Math 409: Discrete Optimization
Today: Min cut / Max Flow Algorithms

Network Flows

$G=(V,E)$ directed graph, $c: E \to \mathbb{R}_{\geq 0}$ edge capacities

$s,t \in V$ $\quad s=$ "source" $\quad t=$ "sink"

FORD FULKERSON ALGORITHM

Input: digraph $G=(V,E)$, $c: E \to \mathbb{Z}_{\geq 0}$, $s,t \in V$

Output: s-t flow f and s-t cut $U \subseteq V$ with value$(f) = c(\delta^{out}(U))$

(1) Set $f(e)=0 \quad \forall e \in E$

(2) Repeat:

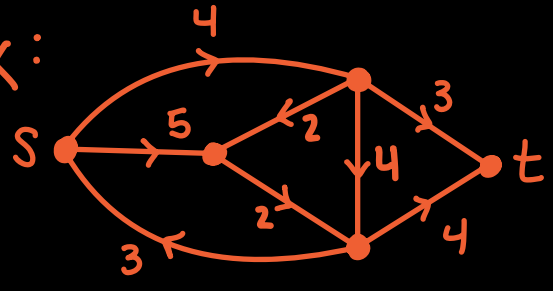
(3) Find an s-t path P in G_f

If none, output f and $U = \{v \in V : \exists \text{ s-v path in } G_f\}$

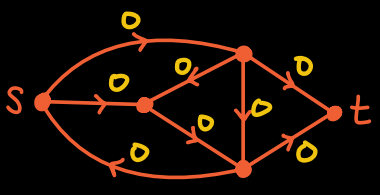
(4) Compute $\gamma = \min\{c_f(e) : e \in P\}$

(5) Augment f along P by γ

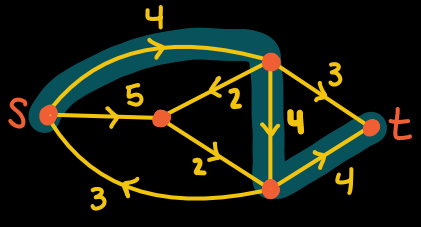
Ex:



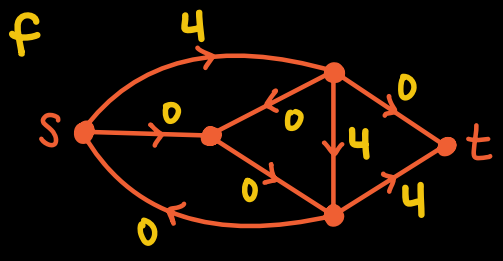
Initialize

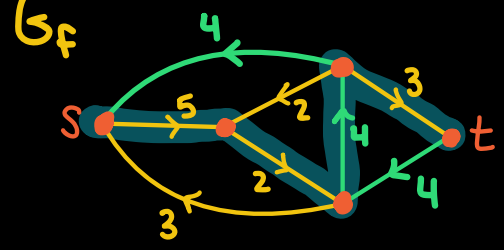


residual graph G_f:

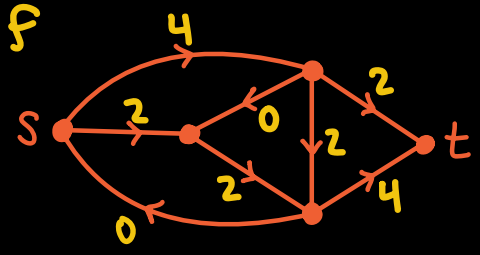


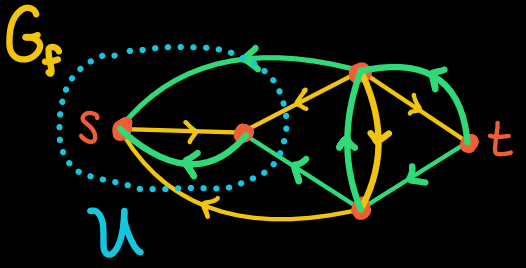
$\gamma = 4$



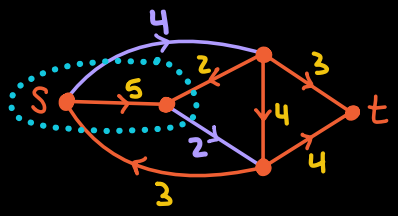


$\gamma = 2$





no s-t path in G_f



$\text{value}(f) = 6 = c(\delta^{out}(U))$

Claim: This alg. terminates after at most $n \cdot C$ repetitions of (2) where $n = |V|$ and $C = \max\{c(e) : e \in E\}$

