CS4102 Algorithms

Spring 2021 – Floryan and Horton

Module 4, Day 1: Horton’s Live Session

Network Flow, Ford-Fulkerson

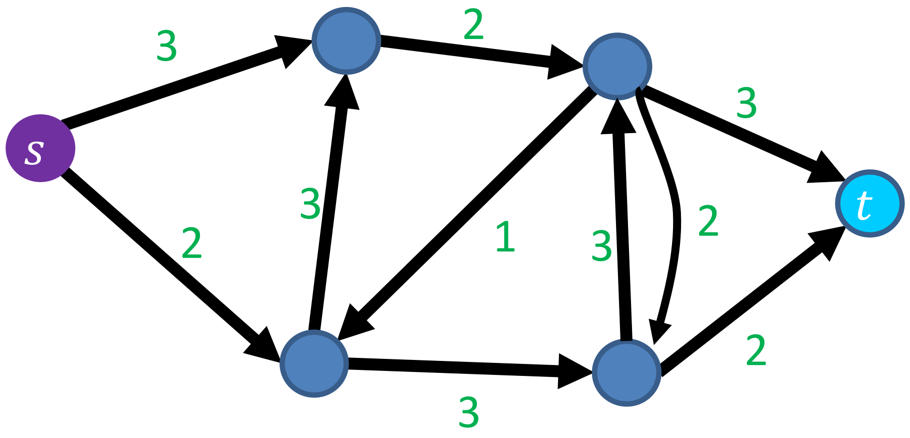
Announcements, Mon., April 19

- Quiz 3 (1^{st} attempt) and Quiz 2 (2^{nd} attempt) will be end of this week
  - Same as before: GradeScope, Friday noon – Sat. 6pm
- Module 3 soft deadline grading is underway, should be done before quiz (that's our goal)
- Recommended schedule for HWs
  - Working on one Module 3 Advanced HW this week.
  - Start Module 4 HWs next week
- Module 4 starts today. Hooray! (?)
  - End of semester makes it a bit condensed. We'll do our best to make it easier on you.
- Today: flow networks, Ford-Fulkerson. Do you understand? Example!

In your textbook

- CLRS 26.1 and 26.2
- Includes simple solutions to the following "complications"
  - What if (u,v) and (v,u) are in the flow graph?
  - What if we need >1 source? >1 sink?
- Note:
  - Recorded lecture emphasized adjacency matrix for storing ***residual network G_f*** for a given ***flow-graph G***
    - Matrix stores both capacities and back-flow values for (u,v)
  - Today we'll just draw graphs
  - We won't draw edges in G_f with value 0 (either capacity or back-flow)

Flow Network



Graph $G = (V, E)$

Source node $s \in V$

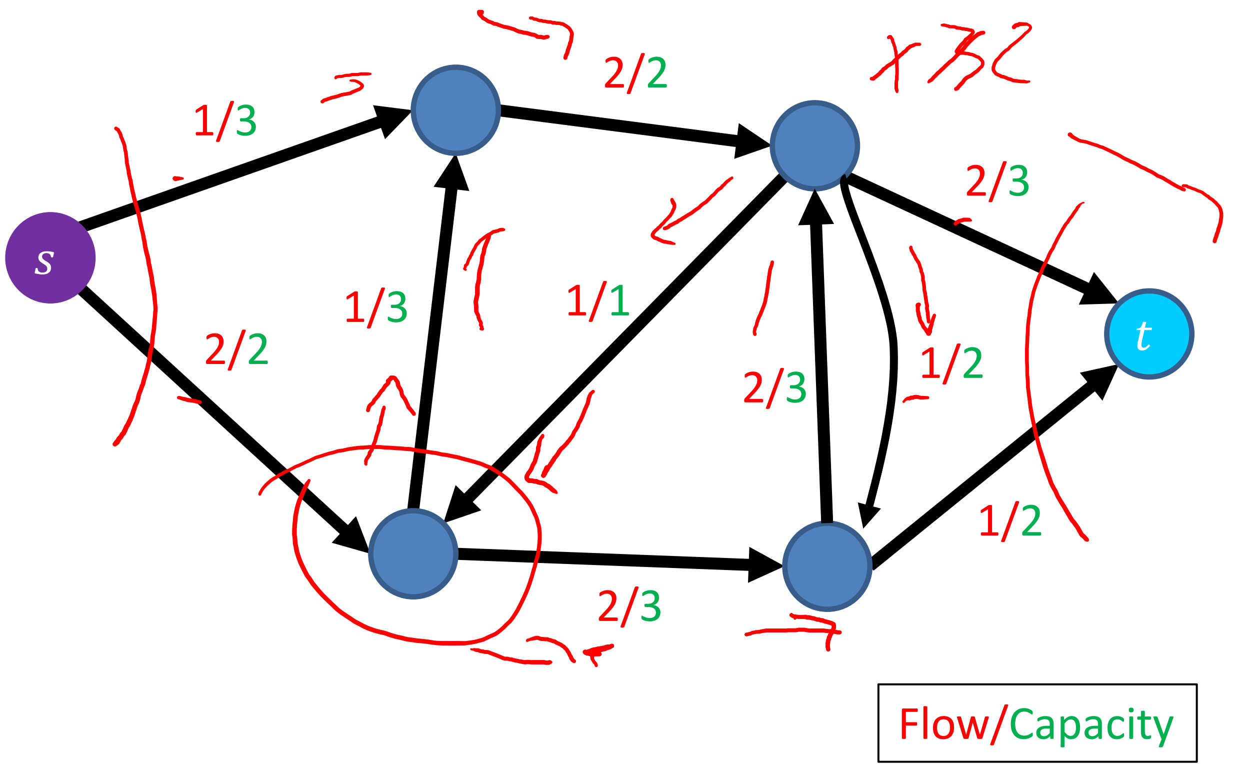
Sink node $t \in V$

Edge Capacities $c(e) \in$ Positive Real numbers

Max flow intuition: If s is a faucet, t is a drain, and s connects to t through a network of pipes with given capacities, what is the maximum amount of water which can flow from the faucet to the drain?

Flow

- Assignment of values to edges
  - $f(e) = n$
  - Amount of water going through that pipe
- Capacity constraint
  - $f(e) \leq c(e)$
  - Flow cannot exceed capacity
- Flow constraint
  - $\forall v \in V - \{s, t\}, inflow(v) = outflow(v)$
  - $inflow(v) = \sum_{x \in V} f(v, x)$
  - $outflow(v) = \sum_{x \in V} f(x, v)$
  - Water going in must match water coming out
- Flow of G: $|f| = outflow(s) - inflow(s)$
  - Net outflow of s



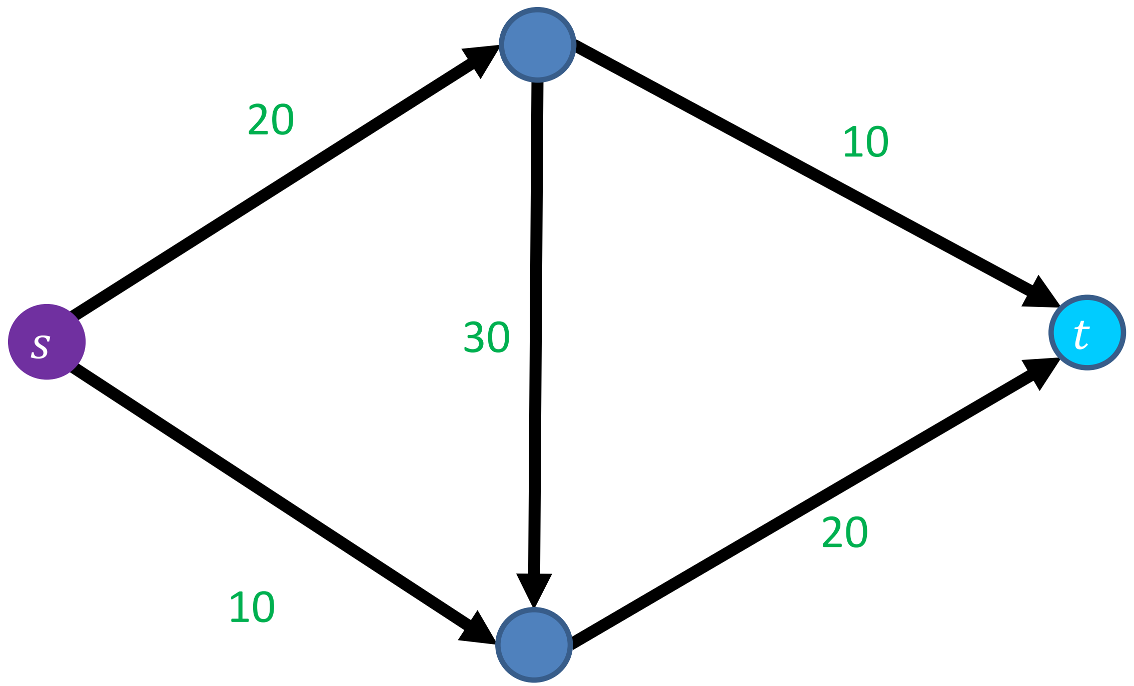
3 in example above

Max Flow

- Of all valid flows through the graph, find the one which maximizes:
  - $|f| = outflow(s) - inflow(s)$

