CSCE 689: Advanced Graph Algorithms

Lecture 2: Maximum s-t Flow

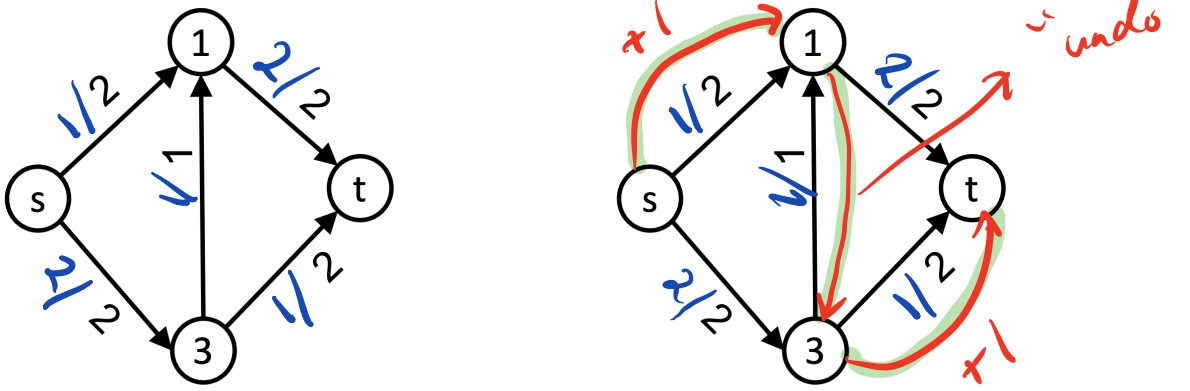
Date: August 30 *Lecturer: Nate Veldt*

Course Logistics

- Homework 1 has been posted, due on Tuesday, Sept 13 at 11:59pm
- Intro video assignment posted
- Can now access recorded lectures via Canvas, course materials through veldt.engr.tamu.edu/689-fall22

1 Finding maximum s-t flow

Recap of our first idea. Repeatedly find paths from s to t, and keep adding flow until there are no more s-t paths.



How do we correct this? Let's try to keep track of flow that we could "undo".

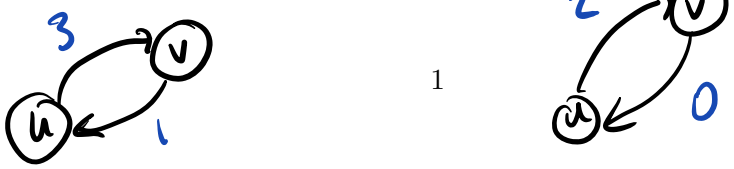
2 The Residual Graph

Given a flow f, for a pair of nodes $(u, v) \in V \times V$, the *residual capacity* for (u, v) is

$$c_f(u, v) = c(u, v) - f(u, v) + f(v, u)$$

Informally, this is the amount of "space" left on the edge $c(u, v)$, plus the amount of flow from v to u that we could "undo".

Typically we will only have one of $\{f(u,v), f(v,u)\}$ be positive



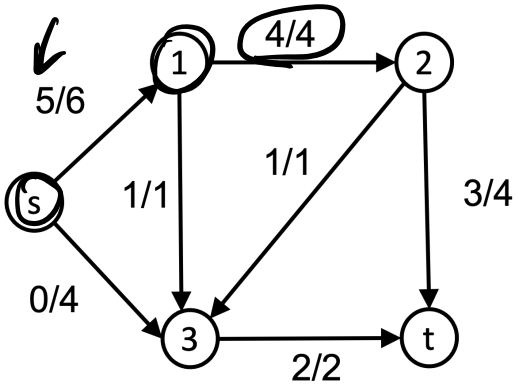
Given a flow f for a graph $G = (V, E, w)$, the *residual graph* $G_f = (V, E_f)$ is the graph where the edge set

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

This graph shows us where we can send more flow to improve on the flow f.

