Data Structures & Algorithms

Adil M. Khan
Professor of Computer Science
Innopolis University

a.khan@innopolis.ru

Recap

- Shortest Path Problem
- Shortest Path Algorithms

    - ➢One-to-All (Dijkstra’s and Bellman-Ford Algorithms)
    - ➢All-to-All (Floyd Warshall’s Algorithm)

Today’s Objectives

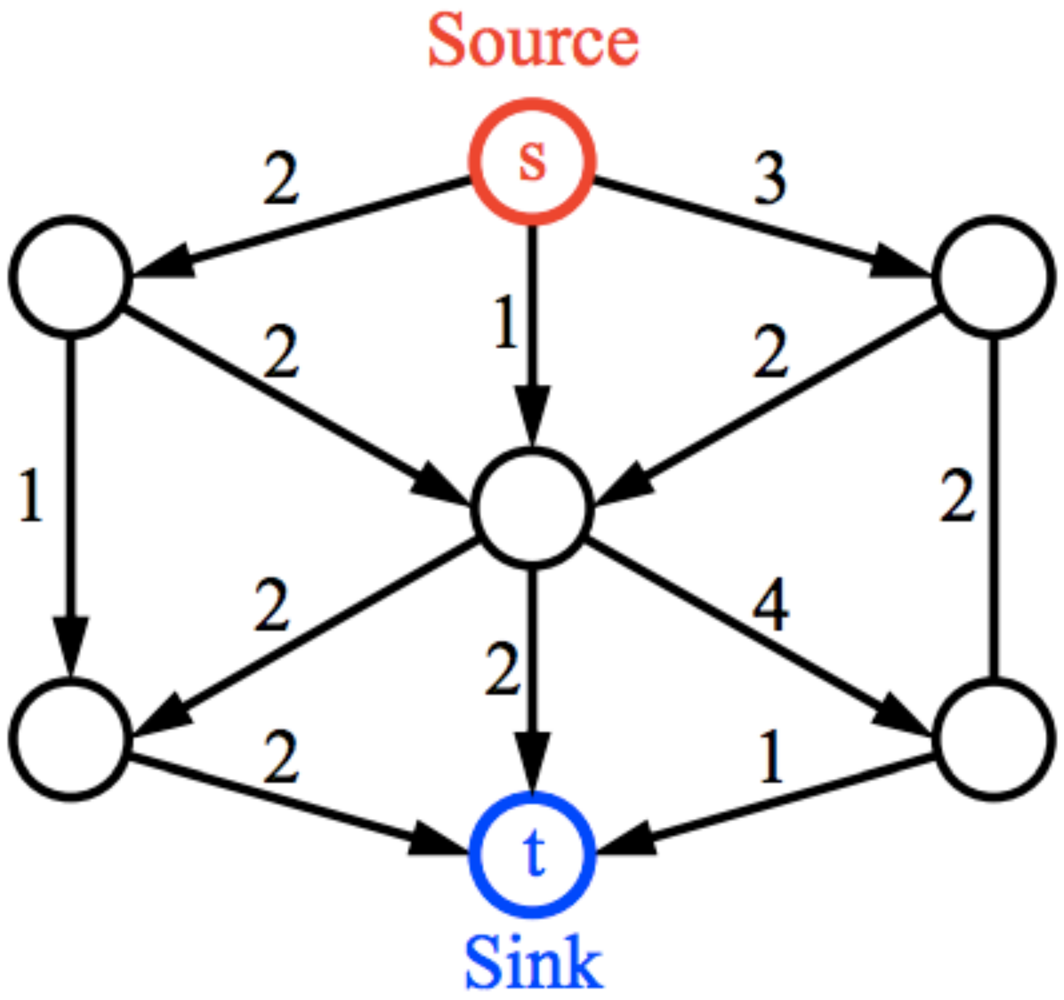
- Flow networks
- Maximum flow Problem
- How to find maximum flow in flow networks?
    - ➢Residual network and augmenting paths
- Time complexity analysis
- Cuts
- Flow across a cut, and cut capacity
- Max-flow min-cut theorem

Flow Networks

- Directed Graph
- Weights on edges, called capacities
- Two special nodes (vertices)

  - Source – “s” – node with no incoming edge
  - Sink – “t” – node with no outgoing edges

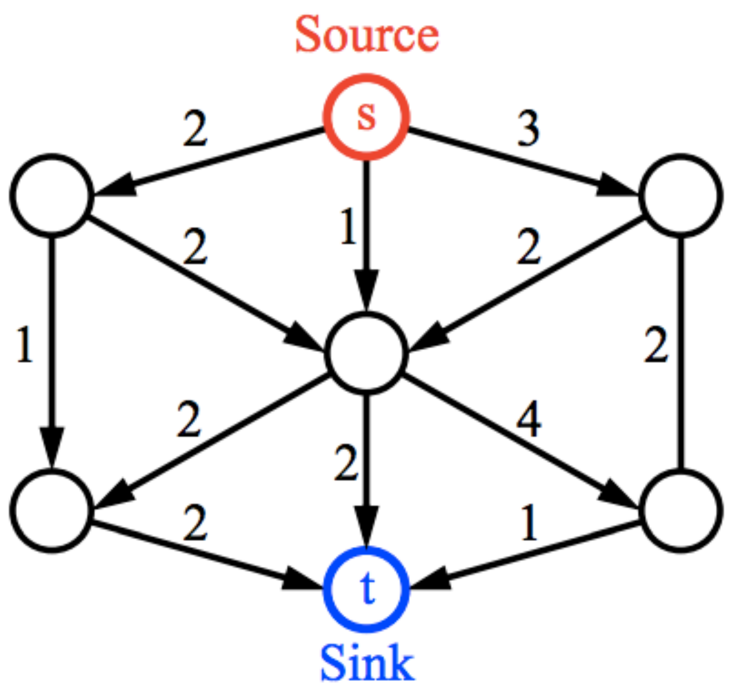
Flow Networks



Max Flow Problem

- “Given a network N (graph G), find a flow f of maximum value.”

- Applications
  - ➢Shipping products
  - ➢Bipartite matching
  - ➢Image segmentation



Capacity and Flow

- Edge Capacities – are non-negative weights on the edges
- Flow – can be thought of as a value such that
  - 0<= flow <= capacity {for a given edge}
  - flow into a node = flow out of a node {for a given node}
  - **Value**: combined flow into the sink {for a given network}

Properties

-Capacity rule: $\forall$ edge (u,v)

$$0 \leq \mathrm{flow}(u,v) \leq \mathbf{capacity}(u,v)$$

-Conservation rule: $\forall$ vertex $v \neq s, t$

$$\sum_{u \in \mathrm{in}(v)} \mathrm{flow}(u,v) = \sum_{w\text{Œ}\mathrm{out}(v)} \mathrm{flow}(v,w)$$

