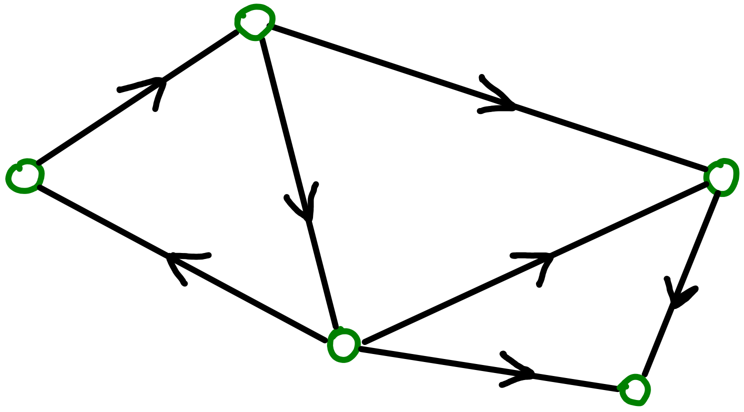
NETWORK FLOW - FLOW NETWORKS

NETWORK FLOW – FLOW NETWORKS

network = directed graph

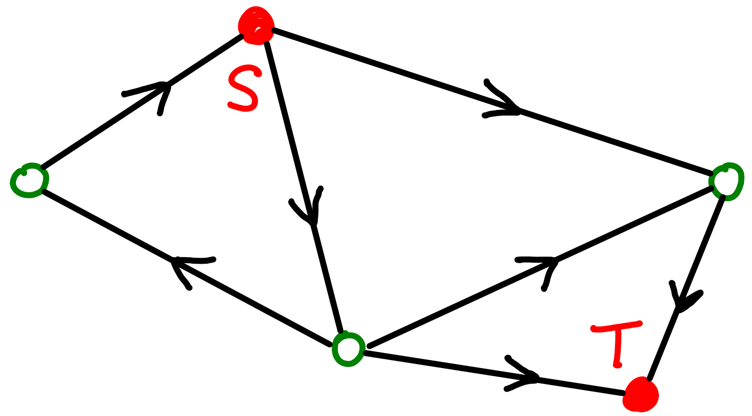


NETWORK FLOW - FLOW NETWORKS

network = directed graph

S : supply

T : target

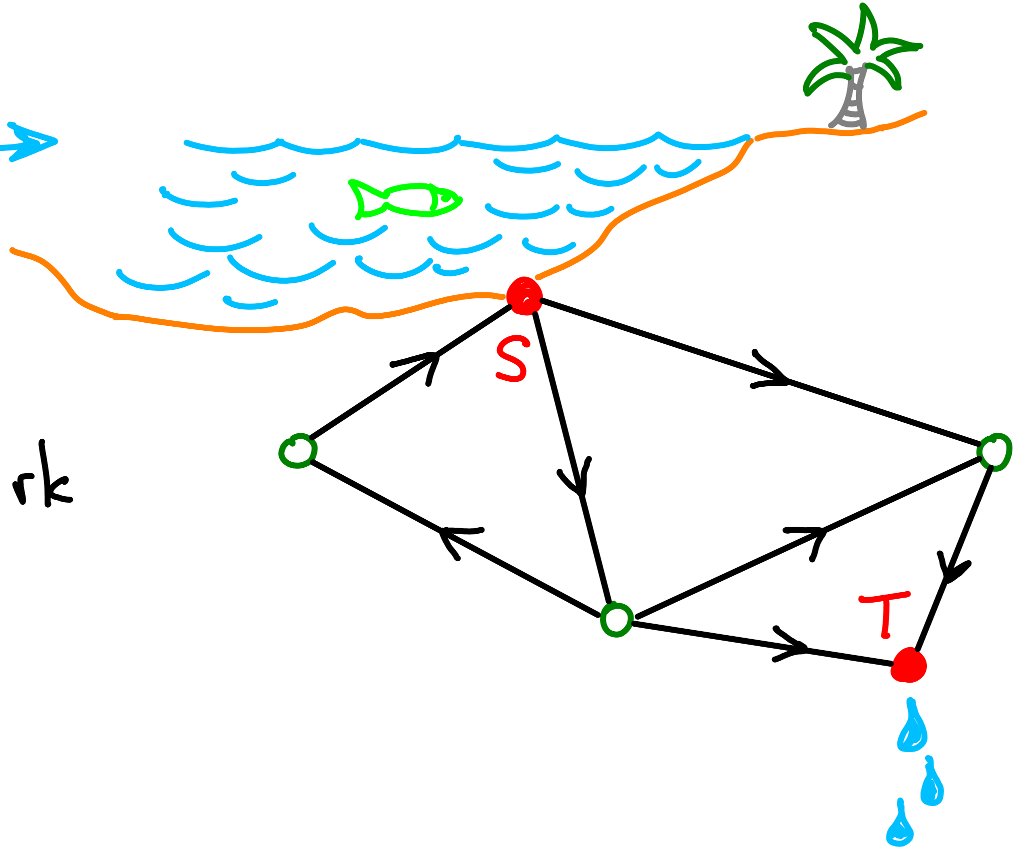


NETWORK FLOW – FLOW NETWORKS

network = directed graph

S : supply : unbounded input

T : target : output limited
only by network

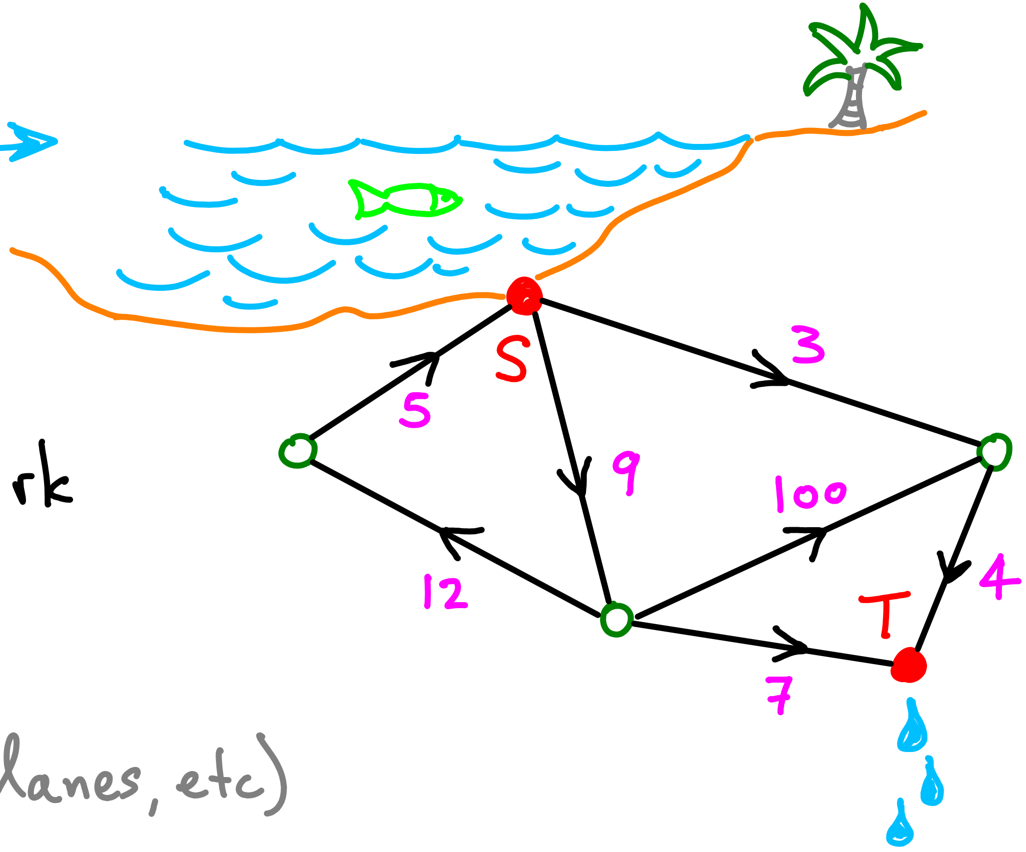


NETWORK FLOW - FLOW NETWORKS

network = directed graph

S : supply : unbounded input

T : target : output limited only by network



Each edge has a capacity

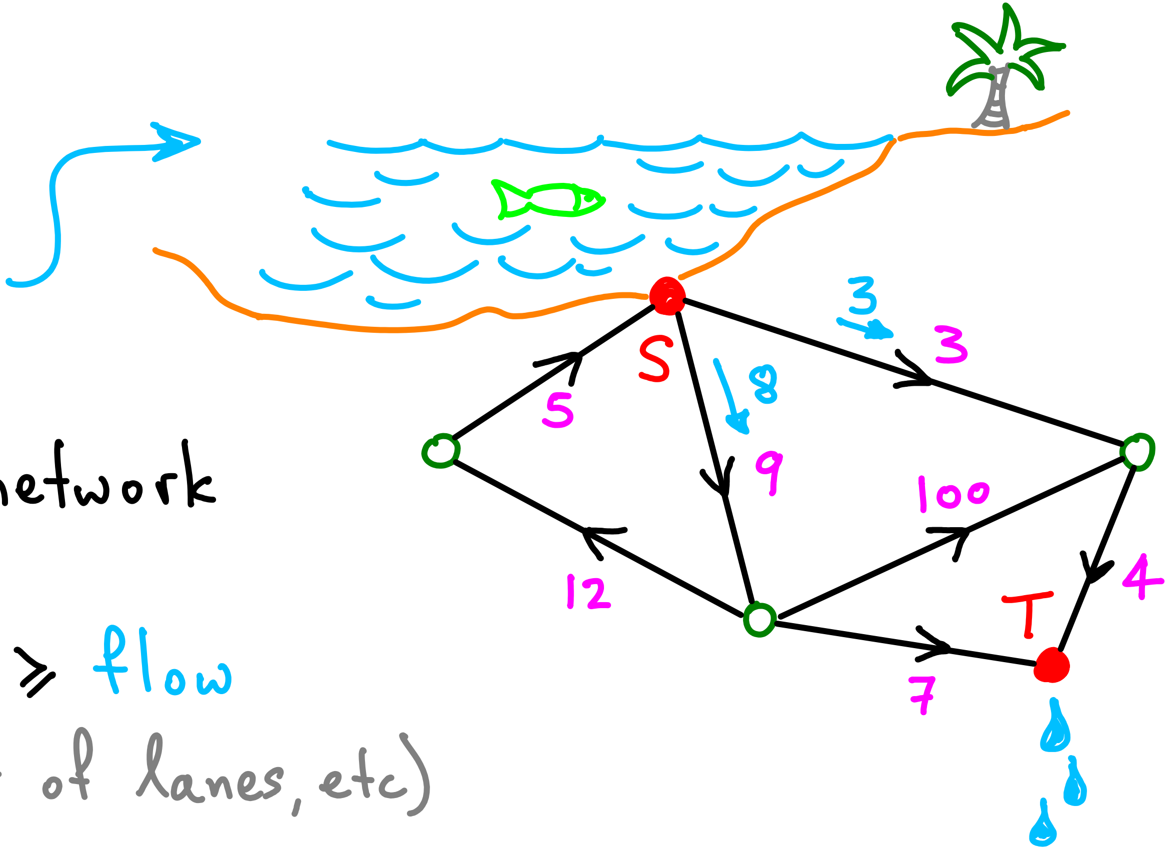
(pipe cross-section, number of lanes, etc)

NETWORK FLOW – FLOW NETWORKS

network = directed graph

S : supply : unbounded input

T : target : output limited only by network



Each edge has a capacity ≥ flow

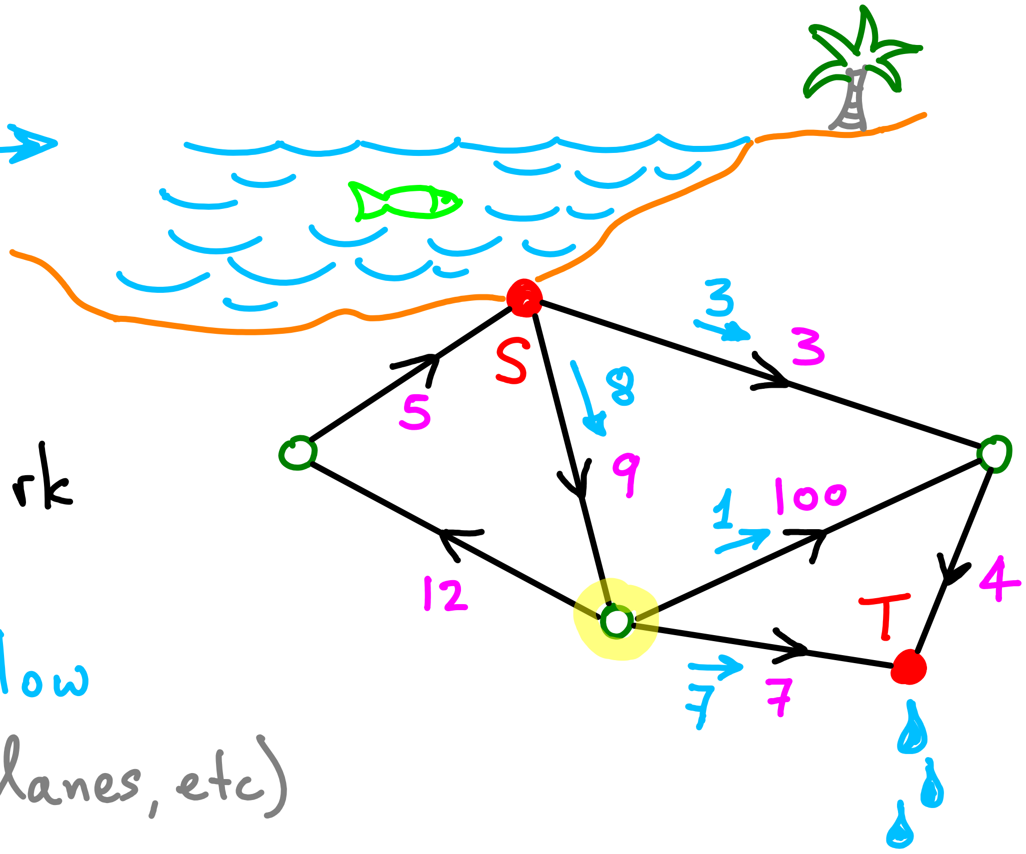
(pipe cross-section, number of lanes, etc)

NETWORK FLOW – FLOW NETWORKS

network = directed graph

S : supply : unbounded input

T : target : output limited only by network



Each edge has a capacity $\geq$ flow

(pipe cross-section, number of lanes, etc)

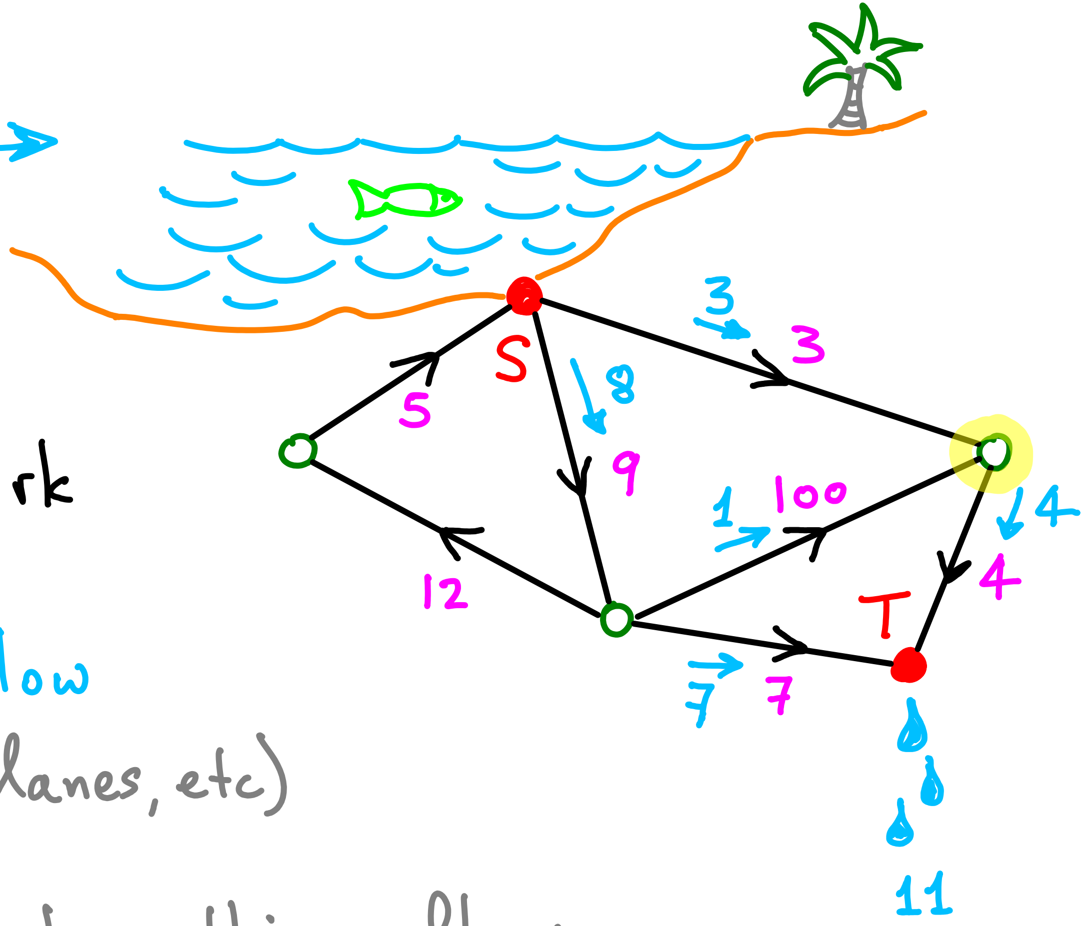
- Vertices : $\sum$ input $= \sum$ output → keep things flowing

NETWORK FLOW - FLOW NETWORKS

network = directed graph

S : supply : unbounded input

T : target : output limited
only by network

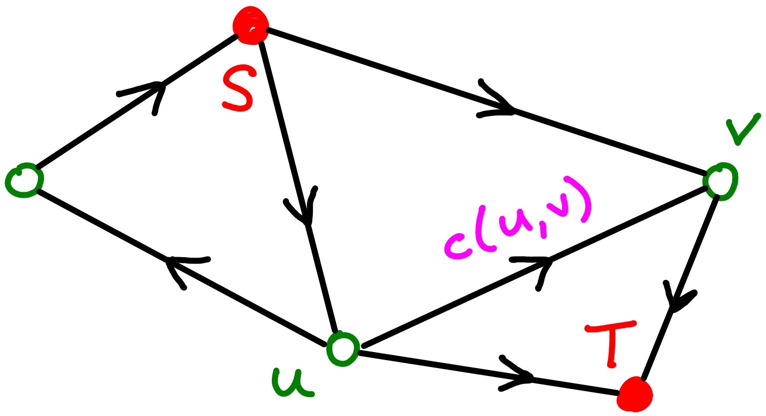


Each edge has a capacity ≥ flow

(pipe cross-section, number of lanes, etc)

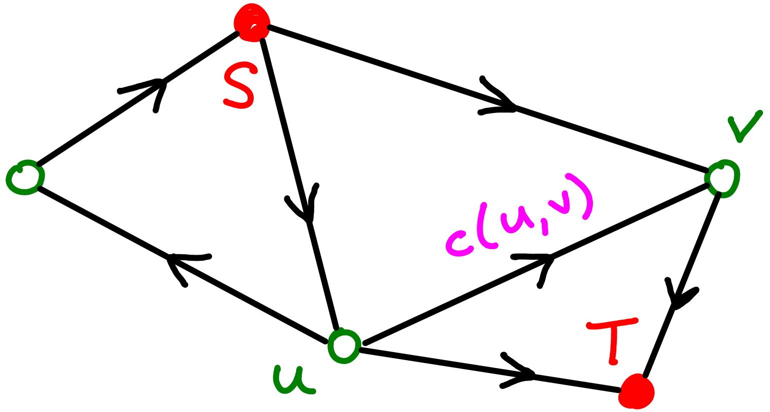
∘ Vertices : Σ input = Σ output → keep things flowing

$c(x,y)$ = capacity of edge $\overrightarrow{xy}$ $\rightarrow$ part of problem input



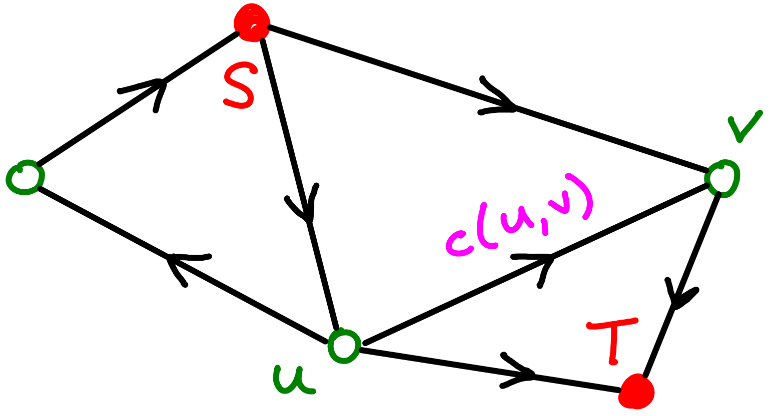
$c(x,y)$ = capacity of edge $\overrightarrow{xy}$ $\rightarrow$ part of problem input

$f(x,y)$ = actual flow in edge $\overrightarrow{xy}$ $\rightarrow$ TBD



$c(x,y)$ = capacity of edge $\vec{xy}$ $\rightarrow$ part of problem input

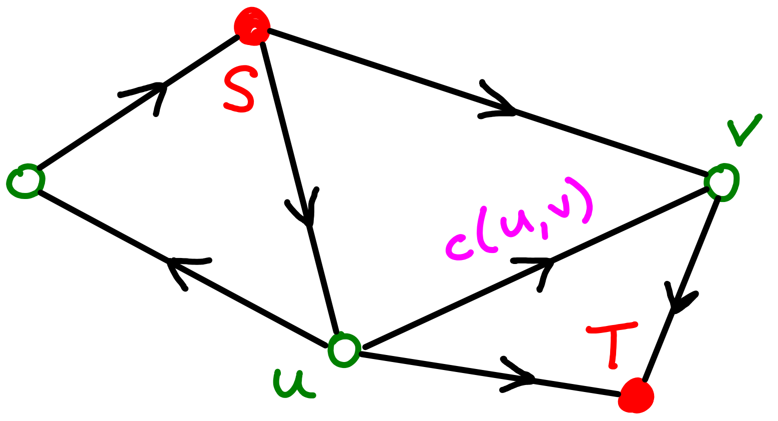
$f(x,y)$ = actual flow in edge $\vec{xy}$ $\rightarrow$ TBD



Capacity constraint: $0 \leq f(x,y) \leq c(x,y)$

$c(x,y)$ = capacity of edge $\overrightarrow{xy}$ → part of problem input

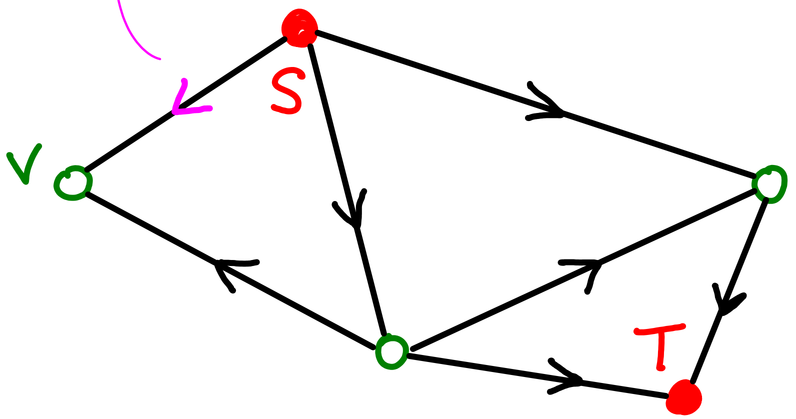
$f(x,y)$ = actual flow in edge $\overrightarrow{xy}$ → TBD

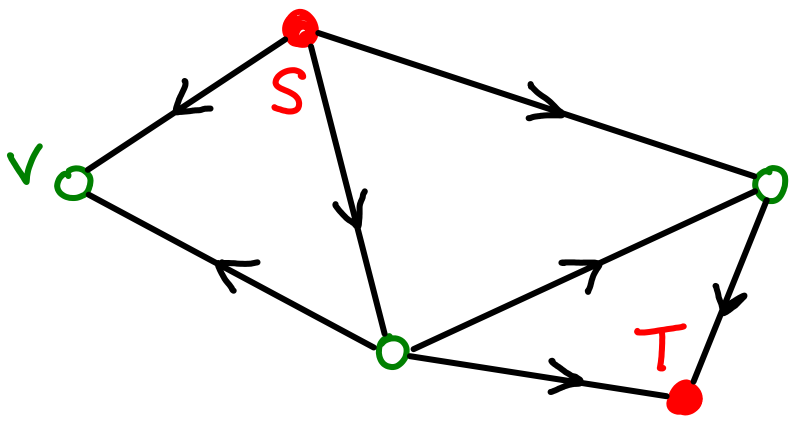


Capacity constraint: $0 \le f(x,y) \le c(x,y)$

Flow conservation: For all $u \neq S,T$: $\displaystyle\sum_{\text{all } v} f(v,u) = \sum_{\text{all } v} f(u,v)$

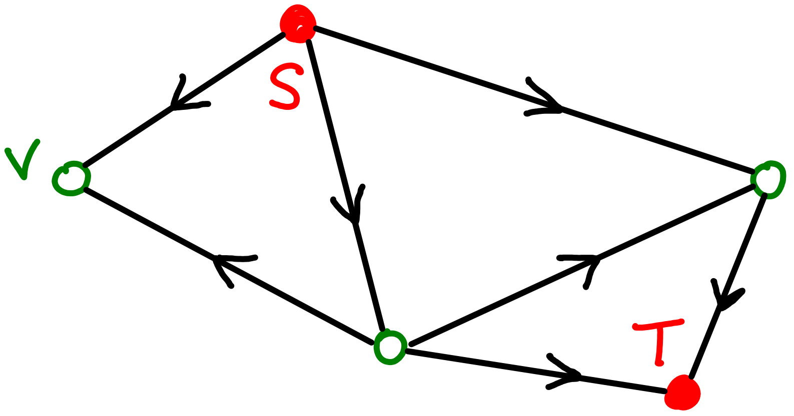
changed direction





v : should be removed

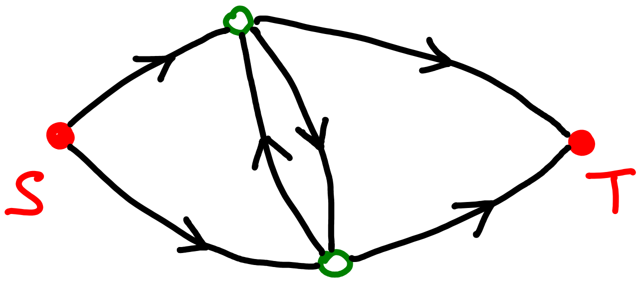
Every vertex should be on ≥ 1 directed path $S \to T$.

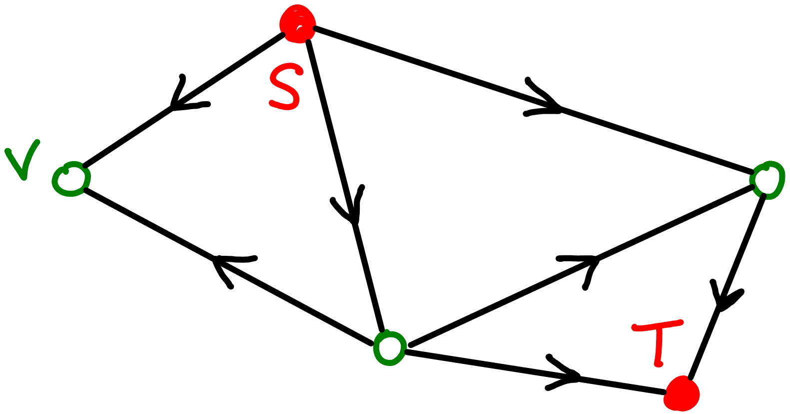


v : should be removed

Every vertex should be on ≥ 1 directed path $S \to T$.

Both directions allowed

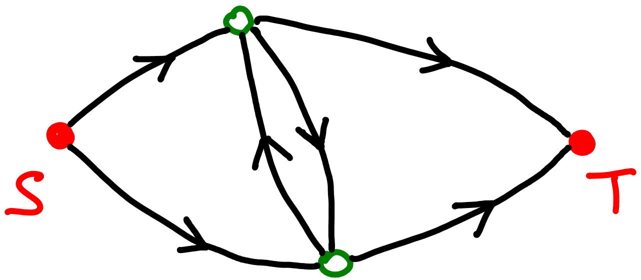


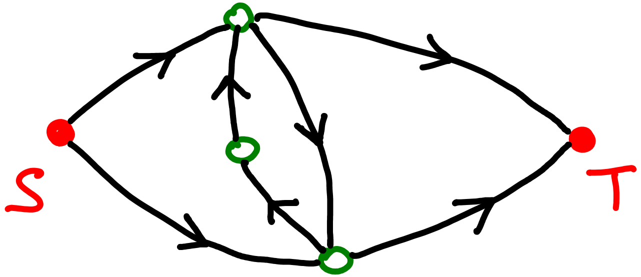


v: should be removed

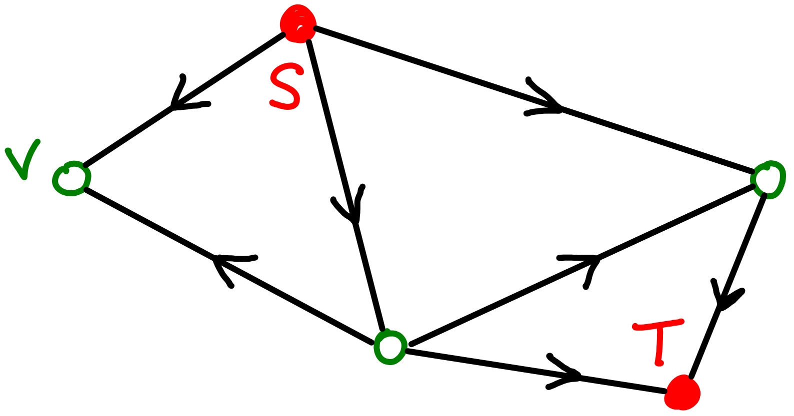
Every vertex should be on ≥ 1 directed path $S \to T$.

Both directions allowed, but we can simulate.



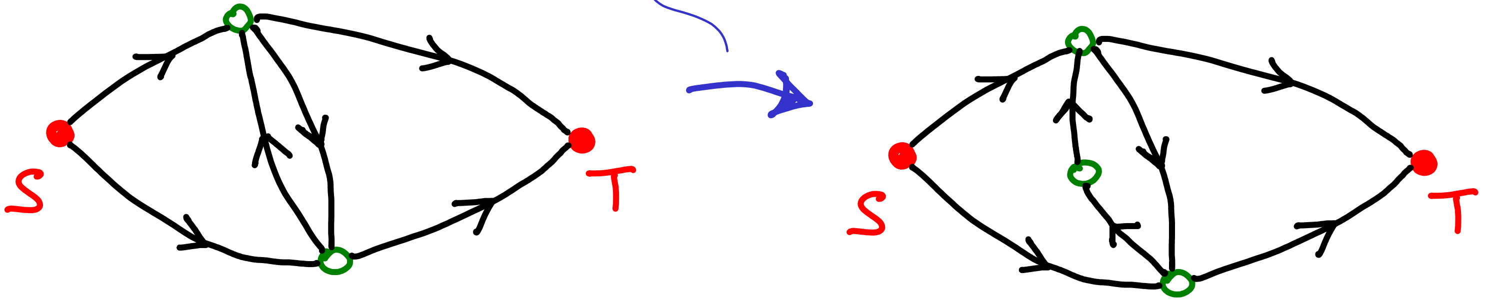


v : should be removed

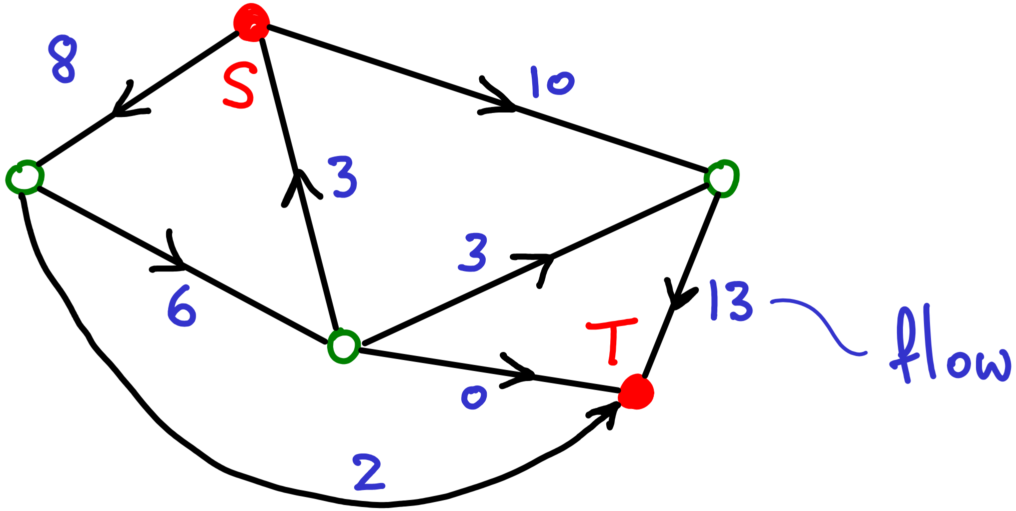


Every vertex should be on ≥ 1 directed path $S \to T$.

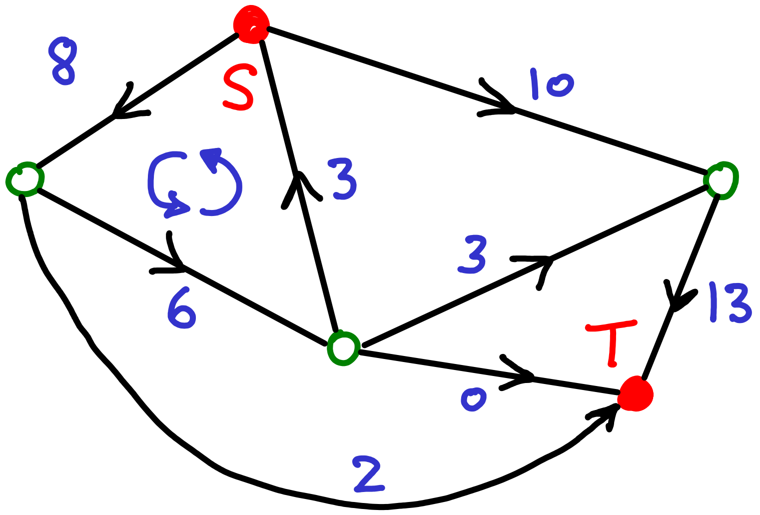
Both directions allowed, but we can simulate.



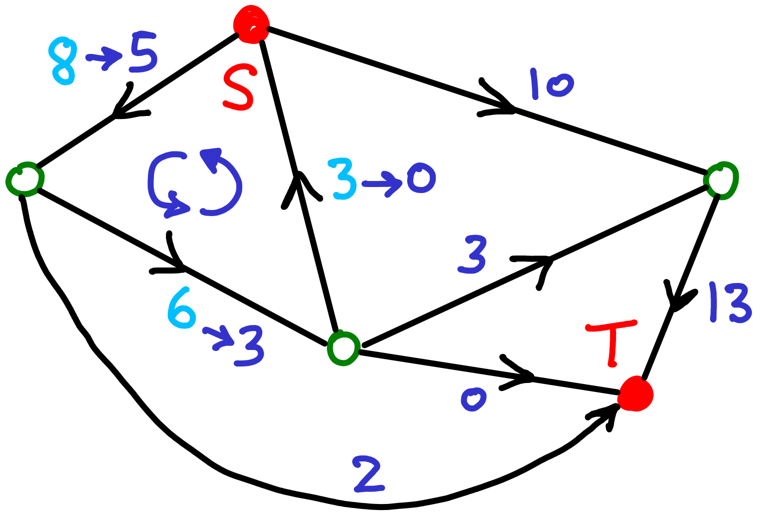
Note: w/ 2 directions, one will always have zero flow.



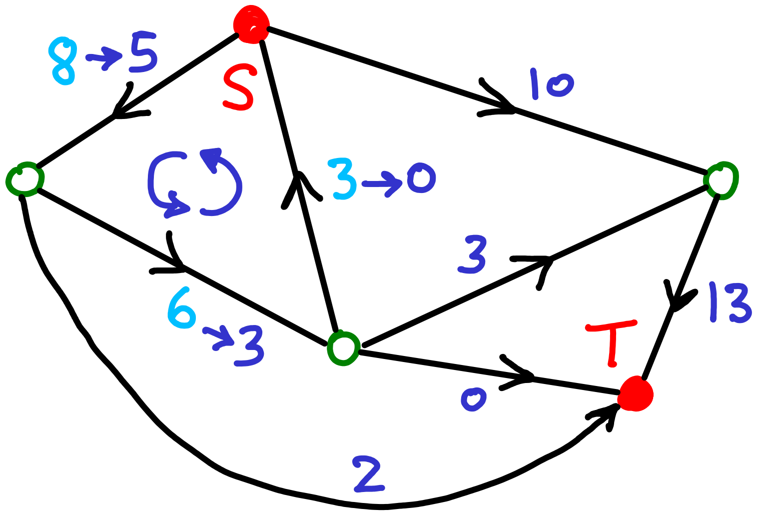
Can we simplify this?



cycle has flow on all edges



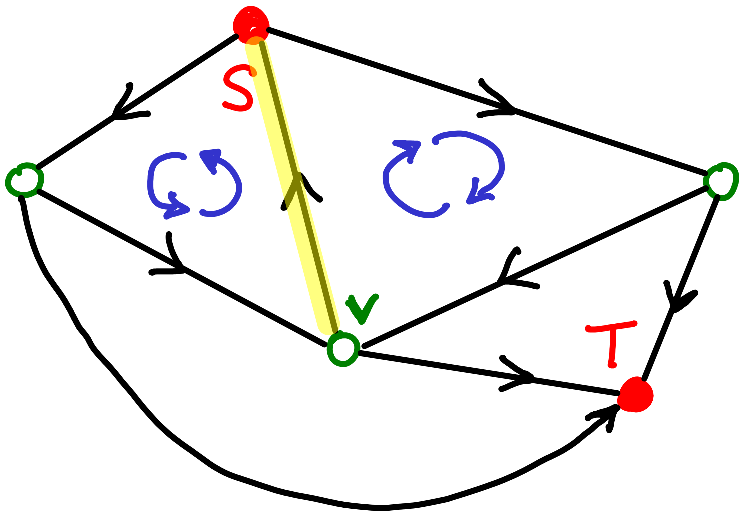
If any cycle has flow on all edges, we can get an equivalent solution with ≥1 edge having zero flow.

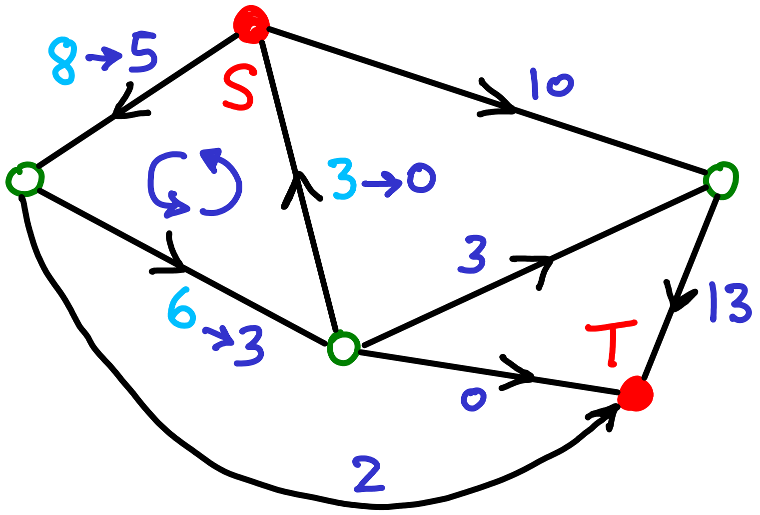


If any cycle has flow on all edges, we can get an equivalent solution with ≥1 edge having zero flow.

Notice any edge $\overrightarrow{vS}$ must be part of a cycle.

(otherwise we would have removed it)

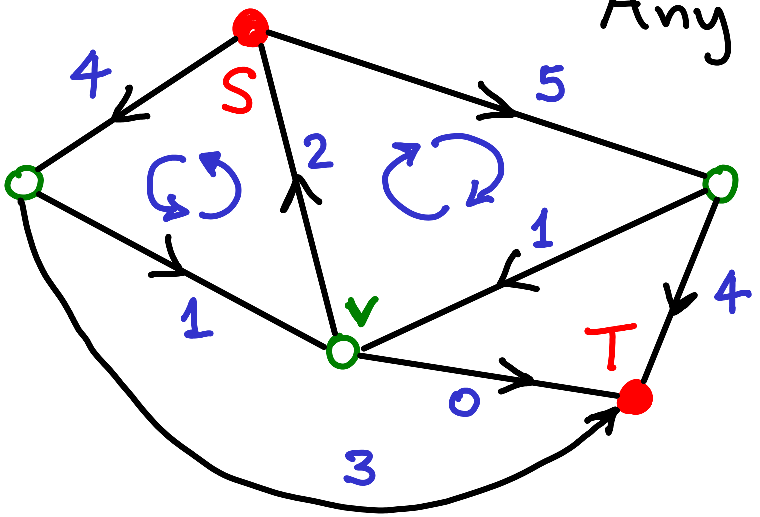


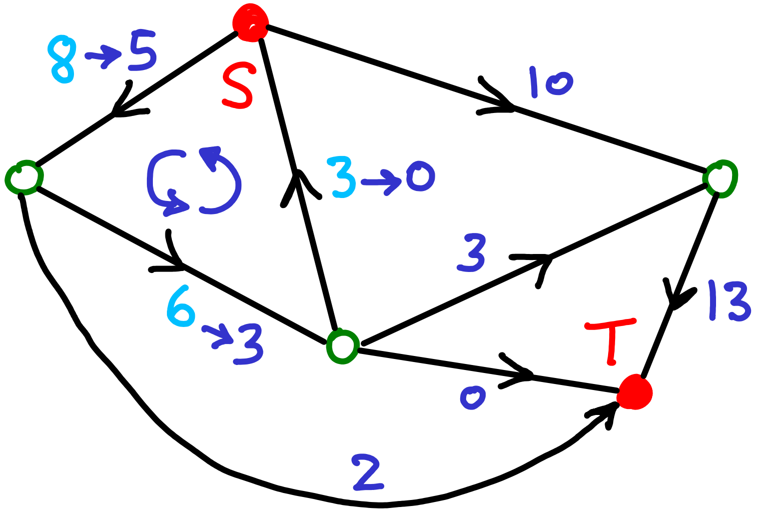


If any cycle has flow on all edges, we can get an equivalent solution with ≥1 edge having zero flow.

Notice any edge $\overrightarrow{vS}$ must be part of a cycle.

Any solution has an equivalent with $f(v,S)=0$

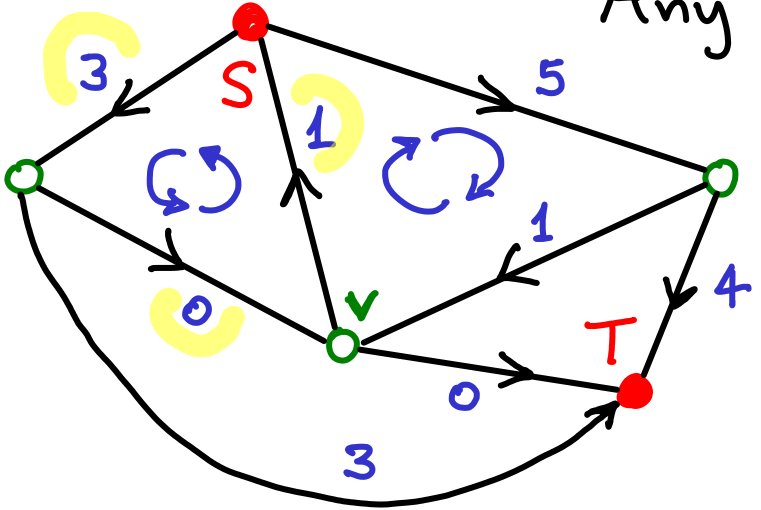


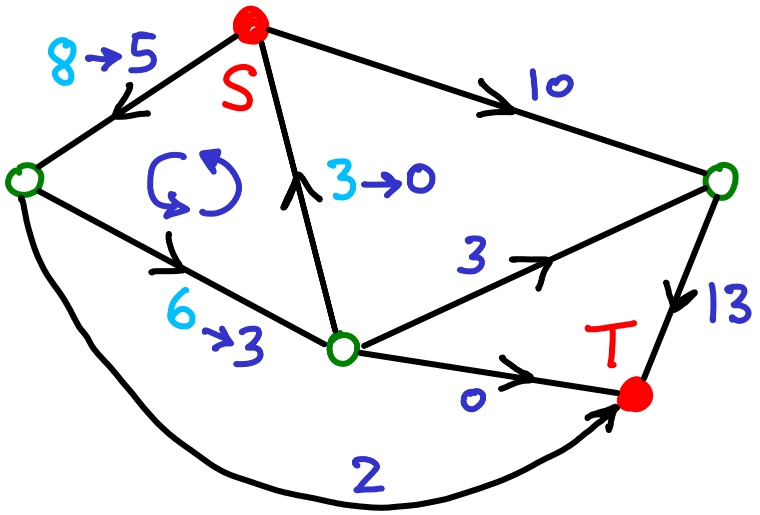


If any cycle has flow on all edges, we can get an equivalent solution with ≥ 1 edge having zero flow.

Notice any edge $\overrightarrow{vS}$ must be part of a cycle.

Any solution has an equivalent with $f(v,S) = 0$

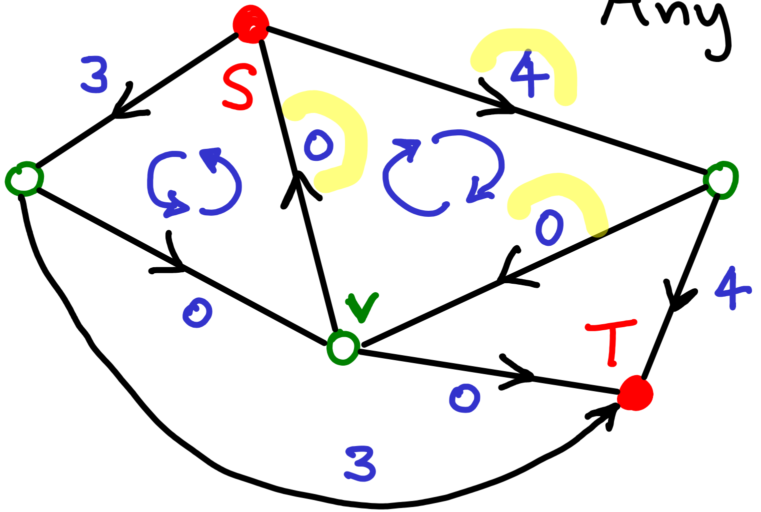


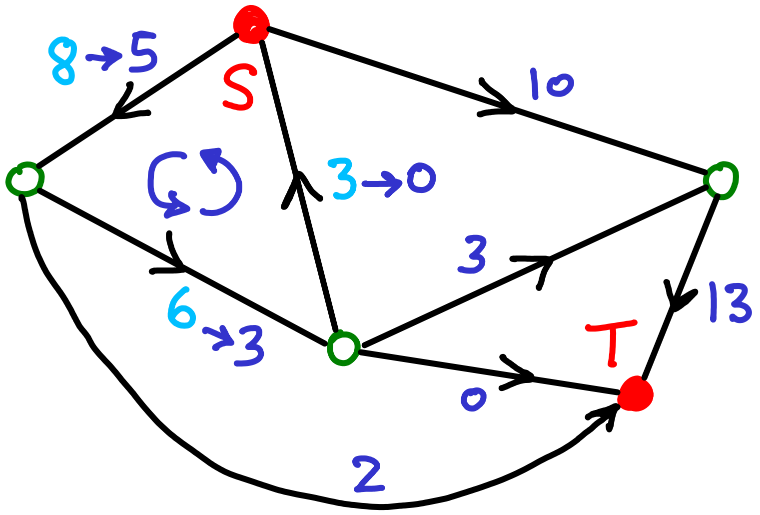


If any cycle has flow on all edges, we can get an equivalent solution with ≥ 1 edge having zero flow.

Notice any edge $\overrightarrow{vS}$ must be part of a cycle.

Any solution has an equivalent with $f(v, S) = 0$



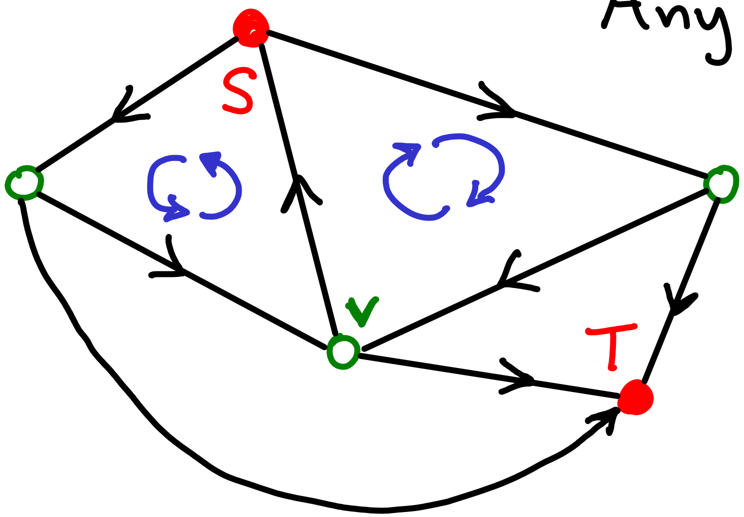


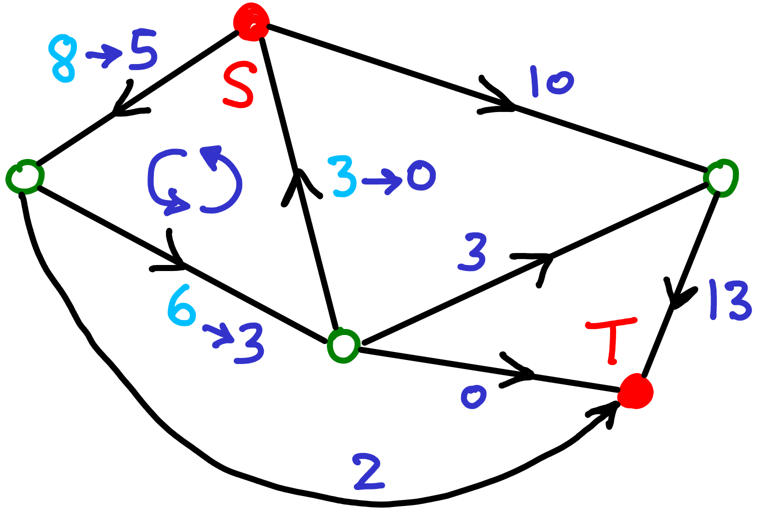
If any cycle has flow on all edges, we can get an equivalent solution with ≥1 edge having zero flow.

