Network Flows & Menger's Theorems
(Bondy & Murty, Chapter 11)

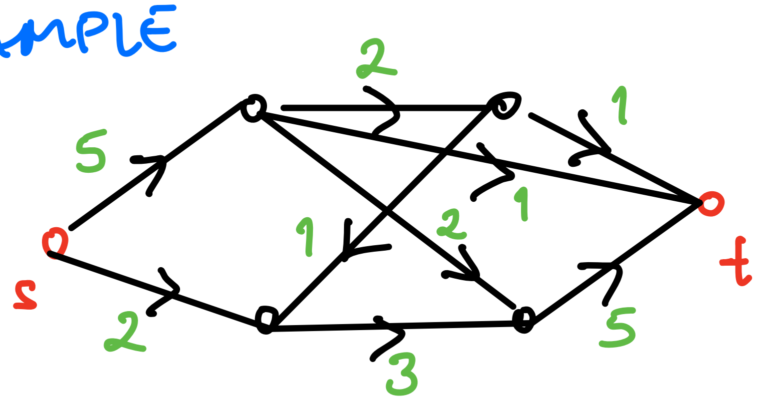
This is a vast generalization of Kuhn's Hungarian algorithm, that augments flow in a network rather than size of a matching.

DEFINITION: Let $D=(V,A)$ be a digraph with 2 distinguished vertices $\underset{\text{source}}{s}, \underset{\text{target (=sink)}}{t} \in V$,

and call $f: A \to \mathbb{R}_{\geq 0}$ an s-t flow if

$\forall x \in V - \{s,t\}$ one has $\displaystyle\sum_{\substack{\text{arcs}\\ \circ\xrightarrow{a}\circ x}} f(a) = \sum_{\substack{\text{arcs}\\ x\circ\xrightarrow{a}\circ}} f(a)$.

EXAMPLE

$D=$ 

flow values $f(a)$ shown

DEFINITION: Given a capacity function $c: A \to \mathbb{R}_{\geq 0}$ say that a flow f obeys c if $f(a) \leq c(a)$ $\forall a \in A$.

The value of the flow f is $v(f) := \displaystyle\sum_{\substack{\text{arcs } a\\ s\circ\xrightarrow{a}\circ}} f(a) - \sum_{\substack{\text{arcs } a\\ s\circ\xleftarrow{a}\circ}} f(a)$.

e.g. f above has $v(f) = 5+2 = 7$

PROBLEM: Given $D=(V,A)$ with $s,t\in V$ and $c: A\to \mathbb{R}_{\geq 0}$, want to find a flow $f: A\to\mathbb{R}_{\geq 0}$ maximizing $v(f)$.

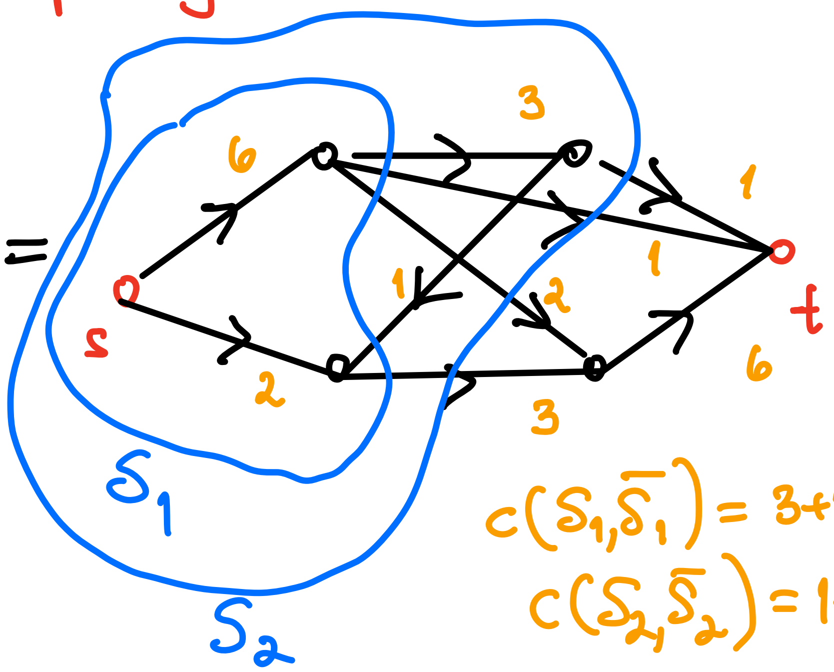
There are some fairly easy constraints...

DEFINITION: For $S\subset V$, define the

cut $A(S,\overline{S}) := \{a\in A : \underset{x\in S}{\circ} \xrightarrow{a} \underset{y\in\overline{S}}{\circ}\}$, where $\overline{S} = V\setminus S$,

and call it an s-t cut if $s\in S$, $t\in\overline{S}$.

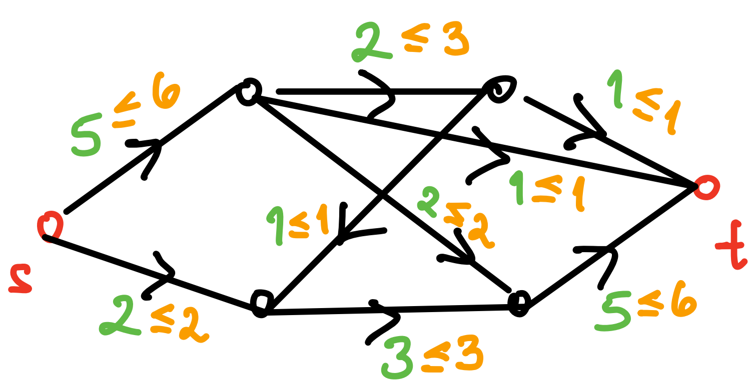
The capacity of the cut $c(S,\overline{S}) := \sum_{a\in A(S,\overline{S})} c(a)$

$D =$ 

$c: A\to\mathbb{R}_{\leq 0}$

$c(S_1,\overline{S}_1) = 3+1+2+3 = 9$

$c(S_2,\overline{S}_2) = 1+1+2+3 = 7$

$D =$ 

f obeys c with $v(f)=7$

PROPOSITION: For any $S \subset V$ with $s \in S$, $t \notin S$ and any s-t flow $f : A \to \mathbb{R}_{\geq 0}$, one has

(i) $$v(f) = \sum_{\substack{\text{arcs } a \\ x \xrightarrow{a} y \\ x \in S,\ y \in \bar{S}}} f(a) - \sum_{\substack{\text{arcs } a \\ x \xleftarrow{a} y \\ x \in S,\ y \in \bar{S}}} f(a) = \sum_{a \in A(S,\bar{S})} f(a) - \sum_{a \in A(\bar{S},S)} f(a)$$

and if f obeys $c : A \to \mathbb{R}_{\geq 0}$, then

(ii) $v(f) \leq c(S,\bar{S})$ with equality $\iff$

$$\begin{cases} f(a) = c(a) \ \forall a \in A(S,\bar{S}) & \text{"the cut is saturated"} \\ \text{AND} \\ f(a) = 0 \ \forall a \in A(\bar{S},S) & \text{"no backflow"} \end{cases}$$

