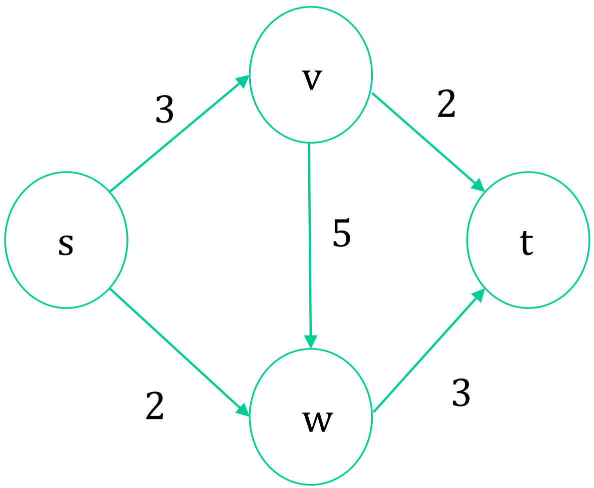
LEC010 Maximum Flow

VG441 SS2020

Cong Shi
Industrial & Operations Engineering
University of Michigan

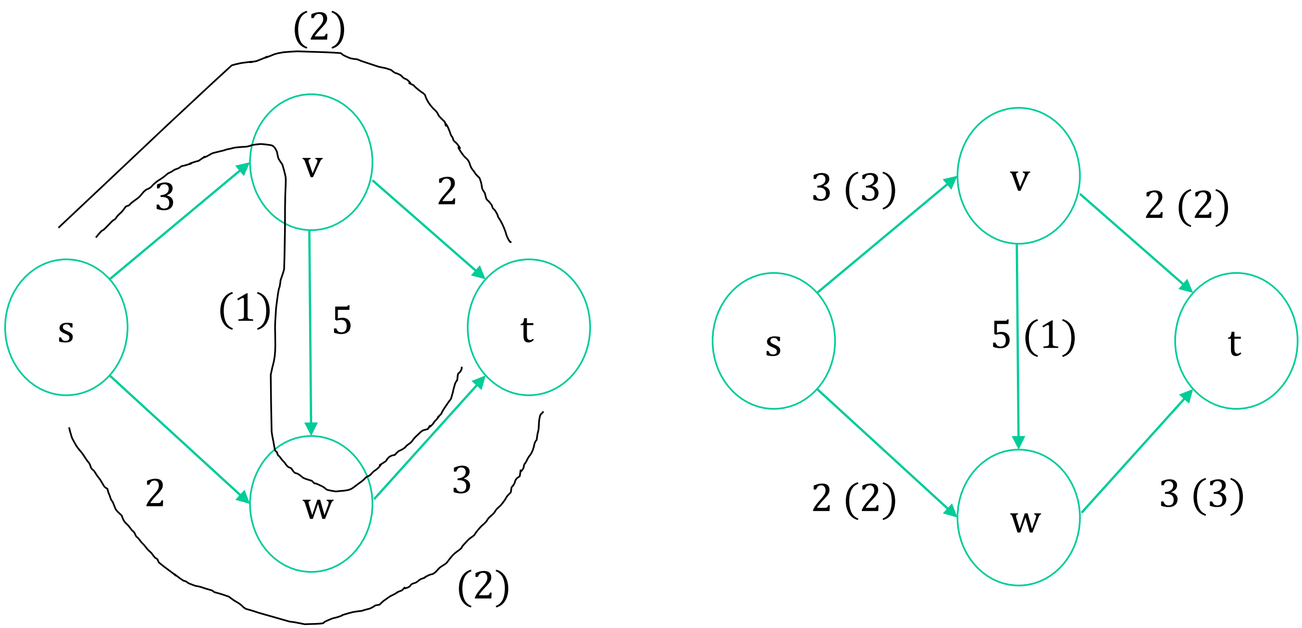
Max Flow



The number on each edge is capacity

Qn: Push as much flow as possible from *s* to *t*

Max Flow



The number on each edge is capacity

Qn: Push as much flow as possible from *s* to *t*

Max Flow

- Input
  - a directed graph G, with vertices V and directed edges E
  - a source vertex $s \in V$ (no edges into s)
  - a sink vertex $t \in V$ (no edges out of t)
  - a nonnegative and integral capacity u_e for each edge $e \in E$
- Feasible solutions – flows
  - Nonnegativity constraints: $f_e \geq 0$ for every edge $e \in E$
  - Capacity constraints: $f_e \leq u_e$ for every edge $e \in E$
  - Conservation constraints: for every vertex v other than s and t

    amount of flow entering v = amount of flow exiting v
- Goal: maximize flow value = flow going out of S

