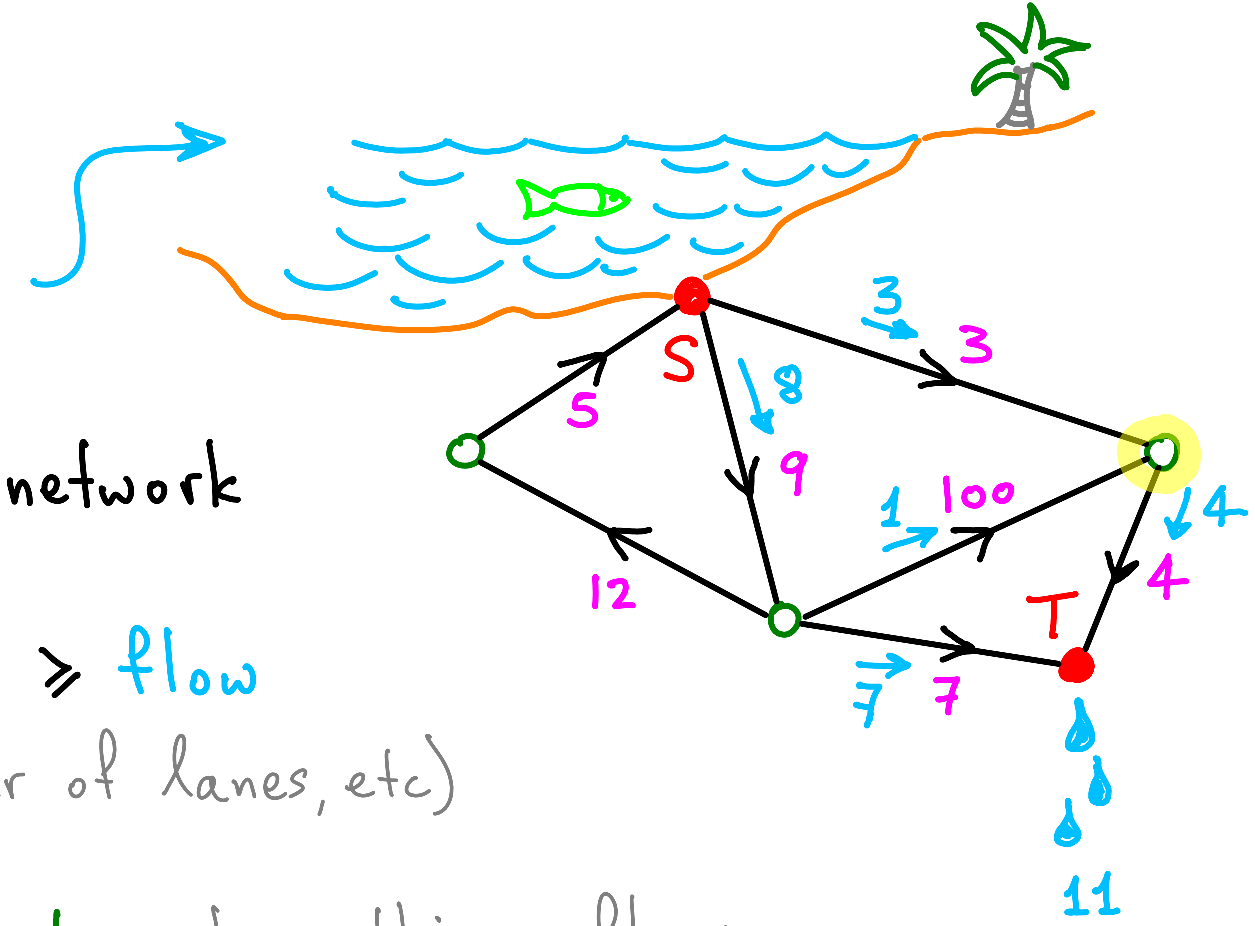
NETWORK FLOW – FLOW NETWORKS

network = directed graph

S : supply : unbounded input

T : target : output limited only by network



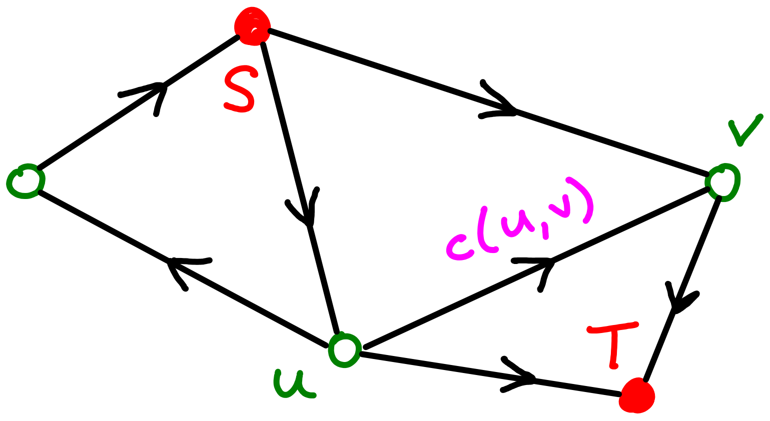
Each edge has a capacity $\geq$ flow

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

∘ Vertices : Σ input $= \Sigma$ output → keep things flowing

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

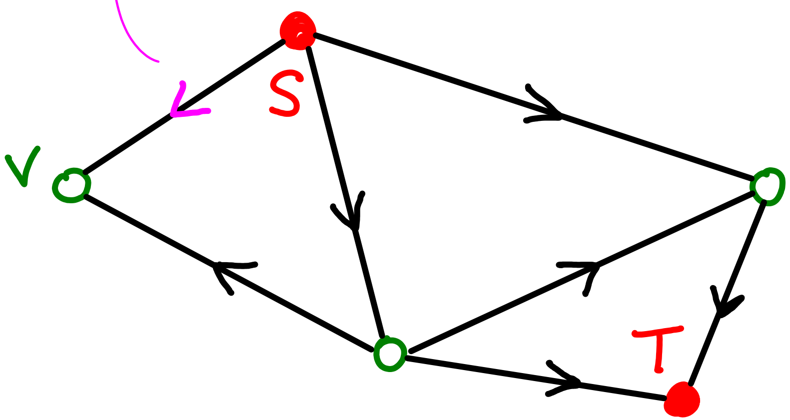
$f(x,y)$ = actual flow in edge $\overrightarrow{xy}$ → TBD



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

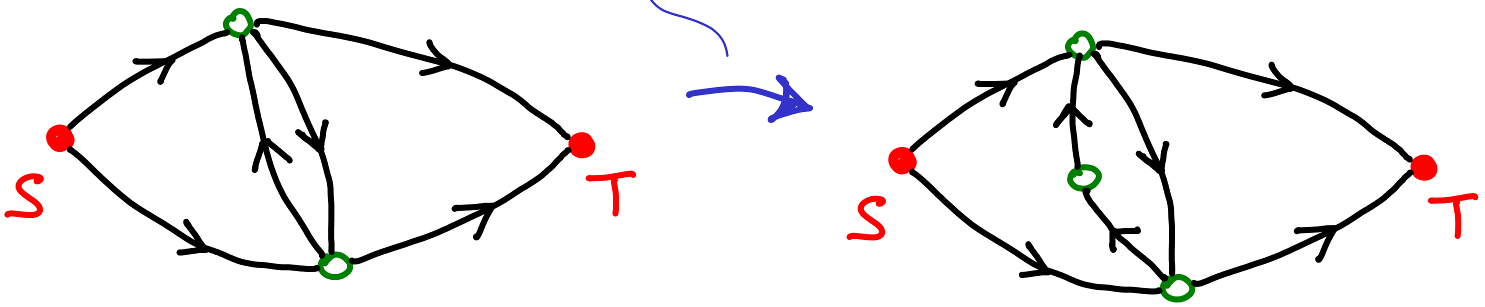


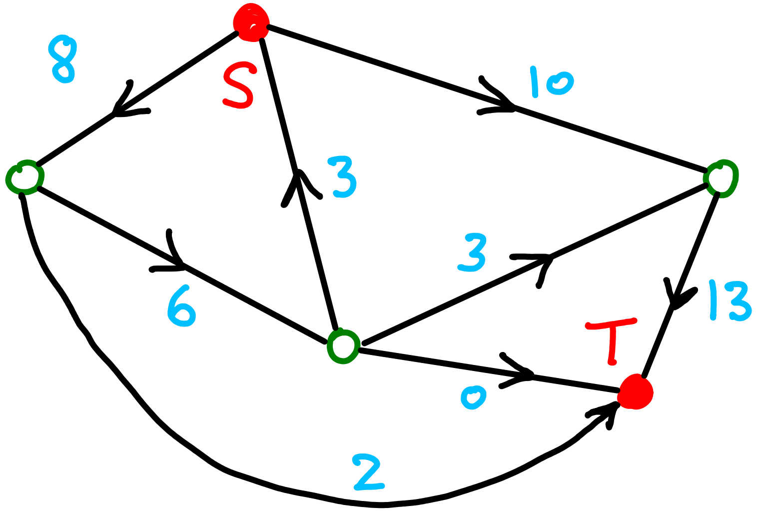
v : should be removed

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

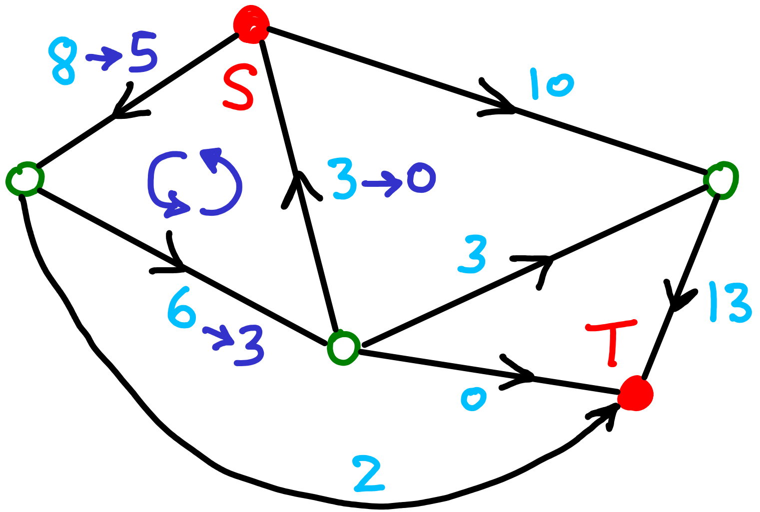
Both directions allowed, but we can simulate.

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





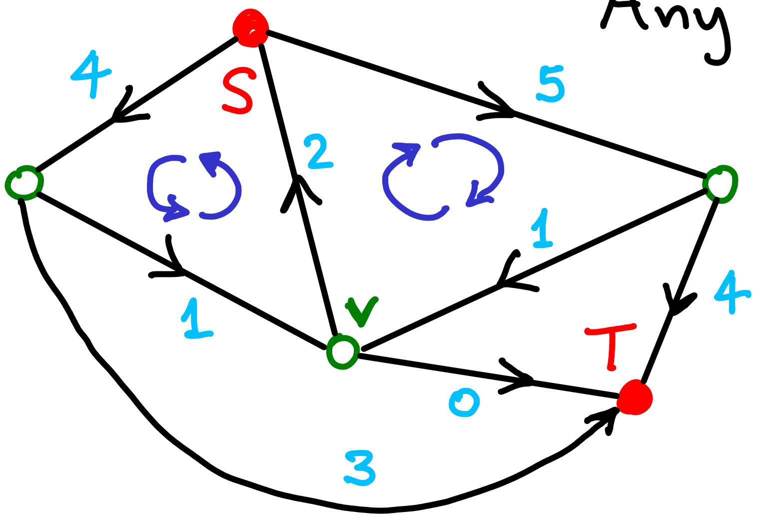
Can we simplify this?

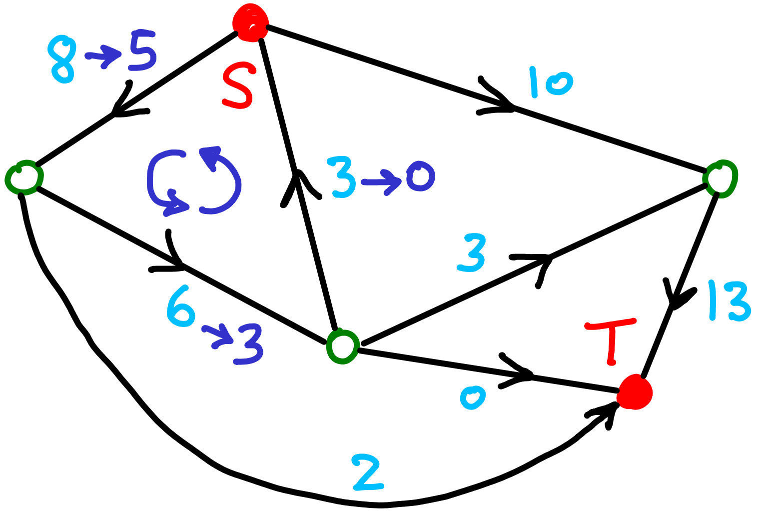


If any cycle has flow on all edges, we can get an equivalent solution with ≥1 edge having zero flow.

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

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

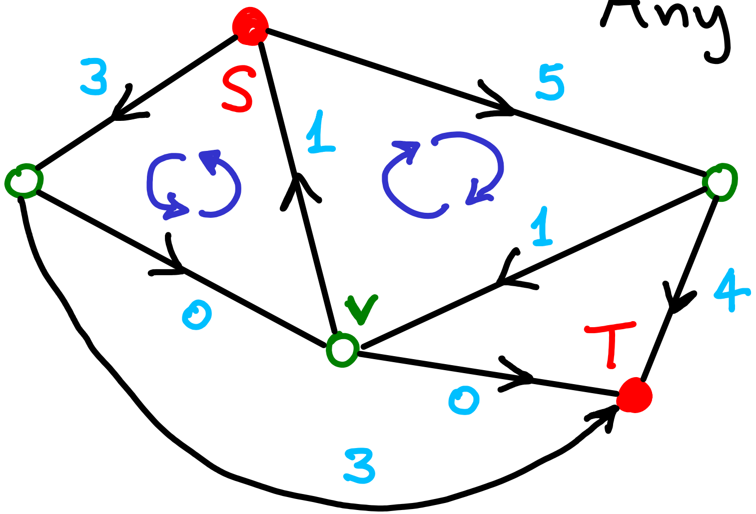


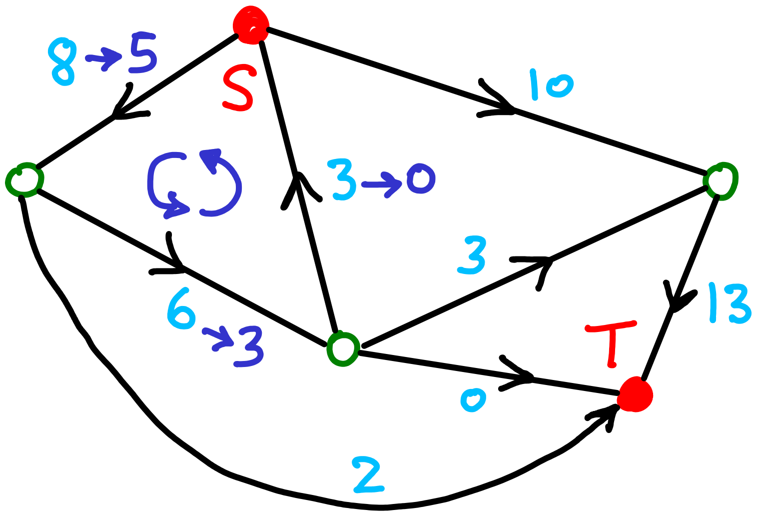


If any cycle has flow on all edges, we can get an equivalent solution with ≥ 1 edge having zero flow.

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

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

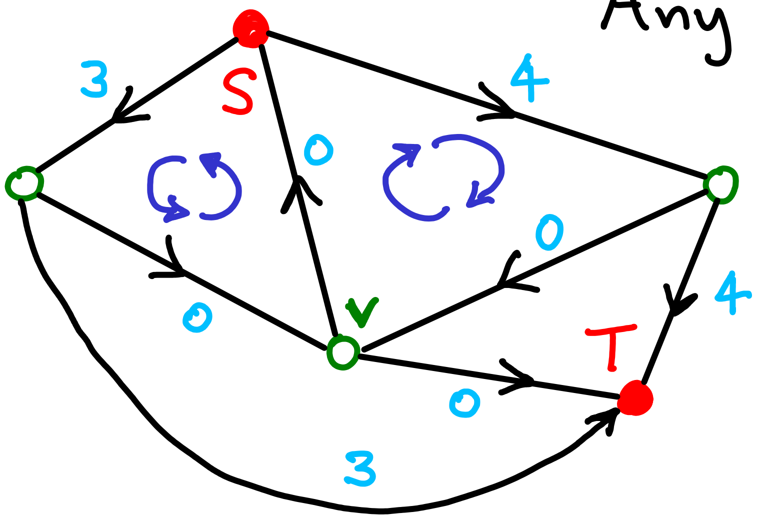


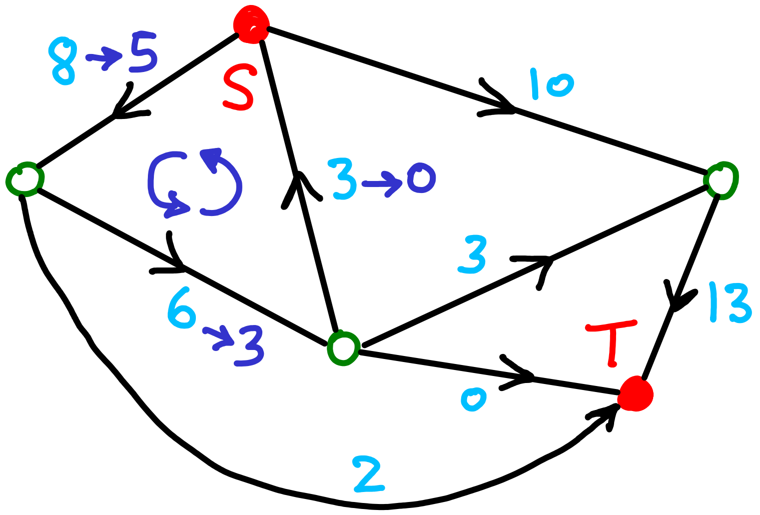


If any cycle has flow on all edges, we can get an equivalent solution with ≥1 edge having zero flow.

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

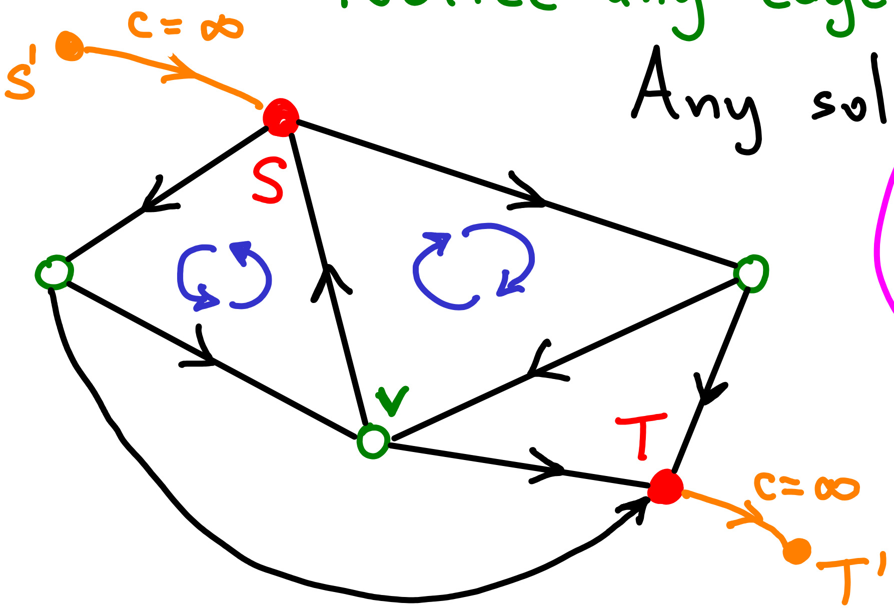
Any solution has an equivalent with $f(v,S) = 0$





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.



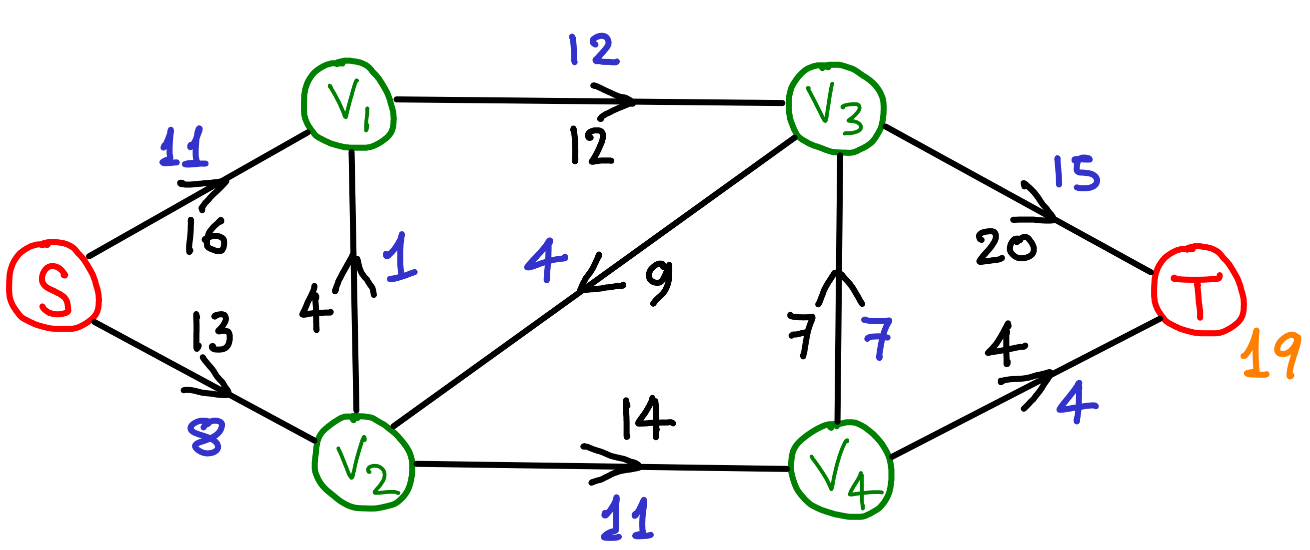
Any solution has an equivalent with $f(v,S)=0$

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

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

(can also resolve multisource/target networks)

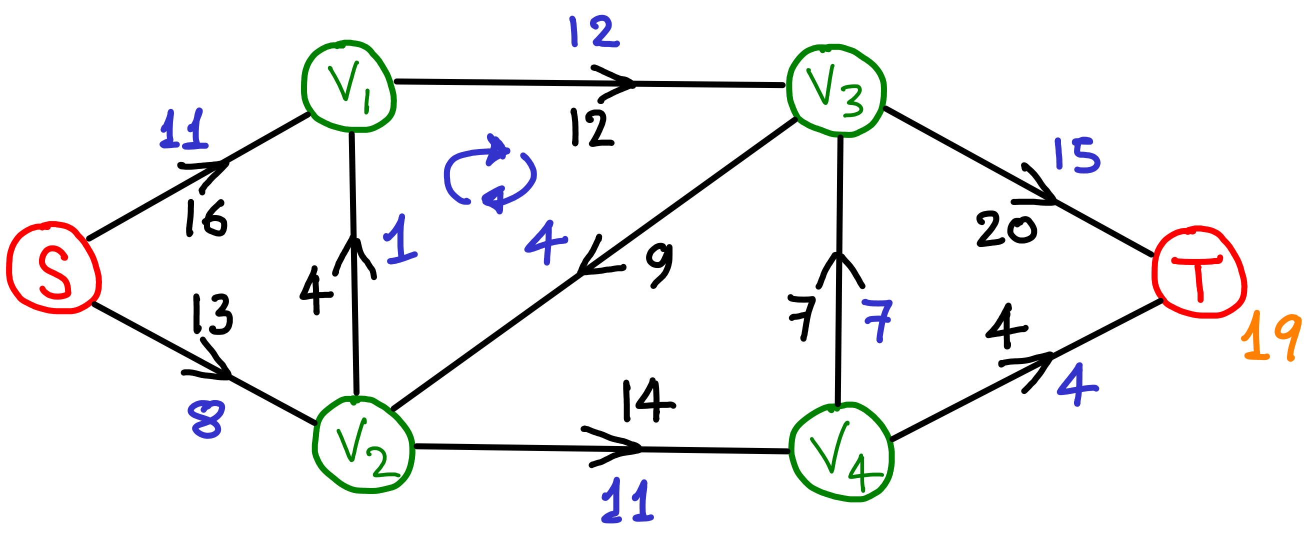
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

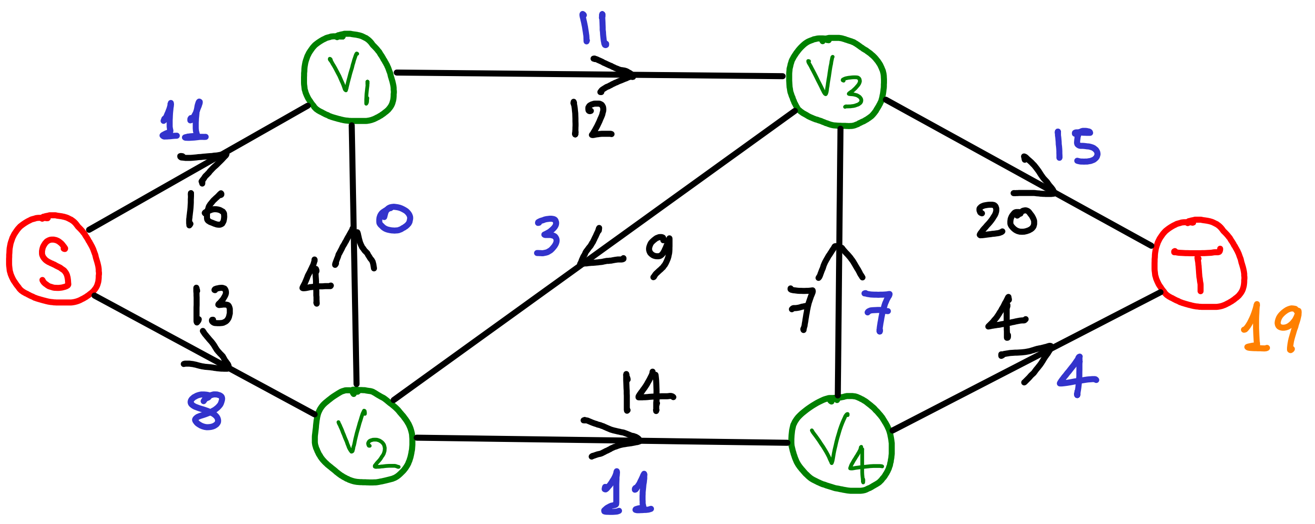
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

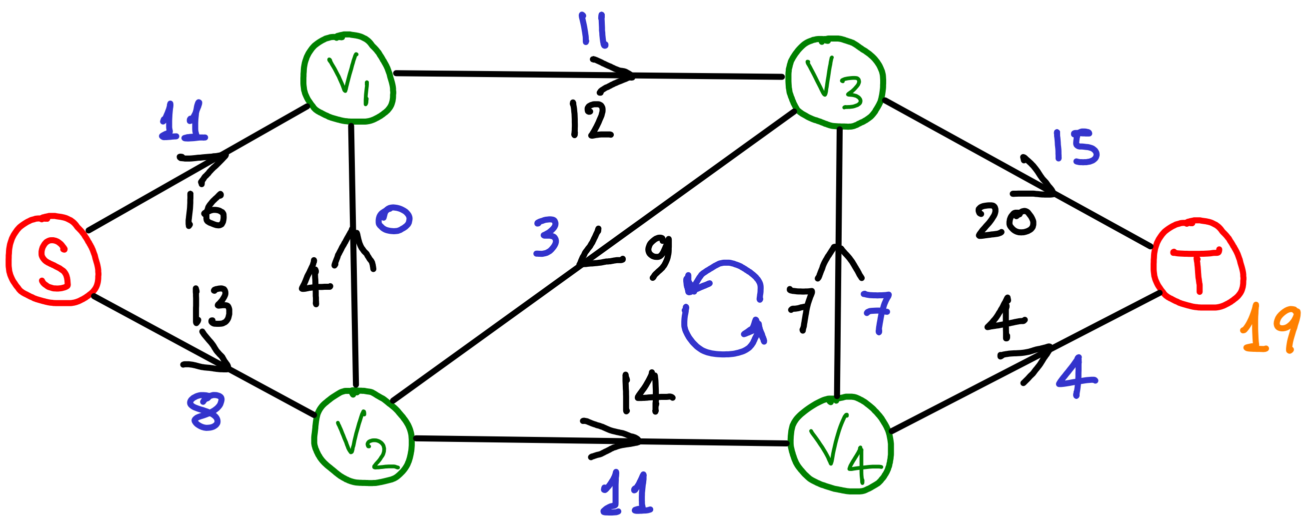
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

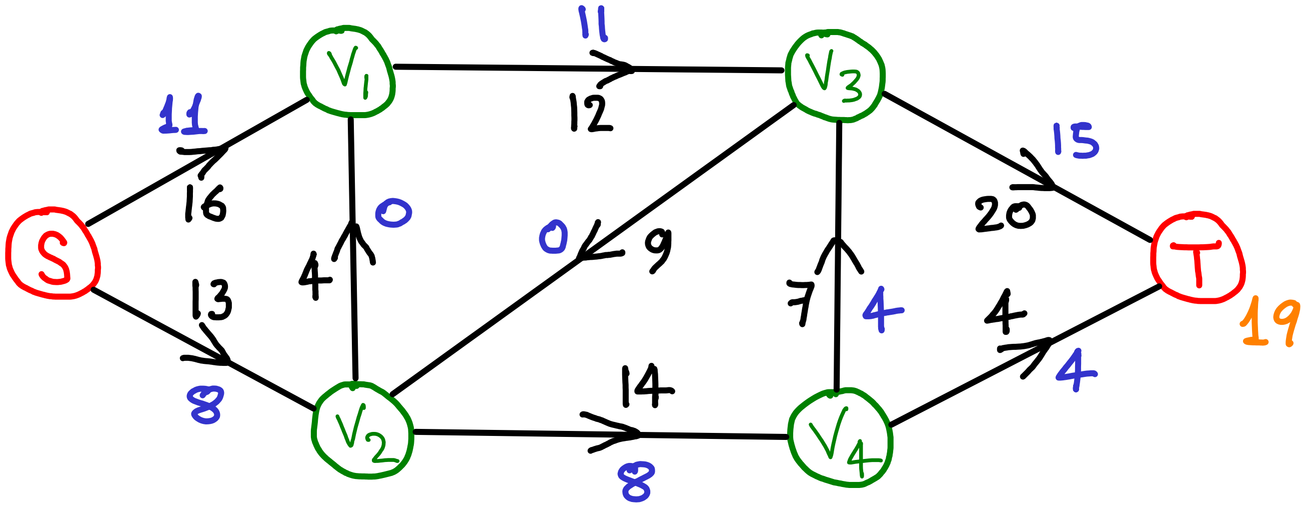
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

