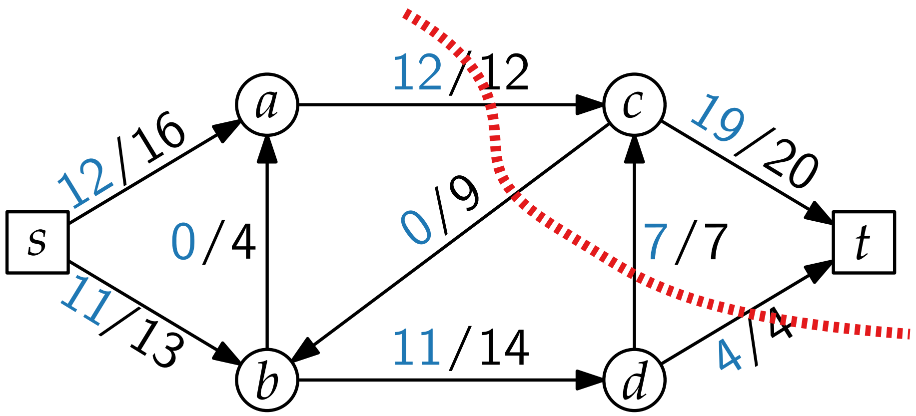
Julius-Maximilians-UNIVERSITÄT WÜRZBURG

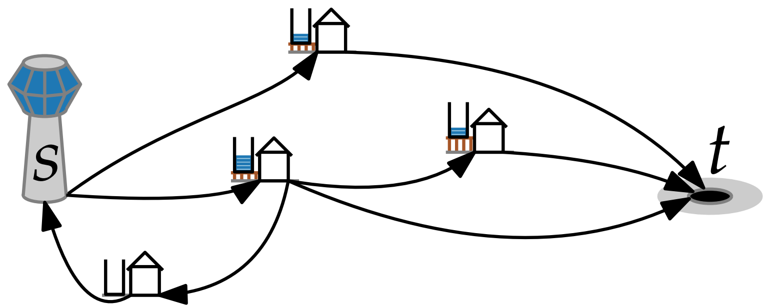
Advanced Algorithms

Maximum Flow Problem

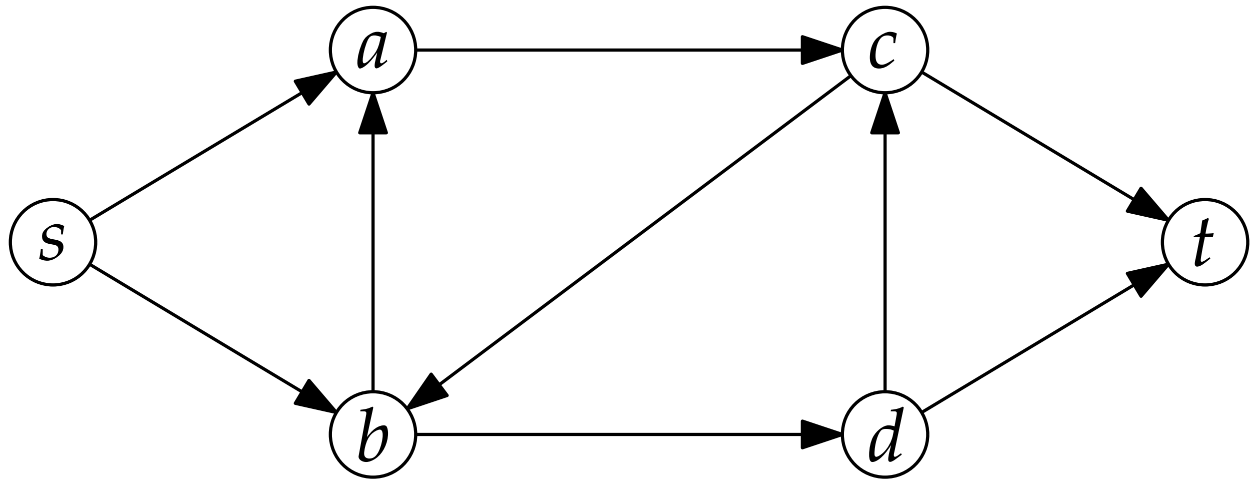
Push–Relabel Algorithm

Alexander Wolff · WS 2022



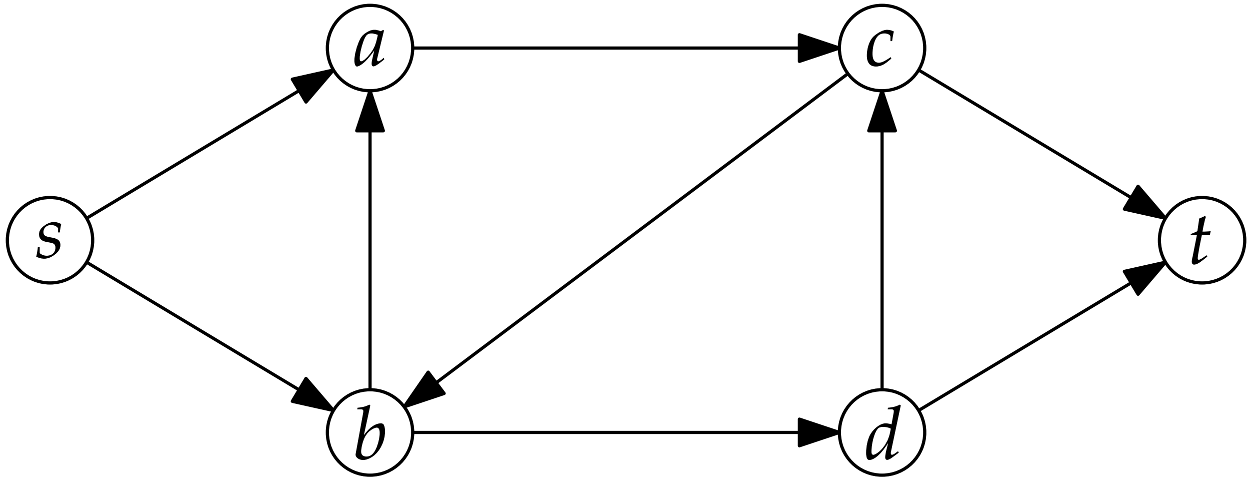


