Maximum flow in networks

and some other Combinatorial optimization problems

Flow network

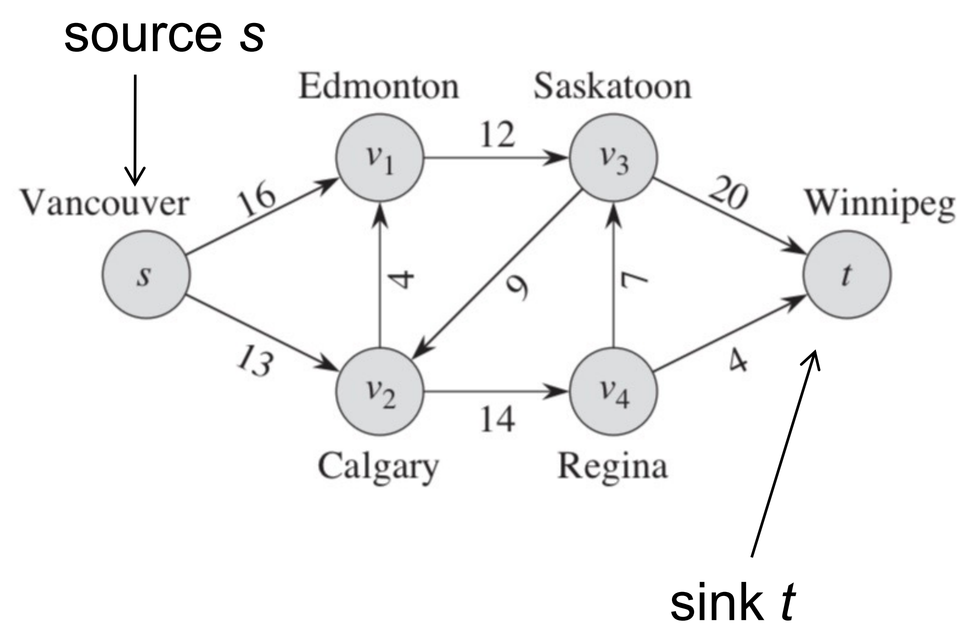
Directed weighted graph $G = (V, E, c)$

$c(p, q) \geq 0$: 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 current
etc.



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

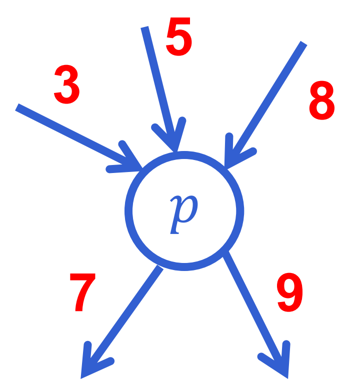
for all $p, q \in V$, $f(p, q) \leq c(p, q)$

Flow network (cont)

Flow conservation

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

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

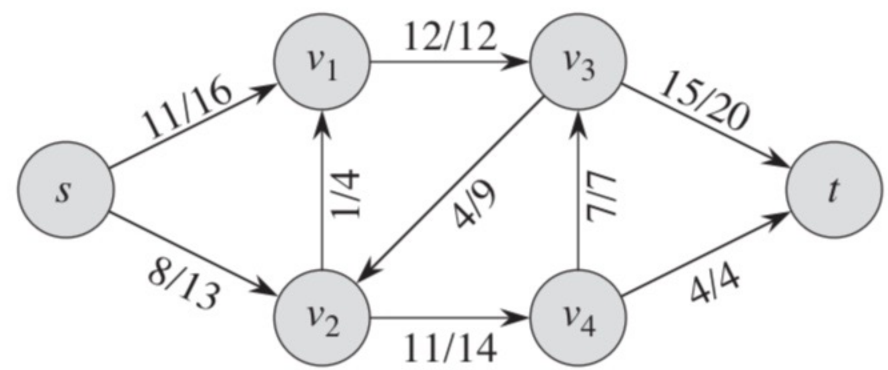


Flow

Flow value (definition):

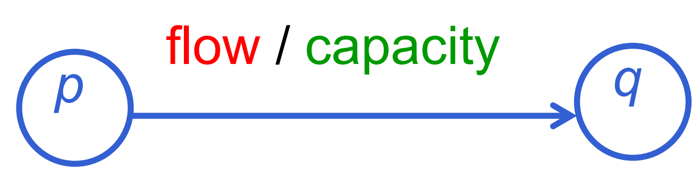
$$|f| = \sum_{p \in V} f(s,p) - \sum_{p \in V} f(p,s)$$

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



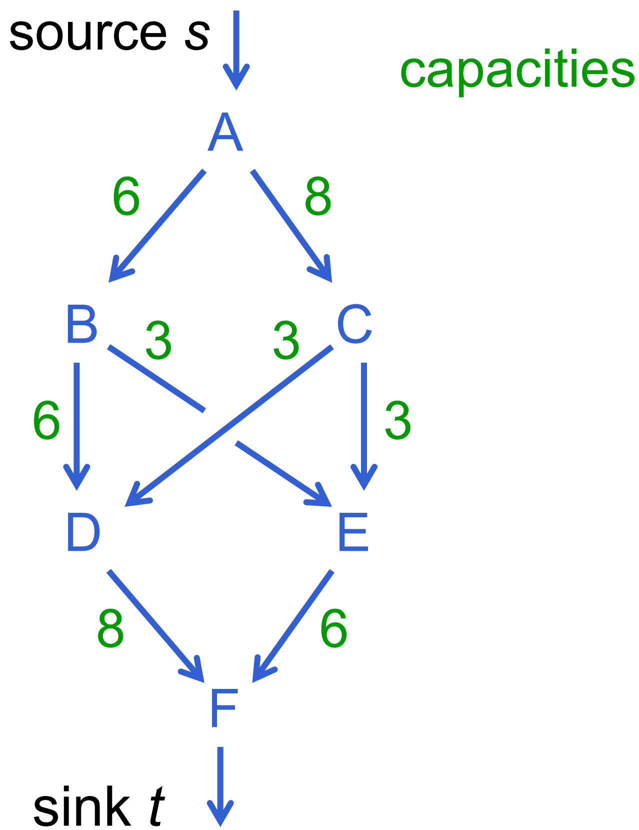
Property:

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



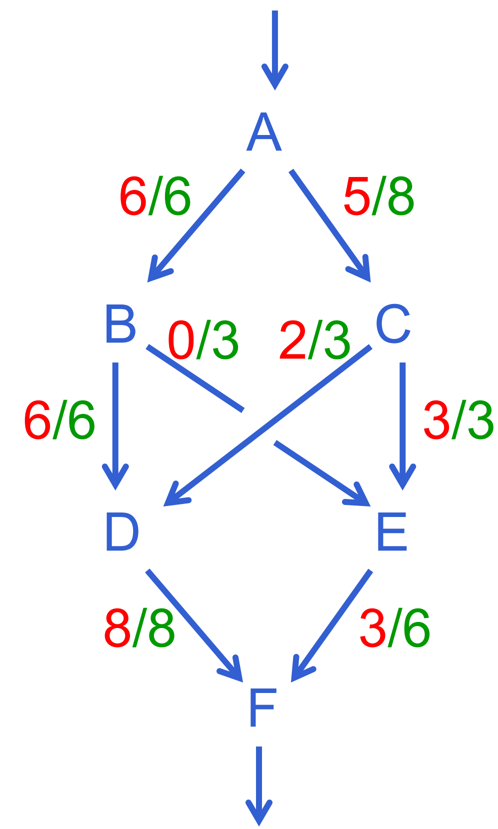
Maximum flow problem

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



maximum flow ?

$|f| = 11$

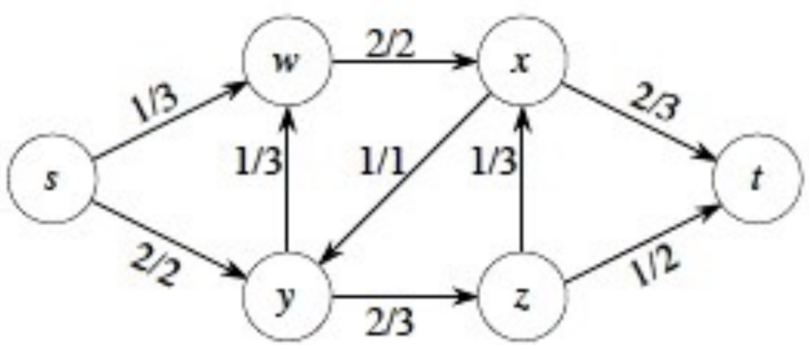


Residual capacity of an edge

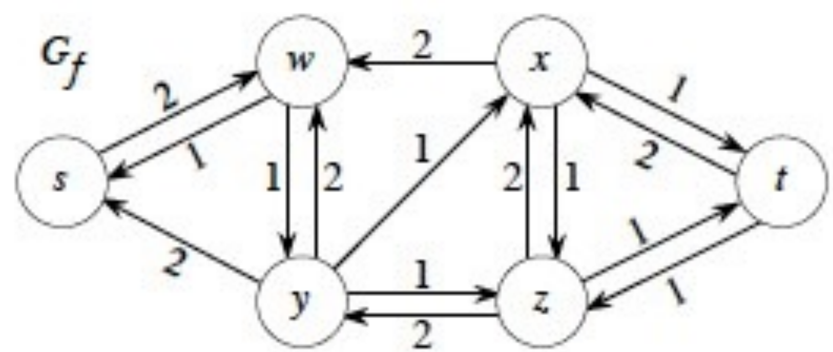
for each edge $(p, q) \in E$ with current flow $f(p, q)$, define

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

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



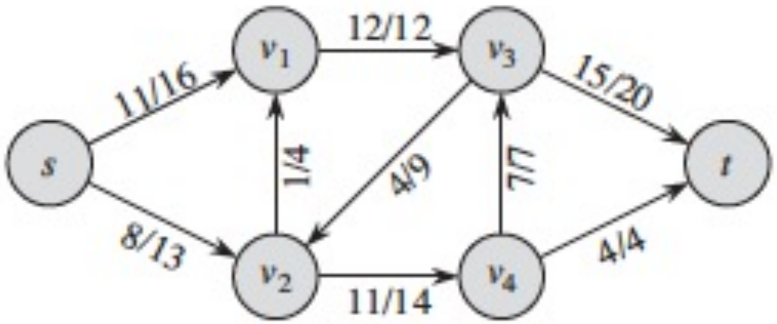
flow network

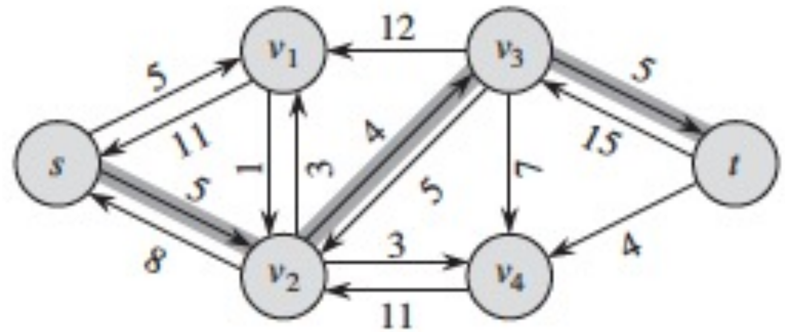


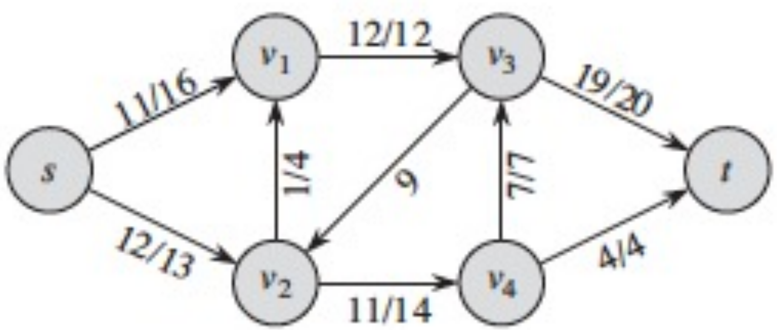
G_f: residual network

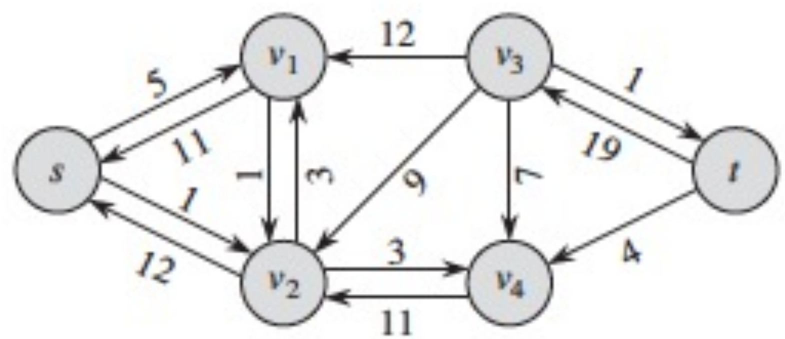
Augmenting path

residual networks









An augmenting path is a simple path in the residual network

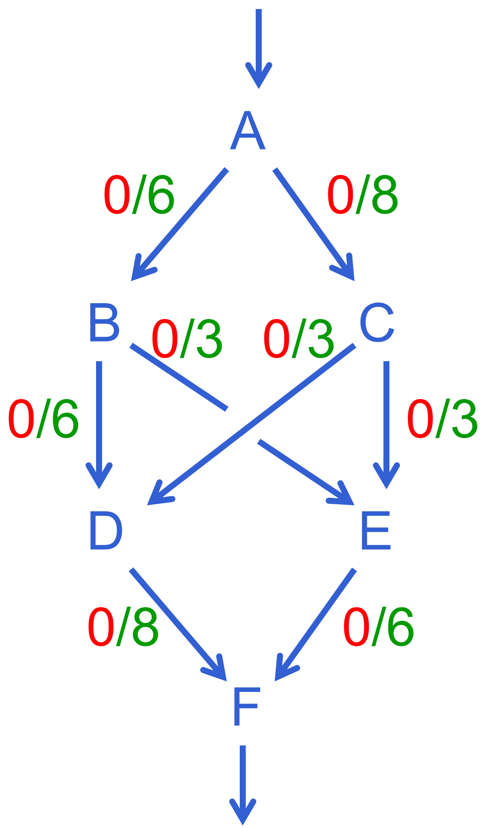
Ford-Fulkerson method (1962)

