Data Structures & Algorithms

Adil M. Khan
Professor of Computer Science
Innopolis University

Recap

- Minimum Spanning Tree
  - Unweighted Graphs (BFS or DFS)
  - Weighted Graphs (Prim’s Algo & Kruskal’s Algorithm)
- Topological Sort
- Shortest Path
  - One-to-All (Dijkstra’s Algorithm)
  - All-to-All (Floyd Warshall’s Algorithm)

Objectives

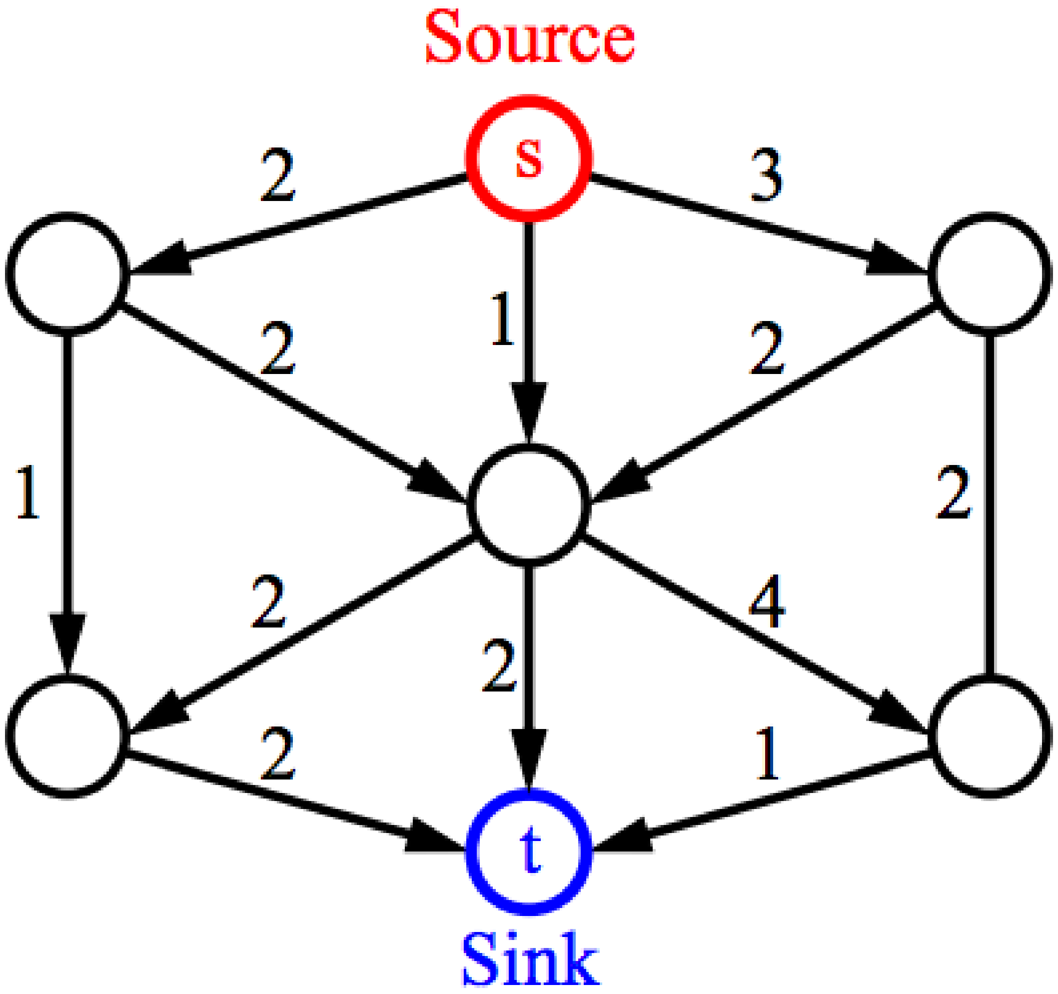
- Flow networks
- Maximum flow
- Where can it be used?
- How to find maximum flow in flow networks?
    - Residual network and augmenting paths
- Time complexity analysis
- Cuts
- Flow across a cut, and cut capacity
- Max-flow min-cut theorem