Flow Networks



Flow Networks

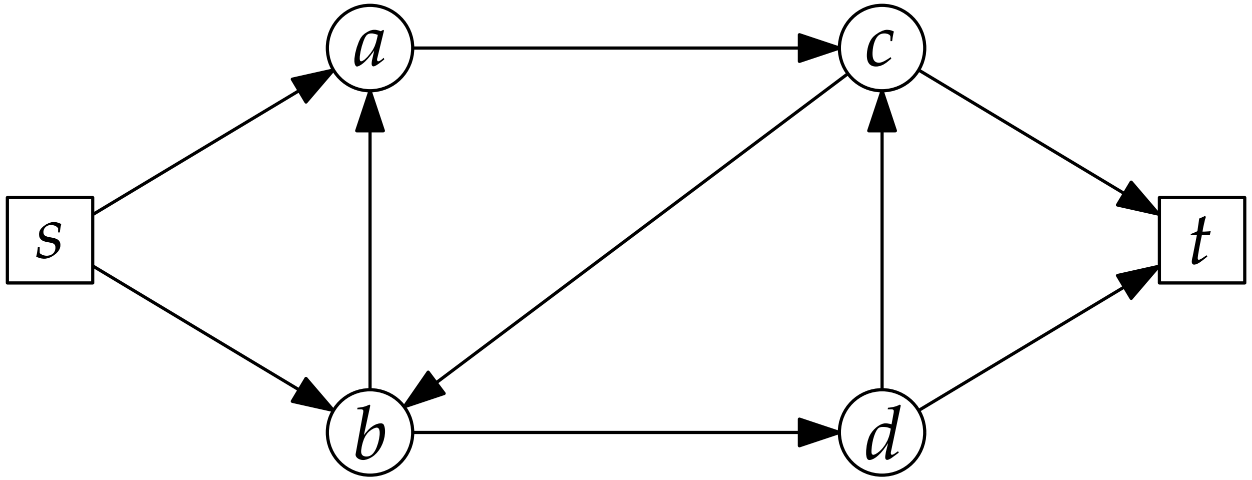
A **flow network** $G = (V, E)$ is a digraph (short for "**di**rected **graph**") with



Flow Networks

A **flow network** $G = (V, E)$ is a digraph (short for "**di**rected **graph**") with

