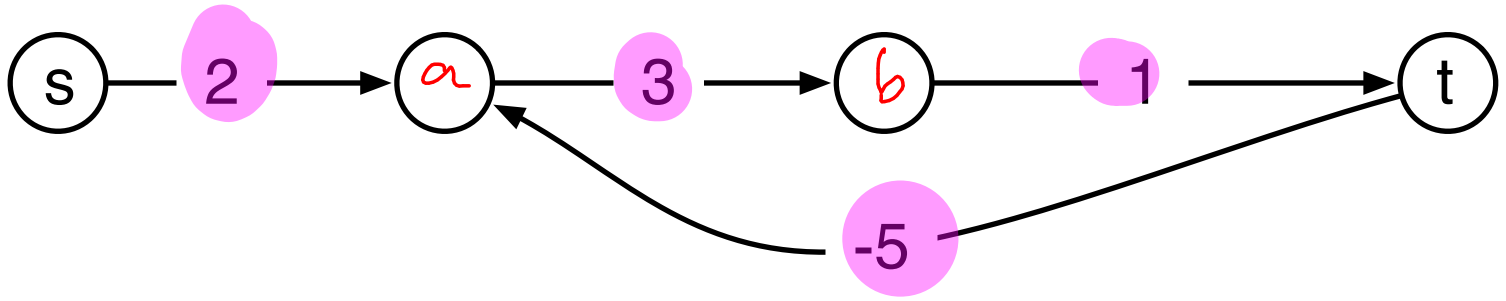
$\Theta(VE)$ time

$\Theta(V)$ space.

negative cycles?

$|V| = 4$

$V - 1 = 3$



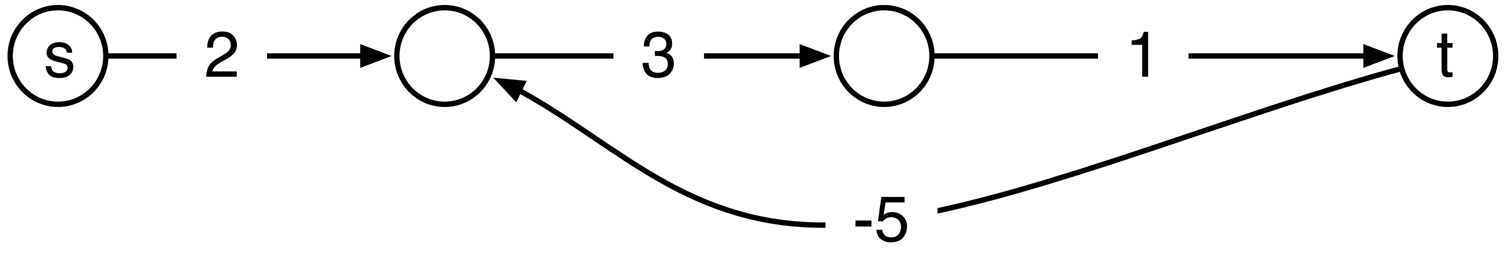
	1	2	↓	↓
s	0	0	0	0
a	2	2	2	2 → 1
b	∞	5	5	5
t	∞	∞	6	6

← Decrease on the V^{th} step

$\Rightarrow \exists$ a negative cycle.

time 3

negative cycles?



s	0	0	0	0
a	2	2	2	1
b		5	5	5
t			6	6

applications of BF

$\Theta(VE) > \Theta(E \log V)$

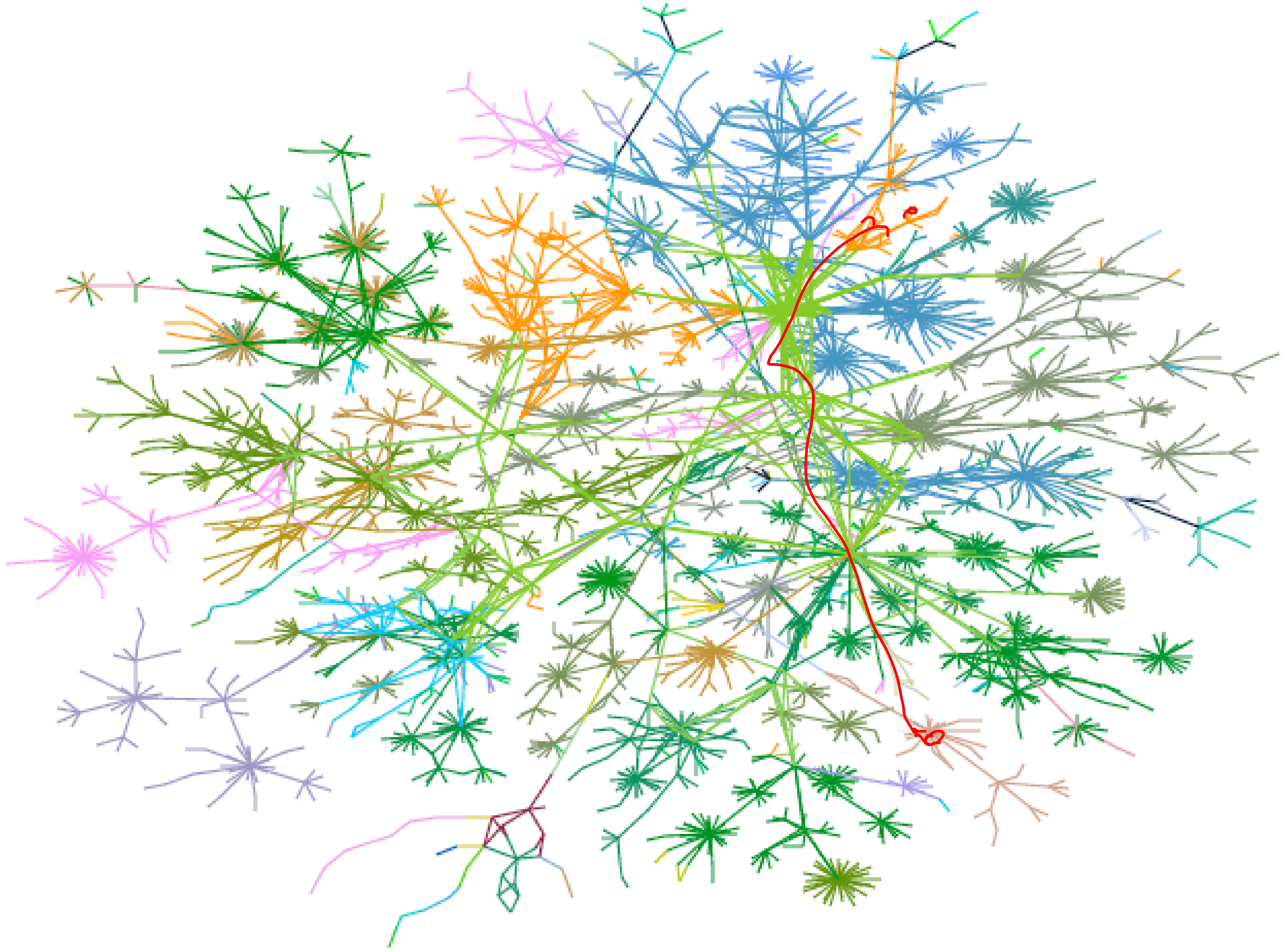
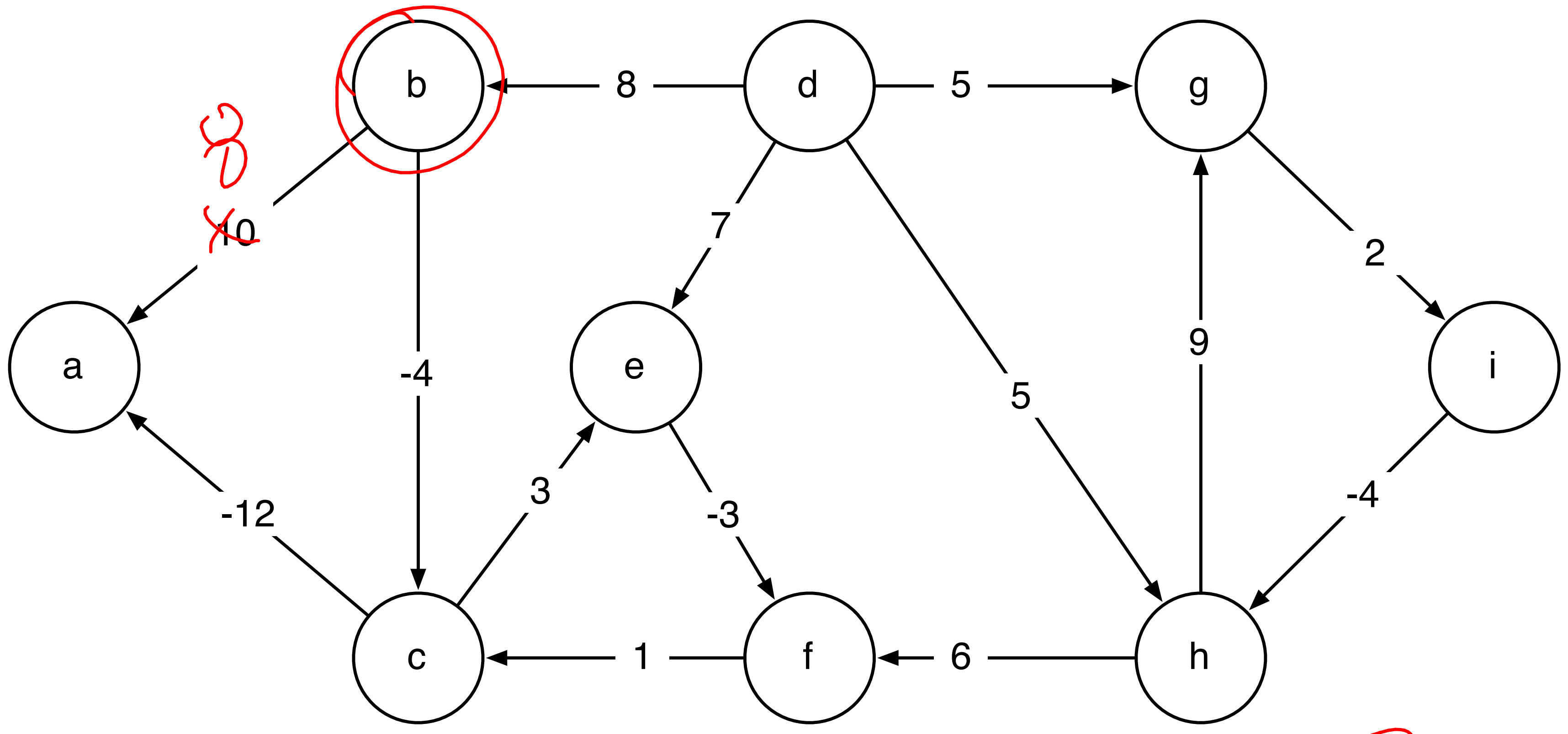


image: cheswick et al

Figure 3: Lucent's intranet as of 1 October 1999.



what happens when
B changes...

		18					
	8	8					
		4					
0	0	0					
	7	7					
		4					
	5	5					
	5	5					
		7					

distance vector

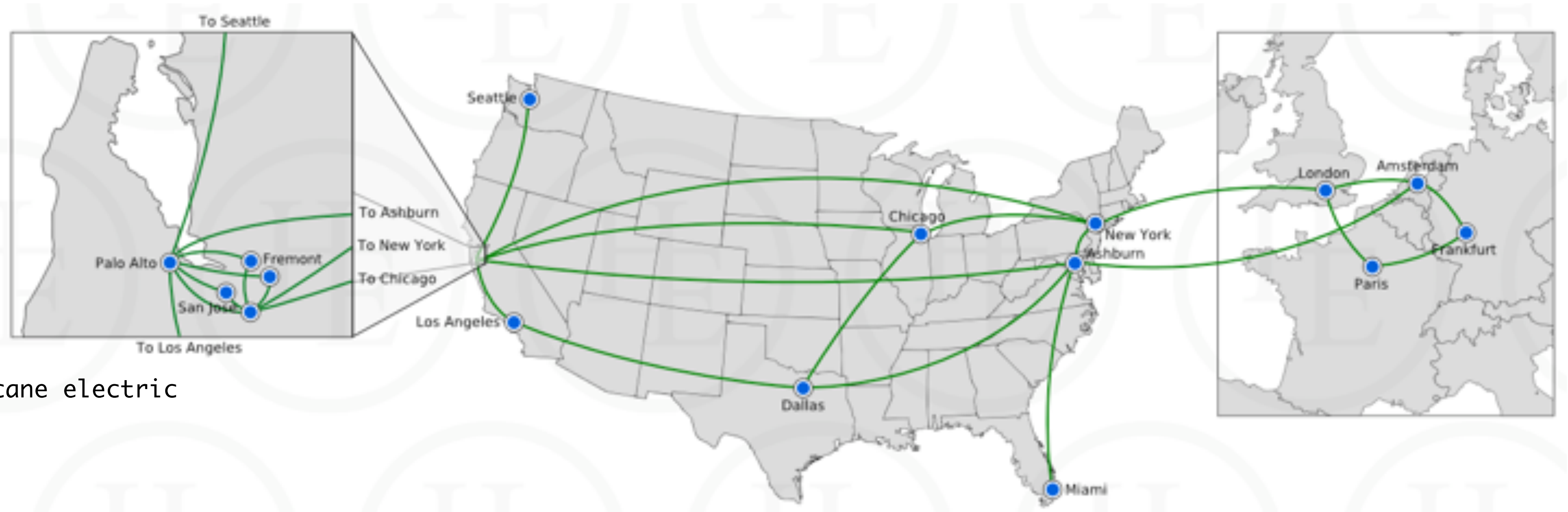
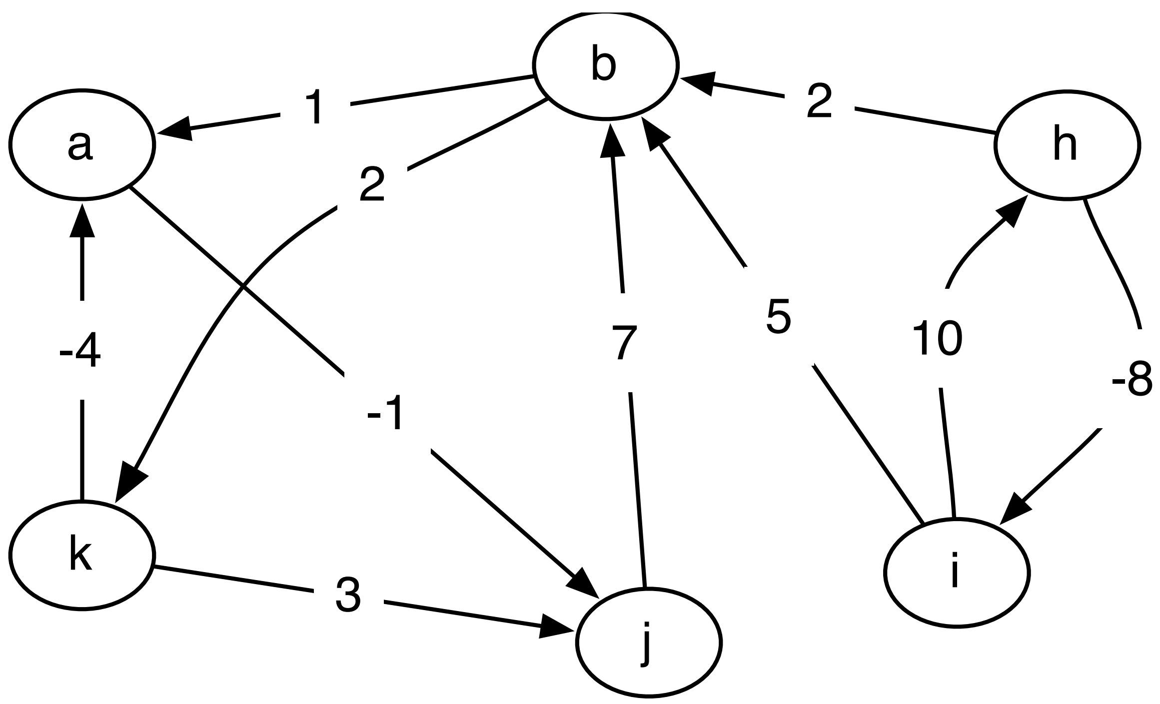