Flow Networks

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

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

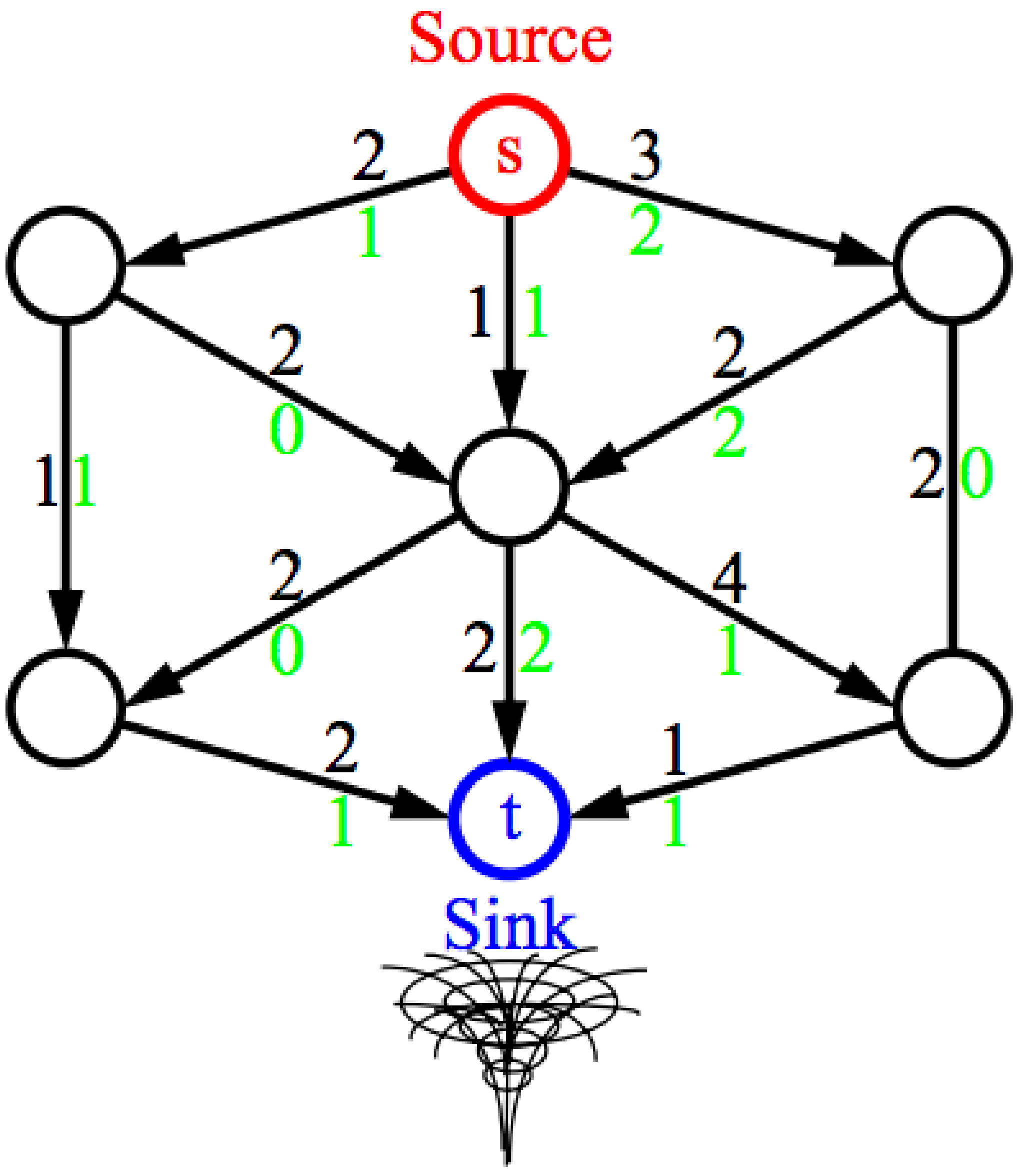
Flow Networks



Capacity and Flow

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

Capacity and Flow



The Logic of Flow

- Flow:

  flow(u,v) ∀ edge(u,v)

  -Capacity rule: ∀ edge (u,v)

  $$0 \leq \text{flow}(u,v) \leq \textbf{capacity}(u,v)$$

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

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

  -Value of flow:

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

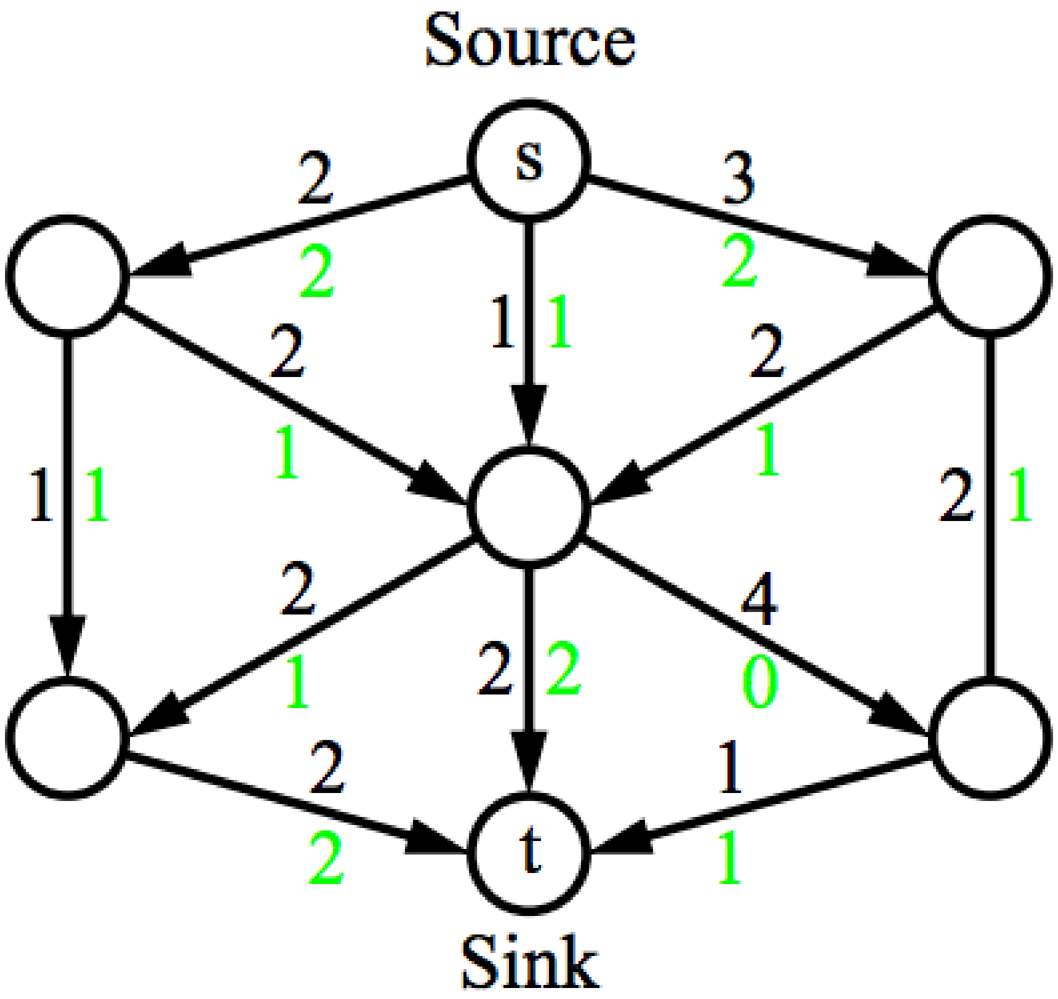
Max Flow Problem

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

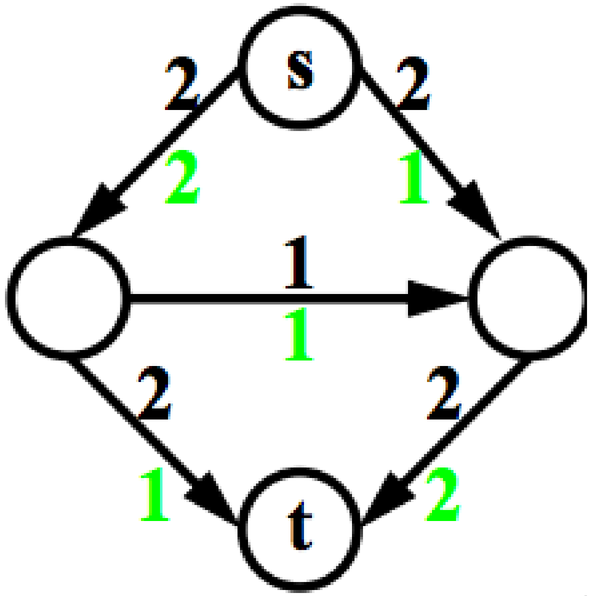
- Applications

  - Traffic movement
  - Hydraulic systems
  - Electrical circuits
  - Layout

Example of Max Flow

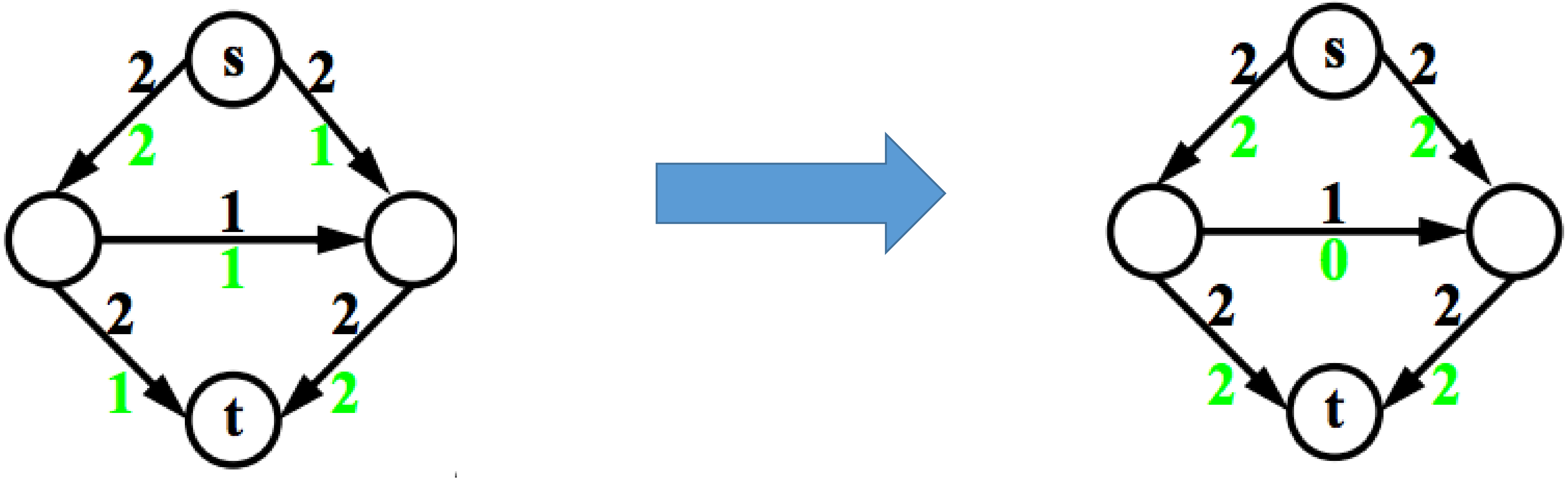


Increasing Flow



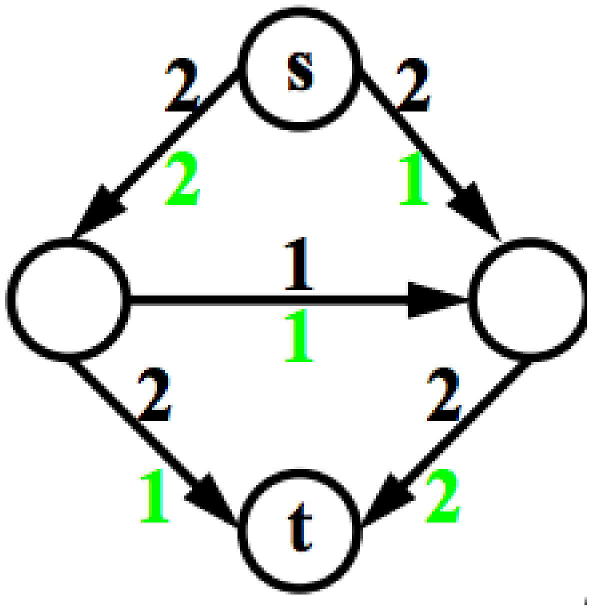
A network with a flow of value **3**

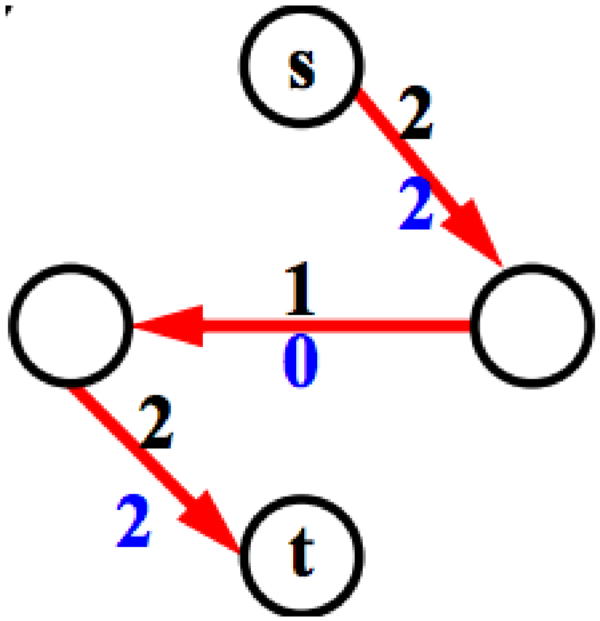
