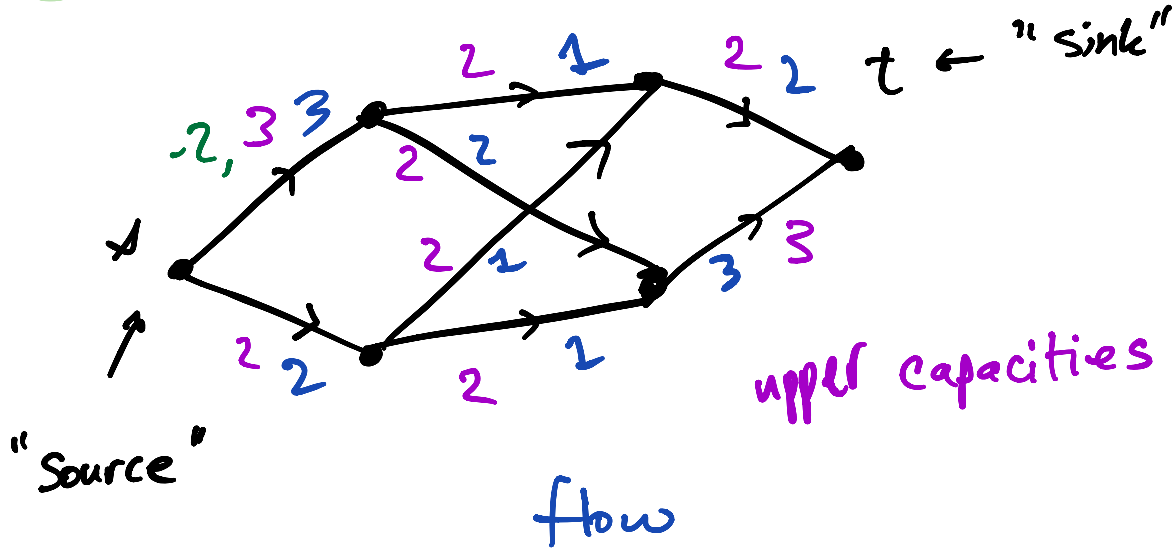
# Lecture 12

Quiz: Mean 92%
median 97%

**Plan:** 1) Def, examples
2) max-flow, min-cut

**Next time:** Fast Algorithms for flows

**Example:** water pipes




- Except at source & sink, water going in must go out.
- flow cannot exceed capacity.

**Question:** how much water can be sent through?

- Obviously can also model electricity, traffic, etc.

more formally:

**Def (Flow Network)**

- $G = (V, E)$ directed graph.
- Two special vertices $s, t \in V$.
  "source" "sink"
- $u : E \to \mathbb{R}$ "upper capacity".
- $l : E \to \mathbb{R}$ "lower capacity".
  (if omitted, default is $l = 0$).

- A flow is function $x: E \to \mathbb{R}$

satisfying

$$\boxed{l(e) \le x_e \le u(e) \quad \forall e \in E}$$

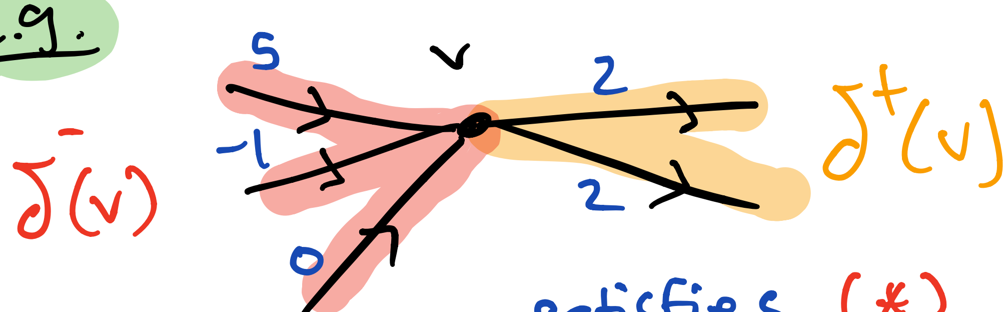
and "flow conservation"

$$\boxed{(*) \quad \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e \qquad \forall v \in V \setminus \{s,t\}.}$$

edges leaving v

edges entering v.

E.g.




satisfies (*).

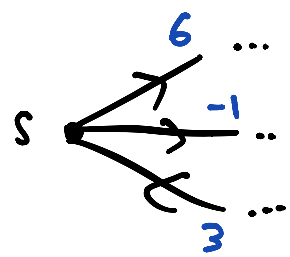
- flow network feasible if it admits any flow.

feasible $\Rightarrow$ $l(e) \le u(e)$.