image: hurricane electric

All-pairs shortest path



BF: $\Theta(VE)$

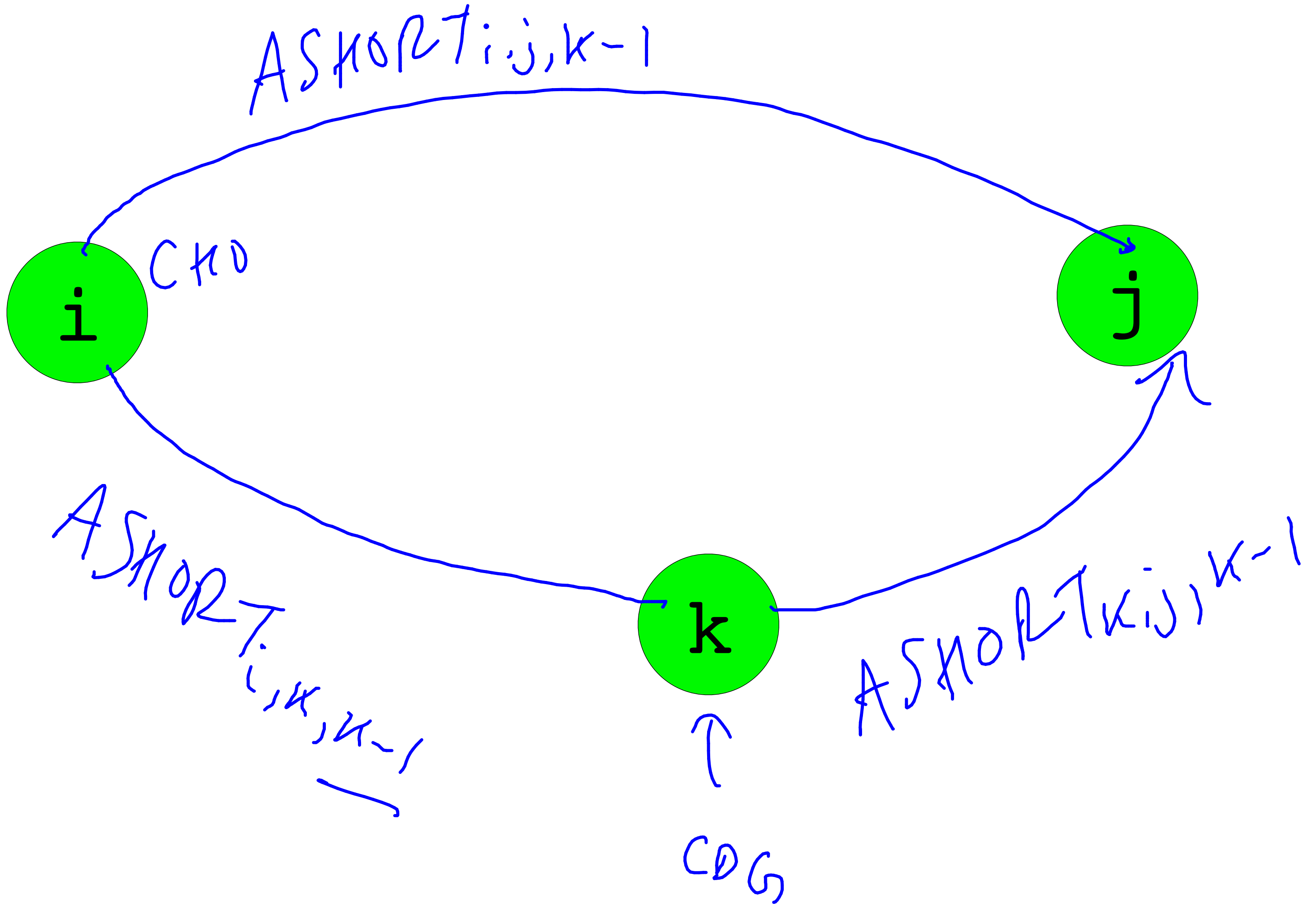
All$_{pairs}$: $V \cdot \Theta(V \cdot E) = \Theta(V^2 E)$

$\text{ASHORT}_{i,j,k} =$ Length of the shortest path from i to j that uses only nodes 1 ... to k.

$ASHORT_{i,j,k} =$

$i, j, k \in [1 \ldots n]$

$G = (V, E)$



$$\mathrm{ASHORT}_{i,j,k} =$$

$$\text{ASHORT}_{i,j,k} = \begin{cases} w_{i,j} & k = 0 \\ \min \begin{cases} \text{ASHORT}_{i,j,k-1} \\ \text{ASHORT}_{i,k,k-1} + \text{ASHORT}_{k,j,k-1} \end{cases} & k \geq 1 \end{cases}$$

Floyd-Warshall(G,W)

→ INIT w/edge weights

for (k = 1 … n)

for (i = 1 … n)

for (j = 1 … n)

$$ASHORT_{i,j} = \min \begin{cases} ASHORT_{i,j} \\ ASHORT_{i,k} + ASHORT_{k,j} \end{cases}$$

```
int graph[128][128], n; // a weighted graph and its size
  void floydWarshall() {
     for( int k = 0; k < n; k++ )
       for( int i = 0; i < n; i++ )
         for( int j = 0; j < n; j++ )
            graph[i][j] = min( graph[i][j], graph[i][k] + graph[k][j] );
  }
  int main {
     // initialize the graph[][] adjacency matrix and n
     // graph[i][i] should be zero for all i.
     // graph[i][j] should be "infinity" if edge (i, j) does not exist
     // otherwise, graph[i][j] is the weight of the edge (i, j)
     floydWarshall();
     // now graph[i][j] is the length of the shortest path from i to j
  }
```

$\Theta(V^3)$ time

$\Theta(V^2)$ space.

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.”

MAX FLOW,

MIN CUT

problem.

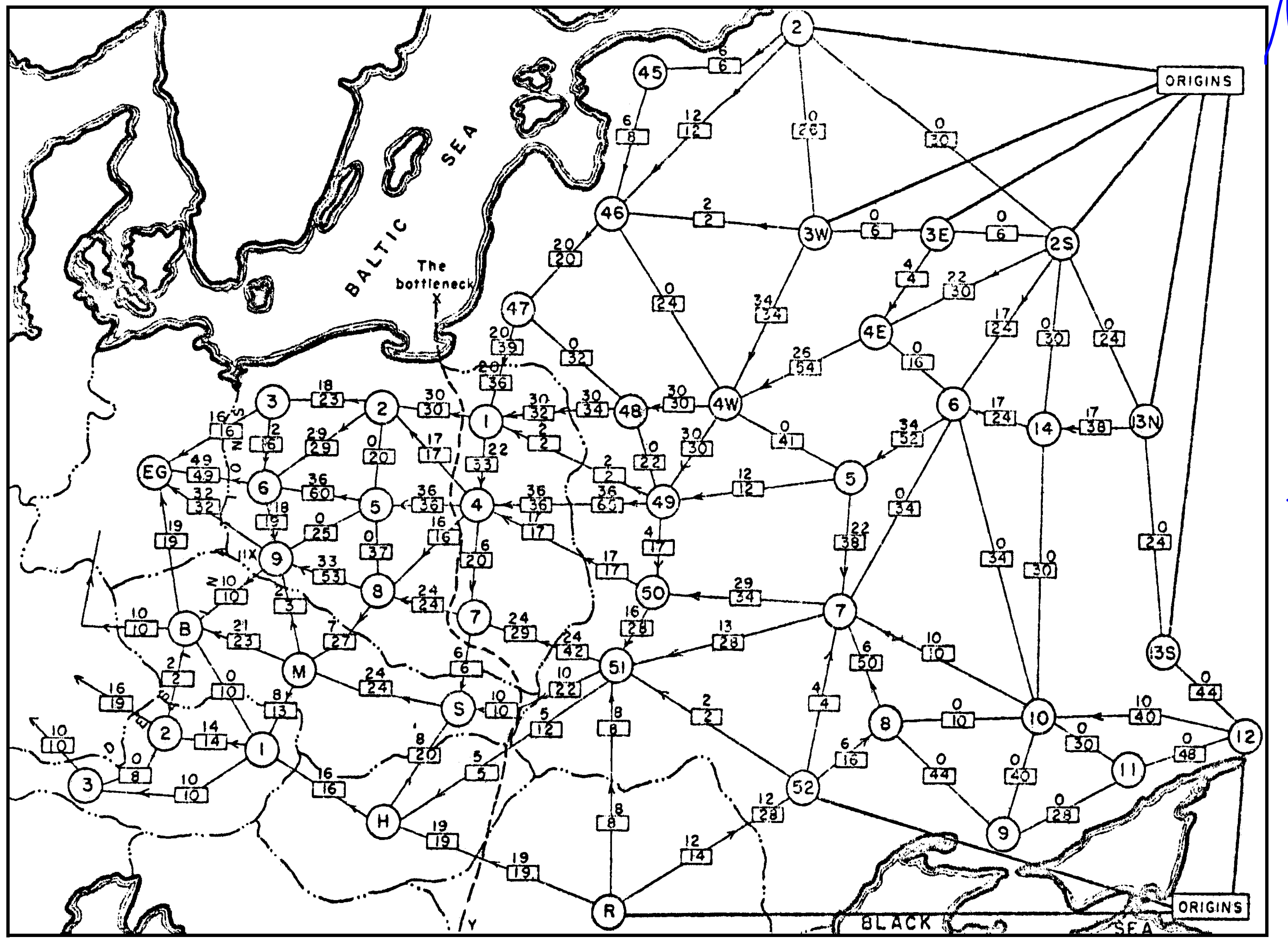


Figure 4 From Harris and Ross [3]: Schematic diagram of the railway network of the Western Soviet Union and East European countries, with a maximum flow of value 163,000 tons from Russia to Eastern Europe and a cut of capacity 163,000 tons indicated as 'The bottleneck'

courtesy Alexander Schrijver

flow networks

$$G = (V, E)$$

source + sink: source node s, sink node t

capacities: $c : E \to \mathbb{R}^+$ positive edge capacity

flow networks

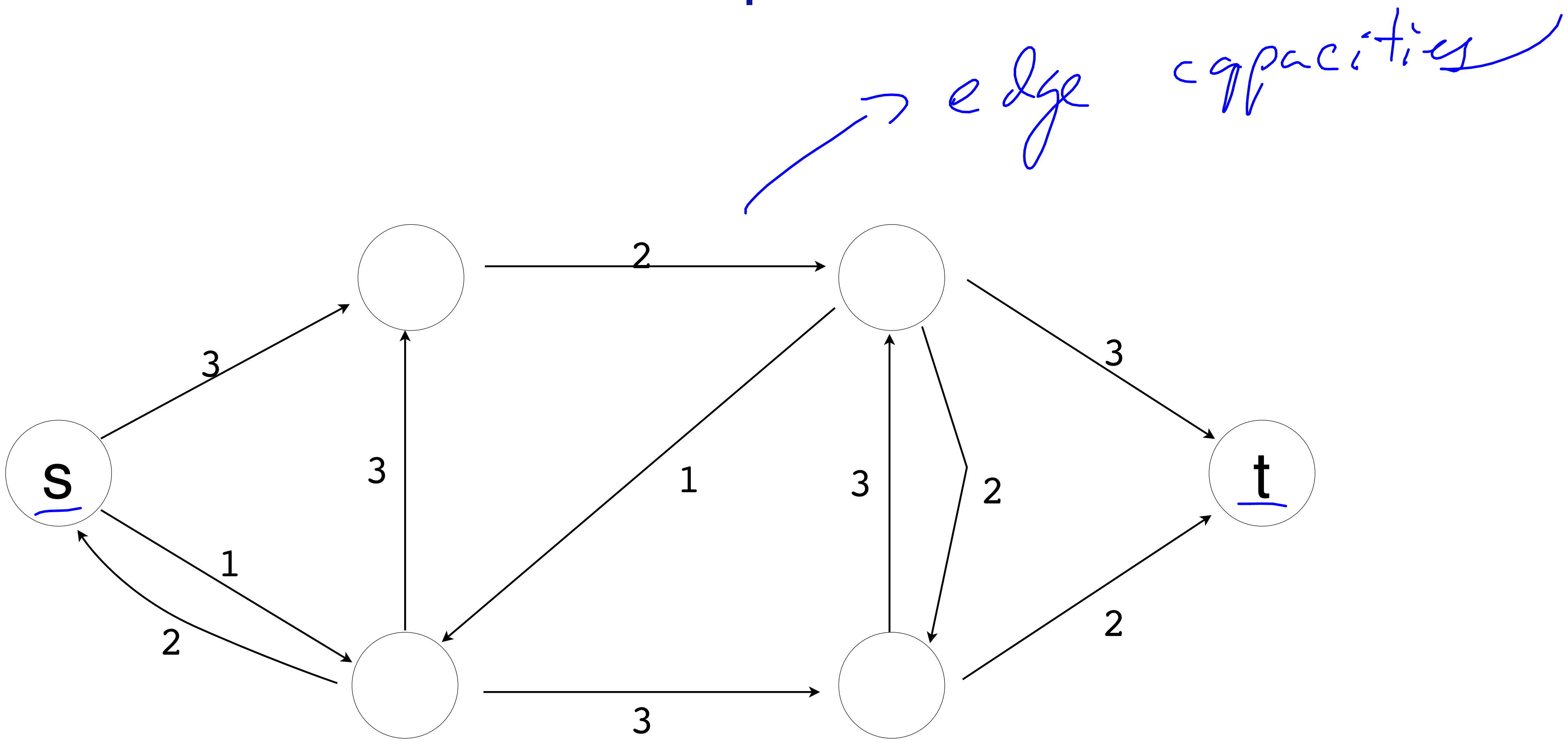
$$G = (V, E)$$

source + sink: node s, and t

capacities: $c(u, v)$

assumed to be 0 if no (u,v) edge

example



flow

map from edges to numbers: $f : E \to \mathbb{R}^+$ maps edges to numbers.

capacity constraint: for every edge $e \in E$, $f(e) \leq c(e)$

flow constraint: → for every node $v \in V - \{s, t\}$, $\text{INFLOW}(v) = \text{OUT}(v)$

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

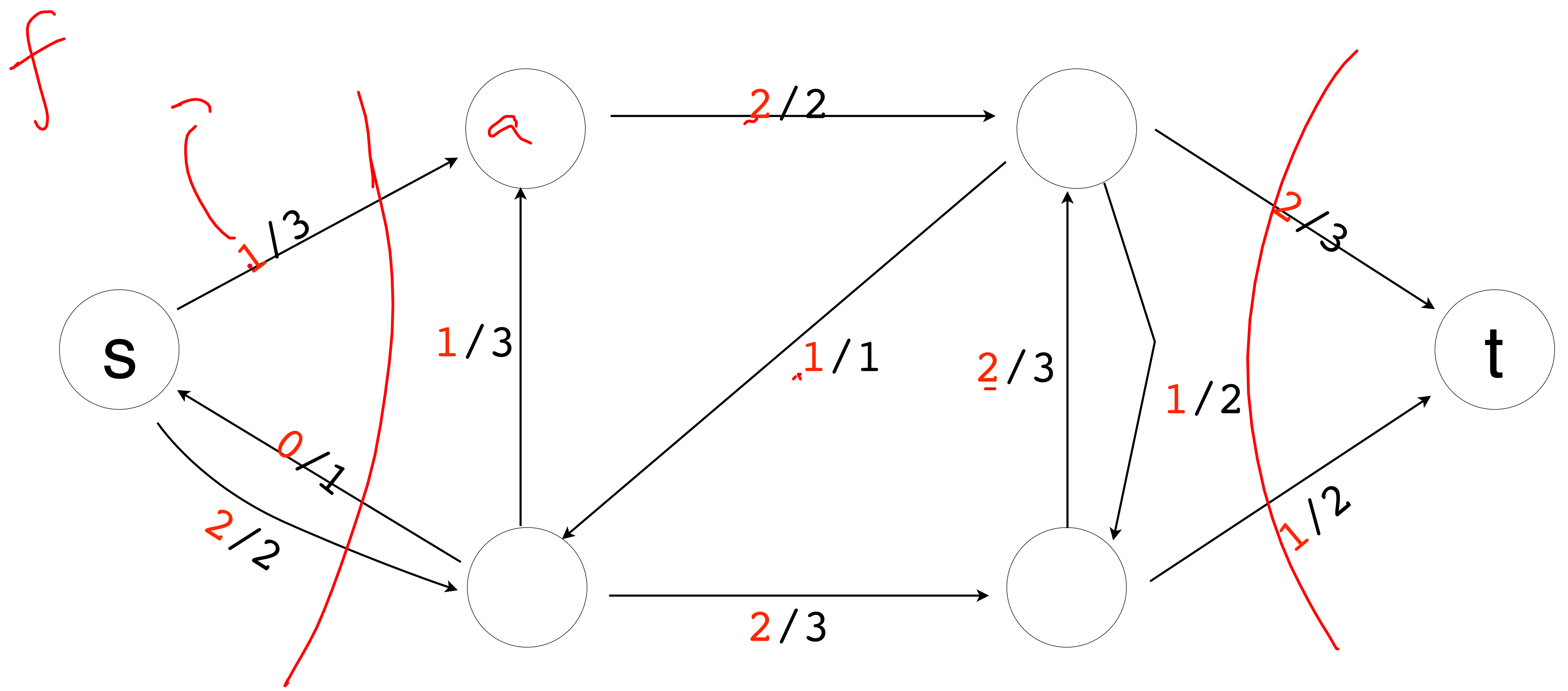
(OUT) (IN)

$|f| =$ OUTflow from s.

