Lectures 7 and 8: Max Flow & Min Cut

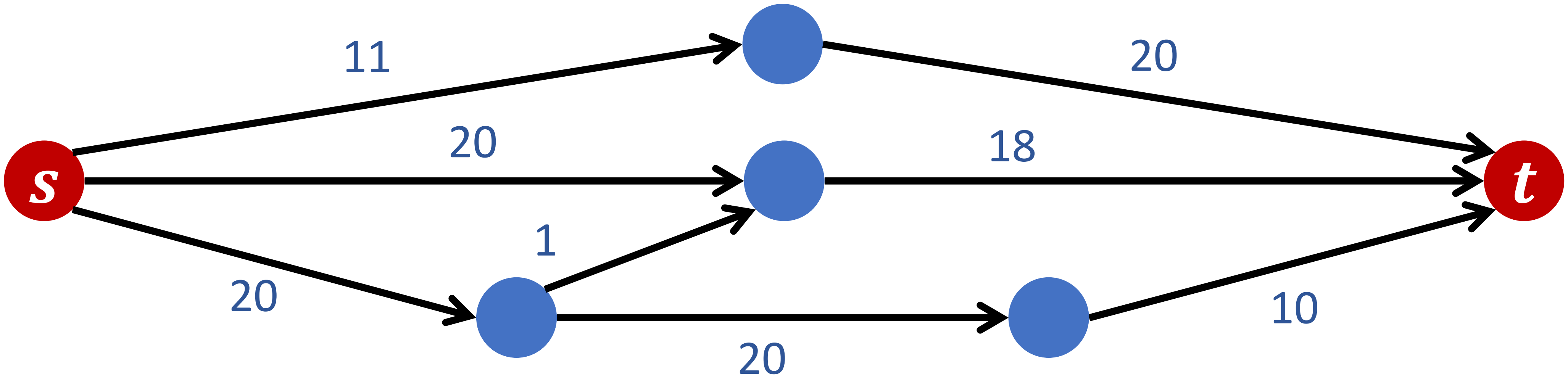
Yury Makarychev
TTIC and the University of Chicago

Maximum Flow & Minimum Cut

Flow Network

An (s, t)-flow network:

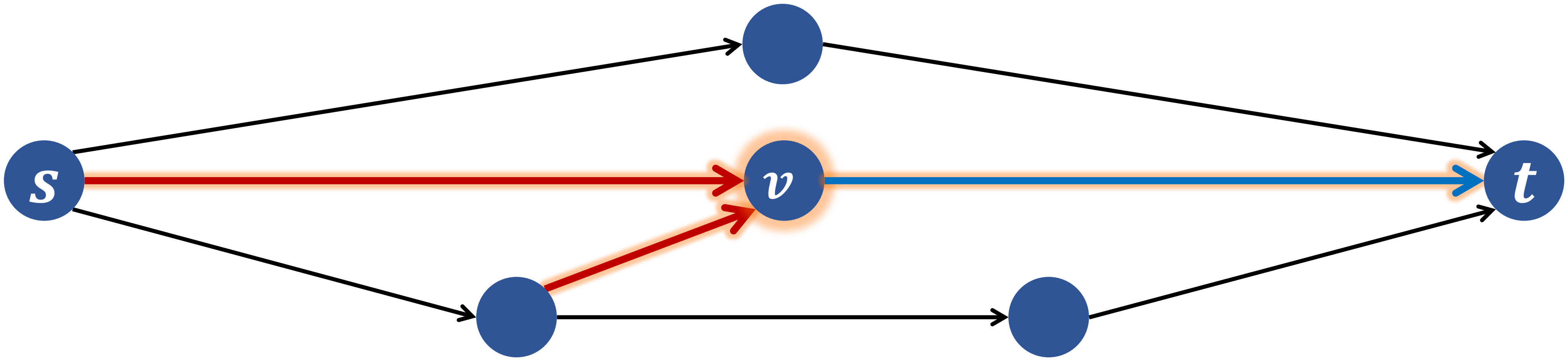
- Directed graph $G = (V, E)$.
- Two designated nodes (terminals): source s and sink t
- Every edge e has a capacity $c(e)$.



Flow Network

$in(v)$ is the set of edges incoming in v
$out(v)$ is the set of edges outgoing from v

Assume: $in(s) = out(t) = \emptyset$



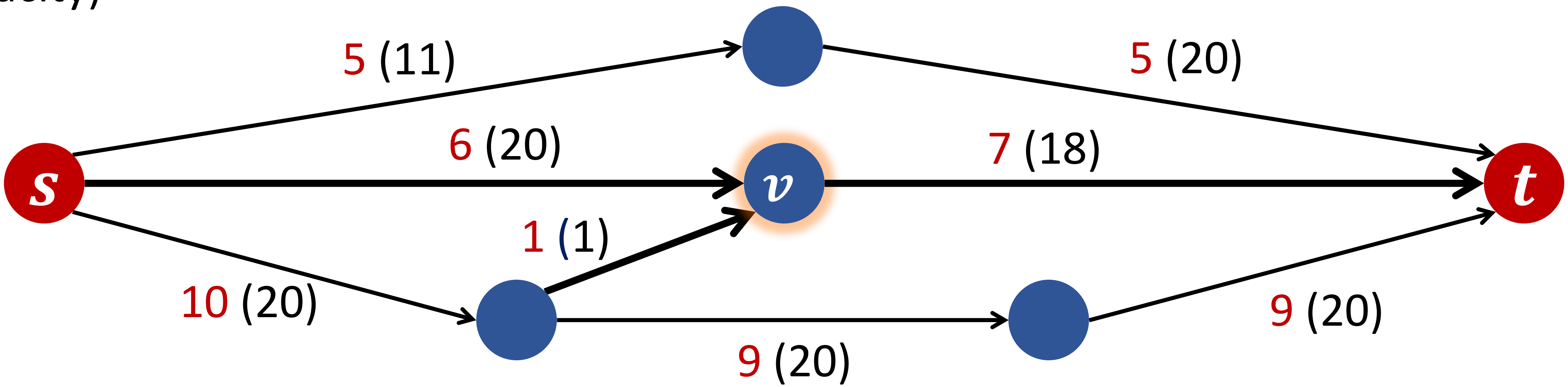
Feasible Flow

Flow $f: E \to \mathbb{R}_{\geq 0}$. We route $f(e)$ units of flow over each edge e.

a. Flow conservation constraints: for all vertices $v \notin \{s, t\}$,

$$\sum_{e \in in(v)} f(e) = \sum_{e \in out(v)} f(e)$$

flow (capacity)



Feasible Flow

Flow $f: E \rightarrow \mathbb{R}_{\geq 0}$. We route $f(e)$ units of flow over each edge e.

a. Flow conservation constraints: for all vertices $v \notin \{s, t\}$,

$$\sum_{e \in in(v)} f(e) = \sum_{e \in out(v)} f(e)$$

b. Capacity constraints: for all edges e:

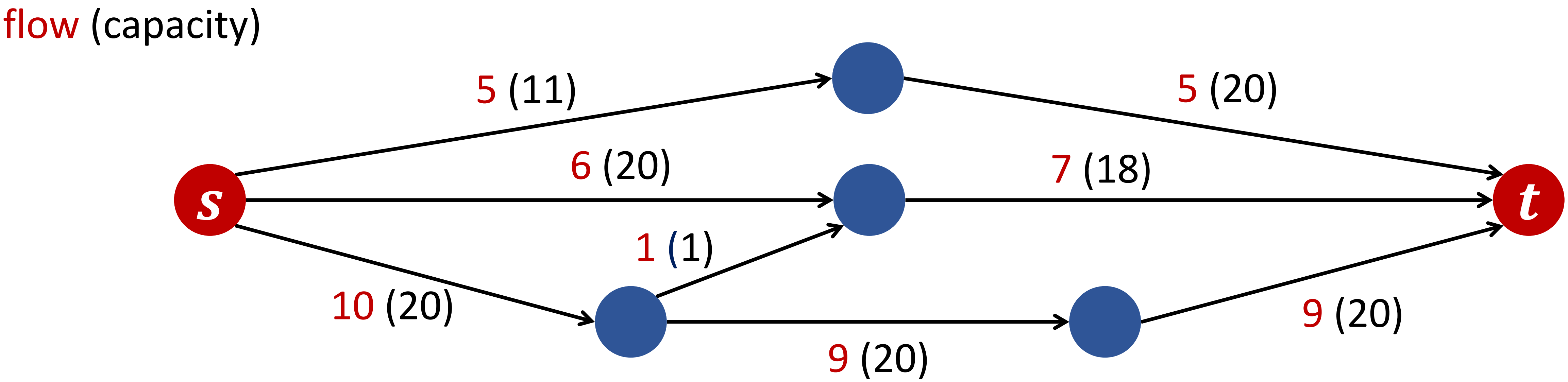
$$f(e) \leq c(e)$$

The value of the flow f equals

$$val(f) = \sum_{e \in out(s)} f(e)$$

Maximum Flow Problem

Given a network $G = (V, E)$, find the maximum feasible flow from s to t.



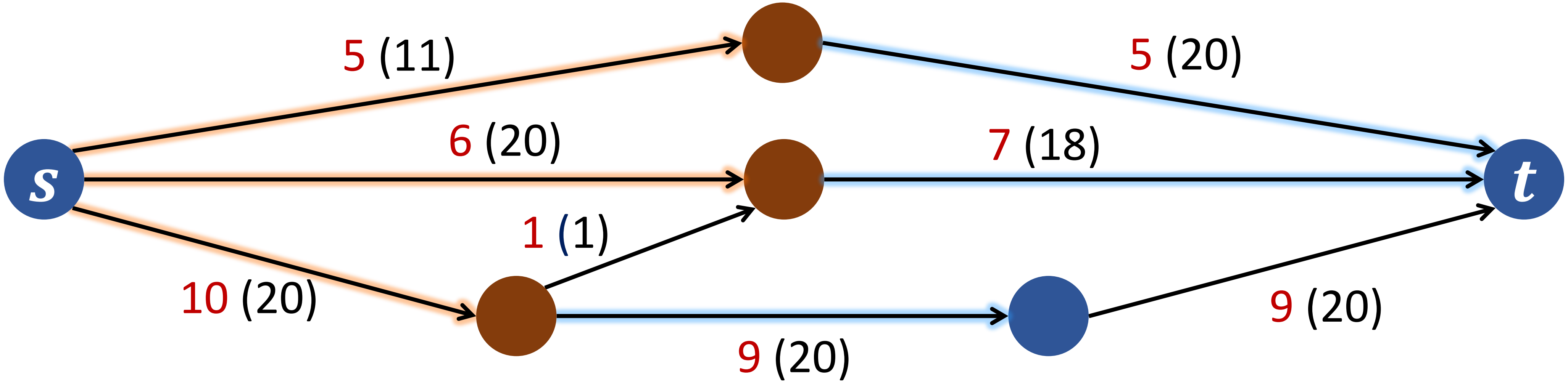
Flow Network

$$f_{in}(S) = \sum_{\substack{(u,v)\in E \\ u\notin S, v\in S}} f(u,v)$$

$$f_{out}(S) = \sum_{\substack{(u,v)\in E \\ u\in S, v\notin S}} f(u,v)$$

$$5 + 6 + 10 = 21$$

$$5 + 7 + 9 = 21$$



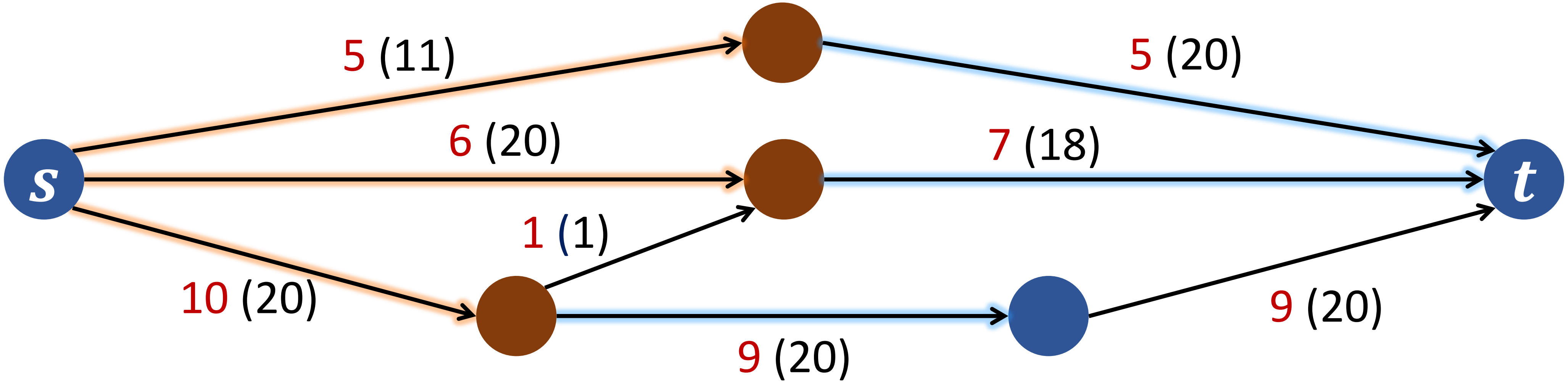
Flow Network

In this example $f_{in}(S) = f_{out}(S)$.

Q: is it always the case?

$5 + 6 + 10 = 21$

$5 + 7 + 9 = 21$



$val(f) = f_{out}(s)$

Flow Network

Claim

a) If $s, t \notin S$ then $f_{in}(S) = f_{out}(S)$

b) If $s, t \in S$ then $f_{in}(S) = f_{out}(S)$

c) If $s \in S$ and $t \notin S$ then $f_{in}(S) = f_{out}(S) - val(f)$

d) If $s \notin S$ and $t \in S$ then $f_{in}(S) = f_{out}(S) + val(f)$

Proof

The claim is very intuitive. E.g.,

(a) says: no flow originates or terminates in S if $s, t \notin S$

(c) says: all flow leaving S either originates at $s \in S$ or first enters S and then leaves it

Prove item (c).

$val(f) = f_{out}(s)$

Proof

If $s \in S$ and $t \notin S$ then $f_{in}(S) = f_{out}(S) - val(f)$

Proof

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

(A) the sum is over all edges entering S

(B) the sum is over all edges entering vertices in S. Those include (A) and edges between vertices in S.

$val(f) = f_{out}(s)$

Proof

If $s \in S$ and $t \notin S$ then $f_{in}(S) = f_{out}(S) - val(f)$

Proof

$$f_{in}(S) = \sum_{u \in S} f_{in}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v) = \sum_{u \in S \setminus \{s\}} f_{in}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v)$$

since $f_{in}(s) = 0$

$val(f) = f_{out}(s)$

Proof

If $s \in S$ and $t \notin S$ then $f_{in}(S) = f_{out}(S) - val(f)$

Proof

$$f_{in}(S) = \sum_{u \in S \setminus \{s\}} f_{in}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v) = \sum_{u \in S \setminus \{s\}} f_{out}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v)$$

the flow conservation constraint
(use that $s, t \notin S \setminus \{s\}$)

$val(f) = f_{out}(s)$

Proof

If $s \in S$ and $t \notin S$ then $f_{in}(S) = f_{out}(S) - val(f)$

Proof

$$f_{in}(S) = \sum_{u \in S \setminus \{s\}} f_{in}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v) = \sum_{u \in S \setminus \{s\}} f_{out}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v)$$

the flow conservation constraint
(use that $s, t \notin S \setminus \{s\}$)

$val(f) = f_{out}(s)$

Proof

If $s \in S$ and $t \notin S$ then $f_{in}(S) = f_{out}(S) - val(f)$

Proof

$$f_{in}(S) = \sum_{u \in S \setminus \{s\}} f_{out}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v)$$

$$= \left(\sum_{u \in S} f_{out}(u) - f_{out}(s) \right) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v)$$

$val(f) = f_{out}(s)$

Proof

If $s \in S$ and $t \notin S$ then $f_{in}(S) = f_{out}(S) - val(f)$

Proof

$$f_{in}(S) = \left(\sum_{u \in S} f_{out}(u) - \sum_{\substack{(u,v) \in E \\ u,v \in S}} f(u,v) \right) - val(f)$$

$$= f_{out}(S) - val(f)$$

■

Corollary

$$f_{in}(t) = f_{out}(V \setminus \{t\}) = f_{in}(V \setminus \{t\}) + val(f) = val(f)$$

why?

why?