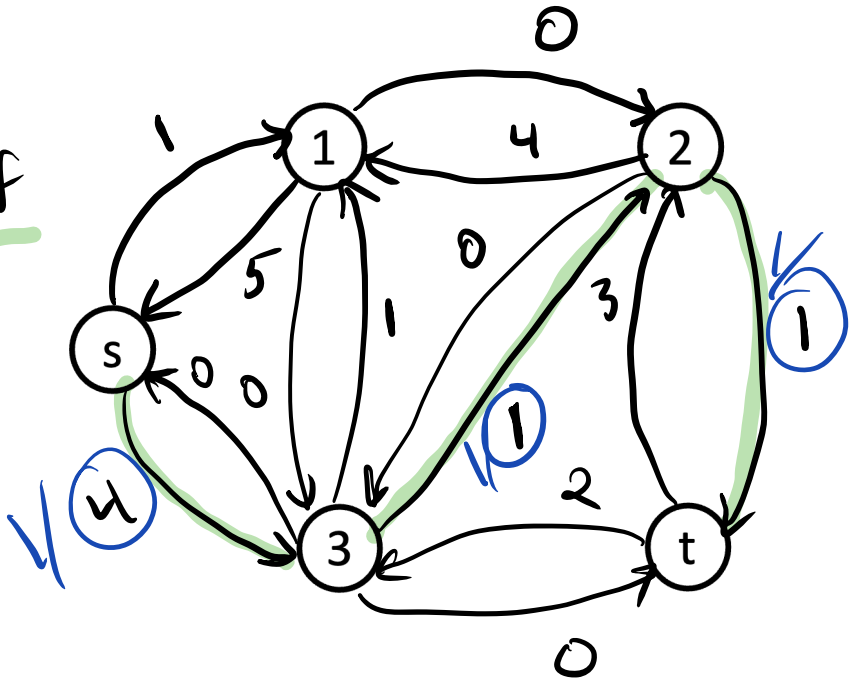
Activity: draw the residual graph for the following flow

G



Key idea: if $(u,v) \in E$ then think about both (u,v) and (v,u) being present in G_f (possibly with capacity 0)

G_f



f was not a max s-t flow

We found an "augmenting path"

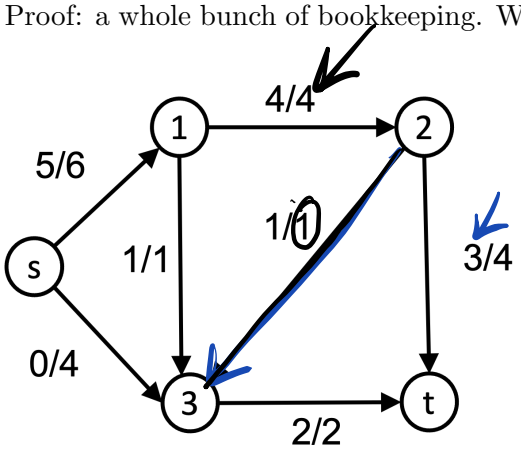
3 Augmenting Flows and Paths

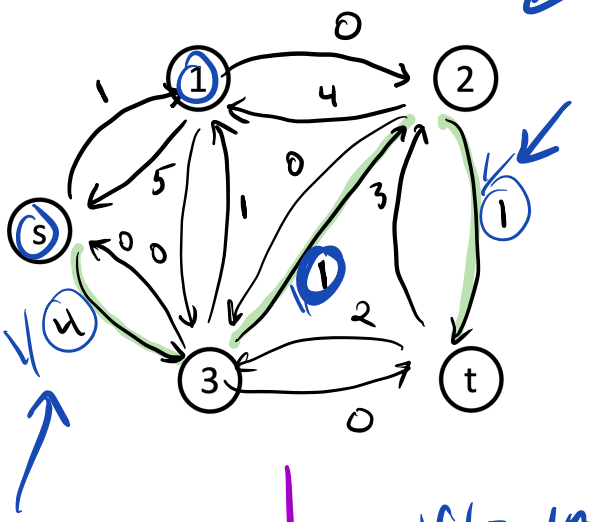
Let f be an s-t flow in $G = (V, E)$ and f' be a flow in the residual graph $G_f = (V, E_f)$. Then we define the *augmentation* of f by f' as:

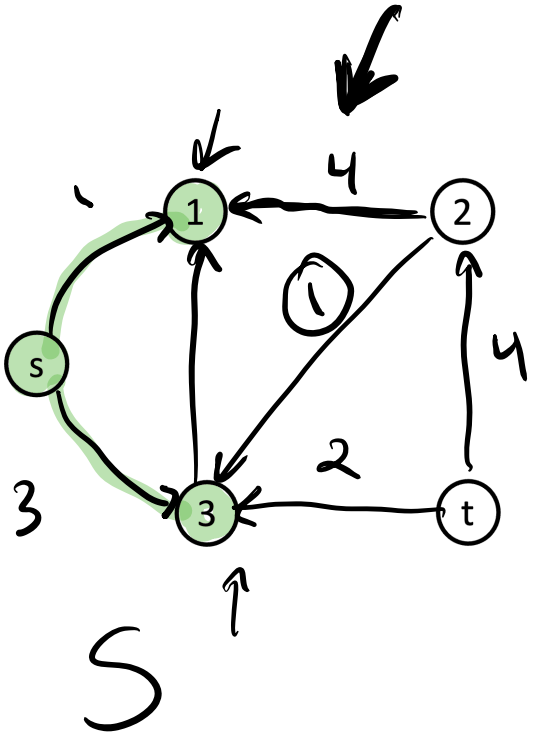
$$f \uparrow f' = f(u, v) + f'(u, v) - f'(v, u) \tag{1}$$

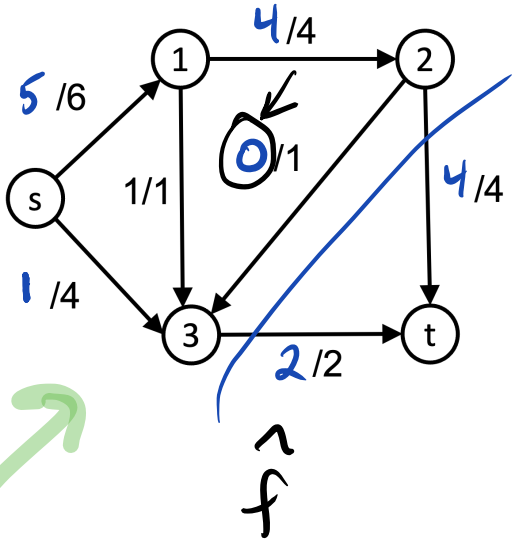
Lemma 1. *The function $f \uparrow f'$ is a valid flow in G, and it has flow value $|f| + |f'|$.*

Proof: a whole bunch of bookkeeping. We will skip this. But we can illustrate it below.









An *augmenting path* p is a simple path (simple = no cycles) from s to t in the residual network G_f.

The *residual capacity* of this path p is the maximum amount we can send on p:

$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is in } p\}$$

Sending $c_f(p)$ flow along every edge in this path gives us a flow f_p in G_f that we can add to f to improve it.

Theorem 2. *(Max-flow Min-cut Theorem) Let f be an s-t flow on some graph $G = (V, E)$. The following three conditions are equivalent:*

1. *f is a maximum s-t flow*
2. *There are no augmenting paths in the residual graph G_f*
3. *$|f| = \textbf{cut}(S)$ for some s-t cut set in G*

Proof: (1) $\Rightarrow$ (2) If G_f had an augmenting path, with flow f', then by Lemma 1, we could find a flow $\hat{f} = f + f'$ with $|\hat{f}| = |f| + |f'| > |f|$ (Contradiction)

(2) $\Rightarrow$ (3)

Let S be the set of nodes reachable from (s) in G_f. In G we know

$$cut(S) = \sum_{\substack{(u,v) \in E \\ u \in S,\, v \in \bar{S}}} c(u,v)$$

Any edge $(u,v) \in \partial S$ must be saturated by f or else (u,v) would be an edge in G_f.

