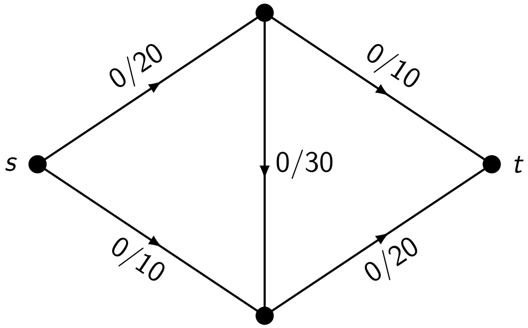
The Ford-Fulkerson algorithm
COMS20010 2020, Video 8-4

John Lapinskas, University of Bristol

The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

Now the definition of value is sorted out, how do we solve the problem?

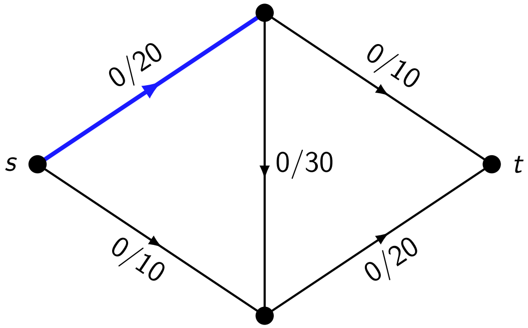
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

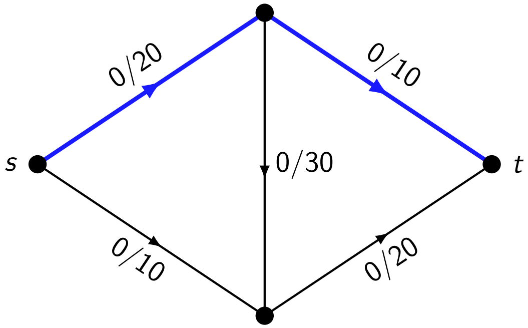
Now the definition of value is sorted out, how do we solve the problem?

How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



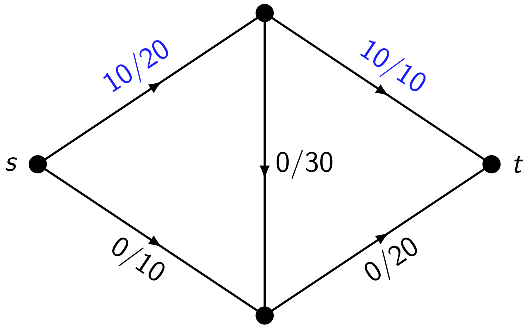
Now the definition of value is sorted out, how do we solve the problem?

How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



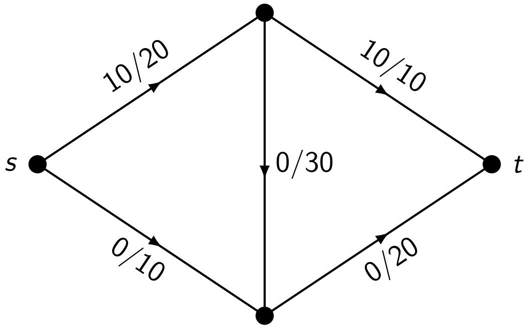
Now the definition of value is sorted out, how do we solve the problem?

How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



Now the definition of value is sorted out, how do we solve the problem?

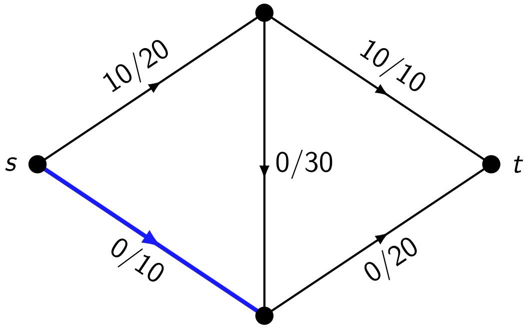
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

Now the definition of value is sorted out, how do we solve the problem?

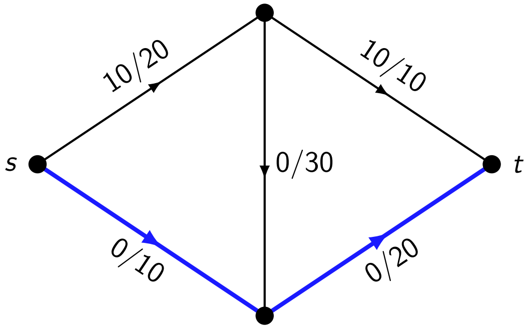
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

Now the definition of value is sorted out, how do we solve the problem?

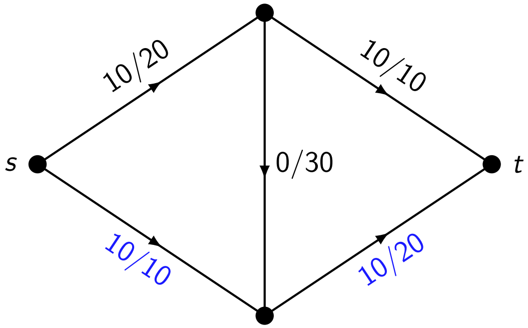
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

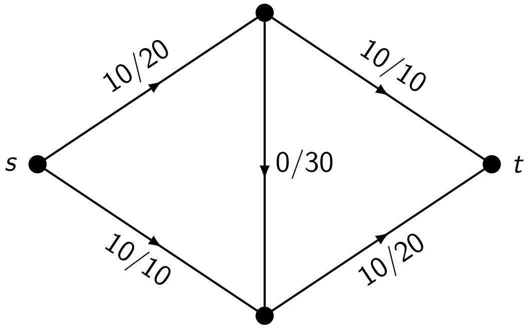
Now the definition of value is sorted out, how do we solve the problem?

How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



Now the definition of value is sorted out, how do we solve the problem?

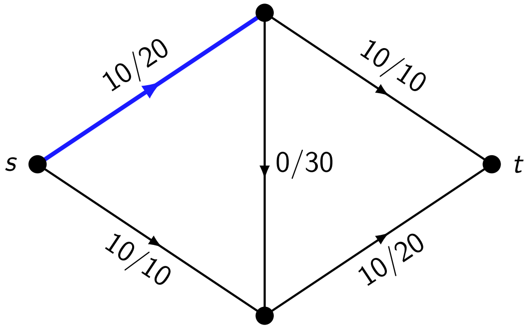
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

Now the definition of value is sorted out, how do we solve the problem?

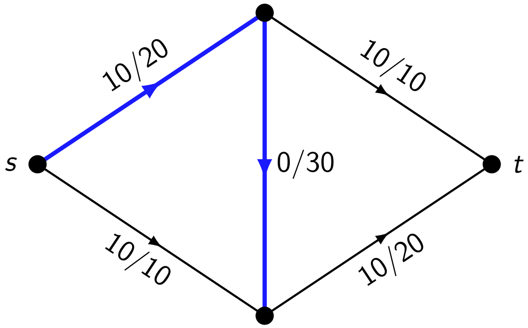
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

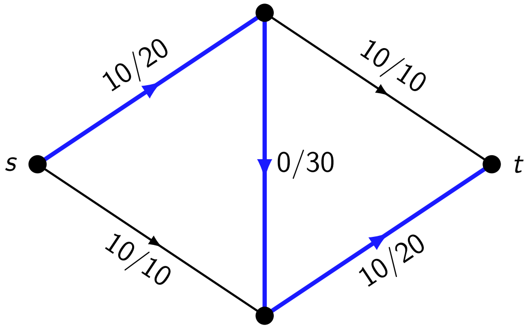
Now the definition of value is sorted out, how do we solve the problem?

How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



Now the definition of value is sorted out, how do we solve the problem?

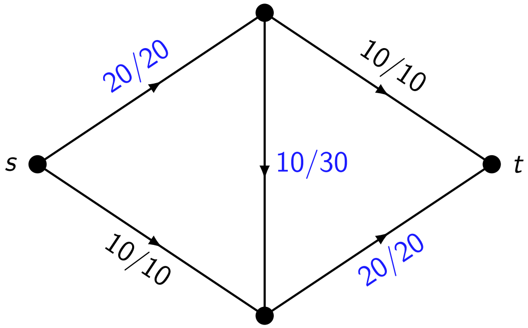
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

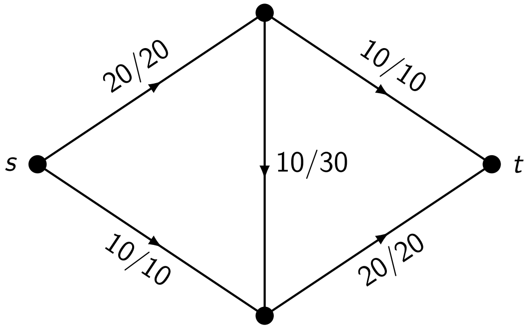
Now the definition of value is sorted out, how do we solve the problem?

How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



Now the definition of value is sorted out, how do we solve the problem?

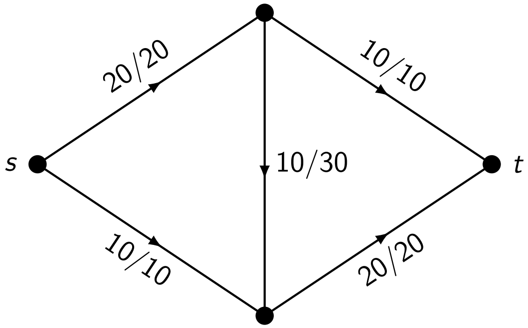
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



This flow has value $20 + 10 = 30$, which is best possible.
So a greedy approach can work...

Now the definition of value is sorted out, how do we solve the problem?

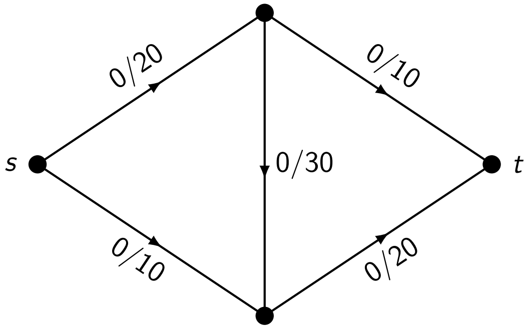
How about a greedy approach? Repeatedly find paths from s to t with unused capacity and "push" more flow down them.



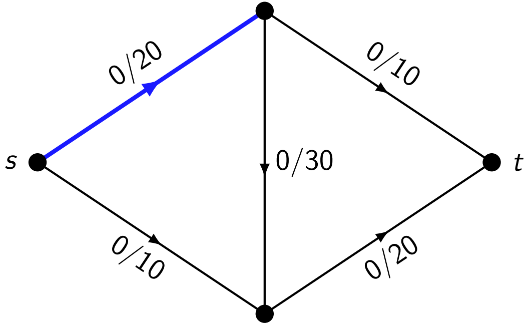
This flow has value $20 + 10 = 30$, which is best possible.
So a greedy approach can work... but it can also fail.

The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

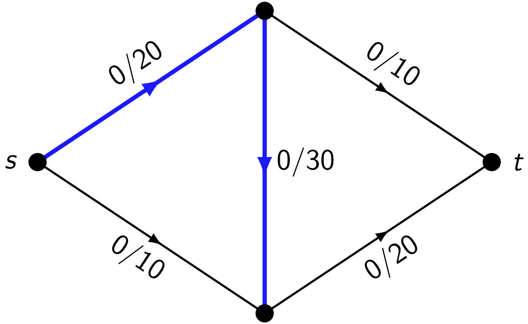
What if we'd chosen this path first instead?



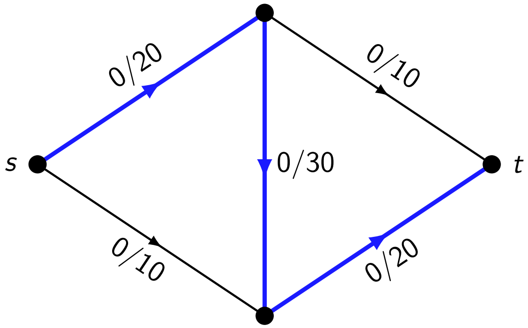
What if we'd chosen this path first instead?



What if we'd chosen this path first instead?

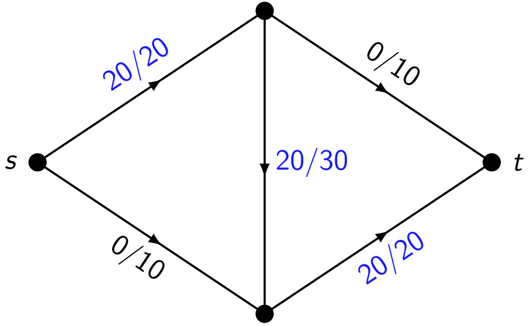


What if we'd chosen this path first instead?

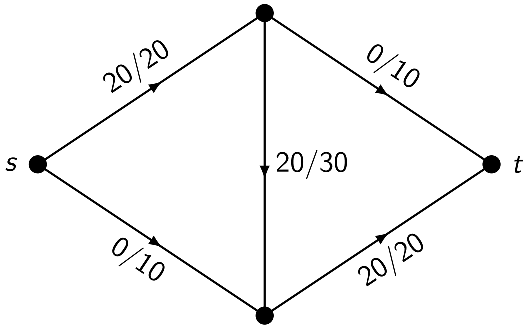


The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

What if we'd chosen this path first instead?

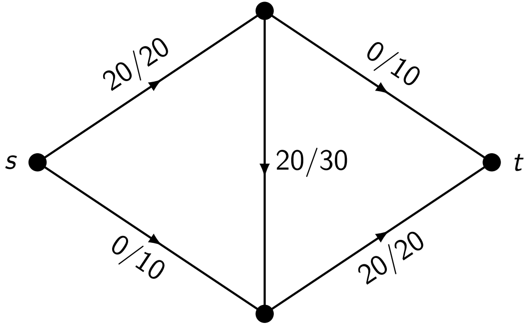


What if we'd chosen this path first instead?



Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

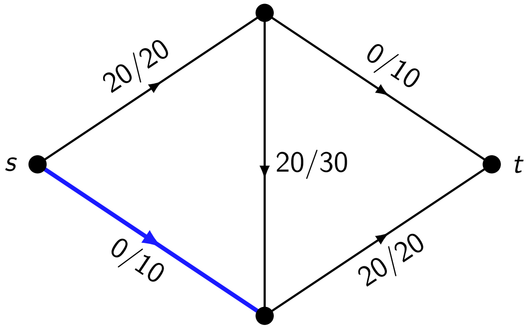
What if we'd chosen this path first instead?



Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

What if we allow ourselves to push flow *backwards* along a path?

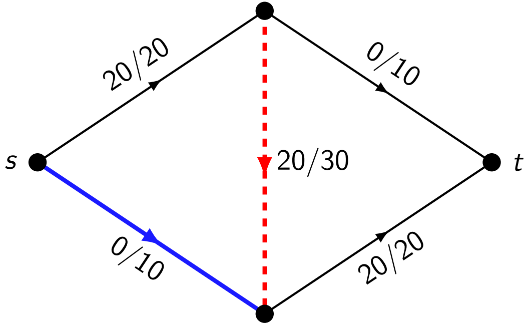
What if we'd chosen this path first instead?



Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

What if we allow ourselves to push flow *backwards* along a path?

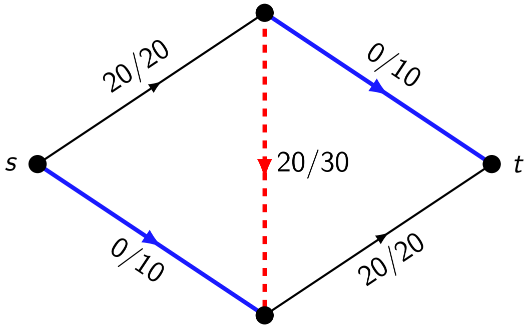
What if we'd chosen this path first instead?



Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

What if we allow ourselves to push flow *backwards* along a path?

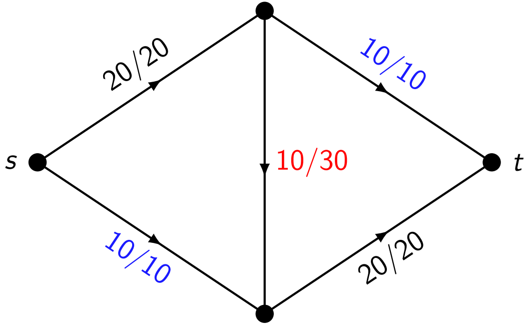
What if we'd chosen this path first instead?



Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

What if we allow ourselves to push flow *backwards* along a path?

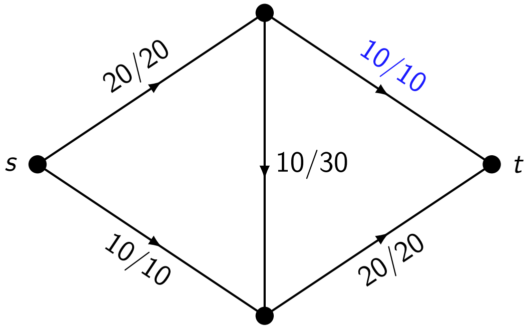
What if we'd chosen this path first instead?



Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

What if we allow ourselves to push flow *backwards* along a path?

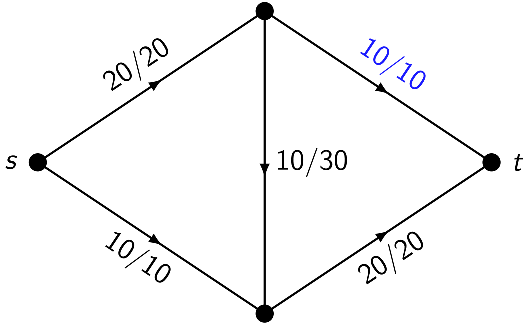
What if we'd chosen this path first instead?



Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

What if we allow ourselves to push flow *backwards* along a path?
We get a maximum flow!

What if we'd chosen this path first instead?



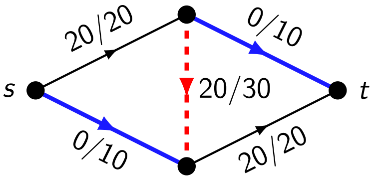
Now there are no more paths from s to t with spare capacity, but our flow only has value 20...

What if we allow ourselves to push flow *backwards* along a path?
We get a maximum flow! Now to generalise this...

A **flow** is a function $f\colon E \to \mathbb{R}$ such that for all $e \in E$ and $v \in V \setminus \{s,t\}$:

- $0 \le f(e) \le c(e)$;
- $f^+(v) := \sum_{u \in N^-(v)} f(u,v) = \sum_{w \in N^+(v)} f(v,w) =: f^-(v)$.

The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

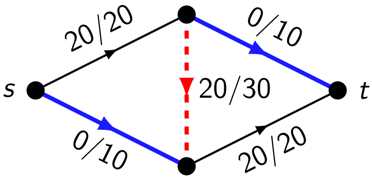


We want to say: an augmenting path for a flow f is an **undirected** path from s to t which we can push flow along. So forward edges e have $f(e) < c(e)$, and backward edges e have $f(e) > 0$.

A **flow** is a function $f\colon E \to \mathbb{R}$ such that for all $e \in E$ and $v \in V \setminus \{s,t\}$:

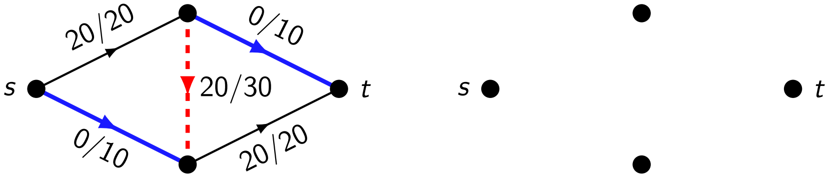
- $0 \le f(e) \le c(e)$;
- $f^+(v) := \sum_{u \in N^-(v)} f(u,v) = \sum_{w \in N^+(v)} f(v,w) =: f^-(v)$.

The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

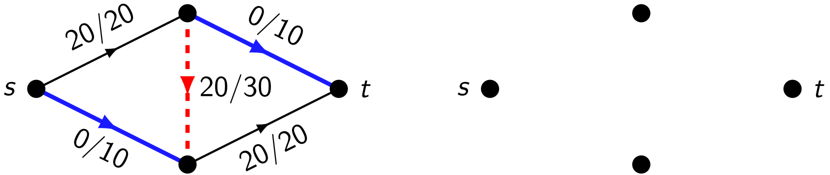


We want to say: an augmenting path for a flow f is an **undirected** path from s to t which we can push flow along. So forward edges e have $f(e) < c(e)$, and backward edges e have $f(e) > 0$.

But there's an annoying technicality with bidirected edges...

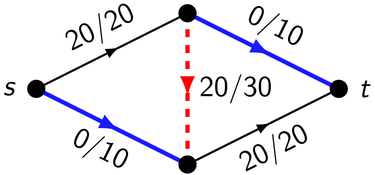


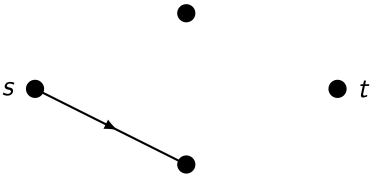
We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:



We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

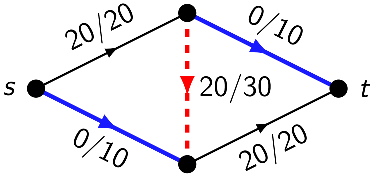
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.

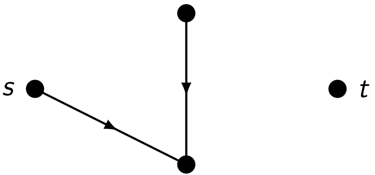




We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

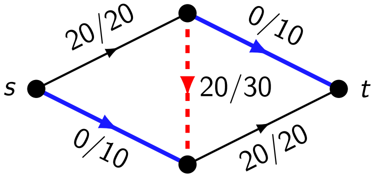
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.

