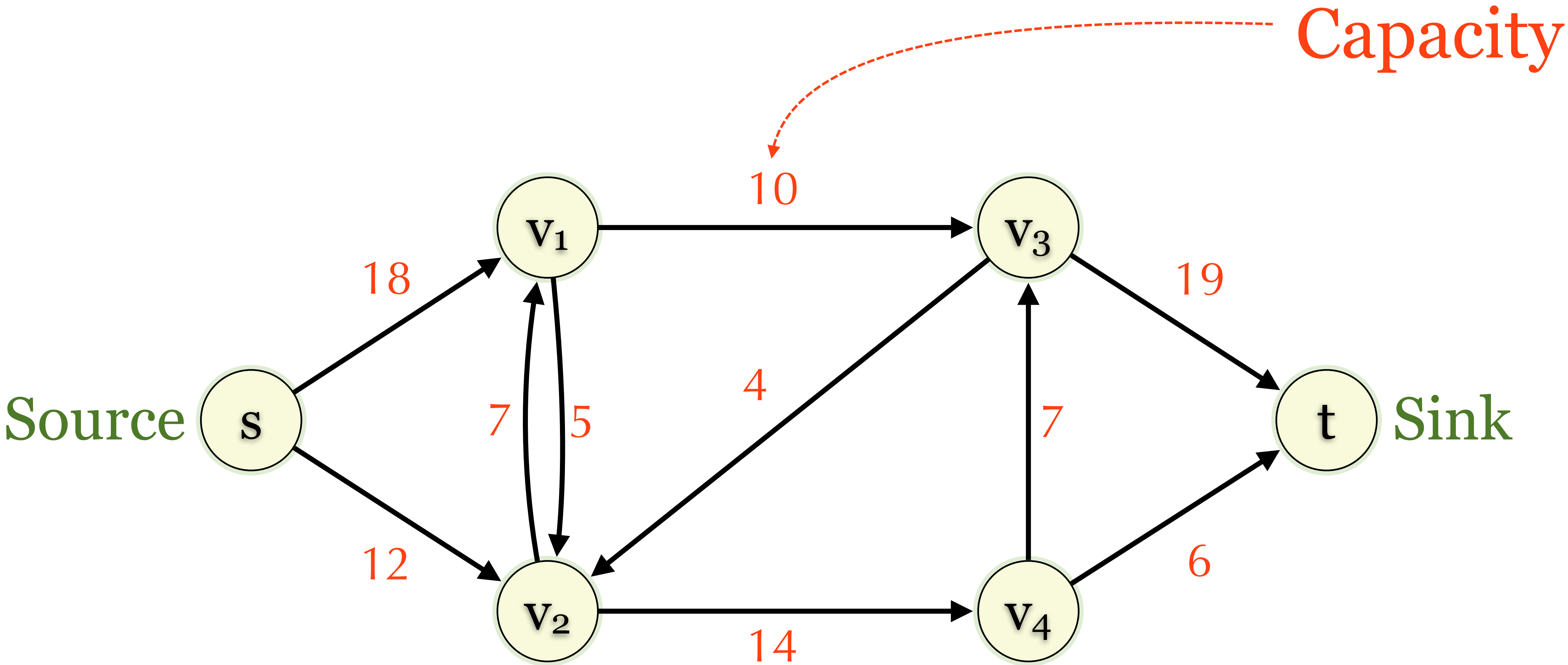
Flow Networks: Algorithms

Flow Networks

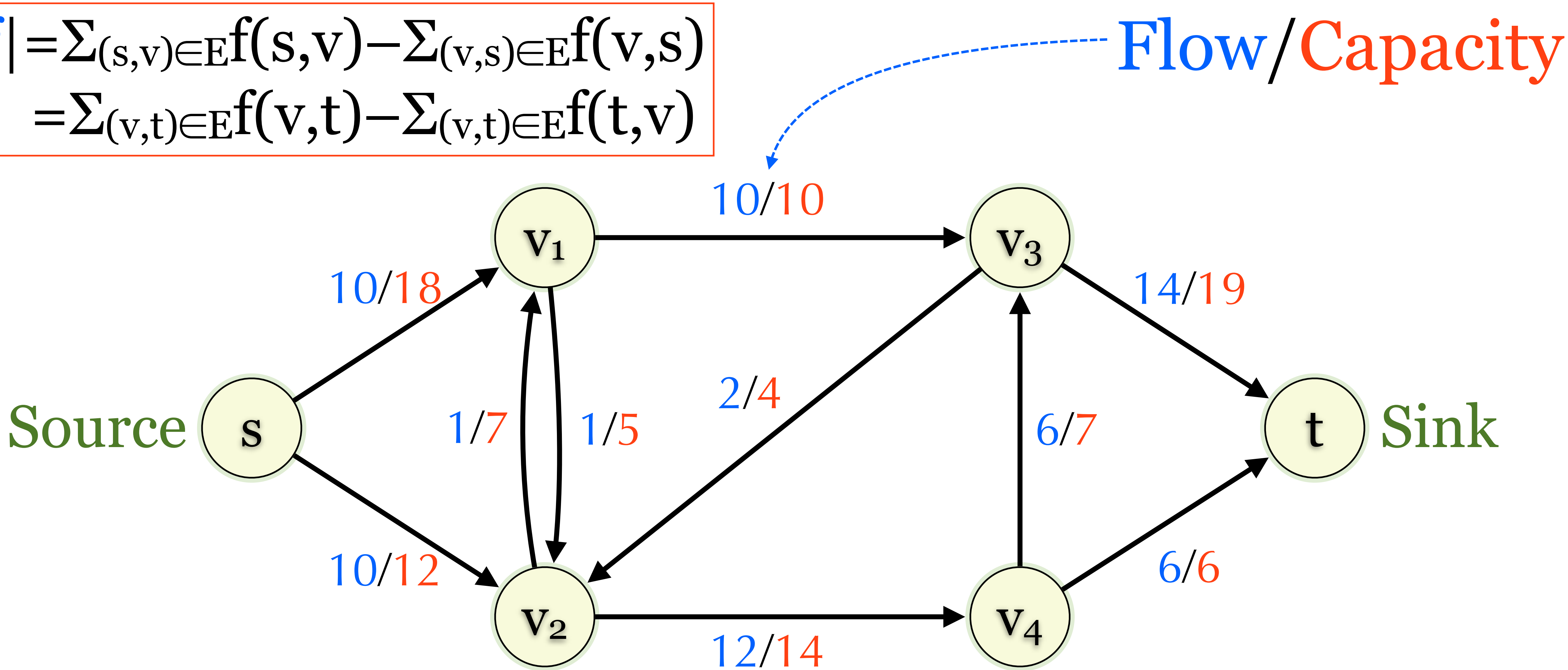
- A network is a directed graph $G=(V,E)$
  - Capacity function $c: E \rightarrow R^+$
  - Cost function $w: E \rightarrow R$ (optional)
- A flow from s to t is a function $f_{s,t}: E \rightarrow R^+$
  - Capacity constraint: $0 \leq f_{s,t}(e) \leq c(e)$
  - Flow conservation: $\forall\, v \in V \setminus \{s,t\}$, $\Sigma_{(u,v)\in E} f_{s,t}(u,v) = \Sigma_{(v,w)\in E} f_{s,t}(v,w)$.

Example: Network



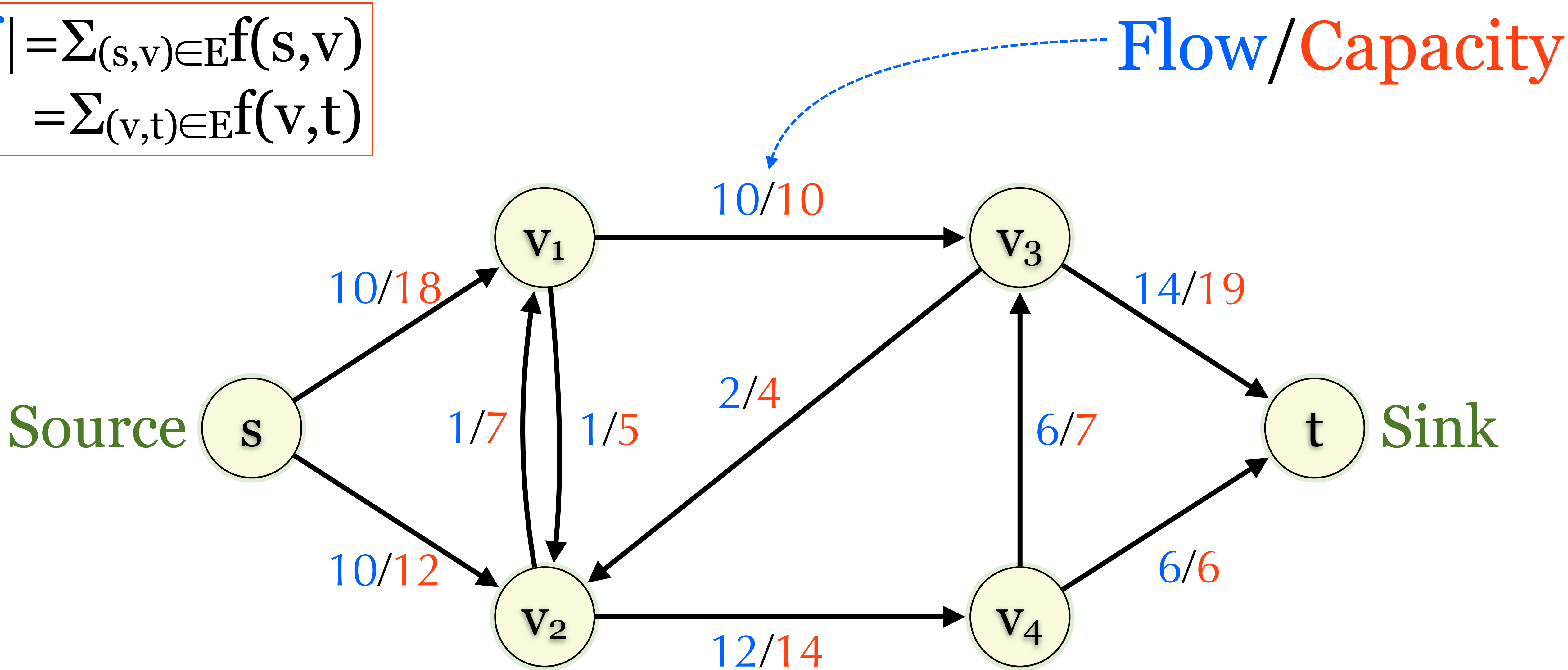
Example: Flow

$$|f| = \Sigma_{(s,v)\in E} f(s,v) - \Sigma_{(v,s)\in E} f(v,s)$$
$$= \Sigma_{(v,t)\in E} f(v,t) - \Sigma_{(v,t)\in E} f(t,v)$$



Example: Flow

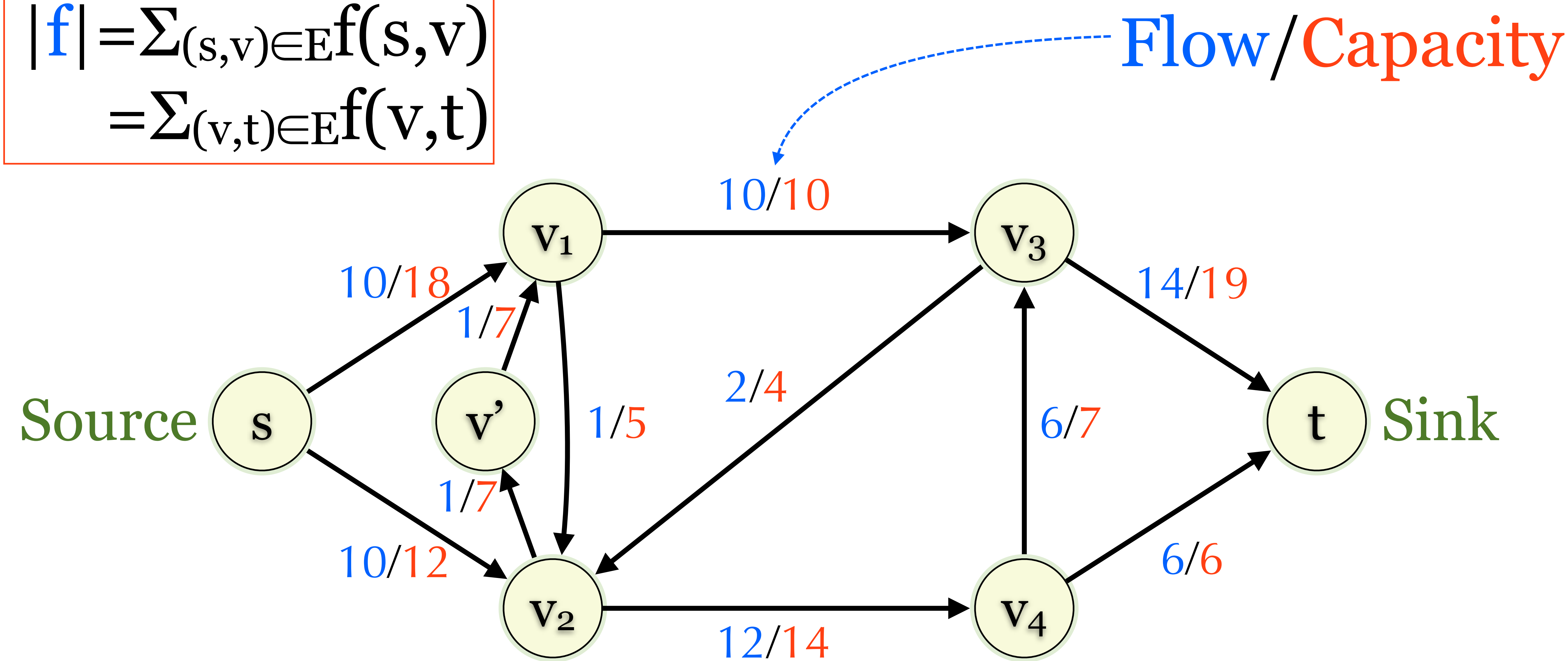
$$|f| = \Sigma_{(s,v)\in E} f(s,v) = \Sigma_{(v,t)\in E} f(v,t)$$



Remove edges incoming to s and outgoing from t

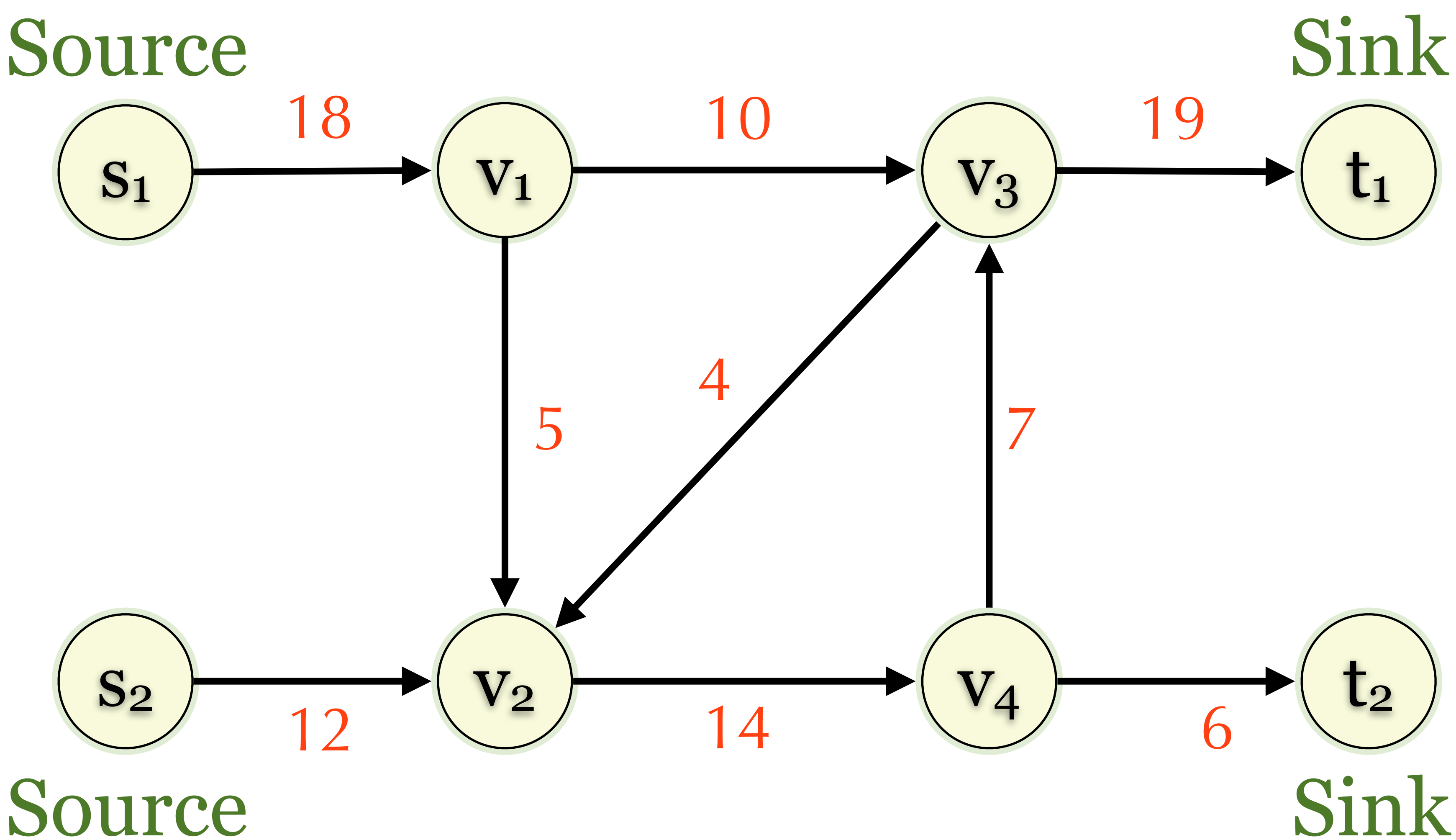
Removing Multiple Edges

$$|f| = \Sigma_{(s,v)\in E} f(s,v) = \Sigma_{(v,t)\in E} f(v,t)$$

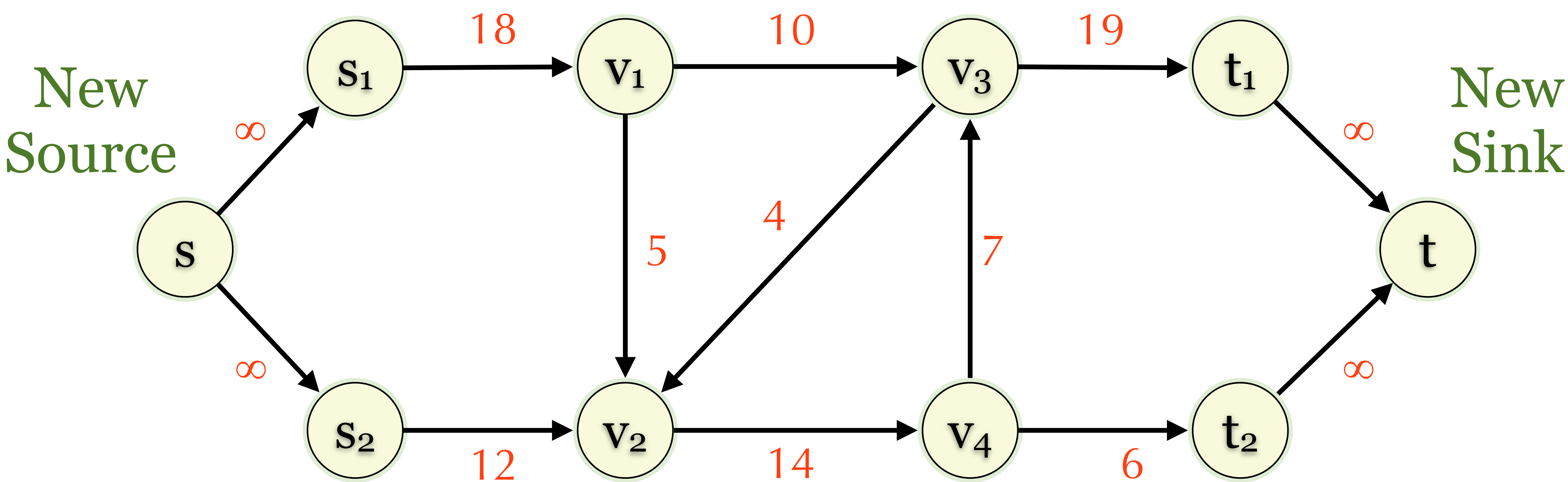


We may assume there is only one edge between any two vertices.

Multiple Sources & Sinks



Multiple Sources & Sinks

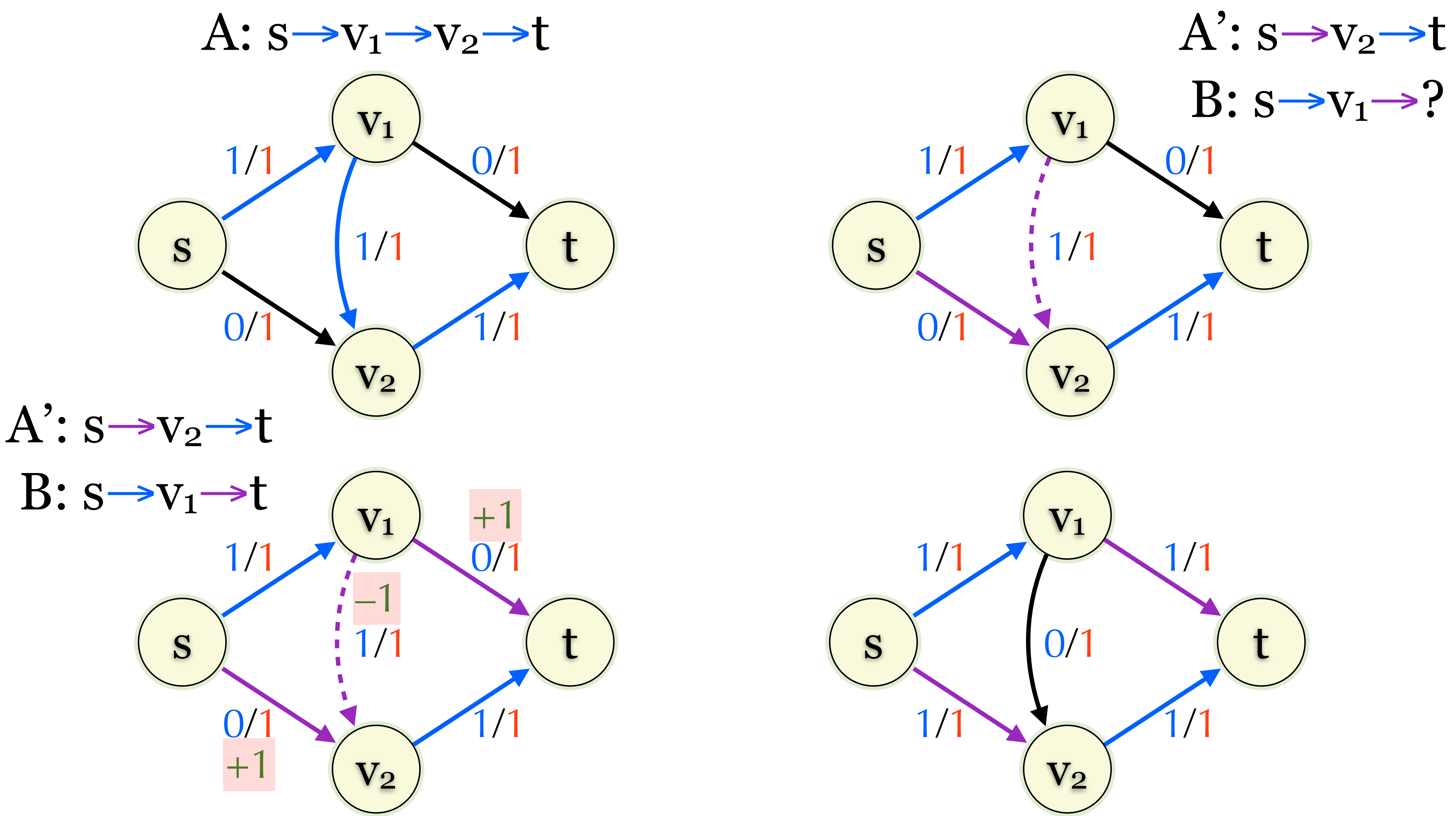


Focus on single source and single sink instances.

Outlines

- Maximum flow
  - Ford-Fulkerson
  - Edmonds-Karp
  - Minimum cut & Applications
- Minimum cost flow
  - Successive shortest path
  - Minimum mean cycle canceling

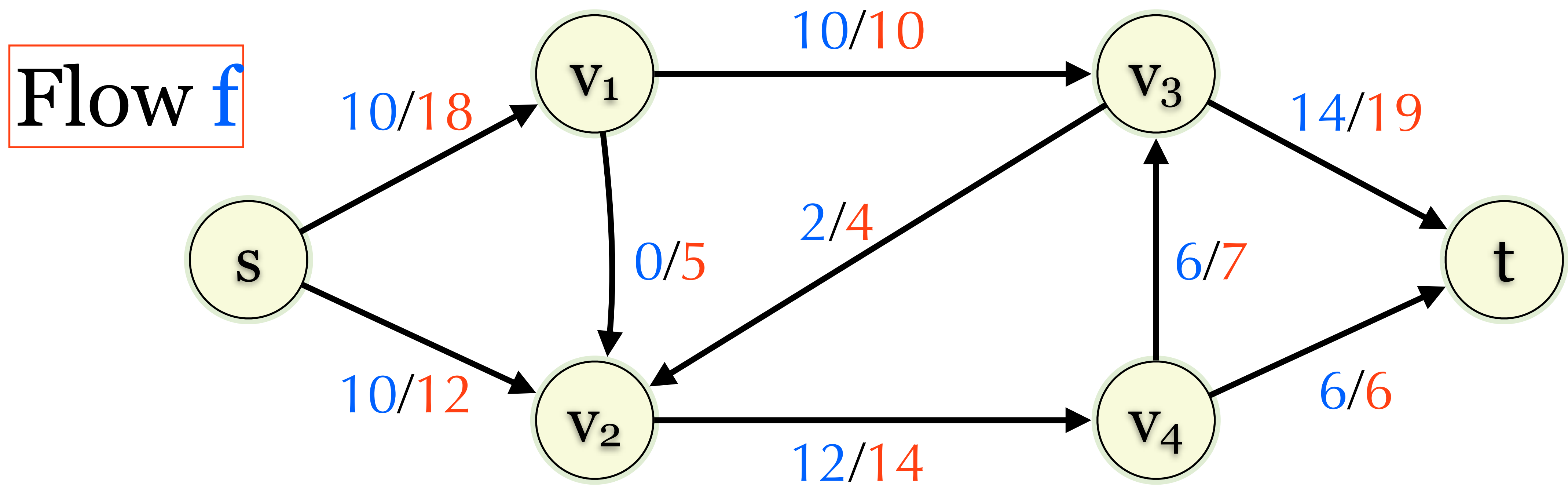
Cancellation

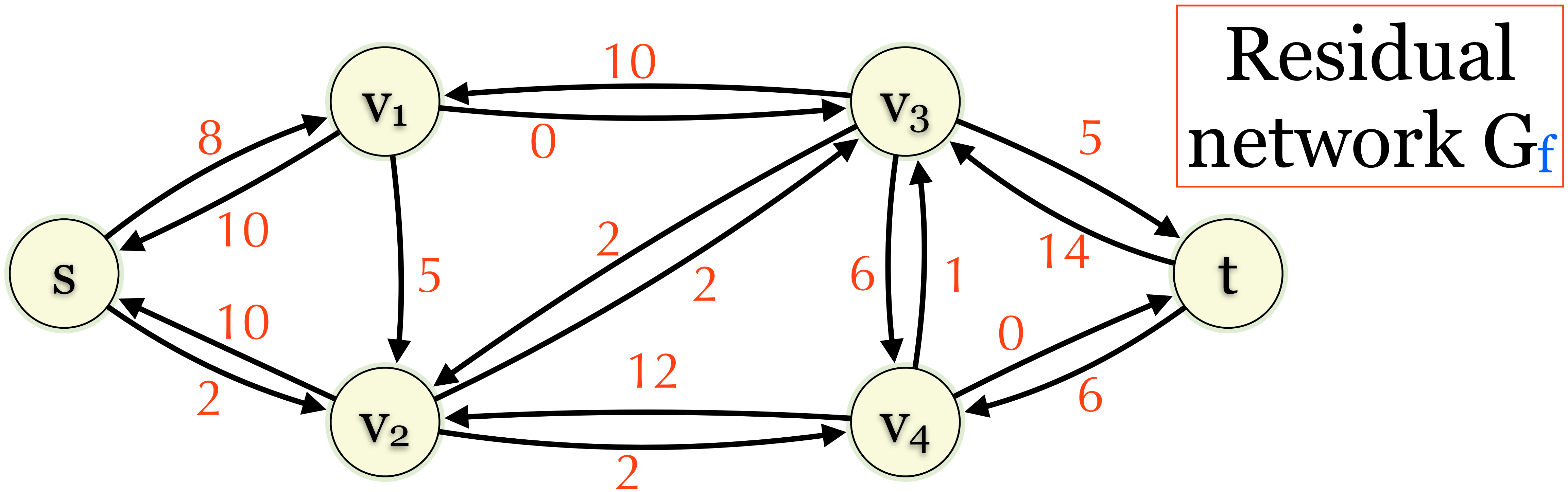


Residual Network

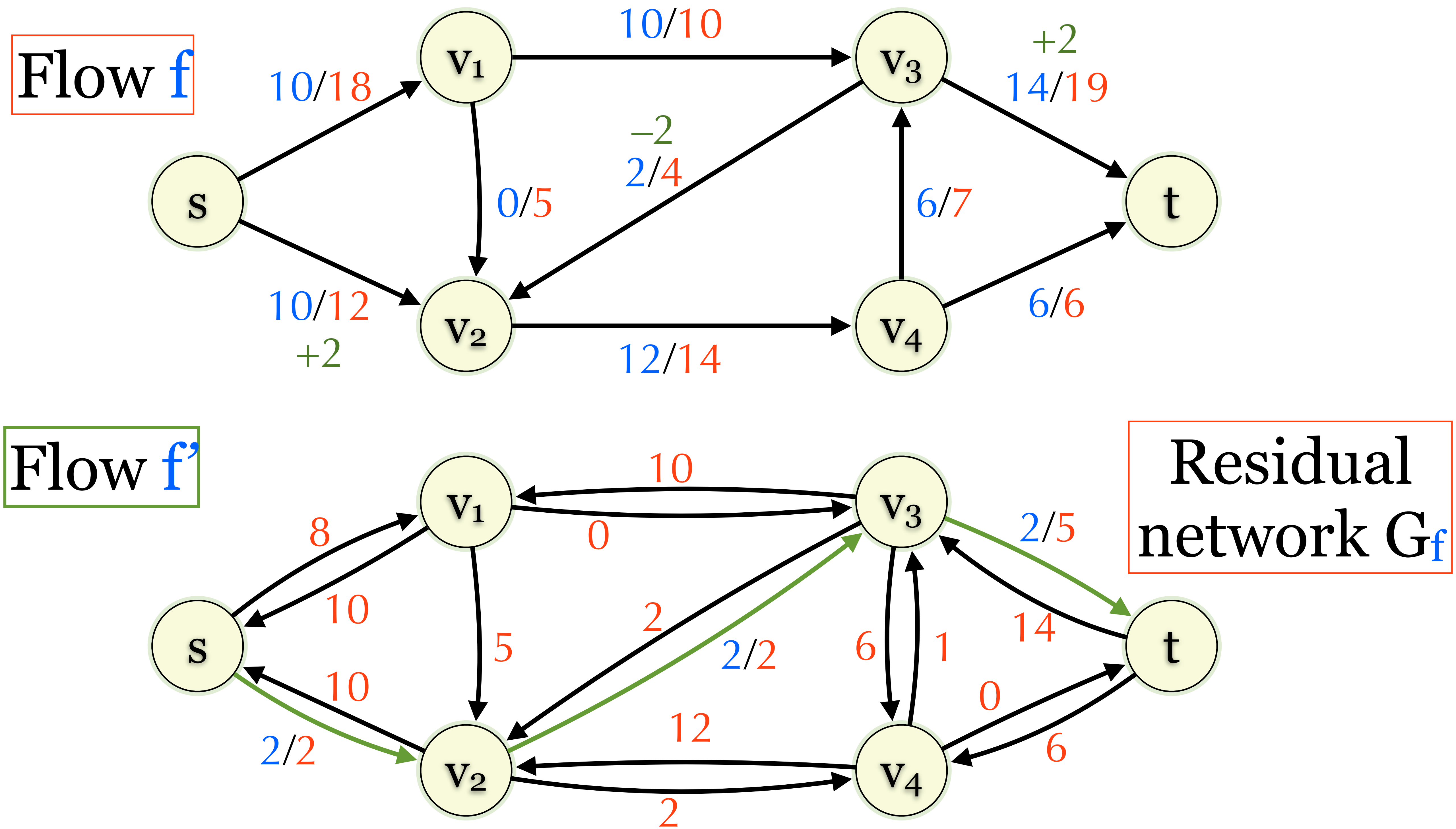
- For each edge $(u,v)\in E$ s.t. $f(u,v)>0$, build edges (u,v), $(v,u)\in E_f$ s.t.
  - $c_f(u,v)=c(u,v)-f(u,v)$
  - $c_f(v,u)=c(v,u)+f(u,v)$ $c(v,u)=0$ if $(v,u)\notin E$
- Residual network G_f is (V,E_f) with capacity c_f.
- Flow f' in residual network:
  - $0\leq f'(u,v)\leq c_f(u,v)$
  - For $v\in V$, $\Sigma_{u\in V}f'(u,v)=\Sigma_{w\in V}f'(v,w)$
- Augment: $(f\uparrow f')(u,v)=f(u,v)+f'(u,v)-f'(v,u)$