- **Value** of flow $x$ is

$$\boxed{|x| := \sum_{\delta^+(s)} x_e - \sum_{\delta^-(s)} x_e}$$

**e.g.**

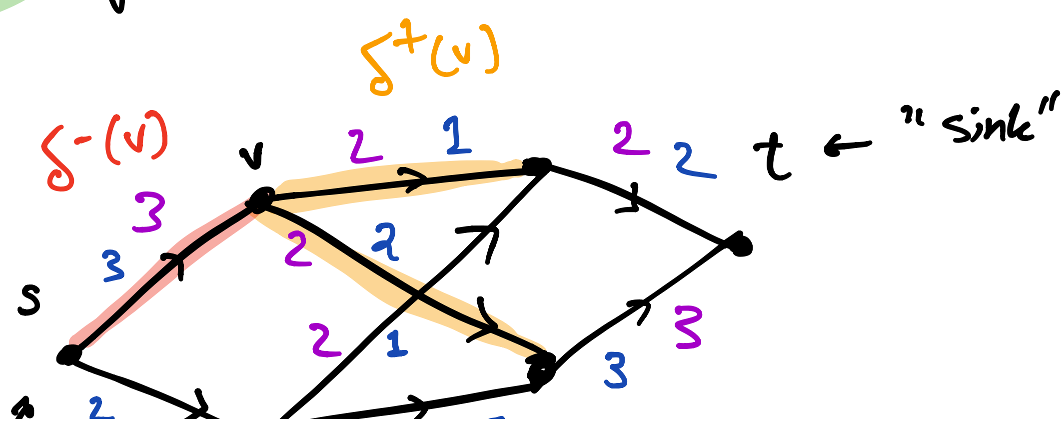


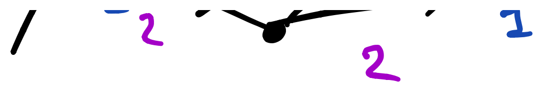

value $(6-1)-(3) = 2.$

- **Maximum flow problem:**

$\boxed{\text{find flow } x \text{ w/ largest value } |x|.}$

**E.g.** (flow from earlier).






"Source"

flow $x$ has value 5

(upper) capacity $u$
lower $l = 0$.

$x$ is maximum!

# Notes

- $|x|$ = amt. <u>leaving</u> source $s$

can also be expressed in terms of sink:

$|x|$ = amt. <u>entering</u> sink $t$

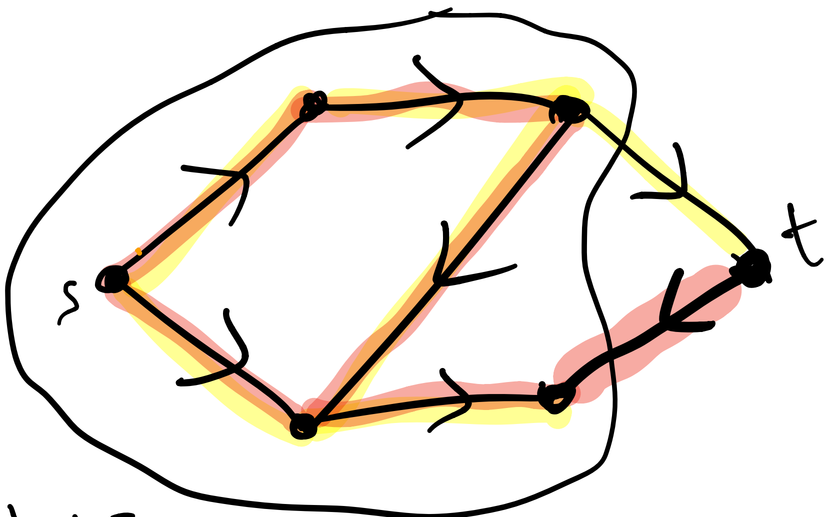
$$= \boxed{\sum_{\delta^-(t)} x_e - \sum_{\delta^+(t)} x_e}$$

<u>Pf</u>:

$$|x| = \sum_{u \in V \setminus t} \left( \sum_{e \in \delta^+(u)} x_e - \sum_{e \in \delta^-(u)} x_e \right)$$

flow cons ⇒ only $u = s$ is nonzero.

$$= \sum_{\delta^-(t)} x_e - \sum_{\delta^+(t)} x_e .$$




$V \setminus \{t\}$

- max flow problem is an LP.

$$\max \sum_{e \in \delta^+(s)} x_e - \sum_{e \in \delta^-(s)} x_e$$

subject to

$$\sum x_e - \sum x_e = 0 \quad \forall v \in V \setminus \{s,t\}$$

$$\sum_{e\in\delta^+(v)} x_e \quad \sum_{e\in\delta^-(v)}$$

and $\ell(e) \le x_e \le u(e) \quad \forall e\in E.$

**Theorem** If $\ell, u$ integral, is <u>integral</u> max flow!* (IP=LP).

*if flow network feasible.

**Proof**: Total unimodularity!

Express constraints in matrix form:

$$\max\{c^T x : \begin{array}{l} Nx \boxed{=} 0 \\ Ix \le u \\ x \ge \ell \end{array}\}$$

flow cons

capacity

So let

$$A = \begin{bmatrix} N \\ \hline I \end{bmatrix} \qquad P = \{x : Ax \,\Delta\, b, \; x \geq \ell\}.$$

why sufficient that $A$ TU?

