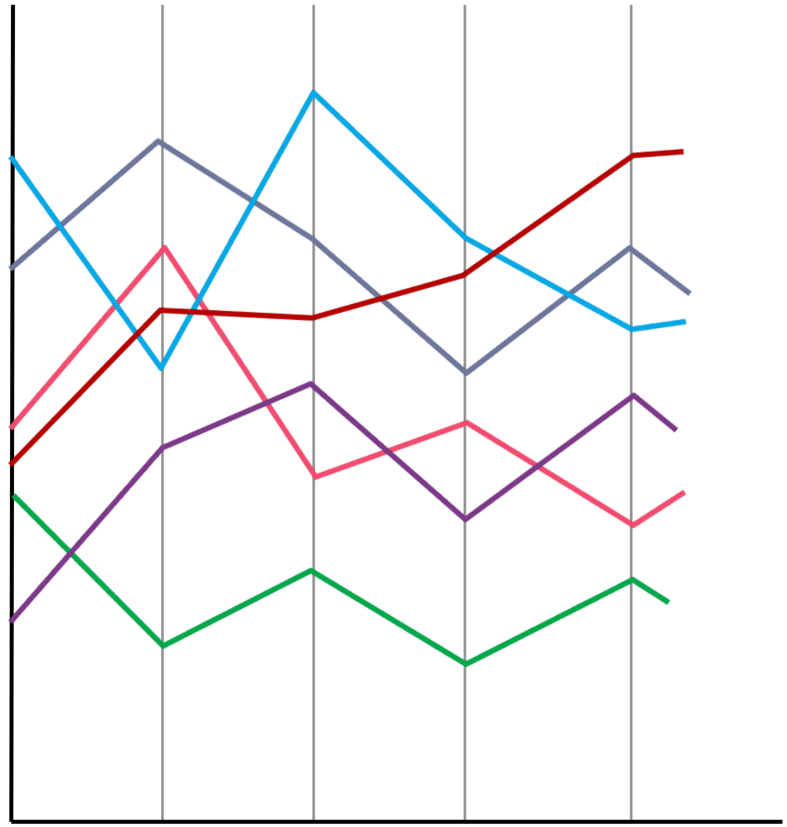
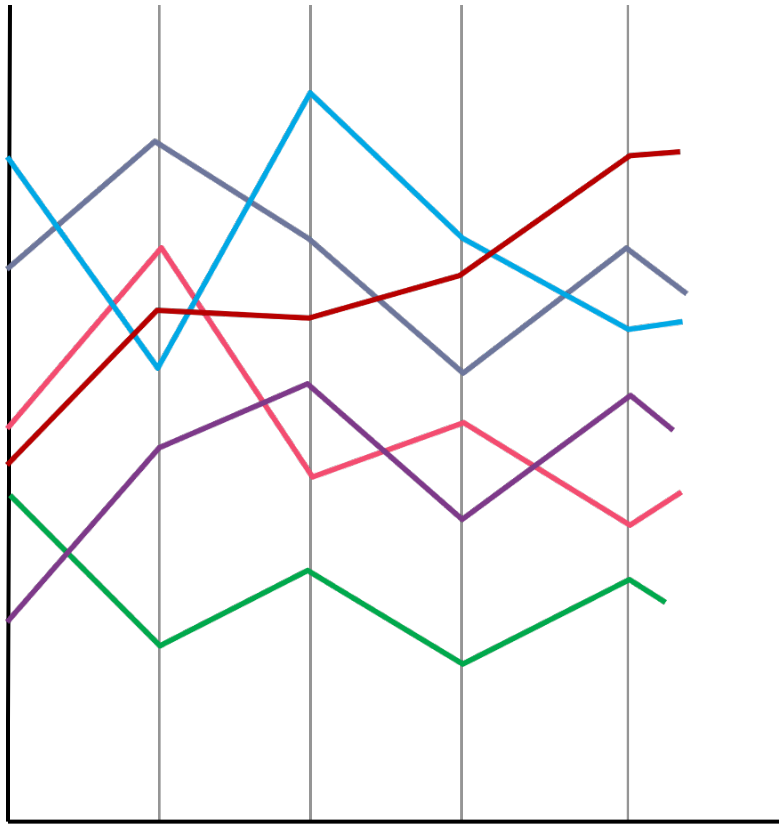
Plot majoration

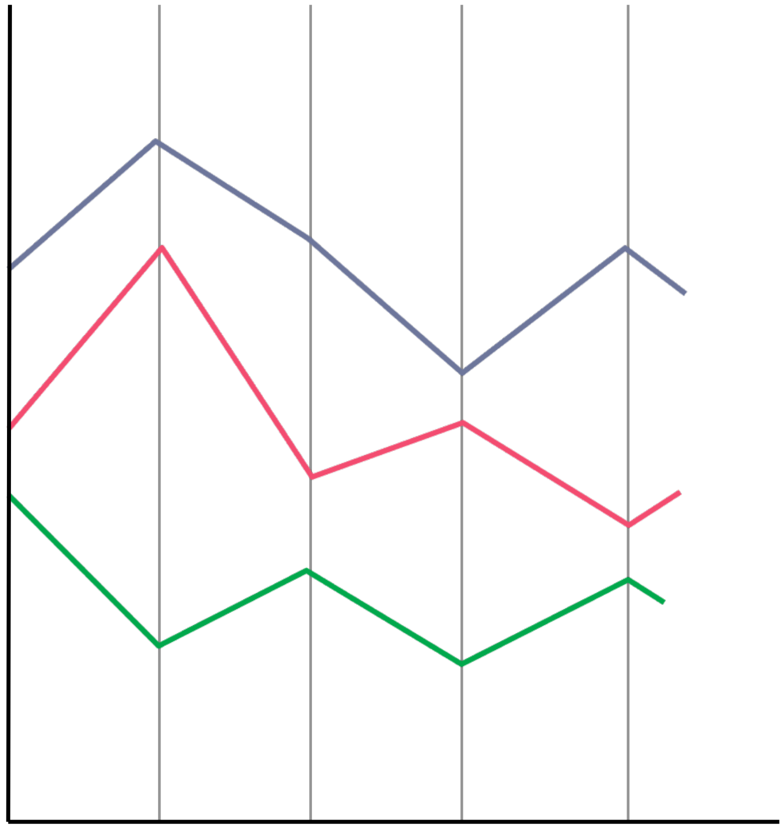
Problem: compute the largest subset of “strictly majorating” plots



Plot majoration

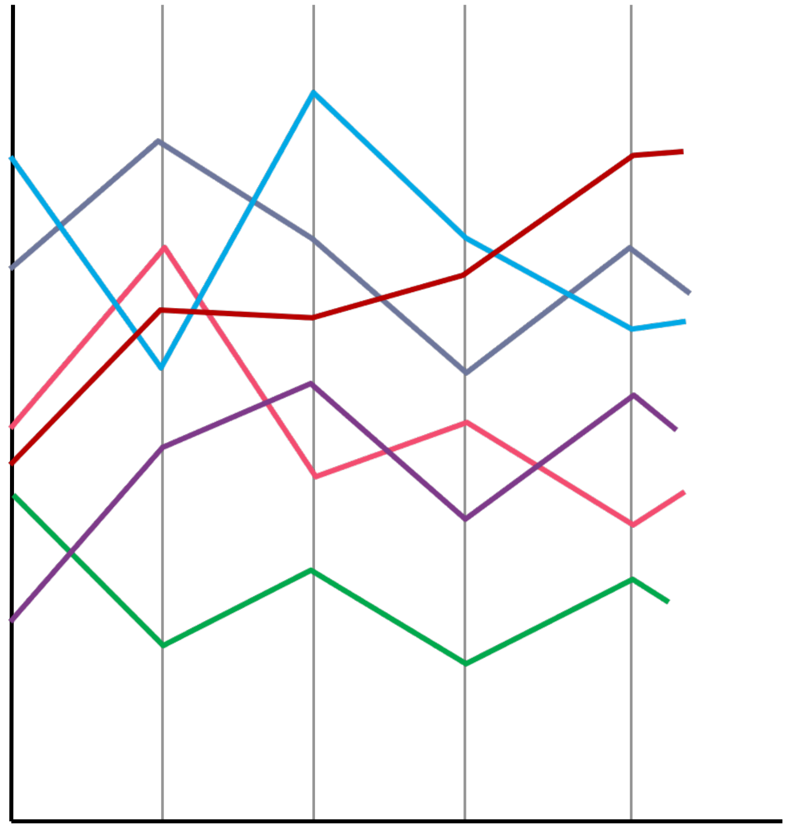
Problem: compute the largest subset of “strictly increasing” plots