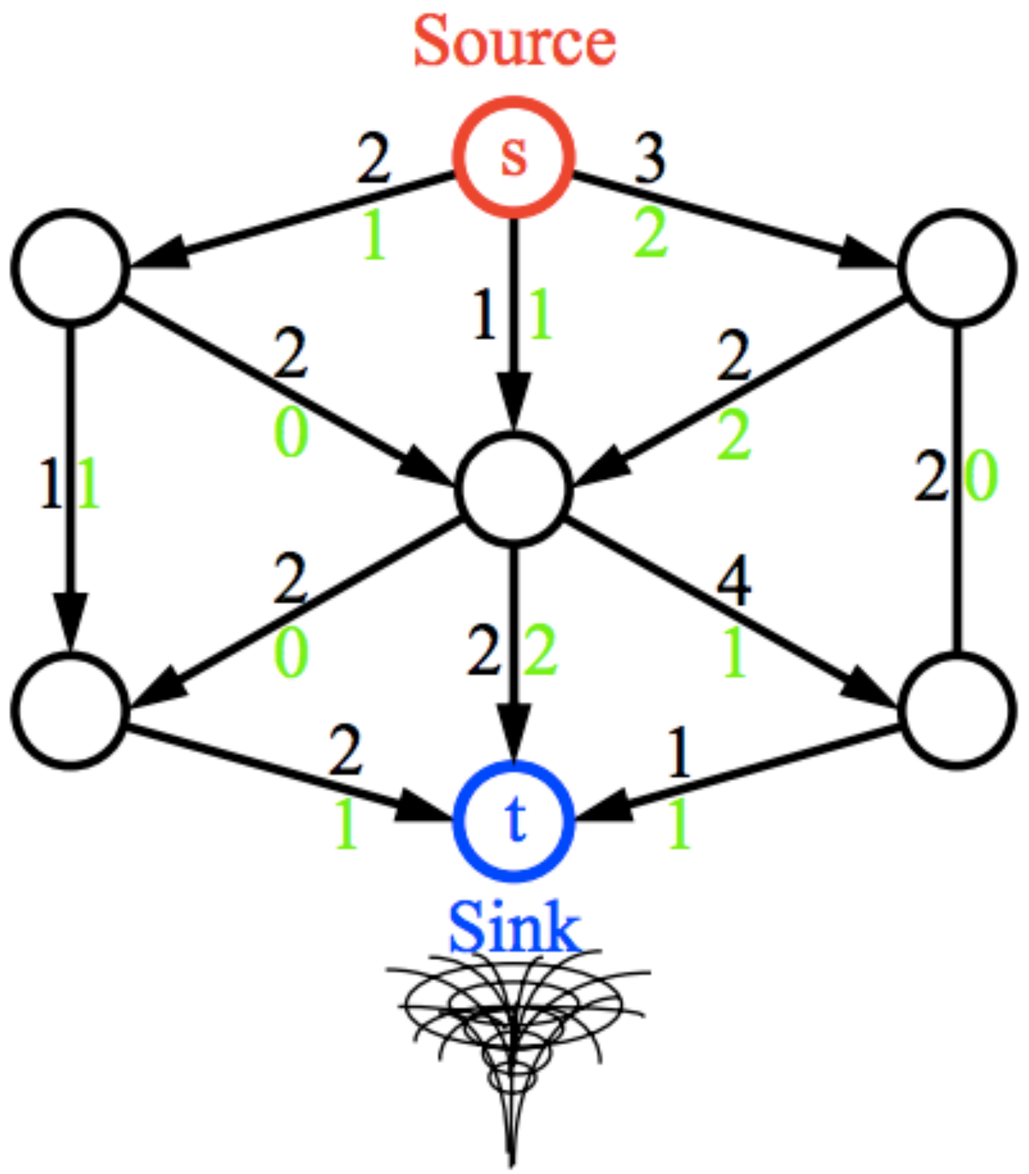
-Value of flow:

$$|f| = \sum_{w \in \mathrm{out}(s)} \mathrm{flow}(s,w) = \sum_{u \in \mathrm{in}(t)} \mathrm{flow}(u,t)$$

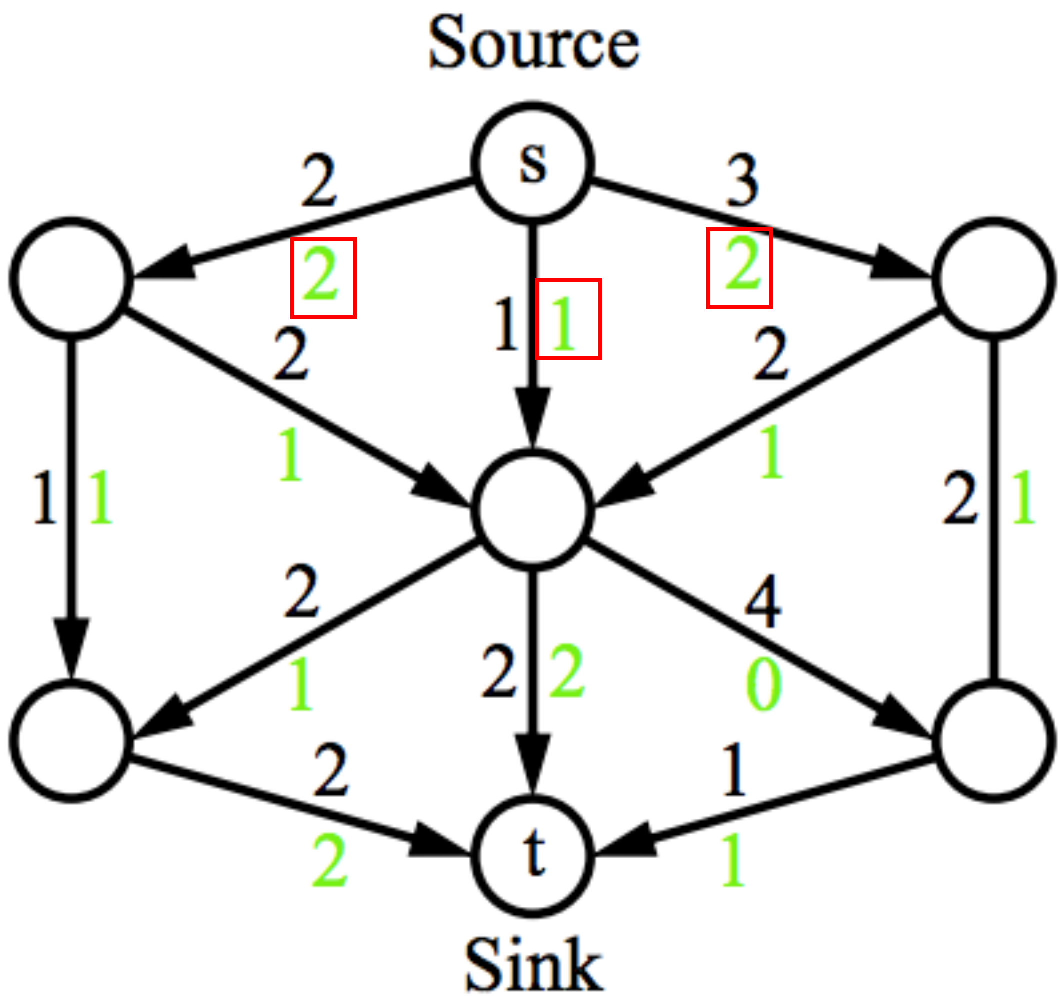
Capacity and Flow

- Thus capacity can be thought of as *bandwidth*
- Whereas flow can be thought of as the *actual load*