Notice any edge $\overrightarrow{vS}$ must be part of a cycle.

Any solution has an equivalent with $f(v,S)=0$

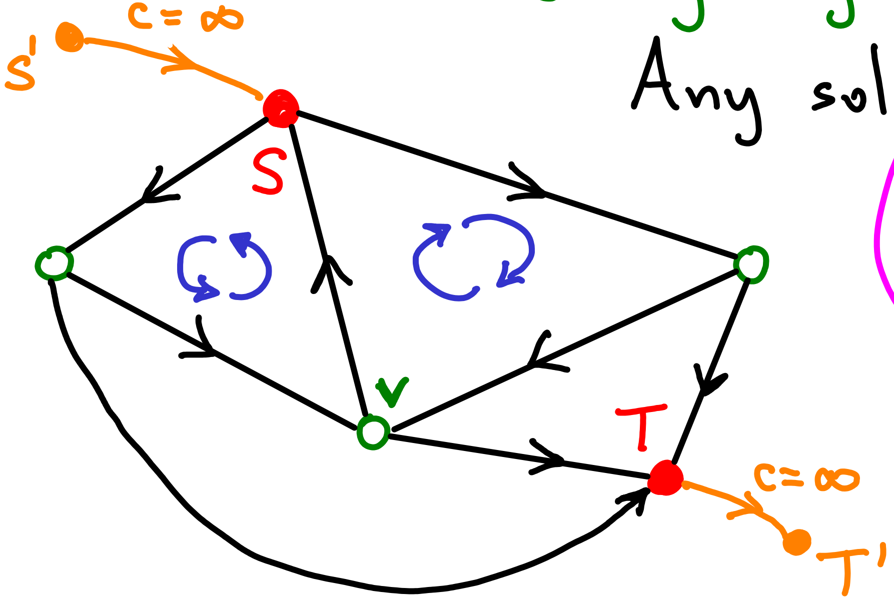
(intuitive... why send product back to limitless source?)





If any cycle has flow on all edges, we can get an equivalent solution with ≥ 1 edge having zero flow.

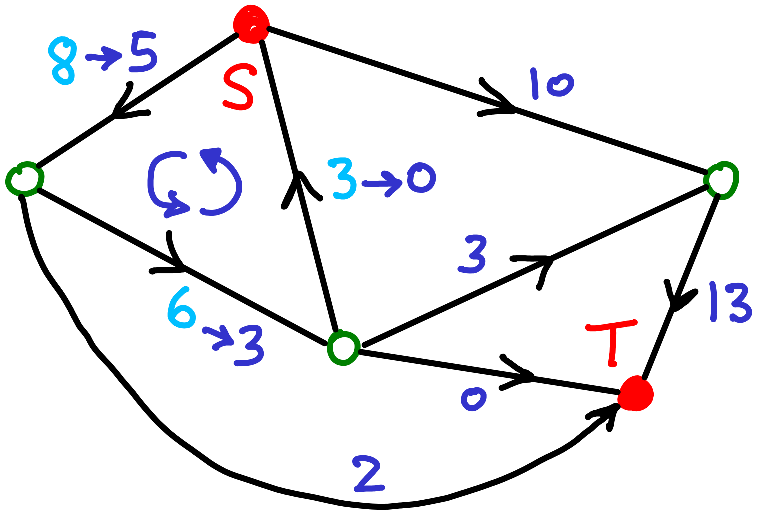
Notice any edge $\overrightarrow{vS}$ must be part of a cycle.



Any solution has an equivalent with $f(v,S) = 0$

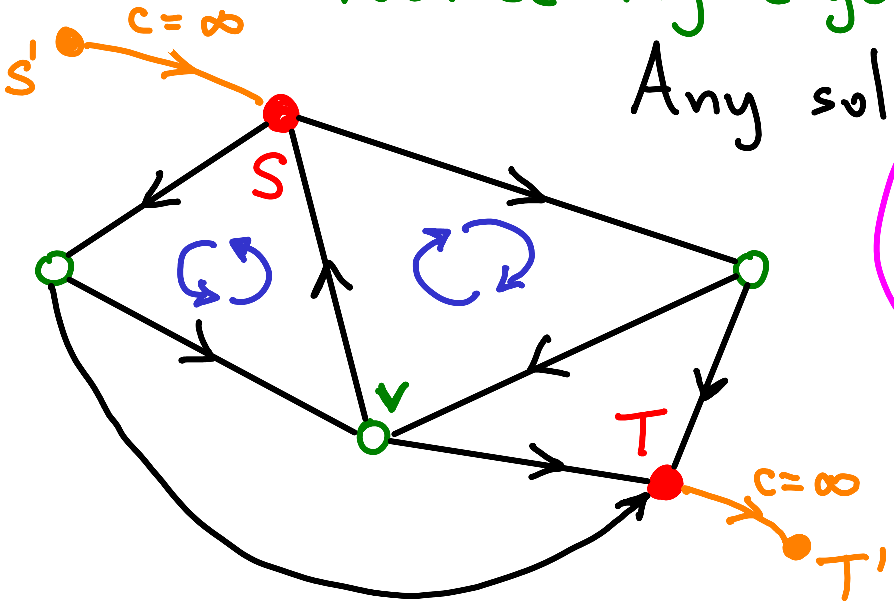
(intuitive... why send product back to limitless source?)

In any case assume $S \rightrightarrows \quad \rightrightarrows T$



If any cycle has flow on all edges, we can get an equivalent solution with ≥1 edge having zero flow.

Notice any edge $\vec{vS}$ must be part of a cycle.



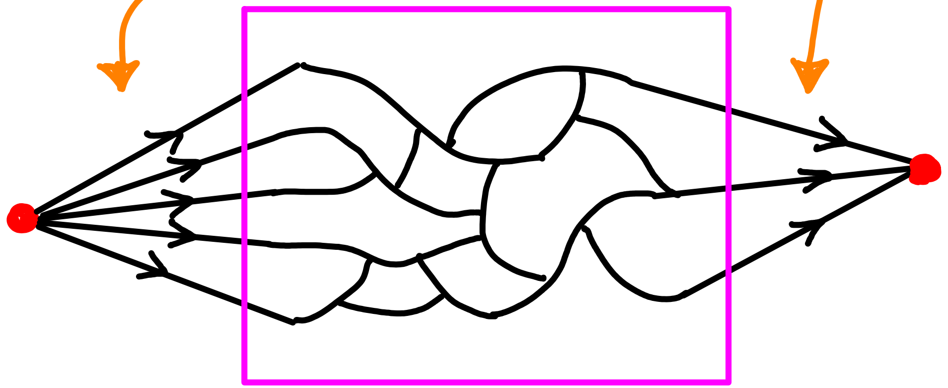
Any solution has an equivalent with $f(v,S)=0$

(intuitive... why send product back to limitless source?)

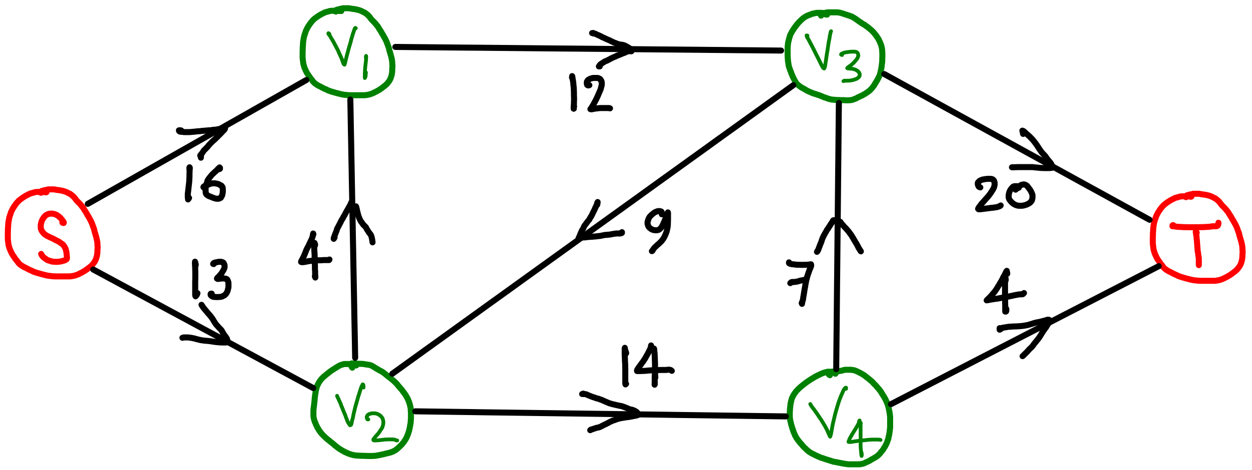
In any case assume $S \rightrightarrows \quad \rightrightarrows T$

(can also resolve multisource/target networks)

Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$

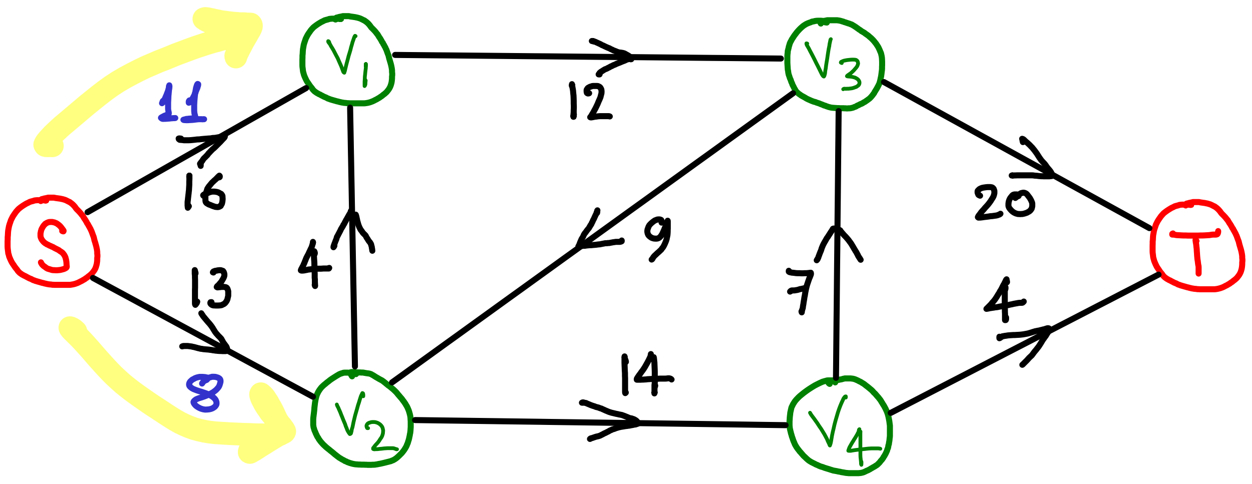


Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

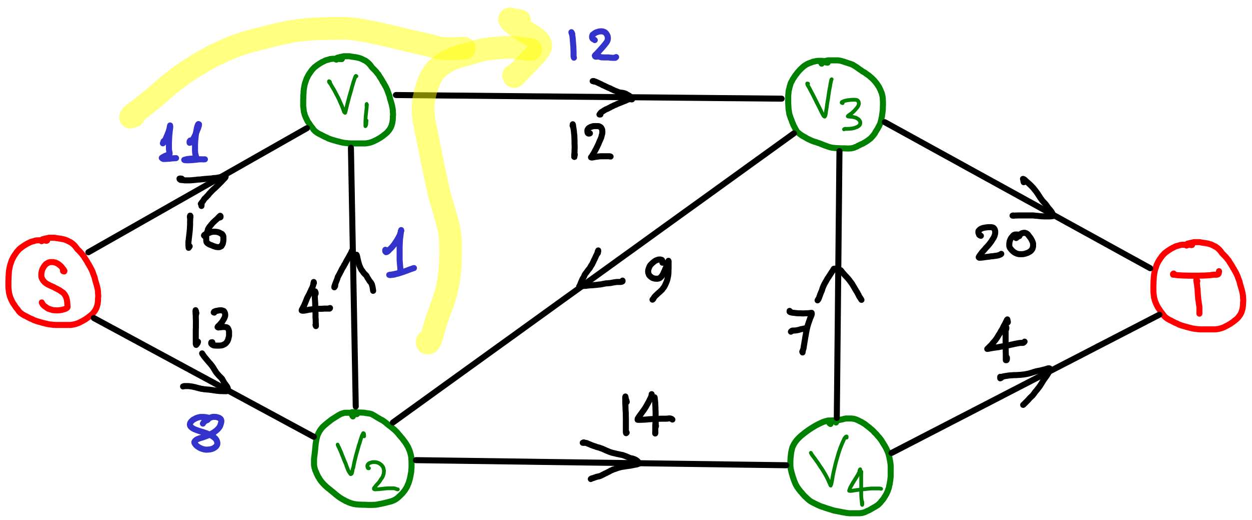
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

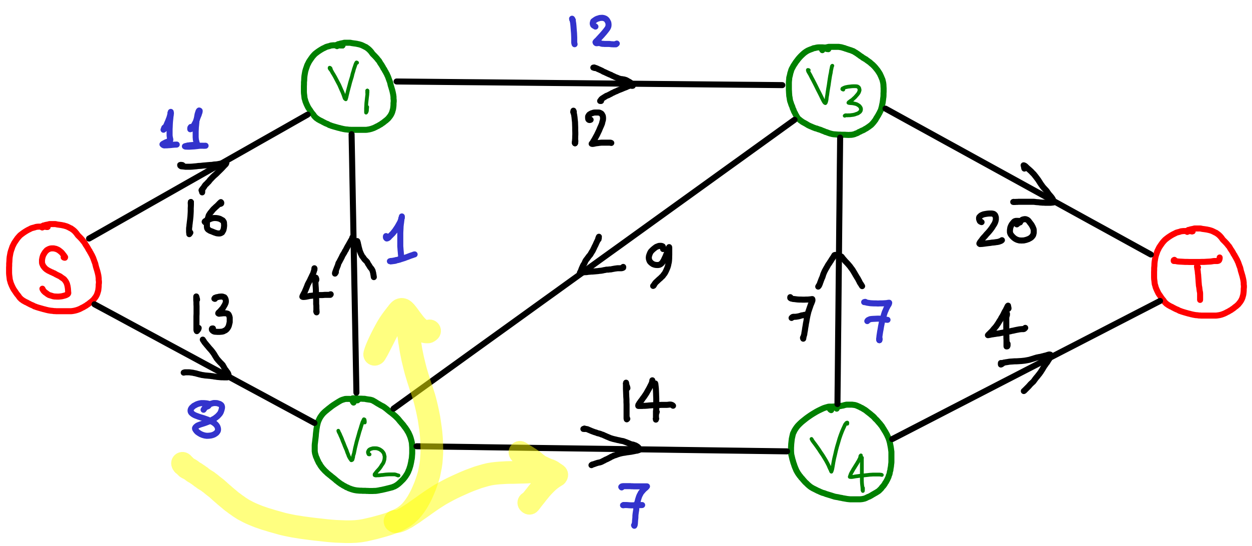
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

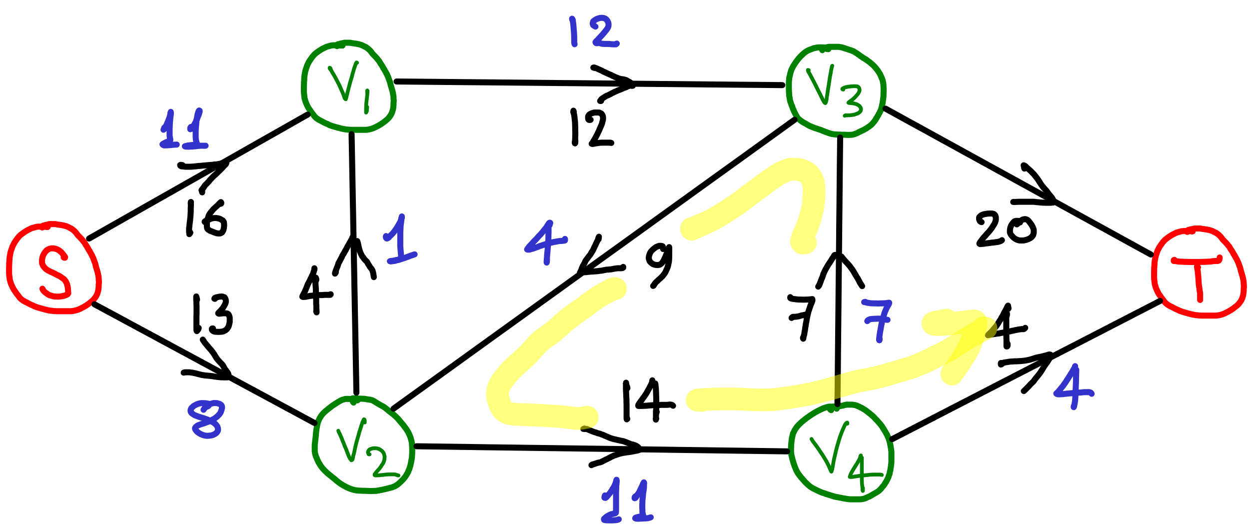
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

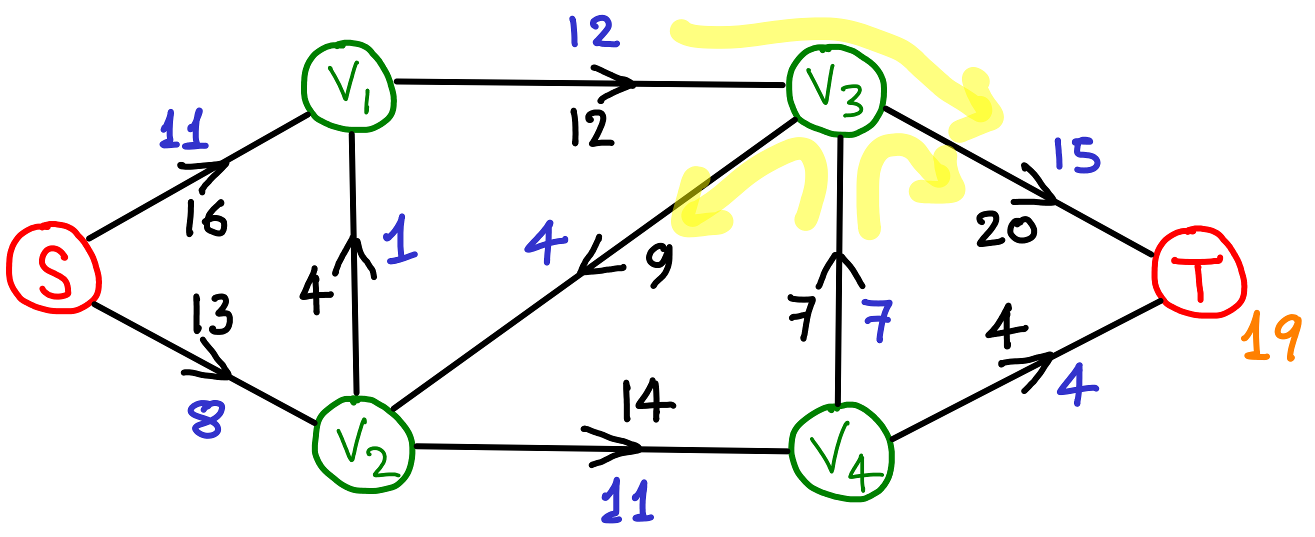
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

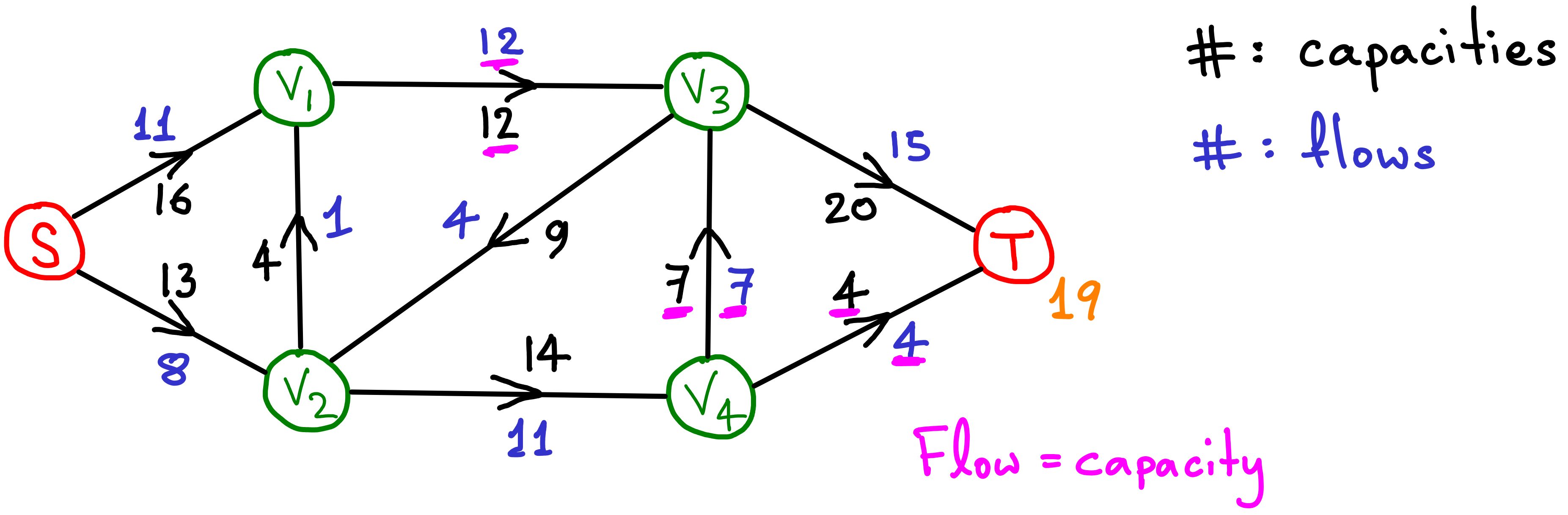
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



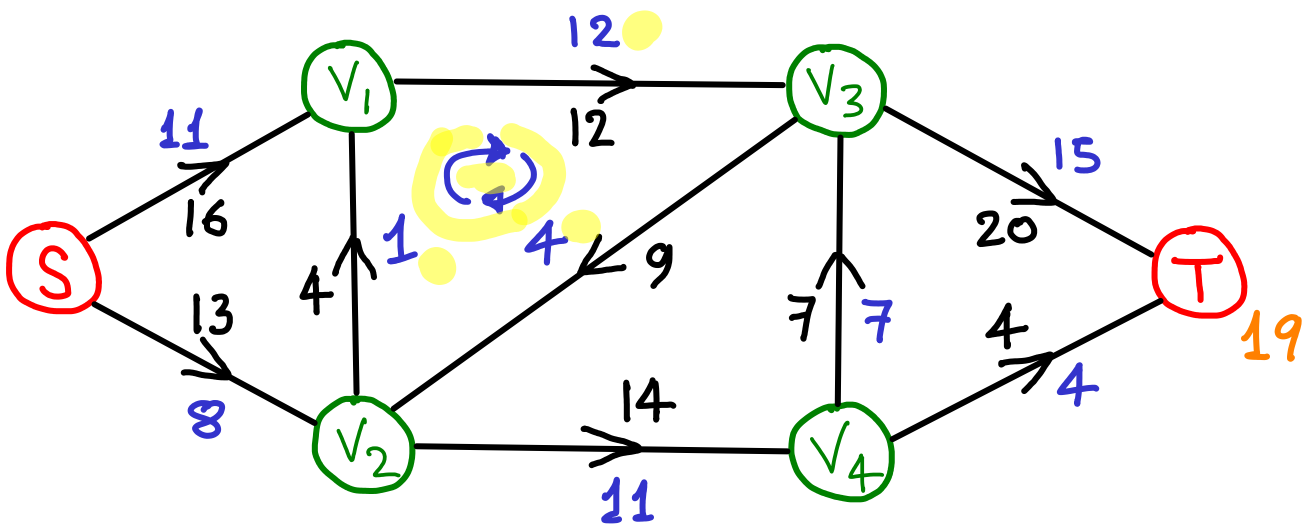
#: capacities

#: flows

Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



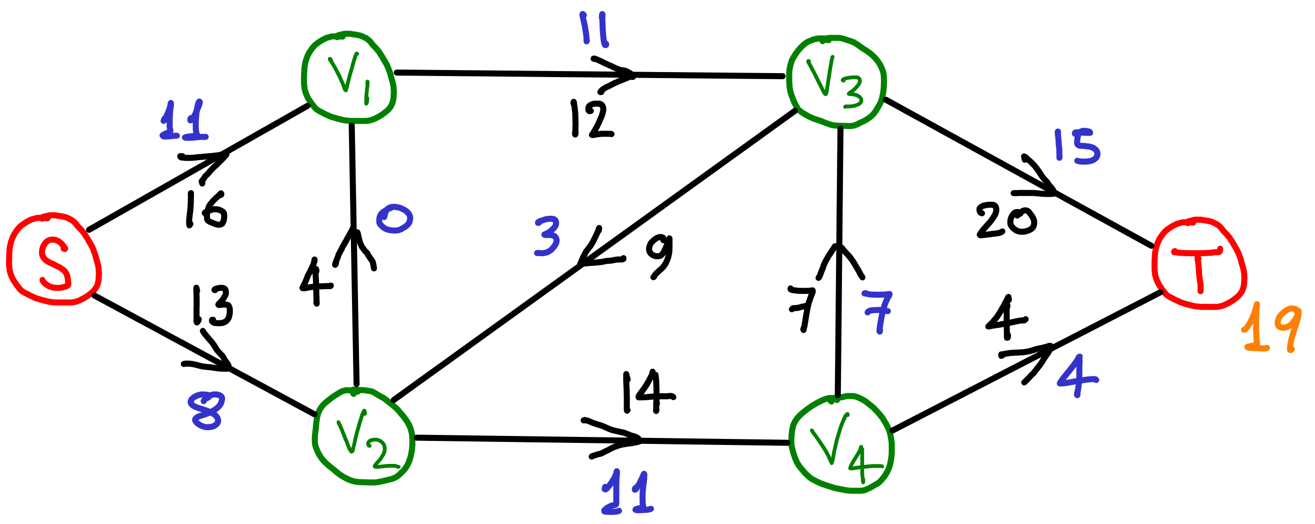
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

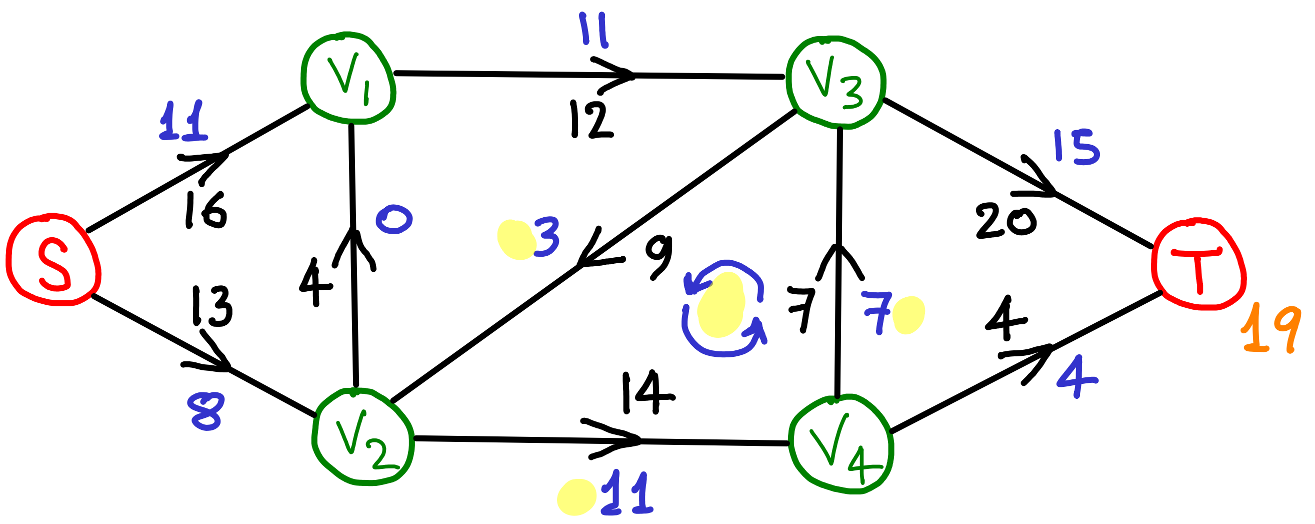
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

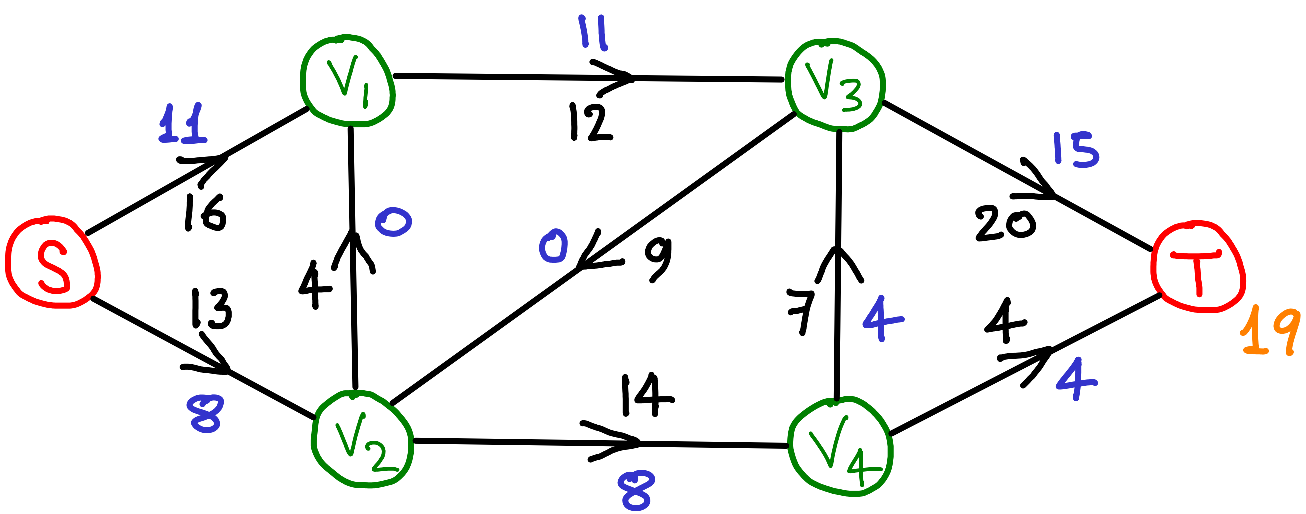
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

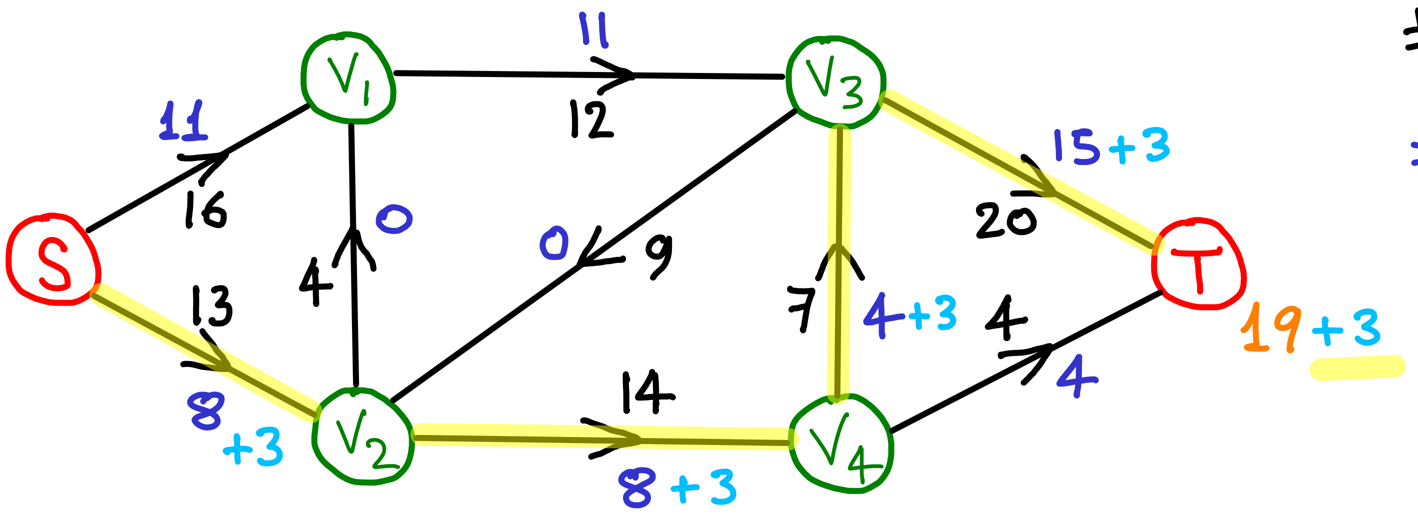
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

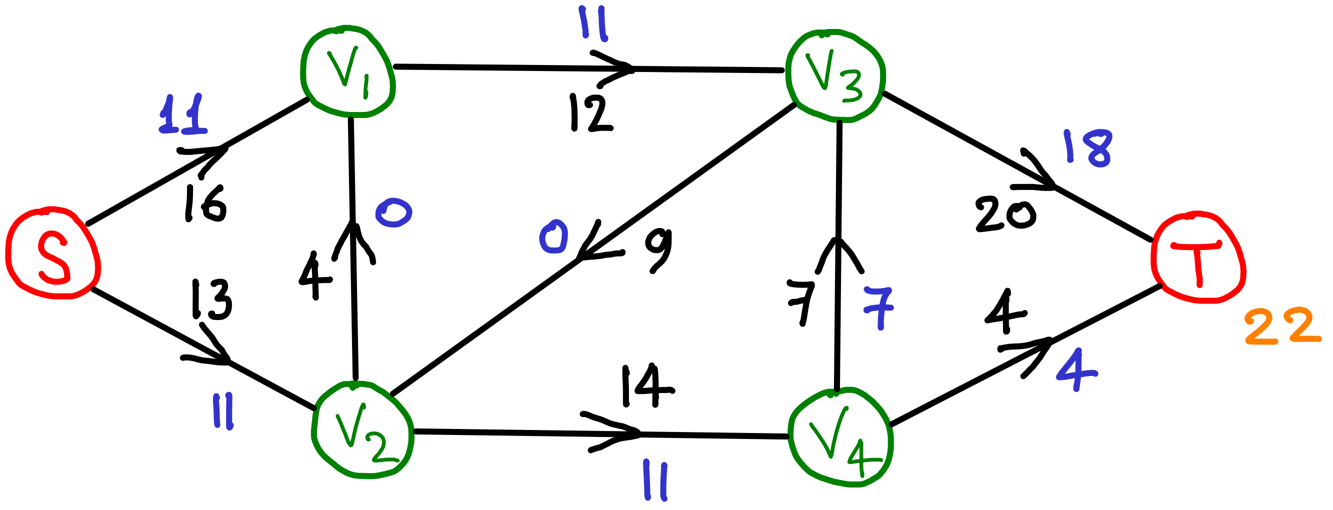
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

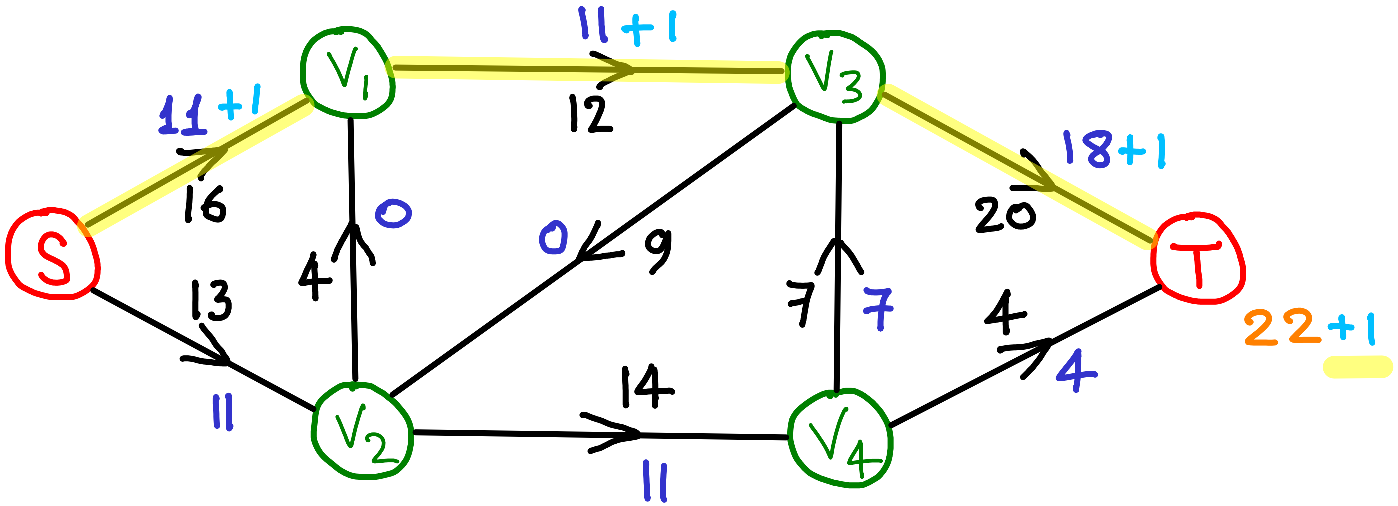
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

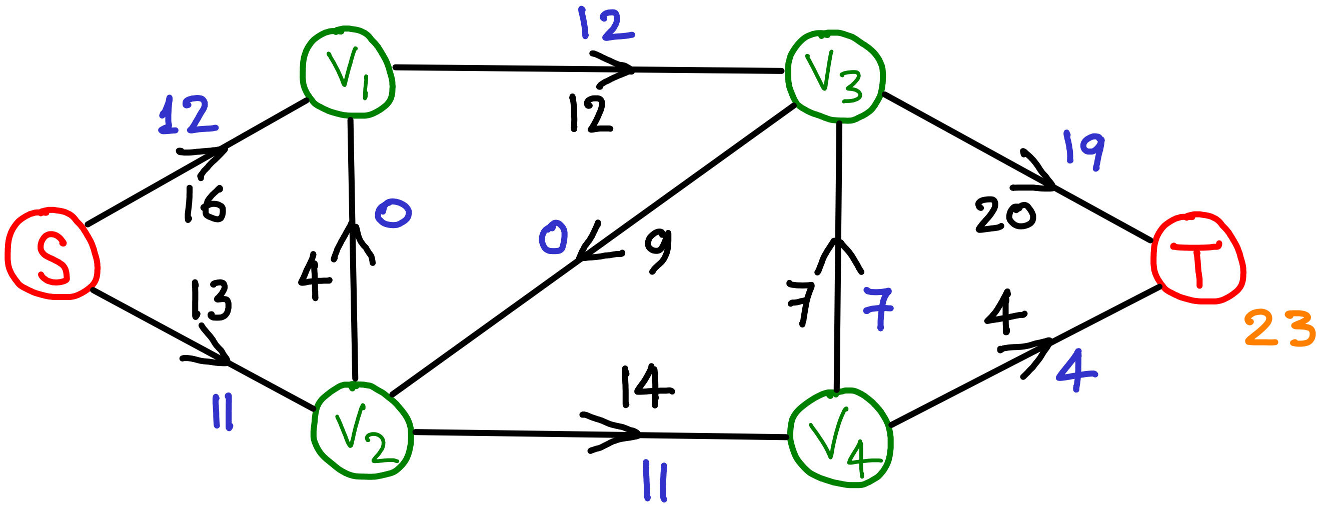
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

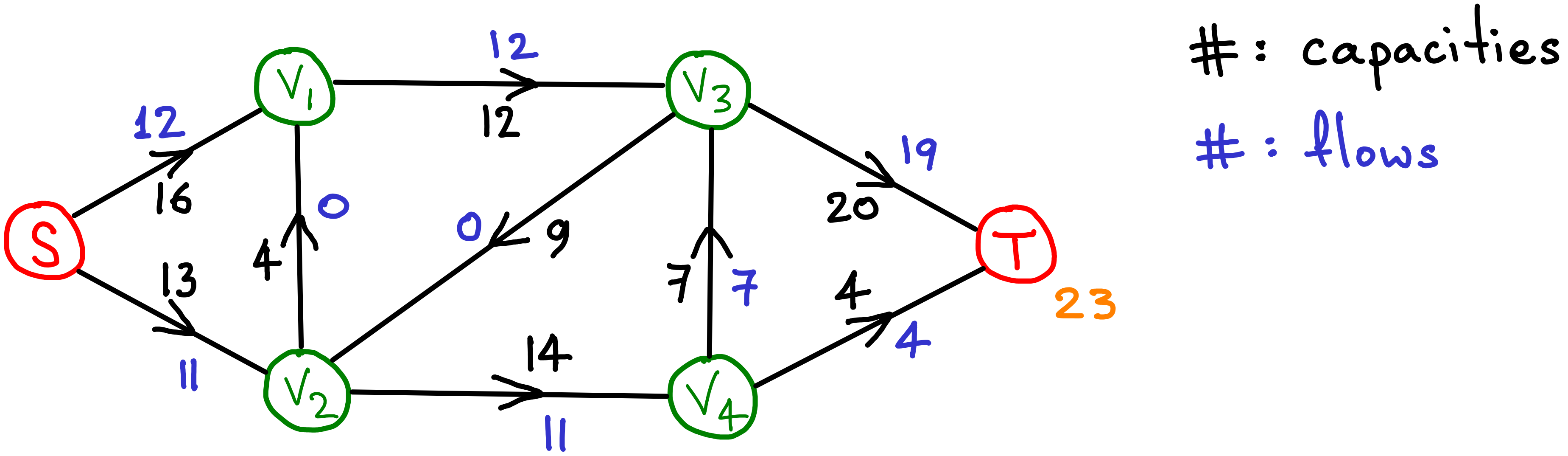
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$

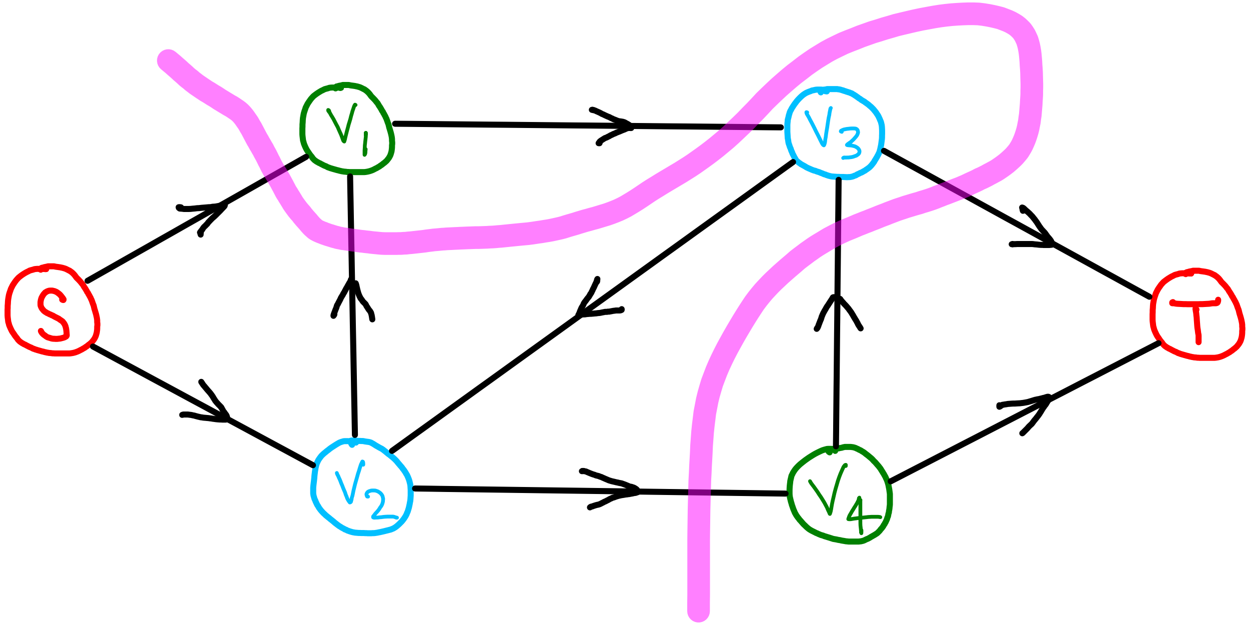


Ideas: - simplify cycles

- find positive paths from S to T

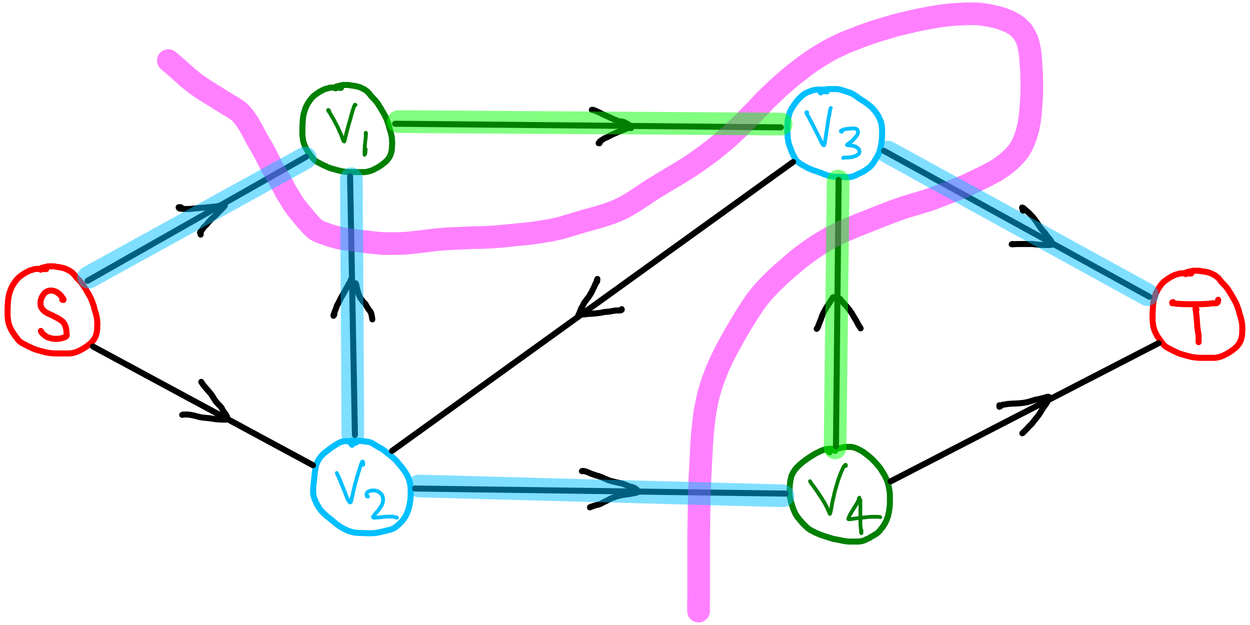
CUTS

- partition vertices into 2 groups.
- S & T separated.



