Max-flow:

Monday 10/6/14

input: directed $G=(V,E)$, $s \in V$, $t \in V$, with for $e \in E$: capacity $c_e > 0$.

output: flow f of max size where

$$\text{Size}(f) = \sum_{w : \overrightarrow{sw} \in E} f_{sw}$$

Constraints:

for all $e \in E$: $0 \leq f_e \leq c_e$

for all $v \in V - \{s,t\}$: flow-in to v = flow-out of v

How does simplex work on the LP for Max-flow?

Start with $f_e = 0$ for all $e \in E$.

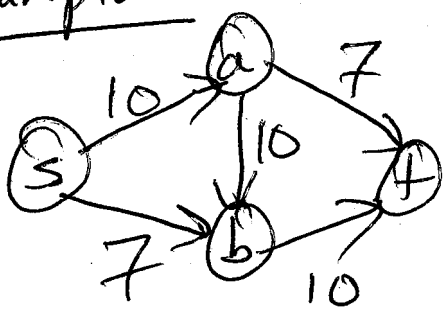
Try to make some edge constraint tight

i.e., for $e \in E$, make $f_e = c_e$.

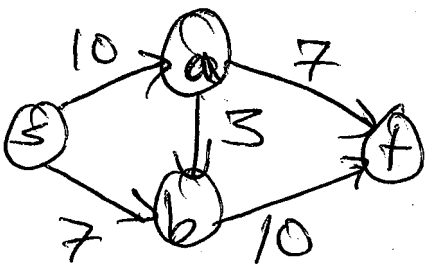
To do this, find an s-t-path P with available capacity & send as much flow as possible along P.

Example:

input =



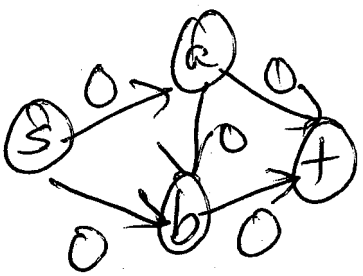
Solution =



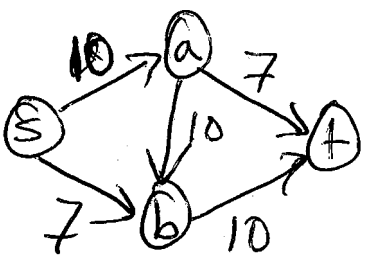
size(f) = 17

Algorithm:

start with =

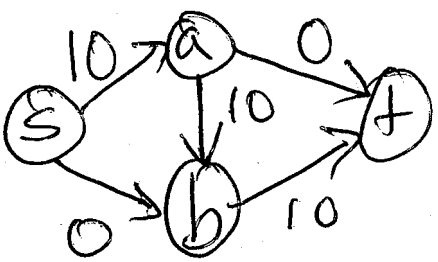


available =

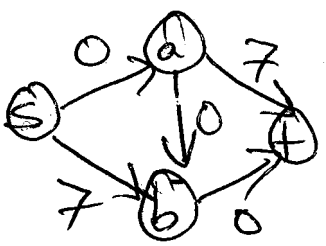


take path $s \to a \to b \to t$ & send 10 units

New flow =



available =



No s-t-path.

Can send flow "backwards" along edge $a \to b$

by decreasing its flow

So add edge $b \to a$ with available capacity $f_{ab} > 0$

For a flow network on $G = (V, E)$

with capacities c_e & a flow f_e

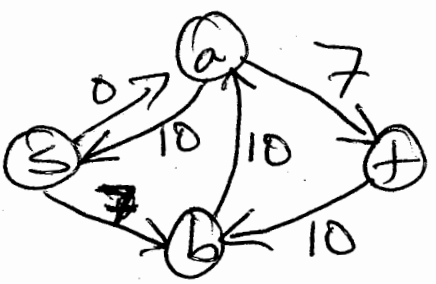
Residual network $G^f = (V, E^f)$

has edges $v \to w$ with residual capacity ~~r_{vw}~~ r_{vw}

where:

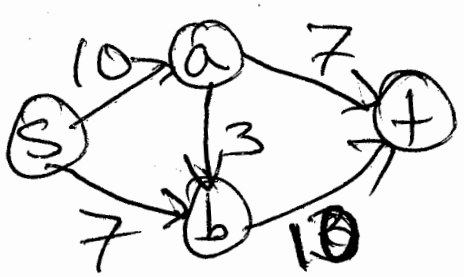
$$r_{vw} = \begin{cases} c_{vw} - f_{vw} & \text{if } \overrightarrow{vw} \in E \text{ \& } f_{vw} < c_{vw} \\ f_{wv} & \text{if } \overrightarrow{wv} \in E \text{ \& } f_{wv} > 0 \end{cases}$$

For earlier example, residual network =



Use path $s \to b \to a \to t$ to send 7 units

New flow =



which is optimal!