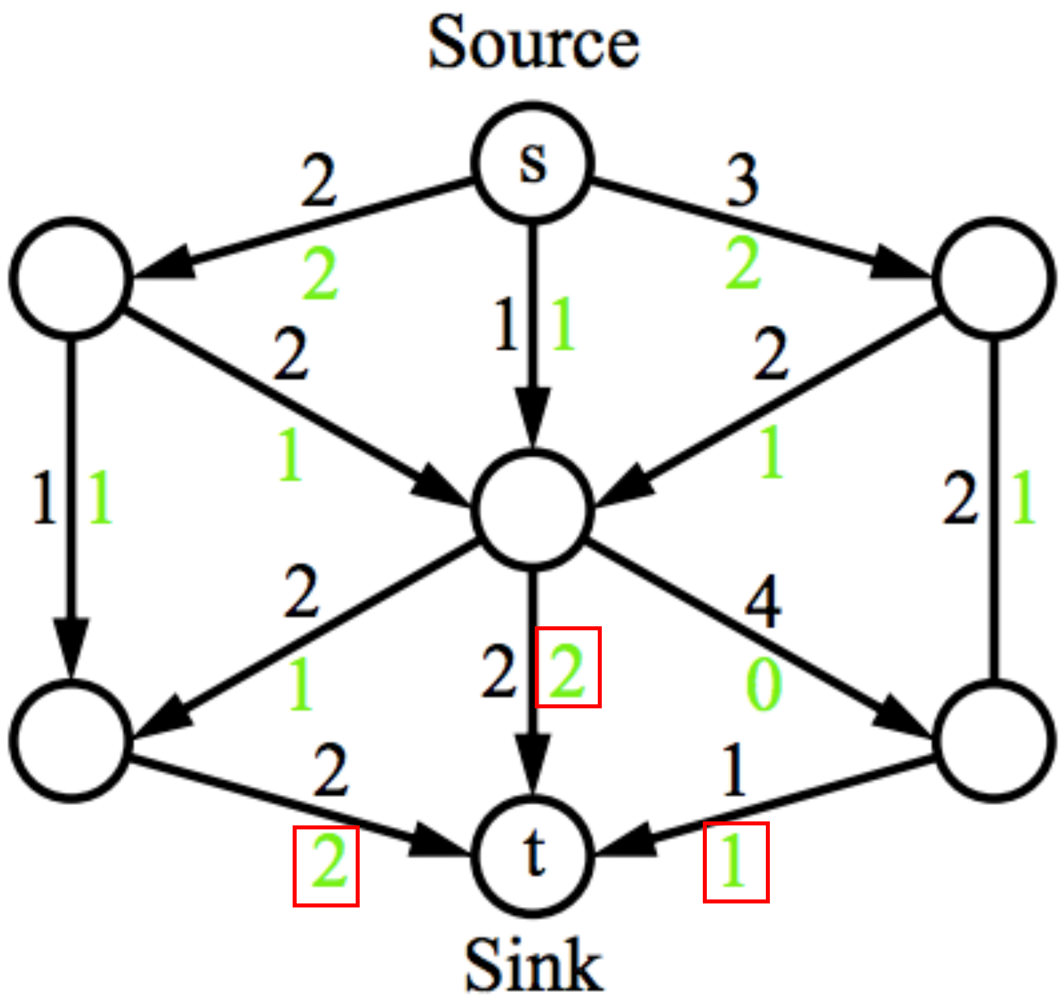
Capacity and Flow



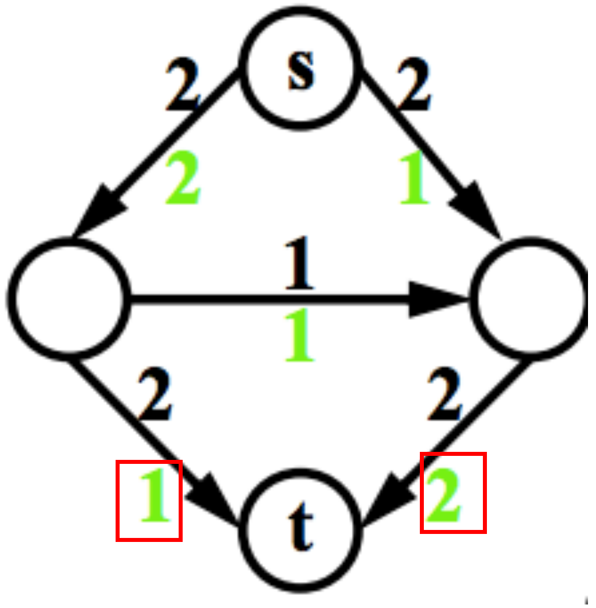
Example of Max Flow



Example of Max Flow

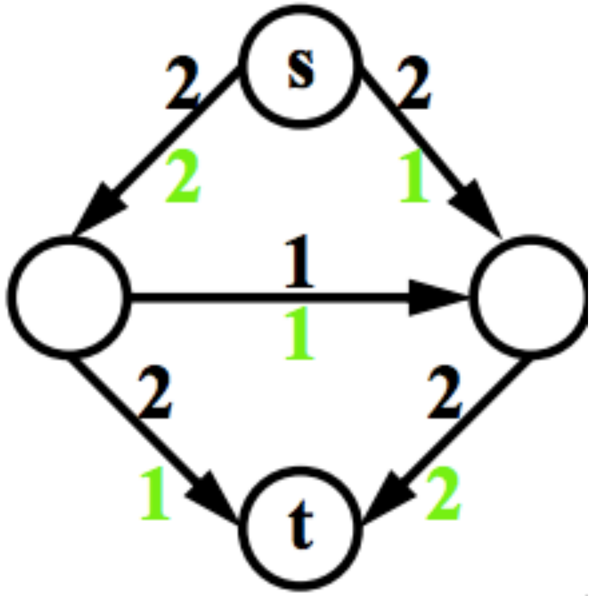


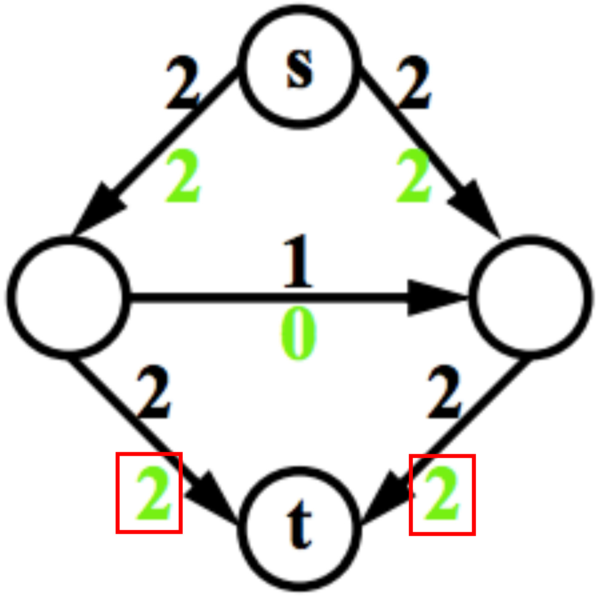
Increasing Flow



A network with a flow
of value **3**

Increasing Flow

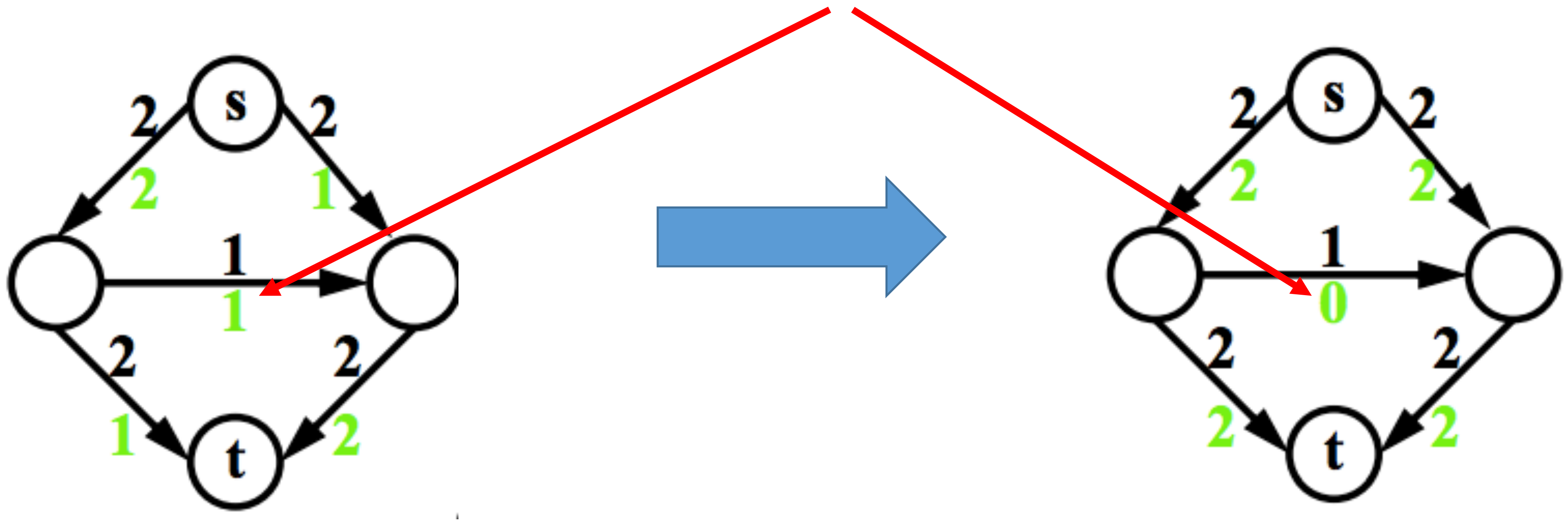