Max Flow

- Attempt #1:

A Naive Greedy Algorithm

initialize $f_e = 0$ for all $e \in E$

repeat

search for an $s - t$ path P such that $f_e < u_e$ for every $e \in P$

// takes $O(|E|)$ time using BFS or DFS

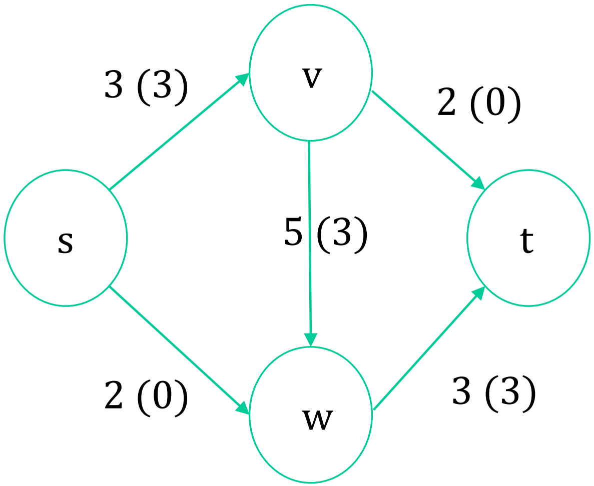
if no such path **then**

halt with current flow $\{f_e\}_{e \in E}$

else

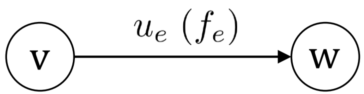
let $\Delta = \underbrace{\min_{e \in P} \overbrace{(u_e - f_e)}^{\text{room on } e}}_{\text{room on } P}$

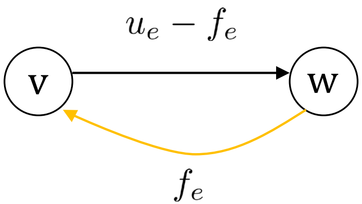
for all edges e of P do increase f_e by Δ



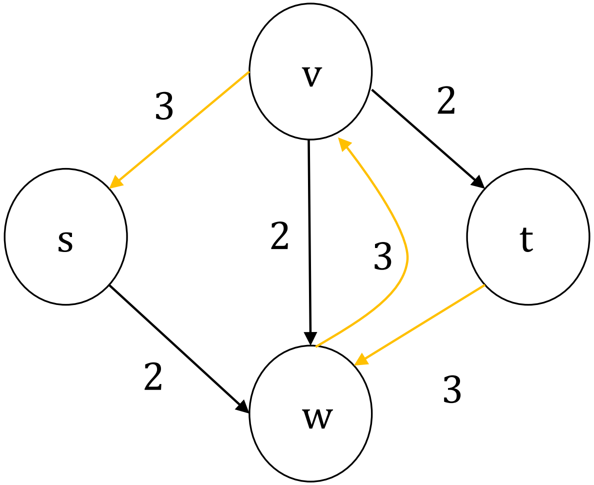
Max Flow

- Attempt #2: Allow "undo" operations





- Residual network



Max Flow

Ford-Fulkerson Algorithm

initialize $f_e = 0$ for all $e \in E$

repeat

search for an $s - t$ path P in the current residual graph G_f such that every edge of P has positive residual capacity

// takes $O(|E|)$ time using BFS or DFS

if no such path **then**

halt with current flow $\{f_e\}_{e \in E}$

else

let $\Delta = \min_{e \in P}$ (e' s residual capacity in G_f)