#: flows

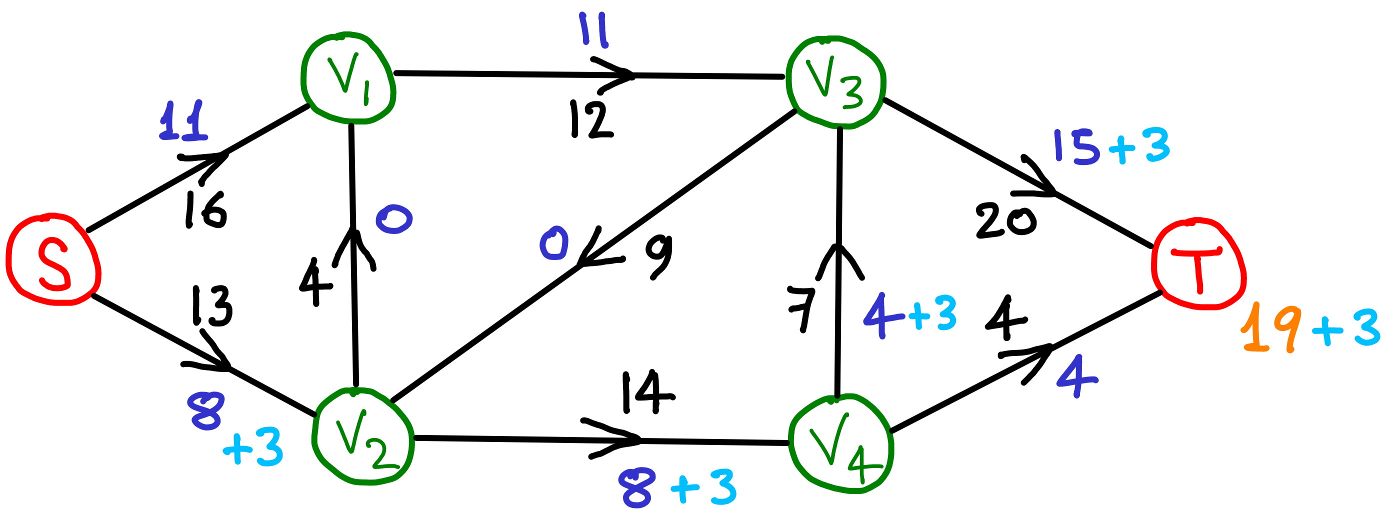
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



#: capacities

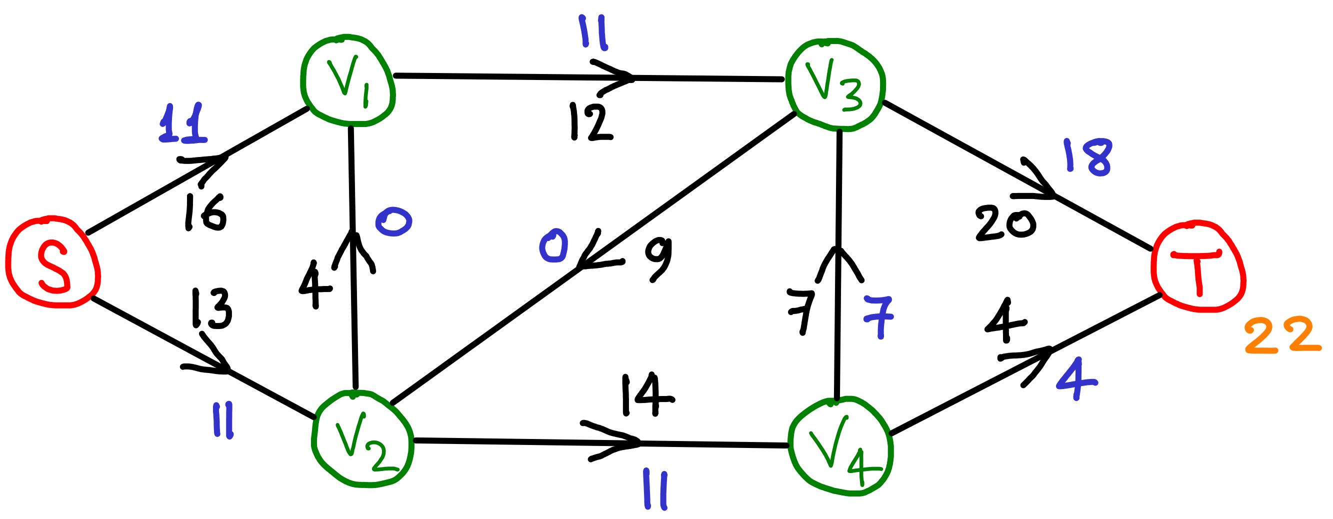
#: flows

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



#: capacities
#: flows

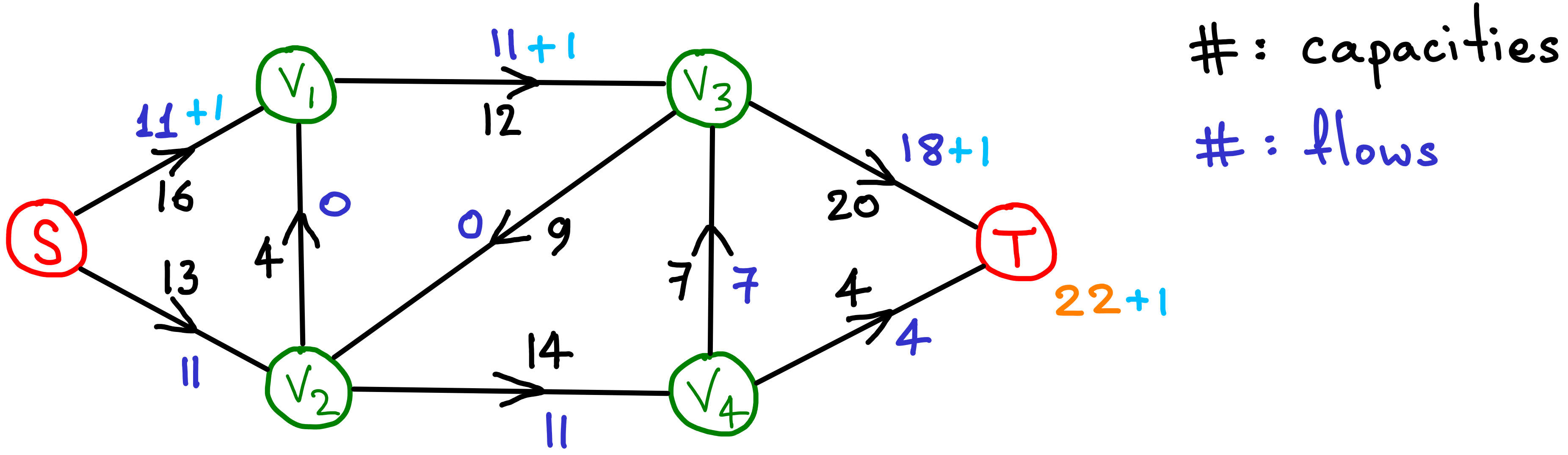
Goal: maximize FLOW VALUE $= \sum_{\text{all } v} f(S,v) = \sum_{\text{all } v} f(v,T)$



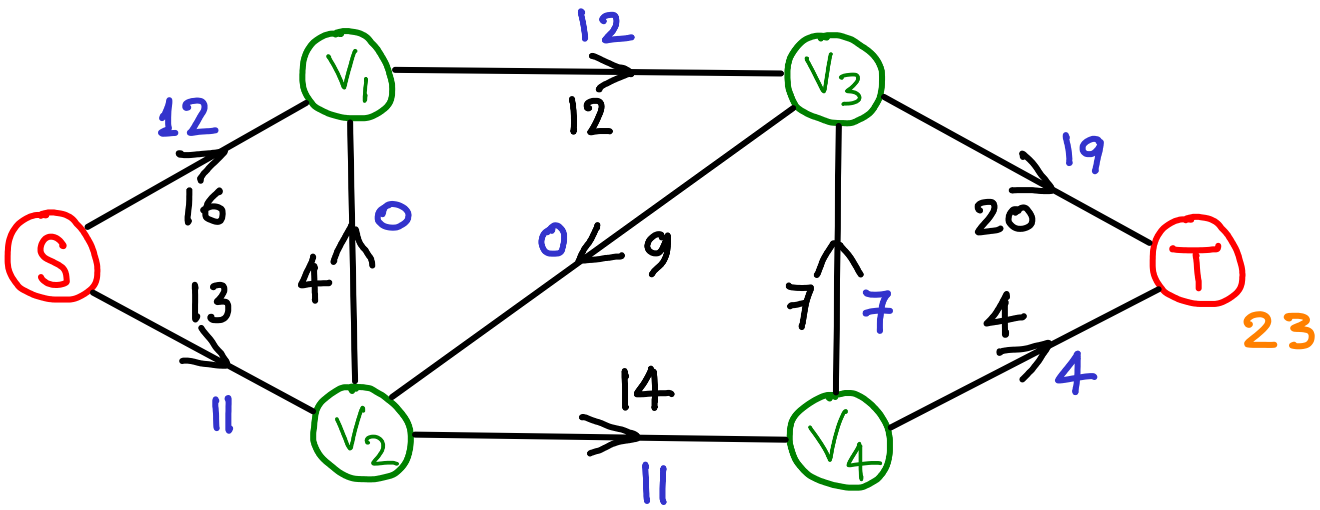
#: capacities

#: flows

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



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



#: capacities

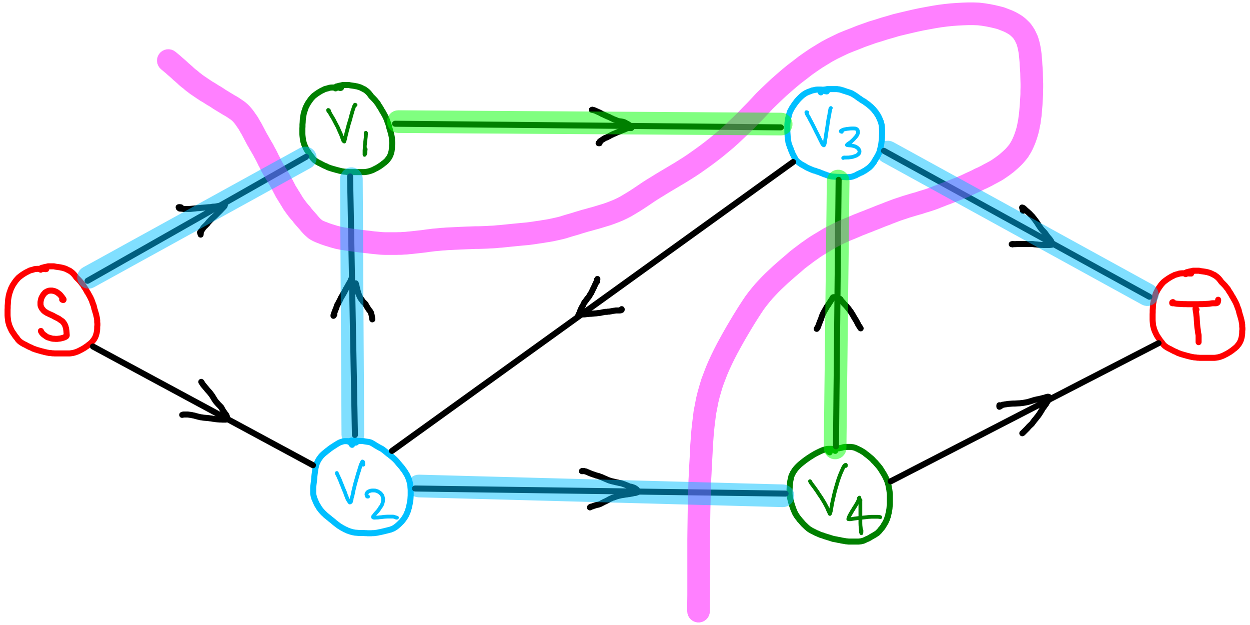
#: flows

Ideas : - simplify cycles

- find positive paths from S to T

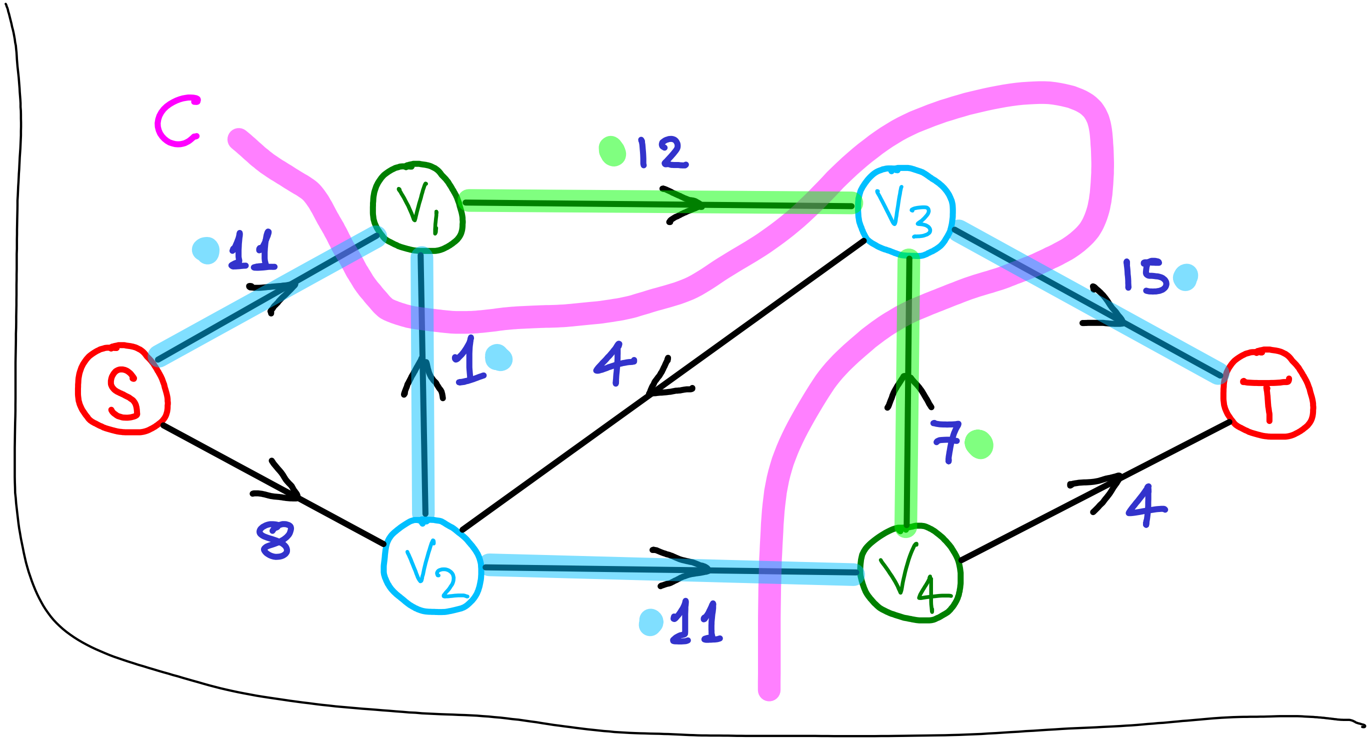
CUTS

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



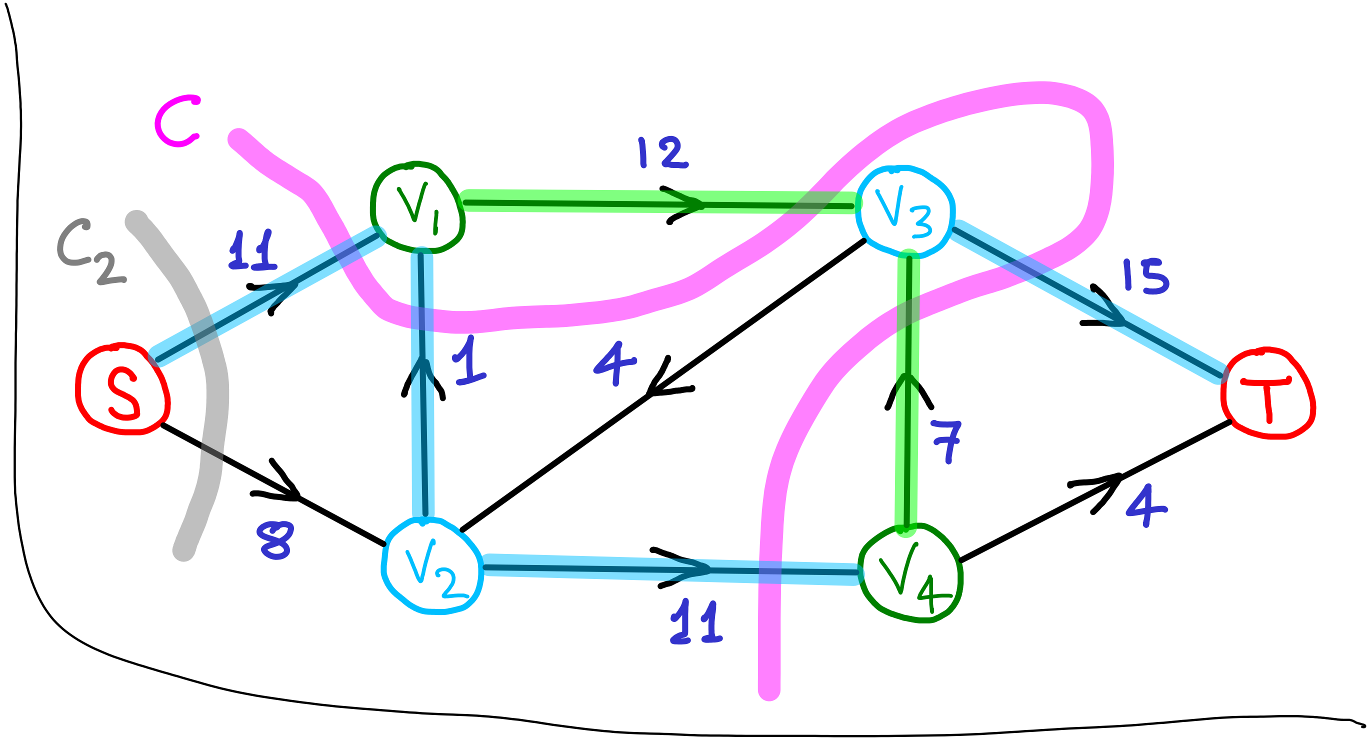
$$\text{Flow}(C) = \sum \text{flow} - \sum \text{flow}$$

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



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

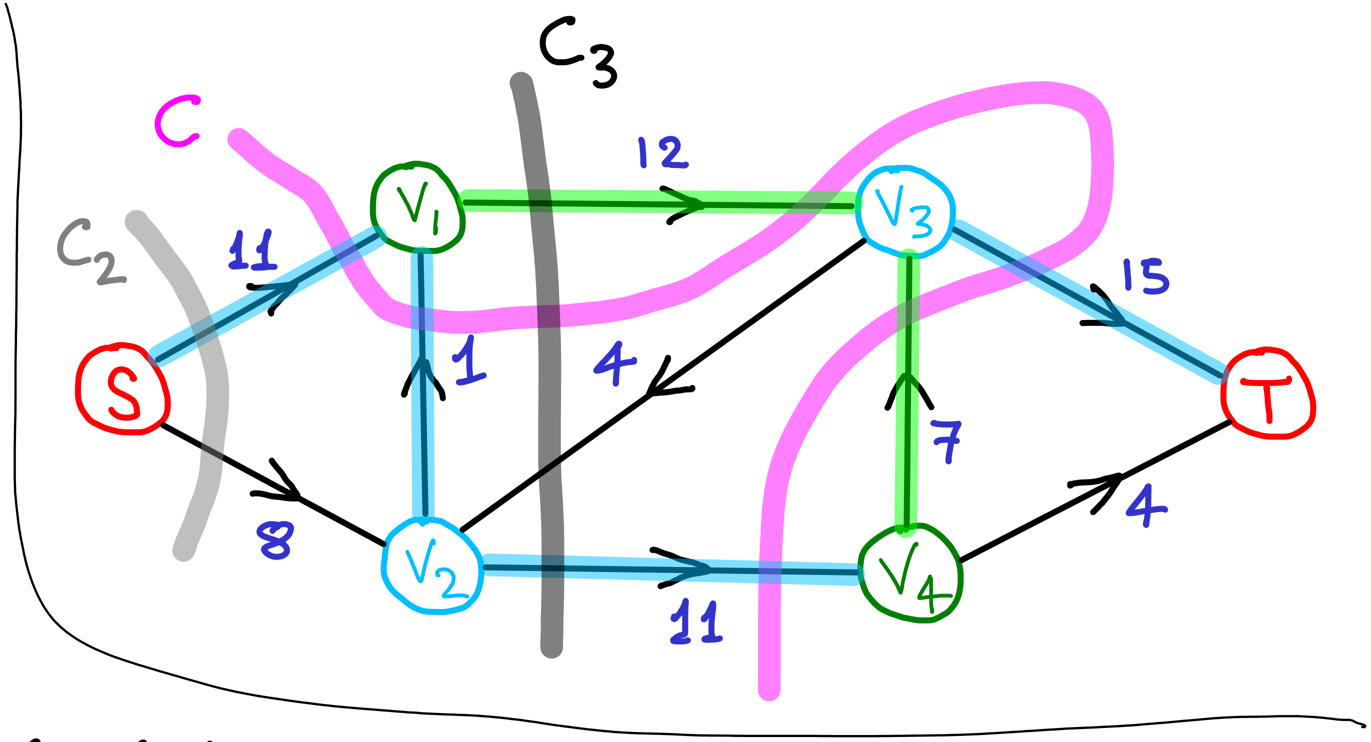
$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$



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

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

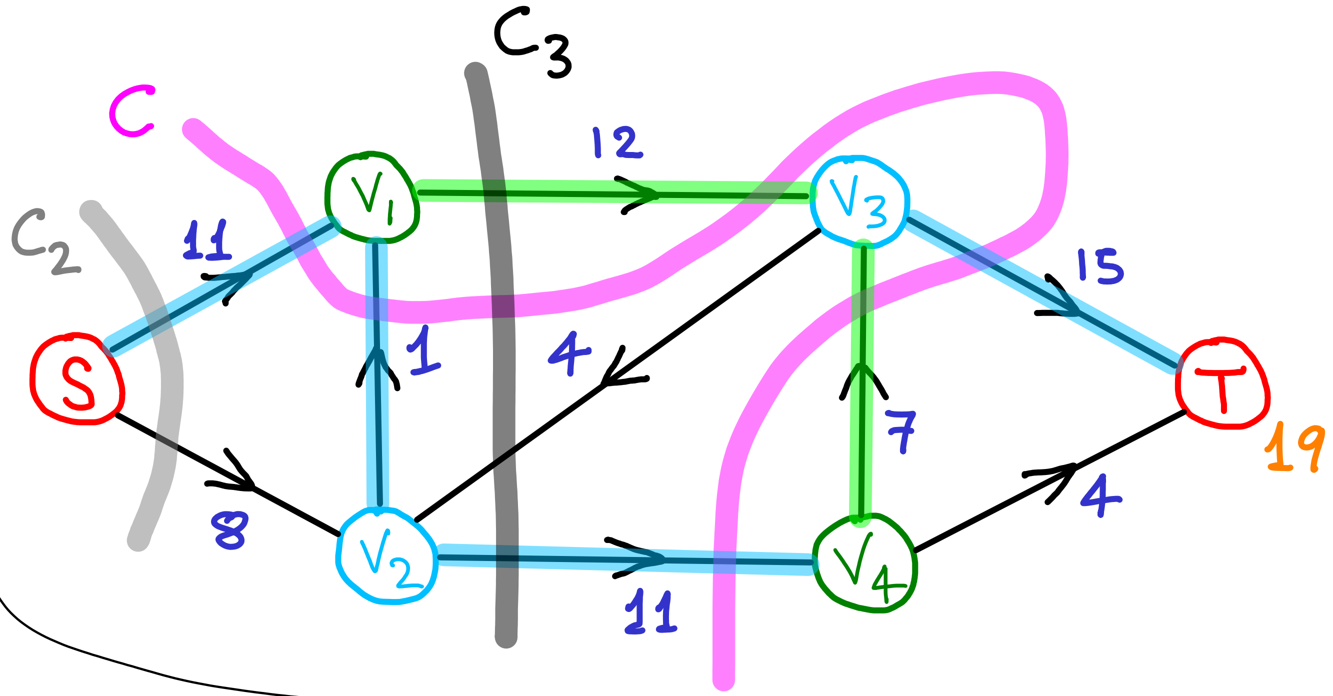
$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$



$$\text{Flow}(C) = \text{Flow}(C_2) = \underset{11+12-4}{\text{Flow}(C_3)}$$

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

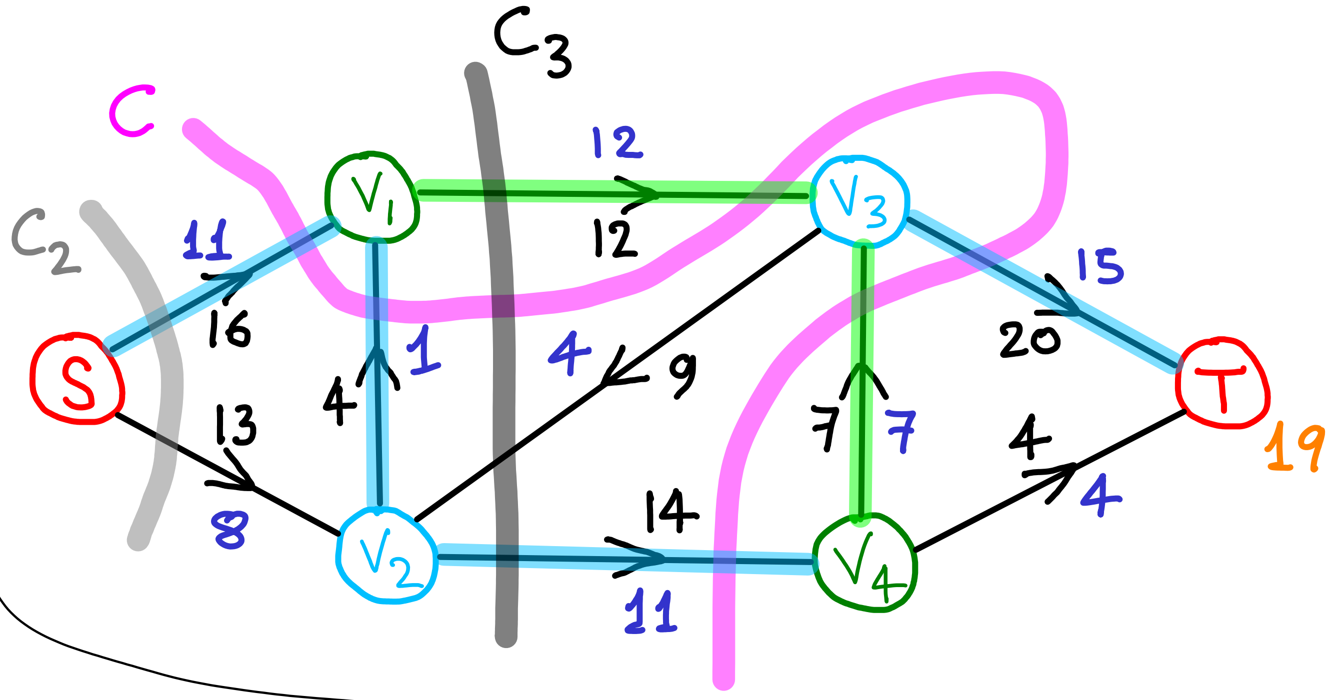
$$= 11 + 1 + 11 + 15 - 12 - 7 = 19$$



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

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

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



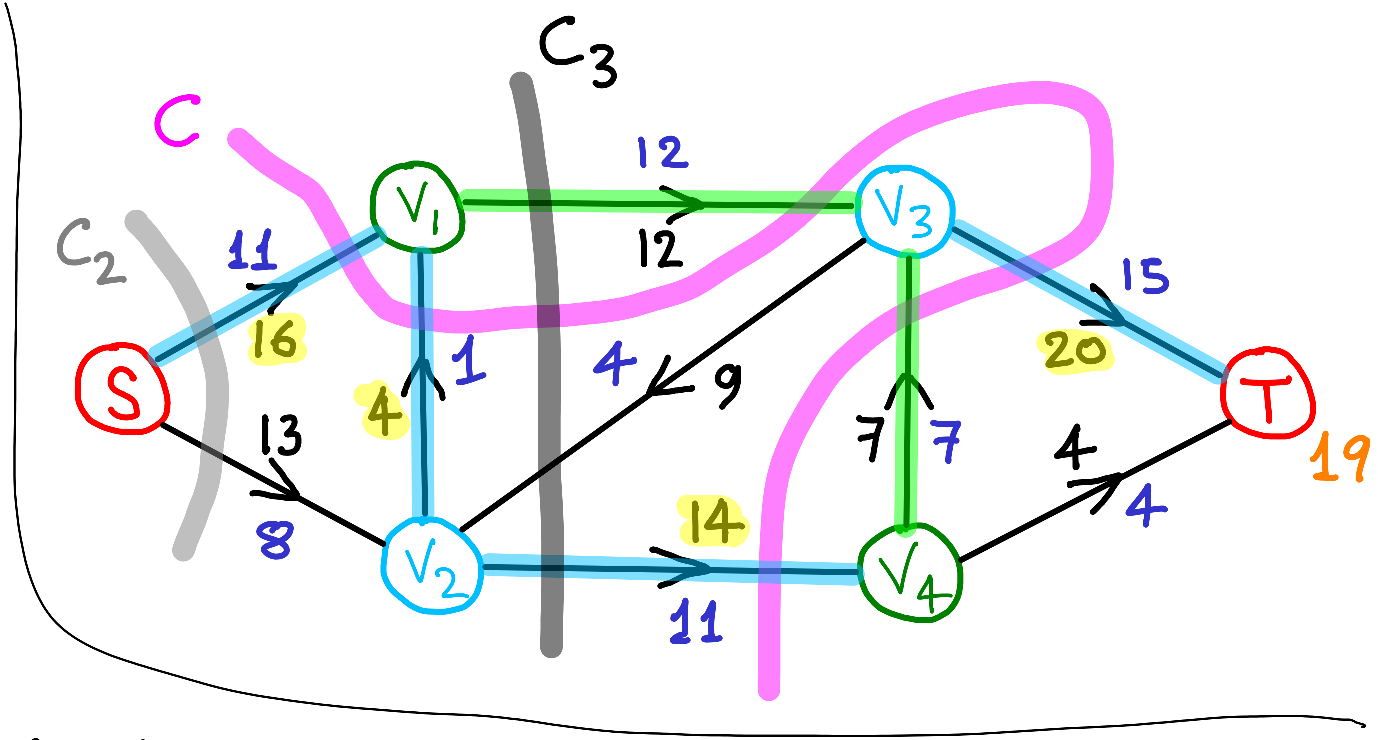
$$\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) = \text{FLOW VALUE}$$

