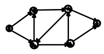
Network Flow

22 May 2025 20:49



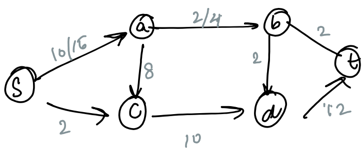
Flow Network → Abstraction of flow

i.) Digraph $G(V,E)$ ← no $\parallel$ edge, no edge entering source, no edge leaving t

→ source (S)

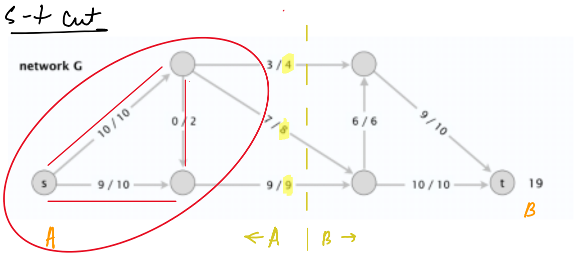
→ sink (t)

ii.) $c(u,v) \geq 0$ $\quad u \rightsquigarrow v$



10/16 = flow / capacity.

s-t cut



Capacity = 4 + 8 + 9
= 21

S-t → partition of graph such that $S \in A$, $t \in B$

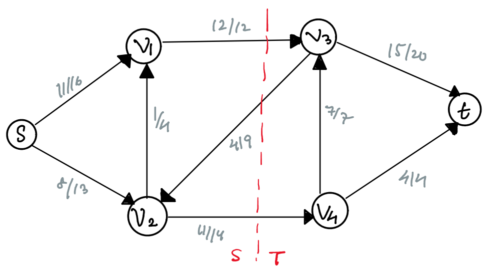
$$\sum C(A,B) = \sum_{e \text{ out of } A} c(e)$$

Capacity of the s-t cut = sum of capacities going out from A

* The net flow across the cut

$$F(A,B) = \sum_{u \in A} \sum_{v \in B} F(u,v) - \sum_{v \in B} \sum_{u \in A} F(v,u)$$

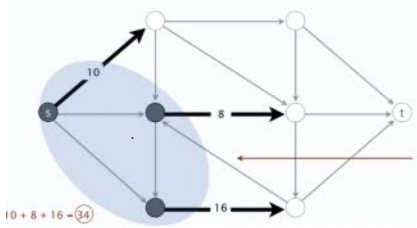
eg:→



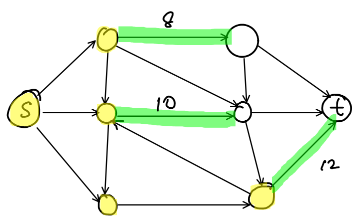
Capacity (S-t) = 12 + 14
= 26

Flow (S-t) = 12 + 11 − 4
= 19.

eg:→



Not from B to A.



O → in set A

capacity = 8 + 12 + 10
= 30

Min cut problem

→ finding cut with minimum capacity.

Max flow problem

→ find a flow of max. value

→ $0 \leq f(e) \leq c(e)$

↳ flow across e

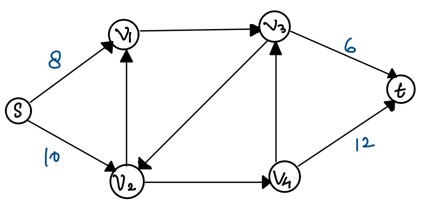
→ $v \in V - \{s,t\}$ → $\sum_{e \text{ out of } v} f(e) = \sum_{e \text{ into } v} f(e)$

value of flow $(f) = \sum_{e \text{ out of } s} f(e)$

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

But in our case no edge into source

$\therefore val(f) = \sum f(s,v)$



$val(f) = 8 + 10 = 18 = 6 + 12.$

FORD – FULKERSON ALGORITHM

* Towards Max-flow algo.

