

# Preflows

## Definition 1

An  $(s, t)$ -preflow is a function  $f : E \mapsto \mathbb{R}^+$  that satisfies

For each edge  $e \in E$

$$0 \leq f(e) \leq c(e)$$

For each vertex  $v \in V$

$$\sum_{e \in E^+} f(e) - \sum_{e \in E^-} f(e) \leq b(v)$$

For each vertex  $v \in V$

$$\sum_{e \in E^+} f(e) - \sum_{e \in E^-} f(e) = 0$$

# Preflows

## Definition 1

An  $(s, t)$ -preflow is a function  $f : E \mapsto \mathbb{R}^+$  that satisfies

1. For each edge  $e$

$$0 \leq f(e) \leq c(e) .$$

(capacity constraints)

2. For each  $v \in V \setminus \{s, t\}$

$$\sum_{e \in \text{out}(v)} f(e) \leq \sum_{e \in \text{into}(v)} f(e) .$$

# Preflows

## Definition 1

An  $(s, t)$ -preflow is a function  $f : E \mapsto \mathbb{R}^+$  that satisfies

1. For each edge  $e$

$$0 \leq f(e) \leq c(e) .$$

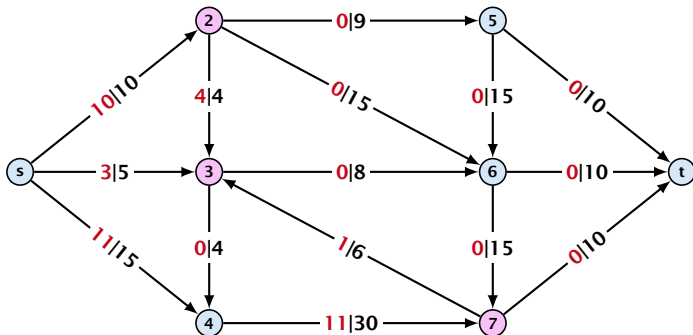
(capacity constraints)

2. For each  $v \in V \setminus \{s, t\}$

$$\sum_{e \in \text{out}(v)} f(e) \leq \sum_{e \in \text{into}(v)} f(e) .$$

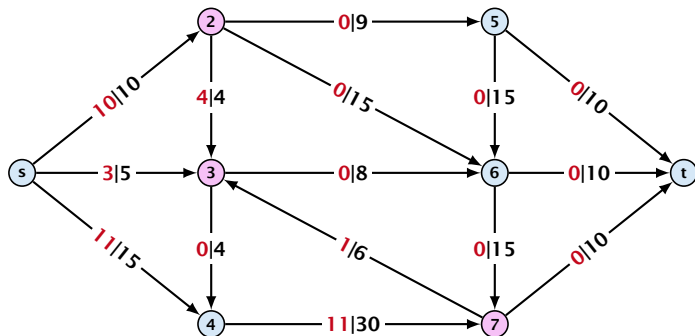
# Preflows

## Example 2



# Preflows

## Example 2



A node that has  $\sum_{e \in \text{out}(v)} f(e) < \sum_{e \in \text{into}(v)} f(e)$  is called an **active node**.

# Preflows

# Preflows

## Definition:

A **labelling** is a function  $\ell : V \rightarrow \mathbb{N}$ . It is **valid** for preflow  $f$  if

- ▶  $\ell(u) \leq \ell(v) + 1$  for all edges  $(u, v)$  in the residual graph  $G_f$  (only non-zero capacity edges!!!)

# Preflows

## Definition:

A **labelling** is a function  $\ell : V \rightarrow \mathbb{N}$ . It is **valid** for preflow  $f$  if

- ▶  $\ell(u) \leq \ell(v) + 1$  for all edges  $(u, v)$  in the residual graph  $G_f$  (only non-zero capacity edges!!!)
- ▶  $\ell(s) = n$



# Preflows

## Definition:

A **labelling** is a function  $\ell : V \rightarrow \mathbb{N}$ . It is **valid** for preflow  $f$  if

- ▶  $\ell(u) \leq \ell(v) + 1$  for all edges  $(u, v)$  in the residual graph  $G_f$  (only non-zero capacity edges!!!)
- ▶  $\ell(s) = n$
- ▶  $\ell(t) = 0$

# Preflows

## Definition:

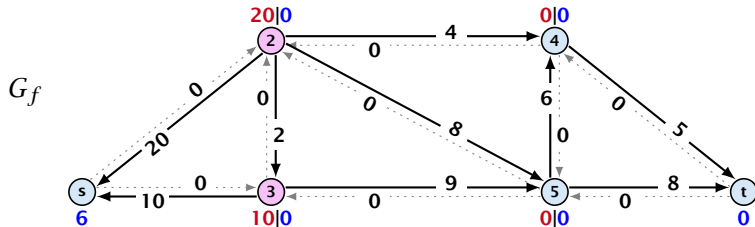
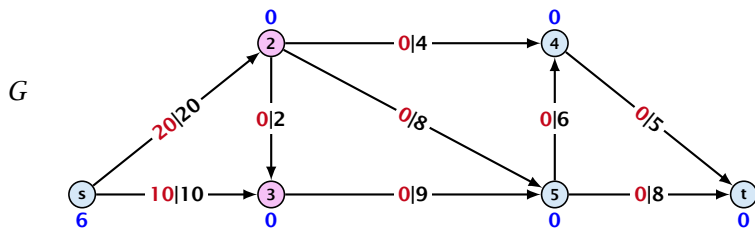
A **labelling** is a function  $\ell : V \rightarrow \mathbb{N}$ . It is **valid** for preflow  $f$  if

- ▶  $\ell(u) \leq \ell(v) + 1$  for all edges  $(u, v)$  in the residual graph  $G_f$  (only non-zero capacity edges!!!)
- ▶  $\ell(s) = n$
- ▶  $\ell(t) = 0$

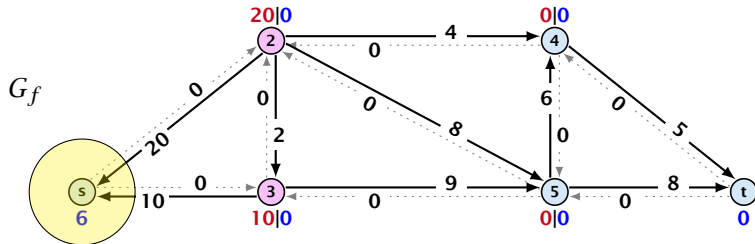
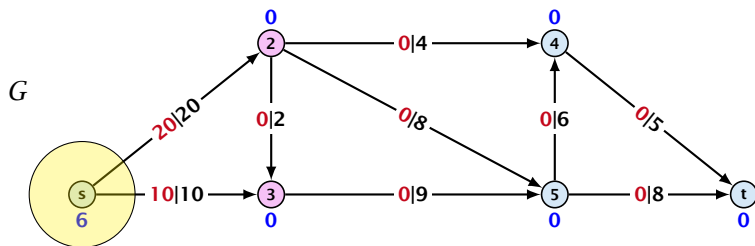
## Intuition:

The labelling can be viewed as a height function. Whenever the height from node  $u$  to node  $v$  decreases by more than 1 (i.e., it goes very steep downhill from  $u$  to  $v$ ), the corresponding edge must be saturated.

# Preflows



# Preflows



# Preflows

# Preflows

## Lemma 3

A *preflow* that has a valid labelling saturates a cut.

# Preflows

## Lemma 3

A *preflow* that has a valid labelling saturates a cut.

### Proof:

- ▶ There are  $n$  nodes but  $n + 1$  different labels from  $0, \dots, n$ .

# Preflows

## Lemma 3

A *preflow* that has a valid labelling saturates a cut.

### Proof:

- ▶ There are  $n$  nodes but  $n + 1$  different labels from  $0, \dots, n$ .
- ▶ There must exist a label  $d \in \{0, \dots, n\}$  such that none of the nodes carries this label.



# Preflows

## Lemma 3

A *preflow* that has a valid labelling saturates a cut.

### Proof:

- ▶ There are  $n$  nodes but  $n + 1$  different labels from  $0, \dots, n$ .
- ▶ There must exist a label  $d \in \{0, \dots, n\}$  such that none of the nodes carries this label.
- ▶ Let  $A = \{v \in V \mid \ell(v) > d\}$  and  $B = \{v \in V \mid \ell(v) < d\}$ .

# Preflows

## Lemma 3

A *preflow* that has a valid labelling saturates a cut.

### Proof:

- ▶ There are  $n$  nodes but  $n + 1$  different labels from  $0, \dots, n$ .
- ▶ There must exist a label  $d \in \{0, \dots, n\}$  such that none of the nodes carries this label.
- ▶ Let  $A = \{v \in V \mid \ell(v) > d\}$  and  $B = \{v \in V \mid \ell(v) < d\}$ .
- ▶ We have  $s \in A$  and  $t \in B$  and there is no edge from  $A$  to  $B$  in the residual graph  $G_f$ ; this means that  $(A, B)$  is a saturated cut.



# Push Relabel Algorithms

# Push Relabel Algorithms

## Idea:

- ▶ start with some preflow and some valid labelling

# Push Relabel Algorithms

## Idea:

- ▶ start with some preflow and some valid labelling
- ▶ successively change the preflow while maintaining a valid labelling

# Push Relabel Algorithms

## Idea:

- ▶ start with some preflow and some valid labelling
- ▶ successively change the preflow while maintaining a valid labelling
- ▶ stop when you have a flow (i.e., no more active nodes)

## Changing a Preflow

An arc  $(u, v)$  with  $c_f(u, v) > 0$  in the residual graph is **admissible** if  $\ell(u) = \ell(v) + 1$  (i.e., it goes downwards w.r.t. labelling  $\ell$ ).

### The push operation

Consider an active node  $u$  with **excess flow**

$f(u) = \sum_{e \in \text{into}(u)} f(e) - \sum_{e \in \text{out}(u)} f(e)$  and suppose  $e = (u, v)$  is an admissible arc with residual capacity  $c_f(e)$ .

We can send flow  $\min\{c_f(e), f(u)\}$  along  $e$  and obtain a new preflow. The old labelling is still valid (!!!).

• The arc  $e$  is deleted from the residual graph

• The node  $u$  becomes inactive



## Changing a Preflow

An arc  $(u, v)$  with  $c_f(u, v) > 0$  in the residual graph is **admissible** if  $\ell(u) = \ell(v) + 1$  (i.e., it goes downwards w.r.t. labelling  $\ell$ ).

### The push operation

Consider an active node  $u$  with **excess flow**

$f(u) = \sum_{e \in \text{into}(u)} f(e) - \sum_{e \in \text{out}(u)} f(e)$  and suppose  $e = (u, v)$  is an admissible arc with residual capacity  $c_f(e)$ .

We can send flow  $\min\{c_f(e), f(u)\}$  along  $e$  and obtain a new preflow. The old labelling is still valid (!!!).

The arc  $e$  is deleted from the residual graph.

The node  $u$  becomes inactive.

## Changing a Preflow

An arc  $(u, v)$  with  $c_f(u, v) > 0$  in the residual graph is **admissible** if  $\ell(u) = \ell(v) + 1$  (i.e., it goes downwards w.r.t. labelling  $\ell$ ).

### The push operation

Consider an active node  $u$  with **excess flow**

$f(u) = \sum_{e \in \text{into}(u)} f(e) - \sum_{e \in \text{out}(u)} f(e)$  and suppose  $e = (u, v)$  is an admissible arc with residual capacity  $c_f(e)$ .

We can send flow  $\min\{c_f(e), f(u)\}$  along  $e$  and obtain a new preflow. The old labelling is still valid (!!!).

## Changing a Preflow

An arc  $(u, v)$  with  $c_f(u, v) > 0$  in the residual graph is **admissible** if  $\ell(u) = \ell(v) + 1$  (i.e., it goes downwards w.r.t. labelling  $\ell$ ).

### The push operation

Consider an active node  $u$  with **excess flow**

$f(u) = \sum_{e \in \text{into}(u)} f(e) - \sum_{e \in \text{out}(u)} f(e)$  and suppose  $e = (u, v)$  is an admissible arc with residual capacity  $c_f(e)$ .

We can send flow  $\min\{c_f(e), f(u)\}$  along  $e$  and obtain a new preflow. The old labelling is still valid (!!!).

## Changing a Preflow

An arc  $(u, v)$  with  $c_f(u, v) > 0$  in the residual graph is **admissible** if  $\ell(u) = \ell(v) + 1$  (i.e., it goes downwards w.r.t. labelling  $\ell$ ).

### The push operation

Consider an active node  $u$  with **excess flow**

$f(u) = \sum_{e \in \text{into}(u)} f(e) - \sum_{e \in \text{out}(u)} f(e)$  and suppose  $e = (u, v)$  is an admissible arc with residual capacity  $c_f(e)$ .

We can send flow  $\min\{c_f(e), f(u)\}$  along  $e$  and obtain a new preflow. The old labelling is still valid (!!!).

- ▶  **saturating push** :  $\min\{f(u), c_f(e)\} = c_f(e)$   
the arc  $e$  is deleted from the residual graph
- ▶  **non-saturating push** :  $\min\{f(u), c_f(e)\} = f(u)$   
the node  $u$  becomes inactive

## Changing a Preflow

An arc  $(u, v)$  with  $c_f(u, v) > 0$  in the residual graph is **admissible** if  $\ell(u) = \ell(v) + 1$  (i.e., it goes downwards w.r.t. labelling  $\ell$ ).

