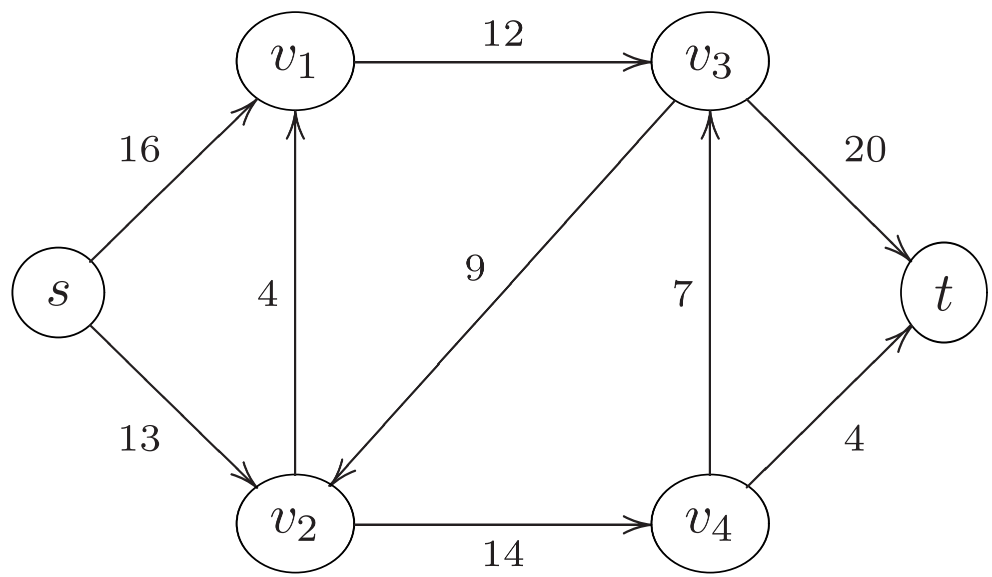
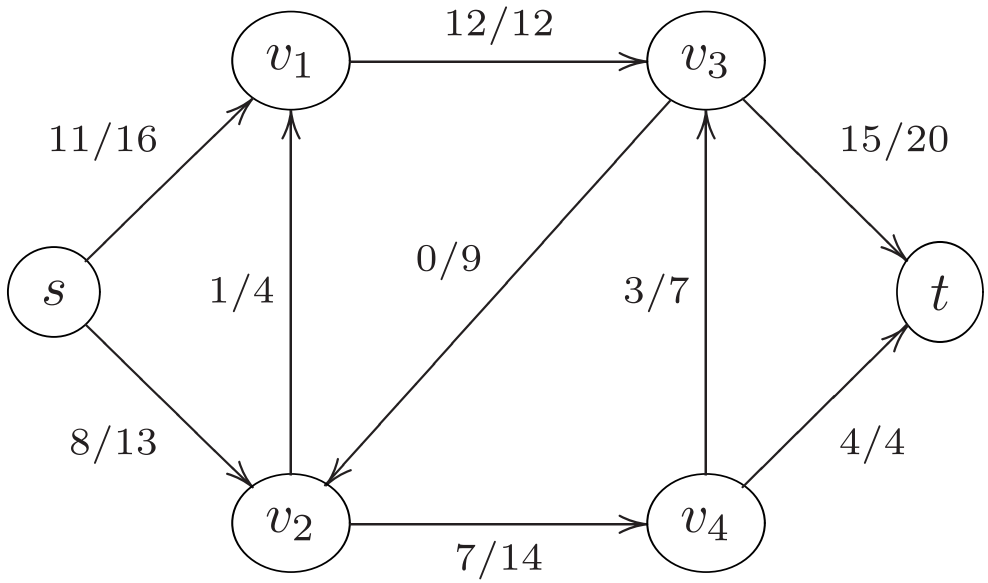
Maximum Flow

Flow Network



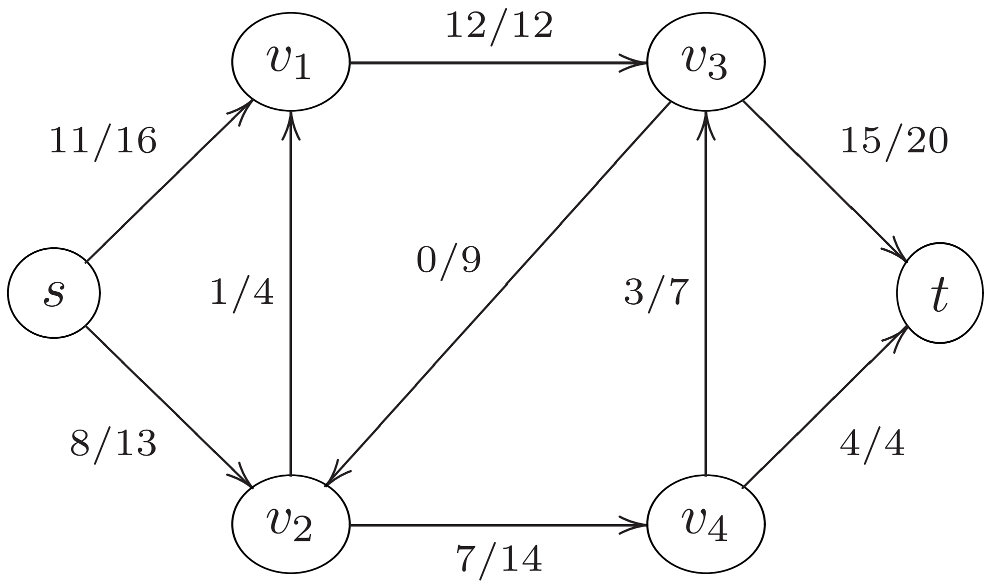
$c(u, v)$ = capacity of edge (u, v).

Flow



$c(u, v)$ = capacity of edge (u, v).
$f(u, v)$ = flow along edge (u, v).

Capacity Constraint

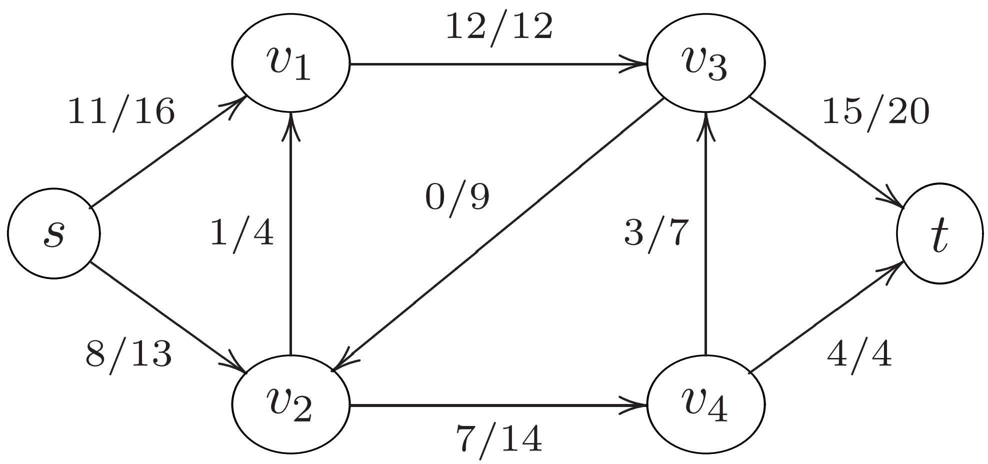


Capacity constraint: $0 \le f(u,v) \le c(u,v)$.

$c(u,v)$ = capacity of edge (u,v).

$f(u,v)$ = flow along edge (u,v).

Flow Conservation



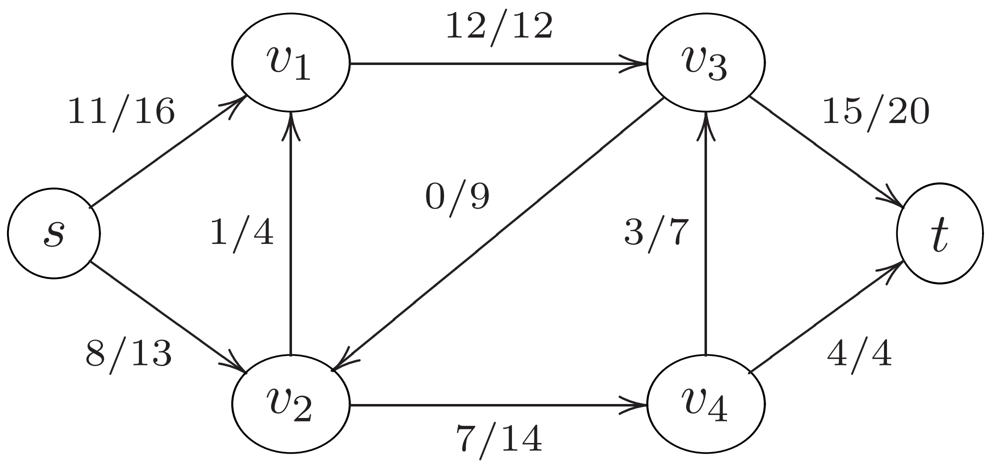
Flow conservation: For all $u \in G.V - \{s, t\}$,

$$\text{flow in to } u = \text{flow out from } u, \text{ or}$$

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v).$$

$f(u, v)$ = flow along edge (u, v).

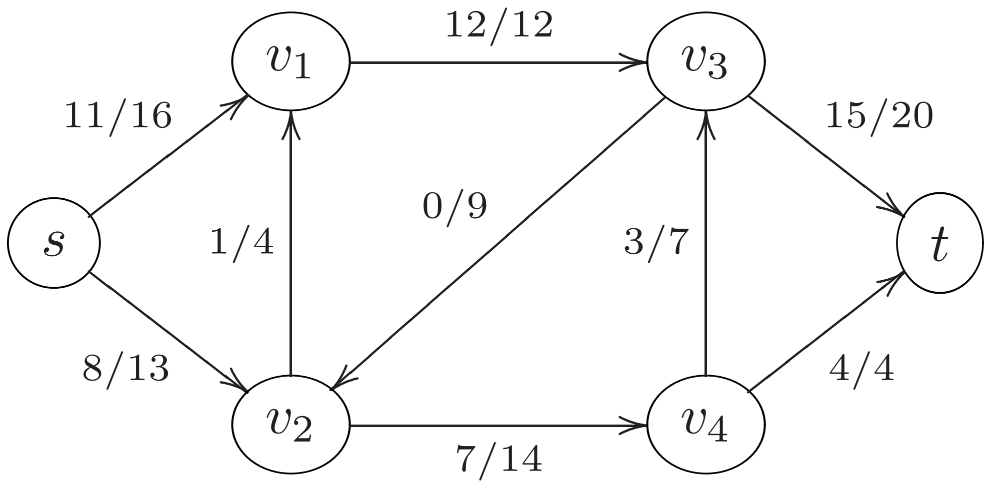
Flow



A flow is a function $f : G.E \to \mathbb{R}$ where

1. Capacity constraint: $0 \leq f(u, v) \leq c(u, v)$;
2. Flow conservation: $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.