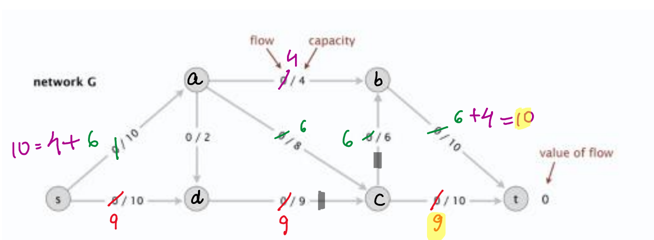
GREEDY ALGO

i.) start with $f(e) = 0$ for all edges

ii) Find a path $s \rightsquigarrow t$, say P such that $f(e) < c(e)$

iii) Augment the path.

iv.) Repeat until stuck.



Total flow = 10 + 9 = 19

i.) for a path $s \to d \to c \to t$

we have min $c(e) = 9$

so we pass 9

ii) for path $s \to a \to c \to b \to t.$

min $c(e) = 6.$

so pass 6.

iii) in

$s \to a \to b \to d$

$f(e) < c(e)$

and min $c(e) = 4$

so pass 4

RESIDUAL GRAPH

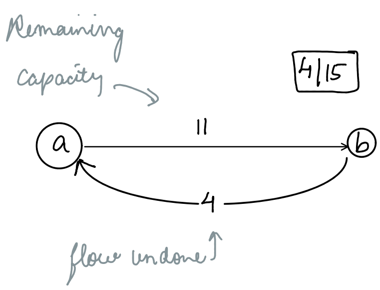
* Residual edge

→ undo 'flow'

→ $e \in E$ & $e^R \in E$

→ residual capacity:

$$c_f(e) = \begin{cases} c(e) - f(e) & : e \in E \\ f(e) & : e^R \in E \end{cases}$$



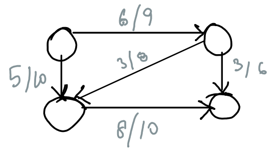
$e = (u,v)$

$e^R = (v,u)$

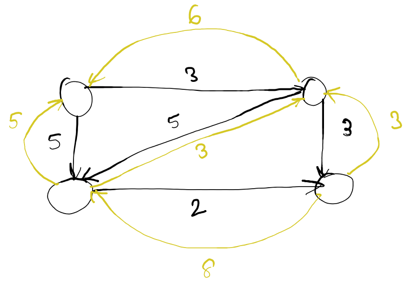
$$c_f(e) = \begin{cases} c(e) - f(e) & : e \in E \\ f(e) & : e^R \in E \end{cases}$$

flow undone ↑

Eg :→



→



Residual graph :→ $G_f = (V, E_f)$

$$E_f = \underline{\{e \mid f(e) < c(e)\}} \cup \underline{\{e^R \mid f(e) > 0\}}$$

Edge if flow < capacity. Residual edge of flow undone

* f' is said to be $\underbrace{\text{Residual flow}}_{\text{flow in } G_f}$ iff $f + f'$ = flow in G

∴ $|E_f| \leq |E|$ — if $f(e) > 0$ the 2 edges, else 1 edge.

Augmented Path

* Simple path $s \to t$ in the residual graph, say G_f.

* **Bottle Neck capacity :→**

The amount by which we can increase flow on each edge in the augmented path P.

given by $\boxed{c_f(P) = \min(c_f, e \in P)}$

Bottleneck capacity = min of the residual capacity of edges that belong to path P.

Let f be a flow in G, then f' is a flow in G' if

$$\boxed{val(f') = val(f) + c_f(P)}$$

→ augmenting → adding to value

$val(f')$ adds possible flow.

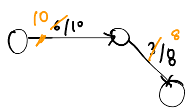
```
Augment (f, c, P)
  b ← BottleNeck capacity.
  For each (e ∈ P)
      If (e ∈ E)
          f(e) = f(e) + b
      else
          f(e^R) = f(e^R) - b.
  return f
```

We have 2 graphs.

i) Original

ii) Residue.