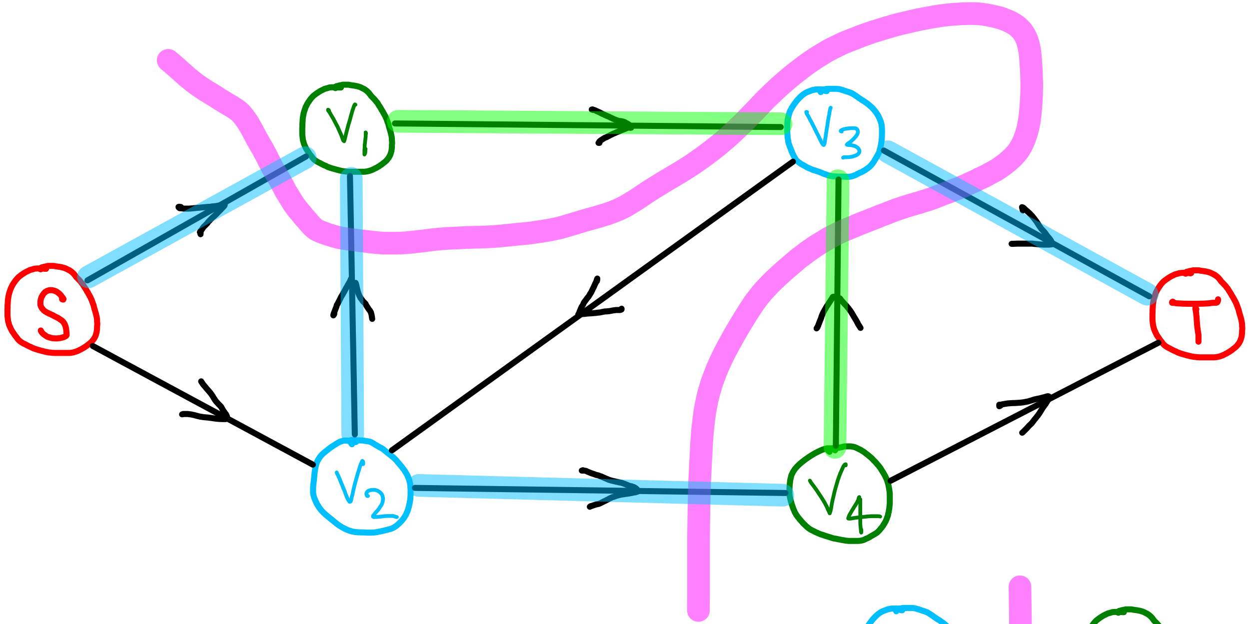
CUTS

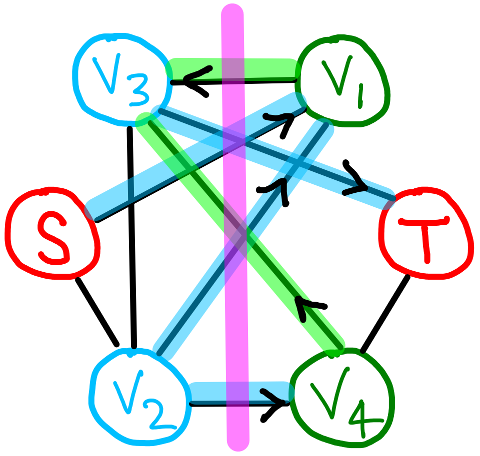
- partition vertices into 2 groups.
- S & T separated.
- edges: either not cut, or cut once → from S to T / from T to S



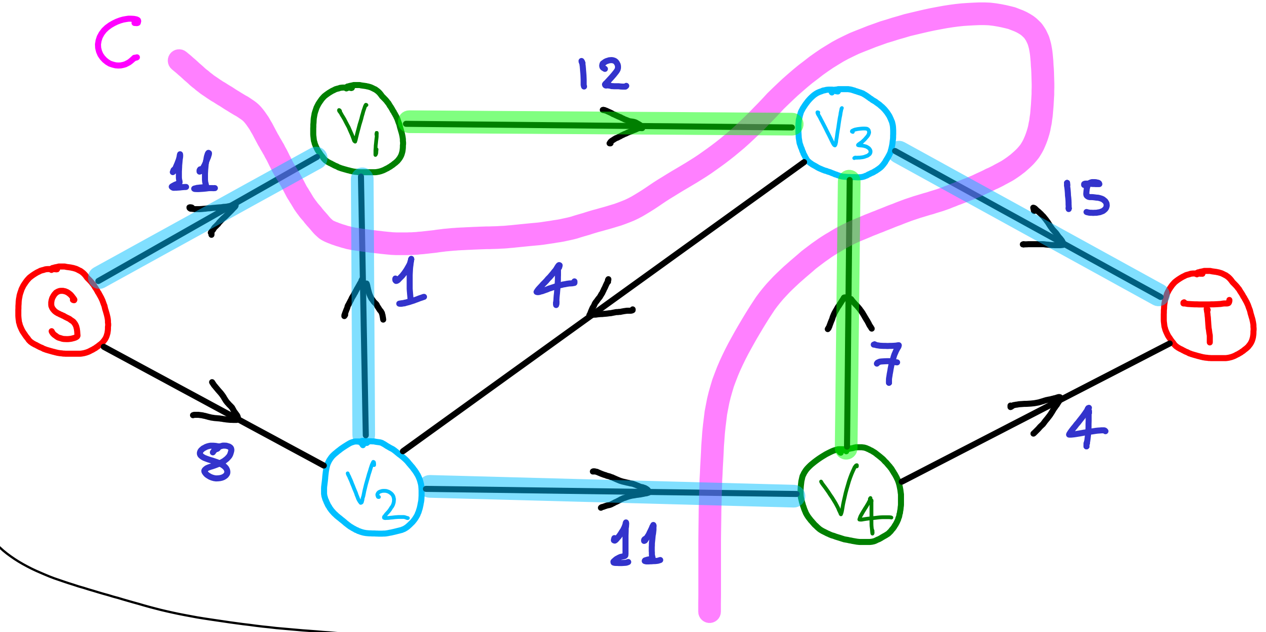
CUTS

- partition vertices into 2 groups.
- S & T separated.
- edges: either not cut, or cut once → { from S to T / from T to S



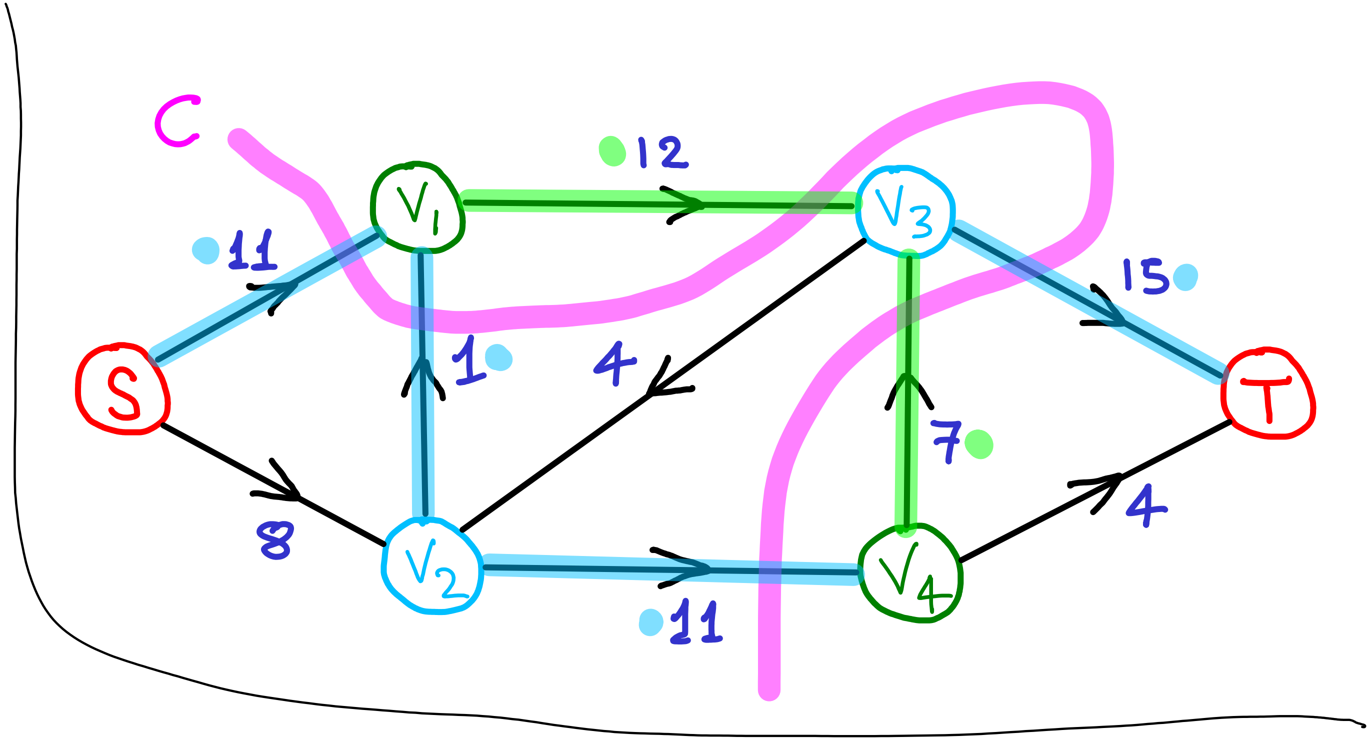


Flow(C) = $\sum$ flow − $\sum$ flow



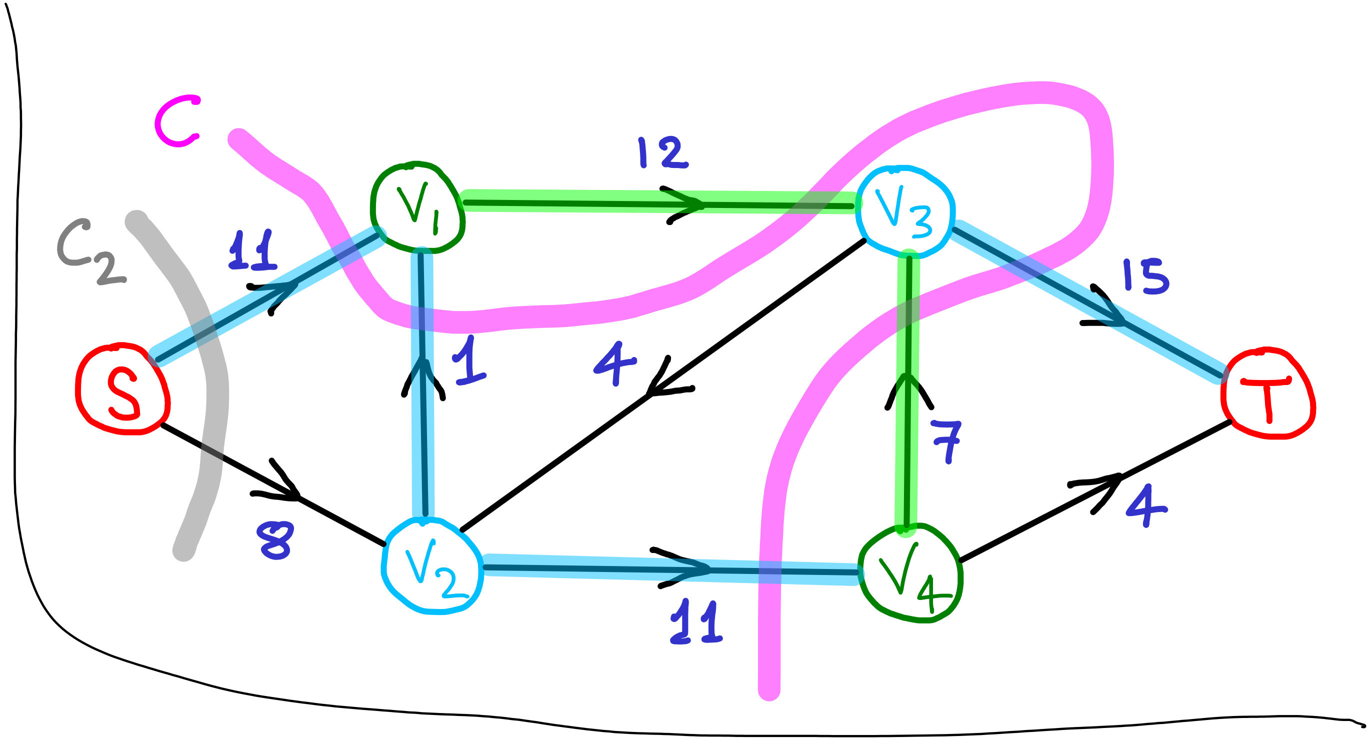
$$\mathrm{Flow}(C) = \sum \mathrm{flow} - \sum \mathrm{flow}$$

$$= 11 + 1 + 11 + 15 - 12 - 7$$



$$\mathrm{Flow}(C) = \sum \mathrm{flow} - \sum \mathrm{flow}$$

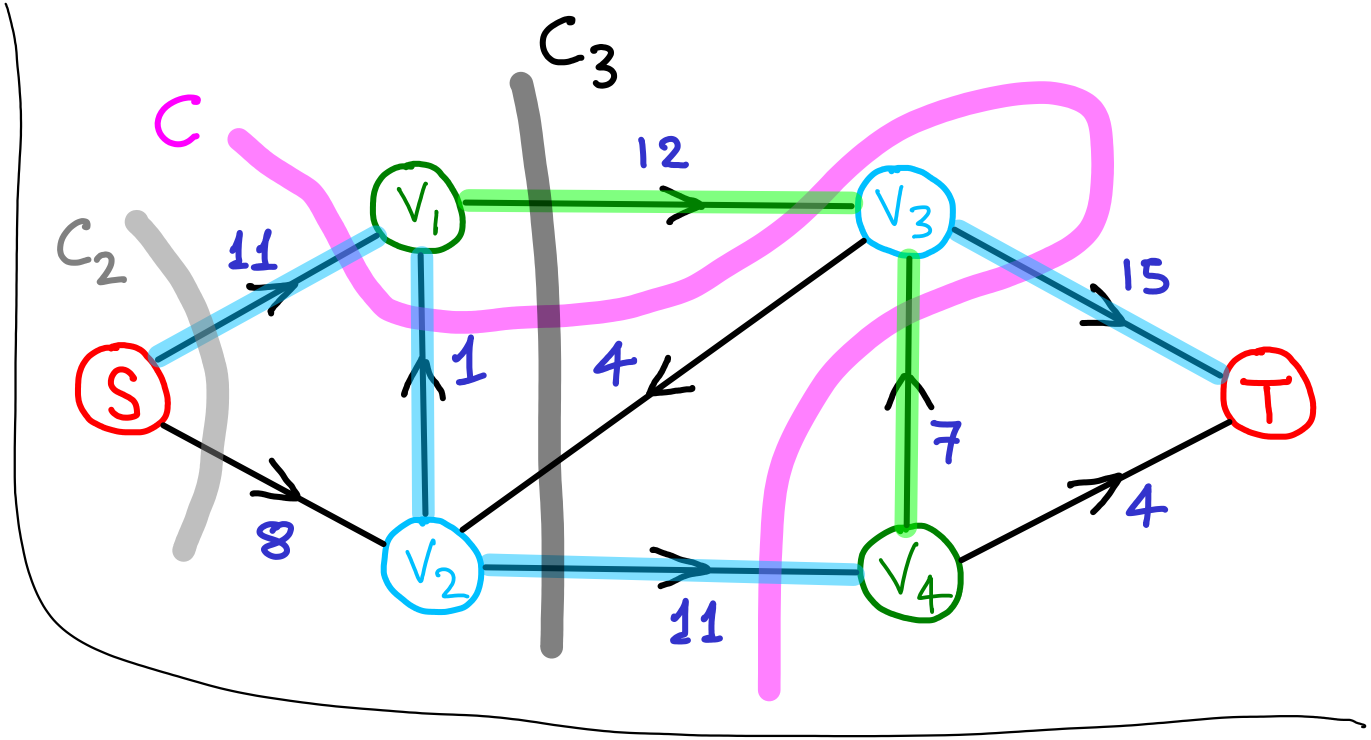
$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$



$$\mathrm{Flow}(C) = \underbrace{\mathrm{Flow}(C_2)}_{11+8}$$

$$\mathrm{Flow}(C) = \sum \mathrm{flow} - \sum \mathrm{flow}$$

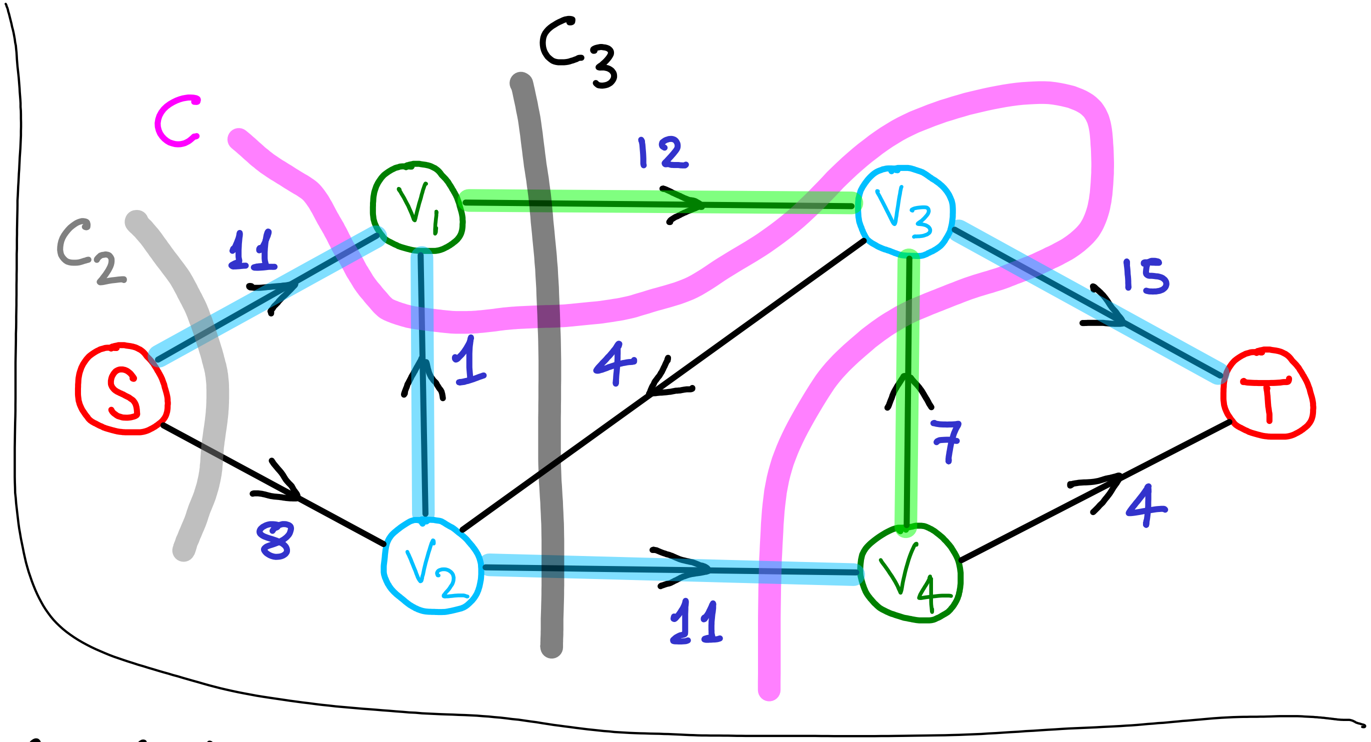
$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$



$$\mathrm{Flow}(C) = \mathrm{Flow}(C_2)$$

$$\mathrm{Flow}(C) = \sum \mathrm{flow} - \sum \mathrm{flow}$$

$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$

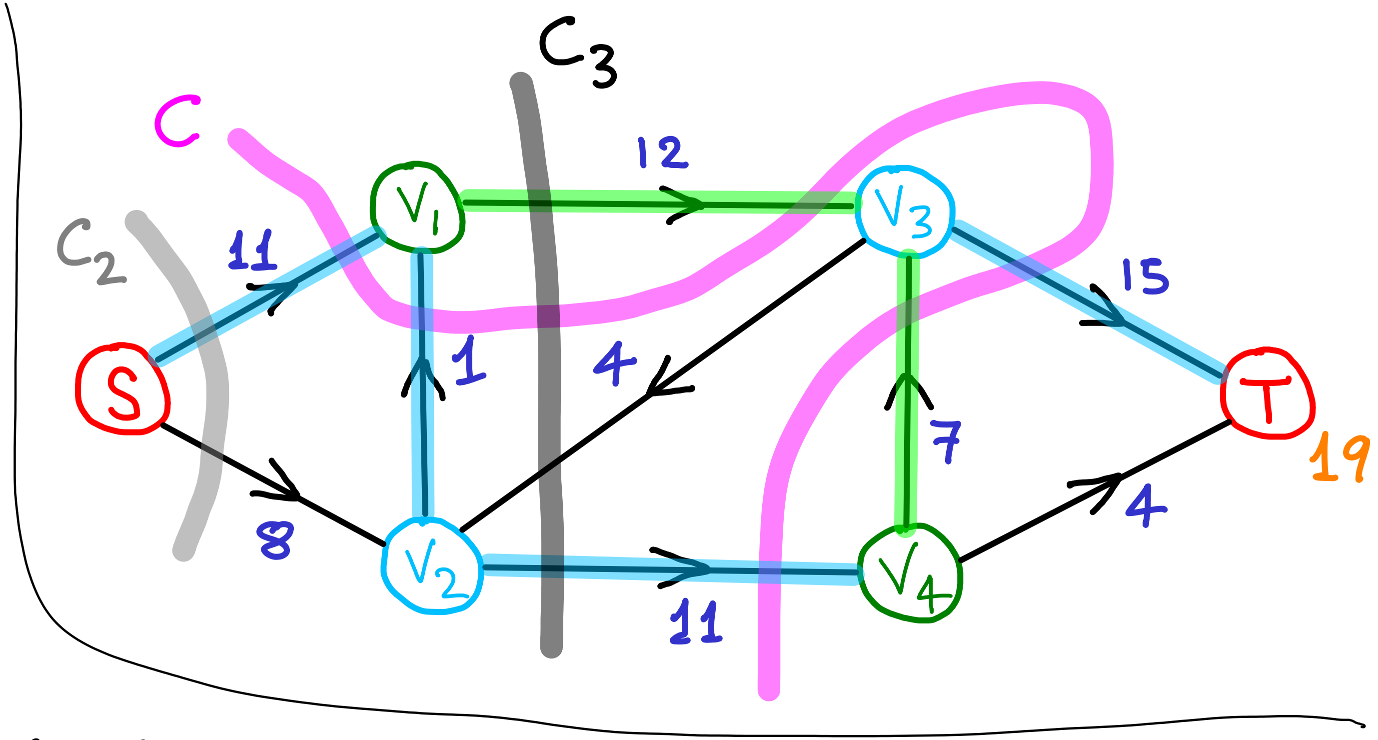


$$\mathrm{Flow}(C) = \mathrm{Flow}(C_2) = \mathrm{Flow}(C_3)$$

$$11 + 12 - 4$$

$$\mathrm{Flow}(C) = \sum \mathrm{flow} - \sum \mathrm{flow}$$

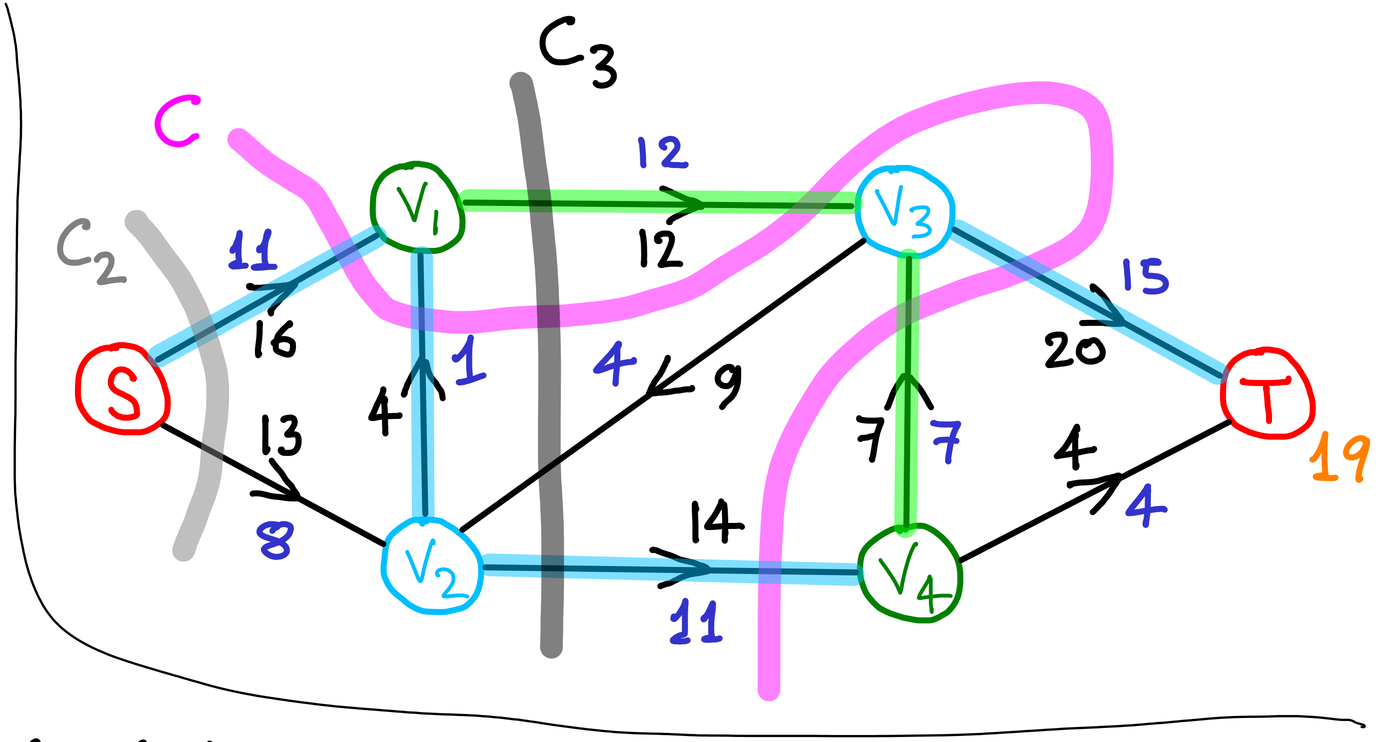
$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$



$$\mathrm{Flow}(C) = \mathrm{Flow}(C_2) = \mathrm{Flow}(C_3) = \mathrm{Flow}(\text{any cut}) = \text{FLOW VALUE}$$

$$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$$

$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$



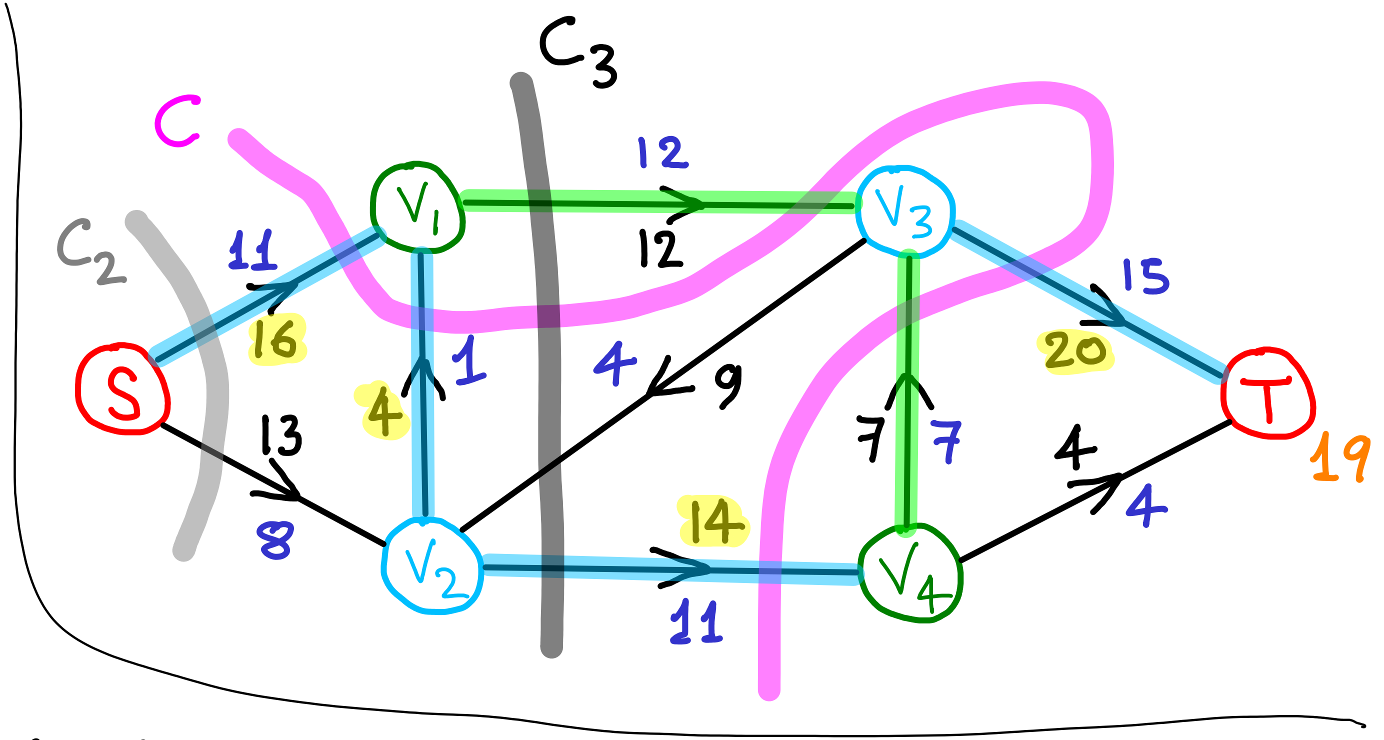
$$\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) = \text{FLOW VALUE}$$

$$\mathrm{Flow}(C) = \sum \mathrm{flow} - \sum \mathrm{flow}$$

$$= 11+1+11+15-12-7 = 19$$

$$\mathrm{Capacity}(C) = \sum \mathrm{capacity}$$

$$= \underline{16+4+14+20} = 54$$



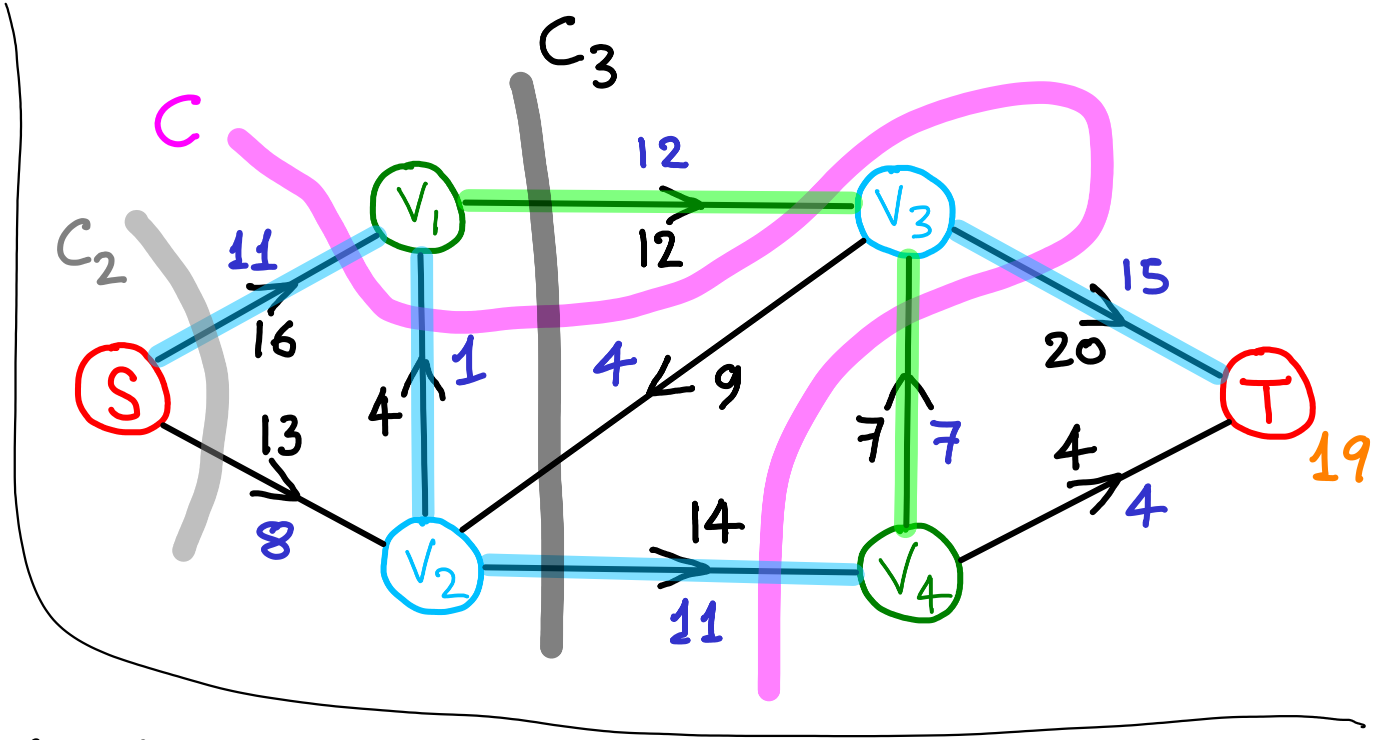
$$\mathrm{Flow}(C) = \mathrm{Flow}(C_2) = \mathrm{Flow}(C_3) = \mathrm{Flow}(\text{any cut}) = \text{FLOW VALUE}$$

$$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$$

$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$

$$\text{Capacity}(C) = \sum \text{capacity}$$

$$= 16 + 4 + 14 + 20 = 54$$



$$\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) = \text{FLOW VALUE}$$

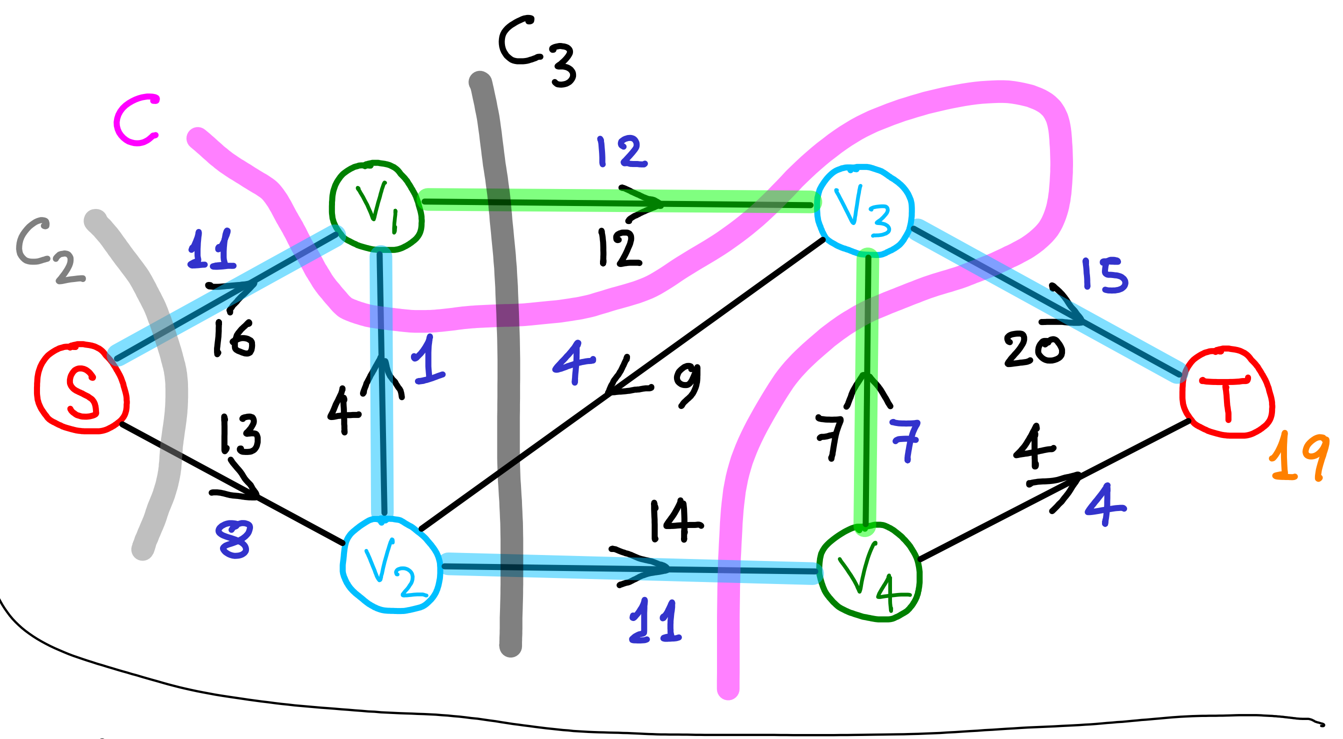
$$\text{Flow}(C) \leq \text{Capacity}(C)$$

$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$

$= 11 + 1 + 11 + 15 - 12 - 7 = 19$

$\text{Capacity}(C) = \sum \text{capacity}$

$= 16 + 4 + 14 + 20 = 54$



(i) $\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) =$ FLOW VALUE

(ii) $\text{Flow}(C) \leq \text{Capacity}(C)$

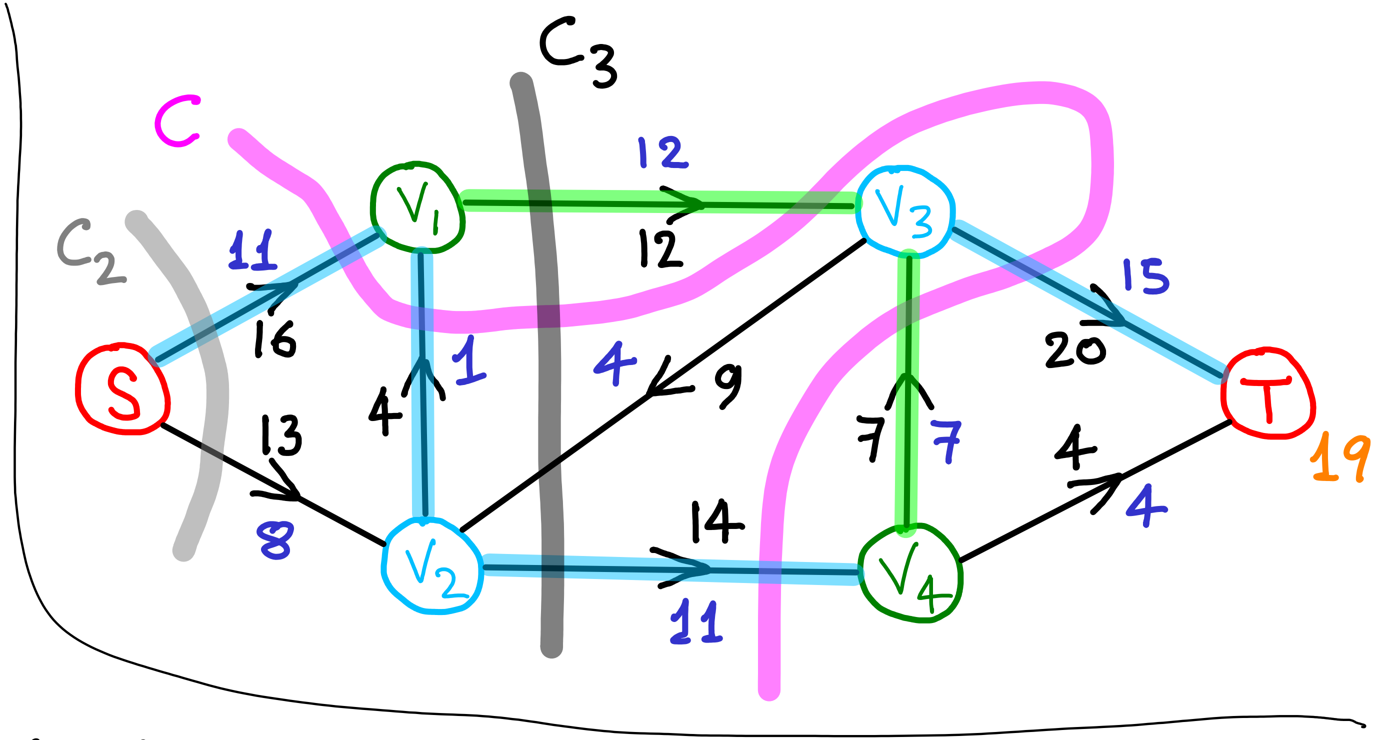
?

$$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$$

$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$

$$\text{Capacity}(C) = \sum \text{capacity}$$

$$= 16 + 4 + 14 + 20 = 54$$



(i) $\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) =$ FLOW VALUE

(ii) $\text{Flow}(C) \leq \text{Capacity}(C)$

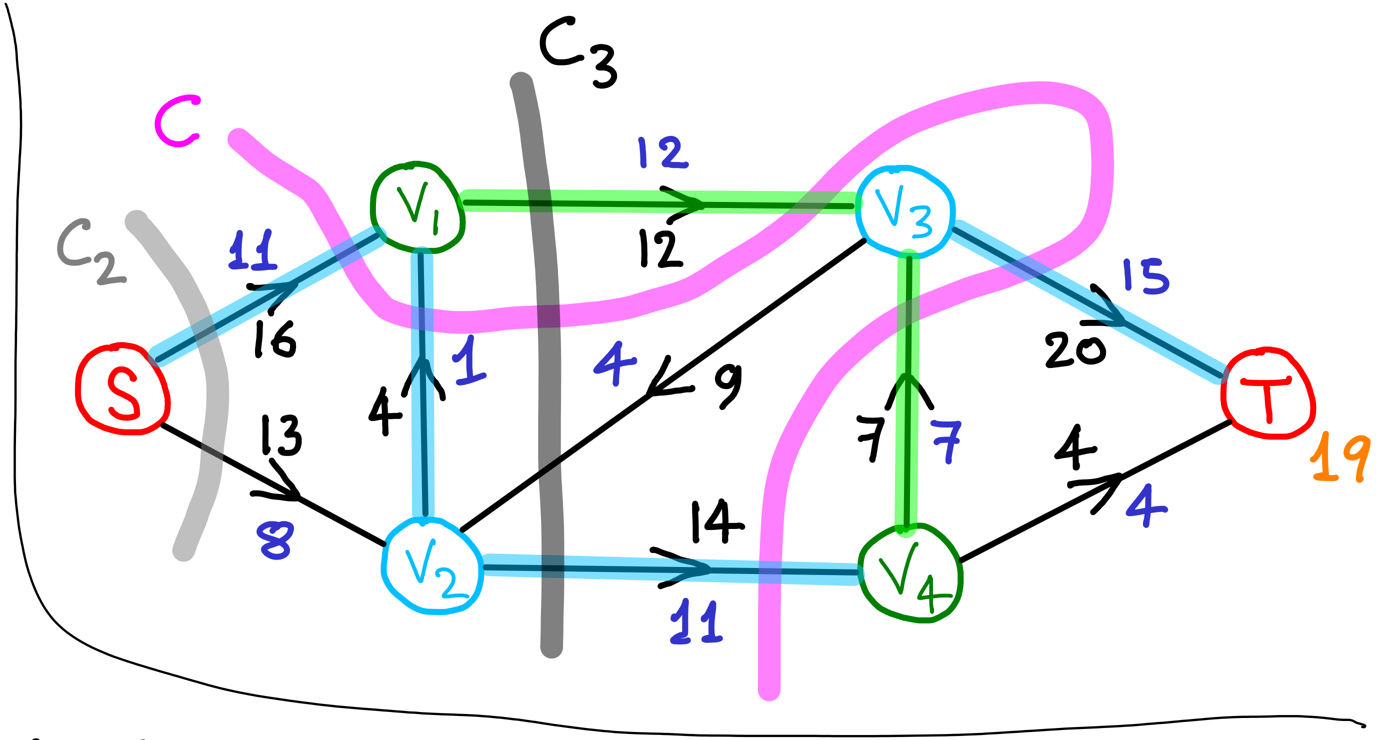
$\rightarrow$ FLOW VALUE $\leq$ $\text{Capacity}(C)$ ...

$$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$$

$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$

$$\text{Capacity}(C) = \sum \text{capacity}$$

$$= 16 + 4 + 14 + 20 = 54$$



(i) $\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) =$ FLOW VALUE

(ii) $\text{Flow}(C) \leq \text{Capacity}(C)$

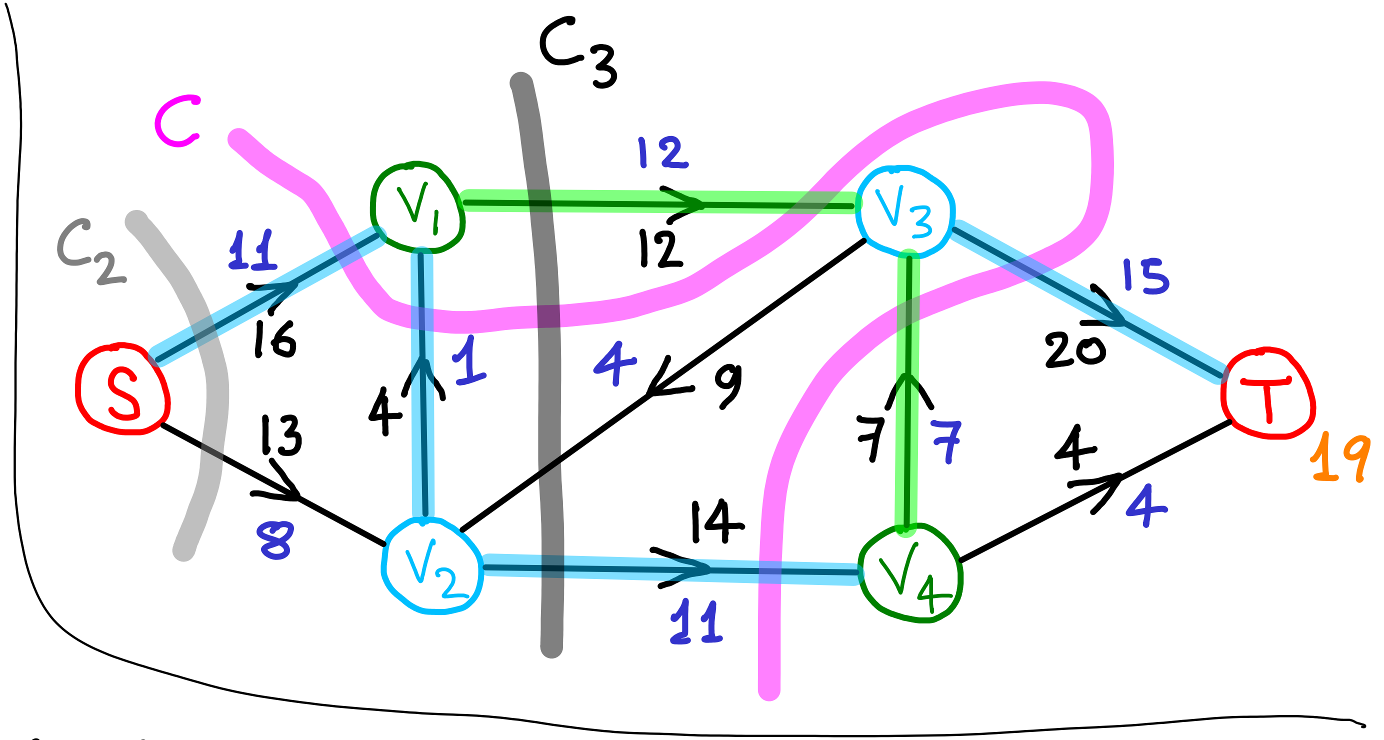
$\rightarrow$ FLOW VALUE $\leq \min\limits_{\text{all cuts}} \{\text{Capacity}\}$

$$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$$

$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$

$$\text{Capacity}(C) = \sum \text{capacity}$$

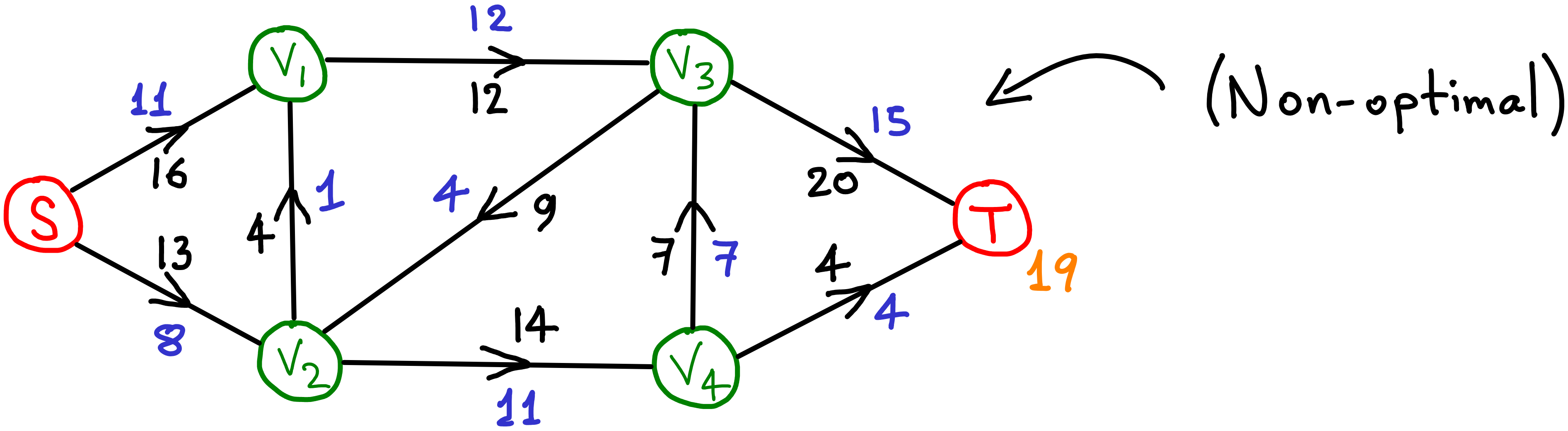
$$= 16 + 4 + 14 + 20 = 54$$

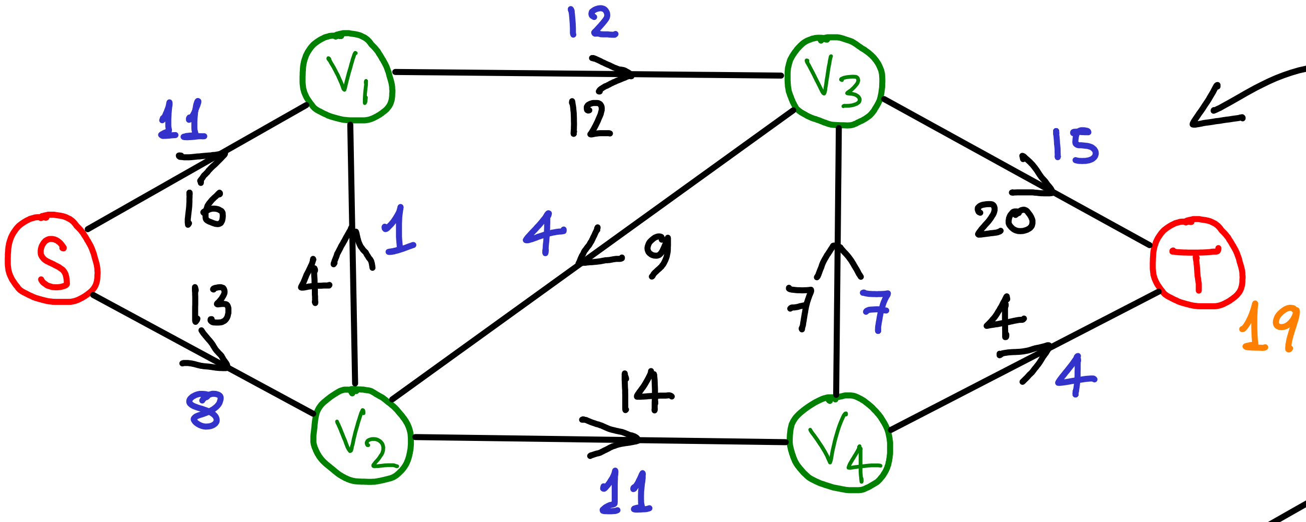


(i) $\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) =$ FLOW VALUE

(ii) $\text{Flow}(C) \leq \text{Capacity}(C)$

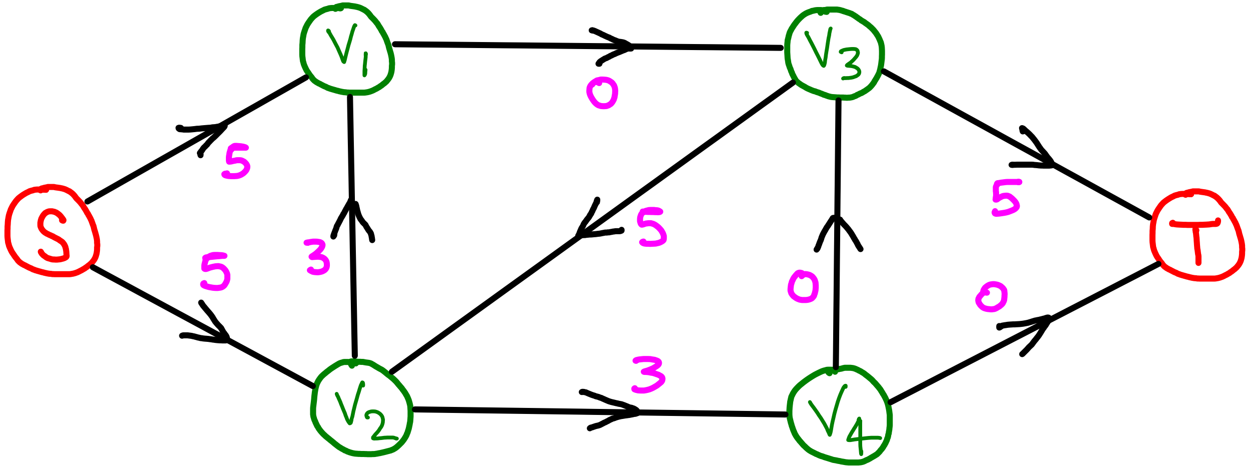
$\rightarrow$ FLOW VALUE $\leq \min_{\text{all cuts}} \{\text{Capacity}\}$ $\longrightarrow$ MAX FLOW $\leq$ MIN CUT