Increasing Flow



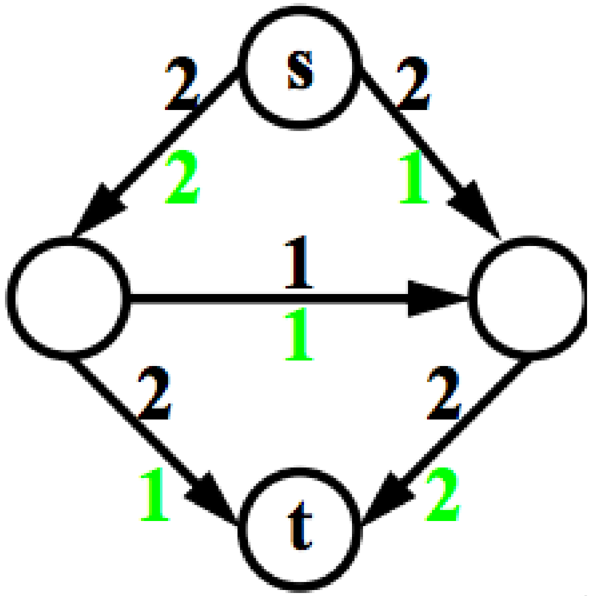
We have increased the flow value to 4!

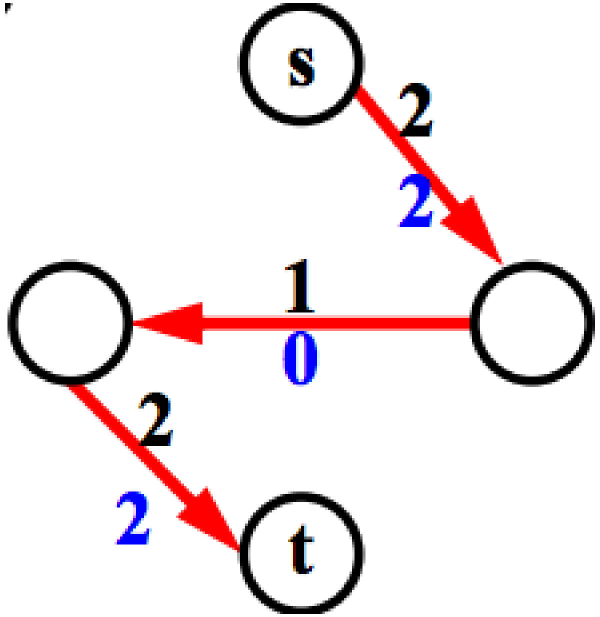
Understanding Increasing Flow





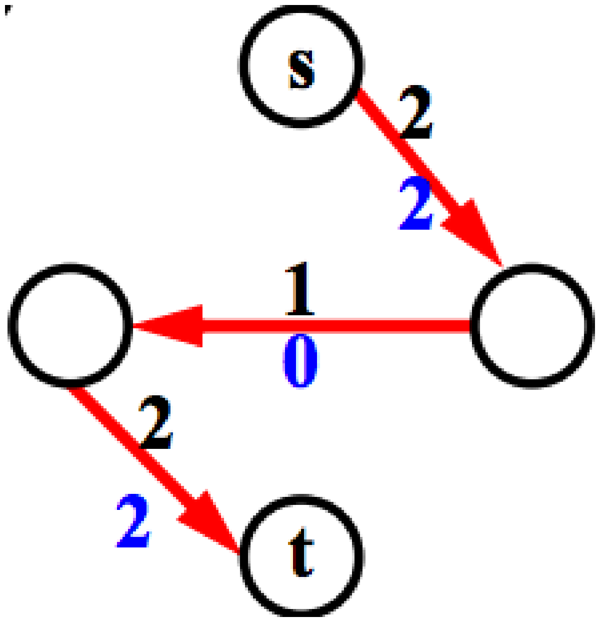
Understanding Increasing Flow





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

Augmenting Path



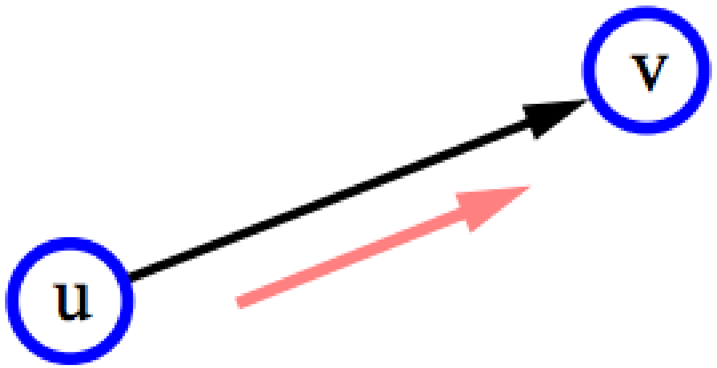
Augmenting
path

Augmenting Path

- **Forward Edges**

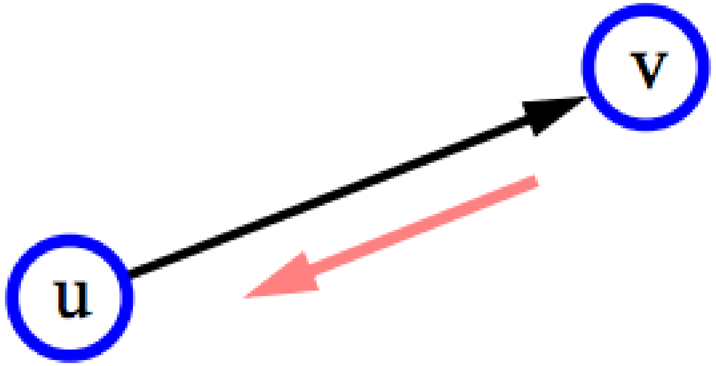
  flow(u,v) < **capacity**(u,v)

  flow can be increased!