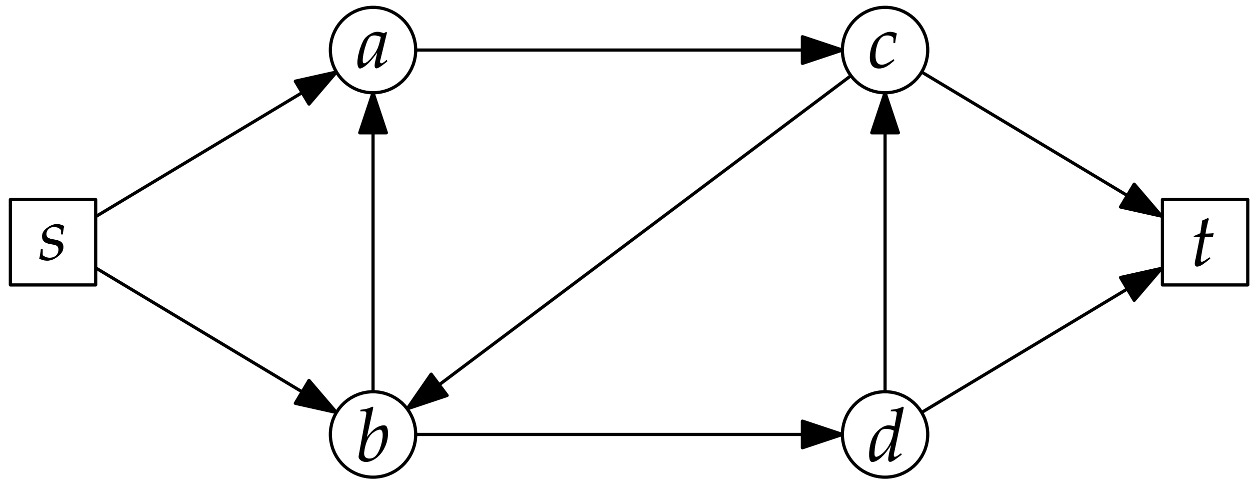
- unique **source** s and **sink** t,



Flow Networks

A **flow network** $G = (V, E)$ is a digraph (short for "**di**rected **graph**") with

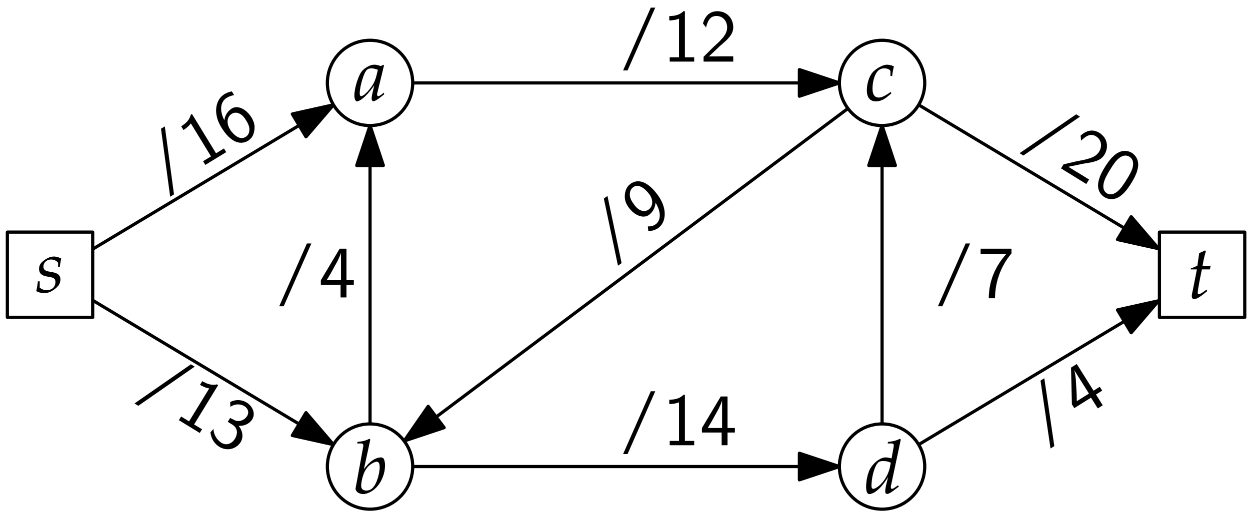
- unique **source** s and **sink** t,
- no antiparallel edges, and