Furthermore, every edge $(v,u) \in E$ with $v \in \bar{S}$ and $u \in S$ must have <u>zero</u> flow on it, or else (u,v) would be an edge in G_f.

So the net flow across cut ∂S is

$$|f| = \sum_{\substack{(u,v)\in E \\ u\in S,\ v\in\bar{S}}} f(u,v) - \sum_{\substack{(v,u)\in E \\ v\in\bar{S},\ u\in S}} f(v,u) = 0$$

$$= \sum_{(u,v)\in \partial S} c(u,v) = \text{cut}(S) \quad \checkmark$$

(2) $\Rightarrow$ (3)

(3) $\Rightarrow$ (1) $\text{cut}(S) \geq |f|$ for any s-t set $S \subseteq U$ and flow, so if $\text{cut}(S) = |f|$, then $|f|$ must be maximized.

4 The Basic Ford-Fulkerson Algorithm

Idea: f is a max-flow if and only if there are no augmenting paths. So let's just keep finding augmenting paths until we're done!

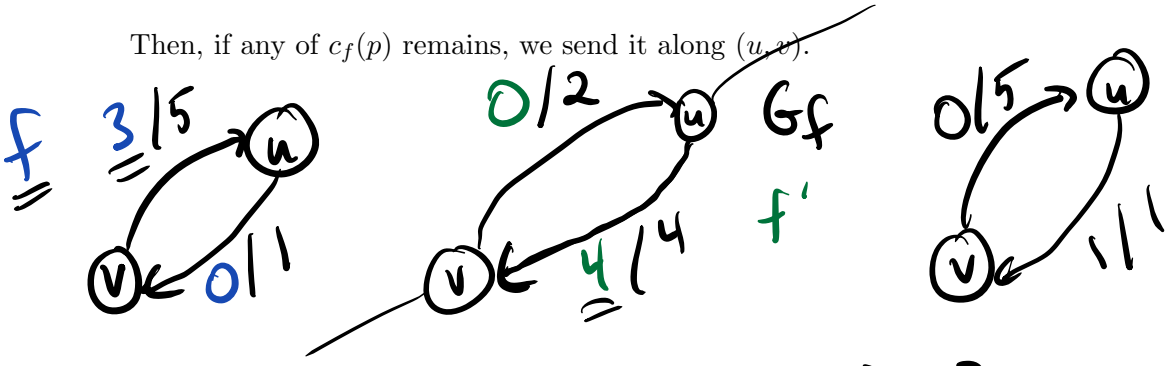
The Ford-Fulkerson algorithm will always maintain the invariant that for any pair (u, v), at most one of $\{f(u, v), f(v, u)\}$ will be greater than zero.

FordFulkersonBasic(G, s, t)

for $(u, v) \in E$ **do**
  $f(u, v) = 0$
while there exists an s-t path p in G_f **do**
  $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
  for $(u, v) \in p$ **do**
    $m = \min\{c_f(p), f(v, u)\}$ // flow to "undo"
    $\ell = c_f(p) - m$ // new flow to send along (u, v)
    $f(v, u) \leftarrow f(v, u) - m$
    $f(u, v) \leftarrow f(u, v) + \ell$

For $(u, v) \in p$, we first use any of the flow $c_f(p)$ to undo flow previously sent on (v, u).

Then, if any of $c_f(p)$ remains, we send it along (u, v).