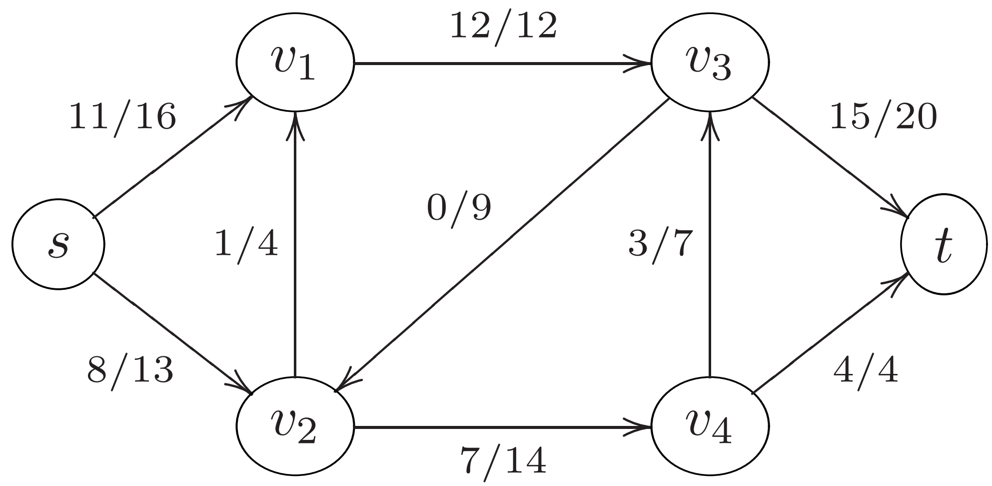
Flow



The value of the flow, $|f|$, is the amount flowing out of node s:

$$|f| = \sum_{v \in G.V} f(s, v).$$

Flow

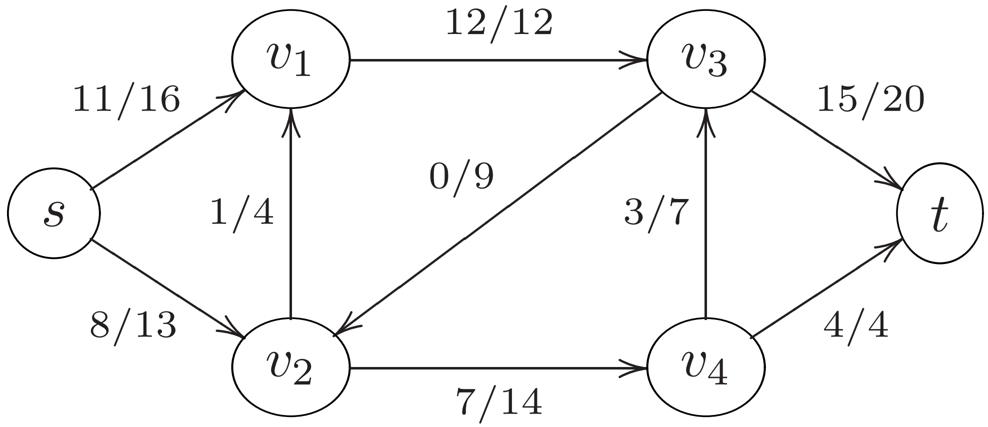


Lemma:

flow out of node s = flow in to node t, or

$$|f| = \sum_{v \in G.V} f(s, v) = \sum_{v \in G.V} f(v, t).$$

Max Flow Problem

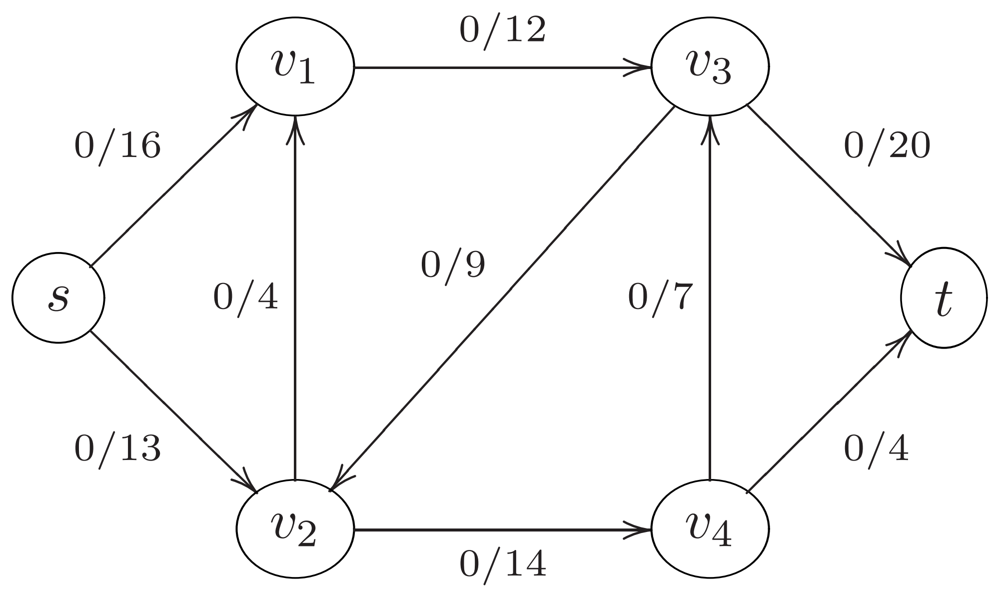


The value of the flow, $|f|$, is the amount flowing out of node s:

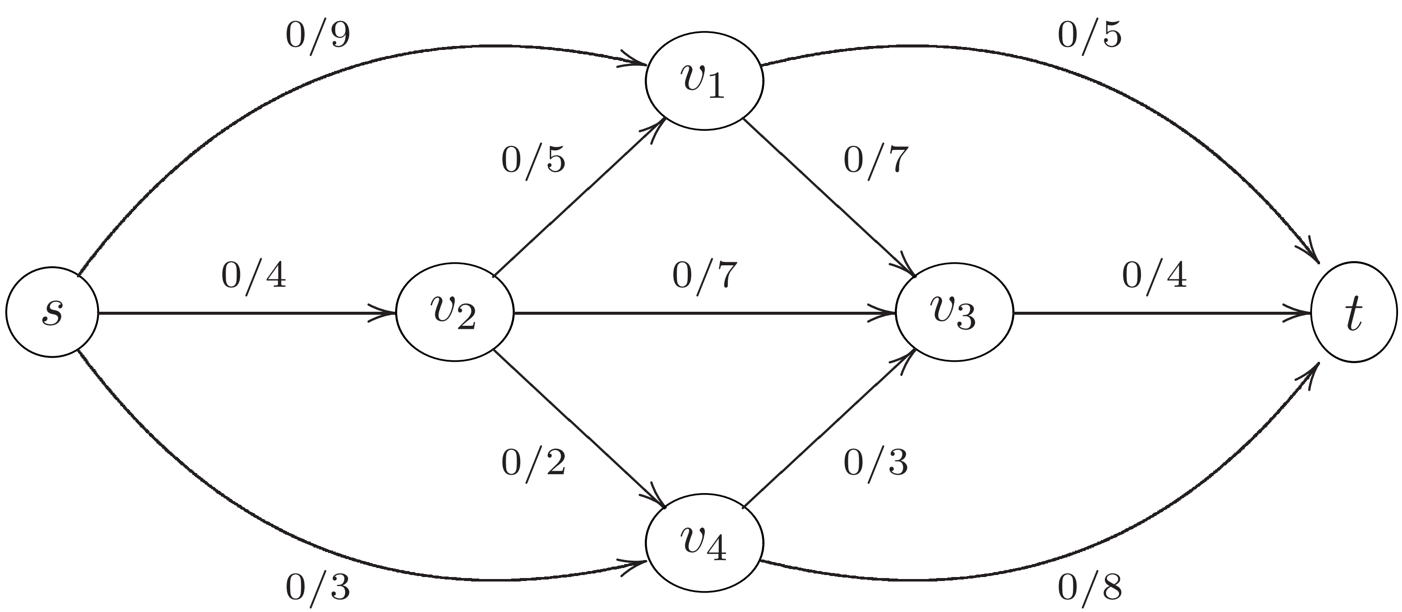
$$|f| = \sum_{v \in G.V} f(s, v).$$

max flow problem: Given a flow network G, find $\max |f|$ over all flows f in G.

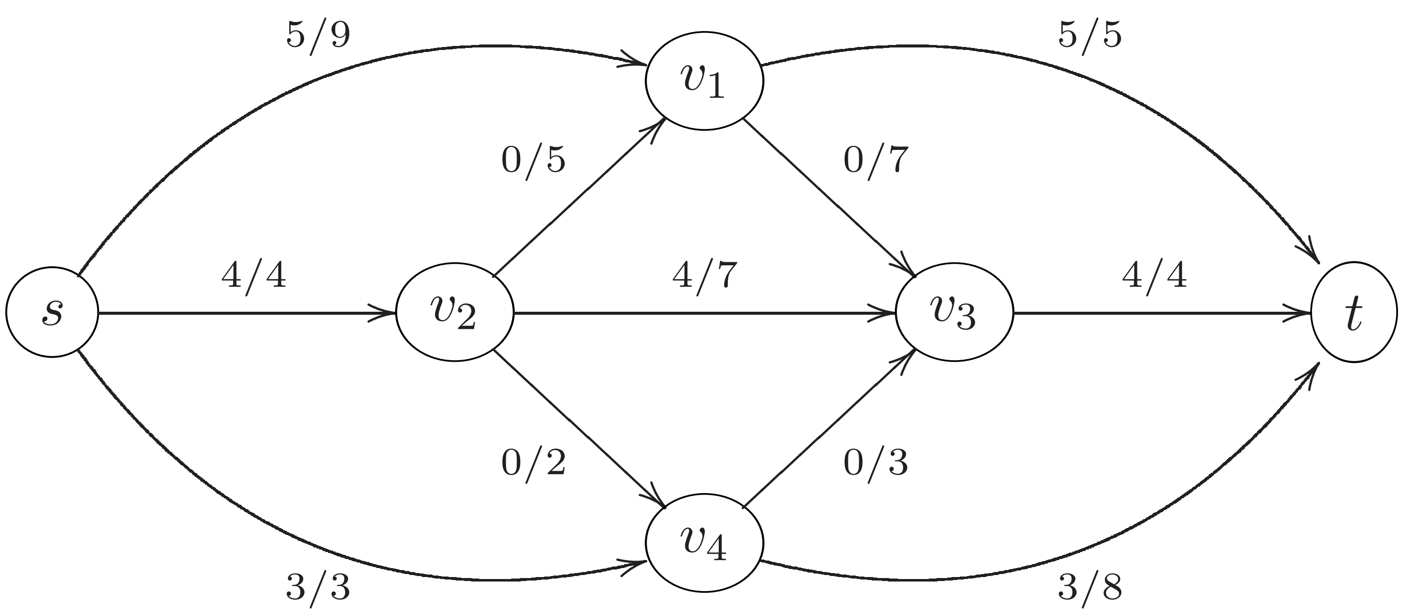
Flow



Flow: Example 2

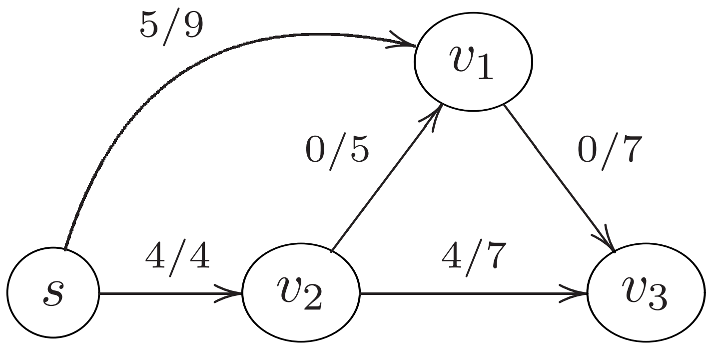


Flow: Example 2

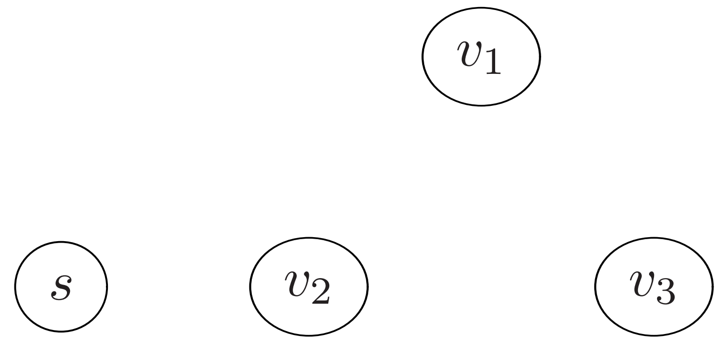


Residual Capacity

Flow network (part):



