# Design and Analysis of Algorithms

Presented by Dr. Li Ning

Shenzhen Institutes of Advanced Technology, Chinese Academy of Science
Shenzhen, China

1 Network Flow

2 Example

3 Network Cut

4 Edge Disjoint Paths

5 Bipartite Matching

# Network Flow

# Network

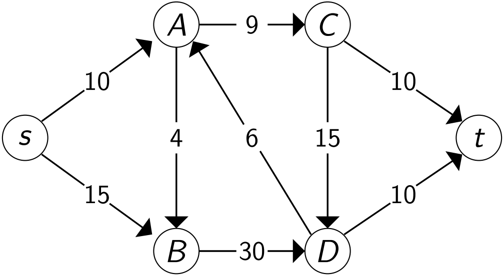
## (Flow) Network

- A directed graph $G = (V, E)$
- Source node $s \in V$: edges out
- Sink node $t \in V$: edges in
- Edge capacity $c(u, v)$, for $(u, v) \in E$

# Network

**(Flow) Network**

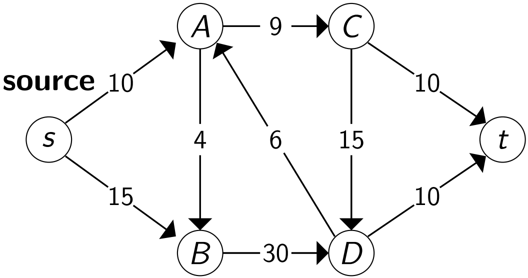
- A directed graph $G = (V, E)$
- Source node $s \in V$: edges out
- Sink node $t \in V$: edges in
- Edge capacity $c(u, v)$, for $(u, v) \in E$



# Network

**(Flow) Network**

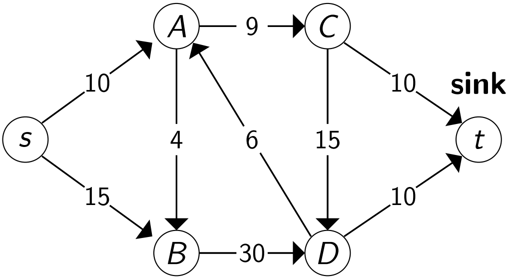
- A directed graph $G = (V, E)$
- Source node $s \in V$: edges out
- Sink node $t \in V$: edges in
- Edge capacity $c(u, v)$, for $(u, v) \in E$



# Network

## (Flow) Network

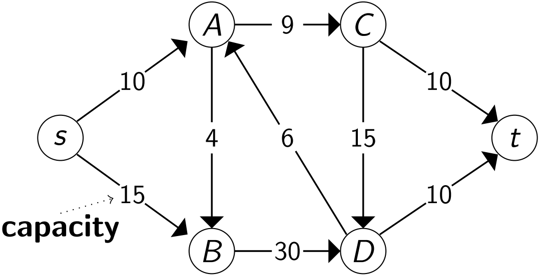
- A directed graph $G = (V, E)$
- Source node $s \in V$: edges out
- Sink node $t \in V$: edges in
- Edge capacity $c(u, v)$, for $(u, v) \in E$



# Network

**(Flow) Network**

- A directed graph $G = (V, E)$
- Source node $s \in V$: edges out
- Sink node $t \in V$: edges in
- Edge capacity $c(u, v)$, for $(u, v) \in E$



# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u, v) \in [0, c(u, v)] : (u, v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,u)\in E} f(v, u)$, for all $v \in V \setminus \{s, t\}$

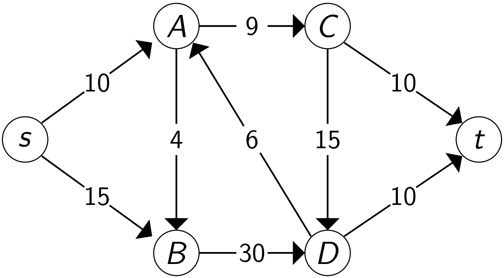
The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s, u)$.

# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u,v) \in [0, c(u,v)] : (u,v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$, for all $v \in V \setminus \{s,t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s,u)$.

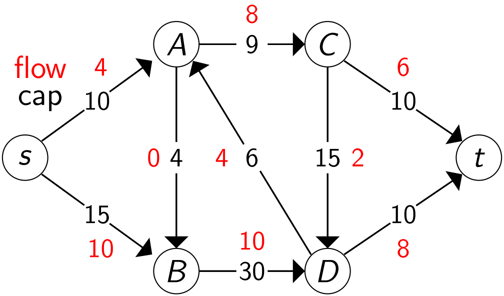


# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u,v) \in [0, c(u,v)] : (u,v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$, for all $v \in V \setminus \{s,t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s,u)$.

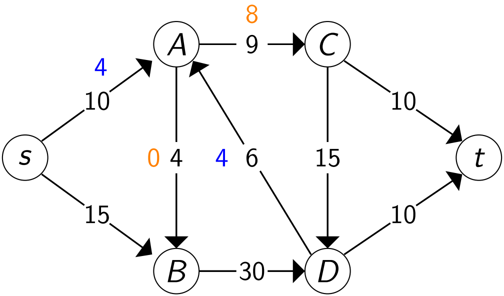


# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u,v) \in [0, c(u,v)] : (u,v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$, for all $v \in V \setminus \{s,t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s,u)$.

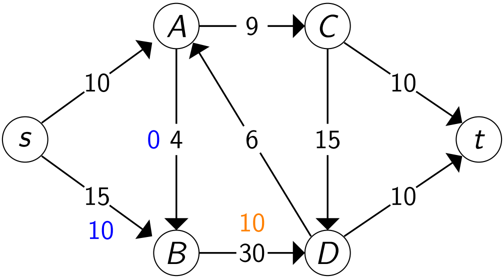


# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u, v) \in [0, c(u, v)] : (u, v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,u)\in E} f(v, u)$, for all $v \in V \setminus \{s, t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s, u)$.

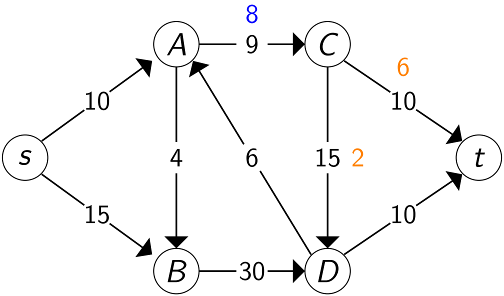


# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u,v) \in [0, c(u,v)] : (u,v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$, for all $v \in V \setminus \{s,t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s,u)$.

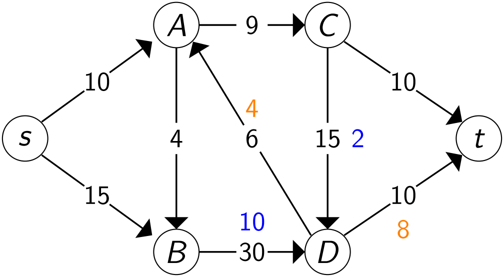


# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u,v) \in [0, c(u,v)] : (u,v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$, for all $v \in V \setminus \{s,t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s,u)$.

