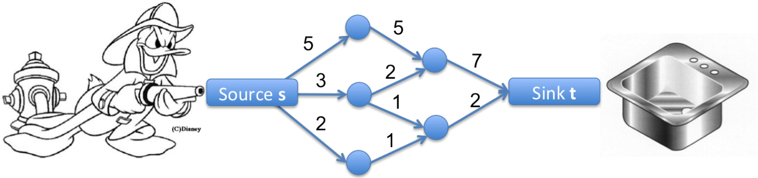
Algorithms: Ford-Fulkerson Method

Alessandro Chiesa, Ola Svensson

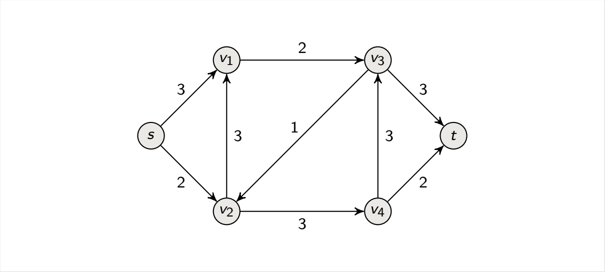
EPFL School of Computer and Communication Sciences

Lecture 18, 16.04.2025



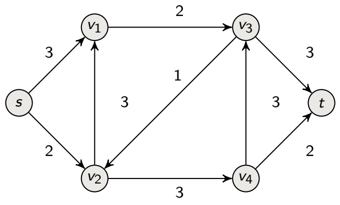
FLOW NETWORKS

Flow Network



- Directed graph $G = (V, E)$
- Each edge (u, v) has a capacity $c(u, v) \geq 0$ ($c(u, v) = 0$ if $(u, v) \notin E$)
- Source s and sink t (flow goes from s to t)
- No antiparallel edges (assumed w.l.o.g. for simplicity)

Definition of a flow



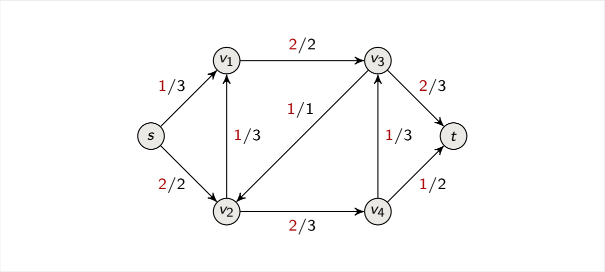
A flow is a function $f : V \times V \to \mathbb{R}$ satisfying:

Capacity constraint: For all $u, v \in V : 0 \leq f(u, v) \leq c(u, v)$

Flow conservation: For all $u \in V \setminus \{s, t\}$,

$$\underbrace{\sum_{v \in V} f(v, u)}_{\text{flow into } u} = \underbrace{\sum_{v \in V} f(u, v)}_{\text{flow out of } u}$$

Definition of a flow



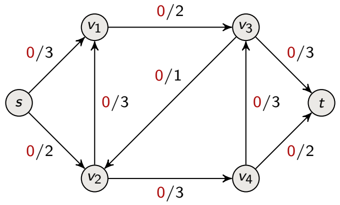
A flow is a function $f : V \times V \to \mathbb{R}$ satisfying:

Capacity constraint: For all $u, v \in V : 0 \leq f(u,v) \leq c(u,v)$

Flow conservation: For all $u \in V \setminus \{s,t\}$,

$$\underbrace{\sum_{v \in V} f(v,u)}_{\text{flow into } u} = \underbrace{\sum_{v \in V} f(u,v)}_{\text{flow out of } u}$$

Definition of a flow



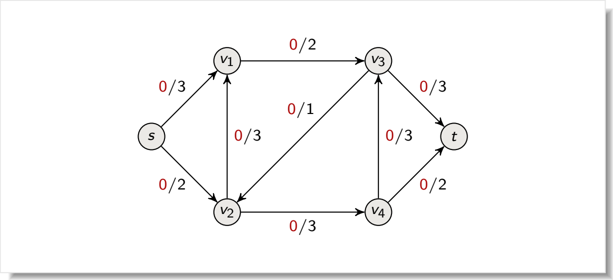
A flow is a function $f : V \times V \to \mathbb{R}$ satisfying:

Capacity constraint: For all $u, v \in V : 0 \leq f(u, v) \leq c(u, v)$

Flow conservation: For all $u \in V \setminus \{s, t\}$,

$$\underbrace{\sum_{v \in V} f(v, u)}_{\text{flow into } u} = \underbrace{\sum_{v \in V} f(u, v)}_{\text{flow out of } u}$$

Value of a flow

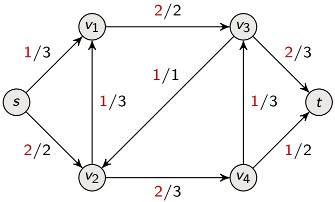


Value of a flow $f = |f|$

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

$=$ flow out of source $-$ flow into source

Value of a flow

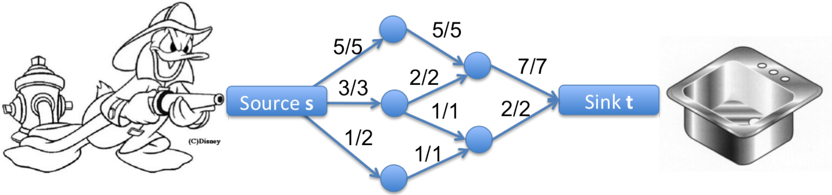


Value of a flow $f = |f|$

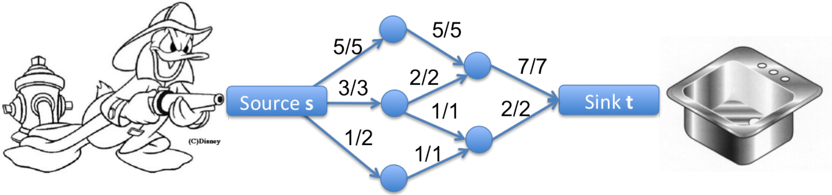
$$= \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

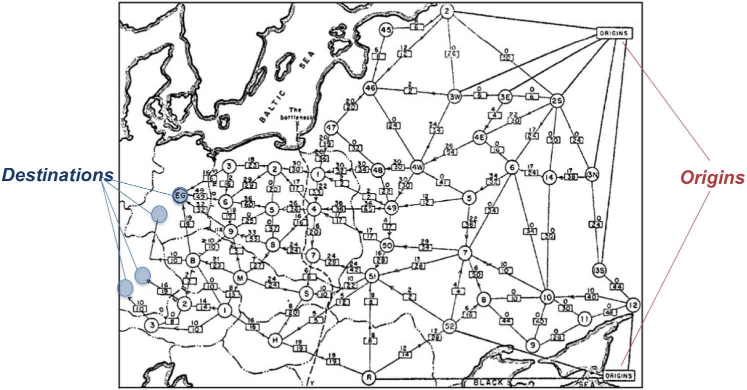
$=$ flow out of source $-$ flow into source

What's the value of this flow?