Residual capacity:

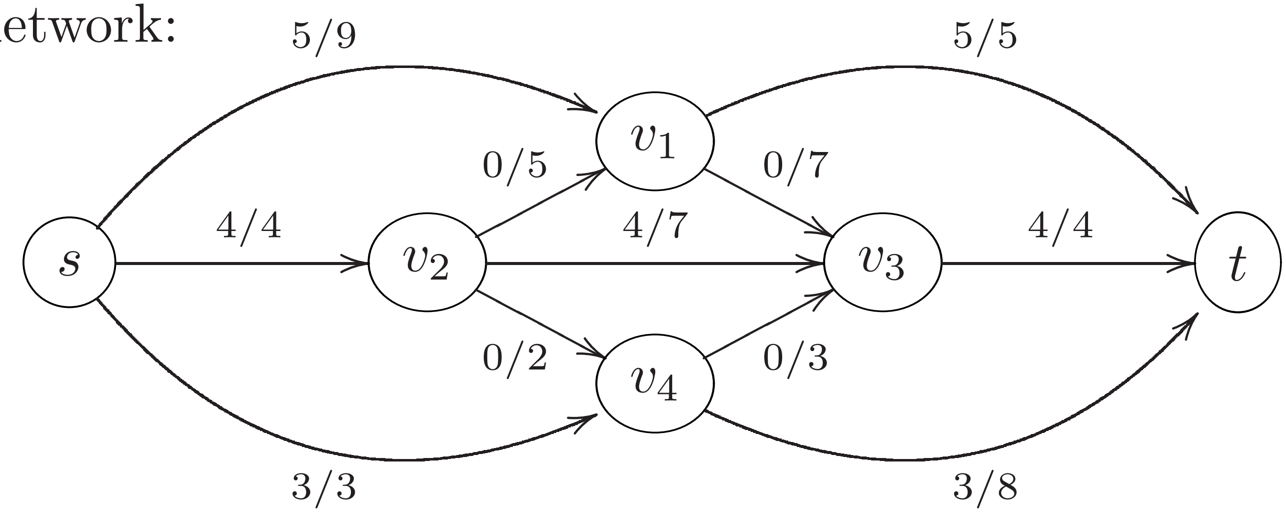


Residual capacity:

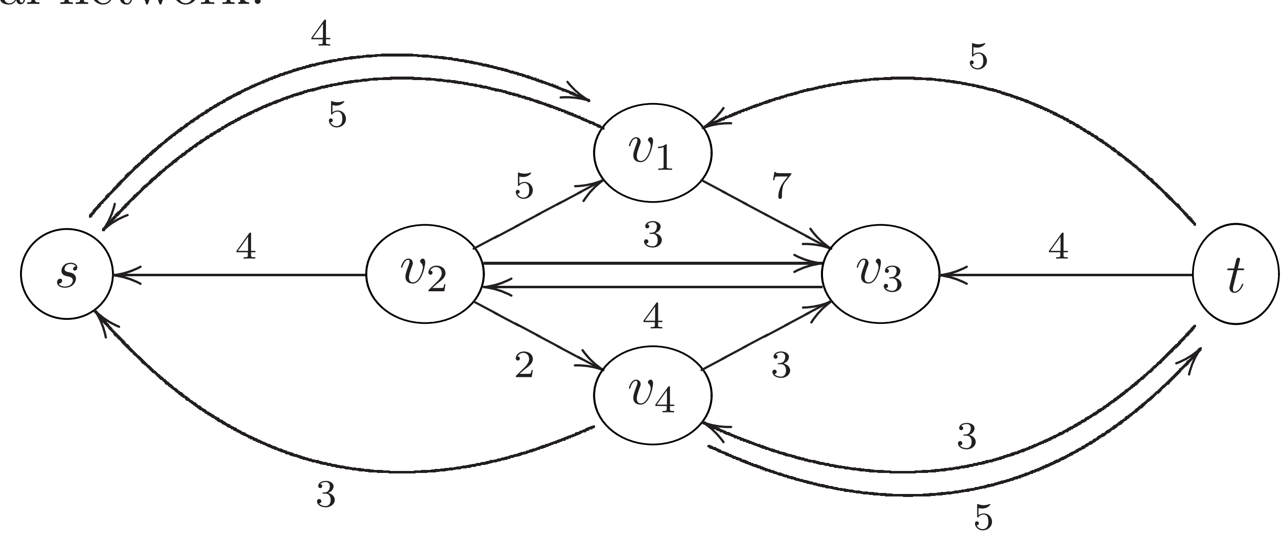
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in G.E, \\ f(v,u) & \text{if } (v,u) \in G.E, \\ 0 & \text{otherwise.} \end{cases}$$

Residual Network

Flow network:

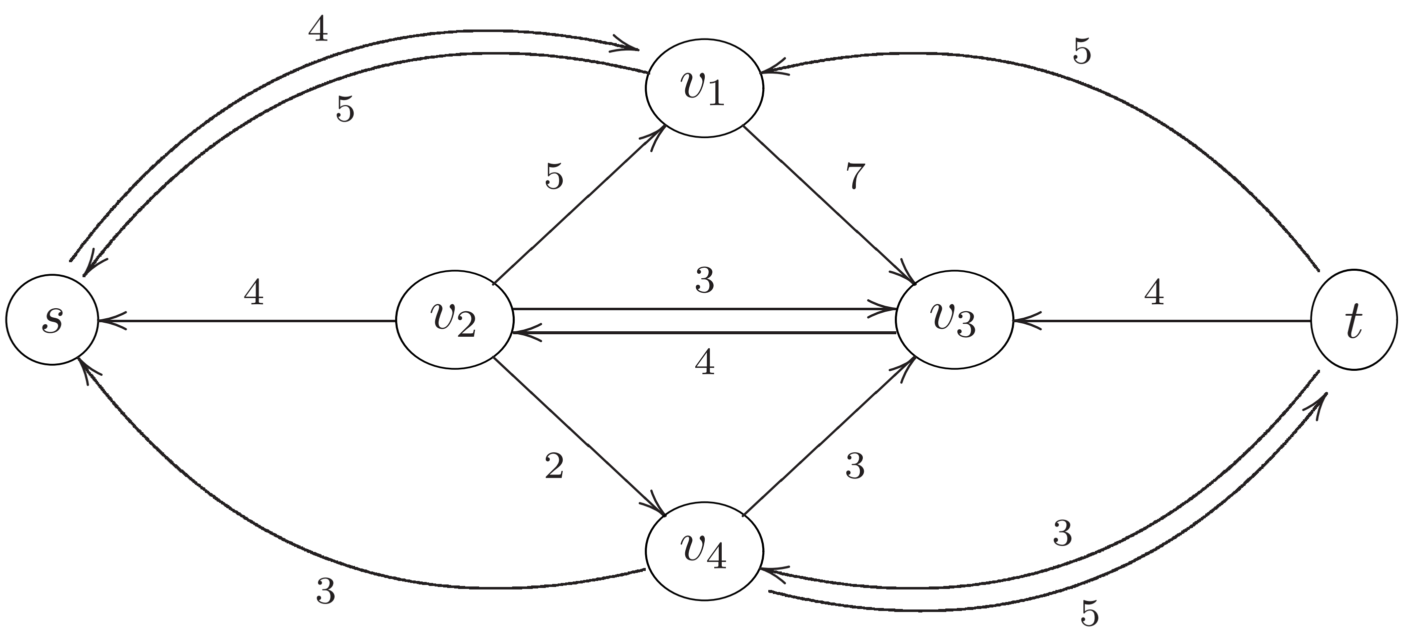


Residual network:



Augmenting Path

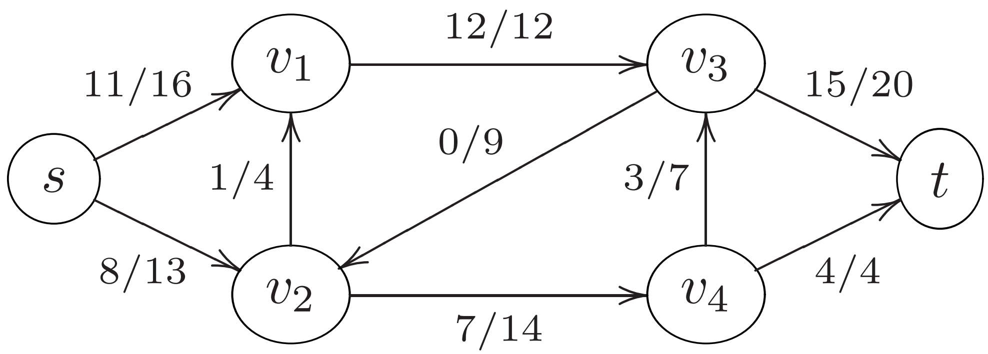
Residual network:



An **augmenting path** is a path from s to t in the residual network.

Residual Network

Flow network:

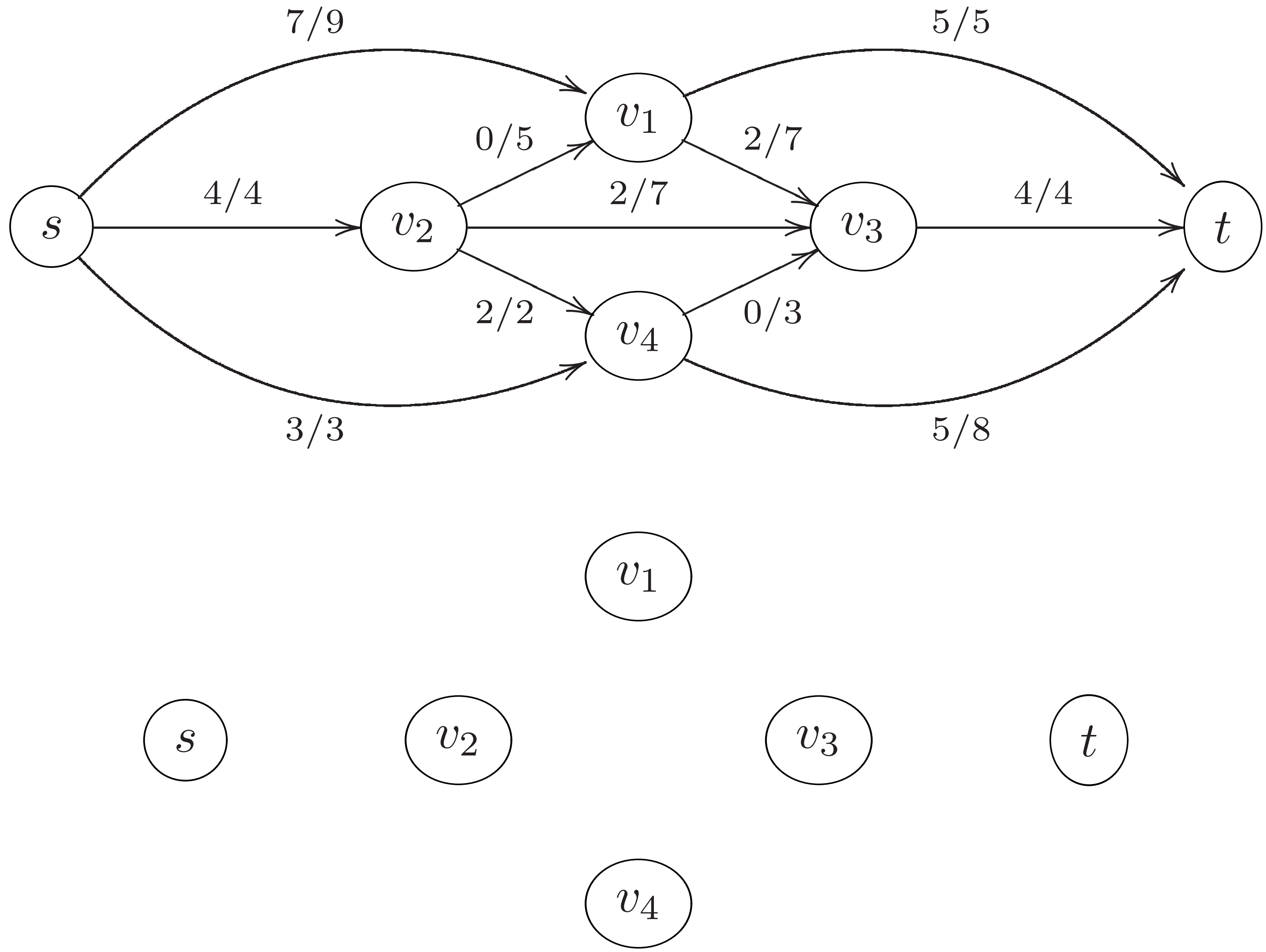


Residual capacity:

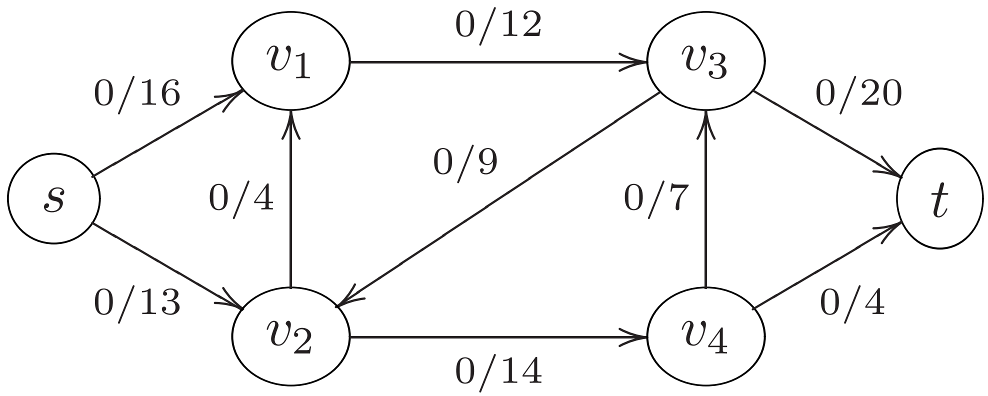
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in G.E, \\ f(v,u) & \text{if } (v,u) \in G.E, \\ 0 & \text{otherwise.} \end{cases}$$

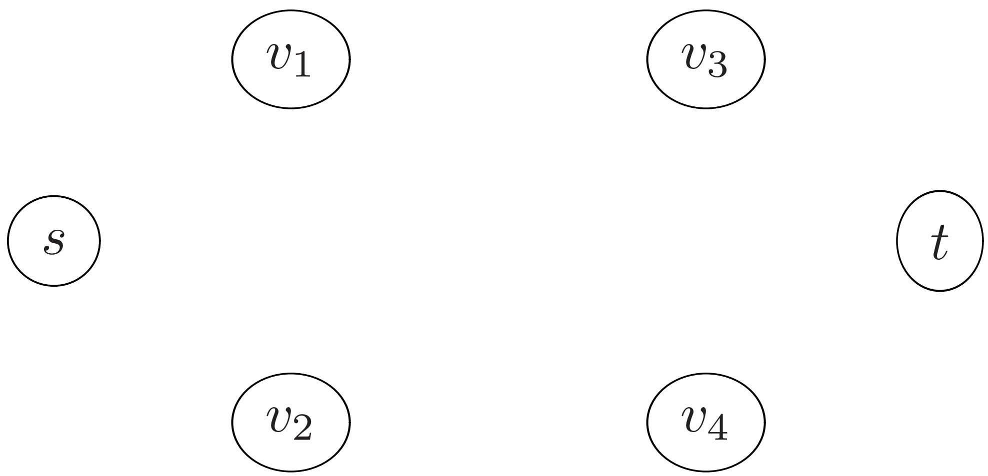
Assume that $G.E$ never contains both (u,v) and (v,u).

What is the residual network?

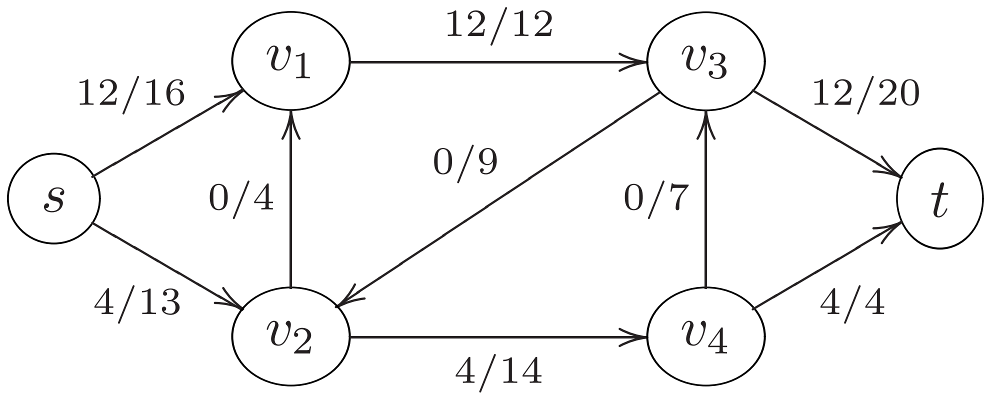


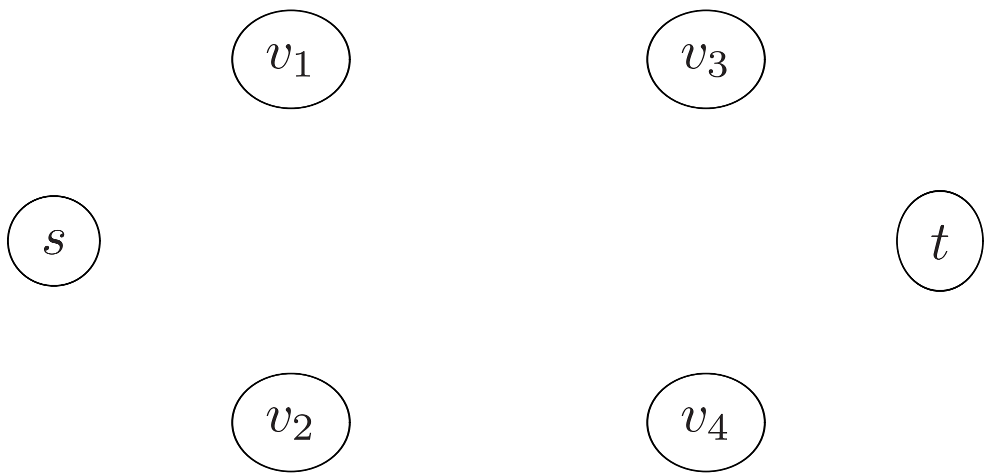
What is the residual network?



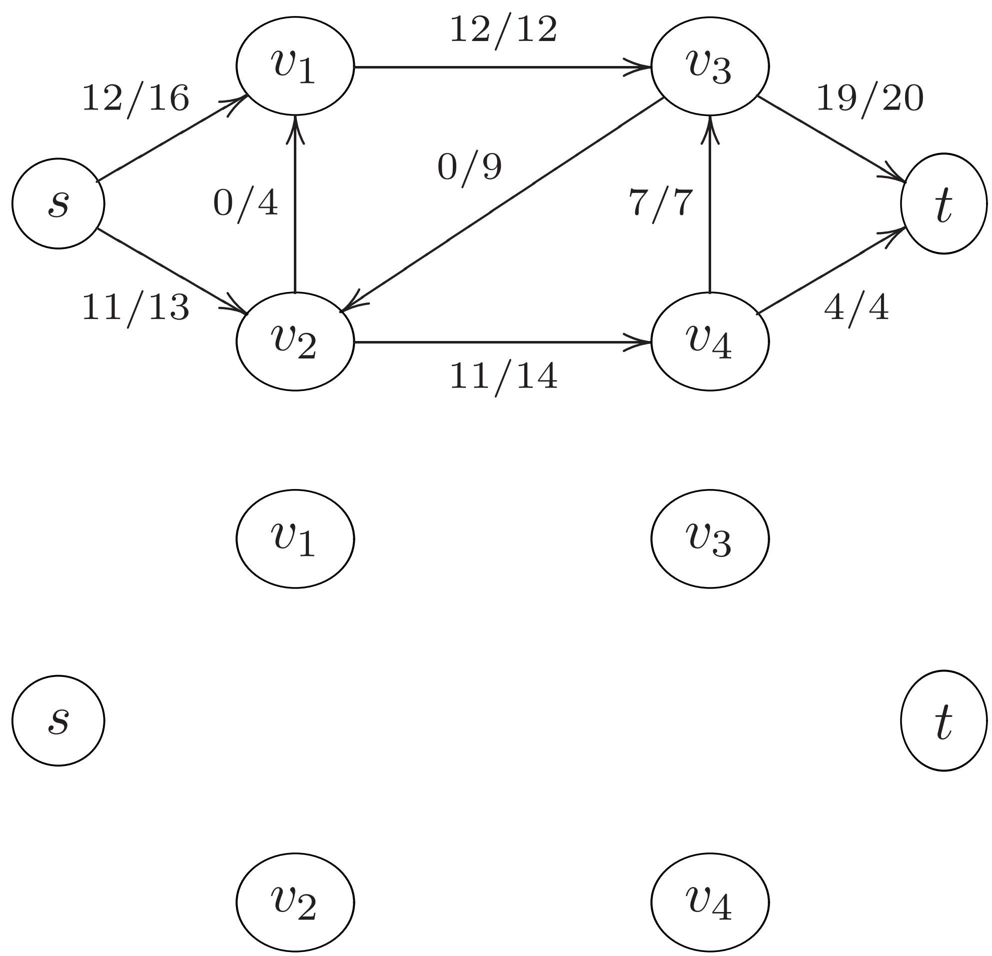


What is the residual network?





What is the residual network?



Residual Network

G is a directed graph where $G.E$ never contains both (u, v) and (v, u).

f is a flow network on G.

Residual capacity:

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

G_f is the residual network whose edges have capacities $c_f(u, v)$.

Ford-Fulkerson Max Flow Algorithm

procedure `FFMaxFlow(G)`

foreach edge $(u, v) \in E(\mathsf{G})$ **do** $f(u, v) \leftarrow 0$;

Compute residual network G_f;

Search for path P in residual network G_f;

while there exists a path P from s to t in G_f **do**

  $x \leftarrow \min\{c_f(u, v) | (u, v) \in P\}$;

  Increase flow in G by x along path P;

  Compute residual network G_f;

  Search for path P in residual network G_f;

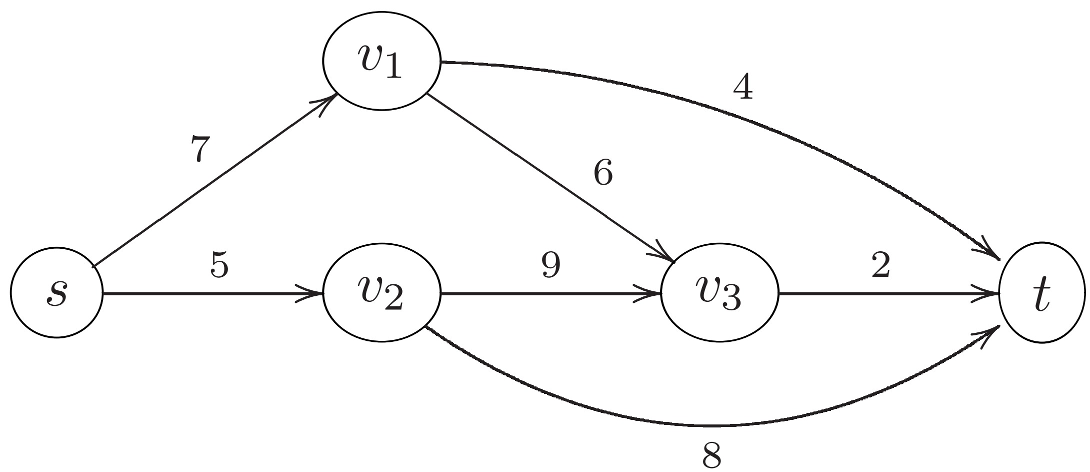
end

Ford-Fulkerson Max Flow Algorithm (Detailed)

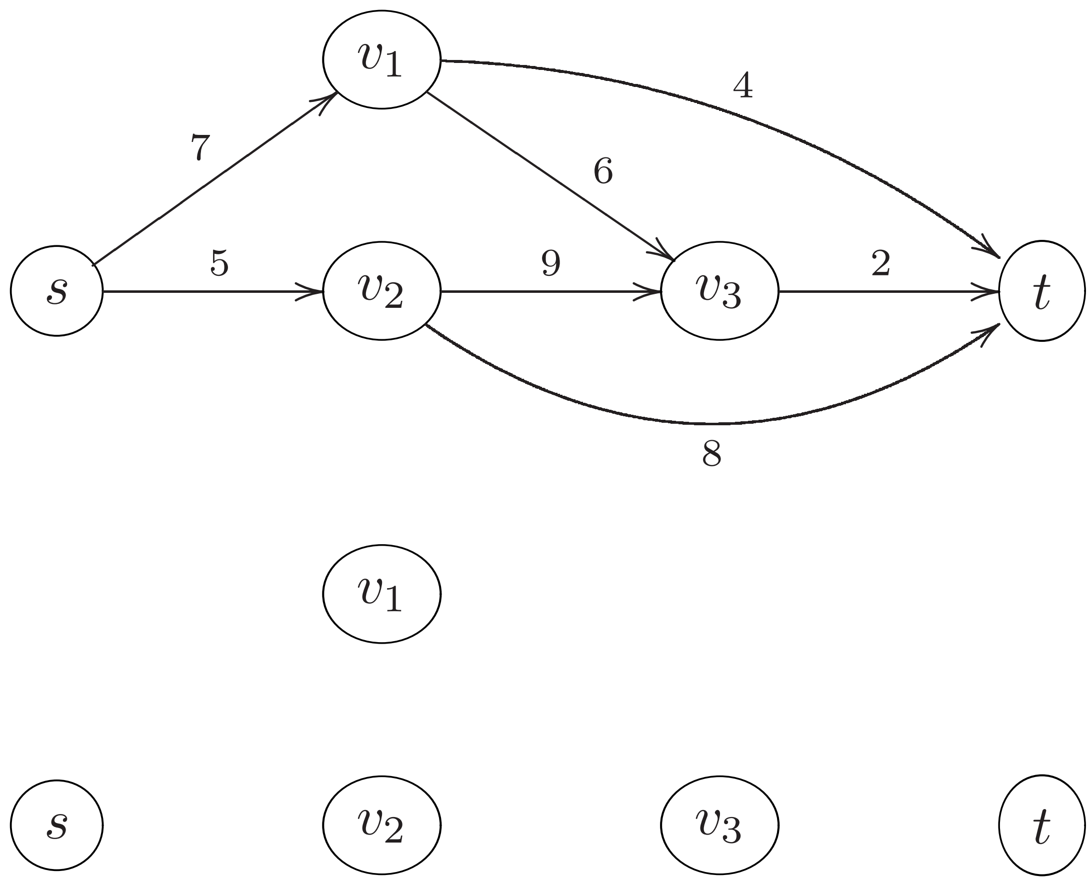
procedure `FFMaxFlow(G)`

foreach edge $(u, v) \in E(\mathsf{G})$ **do** $f(u, v) \leftarrow 0$;
Compute residual network G_f;
Search for path P in residual network G_f;
while there exists a path P from s to t in G_f **do**
    $x \leftarrow \min\{c_f(u, v) | (u, v) \in P\}$;
    /* *Increase flow in* G *by* x *along path* P */
    foreach edge $(u, v) \in P$ **do**
        if $(u, v) \in E(\mathsf{G})$ **then** $f(u, v) \leftarrow f(u, v) + x$;
        else $f(v, u) \leftarrow f(v, u) - x$;   /* $(v, u) \in E(\mathsf{G})$ */
    end
    Compute residual network G_f;
    Search for path P in residual network G_f;
end

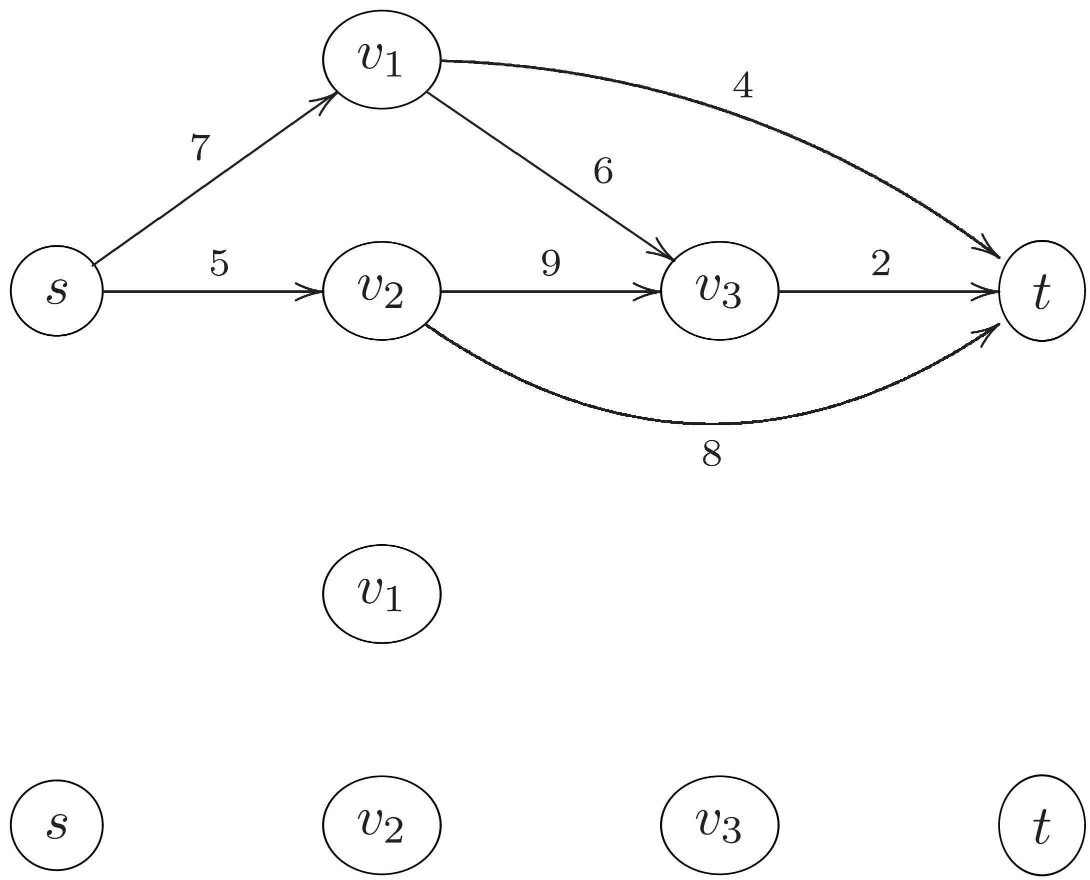
Max Flow Example



Max Flow Example

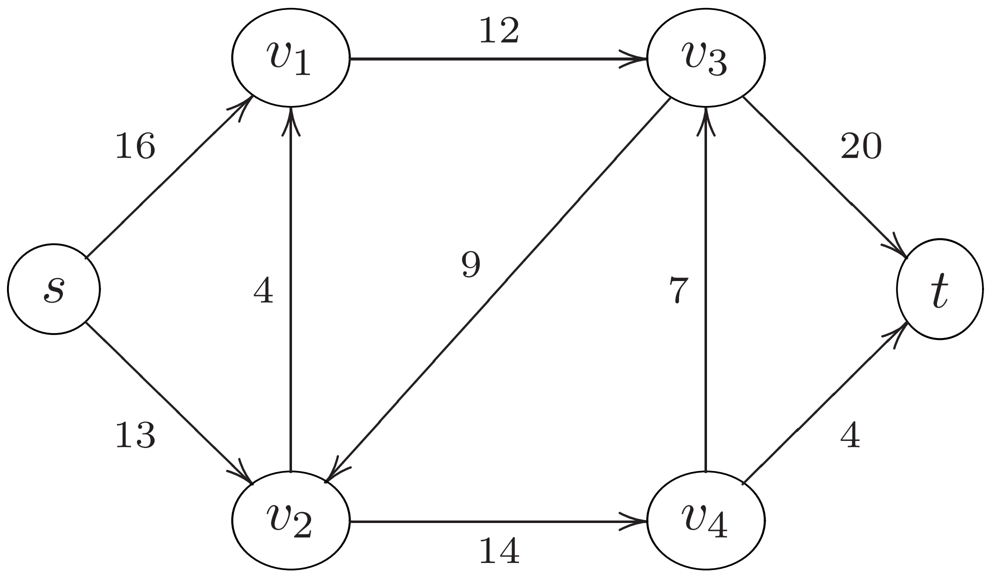


Max Flow Example



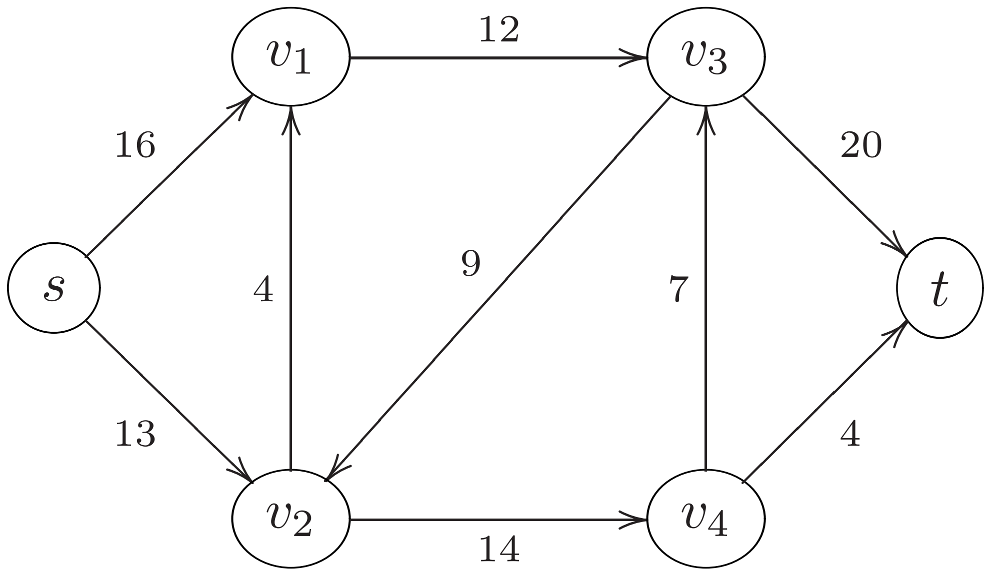
Minimum Cut

Cut



A cut (S, T) of a flow network G is a partition of $G.V$ into S and $T = G.V - S$ such that $s \in S$ and $t \in T$.

Cut

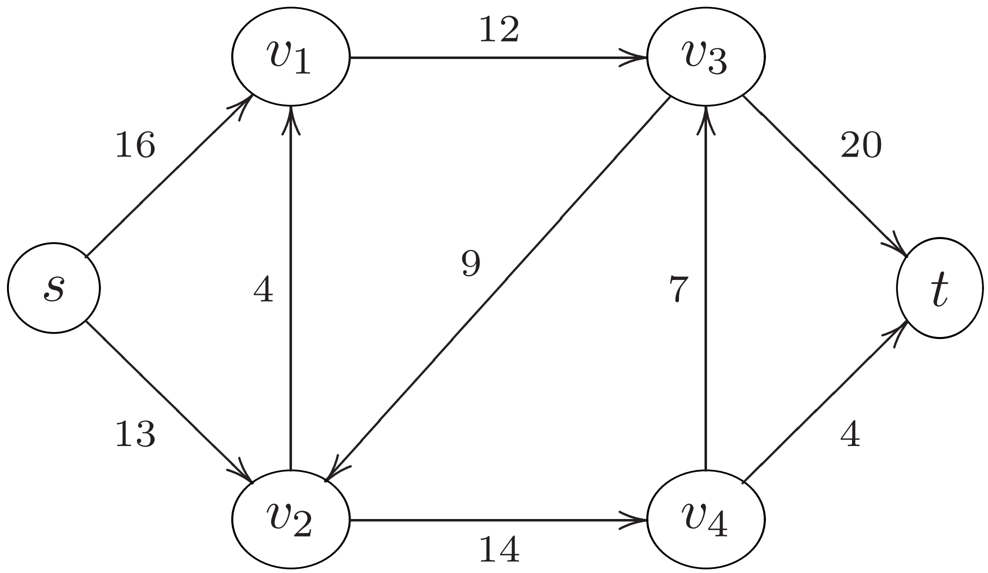


A cut (S, T) of a flow network G is a partition of $G.V$ into S and $T = G.V - S$ such that $s \in S$ and $t \in T$.

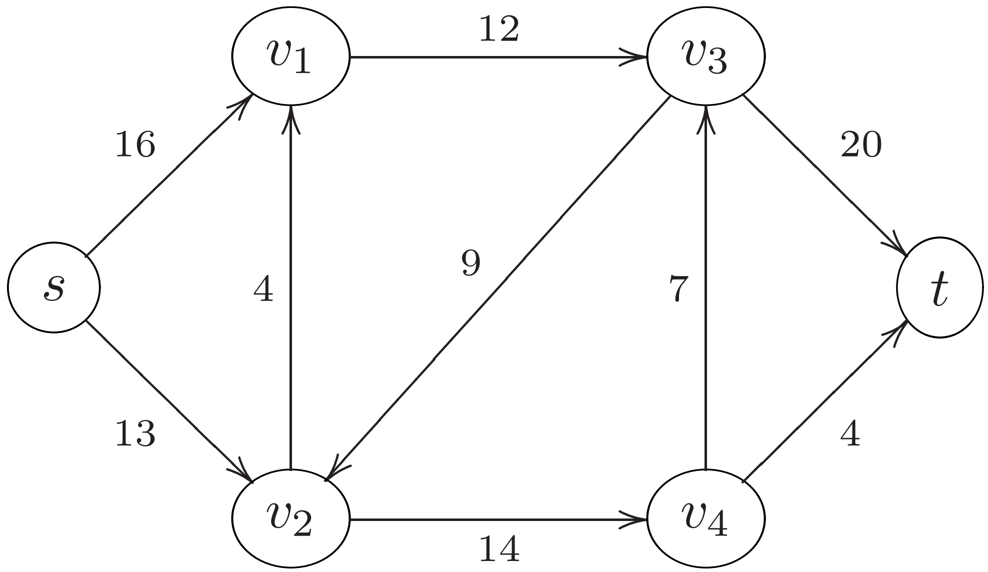
The capacity of the cut (S, T) is:

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

Cut



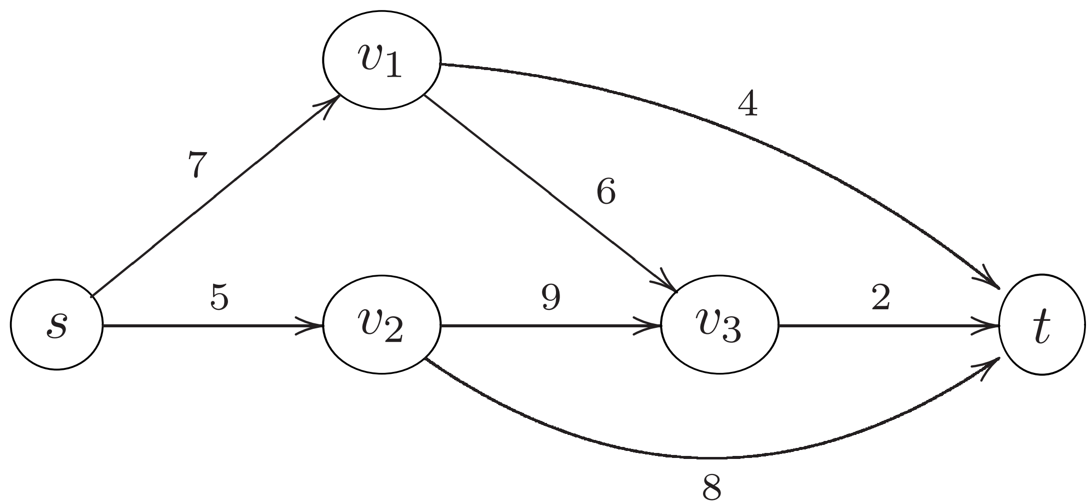
Minimum Cut



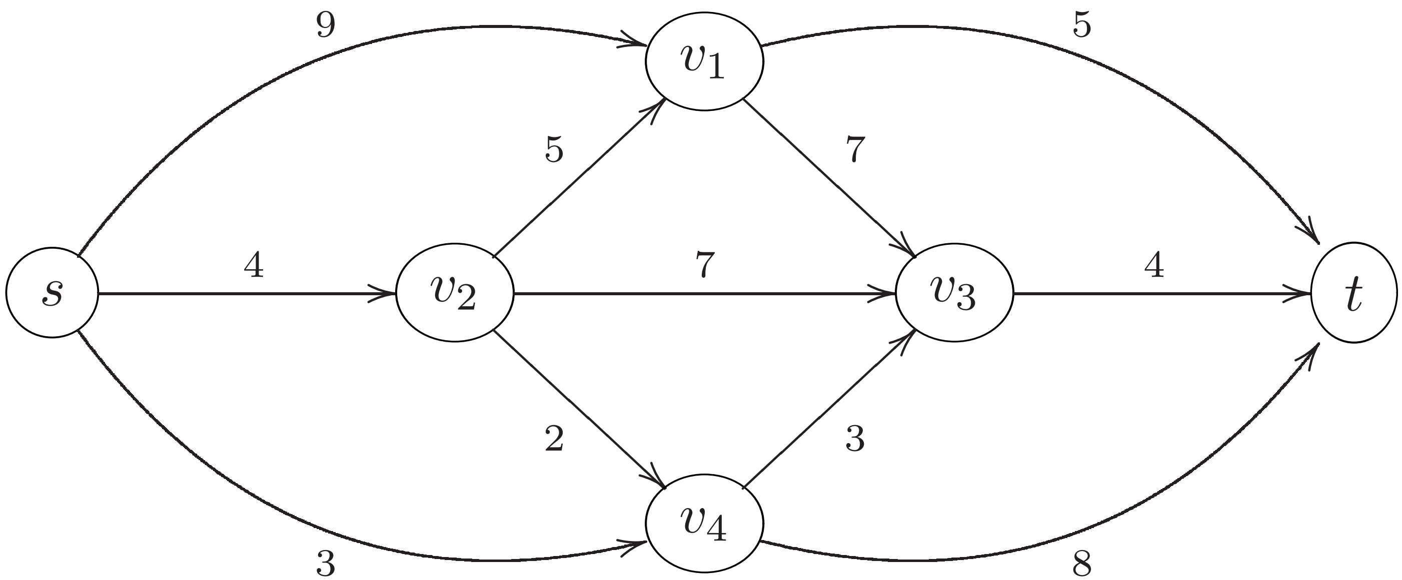
The capacity of the cut (S, T) is $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$.

A mininum cut of G is a cut whose capacity is minimum over all cuts of G.

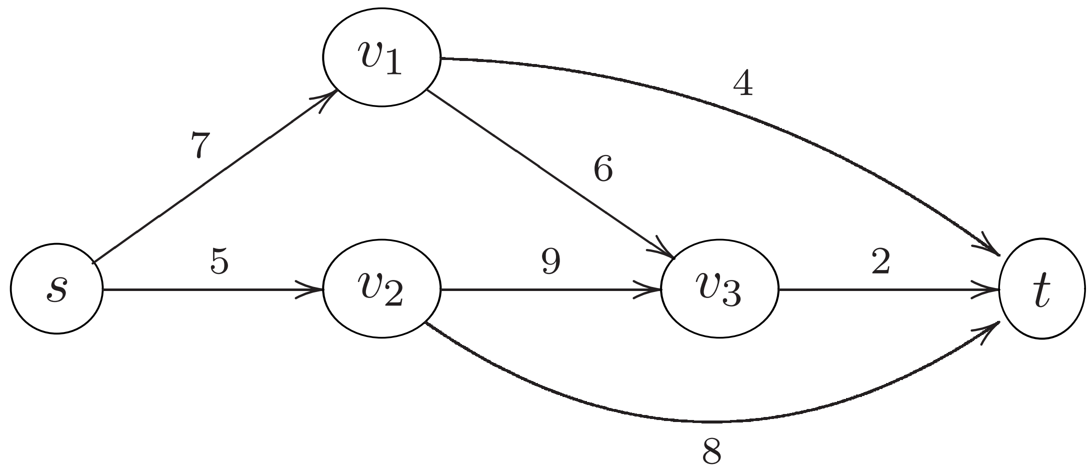
Minimum Cut



Minimum Cut



Flows and Cuts



$|f| = \sum_{v \in G.V} f(s,v)$ is the value of flow f.

$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$ is the capacity of cut (S,T).

Lemma (Cut Lemma)**.** For any flow f and any cut (S,T),

$$|f| \leq c(S,T).$$

max-flow min-cut theorem

Theorem. For any flow network G,

$$\text{max flow of } G = \text{min cut of } G!$$

Max-flow min-cut theorem

Proof: The following three conditions are equivalent:

1. f is a maximum flow of G.
2. There are no augmenting paths in the residual network G_f.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

Max-flow min-cut theorem: (1) $\Rightarrow$ (2).

The following three conditions are equivalent:

1. f is a maximum flow of G.
2. There are no augmenting paths in the residual network G_f.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

(1) $\Rightarrow$ (2): If G_f had an augmenting path P, then we could increase $|f|$ by adding flow along P to f.

Max-flow min-cut theorem: $(2) \Rightarrow (3)$.

The following three conditions are equivalent:

1. f is a maximum flow of G.
2. There are no augmenting paths in the residual network G_f.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

$(2) \Rightarrow (3)$: Assume G_f has no augmenting path.
Let $S = \{v \in G.V : \text{there is a path from } s \text{ to } v \text{ in } G_f\}$.
Let $T = G.V - S$.

Since there is no edge in G_f from any $u \in S$ to any $v \in T$:

- the flow from S to T is $\sum_{u \in S} \sum_{v \in T} c(u, v) = c(S, T)$;
- there is no flow from T to S.

Thus $|f| = c(S, T)$.

Max-flow min-cut theorem: (3) $\Rightarrow$ (1).

The following three conditions are equivalent:

1. f is a maximum flow of G.
2. There are no augmenting paths in the residual network G_f.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

(3) $\Rightarrow$ (1): Assume $|f| = c(S, T)$.
By the cut lemma (slide 10.34), $|f'| \leq c(S, T)$ for any flow f' in G.
Thus, $|f'| \leq c(S, T) = |f|$ so $|f|$ is a maximum flow.

Ford-Fulkerson Max Flow Algorithm
Running Time Analysis

Ford-Fulkerson Max Flow Algorithm: Time

procedure `FFMaxFlow(G)`

foreach edge $(u, v) \in E(\mathsf{G})$ **do** $f(u, v) \leftarrow 0$;

Compute residual network G_f;

Search for path P in residual network G_f;

while there exists a path P from s to t in G_f **do**

  $x \leftarrow \min\{c_f(u, v) | (u, v) \in P\}$;

  Increase flow in G by x along path P;

  Compute residual network G_f;

  Search for path P in residual network G_f;

end

Ford-Fulkerson Max Flow Algorithm: Time

Lemma. If all capacities are integers, then `FFMaxFlow` increases the flow value by a positive integer at each iteration.

Ford-Fulkerson Max Flow Algorithm: Time

Lemma. If all capacities are integers, then `FFMaxFlow` increases the flow value by a positive integer at each iteration.

Idea of proof. Initially, flow is zero so all flows are integers.

If all capacities and flows are integers:

$\Downarrow$

Capacities in the residual network G_f are integers.

$\Downarrow$

$x = \min\{c_f(u, v) | (u, v) \in P\}$ is an integer.

$\Downarrow$

New flow in G is an integer.

$\Downarrow$

All capacities and flows are integers.

$\square$

Apply induction for formal proof.

FF Max Flow Algorithm: Time Analysis

procedure `FFMaxFlow(G)`

foreach edge $(u, v) \in E(\mathsf{G})$ **do** $f(u, v) \leftarrow 0$;

Compute residual network G_f;

Search for path P in residual network G_f;

while there exists a path P from s to t in G_f **do**

  $x \leftarrow \min\{c_f(u, v) | (u, v) \in P\}$;

  Increase flow in G by x along path P;

  Compute residual network G_f;

  Search for path P in residual network G_f;

end

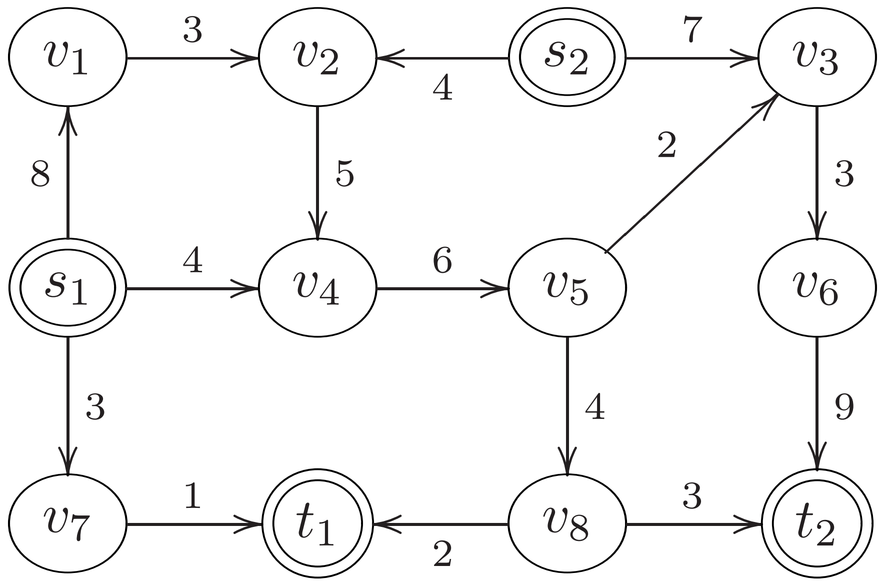
Ford-Fulkerson Max Flow Algorithm: Time

$m = \#$ graph edges.

Proposition. If all capacities are integers, then the Ford-Fulkerson Algorithm runs in $O(m|f^*|)$ time where f^* is the max flow.

Multi-Source/Sink Max-Flow

Multiple Sources and Sinks



Sources: s_1 and s_2.

Sinks: t_1 and t_2.

Flow value $|f| = \sum_{s_i} \sum_{v_j} f(s_i, v_j)$.

Reduction

Multi-Source/Sink Max-Flow Problem: Given a flow network G with multiple sources and sinks, find $\max|f|$ over all flows f in G.

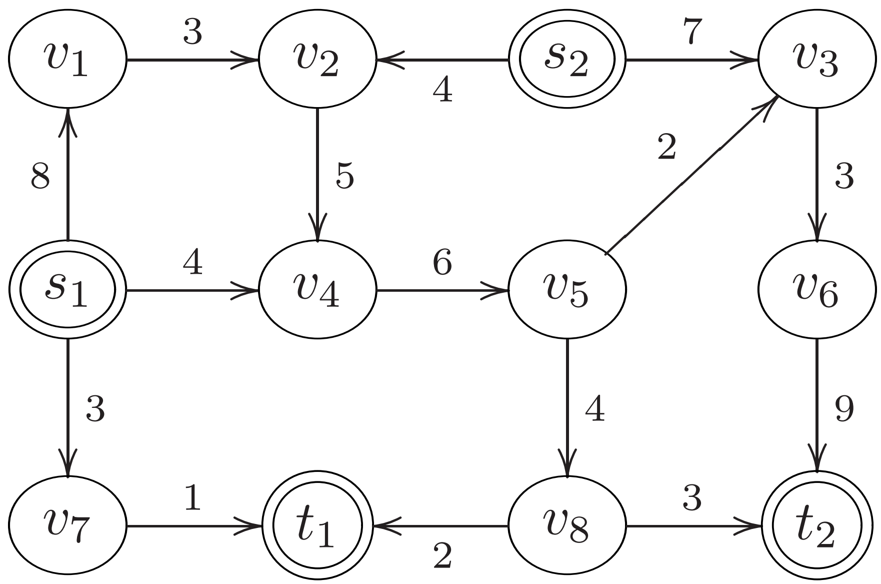
Single Source/Sink Max-Flow Problem: Given a flow network G with one source and sink, find $\max|f|$ over all flows f in G.

Reduce the Multi-Source/Sink Max-Flow Problem to the Single Source Max Flow Problem.

Reduce P to Q: Turn problem P into Q such that the solution to Q gives the solution to P.

Multi-Source/Sink Max-Flow Problem

Reduce Multi-Source/Sink Max-Flow Problem to Single Source/Sink Max-Flow Problem:



Multi-Source/Sink Max-Flow Problem

Reduce the Multi-Source/Sink Max-Flow Problem to the Single Source Max Flow Problem:

Let G be a flow network with multiple sources s_i and sinks t_i.

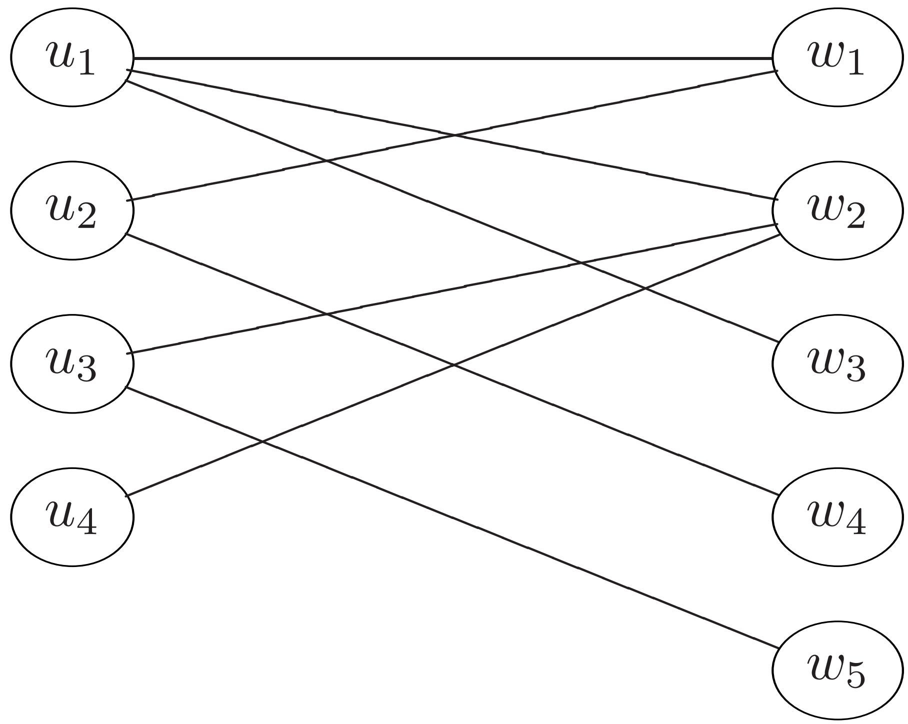
Create flow network G' from G with a single source and sink as follows:

- Add new source s^* and new sink t^*;
- Add directed edges from s^* to each s_i.
  Set capacity of each new edge to ∞.
- Add directed edges from each t_i to t^*.
  Set capacity of each new edge to ∞.

G' has flow with value F from s^* to t^* if and only if G has flow with value F from the s_i to the t_i.

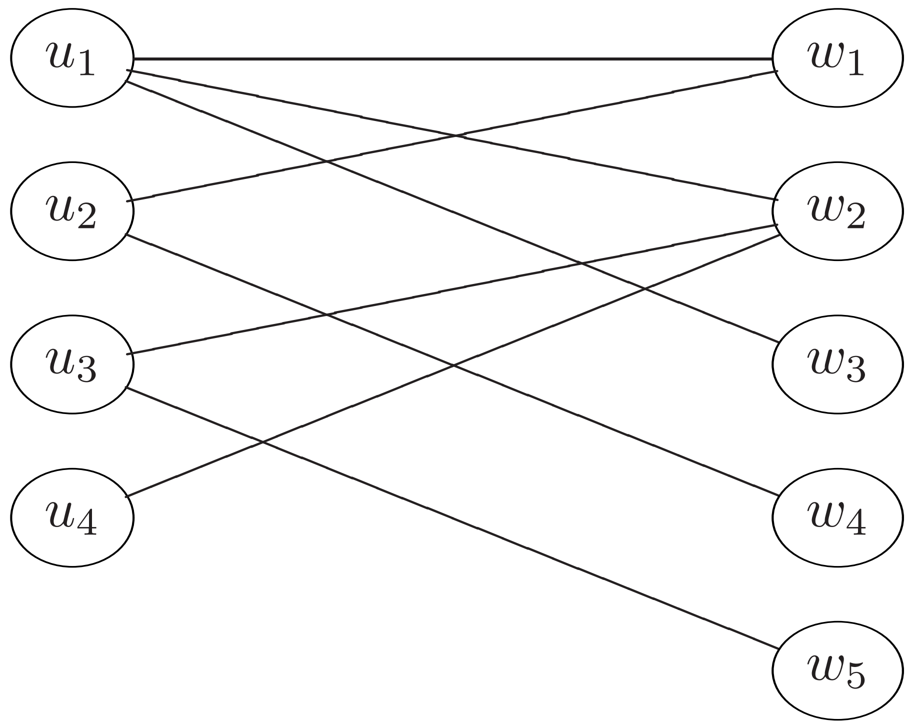
Bipartite Matching

Bipartite Graph



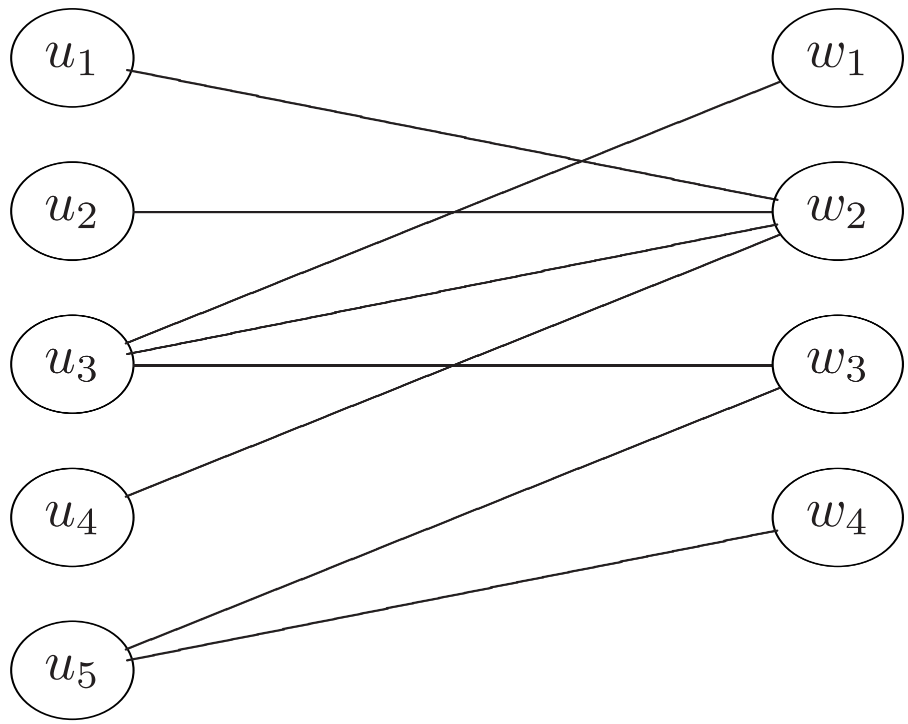
Definition. An undirected graph G is **bipartite** if its vertices can be partitioned into two sets U and W such that every graph edge $e \in G.E$ has one endpoint u_i in U and one endpoint w_j in W.

Bipartite Matching



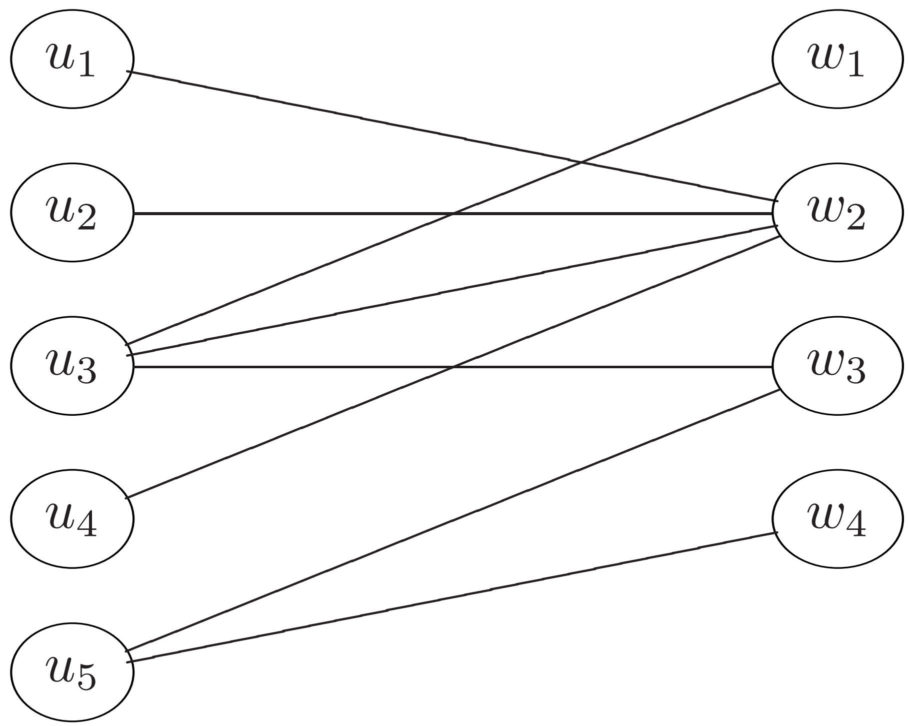
Definition. A **matching** of a bipartite graph G is a subset M of the edges $G.E$ of G such that no two edges share an endpoint.

Bipartite Matching



Definition. A **matching** of a bipartite graph G is a subset M of the edges $G.E$ of G such that no two edges share an endpoint.

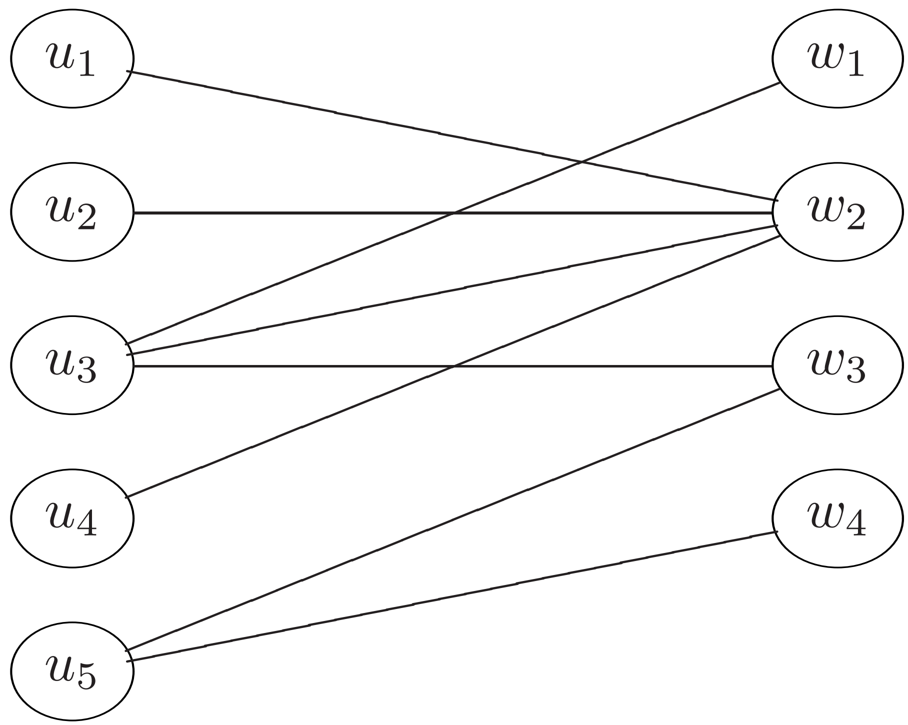
Maximum Bipartite Matching



Definition. A **maximum matching** of a bipartite graph G is a matching with greatest number of edges.

Note: A maximum matching has at most $\min(|U|, |W|)$ edges. (Why?)

Maximum Bipartite Matching



Bipartite Matching Problem: Given a bipartite graph G, find a maximum matching of G.

Reduction

Bipartite Matching Problem: Given a bipartite graph G, find a maximum matching of G.

(Single Source/Sink) Max-Flow Problem: Given a flow network G with one source and sink, find $\max|f|$ over all flows f in G.

Reduce the Bipartite Matching Problem to the (Single Source) Max Flow Problem.

Reduce P to Q: Turn problem P into Q such that the solution to Q gives the solution to P.

Bipartite Matching Problem

Reduce the Bipartite Matching Problem to the (Single Source) Max Flow Problem.

Let G be the bipartite graph whose edges connect $U \subset G.V$ to $W \subset G.V$.

Create a flow network G' from G as follows:

- Add source node s and sink node t;
- Replace each undirected edge (u_i, w_i) of $G.E$ with a directed edge (u_i, w_i);
- Add directed edges from s to each $u_i \in U$;
- Added directed edges from each $w_i \in W$ to t;
- Set the capacity of every edge to 1.

G' has flow with value F from s to t if and only if G has a matching of size F.

Ford-Fulkerson Max Flow Algorithm: Time

$m = \#$ graph edges.

Proposition. If all capacities are integers, then the Ford-Fulkerson Algorithm runs in $O(m|f^*|)$ time where f^* is the max flow.

Bipartite Matching: Time

$m = \#$ graph edges.

Proposition. If all capacities are integers, then the Ford-Fulkerson Algorithm runs in $O(m|f^*|)$ time where f^* is the max flow.

In the reduction of bipartite matching to max flow:

- All edge capacities are 1;
- The max flow $|f^*| \leq n$.

Proposition. Reducing bipartite matching to max flow and applying the Ford-Fulkerson Algorithm takes $O(nm)$ time.