Example


Flow f
s
v1
v2
v3
v4
t
10/18
10/12
10/10
0/5
2/4
12/14
6/7
14/19
6/6


Residual network Gf
s
v1
v2
v3
v4
t
8
10
10
2
10
0
5
2
2
12
2
6
1
5
14
0
6

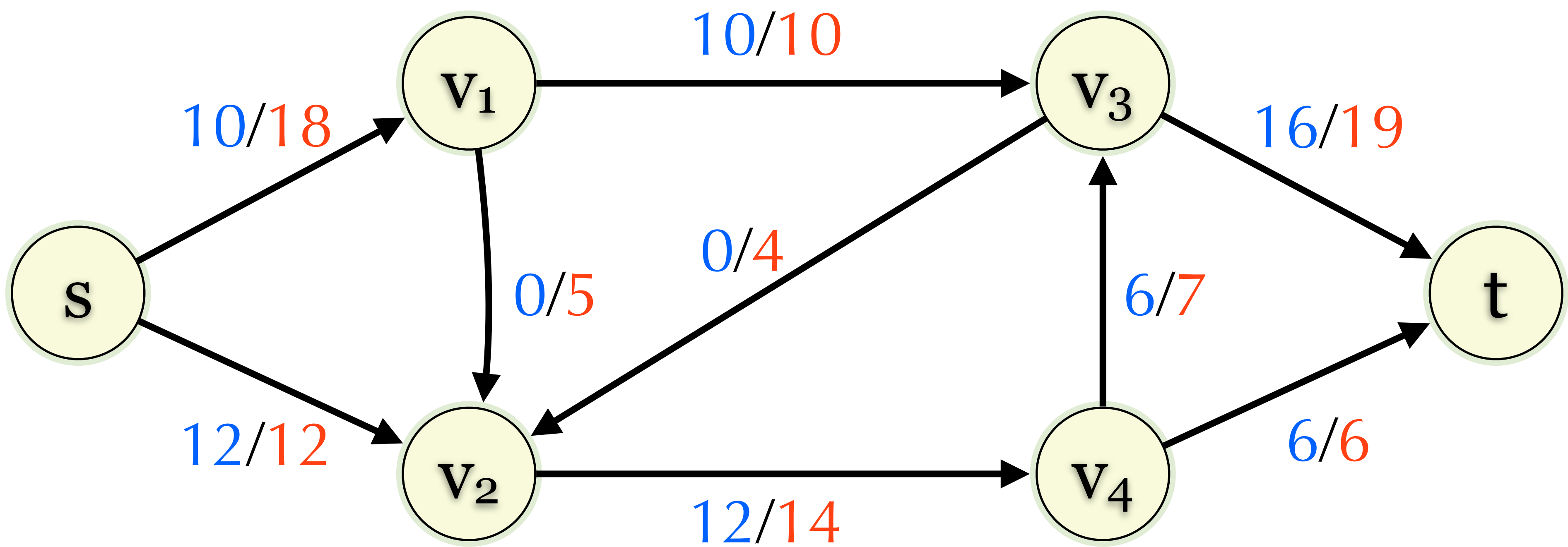
Example



Flow f
s
v1
v2
v3
v4
t
10/18
10/10
+2
14/19
–2
2/4
0/5
6/7
10/12
+2
12/14
6/6
Flow f'
Residual network Gf
8
10
0
10
5
2
2/2
6
1
14
2/5
10
2/2
12
0
6
2

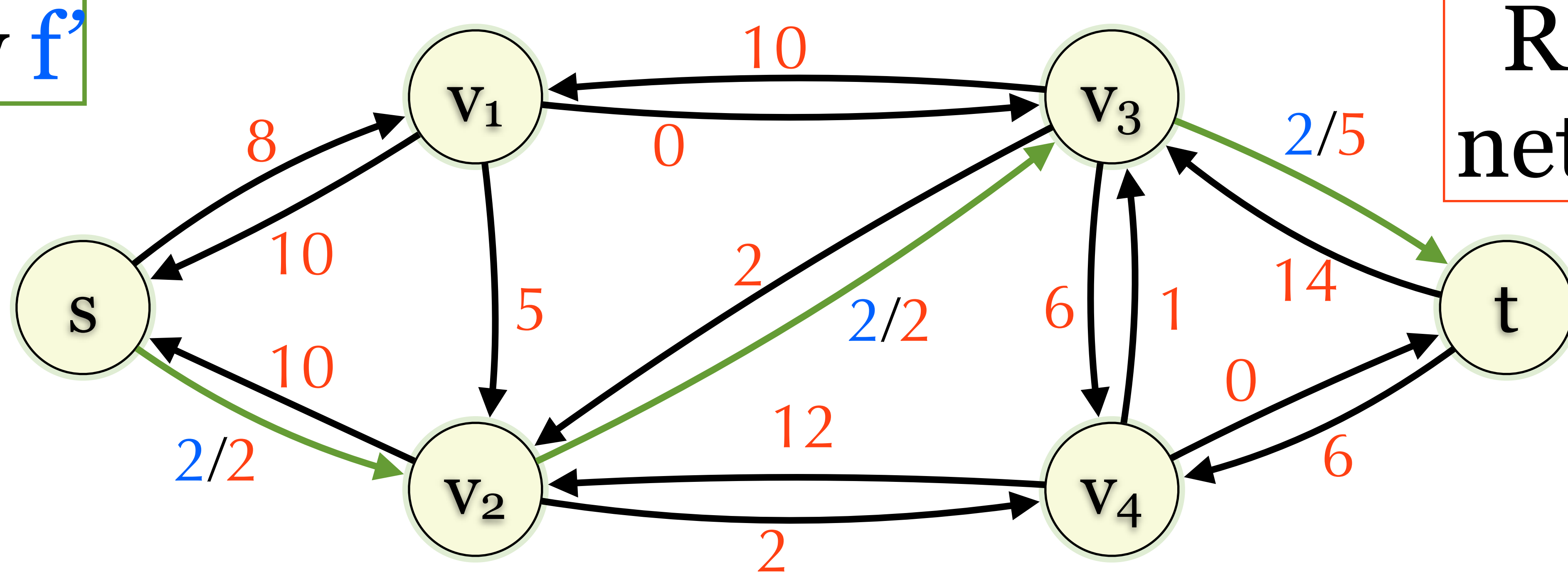
Example

$f \uparrow f'$



Flow f'

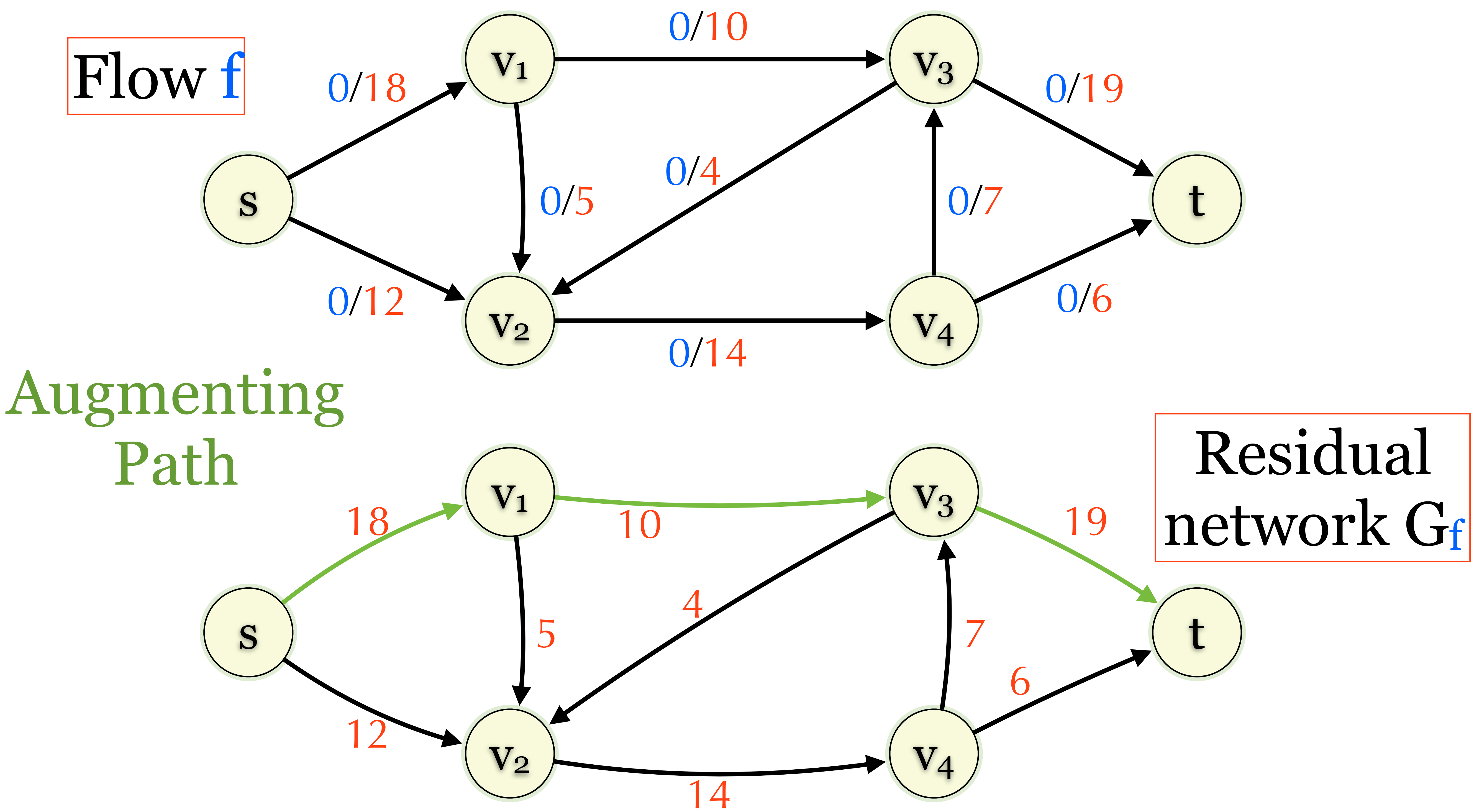
Residual network G_f



Ford-Fulkerson

- Repeat the following
  - Finding augmenting path p on residual network of f
    - Ignore e if $c_f(e)=0$
    - No augmenting path implies the flow is maximum
  - Augment flow f through the path p
    - $f'(u,v)=0$ if $(u,v)\notin p$
    - $f'(u,v)=\min_{e\in p}c_f(e)$ if $(u,v)\in p$

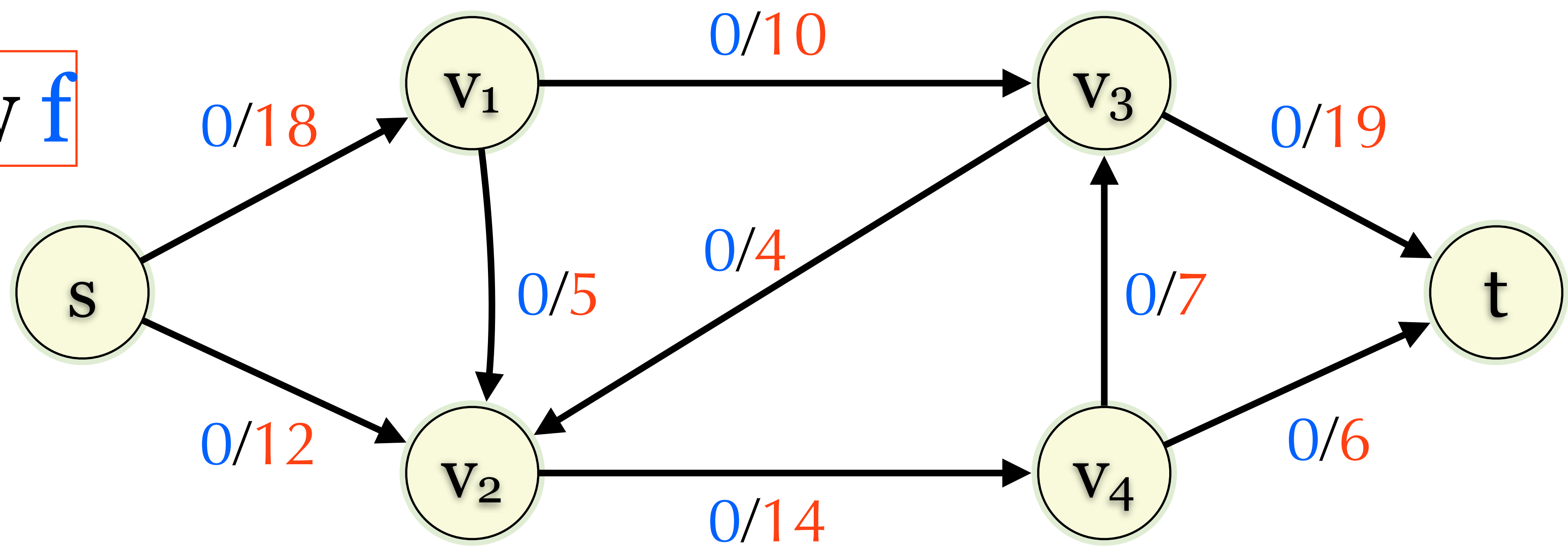
Example



Flow f
0/18
0/10
0/19
0/5
0/4
0/7
0/12
0/14
0/6
s
v1
v2
v3
v4
t
Augmenting Path
Residual network Gf
18
10
19
5
4
7
6
12
14