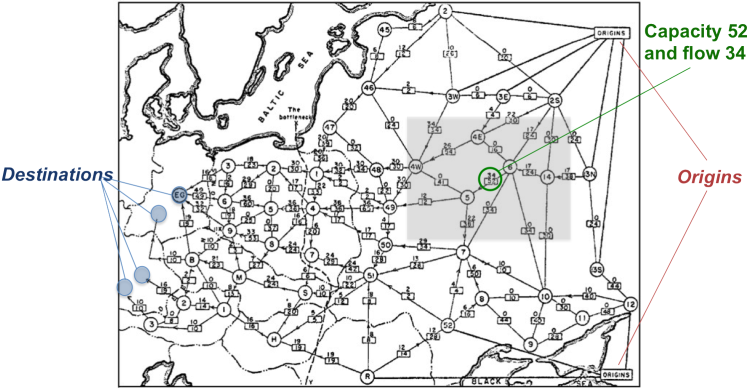


What's the value of this flow? 9



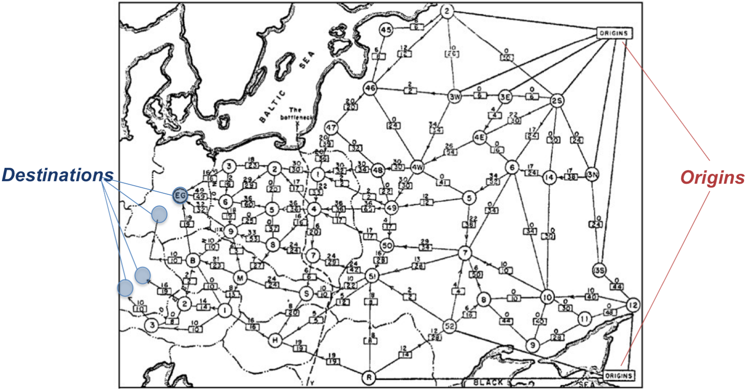


- Schematic diagram of the railway network of the western Soviet union and easter European countries, from Harris & Ross (1955), declassified by pentagon in 1999.



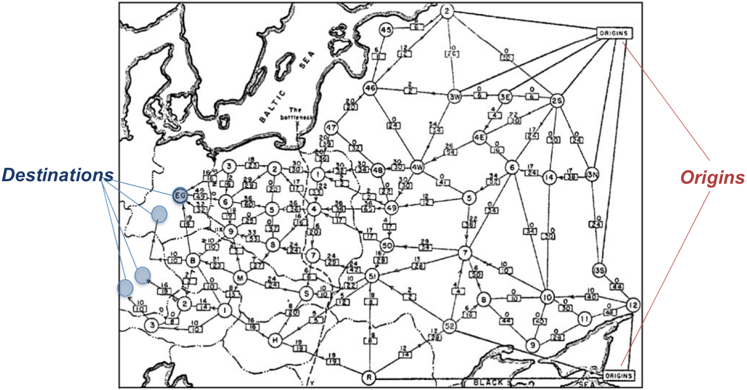
- Schematic diagram of the railway network of the western Soviet union and easter European countries, from Harris & Ross (1955), declassified by pentagon in 1999.

Goal of Soviet union



Goal of Soviet union

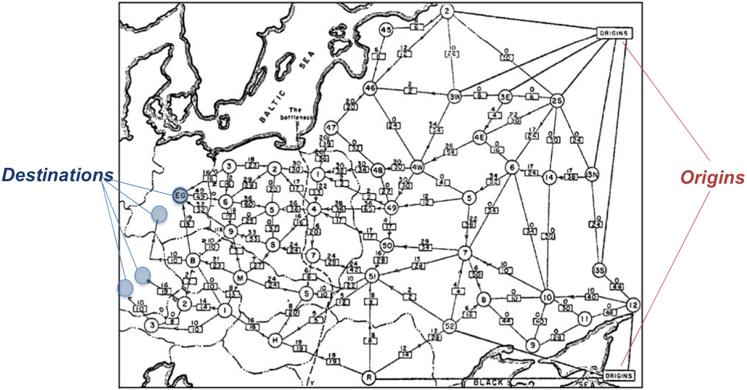
Maximize throughput from the “origins” to the destinations



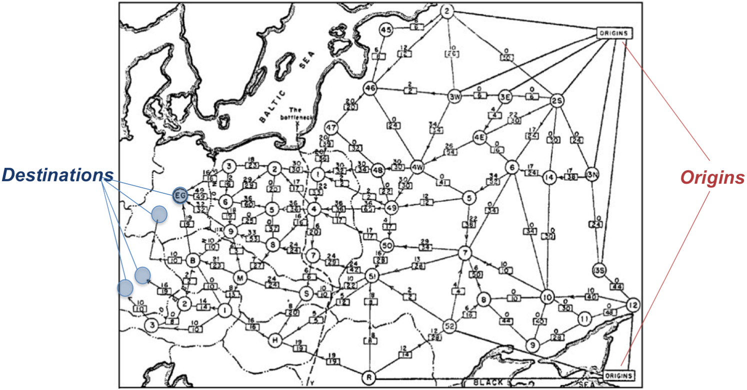
Goal of Soviet union

Maximize throughput from the "origins" to the destinations

Ford-Fulkerson method solves it

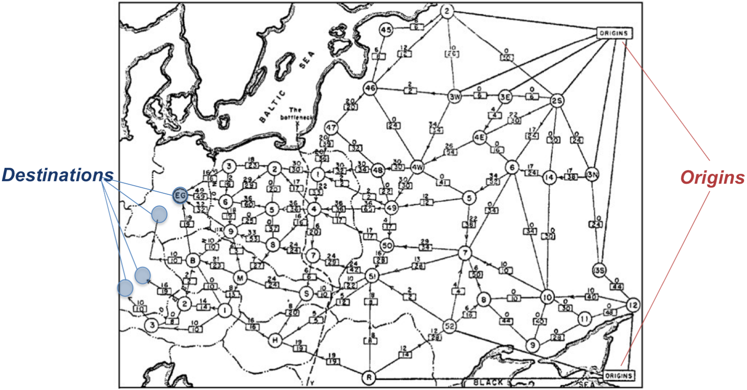


Goal of US Air Force (1950's)



Goal of US Air Force (1950's)

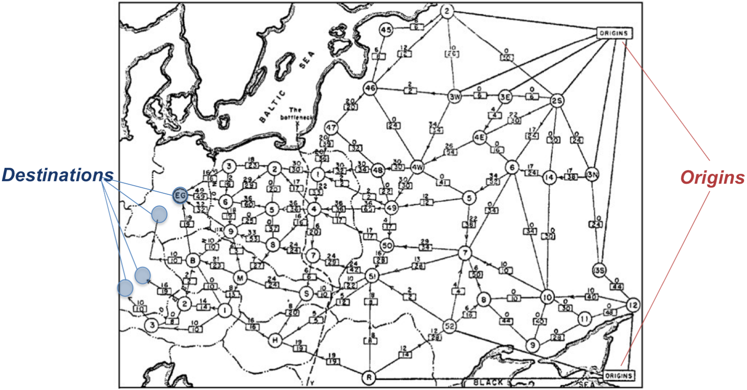
Disrupt flow of goods into satellite countries in the best possible way



Goal of US Air Force (1950's)

Disrupt flow of goods into satellite countries in the best possible way

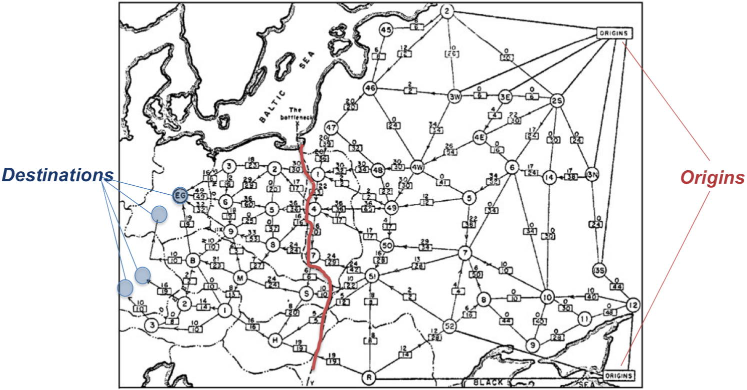
Find a minimum cut (Ford-Fulkerson method solves it)



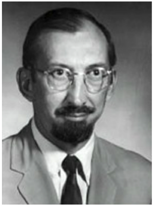
Goal of US Air Force (1950's)

Disrupt flow of goods into satellite countries in the best possible way

Find a minimum cut (Ford-Fulkerson method solves it)



L. R. Ford, Jr. (1927-)



D, R, Fulkerson (1924-1976)

MAXIMUM-FLOW PROBLEM

Ford-Fulkerson Method

The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. **return** f

The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an **augmenting path** p in the **residual network** G_f
3. **augment flow** f along p
4. **return** f

The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an **augmenting path** p in the **residual network** G_f
3. **augment flow** f along p
4. **return** f