### The push operation

Consider an active node  $u$  with **excess flow**

$f(u) = \sum_{e \in \text{into}(u)} f(e) - \sum_{e \in \text{out}(u)} f(e)$  and suppose  $e = (u, v)$  is an admissible arc with residual capacity  $c_f(e)$ .

We can send flow  $\min\{c_f(e), f(u)\}$  along  $e$  and obtain a new preflow. The old labelling is still valid (!!!).

- ▶ **saturating push**:  $\min\{f(u), c_f(e)\} = c_f(e)$   
the arc  $e$  is deleted from the residual graph
- ▶ **non-saturating push**:  $\min\{f(u), c_f(e)\} = f(u)$   
the node  $u$  becomes inactive

# Push Relabel Algorithms

# Push Relabel Algorithms

## The relabel operation

Consider an active node  $u$  that does not have an outgoing admissible arc.

# Push Relabel Algorithms

## The relabel operation

Consider an active node  $u$  that does not have an outgoing admissible arc.

Increasing the label of  $u$  by 1 results in a valid labelling.



# Push Relabel Algorithms

## The relabel operation

Consider an active node  $u$  that does not have an outgoing admissible arc.

Increasing the label of  $u$  by 1 results in a valid labelling.

- ▶ Edges  $(w, u)$  incoming to  $u$  still fulfill their constraint  $\ell(w) \leq \ell(u) + 1$ .

# Push Relabel Algorithms

## The relabel operation

Consider an active node  $u$  that does not have an outgoing admissible arc.

Increasing the label of  $u$  by 1 results in a valid labelling.

- ▶ Edges  $(w, u)$  incoming to  $u$  still fulfill their constraint  $\ell(w) \leq \ell(u) + 1$ .
- ▶ An outgoing edge  $(u, w)$  had  $\ell(u) < \ell(w) + 1$  before since it was not admissible. Now:  $\ell(u) \leq \ell(w) + 1$ .

# Push Relabel Algorithms

## Intuition:

We want to send flow downwards, since the source has a height/label of  $n$  and the target a height/label of  $0$ . If we see an active node  $u$  with an admissible arc we push the flow at  $u$  towards the other end-point that has a lower height/label. If we do not have an admissible arc but excess flow into  $u$  it should roughly mean that the level/height/label of  $u$  should rise. (If we consider the flow to be water than this would be natural).

Note that the above intuition is very incorrect as the labels are integral, i.e., they cannot really be seen as the height of a node.

# Reminder

- ▶ In a **preflow** nodes may not fulfill conservation constraints; a node may have more incoming flow than outgoing flow.
- ▶ Such a node is called **active**.
- ▶ A labelling is **valid** if for every edge  $(u, v)$  in the residual graph  $\ell(u) \leq \ell(v) + 1$ .
- ▶ An arc  $(u, v)$  in residual graph is **admissible** if  $\ell(u) = \ell(v) + 1$ .
- ▶ A **saturating push** along  $e$  pushes an amount of  $c(e)$  flow along the edge, thereby saturating the edge (and making it disappear from the residual graph).
- ▶ A **non-saturating push** along  $e = (u, v)$  pushes a flow of  $f(u)$ , where  $f(u)$  is the **excess flow** of  $u$ . This makes  $u$  inactive.

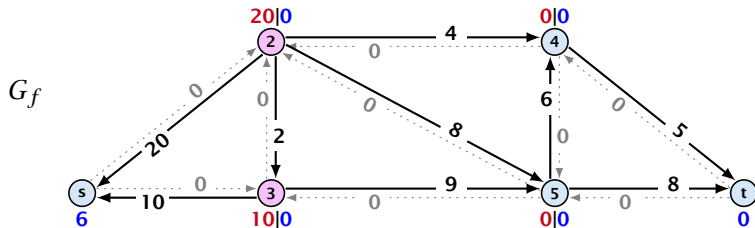
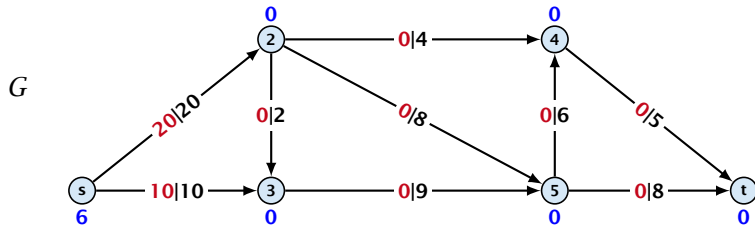
# Push Relabel Algorithms

## Algorithm 3 $\text{maxflow}(G, s, t, c)$

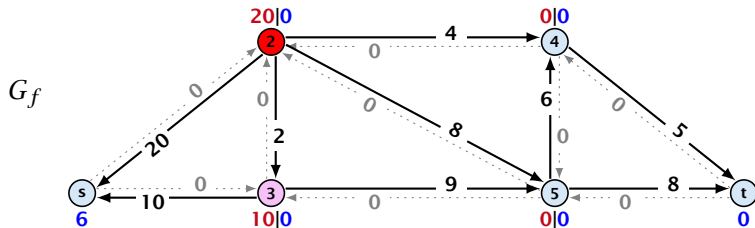
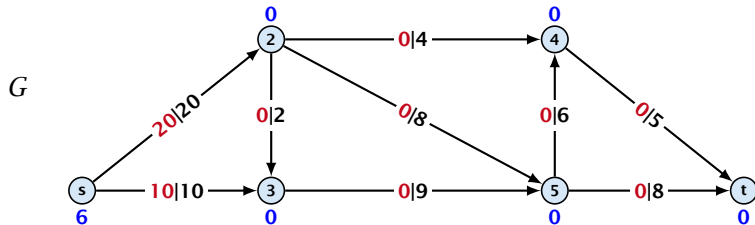
```
1: find initial preflow  $f$ 
2: while there is active node  $u$  do
3:     if there is admiss. arc  $e$  out of  $u$  then
4:          $\text{push}(G, e, f, c)$ 
5:     else
6:          $\text{relabel}(u)$ 
7: return  $f$ 
```



# Preflow Push Algorithm



# Preflow Push Algorithm

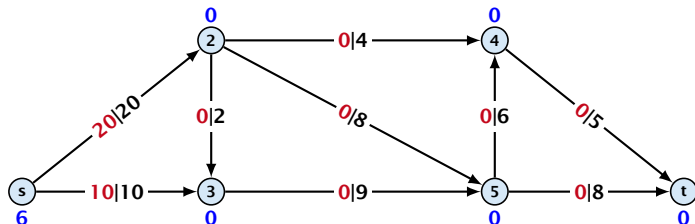




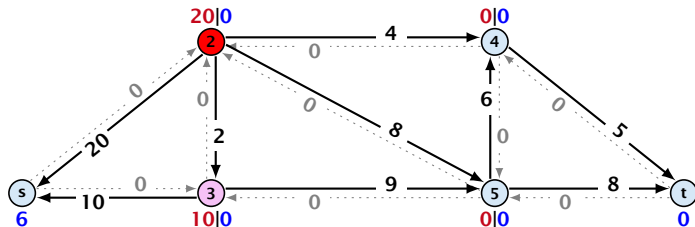
# Preflow Push Algorithm

relabel

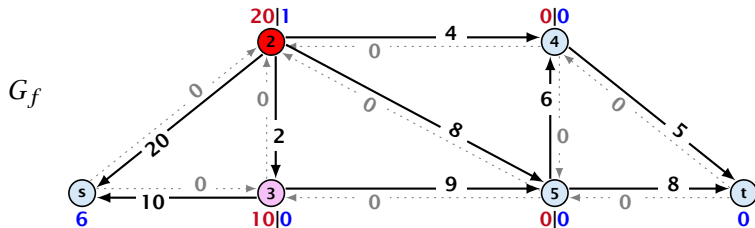
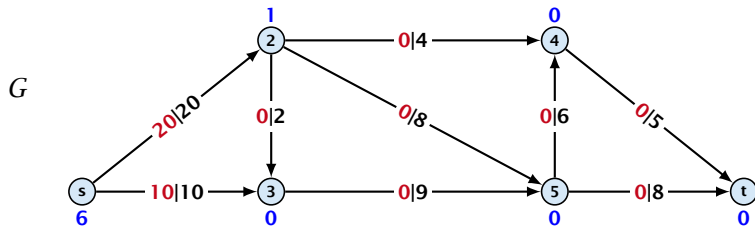
$G$



$G_f$

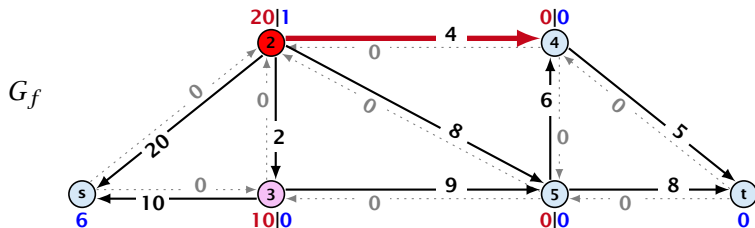
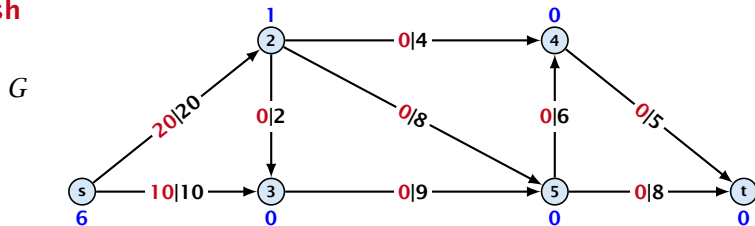


# Preflow Push Algorithm

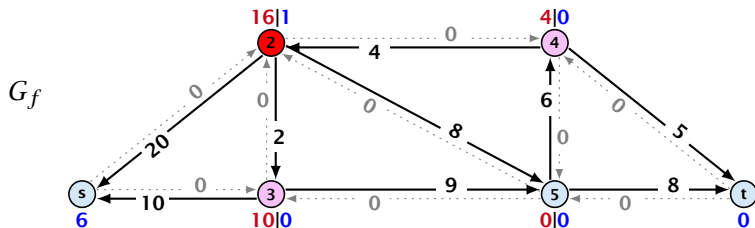
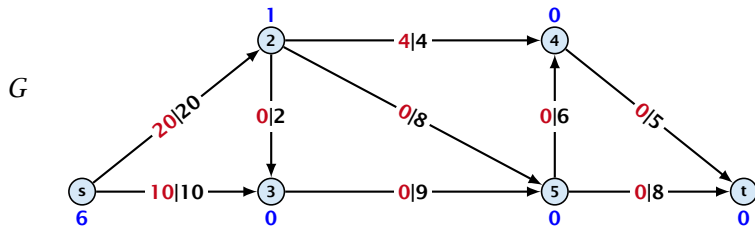


# Preflow Push Algorithm

push



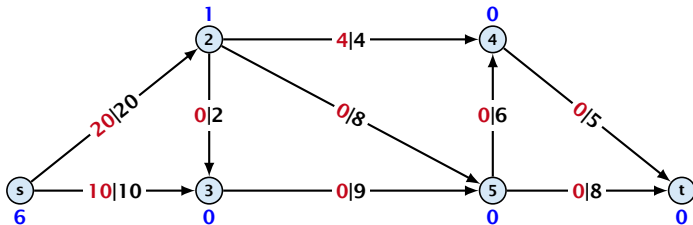
# Preflow Push Algorithm



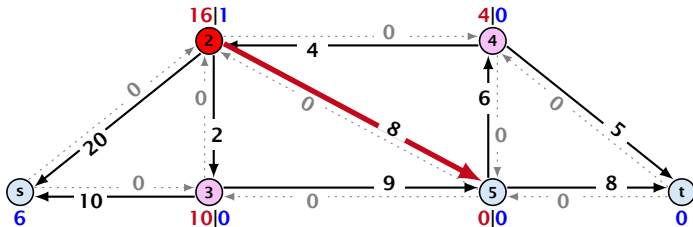
# Preflow Push Algorithm

push

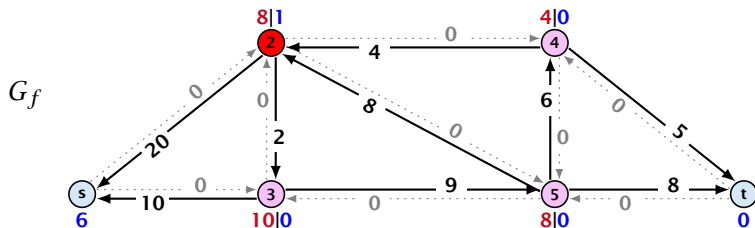
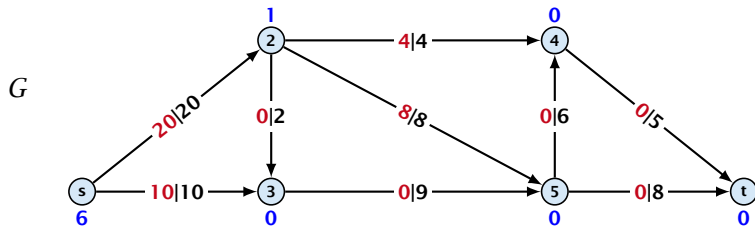
$G$