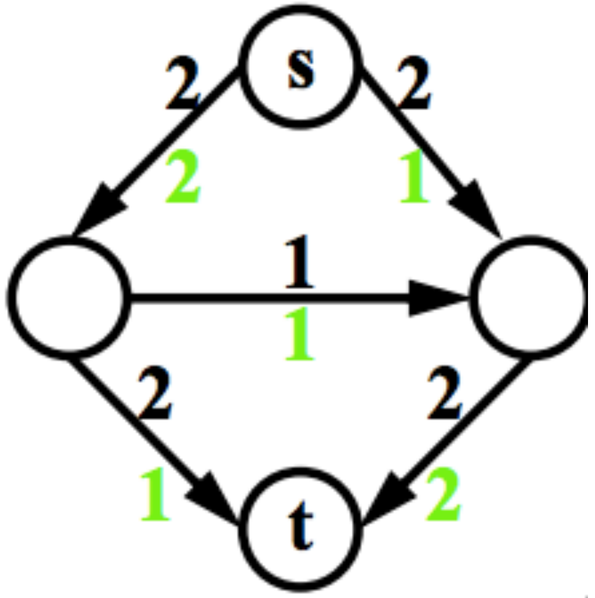
We have increased the flow value to 4!

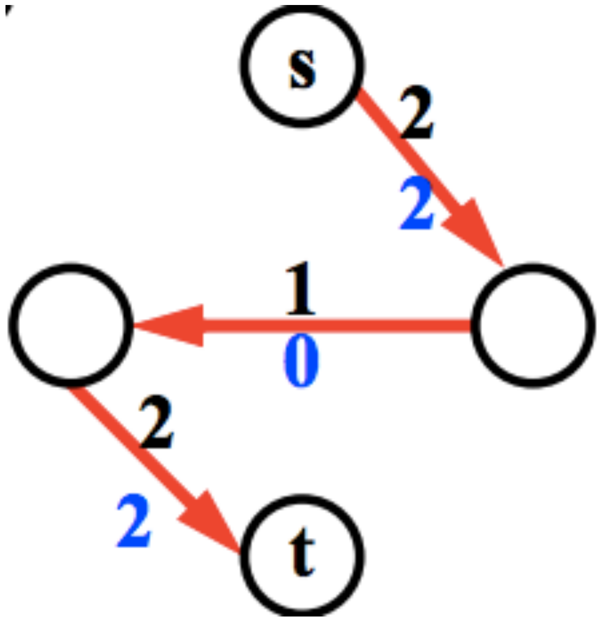
Increasing Flow

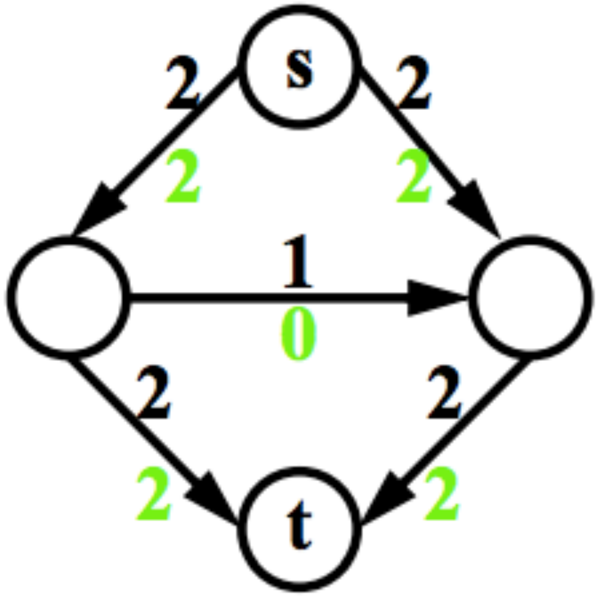
But notice this change!!



Understanding Increasing Flow

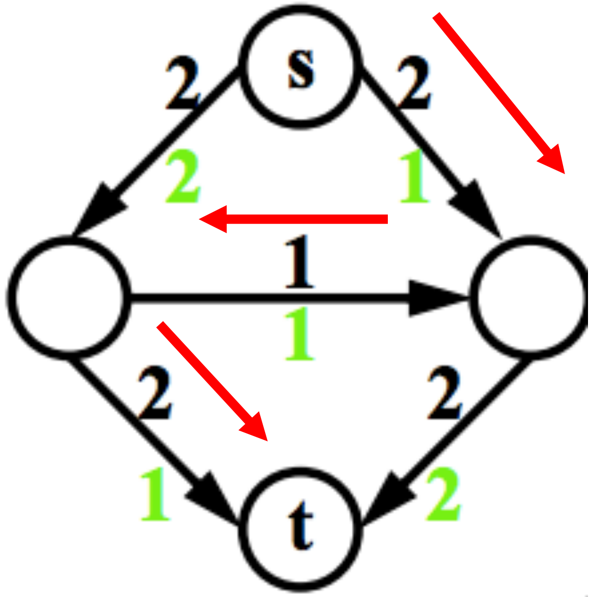






Thus, to increase the net flow, we might have to decrease flow at some edges!