**Exercise:**

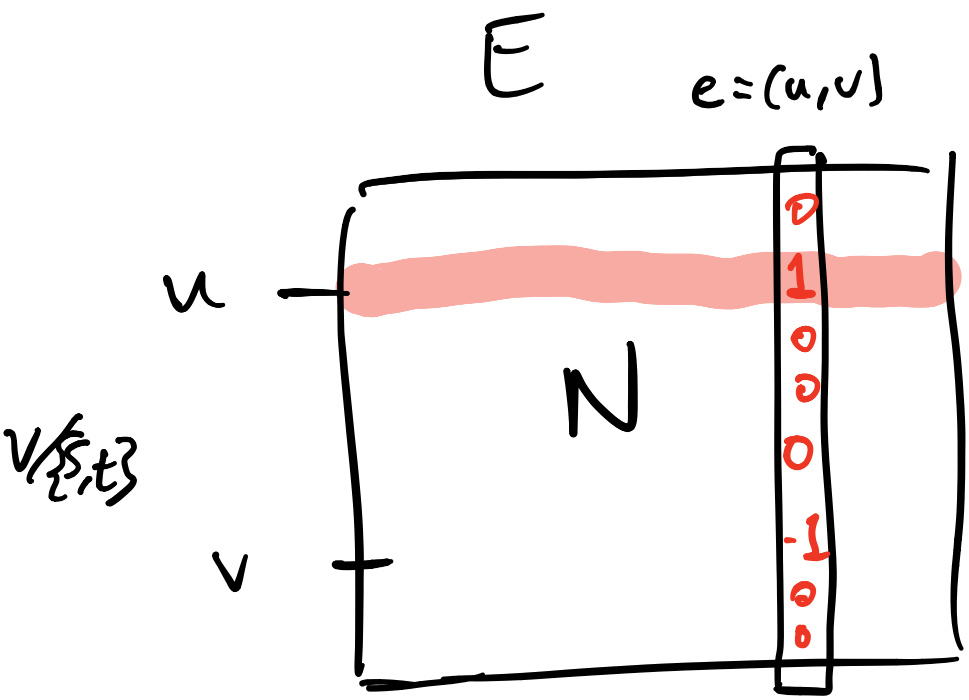
$$P = \{x : Ax \,\Delta\, b, \; x \geq \ell \}$$

is integral if $A$ is TU whenever $\Delta$ has $\geq, \leq, =$ and $b$ and $\ell$ integral. (shift $P$ to get rid of $\ell$).

**Showing $A$ is TU:**

- Enough to show $N$ is TU.
  (just expand down rows of $I$ that appear).

- What is $N$?




$N$ is <u>transpose</u> of submatrix of "directed incidence matrix of $G$.

i.e.

$$N_{ue} = \begin{cases} 1 & \text{if } e \in \delta^+(u) \\ -1 & \text{if } e \in \delta^-(u). \end{cases}$$

- Directed incidence matrices (and their transposes) are TU: Quiz extra credit :)

3 Cases for submatrix $M$ of $N$:

(i) Col of all 0: $\boxed{\det M = 0}$

(ii) Col of one $\pm 1$: expand down it, get smaller submatrix.

(iii) Every column has one $+1$, one $-1$, rest 0's.
sum of rows is zero.

$$\boxed{\mathbb{1}^T M = 0 \Rightarrow \det M = 0.}$$

Could also use discrepancy. □

---

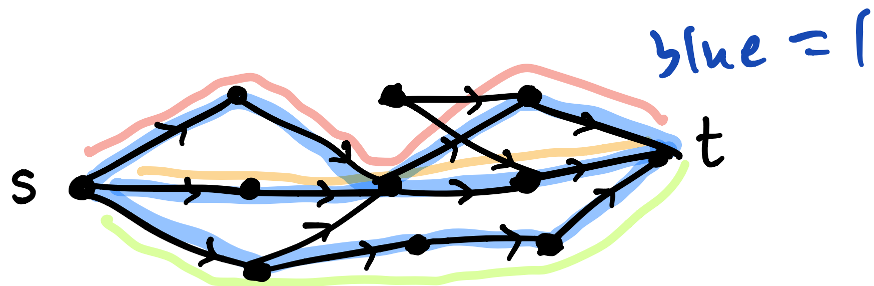
# Special cases:

## 1) Edge-disjoint paths:

max flow is max # edge-disjoint s-t paths in $G$.