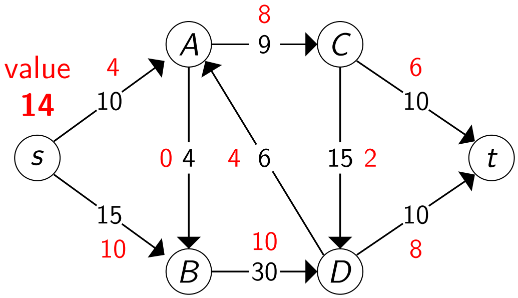


# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u, v) \in [0, c(u, v)] : (u, v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,u)\in E} f(v, u)$, for all $v \in V \setminus \{s, t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s, u)$.

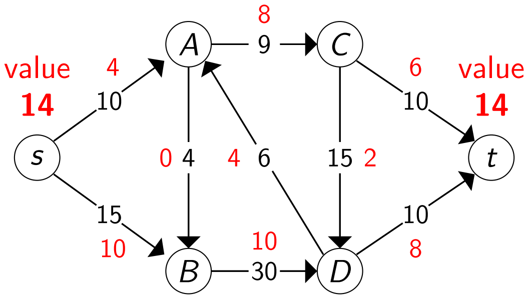


# Flow

**Flow**: given a network, a valid flow function $f$ satisfies

- $f(u,v) \in [0, c(u,v)] : (u,v) \in E \to \mathbb{R}^*$
- $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$, for all $v \in V \setminus \{s,t\}$

The value of a flow $f$ is defined by $v(f) = \sum_{(s,u)\in E} f(s,u)$.



# Flow

$\sum_{(s,u)\in E} f(s,u) = \sum_{(u,t)\in E} f(u,t)$

**Proof**:

- $\sum_{v\in V\setminus\{s,t\}} \left(\sum_{(u,v)\in E} f(u,v) - \sum_{(v,u)\in E} f(v,u)\right) = 0$

# Flow

$\sum_{(s,u)\in E} f(s,u) = \sum_{(u,t)\in E} f(u,t)$

**Proof**:

- $\sum_{v\in V\setminus\{s,t\}} \left( \sum_{(u,v)\in E} f(u,v) - \sum_{(v,u)\in E} f(v,u) \right) = 0$
- For every edge $(u,v) \in E$
  - if $u \neq s$ and $v \neq t$: $f(u,v) - f(u,v) = 0$
  - if $u = s$: $f(s,v)$
  - if $v = t$: $-f(u,t)$

# Flow

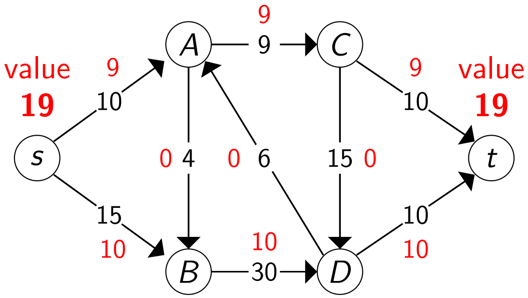
$\sum_{(s,u)\in E} f(s,u) = \sum_{(u,t)\in E} f(u,t)$

**Proof**:

- $\sum_{v\in V\setminus\{s,t\}} \left(\sum_{(u,v)\in E} f(u,v) - \sum_{(v,u)\in E} f(v,u)\right) = 0$
- For every edge $(u,v) \in E$
  - if $u \neq s$ and $v \neq t$: $f(u,v) - f(u,v) = 0$
  - if $u = s$: $f(s,v)$
  - if $v = t$: $-f(u,t)$
- $$\begin{aligned} 0 &= \sum_{v\in V\setminus\{s,t\}} \left(\sum_{(u,v)\in E} f(u,v) - \sum_{(v,u)\in E} f(v,u)\right) \\ &= \sum_{(s,u)\in E} f(s,u) - \sum_{(u,t)\in E} f(u,t) \end{aligned}$$

# Maximum Flow

**Problem**: given a flow network, find the flow of the maximum value.



# Maximum Flow: Greedy Algorithm

**Problem**: given a flow network, find the flow of the maximum value.

**Have a try**

# Maximum Flow: Greedy Algorithm

**Problem**: given a flow network, find the flow of the maximum value.

**Have a try**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$

# Maximum Flow: Greedy Algorithm

**Problem**: given a flow network, find the flow of the maximum value.

**Have a try**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Find a path from $s$ to $t$, such that $f(u, v) < cap(u, v)$ for all $(u, v)$ in the path.

# Maximum Flow: Greedy Algorithm

**Problem**: given a flow network, find the flow of the maximum value.

**Have a try**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Find a path from $s$ to $t$, such that $f(u, v) < cap(u, v)$ for all $(u, v)$ in the path.
3. augment the flow along the path.

# Maximum Flow: Greedy Algorithm

**Problem**: given a flow network, find the flow of the maximum value.

**Have a try**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Find a path from $s$ to $t$, such that $f(u, v) < cap(u, v)$ for all $(u, v)$ in the path.
3. augment the flow along the path.
4. Repeat step 2 and 3, until no such path can be found.

# Maximum Flow: Greedy Algorithm

**Problem**: given a flow network, find the flow of the maximum value.

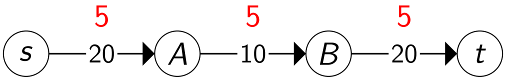
**Have a try**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Find a path from $s$ to $t$, such that $f(u, v) < cap(u, v)$ for all $(u, v)$ in the path.
3. augment the flow along the path. How?
4. Repeat step 2 and 3, until no such path can be found.

# Maximum Flow: Greedy Algorithm

Augment path from $s$ to $t$.

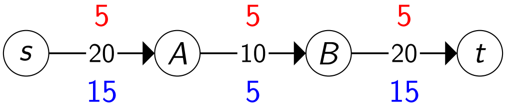
- find the edge $(u, v)$ of the minimum remaining capacity
- increase $f(u, v)$ to $cap(u, v)$
- increase the flow on the other edge by $cap(u, v) - f(u, v)$



# Maximum Flow: Greedy Algorithm

Augment path from $s$ to $t$.

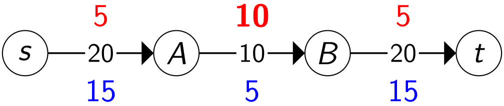
- find the edge $(u, v)$ of the minimum remaining capacity
- increase $f(u, v)$ to $cap(u, v)$
- increase the flow on the other edge by $cap(u, v) - f(u, v)$



# Maximum Flow: Greedy Algorithm

Augment path from $s$ to $t$.

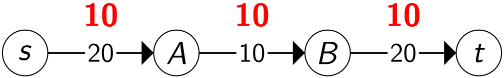
- find the edge $(u, v)$ of the minimum remaining capacity
- increase $f(u, v)$ to $cap(u, v)$
- increase the flow on the other edge by $cap(u, v) - f(u, v)$



# Maximum Flow: Greedy Algorithm

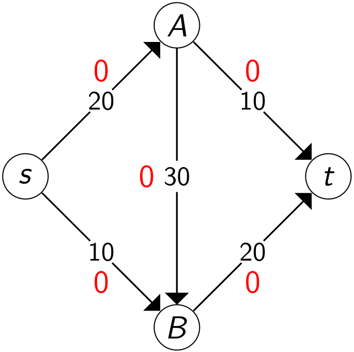
Augment path from $s$ to $t$.