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

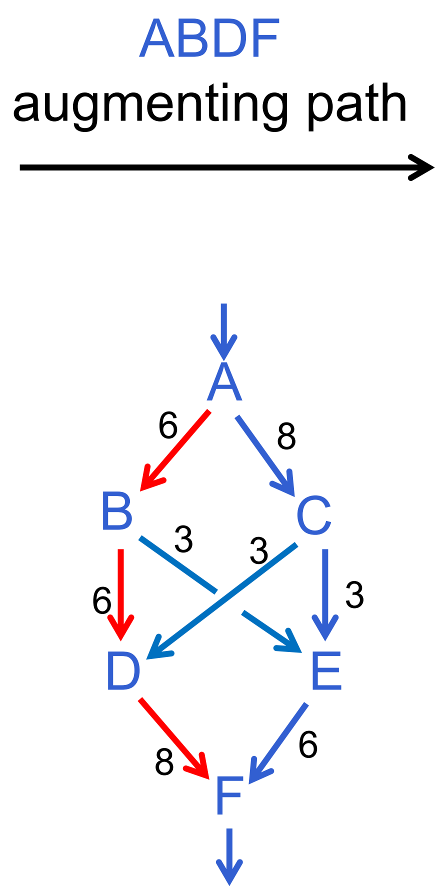
$^{(*)}$ minimum residual capacity of an edge on the path

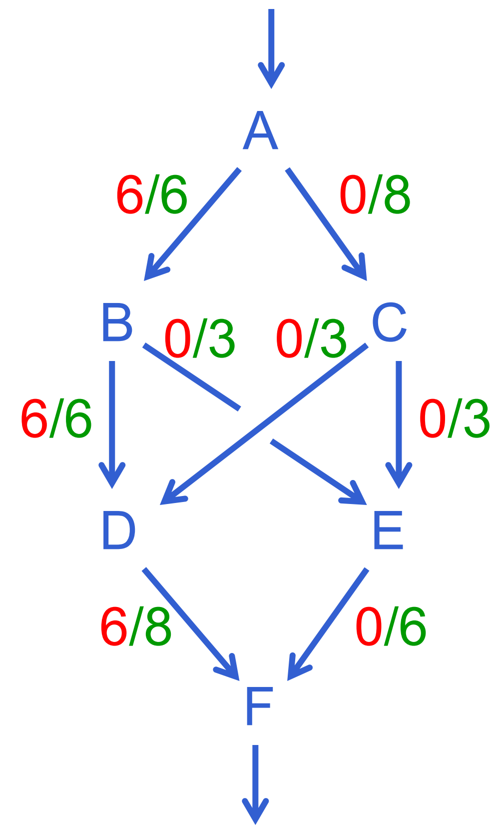
Note: augmenting the flow means incrementing the flow on "forward edges" and decrementing the flow on "backward edges"

Example



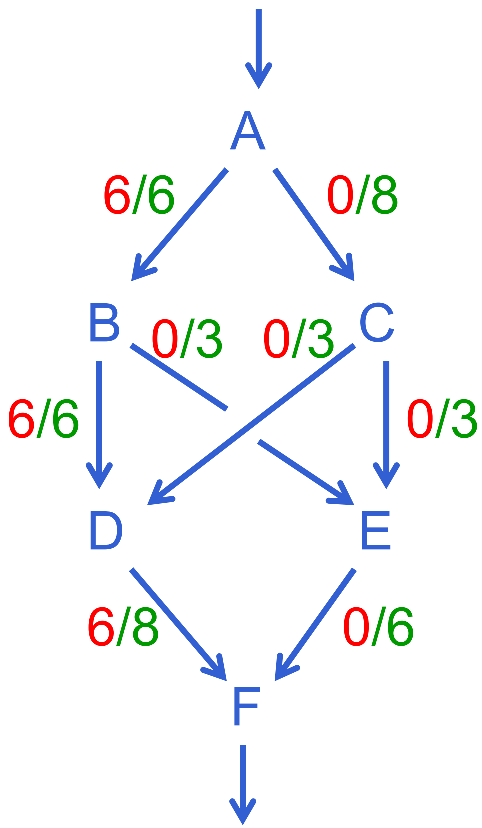
$|f| = 0$





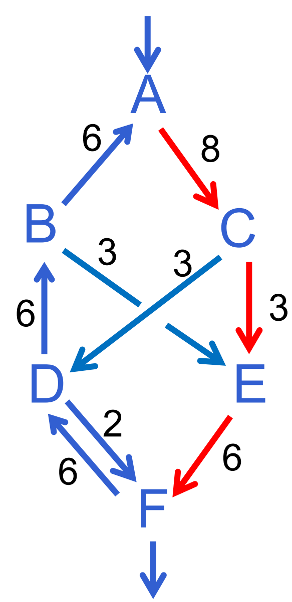
$|f| = 6$

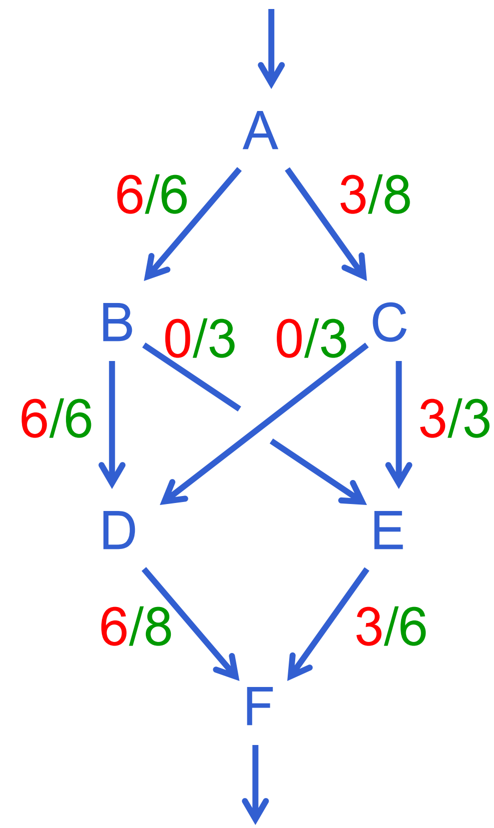
Example (cont)



$|f| = 6$

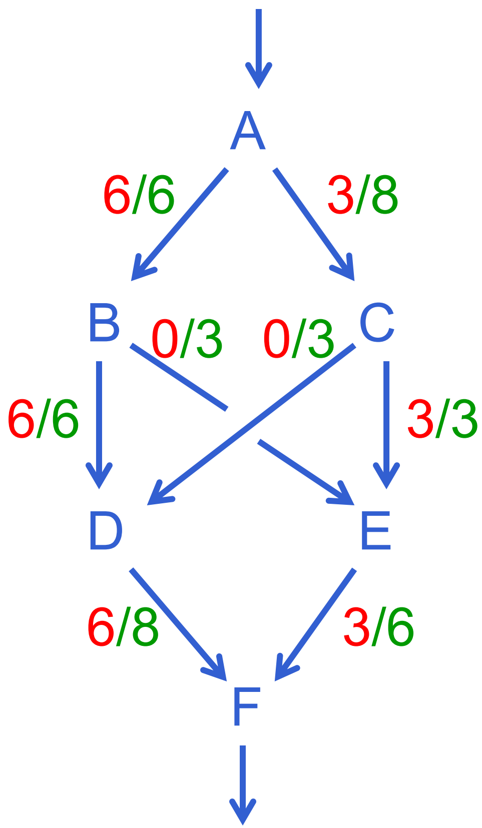
ACEF
augmenting path





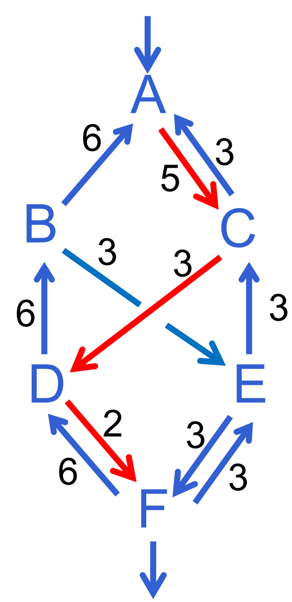
$|f| = 9$

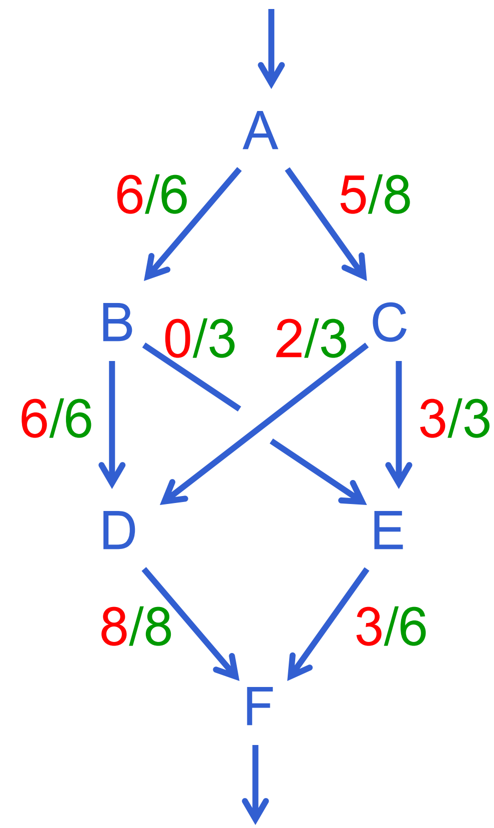
Example (cont)



$|f| = 9$

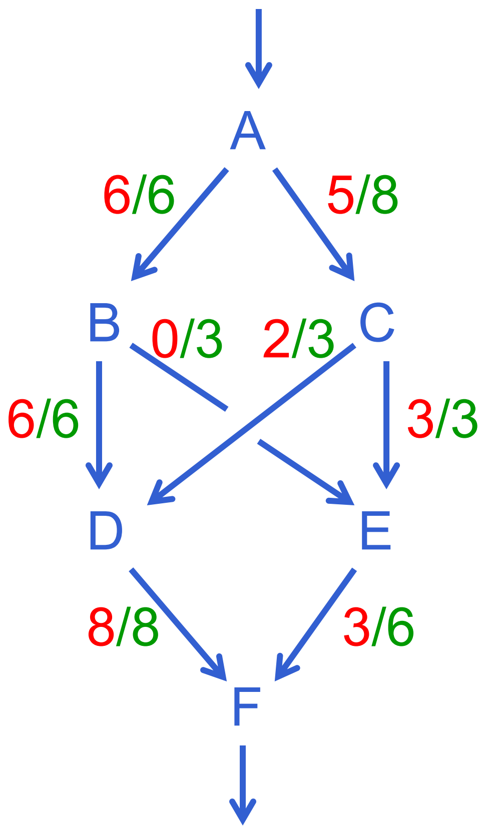
ACDF
augmenting path





$|f| = 11$

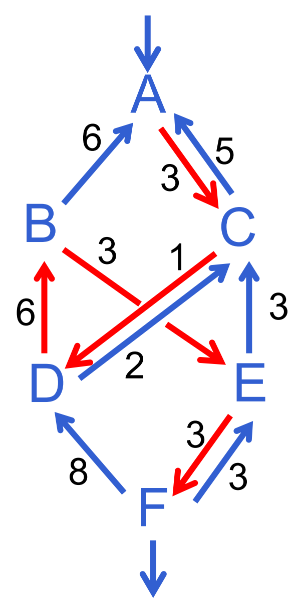
Example (cont)