$G_f$



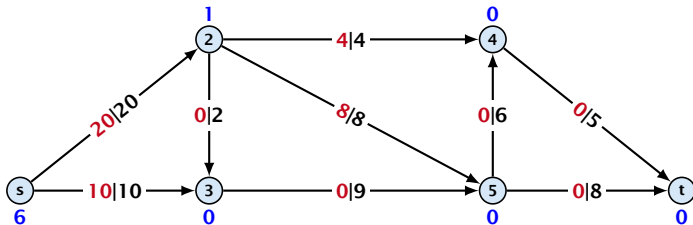
# Preflow Push Algorithm



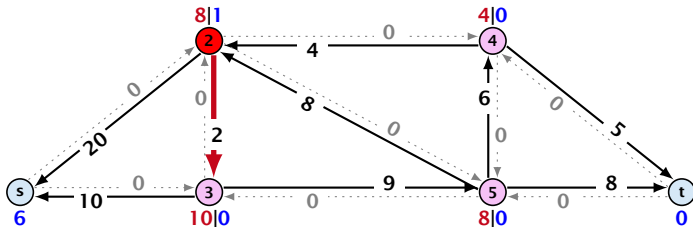
# Preflow Push Algorithm

push

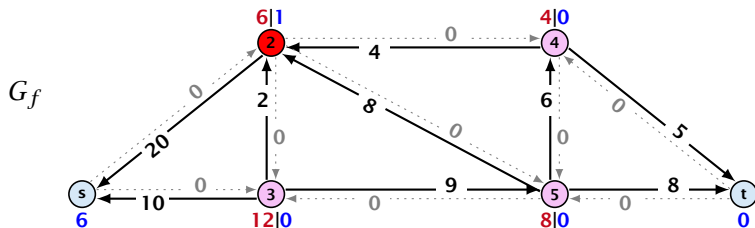
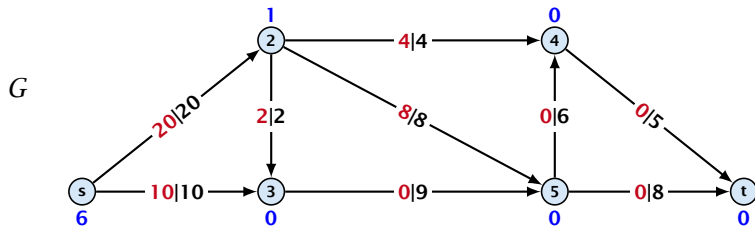
$G$



$G_f$



# Preflow Push Algorithm

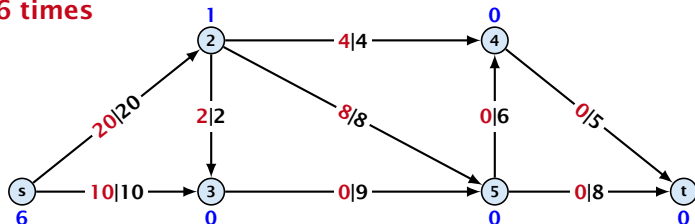




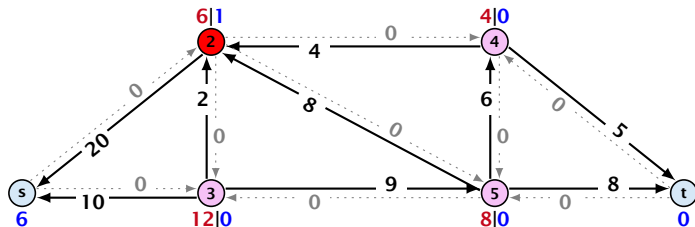
# Preflow Push Algorithm

relabel 6 times

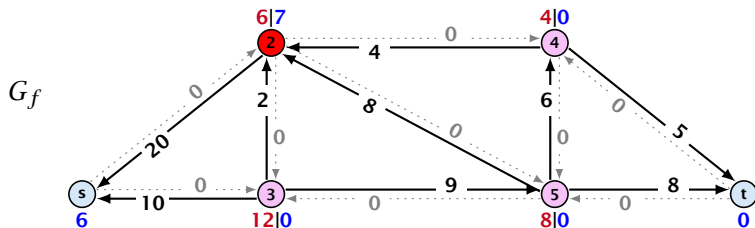
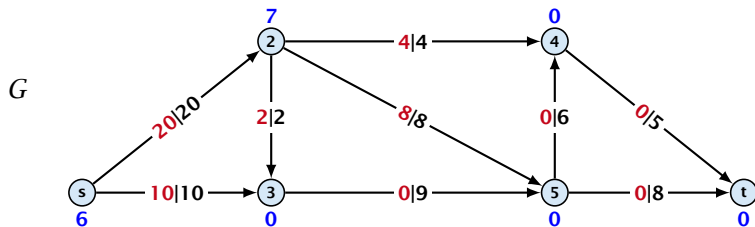
$G$



$G_f$



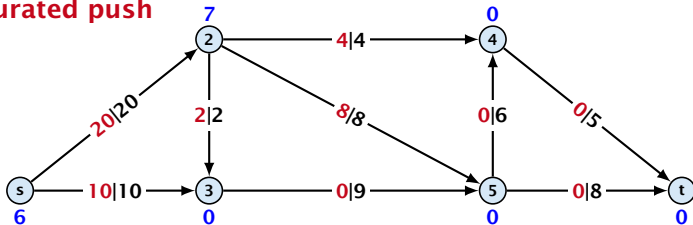
# Preflow Push Algorithm



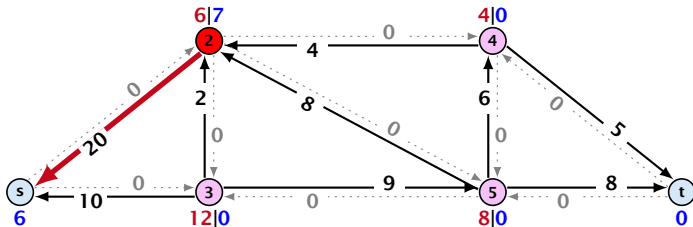
# Preflow Push Algorithm

non-saturated push

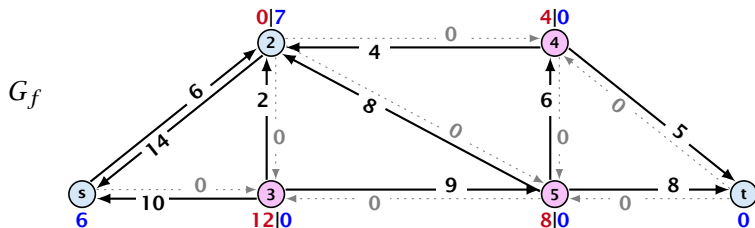
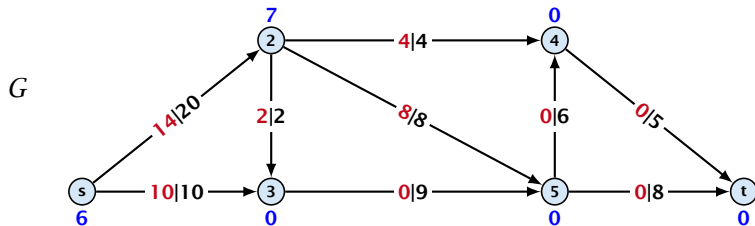
$G$



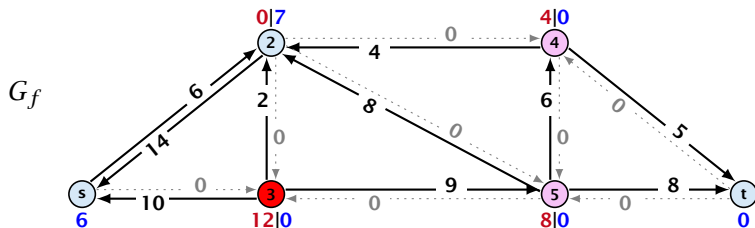
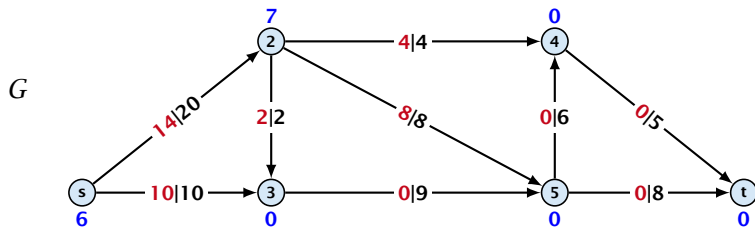
$G_f$



# Preflow Push Algorithm



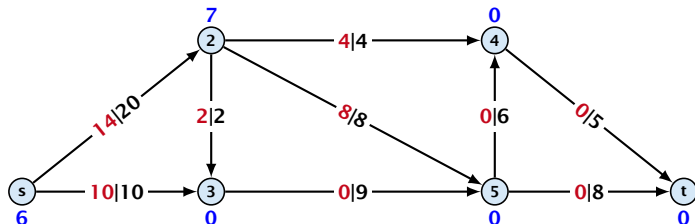
# Preflow Push Algorithm



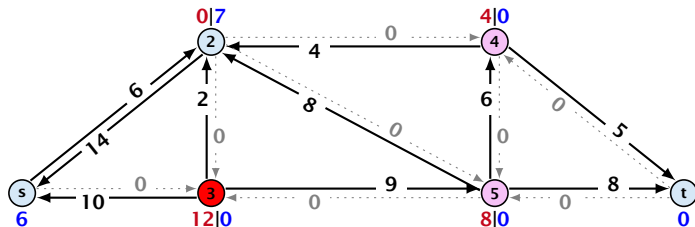
# Preflow Push Algorithm

relabel

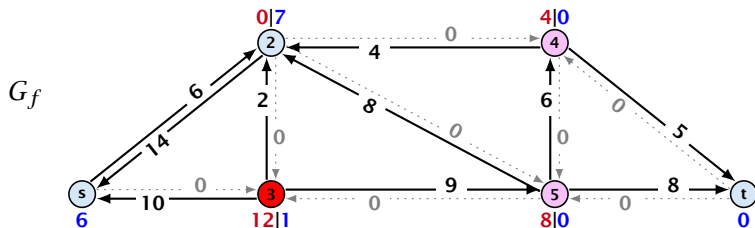
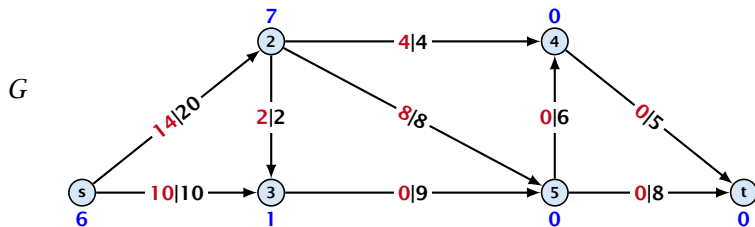
$G$



$G_f$



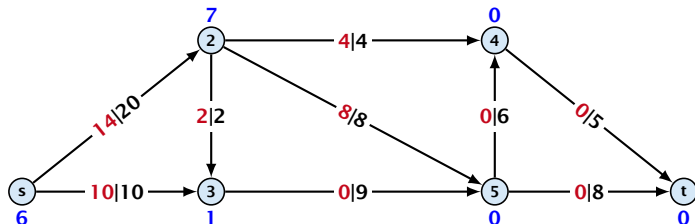
# Preflow Push Algorithm



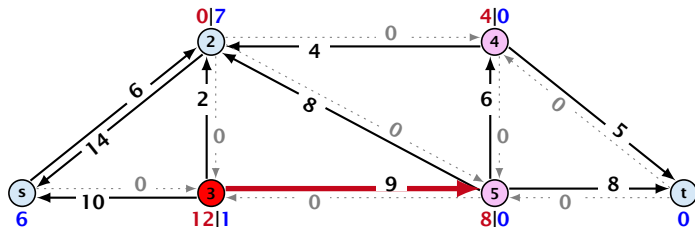
# Preflow Push Algorithm

push

$G$

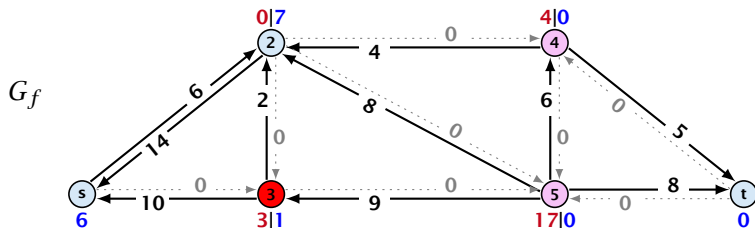
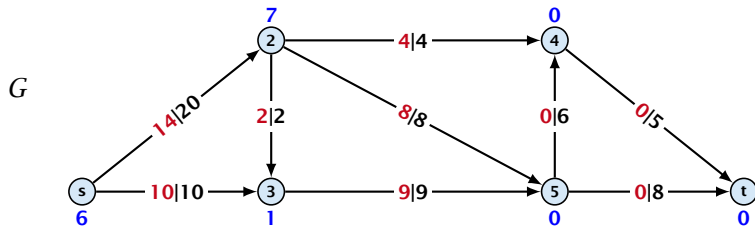