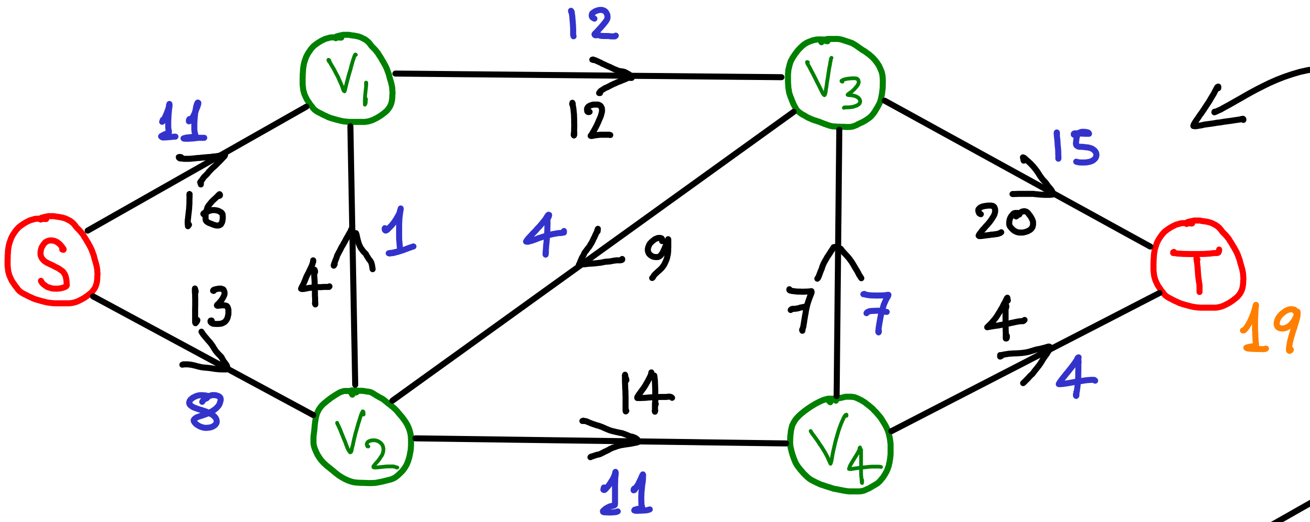

S
V1
V2
V3
V4
T
11
16
13
8
12
12
4
1
4
9
14
11
7
7
20
15
4
4
19
(Non-optimal)



(Non-optimal)

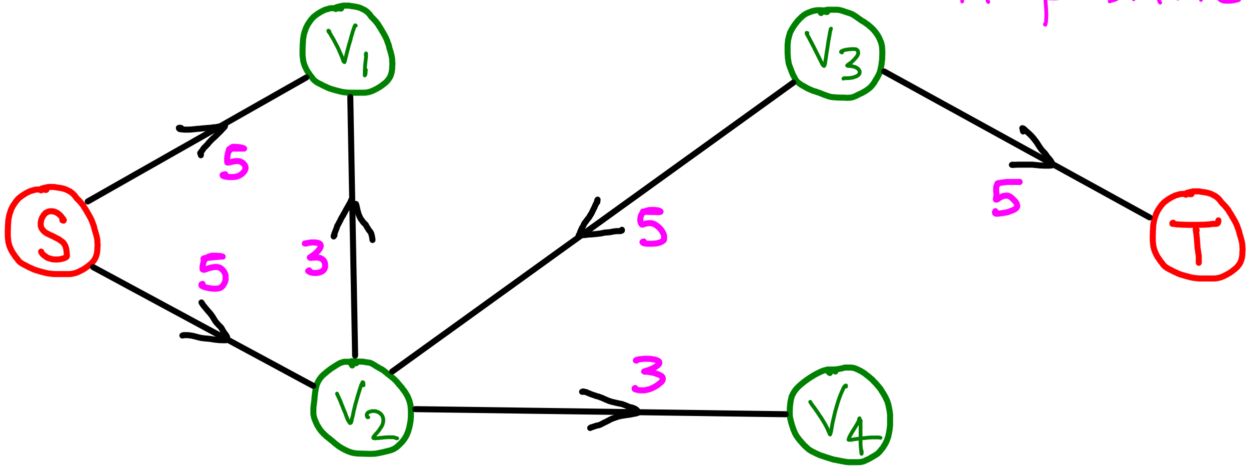
Unused capacities

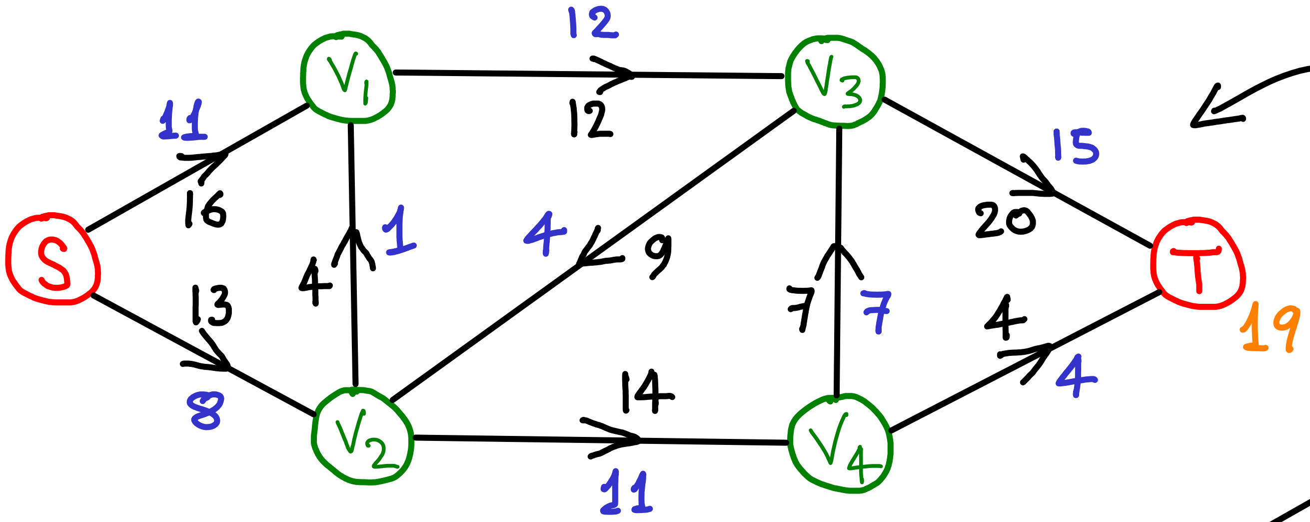




(Non-optimal)

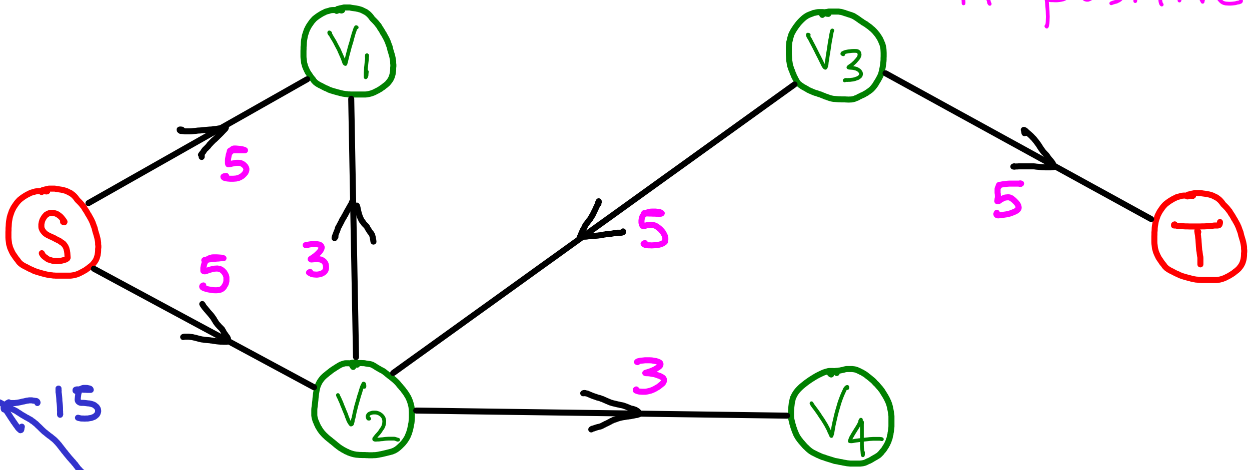
Unused capacities, if positive



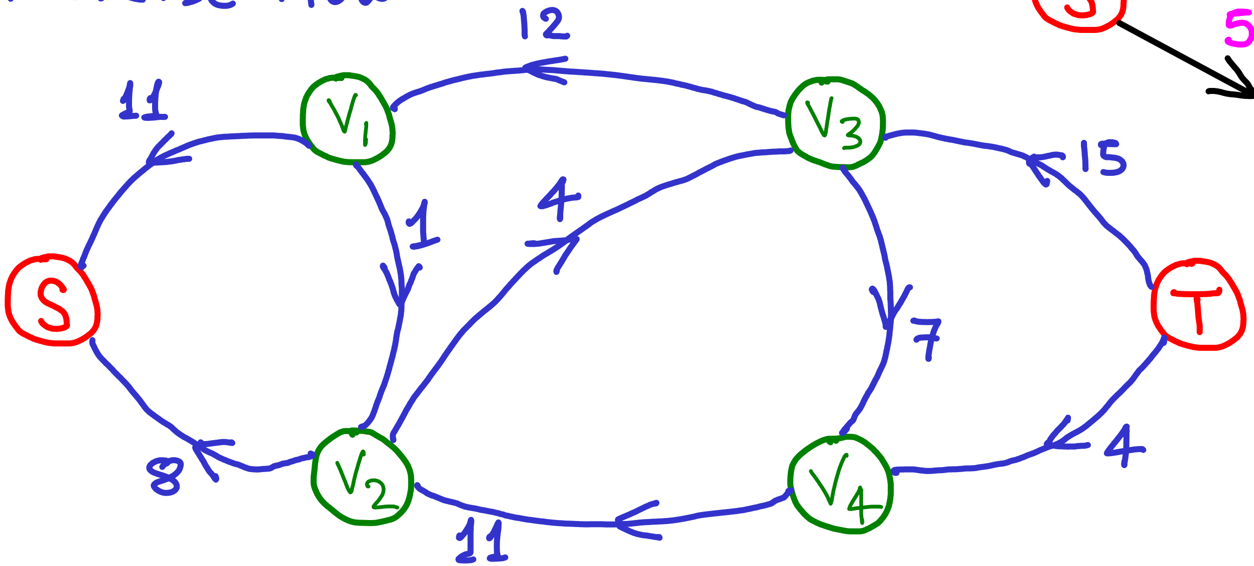


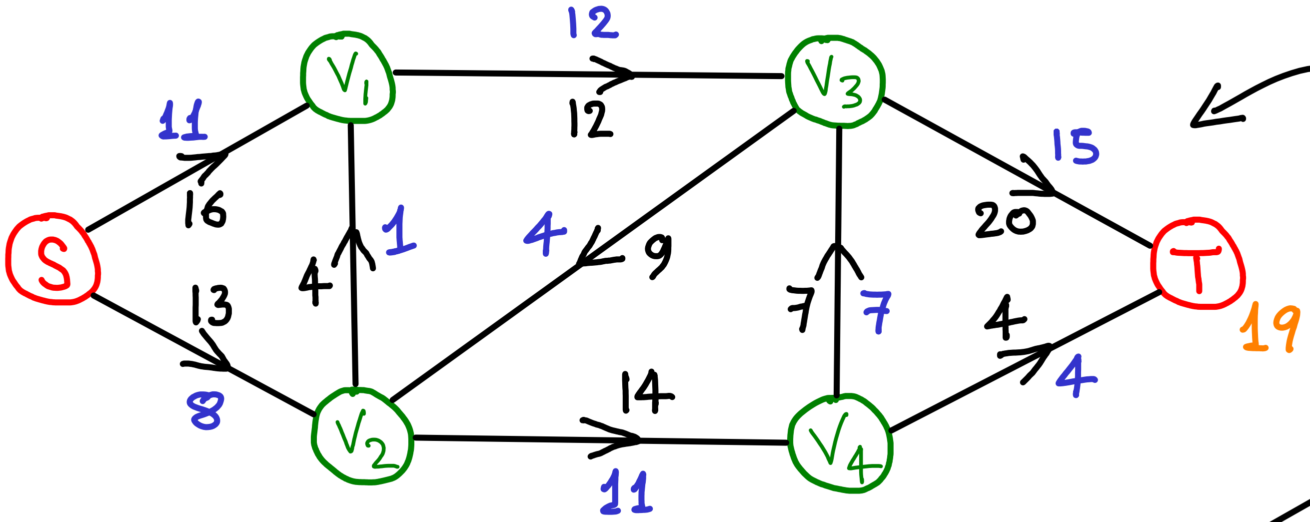
(Non-optimal)

Unused capacities, if positive



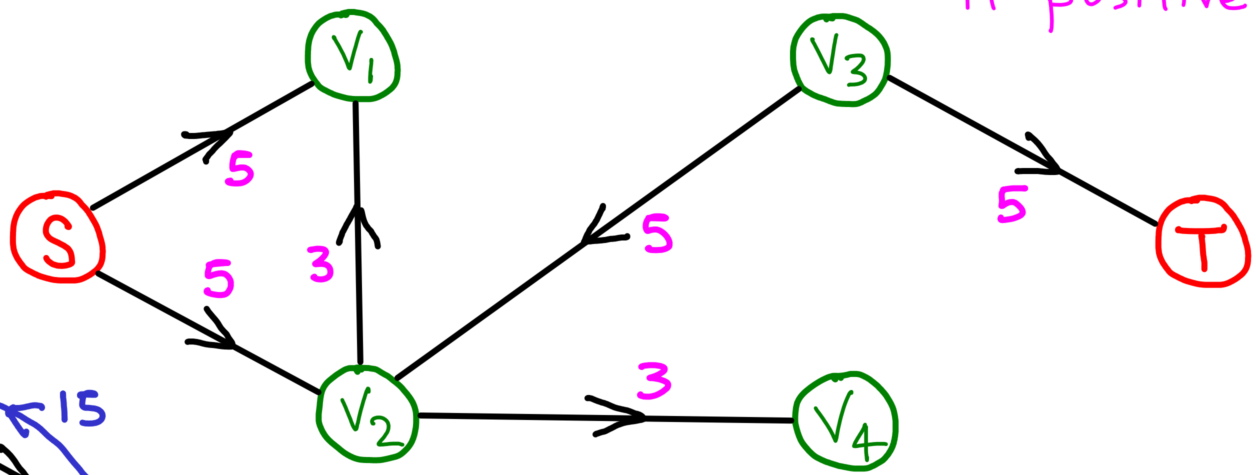
Reverse flow

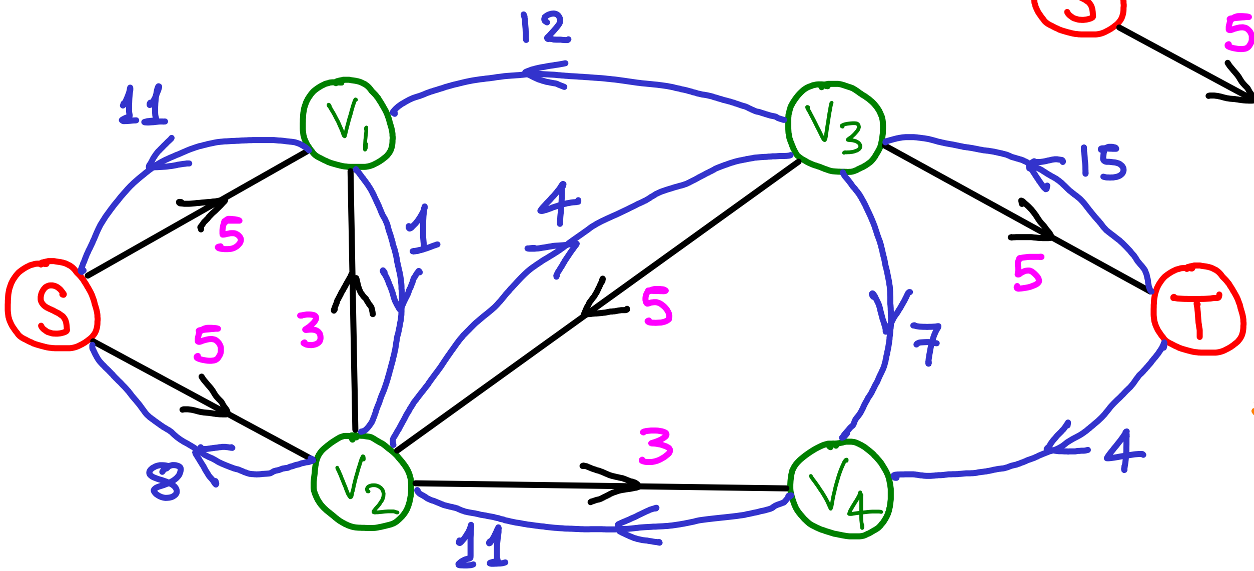




(Non-optimal)

Unused capacities, if positive



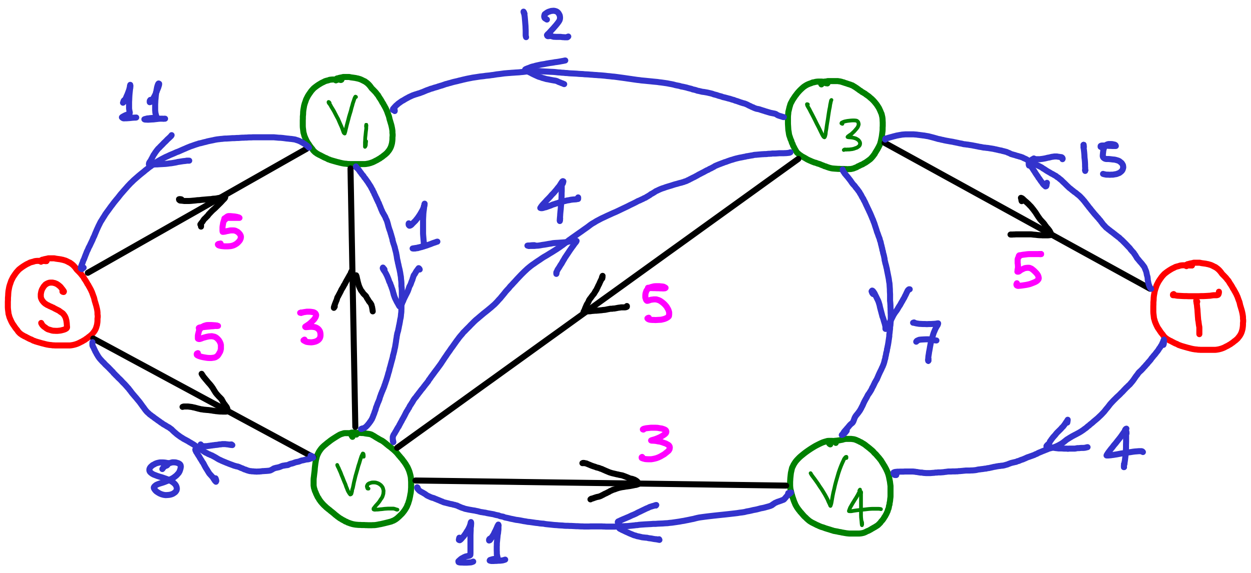


Residual network (all potential flow changes)

Residual network → all potential flow changes

Blue edges = reverse flow = potential flow decrease

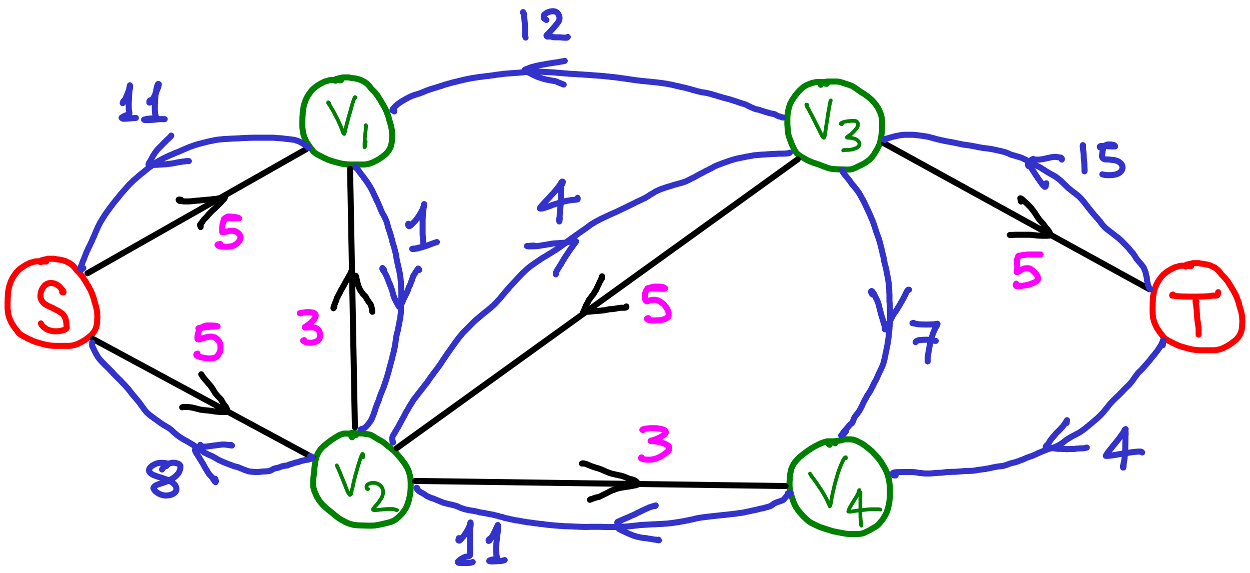
Black edges (pink labels) = potential flow increase



Residual network → all potential flow changes

Blue edges = reverse flow = potential flow decrease

Black edges (pink labels) = potential flow increase

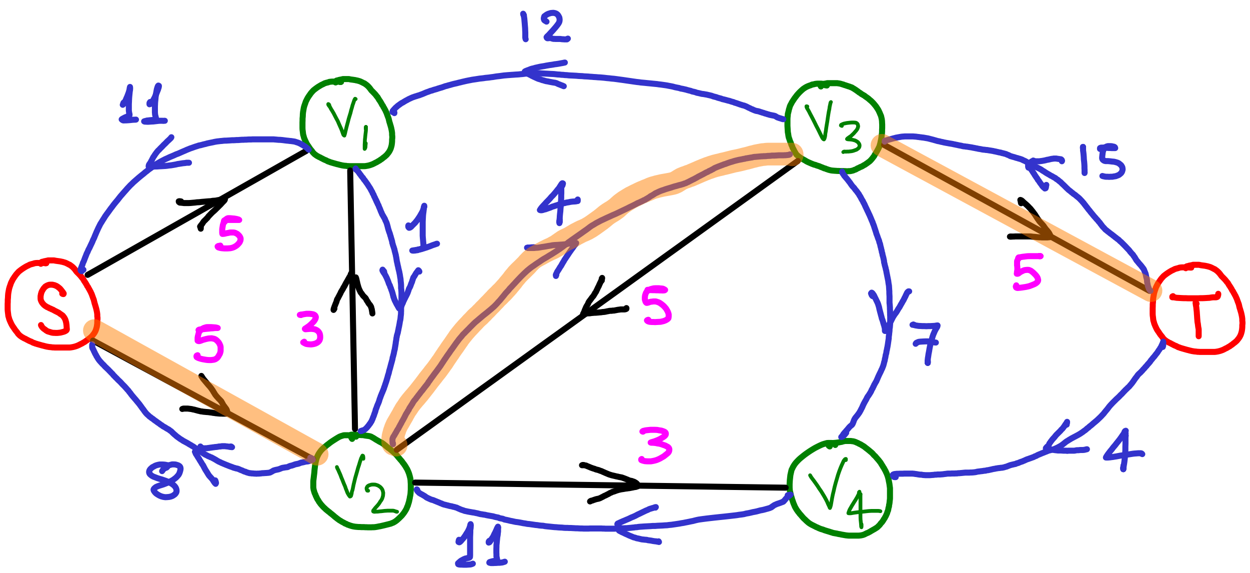


Every edge in network is "represented" here, at least once.

Residual network → all potential flow changes

Blue edges = reverse flow = potential flow decrease

Black edges (pink labels) = potential flow increase



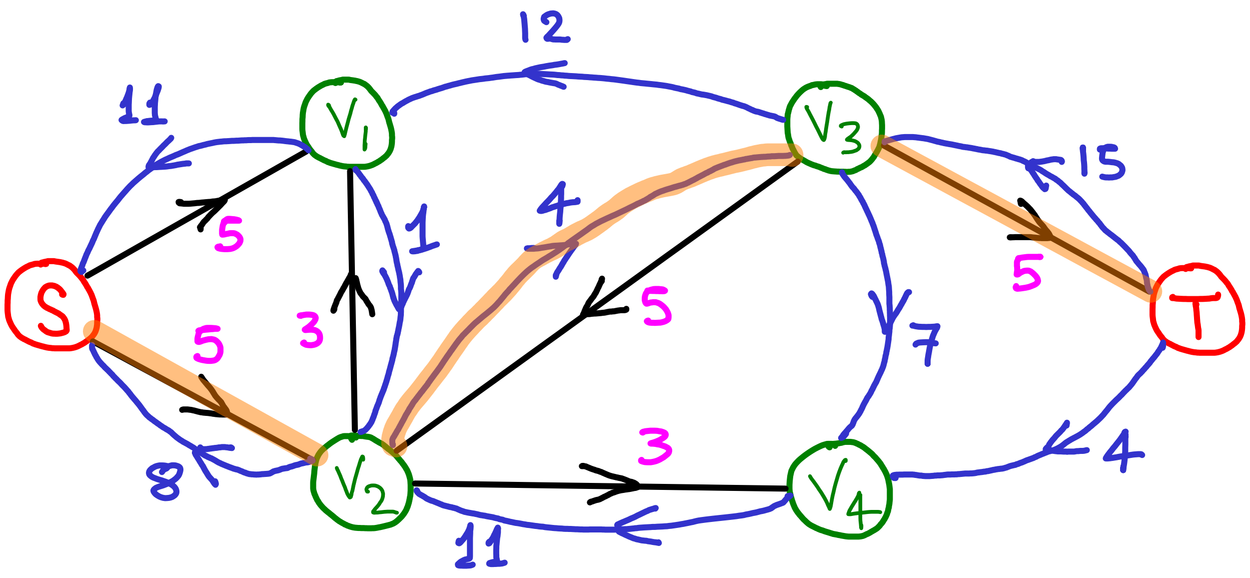
Every edge in network is "represented" here, at least once.

Any directed S→T path in the residual network means we can improve the solution.

Residual network → all potential flow changes

Blue edges = reverse flow = potential flow decrease

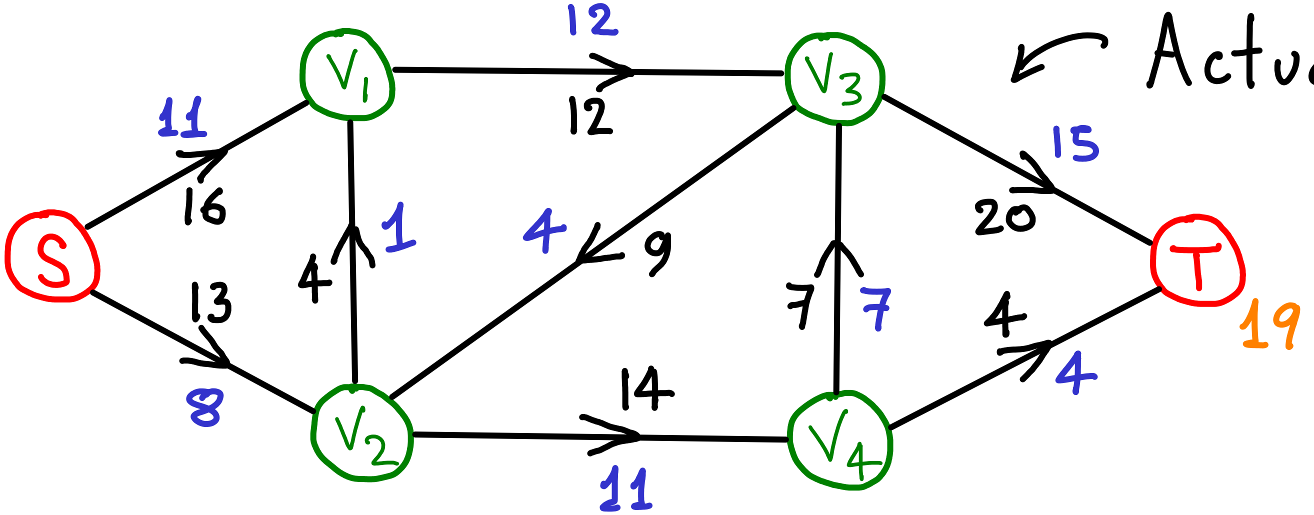
Black edges (pink labels) = potential flow increase



Every edge in network is "represented" here, at least once.

Any directed S→T path in the residual network means we can improve the solution.

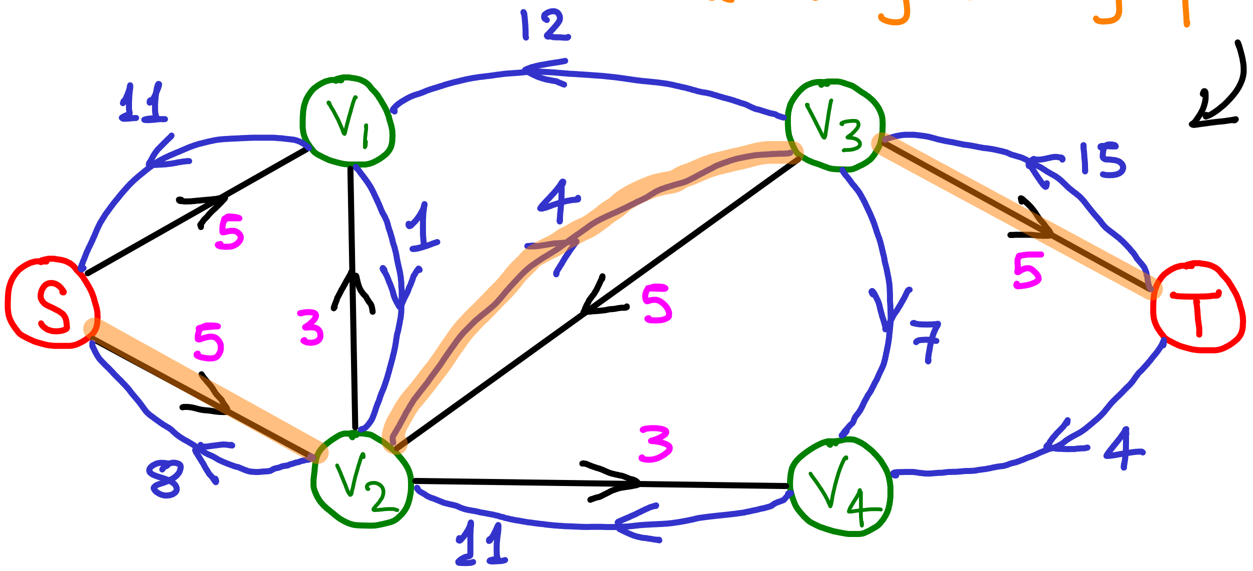
S → V₂ → V₃ → T : AUGMENTING PATH

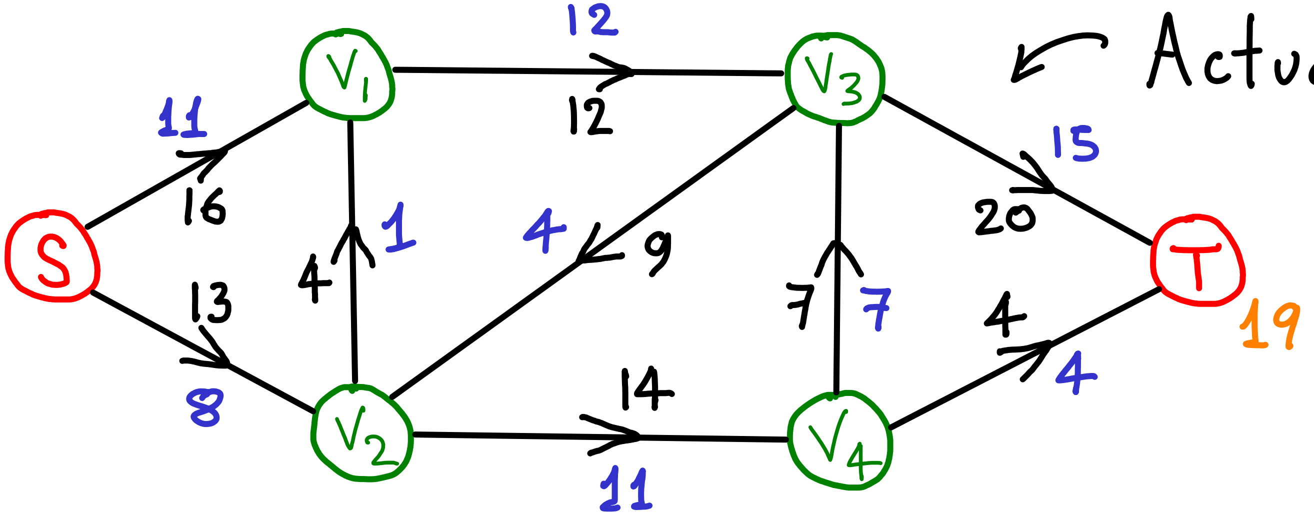


Actual network

Residual network &

an augmenting path

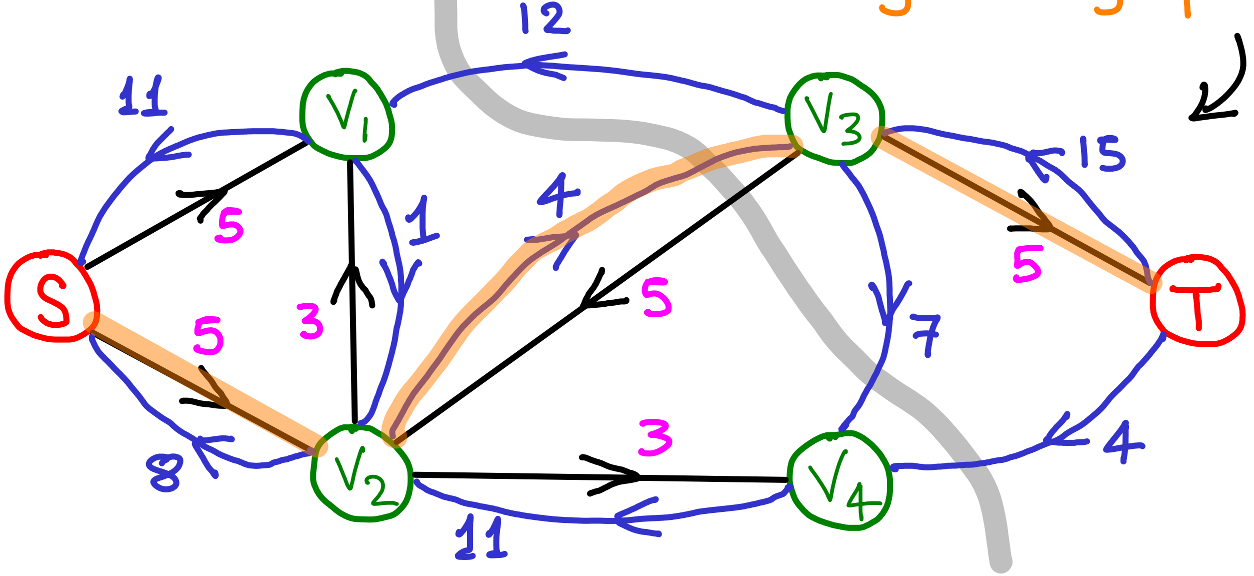




Actual network

Residual network &

an augmenting path

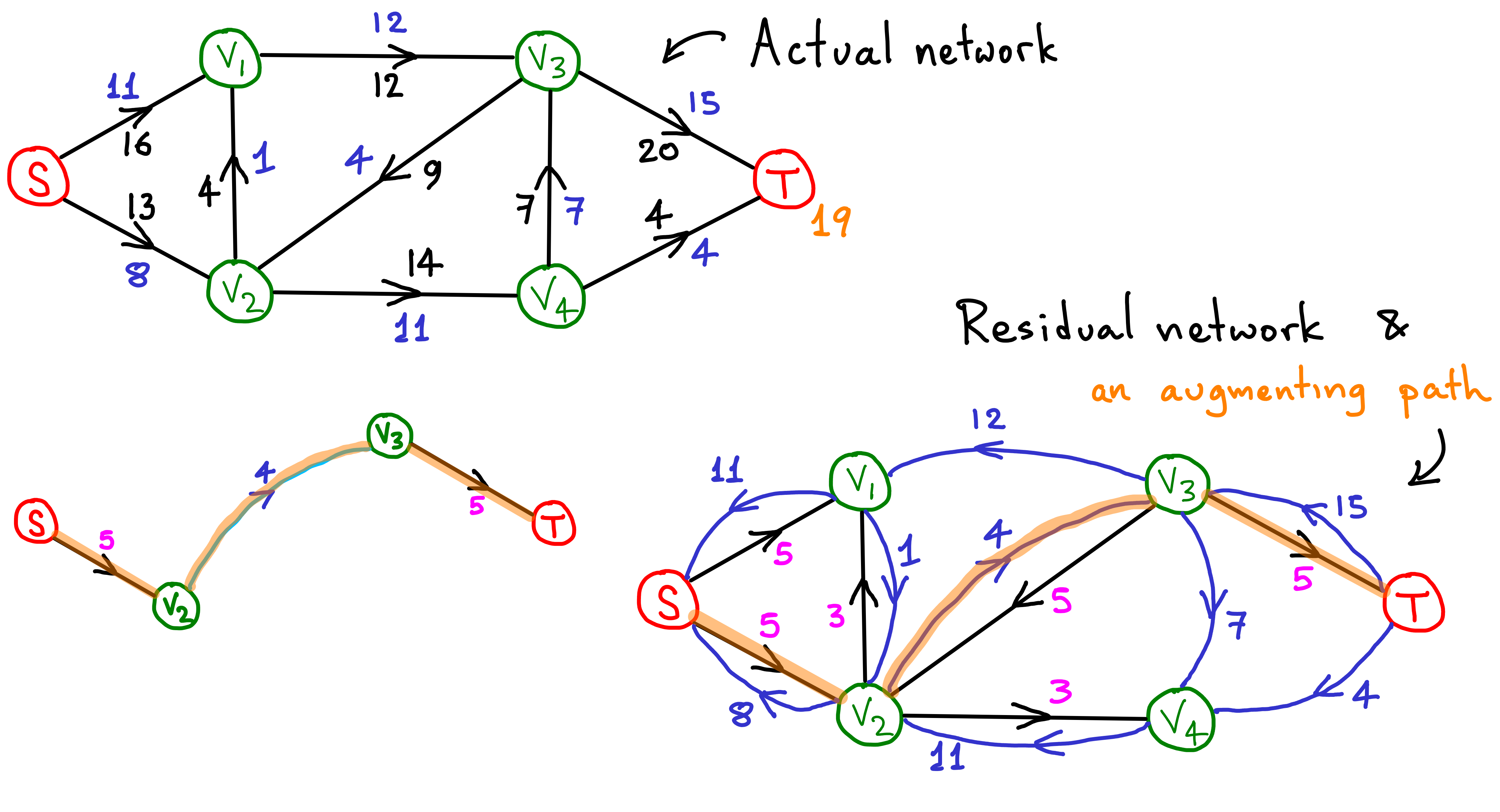


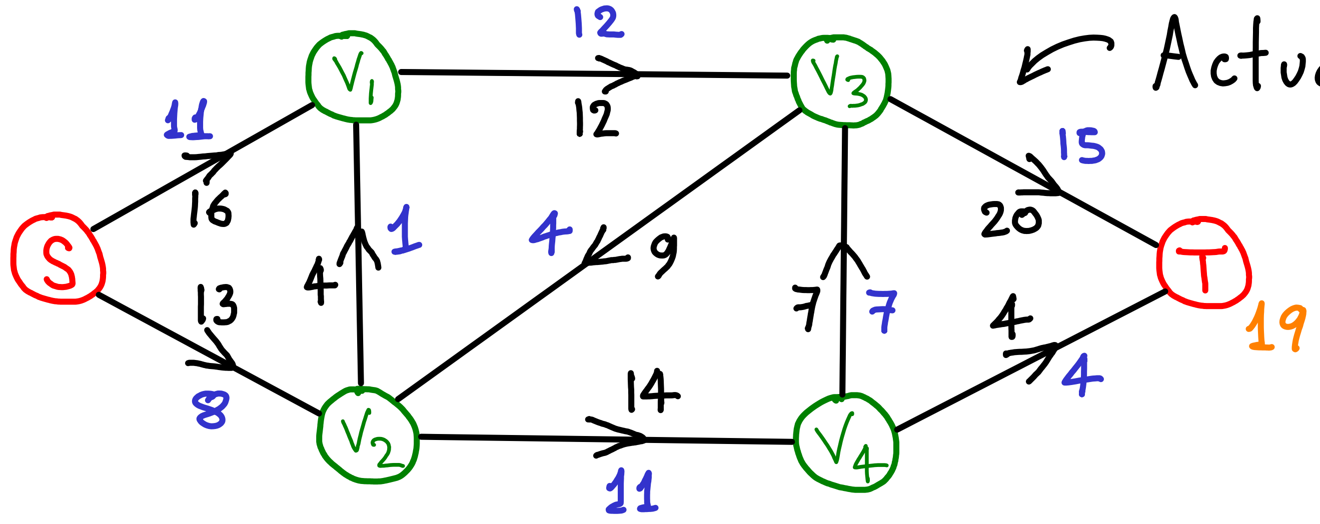
The residual network also has a MIN. cut X.

(min capacity of all cuts)

This gives a "residual capacity" (=4)

= max. increase in flow.

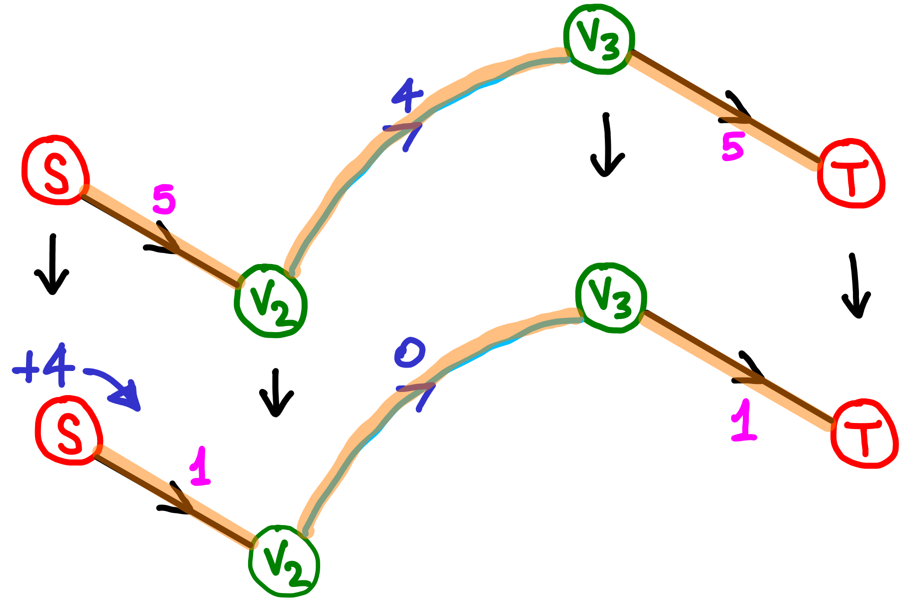

Actual network
Residual network &
an augmenting path
S
V1
V2
V3
V4
T
19

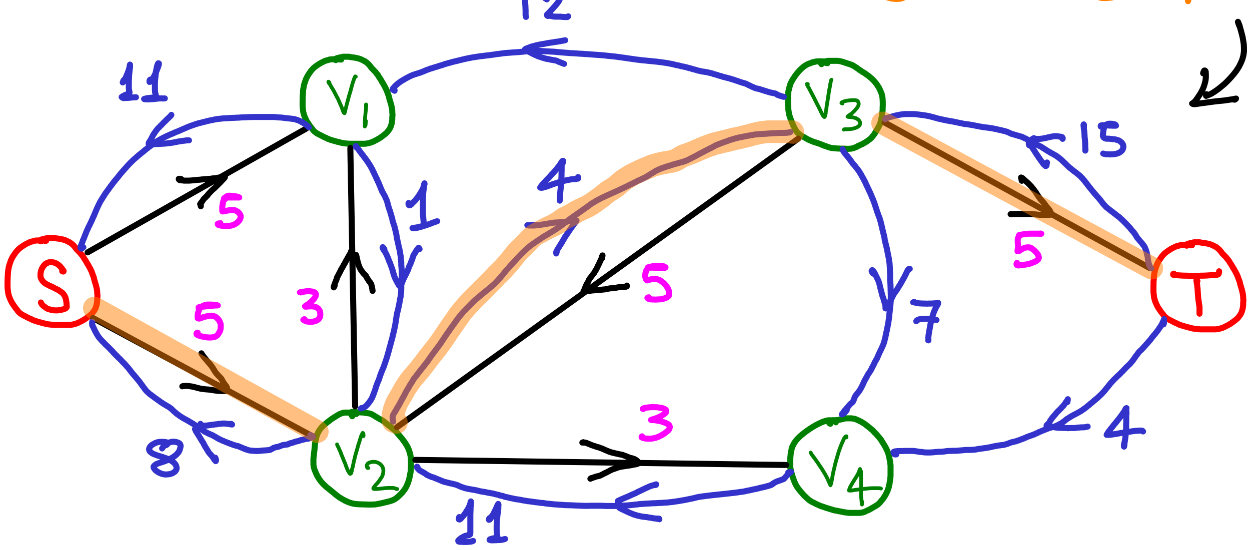


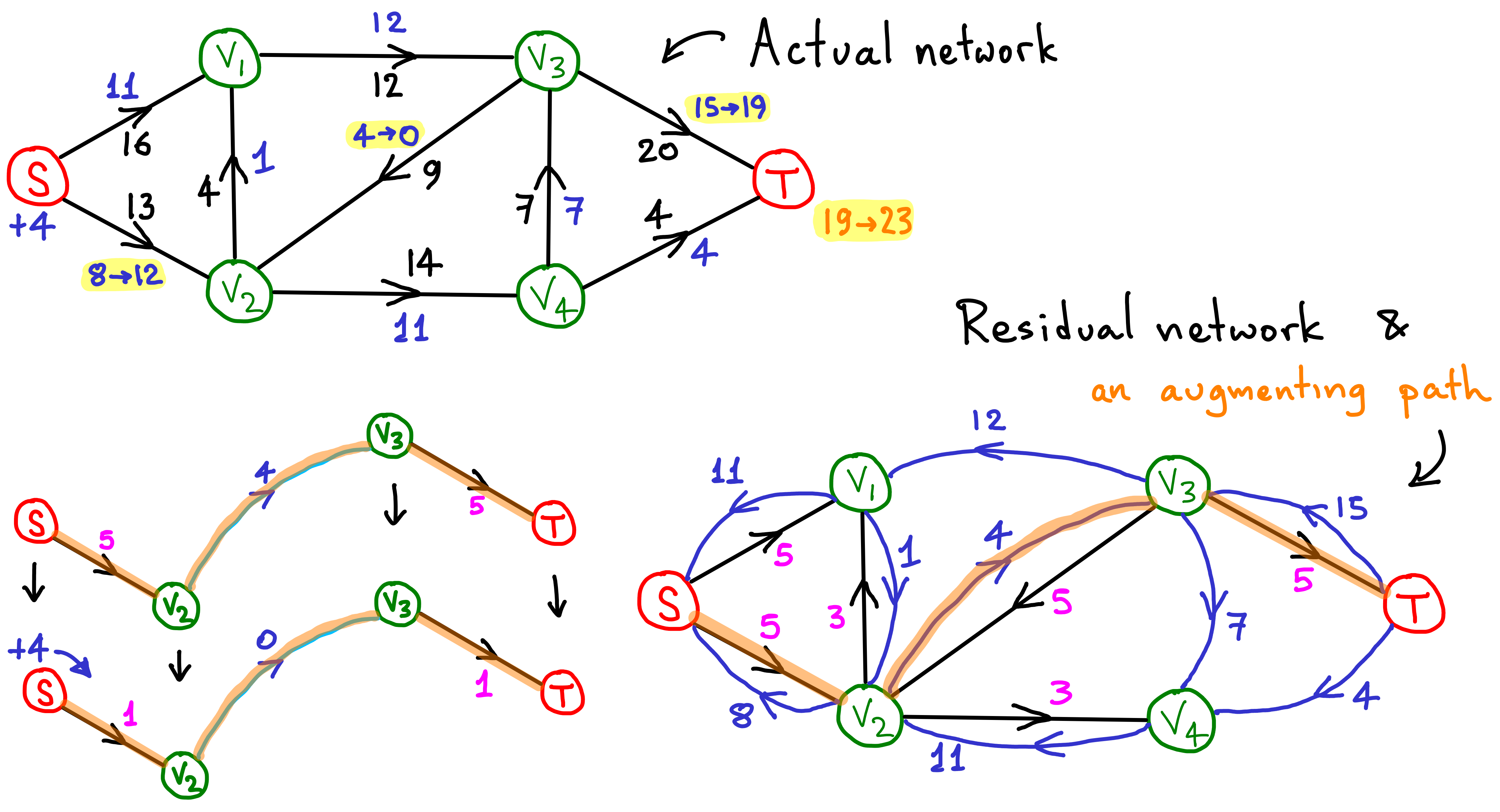
Actual network

Residual network &

an augmenting path

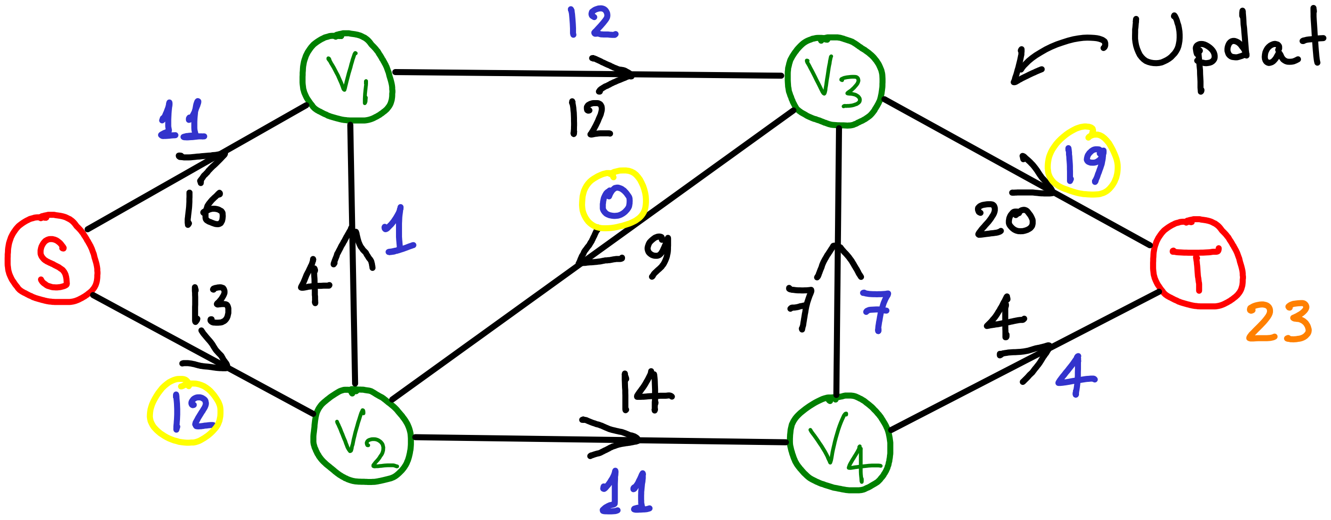




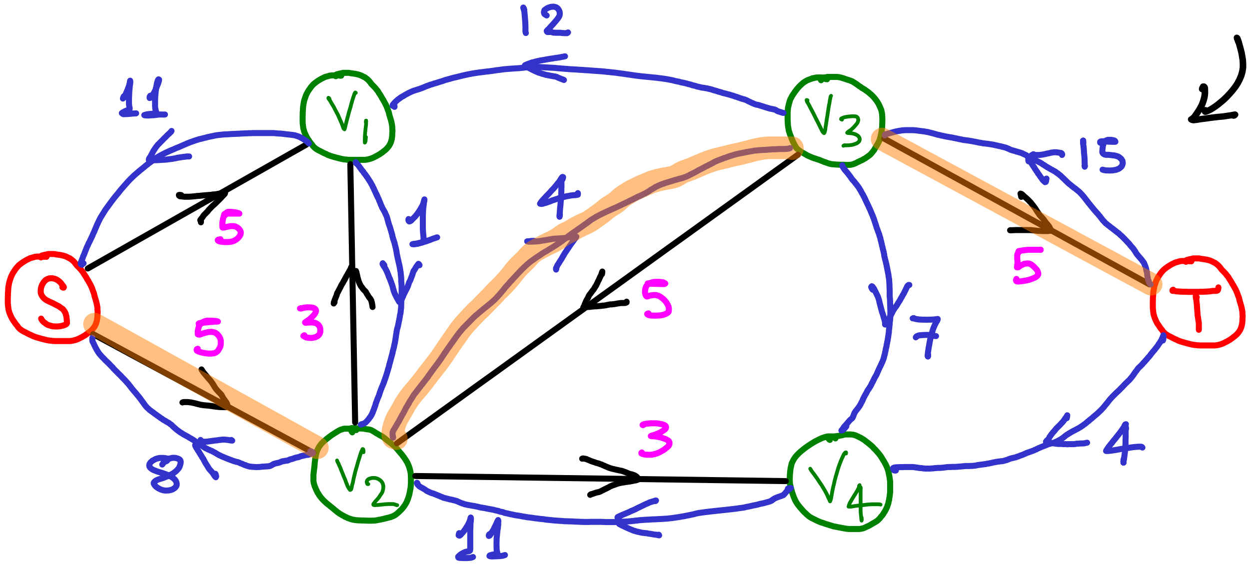


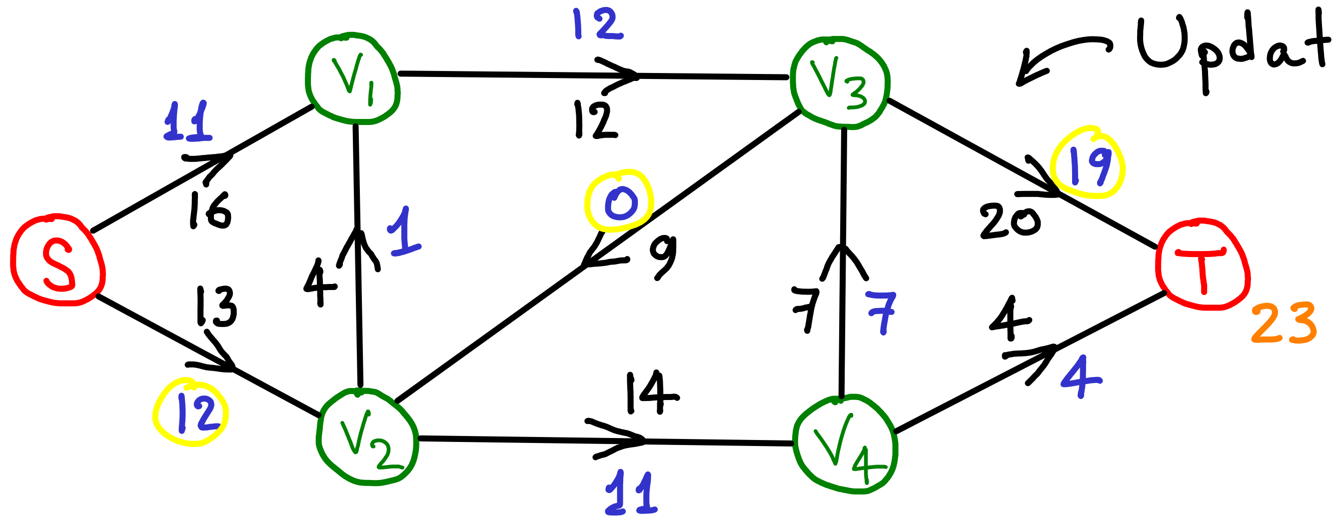
Actual network
S
V1
V2
V3
V4
T
+4
11
16
13
8→12
12
12
4
1
4→0
9
14
11
7
7
15→19
20
4
4
19→23
Residual network &
an augmenting path
5
4
5
0
1
1
+4
12
11
15
5
1
3
4
5
5
7
3
8
11