Flow Networks

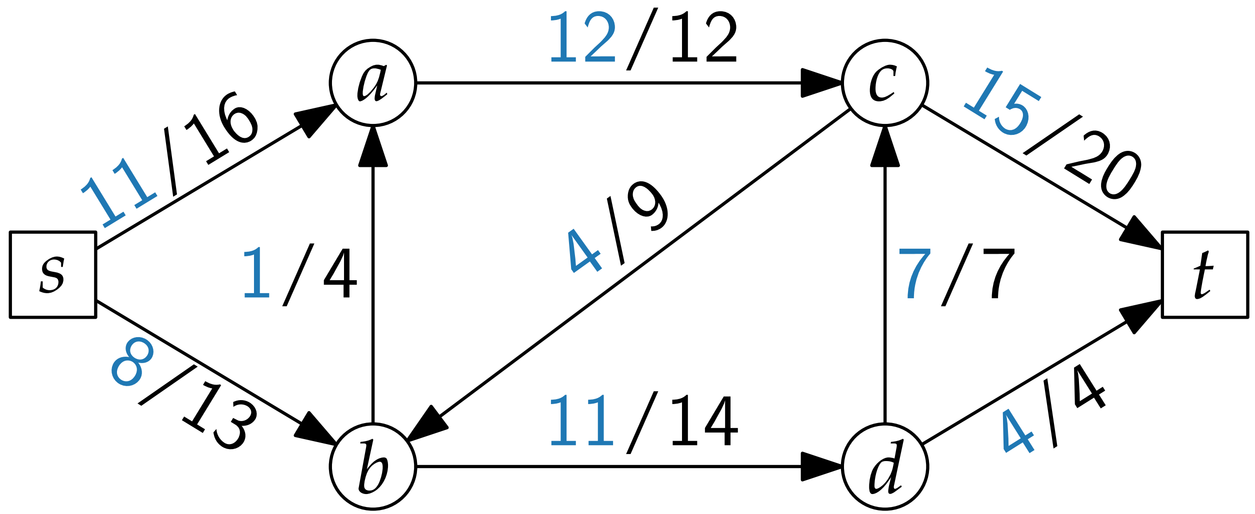
A **flow network** $G = (V, E)$ is a digraph (short for "**di**rected **graph**") with

- unique **source** s and **sink** t,
- no antiparallel edges, and
- a **capacity** $c(u, v) \geq 0$ for every $(u, v) \in E$.



Flow

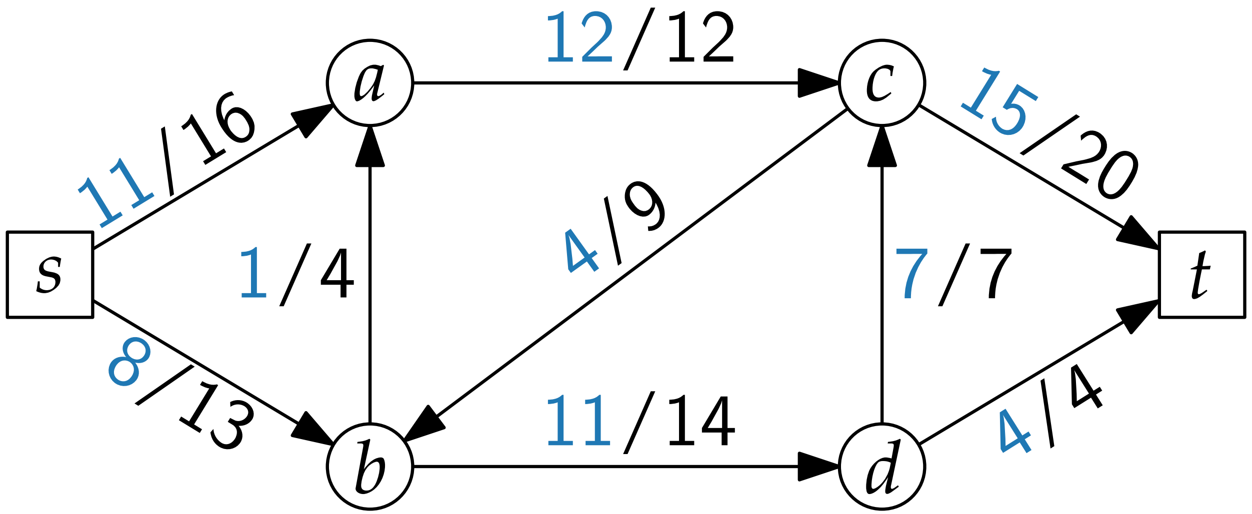
An s–t **flow** in G is a real-valued function $f\colon V \times V \to \mathbb{R}$ that satisfies