Basic idea:

The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an **augmenting path** p in the **residual network** G_f
3.   **augment flow** f along p
4. **return** f

Basic idea:

- As long as there is a path from source to sink, with available capacity on all edges in the path

The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an **augmenting path** p in the **residual network** G_f
3. **augment flow** f along p
4. **return** f

Basic idea:

- As long as there is a path from source to sink, with available capacity on all edges in the path
- send flow along one of these paths and then we find another path and so on

Residual network

- Given a flow f and a network $G = (V, E)$
- the residual network consists of edges with capacities that represent how we can change the flow on the edges

Residual network

- Given a flow f and a network $G = (V, E)$
- the residual network consists of edges with capacities that represent how we can change the flow on the edges

Residual capacity:

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

Residual network

- Given a flow f and a network $G = (V, E)$
- the residual network consists of edges with capacities that represent how we can change the flow on the edges

Residual capacity:

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

Amount of capacity left (for the case $(u,v) \in E$)

Amount of flow that can be reversed (for the case $(v,u) \in E$)

Residual network

- Given a flow f and a network $G = (V, E)$
- the residual network consists of edges with capacities that represent how we can change the flow on the edges

Residual capacity:

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

Amount of capacity left (first case)

Amount of flow that can be reversed (second case)

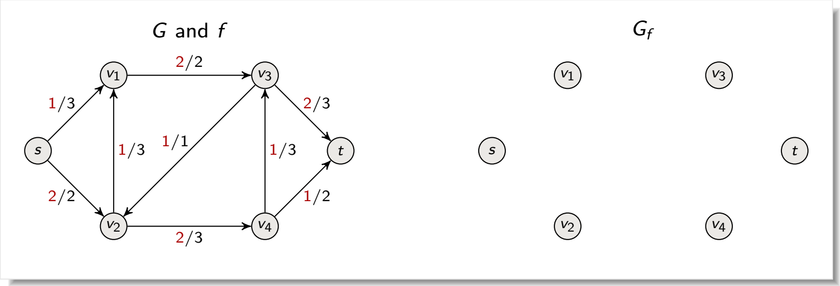
Residual network:

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

Examples

Residual network: $G_f = (V, E_f)$ where $E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$ and

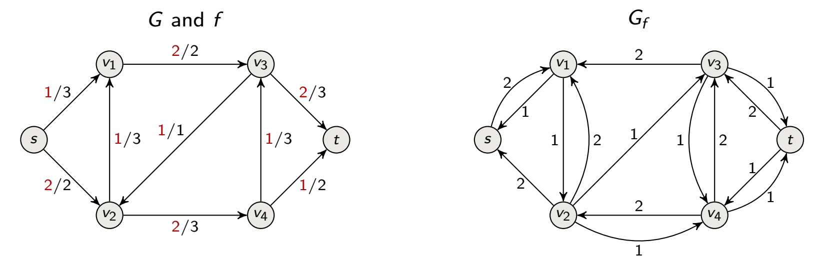
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$



Examples

Residual network: $G_f = (V, E_f)$ where $E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$ and

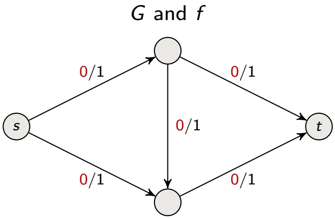
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$



The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. **Initialize flow f to 0**
2. **while** exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. **return** f

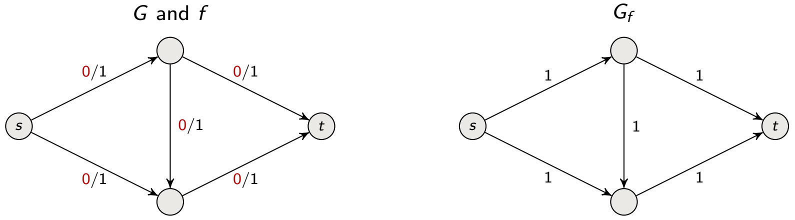


The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f

Augmenting path = simple path from s to t

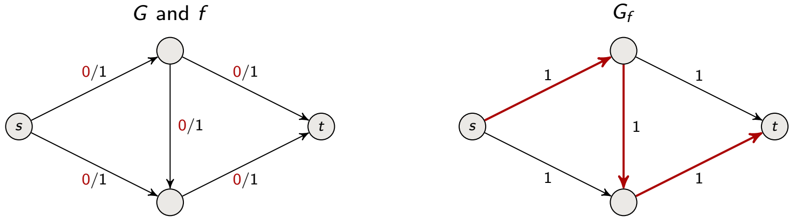


The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f

Exists augmenting path **p**

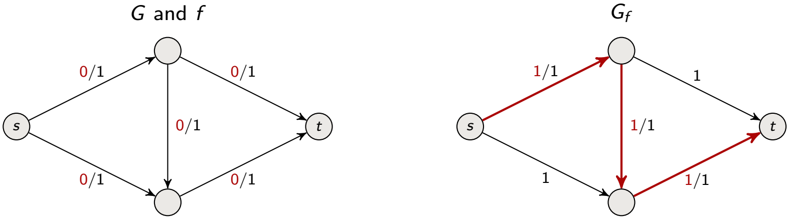


The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f

Exists augmenting path **p** with flow $\mathbf{f_p}$ of value = min capacity on **p**

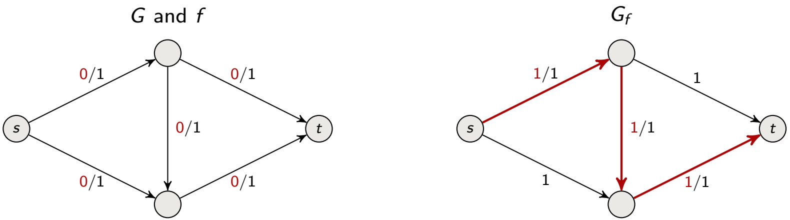


The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an augmenting path p in the residual network G_f
3. **augment flow f along p**
4. **return** f

$\mathbf{f}$ is updated by changing the flow on an edge $(\mathbf{u}, \mathbf{v})$ by $\mathbf{f_p(u, v) - f_p(v, u)}$

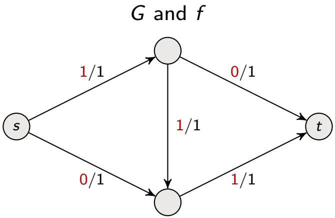


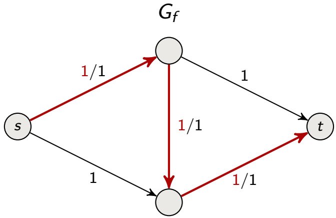
The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an augmenting path p in the residual network G_f
3. **augment flow f along p**
4. **return** f

$\mathbf{f}$ is updated by changing the flow on an edge $(\mathbf{u}, \mathbf{v})$ by $\mathbf{f_p}(\mathbf{u}, \mathbf{v}) - \mathbf{f_p}(\mathbf{v}, \mathbf{u})$

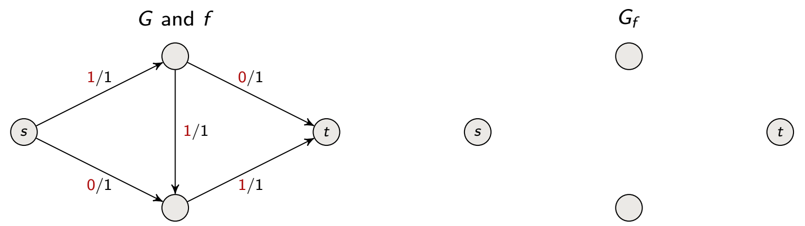




The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

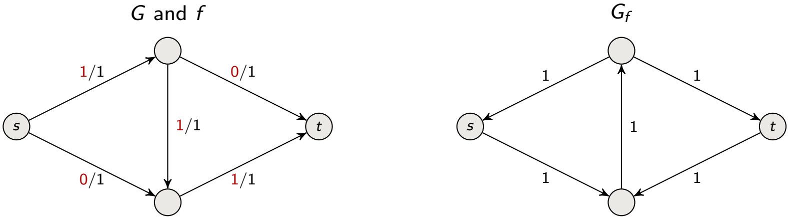
1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f



The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

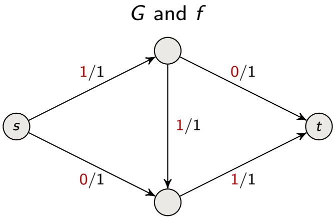
1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f

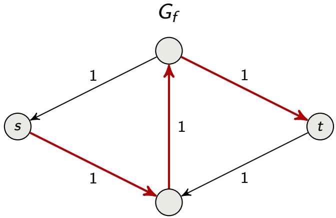


The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f

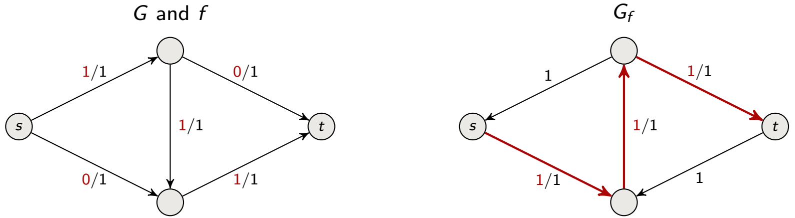




The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

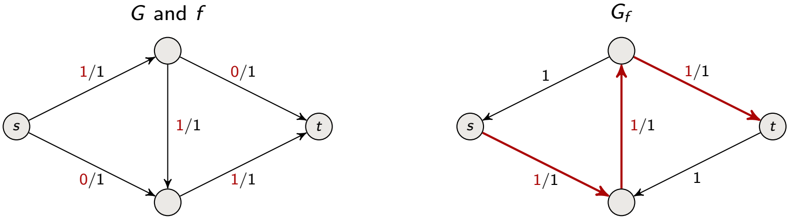
1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f



The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

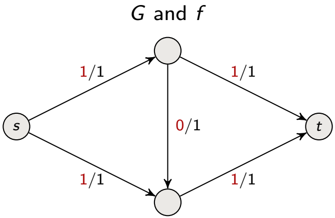
1. Initialize flow f to 0
2. **while** exists an augmenting path p in the residual network G_f
3. **augment flow f along p**
4. **return** f

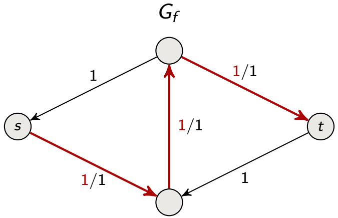


The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an augmenting path p in the residual network G_f
3. **augment flow f along p**
4. **return** f

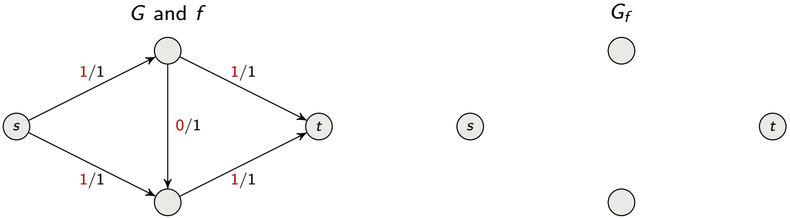




The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

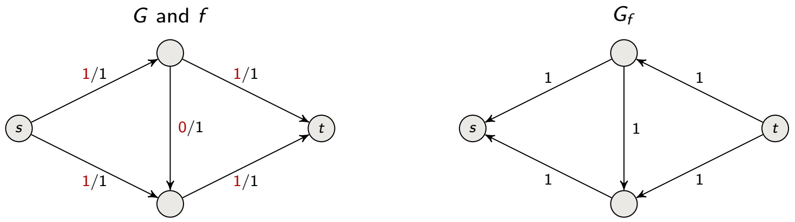
1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f



The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f



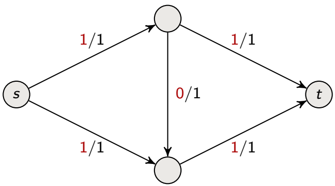
The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

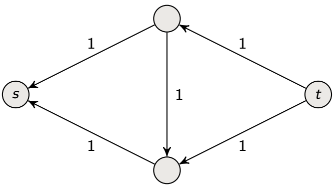
1. Initialize flow f to 0
2. **while exists an augmenting path p in the residual network G_f**
3. augment flow f along p
4. **return** f

No augmenting path and flow of value 2 is optimal

G and f



G_f

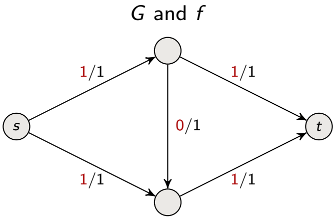


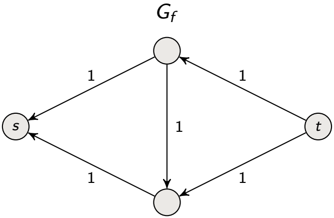
The Ford-Fulkerson Method'54

FORD-FULKERSON-METHOD(G, s, t):

1. Initialize flow f to 0
2. **while** exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. **return** f

No augmenting path and flow of value 2 is optimal





The Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck$=$ min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

The Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck= min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

Min-cut

If no augmenting path exists in residual network, then

- Find set of nodes S reachable from s in residual network
- Set $T = V \setminus S$

S and T define a minimum cut

The Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck= min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

Min-cut

If no augmenting path exists in residual network, then

- Find set of nodes S reachable from s in residual network
- Set $T = V \setminus S$

S and T define a minimum cut

$$\text{Max-flow} = \text{Min-cut}$$

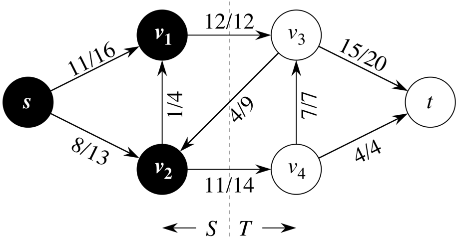
Gives a way to verify that the step-by-step calculations of the flow are correct!

WHY IS RETURNED FLOW OPTIMAL? (MIN-CUTS)

Cuts in flow networks

A cut of flow network $G(V, E)$ is

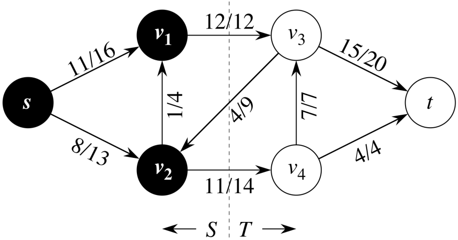
- a partition of V into S and $T = V \setminus S$
- such that $s \in S$ and $t \in T$



Net flow across a cut

The net flow across cut (S, T) is

$$f(S,T) = \underbrace{\sum_{u \in S, v \in T} f(u,v)}_{\text{flow leaving } S} - \underbrace{\sum_{u \in S, v \in T} f(v,u)}_{\text{flow entering } S}$$

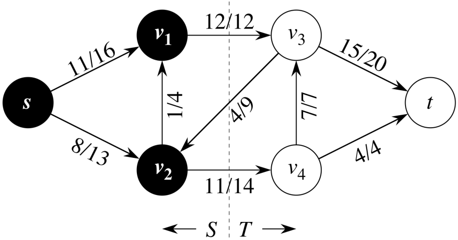


Net flow across a cut

The net flow across cut (S, T) is

$$f(S,T) = \underbrace{\sum_{u \in S, v \in T} f(u,v)}_{\text{flow leaving } S} - \underbrace{\sum_{u \in S, v \in T} f(v,u)}_{\text{flow entering } S}$$

What is the net flow of this cut?

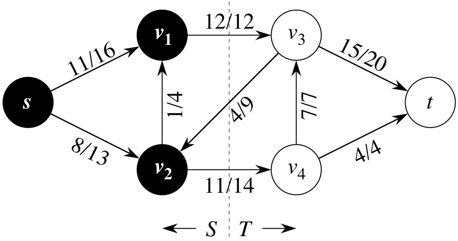


Net flow across a cut

The net flow across cut (S, T) is

$$f(S,T) = \underbrace{\sum_{u \in S, v \in T} f(u,v)}_{\text{flow leaving } S} - \underbrace{\sum_{u \in S, v \in T} f(v,u)}_{\text{flow entering } S}$$

What is the net flow of this cut? $12 + 11 - 4 = 19$

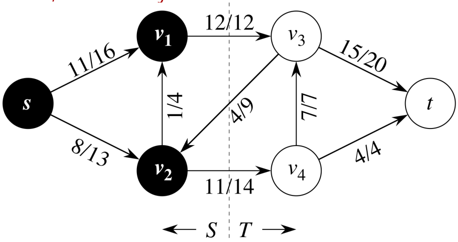


Net flow across a cut

The net flow across cut (S, T) is

$$f(S,T) = \underbrace{\sum_{u \in S, v \in T} f(u,v)}_{\text{flow leaving } S} - \underbrace{\sum_{u \in S, v \in T} f(v,u)}_{\text{flow entering } S}$$

What is the net flow of this cut? $12 + 11 - 4 = 19$ Note that this equals the value of the flow; it's always the case!



Net flow equals flow value for any cut

Theorem

For any cut (S, T), $|f| = f(S, T)$.

Net flow equals flow value for any cut

Theorem

For any cut (S, T), $|f| = f(S, T)$.

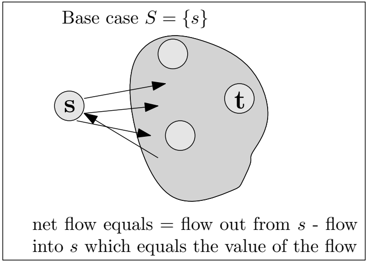
Proof by induction on the size of S.

Net flow equals flow value for any cut

Theorem

For any cut (S, T), $|f| = f(S, T)$.

Proof by induction on the size of S.

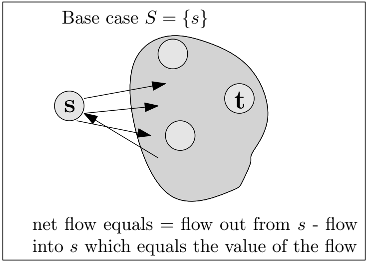


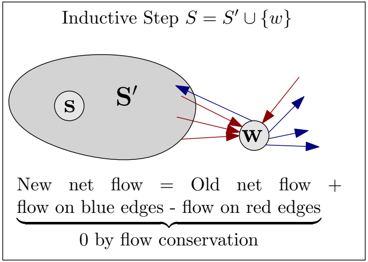
Net flow equals flow value for any cut

Theorem

For any cut (S, T), $|f| = f(S, T)$.

Proof by induction on the size of S.

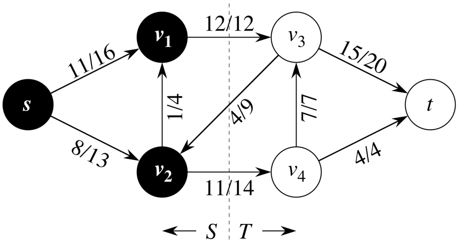




Capacity a cut

The capacity of a cut (S, T) is

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

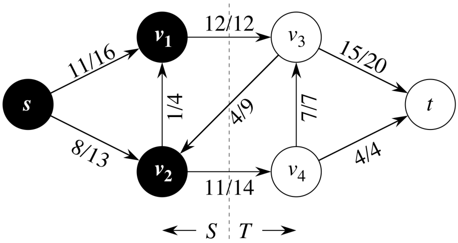


Capacity a cut

The capacity of a cut (S, T) is

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

What is the capacity of this cut?

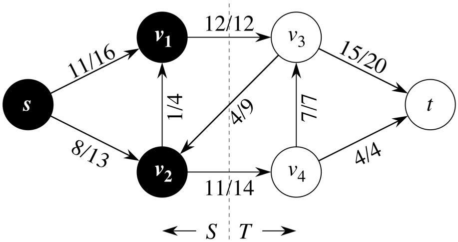


Capacity a cut

The capacity of a cut (S, T) is

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

What is the capacity of this cut? $12 + 14 = 26$



Flow is at most capacity of a cut

For any flow f and any cut (S, T):

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

Flow is at most capacity of a cut

For any flow f and any cut (S, T):

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

Flow is at most capacity of a cut

For any flow f and any cut (S, T):

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

Flow is at most capacity of a cut

For any flow f and any cut (S, T):

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

Flow is at most capacity of a cut

For any flow f and any cut (S, T):

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

Flow is at most capacity of a cut

For any flow f and any cut (S, T):

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

Max-flow is at most capacity of a cut

Max-flow is at most capacity of a cut

Therefore: **max-flow** $\leq$ **min-cut**

Max-flow is at most capacity of a cut

Therefore: **max-flow $\leq$ min-cut**

We shall prove

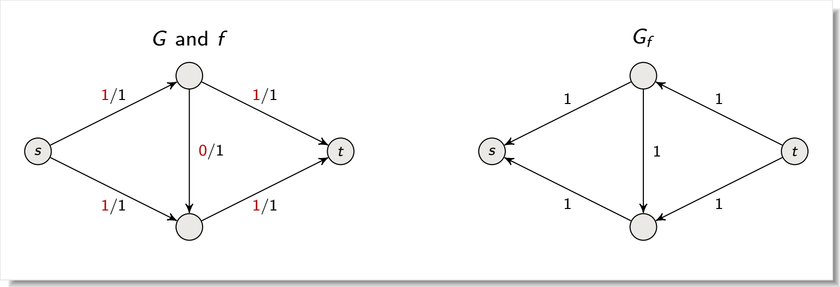
Theorem (max-flow min-cut theorem)

max-flow = min-cut

Examples

Consider f obtained by running Ford-Fulkerson and let

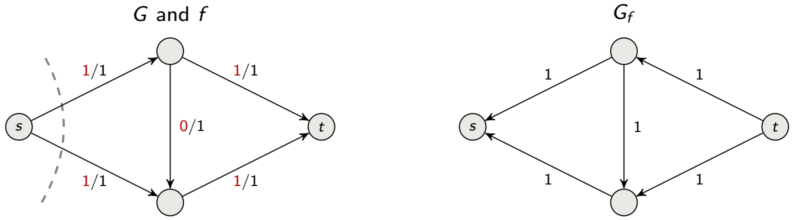
$S = \{v \in V : \text{there is a path from } s \text{ to } v \text{ in } G_f\}$ and $T = V \setminus S$



Examples

Consider f obtained by running Ford-Fulkerson and let

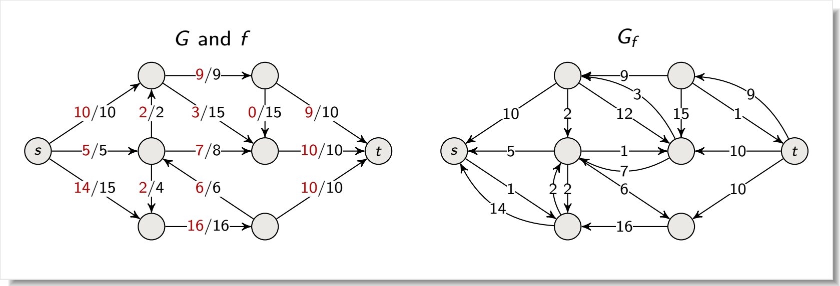
$S = \{v \in V : \text{there is a path from } s \text{ to } v \text{ in } G_f\}$ and $T = V \setminus S$



Examples

Consider f obtained by running Ford-Fulkerson and let

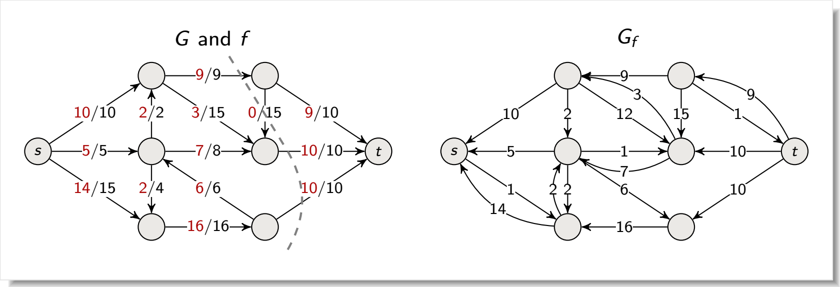
$S = \{v \in V : \text{there is a path from } s \text{ to } v \text{ in } G_f\}$ and $T = V \setminus S$



Examples

Consider f obtained by running Ford-Fulkerson and let

$S = \{v \in V : \text{there is a path from } s \text{ to } v \text{ in } G_f\}$ and $T = V \setminus S$



Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof.

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (1) $\Rightarrow$ (2): Suppose toward contradiction that G_f has an augmenting path p.

However, then Ford-Fulkerson method would augment f by p to obtain a flow if increased value which contradicts that f is a maximum flow

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (2) $\Rightarrow$ (3): $S =$ set of nodes reachable from s in residual network, $T = V \setminus S$

Max-flow min-cut theorem

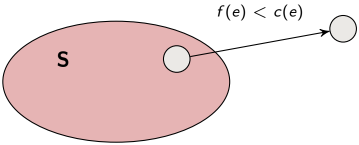
Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

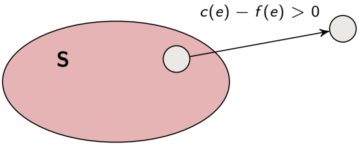
1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (2) $\Rightarrow$ (3): $S =$ set of nodes reachable from s in residual network, $T = V \setminus S$

Every edge flowing out of S in G must be at capacity, otherwise we can reach a node outside S in the residual network.



Original graph



Residual network

Max-flow min-cut theorem

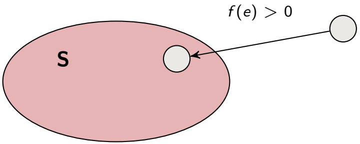
Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

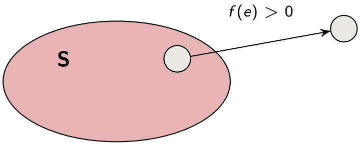
1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (2) $\Rightarrow$ (3): $S =$ set of nodes reachable from s in residual network, $T = V \setminus S$

Every edge flowing into S in G must have flow 0, otherwise we can reach a node outside S in the residual network.



Original graph



Residual network

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (2) $\Rightarrow$ (3): $S =$ set of nodes reachable from s in residual network, $T = V \setminus S$

Therefore

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

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (2) $\Rightarrow$ (3): $S =$ set of nodes reachable from s in residual network, $T = V \setminus S$

Therefore

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

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (2) $\Rightarrow$ (3): $S =$ set of nodes reachable from s in residual network, $T = V \setminus S$

Therefore

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

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (2) $\Rightarrow$ (3): $S =$ set of nodes reachable from s in residual network, $T = V \setminus S$

Therefore

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

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (3) $\Rightarrow$ (1): Recall that $|f| \leq c(S, T)$ for all cuts (S, T).

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (3) $\Rightarrow$ (1): Recall that $|f| \leq c(S, T)$ for all cuts (S, T).

Therefore, if the value of flow is equal to the capacity of some cut, then it cannot be further improved.

Max-flow min-cut theorem

Let $G = (V, E)$ be a flow network with source s and sink t and capacities c and a flow f.

The following are equivalent:

1. f is a maximum flow
2. G_f has no augmenting path
3. $|f| = c(S, T)$ for a minimum cut (S, T)

Proof. (3) $\Rightarrow$ (1): Recall that $|f| \leq c(S, T)$ for all cuts (S, T).

Therefore, if the value of flow is equal to the capacity of some cut, then it cannot be further improved.

So f is a maximum flow

Summary: Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck= min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

Summary: Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck= min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

Min-cut

If no augmenting path exists in residual network, then

- Find set of nodes S reachable from s in residual network
- Set $T = V \setminus S$

S and T define a minimum cut

Summary: Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck= min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

Min-cut

If no augmenting path exists in residual network, then

- Find set of nodes S reachable from s in residual network
- Set $T = V \setminus S$

S and T define a minimum cut

Max-flow = Min-cut

Summary: Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck= min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

Min-cut

If no augmenting path exists in residual network, then

- Find set of nodes S reachable from s in residual network
- Set $T = V \setminus S$

S and T define a minimum cut

$$\text{Max-flow} = \text{Min-cut}$$

Gives a way to verify that the step-by-step calculations of the flow are correct!

Summary: Ford-Fulkerson Method

Max-flow

Start with 0-flow
while there is an augmenting path from s to t in residual network **do**

- Find augmenting path
- Compute bottleneck= min capacity on path
- Increase flow on the path by the bottleneck

When finished, resulting flow is maximal

Min-cut

If no augmenting path exists in residual network, then

- Find set of nodes S reachable from s in residual network
- Set $T = V \setminus S$

S and T define a minimum cut

Max-flow = Min-cut

Gives a way to verify that the step-by-step calculations of the flow are correct!

TIME FOR FINDING MAX-FLOW (OR MIN-CUT)

Upper bound (assuming integral capacities)

Upper bound (assuming integral capacities)

- It takes $O(E)$ time to find a path in the residual network (use for example breadth-first search)

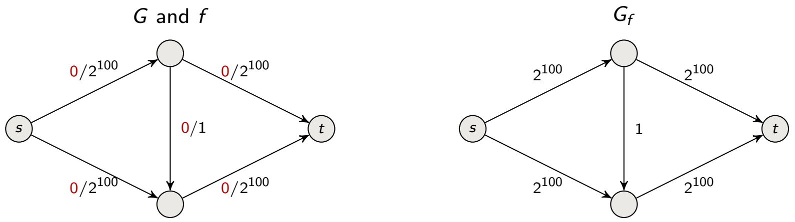
Upper bound (assuming integral capacities)

- It takes $O(E)$ time to find a path in the residual network (use for example breadth-first search)
- Each time the flow value is increased by at least 1

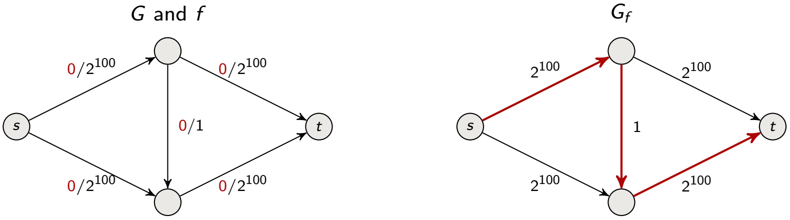
Upper bound (assuming integral capacities)

- It takes $O(E)$ time to find a path in the residual network (use for example breadth-first search)
- Each time the flow value is increased by at least 1
- Running time is $O(E \cdot |f_{\max}|)$ where $|f_{\max}|$ denotes the value of a maximum flow

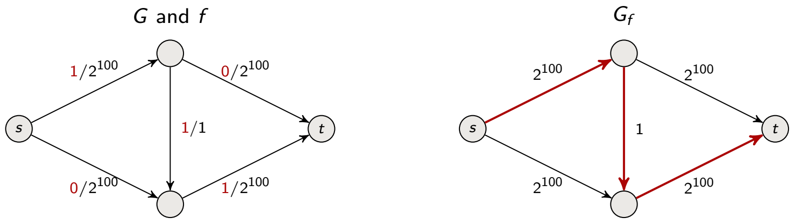
Problematic case



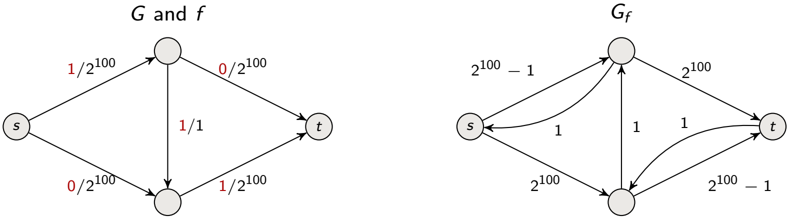
Problematic case



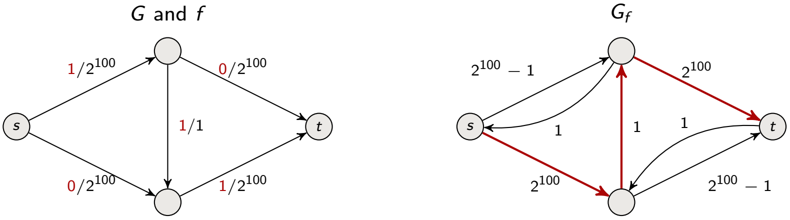
Problematic case



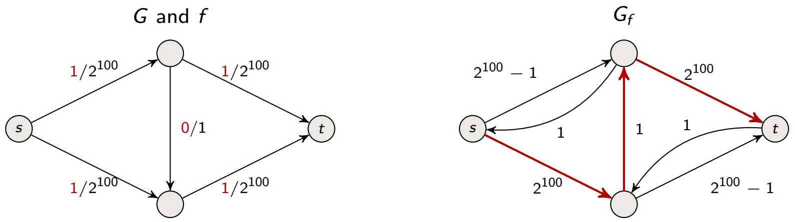
Problematic case



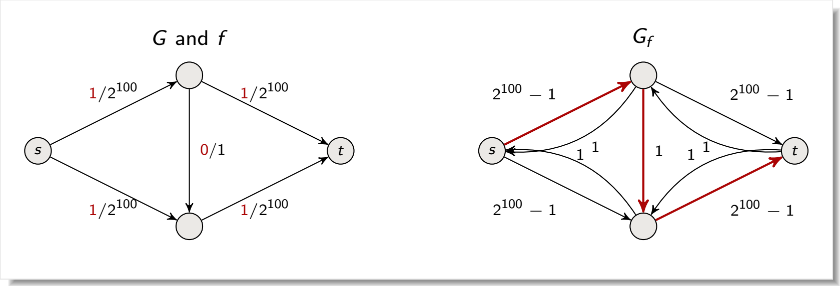
Problematic case



Problematic case



Problematic case



Problematic case

Problematic case

- you graduate

Problematic case

- you graduate
- I retire

Problematic case

- you graduate
- I retire
-

Problematic case

- you graduate
- I retire
-
- The sun stops to shine

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-
- Something happens to the universe

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-
- Something happens to the universe
-

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-
- Something happens to the universe
-
-

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-
- Something happens to the universe
-
-
-

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-
- Something happens to the universe
-
-
-
-

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-
- Something happens to the universe
-
-
-
-
-

Problematic case

- you graduate
- I retire
-
- The sun stops to shine
-
- Something happens to the universe
-
-
-
-
-
- Our algorithm returns a max-flow

Even more bad news

If capacities are irrational then the Ford-Fulkerson method might not terminate

Good news

If we either take the shortest path or the fattest path then this will not happen if the capacities are integers without proof

Good news

If we either take the shortest path or the fattest path then this will not happen if the capacities are integers without proof

BFS shortest path	$\leq \frac{1}{2} E \cdot V$
Fattest path	$\leq E \cdot \log(E \cdot U)$

Good news

If we either take the shortest path or the fattest path then this will not happen if the capacities are integers without proof

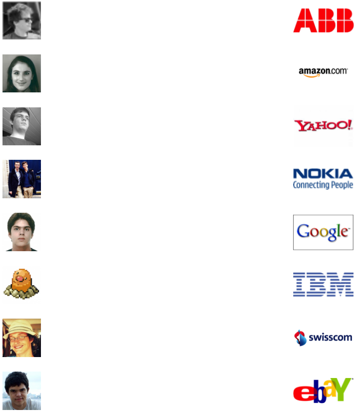
BFS shortest path	$\leq \frac{1}{2} E \cdot V$
Fattest path	$\leq E \cdot \log(E \cdot U)$

- U is the maximum flow value
- Fattest path: choose augmenting path with largest minimum capacity (bottleneck)

APPLICATIONS OF MAX-FLOW

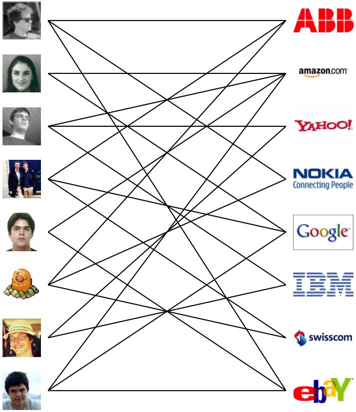
Bipartite matching

- N students apply for M jobs



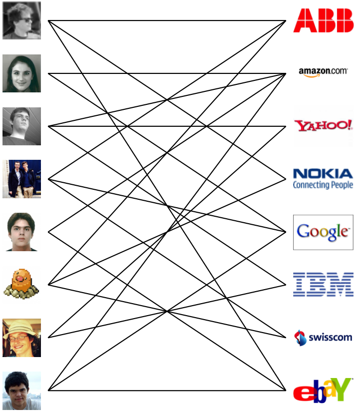
Bipartite matching

- N students apply for M jobs
- Each get several offers

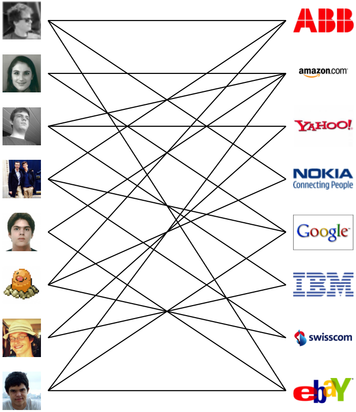


Bipartite matching

- N students apply for M jobs
- Each get several offers
- Is there a way to match all students to jobs? obviously M has to be at least equal to N

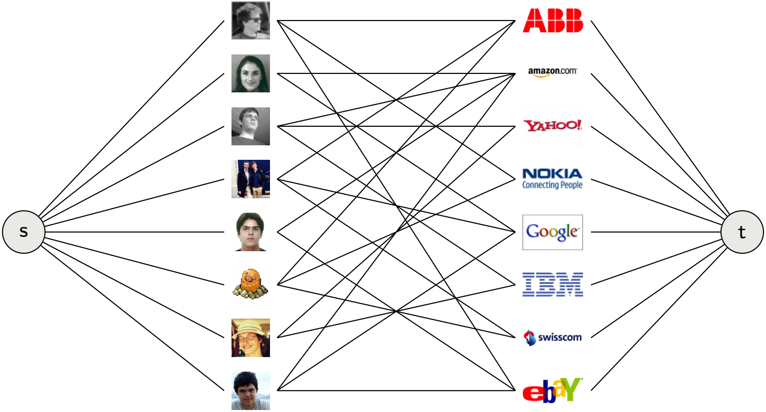


Bipartite matching as flow problem



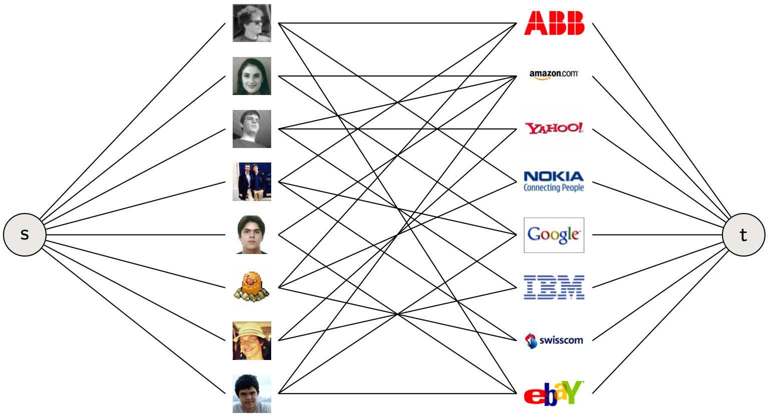
Bipartite matching as flow problem

- Add source s and sink t with edges from s to students and from jobs to t



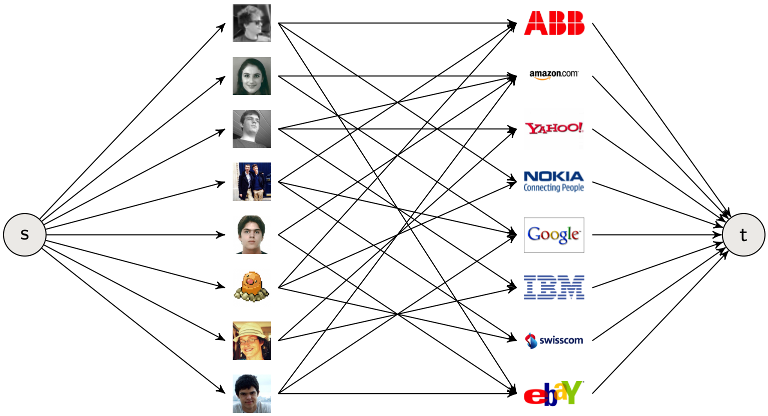
Bipartite matching as flow problem

- Add source s and sink t with edges from s to students and from jobs to t
- All edges have capacity one



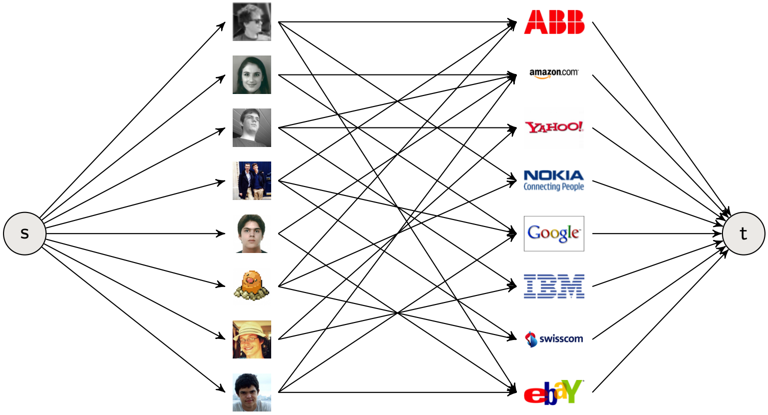
Bipartite matching as flow problem

- Add source s and sink t with edges from s to students and from jobs to t
- All edges have capacity one
- Direction is from left to right



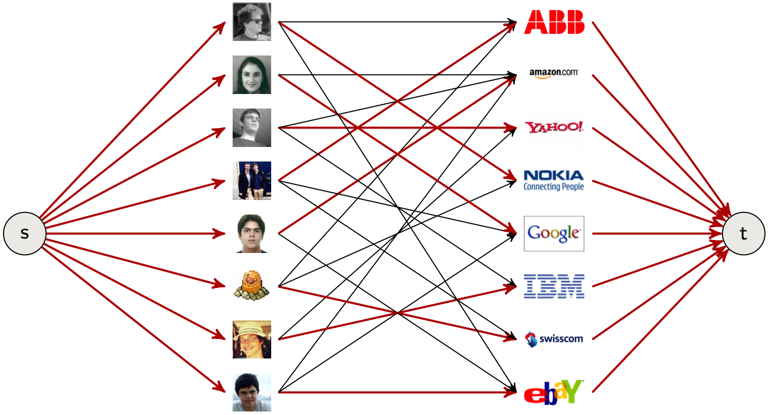
Bipartite matching as flow problem

- Run the Ford-Fulkerson method



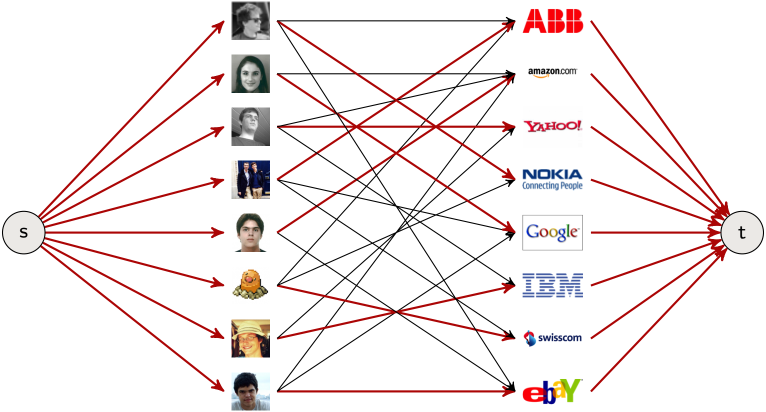
Bipartite matching as flow problem

- Run the Ford-Fulkerson method



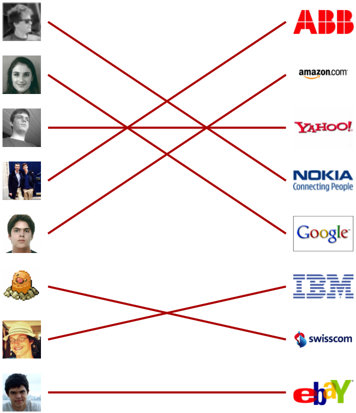
Bipartite matching as flow problem

- Run the Ford-Fulkerson method
- Matching is complete



Bipartite matching as flow problem

- Run the Ford-Fulkerson method
- Matching is complete



Why does it work?

Every matching defines a flow of value equal to the number of edges in matching

- Put flow 1 on
  - Edges of the matching
  - Edges from s to matched student nodes
  - Edges from matched job nodes to t
- Put flow 0 on all other edges

Works because flow conservation is equivalent to: no student is matched more than once, no job is matched more than once

Why does it work?

Every flow during the algorithm defines a matching of size equal to its value

- Flows obtained by Ford-Fulkerson are integer valued if capacities are integral, so value on every edge is 0 or 1
- Edges between students and jobs with flow 1 are a matching by flow conservation
  - There cannot be more than one edge with flow 1 from a student node
  - There cannot be more than one edge with flow 1 into a job node

Why does it work?

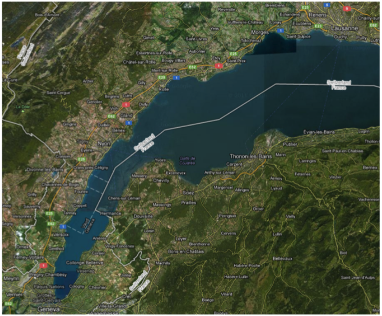
Every flow during the algorithm defines a matching of size equal to its value

- Flows obtained by Ford-Fulkerson are integer valued if capacities are integral, so value on every edge is 0 or 1
- Edges between students and jobs with flow 1 are a matching by flow conservation
  - There cannot be more than one edge with flow 1 from a student node
  - There cannot be more than one edge with flow 1 into a job node

So, maximum flow is a maximum matching!

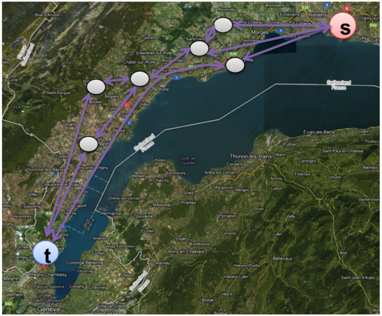
Edge-disjoint paths

- You want to travel to a nice location these winter holidays
- You need to drive from Lausanne to Geneva airport
- Winter season $\Rightarrow$ risk that roads are closed
- How many different routes can you take that does not share a common road?



Edge-disjoint paths as flow network

- s = Lausanne
- t = Geneva airport
- An edge capacity of 1 in both directions for each road
- (make anti-parallel using gadgets)



Solution

- max-flow = # edge-disjoint paths
- min-cut = min #roads to be closed so that there is no route from Lausanne to Geneva airport