Algorithms

Flow Algorithms

Recap

- Hello again!
- Reading & HW 6 due Wednesday
- Finish (?) flows, then back to Chapter 8 by Friday or Monday
- Note for HW:

  Flows take $O(VE)$ time!

  not $n+m$

Thm: (Ford-Fulkerson '54, Elias-Feinstein-Shannon '56)
The max flow value $\leq$ min cut value

First:

Lemma 10.1. *Let f be **any** feasible (s,t)-flow, and let (S,T) be **any** (s,t)-cut. The value of f is at most the capacity of (S,T). Moreover, $|f| = \|S,T\|$ if and only if f saturates every edge from S to T and avoids every edge from T to S.*

Proof: Choose your favorite flow f and your favorite cut (S,T), and then follow the bouncing inequalities:

$$
\begin{aligned}
|f| &= \partial f(s) && \text{[by definition]}\\
&= \sum_{v\in S} \partial f(v) && \text{[conservation constraint]}\\
&= \sum_{v\in S}\sum_{w} f(v\to w) - \sum_{v\in S}\sum_{u} f(u\to v) && \text{[math, definition of } \partial\text{]}\\
&= \sum_{v\in S}\sum_{w\notin S} f(v\to w) - \sum_{v\in S}\sum_{u\notin S} f(u\to v) && \text{[removing edges from } S \text{ to } S\text{]}\\
&= \sum_{v\in S}\sum_{w\in T} f(v\to w) - \sum_{v\in S}\sum_{u\in T} f(u\to v) && \text{[definition of cut]}\\
&\leq \sum_{v\in S}\sum_{w\in T} f(v\to w) && \text{[because } f(u\to v)\geq 0\text{]}\\
&\leq \sum_{v\in S}\sum_{w\in T} c(v\to w) && \text{[because } f(v\to w)\leq c(v\to w)\text{]}\\
&= \|S,T\| && \text{[by definition]}
\end{aligned}
$$

In the second step, we are just adding zeros, because $\partial f(v) = 0$ for every vertex $v \in S \setminus \{s\}$. In the fourth step, we are removing flow values $f(x\to y)$ where

One way is easy!

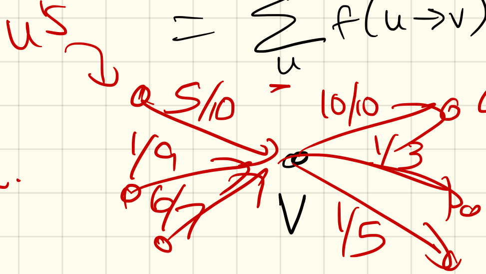
Any flow $\leq$ any cut.

Pf: Pick a flow f:

Let $\delta f(v)$ = flow out of vertex v

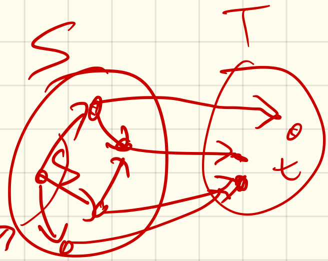
$$= \sum_u f(u \to v) - \sum_w f(v \to w)$$

for any $v \neq s,t$. $= 0$



$$|f| = \sum_{v \in S} \partial(v)$$

Why? cut



$|f|$

Wants $\partial(S)$, $s \in S$

Next:

$$|f| = \sum_{v\in S}\Big[\underbrace{\sum_u f(u\to v) - \sum_w f(v\to w)}_{\delta f(v)}\Big]$$

$$= \sum_{v\in S}\sum_u f(u\to v) - \sum_{v\in S}\sum_w f(v\to w)$$

Then, can remove any $S\to S$ edges, so only $S\to T$ edges left:

$$= \sum_{v\in S}\sum_{\substack{\cancel{w\in S}\\ w\in T}} f(v\to w) - \sum_{v\in S}\sum_{\substack{\cancel{w\in S}\\ w\in T}} f(w\to v)$$

$$\le \sum_{v\in S}\sum_{w\in T} \underbrace{f(v\to w)}_{\le c(v\to w)}$$

$$\le \sum_{v\in S}\sum_{w\in T} c(v\to w)$$

$$= \|S,T\|$$

Next: Show that can get them equal.