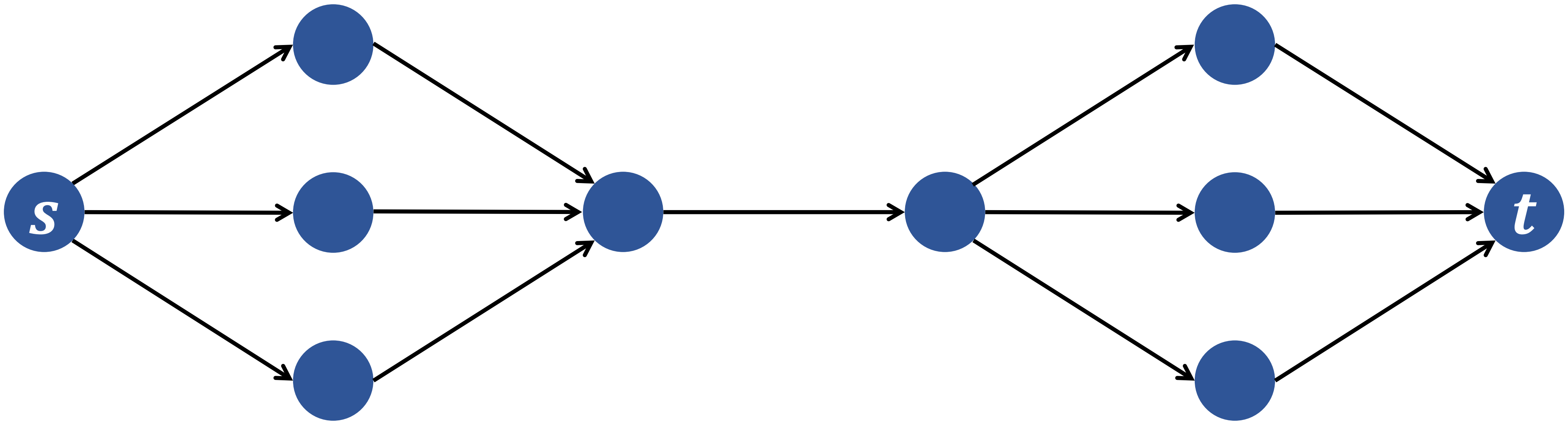
Flow Network

$$val(f) = f_{out}(s) = f_{in}(t) = f_{out}(S) - f_{in}(S)$$

for every set S that contains s but not t

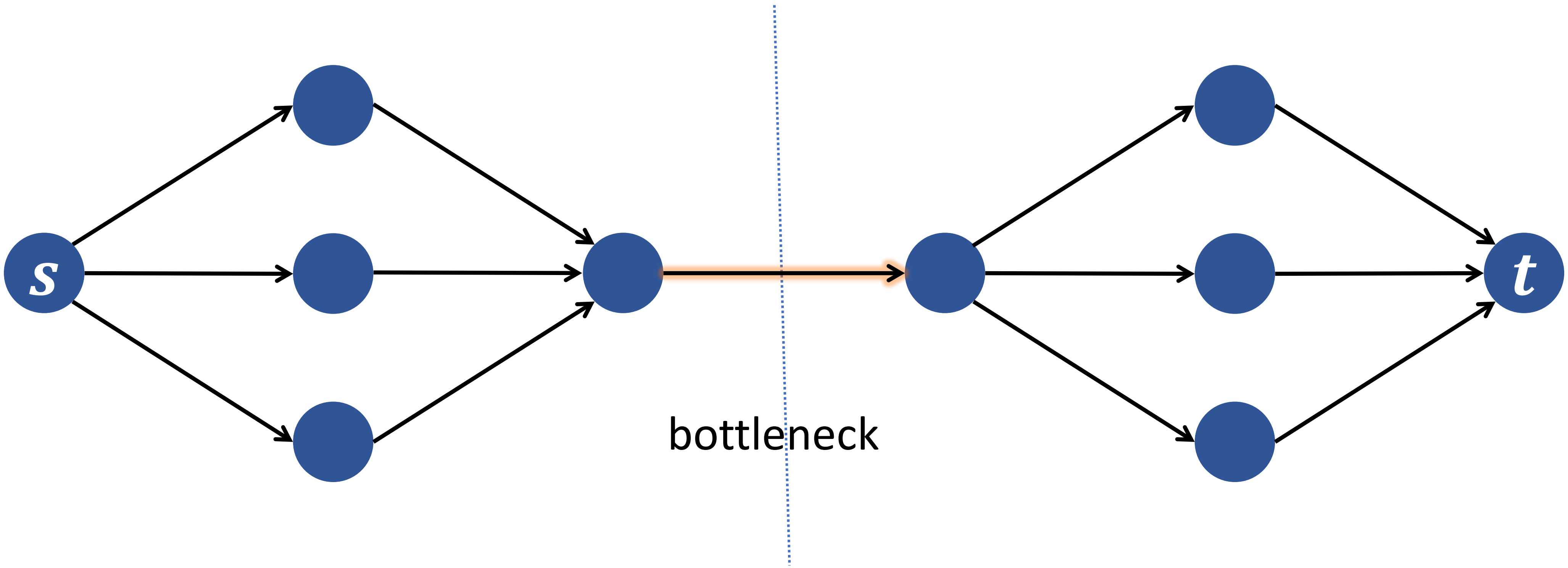
What is the value of the maximum flow?

In this example, all edges have capacity 1.



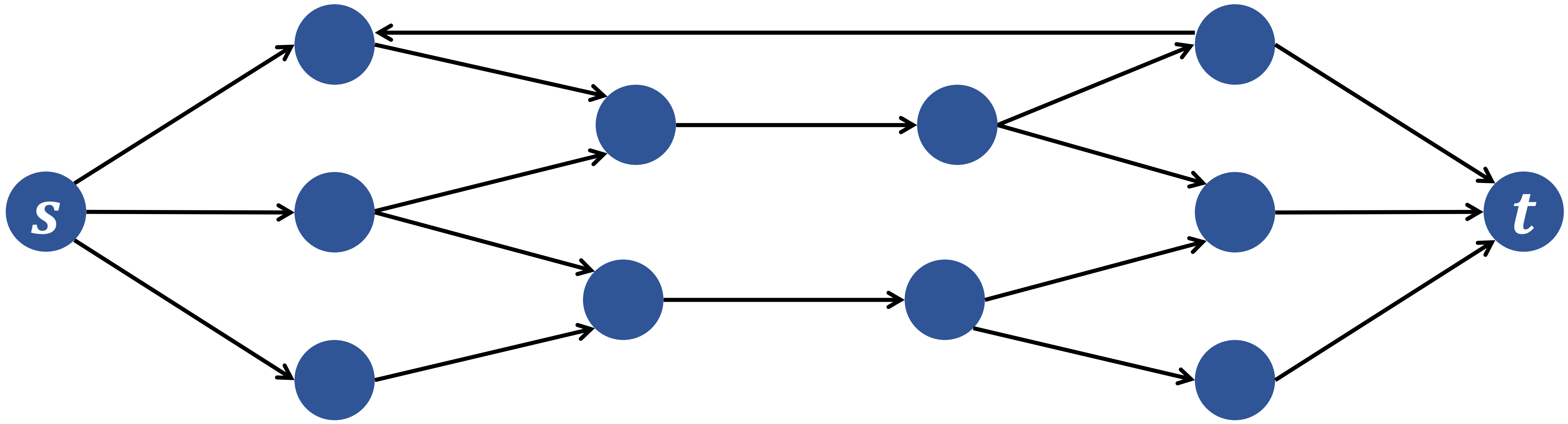
What is the value of the maximum flow?

In this example, all edges have capacity 1.



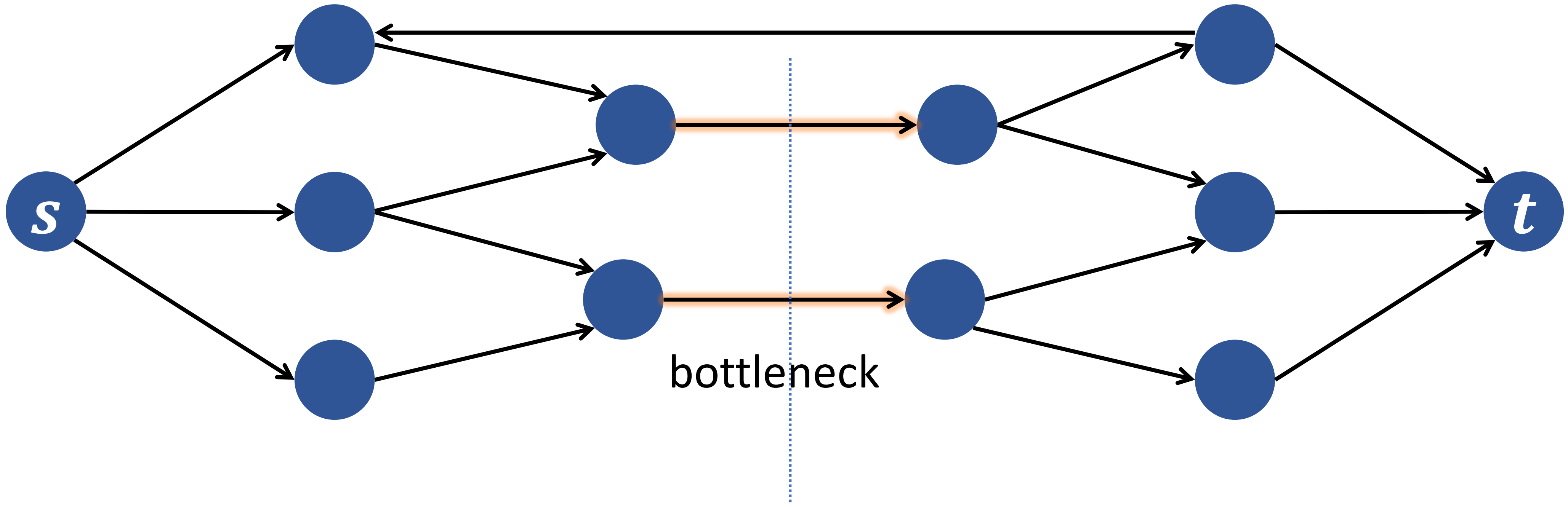
What is the value of the maximum flow?

In this example, all edges have capacity 1.



What is the value of the maximum flow?

In this example, all edges have capacity 1.



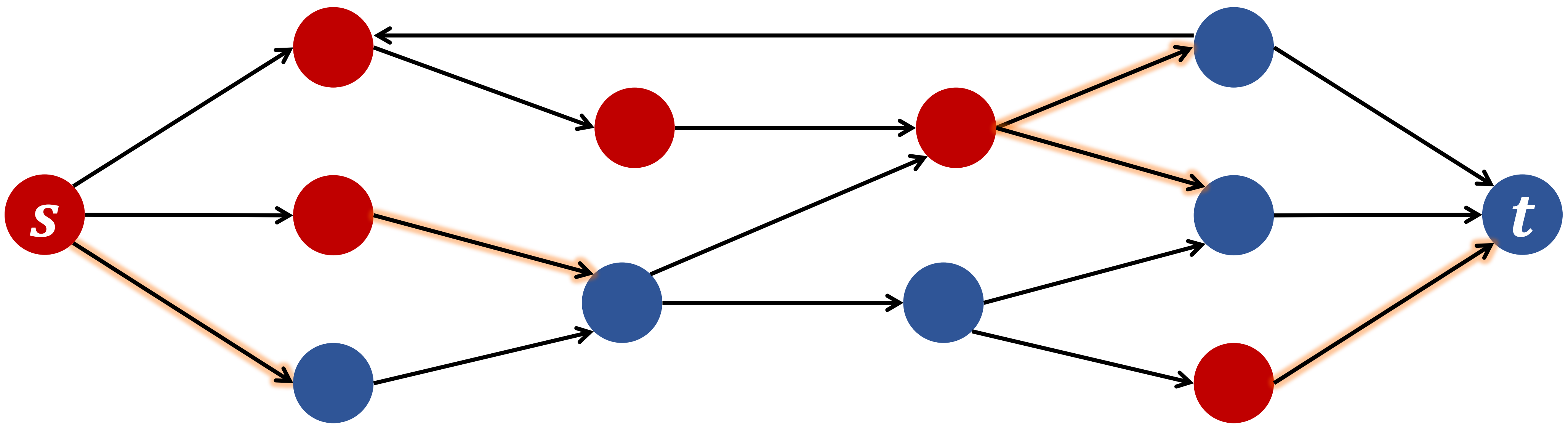
Minimum Cut

Directed Cut

A directed cut is a partition of V into two nonempty sets S and T:

$$V = S \cup T \text{ and } S \cap T = \emptyset$$

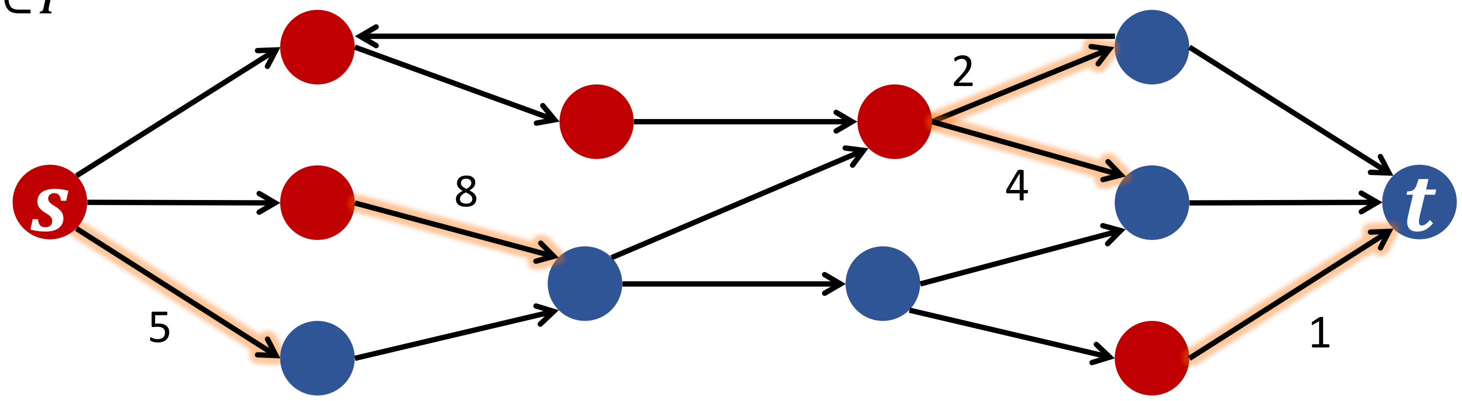
An edge (u, v) is cut if $u \in S$ and $v \in T$. Edges from T to S are not cut!



The capacity or cost of the cut is the total capacity of all cut edges.

The capacity/cost of a cut

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

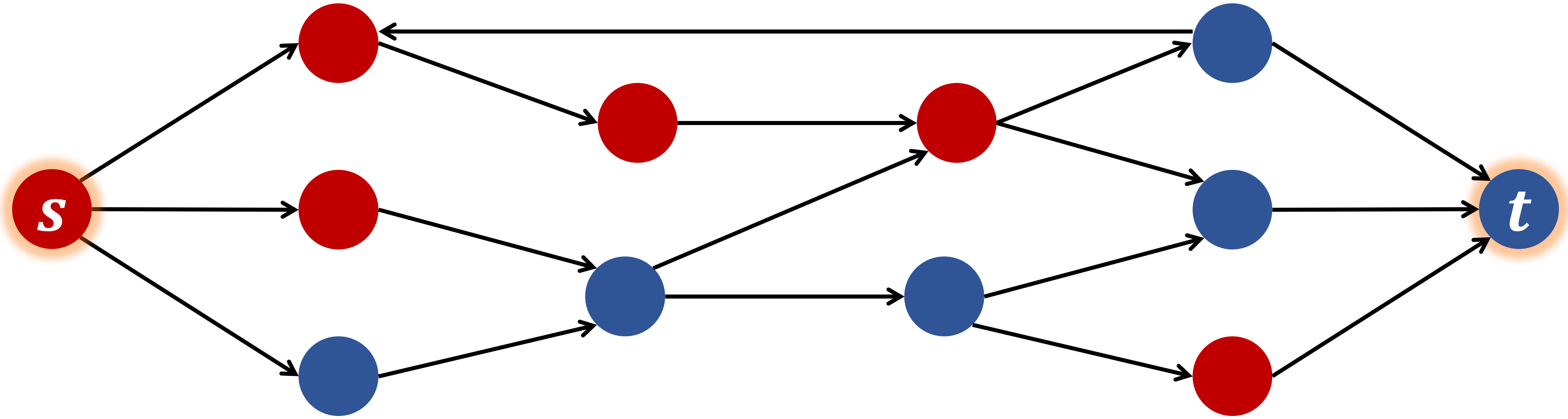


In this example, $cap(S,T) = 5 + 8 + 2 + 4 + 1 = 20.$

s-t Cut

(S, T) is an s-t cut if $s \in S$ and $t \in T$.