Flow

An s–t **flow** in G is a real-valued function $f \colon V \times V \to \mathbb{R}$ that satisfies

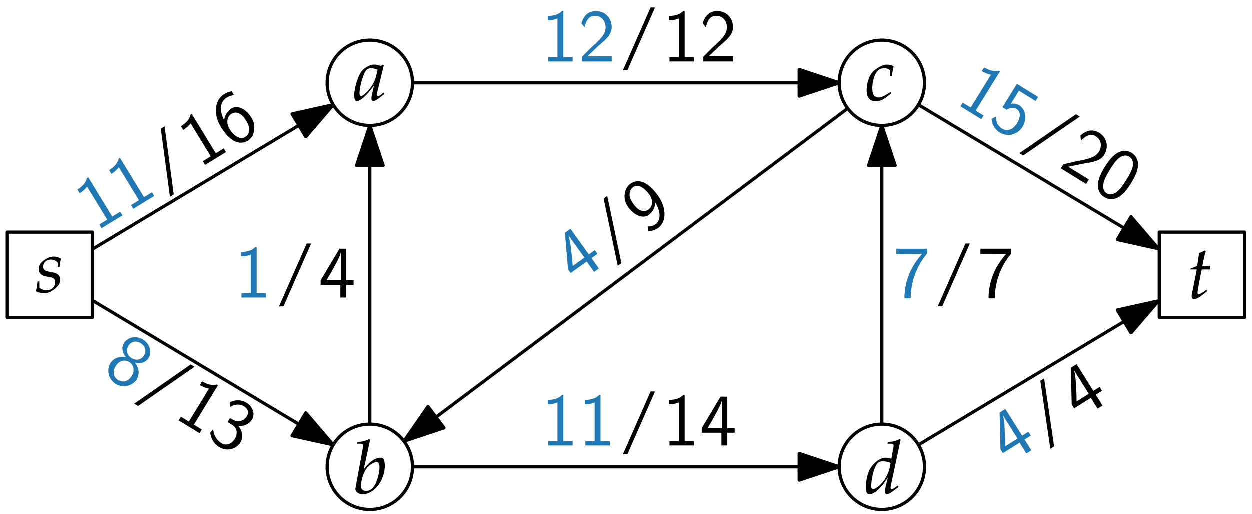
- **flow conservation**,
  $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$ for all $u \in V \setminus \{s,t\}$, and



Flow

An s–t **flow** in G is a real-valued function $f\colon V \times V \to \mathbb{R}$ that satisfies

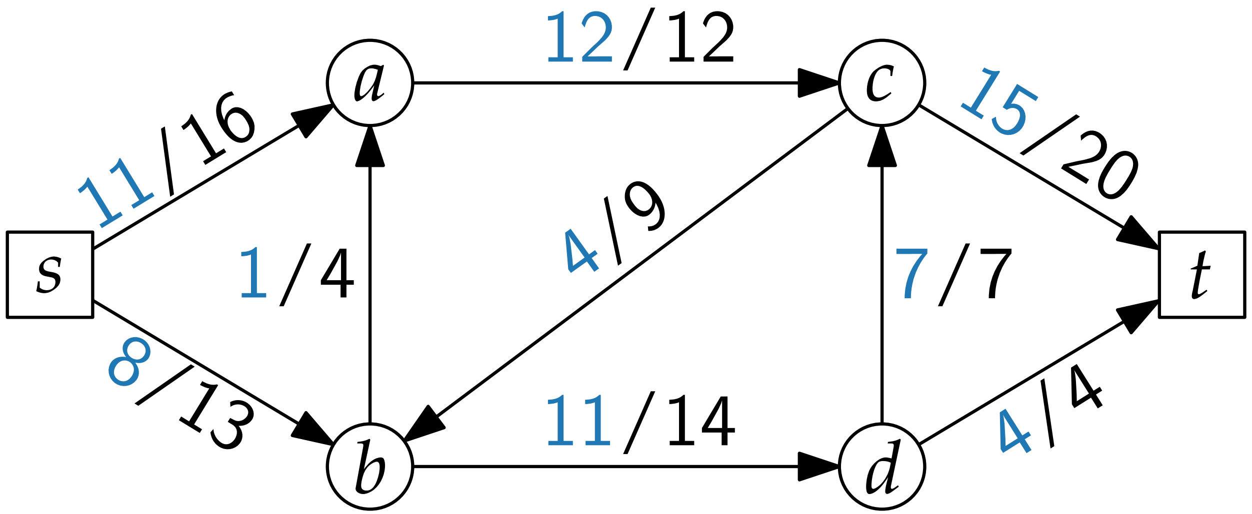
- **flow conservation**,
  $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$ for all $u \in V \setminus \{s,t\}$, and
- **capacity constraint**, $0 \leq f(u,v) \leq c(u,v)$.