---

proof: For (i), we rewrite

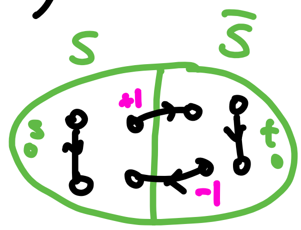
$$v(f) = \sum_{s \xrightarrow{a} \circ} f(a) - \sum_{s \xleftarrow{a} \circ} f(a)$$

$$= \sum_{s \xrightarrow{a} \circ} f(a) - \sum_{s \xleftarrow{a} \circ} f(a) + \sum_{x \in S - \{s\}} \Big(\underbrace{\sum_{x \xrightarrow{a} \circ} f(a) - \sum_{x \xleftarrow{a} \circ} f(a)}_{= 0 \text{ by flow property}} \Big)$$

(NOTE: $t \notin S$)

$$= \sum_{x\in S}\Big(\sum_{x \xrightarrow{a} \circ} f(a) - \sum_{\circ \xrightarrow{a} x} f(a)\Big)$$

$$= \sum_{a\in A} f(a) \Big(\sum_{\substack{x\in S:\\ x \xrightarrow{a} \circ}} (+1) + \sum_{\substack{x\in S:\\ \circ \xrightarrow{a} x}} (-1)\Big)$$



$$= \sum_{a\in A} f(a) \begin{cases} +1 & \text{if } a\in A(S,\bar{S}) \\ -1 & \text{if } a\in A(\bar{S},S) \\ 0 & \text{if } a\notin A(S,\bar{S})\sqcup A(\bar{S},S) \end{cases}$$

$$= \sum_{a\in A(S,\bar{S})} f(a) - \sum_{a\in A(\bar{S},S)} f(a).$$

But then (ii) follows from (i). $\blacksquare$

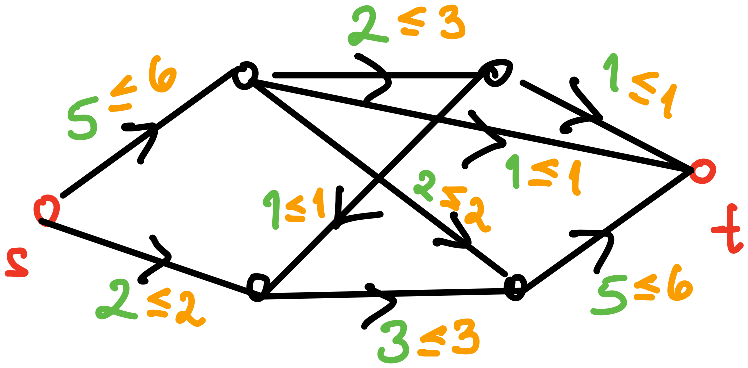
EXAMPLE:

f obeys c

$c(S_1,\bar{S}_1) = 3+1+2+3 = 9 > v(f)$

$c(S_2,\bar{S}_2) = 1+1+2+3 = 7 = v(f)$

$\Downarrow$

$D =$ 

f must be achieving the max flow while obeying c

Q: How to find a max-valued flow f obeying c?

Ford & Fulkerson's (1956) augmenting flow algorithm:

Given $D = (V, A)$ with an s-t flow f obeying c,

create a digraph $D_f = (V, A_f)$ as follows:

for each arc $x \xrightarrow{a} y$ in A, create

- the same arc $x \xrightarrow{a} y$ in A_f if $f(a) < c(a)$
- the opposite arc $x \xleftarrow{a} y$ in A_f if $0 < f(a)$
- both arcs $x \underset{a'}{\overset{a}{\rightleftarrows}} y$ if $0 < f(a) < c(a)$

Then do a breadth-first search for a directed $s \to \to \cdots \to t$ path P in D_f.

If such a P exists, augment the flow along P by ϵ

where $\epsilon := \min \left\{ \begin{array}{ll} c(a) - f(a) \text{ for } \underset{f(a)<c(a)}{x \xrightarrow{a} y} \text{ in } P \\ f(a) \text{ for } \underset{0<f(a)}{x \xleftarrow{a} y} \text{ in } P \end{array} \right\}$

by adding ϵ to $f(a)$ for $f(a) < c(a)$ of a in P

subtracting ϵ from $f(a)$ for $0 < f(a)$ of a in P.

Repeat until no such P exists in D_f.

This works because of ...

THEOREM (Ford & Fulkerson's "Max flow = Min Cut" Thm)

In a digraph $D=(V,A)$ with $s,t \in V$ and $c: A \to \mathbb{R}_{\geq 0}$,
an s-t flow $f: A \to \mathbb{R}_{\geq 0}$ obeying c maximizes $v(f)$

$\iff$ $D_f = (V, A_f)$ has no directed $s \to \to \ldots \to t$ paths

$\iff$ the set $S := \{x \in S : \exists$ a directed $s \to \to \ldots \to x$ path in $A_f\}$

gives a cut $A(S,\bar{S})$ which is saturated with no backflow, i.e. $v(f) = c(S,\bar{S})$.

In particular,

$$\max\{v(f) : f \text{ an s-t flow obeying } c\} = \min\{c(S,\bar{S}) : \begin{matrix} s \in S \subset V \\ t \notin S \end{matrix}\}.$$

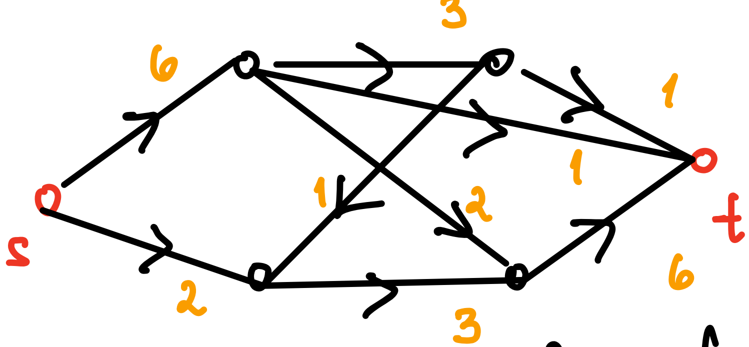
"max flow" (-value) = "min cut" (-capacity)

Also, one can find a max flow f_{max} algorithmically

in polynomial time, if $c: A \to \mathbb{Q}_{\geq 0}$ rational capacities

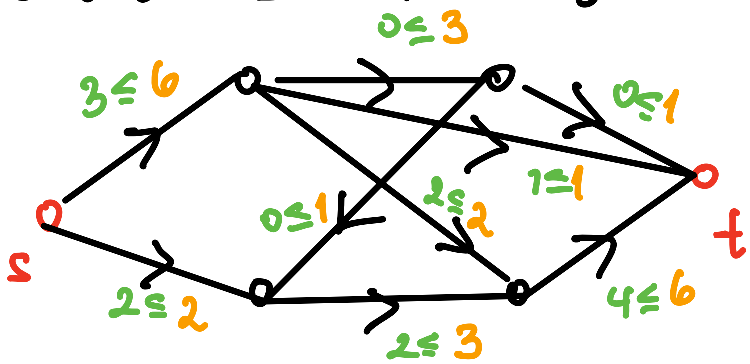
and $v(f_{max}) \in \mathbb{Z}_{\geq 0}$ if $c: A \to \mathbb{Z}_{\geq 0}$.