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

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

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

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



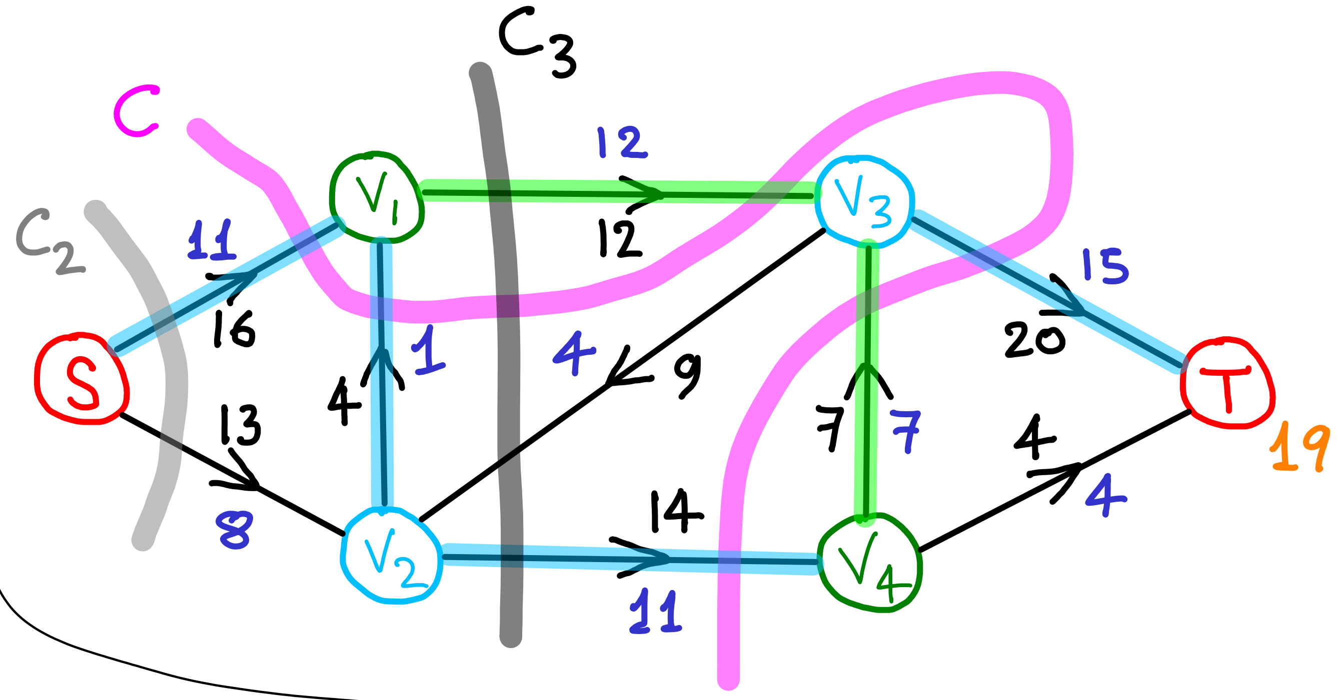
$$\text{Flow}(C) = \text{Flow}(C_2) = \text{Flow}(C_3) = \text{Flow}(\text{any cut}) = \text{FLOW VALUE}$$

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

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

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

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



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

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

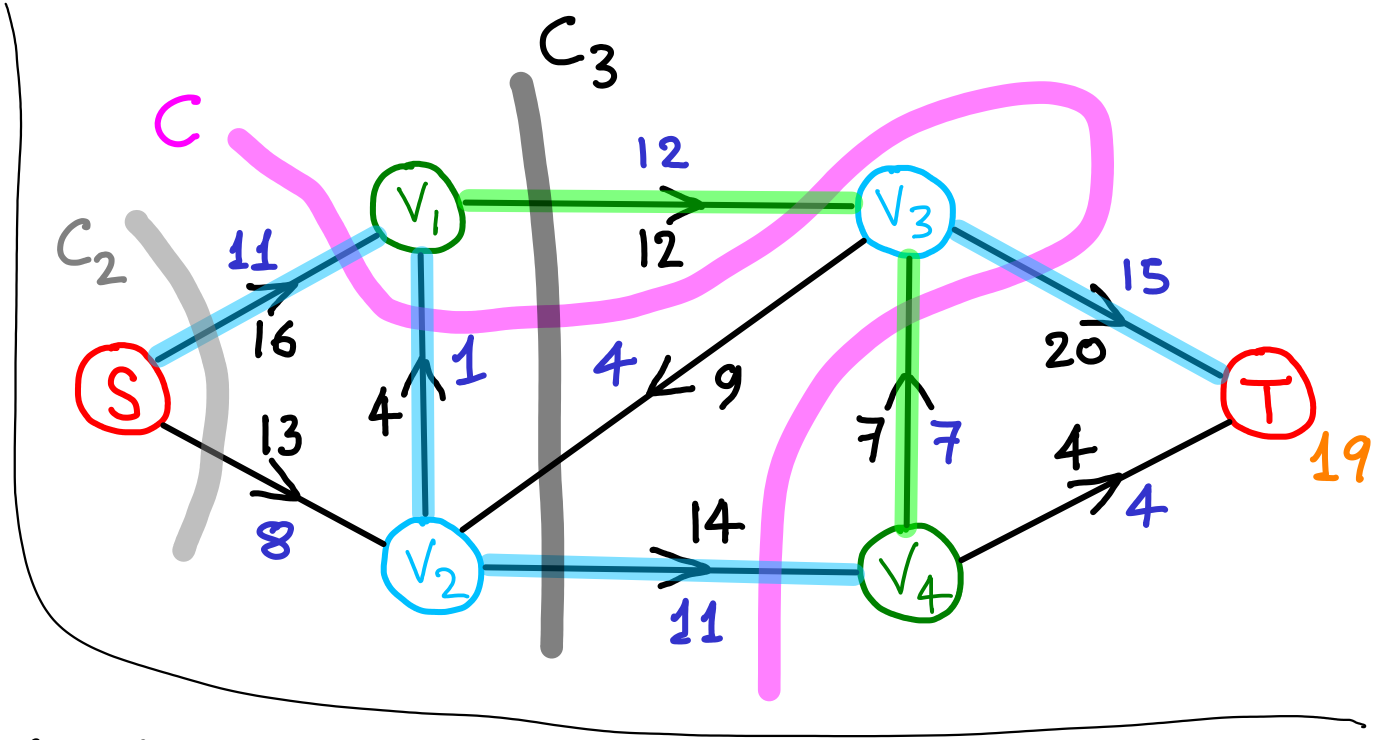
FLOW VALUE $\leq$ Capacity(C) ...

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

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

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

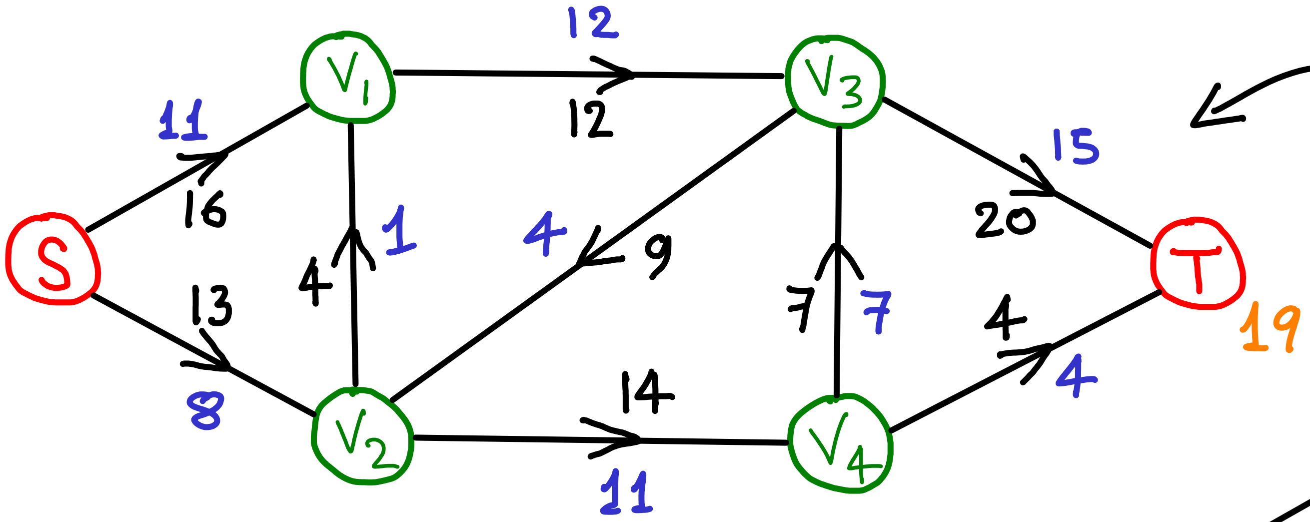
$$= 16+4+14+20 = 54$$



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

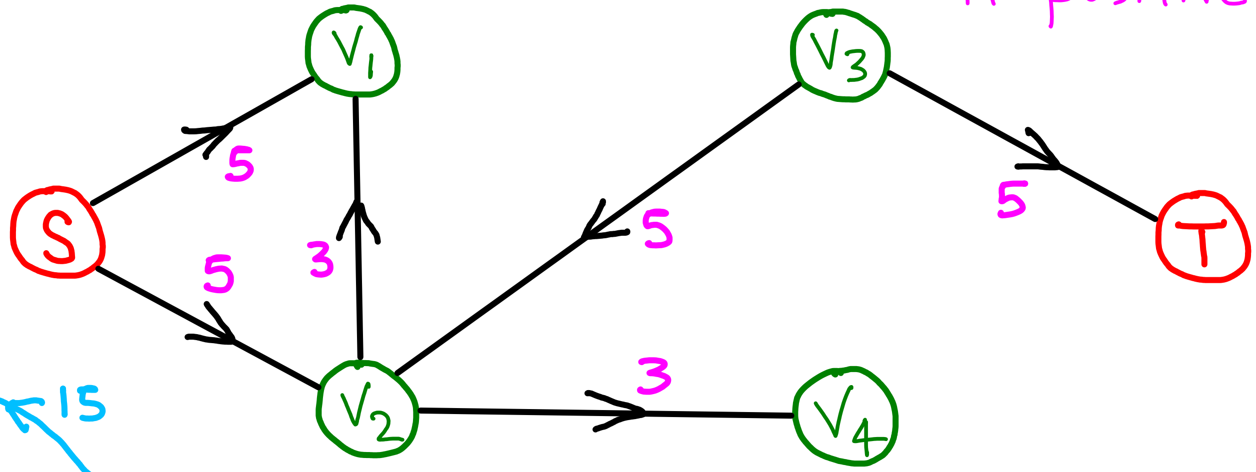
(ii) $\mathrm{Flow}(C) \leq \mathrm{Capacity}(C)$

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

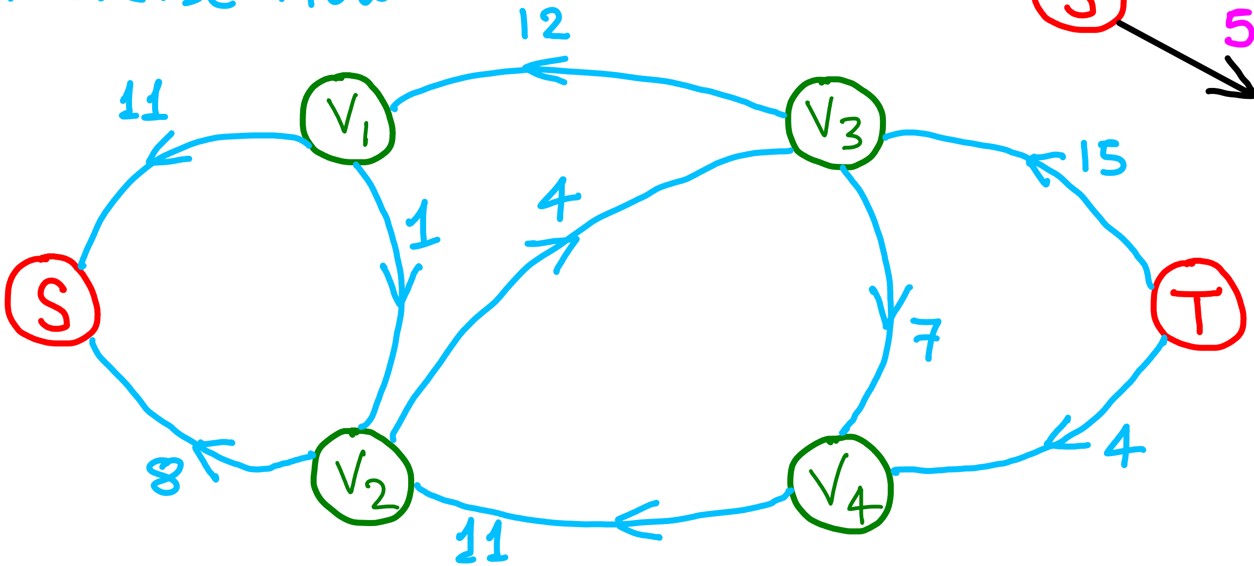


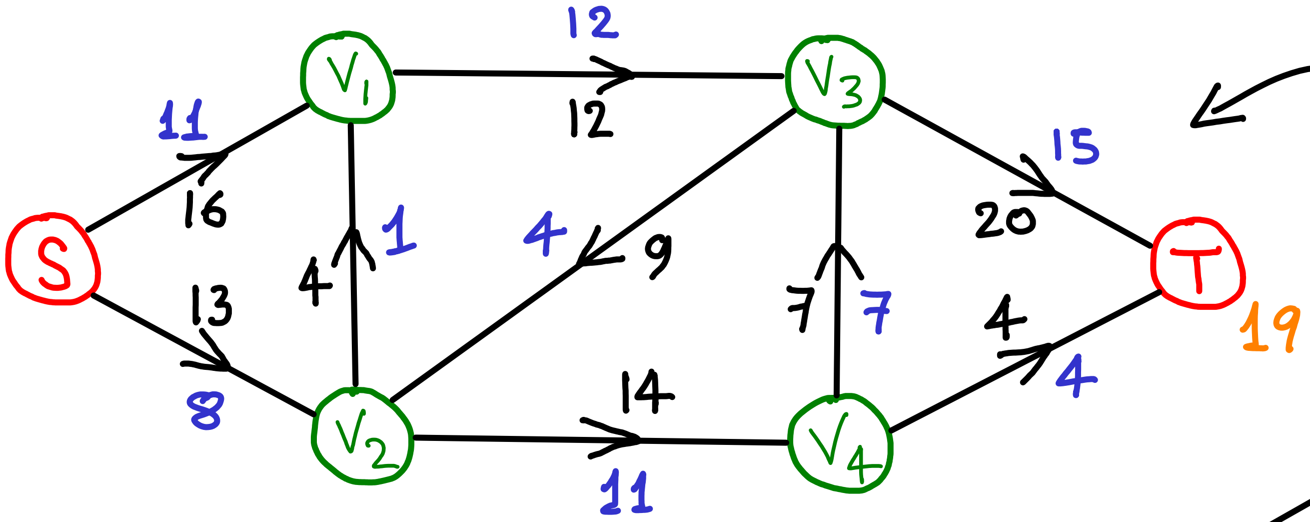
(Non-optimal)

Unused capacities,
if positive



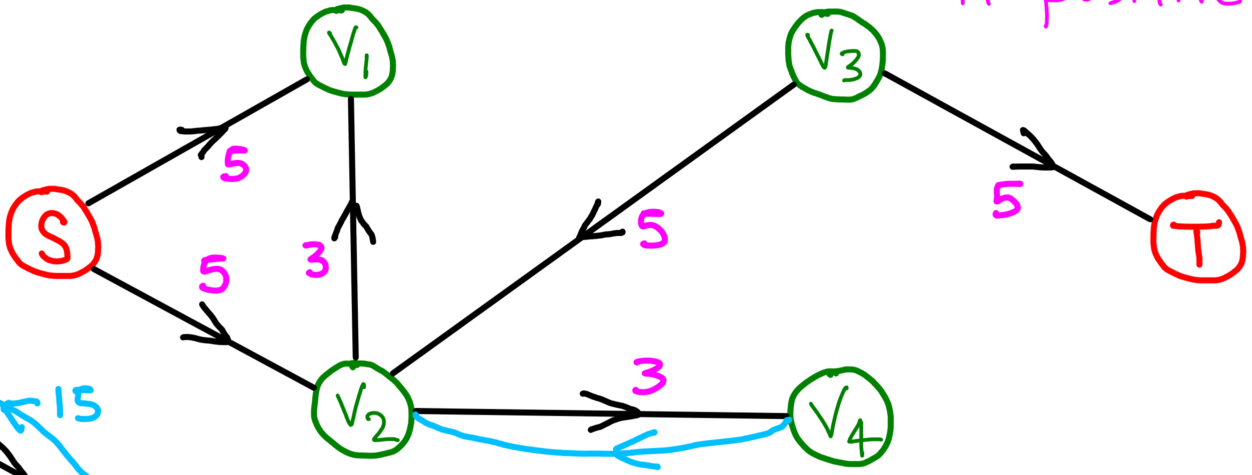
Reverse flow

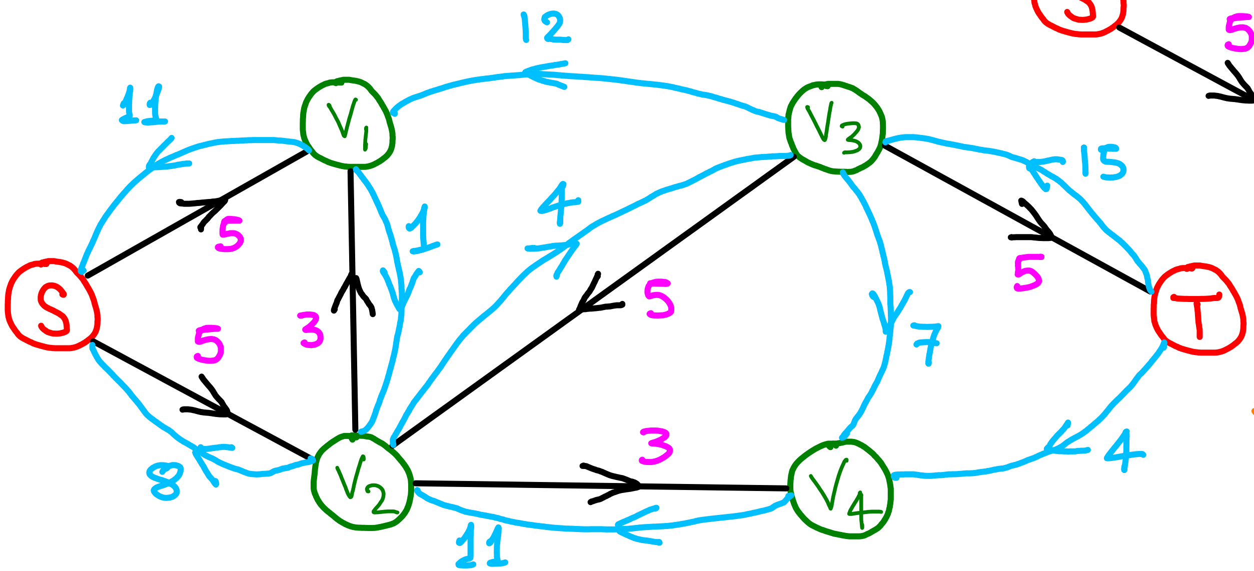




(Non-optimal)

Unused capacities, if positive



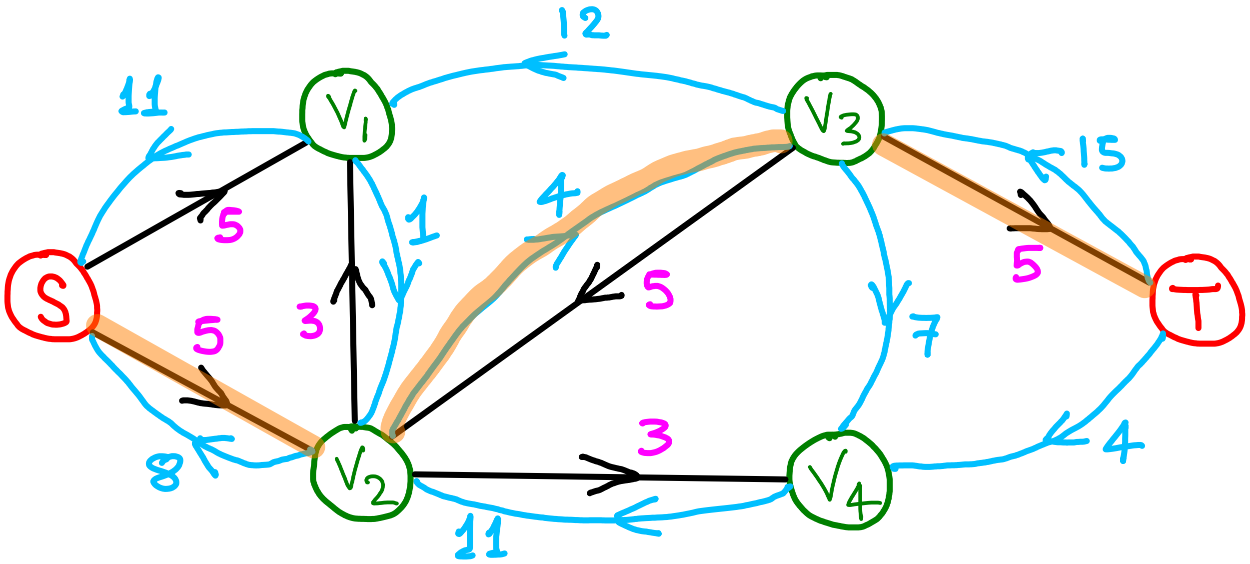


Residual network
(all potential flow changes)

Residual network → all potential flow changes

Blue edges = reverse flow = potential flow decrease

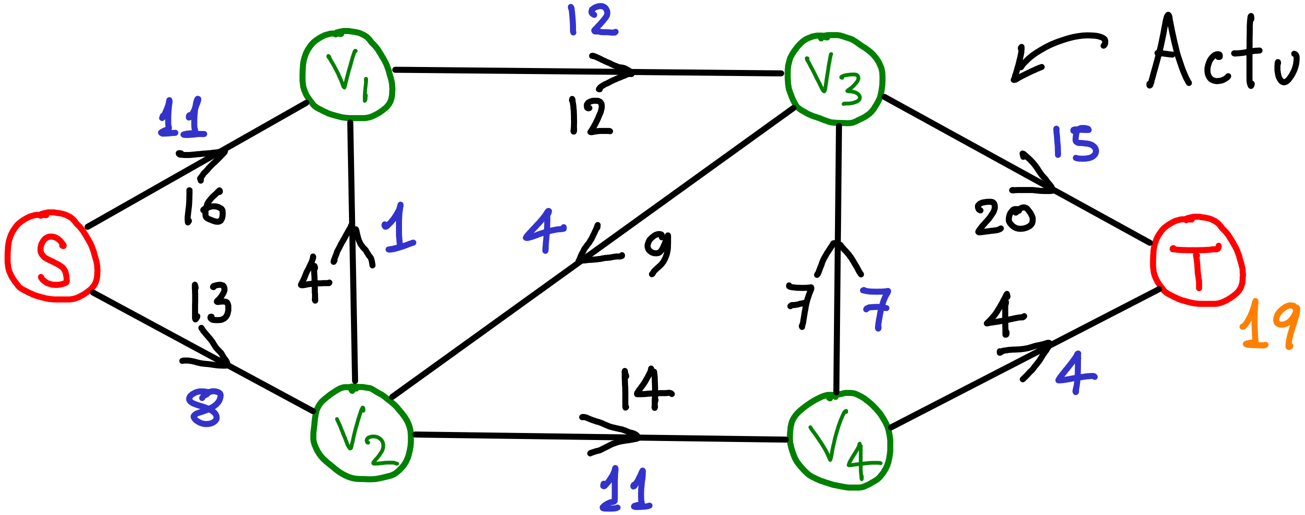
Black edges (pink labels) = potential flow increase



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.

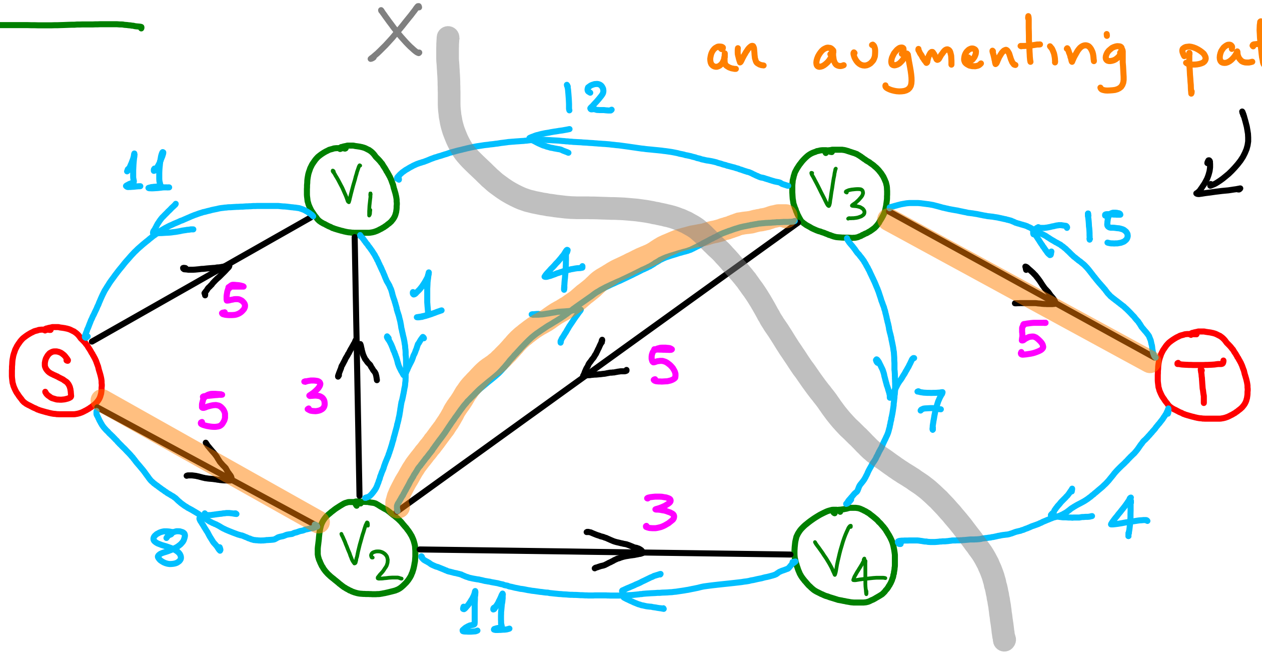
$S \to V_2 \to V_3 \to T$: AUGMENTING PATH



Actual network

Residual network &

an augmenting path

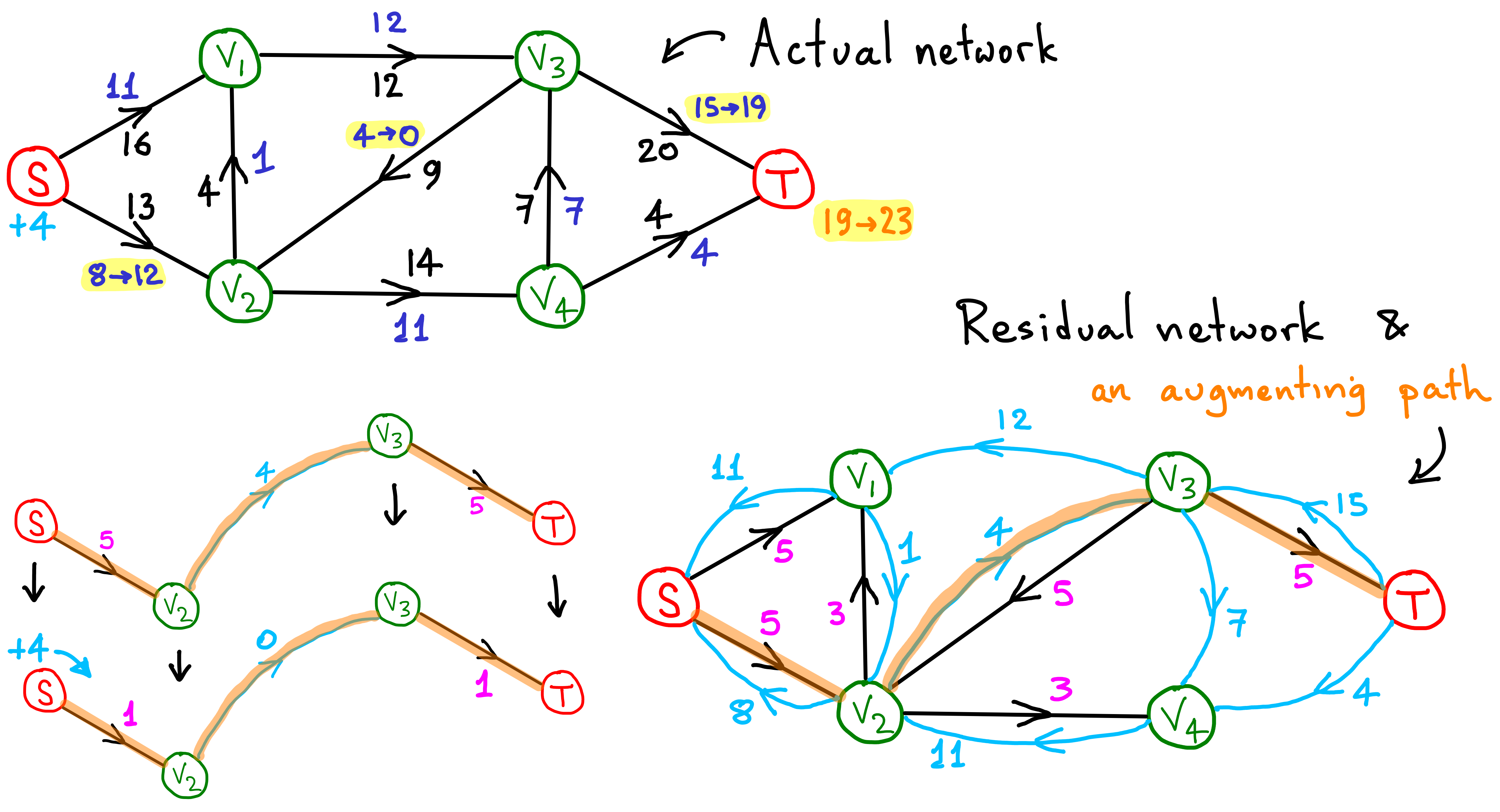


The residual network also has a MIN. cut X.