Ford-Fulkerson algorithm:

input: $G=(V,E)$ with integer capacities $c_e > 0$ for $e \in E$

1) Set $f_e = 0$ for all $e \in E$
2) Build residual network G^f
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

   i.e., $b = \min_{e \in P} r_e$

6) Augment f by b units along P:

   for each $\overrightarrow{vw} \in P$

   if $\overrightarrow{vw}$ is a forward edge (so $\overrightarrow{vw} \in E$)
   then $\uparrow f_{vw}$ by b

   if $\overrightarrow{vw}$ is a backward edge (so $\overrightarrow{wv} \in E$)
   then $\downarrow f_{wv}$ by b.

7) Repeat, until no st-path in G^f.

Running time:

If capacities are integers,

then flow increases by ≥ 1 unit per round

So if C = size of max flow

then $\leq C$ rounds.

each round uses a BFS/DFS,

So $O(|V|+|E|)$ per round.

$= O(|E|)$ assuming G is connected.

$\Rightarrow O(|E|C)$ total time

Pseudopolynomial running time since it depends on the numbers in the input

Edmonds-Karp ['72]: $O(|E||V|)$ rounds

$\Rightarrow O(|V||E|^2)$ total time.

Dinits ['70] showed $O(|V|^2|E|)$

No requirement on the capacities (don't need to be integers)

In G_f, run BFS & take a shortest path to augment

↑ # of edges (don't worry about any weights)

Observations about Ford-Fulkerson:

When can edges get added or deleted from G_f:

1) If $\overrightarrow{vw} \in P$ & is a forward edge then
   a) $\overrightarrow{vw}$ is deleted if it becomes saturated (i.e., if $b = r_{vw}$)
   b) $\overrightarrow{wv}$ is added if $\overrightarrow{vw}$ had zero flow ($f_{vw} = 0$)

2) If $\overrightarrow{zv} \in P$ is a ~~reverse~~ backward edge then
   a) $\overrightarrow{zv}$ is deleted if flow along $\overrightarrow{vz}$ is removed ($b = r_{zv} = f_{vz}$)
   b) $\overrightarrow{vz}$ is added if it was full before ($f_{vz} = c_{vz}$)

Note: New edges are reverse of edges in P (which are in G_f)

Argue that an edge can get added or deleted from G_f at most $O(n)$ times for Edmonds-Karp, and thus $O(mn)$ rounds $m = |E|$, $n = |V|$.

For a BFS tree T,
s = level 0, mark neighbors of s level 1,
their unmarked neighbors as level 2 etc.
For a shortest path P, where

$$P = v_0 \to v_1 \to v_2 \to \cdots \to v_{\ell-1} \to v_\ell$$

So $v_0 = s$ & $v_\ell = t$,
then v_i is on level i, otherwise there is a shorter path.
And max level is $\leq n$.

Claim: During the entire algorithm, for a vertex v, it's level(v) is non-decreasing.

Proof of claim:

for edge $\overrightarrow{vw}$ to get added to G^f
then $\overrightarrow{wv}$ is in P so $level(w) < level(v)$.

So this new edge $\overrightarrow{vw}$ goes from
higher level $\rightarrow$ lower level
$level(v) > level(w)$

So adding $\overrightarrow{vw}$ can't make any vertex closer to s & hence it doesn't $\downarrow$ any vertices level. ☐

At least one edge $\overrightarrow{vw}$ is deleted from G^f in each round because b is set to saturate at least one edge of P.

If $\overrightarrow{vw}$ is deleted with v on level i & w on level $i+1$.

We may add $\overrightarrow{vw}$ back in if $\overrightarrow{wv} \in P$ but then $level(w) < level(v)$ & since $level(w) \geq i+1$ by the claim, so when we add edge $\overrightarrow{vw}$ back in, $level(v) \geq i+2$.

Therefore, an edge can get deleted & added back in $\leq \frac{1}{2}$ times.

Since ≥ 1 edge is deleted per round & are $O(m)$ edges then $O(nm)$ rounds.

Each round takes $O(m)$ time to do a BFS.

So $O(nm^2)$ total time.

Best algorithms take $O(nm)$ time [Orlin '13], [King-Rao-Tarjan '94]

↑ was $O(nm)$ for $m = \Omega(n^{1+\epsilon})$.