How?

Well, take some flow, f:

Either:

① If f is maximum, in which case, find a cut of equal value.

② It isn't, + then find a bigger flow.

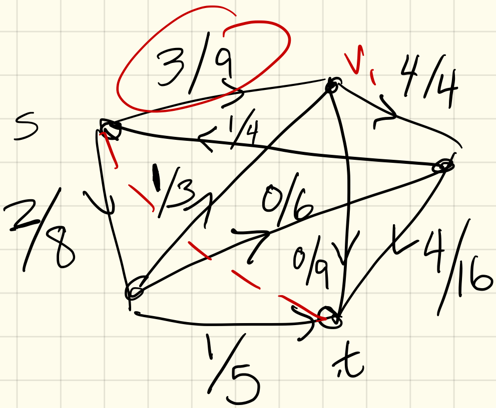
Key tool in proof:

Residual capacity: Given G & F:

$$C_f(u \to v) := \begin{cases} c(u \to v) - f(u \to v) & \text{if } u \to v \in E \\ f(u \to v) & \text{if } v \to u \in E \\ 0 & \text{otherwise} \end{cases}$$

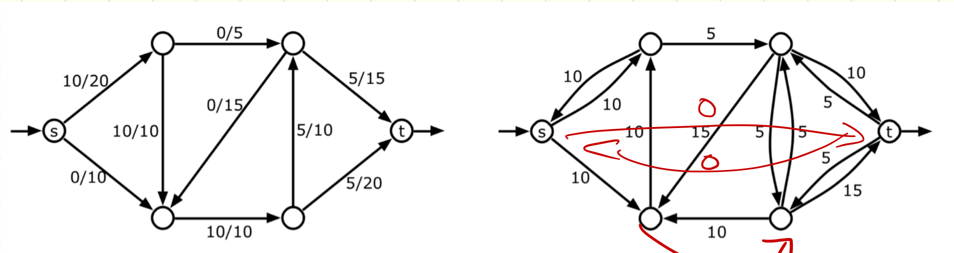
Ex: $S \xrightarrow{3/9} V_1$

$C_f(S \to V_1) = 9 - 3 = 6$



$C_f(V_1 \to S) = 3$

We can visualize this as a new graph, G_f:

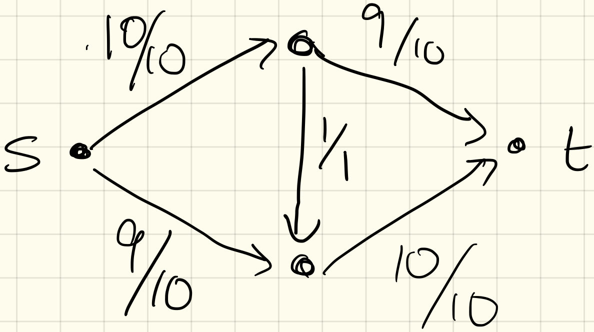


A flow f in a weighted graph G and the corresponding residual graph G_f.

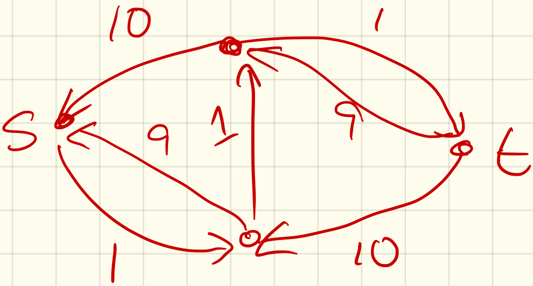
Intution:

A path in $\underline{G_f}$ if a way to send more flow!

