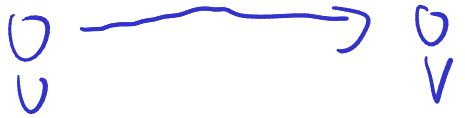
Reminder



- 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 dissappear from the residual graph).
- A deactivating 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 19 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:             push(G, e, f, c)
5:       else
6:             relabel(u)
7: return f
```

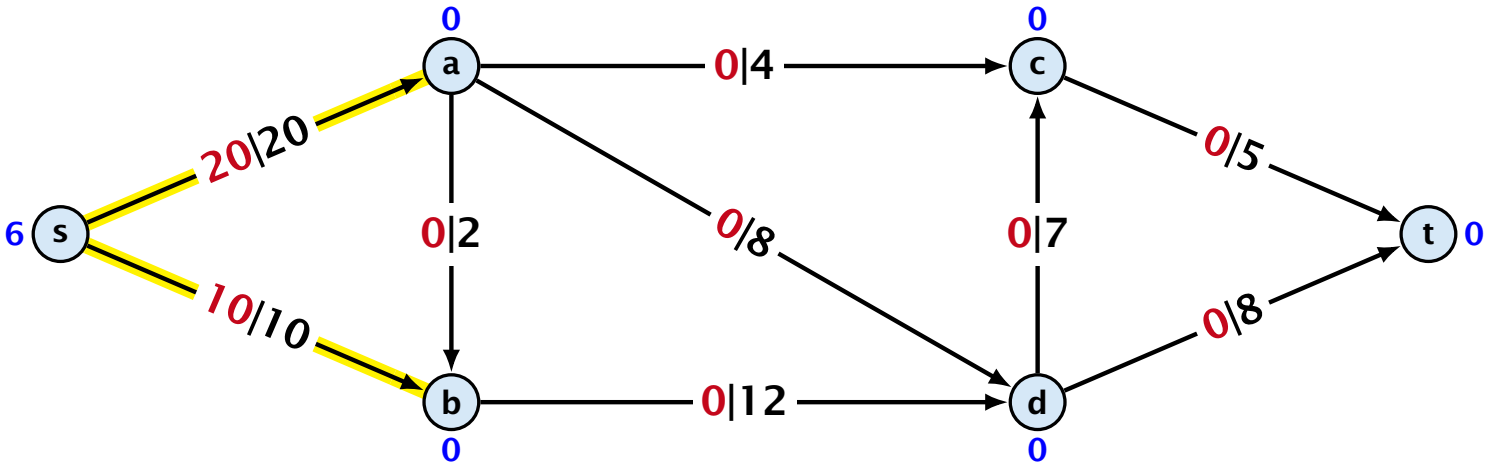
Push Relabel Algorithms

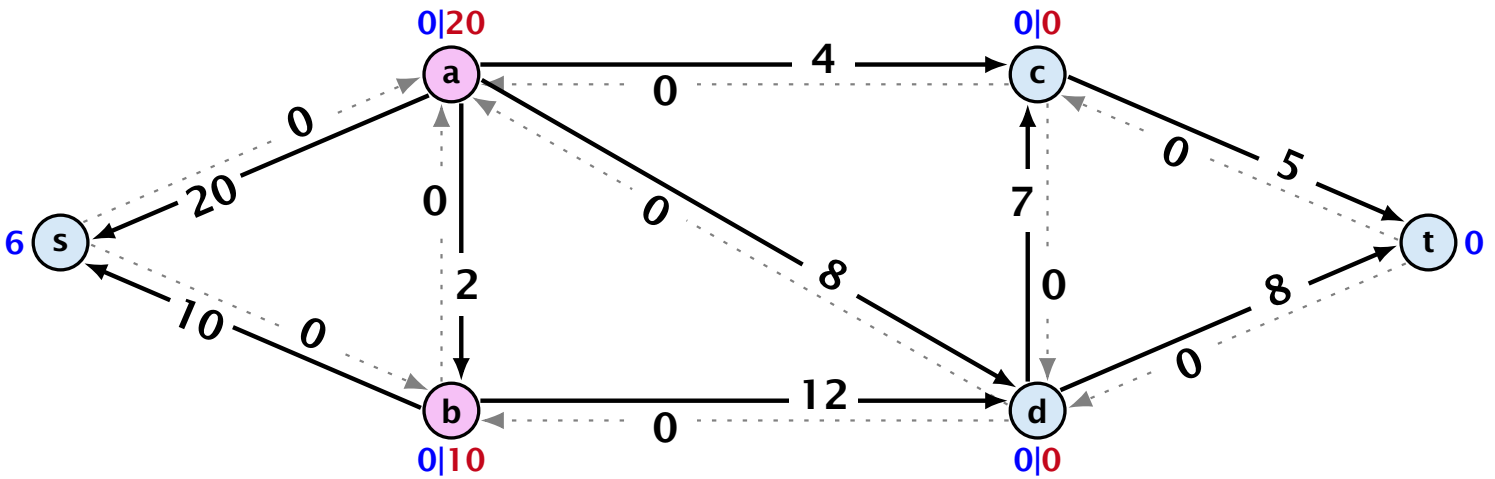
Algorithm 19 maxflow(G, s, t, c)

