Max Flow

→ Formal description of problem

$G = (V, E)$ directed graph. s = source, t = sink

$c : E \to R^+$ $\quad c(x,y)$ = capacity of (x,y)

flow $f : E \to R^+$ s.t.

i) Capacity constraint: for each edge $(x,y) \in E$

$$f(x,y) \leq c(x,y)$$

ii) Conservation constraint: for each vertex $v \in V \setminus \{s,t\}$

$$\sum_{(u,v) \in E} f(u,v) = \sum_{(v,z) \in E} f(v,z)$$

value (f) = flow leaving s. $= \sum_{(s,v) \in E} f(s,v)$

$$= \sum_{(u,t) \in E} f(u,t)$$

Max flow problem $\Rightarrow$ (value (f)) ↑

Redistribution of flow

i) Increase flow along an edge (if current flow is less than capacity)

ii) Reducing flow along an edge (if it is carrying some positive flow)

We add extra edges in G_f to form a residual network. $a \xrightarrow[y]{x} b \Rightarrow a \xrightarrow{x-y} b$

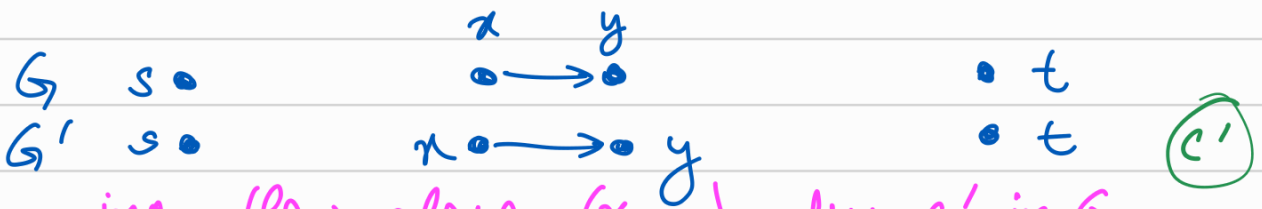
Residual Network:

$G = (V, E)$ and let $f: E \to R^+$ be any valid s-t flow.

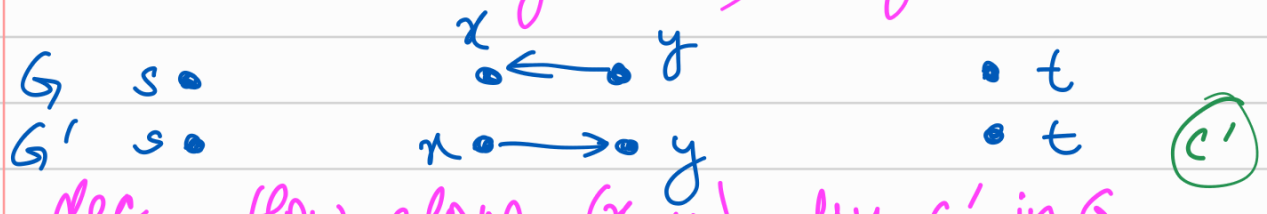
$G_f = (V, E_f)$ for each $(x, y) \in E$

i) if $f(x,y) < c(x,y)$
Forward edge: $c_f(x,y) = c(x,y) - f(x,y)$

ii) if $f(x,y) > 0$ add edge $(y,x) \in E_f$
Backward edge: $c_f(y,x) = f(x,y)$



inc flow along (x,y) by c' in G.



dec flow along (x,y) by c' in G.

(basically inc flow towards some other outgoing edge of y and inc flow along some edge entering $x \Rightarrow$ redistrin).

s-t cuts

$A \subset V$ with $s \in A$ and $t \in \bar{A}$ then

cut $(A, \bar{A}) = \{(x,y) \in E \mid (x \in A \text{ and } y \in \bar{A}) \text{ or } (x \in \bar{A} \text{ and } y \in A)\}$

only A → $\bar{A}$ in capacity

capacity $c(A, \bar{A}) = \sum_{(u,v) \in E,\ u \in A,\ v \in \bar{A}} c(u,v)$ outgoing edges only

Fork-Fulkerson algo (G, s, t) {

$f \leftarrow 0$ $\forall$ edges $(x,y) \in E$.