(Proof) s has $\leq n$ outgoing edges each w/ $c(e) \leq C$

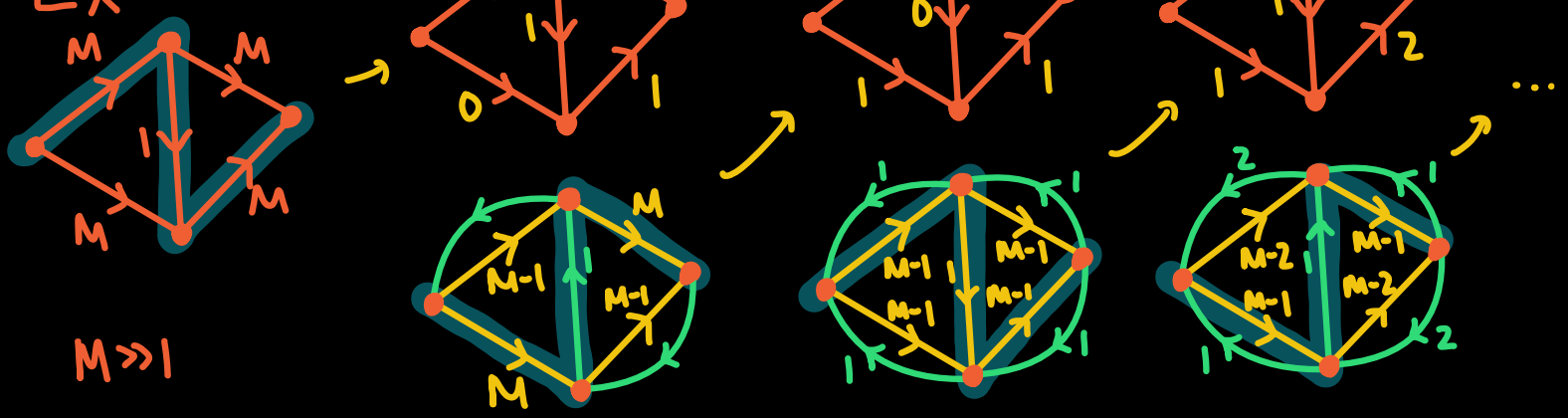
$\Rightarrow$ value of max flow $\leq c(\delta^{out}(s)) = n \cdot C$.

At each repetition, either stop or add ≥ 1 to value(f). □

Perform each (2) in $O(m)$ where $m = |E|$ $\Rightarrow$ total run time $O(n \cdot m \cdot C)$

With arbitrary choice of paths, it can take this long!

Ex:



$M \gg 1$

We can improve run time with a systematic choice of path in G_f.

EDMONDS-KARP ALGORITHM

Input: digraph $G=(V,E)$, $c: E \to \mathbb{R}_{\geq 0}$, $s,t \in V$

Output: s-t flow f and s-t cut $U \subseteq V$ with value$(f) = c(\delta^{out}(U))$