// augment the flow f using the path P

for all edges e of G whose corresponding forward edge is in P **do**

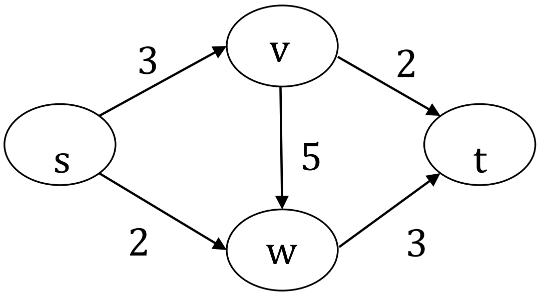
increase f_e by Δ

for all edges e of G whose corresponding reverse edge is in P **do**

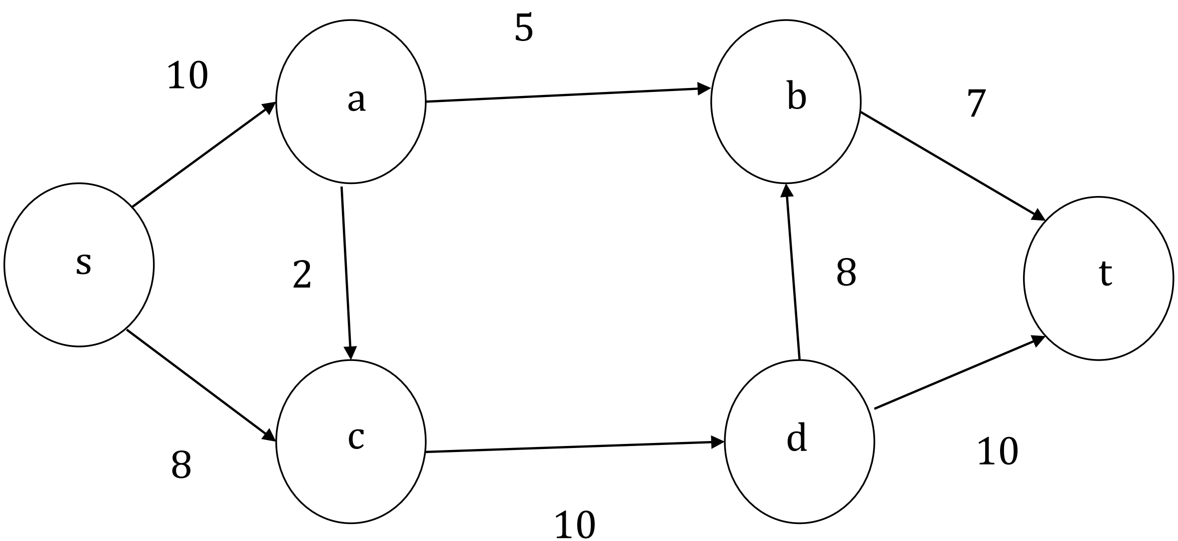
decrease f_e by Δ

Max Flow

- Run Ford-Fulkerson on the simple example

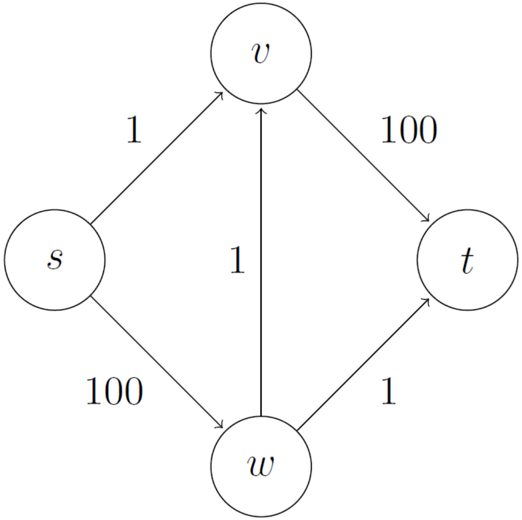


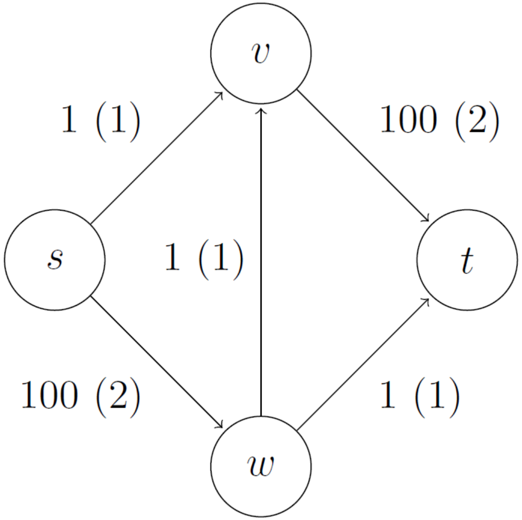
Exercise



15

How do we know we are done?