- find the edge $(u, v)$ of the minimum remaining capacity
- increase $f(u, v)$ to $cap(u, v)$
- increase the flow on the other edge by $cap(u, v) - f(u, v)$



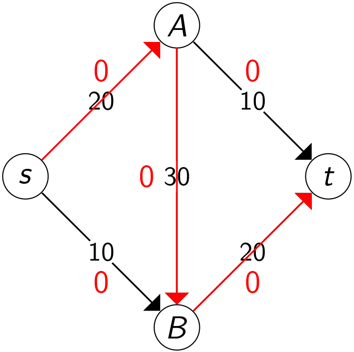
# Maximum Flow: Greedy Algorithm

**Greedy**



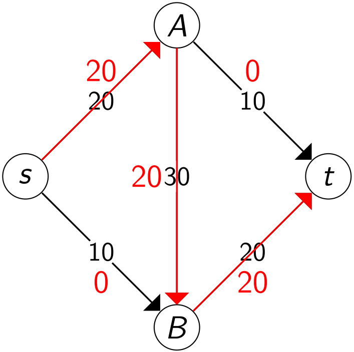
# Maximum Flow: Greedy Algorithm

**Greedy**

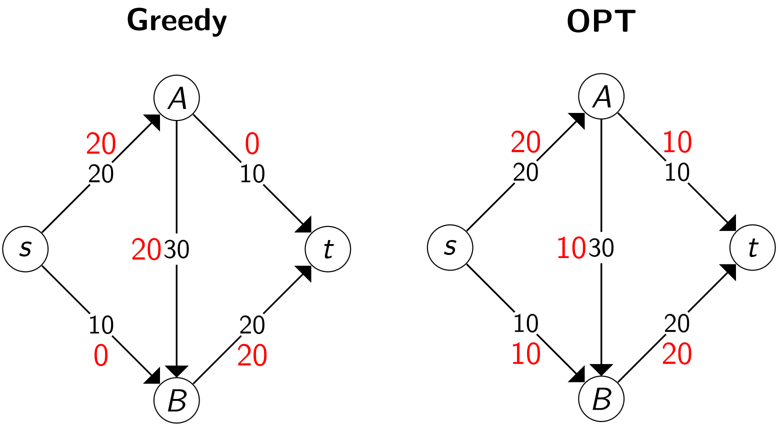


# Maximum Flow: Greedy Algorithm

**Greedy**

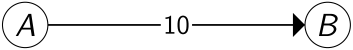


# Maximum Flow: Greedy Algorithm



# Maximum Flow: Greedy Algorithm

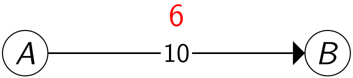
**Undo** the flow. Why and how?



# Maximum Flow: Greedy Algorithm

**Undo** the flow. Why and how?

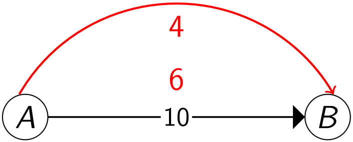
- Original:
  $cap(u, v), f(u, v) = 0$
- Augmented:
  $0 \leq f(u, v) \leq cap(u, v)$



# Maximum Flow: Greedy Algorithm

**Undo** the flow. Why and how?

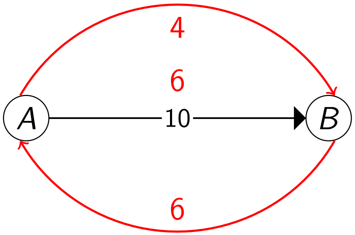
- Original:
  $cap(u, v), f(u, v) = 0$
- Augmented:
  $0 \leq f(u, v) \leq cap(u, v)$
- Residual edges:
  - $(u, v) : cap(u, v) - f(u, v)$



# Maximum Flow: Greedy Algorithm

**Undo** the flow. Why and how?

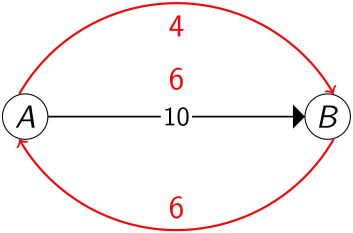
- Original:
  $cap(u, v), f(u, v) = 0$
- Augmented:
  $0 \leq f(u, v) \leq cap(u, v)$
- Residual edges:
  - $(u, v) : cap(u, v) - f(u, v)$
  - $(v, u) : f(u, v)$

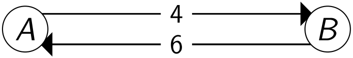


# Maximum Flow: Greedy Algorithm

**Undo** the flow. Why and how?

- Original:
  $cap(u, v), f(u, v) = 0$
- Augmented:
  $0 \leq f(u, v) \leq cap(u, v)$
- Residual edges:
  - $(u, v) : cap(u, v) - f(u, v)$
  - $(v, u) : f(u, v)$



# Maximum Flow: Ford-Fulkerson's Algorithm

$G^f = (V, E^f)$: residual graph with respect to flow $f$

- each edge $(u, v) \in E$, implies
  - edge $(u, v) \in E^f$ with capacity $cap(u, v) - f(u, v)$
  - edge $(v, u) \in E^f$ with capacity $f(u, v)$

# Maximum Flow: Ford-Fulkerson's Algorithm

$G^f = (V, E^f)$: residual graph with respect to flow $f$

- each edge $(u, v) \in E$, implies
  - edge $(u, v) \in E^f$ with capacity $cap(u, v) - f(u, v)$
  - edge $(v, u) \in E^f$ with capacity $f(u, v)$

**Ford-Fulkerson**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$

# Maximum Flow: Ford-Fulkerson's Algorithm

$G^f = (V, E^f)$: residual graph with respect to flow $f$

- each edge $(u, v) \in E$, implies
  - edge $(u, v) \in E^f$ with capacity $cap(u, v) - f(u, v)$
  - edge $(v, u) \in E^f$ with capacity $f(u, v)$

**Ford-Fulkerson**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Update $G^f$ with respect to $f$

# Maximum Flow: Ford-Fulkerson's Algorithm

$G^f = (V, E^f)$: residual graph with respect to flow $f$

- each edge $(u, v) \in E$, implies
  - edge $(u, v) \in E^f$ with capacity $cap(u, v) - f(u, v)$
  - edge $(v, u) \in E^f$ with capacity $f(u, v)$

**Ford-Fulkerson**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Update $G^f$ with respect to $f$
3. In $G^f$, find a path from $s$ to $t$.

# Maximum Flow: Ford-Fulkerson's Algorithm

$G^f = (V, E^f)$: residual graph with respect to flow $f$

- each edge $(u, v) \in E$, implies
  - edge $(u, v) \in E^f$ with capacity $cap(u, v) - f(u, v)$
  - edge $(v, u) \in E^f$ with capacity $f(u, v)$

**Ford-Fulkerson**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Update $G^f$ with respect to $f$
3. In $G^f$, find a path from $s$ to $t$.
4. Augment the flow along the path.