Augmenting Path

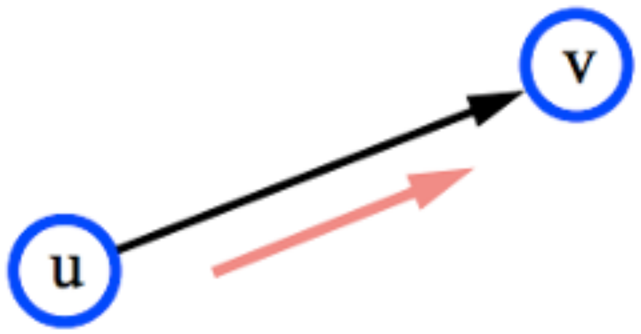


Augmenting
path

- A path from source to sink
- May not exist in the actual network

Augmenting Path

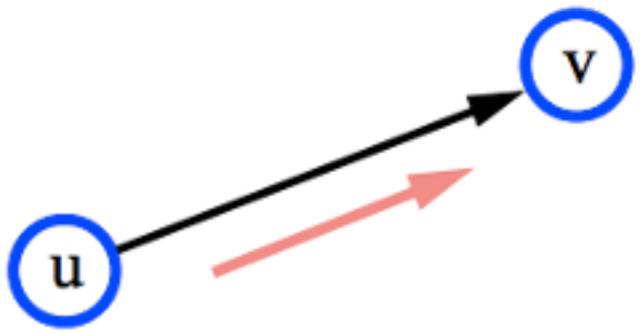
- **Forward Edges**

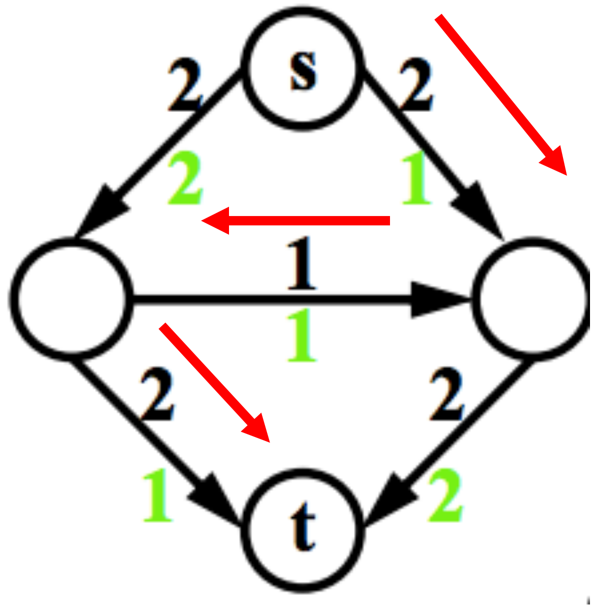


Flow can be increased along these edges

Augmenting Path

- **Forward Edges**



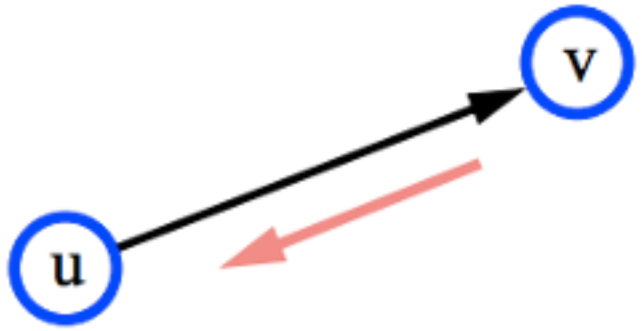


Which edges are forward edges?

Flow can be increased along these edges

Augmenting Path

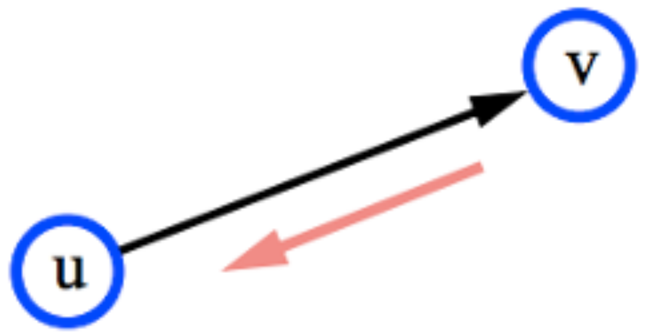
- **Backward Edges**

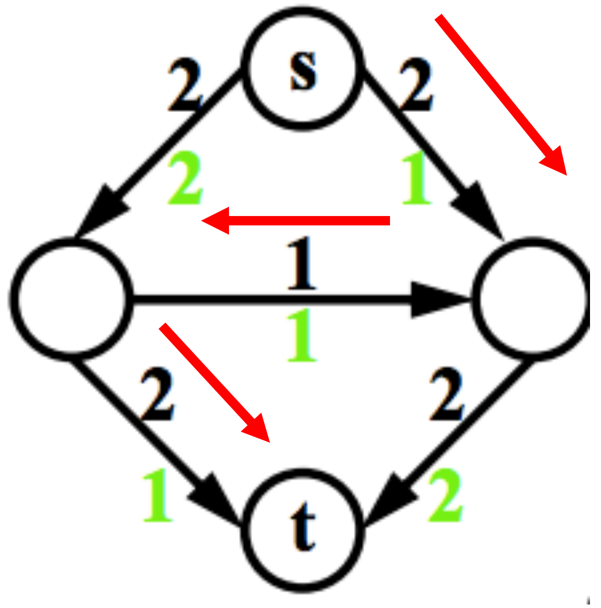


Flow can be decreased along these edges

Augmenting Path

- **Backward Edges**





Which edges are backward edges?