$G_f$





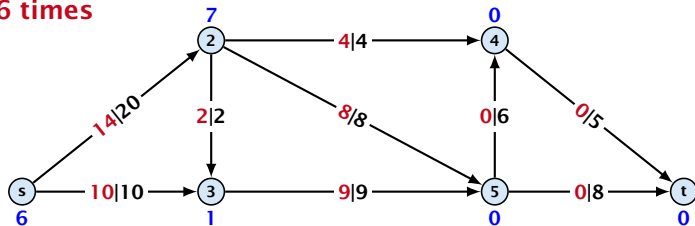
# Preflow Push Algorithm



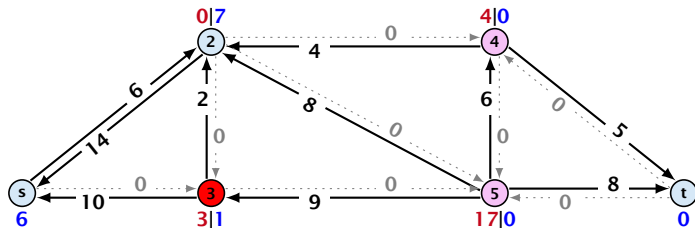
# Preflow Push Algorithm

relabel 6 times

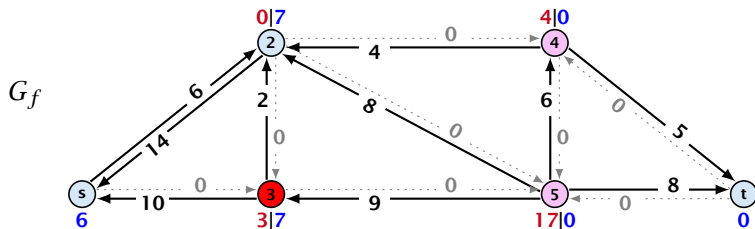
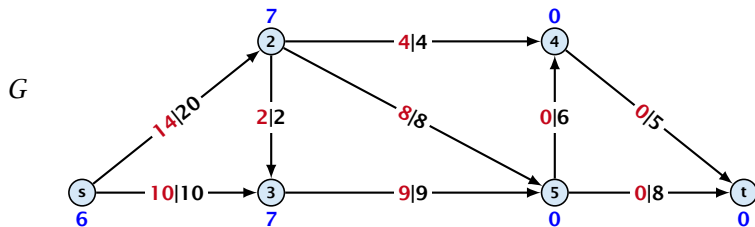
$G$



$G_f$



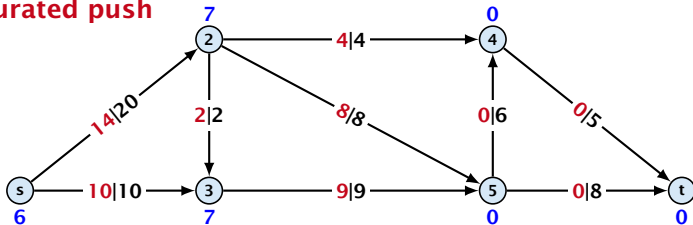
# Preflow Push Algorithm



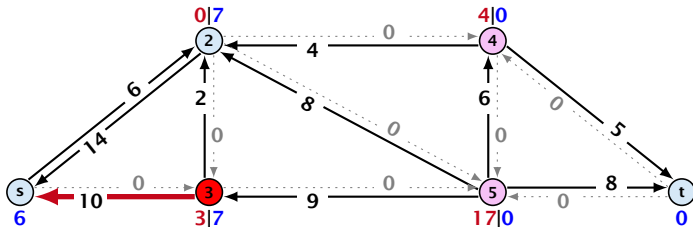
# Preflow Push Algorithm

non-saturated push

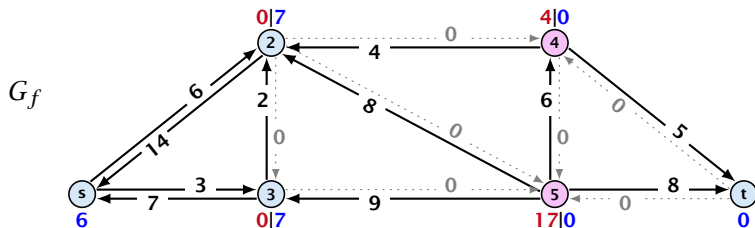
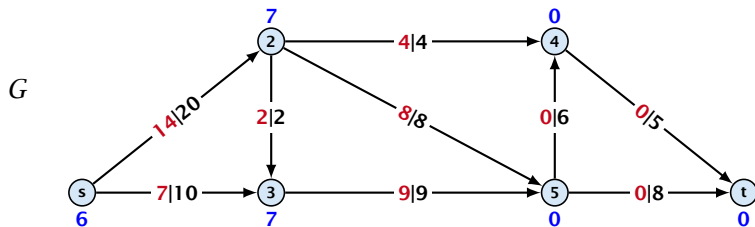
$G$



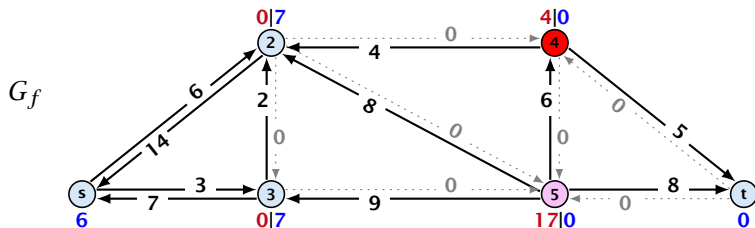
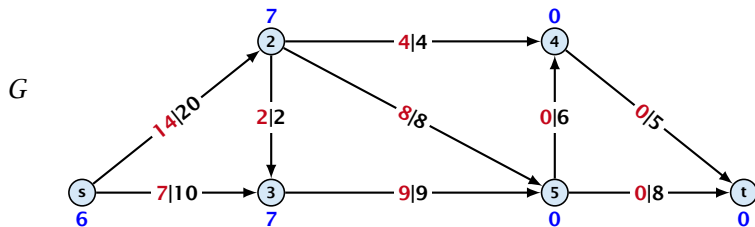
$G_f$



# Preflow Push Algorithm



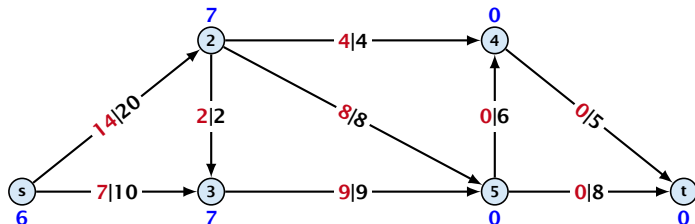
# Preflow Push Algorithm



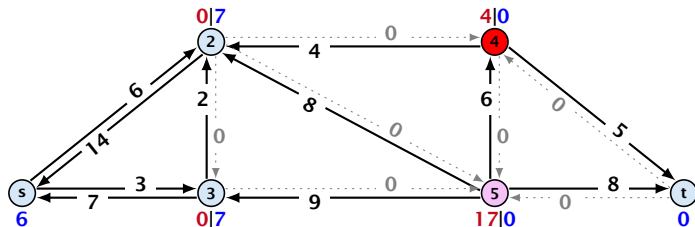
# Preflow Push Algorithm

relabel

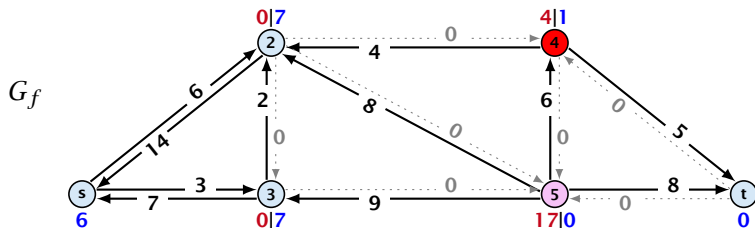
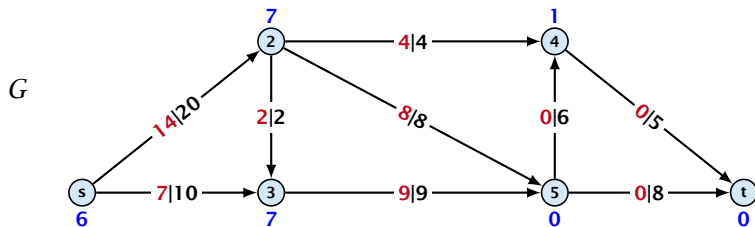
$G$



$G_f$



# Preflow Push Algorithm

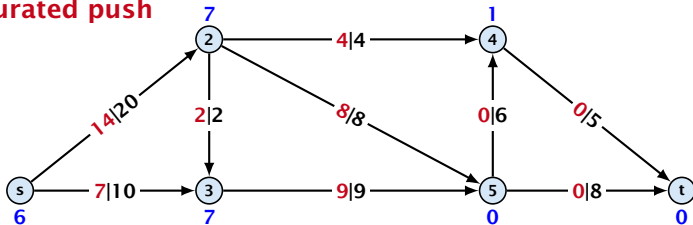




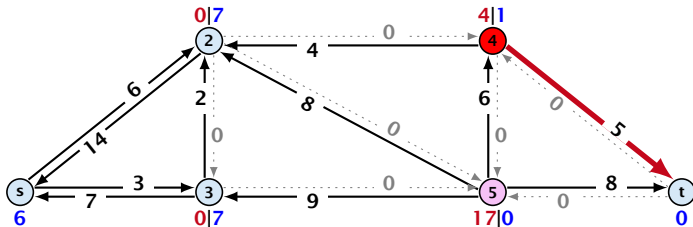
# Preflow Push Algorithm

non-saturated push

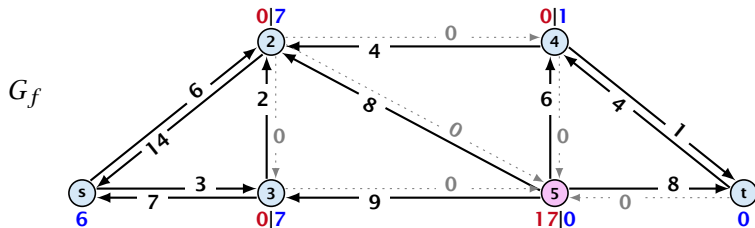
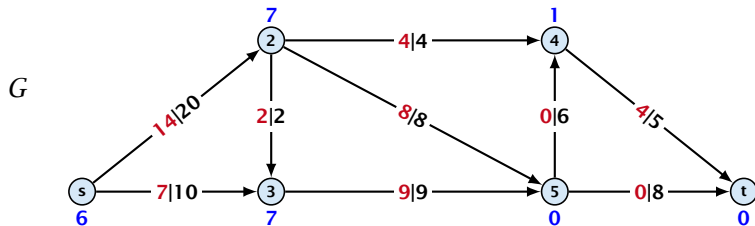
$G$



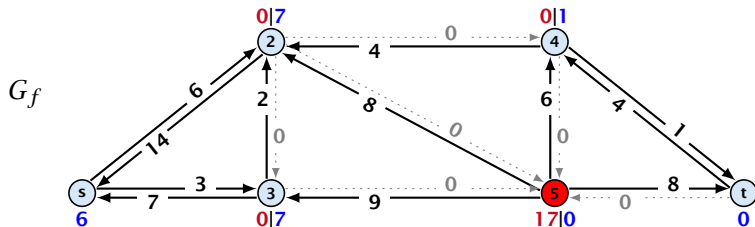
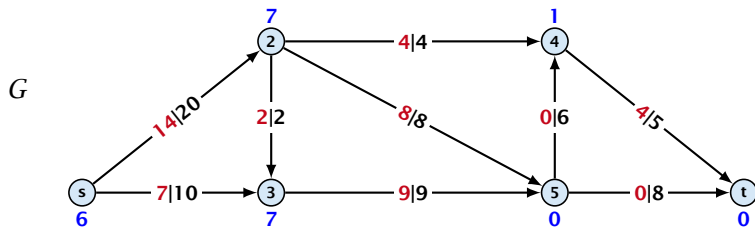
$G_f$