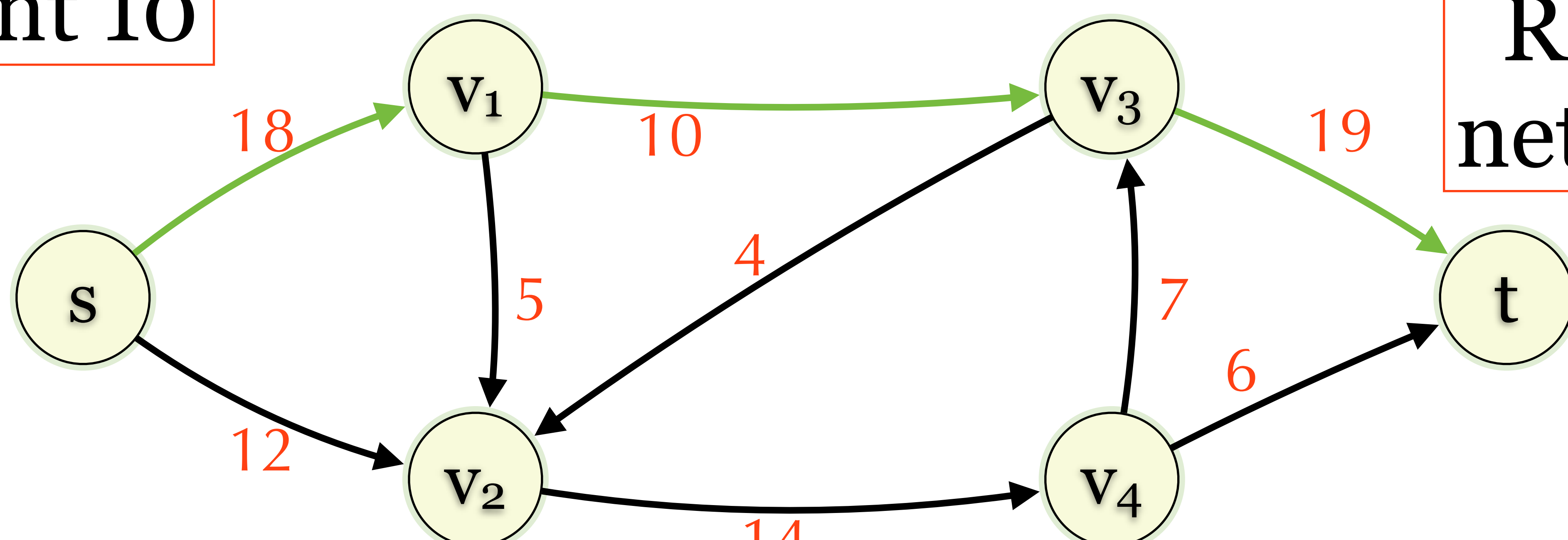
Example

Flow f



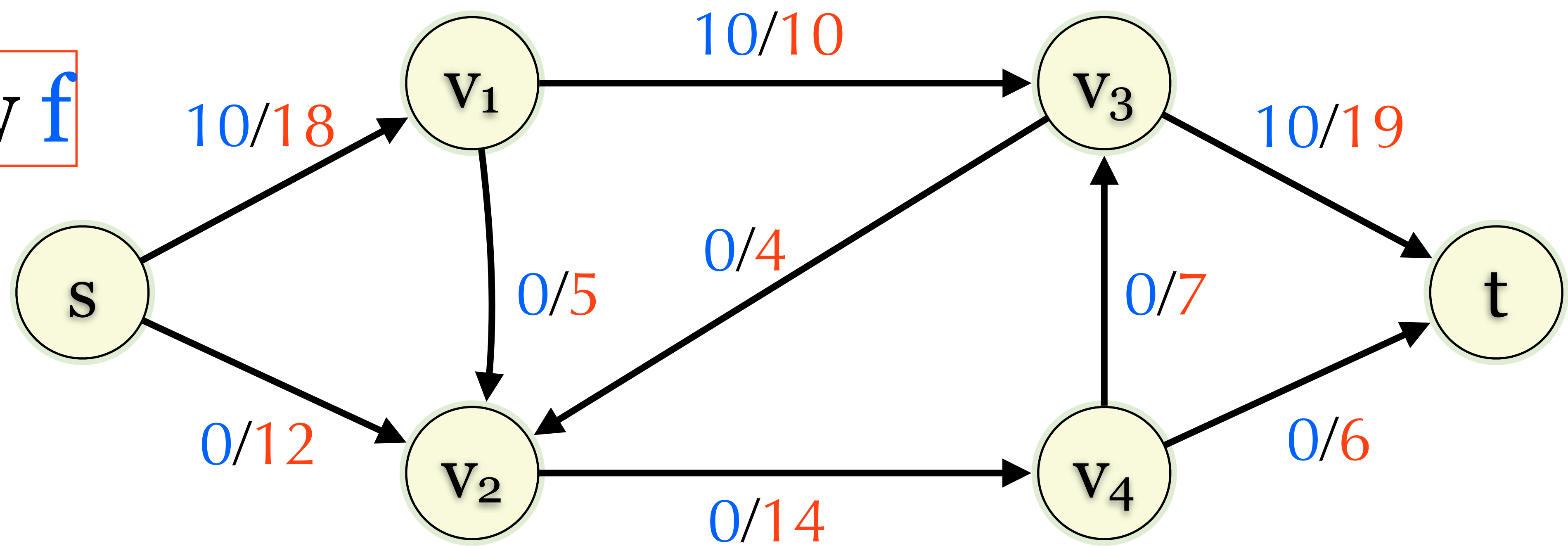
Augment 10

Residual network G_f



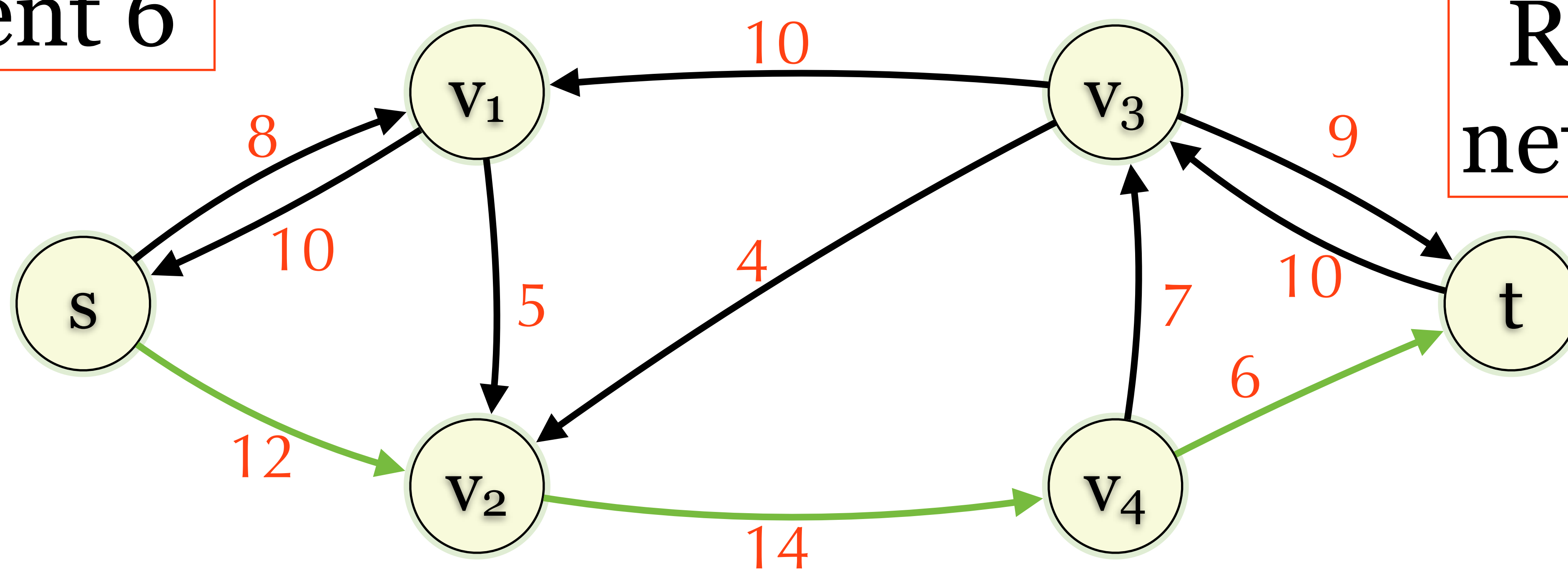
Example

Flow f



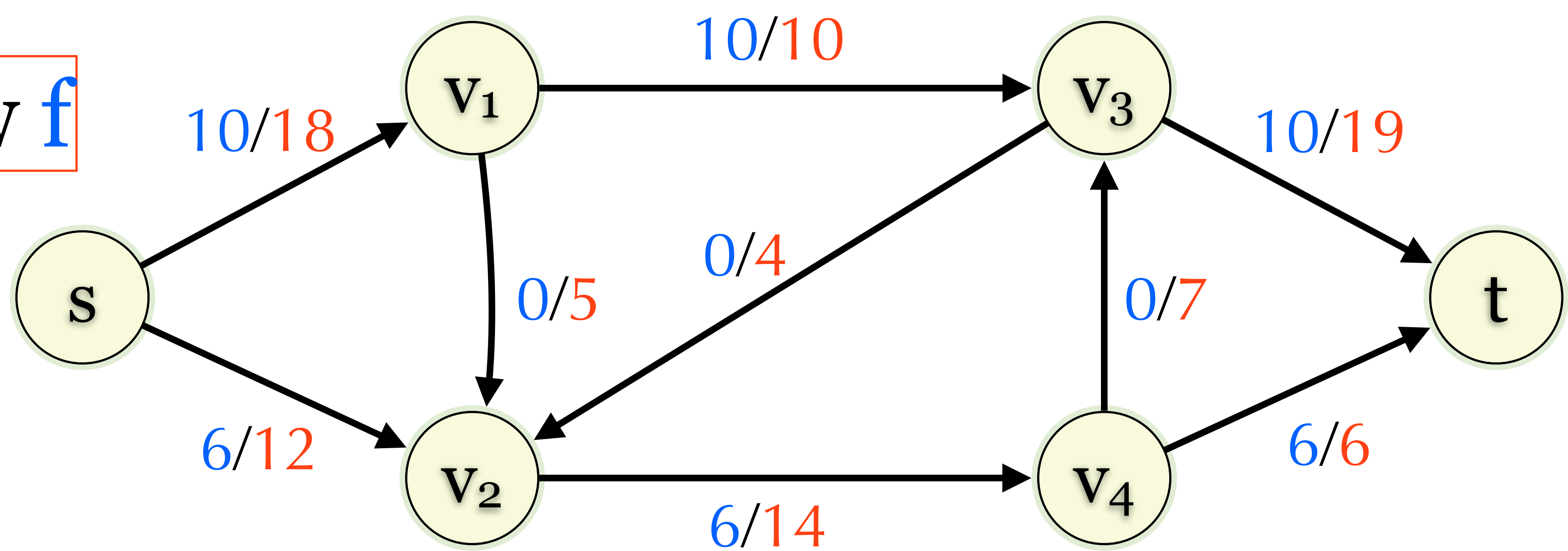
Augment 6

Residual network G_f



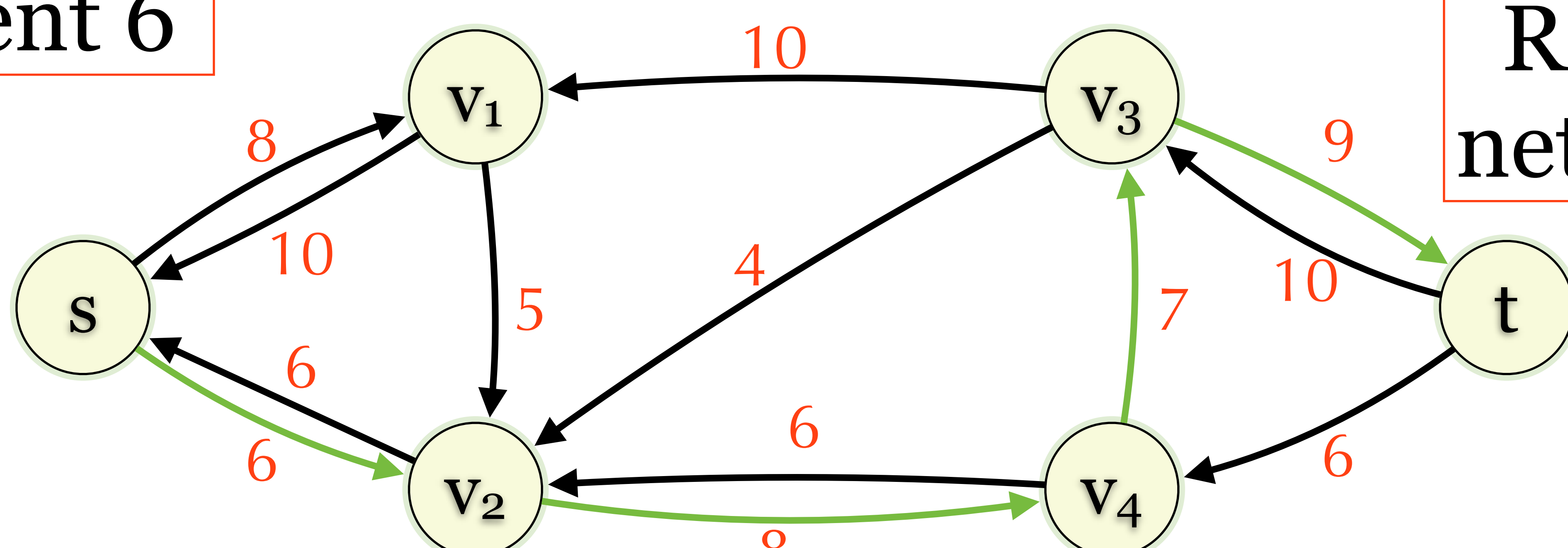
Example

Flow f



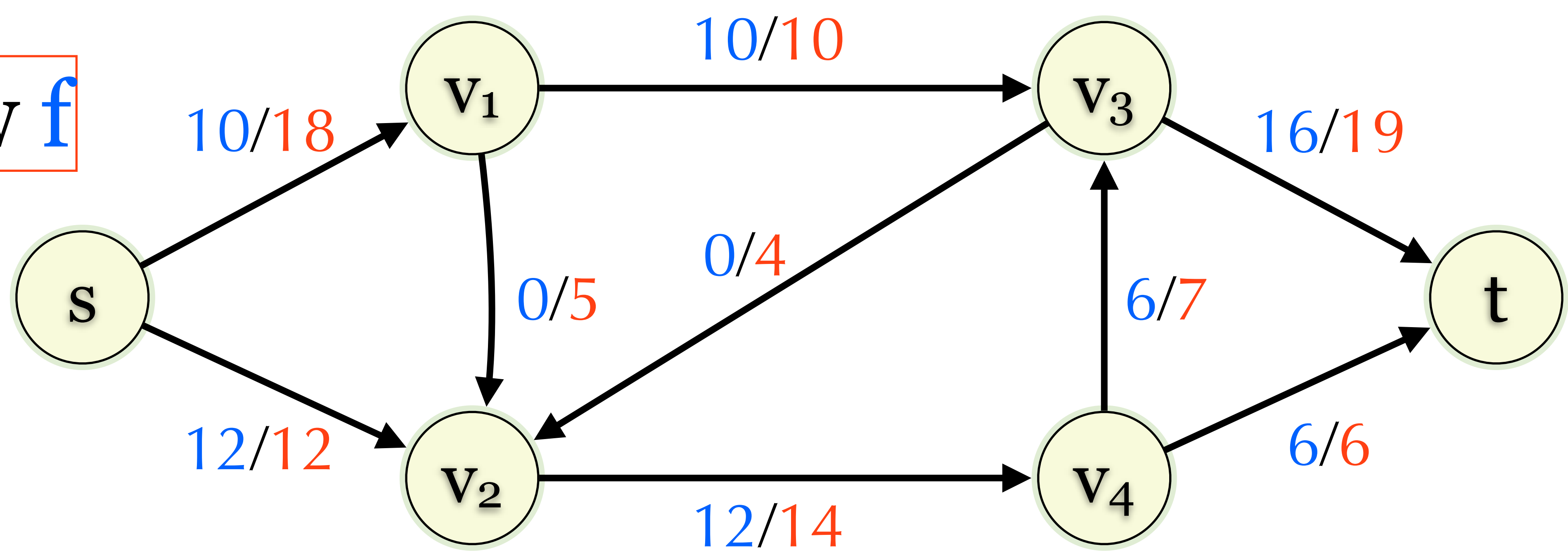
Augment 6

Residual network G_f



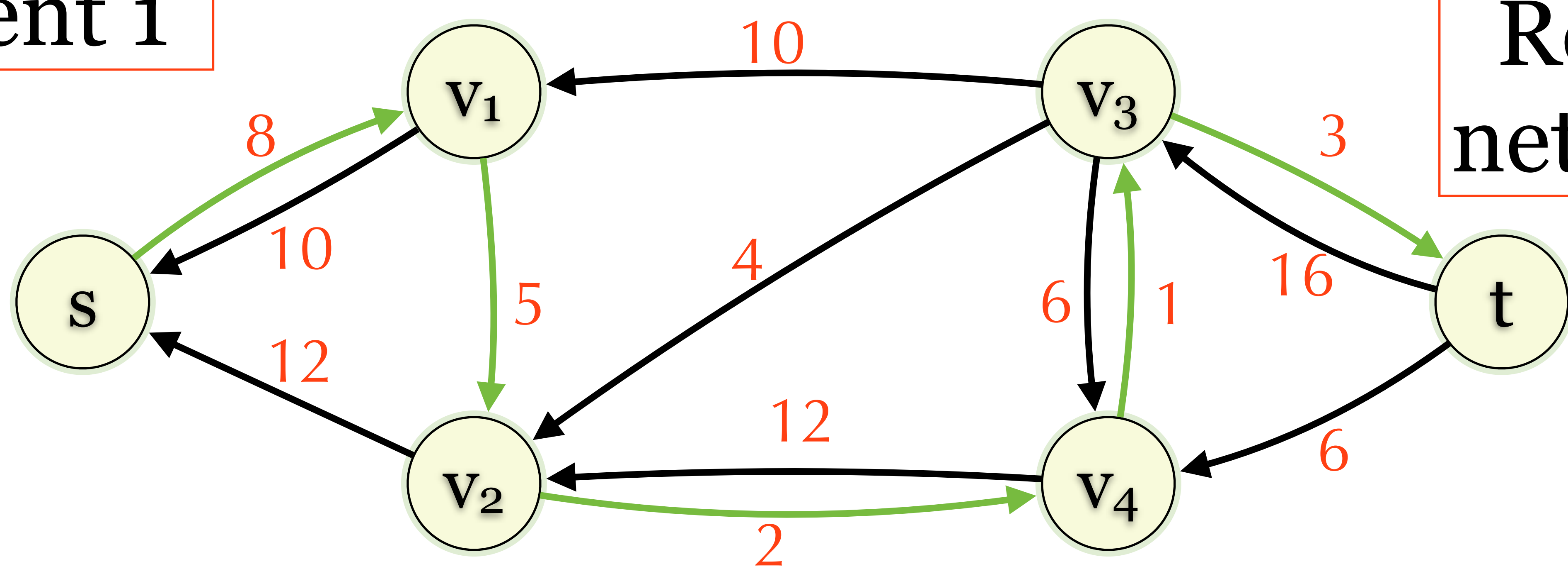
Example

Flow f



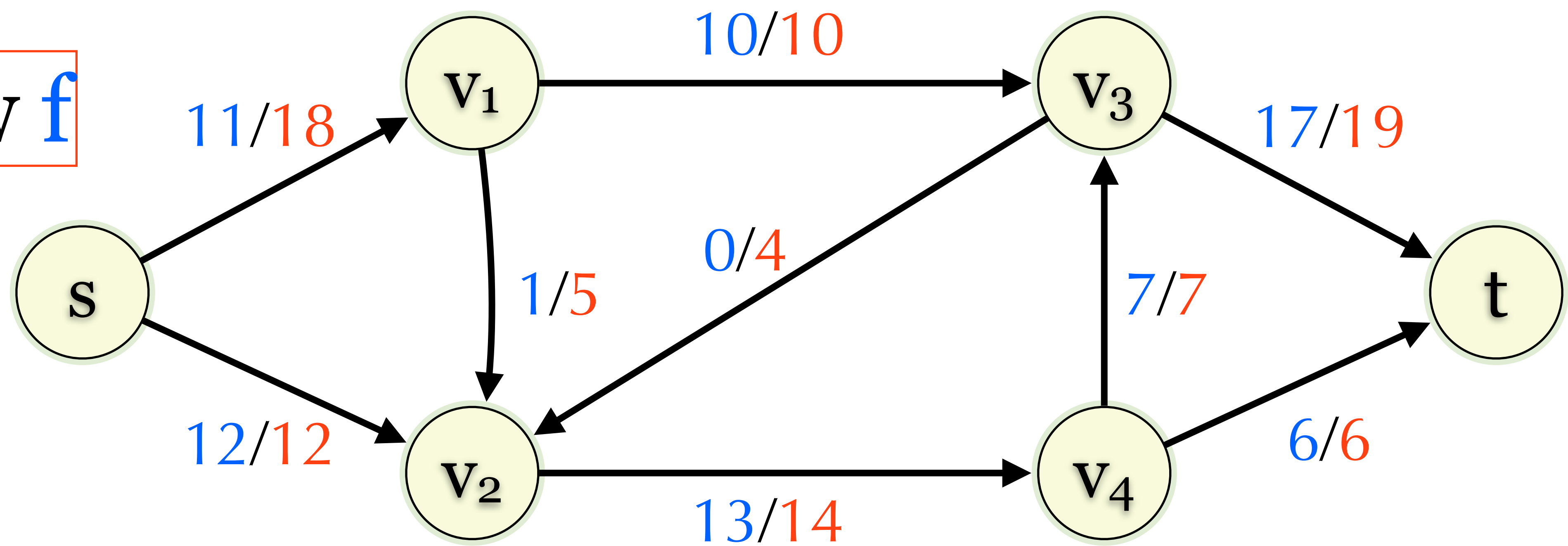
Augment 1

Residual network G_f



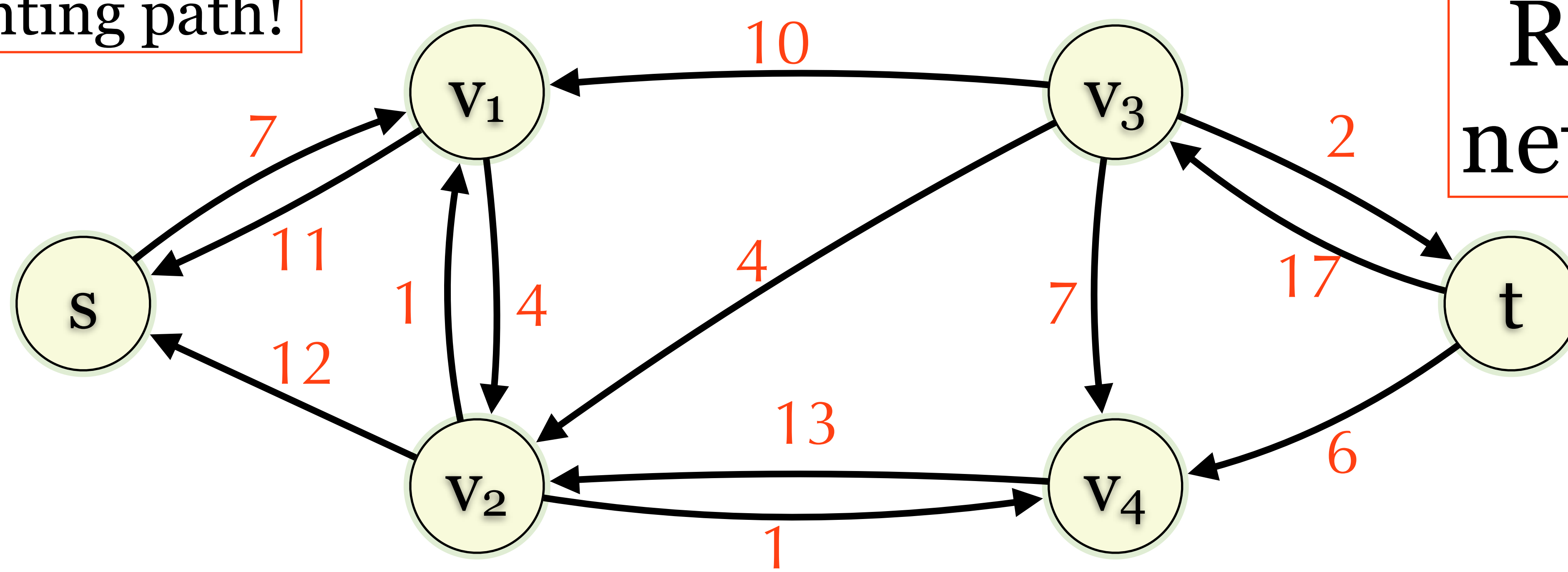
Example

Flow f

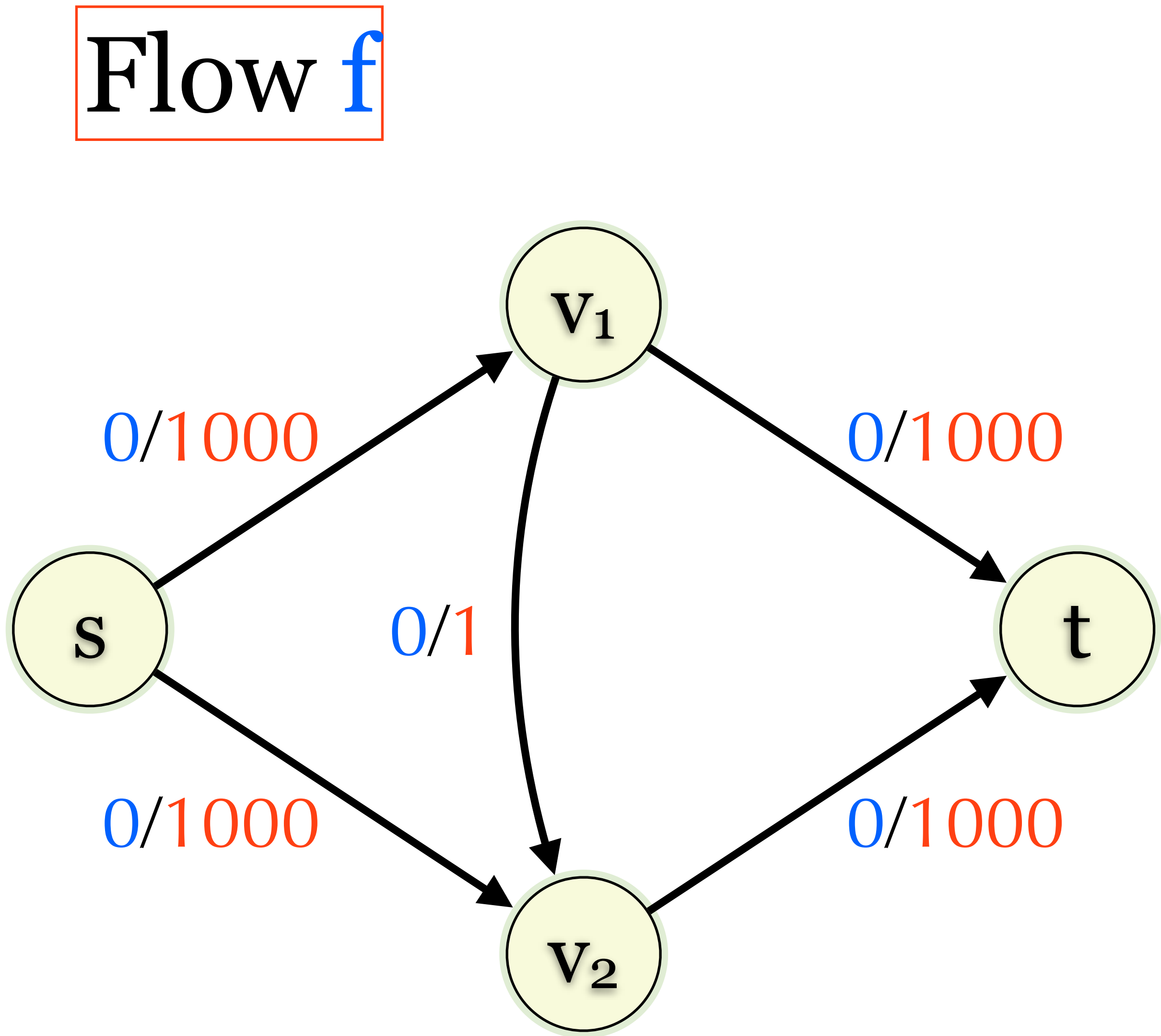


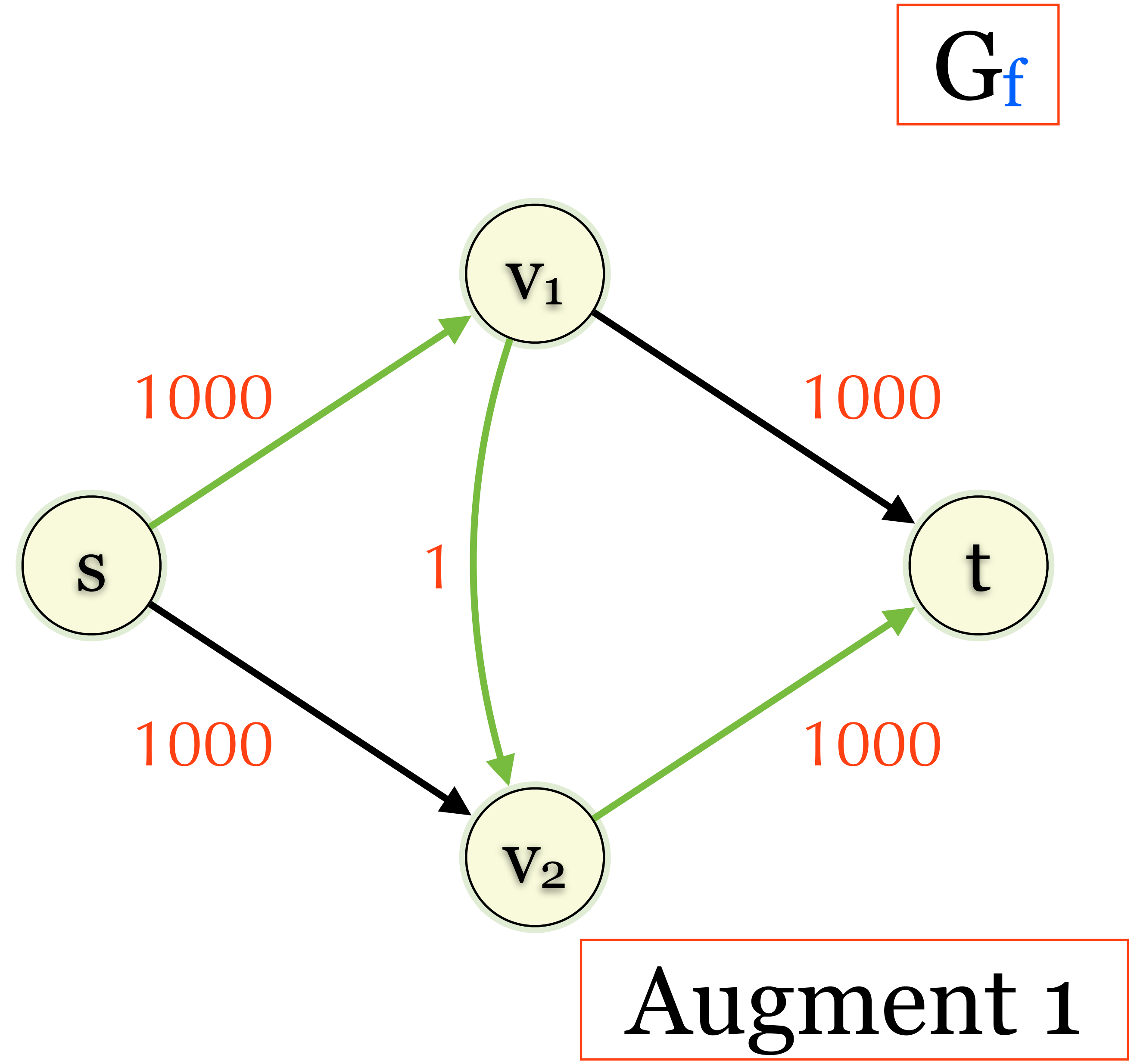
No augmenting path!