EXAMPLE Starting with D and $c: A \to \mathbb{R}_{\geq 0}$ as before



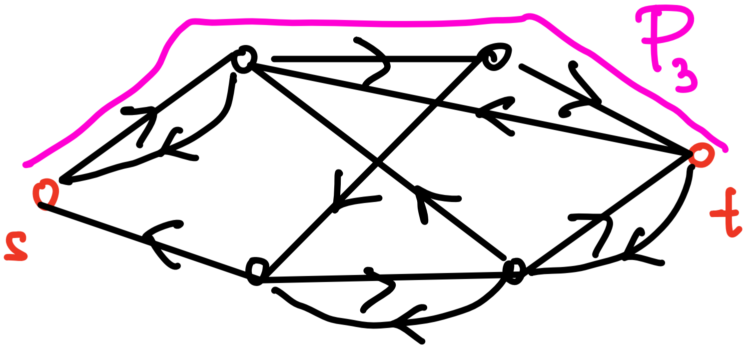
one can augment the zero flow f_0 three times to reach

$f_3 =$



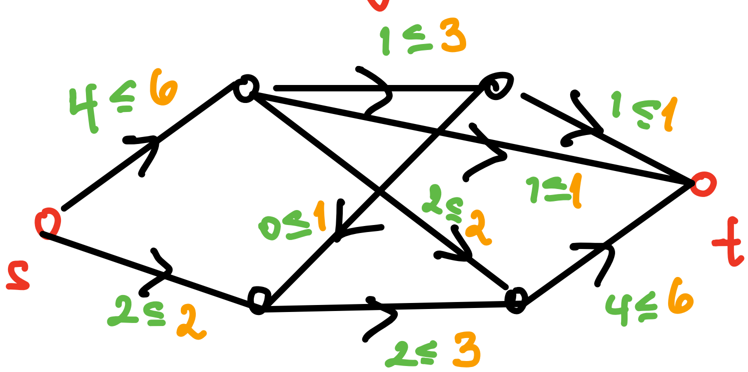
$v(f_3) = 5$

$D_{f_3} =$



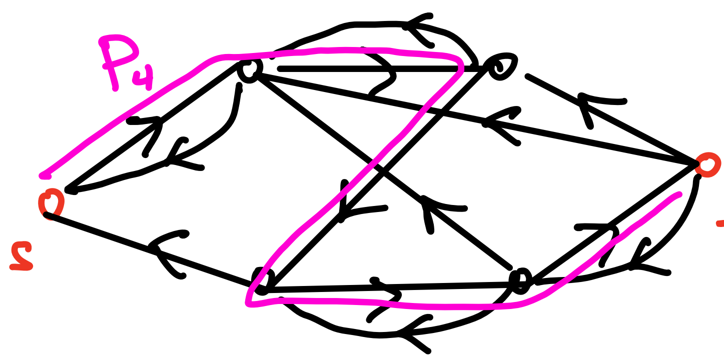
augment f_3 along P

$f_4 =$



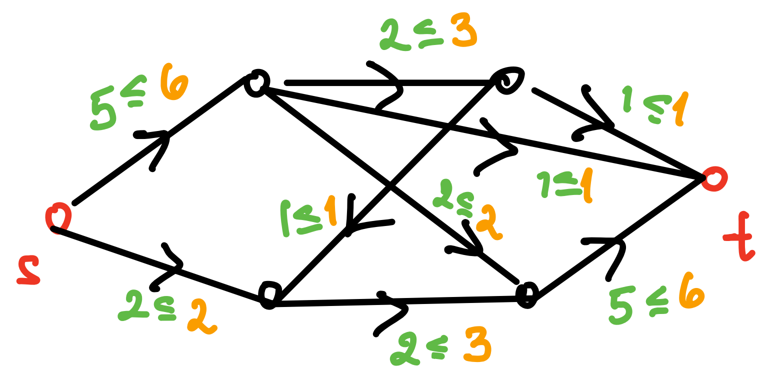
$v(f_3) = 6$

$D_{f_4} =$



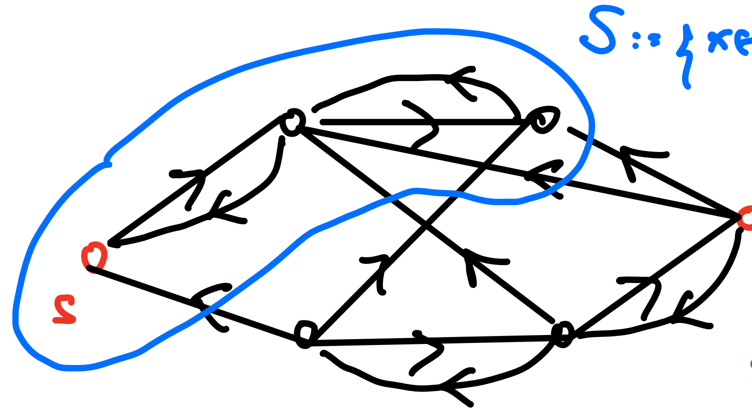
$\{$ augment f_4 along P_4

$f_5 =$



$v(f_5) = 7$

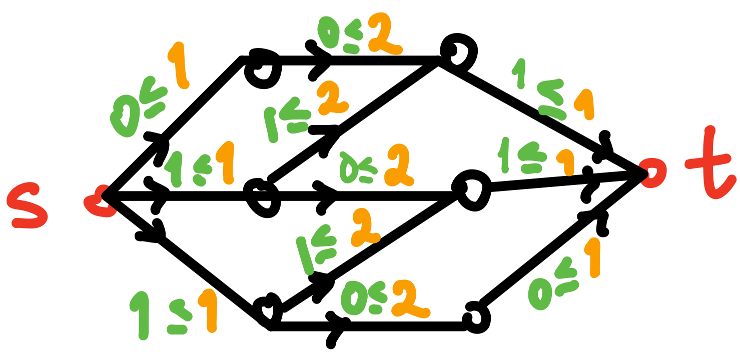
$D_{f_5} =$



$S := \{x \in V :$ reachable from s in $D_{f_5}\}$ gives a cut $A(S,\bar{S})$ which is saturated, with no backflow, so $v(f_5) = c(S,\bar{S})$ is a max flow.

ACTIVE LEARNING:

For this D with $c : A \to \mathbb{R}_{\geq 0}$ as shown,



and the given flow f,

(i) create the digraph D_f

(ii) use it to augment the flow f.