if we set $\boxed{\ell = 0 \quad u = \mathbb{1}}$

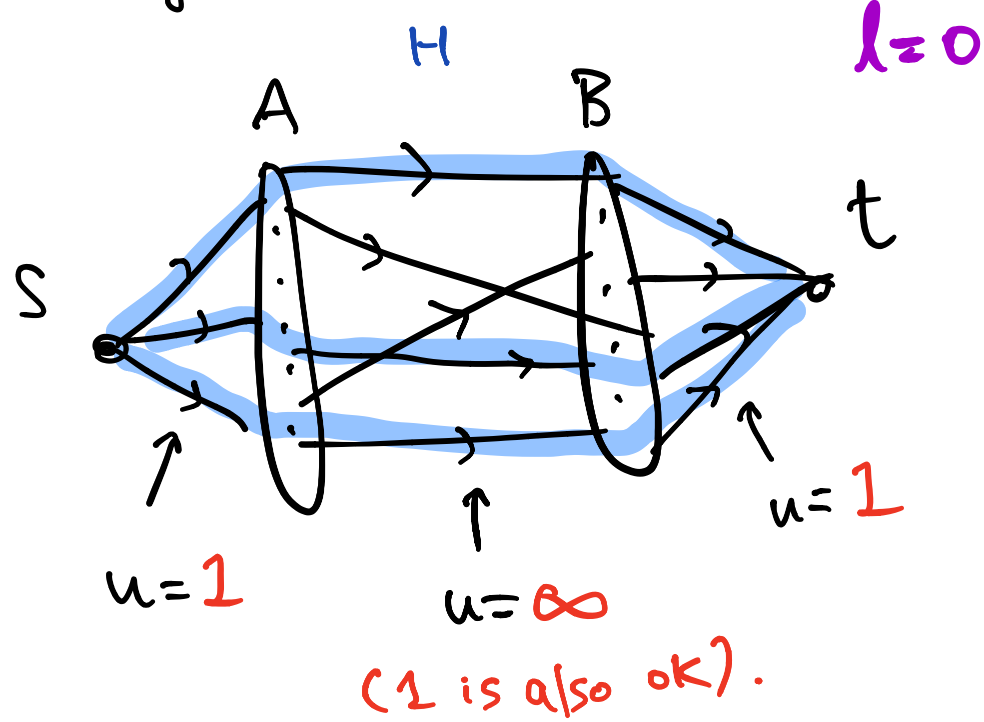
why? Know $\exists$ integer max flow; so takes values in 0-1



Perform flow decomposition:
Remove paths 1 at a time.

## 2) Bipartite matchings

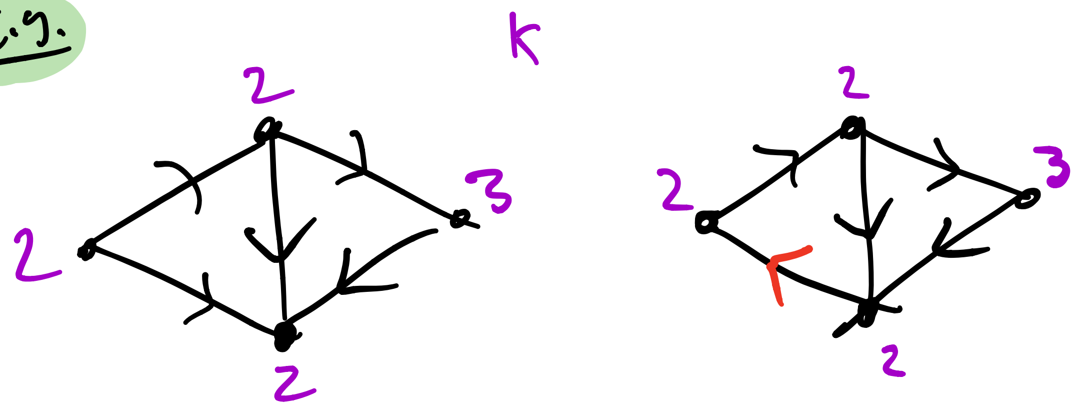
$H=(A,B,E)$ bipartite graph
Direct edges across, add vertices s,t.




integer flow $x$ $\longleftrightarrow$ matching of size $|x|$.

**3) Orientations:** Given directed graph $G$, orient edges so indegree of each vertex $v \le k(v)$.

**E.g.**




**Ex:** Express as max flow.

# s-t cuts

<u>dual</u> of maxflow LP leads

to the notion of cuts.

"cuts obstruct flows"

**DEFS:**

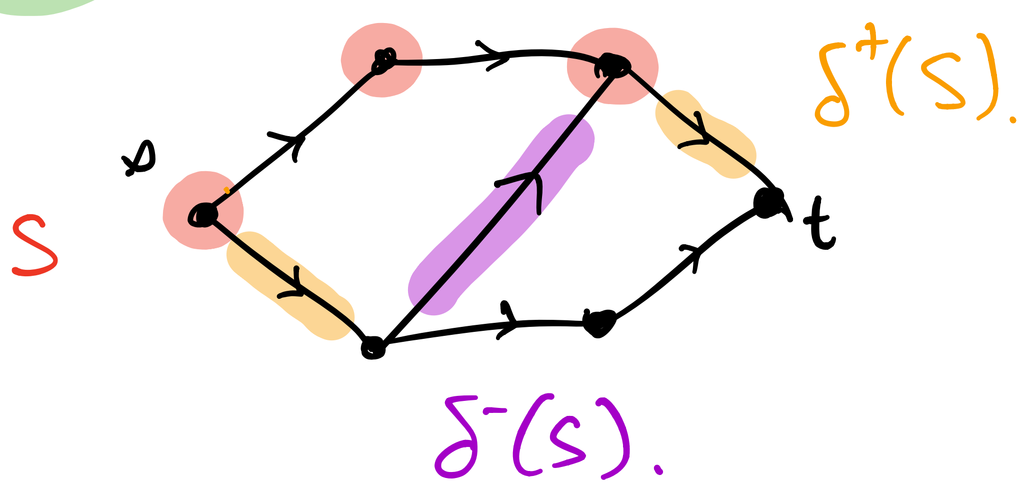
- Given digraph $G=(V,E)$, $S \subseteq V$,