Residual network G_f



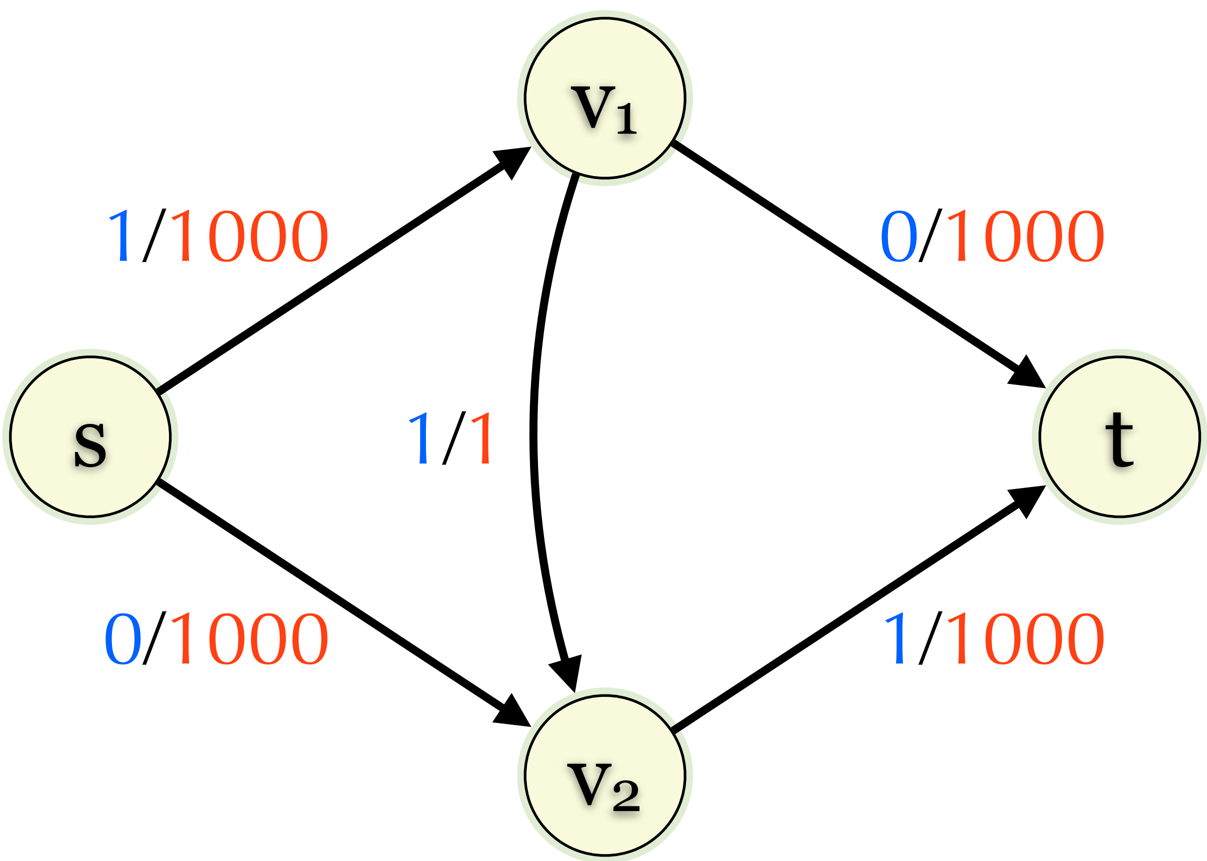
Bad Case for FF



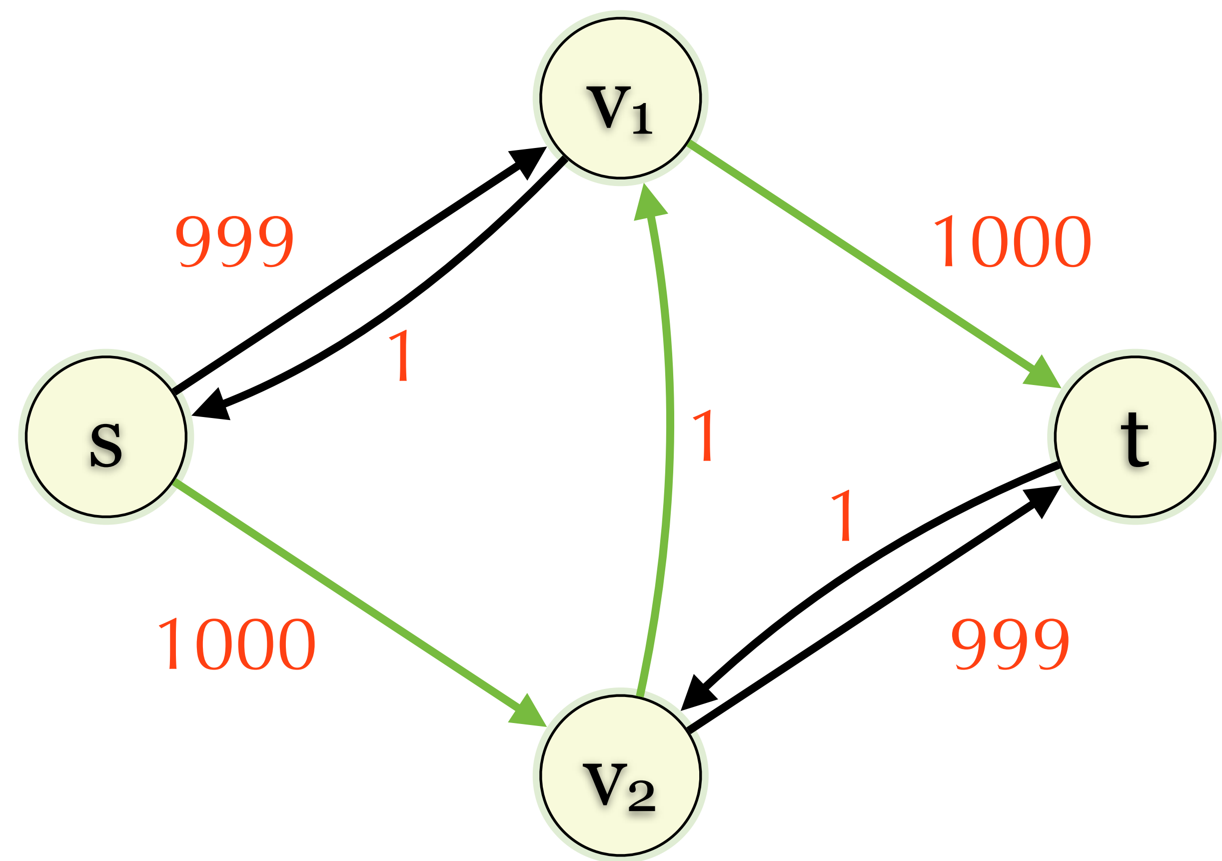


Bad Case for FF

Flow f



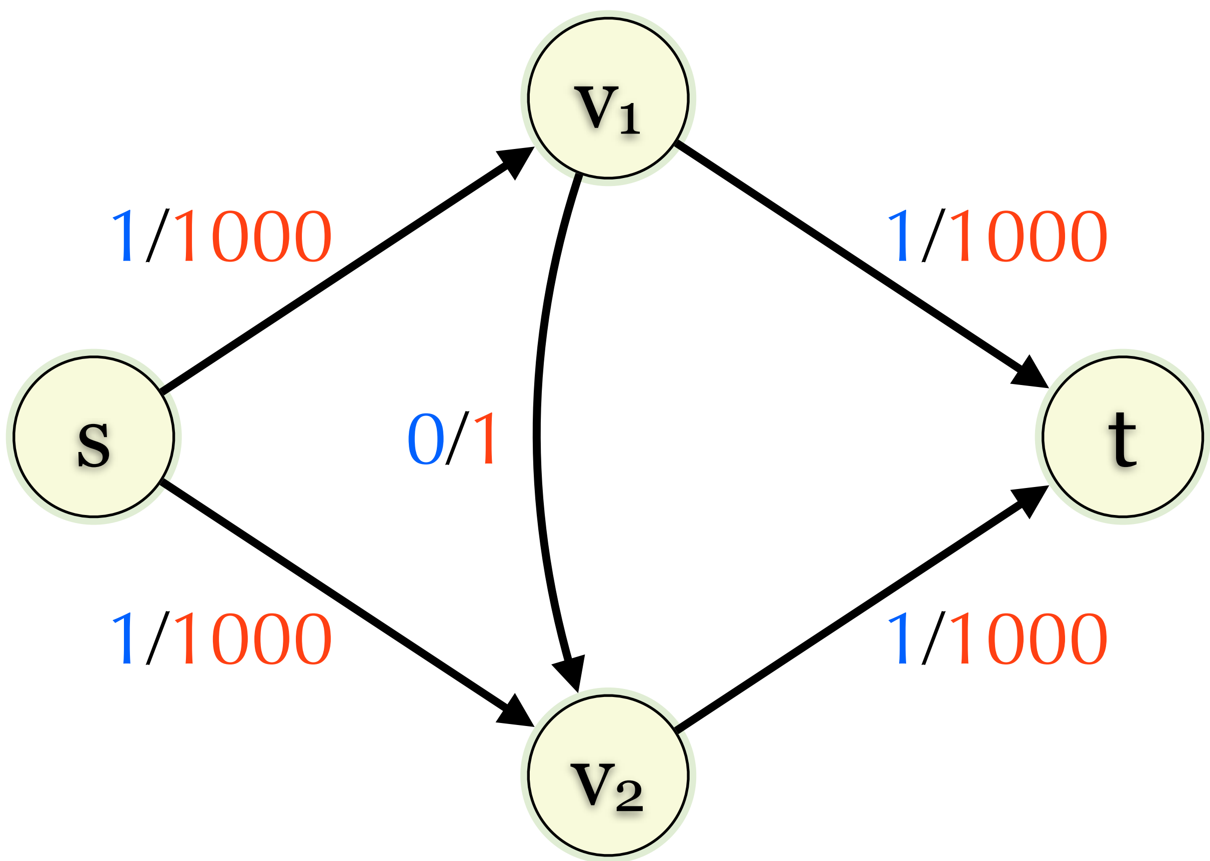
G_f



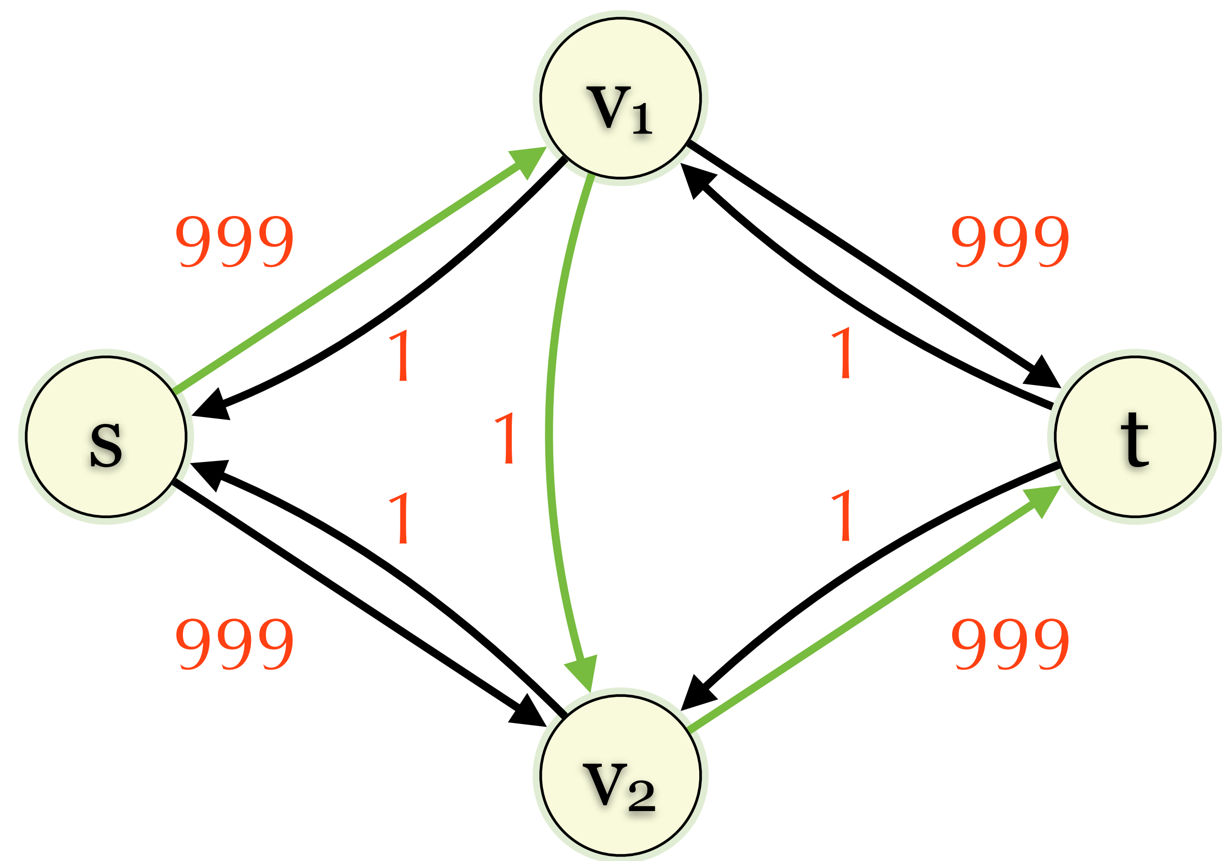
Augment 1

Bad Case for FF

Flow f



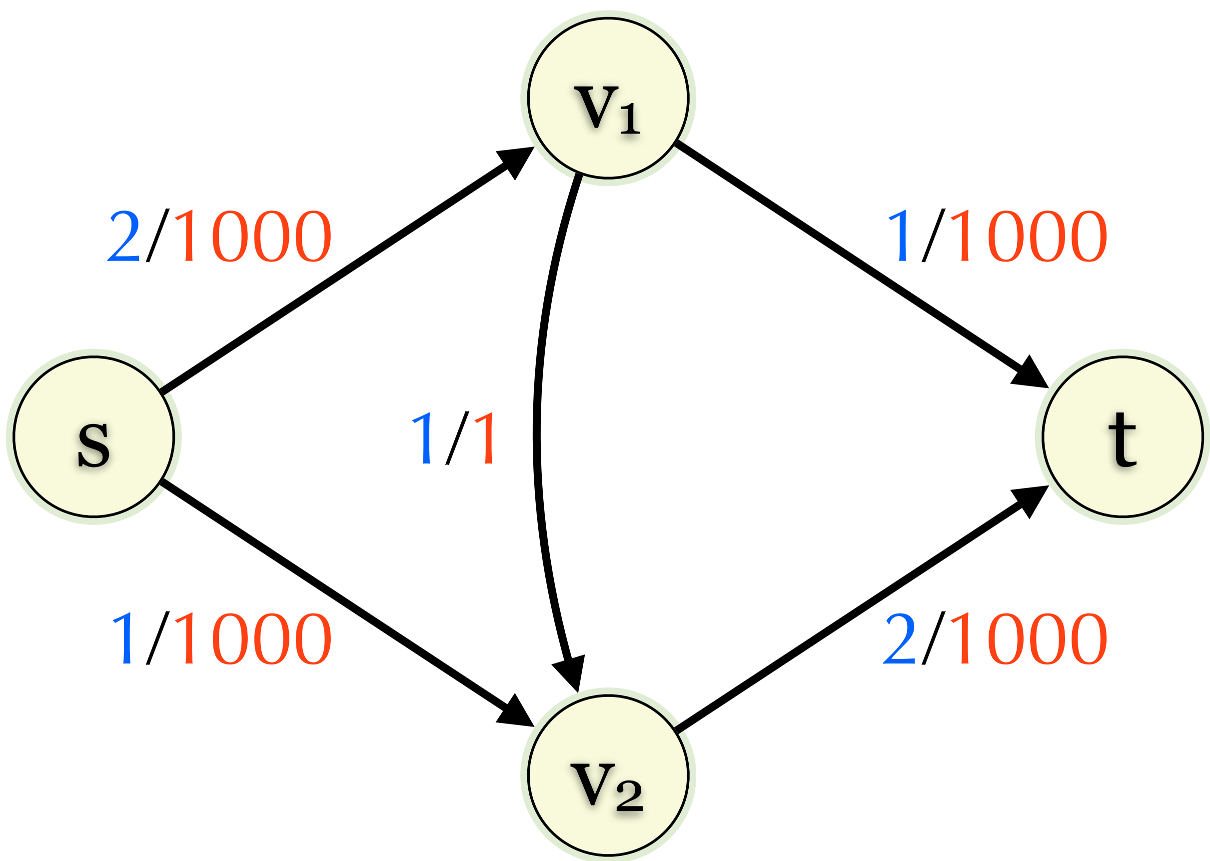
G_f



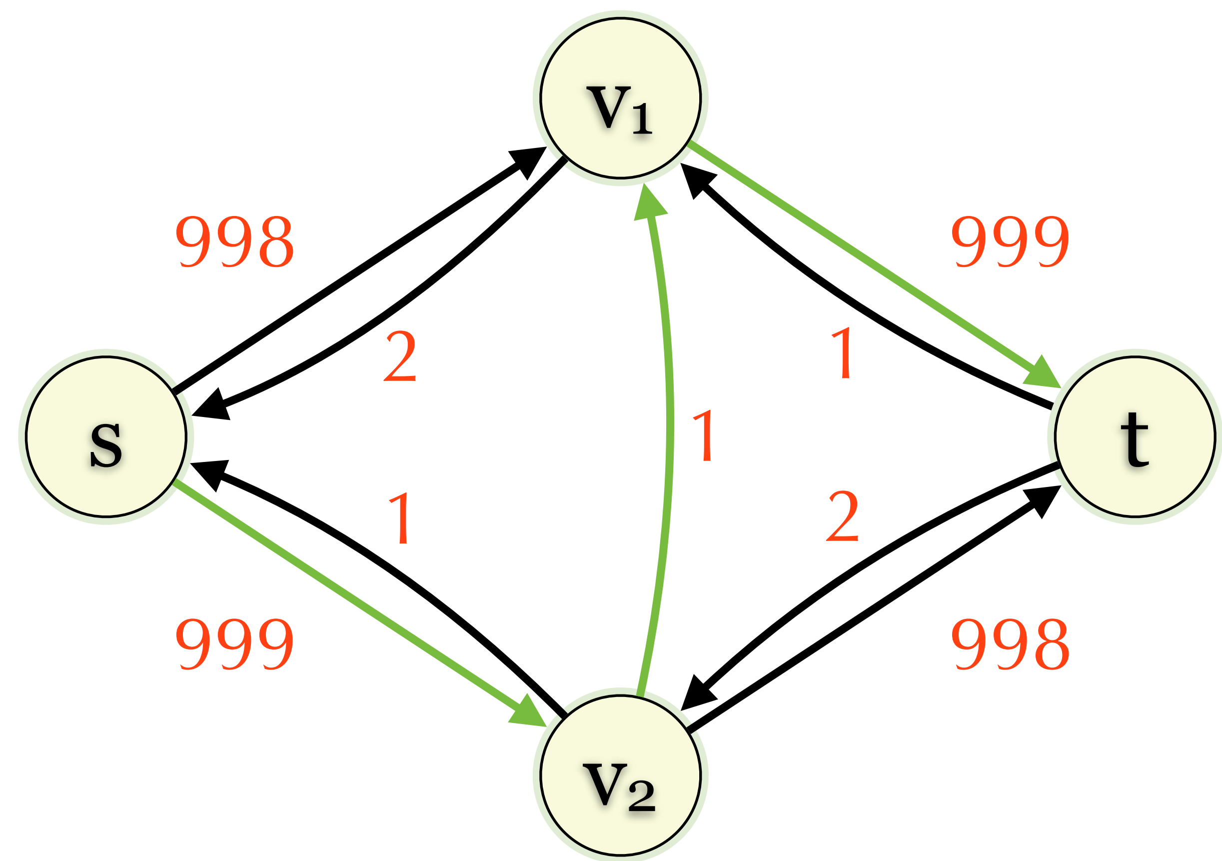
Augment 1

Bad Case for FF

Flow f



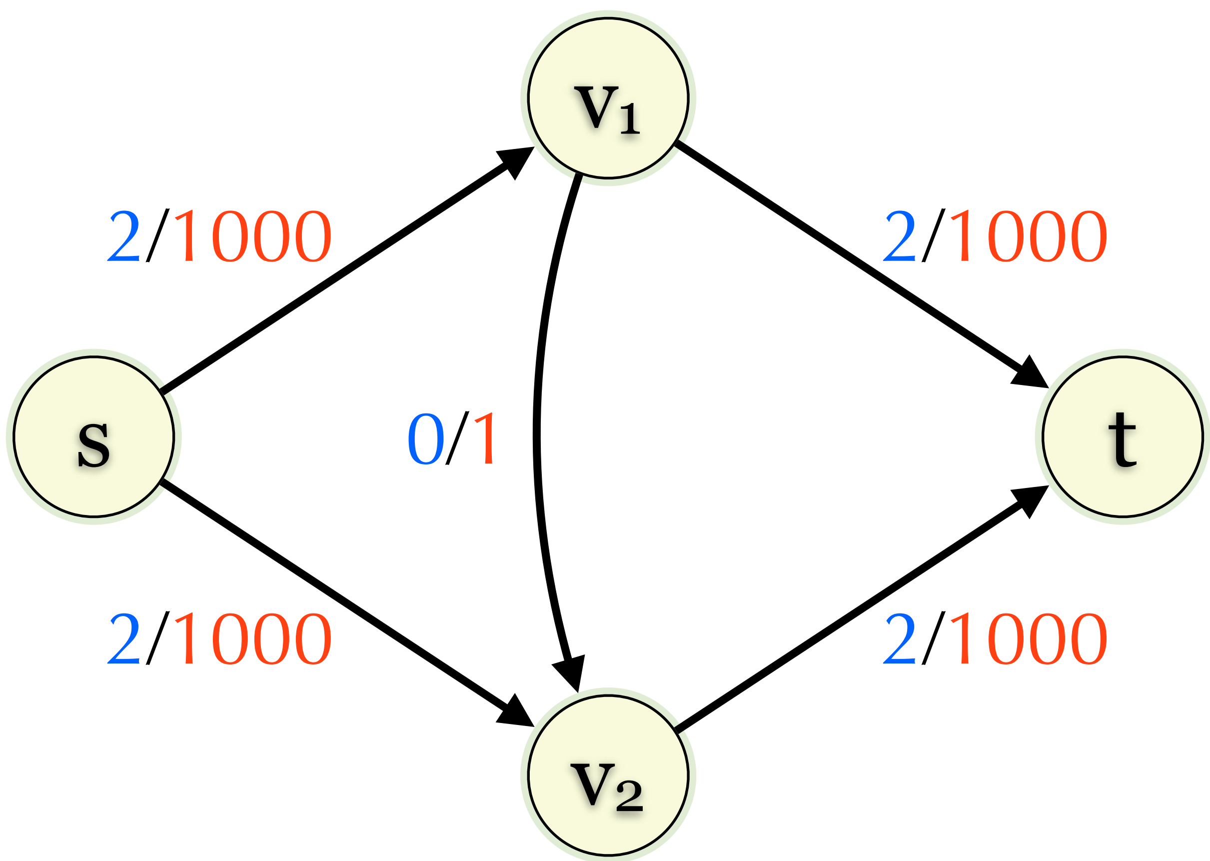
G_f



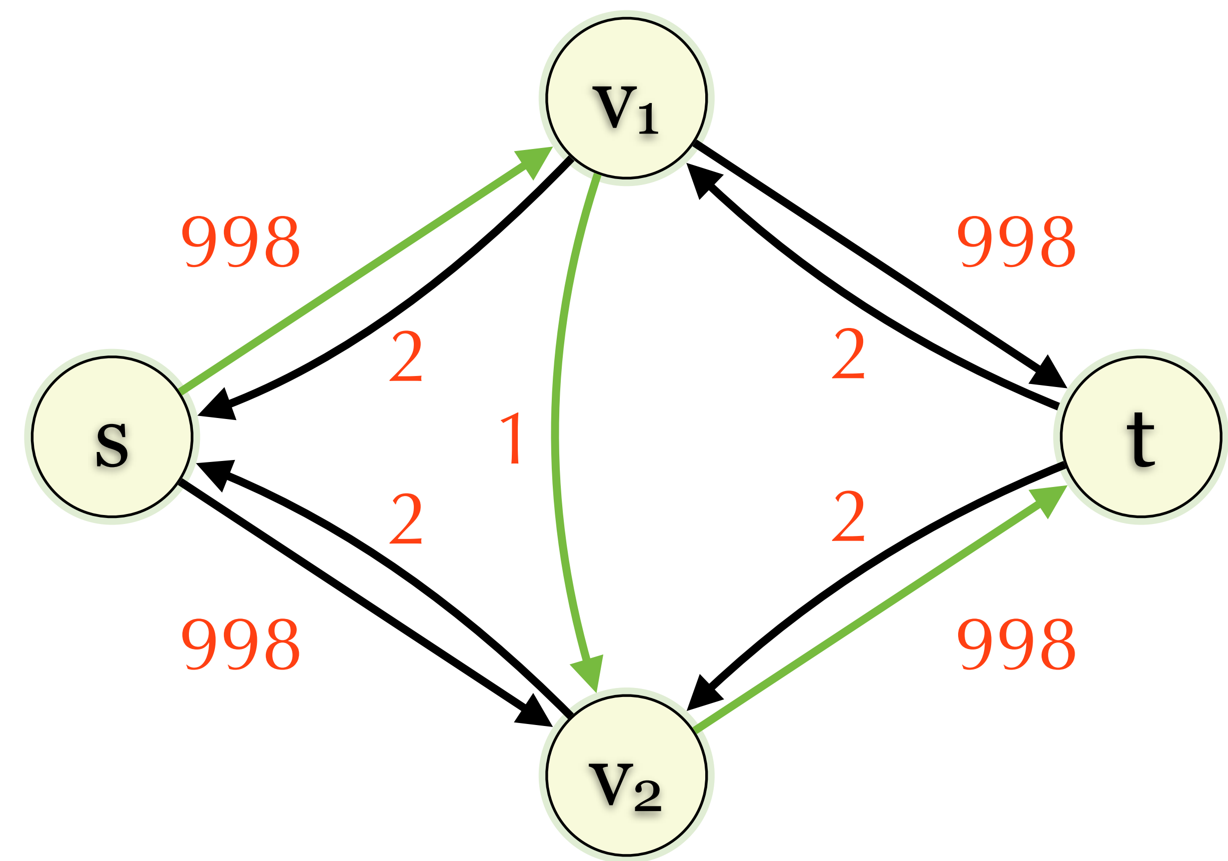
Augment 1

Bad Case for FF

Flow f



G_f

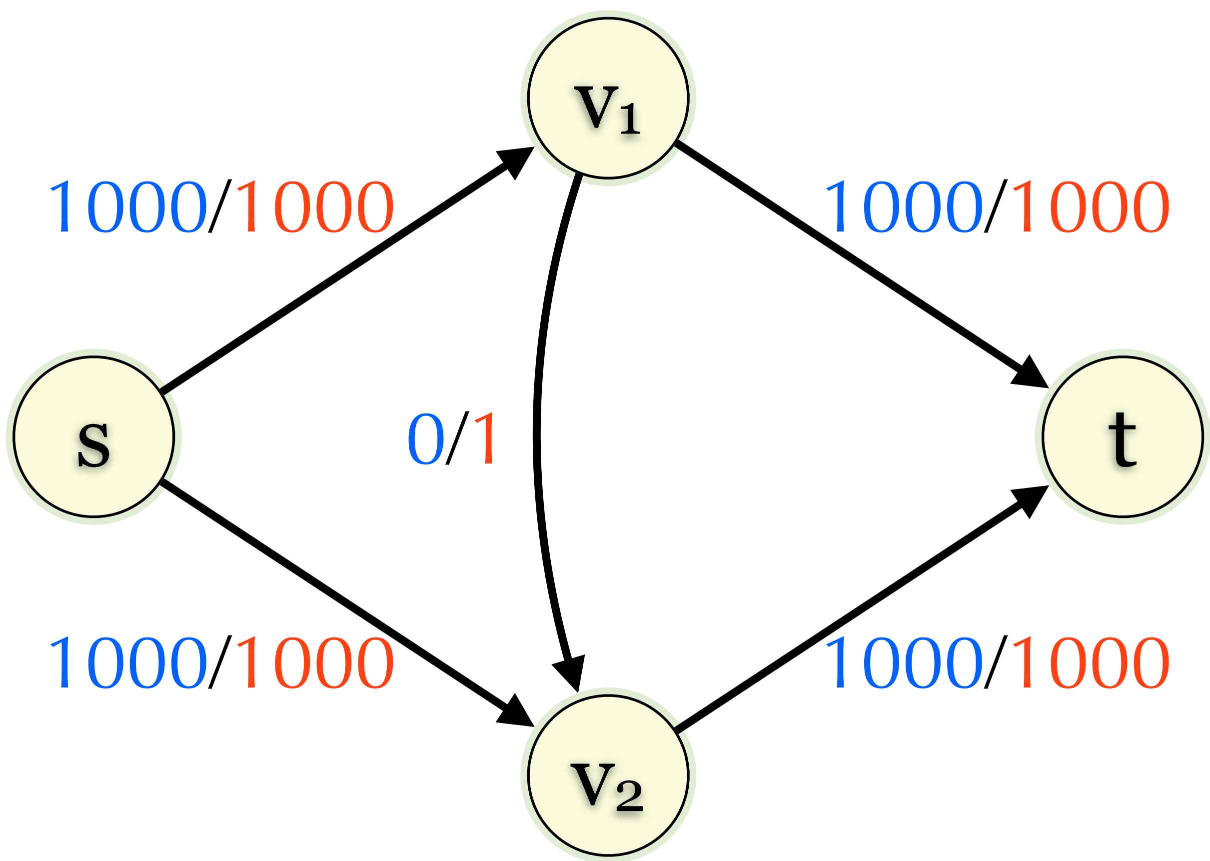


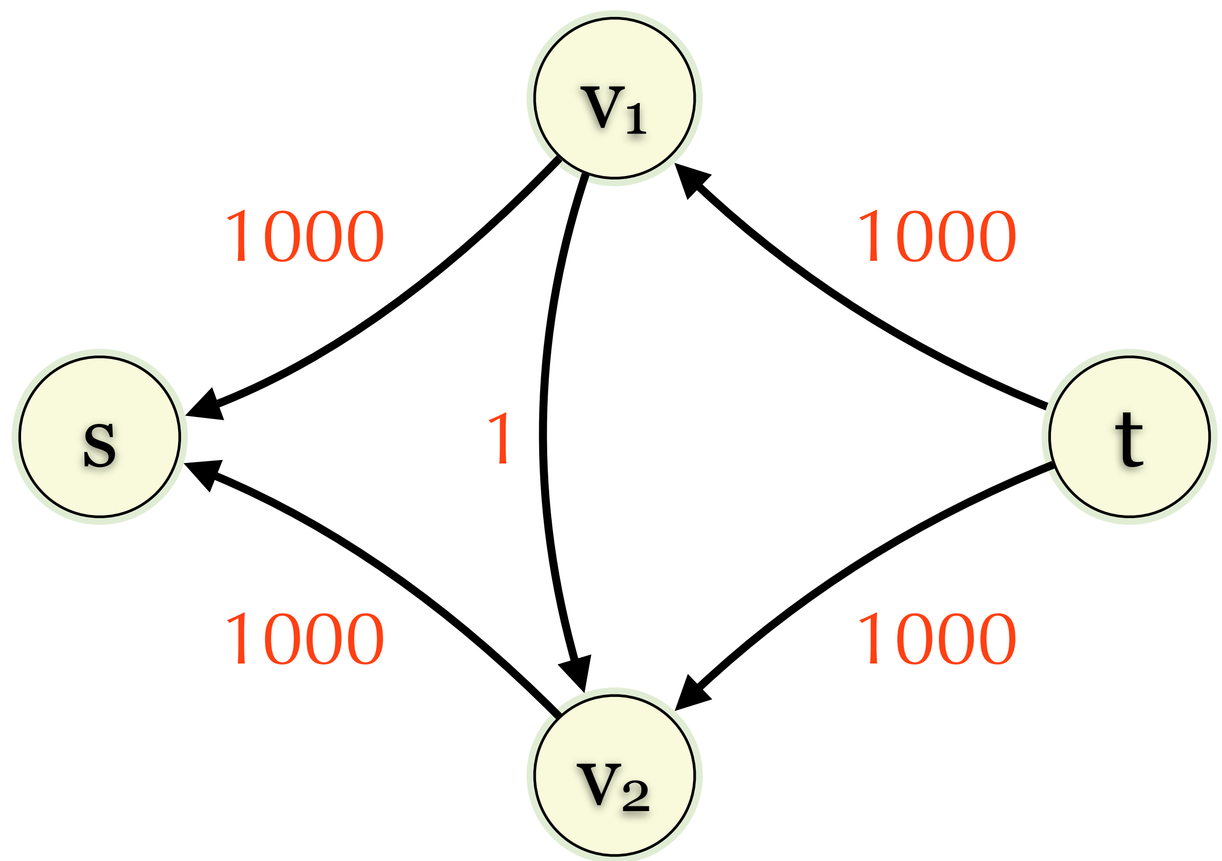
Augment 1

Bad Case for FF

Flow f

G_f





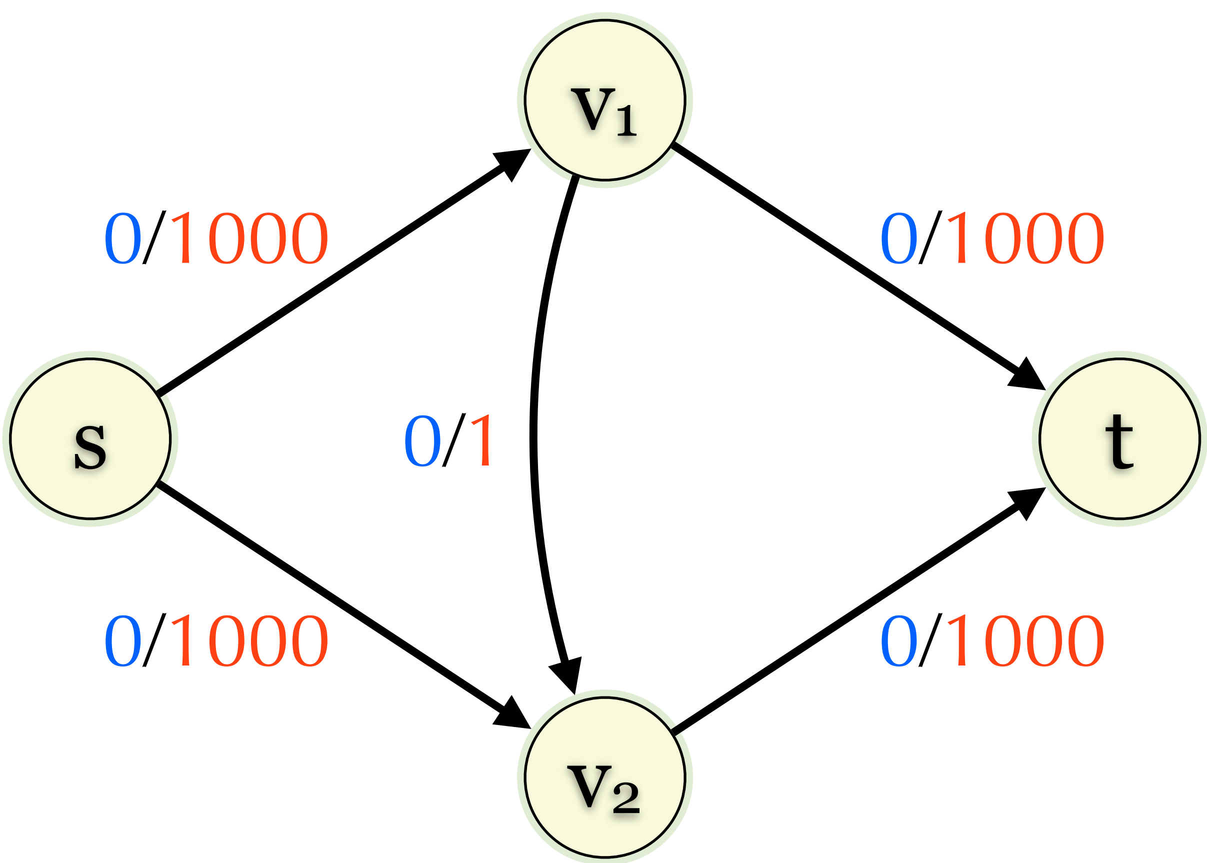
After 2000 augmentations

Other method

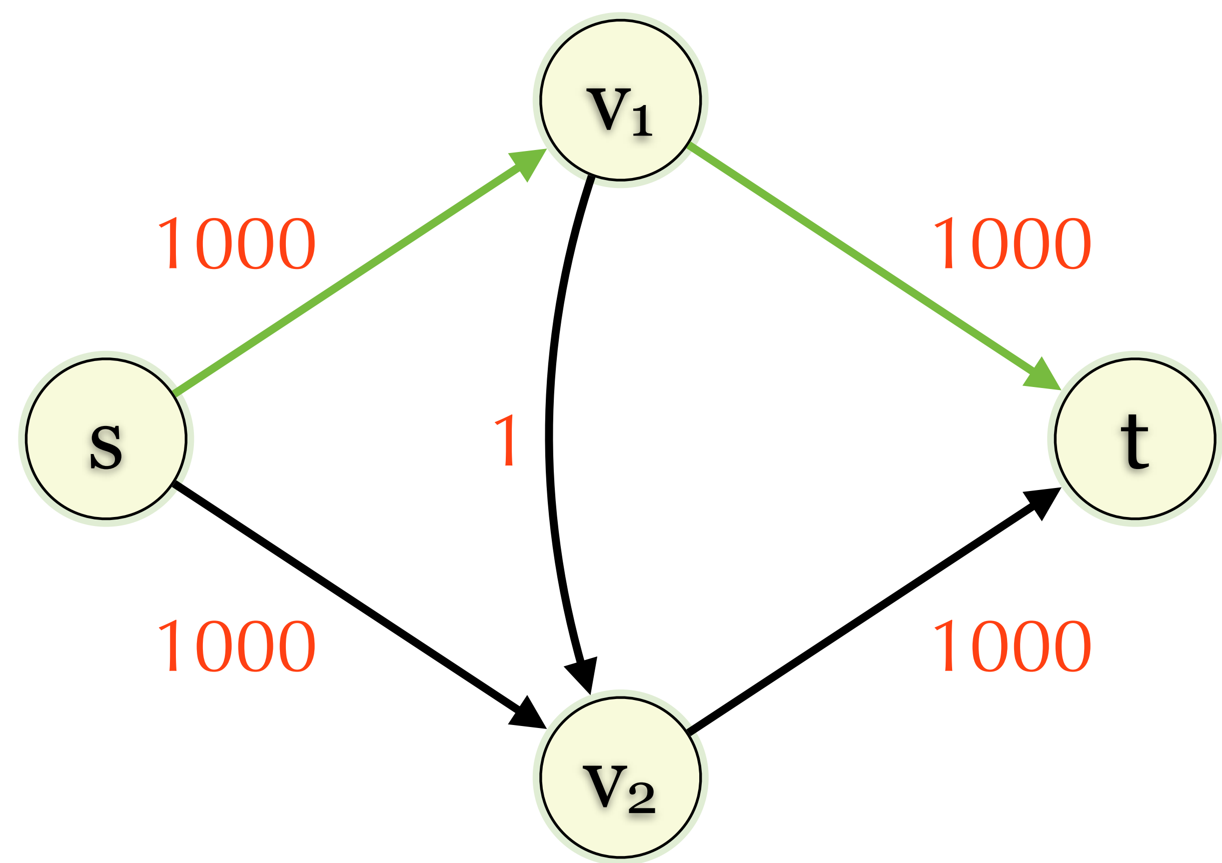
- Edmond-Karp
  - Finding augmenting path by BFS.
  - Performance: $O(VE^2)$
- Push-relabel
  - Presentation: 5 points
  - Performance: $O(V^3)$

Edmond-Karp

Flow f



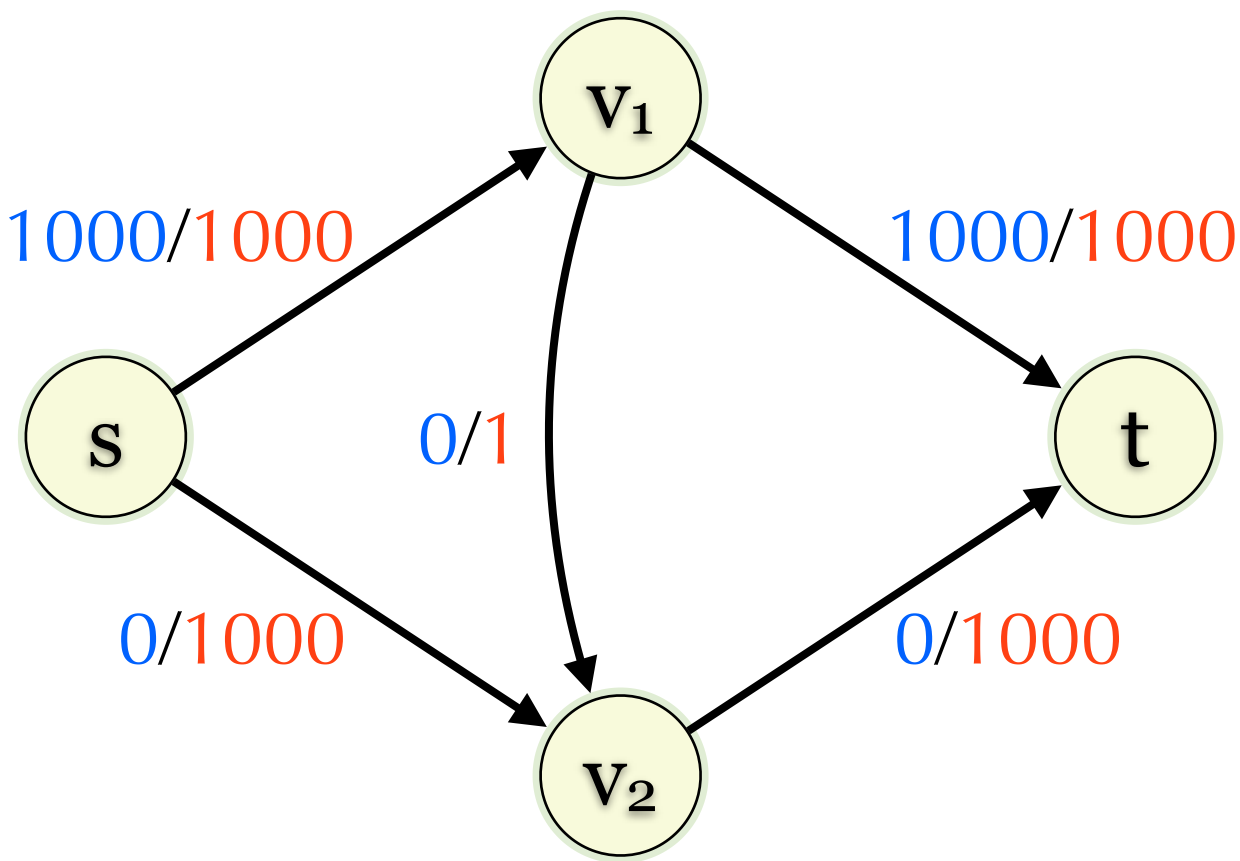
G_f



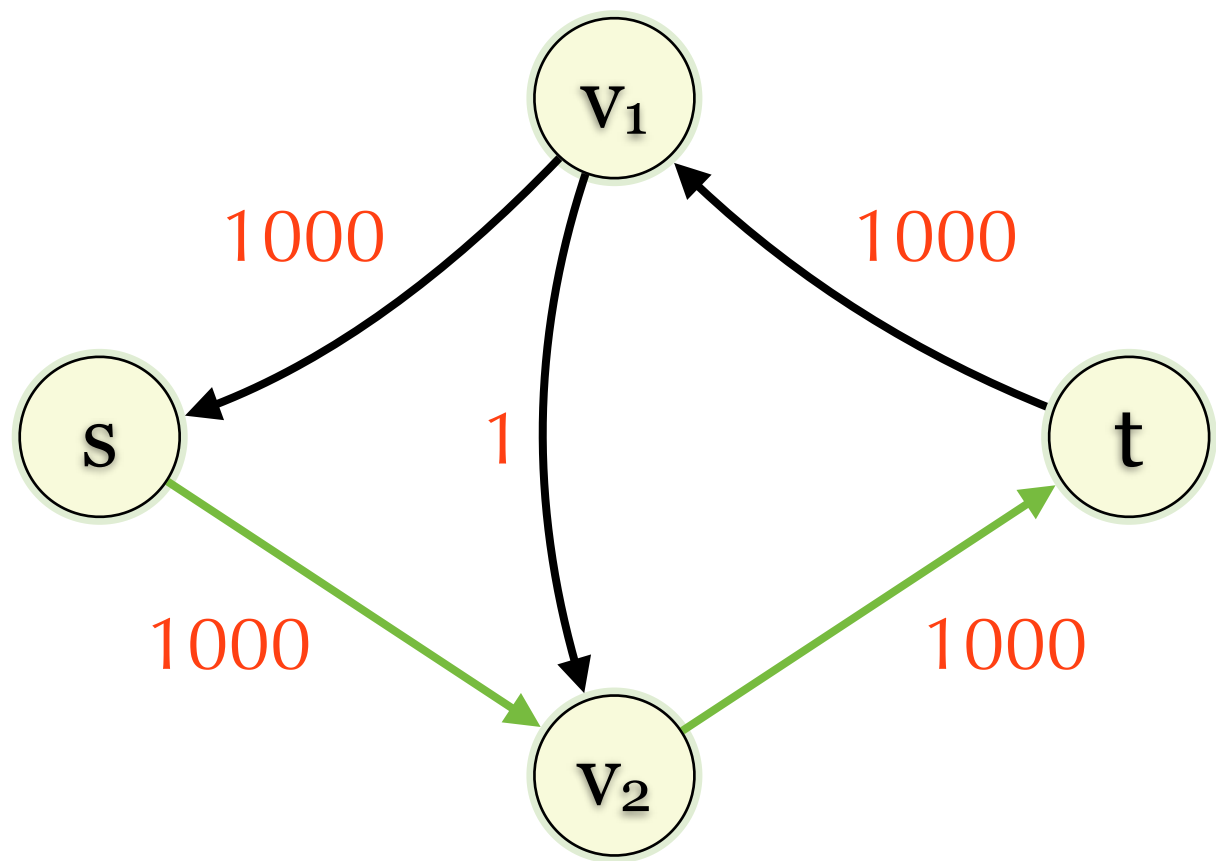
Augment 1000

Edmond-Karp

Flow f



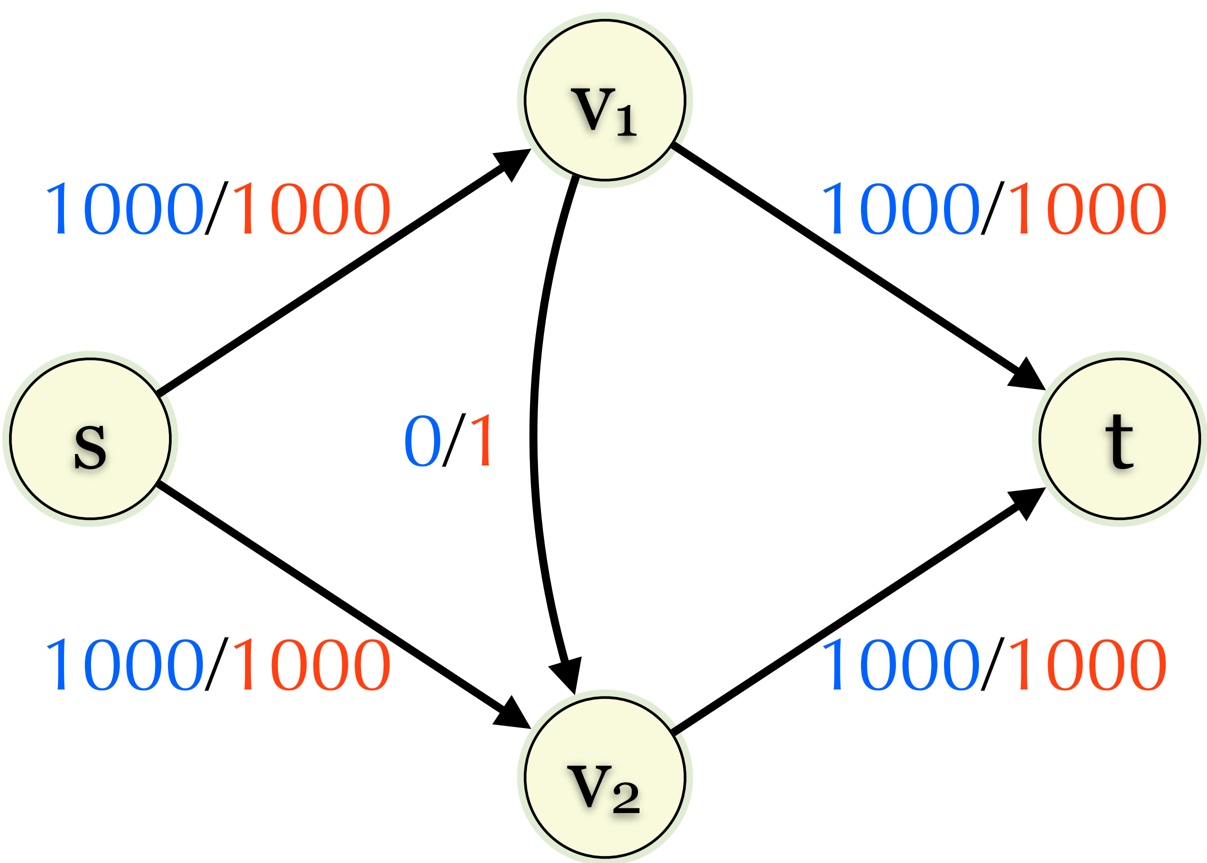
G_f



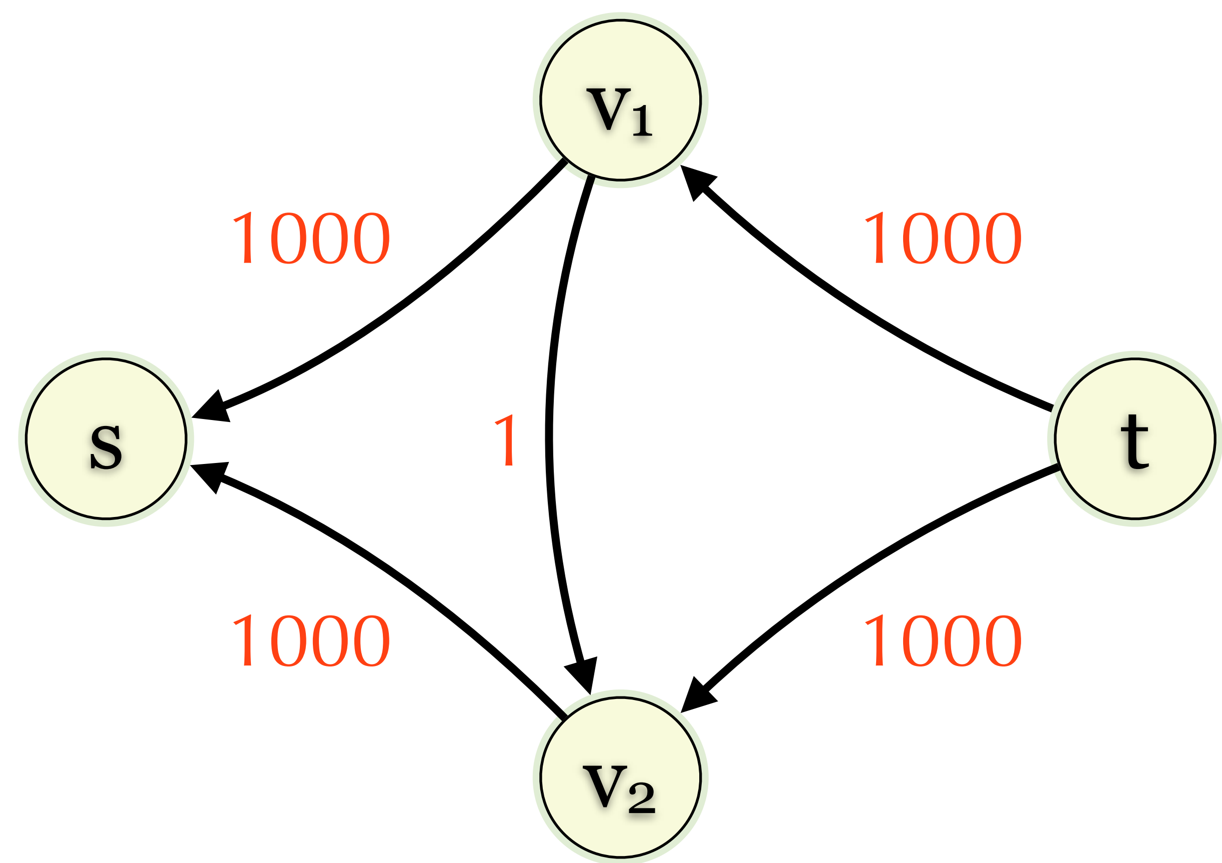
Augment 1000

Edmond-Karp

Flow f



G_f

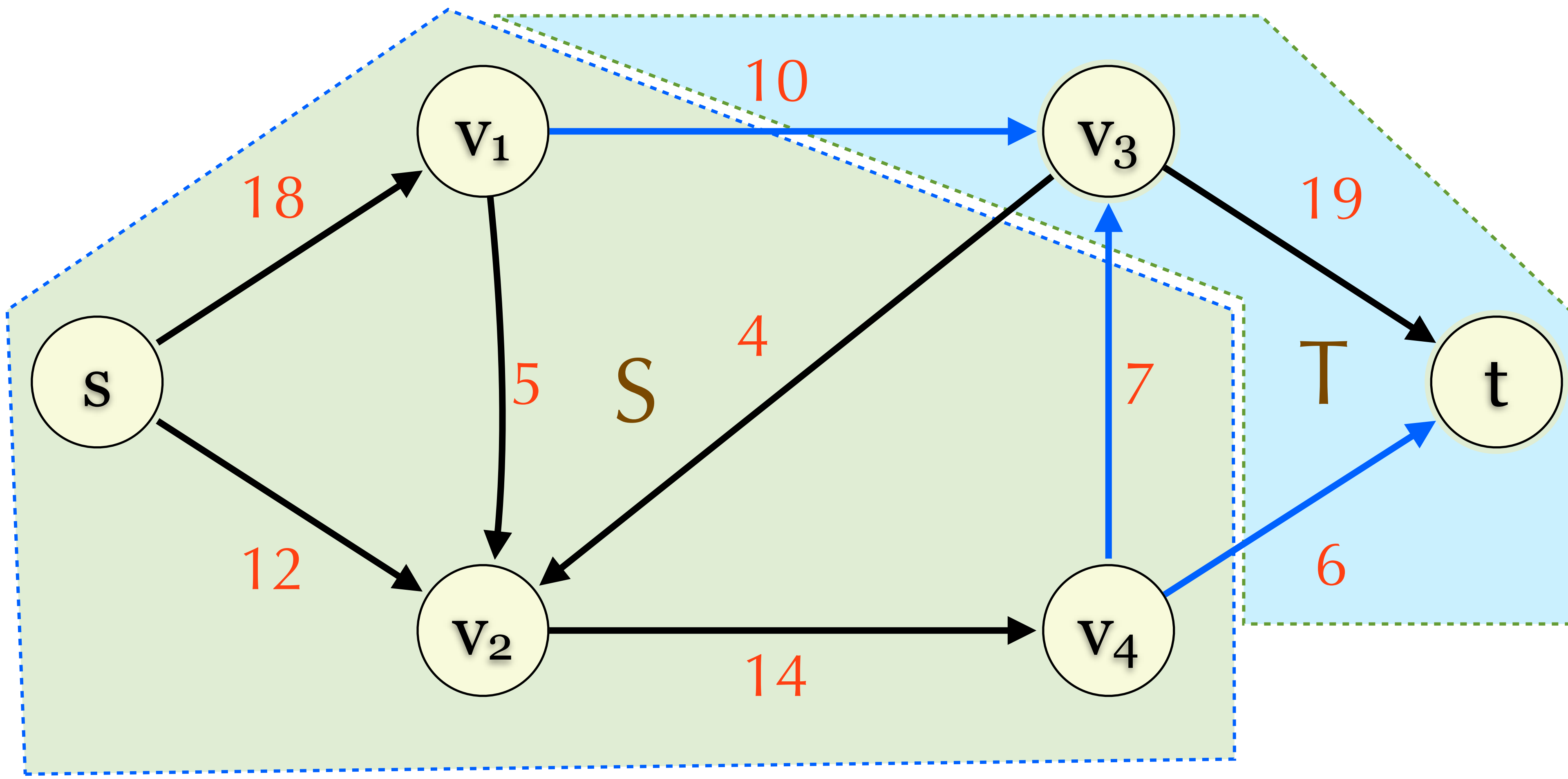


After only 2 augmentations

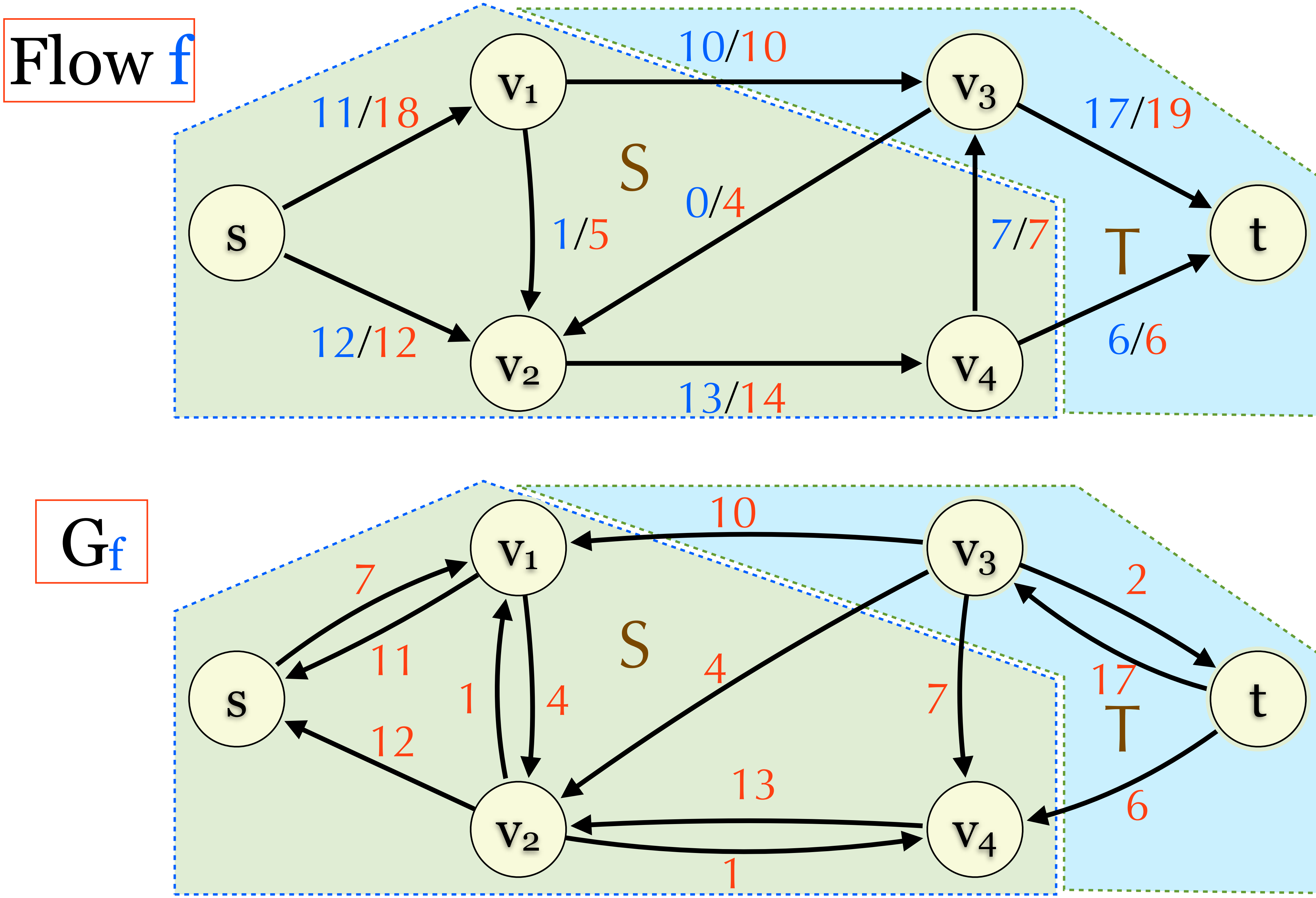
Minimum Cut

- s-t cut (S,T):
  - A partition of V, i.e., $S \cup T = V$ & $S \cap T = \emptyset$.
  - $s \in S$ and $t \in T$.
- Cost of a s-t cut: $c(S,T)=\Sigma_{u \in S, v \in T} c(u,v)$
- Min-cut Max-flow theorem
  - $\max_f |f| = \min_{S \cup T = V, S \cap T = \emptyset} c(S,T)$.
  - If S is the set of vertices reachable from s in G_f where f is the max flow, then $(S, V-S)$ is the min cut.