(1) Set $f(e)=0$ $\forall e \in E$

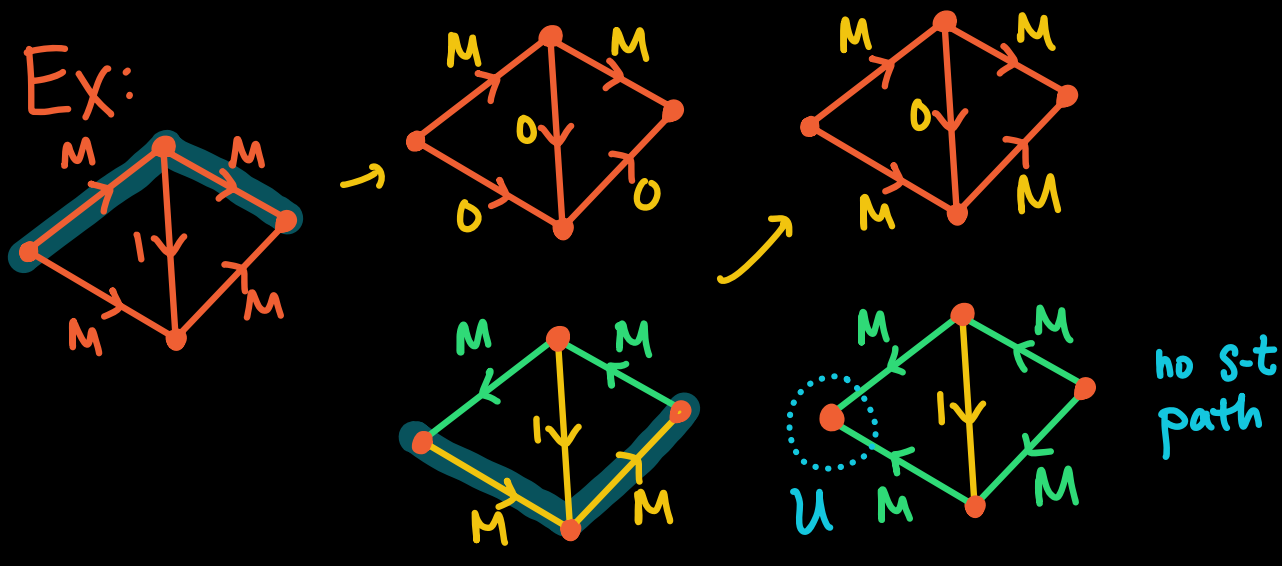
(2) Repeat:

(3) Find the shortest (fewest edges) s-t path P in G_f

If none, output f and $U = \{v \in V : \exists \text{ s-v path in } G_f\}$

(4) Compute $\gamma = \min\{c_f(e) : e \in P\}$

(5) Augment f along P by γ



Claim: Edmonds-Karp terminates after $\leq 2mn$ iterations of step (2) $\Rightarrow$ run time $O(m^2n)$

Main idea: every edge $e \in E$ can appear as the bottleneck ($e \in P$, $c_f(e) = \gamma$) at most n times.

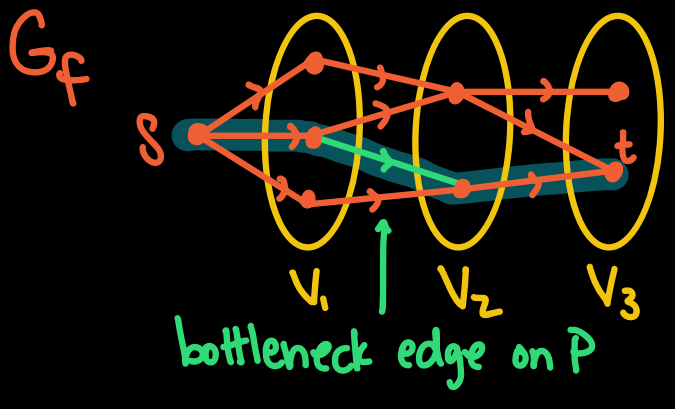
Step 1: Augmenting paths can only increase the distance between s and v in G_f

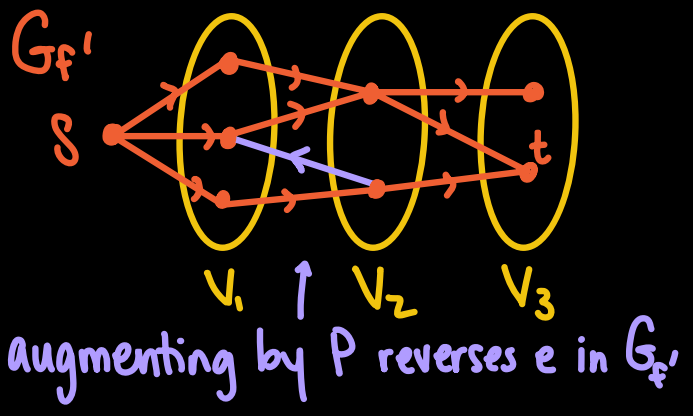
Step 2: If (v,w) was a bottleneck twice with flows f and later f', then s-v dist. in $G_f <$ s-v dist in $G_{f'}$

Since s-v dist. belongs to $\{1, \ldots, n-1, \infty\}$, it can strictly increase at most n times

(Idea of Proof) $V_j = \{v \in V$ with s-v distance j in $G_f\}$

Chosen P path uses edges $V_j \to V_{j+1}$





See notes for more details.

There are many other versions!
Orlin (2012) gave an alg. for Max Flow with running time $O(n \cdot m)$.