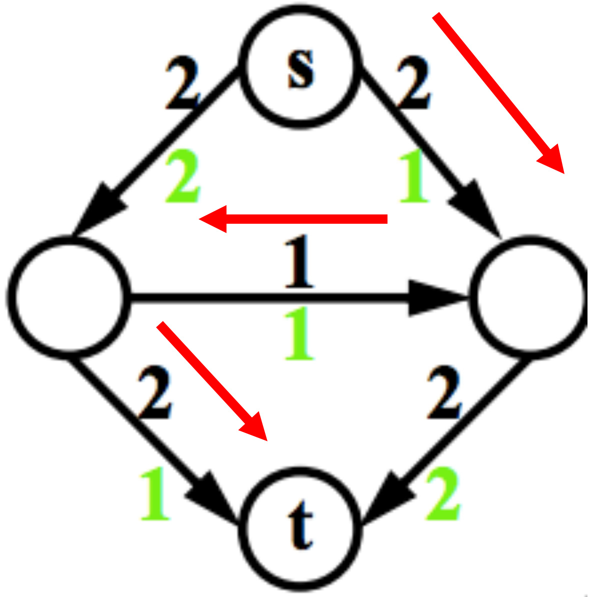
Flow can be decreased along these edges

Formal Definition: Augmenting Path

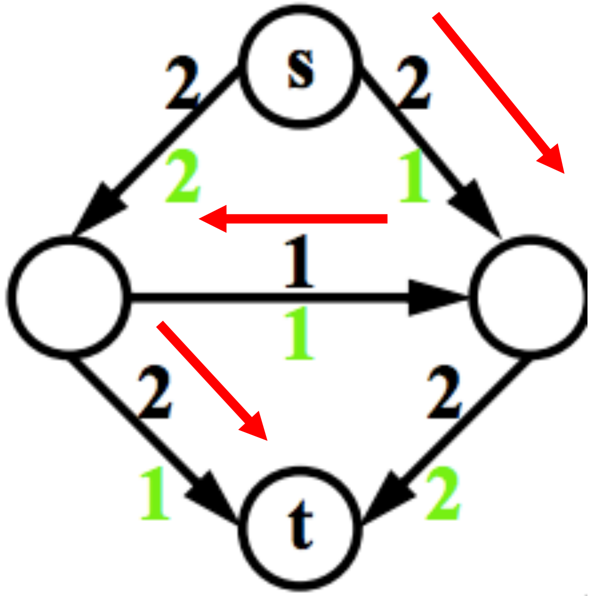
- Let f be a flow in N. The key idea is the following.

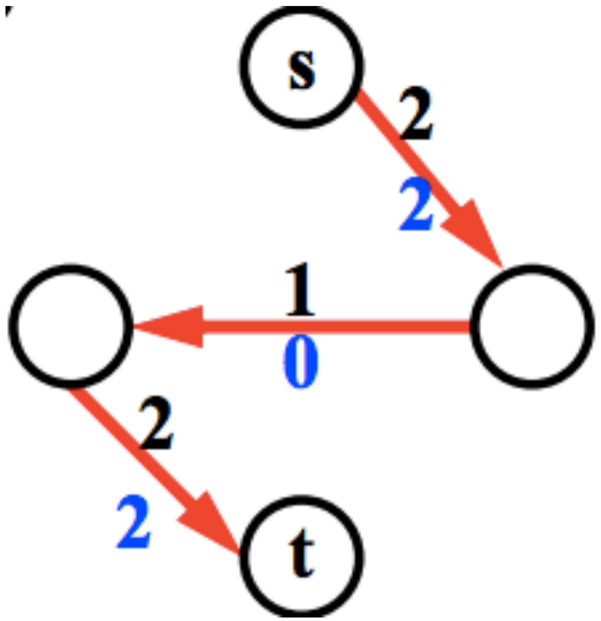
Let $< v_0, v_1, \dots, v_k >$ be a sequence of nodes (not necessarily a path in N), where $v_0 = s$ and $v_k = t$, such that for each $i \in [0: k-1]$ one of the following two holds:

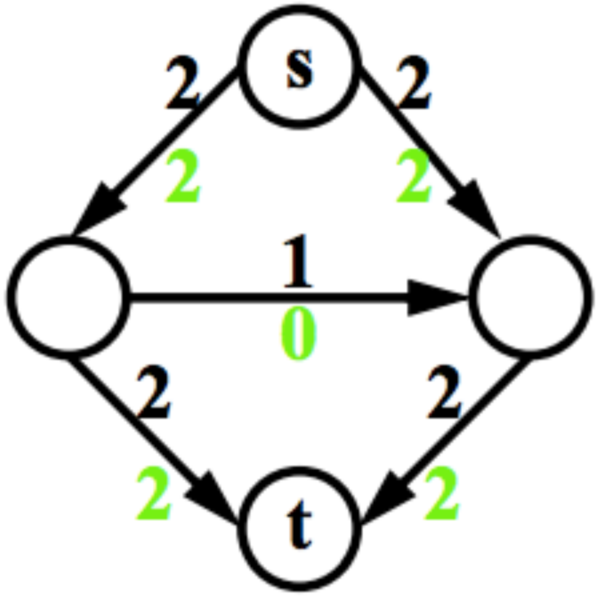
1. Either $(v_i, v_{i+1}) \in E$, and $f(v_i, v_{i+1}) < c(v_i, v_{i+1})$
2. Or, $(v_{i+1}, v_i) \in E$ and $f(v_{i+1}, v_i) > 0$



Increasing the Flow Along Augmenting Path







Ford & Fulkerson Algorithm

```
initialize network with null flow;
Method FindFlow
  if augmenting paths exist then
    find augmenting path;
    increase flow;
    recursive call to FindFlow;
```

Efficiently Finding the Augmenting Paths

- Using *Residual Networks*

Residual Capacity

- To understand Residual Networks, we must first learn what is the *residual capacity*
  - Let $N = (V, E)$ be a flow network with source s and sink t
  - Let f be a flow in N and consider a pair of vertices $u, v \in V$
  - We define the residual capacity as

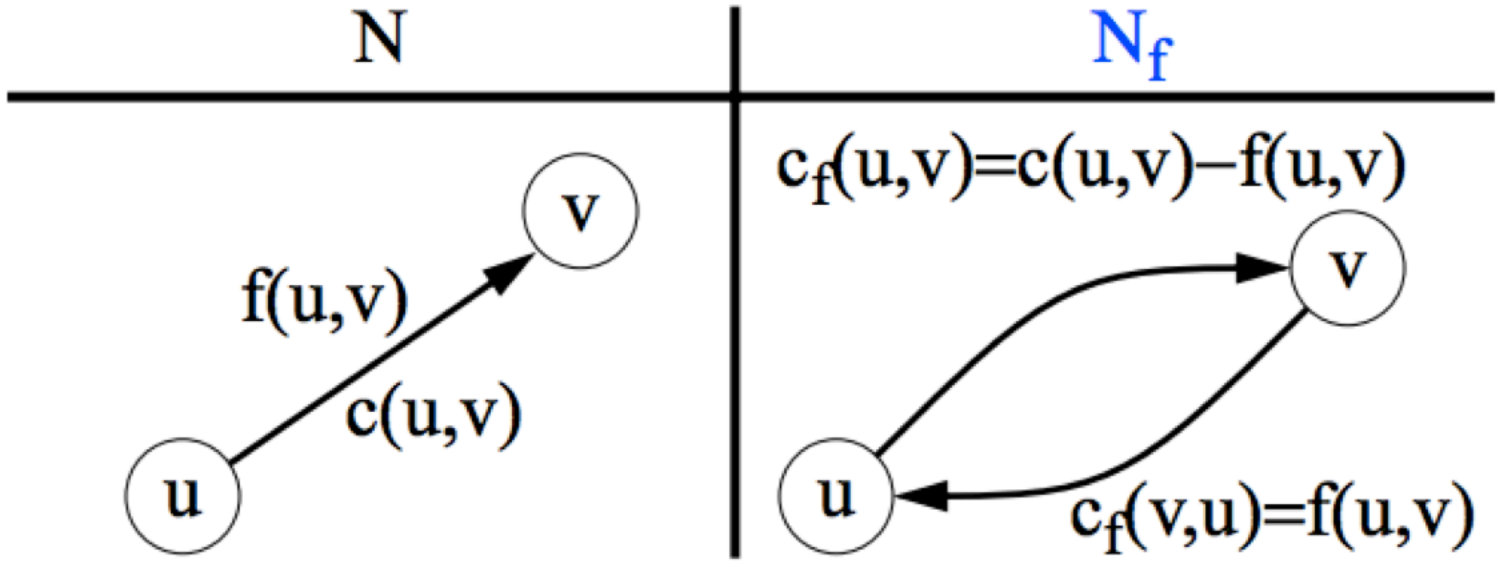
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E , \\ f(v,u) & \text{if } (v,u) \in E , \\ 0 & \text{otherwise} . \end{cases}$$