Augmenting Path

- **Backward Edges**
  flow(u,v) > 0
  flow can be decreased!



Max Flow

A flow has maximum value
if and only if
it has no 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;
```

How to determine Augmenting Paths?

- For this, we will have to first define it

Augmenting Path

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

Let $< v_0 v_1 \ldots v_k >$ 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 $f(v_{i+1}, v_i) > 0$

Apply this rule to the figure on the board!

Things Worth Mentioning

- To augmenting f to get f’, let

$$\delta = \min_{i \in [0:k-1]} \begin{cases} c(v_i, v_{i+1}) - f(v_i, v_{i+1}) & \text{if } (v_i, v_{i+1}) \in E \\ f(v_{i+1}, v_i) & \text{if } (v_{i+1}, v_i) \in E \end{cases}$$

Things Worth Mentioning

- When augmenting f to get f', note that

1.

$0 \leq f'(u,v) \leq c(u,v), \quad \text{for all } (u,v) \in E$

2.

If $(v_{i-1}, v_i) \in E$ and $(v_i, v_{i+1}) \in E$, then both the in-flow and out-flow of v_i increase by δ

If $(v_{i-1}, v_i) \in E$ and $(v_{i+1}, v_i) \in E$, then both the in-flow and out-flow of v_i remain the same

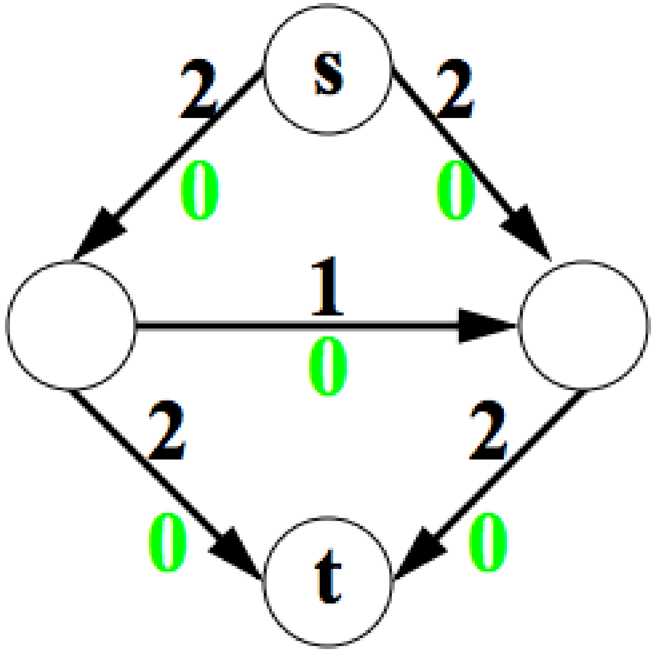
If $(v_i, v_{i-1}) \in E$ and $(v_i, v_{i+1}) \in E$, then both the in-flow and out-flow of v_i remain the same