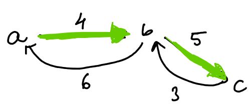
$e \in E$



original

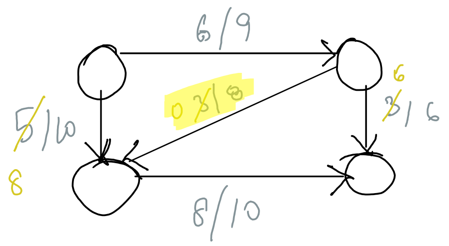
Augmented Path P
= a → b → c

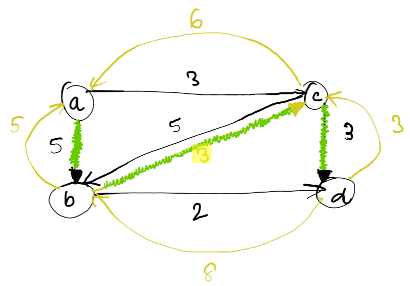
b = 4 (min)



Choose one at a time.

Eg :→ Sometimes the path has opp to what is in original graph then we will have to reduce flow.





Path : a → b → c → d

the edge $(b, c) \notin E$

So we decrease flow in original graph.

Hence collectively we define function, $f \uparrow f'$

i.e. f augmented by f' as

$$(f \uparrow f')(u,v) = \begin{cases} f(u,v) + \overbrace{f'(u,v)}^{\text{add if } \in E} - \overbrace{f'(v,u)}^{\text{sub'g opp.}} & : (u,v) \in E \\ 0 & : \text{otherwise} \end{cases}$$

FORD-FULKERSON ALGO

i.) start with $f(e) = 0 \quad \forall e \in E$

ii.) assign G_f as the residual graph

iii.) as long as an augmented path exist

iv.) augment flow along path p

v.) stop when stuck

FORD-FULK(f, p, c,

i.) $f(e) \leftarrow 0 \quad : e \in E$

ii.) $G_f \leftarrow$ Residual graph

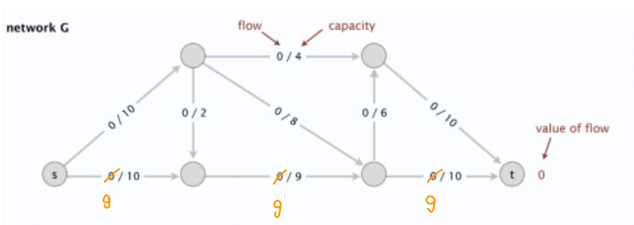
iii.) while (an augmented path persist)

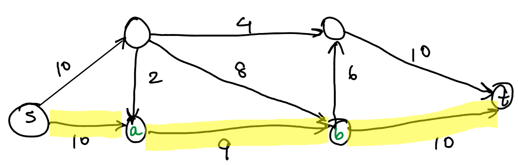
iv.) Augment(f, C, p)

