CMSC 451 - Algorithm Design
Lecture 12 - Network Flow - Basic Concepts

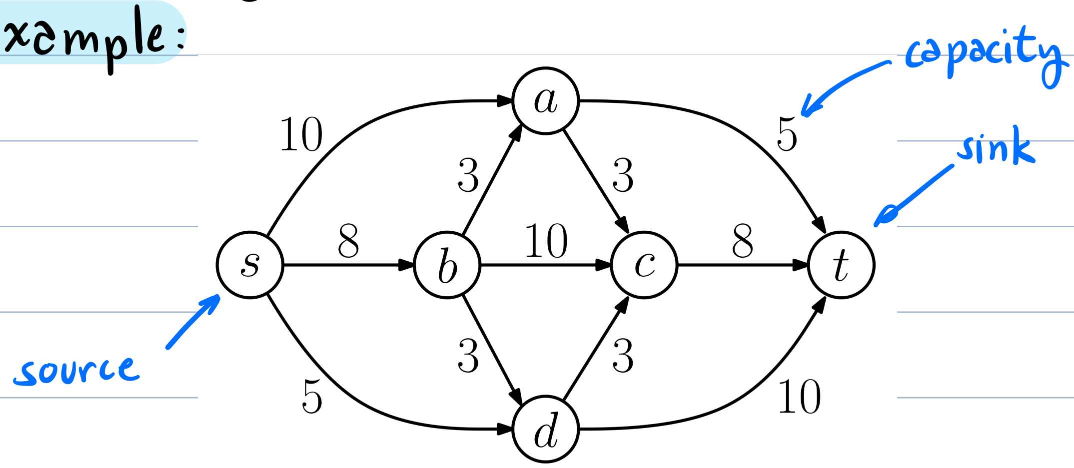
Network Flow is a classical problem, which emerged from the field of operations research, a branch of applied math.

A **flow network** (or **s-t network**) is a directed graph $G=(V,E)$, where each edge $(u,v) \in E$ has an associated capacity $c(u,v) \geq 0$ + there are two special vertices:

source (s) + sink (t)

(No edges enter s, no edges leave t)

Example:



Max-Flow: Thinking of edge (u,v) as a pipe that can carry $c(u,v)$ units of flow, how much flow can we push from s to t?

What do mean by flow?

- A flow is a function f mapping each edge to a real number $f(u,v) \geq 0$
- Satisfies:
    - Capacity constraint:

$$\forall (u,v) \in E,\ f(u,v) \leq c(u,v)$$

(flow cannot exceed capacity)

    - Flow conservation (or balance):

$$\forall v \in V \setminus \{s,t\}$$

$f^{out}(v)$

$$\sum_{(u,v)\in E} f(u,v) = \sum_{(v,w)\in E} f(v,w)$$

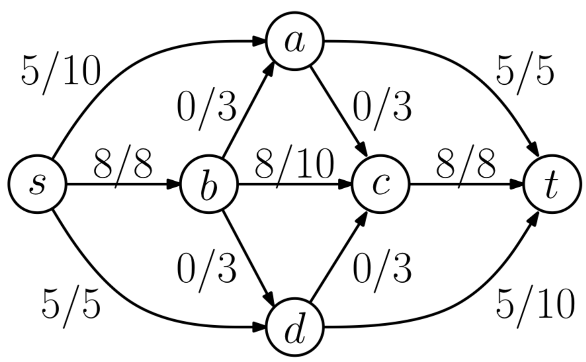
$f^{in}(v)$

(flow in = flow out, except at source + sink)

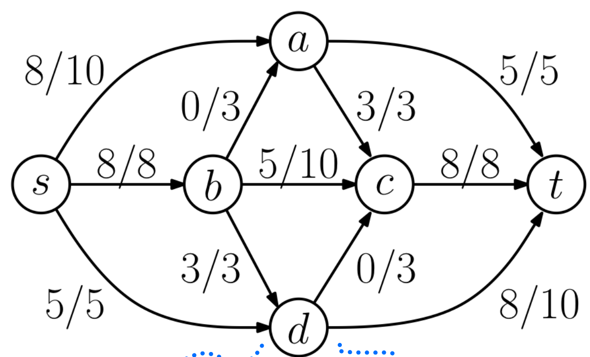
2 Examples:

$$u \xrightarrow{f(u,v)/c(u,v)} v$$

Flow 1: f_1



Flow 2: f_2



$f^{in}(d) = 5+3$

$f^{out}(d) = 0+8$

Want to maximize total flow value, defined

$$|f| = f^{out}(s) = f^{in}(t)$$

Flow conservation implies
flow out of s = flow into t

$|f_1| = 5+8+5 = 18$

$|f_2| = 8+8+5 = 21$ (This the max flow)

Max-Flow Problem: Given a flow network, compute the flow of max total value

This is a heavily studied problem with a long history: $n = |V|$, $m = |E|$, C = sum of capacities

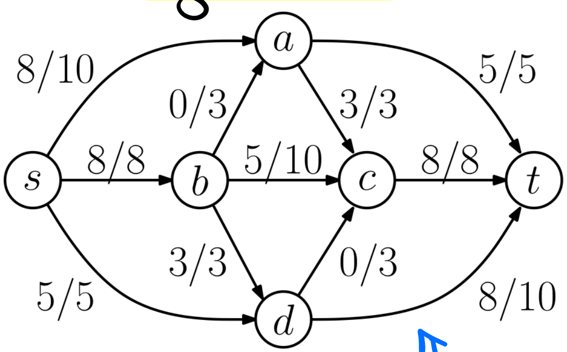
- Ford-Fulkerson (1956) - $O((n+m)C)$
- Dinitz (1970) - $O(n^2 \cdot m)$
- Edmonds-Karp (1972) - $O(n \cdot m^2)$
- Gabow (1985) - $O(nm \log C)$
- Goldberg-Tarjan (1986) - $O(nm \log n^2/m)$

Path-Based View: An equivalent way to view a flow is as a collection of paths from s to t (like wires)

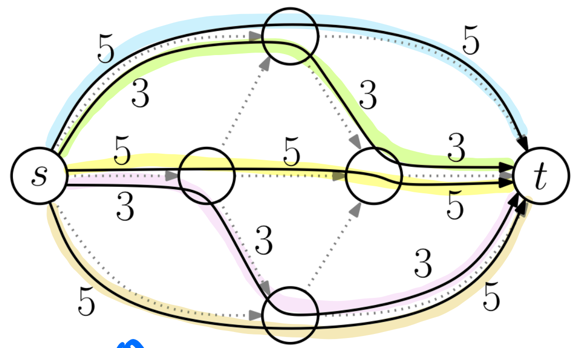
- Each s-t path is assigned a weight
- Sum of weights cannot exceed edge capacity

Example:

Edge-based:



Path-based



$|f| = 8+8+5$
$= 21$

$|f| = 5+3+5+3+5$
$= 21$

Just a different perspectives on the same math. concept

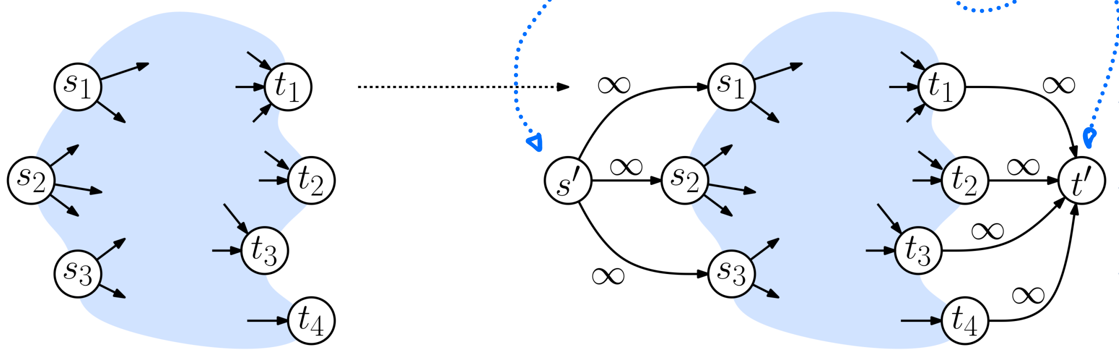
- some algorithms are more edge based
- some are more path based

Proof-Exercise

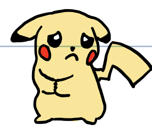
> **Claim:** An s-t network has an edge-based flow of value x iff it has a path-based flow of value x.

Why just one source/sink?

- Can easily simulate multiple sources/sinks.
- Add "super source" and/or "super sink" + connect to others.



- Can you also specify linkages? (e.g. all flow from s_i must go to t_i)
  - No - Called mult-commodity flow
    - NP-hard!

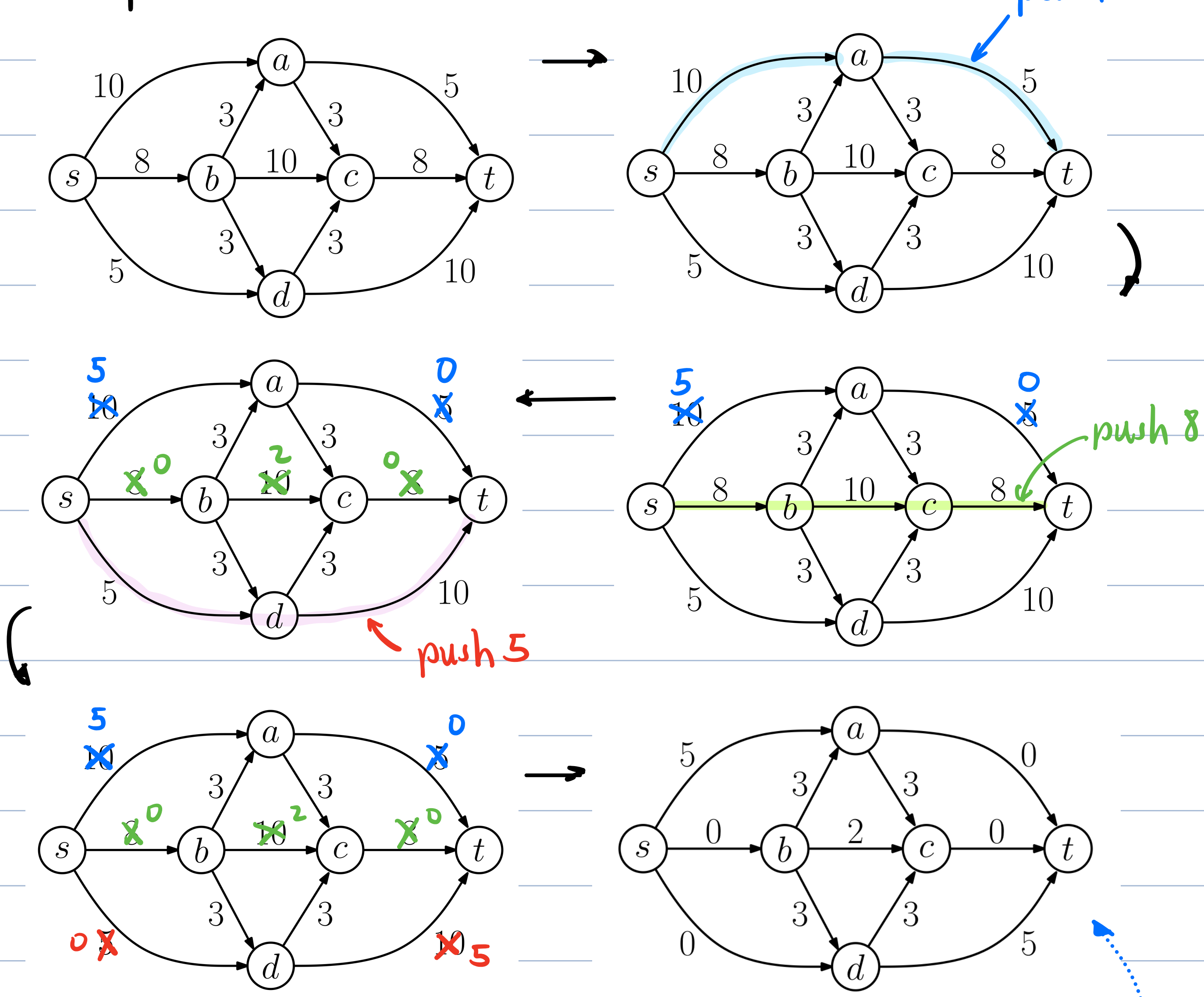


Max-Flow - Why greedy fails

Simple greedy strategy:

- find any path from s to t of strictly positive capacities
- push as much flow as you can along this path
- reduce (remaining) capacities on edges of path

Example:



Total flow = 5 + 8 + 5 = 18

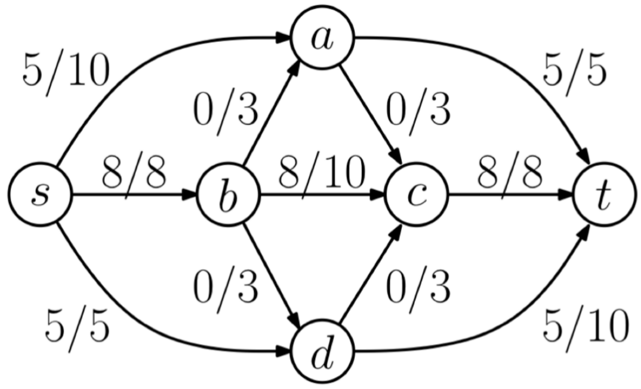
Not optimal!

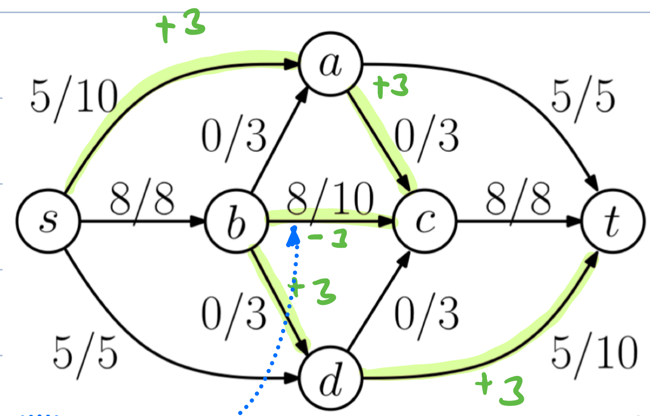
But no more pos.-capacity s-t paths!

How to fix greedy?

- We need to consider s-t paths that both:
  - add new flow where capacity exists
  - reduce flow where flow exists

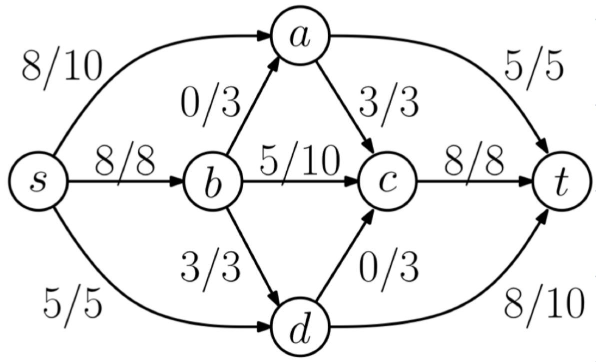
$|f| = 18$ (not optimal)





Edge (b,c) carried flow of 8. We can reduce this by -3

Conserves flow balance



$|f| = 21$ optimal!

How to formalize augmenting/reducing paths?

Residual Network - Given s-t network G + flow f, define G_f to be s-t network

- same vertices as G
- Forward edges (can add more flow)

  for $(u,v) \in E$ s.t. $f(u,v) < c(u,v)$

  add edge (u,v) with capacity:

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

Intuition: can add $c_f(u,v)$ more flow

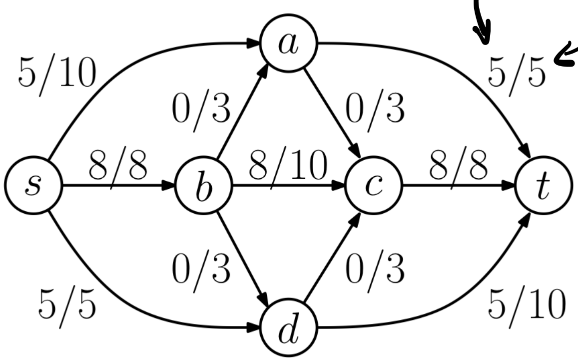
- Backward edges (can reduce existing flow)
  for $(u,v) \in E$ s.t. $f(u,v) > 0$
  add reverse edge (v,u) with capacity:
  $$c_f(v,u) = f(u,v)$$
  Intuition: can turn off $c_f(u,v)$ flow

Example:

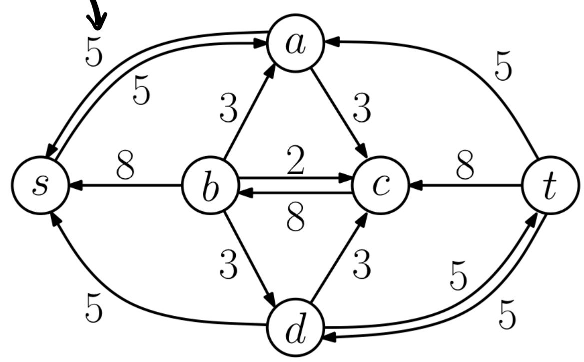
flow f

capacity

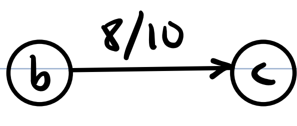
G:



G_f:

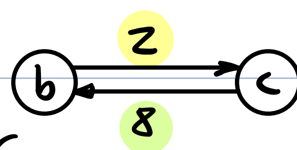


for example:



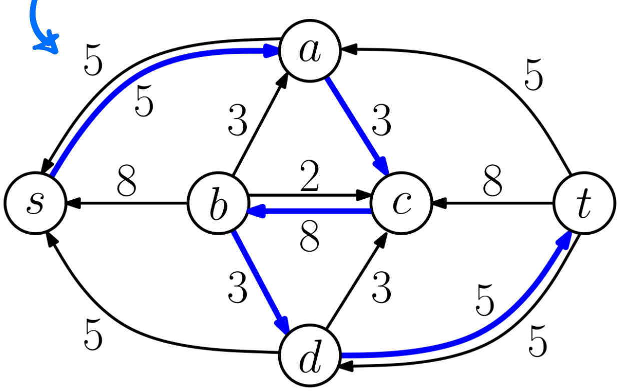
G

can add +2
or
reduce by 8

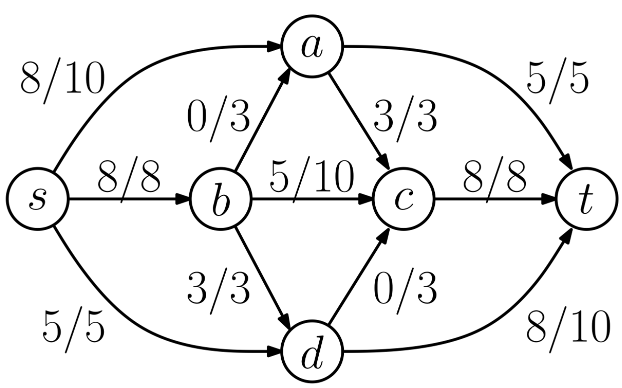


G_f

In G_f, we can push +3 flow on this path.



$|f| = 21$ (optimal)



Questions:

(1) If G_f has a flow, can we use it to increase G's flow?

(2) If we repeat this process, will it lead to the optimal flow (or get stuck, like greedy)?

Both answers are "yes"!

Prove this later

(1) Straightforward.

(2) Not so easy (Min-Cut / Max-Flow Thm)

Claim: Given network G + flow f, if f' is a flow in G_f, then $f+f'$ is a flow in G.

Proof sketch:

Capacity constraint: if (u,v) is forward edge

f valid for G $\Rightarrow$ $0 \leq f(u,v) \leq c(u,v)$

f' valid for G' $\Rightarrow$ $0 \leq f'(u,v) \leq c_f(u,v)$

def. of c_f $\Rightarrow$ $c_f(u,v) = c(u,v) - f(u,v)$

$$\Rightarrow \quad 0 \leq f(u,v) + f'(u,v) \leq \cancel{f(u,v)} + (c(u,v) - \cancel{f(u,v)})$$

$$= c(u,v) \quad \checkmark$$

(Leave flow conservation + backward edges as exercise.)

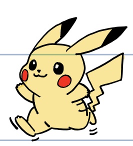
Ford-Fulkerson Algorithm:

Called an augmenting path

- Init flow: $f = 0$
- Find an s-t path in residual graph, G_f
- $f' \leftarrow$ max. flow on this path
- Update residual graph based on new flow: $f \leftarrow f + f'$

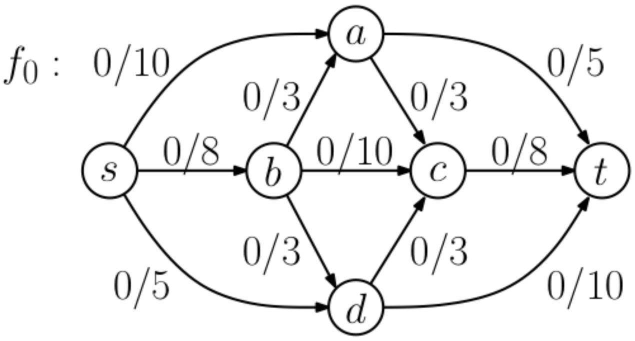
```
ford-fulkerson(G, s, t)
    f ← 0                          // set f(u,v) ← 0, ∀(u,v) ∈ E
    while (true)
        G_f ← compute resid. graph for f      // O(n+m)
        if (G_f has no s-t path)              // DFS → O(n+m)
            return f                          // f is max flow
        π ← any path from s to t in G_f
        c ← min capacity of any edge on π
        increase f by adding +c to edges of π
```

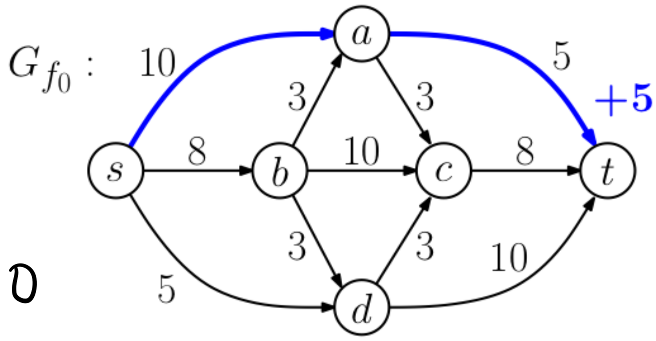
Running time:



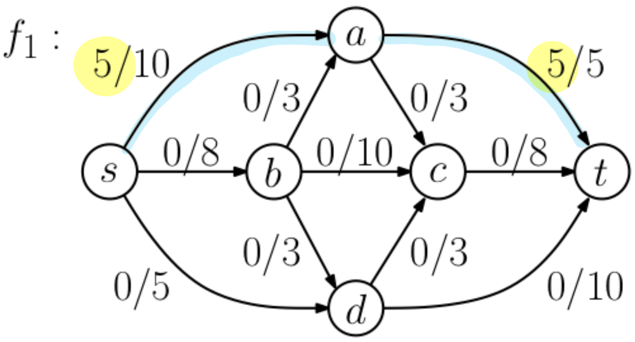
- Each iteration of the while loop can be done in $O(n+m)$ time (DFS)
- We'll discuss number of iterations in next lecture.

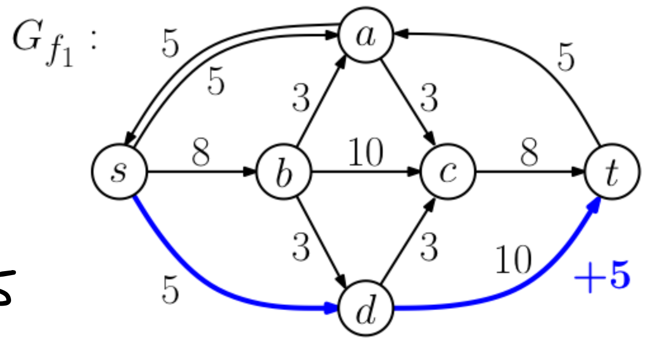
Example:



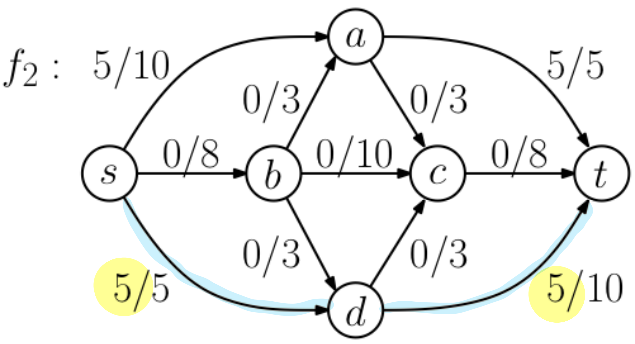


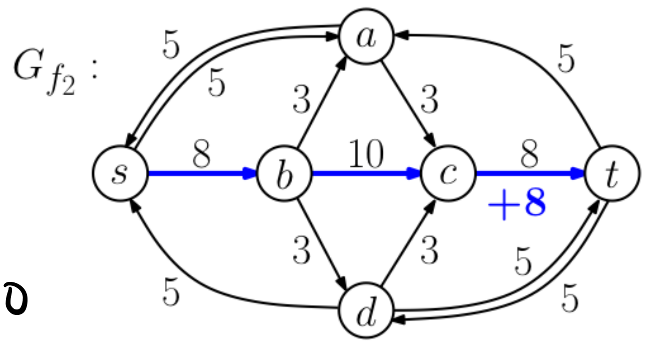
$|f_0| = 0$



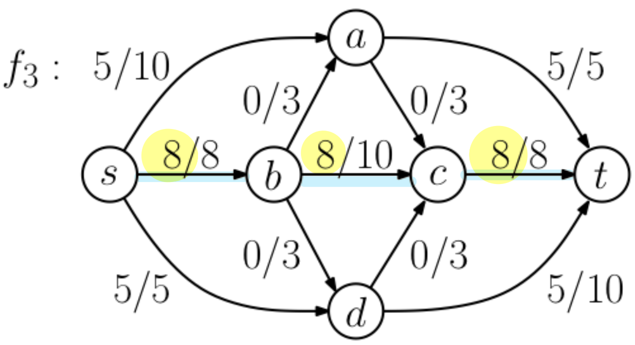


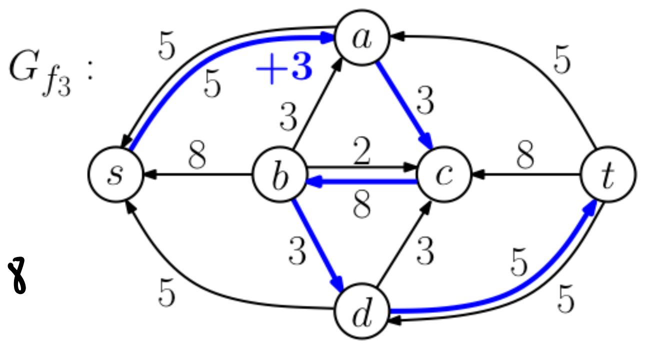
$|f_1| = 5$





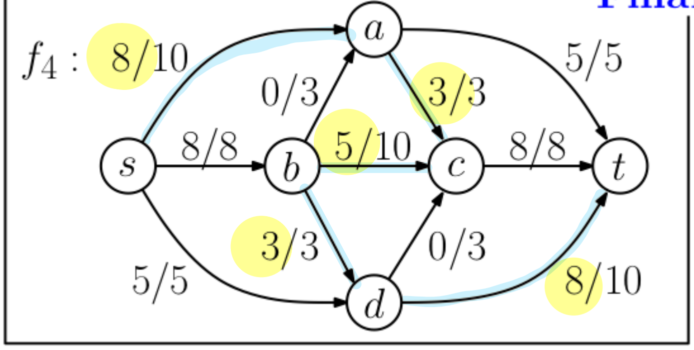
$|f_2| = 10$





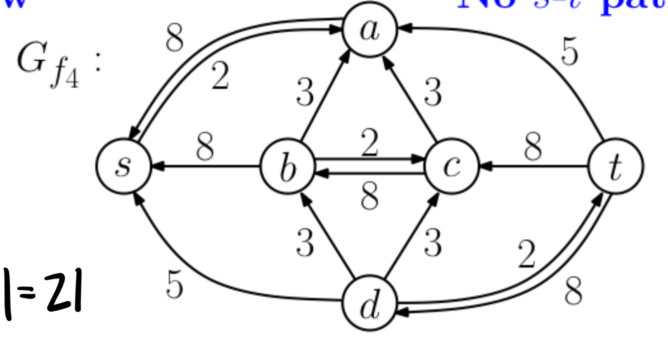
$|f_3| = 18$

Final flow



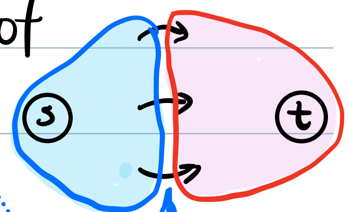
$|f_4| = 21$

No s-t path



Correctness:

- Easy to see that F-F produces a valid flow.
- On termination - Is it optimal?
- To prove this we need to introduce a related concept - cut
- Intuitively - Flow cannot be increased because it saturates all edges of a bottleneck.
- Removing these bottleneck edges cuts the network in two



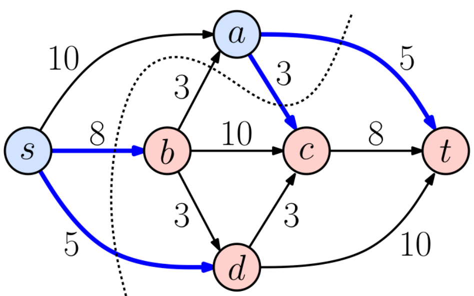
Definition: Given an s-t network, a cut is a partition of the vertex set X, Y such that $s \in X$ & $t \in Y$.

The capacity of cut (X, Y) is the sum of capacities from X to Y,

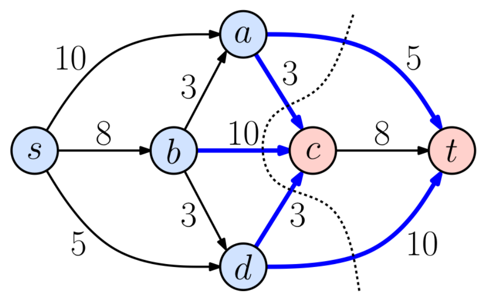
$$c(X,Y) = \sum_{x \in X} \sum_{y \in Y} c(x,y)$$

if $(x,y) \notin E$, $c(x,y) = 0$

$X = \{s, a\} \quad Y = \{b, c, d, t\}$
$c(X,Y) = 5 + 3 + 8 + 5 = 21$



$X = \{s, a, b, d\} \quad Y = \{c, t\}$
$c(X,Y) = 5 + 3 + 10 + 3 + 10 = 31$



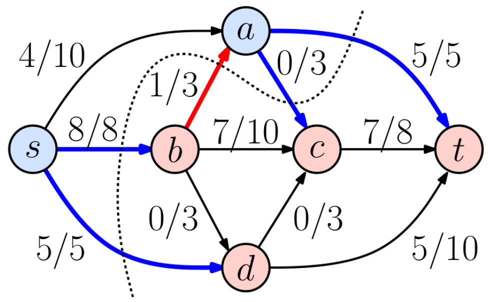
Definition: Given G, a flow f, + a cut (X,Y) define the net flow across the cut to be sum of $X \to Y$ flows minus the $Y \to X$ flows.

$$f(X,Y) = \sum_{(x,y) \in X \times Y} f(x,y) - \sum_{(y,x) \in Y \times X} f(y,x)$$

Example: Two cuts on the same flow, $|f| = 17$

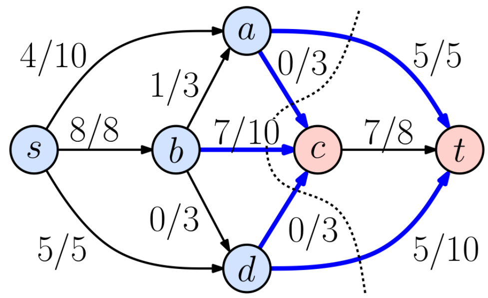
$X = \{s, a\}$ $\quad Y = \{b, c, d, t\}$

$f(X,Y) = (5+0+8+5) - 1 = 17$



$X = \{s, a, b, d\}$ $\quad Y = \{c, t\}$

$f(X,Y) = 5+0+7+0+5 = 17$

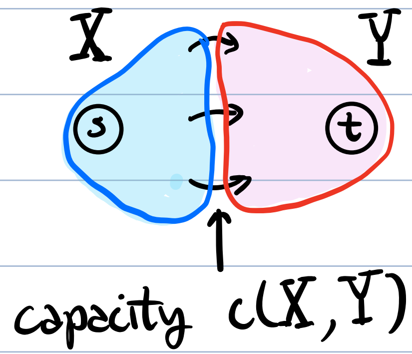


Intuitively – The flow across any cut $= |f|$ (by flow conservation)

> **Lemma:** Given any network G, any flow f, and any cut (X,Y), $f(X,Y) = |f|$.

(Proof given in pdf lecture notes)

Given any cut (X, Y), all the flow must cross over the edges of the cut. Thus:



> **Lemma:** Given any network G, any flow f, and any cut (X, Y):
>
> $$|f| \leq c(X, Y)$$

This holds for every cut, so it holds for the minimum-capacity cut.

To prove that the F-F algorithm is optimal, it suffices to prove that the F-F flow equals the min-capacity cut.

This is a consequence of the following famous theorem

> **Max-Flow/Min-Cut Theorem:**
>
> The following are equivalent:
>
> (i) f is a max flow for G
>
> (ii) Residual network G_f has no s-t path
>
> (iii) $|f| = c(X, Y)$ for some cut (X, Y) of G

Proof:

(i) $\Rightarrow$ (ii) [by contradiction]

If G_f had an s-t path, we could push flow on this path, increasing flow value. ✓

(ii) $\Rightarrow$ (iii)

- Let X = vertices reachable from s in G_f
- Let $Y = V \setminus X$ (all remaining vertices)
- Since no s-t path, $t \in Y$.

$\Rightarrow$ (X, Y) is a cut

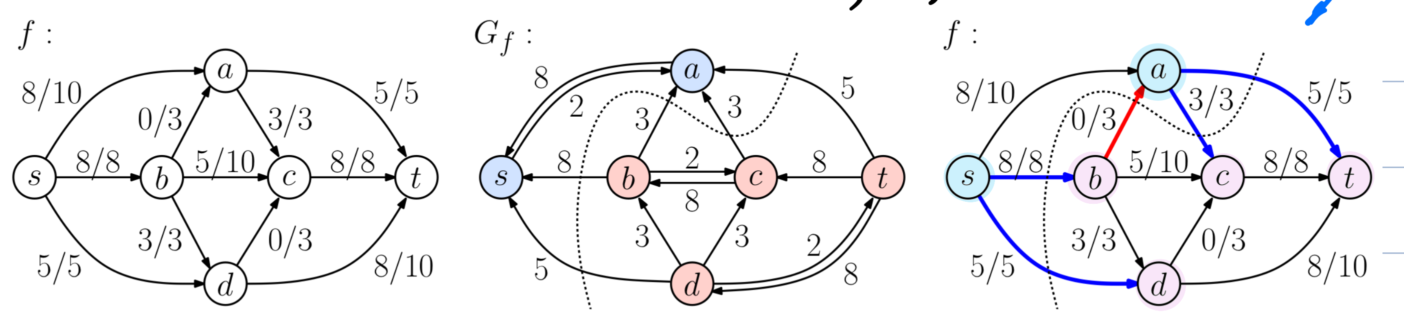
$\Rightarrow$ for each $(x,y) \in X \times Y$

$f(x,y) = c(x,y)$ ← Edges crossing cut are saturated

$\Rightarrow$ $f(X,Y) = c(X,Y)$

- By previous lemma, $f(X,Y) = |f|$

$\Rightarrow$ $|f| = c(X,Y)$ ✓



(iii) $\Rightarrow$ (i)

- By previous lemma $|f| \le c(X,Y)$ for all flows f + all cuts (X,Y)
- If equality is attained for any flow and any cut, this flow must be maximum. ✓

Summary:

- Flow networks + flows
- Max-flow problem
  - Greedy fails
- Residual network + augmenting paths
- Ford-Fulkerson Algorithm

  (find augmenting path, update residual)
- Cuts + capacities
- Max-Flow/Min-Cut Theorem

  (Max flow ≡ Min Cut)
- Whew!