while (there is an s-t path in G_f) {

let P = path (s,t) in G_f

c' = bottleneck capacity of P;

for (each $(x,y) \in P$) {

if ((x,y) = forward) {

$f(x,y) = f(x,y) + c'$;

$c(x,y) = c(x,y) - c'$;

if ($c(x,y) = 0$) remove (x,y); }

else {

$f(y,x) = f(y,x) - c'$;

$c(y,x) = f(x,y)$; } }

return f; }

f_{in} and f_{out}

f_{in}  f_{out}

$$\sum_{(z,u) \in E} f(z,u) = \sum_{(u,v) \in E} f(u,v)$$

$f_{in}(u)$ $f_{out}(u)$

$$f_{out}(u) - f_{in}(u) = \begin{cases} 0 & u = V \setminus \{s,t\} \\ value(f) & u = s \\ -value(f) & u = t \end{cases}$$

$f_{out}(s) = f_{out}(A) = value(f) = f_{in}(\bar{A}) = f_{in}(t)$

$f_{in}(s) = f_{in}(A) = 0 = f_{out}(\bar{A}) = f_{out}(t)$

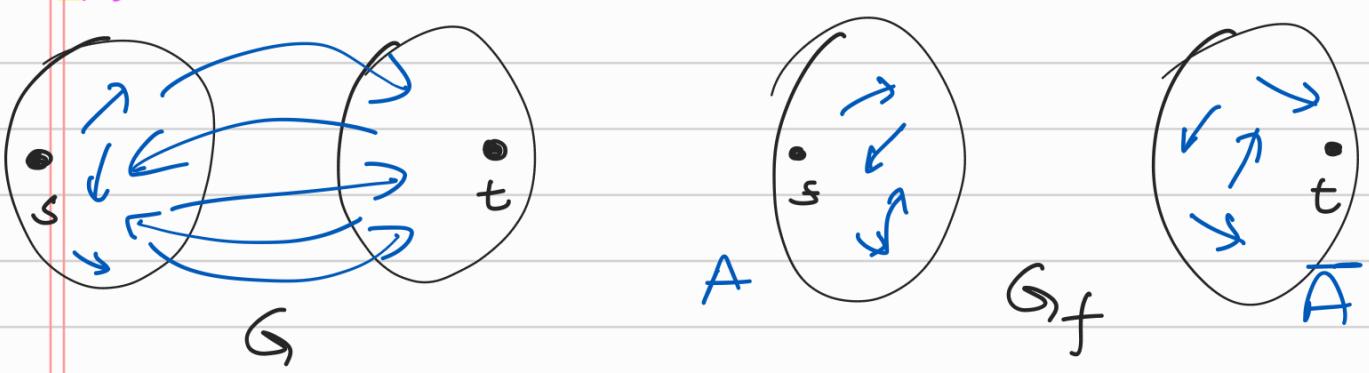
$$f_{out}(A) = \sum_{(x,y)\in E,\ x\in A,\ y\in \bar{A}} f(x,y)$$

$$f_{in}(A) = \sum_{(x,y)\in E,\ x\in \bar{A},\ y\in A} f(x,y)$$

Lemma: $f_{out}(A) - f_{in}(A) \leq c(A, \bar{A})$

The max s-t flow is bounded by the capacity of every s-t cut.

$\Rightarrow$ The max s-t flow is equal to the minimum s-t cut.



A: set of vertices reachable from s in G_f.

value $(f) = c(A, \bar{A})$

i) all outgoing edges must be fully saturated $\Rightarrow f(x,y) = c(x,y)$

ii) every incoming edge must have zero flow. $f(y,x) = 0$

Time $= O(mC)$

C = max edge capacity.

} NOT polynomial even for integer cap. network

Integrality of max flow

Given network $G=(V, E)$ and $s, t \in V$ if edge capacities are integers, then $\exists$ a max s-t flow f which is integral, $f(e) \in \mathbb{Z}$ $\forall e \in E$.

Ford Fulkerson computes a max flow which is integral, if capacities are integral.

→ Modification of FF Algo

instead of any s-t path, we pick the path of the max-capacity.

```
f ← 0 ; K ← max-cap(E);
While (K ≥ 1) {
    While (∃ any s-t in G_f with cap ≥ K) {
        Pick P.   // for each (x,y) ∈ P {
            if (x,y) ∈ fwd edge  f(x,y) += c';
            else                 f(y,x) -= c';
    K ← K/2; }
```

Time = $O(m^2 \log c_{max})$

distance means no. of edges in the path from s to t.