# Maximum Flow: Ford-Fulkerson's Algorithm

$G^f = (V, E^f)$: residual graph with respect to flow $f$

- each edge $(u, v) \in E$, implies
  - edge $(u, v) \in E^f$ with capacity $cap(u, v) - f(u, v)$
  - edge $(v, u) \in E^f$ with capacity $f(u, v)$

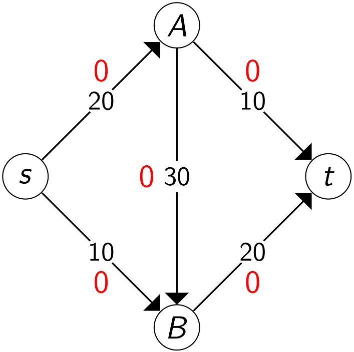
**Ford-Fulkerson**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$
2. Update $G^f$ with respect to $f$
3. In $G^f$, find a path from $s$ to $t$.
4. Augment the flow along the path.
5. Repeat step 2 and 4, until no such path can be found.

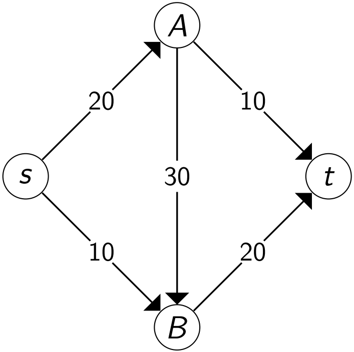
# Example

# Example

**Origin**

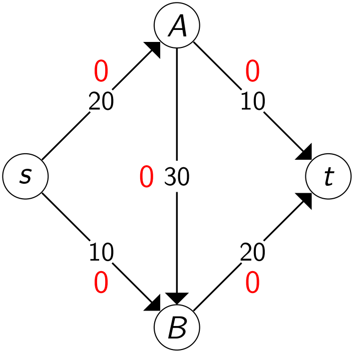



**Residual**

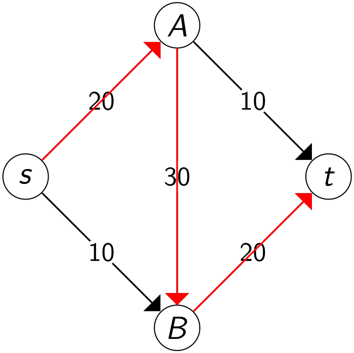


# Example

**Origin**

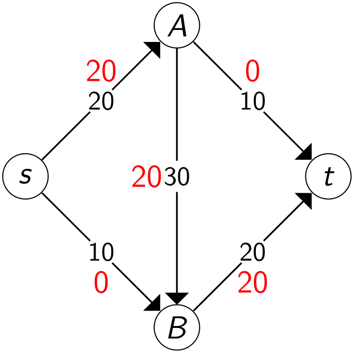



**Residual**

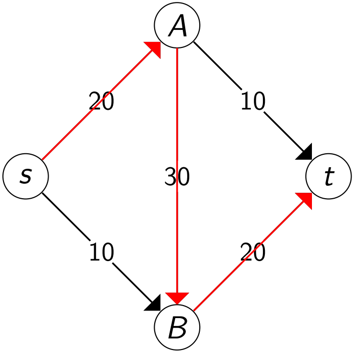


# Example

**Origin**

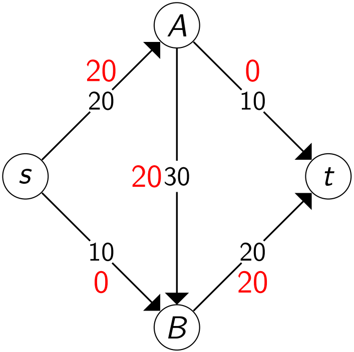



**Residual**

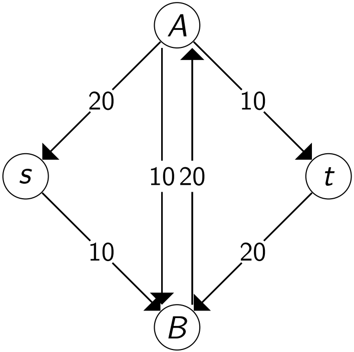


# Example

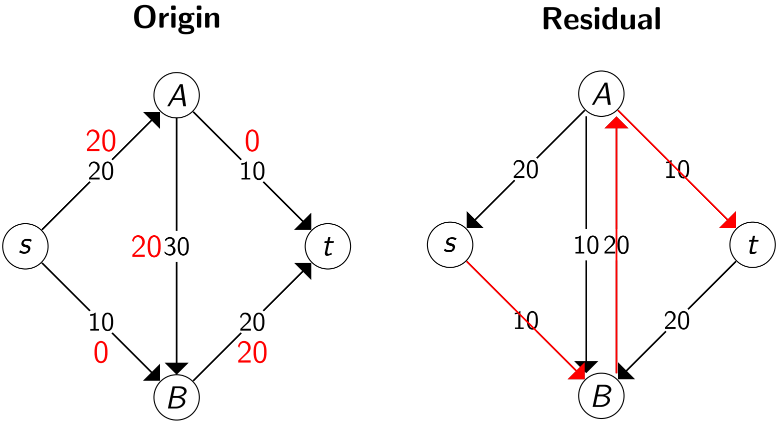
**Origin**



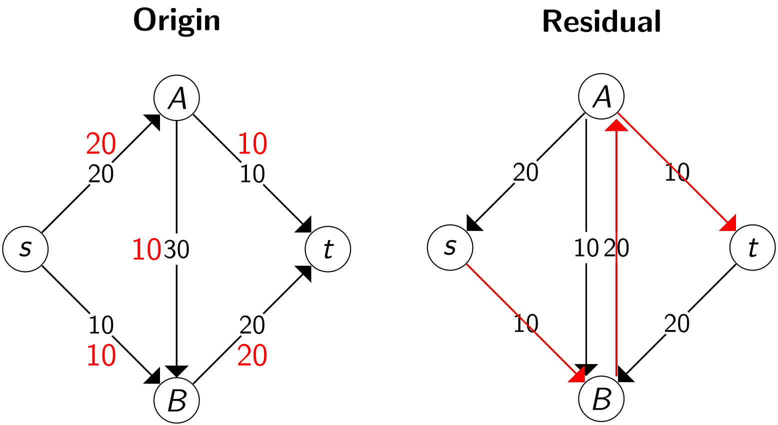

**Residual**



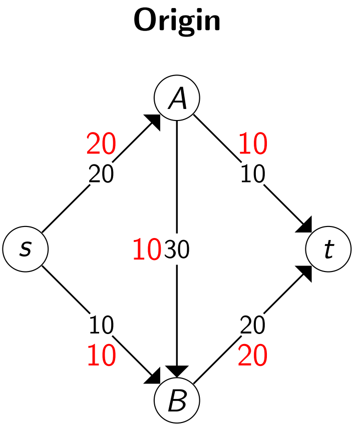
# Example

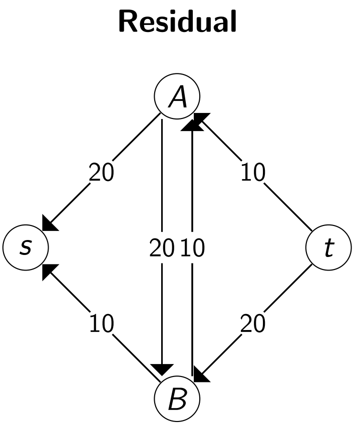

Origin
A
s
t
B
20
20
0
10
20 30
10
0
20
20
Residual
A
s
t
B
20
10
10 20
10
20

# Example


Origin
A
B
s
t
20
20
10
10
10
30
10
10
20
20
Residual
A
B
s
t
20
10
10
20
10
20

# Example


Origin
A
s
t
B
20
20
10
10
10
30
10
10
20
20


Residual
A
s
t
B
20
10
20
10
10
20

# Network Cut

# Cut

**Cut**: given a network with

- directed graph $G = (V, E)$
- source $s \in V$
- sink $t \in V$
- capacity $cap(u, v)$ for all $(u, v) \in E$