**Cut** is set of edges leaving $S$:

$$\delta^+(S) = \{(u,v) \in E : u \in S,\ v \in V \setminus S\}.$$

$$\delta^-(S) := \delta^+(V \setminus S).$$

**E.g.**



- Abuses of notation:

  (i) for $v \in V$, $\delta^{\pm}(v) := \delta^{\pm}(\{v\})$

  (ii) Conflate $S$, $\delta^+(S)$.

- $S$ is **s-t cut** if $s \in S$, $t \notin S$.

how do cuts bound flows?

- Given flow network $G=(V,E)$, capacities $u, l$,

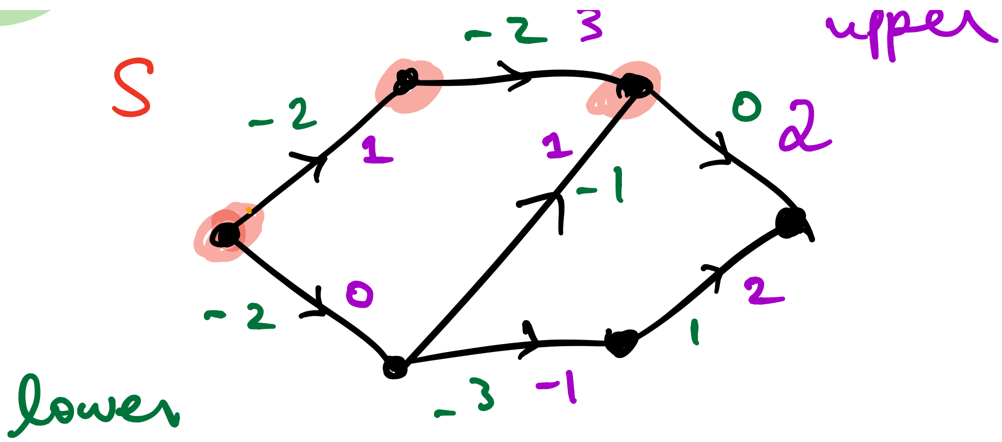
  **capacity of cut $S$:**

$$c(S) = \boxed{\sum_{e\in\delta^+(S)} u(e) - \sum_{e\in\delta^-(S)} l(e)}$$

most that could leaves

min that could enter.

**e.g.**




$c(S) = 0 + 2 - (-1) = 3$

- By design, $\forall$ flow $x$

$$c(S) \ge \underbrace{\sum_{e\in\delta^+(S)} x_e - \sum_{e\in\delta^-(S)} x_e}_{*\ \text{amt. leaving } S.}$$

- If $S$ is $s$-$t$ cut,

$$\boxed{* = |x|.}$$

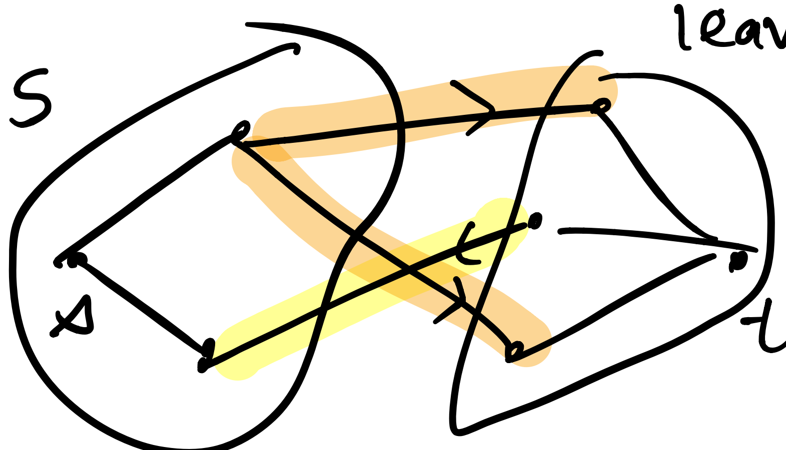
why? similar argument

to show: out of $s$ = into $t$:

$$|x| = \sum_{\delta^+(s)} x_e - \sum_{\delta^+(s)} x_e$$

$$\overset{\text{flow cons.}}{=} \sum_{v \in S} \left( \sum_{\delta^+(v)} x_e - \sum_{\delta^-(v)} x_e \right)$$

cancels except $\delta^+(S)$, $\delta^-(S)$; leaves $*$.




- Thus **weak duality** holds:

$$\boxed{\max_{\text{flow } x} |x| \le \min_{\substack{\text{s.t. cut} \\ S}} C(S).}$$

- if flow instance feasible, strong duality holds!