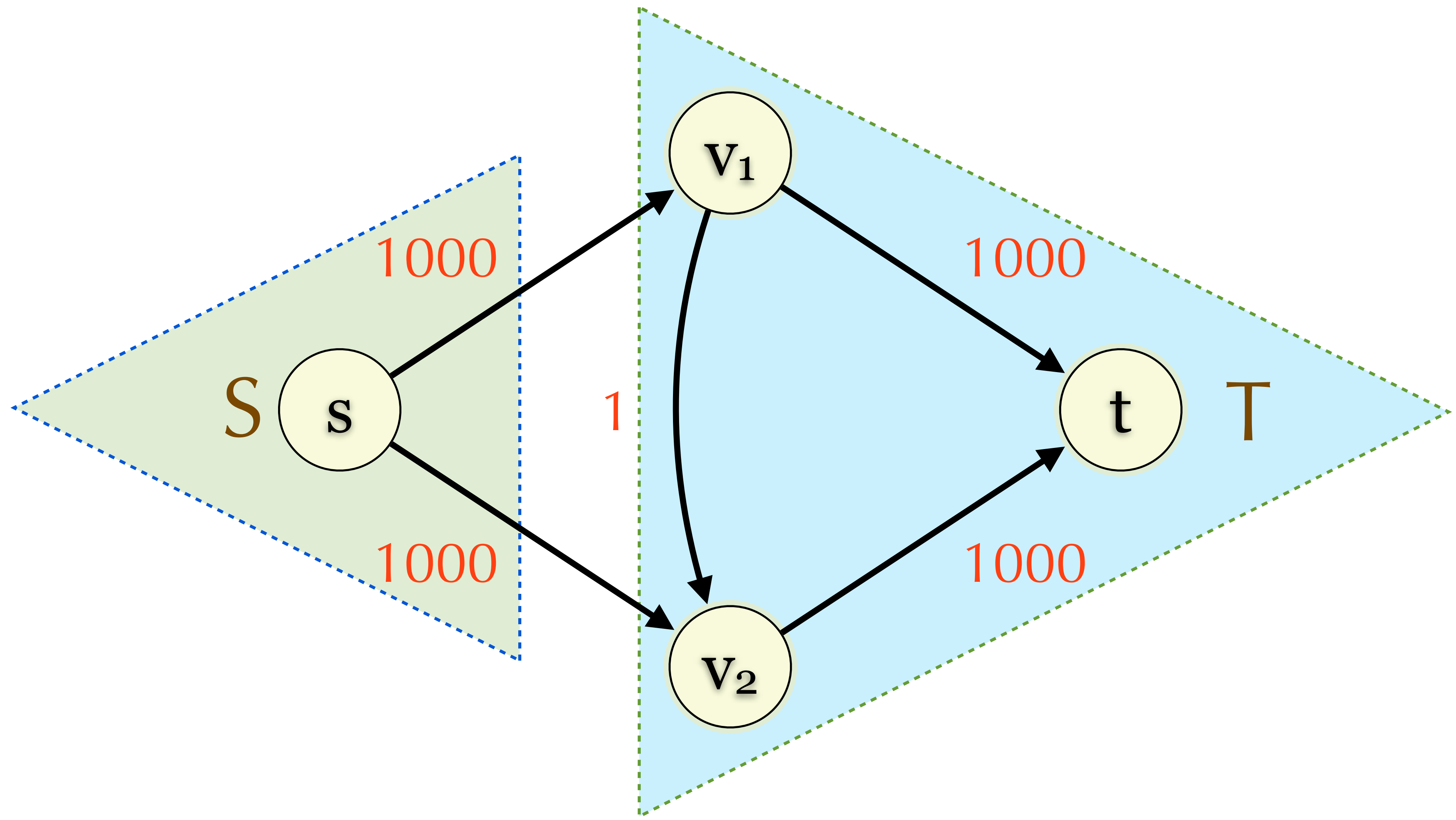
Minimum Cut: Example



Minimum Cut: example



Minimum Cut: example



Tools and Products

- In order to make a living, a factory has to buy some tools to manufacture some products.
  - Products: $p_1,\ldots,p_m$
  - Tools: $q_1,\ldots,q_n$
  - To make product p_i, the tools in Q_i are needed.
- How can the factory maximize the total profit?
  - $r(p)$: the profit of p
  - $c(q)$: the cost q
  - Total profit: $\max_{Q\subseteq\{q_1,\ldots,q_n\}}\Sigma_{Q_i\subseteq Q}r(p_i)-\Sigma_{q\in Q}c(q)$

Tools and Products

- Brute force?
  - Try all $Q \subseteq \{q_1, \ldots, q_n\}$. $O(mn2^n)$
- Build a flow network and find the min cut by Edmond-Karp.
  - $O((n+m)^5)$
- Any other faster idea?

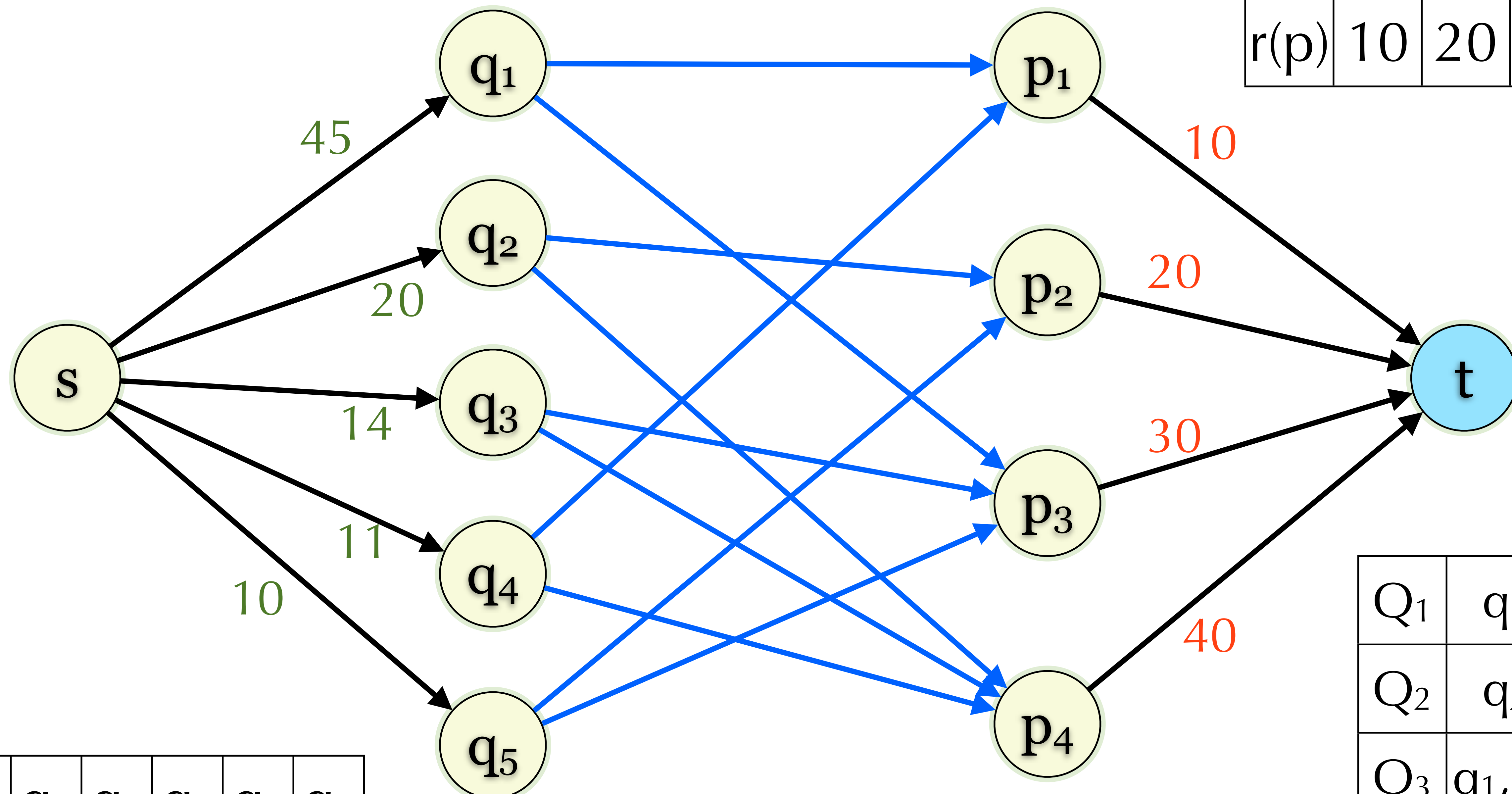
Tools and Products

- Build a graph $G=(V,E)$
  - $V=\{s,t\}\cup V_P \cup V_Q$
    - $V_P=\{p_1,\ldots,p_m\}$, $V_Q=\{q_1,\ldots,q_n\}$.
  - $E=E_Q \cup E_B \cup E_P$
    - $E_Q=\{(s,v_Q):v_Q \in V_Q\}$ $w(s,v_Q)=c(v_Q)$
    - $E_B=\{(v_Q,v_P):v_P \text{ needs } v_Q\}$ $w(e_B)=\infty$
    - $E_P=\{(v_P,t):v_P \in V_P\}$ $w(v_P,t)=r(v_P)$

Solution by Min Cut

$$\boxed{\Sigma_{p\in V_P} r(p) - \min_{(S,T)} c(S,T)}$$

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40



q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

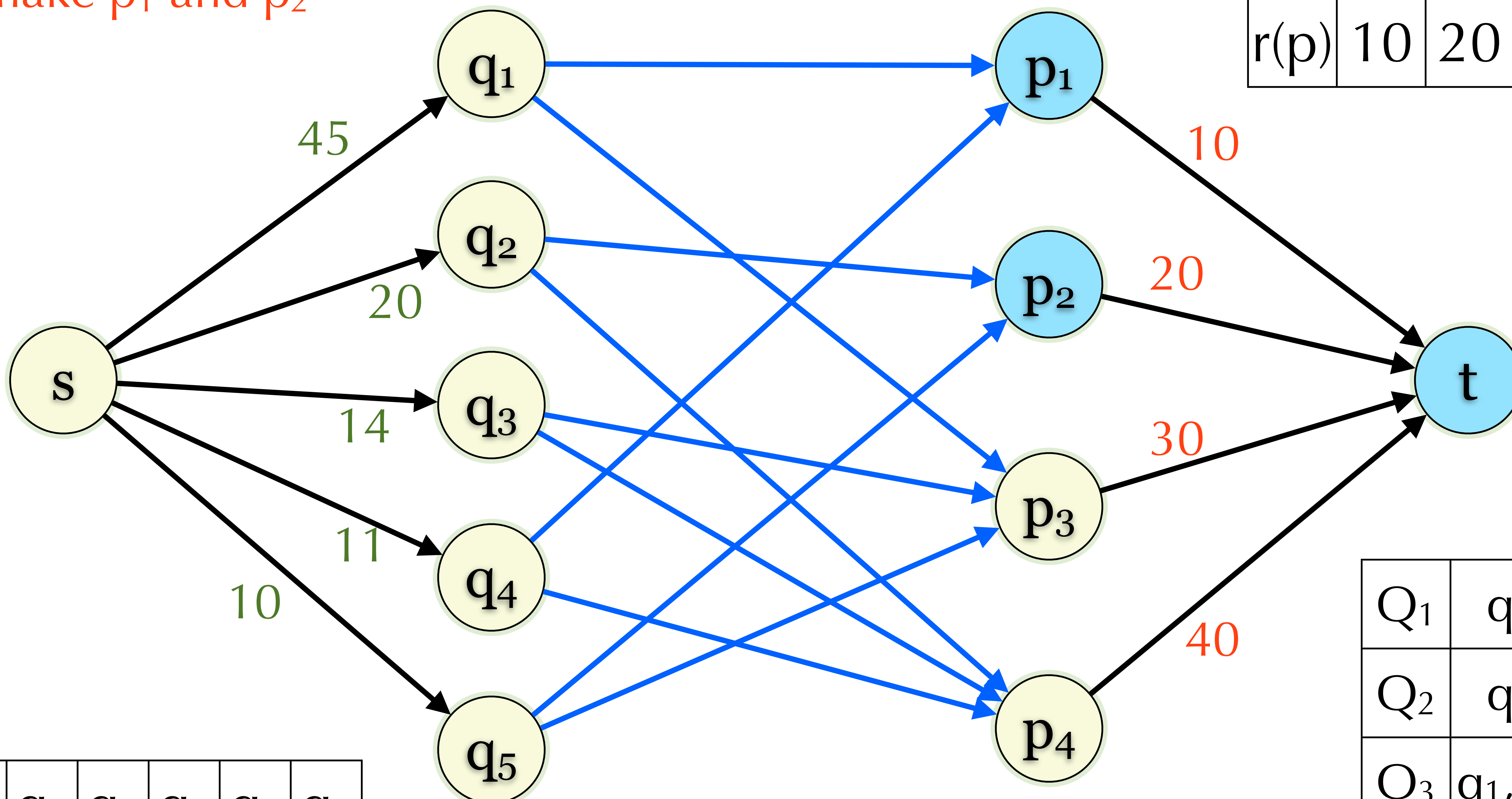
All blue edges have capacity ∞.

Q_1	q_1,q_4
Q_2	q_2,q_5
Q_3	q_1,q_3,q_5
Q_4	q_2,q_3,q_4

Why it works?

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40

To make p_1 and p_2



Q	
Q_1	q_1, q_4
Q_2	q_2, q_5
Q_3	q_1, q_3, q_5
Q_4	q_2, q_3, q_4

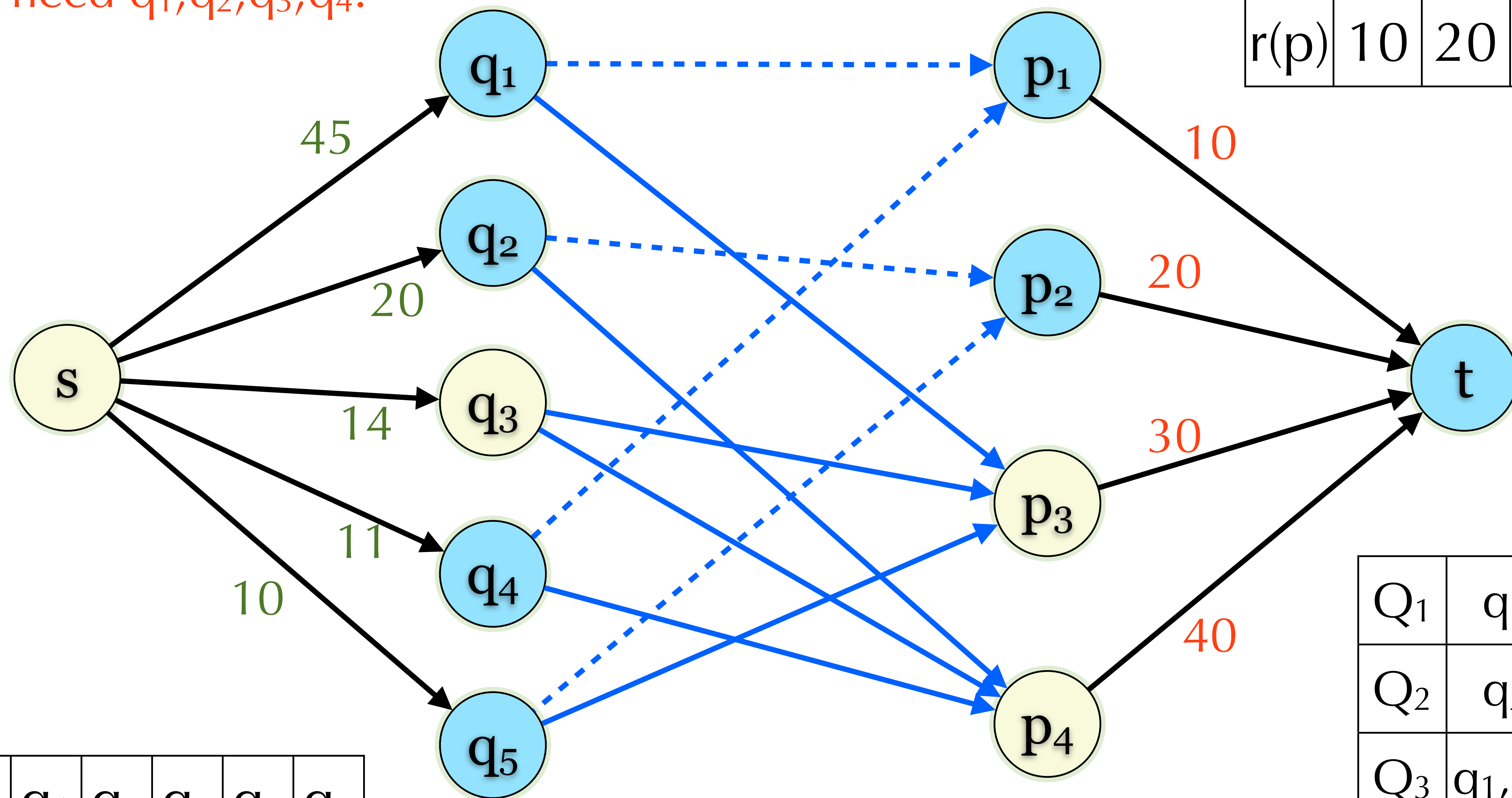
q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

All blue edges have capacity ∞.

Why it works?

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40

You need q_1, q_2, q_3, q_4.



Q	
Q_1	q_1, q_4
Q_2	q_2, q_5
Q_3	q_1, q_3, q_5
Q_4	q_2, q_3, q_4

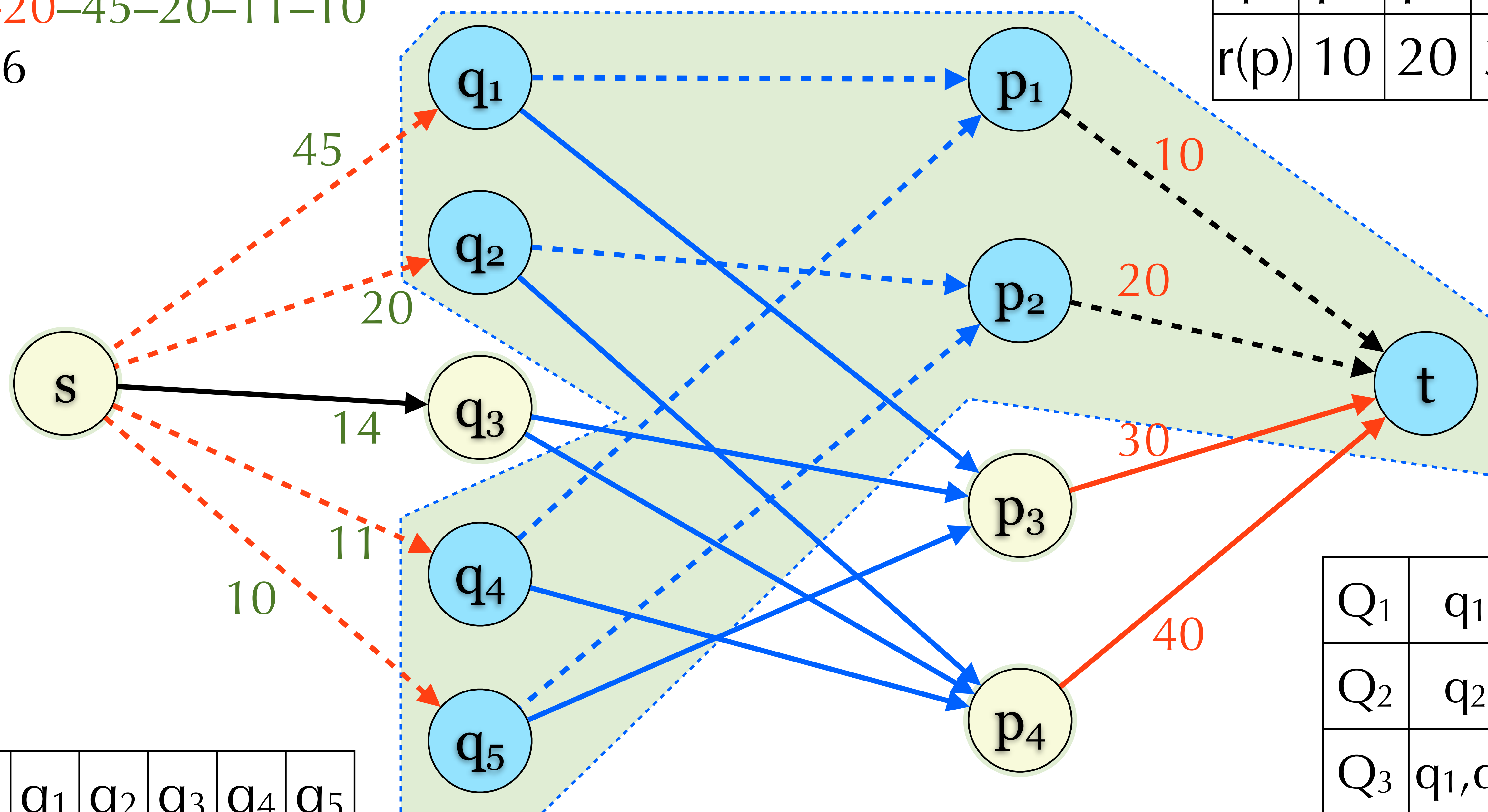
q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

All blue edges have capacity ∞.

Why it works?

No profit!
10+20–45–20–11–10
=–56

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40



All blue edges have capacity ∞.

q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

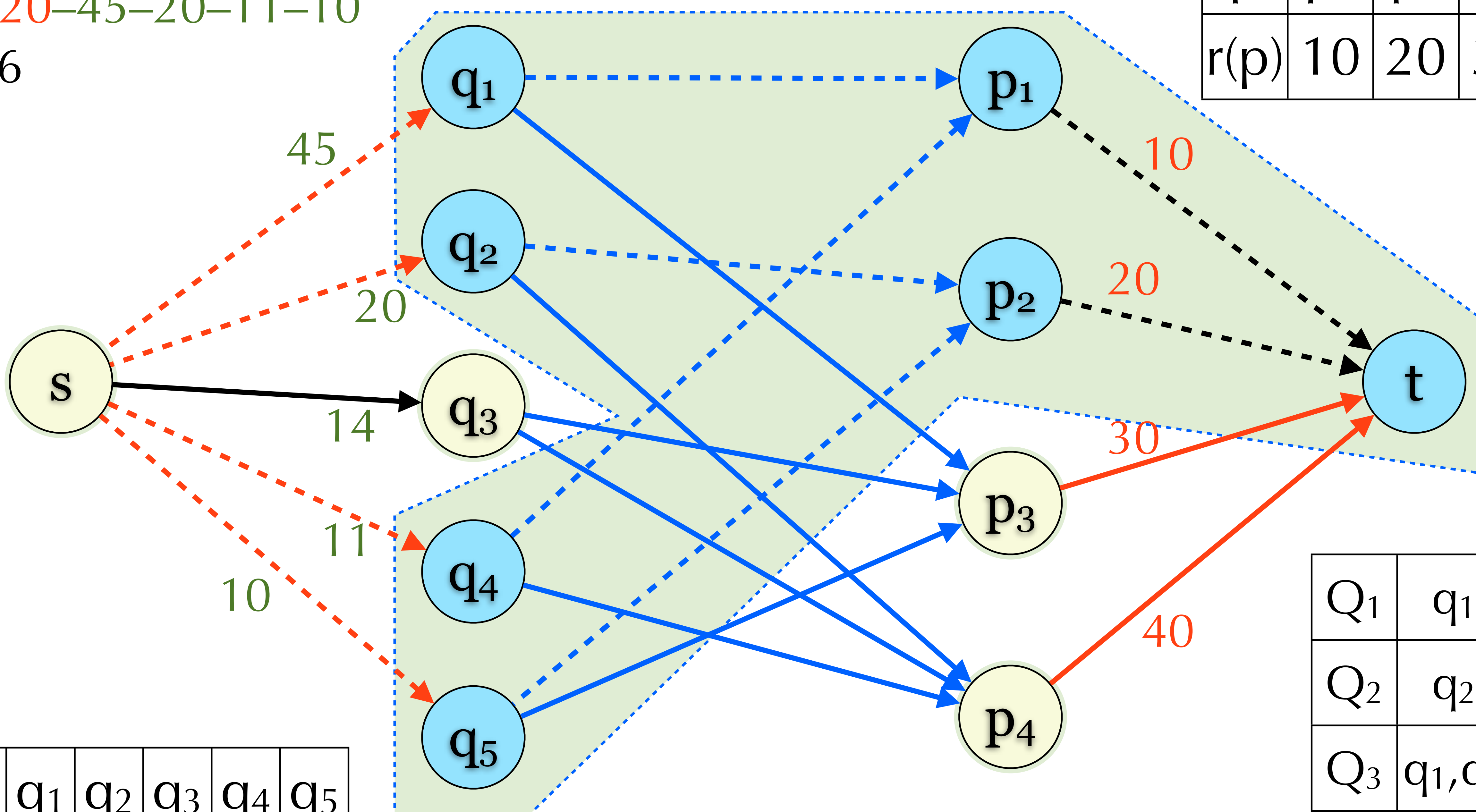
Q_1	q_1,q_4
Q_2	q_2,q_5
Q_3	q_1,q_3,q_5
Q_4	q_2,q_3,q_4

$\Sigma_{p \in V_P} r(p) - c(S,T)$
$= 100 - 30 - 40 - 45 - 20 - 11 - 10$
$= 10 + 20 - 45 - 20 - 11 - 10$
$= -56$

All tools needed are in T.
No blue edges $(u,v) \in S \times T$.

No profit!
$10 + 20 - 45 - 20 - 11 - 10$
$= -56$

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40



Q_1	q_1, q_4
Q_2	q_2, q_5
Q_3	q_1, q_3, q_5
Q_4	q_2, q_3, q_4

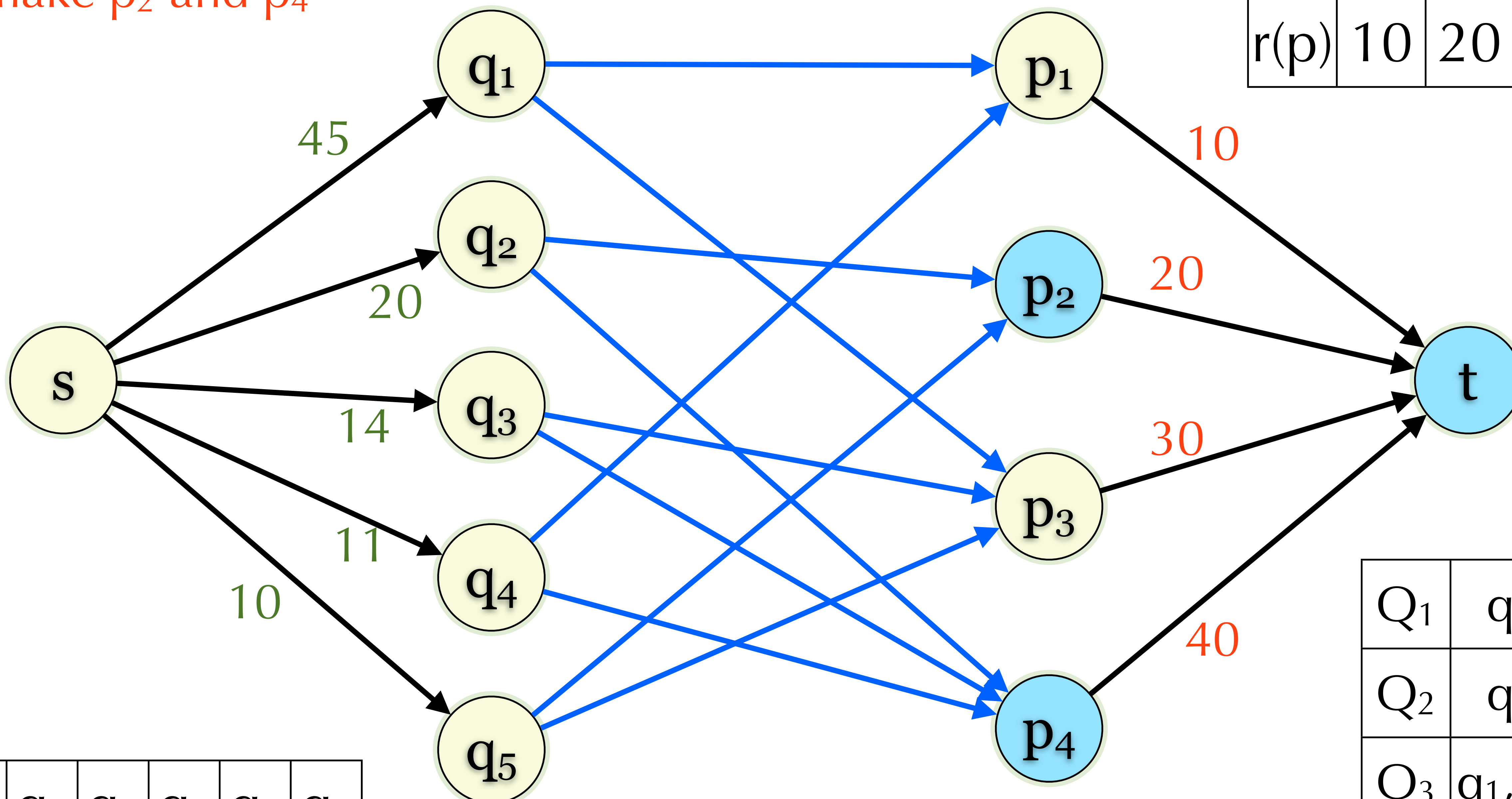
q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

All blue edges have capacity ∞.

