# 9 March 2018

$n =$ # vertices
$m =$ # edges

## FORD-FULKERSON ALGORITHM

Preprocess $G$ to remove isolated vertices.
Initialize $f(e) = 0 \quad \forall e$

do

$O(m)$

compute residual graph $G_f$ — e.g. BFS or DFS, $O(m)$

if $G_f$ contains an augmenting path $P$

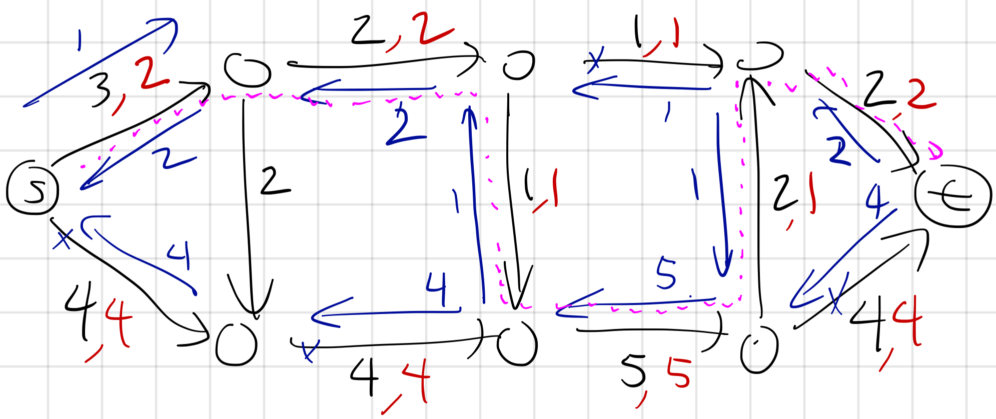
augment $f$ using $P$

end if

until $G_f$ has no augmenting path

output $f$

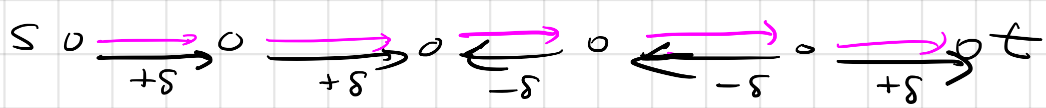
any choice of aug't path is considered a valid implementation of Ford-Fulkerson.




Terminate.

$v(f) = 6.$

Augmenting $f$ using $P$ preserves flow conservation and increases $v(f)$ by $\text{bottleneck}(f, P)$.




if $\text{bottleneck}(f, P) = \delta > 0$

Each iteration of the algorithm increases $v(f)$ by at least $1$.
Value $v(f)$ is initially $\emptyset$ and never exceeds

$$C \triangleq \sum_{e \text{ from } s} c(e)$$

Hence $\leq C$ iterations, and running time $O(\overset{\#\text{ edges}}{m}C)$.

pseudopolynomial!
(To be made polynomial next week.)

# Correctness of Ford-Fulkerson.

Recall termination condition: $G_f$ has no path from $s$ to $t$.

That means if we define
$$A = \{\text{vertices reachable from } s \text{ in } G_f\}$$
$$B = \{\text{vertices not reachable from } s \text{ in } G_f\}$$
then at termination, $s \in A$, $t \in B$, and $A, B$ constitute a partition of $V(G)$. i.e. $A, B$ is an $s$-$t$ cut.

Recall that for <u>any</u> flow $f'$, the flow-cut inequality says
$$v(f') \leq c(A, B)$$

So if we prove $\boxed{v(f) = c(A,B)}$ then $\forall f'$ $v(f') \leq v(f)$, i.e. $v(f)$ is maximum among all flows.

$$v(f) = \sum_{e \in E(A,B)} f(e) - \sum_{e \in \underline{E(B,A)}} f(e)$$

every $e \in E(A,B)$ has $c_f(e) = 0$

$G_f$ has no backward edges from $A$ to $B$, so $f(e) = 0$ $\forall e \in E(B,A)$

$\boxed{G_f \text{ has no edges from } A \text{ to } B}$

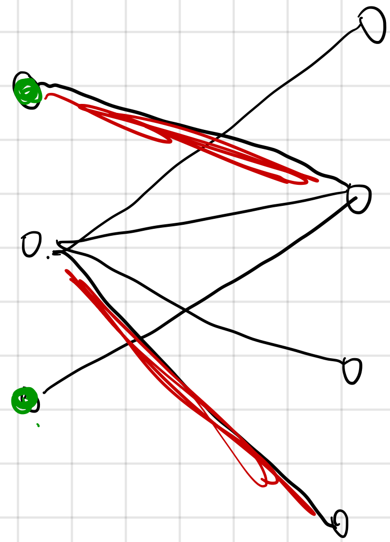
$$= \sum_{e \in E(A,B)} c(e) - \emptyset$$
$$= c(A, B)$$

<u>Theorem</u> (Max-Flow Min-Cut) In any flow network,
$$\max\{v(f) \mid f \text{ is a flow}\} = \min\{c(A,B) \mid (A,B) \text{ is an } s\text{-}t \text{ cut}\}.$$

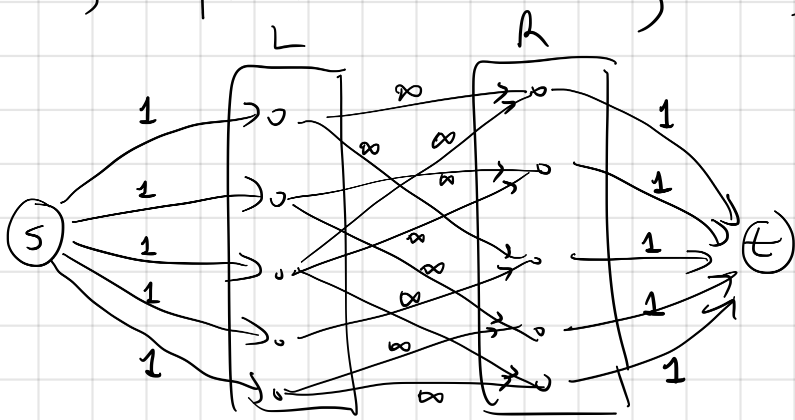
## A First Application: Maximum Bipartite Matching

Given an undirected bipartite graph $G = (V, E)$,
$V = L \cup R$, every edge has one endpt in $L$, one in $R$...
Find a matching of maximum cardinality.
↳ set of edges s.t. each vertex belongs to exactly 1.

At most 1 green vertex can belong to the matching.



Solving bipartite max matching using maximum flow:




Choosing edge $(u,v)$ to belong to the matching will be expressed by sending flow

$s \to u \to v \to t$

in this network.