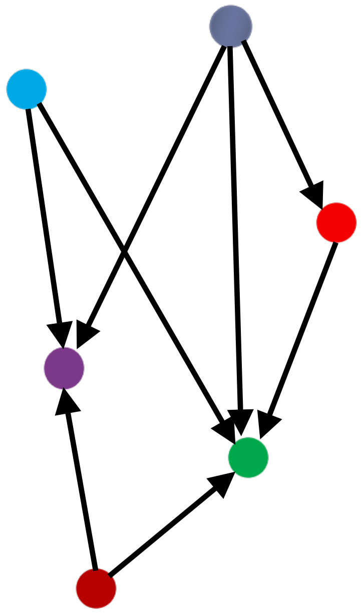


Plot majoration

Problem: compute the largest subset of “strictly majorating” plots

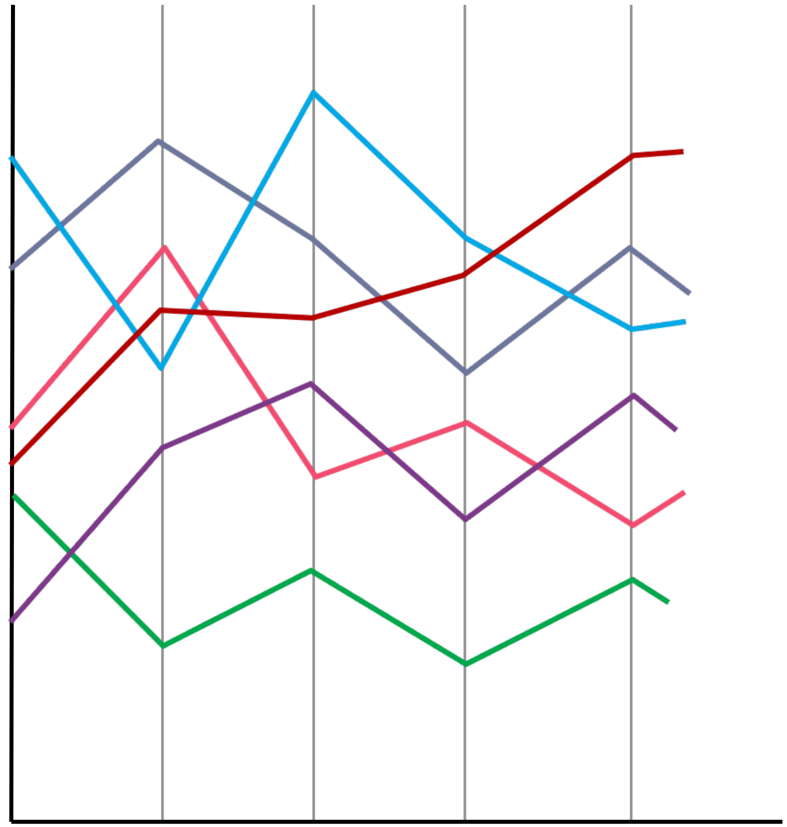


DAG of plot majoration

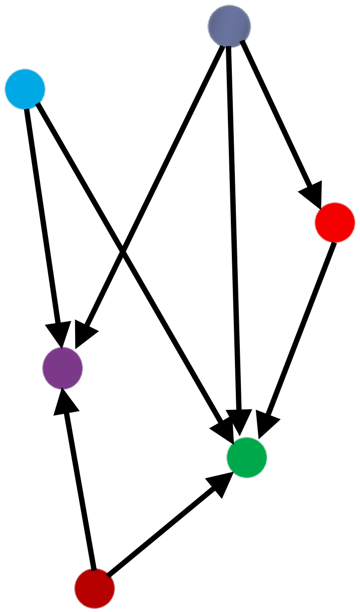


Plot majoration

Problem: compute the largest subset of “strictly majorating” plots

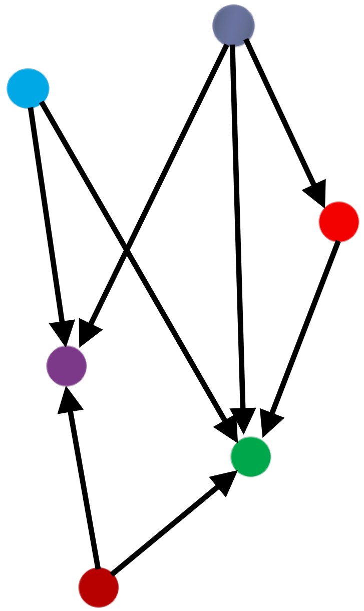


DAG of plot majoration



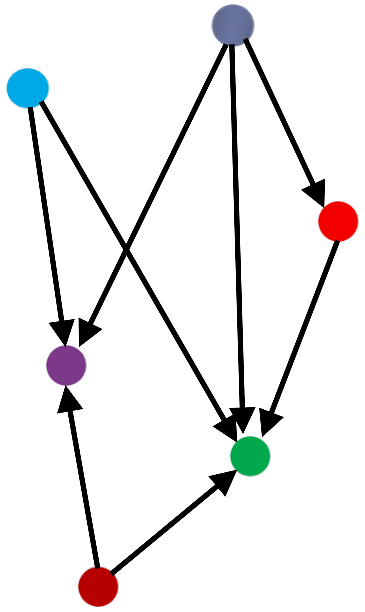
Problem: compute the **longest** path in a DAG

Longest path in a DAG

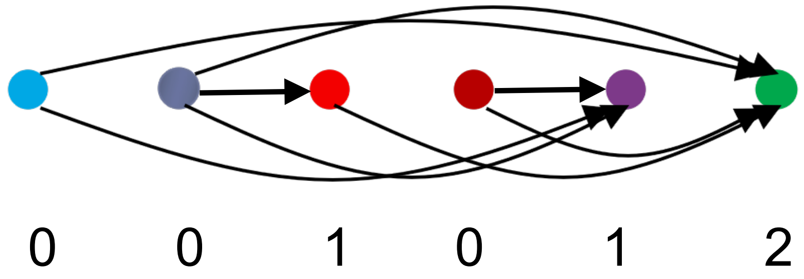


Longest path in a DAG

```
compute the topological sort of the graph
for all $u \in V$ initialize $l(u) = 0$
for all $u \in V$ in topological order
        $l(u) = \max_{(v,u)\in E} l(v) + 1$
output $u$ with maximum $l(u)$
```

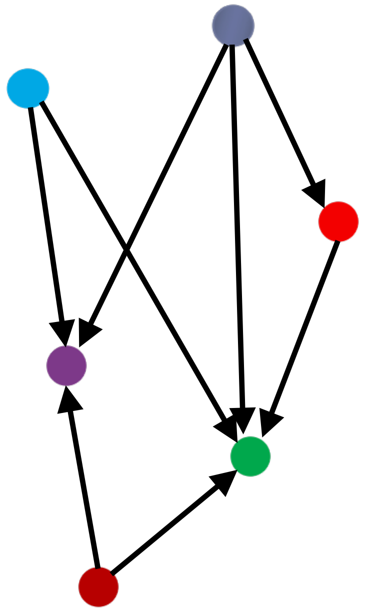


$l(u)$: nb of edges on the longest path ending at u

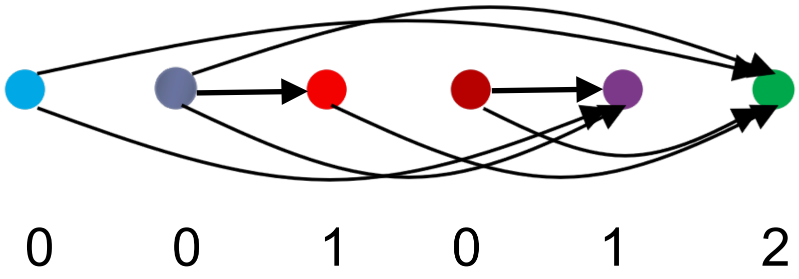


Longest path in a DAG

```
compute the topological sort of the graph
for all $u \in V$ initialize $l(u) = 0$
for all $u \in V$ in topological order
        $l(u) = \max_{(v,u)\in E} l(v) + 1$
output $u$ with maximum $l(u)$
```



$l(u)$: nb of edges on the longest path ending at u



A longest path in a DAG can be computed in time $O(n+m)$

Maximum flow in networks

and some other Combinatorial optimization problems

Flow network

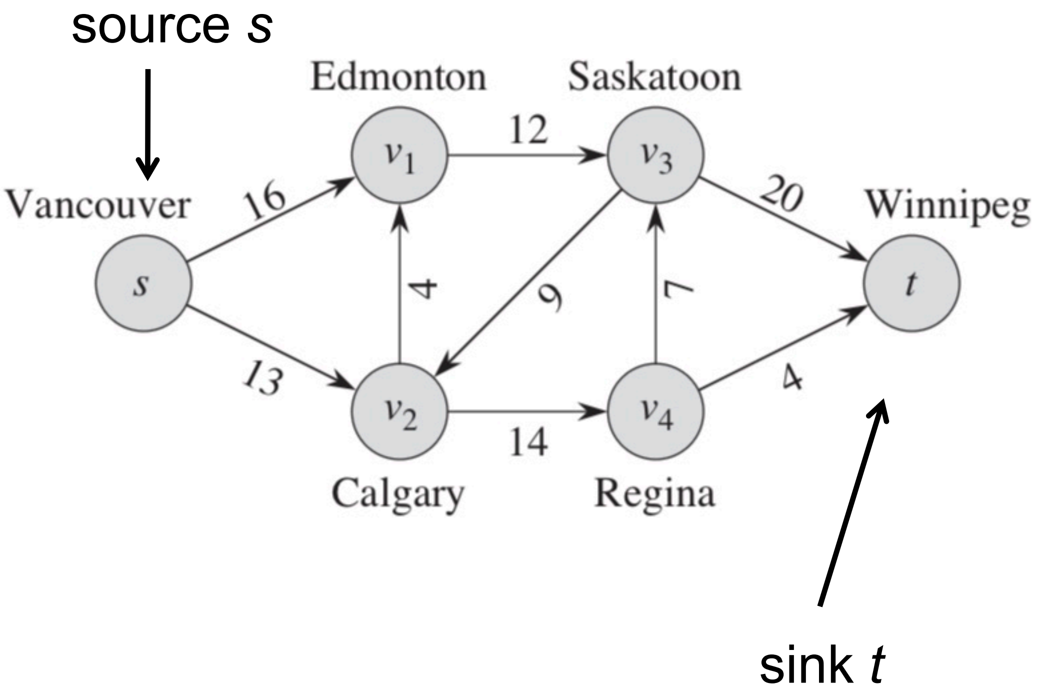
Directed weighted graph $G = (V, E, c)$

$c(p, q)$: capacity of edge (p, q)
source $s \in V$, sink $t \in V$

Accessiblity assumption: all nodes appear on a path from s to t

Examples:

- Water systems
- Production lines
- Traffic roads
- Transportation of goods
- Electricity
- etc.



source s

sink t

Flow network (cont)

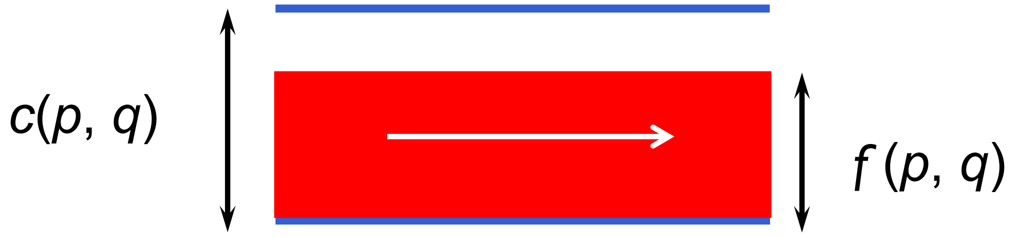
Capacity $c: V \times V \rightarrow \mathbb{R}$ with $c(p,q) \geq 0$

if $(p,q) \notin E$, then assume $c(p,q) = 0$

Flow $f: V \times V \rightarrow \mathbb{R}$

Capacity constraint

for all $p,q \in V$, $f(p,q) \leq c(p,q)$

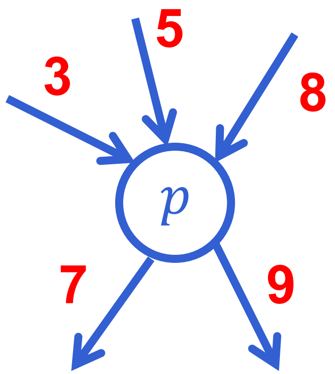


Flow network (cont)

Flow conservation

for all $p \in V \backslash \{s, t\}$,

$$\sum\{f(q,p) | (q,p) \in E\} = \sum\{f(p,q) | (p,q) \in E\}$$

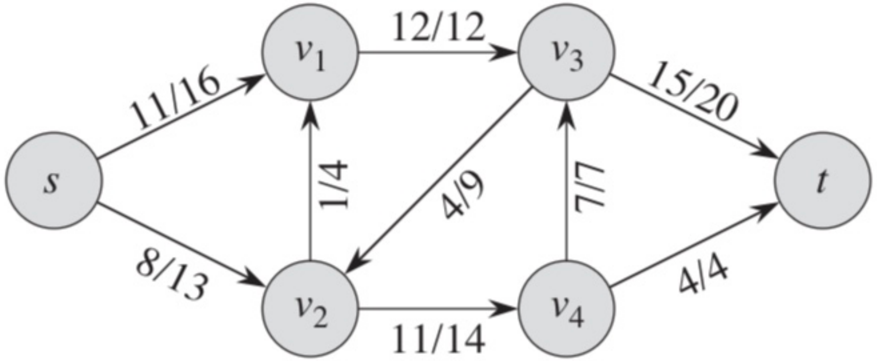


Flow

Flow value (definition):