v.) update G_f

vi.) Return f

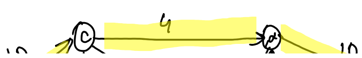
Eg :→

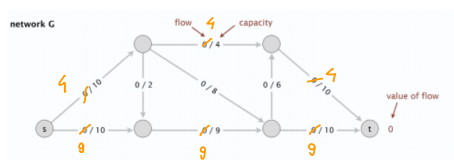




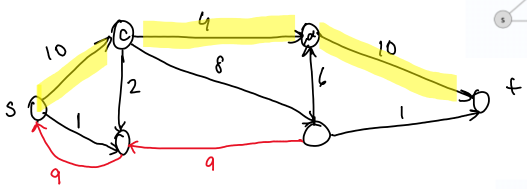
S → a → b → t

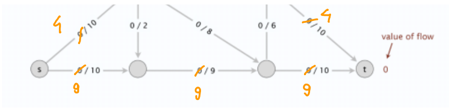
b = 9





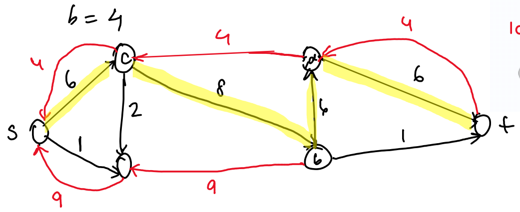
$b = 4$

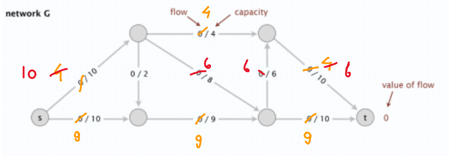




$s \to c \to d \to t$

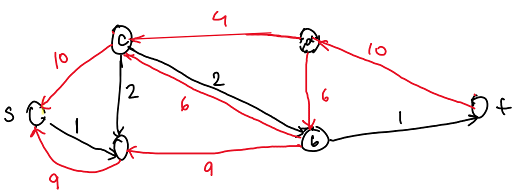
$b = 4$

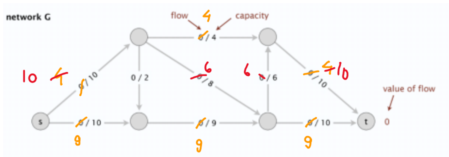




$s \to c \to b \to d \to t$

$b = 6$





No augmented path. $\therefore$ from the orig graph, flow $= 10 + 9 = 19$ ✓

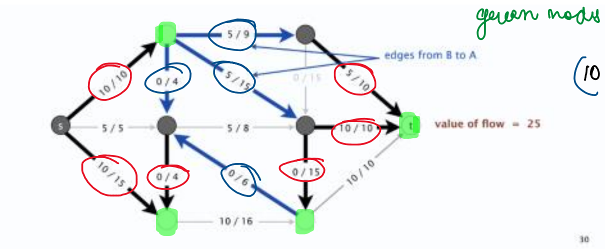
Relation between flow and cut.

The value of flow across cut (A, B) is defined by

$$\boxed{val(f) = \sum_{e \text{ out of } A} f - \sum_{e \text{ into } A} f}$$

← flow value lemma

green nodes are B.

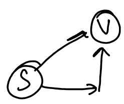


$(10 + \ 10 + 0 + 0 + 10 + 5)$

$-(5 + 5 + 0 + 0)$

$= 35 - 10 = 25$,

Proof of flow value lemma.



$$val(f) = \sum_{u \in V} f(s, u)$$

$$= \sum_{v \in A} \left(f(v, e) - f(e, v) \right)$$

→ flow of conservation all terms except $v = s$ are 0 remains $f(s, u)$

$$= \sum f(A, e) - \sum f(e, A)$$

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

Weak duality.

Weak duality.

given a flow f, $V(f) \leq C(A,B)$ $\longrightarrow$ $val(f) \leq$ capacity of cut (A,B)

Pf:→

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

$$\leq \sum_{e \text{ out of } A} f(e) \leq \sum_{e \text{ out of } A} c(e) = cap(A,B)$$

$\therefore$ $val(f) \leq cap(A,B)$

Max flow Min cut theorem

Augmenting path theorem : max when no P

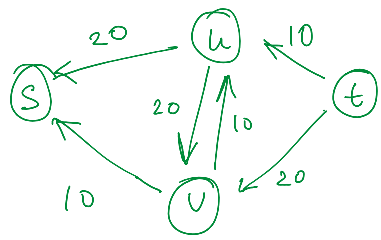
Max flow Min cut : Max flow = cap of min-cut

If f is a flow in $G(V,E)$ then following holds:-

* i.) f is max flow in G

ii.) The residual network G_f → no augmenting Path

iii) $|f| = c(S,T)$ for some cut (S,t) of G



Pf:→ * (ii → i) suppose for any cut, cap(A,B) = val(f)

→ Then for any flow f', $val(f') \leq cap(A,B) = val(f)$

→ f is Max flow

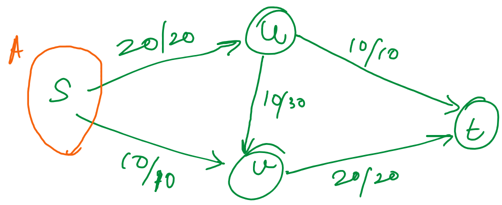
* (ii → i) we prove contrapositive, $\sim ii \Rightarrow \sim i$.

suppose P exist

f will not be max.

$\therefore$ P does not exist

* (iii) by flow value lemma



Max flow = 30

cap(A,B) = 20 + 10 = 30 ← Min

cap if S,u = 30+10 = 40

S,v = 20+20 = 40