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

(Net outflow from s)

example



|f| = 3

max flow problem

given a graph G, compute

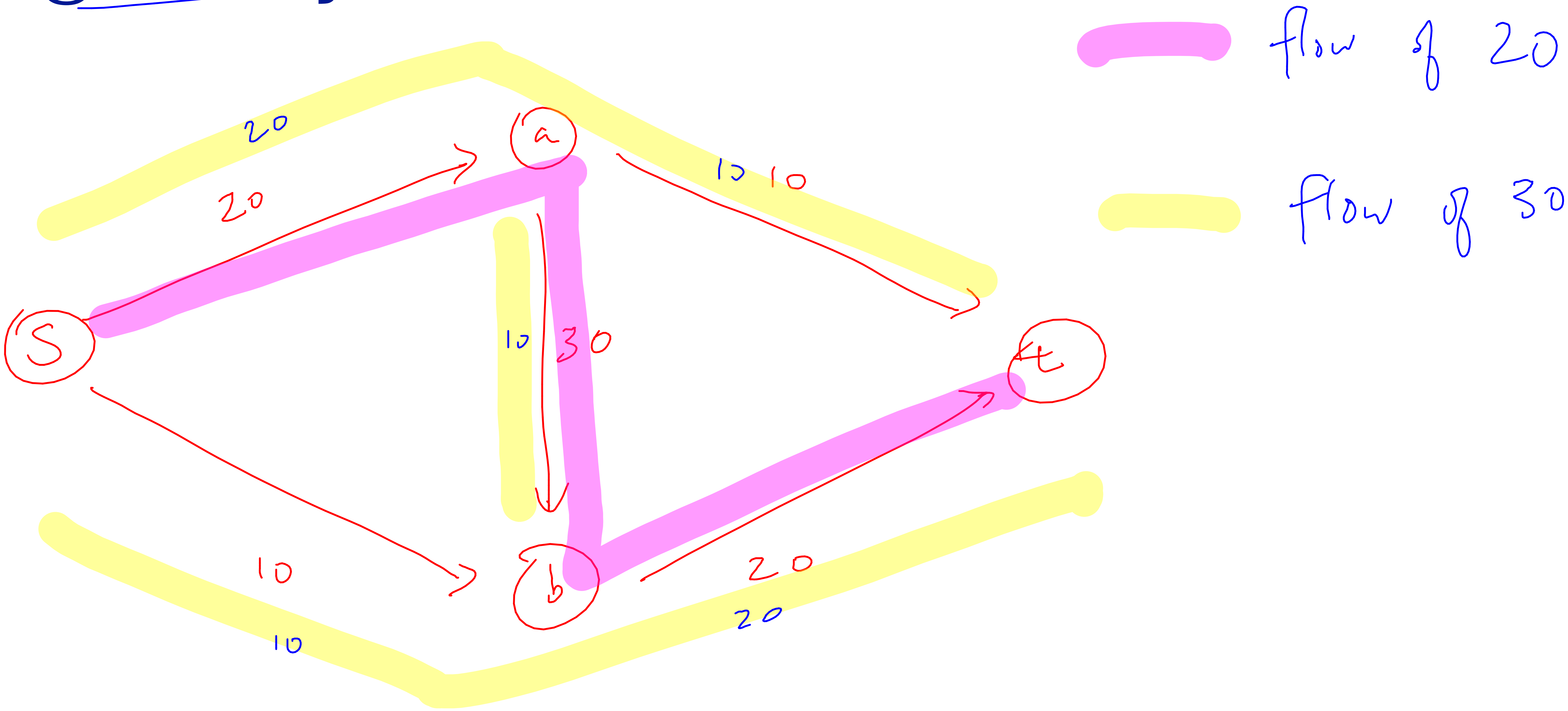
$G=(V,E)$

c : capacities

$\text{ARGMAX}_f \; |f|$

over all valid flows, find the maximum one

greedy solution?



hundreds of applications

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

will discuss many of these applications soon

Algorithms for max flow

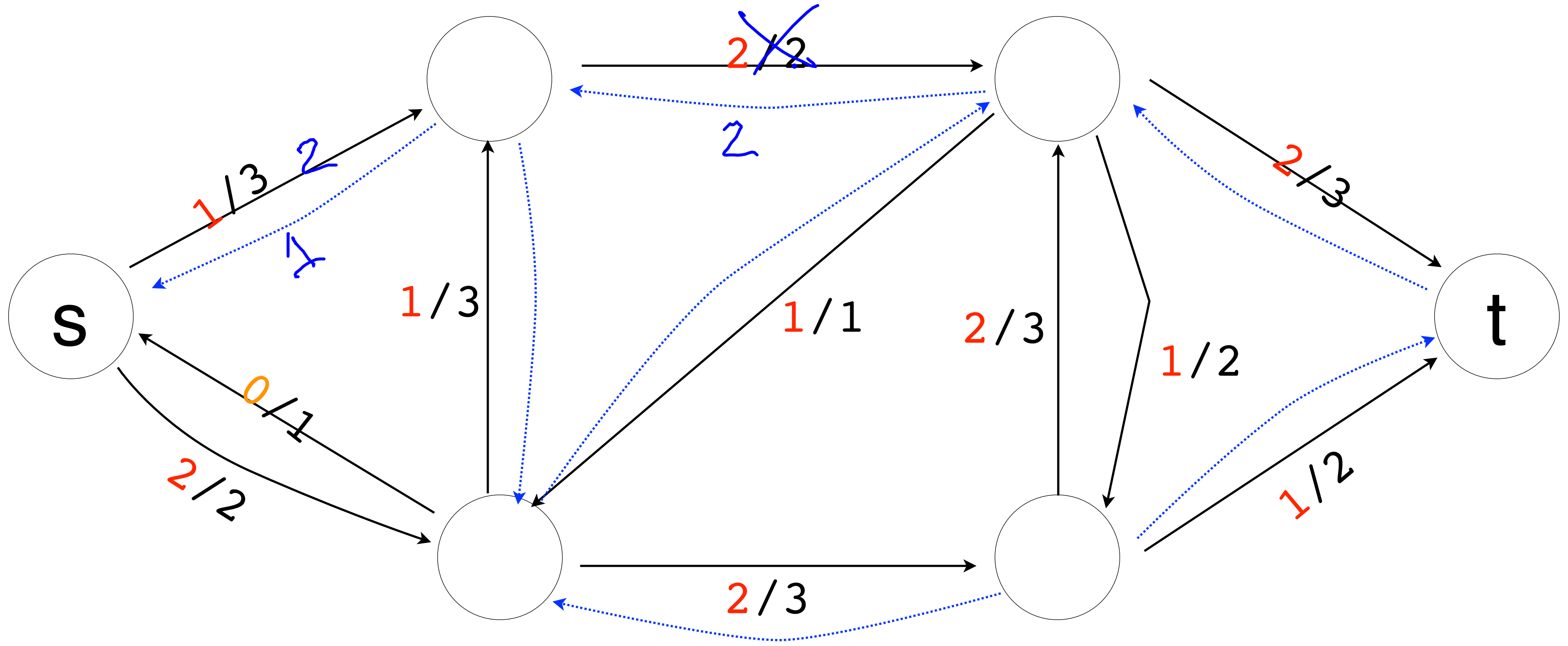
Residual graphs

$G_f = (V, E_f)$ based on flow f.

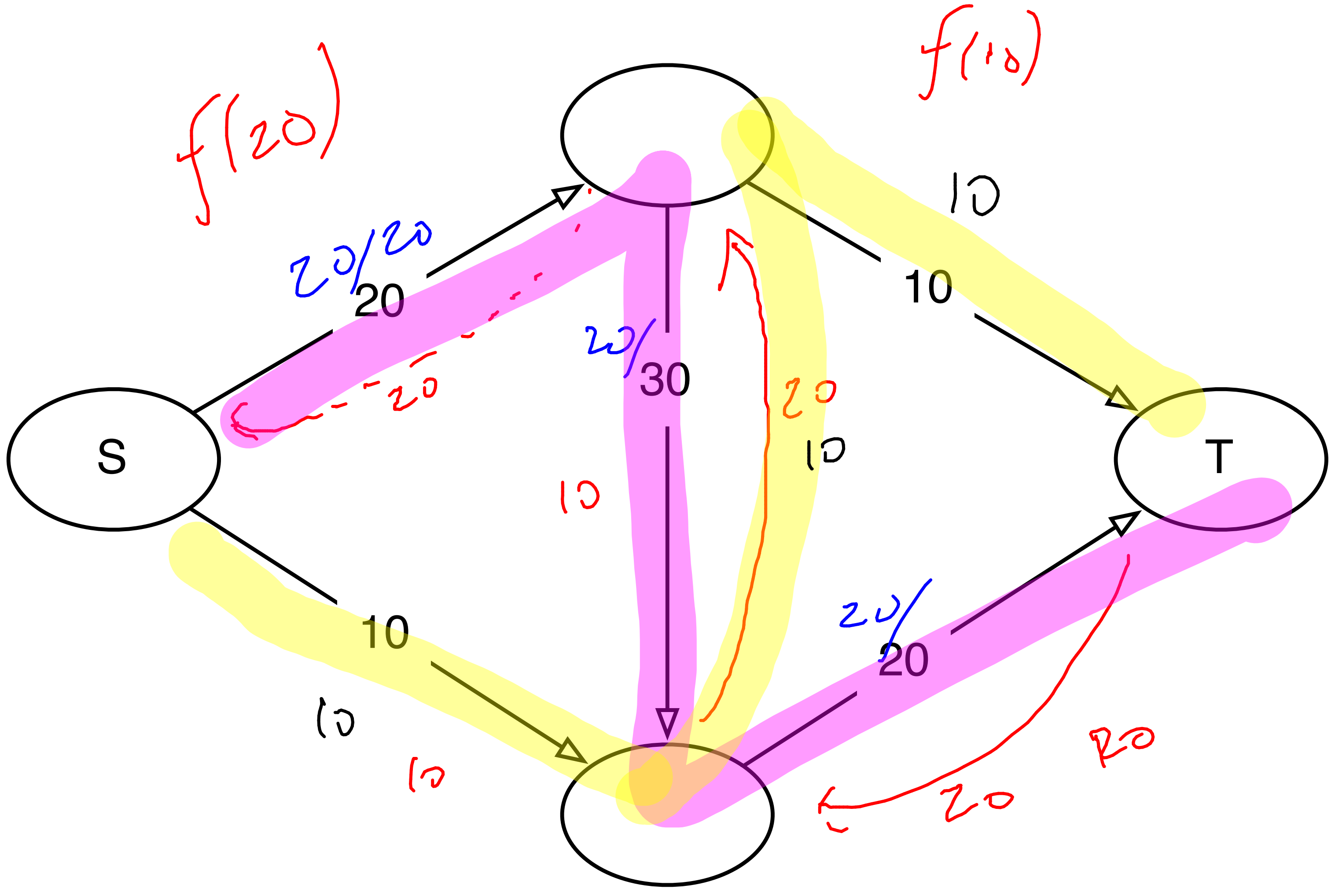
new set of E_f : if $f(e) > 0$ in f, then (e = (u,v))

- add the edge $e = (u,v)$ w/capacity $c(e) - f(e)$
- add the edge $e' = (v,u)$ w/capacity $f(e)$

example residual graph



why residual graphs ?



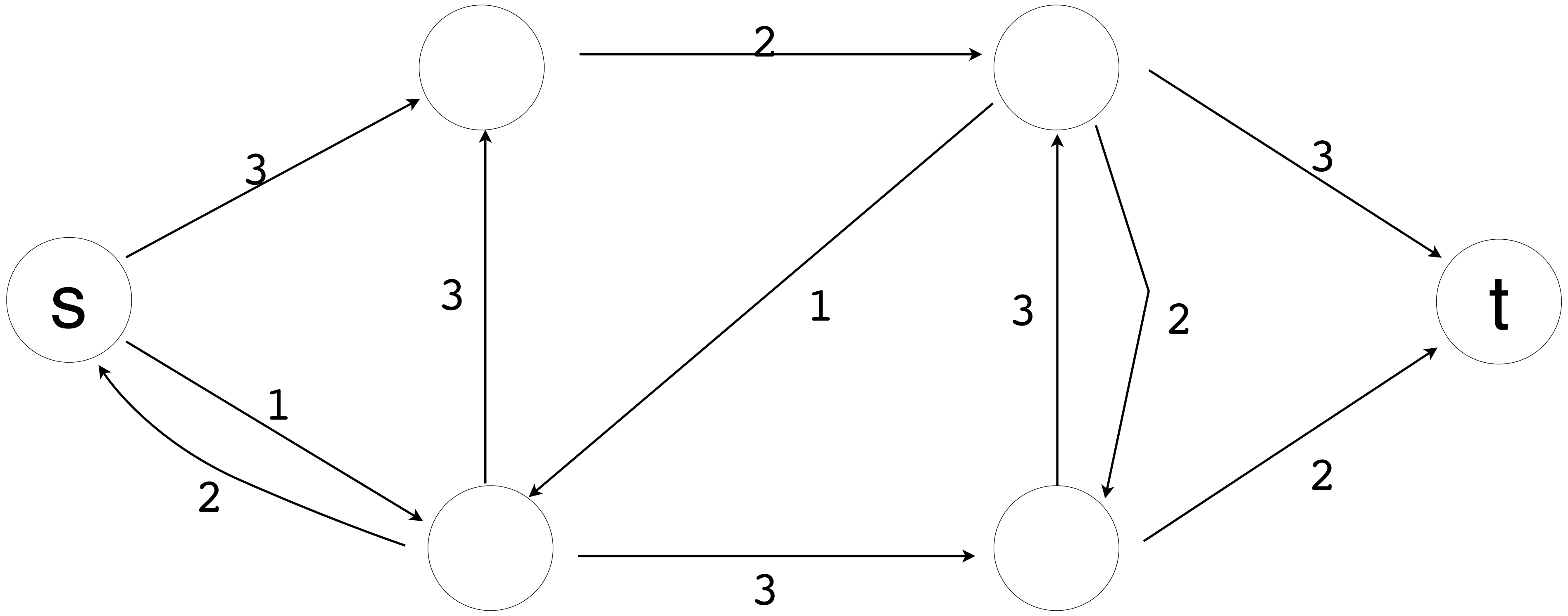
augmenting paths

def:

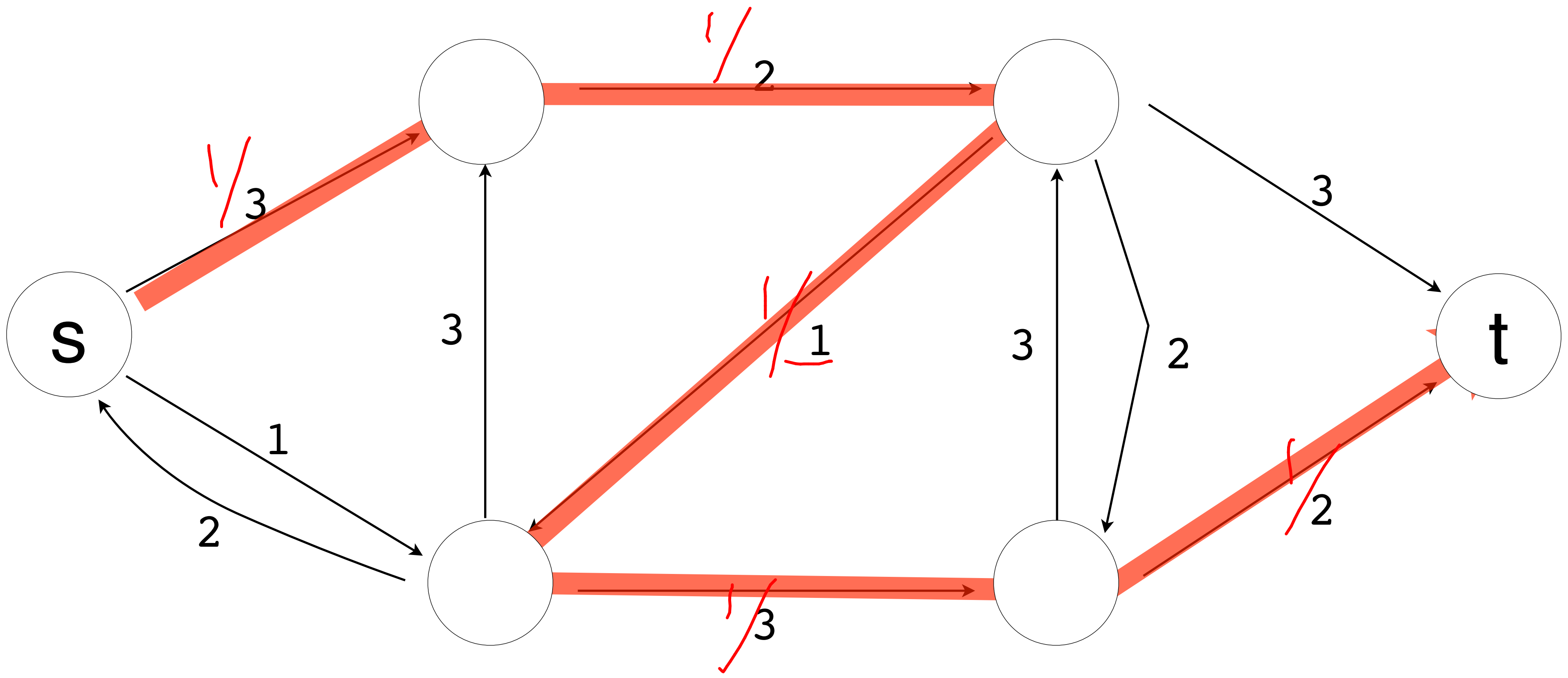
ford-fulkerson

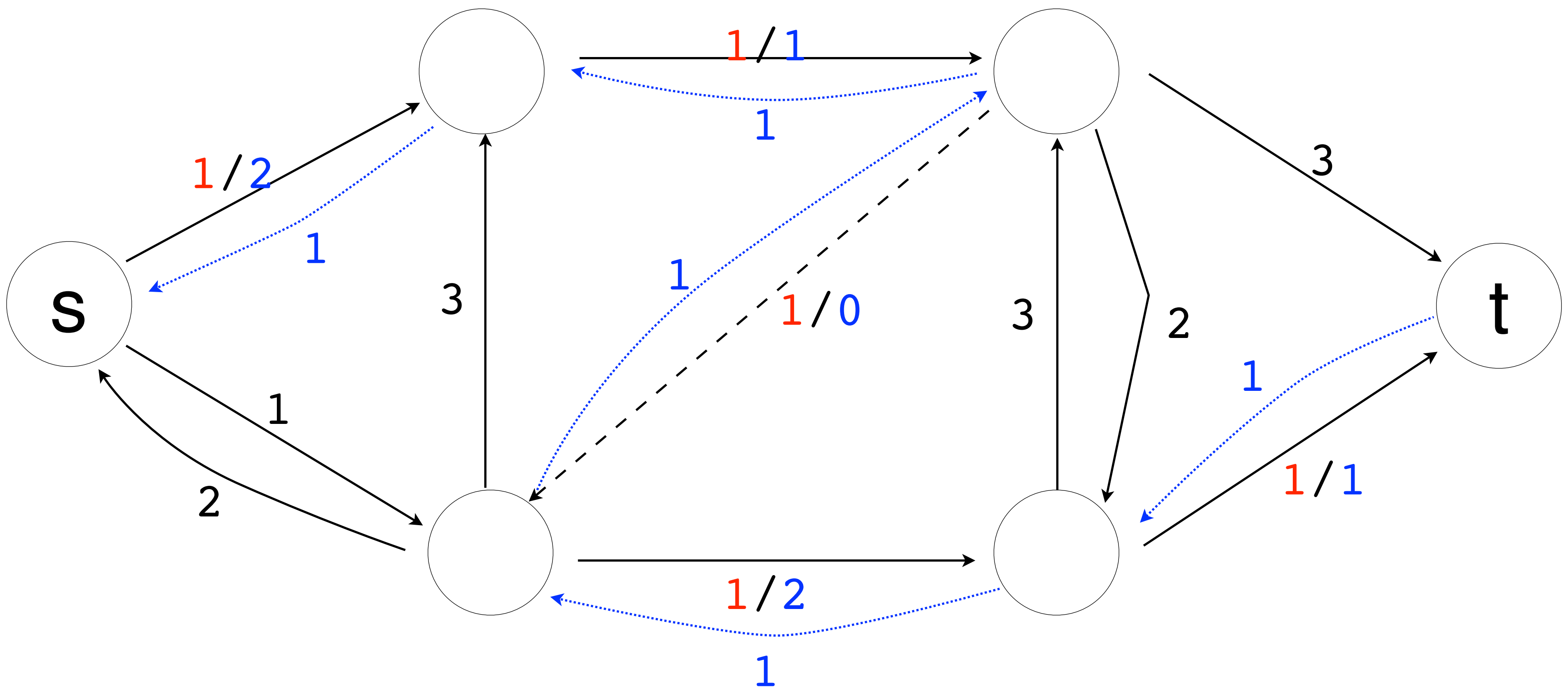
initialize $f(u,v) \leftarrow 0 \; \forall u,v$
while exists an augmenting path *p* in G_f
    augment *f* with $c_f(p) = \min_{(u,v)\in p} c_f(u,v)$

→ i.e a path from s to t

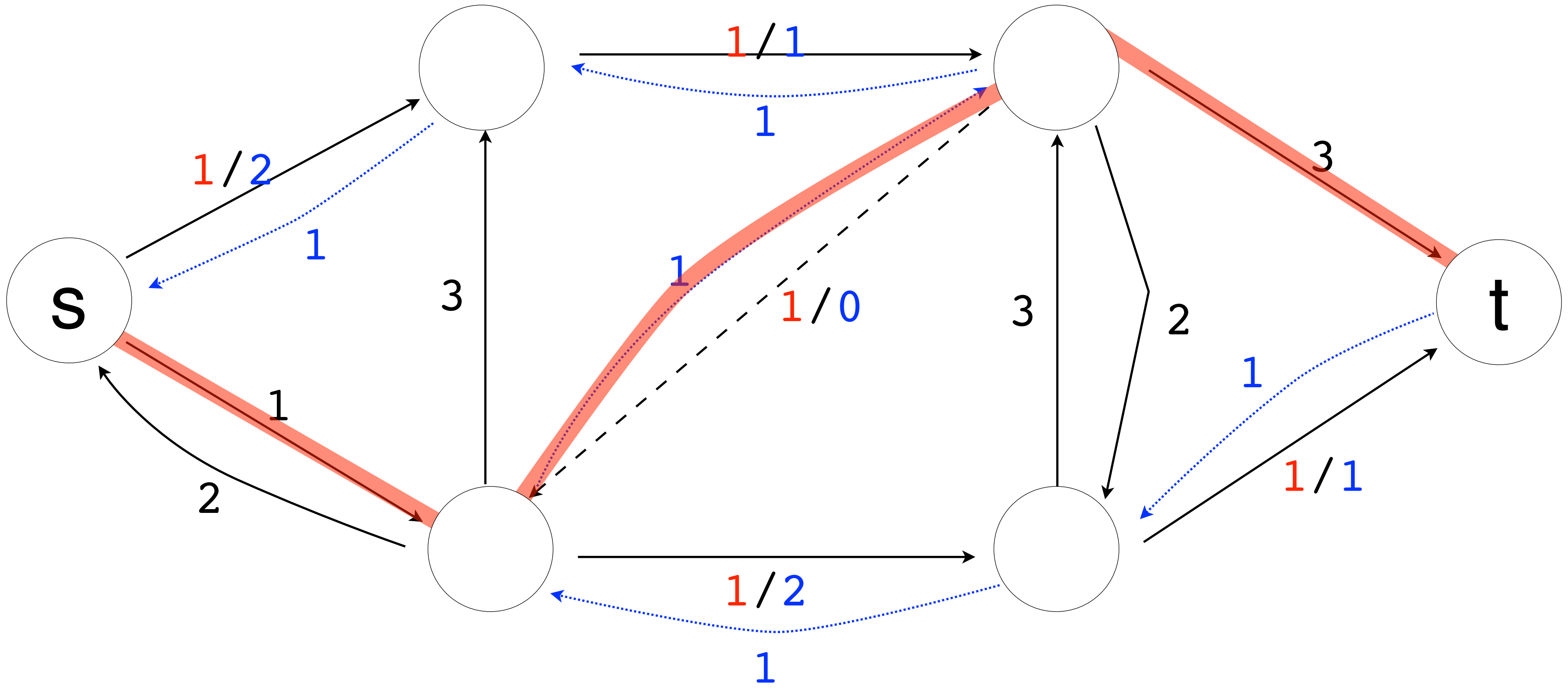


s
t
3
1
2
3
2
1
3
3
2
3
3
2

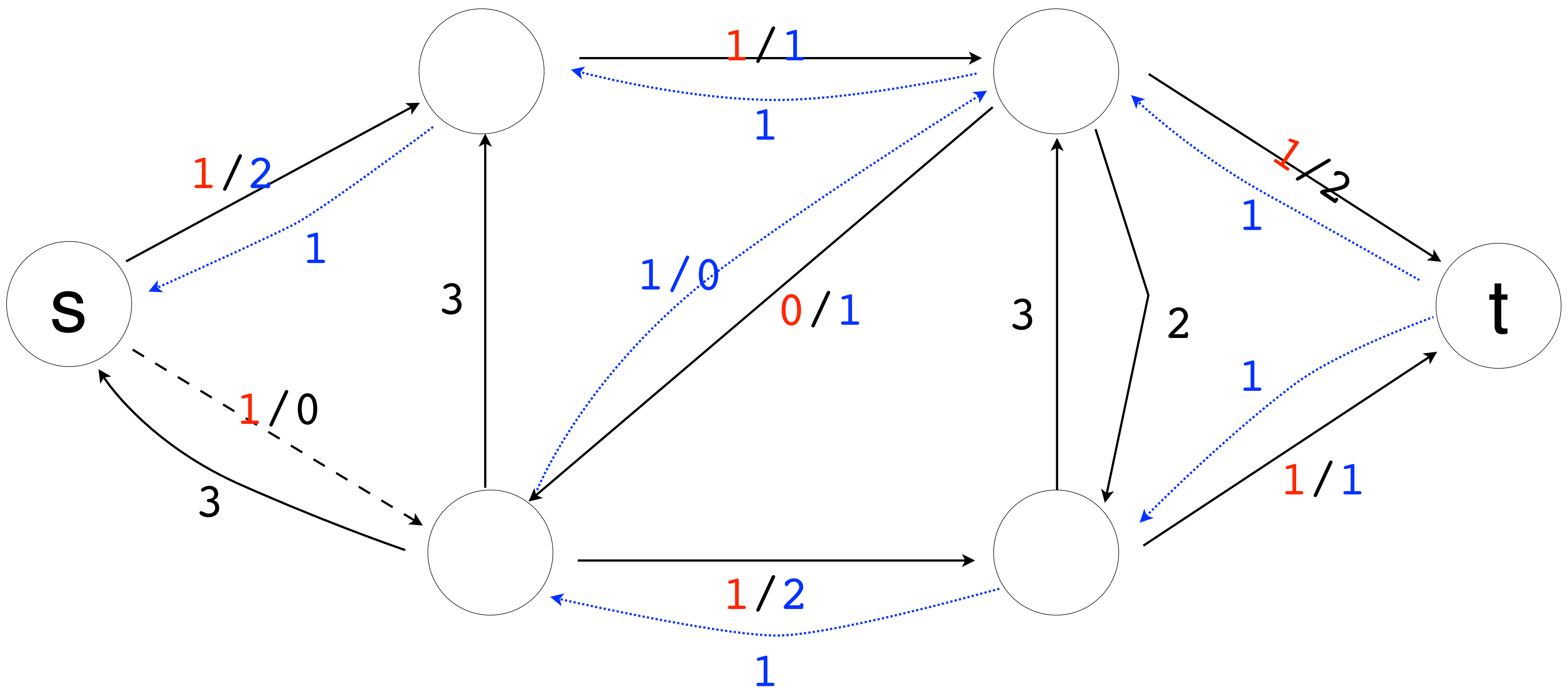

s
t
3
2
3
1
1
3
2
3
3
3
2
2
1
1
1
1
1



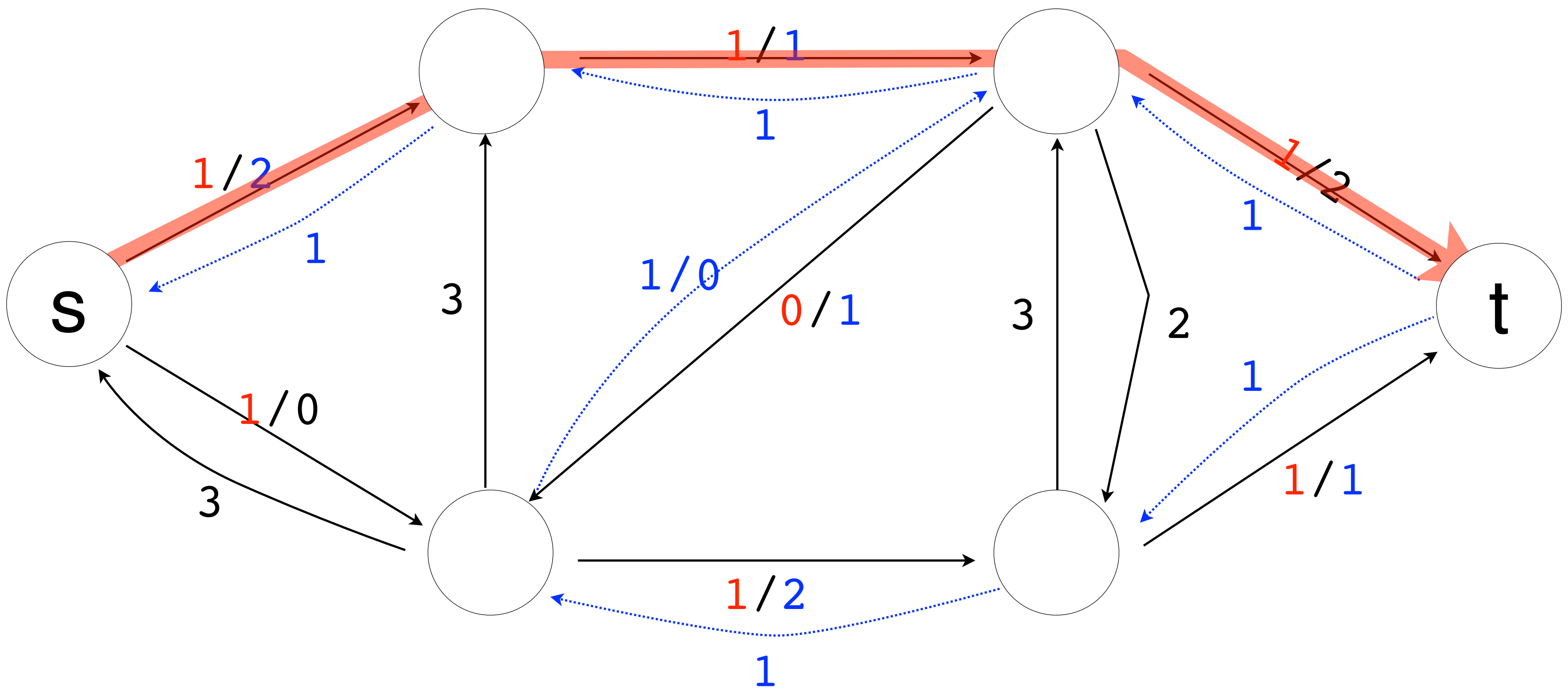
s
t
1 / 2
1
1 / 1
1
3
1
1 / 0
3
3
2
1
1 / 1
1
2
1 / 2
1



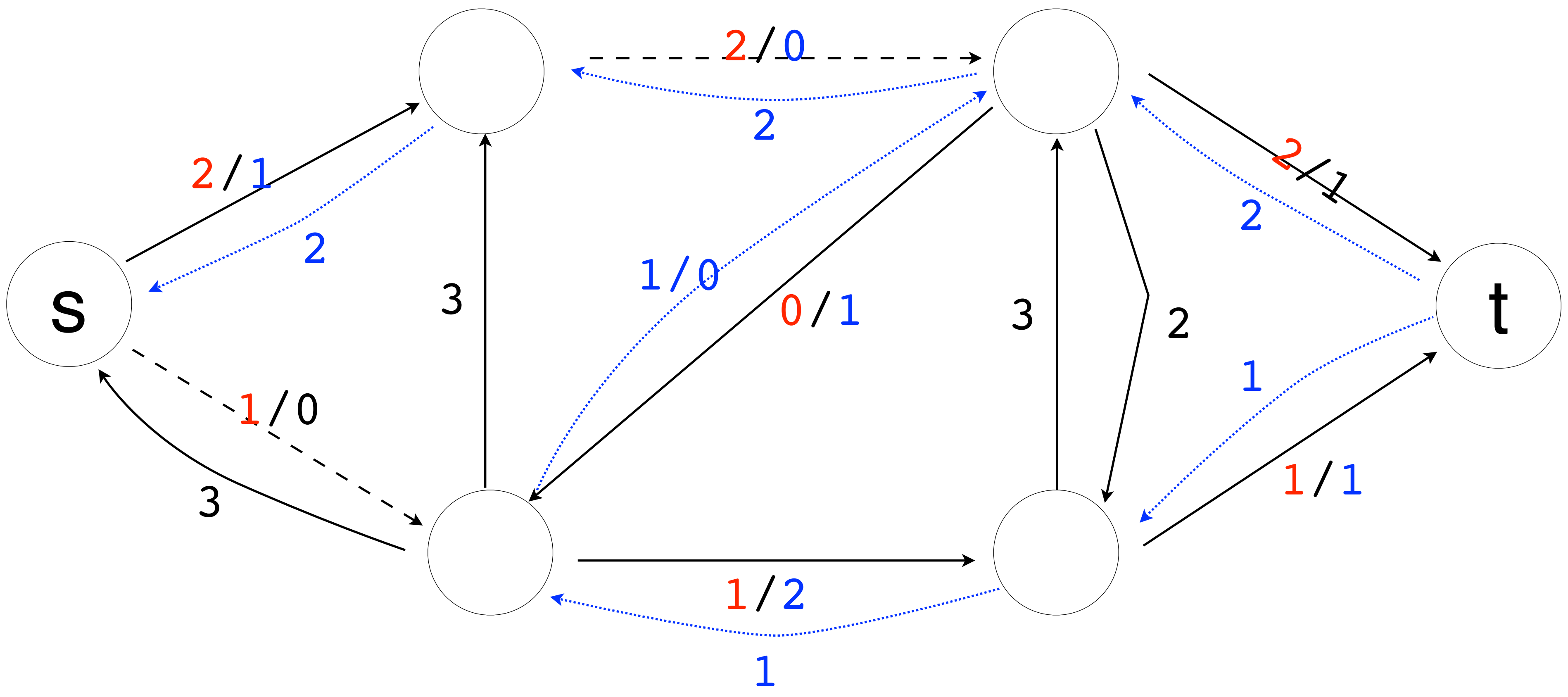
s
t
1 / 2
1
1 / 1
1
3
1
1 / 0
3
2
3
1
1 / 1
1
2
1 / 2
1