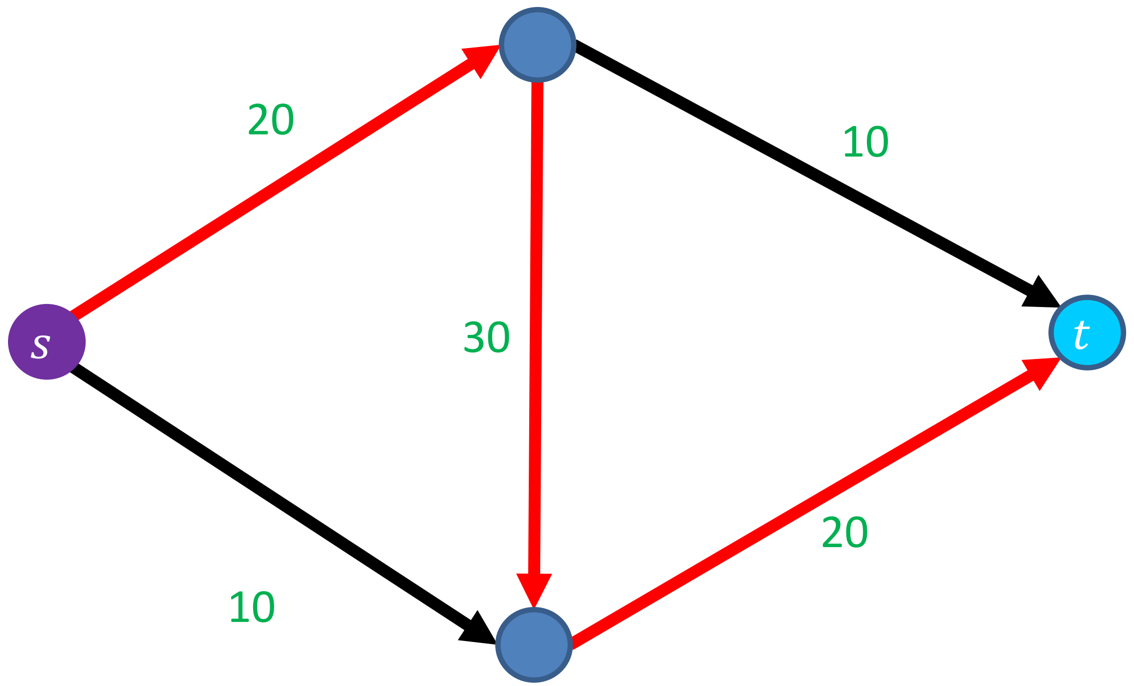
Greedy doesn't work

Saturate Highest Capacity Path First



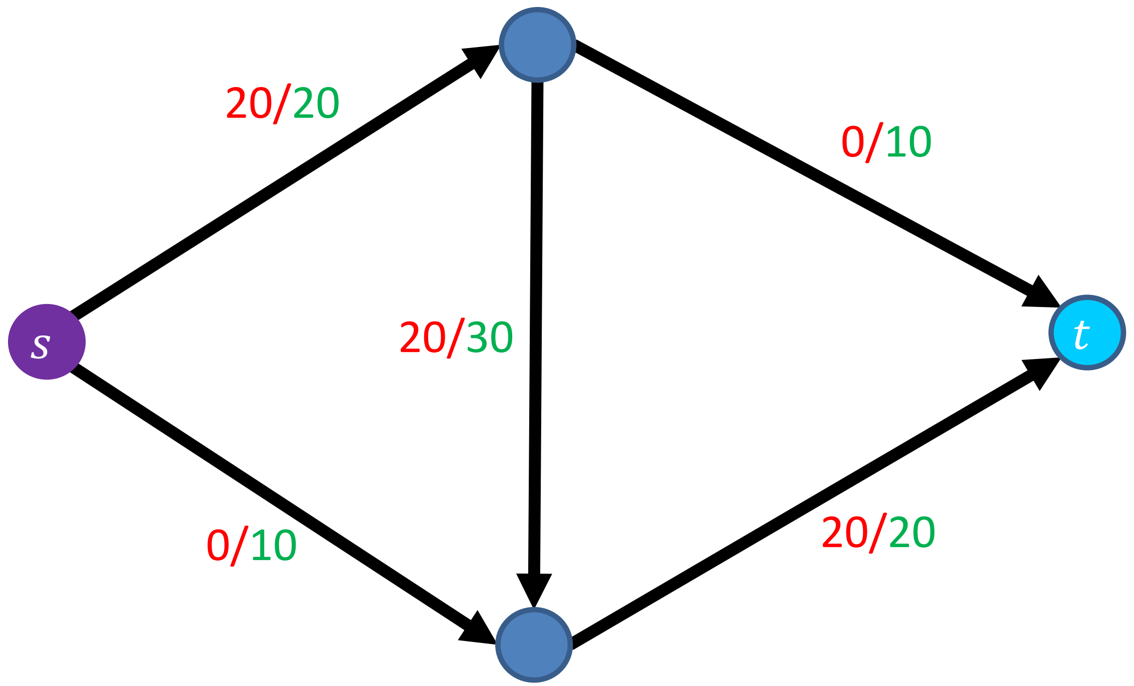
Greedy doesn't work

Saturate Highest Capacity Path First



Greedy doesn't work

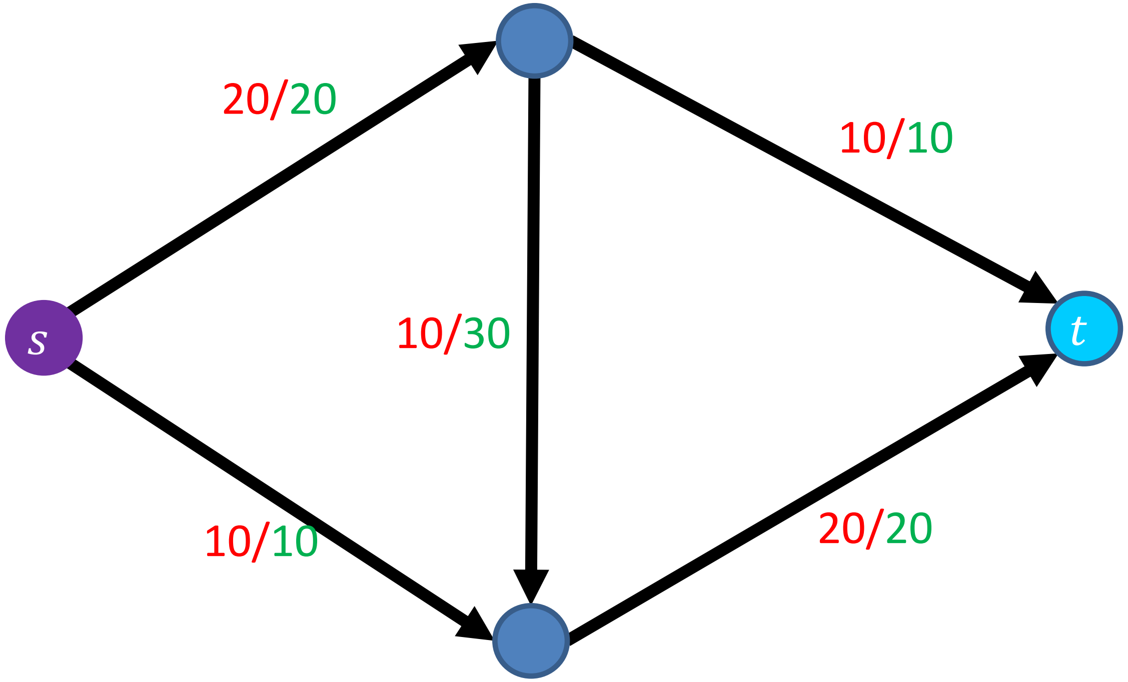
Saturate Highest Capacity Path First



Overall Flow: $|f| = 20$

Greedy doesn't work

Better Solution



Overall Flow: $|f| = 30$

Residual Graph G_f

- Keep track of net available flow along each edge
- "Forward edges": weight is equal to available flow along that edge in the flow graph
  - $w(e) = c(e) - f(e)$

Flow I *could* add

- "Back edges": weight is equal to flow along that edge in the flow graph
  - $w(e) = f(e)$

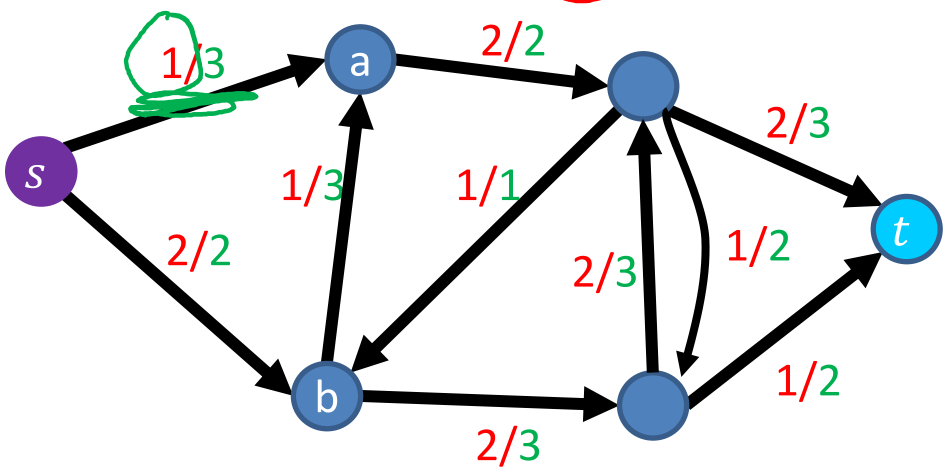
$10 \to 20$

$10 \to$

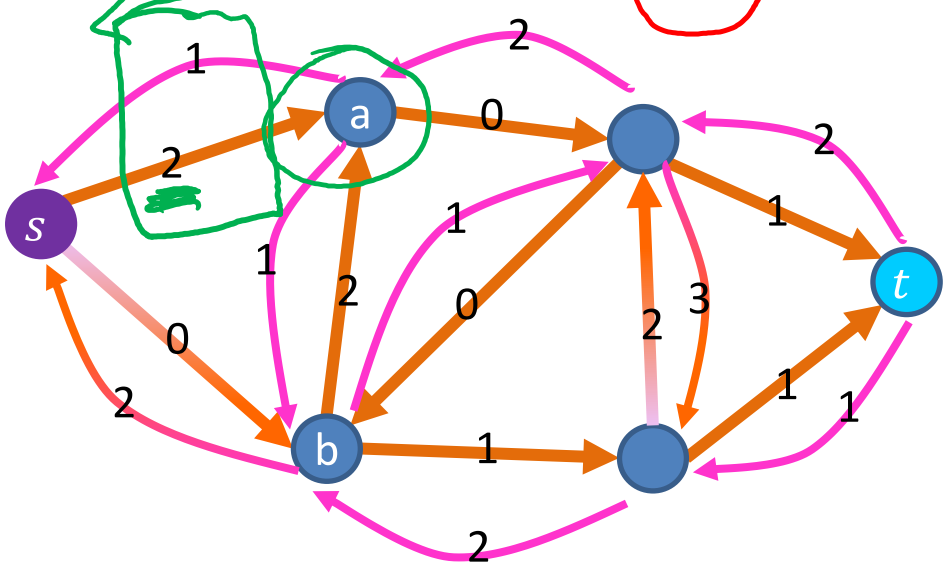
$\leftarrow 10$

Flow I *could* remove

Flow Graph G



Residual Graph G_f

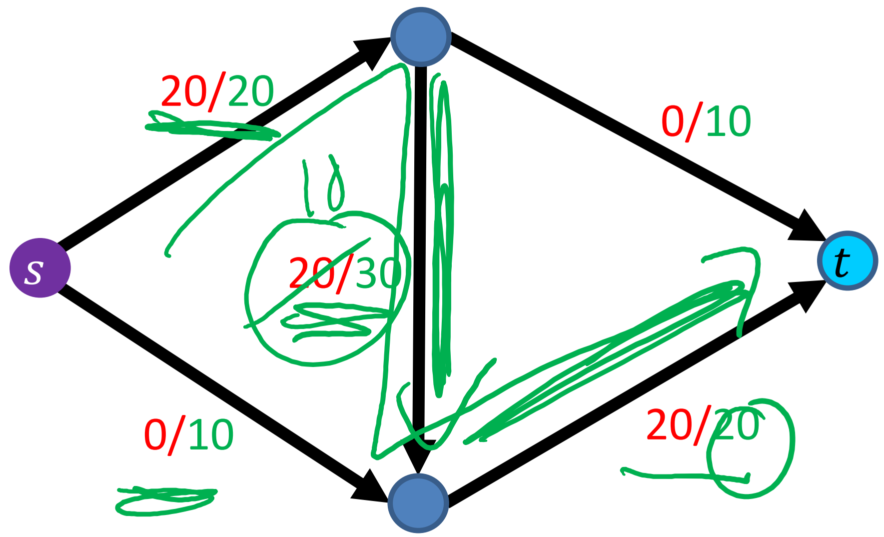


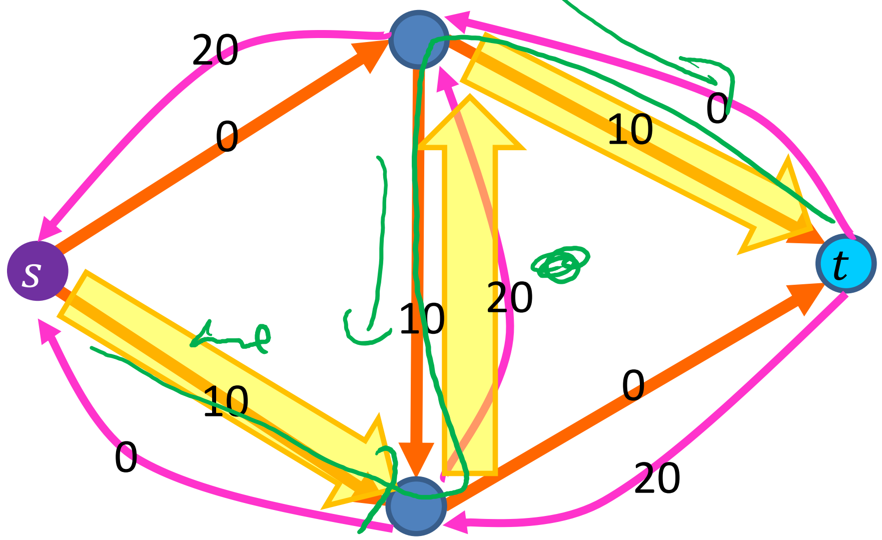
Residual Graphs Example

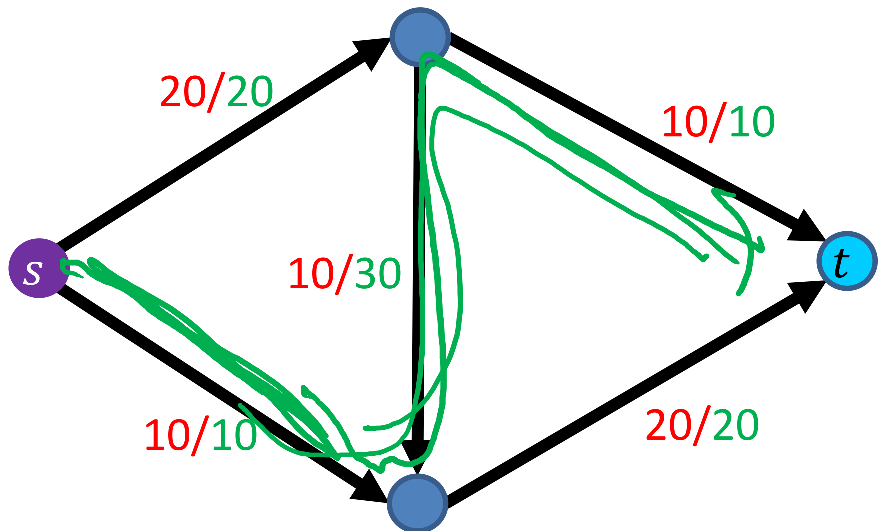
DFS

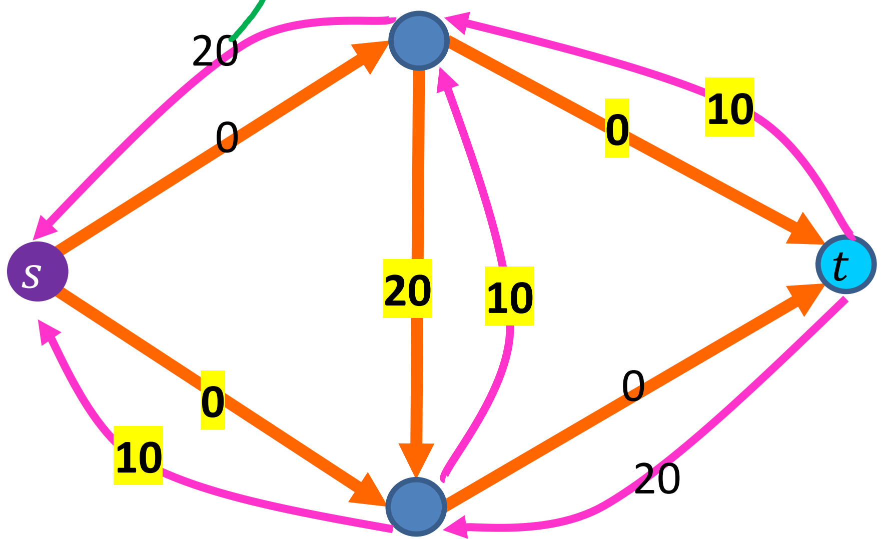
Flow Graph

Residual Graph







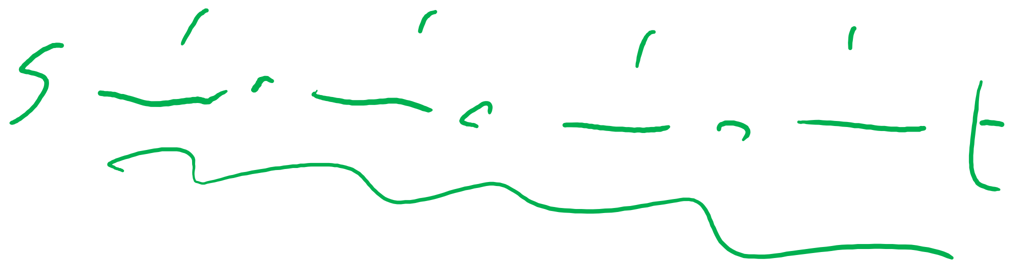


Ford-Fulkerson Algorithm

Define an **augmenting path** to be a path from $s \to t$ in the residual graph G_f (using edges of non-zero weight)