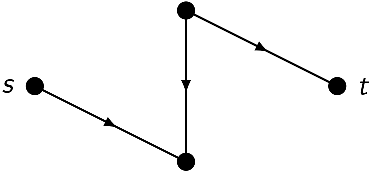




We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

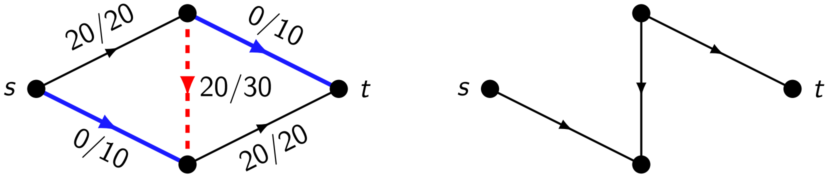
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.





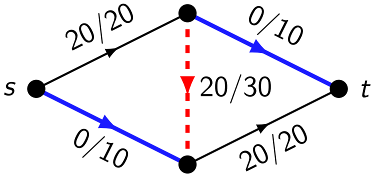
We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

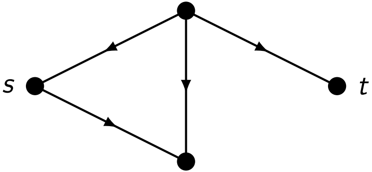
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.



We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

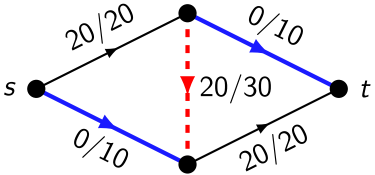
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

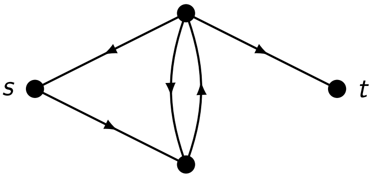




We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

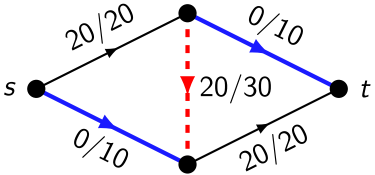
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

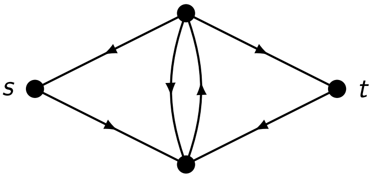




We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

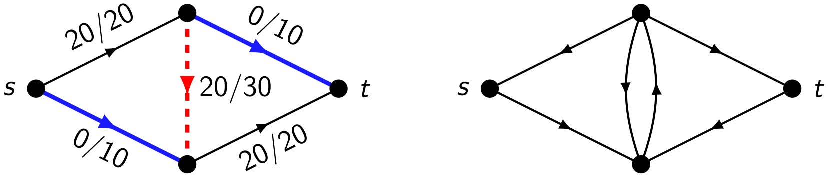
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)





We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

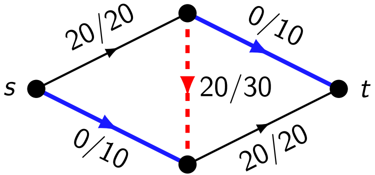
- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

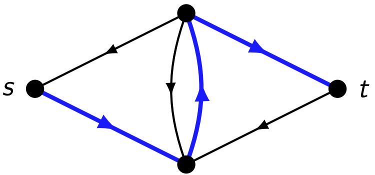


We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

An **augmenting path** P is a directed path from s to t in G_f.

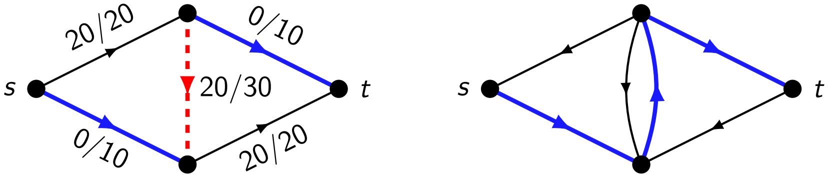




We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

An **augmenting path** P is a directed path from s to t in G_f.

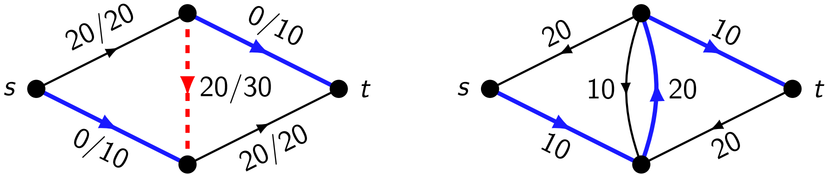


We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

An **augmenting path** P is a directed path from s to t in G_f.

The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.

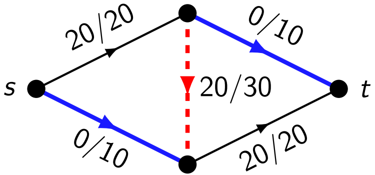


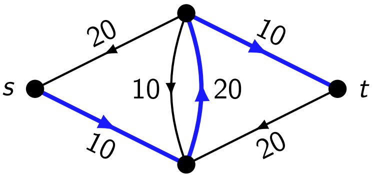
We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

An **augmenting path** P is a directed path from s to t in G_f.

The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.





We define the **residual graph** G_f of (G, c, s, t) on $V(G)$ as follows:

- If $(u, v) \in E(G)$ with $f(e) < c(e)$, add (u, v) to $E(G_f)$; call this a **forward edge**.
- If $(u, v) \in E(G)$ with $f(e) > 0$, add (v, u) to $E(G_f)$; call this a **backward edge**. (An edge can be both forward and backward!)

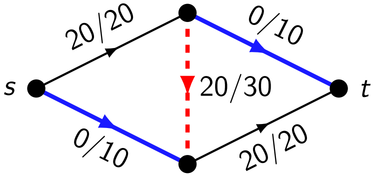
An **augmenting path** P is a directed path from s to t in G_f.

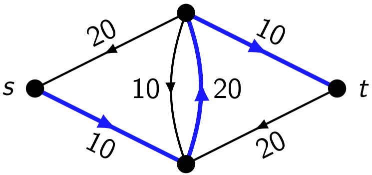
The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.

The **residual capacity** of P is the minimum residual capacity of its edges. (This is the amount of flow we can push through P.)

$(u, v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u, v) \in E(G_f)$.
$(u, v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v, u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.

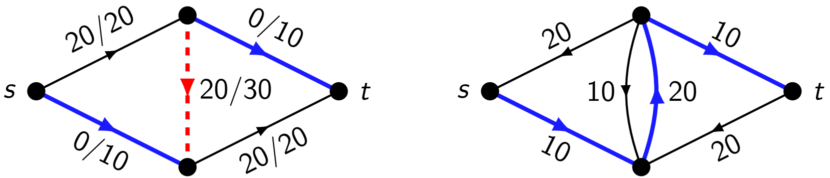




Define $\texttt{Push}(G, c, s, t, f, P)$ as follows:

$(u, v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u, v) \in E(G_f)$.
$(u, v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v, u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.

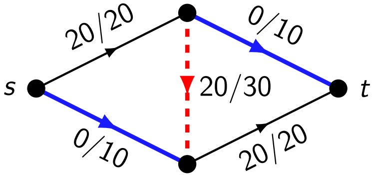


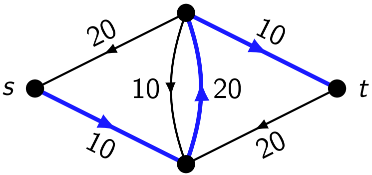
Define $\texttt{Push}(G, c, s, t, f, P)$ as follows:

- Let C be the residual capacity of P. (Here C is 10.)

$(u, v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u, v) \in E(G_f)$.
$(u, v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v, u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.



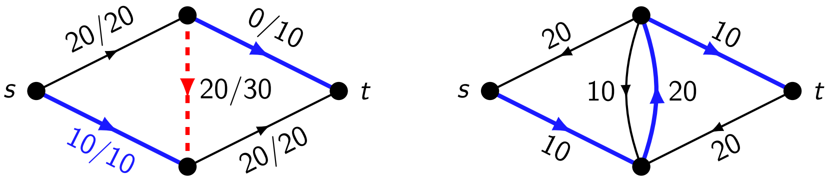


Define $\texttt{Push}(G, c, s, t, f, P)$ as follows:

- Let C be the residual capacity of P. (Here C is 10.)
- For each edge (u, v) of P: if $c(u, v) - f(u, v) \geq C$, then add C to $f(u, v)$; otherwise, we have $f(v, u) \geq C$, so subtract C from $f(v, u)$.

$(u, v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u, v) \in E(G_f)$.
$(u, v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v, u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.

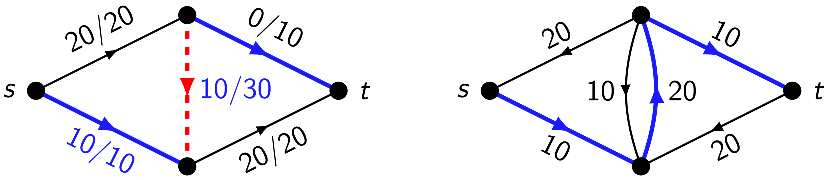


Define $\texttt{Push}(G, c, s, t, f, P)$ as follows:

- Let C be the residual capacity of P. (Here C is 10.)
- For each edge (u, v) of P: if $c(u, v) - f(u, v) \geq C$, then add C to $f(u, v)$; otherwise, we have $f(v, u) \geq C$, so subtract C from $f(v, u)$.

$(u, v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u, v) \in E(G_f)$.
$(u, v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v, u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.

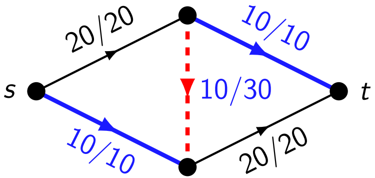


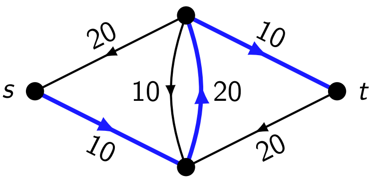
Define $\texttt{Push}(G, c, s, t, f, P)$ as follows:

- Let C be the residual capacity of P. (Here C is 10.)
- For each edge (u, v) of P: if $c(u, v) - f(u, v) \geq C$, then add C to $f(u, v)$; otherwise, we have $f(v, u) \geq C$, so subtract C from $f(v, u)$.

$(u,v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u,v) \in E(G_f)$.
$(u,v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v,u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u,v) in G_f is $\max\{c(u,v) - f(u,v), f(v,u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.



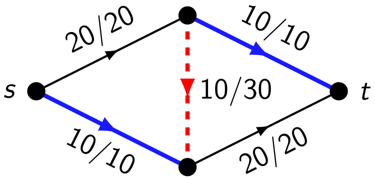


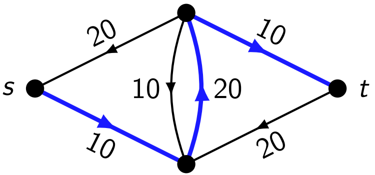
Define $\mathtt{Push}(G, c, s, t, f, P)$ as follows:

- Let C be the residual capacity of P. (Here C is 10.)
- For each edge (u,v) of P: if $c(u,v) - f(u,v) \geq C$, then add C to $f(u,v)$; otherwise, we have $f(v,u) \geq C$, so subtract C from $f(v,u)$.

$(u, v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u, v) \in E(G_f)$.
$(u, v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v, u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u, v) in G_f is $\max\{c(u, v) - f(u, v), f(v, u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.



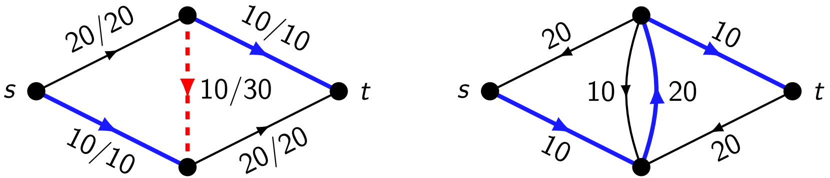


Define $\texttt{Push}(G, c, s, t, f, P)$ as follows:

- Let C be the residual capacity of P. (Here C is 10.)
- For each edge (u, v) of P: if $c(u, v) - f(u, v) \geq C$, then add C to $f(u, v)$; otherwise, we have $f(v, u) \geq C$, so subtract C from $f(v, u)$.

$(u,v) \in E(G)$ with $f(e) < c(e)$ yields a **forward edge** $(u,v) \in E(G_f)$.
$(u,v) \in E(G)$ with $f(e) > 0$ yields **backward edge** $(v,u) \in E(G_f)$.

An **augmenting path** P is a directed path from s to t in G_f.
The **residual capacity** of (u,v) in G_f is $\max\{c(u,v) - f(u,v), f(v,u)\}$.
The **residual capacity** of P is the minimum residual capacity of its edges.



Define $\texttt{Push}(G, c, s, t, f, P)$ as follows:

- Let C be the residual capacity of P. (Here C is 10.)
- For each edge (u,v) of P: if $c(u,v) - f(u,v) \geq C$, then add C to $f(u,v)$; otherwise, we have $f(v,u) \geq C$, so subtract C from $f(v,u)$.

Lemma 3: $\texttt{Push}(G, c, s, t, f, P)$ returns a new flow f', with value $v(f') = v(f) + C$, in $O(|V(G)|)$ time. $\square$

The Ford-Fulkerson Algorithm

Algorithm: FordFulkerson

Input : A (weakly connected) flow network (G, c, s, t).
Output : A flow f with no augmenting paths.

```
begin
    Construct the flow f with f(e) = 0 for all e ∈ E(G).
    Construct the residual graph G_f.
    while G_f contains a path P from s to t do
        Find P using depth-first (or breadth-first) search.
        Update f ← Push(G, c, s, t, f, P).
        Update G_f on the edges of P.
    Return f.
```

The Ford-Fulkerson Algorithm

Algorithm: FordFulkerson

Input : A (weakly connected) flow network (G, c, s, t).
Output : A flow f with no augmenting paths.

```
1 begin
2     Construct the flow f with f(e) = 0 for all e ∈ E(G).
3     Construct the residual graph G_f.
4     while G_f contains a path P from s to t do
5         Find P using depth-first (or breadth-first) search.
6         Update f ← Push(G, c, s, t, f, P).
7         Update G_f on the edges of P.
8     Return f.
```

By Lemma 3, every iteration of 4–7 increases $v(f)$ by at least 1. So if f^* is a maximum flow, there are at most $v(f^*)$ iterations in total.

The Ford-Fulkerson Algorithm

Algorithm: FORDFULKERSON

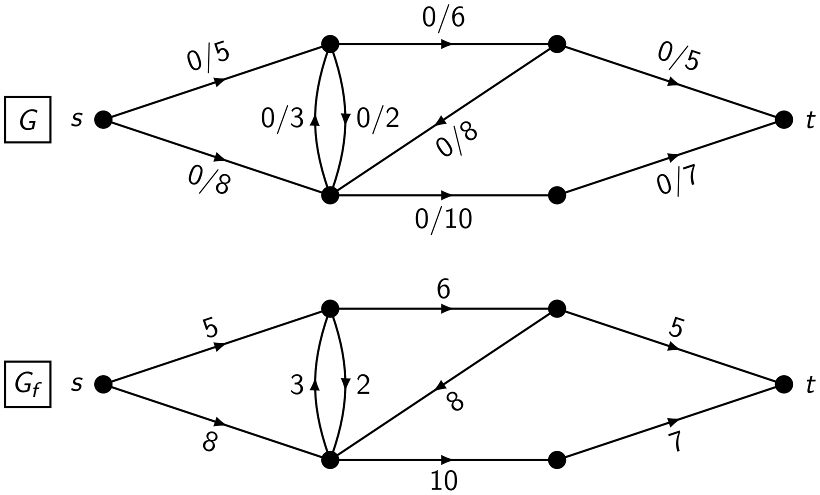
Input : A (weakly connected) flow network (G, c, s, t).
Output : A flow f with no augmenting paths.

```
1 begin
2     Construct the flow f with f(e) = 0 for all e ∈ E(G).
3     Construct the residual graph G_f.
4     while G_f contains a path P from s to t do
5         Find P using depth-first (or breadth-first) search.
6         Update f ← Push(G, c, s, t, f, P).
7         Update G_f on the edges of P.
8     Return f.
```

By Lemma 3, every iteration of 4–7 increases $v(f)$ by at least 1. So if f^* is a maximum flow, there are at most $v(f^*)$ iterations in total.

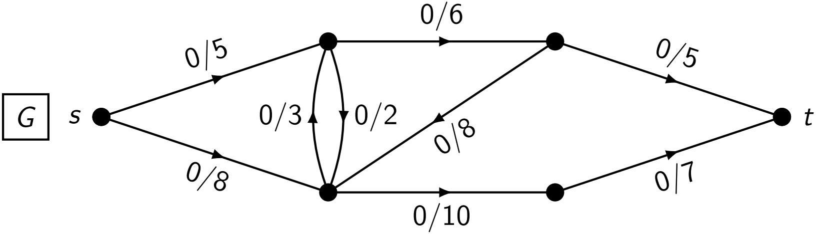
Every step takes $O(|E|)$ time or $O(|V|)$ time, and since G is weakly connected we have $|V| = O(|E|)$. So the running time is $O(v(f^*)|E|)$.

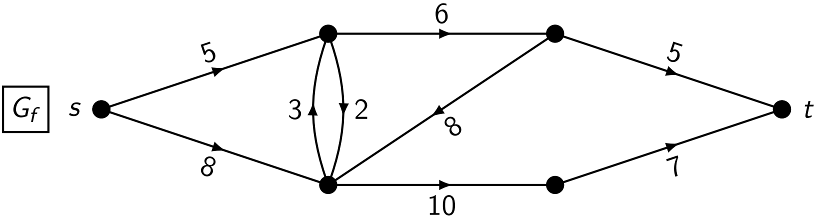
Worked example



Initialise flow and construct G_f.

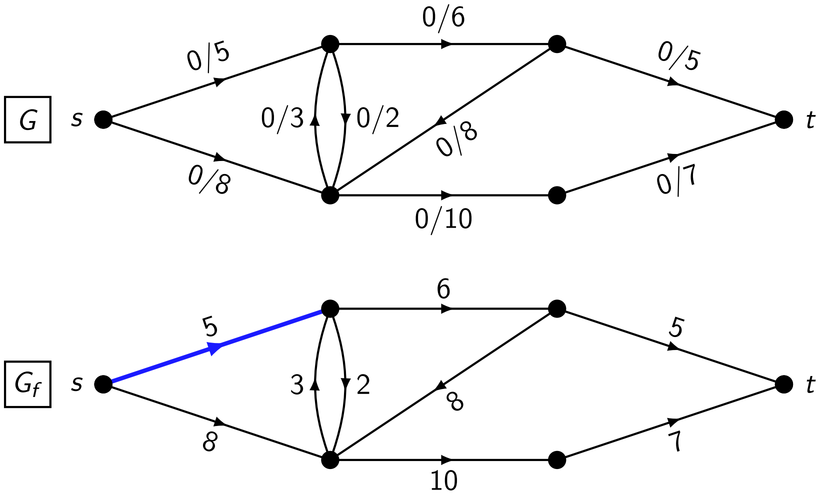
Worked example





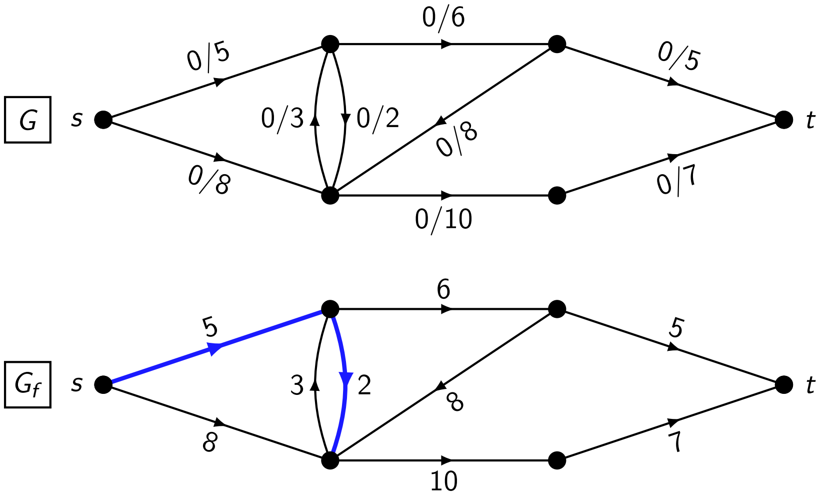
Apply depth-first search to find an augmenting path in G_f.

Worked example



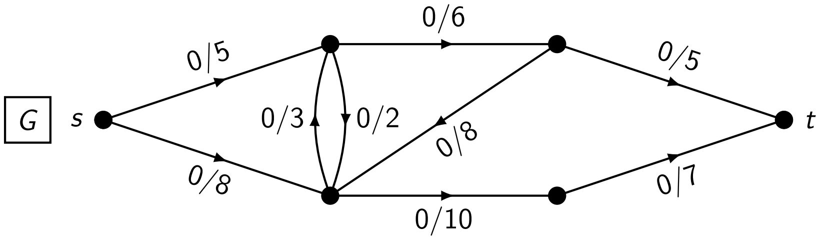
Apply depth-first search to find an augmenting path in G_f.

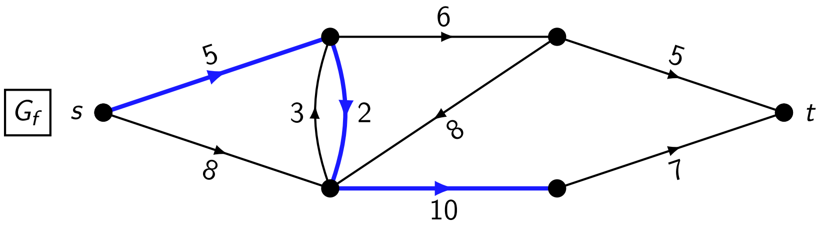
Worked example



Apply depth-first search to find an augmenting path in G_f.