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



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



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

ford-fulkerson

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

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

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

$\Theta(E|f|)$

Cuts

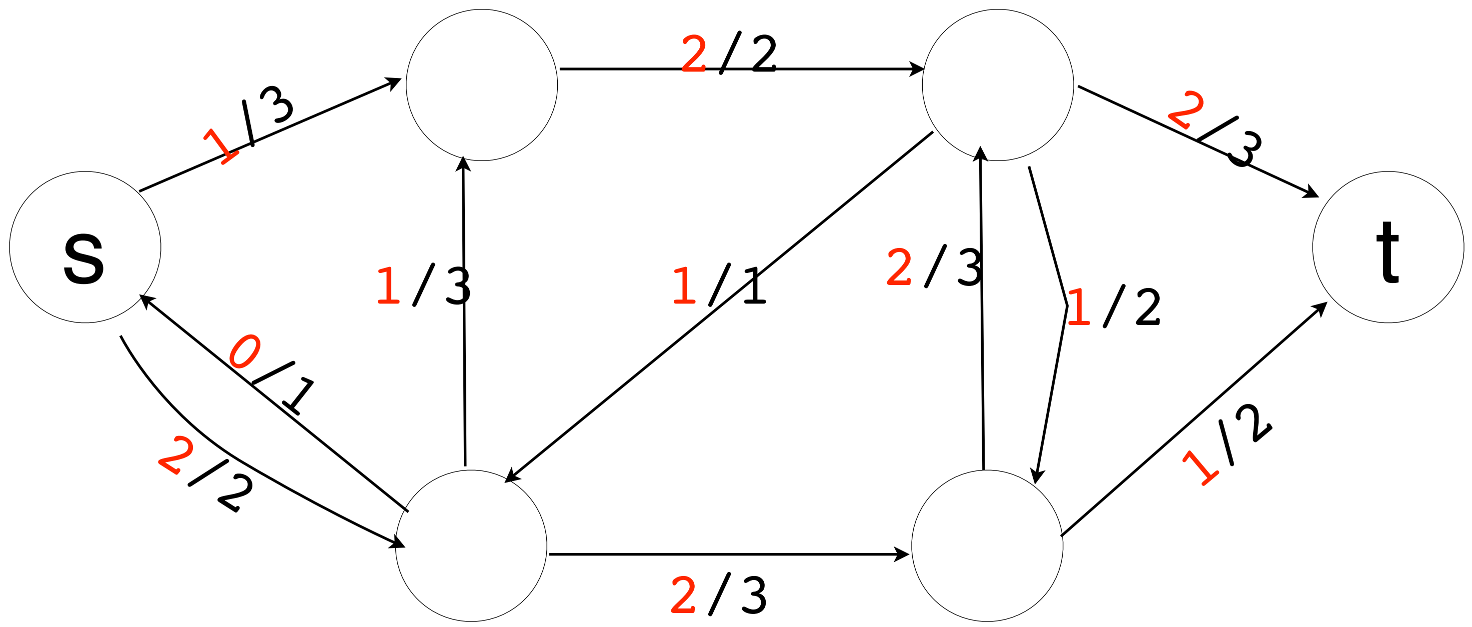
Def of a cut:

cost of a cut:

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

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

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

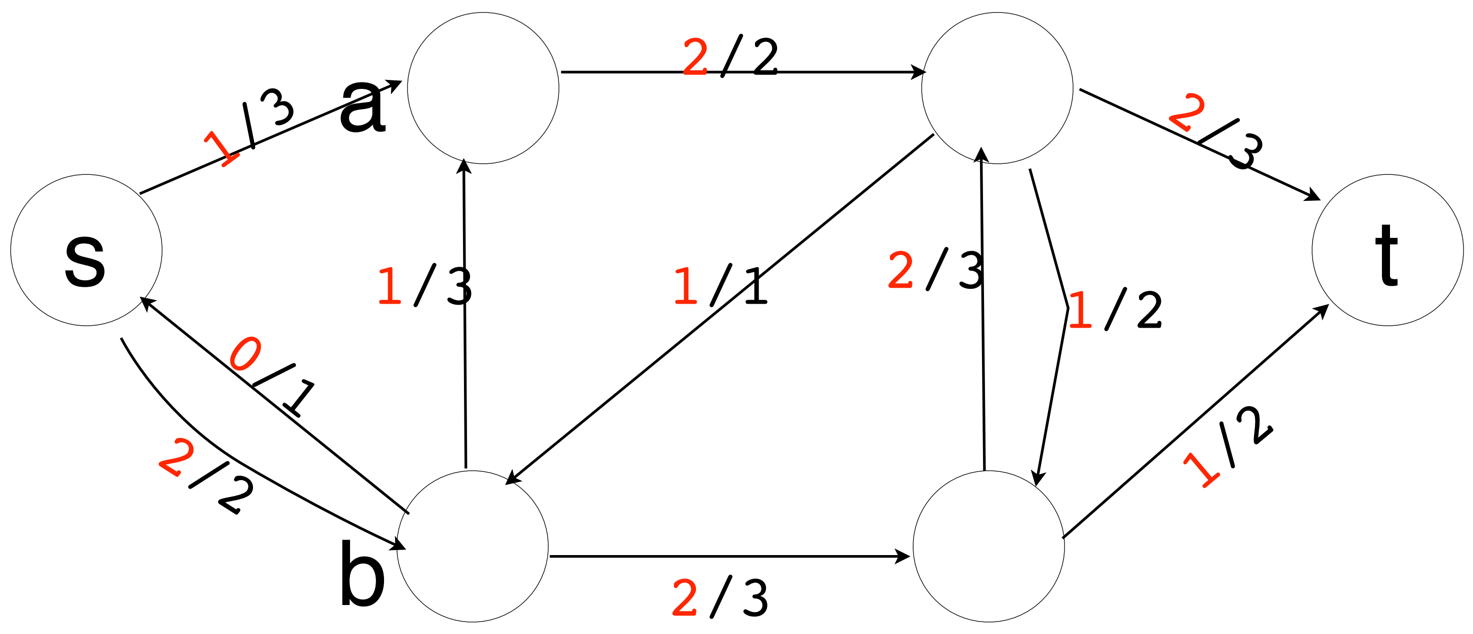


example:

A property to remember

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

proof:

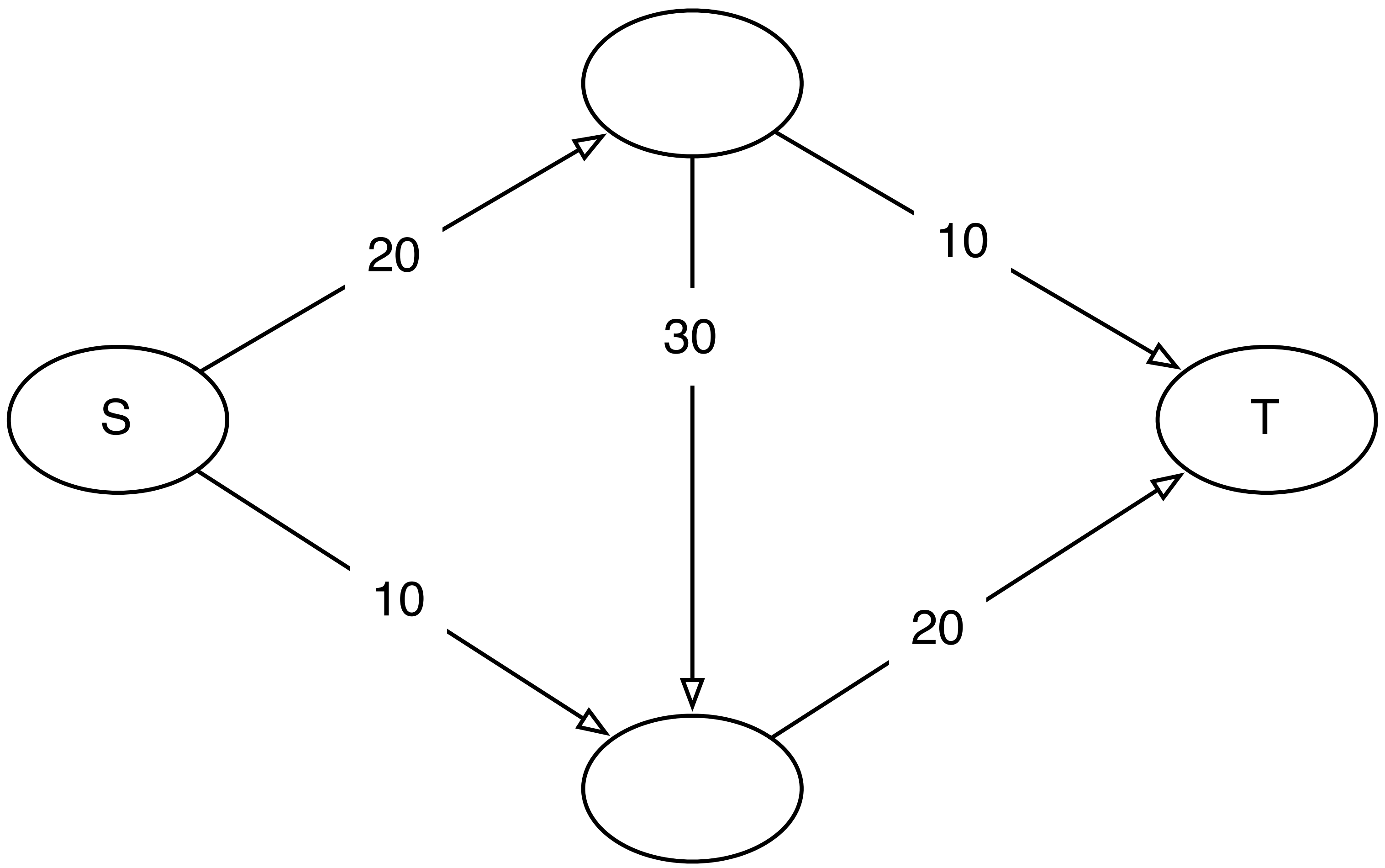


s
a
b
t
1/3
2/2
2/3
1/3
1/1
2/3
1/2
0/1
2/2
2/3
1/2

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

(finishing proof)

why residual graphs ?



augmenting paths

def:

Thm: max flow = min cut

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

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

thm: max flow = min cut

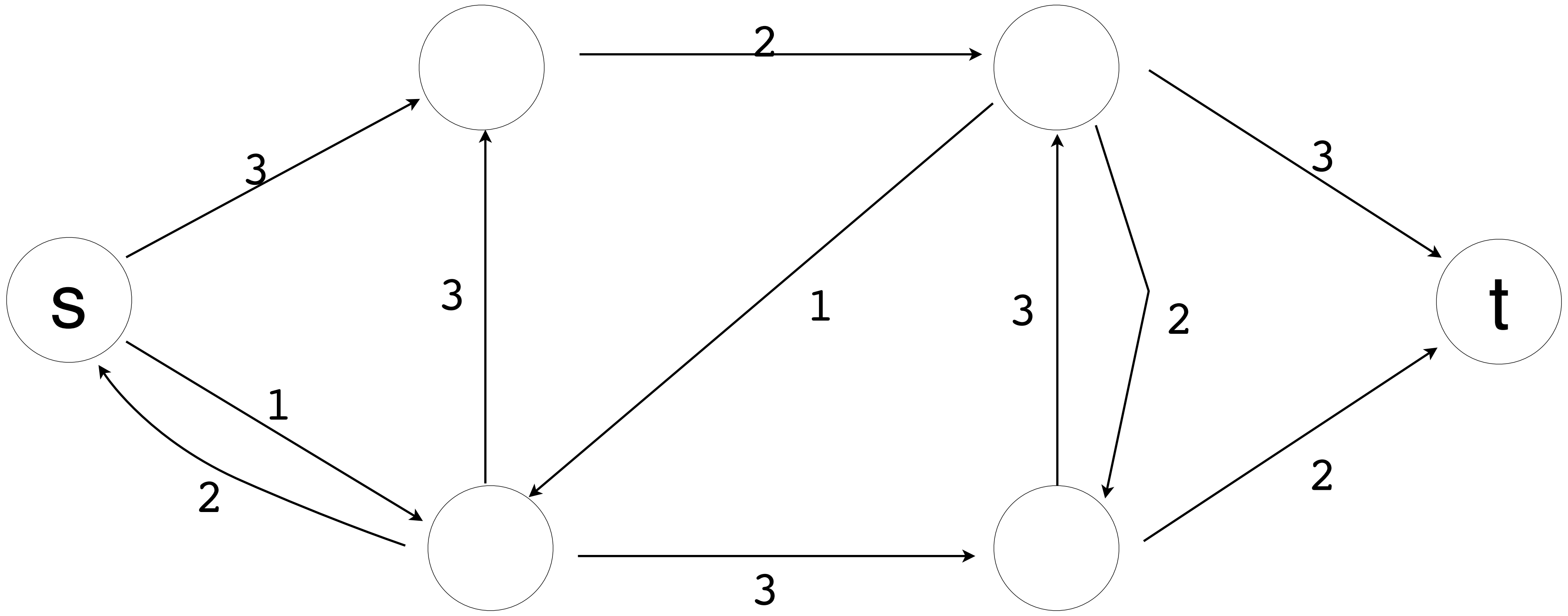
$$\max_f |f| = \min_{S,T} ||S, T||$$

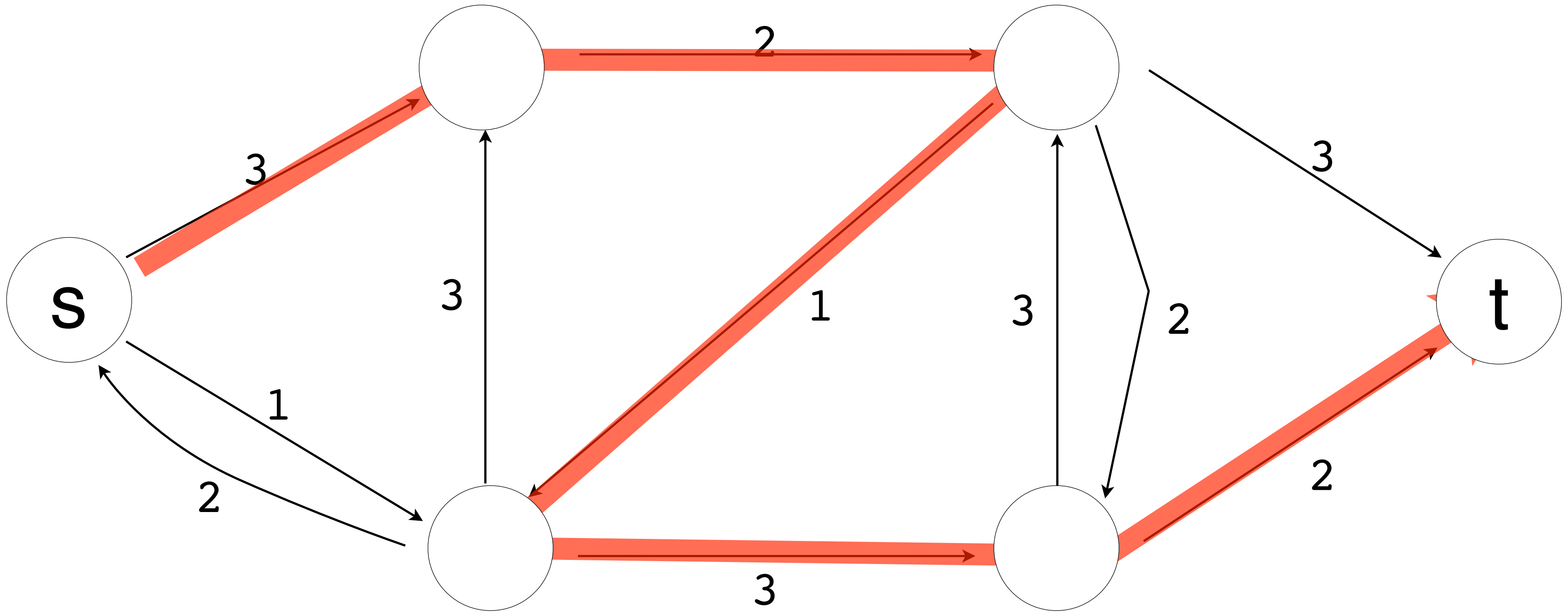
ford-fulkerson

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

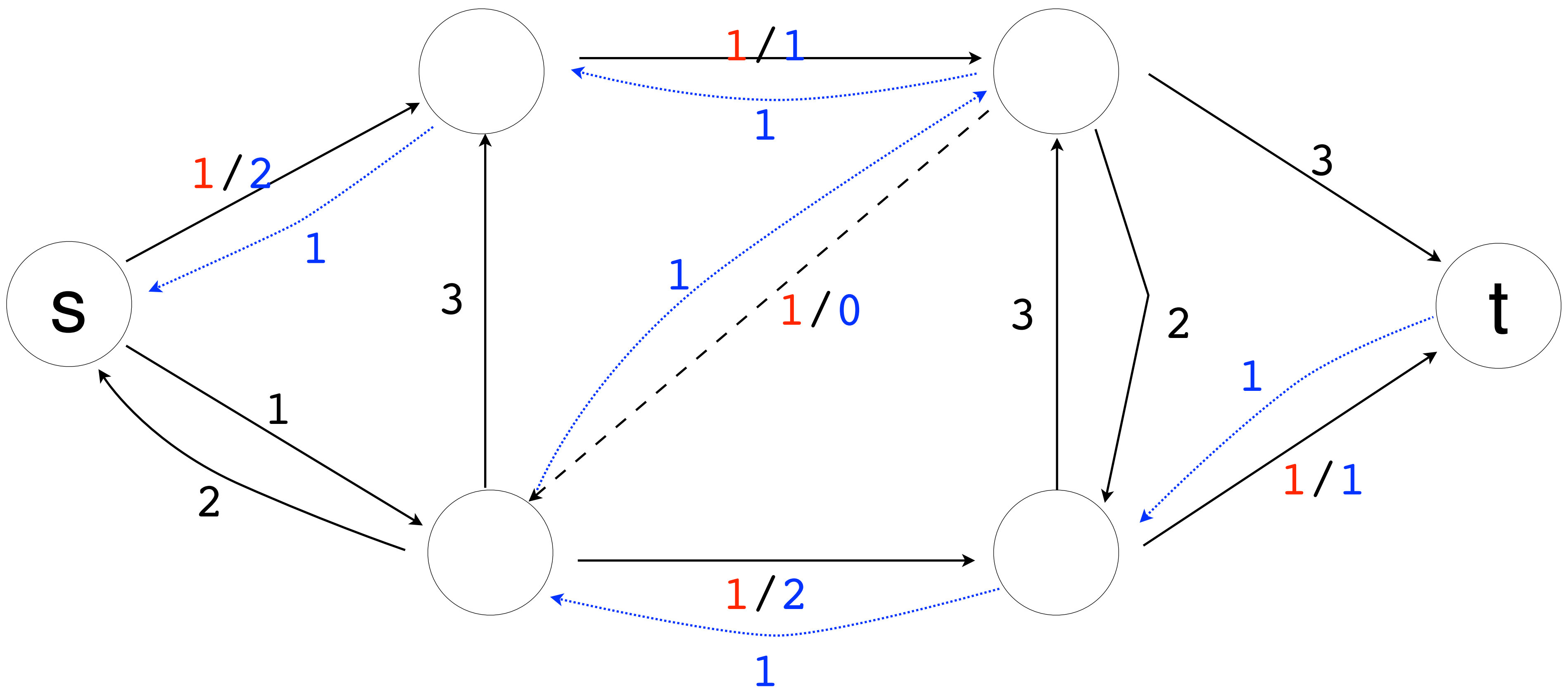
while exists an augmenting path *p* in G_f

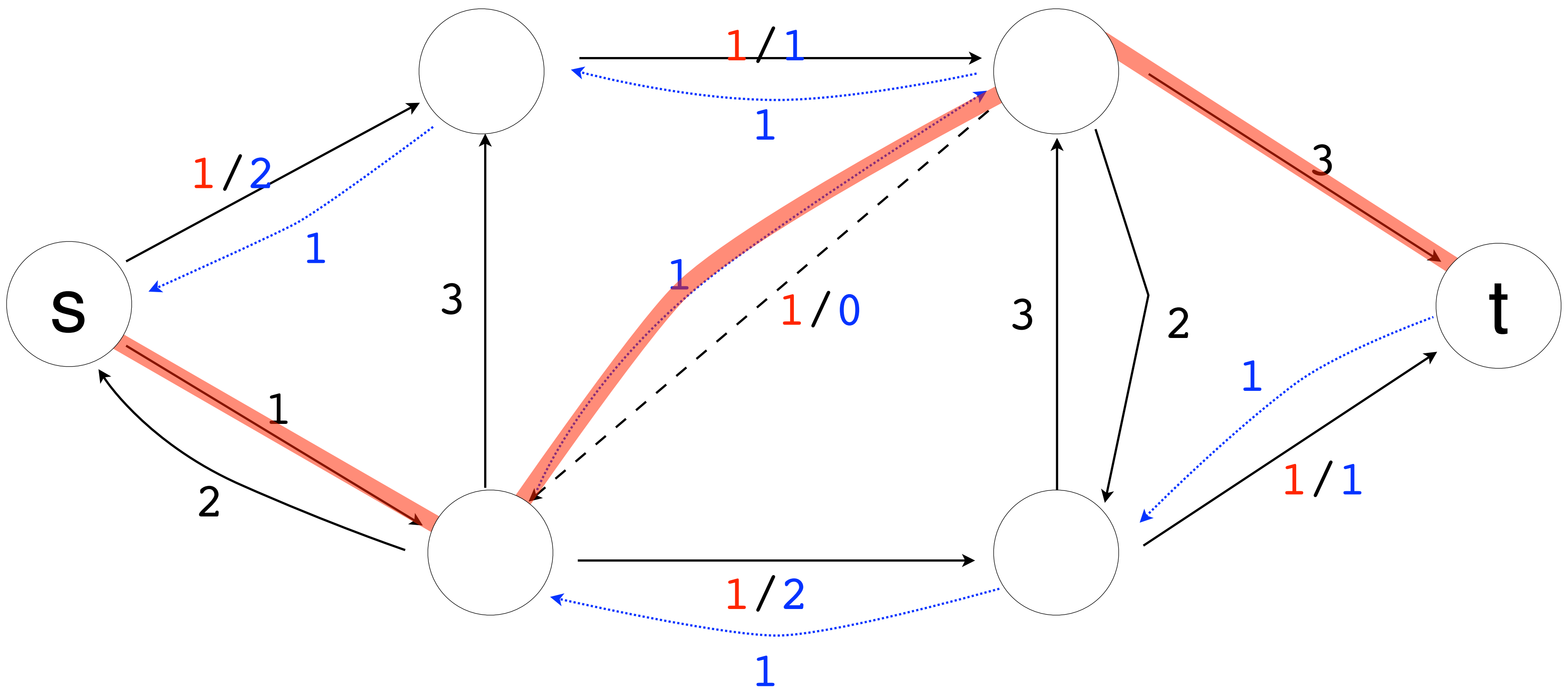
augment *f* with $c_f(p) = \min_{(u,v)\in p} c_f(u,v)$

