Thursday, Nov. 3

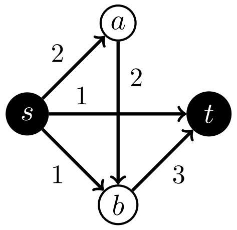
1. Welcome!
2. HW 6 due today-ish
   - ↳ if later than friday, toss an email
3. HW 7 up tomorrow
   - ↳ the last one!
4. Questions?
5. Flows day 2
6. Outro
   - ↳ no Small Work

MSCS & Society
11/7 • JBD • 4:40
Federico Ardila-Mantilla

Math 494: Discrete Optimization 11/3

Today we're continuing our discussion of flows. We worked very hard last time to come up with a linear program for maximum flow, and gave a general form for the dual program. However, that dual was tough to construct without knowing what the matrix really looks like. Let's try an example here (the smallest it could be to really be interesting, but still kinda big).

Example: For the network below, construct the maximum flow LP, write it in a matrix version, and construct the dual. Compare it to the general form that we developed last time.



Primal

$$\begin{cases} \max\ f_{sa} + f_{st} + f_{sb} \\ \text{st} \quad f_{ab} - f_{sa} = 0 \\ \quad f_{bt} - f_{sb} - f_{ab} = 0 \\ \quad f_{sa} \le 2 \\ \quad f_{st} \le 1 \\ \quad f_{sb} \le 1 \\ \quad f_{ab} \le 2 \\ \quad f_{bt} \le 3 \end{cases}$$

$\downarrow$ matrix version

$$\begin{cases} \max \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_{sa} \\ f_{st} \\ f_{sb} \\ f_{ab} \\ f_{bt} \end{bmatrix} \\ \begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{sa} \\ f_{st} \\ f_{sb} \\ f_{ab} \\ f_{bt} \end{bmatrix} \begin{matrix} = \\ = \\ \\ \le \\ \\ \\ \end{matrix} \begin{bmatrix} 0 \\ 0 \\ 2 \\ 1 \\ 1 \\ 2 \\ 3 \end{bmatrix} \\ f_e \ge 0 \end{cases}$$

dual!

$$\begin{cases} \min \begin{bmatrix} 0 & 0 & 2 & 1 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} y_a \\ y_b \\ l_{sa} \\ l_{st} \\ l_{sb} \\ l_{ab} \\ l_{bt} \end{bmatrix} \\ \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_a \\ y_b \\ l_{sa} \\ l_{st} \\ l_{sb} \\ l_{ab} \\ l_{bt} \end{bmatrix} \ge \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \end{cases}$$

y free, $l \ge 0$

$\downarrow$ eq. form

$$\begin{cases} \min \sum c_e l_e \\ \text{st} \quad -y_a + l_{sa} \ge 1 \\ \quad + l_{st} \ge 1 \\ \quad -y_b + l_{sb} \ge 1 \\ \quad y_a - y_b + l_{ab} \ge 0 \\ \quad y_b + l_{bt} \ge 0 \\ \quad y \text{ free}, l \ge 0 \end{cases}$$

if $y_s = -1$
$y_t = 0$

$$\begin{cases} \min \sum c_e l_e \\ \text{st} \quad -1 - y_a + l_{sa} \ge 0 \\ \quad -1 - 0 + l_{st} \ge 0 \\ \quad -1 - y_b + l_{sb} \ge 0 \\ \quad y_a - y_b + l_{ab} \ge 0 \\ \quad y_b - 0 + l_{bt} \ge 0 \\ \quad y \text{ free}, l \ge 0 \end{cases} \rightarrow \begin{matrix} y_u - y_v + l_{uv} \ge 0 \\ \forall\, uv \in E \end{matrix}$$