Overview: Repeatedly add the flow of any augmenting path

Ford-Fulkerson max-flow algorithm:



- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Ford-Fulkerson Algorithm

Define an **augmenting path** to be a path from $s \to t$ in the residual graph G_f (using edges of non-zero weight)

Overview: Repeatedly add the flow of any augmenting path

Ford-Fulkerson max-flow algorithm:

- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Ford-Fulkerson approach: take
<u>any</u> augmenting path
(will revisit this later)

Ford-Fulkerson Algorithm

Define an **augmenting path** to be a path from $s \to t$ in the residual graph G_f (using edges of non-zero weight)

Overview: Repeatedly add the flow of any augmenting path

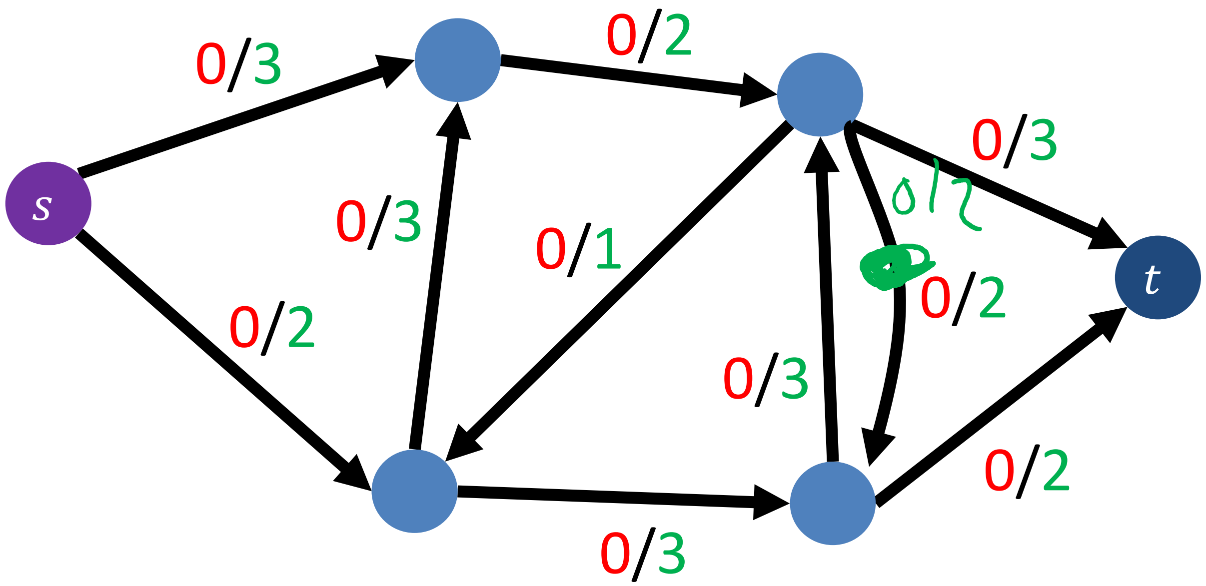
Ford-Fulkerson max-flow algorithm:

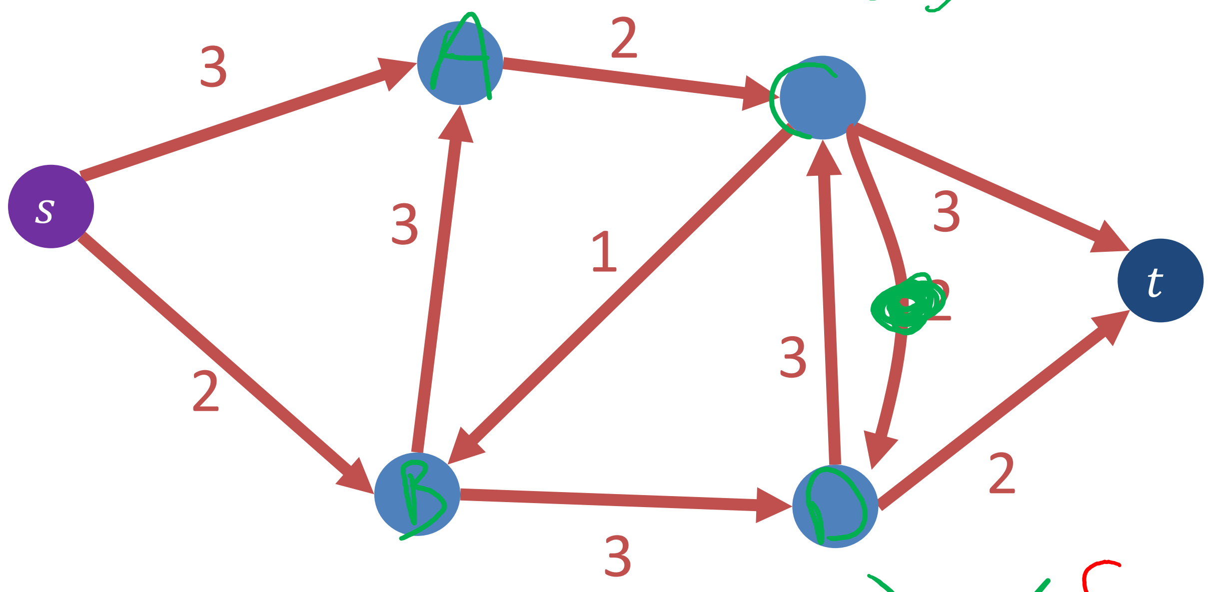
- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Ford-Fulkerson approach: take <u>any</u> augmenting path (will revisit this later)

($c_f(u, v)$ is the weight of edge (u, v) in the residual network G_f)