Residual Newtork

- Given a flow network $N = (V, E)$ and a flow f,
- The residual network of N induced by f is $N_f = (V, E_f)$ where

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

Residual Network



$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \,, \\ f(v,u) & \text{if } (v,u) \in E \,, \\ 0 & \text{otherwise} \,. \end{cases}$$

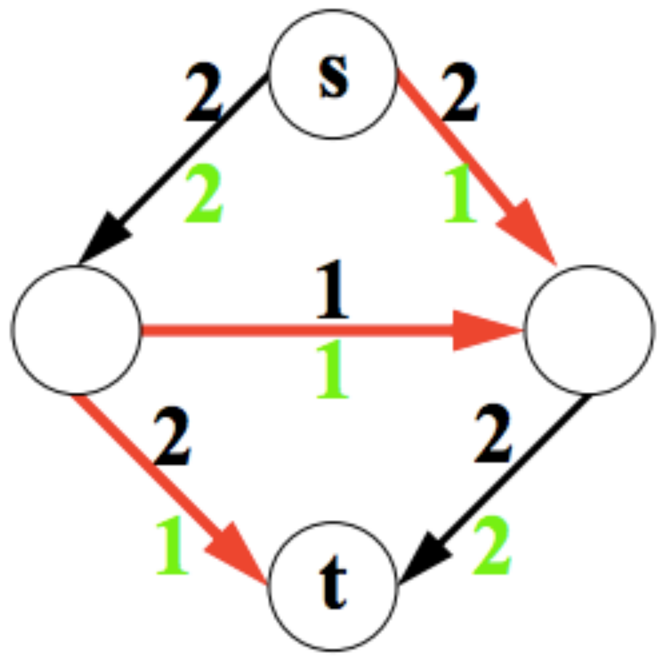
$$E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$$

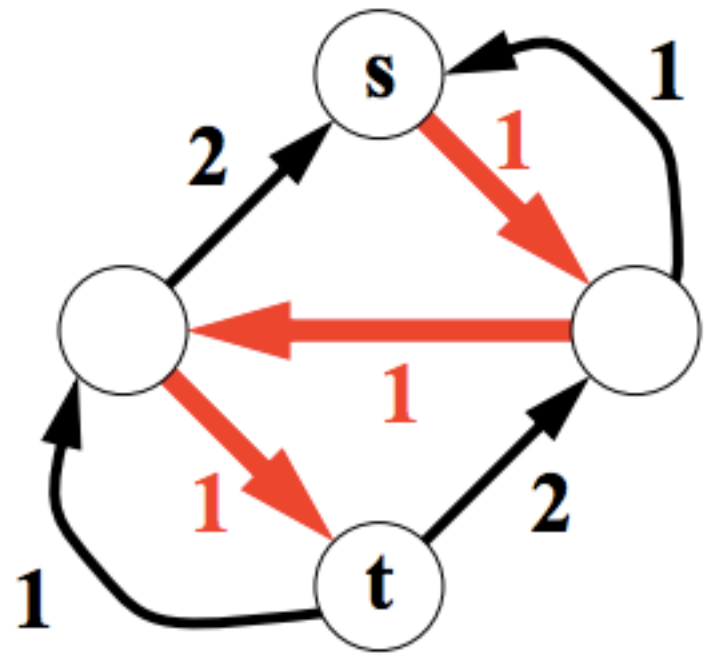
Residual Network

Augmenting path in network N $\longleftrightarrow$ Directed path in the residual network N_f

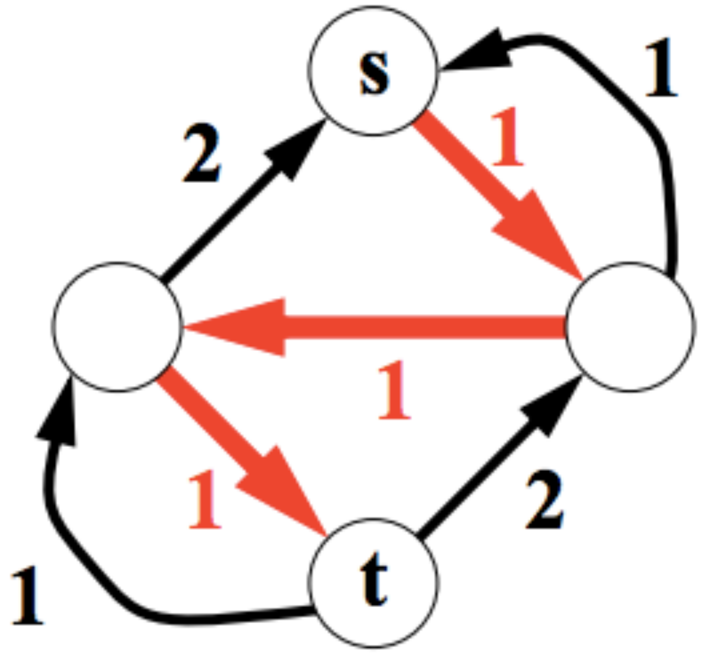
Residual Network

Augmenting path in network N ⟷ Directed path in the residual network N_f





Residual Network



Augmenting paths can be found performing a depth-first search on the residual network N_f

Ford & Fulkerson Algorithm

```
Part I: Setup
Start with null flow:
  $f(u,v) = 0 \; \forall \; (u,v) \in E$;
Initialize residual network:
  $N_f = N$;
```

Ford & Fulkerson Algorithm

Part II: Loop

```
repeat
   search for directed path p in $N_f$ from s to t
   if (path p found)
         $Df = \min \{c_f(u,v), (u,v) \in p\}$;
         for (each $(u,v) \in p$) do
            if (forward (u,v))
               $f(u,v) = f(u,v) + Df$;
            if (backward (u,v))
               $f(u,v) = f(u,v) - Df$;
         update $N_f$;
until (no augmenting path);
```

Time Complexity

- Ford-Fulkerson algorithm stops within finite rounds of the loop
- Within each iteration of the loop, the value of f increases by at least 1
- If f^* is the maximum flow, then the algorithm executes the loop at most $|f^*|$ times
- Within each iteration the path can be found using DFS or BFS – $O(|V| + |E|)$ -- $O(|E|)$
- Thus the running time is $O(|E|.|f^*|)$

Time Complexity

- The problem with the original algorithm, however, is that it is strongly dependent on the maximum flow value $|f^*|$
- For example, if $|f^*| = 2^n$, the algorithm may take exponential time
- Then, along came Edmonds & Karp

Max Flow: Improvement

- Theorem: [Edmonds & Karp, 1972]

- By using BFS, a maximum flow can be computed in time…

$$O(|V|.|E|^2)$$

Ford & Fulkerson Algorithm

- How can we prove that this algorithm is correct?