Proof of Ford-Fulkerson:

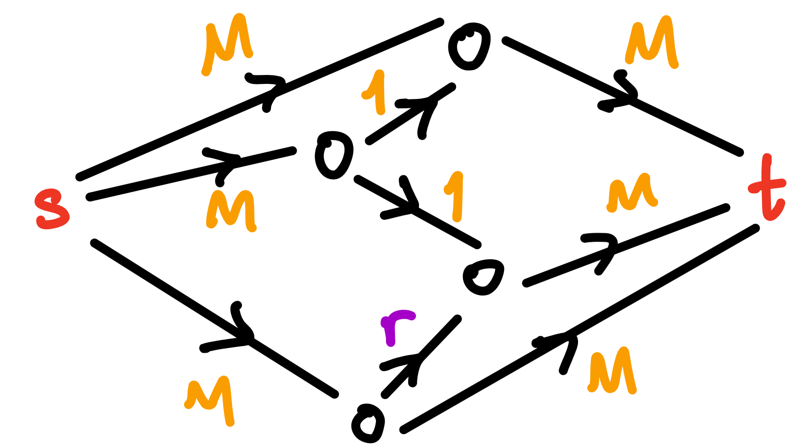
If D_f contains an $s \to \dots \to t$ path P, then one can augment f by ϵ, so $v(f)$ is not maximized.

If D_f has no $s \to \dots \to t$ path P, then
$S := \{x \in V : x \text{ has a path } s \to \dots \to x \text{ in } D_f\}$
gives a cut $A(S, \bar{S})$ whose capacity is saturated with no backflow, so $v(f) = c(S, \bar{S})$ and f achieves the max $v(f)$.

The rest follows ▨

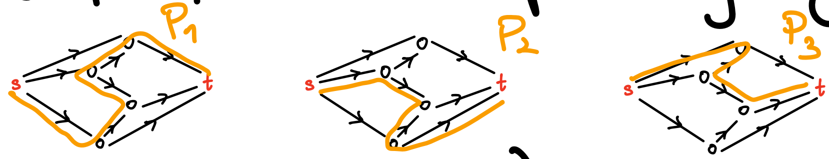
REMARKS

① Ford & Fulkerson gave this example of D with $c : A \to \mathbb{R}_{\geq 0}$ having one irrational capacity $c(a) \notin \mathbb{Q}$ leading to bad behavior in their algorithm:

$D =$ 

$c : A \to \mathbb{R}_{\geq 0}$
with $r := \frac{\sqrt{5} - 1}{2} \notin \mathbb{Q}$
(so $r^2 = 1 - r$)
and $M \geq 2$.

One can reach a flow f after which repeatedly augmenting P_1, P_2, P_1, P_3 here



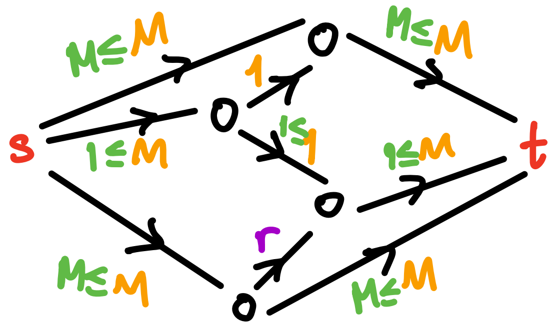
creates flows with values

$1+2(r+r^2)$
$1+2(r+r^2+r^3+r^4)$
$1+2(r+r^2+r^3+r^4+r^5+r^6)$
$\vdots$

not terminating, converging to $1+2(r+r^2+r^3+r^4+\dots)=3+2r$,

and there is a valid f_{max} with $v(f_{max})=2M+1$ $(>3+2r)$:

f_{max}



---

② For $G=(X\sqcup Y, E)$ bipartite,

creating $D = (V, A)$, with $V = \{s\}\sqcup X \sqcup Y \sqcup \{t\}$, A = shown below

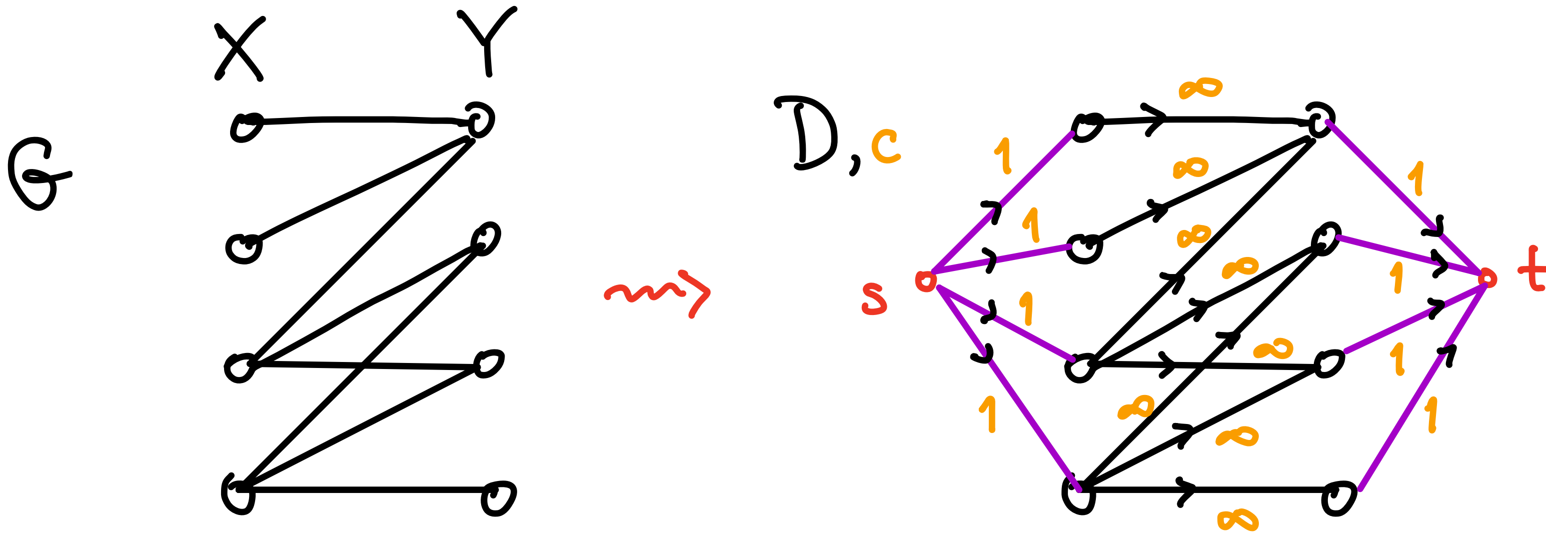
with capacities as shown below gives a network where

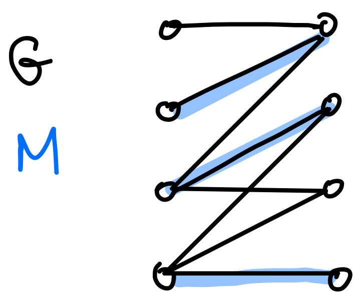
augmenting flow algorithm = Kuhn's Hungarian algorithm

