NDMI011: Combinatorics and Graph Theory 1

Lecture #6

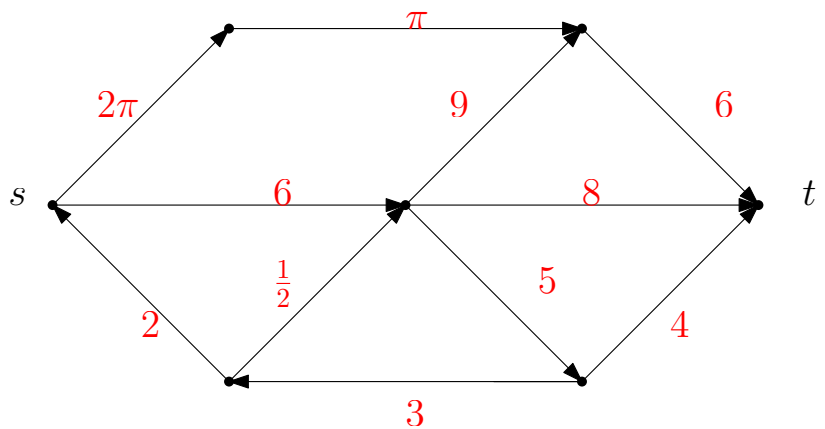
Flows and cuts in networks

Irena Penev

November 3, 2021

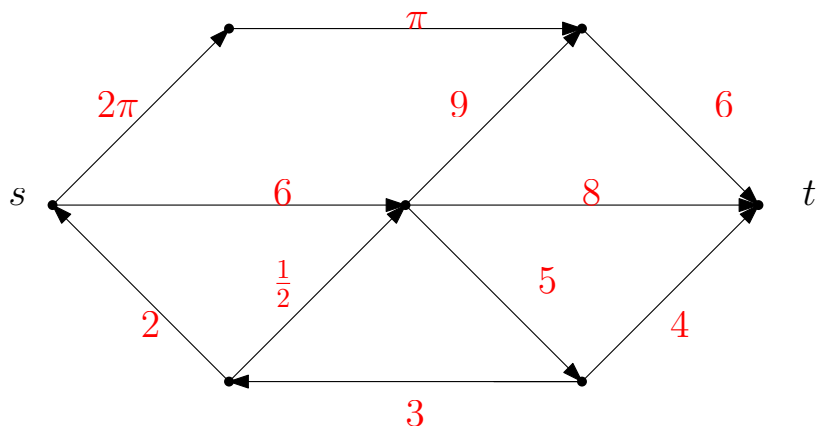
Definition

A *network* is an ordered four-tuple (G, s, t, c), where G is an oriented graph, s and t are two distinct vertices of this graph (called the *source* and *sink*, respectively), and $c : E(G) \to [0, +\infty)$ is a function, called the *capacity function*. The *capacity* of an edge $e \in E(G)$ is the number $c(e)$.



Definition

A *network* is an ordered four-tuple (G, s, t, c), where G is an oriented graph, s and t are two distinct vertices of this graph (called the *source* and *sink*, respectively), and $c : E(G) \to [0, +\infty)$ is a function, called the *capacity function*. The *capacity* of an edge $e \in E(G)$ is the number $c(e)$.

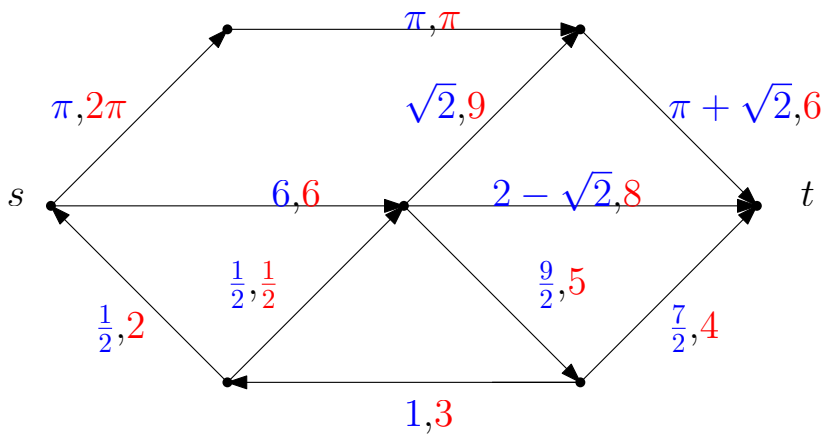


- Networks can be used to model, for example, a system of pipes used to transport some resource, such as water or oil; capacities would be the number of units of volume that a given pipe can transport per unit time.

Definition

A *feasible flow* (or simply *flow*) in a network (G, s, t, c) is a function $f : E(G) \to [0, +\infty)$ s.t.:

- $f(e) \leq c(e)$ for all $e \in E(G)$;
- for all $v \in V(G) \setminus \{s, t\}$, we have
$$\sum_{(x,v) \in E(G)} f(x, v) = \sum_{(v,y) \in E(G)} f(v, y).$$

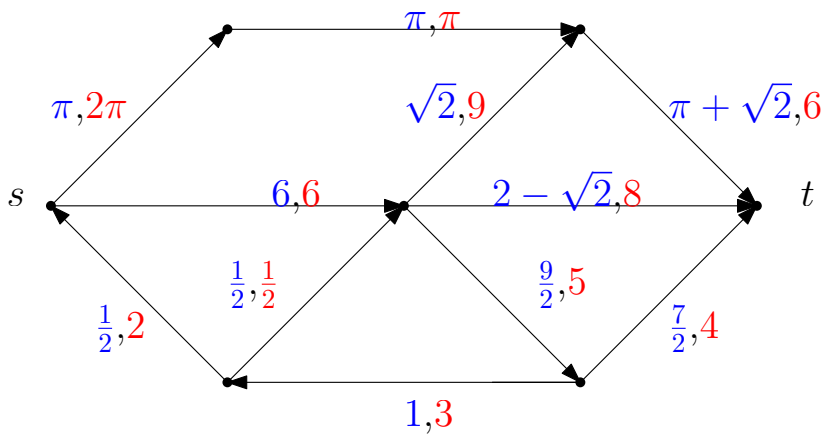


Definition

The *value* of a flow f in a network (G, s, t, c) is

$$val(f) = \Big(\sum_{(s,x)\in E(G)} f(s,x) \Big) - \Big(\sum_{(x,s)\in E(G)} f(x,s) \Big).$$

A *maximum flow* in (G, s, t, c) is a flow f^* that has maximum value, i.e. one that satisfies $val(f) \leq val(f^*)$ for all flows f.



- The value of the flow above is $\pi + 6 - \frac{1}{2} = \frac{11}{2} + \pi$.

Theorem 1.1

Every network (G, s, t, c) has a maximum flow.

Proof. Omitted.

Theorem 1.1

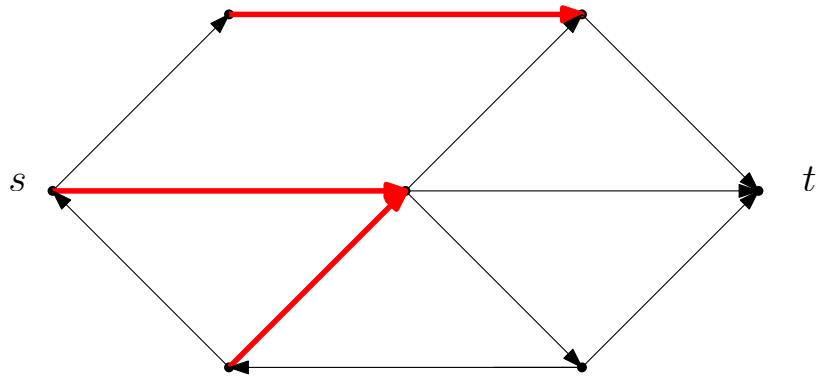
Every network (G, s, t, c) has a maximum flow.

Proof. Omitted.

- Theorem 1.1 should seem plausible, but the proof is not obvious (since the number of flows is, typically, infinite).
- The proof relies on certain results from analysis, which we omit.

Definition

An s,t-*cut*, or simply *cut*, in a network (G, s, t, c) is a set $R \subseteq E(G)$ such that $G \setminus R$ contains no directed path from s to t. The *capacity* of the cut R is $c(R) = \sum_{e \in R} c(e)$.



Max-flow min-cut theorem

The maximum value of a flow in a network is equal to the minimum capacity of a cut in that network.