If $(v_i, v_{i-1}) \in E$ and $(v_{i+1}, v_i) \in E$, then both the in-flow and out-flow of v_i decrease by δ

Back to Ford & Fulkerson Algorithm

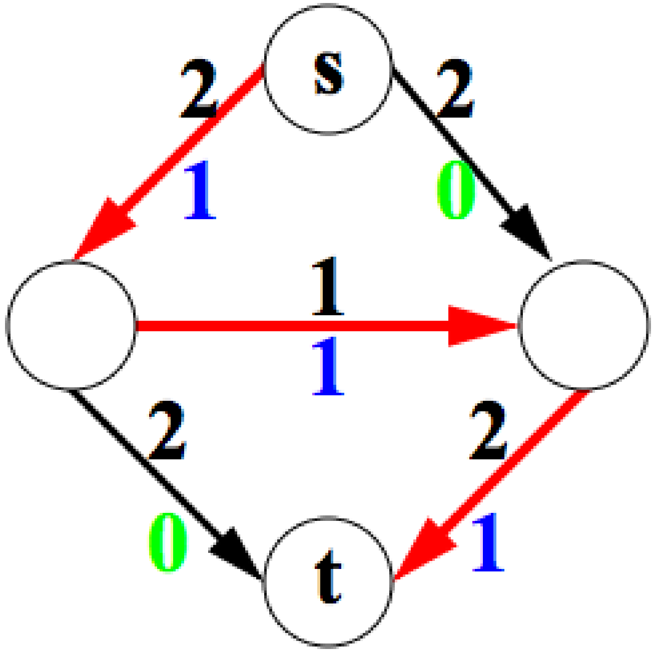
```
initialize network with null flow;
Method FindFlow
   if augmenting paths exist then
      find augmenting path;
      increase flow;
      recursive call to FindFlow;
```

Finding the Max Flow



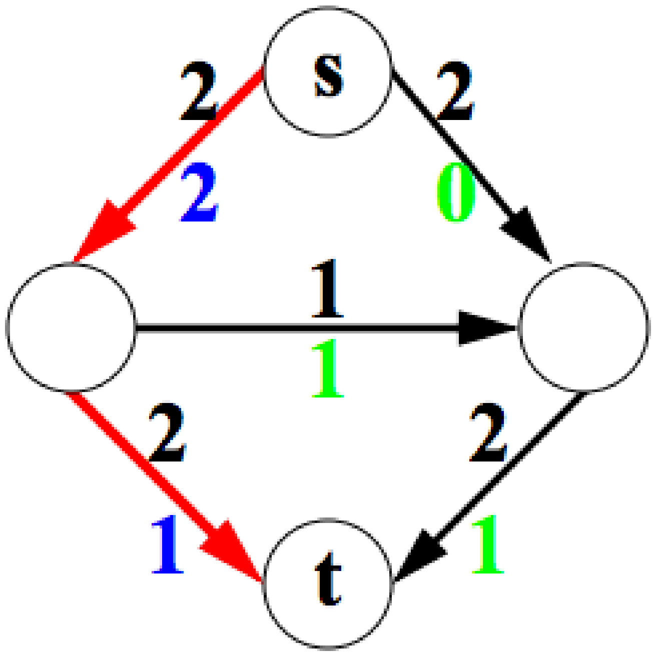
Initialize the network with a null flow. Note the **capacities above the edges**, and the flow below the edges.

Finding the Max Flow



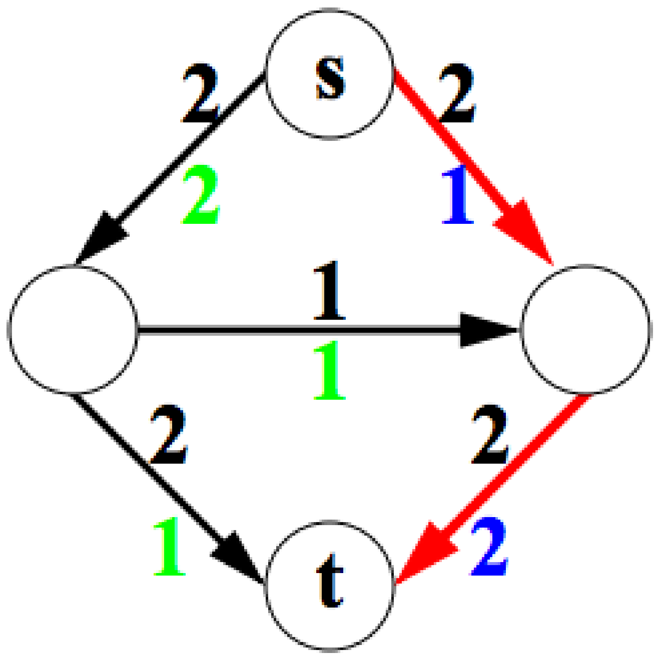
Send one unit of flow through the network. Note the path of the flow unit traced in red. The incremented flow values are in blue.

Finding the Max Flow



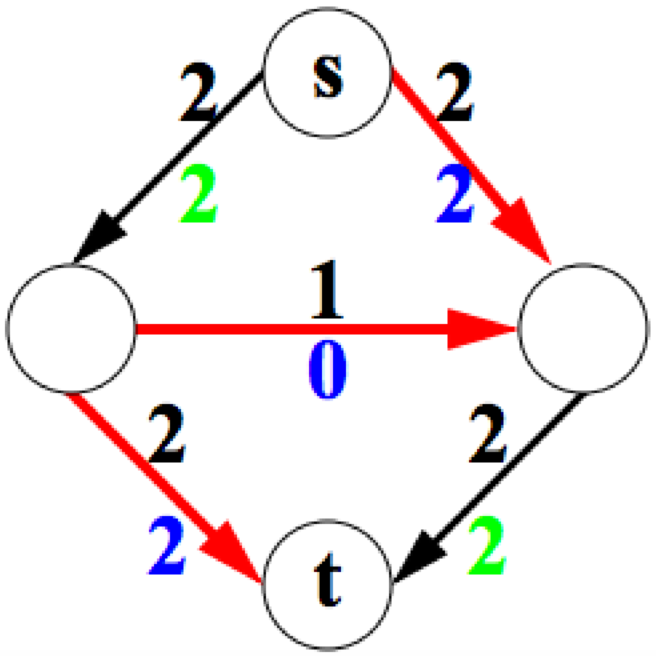
Send another unit of flow through the network.

Finding the Max Flow



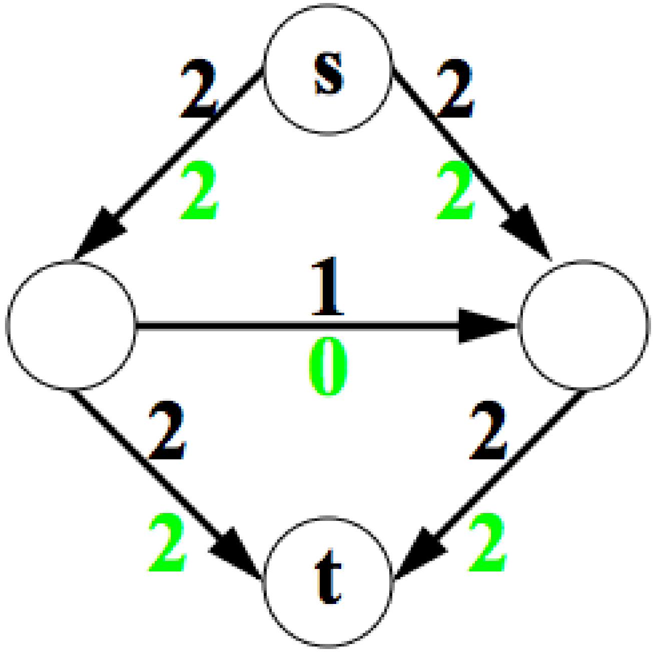
Send another unit of flow through the network. Note that there still exists an augmenting path, that can proceed *backward* against the directed central edge.

Finding the Max Flow



Send one unit of flow through the augmenting path. Note that there are no more augmenting paths. That means...

Finding the Max Flow