Why it works?

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40

To make p_2 and p_4



Q	
Q_1	q_1, q_4
Q_2	q_2, q_5
Q_3	q_1, q_3, q_5
Q_4	q_2, q_3, q_4

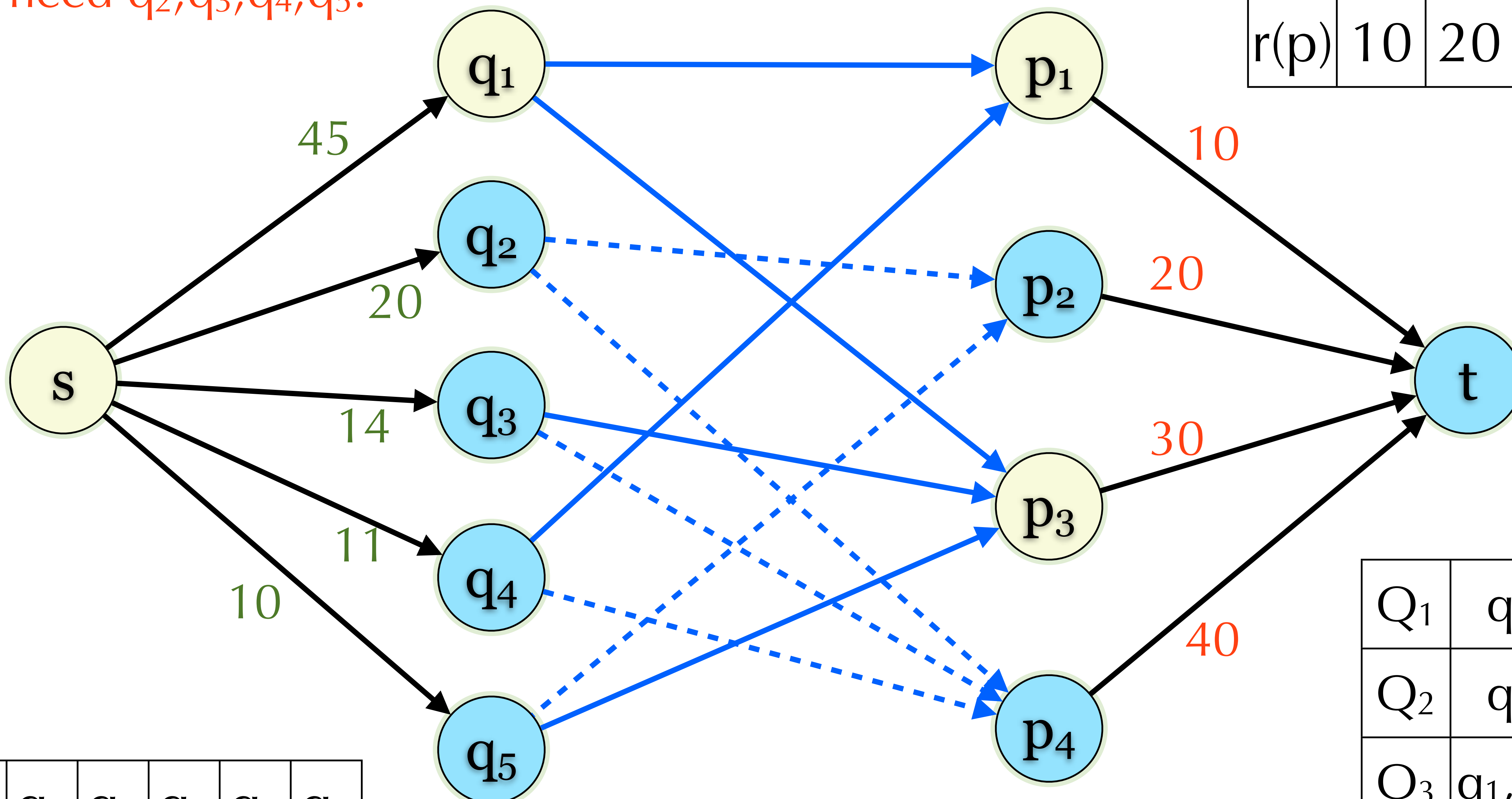
q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

All blue edges have capacity ∞.

Why it works?

You need q_2, q_3, q_4, q_5.

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40



q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

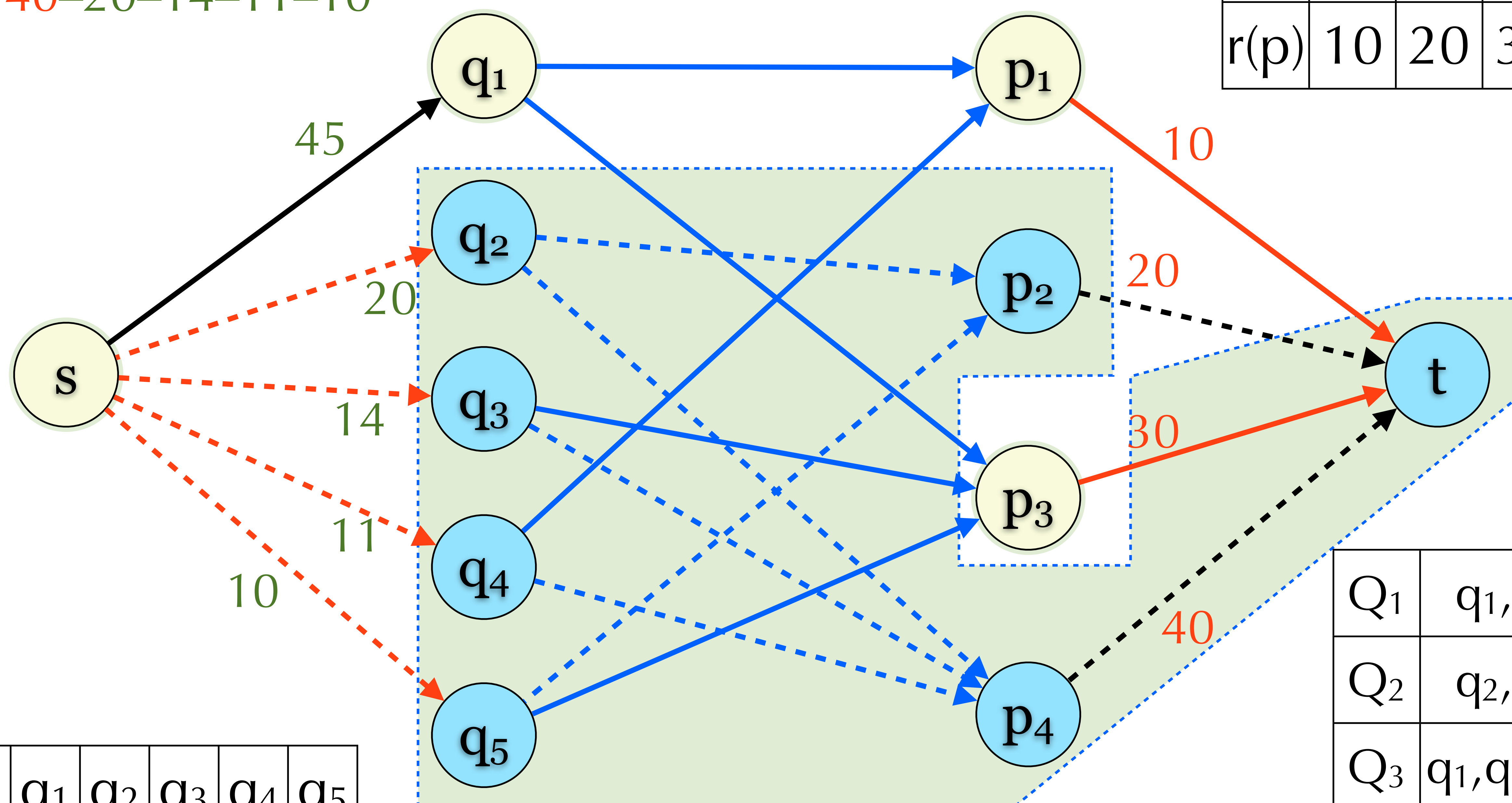
All blue edges have capacity ∞.

Q_1	q_1, q_4
Q_2	q_2, q_5
Q_3	q_1, q_3, q_5
Q_4	q_2, q_3, q_4

Why it works?

Profit!
20+40–20–14–11–10
=5

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40



All blue edges have capacity ∞.

q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

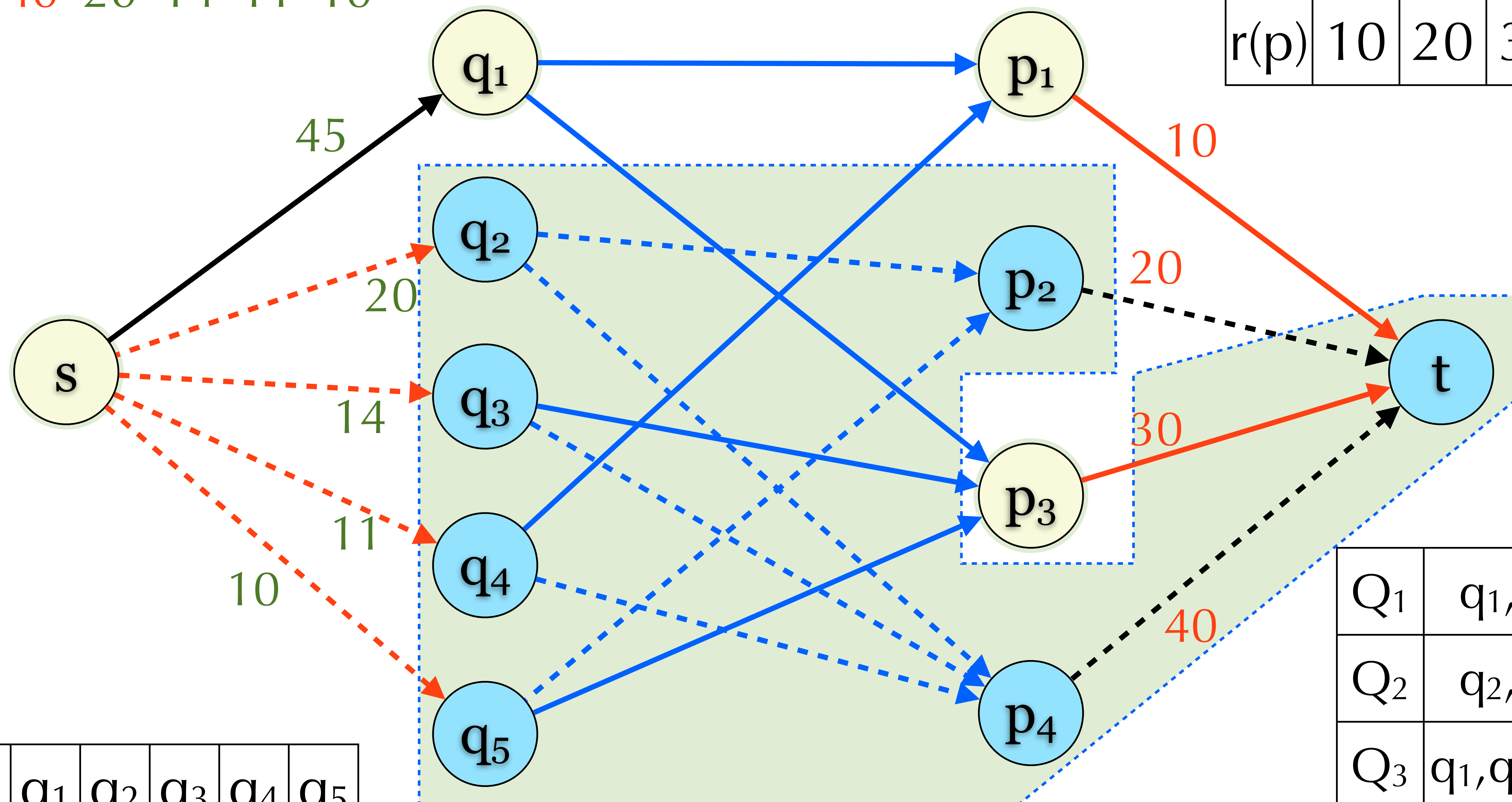
Q_1	q_1, q_4
Q_2	q_2, q_5
Q_3	q_1, q_3, q_5
Q_4	q_2, q_3, q_4

$\Sigma_{p\in V_P} r(p) - c(S,T)$
$= 100 - 10 - 30 - 20 - 14 - 11 - 10$
$= 20 + 40 - 20 - 14 - 11 - 10$
$= 5$

All tools needed are in T.
No blue edges $(u,v) \in S \times T$.

Profit!
$20 + 40 - 20 - 14 - 11 - 10$
$= 5$

p	p_1	p_2	p_3	p_4
r(p)	10	20	30	40



Q_1	q_1, q_4
Q_2	q_2, q_5
Q_3	q_1, q_3, q_5
Q_4	q_2, q_3, q_4

q	q_1	q_2	q_3	q_4	q_5
c(q)	45	20	14	11	10

All blue edges have capacity ∞.

Tools and Products

- If we want to produce all products in P, then we need $Q=\{q: \exists p \in P, (q,p) \in E\}$.
- Let $T=\{t\} \cup P \cup Q$ and $S=V \setminus T$.
  - For $(u,v) \in S \times T$, $c(u,v)<\infty$.
  - $c(S,T)<\infty$
- The rest part: Show every (S,T) corresponds to a strategy if $c(S,T)<\infty$.

Tools and Products

- Let $T=\{t\}\cup P\cup Q$ where P is a subset of V_P and Q is a subset of V_Q.
  - $S=V\setminus T=\{s\}\cup(V_P\setminus P)\cup(V_Q\setminus Q)$
- $c(S,T)<\infty$ implies $E\cap((V_Q\setminus Q)\times P)=\emptyset$
  - If we buy all tools in Q, then all product in P can be made.
- $\Sigma_{p\in V_P}r(p)-c(S,T)$
  $=\Sigma_{p\in V_P}r(p)-\Sigma_{p\notin P}r(p)-\Sigma_{q\in Q}c(q)$
  $=\Sigma_{p\in P}r(p)-\Sigma_{q\in Q}c(q)$

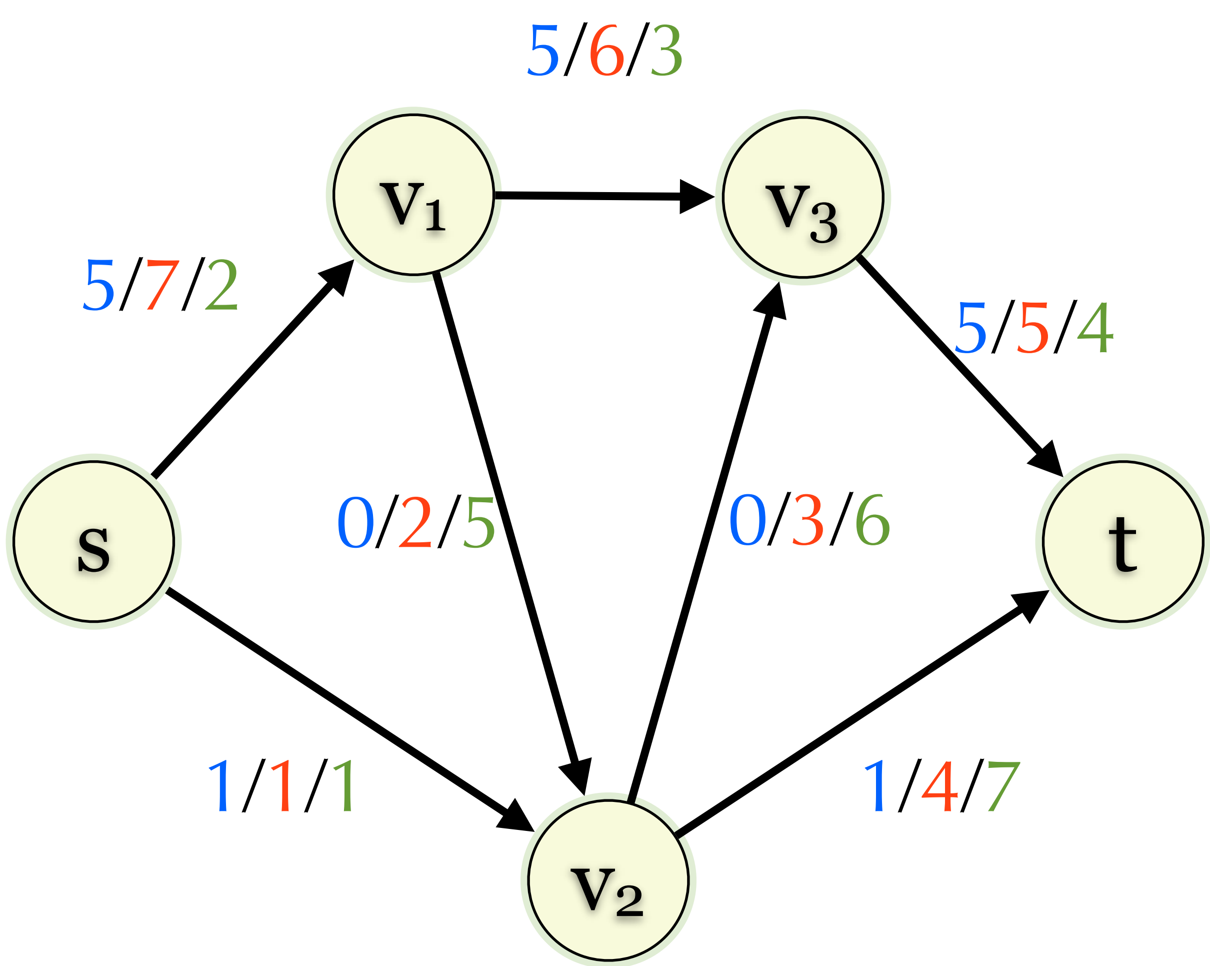
Minimum Cost Flow

- $e \in E$ has capacity $c(e)$ and cost $w(e)$
- Objective:
  - Find a flow f s.t. $|f|=f^*$
  - $\Sigma_{e \in E} f(e)w(e)$ is minimized
- How?
  - Successive shortest path
  - Negative cycle canceling

Example

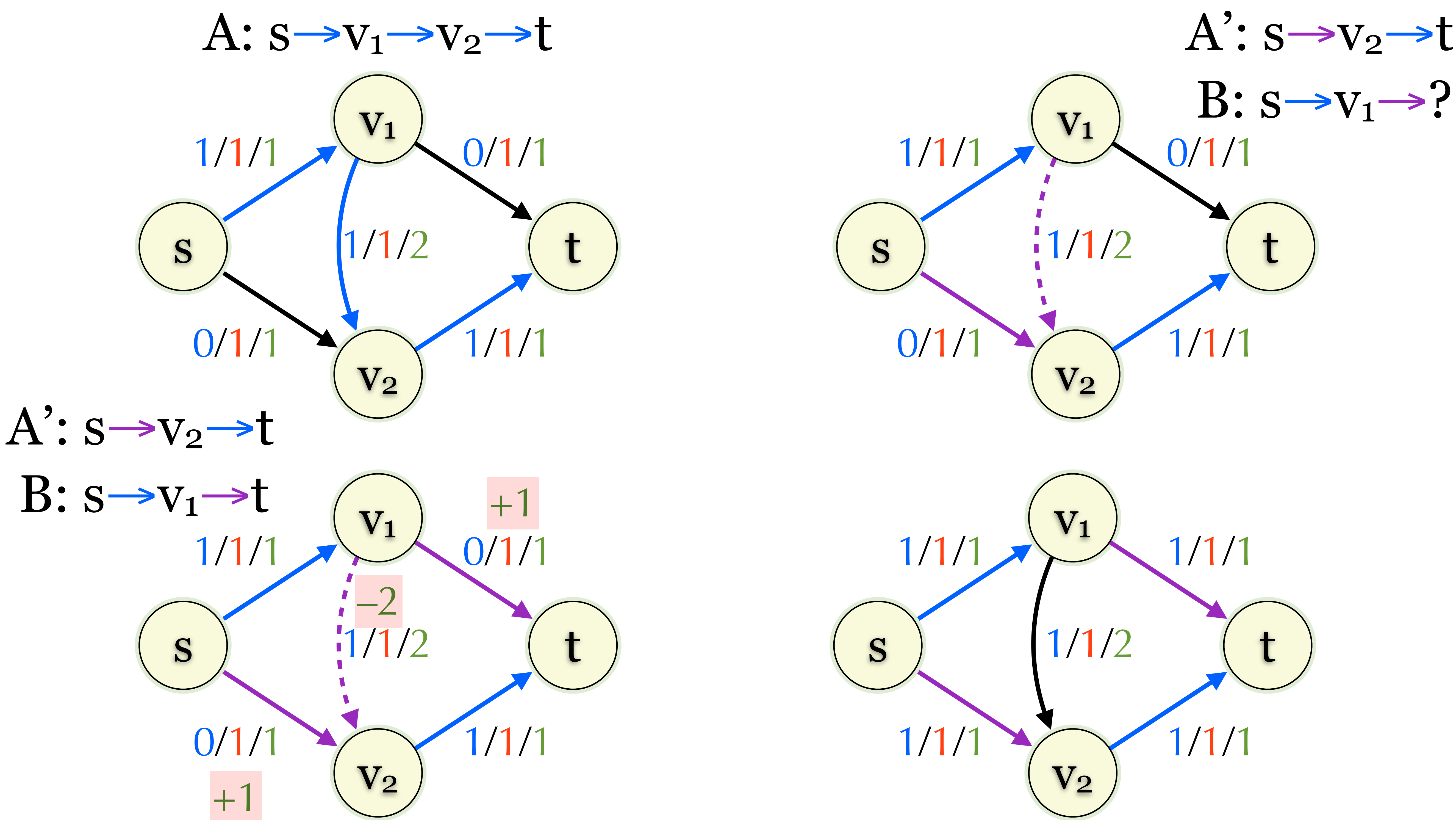
Flow f

$|f|=6$



$\text{Cost}=1\times1+5\times2+5\times3+5\times4+0\times5+0\times6+1\times7=53$

Cancellation

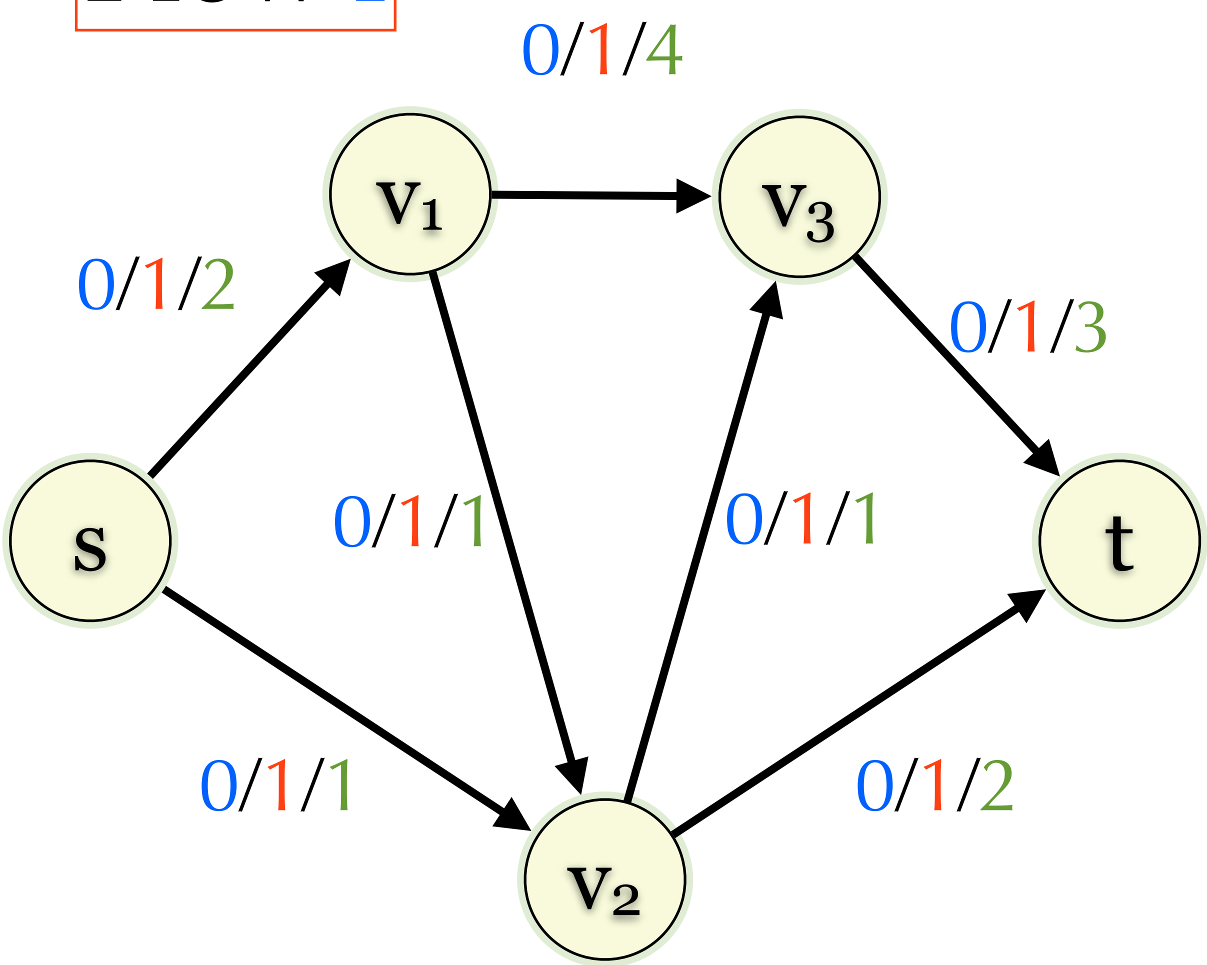


Residual Network

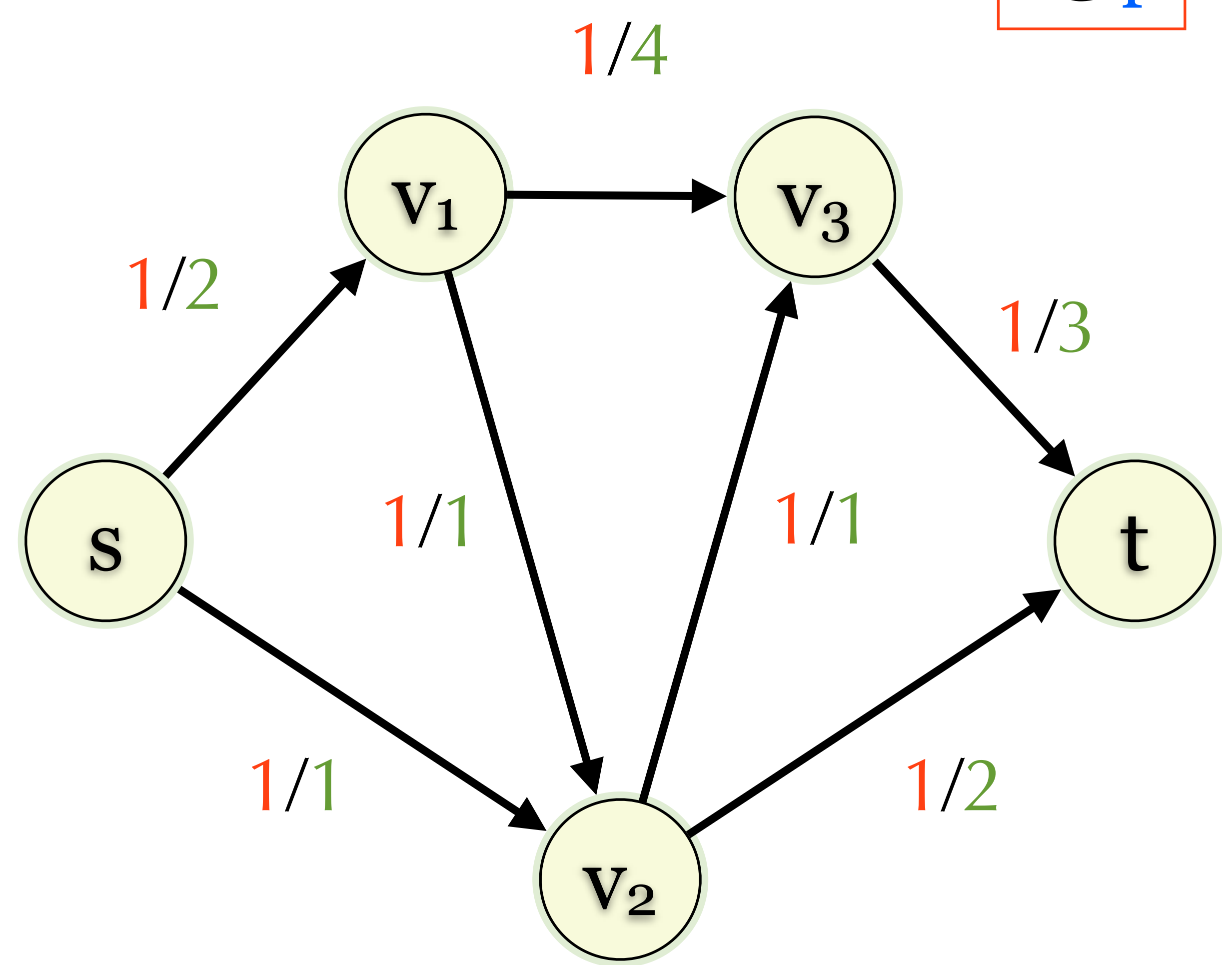
- For each edge $e=(u,v)\in E$ s.t. $f(u,v)>0$, build edges (u,v), $(v,u)\in E_f$ s.t.
  - $c_f(u,v)=c(u,v)-f(u,v)$ weight $w(u,v)$
  - $c_f(v,u)=f(u,v)$ weight $-w(u,v)$
- Multiple edges: Add dummy vertices and edges.
- Residual network G_f is (V,E_f) with capacity c_f.