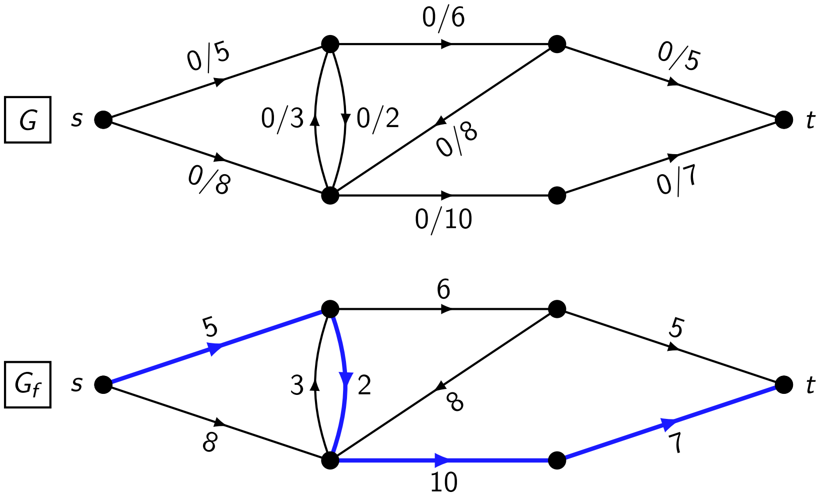
Worked example





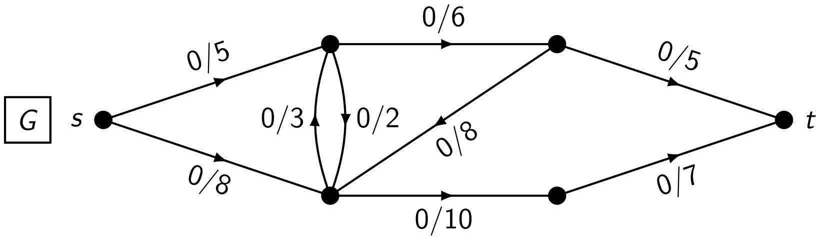
Apply depth-first search to find an augmenting path in G_f.

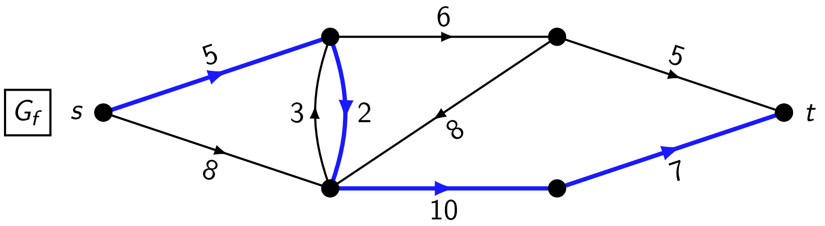
Worked example



Apply depth-first search to find an augmenting path in G_f.

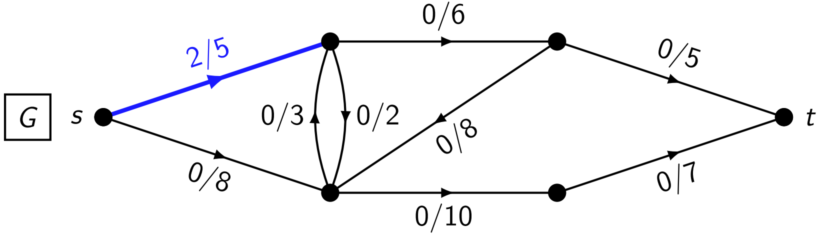
Worked example

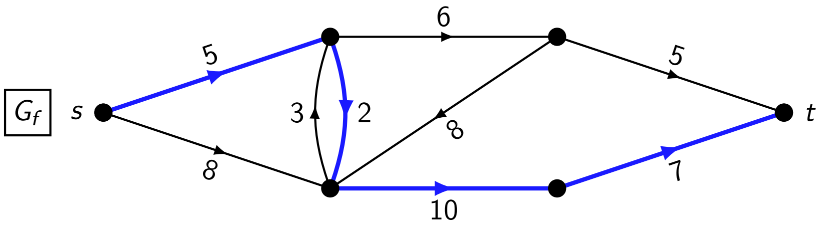




Push flow along the path. (This path has residual capacity 2.)

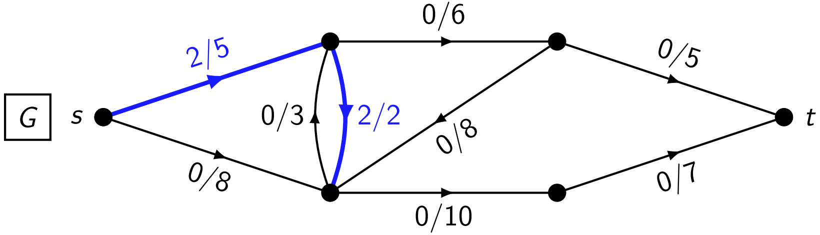
Worked example

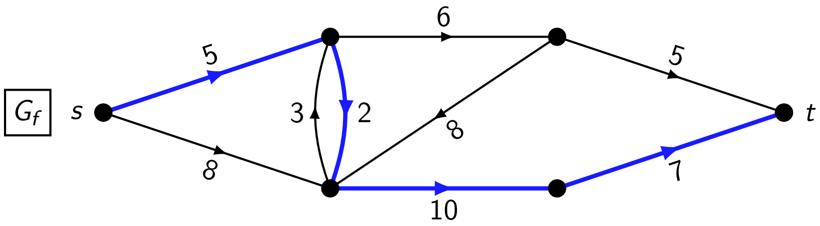




Push flow along the path. (This path has residual capacity 2.)

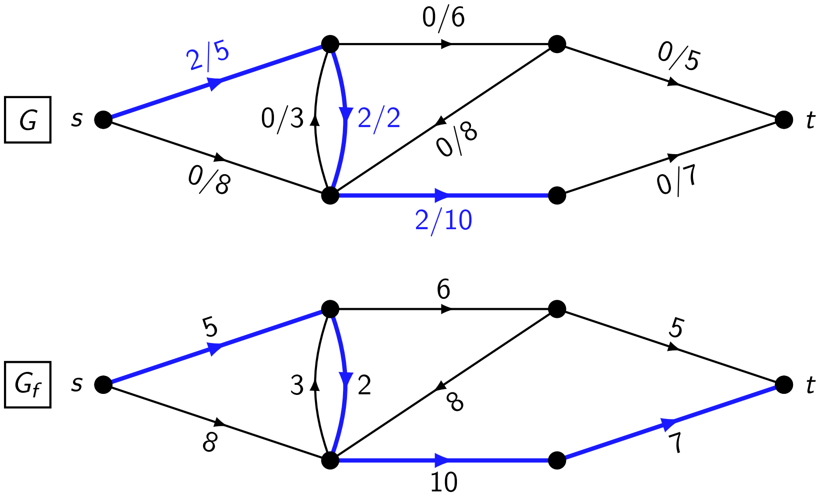
Worked example





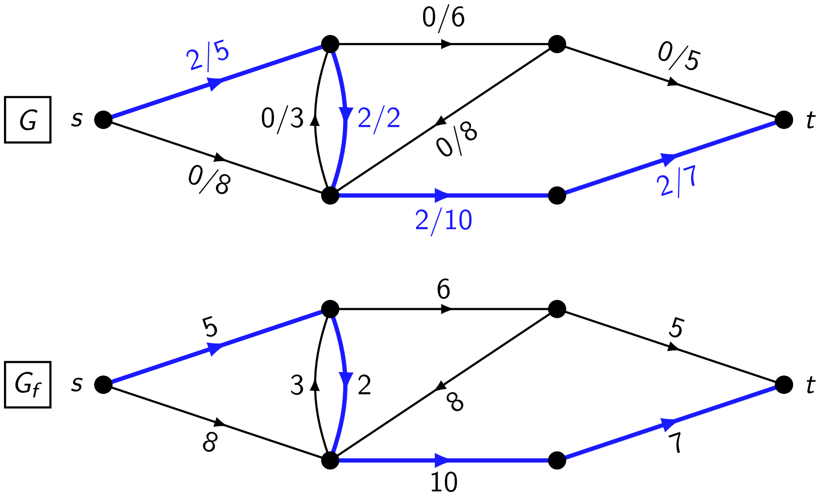
Push flow along the path. (This path has residual capacity 2.)

Worked example



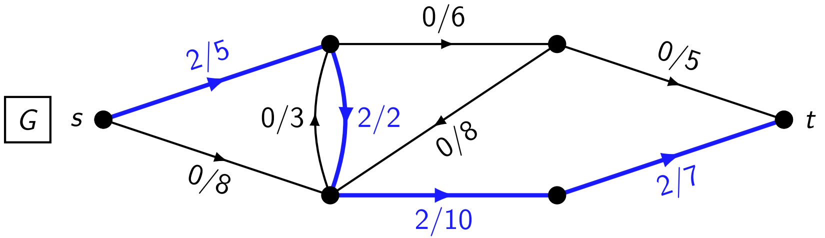
Push flow along the path. (This path has residual capacity 2.)

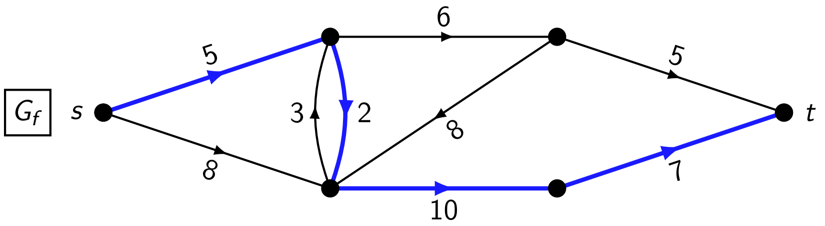
Worked example



Push flow along the path. (This path has residual capacity 2.)

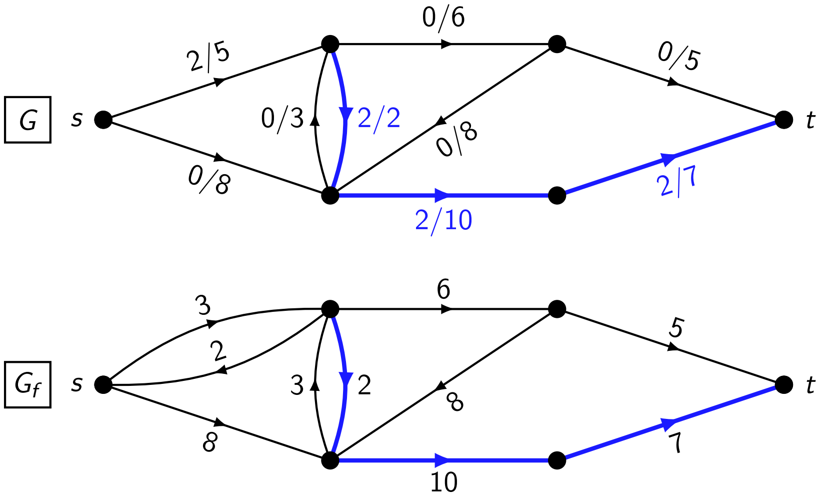
Worked example





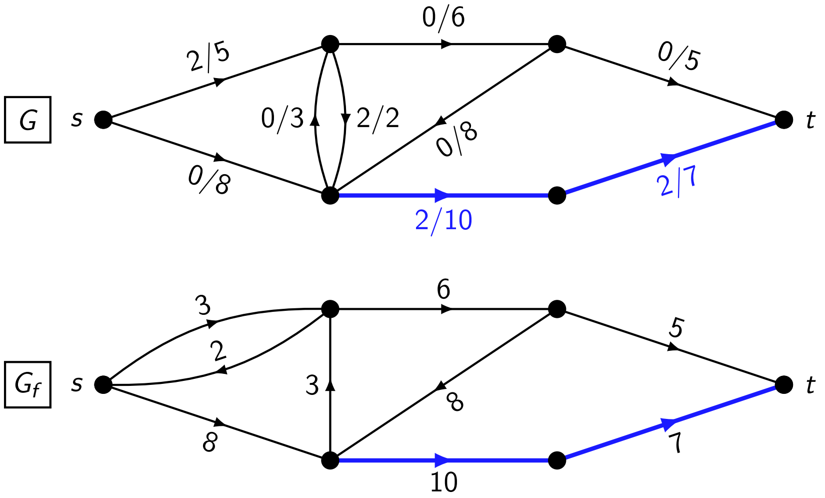
Update G_f along the path.

Worked example



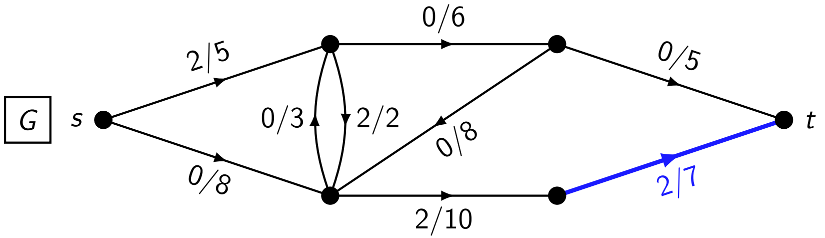
Update G_f along the path.

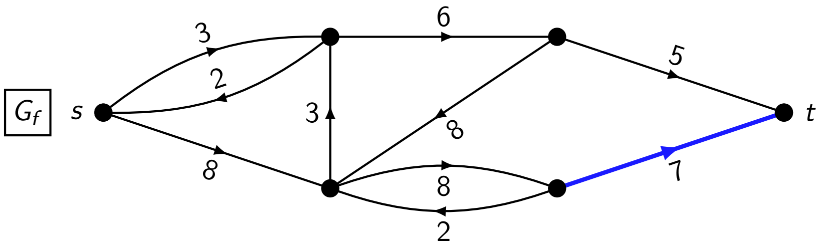
Worked example



Update G_f along the path.

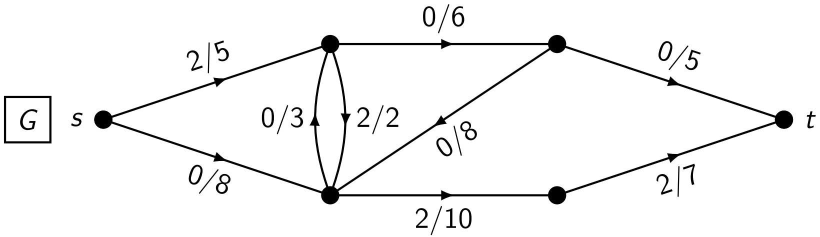
Worked example

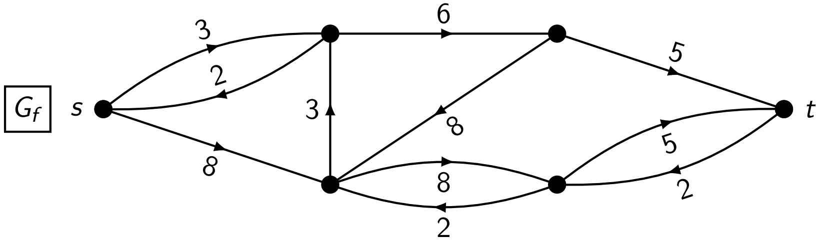




Update G_f along the path.

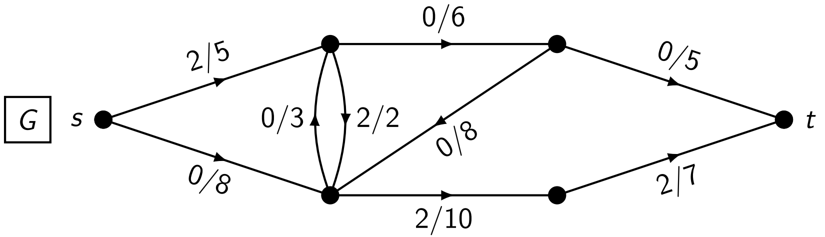
Worked example

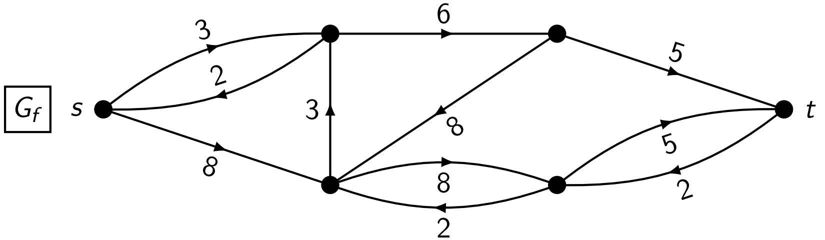




Update G_f along the path.

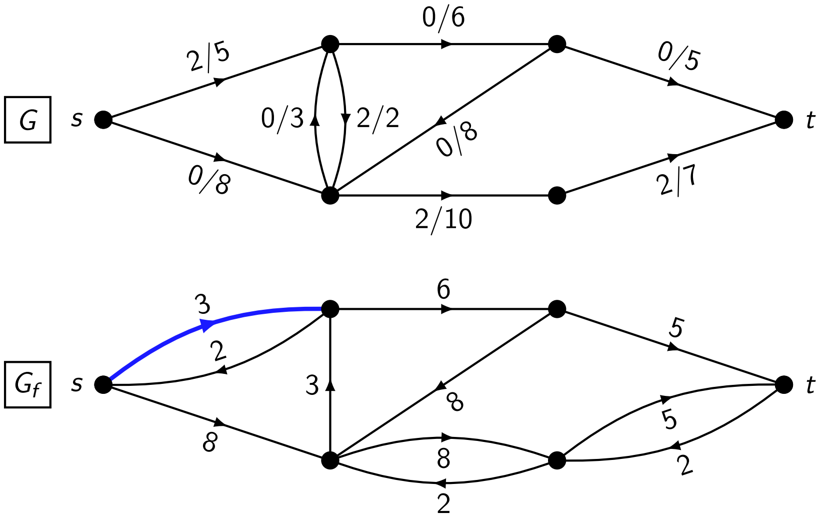
Worked example





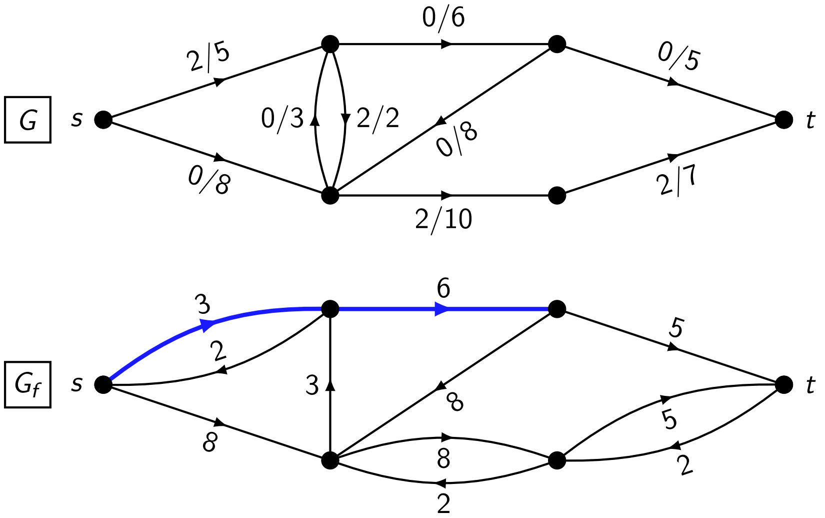
Apply depth-first search to find an augmenting path in G_f.

Worked example



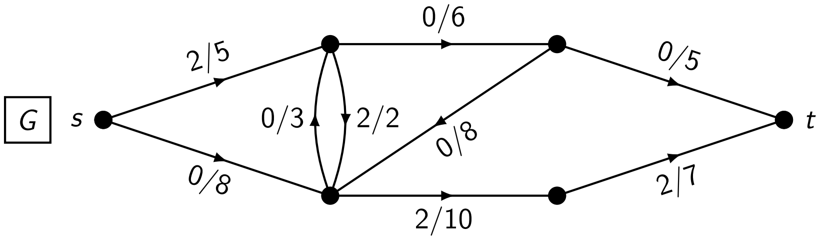
Apply depth-first search to find an augmenting path in G_f.

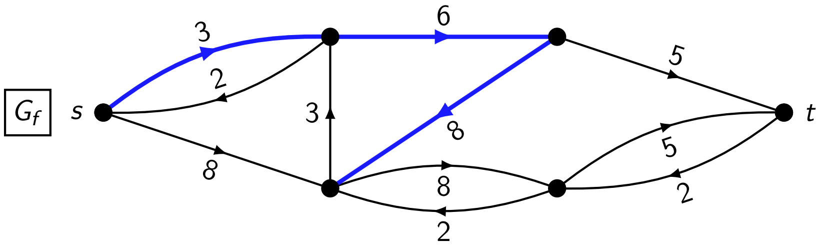
Worked example



Apply depth-first search to find an augmenting path in G_f.

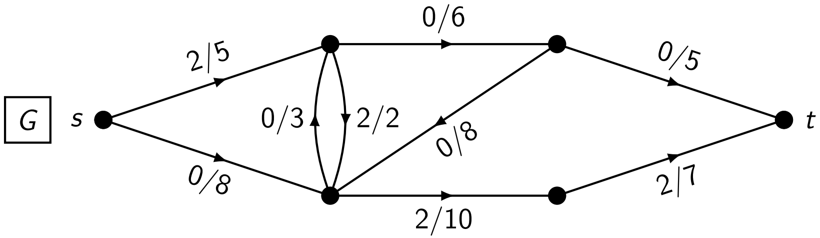
Worked example

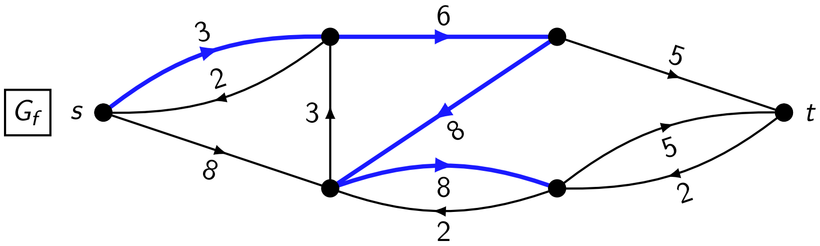




Apply depth-first search to find an augmenting path in G_f.