**Theorem** (max-flow, min-cut)

For any feasible flow network,

$$\max_{\text{flows } x} |x| = \min_{s\text{-}t \text{ cuts } S} C(S)$$

**Proof**: Could use LP duality, TU.

**Today**: "Primal-Dual";

Develop algorithm to find flow, show at termination find matching cut.

## Algorithm (Augmenting flows)
Ford-Fulkerson

- Assume begin w/ feasible flow x.
  e.g. if $l=0$ can take $x=0$

## Repeat until termination:

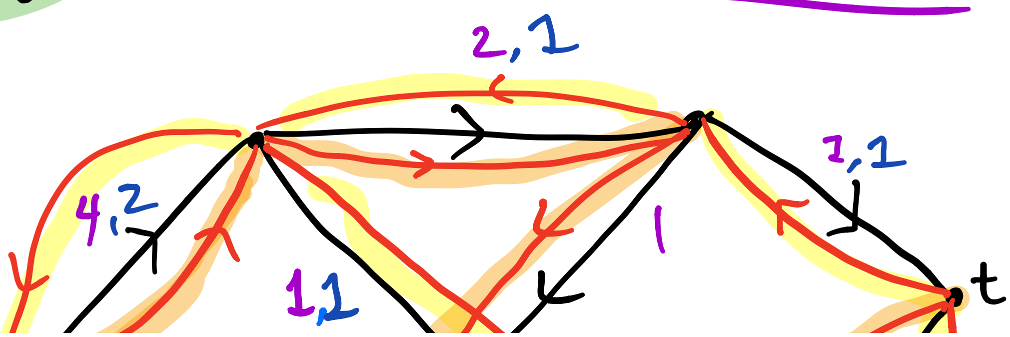
- Define residual graph $G_x = (V, E_x)$ unweighted

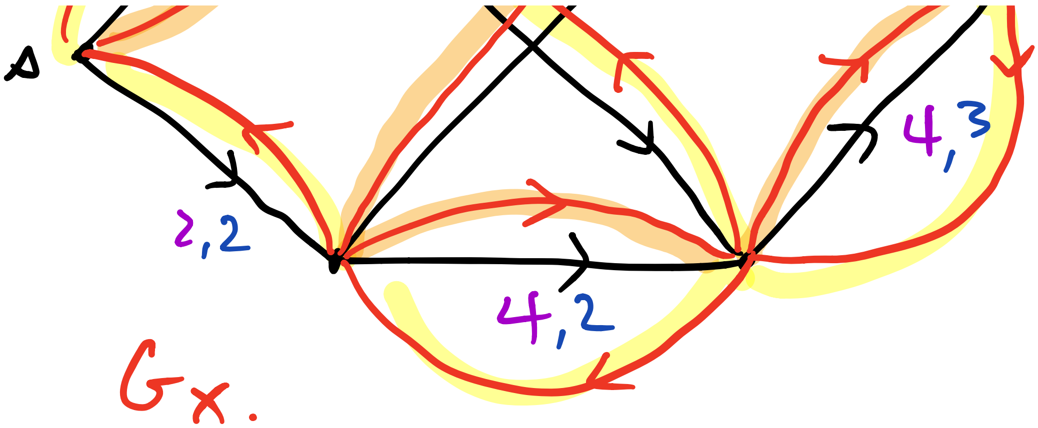
(a) $(i,j) \in E_x$ if $(i,j) \in E$, $x_e < u(e)$
"could increase"

(b) $(i,j) \in E_x$ if $(j,i) \in E$, $x_e > l(e)$.
"could decrease"

if both (a), (b), add both!

e.g. $l=0$, $u$, $x$






(a) "forward" arcs,
(b) "backward" arcs

## Case (i):

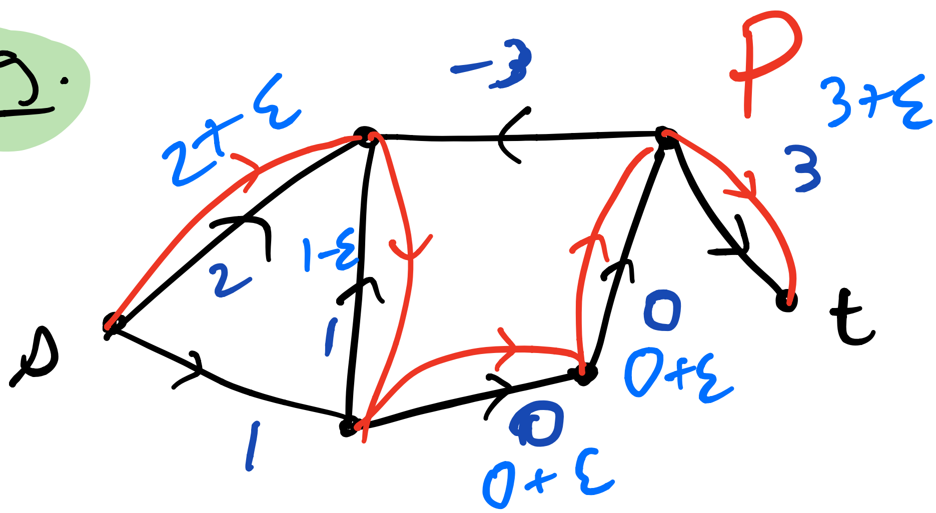
If $\exists$ directed $s$-$t$ path $P$ in $G_x$:

- "augment $x$ along $P$" by $\epsilon$:

$$x'_e = \begin{cases} x_e + \epsilon & \text{if } e \text{ forward in } P \\ x_e - \epsilon & \text{if } e \text{ backward in } P \\ x_e & \text{if } e \notin P. \end{cases}$$

- Observe $x'$ satisfies flow conservation:

e.g.