$c_f(p) = 4 \quad f(v,u) = 3$

$m = \min\{4, 3\} = 3$

$\ell = 4 - m = 1$

$f(v,u) = f(v,u) - m$
$= 3 - 3 = 0$

$f(u,v) = f(u,v) + \ell$
$= 0 + 1$

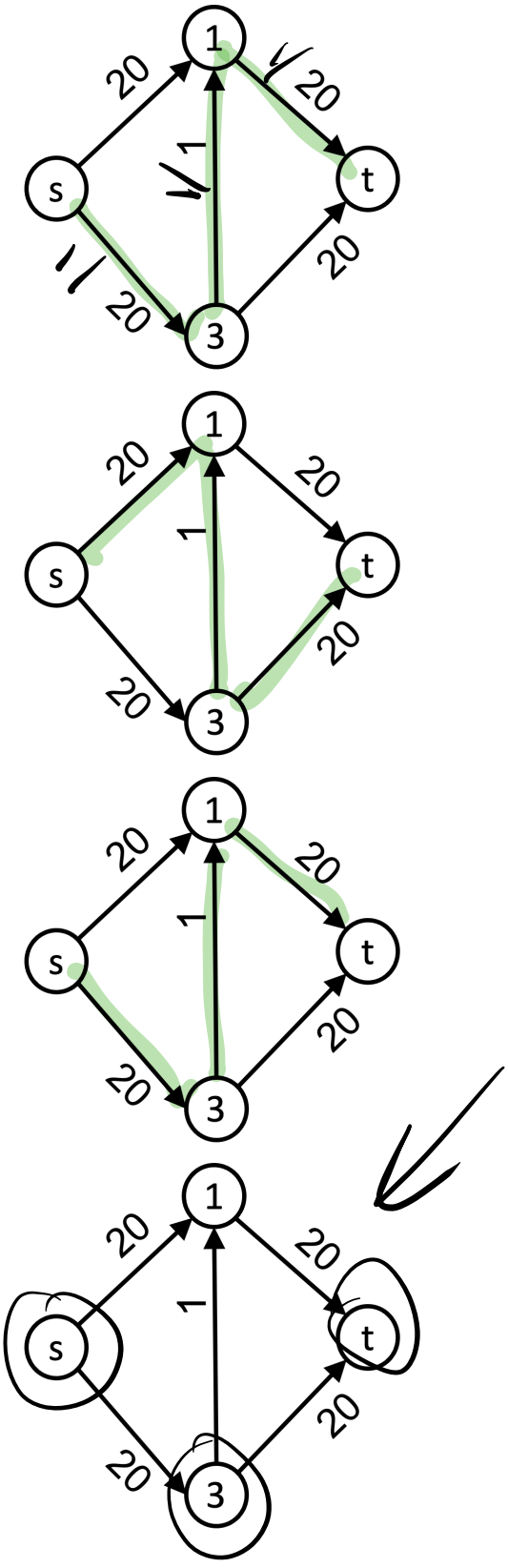
5 Runtime Analysis

- f^* is the maximum flow and $|f^*|$ the maximum flow value
- Assume all weights are integers.
- Let f be the flow we are growing as the algorithm progresses.

We need to answer the following questions:

1. What is the runtime complexity for finding an s-t path p in G_f?
2. What is the minimum amount by which we can increase f in each iteration?
3. What is the maximum number of paths we might have to find before we are done?
4. What is an overall runtime bound for FordFulkersonBasic?

6 How bad can the runtime be in practice?



7 The Edmonds-Karp Algorithm

The Edmonds-Karp Algorithm is a variation on Ford-Fulkerson that chooses an augmenting path p by finding the directed path from s to t with the smallest number of edges. This is accomplished using a:

BFS

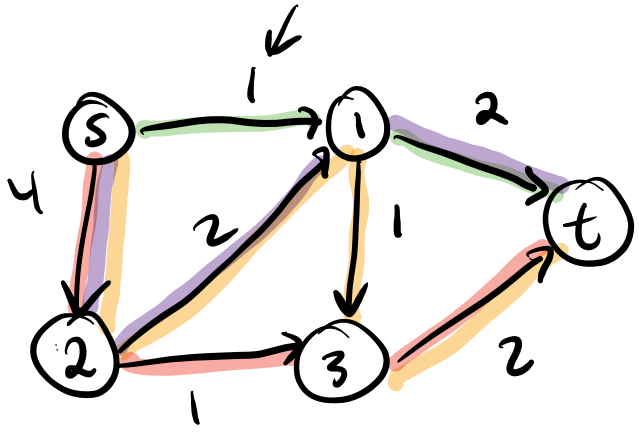
7.1 Shortest path distances increases monotonically

Let f be an s-t flow for input $G = (V, E, s, t)$ and G_f be the residual graph. Define

$$\delta_f(s, v) = \text{ the shortest unweighted path distance from } s \text{ to } v \text{ in } G_f$$

Lemma 3 *For every $v \in V$, the distance $\delta_f(s, v)$ increases monotonically with each flow augmentation.*

Translation: as we keep finding augmenting paths p and sending more flow f_p to f, the distance between s and every node either stays the same, or increases.



Dist(s,t)
2
3
3
4

Total runtime will be $O(VE^2)$

no $|f^*|$

Theorem 4. *The total number of flow augmentation steps performed by Edmonds-Karp is $O(VE)$.*

Proof.

- Let p be an augmenting path in G_f.
- An edge $(u, v) \in p$ is *critical* if $c_f(p) = c_f(u, v)$, meaning it is the smallest capacity edge in that path.

"weakest link"

- When we push $c_f(p)$ flow through p, the edge (u, v) disappears from G_f

Otherwise it was not the weakest link.

- At least one edge on each path p is critical.
- Claim: Each of the $|E|$ edges can be critical at most $|V|/2$ times.

If we prove the claim, we are done.
There are $|E|$ edges, each can be critical $\frac{|V|}{2}$,
and at least one is critical in each iteration.
If there are more than $|E| \cdot \frac{|V|}{2}$ iterations,
by pigeonhole principle, one edge was critical
more than $\frac{|V|}{2}$ times, contradiction.

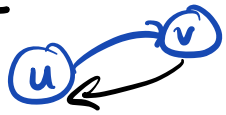
$$|E| \frac{|V|}{2} = O(|E||V|) \text{ iterations.}$$

Proving the claim: (u,v) becomes critical at most $|V|/2$ times.

(u,v) is a pair such that $(u,v) \in E$ or $(v,u) \in E$.

- Let u and v be nodes in some edge in E.
- When (u, v) is critical for the first time, $\delta_f(s, v) = \delta_f(s, u) + 1$

  Why? Because we follow shortest paths.



- Then (u, v) disappears from the residual graph, and can only re-appear after (v, u) is on some future augmenting path. Say that (v, u) is on an augmenting path when the new flow on G is f', then

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$$

- We know that $\delta_f(s, v) \leq \delta_{f'}(s, v)$ by Lemma 3
- So we have

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 \geq \delta_f(s, v) + 1 = \delta_f(s, u) + 2$$

- From the first to the second time (u, v) becomes critical, the distance from s to u increases by at least 2.
- If (u, v) becomes critical more than $|V|/2$ times, then the distance from s to u would be greater than $|V| - 2$.

Contradiction. In an iteration where (u,v) is on a critical path, the distance from s to u can be at most $|V|-2$.

- Thus, (u, v) becomes critical at most $|V|/2 = O(V)$ times.

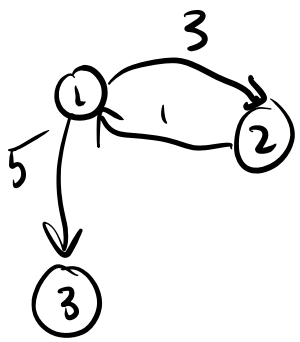
□

Representing a graph by a matrix

The adjacency matrix A of a weighted directed graph G with n nodes is an $n \times n$ matrix where

$$A(i,j) = w(i,j)$$

G



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$