```
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:             push(G, e, f, c)
5:       else
6:             relabel(u)
7: return f
```

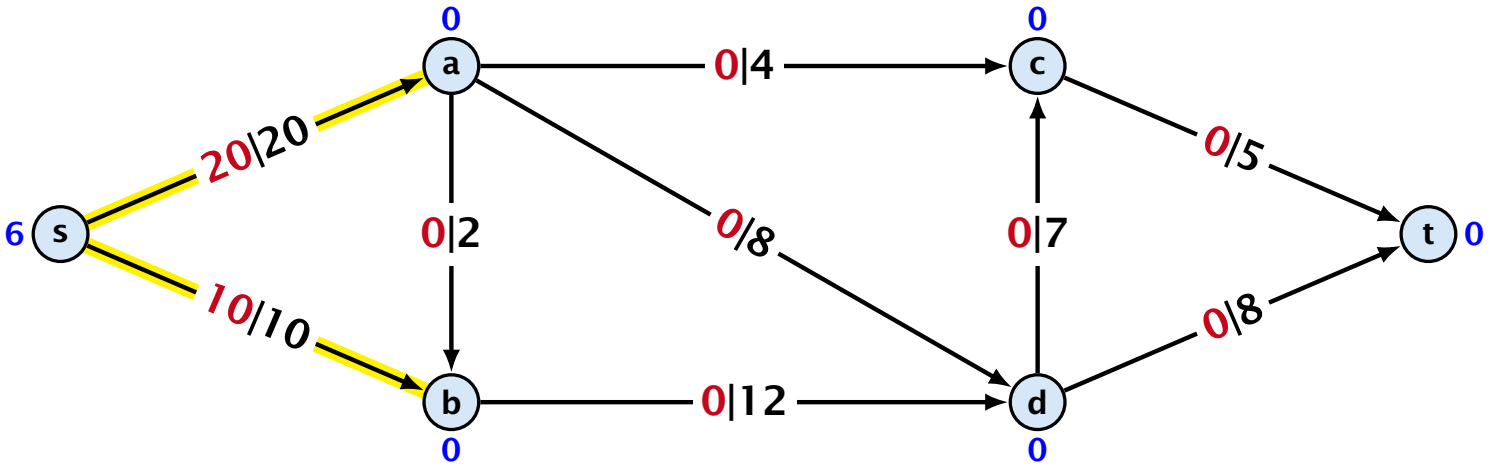
In the following example we always stick to the same active node u until it becomes inactive but this is not required.

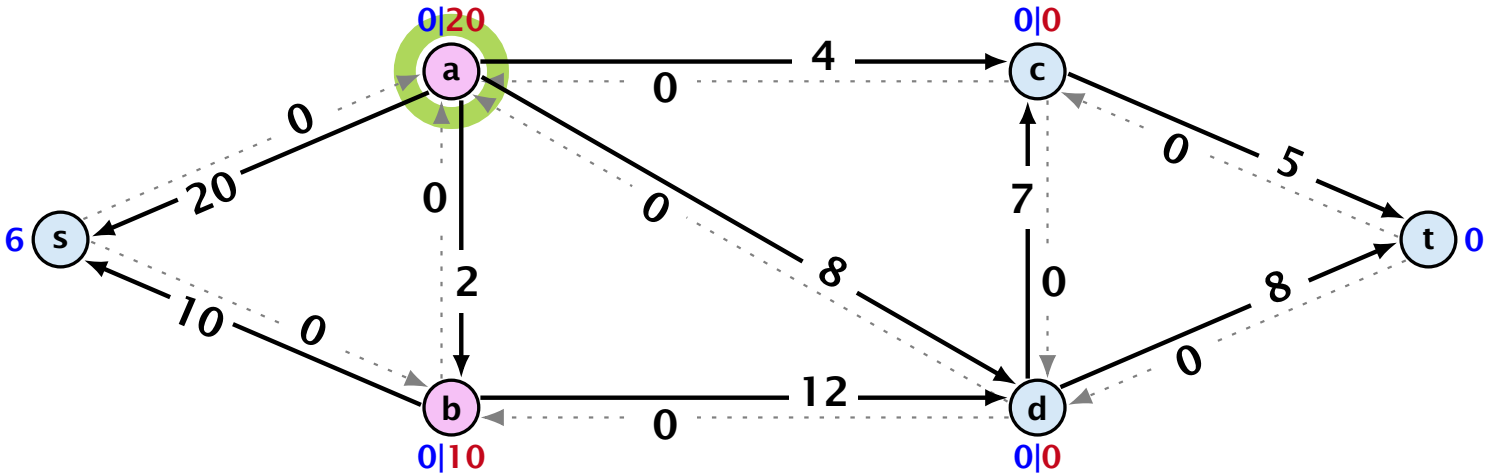
Preflow Push



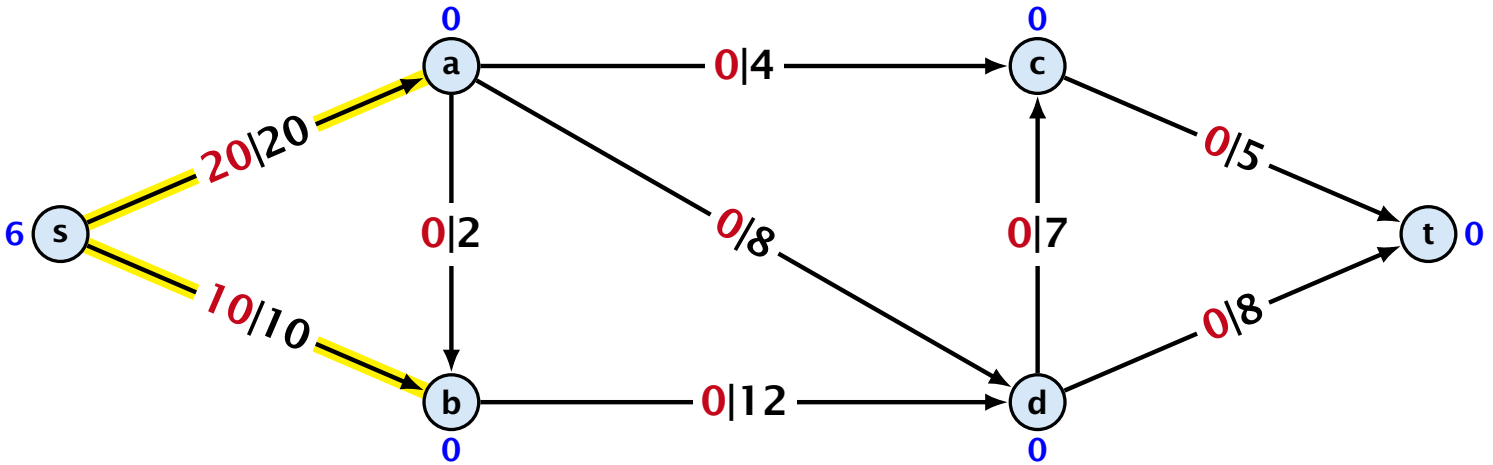


Preflow Push

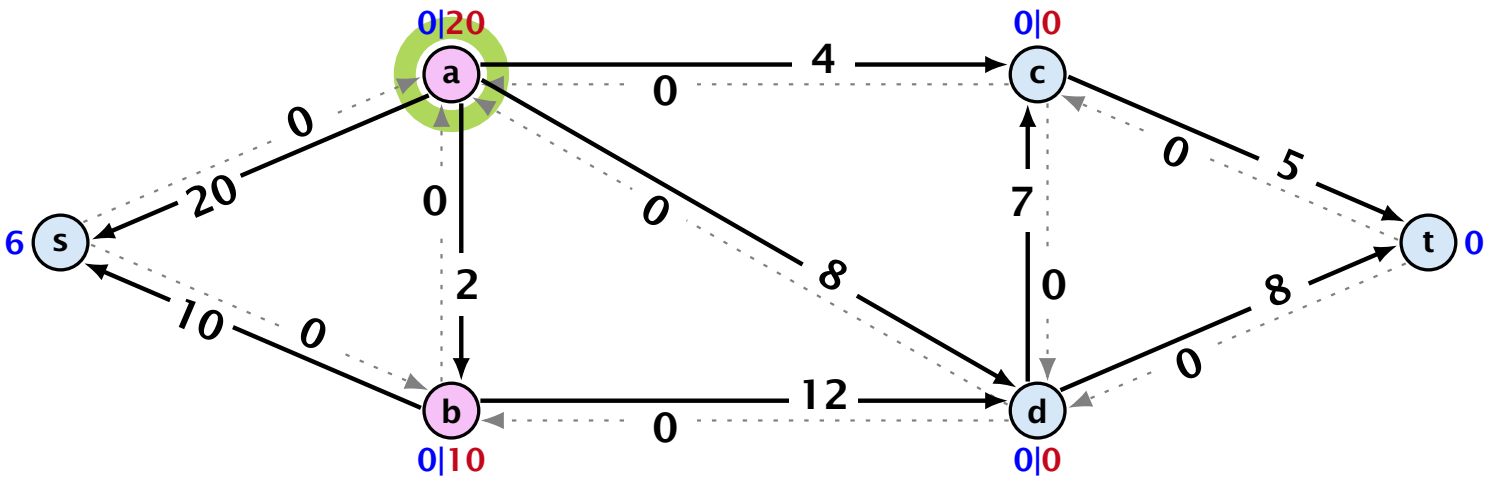




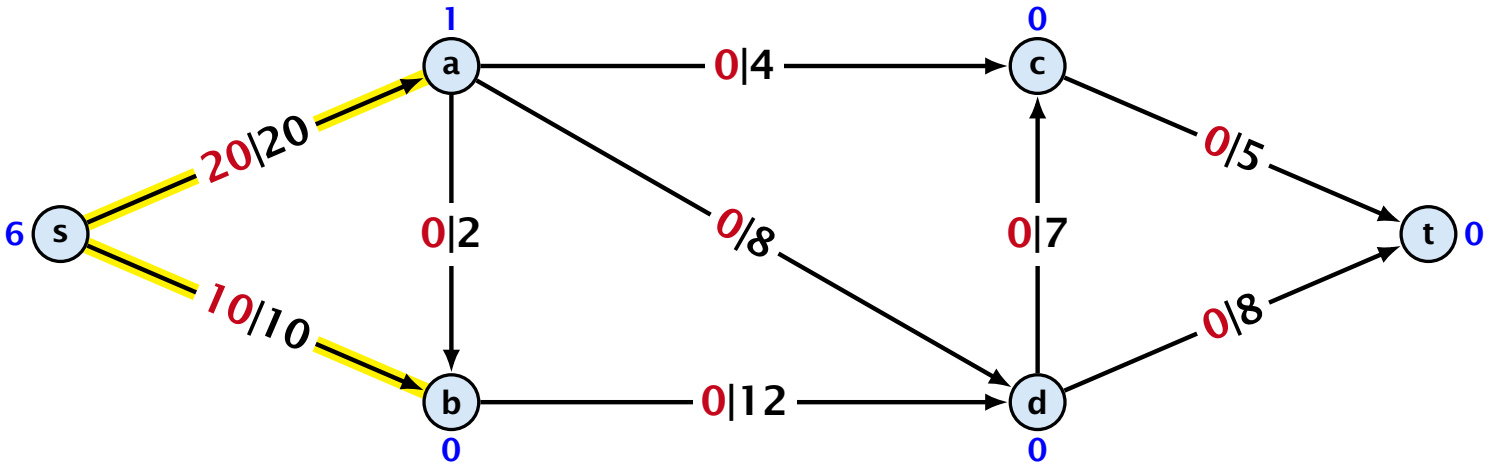
Preflow Push

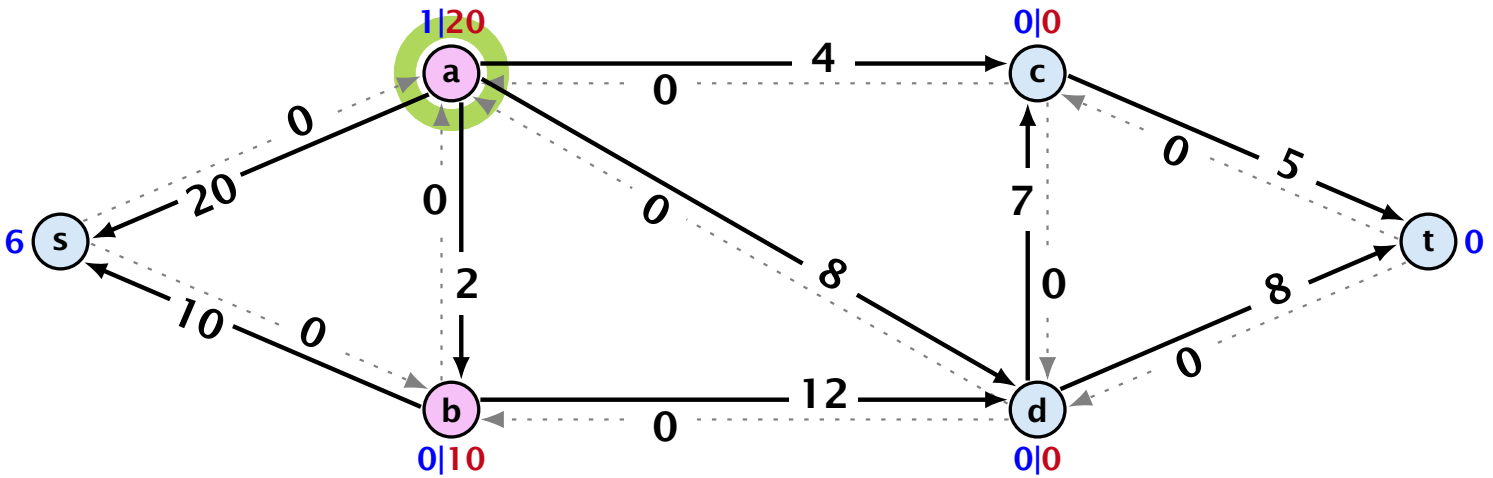


relabel to 1

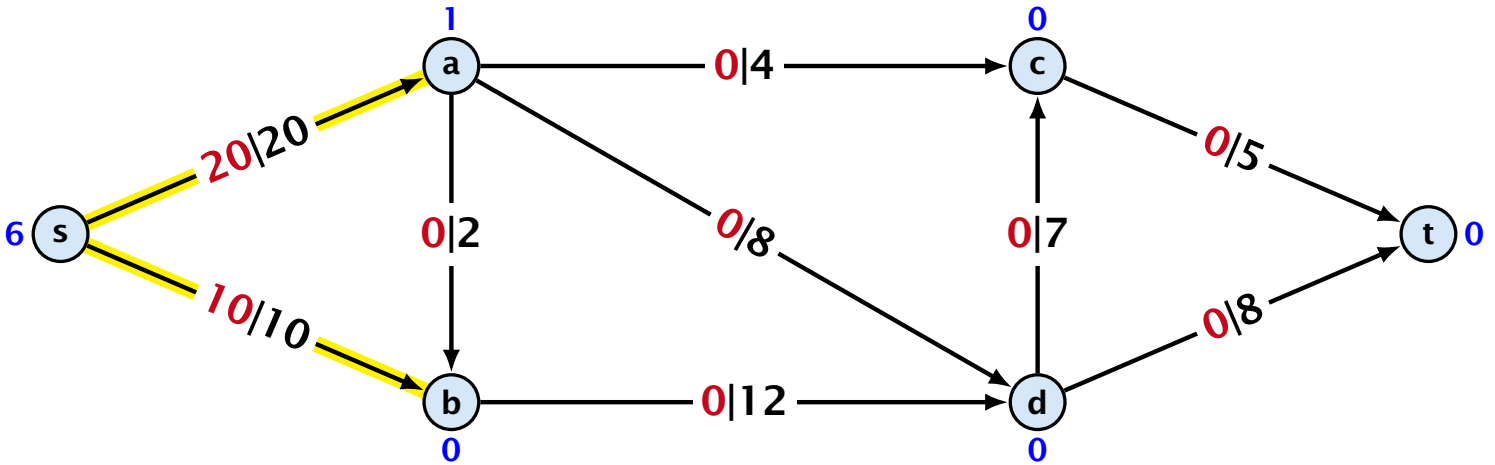


Preflow Push

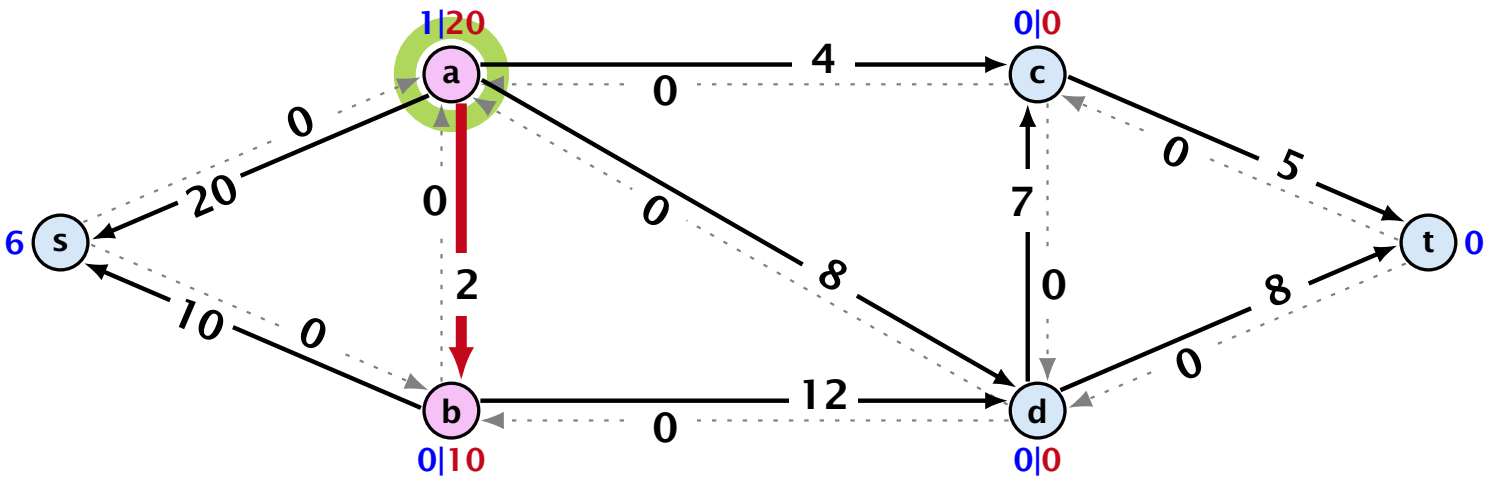




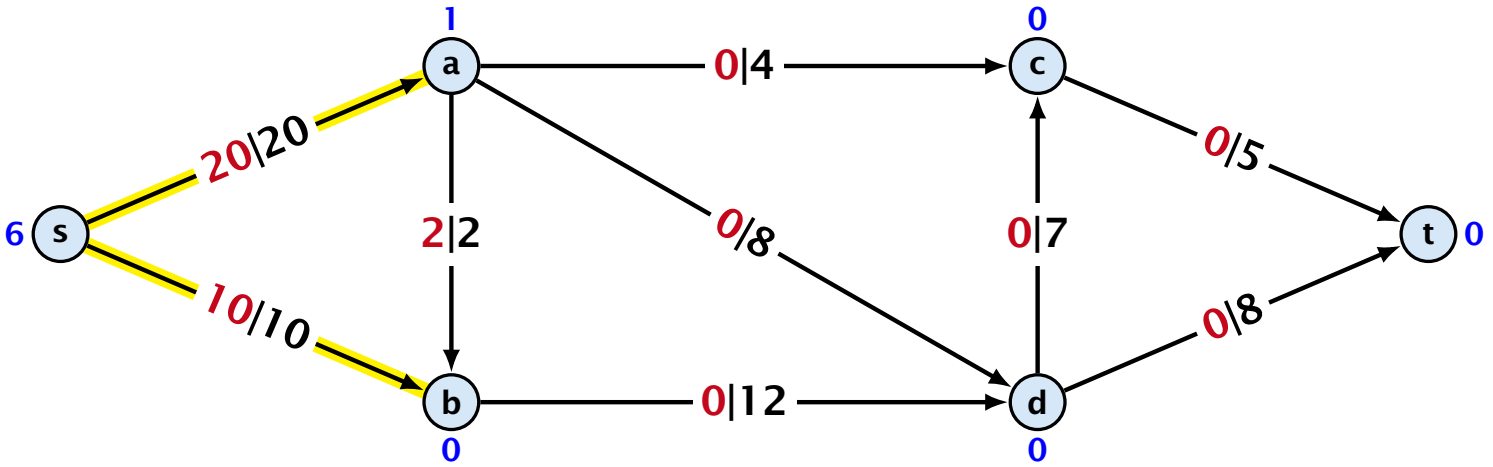
Preflow Push

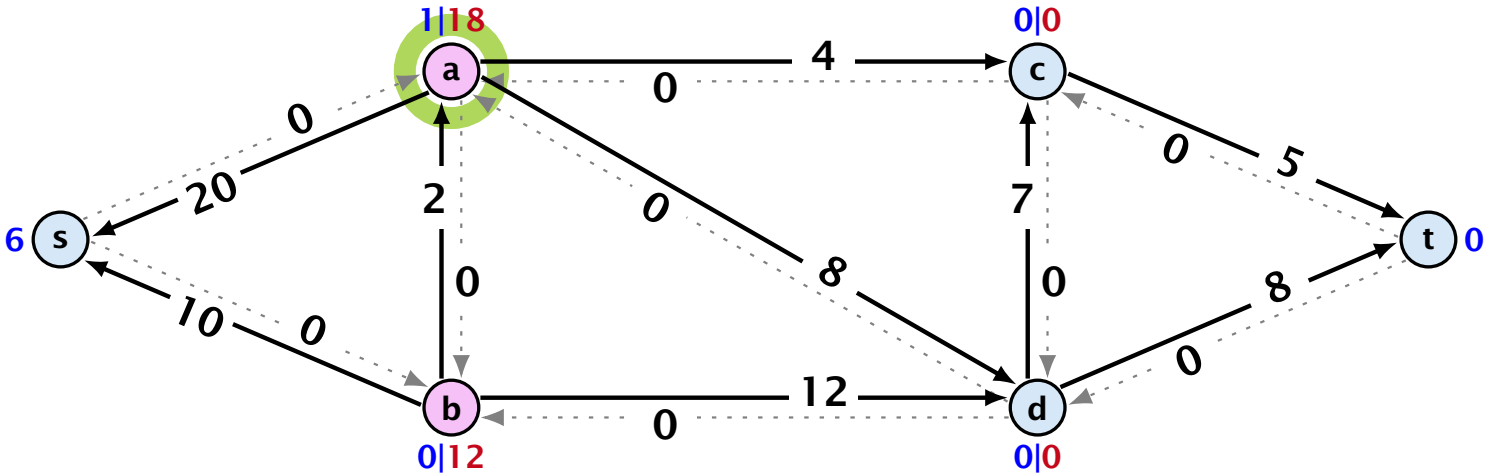


saturating push

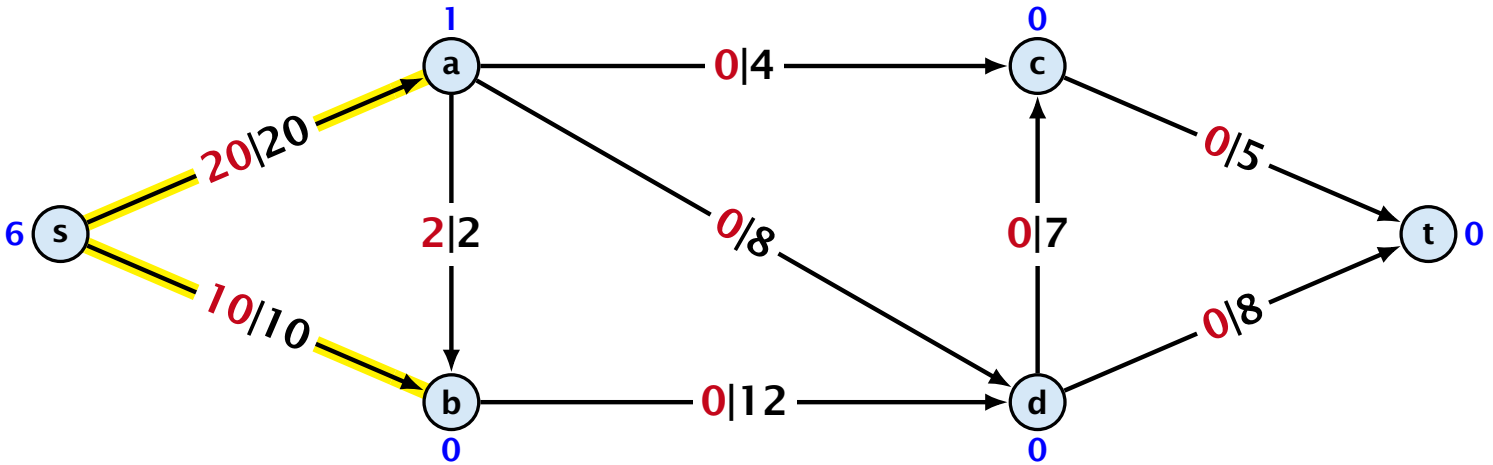


Preflow Push

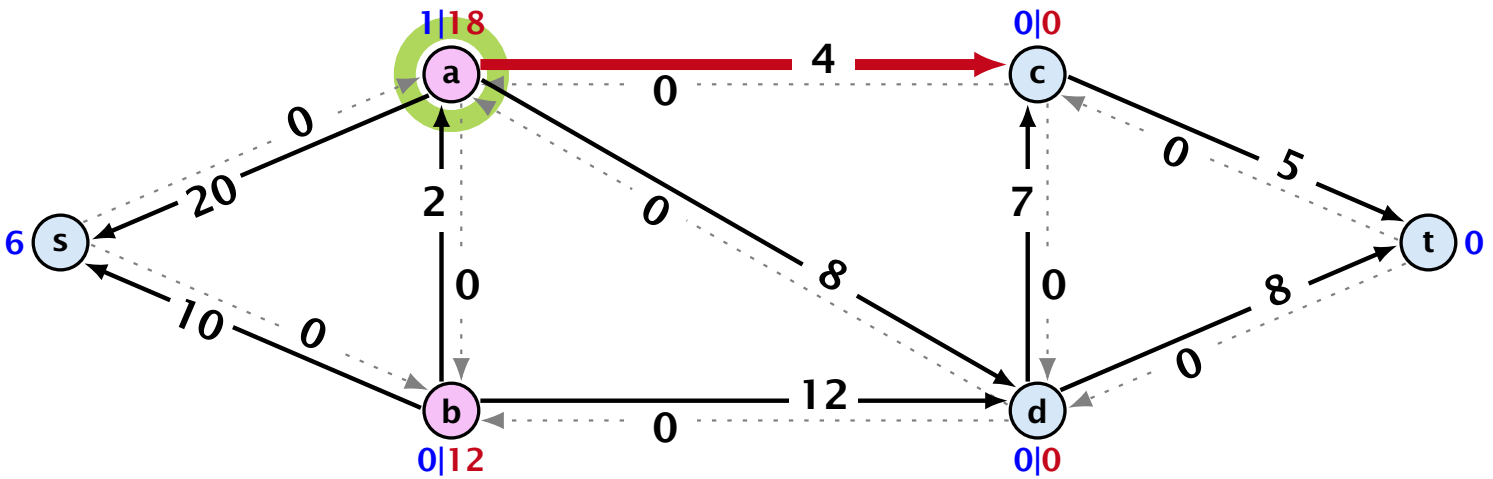




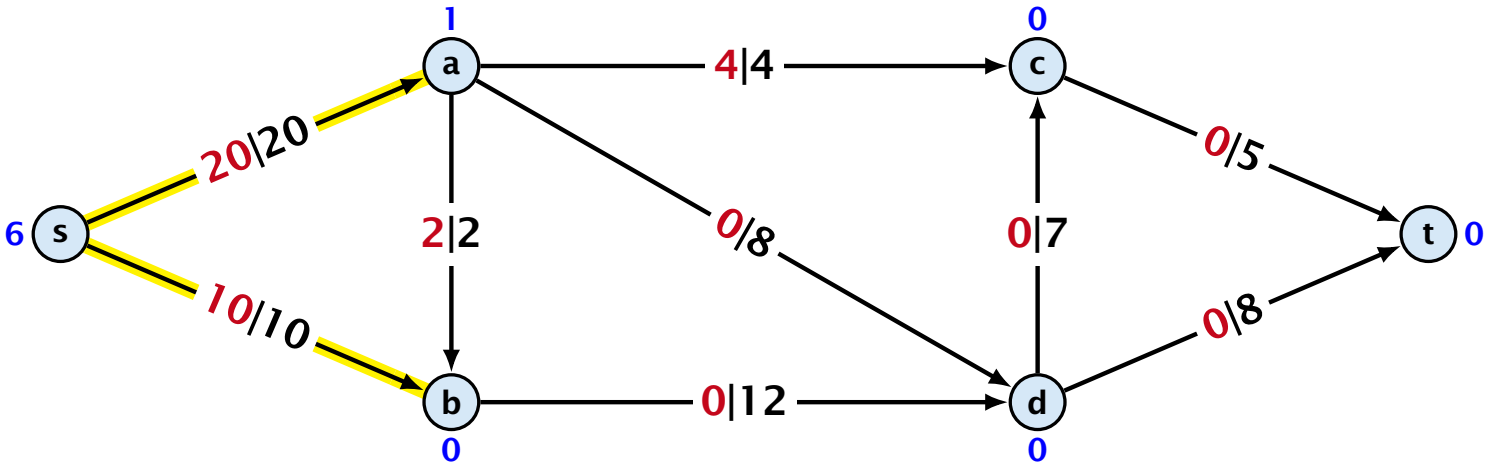
Preflow Push

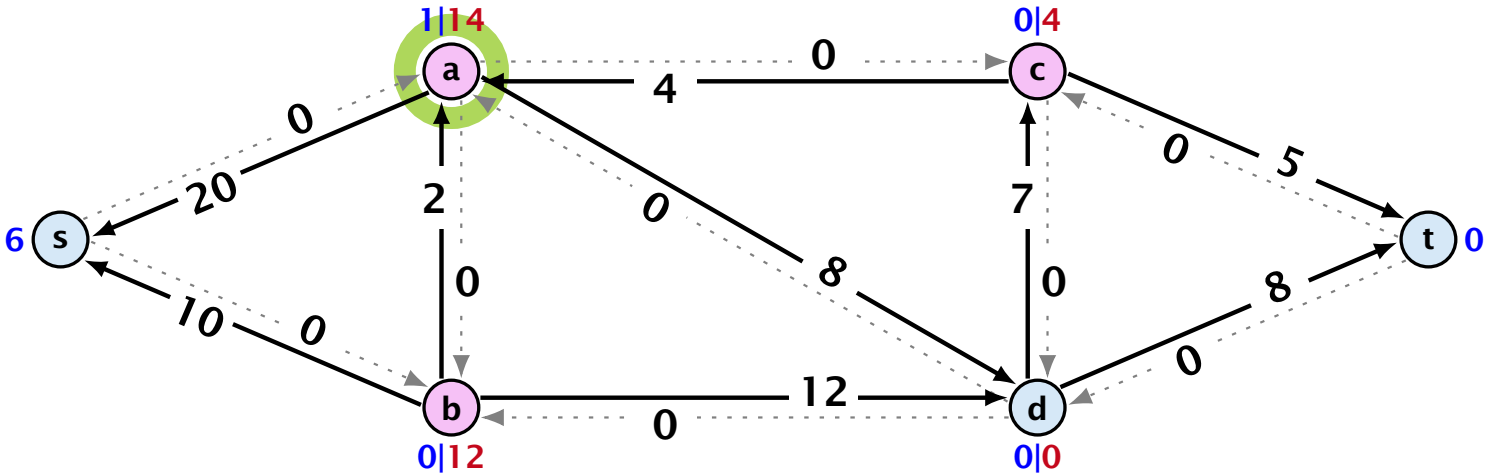


saturating push

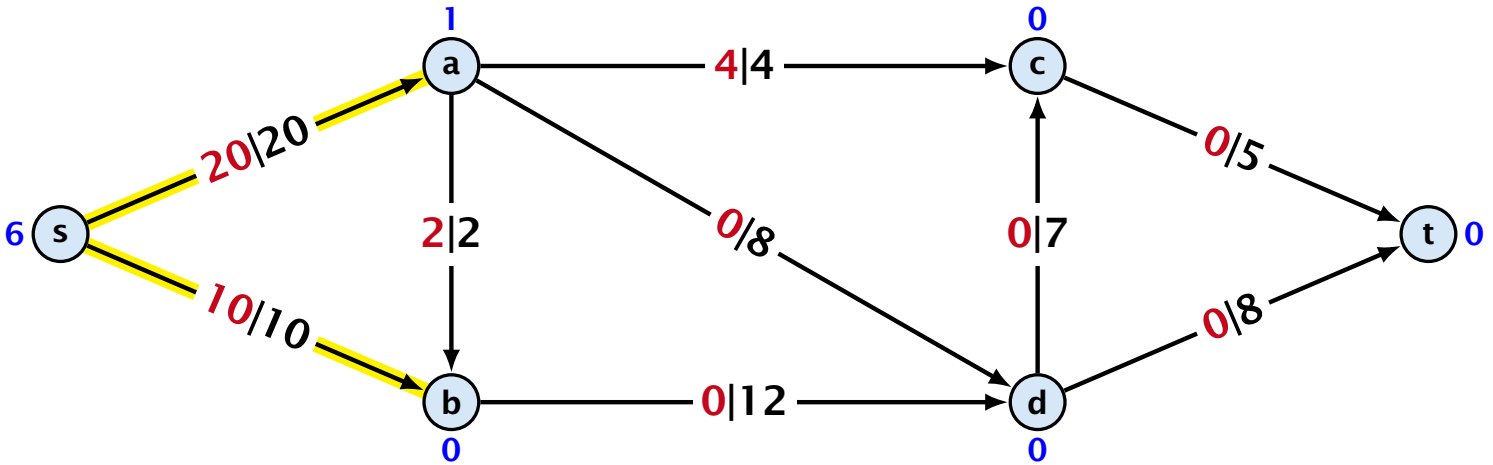


Preflow Push

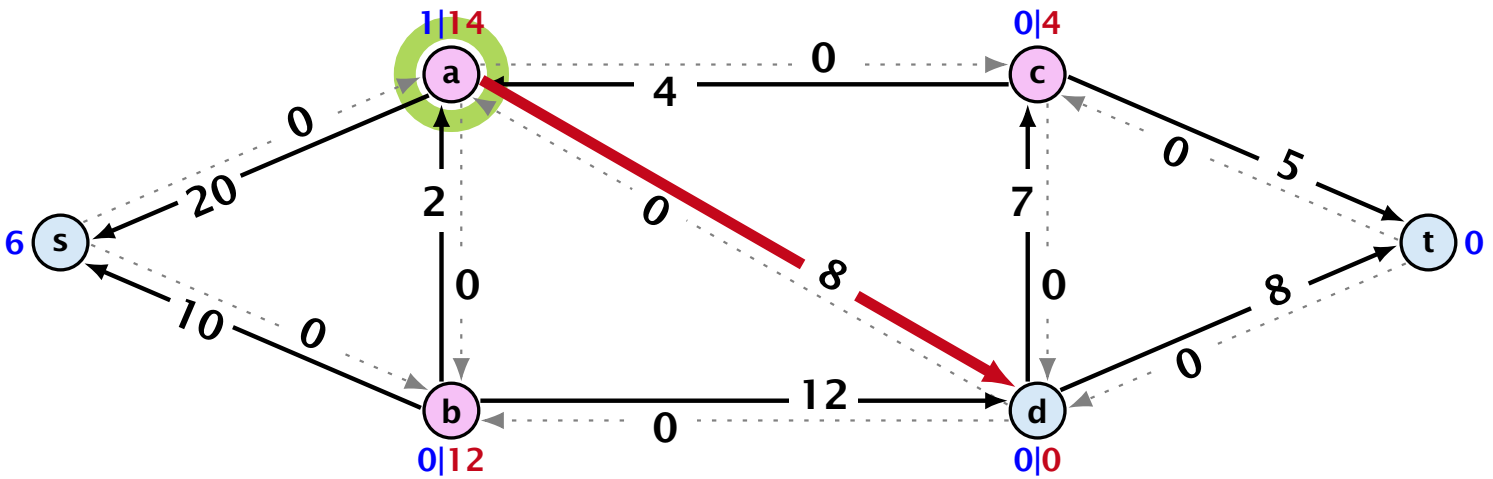




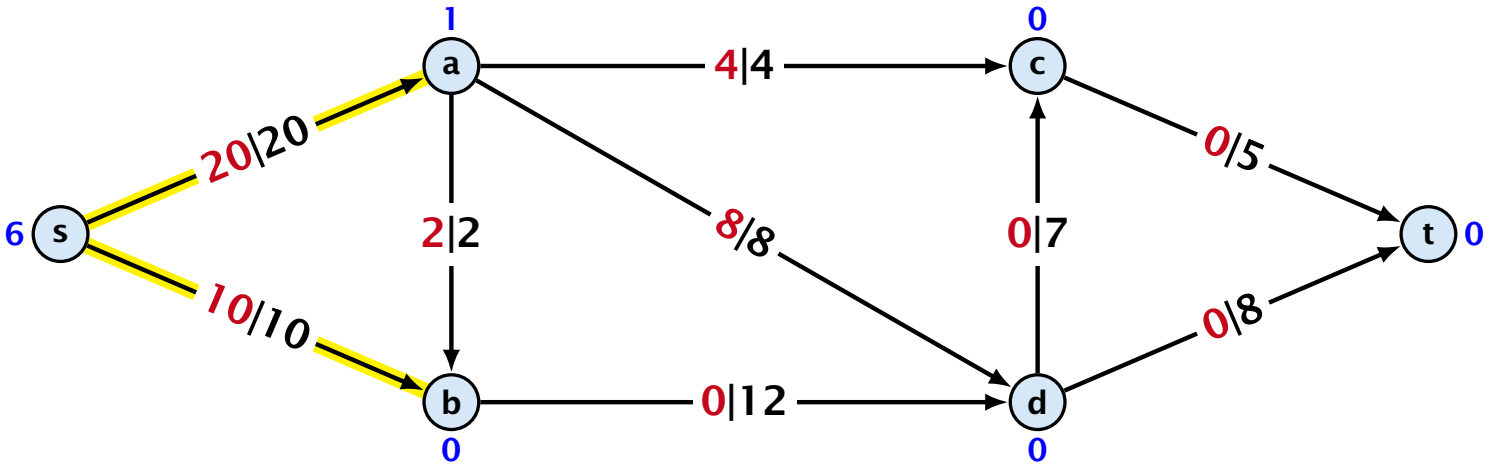
Preflow Push

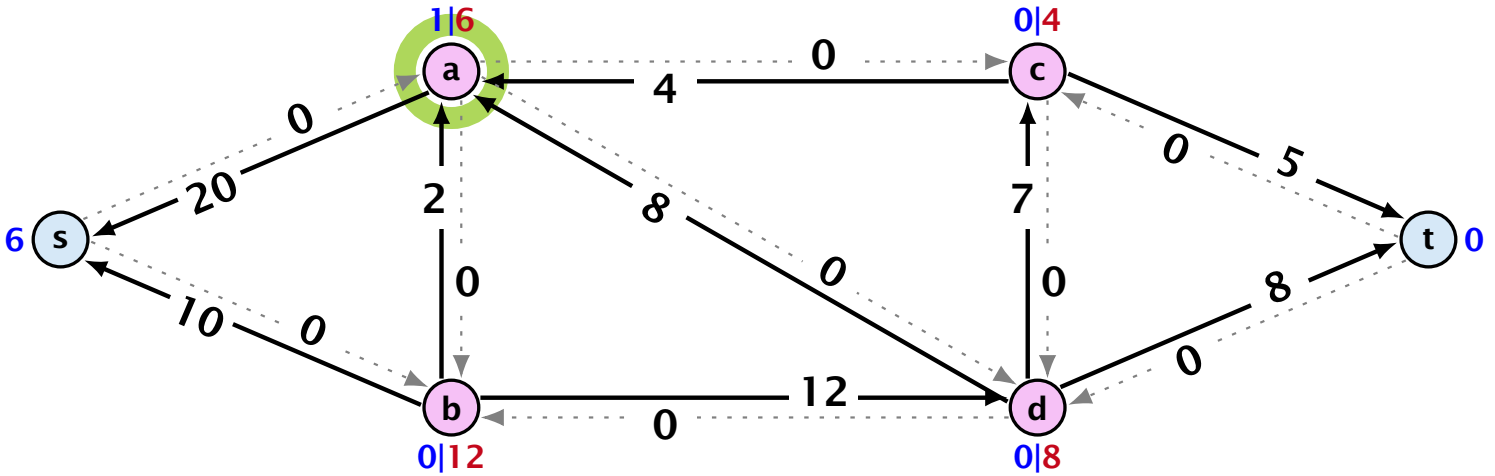


saturating push

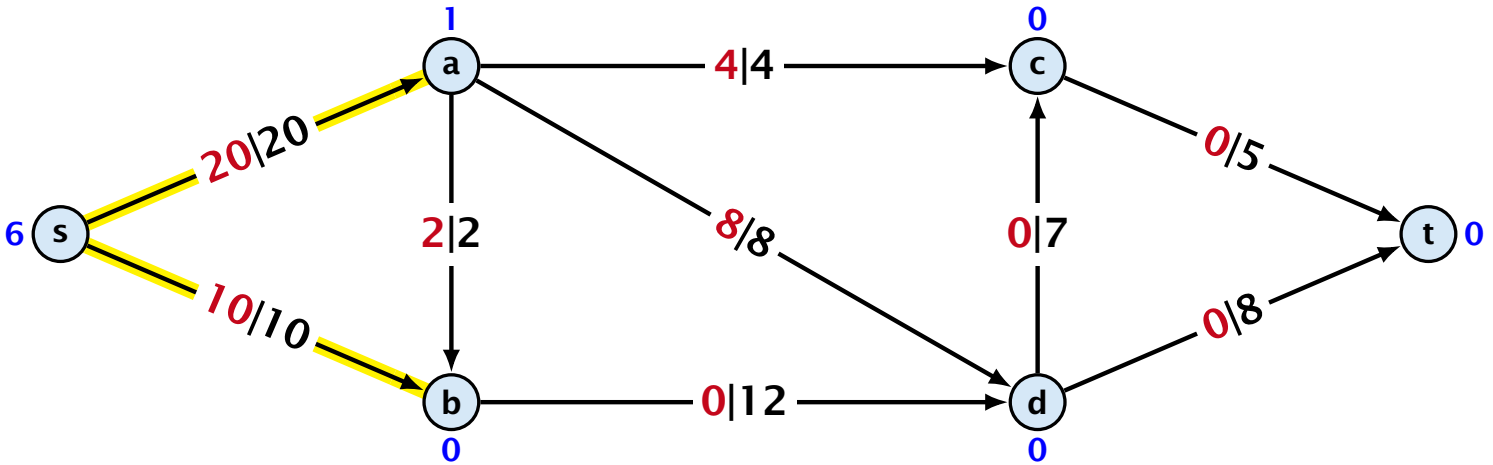


Preflow Push

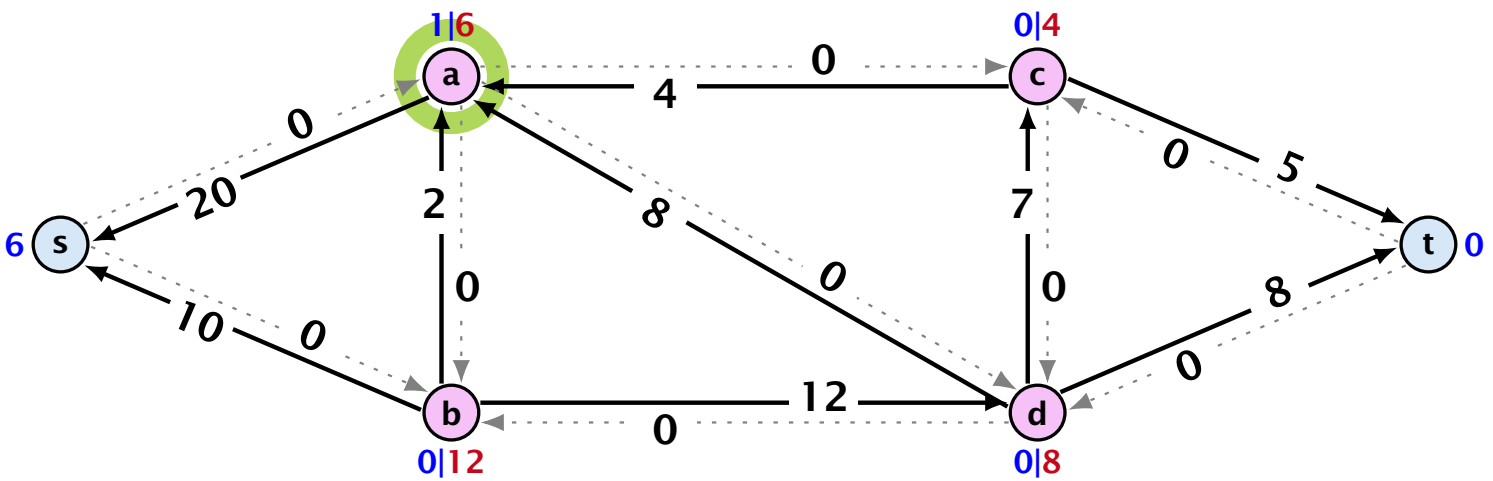




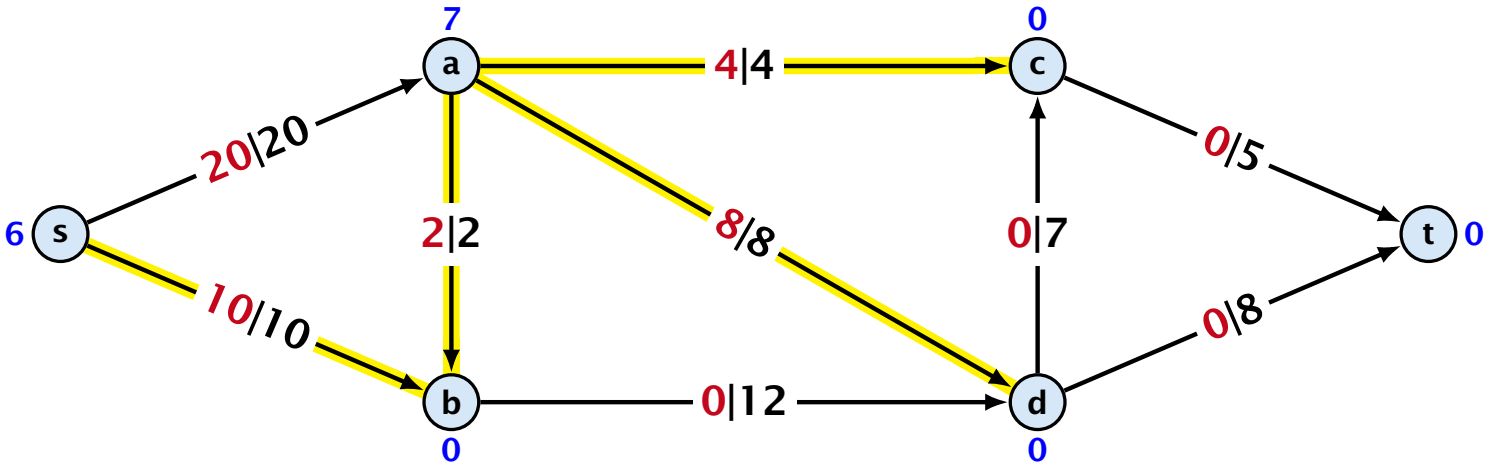
Preflow Push

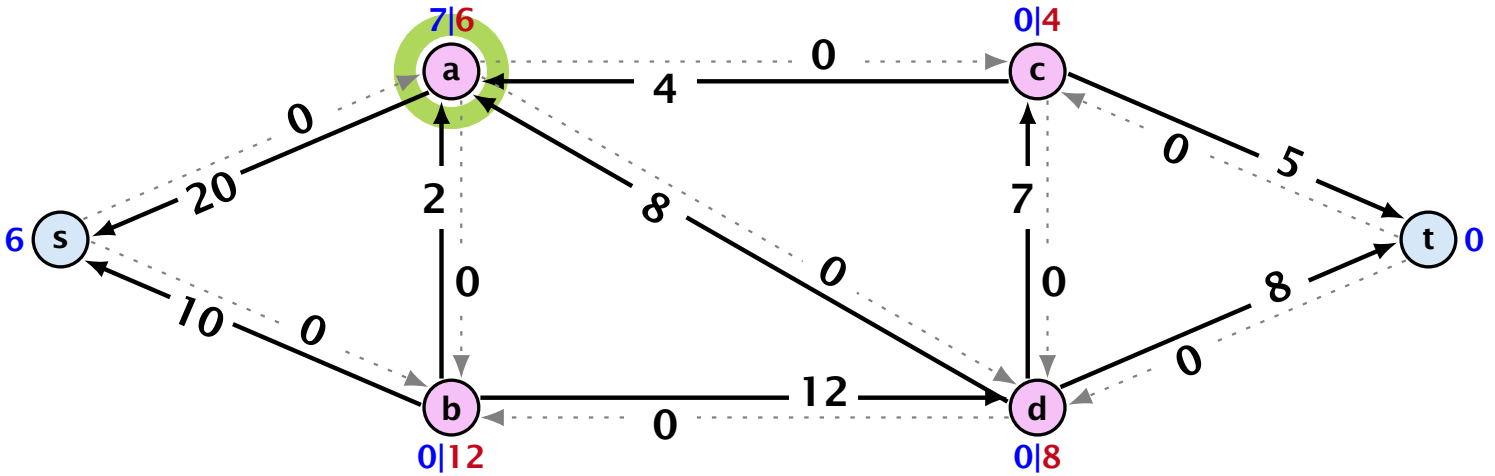


relabel to 7

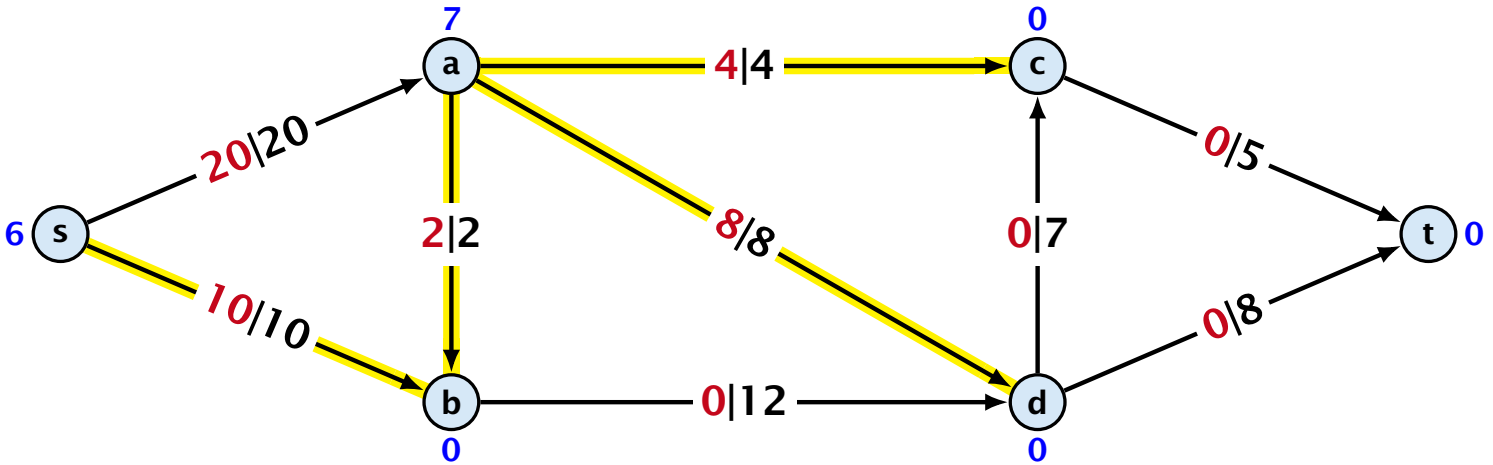


Preflow Push

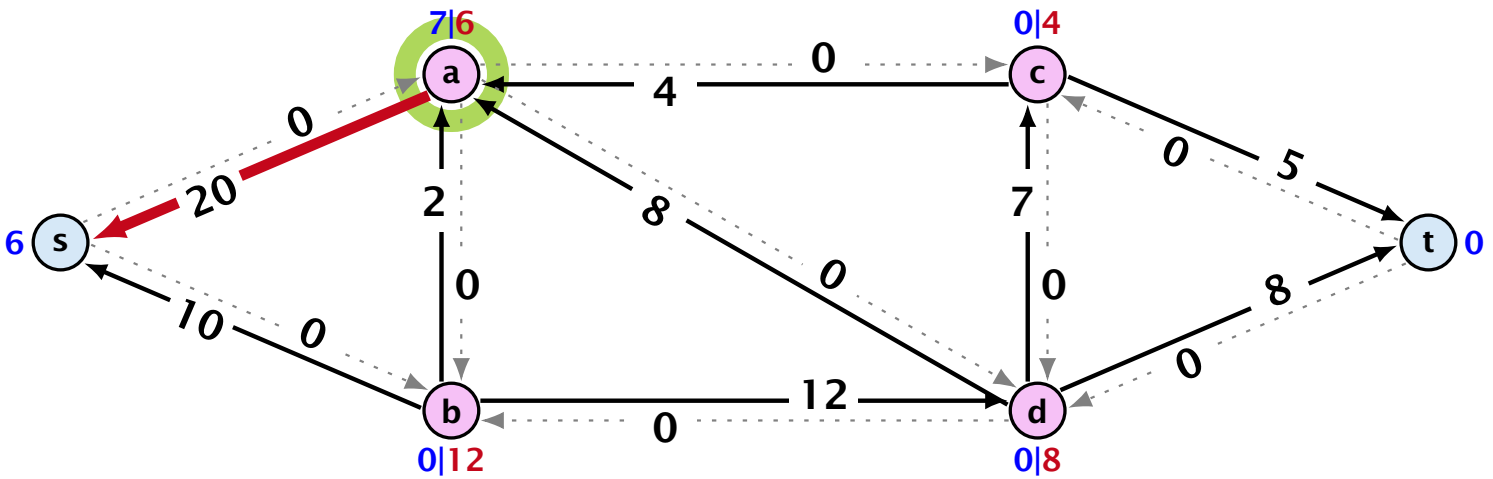




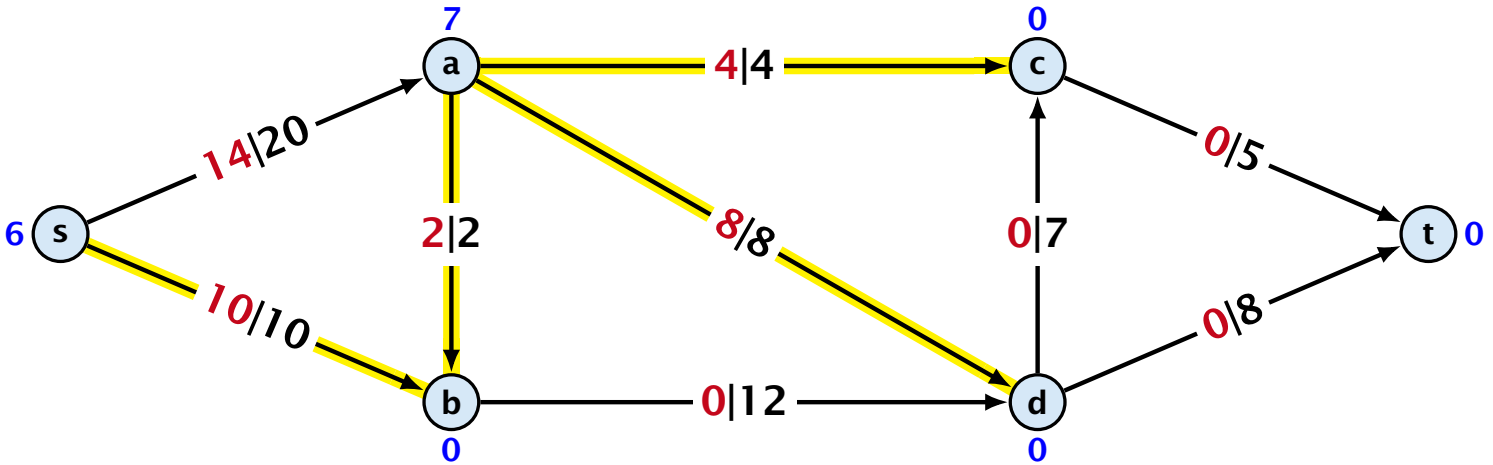
Preflow Push

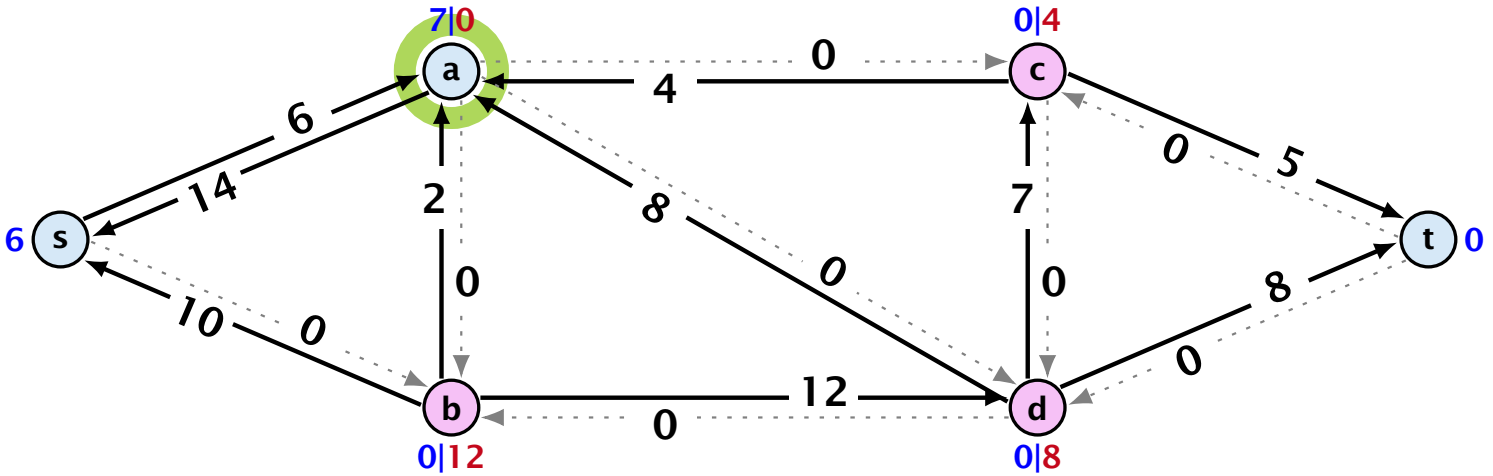


deactivating push

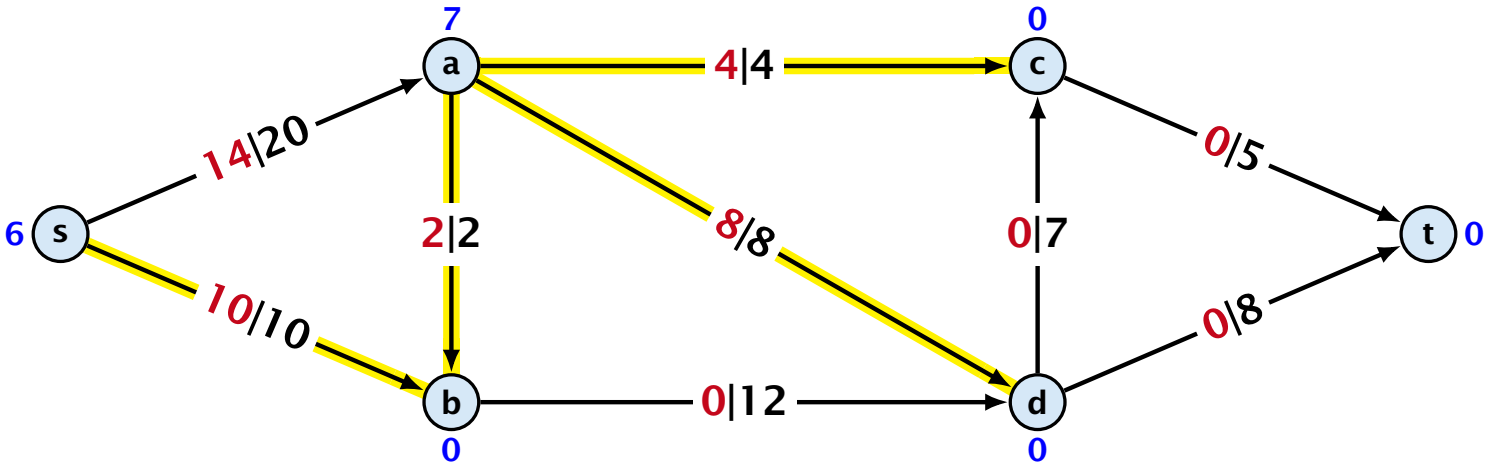


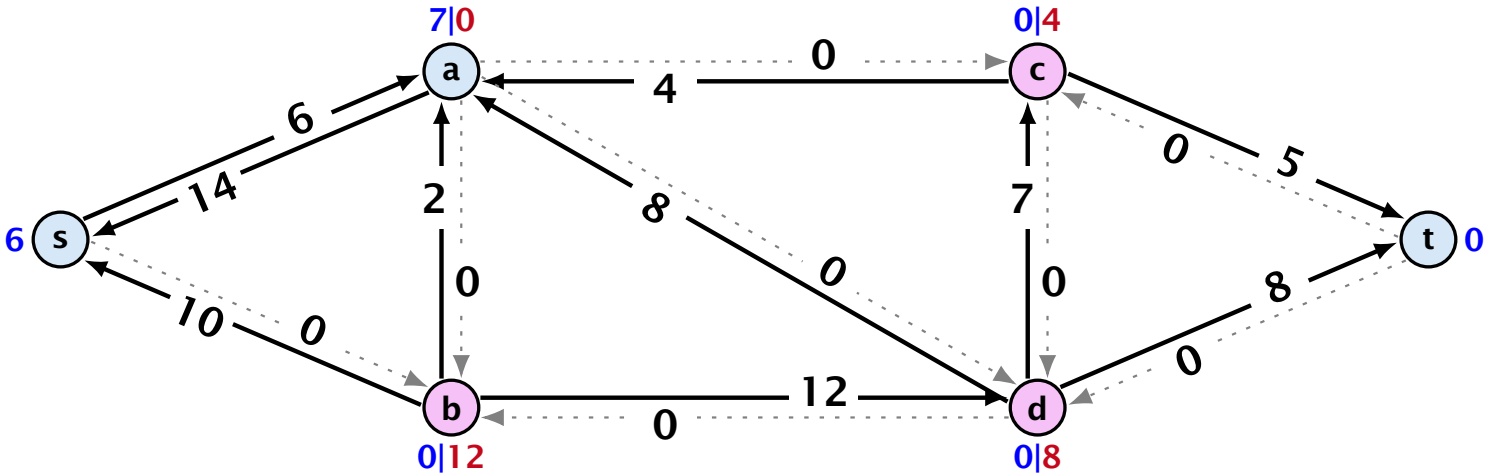
Preflow Push



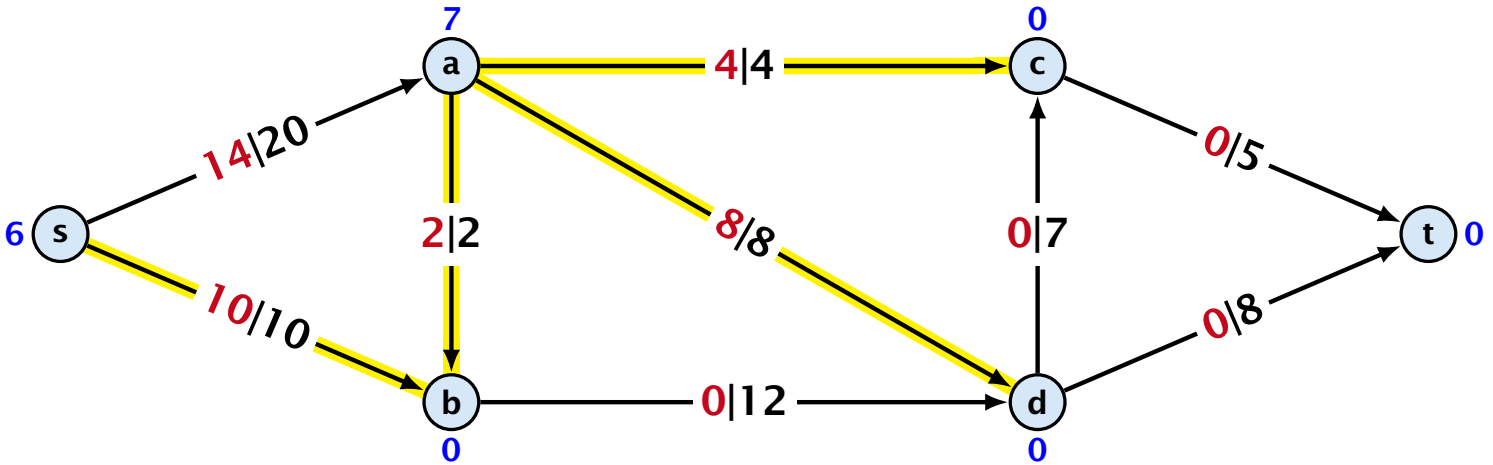


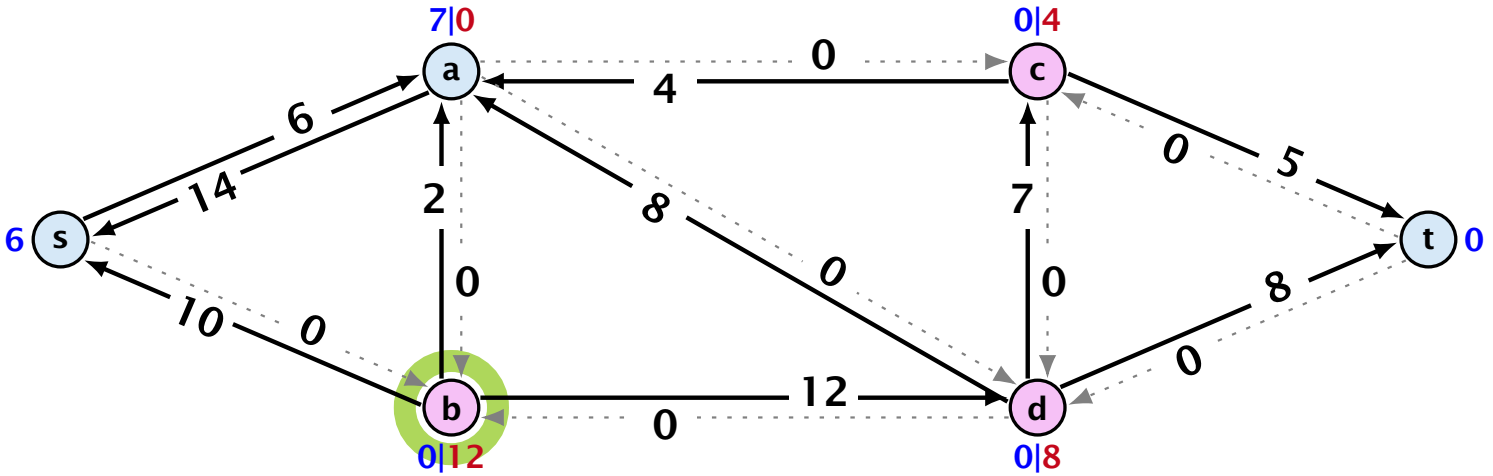
Preflow Push



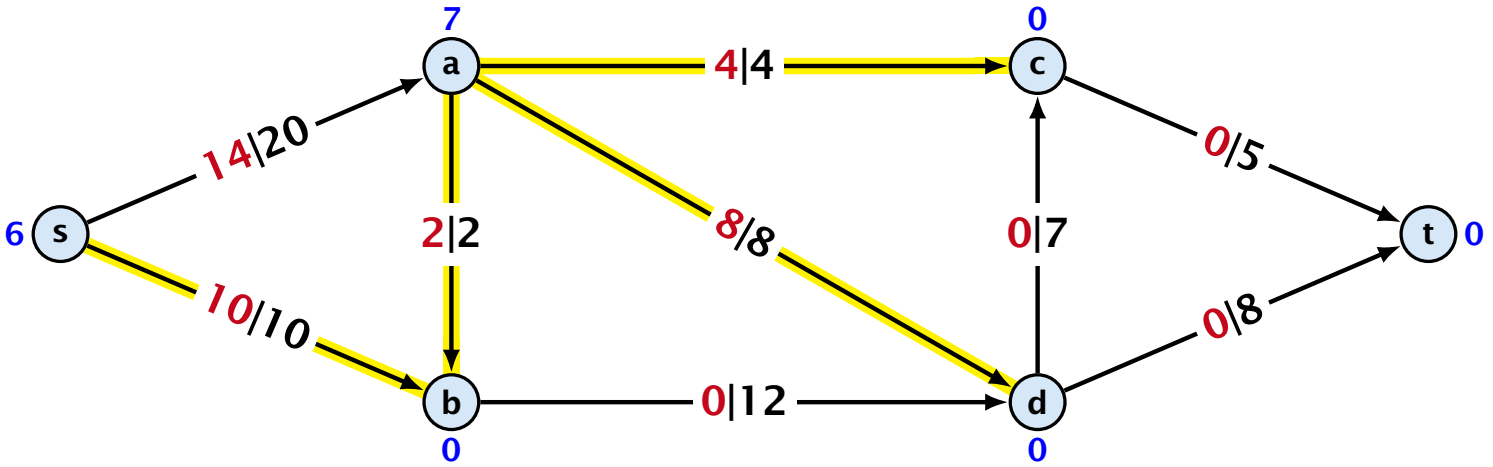


Preflow Push

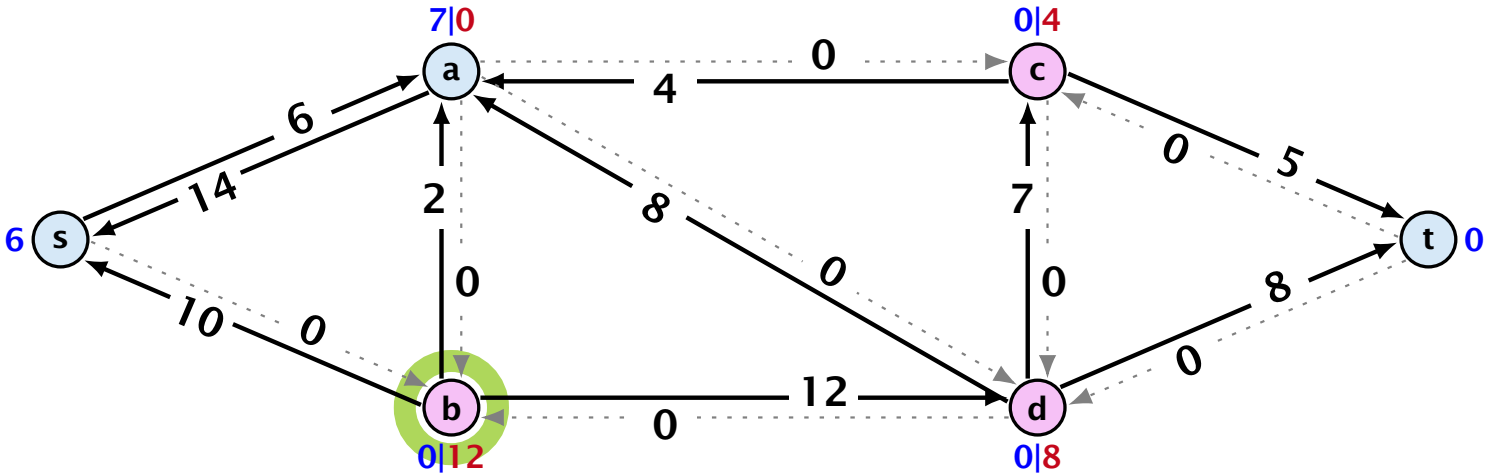




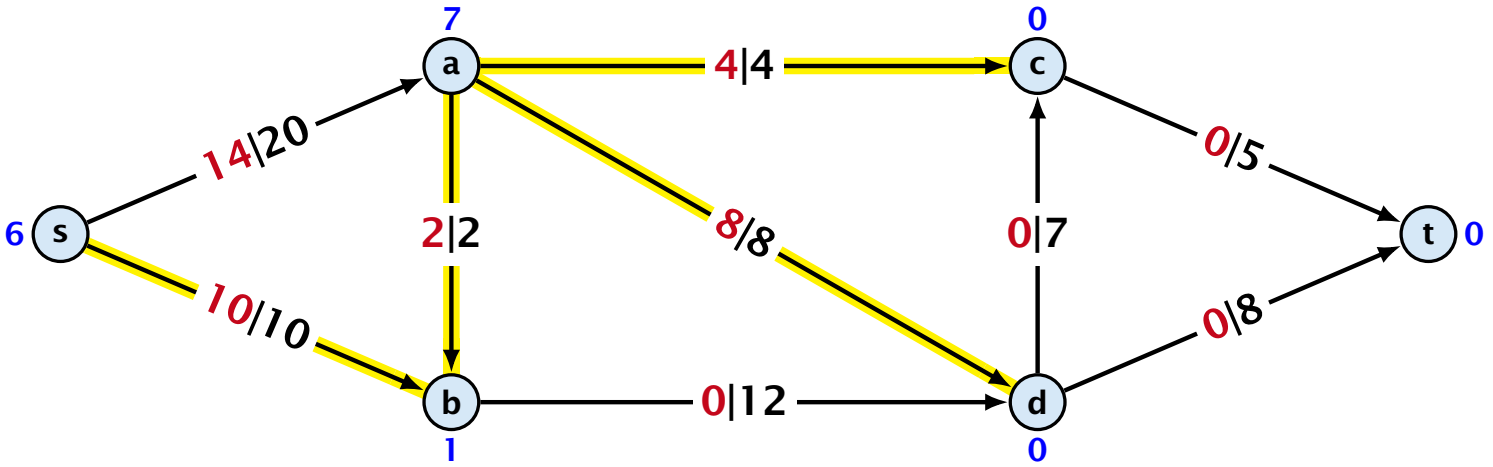
Preflow Push

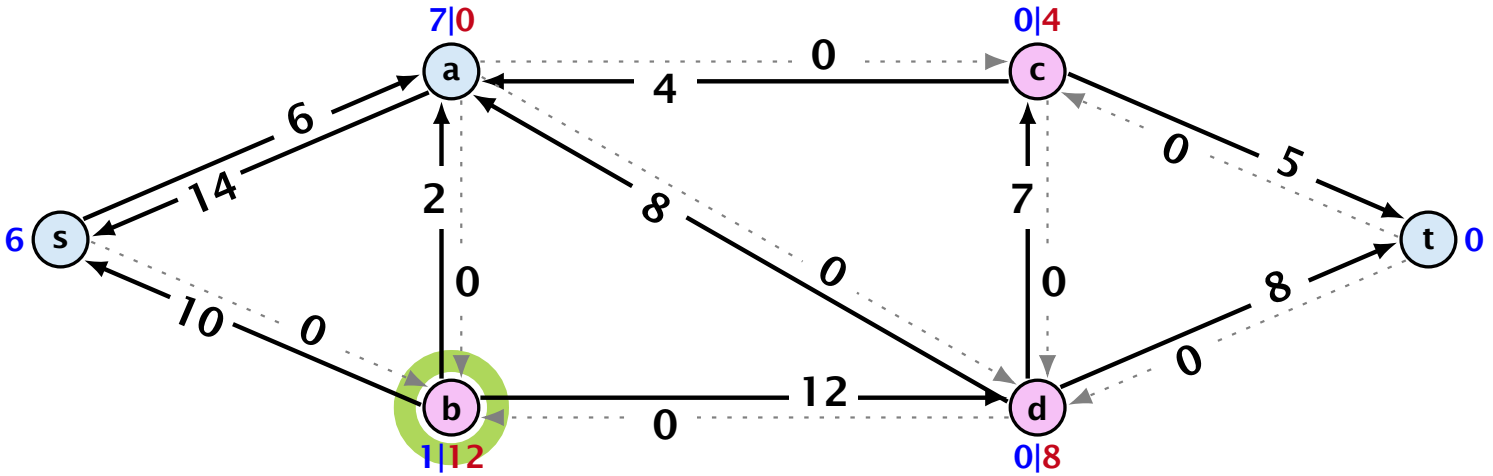


relabel to 1

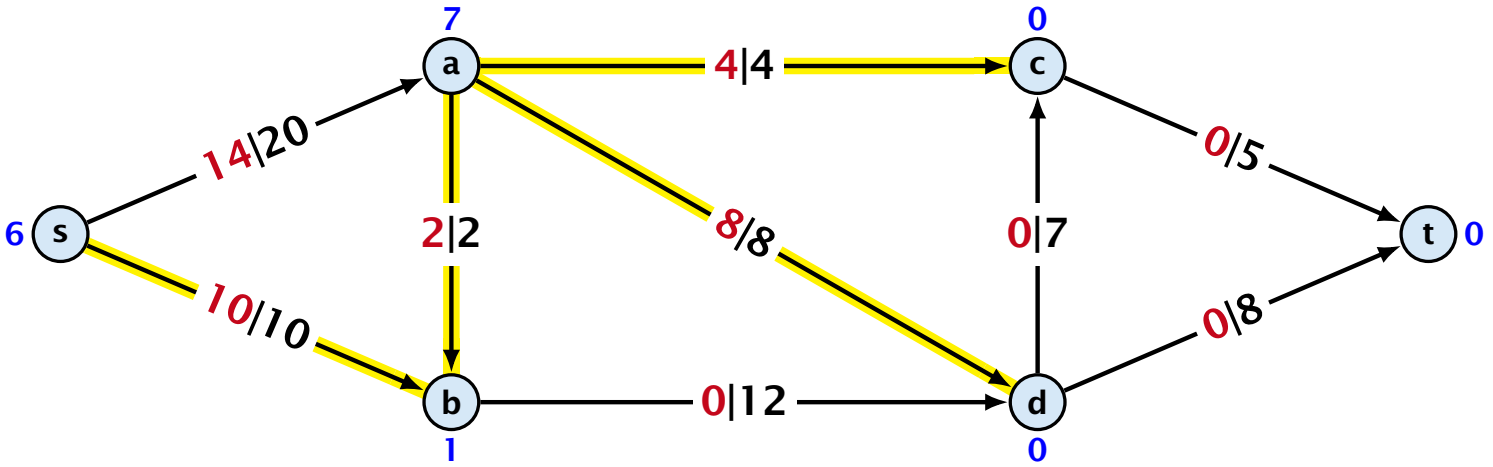


Preflow Push

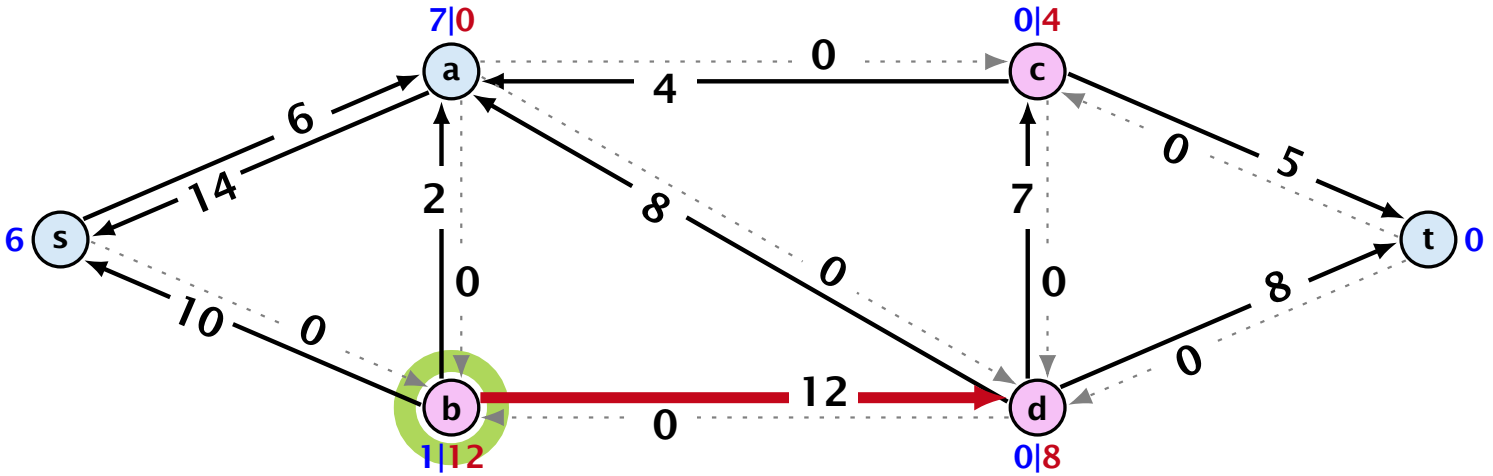




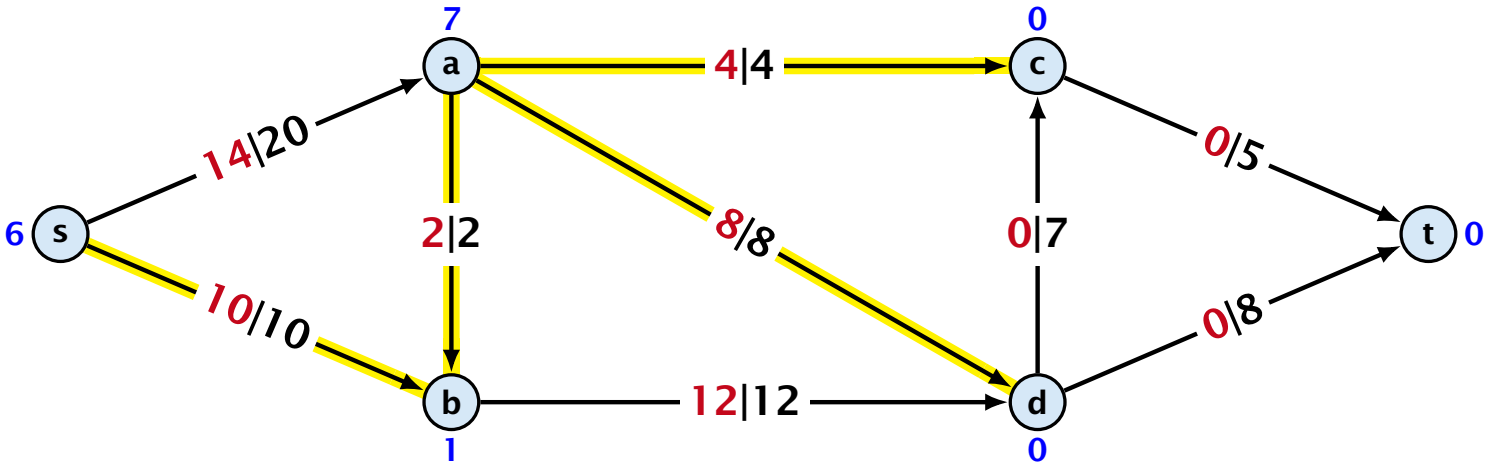
Preflow Push

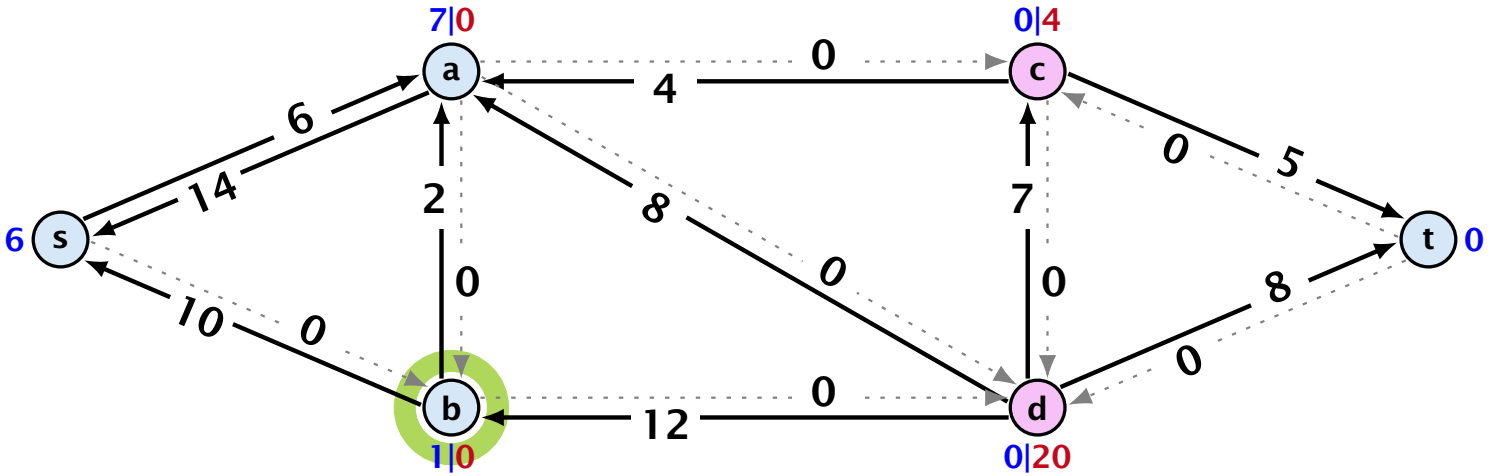


saturating and deactivating push

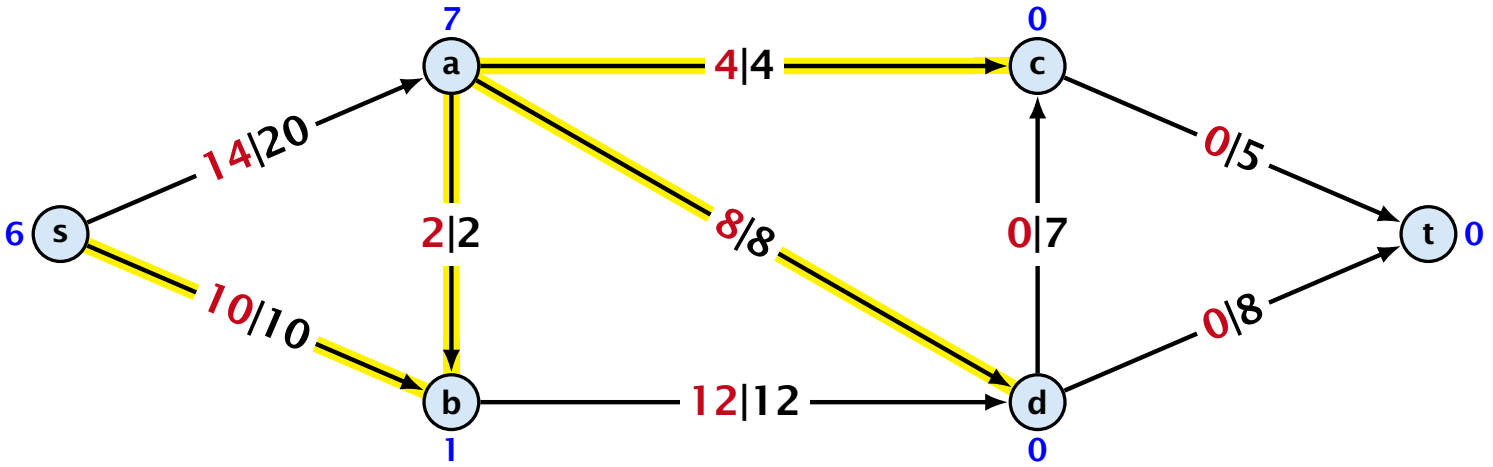


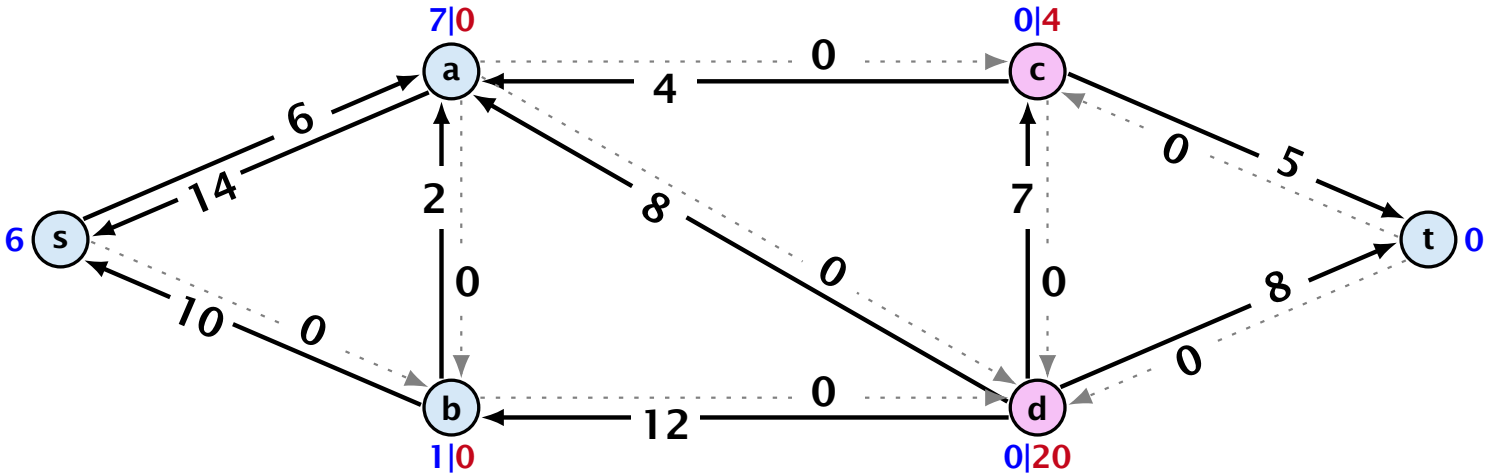
Preflow Push



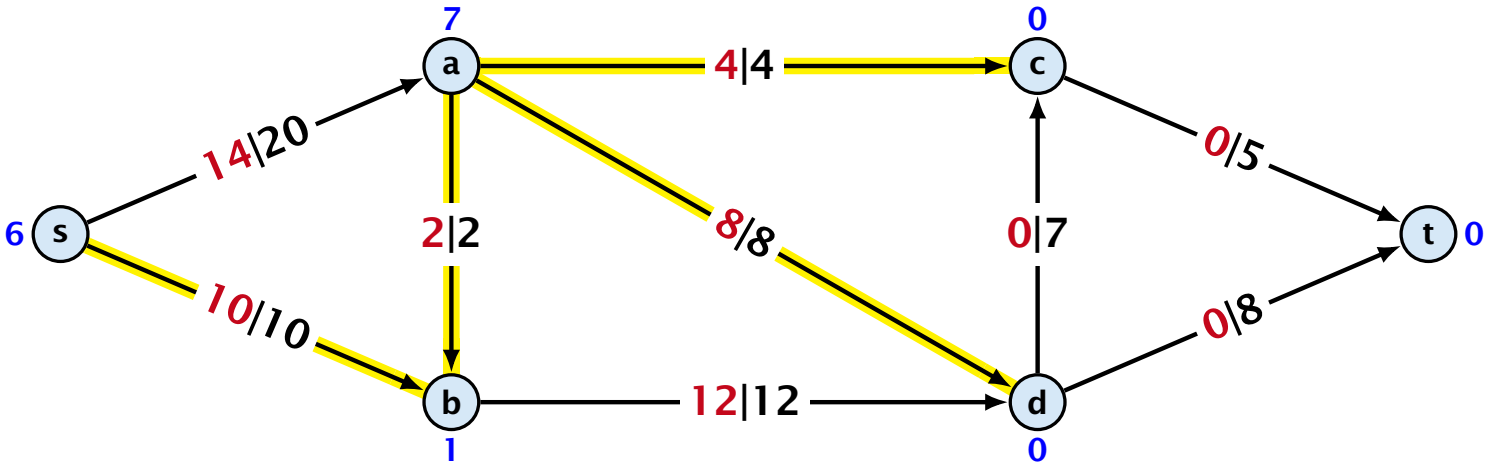


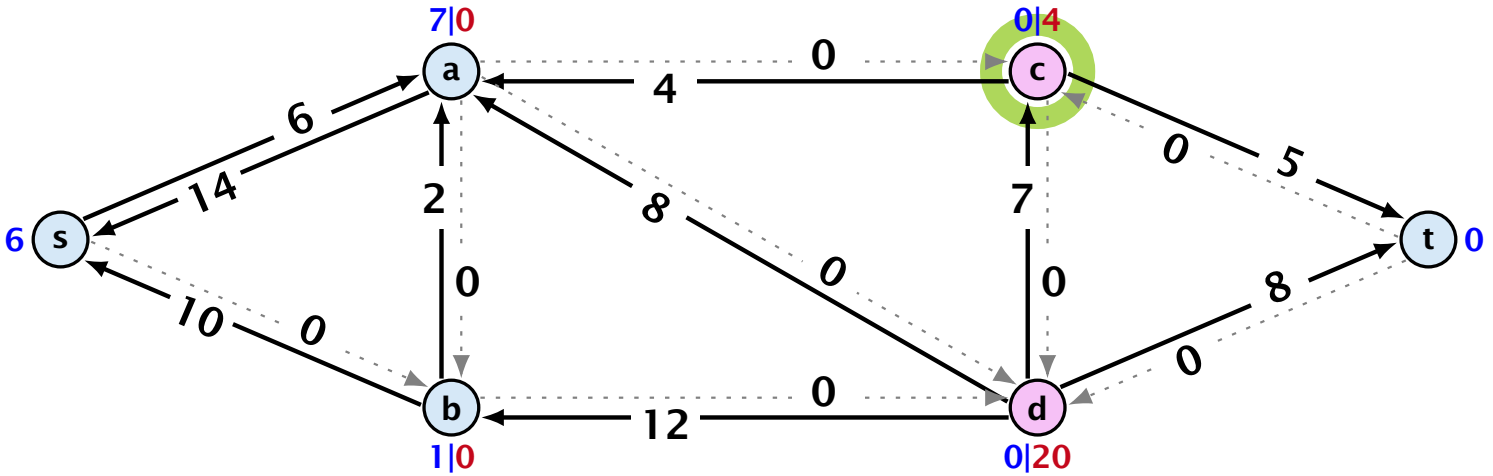
Preflow Push



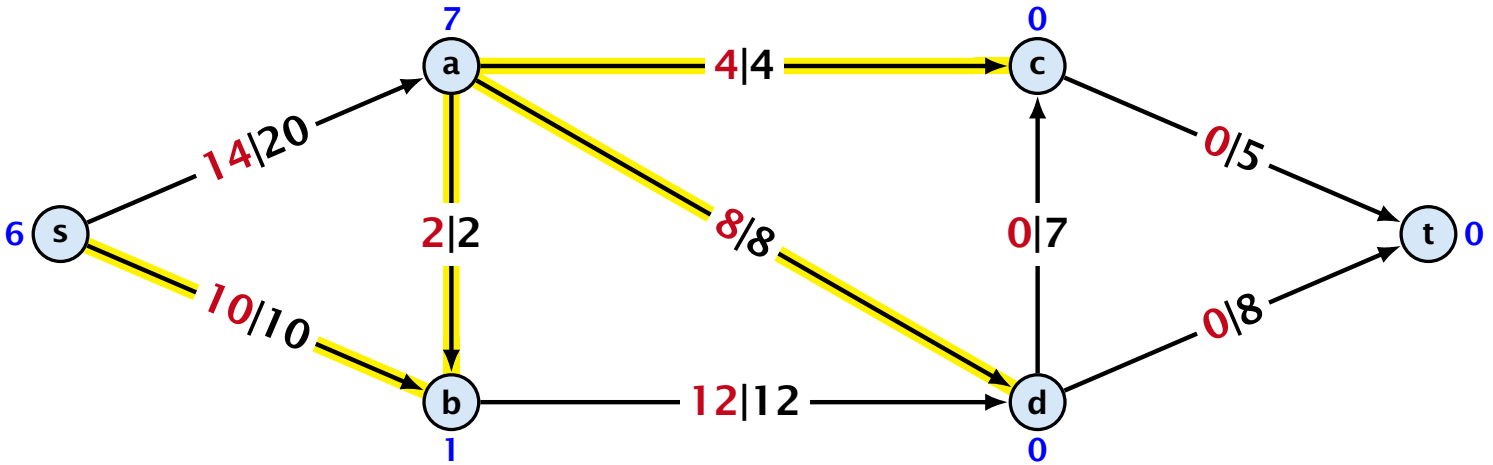


Preflow Push

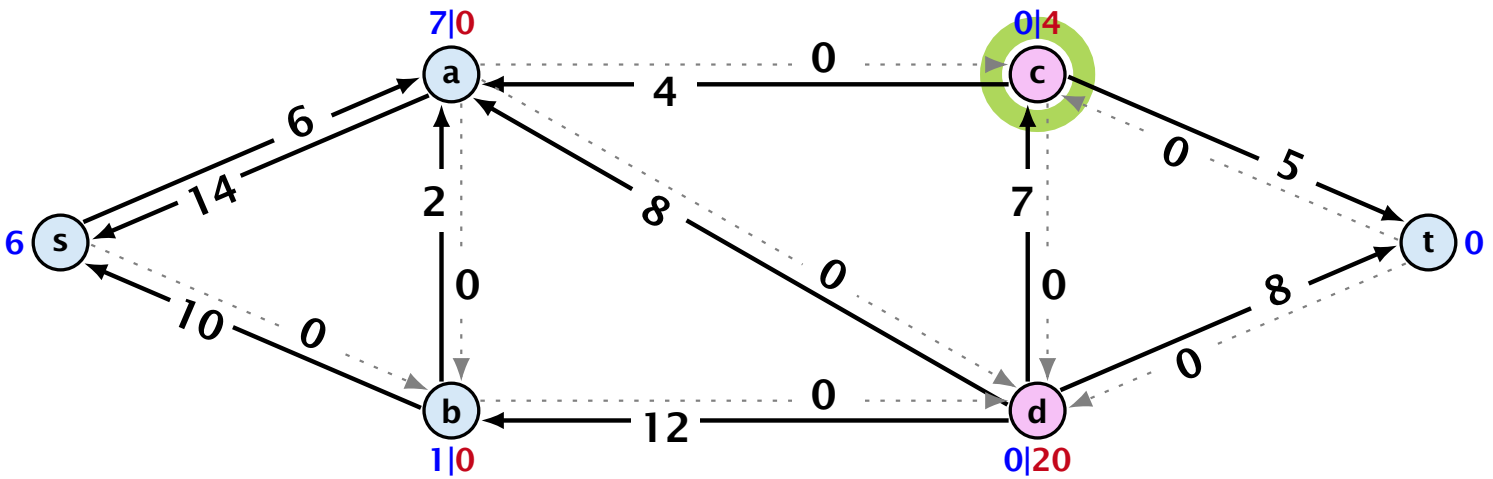




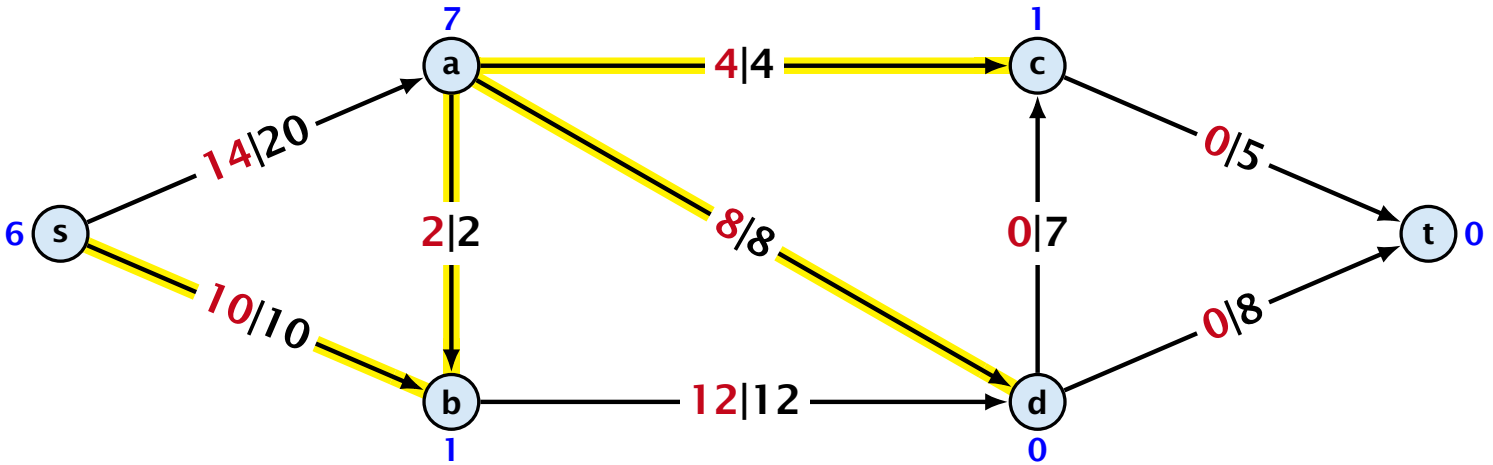
Preflow Push

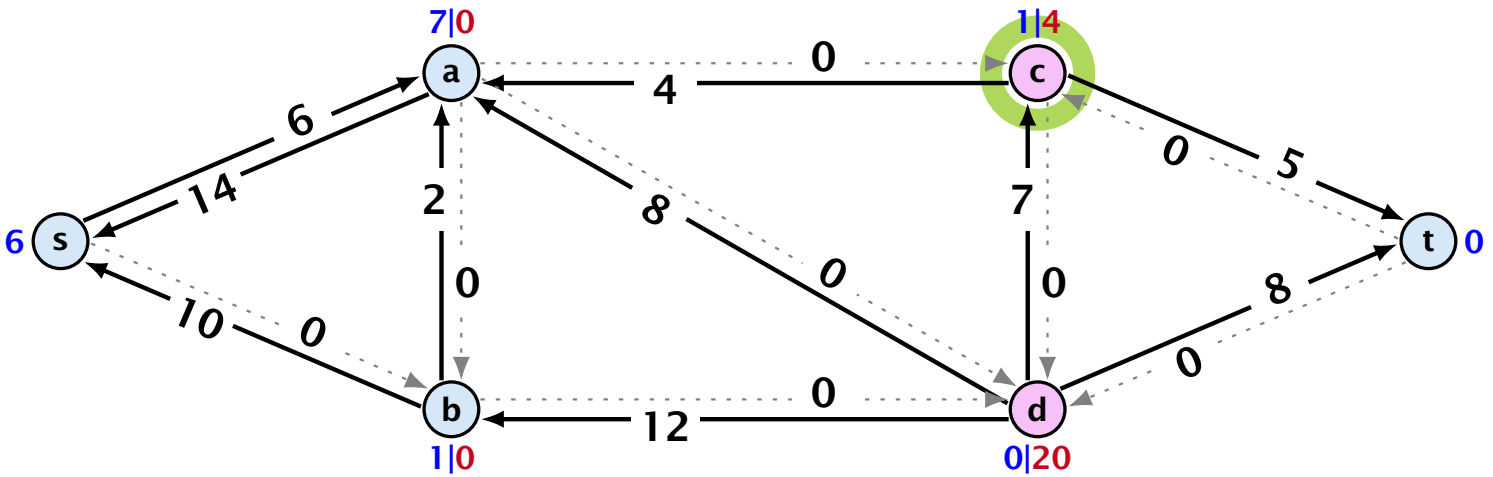


relabel to 1

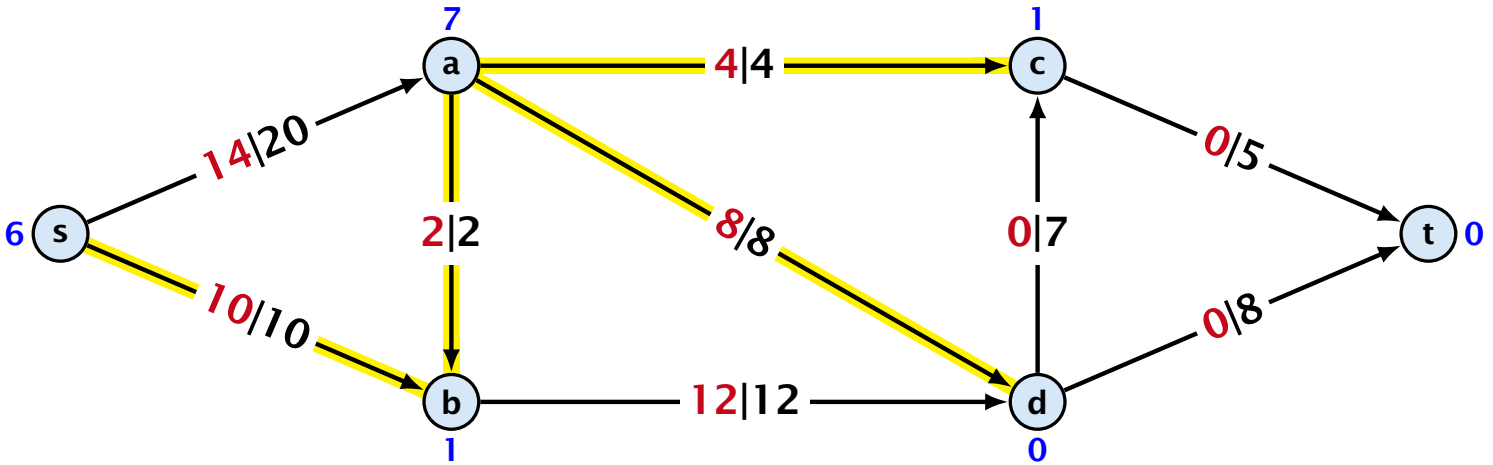


Preflow Push

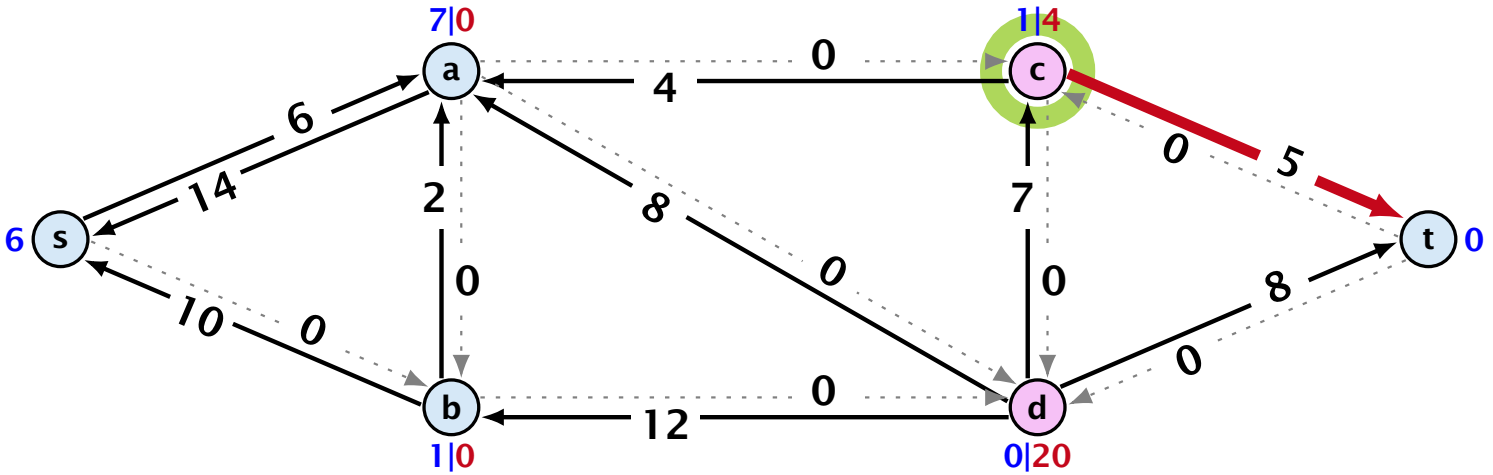