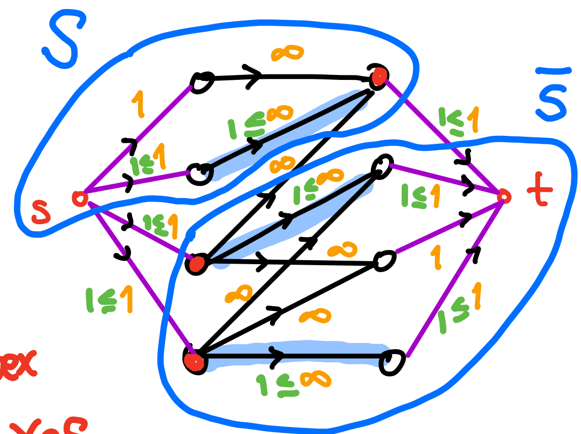
"max flow = min cut" Theorem = König-Egerváry Theorem

max size matching $\nu(G)$ = min size vertex cover $\tau(G)$

EXAMPLE







W = min vertex cover $= X \cap \bar{S} \sqcup Y \cap S$

③ Schrijver §4.6 describes a common generalization of Kuhn's maximum-weight bipartite matching algorithm and the Ford-Fulkerson augmenting flow algorithm, that finds a max flow with minimum cost given some cost function $k: A \to \mathbb{R}_{\geq 0}$, and where $\mathrm{cost}(f) := \sum_{a \in A} k(a) \cdot f(a)$

Connectivity (Chap. 3) & Menger's Theorem's (§11.4)

Trying to answer the question of how many vertices or edges must one remove from a connected graph G to disconnect it.

DEFINITION: For $k = 0, 1, 2, \ldots$ a multi graph $G = (V, E)$ is k-edge-connected if

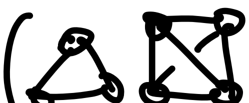
$|V| \geq 2$ and

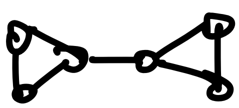
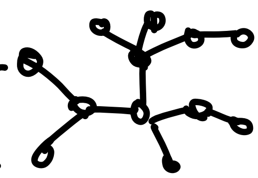
one must remove at least k edges to disconnect G.

The edge-connectivity

$$\kappa'(G) := \max\{k : G \text{ is } k\text{-edge-connected}\}$$
$$= \min \# \text{ of edges needed to remove to disconnect } G.$$

EXAMPLES

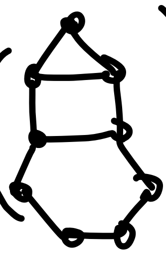
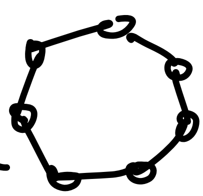
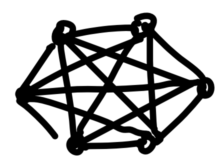
$\kappa'\Big($  $\Big) = 0$ and same whenever G is disconnected.

$\kappa'\Big($  $\Big) = 1 = \kappa'\Big($  $\Big) = \kappa'\Big($  $\Big)$

P_n

trees T with $|V| \geq 2$

C_n $n \geq 2$ K_n $n \geq 2$

$\kappa'\Big($  $\Big) = 2 = \kappa'\Big($  $\Big)$ $\kappa'\Big($  $\Big) = n-1$

DEFINITION: For $k = 0, 1, 2, \ldots$

a simple graph $G = (V, E)$ is **k-vertex-connected** if

either $G = K_m$ for some $m \geq k+1$

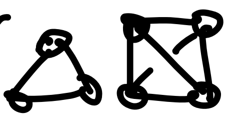
or $G \neq K_m \ \forall m$, and it requires removing at least k vertices to disconnect G.

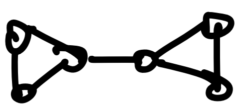
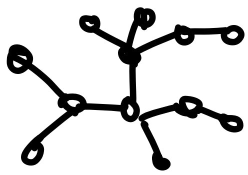
The **vertex-connectivity**

$\kappa(G) := \max\{k : G \text{ is } k\text{-vertex-connected}\}$

$= \min \#$ of vertices needed to remove to **disconnect** G.

EXAMPLES

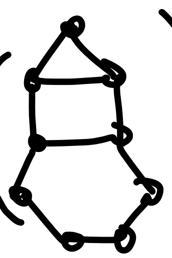
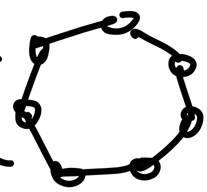
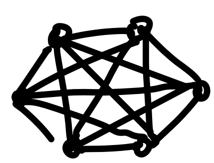
$\kappa\Big($  $\Big) = 0$ and same whenever G is **disconnected**.

$\kappa\Big($  $\Big) = 1 = \kappa\Big($  $\Big) = \kappa\Big($  $\Big)$

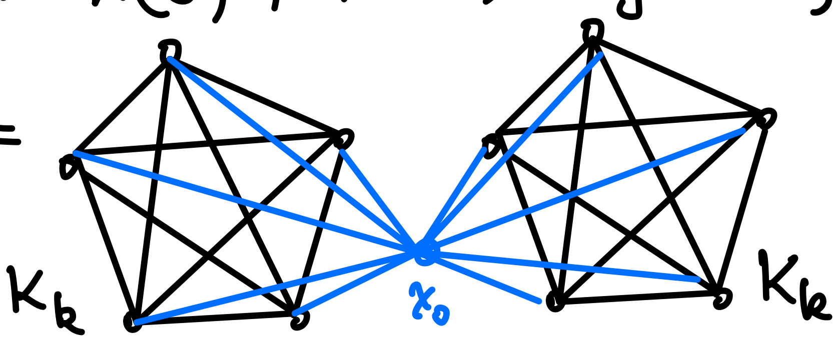
P_n

trees T with $|V| \geq 2$

$C_n \quad n \geq 3$ $\qquad\qquad$ $K_n \quad n \geq 2$

$\kappa\Big($  $\Big) = 2 = \kappa\Big($  $\Big)$ $\qquad$ $\kappa\Big($  $\Big) = n-1$

But $\kappa(G) \neq \kappa'(G)$ in general, e.g.

$G =$ 

has $\kappa(G) = 1$

$< \kappa'(G) = k$

ACTIVE LEARNING: Let $\delta(G) := \min\{d_G(x) : x \in V\}$
$= \min$ vertex degree in G.

(a) Prove $\kappa'(G) \leq \delta(G)$ $\forall$ multigraphs $G=(V,E)$ with $|V| \geq 2$.

(b) Prove $\kappa(G) \leq \delta(G)$ $\forall$ simple graphs $G=(V,E)$ with $|V| \geq 2$.

We'll see later from Menger's Theorems how to compute $\kappa'(G), \kappa(G)$ and also that $\kappa(G) \leq \kappa'(G)$.