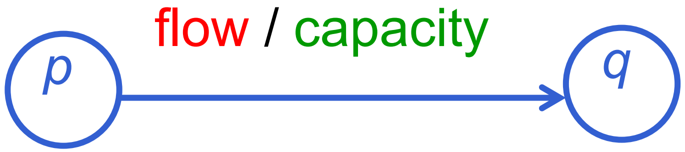
$$|f| = \sum_{p \in V} f(s,p) - \sum_{p \in V} f(p,s)$$

what flows out of the source minus what flows into the source



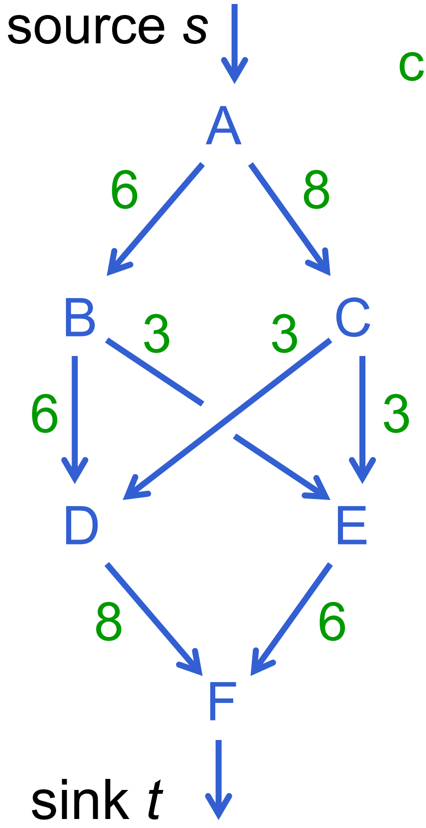
Property:

$$|f| = \sum_{p \in V} f(p,t) - \sum_{p \in V} f(t,p)$$



Maximum flow problem

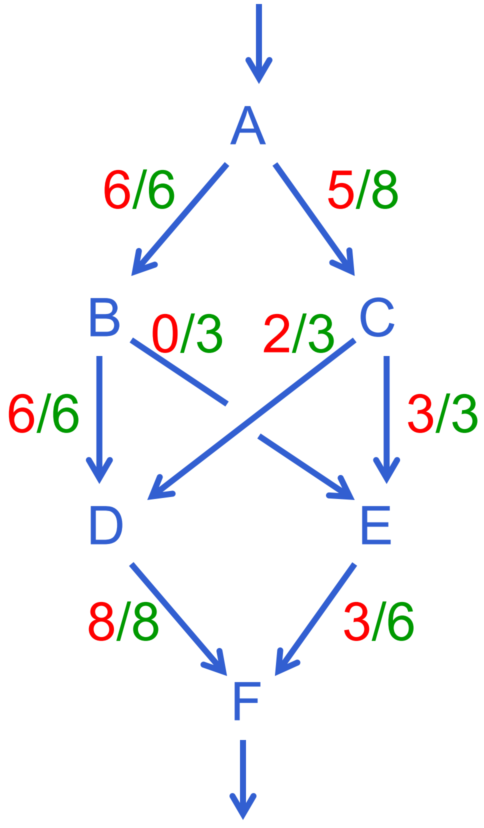
Given a flow network $G = (S, A, c)$, compute the **maximum flow**, *i.e.* the flow of maximum value $|f|$



capacities

maximum flow ?

$|f| = 11$

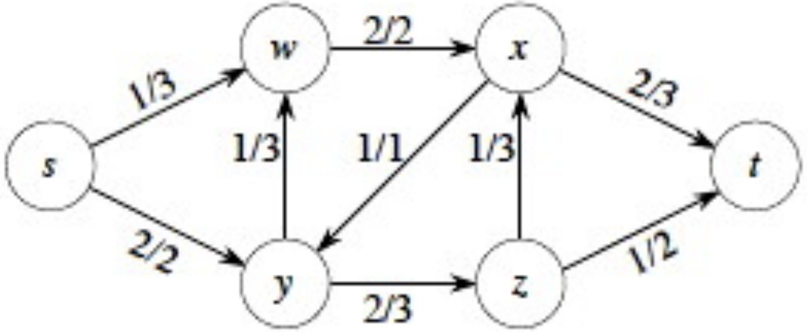


Residual capacity of an edge

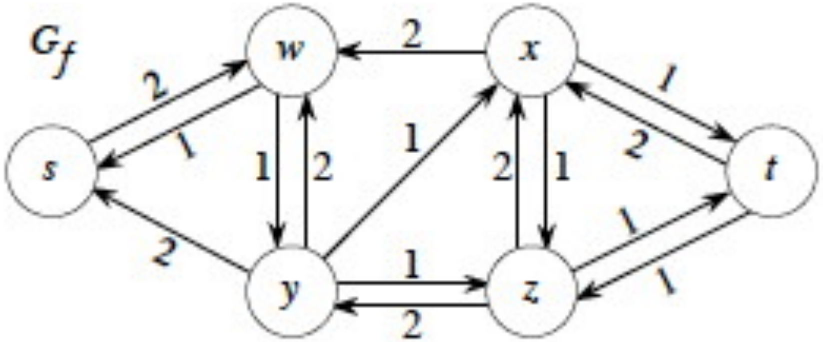
for each edge $(p,q) \in E$ with current flow $f(p,q)$, define

$$c_f(p,q) = c(p,q) - f(p,q)$$

$$c_f(q,p) = f(p,q)$$



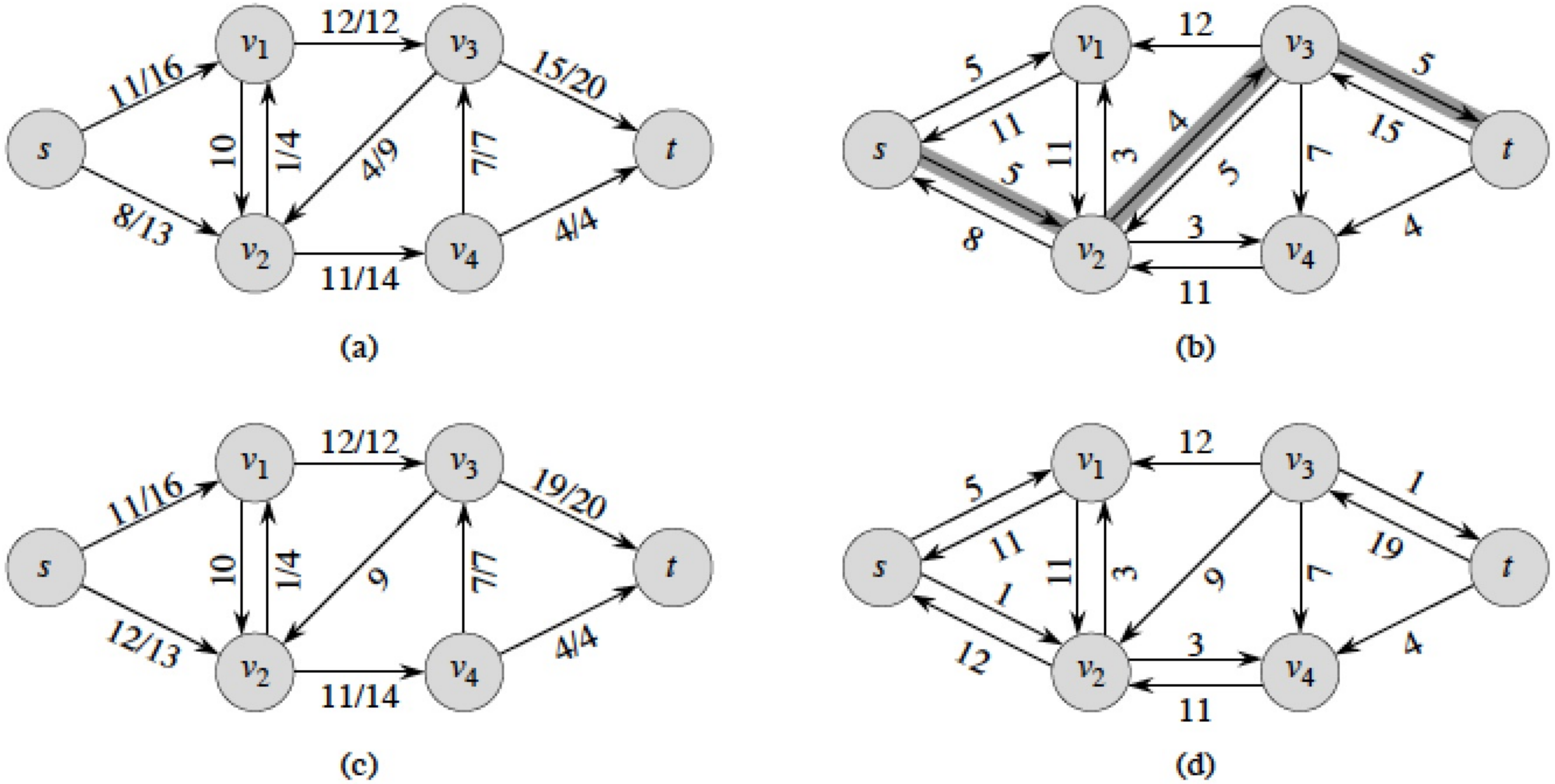
flow network



G_f: residual network

Augmenting path

(b), (d): residual networks



(a)

(b)

(c)

(d)

An augmenting path is a simple path in the residual network

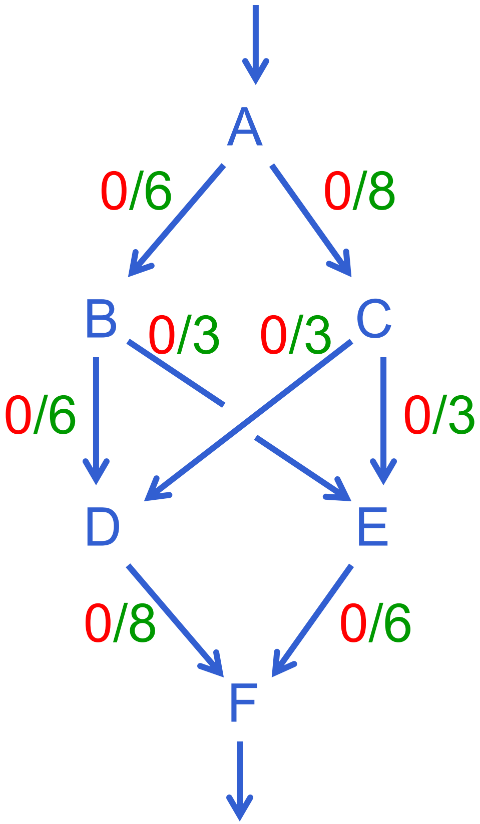
Ford-Fulkerson method (1962)

```
initialize flow $f$ to $0$ ;
while there exists an augmenting path from $s$ to $t$ do
        augment flow $f$ along this path by the residual
        capacity of the path(*)
return $f$
```

(*) minimum residual capacity of an edge on the path

Note: augmenting the flow means incrementing the flow on “forward edges” and decrementing the flow on “backward edges”

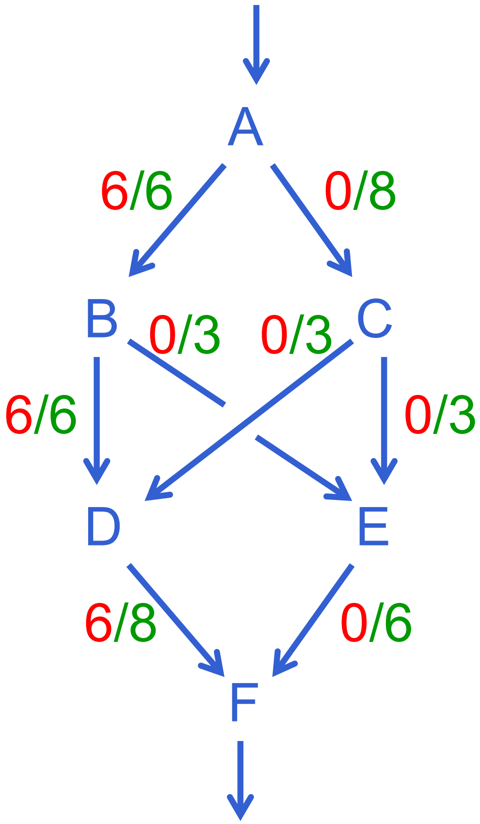
Example



$|f| = 0$

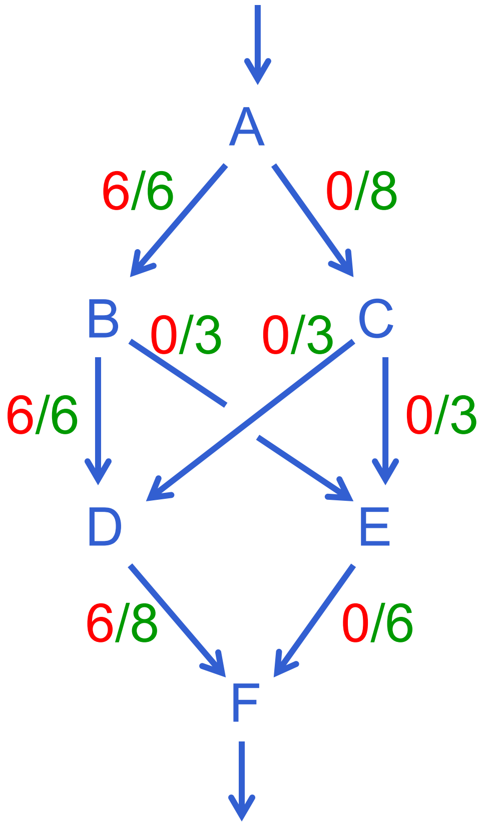
ABDF
augmenting path

augmenting
the flow along ABDF



$|f| = 6$

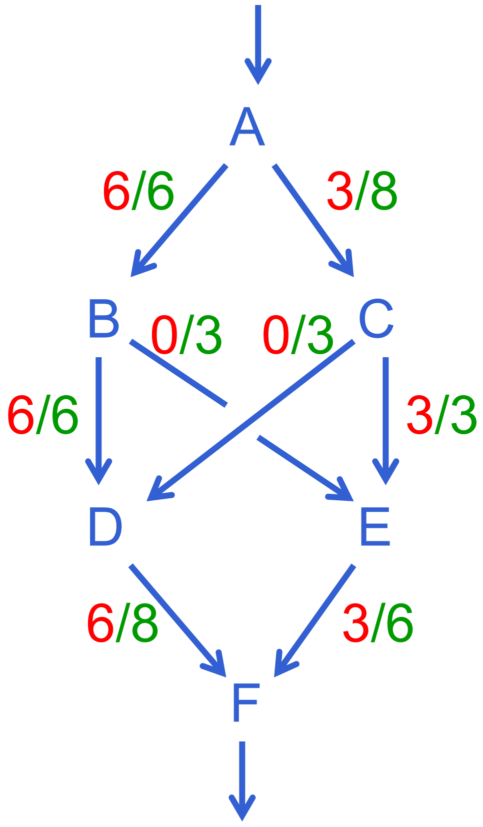
Example (cont)



$|f| = 6$

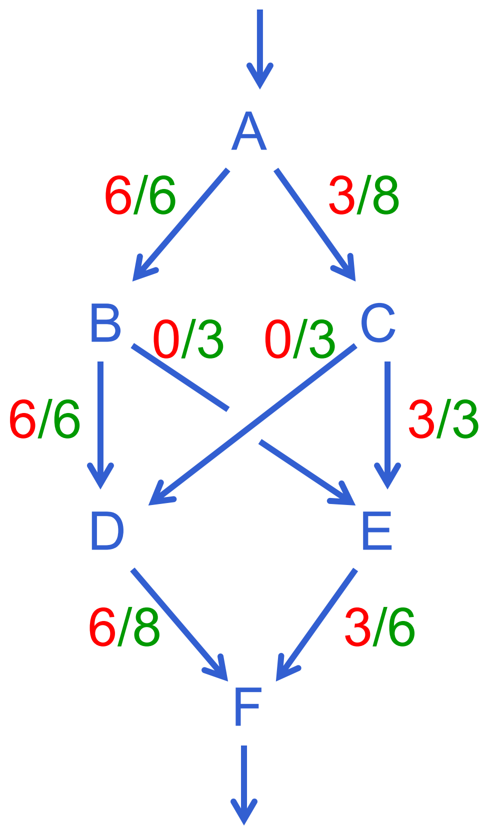
ACEF
augmenting path

augmenting
the flow along ACEF



$|f| = 9$

Example (cont)

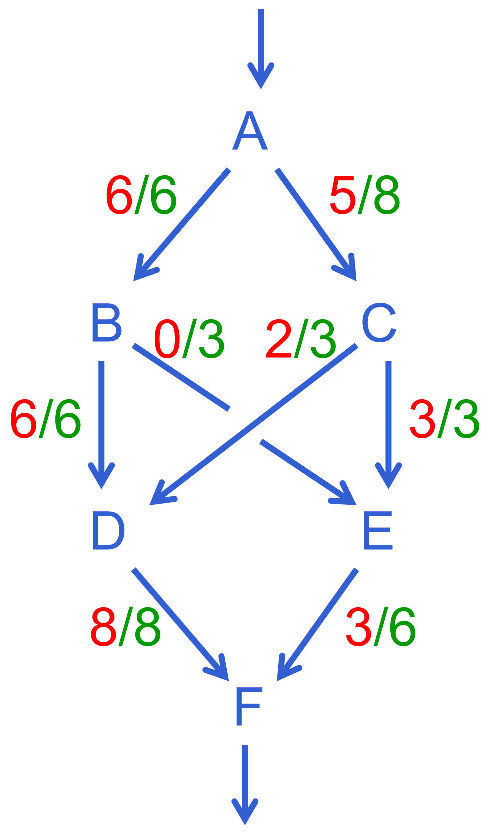


$|f| = 9$

ACDF
augmenting path

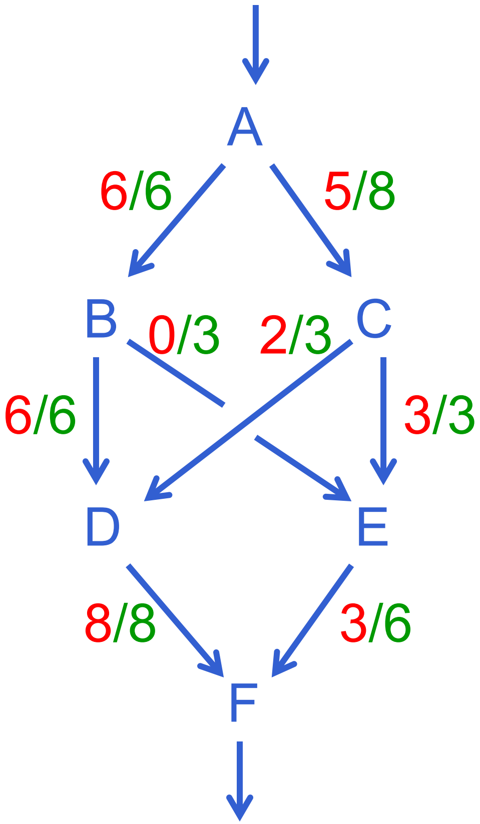
→

augmenting
the flow along ACDF



$|f| = 11$

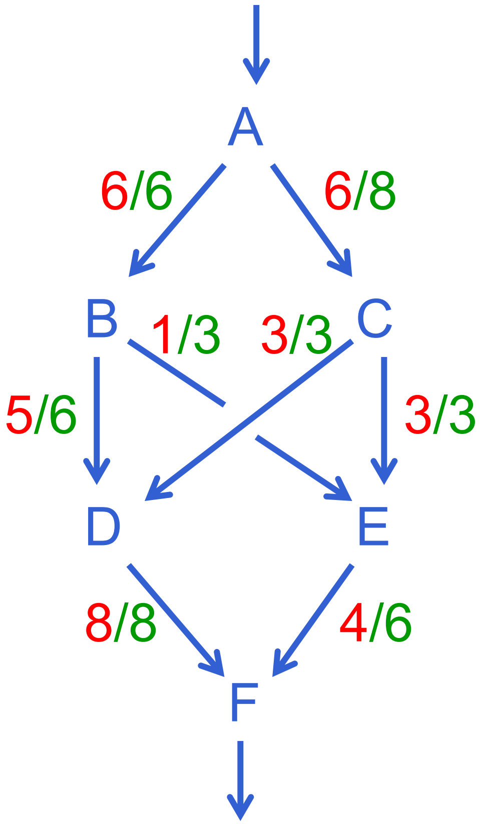
Example (cont)



$| f | = 11$

ACDBEF
augmenting path

augmenting
the flow along ACDBEF



$| f | = 12$
maximum

Cut

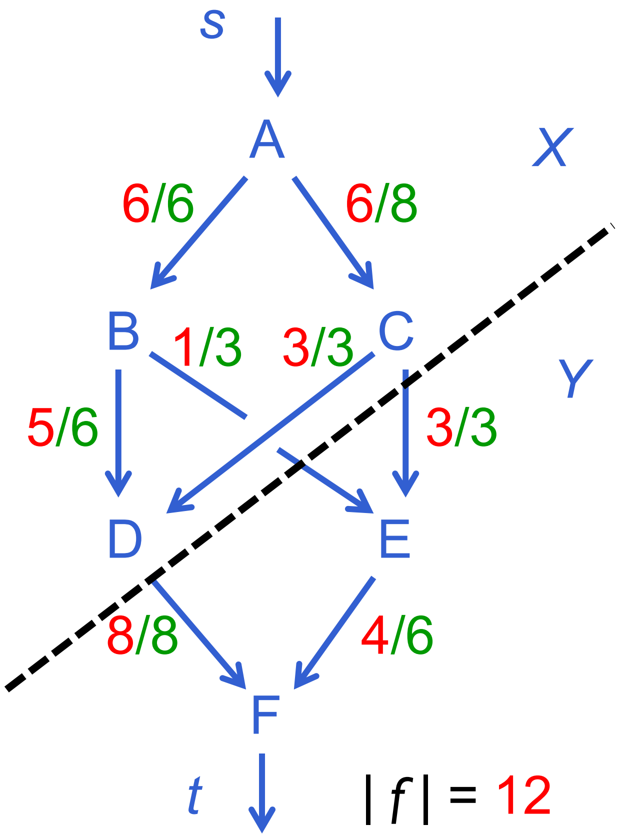
Cut (definition)**:**
(X, Y) cut of $G = (V, E, c)$:
(X, Y) partition of V such that $s \in X, t \in Y$

Capacity of the cut:

$$c(X, Y) = \sum\{c(x, y) | x \in X, y \in Y\}$$

Flow through the cut:

$$f(X, Y) = \sum\{f(x, y) | x \in X, y \in Y\} - \sum\{f(y, x) | x \in X, y \in Y\}$$

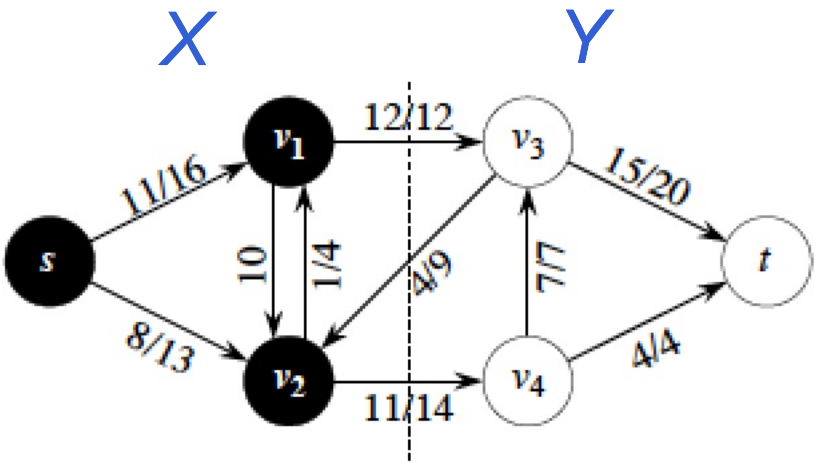


| *f* | = 12

X = {A,B,C,D}   *Y* = {E,F}   *c*(*X*, *Y*) = 14   *f* (*X*, *Y*) = 12

Cut (cont)

Note that the flow from Y to X ***is*** counted negatively, but the capacity does ***not*** take into account edges from Y to X



$c(X, Y) = 26$ $f(X, Y) = 19$

Properties

Properties Let (X, Y) be a cut. Then

(*i*) $f\,(X, Y) = |\,f\,|$

(*ii*) $f\,(X, Y) \leq c(X, Y)$

The maximum flow is bounded by the minimum capacity of a cut

Properties

Properties Let (X, Y) be a cut. Then

(*i*) $f\,(X, Y) \;=\; |\, f\,|$

(*ii*) $f\,(X, Y) \;\leq\; c(X, Y)$

The maximum flow is bounded by the minimum capacity of a cut

Theorem (*max-flow min-cut theorem*)
The following conditions are equivalent:

(*i*) f is a maximum flow

(*ii*) there is no augmenting paths in the residual network

(*iii*) $|\, f\,| \;=\; c(X', Y')$ for some cut (X', Y')

$\Rightarrow$ *maximum flow equals minimum cut capacity*

Minimum cut

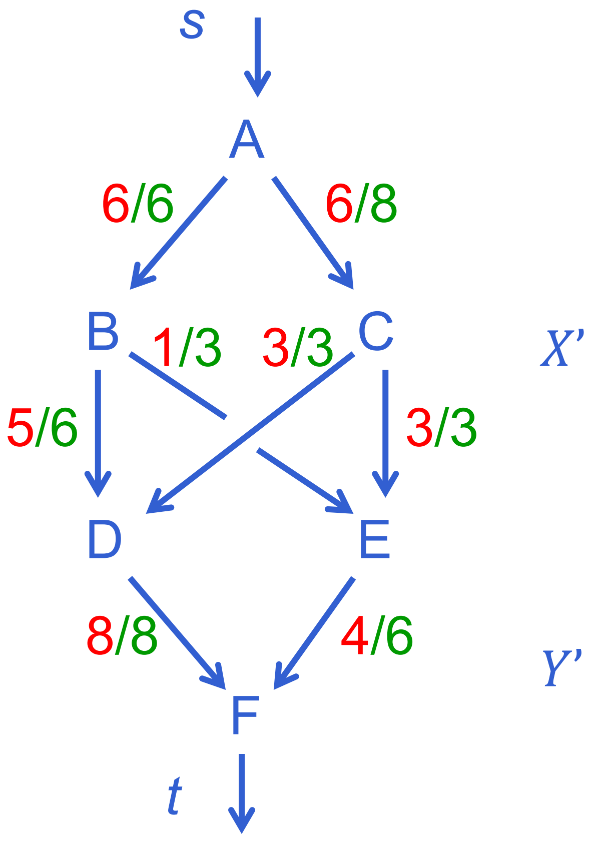
$X' =$

$Y' =$

$c(X', Y') = 12$

(X', Y') of minimum capacity

$f\ (X', Y') = 12$ is the maximum flow



$|f| = 12$

Minimum cut

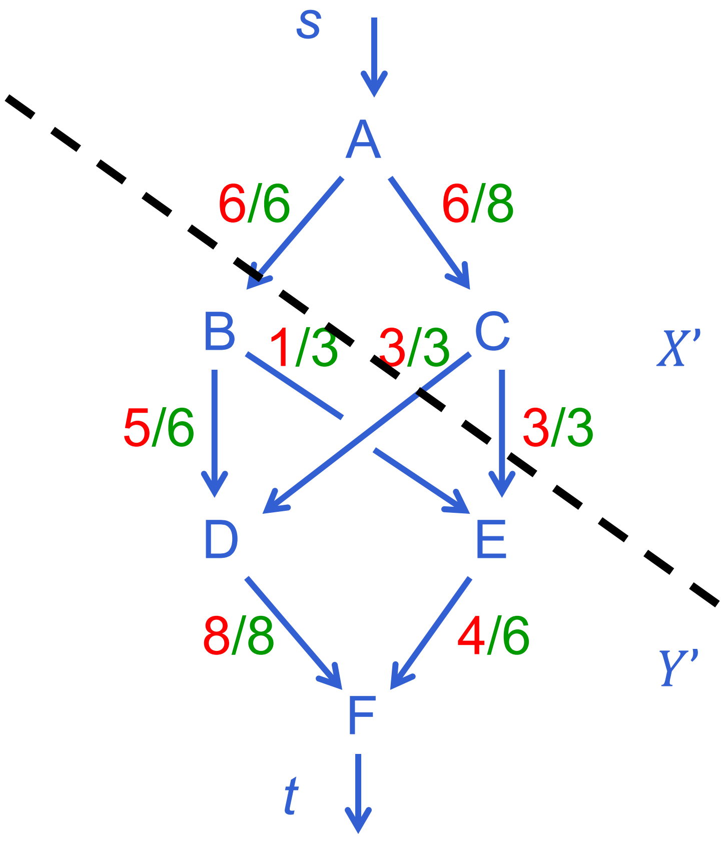
$X' = \{A, C\}$

$Y' = \{B, D, E, F\}$

$c(X', Y') = 12$

(X', Y') of minimum capacity

$f\,(X', Y') = 12$ is the maximum flow



$|f| = 12$

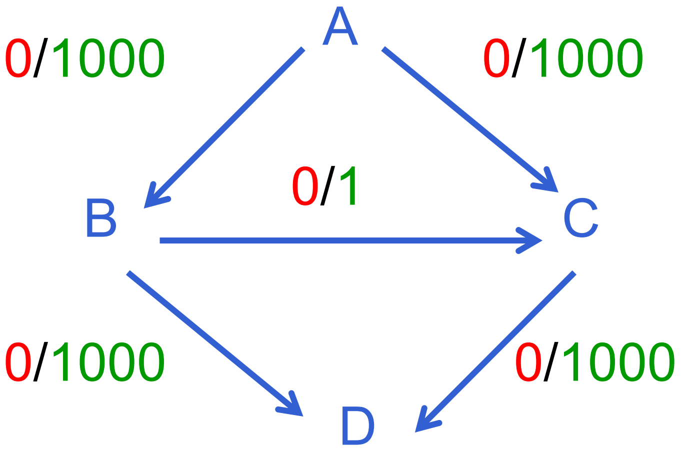
Implementation of Ford-Fulkerson method

```
initialize flow $f$ to $0$ ;
while there exists an augmenting path from $s$ to $t$ do
        augment flow $f$ along this path
return $f$
```

How to choose the augementing path?

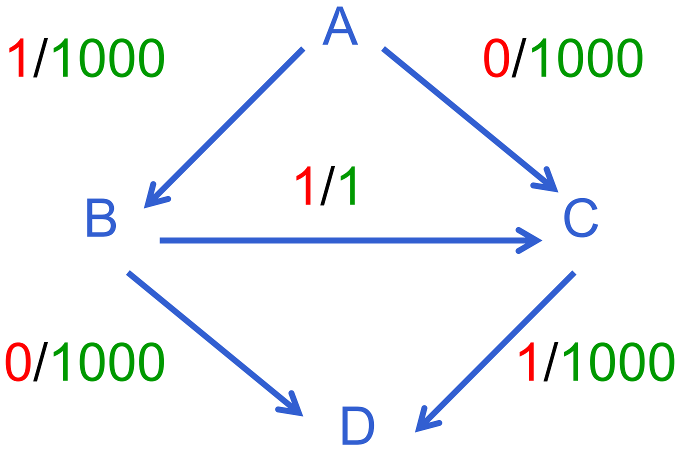
Example

The number of iterations depends on the choice of the paths



Example (cont)

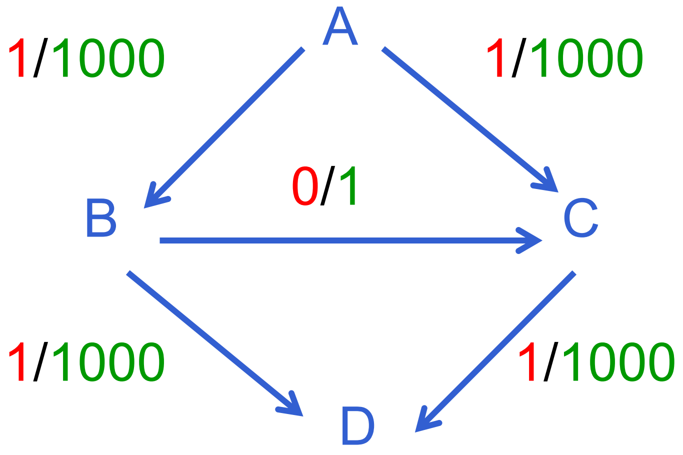
The number of iteration depends on the choice of the paths



augmentation	path
1	ABCD

Example (cont)

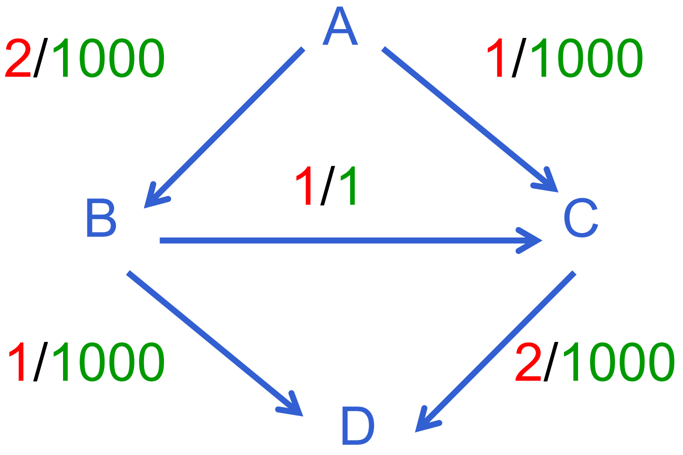
The number of iteration depends on the choice of the paths



augmentation	path
1	ABCD
1	ACBD

Example (cont)

The number of iteration depends on the choice of the paths

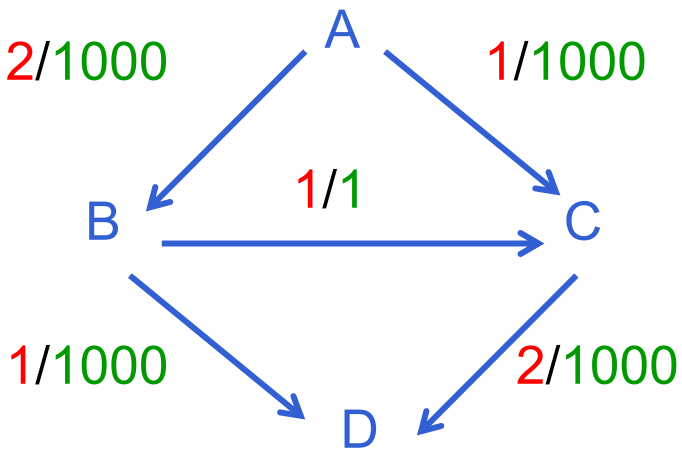


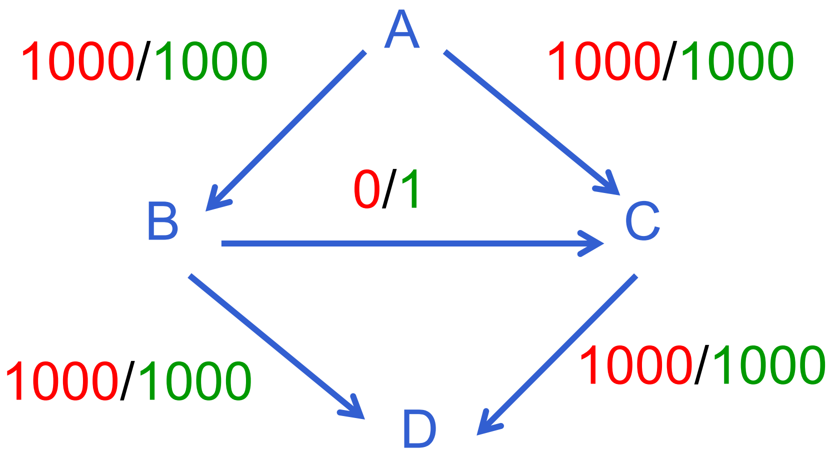
augmentation	path
1	ABCD
1	ACBD
1	ABCD

etc.

Example (cont)

The number of iteration depends on the choice of the paths





augmentation	path
1	ABCD
1	ACBD
1	ABCD

etc.

augmentation	path
1000	ABD
1000	ACD

maximum flow

Integer-valued flow

- If all capacities are integers, then all intermediate flow values and residual capacities are integers as well

- If C is the max-flow, then Ford-Fulkerson makes at most C iterations $\Rightarrow O(|E| \cdot C)$ time

Edmonds-Karp algorithm

Main idea: To augment the flow, choose the **shortest**[(*)] augmenting path in the residual network (using BFS)

[(*)] in terms of number of edges, i.e. without weights

Theorem
Computing the maximum flow using this strategy requires at most $n \cdot m$ augmentations. The running time is $O(n \cdot m^2)$

This strategy is known as the *Edmonds-Karp algorithm* (1972), but was discovered by Dinitz (1970)

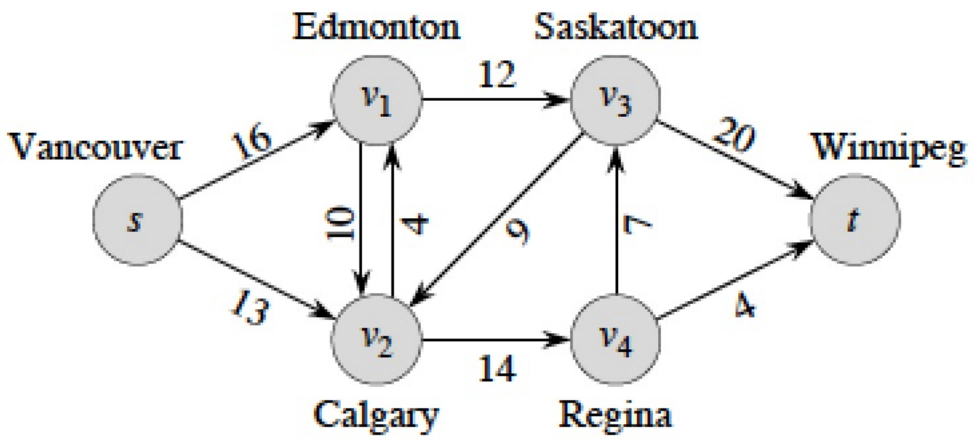
Other strategies

Push-relabel algorithm : $O(n^2 \cdot m)$

Relabel-to-front algorithm *:* $O(n^3)$

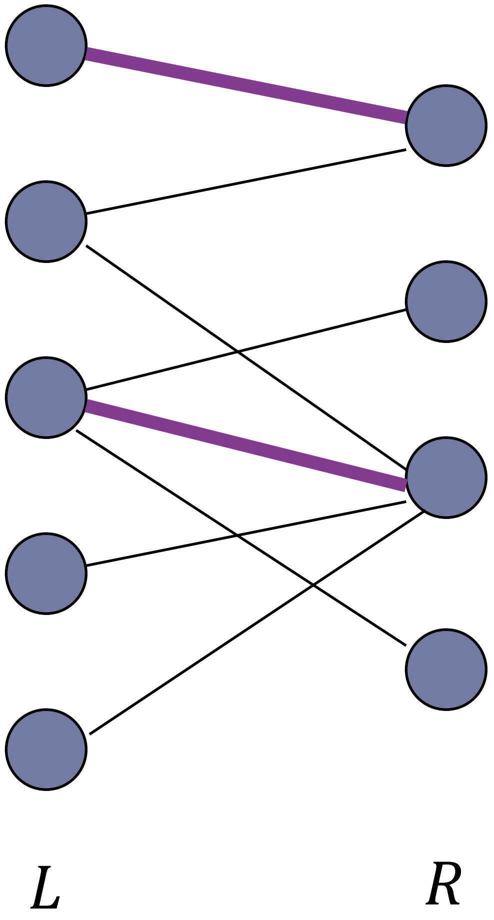
Exercise

- Run the Ford-Fulkerson algorithm on the following network:



Maximum bipartite matching

Maximum matching



Bipartite graph $G = (V, E)$, $V = L \uplus R$, and

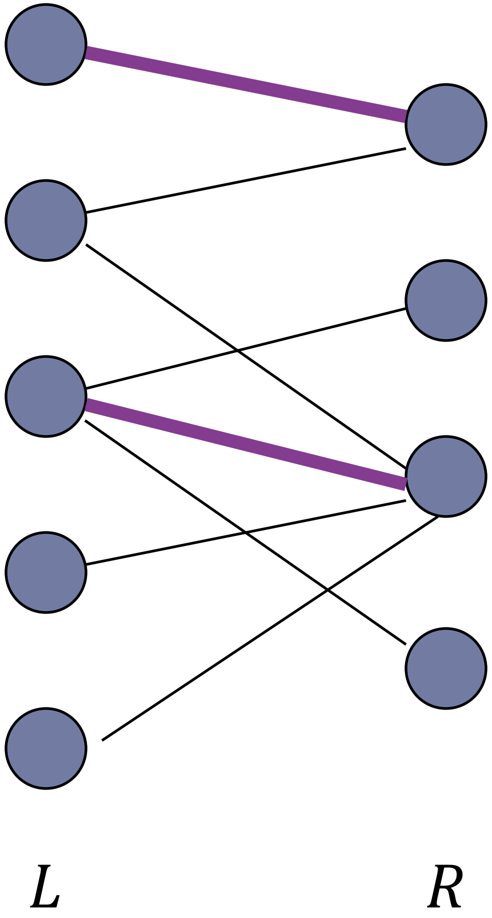
$$\forall (p, q) \in E, p \in L \; et \; q \in R$$

Matching: $C \subseteq E$ such that for all $p \in V$ $\exists$ at most one edge in C incident to p (i.e. having p as one of the endpoints)

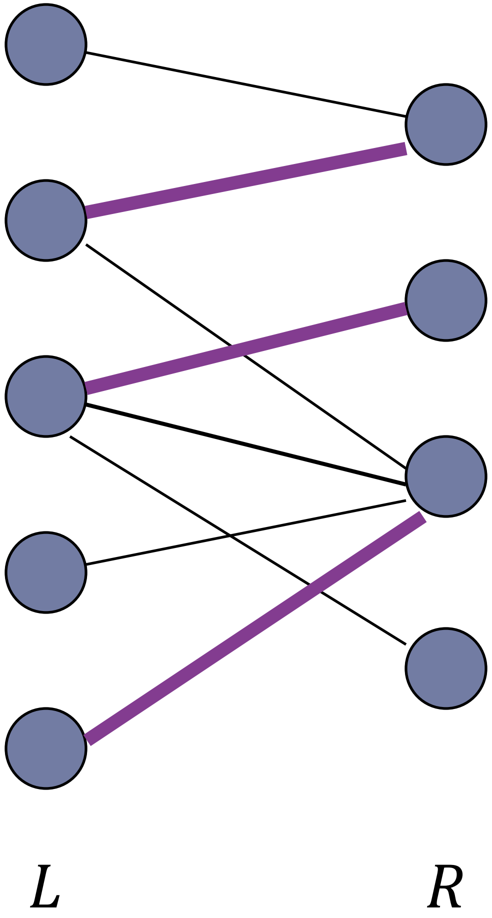
Maximum matching: matching with the maximum number of edges

NB: maximum ≠ maximal (by inclusion!)

Maximum vs maximal matching



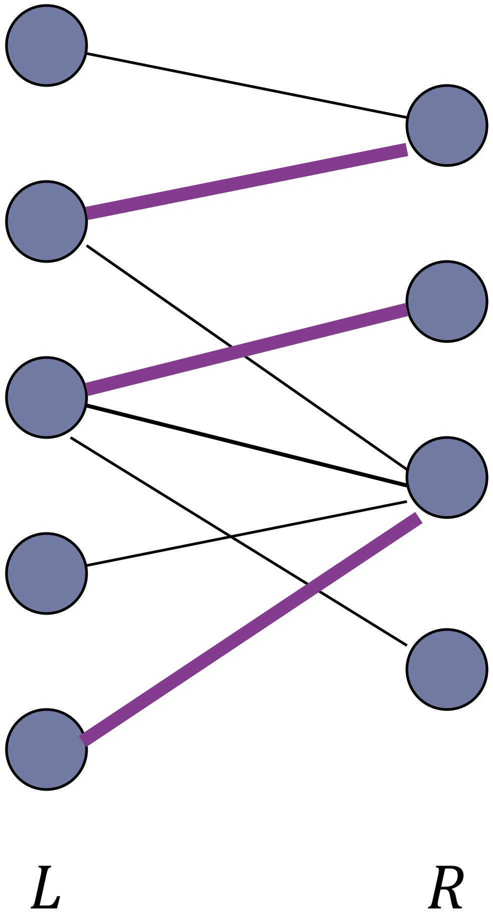
maximal matching



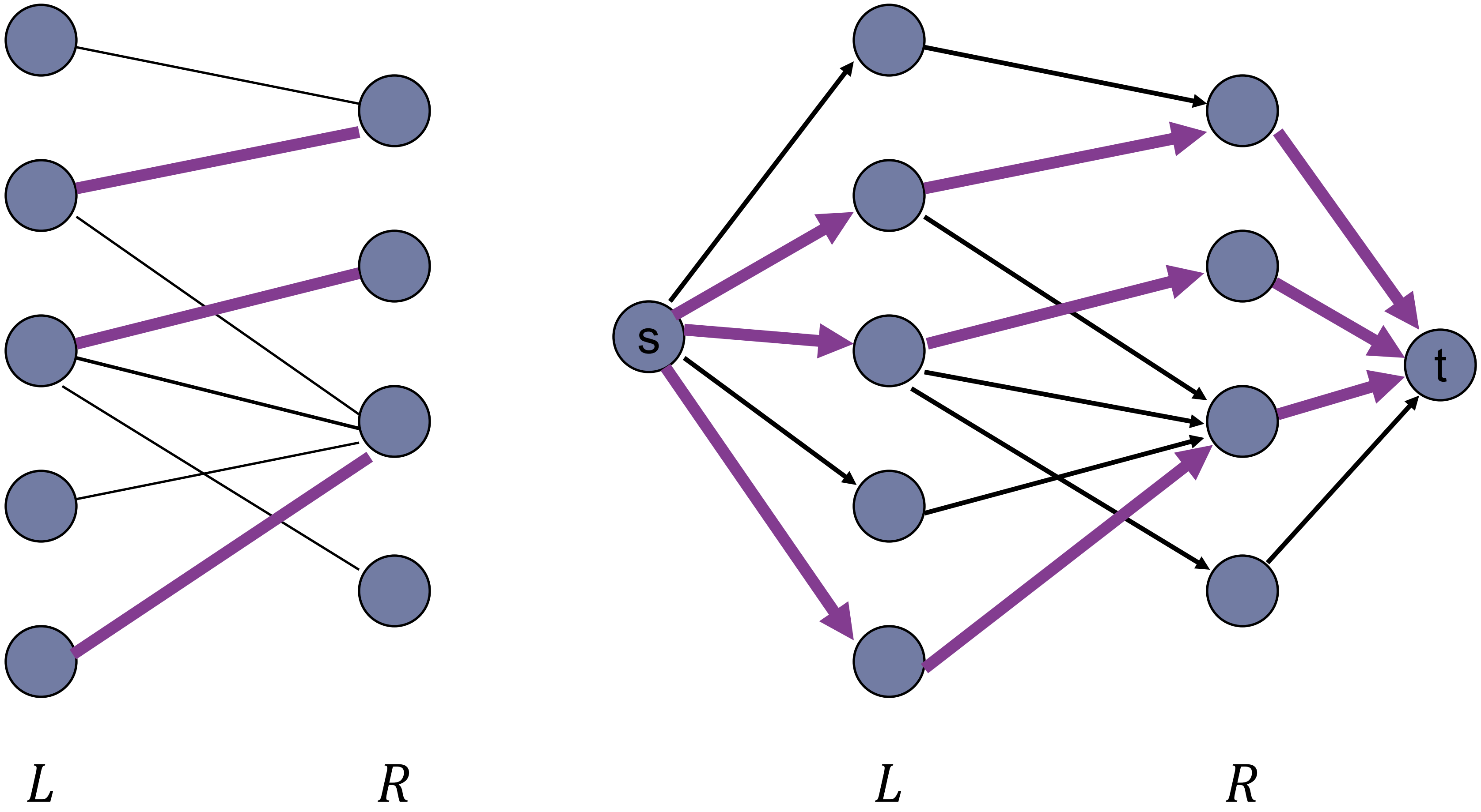
maximum matching

A maximum matching is maximal, but there are maximal matching of smaller size. Computing the smallest maximal matching is difficult!

Encoding by maximum flow



Encoding by maximum flow



Encoding of a bipartite graph by a directed graph. Maximum matching and corresponding maximal flow. Each edge has capacity 1.

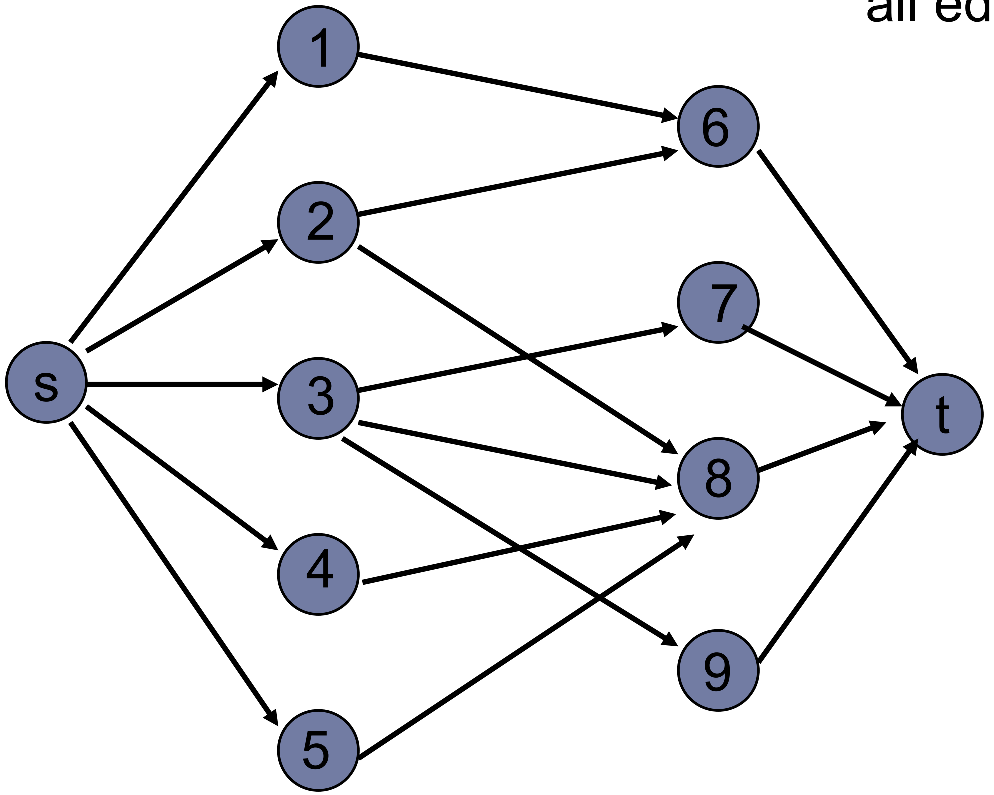
Correctness: Let $G = (V = L \uplus R, E)$ be a bipartite graph and G' be the corresponding directed graph. If C is a matching of G, then there exists a flow in G' of value $|C|$. Conversely, if f is a flow in G' (of an integer value), then there exists a matching in G of cardinality f.

The complexity can be shown to be $O(n \cdot m).$

Improvements have been proposed: for example, the Hopcroft-Karp algorithm works in time $O(\sqrt{n} \cdot m)$

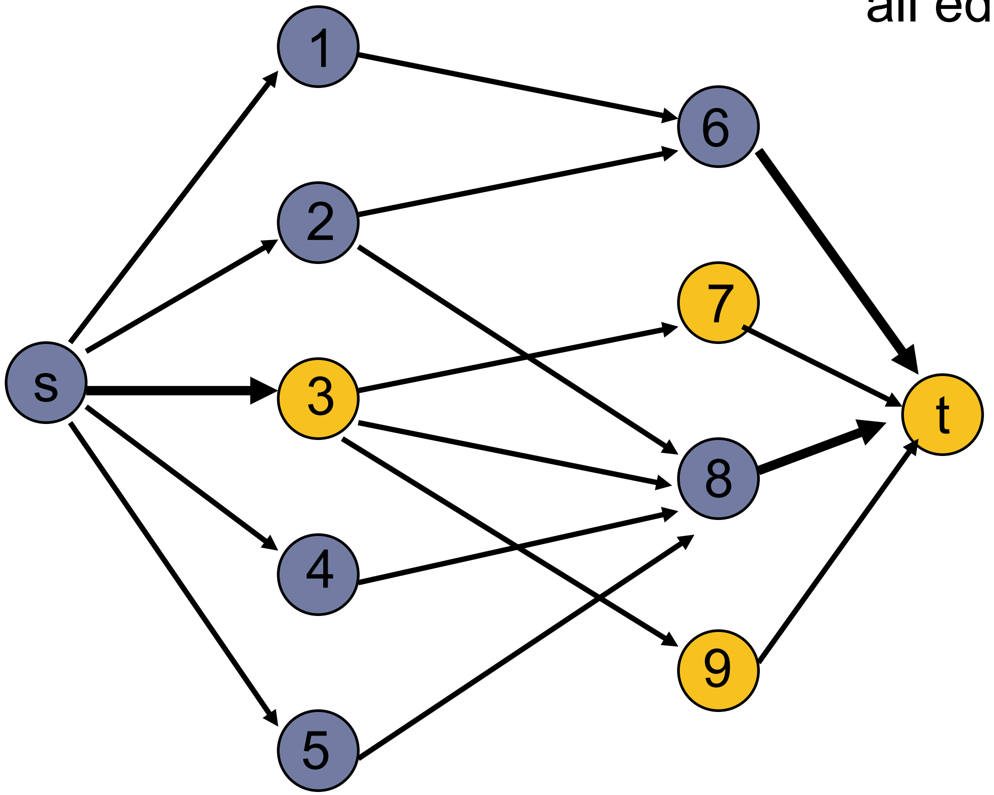
What about min cut here?

all edges have capacity 1



Question: we know that max flow is 3, can you find a min cut with capacity 3?

What about min cut here?

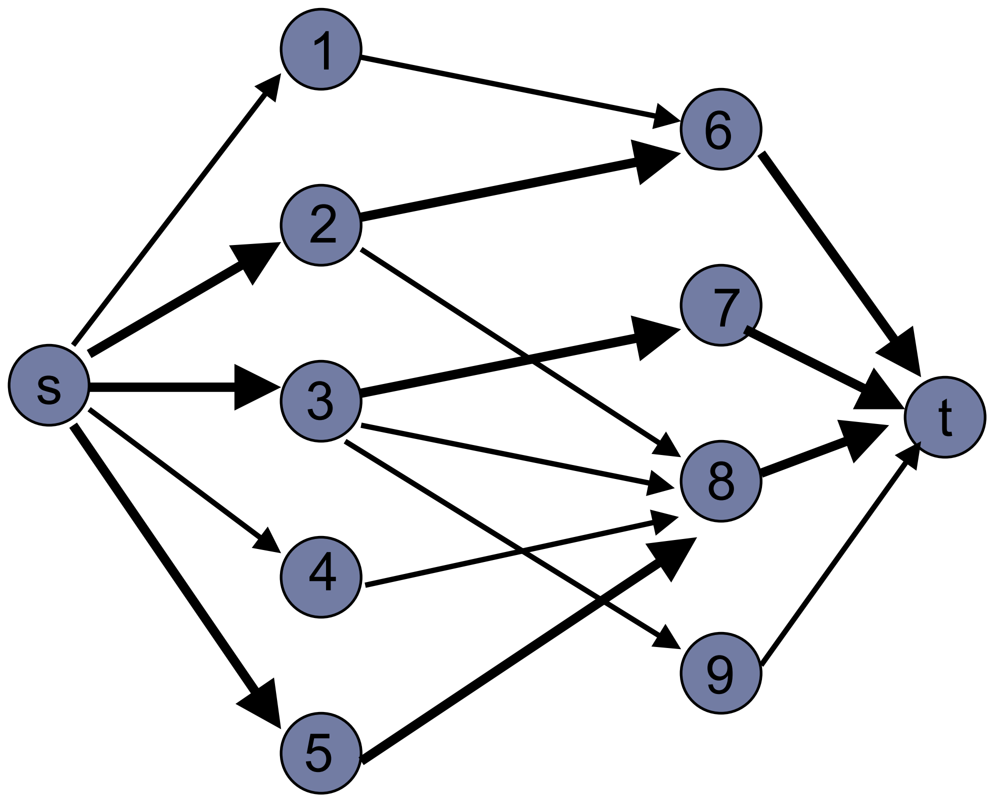


all edges have capacity 1

Question: we know that max flow is 3, can you find a min cut with capacity 3?

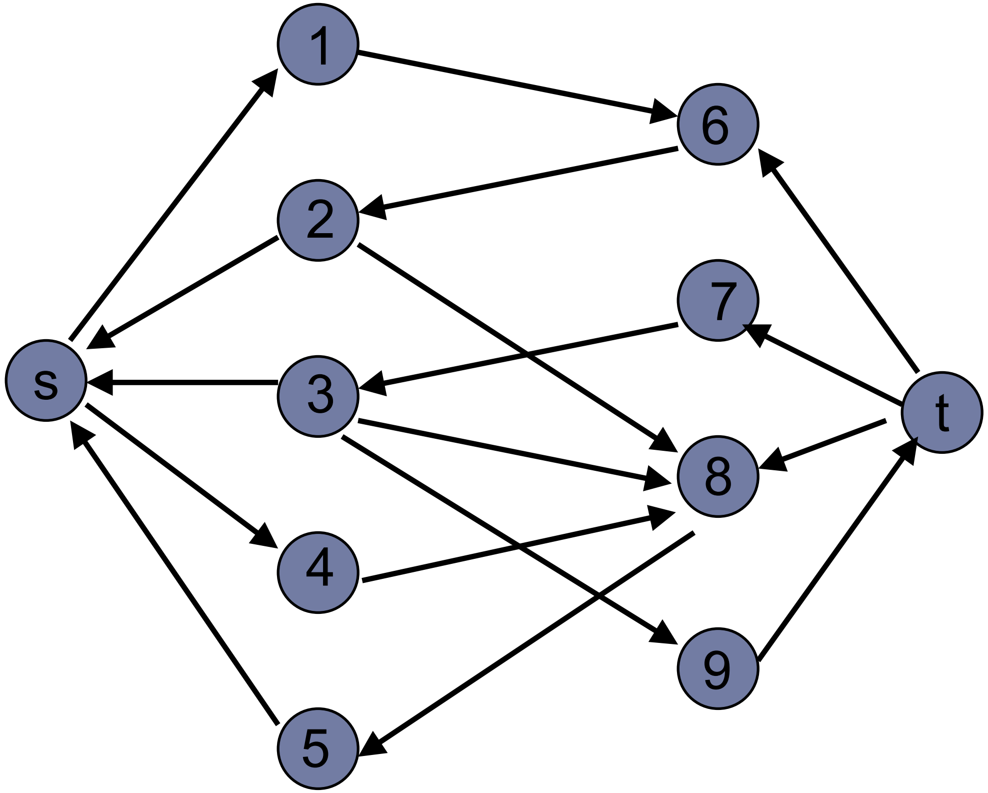
How to obtain min-cut from maxflow

Consider the residual network and compute all nodes accessible from *s*



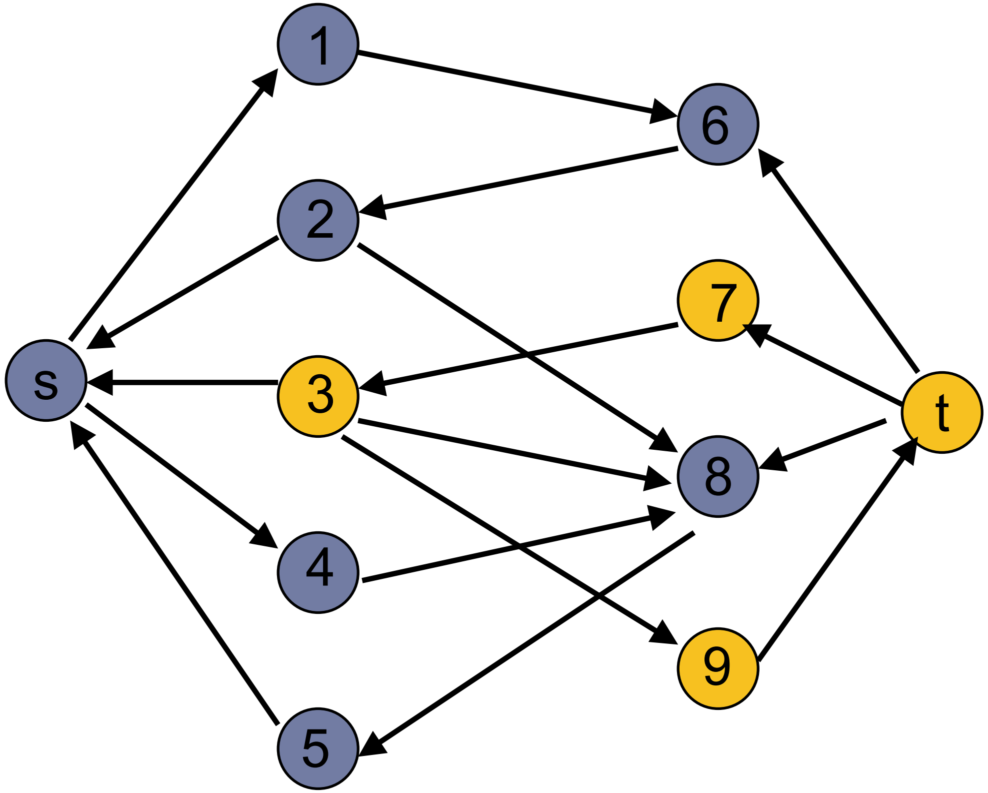
How to obtain min-cut from maxflow

Consider the residual network and compute all nodes accessible from *s*



How to obtain min-cut from maxflow

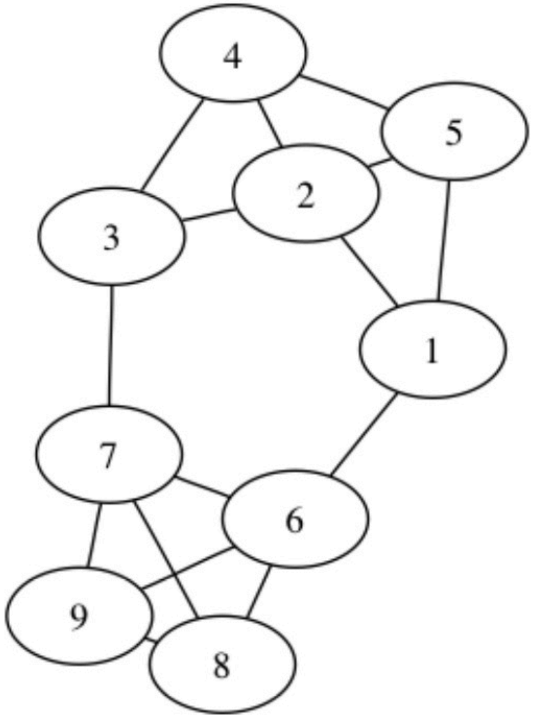
Consider the residual network and compute all nodes accessible from *s*

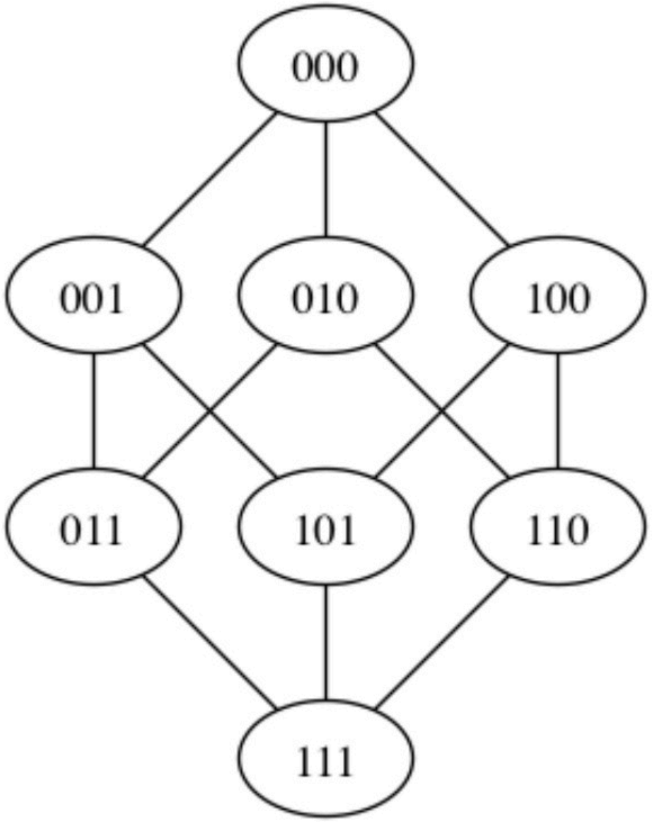


Edge connectivity

Edge connectivity

- Edge connectivity (network reliability) of an *undirected graph* = minimum number of edges that has to be deleted to make the graph disconnected

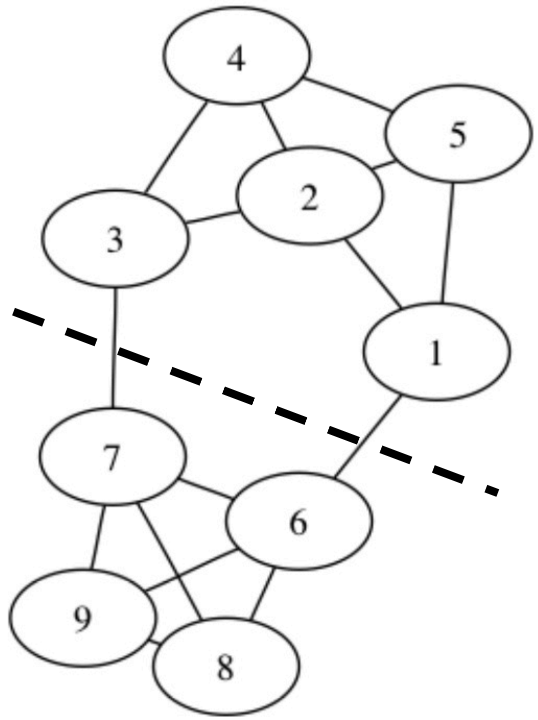




Edge connectivity and cuts

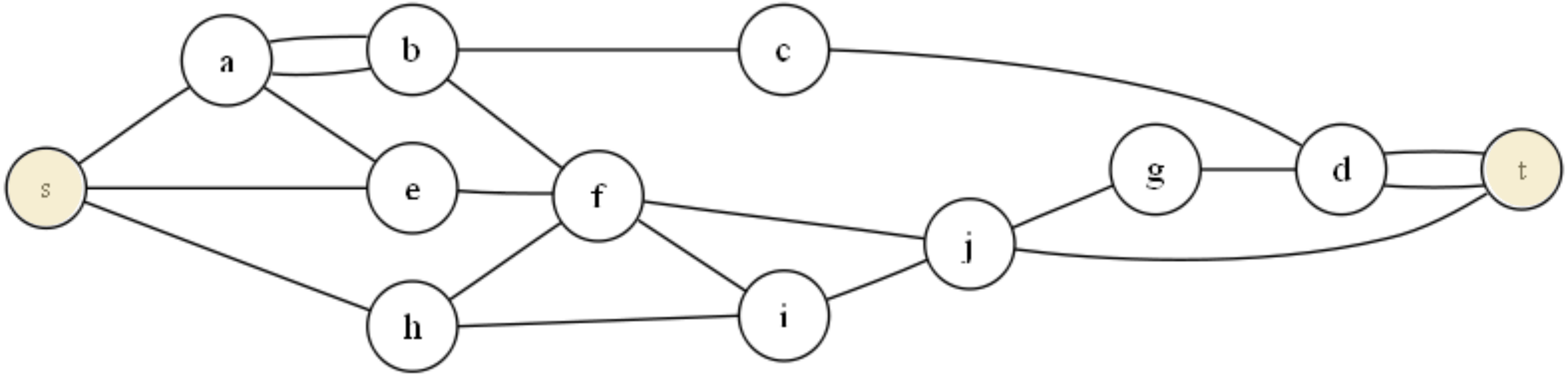
- find a cut such that the number of edges crossing the cut is minimized

- minimum number of edges crossing a cut = edge connectivity



Edge connectivity and max flow

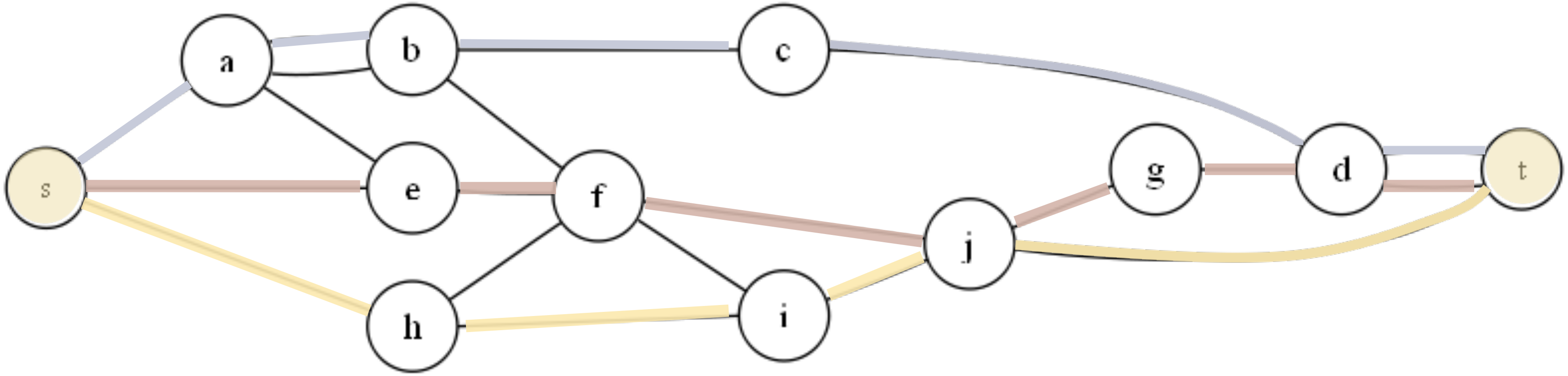
- Assume we know one node on each side of the min cut



- Then we can
  - turn the graph into a weighted directed graph (each edge weighted 1)
  - use Ford-Fulkerson to compute max flow (=edge connectivity)
- *In this case*: max flow = nb of edge-disjoint paths

Edge connectivity and max flow

- Assume we know one node on each side of the min cut



- Then we can
  - turn the graph into a weighted directed graph (each edge weighted 1)
  - use Ford-Fulkerson to compute max flow (=edge connectivity)
- *In this case*: max flow = nb of edge-disjoint paths

Computing edge connectivity

```
turn the graph into directed graph, set all edge capacities to 1
pick any node $v$
for all $u \in V \setminus \{v\}$
    run max-flow algorithm with source $v$ and sink $u$
output the minimum flow obtained
```