# Preflow Push Algorithm



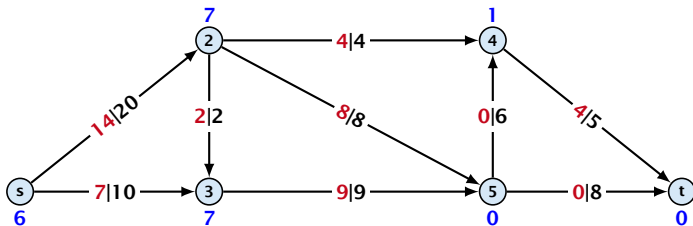
# Preflow Push Algorithm



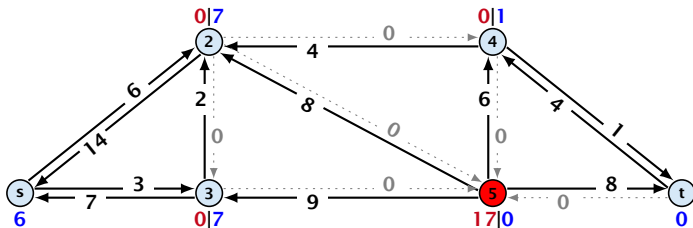
# Preflow Push Algorithm

relabel

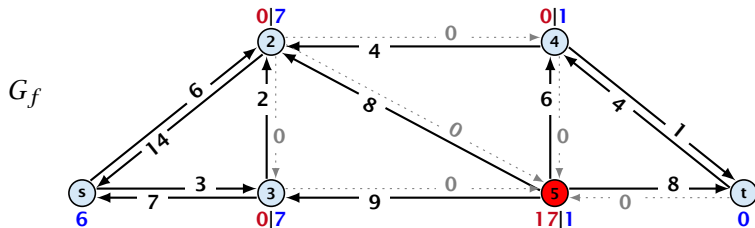
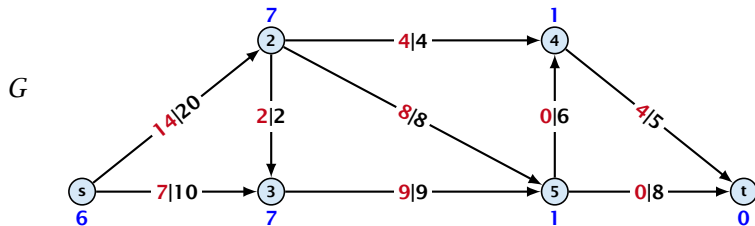
$G$



$G_f$



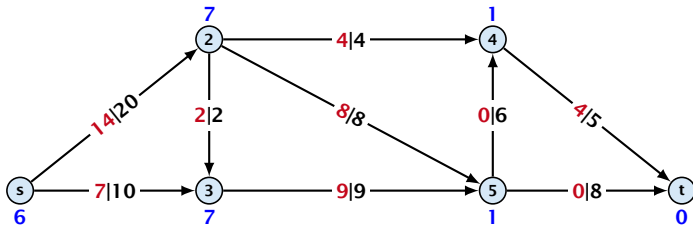
# Preflow Push Algorithm



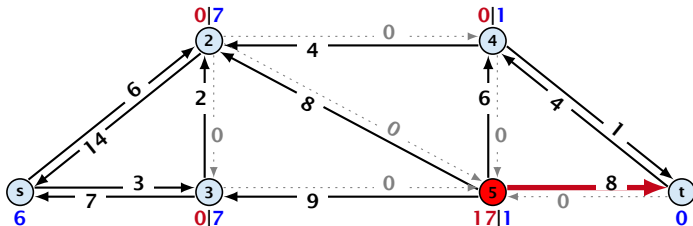
# Preflow Push Algorithm

push

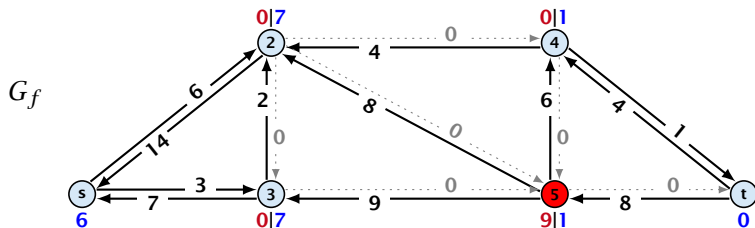
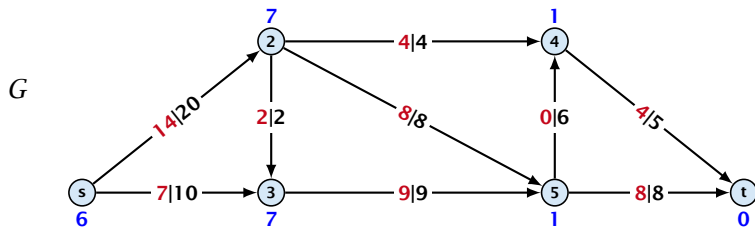
$G$



$G_f$



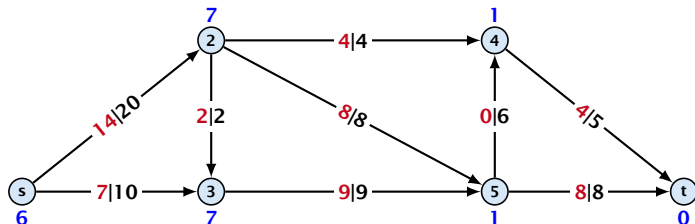
# Preflow Push Algorithm



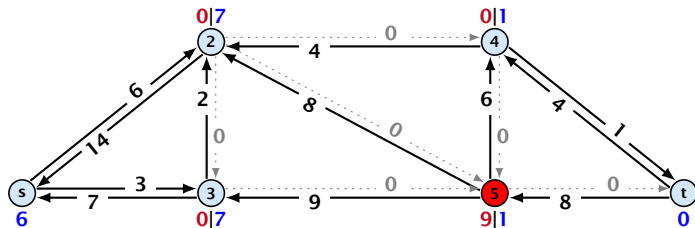
# Preflow Push Algorithm

relabel

$G$

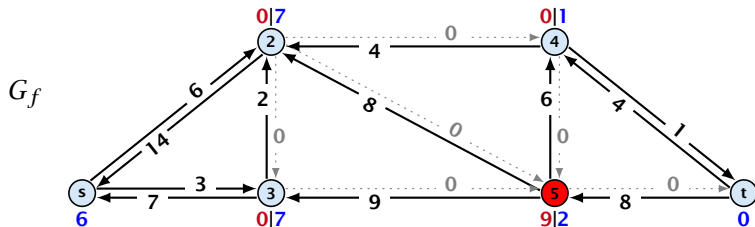
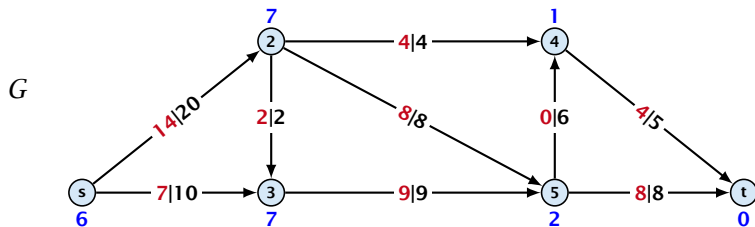


$G_f$





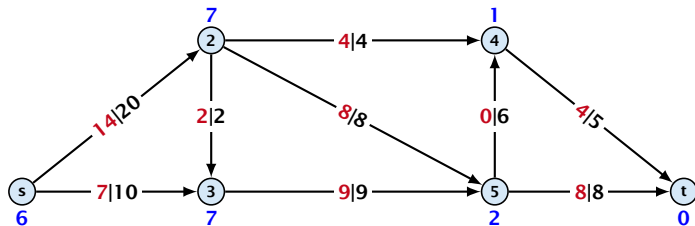
# Preflow Push Algorithm



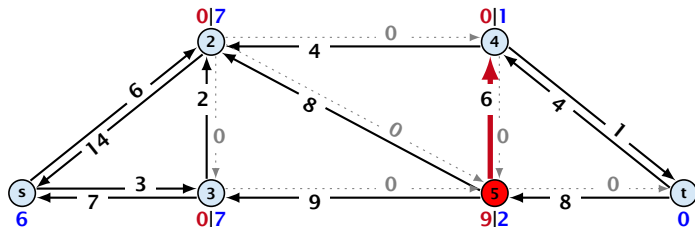
# Preflow Push Algorithm

push

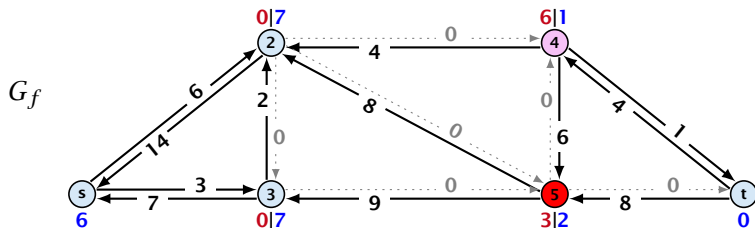
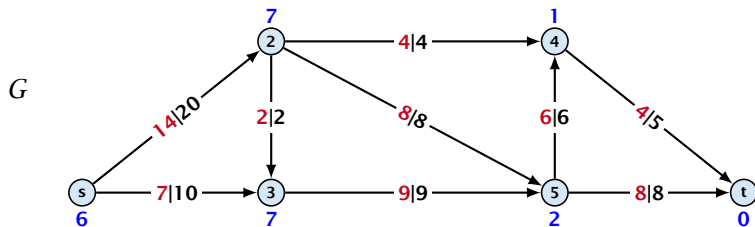
$G$



$G_f$



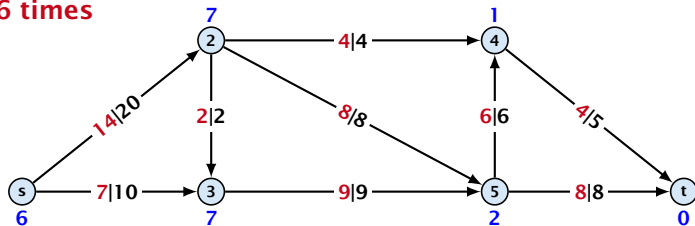
# Preflow Push Algorithm



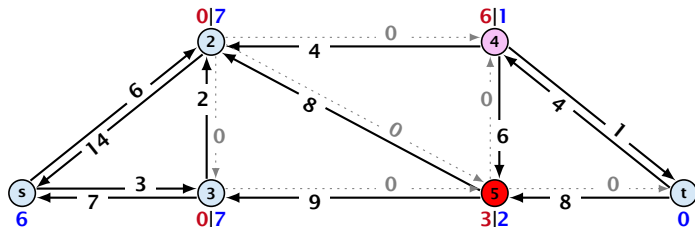
# Preflow Push Algorithm

relabel 6 times

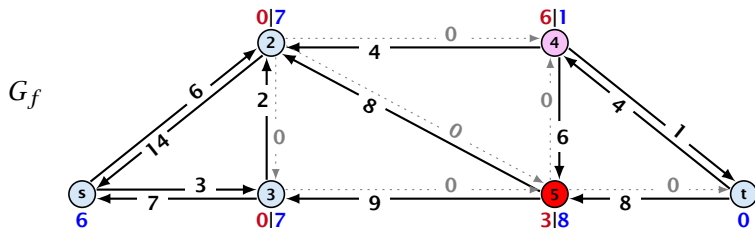
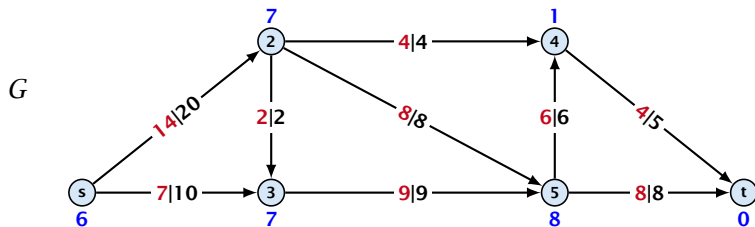
$G$



$G_f$



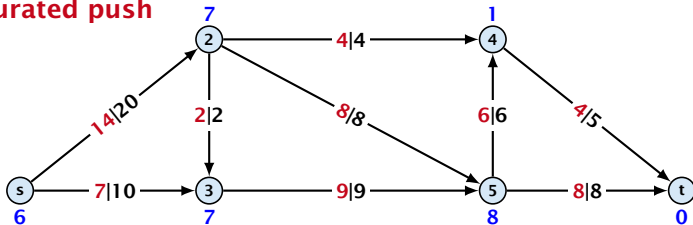
# Preflow Push Algorithm



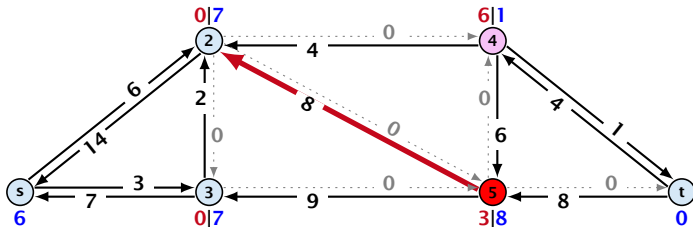
# Preflow Push Algorithm

non-saturated push

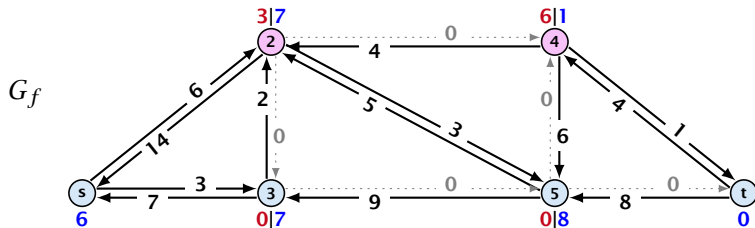
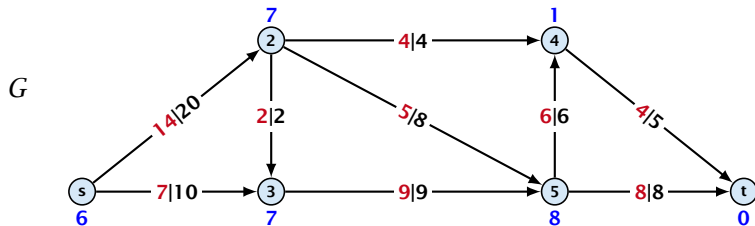
$G$