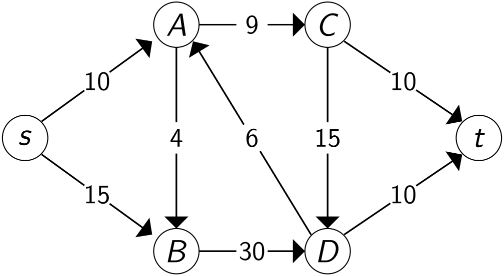
A $s - t$ cut is a partition $(V_s, V_t)$ of $V$, such that

- $s \in V_s$
- $t \in V_t$

# Cut

**Capacity of the cut**: for a cut $(V_s, V_t)$, its capacity is defined by

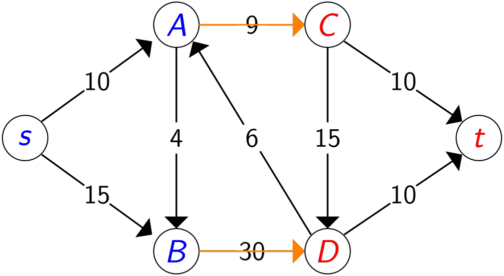
$$cap(A, B) = \sum_{(u,v) \in E, u \in V_s, v \in V_t} cap(u, v)$$



# Cut

**Capacity of the cut**: for a cut $(V_s, V_t)$, its capacity is defined by

$$cap(A, B) = \sum_{(u,v)\in E, u\in V_s, v\in V_t} cap(u, v)$$

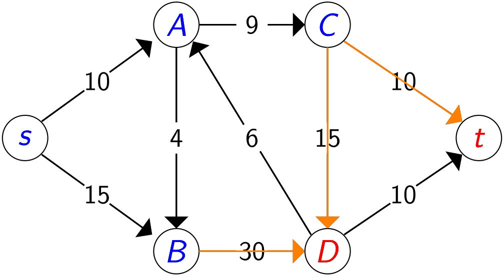



$\text{cap}(A, B) = 39$

# Cut

**Capacity of the cut**: for a cut $(V_s, V_t)$, its capacity is defined by

$$cap(A, B) = \sum_{(u,v)\in E, u\in V_s, v\in V_t} cap(u, v)$$

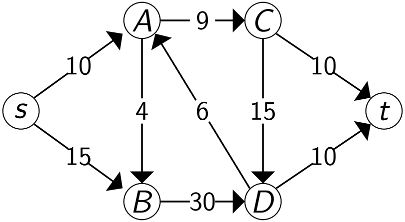



$\text{cap}(A, B) = 55$

# Cut

**Lemma**: For any $s - t$ cut $(V_s, V_t)$, and a flow $f$,

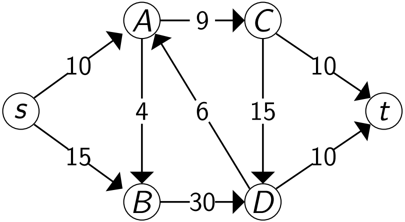
$$v(V_s) = \sum_{(u,v)\in E, u\in V_s, v\in V_t} f(u,v) - \sum_{(u,v)\in E, u\in V_t, v\in V_s} f(u,v) = v(f)$$



# Cut

**Lemma**: For any $s - t$ cut $(V_s, V_t)$, and a flow $f$,

$$v(V_s) = \sum_{(u,v)\in E, u\in V_s, v\in V_t} f(u,v) - \sum_{(u,v)\in E, u\in V_t, v\in V_s} f(u,v) = v(f)$$



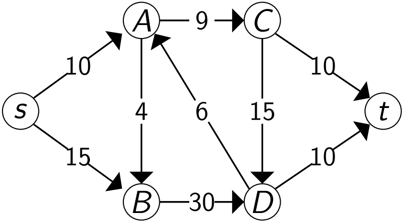

$$V_s = \{s\}$$

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

# Cut

**Lemma**: For any $s - t$ cut $(V_s, V_t)$, and a flow $f$,

$$v(V_s) = \sum_{(u,v)\in E, u\in V_s, v\in V_t} f(u,v) - \sum_{(u,v)\in E, u\in V_t, v\in V_s} f(u,v) = v(f)$$




add $w$ to $V_s$

$- \sum_{(u,w)\in E} f(u,w)$

$+ \sum_{(w,v)\in E} f(w,v)$

# Cut

$$v(V_s) = \sum_{(u,v)\in E, u\in V_s, v\notin V_s} f(u,v) - \sum_{(v,u)\in E, u\in V_s, v\notin V_s} f(u,v)$$

$$\begin{aligned}
v(V_s \cup \{w\}) &= v(V_s) \\
&\quad + \sum_{(w,v)\in E, v\notin V_s} f(w,v) - \sum_{(u,w)\in E, u\notin V_s} f(u,w) \\
&\quad - \sum_{(u,w)\in E, u\in V_s} f(u,w) + \sum_{(w,v)\in E, v\in V_s} f(w,v) \\
&= v(V_s) + \sum_{(w,v)\in E} f(w,v) - \sum_{(u,w)\in E} f(u,w) \\
&= v(V_s).
\end{aligned}$$

# Augmenting Path

**Theorem**: the flow is maximum if there is no augmenting path in the residual graph.

# Augmenting Path

**Theorem**: the flow is maximum if there is no augmenting path in the residual graph.

**Proof**: Let $f$ be the flow, with respect to which the residual graph $G^f$ has no augmenting path. Assume there is $f'$ with $v(f') > v(f)$. Initialize $P = \emptyset$.

# Augmenting Path

**Theorem**: the flow is maximum if there is no augmenting path in the residual graph.

**Proof**: Let $f$ be the flow, with respect to which the residual graph $G^f$ has no augmenting path. Assume there is $f'$ with $v(f') > v(f)$. Initialize $P = \emptyset$.

- for edge $(u, v) \in E$
  - if $f'(u, v) > f(u, v)$, then put $(u, v)$ to $P$
  - if $f'(u, v) < f(u, v)$, then put $(v, u)$ to $P$

# Augmenting Path

**Theorem**: the flow is maximum if there is no augmenting path in the residual graph.

**Proof**: Let $f$ be the flow, with respect to which the residual graph $G^f$ has no augmenting path. Assume there is $f'$ with $v(f') > v(f)$. Initialize $P = \emptyset$.

- for edge $(u, v) \in E$
  - if $f'(u, v) > f(u, v)$, then put $(u, v)$ to $P$
  - if $f'(u, v) < f(u, v)$, then put $(v, u)$ to $P$
- $s$ is connected to $t$ through $P$. Otherwise, let

$$P_s = \{u | s \textit{ is connected to } u, \textit{ through } P\}.$$

Then $v(f') = v_{f'}(P_s) \leq v_f(P_s) = v(f)$.