Example

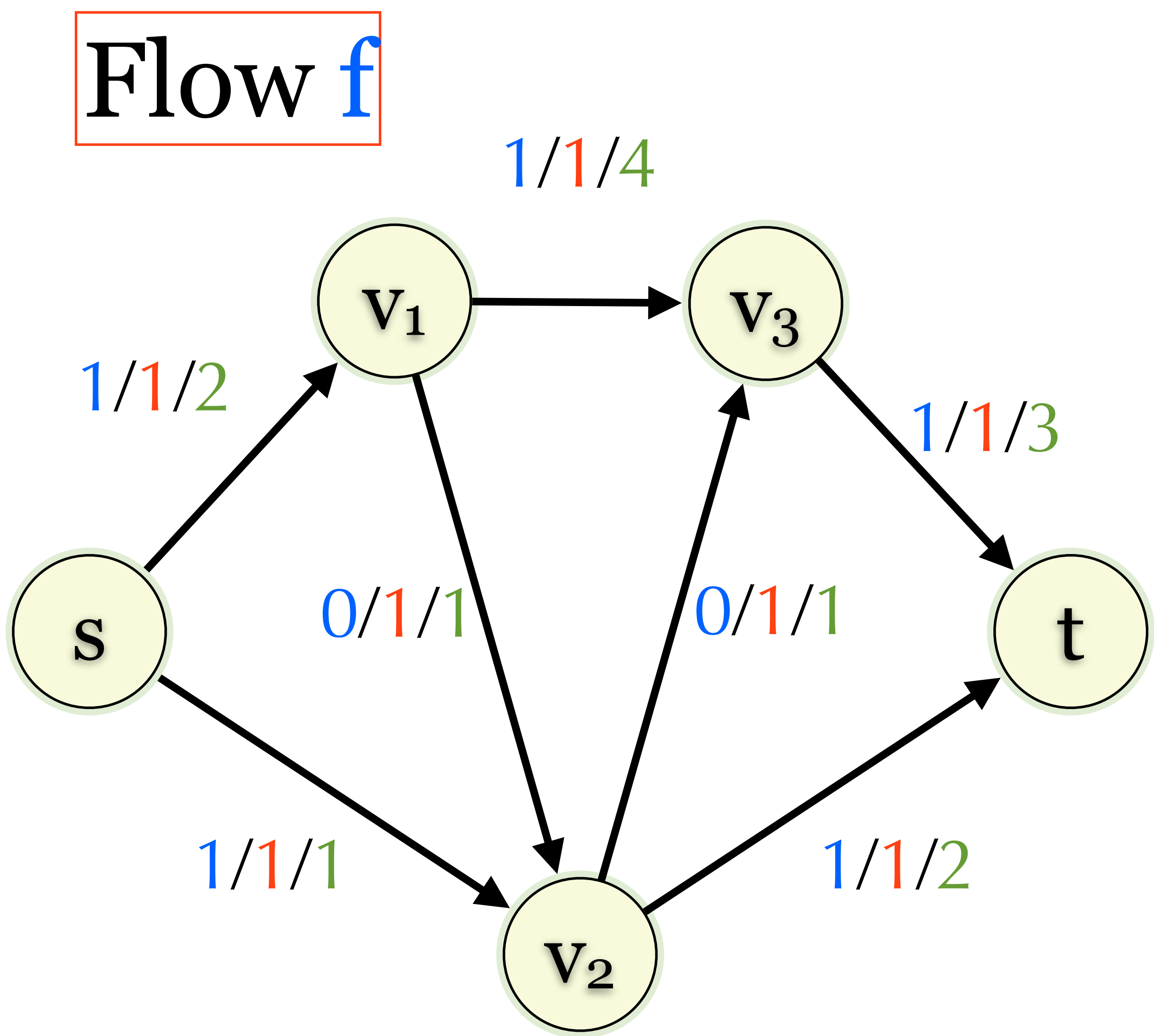
Flow f



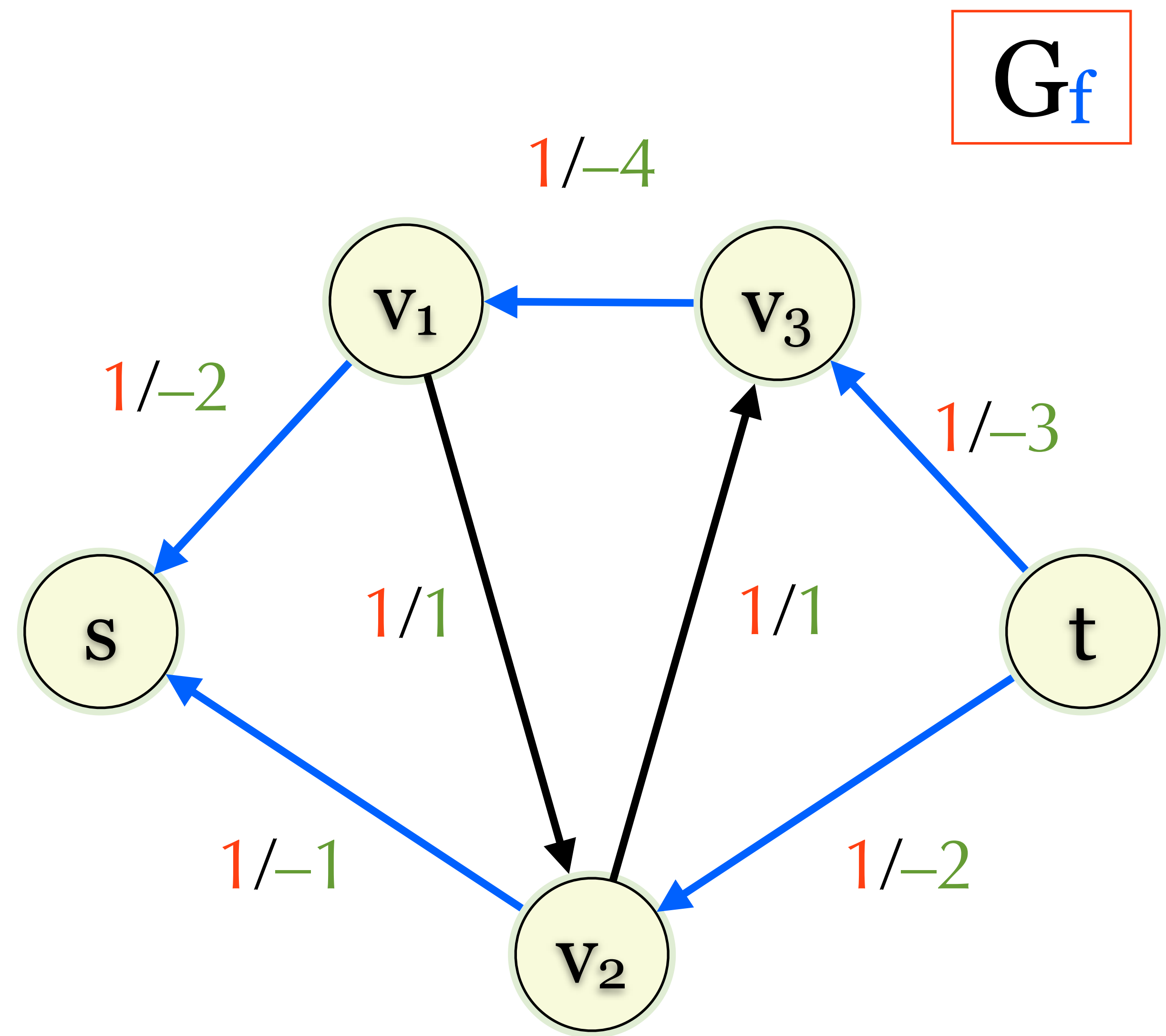
G_f



Example



Flow f
1/1/4
1/1/2
1/1/3
0/1/1
0/1/1
1/1/1
1/1/2
s
v1
v2
v3
t



Gf
1/–4
1/–2
1/–3
1/1
1/1
1/–1
1/–2
s
v1
v2
v3
t

Successive Shortest Path

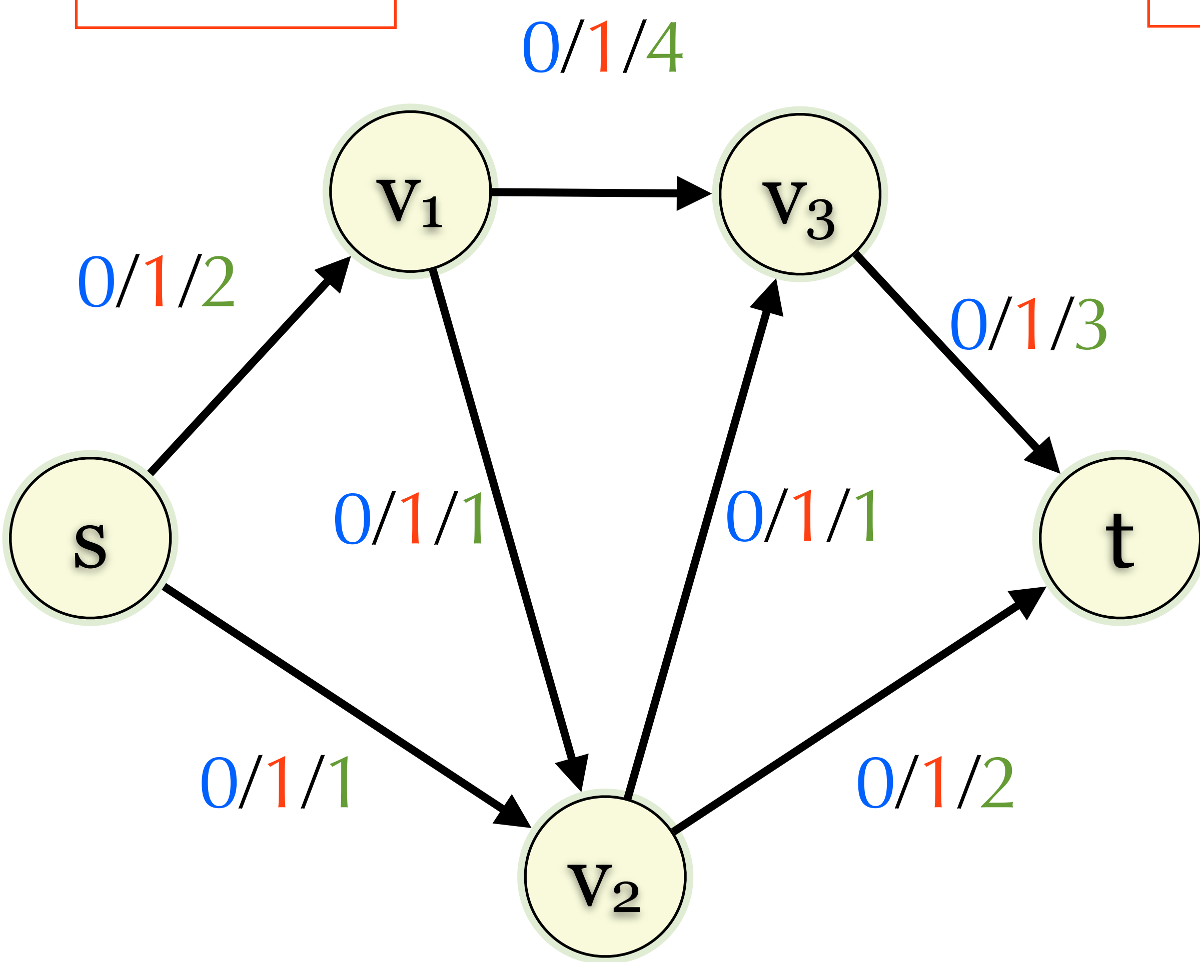
- Like Ford-Fulkerson, we continue finding augmenting path, until $|f|=f^*$.
- Always find the augmenting path with minimum weight.
  - $c_f(u,v)\neq 0$ iff $(u,v)\in E_f$
  - By Bellman-Ford! Not by Dijkstra's!
- Thm: SSP doesn't generate negative cycle.
  - 2pts

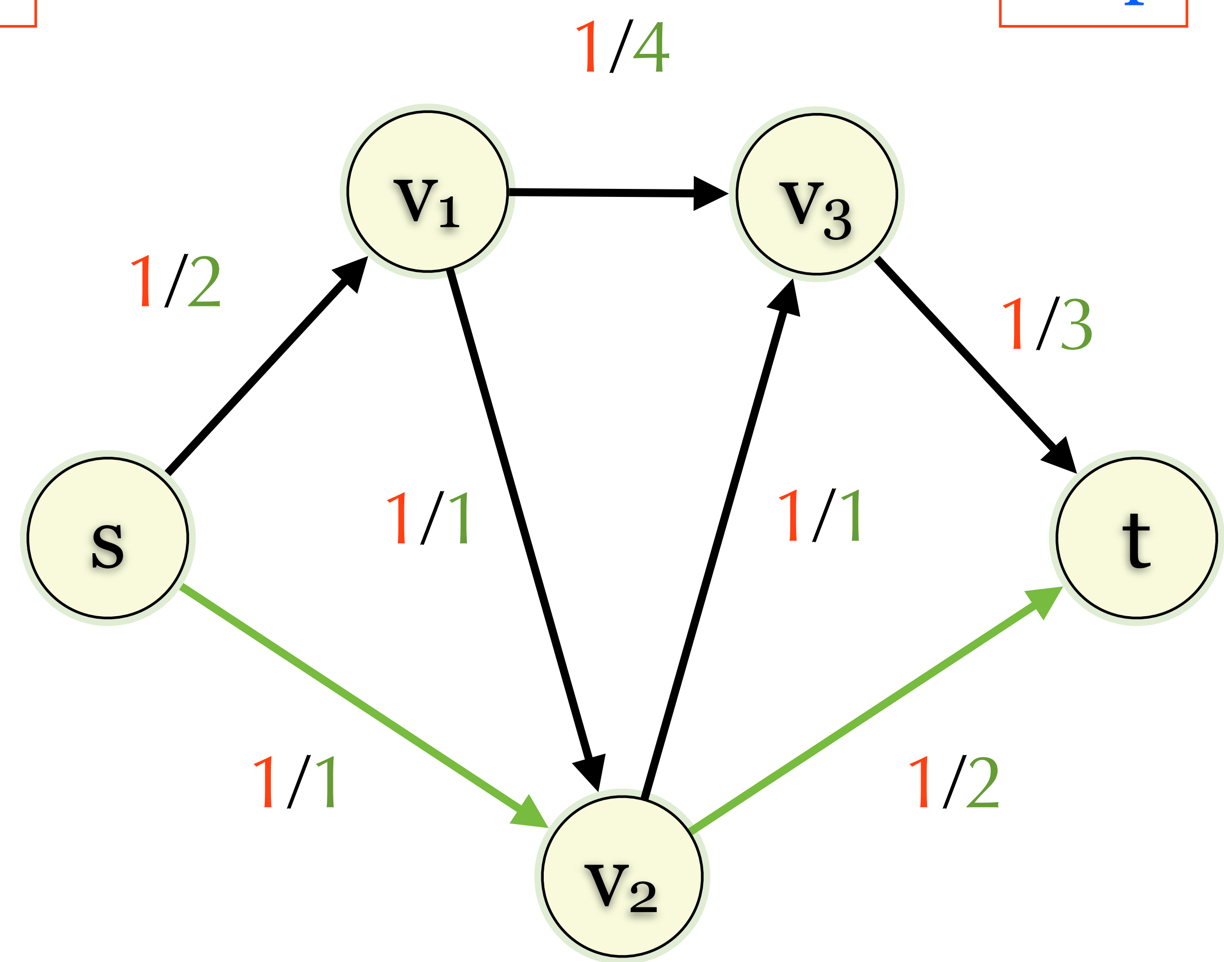
Example

Flow f

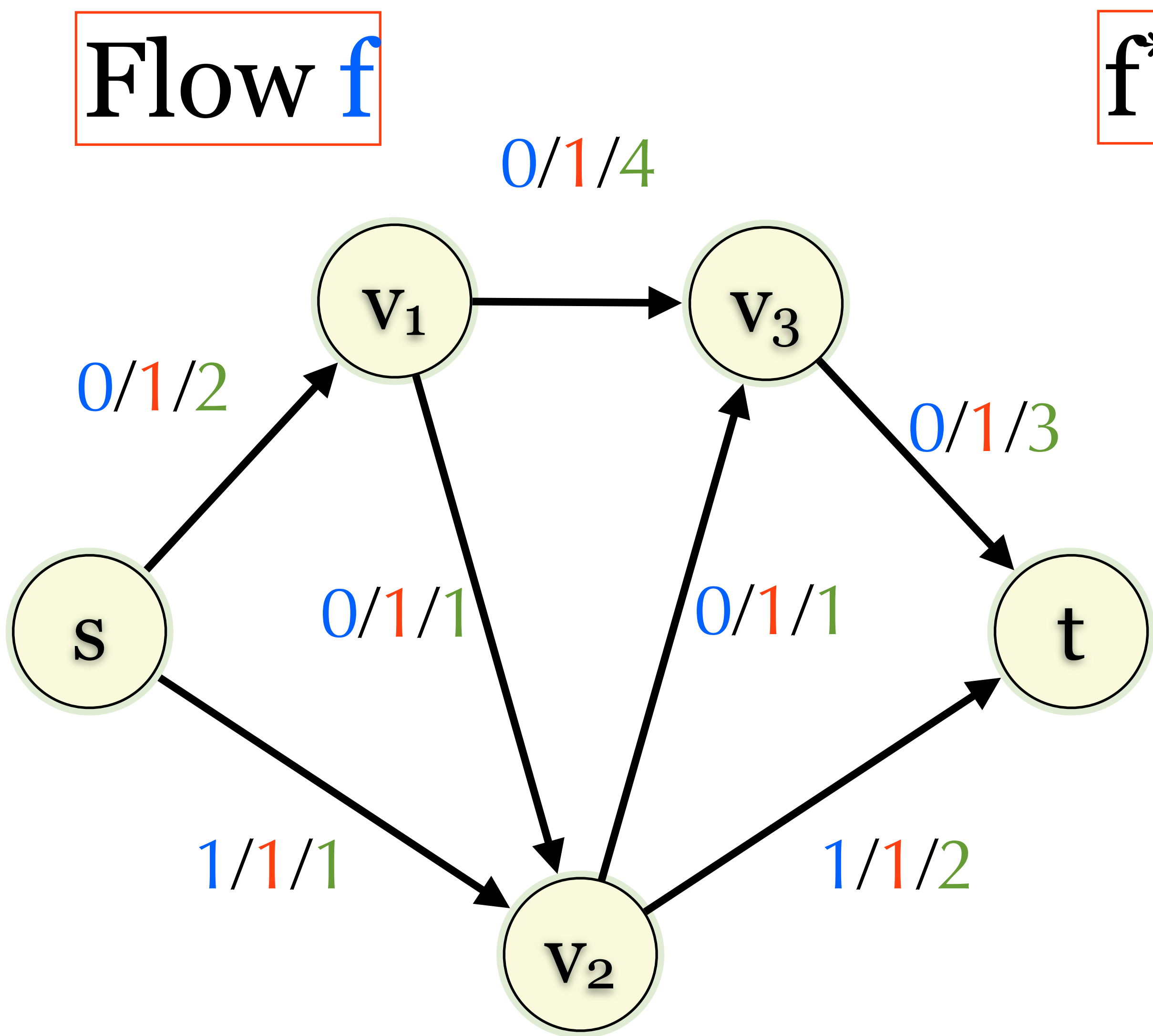
$f^*=2$

G_f

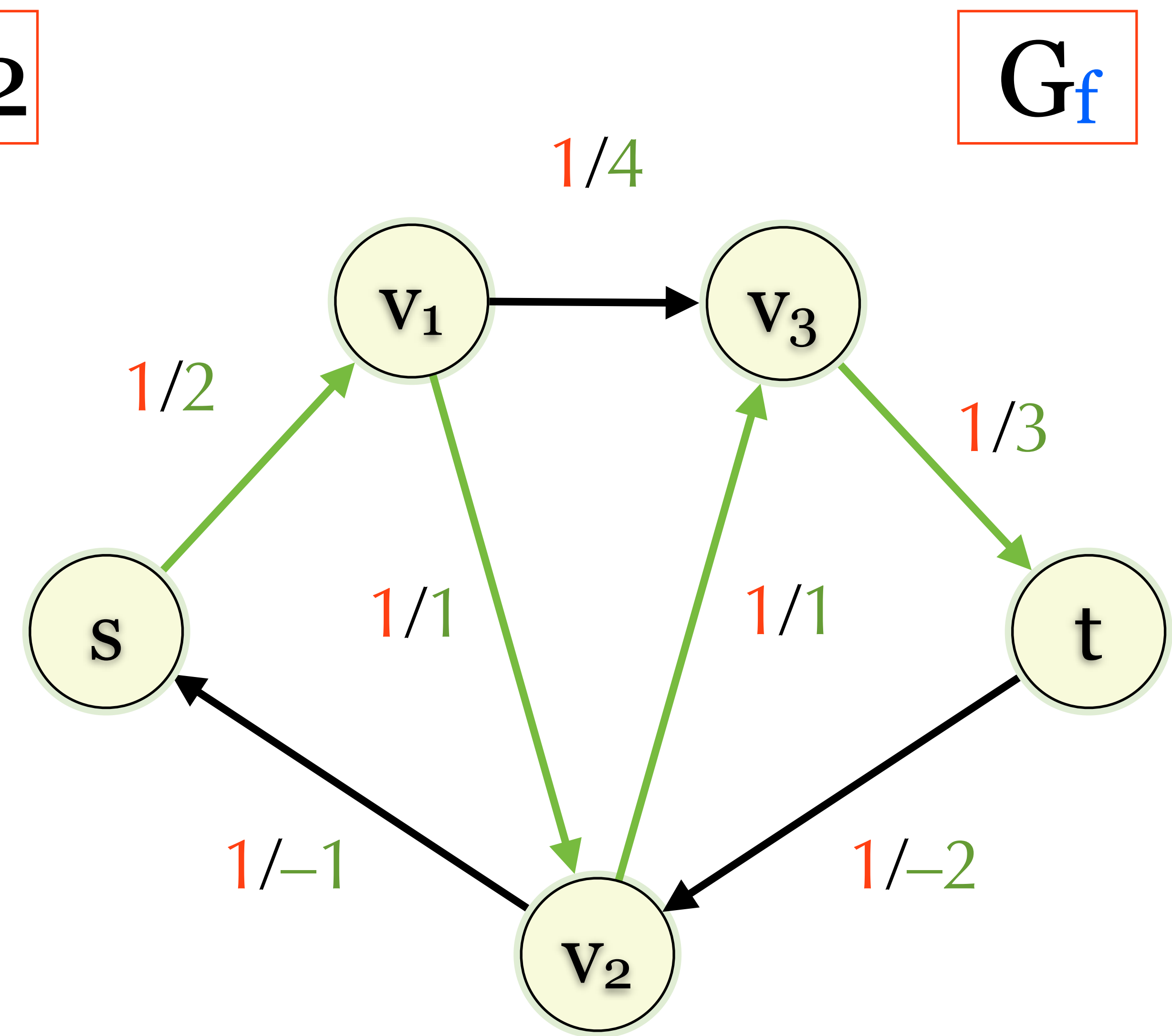




Example

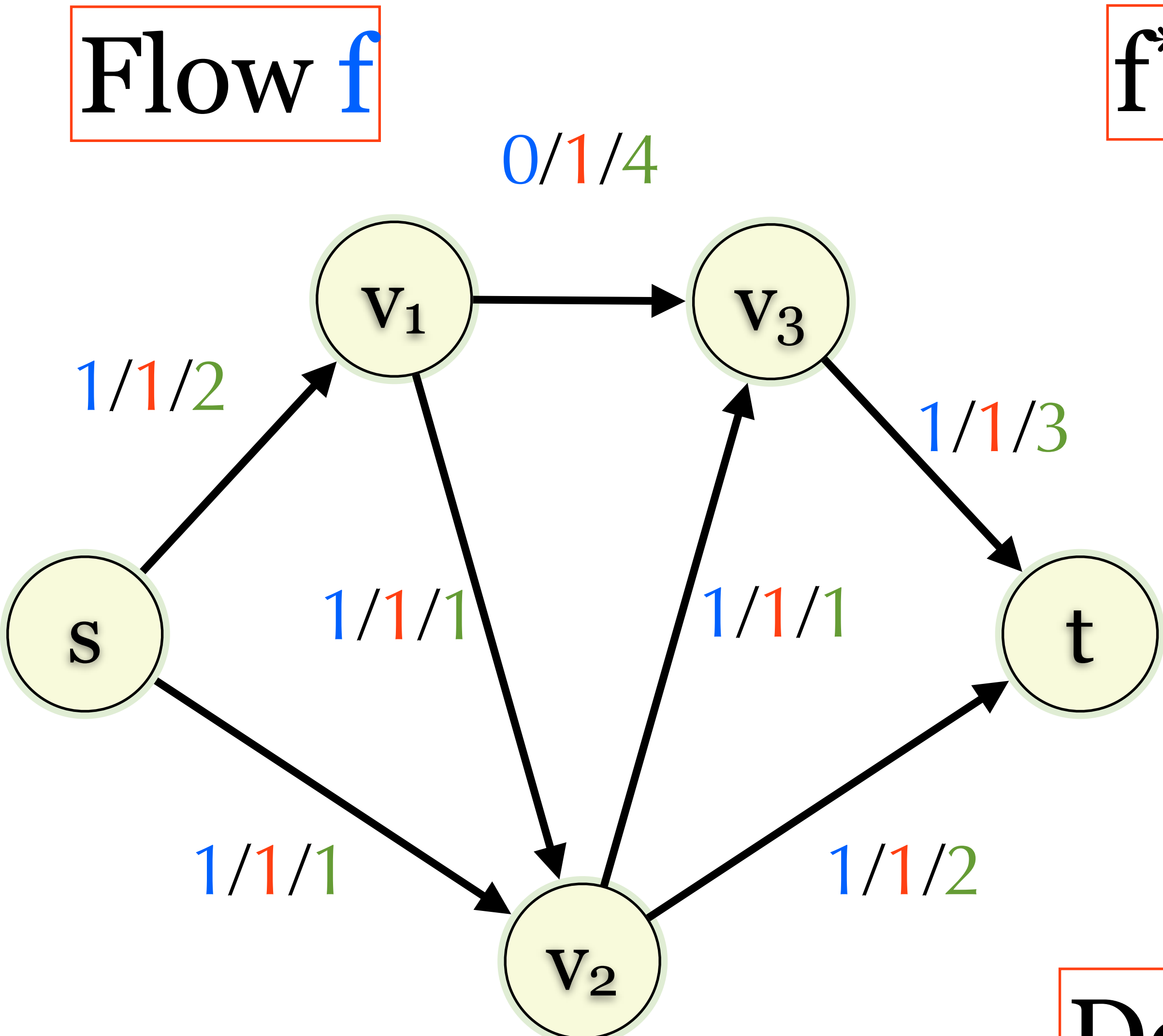

Flow f
0/1/4
0/1/2
0/1/3
0/1/1
0/1/1
1/1/1
1/1/2
s
v1
v2
v3
t

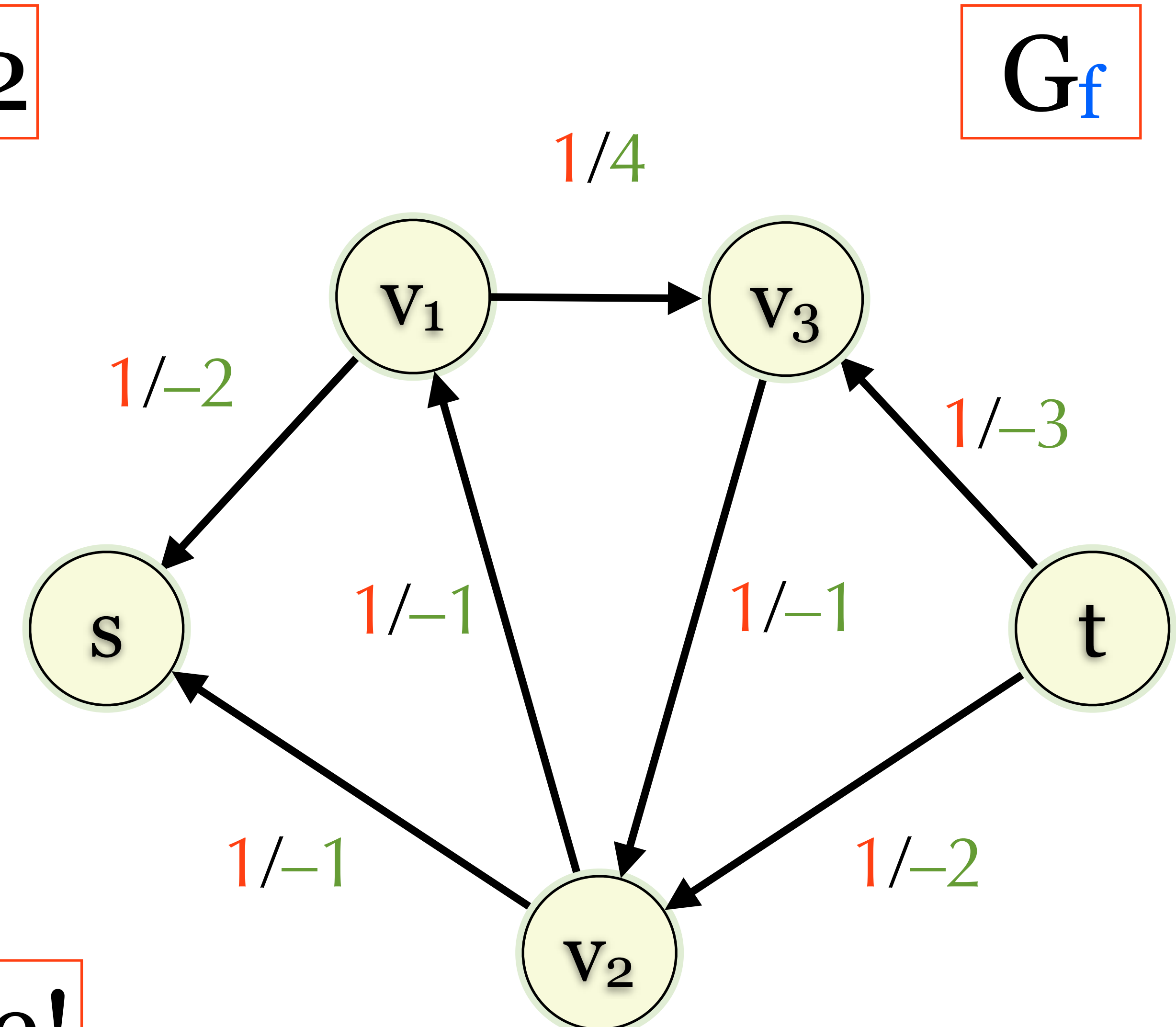
$f^* = 2$


Gf
1/4
1/2
1/3
1/1
1/1
1/–1
1/–2
s
v1
v2
v3
t

Example

$f^* = 2$





Done!

Neg Cycle Cancellation

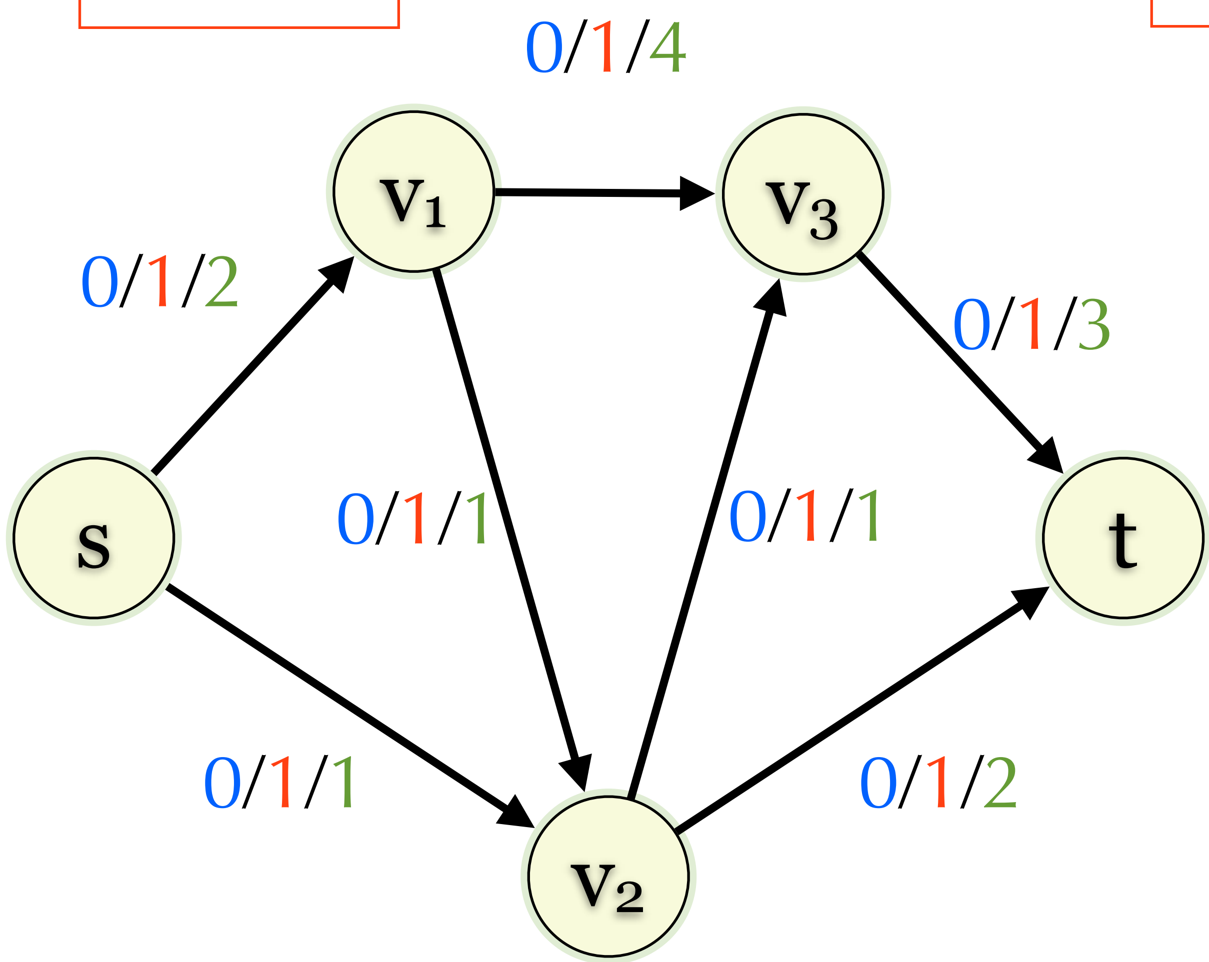
- Run Edmond-Karp until $|f|=f^*$.
- Keep finding negative cycles.
  - Augment the negative cycle.
- Using Karp's algorithm
  - Always cancel minimum mean cycle
  - Can be done in polynomial time!
- Bonus: Karp's algorithm (3 pts)
  - Find the minimum mean cycle in a directed graph.