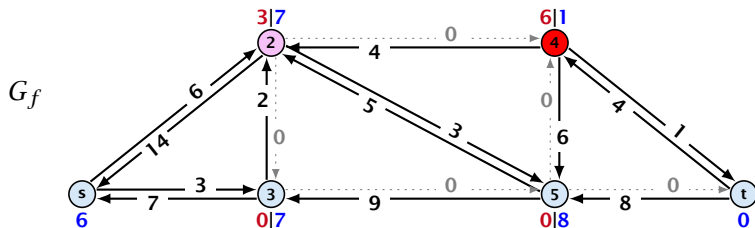
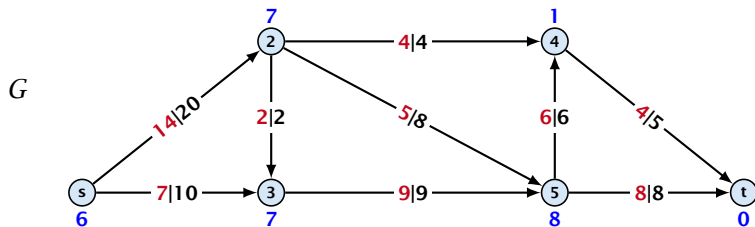
$G_f$



# Preflow Push Algorithm



# Preflow Push Algorithm

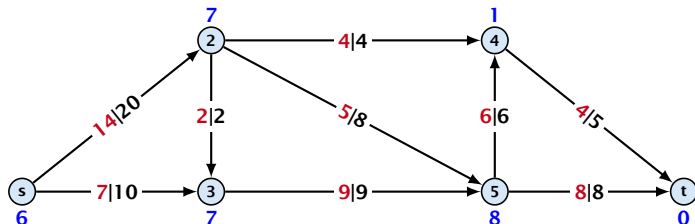




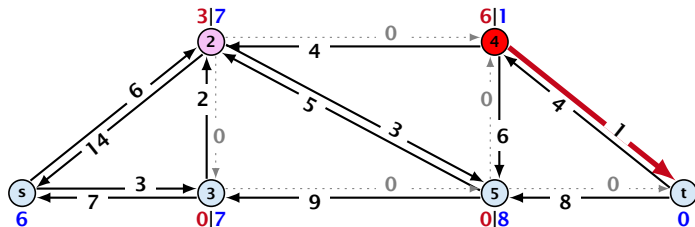
# Preflow Push Algorithm

push

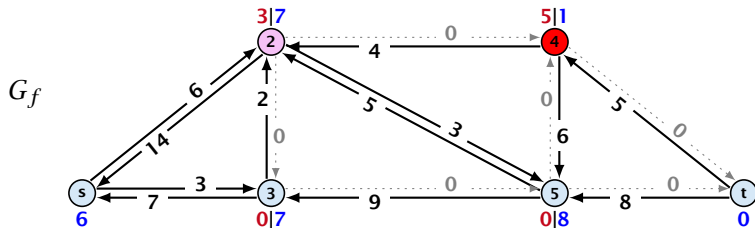
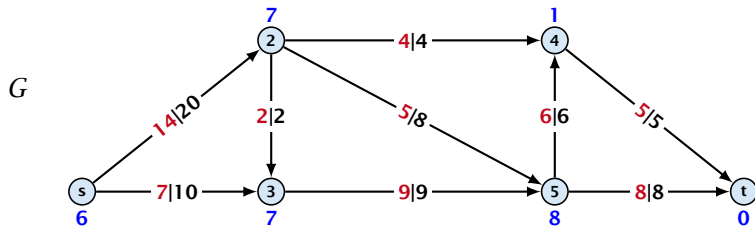
$G$



$G_f$



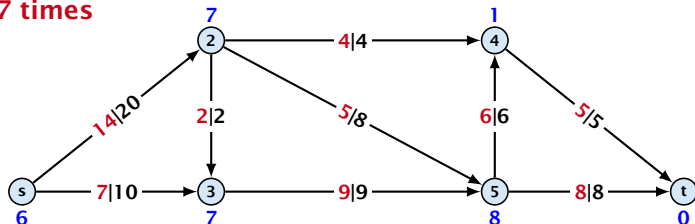
# Preflow Push Algorithm



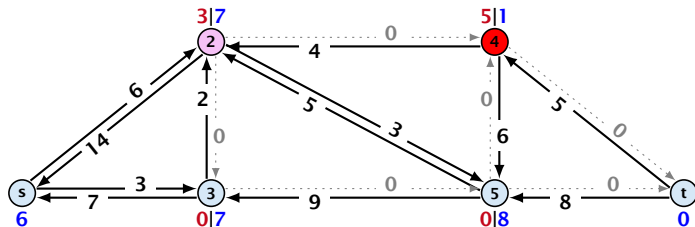
# Preflow Push Algorithm

relabel 7 times

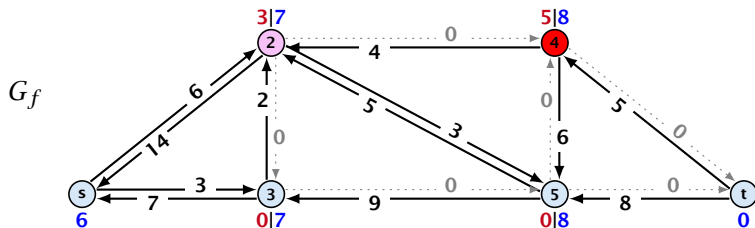
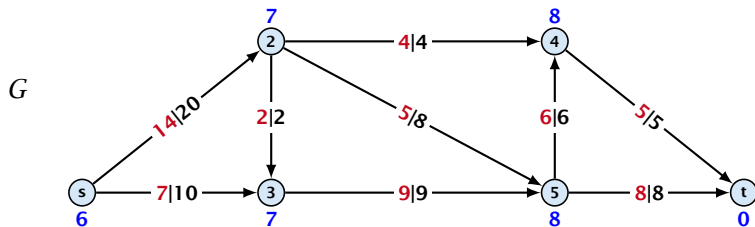
$G$



$G_f$

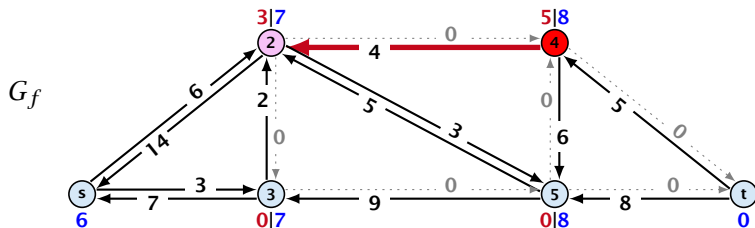
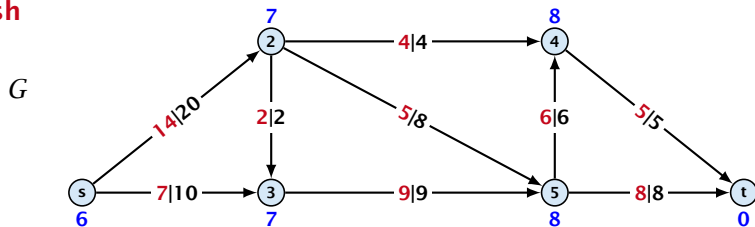


# Preflow Push Algorithm

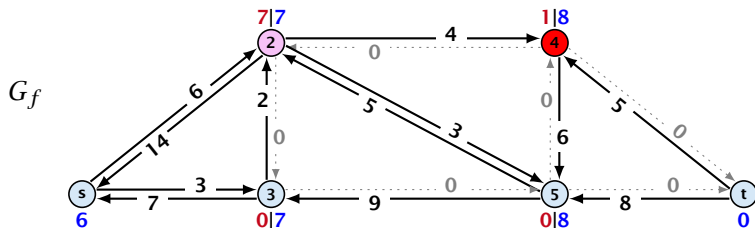
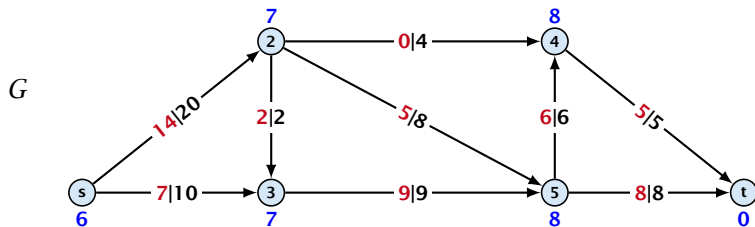


# Preflow Push Algorithm

push



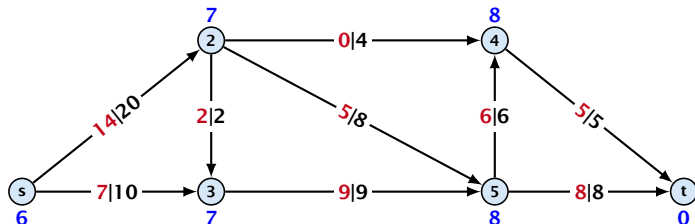
# Preflow Push Algorithm



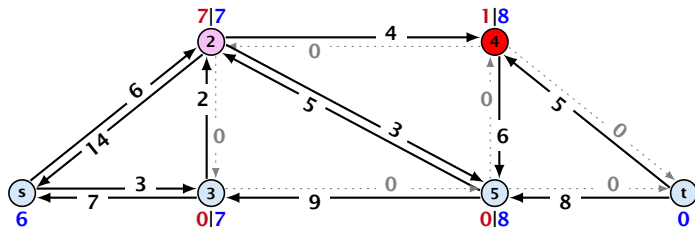
# Preflow Push Algorithm

relabel

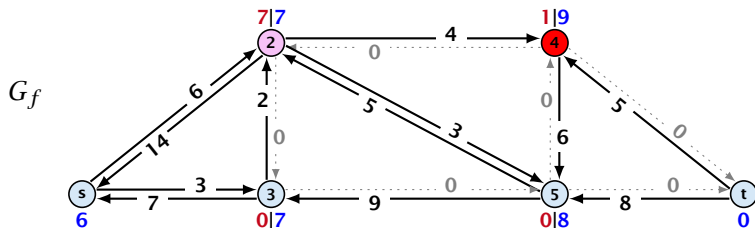
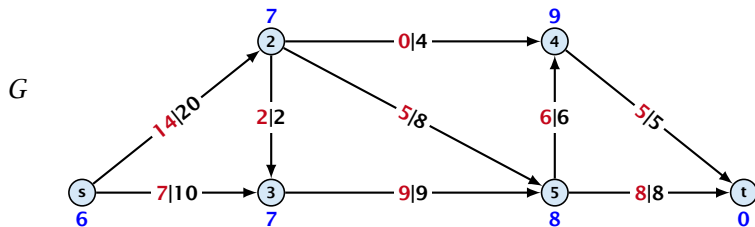
$G$



$G_f$



# Preflow Push Algorithm

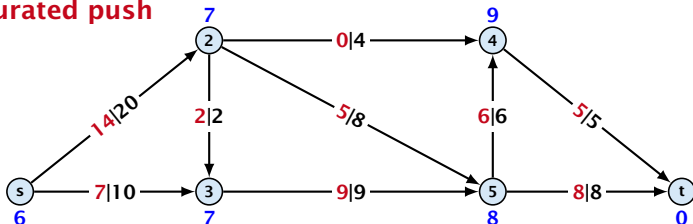




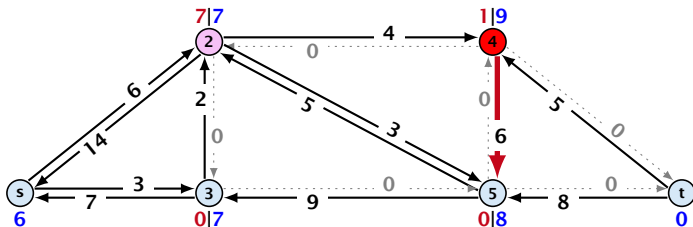
# Preflow Push Algorithm

non-saturated push

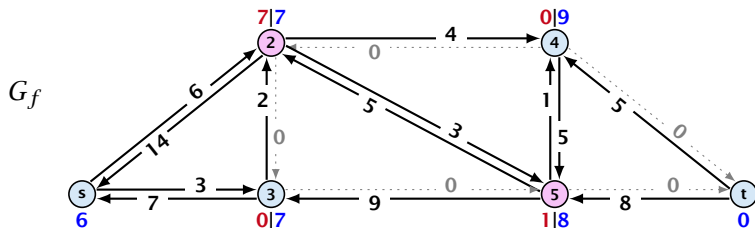
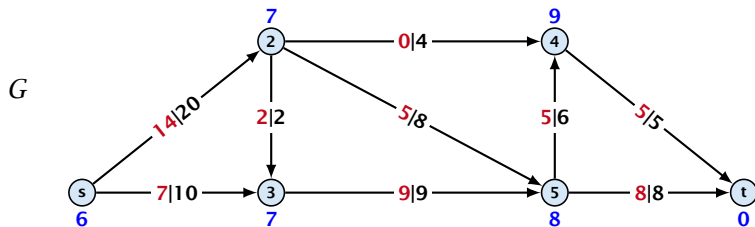
$G$