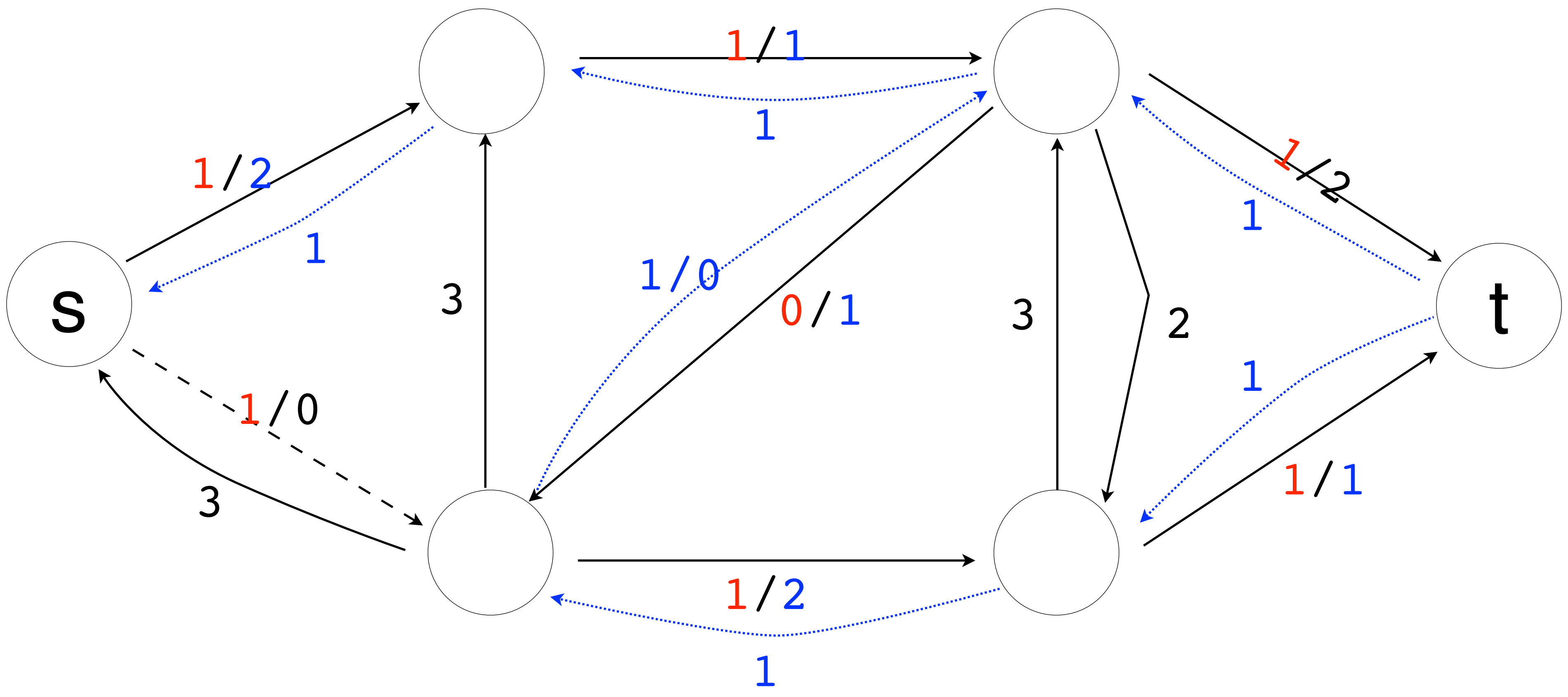

s
t
3
1
2
3
2
1
3
3
2
3
2

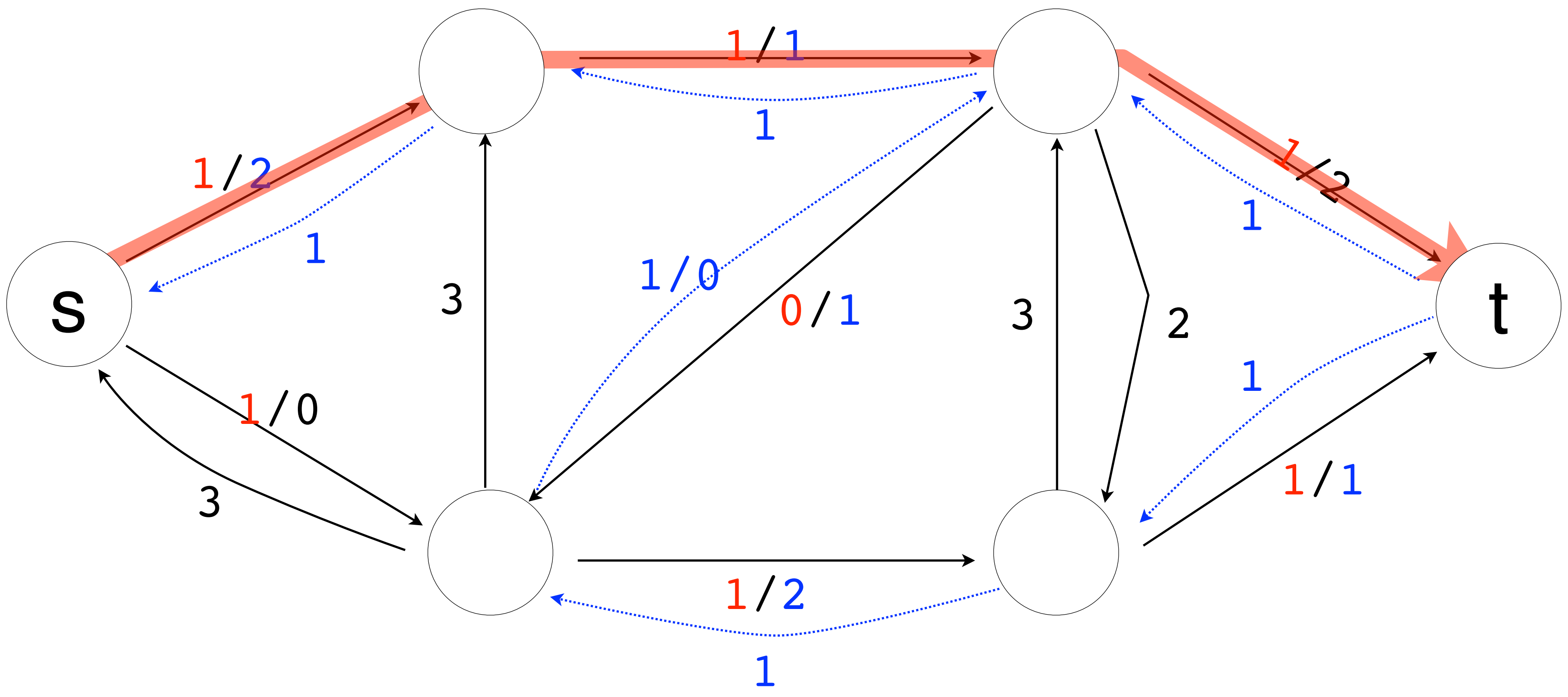


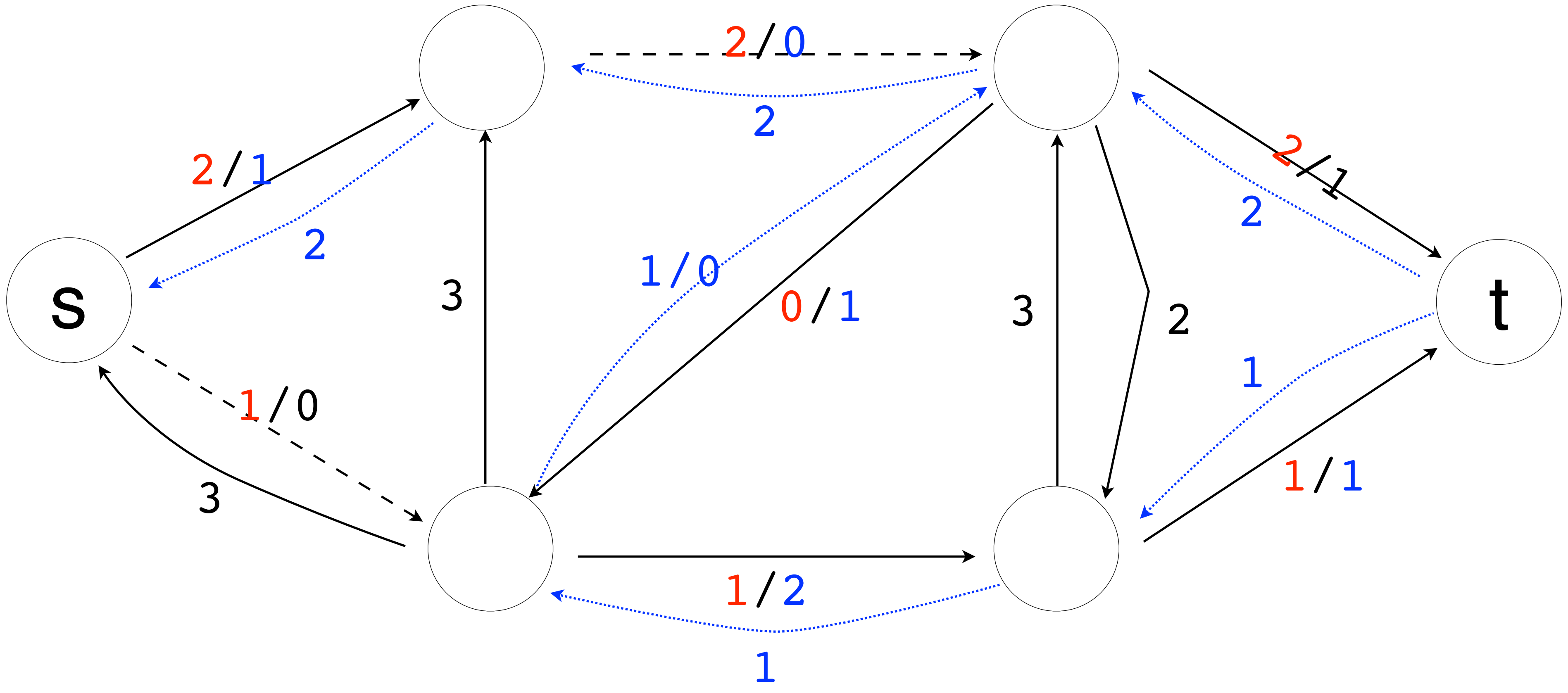
s
t
3
2
3
1
3
2
3
1
2
3
2


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


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


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


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



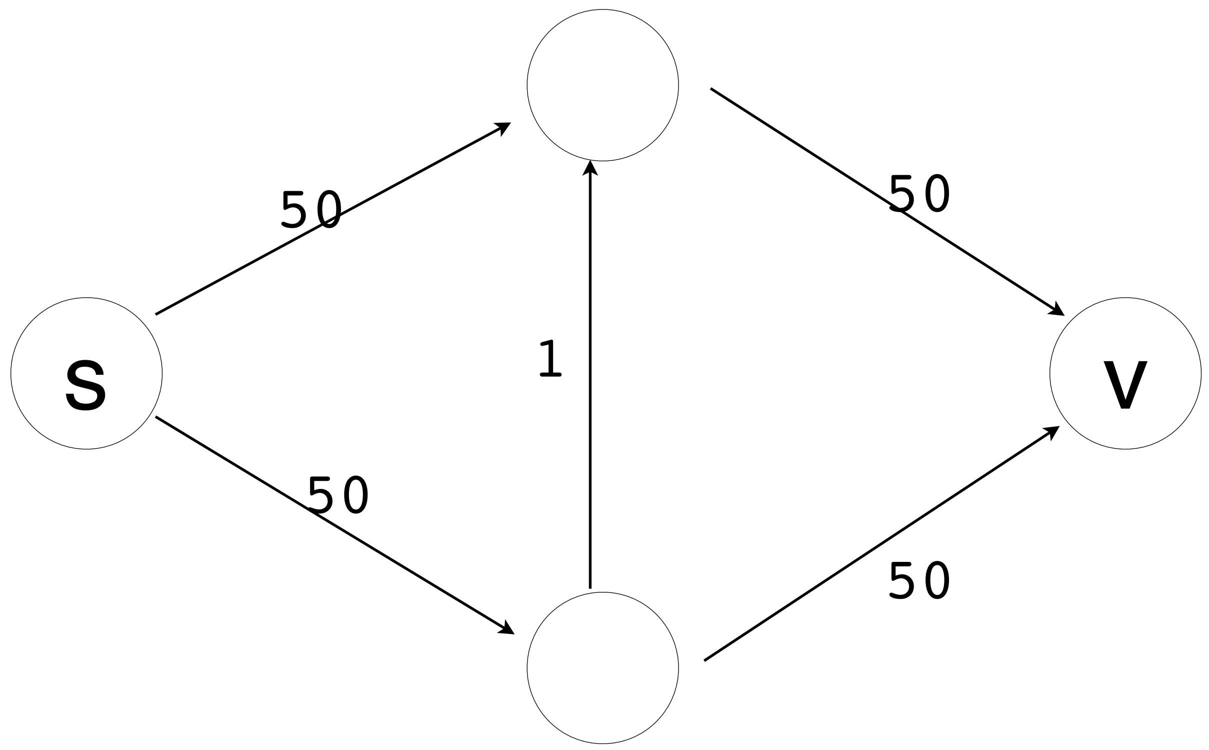
s
t
2 / 1
2
2 / 0
2
2 / 1
2
3
1 / 0
0 / 1
3
2
1 / 0
3
1
1 / 1
1 / 2
1

ford-fulkerson

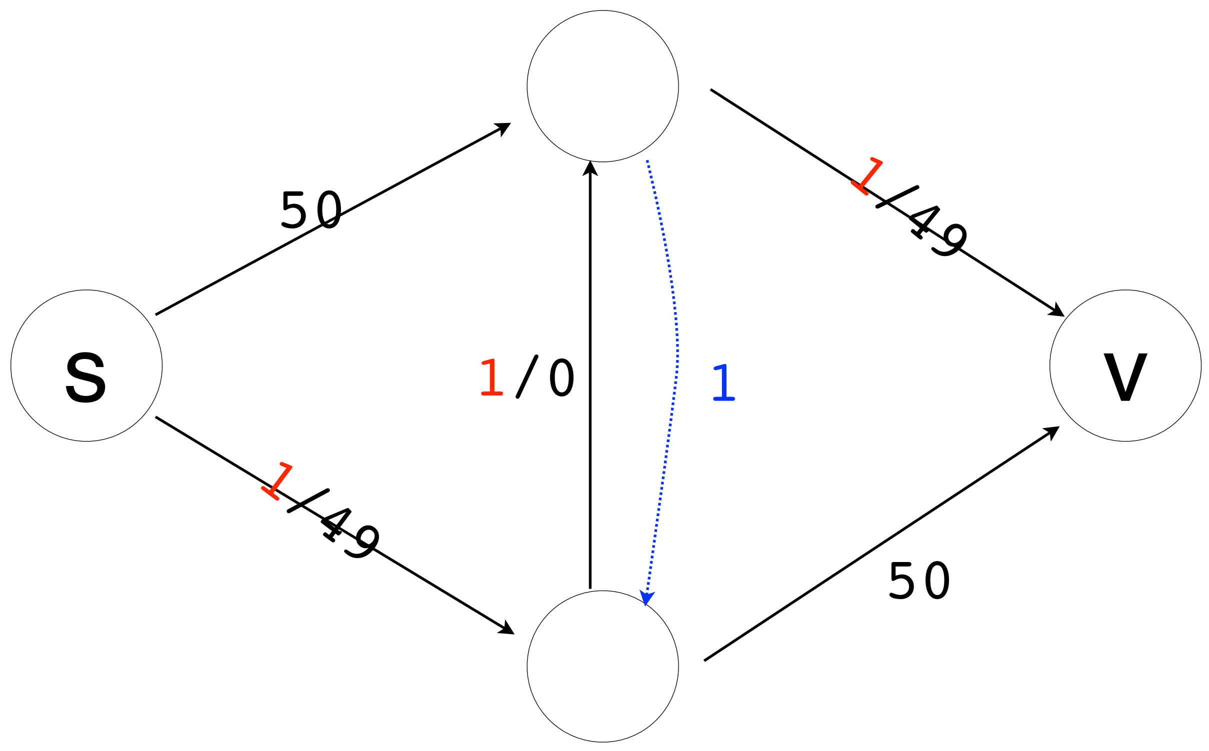
initialize $f(u,v) \leftarrow 0 \; \forall u,v$
while exists an augmenting path *p* in G_f
augment *f* with $c_f(p) = \min_{(u,v)\in p} c_f(u,v)$

time to find an augmenting path:

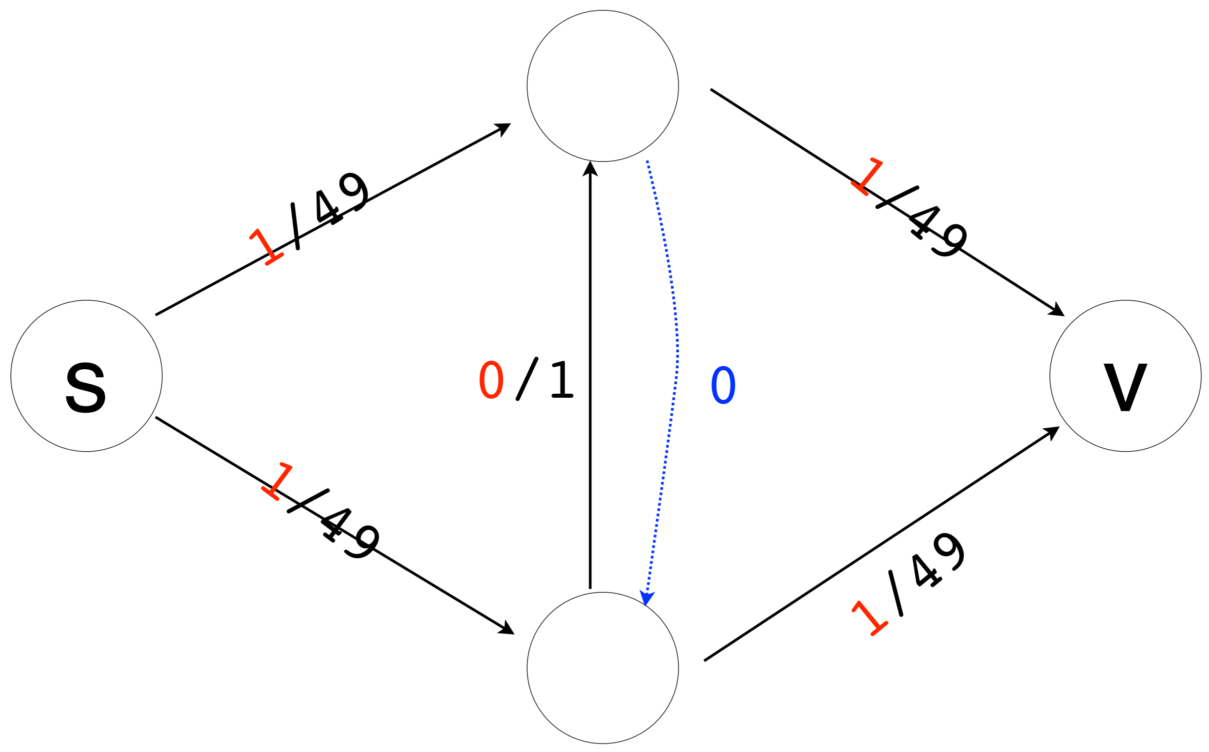
number of iterations of while loop:



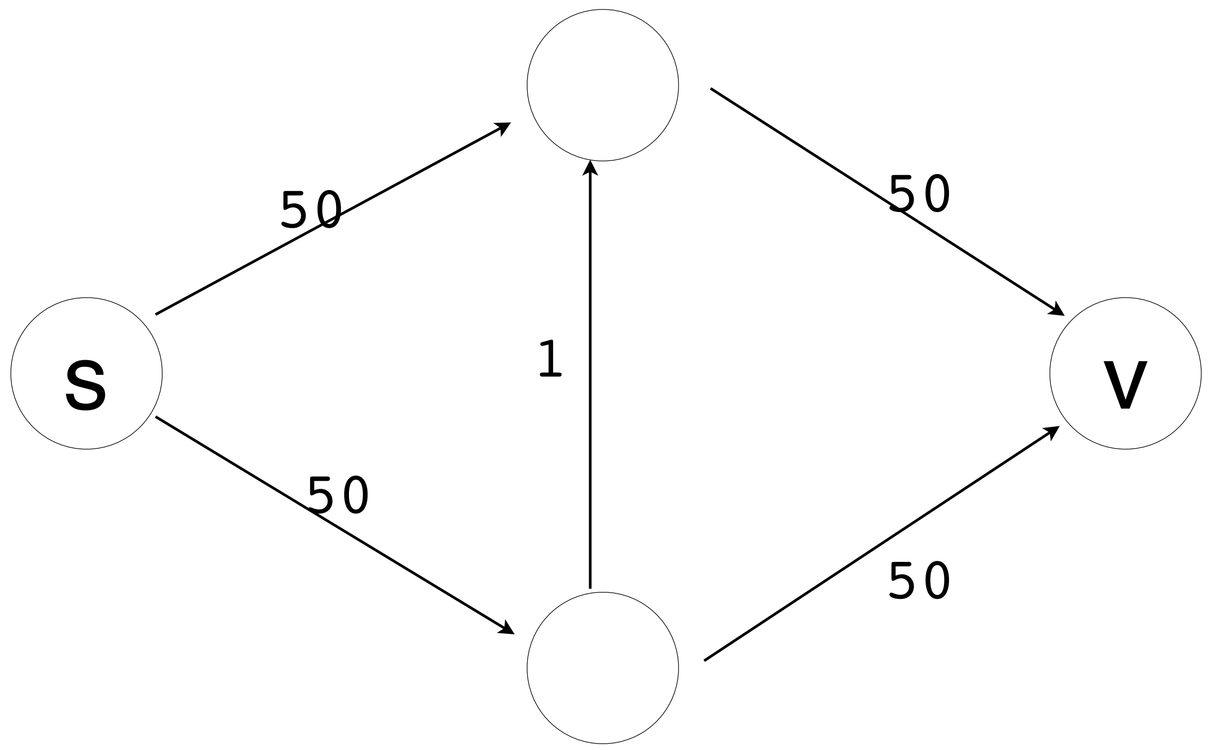
50
50
1
50
50
S
V



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


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

root of the problem



edmonds-karp 2

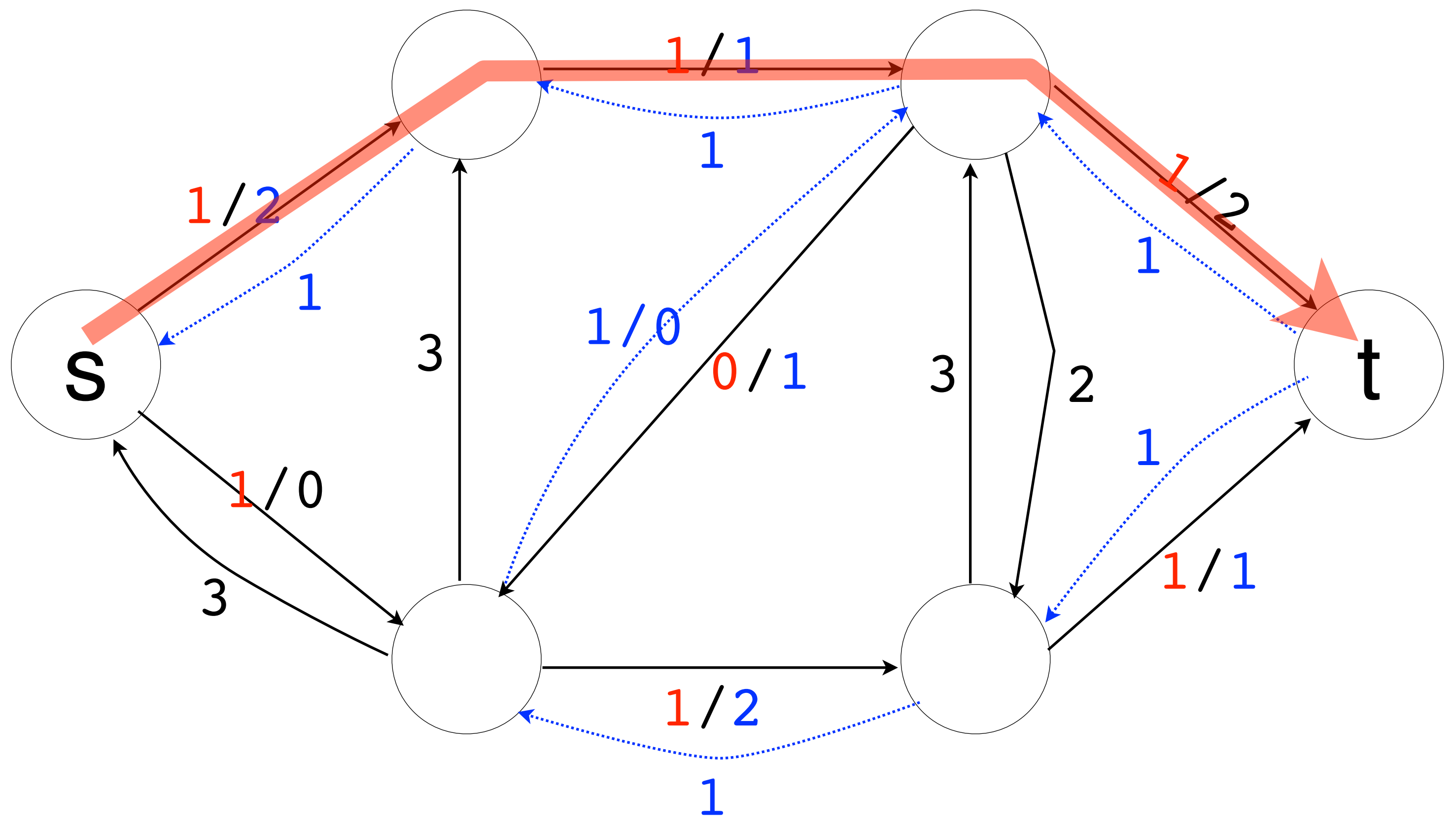
choose path with fewest edges first.

