CS4102 Algorithms

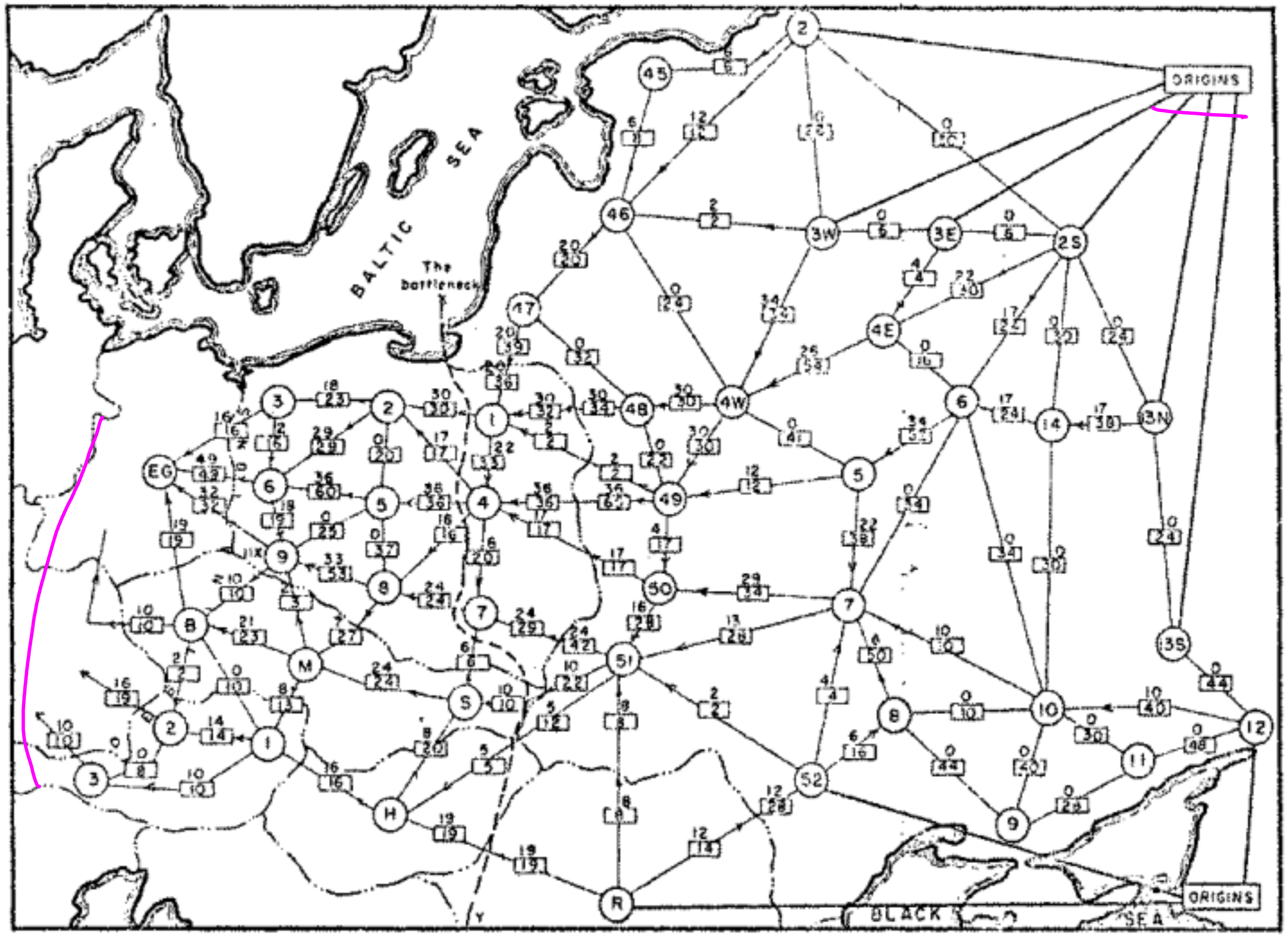
Spring 2020

Today's Keywords

- Graphs
- **MaxFlow/MinCut**
- Ford-Fulkerson
- Edmunds-Karp

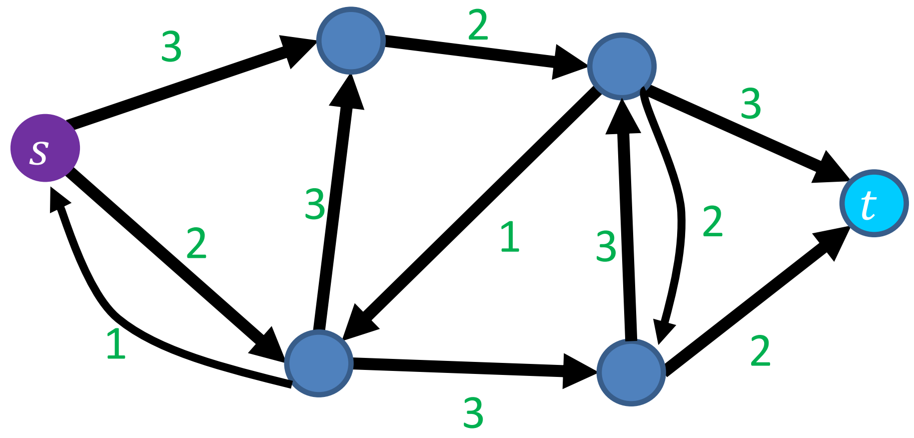
CLRS Readings

- Chapter 25, 26



Railway map of Western USSR, 1955

Flow Network



Graph $G = (V, E)$

Source node $s \in V$

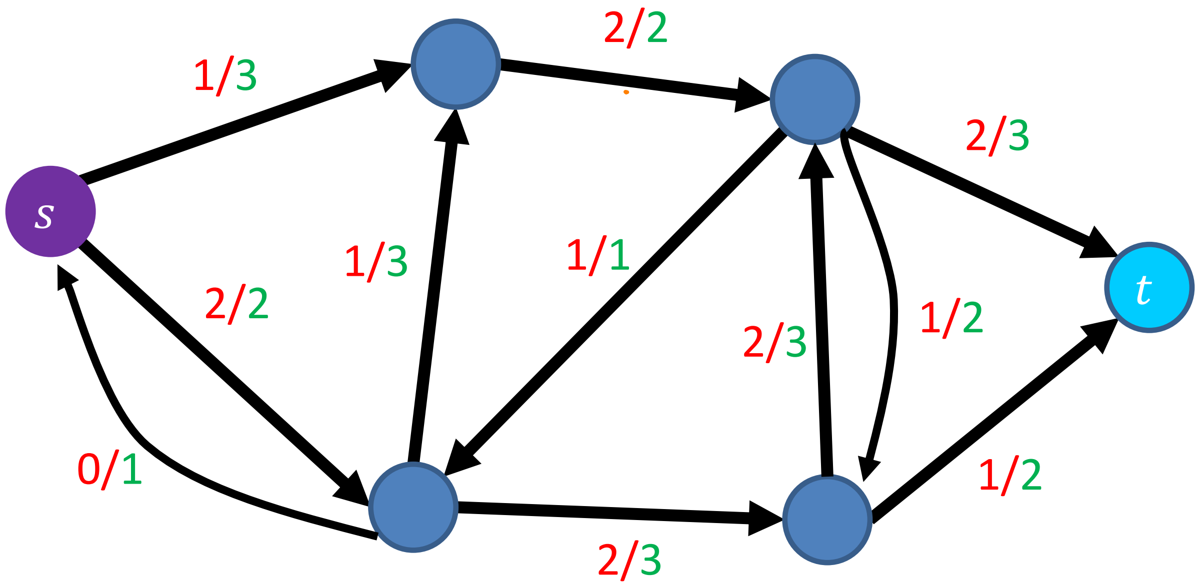
Sink node $t \in V$

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



Flow/Capacity

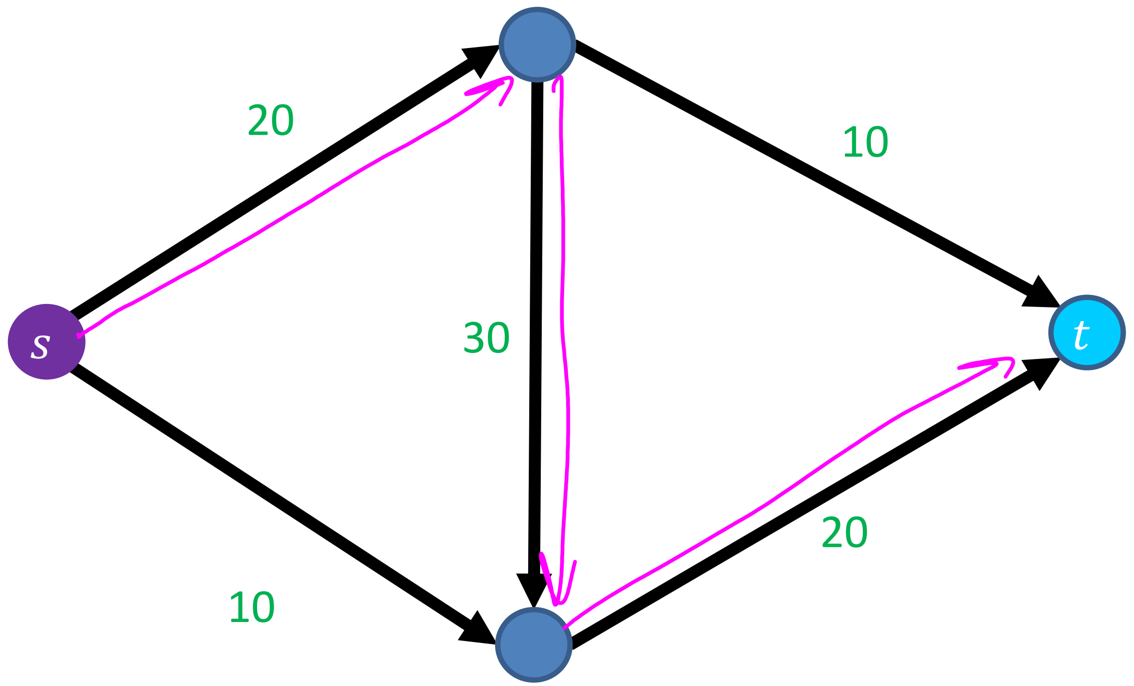
3 in example above

Max Flow

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

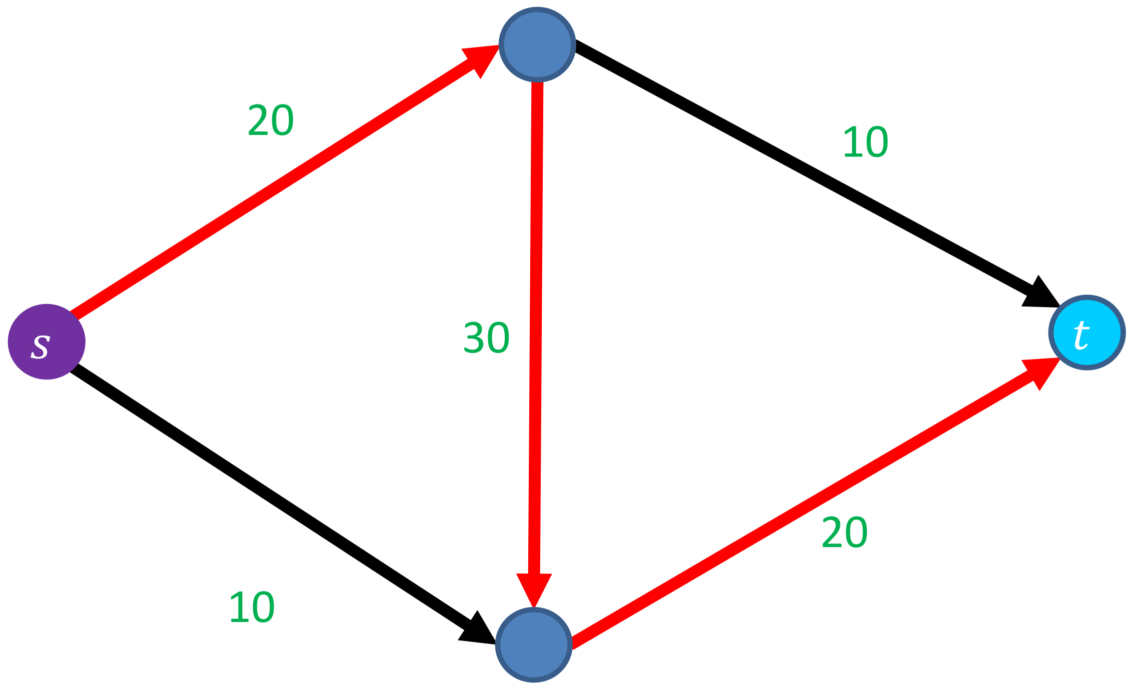
Greedy doesn't work

Saturate Highest Capacity Path First



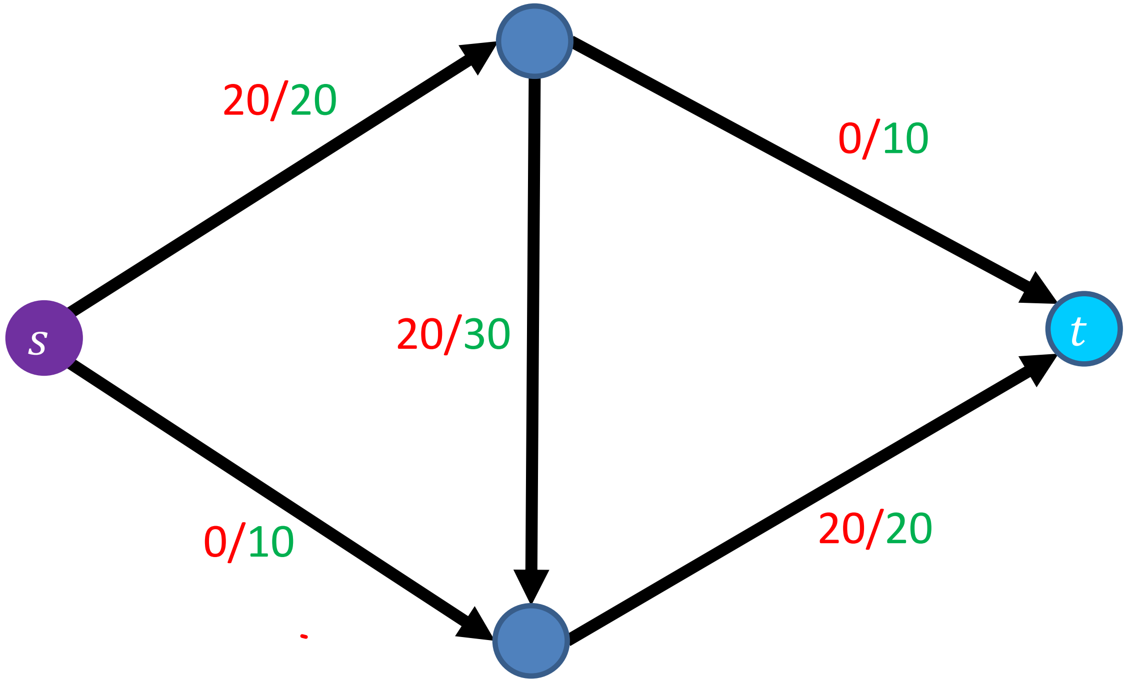
Greedy doesn't work

Saturate Highest Capacity Path First



Greedy doesn't work

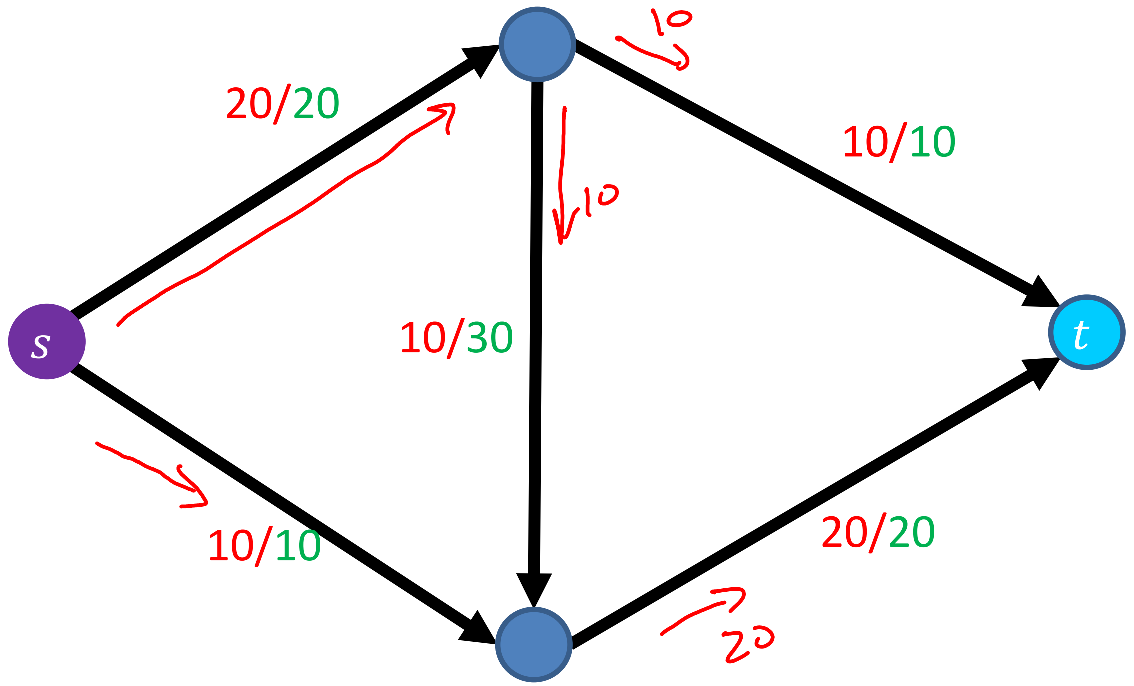
Saturate Highest Capacity Path First



Overall Flow: $|f| = 20$

Greedy doesn't work

Better Solution



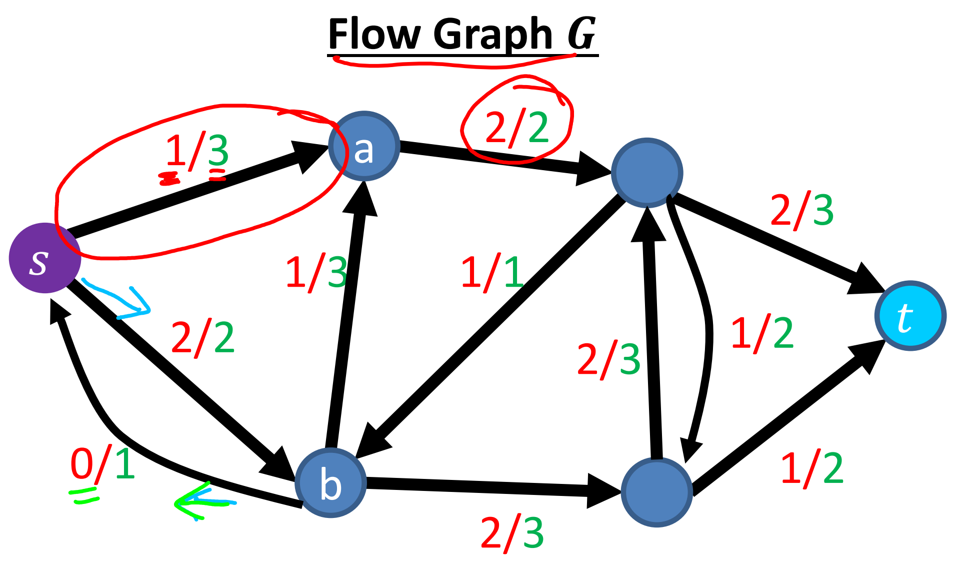
Overall Flow: $|f| = \underline{30}$

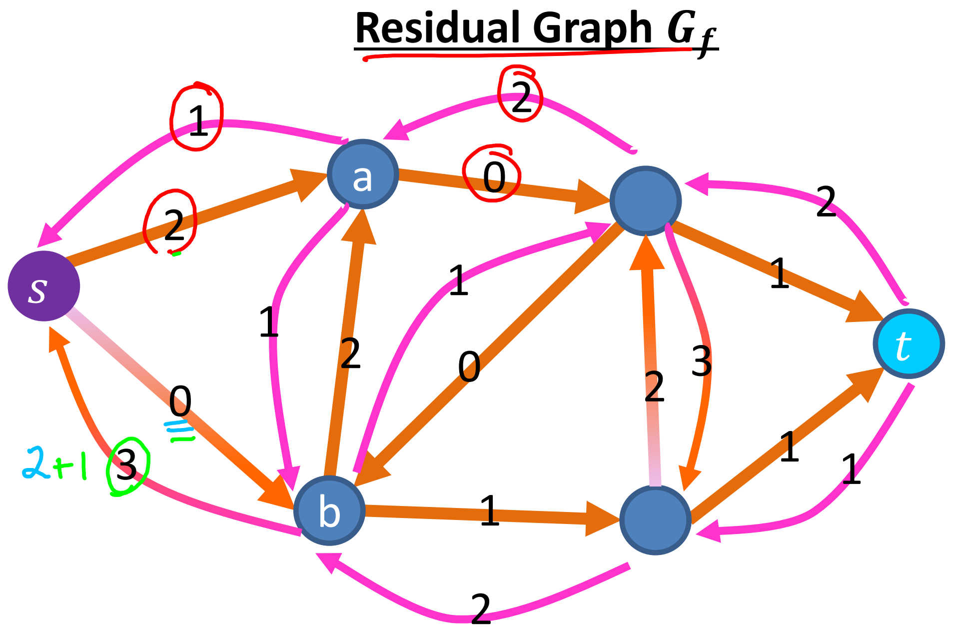
Residual Graph G_f

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

Flow I *could* add

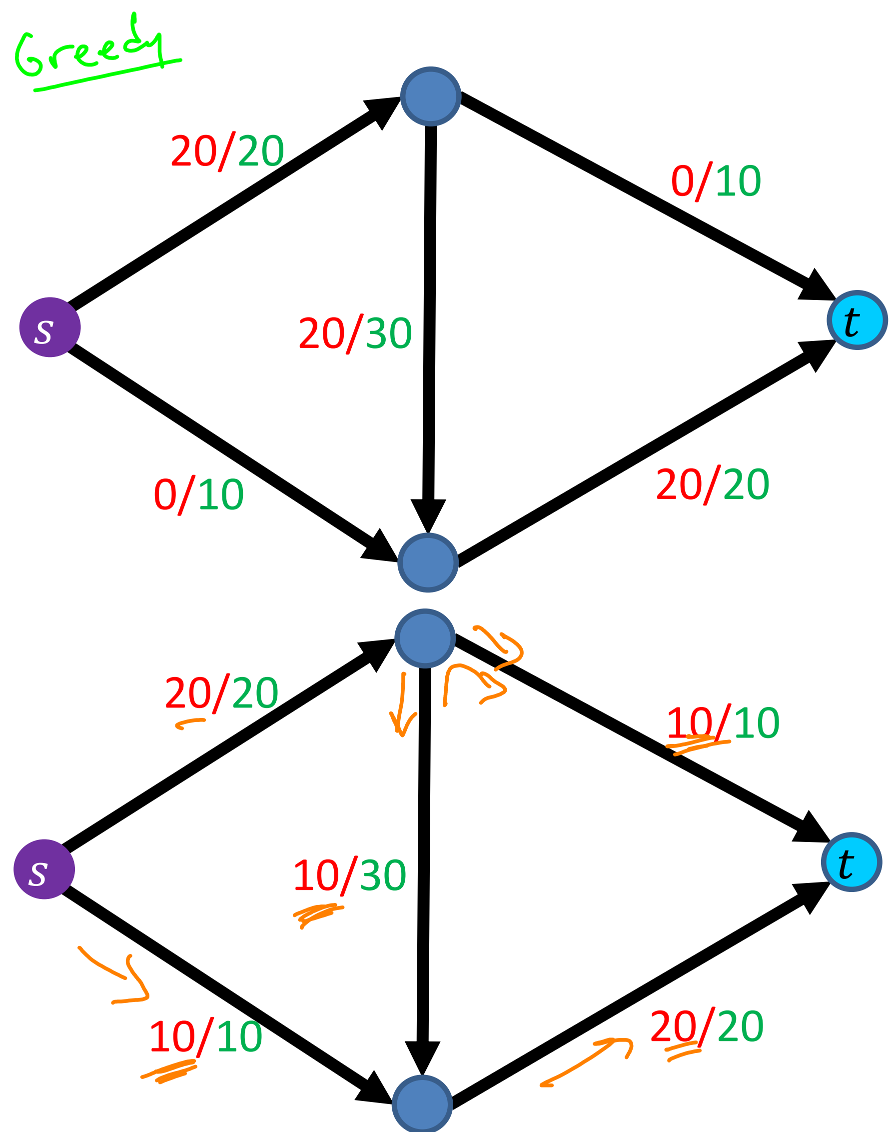
Flow I *could* remove



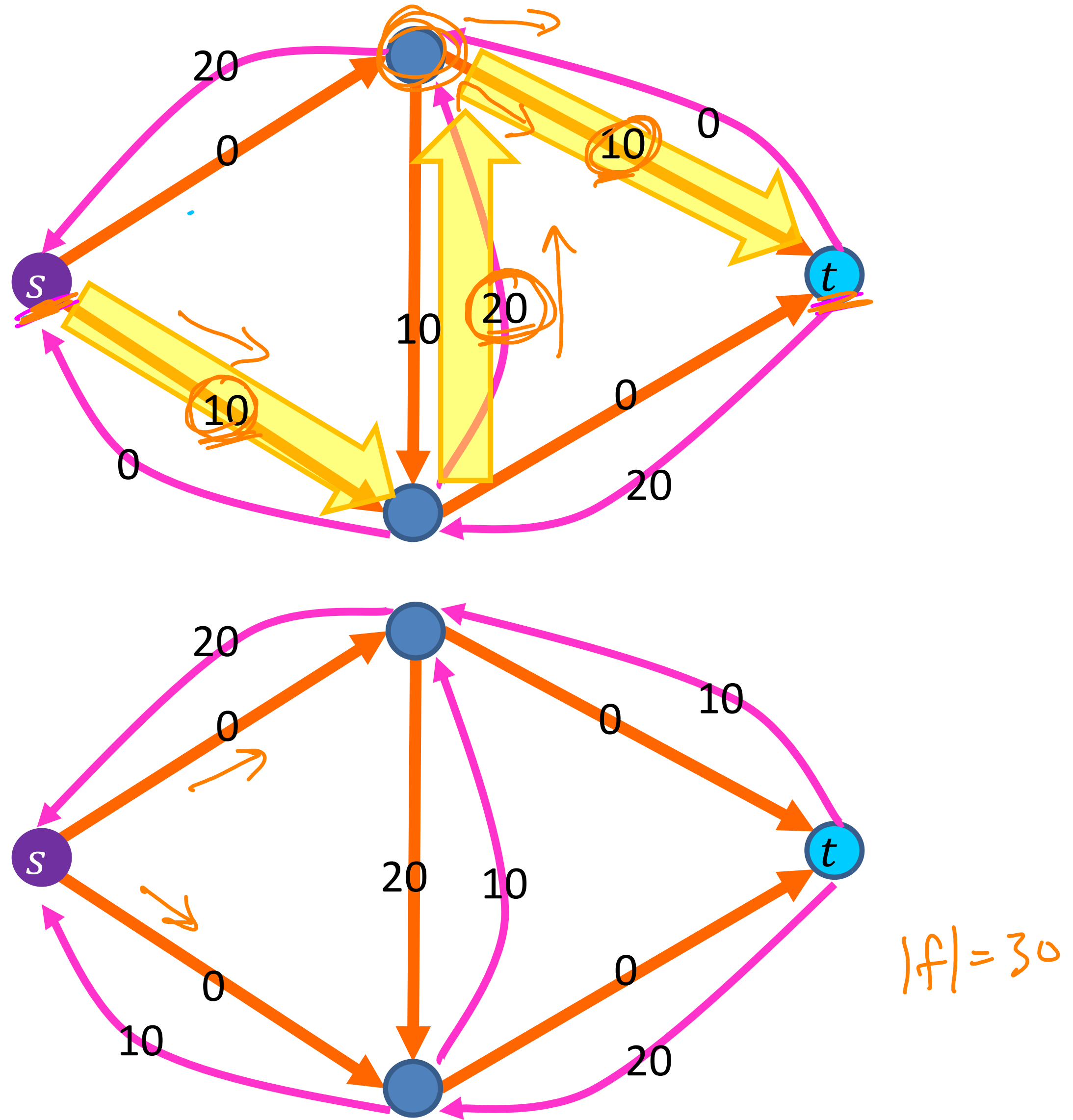


Residual Graphs Example

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 any 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

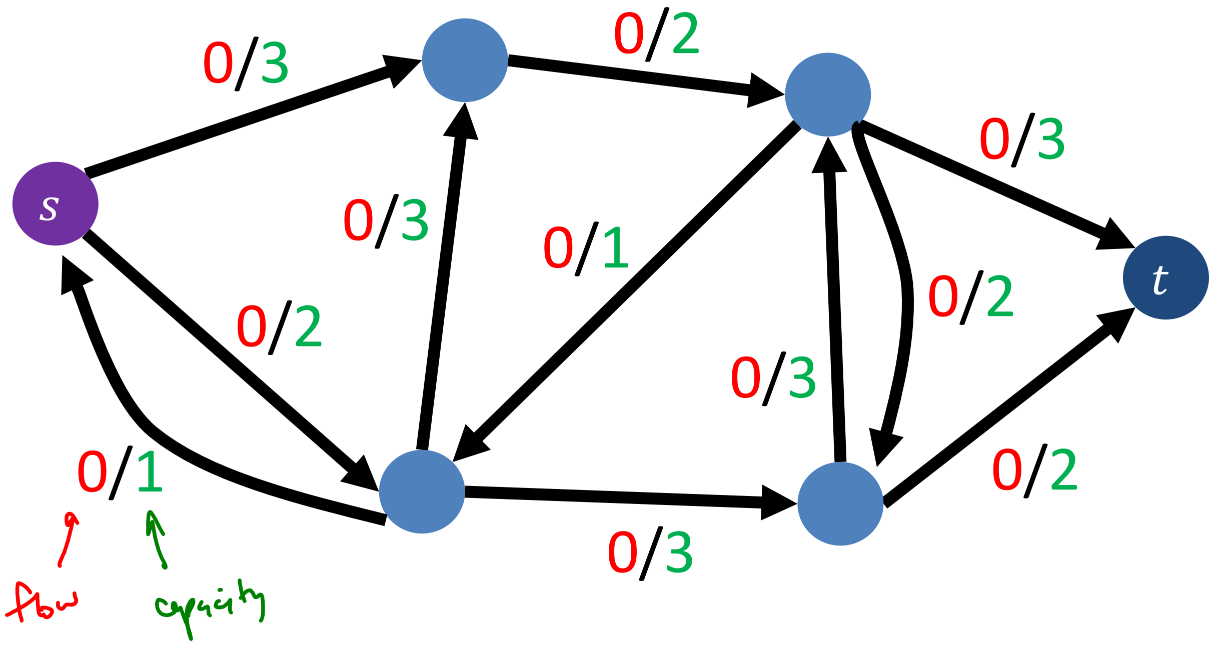
Ford-Fulkerson max-flow algorithm:

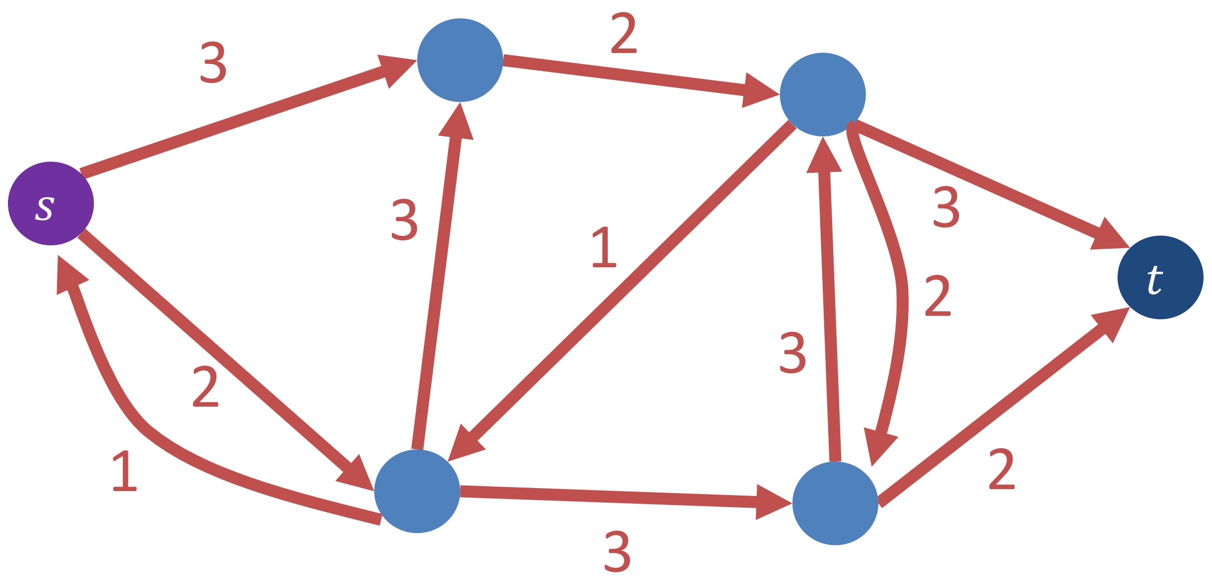
- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u,v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

Ford-Fulkerson approach: take any 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

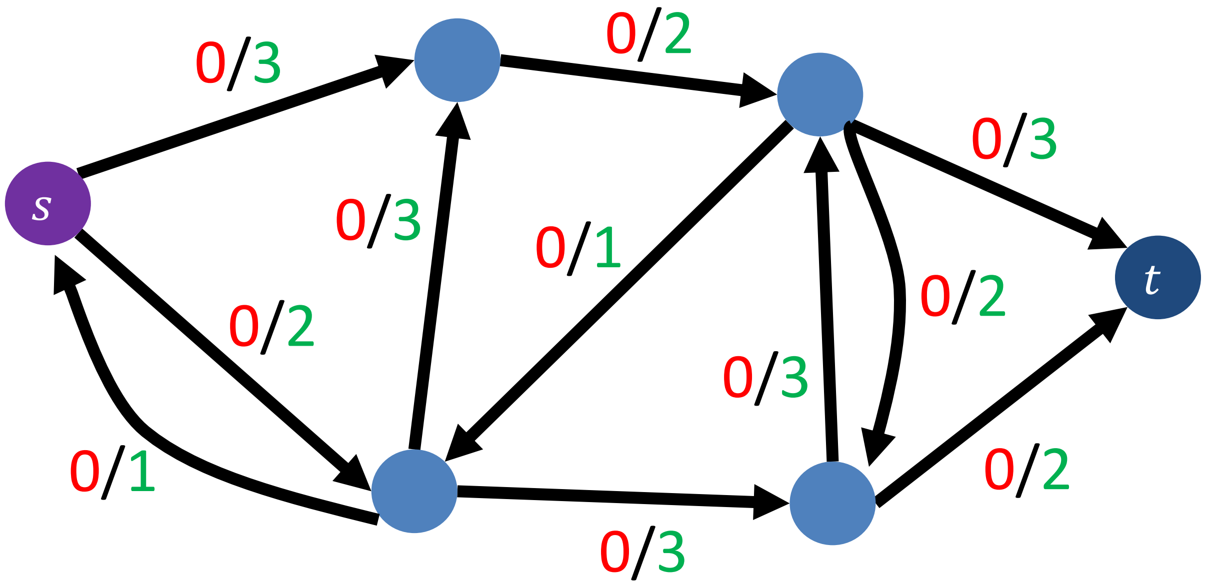




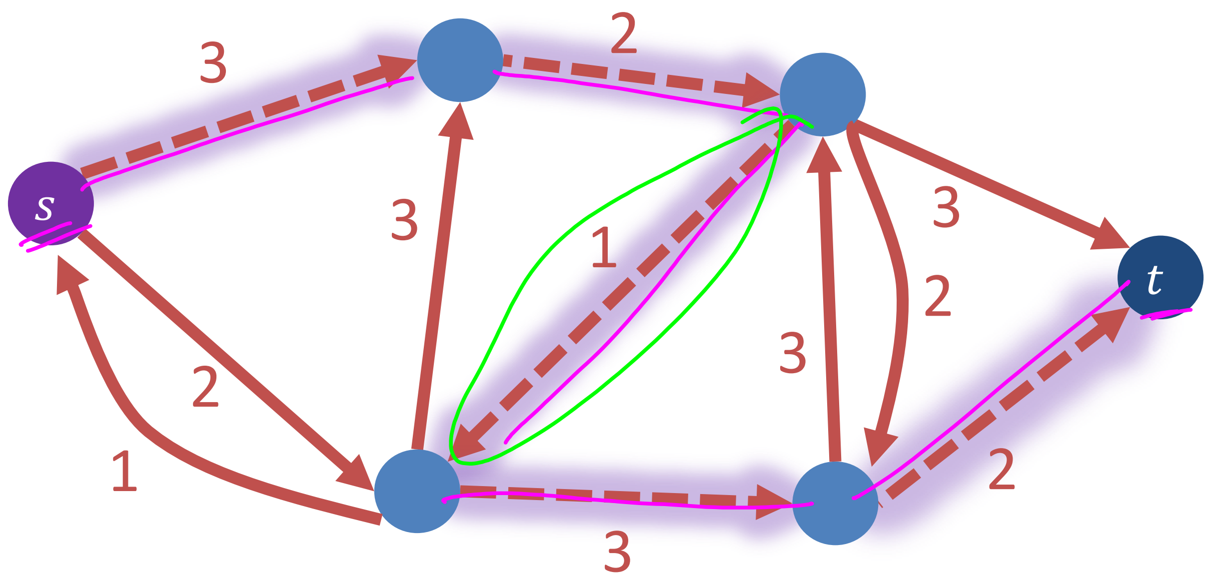
Initially: $f(e) = 0$ for all $e \in E$

Residual graph G_f

Ford-Fulkerson Example

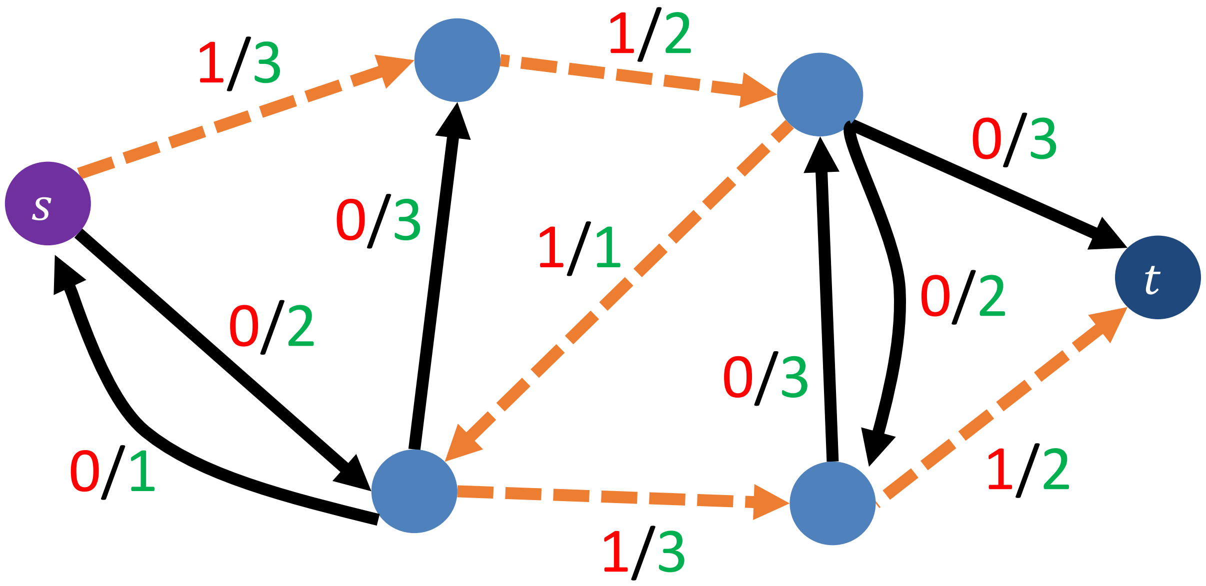


Increase flow by 1 unit

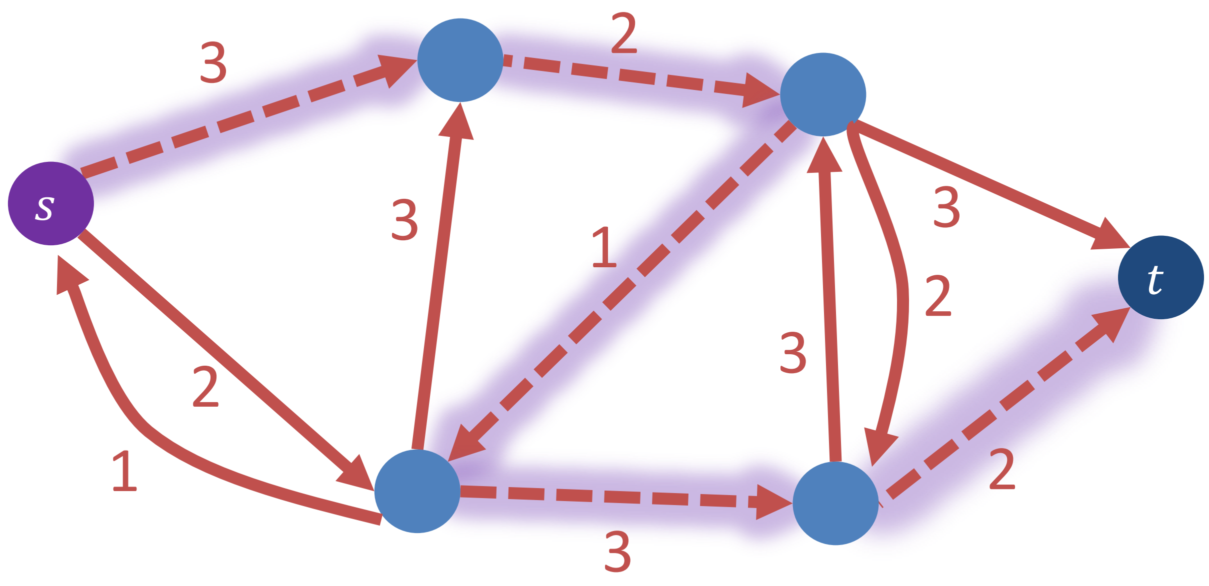


Residual graph G_f

Ford-Fulkerson Example

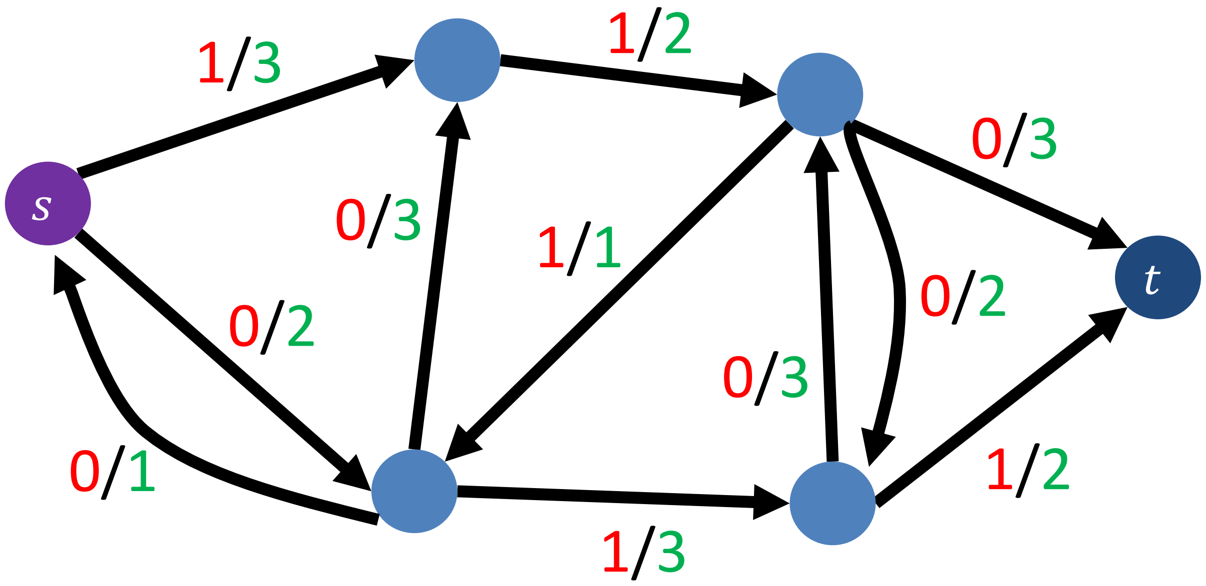


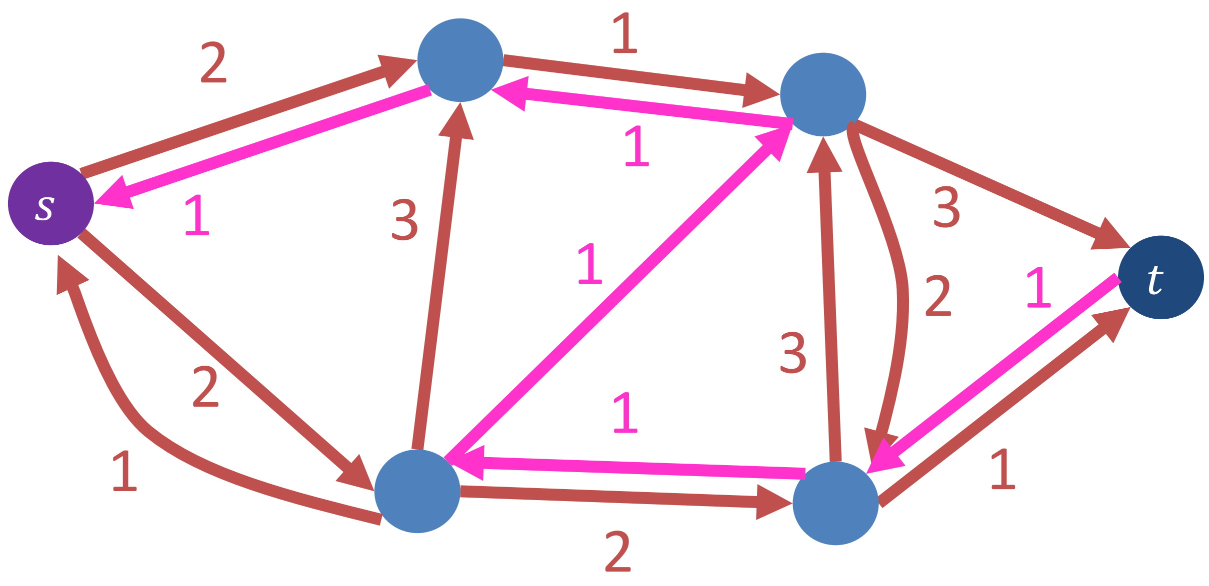
Increase flow by 1 unit



Residual graph G_f

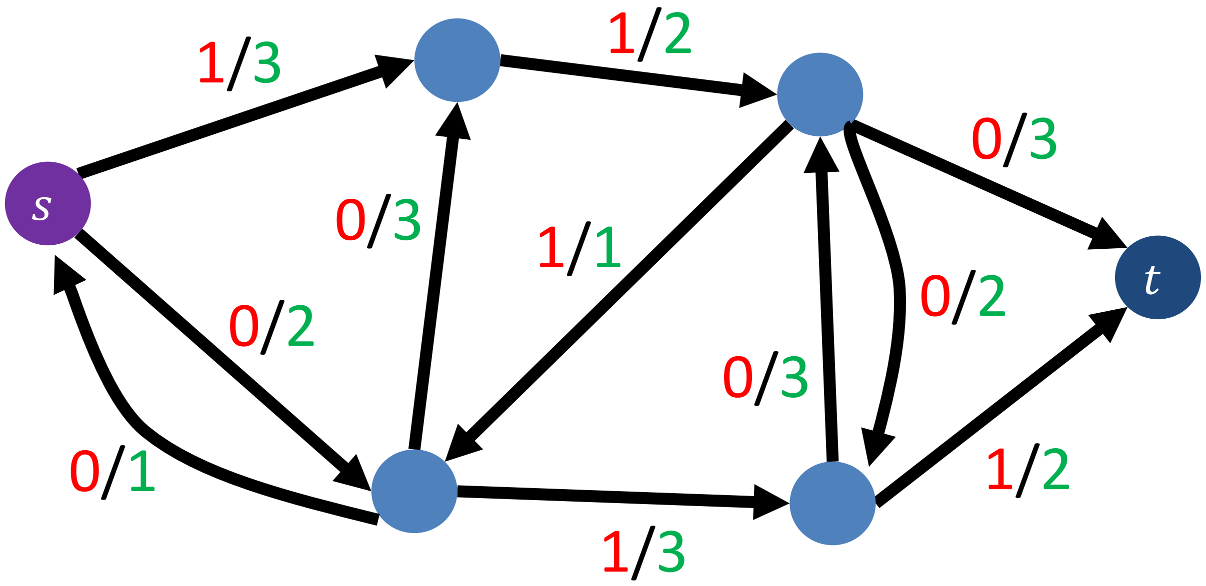
Ford-Fulkerson Example



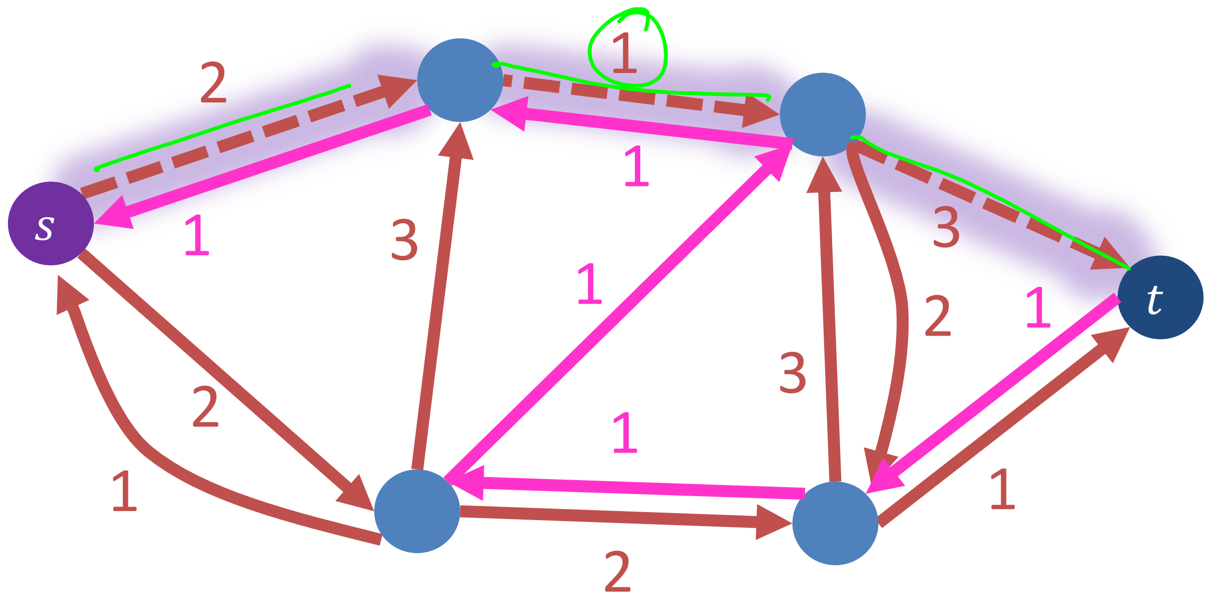


Residual graph G_f

Ford-Fulkerson Example



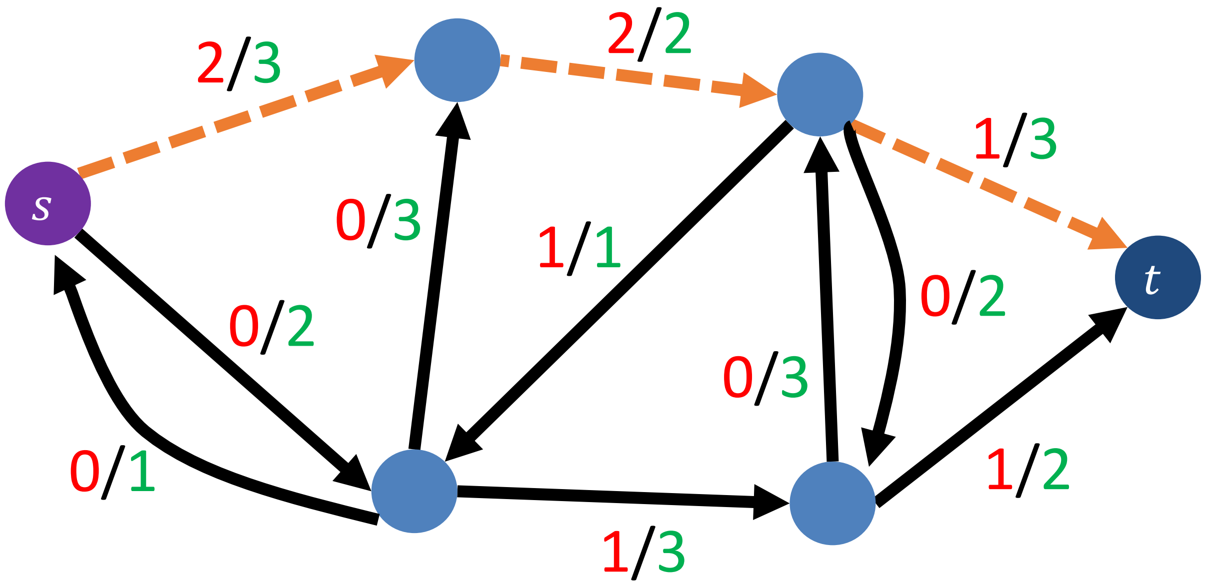
Increase flow by 1 unit

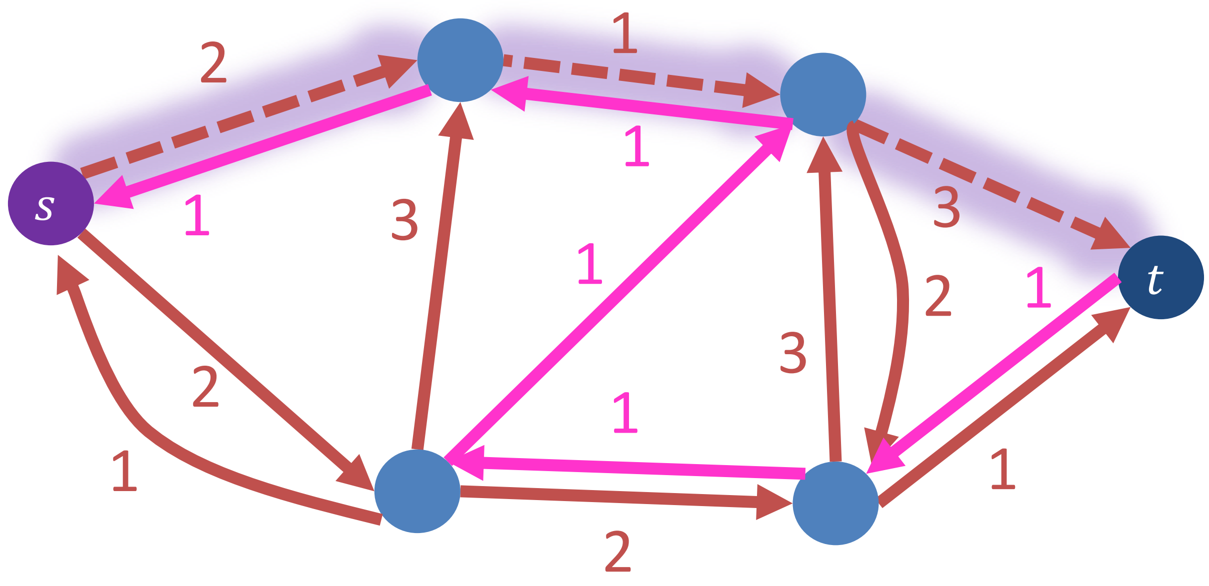


Residual graph G_f

Ford-Fulkerson Example

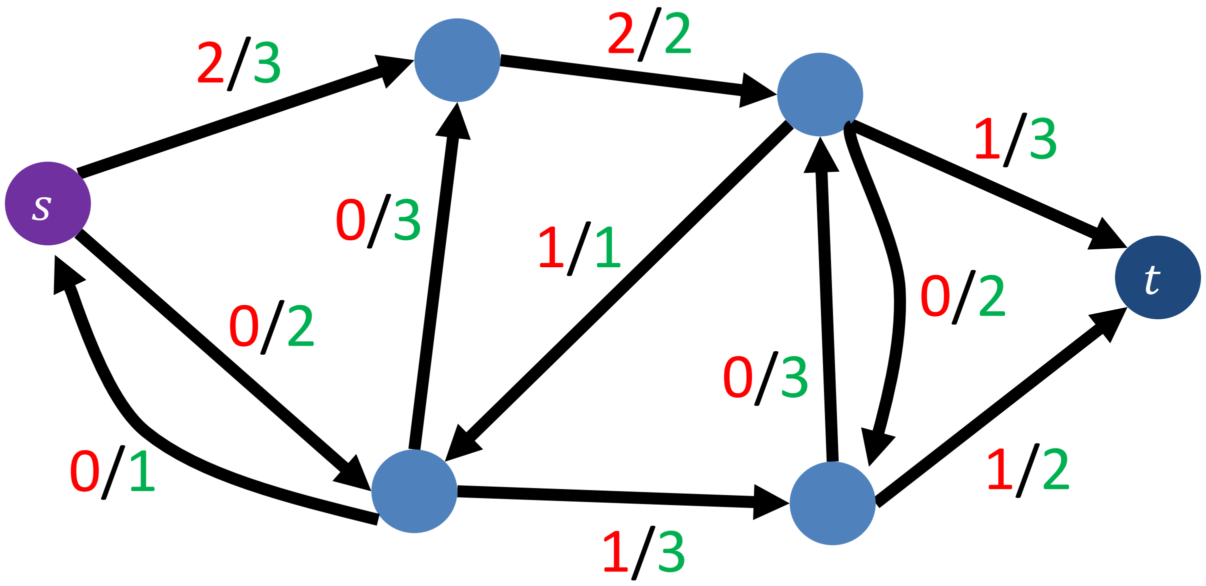
Increase flow by 1 unit



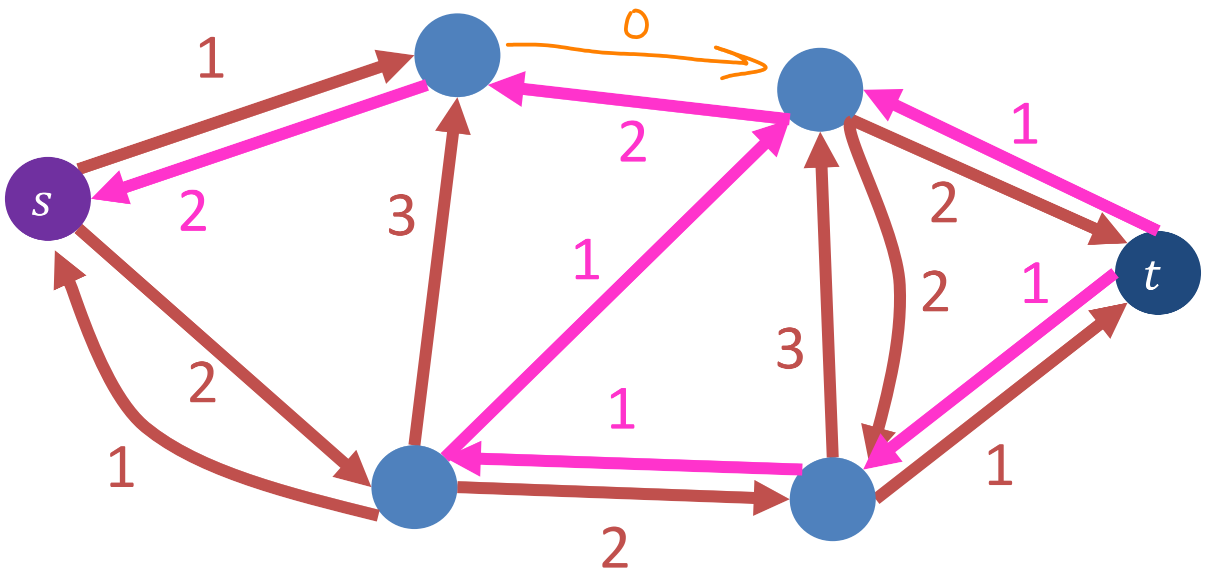


Residual graph G_f

Ford-Fulkerson Example

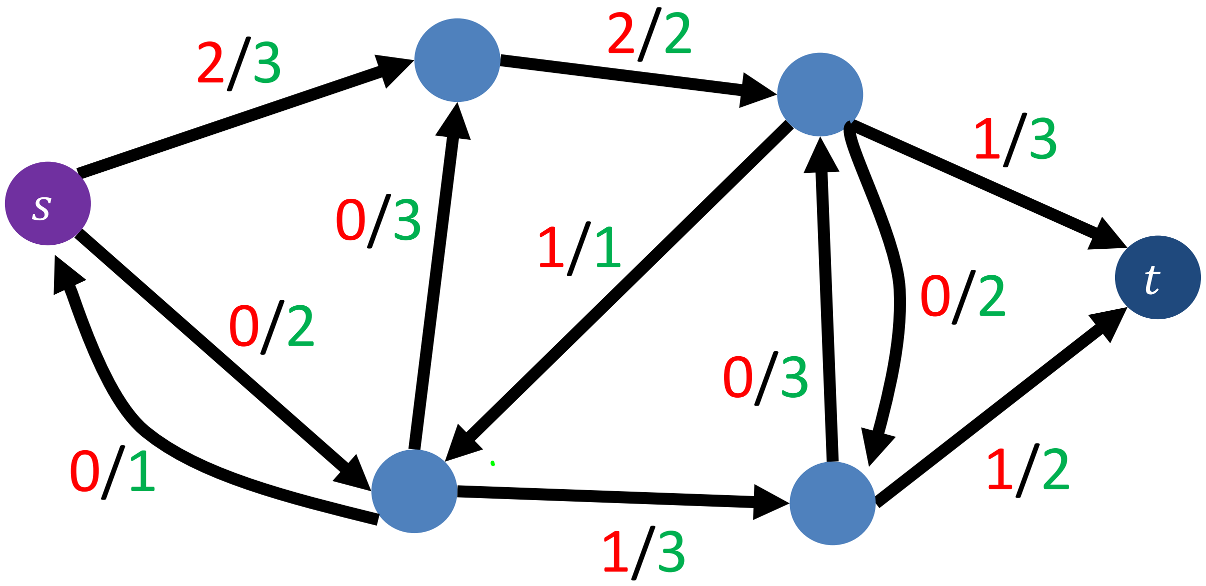


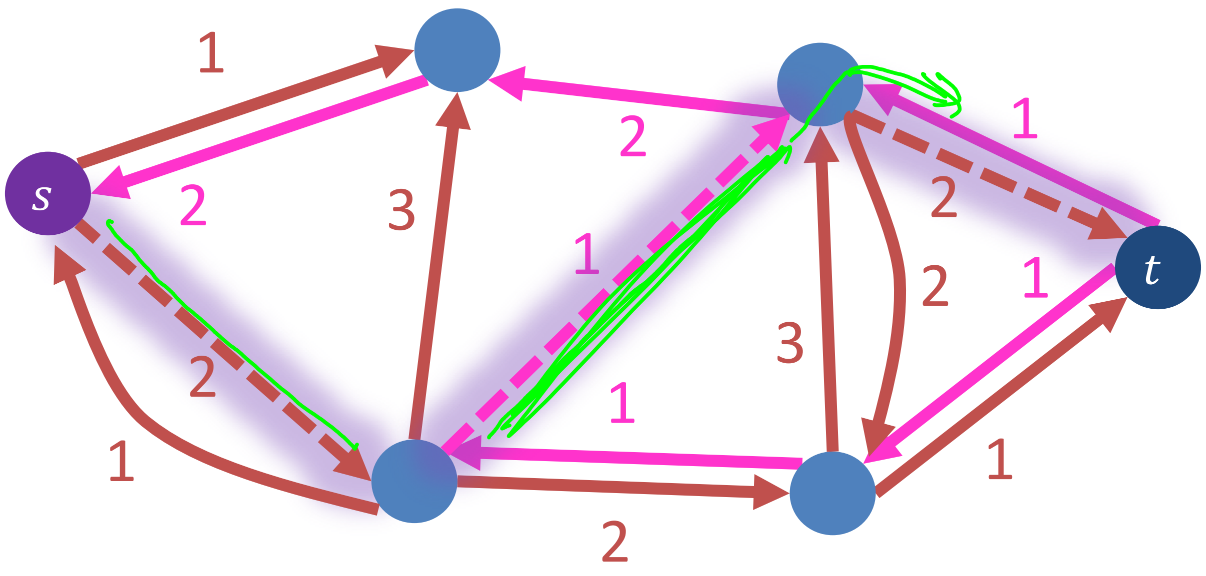
Increase flow by 1 unit



Residual graph G_f

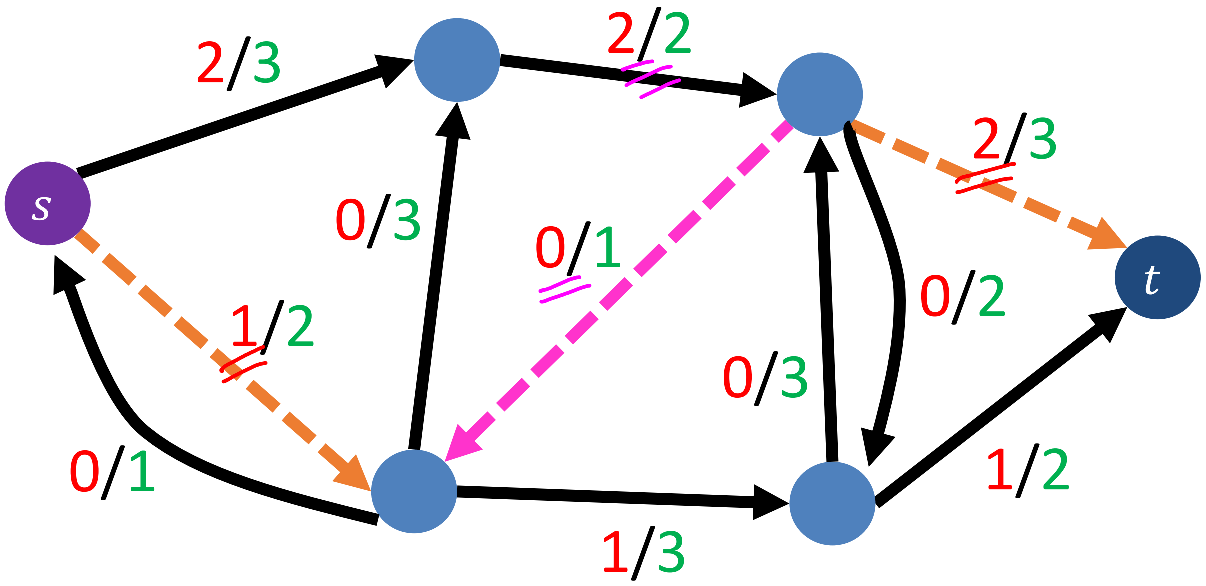
Ford-Fulkerson Example



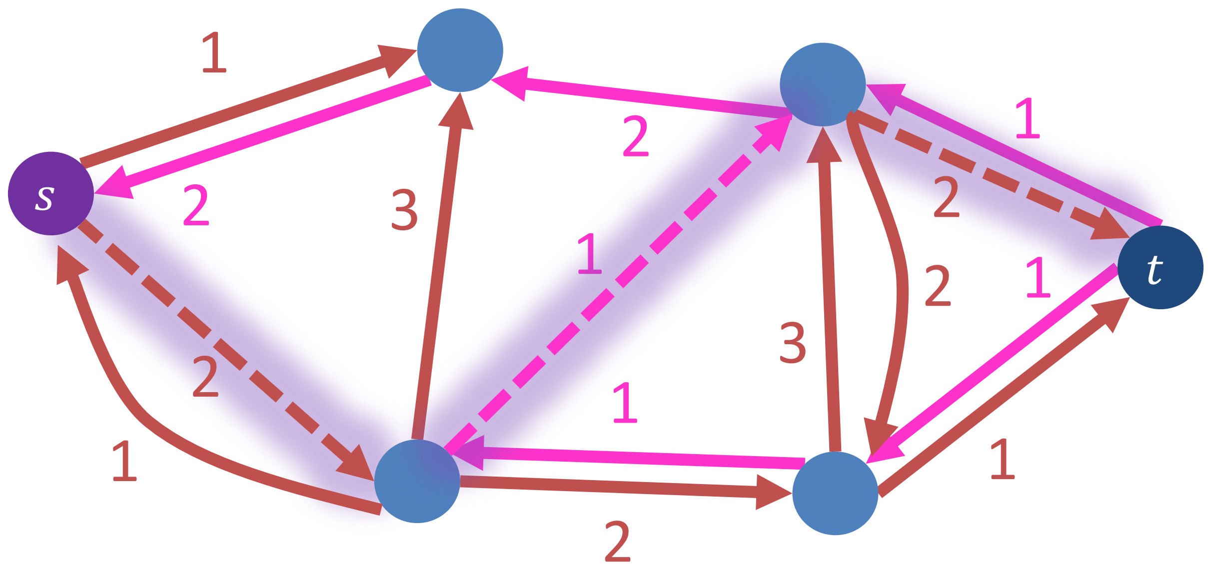


Residual graph G_f

Ford-Fulkerson Example

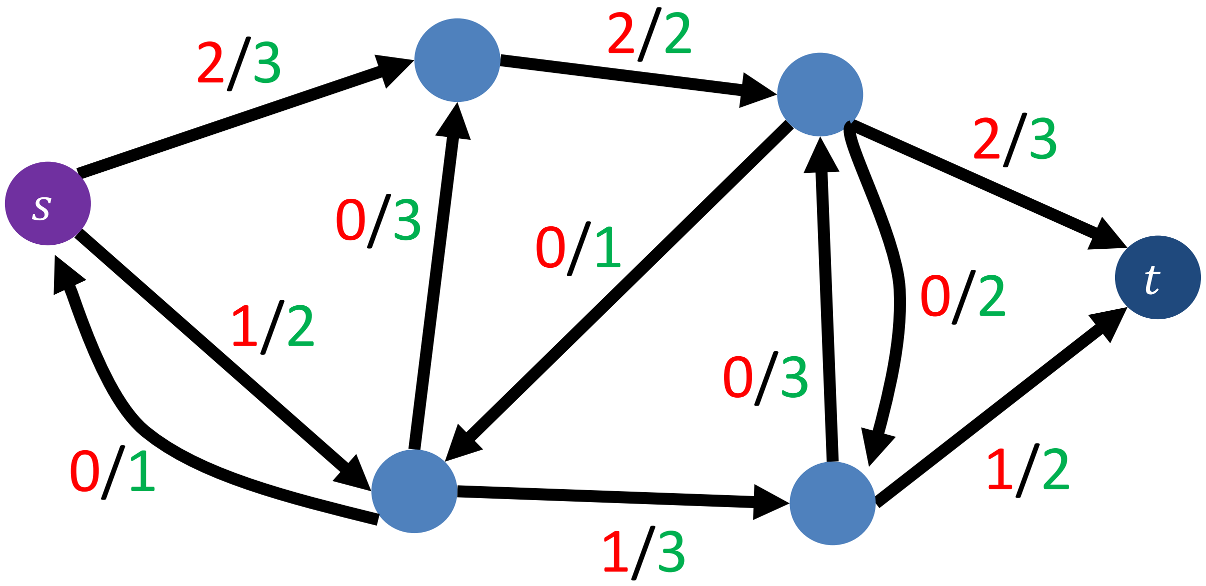


Increase flow by 1 unit

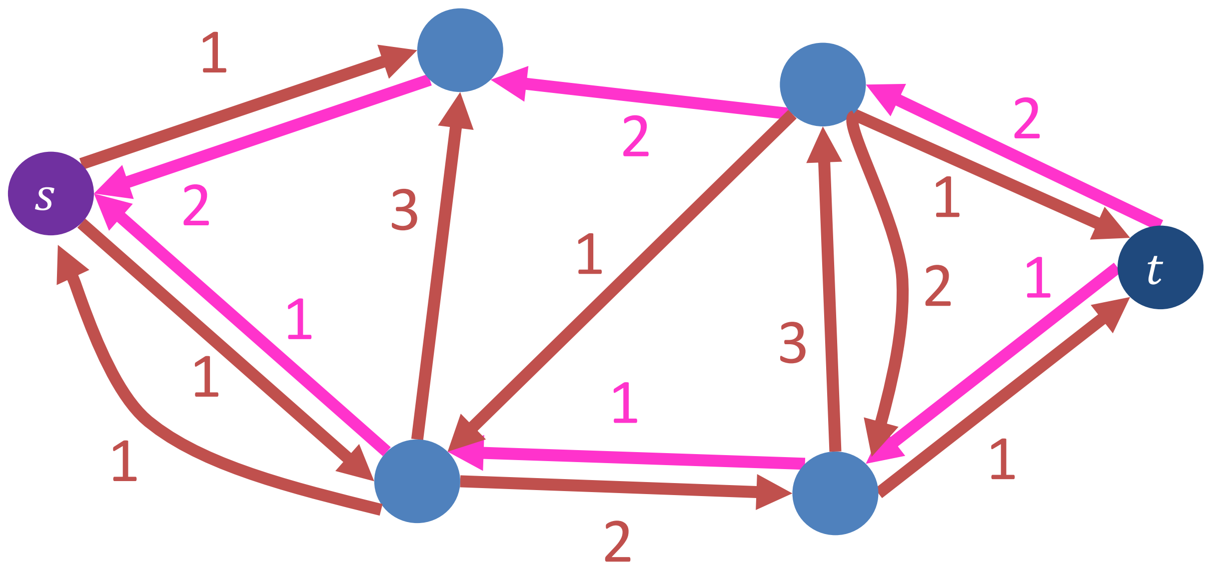


Residual graph G_f

Ford-Fulkerson Example

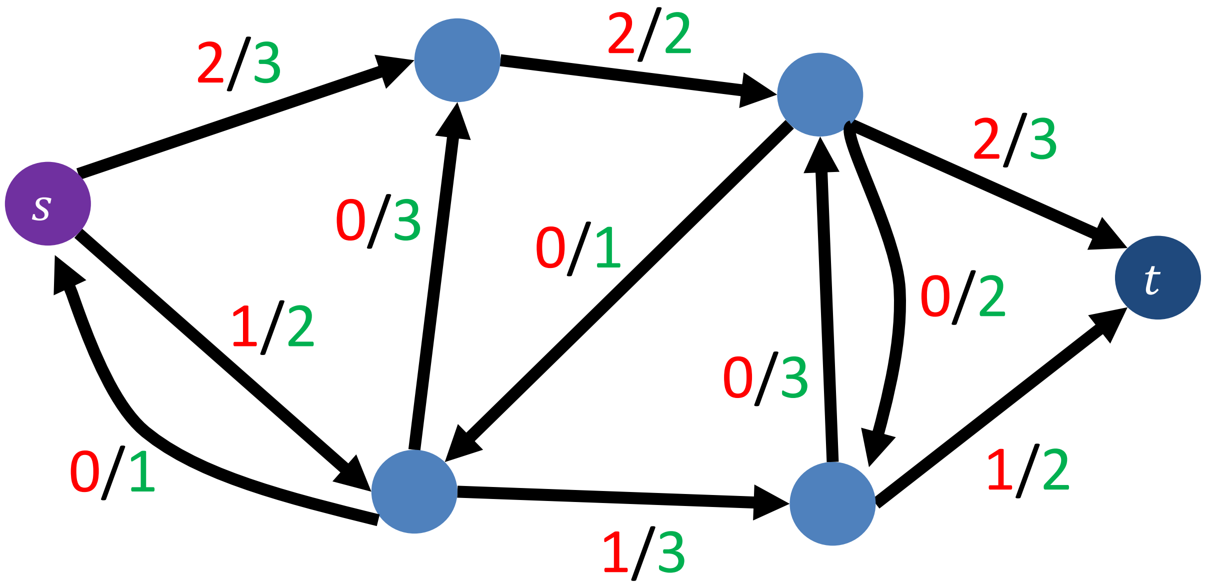


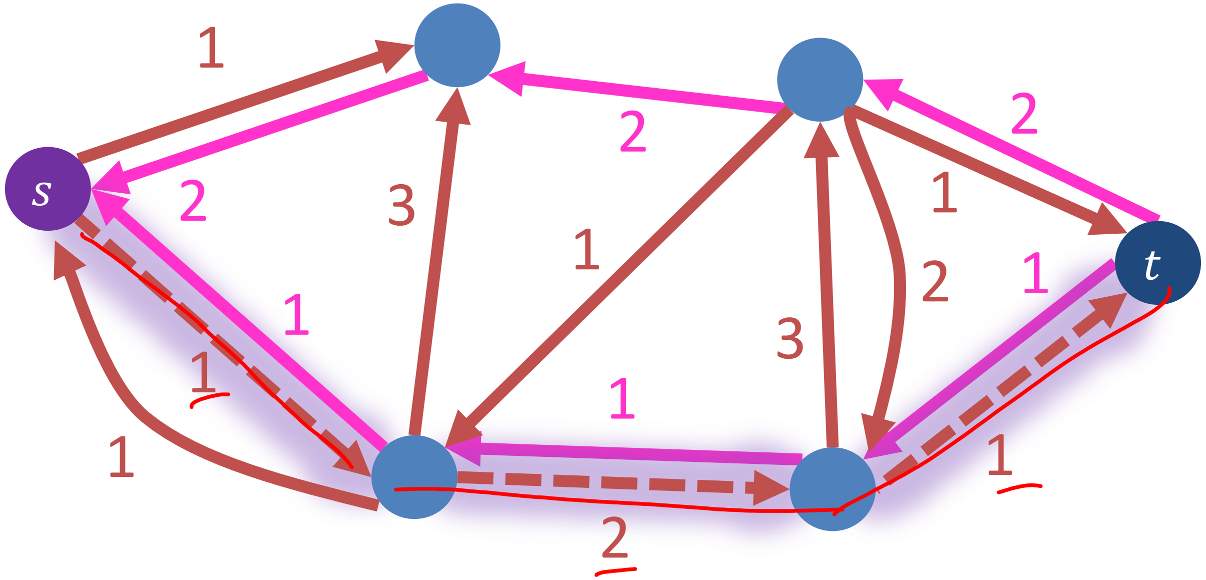
Increase flow by 1 unit



Residual graph G_f

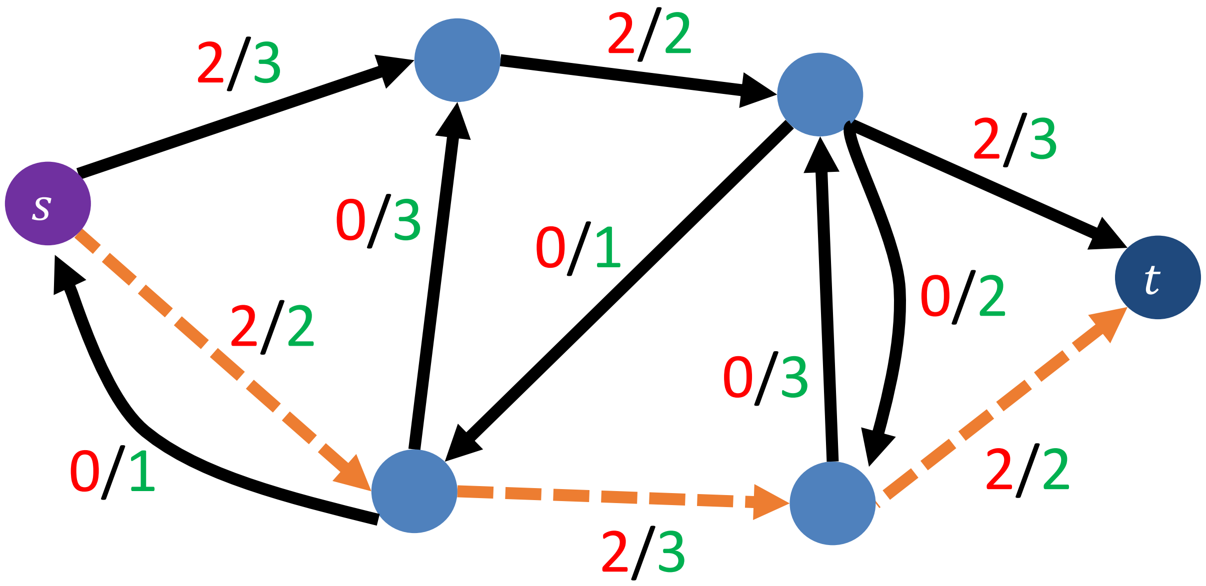
Ford-Fulkerson Example



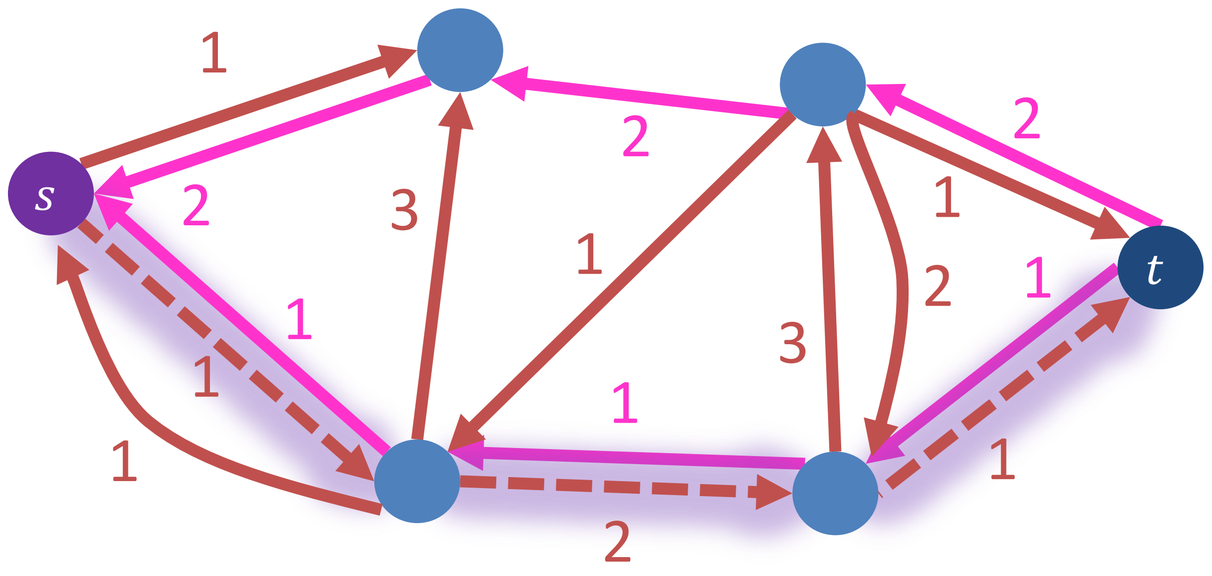


Residual graph G_f

Ford-Fulkerson Example

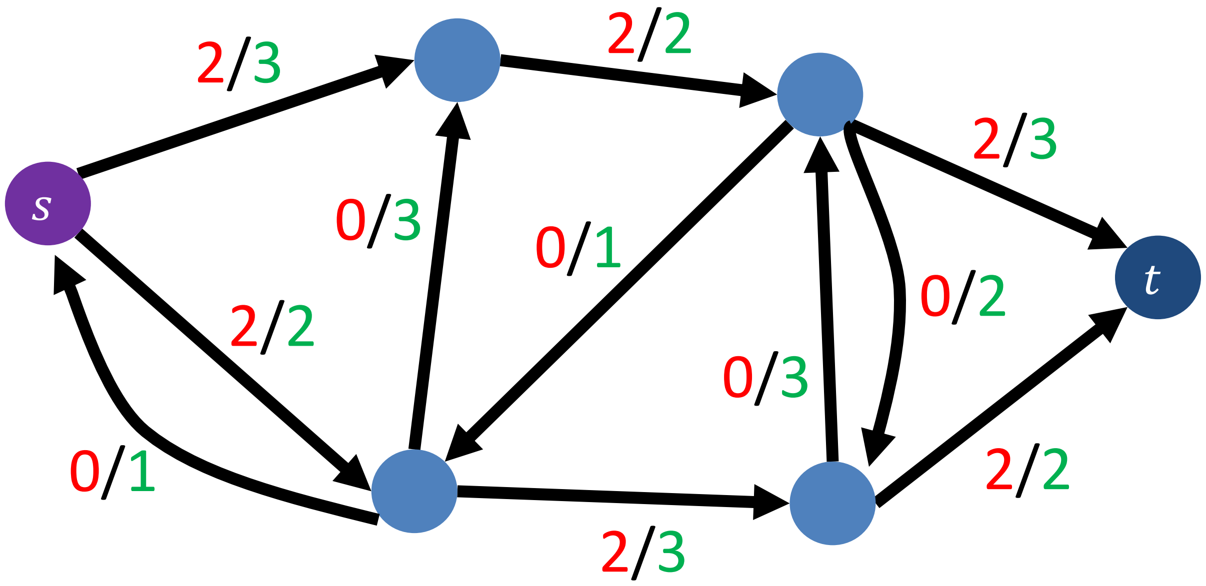


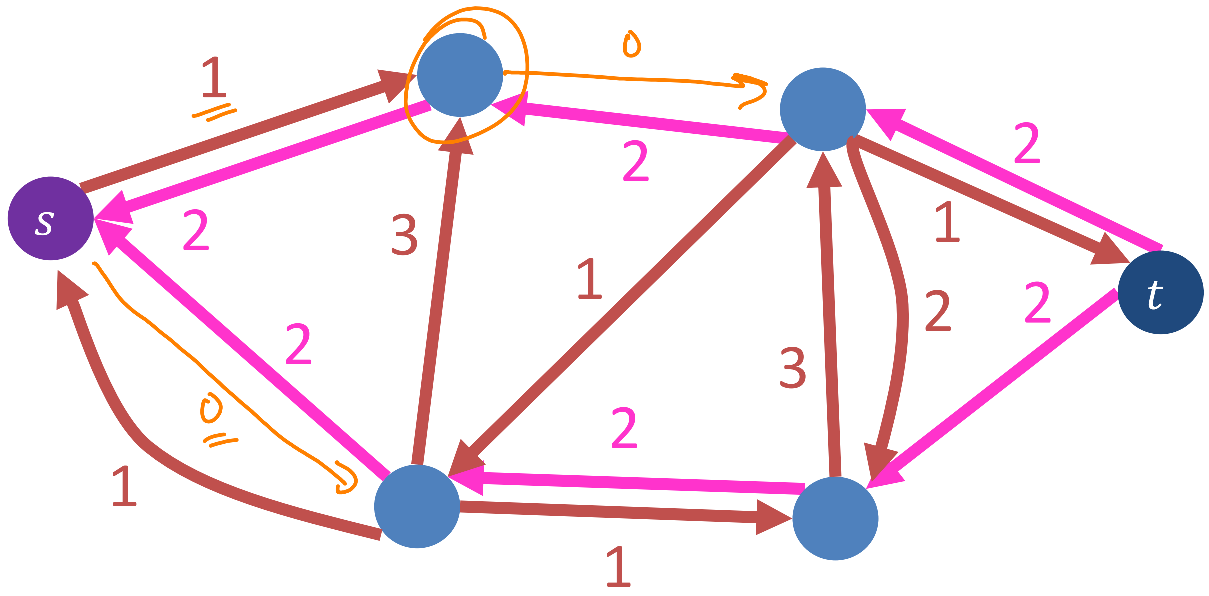
Increase flow by 1 unit



Residual graph G_f

Ford-Fulkerson Example

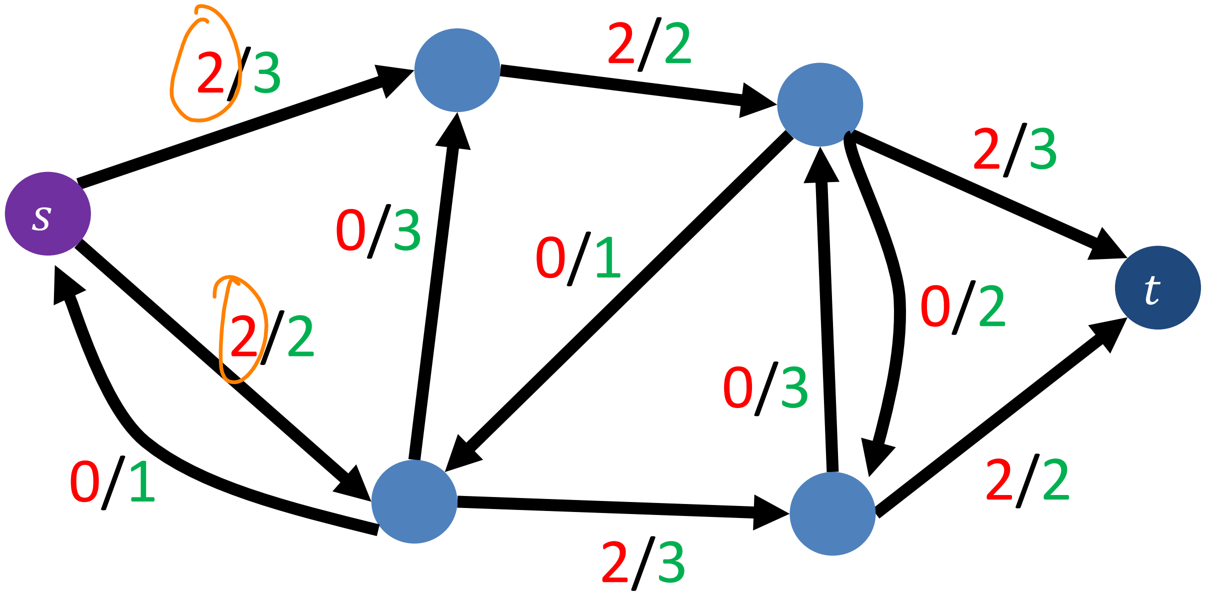


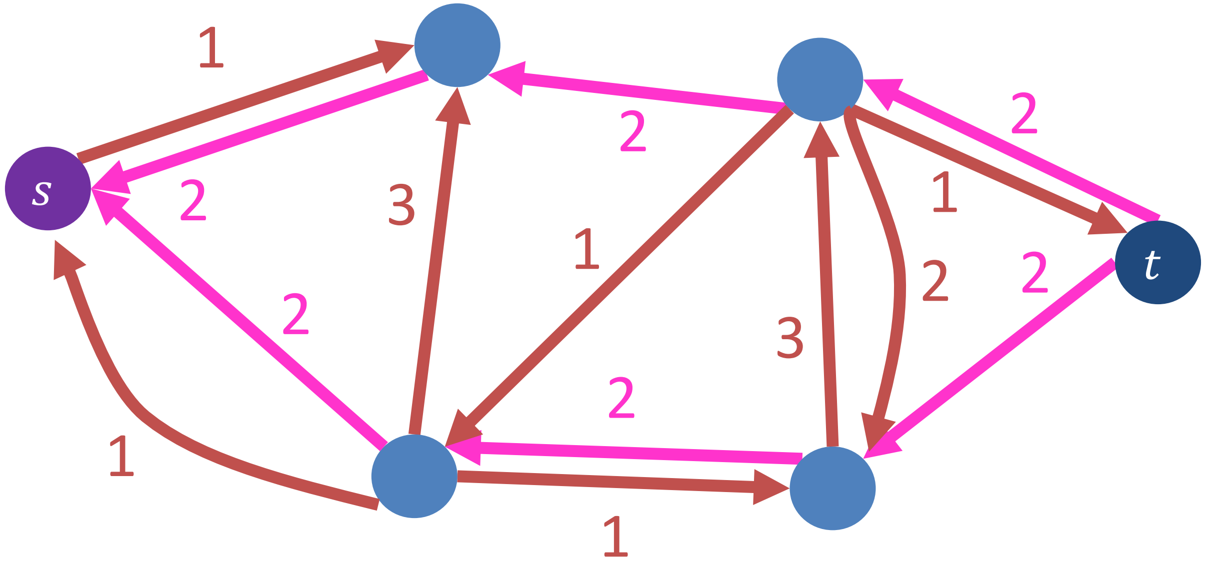


Residual graph G_f

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: BFS

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

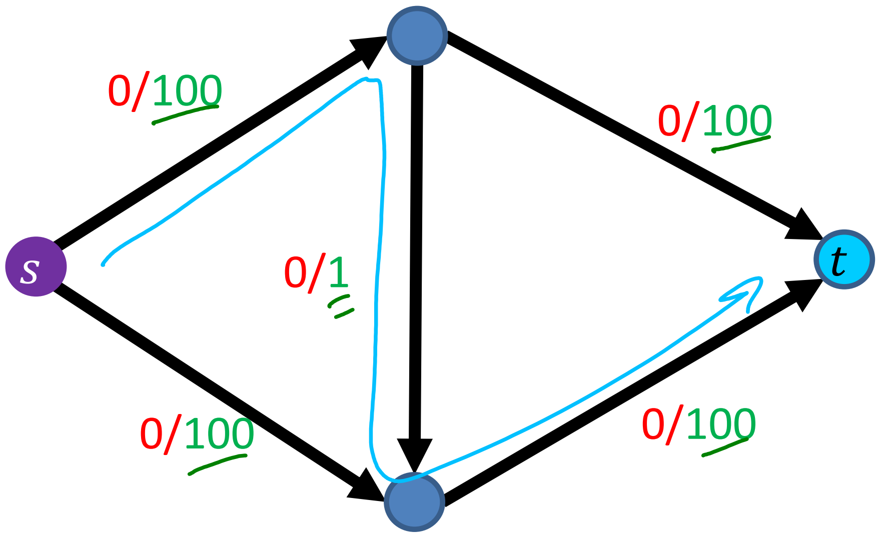
Time to find an augmenting path: BFS: $\Theta(V + E)$

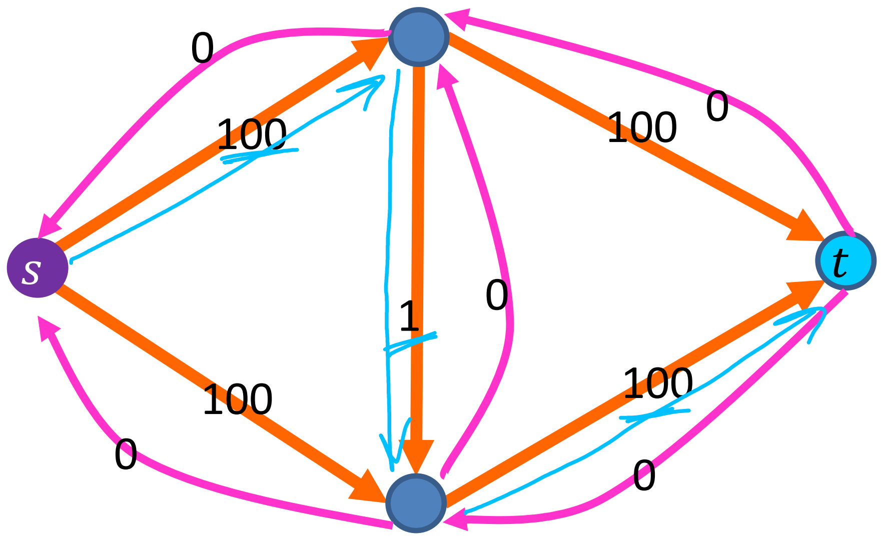
Number of iterations of While loop: $|f|$

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

Why might we loop $|f|$ times?

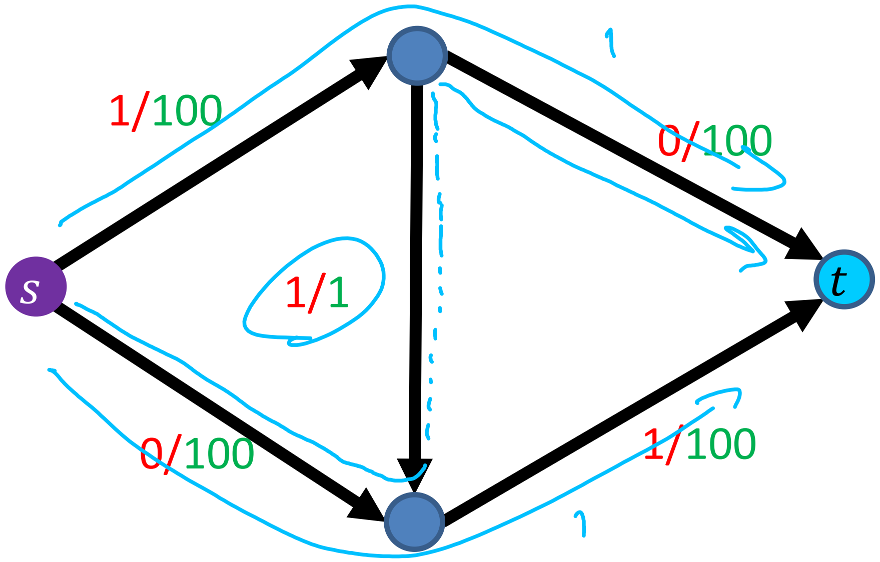
- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u, v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

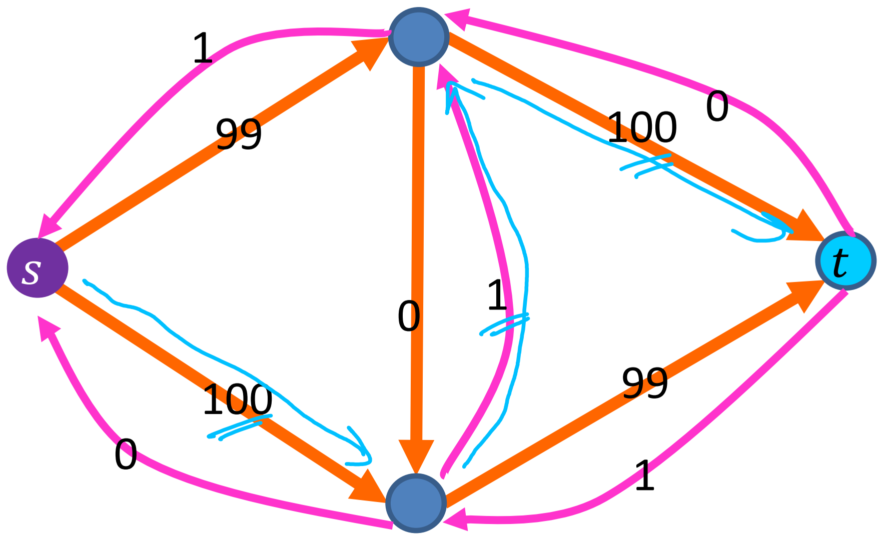




Why might we loop $|f|$ times?

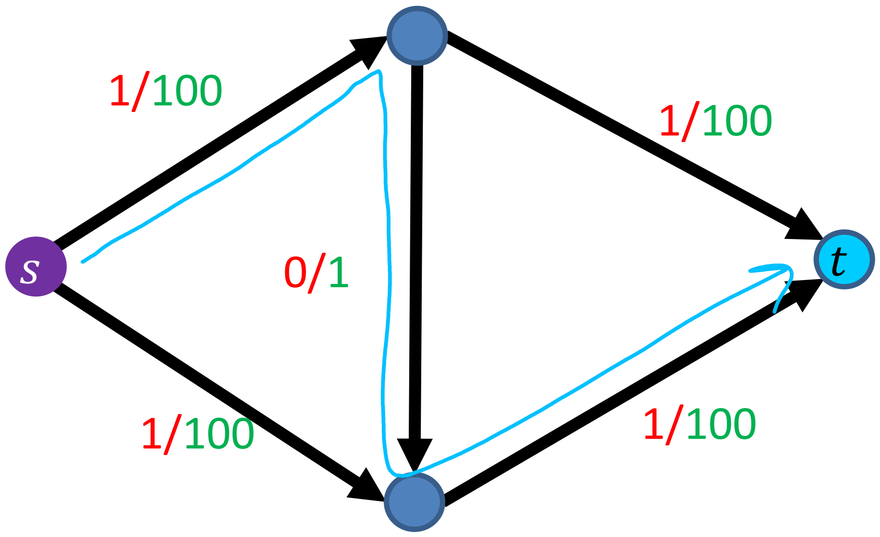
- Initialize $f(e) = 0$ for all $e \in E$
- Construct the residual network G_f
- While there is an augmenting path p in G_f:
  - Let $c = \min_{u,v \in p} c_f(u,v)$
  - Add c units of flow to G based on the augmenting path p
  - Update the residual network G_f for the updated flow

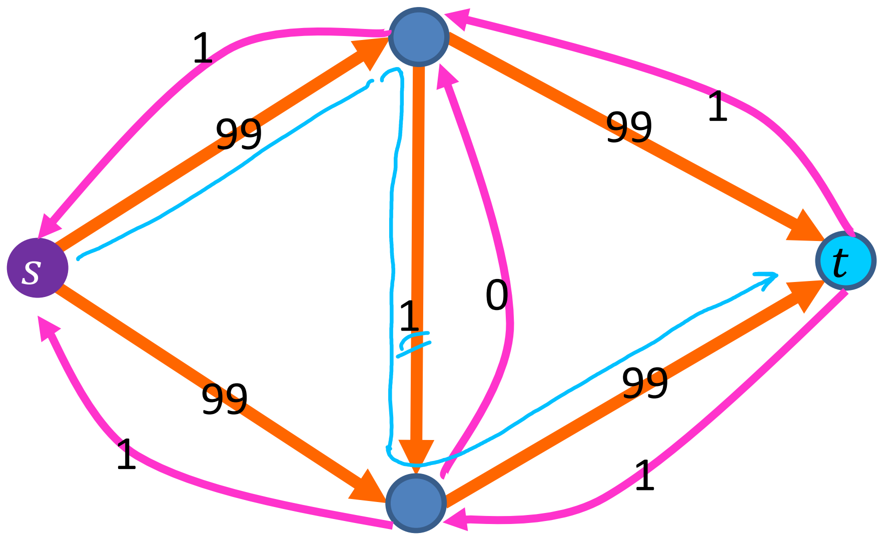




Why might we loop $|f|$ times?

- 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



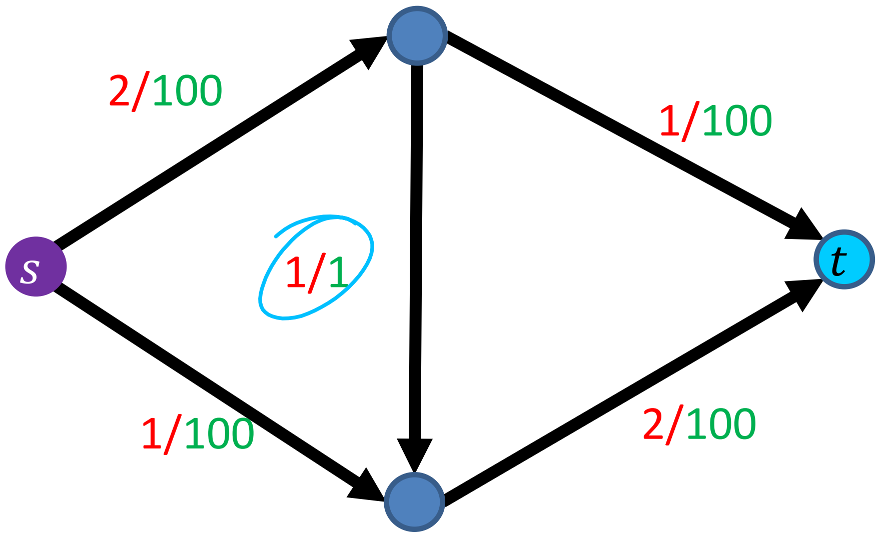


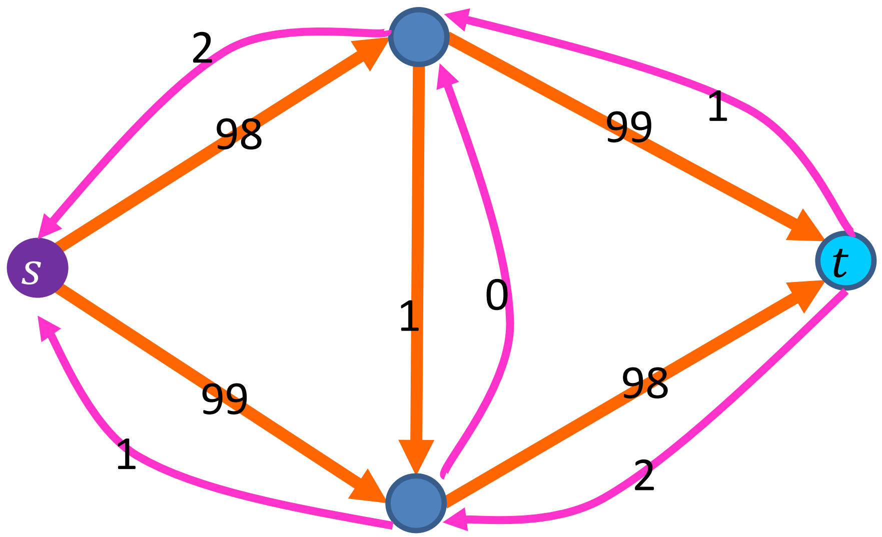
Why might we loop $|f|$ times?

- 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:** $\Theta(\min(|E||f^*|, |V||E|^2)) = O(|V||E|^2)$

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: $O(|V| \cdot |E|)$
  - Let $c = \min_{u,v \in p} c_f(u, v)$ $|E|$
  - 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:** $\Theta(\min(|E||f^*|, |V||E|^2))$

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

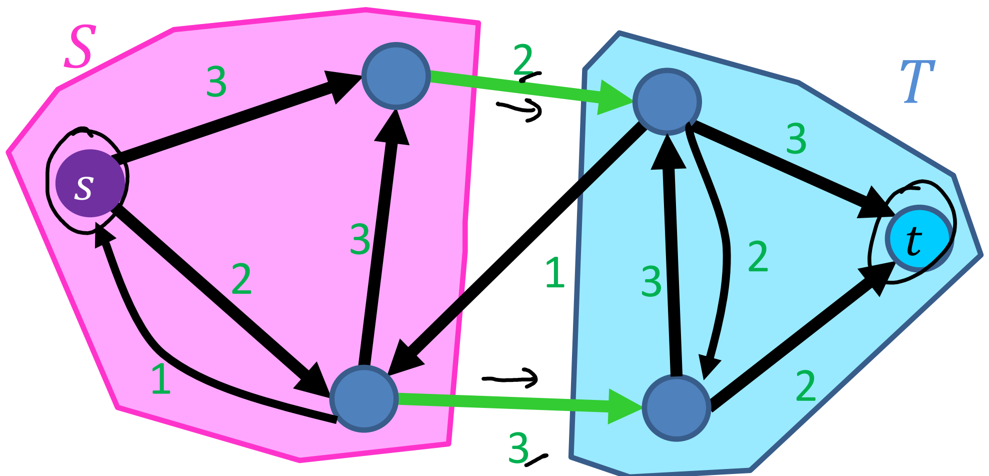
Proof: See CLRS (Chapter 26.2)

Showing Correctness of Ford-Fulkerson

min cut

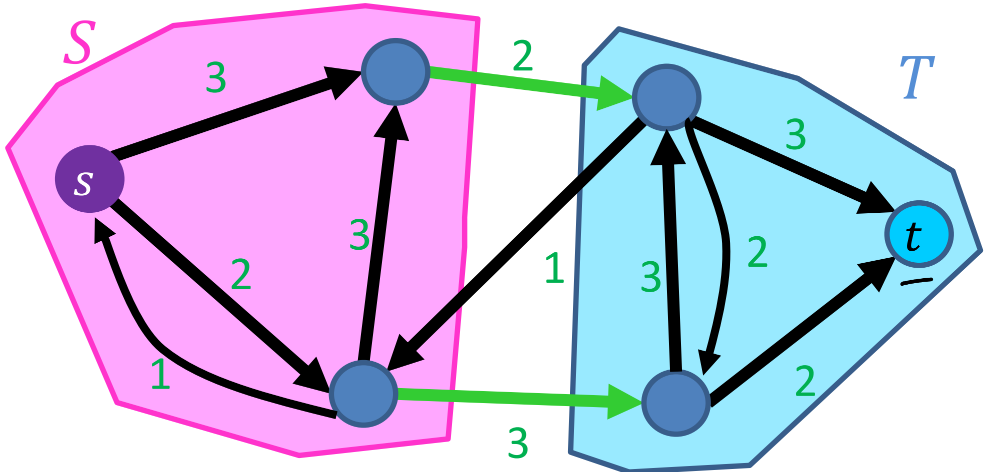
- Consider cuts which separate s and t
  - Let $s \in S$, $t \in T$, s.t. $V = S \cup T$
- Cost of cut $(S, T) = ||S, T||$
  - Sum **capacities** of edges which go from S to T – minimize
  - This example: 5

$s \longrightarrow t$



Maxflow≤MinCut

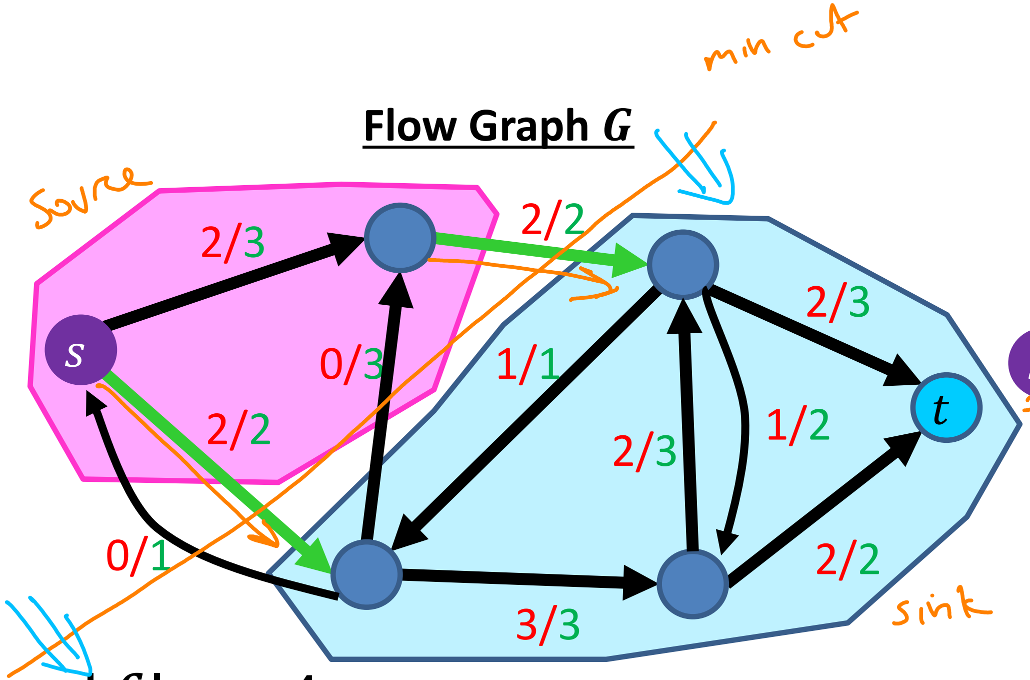
- Max flow upper bounded by any cut separating s and t
- Why? "Conservation of flow"
  - All flow exiting s must eventually get to t
  - To get from s to t, all "tanks" must cross the cut
- Conclusion: If we find the minimum-cost cut, we've found the maximum flow
  - $\max_f |f| \leq \min_{S,T} ||S,T||$

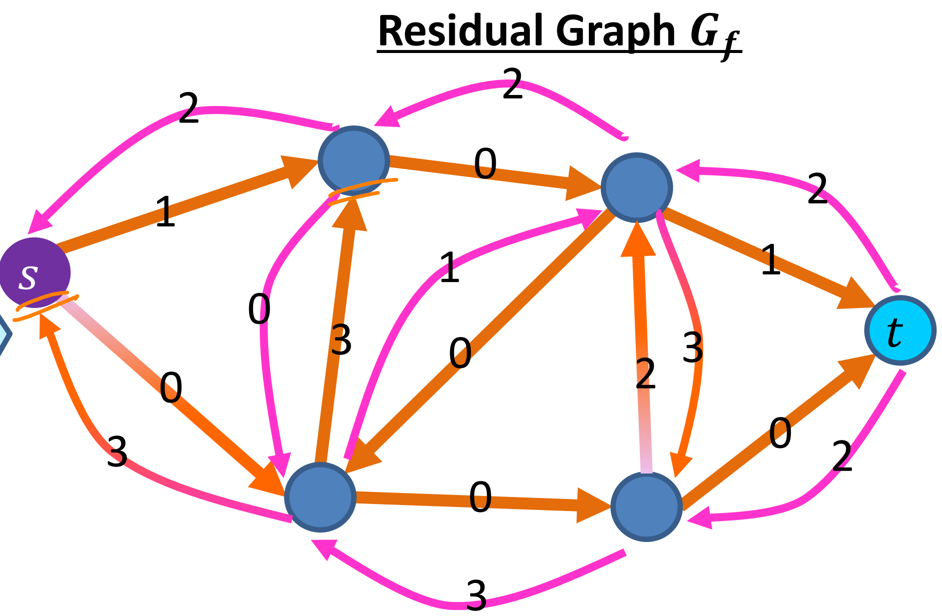


Maxflow/Mincut Theorem

- To show Ford-Fulkerson is correct:
  - Show that when there are no more augmenting paths, there is a cut with cost equal to the flow
- Conclusion: the maximum flow through a network matches the minimum-cost cut
  - $\max_f |f| = \min_{S,T} ||S,T||$
- Duality
  - When we've maximized max flow, we've minimized min cut (and vice-versa), so we can check when we've found one by finding the other

Example: Maxflow/Mincut





$$|f| = 4$$

$$||S, T|| = 4$$

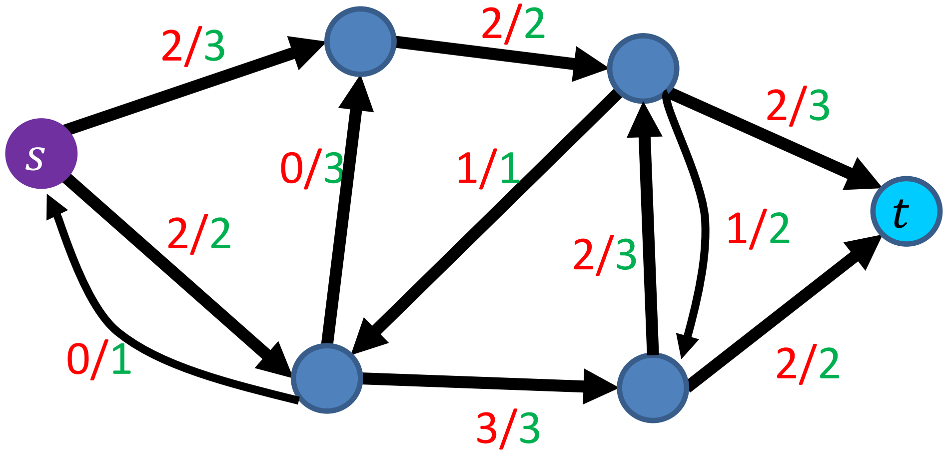
No Augmenting Paths

Idea: When there are no more augmenting paths, there exists a cut in the graph with cost matching the flow

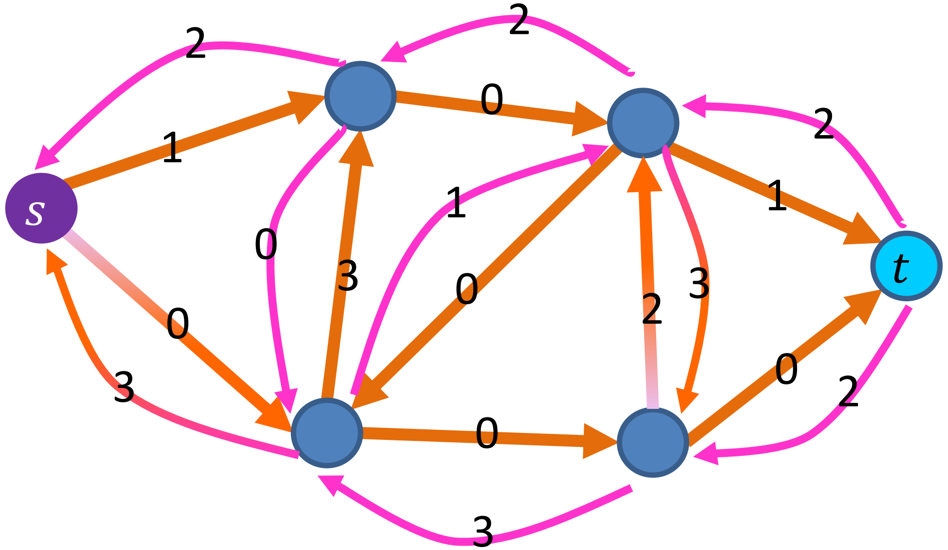
Proof: Maxflow/Mincut Theorem

- If $|f|$ is a max flow, then G_f has no augmenting path
  - Otherwise, use that augmenting path to "push" more flow

Flow Graph G

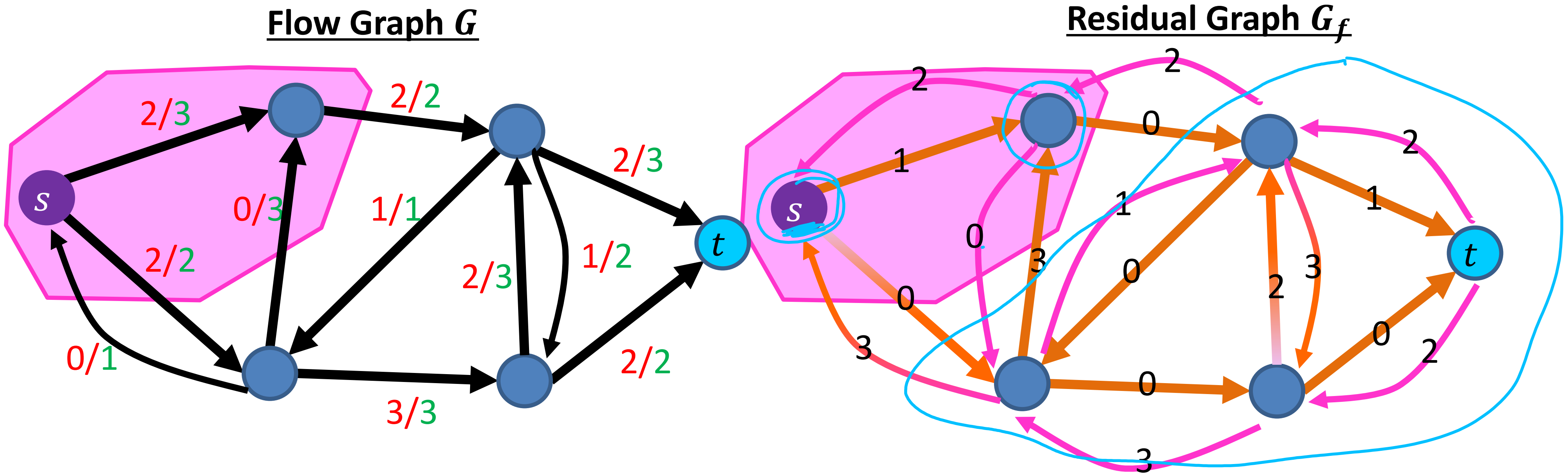


Residual Graph G_f



Proof: Maxflow/Mincut Theorem

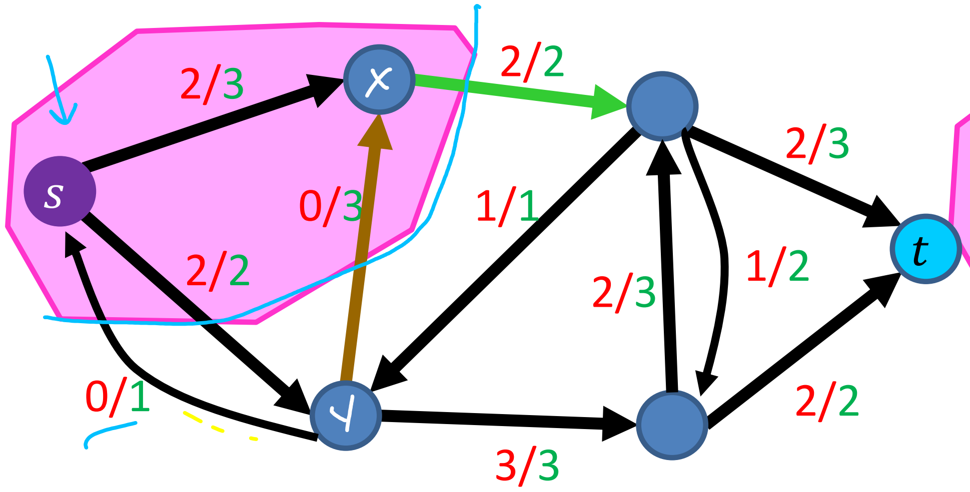
- If $|f|$ is a max flow, then G_f has no augmenting path
  - Otherwise, use that augmenting path to "push" more flow
- Define S = nodes reachable from source node s by positive-weight edges in the residual graph
  - $T = V - S$
  - S separates s , t (otherwise there's an augmenting path)



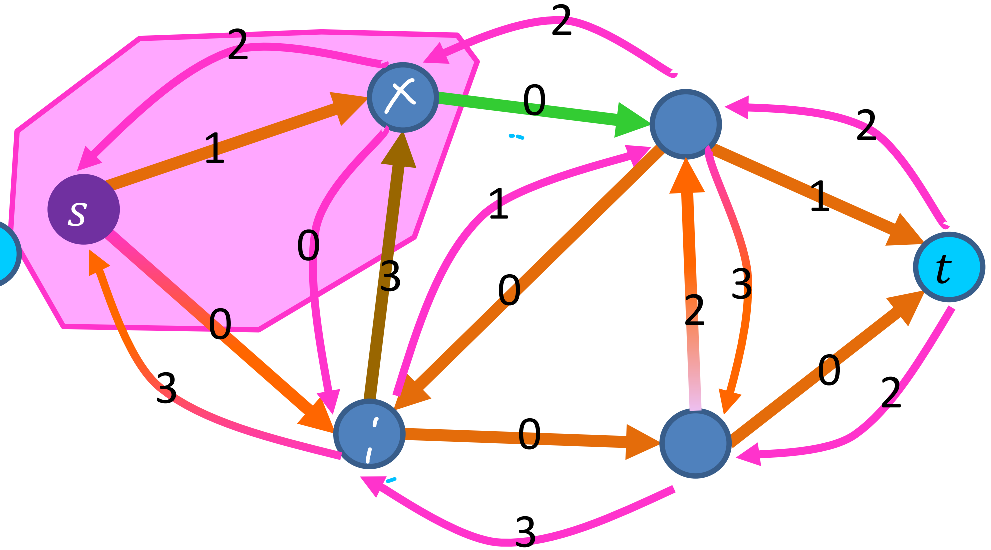
Proof: Maxflow/Mincut Theorem

- To show: $||S, T|| = |f|$
  - Weight of the cut matches the flow across the cut
- Consider edge (u, v) with $u \in S$, $v \in T$
  - $f(u, v) = c(u, v)$, because otherwise $w(u, v) > 0$ in G_f, which would mean $v \in S$
- Consider edge (y, x) with $y \in T$, $x \in S$
  - $f(y, x) = 0$, because otherwise the back edge $w(y, x) > 0$ in G_f, which would mean $\cancel{x}$ y $\in S$

Flow Graph G



Residual Graph G_f



Proof Summary

1. The flow $|f|$ of G is upper-bounded by the sum of capacities of edges crossing any cut separating source s and sink t

   $s \longrightarrow t$

2. When Ford-Fulkerson terminates, there are no more augmenting paths in G_f

3. When there are no more augmenting paths in G_f then we can define a cut $S =$ nodes reachable from source node s by positive-weight edges in the residual graph

4. The sum of edge capacities crossing this cut must match the flow of the graph

5. Therefore this flow is maximal

Max flow = Min Cut

Other Maxflow algorithms

- **Ford-Fulkerson**
  - $\boldsymbol{\Theta(E|f|)}$
- **Edmonds-Karp**
  - $\boldsymbol{\Theta(E^2V)}$
- Push-Relabel (Tarjan)
  - $\Theta(EV^2)$
- Faster Push-Relabel (also Tarjan)
  - $\Theta(V^3)$