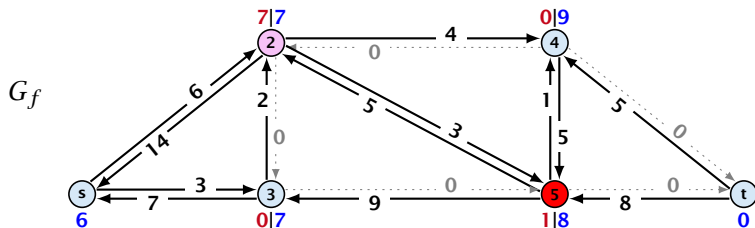
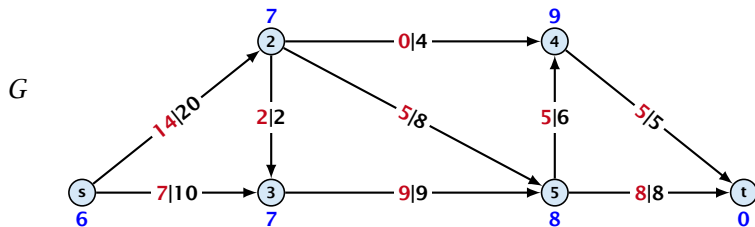
$G_f$



# Preflow Push Algorithm



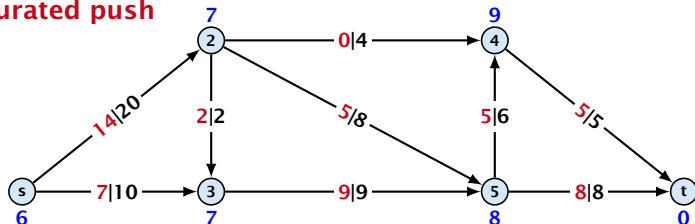
# Preflow Push Algorithm



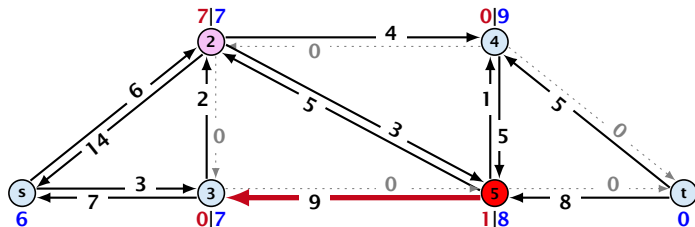
# Preflow Push Algorithm

non-saturated push

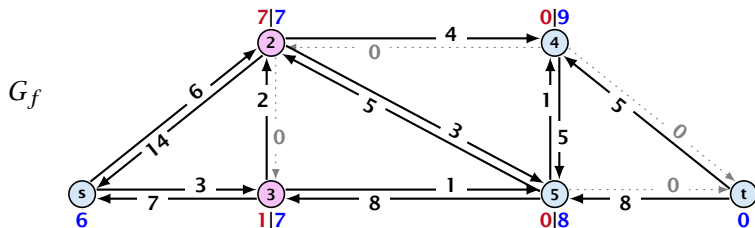
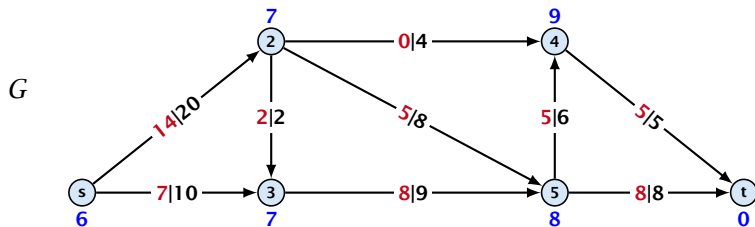
$G$



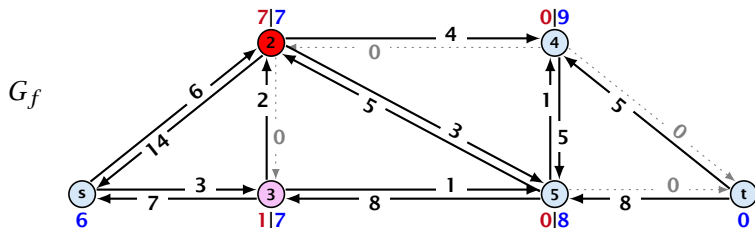
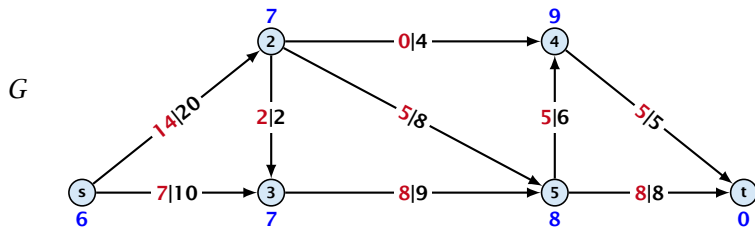
$G_f$



# Preflow Push Algorithm



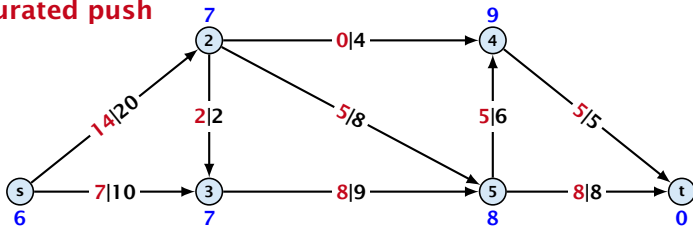
# Preflow Push Algorithm



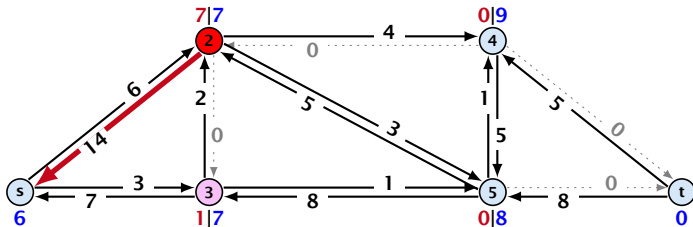
# Preflow Push Algorithm

non-saturated push

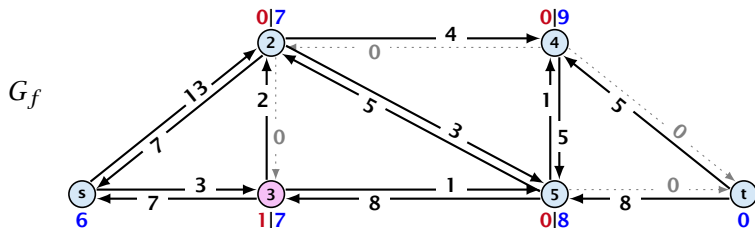
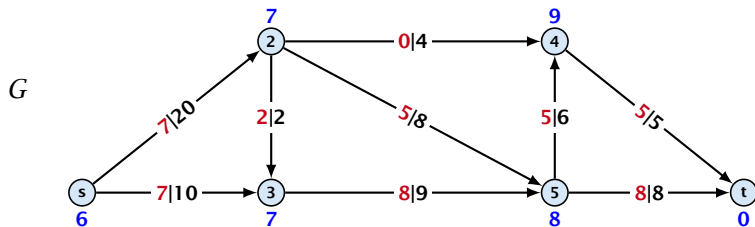
$G$



$G_f$

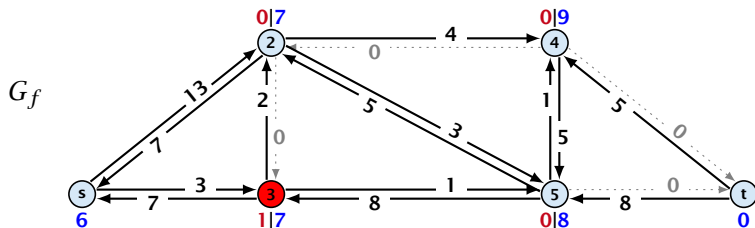
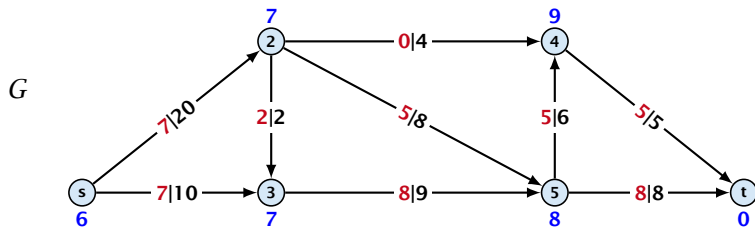


# Preflow Push Algorithm





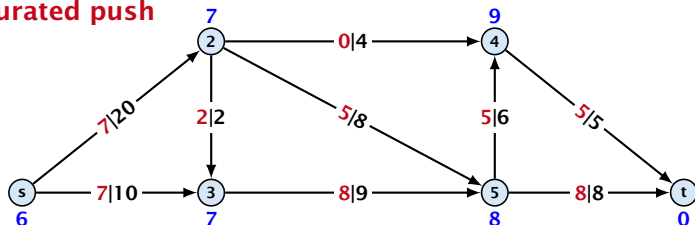
# Preflow Push Algorithm



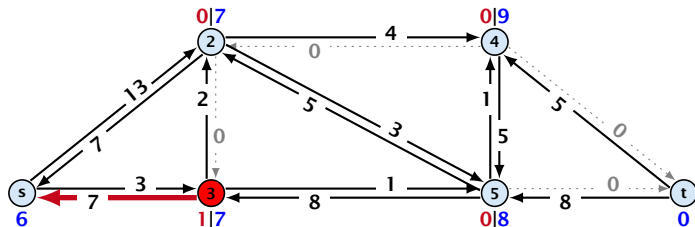
# Preflow Push Algorithm

## non-saturated push

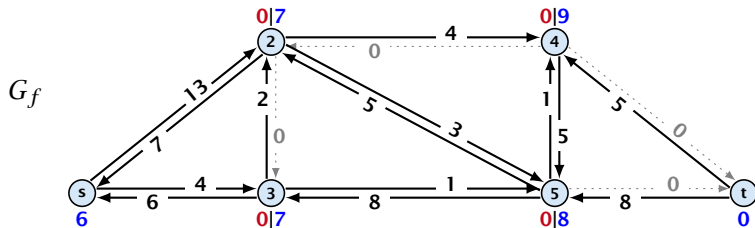
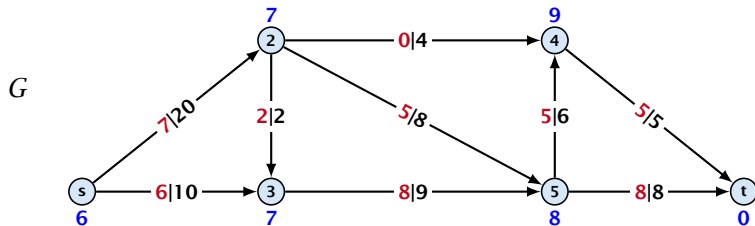
$G$



$G_f$



# Preflow Push Algorithm



# Analysis

## Lemma 5

*An active node has a path to  $s$  in the residual graph.*

# Analysis

## Lemma 5

*An active node has a path to  $s$  in the residual graph.*

### Proof.

- ▶ Let  $A$  denote the set of nodes that can reach  $s$ , and let  $B$  denote the remaining nodes. Note that  $s \in A$ .

# Analysis

## Lemma 5

*An active node has a path to  $s$  in the residual graph.*

### Proof.

- ▶ Let  $A$  denote the set of nodes that can reach  $s$ , and let  $B$  denote the remaining nodes. Note that  $s \in A$ .
- ▶ In the following we show that a node  $b \in B$  has excess flow  $f(b) = 0$  which gives the lemma.

# Analysis

## Lemma 5

*An active node has a path to  $s$  in the residual graph.*

### Proof.

- ▶ Let  $A$  denote the set of nodes that can reach  $s$ , and let  $B$  denote the remaining nodes. Note that  $s \in A$ .
- ▶ In the following we show that a node  $b \in B$  has excess flow  $f(b) = 0$  which gives the lemma.
- ▶ In the residual graph there are no edges into  $A$ , and, hence, no edges leaving  $A$ /entering  $B$  can carry any flow.

# Analysis

## Lemma 5

*An active node has a path to  $s$  in the residual graph.*

### Proof.

- ▶ Let  $A$  denote the set of nodes that can reach  $s$ , and let  $B$  denote the remaining nodes. Note that  $s \in A$ .
- ▶ In the following we show that a node  $b \in B$  has excess flow  $f(b) = 0$  which gives the lemma.
- ▶ In the residual graph there are no edges into  $A$ , and, hence, no edges leaving  $A$ /entering  $B$  can carry any flow.
- ▶ Let  $f(B) = \sum_{v \in B} f(v)$  be the excess flow of all nodes in  $B$ .



Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$f(B)$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$f(B) = \sum_{b \in B} f(b)$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \sum_{b \in B} \sum_{v \in A} f(v, b) - \sum_{b \in B} \sum_{v \in A} f(b, v) + \sum_{b \in B} \sum_{v \in B} f(v, b) - \sum_{b \in B} \sum_{v \in B} f(b, v) \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \sum_{b \in B} \sum_{v \in A} f(v, b) - \sum_{b \in B} \sum_{v \in A} f(b, v) + \sum_{b \in B} \sum_{v \in B} f(v, b) - \sum_{b \in B} \sum_{v \in B} f(b, v) \\ &= 0 \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \sum_{b \in B} \sum_{v \in A} f(v, b) - \sum_{b \in B} \sum_{v \in A} f(b, v) \end{aligned}$$



Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \sum_{b \in B} \sum_{v \in A} \boxed{f(v, b)} - \sum_{b \in B} \sum_{v \in A} f(b, v) \\ &\quad \quad \quad = \mathbf{0} \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \sum_{b \in B} \sum_{v \in A} f(v, b) - \sum_{b \in B} \sum_{v \in A} f(b, v) \\ &= \mathbf{0} \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \sum_{b \in B} \sum_{v \in A} f(b, v) \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \sum_{b \in B} \sum_{v \in A} f(b, v) \geq 0 \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \qquad \qquad \qquad - \sum_{b \in B} \sum_{v \in A} f(b, v) \\ &\leq 0 \end{aligned}$$

Let  $f : E \rightarrow \mathbb{R}_0^+$  be a preflow. We introduce the notation

$$f(x, y) = \begin{cases} 0 & (x, y) \notin E \\ f((x, y)) & (x, y) \in E \end{cases}$$

We have

$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left( \sum_{v \in V} f(v, b) - \sum_{v \in V} f(b, v) \right) \\ &= \sum_{b \in B} \left( \sum_{v \in A} f(v, b) + \sum_{v \in B} f(v, b) - \sum_{v \in A} f(b, v) - \sum_{v \in B} f(b, v) \right) \\ &= \qquad \qquad \qquad - \sum_{b \in B} \sum_{v \in A} f(b, v) \\ &\leq 0 \end{aligned}$$

Hence, the excess flow  $f(b)$  must be 0 for every node  $b \in B$ .

# Analysis

## Lemma 6

*The label of a node cannot become larger than  $2n - 1$ .*

# Analysis

## Lemma 6

*The label of a node cannot become larger than  $2n - 1$ .*

### Proof.

- ▶ When increasing the label at a node  $u$  there exists a path from  $u$  to  $s$  of length at most  $n - 1$ . Along each edge of the path the height/label can at most drop by  $1$ , and the label of the source is  $n$ .



# Analysis

## Lemma 6

*The label of a node cannot become larger than  $2n - 1$ .*

### Proof.

- ▶ When increasing the label at a node  $u$  there exists a path from  $u$  to  $s$  of length at most  $n - 1$ . Along each edge of the path the height/label can at most drop by  $1$ , and the label of the source is  $n$ .

## Lemma 7

*There are only  $\mathcal{O}(n^2)$  relabel operations.*

# Analysis

## Lemma 8

The number of *saturating pushes* performed is at most  $\mathcal{O}(mn)$ .

# Analysis

## Lemma 8

The number of *saturating pushes* performed is at most  $\mathcal{O}(mn)$ .

### Proof.

- ▶ Suppose that we just made a saturating push along  $(u, v)$ .

# Analysis

## Lemma 8

The number of *saturating pushes* performed is at most  $\mathcal{O}(mn)$ .

### Proof.

- ▶ Suppose that we just made a saturating push along  $(u, v)$ .
- ▶ Hence, the edge  $(u, v)$  is deleted from the residual graph.

# Analysis

## Lemma 8

The number of *saturating pushes* performed is at most  $\mathcal{O}(mn)$ .

### Proof.

- ▶ Suppose that we just made a saturating push along  $(u, v)$ .
- ▶ Hence, the edge  $(u, v)$  is deleted from the residual graph.
- ▶ For the edge to appear again, a push from  $v$  to  $u$  is required.

# Analysis

## Lemma 8

The number of  *saturating pushes*  performed is at most  $\mathcal{O}(mn)$ .

### Proof.

- ▶ Suppose that we just made a saturating push along  $(u, v)$ .
- ▶ Hence, the edge  $(u, v)$  is deleted from the residual graph.
- ▶ For the edge to appear again, a push from  $v$  to  $u$  is required.
- ▶ Currently,  $\ell(u) = \ell(v) + 1$ , as we only make pushes along admissible edges.

# Analysis

## Lemma 8

The number of *saturating pushes* performed is at most  $\mathcal{O}(mn)$ .

### Proof.

- ▶ Suppose that we just made a saturating push along  $(u, v)$ .
- ▶ Hence, the edge  $(u, v)$  is deleted from the residual graph.
- ▶ For the edge to appear again, a push from  $v$  to  $u$  is required.
- ▶ Currently,  $\ell(u) = \ell(v) + 1$ , as we only make pushes along admissible edges.
- ▶ For a push from  $v$  to  $u$  the edge  $(v, u)$  must become admissible. The label of  $v$  must increase by at least 2.

# Analysis

## Lemma 8

The number of *saturating pushes* performed is at most  $\mathcal{O}(mn)$ .

### Proof.

- ▶ Suppose that we just made a saturating push along  $(u, v)$ .
- ▶ Hence, the edge  $(u, v)$  is deleted from the residual graph.
- ▶ For the edge to appear again, a push from  $v$  to  $u$  is required.
- ▶ Currently,  $\ell(u) = \ell(v) + 1$ , as we only make pushes along admissible edges.
- ▶ For a push from  $v$  to  $u$  the edge  $(v, u)$  must become admissible. The label of  $v$  must increase by at least 2.
- ▶ Since the label of  $v$  is at most  $2n - 1$ , there are at most  $n$  pushes along  $(u, v)$ .



## Lemma 9

The number of *non-saturating pushes* performed is at most  $\mathcal{O}(n^2m)$ .

## Lemma 9

The number of *non-saturating pushes* performed is at most  $\mathcal{O}(n^2m)$ .

### Proof.

- ▶ Define a potential function  $\Phi(f) = \sum_{\text{active nodes } v} \ell(v)$

## Lemma 9

The number of *non-saturating pushes* performed is at most  $\mathcal{O}(n^2m)$ .

### Proof.

- ▶ Define a potential function  $\Phi(f) = \sum_{\text{active nodes } v} \ell(v)$
- ▶ A saturating push increases  $\Phi$  by  $\leq 2n$  (when the target node becomes active it may contribute at most  $2n$  to the sum).

## Lemma 9

The number of *non-saturating pushes* performed is at most  $\mathcal{O}(n^2m)$ .

### Proof.

- ▶ Define a potential function  $\Phi(f) = \sum_{\text{active nodes } v} \ell(v)$
- ▶ A saturating push increases  $\Phi$  by  $\leq 2n$  (when the target node becomes active it may contribute at most  $2n$  to the sum).
- ▶ A relabel increases  $\Phi$  by at most 1.

## Lemma 9

The number of *non-saturating pushes* performed is at most  $\mathcal{O}(n^2m)$ .

### Proof.

- ▶ Define a potential function  $\Phi(f) = \sum_{\text{active nodes } v} \ell(v)$
- ▶ A saturating push increases  $\Phi$  by  $\leq 2n$  (when the target node becomes active it may contribute at most  $2n$  to the sum).
- ▶ A relabel increases  $\Phi$  by at most  $1$ .
- ▶ A non-saturating push decreases  $\Phi$  by at least  $1$  as the node that is pushed from becomes inactive and has a label that is strictly larger than the target.

## Lemma 9

The number of *non-saturating pushes* performed is at most  $\mathcal{O}(n^2m)$ .

### Proof.

- ▶ Define a potential function  $\Phi(f) = \sum_{\text{active nodes } v} \ell(v)$
- ▶ A saturating push increases  $\Phi$  by  $\leq 2n$  (when the target node becomes active it may contribute at most  $2n$  to the sum).
- ▶ A relabel increases  $\Phi$  by at most  $1$ .
- ▶ A non-saturating push decreases  $\Phi$  by at least  $1$  as the node that is pushed from becomes inactive and has a label that is strictly larger than the target.
- ▶ Hence,

$$\begin{aligned} \# \text{non-saturating\_pushes} &\leq \# \text{relabels} + 2n \cdot \# \text{saturating\_pushes} \\ &\leq \mathcal{O}(n^2m) . \end{aligned}$$

## Theorem 10

*There is an implementation of the generic push relabel algorithm with running time  $\mathcal{O}(n^2m)$ .*

# Analysis

## Proof:

For every node maintain a list of admissible edges starting at that node. Further maintain a list of active nodes.

A push along an edge  $(u, v)$  can be performed in constant time

- check whether edge  $(u, v)$  needs to be added to  $E(u)$
- check whether  $(u, v)$  needs to be deleted (aftering push)
- check whether  $v$  becomes inactive and has to be deleted from the set of active nodes

A relabel at a node  $u$  can be performed in time  $\mathcal{O}(n)$

- check for all outgoing edges if they become inadmissible
- check for all incoming edges if they become inadmissible



# Analysis

## Proof:

For every node maintain a list of admissible edges starting at that node. Further maintain a list of active nodes.

A push along an edge  $(u, v)$  can be performed in constant time  
check whether  $v$  is active (if not, add  $v$  to the set of active nodes)  
check whether  $u$  needs to be relabel (returning push)  
check whether  $u$  becomes inactive and has to be deleted  
from the set of active nodes

A relabel at a node  $u$  can be performed in time  $\mathcal{O}(n)$   
check for all outgoing edges if they become admissible  
check for all incoming edges if they become inadmissible

# Analysis

## Proof:

For every node maintain a list of admissible edges starting at that node. Further maintain a list of active nodes.

A push along an edge  $(u, v)$  can be performed in constant time

- ▶ check whether edge  $(v, u)$  needs to be added to  $G_f$
- ▶ check whether  $(u, v)$  needs to be deleted (saturating push)
- ▶ check whether  $u$  becomes inactive and has to be deleted from the set of active nodes

A relabel at a node  $u$  can be performed in time  $\mathcal{O}(n)$

# Analysis

## Proof:

For every node maintain a list of admissible edges starting at that node. Further maintain a list of active nodes.

A push along an edge  $(u, v)$  can be performed in constant time

- ▶ check whether edge  $(v, u)$  needs to be added to  $G_f$
- ▶ check whether  $(u, v)$  needs to be deleted (saturating push)
- ▶ check whether  $u$  becomes inactive and has to be deleted from the set of active nodes

A relabel at a node  $u$  can be performed in time  $\mathcal{O}(n)$

# Analysis

## Proof:

For every node maintain a list of admissible edges starting at that node. Further maintain a list of active nodes.

A push along an edge  $(u, v)$  can be performed in constant time

- ▶ check whether edge  $(v, u)$  needs to be added to  $G_f$
- ▶ check whether  $(u, v)$  needs to be deleted (saturating push)
- ▶ check whether  $u$  becomes inactive and has to be deleted from the set of active nodes

A relabel at a node  $u$  can be performed in time  $\mathcal{O}(n)$

# Analysis

## Proof:

For every node maintain a list of admissible edges starting at that node. Further maintain a list of active nodes.

A push along an edge  $(u, v)$  can be performed in constant time

- ▶ check whether edge  $(v, u)$  needs to be added to  $G_f$
- ▶ check whether  $(u, v)$  needs to be deleted (saturating push)
- ▶ check whether  $u$  becomes inactive and has to be deleted from the set of active nodes

A relabel at a node  $u$  can be performed in time  $\mathcal{O}(n)$

- ▶ check for all outgoing edges if they become admissible
- ▶ check for all incoming edges if they become non-admissible

# Analysis

## Proof:

For every node maintain a list of admissible edges starting at that node. Further maintain a list of active nodes.

A push along an edge  $(u, v)$  can be performed in constant time

- ▶ check whether edge  $(v, u)$  needs to be added to  $G_f$
- ▶ check whether  $(u, v)$  needs to be deleted (saturating push)
- ▶ check whether  $u$  becomes inactive and has to be deleted from the set of active nodes

A relabel at a node  $u$  can be performed in time  $\mathcal{O}(n)$

- ▶ check for all outgoing edges if they become admissible
- ▶ check for all incoming edges if they become non-admissible

## Analysis

For special variants of push relabel algorithms we organize the neighbours of a node into a linked list (possible neighbours in the residual graph  $G_f$ ). Then we use the discharge-operation:

### Algorithm 4 discharge( $u$ )

```
1: while  $u$  is active do  
2:    $v \leftarrow u.current\text{-neighbour}$   
3:   if  $v = \text{null}$  then  
4:     relabel( $u$ )  
5:      $u.current\text{-neighbour} \leftarrow u.neighbour\text{-list-head}$   
6:   else  
7:     if  $(u, v)$  admissable then push( $u, v$ )  
8:     else  $u.current\text{-neighbour} \leftarrow v.next\text{-in-list}$ 
```

Note that  $u.current\text{-neighbour}$  is a global variable. It is only changed within the discharge routine, but keeps its value between consecutive calls to discharge.

## Lemma 11

If  $v = \text{null}$  in Line 3, then there is no outgoing admissible edge from  $u$ .

### Proof.

- ▶ While pushing from  $u$  the current-neighbour pointer is only advanced if the current edge is not admissible.
- ▶ The only thing that could make the edge admissible again would be a relabel at  $u$ .
- ▶ If we reach the end of the list ( $v = \text{null}$ ) all edges are not admissible. □

This shows that  $\text{discharge}(u)$  is correct, and that we can perform a relabel in Line 4.