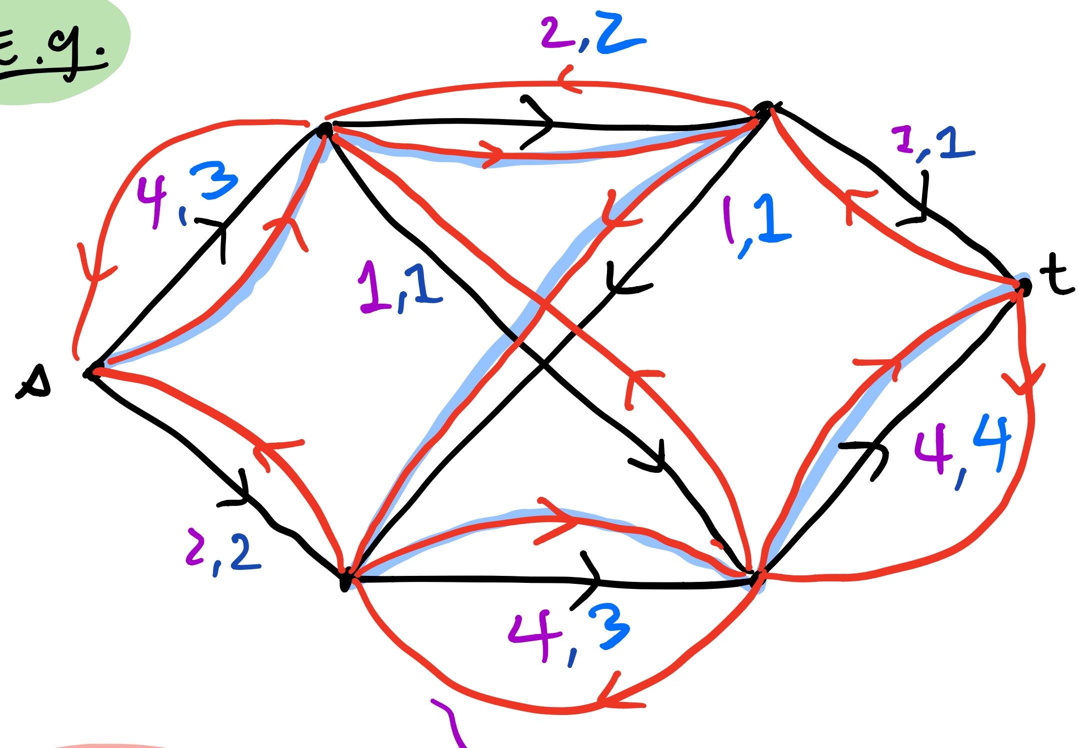



- For $x'$ to be feasible, choose

$$\epsilon = \min \left\{ \min_{\text{forward } e \in P} u(e) - x_e \,,\ \min_{\text{backward } e \in P} x_e - l(e) \right\}$$

- We have $|x'| = |x| + \epsilon$.
- Set $x \leftarrow x'$; start over.

E.g.