(min capacity of all cuts)

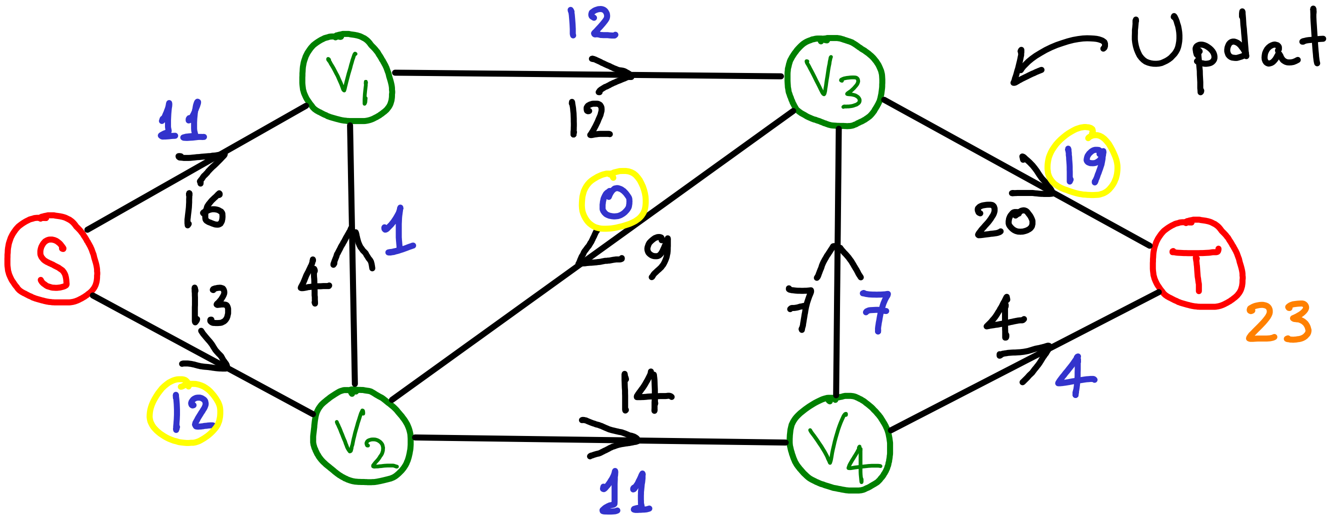
This gives a "residual capacity" (=4)

= max. increase in flow.

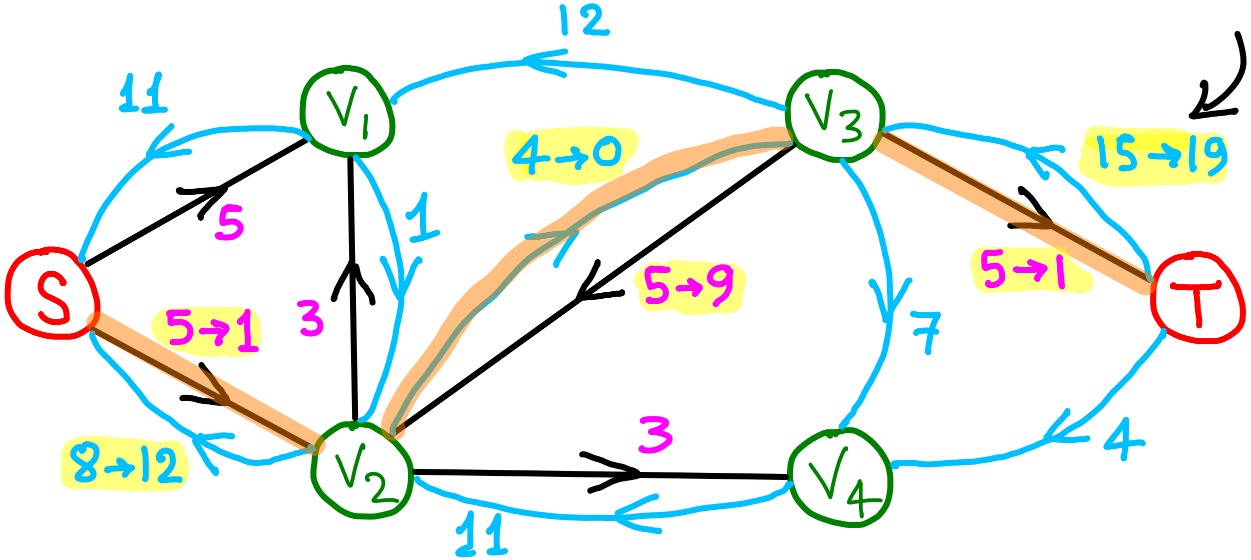


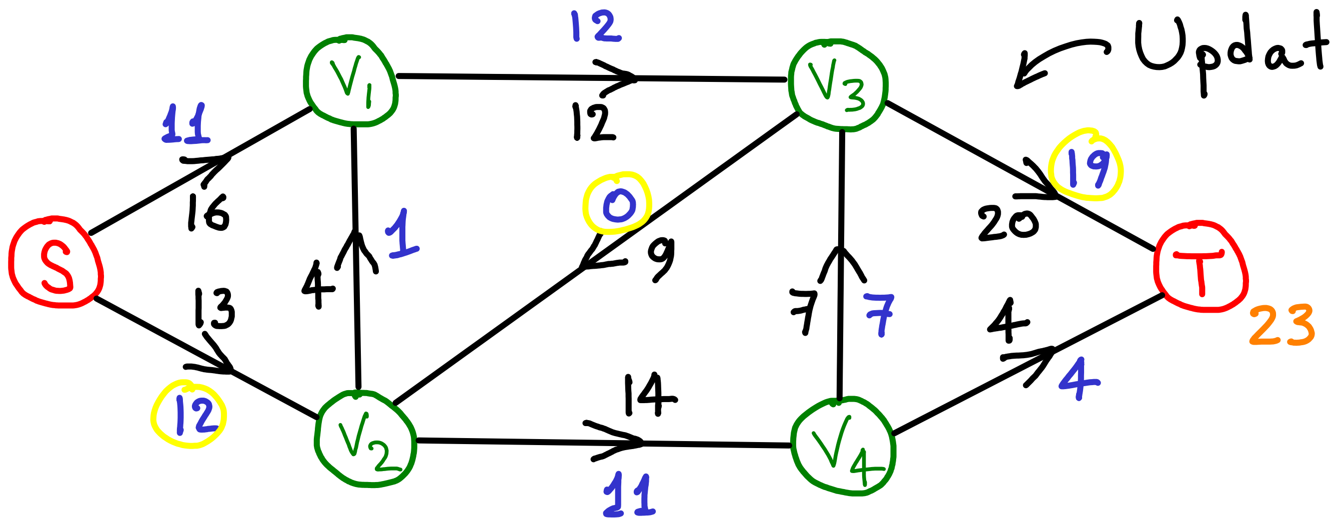
Actual network
Residual network &
an augmenting path

Updated flow



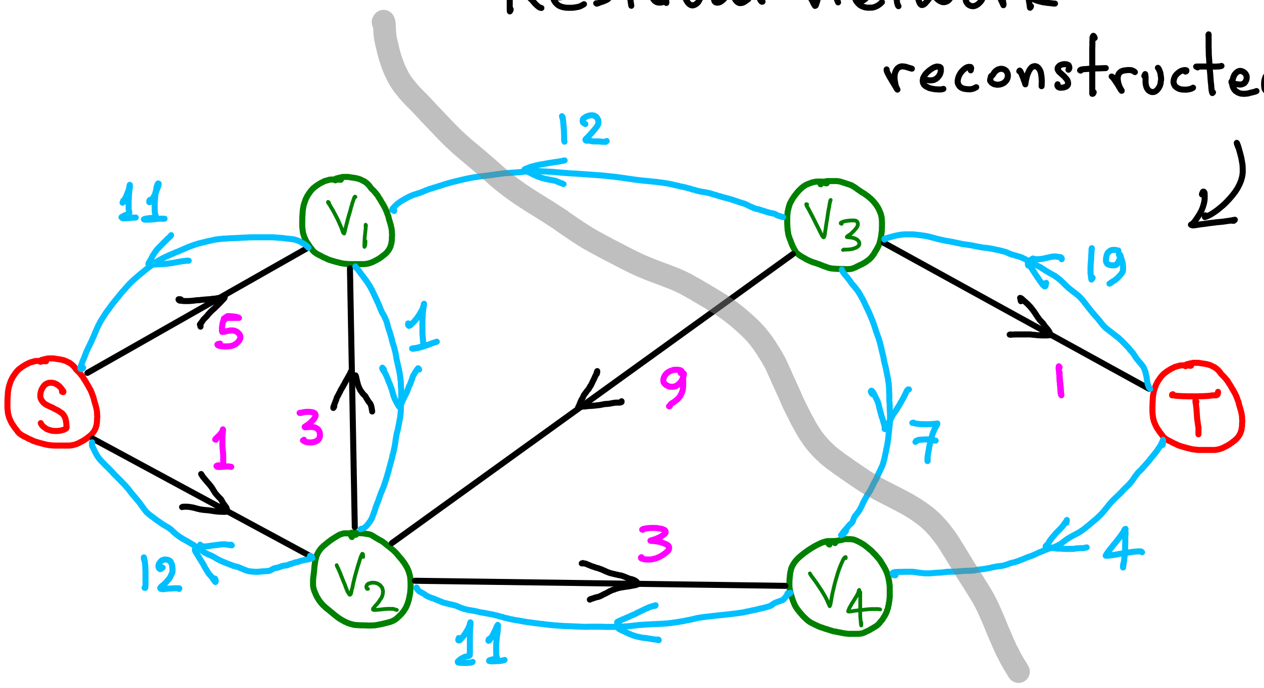
Residual network reconstructed





Updated flow

Residual network reconstructed



Residual capacity = 0

no augmenting paths

if ∃ augmenting path then flow can increase

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

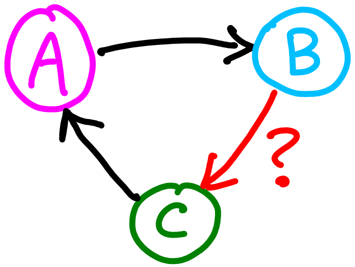
---

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



Implies MAX FLOW = MIN CUT

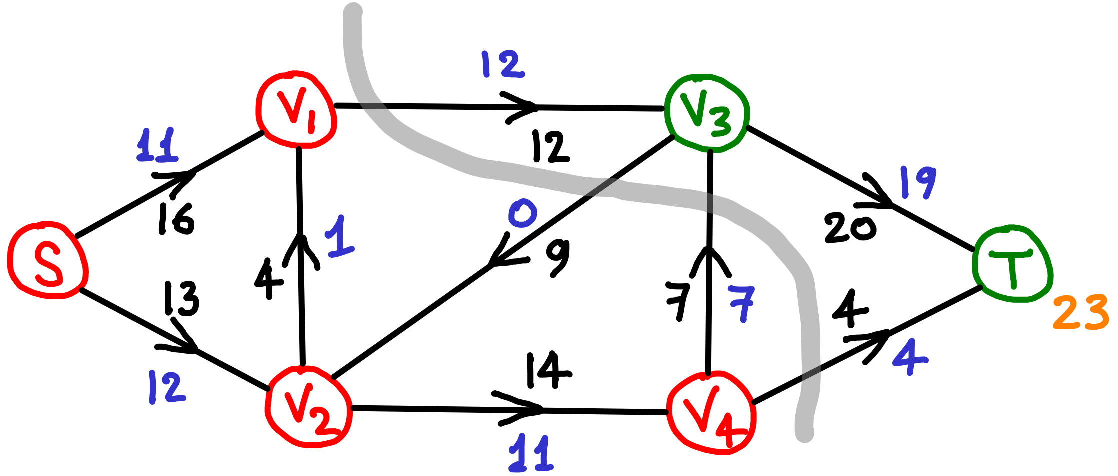
no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

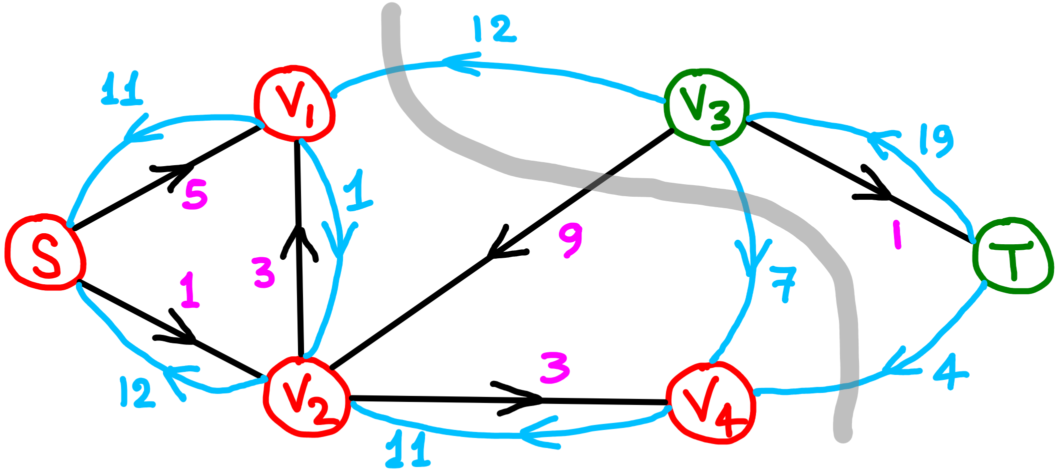
---

In residual network, let X: {vertices reachable from S}, Y = V - X

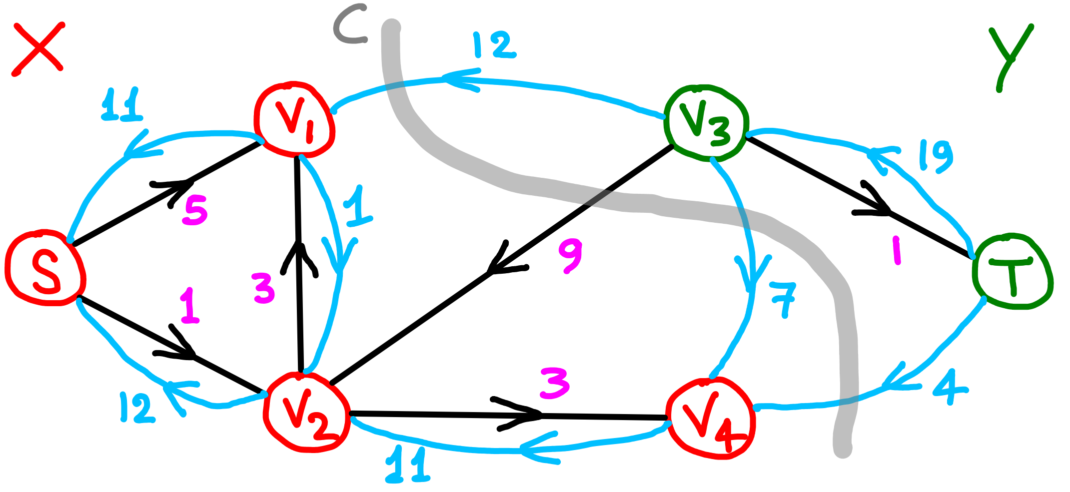
no augmenting path → T is in Y

↳ X & Y define a cut, C



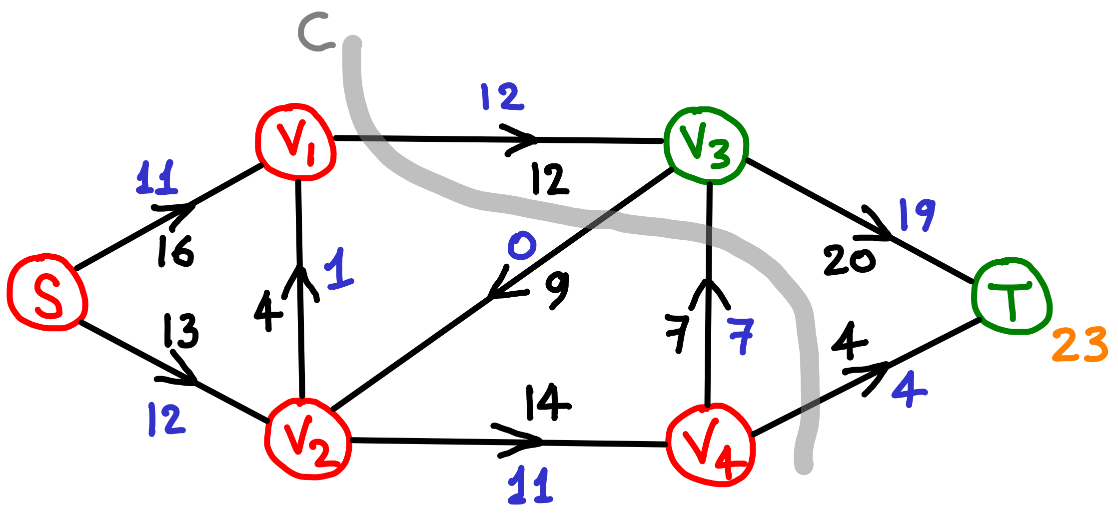


no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut

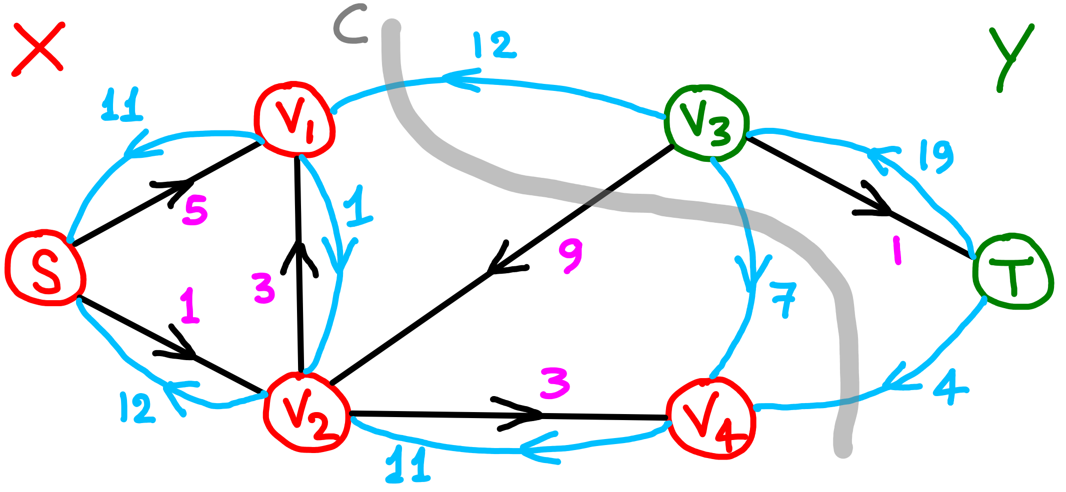


Look at any edge $\overline{pq}$ crossing C in flow network.

- $\overrightarrow{pq}$ is not in residual.
  (otherwise q would be in X)



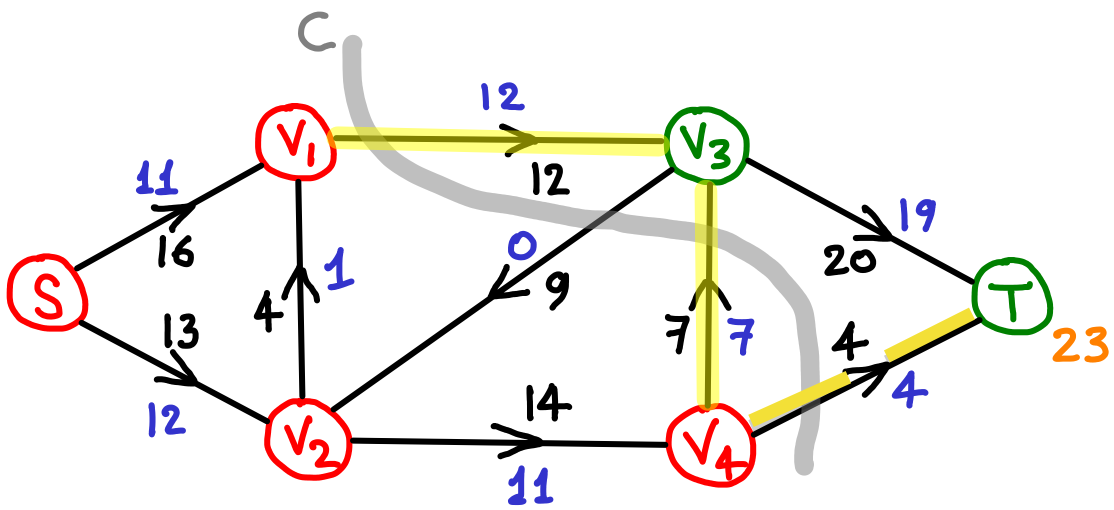
no augmenting path $\xrightarrow{?}$ flow equals capacity of some cut



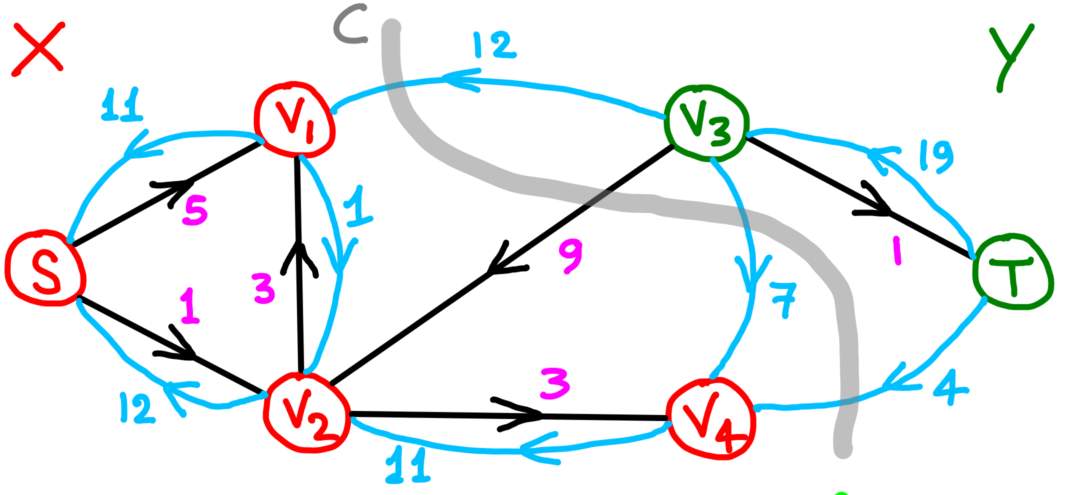
Look at any edge $\overline{pq}$ crossing C in flow network.

- $\overrightarrow{pq}$ is not in residual.
  (otherwise q would be in X)

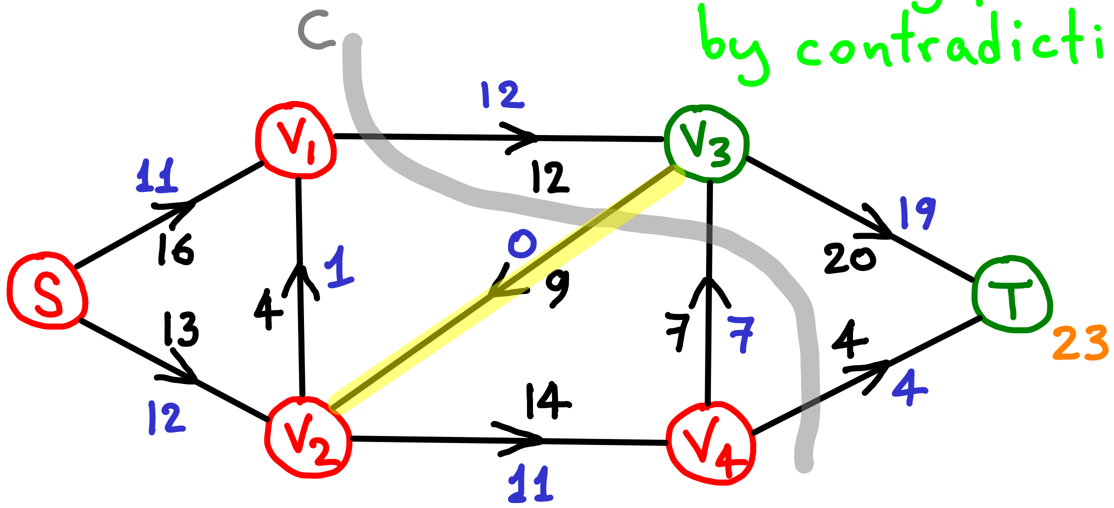
  → if $\overrightarrow{pq}$ in flow network then $\text{Flow}(\overrightarrow{pq}) = \text{Capacity}(\overrightarrow{pq})$



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



easy proof
by contradiction



Look at any edge $\overline{pq}$ crossing C in flow network.