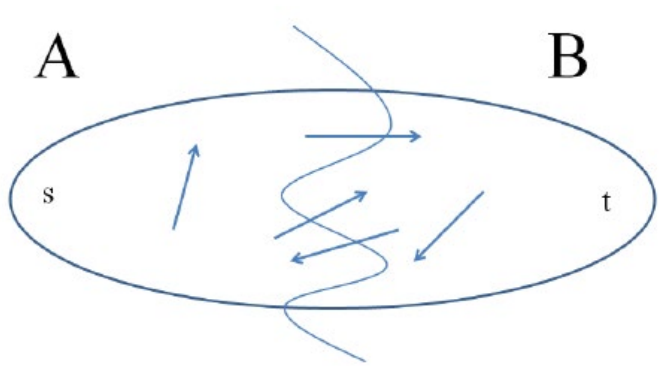
Two-Step Paradigm:

- Identify "optimality condition"
- Design an algorithm that terminates w/ the optimality condition satisfied

(s,t) cuts

- "Dual" flows

Definition An (s,t) -cut of a graph $G = (V, E)$ is a partition of V into sets A, B with $s \in A$ and $t \in B$



The capacity of an (s,t) -cut (A, B) is defined as

$$\sum_{e \in \delta^+(A)} u_e$$

Equivalence of (1) (2) (3)

- Max-Flow-Min-Cut Theorem

(1) f is a maximum flow of G

(2) there is an (s,t)-cut (A,B) s.t. the value of f equals the capacity of (A,B)

(3) there is no s–t path (with positive residual capacity) in the residual G_f

(2) => (1)

(2) there is an (s,t)-cut (A,B) s.t. the value of f equals the capacity of (A,B)

implies

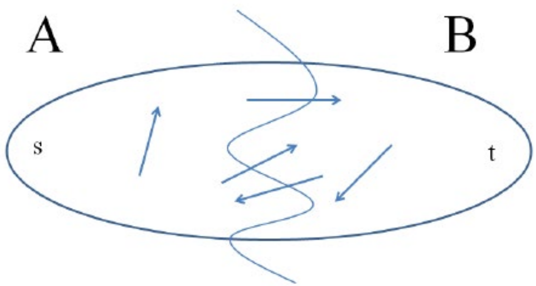
(1) f is a maximum flow of G

Claim:

for every flow f and every (s,t)-cut (A,B) value of $f \leq$ capacity of (A,B)

$$\text{value of } f = \underbrace{\sum_{e\in\delta^+(s)} f_e}_{\text{flow out of } s} = \sum_{e\in\delta^+(s)} f_e - \underbrace{\sum_{e\in\delta^-(s)} f_e}_{\text{vacuous sum}} \quad \text{and} \quad \underbrace{\sum_{e\in\delta^+(v)} f_e}_{\text{flow out of } v} - \underbrace{\sum_{e\in\delta^-(v)} f_e}_{\text{flow into of } v} = 0$$

$$\begin{aligned}
\text{value of } f &= \sum_{v\in A}\left(\sum_{e\in\delta^+(v)} f_e - \sum_{e\in\delta^-(v)} f_e\right) \\
&= \sum_{e\in\delta^+(A)} \underbrace{f_e}_{\leq u_e} - \sum_{e\in\delta^-(A)} \underbrace{f_e}_{\geq 0} \\
&\leq \sum_{e\in\delta^+(A)} u_e \\
&= \text{capacity of } (A,B)
\end{aligned}$$



(1)=>(3)

(1) f is a maximum flow of G

implies

(3) there is no s–t path (with positive residual capacity) in the residual G_f

用 contrapositive

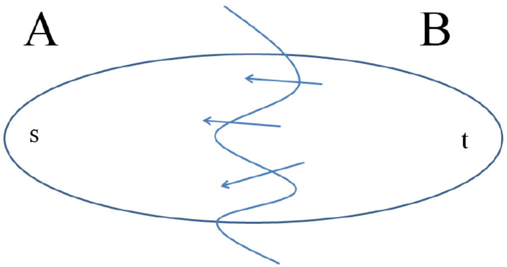
(3)=>(2)

(3) there is no s–t path (with positive residual capacity) in the residual G_f

implies

(2) there is an (s, t) -cut (A, B) s.t. the value of f equals the capacity of (A, B)

$$A = \{v \in V : \text{ there is an } s \rightsquigarrow v \text{ path in } G_f\}$$



Run BFS from s until stuck

(1) $\forall e \in \delta^+(A)$, $U_e - f_e = 0$ (no forward edges)
(2) $\forall e \in \delta^-(A)$, $f_e = 0$ (no "flow-inducded" backward edges)

$$\text{value of } f = \sum_{e \in \delta^+(A)} f_e - \sum_{e \in \delta^-(A)} f_e = \sum_{e \in \delta^+(A)} u_e = \text{cap}(A, B)$$