- *In other words, how do we know that when this algorithm terminates, we have actually found the maximum flow?*

CUTS

- A cut (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V - S$, such that $s \in S$ and $t \in T$

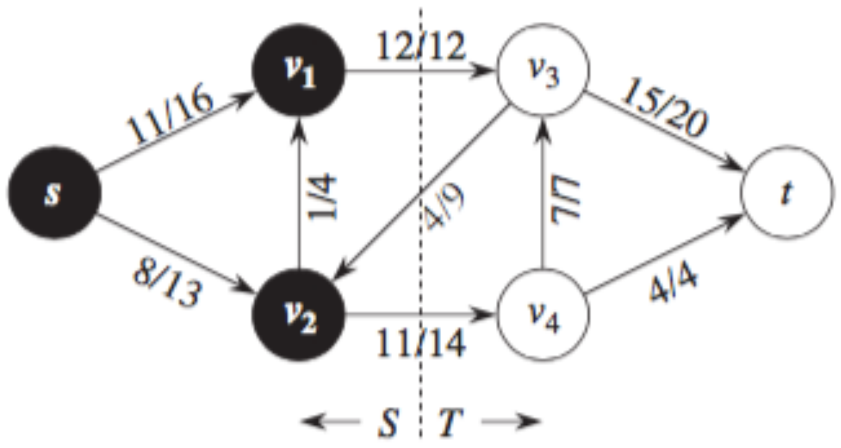


Figure 26.5 A cut (S, T) in the flow network of Figure 26.1(b), where $S = \{s, v_1, v_2\}$ and $T = \{v_3, v_4, t\}$. The vertices in S are black, and the vertices in T are white. The net flow across (S, T) is $f(S, T) = 19$, and the capacity is $c(S, T) = 26$.

CUTS

- Three Important points
  - Net flow across a cut
  - Capacity of a cut
  - And the relationship between the flow of the network and the capacity of a cut

Net Flow Across a Cut

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)\,.$$

Lemma 26.4

Let f be a flow in a flow network G with source s and sink t, and let (S,T) be any cut of G. Then the net flow across (S,T) is $f(S,T) = |f|$.

Cormen's – Ch: 26

CUTS

- Three Important points
  - ~~Net flow across a cut~~
  - Capacity of a cut
  - And the relationship between the flow of the network and the capacity of a cut

Cut Capacity

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v) .$$

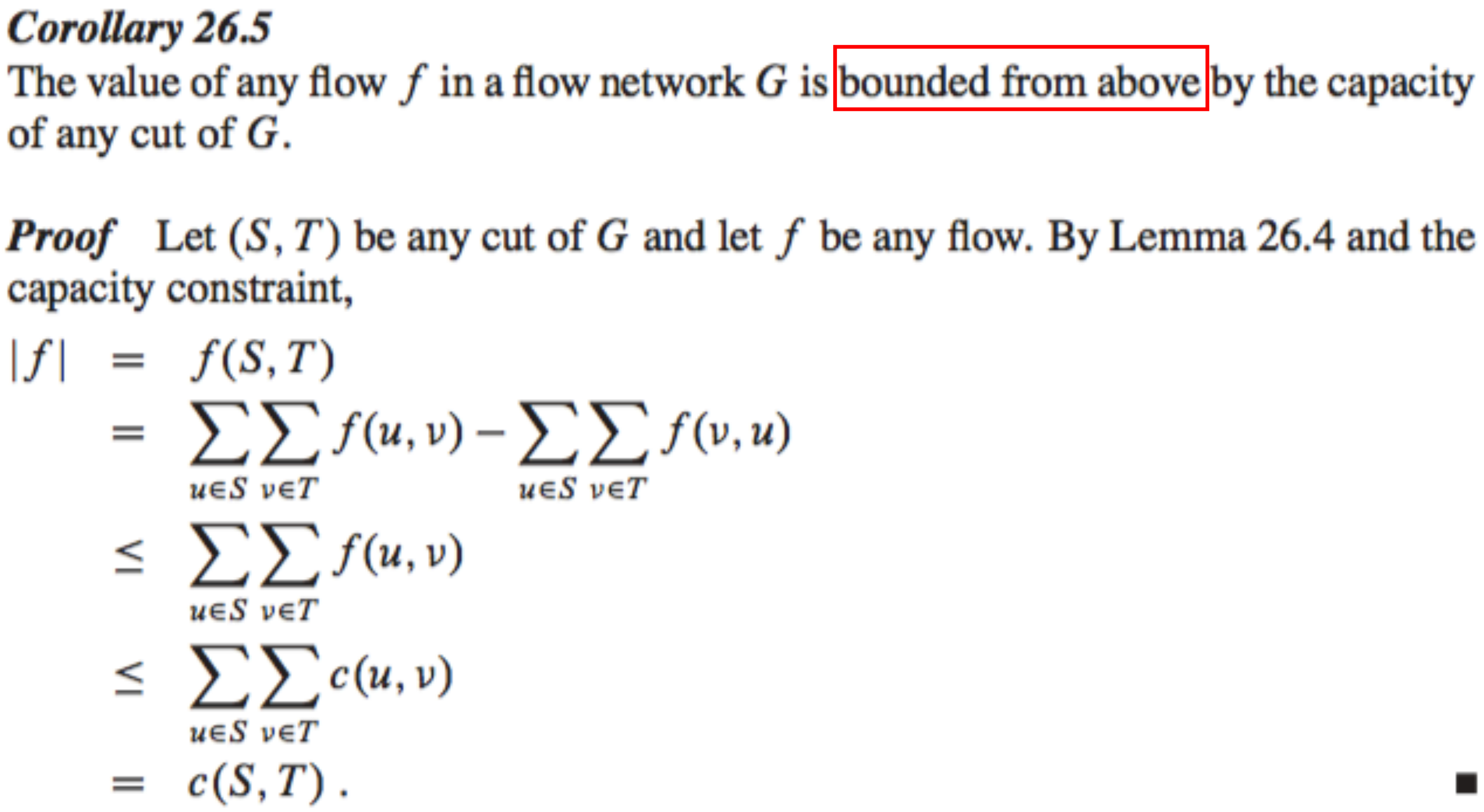
A ***minimum cut*** of a network is a cut whose capacity is minimum over all cuts of the network.

CUTS

- Three Important points

  - ~~Net flow across a cut~~
  - ~~Capacity of a cut~~
  - And the relationship between the flow of the network and the capacity of a cut

Relationship b/w flow and Capacity of a cut



Corollary 26.5
The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

Proof Let (S, T) be any cut of G and let f be any flow. By Lemma 26.4 and the capacity constraint,

$$\begin{aligned} |f| &= f(S,T) \\ &= \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{u\in S}\sum_{v\in T} f(v,u) \\ &\le \sum_{u\in S}\sum_{v\in T} f(u,v) \\ &\le \sum_{u\in S}\sum_{v\in T} c(u,v) \\ &= c(S,T)\,. \end{aligned}$$

■

That is …

(value of maximum flow)

=

(capacity of minimum cut)

Max-flow Min-cut Theorem

If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

Summary

- Flow networks
- Maximum flow
- Where can it be used?
- How to find maximum flow in flow networks?
  - ➢Residual network and augmenting paths
- Time complexity analysis
- Cuts
- Flow across a cut, and cut capacity
- Max-flow min-cut theorem