- $\overrightarrow{pq}$ is not in residual.
  (otherwise q would be in X)

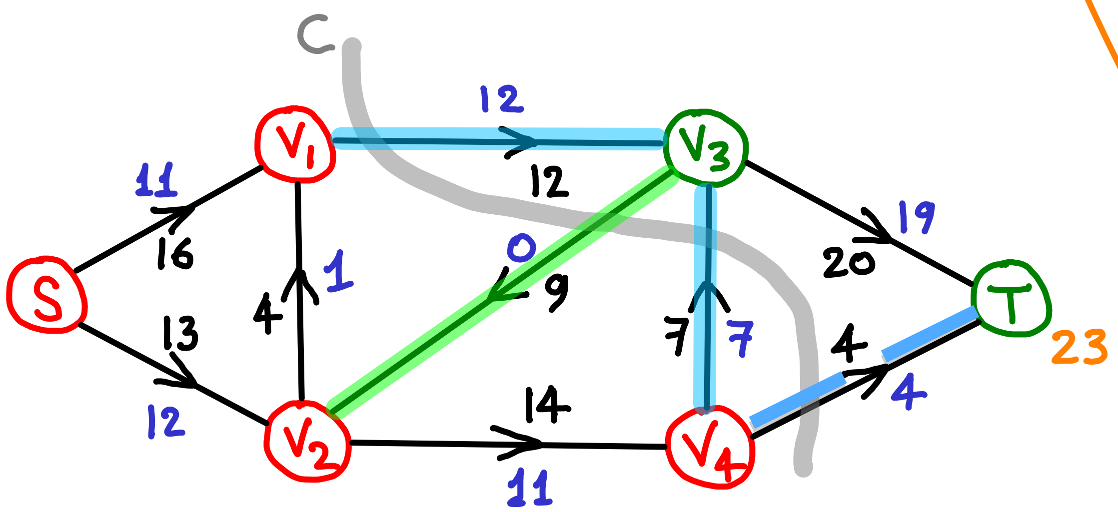
→ if $\overrightarrow{pq}$ in flow network
then $\text{Flow}(\overrightarrow{pq}) = \text{Capacity}(\overrightarrow{pq})$

→ if $\overleftarrow{pq}$ in flow network
then $\text{Flow}(\overleftarrow{pq}) = 0$

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

---

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



For edges crossing $C(X,Y)$

if $\overrightarrow{pq}$ in flow network
then $\text{Flow}(\overrightarrow{pq}) = \text{Capacity}(\overrightarrow{pq})$

if $\overleftarrow{pq}$ in flow network
then $\text{Flow}(\overleftarrow{pq}) = 0$

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 Flow($\vec{pq}$)
    else subtract w from Flow($\overleftarrow{pq}$)

O(path length)

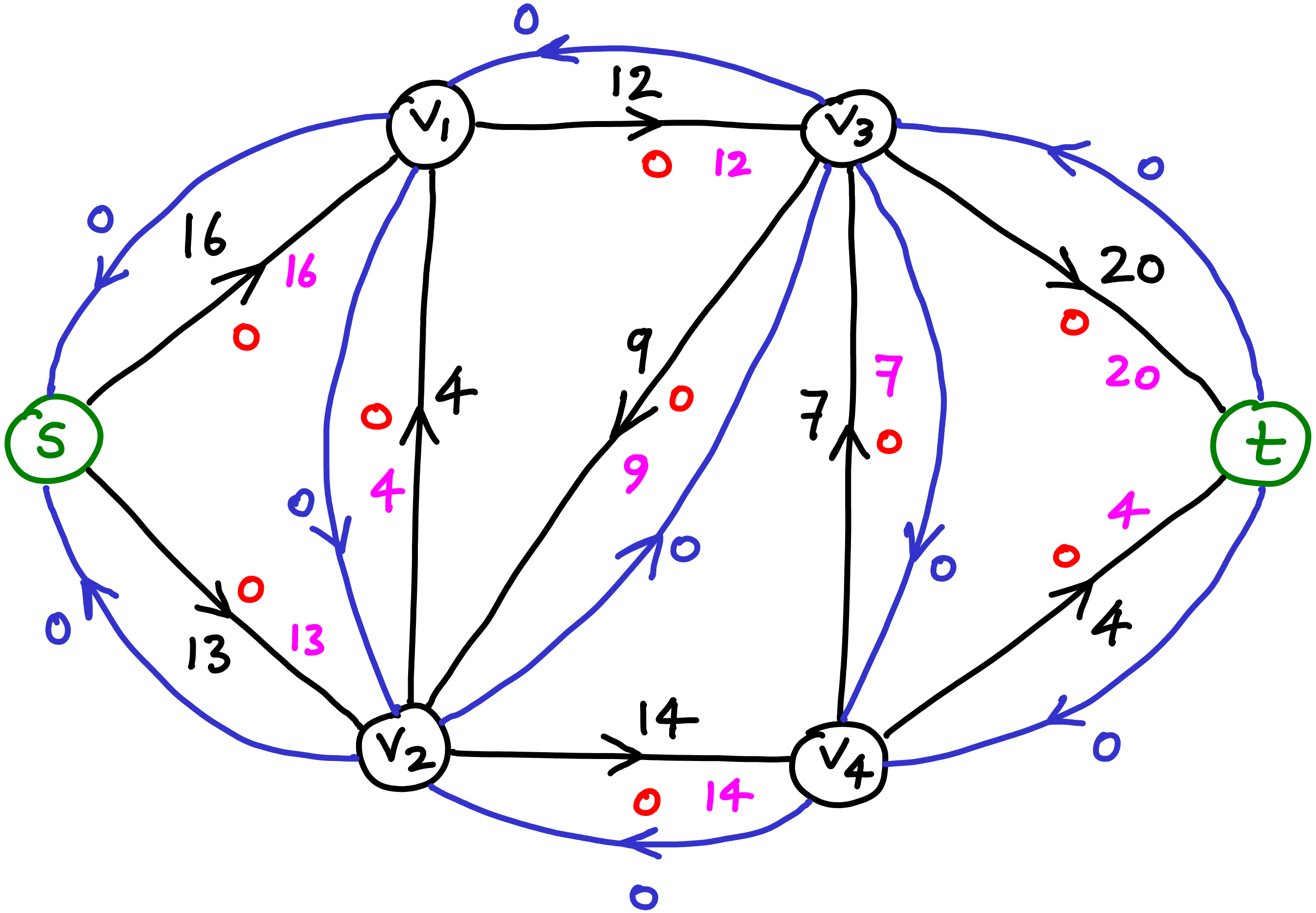
forms basis of more advanced algorithms (e.g. Edmonds-Karp)

\# : capacity (normal edge)

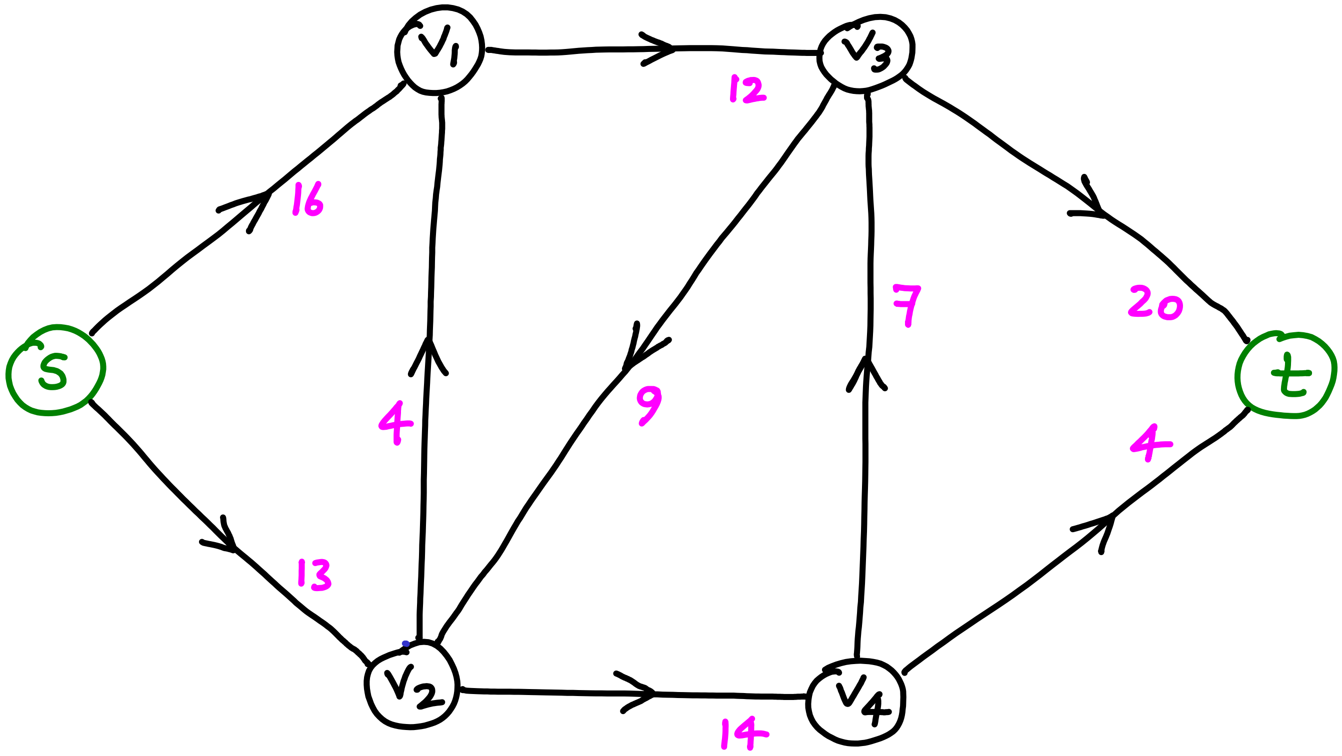
\# : flow

\# : leftover (residual edge)

\# : reverse (residual edge)



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



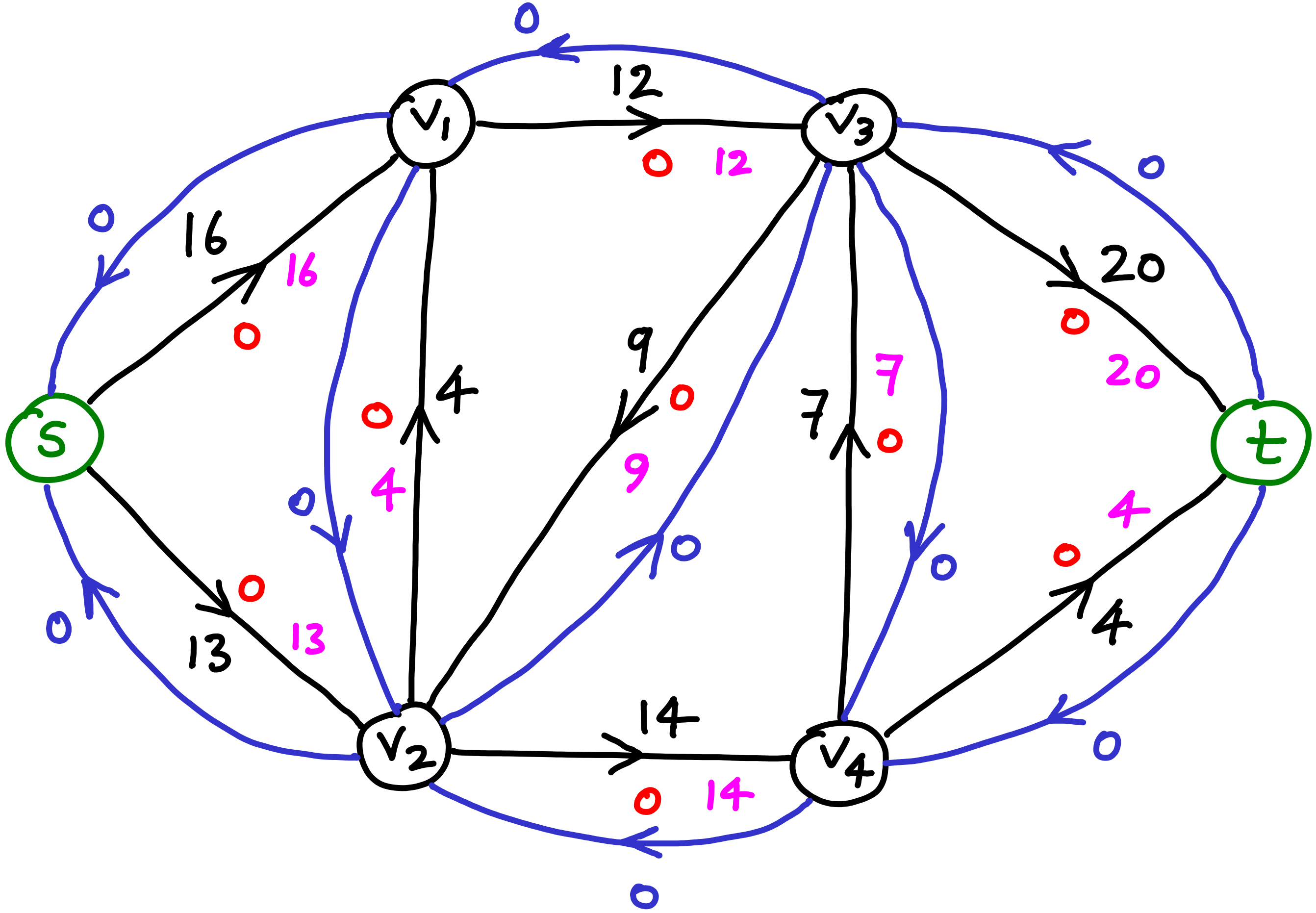
Residual graph

#: capacity (normal edge)

#: flow

#: leftover (residual edge)

#: reverse (residual edge)



#: capacity (normal edge)

#: flow

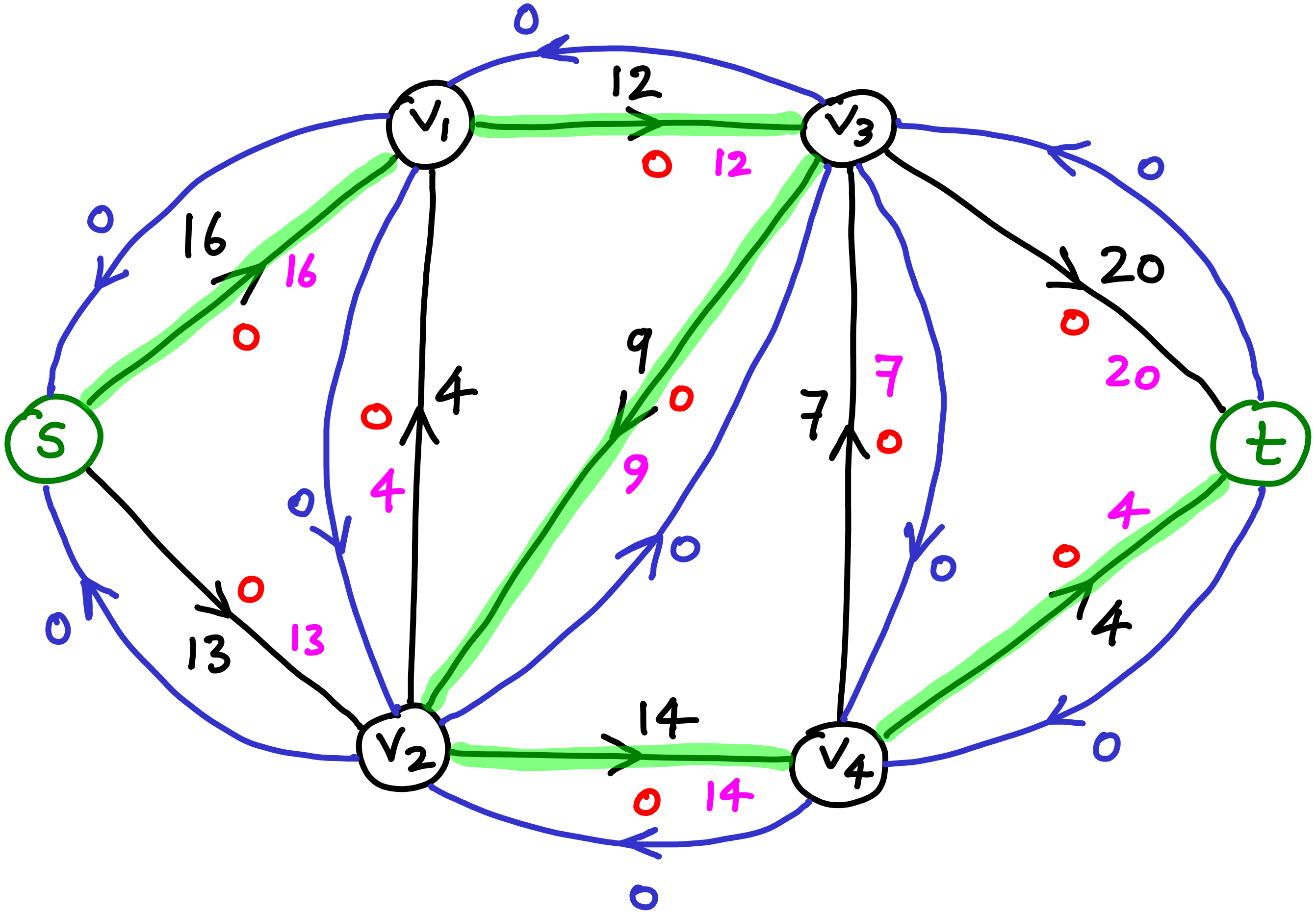
#: leftover (residual edge)

#: reverse (residual edge)

↝: augmenting path

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

16 12 9 14 4



#: capacity (normal edge)

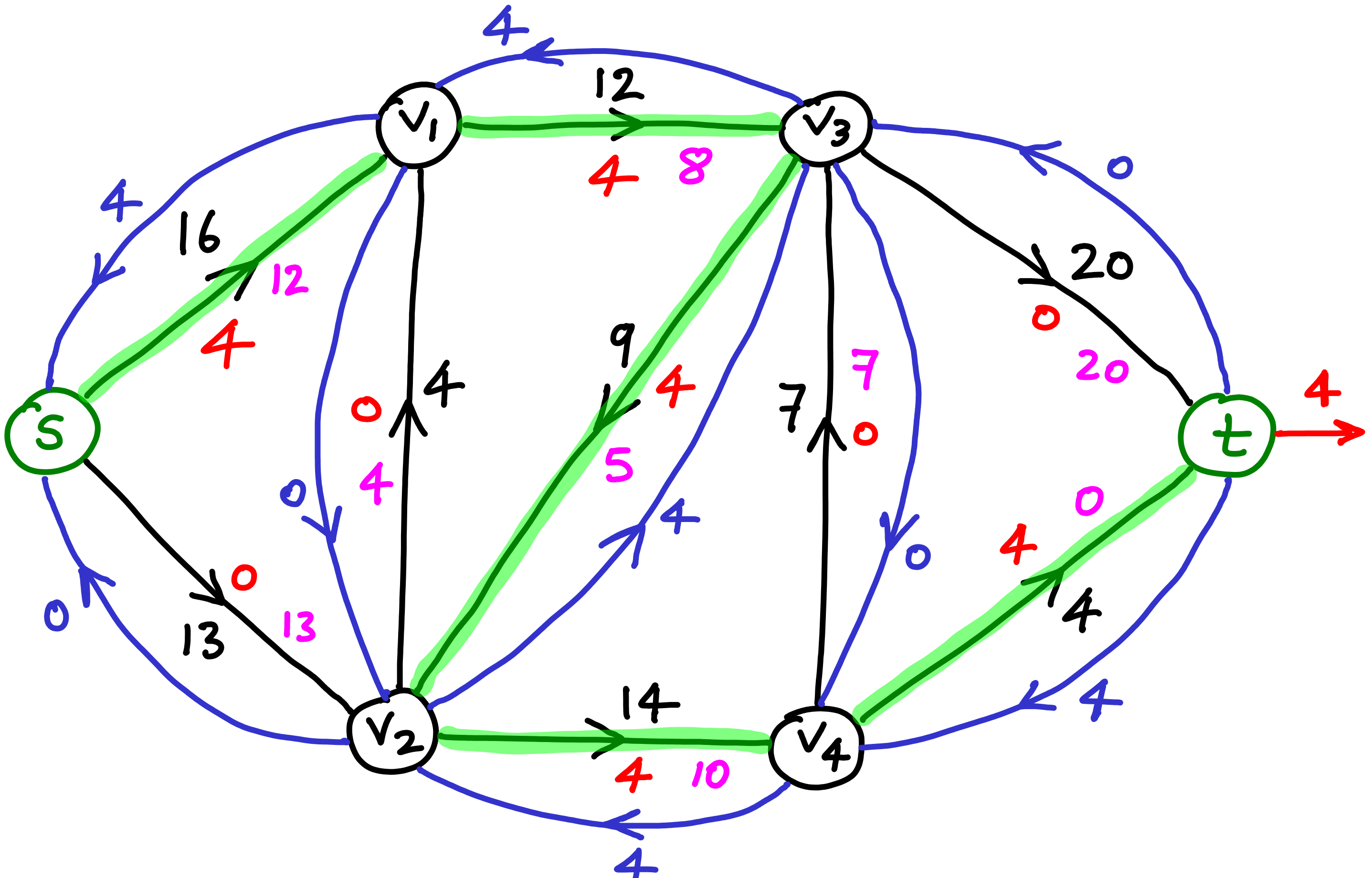
#: flow

#: leftover (residual edge)

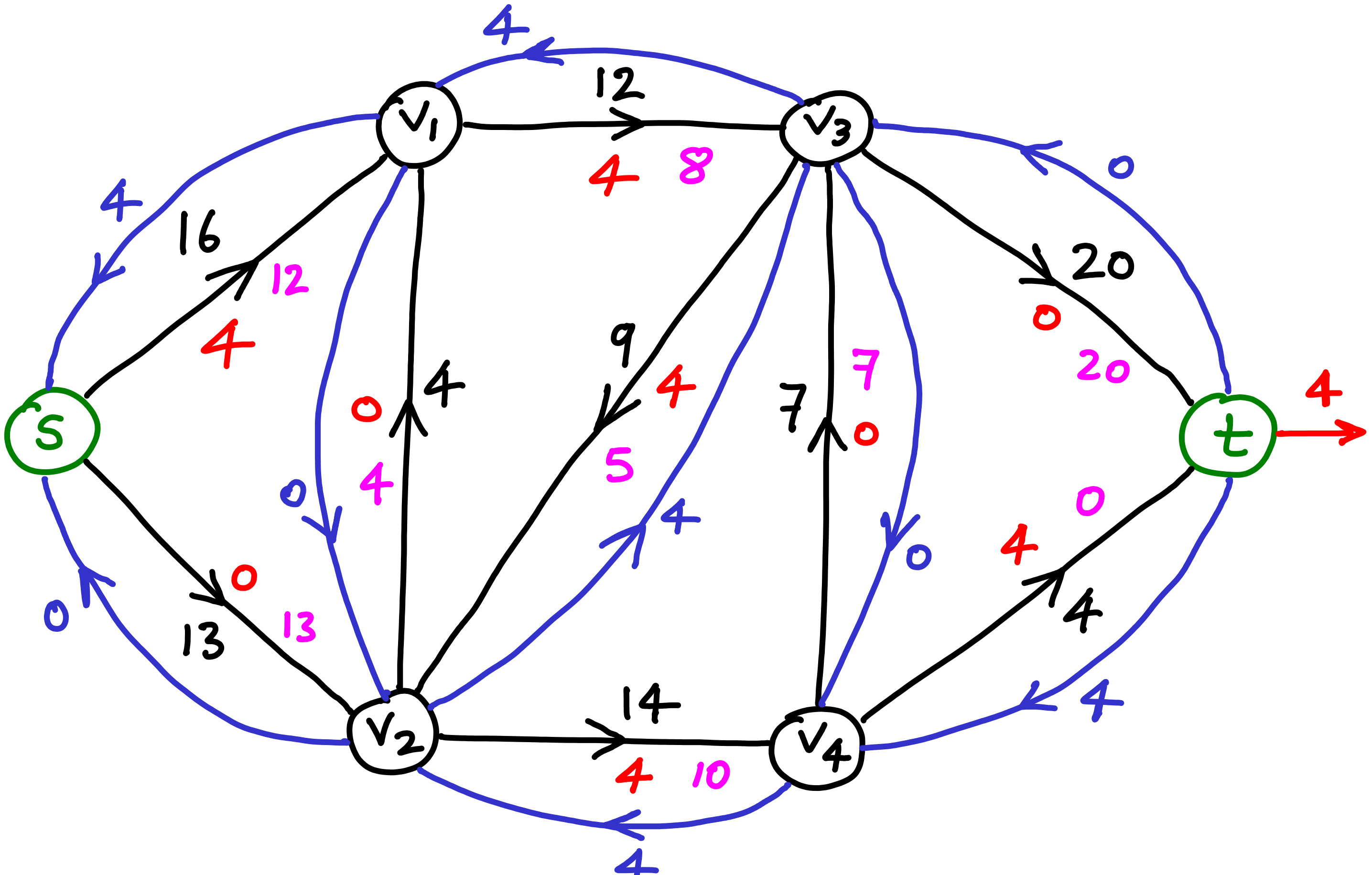
#: reverse (residual edge)

⤳: augmenting path

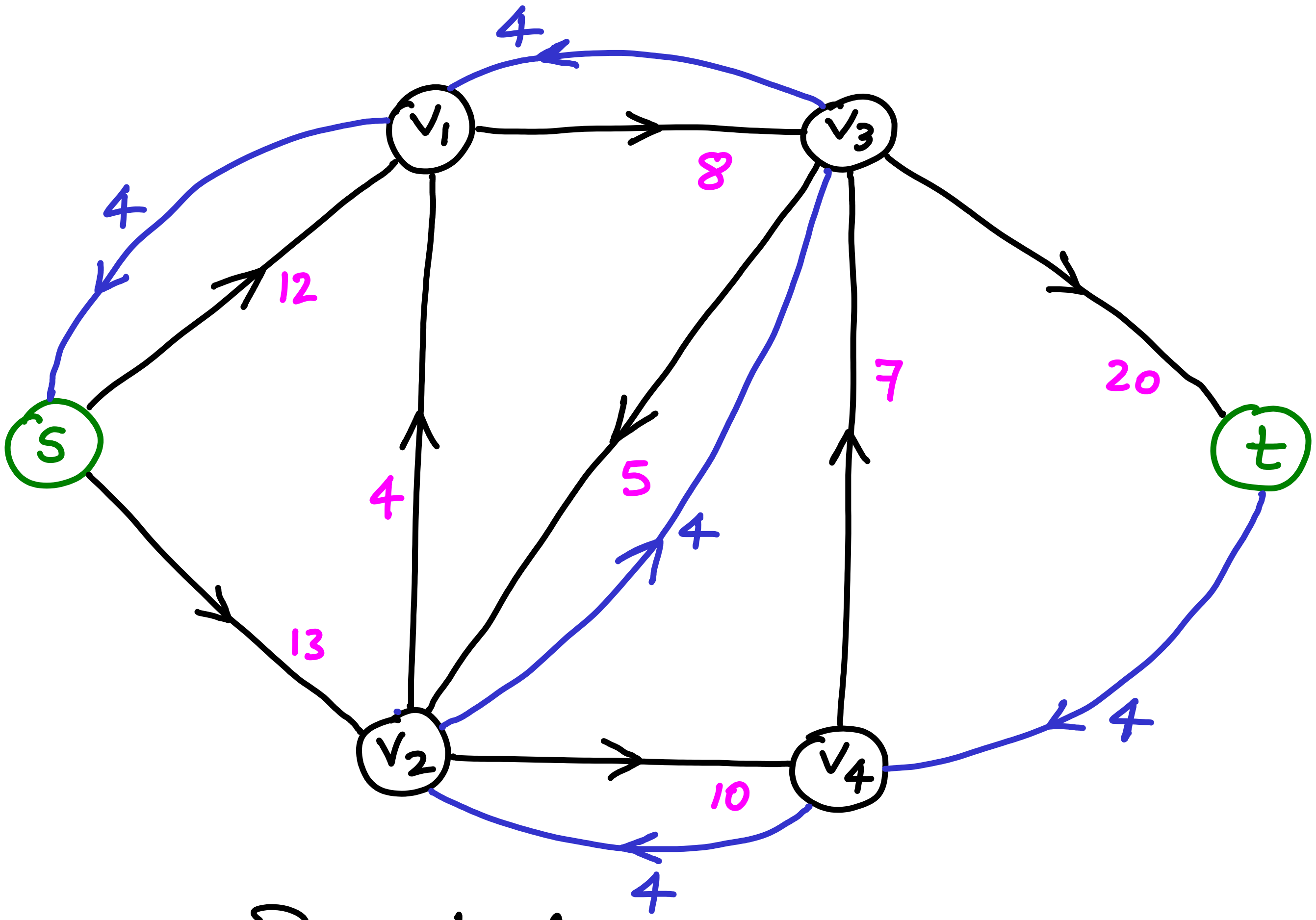
$s \xrightarrow[16]{} v_1 \xrightarrow[12]{} v_3 \xrightarrow[9]{} v_2 \xrightarrow[14]{} v_4 \xrightarrow[4]{} t$



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



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



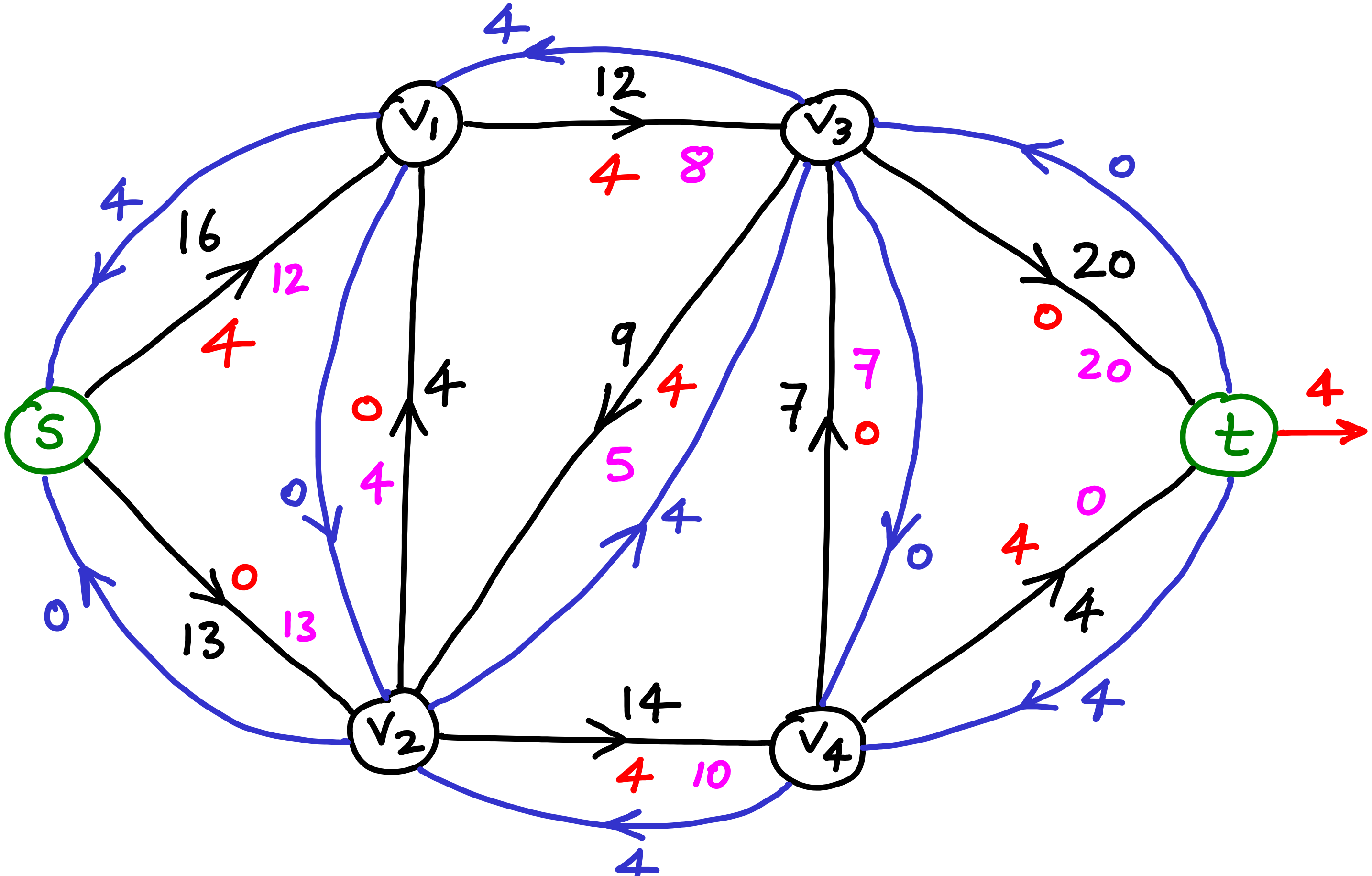
Residual graph

#: capacity (normal edge)

#: flow

#: leftover (residual edge)

#: reverse (residual edge)



#: capacity (normal edge)