Updated flow



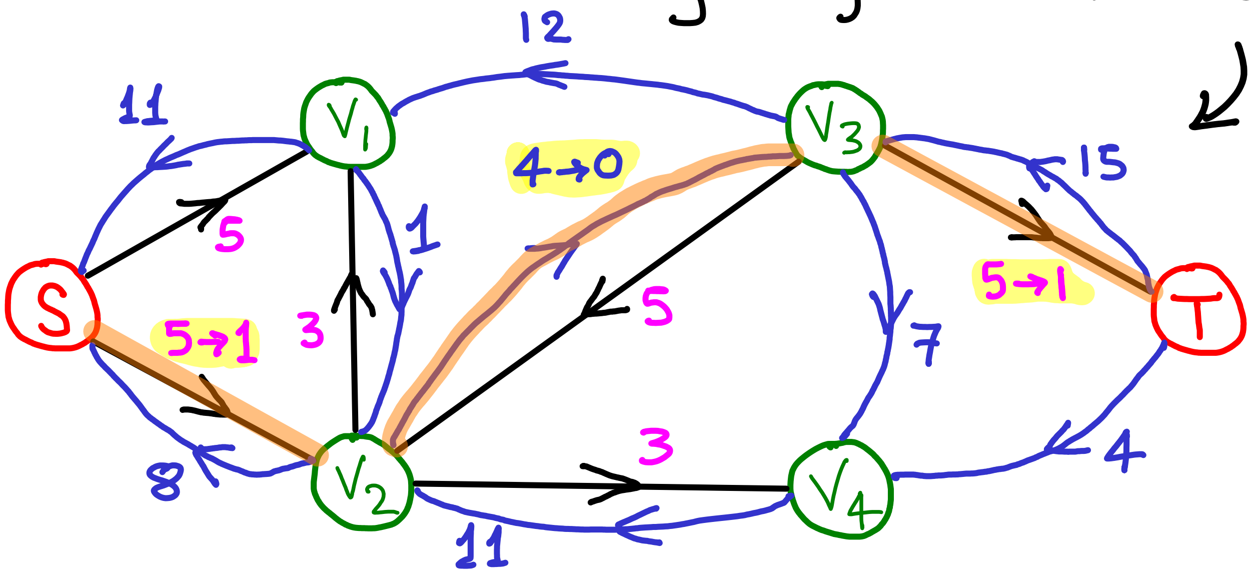
Residual network
to be reconstructed

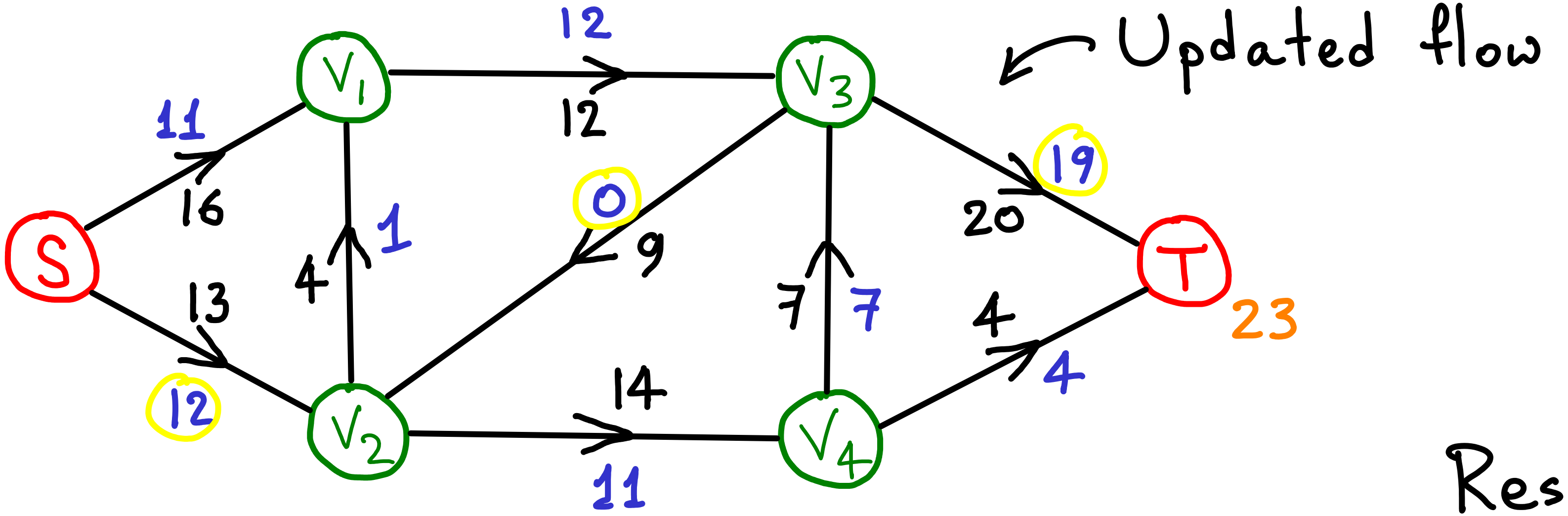


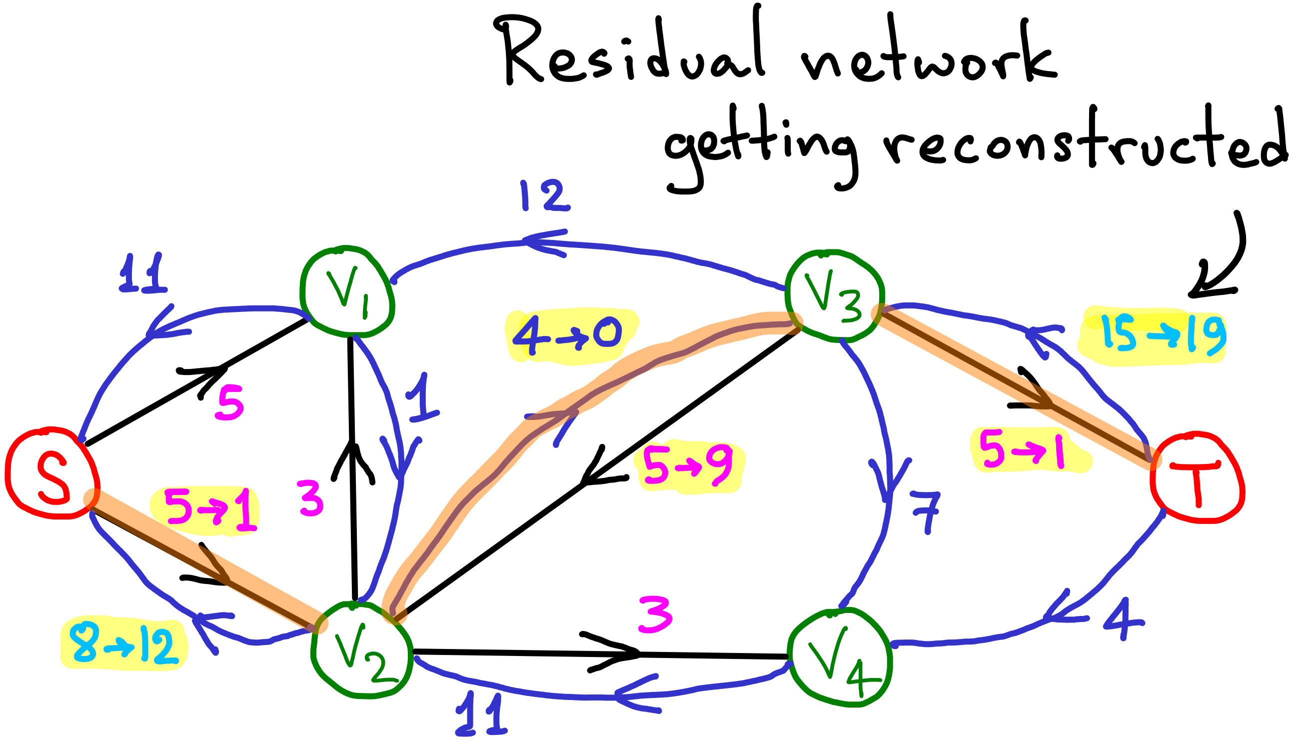


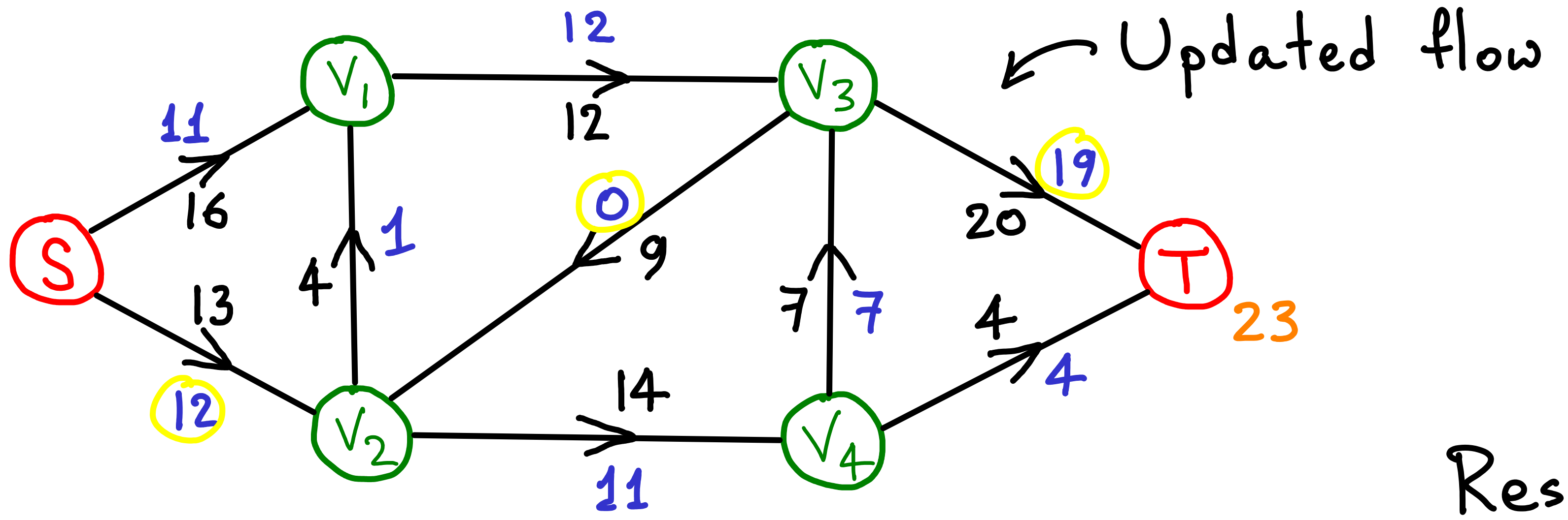
Updated flow

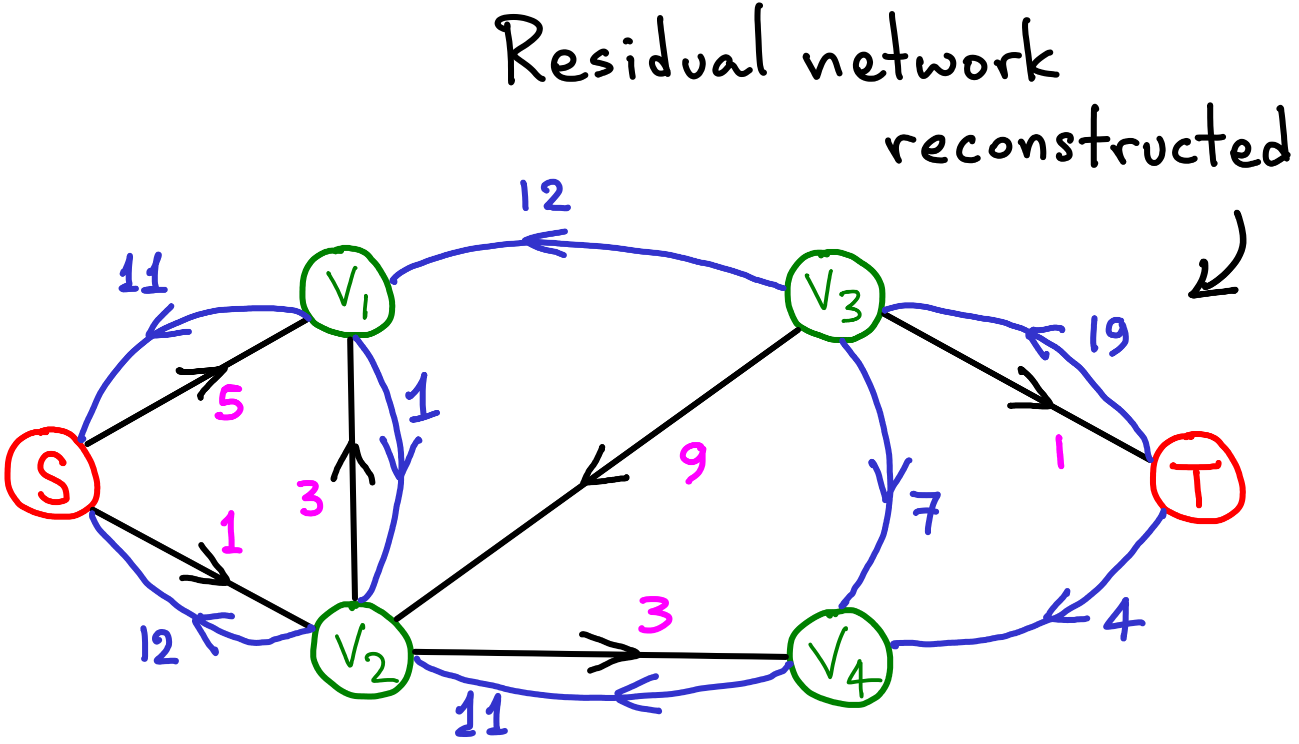
Residual network getting reconstructed

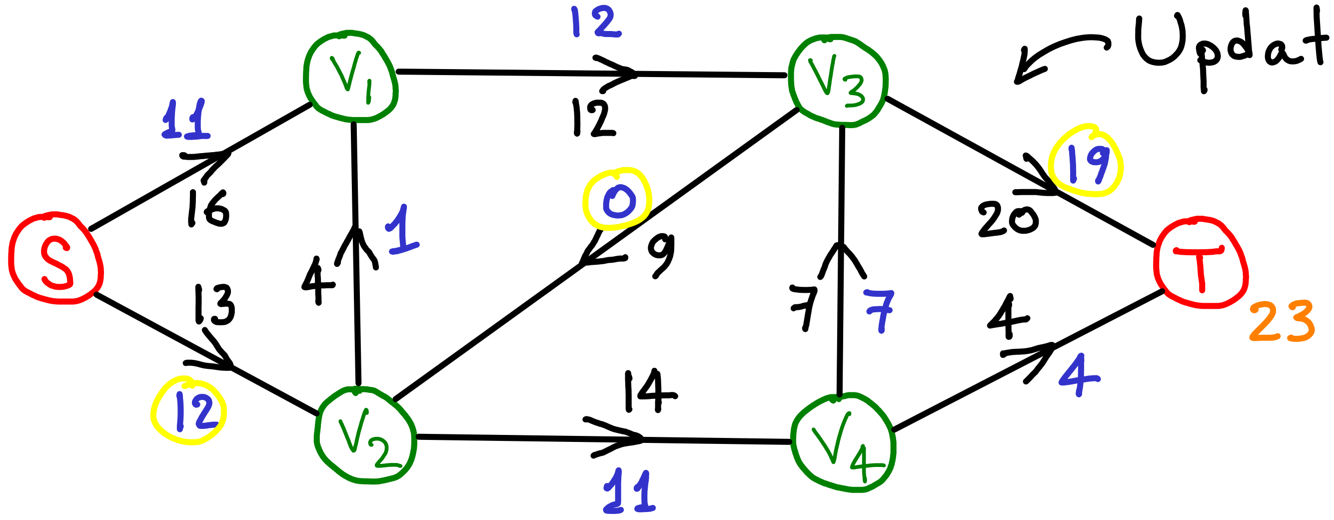



Updated flow
S
V1
V2
V3
V4
T
11
16
13
12
12
12
4
1
0
9
14
11
7
7
20
19
4
4
23


Residual network
getting reconstructed
S
V1
V2
V3
V4
T
11
5
5→1
8→12
3
1
12
4→0
5→9
3
11
7
15→19
5→1
4

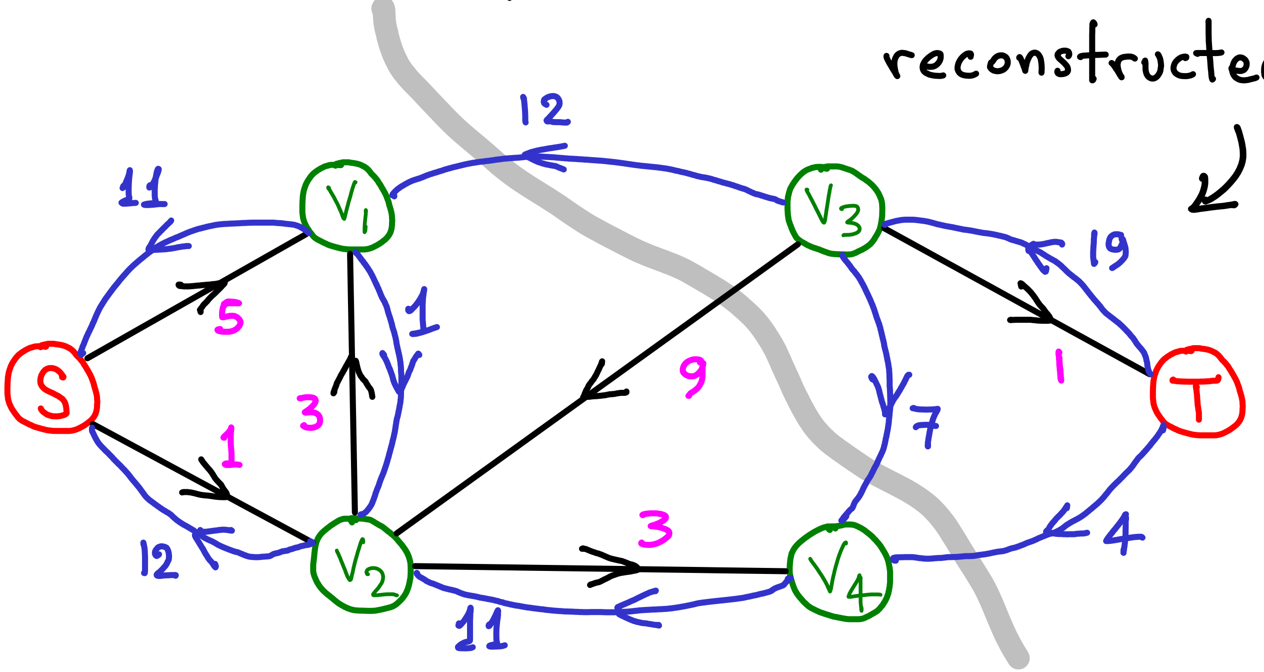

Updated flow
S
V1
V2
V3
V4
T
12
12
11
16
13
12
4
1
0
9
7
7
20
19
14
11
4
4
23


Residual network reconstructed
S
V1
V2
V3
V4
T
12
11
5
1
12
3
1
9
7
1
19
3
11
4



Updated flow

Residual network reconstructed



Residual capacity = 0

no augmenting paths

if $\exists$ augmenting path then flow can increase

if $\exists$ augmenting path then flow can increase

if MAX flow then no augmenting path.

if $\exists$ augmenting path then flow can increase

if MAX flow then no augmenting path.

---

if flow equals capacity of some cut, then we have MAX flow

if $\exists$ augmenting path then flow can increase

if MAX flow then no augmenting path.

---

if flow equals capacity of some cut, then we have MAX flow

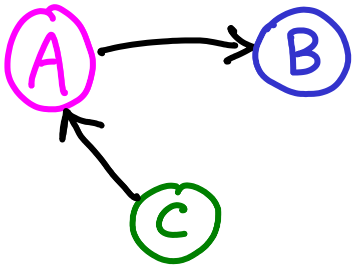
(MAX FLOW $\leq$ MIN CUT)

if ∃ augmenting path then flow can increase

if $\underbrace{\text{MAX flow}}_{A}$ then $\underbrace{\text{no augmenting path.}}_{B}$

if $\underbrace{\text{flow equals capacity of some cut}}_{C}$, then we have $\underbrace{\text{MAX flow}}_{A}$

(MAX FLOW ≤ MIN CUT)



if ∃ augmenting path then flow can increase

if $\underbrace{\text{MAX flow}}_{A}$ then $\underbrace{\text{no augmenting path}}_{B}$.

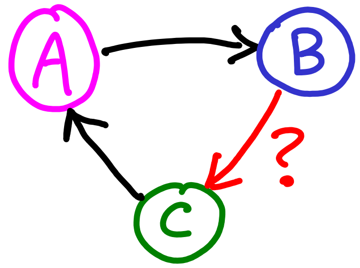
---

if $\underbrace{\text{flow equals capacity of some cut}}_{C}$, then we have $\underbrace{\text{MAX flow}}_{A}$

(MAX FLOW ≤ MIN CUT)

To prove: if $\underbrace{\text{no augmenting path}}_{B}$

then $\underbrace{\text{flow equals capacity of some cut}}_{C}$



if ∃ augmenting path then flow can increase

if $\underbrace{\text{MAX flow}}_{A}$ then $\underbrace{\text{no augmenting path.}}_{B}$

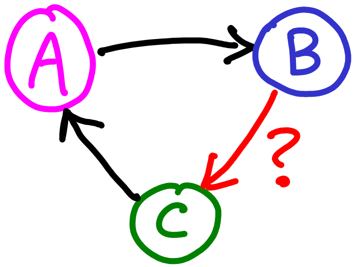
---

if $\underbrace{\text{flow equals capacity of some cut}}_{C}$, then we have $\underbrace{\text{MAX flow}}_{A}$

(MAX FLOW $\leq$ MIN CUT)

To prove: if $\underbrace{\text{no augmenting path}}_{B}$

then $\underbrace{\text{flow equals capacity of some cut}}_{C}$

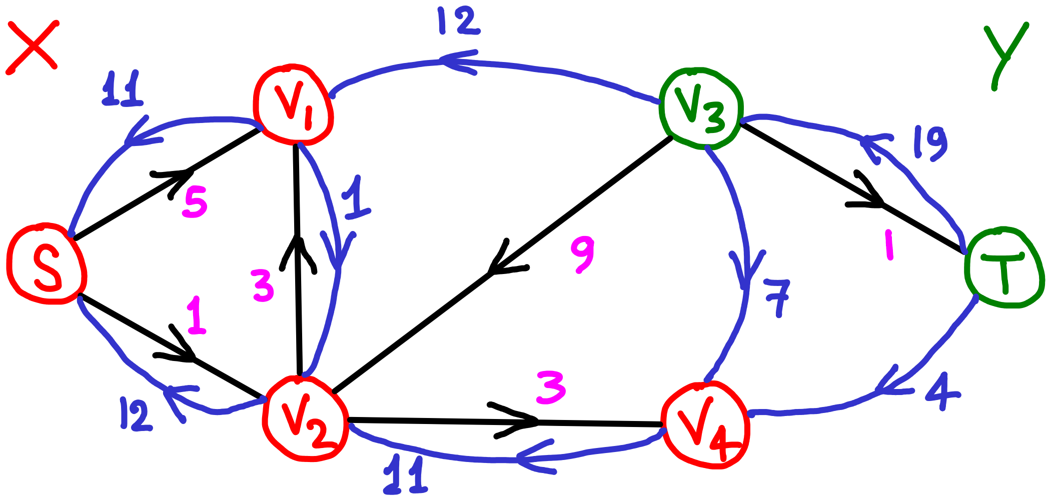


Implies MAX FLOW = MIN CUT

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

---



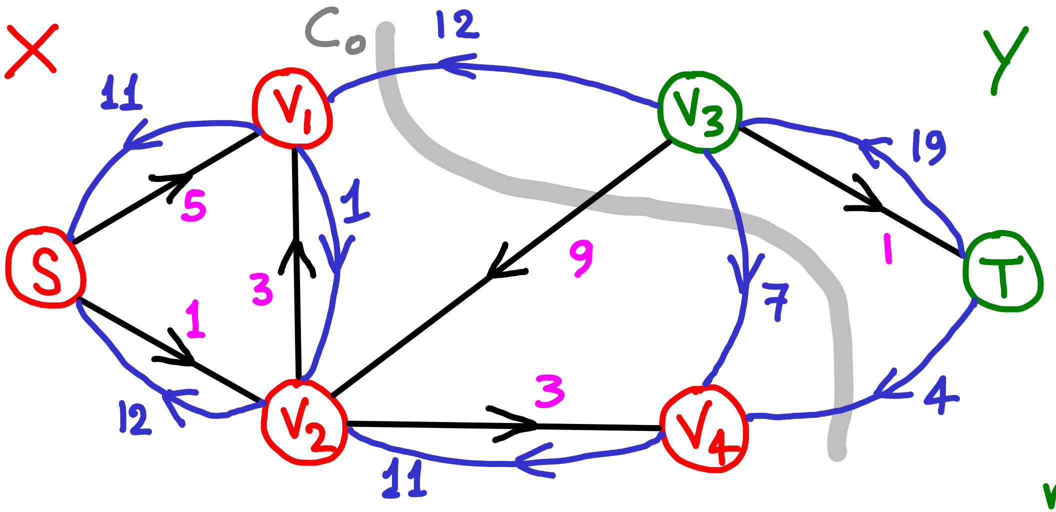
In residual network:

X : {vertices reachable from S}

Y = V - X

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

---



In residual network:

X : {vertices reachable from S}

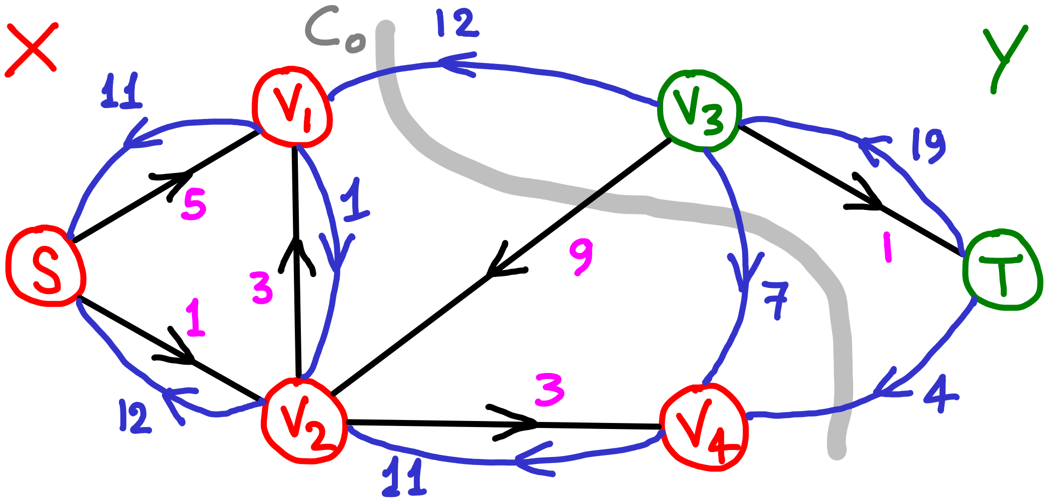
Y = V - X

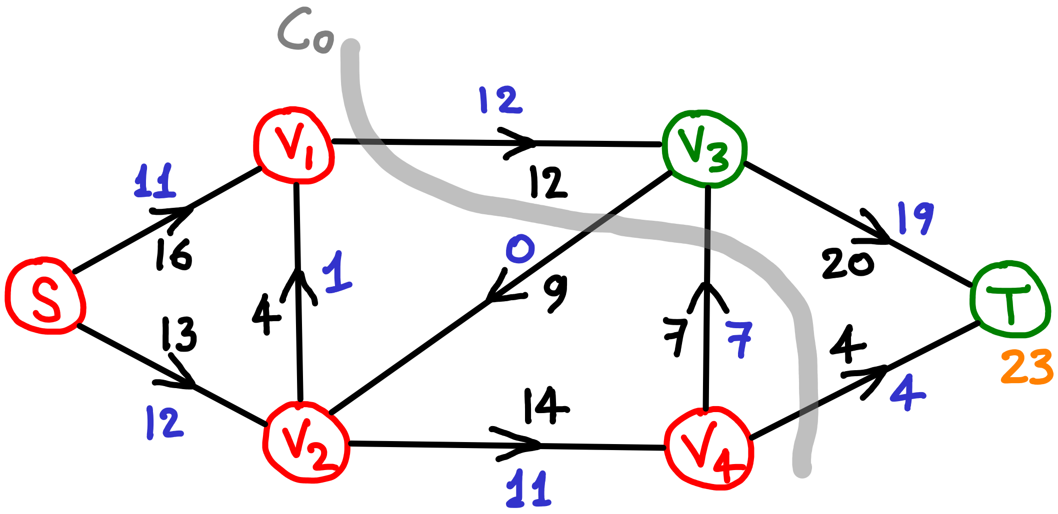
no augmenting path

↳ T is in Y

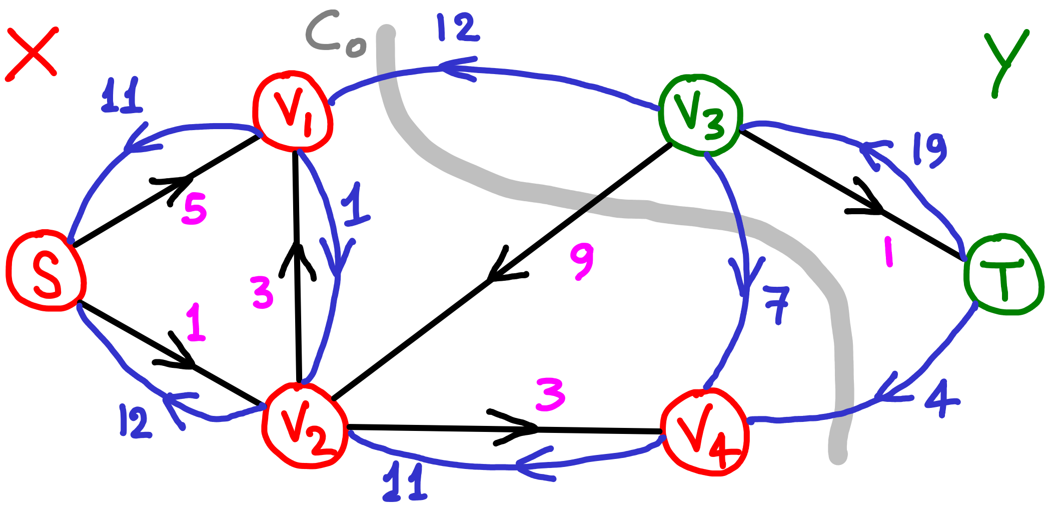
↳ X & Y define a cut, C_0

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

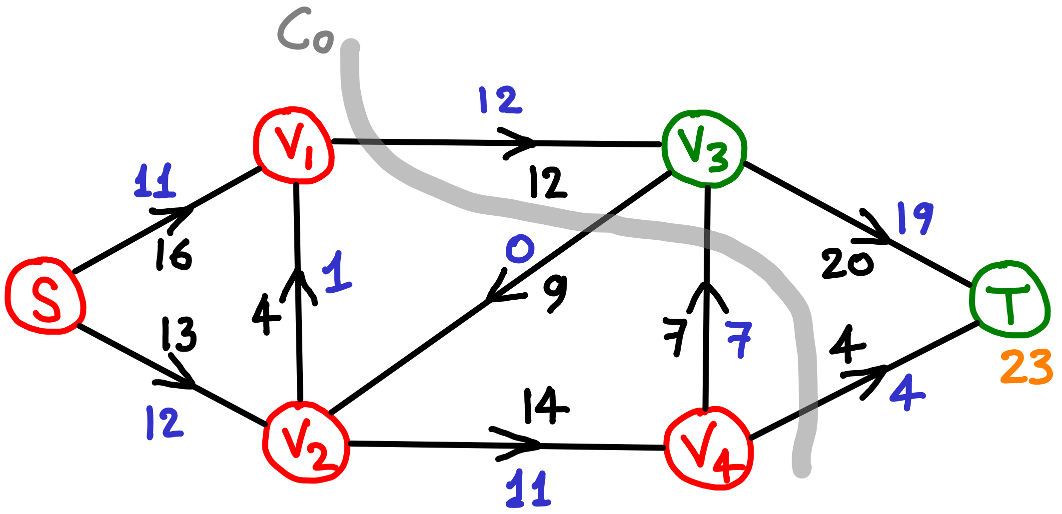




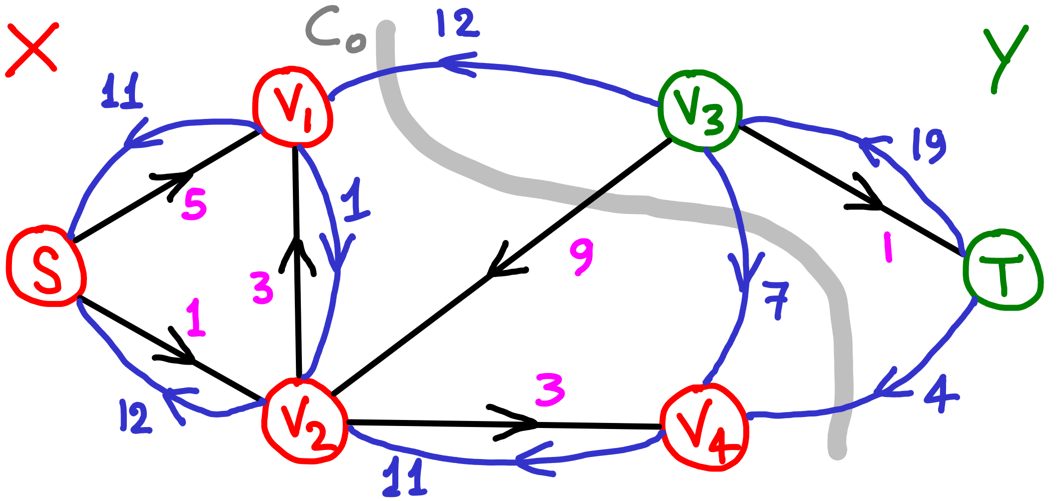
no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut



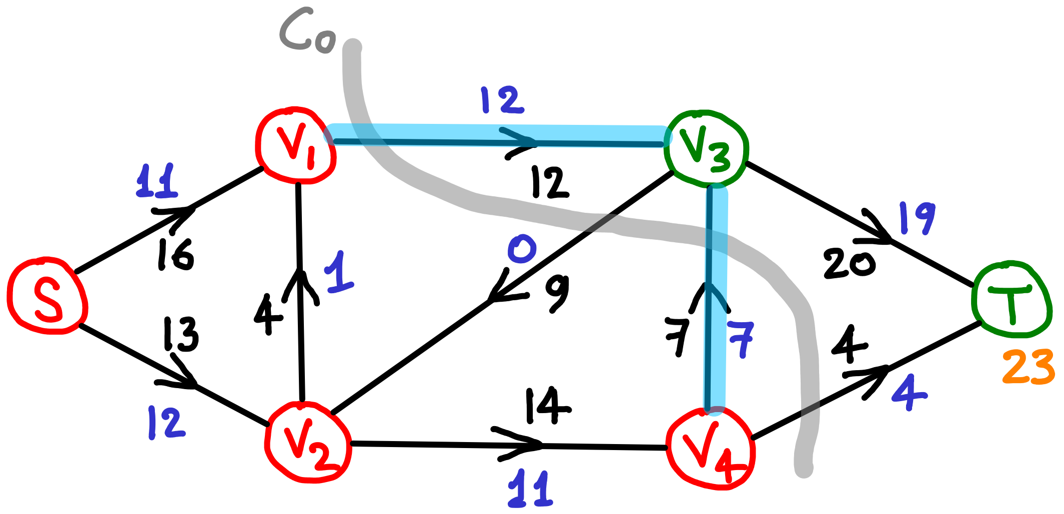
By definition of C_0, no edge exists from X to Y in residual.



no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut



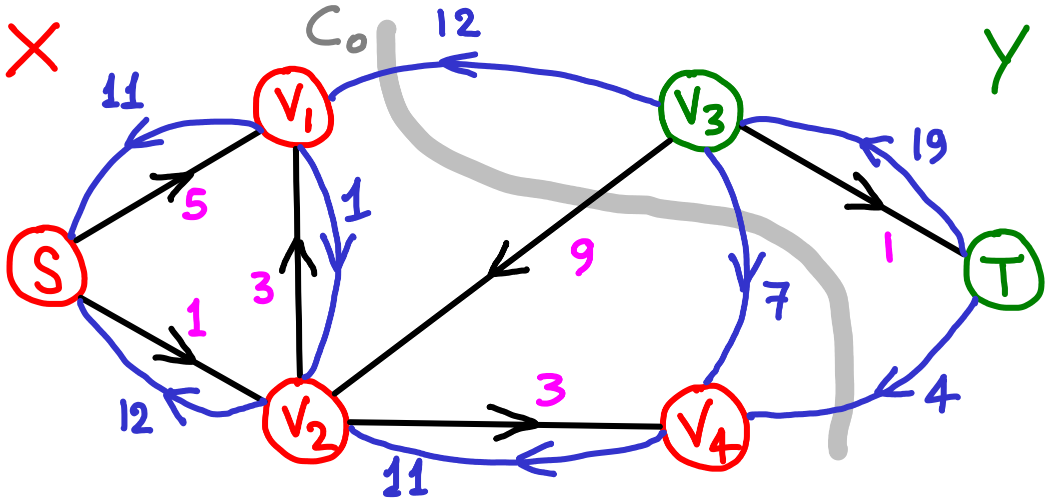
By definition of C_0, no edge exists from X to Y in residual.



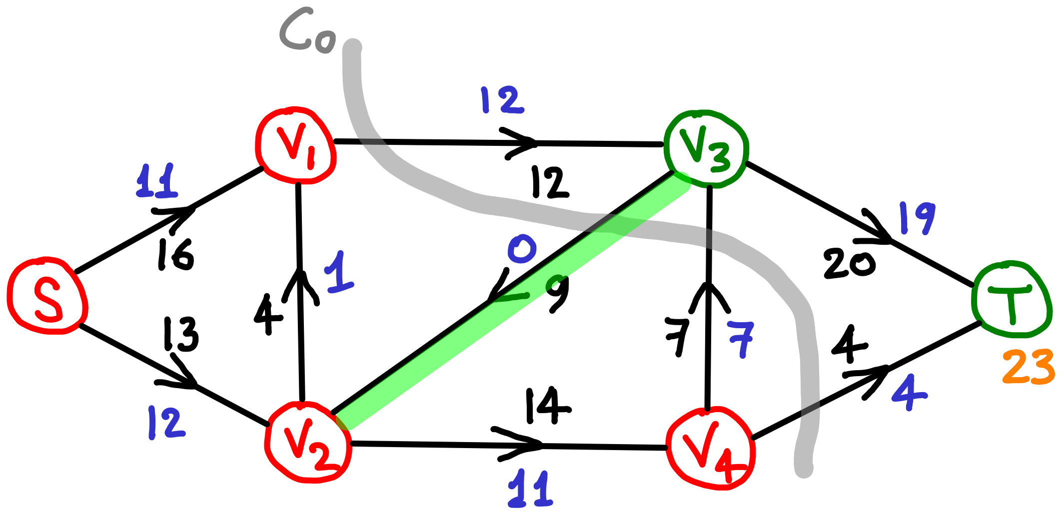
- If $\exists\ \vec{pq}$ from X to Y in flow network then $\text{Flow}(\vec{pq}) = \text{Capacity}(\vec{pq})$

otherwise we get X → Y in residual

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut



By definition of C_0, no edge exists from X to Y in residual.