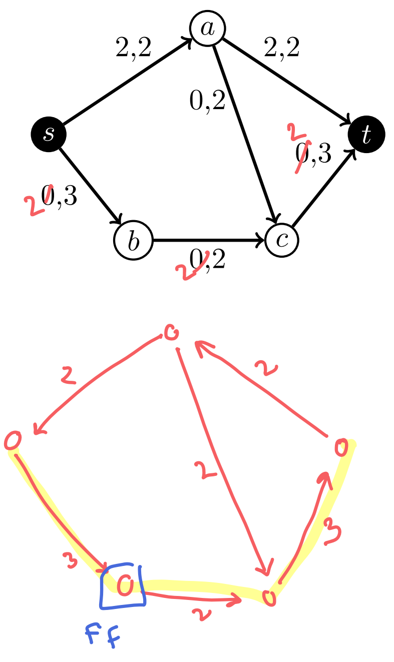
We'll now transition to developing an algorithm for this optimization problem, calling back to the example we did the other day about augmenting a flow. Our algorithm will need to be able to do two things: send more flow if possible, and reroute flow if necessary to allow for a better arrangement. Our algorithm, Ford-Fulkerson, will do just that. However, we need to define the concept of a *residual graph.*

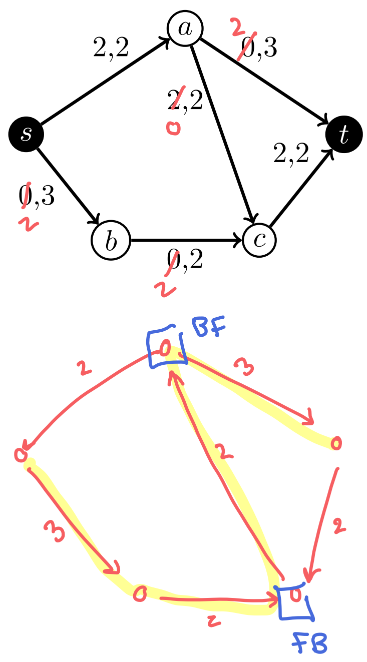
Definition. *(residual graph, augmenting path) The* residual graph G_f *of a network* (G, c, s, t) *with respect to a flow* f *is a weighted directed graph such that* $V(G_f) = V(G)$ *and* $E(G_f)$ *has two types of edges: for any edge* (uv) *in the original network,*

$$w_f(uv) = c(uv) - f(uv) \qquad\qquad w_f(vu) = f(uv)$$

An augmenting path *for the* f *corresponds to an* $s - t$ *path in* G_f.

Example: These are the networks with flows that we saw last time. Build the residual graph for each.





Find an $s-t$ path in G_f for each network. How much do you think we can augment by, and can you make a general statement for augmenting flow along a path?

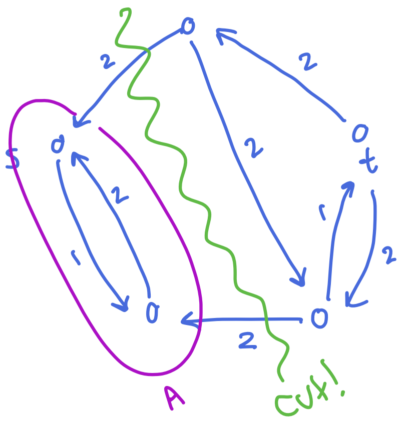
let $\delta(P) = \min\{w_f(e) : e \in P\}$

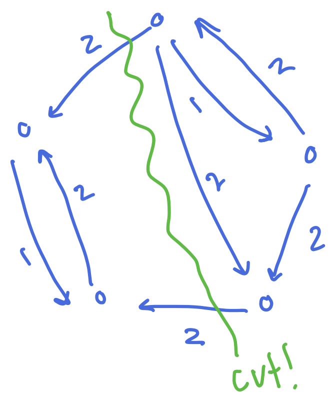
forward edge $\rightarrow +\delta$

back edge $\rightarrow -\delta$

This is the basic step of the Ford-Fulkerson Algorithm: for a flow at any stage, build the residual graph, look for a path, and augment the flow as much as you can.

Example: Build the residual graphs for your newly augmented flow from the examples above. Can you see the minimum cut lurking around the residual?





Example: For an $s-t$ path P in G_f, define $\delta(P) = \min\{w_f(e) : e \in P\}$. Consider a new flow f^*, where

$$f^*(uv) = \begin{cases} f(uv) & \text{if } uv \notin P \\ f(uv) + \delta(P) & \text{if } uv \text{ is a forward edge in } P \\ f(uv) - \delta(P) & \text{if } uv \text{ is a backwards edge in } P \end{cases}$$

Claim. *f^* is a feasible flow.*

Pf: need to show capacity and conservation constraints hold.