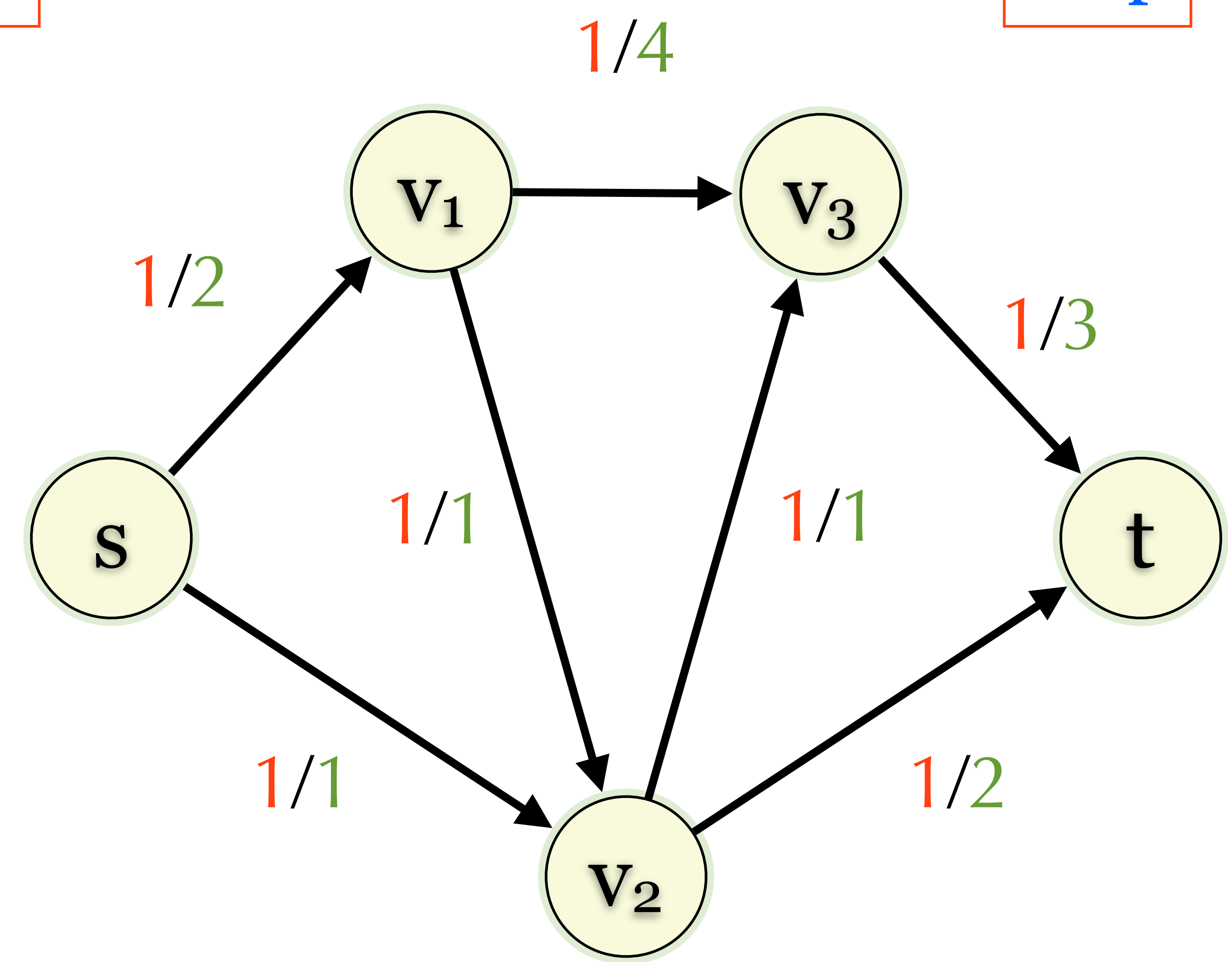
Example

Flow f

$f^*=2$

G_f

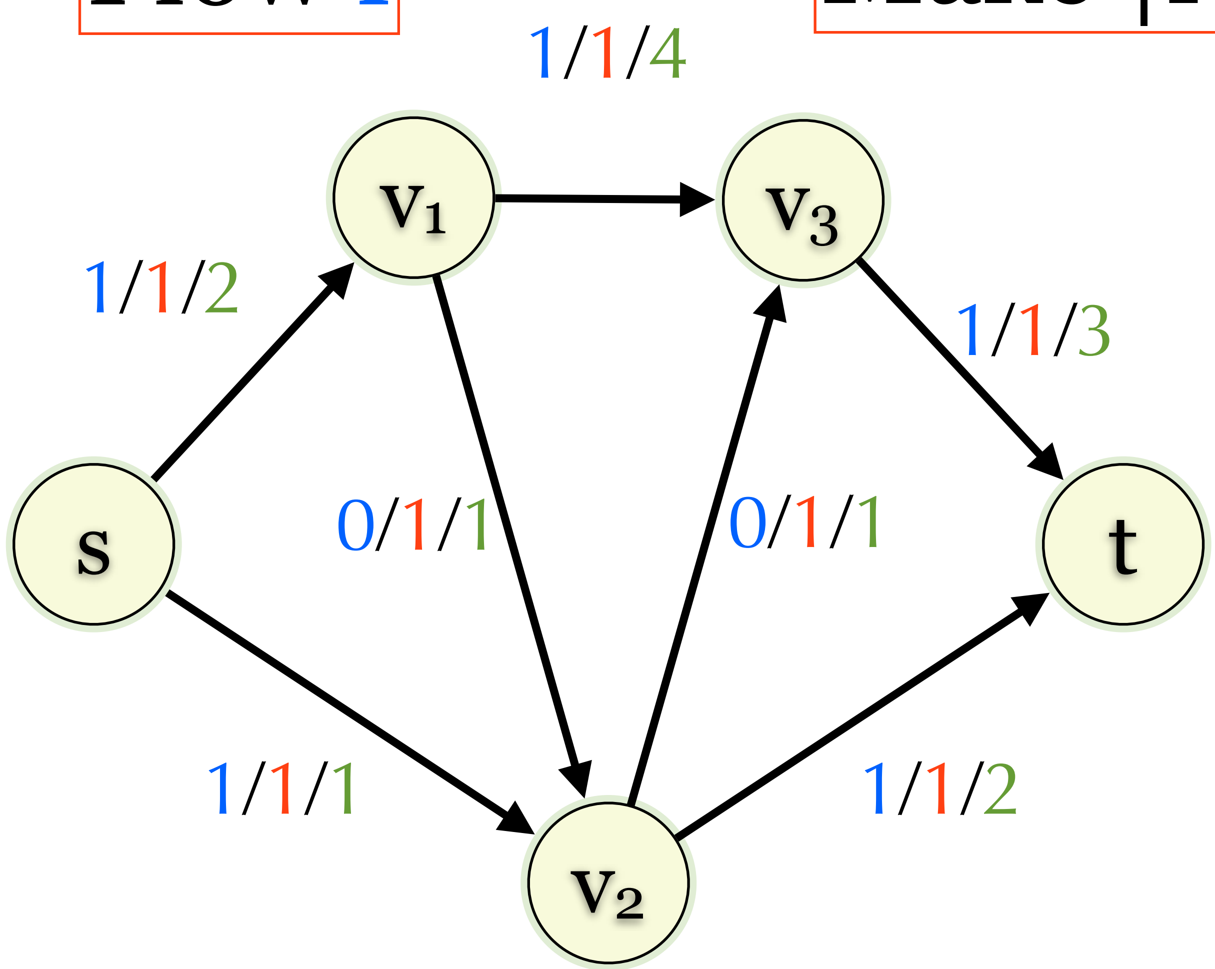




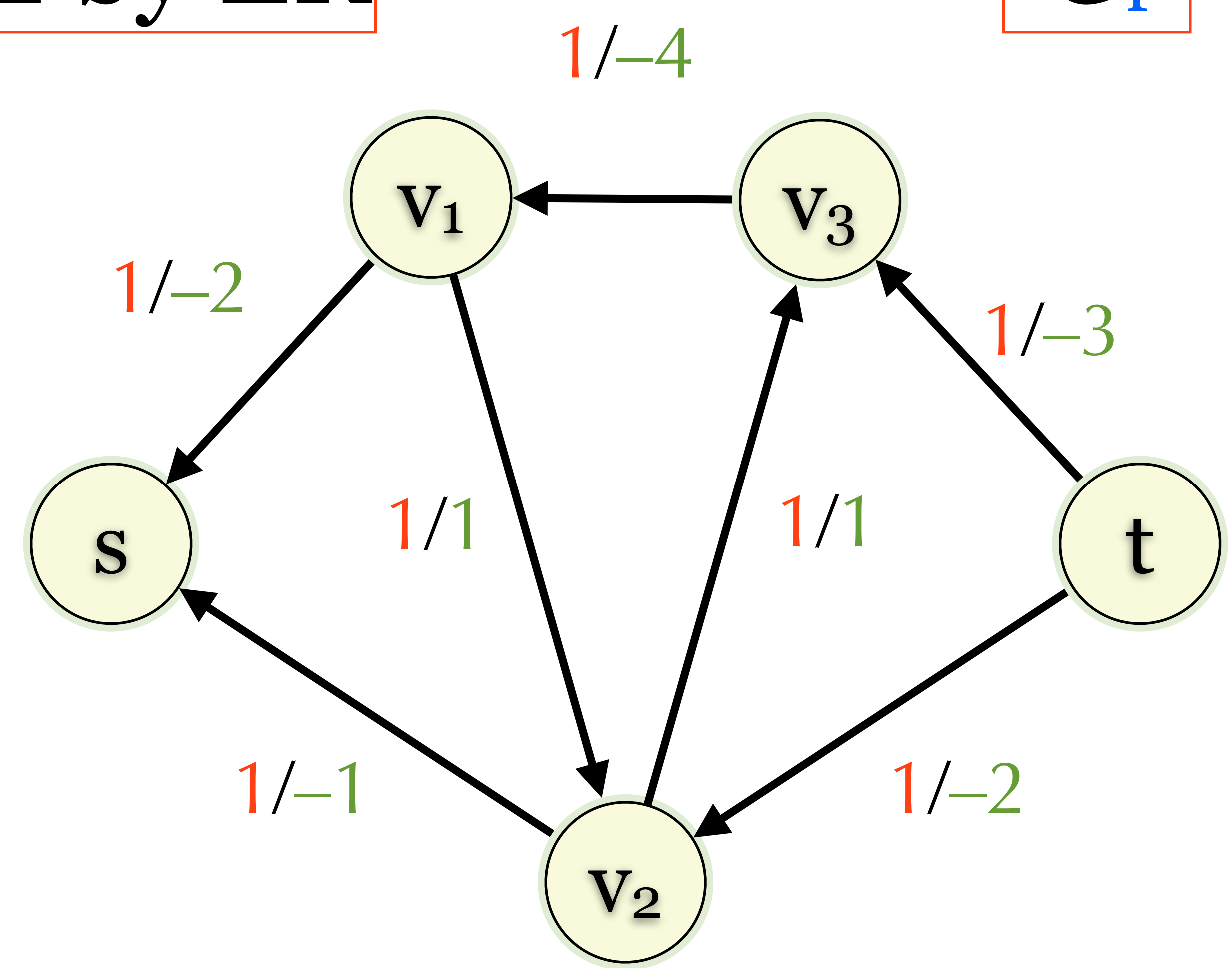
Example

Flow f

Make |f|=2 by EK

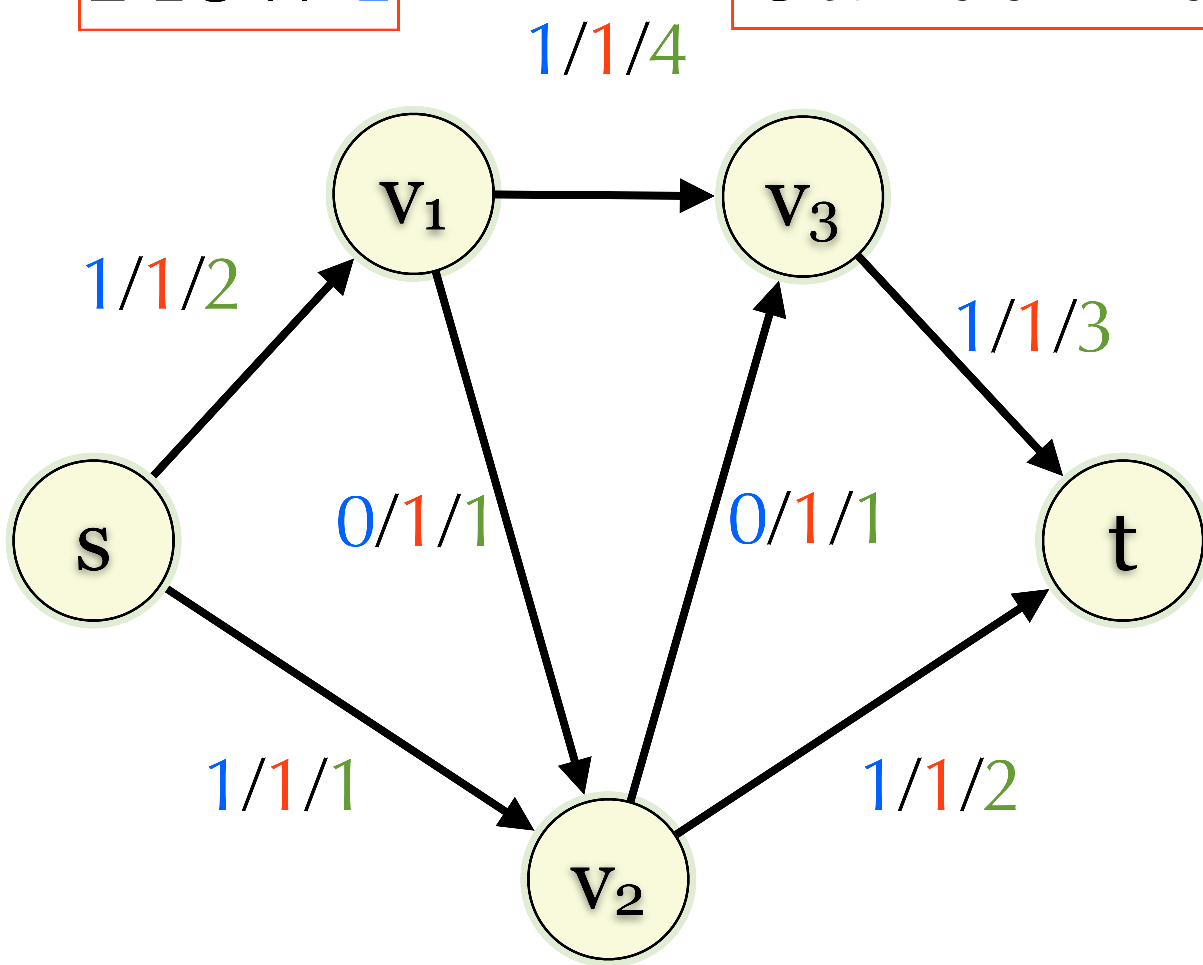


G_f



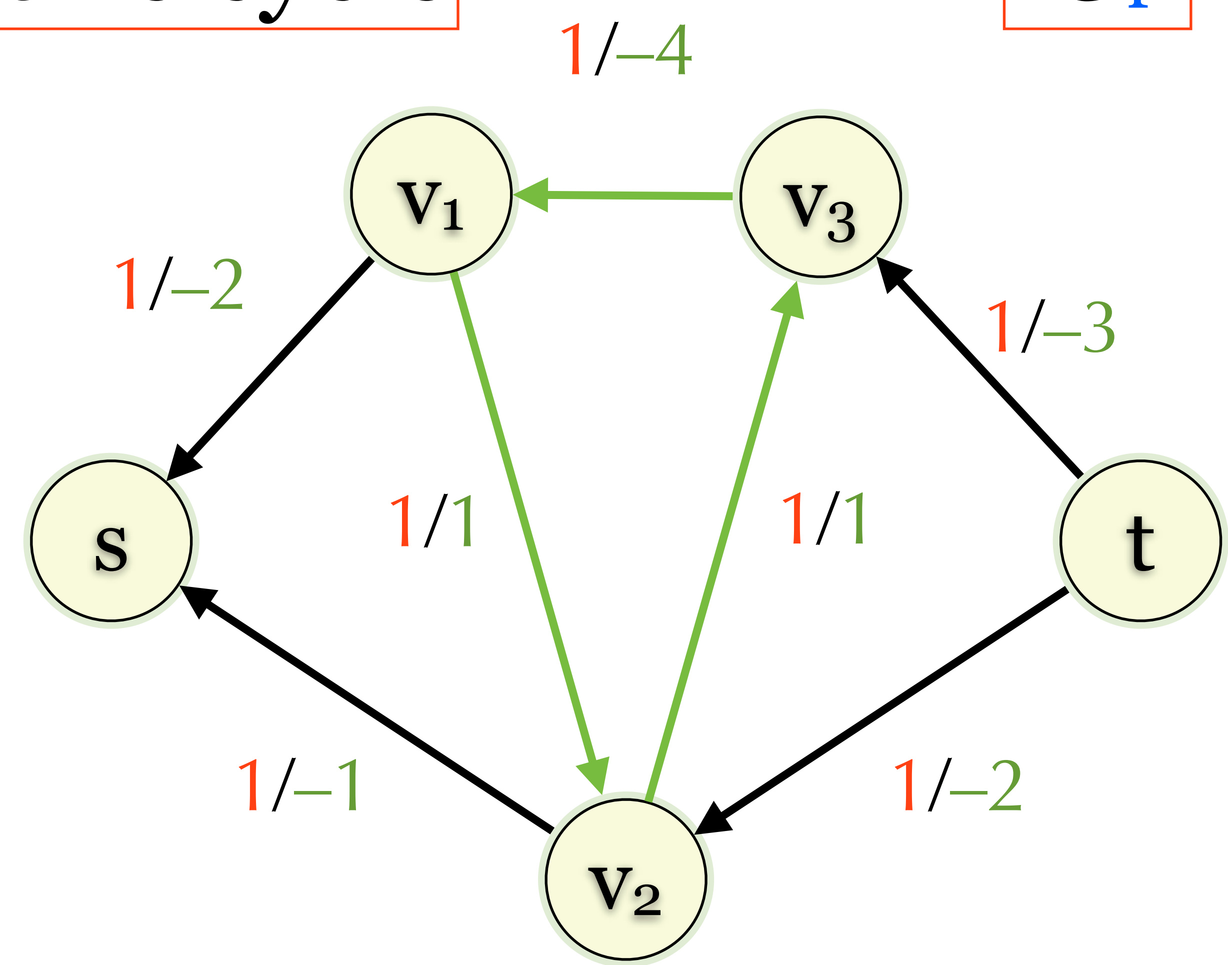
Example

Flow f



Cancel negative cycle

G_f

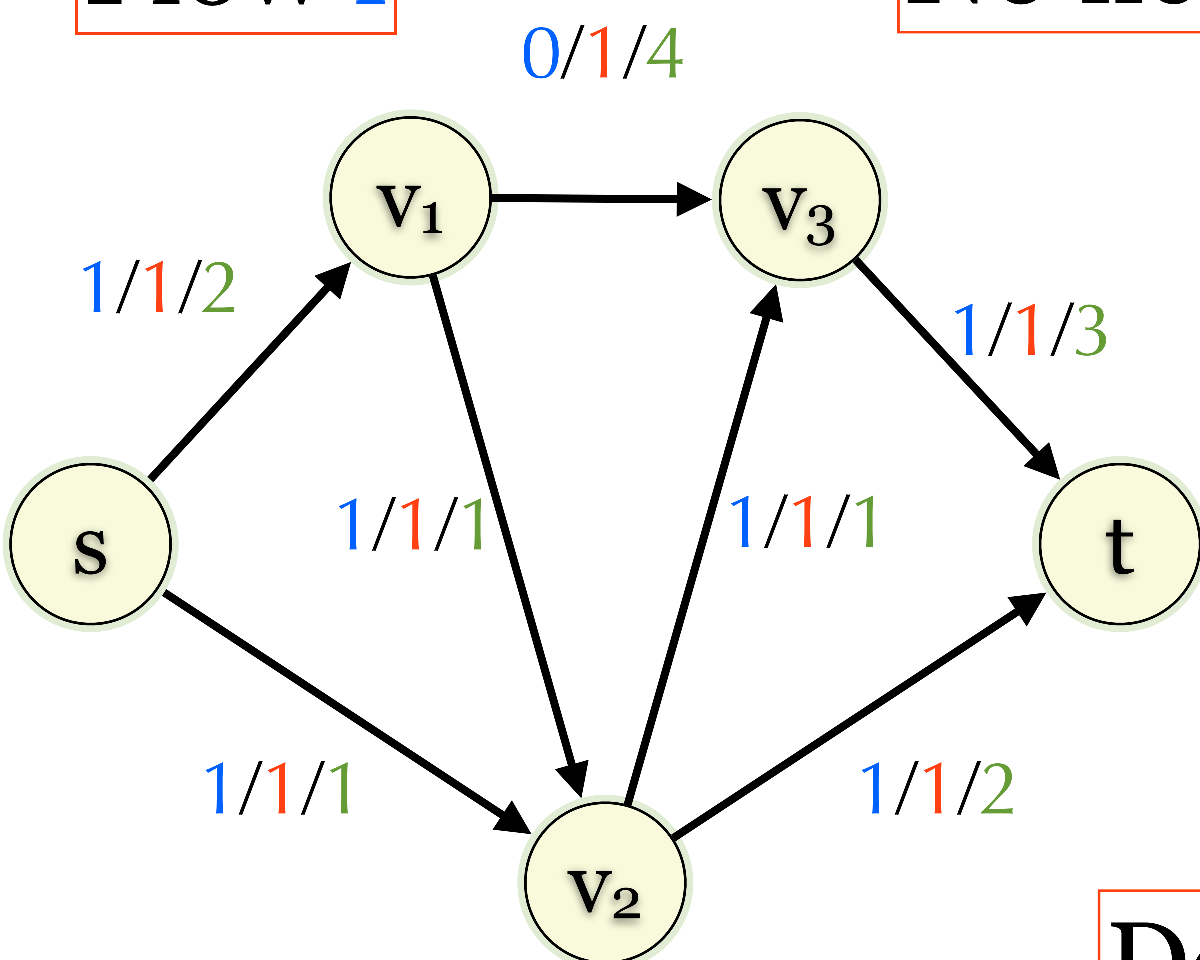


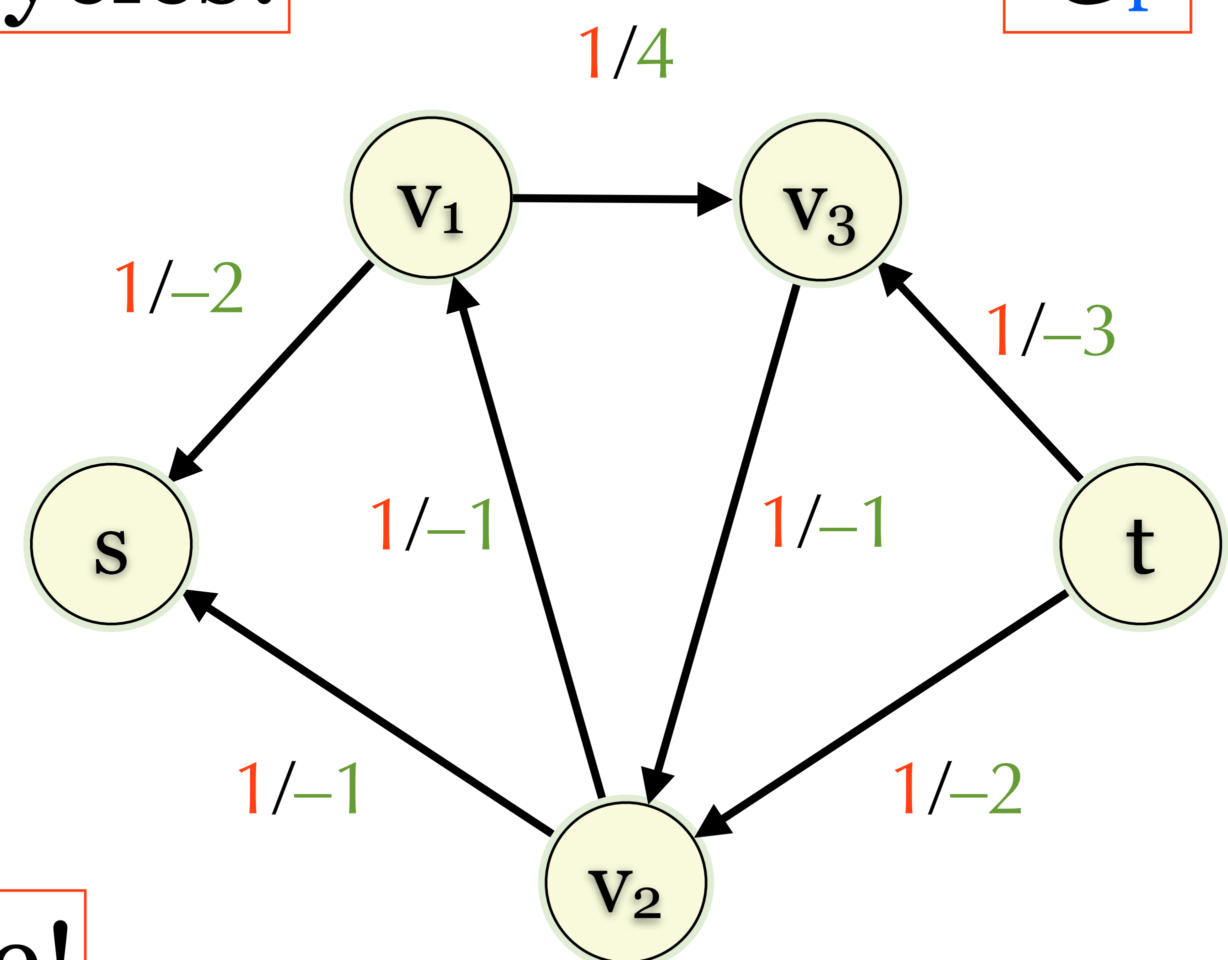
Example

Flow f

No neg cycles!

G_f





Done!

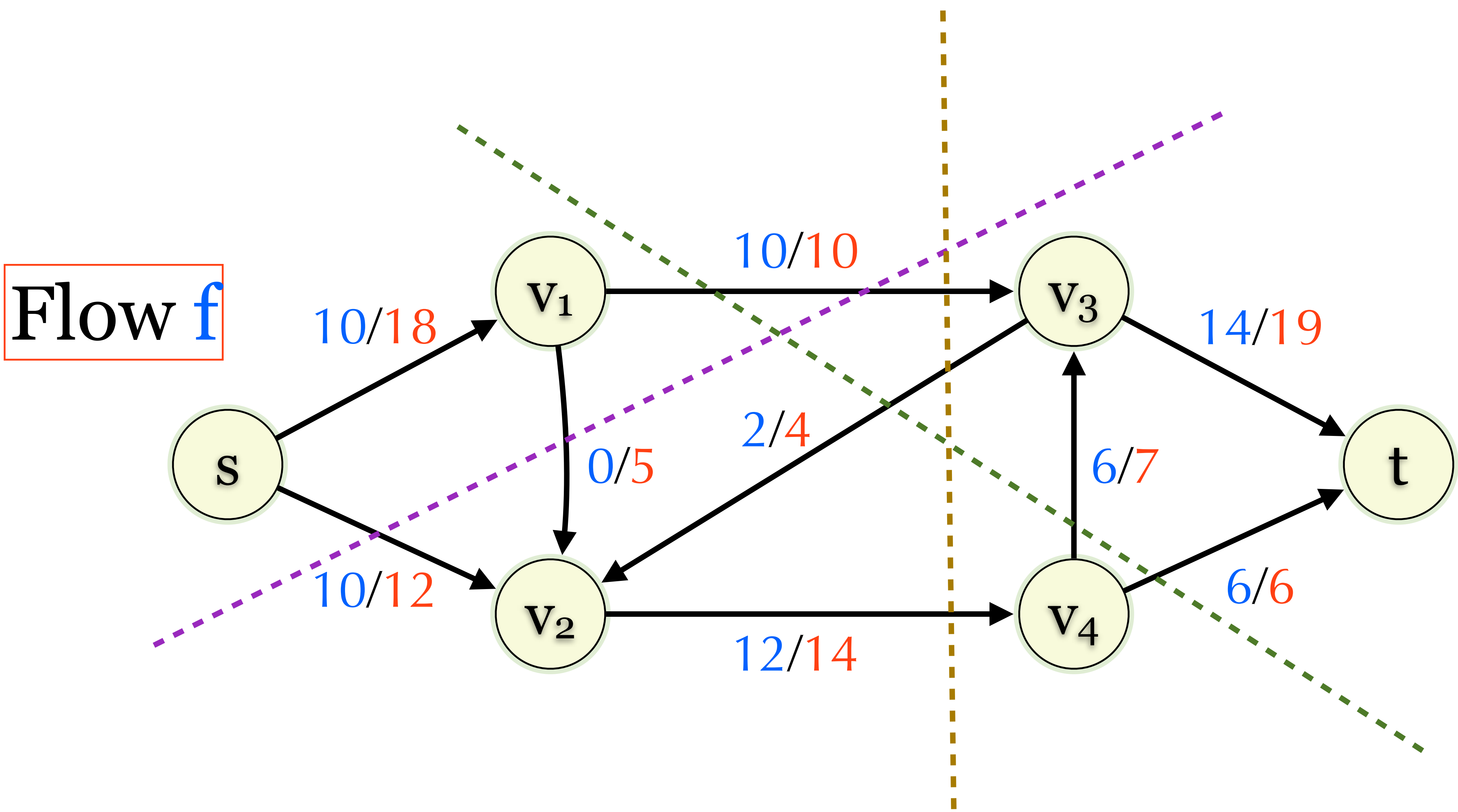
Lemma 26.4

- Let f be a flow in a flow network G with source s and sink t, and let (S,T) be any cut of G. Then the net flow across (S,T) is $f(S,T)=\Sigma_{(u,v)\in S\times T}f(u,v)-\Sigma_{(u,v)\in T\times S}f(u,v)=|f|$.
- Proof:
- Key tool: Flow conservation
  - $\forall v\in V\setminus\{s,t\}$, $\Sigma_{(u,v)\in E}f(u,v)=\Sigma_{(v,w)\in E}f(v,w)$.
  - $\forall v\in V\setminus\{s,t\}$, $\Sigma_{u\in V}f(u,v)-\Sigma_{u\in V}f(v,u)=0$.

Proof

- $|f|=\Sigma_{u\in V}f(s,u)$
  $=\Sigma_{u\in V}f(s,u)+\Sigma_{v\in S\setminus\{s\}}(\Sigma_{u\in V}f(v,u)-\Sigma_{u\in V}f(u,v))$
  $=\Sigma_{v\in S}(\Sigma_{u\in V}f(v,u)-\Sigma_{u\in V}f(u,v))$ $\Sigma_{u\in V}f(u,s)=0$
  $=\Sigma_{v\in S}\Sigma_{u\in V}f(v,u)-\Sigma_{v\in S}\Sigma_{u\in V}f(u,v)$
  $=\Sigma_{v\in S}\Sigma_{u\in S}f(v,u)+\Sigma_{v\in S}\Sigma_{u\in T}f(v,u)$
  $-\Sigma_{v\in S}\Sigma_{u\in S}f(u,v)-\Sigma_{v\in S}\Sigma_{u\in T}f(u,v)$
  $=\Sigma_{v\in S}\Sigma_{u\in T}f(v,u)-\Sigma_{v\in S}\Sigma_{u\in T}f(u,v)$
  $=\Sigma_{(u,v)\in S\times T}f(u,v)-\Sigma_{(u,v)\in T\times S}f(u,v)$
  $f(u,v)=0$ if $(u,v)\notin E$
  $=f(S,T)$

Example



Min-Cut Max-Flow Thm

- If f is a flow in a flow network $G=(V,E)$ with source s and sink t, then the following conditions are equivalent:
  1. f is a maximum flow in G.
  2. The residual network G_f contains no augmenting paths.
  3. $|f|=c(S,T)$ for some s-t cut (S,T) of G.

(1) implies (2)

- (1) implies (2) iff not (2) implies not (1).
- If there is an augmenting path $\langle v_0,\ldots,v_k\rangle$ in G_f, then $c(v_{i-1},v_i)>0$ for $0<i\leq k$.
- We can augment f by f' where
  - $f'(e)=0$ if e is not on the path
  - $f'(e)=\min_{0<i\leq k} c(v_{i-1},v_i)$
- f is not a maximum flow.

(2) implies (3)

- Let S be the set of vertices reachable without using (u,v) s.t. $c_f(u,v)=0$ from s in G_f. Let $T=V\setminus S$.
- For every $(u,v)\in(S\times T)\cap E$, $f(u,v)=c(u,v)$.
  - If not, then $c_f(u,v)>0$ and $v\in S$. Contradiction
- For every $(u,v)\in(T\times S)\cap E$, $f(u,v)=0$.
  - If not, then $c_f(v,u)>0$ and $u\in S$. Contradiction
- $|f|=f(S,T)$ Lemma 26.4
  $=\Sigma_{(u,v)\in S\times T}f(u,v)-\Sigma_{(u,v)\in T\times S}f(u,v)$ $f(e)=0$ if $e\notin E$
  $=\Sigma_{(u,v)\in S\times T}c(u,v)-0$
  $=c(S,T)$

(3) implies (1)

- Goal: For every s-t cut (S,T) & flow f, $c(S,T) \geq |f|$.
  - If $c(S,T)=|f|$, then f is a maximum flow and (S,T) is a minimum cut.
- $|f|=f(S,T)$ Lemma 26.4
  $=\Sigma_{(u,v)\in(S\times T)\cap E}f(u,v)-\Sigma_{(u,v)\in(T\times S)\cap E}f(u,v)$
  $\leq\Sigma_{(u,v)\in(S\times T)\cap E}c(u,v)-0$
  $=c(S,T)$

Edmonds-Karp: Time Complexity

- Time complexity: $O(|V||E|^2)$
  - Augmentation takes $O(|V|+|E|)$
  - Total augmentation: $O(|V||E|)$
- Proof idea:
  - For every edge (u,v), we only set f(u,v) to c(u,v) or 0 $O(|V|)$ times.
    - Each augmentation changes f(u,v) at most once. $|E| \times O(|V|) = O(|V||E|)$

Lemma 26.7

- If the Edmonds-Karp algorithm is run on a flow network $G=(V,E)$ with source s and sink t, then for all vertices $v \in V \setminus \{s,t\}$, the shortest-path distance $\delta_f(s,v)$ in the residual network G_f monotonically increases with each flow augmentation.

Proof

- BWOC. Assume the lemma is true before we augment f into f'. Let v be the vertex of minimum $\delta_{f'}(s,v)$ s.t. $\delta_{f'}(s,v)<\delta_f(s,v)$.
- $v \neq s$: $\exists u \in V$ s.t. $\delta_{f'}(s,u)+1=\delta_{f'}(s,v)$. $\delta_{f'}(s,u)<\delta_{f'}(s,v)$
- Note: $\delta_f(s,u) \leq \delta_{f'}(s,u)$
- If $(u,v) \in E_f$

  $\delta_f(s,v) \leq \delta_f(s,u)+1$ triangle inequality

  $\leq \delta_{f'}(s,u)+1$

  $=\delta_{f'}(s,v)$

  $< \delta_f(s,v)$ contradiction

Proof

- We have $(u,v) \notin E_f$
- Note: $(u,v) \in E_{f'}$
- (v,u) is on the augmenting path when we augment f into f'.
- $\delta_f(s,v) = \delta_f(s,u) - 1$ triangle inequality
  $\leq \delta_{f'}(s,u) - 1$
  $= \delta_{f'}(s,v) - 2$ contradiction

Theorem 26.8

- If the Edmonds-Karp algorithm is run on a flow network $G=(V,E)$ with source s and sink t, then the total number of flow augmentations performed by the algorithm is $O(|V||E|)$.

Proof

- If we set $f(u,v)$ to $c(u,v)$, then $(u,v) \notin E_f$.
- If we set $f(u,v)$ to 0, then $(v,u) \notin E_f$.
- Suppose we augment f to f'.
- To create (u,v) in the residual network, we have to augment (v,u).
  - $(u,v) \notin E_f$ and $(u,v) \in E_{f'}$: $\delta_f(s,v)+1=\delta_f(s,u)$
- To remove (u,v) in the residual network, we have to augment (u,v): set $f(u,v)=c(u,v)$ or $f(v,u)=0$
  - $(u,v) \in E_f$ and $(u,v) \notin E_{f'}$: $\delta_f(s,v)=\delta_f(s,u)+1$

Proof

- Suppose we augment the flow $f \to f' \to \ldots \to f'' \to f'''$.
- Consider (u,v): removed by augmenting f to f' and created by augmenting f'' into f'''.
  - $\delta_{f''}(s,u)=\delta_{f''}(s,v)+1 \geq \delta_f(s,v)+1=\delta_f(s,u)+2$
- This process increase the shortest distance by 2, but $\delta_{f^*}(s,u)>|V|-1$ implies $\delta_{f^*}(s,u)=\infty$ for flow f^*.
  - This process can happen at most O(|V|) times.
- But when we augment a flow, we always remove and create some edges in the residual network.
  - We can augment only O(|V||E|) times.