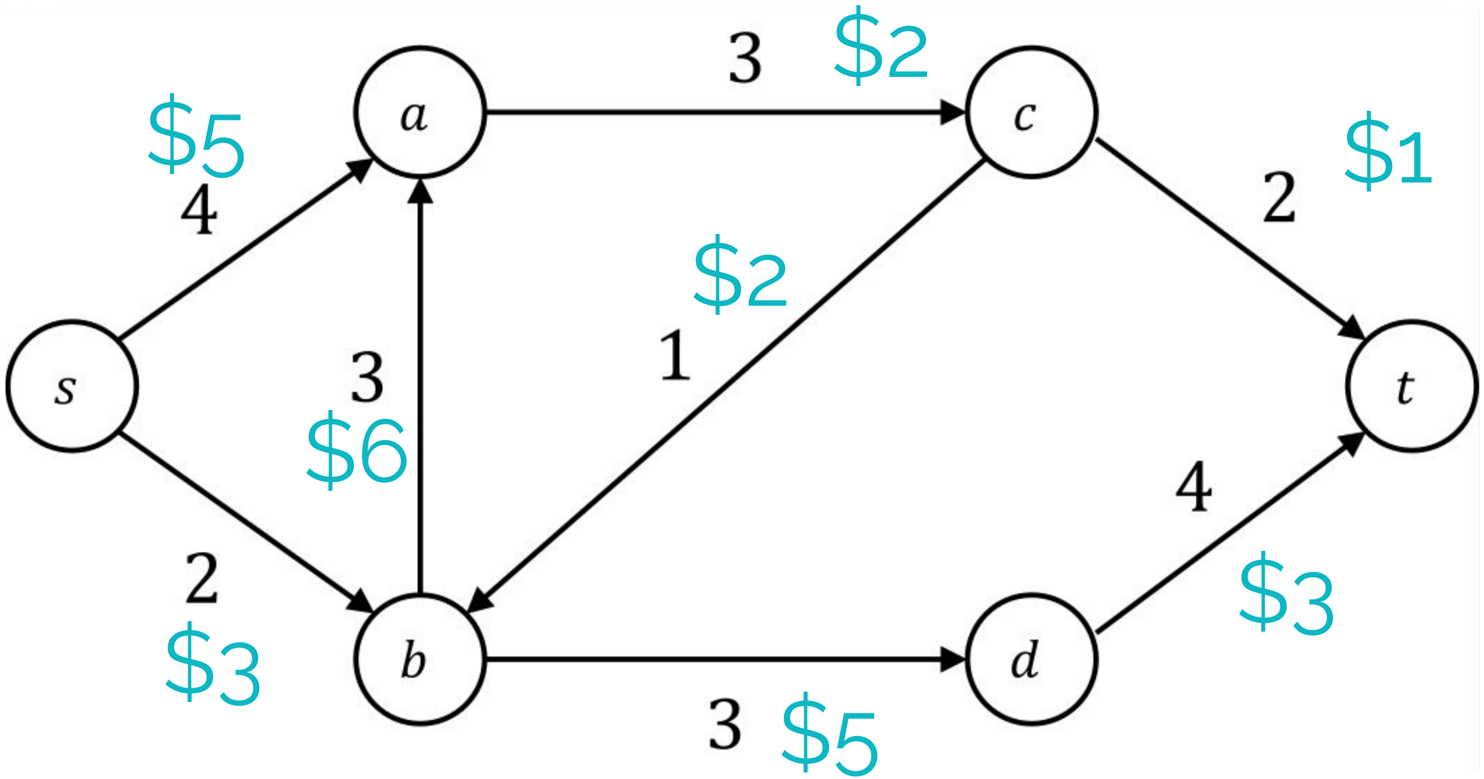
15451 Spring 2023

Min-cost flows

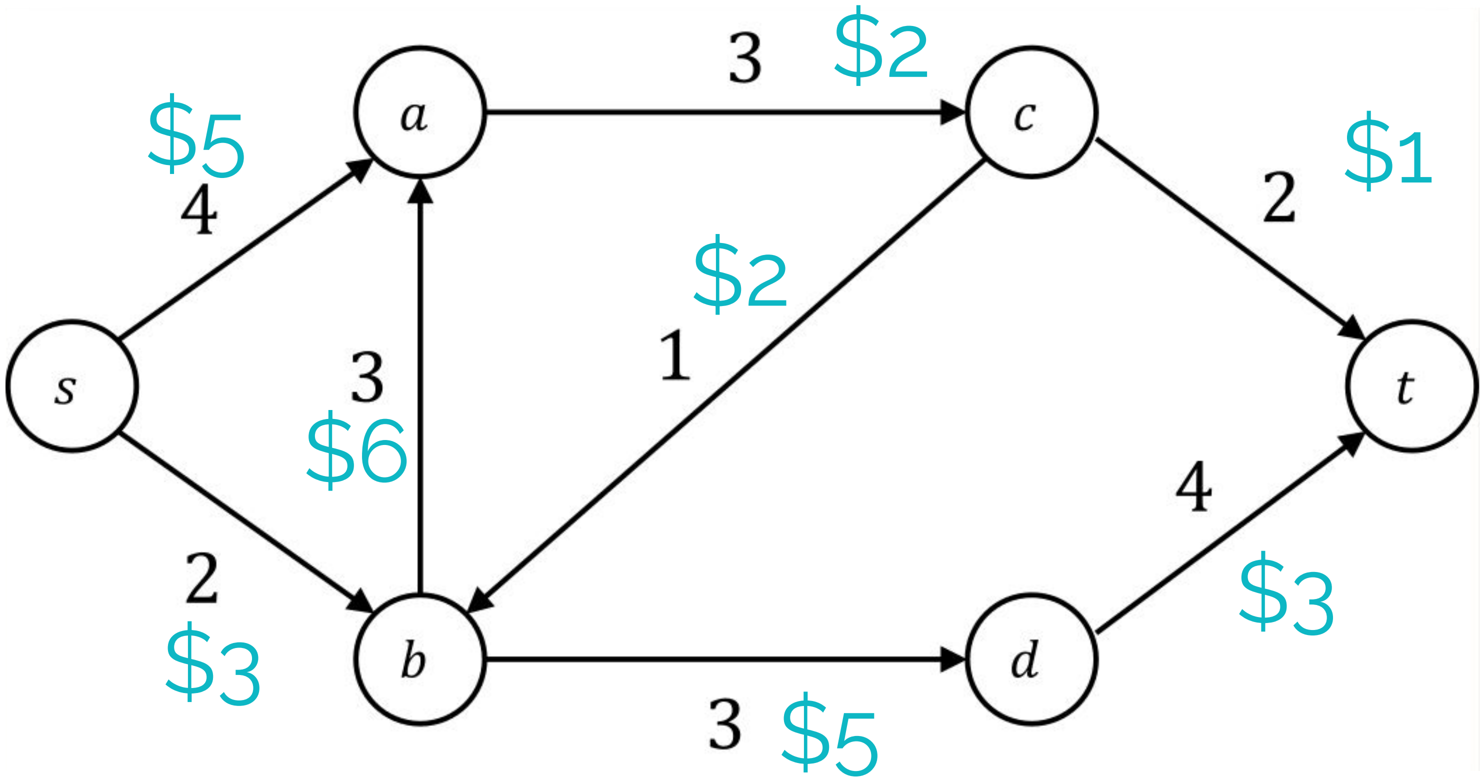
Elaine Shi

Min-cost flow



Each edge has a cost, represents **cost per unit flow**

Goal: find a flow of max value, minimizing cost



Each edge has a cost, represents **cost per unit flow**

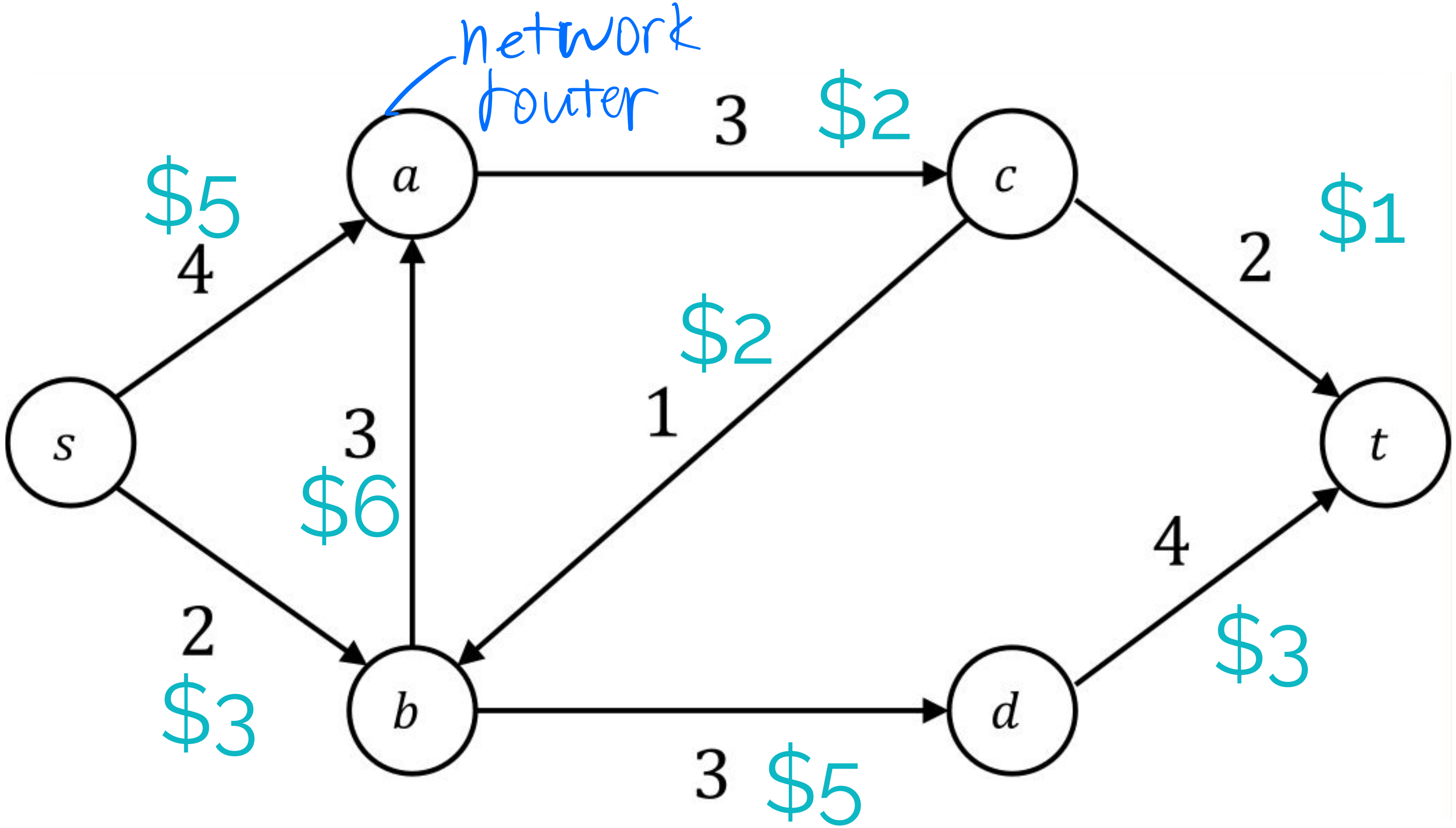
Cost of a flow

flow f

Cost per unit of flow on edge e

$$\mathrm{cost}(f) = \sum_{e \in E} \$(e) f(e).$$

Applications of min-cost flows



Routing flows on a network (used by Akamai in its early days)

Negative cost and negative cycles?

- Allow negative edges
- For now, assume no negative cycle in the original graph
  - Will be relaxed by the end of this lecture

Residual graph for min-cost flows

Residual capacity

$$
\begin{aligned}
c_f(e) &= c(e) - f(e), \\
c_f(\overleftarrow{e}) &= f(e)
\end{aligned}
$$

backedge

Residual graph for min-cost flows

Residual capacity

$$\begin{aligned} c_f(e) &= c(e) - f(e), \\ c_f(\overleftarrow{e}) &= f(e) \end{aligned}$$

Residual cost

$$\begin{aligned} \$_f(e) &= \$(e) \\ \$_f(\overleftarrow{e}) &= -\$(e) \end{aligned}$$

Residual graph for min-cost flows

Residual capacity

$$\begin{aligned} c_f(e) &= c(e) - f(e), \\ c_f(\overleftarrow{e}) &= f(e) \end{aligned}$$

Residual cost

$$\begin{aligned} \$_f(e) &= \$(e), \\ \$_f(\overleftarrow{e}) &= -\$(e) \end{aligned}$$

How do we find min-cost max flow?

Run FF but how should we select augmenting path?

How do we find min-cost max flow?

FF but select **cheapest** augmenting path w.r.t. residual costs

How do we find min-cost max flow?

FF but select **cheapest** augmenting path w.r.t. residual costs

$O(mn)$

How to find the cheapest augmenting path?

X Dijkstra

Bellman-Ford ?

Does this algorithm give a min-cost max flow ?

How do we find min-cost max flow?

FF but select **cheapest** augmenting path w.r.t. residual costs

How to find the cheapest augmenting path?

Dijkstra?

How do we find min-cost max flow?

FF but select **cheapest** augmenting path w.r.t. residual costs

How to find the cheapest augmenting path?

Dijkstra?

Negative edges!

How do we find min-cost max flow?

FF but select **cheapest** augmenting path w.r.t. residual costs

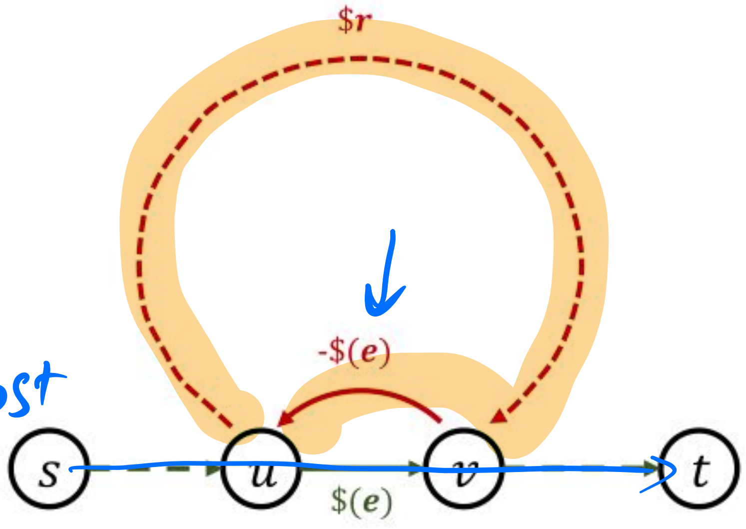
How to find the cheapest augmenting path?

Bellman-ford? … as long as no negative cycles

Theorem: original graph has no negative cost cycle $\Rightarrow$ in any iter, residual graph has no negative cost cycle

Theorem: original graph has no negative cost cycle ⇒ in any iter, residual graph has no negative cost cycle

Proof: simple case



Consider the first iter in which we see a negative cost cycle.

simple case: this negative cost cycle uses 1 back edge

$r - \$(e) < 0 \Rightarrow r < \$(e)$

$s \to u \to v \to t$ (blue path) cannot have been the cheapest path earlier

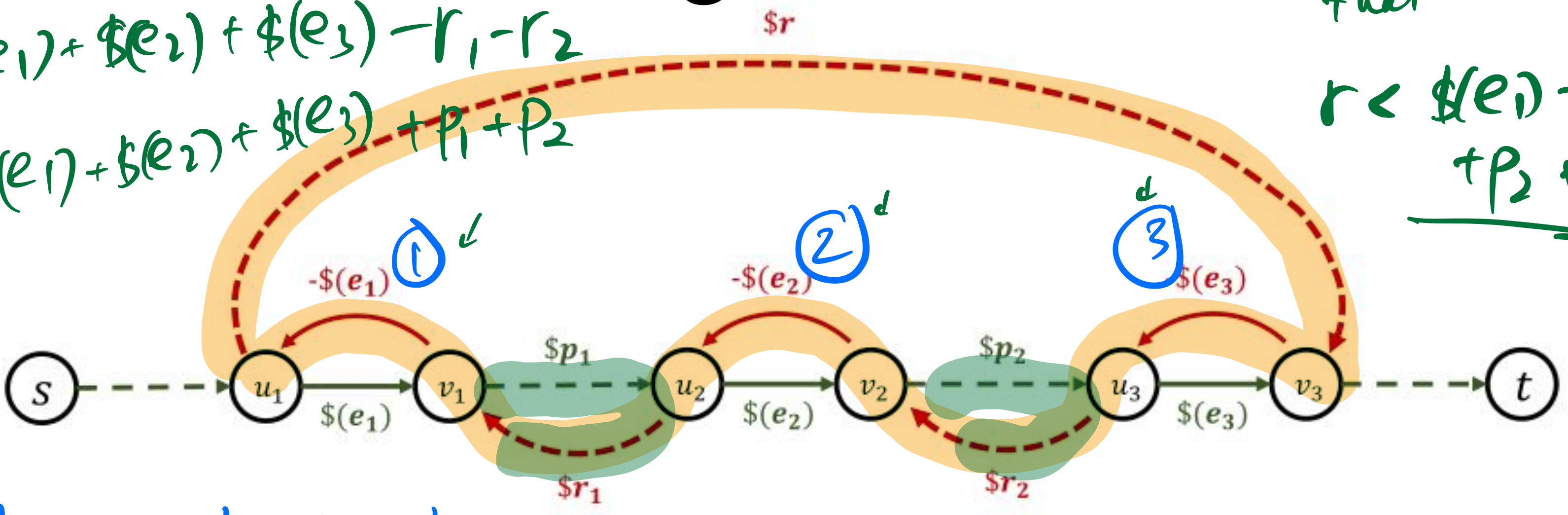
Proof (cont'd): more general case

It suffices to show that

$r < \$(e_1) + p_1 + \$(e_2) + p_2 + \$(e_3)$

$r < \$(e_1) + \$(e_2) + \$(e_3) - r_1 - r_2$

$\leq \$(e_1) + \$(e_2) + \$(e_3) + p_1 + p_2$



$r - \$(e_1) - \$(e_2) - \$(e_3) + r_1 + r_2 < 0$

$p_1 + r_1 \geq 0$

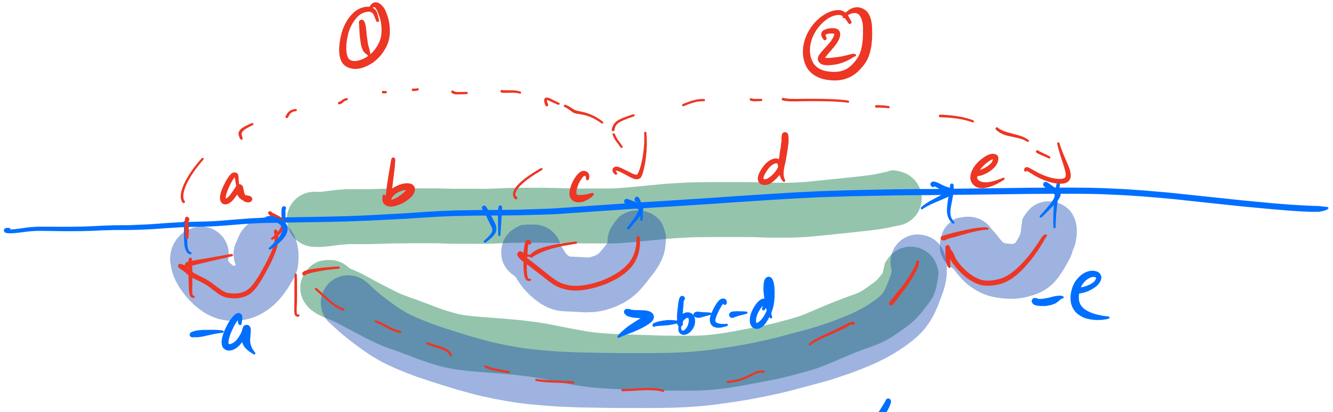
$p_2 + r_2 \geq 0$

$-r_1 \leq p_1$

$-r_2 \leq p_2$

Claim: $s \to u_1 \to v_1 \to u_2 \to v_2 \to u_3 \to v_3 \to t$ could have been the cheapest path earlier

in $s \to u_1 \longrightarrow v_3 \to t$ is cheaper



$① + ② < a + b + 2c + d + e$ ✓

either $① < a + b + c$

$② < c + d + e$.

Running time of the algorithm?

FF but select **cheapest** augmenting path w.r.t. residual costs

Bellman-Ford $O(mn)$

iters? $|F|$

total tim: $O(mn|F|)$

Running time of the algorithm?

FF but select **cheapest** augmenting path w.r.t. residual costs

FF but select **cheapest** augmenting path w.r.t. residual costs

How to find the cheapest augmenting path?

Does this algorithm give a min-cost max flow ?

"Cost optimality of flows"

A flow is cost optimal if it is the **cheapest** of all possible flows of the **same value**.

Theorem:

A flow f is cost optimal iff there is no negative-cost cycle in G_f residual graph for f

If this theorem holds, then our algorithm earlier indeed finds the min-cost max flow

- since we proved there is no negative-cost cycle in any iter

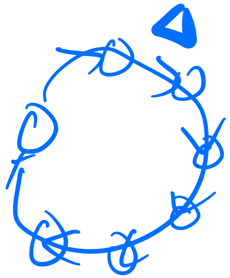
Theorem:

A flow f is cost optimal iff there is no negative-cost cycle in G_f

Proof:

1. Negative-cost cycle ⇒ not optimal (easy)
2. Not optimal ⇒ negative-cost cycle

1. Negative-cost cycle ⇒ not optimal (easy)



cycle C

Adding a cycle to the flow

- Does it affect flow conservation?
- Does it change the flow value = flow out of s − flow into s
- Can it change the cost?

↳ change in cost $= \Delta \cdot \sum_{e \in C} \$(e)$

if $\sum_{e \in C} \$(e) < 0$ then cost will drop

2. Not optimal ⇒ negative-cost cycle

Suppose f is not cost optimal, i.e., $\exists\ f'$ of the same value but cheaper cost

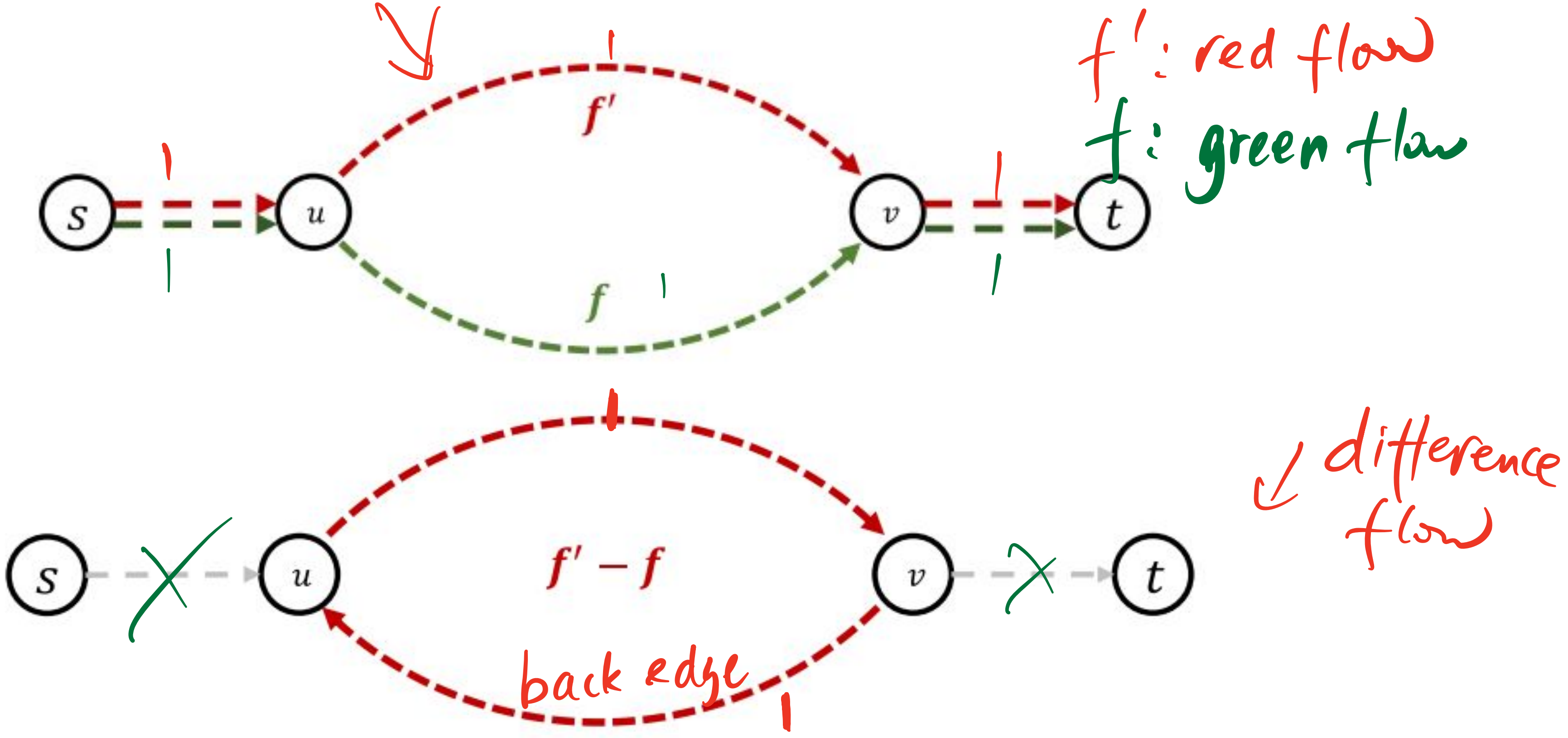
Consider the flow **$f'-f$**

- If $f'(u, v) - f(u, v) \geq 0$, then (u, v) has $f'(u, v) - f(u, v)$ flow
- If $f'(u, v) - f(u, v) < 0$, then (v, u) has $f(u, v) - f'(u, v)$ flow

$f'-f$ represents how much flow we need to augment to f in order to get f'

Consider the flow f'-f

- If $f'(u, v) - f(u, v) \geq 0$, then (u, v) has $f'(u, v) - f(u, v)$ flow
- If $f'(u, v) - f(u, v) < 0$, then (v, u) has $f(u, v) - f'(u, v)$ flow



Consider the flow f'-f

$\leq cap(u,v) - f(u,v)$

- If f'(u, v) - f(u, v) ≥ 0, then (u, v) has f'(u, v) - f(u, v) flow
- If f'(u, v) - f(u, v) < 0, then (v, u) has f(u, v) - f'(u, v) flow

$\leq f(u,v)$

Claim: f' - f is a valid flow of G_f ✓ residual graph of f

Consider the flow **f'-f**

- If $f'(u, v) - f(u, v) \geq 0$, then (u, v) has $f'(u, v) - f(u, v)$ flow
- If $f'(u, v) - f(u, v) < 0$, then (v, u) has $f(u, v) - f'(u, v)$ flow

Claim: $f' - f$ is a valid flow of G_f

$f' - f$ represents how much flow we need to augment to f in order to get f'

2. Not optimal ⇒ negative-cost cycle

Claim:

f'-f is a collection of flows on cycles (also called "circulation")

Proof:

For every vertex including s and t, "flow in" = "flow out" in f' - f

recall that f'-f is a valid flow in G_f

2. Not optimal ⇒ negative-cost cycle

Claim:

f'-f is a collection of flows on cycles (also called circulation)

Since cost(f' - f) = cost(f') - cost(f) < 0

At least one cycle in f'-f has negative cost

Summary:

- cheapest augmenting path algorithm augments the flow while maintaining cost optimality
- running time? $O(\underbrace{n \cdot m}\cdot\underbrace{|F|})$

Cycle-canceling algorithm

(another algorithm for min-cost max flow)

Recall

A flow f is cost optimal iff there is no negative-cost cycle in G_f

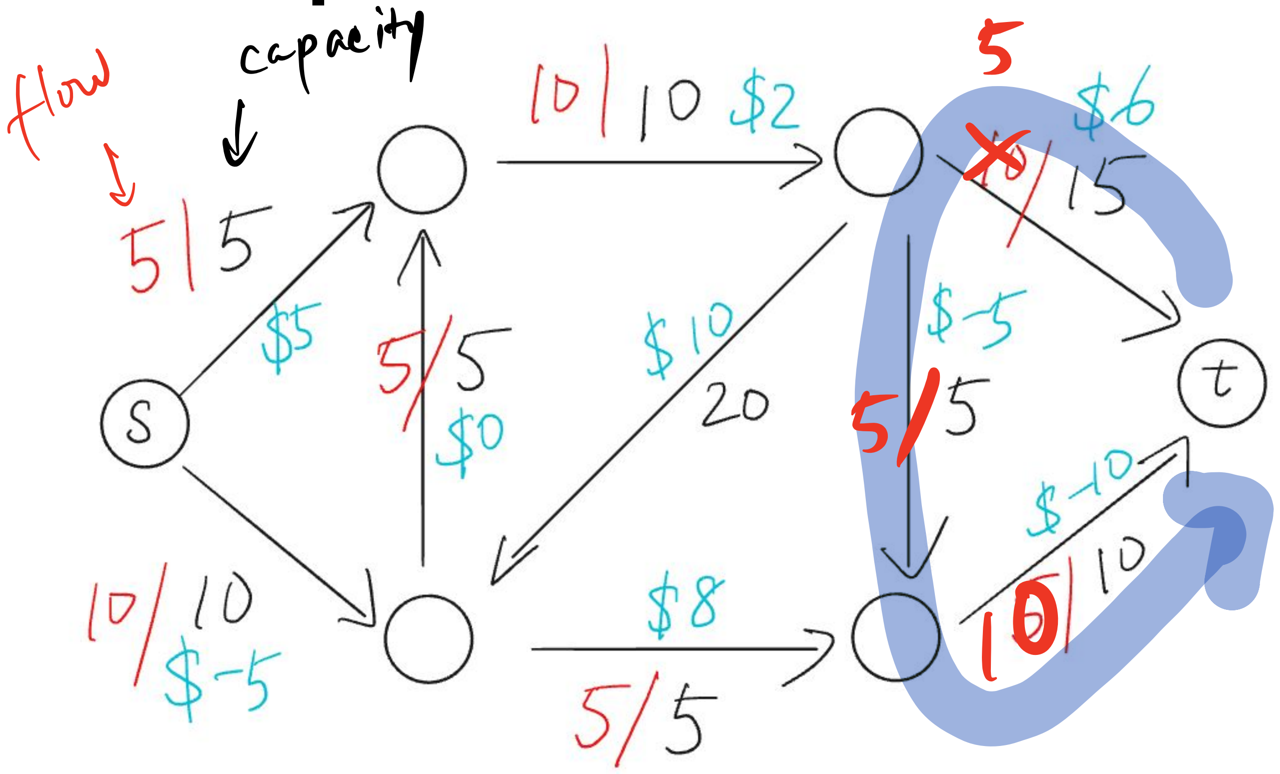
Cycle-canceling algorithm for min-cost flow

Find a max flow f (e.g. use FF, E-K, Dinic)

While ∃ negative-cost cycle in G_f:

    Augment the max possible amt of flow along the cycle

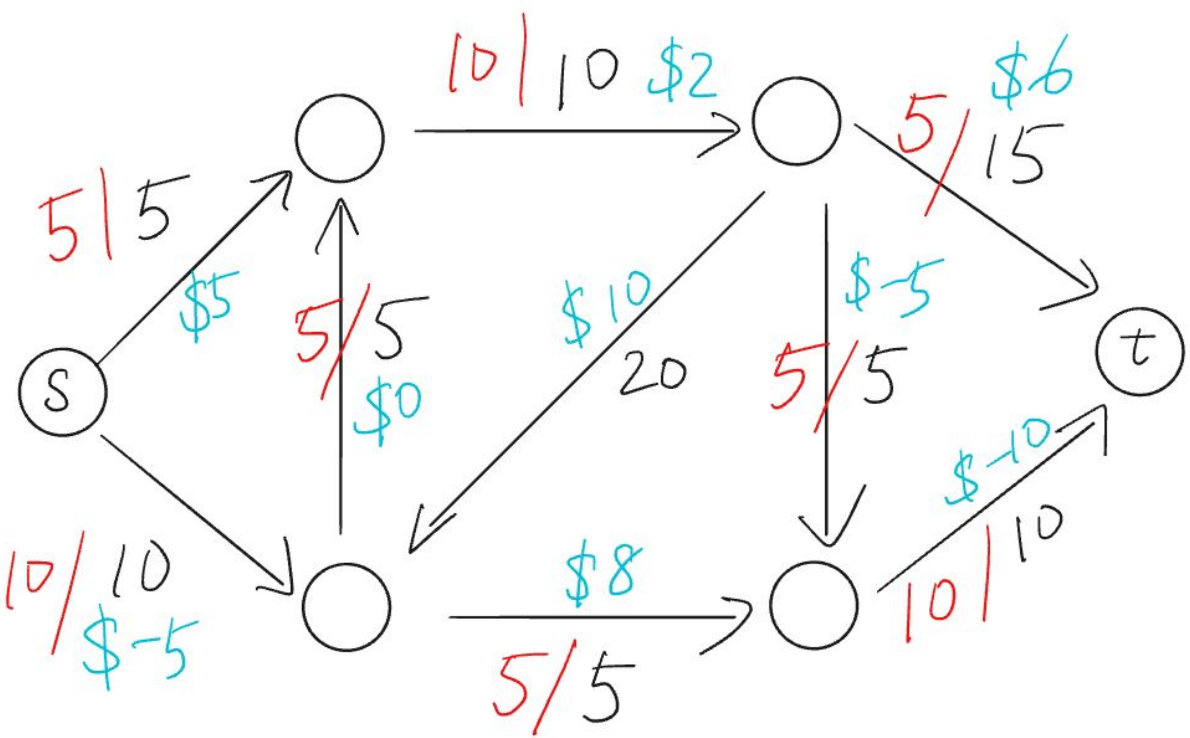
Example:



flow
capacity
5/5
$5
10/10 $2
5/5
$0
$10
20
5
~~10~~/15
$6
$-5
5/5
t
S
$-10
~~5~~ 10/10
10/10
$-5
$8
5/5

max flow

Example (cont'd) : improved cost



Cycle-canceling algorithm for min-cost flow

Find a max flow f

While $\exists$ negative-cost cycle in G_f:

Augment the max possible amt of flow along the cycle

How can we find negative cycles?

Bellman-Ford

Quick recap of Bellman-Ford

DP

in each iteration i,

find s for each v

$d(v, i)$

shortest distance from s to v traversing at most i vertices

a modification of Bellman-Ford lets you retrieve a negative cycle in $O(m \cdot n)$

Cycle-canceling algorithm for min-cost flow

Find a max flow f

While ∃ negative-cost cycle in G_f: $O(mn)$

Augment the max possible amt of flow along the cycle

Works even if input graph has negative-cost cycle

Running time of cycle canceling algorithm $O(nm^2UC)$

and costs

Assume: all capacities are integers, and edge capacities are at most U, and costs are between -C and C

How many iters does the cycle-cancelling alg take to complete?

Initially, I start with some max flow whose total cost $\leq m \cdot U \cdot C$

every iter improves the cost by at least 1

in the end, the min-cost $\geq -mUC$. the total decrease in cost $\leq 2mUC$

Running time of cycle canceling algorithm

Assume: all capacities are integers, and edge capacities are at most U, and costs are between -C and C

Starting from some max flow, each iteration improves the cost by at most 1

How much can we lower the cost starting from the initial max flow?

Possible to improve the running time by choosing the negative-cost cycle more cleverly

Goldberg-Tarjan

min mean cost

Max flow

Min-cost max flow

(After spring break)

Linear programming