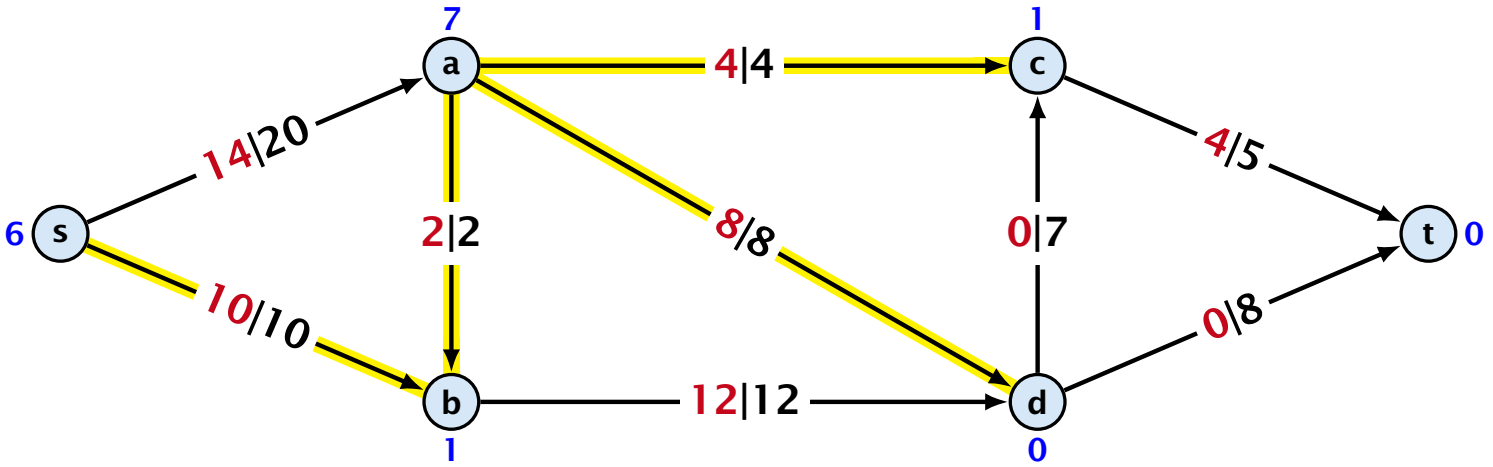
Preflow Push

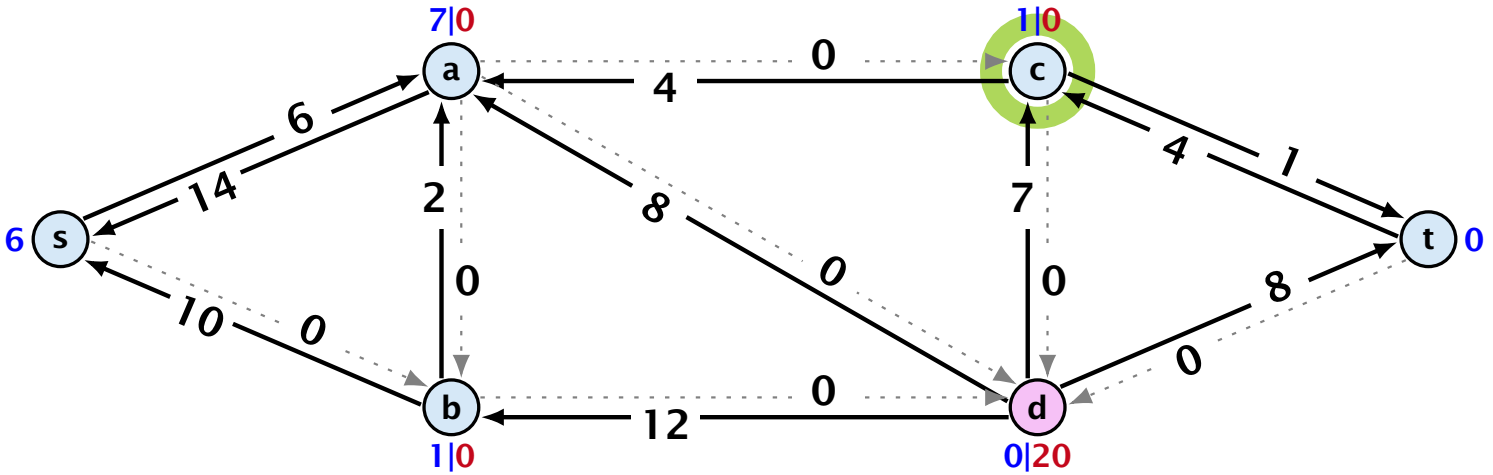


deactivating push

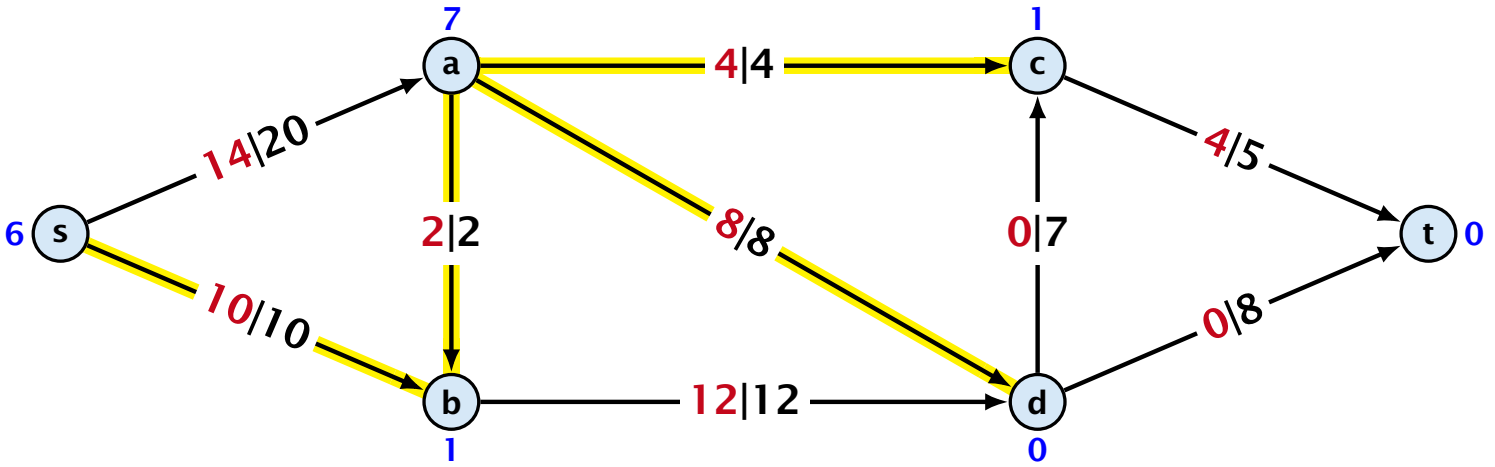


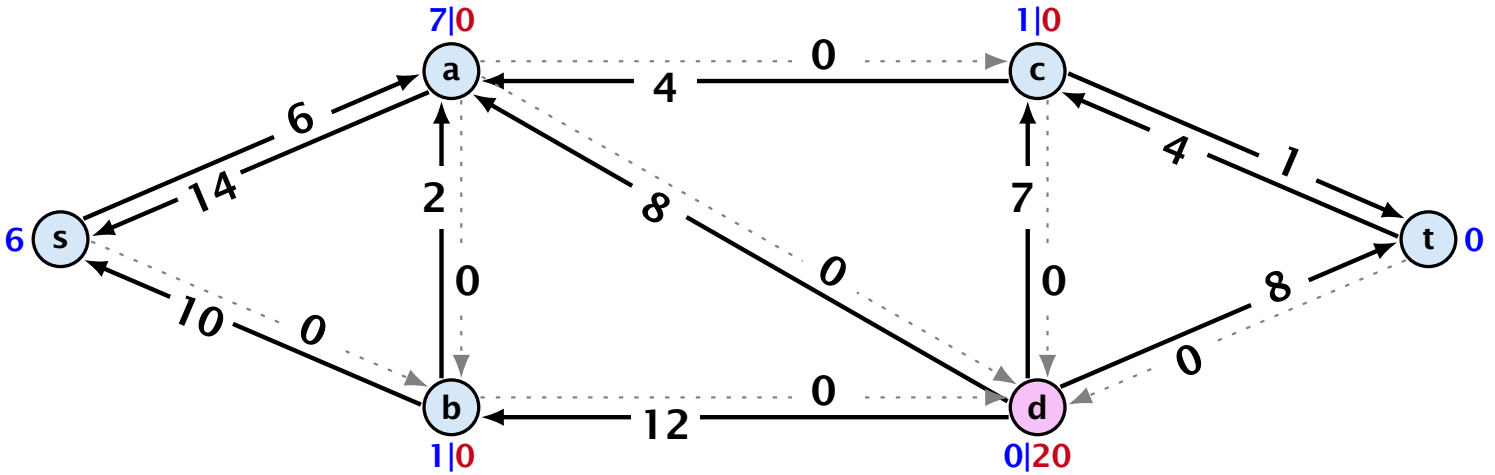
Preflow Push



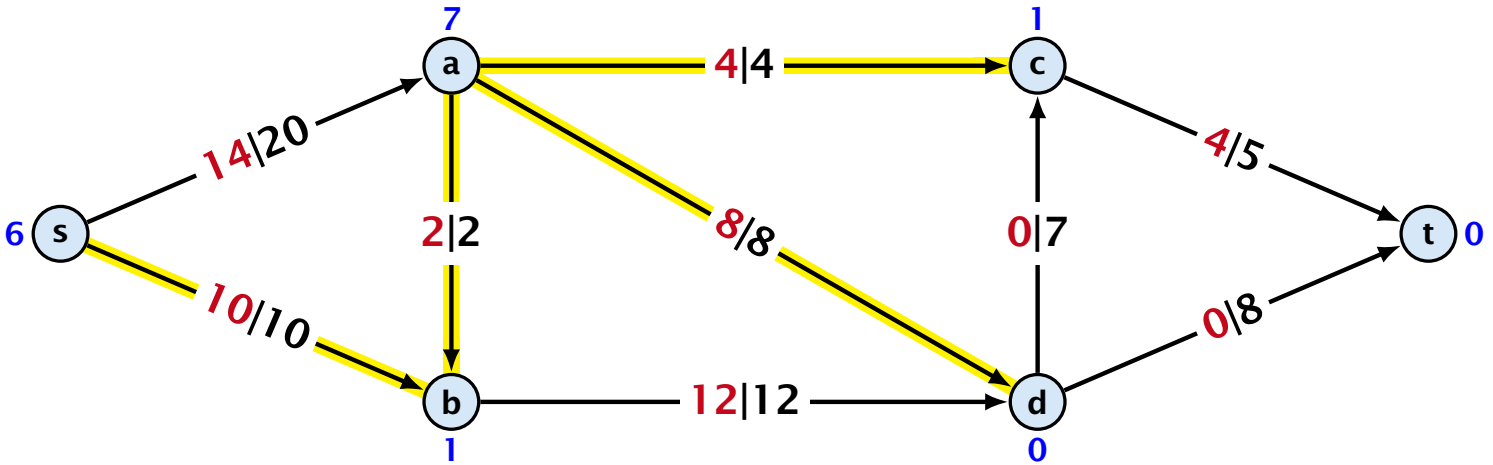


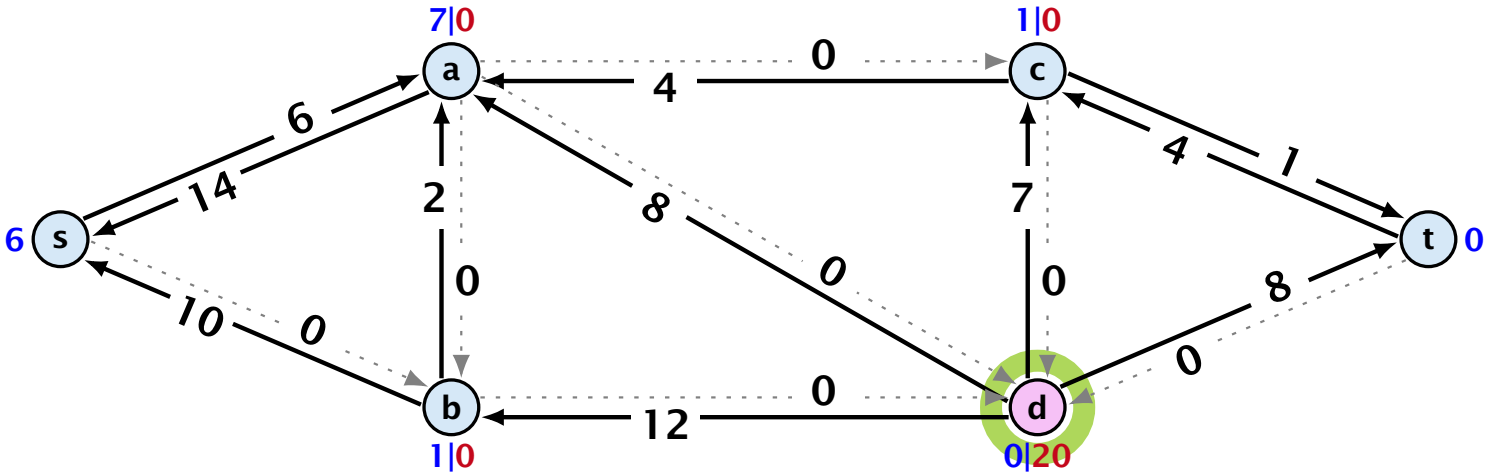
Preflow Push



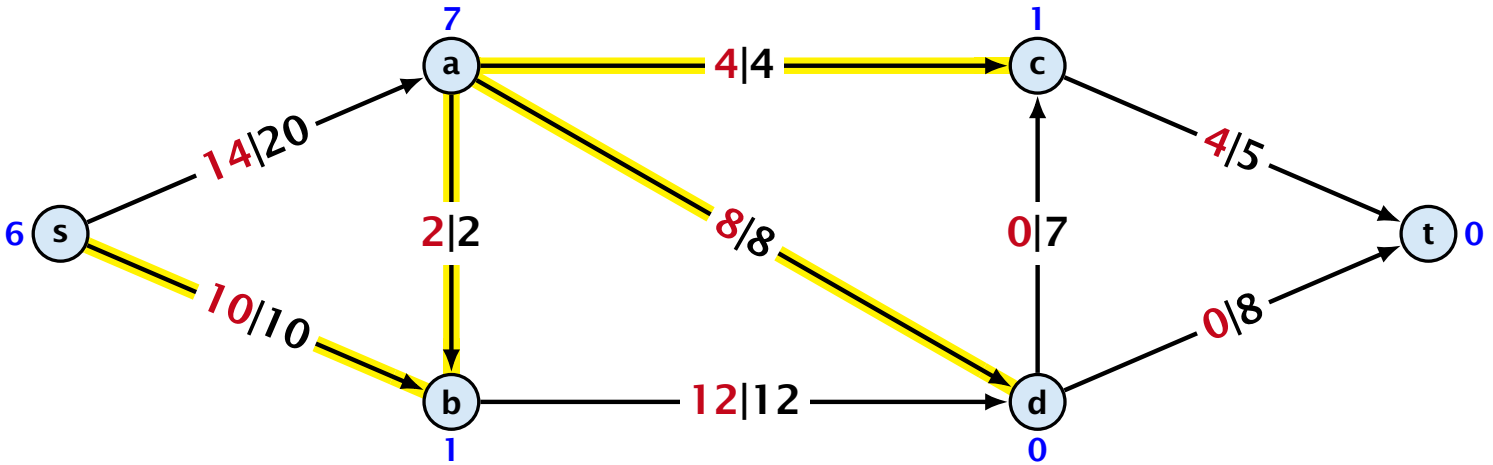


Preflow Push

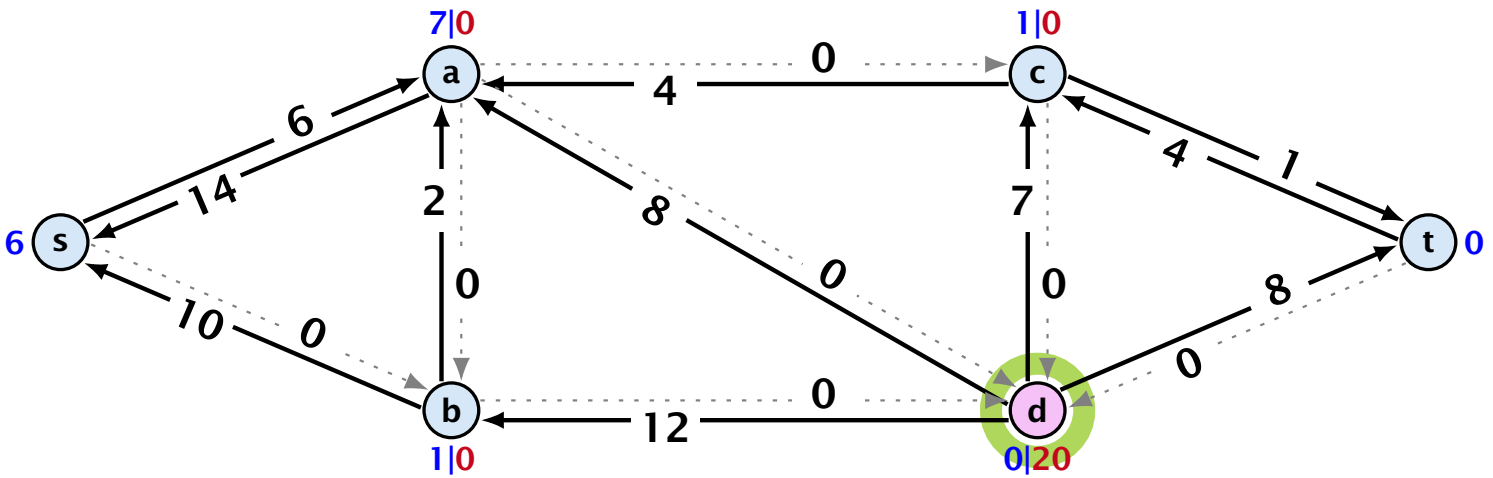




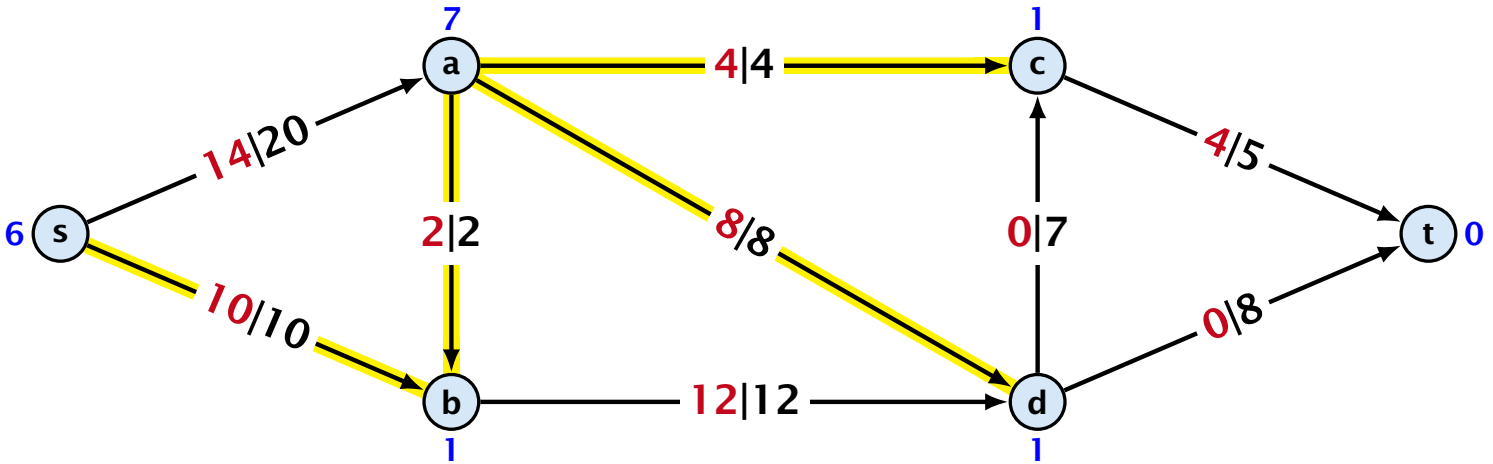
Preflow Push

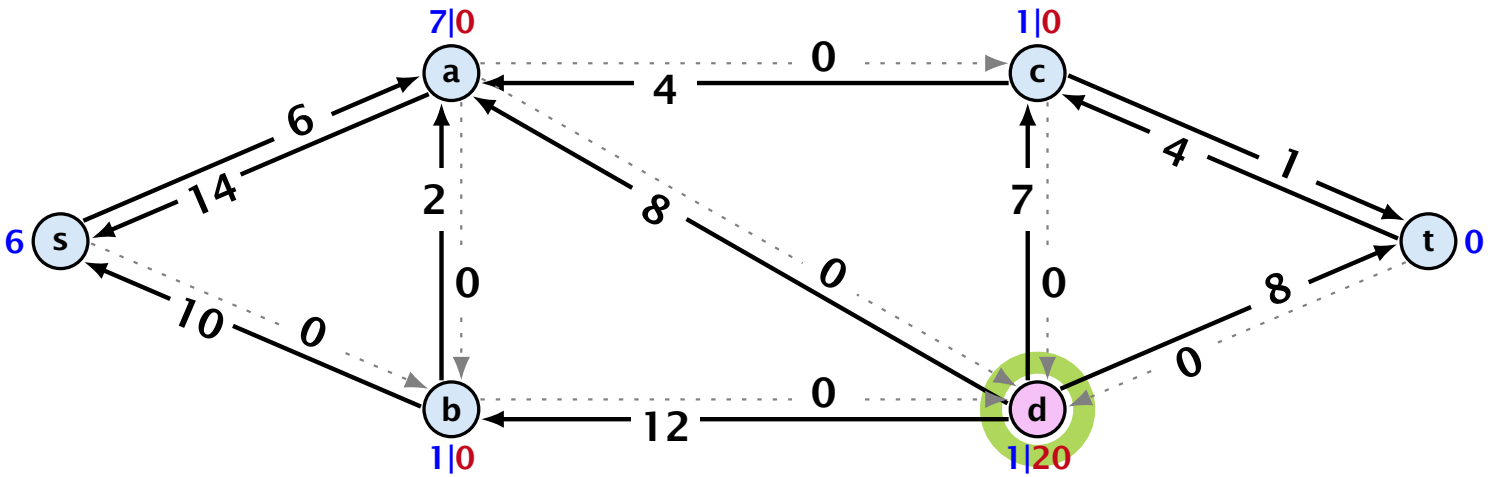


relabel to 1

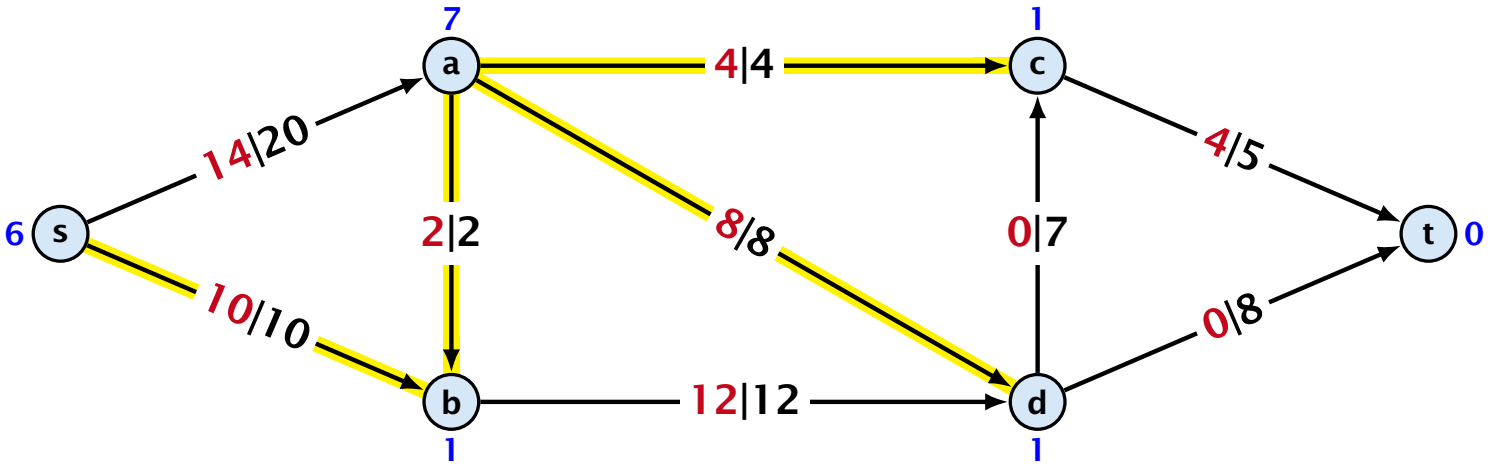


Preflow Push

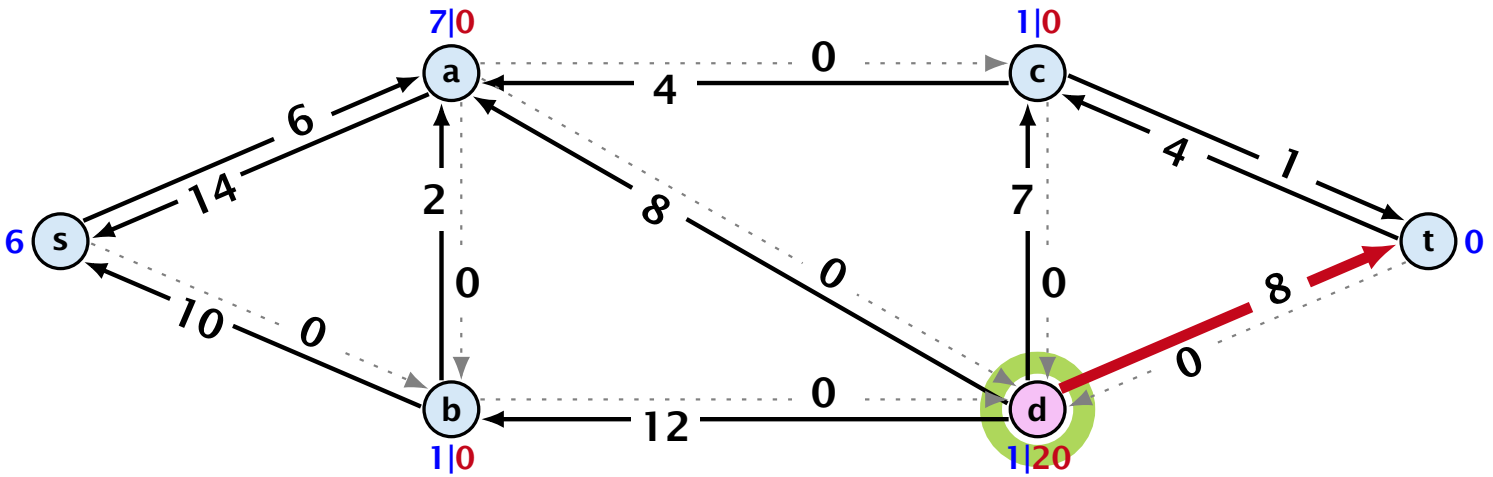




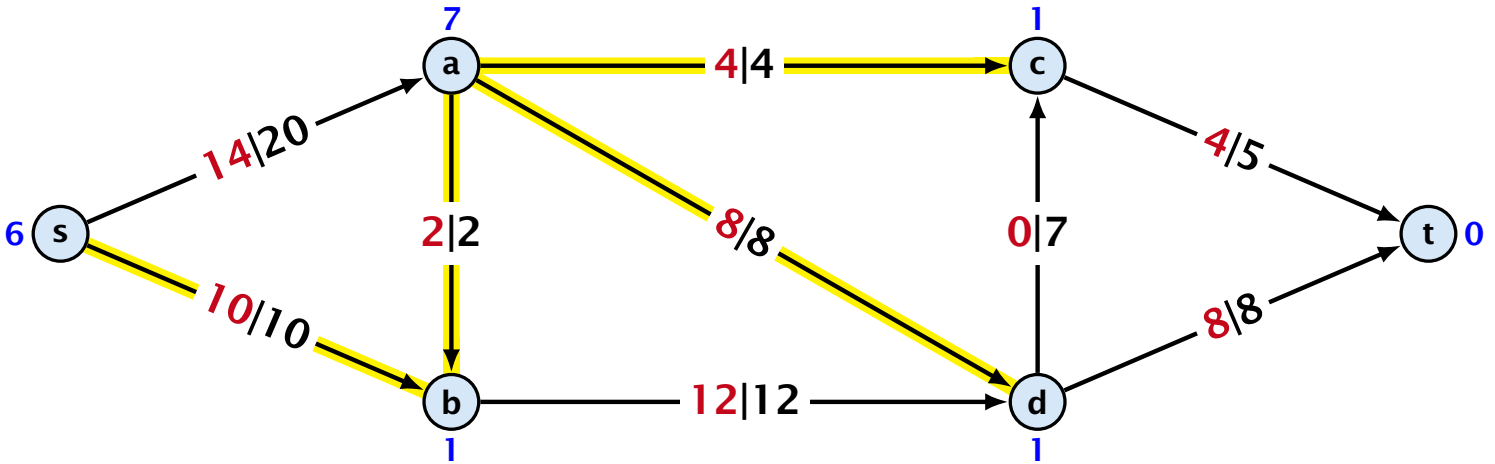
Preflow Push

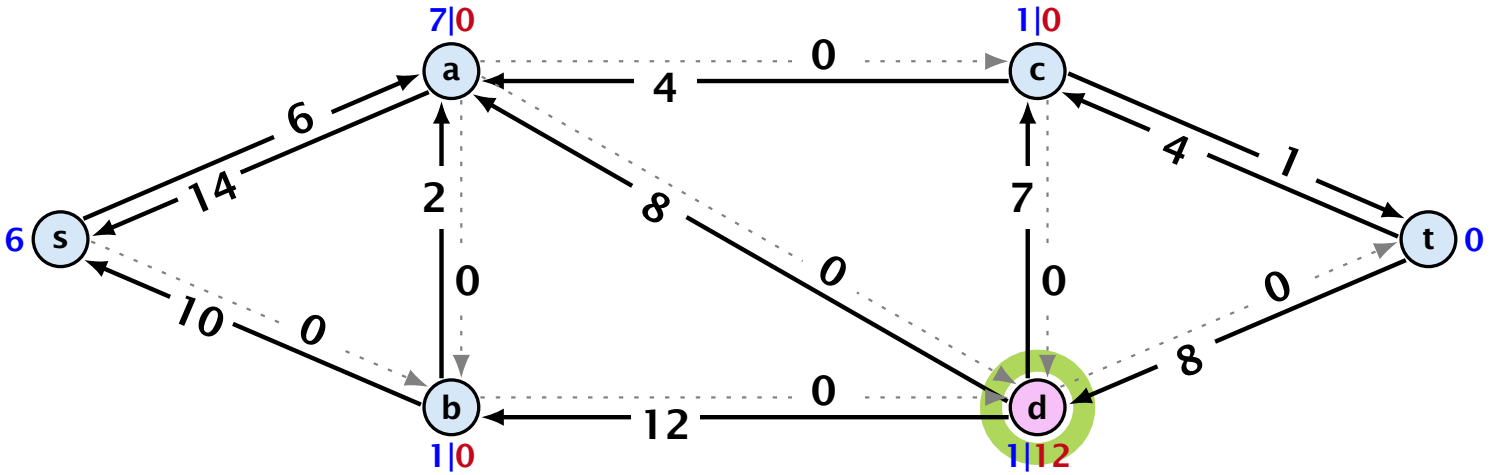


saturating push

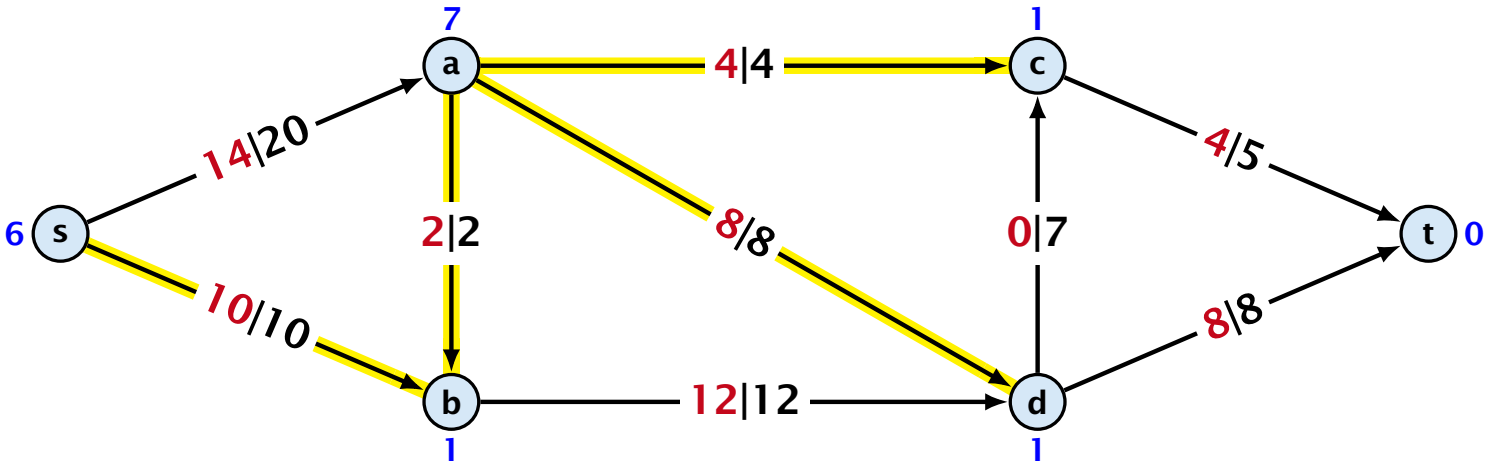


Preflow Push

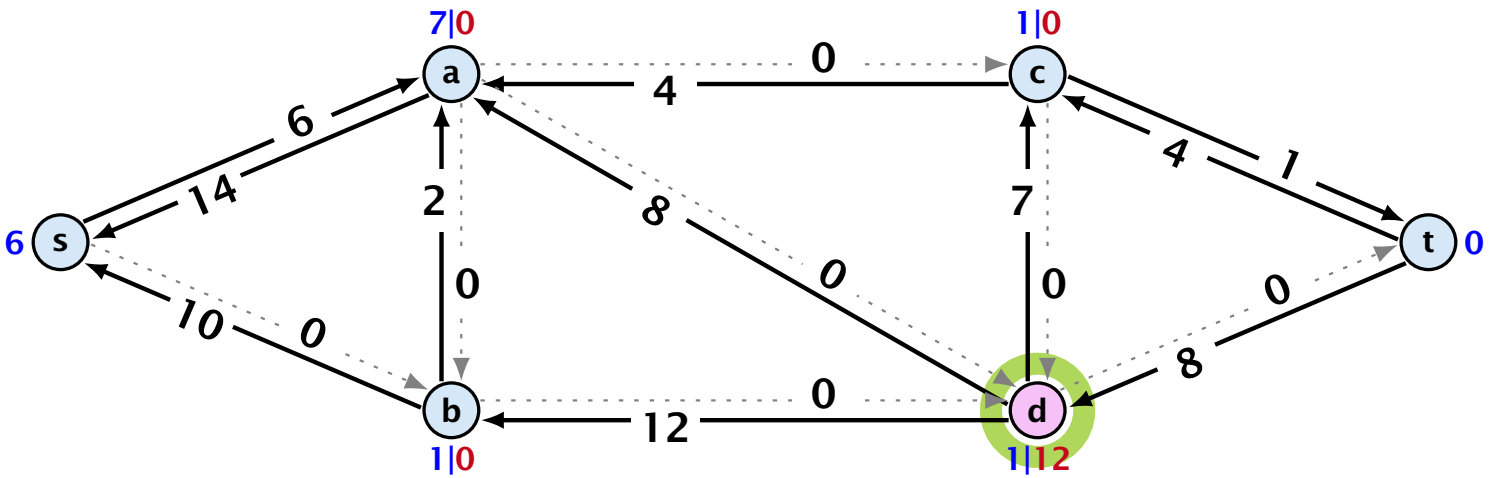




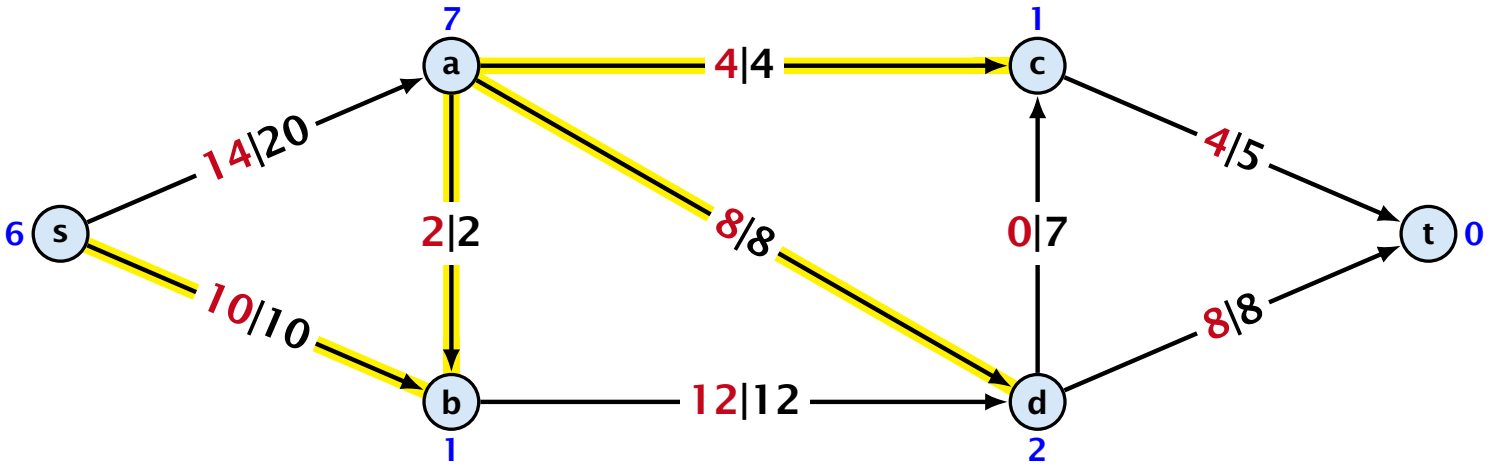
Preflow Push

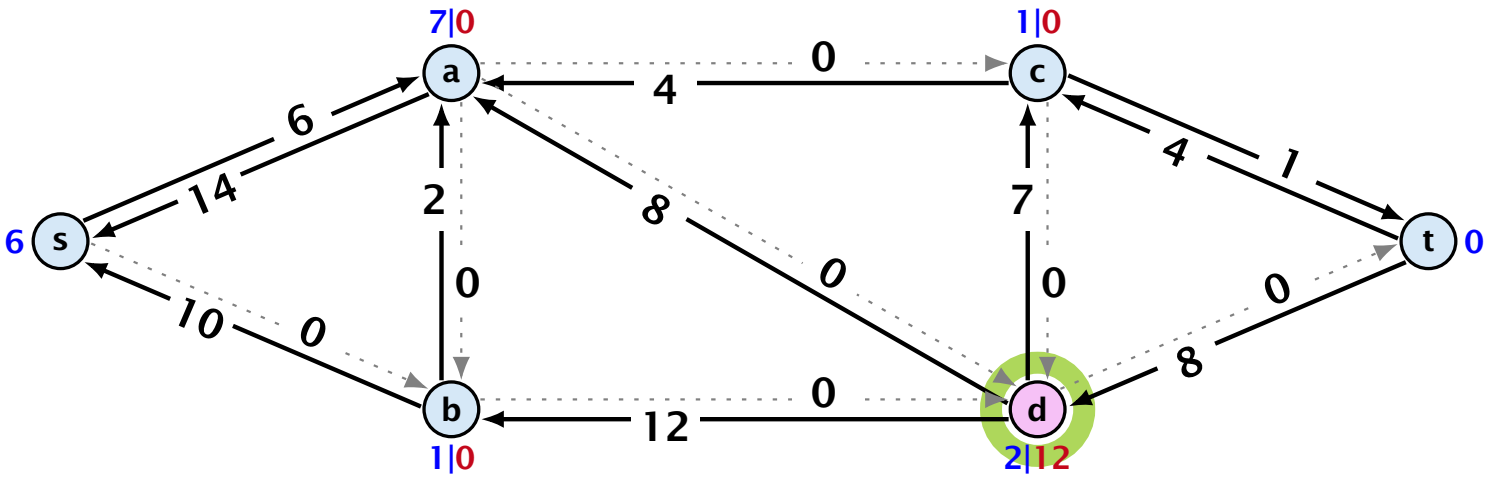


relabel to 2

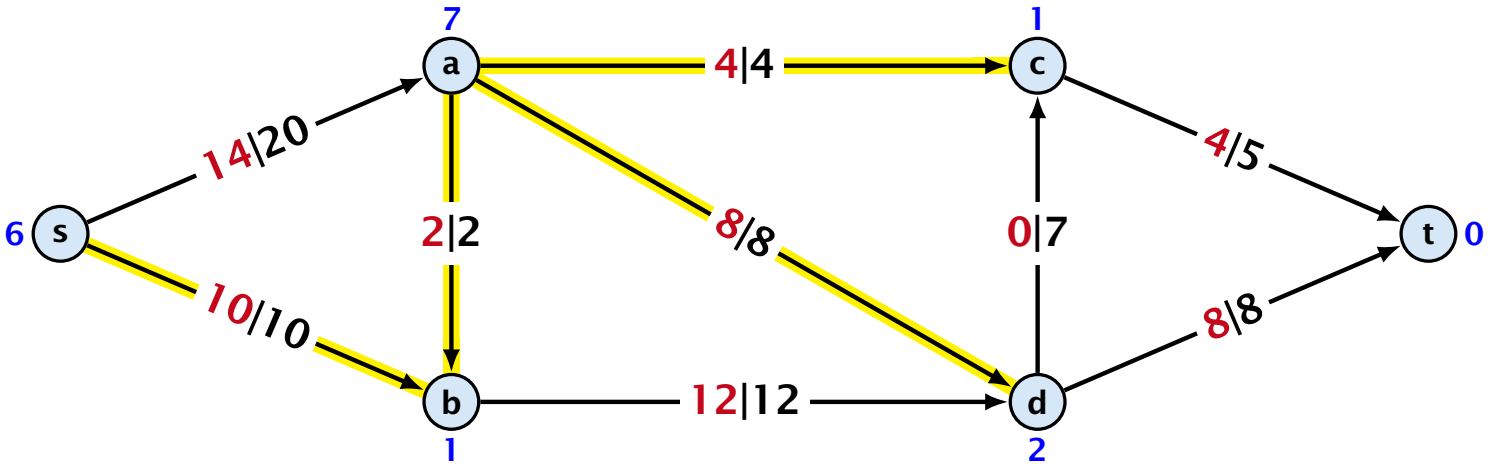


Preflow Push

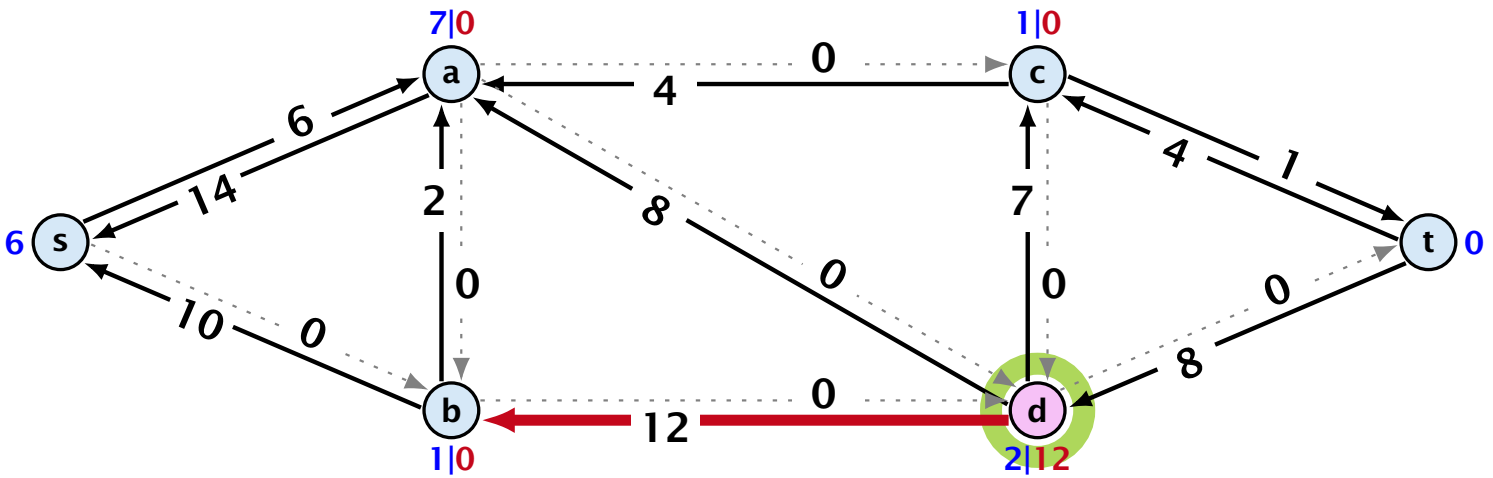




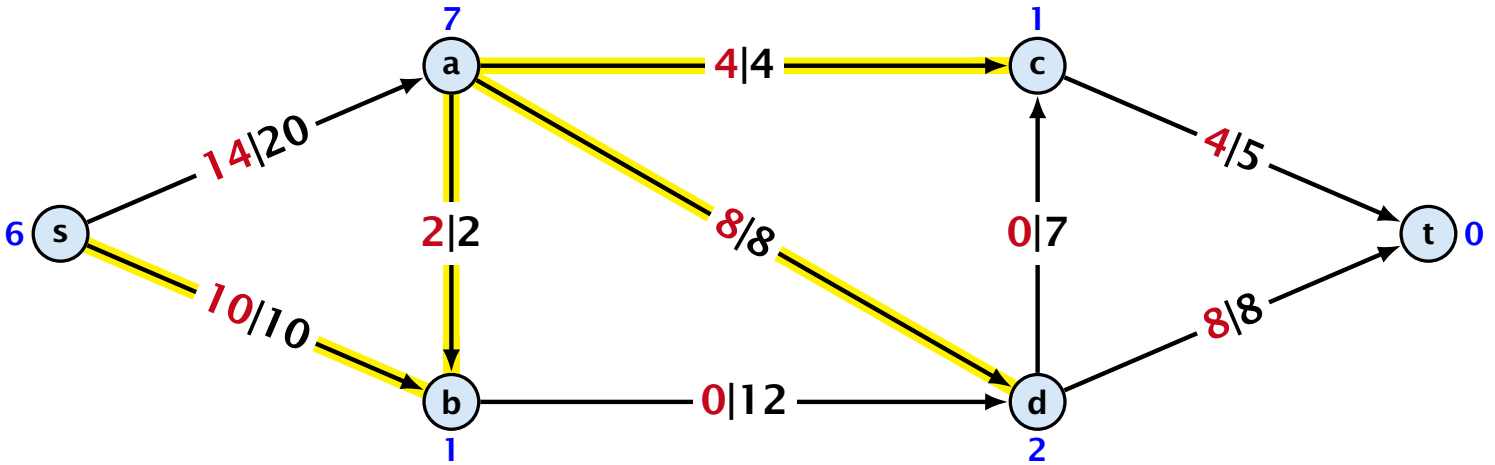
Preflow Push

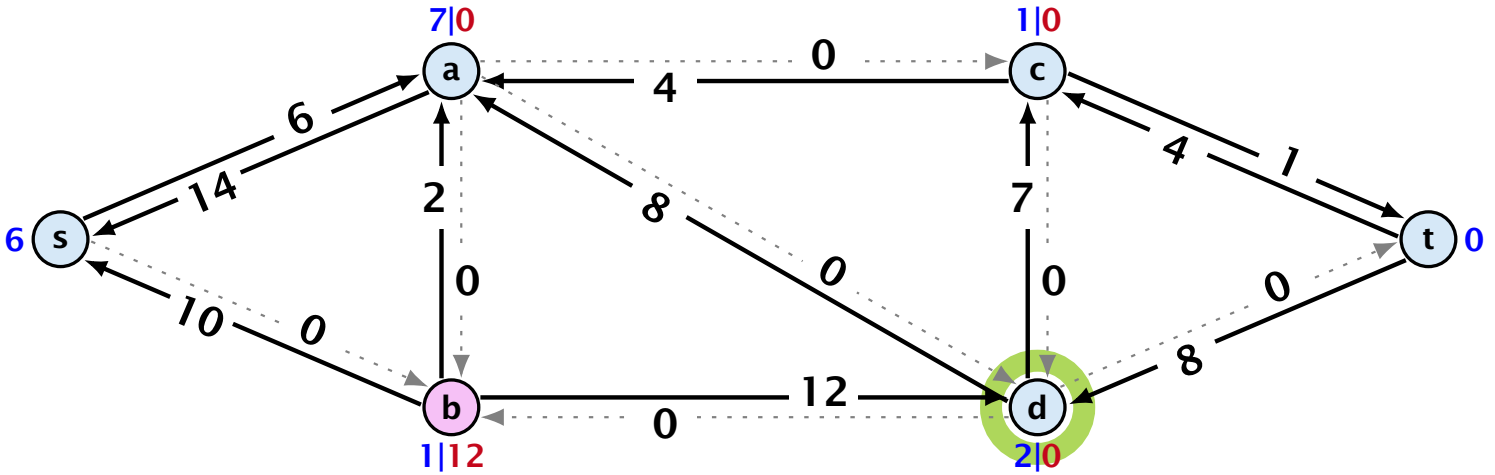


saturating and deactivating push

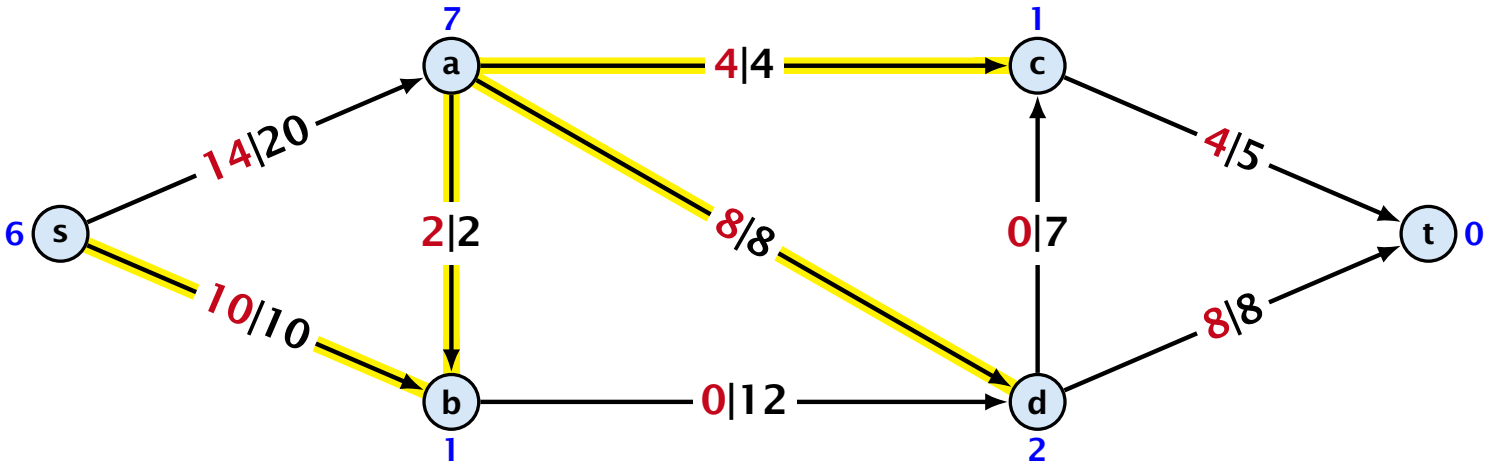


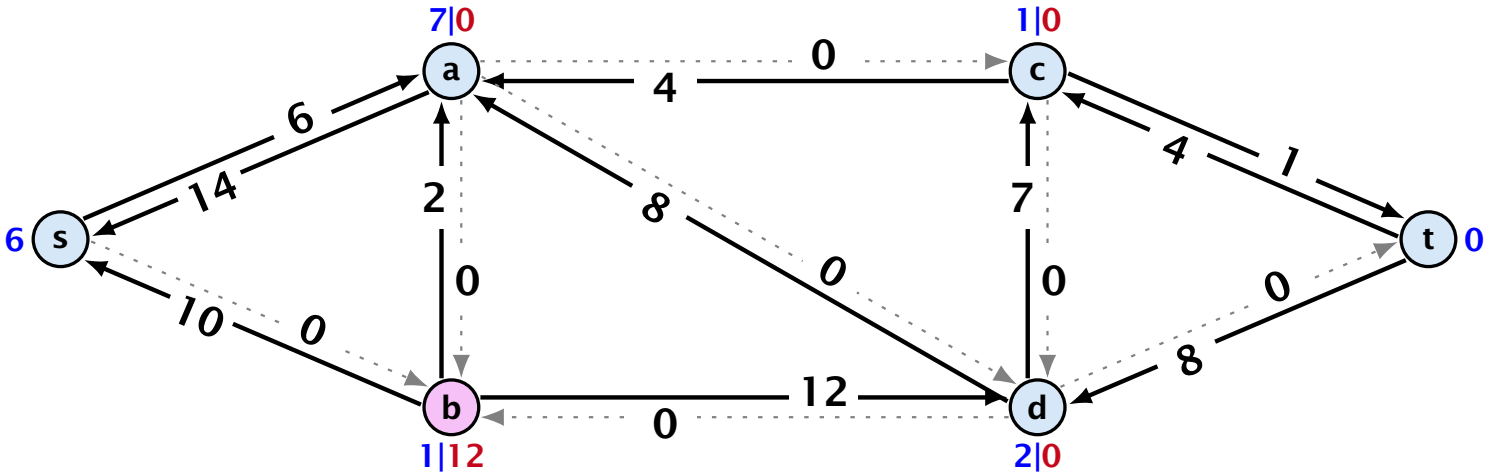
Preflow Push



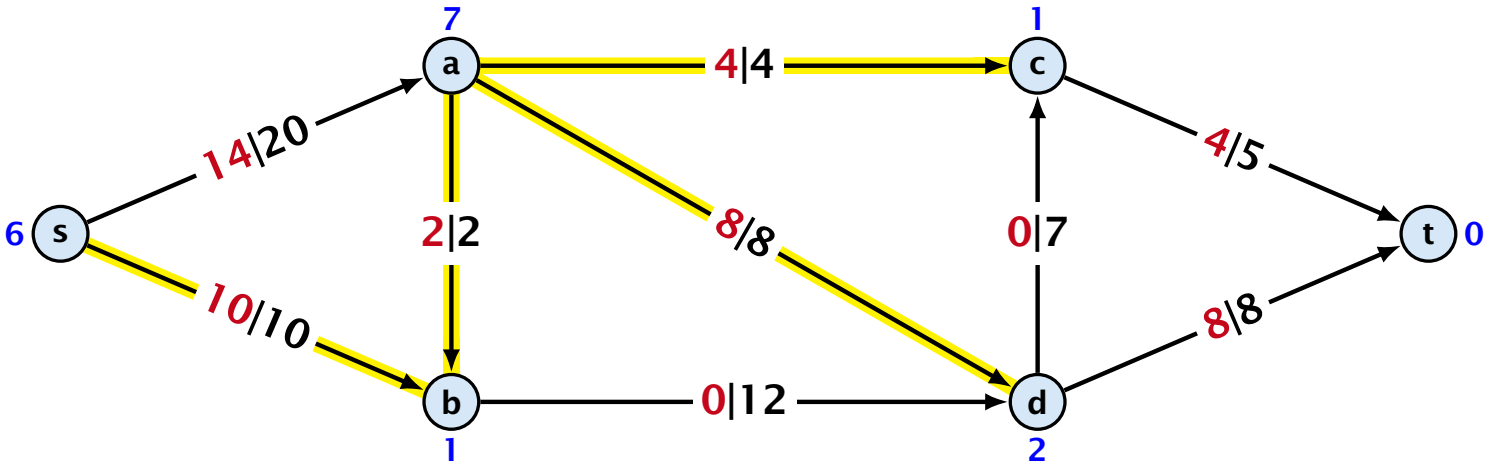


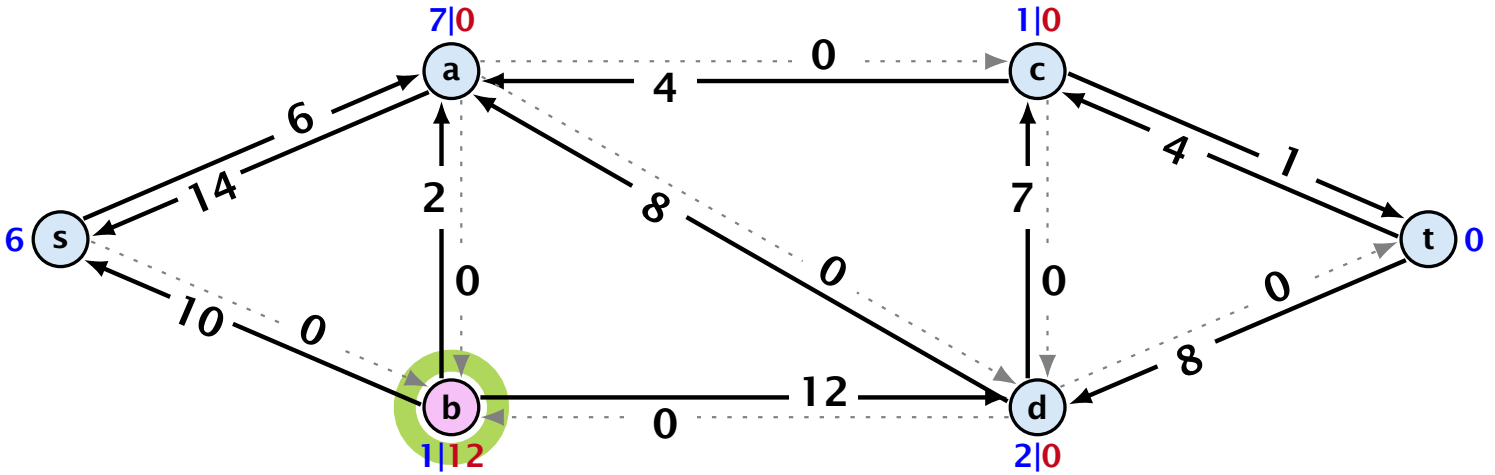
Preflow Push



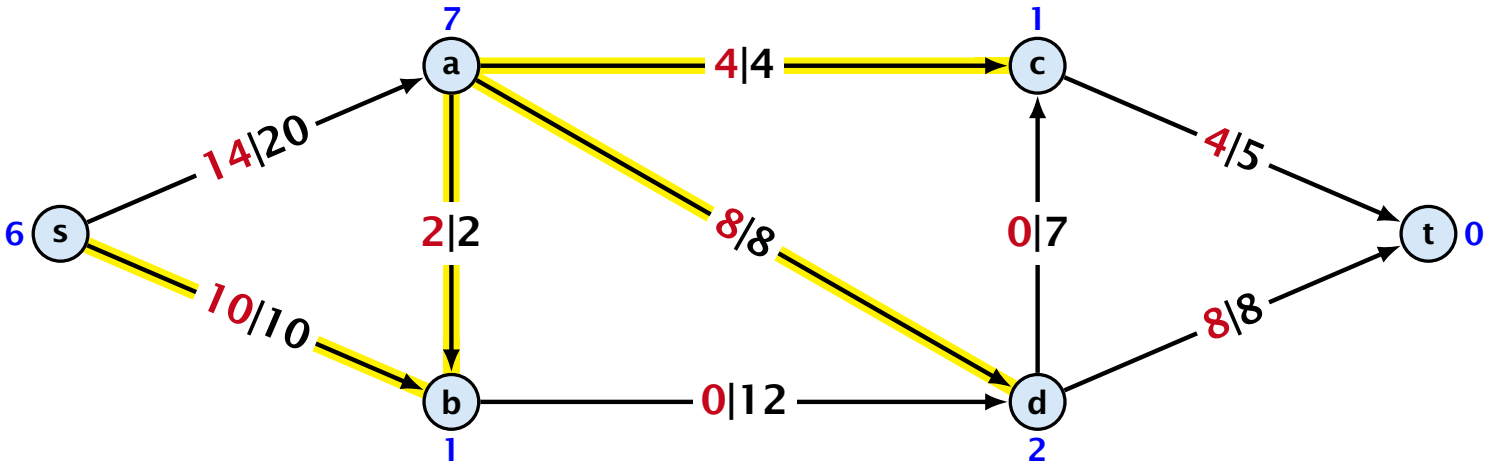


Preflow Push

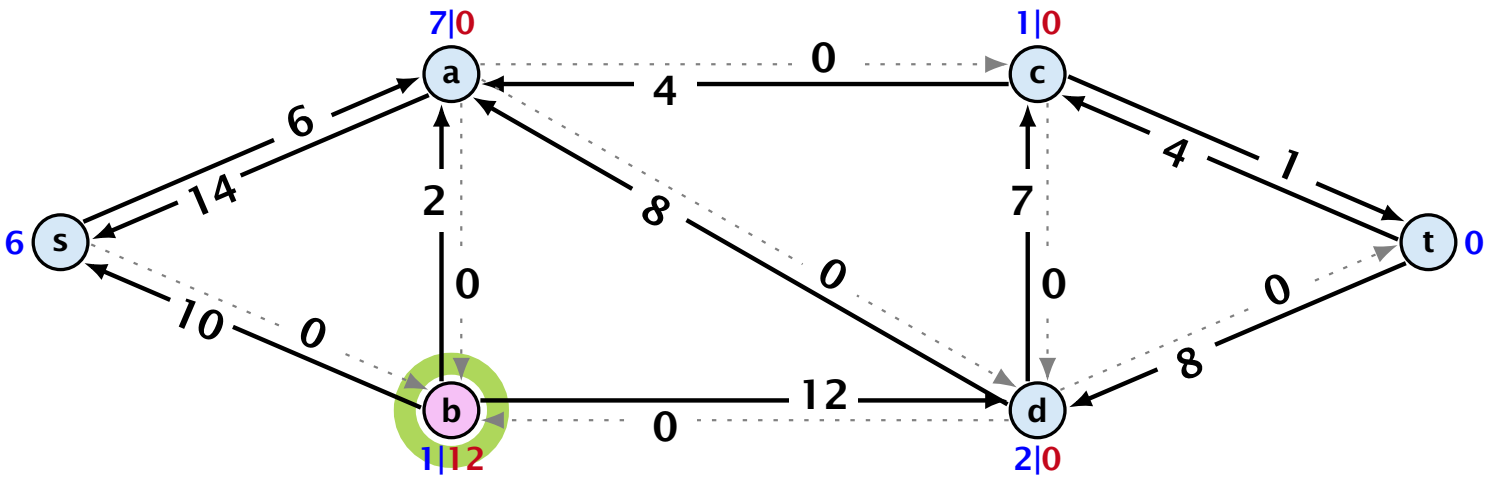




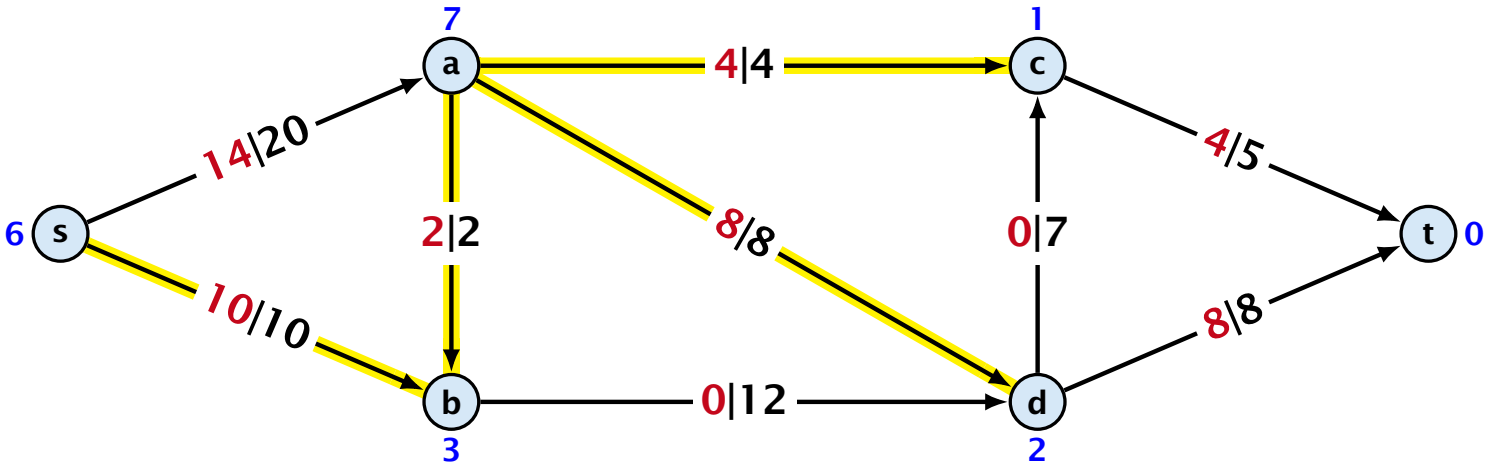
Preflow Push

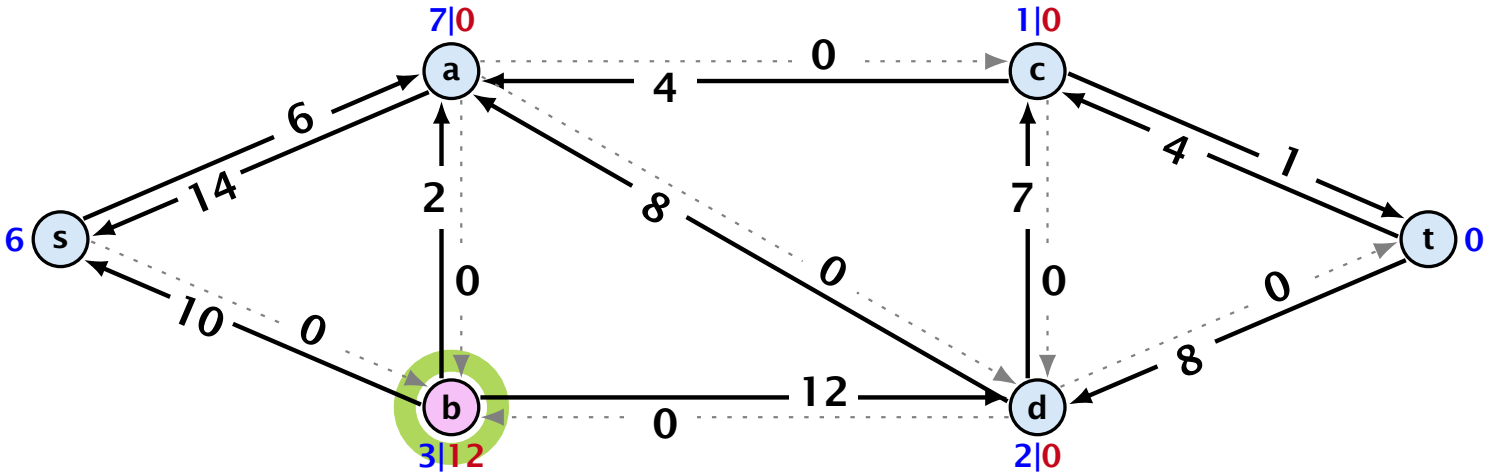


relabel to 3

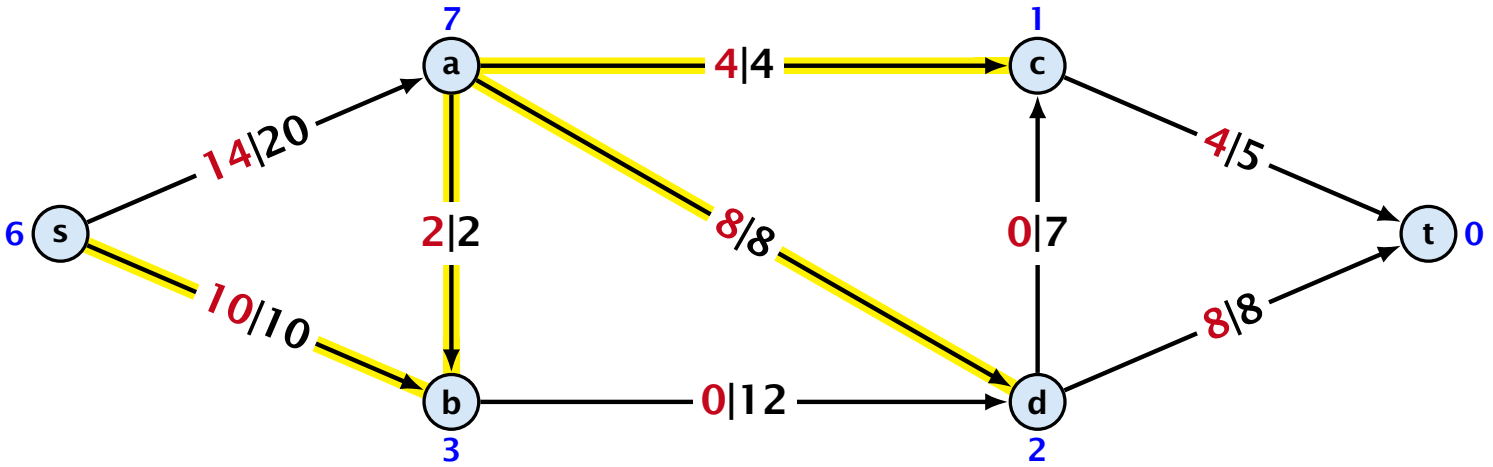


Preflow Push

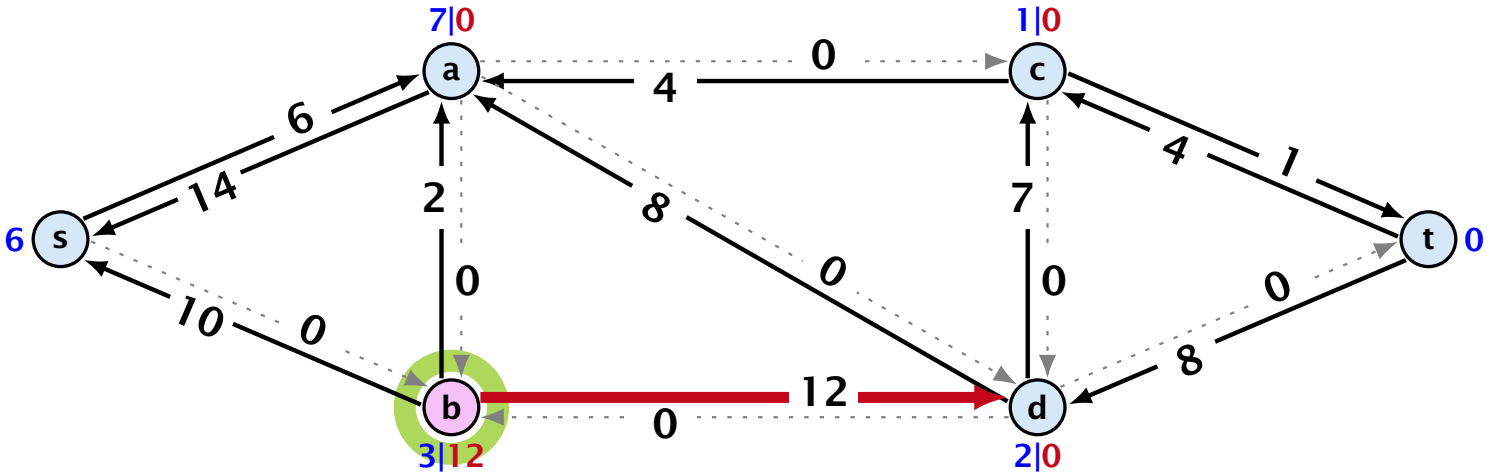




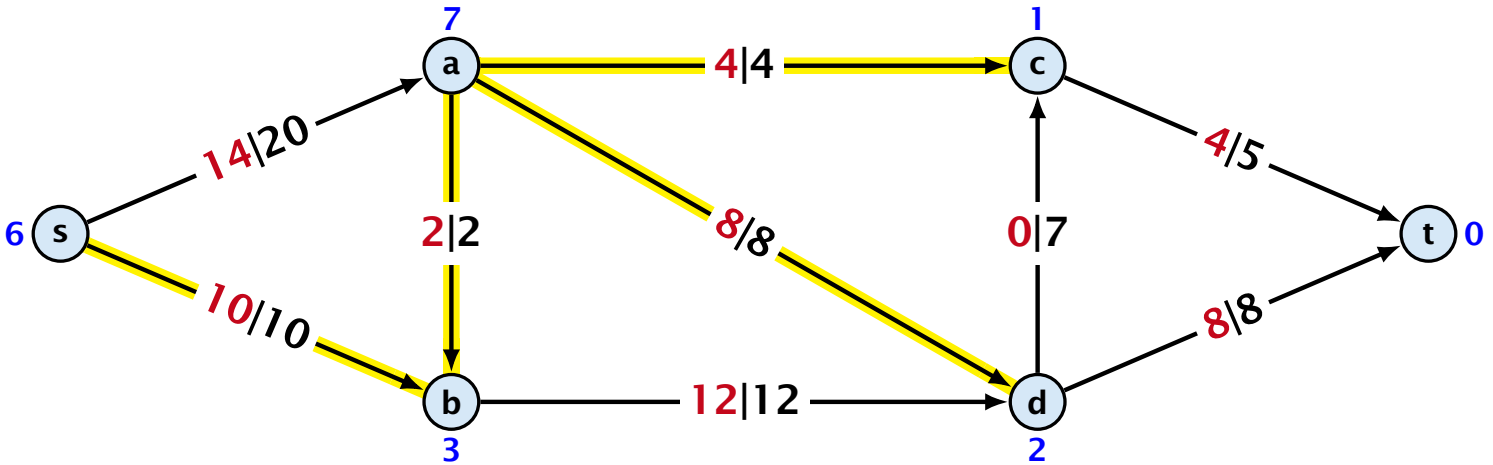
Preflow Push

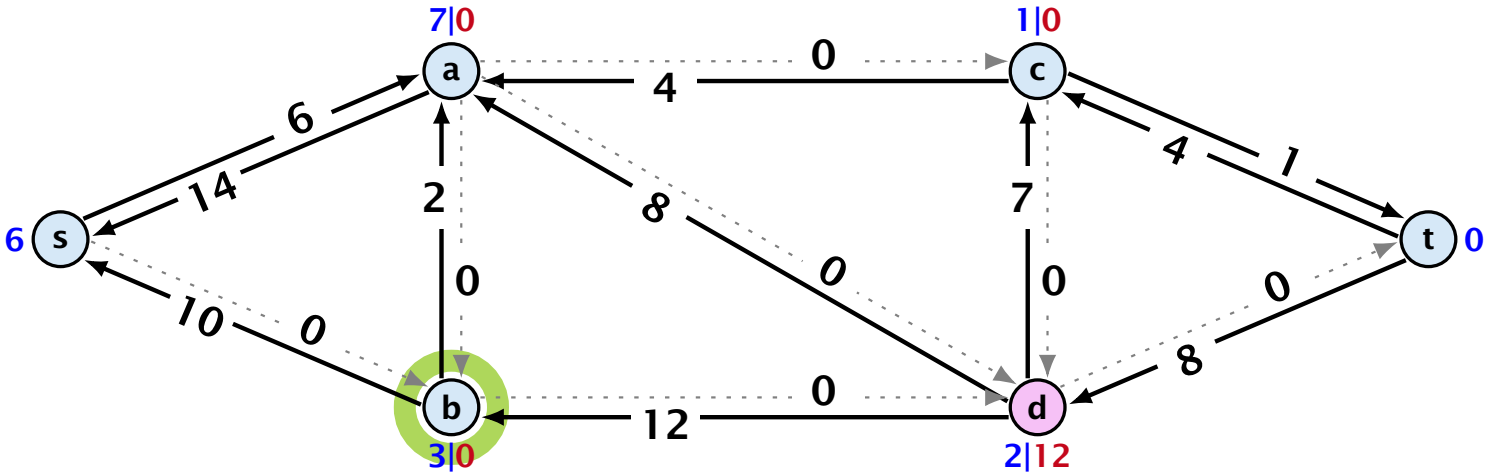


saturating and deactivating push

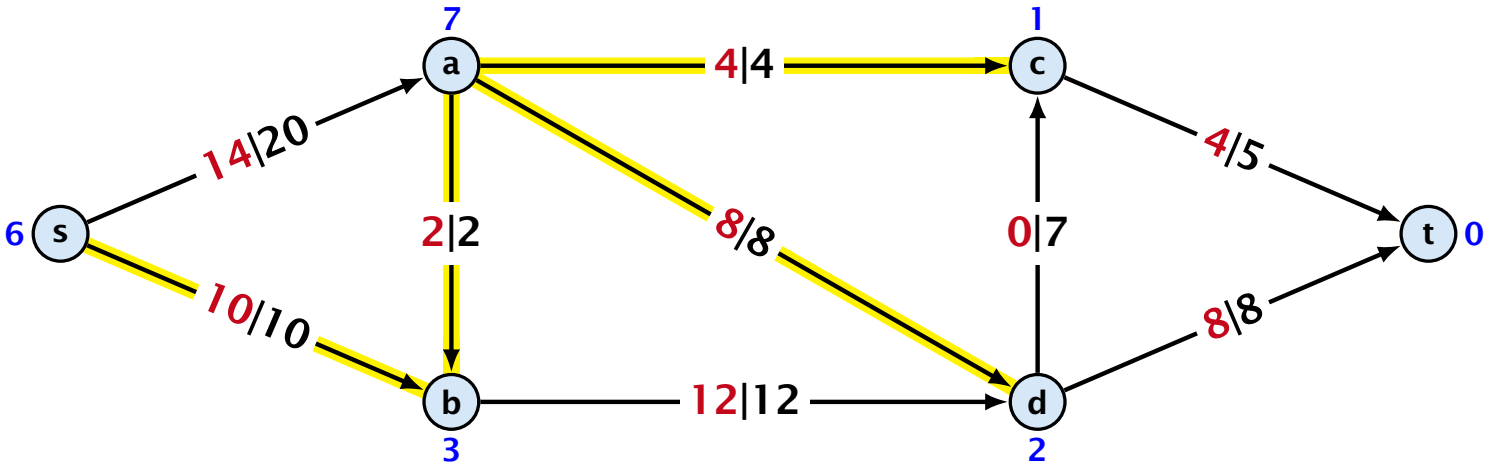


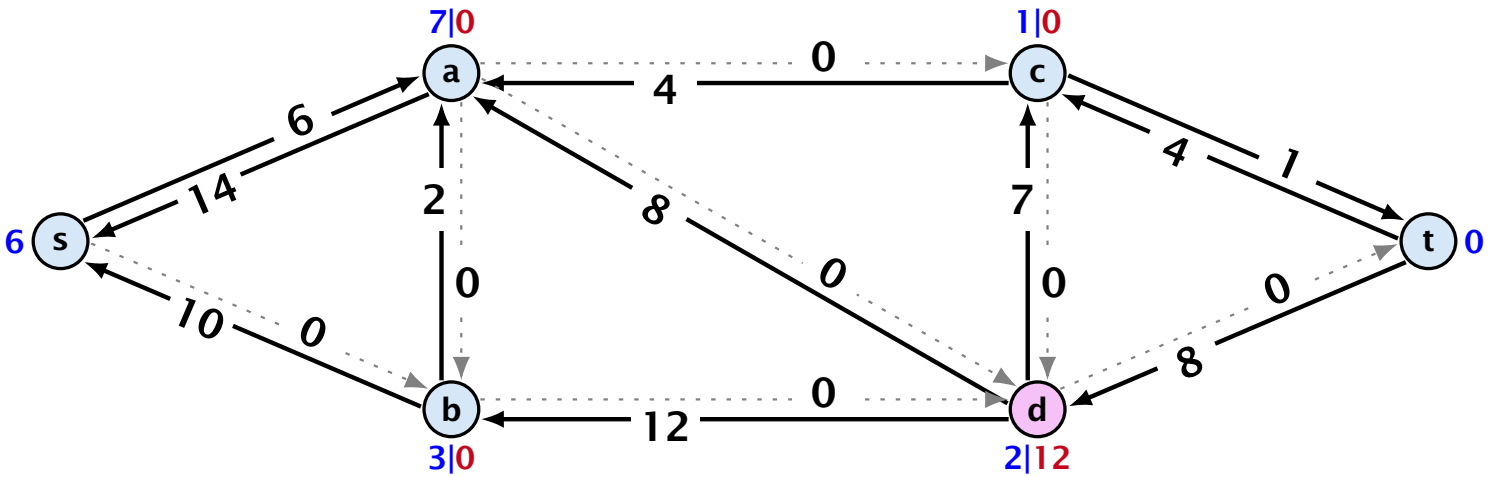
Preflow Push



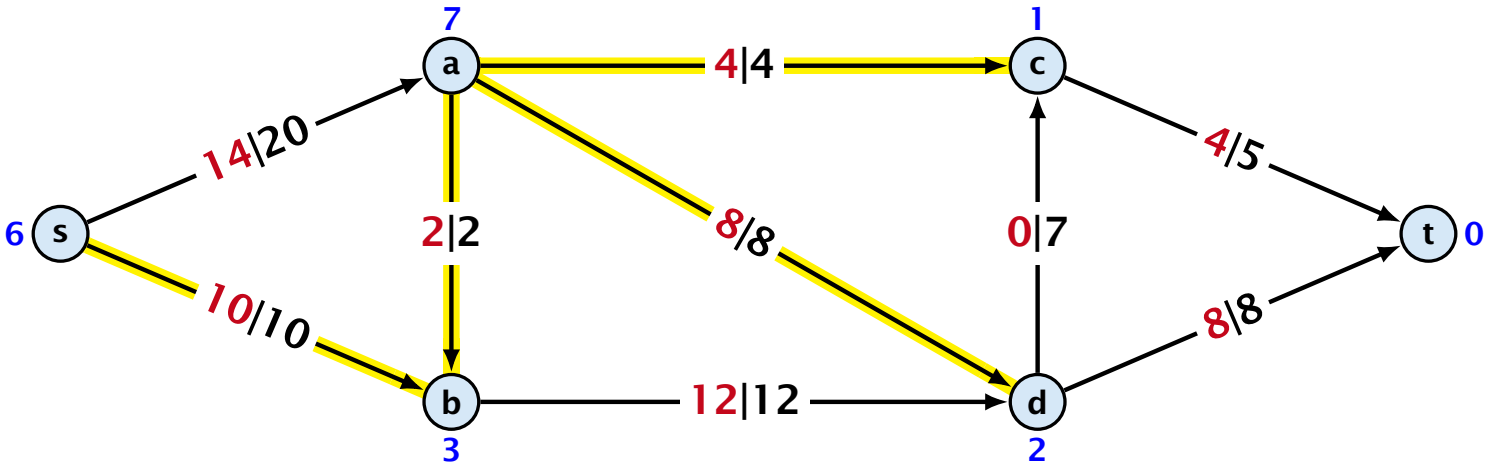


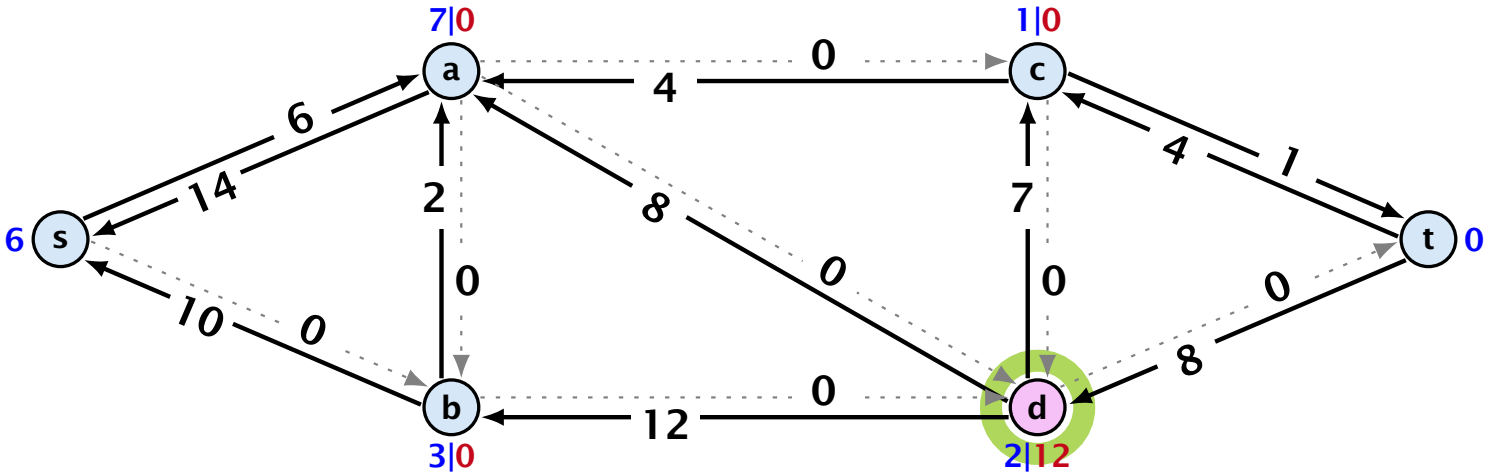
Preflow Push



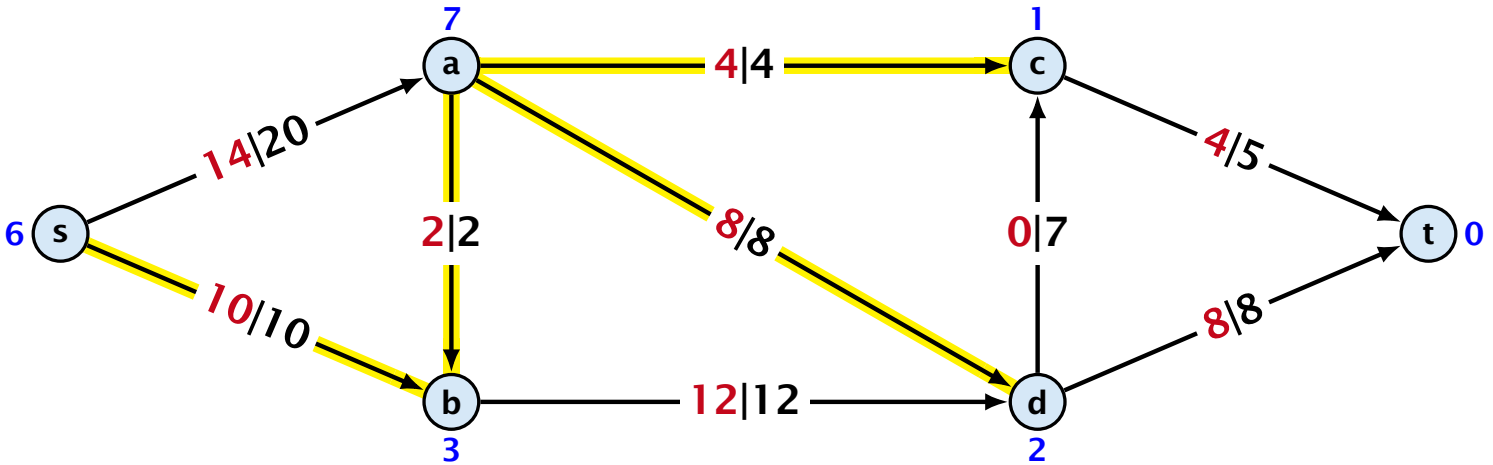


Preflow Push

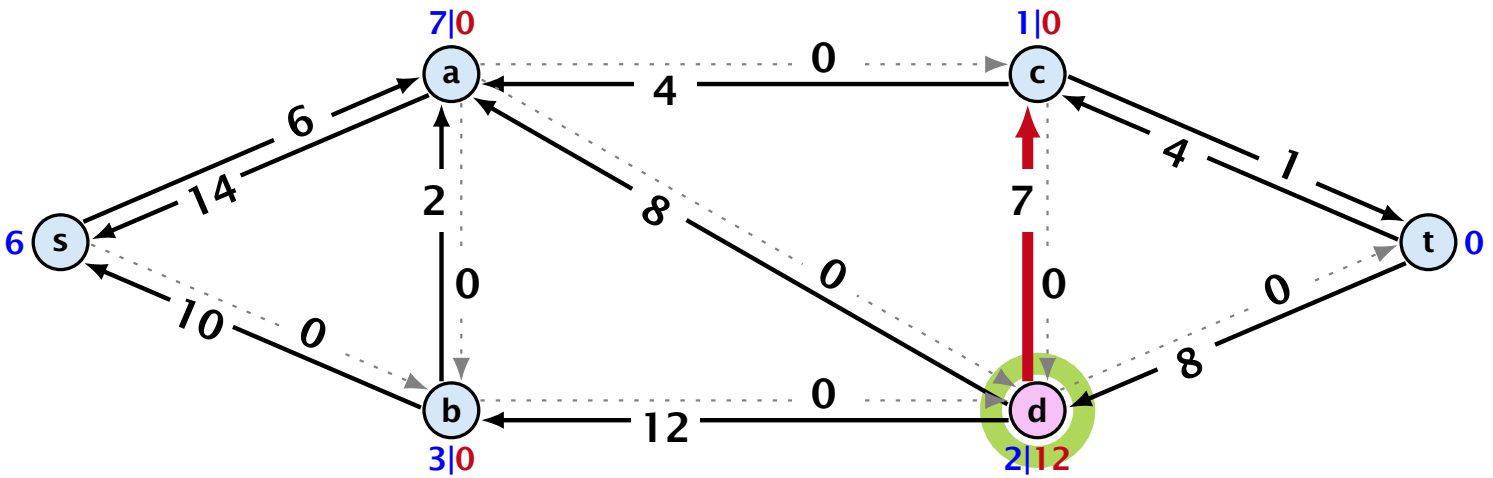




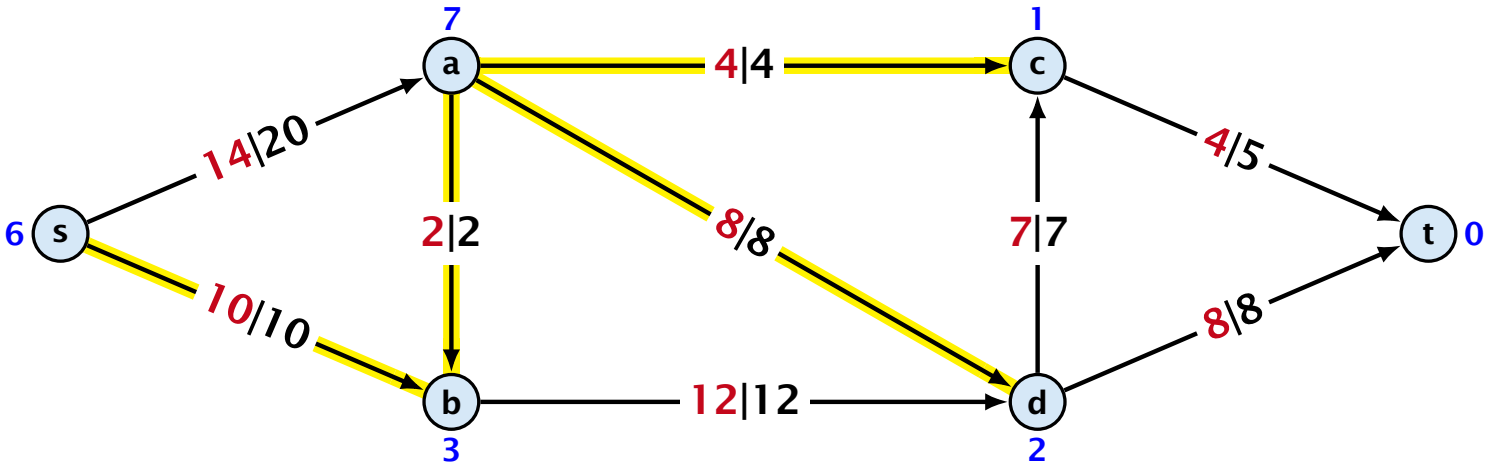
Preflow Push

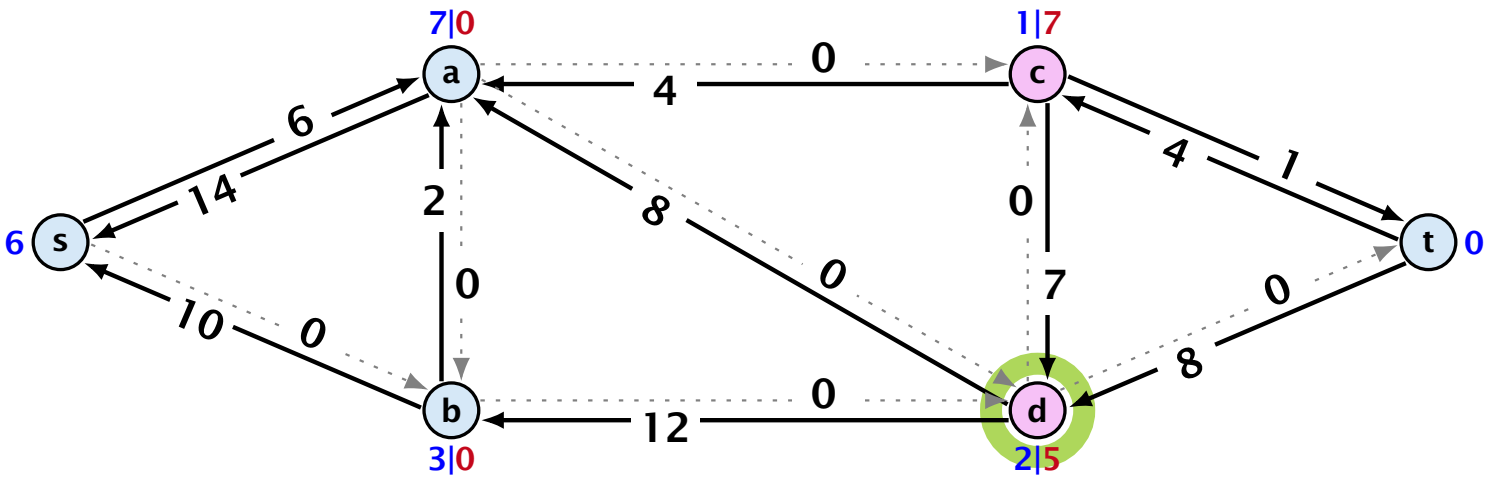


saturating push

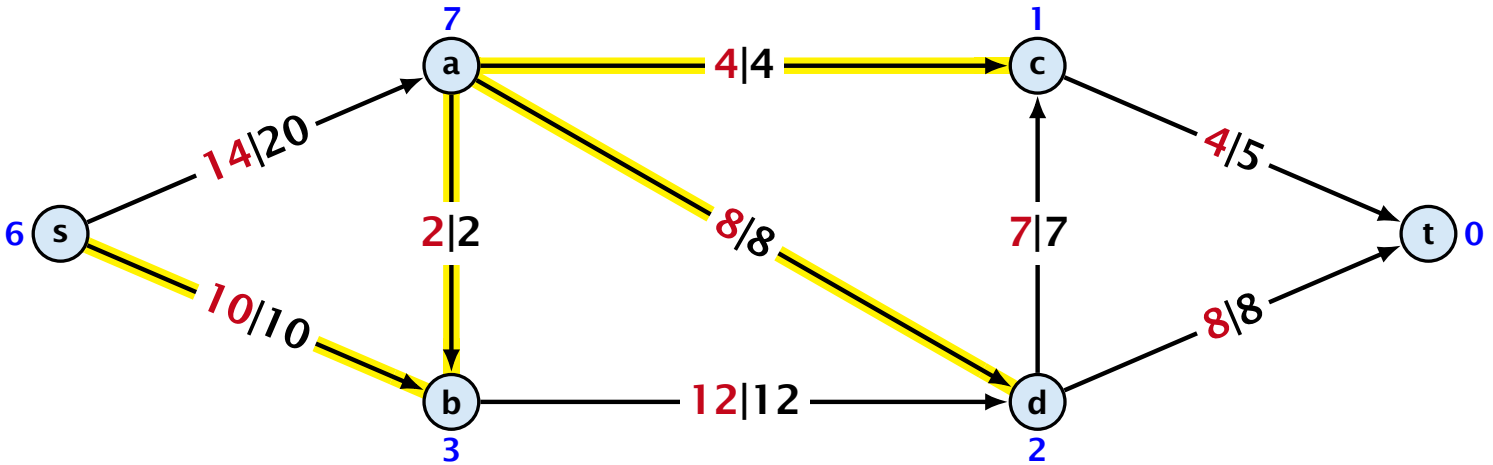


Preflow Push

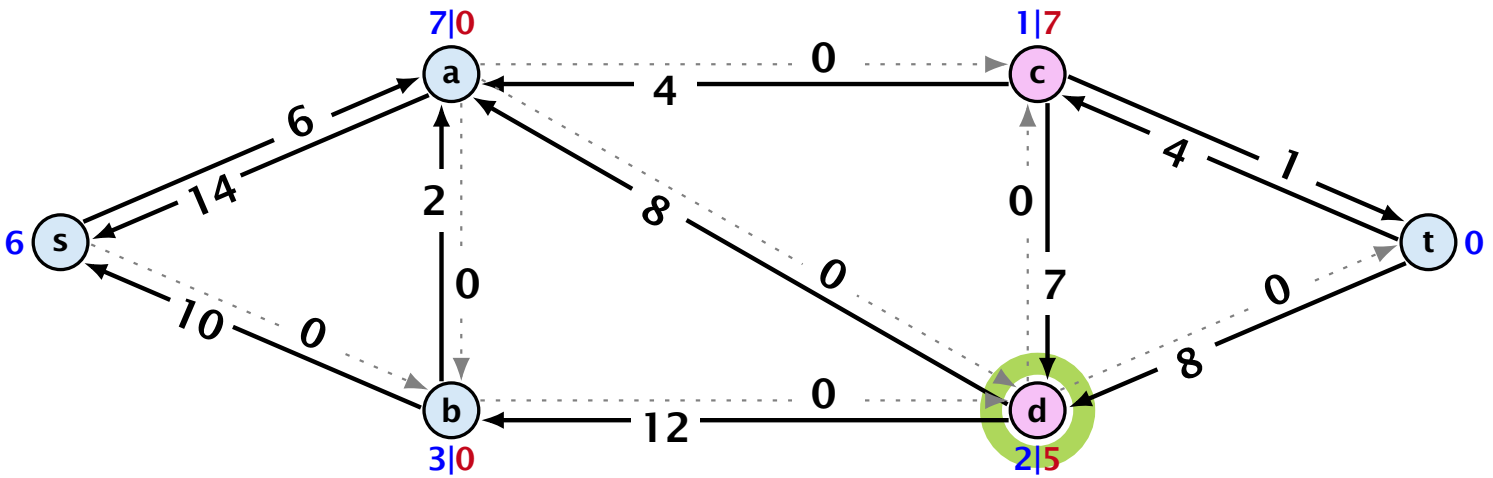




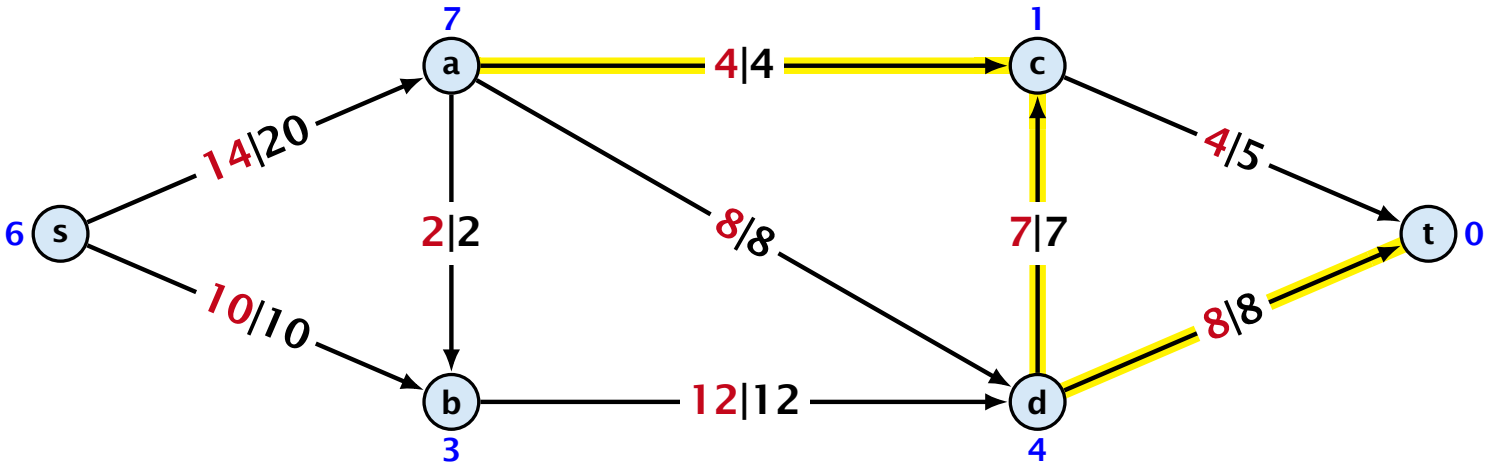
Preflow Push

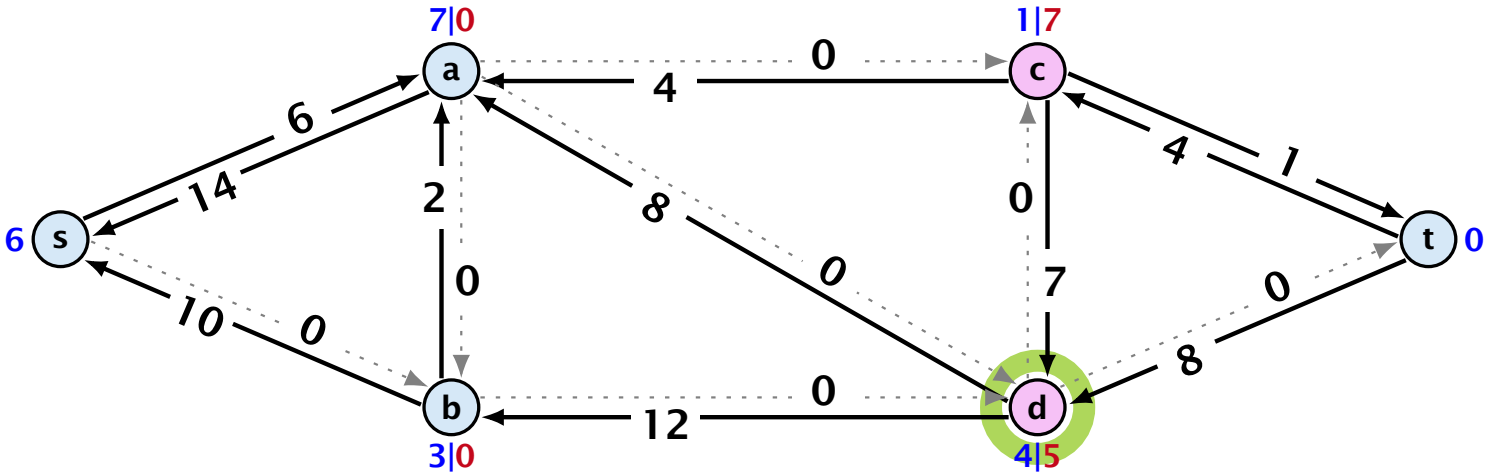


relabel to 4

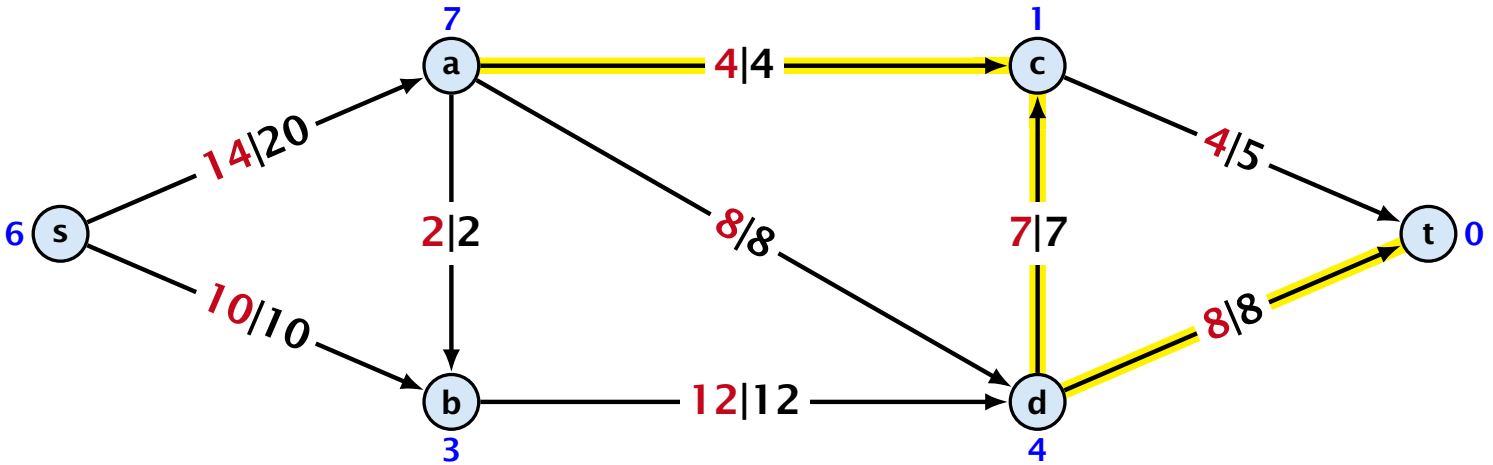


Preflow Push

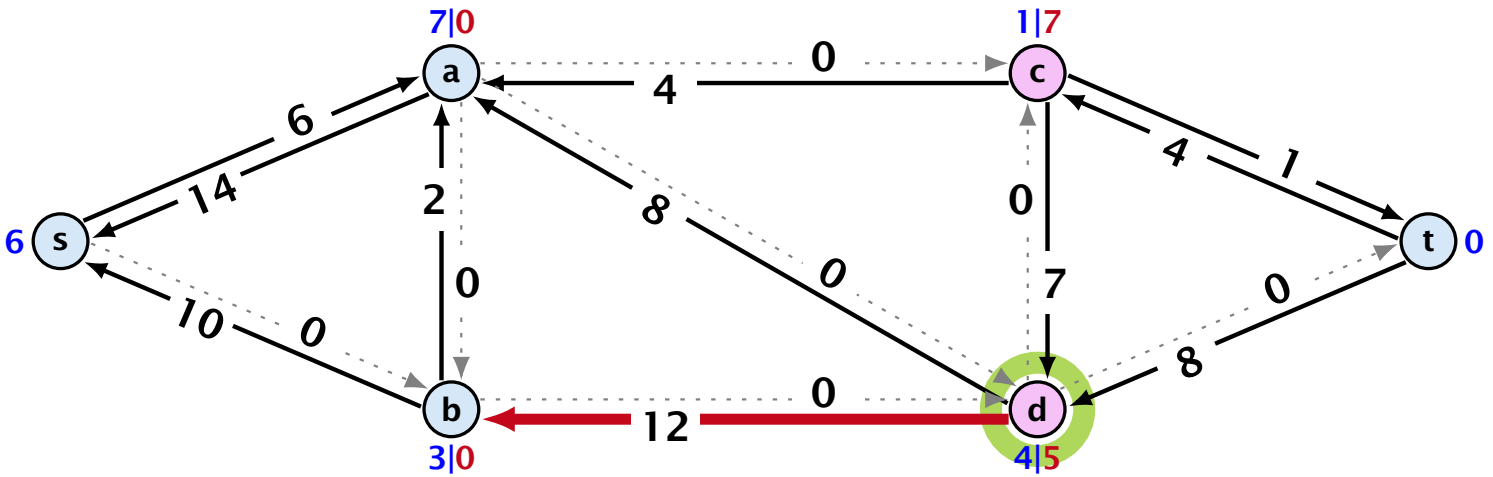




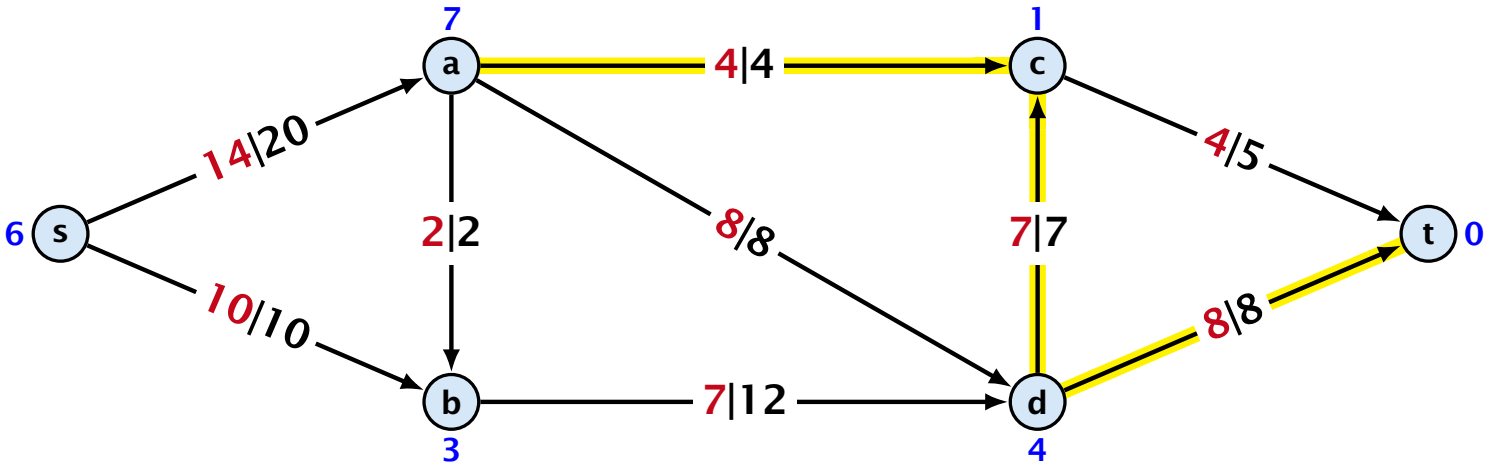
Preflow Push

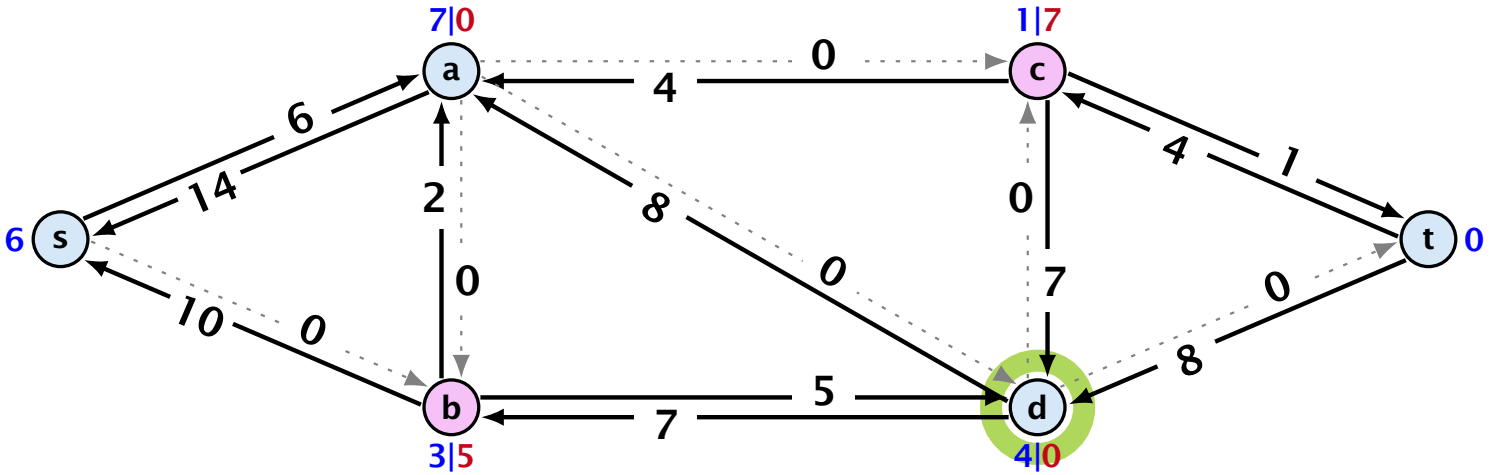


deactivating push

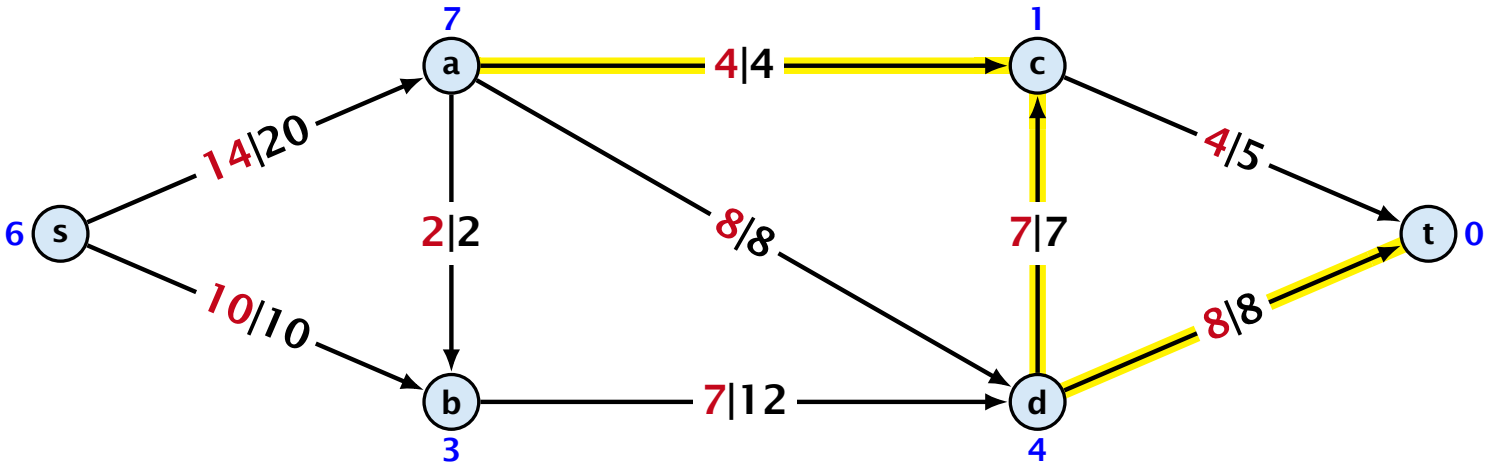


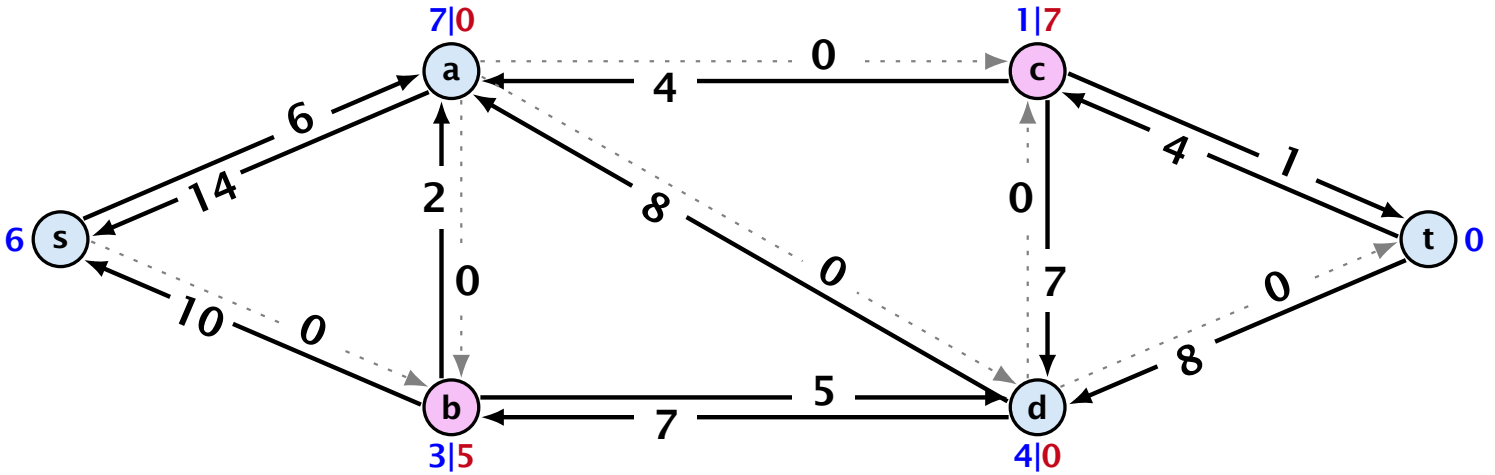
Preflow Push



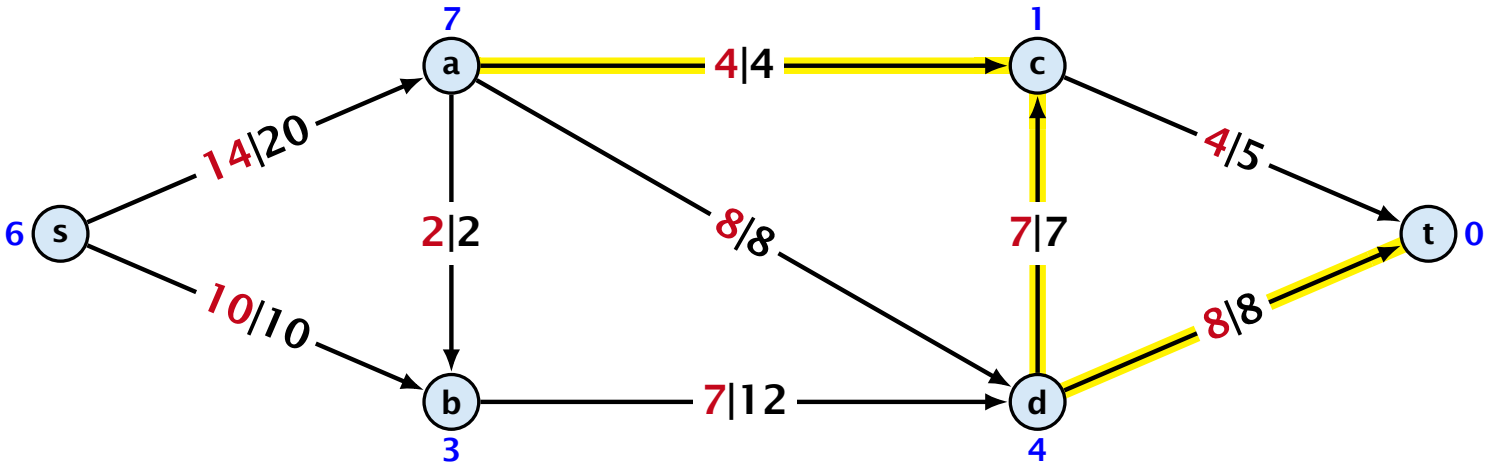


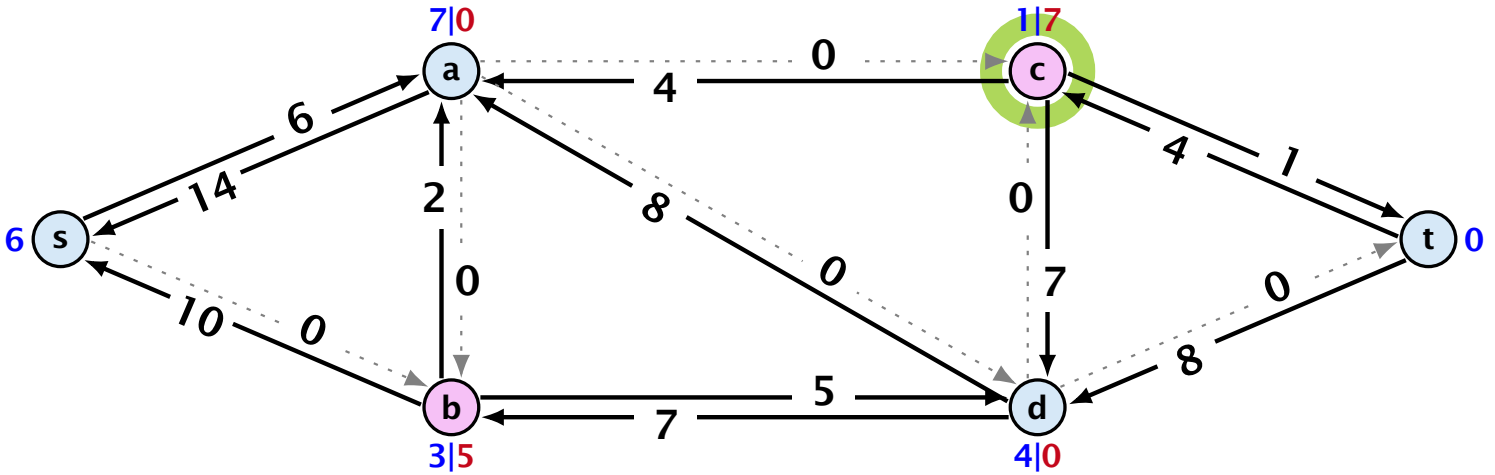
Preflow Push



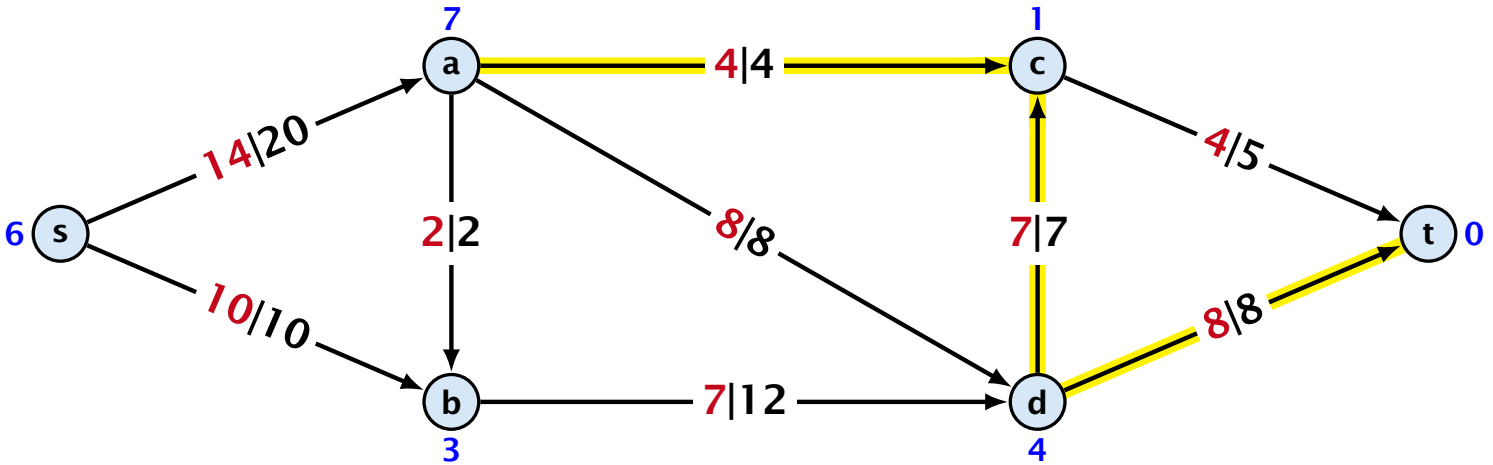


Preflow Push

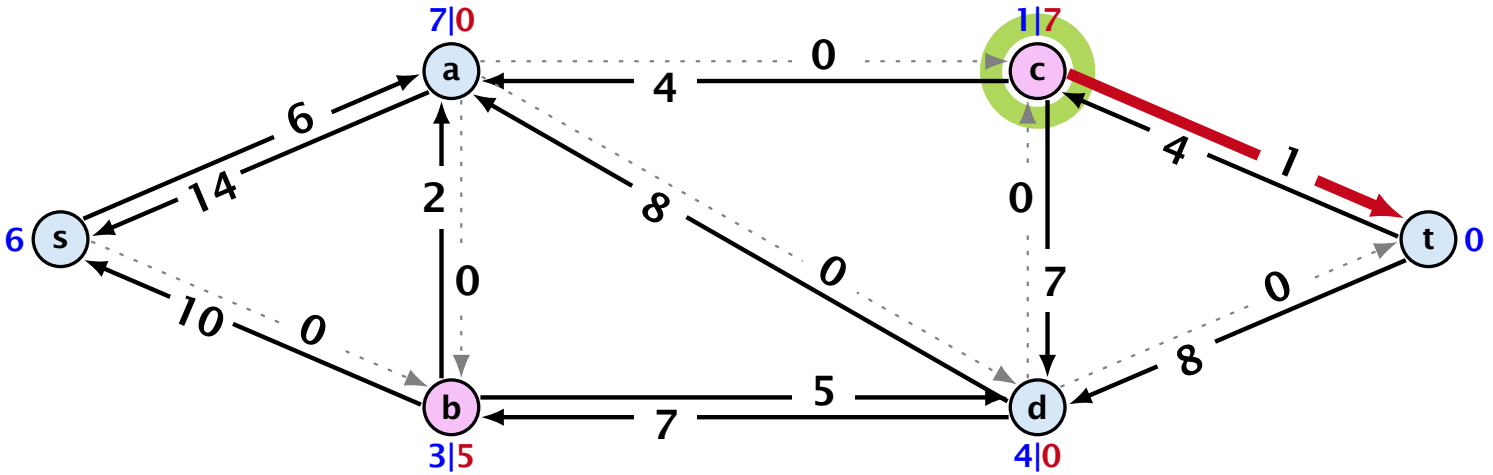




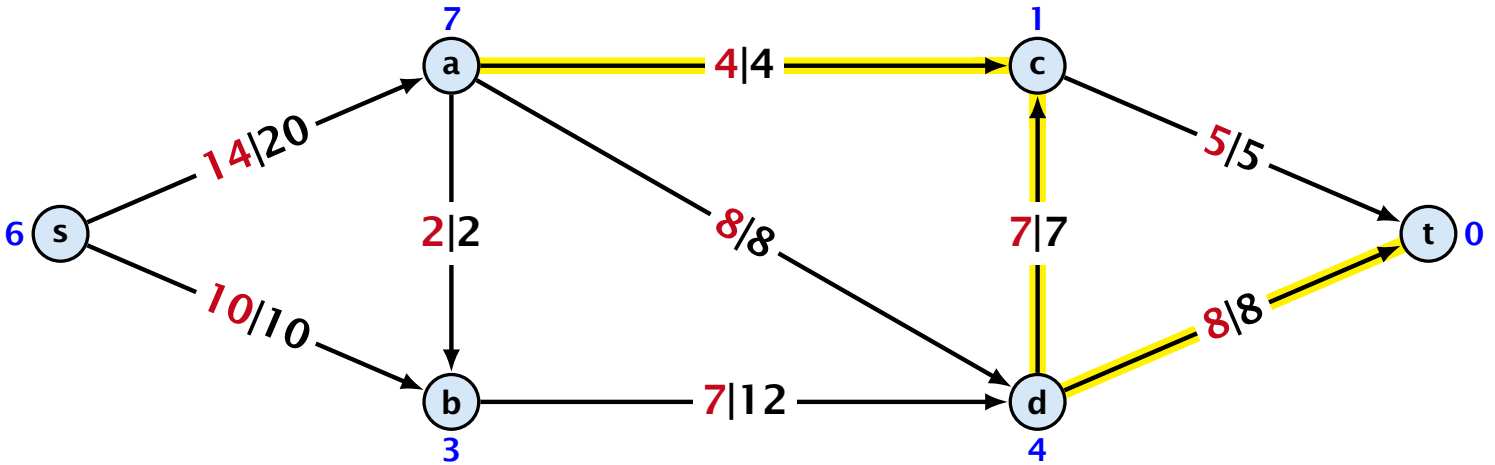
Preflow Push

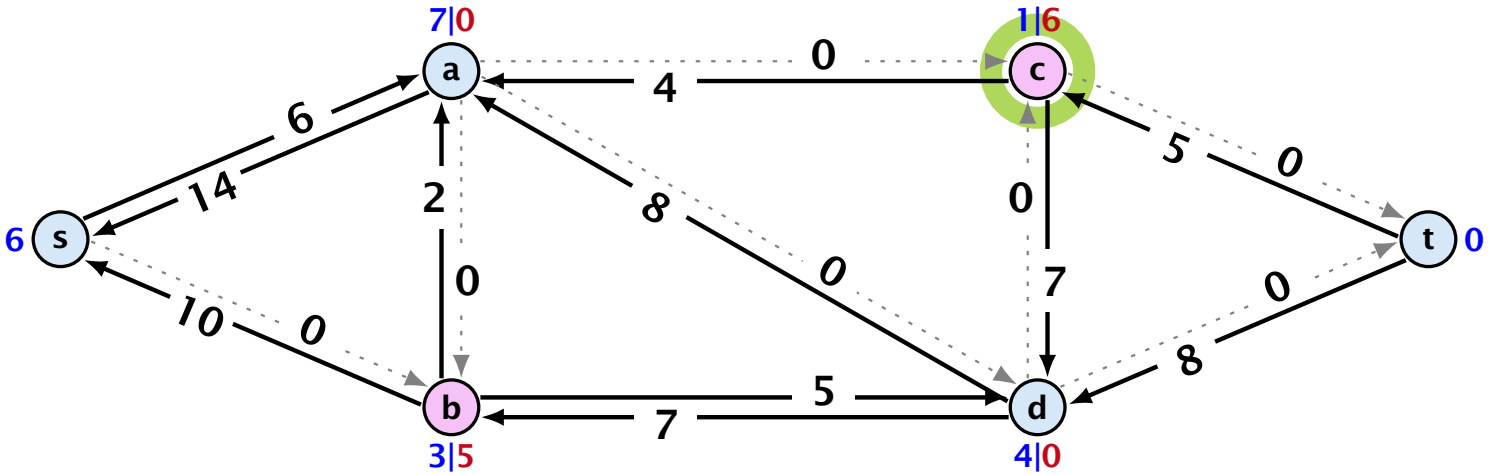


saturating push

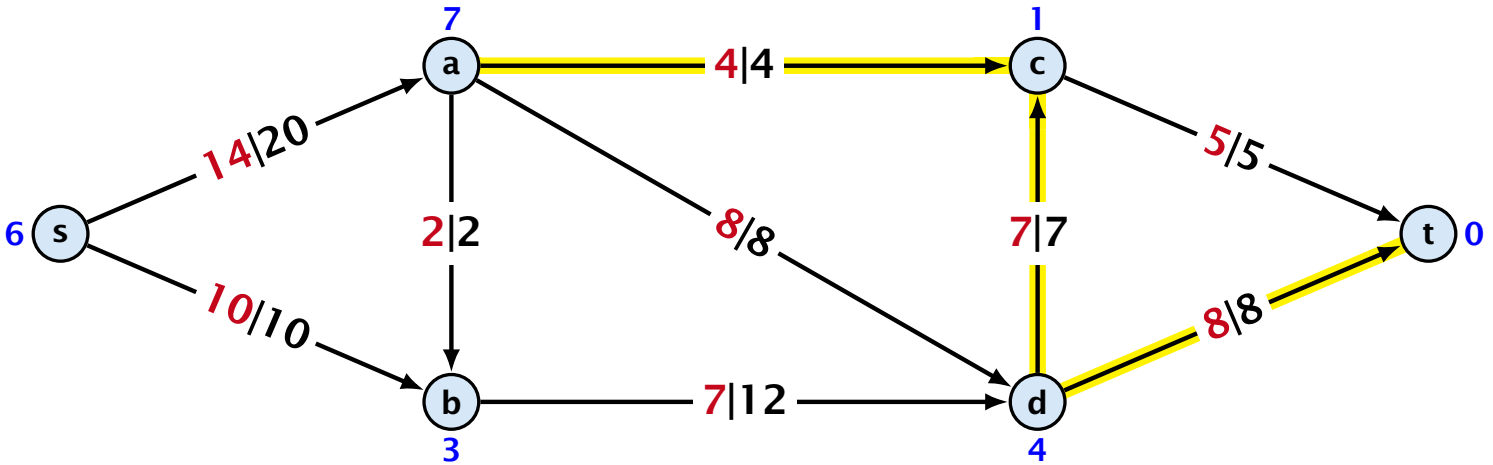


Preflow Push

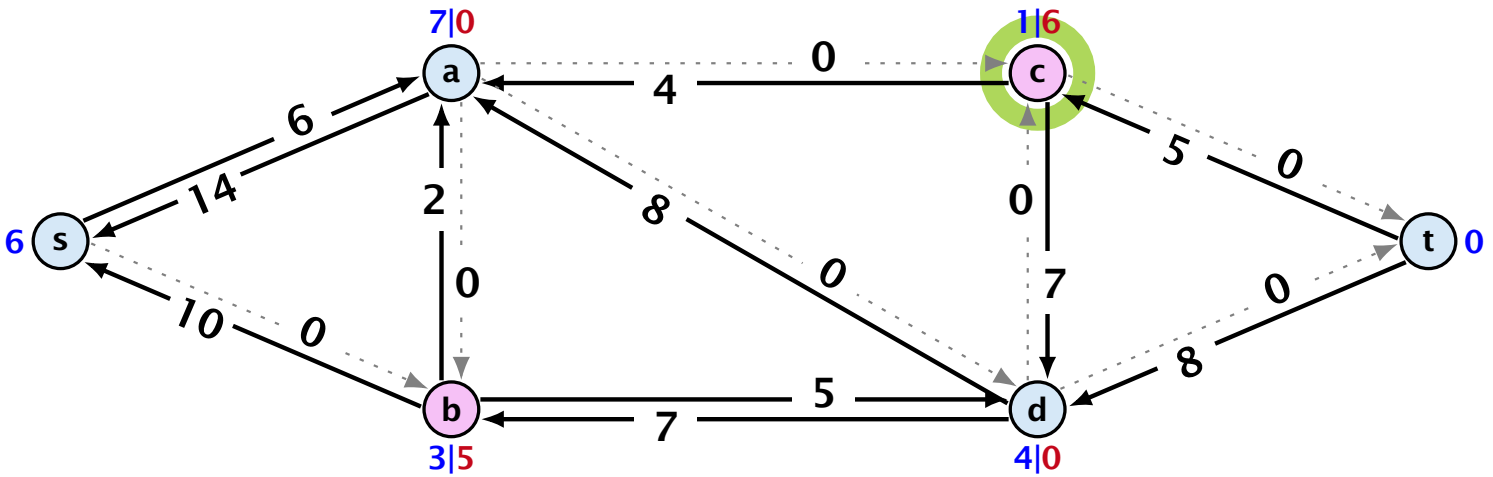




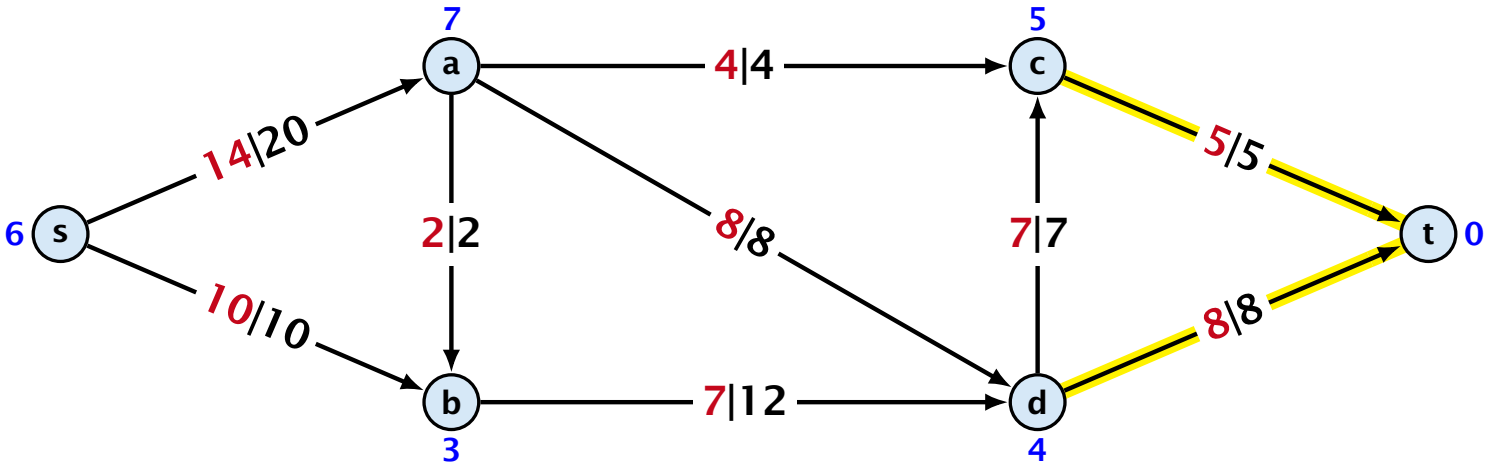
Preflow Push

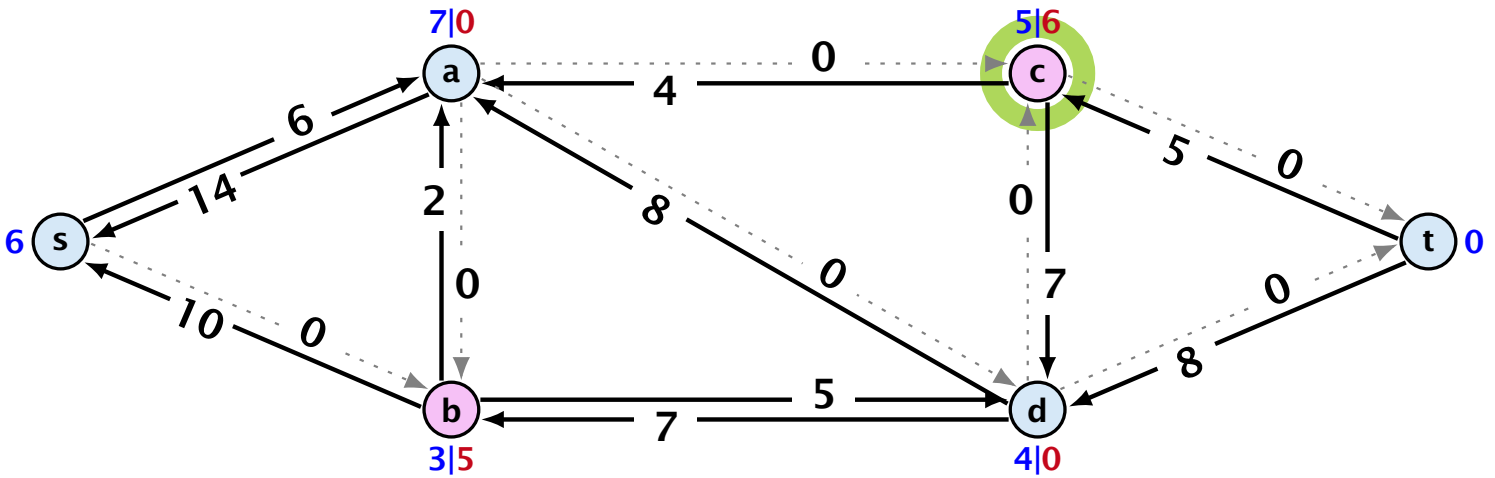


relabel to 5

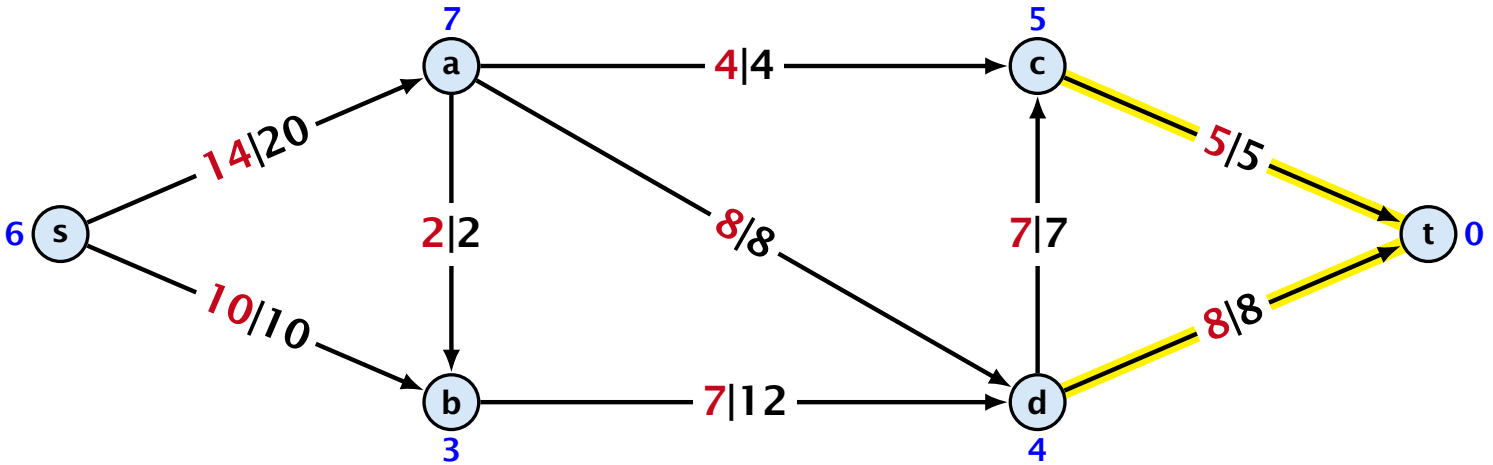


Preflow Push

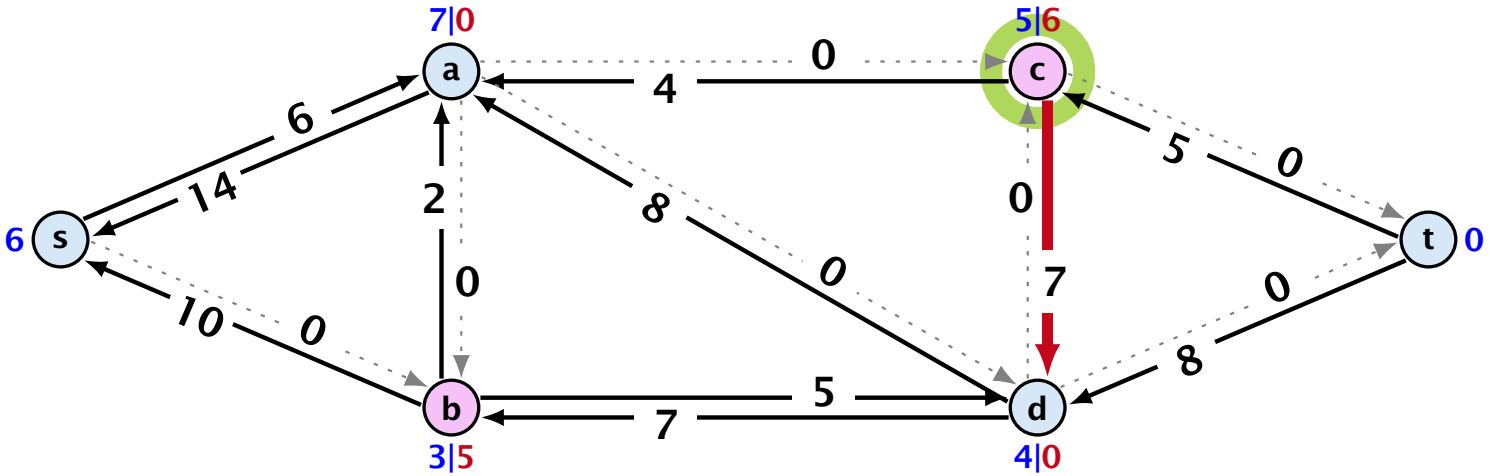




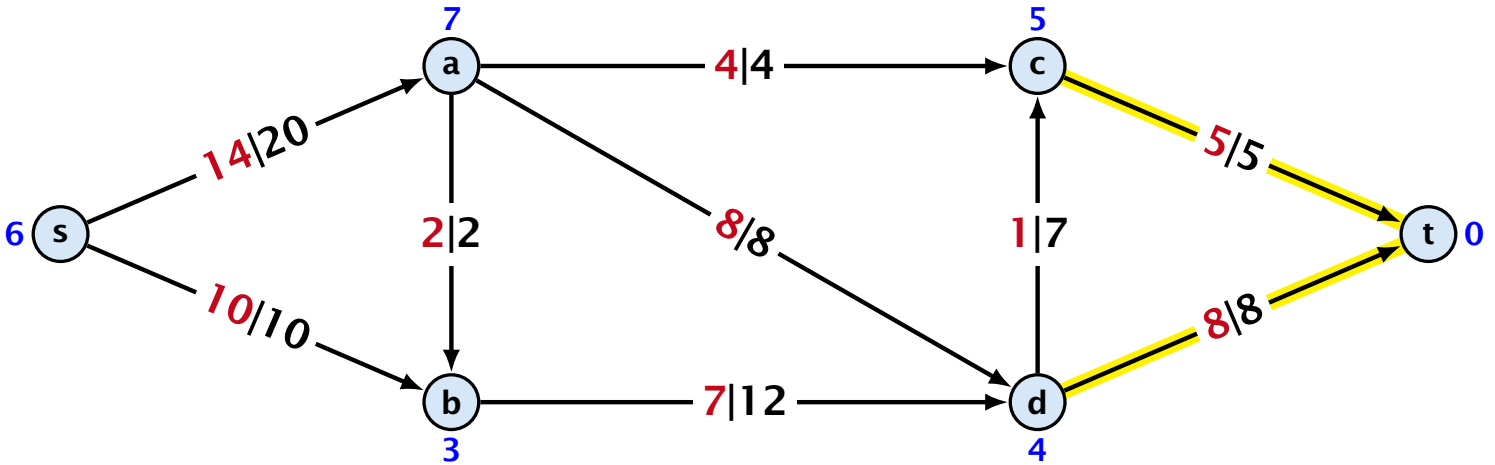
Preflow Push

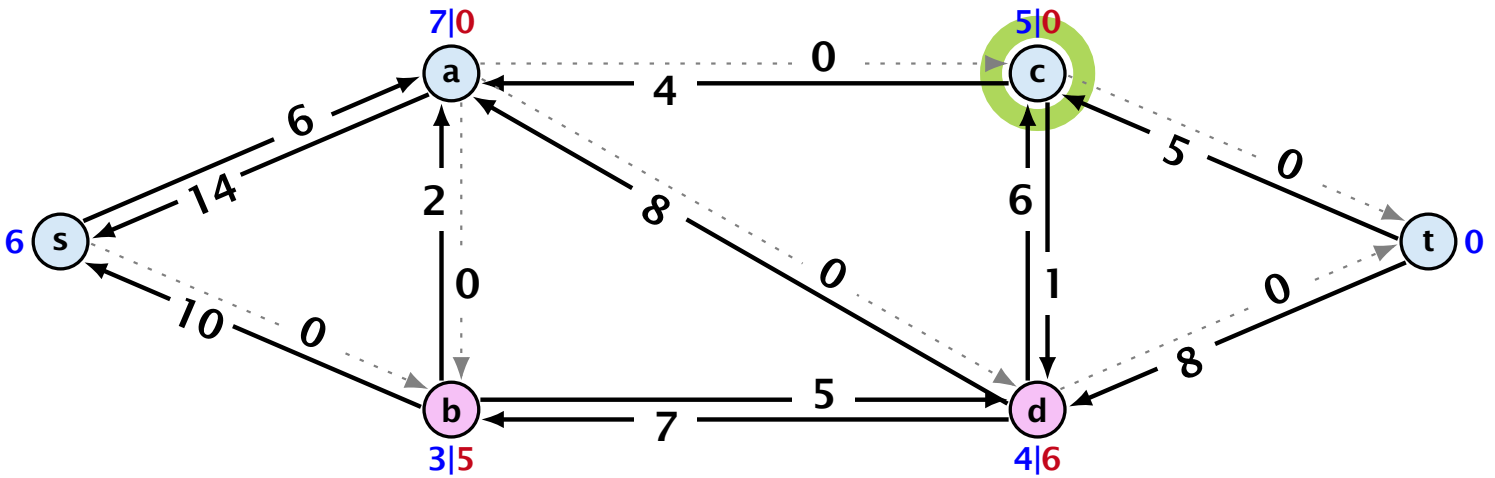


deactivating push

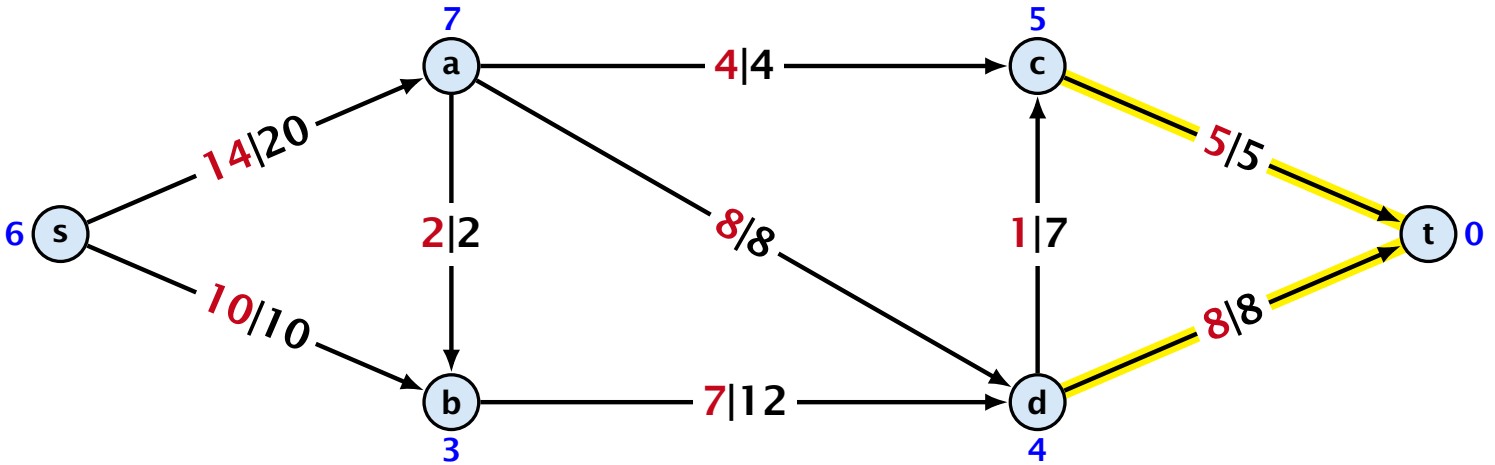


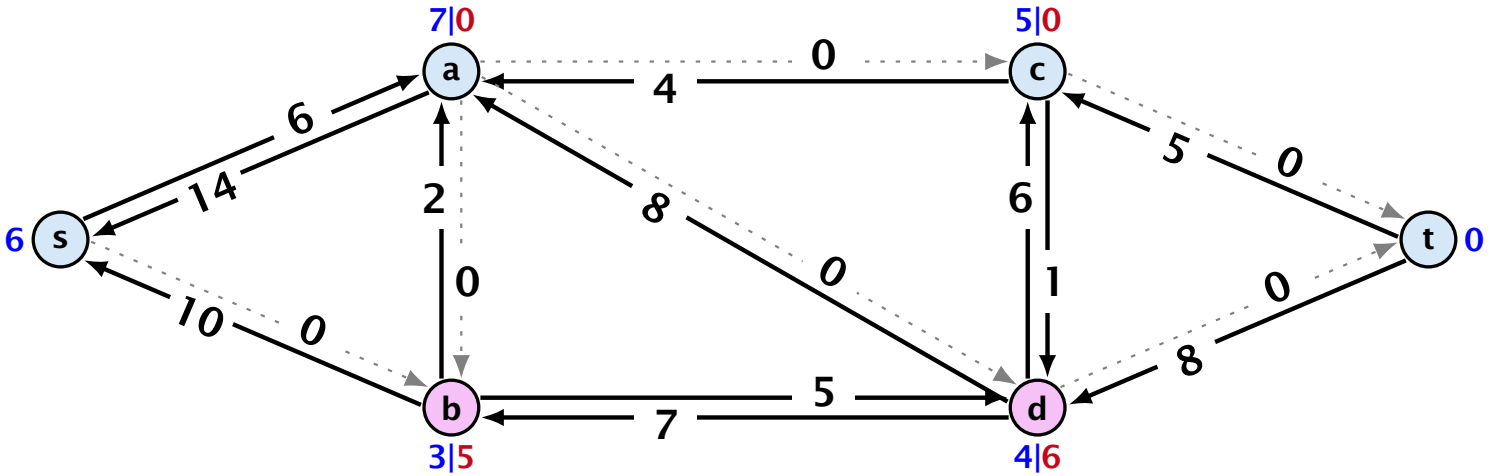
Preflow Push



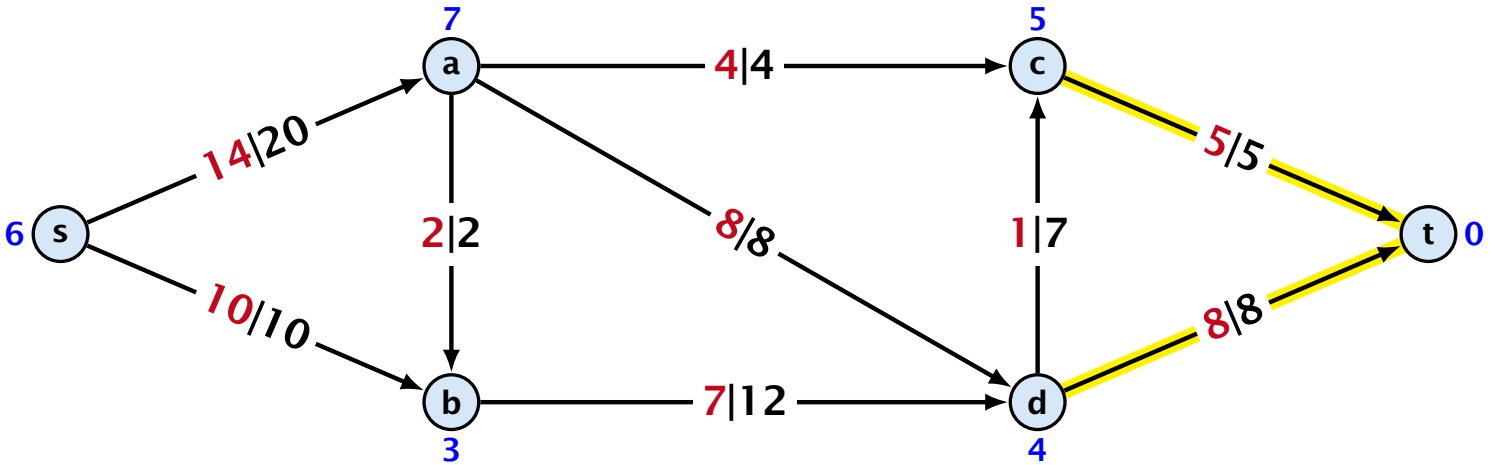


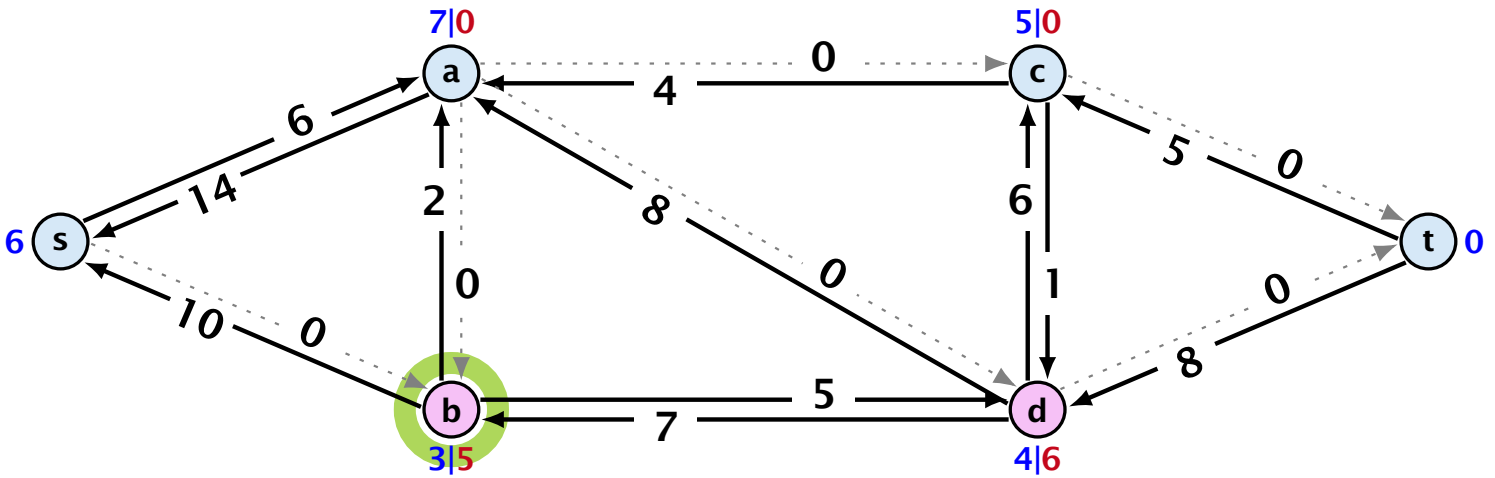
Preflow Push



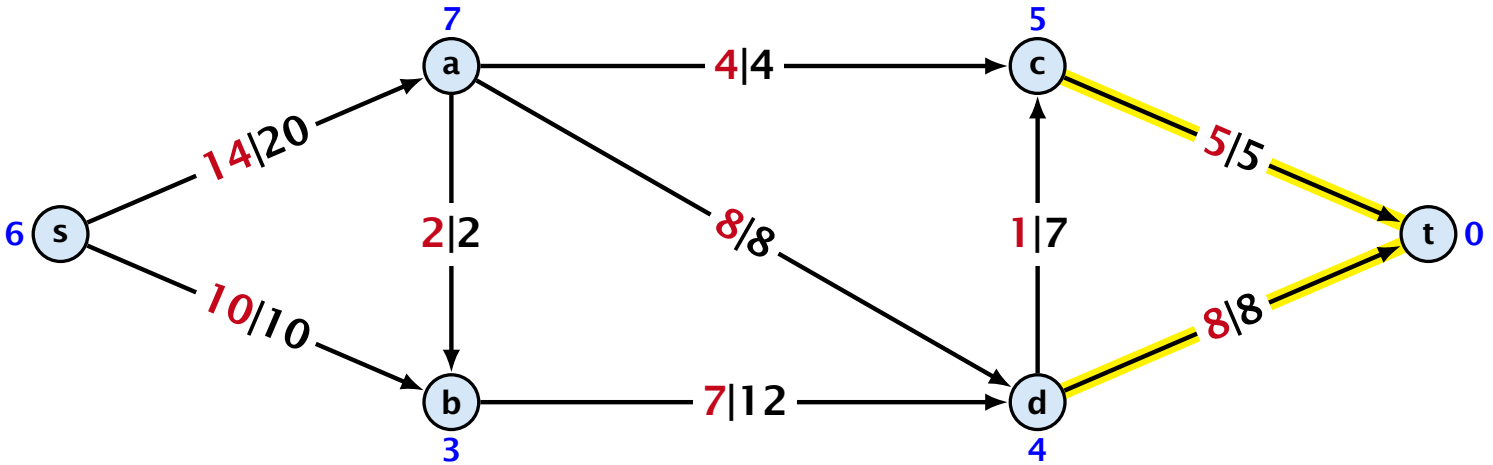


Preflow Push

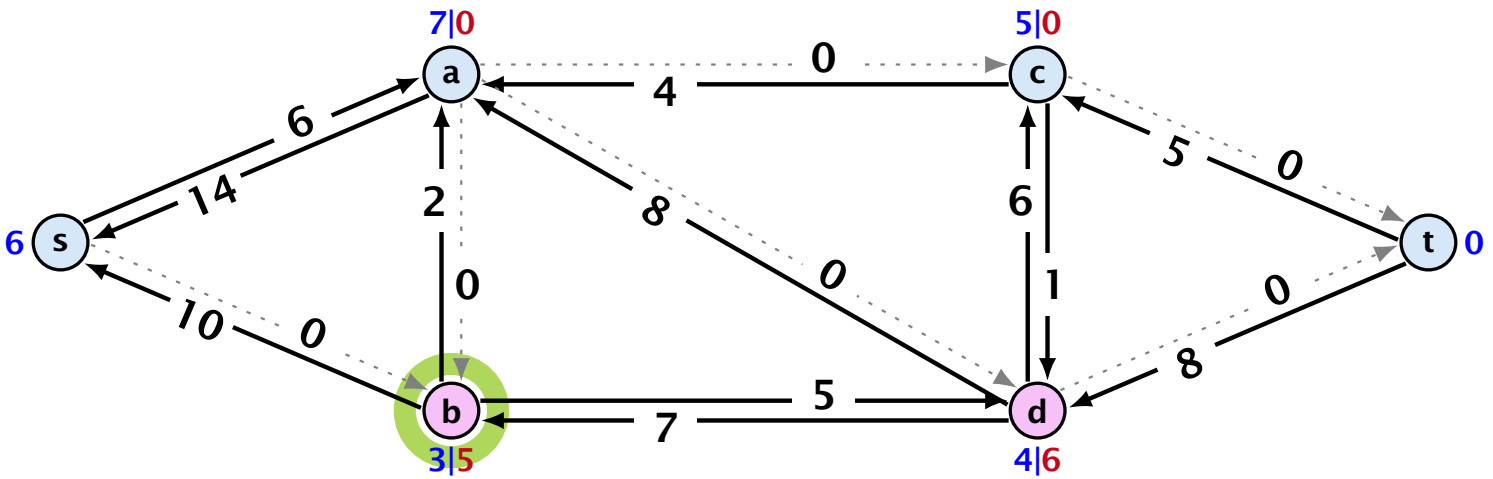




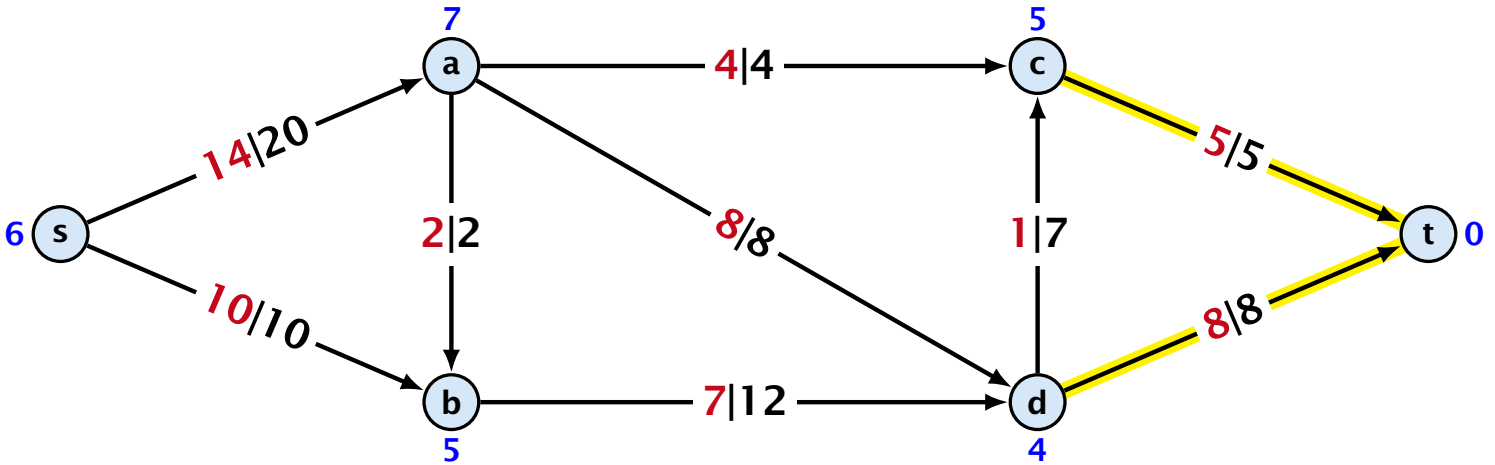
Preflow Push

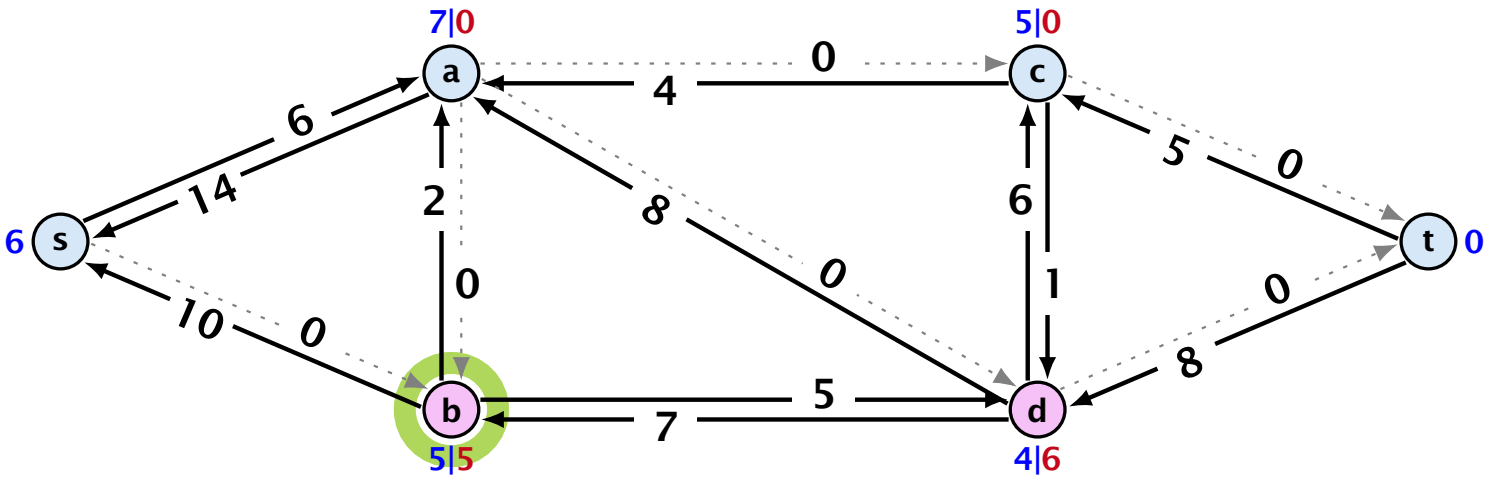


relabel to 5

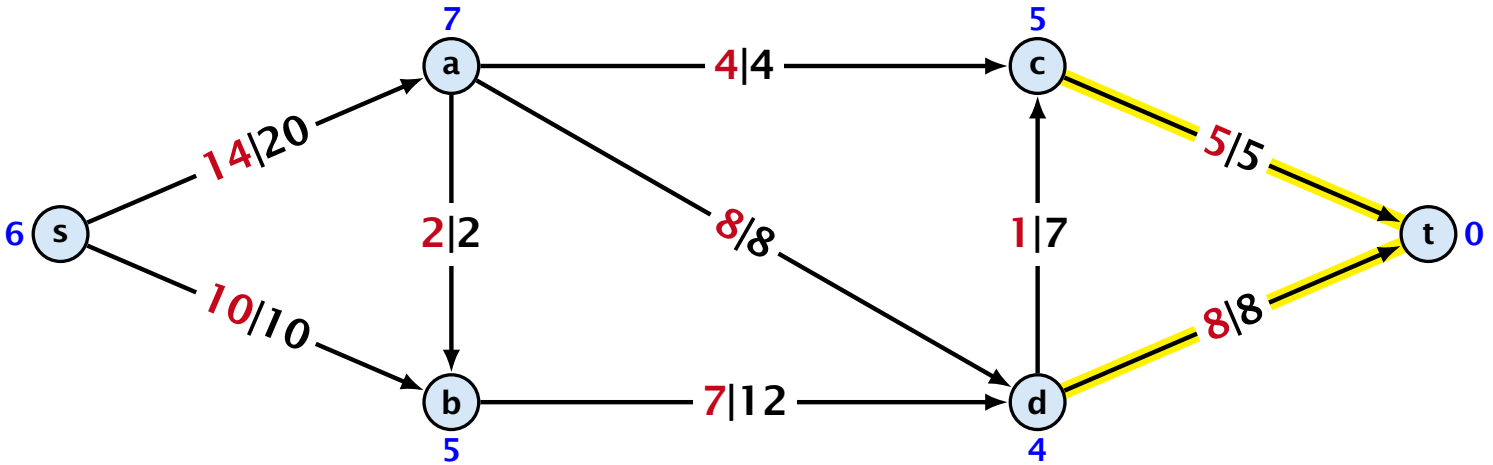


Preflow Push

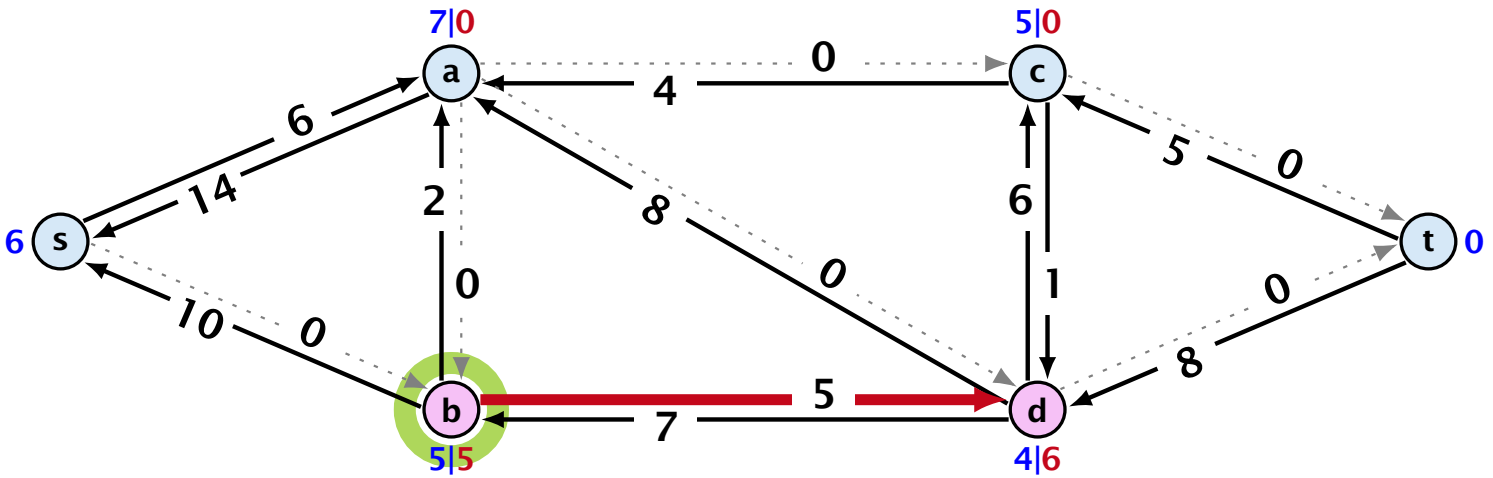




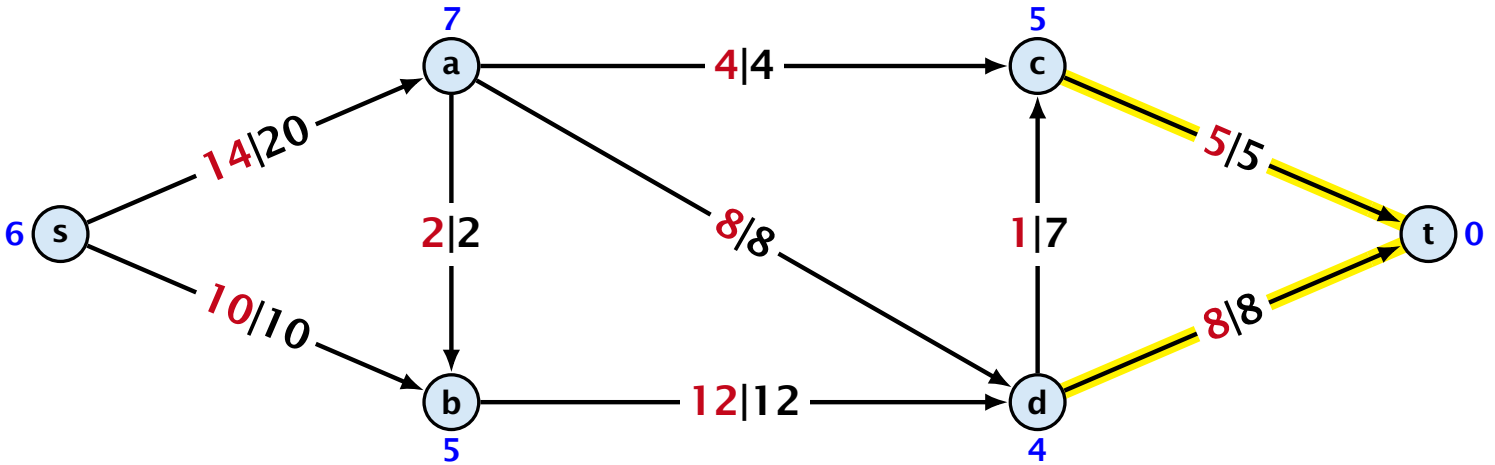
Preflow Push

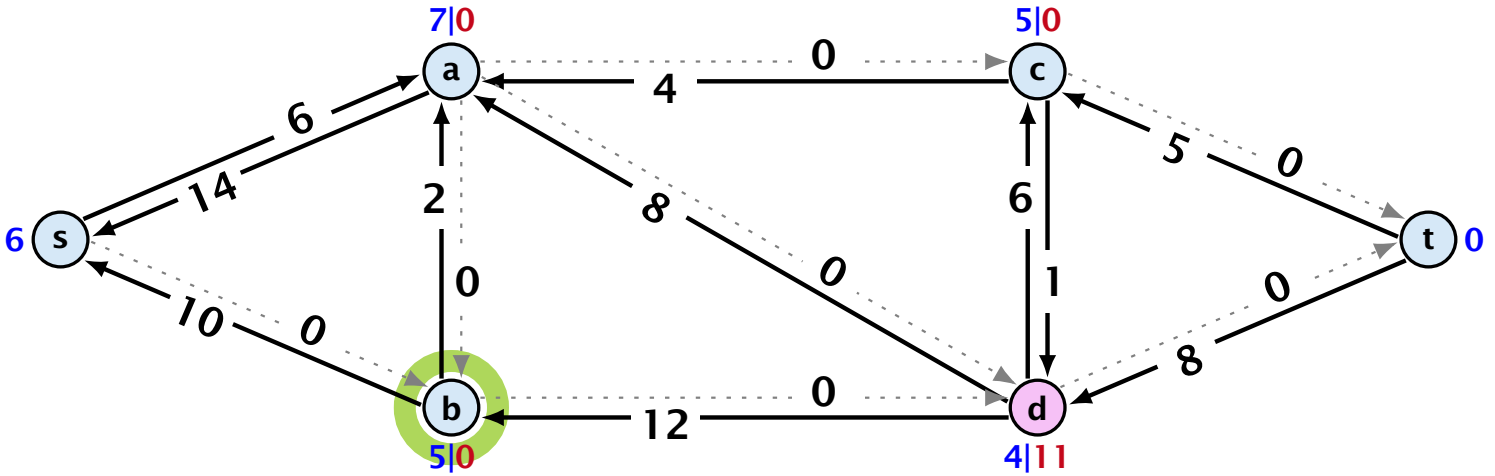


saturating and deactivating push

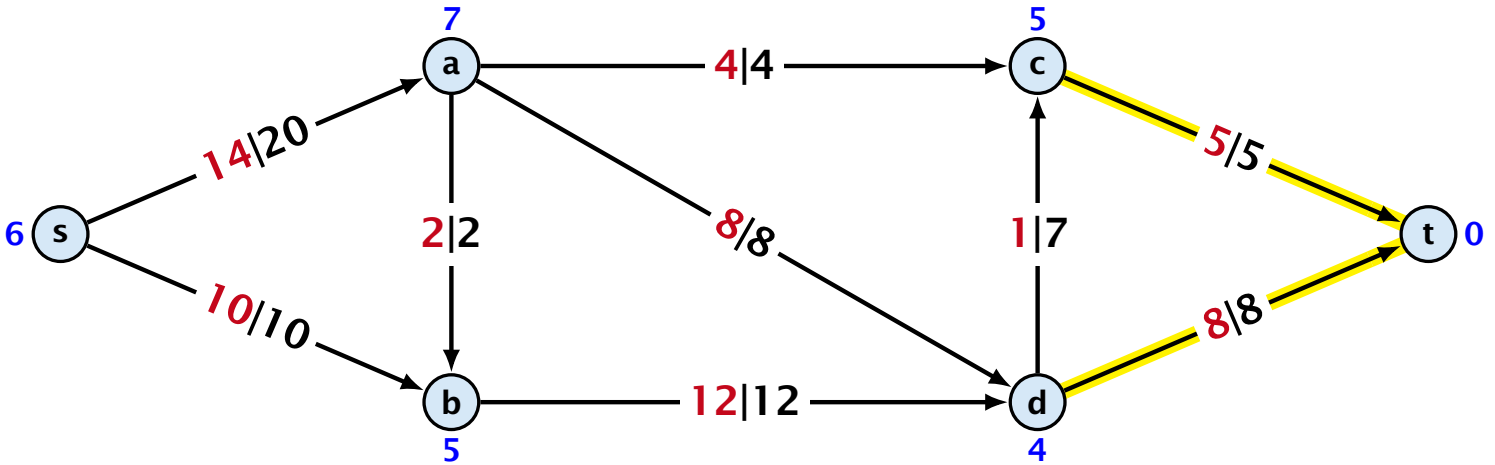


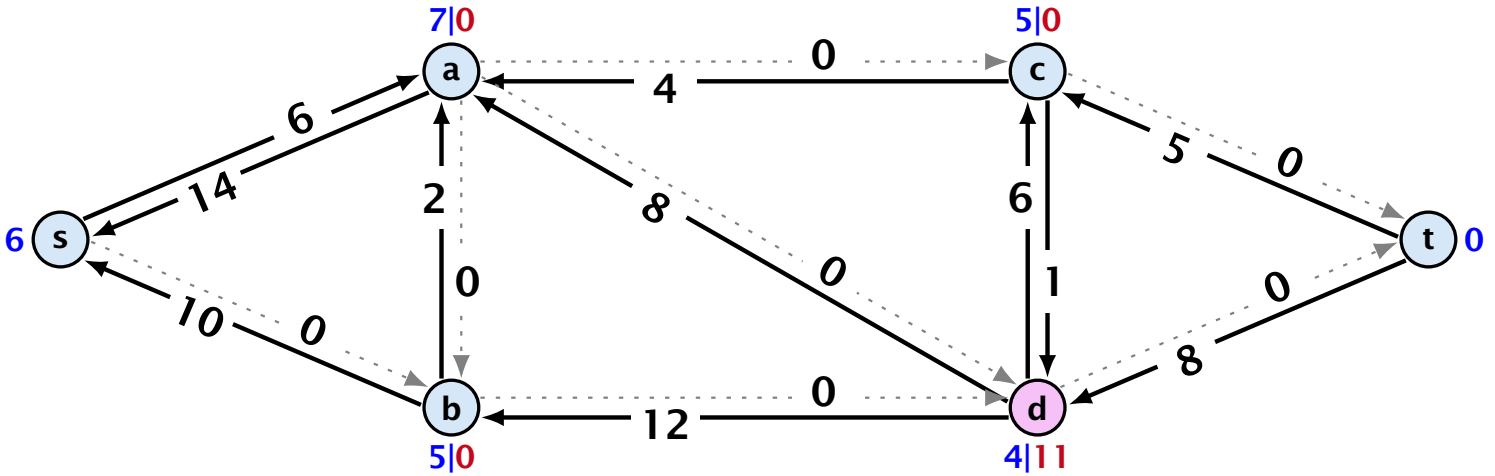
Preflow Push



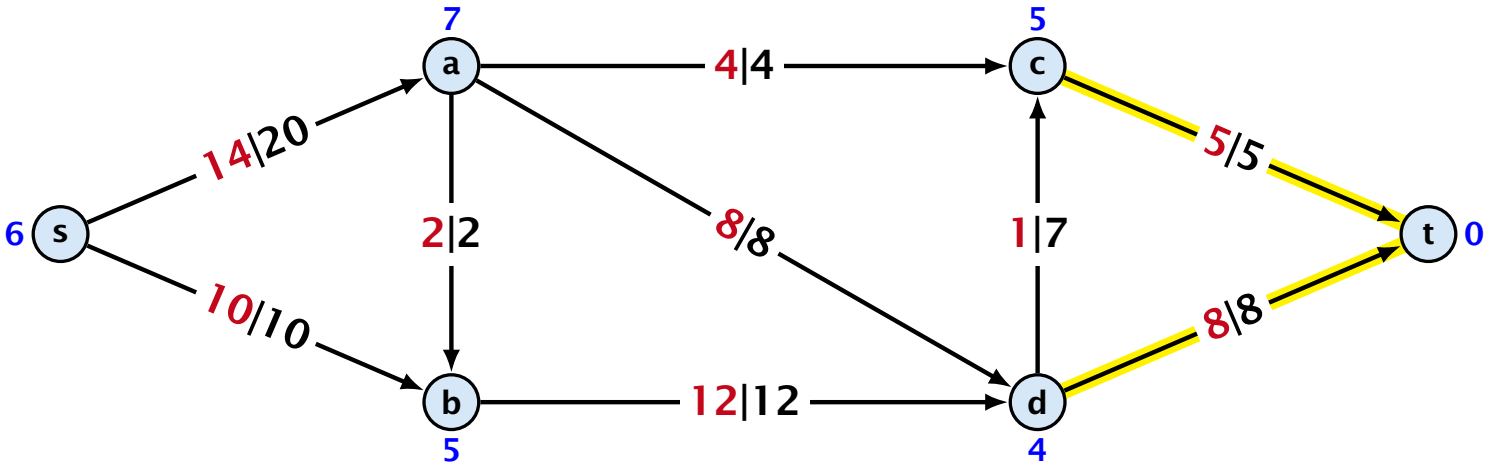


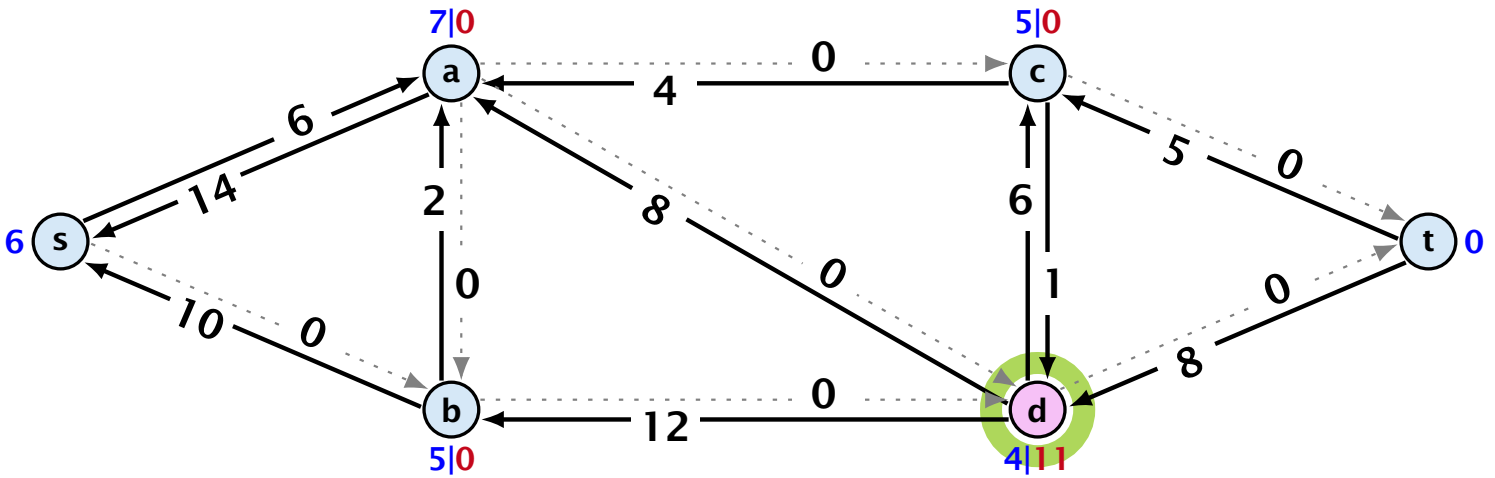
Preflow Push



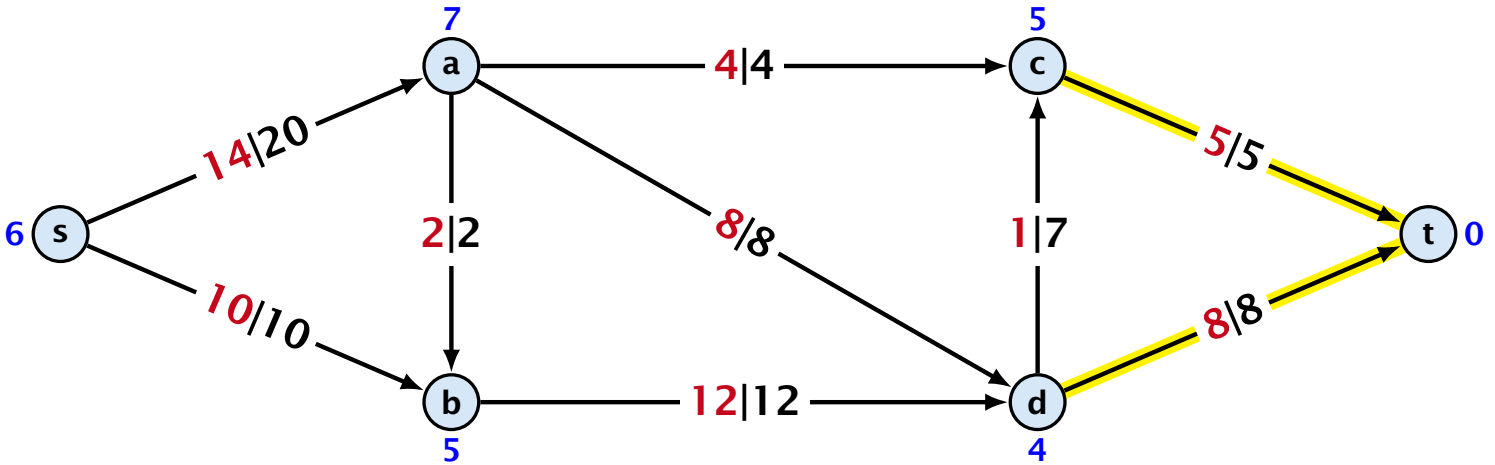


Preflow Push

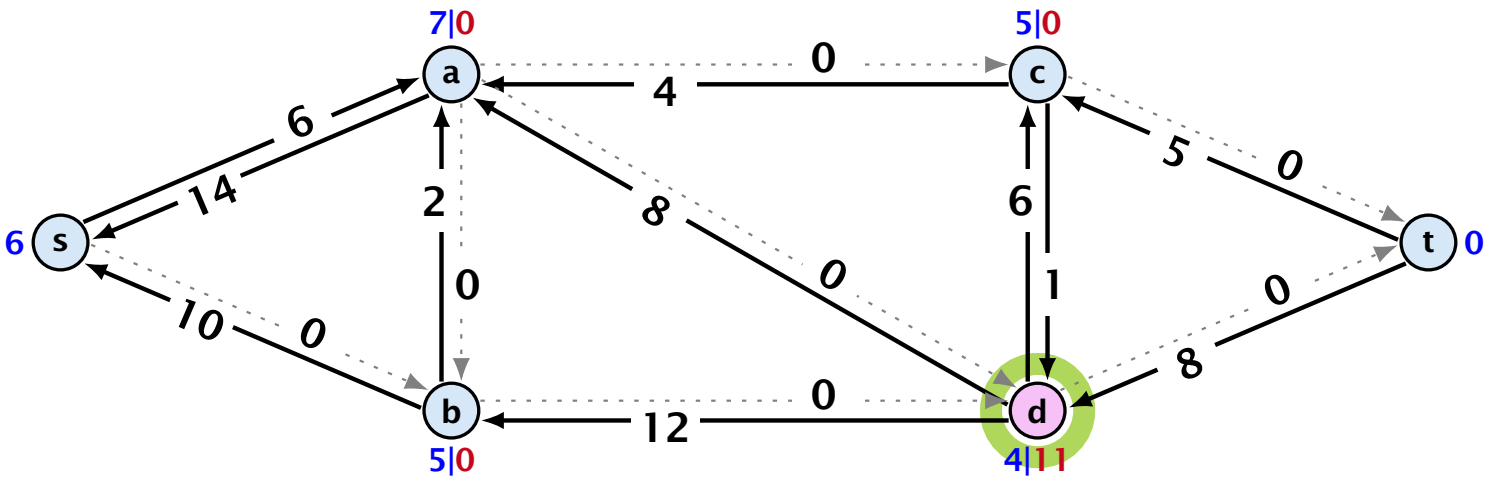




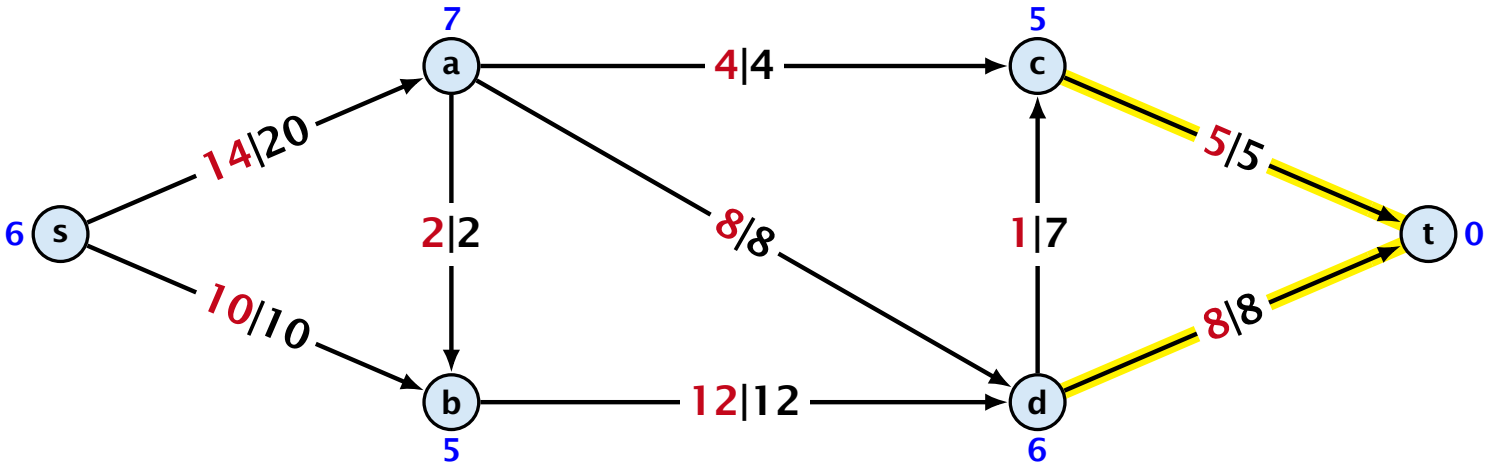
Preflow Push

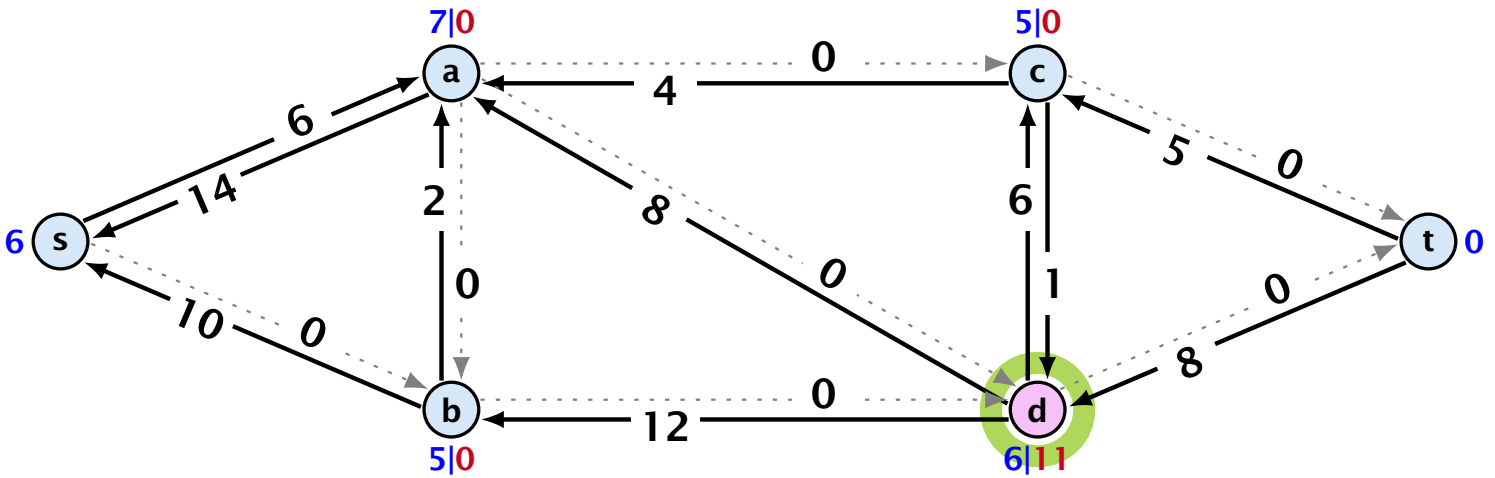


relabel to 6

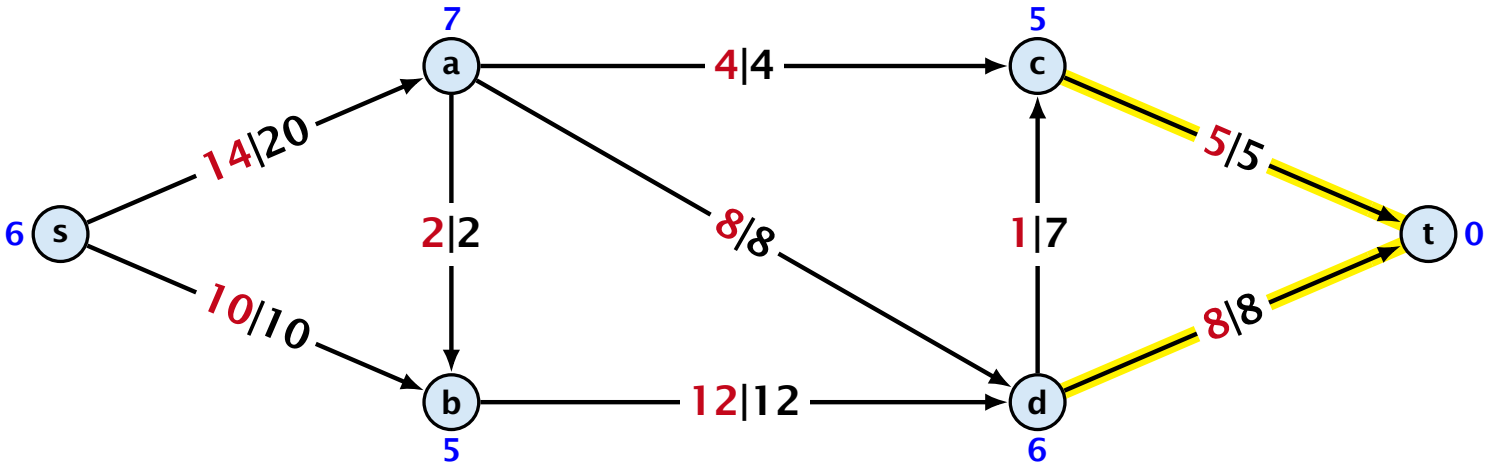


Preflow Push

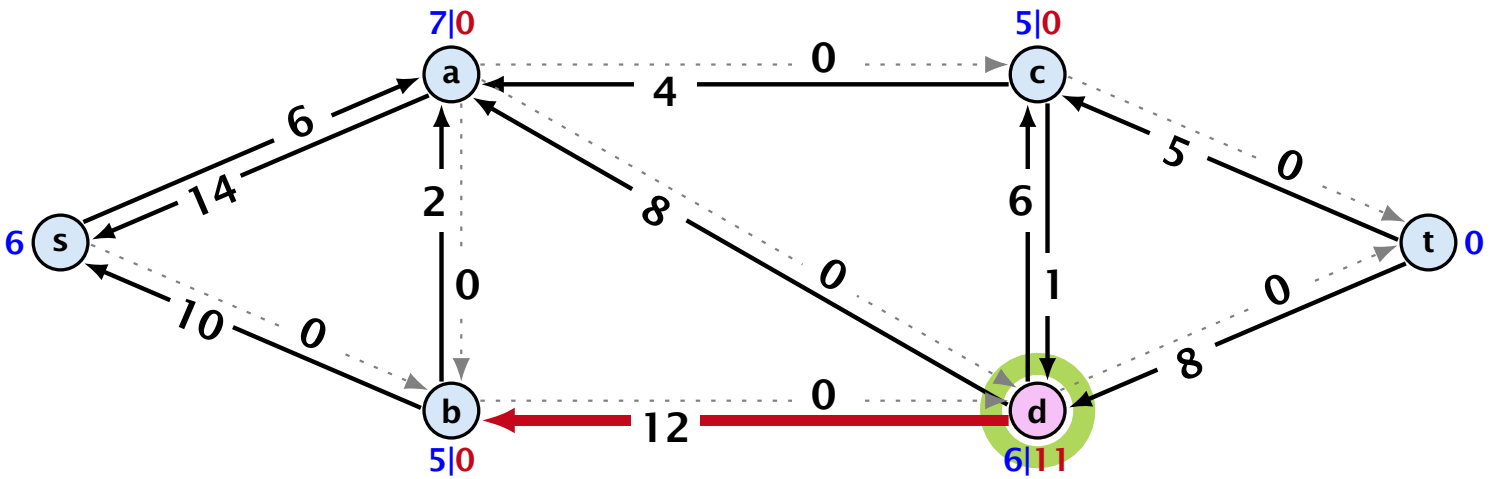




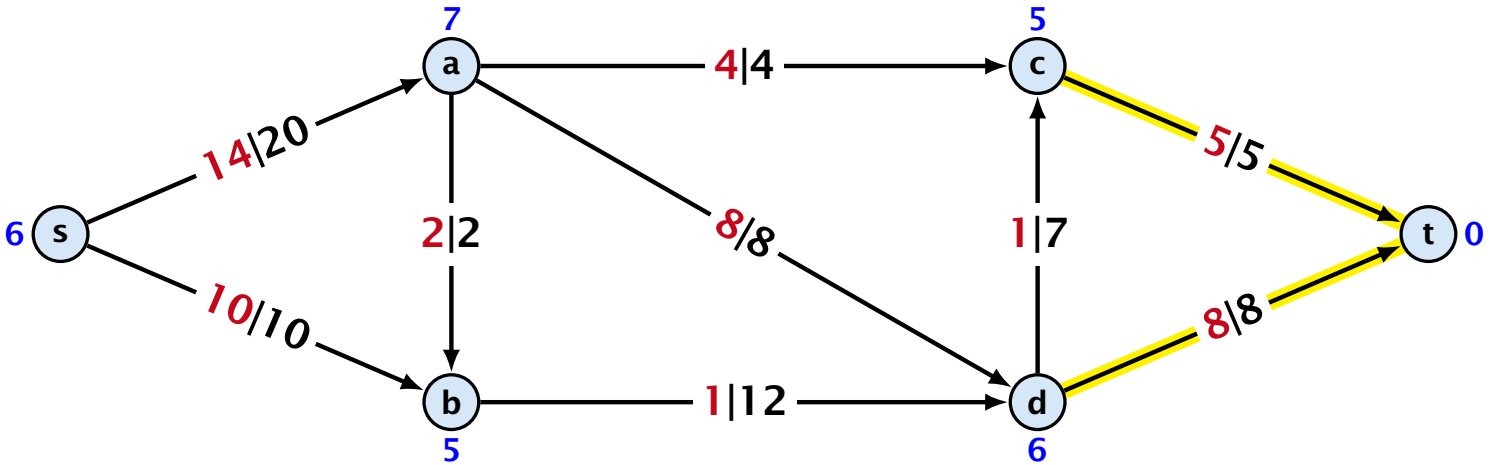
Preflow Push

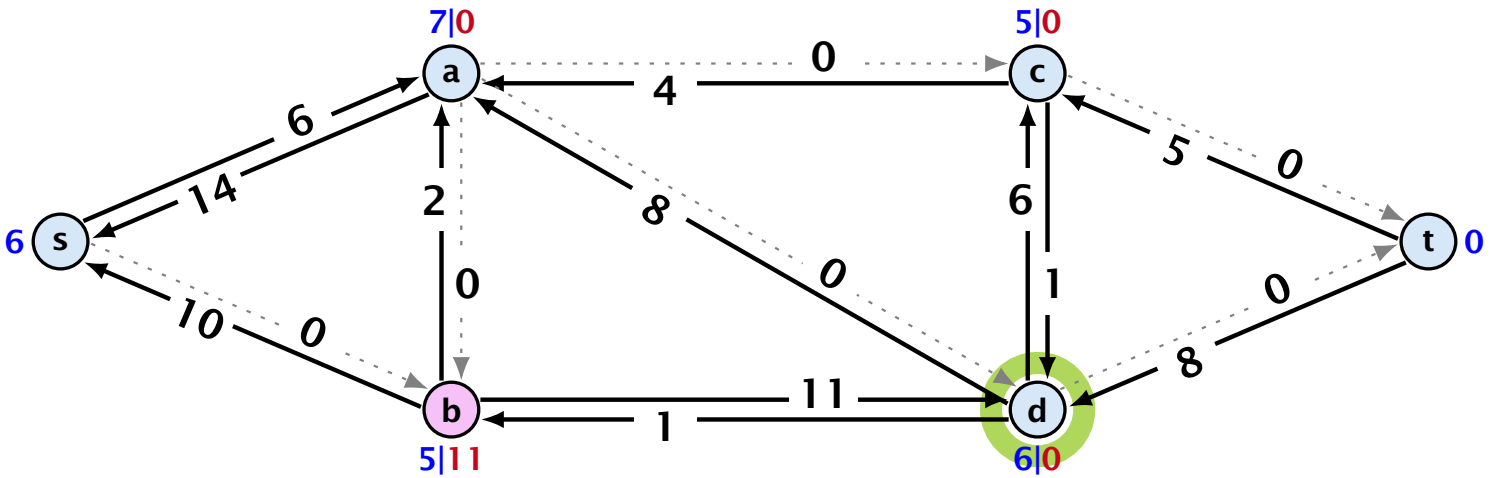


deactivating push

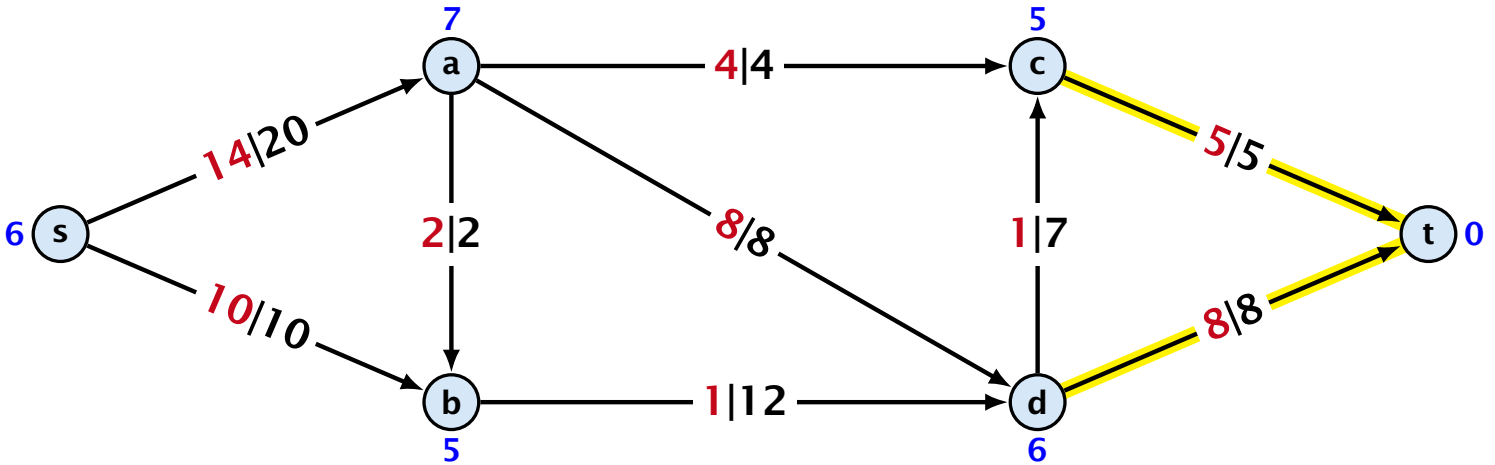


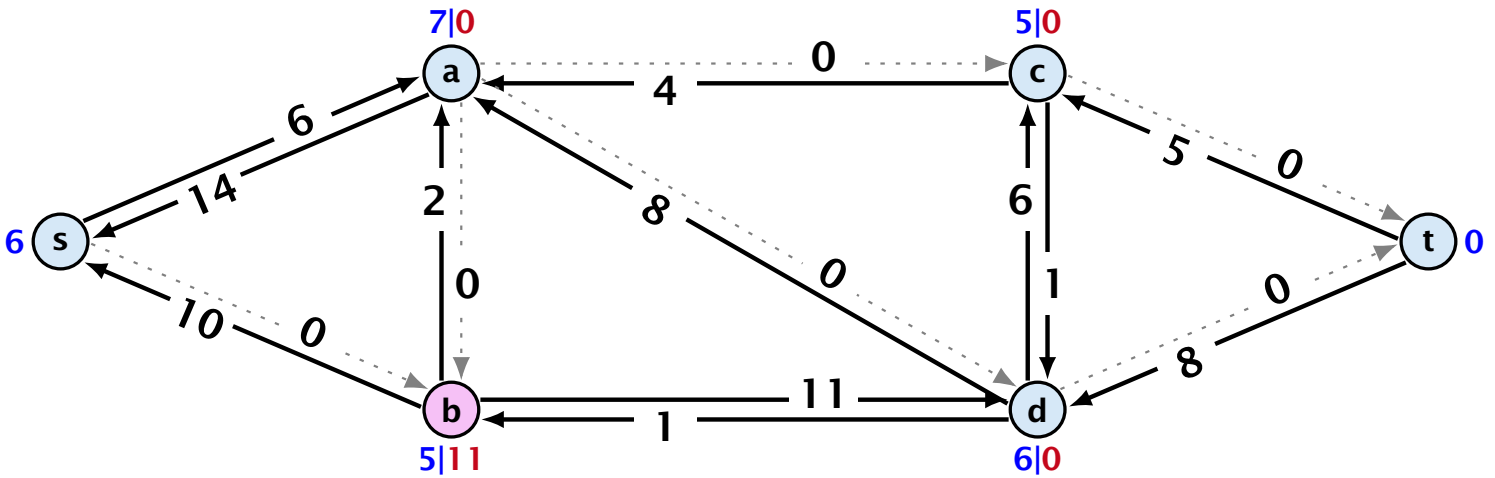
Preflow Push



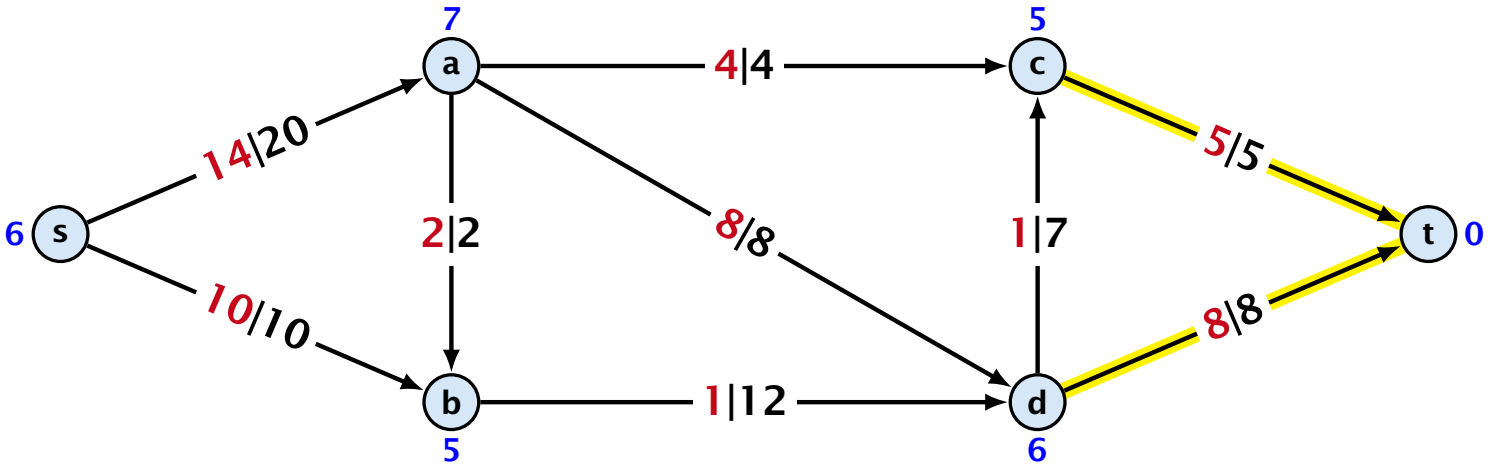


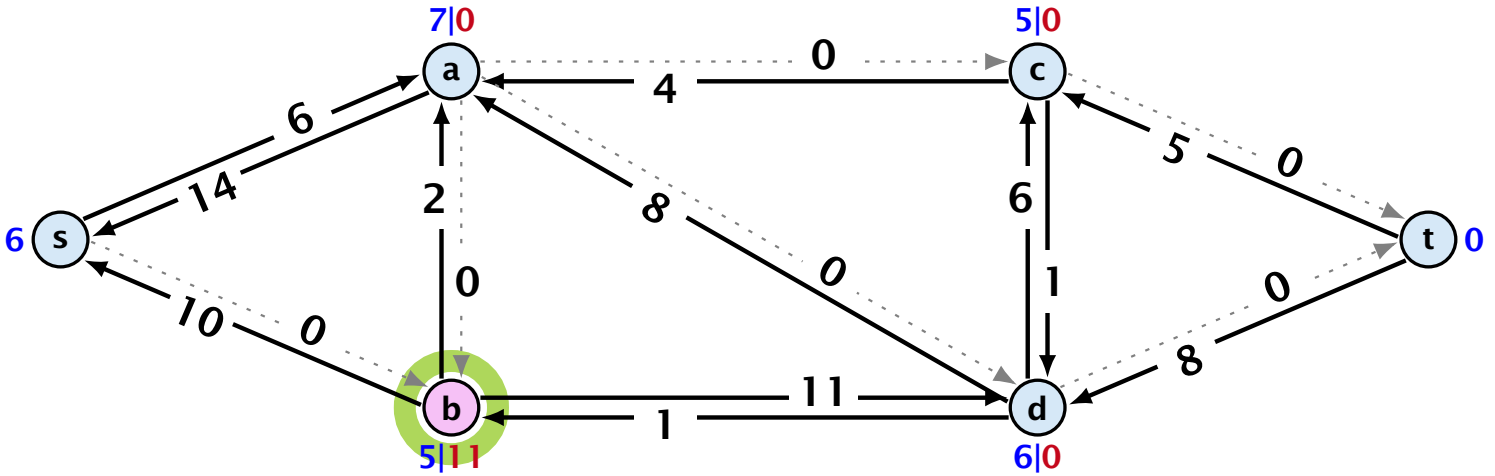
Preflow Push



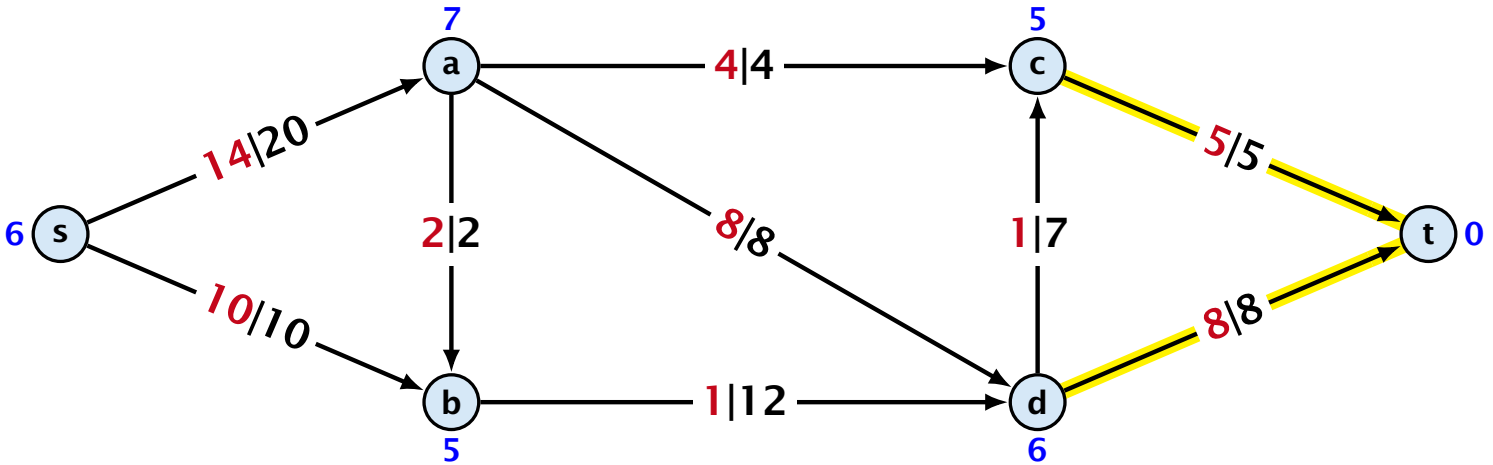


Preflow Push

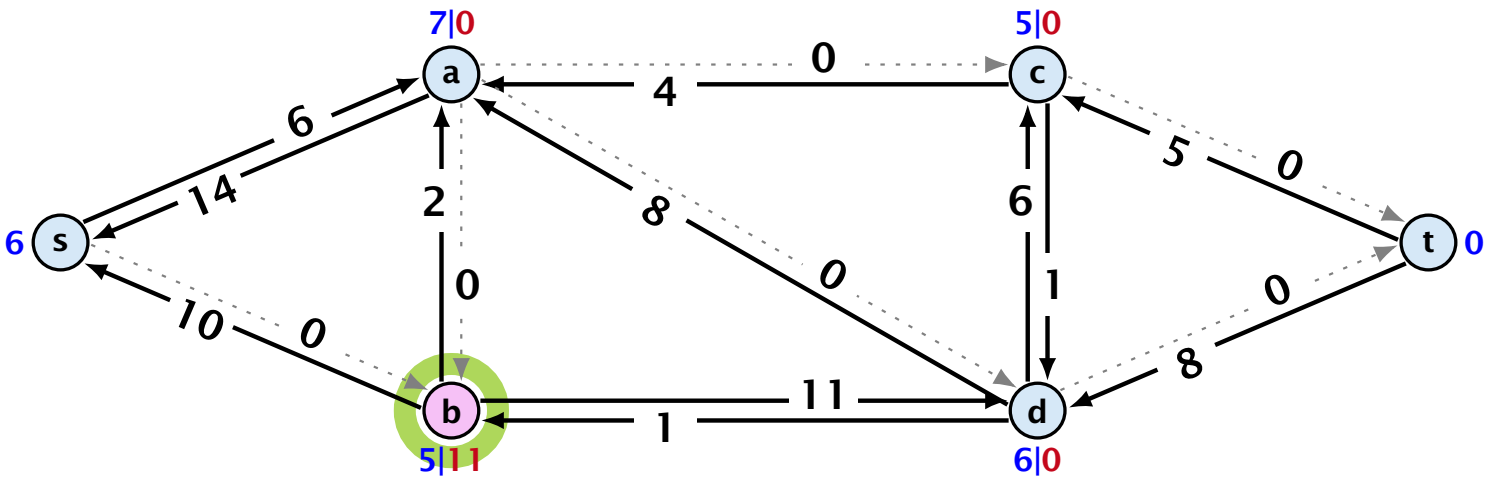




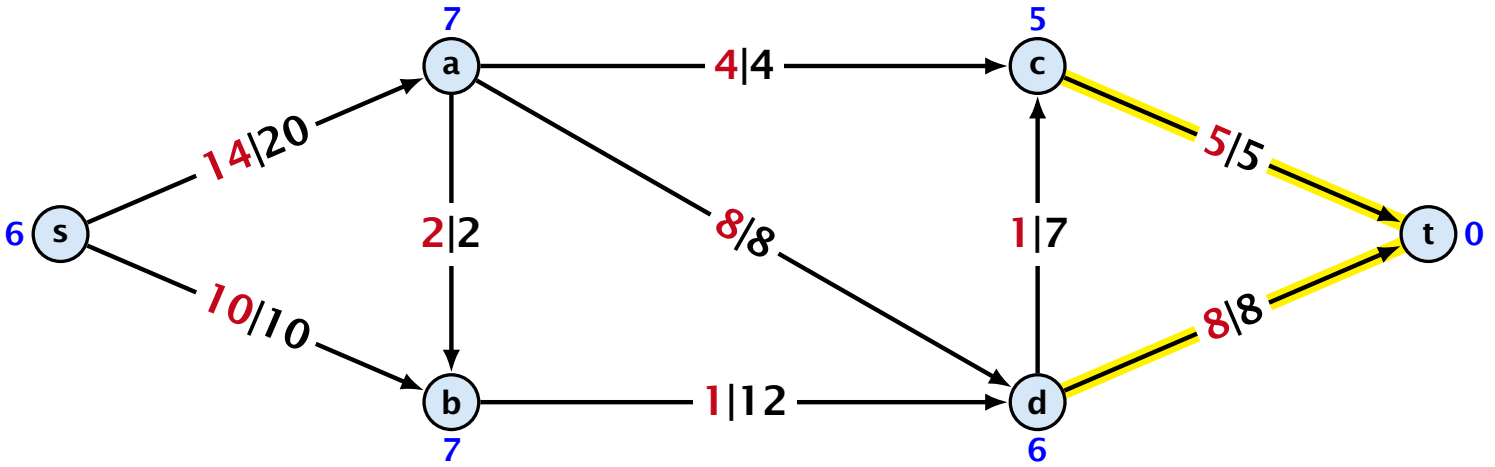
Preflow Push

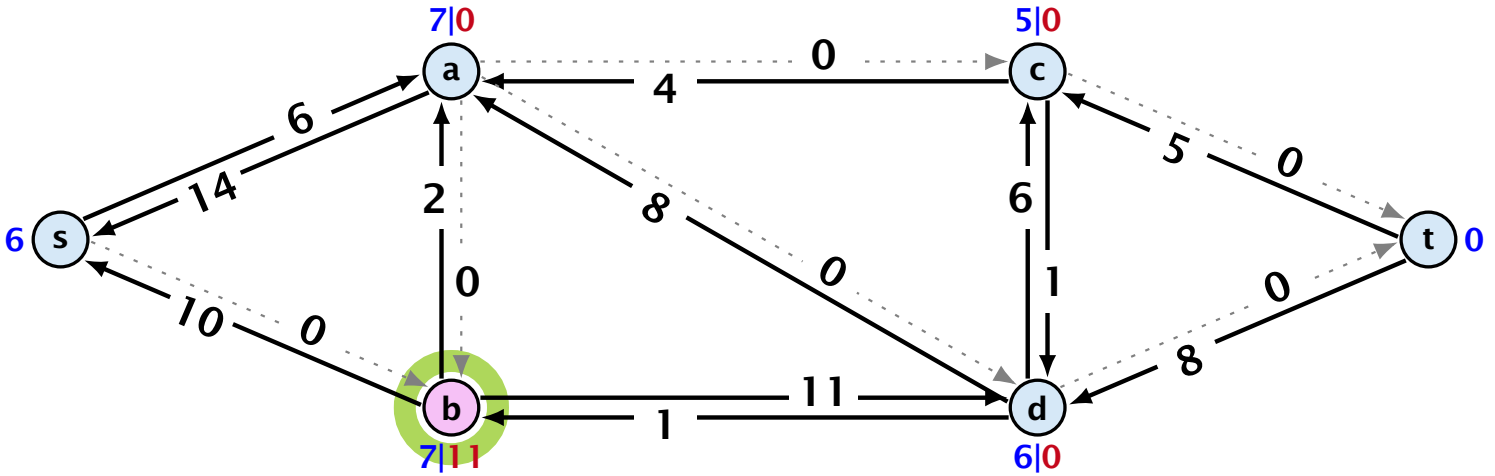


relabel to 7

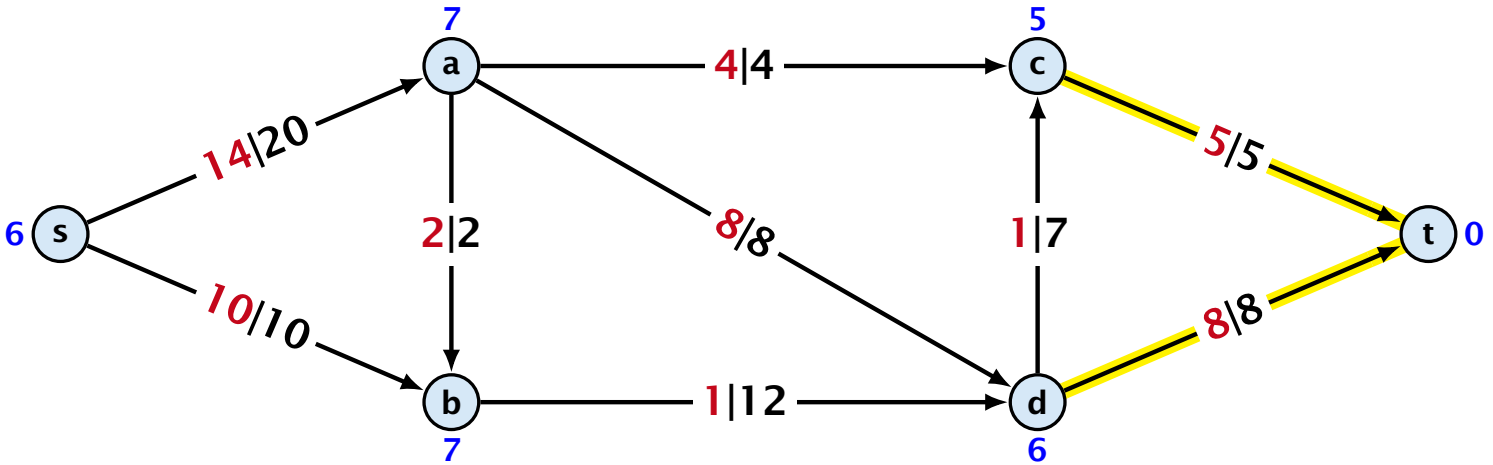


Preflow Push

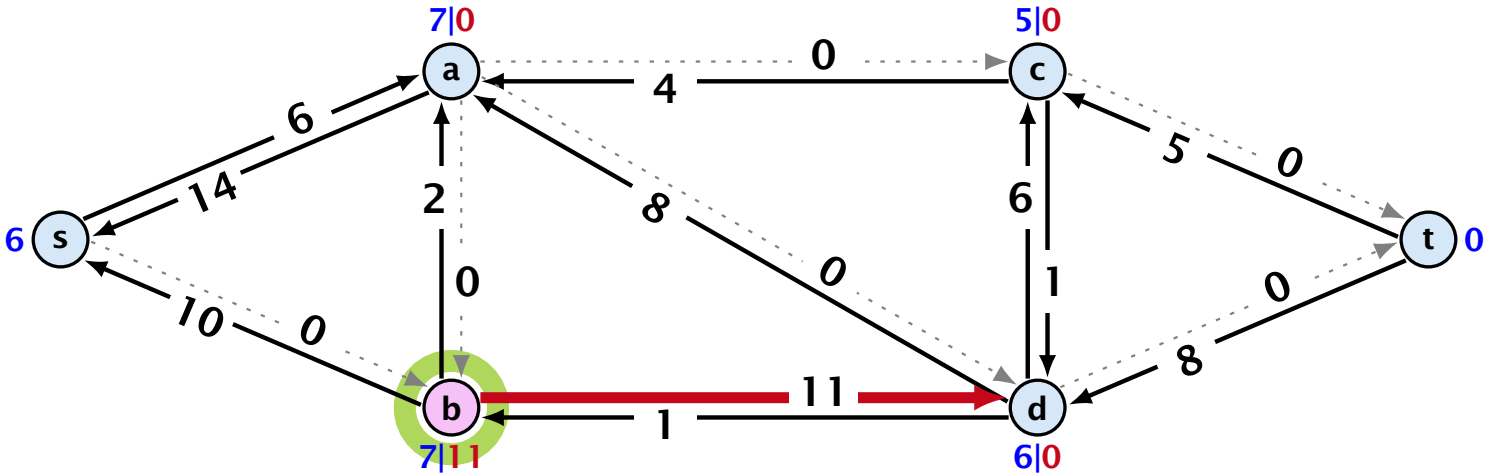




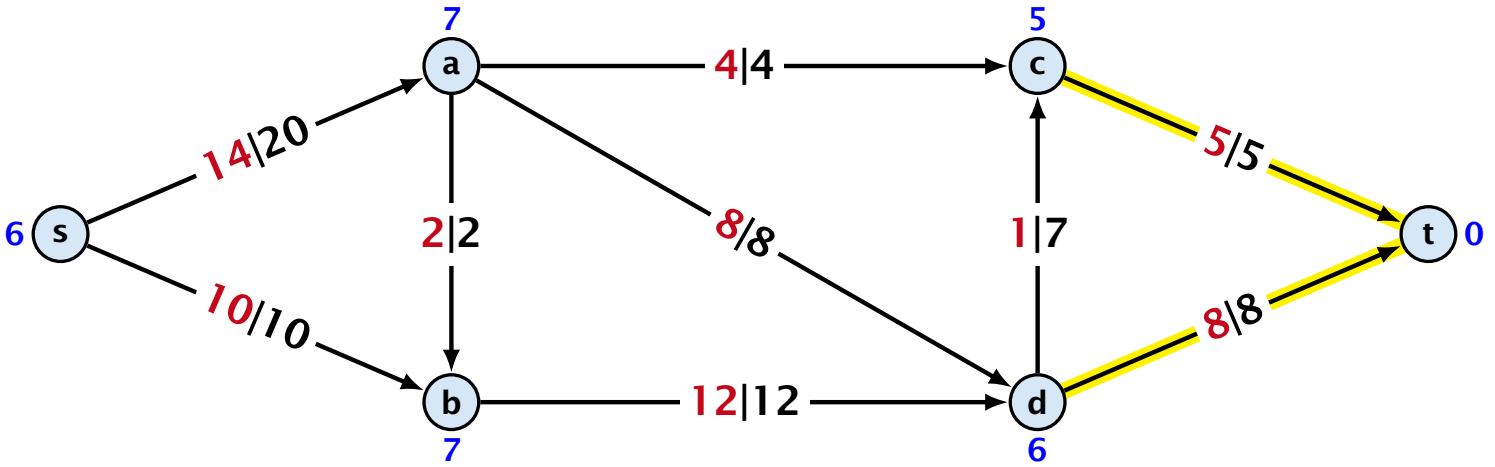
Preflow Push

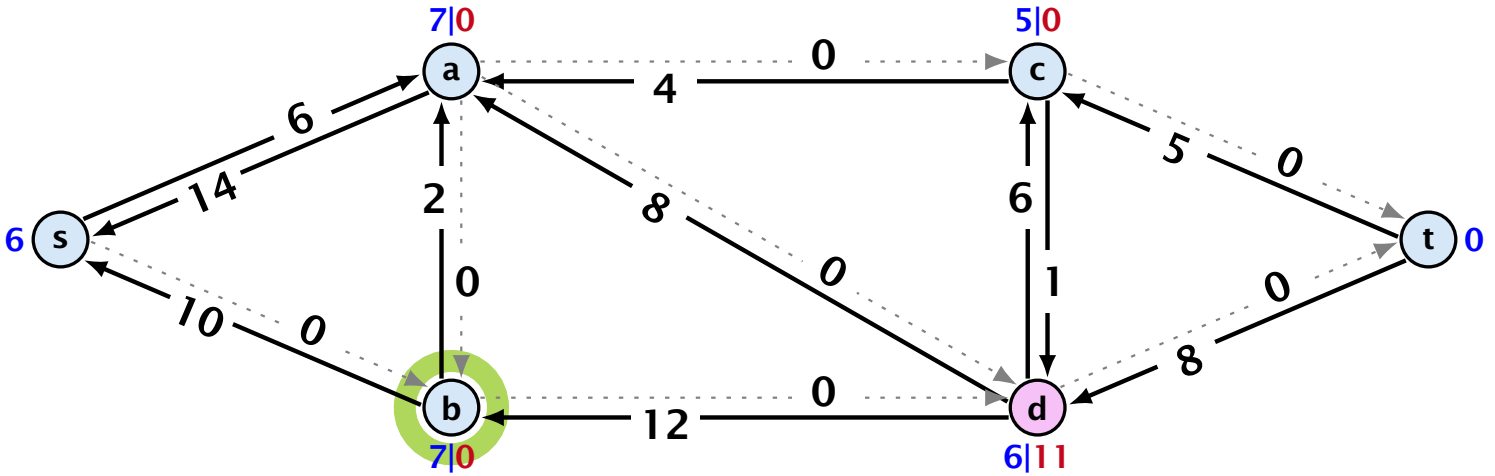


saturating and deactivating push

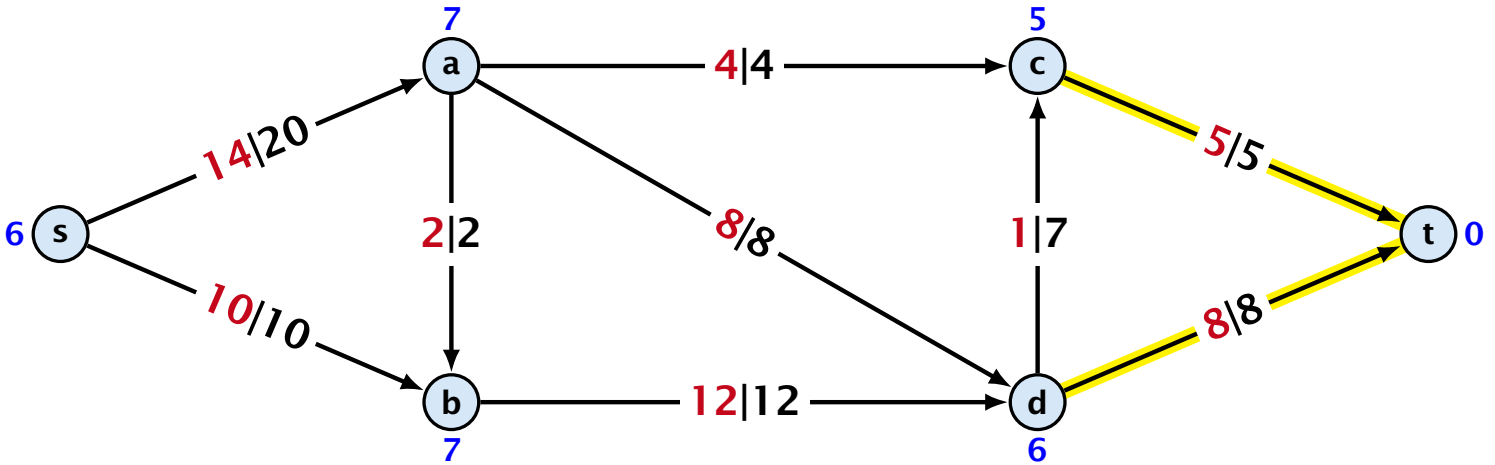


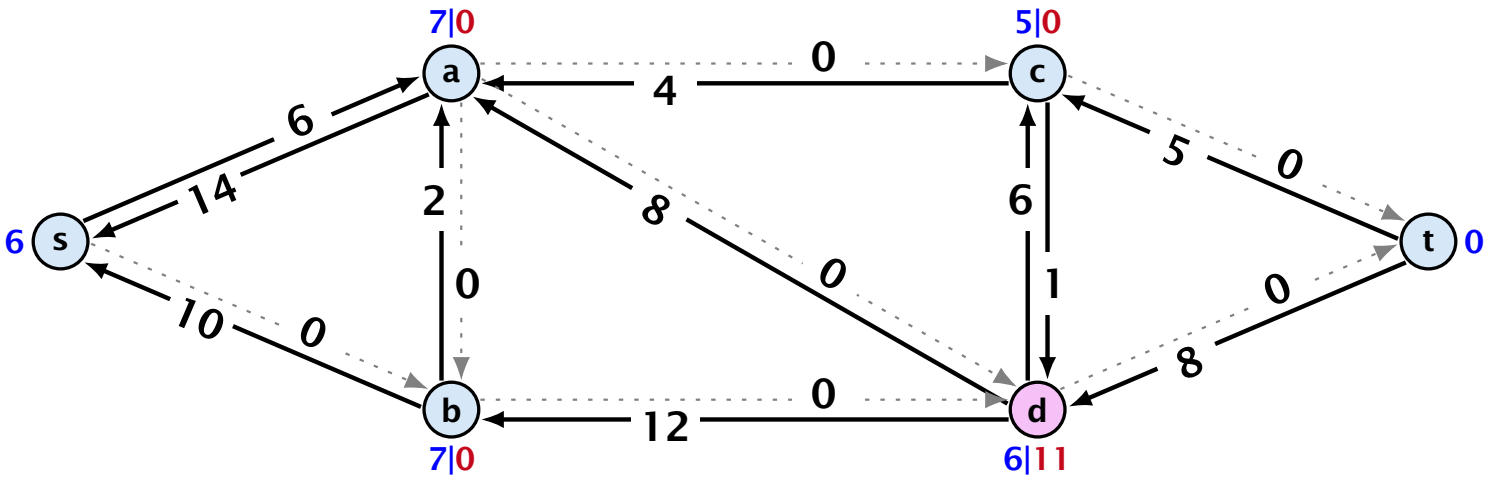
Preflow Push



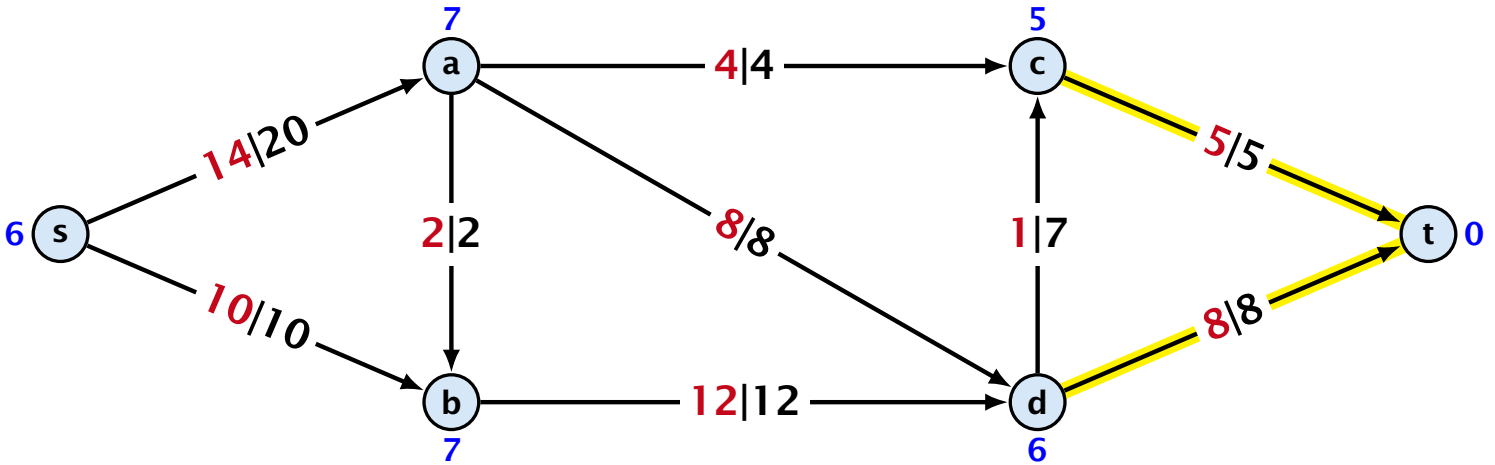


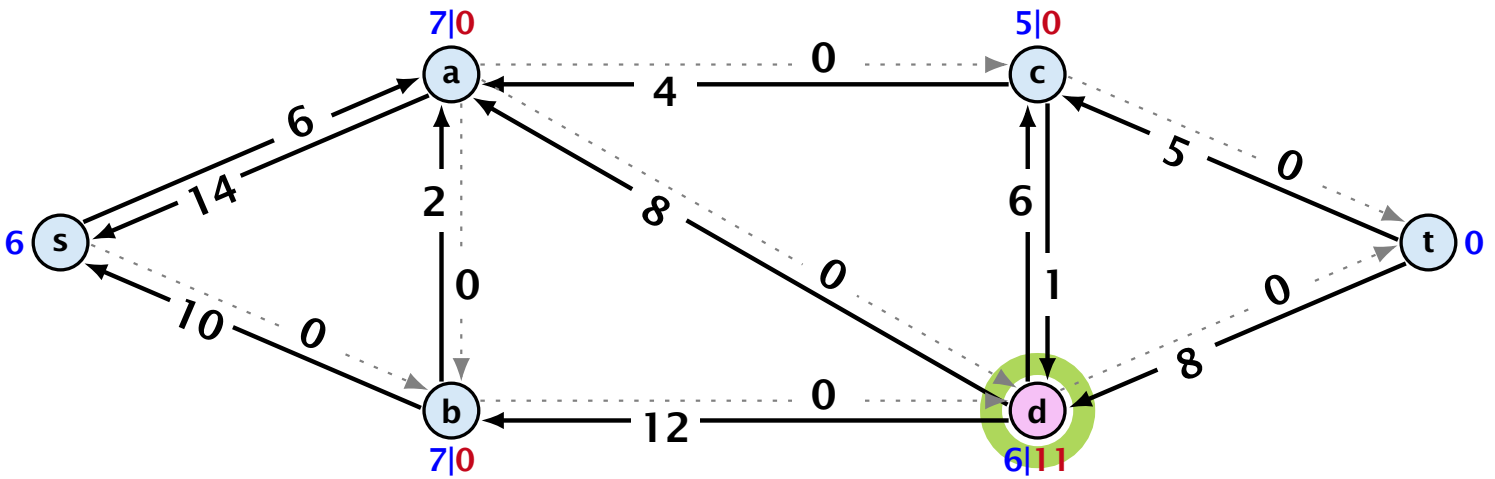
Preflow Push



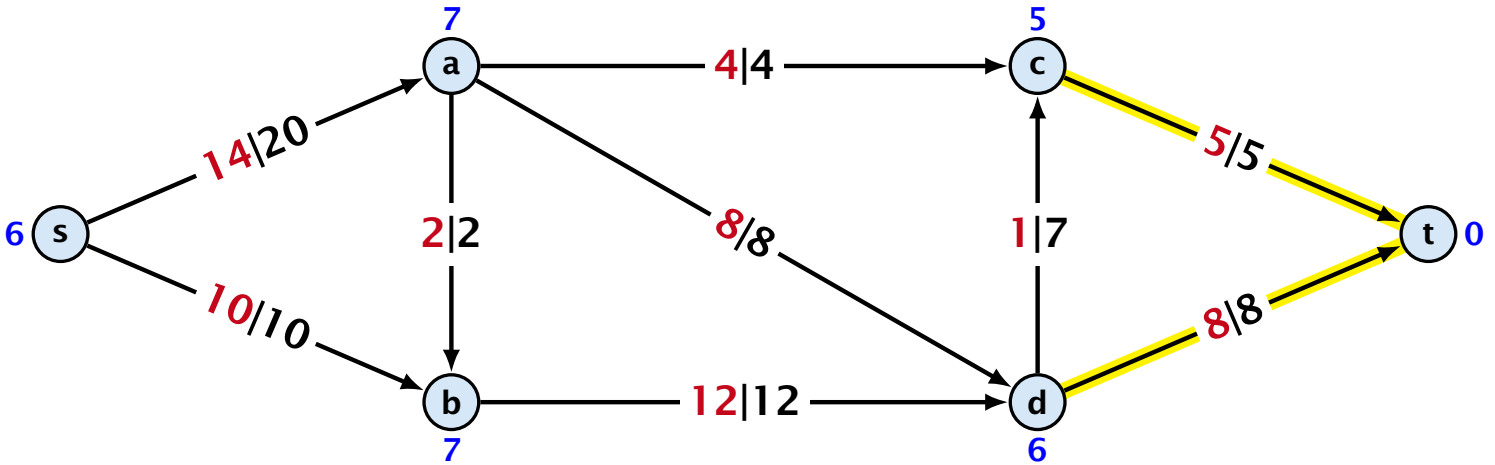


Preflow Push

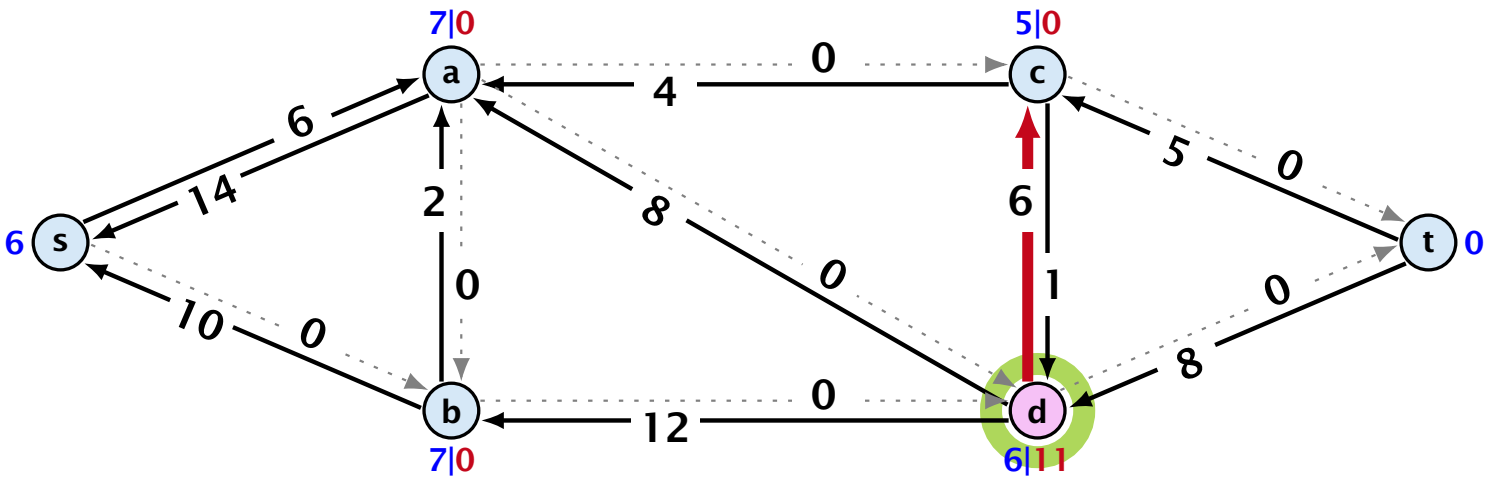




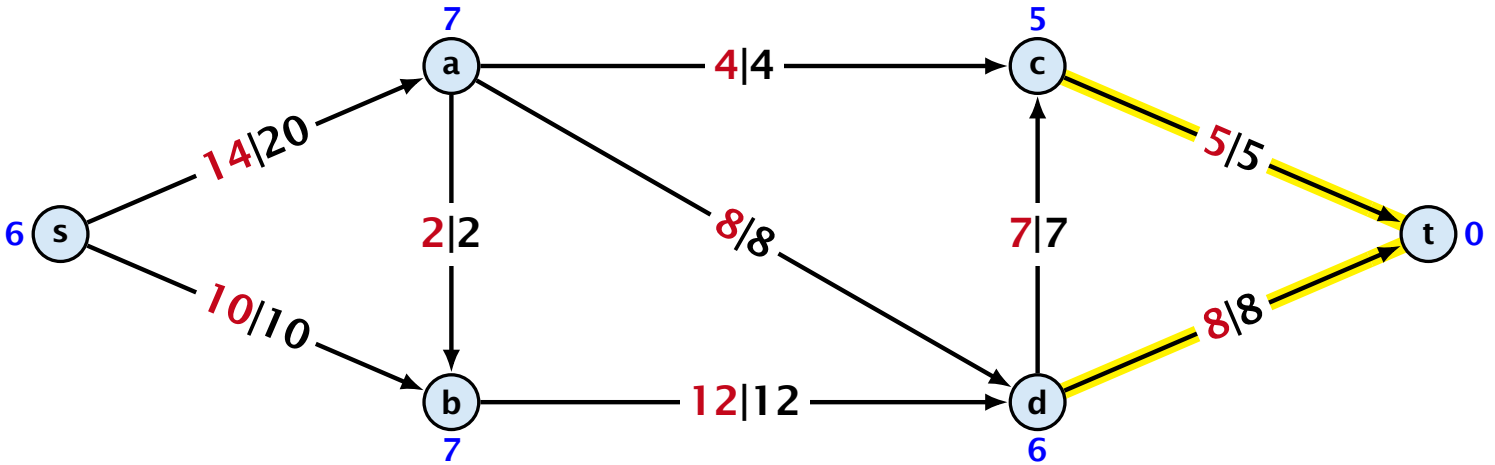
Preflow Push

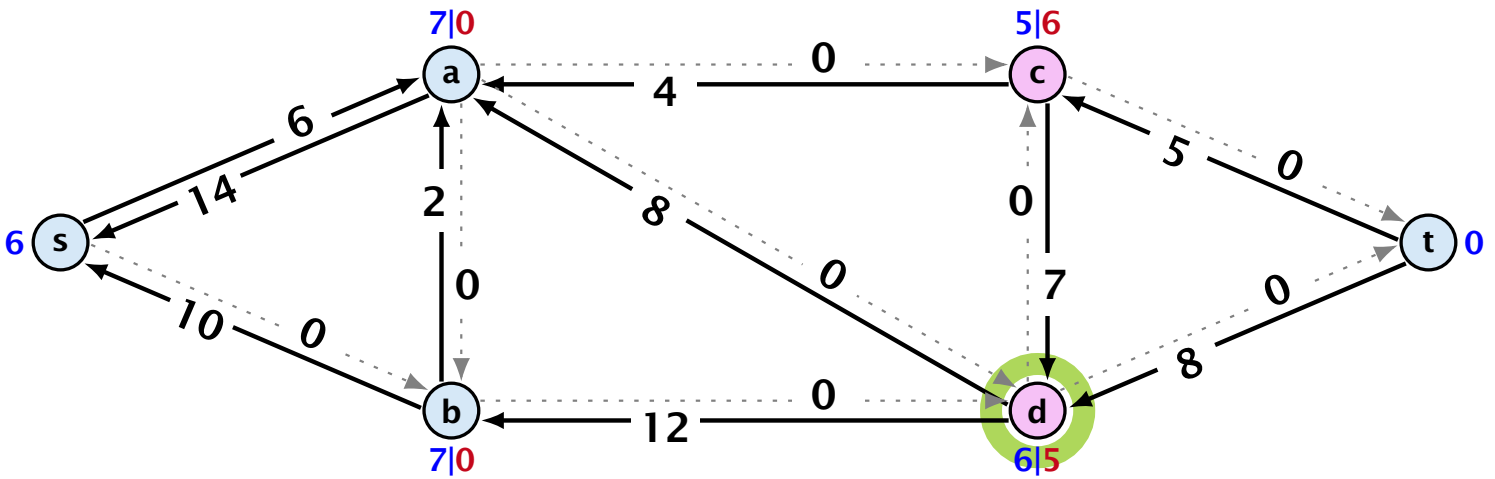


saturating push

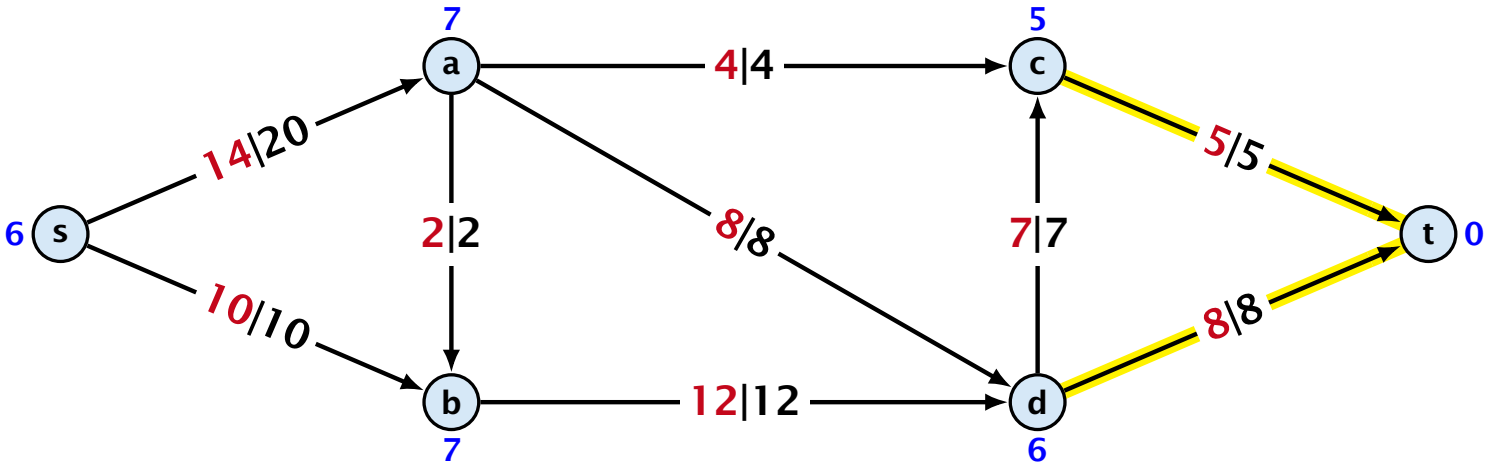


Preflow Push

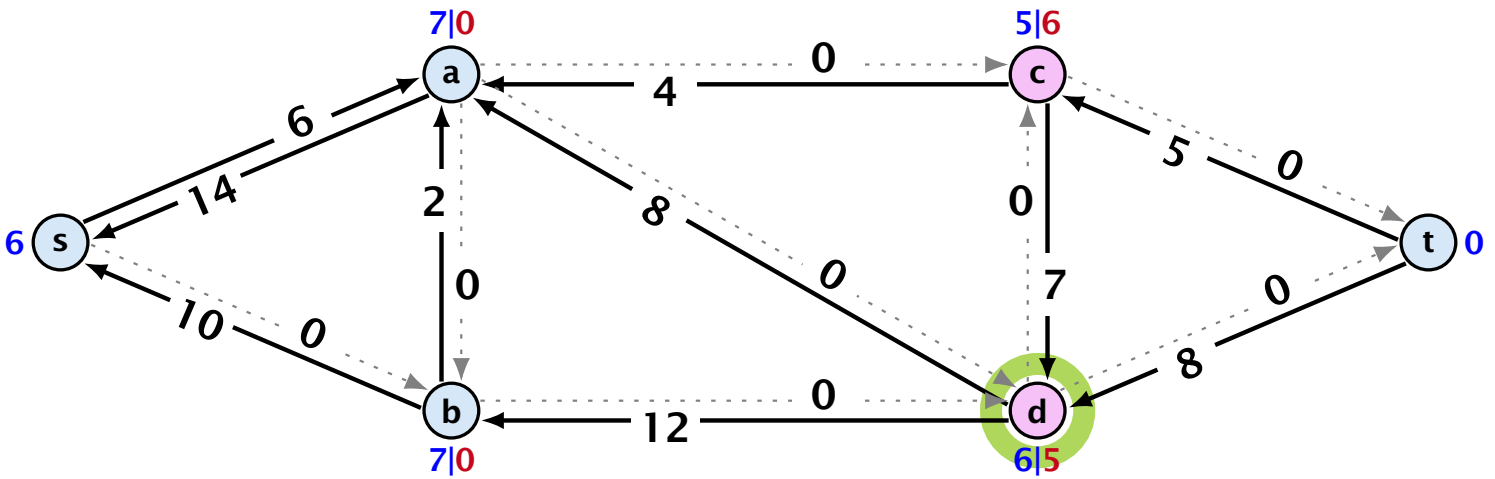




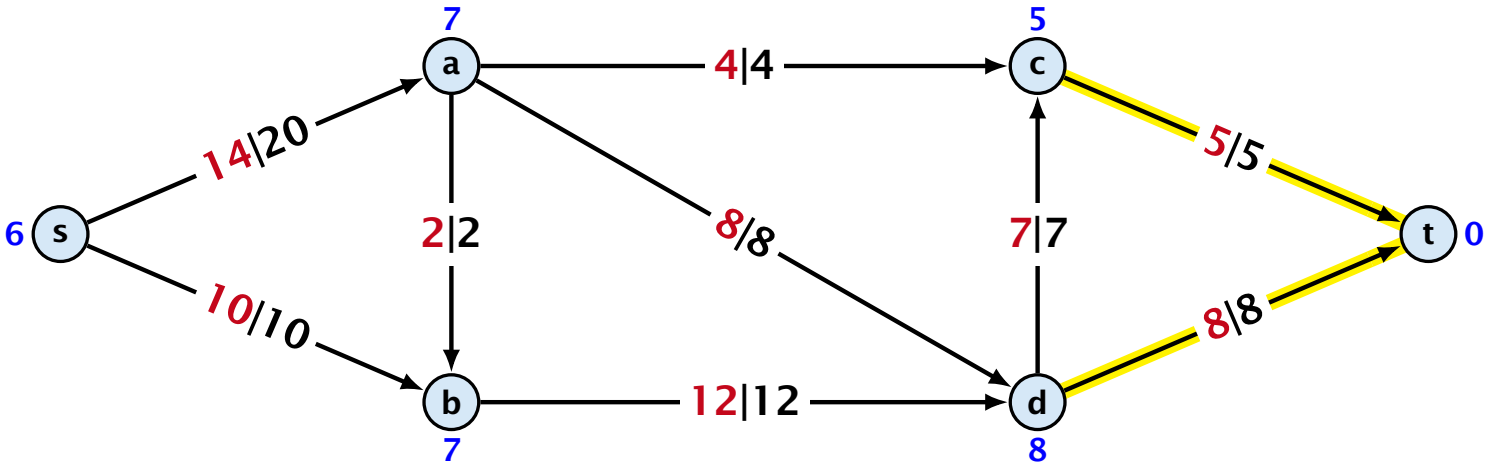
Preflow Push

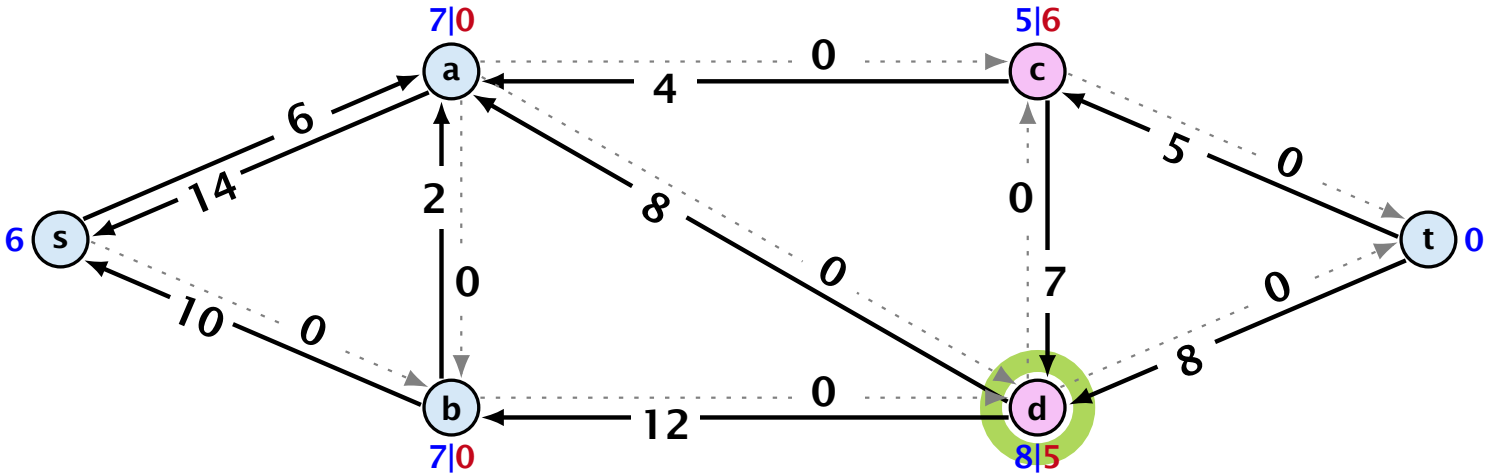


relabel to 8

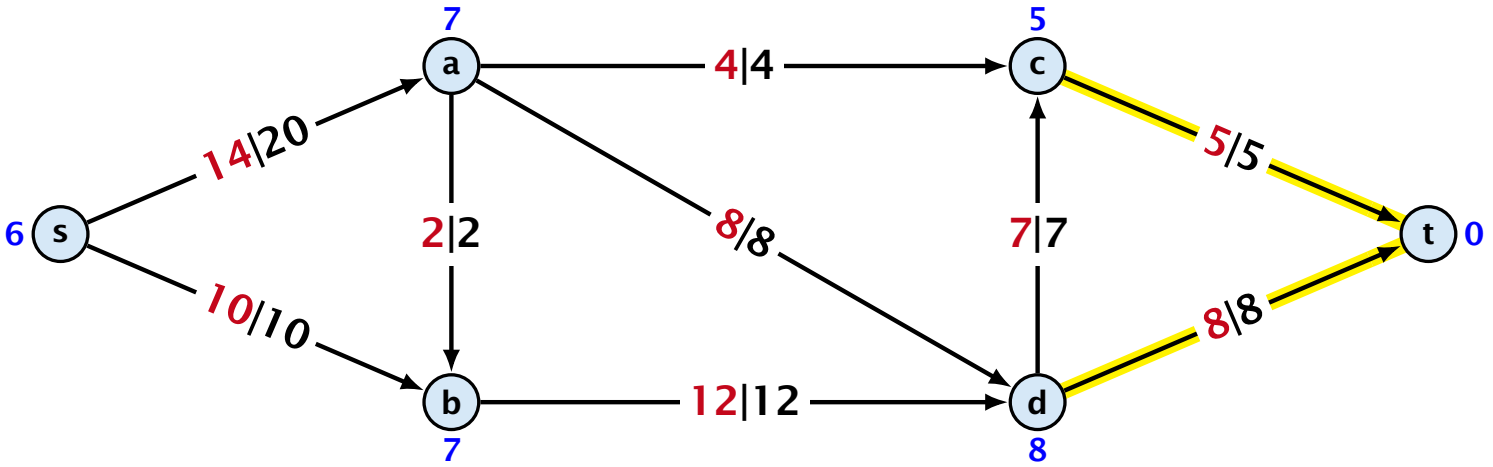


Preflow Push

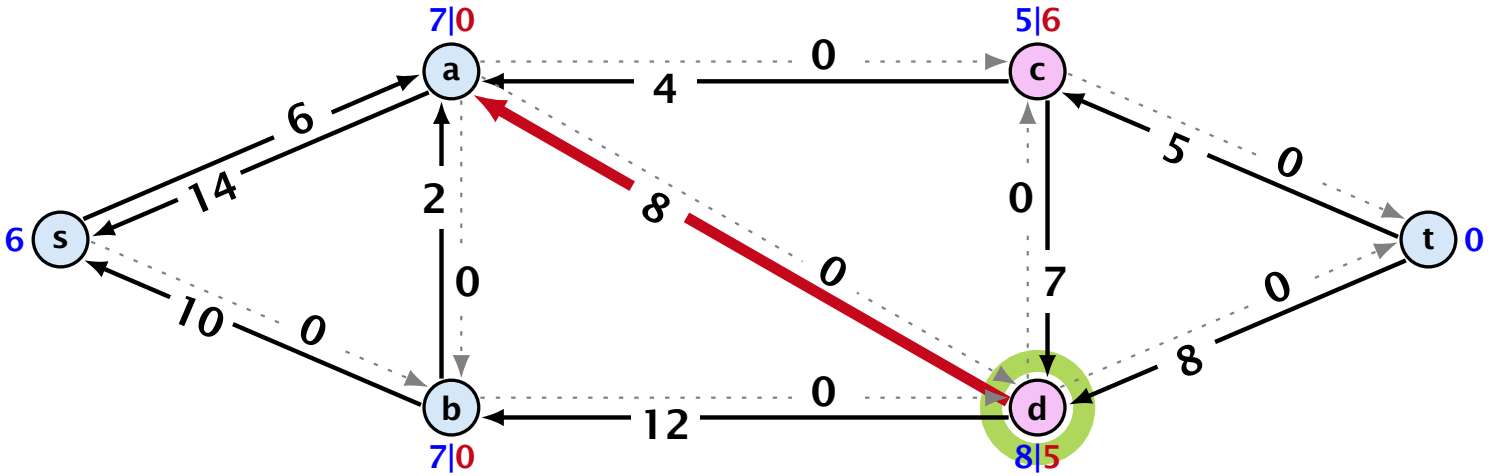




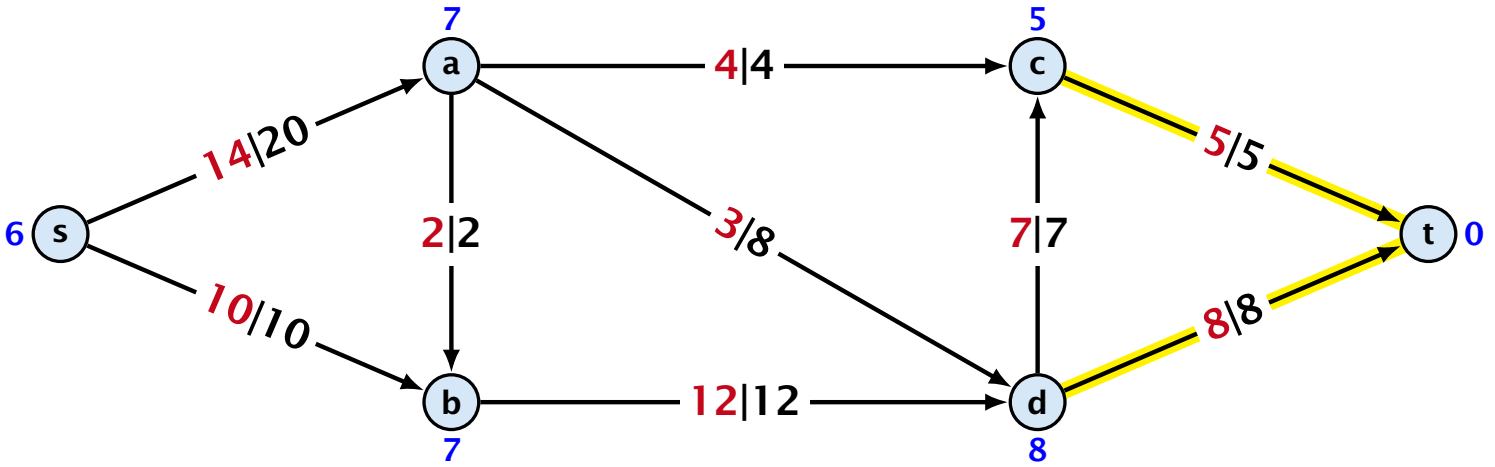
Preflow Push

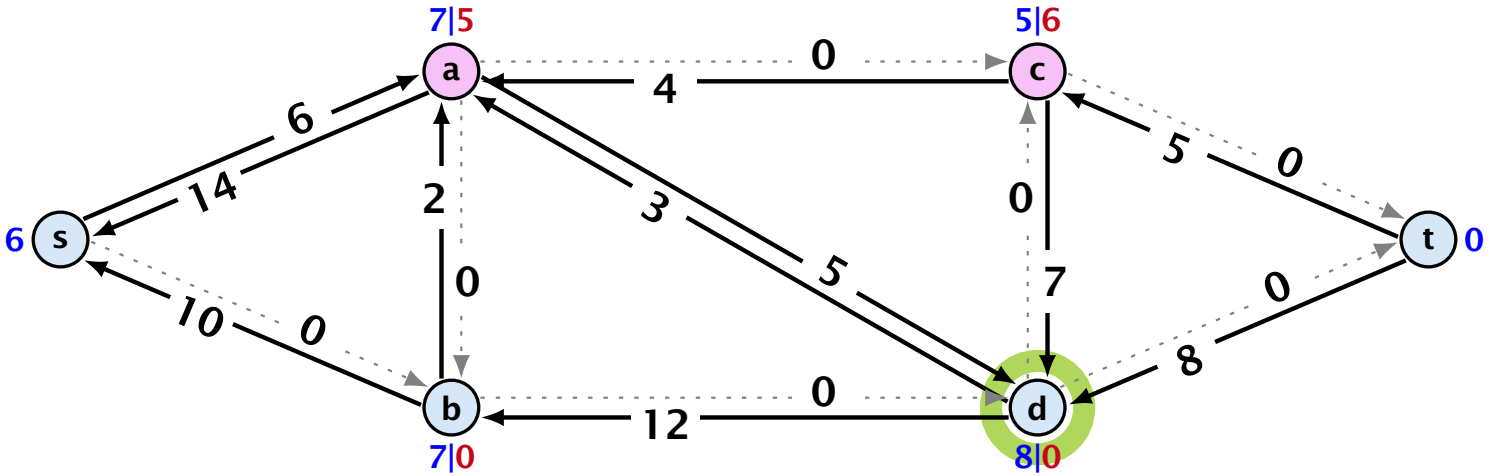


deactivating push

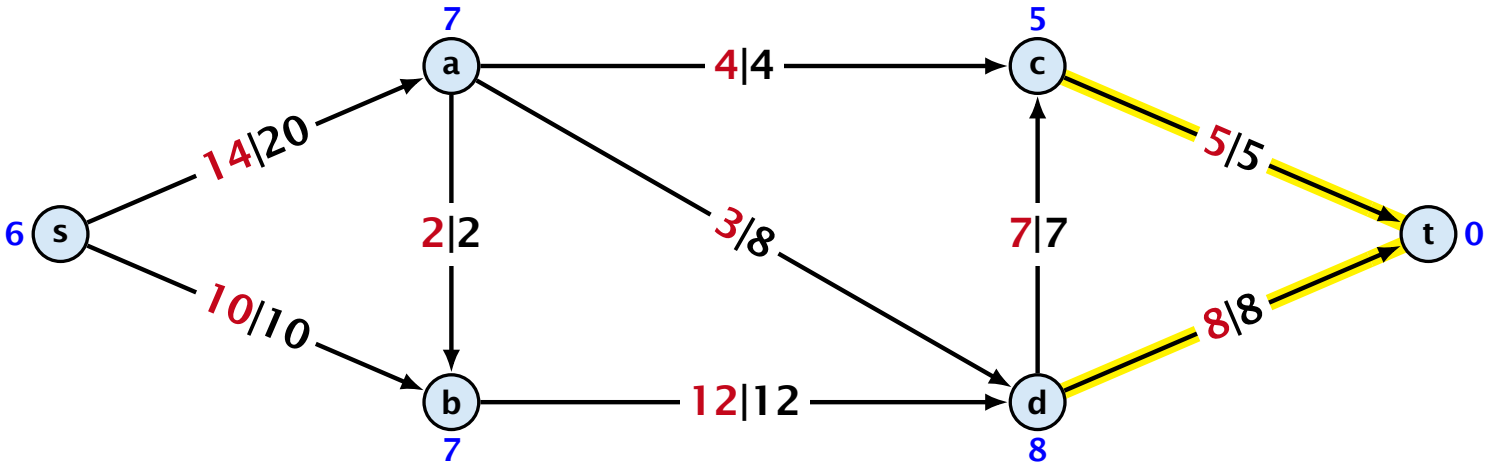


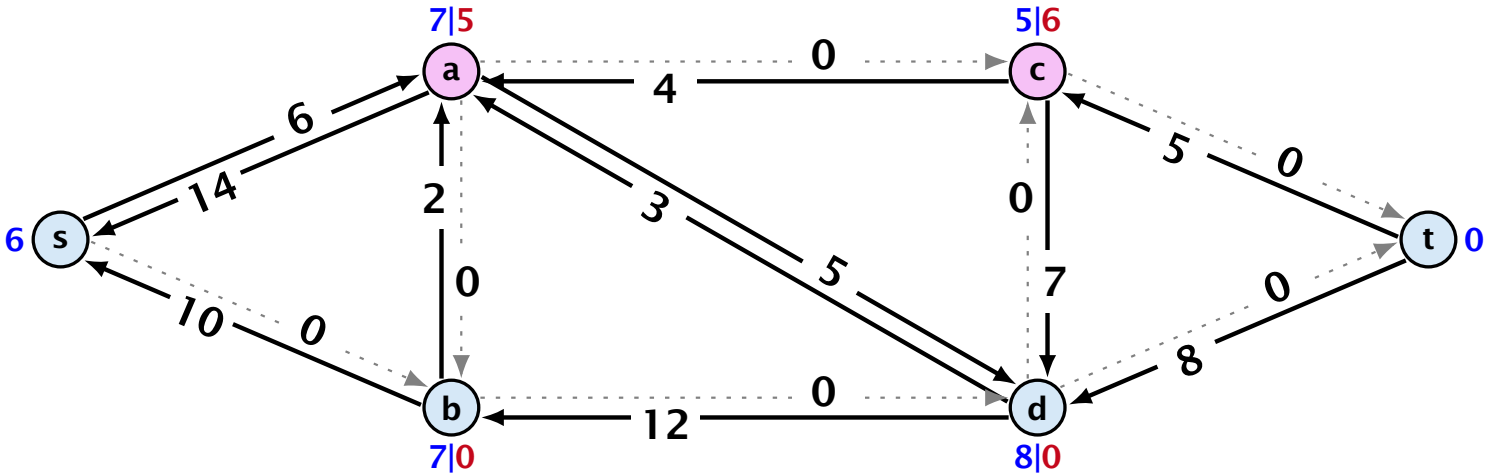
Preflow Push



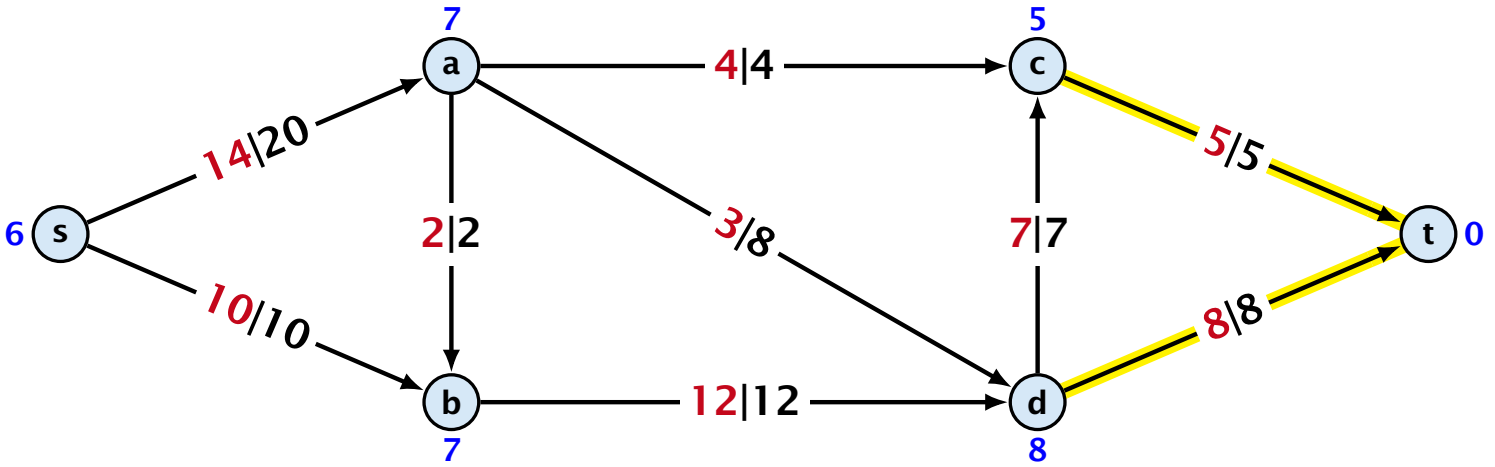


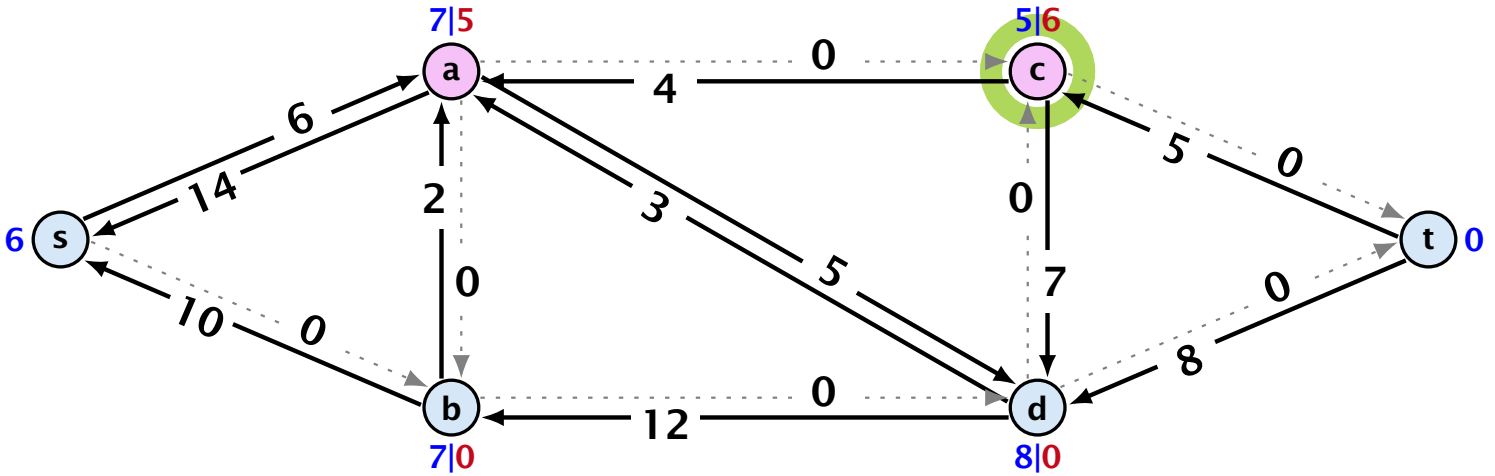
Preflow Push



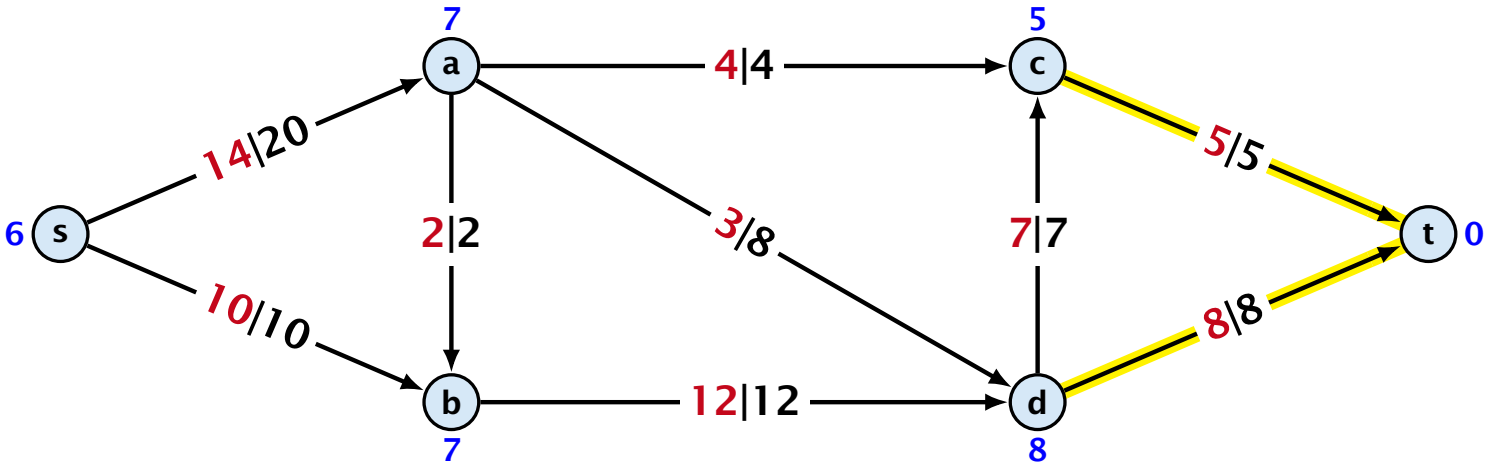


Preflow Push

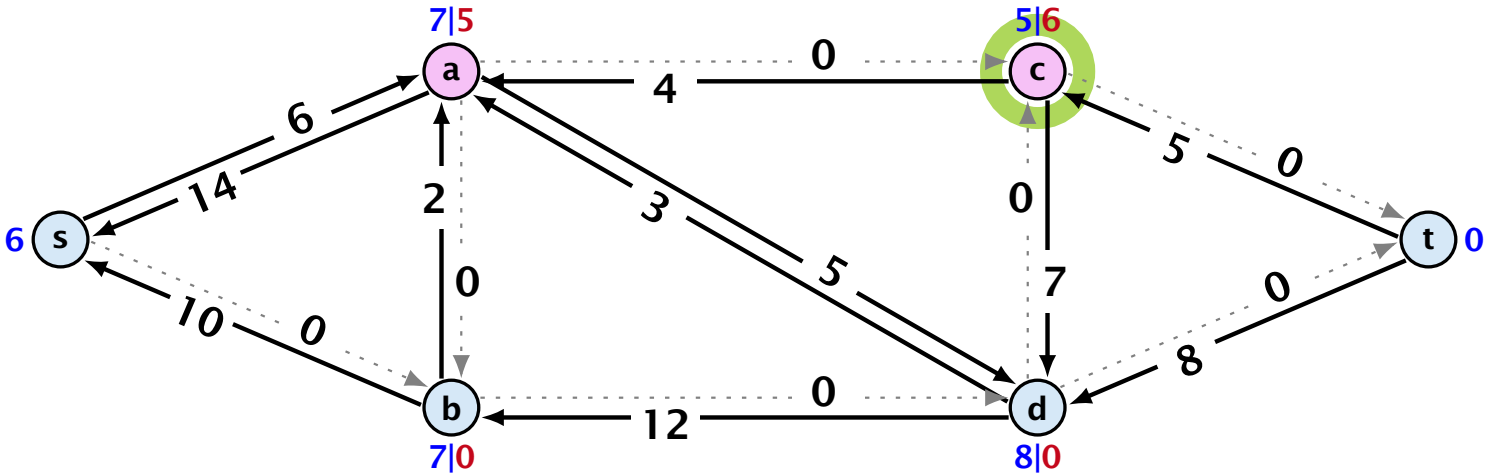




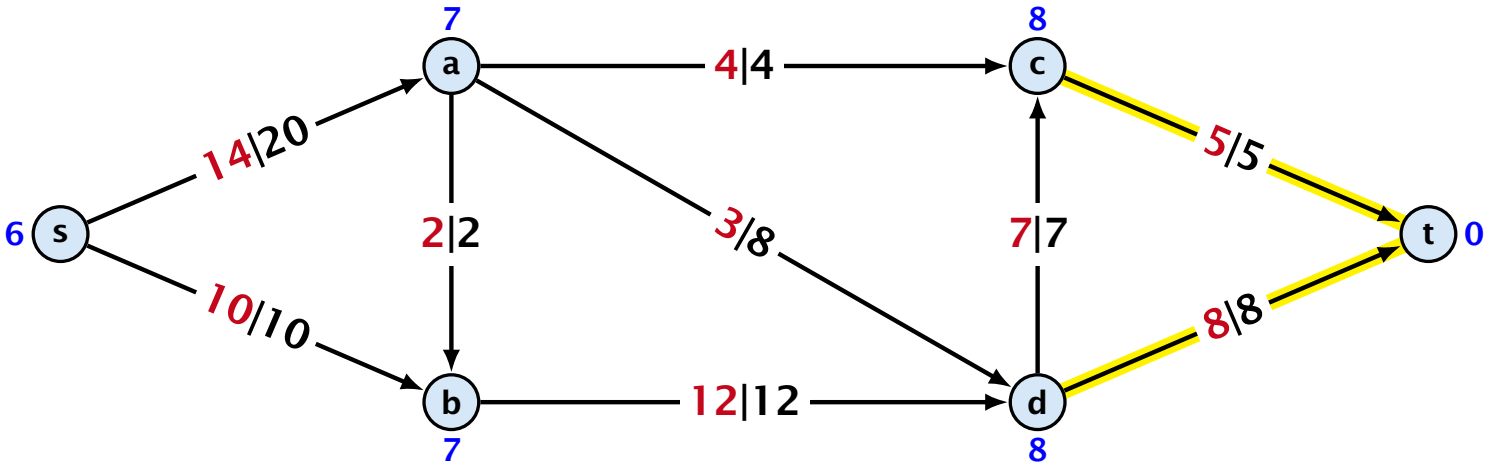
Preflow Push

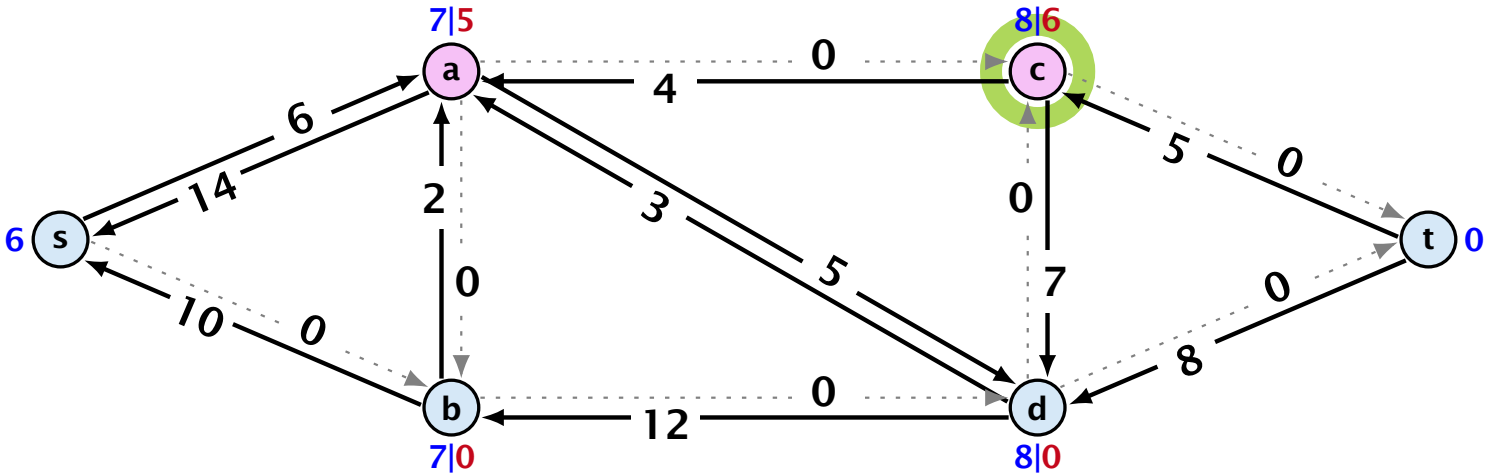


relabel to 8

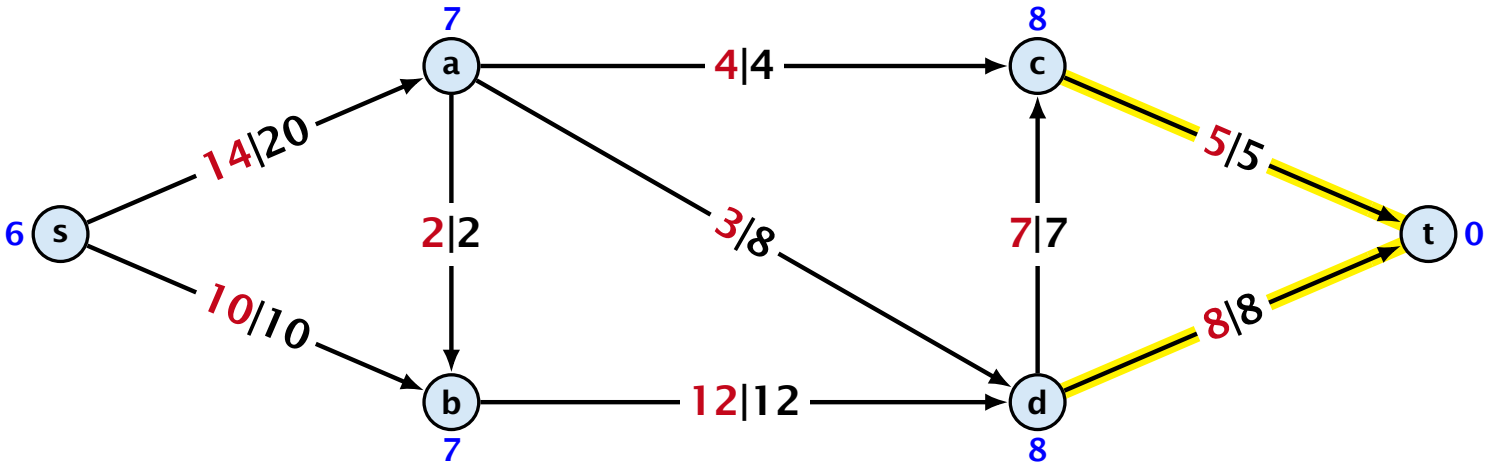


Preflow Push

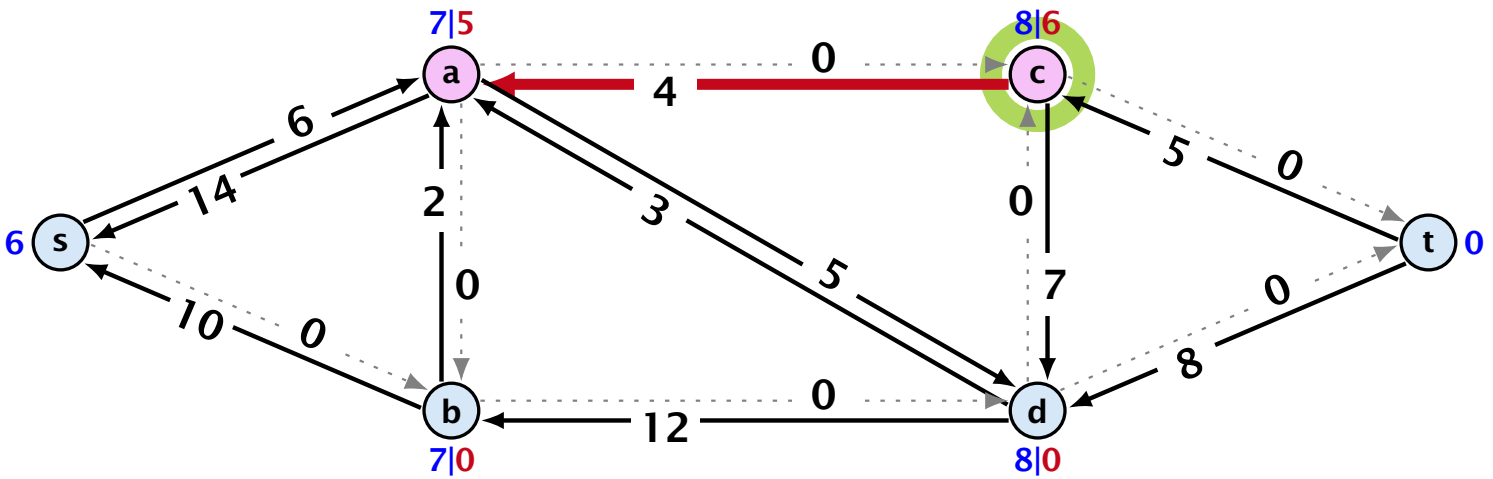




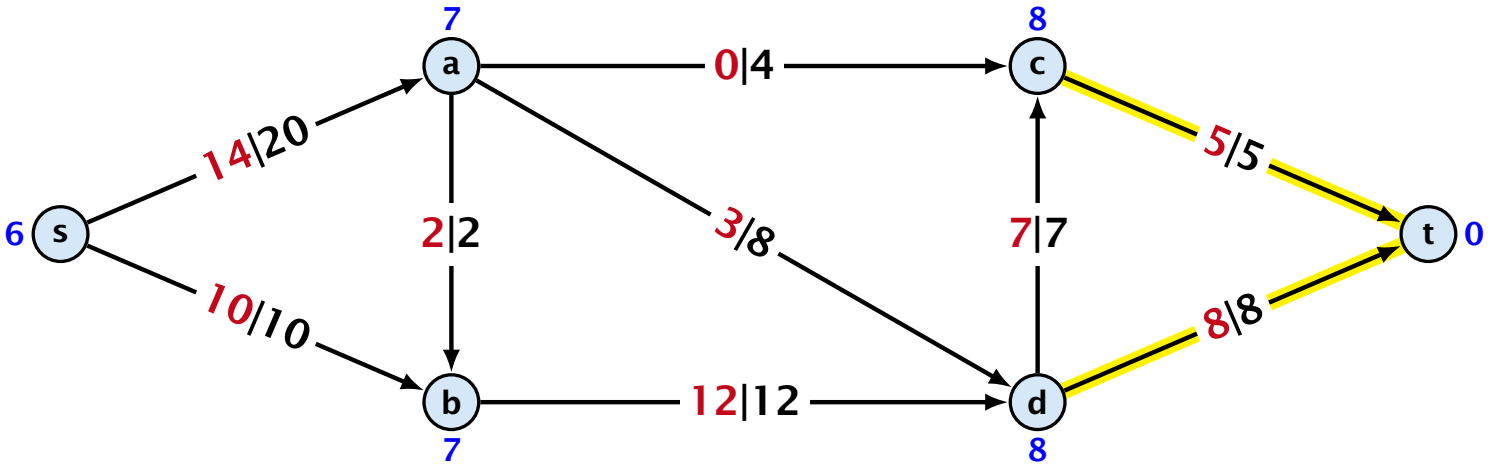
Preflow Push

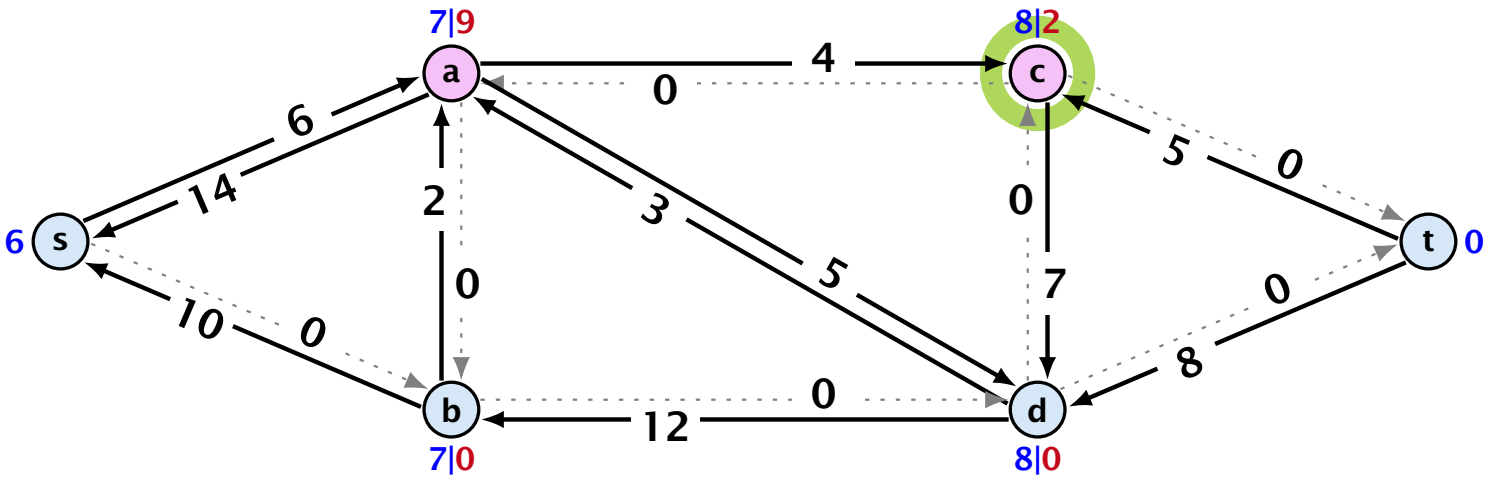


saturating push

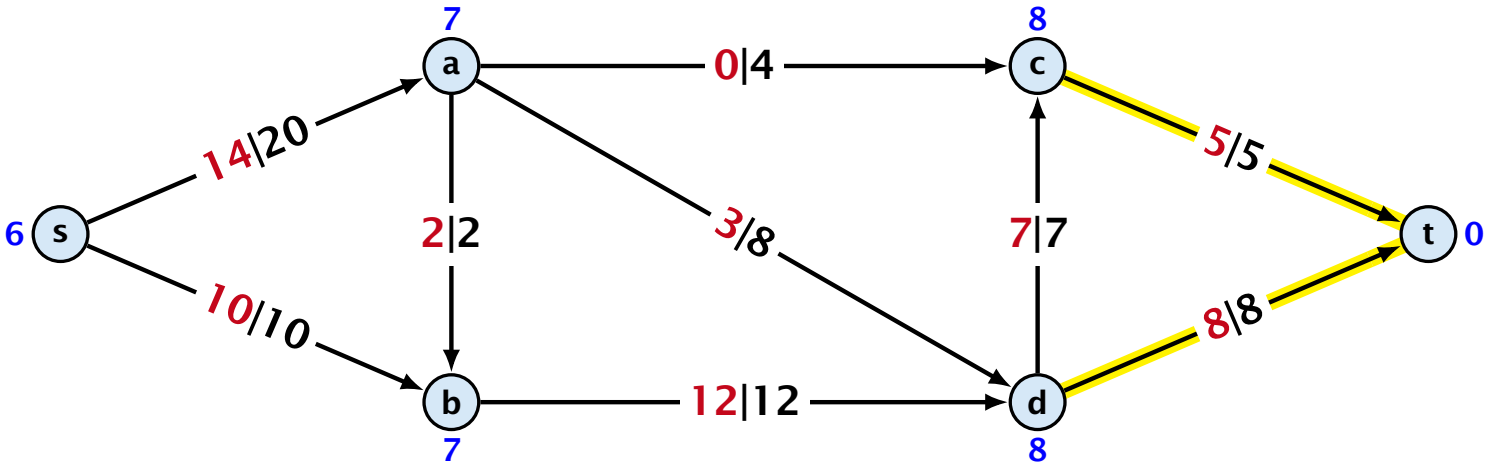


Preflow Push

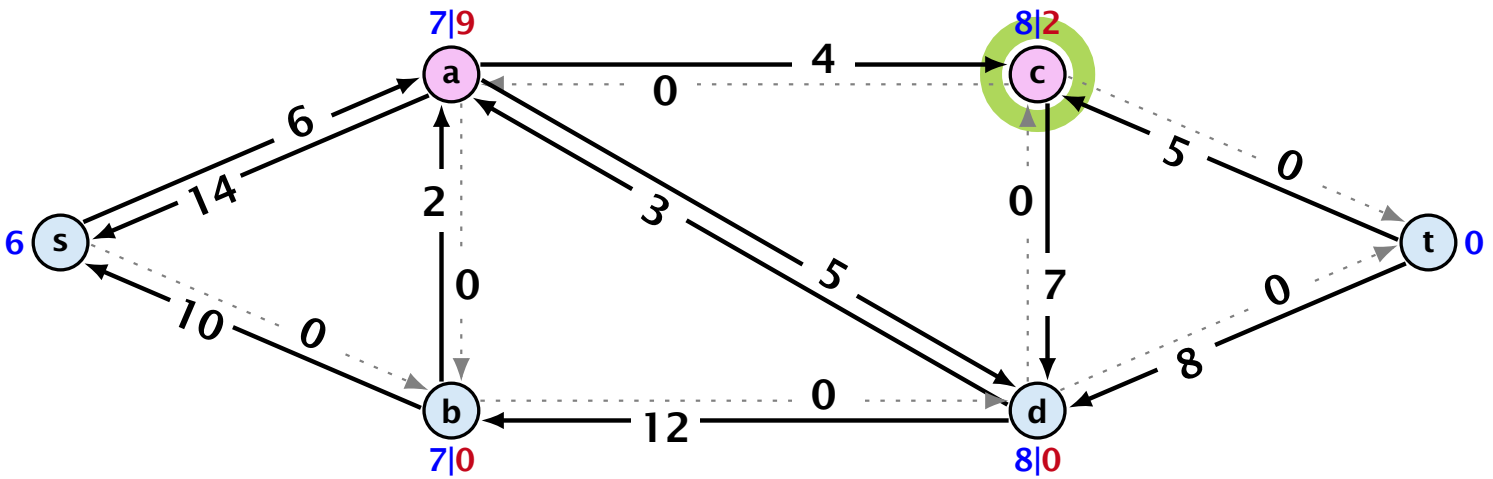




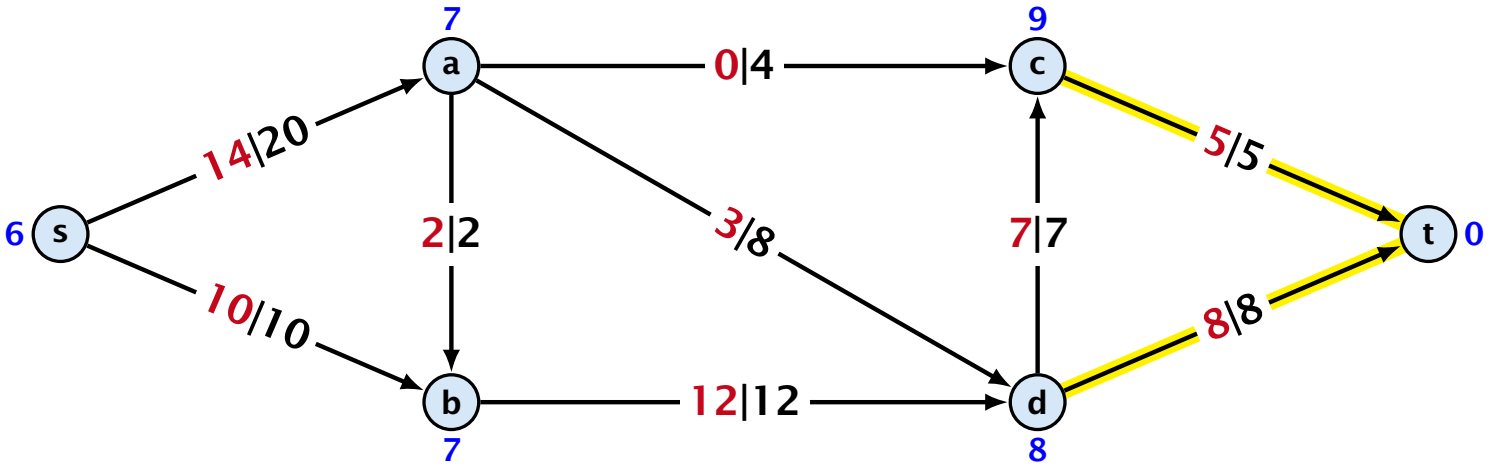
Preflow Push

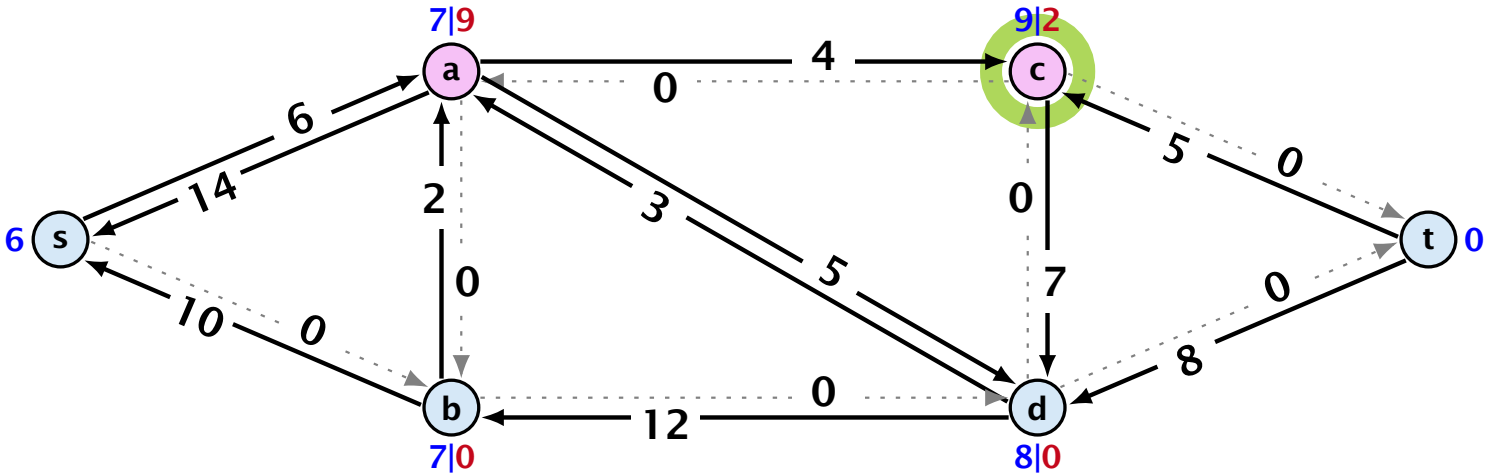


relabel to 9

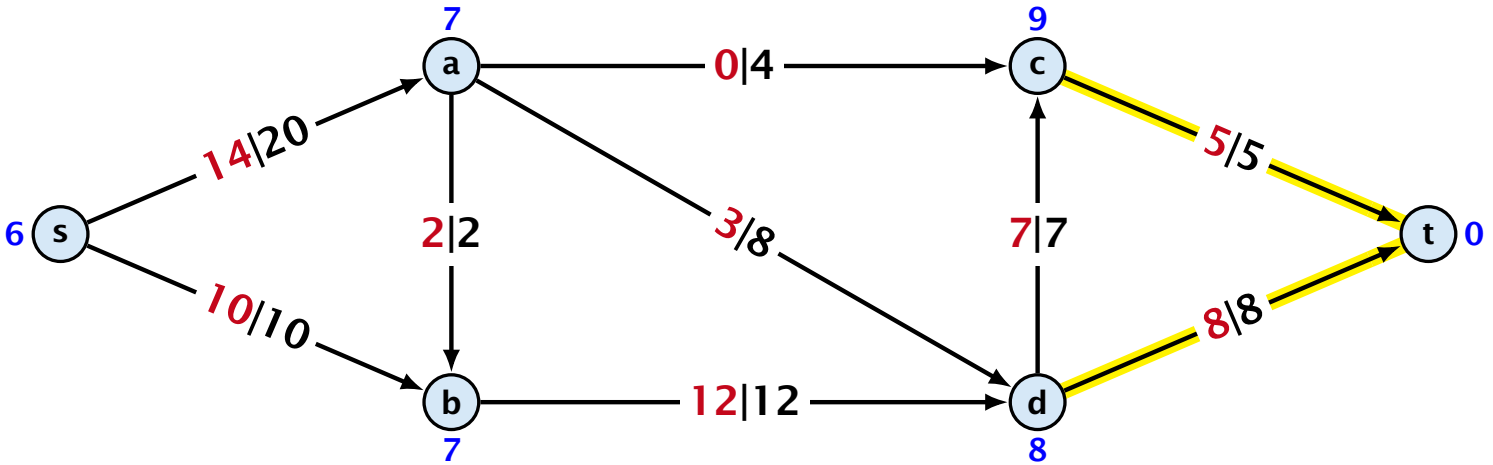


Preflow Push

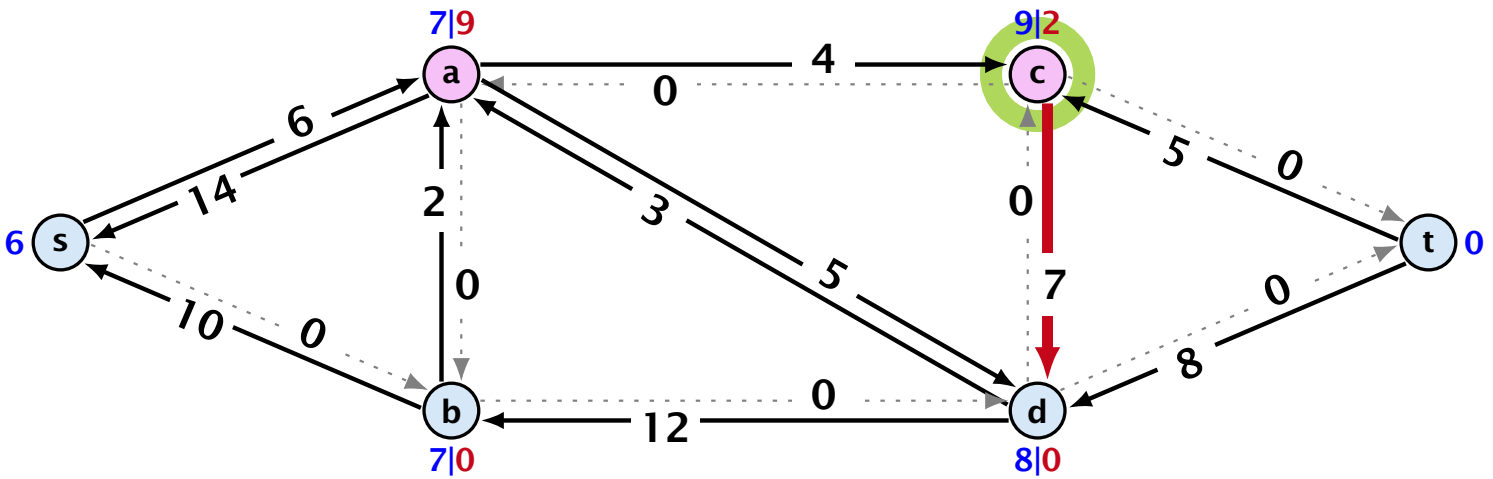




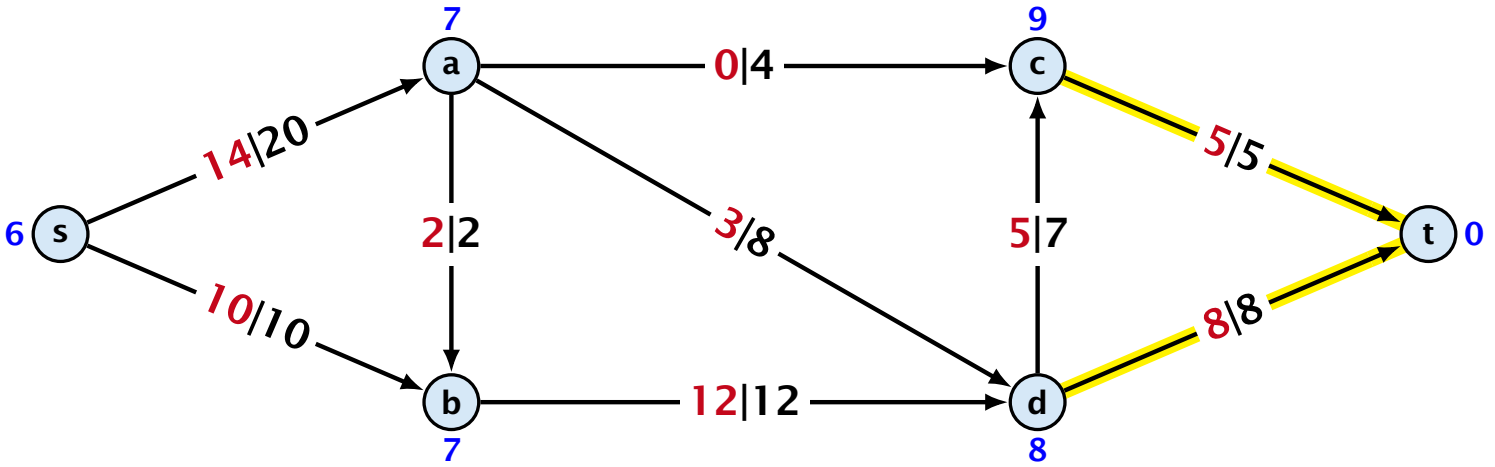
Preflow Push

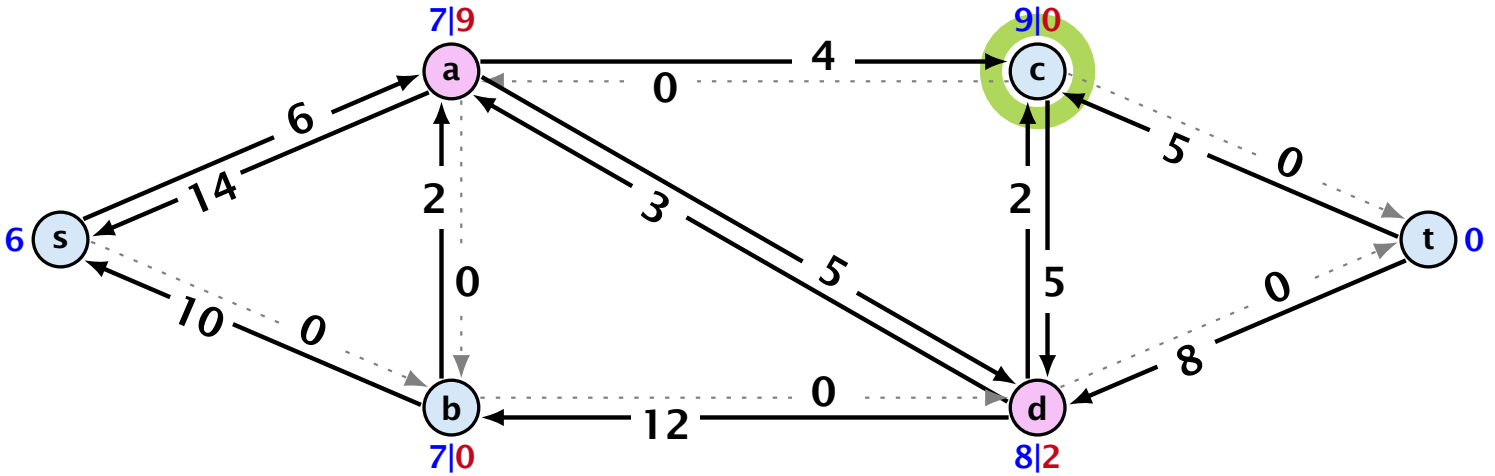


deactivating push

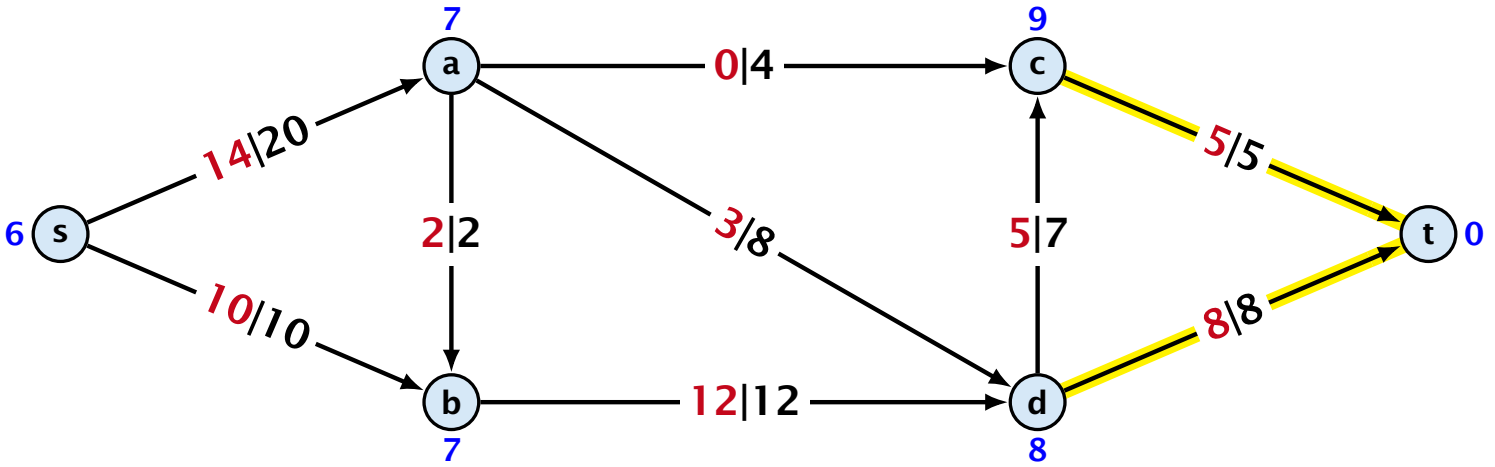


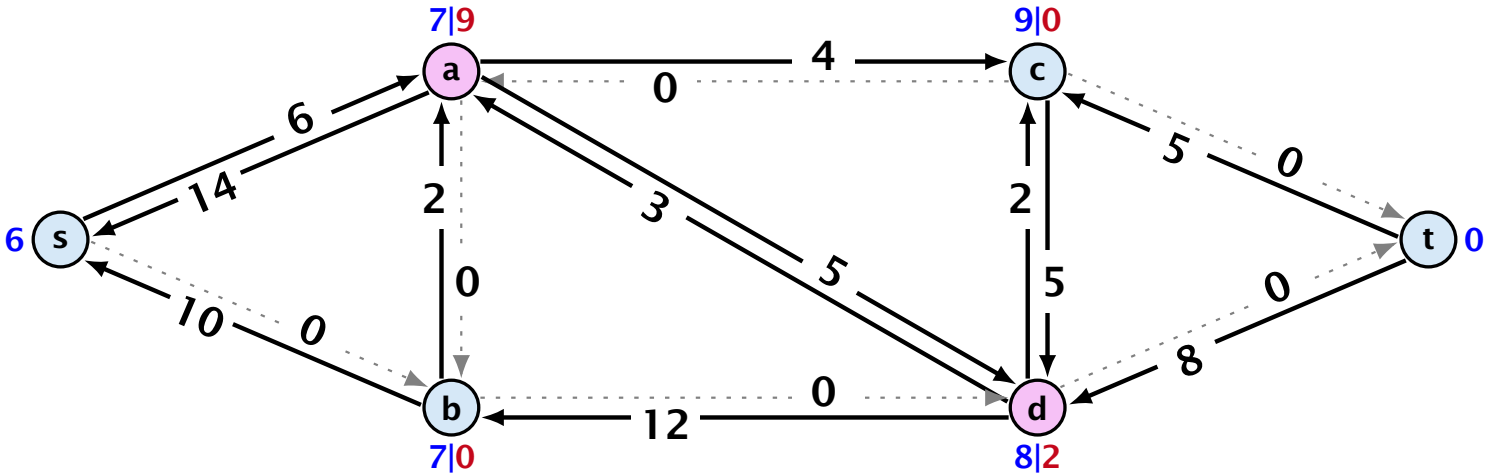
Preflow Push



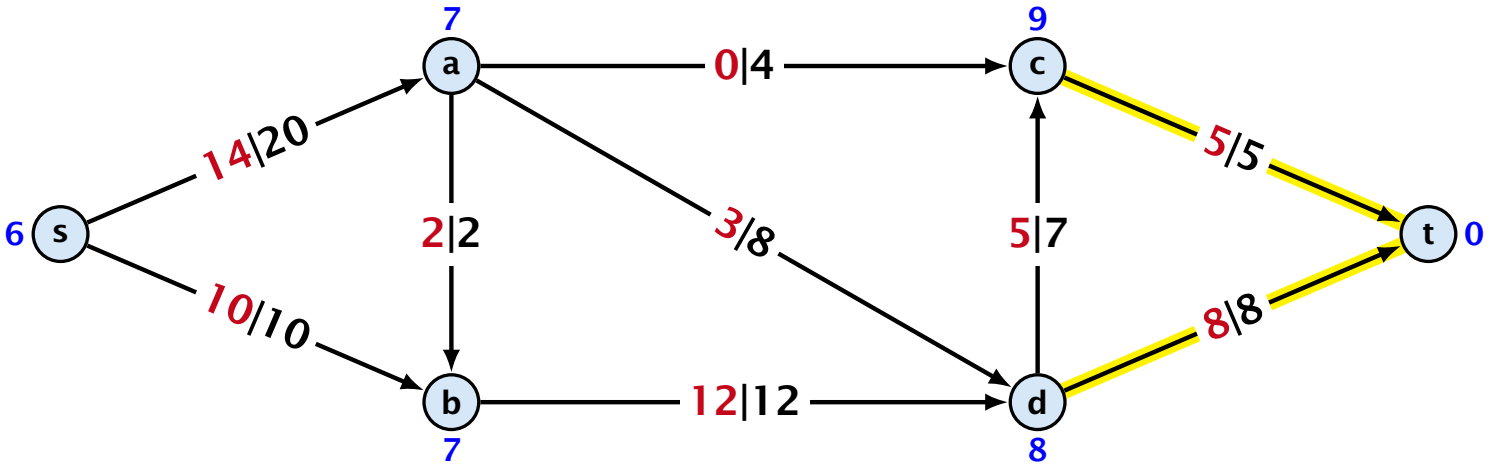


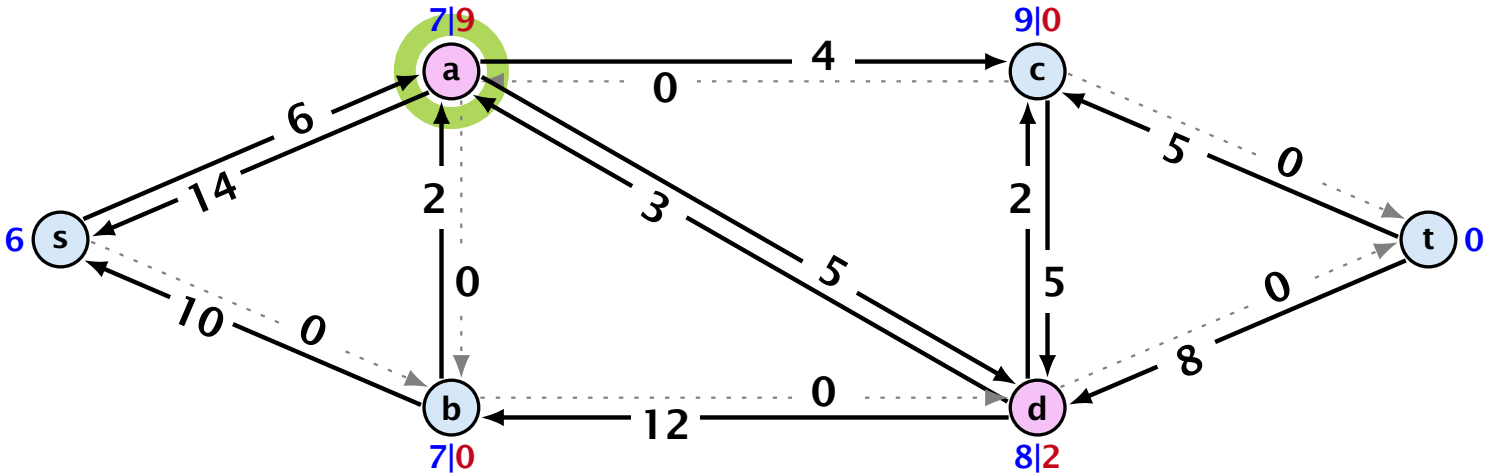
Preflow Push



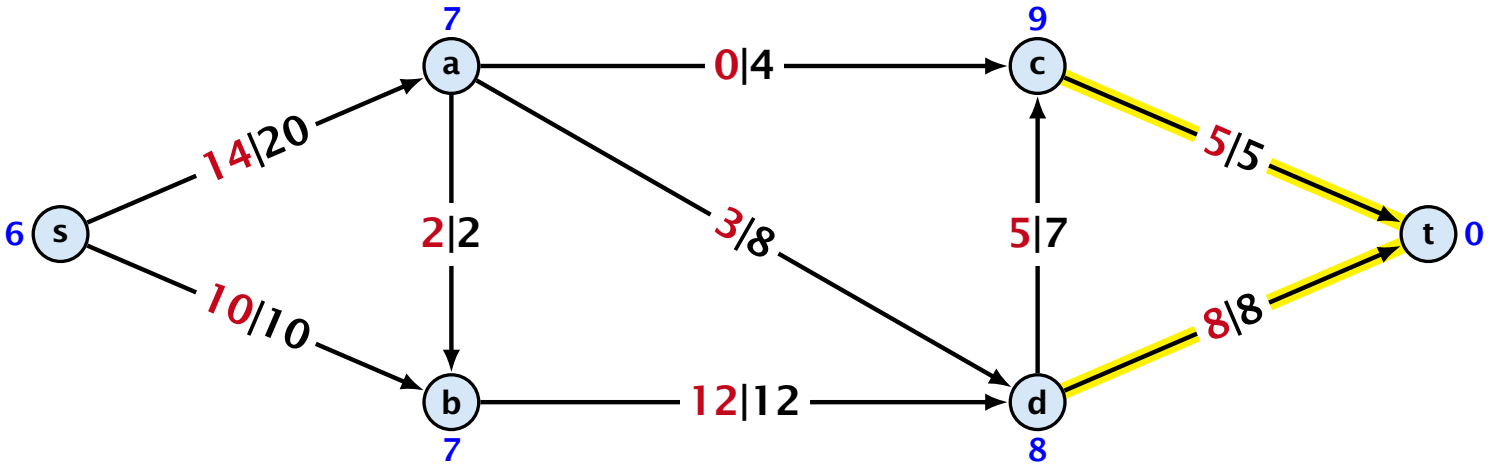


Preflow Push

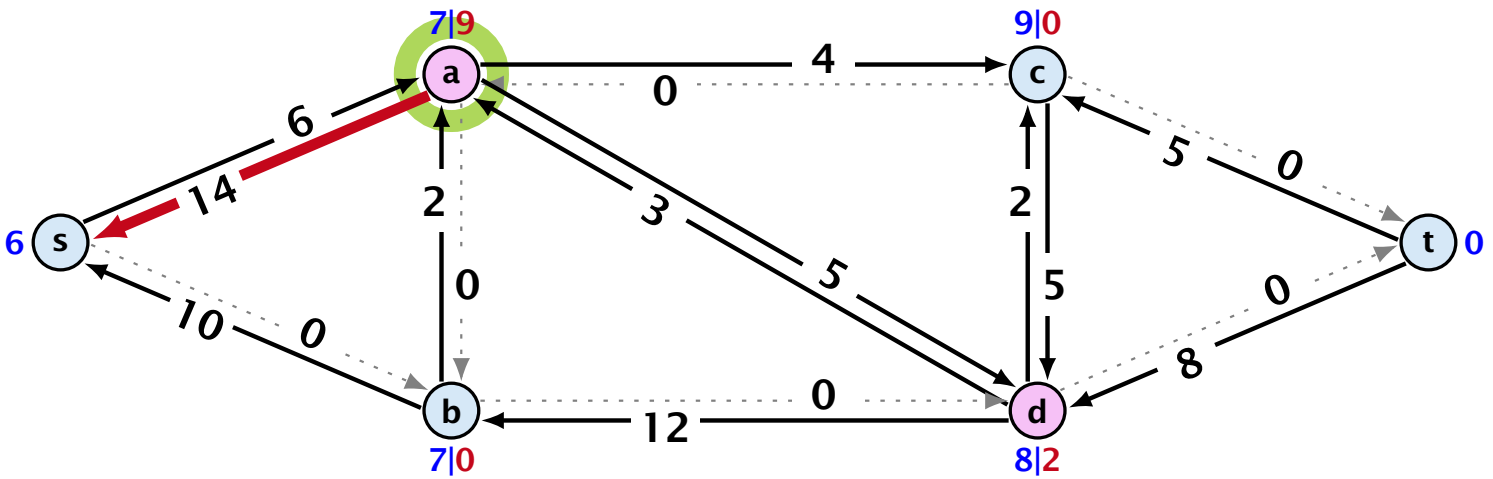




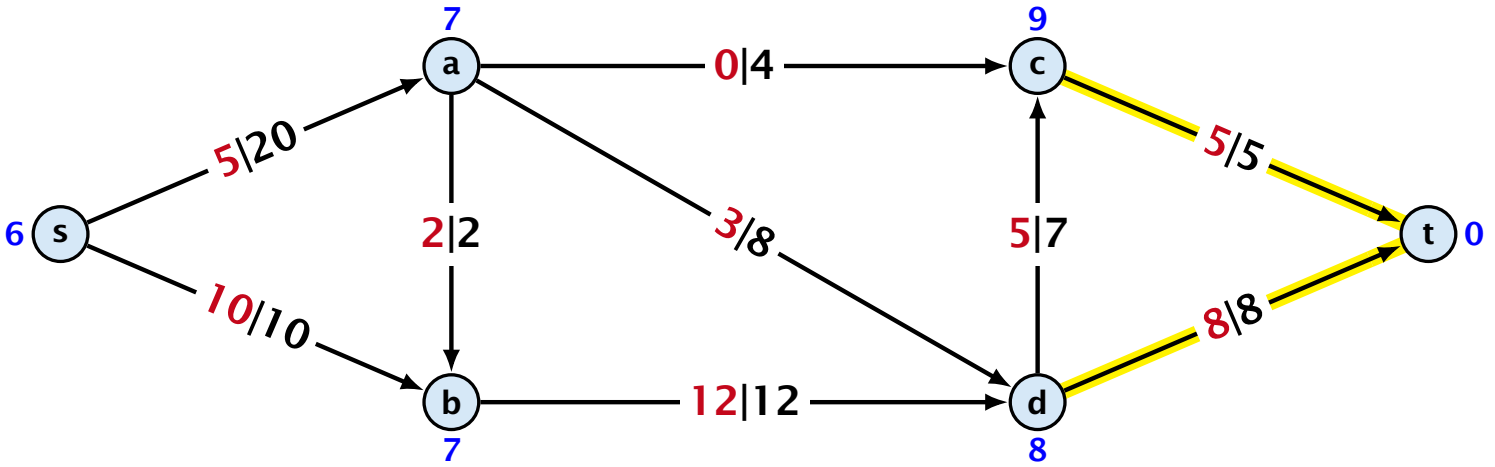
Preflow Push

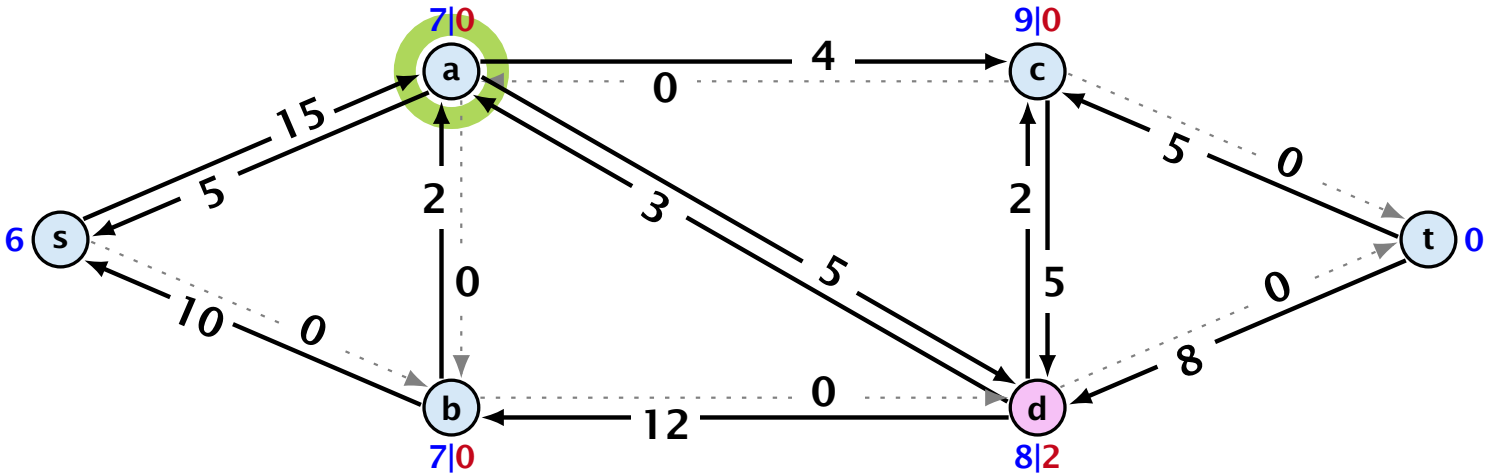


deactivating push

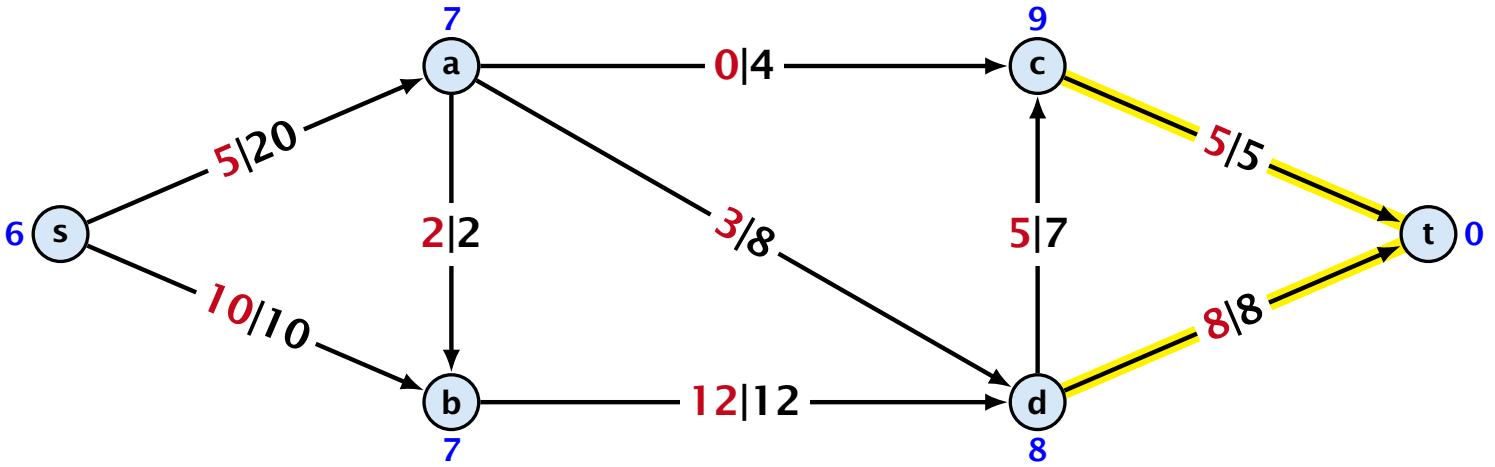


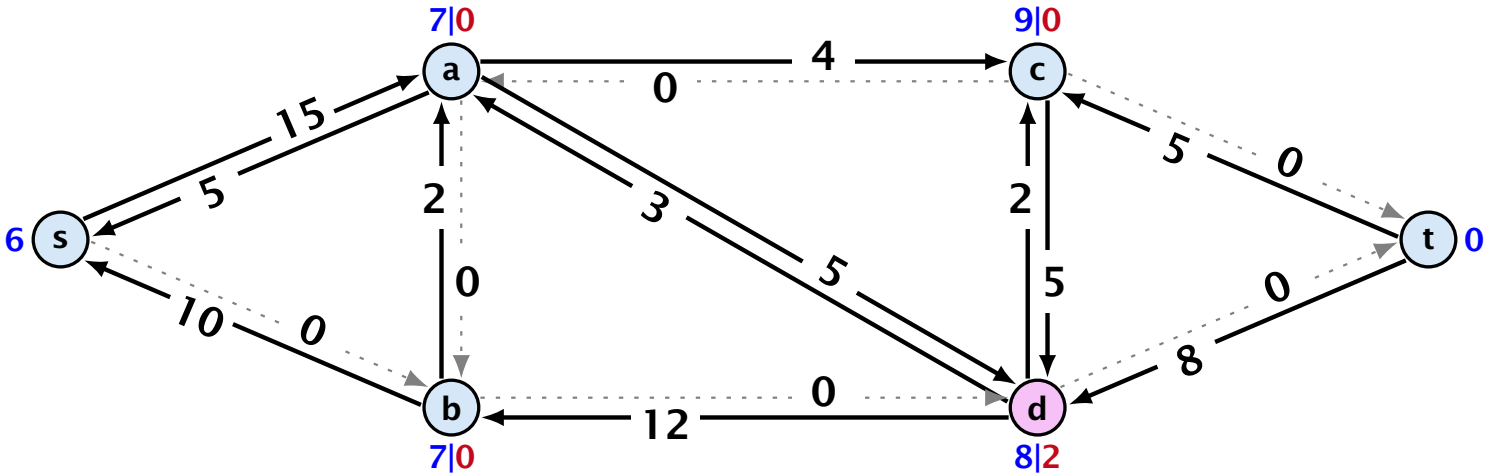
Preflow Push



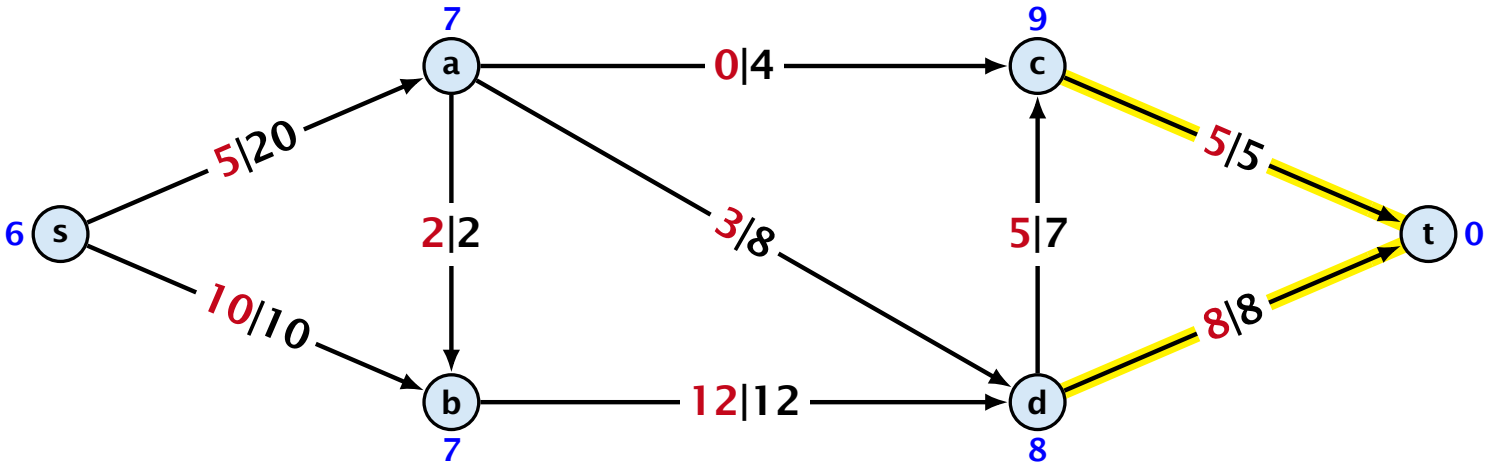


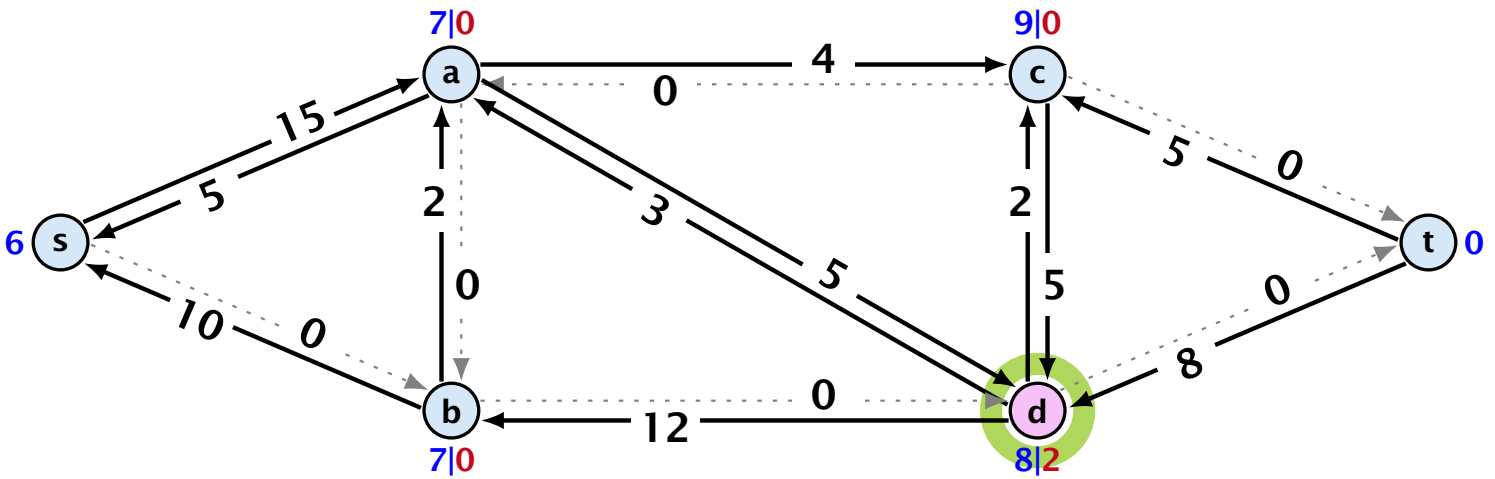
Preflow Push



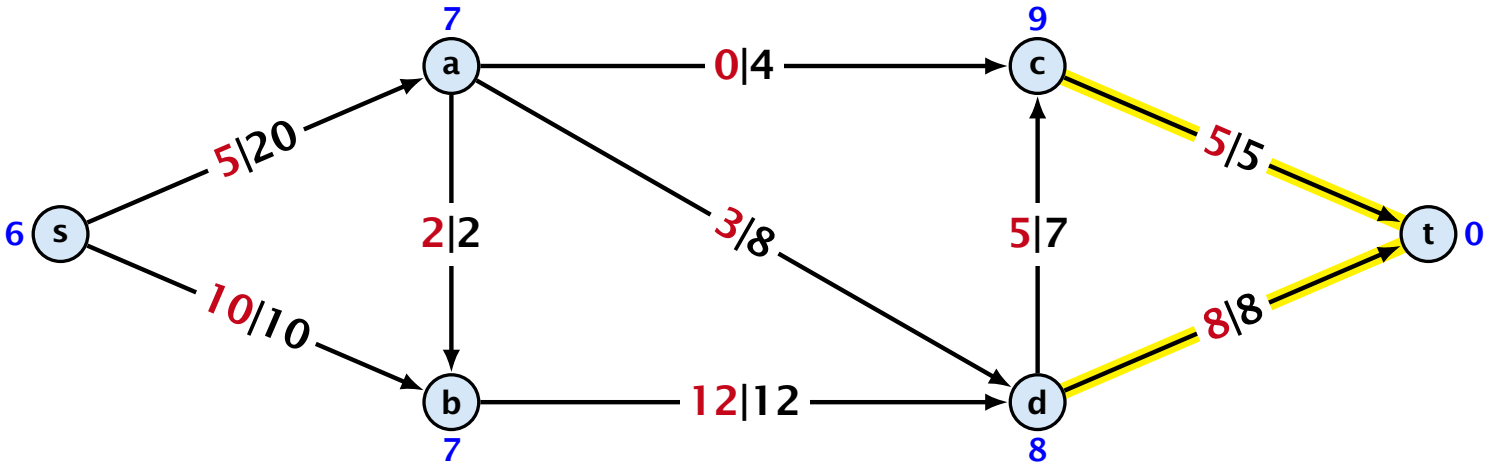


Preflow Push

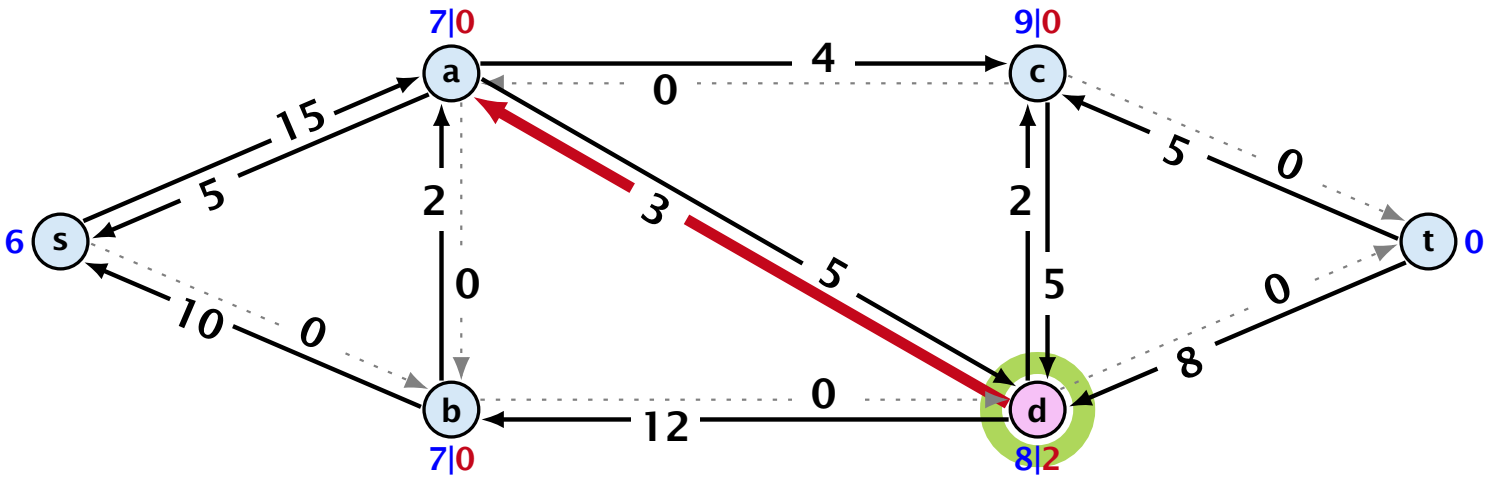




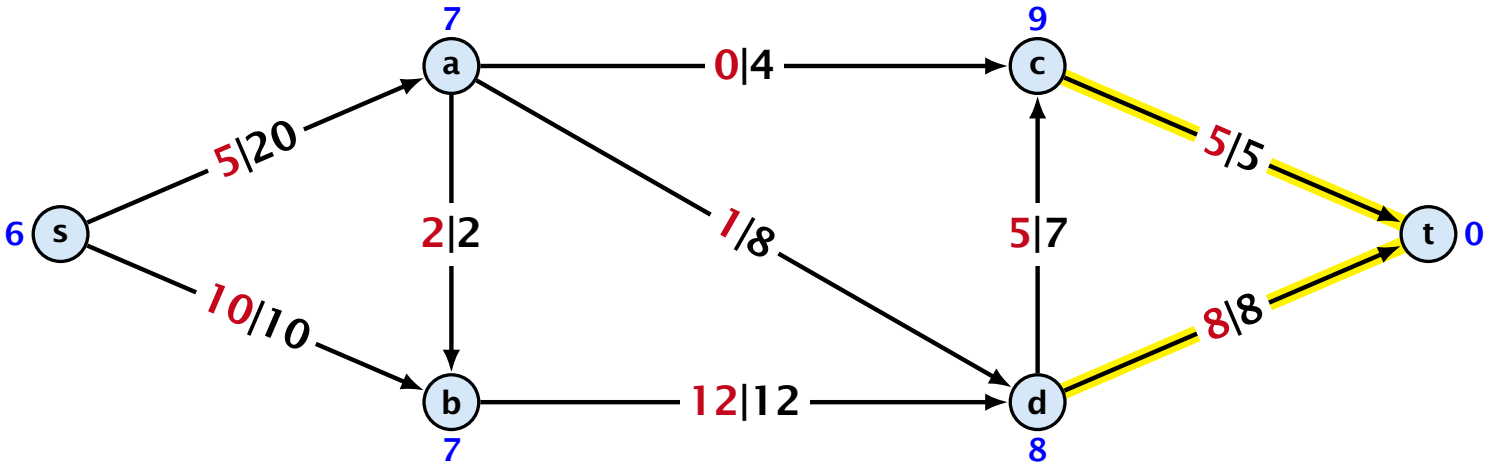
Preflow Push

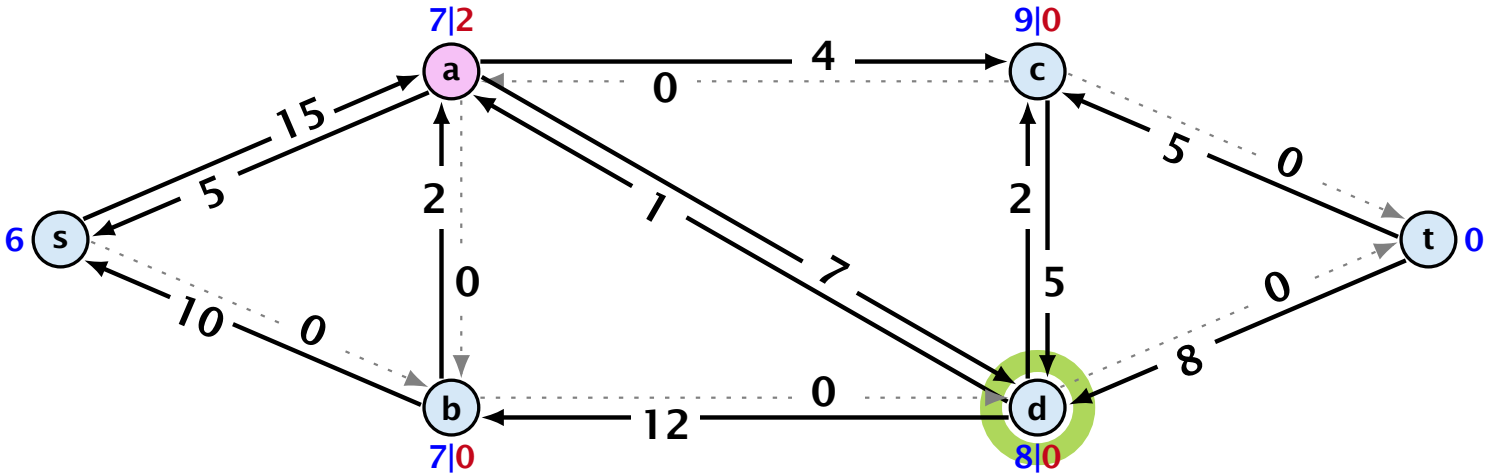


deactivating push

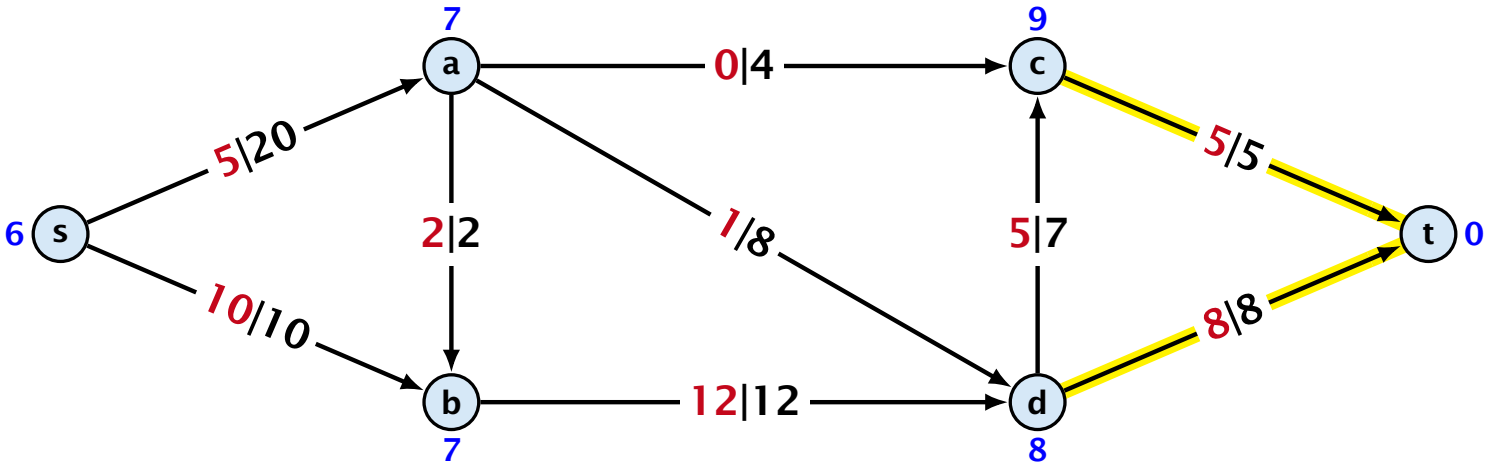


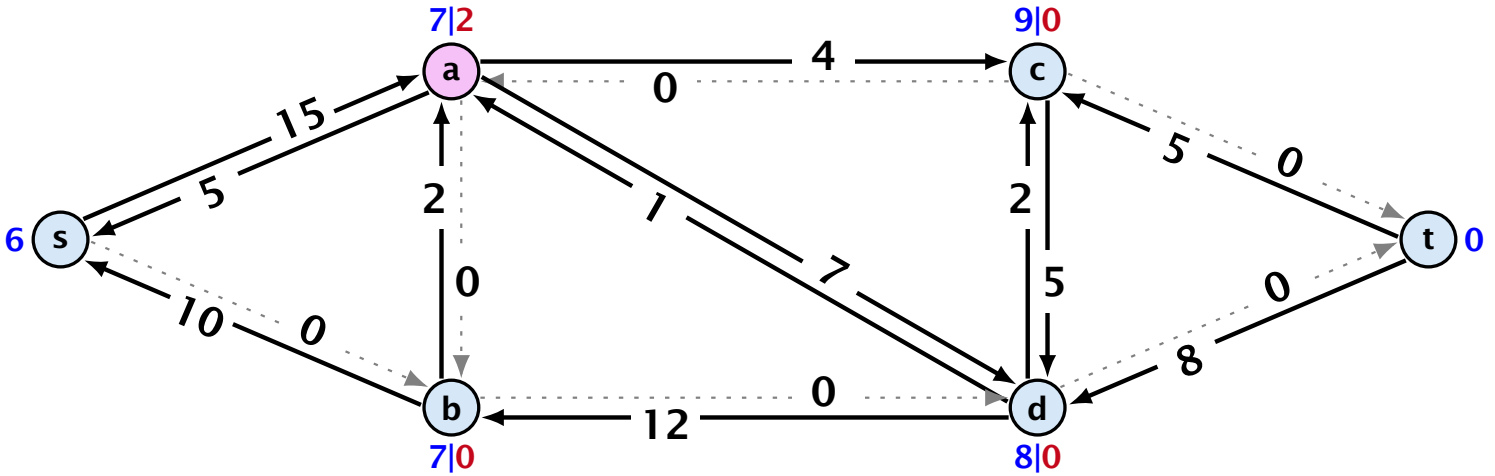
Preflow Push



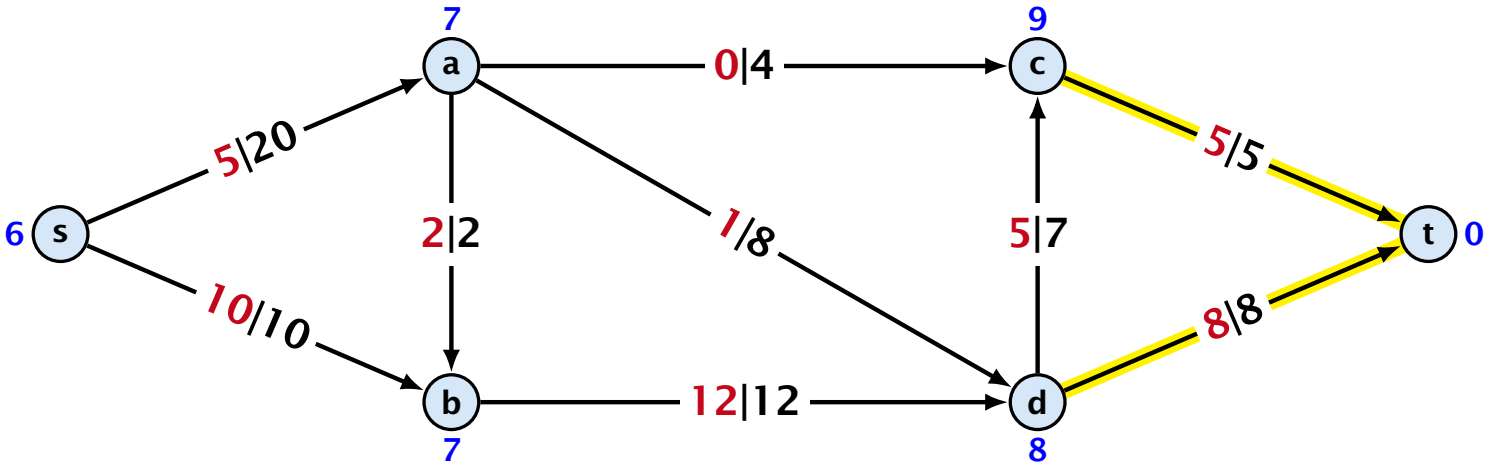


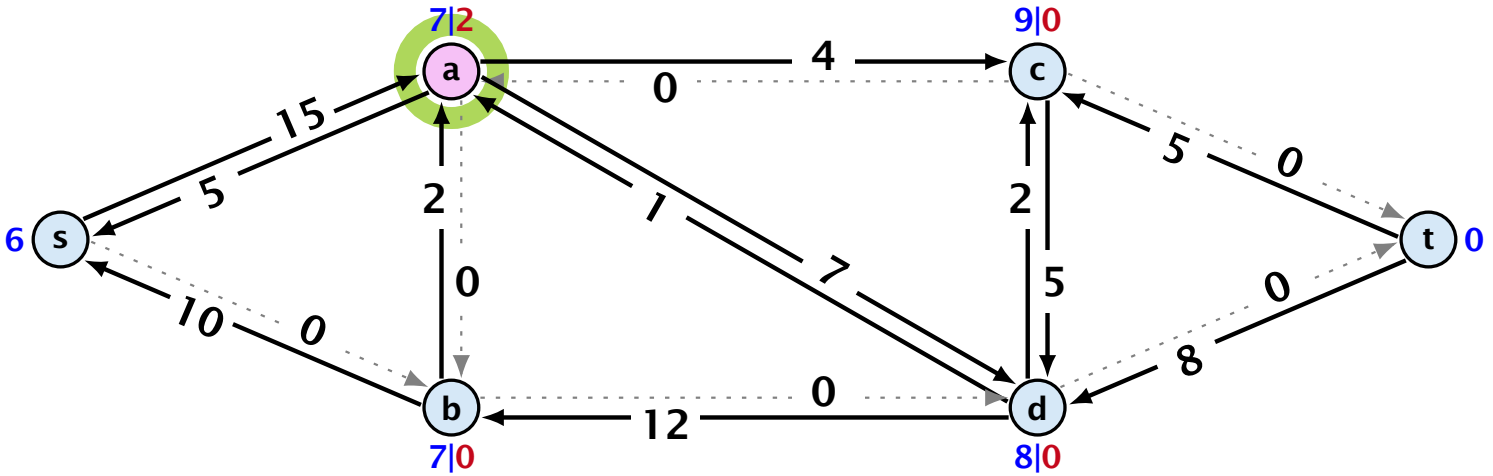
Preflow Push



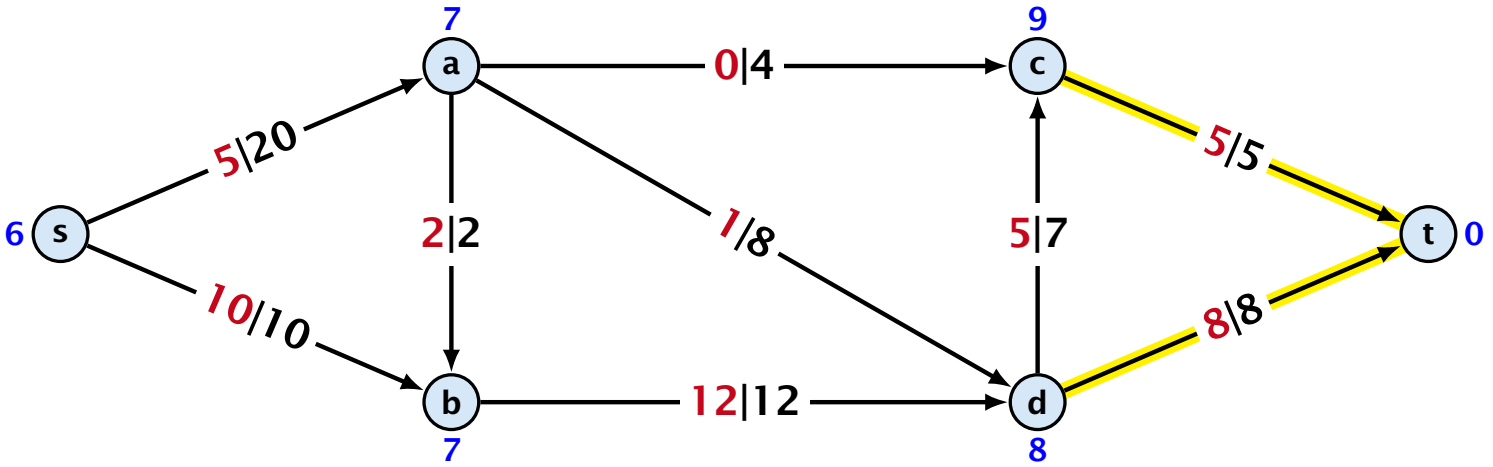


Preflow Push

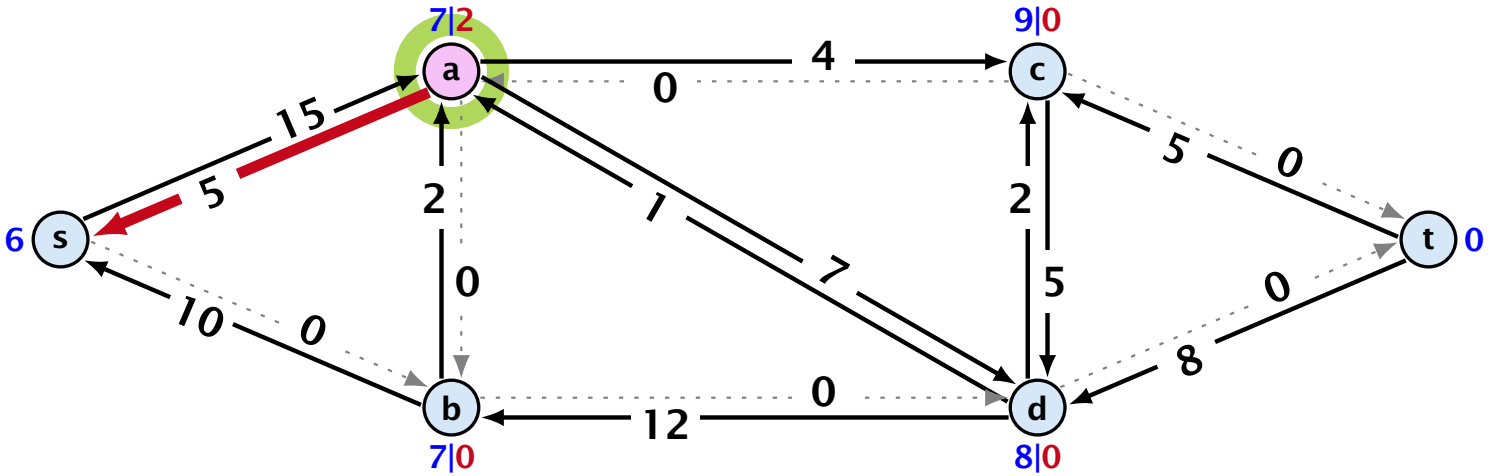




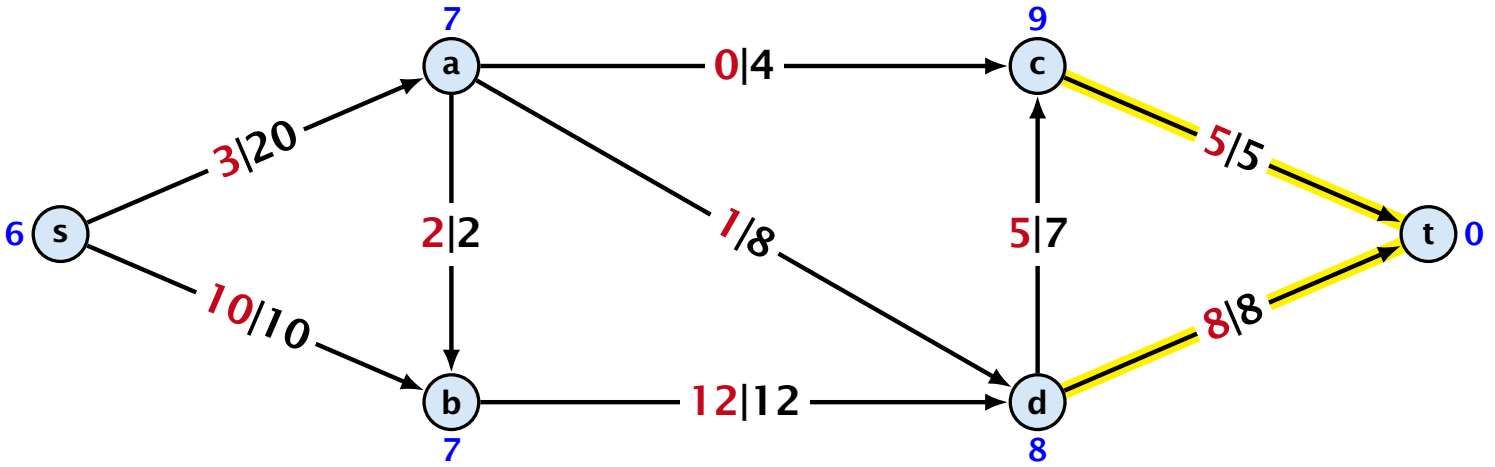
Preflow Push

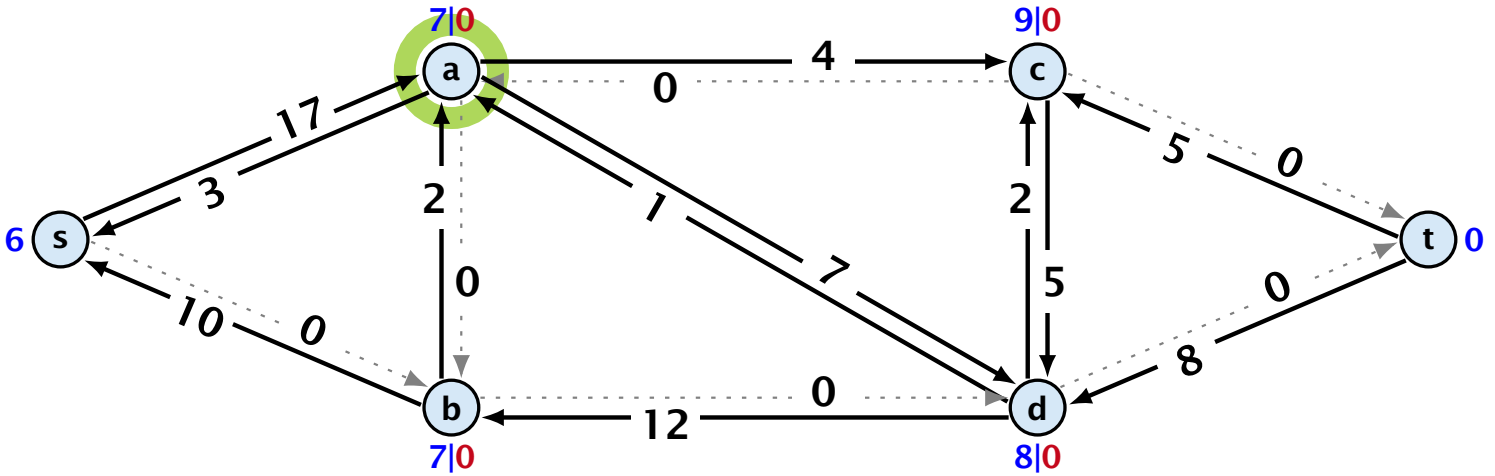


deactivating push

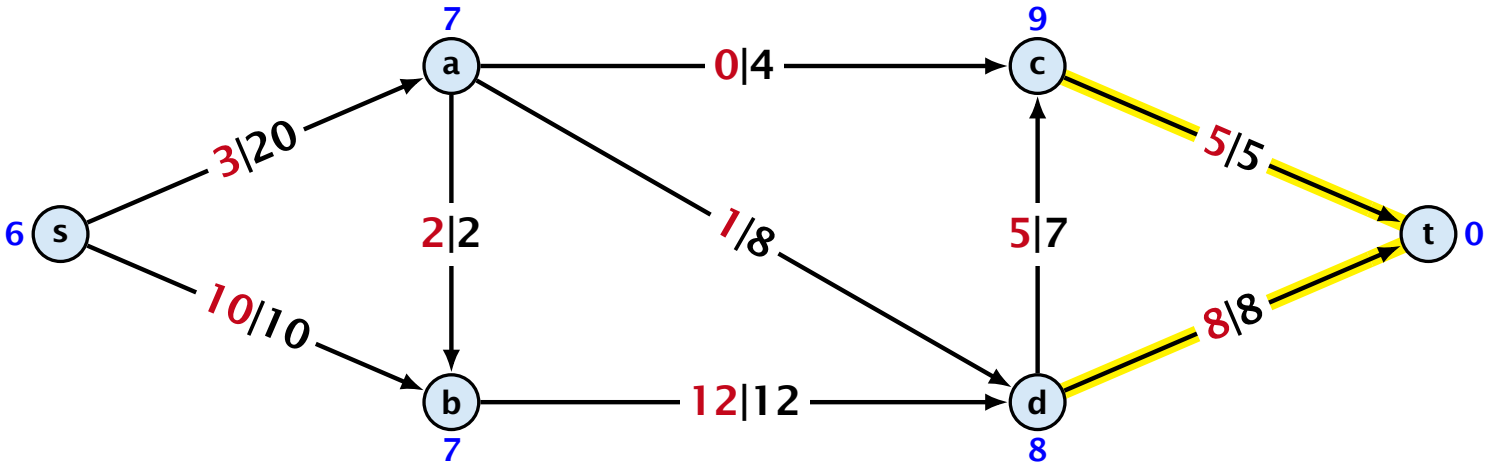


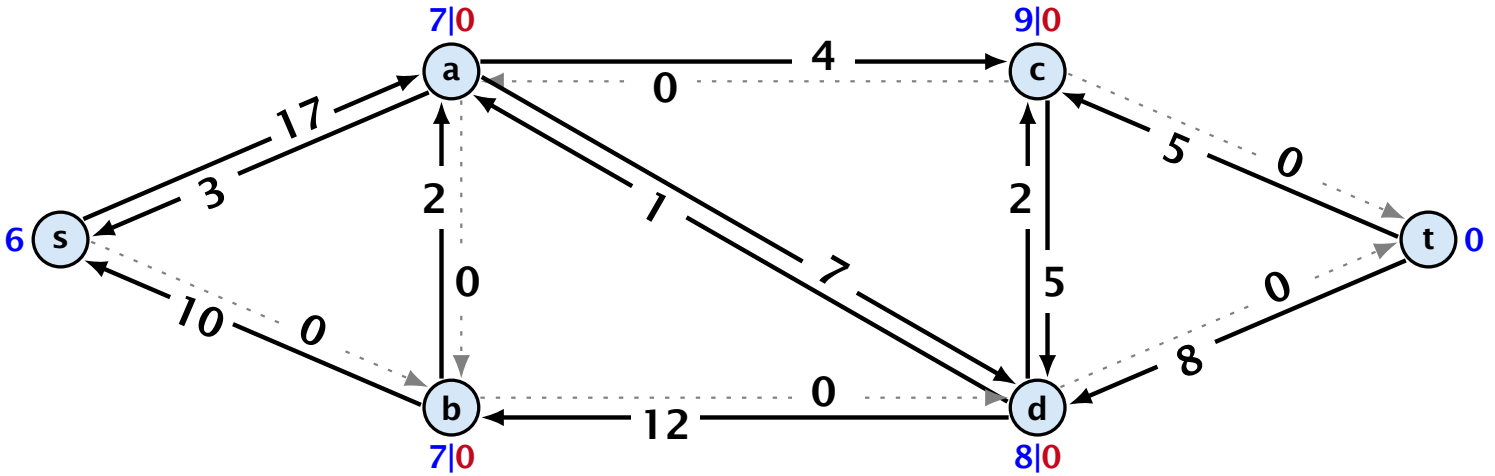
Preflow Push





Preflow Push





Analysis

Lemma 69

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

Analysis

Lemma 69

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 69

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 69

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 69

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 $f : E \to \mathbb{R}_0^+$ 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 $f : E \to \mathbb{R}_0^+$ 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 $f : E \to \mathbb{R}_0^+$ 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 $f : E \to \mathbb{R}_0^+$ 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 $f : E \to \mathbb{R}_0^+$ 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

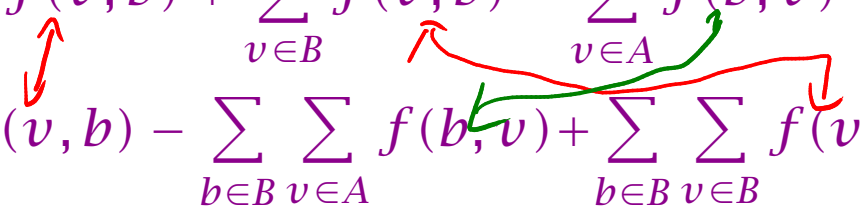
$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left(\sum_{v \in V} f(v,b) - \sum_{v \in V} f(b,v) \right) \\ &= \sum_{b \in B} \left(\sum_{v \in A} f(v,b) + \sum_{v \in B} f(v,b) - \sum_{v \in A} f(b,v) - \sum_{v \in B} f(b,v) \right) \end{aligned}$$

Let $f : E \to \mathbb{R}_0^+$ 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 $f : E \to \mathbb{R}_0^+$ 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

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

Let $f : E \to \mathbb{R}_0^+$ 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 $f : E \to \mathbb{R}_0^+$ 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

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

Let $f : E \to \mathbb{R}_0^+$ 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

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

Let $f : E \to \mathbb{R}_0^+$ 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 $f : E \to \mathbb{R}_0^+$ 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

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

Let $f : E \to \mathbb{R}_0^+$ 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

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

Let $f : E \to \mathbb{R}_0^+$ 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

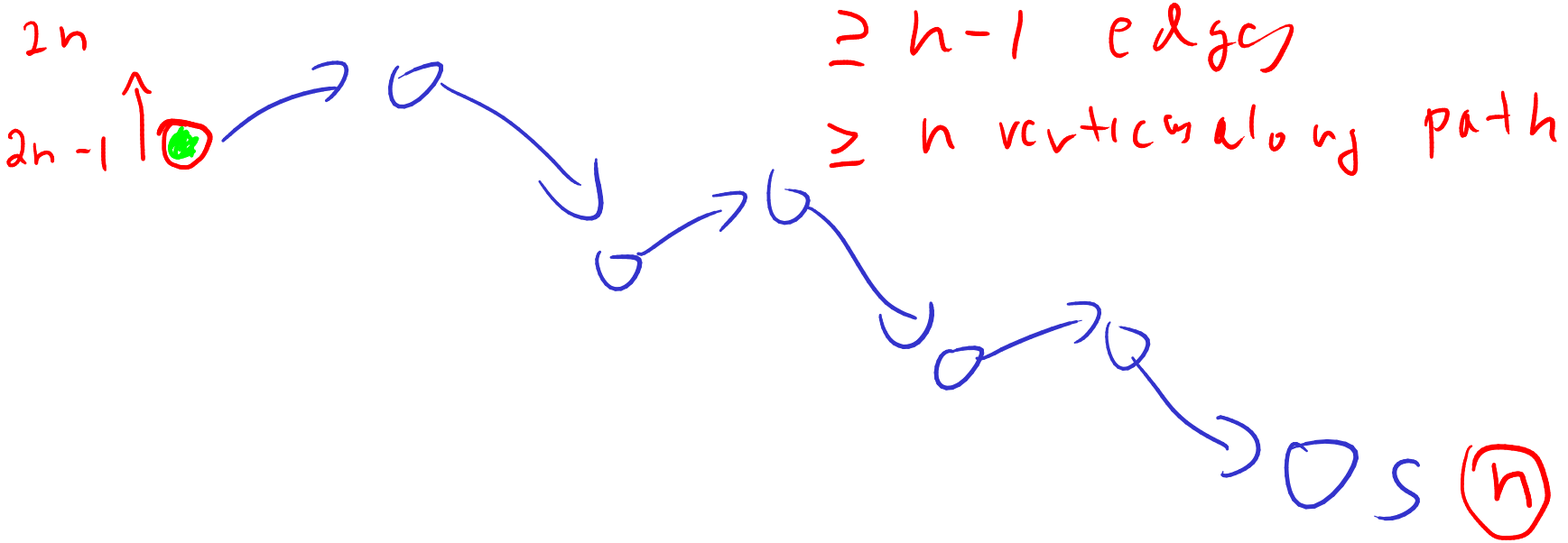
$$\begin{aligned} f(B) &= \sum_{b \in B} f(b) \\ &= \sum_{b \in B} \left(\sum_{v \in V} f(v,b) - \sum_{v \in V} f(b,v) \right) \\ &= \sum_{b \in B} \left(\sum_{v \in A} f(v,b) + \sum_{v \in B} f(v,b) - \sum_{v \in A} f(b,v) - \sum_{v \in B} f(b,v) \right) \\ &= - \sum_{b \in B} \sum_{v \in A} f(b,v)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

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

Analysis

Lemma 70

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



Analysis

Lemma 70

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 70

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 71

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

Analysis

Lemma 72

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

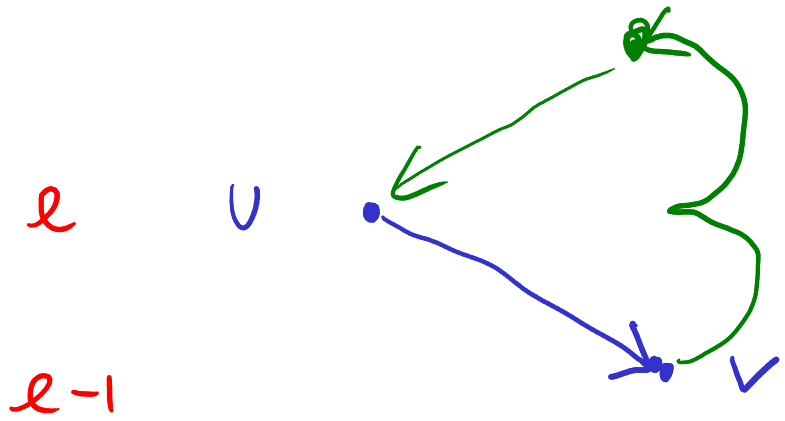
Analysis

Lemma 72

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 72

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 72

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 72

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 72

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 72

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 73

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

Lemma 73

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

Proof.

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

Lemma 73

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

Proof.

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

Lemma 73

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

Proof.

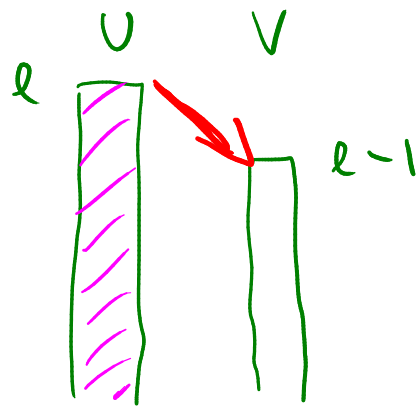
- Define a potential function $\Phi(f) = \sum_{\text{active nodes} v} \ell(v)$
- A saturating push increases Φ by $\leq 2n$ (when the target node becomes active it may contribute at most $2n$ to the sum).
- A relabel increases Φ by at most 1.

Lemma 73

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

Proof.

- Define a potential function $\Phi(f) = \sum_{\text{active nodes} v} \ell(v)$
- A saturating push increases Φ by $\leq 2n$ (when the target node becomes active it may contribute at most $2n$ to the sum).
- A relabel increases Φ by at most 1.
- A deactivating push decreases Φ 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 73

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

Proof.

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

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

Analysis

Theorem 74

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

#pushes + n #relabels

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 v
- 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 $\Theta(n)$

- 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 v
- check whether (u, v) needs to be deleted (saturating 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 $\Theta(n)$

- 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 $\Theta(n)$

- 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

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 $\Theta(n)$

- 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

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:

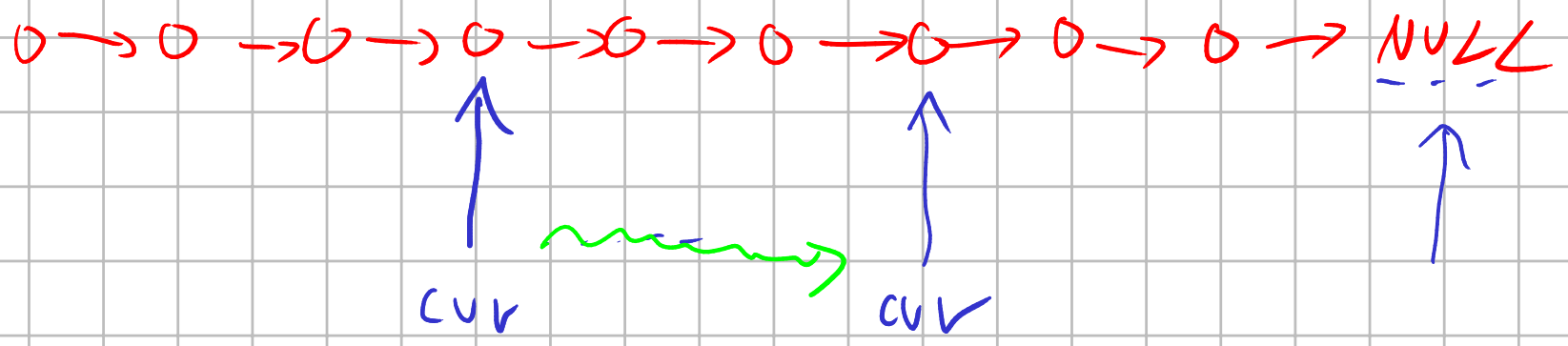
Algorithm 20 discharge(u)

```
1: while u is active do
2:     v ← u.current-neighbour
3:     if v = null then
4:         relabel(u)
5:         u.current-neighbour ← u.neighbour-list-head
6:     else
7:         if (u, v) admissible then push(u, v)
8:         else u.current-neighbour ← v.next-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.

vertex # v

deg(v) + 1



Lemma 75

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

Proof.

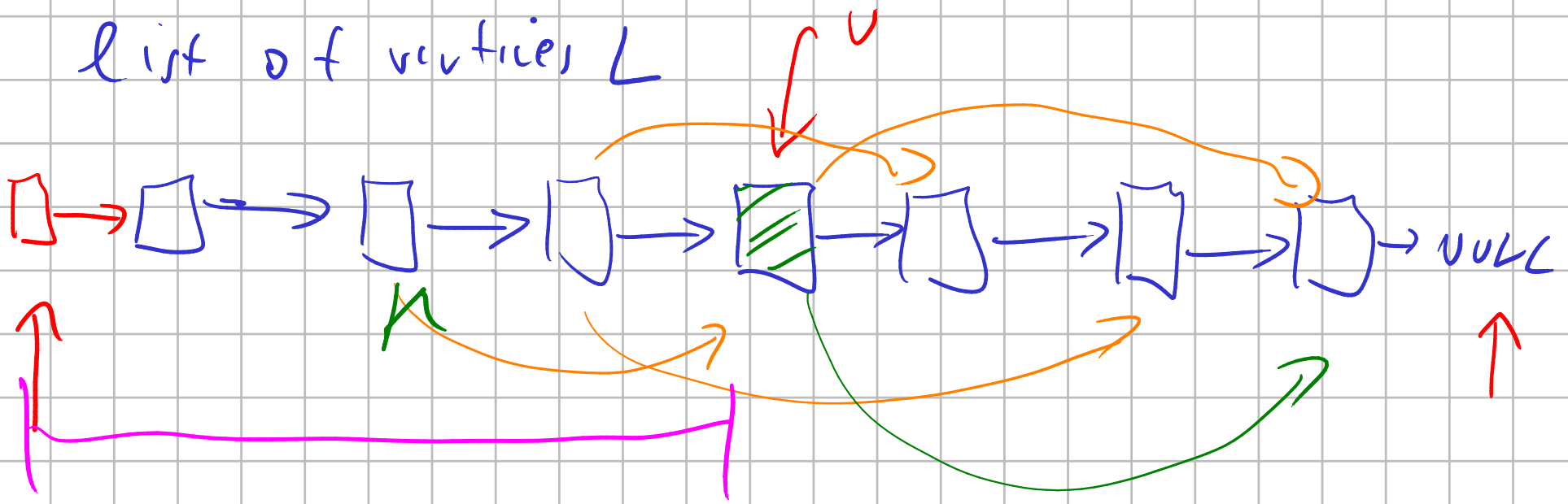
- While pushing from u the current-neighbour pointer is only advanced if the current edge is not admissible.
- The only thing that could make the edge admissible again would be a relabel at u.
- If we reach the end of the list ($v = \text{null}$) all edges are not admissible. □

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

13.2 Relabel to Front

Algorithm 21 relabel-to-front(G, s, t)

```
 1: initialize preflow
 2: initialize node list L containing V \ {s,t} in any order
 3: foreach u ∈ V \ {s,t} do
 4:         u.current-neighbour ← u.neighbour-list-head
 5: u ← L.head
 6: while u ≠ null do
 7:         old-height ← ℓ(u)
 8:         discharge(u)
 9:         if ℓ(u) > old-height then // relabel happened
10:             move u to the front of L
11:         u ← u.next
```


list of vertices L
v
u
v
u
v
L

13.2 Relabel to Front

Lemma 76 (Invariant)

In Line 6 of the relabel-to-front algorithm the following invariant holds.

1. *The sequence L is topologically sorted w.r.t. the set of admissible edges; this means for an admissible edge (x, y) the node x appears before y in sequence L.*
2. *No node before u in the list L is active.*

Proof:

- Initialization:
  1. In the beginning s has label $n \geq 2$, and all other nodes have label 0. Hence, no edge is admissible, which means that any ordering L is permitted.
  2. We start with u being the head of the list; hence no node before u can be active
- Maintenance:
  1.
     - Pushes do no create any new admissible edges. Therefore, if discharge() does not relabel u, L is still topologically sorted.
     - After relabeling, u cannot have admissible incoming edges as such an edge (x, u) would have had a difference $\ell(x) - \ell(u) \geq 2$ before the re-labeling (such edges do not exist in the residual graph).
       Hence, moving u to the front does not violate the sorting property for any edge; however it fixes this property for all admissible edges leaving u that were generated by the relabeling.

13.2 Relabel to Front

Proof:

- Maintenance:
  2. If we do a relabel there is nothing to prove because the only node before u' (u in the next iteration) will be the current u; the discharge(u) operation only terminates when u is not active anymore.

     For the case that we do not relabel, observe that the only way a predecessor could be active is that we push flow to it via an admissible arc. However, all admissible arc point to successors of u.

Note that the invariant means that for $u = \text{null}$ we have a preflow with a valid labelling that does not have active nodes. This means we have a maximum flow.