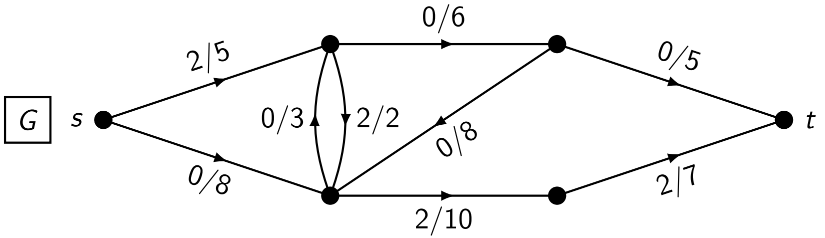
Worked example

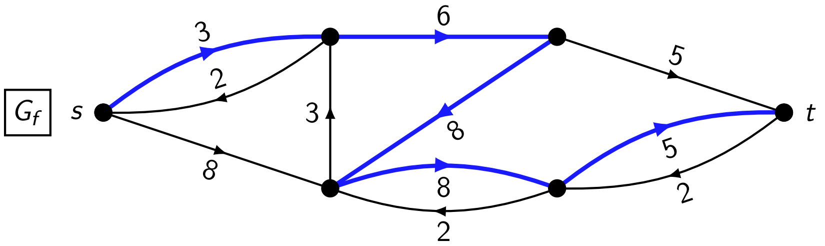




Apply depth-first search to find an augmenting path in G_f.

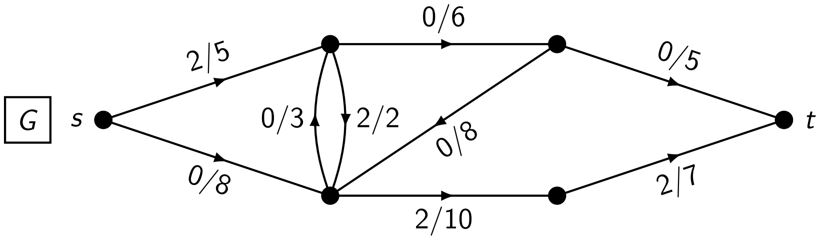
Worked example

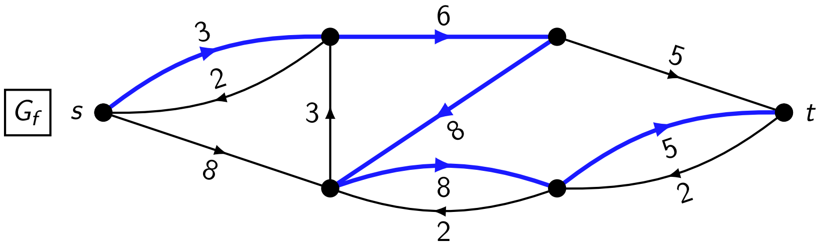




Apply depth-first search to find an augmenting path in G_f.

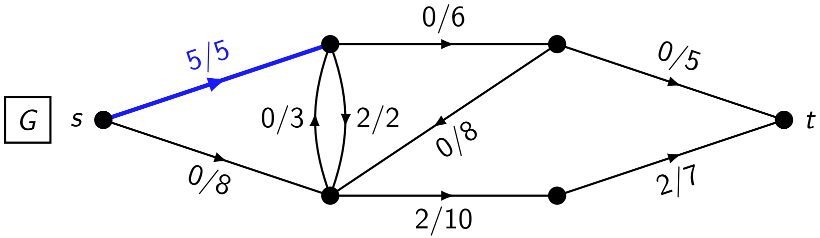
Worked example

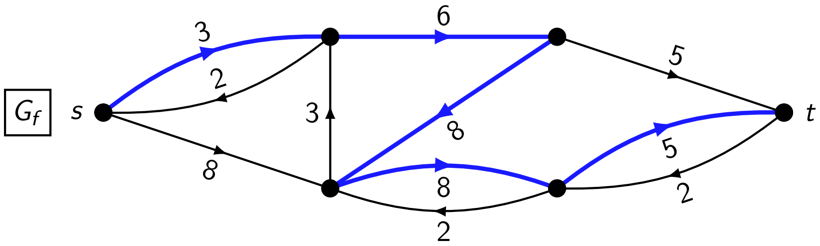




Push flow along the path. (This path has residual capacity 3.)

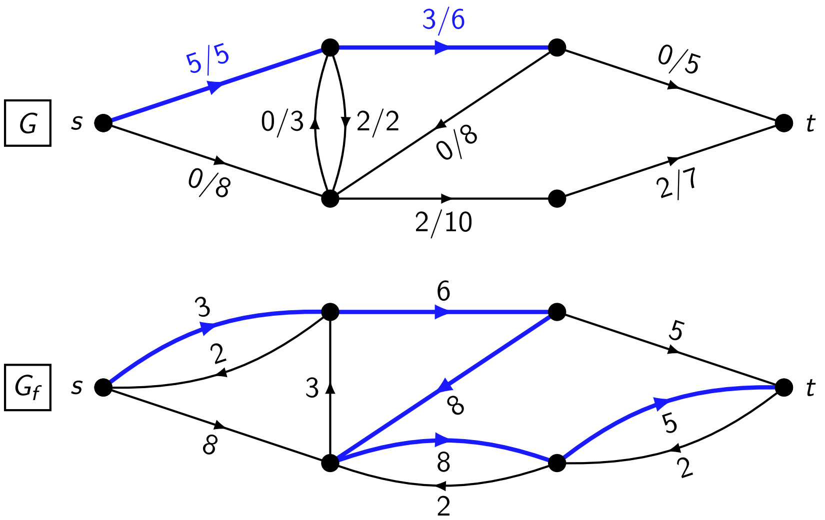
Worked example





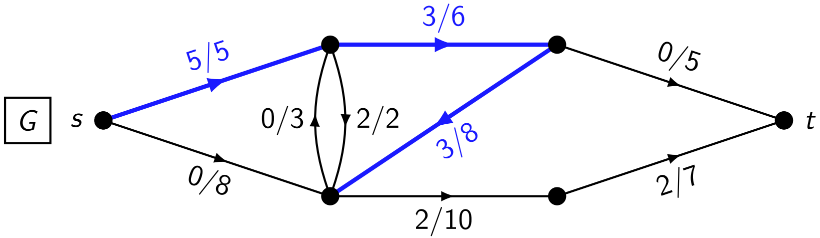
Push flow along the path. (This path has residual capacity 3.)

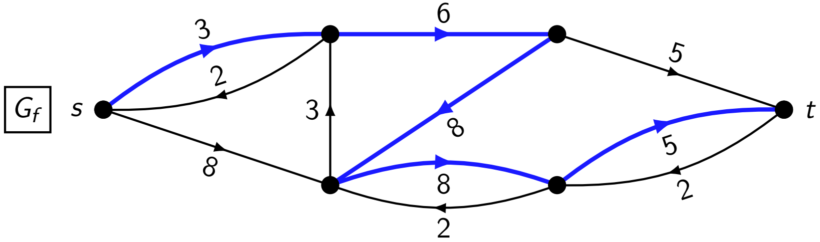
Worked example



Push flow along the path. (This path has residual capacity 3.)

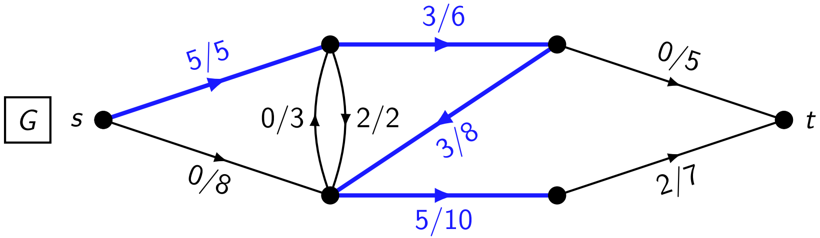
Worked example

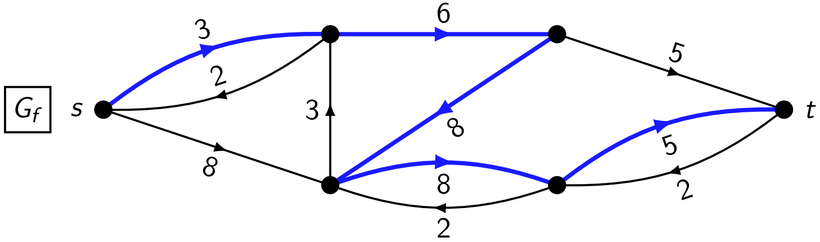




Push flow along the path. (This path has residual capacity 3.)

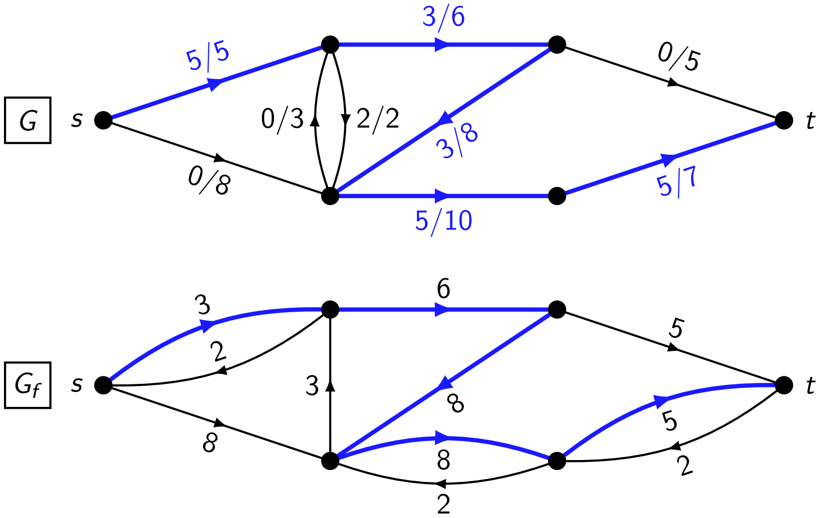
Worked example





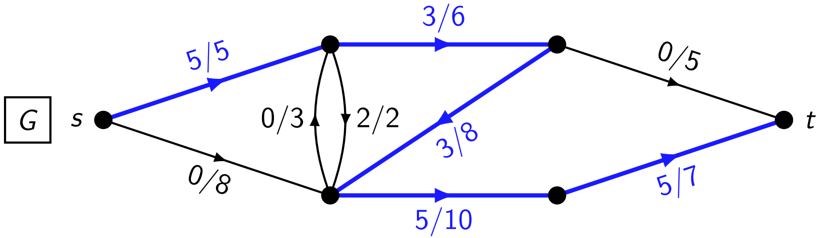
Push flow along the path. (This path has residual capacity 3.)

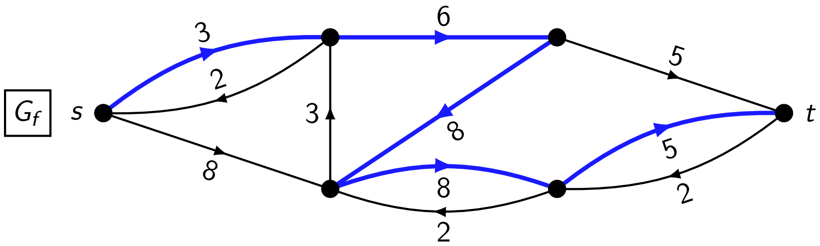
Worked example



Push flow along the path. (This path has residual capacity 3.)

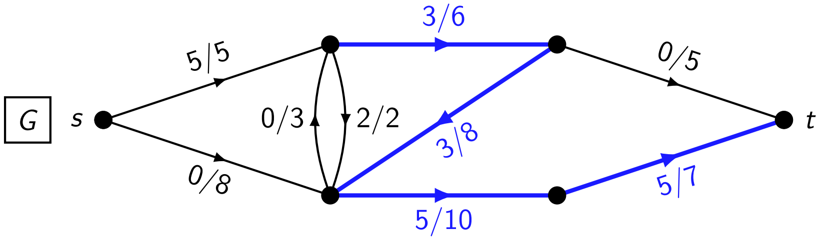
Worked example

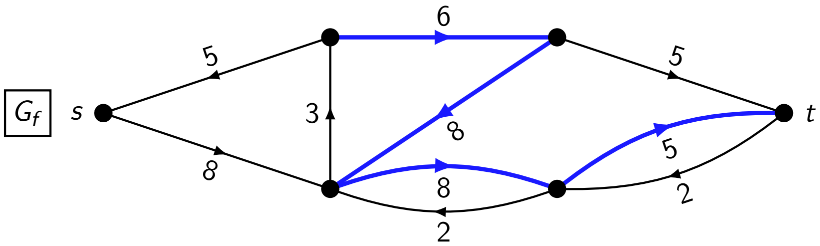




Update G_f along the path.

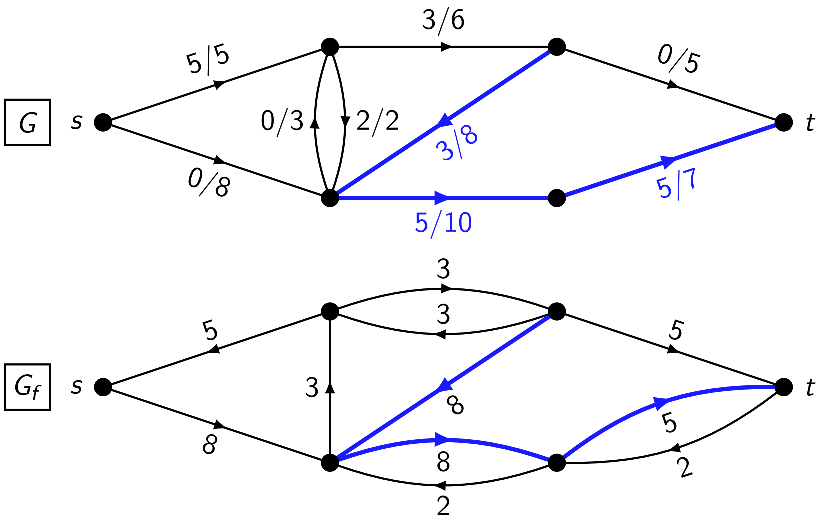
Worked example





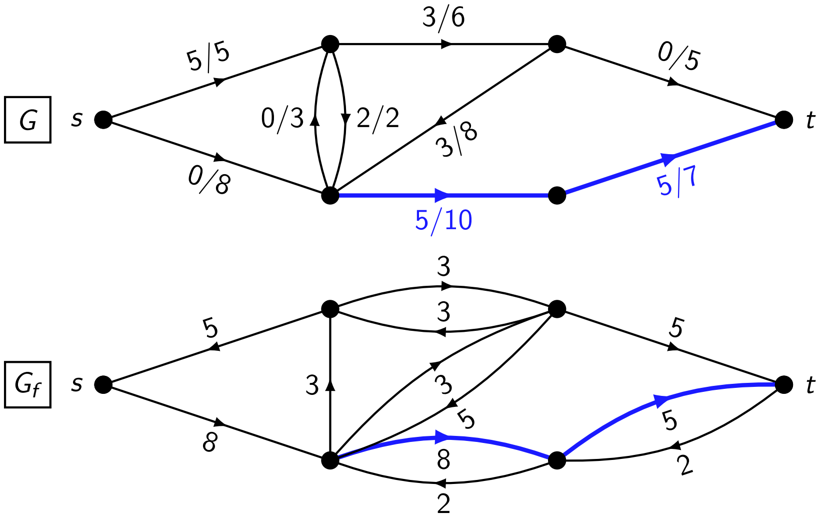
Update G_f along the path.

Worked example



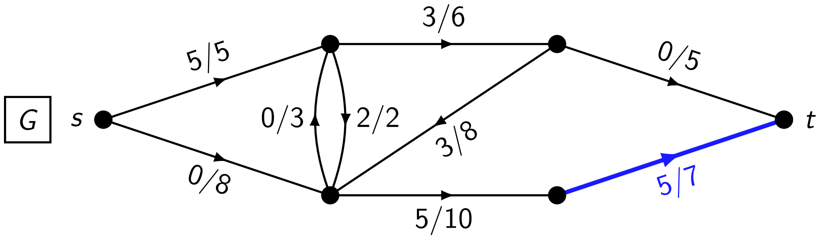
Update G_f along the path.

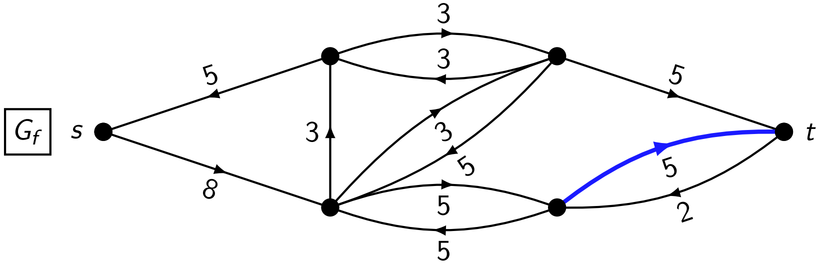
Worked example



Update G_f along the path.

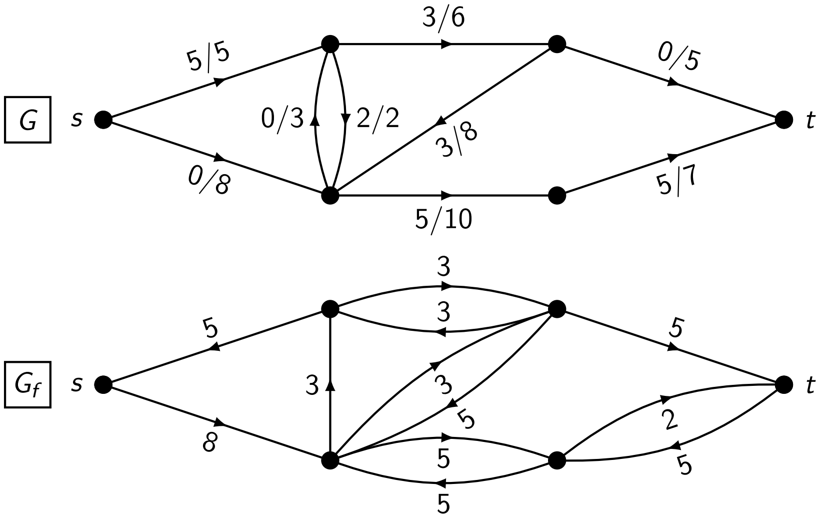
Worked example





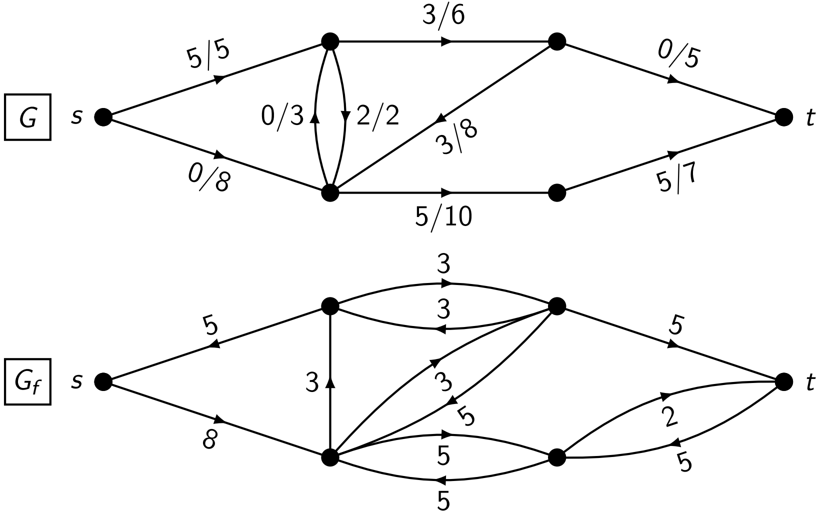
Update G_f along the path.

Worked example



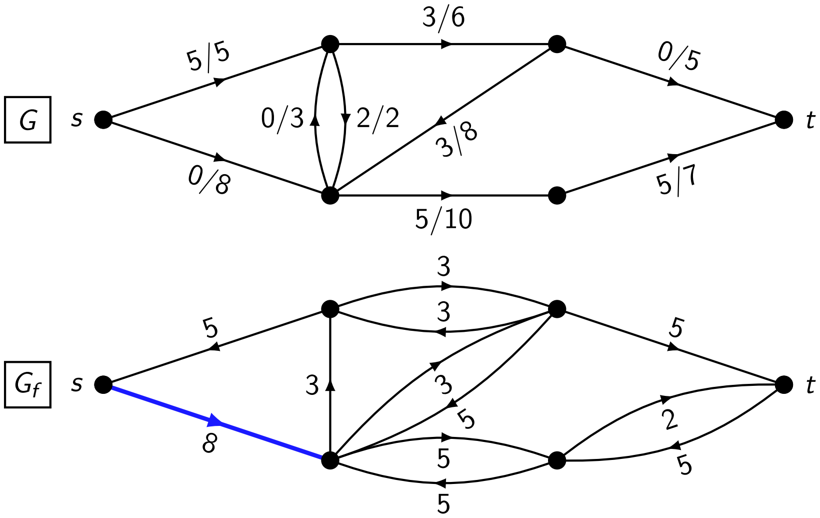
Update G_f along the path.