Ford-Fulkerson Example

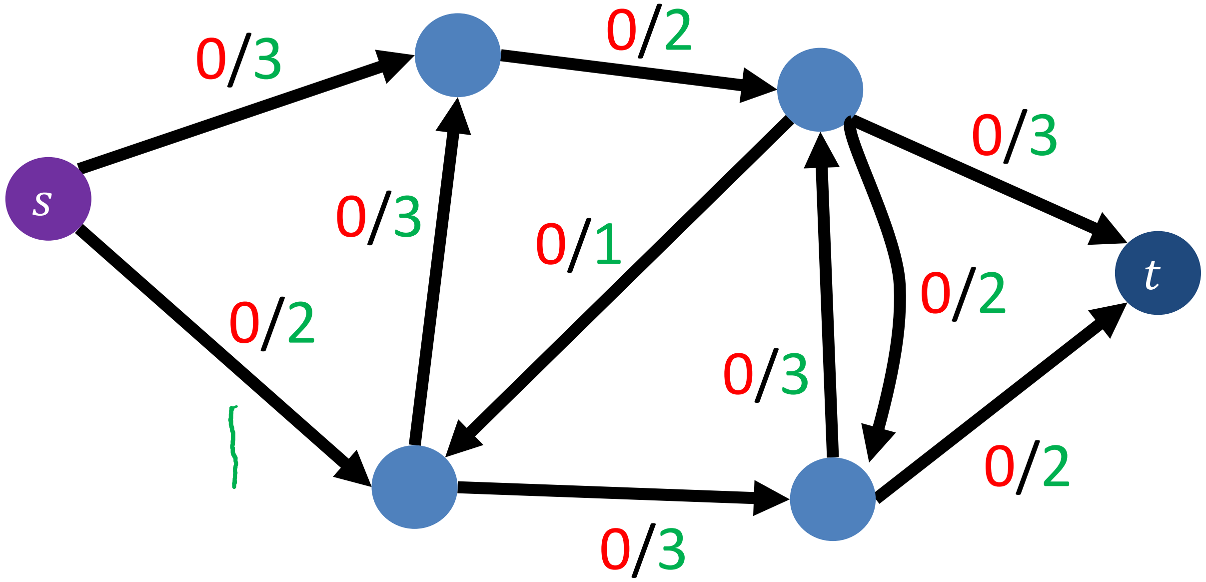




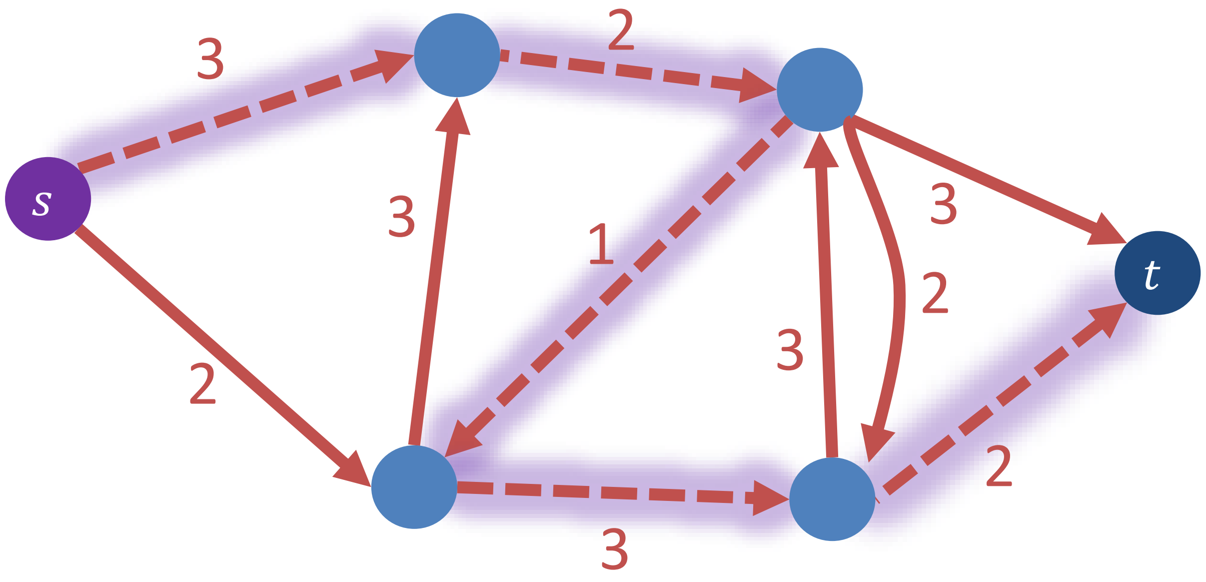
Initially: $f(e) = 0$ for all $e \in E$

Residual graph G_f

Ford-Fulkerson Example

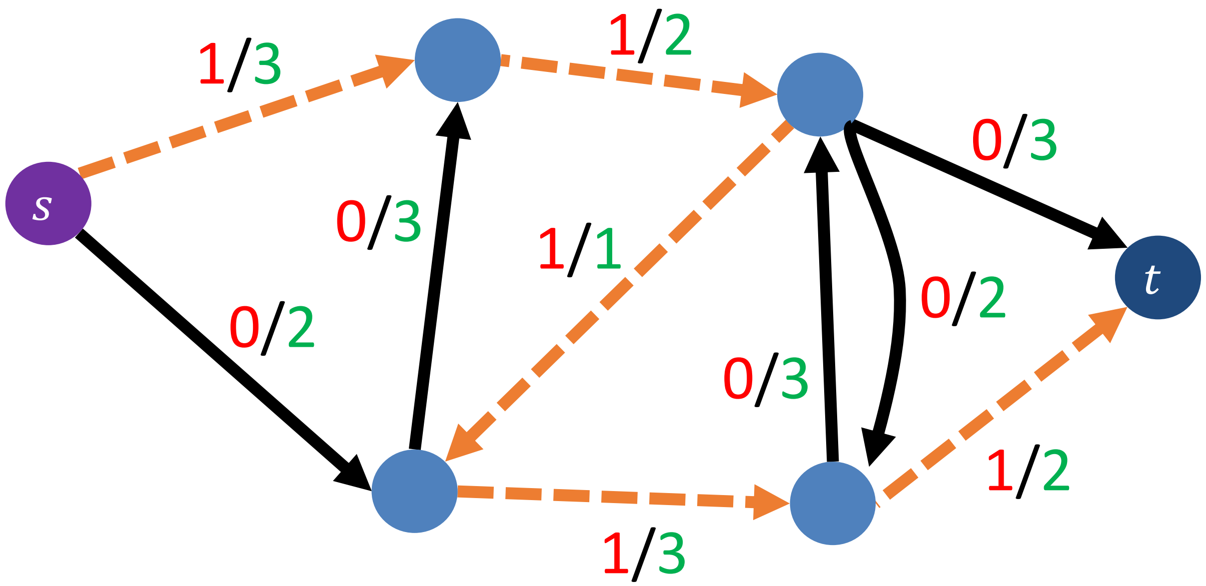


Increase flow by 1 unit

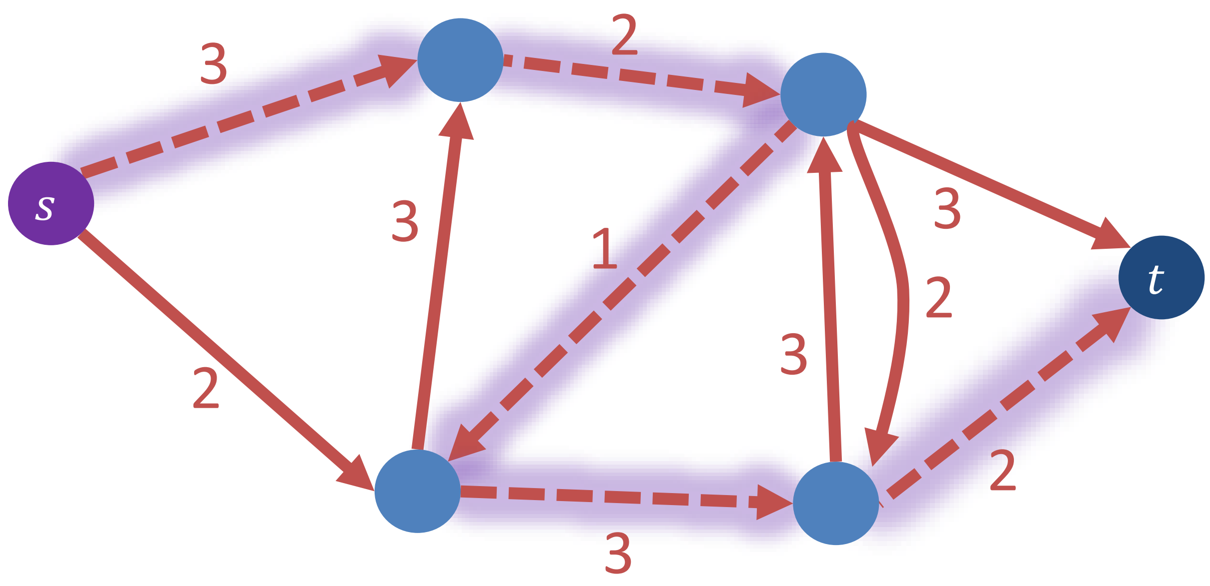


Residual graph G_f

Ford-Fulkerson Example

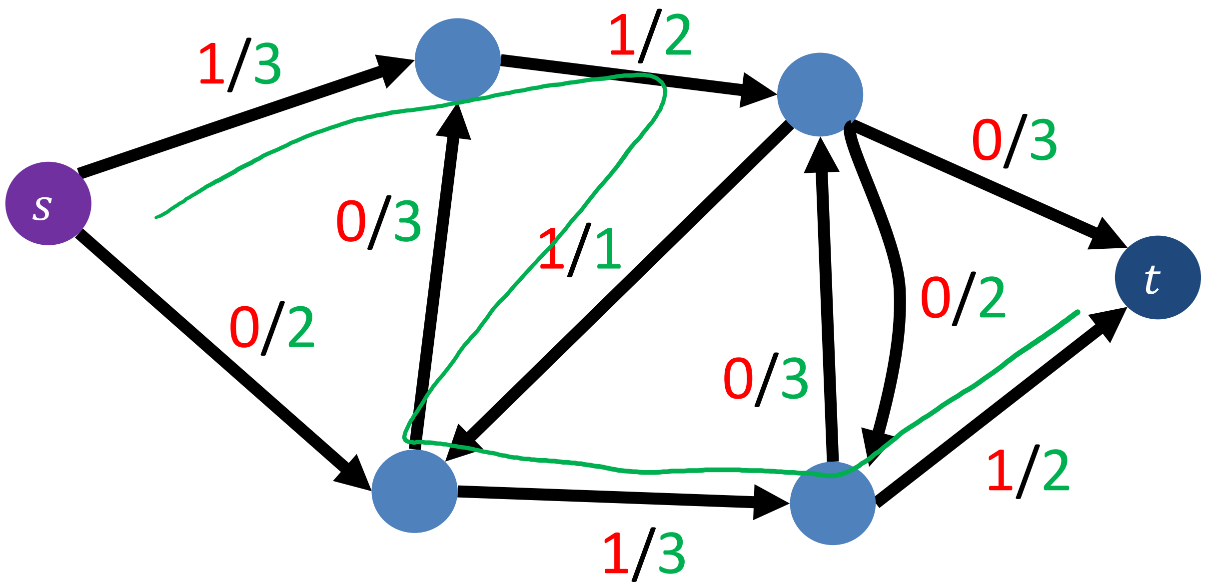


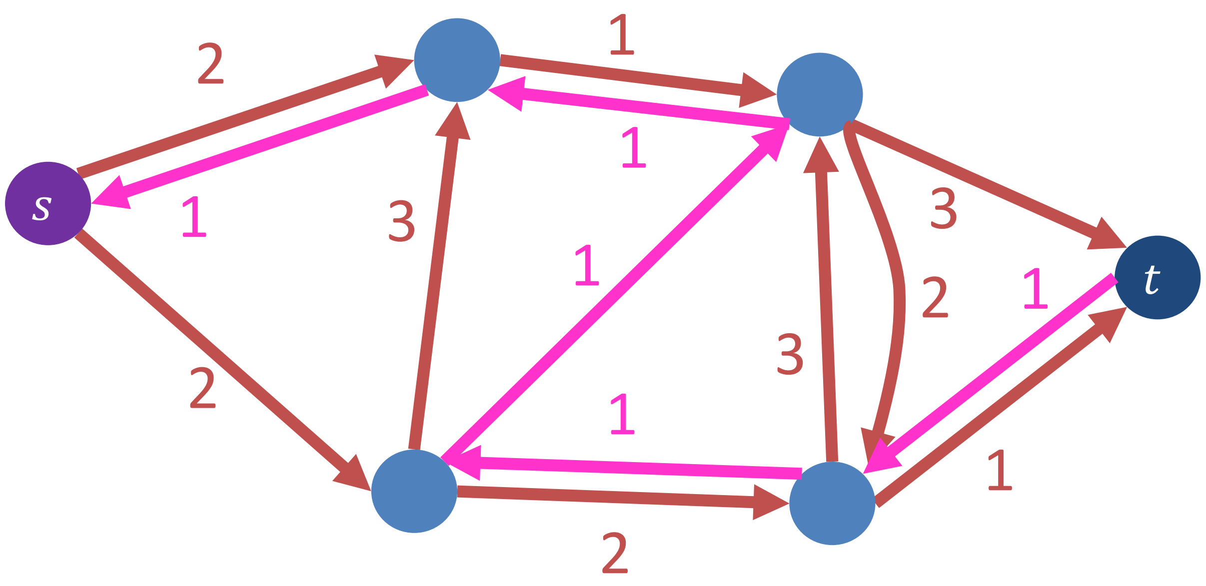
Increase flow by 1 unit



Residual graph G_f

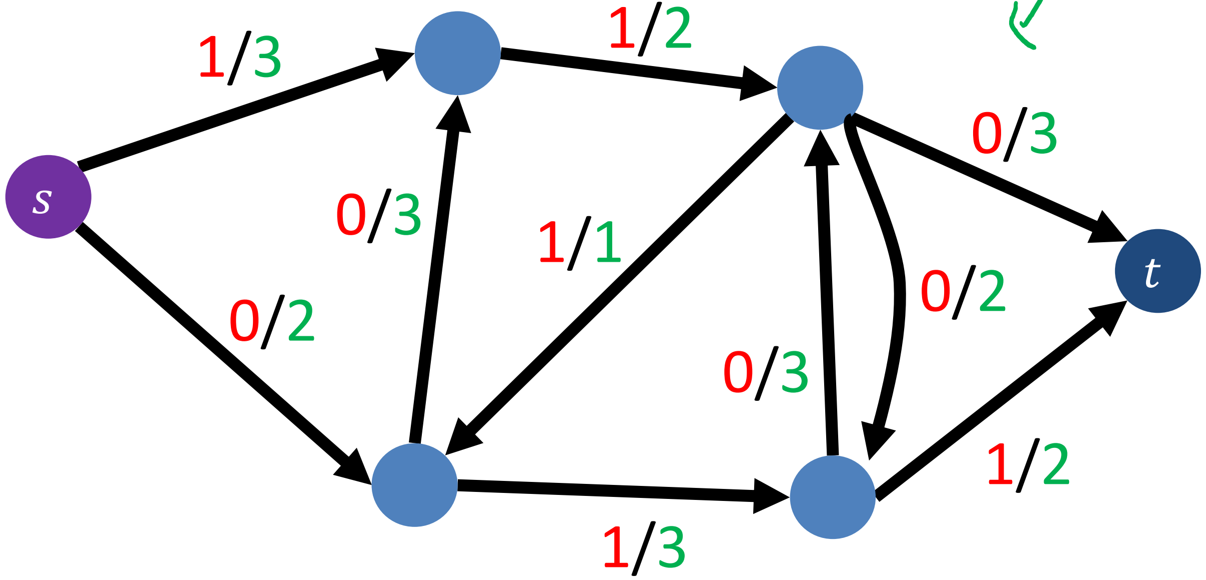
Ford-Fulkerson Example



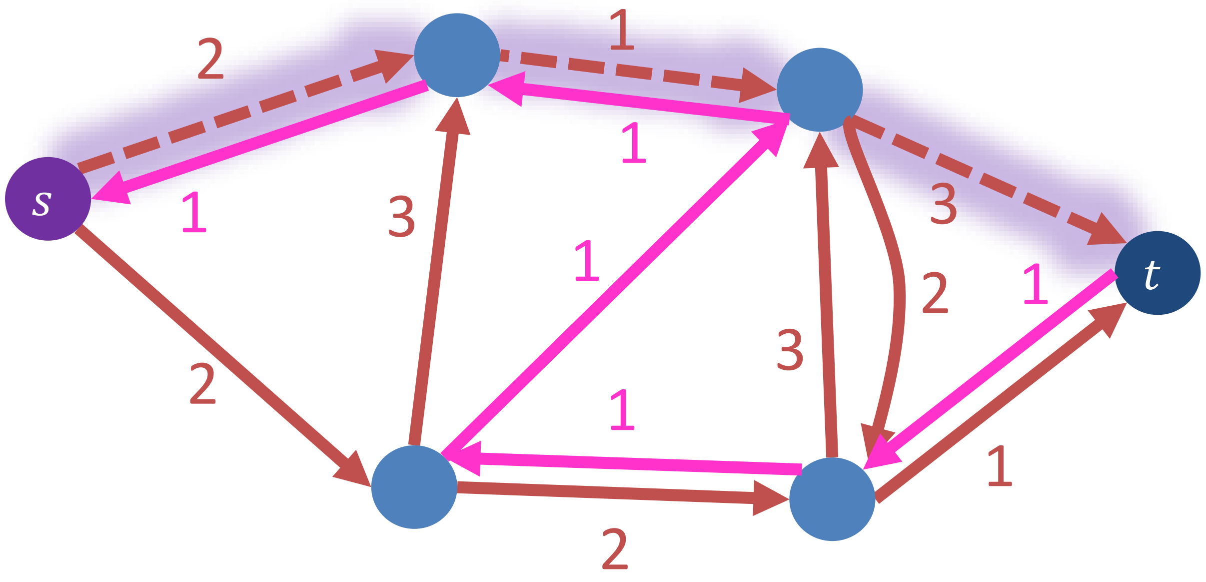


Residual graph G_f

Ford-Fulkerson Example

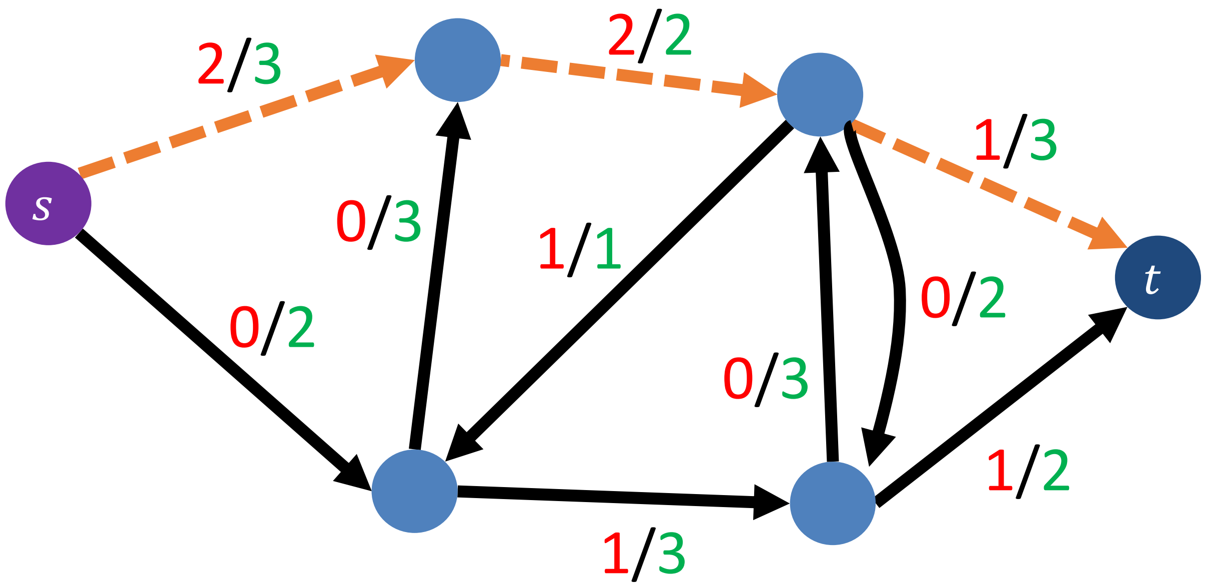


Increase flow by 1 unit

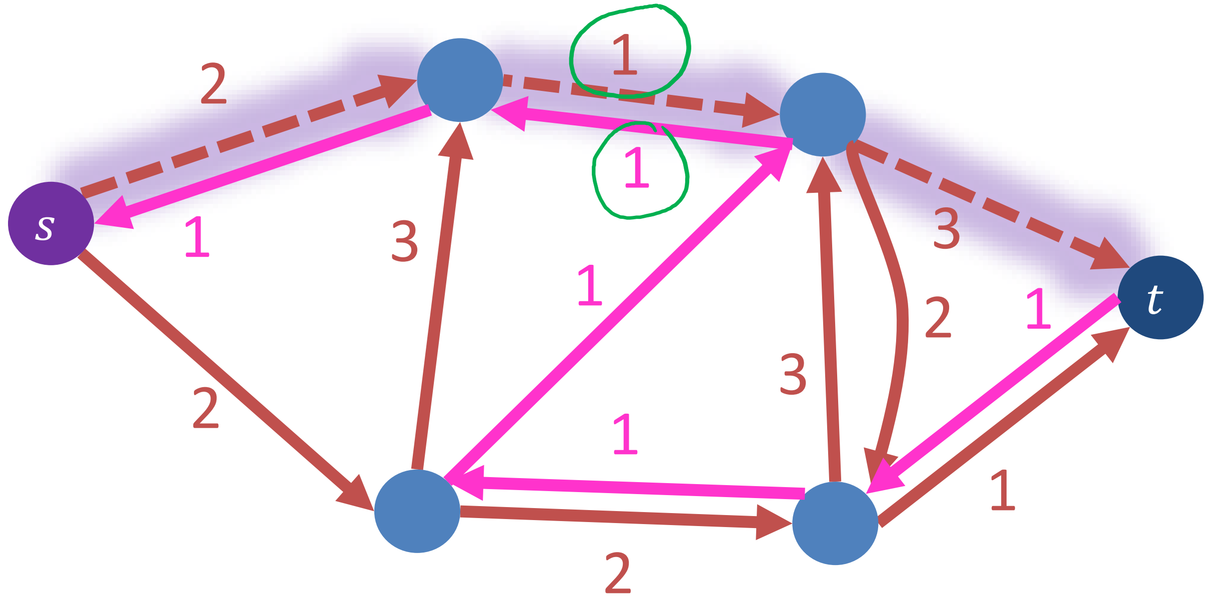


Residual graph G_f

Ford-Fulkerson Example

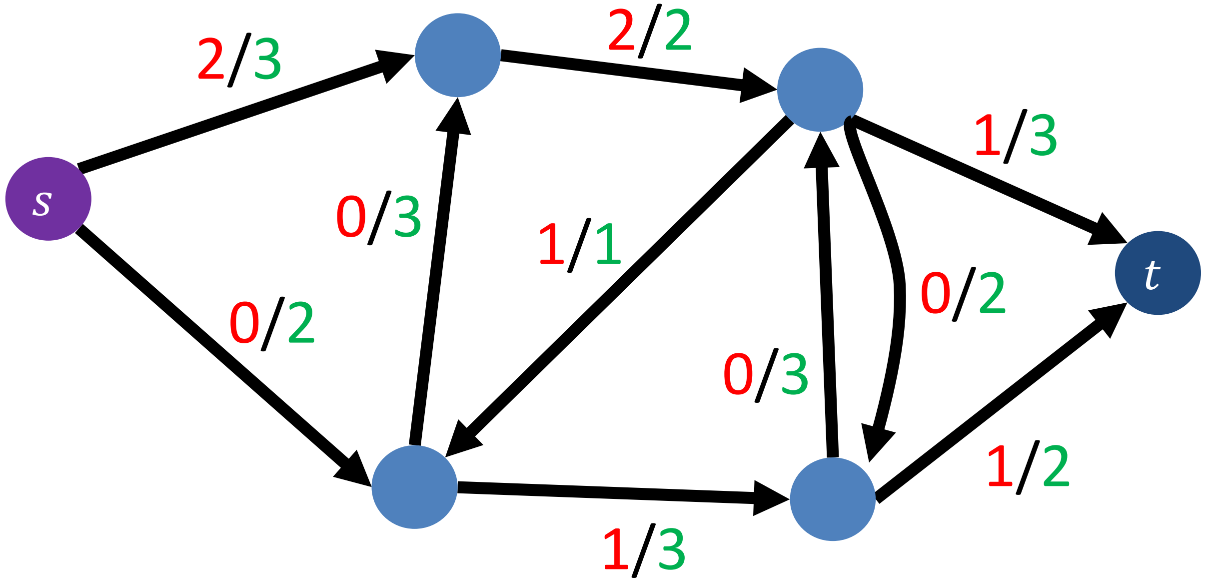


Increase flow by 1 unit

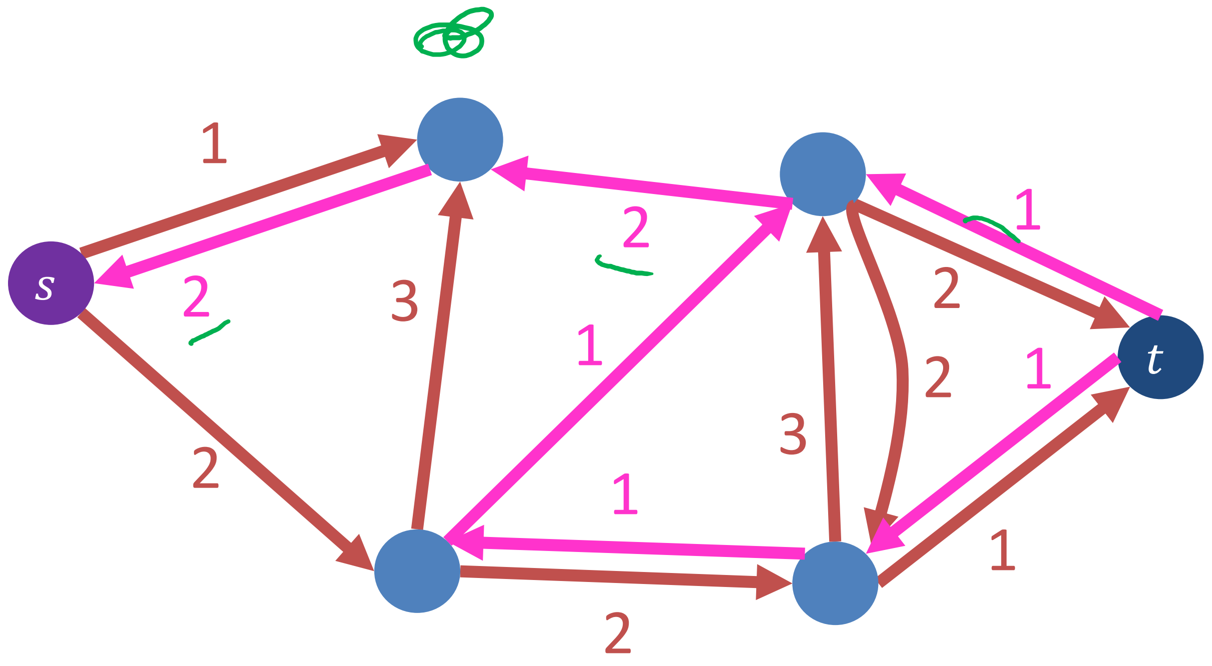


Residual graph G_f

Ford-Fulkerson Example

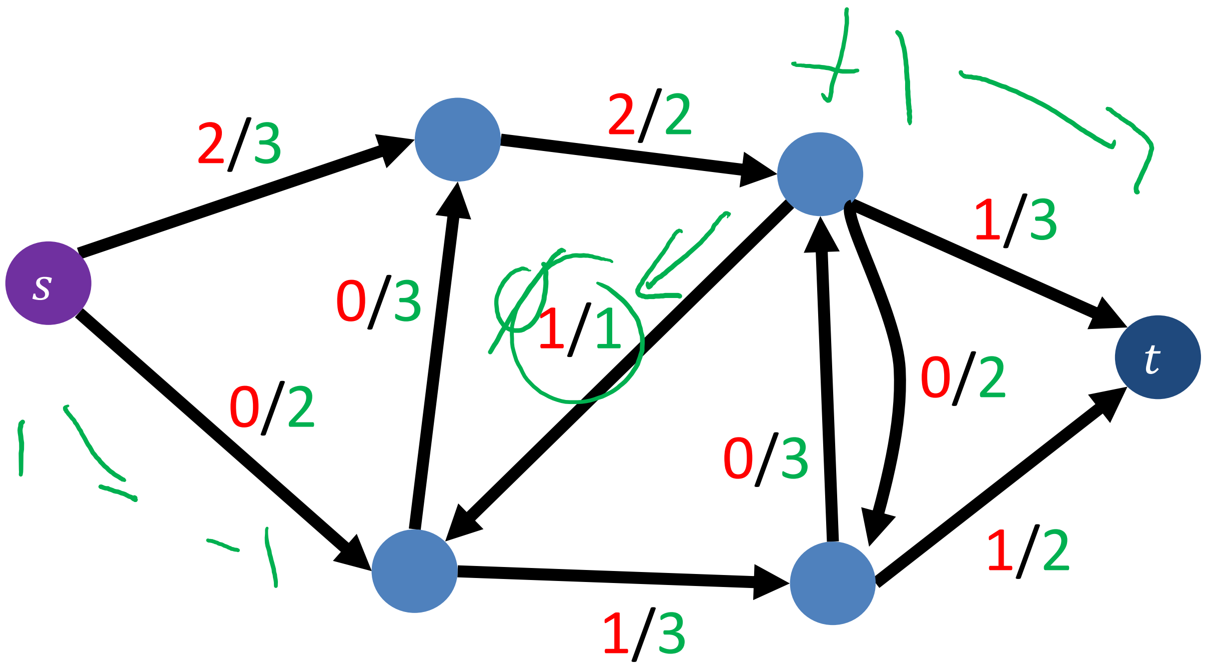


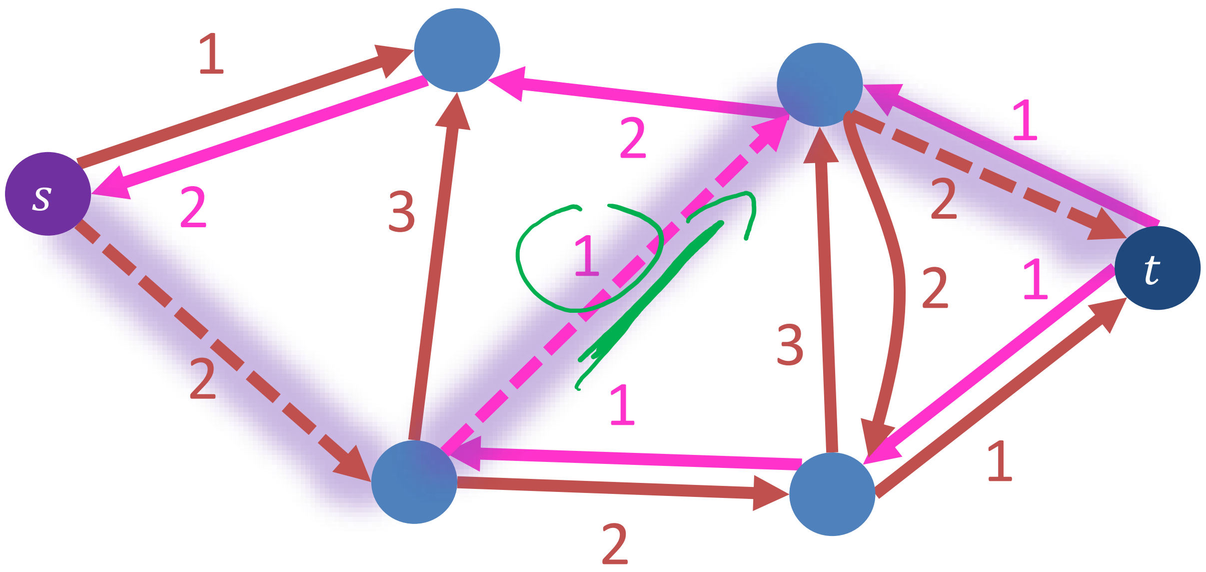
Increase flow by 1 unit



Residual graph G_f

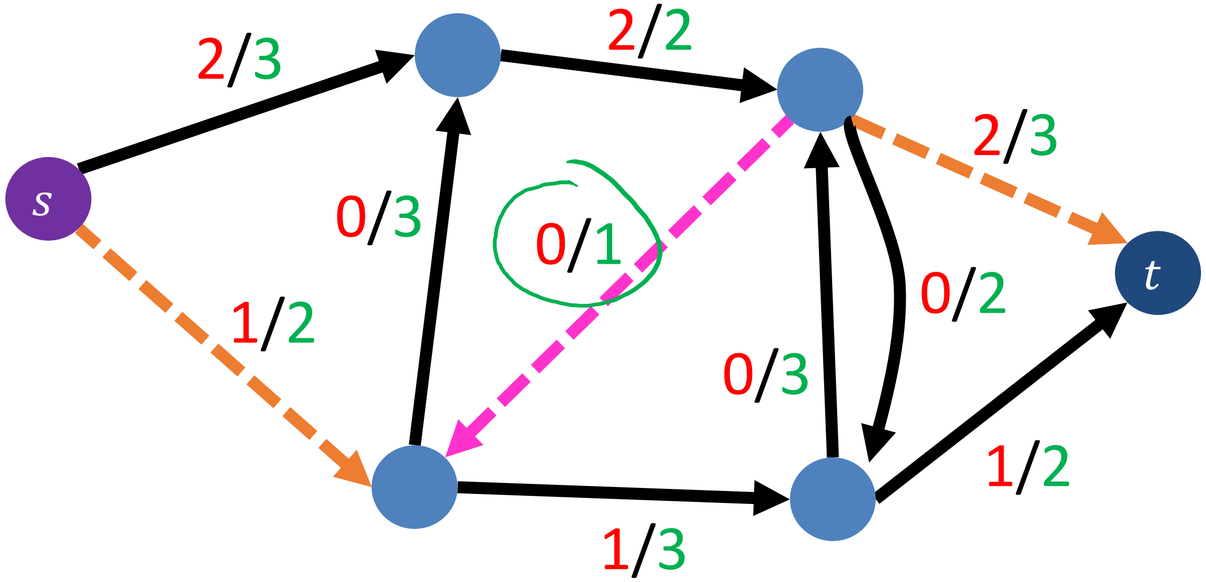
Ford-Fulkerson Example



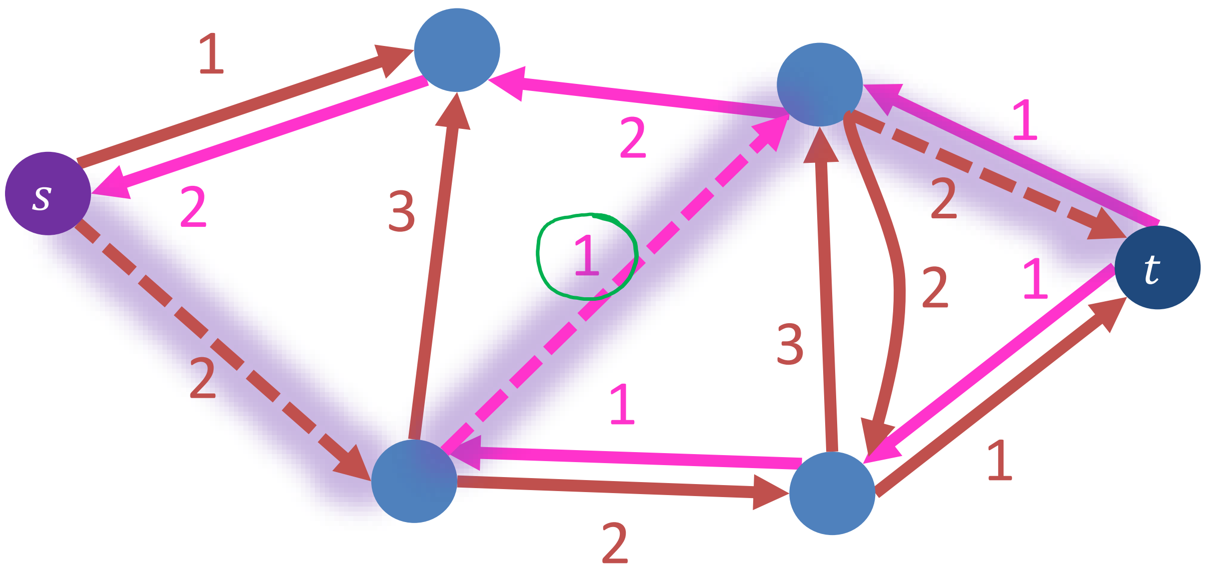


Residual graph G_f

Ford-Fulkerson Example

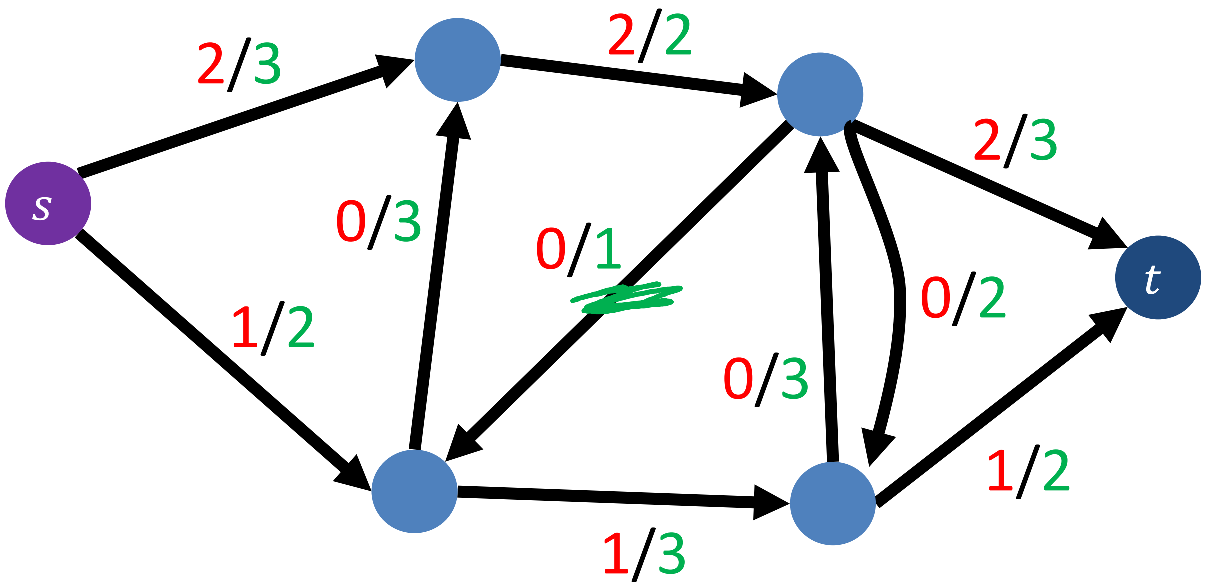


Increase flow by 1 unit

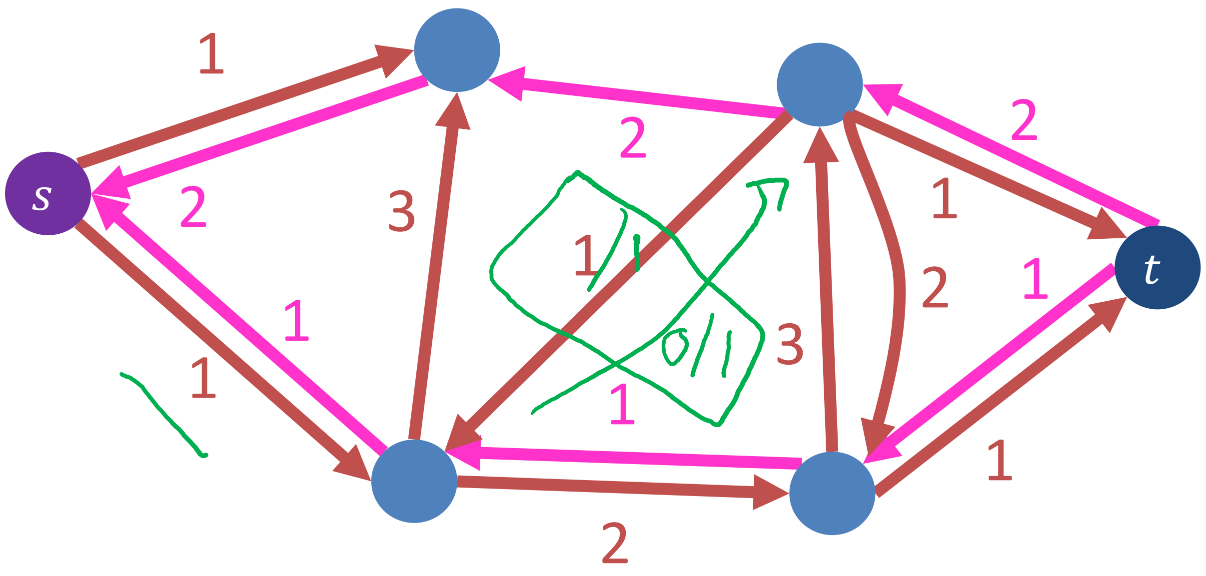


Residual graph G_f

Ford-Fulkerson Example

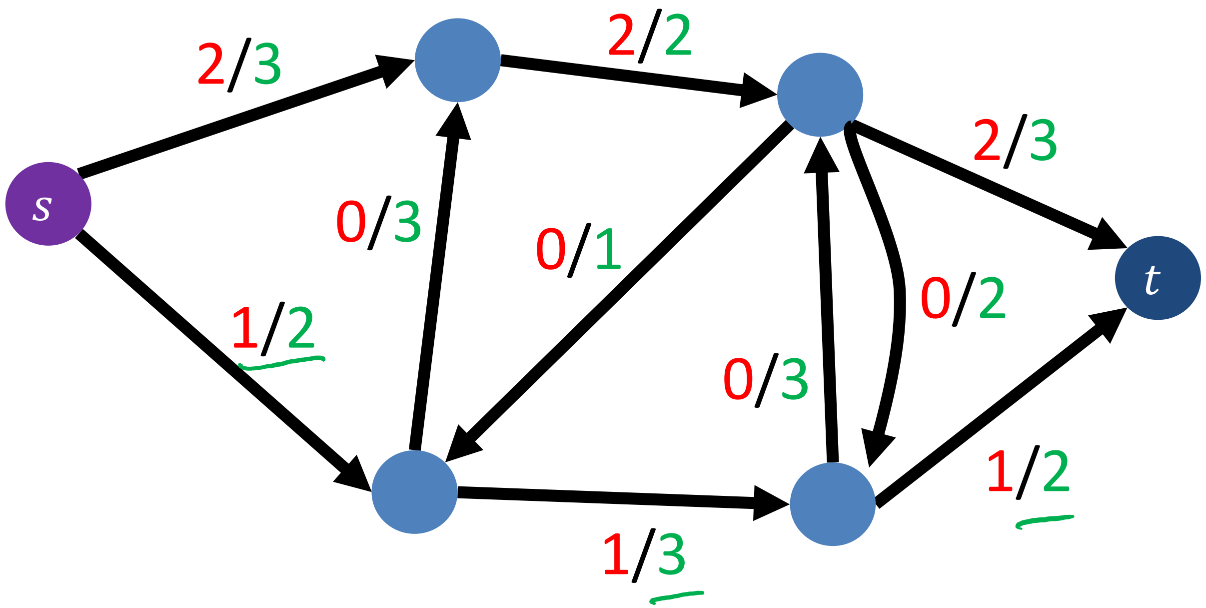


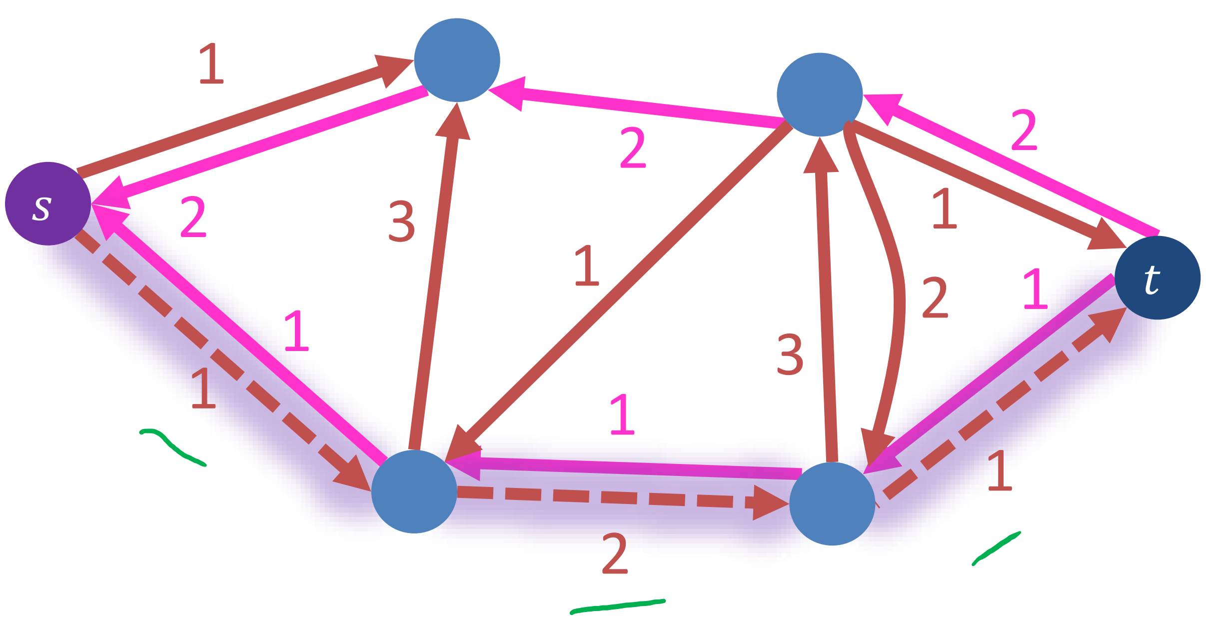
Increase flow by 1 unit



Residual graph G_f

Ford-Fulkerson Example

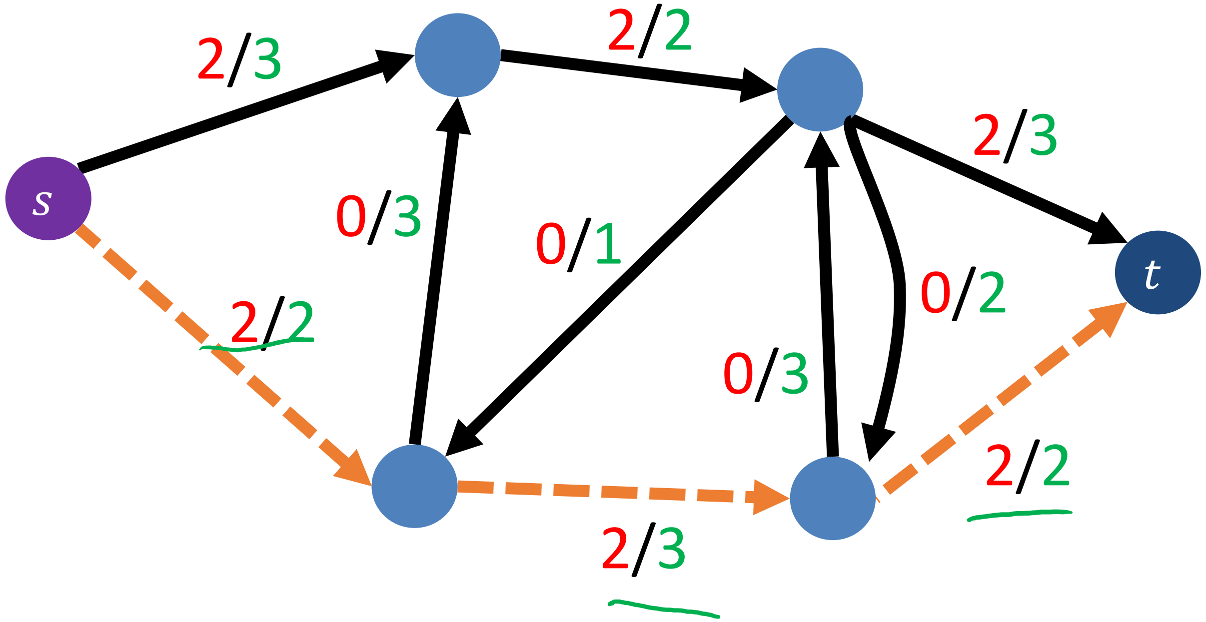


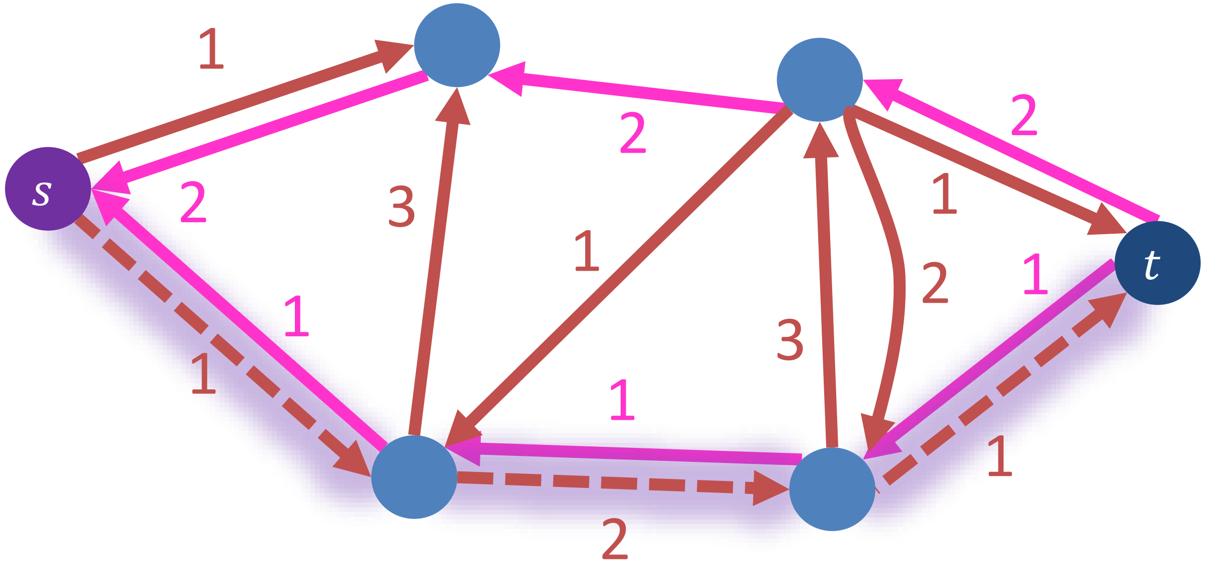


Residual graph G_f

Ford-Fulkerson Example

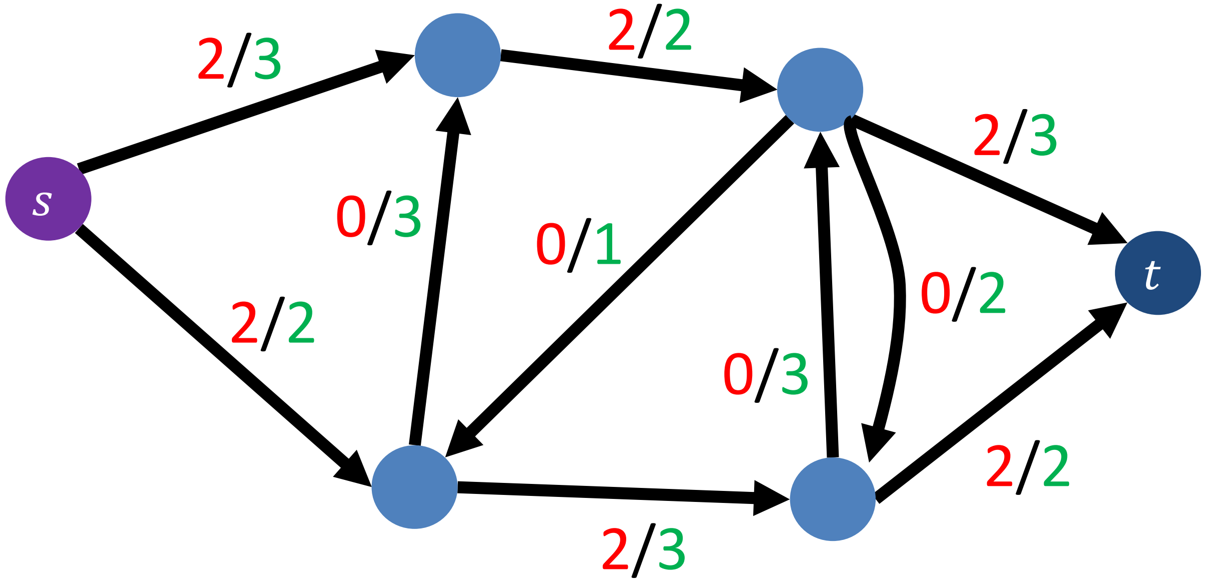
Increase flow by 1 unit

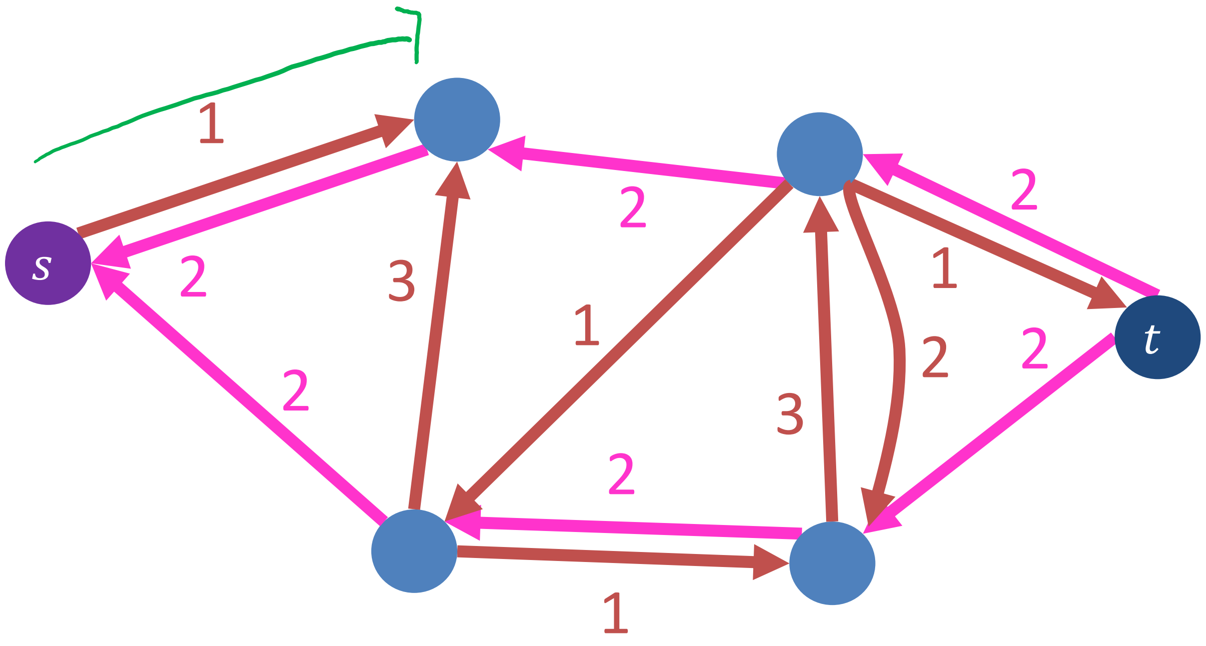




Residual graph G_f

Ford-Fulkerson Example

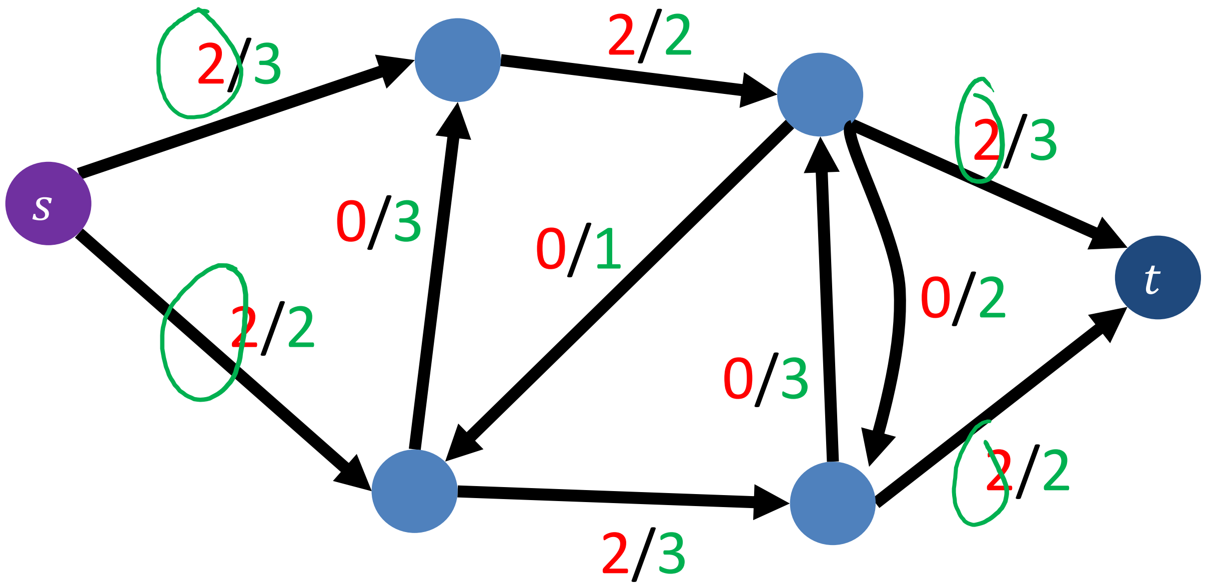


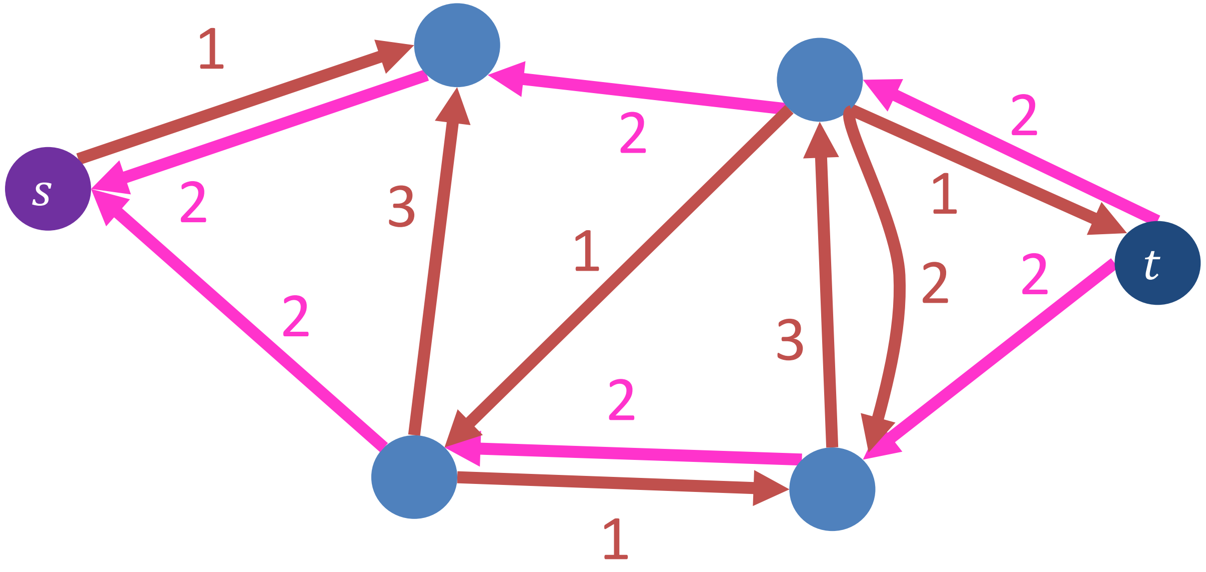


Residual graph G_f

Ford-Fulkerson Example

No more augmenting paths





Maximum flow: 4

Residual graph G_f

Ford-Fulkerson Algorithm - Runtime

Define an **augmenting path** to be a path from $s \to t$ in the residual graph G_f (using edges of non-zero weight)

Overview: Repeatedly add the flow of any augmenting path

Ford-Fulkerson max-flow algorithm:

- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Time to find an augmenting path:

Number of iterations of While loop:

Ford-Fulkerson Algorithm - Runtime

Define an **augmenting path** to be a path from $s \to t$ in the residual graph G_f (using edges of non-zero weight)

Overview: Repeatedly add the flow of any augmenting path

Ford-Fulkerson max-flow algorithm:

- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

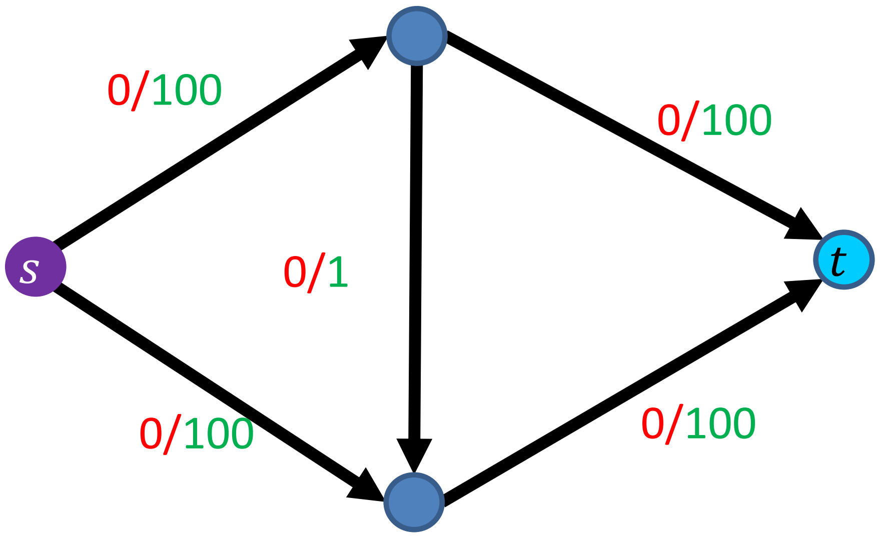
Time to find an augmenting path: $\text{BFS: } \Theta(V + E)$

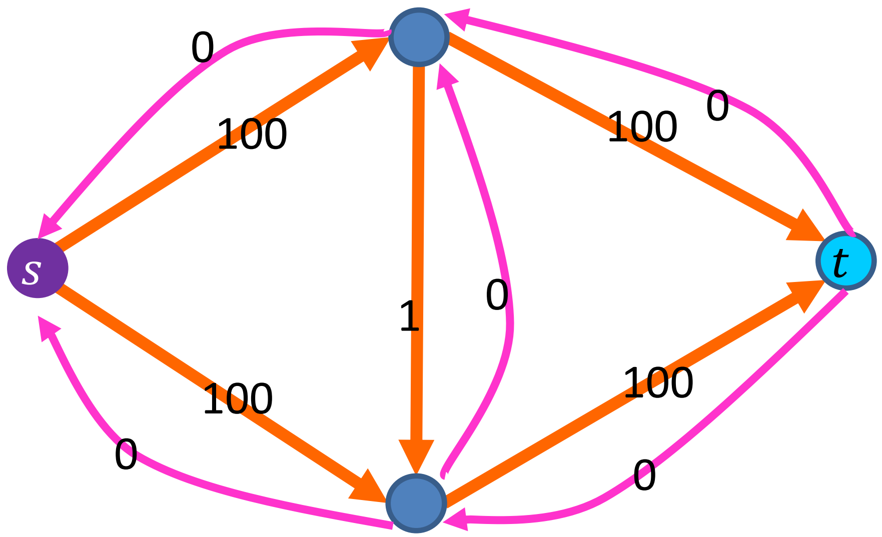
Number of iterations of While loop: $|f|$

$$\Theta(E \cdot |f|)$$

Why might we loop $|f|$ times?

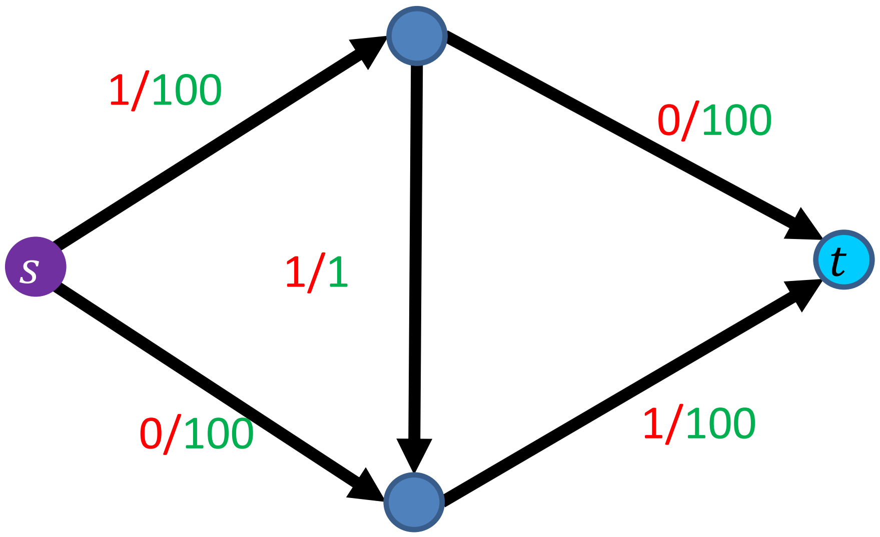
- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

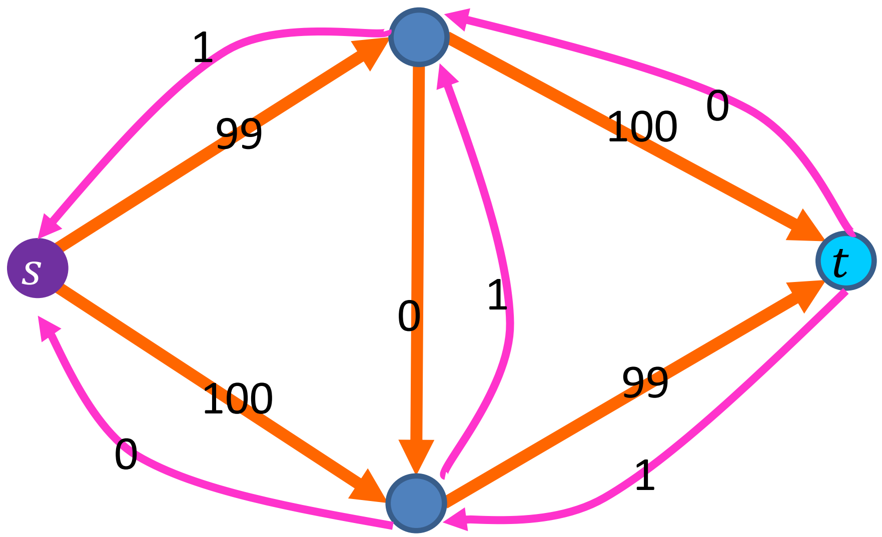




Why might we loop $|f|$ times?

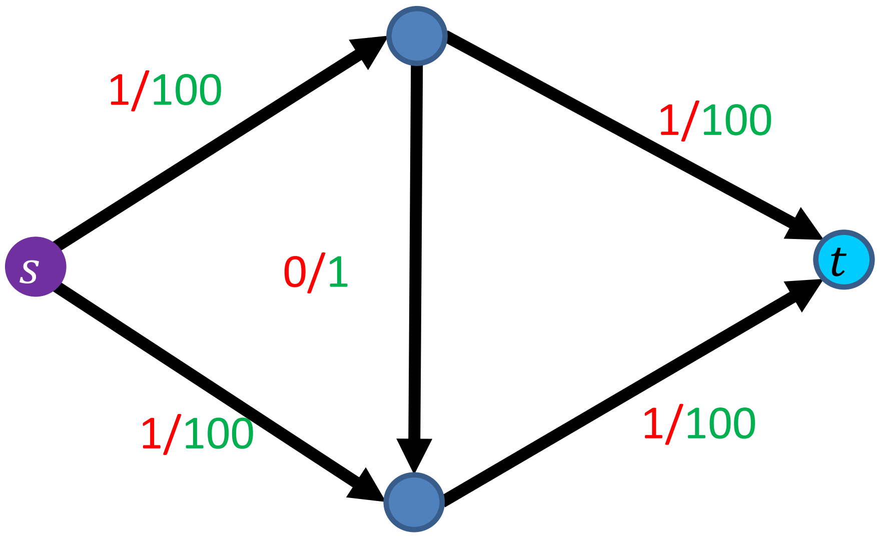
- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u,v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

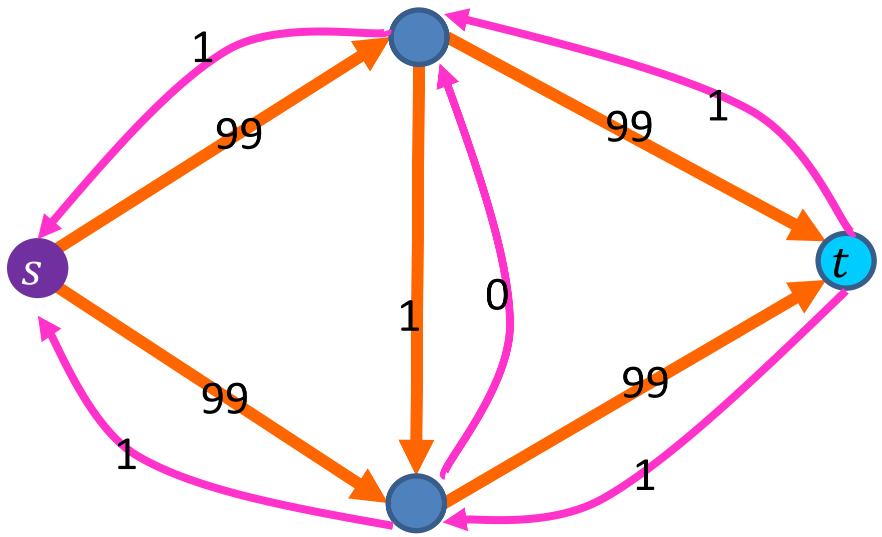




Why might we loop $|f|$ times?

- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow



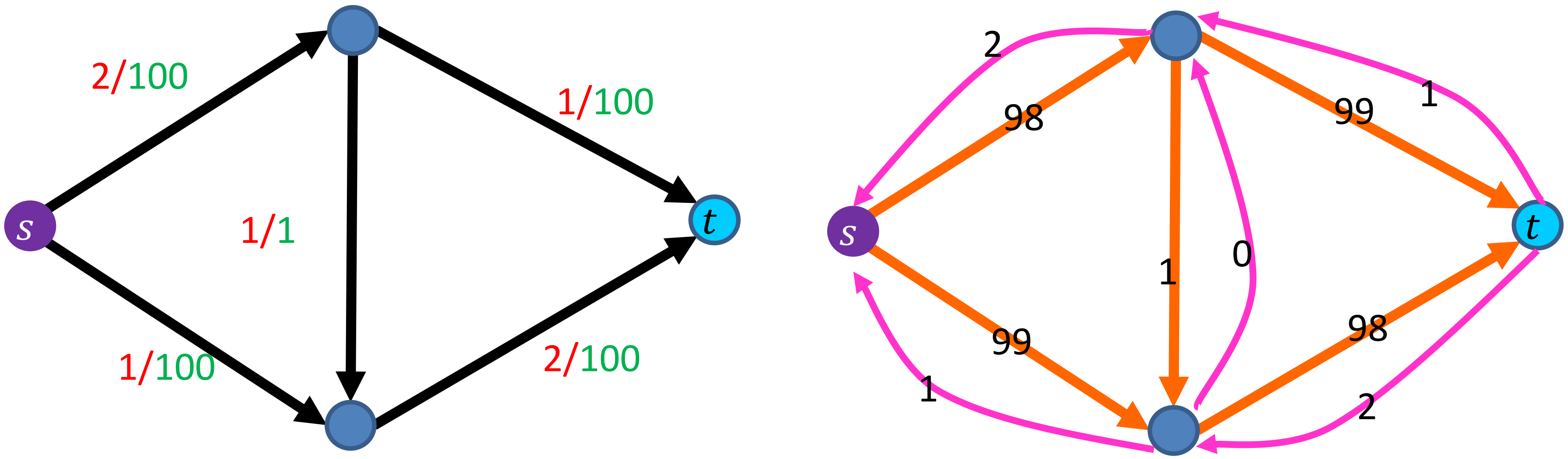


Why might we loop $|f|$ times?

- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Each time we increase flow by 1

Loop runs 200 times



Can We Avoid this?

- **Edmonds-Karp Algorithm:** choose augmenting path with fewest hops
- **Running time:** $\Theta\left(\min\left(|E||f^*|, |V||E|^2\right)\right) = O\left(|V||E|^2\right)$

Edmonds-Karp max-flow algorithm:

- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path in G_f, let p be the path with fewest hops:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Proof: See CLRS (Chapter 26.2)

Can We Avoid this?

- **Edmonds-Karp Algorithm:** choose augmenting path with fewest hops
- **Running time:** $\Theta\left(\min\left(|E||f^*|, |V||E|^2\right)\right)$

How to find this?
Use breadth-first search (BFS)!

Edmonds-Karp = Ford-Fulkerson using BFS to find augmenting path

Edmonds-Karp max-flow algorithm:

- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path in G_f, let p be the path with fewest hops:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Proof: See CLRS (Chapter 26.2)