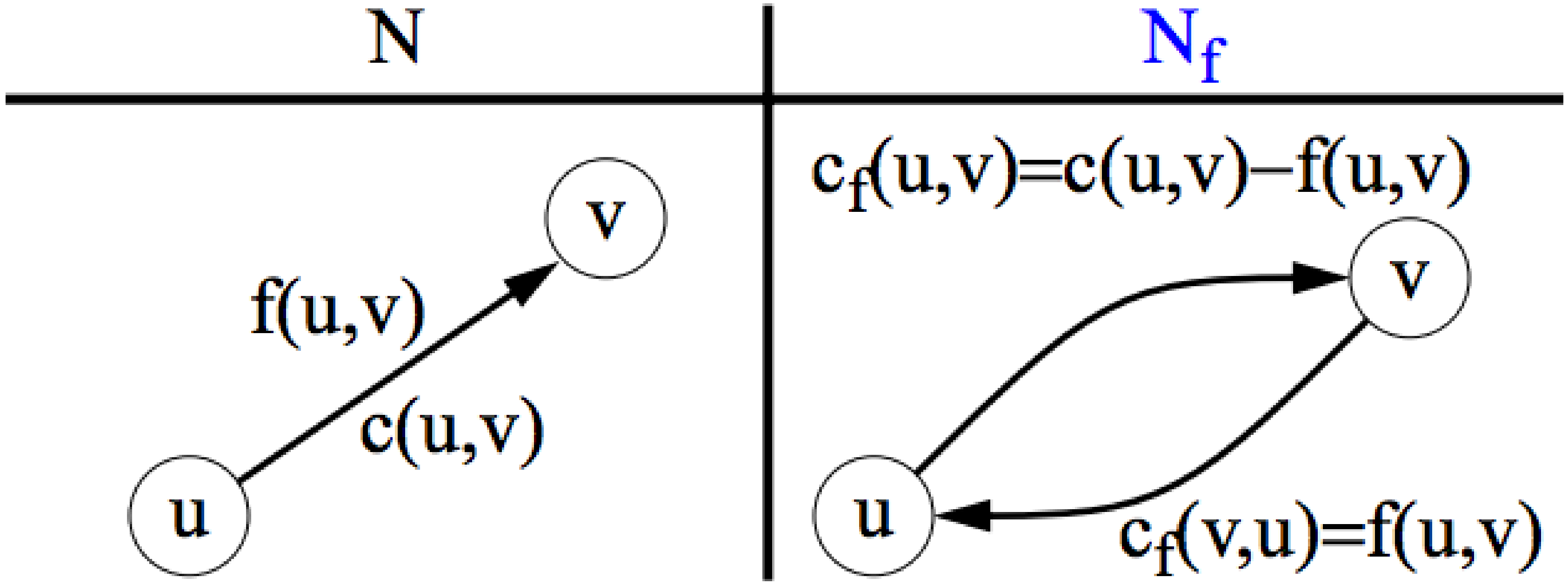
With the help of both Ford & Fulkerson, we have achieved this network's ***maximum flow***.

Improving the Algorithm

- Residual Network

Residual Network

- Residual Network $N_f = (V, Ef, cf, s, t)$



Residual Network

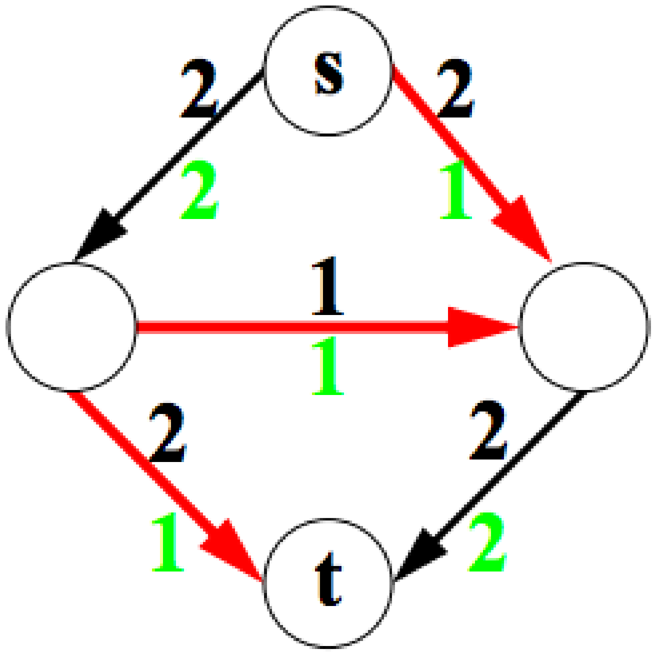
- In the residual network N_f, all edges (w,z) with capacity cf(w,z) = 0 are removed.

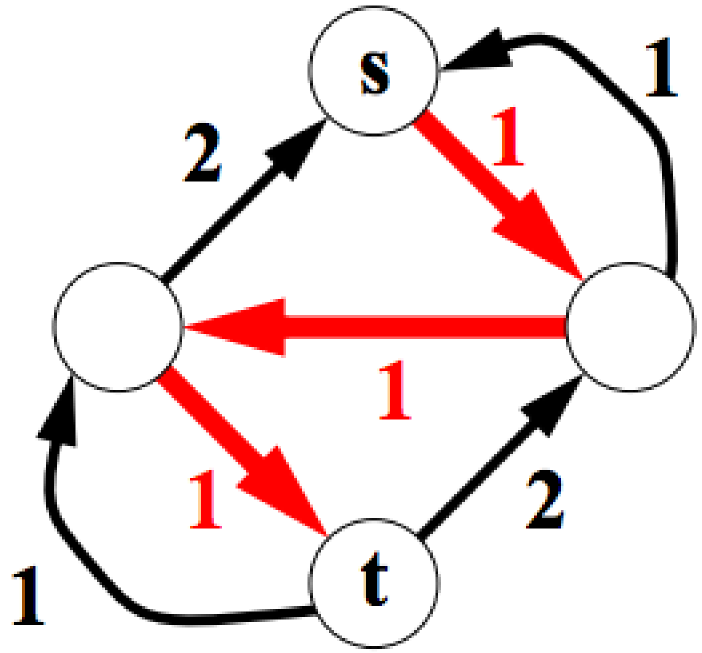
Residual Network

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

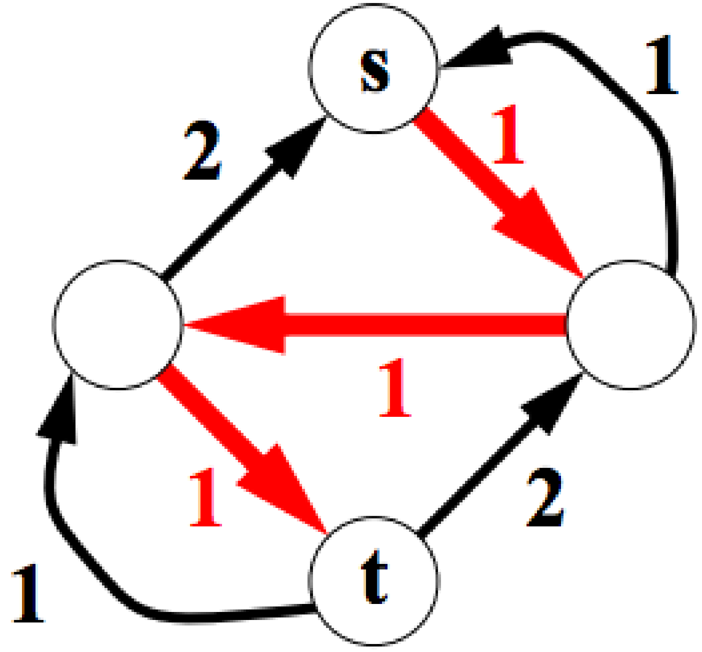
Residual Network

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

Improved Algorithm

Part I: Setup

Start with null flow:

$f(u,v) = 0 \ \forall \ (u,v) \in E;$

Initialize residual network:

$N_f = N;$

Improved 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) ∈ p};
         for (each (u,v) ∈ 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…

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

CUTS

- What is a *cut*?

- 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 $t \in T$

Net Flow Across a Cut

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

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.

Example

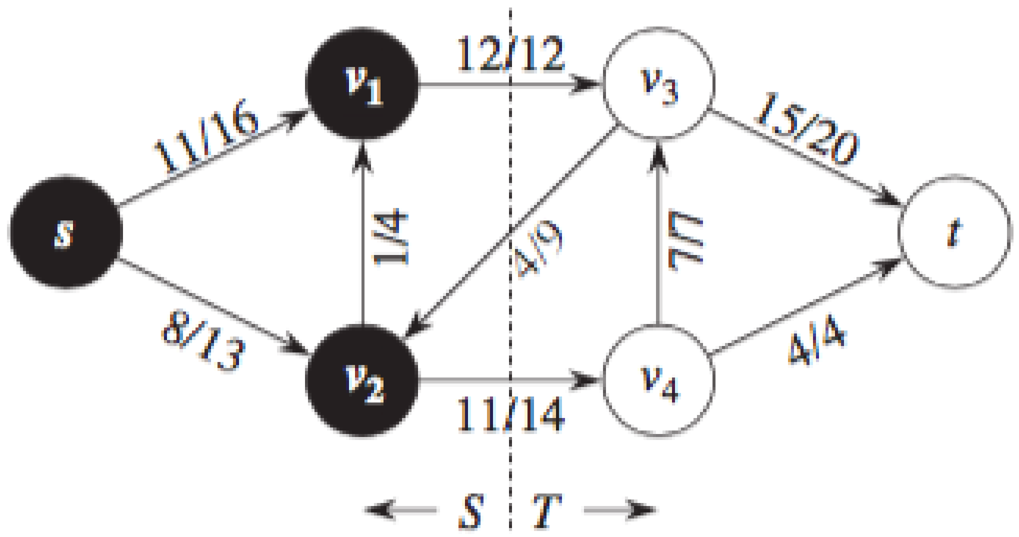