#: flow

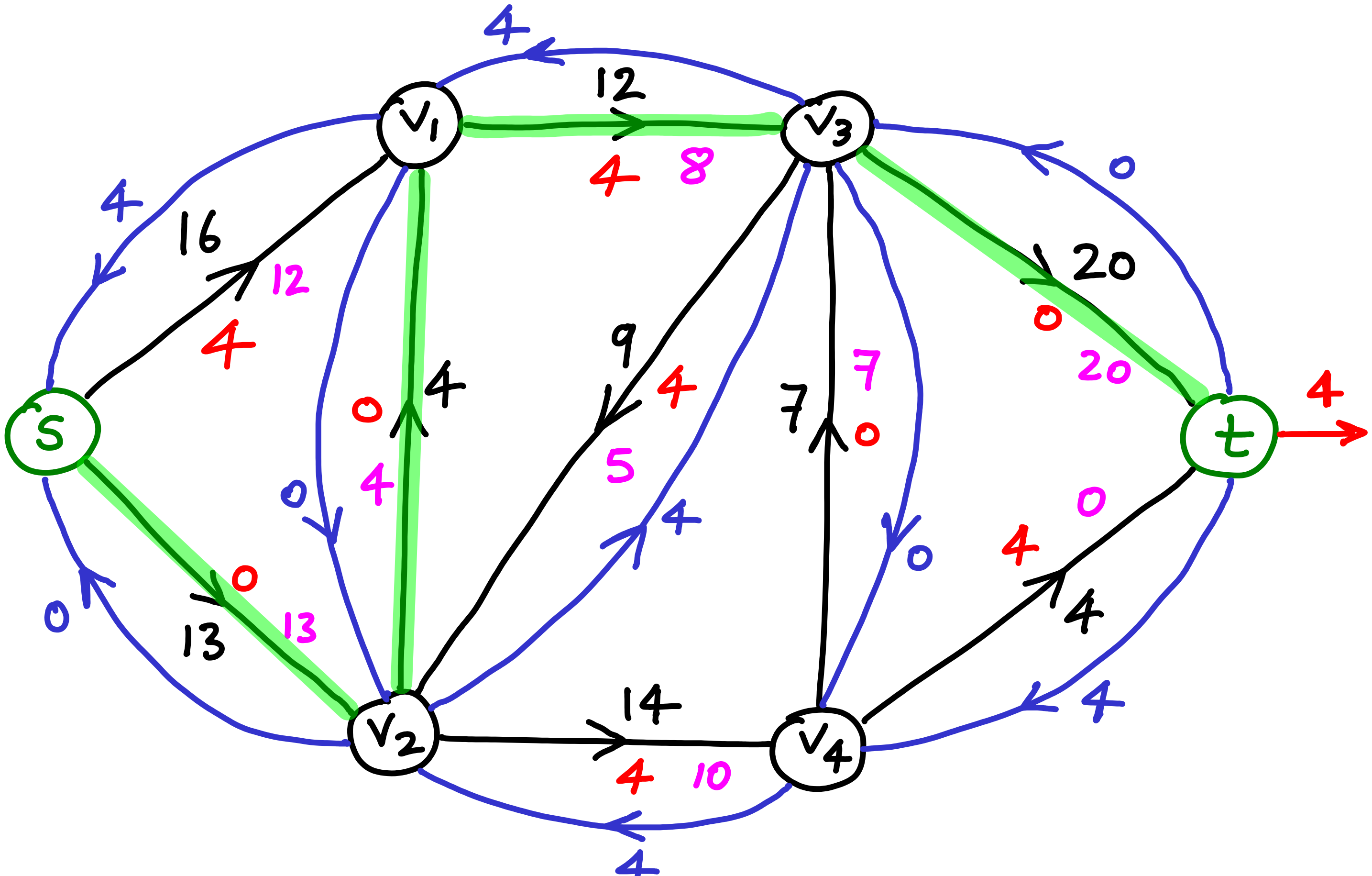
#: leftover (residual edge)

#: reverse (residual edge)

~>: augmenting path

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

13 4 8 20



#: capacity (normal edge)

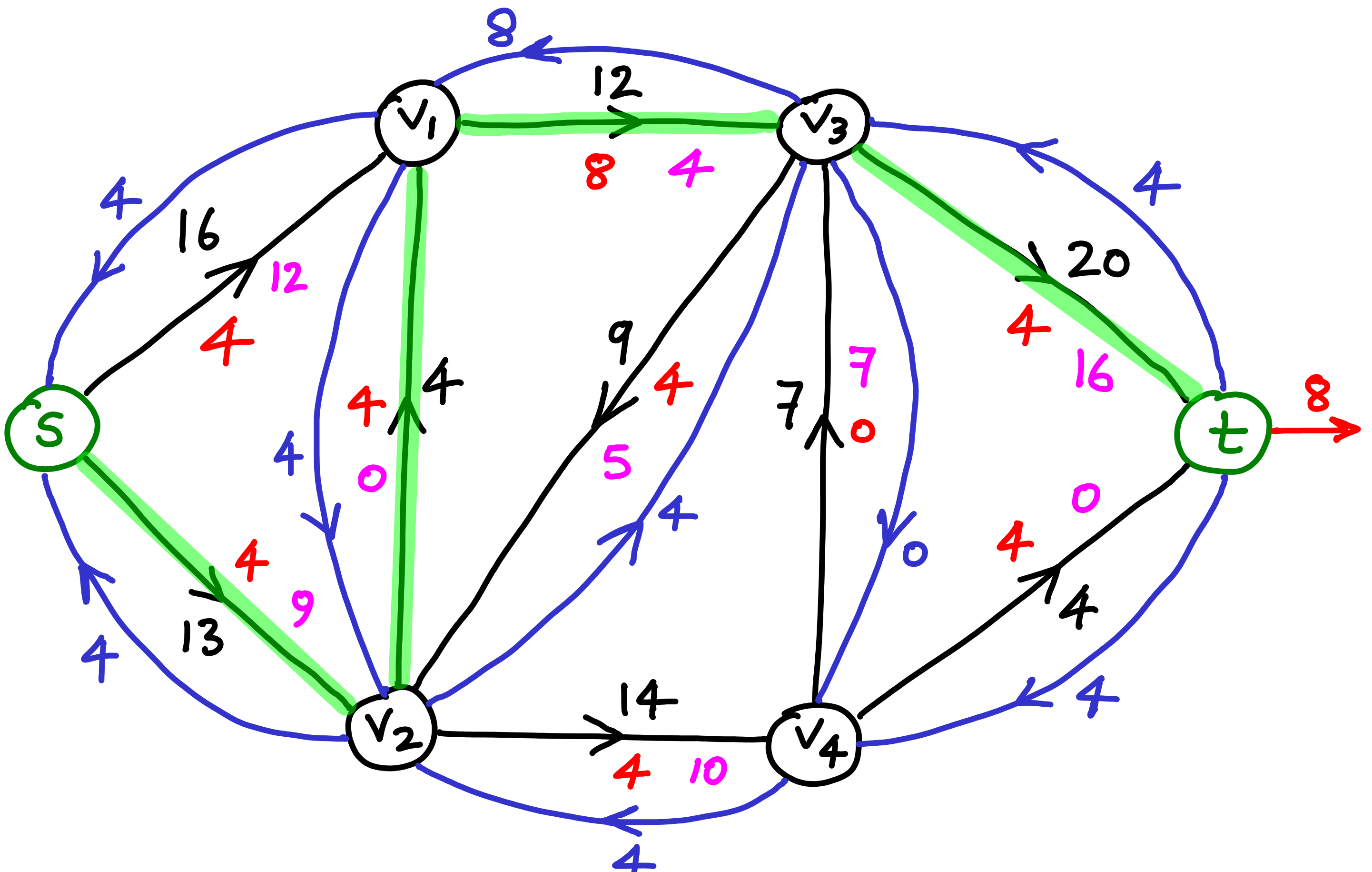
#: flow

#: leftover (residual edge)

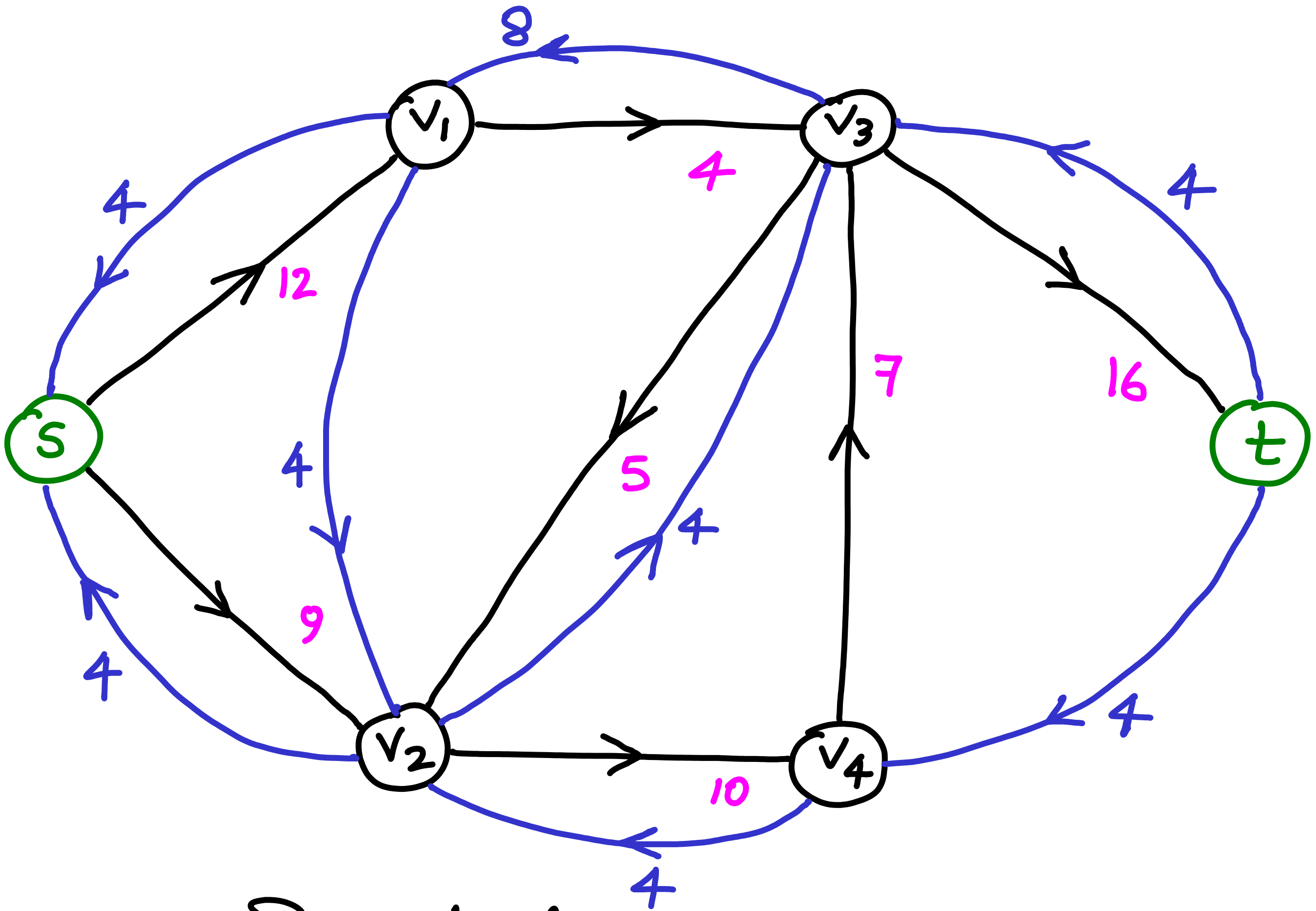
#: reverse (residual edge)

~>: augmenting path

$$s \xrightarrow[13]{} v_2 \xrightarrow[4]{} v_1 \xrightarrow[8]{} v_3 \xrightarrow[20]{} t$$



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



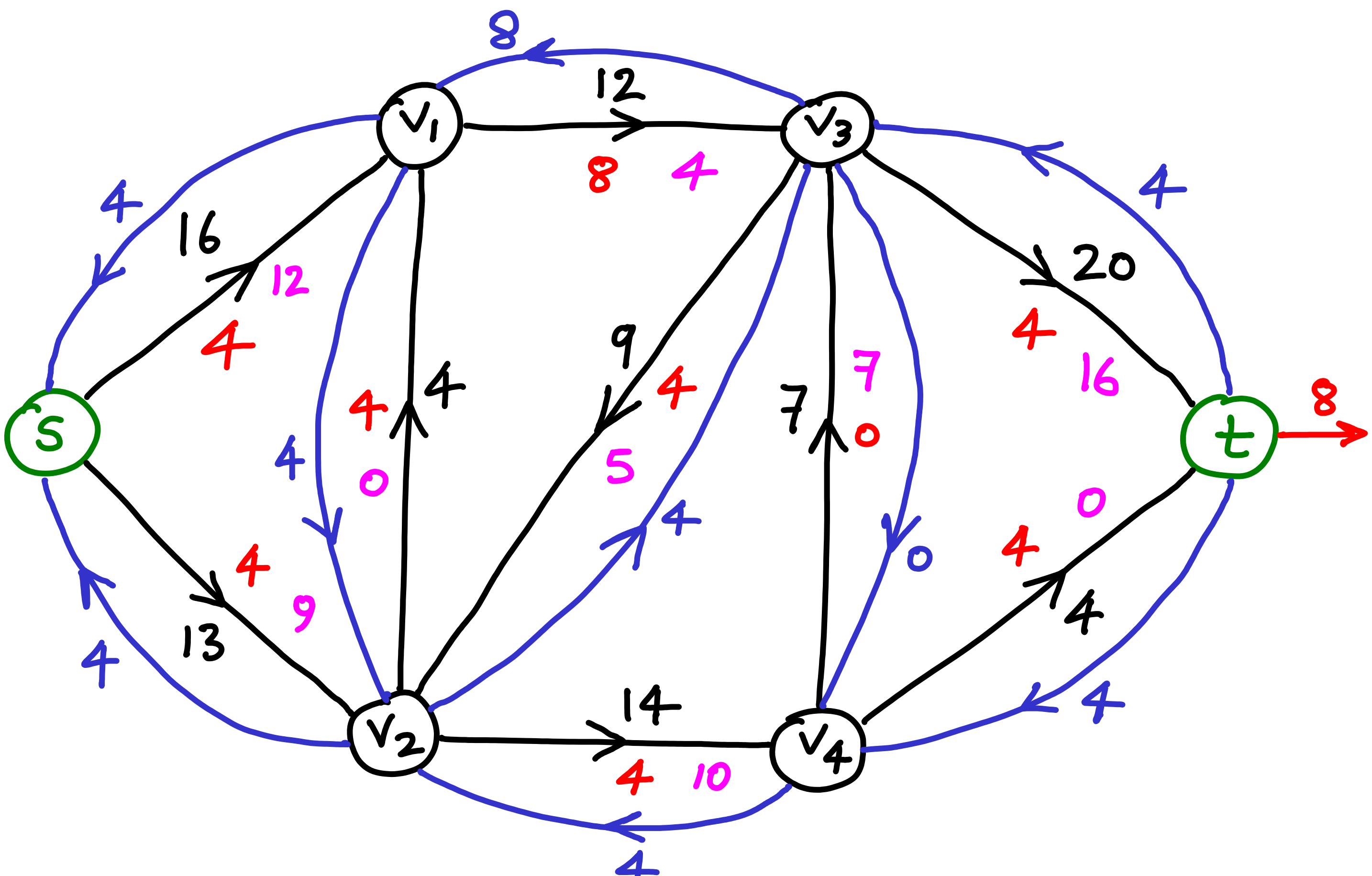
Residual graph

\# : capacity (normal edge)

\# : flow

\# : leftover (residual edge)

\# : reverse (residual edge)



#: capacity (normal edge)

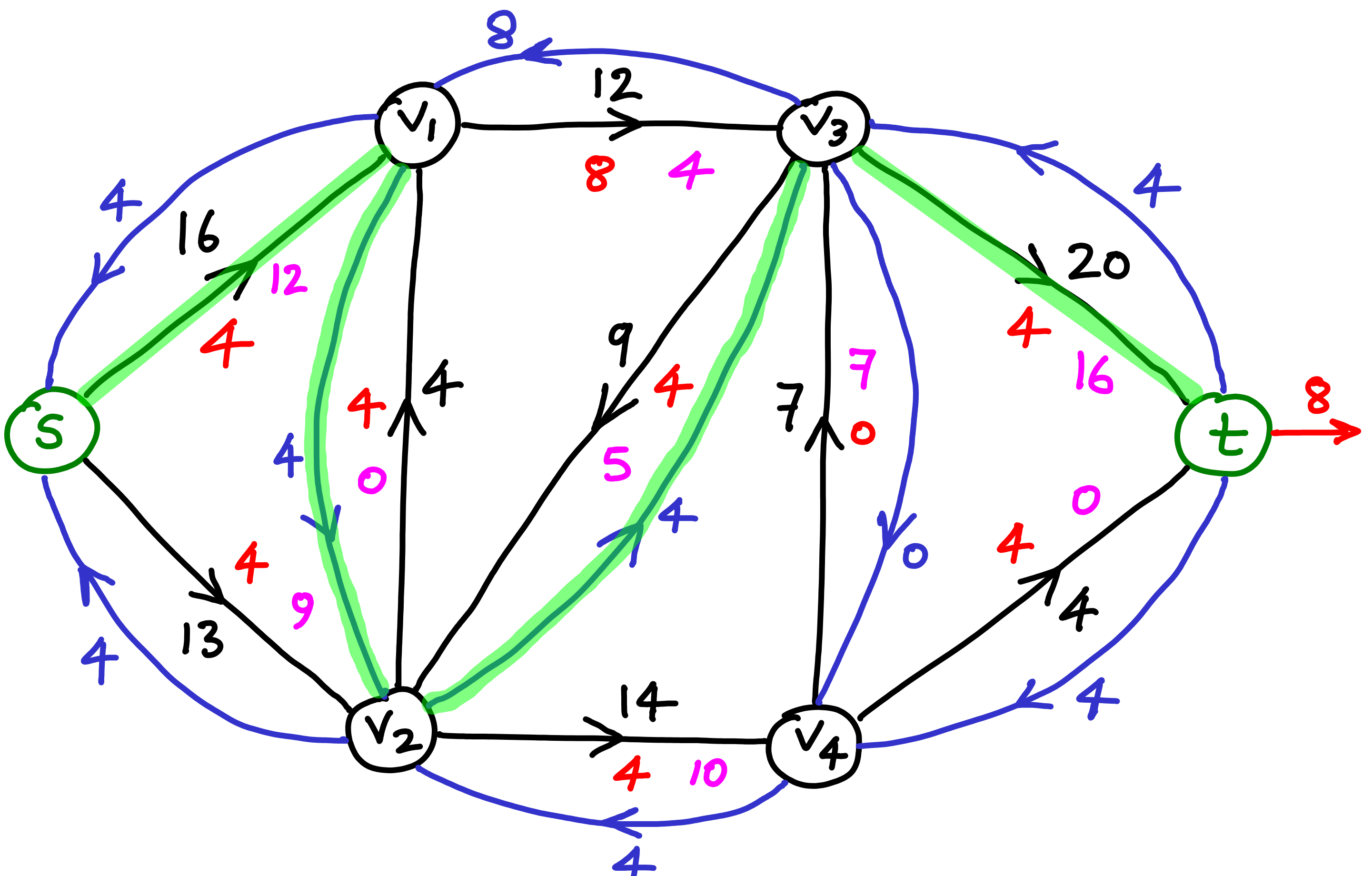
#: flow

#: 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)

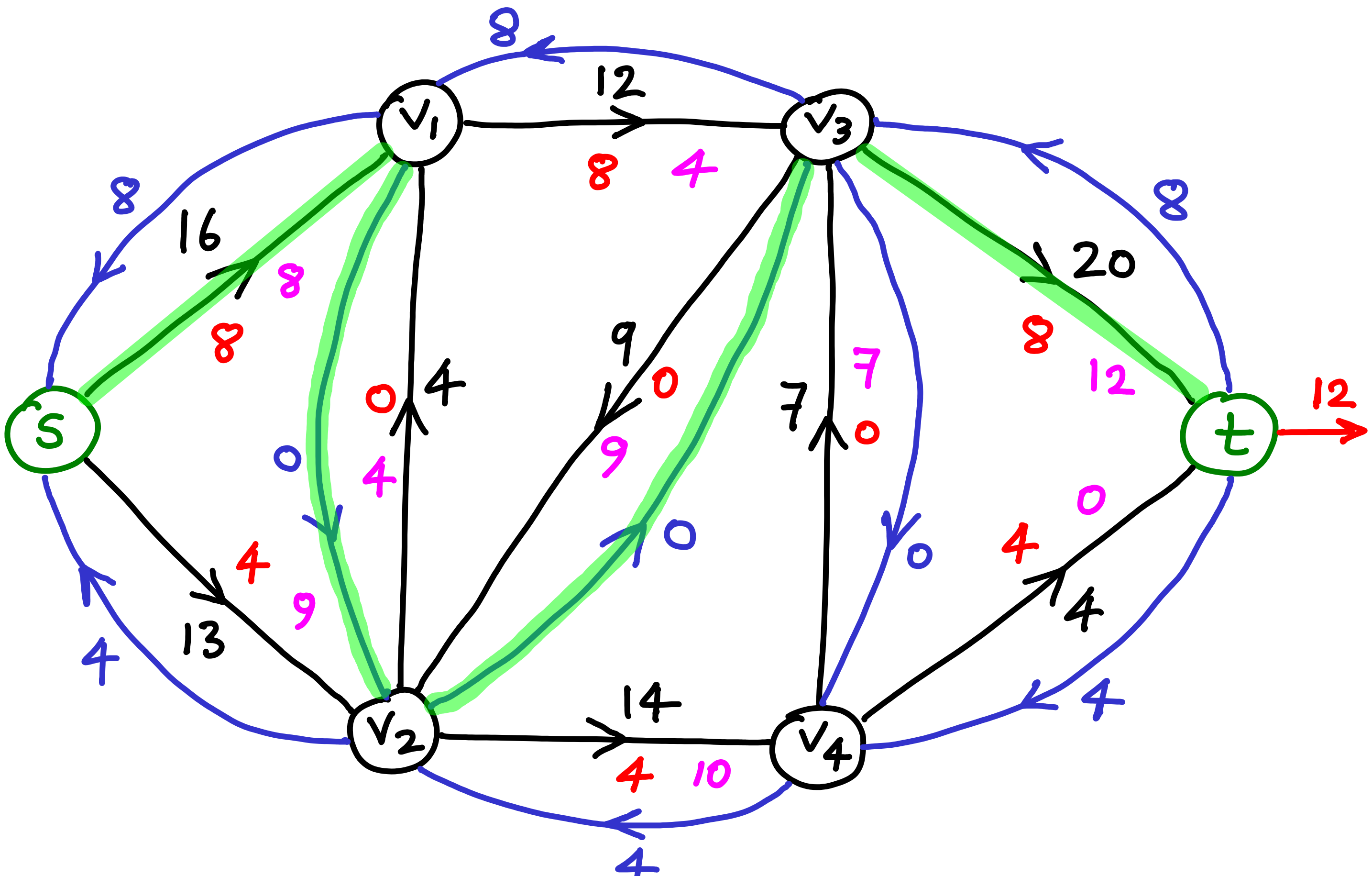
#: flow

#: leftover (residual edge)

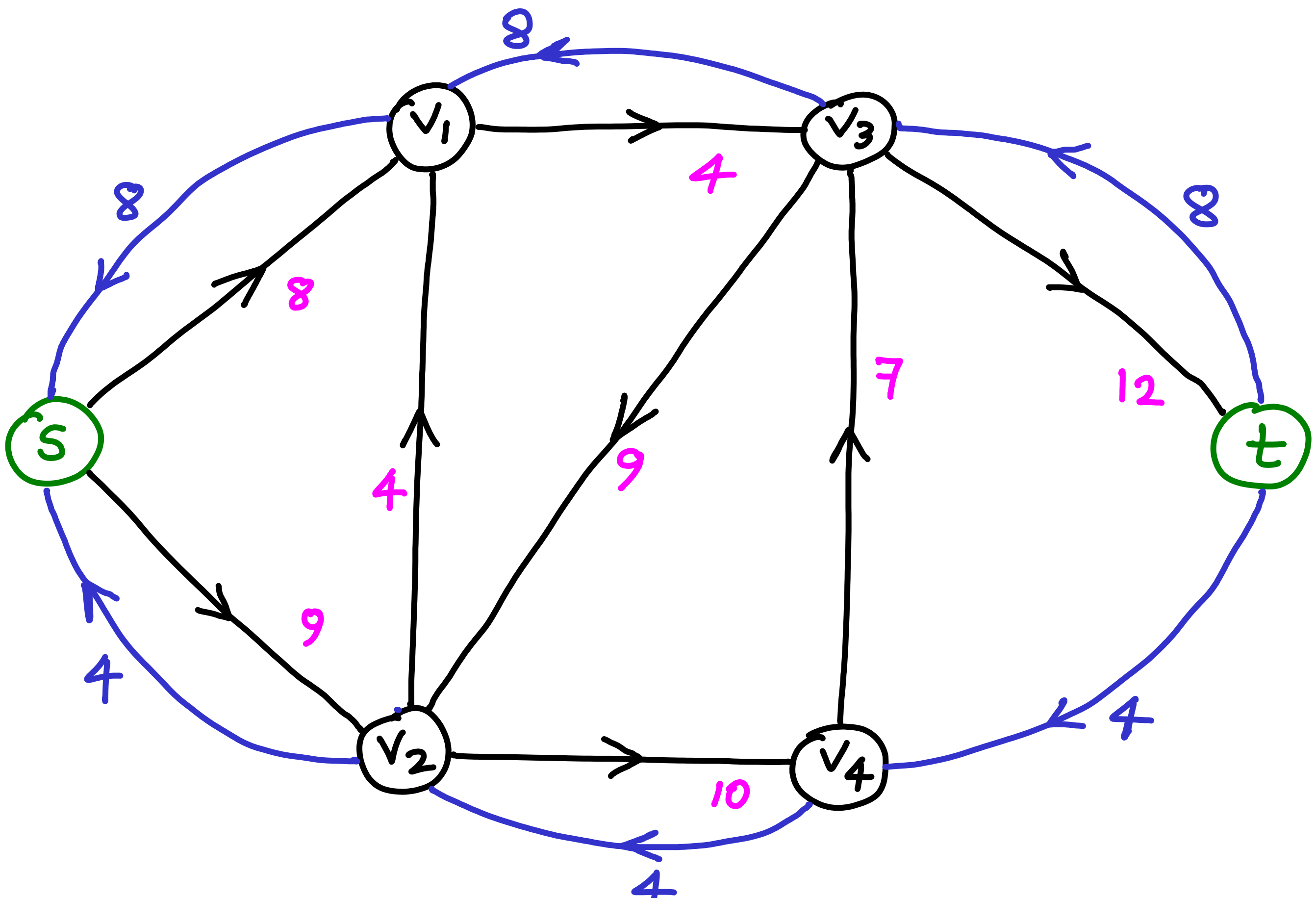
#: reverse (residual edge)

: augmenting path

$s \underset{12}{\longrightarrow} v_1 \underset{4}{\longrightarrow} v_2 \underset{4}{\longrightarrow} v_3 \underset{16}{\longrightarrow} t$



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



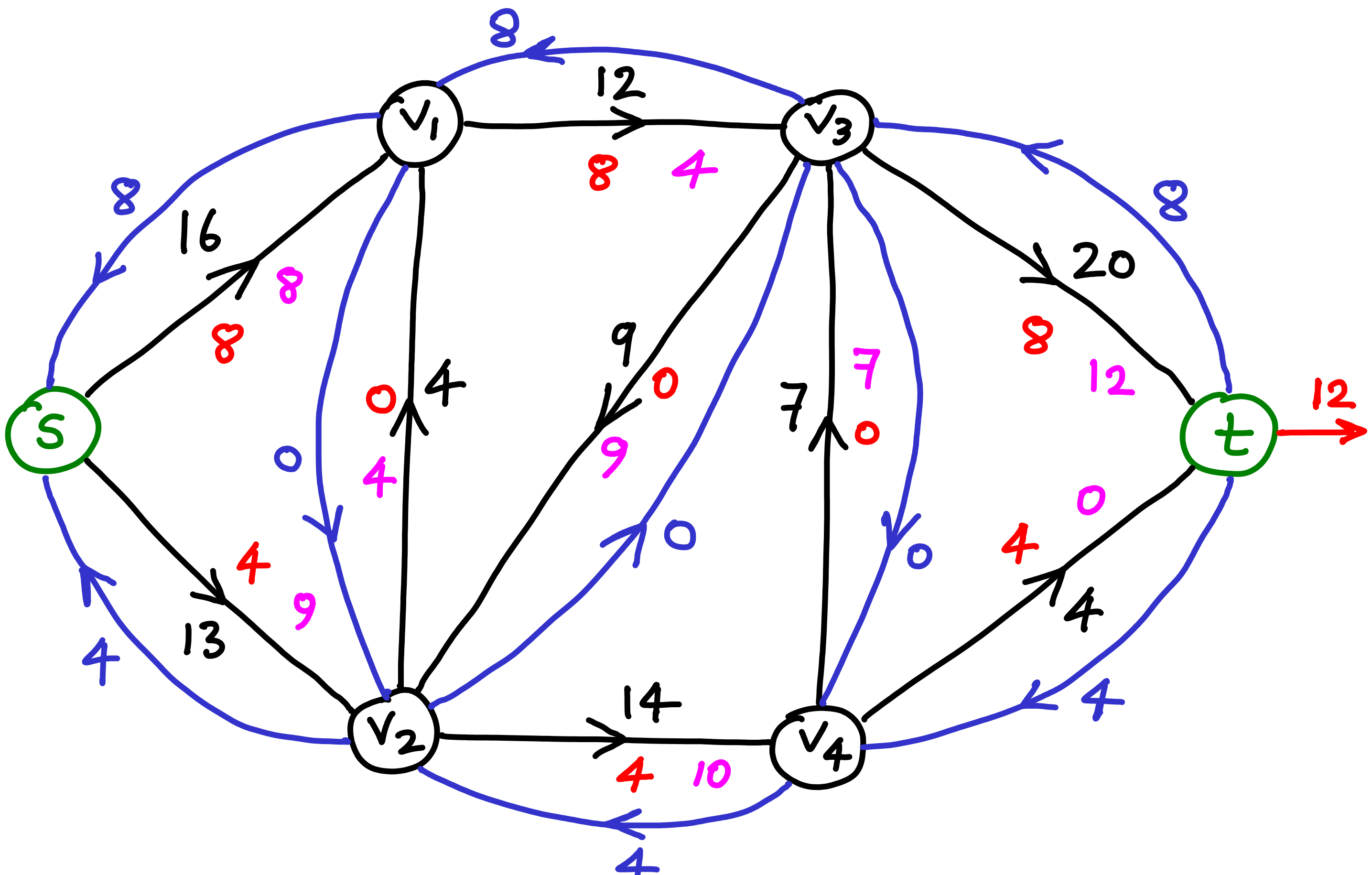
Residual graph

#: capacity (normal edge)