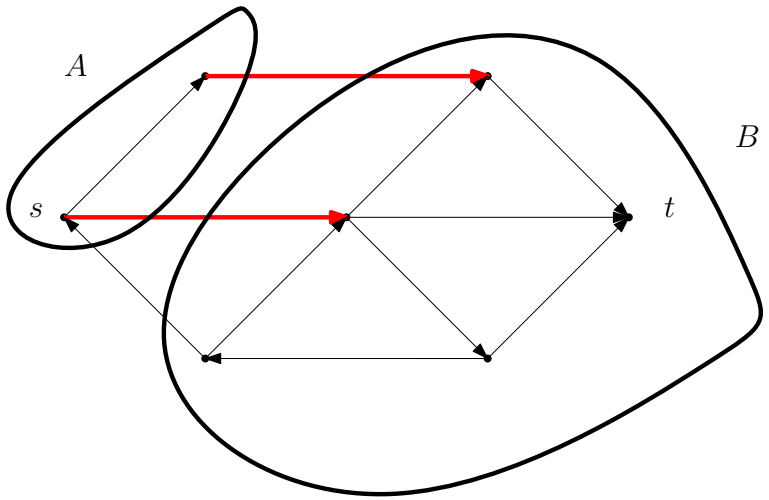
- For a network (G, s, t, c), a flow f in that network, and a set of edges $R \subseteq E(G)$, we write

$$c(R) = \sum_{e \in R} c(e) \qquad \text{and} \qquad f(R) = \sum_{e \in R} f(e).$$

- For a network (G, s, t, c), a flow f in that network, and a set of edges $R \subseteq E(G)$, we write

$$c(R) = \sum_{e \in R} c(e) \qquad \text{and} \qquad f(R) = \sum_{e \in R} f(e).$$

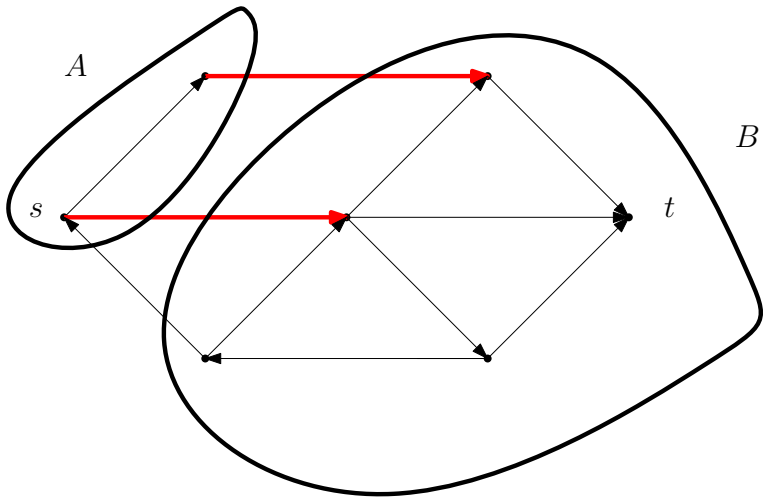
- For a directed graph G and disjoint sets $A, B \subseteq V(G)$, we set $S(A, B) = \{(a, b) \in E(G) \mid a \in A, b \in B\}$.



- For a network (G, s, t, c), a flow f in that network, and a set of edges $R \subseteq E(G)$, we write

$$c(R) = \sum_{e \in R} c(e) \qquad \text{and} \qquad f(R) = \sum_{e \in R} f(e).$$

- For a directed graph G and disjoint sets $A, B \subseteq V(G)$, we set $S(A, B) = \{(a, b) \in E(G) \mid a \in A, b \in B\}$.



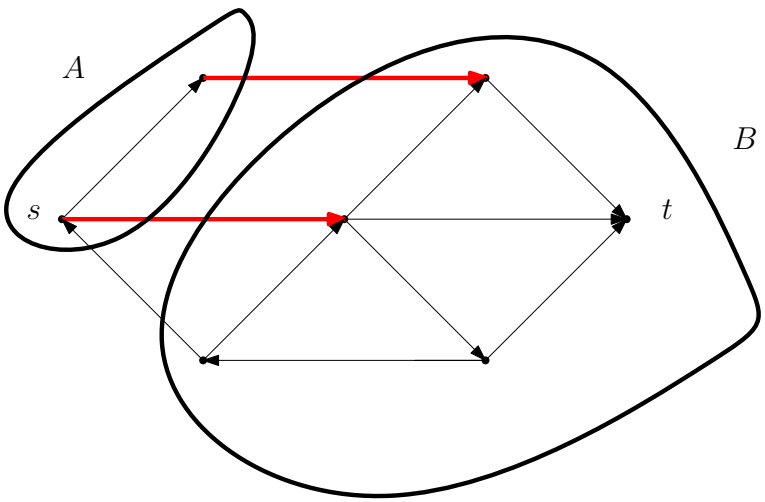
- For a network (G, s, t, c), disjoint sets $A, B \subseteq V(G)$, and a flow f, we write

$$c(A, B) = c(S(A, B)) \qquad \text{and} \qquad f(A, B) = f(S(A, B)).$$

Proposition 2.1

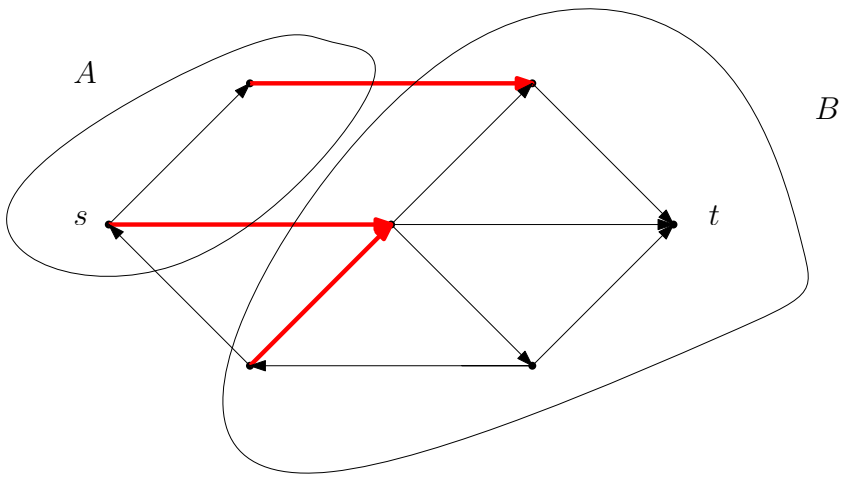
Let (G, s, t, c) be a network, and let (A, B) be a partition of $V(G)$ such that $s \in A$ and $t \in B$. Then $S(A, B)$ is a cut in (G, s, t, c).

Proof. Lecture Notes.



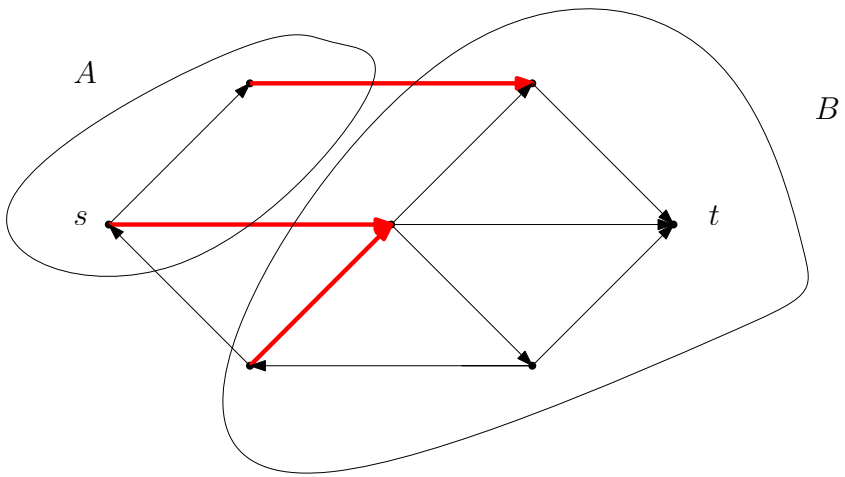
Proposition 2.2

Let (G, s, t, c) be a network, and let R be a cut in this network. Then there exists a partition (A, B) of $V(G)$ such that $s \in A$, $t \in B$, and $S(A, B) \subseteq R$.



Proposition 2.2

Let (G, s, t, c) be a network, and let R be a cut in this network. Then there exists a partition (A, B) of $V(G)$ such that $s \in A$, $t \in B$, and $S(A, B) \subseteq R$.



Proof (outline). Let A be the set of all vertices $v \in V(G)$ such that $G \setminus R$ contains a directed path from s to v, and set $B = V(G) \setminus A$.