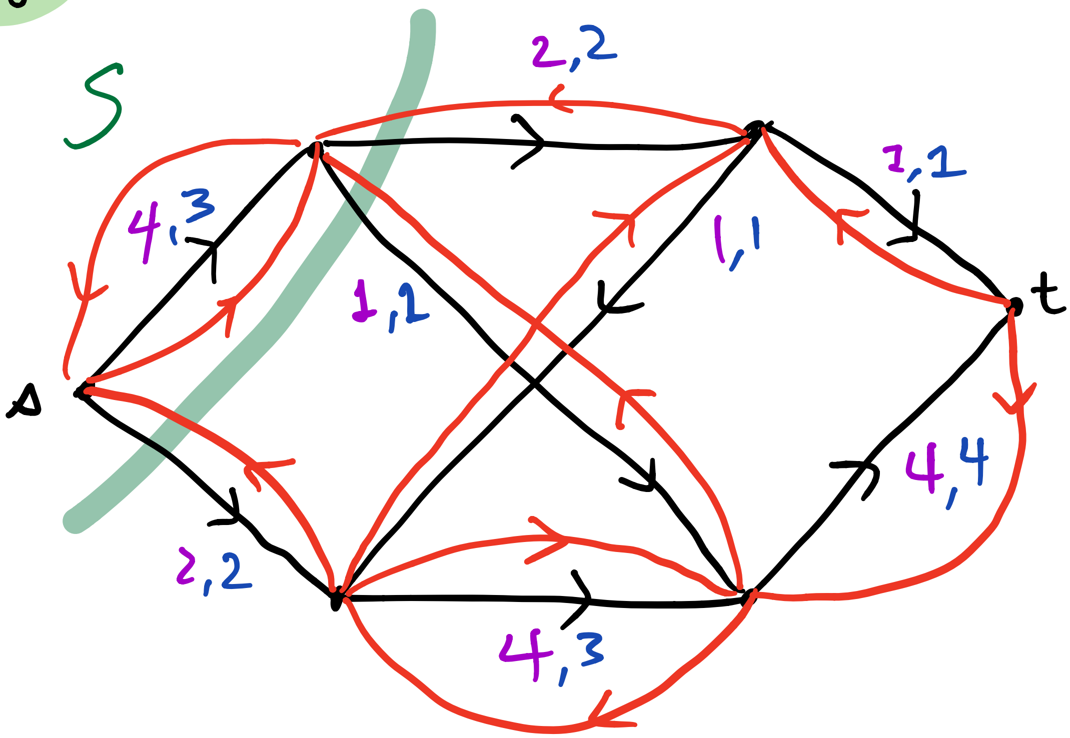



Case (ii): No directed s-t path in $G_x$.

- Let $S = \{$ all vertices reachable from $s$ in $G_x$ $\}$

- By assumption, $S$ is $s$-$t$ cut.

E.g.




- Claim: $C(S) = |X|$.

why? if $e \in \delta^+(S)$, $\boxed{x_e = u(e)}$

similarly, if $e \in \delta^-(S)$, $\boxed{x_e = \ell(e)}$

Means

$$c(S) = \sum_{e\in\delta^+(S)} u(e) - \sum_{e\in\delta^-(S)} \ell(e)$$

$$= \sum_{e\in\delta^-(S)} x_e - \sum_{e\in\delta^+(S)} x_e = |x|$$

- Terminate, output $x$, $S$.

---

**Does this prove theorem?**

---

Not quite!

What if algorithm never terminates?

- if rational, ok b/c can assume integral & always increase flow by $\geq 1$.

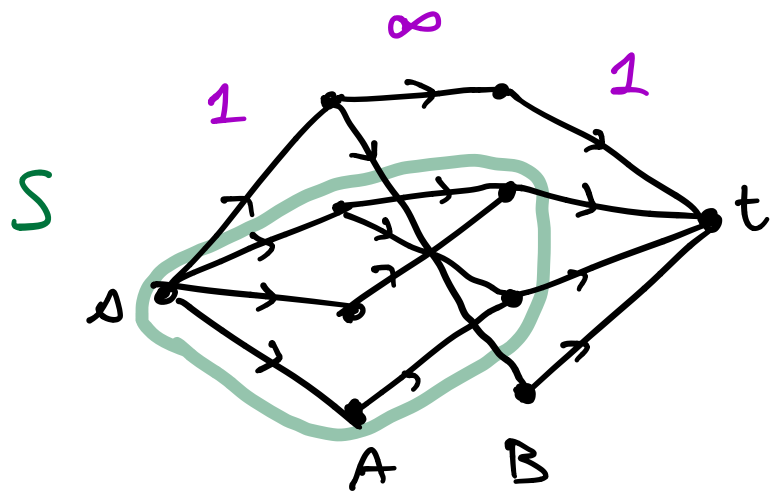
- if capacities irrational, can run forever (even if $|V| = 6$ )
- However, we can fix this: Because a max flow $x$ exists (maxflow is an LP ) just start alg. with max flow $x$, must terminate immediately. (else would increase value.). $\square$

---

# Applications of Max-Flow Min-Cut:

## 1) König's Theorem:

- Recall flow network capturing bipartite matching.




- A min cut cannot contain any of the ∞ edges; thus is in original graph.

$$\boxed{C = (A \setminus S) \cup (B \cap S)}$$

is vertex cover;

$$\boxed{c(S) = |C|} \; !$$

- Thus max matching = min vertex cover; another way to show König's!

## 2) Edge-disjoint paths

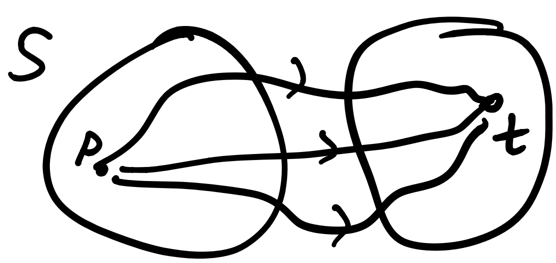
## Menger's Theorem:

In directed graph $G=(V,E)$, $s,t \in V$, $\exists\, k$ edge-disjoint $s$-$t$ paths

$$\Longleftrightarrow$$

$\forall\ S \subseteq V \setminus \{t\}$ with $s \in S$,

$$\boxed{|\delta^+(S)| \geq k}.$$



Next time:

efficiently finding flows.