#: flow

#: leftover (residual edge)

#: reverse (residual edge)



#: capacity (normal edge)

#: flow

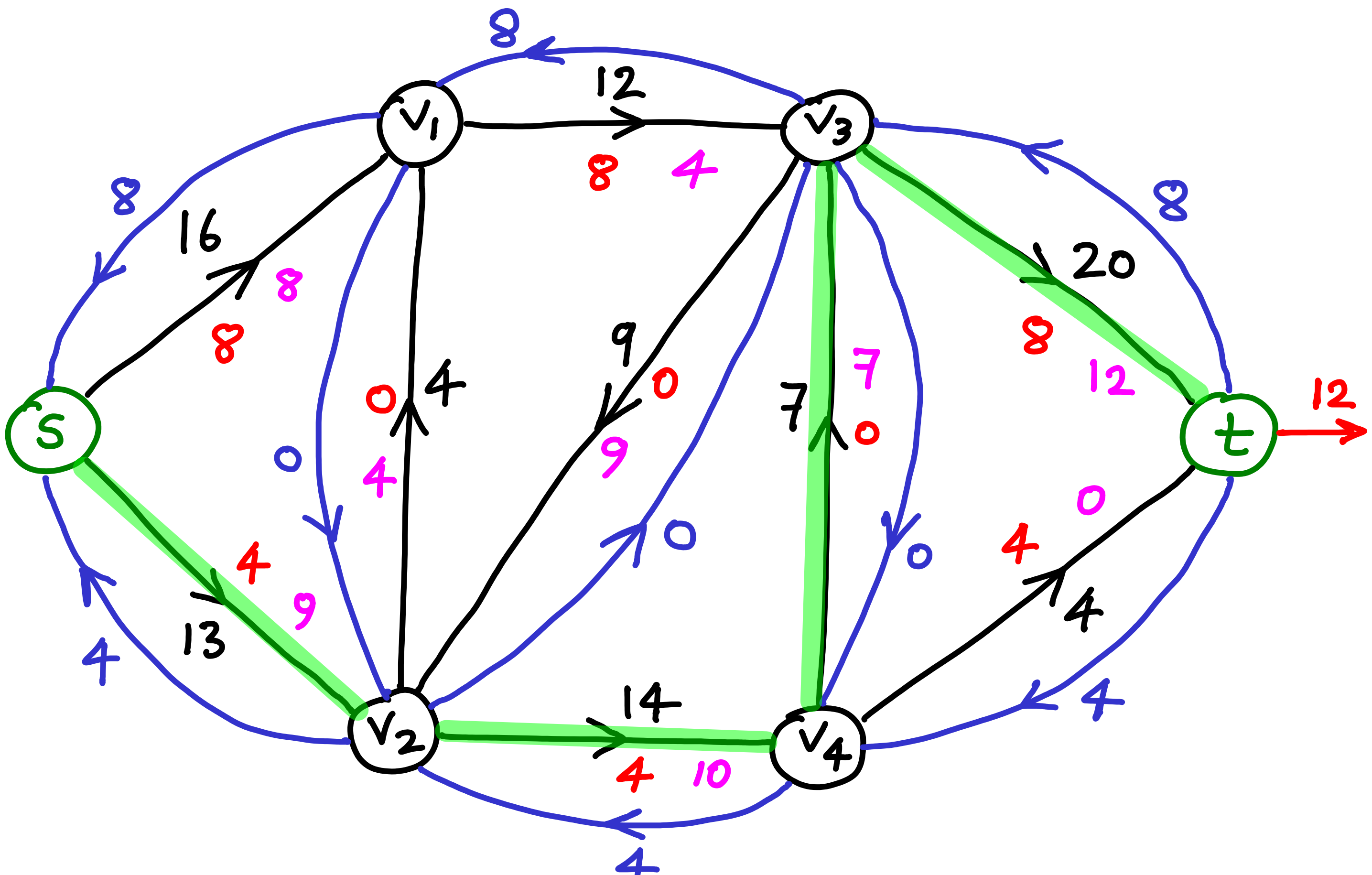
#: leftover (residual edge)

#: 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

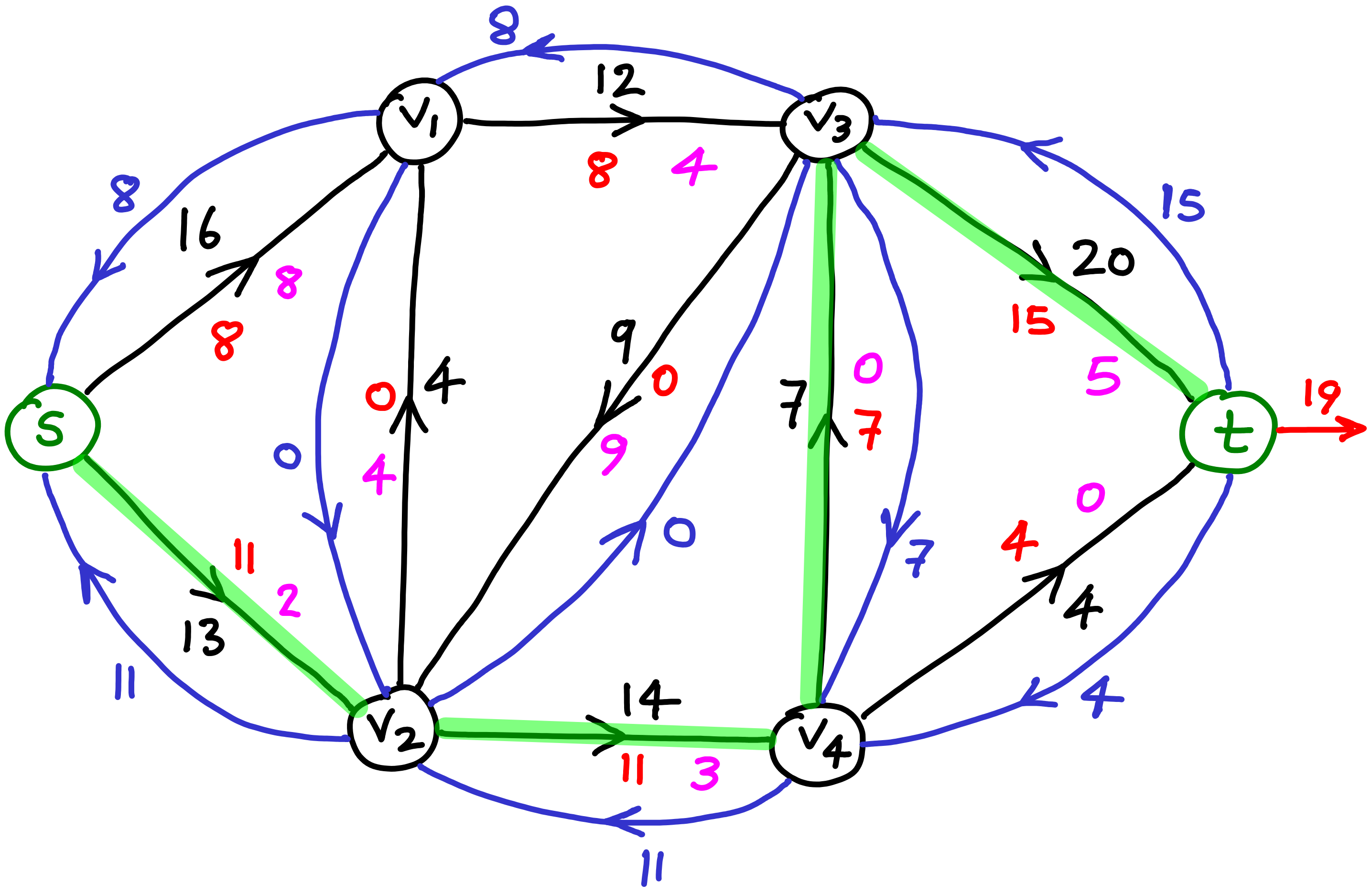
: leftover (residual edge)

: reverse (residual edge)

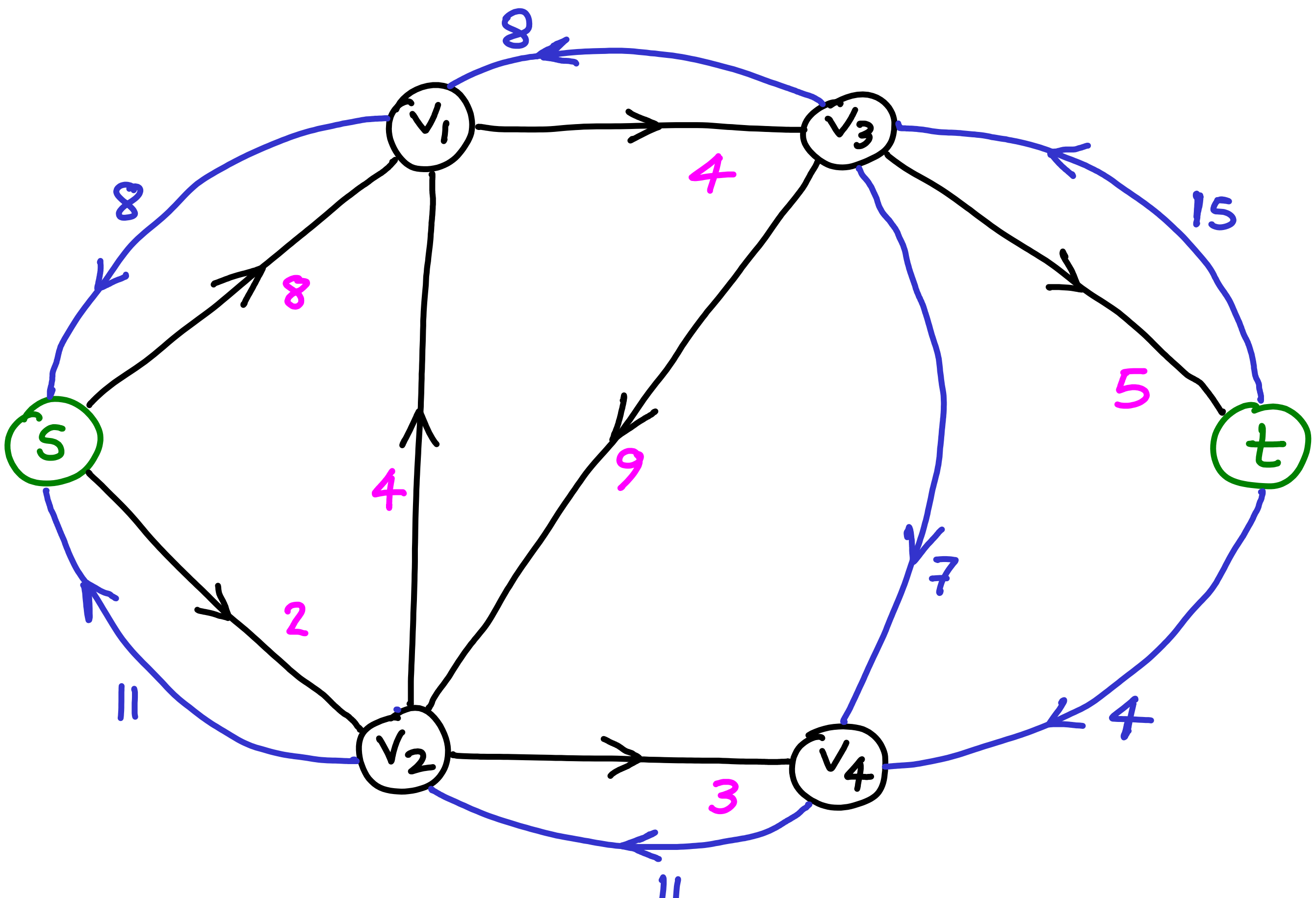
↝ : augmenting path

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

9 10 7 12



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



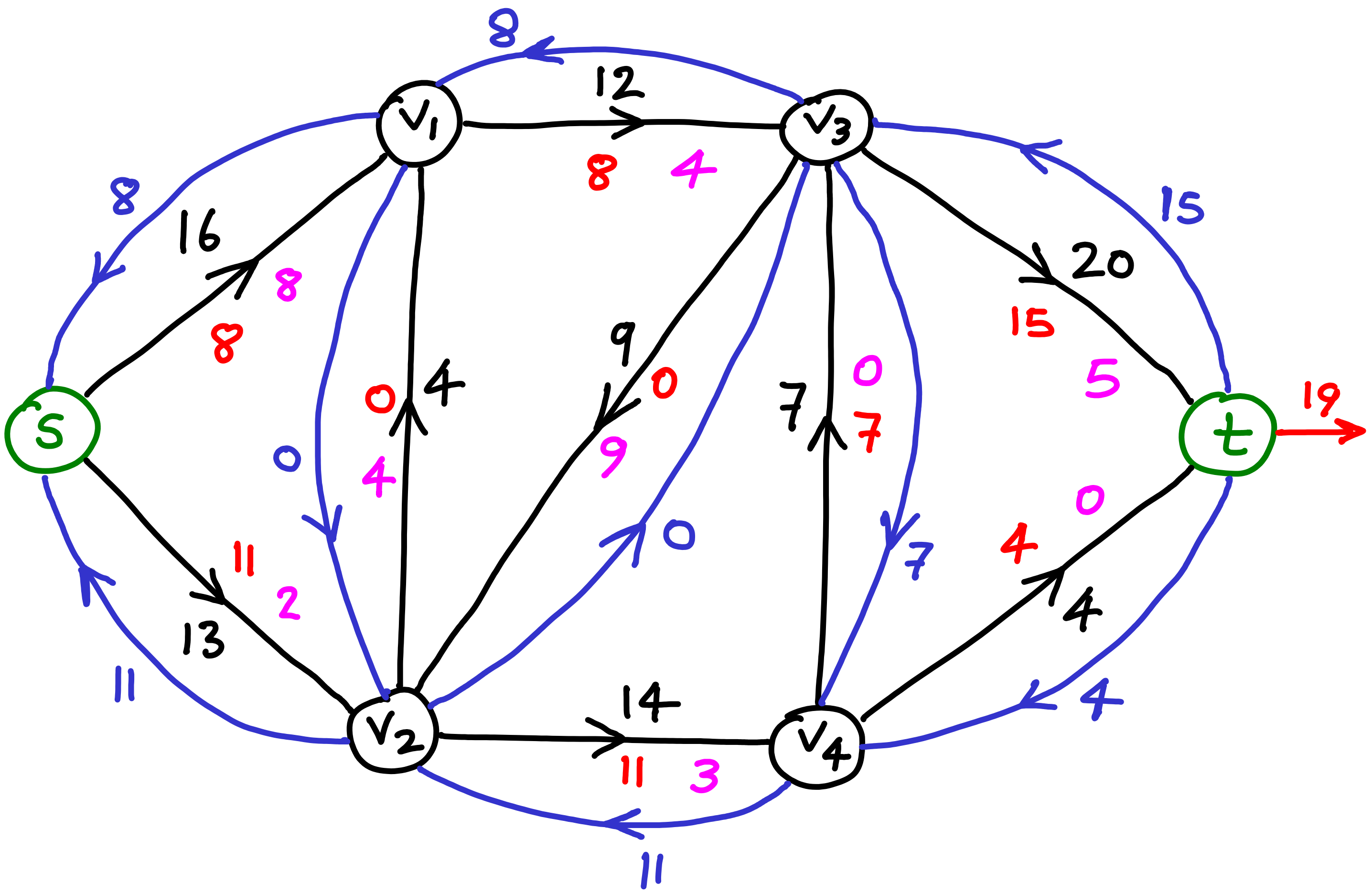
Residual graph

#: capacity (normal edge)

#: flow

#: leftover (residual edge)

#: reverse (residual edge)



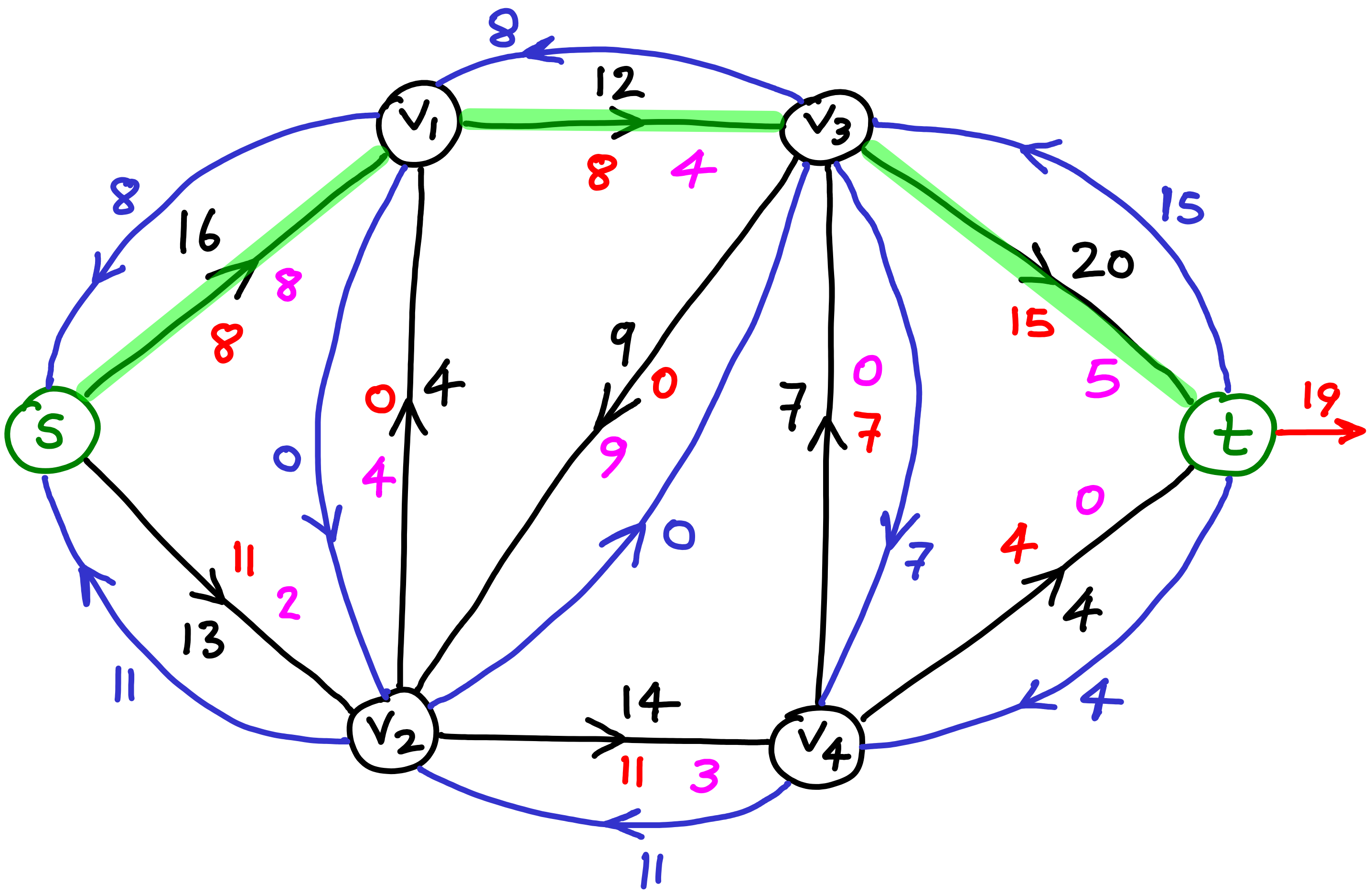
: capacity (normal edge)

: 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

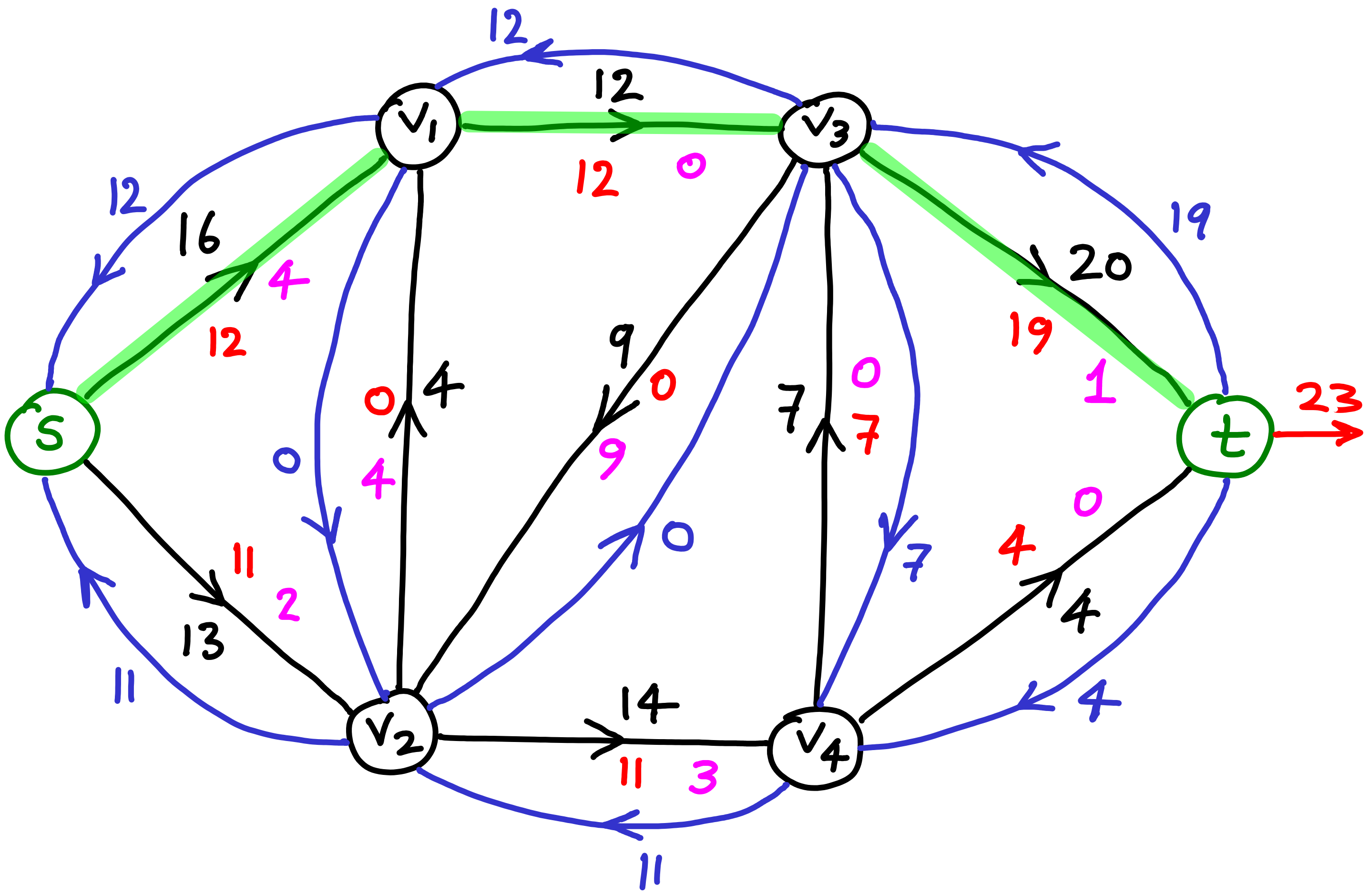
: leftover (residual edge)

: reverse (residual edge)

↝ : augmenting path

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

8   4   5

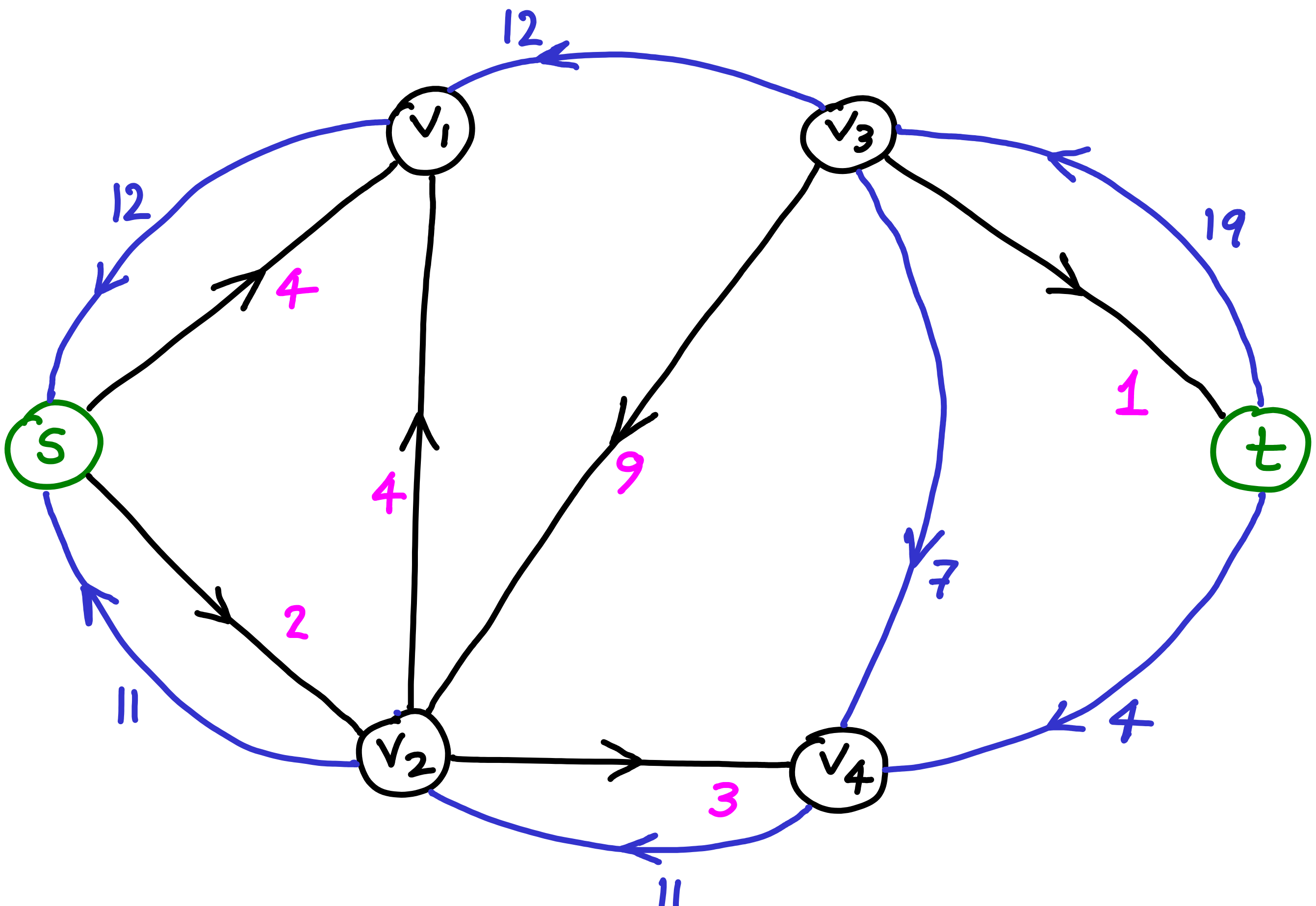


#: capacity (normal edge)

#: flow

#: leftover (residual edge)

#: reverse (residual edge)

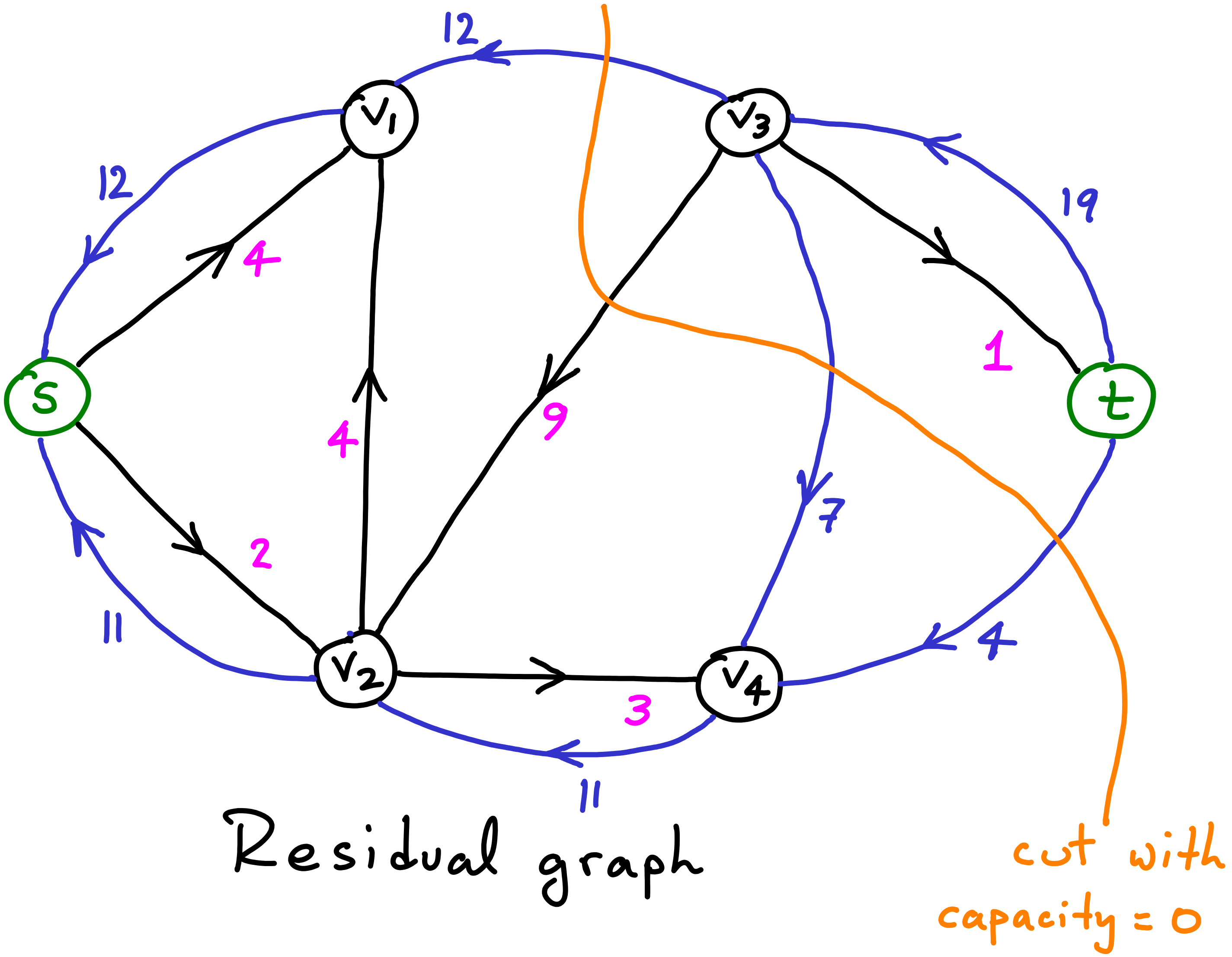


Residual graph

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

No augmenting path.