Another example:



G_f:



Aside:

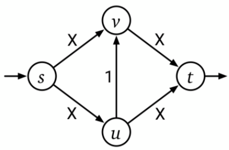
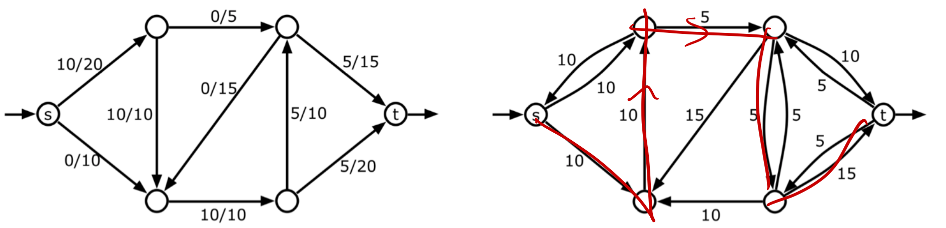


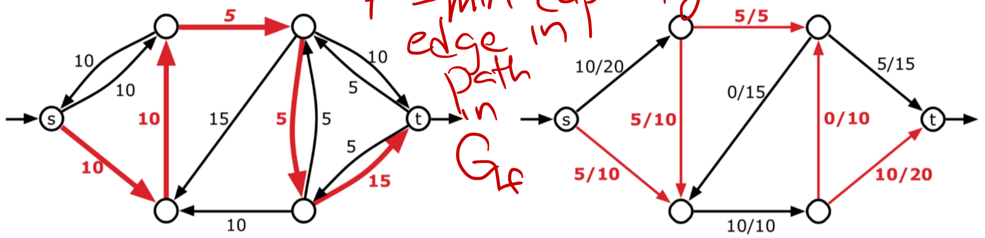
Figure 10.7. Edmonds and Karp's bad example for the Ford-Fulkerson algorithm.

Augmenting a path:

Suppose there is a path in G_f from s to t:



A flow f in a weighted graph G and the corresponding residual graph G_f.



$F = $ min capacity edge in path in G_f

An augmenting path in G_f with value $F = 5$ and the augmented flow f'.

$$f' = \begin{cases} \text{if } u \to v \text{ not on aug. path, } f'(u\to v) \text{ is } f(u \to v) \\ \text{if } u \to v \text{ is in } G, \\ \quad f'(u\to v) = f(u \to v) + F \\ \text{otherwise } (u \to v \text{ is not in } G) \\ \quad f'(v \to u) = f(v \to u) - F \end{cases}$$

Claim: f' is also a feasible flow!

Why?

- For any $u \to v$ not on augmenting path,
  same flow value

- For $u \to v$ on augmenting path,

$$f'(u \to v) = f(u \to v) + F \geq f(u \to v) \geq 0$$

Still feasible!

if forward, $F \leq c(e) - f(e)$

so $f(e) + F \leq c(e)$

still valid

<u>So:</u> f wasn't a max flow, since f' is larger.

On other hand:

if G_f has no $s \to t$ path, find $|S|$ = set of vertices that s can reach.

<u>Claim:</u> $(S, V-S)$ is a cut.

Why?

If not, you'd have a path.

Even better, each edge out of S is at capacity. (so $f(\overset{uv}{e}) = c(e)$) otherwise, v would be in S too!

Immediate Algorithm:

Start with $f = 0$.

Build $G_f = G$

WFS(G_f, s)] $O(V+E)$

While t + s in same component:

find $s \to t$ path via WFS

Augement along the path to get f'] $O(E)$

$f \leftarrow f'$

Build G_f: $O(E)$

WFS(G_f, s)] $O(V+E)$

$|f^*|$

Runtime: ~~If~~ I push ≥ 1 unit of flow each iteration

$O((V+E)|f^*|)$

Why all this integrality stuff?
We are assuming each path pushes at least 1 more unit of flow!

Can it be that bad?

Yes:

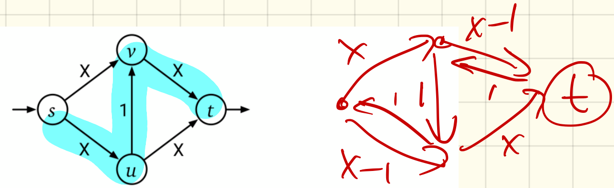


Figure 10.7. Edmonds and Karp's bad example for the Ford-Fulkerson algorithm.

How "big" is f?
(Remember, not part of input!)
Input: G, + capacities on edges
↳ here, size is $O(\log X)$
FF take $O(X)$ time

What if it's $\underline{\text{not}}$ integers?