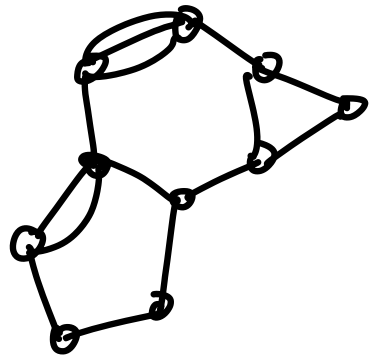
DEFINITION: A multigraph $G=(V,E)$ is called

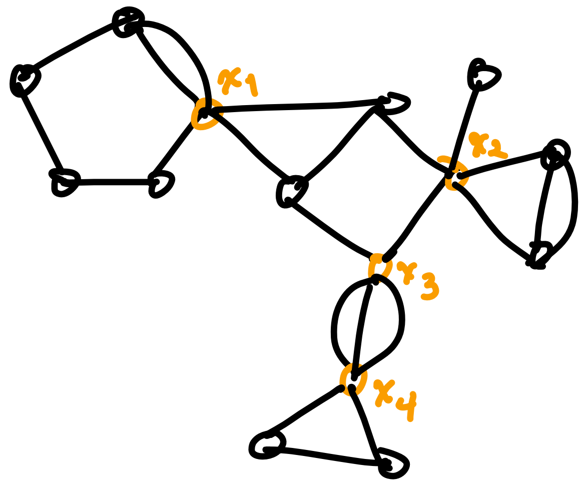
biconnected or 2-connected or a block — when it is 2-vertex-connected, i.e., connected and containing no articulation or cut-vertices $x \in V$

← vertices x for which $G - \{x\}$ has more components than x

When G is connected but not 2-connected $(\kappa(G)=1)$, one defines the blocks $G_i = (V_i, E_i)$ of G by removing each cut vertex $x_1, x_2, \ldots, x_\ell$ and creating a new copy of x_i to go in each new component that removing it creates. One then records how the blocks G_i attached at the cut-vertices with the bipartite block-cutvertex tree of G.

EXAMPLES:

$G =$  a block

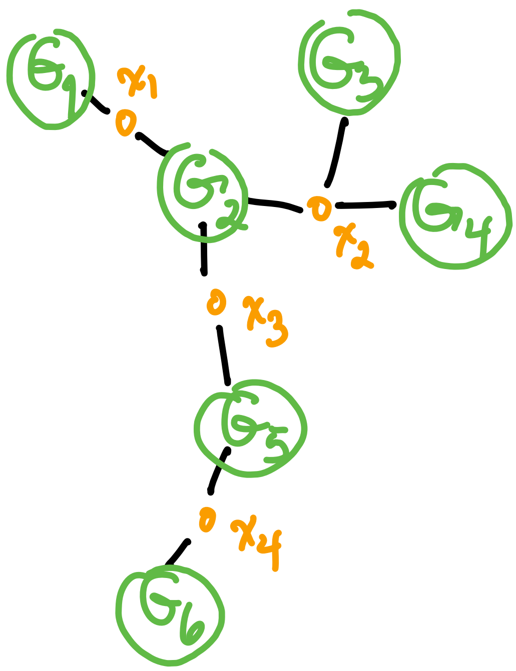
$G =$ 

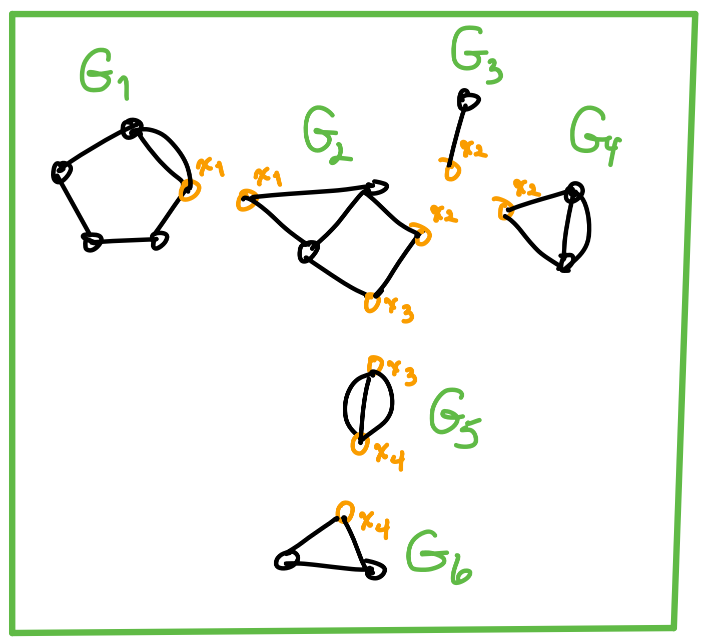
is not itself a block:

G has cut-vertices x_1, x_2, x_3, x_4

creating 6 blocks $G_1, G_2, G_3, G_4, G_5,$

block-cut-vertex tree:





ACTIVE LEARNING:

Explain why the block-cut-vertex bipartite graph is always a tree.

Many graph algorithms as their

- first step use breadth first search on G to find its connected components
- within each connected component find its blocks and block-cut-vertex tree.

An algorithm of Hopcroft & Tarjan (1973) does the latter quickly.

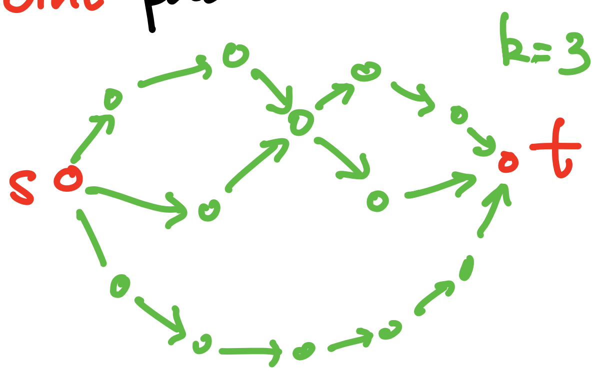
THEOREMS$^{(4)}$: (Menger 1927)

2 Arc/edge versions:

Fix a digraph $D=(V,A)$
or multigraph $G=(V,E)$ with chosen vertices $s,t \in V$.

Then it requires removing at least k arcs (edges)
to destroy all directed (undirected) $s \to t$ paths

$\iff \exists$ k arc-disjoint (edge-) paths $s \to t$

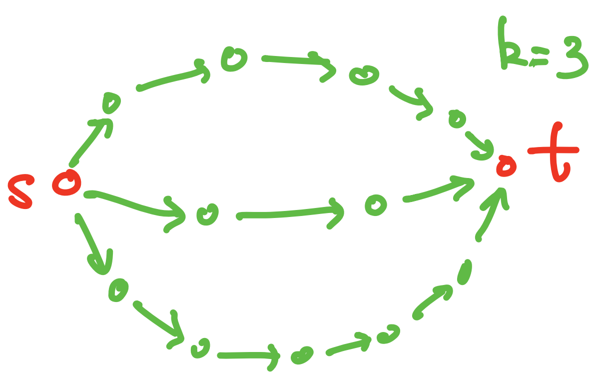


2 Vertex versions:

Assume same hypotheses, except $s \neq t$ and s,t nonadjacent (!)