# Augmenting Path

**Theorem**: the flow is maximum if there is no augmenting path in the residual graph.

**Proof**: Let $f$ be the flow, with respect to which the residual graph $G^f$ has no augmenting path. Assume there is $f'$ with $v(f') > v(f)$. Initialize $P = \emptyset$.

- for edge $(u, v) \in E$
  - if $f'(u, v) > f(u, v)$, then put $(u, v)$ to $P$
  - if $f'(u, v) < f(u, v)$, then put $(v, u)$ to $P$
- $s$ is connected to $t$ through $P$. Otherwise, let

  $$P_s = \{u | s \textit{ is connected to } u, \textit{ through } P\}.$$

  Then $v(f') = v_{f'}(P_s) \leq v_f(P_s) = v(f)$.
- **Contradiction**!

# Augmenting Path

$v(f') = v_{f'}(P_s) \leq v_f(P_s) = v(f)$.

- $(u, v) \in E, u \in P_s, v \notin P_s$, then $f'(u, v) \leq f(u, v)$. Otherwise, $(u, v) \in P$
- $(v, u) \in E, u \in P_s, v \notin P_s$, then $f'(v, u) \geq f(v, u)$. Otherwise, $(u, v) \in P$

For any edge in $P$, it has remaining capacity, with respect to flow $f$: for any edge $(u, v) \in P$, one of the following holds

- $f(u, v) < f'(u, v) \leq c(u, v)$; or
- $f(v, u) > f'(v, u) \geq 0$.

# Duality between Flow and Cut

**Weak Duality**: given a flow $f$, for any $s - t$ cut $(A, B)$, it holds that

$$v(f) \leq cap(A, B)$$

**Problem**: given a flow network, find the cut of the minimum capacity.

**Claim**: maximum flow $\leq$ minimum cut

**Corollary**: for a flow $f$ and a cut $(A, B)$, if $v(f) = cap(A, B)$, then

- $f$ is a maximum flow
- $(A, B)$ is a minimum cut

# Max-Flow Min-Cut Theorem

**Theorem**: the value of the maximum flow is equal to the value of the minimum cut.

**Proof**:

# Max-Flow Min-Cut Theorem

**Theorem**: the value of the maximum flow is equal to the value of the minimum cut.

**Proof**:

- maximum flow $\leq$ minimum-cut. Trivial, since

$$v(f) \leq cap(A, B)$$

# Max-Flow Min-Cut Theorem

**Theorem**: the value of the maximum flow is equal to the value of the minimum cut.

**Proof**:

- maximum flow $\leq$ minimum-cut. Trivial, since

$$v(f) \leq cap(A, B)$$

- minimum-cut $\leq$ maximum flow. Consider a maximum flow $f$. Let $P_s = \{$ nodes connected from $s$ in $G^f$ $\}$. For $u \in P_s$ and $v \notin P_s$:

# Max-Flow Min-Cut Theorem

**Theorem**: the value of the maximum flow is equal to the value of the minimum cut.

**Proof**:

- maximum flow $\leq$ minimum-cut. Trivial, since

$$v(f) \leq cap(A, B)$$

- minimum-cut $\leq$ maximum flow. Consider a maximum flow $f$. Let $P_s = \{$ nodes connected from $s$ in $G^f$ $\}$. For $u \in P_s$ and $v \notin P_s$:
  - if $(u, v) \in E$, then $f(u, v) = cap(u, v)$. Otherwise, $(u, v)$ is in $G^f$, and thus $v \in P_s$. Contradiction!

# Max-Flow Min-Cut Theorem

**Theorem**: the value of the maximum flow is equal to the value of the minimum cut.

**Proof**:

- maximum flow $\leq$ minimum-cut. Trivial, since

$$v(f) \leq cap(A, B)$$

- minimum-cut $\leq$ maximum flow. Consider a maximum flow $f$. Let $P_s = \{$ nodes connected from $s$ in $G^f$ $\}$. For $u \in P_s$ and $v \notin P_s$:
  - if $(u, v) \in E$, then $f(u, v) = cap(u, v)$. Otherwise, $(u, v)$ is in $G^f$, and thus $v \in P_s$. Contradiction!
  - if $(v, u) \in E$, then $f(v, u) = 0$. Otherwise, $(u, v)$ is in $G^f$, and thus $v \in P_s$. Contradiction!

# Max-Flow Min-Cut Theorem

**Theorem**: the value of the maximum flow is equal to the value of the minimum cut.

**Proof**:

- maximum flow $\leq$ minimum-cut. Trivial, since

$$v(f) \leq cap(A, B)$$

- minimum-cut $\leq$ maximum flow. Consider a maximum flow $f$. Let $P_s = \{$ nodes connected from $s$ in $G^f$ $\}$. For $u \in P_s$ and $v \notin P_s$:
  - if $(u, v) \in E$, then $f(u, v) = cap(u, v)$. Otherwise, $(u, v)$ is in $G^f$, and thus $v \in P_s$. Contradiction!
  - if $(v, u) \in E$, then $f(v, u) = 0$. Otherwise, $(u, v)$ is in $G^f$, and thus $v \in P_s$. Contradiction!

  Thus $v(f) = v_f(P_s) = cap(P_s, V \setminus P_s)$.

# The Ford-Fulkerson Algorithm

To find the value of the minimum cut, run Ford-Fulkerson to find the value of the maximum flow.

**Complexity**: for a network of maximum capacity $C$

- input size: $\log C$
- number of iterations: $C$ in the worst case

# The Edmonds-Karp Algorithm

**Ford-Fulkerson**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$.
2. Update $G^f$ with respect to $f$.
3. In $G^f$, find a path from $s$ to $t$.
4. Augment the flow along the path.
5. Repeat step 2 and 4, until no such path can be found.

**Edmonds-Karp**

1. Initially, $f(u, v) = 0$ for all $(u, v) \in E$.
2. Update $G^f$ with respect to $f$.
3. In $G^f$, find the shortest (hop-distance) path from $s$ to $t$.
4. Augment the flow along the path.
5. Repeat step 2 and 4, until no such path can be found.

# The Edmonds-Karp Algorithm

Define $\delta_f(u, v)$ as the hop-distance from $u$ to $v$, in $G^f$.

**Lemma**: consider a flow $f$, let $f'$ be the result after the flow augmentation. Then $\delta_{f'}(s, v) \geq \delta_f(s, v)$ for all $v \in V \setminus \{s, t\}$.

**Proof**: Assume that

$$\{v \in V \setminus \{s, t\} | \delta_{f'}(s, v) < \delta_f(s, v)\} \neq \emptyset.$$

Select $v^*$ from the set with minimum $\delta_{f'}(s, v)$. Let $u$ be the previous node in shortest (hop-distance) path from $s$ to $v^*$, in $G^{f'}$. Then

$$\delta_f(s, u) \leq \delta_{f'}(s, u) = \delta_{f'}(s, v^*) - 1 < \delta_f(s, v^*) - 1$$

$$\begin{tikzcd} s \arrow[rr, dotted] & & u \arrow[r] & v^* \end{tikzcd} \qquad \begin{tikzcd} s \arrow[rr, dotted] & & v^* \arrow[r] & u \end{tikzcd}$$

$G^{f'}$ $\qquad$ $G^f$

# The Edmonds-Karp Algorithm

$$s \dashrightarrow v^* \rightarrow u$$

"$f(v^*, u)$ is changed" implies $v^*$ is on the shortest path from $s$ to $u$ in $G^f$, thus

$$\delta_f(s, v^*) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v^*) - 2.$$

**Contradiction** to $\delta_{f'}(s, v^*) < \delta_f(s, v^*)$.

# The Edmonds-Karp Algorithm: $O(|V||E|)$

**Theorem**: The number of flow augmentation performed by Edmonds-Karp is $O(|V||E|)$.
**Proof**: In the augmenting path, $(u, v)$ is **critical** if it has the minimum remaining capacity among the edges in the path.