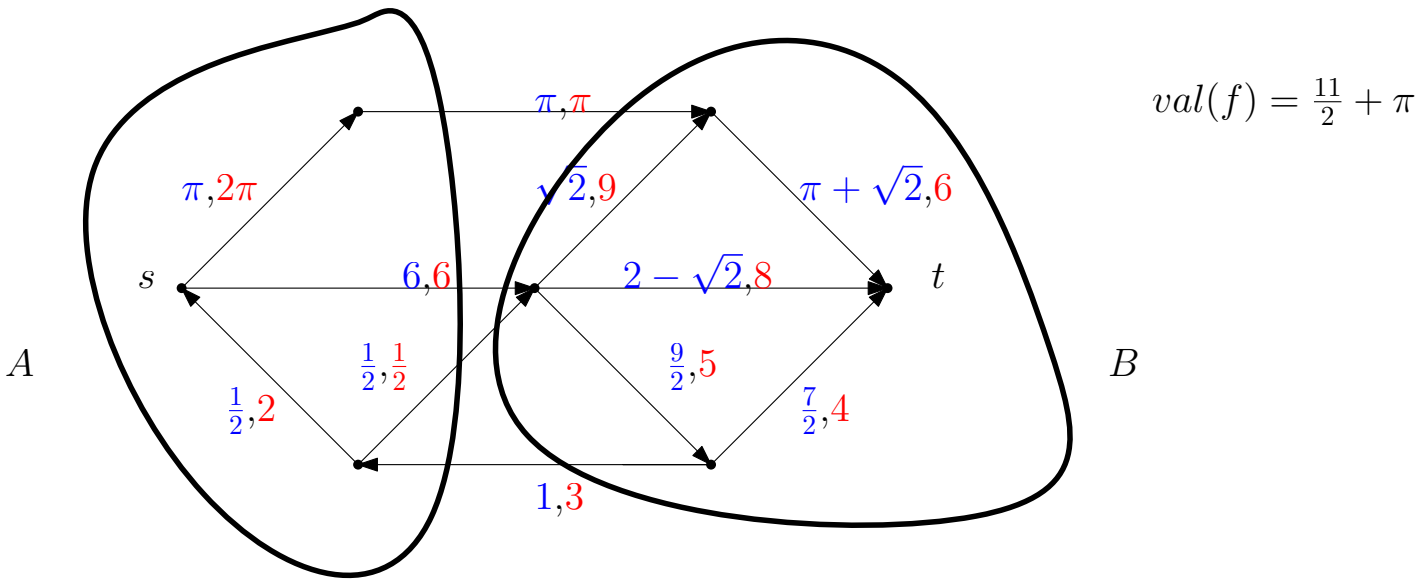
Lemma 2.3

Let f be a flow in a network (G, s, t, c), and let (A, B) be a partition of $V(G)$ such that $s \in A$ and $t \in B$. Then $val(f) = f(A, B) - f(B, A)$. In particular,[a] we have that

$$val(f) = \Big(\sum_{(x,t) \in E(G)} f(x, t) \Big) - \Big(\sum_{(t,x) \in E(G)} f(t, x) \Big).$$

[a] This happens if we take $A = V(G) \setminus \{t\}$ and $B = \{t\}$.

Proof. Lecture Notes.



Corollary 2.4

Let f be a flow in a network (G, s, t, c), and let R be a cut. Then $val(f) \leq c(R)$.

Proof.

Corollary 2.4

Let f be a flow in a network (G, s, t, c), and let R be a cut. Then $val(f) \leq c(R)$.

Proof. By Proposition 2.2, there exists a partition (A, B) of $V(G)$ such that $s \in A$, $t \in B$, and $S(A, B) \subseteq R$.

Corollary 2.4

Let f be a flow in a network (G, s, t, c), and let R be a cut. Then $val(f) \leq c(R)$.

Proof. By Proposition 2.2, there exists a partition (A, B) of $V(G)$ such that $s \in A$, $t \in B$, and $S(A, B) \subseteq R$.

Corollary 2.4

Let f be a flow in a network (G, s, t, c), and let R be a cut. Then $val(f) \leq c(R)$.

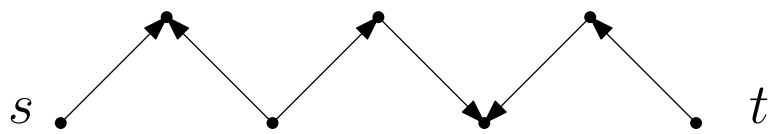
Proof. By Proposition 2.2, there exists a partition (A, B) of $V(G)$ such that $s \in A$, $t \in B$, and $S(A, B) \subseteq R$. Then

$$
\begin{aligned}
val(f) &= f(A, B) - f(B, A) && \text{by Lemma 2.3} \\
&\leq f(A, B) && \text{because } f(e) \geq 0 \ \forall e \in E(G) \\
&\leq c(A, B) && \text{because } f(e) \leq c(e) \ \forall e \in E(G) \\
&\leq c(R) && \text{because } S(A, B) \subseteq R \text{ and} \\
& && \text{and } c(e) \geq 0 \ \forall e \in E(G)
\end{aligned}
$$

which is what we needed to show.

Definition

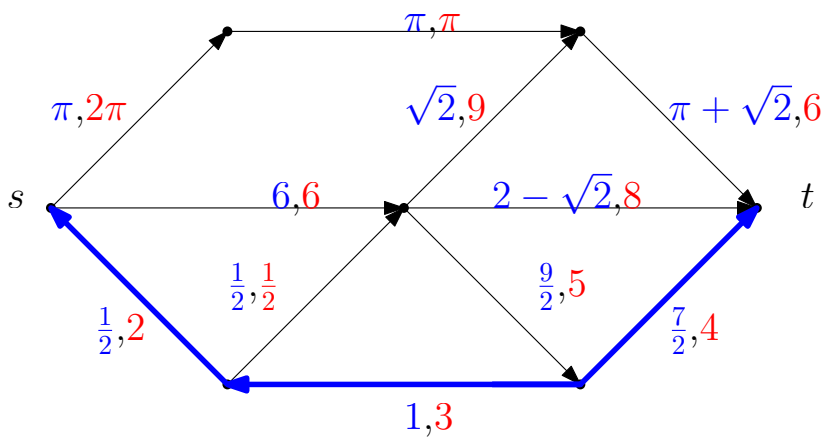
An (s, t)-*path* in a network (G, s, t, c) is a sequence $v_0, v_1, \ldots, v_\ell$ of pairwise distinct vertices of G such that $v_0 = s$, $v_\ell = t$, and for all $i \in \{0, \ldots, \ell - 1\}$, we have that one of (v_i, v_{i+1}) and (v_{i+1}, v_i) belongs to $E(G)$.



Definition

Given a flow f in the network (G, s, t, c), an (s, t)-path $v_0, v_1, \ldots, v_\ell$ in (G, s, t, c) is said to be an *f-augmenting path* if the following two conditions are satisfied:

- for all $i \in \{1, \ldots, \ell - 1\}$ such that $(v_i, v_{i+1}) \in E(G)$, we have that $f(v_i, v_{i+1}) < c(v_i, v_{i+1})$;
- for all $i \in \{1, \ldots, \ell - 1\}$ such that $(v_{i+1}, v_i) \in E(G)$, we have that $f(v_{i+1}, v_i) > 0$.



Lemma 2.5

Let f be a flow in a network (G, s, t, c). Then f is a maximum flow if and only if there does not exist an f-augmenting path in (G, s, t, c). Furthermore, if f is a maximum flow, then there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof.

Lemma 2.5

Let f be a flow in a network (G, s, t, c). Then f is a maximum flow if and only if there does not exist an f-augmenting path in (G, s, t, c). Furthermore, if f is a maximum flow, then there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof. It suffices the prove the following two statements:

(a) If there exists an f-augmenting path in (G, s, t, c), then f is not a maximum flow in (G, s, t, c).

(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

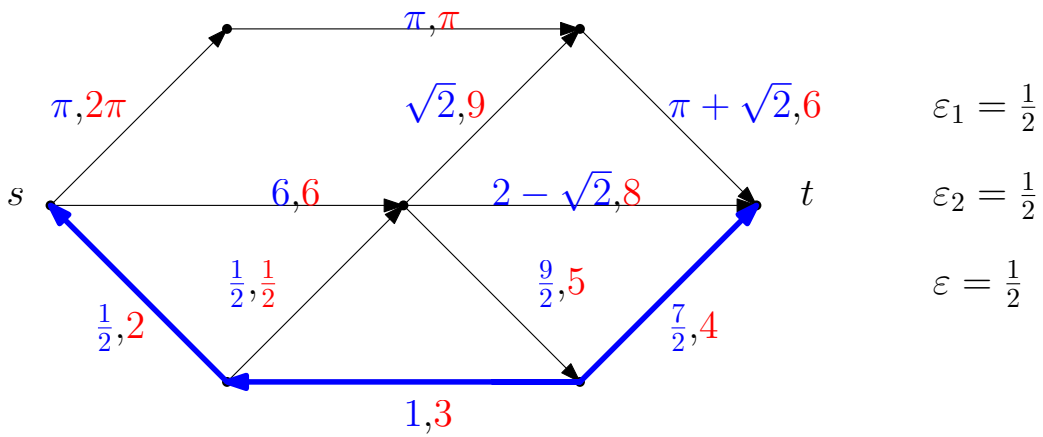
(a) If there exists an f-augmenting path in (G, s, t, c), then f is not a maximum flow in (G, s, t, c).

Proof of (a). Suppose that $v_0, \ldots, v_\ell$ (with $v_0 = s$ and $v_\ell = t$) is an f-augmenting path in (G, s, t, c).

(a) If there exists an f-augmenting path in (G, s, t, c), then f is not a maximum flow in (G, s, t, c).

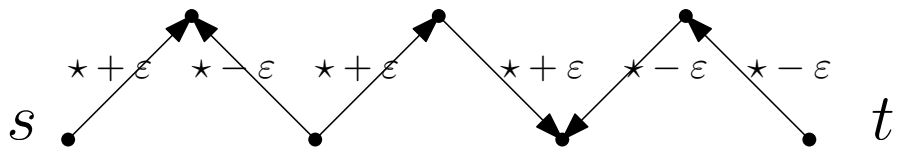
Proof of (a). Suppose that $v_0, \ldots, v_\ell$ (with $v_0 = s$ and $v_\ell = t$) is an f-augmenting path in (G, s, t, c). Now, set

- $\varepsilon_1 = \min\Big(\{c(v_i, v_{i+1}) - f(v_i, v_{i+1}) \mid 0 \leq i \leq \ell - 1, (v_i, v_{i+1}) \in E(G)\} \cup \{\infty\}\Big)$;
- $\varepsilon_2 = \min\Big(\{f(v_{i+1}, v_i) \mid 0 \leq i \leq \ell - 1, (v_{i+1}, v_i) \in E(G)\} \cup \{\infty\}\Big)$;
- $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\}$.



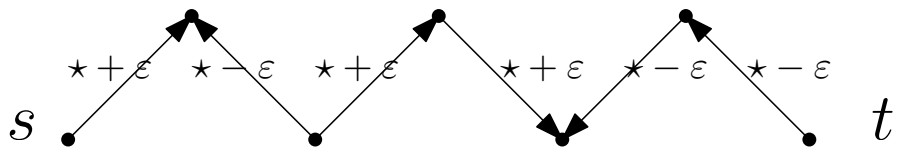
We now define a new flow f' as follows:

- $f'(v_i, v_{i+1}) = f(v_i, v_{i+1}) + \varepsilon$ for all $i \in \{0, \ldots, \ell - 1\}$ such that $(v_i, v_{i+1}) \in E(G)$;
- $f'(v_{i+1}, v_i) = f(v_{i+1}, v_i) - \varepsilon$ for all $i \in \{0, \ldots, \ell - 1\}$ such that $(v_{i+1}, v_i) \in E(G)$;
- $f'(e) = f(e)$ for all other edges e.



We now define a new flow f' as follows:

- $f'(v_i, v_{i+1}) = f(v_i, v_{i+1}) + \varepsilon$ for all $i \in \{0, \ldots, \ell - 1\}$ such that $(v_i, v_{i+1}) \in E(G)$;
- $f'(v_{i+1}, v_i) = f(v_{i+1}, v_i) - \varepsilon$ for all $i \in \{0, \ldots, \ell - 1\}$ such that $(v_{i+1}, v_i) \in E(G)$;
- $f'(e) = f(e)$ for all other edges e.



Then $val(f) < val(f')$, and so f is not a maximum flow. This proves (a).

(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof of (b). Suppose that (G, s, t, c) does not admit an f-augmenting path.

(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

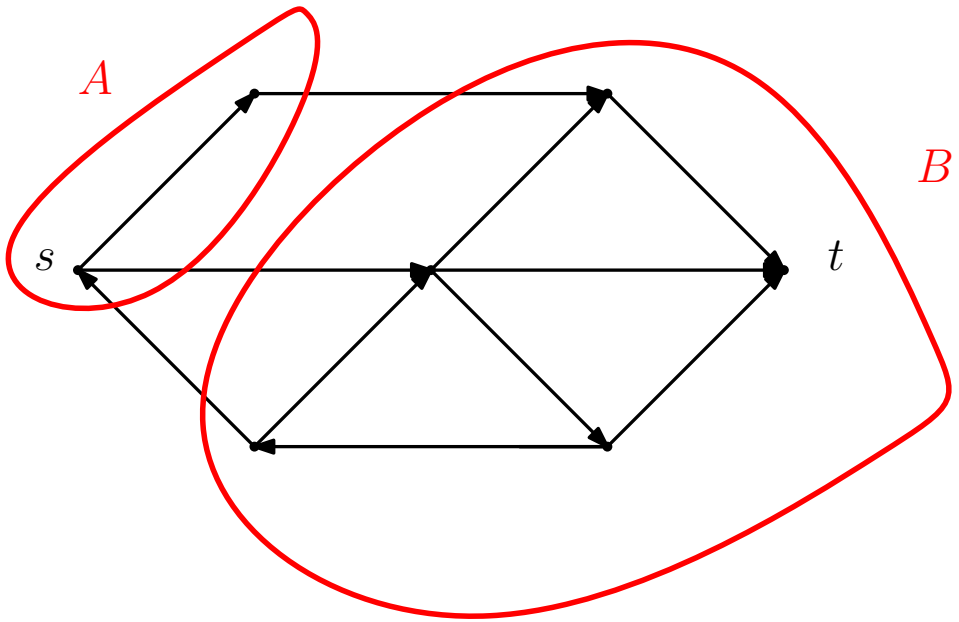
Proof of (b). Suppose that (G, s, t, c) does not admit an f-augmenting path. Let A to be the set of all vertices $v \in V(G)$ such that there exists an f-augmenting path from s to v. Let $B = V(G) \setminus A$.

(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof of (b). Suppose that (G, s, t, c) does not admit an f-augmenting path. Let A to be the set of all vertices $v \in V(G)$ such that there exists an f-augmenting path from s to v. Let $B = V(G) \setminus A$. Clearly, $s \in A$ and $t \notin A$.

(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof of (b). Suppose that (G, s, t, c) does not admit an f-augmenting path. Let A to be the set of all vertices $v \in V(G)$ such that there exists an f-augmenting path from s to v. Let $B = V(G) \setminus A$. Clearly, $s \in A$ and $t \notin A$. Then $f(A, B) = c(A, B)$ and $f(B, A) = 0$.



(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof of (b) (continued). Reminder: $f(A, B) = c(A, B)$ and $f(B, A) = 0$.

So,

$$\begin{aligned} val(f) &= f(A, B) - f(B, A) && \text{by Lemma 2.3} \\ &= c(A, B) && \text{because } f(A, B) = c(A, B) \text{ and } f(B, A) = 0 \end{aligned}$$

(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof of (b) (continued). Reminder: $f(A, B) = c(A, B)$ and $f(B, A) = 0$.

So,

$$\begin{aligned} val(f) &= f(A, B) - f(B, A) && \text{by Lemma 2.3} \\ &= c(A, B) && \text{because } f(A, B) = c(A, B) \text{ and } f(B, A) = 0 \end{aligned}$$

By Proposition 2.1, we know that $R := S(A, B)$ is a cut, and by what we just showed, $val(f) = c(A, B) = c(R)$.

(b) If there does not exist an f-augmenting path in (G, s, t, c), then f is a maximum flow in (G, s, t, c), and furthermore, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$.

Proof of (b) (continued). Reminder: $f(A, B) = c(A, B)$ and $f(B, A) = 0$.

So,

$$\begin{aligned} val(f) &= f(A, B) - f(B, A) && \text{by Lemma 2.3} \\ &= c(A, B) && \text{because } f(A, B) = c(A, B) \text{ and } f(B, A) = 0 \end{aligned}$$

By Proposition 2.1, we know that $R := S(A, B)$ is a cut, and by what we just showed, $val(f) = c(A, B) = c(R)$. It now follows from Corollary 2.4 that f is a maximum flow in (G, s, t, c).

Max-flow min-cut theorem

The maximum value of a flow in a network is equal to the minimum capacity of a cut in that network.

Proof. Let (G, s, t, c) be a network, and let f be a maximum flow in it (the existence of such a flow is guaranteed by Theorem 1.1). By Lemma 2.5, there exists a cut R in (G, s, t, c) such that $val(f) = c(R)$. Furthermore, for any cut R' in (G, s, t, c), Corollary 2.4 guarantees that $val(f) \leq c(R')$, and consequently, $c(R) \leq c(R')$; thus, R is a cut of minimum capacity in (G, s, t, c).

- Our next goal is to show how to find a maximum flow and a minimum cut in a network.

- Our next goal is to show how to find a maximum flow and a minimum cut in a network.
- The idea is to repeatedly find augmenting paths and update the flow (increasing its value).

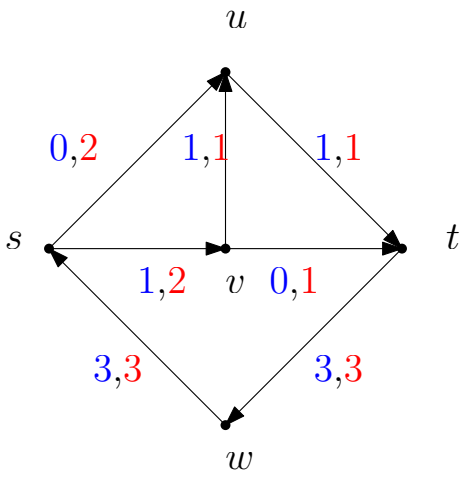
- Our next goal is to show how to find a maximum flow and a minimum cut in a network.
- The idea is to repeatedly find augmenting paths and update the flow (increasing its value).
- When no augmenting path exists, we instead find a cut whose capacity is equal to the value of our flow, which (by Corollary 2.4) guarantees that this cut is of minimum capacity.

Suppose that f is a flow in a network (G, s, t, c). We now either find an f-augmenting path in (G, s, t, c), or we find a cut whose capacity is $val(f)$, as follows:

1. Set $A := \{s\}$.
2. While $t \notin A$:
   1. Either find vertices $x \in A$ and $y \in V(G) \setminus A$ such that
      - $(x, y) \in E(G)$ and $f(x, y) < c(x, y)$, or
      - $(y, x) \in E(G)$ and $f(y, x) > 0$,

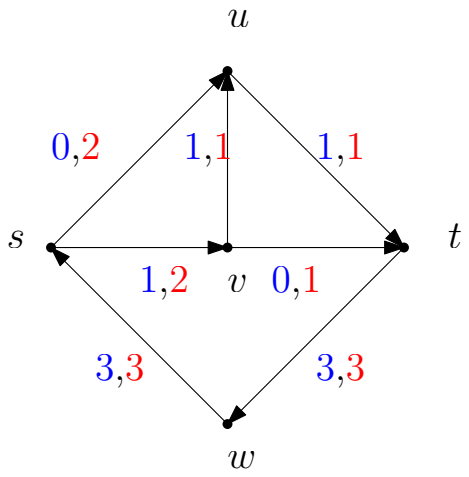
      or determine that such x and y do not exist.
   2. If we found x and y, then we set backpoint$(y) = x$, and we update $A := A \cup \{y\}$.
   3. Otherwise, we stop and return the cut $S(A, V(G) \setminus A)$.[1]
3. Construct an f-augmenting path by following backpoints starting from t, and return this path.

[1] In this case, an argument analogous to the proof of Lemma 2.5 guarantees that $c(A, V(G) \setminus A) = val(f)$.

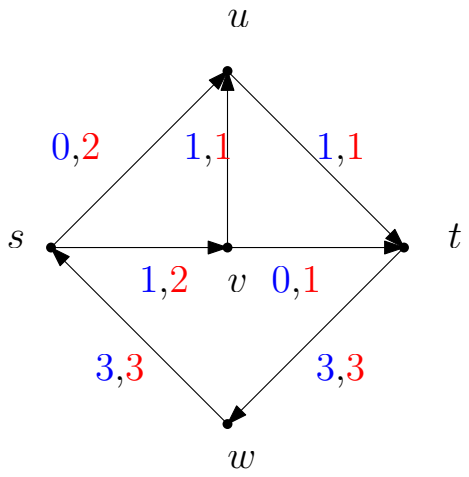


Example 3.1

Consider the flow f in the network (G, s, t, c) as in the figure above. Either find an f-augmenting path, or find a cut whose capacity is $val(f)$.

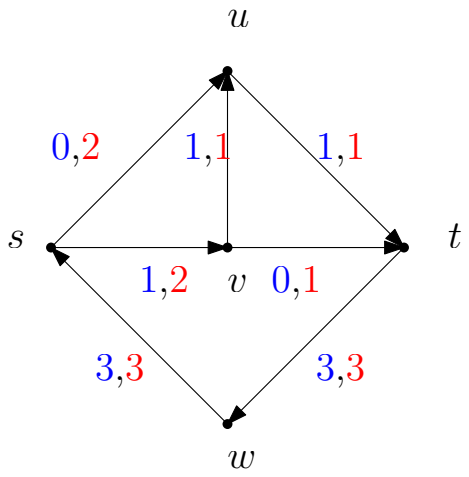


Solution. We begin with $A = \{s\}$. We now iterate several times.



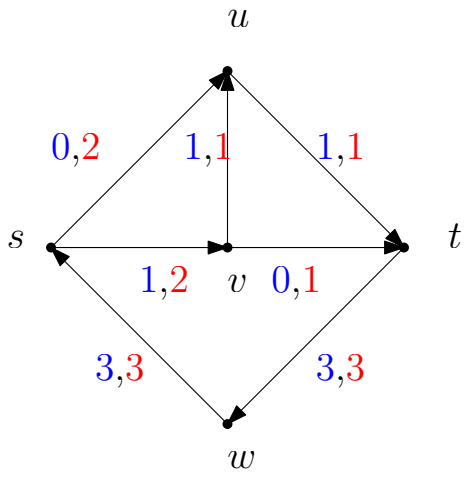
Solution. We begin with $A = \{s\}$. We now iterate several times.

1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.



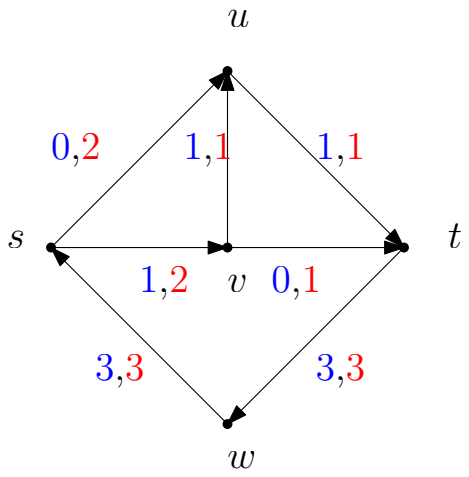
Solution. We begin with $A = \{s\}$. We now iterate several times.

1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.
2. We select $s \in A$ and $w \in V(G) \setminus A$, and we set $A := \{s, u, w\}$ and backpoint$(w) = s$.



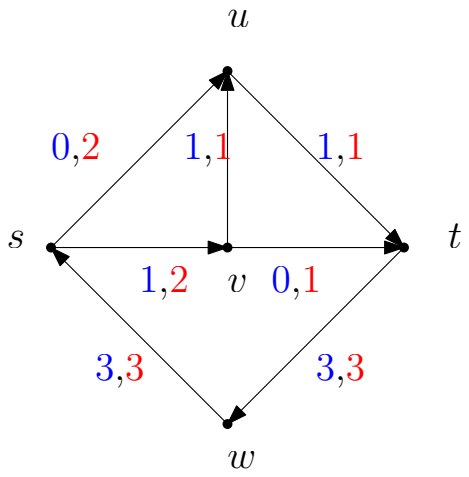
Solution. We begin with $A = \{s\}$. We now iterate several times.

1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.
2. We select $s \in A$ and $w \in V(G) \setminus A$, and we set $A := \{s, u, w\}$ and backpoint$(w) = s$.
3. We select $u \in A$ and $v \in V(G) \setminus A$, and we set $A := \{s, u, w, v\}$ and backpoint$(v) = u$.



Solution. We begin with $A = \{s\}$. We now iterate several times.

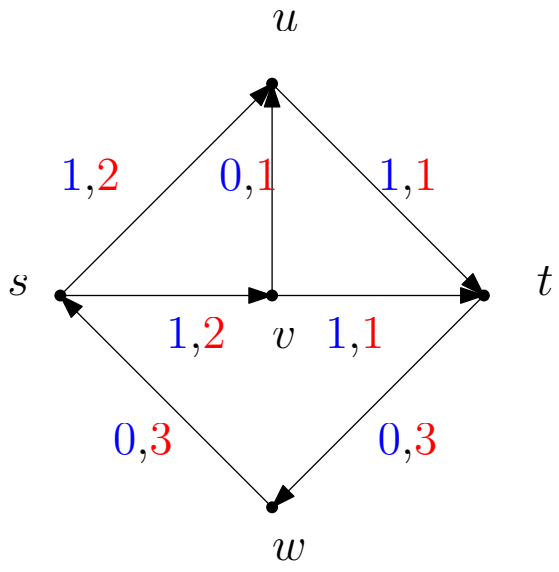
1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.
2. We select $s \in A$ and $w \in V(G) \setminus A$, and we set $A := \{s, u, w\}$ and backpoint$(w) = s$.
3. We select $u \in A$ and $v \in V(G) \setminus A$, and we set $A := \{s, u, w, v\}$ and backpoint$(v) = u$.
4. We select $v \in A$ and $t \in V(G) \setminus A$, and we set $A := \{s, u, w, v, t\}$ and backpoint$(t) = v$.



Solution. We begin with $A = \{s\}$. We now iterate several times.

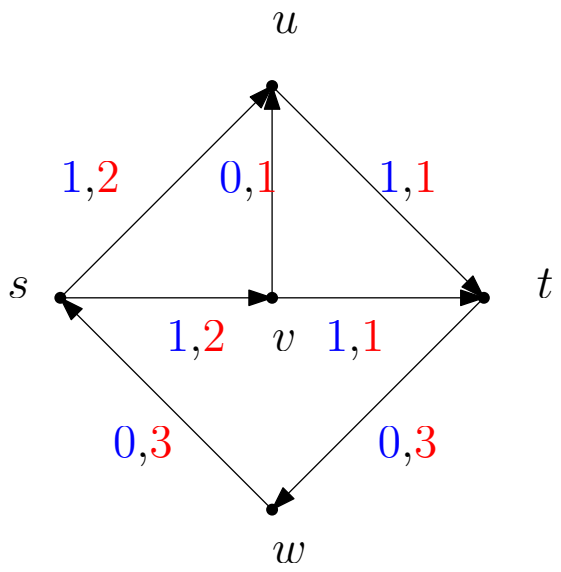
1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.
2. We select $s \in A$ and $w \in V(G) \setminus A$, and we set $A := \{s, u, w\}$ and backpoint$(w) = s$.
3. We select $u \in A$ and $v \in V(G) \setminus A$, and we set $A := \{s, u, w, v\}$ and backpoint$(v) = u$.
4. We select $v \in A$ and $t \in V(G) \setminus A$, and we set $A := \{s, u, w, v, t\}$ and backpoint$(t) = v$.

We now reconstruct our f-augmenting path: s, u, v, t. (It is easy to see that this really is an f-augmenting path.)

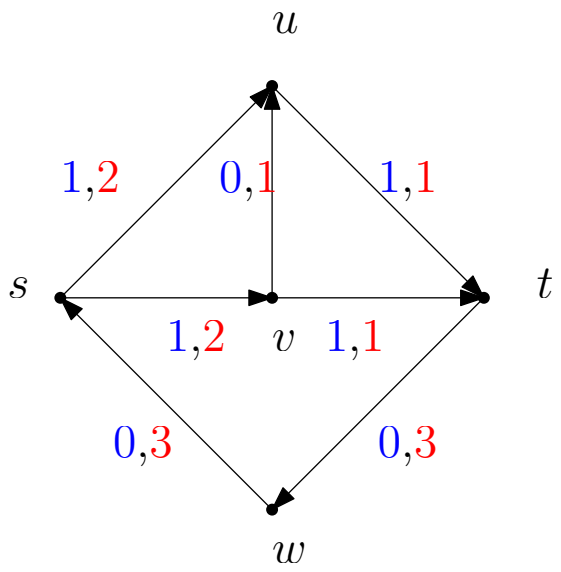


Example 3.2

Consider the flow f in the network (G, s, t, c) as in the figure above. Either find an f-augmenting path, or find a cut whose capacity is $val(f)$.

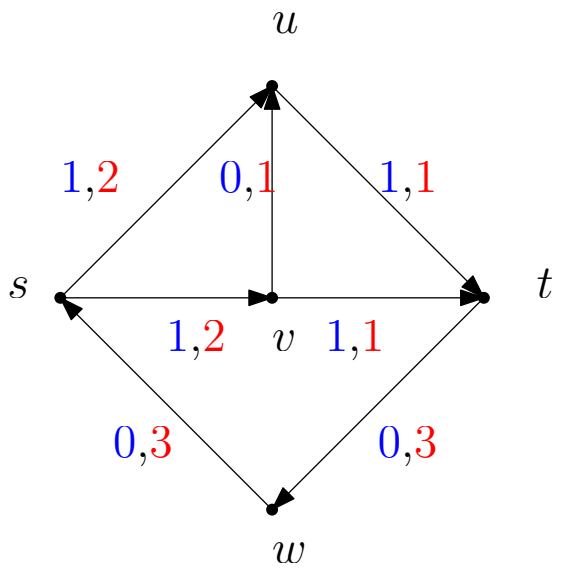


Solution. We begin with $A = \{s\}$. We now iterate several times.



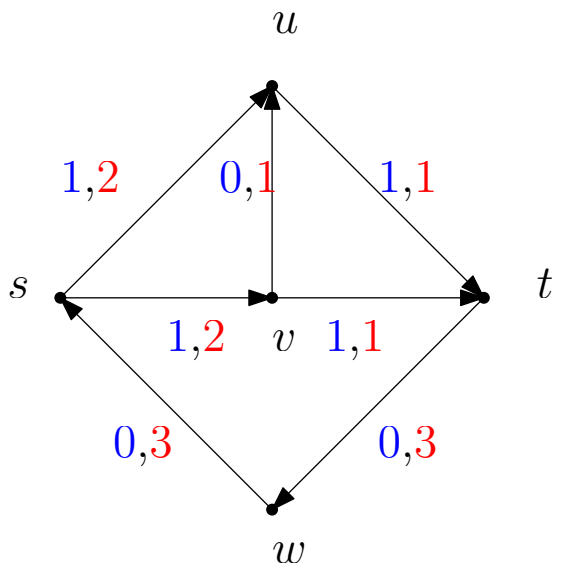
Solution. We begin with $A = \{s\}$. We now iterate several times.

1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.



Solution. We begin with $A = \{s\}$. We now iterate several times.

1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.
2. We select $s \in A$ and $v \in V(G) \setminus A$, and we set $A := \{s, u, v\}$ and backpoint$(v) = s$.



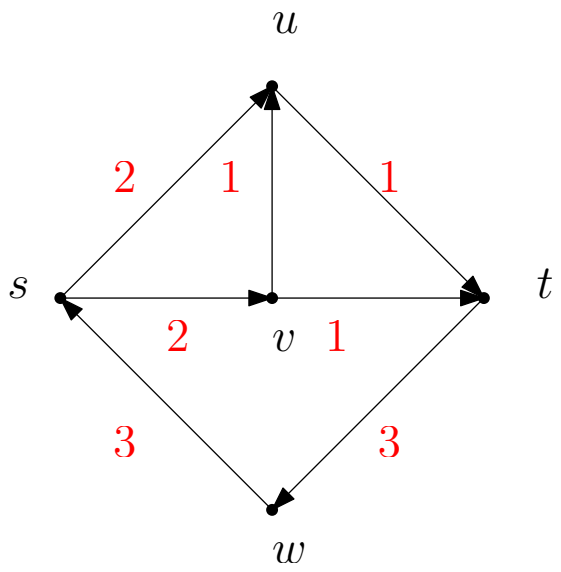
Solution. We begin with $A = \{s\}$. We now iterate several times.

1. We select $s \in A$ and $u \in V(G) \setminus A$, and we set $A := \{s, u\}$ and backpoint$(u) = s$.
2. We select $s \in A$ and $v \in V(G) \setminus A$, and we set $A := \{s, u, v\}$ and backpoint$(v) = s$.

There are now no further vertices that we can select, and $t \notin A$. We now see that $S(A, V(G) \setminus A) = \{(u, t), (v, t)\}$ is a cut whose capacity is 2, which is precisely equal to $val(f)$.

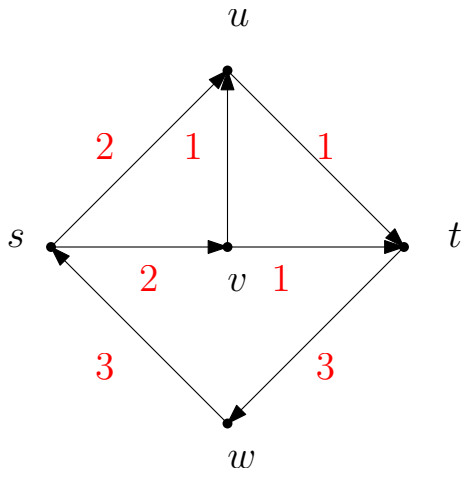
We now describe the Ford-Fulkerson algorithm, which finds a maximum flow in a network (G, s, t, c). Its steps are as follows:

1. Set $f(e) := 0$ for all $e \in E(G)$.
2. While there exists an f-augmenting path in the network:
   1. Find an f-augmenting path $v_0, \ldots, v_\ell$ (with $v_0 = s$ and $v_\ell = t$).
   2. Set
      - $\varepsilon_1 = \min\Big(\{c(v_i, v_{i+1}) - f(v_i, v_{i+1}) \mid 0 \leq i \leq \ell - 1, (v_i, v_{i+1}) \in E(G)\} \cup \{\infty\}\Big)$;
      - $\varepsilon_2 = \min\Big(\{f(v_{i+1}, v_i) \mid 0 \leq i \leq \ell - 1, (v_{i+1}, v_i) \in E(G)\} \cup \{\infty\}\Big)$;
      - $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\}$.
   3. Update f as follows:
      - $f(v_i, v_{i+1}) := f(v_i, v_{i+1}) + \varepsilon$ for all $i \in \{0, \ldots, \ell - 1\}$ such that $(v_i, v_{i+1}) \in E(G)$;
      - $f(v_{i+1}, v_i) := f(v_{i+1}, v_i) - \varepsilon$ for all $i \in \{0, \ldots, \ell - 1\}$ such that $(v_{i+1}, v_i) \in E(G)$.
3. Return f.

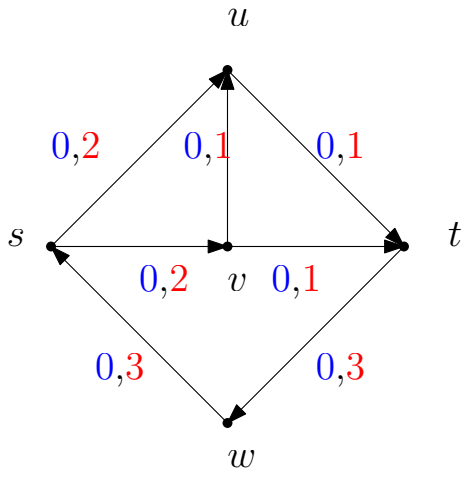


Example 3.3

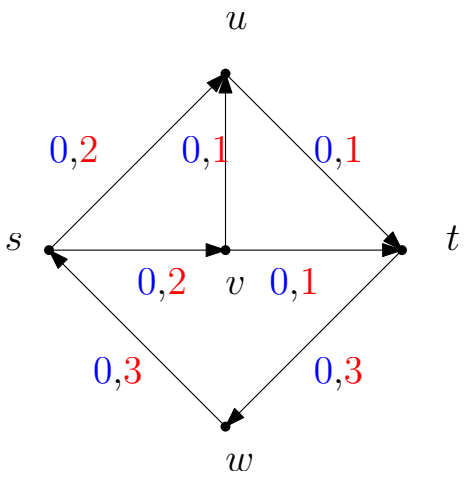
Find a maximum flow and an a cut of minimum capacity in the network represented in the figure above.



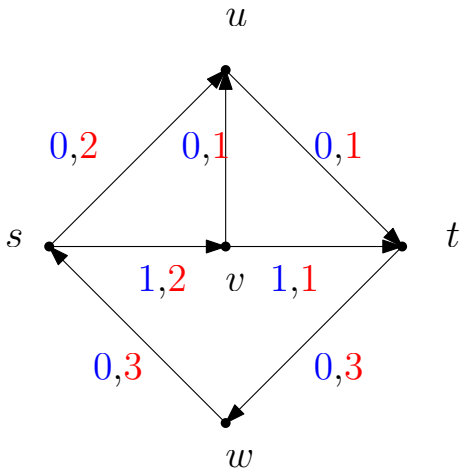
Solution. We first set $f(e) = 0$ for all $e \in E(G)$.

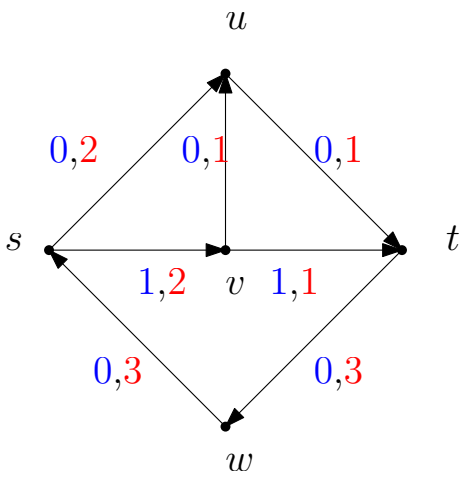


We now iterate several times.

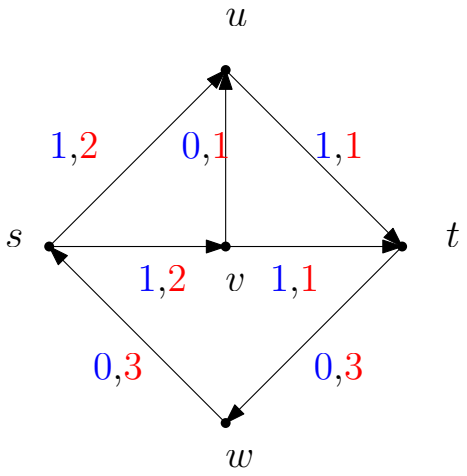


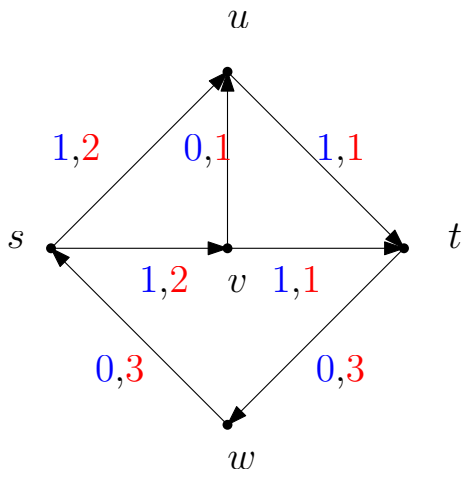
(1) We find an augmenting path s, v, t, we get $\varepsilon = 1$, and we update f as in the picture below.



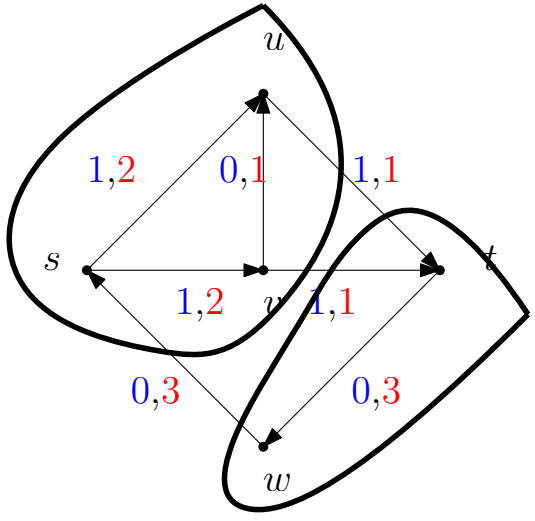


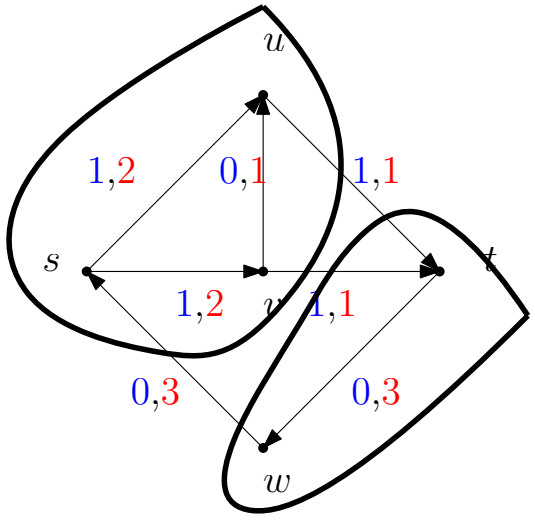
(2) We find an augmenting path s, u, t, we get $\varepsilon = 1$, and we update f as in the picture below.





(3) We find a cut $S(\{s, u, v\}, \{w, t\}) = \{(u, t), (v, t)\}$ of capacity is 2, which is precisely equal to $val(f)$.





- The flow f (blue) is a maximum flow.
- The cut $S(\{s, u, v\}, \{w, t\}) = \{(u, t), (v, t)\}$ is a minimum capacity cut.

- But is the Ford-Fulkerson algorithm correct?

- But is the Ford-Fulkerson algorithm correct?
- For this, it would need to have the following two properties:
  (1) the algorithm terminates for every input network (G, s, t, c);
  (2) if, given an input network (G, s, t, c), the algorithm returns a flow f, then f is indeed a maximum flow in (G, s, t, c).

- But is the Ford-Fulkerson algorithm correct?
- For this, it would need to have the following two properties:
  (1) the algorithm terminates for every input network (G, s, t, c);
  (2) if, given an input network (G, s, t, c), the algorithm returns a flow f, then f is indeed a maximum flow in (G, s, t, c).
- (2) is definitely holds: the algorithm returns f only if there is no f-augmenting path in the input network (G, s, t, c), and in this case, Lemma 2.5 guarantees that f is a maximum flow in (G, s, t, c).

- But is the Ford-Fulkerson algorithm correct?
- For this, it would need to have the following two properties:
  (1) the algorithm terminates for every input network (G, s, t, c);
  (2) if, given an input network (G, s, t, c), the algorithm returns a flow f, then f is indeed a maximum flow in (G, s, t, c).
- (2) is definitely holds: the algorithm returns f only if there is no f-augmenting path in the input network (G, s, t, c), and in this case, Lemma 2.5 guarantees that f is a maximum flow in (G, s, t, c).
- But (1) may fail!

- But is the Ford-Fulkerson algorithm correct?
- For this, it would need to have the following two properties:
  (1) the algorithm terminates for every input network (G, s, t, c);
  (2) if, given an input network (G, s, t, c), the algorithm returns a flow f, then f is indeed a maximum flow in (G, s, t, c).
- (2) is definitely holds: the algorithm returns f only if there is no f-augmenting path in the input network (G, s, t, c), and in this case, Lemma 2.5 guarantees that f is a maximum flow in (G, s, t, c).
- But (1) may fail!
  - The good news is that this is only possible if some of the capacities in the network are irrational.
  - If all capacities are rational, then the algorithm terminates and correctly outputs a maximum flow.

- But is the Ford-Fulkerson algorithm correct?
- For this, it would need to have the following two properties:
  (1) the algorithm terminates for every input network (G, s, t, c);
  (2) if, given an input network (G, s, t, c), the algorithm returns a flow f, then f is indeed a maximum flow in (G, s, t, c).
- (2) is definitely holds: the algorithm returns f only if there is no f-augmenting path in the input network (G, s, t, c), and in this case, Lemma 2.5 guarantees that f is a maximum flow in (G, s, t, c).
- But (1) may fail!
  - The good news is that this is only possible if some of the capacities in the network are irrational.
  - If all capacities are rational, then the algorithm terminates and correctly outputs a maximum flow.
  - We first deal with the case when the capacities are integers.

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

Proof.

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

Proof. If we begin with an integer flow in the network (G, s, t, c), and we find an augmenting path, then since all capacities are integers, the number ε (defined as in the description of the Ford-Fulkerson algorithm) will be a positive integer;

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

Proof. If we begin with an integer flow in the network (G, s, t, c), and we find an augmenting path, then since all capacities are integers, the number ε (defined as in the description of the Ford-Fulkerson algorithm) will be a positive integer; so, the updated flow will still be an integer flow, since the flow through an edge can either remain unchanged, or increase by ε, or decrease by ε.

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

Proof (continued). Now, the initial flow created by the Ford-Fulkerson algorithm for the network (G, s, t, c) is the zero-flow (and so in particular, an integer flow), and by what we just proved, after each iteration, the new flow is still an integer flow.

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

Proof (continued). Now, the initial flow created by the Ford-Fulkerson algorithm for the network (G, s, t, c) is the zero-flow (and so in particular, an integer flow), and by what we just proved, after each iteration, the new flow is still an integer flow. The algorithm terminates because after each iteration, the value of the flow increases by a positive integer (namely, by the ε that we compute for that iteration), and the maximum value of the flow is bounded (e.g. by the sum of capacities), and so there can be only finitely many iterations.

Theorem 3.4

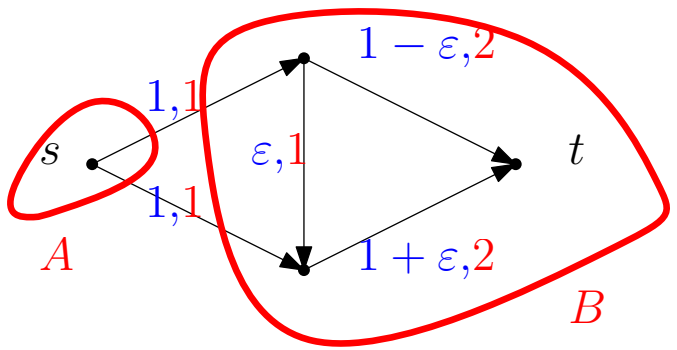
Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

Proof (continued). The fact that the algorithm returns a correct answer follows from its stopping criterion: the algorithm terminates and returns a flow f once there are no f-augmenting paths, and in this case, Lemma 2.5 from Lecture Notes 6 guarantees that f is a maximum flow.

- Theorem 3.4 does **not** state that every maximum flow in a network with integer capacities is an integer flow.

- Theorem 3.4 does **not** state that every maximum flow in a network with integer capacities is an integer flow.
- It merely guarantees that at least one maximum flow in such a network is an integer flow.

- Theorem 3.4 does **not** state that every maximum flow in a network with integer capacities is an integer flow.
- It merely guarantees that at least one maximum flow in such a network is an integer flow.
- For instance, the flow in the picture below is maximum for any value of $\varepsilon \in [0, 1]$, but only two values of ε (namely, $\varepsilon = 0$ and $\varepsilon = 1$) yield an integer flow.



$\varepsilon \in [0, 1]$

$val(f) = 2$

$c(A, B) = 2$

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

- Theorem 3.4 is important for certain theoretical applications (e.g. matching theory), as well for certain practical applications.

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

- Theorem 3.4 is important for certain theoretical applications (e.g. matching theory), as well for certain practical applications.
  - Consider, for example, a network that models a transportation network of trucks, where the capacity of a truck is the number of containers that it can carry.

Theorem 3.4

Let (G, s, t, c) be a network in which all capacities are non-negative integers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is a non-negative integer. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative integers.

- Theorem 3.4 is important for certain theoretical applications (e.g. matching theory), as well for certain practical applications.
  - Consider, for example, a network that models a transportation network of trucks, where the capacity of a truck is the number of containers that it can carry.
  - Certainly, we would want a maximum flow that is an integer flow. (A truck should not transport $\frac{7}{3}$ or $\sqrt[3]{\pi}$ containers!)

Theorem 3.5

Let (G, s, t, c) be a network in which all capacities are non-negative rational numbers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is an non-negative rational number. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative rational numbers.

Theorem 3.5

Let (G, s, t, c) be a network in which all capacities are non-negative rational numbers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is an non-negative rational number. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative rational numbers.

Proof (outline). Let d be a positive integer such that all capacities in (G, s, t, c) are integer multiples of $\frac{1}{d}$. (To see that d exists, we can first write all capacities in (G, s, t, c) as fractions, and then we take d to be the least common multiple of the denominators of the capacities.)

Theorem 3.5

Let (G, s, t, c) be a network in which all capacities are non-negative rational numbers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is an non-negative rational number. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative rational numbers.

Proof (outline). Let d be a positive integer such that all capacities in (G, s, t, c) are integer multiples of $\frac{1}{d}$. (To see that d exists, we can first write all capacities in (G, s, t, c) as fractions, and then we take d to be the least common multiple of the denominators of the capacities.) Now the proof is completely analogous to that of Theorem 3.4, except that instead of integers, we have integer multiples of $\frac{1}{d}$ (for flows and capacities) throughout.

Theorem 3.5

Let (G, s, t, c) be a network in which all capacities are non-negative rational numbers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is an non-negative rational number. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative rational numbers.

Theorem 3.5

Let (G, s, t, c) be a network in which all capacities are non-negative rational numbers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is an non-negative rational number. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative rational numbers.

- The key point of the proof of Theorem 3.5 is that there exists some positive integer d such that in each iteration, the value of the flow increases by at least $\frac{1}{d}$, and so there cannot be infinitely many iterations.

Theorem 3.5

Let (G, s, t, c) be a network in which all capacities are non-negative rational numbers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is an non-negative rational number. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative rational numbers.

- The key point of the proof of Theorem 3.5 is that there exists some positive integer d such that in each iteration, the value of the flow increases by at least $\frac{1}{d}$, and so there cannot be infinitely many iterations.
- If (some of) our capacities are irrational, such a d need not exist.

Theorem 3.5

Let (G, s, t, c) be a network in which all capacities are non-negative rational numbers. Then, for input (G, s, t, c), the Ford-Fulkerson algorithm terminates and outputs a maximum flow, and furthermore, the output flow through each edge is an non-negative rational number. In particular, some maximum flow in (G, s, t, c) has the property that flows through all edges are non-negative rational numbers.

- The key point of the proof of Theorem 3.5 is that there exists some positive integer d such that in each iteration, the value of the flow increases by at least $\frac{1}{d}$, and so there cannot be infinitely many iterations.
- If (some of) our capacities are irrational, such a d need not exist.
    - For a concrete example, see the Lecture Notes.