Max-flow:

Given a flow network, find a flow f of max size.

Ford-Fulkerson algorithm:

input: flow network with integer capacities c_e for $e \in E$.

1) Set $f_e = 0$ for all $e \in E$
2) Build residual network G_f:
   for $\overrightarrow{vw} \in E$:
   if $f_{vw} < c_{vw}$: add $\overrightarrow{vw}$ with residual capacity $c_{vw} - f_{vw}$
   if $f_{vw} > 0$: add $\overrightarrow{wv}$ with residual capacity f_{vw}
3) Check for a st-path P in G_f
4) If no st-path return (f)
5) Otherwise, let b = min residual capacity along P in G_f
6) Augment f by b units along P
7) Repeat, until no st-path in G_f

Running time:

Let C = size of max-flow

$O(m)$ time per round

& C rounds

$\Rightarrow O(mC)$ time

Want running time independent of capacities.

[Edmonds-Karp '72]: $O(mn)$ rounds

$\Rightarrow O(m^2 n)$ total time.

Any positive capacities (don't need to be integers).

In step (3), run BFS & take a shortest path P to augment

↑ fewest edges (so weights don't matter)

To argue $O(mn)$ rounds, we'll show:

a) in every round, ≥ 1 edge is deleted from G_f

& b) an edge is added or deleted $\leq n$ times.

Since are m edges, then $\leq nm$ rounds.

(a) is easy to see:

On P, ≥ 1 edge e is fully capacitated afterwards, because $b = mn$ residual capacity along P.

Then e is deleted afterwards.

For (b), let's first look at when an edge is added or deleted to G_f.

if $\overrightarrow{vw} \in P$ & is a forward edge (so $\overrightarrow{vw} \in E$) then:

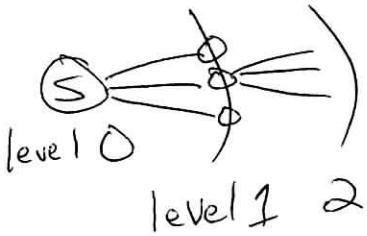
- $\overrightarrow{vw}$ is deleted if it becomes saturated (i.e. set $f_{vw} = c_{vw}$)
- $\overrightarrow{wv}$ is added if previously $f_{vw} = 0$

if $\overrightarrow{zv} \in P$ & is a backward edge then:

- $\overrightarrow{zv}$ is deleted if remove all flow along $\overrightarrow{vz}$ (i.e. set $f_{vz} = 0$)
- $\overrightarrow{vz}$ is added if it previously was full $f_{vz} = c_{vz}$

In G^f let level(v) = distance of v from s
↑ min # of edges (ignore weights)

level(s) = 0 & do BFS from s:



G^f is changing so level(v) may change.

Claim: level(v) does not decrease. (in Edmonds-Karp algorithm)

Proof:

Edmonds-Karp algorithm takes a shortest path P.

So $P = v_0 \to v_1 \to \cdots \to v_\ell$

where $v_0 = s$ & $v_\ell = t$

level(v_0) = 0

note level(v_{i+1}) $\leq$ level(v_i) + 1 by definition of level().

if level(v_{i+1}) $\not\leq$ level(v_i) then there is a shorter path to v_{i+1} & hence to t.

Thus, level(v_{i+1}) = level(v_i) + 1.

So level(v_i) = i for $0 \leq i \leq \ell$.

To $\downarrow$ level(v) we need to add an edge $\overrightarrow{uv}$ where level$(u) <$ level$(v) - 1$, so then we get s & v closer via u.

To add edge $\overrightarrow{uv}$ to G^f
then $\overrightarrow{vu} \in P$ but we know the level $\uparrow$ along P so level$(v) =$ level$(u) - 1$. ∎

So level(v) can stay the same or increase.

In each round ≥ 1 edge is deleted from G^f.
Suppose edge $\overrightarrow{vw}$ is deleted with level$(v) = i$ from G^f & level$(w) = i+1$.
& suppose later we add $\overrightarrow{vw}$ back into G^f.
— to add $\overrightarrow{vw}$ in to G^f we know
$\overrightarrow{wv} \in P$ & level$(v) =$ level$(w) + 1$.
Since level(w) didn't decrease,
level$(v) =$ level$(w) + 1 \geq i + 2$.

So if edge $\overrightarrow{vw}$ is deleted & later added back into G^f then

level(v) goes ↑ by ≥ 2.

$\Rightarrow$ an edge is deleted & added back in $\leq \frac{n}{2}$ times.

Since ≥ 1 edge is deleted each round

$\Rightarrow \leq nm$ rounds.

$\Rightarrow O(nm^2)$ total time. ∎

Best algorithm is $O(nm)$ total time by [Orlin '13]