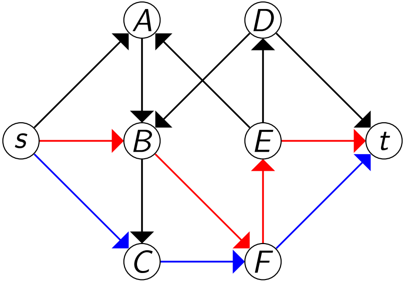
For each edge $(u, v) \in E$

- first time being a critical edge, $f(u, v)$ is increased to $c(u, v)$.
$$\delta_f(s, v) = \delta_f(s, u) + 1$$
- to appear again in the residual graph, $(v, u)$ should be involved in some augmenting path
$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$$
- thus
$$\delta_{f'}(s, u) \geq \delta_f(s, v) + 1 = \delta_f(s, u) + 2$$
- $(u, v)$ becomes critical for at most $|V|/2 + 1$ times.

# Edge Disjoint Paths

# Edge Disjoint Paths

**Edge-disjoint paths**: two paths are edge-disjoint if they have no edges in common.




**Problem**: given a directed graph $G = (V, E)$, source node $s \in V$ and target node $t \in V$, how many edge-disjoint $s - t$ paths are there?

# Edge Disjoint Paths

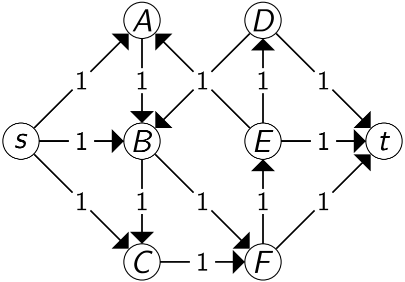
**Problem**: given a directed graph $G = (V, E)$, source node $s \in V$ and target node $t \in V$, how many edge-disjoint $s - t$ paths are there?

**Network Connectivity**: given a directed graph $G = (V, E)$, source node $s$ and target node $t$, find minimum number of edges whose removal disconnects $t$ from $s$.

[Menger 1927] **Theorem**: $\#$ edge-disjoint $s - t$ paths $=$ network connectivity