Then it requires removing at least k vertices
to destroy all directed (undirected) $s \to t$ paths

$\iff \exists$ k vertex-disjoint paths $s \to t$



COROLLARY:

For $k \geq 2$, $G=(V,E)$ is k-vertex-connected
(k-edge-connected)

$\iff |V| \geq 2$ and between any $s,t \in V$
$\exists$ k vertex-disjoint paths $s \to t$

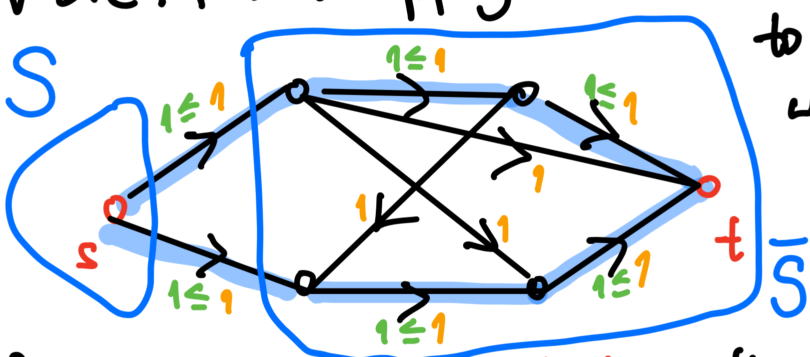
In particular, $\kappa(G) \leq \kappa'(G)$, since
vertex-disjoint paths are always edge-disjoint.

---

proof of Menger's Theorems:

Digraph arc version:

Given $D=(V,A)$ and $s,t \in V$, assign capacities
$c(a)=1$ $\forall a \in A$ and apply the MaxFlow = Min Cut Thm.
to a max flow f
with all $f(a) \in \mathbb{Z}$,
so $f(a) \in \{0,1\}$.



A flow f obeying c with $v(f)=k$ is the same as
k arc-disjoint $s \to t$ paths

An s-t-cut $(S,\bar{S})$ with $c(S,\bar{S})=k$ is the same as
a choice of k arcs whose removal destroys all $s \to t$ paths.

Hence MaxFlow = Min Cut is Menger in this case.

Undirected graph edge version:

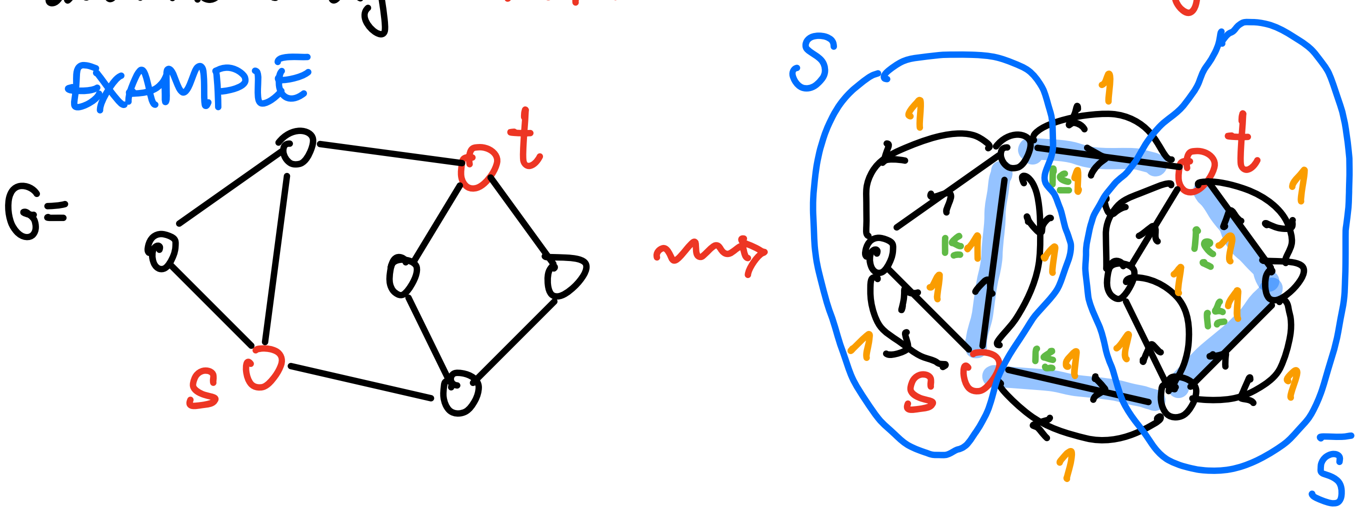
Given $G=(V,E)$ create $D=(V,A)$

via

and again assign capacities $c(a)=1 \ \forall a \in A$.

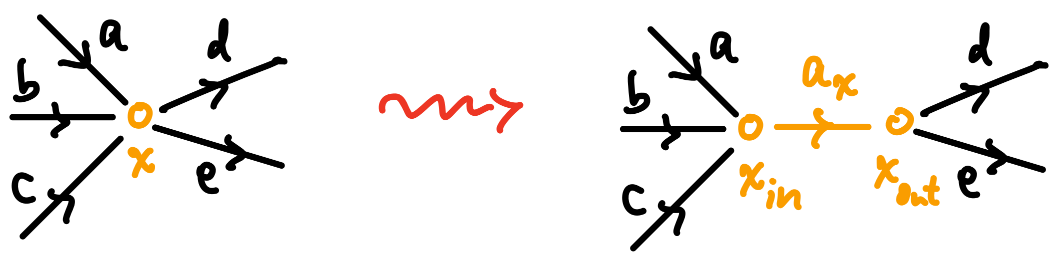
One can check that a max flow f obeying c never uses flow along both arcs a_e, a'_e that come from some e, and hence again Max Flow = Min Cut is Menger here.

EXAMPLE



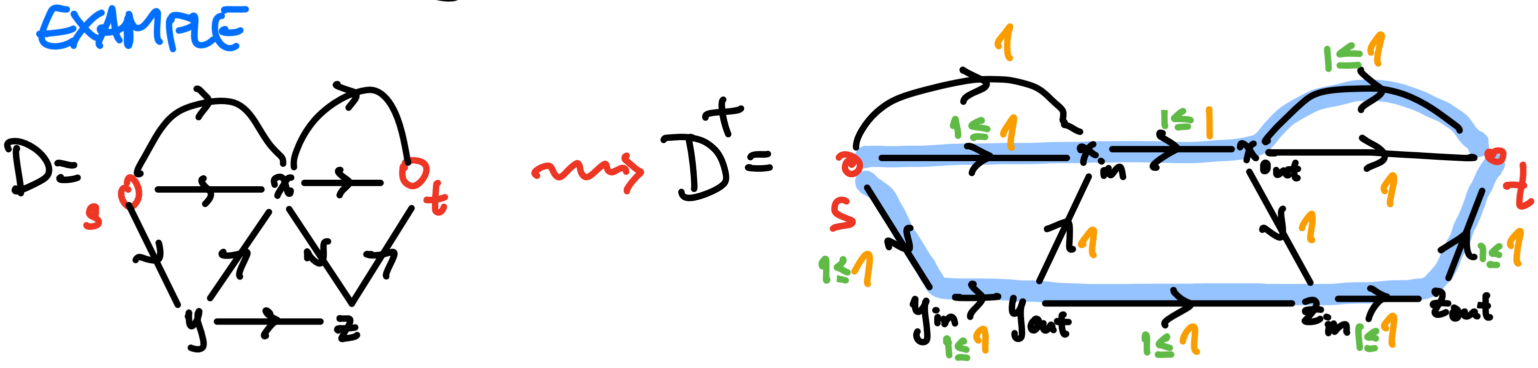
Digraph vertex version:

Given $D=(V,A)$ and $s,t \in V$, create $D^+=(V^+, A^+)$ having for each $x \in V-\{s,t\}$ two vertices x_{in}, x_{out} as follows:



Again assign capacities $c(a)=1 \ \forall a \in A^+$, and apply MaxFlow = MinCut to D^+, to any max flow f with values $f(a) \in \{0,1\}$.

