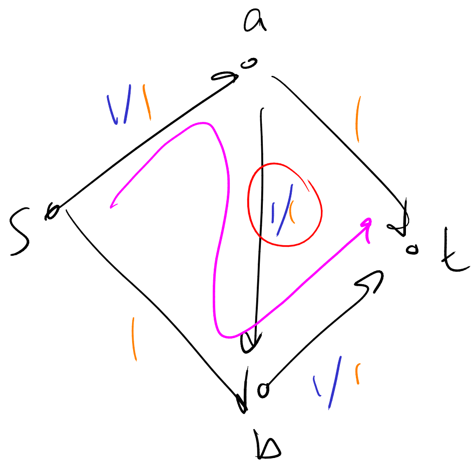
GreedyFlow(G, s, t):

while there exists an unsaturated path P from $s \to t$:

$\alpha \leftarrow$ minimum remaining capacity along P

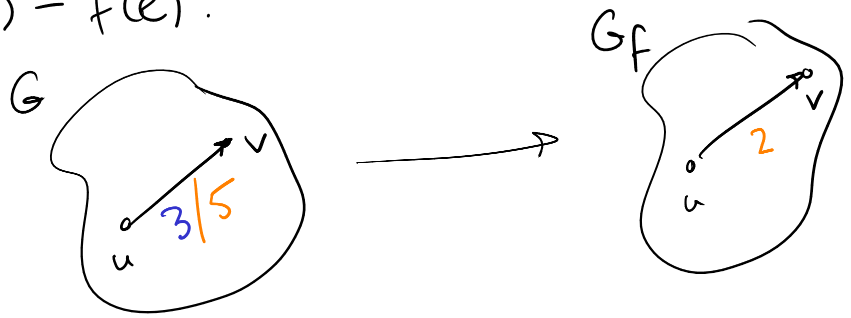
for each edge $e \in P$:

Add α to $f(e)$.

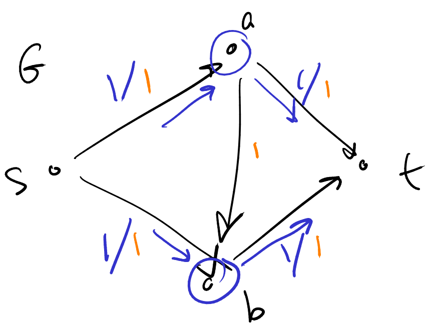


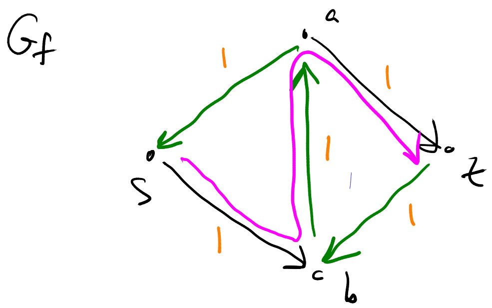
<u>Def'n</u> Given a flow network G, with flow f, define the <u>residual network</u> G_f as follows:

- The vertices of G_f are the same as G.
- For each edge $e = (u, v)$ in G, if $f(e) < c(e)$, then add an edge $u \to v$ in G_f with capacity $c(e) - f(e)$.



- For each edge $e = (u, v)$ in G, if $f(e) > 0$, then add an edge $v \to u$ in G_f with capacity $f(e)$.





Ford-Fulkerson (G, s, t):

while there exists a path P in G_f from $s \to t$:

 $\alpha \leftarrow$ minimum remaining capacity along P

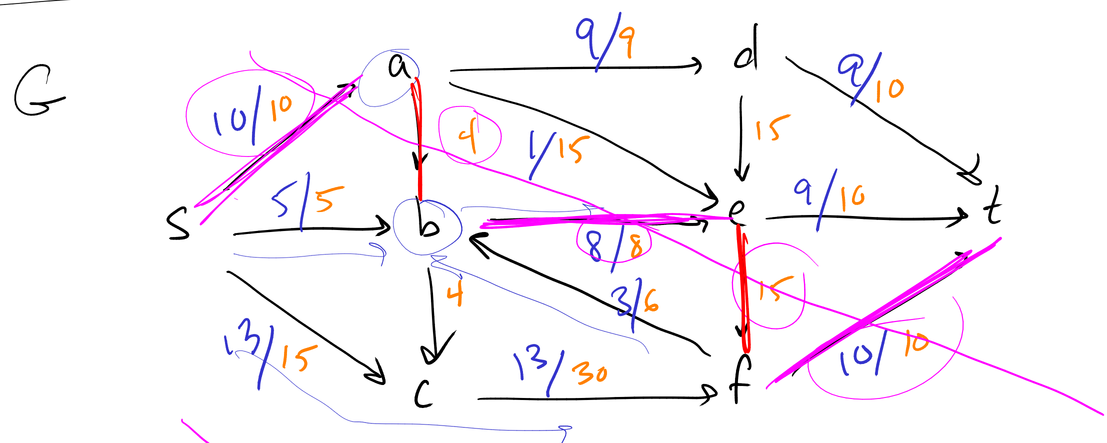
 for each edge $e \in P$:

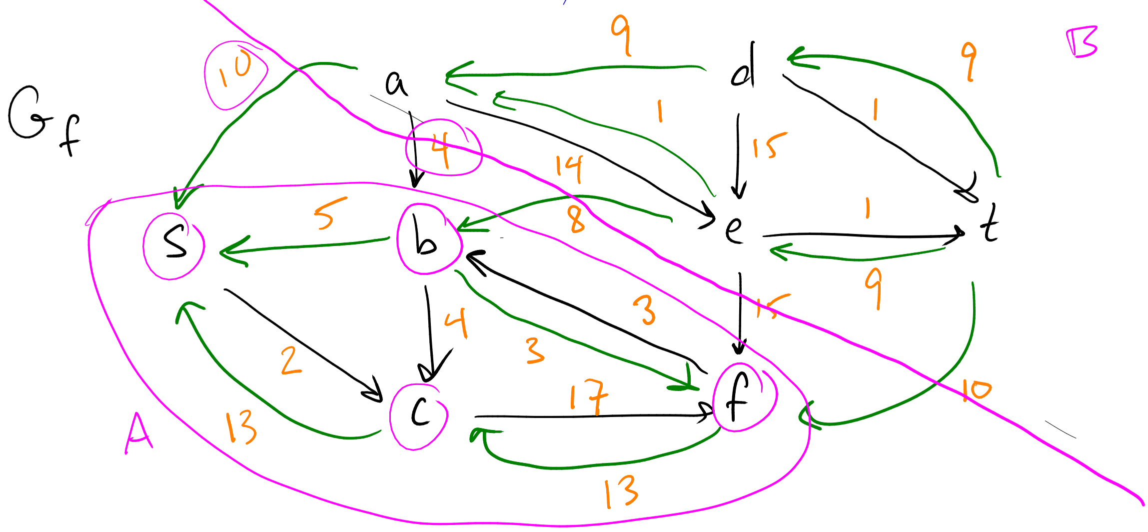
  if e is a "forwards edge":

   Add α to $f(e)$.

  if e is a "backwards edge":

   Subtract α from $f(e)$.



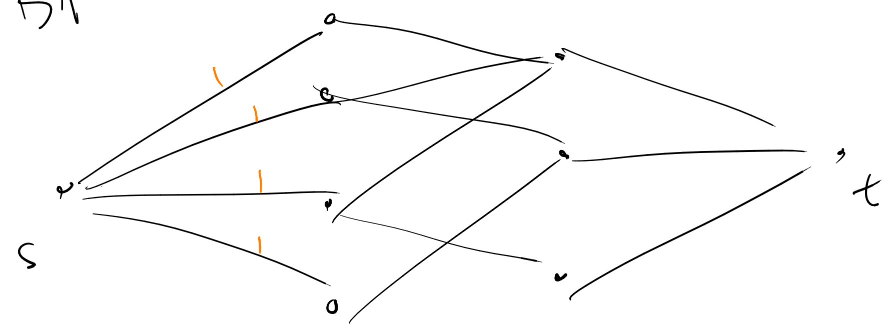


How long does F-F take?

- Always add flow to an edge coming out of s each loop
- Hence, value of the flow increases by at least 1 each loop.

- Finding path + updating everything can be done in $O(E)$ time (BFS/DFS).
- Hence algorithm takes $O(C \cdot E)$ where C = capacity coming out of s.

eg Bipartite matching:



$$C = O(V)$$
$$E = O(V^2) + O(V) + O(V) = O(V^2).$$

Hence finding max flow takes $O(CE) = O(V^3)$.