Worked example



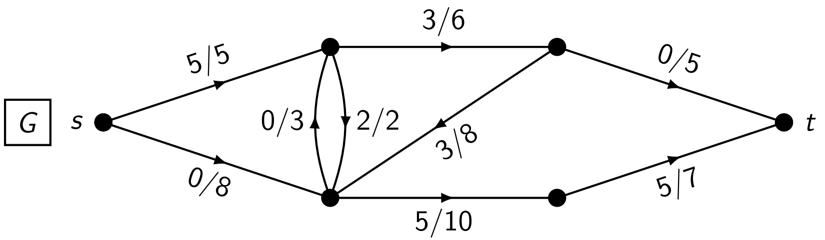
Apply depth-first search to find an augmenting path in G_f.

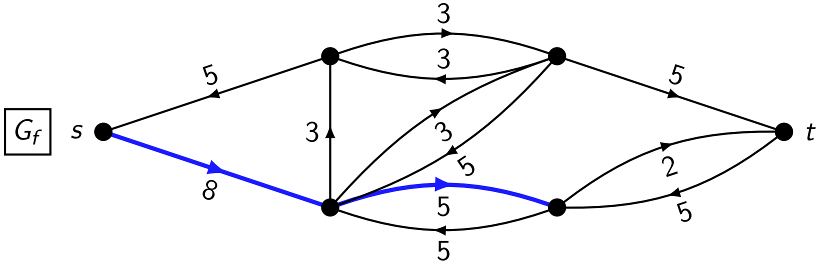
Worked example



Apply depth-first search to find an augmenting path in G_f.

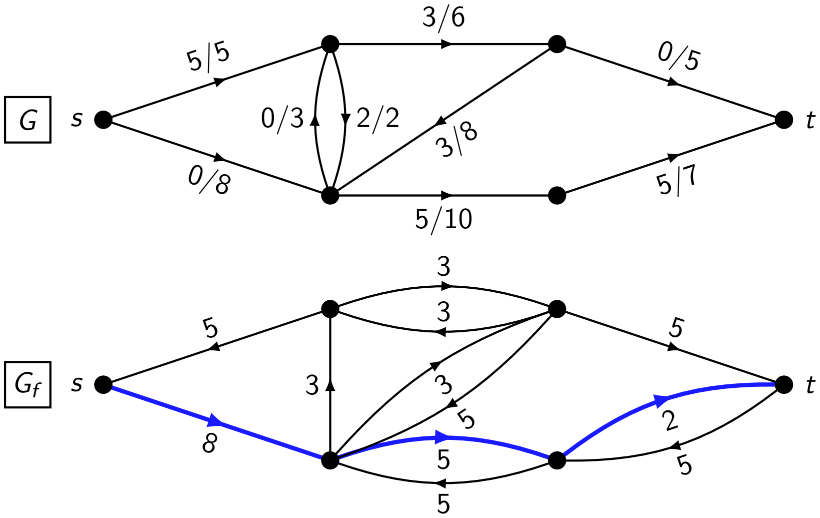
Worked example





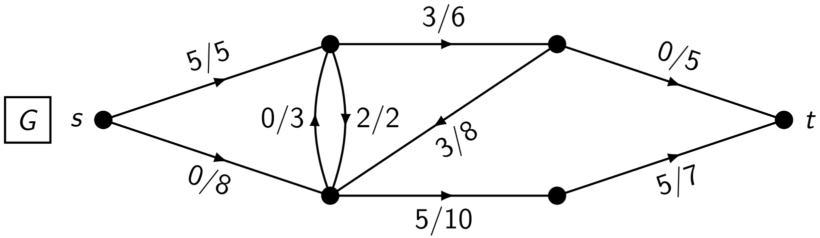
Apply depth-first search to find an augmenting path in G_f.

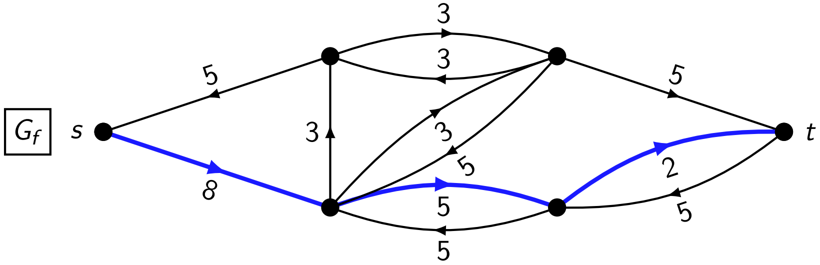
Worked example



Apply depth-first search to find an augmenting path in G_f.

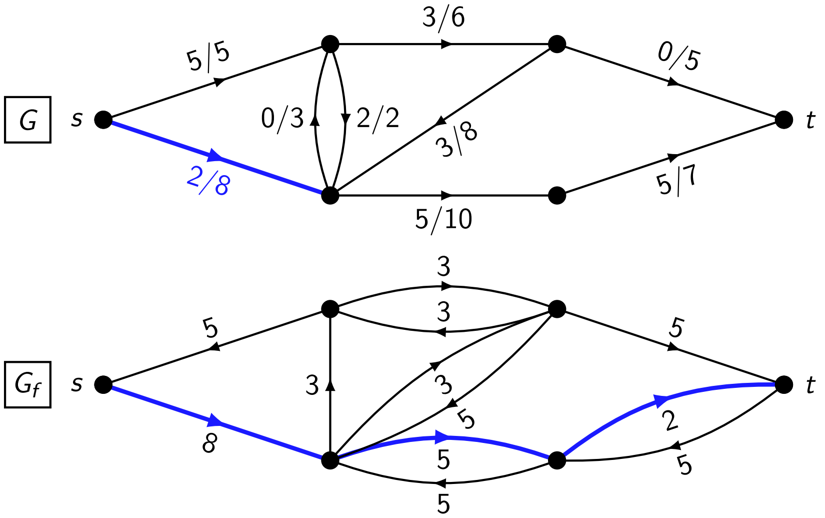
Worked example



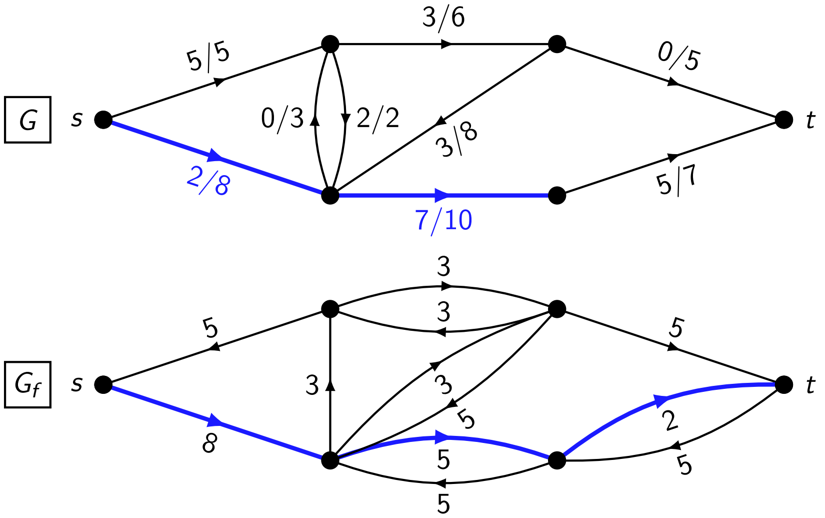


Push flow along the path. (This path has residual capacity 2.)

Worked example

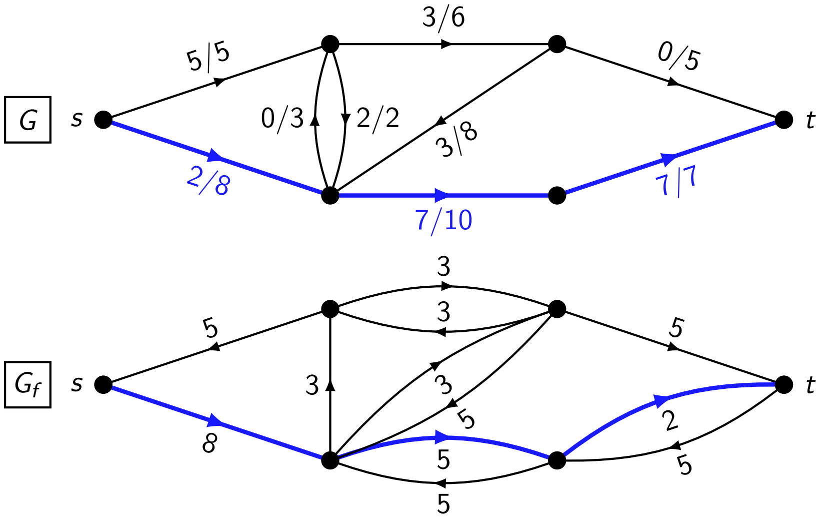


Push flow along the path. (This path has residual capacity 2.)



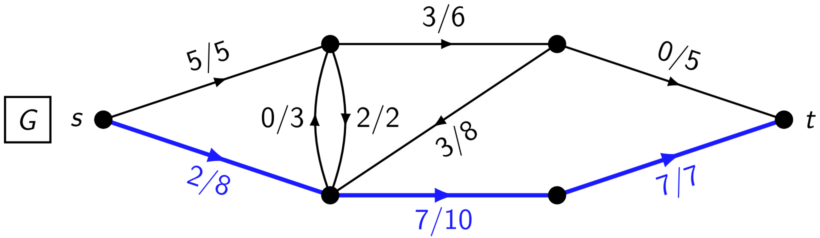
Push flow along the path. (This path has residual capacity 2.)

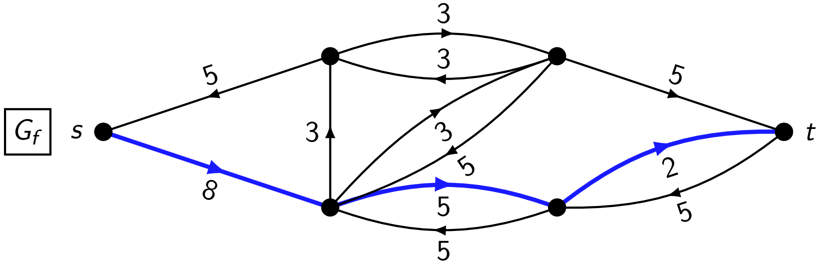
Worked example



Push flow along the path. (This path has residual capacity 2.)

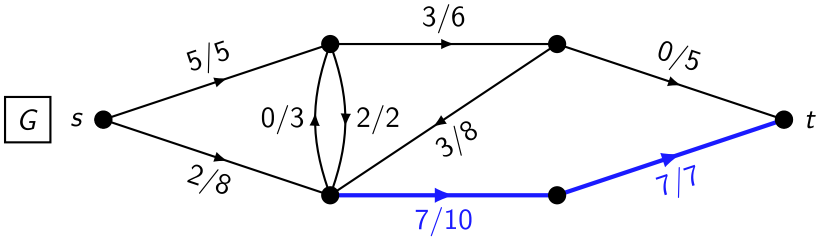
Worked example

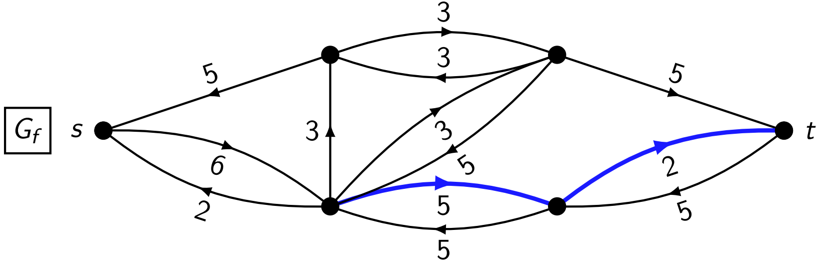




Update G_f along the path.

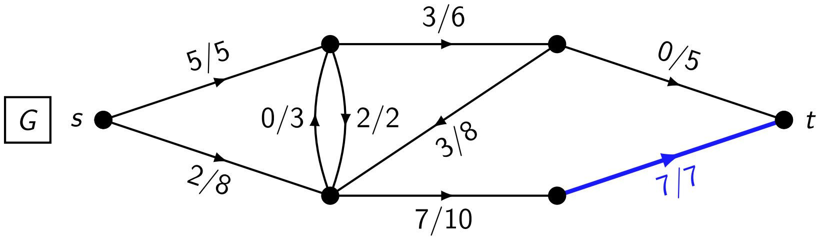
Worked example

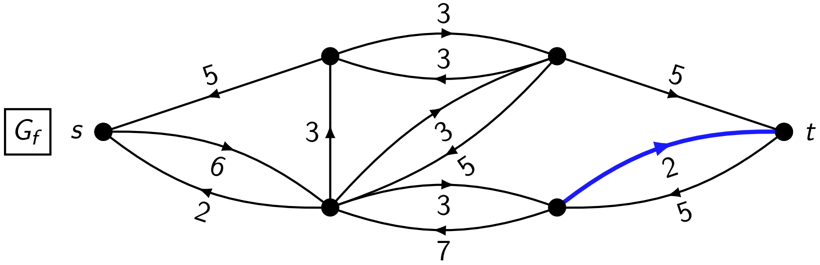




Update G_f along the path.

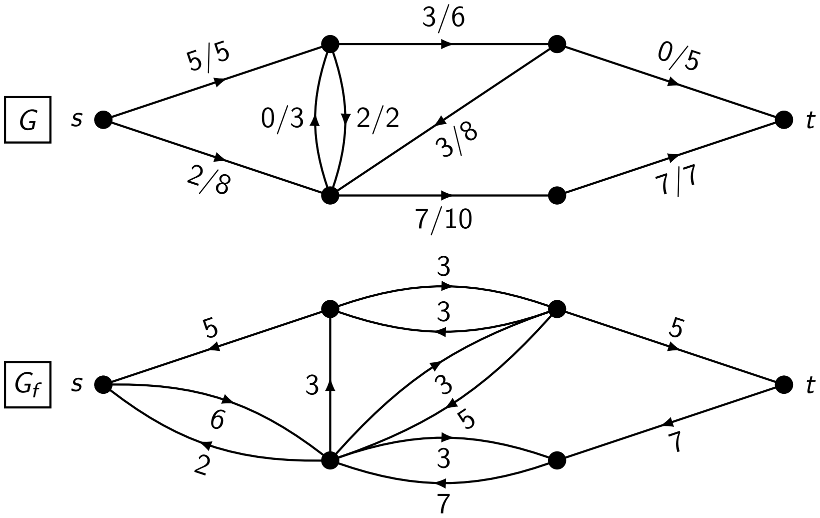
Worked example





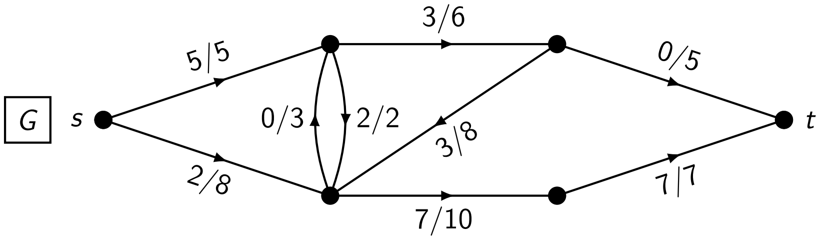
Update G_f along the path.

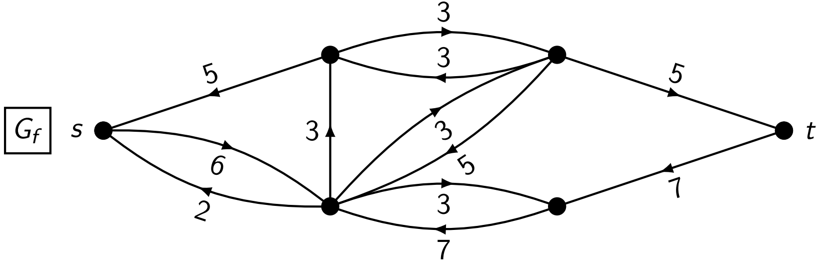
Worked example



Update G_f along the path.

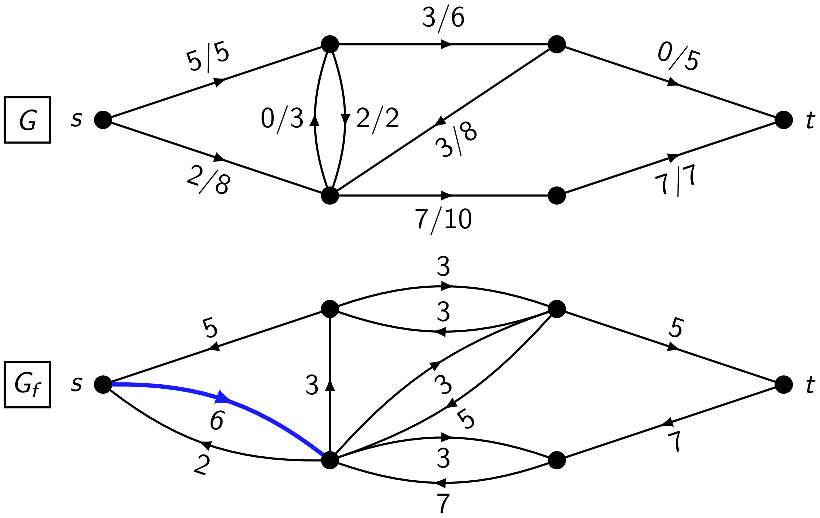
Worked example





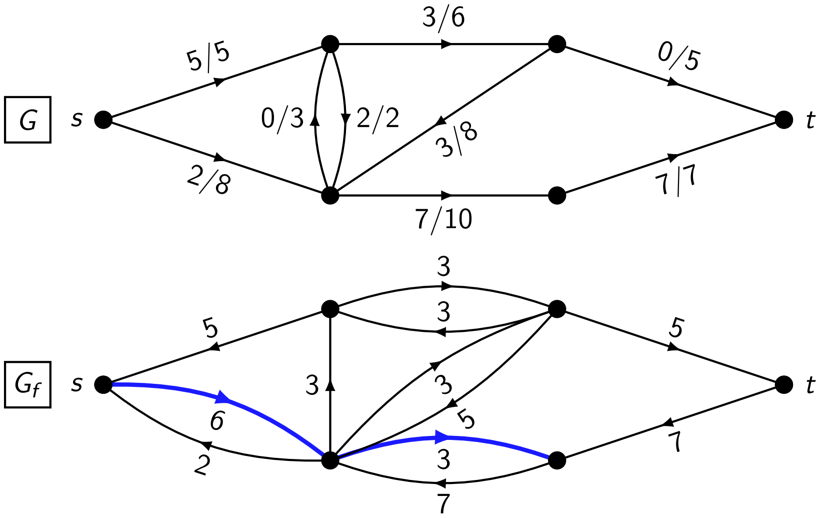
Apply depth-first search to find an augmenting path in G_f.

Worked example



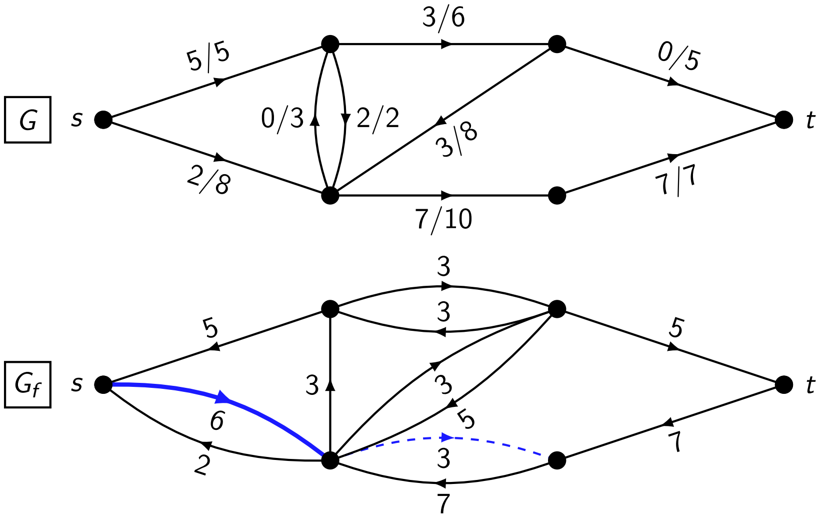
Apply depth-first search to find an augmenting path in G_f.

Worked example



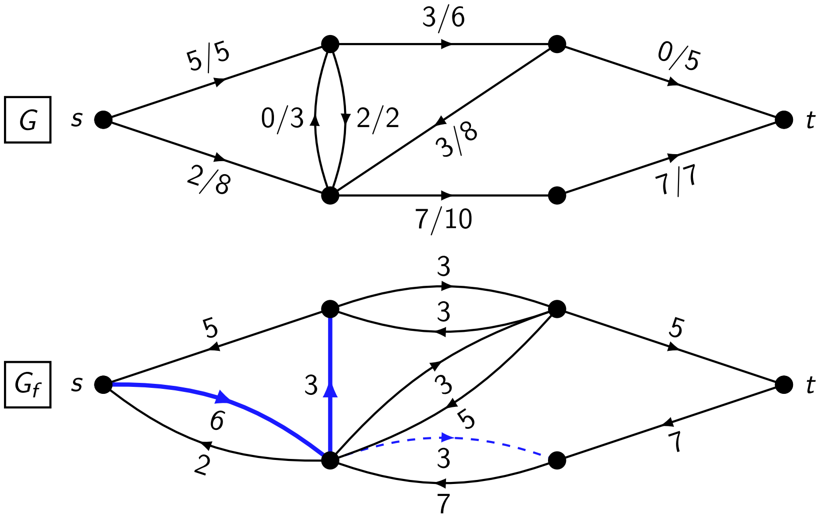
Apply depth-first search to find an augmenting path in G_f.