EXAMPLE

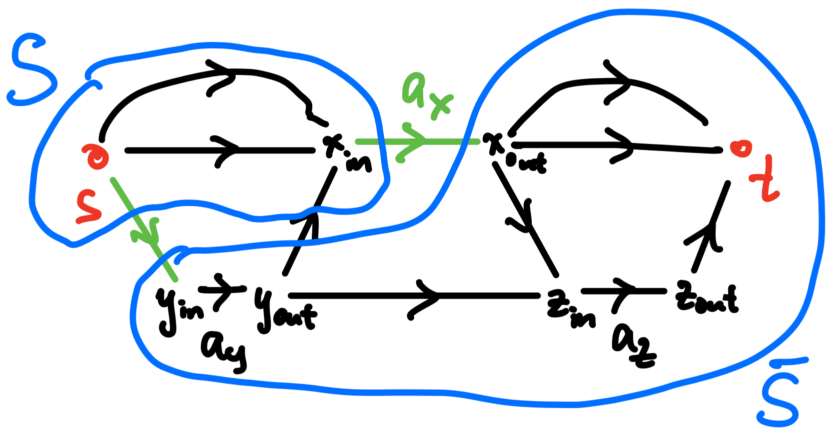


It's still easy to see that such a flow f with $v(f)=k$ corresponds to k arc-disjoint $s \to t$ paths in D^+ which corresponds to k vertex-disjoint $s \to t$ paths in D.

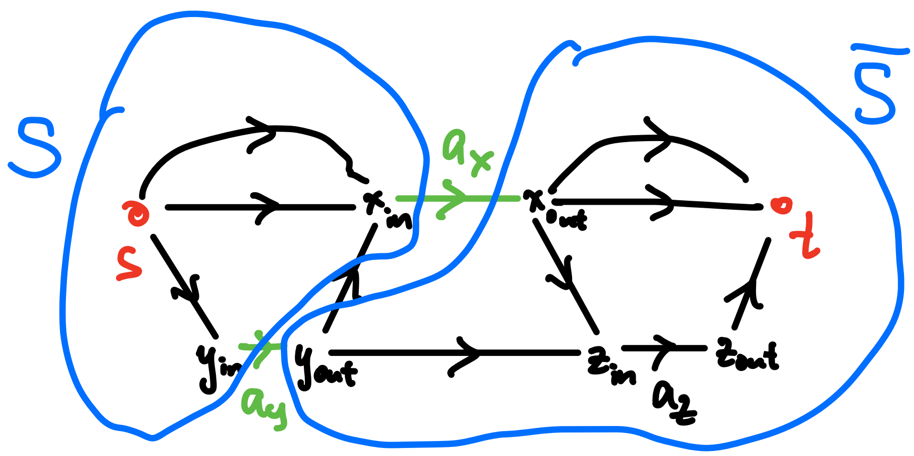
One has to be a bit more careful to check that an s-t cut $A(S,\bar{S})$ in D^+ with $c(S,\bar{S})=k=v(f)$ can be altered to give one that only uses a_x arcs $x_{in} \xrightarrow{a_x} x_{out}$ by pushing the other arcs forward or backward.

Then the s-t cuts $A(S,\bar{S})$ using only a_x arcs correspond to vertices whose removal destroys all $s \to t$ paths.

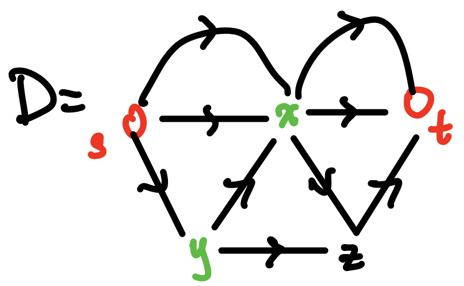
EXAMPLE:



push $\rightsquigarrow$ forward



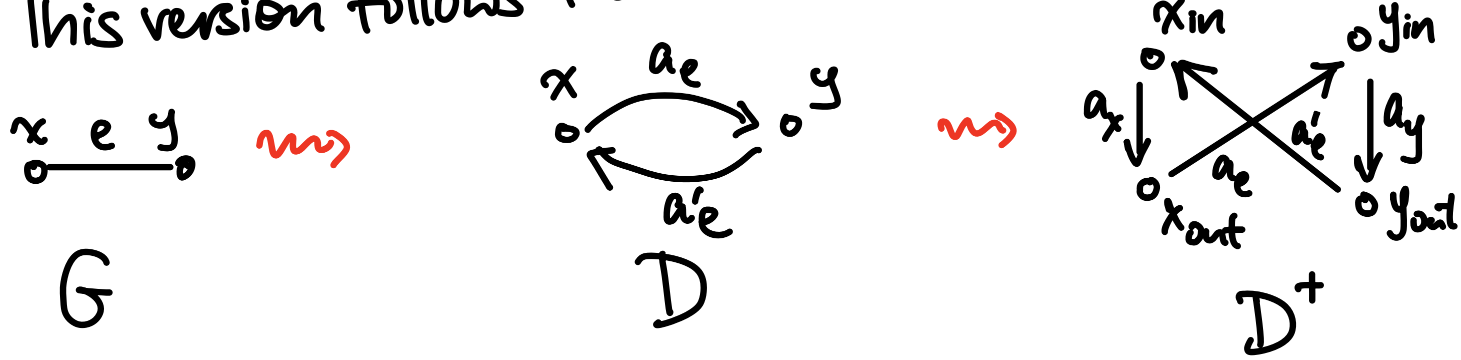
which corresponds to $\{x, y\}$ whose removal from



destroys all $s \to t$ paths.

Graph vertex version:

This version follows from the construction



similarly to the arc/edge versions. ∎

COROLLARY: For a multigraph $G=(V,E)$, the vertex and edge connectivities

$$\kappa(G), \quad \kappa'(G)$$

can be computed in polynomial time

as
$$\kappa(G) = \min_{s,t\in V} \left\{ \begin{array}{l} \text{max flow value } v(f) \\ \text{in the digraph } D \text{ for } G,s,t \end{array} \right\}$$

$$\kappa'(G) = \min_{s,t\in V} \left\{ \begin{array}{l} \text{max flow value } v(f) \\ \text{in the digraph } D^{+} \text{ for } G,s,t \end{array} \right\}$$

(using $|V|^2$ instances of Ford-Fulkerson algorithm).