$\delta_f(s,v)$:

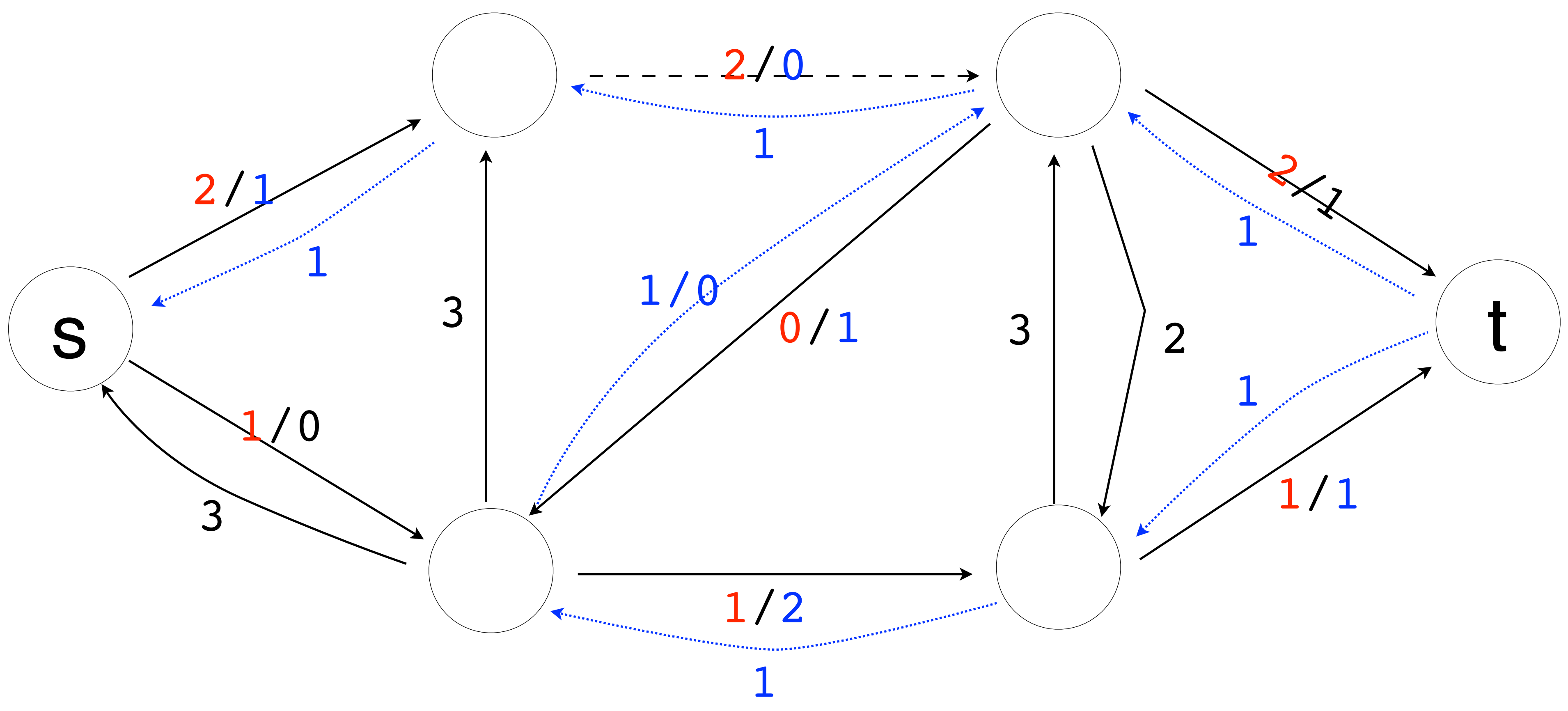
lemma:

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

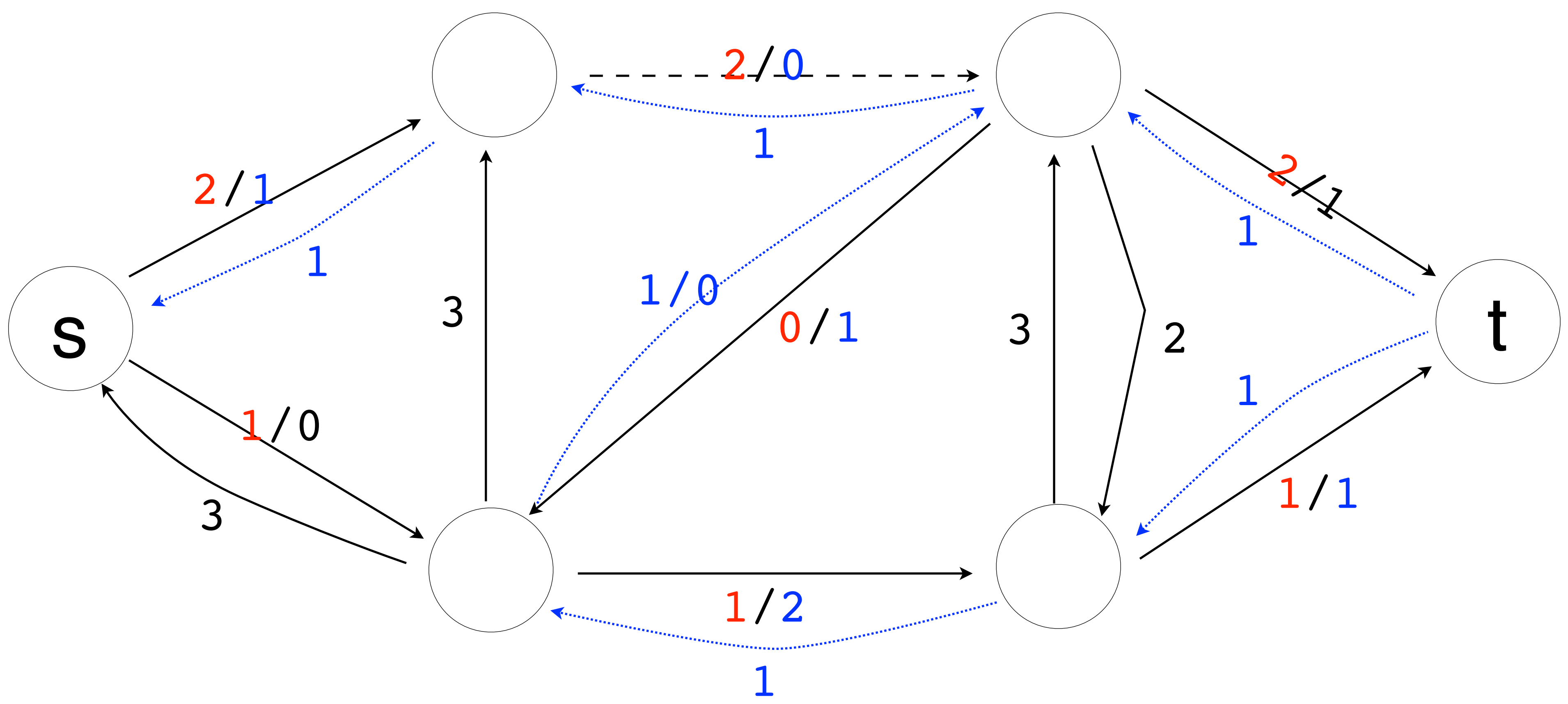
$\delta_{i+1}(v) \geq \delta_i(v)$



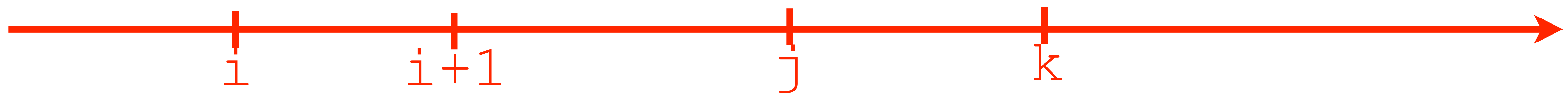
for every augmenting path, some edge is critical.



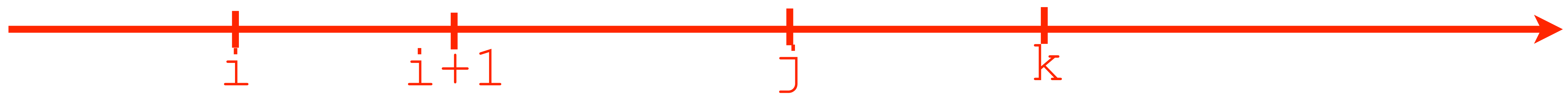
critical edges are removed in next residual graph.



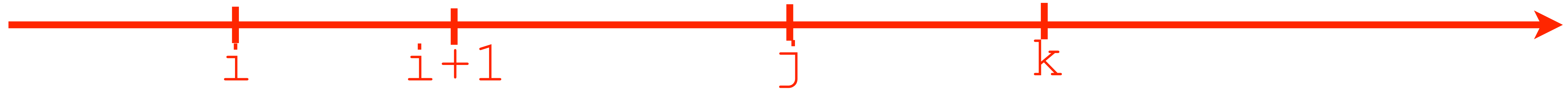
key idea: how many times can an edge be critical?



Outline of the argument



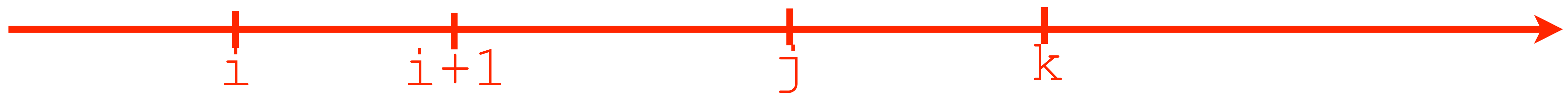
first time (u,v) is critical:



time i: (u,v) is critical:

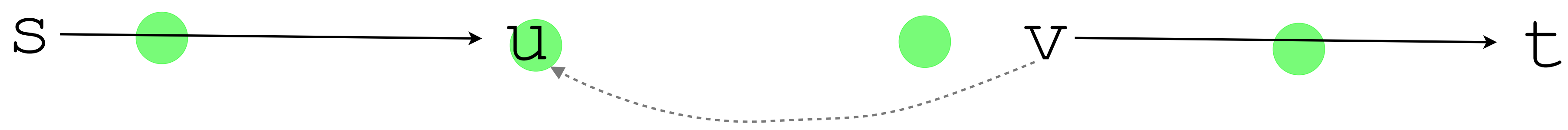
$$\delta_{i+1}(s,v) \geq \delta_i(s,v) + 1$$

time j: Edge (u,v) STRIKES BACK

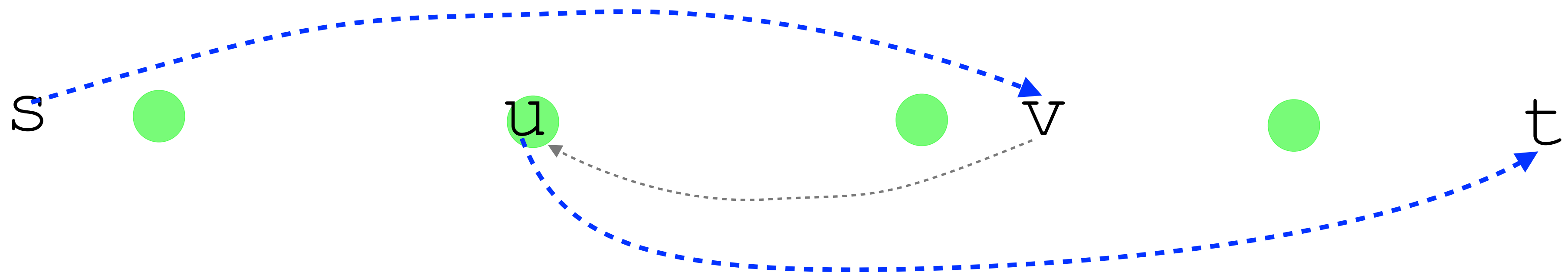


time i: (u,v) is critical:

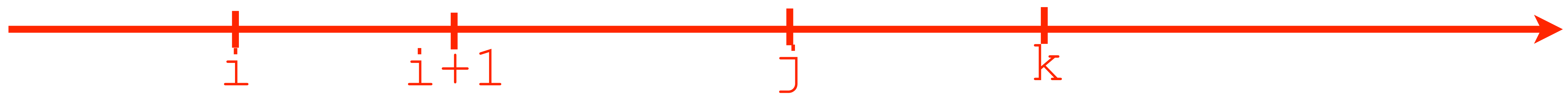
$$\delta_{i+1}(s,v) \geq \delta_i(s,v) + 1$$



time j: Edge (u,v) STRIKES BACK



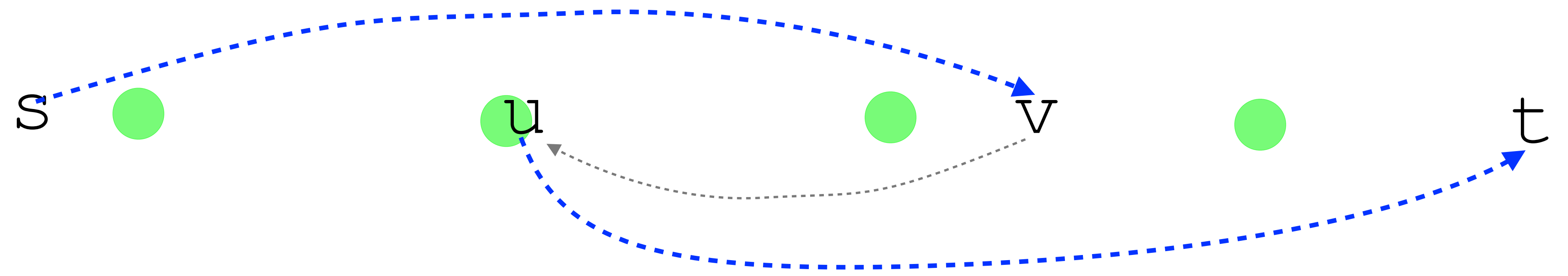
$$\delta_j(s,u) = \delta_j(s,v) + 1$$

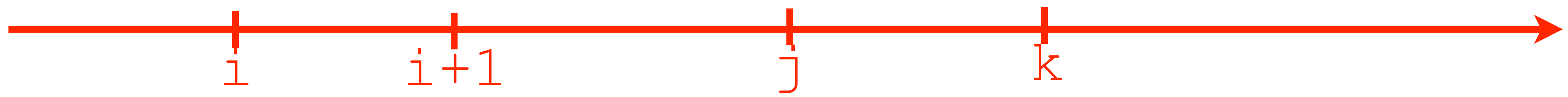


time j: Edge (u,v) STRIKES BACK

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

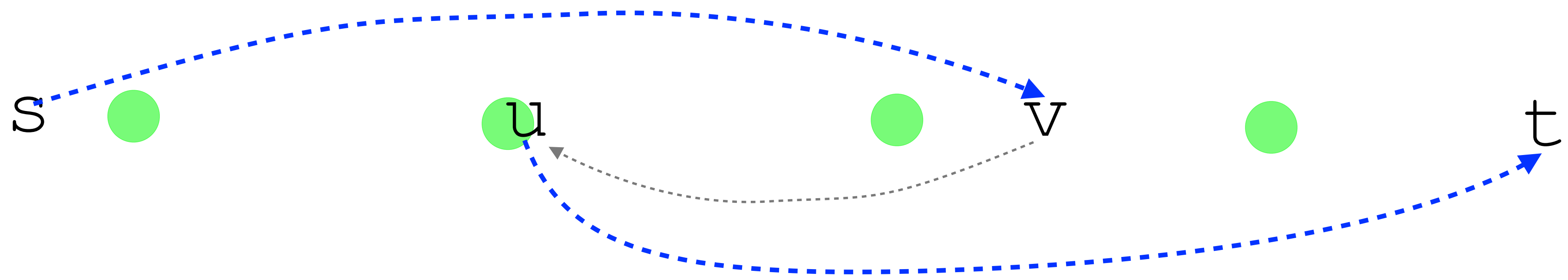
$$\delta_j(s, u) = \delta_j(s, v) + 1$$





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?

edge critical only times.

there are only edges.

ergo, total # of augmenting paths:

time to find an augmenting path:

total running time of E-K algorithm:

ff	$O(E\|f^*\|)$
ek2	
push-relabel	
faster push-relabel	