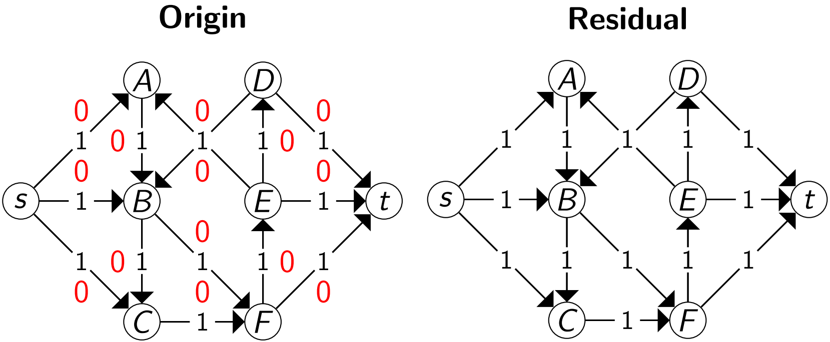
# Edge Disjoint Paths

$w(u, v) = 1$ for all $(u, v) \in E$

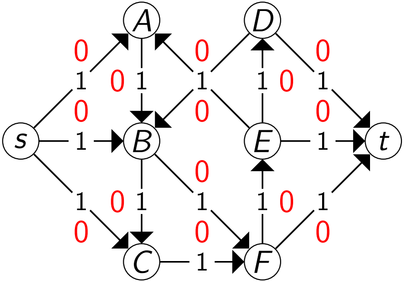



**Reduction**: finding the maximum flow from $s$ to $t$.

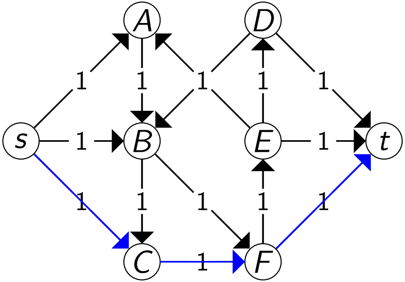
# Example



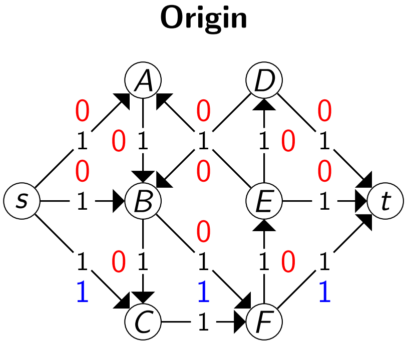
**Origin**

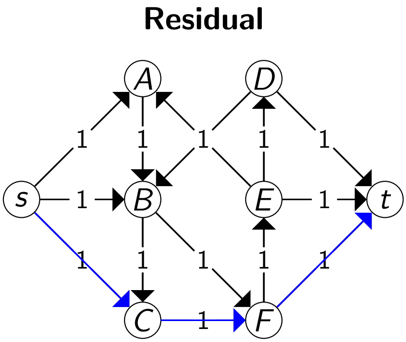



**Residual**



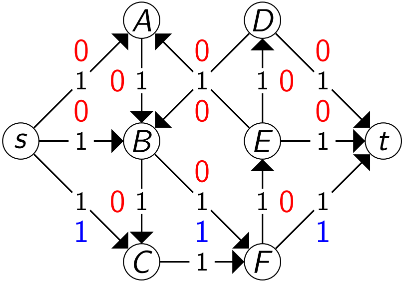
# Example



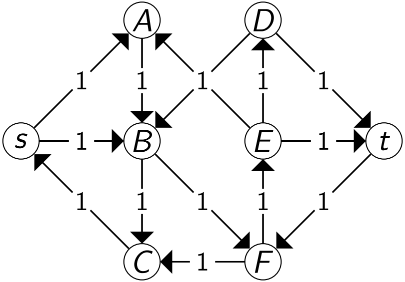



# Example

**Origin**

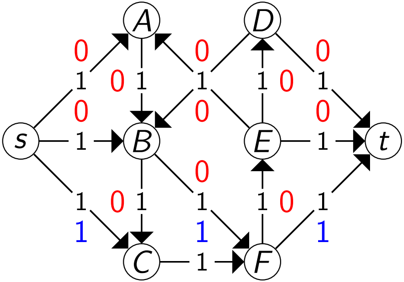



**Residual**

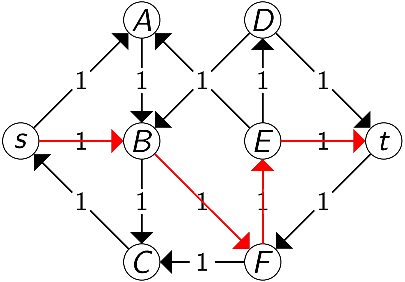


# Example

**Origin**

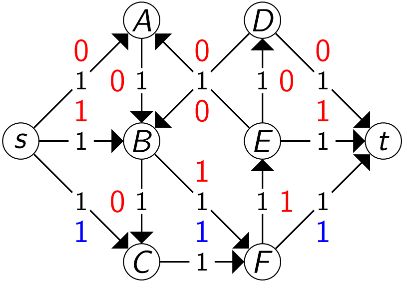



**Residual**

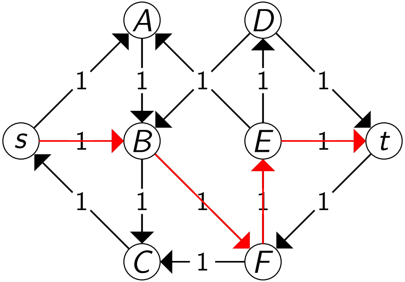


# Example

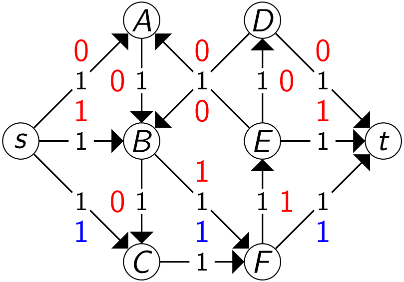
**Origin**



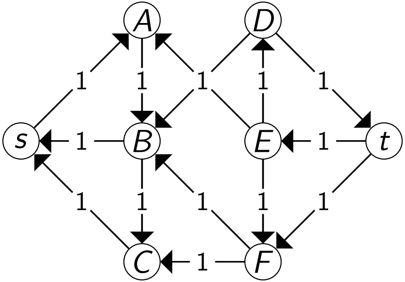

**Residual**



**Origin**

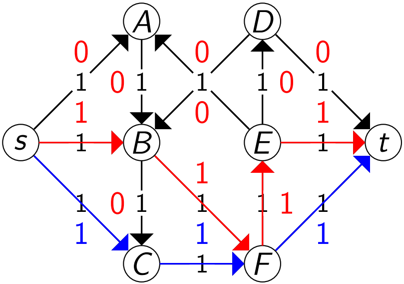



**Residual**

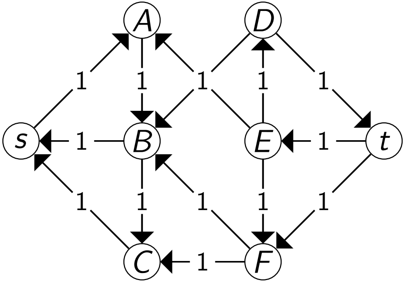


# Example

**Origin**




**Residual**



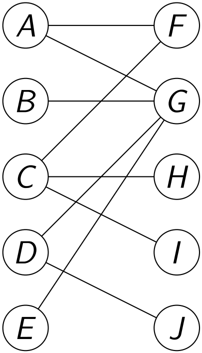
# Bipartite Matching

# Bipartite Graph

Undirected graph $G = (V, E)$ is **bipartite**, if

- $V$ is divided into $L$ and $R$
- $u \in L$ and $v \in R$, for all $(u, v) \in E$

**Matching**: $M \subseteq E$, for each $v \in V$, it is incident in at most one edge of $M$.

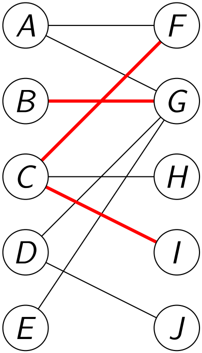


# Bipartite Graph

Undirected graph $G = (V, E)$ is **bipartite**, if

- $V$ is divided into $L$ and $R$
- $u \in L$ and $v \in R$, for all $(u, v) \in E$

**Matching**: $M \subseteq E$, for each $v \in V$, it is incident in at most one edge of $M$.

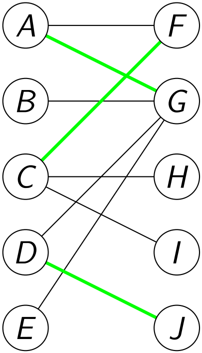


# Bipartite Graph

Undirected graph $G = (V, E)$ is **bipartite**, if

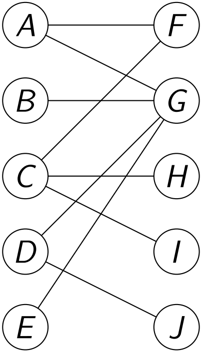
- $V$ is divided into $L$ and $R$
- $u \in L$ and $v \in R$, for all $(u, v) \in E$

**Matching**: $M \subseteq E$, for each $v \in V$, it is incident in at most one edge of $M$.



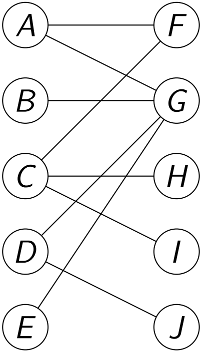
# Maximum Bipartite Graph

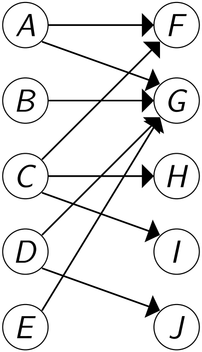
**Problem**: given a bipartite graph, find the matching of the maximum size.



# Maximum Bipartite Graph

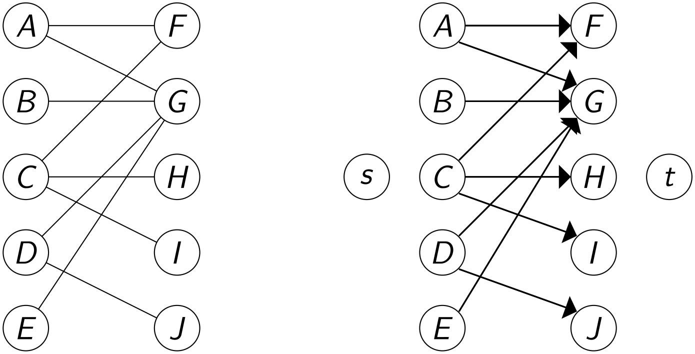
**Problem**: given a bipartite graph, find the matching of the maximum size.



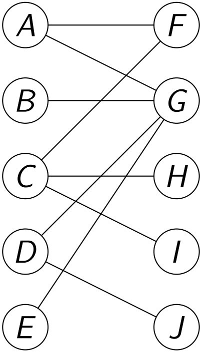
# Maximum Bipartite Graph

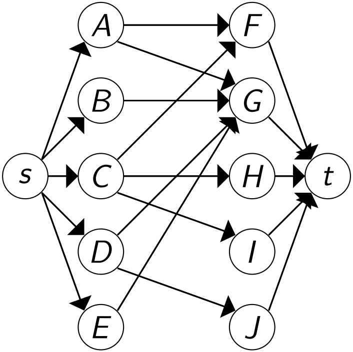
**Problem**: given a bipartite graph, find the matching of the maximum size.



# Maximum Bipartite Graph

**Problem**: given a bipartite graph, find the matching of the maximum size.



# THANK YOU