$O(mn)$

```
While (∃ s-t path P in G_f) {
    Let P be the shortest s-t path in G_f.
    for each (x,y) ∈ P
        if (x,y) ∈ fwd   f(x,y) += c';
        else             f(y,x) -= c';
```

$O(m^2 n)$

$O(m)$ BFS

Consider f when $\exists$ no s-t path with cap $\geq K$.

A: set of v reachable from s, cap $\geq K$.

$A \longrightarrow \overline{A}$ $\quad c(x,y) - f(x,y) < K \Rightarrow f > c - K$

$\overline{A} \longrightarrow A$ $\quad f(x,y) < K \Rightarrow f < K$

$f_{out} - f_{in} > c - mK \Rightarrow f_{max} \cdot mK$

Real Edge Weights

Shortest s-t path, any edge can disappear & re-appear $O(n)$ times.

→ After the i^{th} iteration

1. Disappearance of a Forward Edge (x,y)
$f(x,y) = c(x,y) \Rightarrow (x,y) \in P$ selected in i^{th} iter

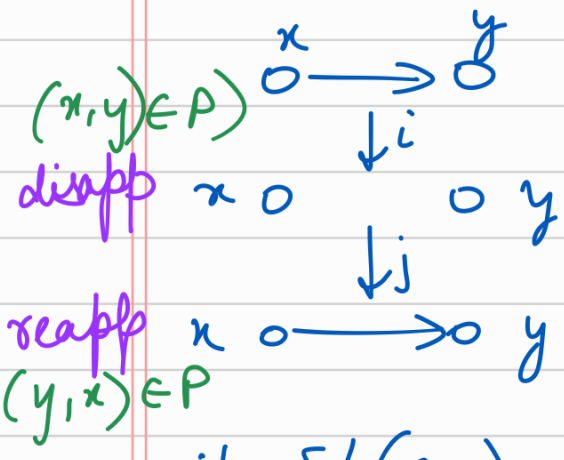
2. Disappearance of a Backward Edge (x,y)
$f(y,x) = 0 \Rightarrow (x,y) \in P$ selected in i^{th} iter

3. Reappearance of a Forward Edge (x,y)
$f(x,y)$ dec from $c(x,y) \Rightarrow$ So $(y,x) \in P$ selected in i^{th} iter

4. Reappearance of a Backward Edge (x,y)
$f(y,x)$ flow ↑ from 0 $\Rightarrow (y,x) \in P$ selected in i^{th} iter

(x,y) disappears $\Rightarrow (x,y) \in P$
(x,y) reappears $\Rightarrow (y,x) \in P$

→ Bound on the Time Complexity



$\delta(s,y) = \delta(s,x) + 1$

$\delta(s,x) \leq \delta'(s,x)$

$\delta'(s,x) = \delta'(s,y) + 1$

if $\delta'(s,y) \geq \delta(s,y)$ (proved below)

$\delta'(s,x) = \delta'(s,y) + 1 \geq \delta(s,y) + 1 = \delta(s,x) + 2$

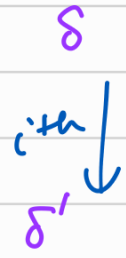
$\Rightarrow$ dist from s to v inc monotonically in G_f.

- Whenever an edge (x,y) reappears in the residual network, distance of x from s inc by atleast 2 units.

- any edge (x, y) can reappear / disappear $\leq \frac{n-1}{2}$ times in the algo.
- one while loop $\rightarrow O(m)$ and total while loops $= O(mn) \Rightarrow O(m^2 n)$

Proof for monotonically inc δ

Let x be the nearest defaulter. We assume



$\delta'(s,u) \geq \delta(s,u)$

$\delta(s,x) > \delta'(s,x)$

$u \rightarrow x$ must have appeared in i^{th} iteration

$\Rightarrow (u,x) \in P_i \Rightarrow \delta(s,x) = \delta(s,u) + 1$

$\delta(s,x) + 1 > \delta'(s,x) + 1 \Rightarrow (\delta'(s,u) + 1) + 1$

$\delta(s,u) > \delta'(s,u) + 2$ Contradiction