- If $\exists\ \overrightarrow{pq}$ from X to Y in flow network then Flow($\overrightarrow{pq}$) = Capacity($\overrightarrow{pq}$)

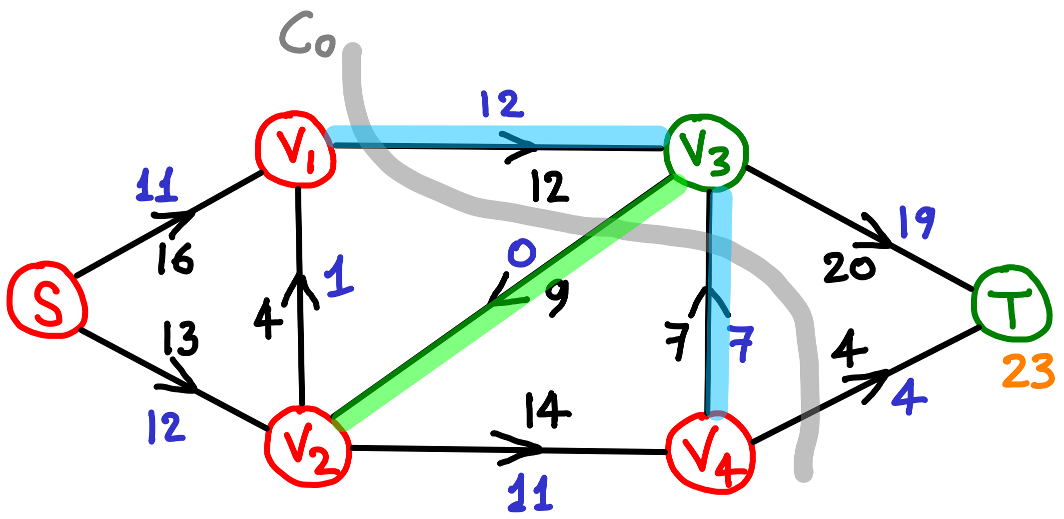
  otherwise we get X $\rightarrow$ Y in residual

- If $\exists\ \overleftarrow{pq}$ from Y to X in flow network then Flow($\overleftarrow{pq}$) = 0

  otherwise we get X $\rightarrow$ Y in residual

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

---



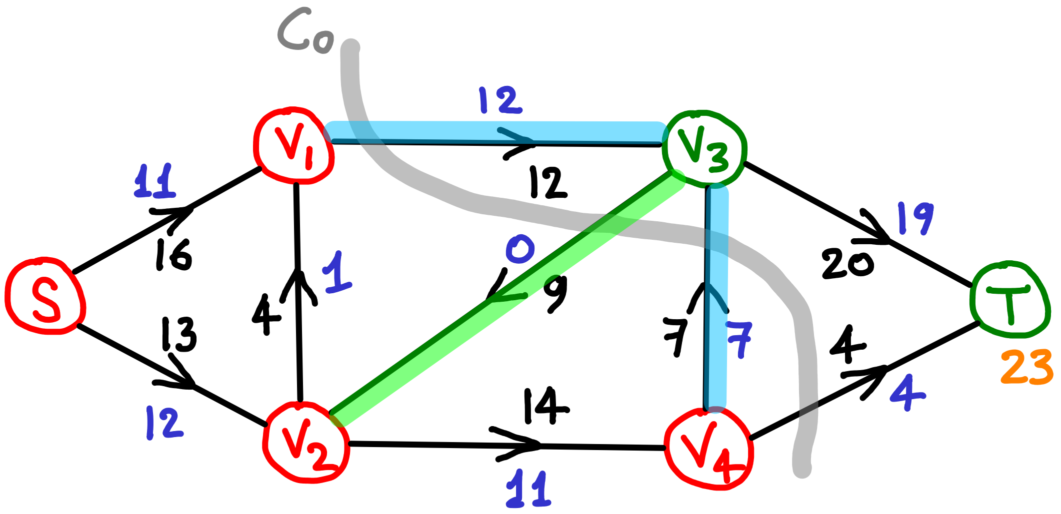
- If $\exists\ \overrightarrow{pq}$ from X to Y in flow network then $\text{Flow}(\overrightarrow{pq}) = \text{Capacity}(\overrightarrow{pq})$

- If $\exists\ \overleftarrow{pq}$ from Y to X in flow network then $\text{Flow}(\overleftarrow{pq}) = 0$

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

---

$$\left.\begin{aligned} Flow(C) &= \sum flow - \sum flow \\ Capacity(C) &= \sum capacity \end{aligned}\right\}$$ fundamental rules



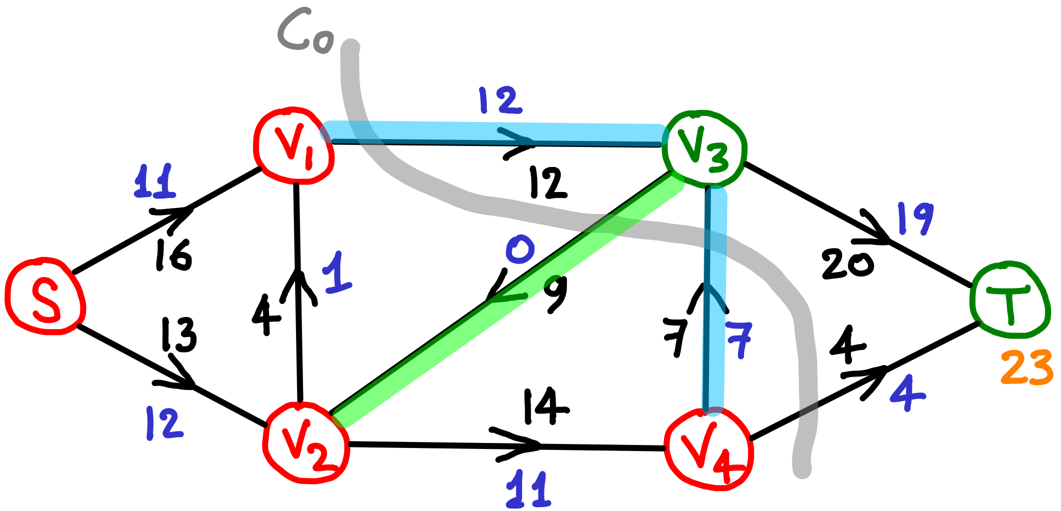
- If $\exists \overrightarrow{pq}$ from X to Y in flow network then $Flow(\overrightarrow{pq}) = Capacity(\overrightarrow{pq})$

- If $\exists \overleftarrow{pq}$ from Y to X in flow network then $Flow(\overleftarrow{pq}) = 0$

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

$$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$$

$$\text{Capacity}(C) = \sum \text{capacity}$$

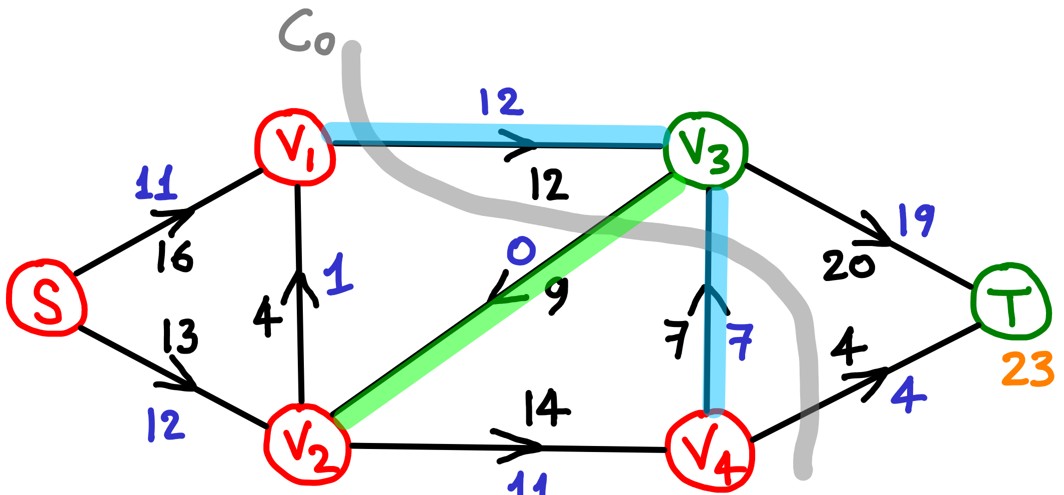


- If $\exists\ \overrightarrow{pq}$ from X to Y in flow network
  then $\text{Flow}(\overrightarrow{pq}) = \text{Capacity}(\overrightarrow{pq})$
  $$\sum_{C_0} \text{flow} = \sum_{C_0} \text{capacity}$$
- If $\exists\ \overleftarrow{pq}$ from Y to X in flow network
  then $\text{Flow}(\overleftarrow{pq}) = 0$
  $$\sum_{C_0} \text{flow} = 0$$

no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

---

$$\left.\begin{aligned} Flow(C) &= \sum flow - \sum flow \\ Capacity(C) &= \sum capacity \end{aligned}\right\} \quad Flow(C_0) = \sum_{C_0} capacity - 0$$



- If $\exists\ \overrightarrow{pq}$ from X to Y in flow network

  then $Flow(\overrightarrow{pq}) = Capacity(\overrightarrow{pq})$

  $$\sum_{C_0} flow = \sum_{C_0} capacity$$

- If $\exists\ \overleftarrow{pq}$ from Y to X in flow network

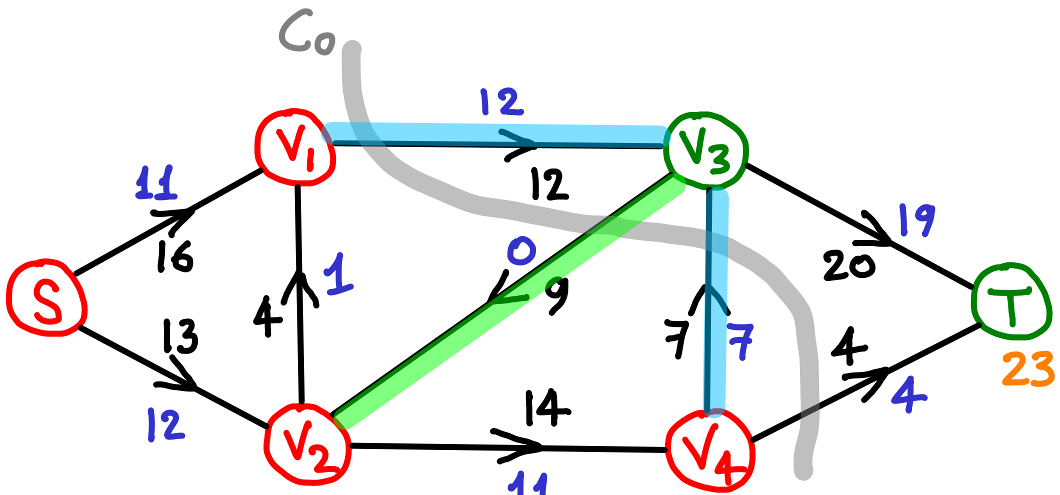
  then $Flow(\overleftarrow{pq}) = 0$

  $$\sum_{C_0} flow = 0$$

no augmenting path $\xrightarrow{\checkmark}$ flow equals capacity of some cut

---

$$\left.\begin{array}{l} \text{Flow}(C) = \sum \text{flow} - \sum \text{flow} \\ \text{Capacity}(C) = \sum \text{capacity} \end{array}\right\} \begin{array}{l} \text{Flow}(C_0) = \sum_{C_0} \text{capacity} - 0 \\ \text{Flow}(C_0) = \text{Capacity}(C_0) \end{array}$$



- If $\exists\ \overrightarrow{pq}$ from X to Y in flow network
  then $\text{Flow}(\overrightarrow{pq}) = \text{Capacity}(\overrightarrow{pq})$
  $$\sum_{C_0} \text{flow} = \sum_{C_0} \text{capacity}$$
- If $\exists\ \overleftarrow{pq}$ from Y to X in flow network
  then $\text{Flow}(\overleftarrow{pq}) = 0$
  $$\sum_{C_0} \text{flow} = 0$$

FORD - FULKERSON method

- start w/ zero flow
- while augmenting path exists

  increase flow on an augmenting path $\left\{ \text{increases flow value} \right.$

FORD - FULKERSON method

- start w/ zero flow
- while augmenting path exists

  increase flow on an augmenting path — increases flow value

  - find min-weight w on aug. path
  - for every edge pq on aug. path
    - if $\overrightarrow{pq}$ in flow network
      add w to Flow($\overrightarrow{pq}$)
      else subtract w from Flow($\overrightarrow{pq}$)

FORD - FULKERSON method

- start w/ zero flow
- while augmenting path exists

  how do we find one?

  increase flow on an augmenting path

  increases flow value

  - find min-weight w on aug. path
  - for every edge pq on aug. path
    - if $\overrightarrow{pq}$ in flow network
      add w to Flow($\overrightarrow{pq}$)
      else subtract w from Flow($\overleftarrow{pq}$)

  O(path length)

FORD - FULKERSON method

- start w/ zero flow
- while augmenting path exists

how many times?

how do we find one?

increase flow on an augmenting path

increases flow value

- find min-weight w on aug. path
- for every edge pq on aug. path
  - if $\vec{pq}$ in flow network
    add w to $\mathrm{Flow}(\vec{pq})$
    else subtract w from $\mathrm{Flow}(\vec{pq})$

$O(\text{path length})$

FORD - FULKERSON method

- start w/ zero flow
- while augmenting path exists

how many times?

how do we find one?

increase flow on an augmenting path

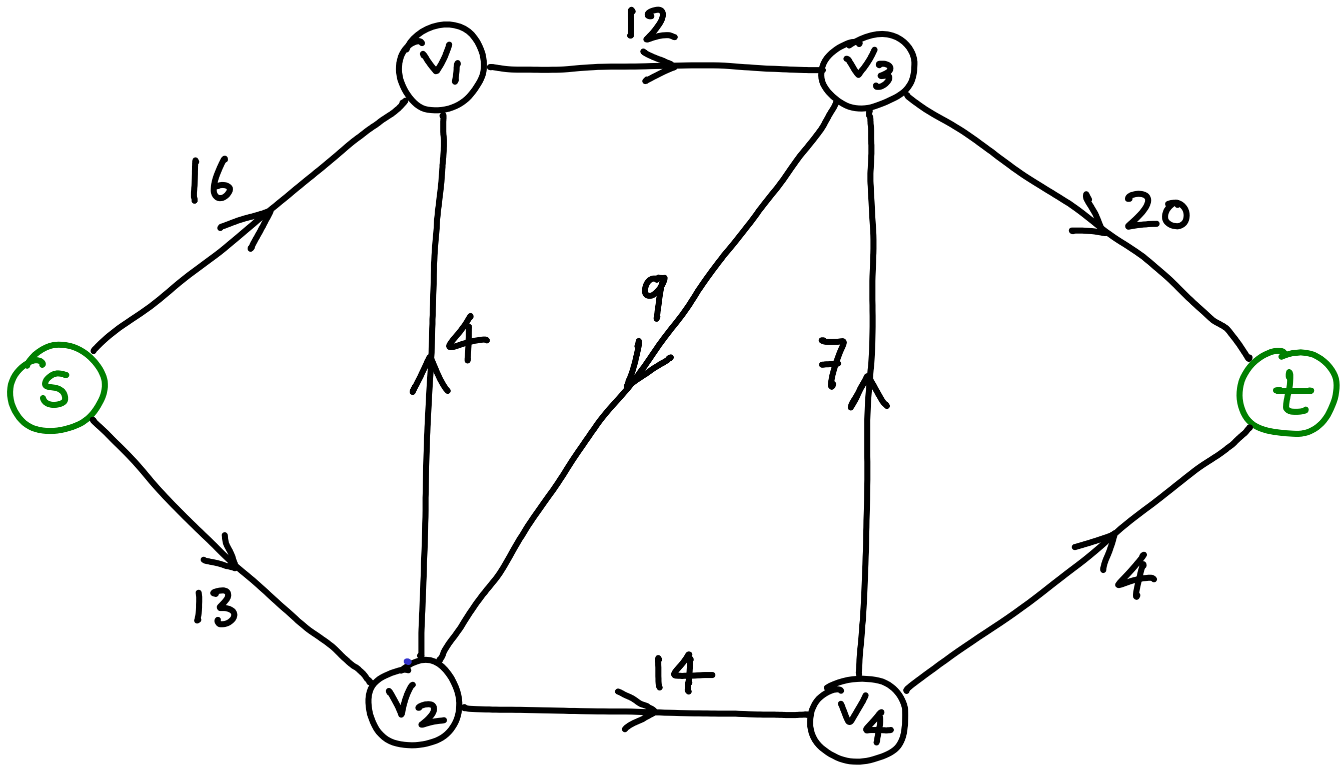
increases flow value

- find min-weight w on aug. path
- for every edge pq on aug. path
  - if $\overrightarrow{pq}$ in flow network
    add w to $Flow(\overrightarrow{pq})$
    else subtract w from $Flow(\overleftarrow{pq})$

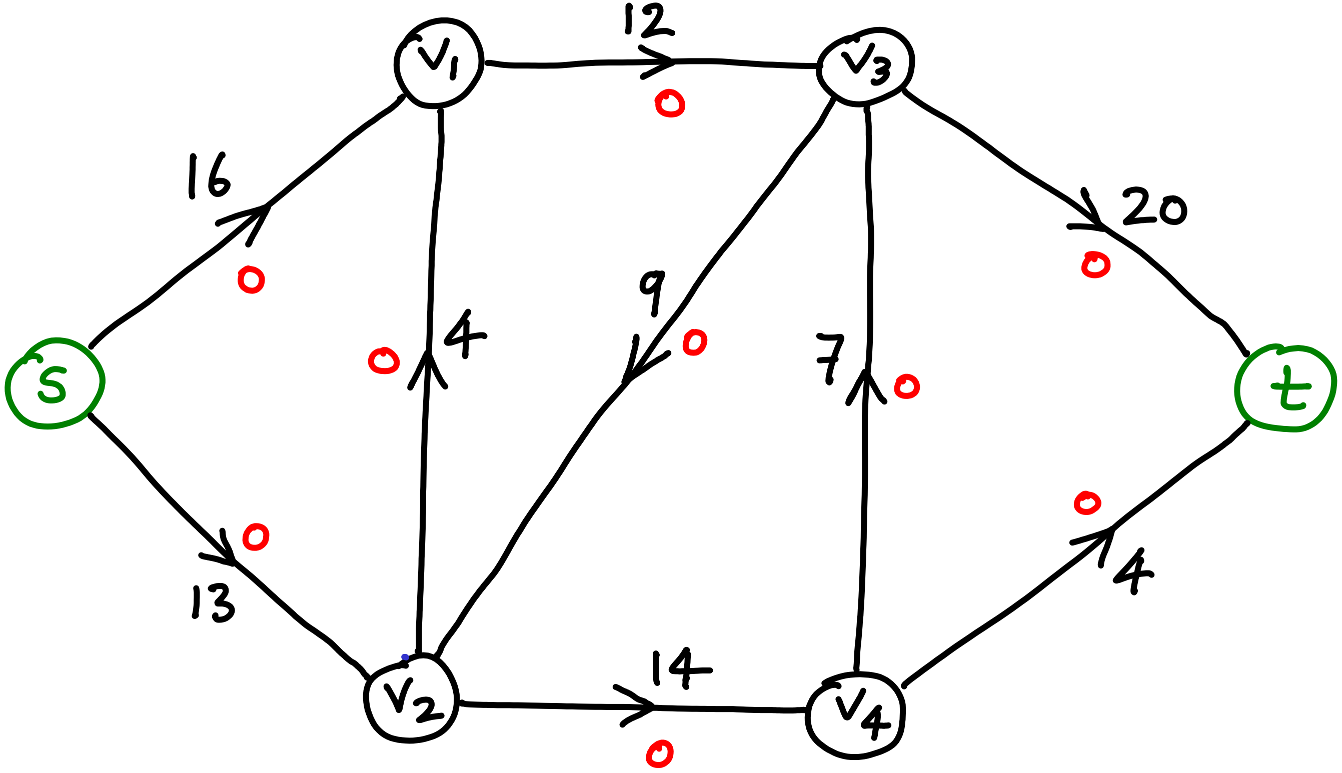
O(path length)

forms basis of several algorithms (e.g. Edmonds-Karp)

: capacity (normal edge)



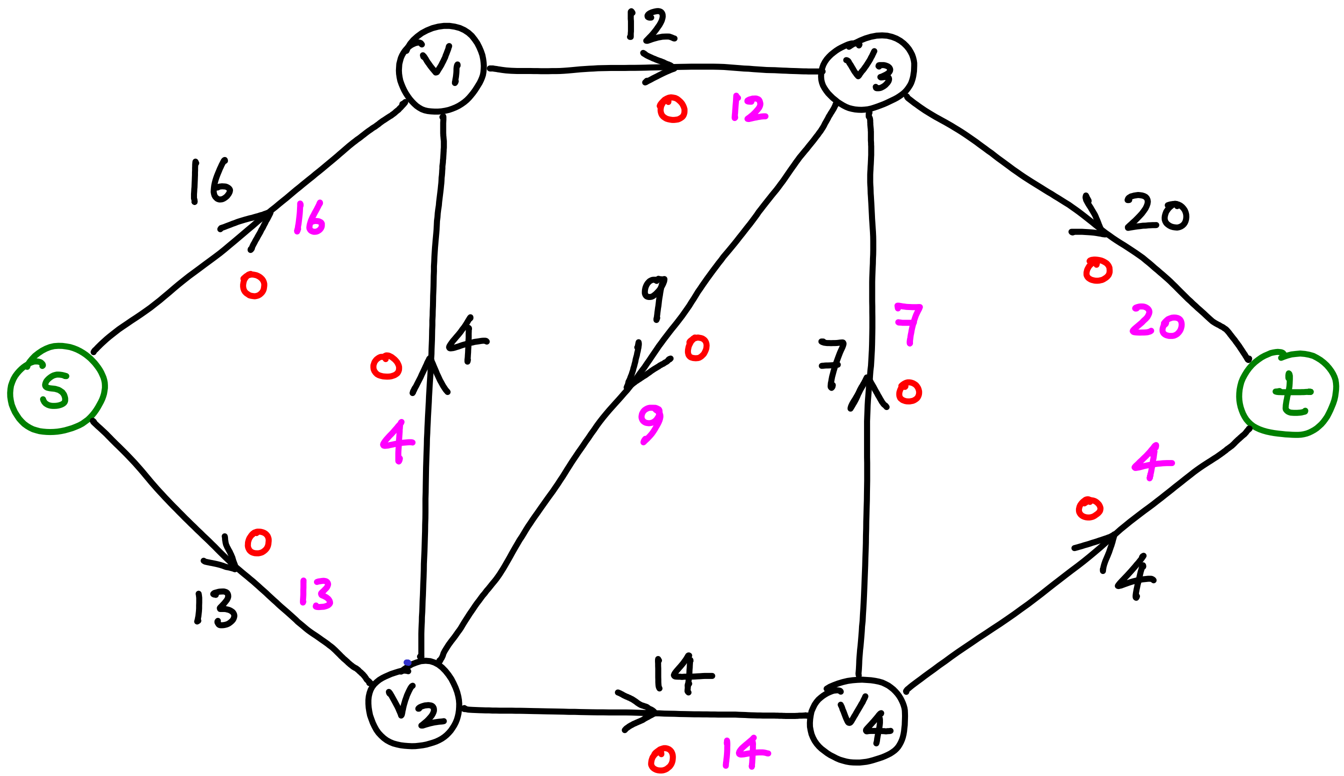
: capacity (normal edge)
: flow



#: capacity (normal edge)

#: flow

#: leftover (residual edge)

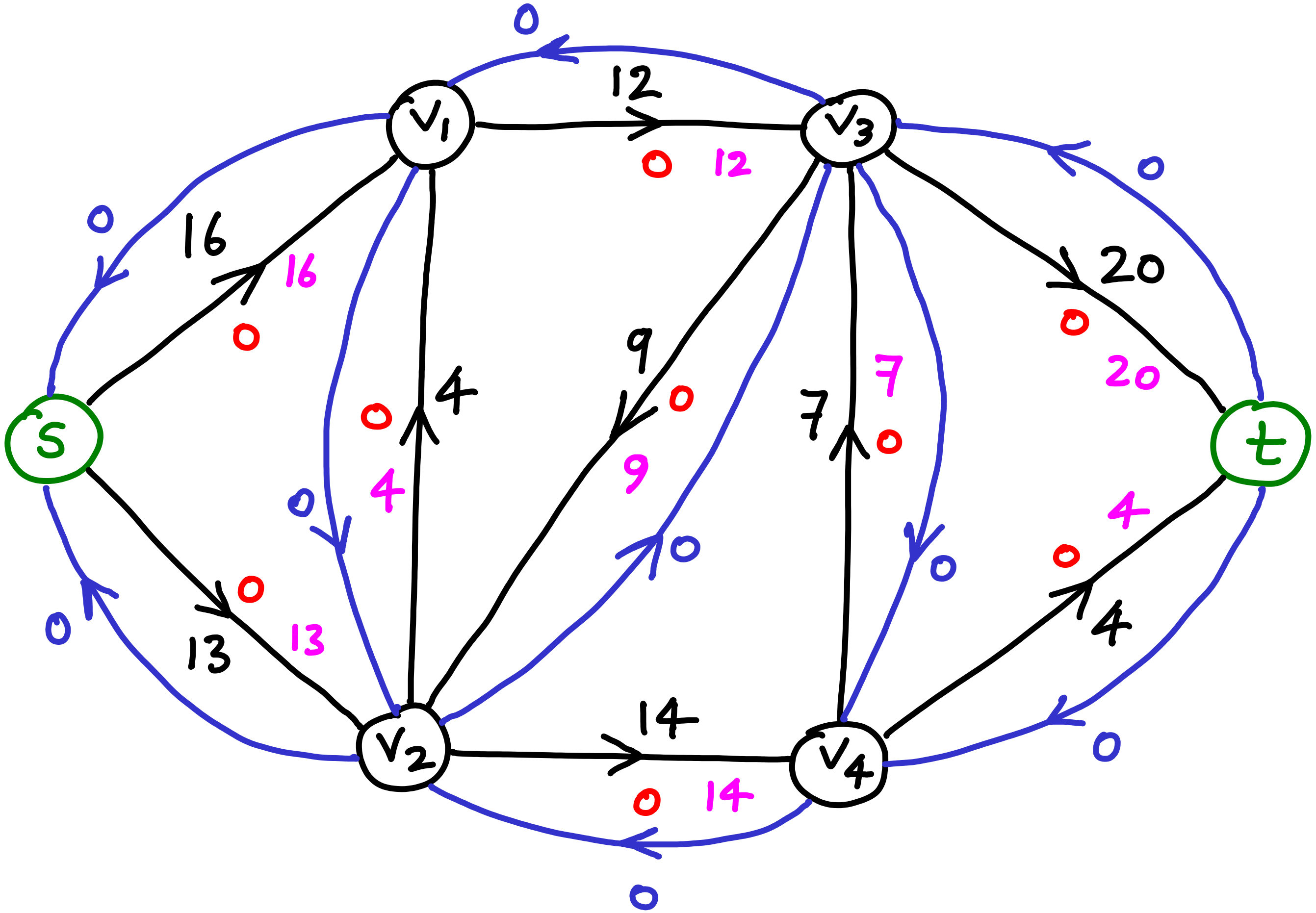


#: capacity (normal edge)

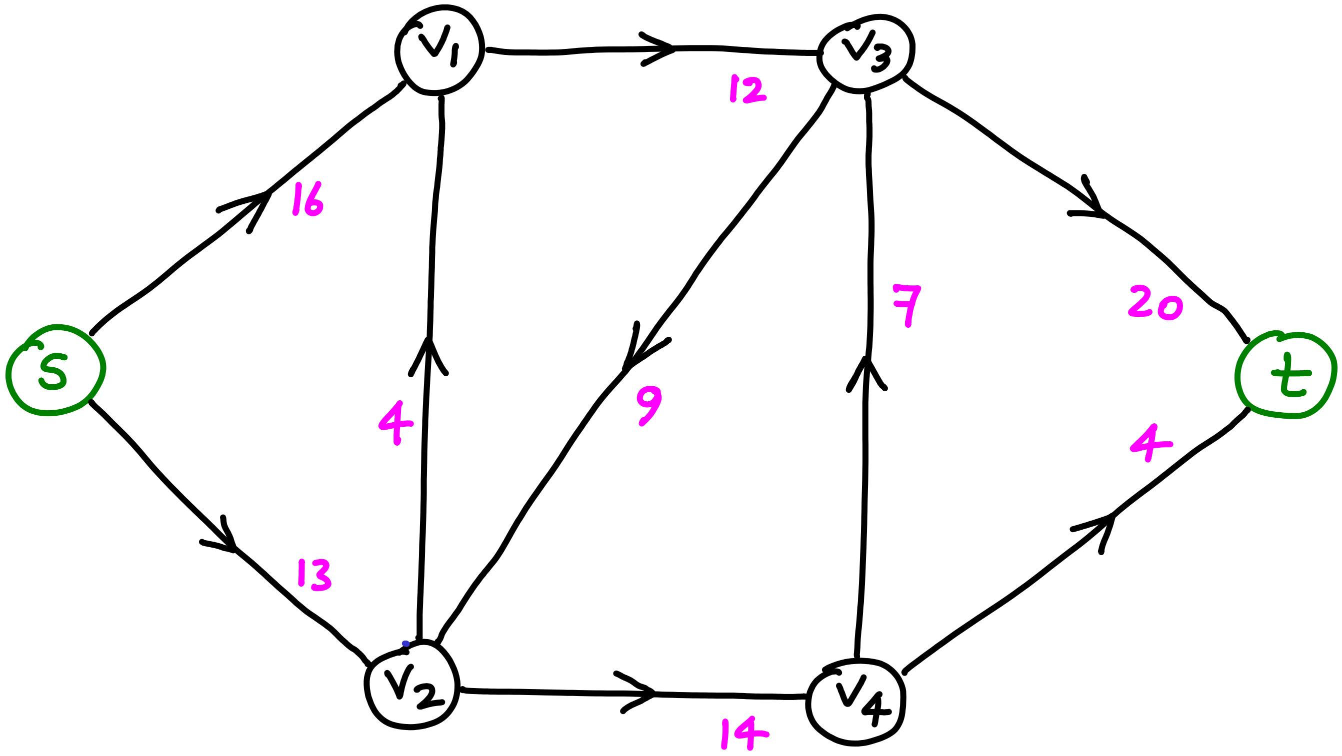
#: flow

#: leftover (residual edge)

#: reverse (residual edge)



: capacity (normal edge)
: flow
: leftover (residual edge)
: reverse (residual edge)



Residual graph

\# : capacity (normal edge)

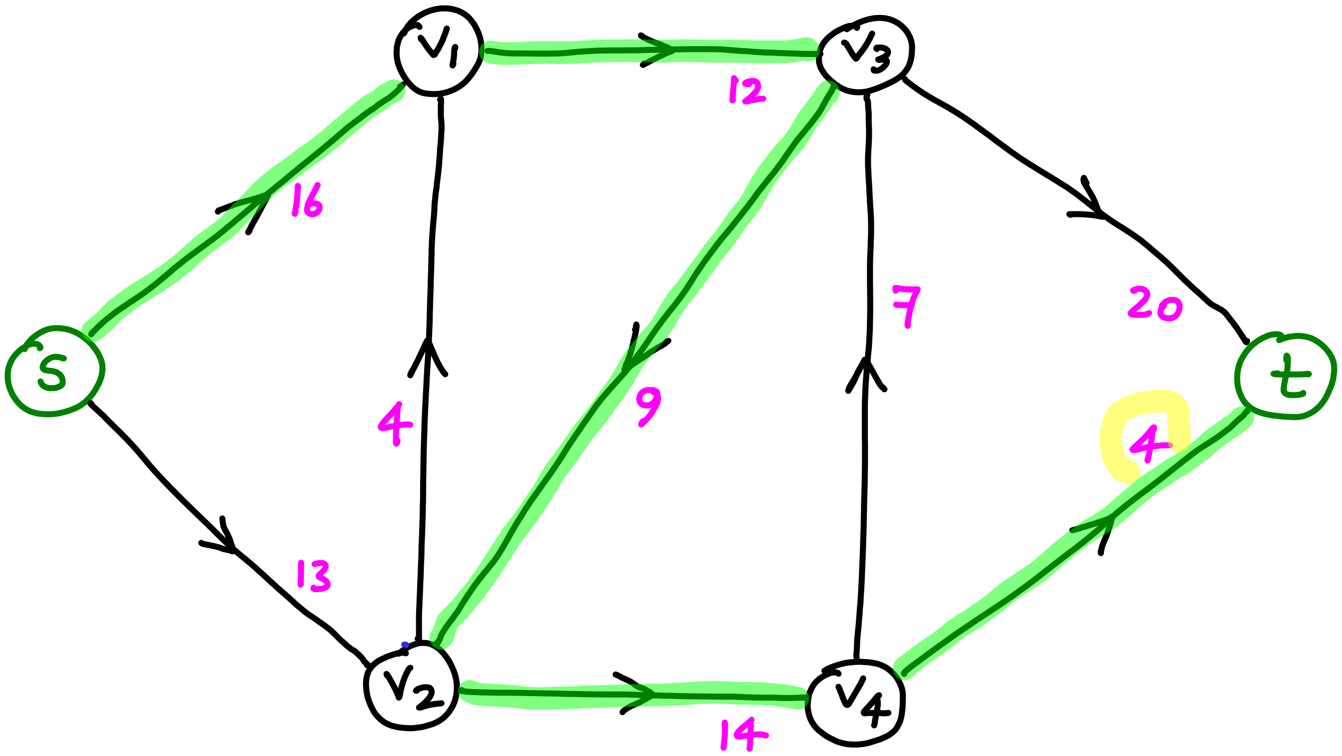
\# : flow

\# : leftover (residual edge)

\# : reverse (residual edge)

⤳ : augmenting path

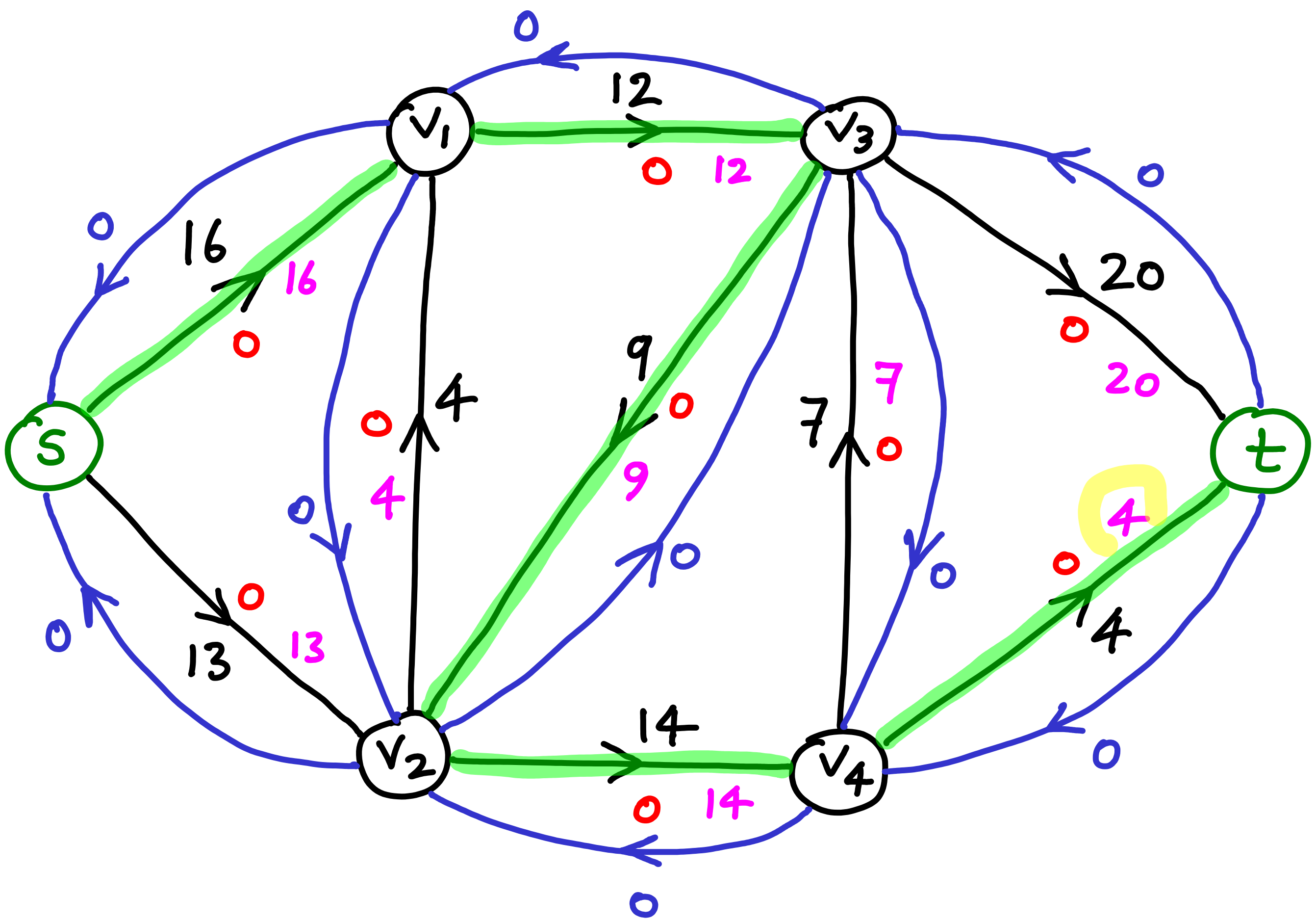
$$s \xrightarrow[16]{} v_1 \xrightarrow[12]{} v_3 \xrightarrow[9]{} v_2 \xrightarrow[14]{} v_4 \xrightarrow[4]{} t$$



: capacity (normal edge)
: flow
: leftover (residual edge)
: reverse (residual edge)
↝ : augmenting path

$s \rightarrow v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow t$

16 12 9 14 4



#: capacity (normal edge)

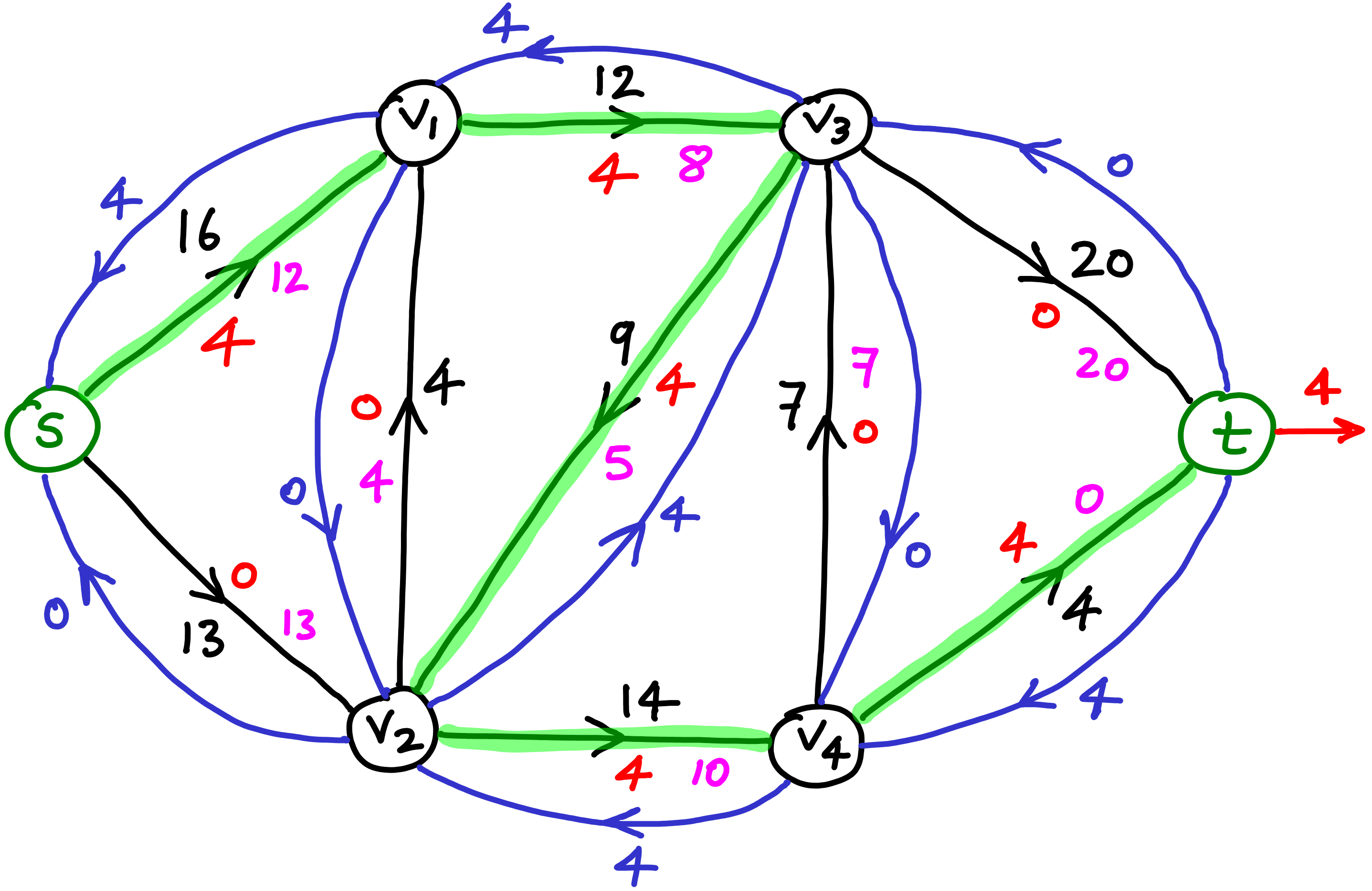
#: flow

#: leftover (residual edge)

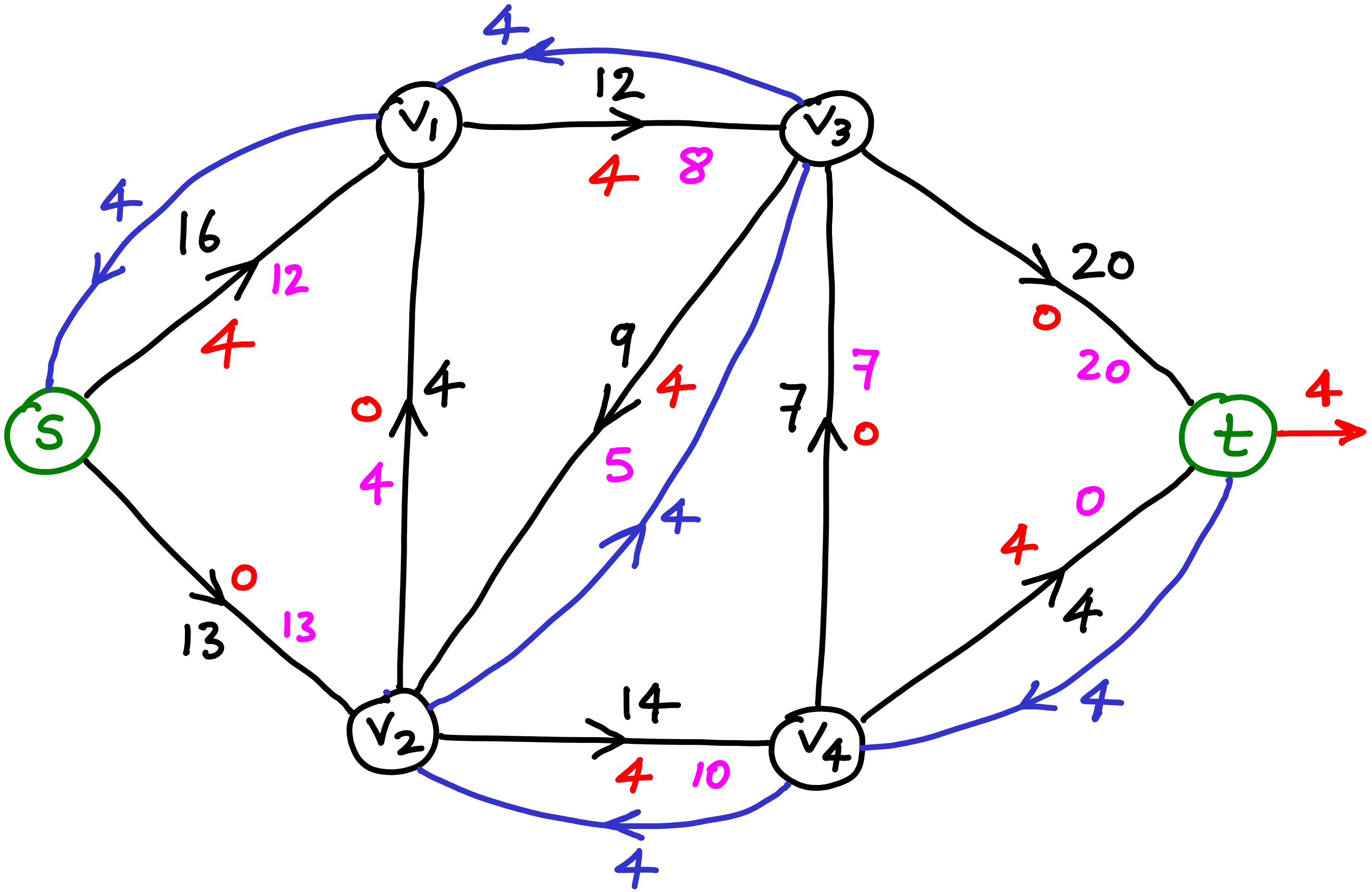
#: reverse (residual edge)

↝: augmenting path

$s \xrightarrow{16} v_1 \xrightarrow{12} v_3 \xrightarrow{9} v_2 \xrightarrow{14} v_4 \xrightarrow{4} t$



#: capacity (normal edge)
#: flow
#: leftover (residual edge)
#: reverse (residual edge)

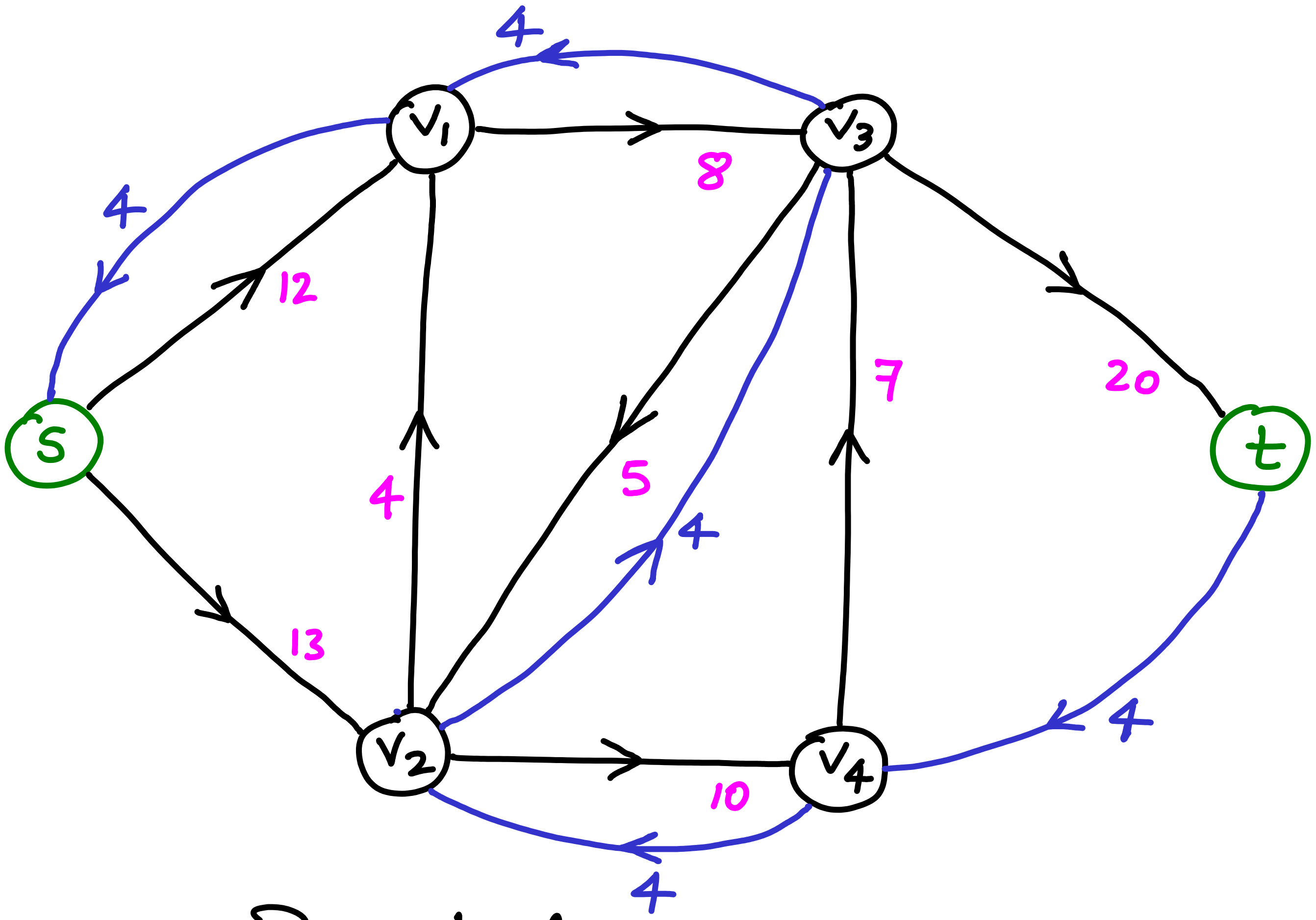


#: capacity (normal edge)

#: flow

#: leftover (residual edge)

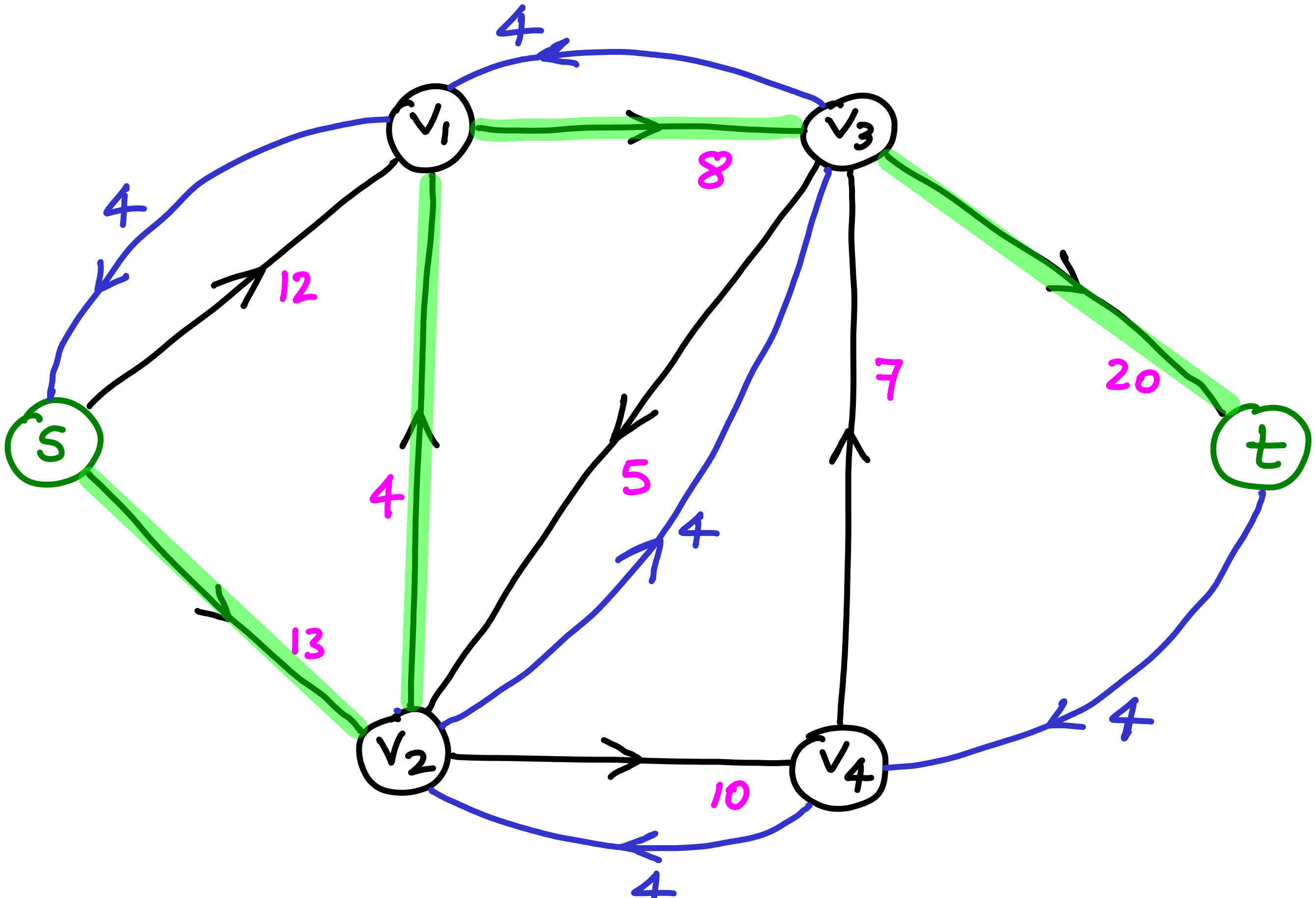
#: reverse (residual edge)



Residual graph

: capacity (normal edge)
: flow
: leftover (residual edge)
: reverse (residual edge)
⇝ : augmenting path

$$\underset{13}{s \to} \underset{4}{v_2 \to} \underset{8}{v_1 \to} \underset{20}{v_3 \to} t$$



#: capacity (normal edge)

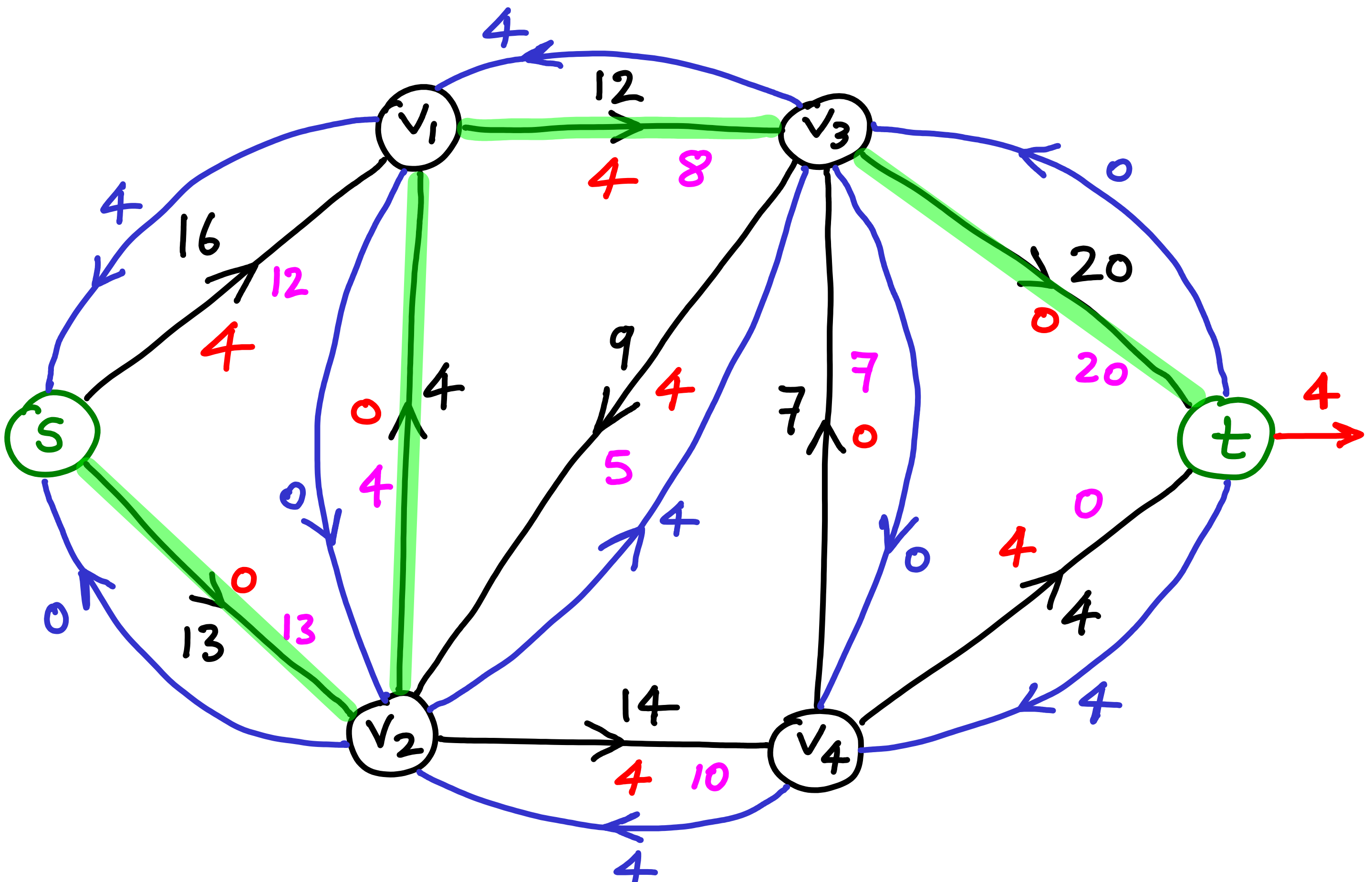
#: flow

#: leftover (residual edge)