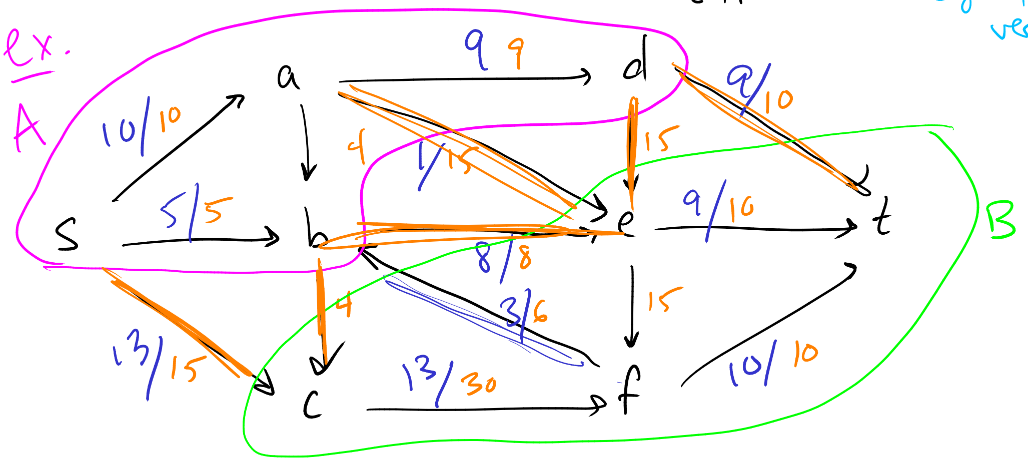
Def'n Given a flow network G, an <u>s-t cut</u> is a partition of the vertices into 2 sets (A, B) such that $s \in A$ and $t \in B$.

Def'n The capacity of an s-t cut (A,B) is

$$c(A,B) = \sum_{e: A \to B} c(e).$$

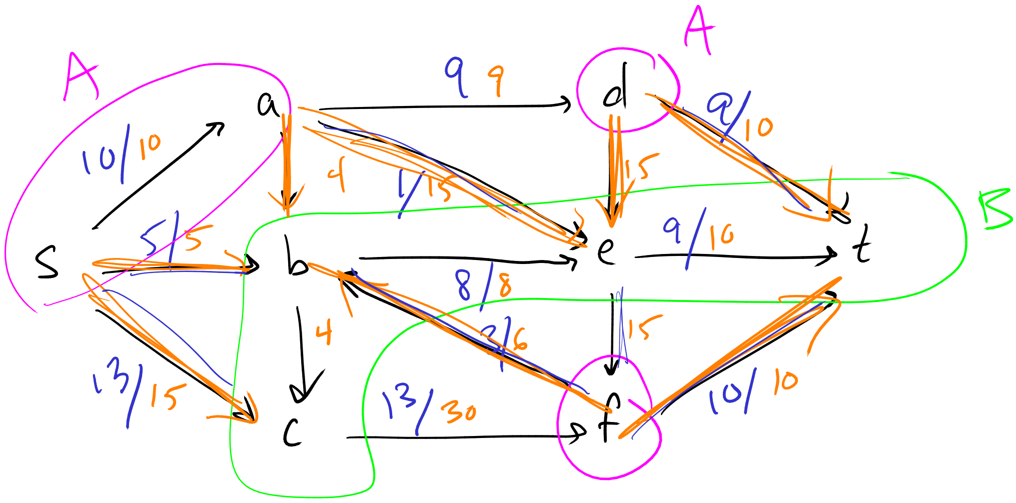
edges from vertex in A to vertex in B

ex.



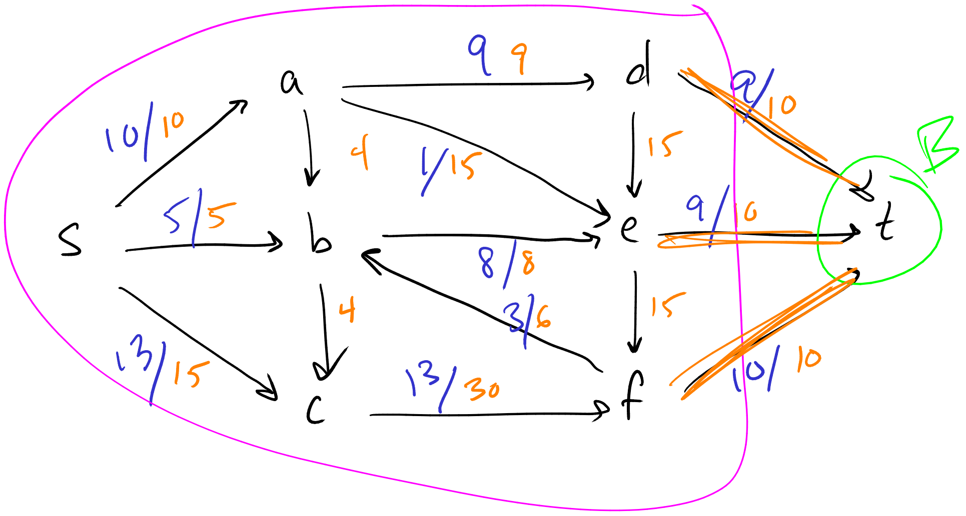
$c(A,B) =$
$15 + 4 + 8 + 15 + 15 + 10$
$= 67$