Worked example



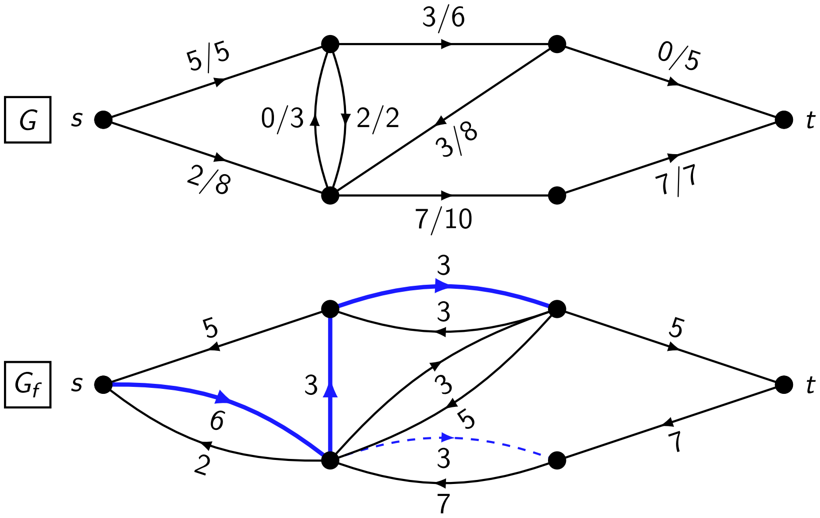
Apply depth-first search to find an augmenting path in G_f.

Worked example



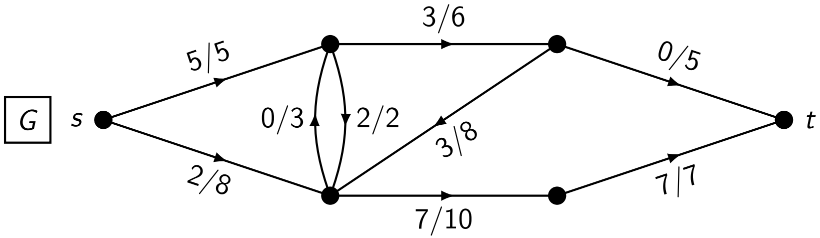
Apply depth-first search to find an augmenting path in G_f.

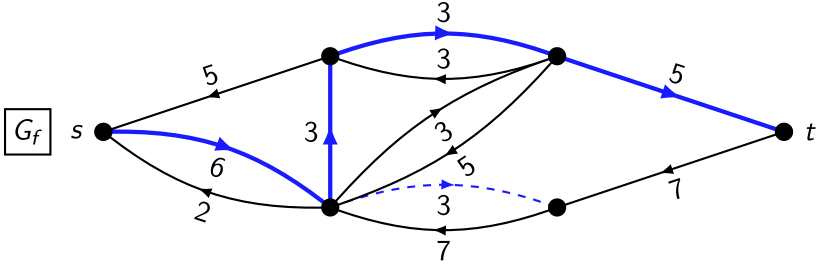
Worked example



Apply depth-first search to find an augmenting path in G_f.

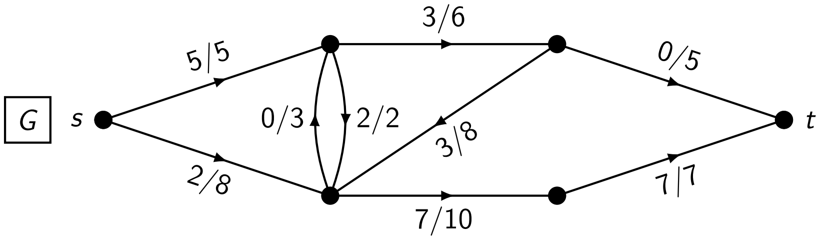
Worked example

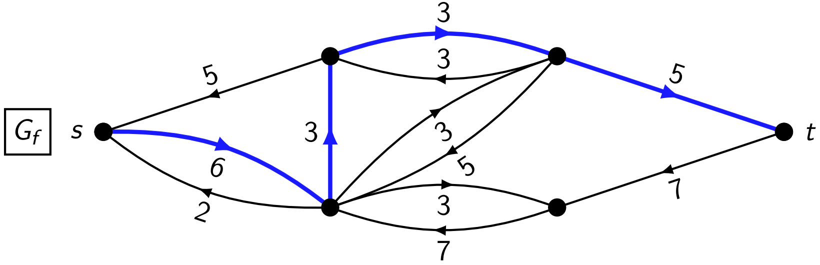




Apply depth-first search to find an augmenting path in G_f.

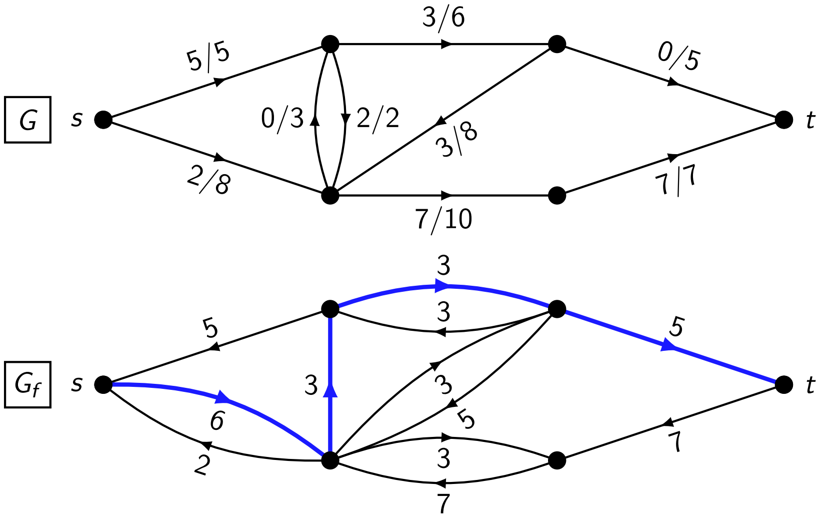
Worked example





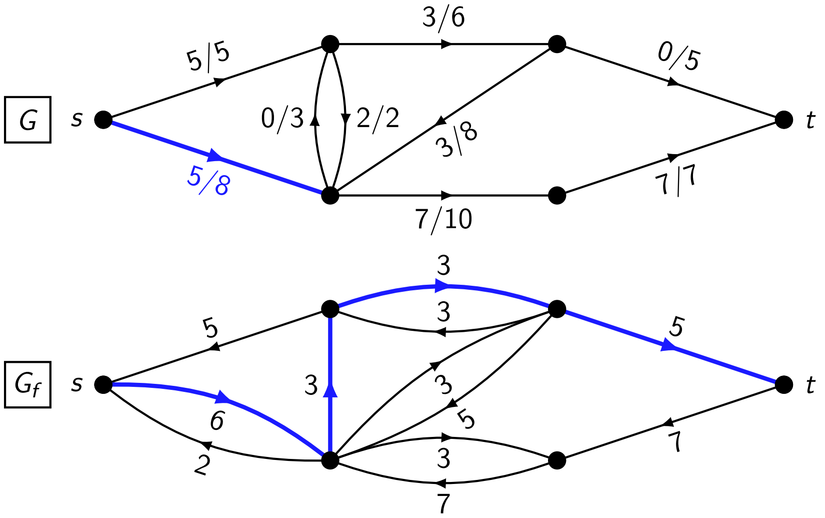
Apply depth-first search to find an augmenting path in G_f.

Worked example



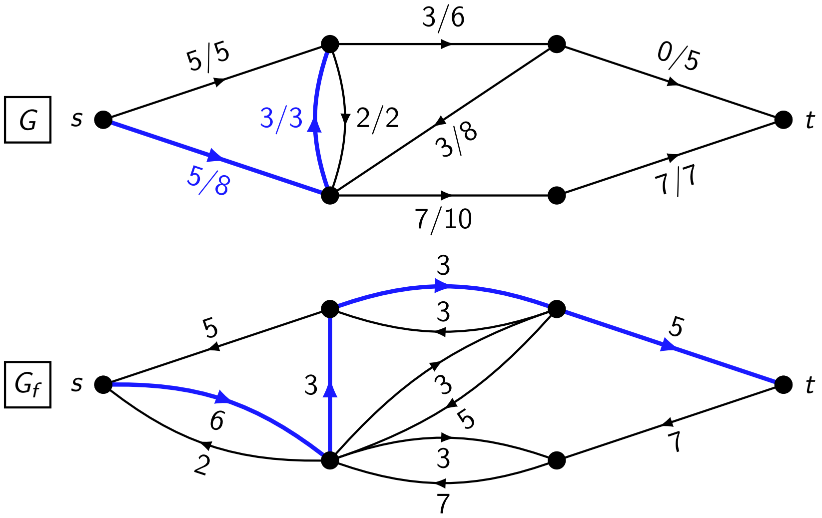
Push flow along the path. (This path has residual capacity 3.)

Worked example



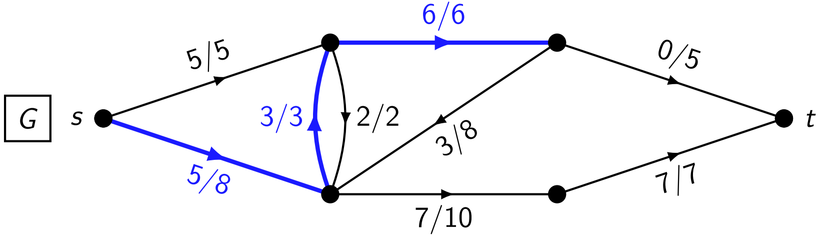
Push flow along the path. (This path has residual capacity 3.)

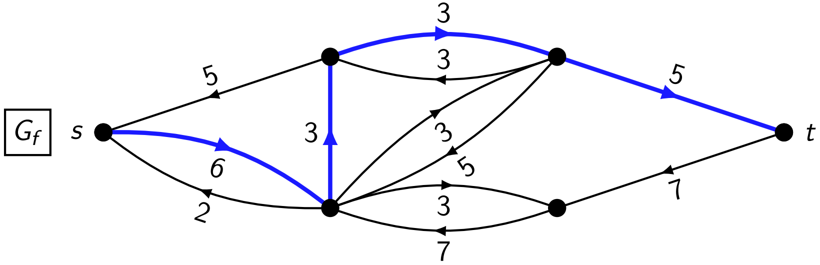
Worked example



Push flow along the path. (This path has residual capacity 3.)

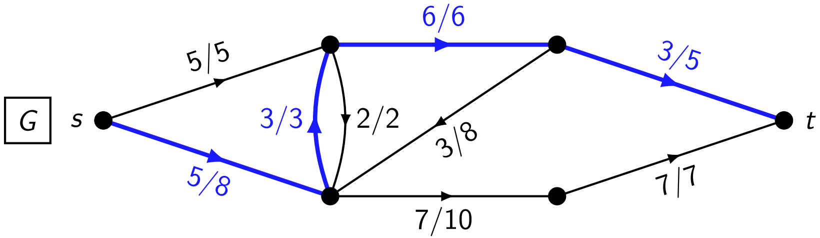
Worked example

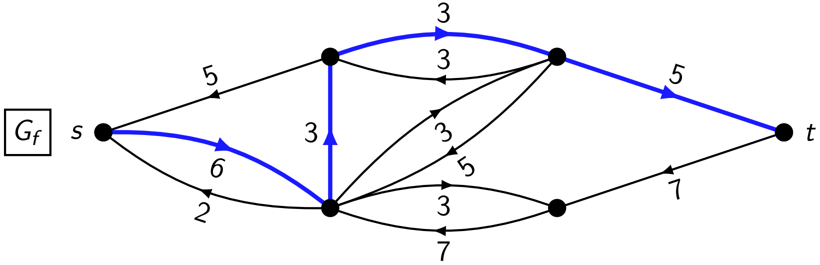




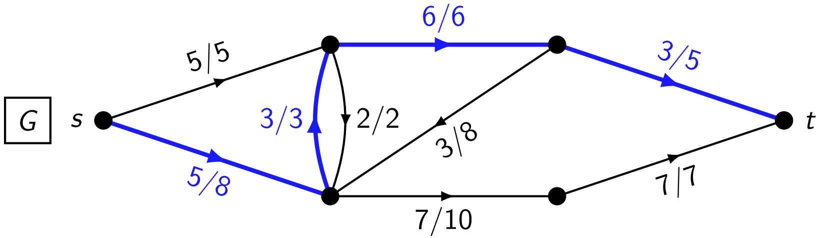
Push flow along the path. (This path has residual capacity 3.)

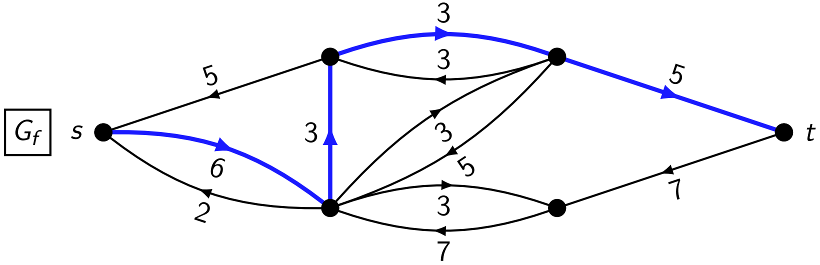
Worked example





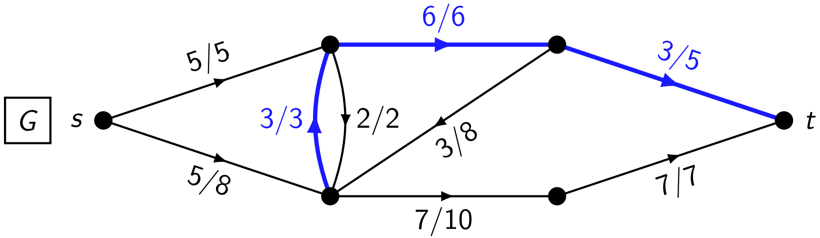
Push flow along the path. (This path has residual capacity 3.)

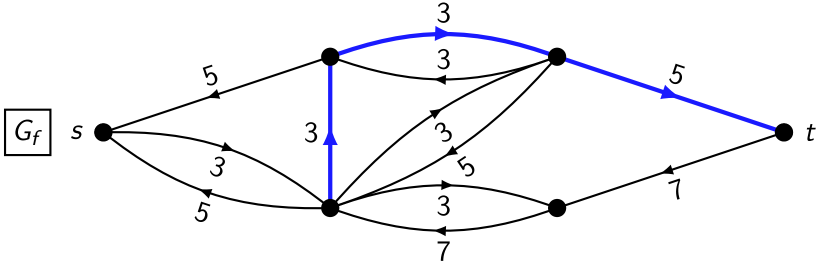




Update G_f along the path.

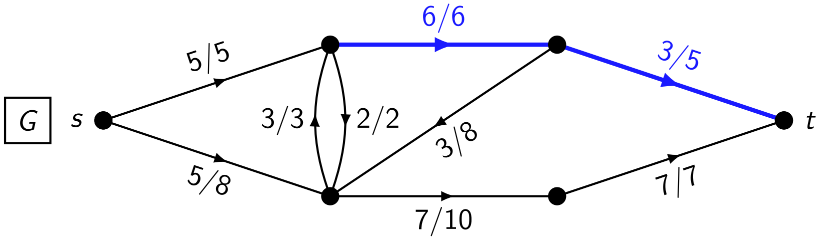
Worked example

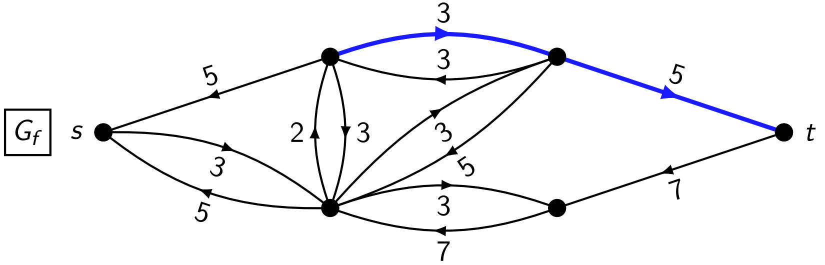




Update G_f along the path.

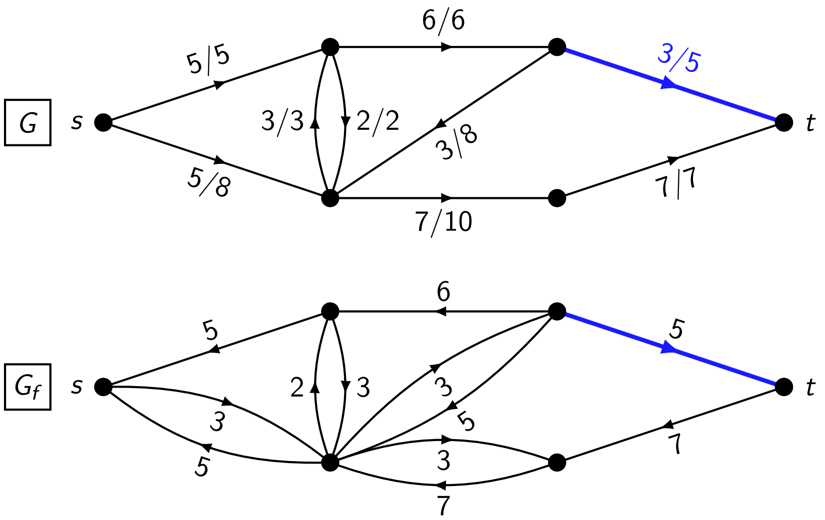
Worked example



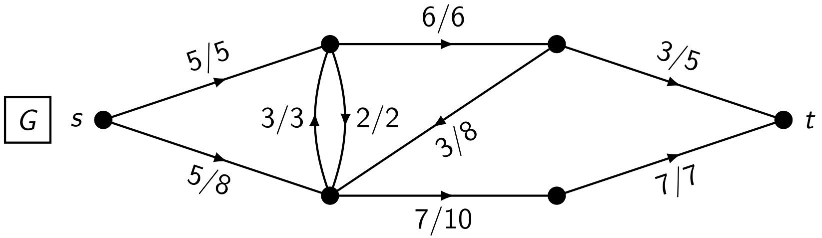


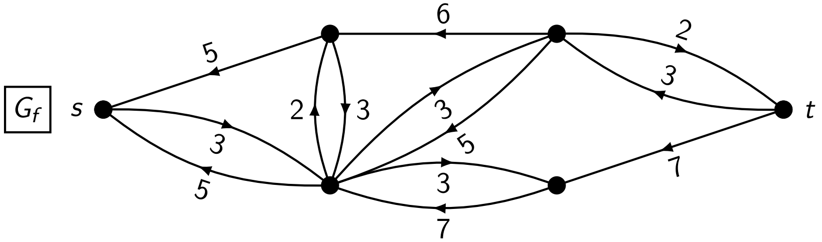
Update G_f along the path.

Worked example



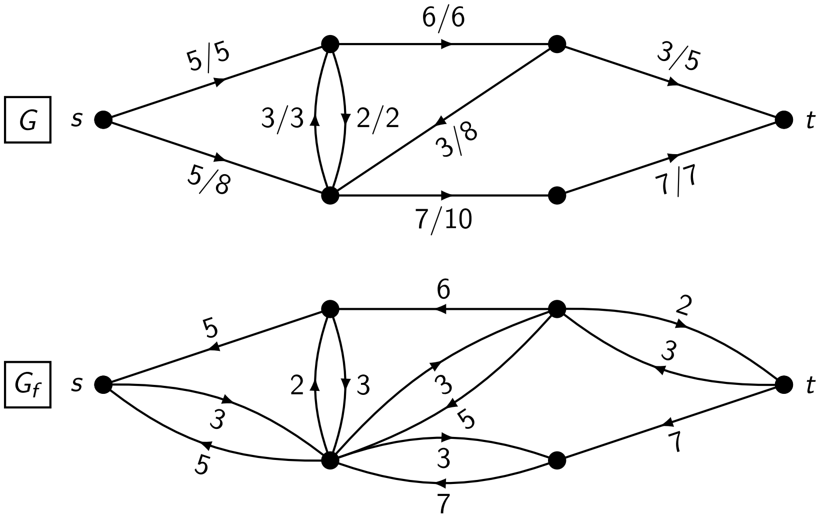
Update G_f along the path.





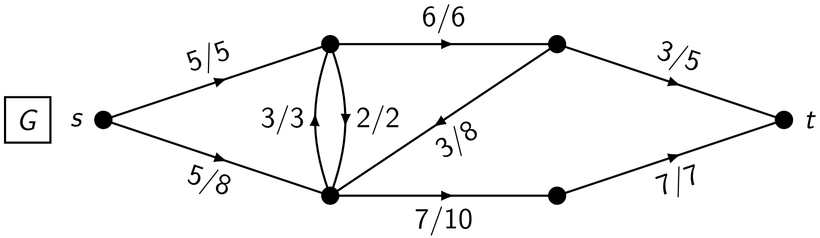
Update G_f along the path.

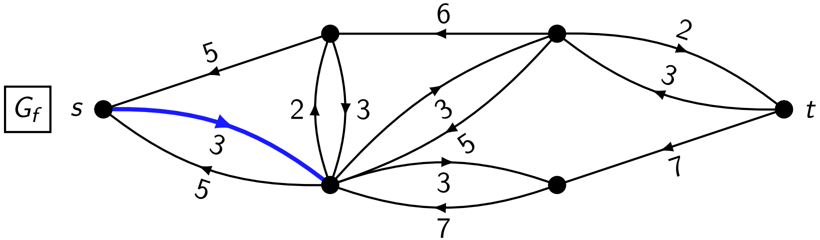
Worked example



Apply depth-first search to find an augmenting path in G_f.