#: reverse (residual edge)

↝: augmenting path

$$\underset{}{s} \xrightarrow{} \underset{13}{v_2} \xrightarrow{} \underset{\textcircled{4}}{v_1} \xrightarrow{} \underset{8}{v_3} \xrightarrow{} \underset{20}{t}$$



#: capacity (normal edge)

#: flow

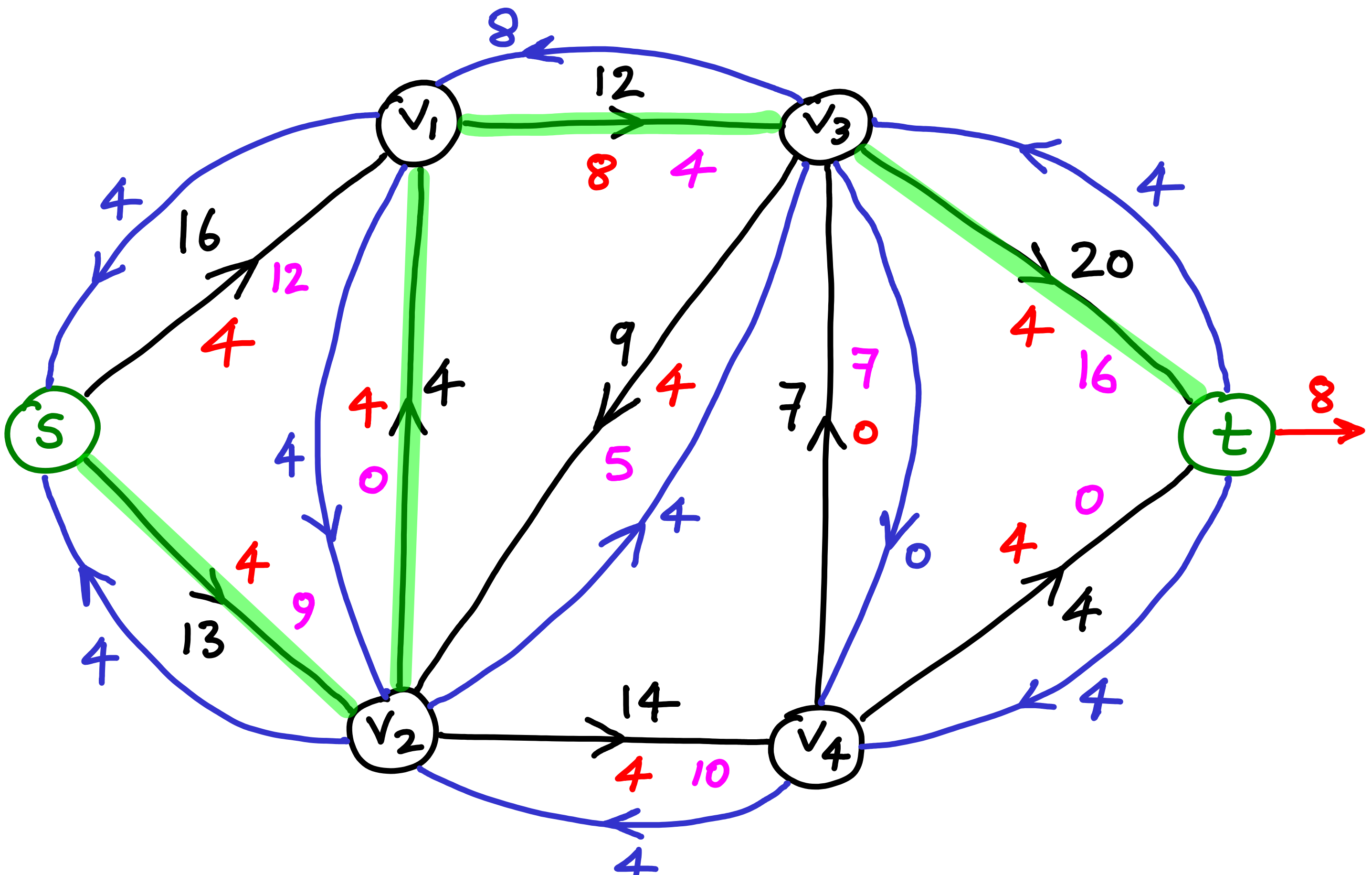
#: leftover (residual edge)

#: reverse (residual edge)

↝: augmenting path

$s \to v_2 \to v_1 \to v_3 \to t$

13 4 8 20

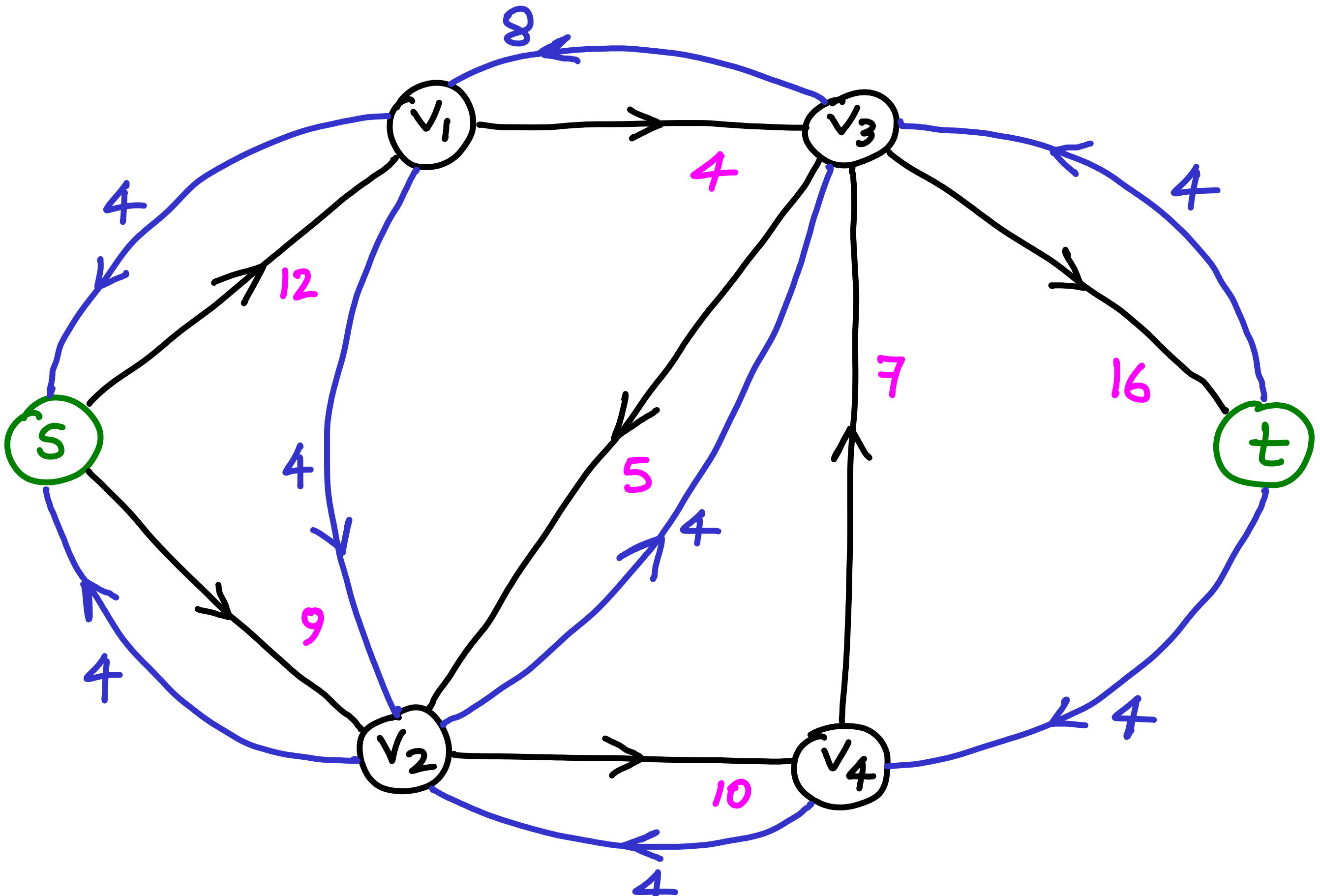


#: capacity (normal edge)

#: flow

#: leftover (residual edge)

#: reverse (residual edge)



Residual graph

#: capacity (normal edge)

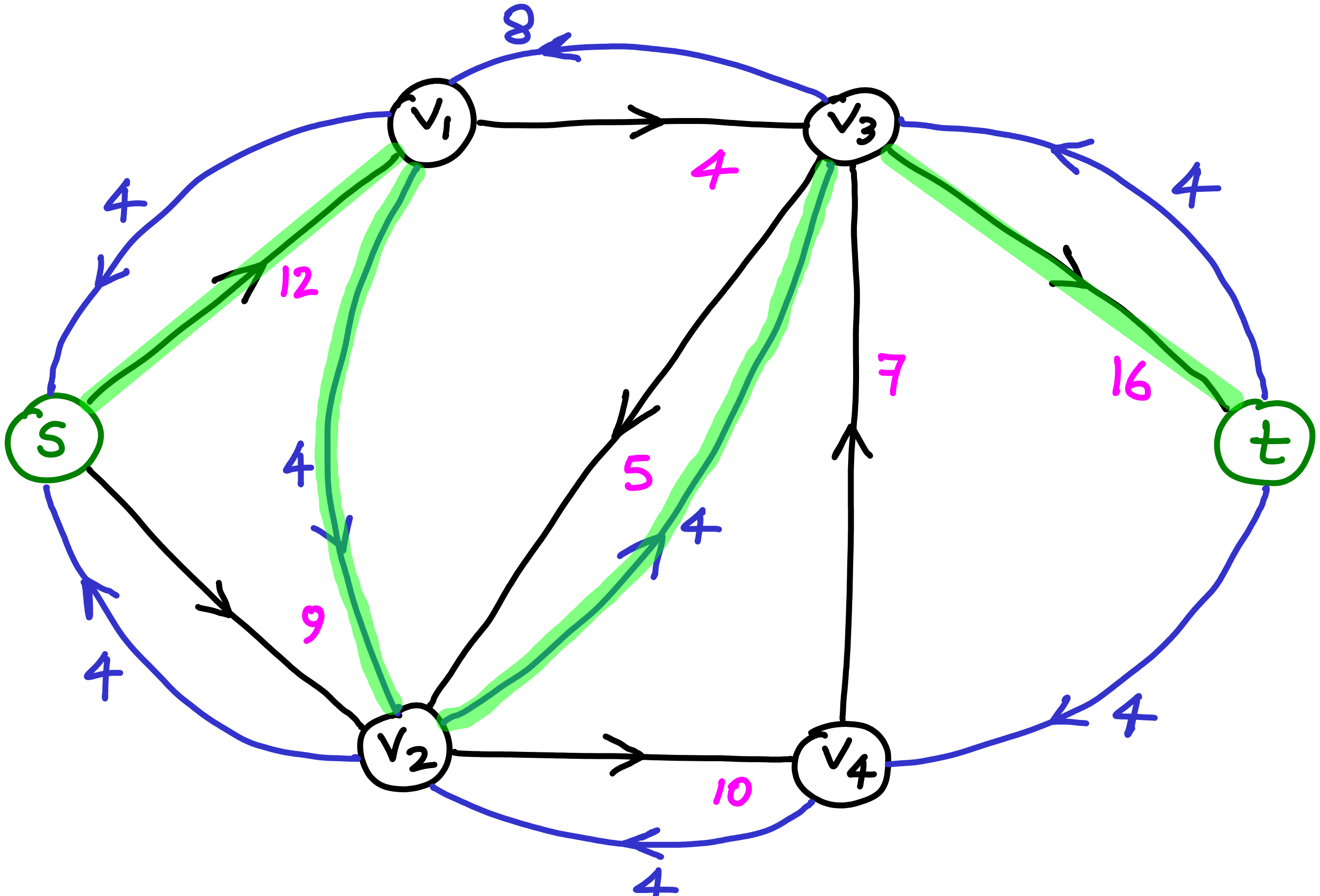
#: flow

#: leftover (residual edge)

#: reverse (residual edge)

⤳ : augmenting path

$s \xrightarrow[12]{} v_1 \xrightarrow[4]{} v_2 \xrightarrow[4]{} v_3 \xrightarrow[16]{} t$



#: capacity (normal edge)

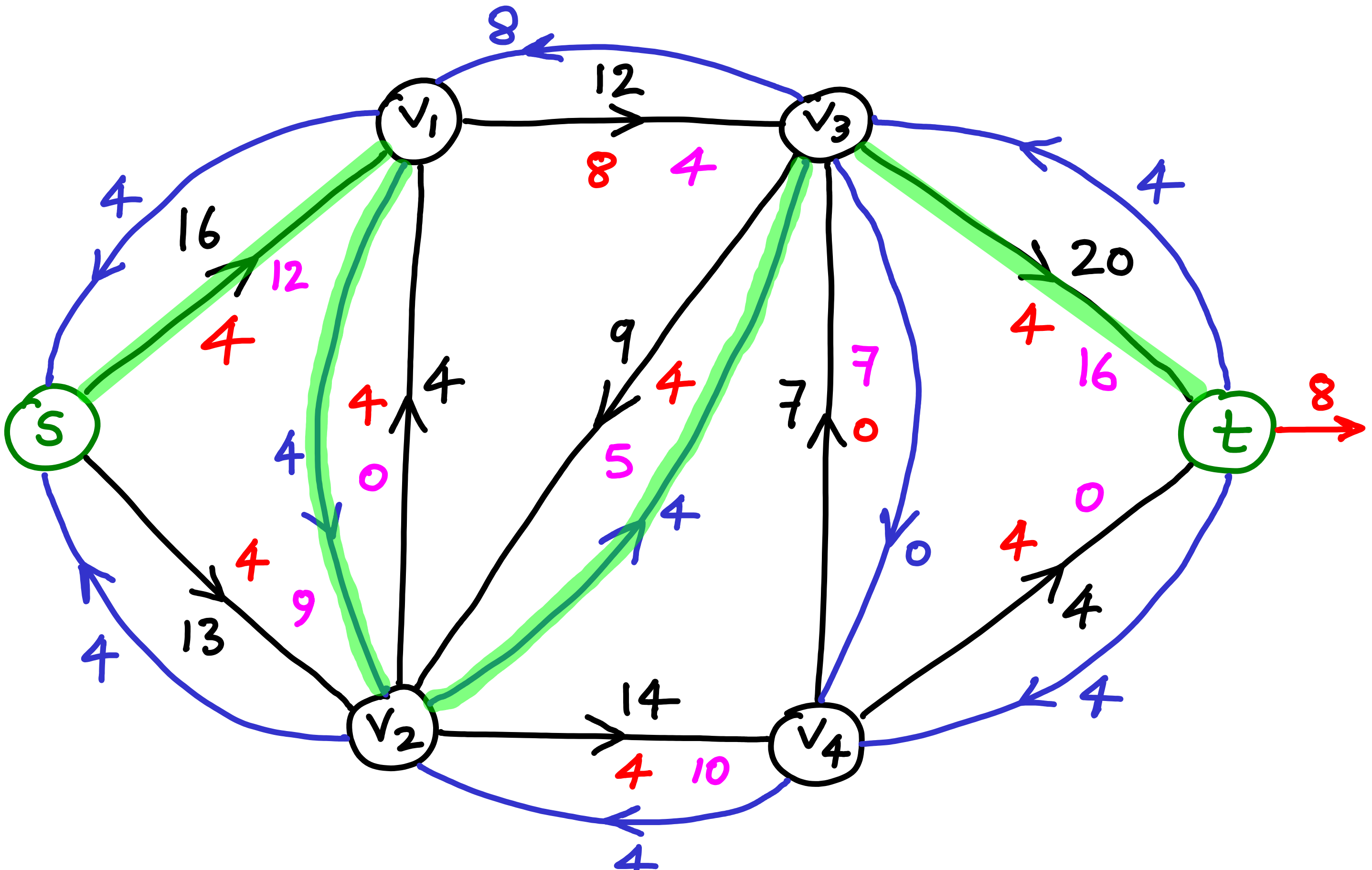
#: flow

#: leftover (residual edge)

#: reverse (residual edge)

↝: augmenting path

$s \xrightarrow[12]{} v_1 \xrightarrow[4]{} v_2 \xrightarrow[4]{} v_3 \xrightarrow[16]{} t$



: capacity (normal edge)

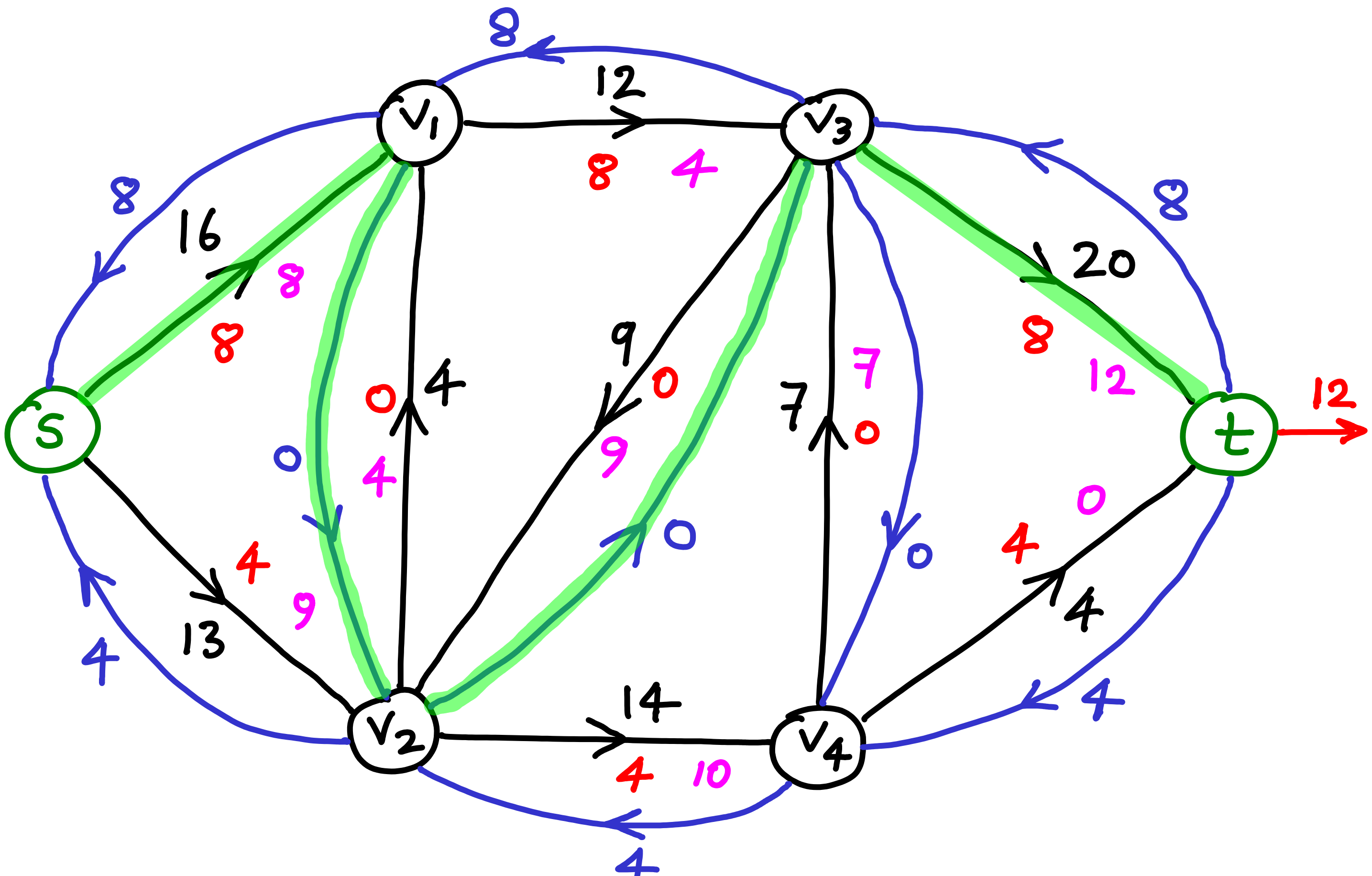
: flow

: leftover (residual edge)

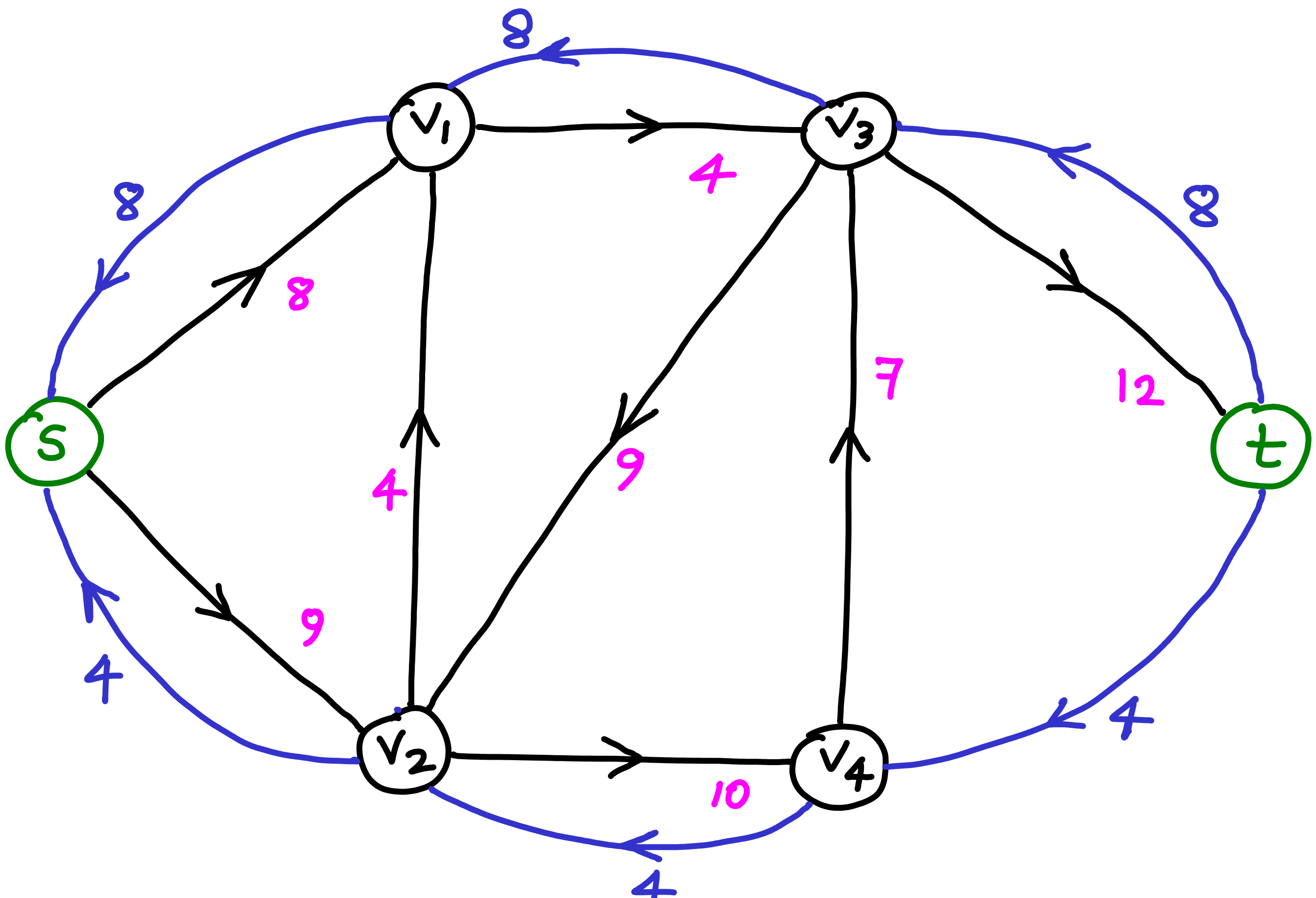
: reverse (residual edge)

⤳ : augmenting path

$s \xrightarrow[12]{} v_1 \xrightarrow[4]{} v_2 \xrightarrow[4]{} v_3 \xrightarrow[16]{} t$



: capacity (normal edge)
: flow
: leftover (residual edge)
: reverse (residual edge)



Residual graph

: capacity (normal edge)

: flow

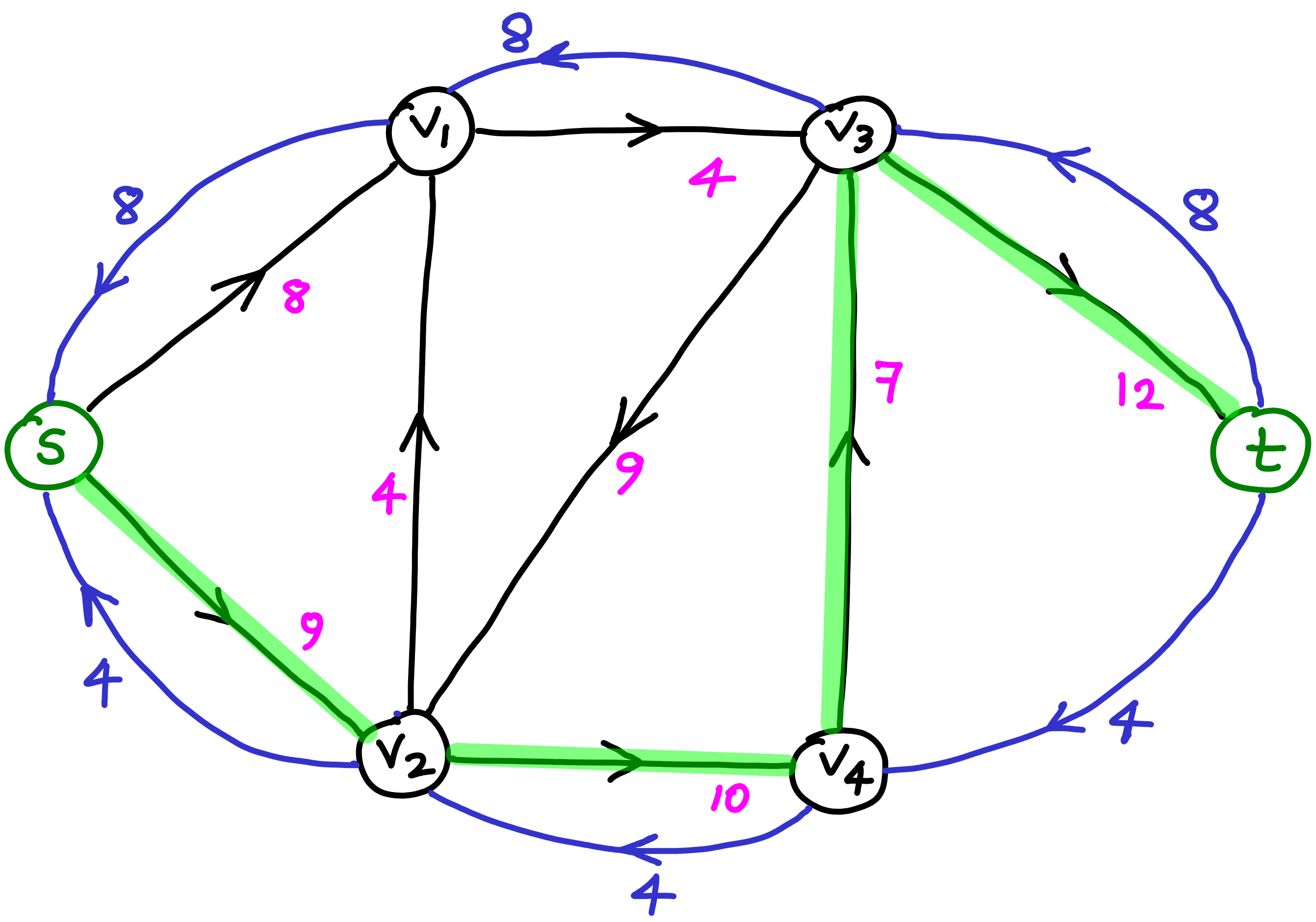
: leftover (residual edge)

: reverse (residual edge)

↝ : augmenting path

$s \to v_2 \to v_4 \to v_3 \to t$

9 10 7 12



\#: capacity (normal edge)

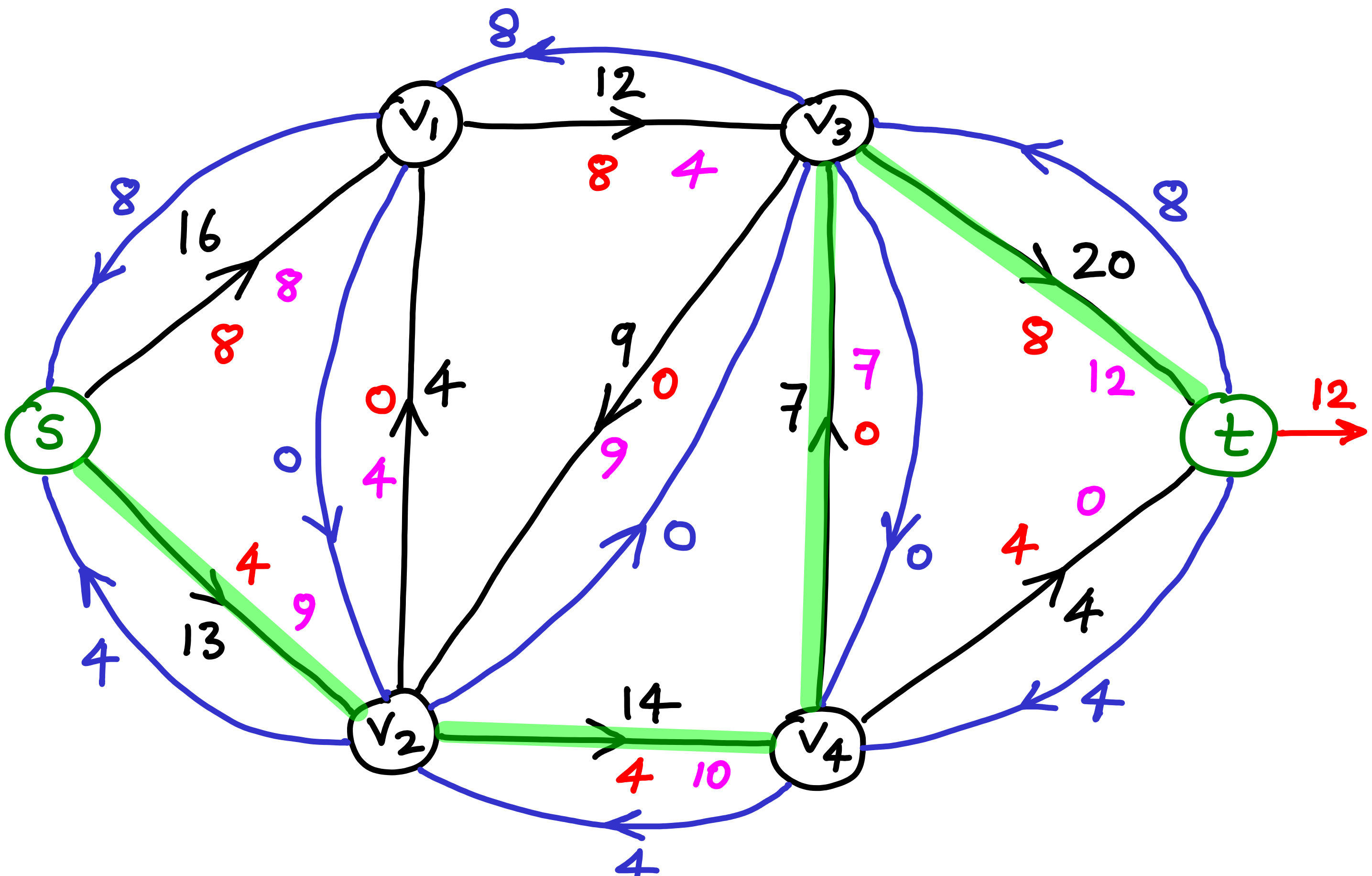
\#: flow

\#: leftover (residual edge)

\#: reverse (residual edge)

⤳: augmenting path

$$\underset{}{s} \rightarrow \underset{9}{} v_2 \rightarrow \underset{10}{} v_4 \rightarrow \underset{7}{} v_3 \rightarrow \underset{12}{} t$$



#: capacity (normal edge)

#: flow

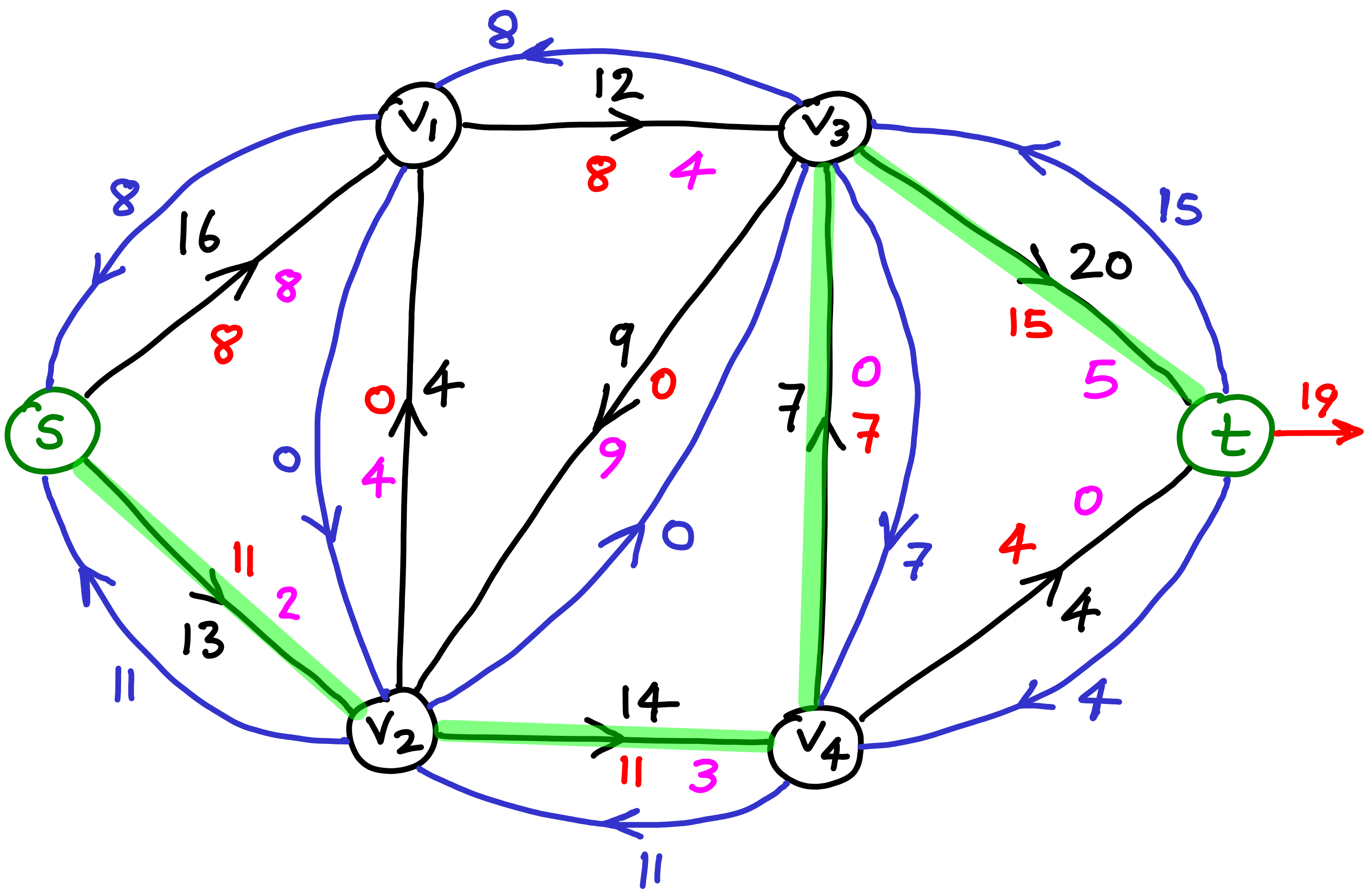
#: leftover (residual edge)

#: reverse (residual edge)

↝: augmenting path

$s \to v_2 \to v_4 \to v_3 \to t$

9 10 7 12

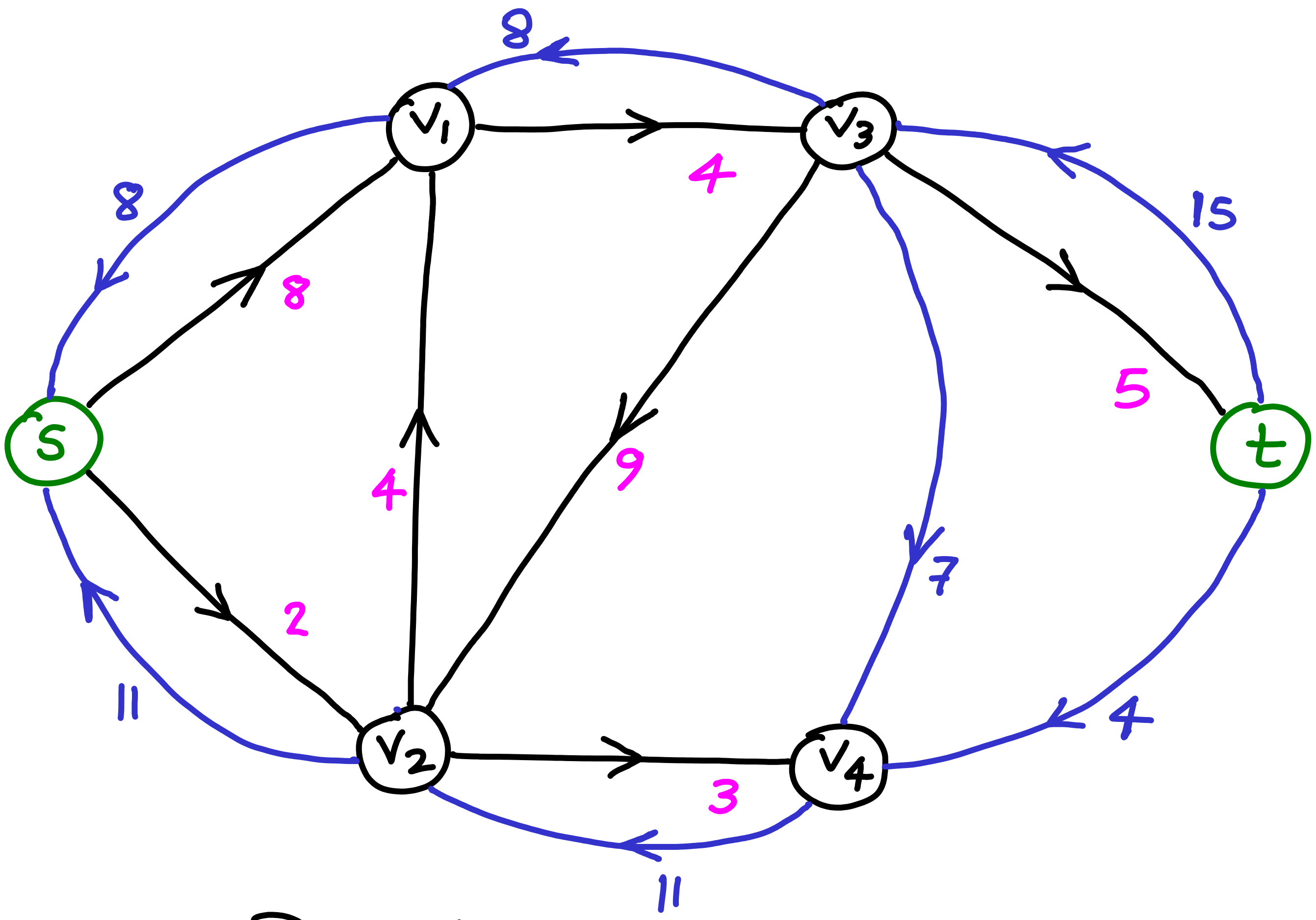


: capacity (normal edge)

: flow

: leftover (residual edge)

: reverse (residual edge)



Residual graph

#: capacity (normal edge)

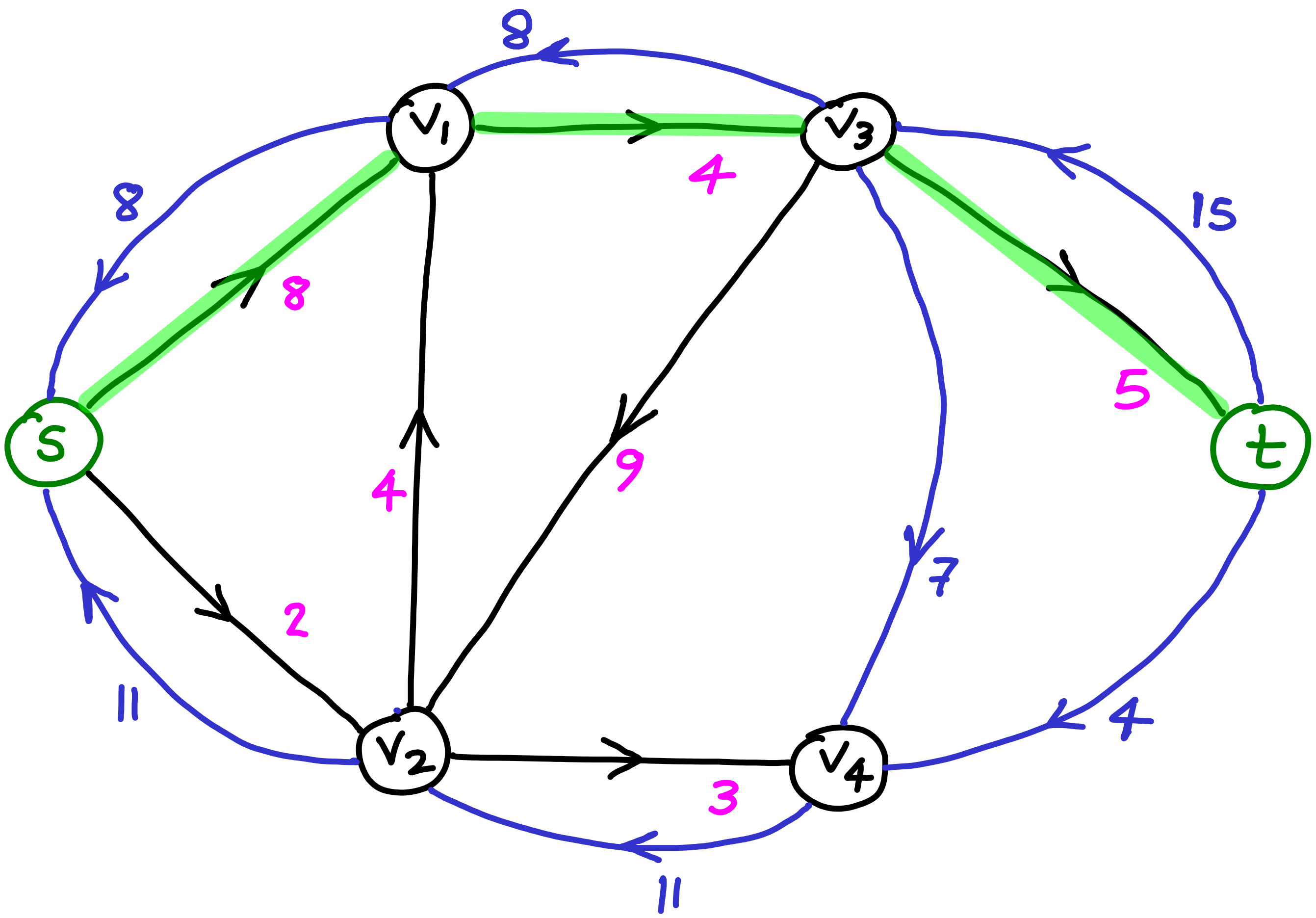
#: flow

#: leftover (residual edge)

#: reverse (residual edge)

↝: augmenting path

$$\underset{8}{s \rightarrow v_1} \underset{4}{\rightarrow v_3} \underset{5}{\rightarrow t}$$



#: capacity (normal edge)

#: flow

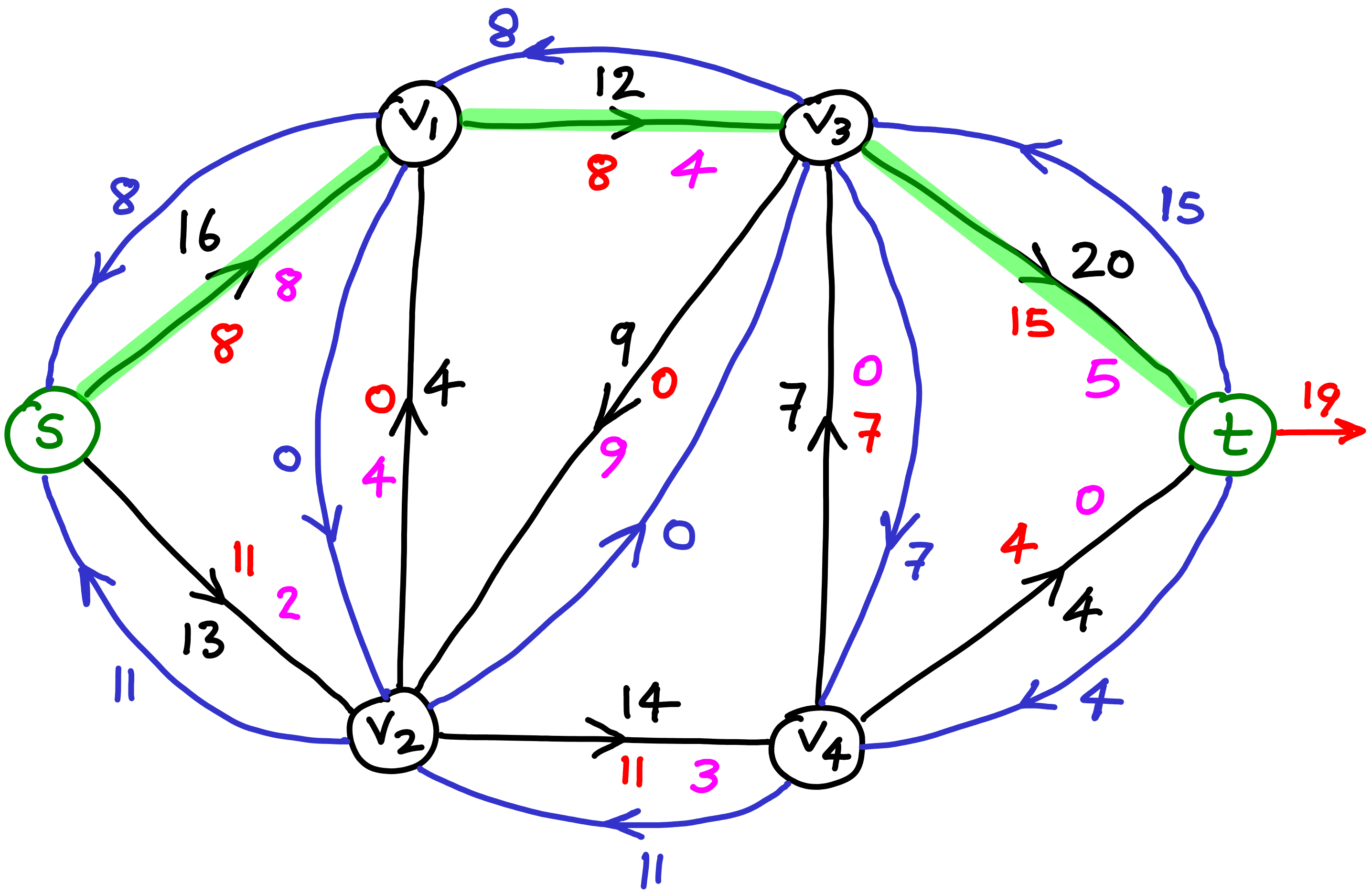
#: leftover (residual edge)

#: reverse (residual edge)

↝: augmenting path

$s \to v_1 \to v_3 \to t$

8 4 5



#: capacity (normal edge)

#: flow

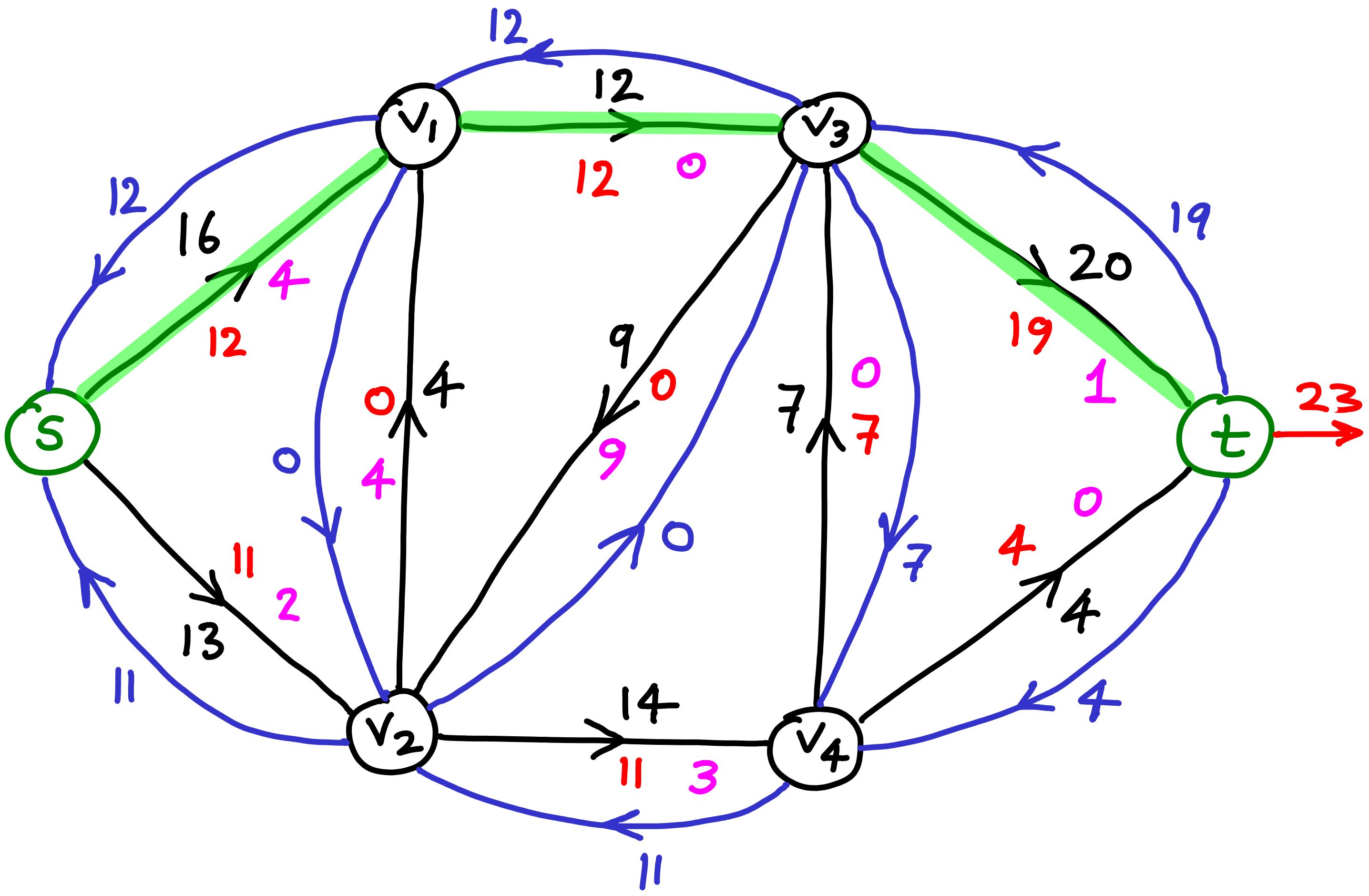
#: leftover (residual edge)