$v(A,B) = 13 + 8 + 1 + 9 - 3$
$= 28.$



$c(A,B) = 15+5+4+15+15+10+6+10 = 80.$

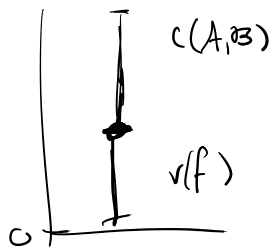
$v(A,B) = \cancel{13}+5+1+9+10+3\cancel{-13} = 28$



$c(A,B) = 10+10+10 = 30.$

$v(A,B) = 9+9+10 = 28.$

Lemma (bottleneck): Let f be a flow on G and (A,B) be an s-t cut. Then $v(f) \le c(A,B)$.



Corollary: If $v(f) = c(A,B)$, then f is a max flow and (A,B) is a "min cut" (i.e. cut w/ min possible capacity).

Def'n Given G and an s-t cut (A,B), the net flow with respect to (A,B) is:

$$v(A,B) = \sum_{e:A\to B} f(e) - \sum_{e:B\to A} f(e)$$

("net exports from A to B").

Lemma: For any flow and any s-t cut (A,B),

$$v(A,B) = v(f).$$

Proof $v(f) = \sum_{e \text{ out of } s} f(e) = \sum_{x \in A} f^{out}(x) - f^{in}(x) = \sum_{e: A \to B} f(e) - \sum_{e: B \to A} f(e) = v(A,B)$

flow coming into x

Lemma $v(A,B) \le c(A,B)$.

Proof
$$v(A,B) = \sum_{e: A\to B} f(e) - \sum_{e: B \to A} f(e)$$
$$\le \sum_{e: A \to B} f(e)$$
$$\le \sum_{e: A \to B} c(e) = c(A,B)$$

Note $v(f) = v(A,B) \le c(A,B) \longrightarrow$ bottleneck lemma.

Theorem. Given a flow network G and flow f, the following 3 statements are equivalent:

(1) $v(f) = c(A,B)$ for some s-t cut (A,B).

(2) f is a max flow, and (A,B) is a min cut.

(3) There are no paths from $s \to t$ in G_f. ←

Proof

$(1) \Rightarrow (2)$: Corollary from before.

$(2) \Rightarrow (3)$: ie. if f is max flow, no $s \to t$ in G_f.

Contrapositive: if there is $s \to t$ path in G_f, then f not max flow.

$s \to t$ path in G_f is an "augmenting path", we can increase flow along it, so f is not max.

$(3) \Rightarrow (1)$: No $s \to t$ paths in G_f, then $v(f) = c(A,B)$ for some (A,B),

Suppose no $s \to t$ paths in G_f. Define A = all vertices reachable in G_f from s, $B = V - A$. Note $s \in A$, since s is reachable from itself. $t \in B$ because we assumed t not reachable from s.

Claim: all edges from $A \to B$ are full, all $B \to A$ are empty.

- edges $A \to B$ must be full, since o'wise there would be an edge $A \to B$ with remaining capacity in G_f, but by def'n B is not reachable
- edges $B \to A$ must be empty, since o'wise there would be a backwards edge in G_f.

So, $\underline{v(f)} = v(A,B) = \sum_{e: A \to B} f(e) - \sum_{e: B \to A} f(e)$

lemma.

$$= \sum_{e: A \to B} c(e) - \sum_{e: B \to A} 0$$

$$= \sum_{e: A \to B} c(e) = \underline{c(A,B)}.$$

$\blacksquare$

<u>Cor</u>: F-F stops when (3) is true, hence also (2) is true, ie. F-F correctly finds a max flow (+ min cut).