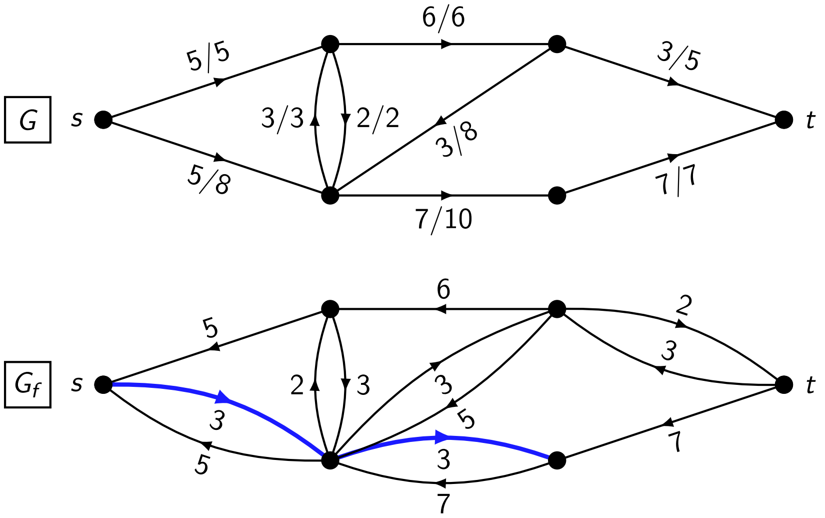
Worked example





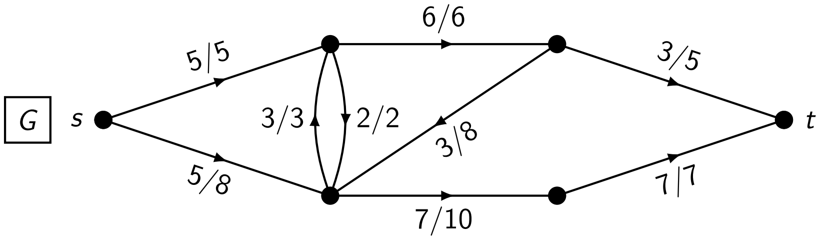
Apply depth-first search to find an augmenting path in G_f.

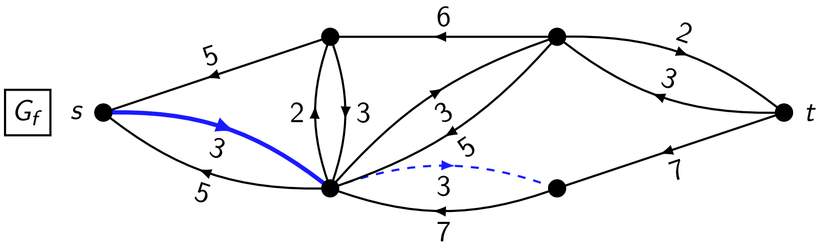
Worked example



Apply depth-first search to find an augmenting path in G_f.

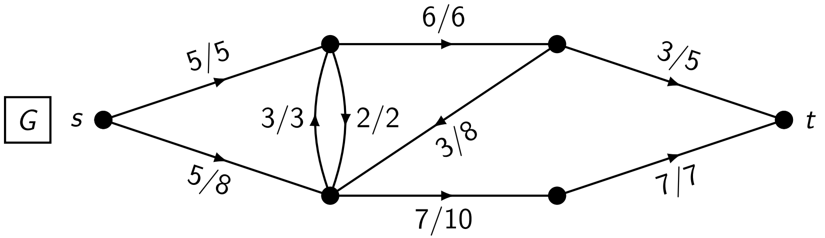
Worked example

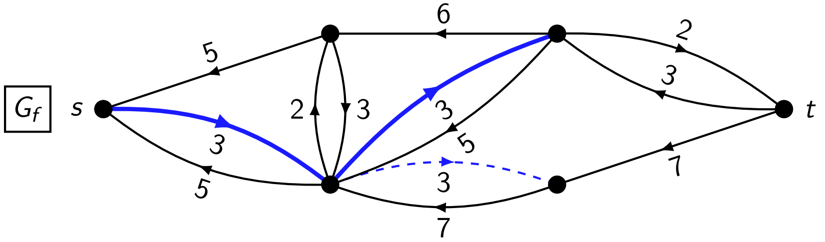




Apply depth-first search to find an augmenting path in G_f.

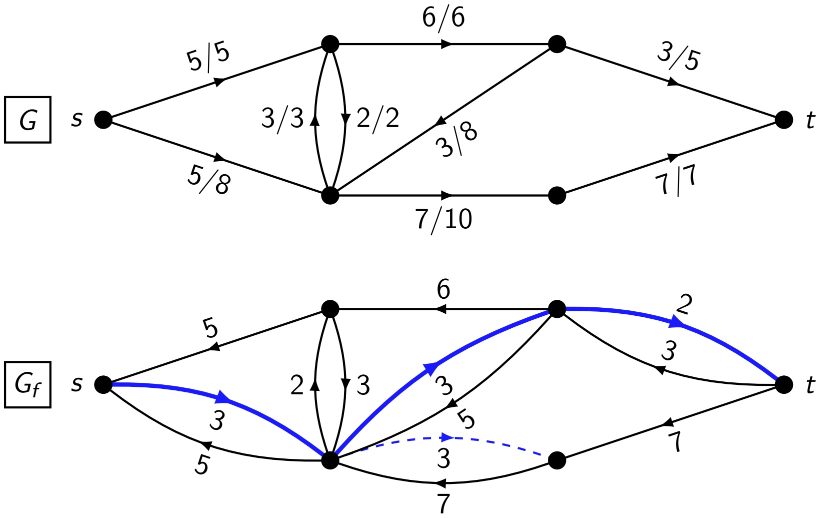
Worked example





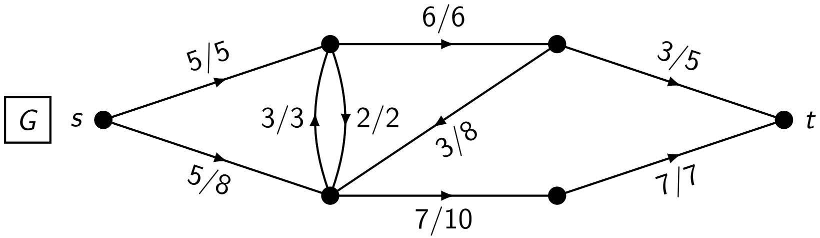
Apply depth-first search to find an augmenting path in G_f.

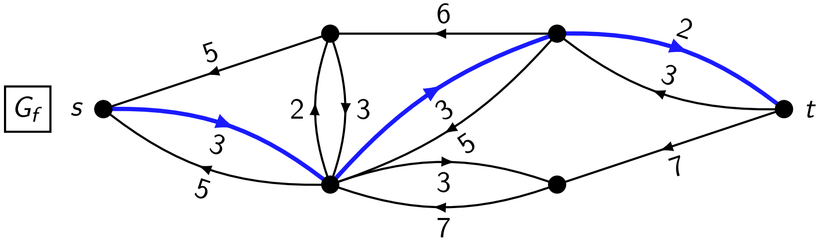
Worked example



Apply depth-first search to find an augmenting path in G_f.

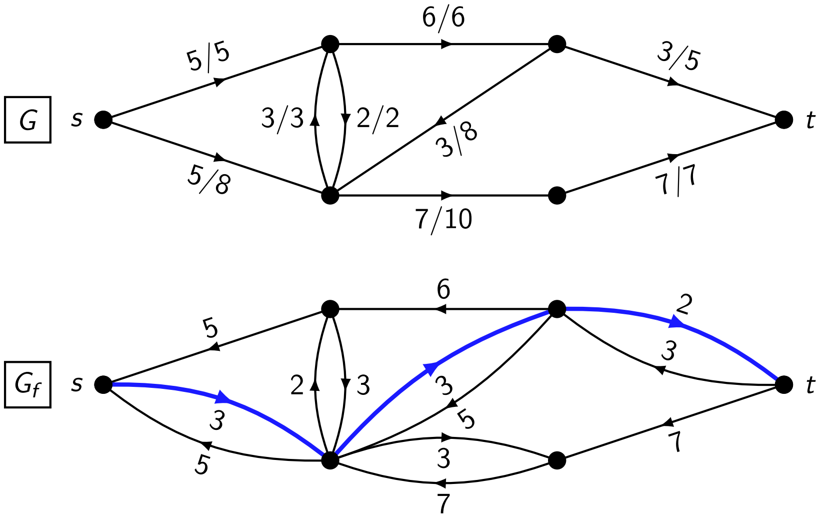
Worked example





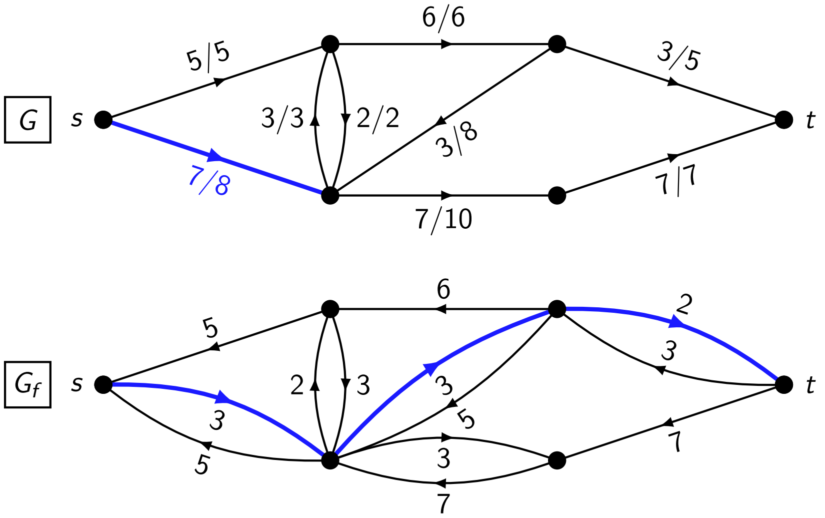
Apply depth-first search to find an augmenting path in G_f.

Worked example



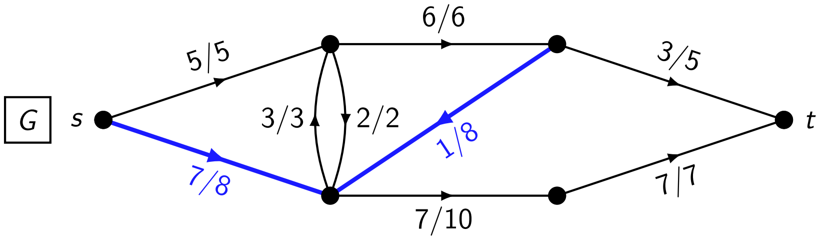
Push flow along the path. (This path has residual capacity 2.)

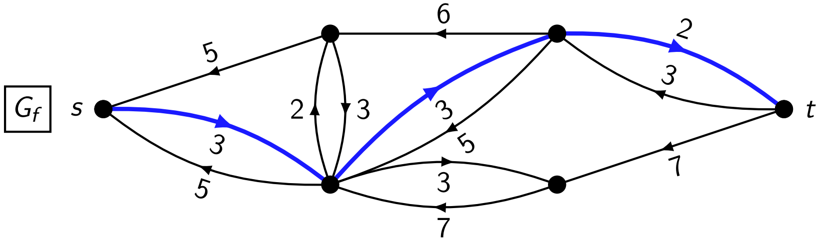
Worked example



Push flow along the path. (This path has residual capacity 2.)

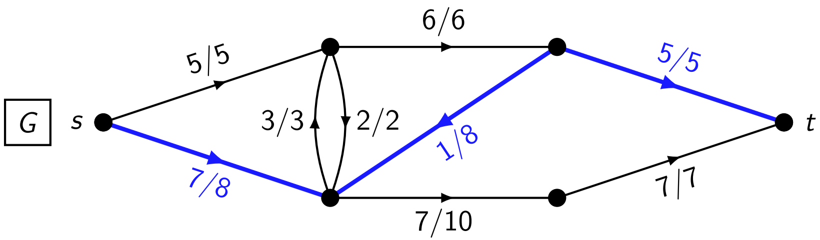
Worked example

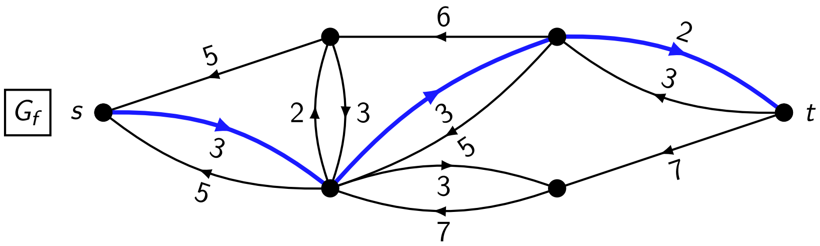




Push flow along the path. (This path has residual capacity 2.)

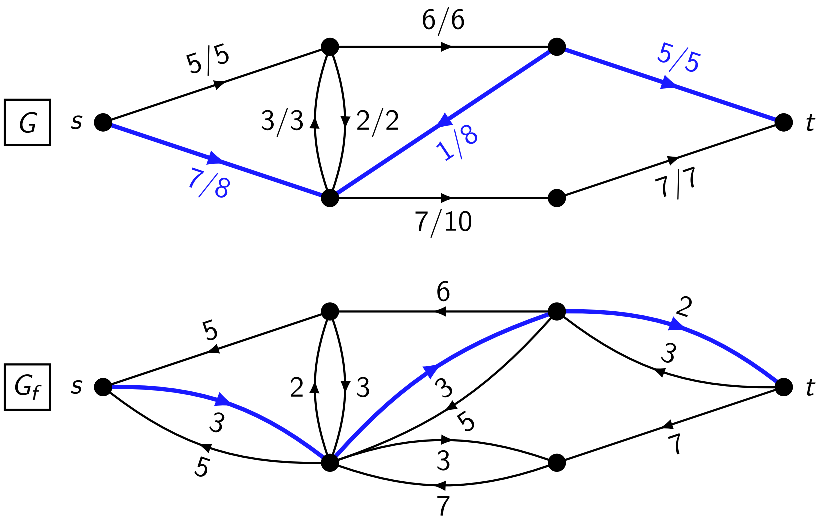
Worked example





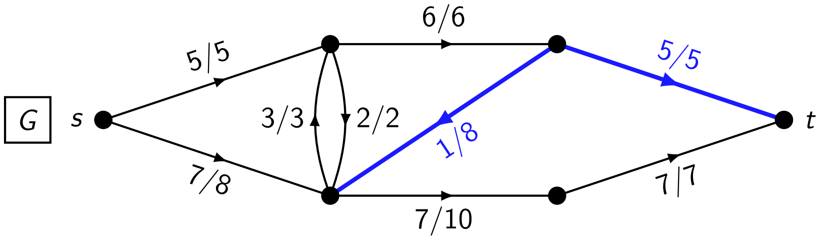
Push flow along the path. (This path has residual capacity 2.)

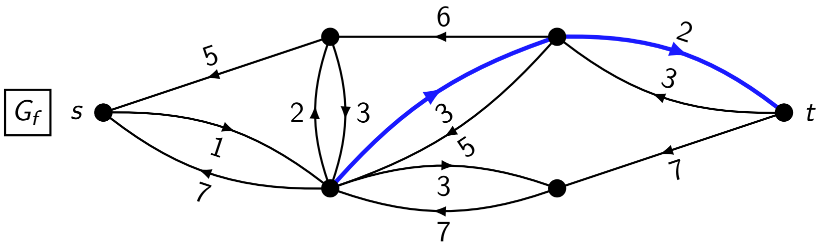
Worked example



Update G_f along the path.

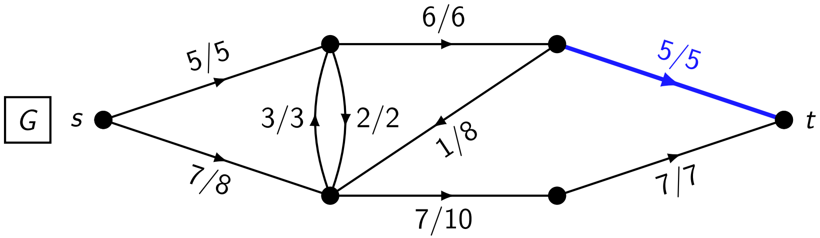
Worked example

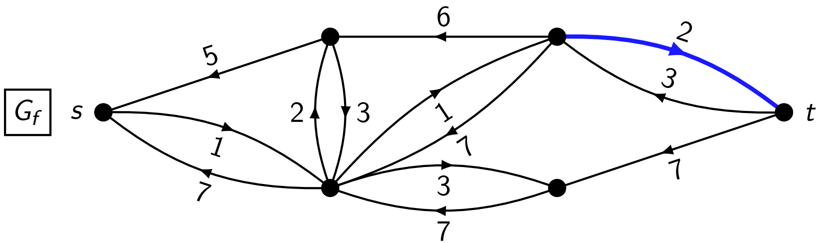




Update G_f along the path.

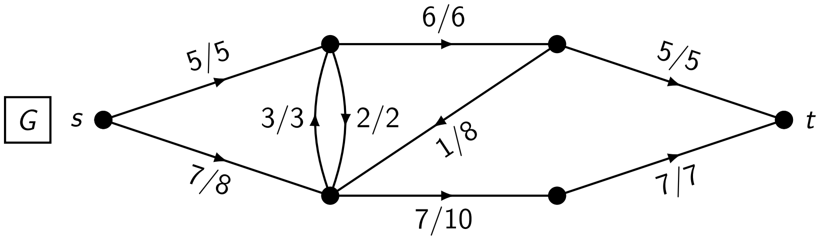
Worked example

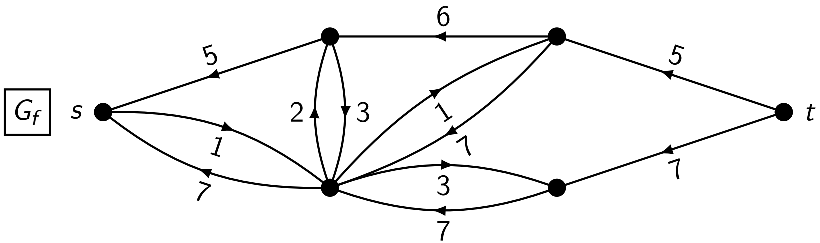




Update G_f along the path.

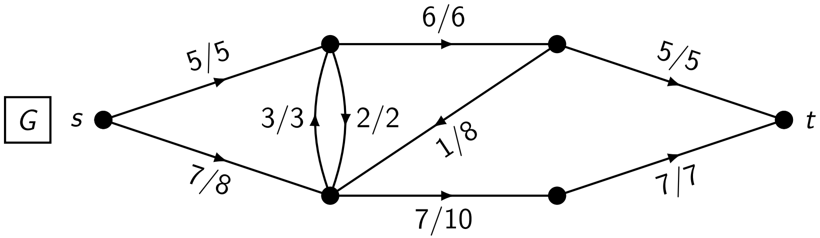
Worked example

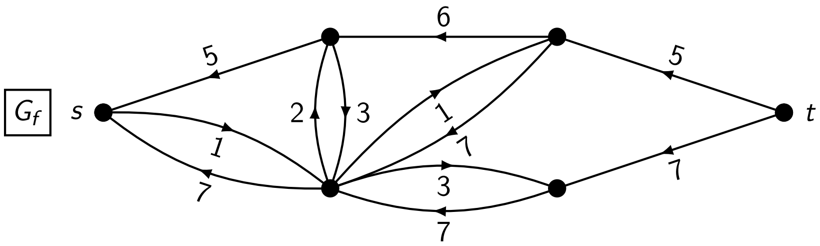




Update G_f along the path.

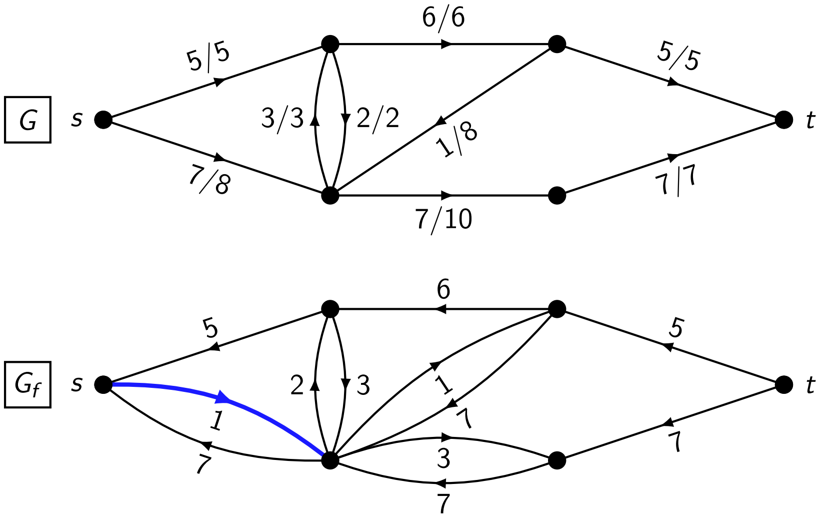
Worked example





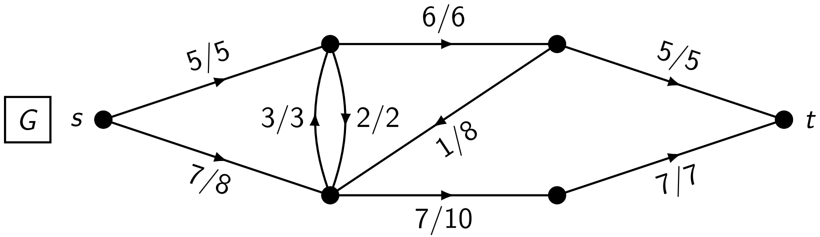
Apply depth-first search to find an augmenting path in G_f.

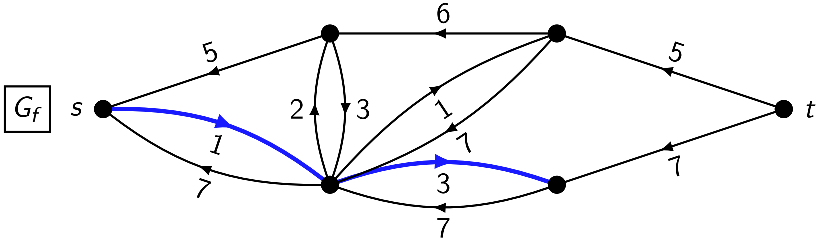
Worked example



Apply depth-first search to find an augmenting path in G_f.