#: reverse (residual edge)

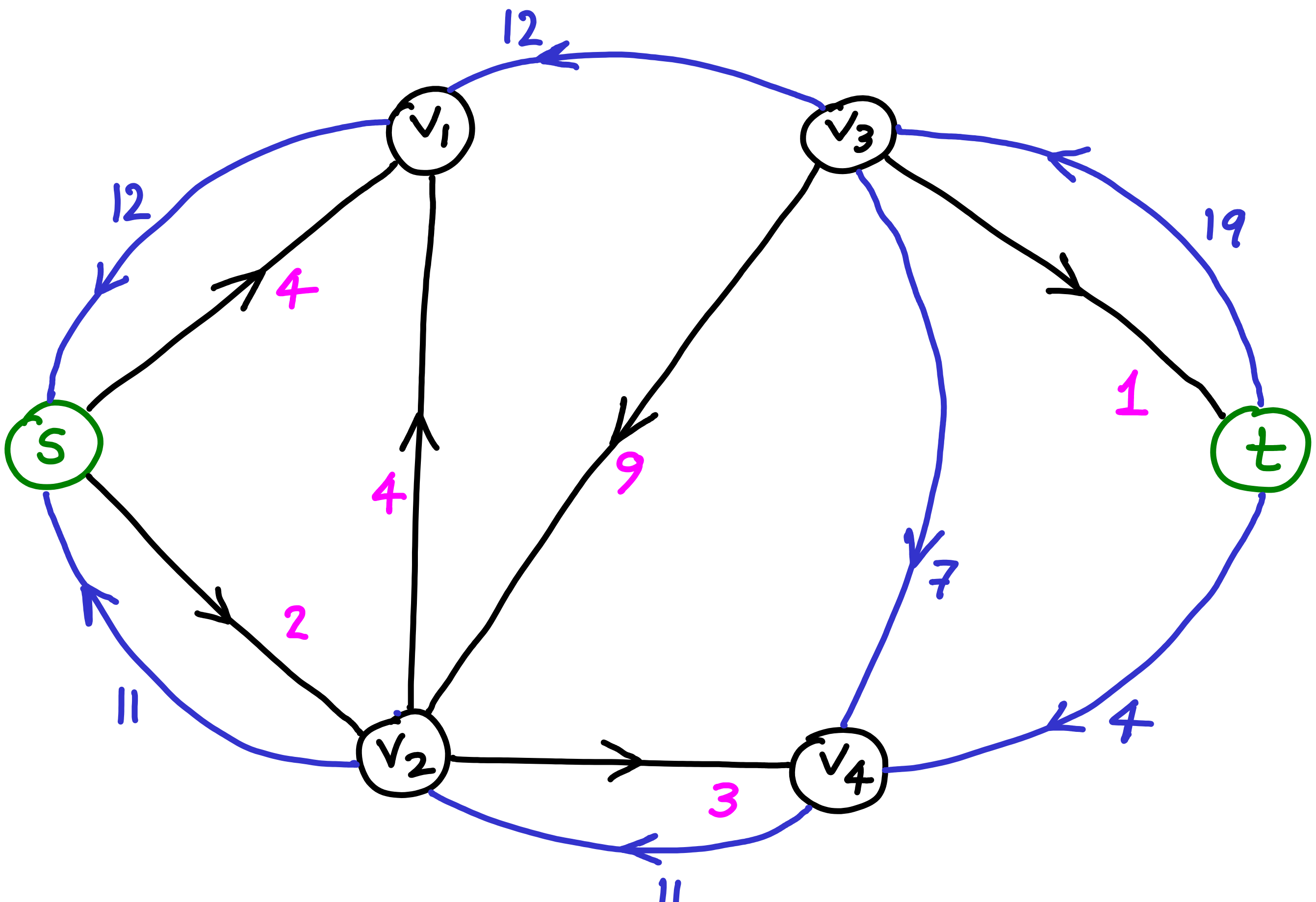
↝: augmenting path

$s \rightarrow v_1 \rightarrow v_3 \rightarrow t$

8 4 5



: capacity (normal edge)
: flow
: leftover (residual edge)
: reverse (residual edge)

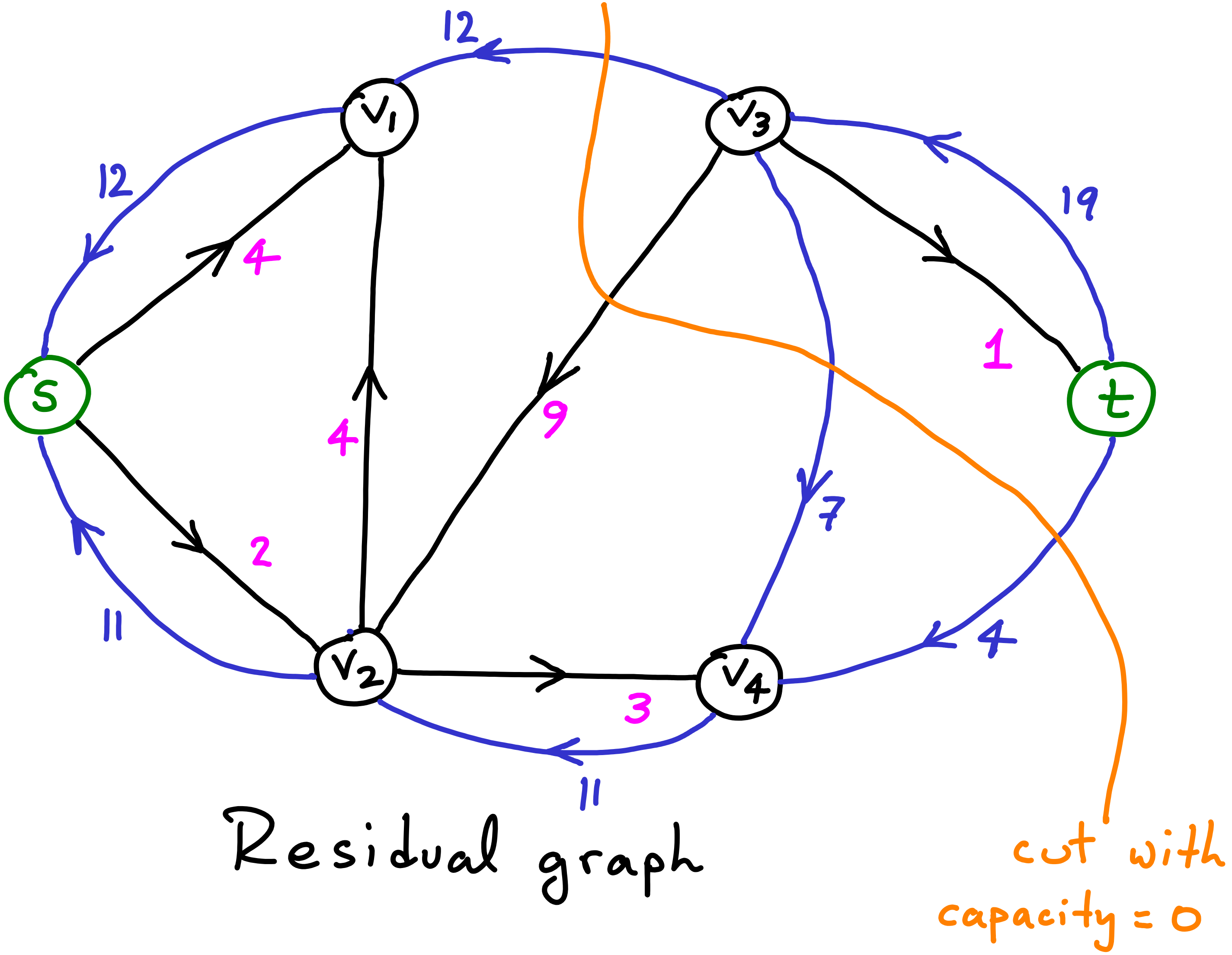


Residual graph

#: capacity (normal edge)
#: flow
#: leftover (residual edge)
#: reverse (residual edge)

No augmenting path.

DONE



Residual graph

FORD - FULKERSON method

how many times ?

how do we find one ?

- while augmenting path exists
  - increase flow on an augmenting path

increases flow value.

O(path length)

Arbitrarily?

FORD - FULKERSON method

- while augmenting path exists
  increase flow on an augmenting path

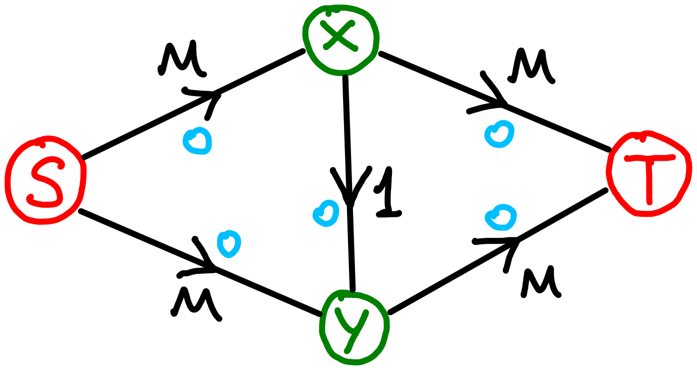
how many times?

how do we find one?

increases flow value.

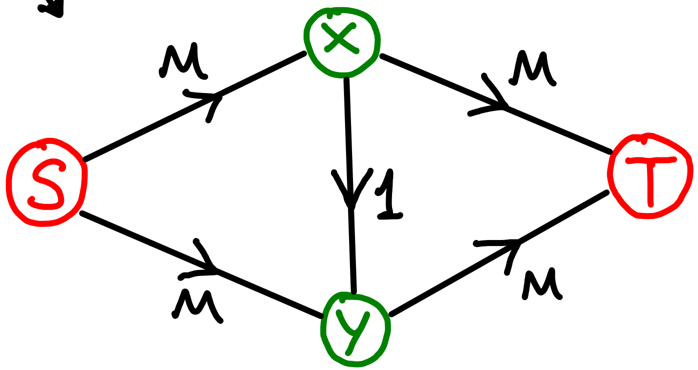
O(path length)

Arbitrarily?



$M = 10^6$

Residual



FORD - FULKERSON method

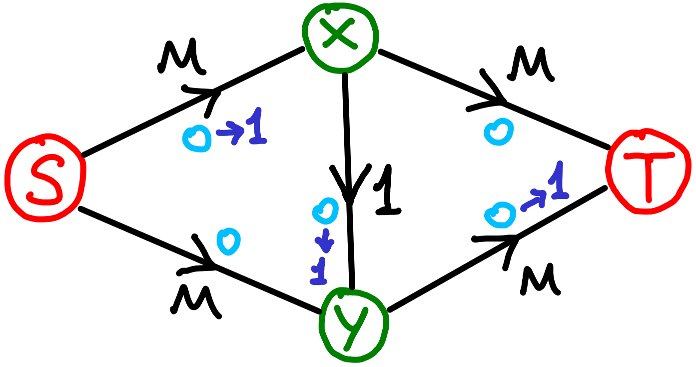
- while augmenting path exists → how many times?

  increase flow on an augmenting path → how do we find one?

  increases flow value.

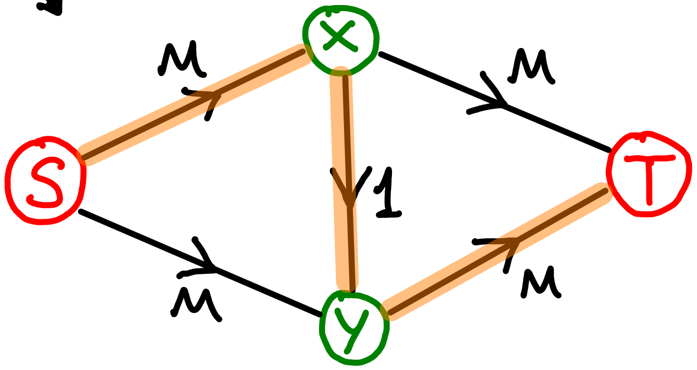
  O(path length)

Arbitrarily?



$M = 10^6$

Residual

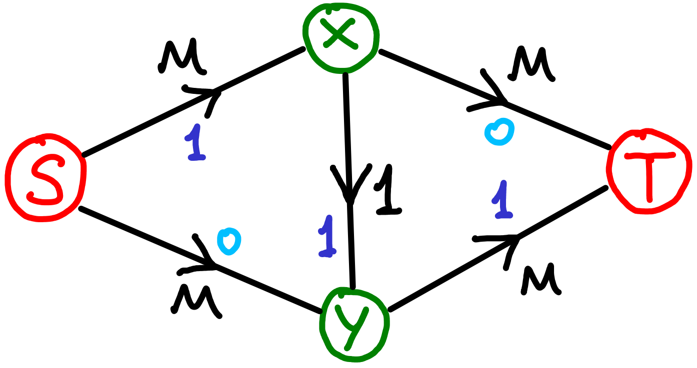


FORD - FULKERSON method

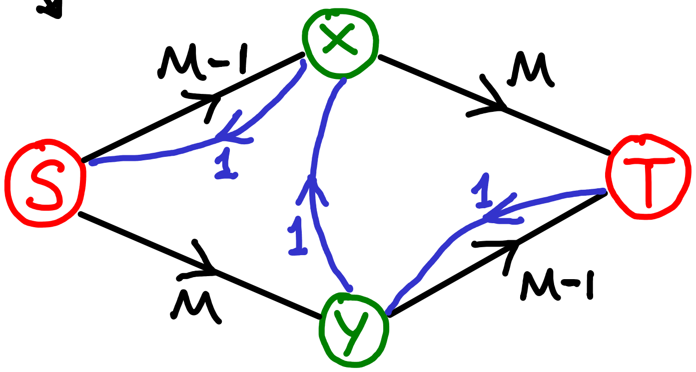
- while augmenting path exists → how many times?
  - increase flow on an augmenting path → how do we find one?

increase flow on an augmenting path: increases flow value. $O(\text{path length})$

Arbitrarily?



Residual

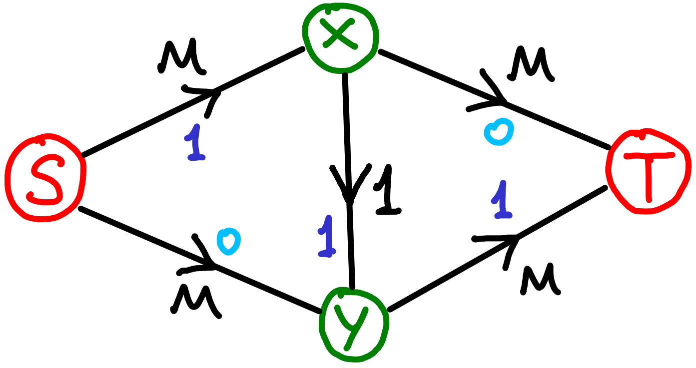


$M = 10^6$

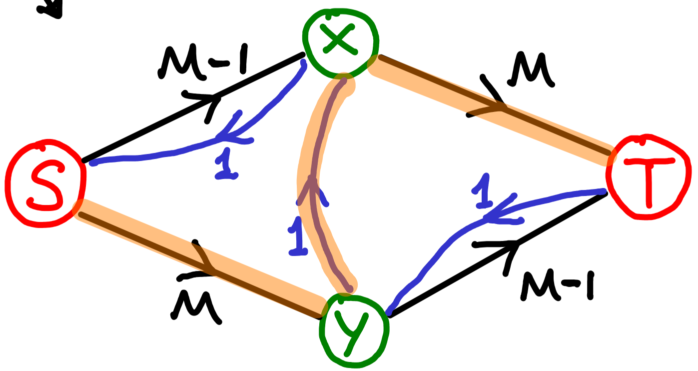
FORD - FULKERSON method

- while augmenting path exists → how many times?
  increase flow on an augmenting path → how do we find one?
  } increases flow value. $O(\text{path length})$

Arbitrarily?



Residual



$M = 10^6$

FORD-FULKERSON method

- while augmenting path exists
  increase flow on an augmenting path

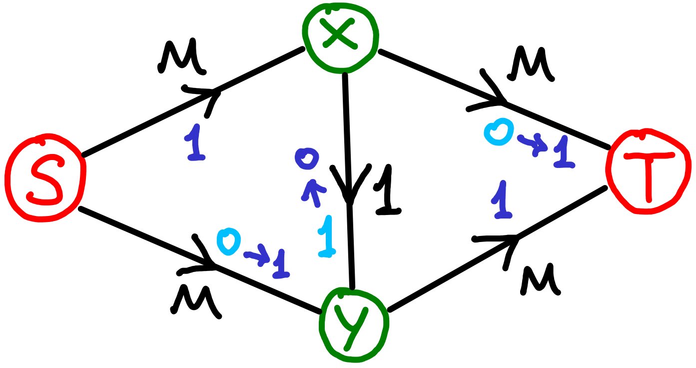
how many times?

how do we find one?

increases flow value.

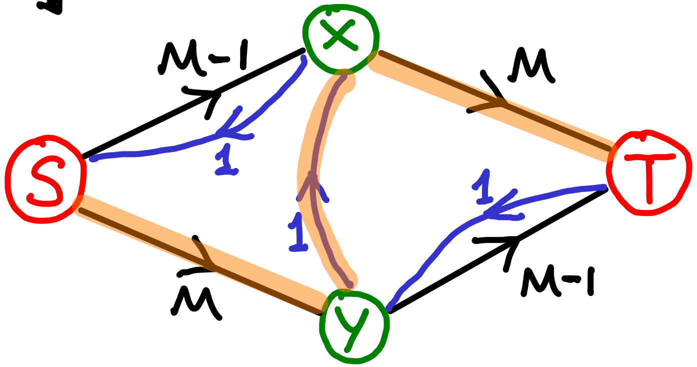
O(path length)

Arbitrarily?



$M = 10^6$

Residual



FORD - FULKERSON method

- while augmenting path exists
  increase flow on an augmenting path

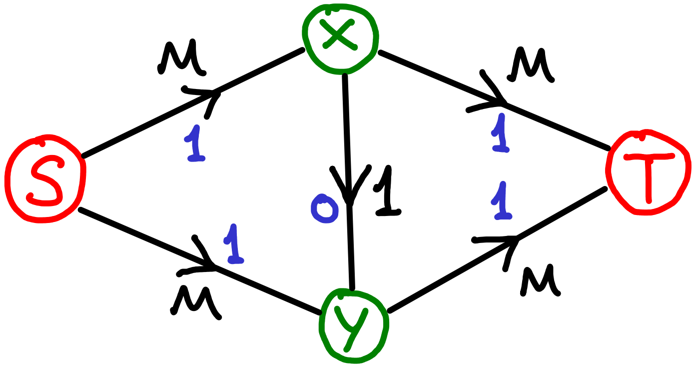
how many times ?

how do we find one ?

increases flow value.

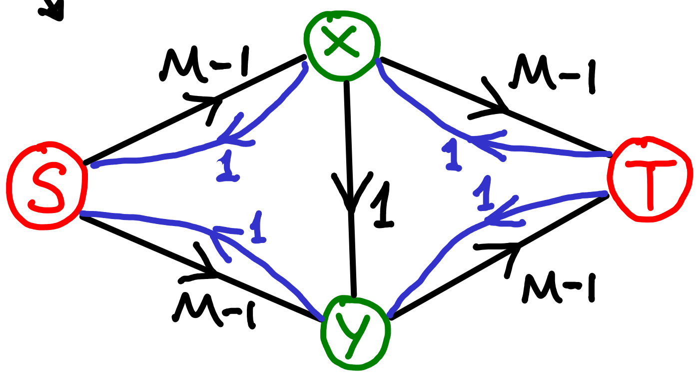
O(path length)

Arbitrarily?



$M = 10^6$

Residual



FORD - FULKERSON method

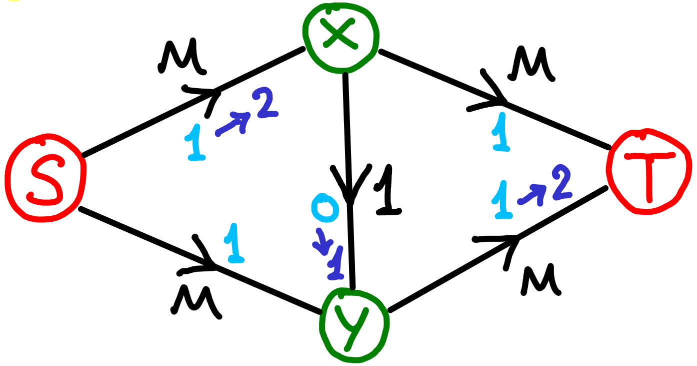
- while augmenting path exists → how many times ?

  increase flow on an augmenting path → how do we find one ?

  { increases flow value.
    O(path length)

Arbitrarily? → #iterations = M

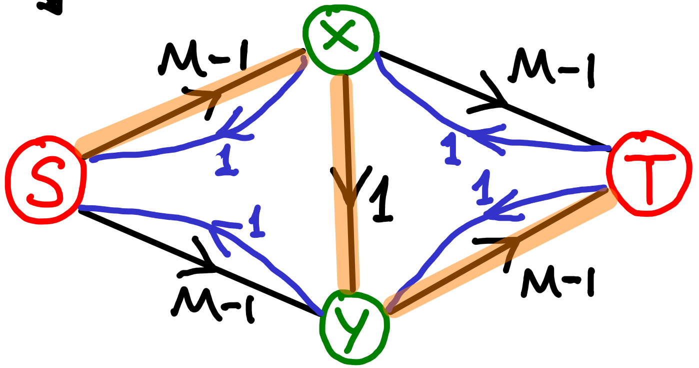
(or even DFS)



$M = 10^6$

etc

Residual



FORD - FULKERSON method

how many times?

how do we find one?

- while augmenting path exists

  increase flow on an augmenting path

increases flow value.

O(path length)

Arbitrarily? NO. In fact if capacities are irrational there are networks causing non-termination & no convergence. (see links)

FORD - FULKERSON method

how many times?

how do we find one?

- while augmenting path exists

  increase flow on an augmenting path

increases flow value.

O(path length)

Arbitrarily? NO. In fact if capacities are irrational there are networks causing non-termination & no convergence. (see links)

For integer capacities, #iterations ≤ MAX flow

For rational, transform to integer (multiply)

FORD - FULKERSON method

- while augmenting path exists

  how many times?

  increase flow on an augmenting path

  use BFS to find one

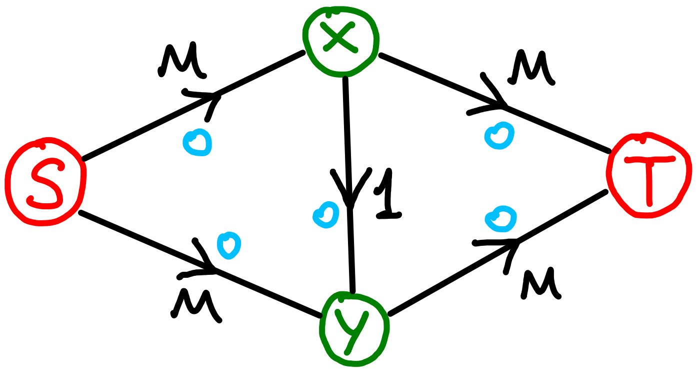
  increases flow value.

  O(path length)

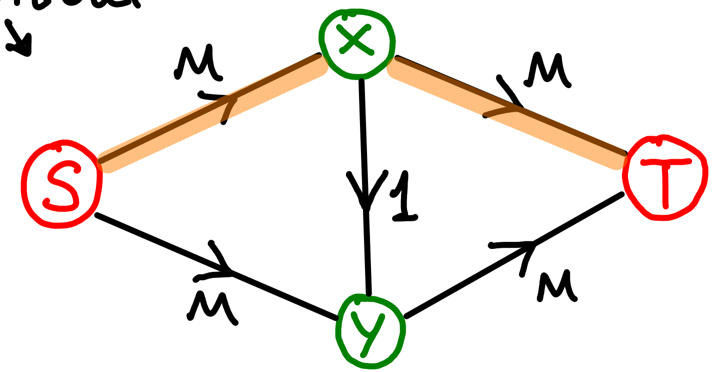
FORD - FULKERSON method

- while augmenting path exists → how many times ?
  increase flow on an augmenting path → use BFS to find one

  increases flow value.
  O(path length)



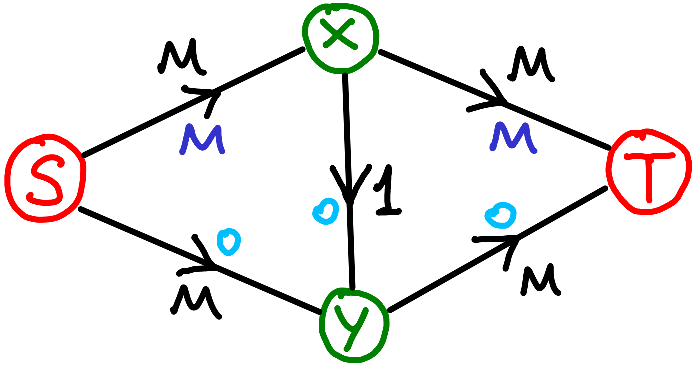
Residual



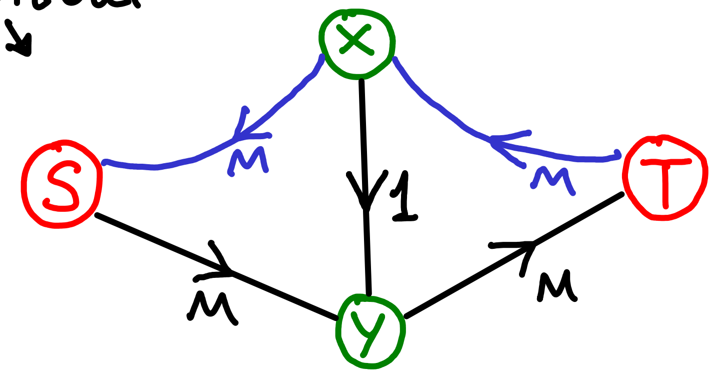
FORD - FULKERSON method

- while augmenting path exists → how many times?
  increase flow on an augmenting path → use BFS to find one

  { increases flow value.
  O(path length)



Residual



FORD - FULKERSON method

how many times ?

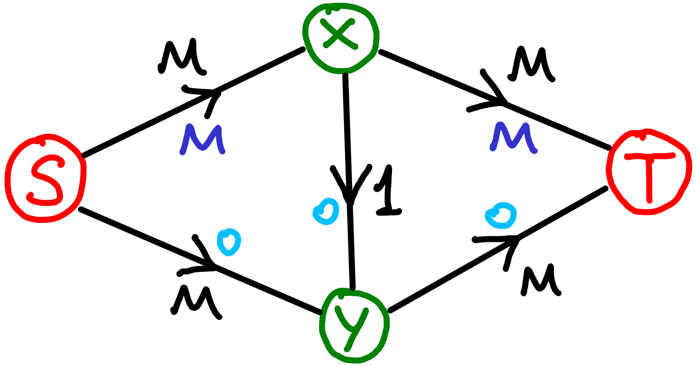
use BFS to find one

- while augmenting path exists

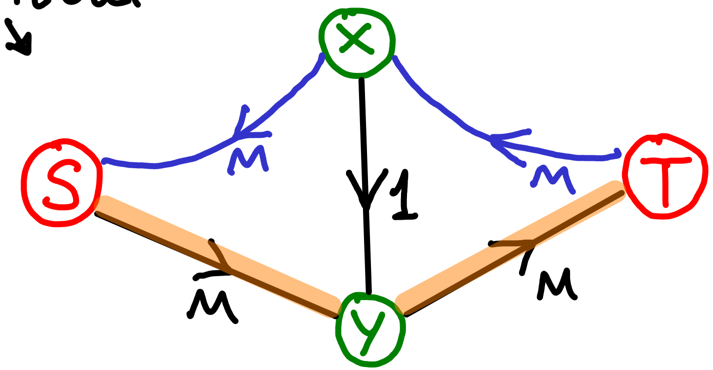
  increase flow on an augmenting path

increases flow value.

O(path length)



Residual



FORD - FULKERSON method

- while augmenting path exists

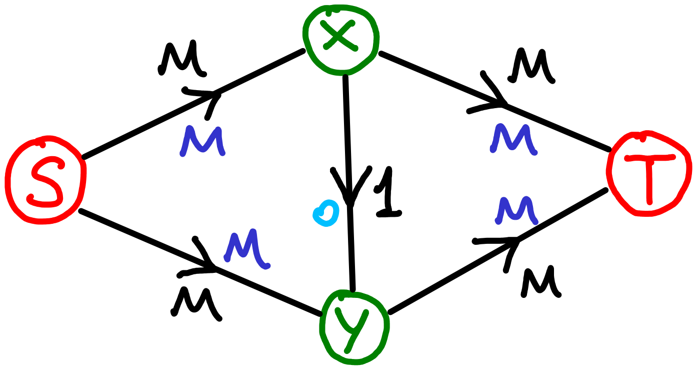
  how many times ?

  increase flow on an augmenting path

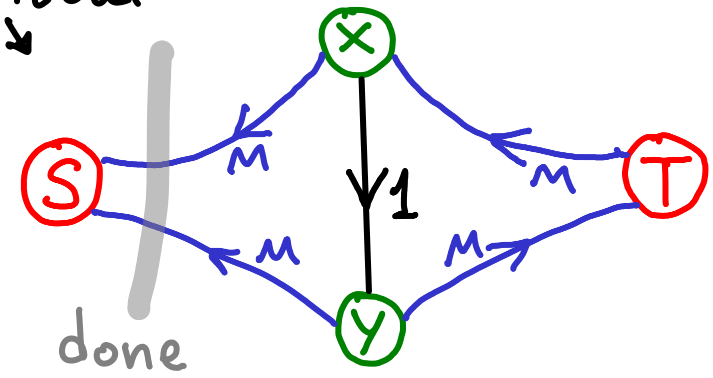
  use BFS to find one

  increases flow value.

  $O(\text{path length})$



Residual



FORD - FULKERSON method

- while augmenting path exists → how many times?
  increase flow on an augmenting path → use BFS to find one

  increases flow value.
  O(path length)

Using BFS: Edmonds-Karp algorithm

FORD - FULKERSON method

- while augmenting path exists — how many times?

  increase flow on an augmenting path — use BFS to find one: $O(E)$

  increases flow value.

  $O(\text{path length})$

Using BFS: Edmonds-Karp algorithm : $O(V \cdot E)$ iterations

so $O(V \cdot E^2)$ time

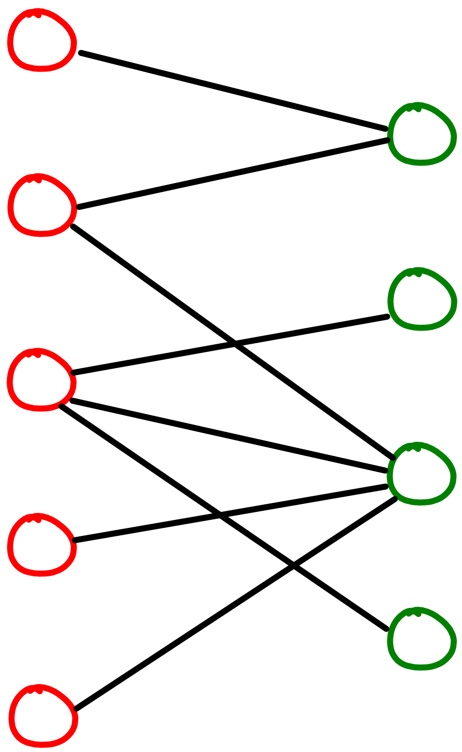
FORD - FULKERSON method

- while augmenting path exists → how many times ?
  increase flow on an augmenting path → use BFS to find one: $O(E)$
  increases flow value.
  $O(\text{path length})$

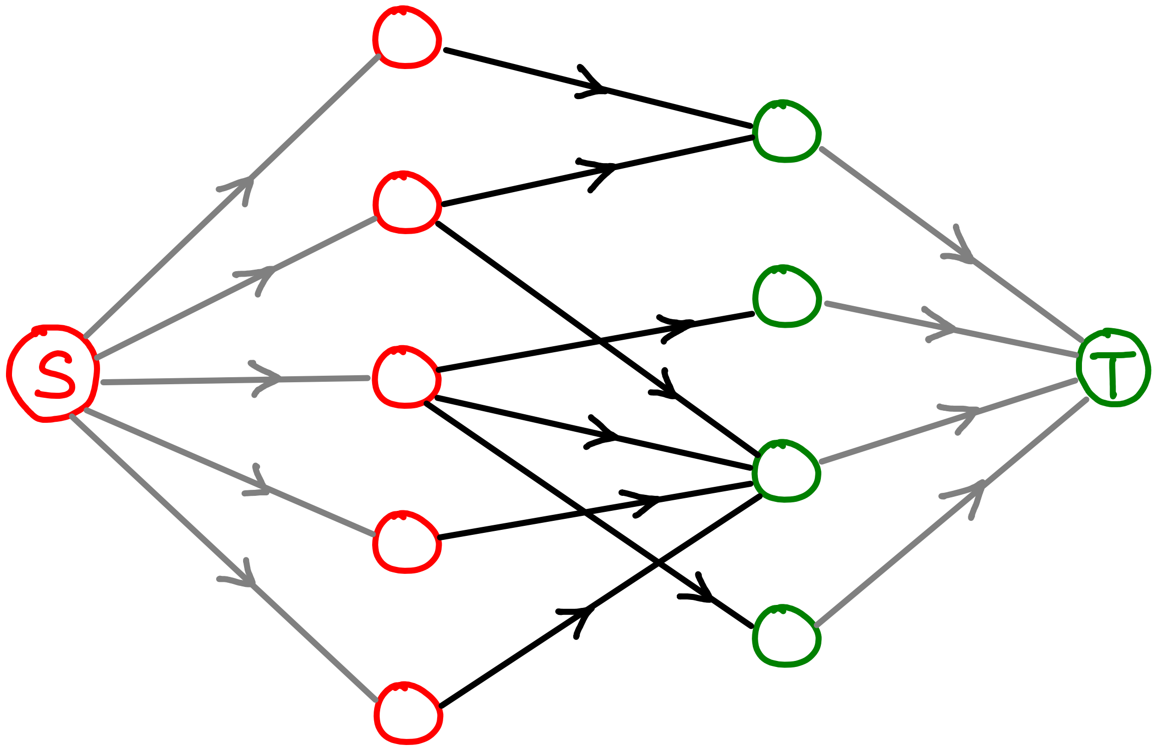
Using BFS: Edmonds-Karp algorithm : $O(V \cdot E)$ iterations
so $O(V \cdot E^2)$ time

$O(V^2 E)$ & $O(V^3)$ algorithms also exist.

FINDING A MAXIMUM BIPARTITE MATCHING

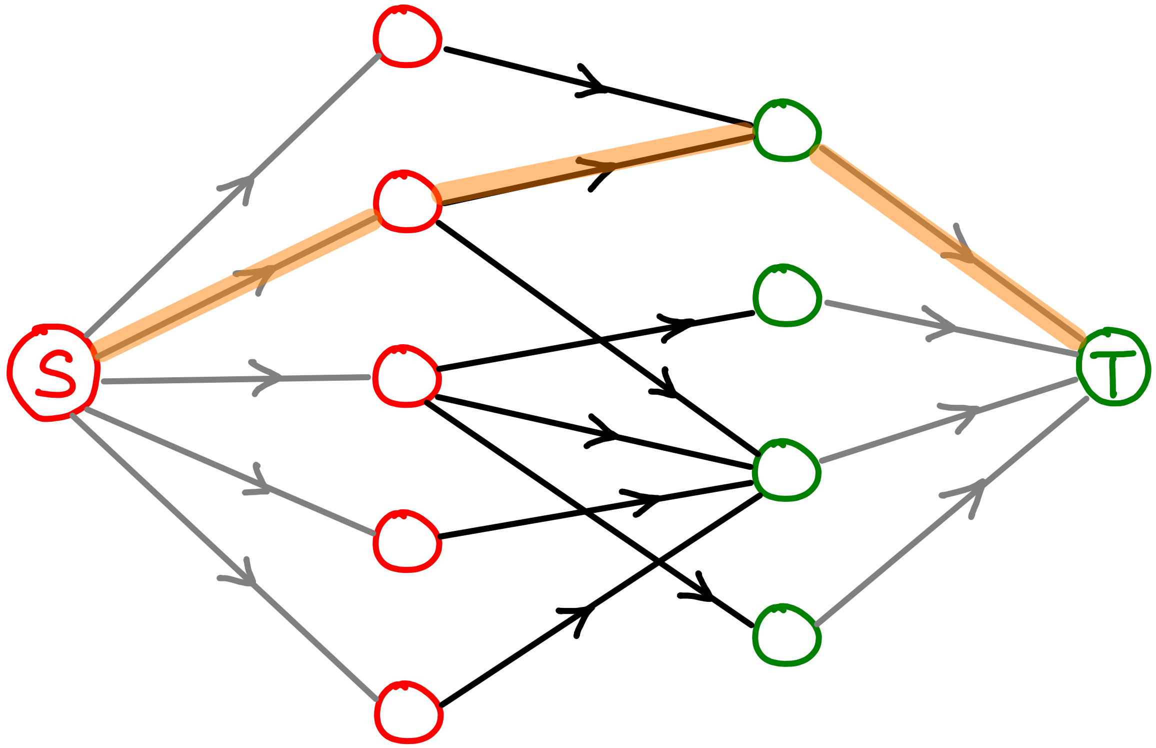


FINDING A MAXIMUM BIPARTITE MATCHING



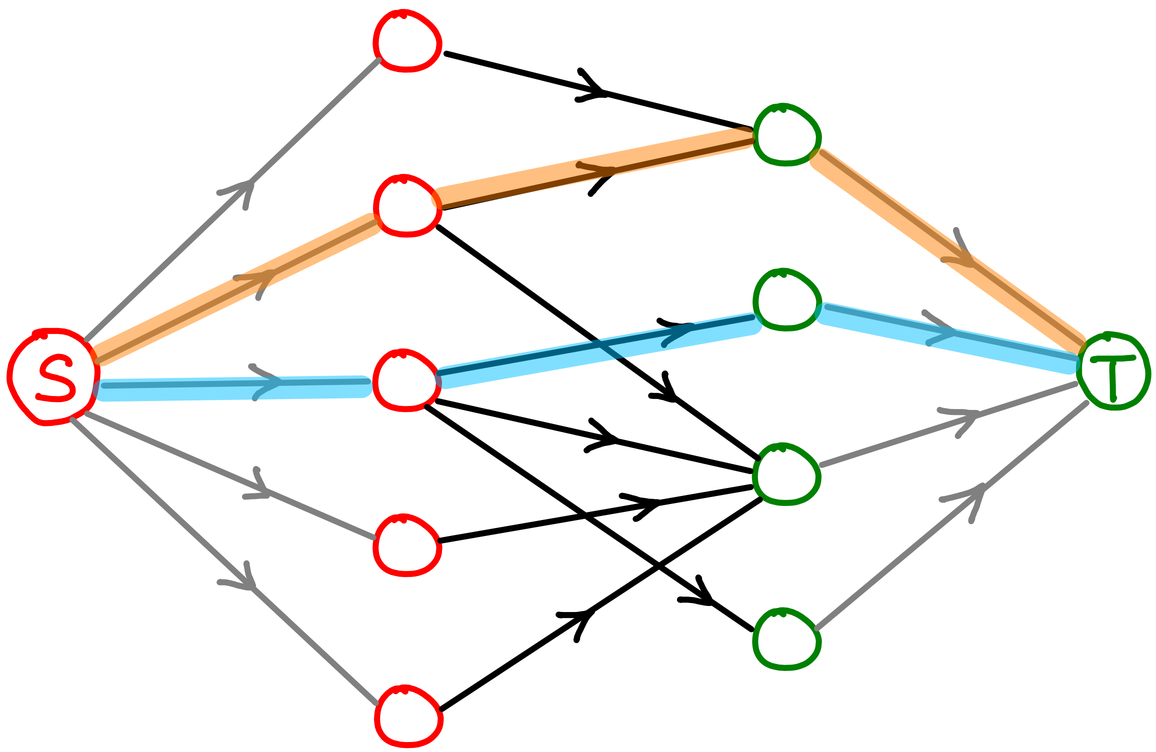
All capacities = 1

FINDING A MAXIMUM BIPARTITE MATCHING



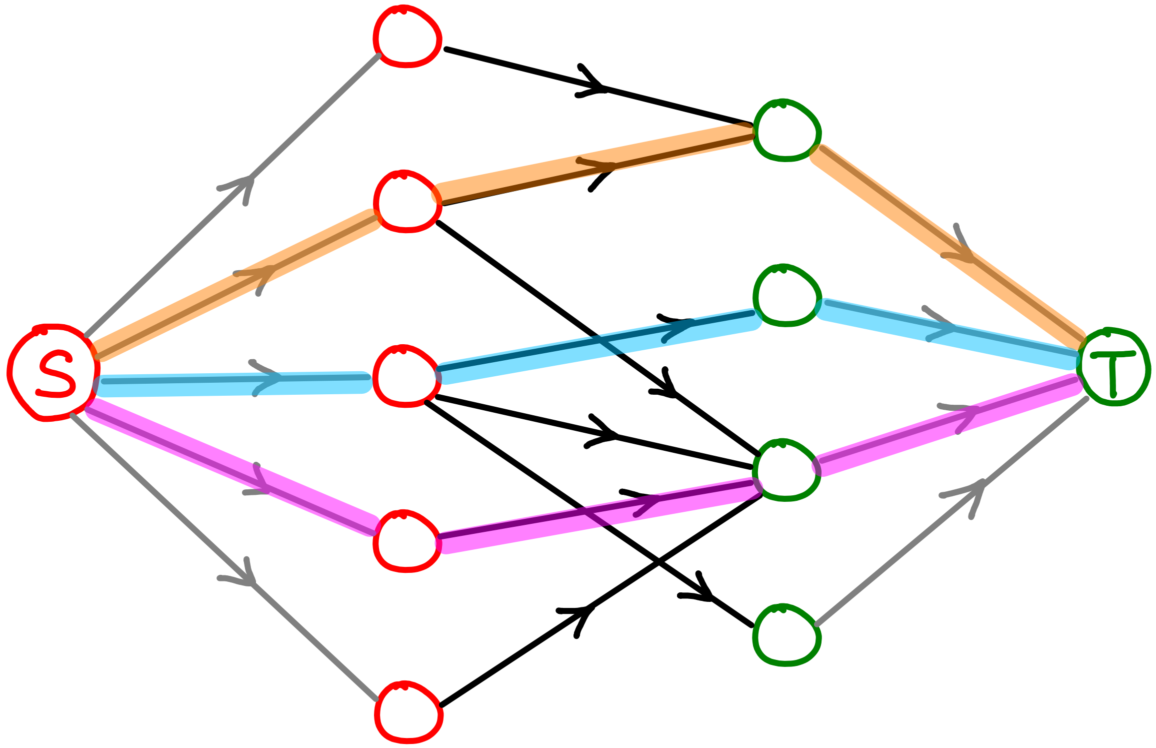
All capacities = 1

FINDING A MAXIMUM BIPARTITE MATCHING



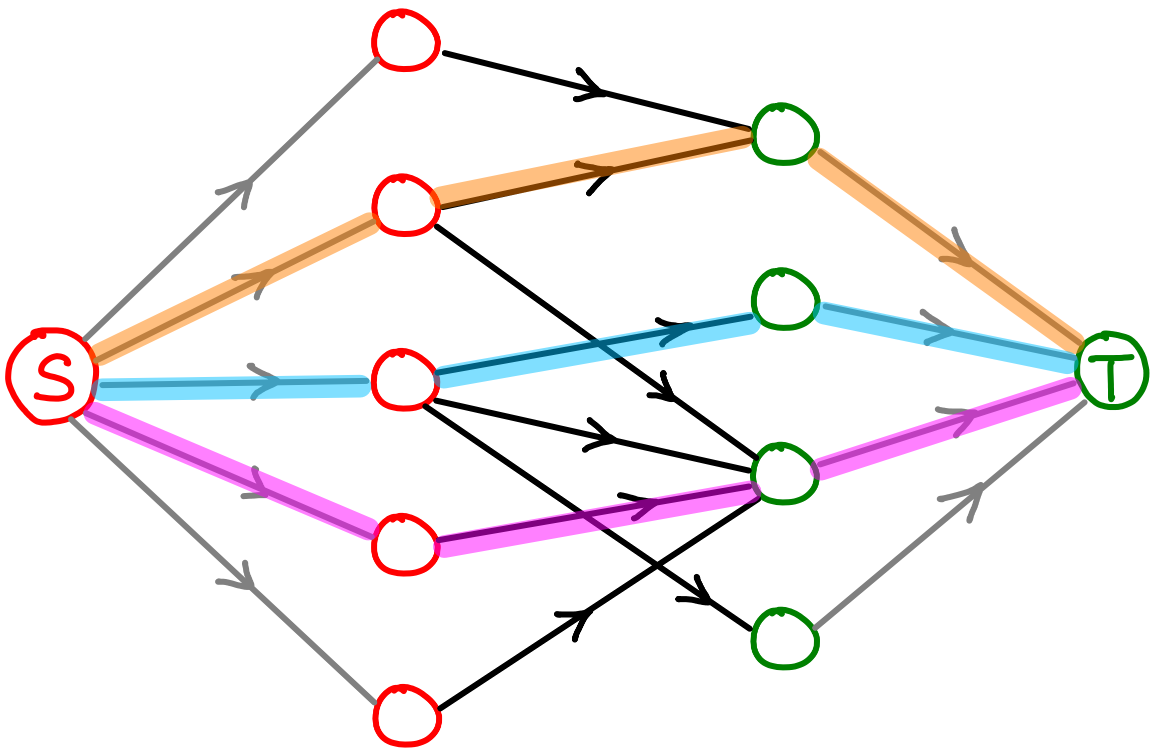
All capacities = 1

FINDING A MAXIMUM BIPARTITE MATCHING



All capacities = 1

FINDING A MAXIMUM BIPARTITE MATCHING

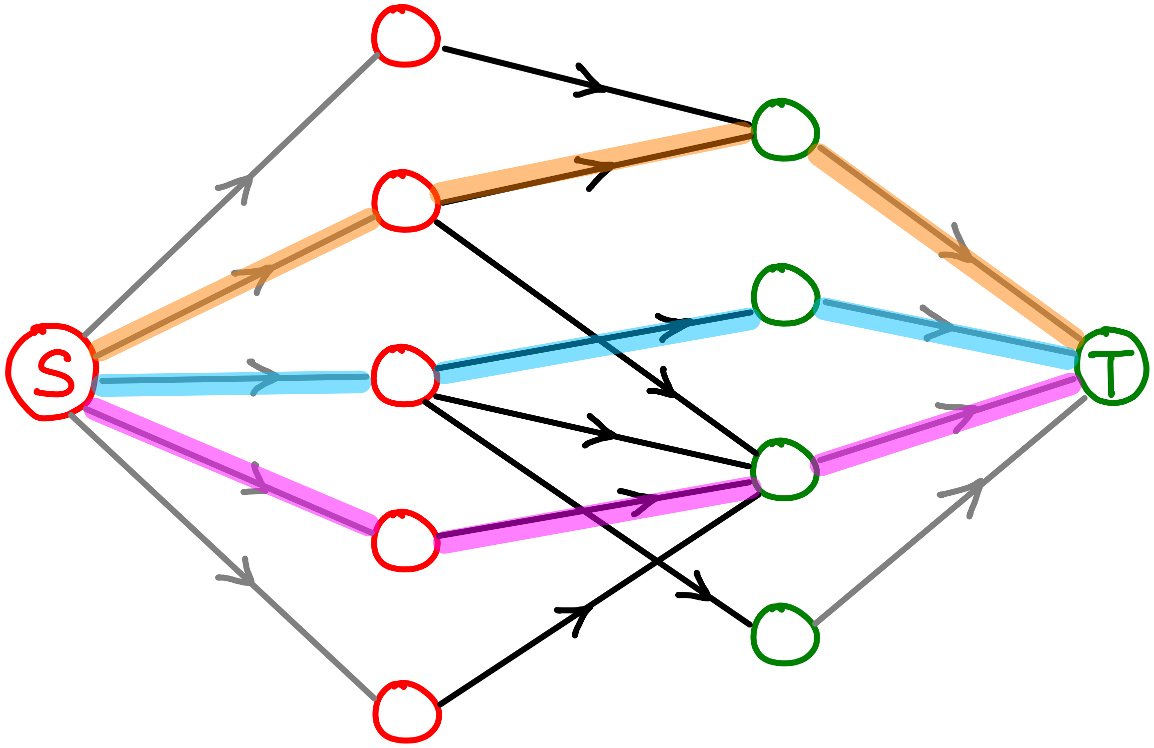


All capacities = 1

(integer)

- MAX FLOW = O(V)

FINDING A MAXIMUM BIPARTITE MATCHING



All capacities = 1

(integer)

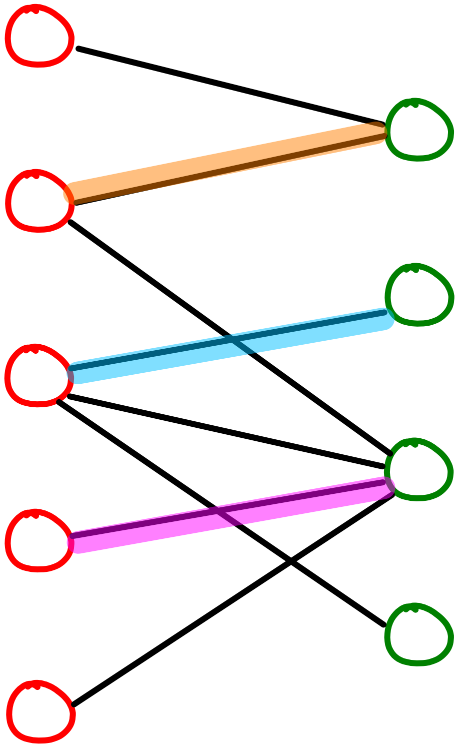
• MAX FLOW = O(V)

iterations

BFS

so O(V·E) time

FINDING A MAXIMUM BIPARTITE MATCHING



All capacities = 1

(integer)

• MAX FLOW = $O(V)$

iterations

BFS

so $O(V \cdot E)$ time

* it can be shown that integer input gives integer flow on edges