*capacity: only need to check forward edges (places we add flow). Note, for every forward edge uv, $\delta(P) \le c(uv) - f(uv) \to f(uv) + \delta(P) \le c(uv)$

*conservation: let $v \in V \setminus \{s,t\} \in P$. We proceed w/ cases.

$\boxed{1}$ $\xrightarrow{F} \underset{v}{\circ} \xrightarrow{F}$: flow in $+\delta$, flow out $+\delta$, no change.

$\boxed{2}$ $\xrightarrow{F} \circ \xleftarrow{B}$: flow in $+\delta$, flow in $-\delta$, no change.

$\boxed{3}$ $\xleftarrow{B} \circ \xrightarrow{F}$: flow out $-\delta$, flow out $+\delta$, no change.

$\boxed{4}$ $\xleftarrow{B} \circ \xleftarrow{B}$: flow in $-\delta$, flow out $-\delta$, no change.

Algorithm 1 Ford-Fulkerson Algorithm $FF(G, c, s, t)$

```
1: Initialize $f(uv) = 0$ for every edge $uv \in E(G)$.
2: Construct $G_f$ with respect to $f$.
3: while there exists an $s-t$ path $P$ in $G_f$ do
4:     $\delta(P) = \min\{w_f(e) : e \in P\}$
5:     Augment the flow:
           $f(uv) = \begin{cases} f(uv) & \text{if } uv \notin P \\ f(uv) + \delta(P) & \text{if } uv \text{ is a forward edge in } P \\ f(uv) - \delta(P) & \text{if } uv \text{ is a backwards edge in } P \end{cases}$
6:     $|f| = |f| + \delta(P)$
7:     Reconstruct $G_f$ with respect to $f$.
8: end while
```

Question: How can we find a the $s-t$ path during this algorithm?

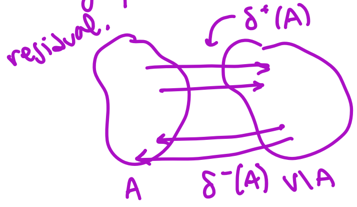
BFS

Example: Prove the following claim.

Claim. *Let (G, c, s, t) be a network with flow f. Then f is maximum if and only if there is no augmenting path. Moreover, this maximum flow defines a cut A such that $|f| = c(A)$, and thus maximum flow equals minimum cut.*

Pf: ($\Rightarrow$) assume f is max. Then no avg. path.

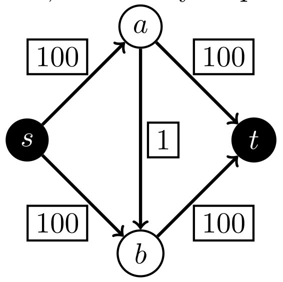
($\Leftarrow$) assume no avg. path. Let A be the set of s reachable vertices in the residual graph. Note, $s \in A$, $t \notin A$, thus A forms a cut.



Note, $\delta^+(A) = \emptyset$. If there were, we didn't build A right. This means any edge in (G, c, s, t) that goes from A to $V \setminus A$ is maxed out. This is the entire flow, so $|f| = c(\delta^+(A))$ in the network.

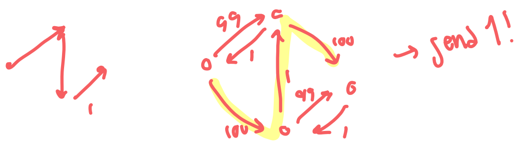
Thus f is a max flow, since it achieves the value of a cut. This cut is therefore min.

Example: We don't go into run time much in this class, but it turns out that Ford-Fulkerson can do a really bad job of building flow if we're unlucky. Considering the graph below, how many steps will F-F take in a best case scenario? How about a worst case one?



Best case: 2 steps! :)

Worst case: 200 steps! :(



Question: can we do anything about this?

yes! implementing a shortest st path will help.

↳ Edmonds-Karp

There is so much more to mention about network flows, including a labeling version of F-F, path packing versions of the LPs that correspond to flow decompositions, integrality of flows created and when F-F might not terminate, but we can't really do all that here. If it's interesting, you could explore it for a project!