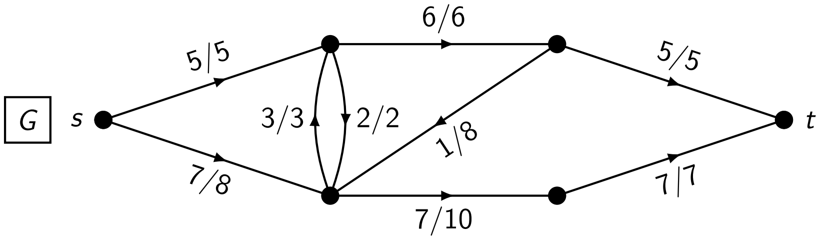
Worked example

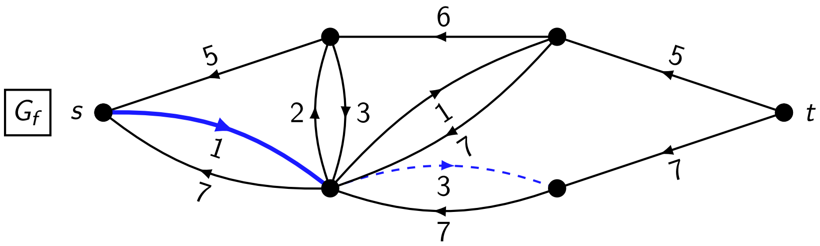




Apply depth-first search to find an augmenting path in G_f.

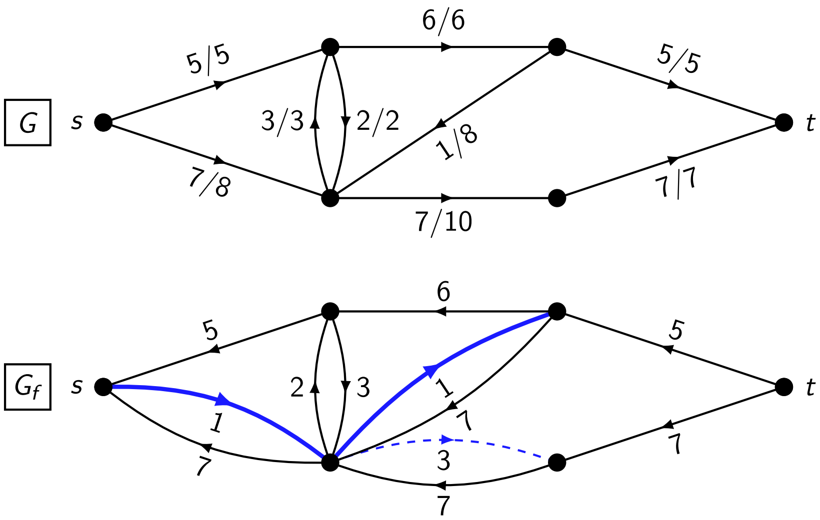
Worked example





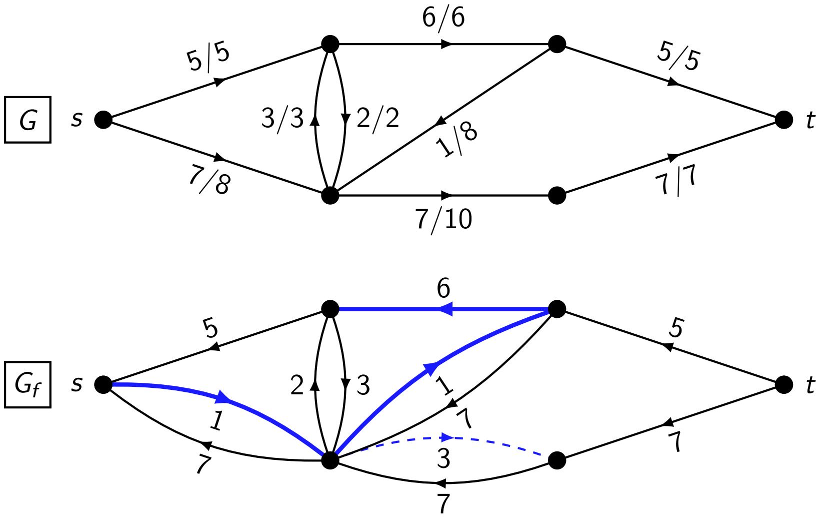
Apply depth-first search to find an augmenting path in G_f.

Worked example



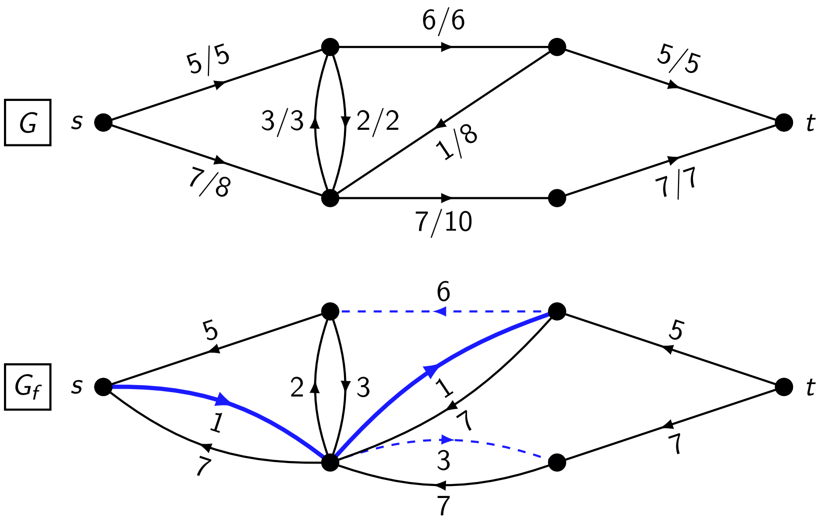
Apply depth-first search to find an augmenting path in G_f.

Worked example



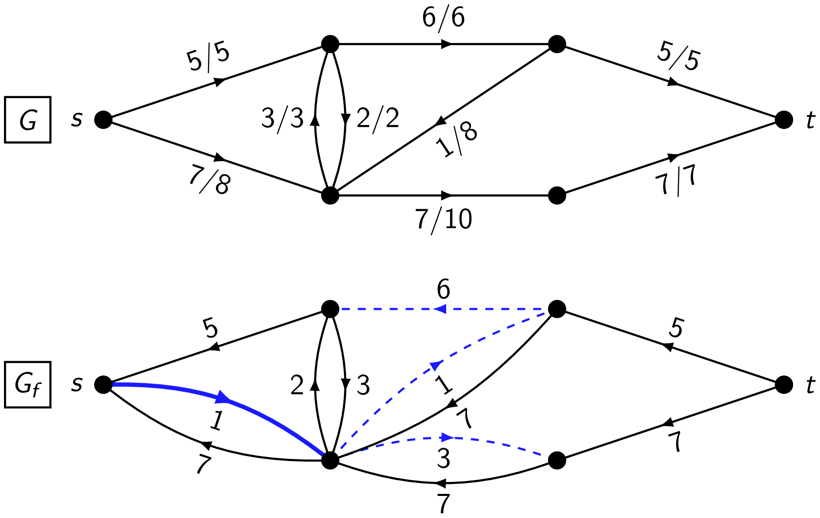
Apply depth-first search to find an augmenting path in G_f.

Worked example



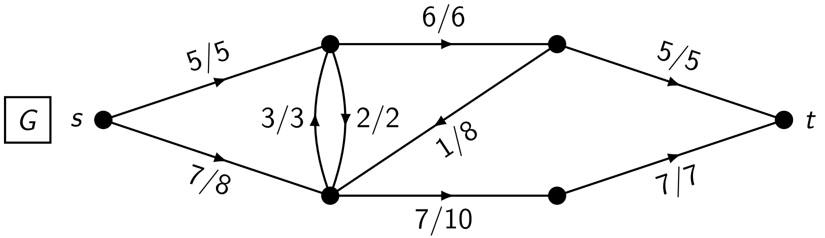
Apply depth-first search to find an augmenting path in G_f.

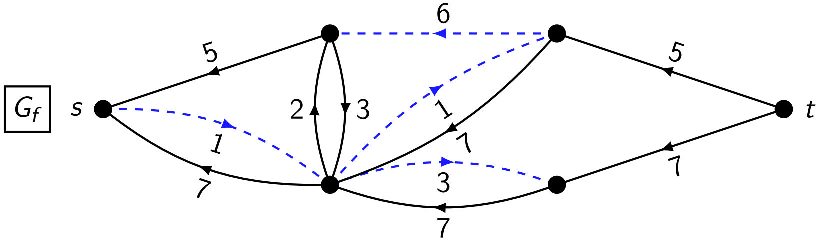
Worked example



Apply depth-first search to find an augmenting path in G_f.

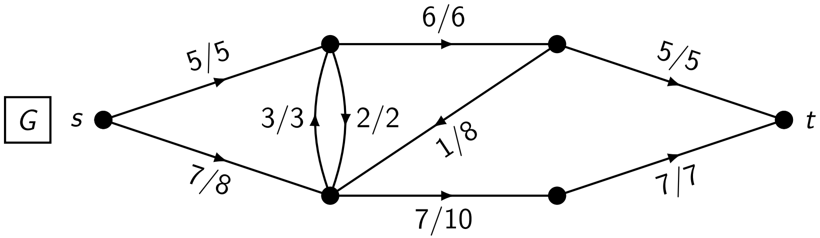
Worked example

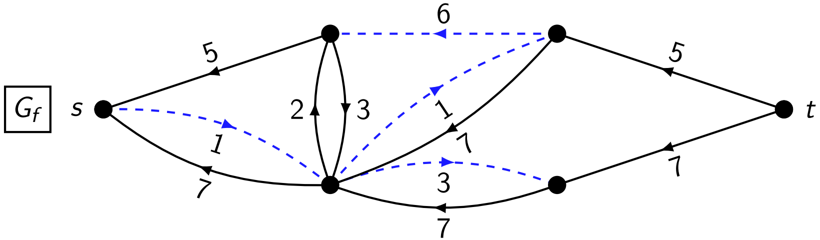




Apply depth-first search to find an augmenting path in G_f.

Worked example



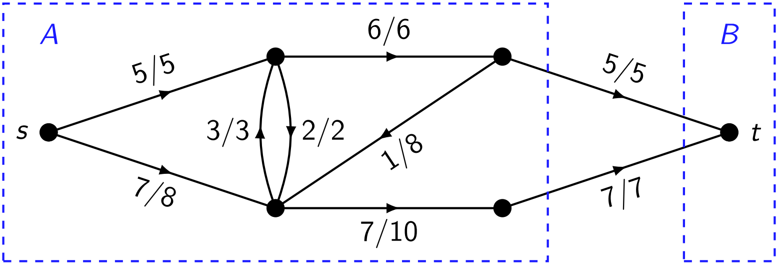


No such path exists, so we're done! This flow has value $5 + 7 = 12$.

Why does this work?

A **cut** is any pair of disjoint sets $A, B \subseteq V$ with $A \cup B = V$, $s \in A$ and $t \in B$. (So A and B partition V, the source is in A and the sink is in B.)

Lemma 2: For all cuts (A, B), $v(f) = f^+(A) - f^-(A) = f^-(B) - f^+(B)$.



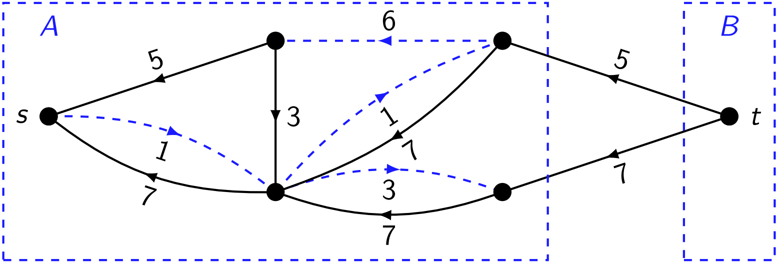
Write $c^+(A) = \sum_{e \text{ out of } A} c(e)$. By Lemma 2, **any** flow g has value $v(g) = g^+(A) - g^-(A) \leq c^+(A) = 12$, and our output flow has value 12. So it must be maximum.

We can use the same argument to prove Ford-Fulkerson always works.

Lemma 2: For all cuts (A, B), $v(f) = f^+(A) - f^-(A) = f^-(B) - f^+(B)$.

To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.

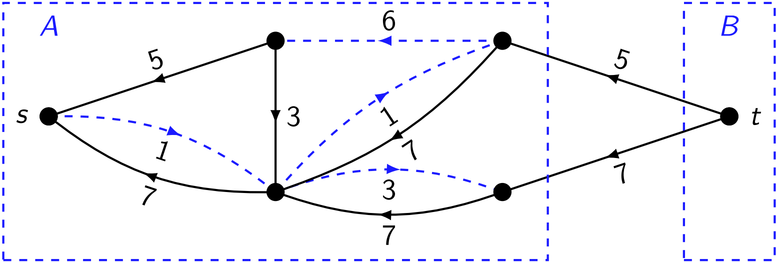
To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.



We take $A = \{v \in V(G)\colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

Lemma 2: For all cuts (A, B), $v(f) = f^+(A) - f^-(A) = f^-(B) - f^+(B)$.

To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.

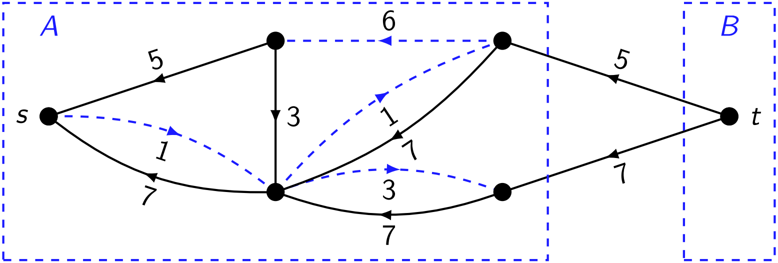


We take $A = \{v \in V(G) \colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

(A, B) is a cut: $s \in A$, and $t \notin A$ since f has no augmenting paths. $\checkmark$

Lemma 2: For all cuts (A, B), $v(f) = f^+(A) - f^-(A) = f^-(B) - f^+(B)$.

To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.

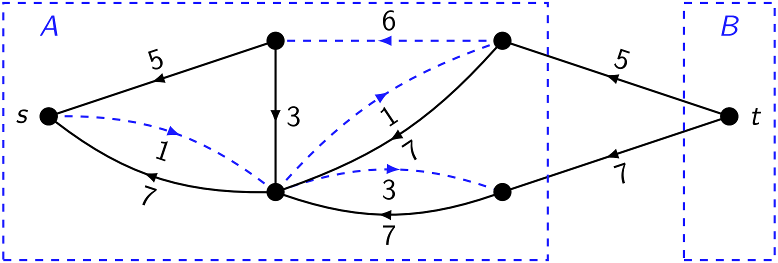


We take $A = \{v \in V(G) \colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

(A, B) is a cut: $\checkmark$

Lemma 2: For all cuts (A, B), $v(f) = f^+(A) - f^-(A) = f^-(B) - f^+(B)$.

To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.

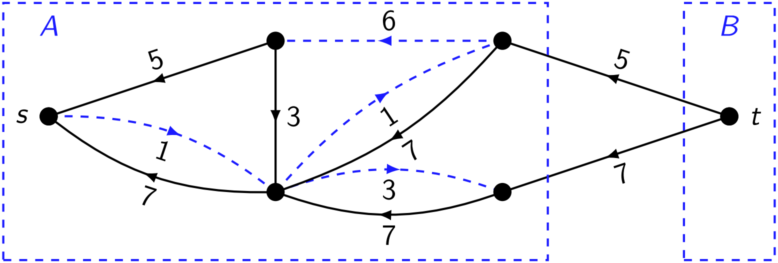


We take $A = \{v \in V(G) \colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

(A, B) is a cut: ✓

$f^+(A) = c^+(A)$: No $A \to B$ forward edges in G_f $\Rightarrow$ every $A \to B$ edge in G is filled to capacity $\Rightarrow f^+(A) = c^+(A)$. ✓

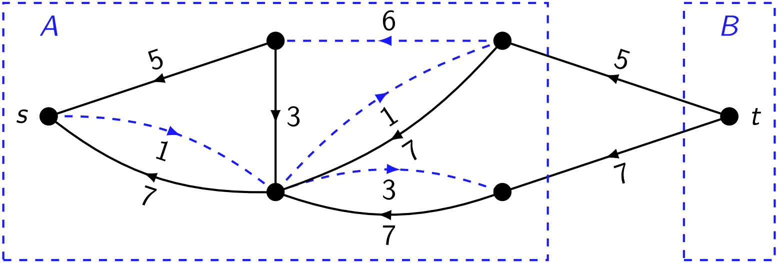
To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.



We take $A = \{v \in V(G) \colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

$\boldsymbol{(A, B)}$ **is a cut:** ✓ $\boldsymbol{f^+(A) = c^+(A)}$**:** ✓

To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.

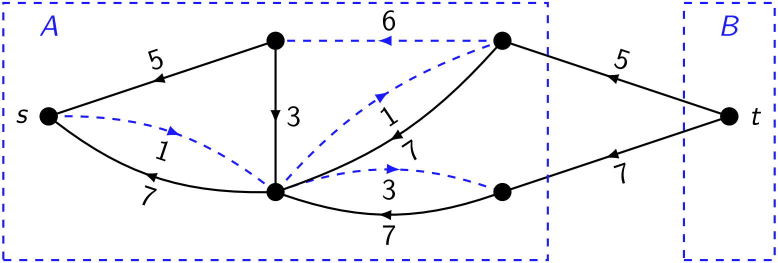


We take $A = \{v \in V(G) \colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

(A, B) is a cut: ✓ **$f^+(A) = c^+(A)$:** ✓

$f^-(A) = 0$: No $A \to B$ backward edges in G_f $\Rightarrow$ every $B \to A$ edge in G has zero flow $\Rightarrow f^-(A) = 0$. ✓

To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.

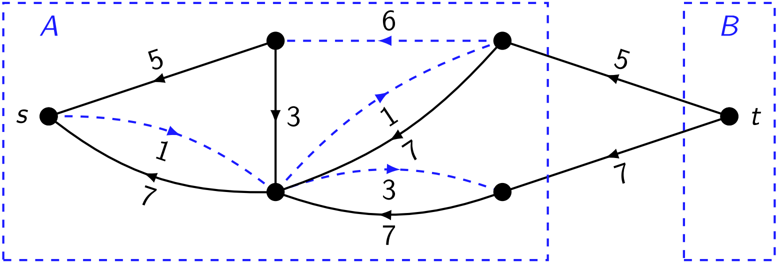


We take $A = \{v \in V(G)\colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

(A, B) is a cut: ✓ **$f^+(A) = c^+(A)$:** ✓ **$f^-(A) = 0$:** ✓

Lemma 2: For all cuts (A, B), $v(f) = f^+(A) - f^-(A) = f^-(B) - f^+(B)$.

To prove the flow f returned by Ford-Fulkerson is **always** maximum by this argument, we will show there is **always** a cut (A, B) with $v(f) = c^+(A)$, i.e. with $f^+(A) = c^+(A)$ and $f^-(A) = 0$.



We take $A = \{v \in V(G) \colon v \text{ reachable from } s \text{ in } G_f\}$, and $B = V(G) \setminus A$.

$\boldsymbol{(A, B)}$ **is a cut:** ✓ $\boldsymbol{f^+(A) = c^+(A)}$**:** ✓ $\boldsymbol{f^-(A) = 0}$**:** ✓

So by Lemma 2, every other flow g has value $g^+(A) - g^-(A) \leq c^+(A) = v(f)$. Thus f is a maximum flow and Ford-Fulkerson is correct. □

We have proved three results for the price of one!

Theorem: Ford-Fulkerson always returns a maximum flow. □

We have proved three results for the price of one!

Theorem: Ford-Fulkerson always returns a maximum flow. □

Theorem: There is always a maximum flow with integer values.

We have proved three results for the price of one!

Theorem: Ford-Fulkerson always returns a maximum flow. □

Theorem: There is always a maximum flow with integer values.

Proof: The maximum flow returned by Ford-Fulkerson has this property. (We can prove this easily with a loop invariant: f starts with value zero, and each iteration of the main loop adds an integer to f's value.) □

We have proved three results for the price of one!

Theorem: Ford-Fulkerson always returns a maximum flow. □

Theorem: There is always a maximum flow with integer values.

Proof: The maximum flow returned by Ford-Fulkerson has this property. (We can prove this easily with a loop invariant: f starts with value zero, and each iteration of the main loop adds an integer to f's value.) □

Max-flow min-cut theorem: The value of a maximum flow is equal to the minimum capacity of a cut, i.e. the minimum value of $c^+(A)$ over all cuts (A, B).

We have proved three results for the price of one!

Theorem: Ford-Fulkerson always returns a maximum flow. $\square$

Theorem: There is always a maximum flow with integer values.

Proof: The maximum flow returned by Ford-Fulkerson has this property. (We can prove this easily with a loop invariant: f starts with value zero, and each iteration of the main loop adds an integer to f's value.) $\square$

Max-flow min-cut theorem: The value of a maximum flow is equal to the minimum capacity of a cut, i.e. the minimum value of $c^+(A)$ over all cuts (A, B).

Proof: Let f be a maximum flow, and let (A, B) be a cut minimising $c^+(A)$. We already proved $v(f) \leq c^+(A)$. Moreover, there is no augmenting path for f, so exactly as before, there is a cut (A', B') with $c^+(A') = v(f)$; thus $v(f) \geq c^+(A)$. The result follows. $\square$