DONE



Residual graph

FORD - FULKERSON method

- while augmenting path exists

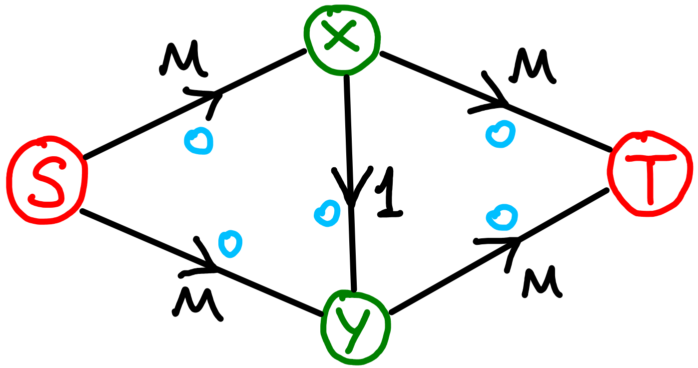
  how many times?

  increase flow on an augmenting path

  how do we find one?

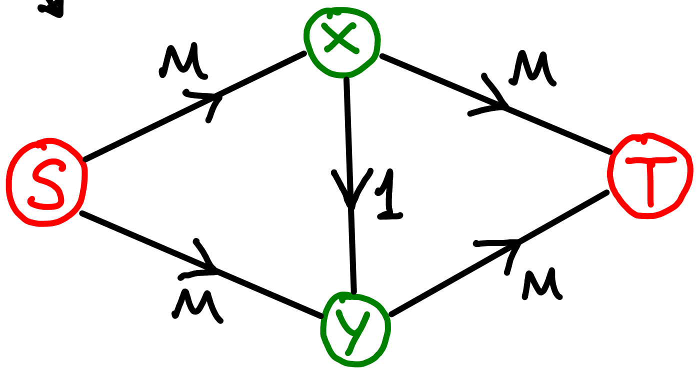
  increases flow value.
  $O(\text{path length})$

Arbitrarily?



$M = 10^6$

Residual



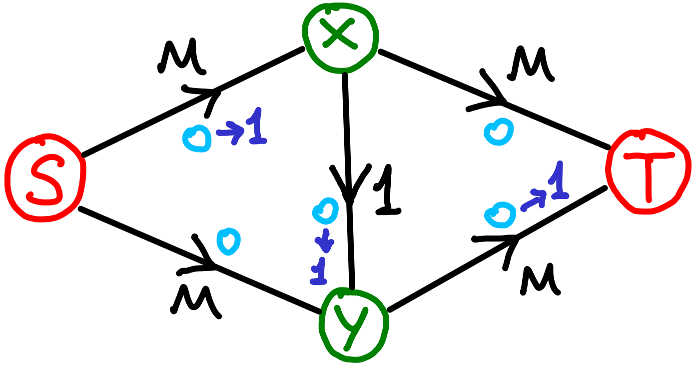
FORD - FULKERSON method

- while augmenting path exists → how many times?

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

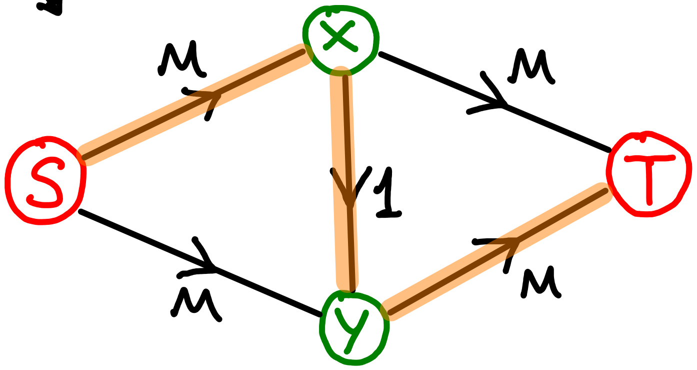
  { increases flow value. / O(path length)

Arbitrarily?



$M = 10^6$

Residual



FORD - FULKERSON method

- while augmenting path exists

  increase flow on an augmenting path

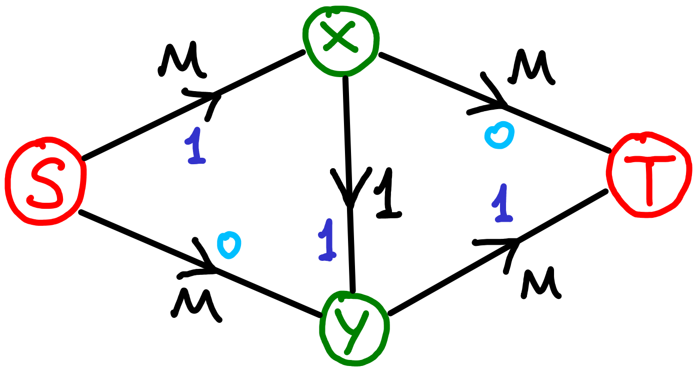
how many times?

how do we find one?

increases flow value.

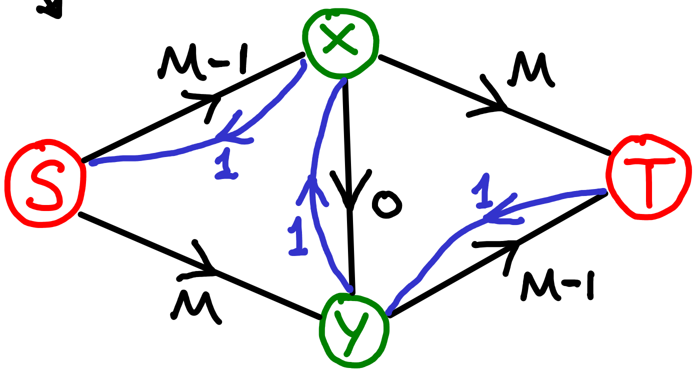
O(path length)

Arbitrarily?



$M = 10^6$

Residual



FORD-FULKERSON method

how many times?

how do we find one?

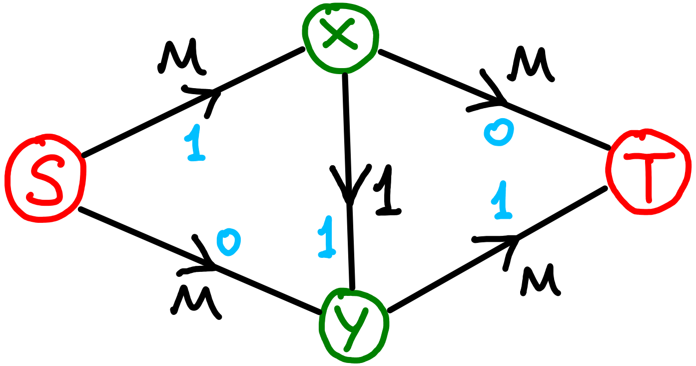
- while augmenting path exists

  increase flow on an augmenting path

  increases flow value.

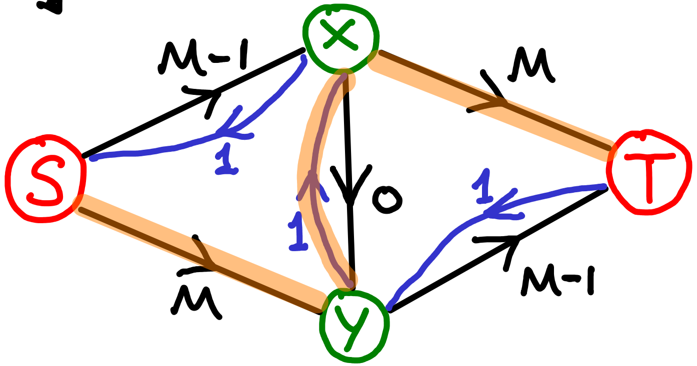
  O(path length)

Arbitrarily?



$M = 10^6$

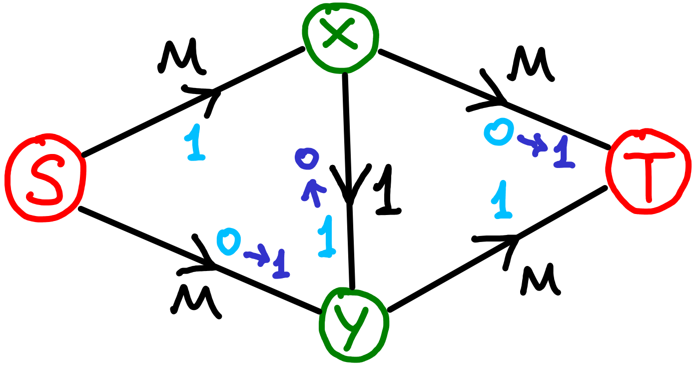
Residual



FORD - FULKERSON method

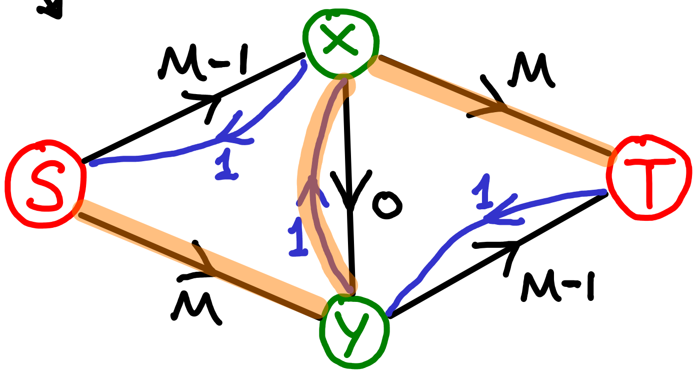
- while augmenting path exists → how many times?
  increase flow on an augmenting path → **how do we find one?**
  - increases flow value.
  - O(path length)

Arbitrarily?



$M = 10^6$

Residual

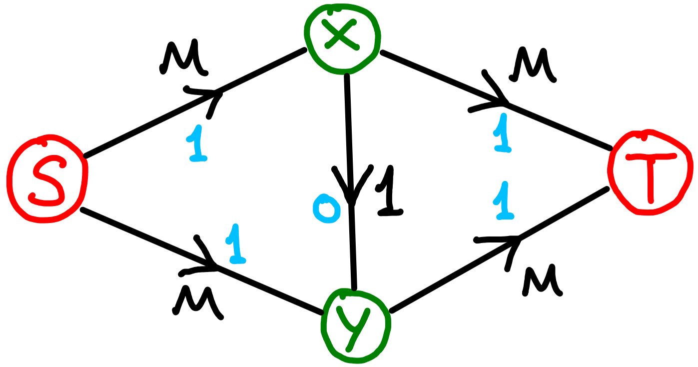


FORD - FULKERSON method

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

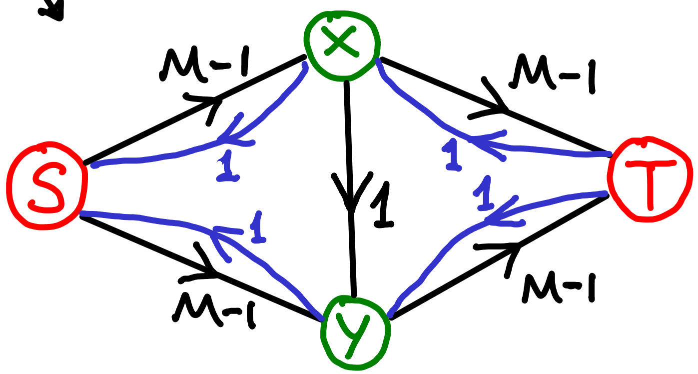
increases flow value.
O(path length)

Arbitrarily?



$M = 10^6$

Residual



FORD - FULKERSON method

- while augmenting path exists

  increase flow on an augmenting path

how many times?

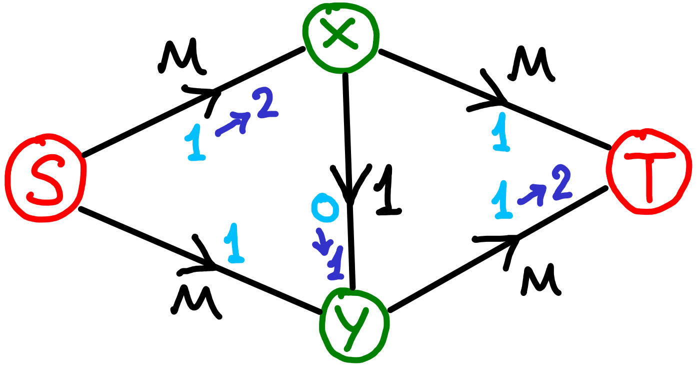
how do we find one?

increases flow value.

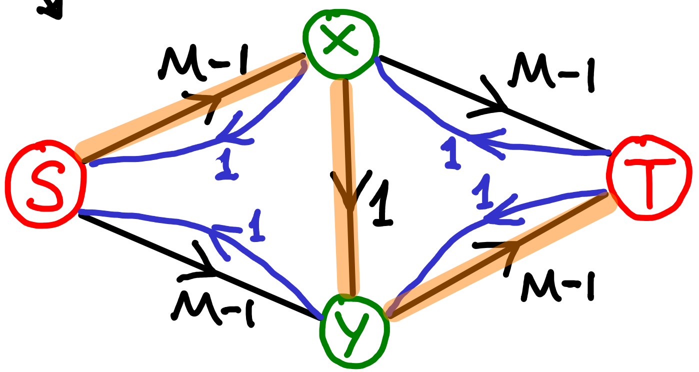
O(path length)

Arbitrarily? → # iterations = M

(or even DFS)



Residual



etc

$M = 10^6$

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 $\le$ MAX flow

For rational, transform to integer (multiply)

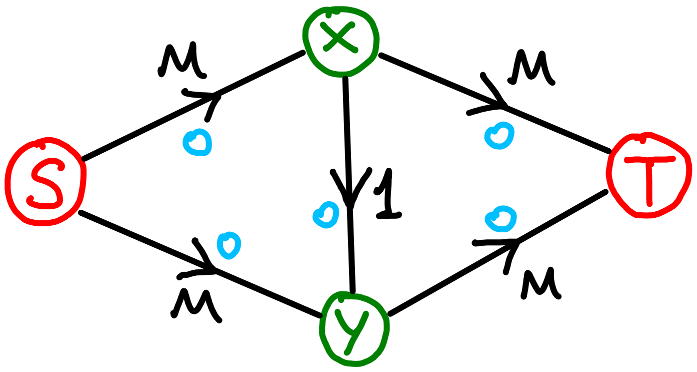
FORD - FULKERSON method

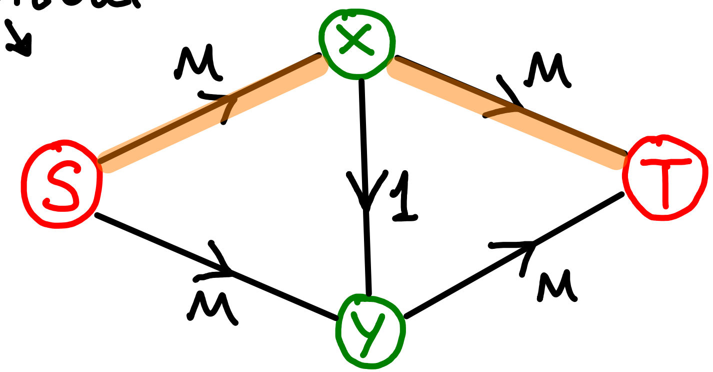
- while augmenting path exists — how many times ?
  - increase flow on an augmenting path — use BFS to find one

increases flow value.

O(path length)

Residual





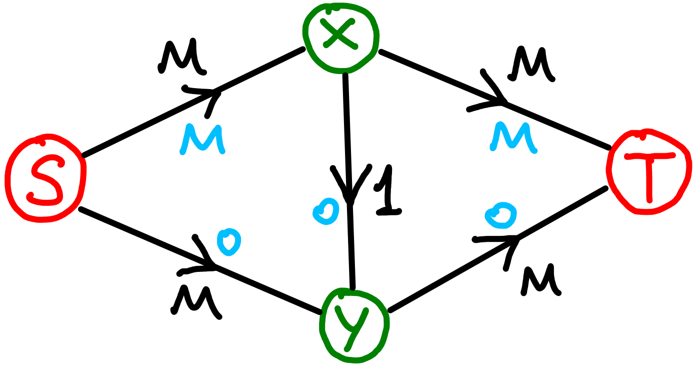
FORD - FULKERSON method

how many times ?

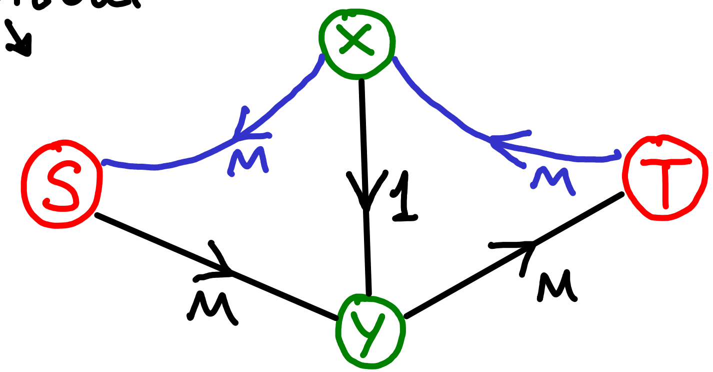
use BFS to find one

- while augmenting path exists
  increase flow on an augmenting path

$\left\{ \begin{array}{l} \text{increases flow value.} \\ O(\text{path length}) \end{array} \right.$



Residual



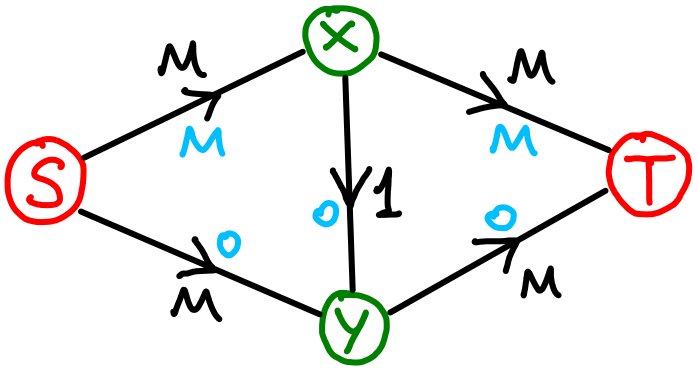
FORD - FULKERSON method

- while augmenting path exists → how many times ?

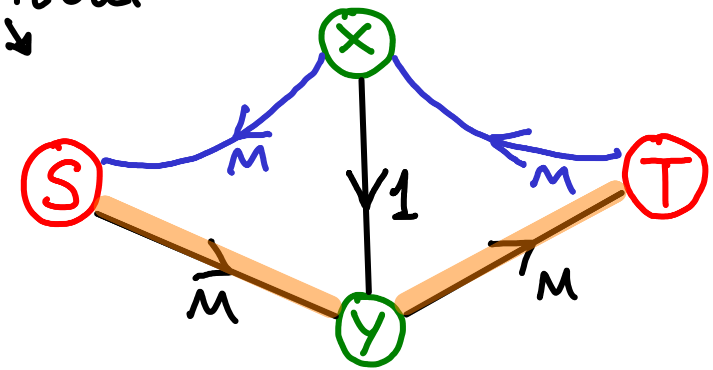
  increase flow on an augmenting path → use BFS to find one

  increases flow value.

  $O(\text{path length})$



Residual



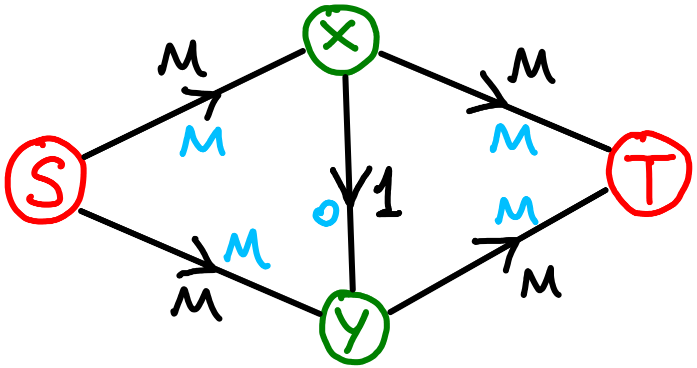
FORD - FULKERSON method

how many times ?

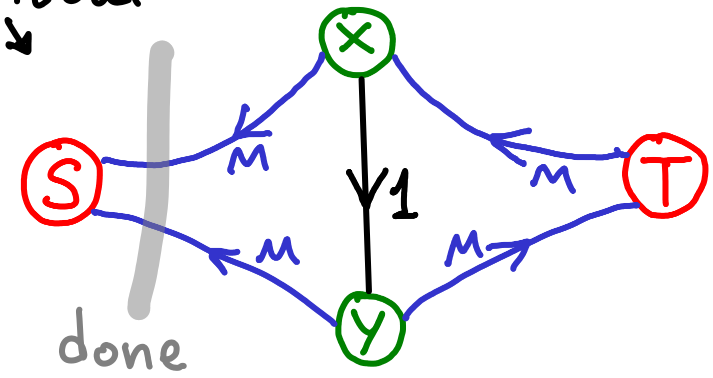
use BFS to find one

- while augmenting path exists
  increase flow on an augmenting path

increases flow value.
O(path length)



Residual



FORD - FULKERSON method

how many times ?

- while augmenting path exists

  use BFS to find one: $O(E)$

  increase flow on an augmenting path

  increases flow value.

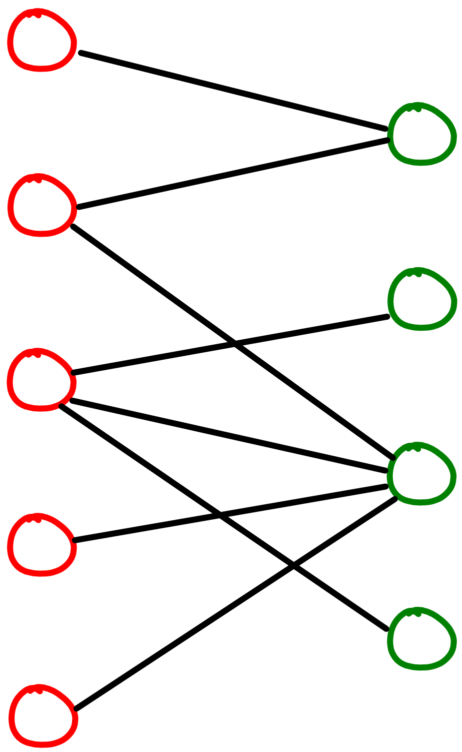
  $O(\text{path length})$

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

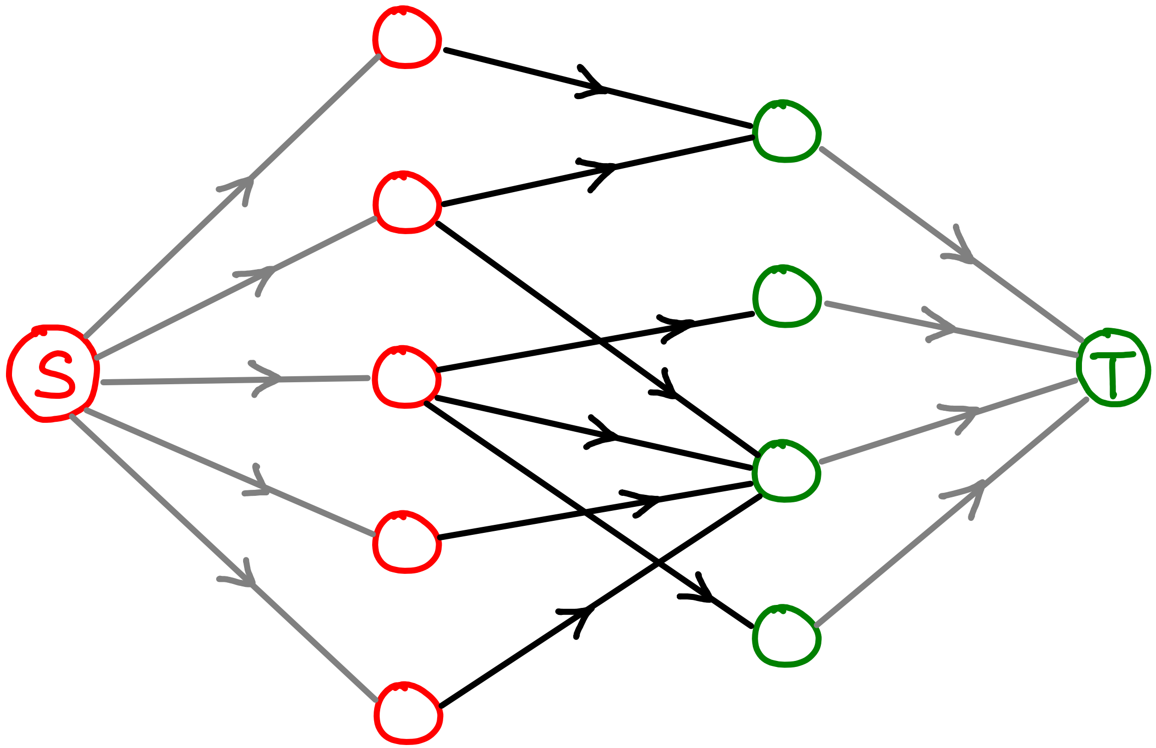
so $O(V \cdot E^2)$ time

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

FINDING A MAXIMUM BIPARTITE MATCHING

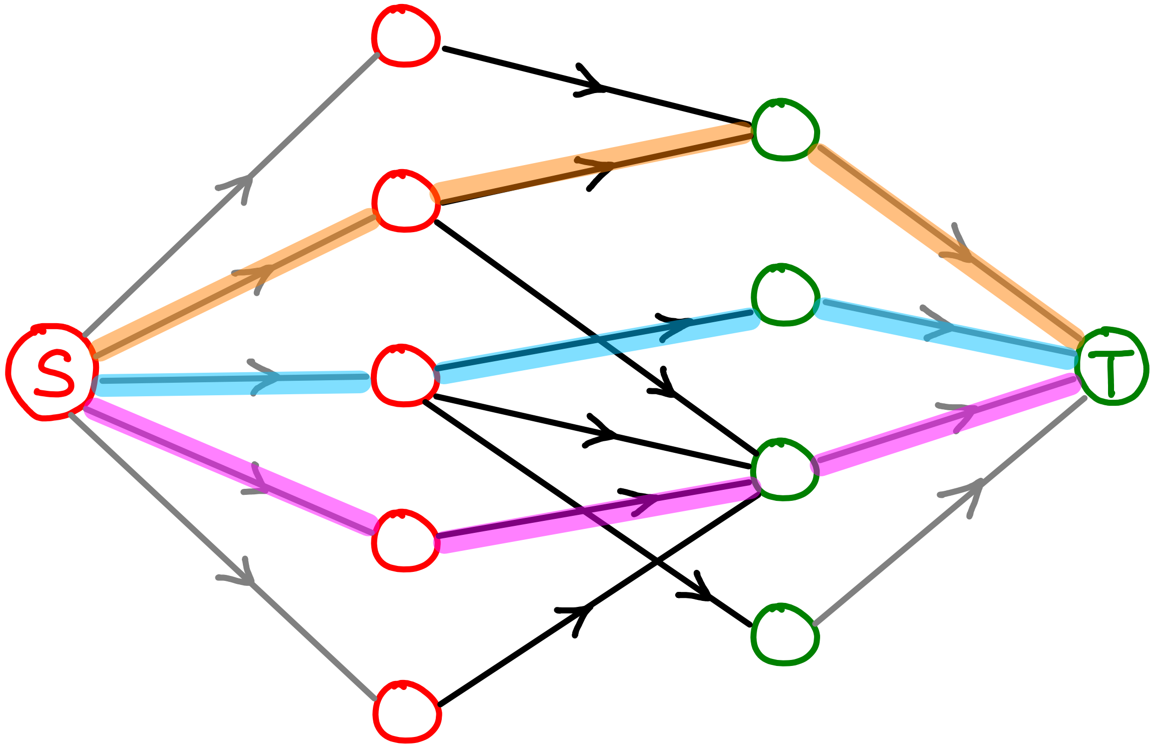


FINDING A MAXIMUM BIPARTITE MATCHING



All capacities = 1

FINDING A MAXIMUM BIPARTITE MATCHING



All capacities = 1

(integer)

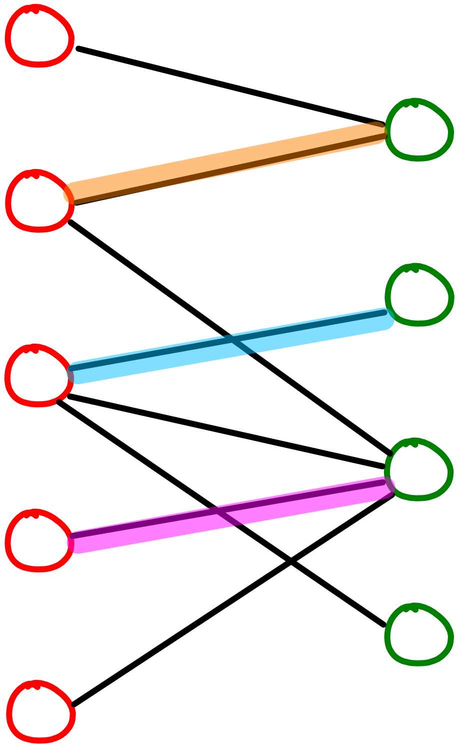
• MAX FLOW = $O(V)$

iterations

BFS

so $O(V \cdot E)$ 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