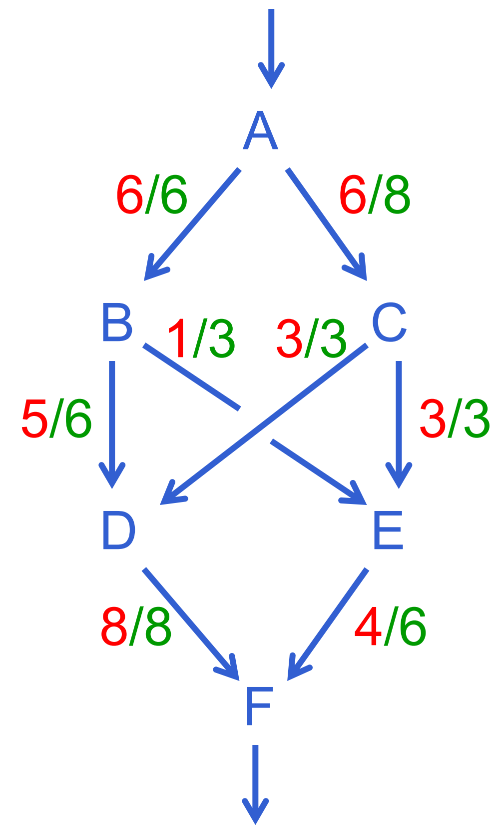


$|f| = 11$

ACDBEF

augmenting path



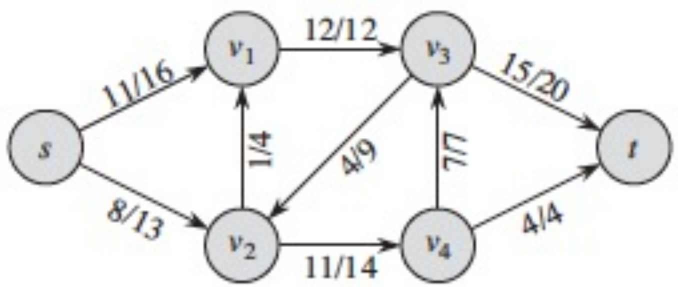


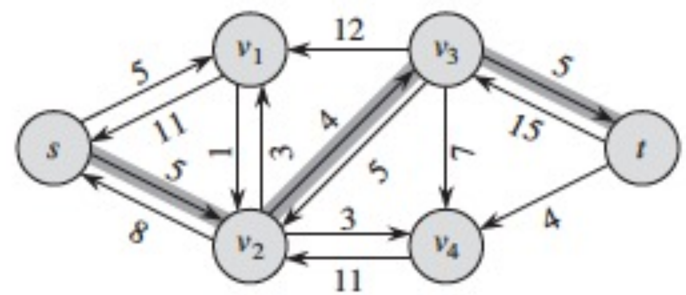
$|f| = 12$

maximum

Max-flow: what we have seen

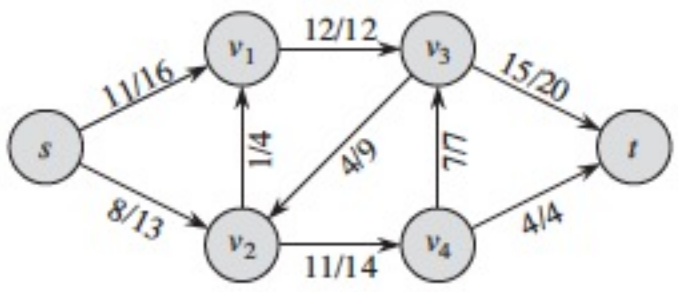
- Flow network, flow, residual network, augmenting path, Ford-Fulkerson method

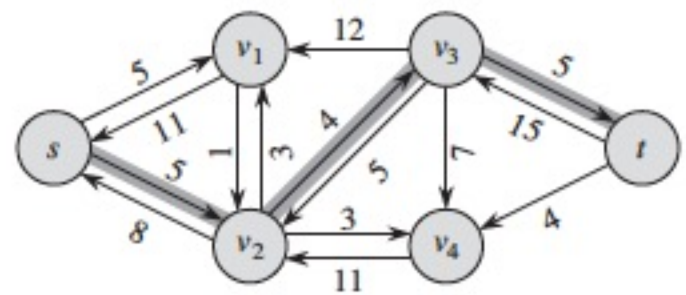




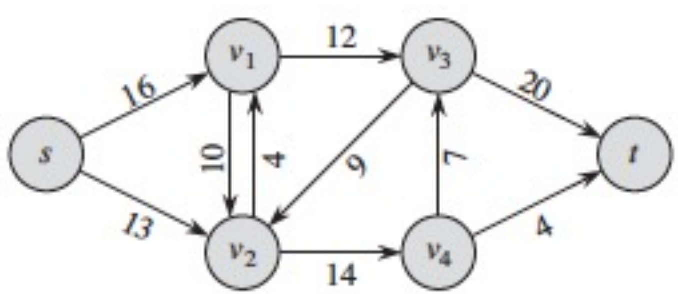
Max-flow: what we have seen

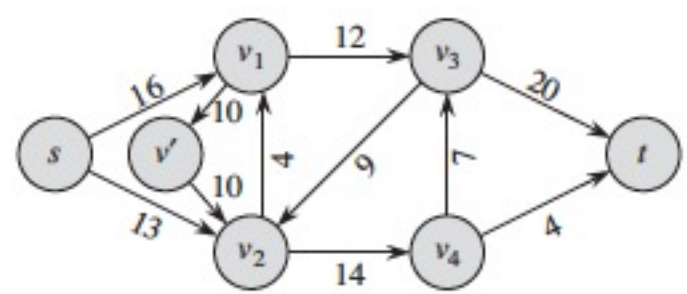
- Flow network, flow, residual network, augmenting path, Ford-Fulkerson method





- antiparallel edges (*technical remark*)





Cut

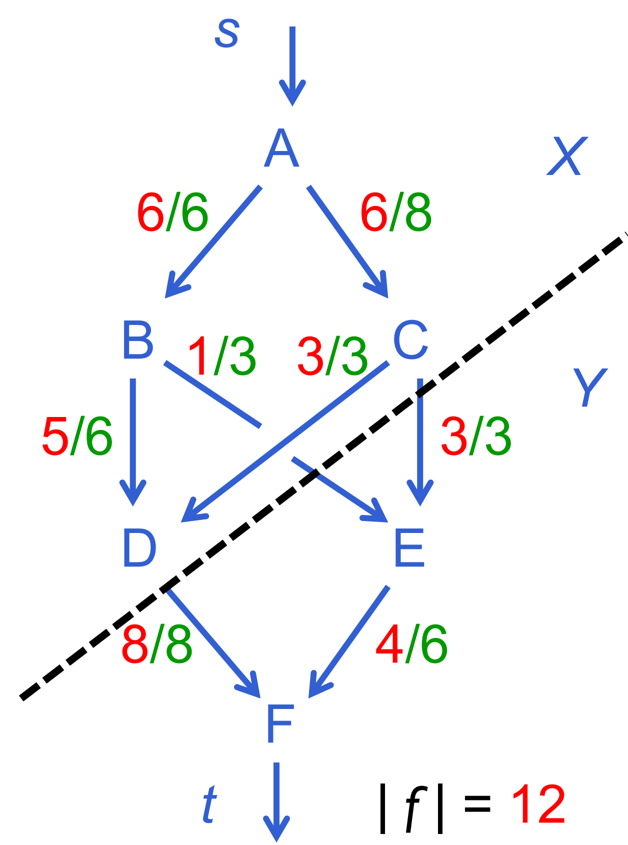
Cut (definition)**:**
(X, Y) cut of $G = (V, E, c)$:
(X, Y) partition of V such that $s \in X, t \in Y$

Capacity of the cut:

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

Flow through the cut:

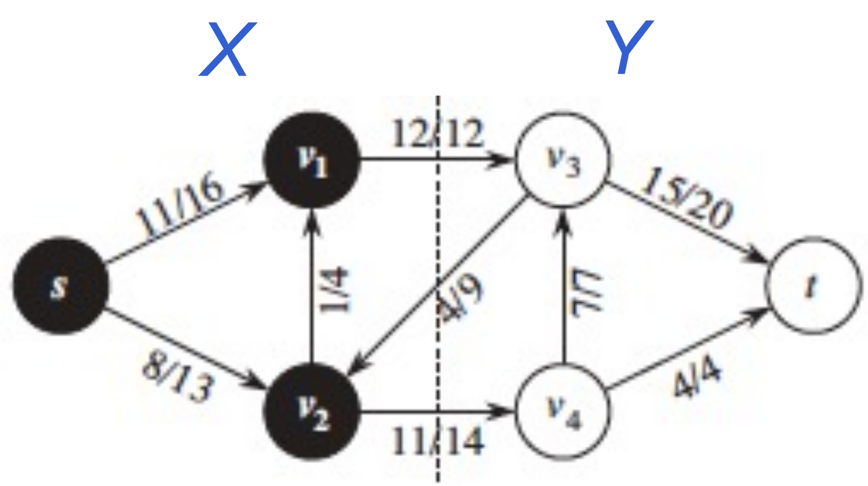
$$f(X, Y) = \sum\{f(x, y) | x \in X, y \in Y\} - \sum\{f(y, x) | x \in X, y \in Y\}$$



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



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

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')

⇒ maximum flow equals minimum cut capacity

Minimum cut

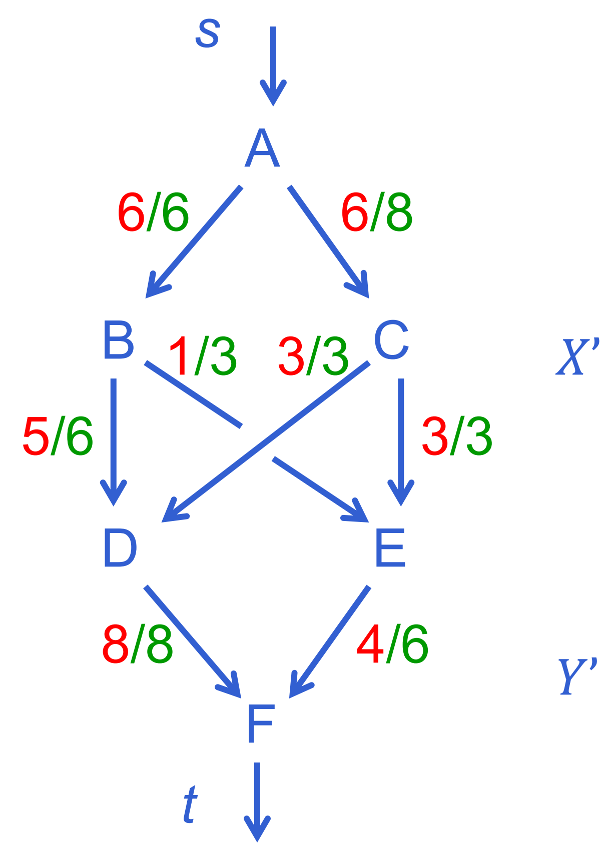
$X' =$

$Y' =$

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

(X', Y') of minimum capacity

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



$|f| = 12$

Minimum cut

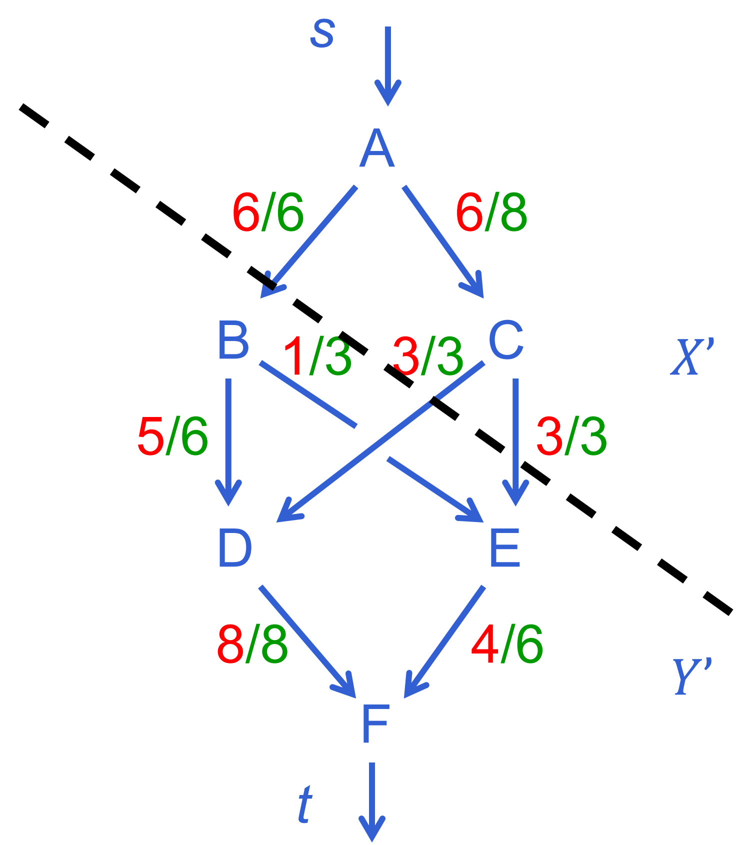
$X' = \{A, C\}$

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

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

(X', Y') of minimum capacity

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



$|f| = 12$

Implementation of Ford-Fulkerson method

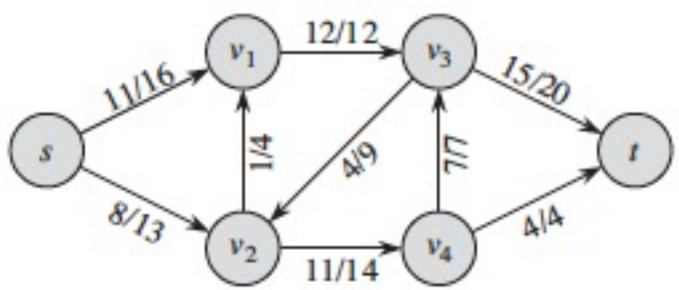
```
initialize flow f to 0 ;
while there exists an augmenting path from s to t do
        augment flow f along this path
return f
```

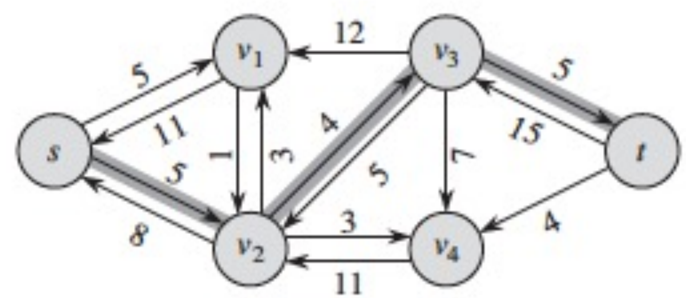
How to choose the augementing path?

Integer-valued flow

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

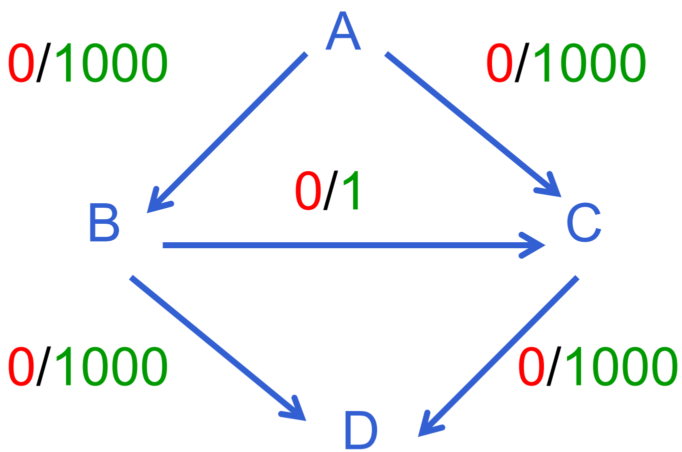
- If C is the max-flow, then Ford-Fulkerson makes at most C iterations $\Rightarrow O(|E| \cdot C)$ time





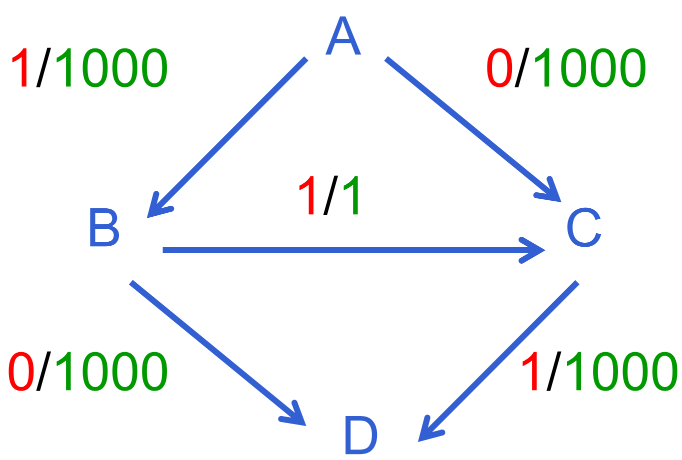
Example

The number of iterations depends on the choice of the paths



Example (cont)

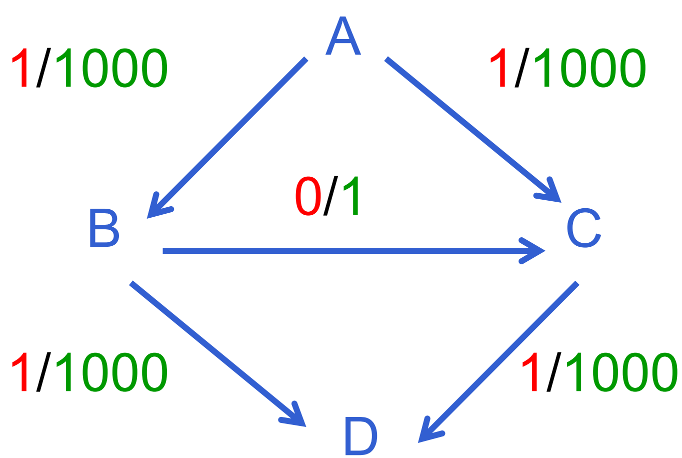
The number of iteration depends on the choice of the paths



augmentation	path
1	ABCD

Example (cont)

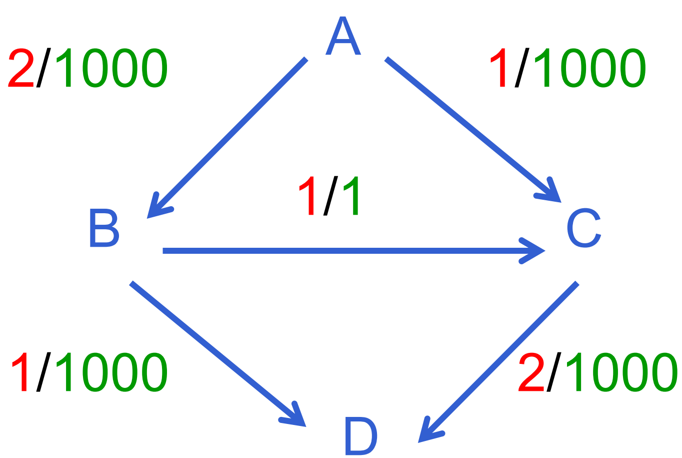
The number of iteration depends on the choice of the paths



augmentation	path
1	ABCD
1	ACBD

Example (cont)

The number of iteration depends on the choice of the paths

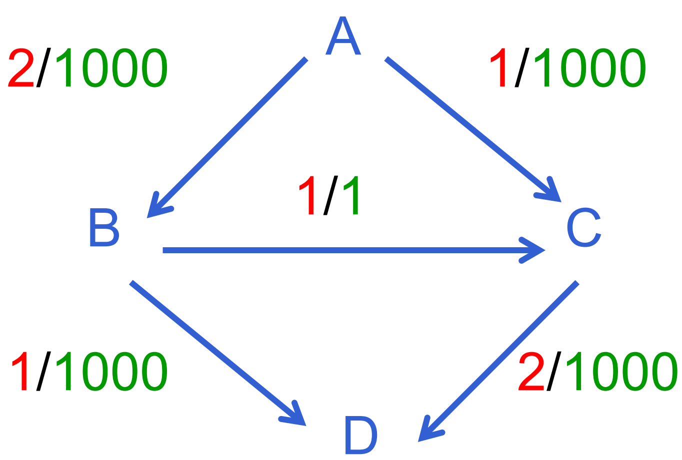


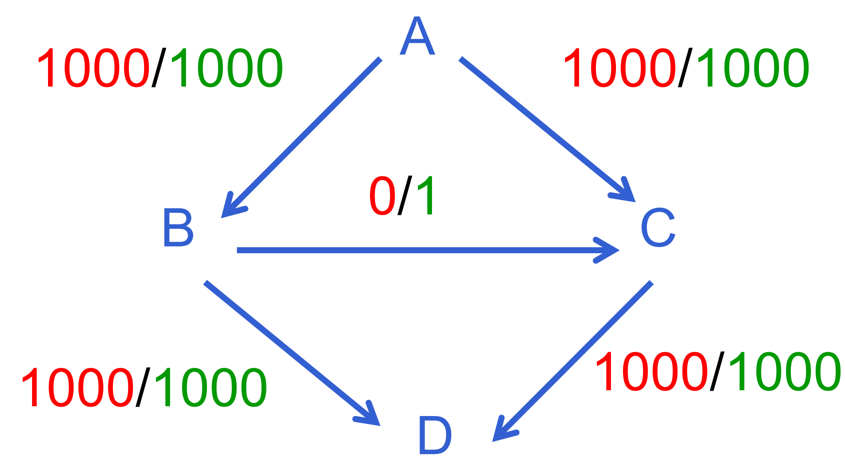
augmentation	path
1	ABCD
1	ACBD
1	ABCD

etc.

Example (cont)

The number of iteration depends on the choice of the paths





augmentation	path	augmentation	path
1	ABCD	1000	ABD
1	ACBD	1000	ACD
1	ABCD	maximum flow	
etc.			

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 $O(n \cdot m^2)$

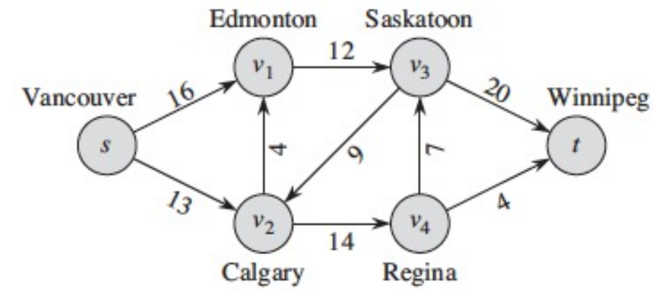
This strategy is known as the *Edmonds-Karp algorithm* (1972), but was discovered by Dinitz (1970)

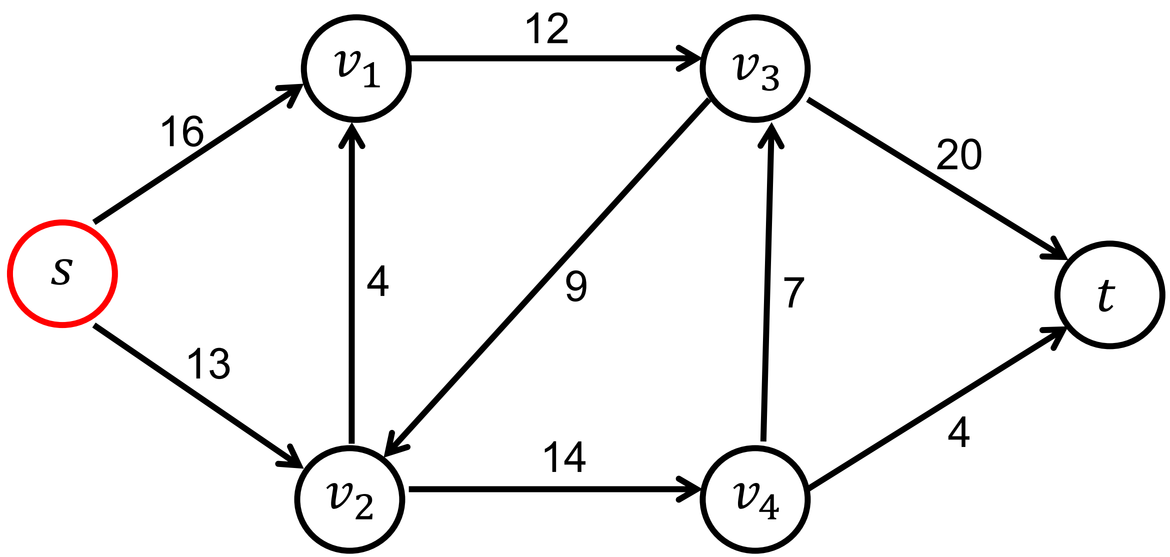
Other strategies

Push-relabel algorithm (Dinitz 70): $O(n^2 \cdot m)$

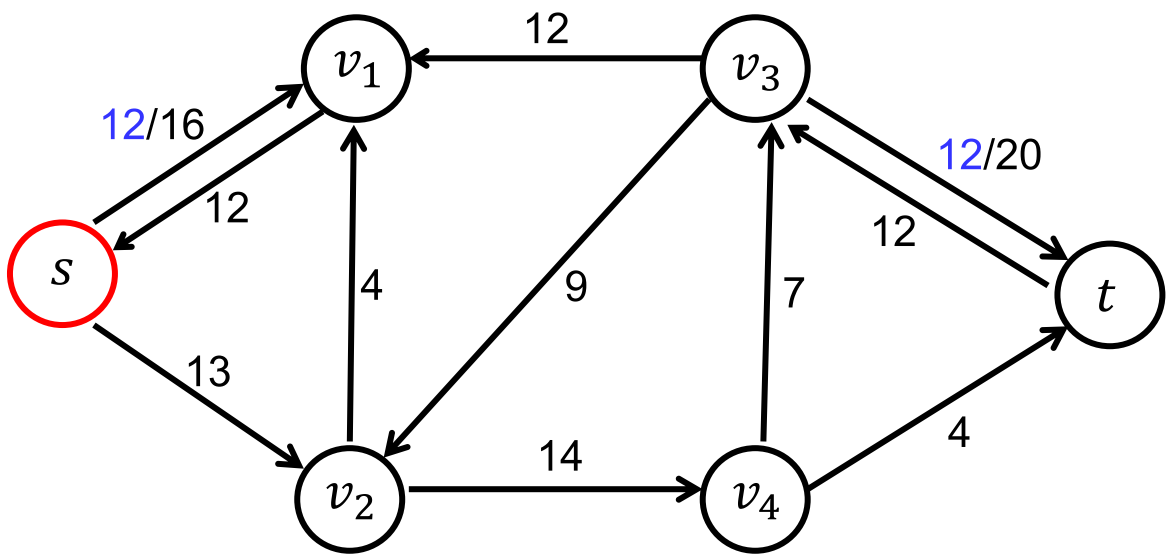
Relabel-to-front algorithm (Karzanov 74): $O(n^3)$

Ford-Fulkerson: example



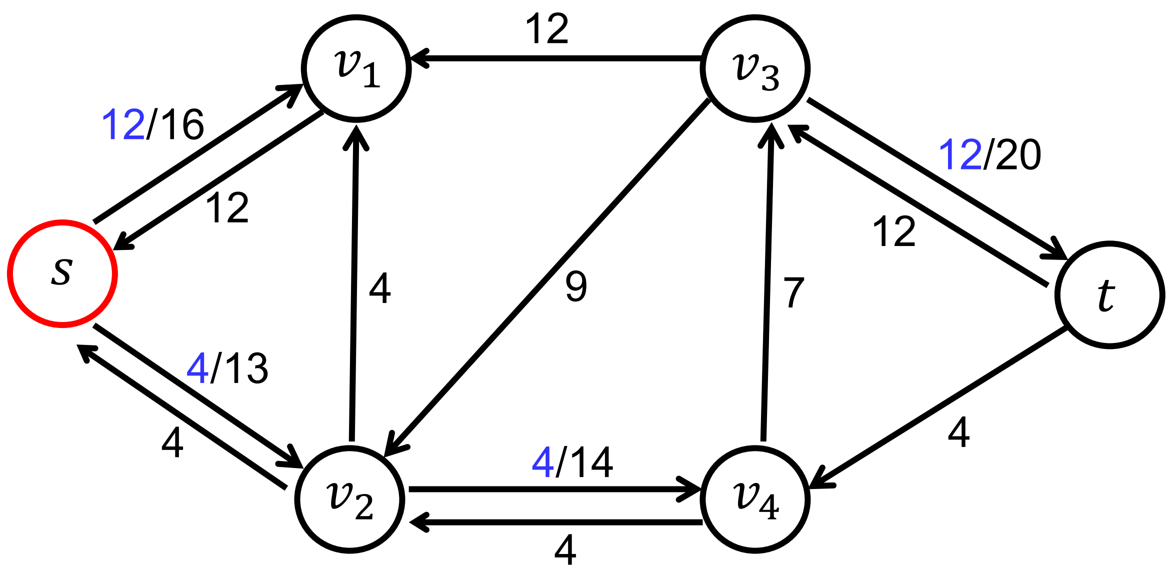


Ford-Fulkerson: example



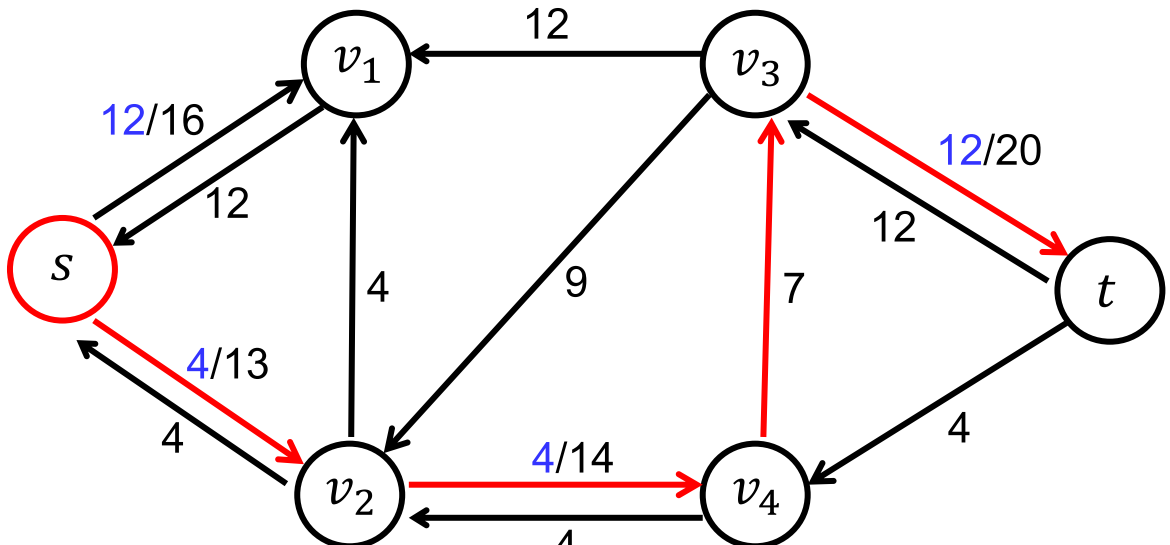
$$|f| = 12$$

Ford-Fulkerson: example



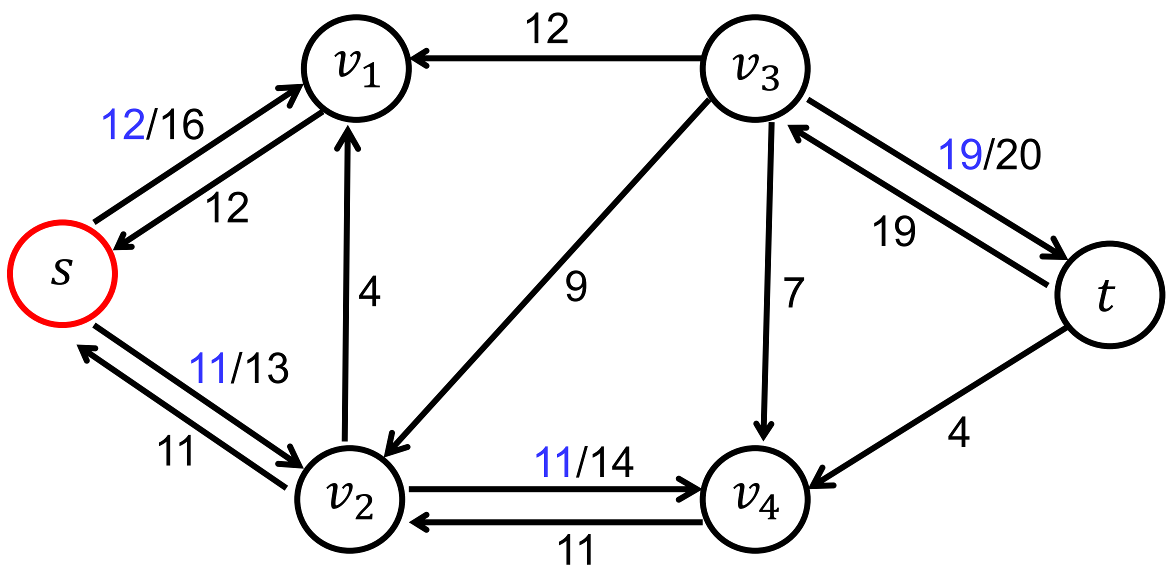
$$|f| = 12 + 4$$

Ford-Fulkerson: example



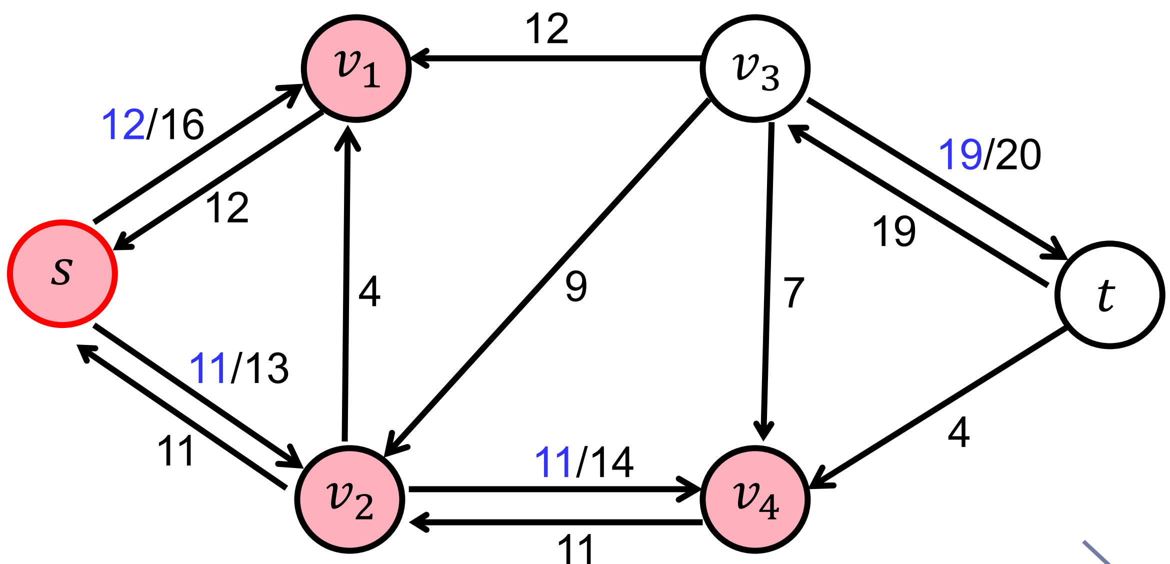
$$|f| = 12 + 4 + 7$$

Ford-Fulkerson: example

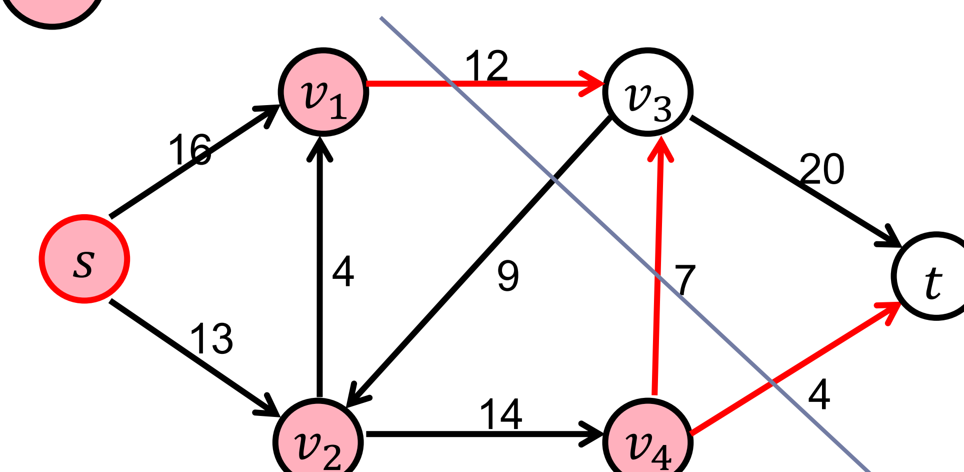


$$|f| = 12 + 4 + 7 = 23$$

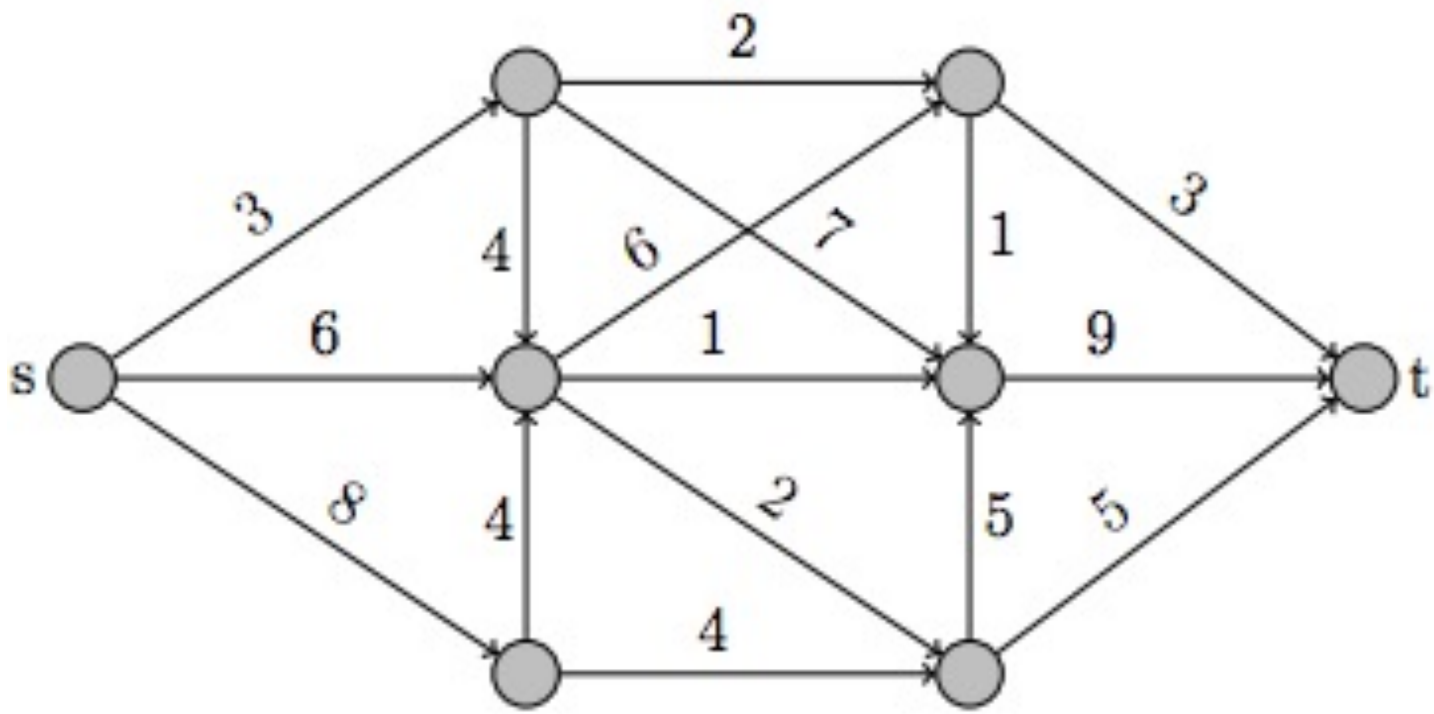
Ford-Fulkerson: example



$$|f| = 12 + 4 + 7 = 23$$

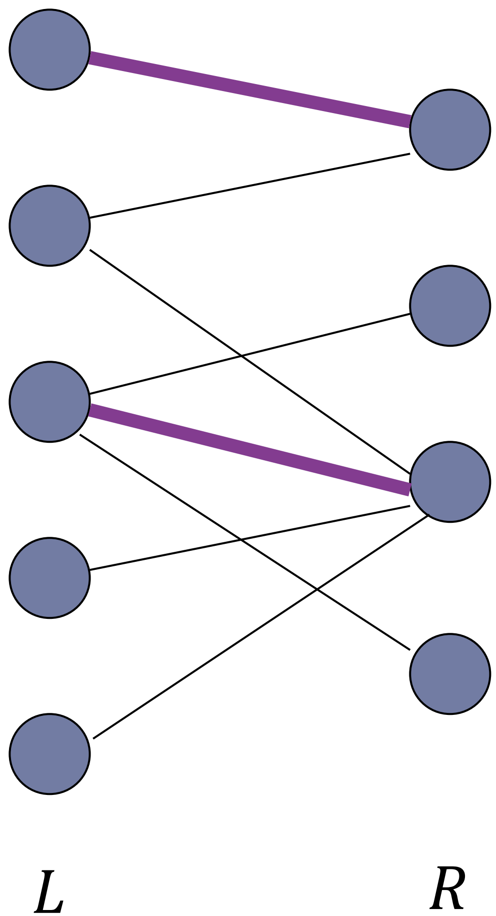


Quiz 3.2



Maximum bipartite matching

Maximum matching



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

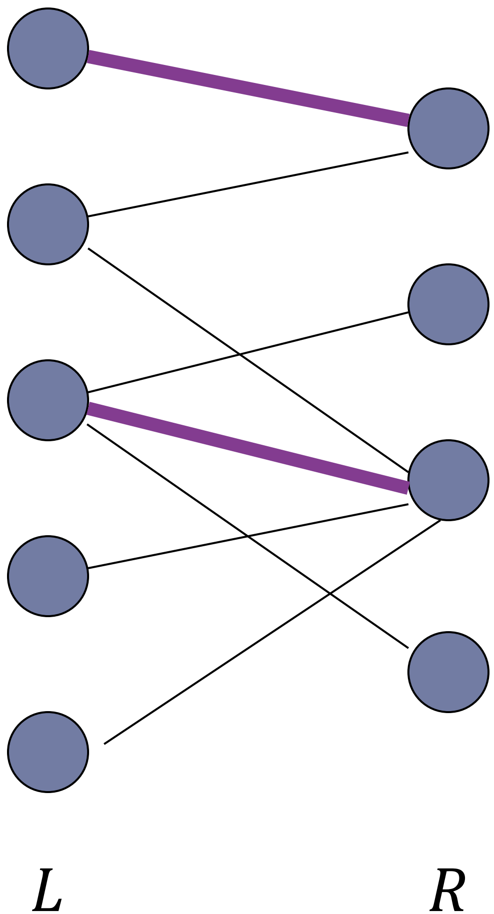
$$\forall (p, q) \in E, p \in L \ et \ q \in R$$

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)

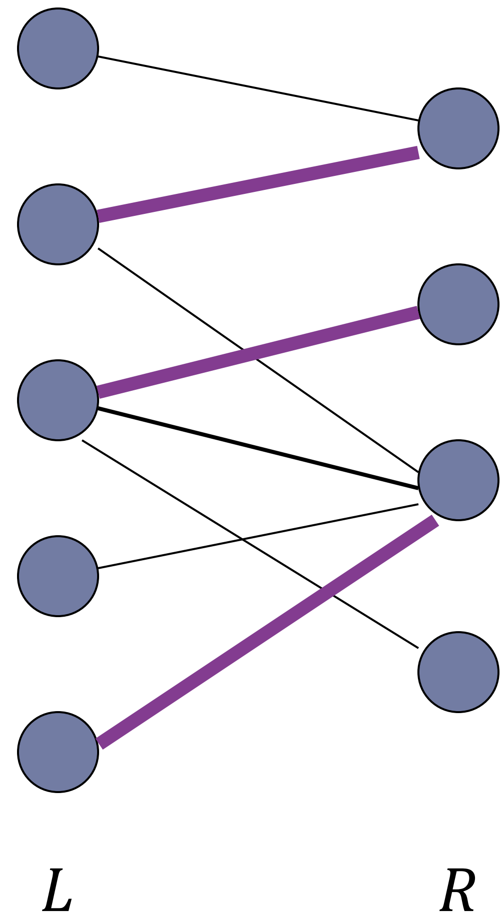
Maximum matching: matching with the maximum number of edges

NB: maximum ≠ maximal (by inclusion!)

Maximum vs maximal matching



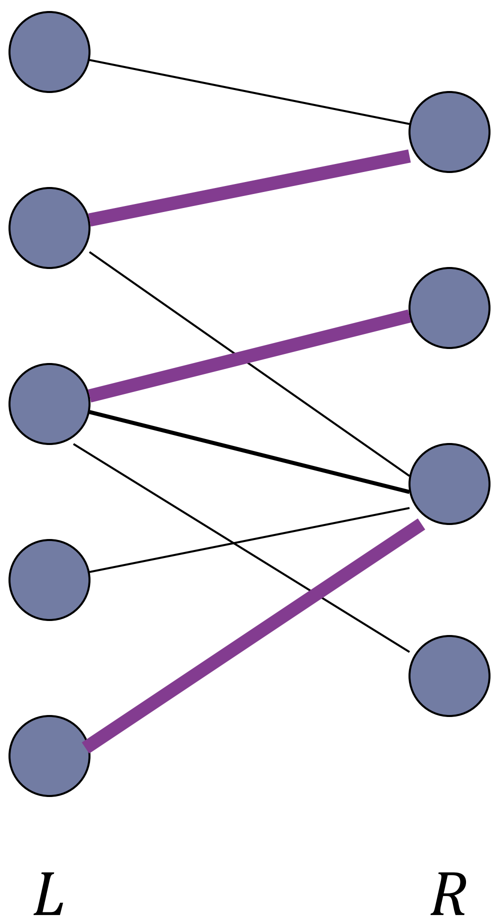
maximal matching



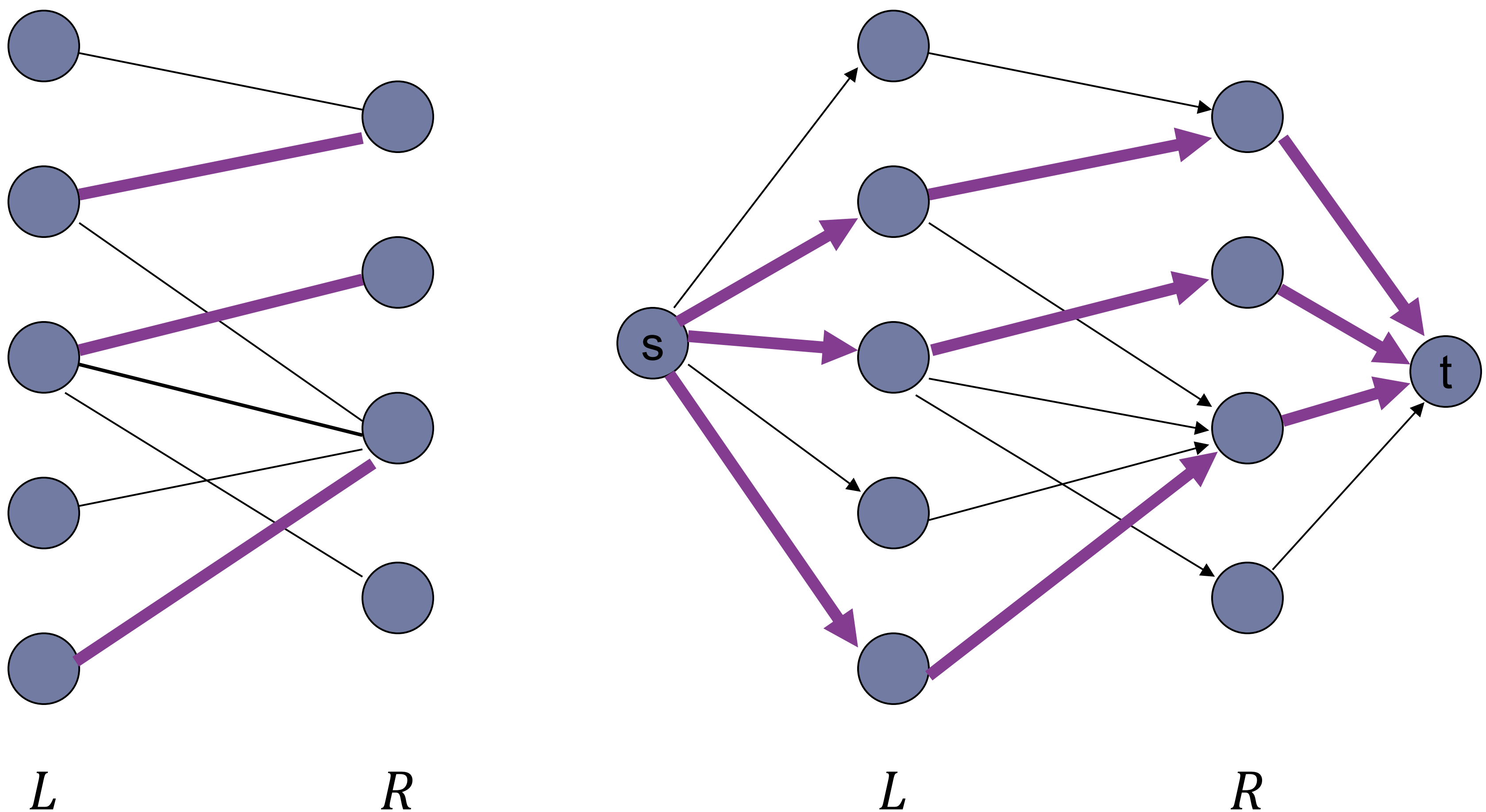
maximum matching

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.

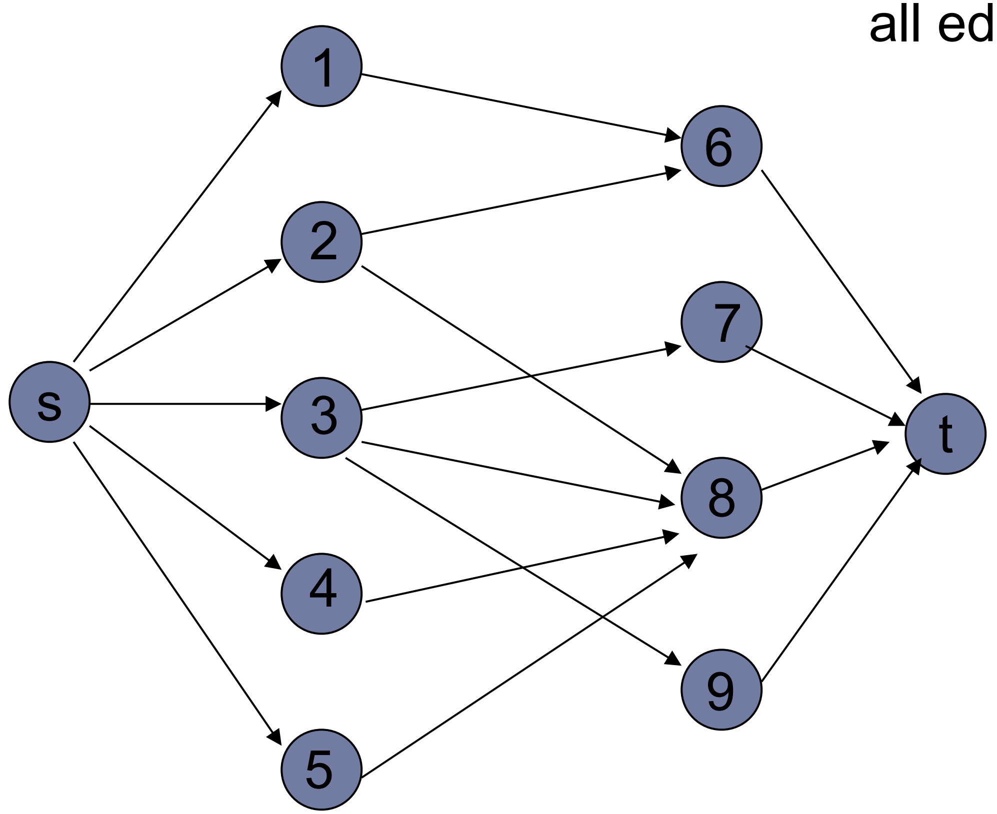
Encoding by maximum flow

- **Correctness**: Let G be a bipartite graph and G' the corresponding flow network. Then the maximum matching has $|f|$ edges, where f is max flow.

- 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?

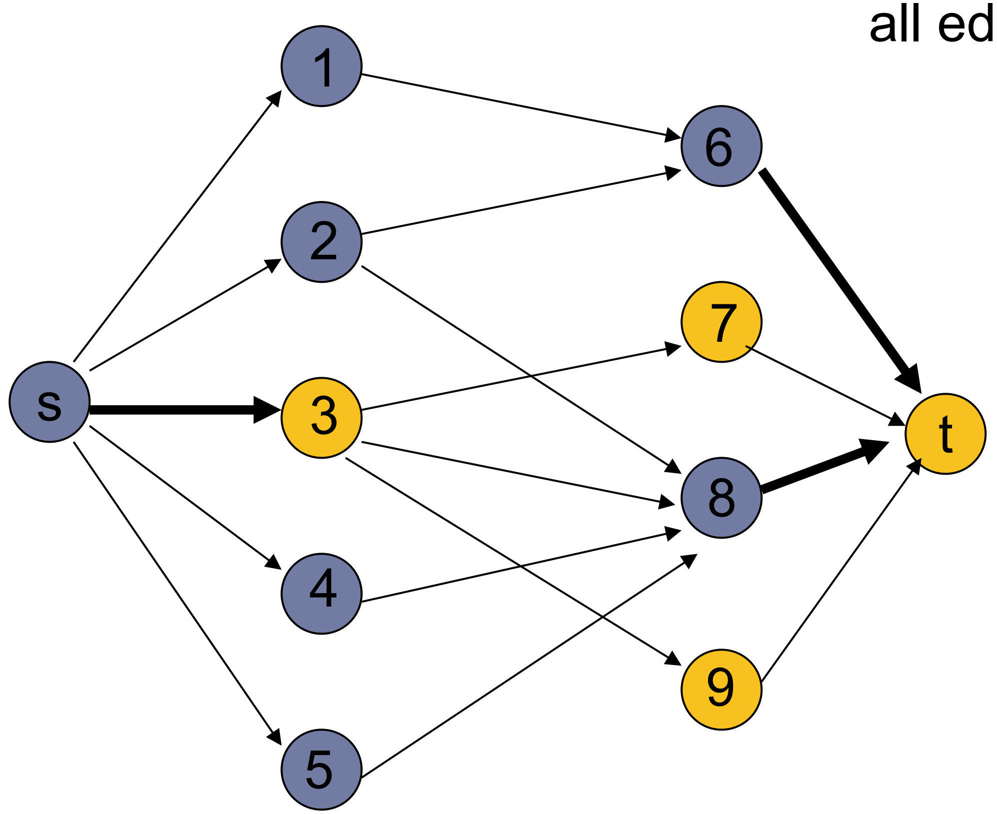
all edges have capacity 1



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

What about min cut here?

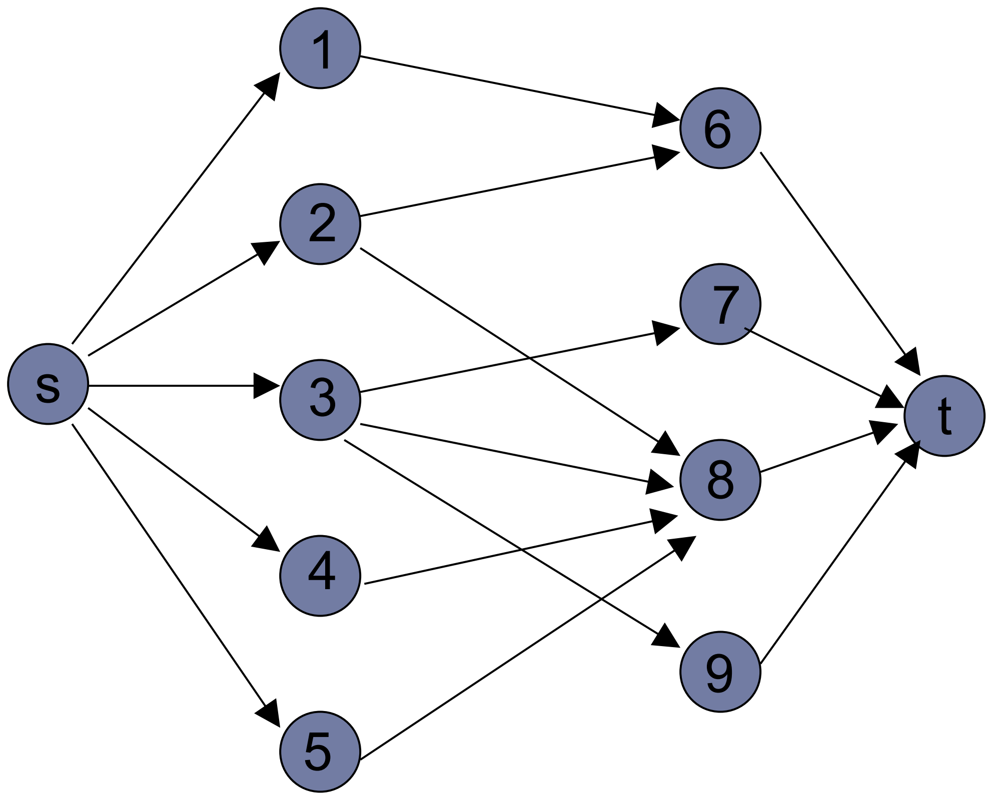
all edges have capacity 1



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

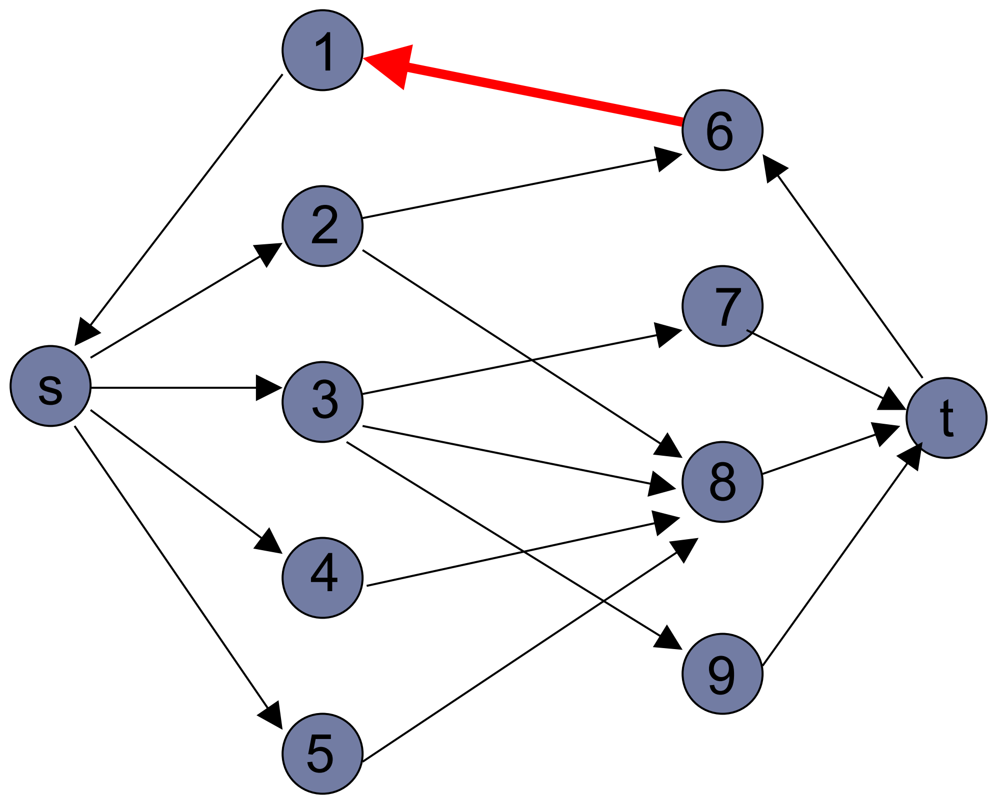
FF on bipartite graphs: a closer look

all edges have capacity 1



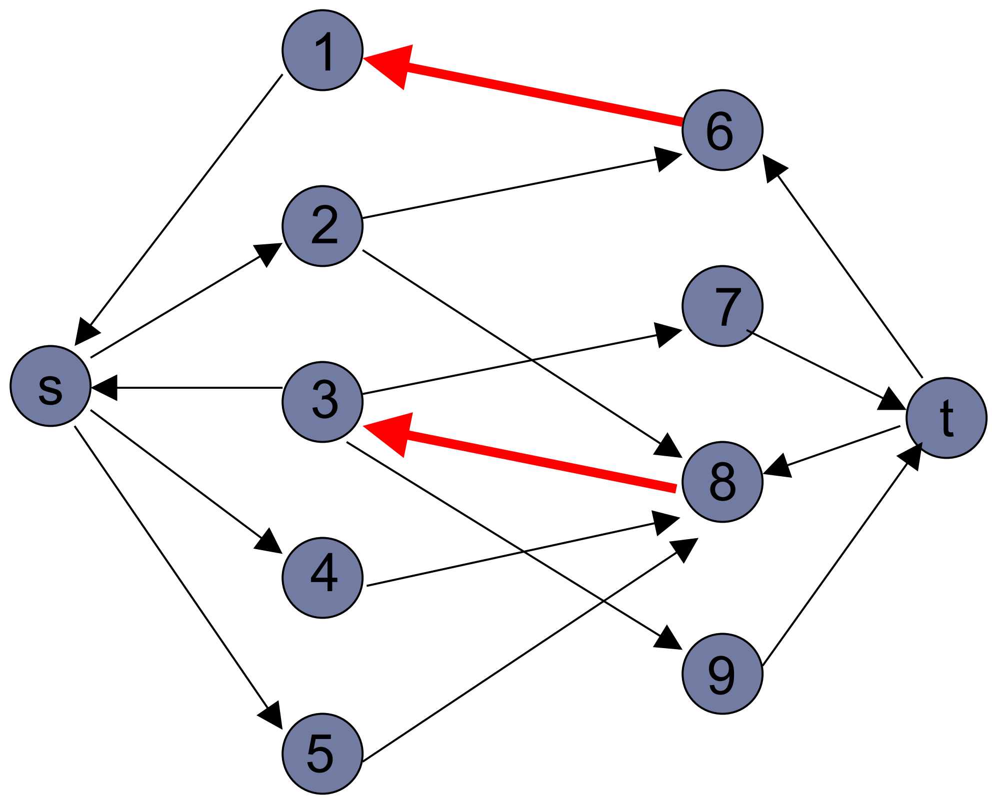
FF on bipartite graphs: a closer look

all edges have capacity 1



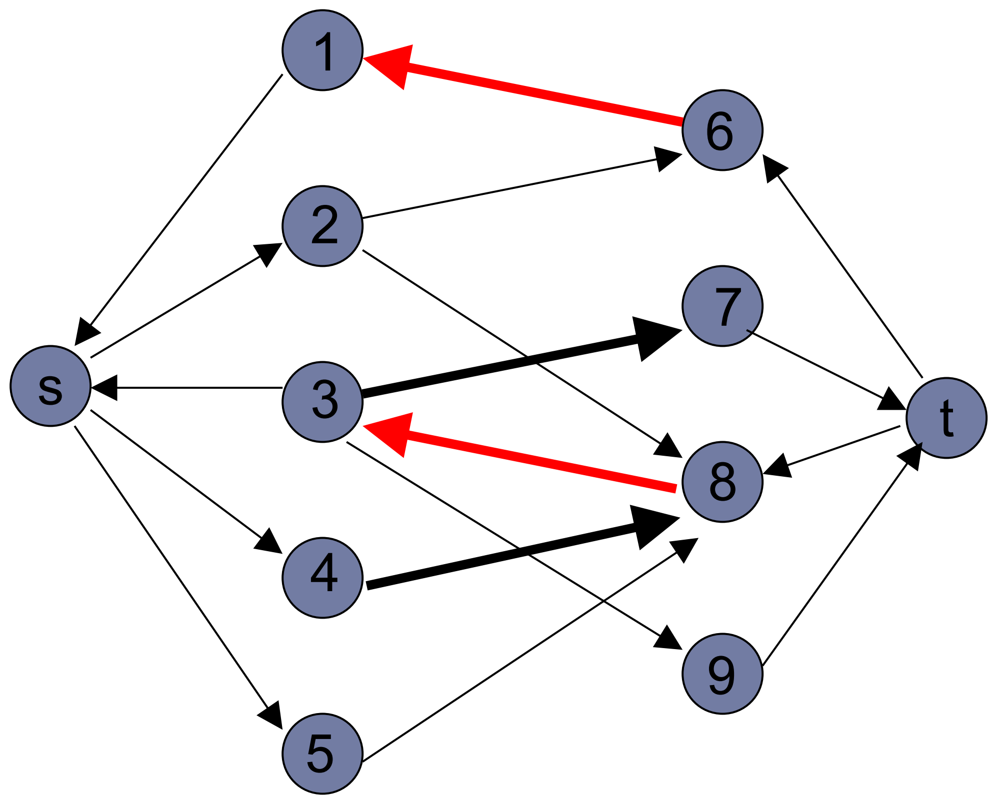
FF on bipartite graphs: a closer look

all edges have capacity 1



FF on bipartite graphs: a closer look

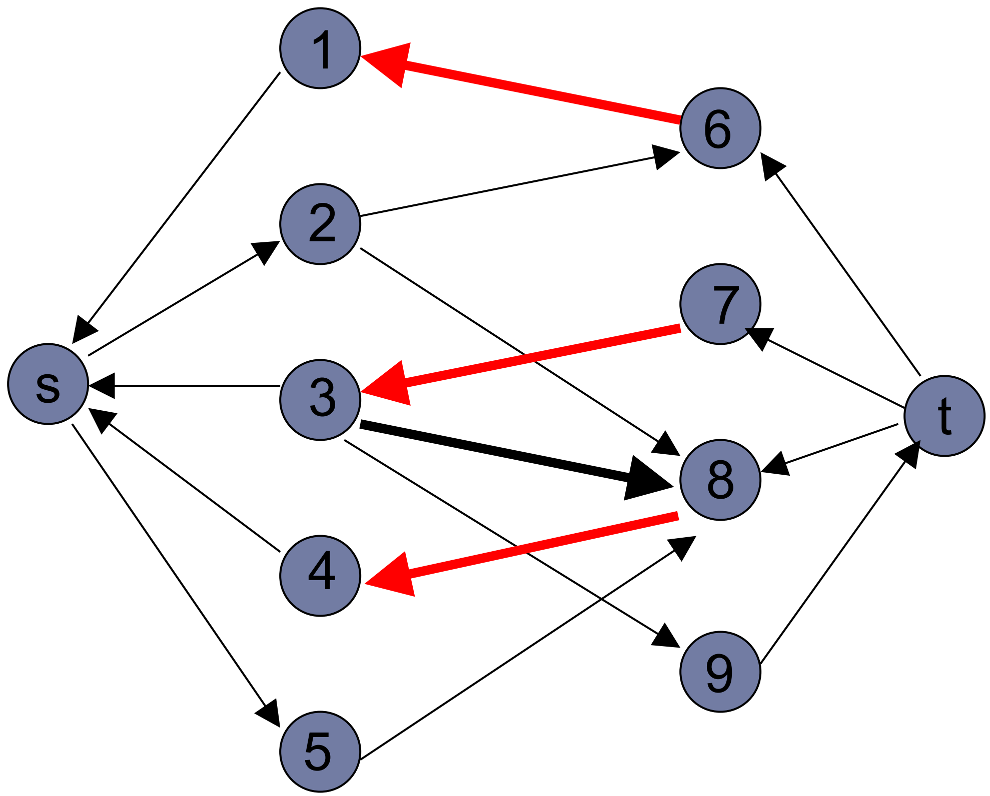
all edges have capacity 1



augmenting path = alternating path

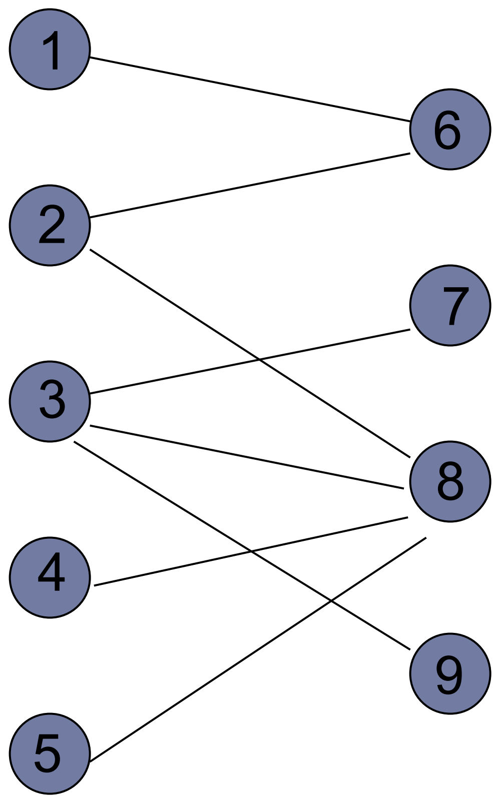
FF on bipartite graphs: a closer look

all edges have capacity 1



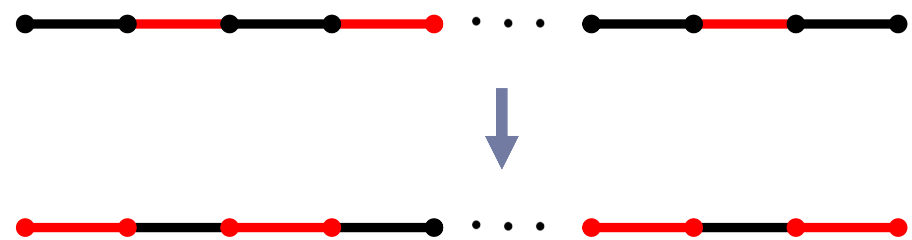
augmenting path = alternating path

FF on bipartite graphs: a closer look

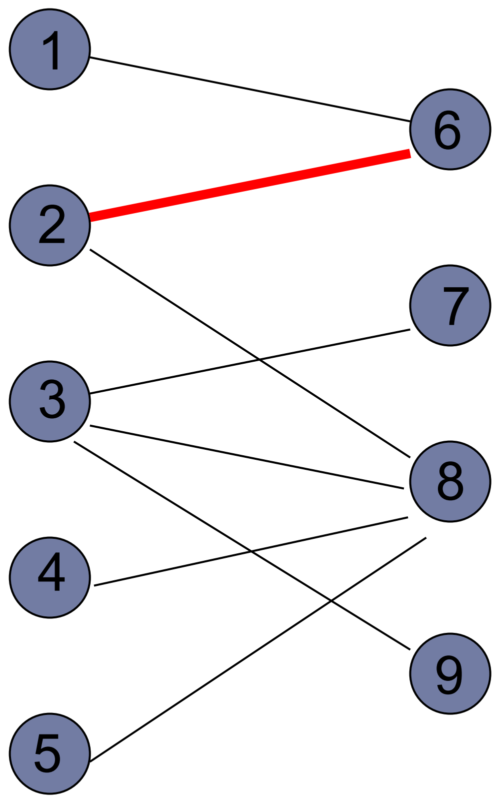


Kuhn's algorithm (equivalent to FF on bipartite graphs):

1. start with empty matching
2. **while** there is an alternating path
   "flip colors"



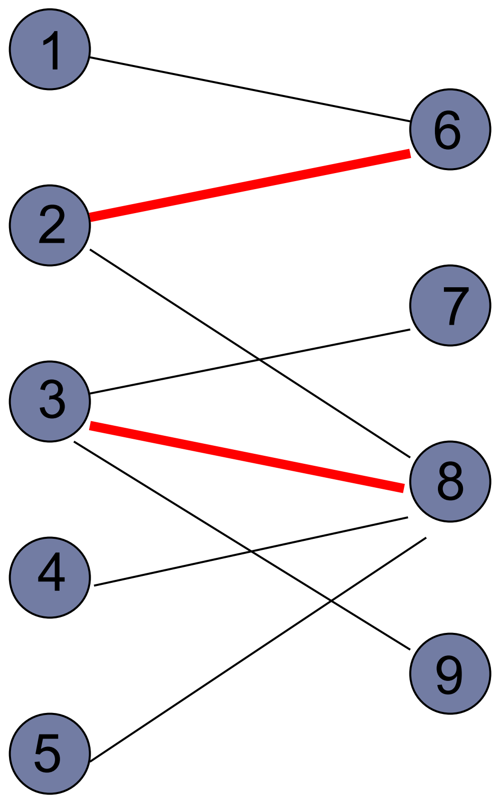
FF on bipartite graphs: a closer look



Kuhn's algorithm (equivalent to FF on bipartite graphs):

1. start with empty matching
2. **while** there is an alternating path
   "flip colors"

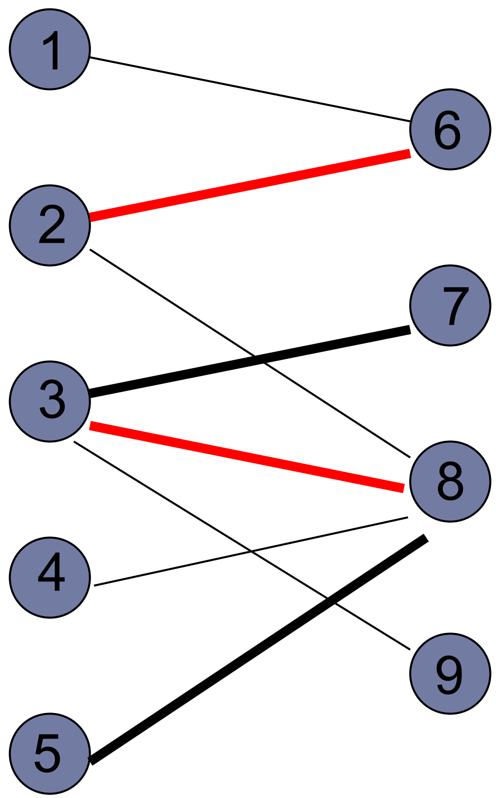
FF on bipartite graphs: a closer look



Kuhn's algorithm (equivalent to FF on bipartite graphs):

1. start with empty matching
2. **while** there is an alternating path
   "flip colors"

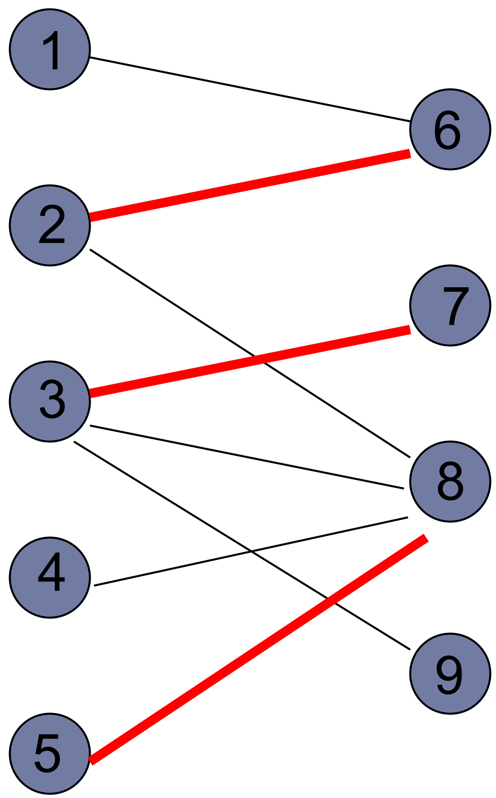
FF on bipartite graphs: a closer look



Kuhn's algorithm (equivalent to FF on bipartite graphs):

1. start with empty matching
2. **while** there is an alternating path
   "flip colors"

FF on bipartite graphs: a closer look



Kuhn's algorithm (equivalent to FF on bipartite graphs):

1. start with empty matching
2. **while** there is an alternating path
   "flip colors"