(S, T) is a minimum s-t cut if $cap(S, T) \leq cap(S', T')$ for every s-t cut (S', T').



$f_{in}(S) = f_{out}(S) - val(f)$

Weak Duality

Let f be a feasible s-t flow and (S, T) be an s-t cut. Then

$$val(f) \leq cap(S, T)$$

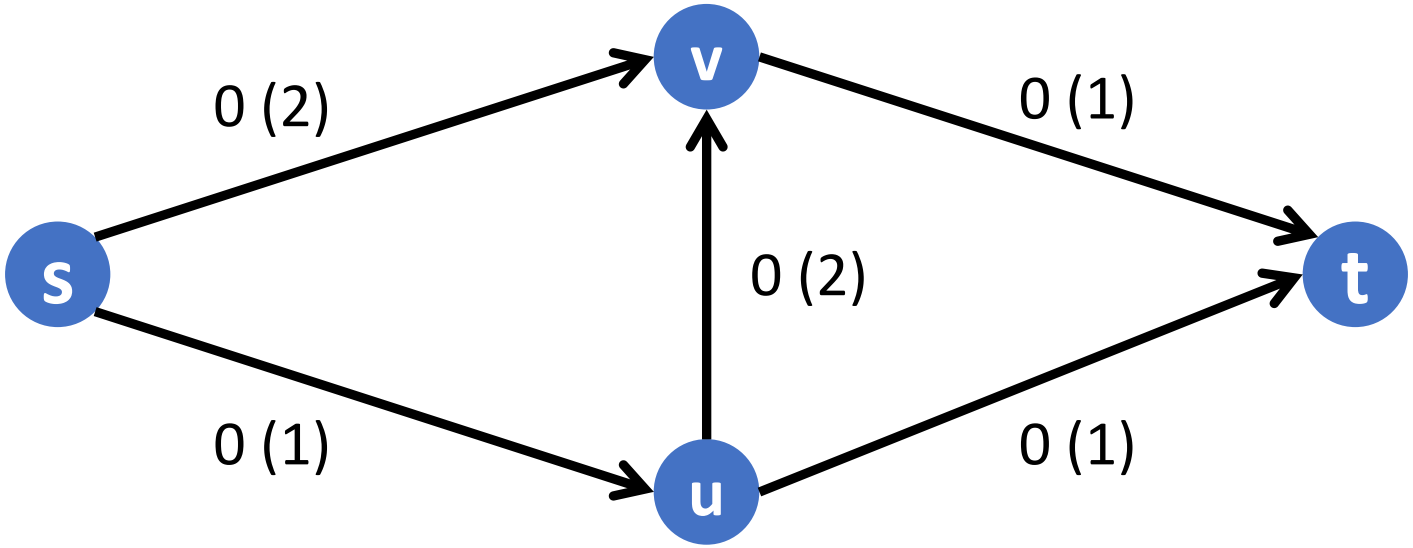
Proof: Since $s \in S$ and $t \notin S$,

$$val(f) = f_{out}(S) - f_{in}(S) \leq f_{out}(S) = \sum_{\substack{(u,v)\in E \\ u\in S, v\in T}} f(u, v)$$

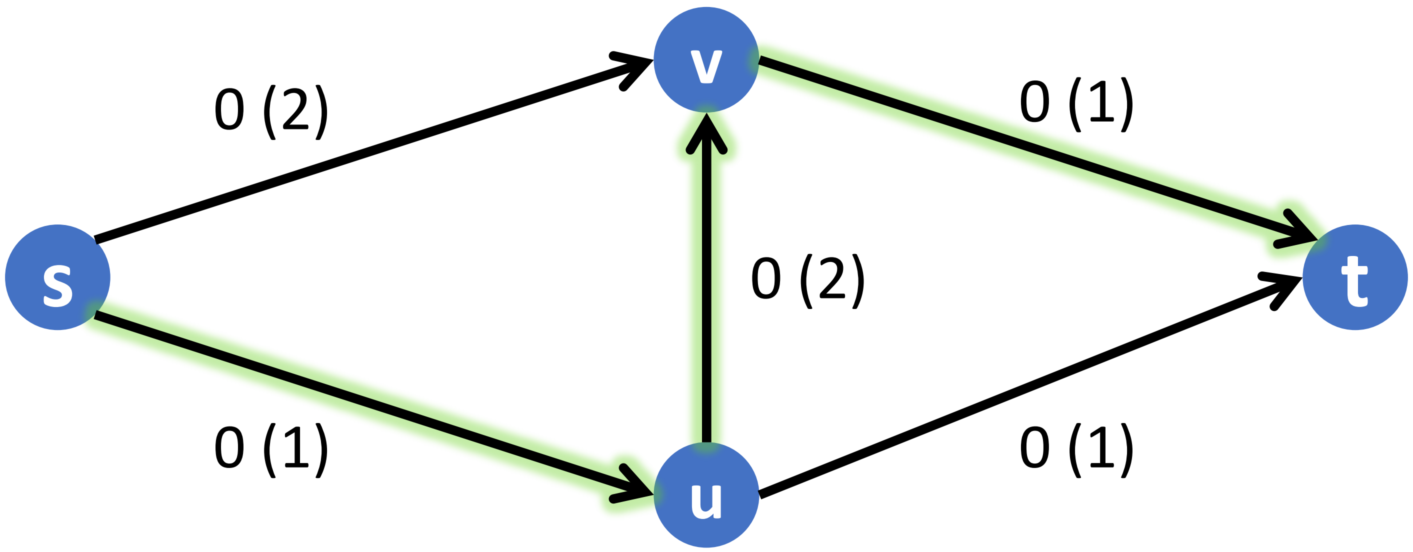
$$\leq \sum_{\substack{(u,v)\in E \\ u\in S, v\in T}} cap(u, v) = cap\,(S, T)$$

Ford–Fulkerson Algorithm

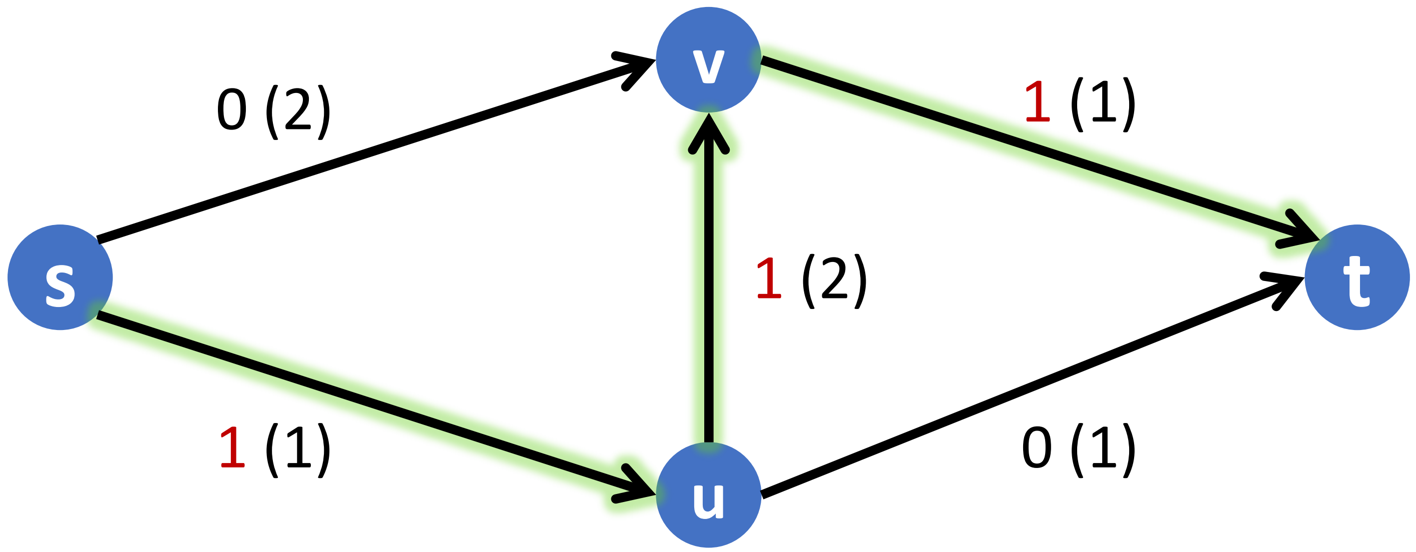
Attempt #1: Naïve Greedy Algorithm



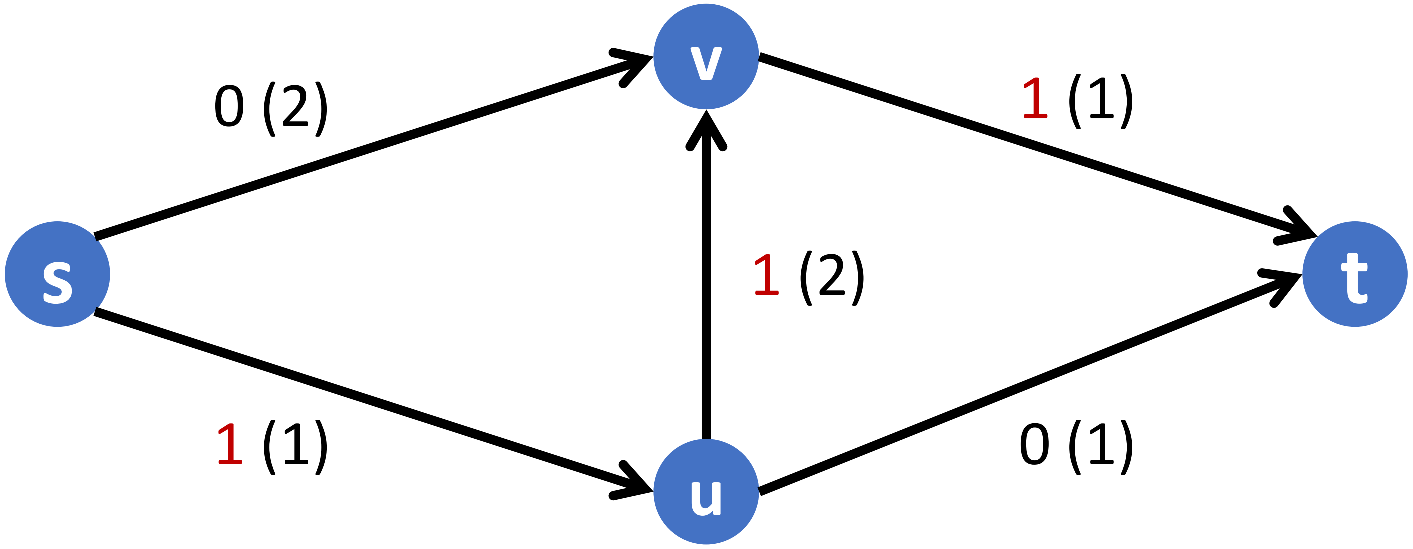
Attempt #1: Naïve Greedy Algorithm



Attempt #1: Naïve Greedy Algorithm

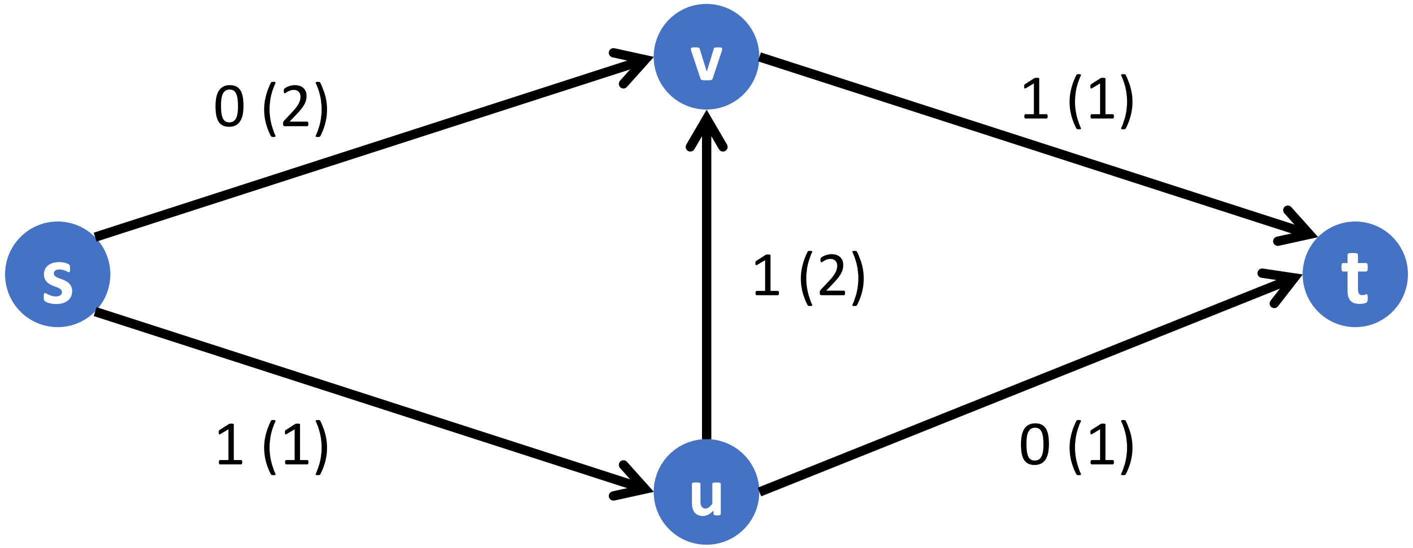


Two edges are at capacity and cannot route any more flow.
The algorithms cannot increase the flow.



Maximum flow: 2

Our algorithm found a flow of value 1



Given a flow f, define the residual network G_f

G_f is a directed graph on V with the following edges:

- edges $(u, v) \in E$ with $f(u, v) < c(u, v)$

  edge (u, v) is a forward edge of G_f

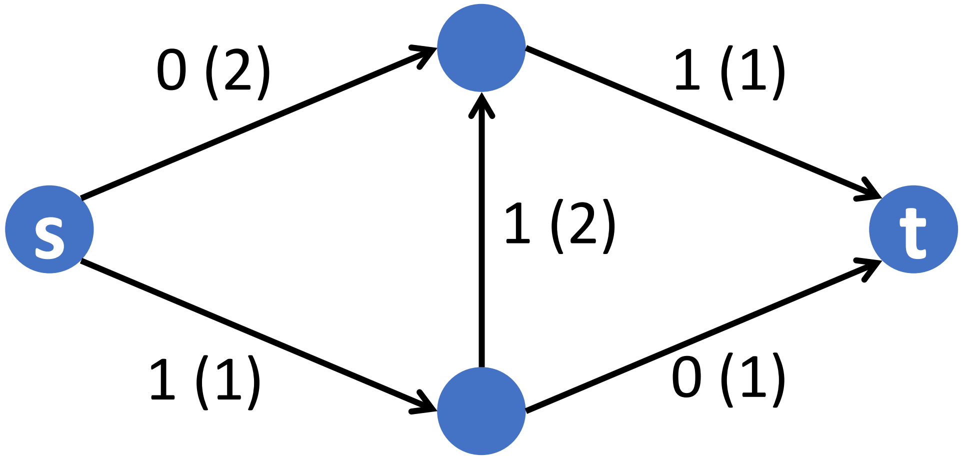
  its residual capacity $c_f(u, v) = c(u, v) - f(u, v)$

- edges $e' = (v, u)$ where $(u, v) \in E$ and $f(u, v) > 0$

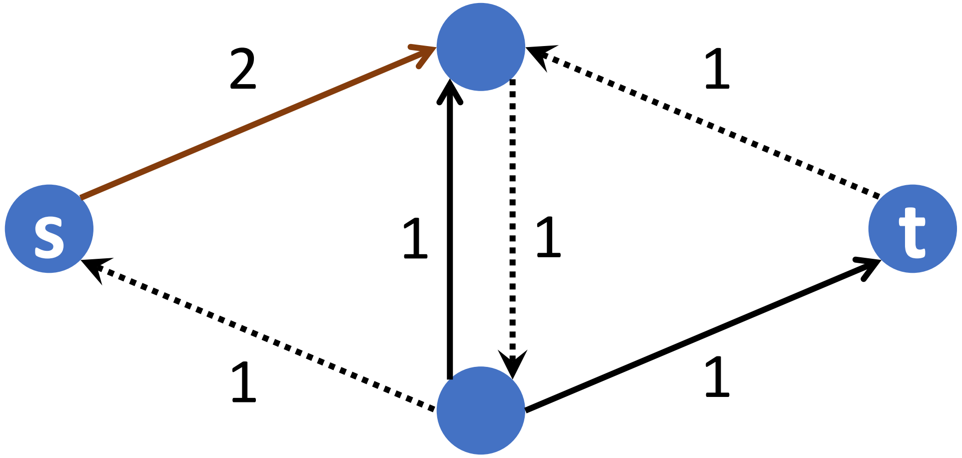
  edge (v, u) is a backward edge

  $c_f(v, u) = f(u, v)$

Residual Network



Flow network G



Residual network G_f

Forward edges – solid lines

Backward edges – dotted lines

Augmenting the flow along a path

Let f be a feasible s-t flow and P be a path in G_f.

Let $\delta \leq c_f(e)$ for every $e \in P$.

- for every forward edge $(u, v) \in P$, increase $f(u, v)$ by δ

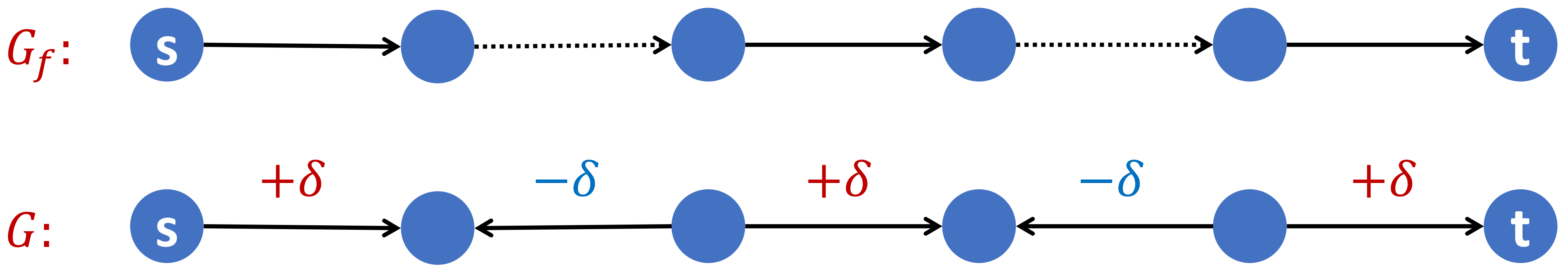
$$f'(u, v) = f(u, v) + \delta$$

- for every backward edge (v, u), decrease $f(u, v)$ by δ

$$f'(u, v) = f(u, v) - \delta$$

Augmenting the Flow

- $f'(u,v) = f(u,v) + \delta$ if (u,v) is a forward edge
- $f'(u,v) = f(u,v) - \delta$ if (v,u) is a backward edge



Augmenting the Flow

Claim: Obtained flow f' is a feasible flow.

Proof: Verify that f' satisfies capacity constraints.

If $(u, v) \in P$ and (u, v) is a forward edge,

$$f'(u,v) = f(u,v) + \delta \leq f(u,v) + c_f(u,v) = c(u,v)$$
$$\geq 0$$

If $(v, u) \in P$ and (v, u) is a backward edge,

$$f'(u,v) = f(u,v) - \delta \geq f(u,v) - c_f(v,u) = 0$$
$$\leq f(u,v) \leq c(u,v)$$

Otherwise, $f'(e) = f(e)$

Augmenting the Flow

Claim: Obtained flow f' is a feasible flow.

Proof: Verify that f' satisfies flow conservation constraints.

G_f: $v \to u \to w$	$f_{in}(u)$	$f_{out}(u)$
$\bullet \rightarrow u \rightarrow \bullet$	$+\delta$	$+\delta$
$\bullet \rightarrow u \leftarrow \bullet$	$+\delta + (-\delta) = 0$	0
$\bullet \leftarrow u \rightarrow \bullet$	0	$-\delta + \delta = 0$
$\bullet \leftarrow u \leftarrow \bullet$	$-\delta$	$-\delta$

Augmentation: Flow Value

Q: What is the value of the augmented flow f'?

$$val(f') = ?$$

Augmentation: Flow Value

Q: What is the value of the augmented flow f'?

$$val(f') = val(f) + \delta$$

Q: What is the best choice of δ for a given path P?

Augmentation: Flow Value

Q: What is the value of the augmented flow f'?

$$val(f') = val(f) + \delta$$

Q: What is the best choice of δ for a given path P?

A: $\delta = \min_{e \in P} c_f(e)$

Q: What happens with the residual capacities of edges on P?

Augmentation: Flow Value

Q: What is the value of the augmented flow f'?

$$val(f') = val(f) + \delta$$

Q: What is the best choice of δ for a given path P?

A: $\delta = \min_{e \in P} c_f(e)$

Q: What happens with the residual capacities of edges on P?

Forward edge: $f'(e) = f(e) + \delta \quad \Rightarrow \quad c_{f'}(e) = c_e - f'(e) = c_f(e) - \delta$

Backward edge: $f'(e') = f(e') - \delta \Rightarrow \quad c_{f'}(e) = f'(e') = f(e') - \delta = c_f(e) - \delta$

here e is the backward edge for e'

Augmentation: Flow Value

Summary

A bottleneck edge on P is the edge with the least residual capacity δ.

- $val(f') = val(f) + \delta$
- The residual capacity of each edge on P decreases by δ.
- Bottleneck edges disappear.

Ford-Fulkerson Algorithm

- start with an empty flow f: $f(e) = 0$ for all $e \in E$. $G_f = G$
- while there is an s-t path P in G_f

  augment flow f along path P

  update G_f
- return f

Ford-Fulkerson Algorithm

- start with an empty flow f: $f(e) = 0$ for all $e \in E$. $G_f = G$
- while there is an s-t path P in G_f

  augment flow f along path P

  update G_f
- return f

TODO items:

- Prove that the algorithm finds an optimal flow (if it stops)
- Does the algorithm stop? Find its running time.

Optimality

Need to prove: when the algorithm, f is a maximum flow.

The Max Flow / Min Cut Theorem:

1. If there is no s-t path P in G_f then f is a maximum flow.
2. (Strong Duality)

$$val(f) = cap(A, B)$$

where (A, B) is a minimum cut.

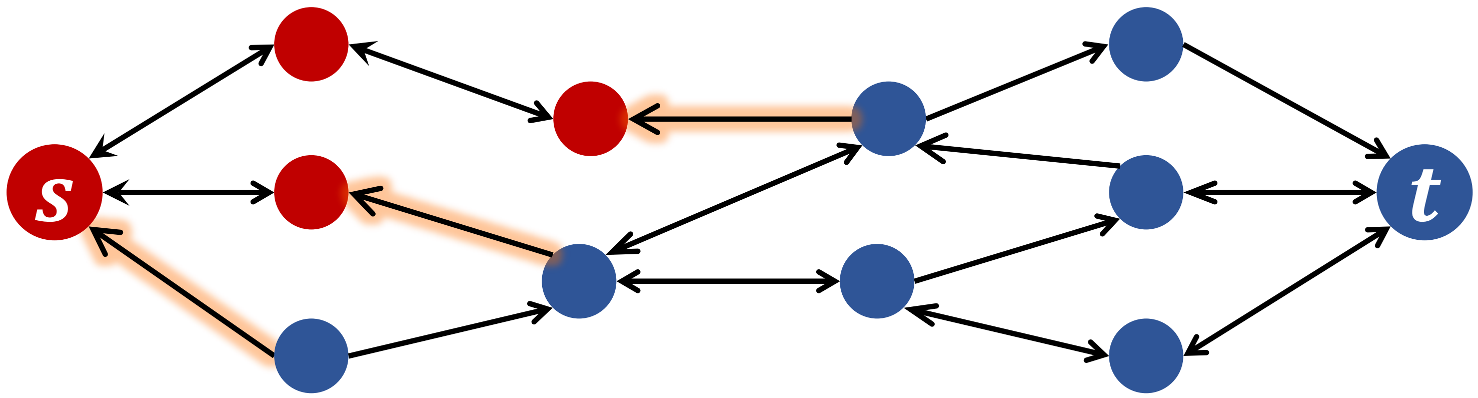
Optimality

Let $\quad A = \{u\text{: there is a path from } s \text{ to } u \text{ in } G_f\}$ (vertices reachable from s)

$B = V \setminus A$

Note that

- $s \in A$ (trivially)
- $t \notin A$, since there is no s-t path in G_f. Thus, $t \in B$.



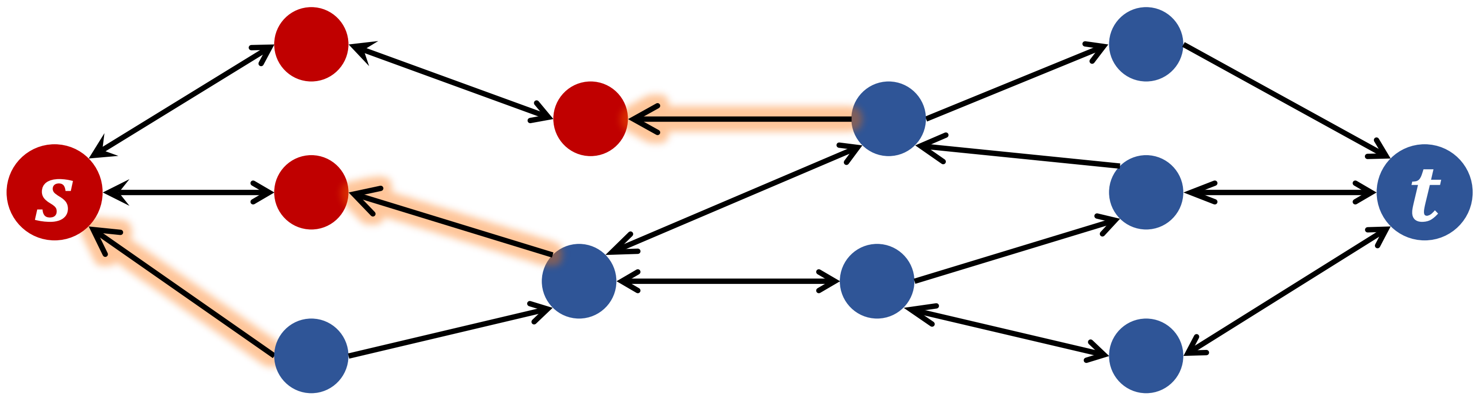
Optimality

Thus, (A, B) is an s-t cut

There are no edges from A to B in G_f. Why?

$\Rightarrow$ if $u \in A, v \in B, (u, v) \in E$, then $f(u, v) = c(u, v)$

$\Rightarrow$ if $u \in B, v \in A, (u, v) \in E$, then $f(u, v) = 0$



Optimality

$\Rightarrow$ if $u \in A$, $v \in B$, $(u, v) \in E$, then $f(u, v) = c(u, v)$

$\Rightarrow$ if $u \in B$, $v \in A$, $(u, v) \in E$, then $f(u, v) = 0$

$$cap(A, B) = \sum_{\substack{(u,v)\in E \\ u\in A, v\in B}} c(u, v) = \sum_{\substack{(u,v)\in E \\ u\in A, v\in B}} f(u, v) = f_{out}(A) = f_{in}(A) + val(f) = val(f)$$

Using weak duality:

$$val(f') \leq cap(A, B) = val(f) \leq cap(A', B')$$

Thus f is a maximum flow, (A, B) is a minimum cut, and $val(f) = cap\ (A, B)$.

Integrality

Assume that all capacities are integers.

Prove by induction that after each iteration:

- The flow will be integral.
- All residual capacities will be integral.
- The bottleneck capacity δ will be integral.

The algorithm returns an integral maximum flow.

Corollary: There is an integral maximum flow.

Note that there may be a fractional maximum flow as well.

Running Time

In each iteration the flow increases by 1.

Therefore, the algorithm stops in at most val^* iterations.

$$val^* \leq C = \sum_{e \in out(s)} c(e)$$

Each iteration can be implemented in $O(n)$ time (use BFS or DFS).

Running time: $O(val^* \cdot n)$.

If all capacities are rational numbers, the algorithm also always terminates.

Variants of Ford-Fulkerson

There are various rules for choosing path P.

The running time depends on the rule.

1. Scaling variant: $O((\log C + 1)\, m^2)$ (see the textbook)
2. Edmonds–Karp: $O(m^2 n)$ doesn't depend on the capacities (strongly polynomial-time algorithm).

   The algorithm uses BSF to find a shortest path P between s and t.

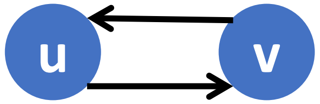
Finding a Minimum Cut

Q: Can we find a minimum cut (A, B) in G using the Ford-Fulkerson algorithm?

Max Flow & Min Cut in Undirected Graphs

Undirected Graphs

We can reduce the case of undirected graphs to that of directed.



The capacity of an s-t cut (S, T) is the total capacity of edges from S to T.

The Max Flow / Min Cut Theorem also holds or undirected graphs.

Applications of Max Flow and Min Cut

Routing

- Routing vehicles on the road.
- Routing electricity.
- Routing internet traffic.

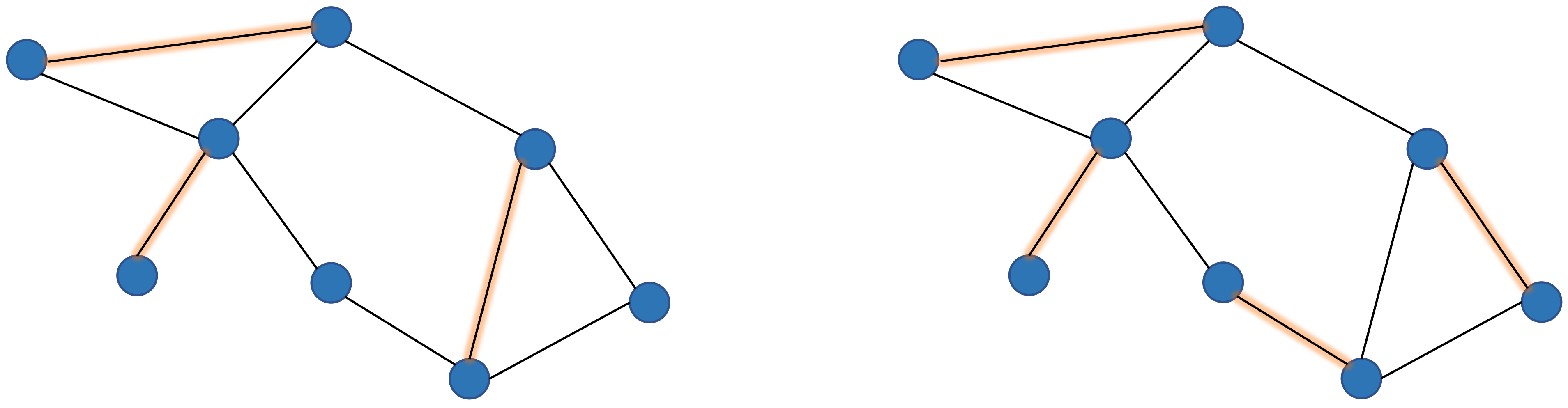
Bipartite Matching

Matching

Consider an undirected graph.

A subset of edges M is a matching if no two edges in M share a vertex.

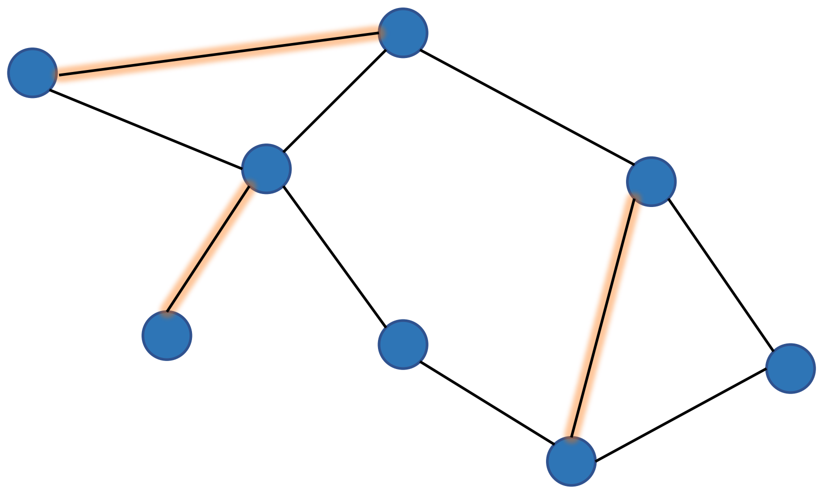
A vertex u is matched by M if there is an edges $(u, v) \in M$ for some v.



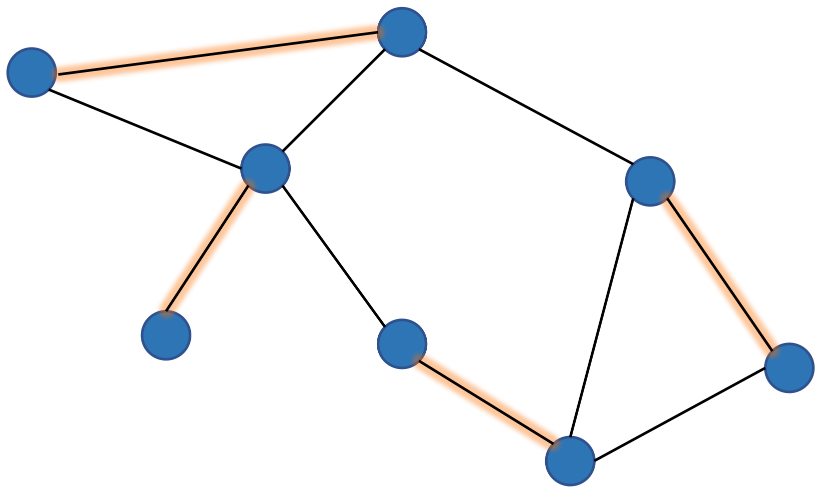
Matching

A matching is a perfect matching in G if all vertices are matched.

not a perfect matching



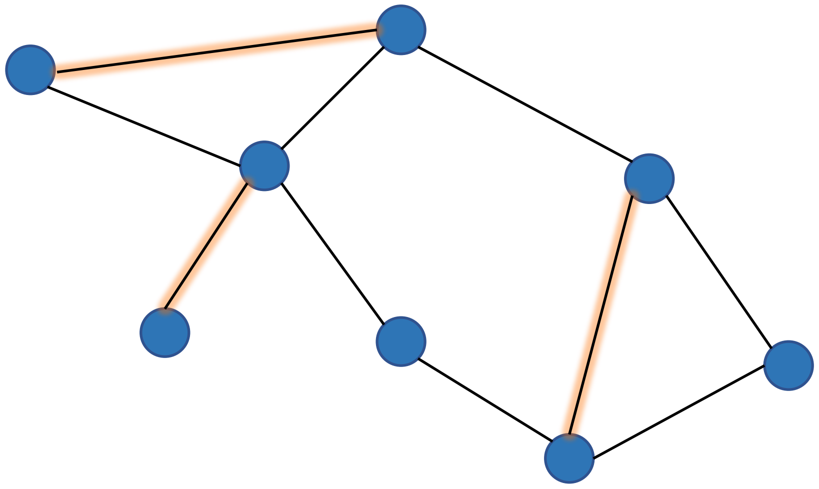
perfect matching



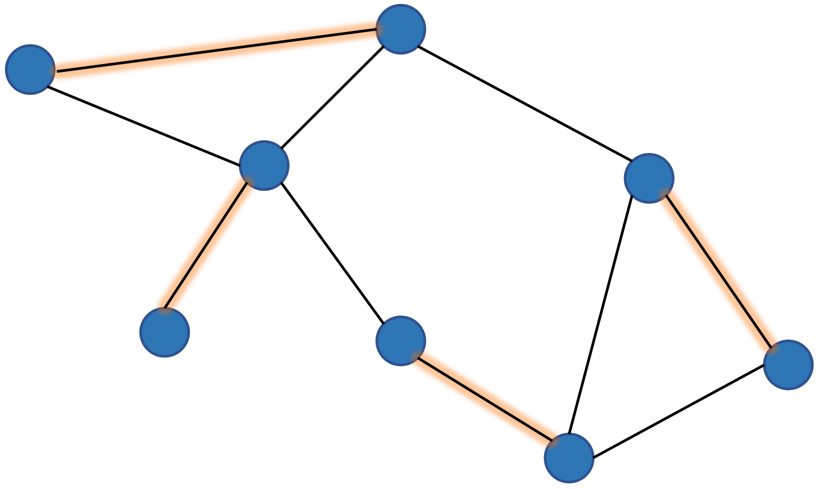
Matching

M is a maximum matching if $|M| \geq |M'|$ for every matching M'.
M is a maximal matching if $M \cup \{e\}$ is not a matching for every $e \notin M$.

maximal matching
(not maximum)



maximum matching

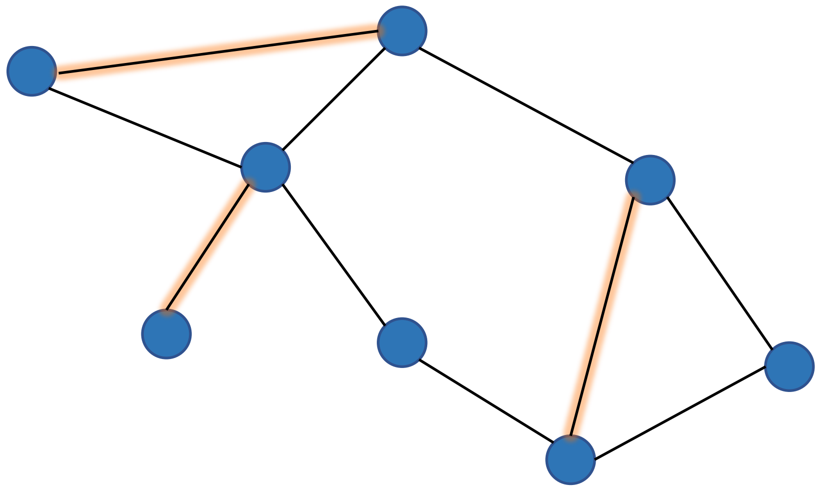


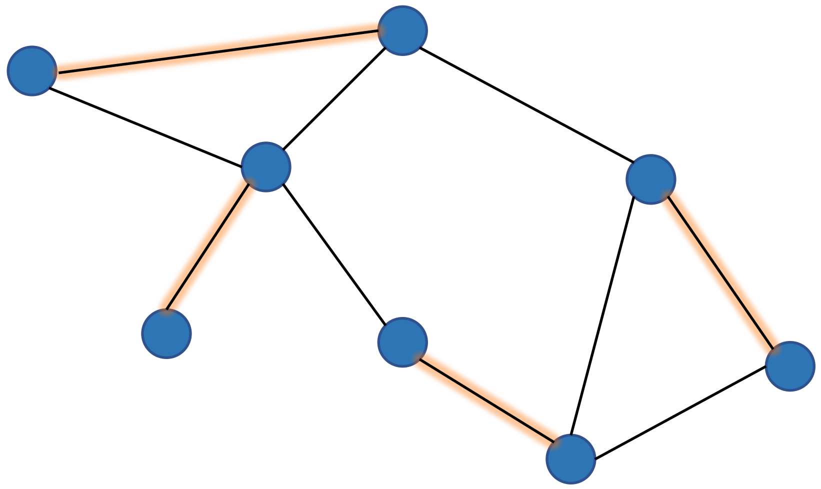
Greedy Algorithm for Finding Matchings

Consider a greedy algorithm:

- start with $M = \emptyset$
- while there is an edge $e \notin M$ s.t. $M \cup \{e\}$ is a matching
  add e to M

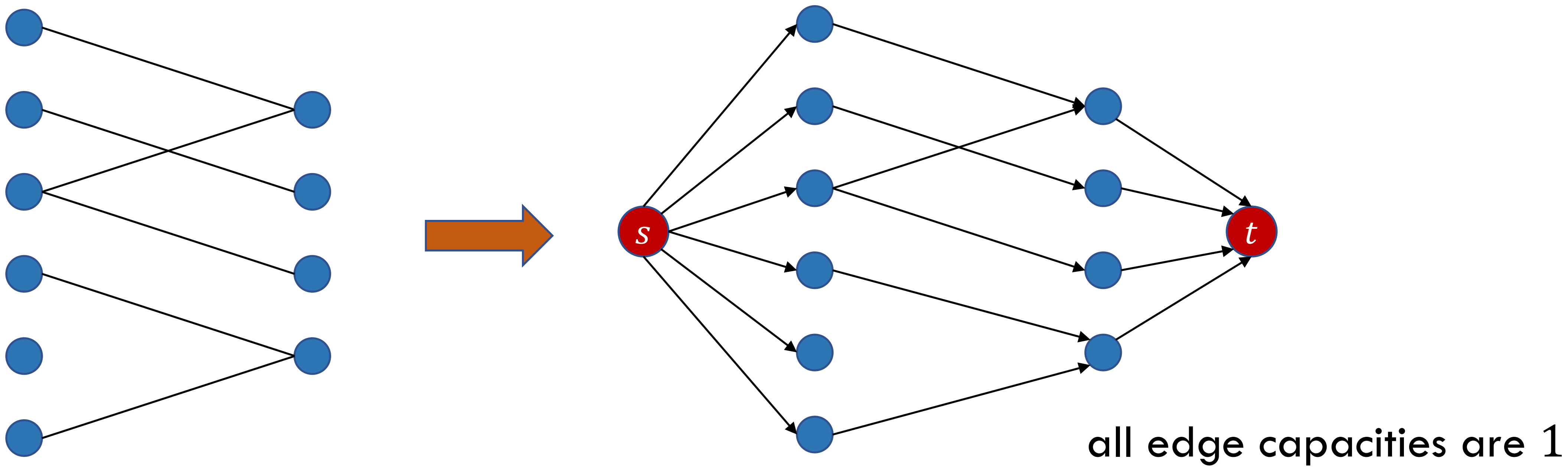
Q: What kind of matching will this algorithm find?





Matching in Bipartite Graphs

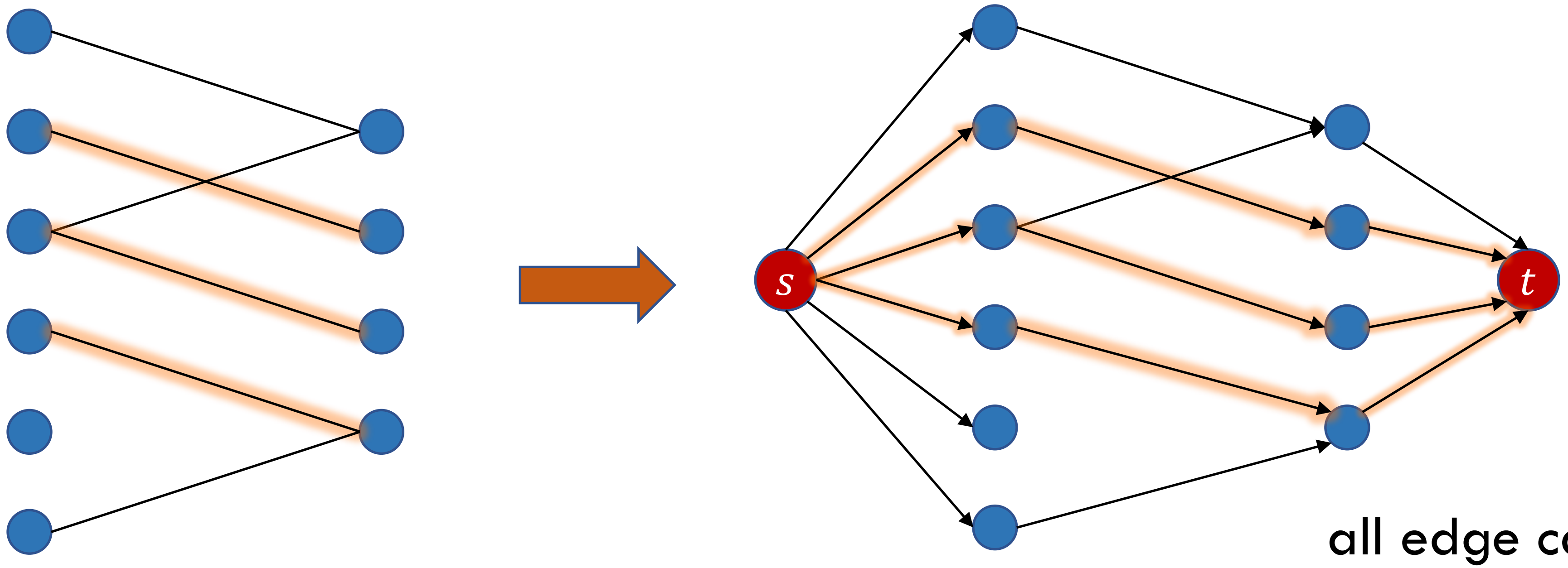
Assume that $G = (L \cup R, E)$ is a bipartite graph. Transform G to an s-t flow network G'. Then the size of the maximum matching in G equals the value of the maximum s-t flow in G'.



Matching in Bipartite Graphs

If M is a matching, then there is a flow f in G' of value $|M|$.

$$val(f^*) \geq \max_{\mathrm{M}} |M|$$



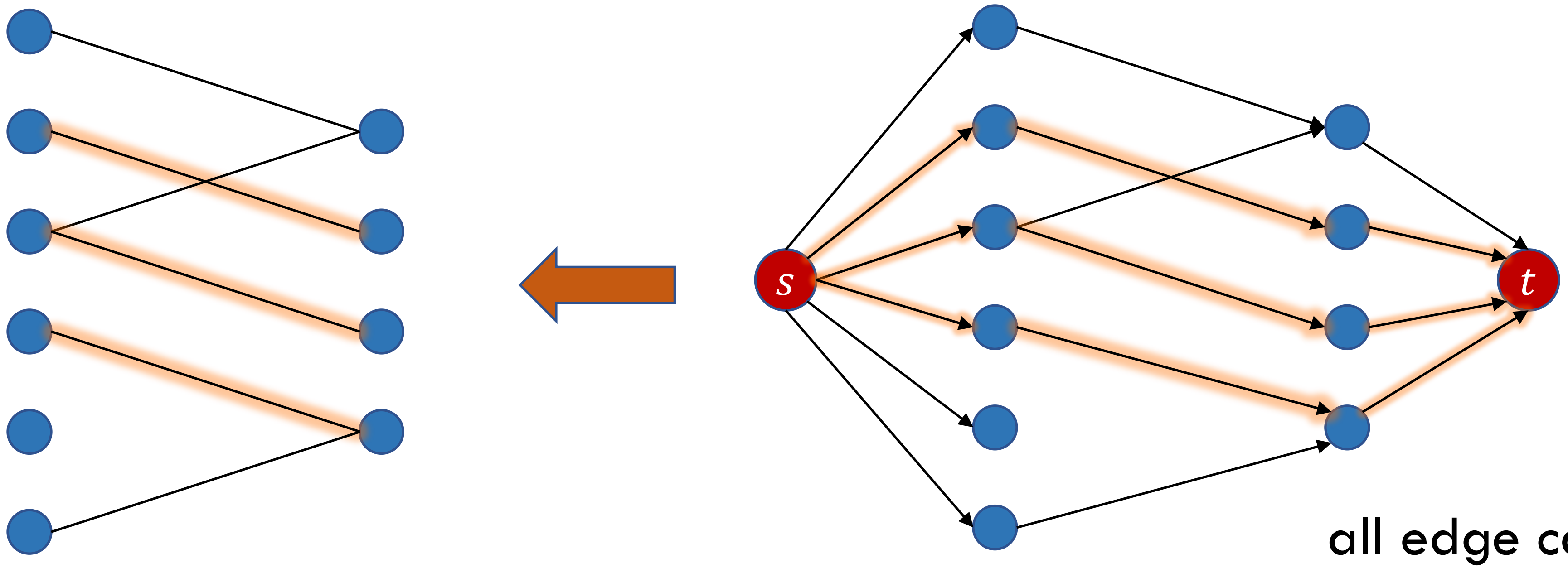
all edge capacities are 1

Matching in Bipartite Graphs

Let f^* be an integral maximum flow in G'. Note $f(e) \in \{0,1\}$ for every $e \in E$.

Let M be the set of edges between L and R used by the flow.

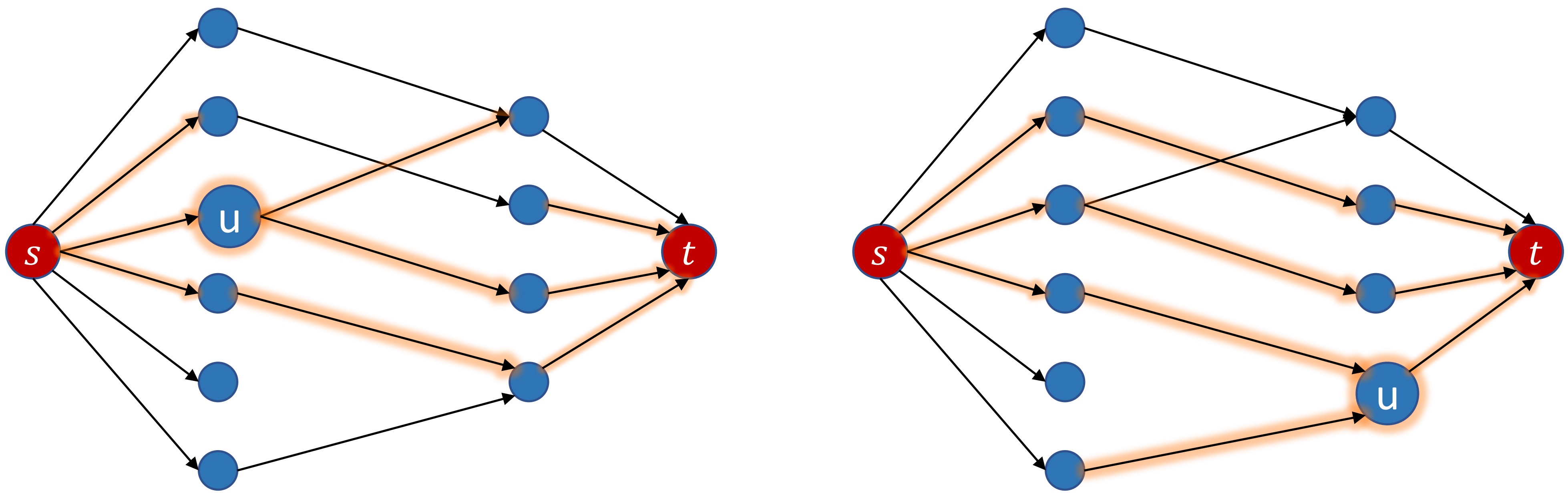
Is M a matching?



all edge capacities are 1

Matching in Bipartite Graphs

Is M a matching? Is it possible that a vertex u in incident to two edges in M?



all edge capacities are 1

Matching in Bipartite Graphs

- Transform the graph into a flow network
- Find an integral maximum flow using Ford-Fulkerson
- Return the corresponding matching

Running time: $O(m\, f^*) = O(mn)$.

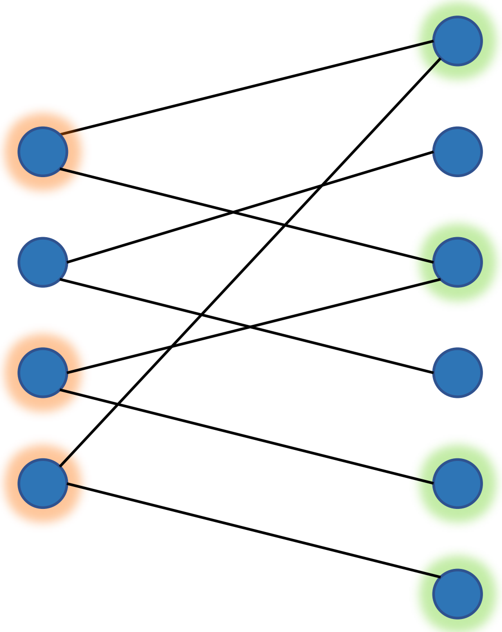
Matching in Bipartite Graphs

- Transform the graph into a flow network
- Find an integral maximum flow using Ford-Fulkerson
- Return the corresponding matching

Running time: $O(m\, f^*) = O(mn)$.

Hall's theorem

Consider a bipartite graph $G = (L \cup R, E)$. For every subset $A \subset L$, let $N(A)$ be the set of neighbors of A in R.



Hall's theorem

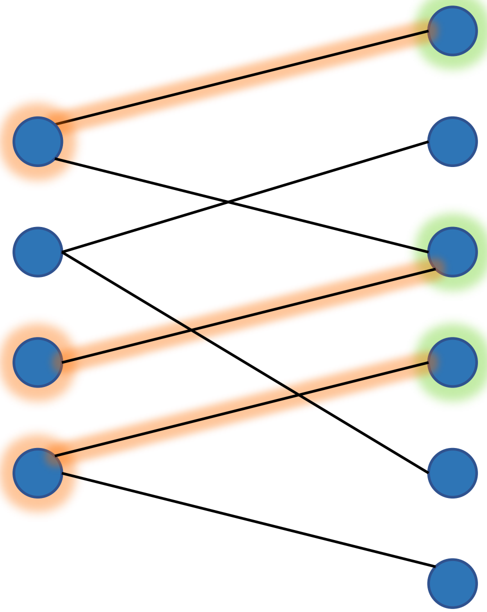
Consider a bipartite graph $G = (L \cup R, E)$. For every subset $A \subset L$, let $N(A)$ be the set of neighbors of A in R.

Assume that there is a matching M of size $|L|$.

I.e., every vertex L is matched.

Then every $A \subset L$ is matched with exactly $|A|$ vertices in R. All of them are neighbors of A.

Thus, $|N(A)| \geq |A|$.



Hall's theorem

Hall's Theorem

There is a matching M of size $|L|$ if and only if

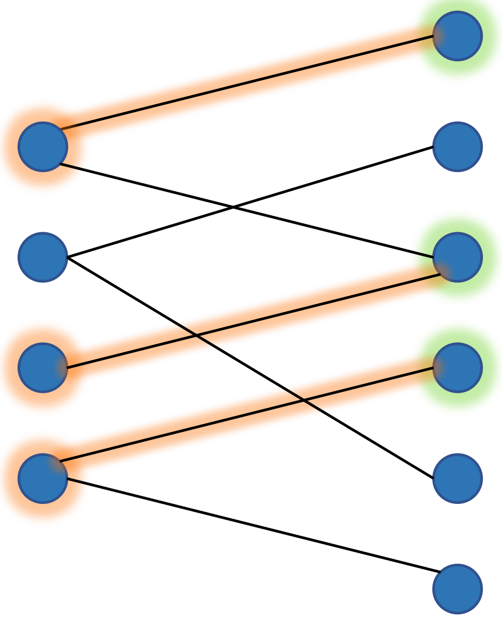
$$|N(A)| \geq |A|$$

for every $A \subset L$.

Proof:

We showed that "$\Rightarrow$" holds.

Now we prove "$\Leftarrow$".

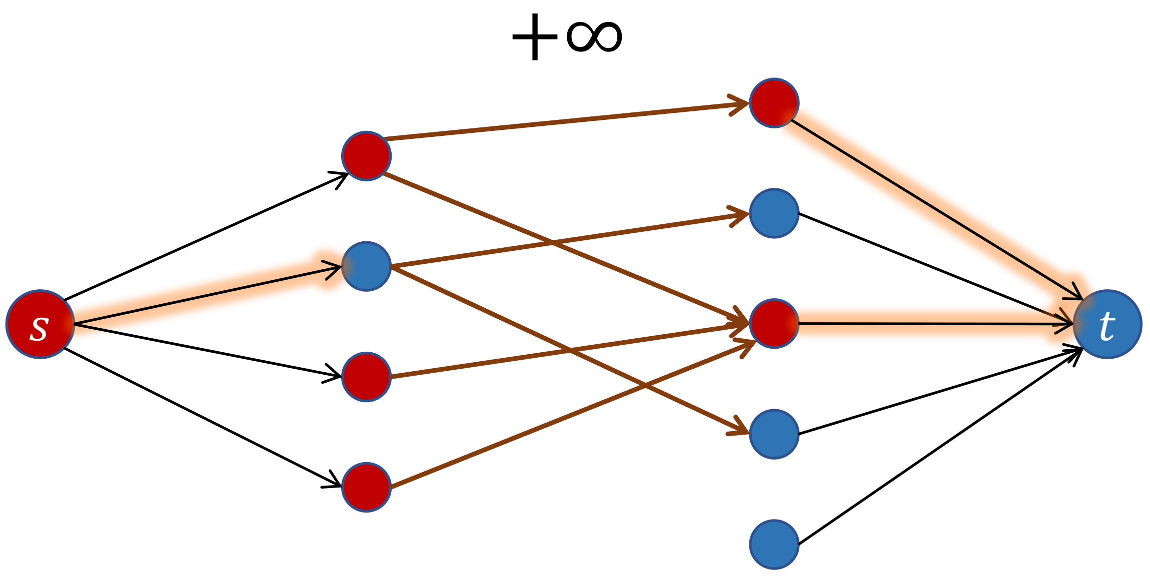


Hall's theorem

Assume that $|M| < |L|$ for a maximum matching M

We will show that $|N(A)| < |A|$ for some set $A \subset L$.

Max Flow / Min Cut Thm $\Rightarrow$ There is a cut (P, Q) of size $|M| < |A|$ in G'.



What edges are cut?

- edges from s to $Q \cap L$
- edges from $P \cap R$ to t

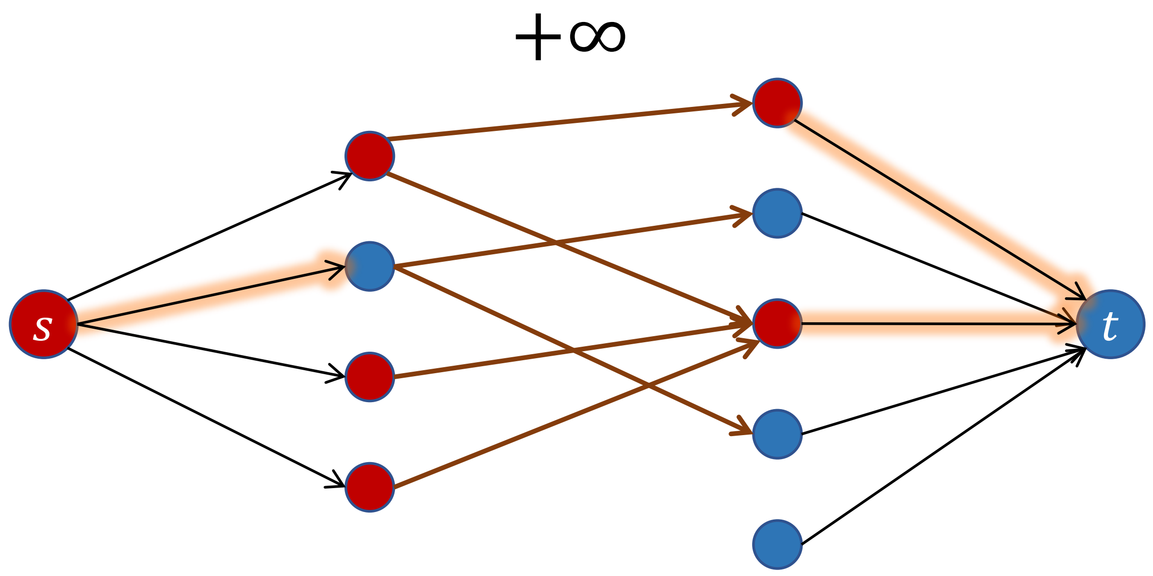
? edges from $P \cap L$ to $Q \cap R$

Hall's theorem

Thus, $|\mathrm{L}| > cap(P, Q) = |Q \cap L| + |P \cap R|$.

That is,

(#blue vertices on the left) + (#red vertices on the right) $< |L|$



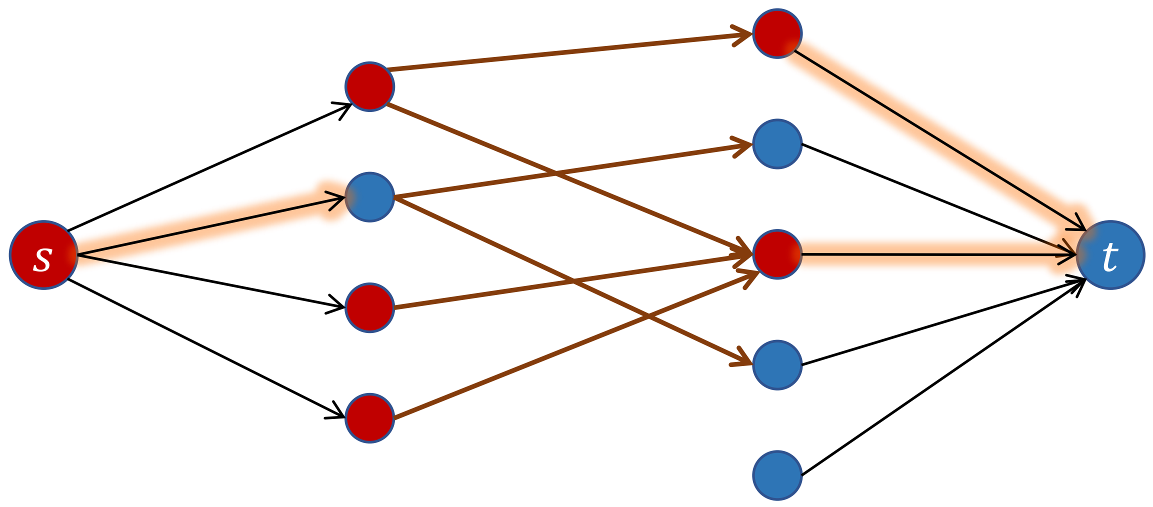
Hall's theorem

Thus, $|\mathrm{L}| > cap(P, Q) = |Q \cap L| + |P \cap R|$.

That is,

(#blue vertices on the left) + (#red vertices on the right) $< |L|$

(#red vertices on the right) < (#red vertices on the left)

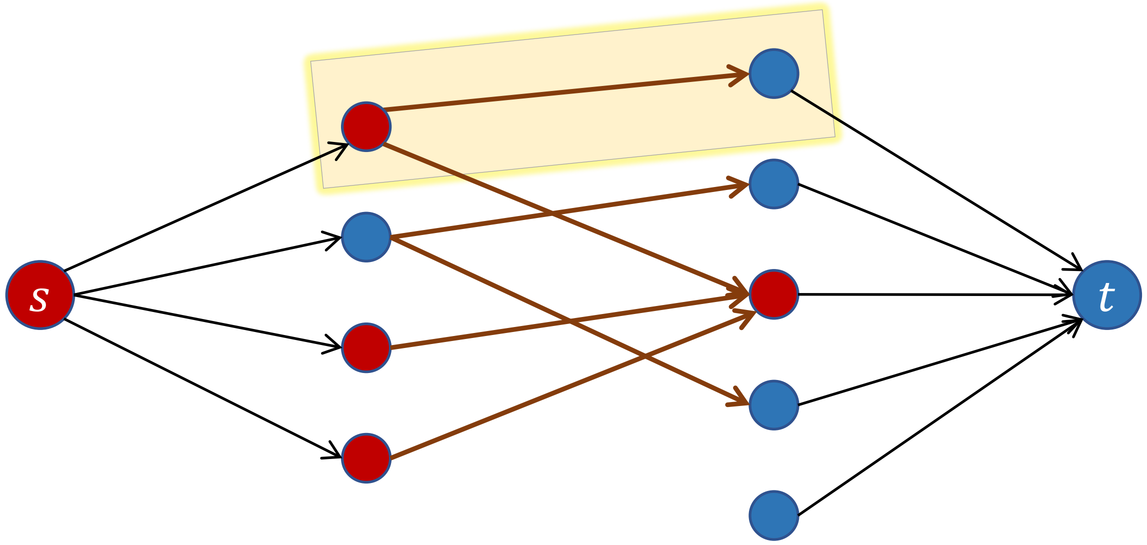


Hall's theorem

Let $A = P \cap L$ (red vertices on the left). What is the size of $N(A)$?

All vertices in $\mathrm{N}(A)$ are on the right.

Q: Can a vertex in $N(A)$ be blue?



(#red vertices on the right)
$<$ (#red vertices on the left)

Hall's theorem

Let $A = P \cap L$ (red vertices on the left). What is the size of $N(A)$?

$$\begin{aligned} A \quad &= P \cap L = \{\text{red vertices on the left}\} \\ N(A) &\subseteq P \cap R = \{\text{red vertices on the right}\} \end{aligned}$$

$$|N(A)| < |A|$$

$\blacksquare$

Hall's theorem

Corollary

Let G be a bipartite graph with $|L| = |R|$.

There exists a perfect matching in G

if and only if

$|N(A)| \geq |A|$ for every $A \subset L$

Assignment Problems

Basic Assignment Problem

There are m people and $n < m$ jobs.

- For every job i, we are given the list of people S_i that can perform it.
- We may assign only one to each person.

Find an assignment of jobs to people (if it exists).

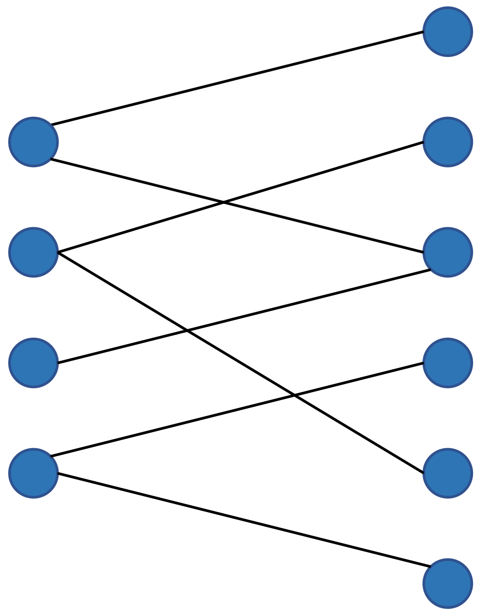
Solution:

Consider a bipartite graph with

$L = \{1, \ldots, m\}$ representing jobs and

$R = \{1, \ldots, n\}$ representing people

connect job i with people $j \in S_i$



Job Assignment Problem

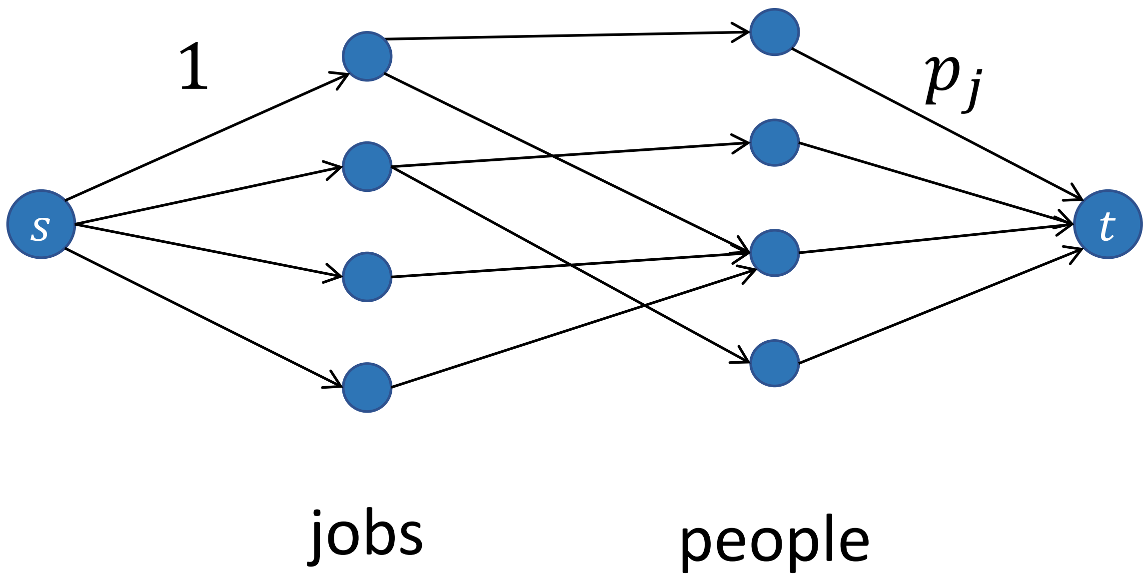
There are m people and m jobs.

- For every job i, we are given the list of people S_i that can perform it.
- We may assign p_j jobs to person j.

Find an assignment of jobs to people subject to the constraints above so as to maximize the number of assigned jobs.

Q: Suggestions?

Job Assignment Problem



Project Scheduling

There are n projects: $1, \dots, n$

We are given a set of dependencies between projects of the following form

in order to complete project i we need to complete project j first

E.g.,

1 depends on 3, 2 depends on 3 and 4, 4 depends on 6, etc

A feasible schedule A is a subset of projects s.t. if $i \in A$ then all projects j that i depends on are also in A.

Project Scheduling

Some projects are profitable and some are not.

For each project i, we are given p_i.

- If $p_i \geq 0$, project i has profit p_i.
- If $p_i < 0$, project i has cost $-p_i$.

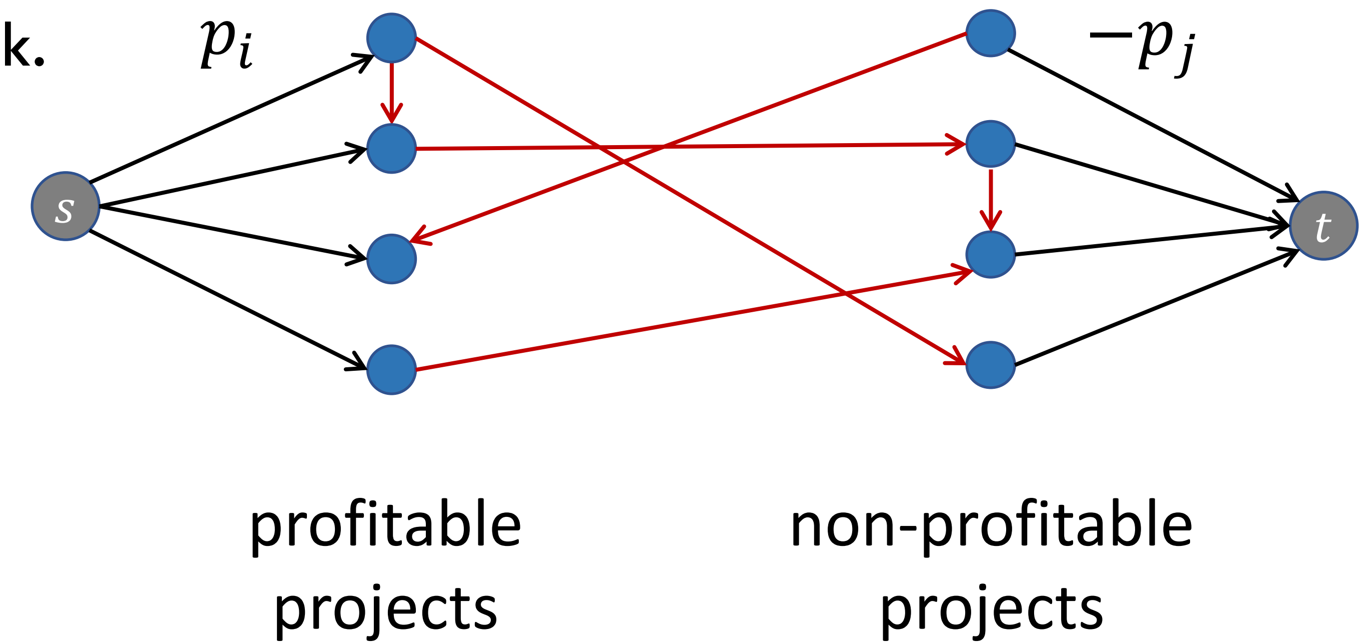
Out goal is to find a feasible schedule A that maximizes our profit

$$\sum_{i \in A} p_i$$

Project Scheduling

Construct the following s-t network.

If i depends on j, we have an edge (i, j) of ∞ capacity.



Observation:

S is a feasible schedule

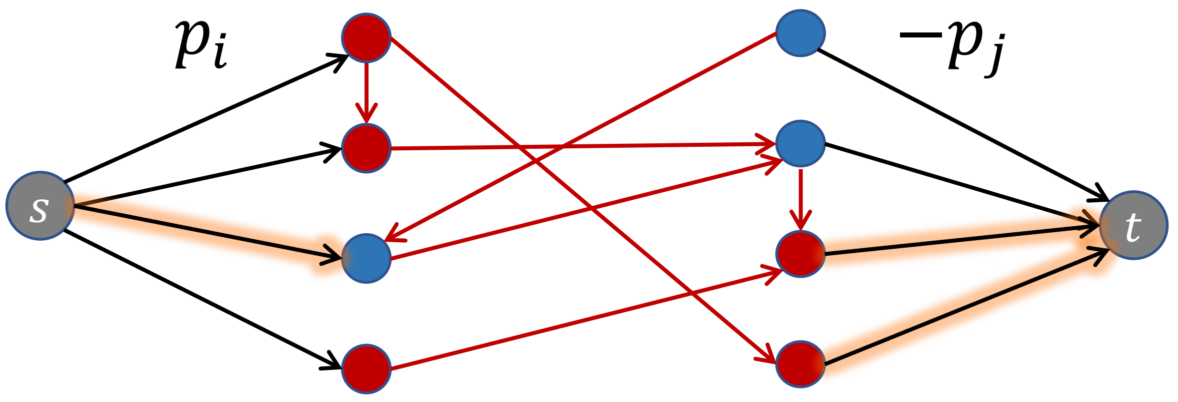
$\Updownarrow$

cut $(A \cup \{s\}, B \cup \{t\})$ has finite cost where $B = \{1, \ldots, n\} \setminus A$

Project Scheduling

There is a one-to-one correspondence between cuts of finite capacity and feasible schedules!

How are their costs & profits related?



$cap(A \cup \{s\}, B \cup \{t\})$ **equals**

the profit of non-scheduled jobs on the left

$+$

the cost of scheduled jobs on the right

Project Scheduling

$\boldsymbol{cap(A \cup \{s\}, B \cup \{t\})}$ **equals**

the profit of non-scheduled jobs on the left

$$+$$

the cost of scheduled jobs on the right

$$cap(A \cup \{s\}, B \cup \{t\}) = \sum_{\substack{i \notin A \\ p_i \geq 0}} p_i + \sum_{\substack{i \in A \\ p_i < 0}} (-p_i) = \left(\sum_{p_i \geq 0} p_i \right) - \sum_{i \in A} p_i$$

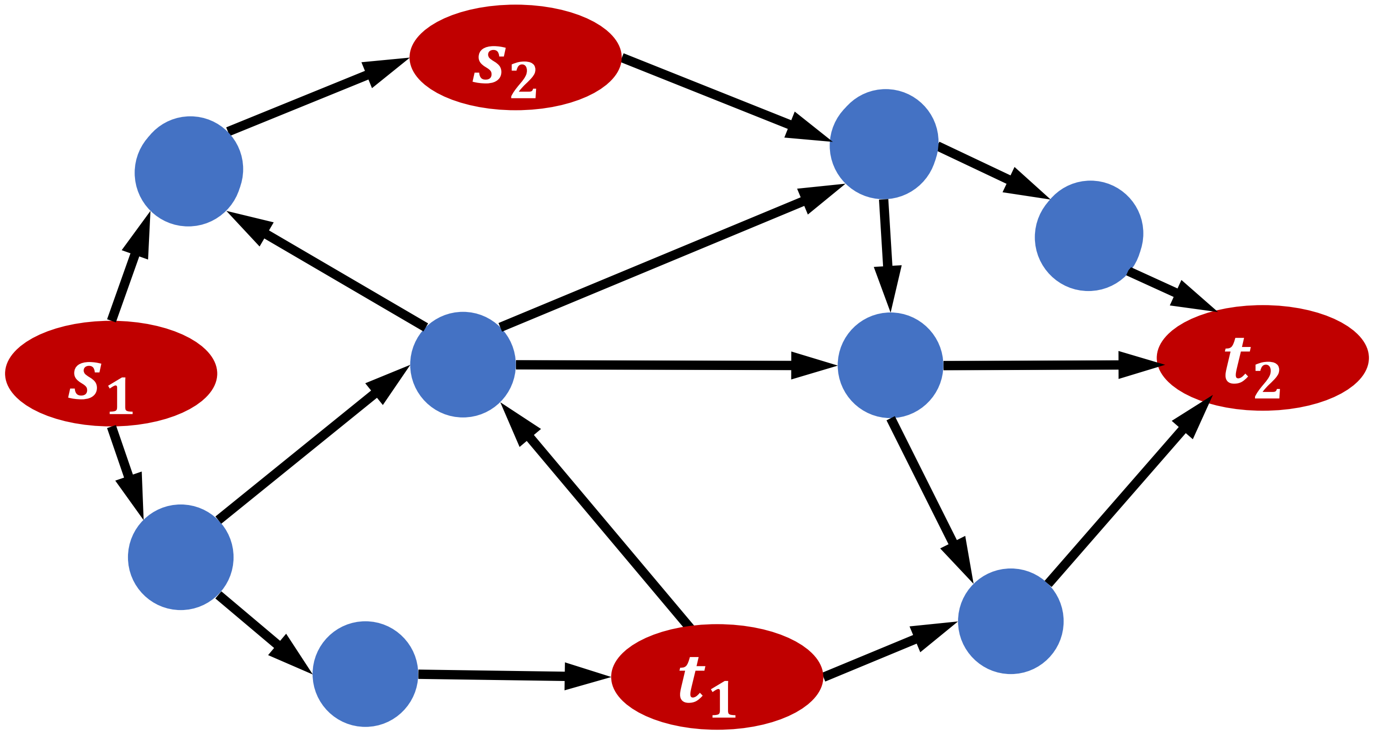
Q: How can we solve the problem now?

Multicommodity Flow

Multiple Sources & Sinks

Given: Flow network w/ many sources: $s_1, \dots, s_{k'}$ and
many sinks: $t_1, \dots, t_{k''}$.

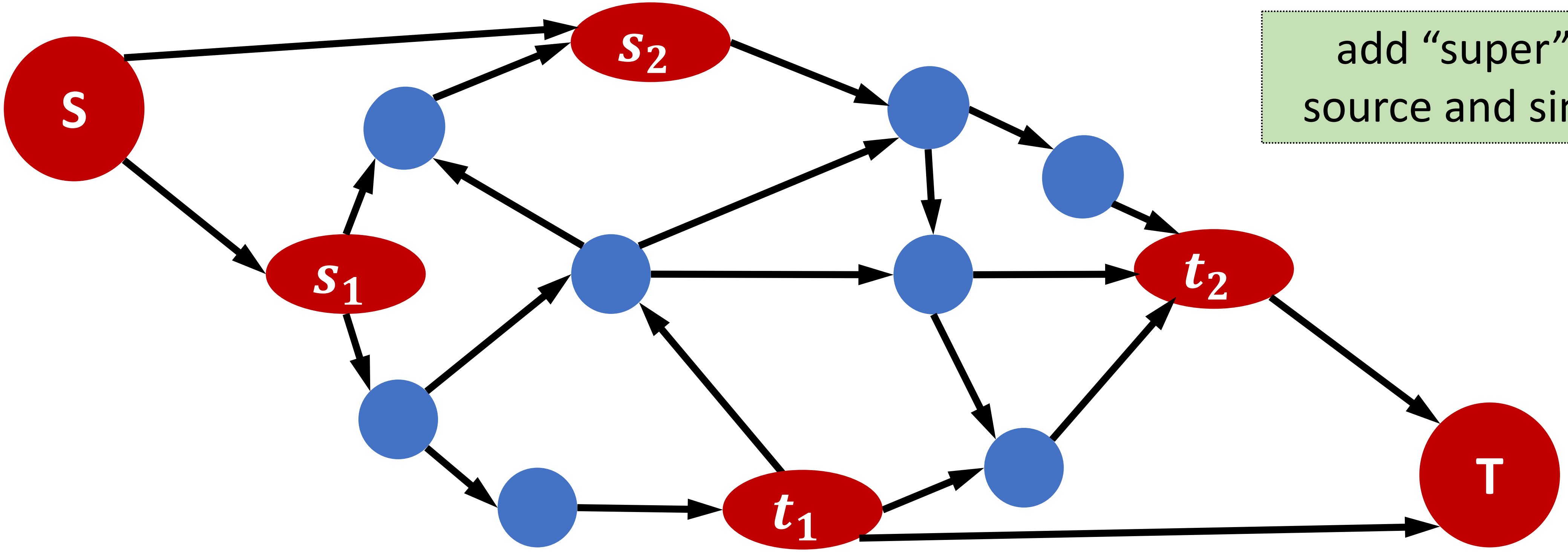
Goal: Maximize the total amount of flow from all sources to all sinks.



Multiple Sources & Sinks

Given: Flow network w/ many sources: $s_1, \dots, s_{k'}$ and
many sinks: $t_1, \dots, t_{k''}$.

Goal: Maximize the total amount of flow from all sources to all sinks.



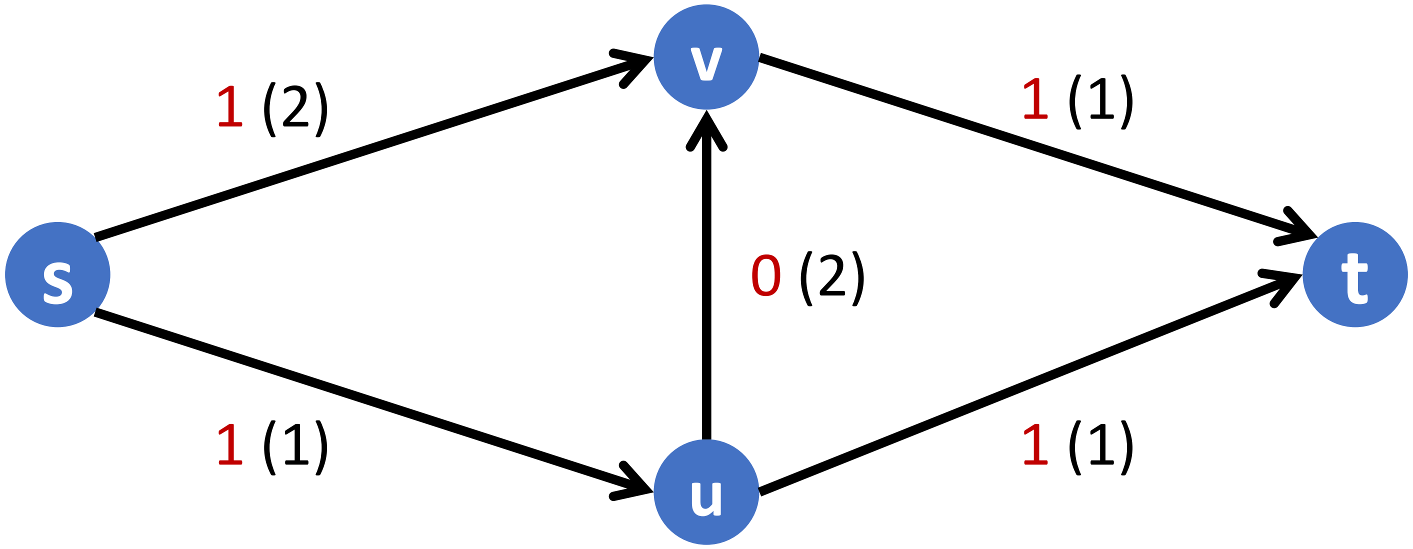
add "super" source and sink

Flow-Path Decomposition

Flow-Path Decomposition

Imagine that we are solving a truck routing problem.

We want to find a route for each truck.

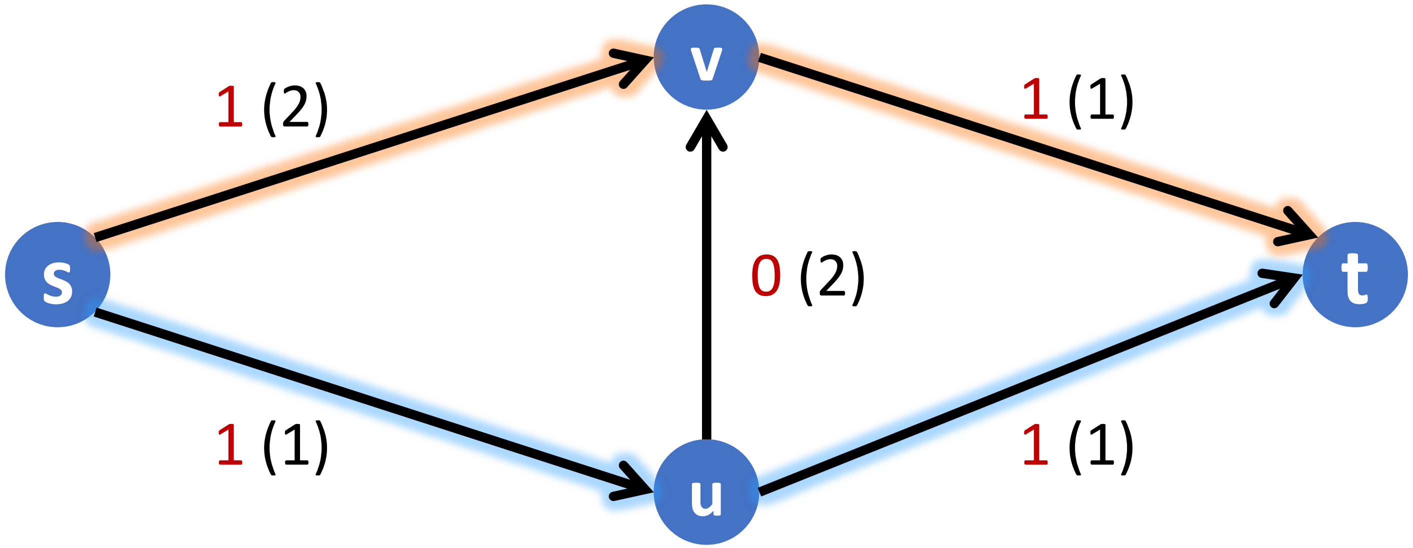


Flow-Path Decomposition

Imagine that we are solving a truck routing problem.

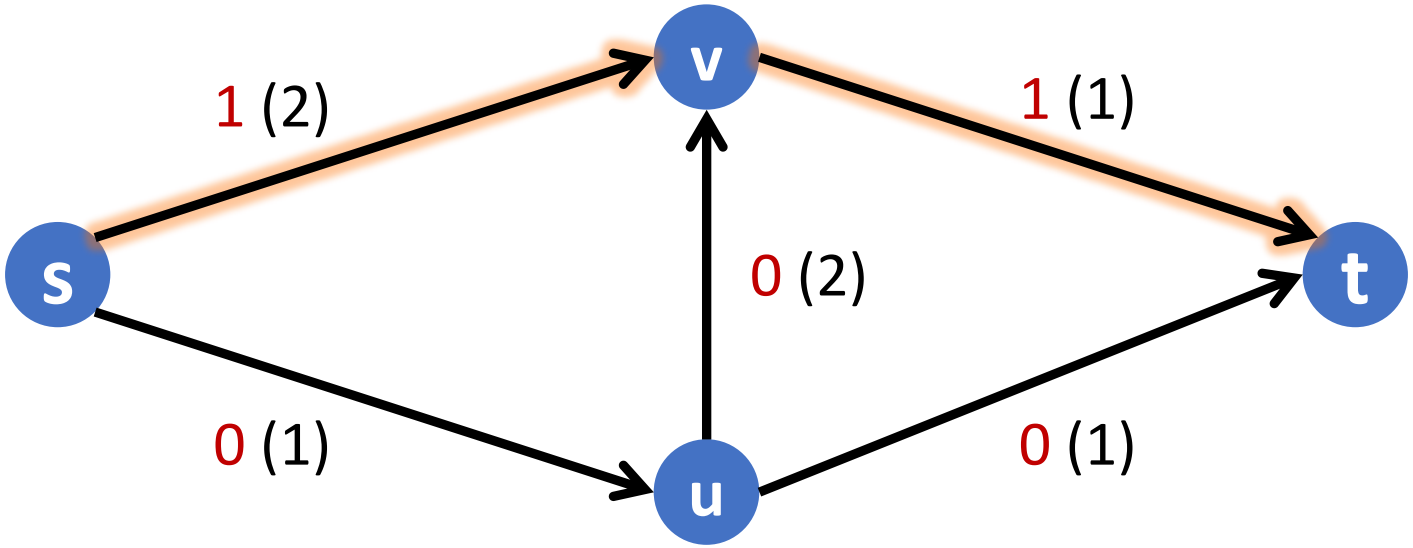
We want to find a route for each truck.

In this example we have 2 routes.



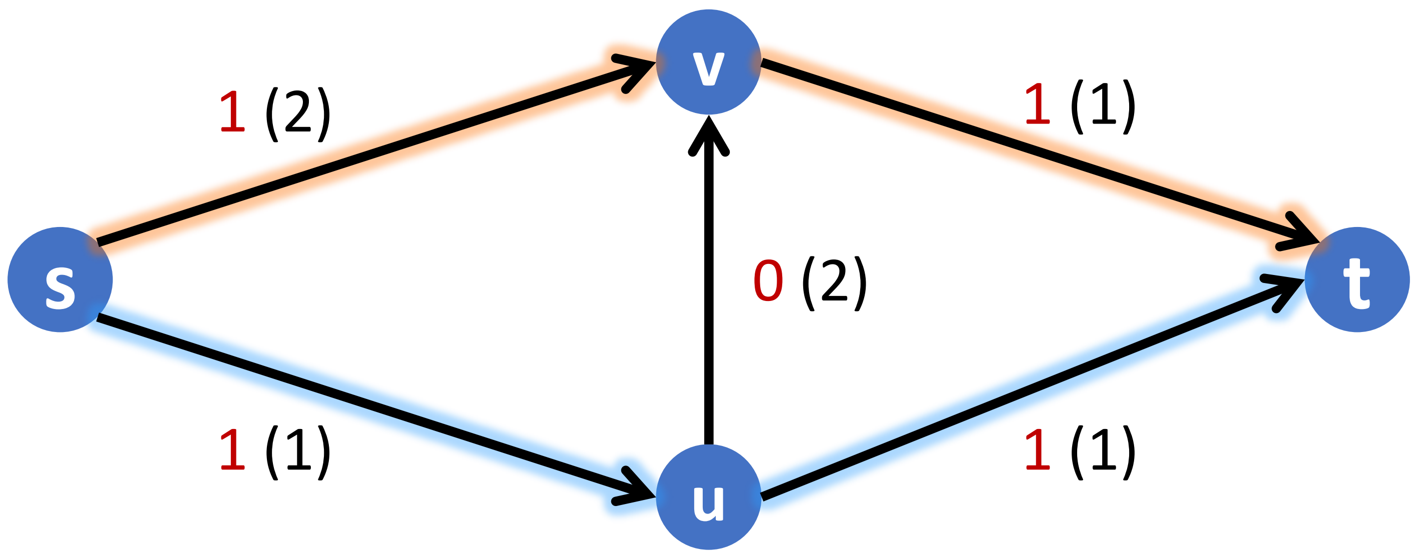
Flow-Path Decomposition

A flow f_i is a path flow if it uses a single s-t path.



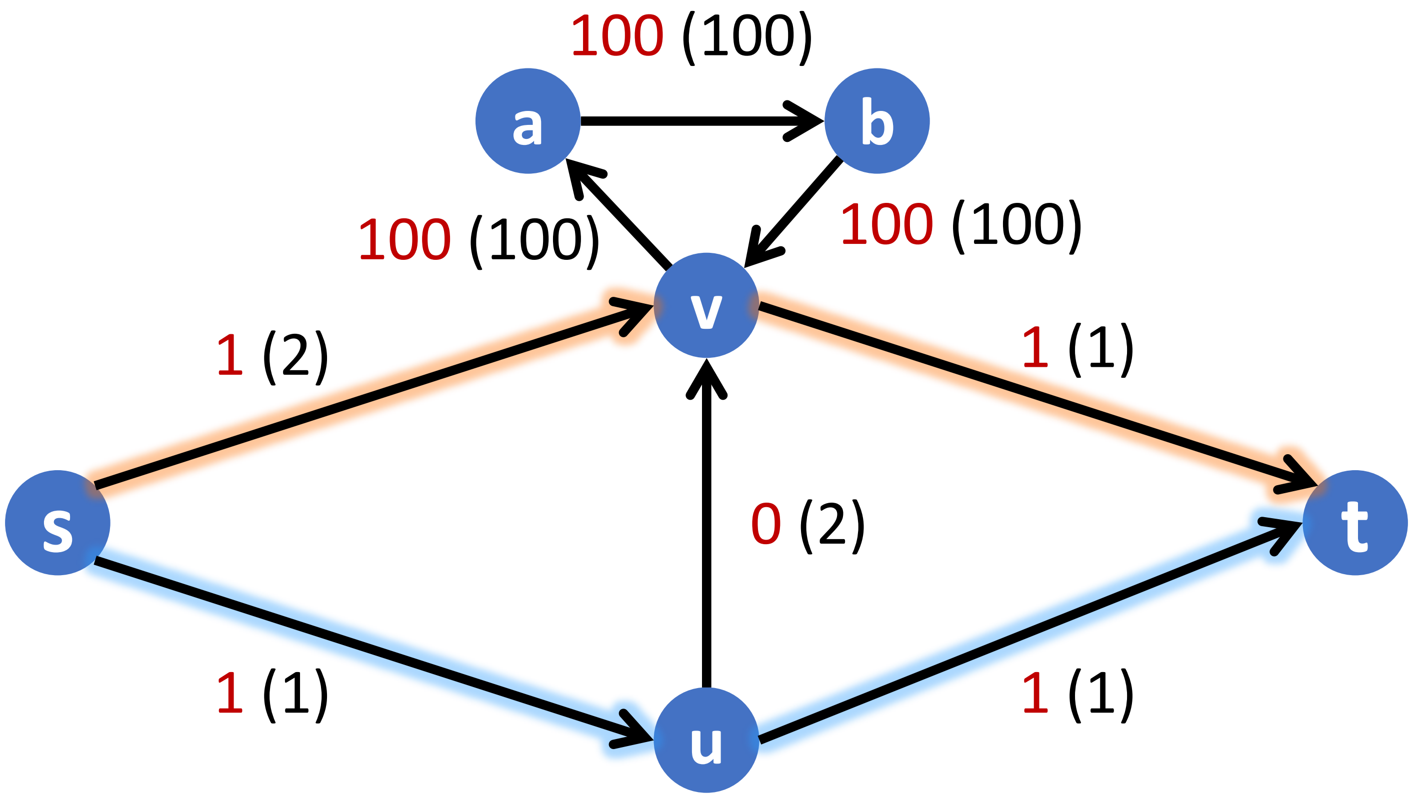
Flow-Path Decomposition

This flow is a sum of two path flows f_1 and f_2 of value 1 each.



Flow-Path Decomposition

Q: Can we represent every flow as a sum of path flows?



Flow-Path Decomposition

A flow f_i is a path flow if it uses a single s-t path.

A flow g_j is a cycle flow if it uses a single cycle.

Theorem

For every flow f there exist path flows $f_1, \ldots, f_a$ and cycle flows $g_1, \ldots, g_b$ s.t.

$$f(e) = \sum_i f_i(e) + \sum_j g_j(b)$$

Note $val(f) = \sum_i val\big(f_i(e)\big)$. If we remove the cyclic component, we will not change the value of the flow.

Edge and Vertex Disjoint Paths

Edge Disjoint Paths: **Menger's Theorem**

Consider a directed or undirected graph G and two vertices s and t.

We say that s-t paths $P_1, \ldots, P_k$ are edge disjoint if no two of them share a common edge.

Q: What is the maximum number of edge disjoint paths between s and t?

A: It is equal to the size of the minimum cut between s and t (in which all edges have capacity 1).

Proof: use path flow decomposition.

Vertex Disjoint Paths:
Menger's Theorem

Consider a directed or undirected graph G and two vertices s and t.

We say that s-t paths $P_1, \dots, P_k$ are vertex disjoint if no two of them share a common edge.

Q: What is the maximum number of vertex disjoint paths between s and t?

A: It is equal to the size of the minimum vertex separator between s and t.

Exercise: Prove this.