Figure 26.5 A cut (S, T) in the flow network of Figure 26.1(b), where $S = \{s, v_1, v_2\}$ and $T = \{v_3, v_4, t\}$. The vertices in S are black, and the vertices in T are white. The net flow across (S, T) is $f(S, T) = 19$, and the capacity is $c(S, T) = 26$.

Important Points

For a given flow f , the net flow across any cut is the same, and it equals $|f|$, the value of the flow

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 $f(S, T) = |f|$.

Proof can be found in Cormen's – Ch: 26

Important Points

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.

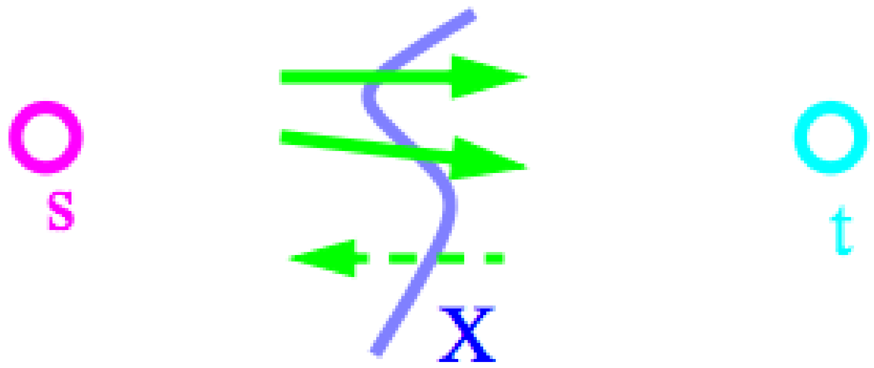
Proof Let (S, T) be any cut of G and let f be any flow. By Lemma 26.4 and the capacity constraint,

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

■

Intuitively

- Let f be a flow of value |f| and X a cut ofcapacity |X|. Then, |f| <=|X|.



- Hence, if we find a flow f* of value |f*| and a cut X* of capacity |X*|=|f*|, then f* must be the maximum flow and X* must be the minimum cut.

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.

Proof

- Intuitive explanation on the whiteboard!

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.

Did we achieve today's Objectives?

- 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