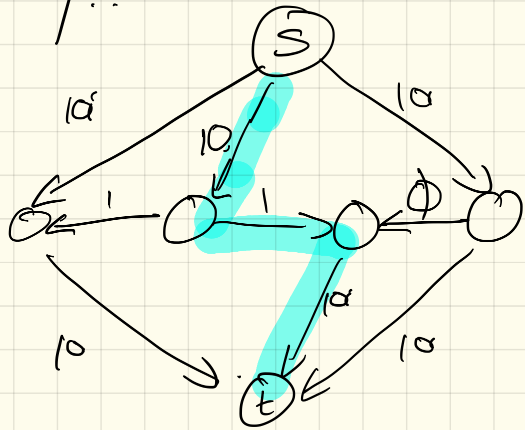
Messy!!

$$\phi = \frac{1+\sqrt{5}}{2}$$

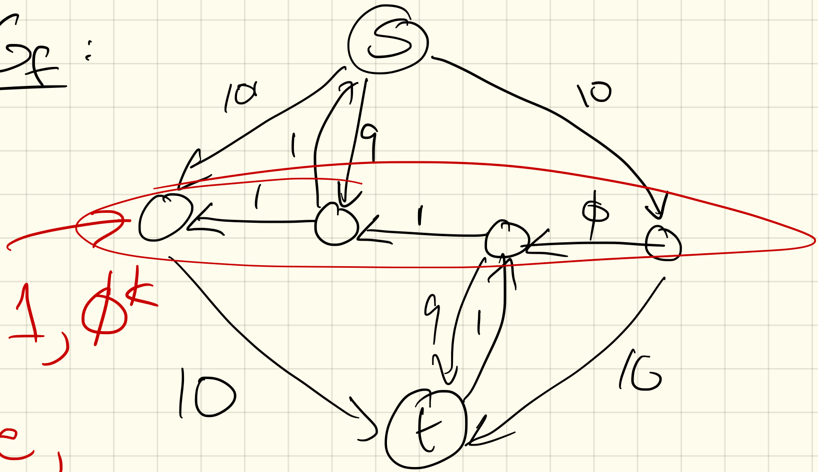
$\downarrow$

Why?

$$1-\phi = \phi^2$$



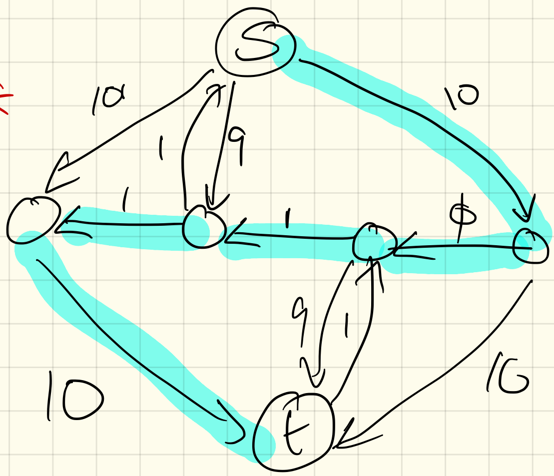
$\underline{G_f}$:



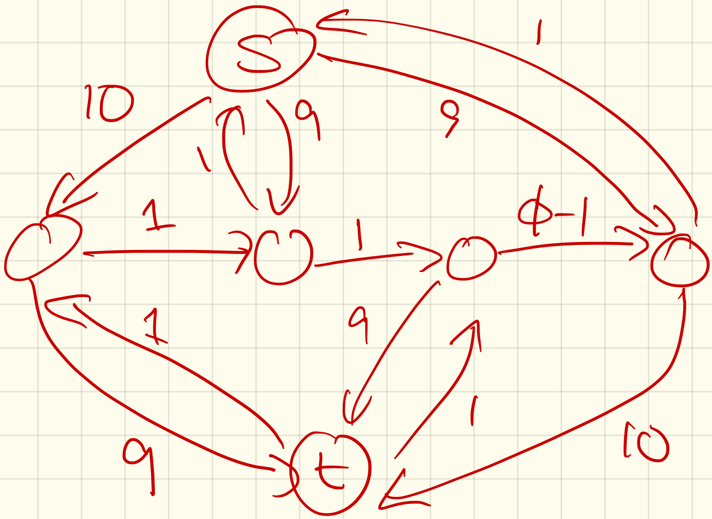
$\phi^{k+1}, 1, \phi^k$

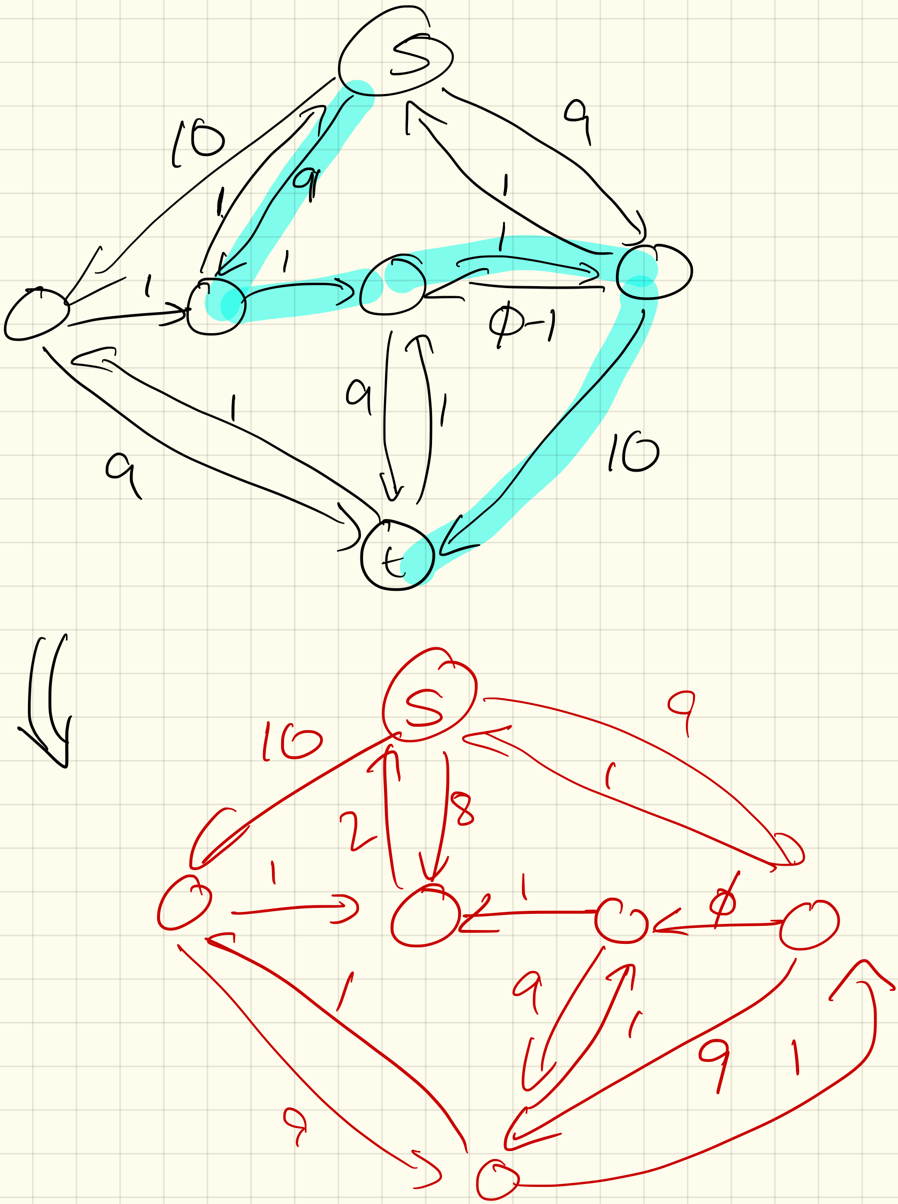
here, $k=1$

old G_f



new G_f




s
10
9
1
ϕ-1
t
2
8
ϕ

Continue to push:

Ends with:

ϕ, $\cancel{\phi}$ 1, and $1-\phi = \phi^2$

$= \phi^{k-1}$, 1, $+ \phi^{k}$

Repeat:

- ϕ^2, 1, $+\phi^3$

then

- ϕ^3, 1, ϕ^4

$$\vdots$$

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots < 1$$

Next Section:

F-F just push on any path.

Could we go faster by choosing some "good" path?

- Edmonds-Karp:

- Dinitz: