Analysis and Synthesis of Algorithms
Design of Algorithms

Maximum Flow Algorithms

Introduction
Ford-Fulkerson & Edmonds-Karp Algorithms

Outline

- Maximum Flow in Graphs
  - Motivation
  - Definitions & Properties
  - Ford-Fulkerson Method
  - Maximum-Flow Minimum-Cut Theorem
  - Ford-Fulkerson Algorithm Analysis
  - Edmonds-Karp Algorithm
  - Edmonds-Karp Algorithm Analysis

Problem: Water Supply to Lisbon

- Determine the Maximum Water Volume (per second) that can reach Lisbon from the "Castelo do Bode" Dam
  - There is a network of pipes for water transport
  - Each pipe has a maximum capacity (rate) of K cubic meters/sec

- **Goal:** Find an efficient algorithm to solve this problem

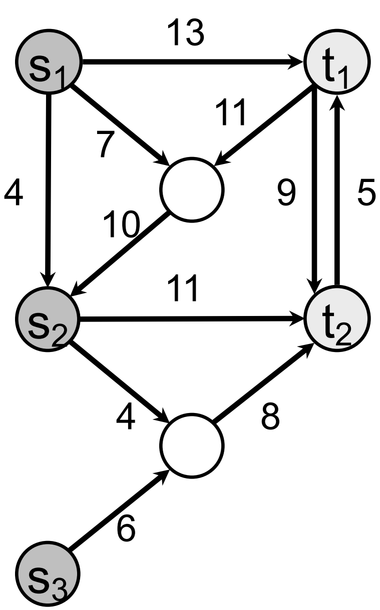
- Other Application Domains:
  - Water, Oil or Gas
  - Containers
  - Electricity
  - Bytes
  - …

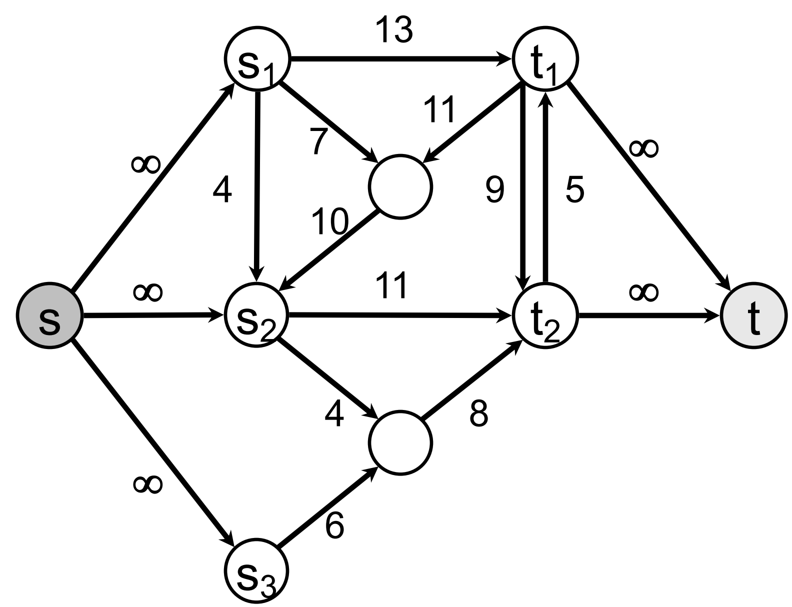
Maximum Flows in Graphs

- Given a Directed Graph G=(V, E):
  - *Source* node *s* and a *Sink* node *t*
  - Each edge (u,v) has a non-negative capacity c(u,v)
    - The edge capacity c(u,v) indicates the maximum value of flow that is possible to send from u to v through the edge (u,v)
    - Compute the Maximum Value of "flow" that can be
      - "Pushed" from the Source to the Sink
      - Subject to Edge Capacity Constraints
  - Ignore self-loops and multi edges (without loss of generality)

Multiple Sources and Sinks

- For Networks with Multiple Sources and/or Sinks:
  - Define a Super-Source connected to all Sources
  - Define a Super-Sink connected to all Sinks
  - Infinite capacity between super-source/sink and sources/sinks





Augment the Graph to make it with one Source and one Sink!

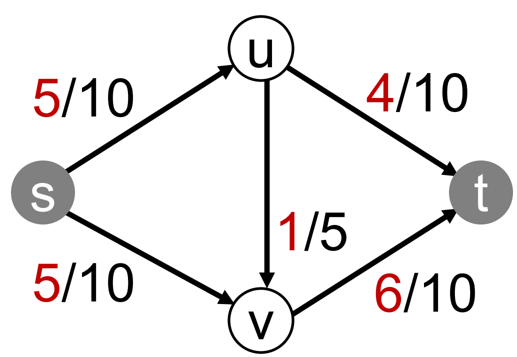
Maximum Flows - Definitions

- A **Flow Network** G = (V, E) is a directed graph in which each edge (u,v) has a capacity **c(u, v) ≥ 0**
  - If (u,v) ∉ E, then c(u,v) = 0
- Two Special Nodes: **Source *s*** and **Sink *t***
- All Nodes of G belong to a path from ***s*** to ***t***
  - Connected graph, |E| ≥ |V| - 1

- A **Flow** of G = (V, E) is a function f : V × V → R such that:
  - f(u, v) ≤ c(u, v) for u, v ∈ V (capacity constraint)
  - f(u, v) = - f(v, u) for u, v ∈ V (symmetry)
  - for u ∈ V - { *S*, *T* }: $\sum_{v \in V} f(u,v) = 0$ (flow invariant/conservation)

Maximum Flows - Definitions

- **Flow Value:** $F = \sum_{v \in V} f(s, v)$
- **Maximum Flow Problem:**
  – Given the flow network G with Source **s** and Sink **t**, compute the maximum flow value from **s** to **t**.
- Example:

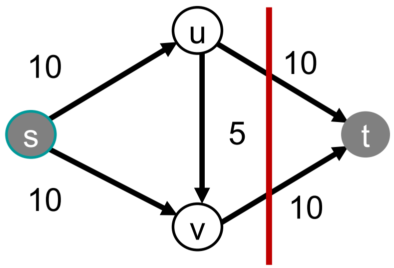


Flow Value: 10
Maximum Flow: 20

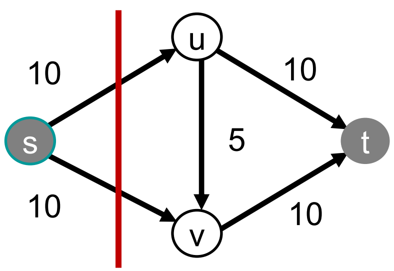
Maximum Flows - Observations

Q: What is the Capacity of a **Cut** that separates s from t?

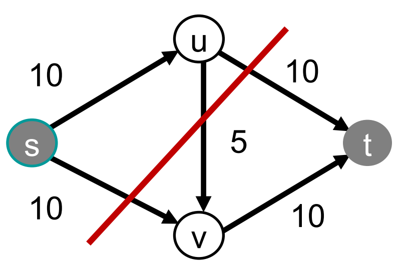
Considering edges that cross from source to sink only. **Why?**



Cut Capacity: 20



Cut Capacity: 20



Cut Capacity: 25

Minimum Cut Capacity: 20

Maximum Flows — Properties

- Given two sets of Nodes X and Y: $f(X,Y) = \sum_{x \in X} \sum_{y \in Y} f(x,y)$

- For flow network G = (V, E); f a flow in G, and sets of nodes X, Y, Z $\subseteq$ V:
  - $f(X,X) = 0$ (term cancellation)
  - $f(X,Y) = -f(Y,X)$ (symmetry)
  - If $X \cap Y = \varnothing$:
    - $f(X \cup Y,Z) = f(X,Z) + f(Y,Z)$ (summation expansion)
    - $f(Z,X \cup Y) = f(Z,X) + f(Z,Y)$ (summation expansion)

Lemma 26.1 from the book

Ford-Fulkerson Method

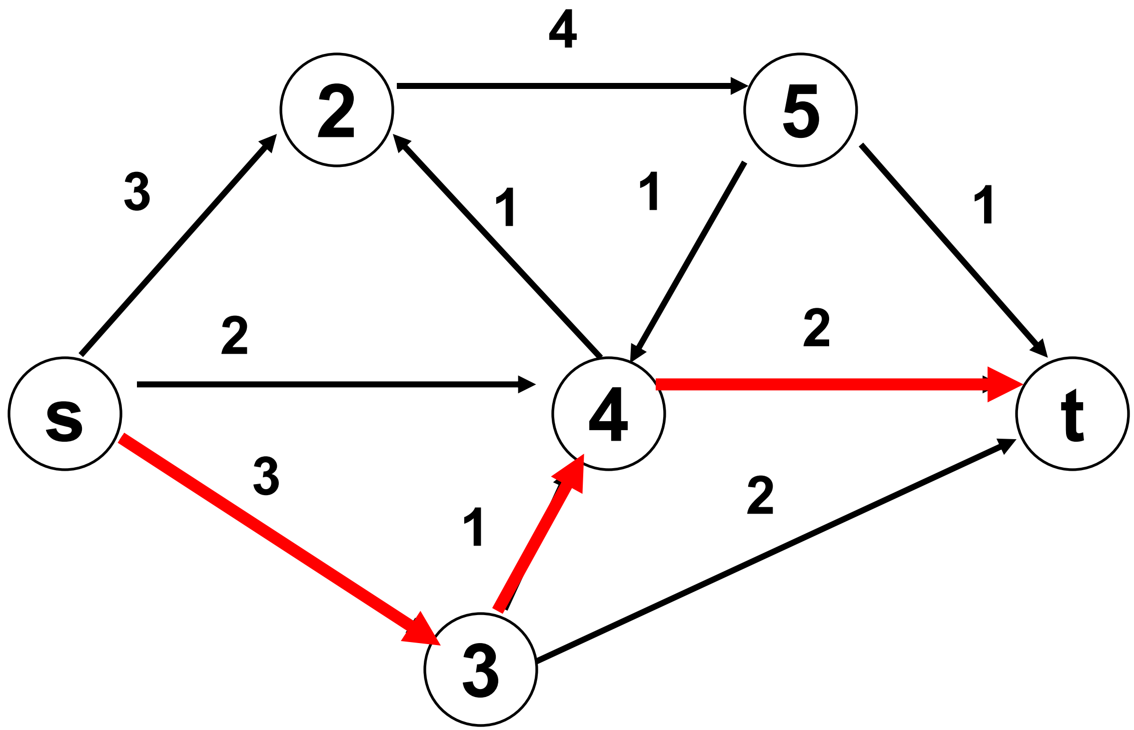
- Definitions:
  - Residual Network
  - Augmenting Paths
  - Cuts on Flow Networks
    - Maximum-Flow / Minimum Cut Theorem

- Ford-Fulkerson Algorithm
  - Greedy Algorithm
  - Complexity
    - Convergency Problems

Ford-Fulkerson Method Outline

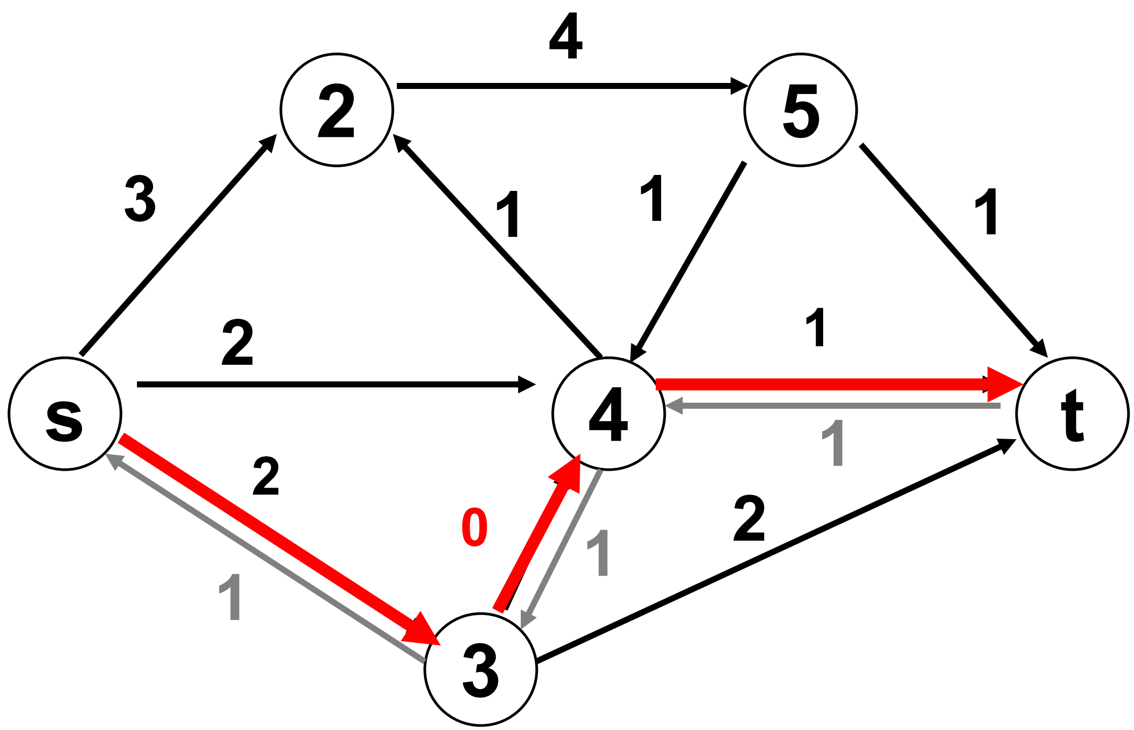
```
Ford-Fulkerson-Method(Graph G, node s, node t)
    initialize flow f to 0;
    while (exists an augmenting path P) do
        increase flow along P;
        update residual network;
return f;
```

Ford-Fulkerson Max Flow



Find any s-t path in G(x)

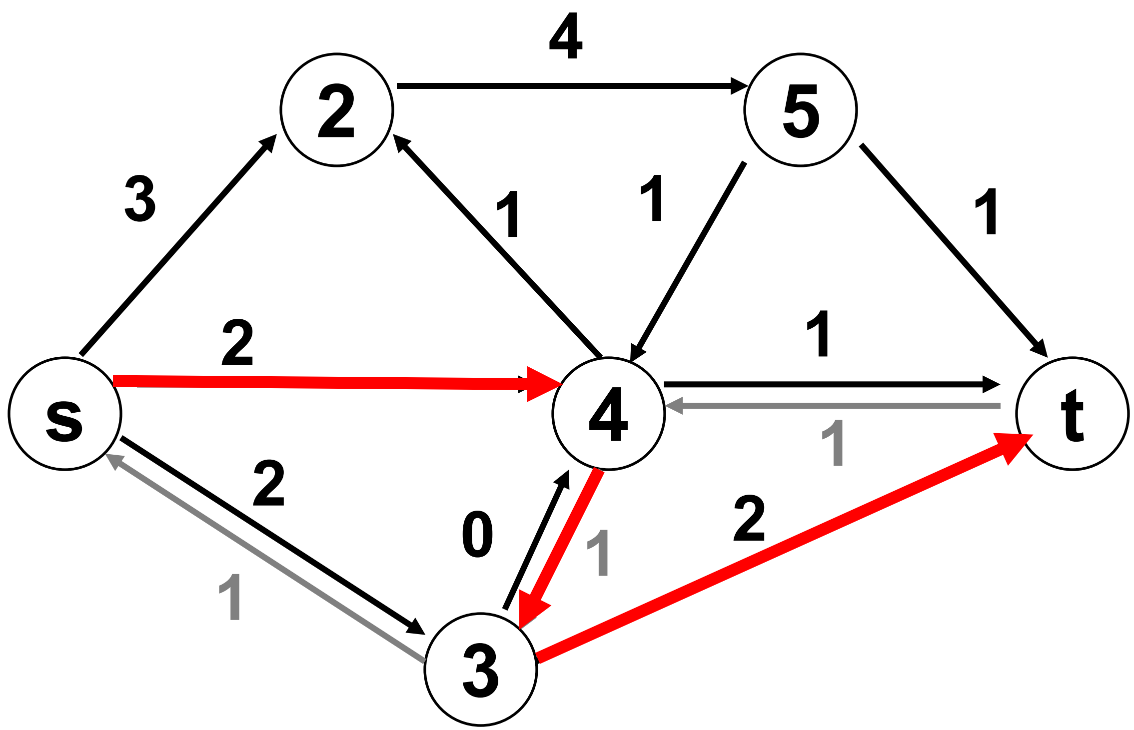
Ford-Fulkerson Max Flow



Determine the capacity Δ of the path

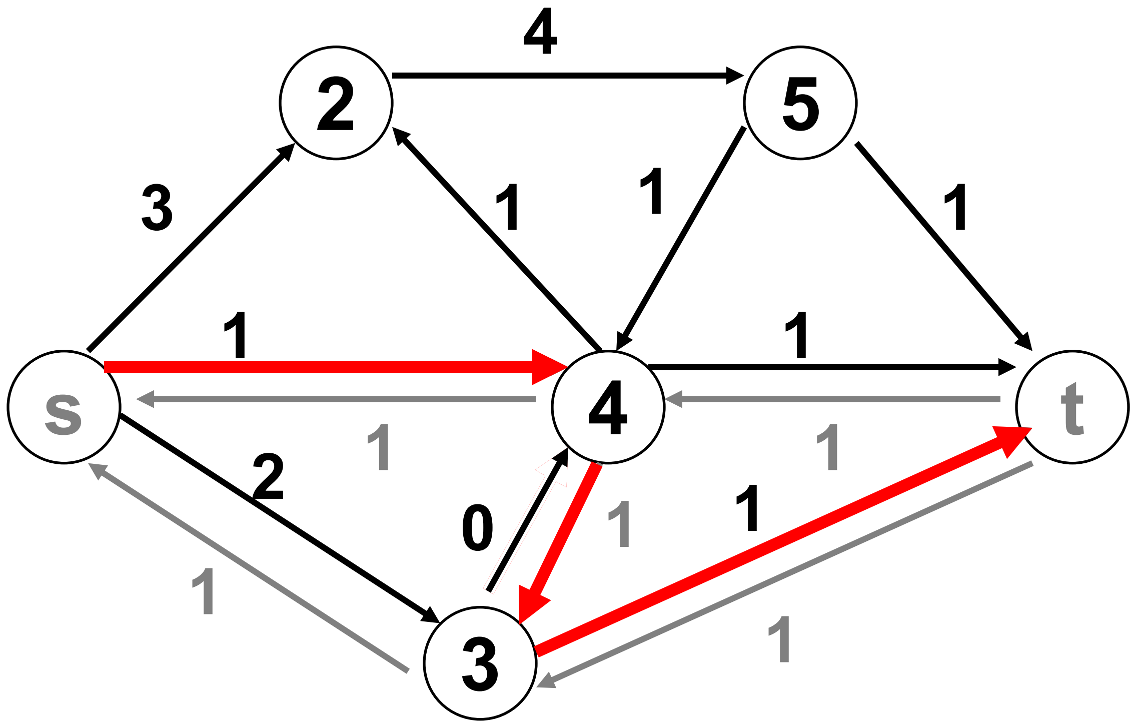
Send Δ units of flow along the path (update flow)
update residual capacities (reverse path)

Ford-Fulkerson Max Flow



Find any s-t path in G(x)

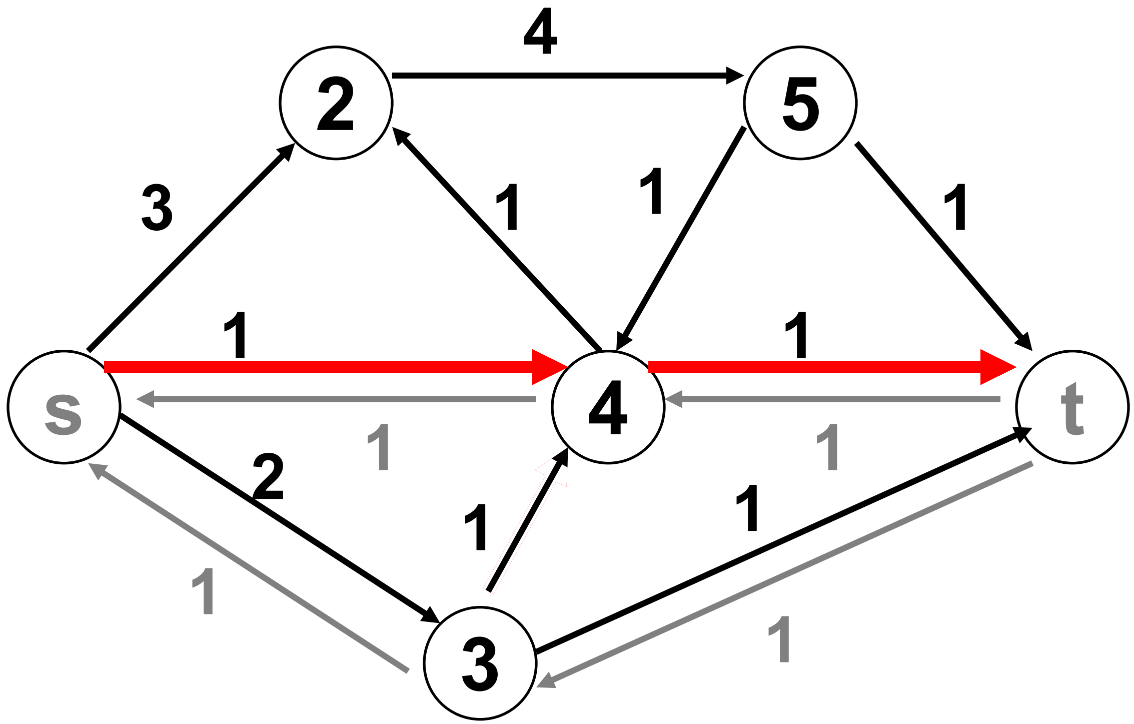
Ford-Fulkerson Max Flow



Determine the capacity Δ of the path.

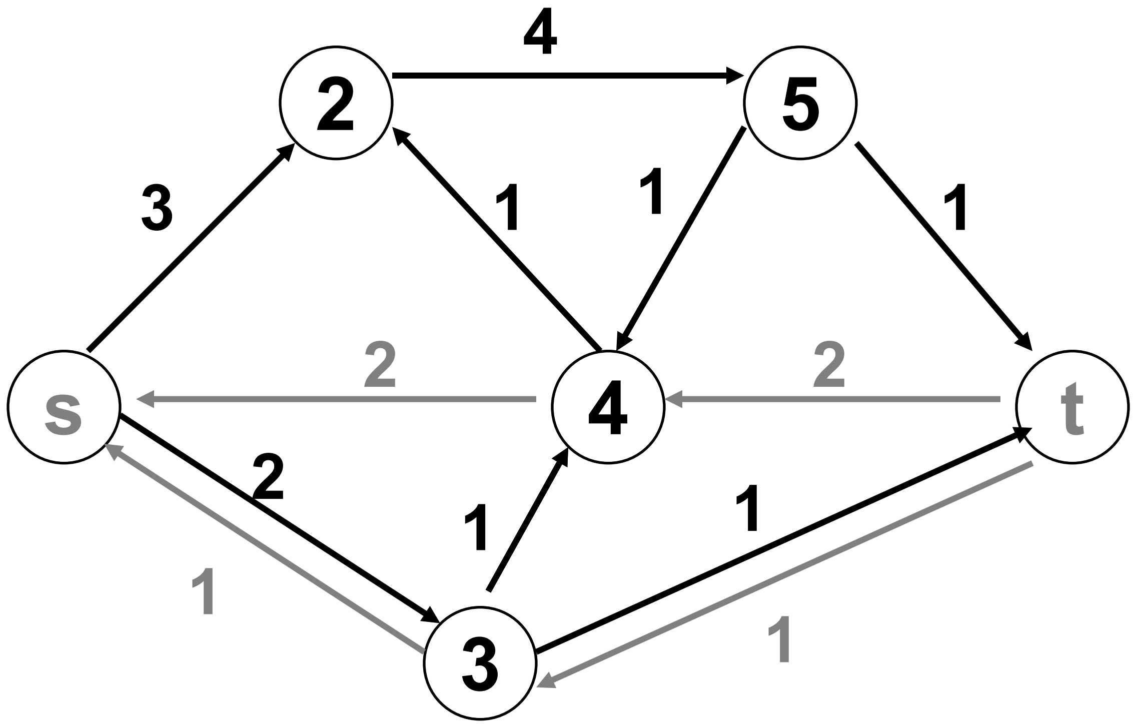
Send Δ units of flow in the path (update flow)
update residual capacities

Ford-Fulkerson Max Flow



Find any s-t path in G(x)

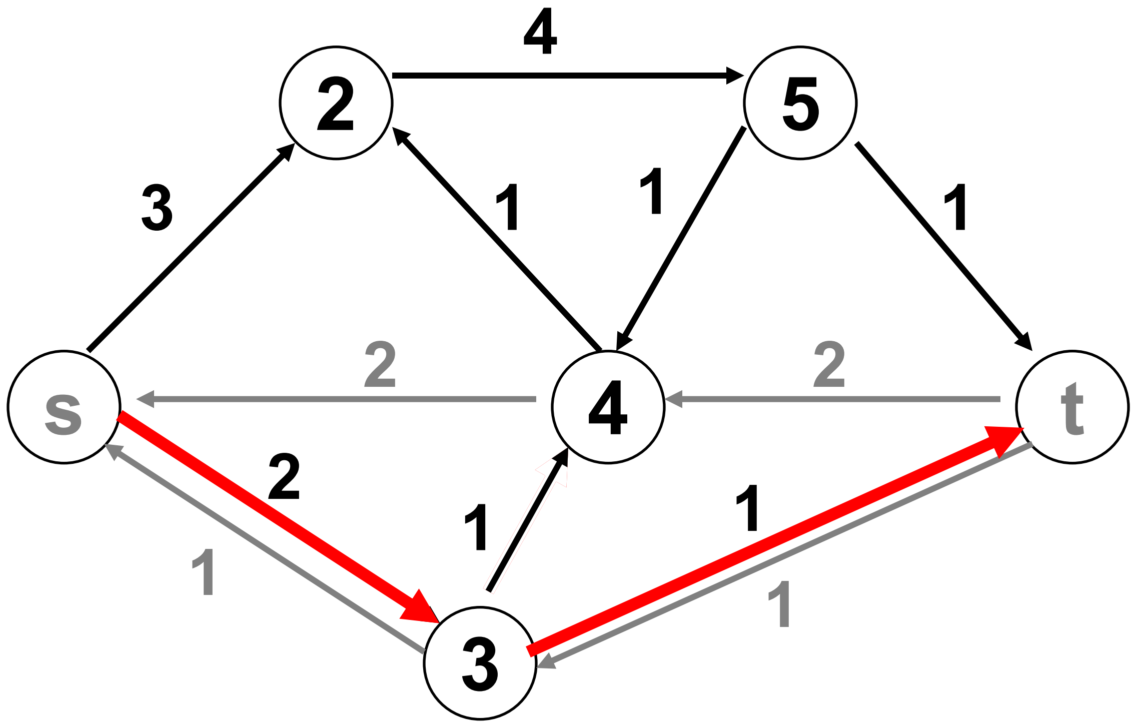
Ford-Fulkerson Max Flow



Determine the capacity Δ of the path

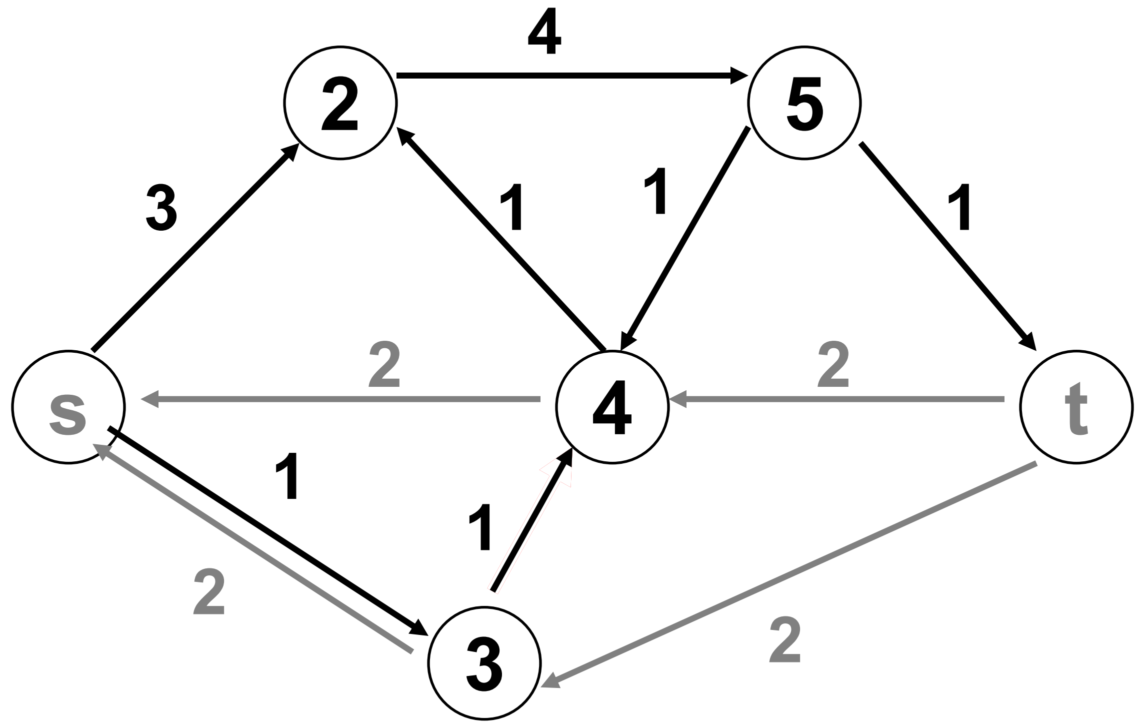
Send Δ units of flow in the path (update flow)
update residual capacities (reverse path)

Ford-Fulkerson Max Flow



Find any s-t path in G(x)

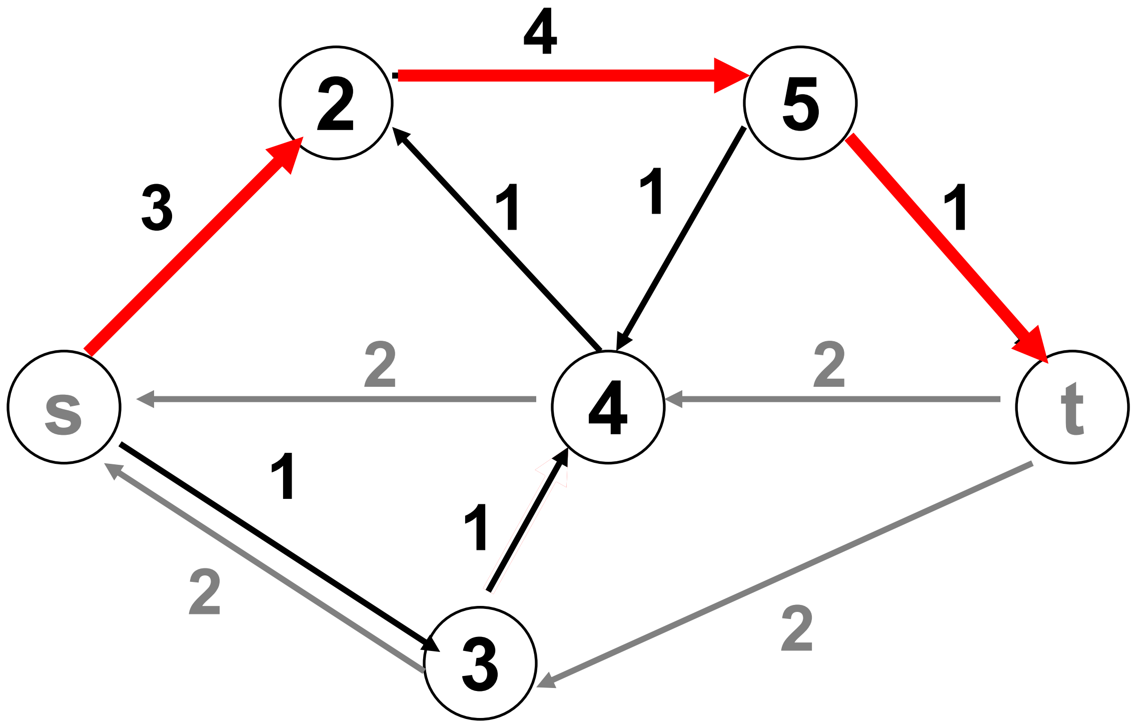
Ford-Fulkerson Max Flow



Determine the capacity Δ of the path

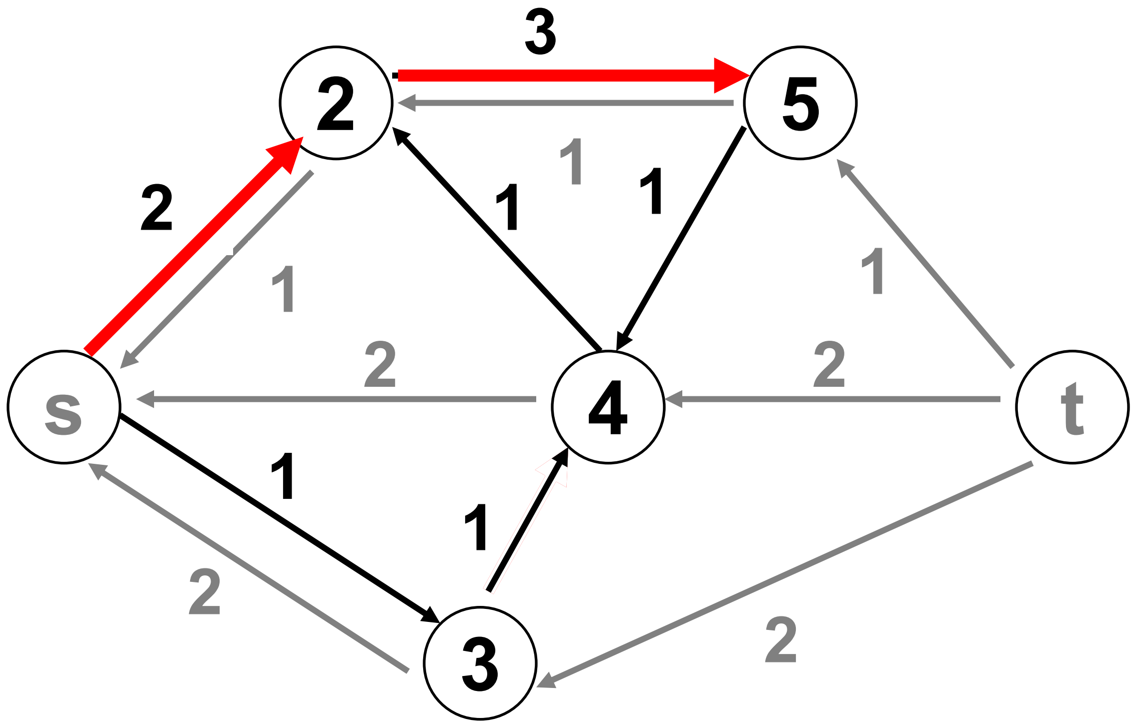
Send Δ units of flow in the path (update flow)
Update residual capacities (reverse path)

Ford-Fulkerson Max Flow



Find any s-t path in G(x)

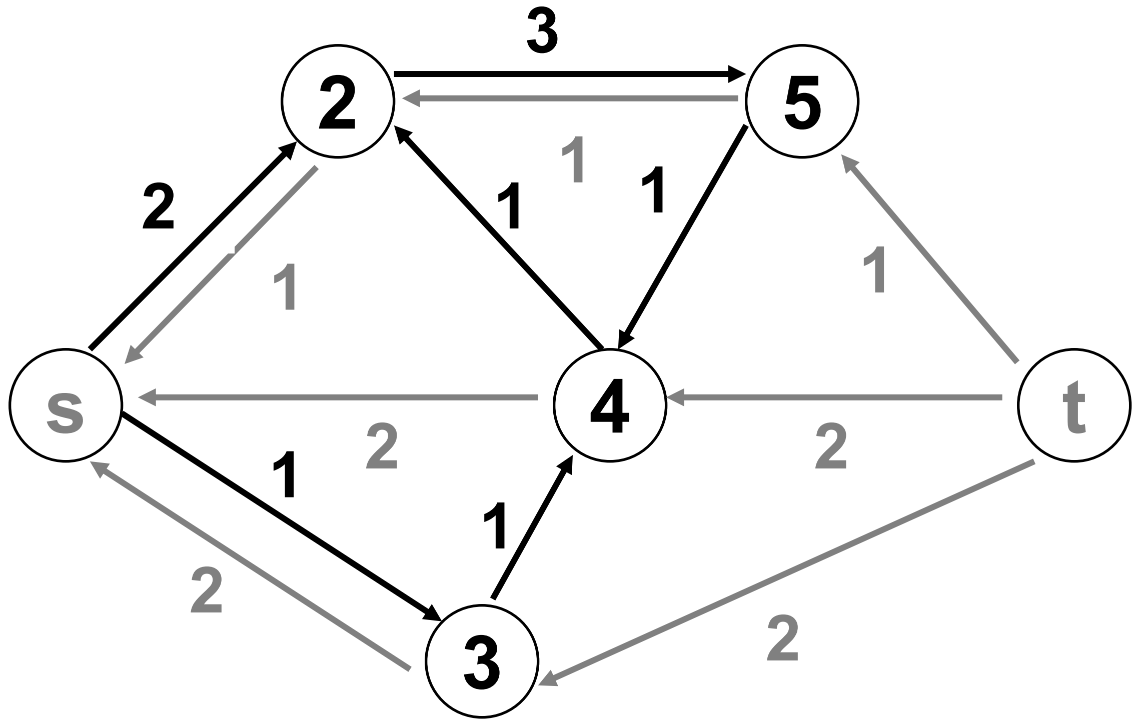
Ford-Fulkerson Max Flow



Determine the capacity Δ of the path

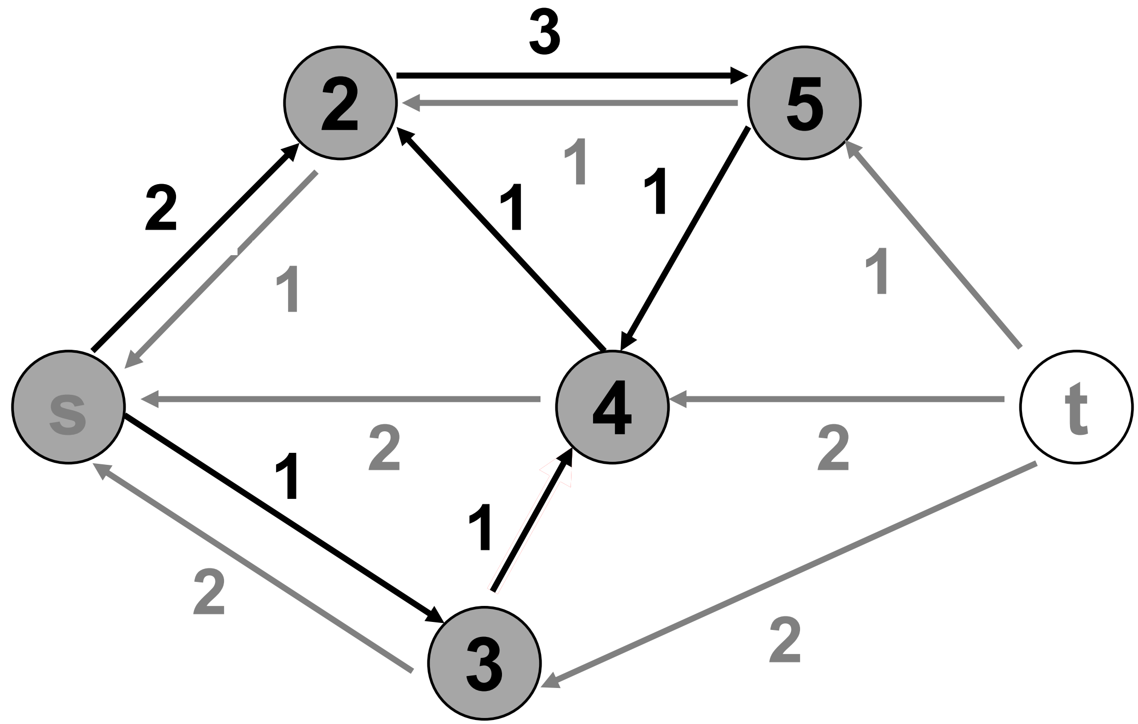
Send Δ units of flow in the path (update flow)
Update residual capacities (reverse path)

Ford-Fulkerson Max Flow



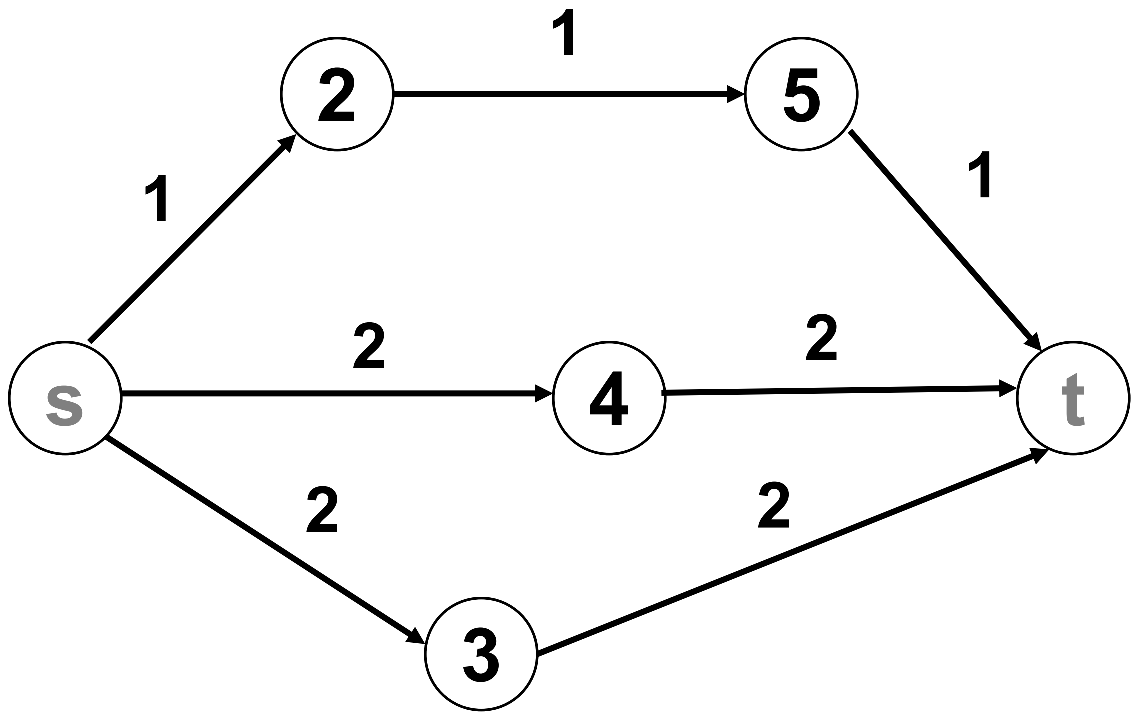
There is no s-t path in the residual network
We cannot reach node t
This flow is Optimal

Ford-Fulkerson Max Flow



These are the nodes that are reachable from node s

Ford-Fulkerson Max Flow



Here is the Optimal Flow = 5

Residual Network

- Given G = (V, E), a flow f, and u,v ∈ V

$$F = \sum_{v \in V} f(s, v)$$

  – **residual capacity** of (u,v) is the additional flow that is possible to send from u to v
    - $\mathbf{c_f(u,v) = c(u,v) - f(u,v)}$
  – **residual network** of G:
  – $G_f = (V, E_f)$, where $\mathbf{E_f = \{ (u,v) \in V \times V : c_f(u,v) > 0 \}}$
    - Each edge (residual) of G_f only allows positive flow

Residual Network (Cont.)

- Let $G = (V, E)$, f a flow, G_f residual network; f' a flow in G_f

- **Added Flow** $f + f'$ defined for each pair $u,v \in V$:
  - $(f + f')(u,v) = f(u,v) + f'(u,v)$
  - Value of the added flow $|f + f'| = |f| + |f'|$
  - Flow Properties are respected:
    - Capacity, symmetry and flow conservation
    - **Obs:** f' is defined in G_f and is a flow
  - Computation of the flow:

$$
\begin{aligned}
|f + f'| &= \sum_{v \in V} (f + f')(s,v) \\
&= \sum_{v \in V} \left(f(s,v) + f'(s,v)\right) \\
&= \sum_{v \in V} f(s,v) + \sum_{v \in V} f'(s,v) \\
&= |f| + |f'|
\end{aligned}
$$

Augmenting Paths

- Given G = (V, E) and the flow f
  - **Augmenting Path** *p*:
    - Simple path from *s* to *t* in residual network G_f
  - **Residual Capacity** of *p*:
    - $c_f(p) = \min \{ c_f(u,v) : (u,v) \text{ in } p \}$
  - $c_f(p)$ allows the definition of a flow f_p in G_f, $|f_p| = c_f(p) > 0$
  - $\mathbf{f' = f + f_p}$ is a flow in G, with value $|f'|=|f| + |f_p| > |f|$

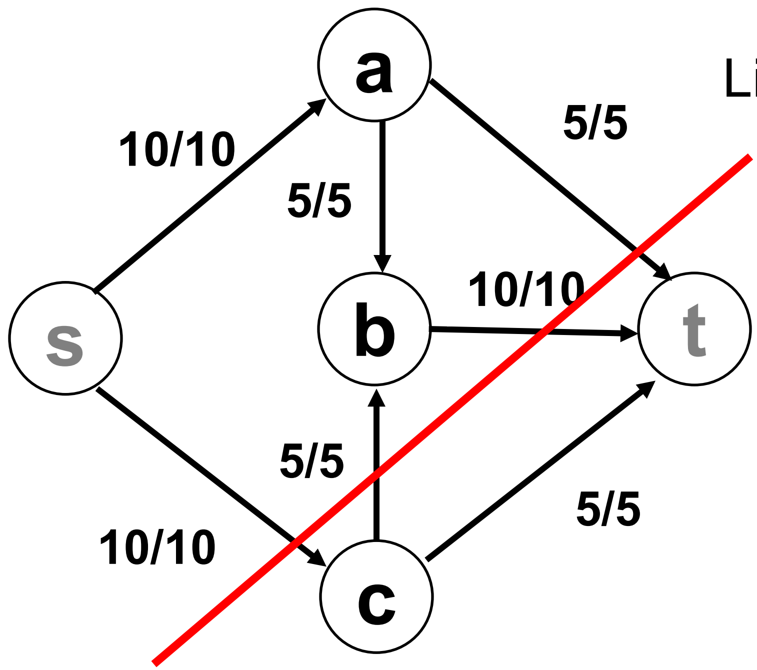
DEI DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Cuts and Flows in a Flow Network

- A **Cut** (S, T) of G = (V, E) is a partition of V in S and T = (V – S), such that s ∈ S and t ∈ T
  – **liquid flow of the cut** (S, T): $f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v)$
  – **cut capacity** (S, T): $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$
  – **Obs:** Cut includes only positive capacity edges; liquid flow also negative flows.

- If G = (V, E) has flow f, then the "liquid" flow throught the cut (S, T) is f(S,T) = |f|
  – T = (V-S); $f(S, T \cup S) = f(S,T) + f(S,S)$; $f(S,T) = f(S,V) - f(S,S)$
  – $f(S,T) = f(S,V) - f(S,S) = f(S,V) = f(s,V) + f(S - s,V) = f(s,V) = |f|$
    - **Obs:** for u ∈ S - s, f(u, V) = 0

Cuts and Flows in a Flow Network



Liquid Cut Flow = 20

Cut Capacity = 25

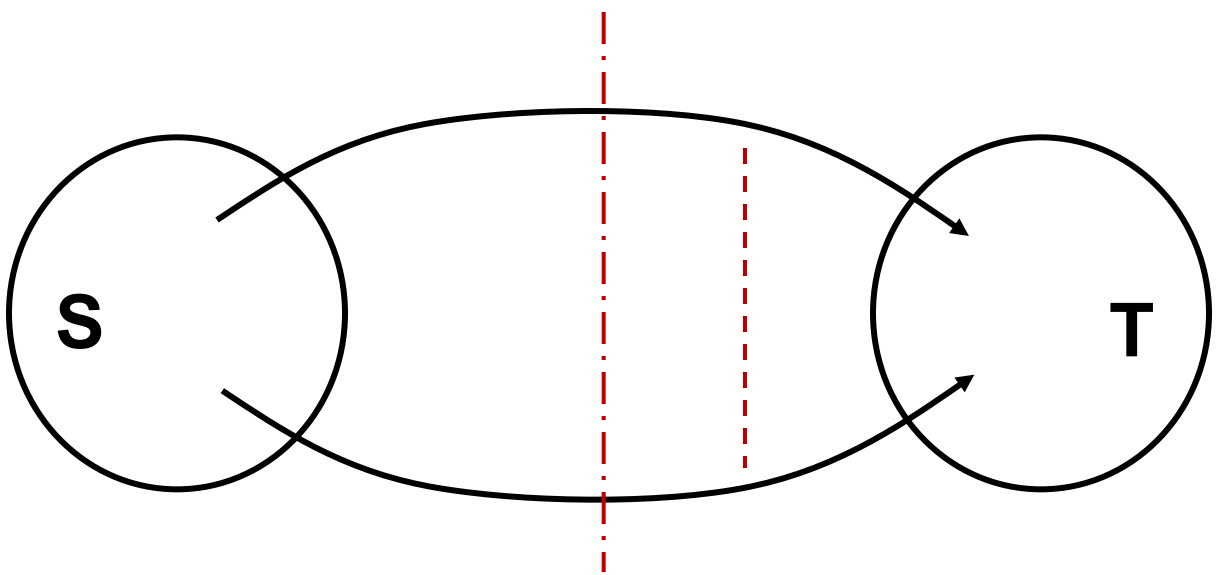
Cut Capacity includes only positive capacity edges.

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA DEI
U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Cuts in a Flow Network(Cont.)

- Any **flow value** is **upper bounded** by the **capacity** of any **cut** in G
  - (S,T) <u>**any**</u> Cut, and flow f :

$$|f| = f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) \leq \sum_{u \in S} \sum_{v \in T} c(u,v) = c(S,T)$$



Max Flow/Min Cut Theorem

- Let G = (V, E), with source *s* and sink *t*, and flow *f*, then the following prepositions are equivalent:
  1. *f* is a maximum flow in G
  2. The residual network G_f has no augmenting paths
  3. $|f| = c(S,T)$ for a cut (S,T) of G

- Proof: 1. ⇒ 2.
  - Assume *f* is a maximum flow in G and that G_f has an augmenting path
  - Then it is possible to define a new flow $f + f_p$ with value $|f| + |f_p| > |f|$; a contradiction

Max Flow/Min Cut Theorem (Cont.)

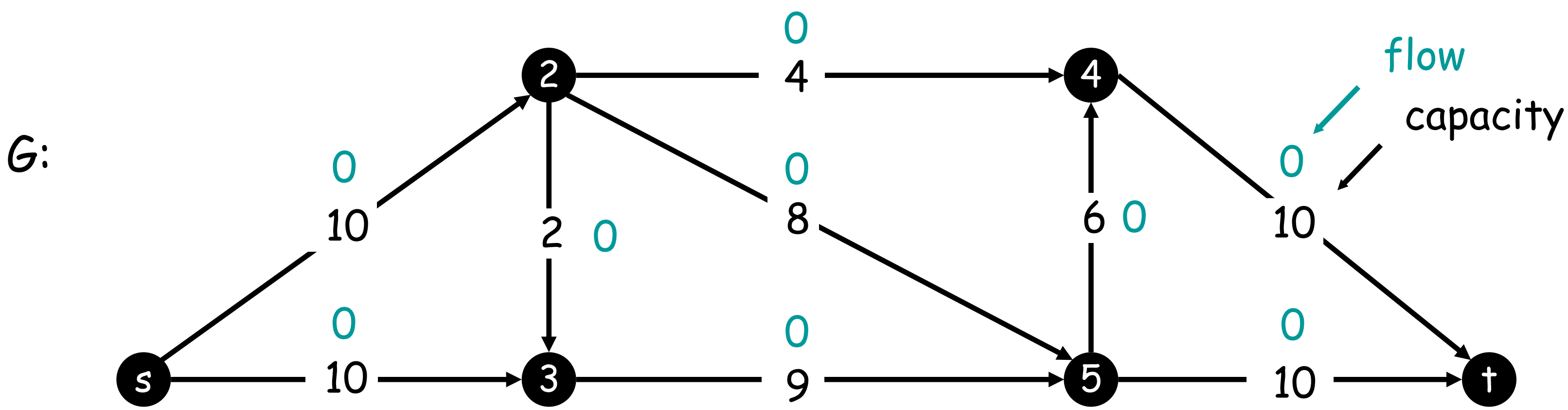
- Proof 2. $\Rightarrow$ 3.
  - G_f **has no Augmenting Paths** *i.e.*, no path from s to t.
  - S = { v $\in$ V : exists a path from s to v in G_f }; T = (V-S); s $\in$ S and t $\in$ T
  - With u $\in$ S and v $\in$ T, we have f(u,v) = c(u,v), as otherwise v would belong to S; and thus
    - |f| = f(S,T) = c(S,T)

- Proof 3. $\Rightarrow$ 1.
  - Given that |f| $\leq$ c(S,T), for **any** cut (S,T) in G
  - As |f| = c(S,T) (defined above), then f is a maximum flow

Ford-Fulkerson Basic Algorithm

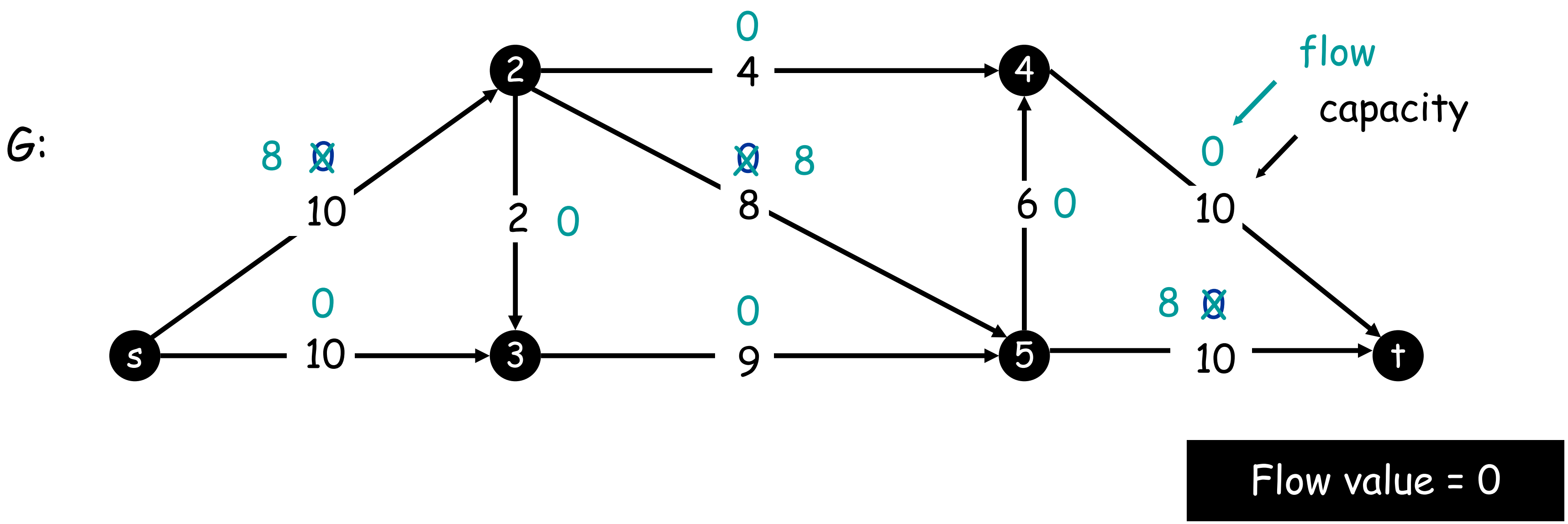
```
Ford-Fulkerson(Graph G, node s, node t)
   foreach (u,v) ∈ E[G] do
          f[u,v] = 0;
          f[v,u] = 0;
   while exists an augmenting path p in residual network G_f do
      compute c_f(p);
      foreach (u,v) ∈ p do
          f[u,v] = f[u,v] + c_f(p)  // Increase flow value
          f[v,u] = f[v,u]  - c_f(p)   // increase reverse flow
```

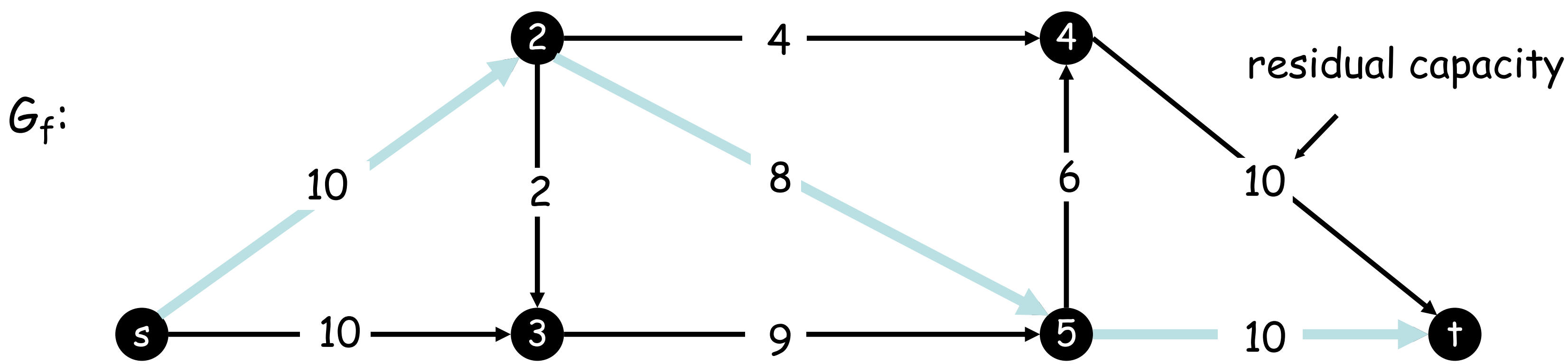
Ford-Fulkerson Algorithm



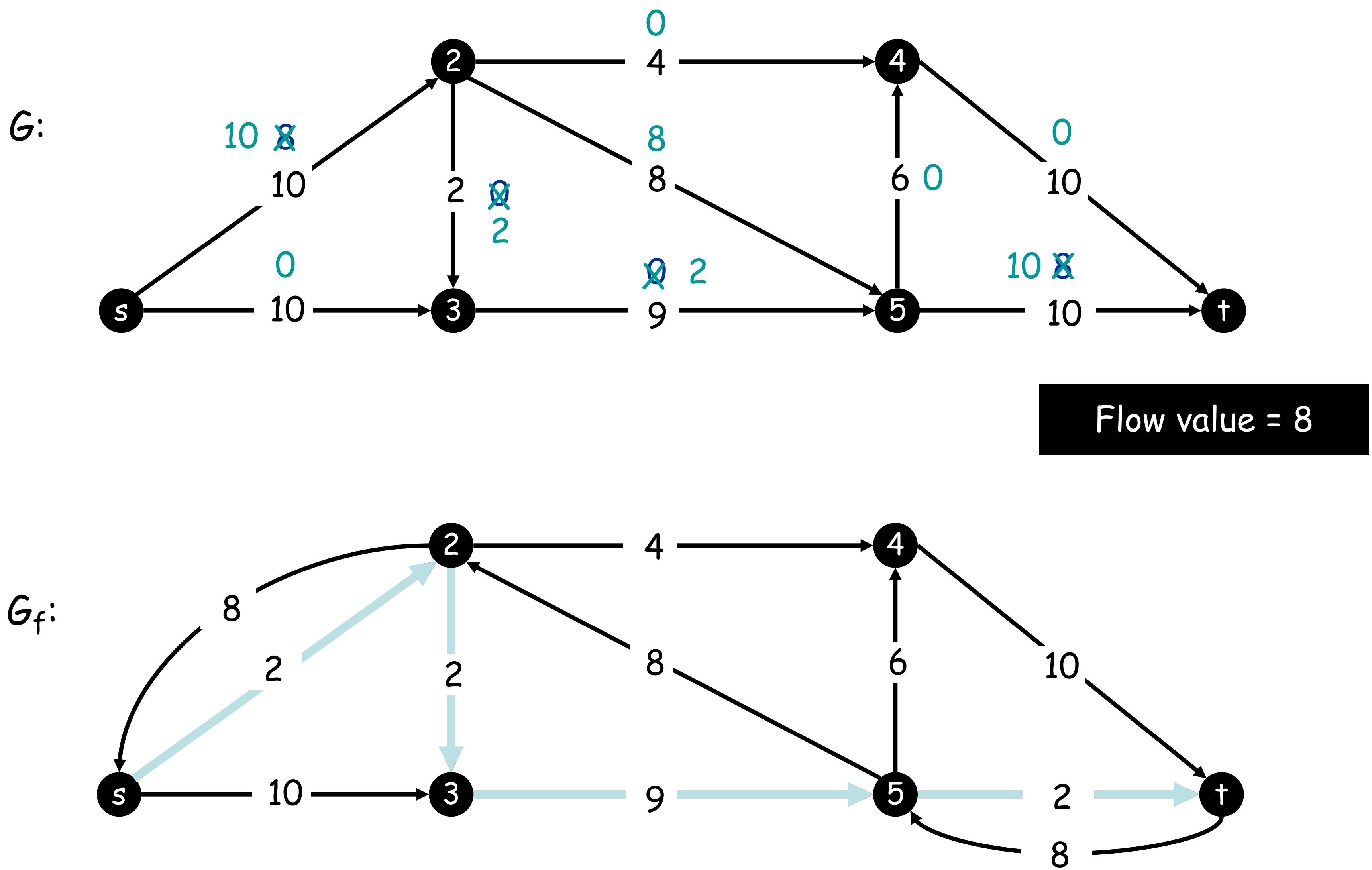
Flow value = 0

Ford-Fulkerson Algorithm

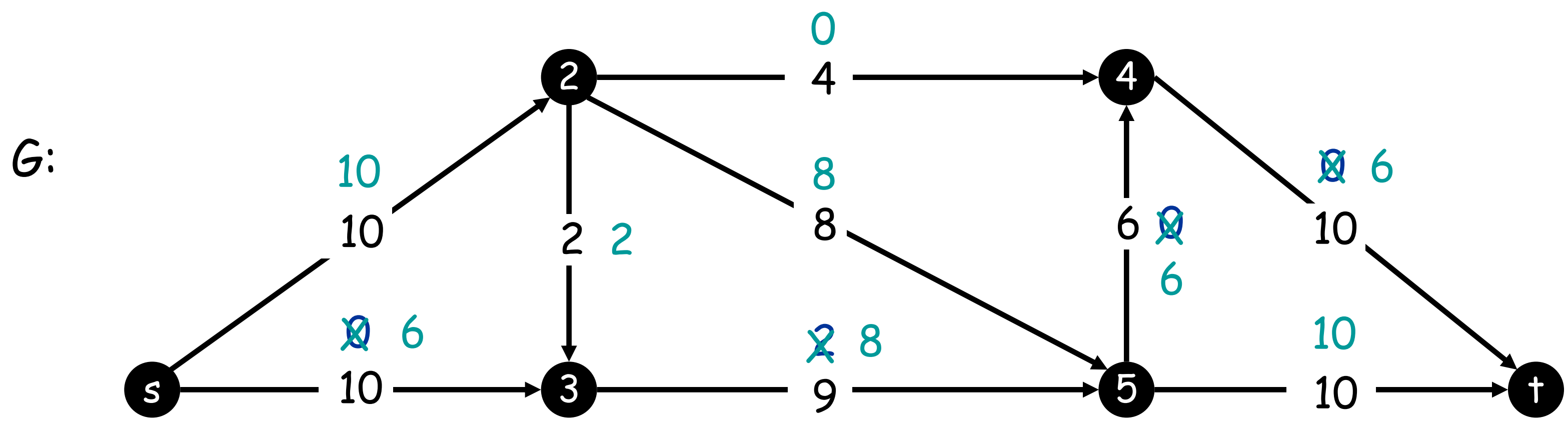




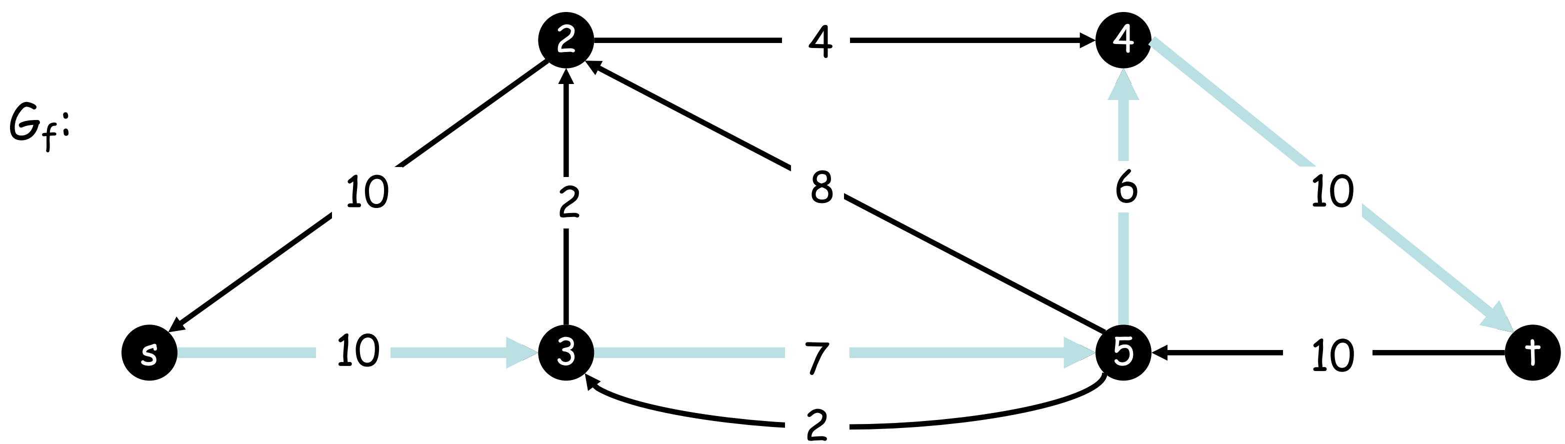
Ford-Fulkerson Algorithm



Ford-Fulkerson Algorithm

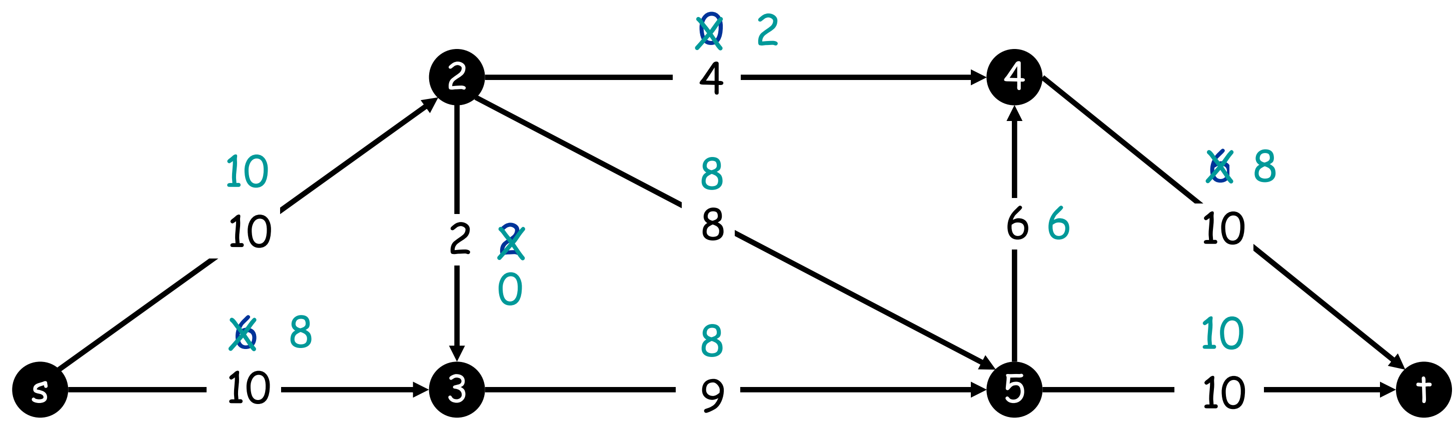


Flow value = 10



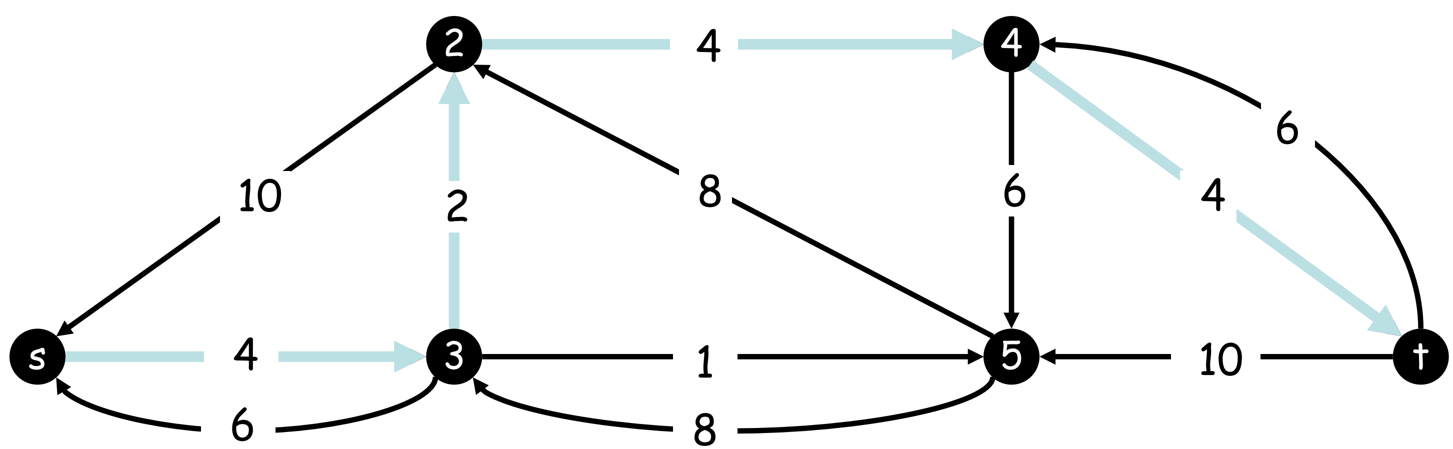
Ford-Fulkerson Algorithm

G:



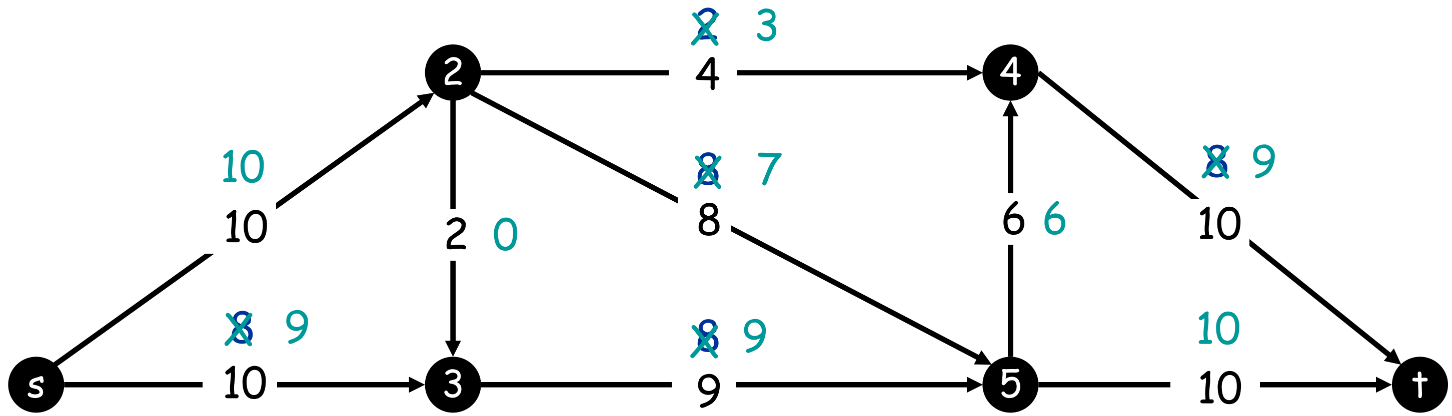
Flow value = 16

G_f:



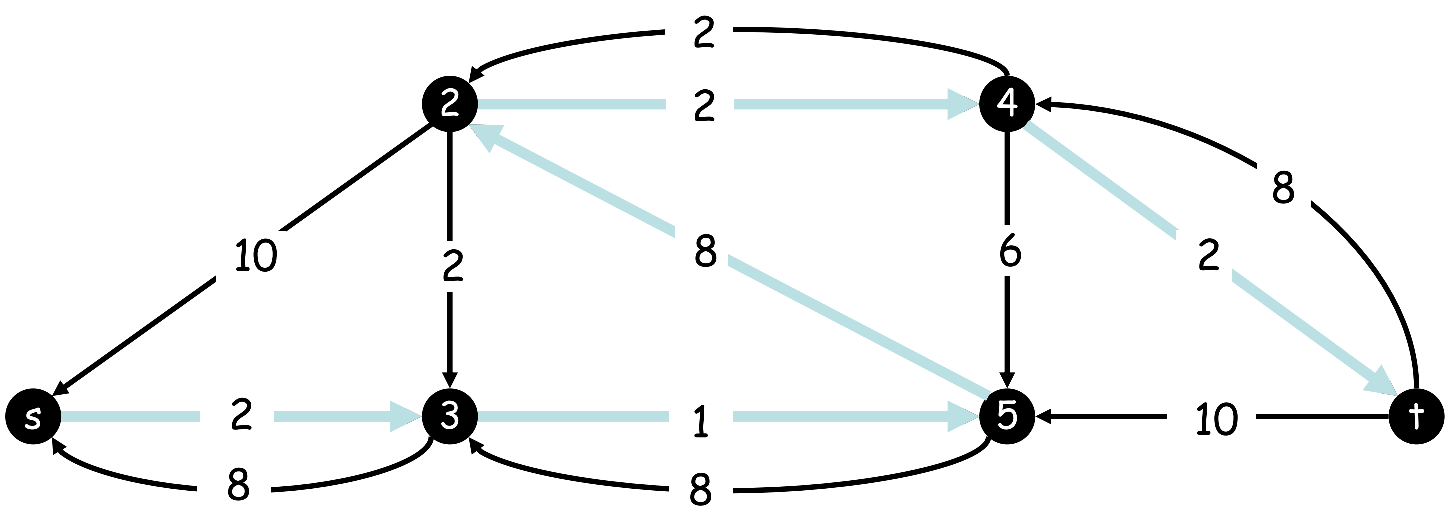
Ford-Fulkerson Algorithm

G:

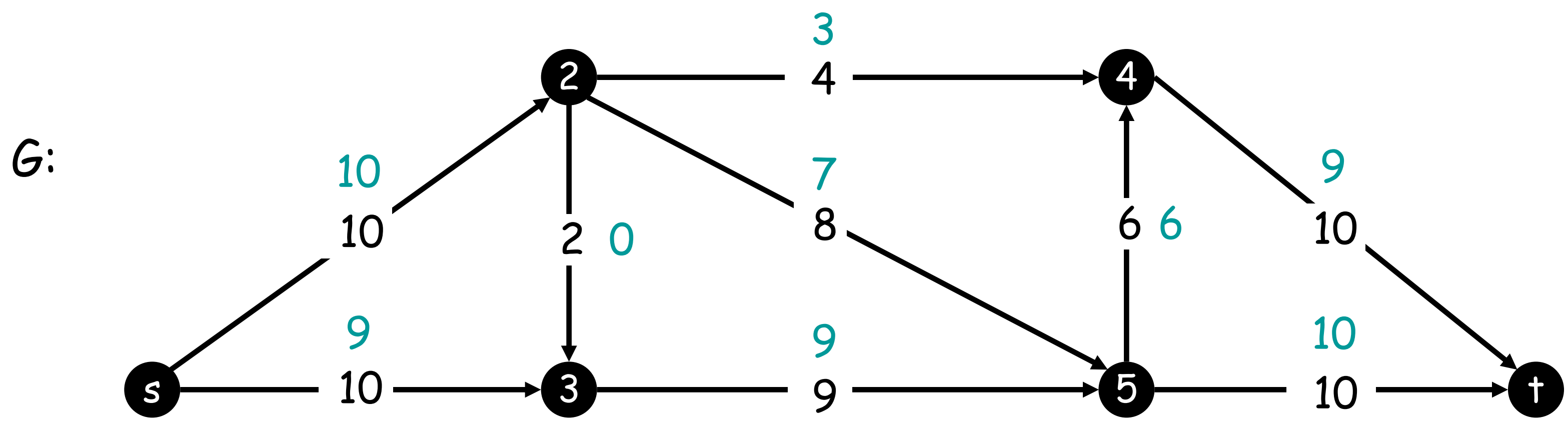


Flow value = 18

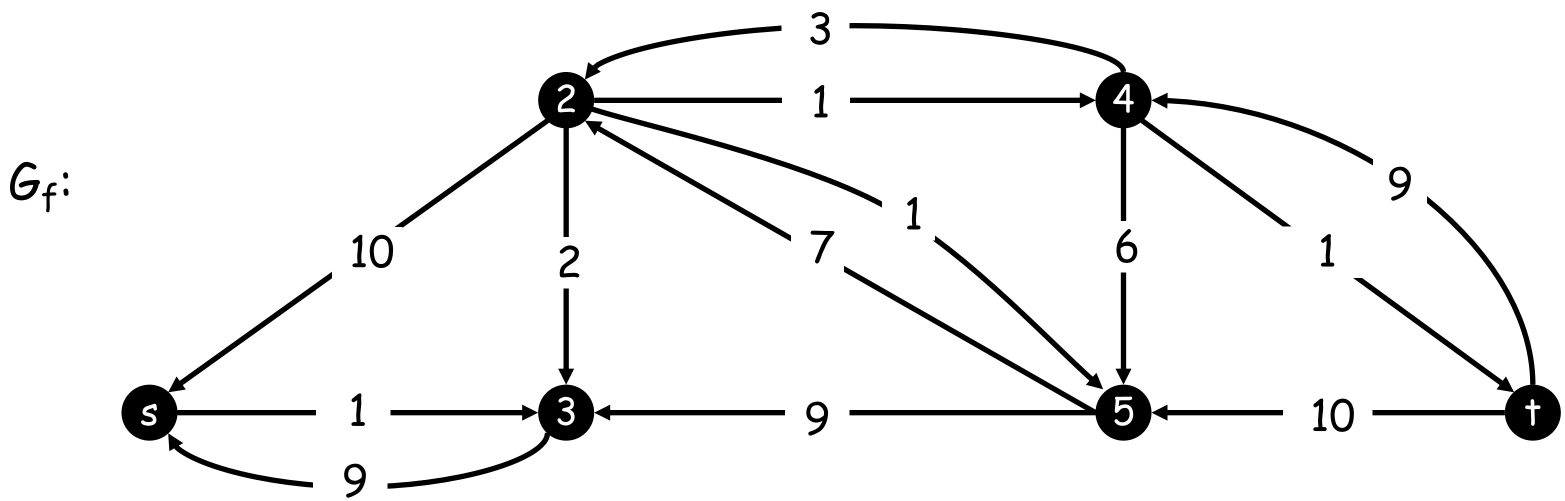
G_f:



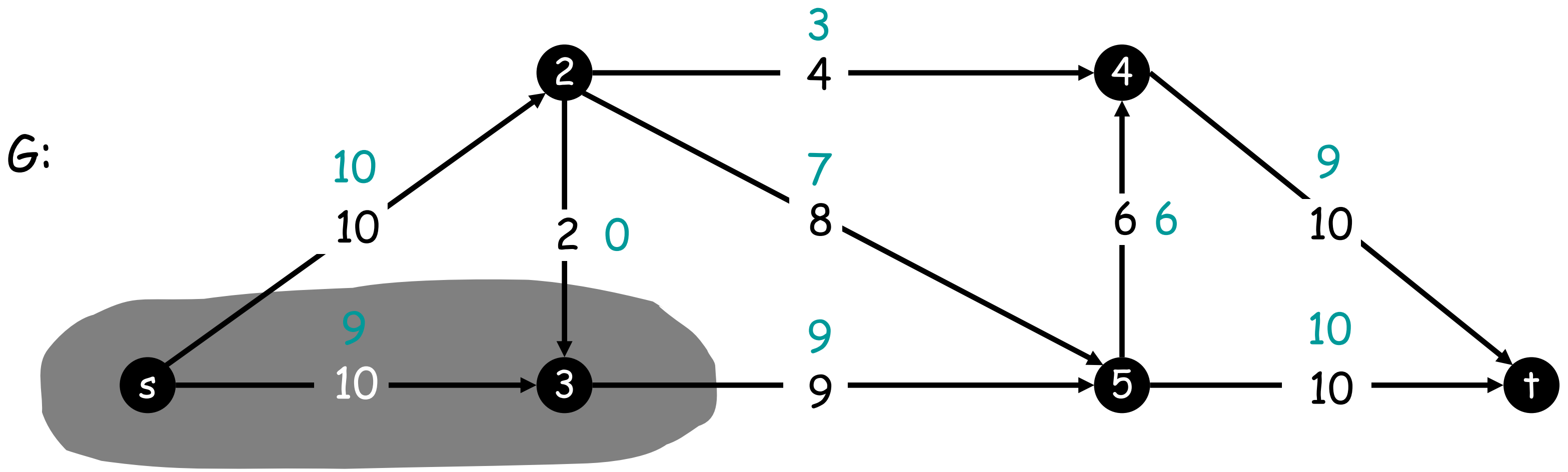
Ford-Fulkerson Algorithm



Flow value = 19

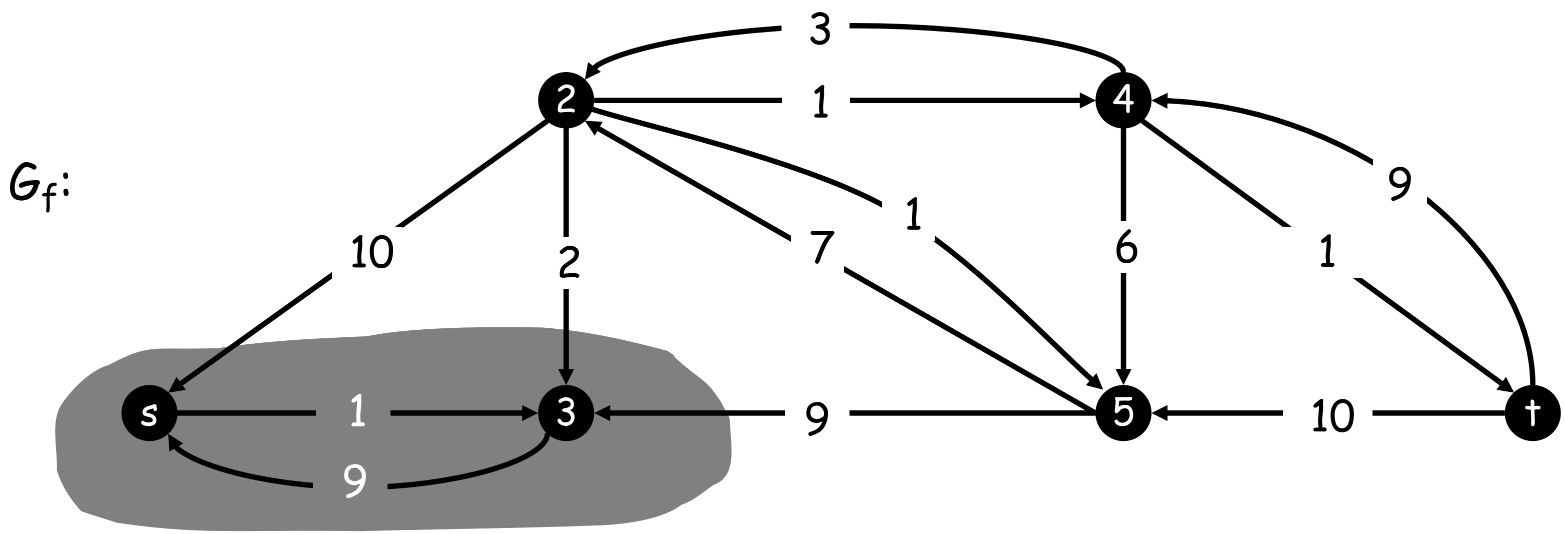


Ford-Fulkerson Algorithm



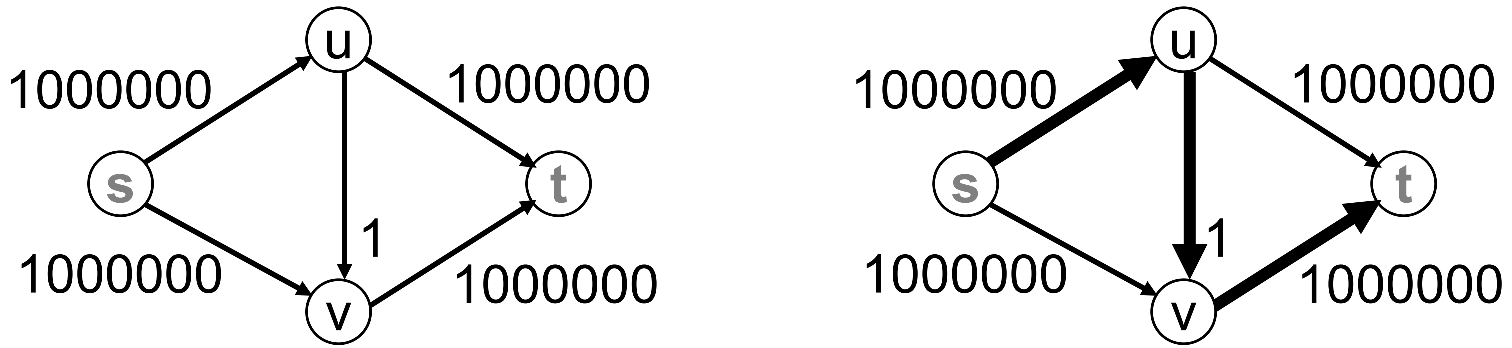
Cut capacity = 19

Flow value = 19



Analysis of Basic Algorithm

- Number of Flow increases might be high…



flow network

augmenting path with residual capacity = 1

- Maximum Flow = 2000000
  - Worst-case: number of augmenting paths is 2000000

Analysis of Basic Algorithm (cont.)

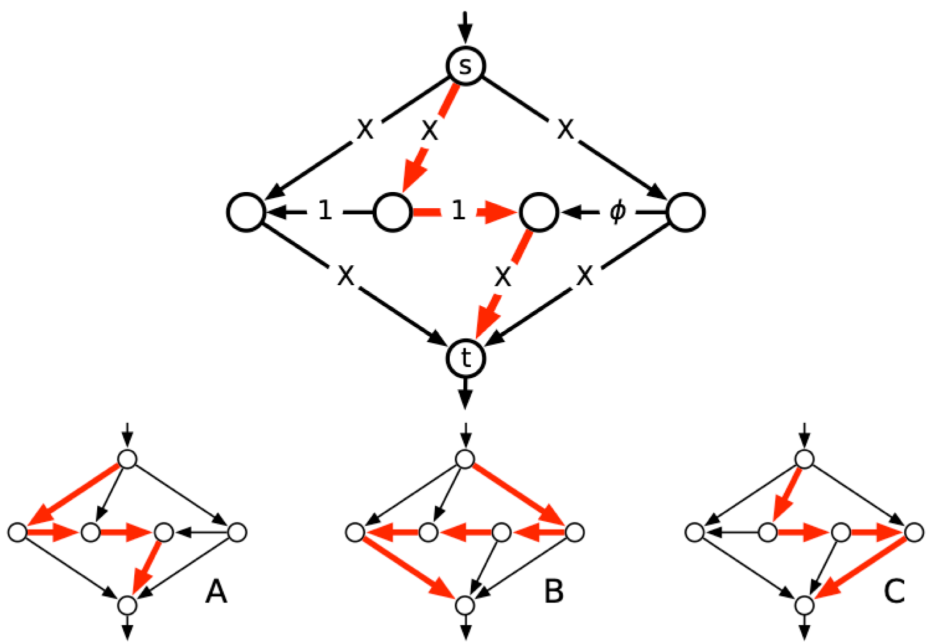
- For Rational Values of Capacities
  - Convert all capacities to integers by scaling
  - Number of augmenting paths limited by the maximum value of the flow $|f^*|$
  - Complexity: $O(E\ |f^*|)$
    - For example: DFS to find augmenting paths

Analysis of Basic Algorithm (cont.)

- For Irrational Values of Capacities
  - Basic Algorithm might never terminate...
  - Basic Algorithm might even converge to incorrect value...

Analysis of Basic Algorithm (cont.)

- For Irrational Values of Capacities
  - Basic Algorithm might never terminate…
  - Basic Algorithm might even converge to incorrect value…



$X \geq 2$

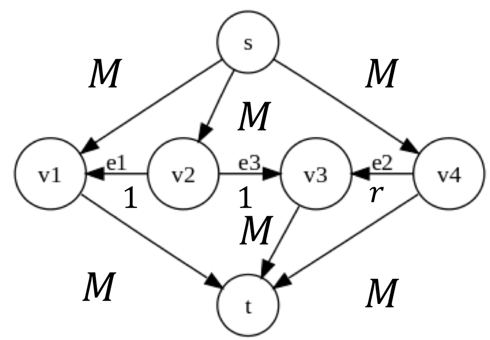
$\emptyset = (\sqrt{5} - 1)/2$

$1 - \emptyset = \emptyset^2$

Analysis of Basic Algorithm (cont.)

- For Irrational Values of Capacities
  - Basic Algorithm might never terminate...
  - Basic Algorithm might even converge to incorrect value...

$$p_1 = \{s, v_4, v_3, v_2, v_1, t\}$$
$$p_2 = \{s, v_2, v_3, v_4, t\}$$
$$p_3 = \{s, v_1, v_2, v_3, t\}$$



$M \geq 2$

$r = (\sqrt{5} - 1)/2$

$1 - r = r^2$

Step	Augmenting path	Sent flow	Residual capacities		
			e_1	e_2	e_3
0			$r^0 = 1$	r	1
1	$\{s, v_2, v_3, t\}$	1	r^0	r^1	0
2	p_1	r^1	$r^0 - r^1 = r^2$	0	r^1
3	p_2	r^1	r^2	r^1	0
4	p_1	r^2	0	$r^1 - r^2 = r^3$	r^2
5	p_3	r^2	r^2	r^3	0

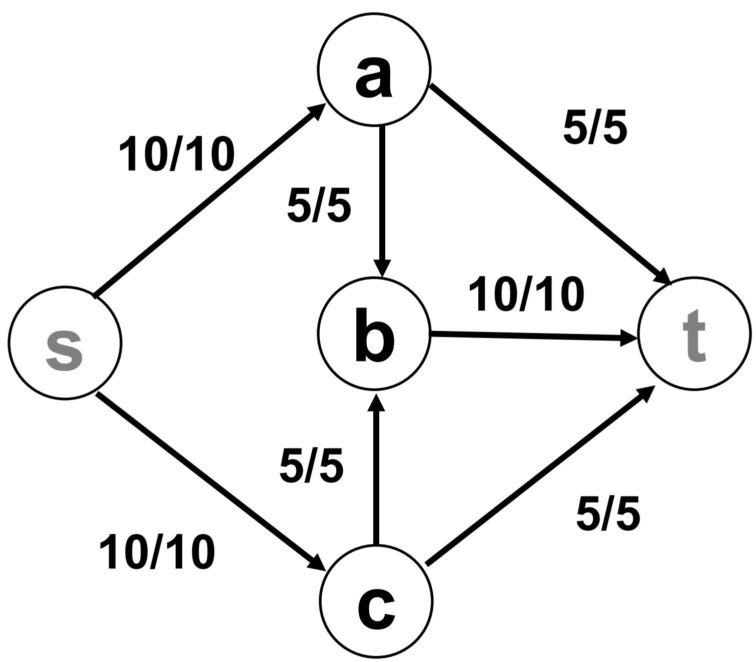
Flow converges to:

$$1 + 2\sum_{i=1}^{\infty} r^i = 3 + 2r$$

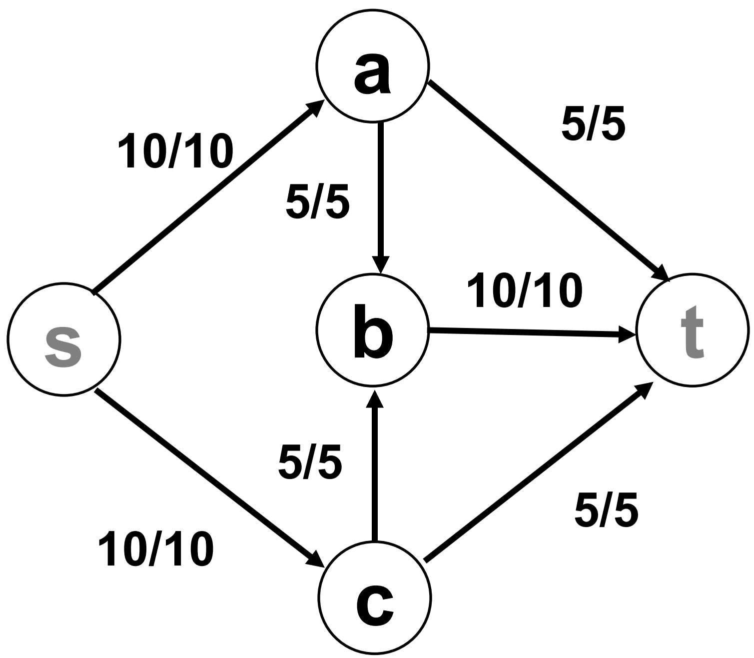
But obvious Max flow is:
$2M+1$

Edmonds-Karp Algorithm

- Choose Augmenting Paths using Shortest Path
  - Each Edge has a distance of 1
  - Use BFS in G_f to identify shortest path
  - Complexity: $O(V E^2)$



Edmonds-Karp Algorithm



BFS Augmenting Paths

- s → a → t, flow: 5, len: 2
- s → c → t, flow: 5, len: 2
- s → a → b → t, flow: 5, len: 3
- s → c → b → t, flow: 5, len: 3

Total Flow: 20

Edmonds-Karp Algorithm - Analysis

- Definitions:
  - $\delta_f(s,v)$: shortest distance from s to v in residual network G_f
  - Sequence of actions:
    - $f \rightarrow G_f \rightarrow BFS \rightarrow p \rightarrow f' \rightarrow G_{f'} \rightarrow BFS \rightarrow p'$
- Results:
  - $\delta_f(s,v)$ increases monotonically with each increase in flow
  - Number of flow increases is O(V E)
  - Execution Time is O(V E^2)
    - O(E) due to BFS and the increase of flow at each step

Edmonds-Karp Algorithm - Analysis

$\delta_f(s,v)$ grows monotonically with each increase of flow

Proof (by contradiction): Consider the first node $v \in V$ such that, after the increase of flow (from f to f'), the distance for the shortest path decreases, $\delta_{f'}(s,v) < \delta_f(s,v)$

- Let $p = \langle s,\ldots,u,v \rangle$ be the shortest path from s to v in $G_{f'}$ with $(u,v) \in E_{f'}$ and $\delta_{f'}(s,u) = \delta_{f'}(s,v)-1$
- Because how we choose v, we know that the distance of vertex u did not decrease, i.e, $\delta_{f'}(s,u) \geq \delta_f(s,u)$
- **Claim:** $(u,v) \notin E_f$ why? If $(u,v) \in E_f$ we would have had:

$$\begin{aligned}\delta_f(s,v) &\leq \delta_f(s,u)+1\\ &\leq \delta_{f'}(s,u)+1\\ &= \delta_{f'}(s,v)\end{aligned}$$

Contradicts original assumption $\delta_{f'}(s,v) < \delta_f(s,v)$

Edmonds-Karp Algorithm - Analysis

$\delta_f(s,v)$ grows monotonically with each increase of flow

Proof (by contradiction): Consider the first node $v \in V$ such that, after the increase of flow (from f to f'), the distance for the shortest path decreases, $\delta_{f'}(s,v) < \delta_f(s,v)$

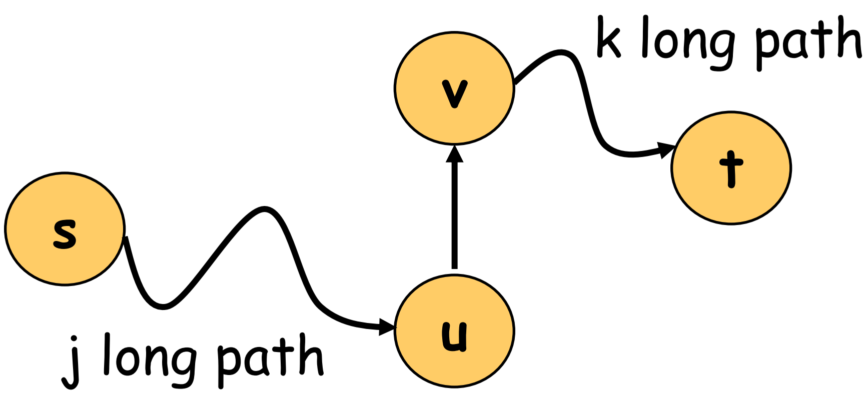
- How can we have $(u,v) \notin E_f$ and $(u,v) \in E_{f'}$?
- flow increase from v to u
- Increase always along the shortest path, then the shortest path between s and u in G_f has (v,u) as its last edge:

$$\begin{aligned}\delta_f(s,v) &= \delta_f(s,u)-1\\ &\leq \delta_{f'}(s,u)-1\\ &= \delta_{f'}(s,v)-2\end{aligned}$$

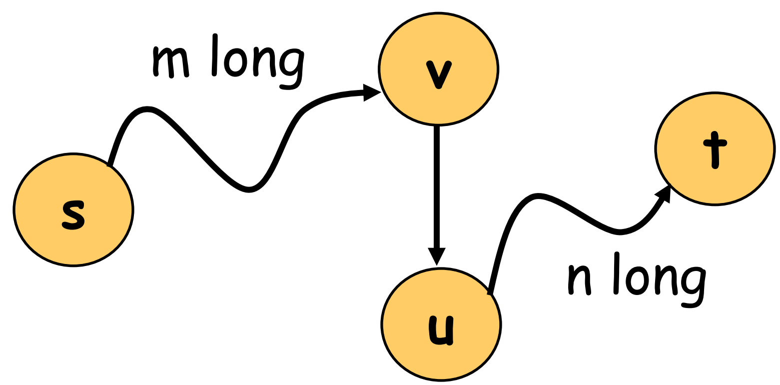
Contradicts original assumption $\delta_{f'}(s,v) < \delta_f(s,v)$

Edmonds-Karp Algorithm - Analysis

- Always choose an augmenting path with as few edges as possible (say p edges).
  - This might create a new backedge, *e.g.*, (v, u).
  - This new edge can't be in a new path of p edges…
  - Meanwhile, at least one original edge has been eliminated.
  - Thus, there are at most E iterations using p long paths.



j+1+k is a minimal length path (p)



$m \geq j+1$ and $n \geq k+1$
so $m+n > j+k+1 = p$

Edmonds-Karp Algorithm - Analysis

- Number of flow increases is O(V E)
  - Edge (u,v) in Residual G_f is **critical** if Residual Capacity of p is the same as the Edge Capacity c(u,v)
    - **Obs:** critical edge disappears after a new flow
  - How many Times can an edge (u,v) be critical?
    - As augmenting paths are all shortest paths, $\delta_f(s,v) = \delta_f(s,u) + 1$
    - (u,v) will only become part of residual network after edge (v,u) appears in an augmenting path (with flow f')
      - Given that, $\delta_{f'}(s,u) = \delta_{f'}(s,v) + 1$
      - Given the, $\delta_f(s,v) \leq \delta_{f'}(s,v)$ (previous result)
      - We get,

$$\begin{aligned}\delta_{f'}(s,u) &= \delta_{f'}(s,v) + 1\\ &\geq \delta_f(s,v) + 1\\ &= \delta_f(s,u) + 2\end{aligned}$$

Conclusion: From the time (u,v) first becomes critical to the time it next becomes critical, distance from s to u increased by at least 2.

Edmonds-Karp Algorithm - Analysis

– Distance from s to u increases at least 2 units each time the edge (u,v) is critical
  - In the limit, distance from s to u is no larger than |V| - 2
  - Therefore (u,v) can only be critical O(V) times
  - There are O(E) pairs of vertices or edges
  - During execution of the Edmonds-Karp algorithm the total number of times that edges can be critical is O(V E) – the number of flow increases

Edmonds-Karp Algorithm - Analysis

- Always choose an augmenting path with as few edges as possible
  - At most E iterations use p-edge augmenting path
  - Longest augmenting path has V-1 edges.
  - Thus, there are at most O(VE) augmentations.
  - How long does it take to find and process a shortest augmenting path?

- Complexity of Edmonds-Karp is $O(V E^2)$
  - Complexity of BFS is O(V+E) = O(E) (given V = O(E))
  - Increases of flow is O(V E)

Summary

- Maximum Flows in Graphs
  - Ford-Fulkerson Method
    - Generic Algorithm Analysis
    - Complexity Analysis (integer values): **O(E |f*|)**
    - With Irrational Values is may never terminate
  - Edmonds-Karp Algorithm
    - Complexity analysis **O(V E^2)**