Flow

An s–t **flow** in G is a real-valued function $f\colon V \times V \to \mathbb{R}$ that satisfies

- **flow conservation**,
  $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$ for all $u \in V \setminus \{s,t\}$, and
- **capacity constraint**, $0 \leq f(u,v) \leq c(u,v)$.



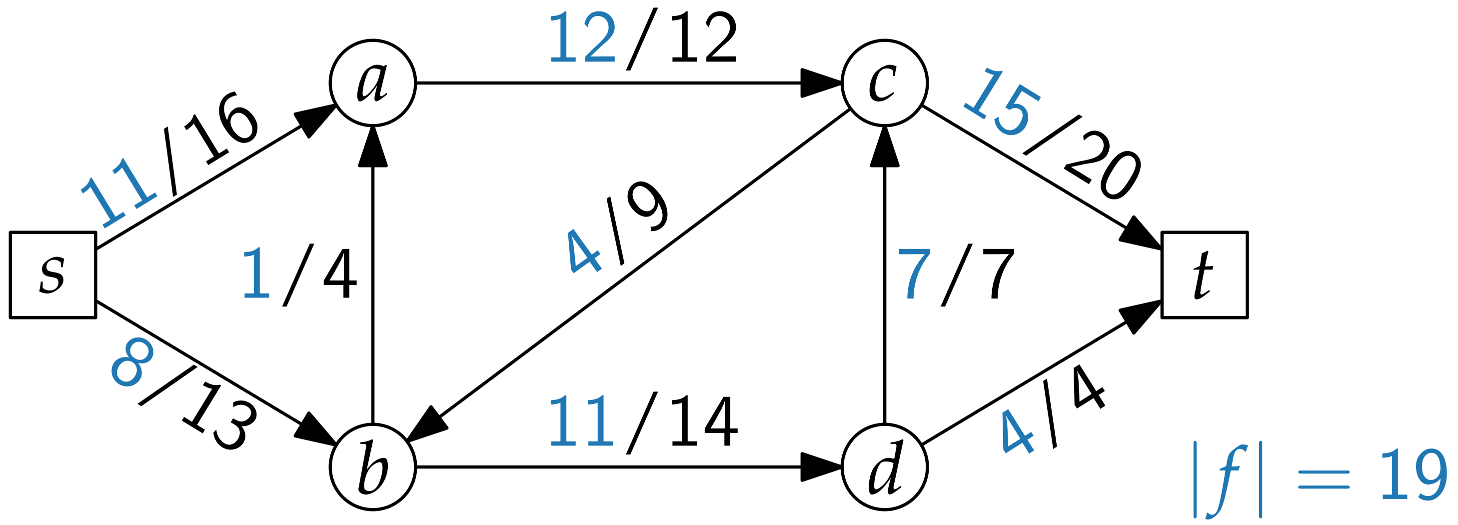
The **value** $|f|$ of an s–t flow f is defined as

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

Flow

An s–t **flow** in G is a real-valued function $f\colon V \times V \to \mathbb{R}$ that satisfies

- **flow conservation**,
  $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$ for all $u \in V \setminus \{s,t\}$, and
- **capacity constraint**, $0 \le f(u,v) \le c(u,v)$.



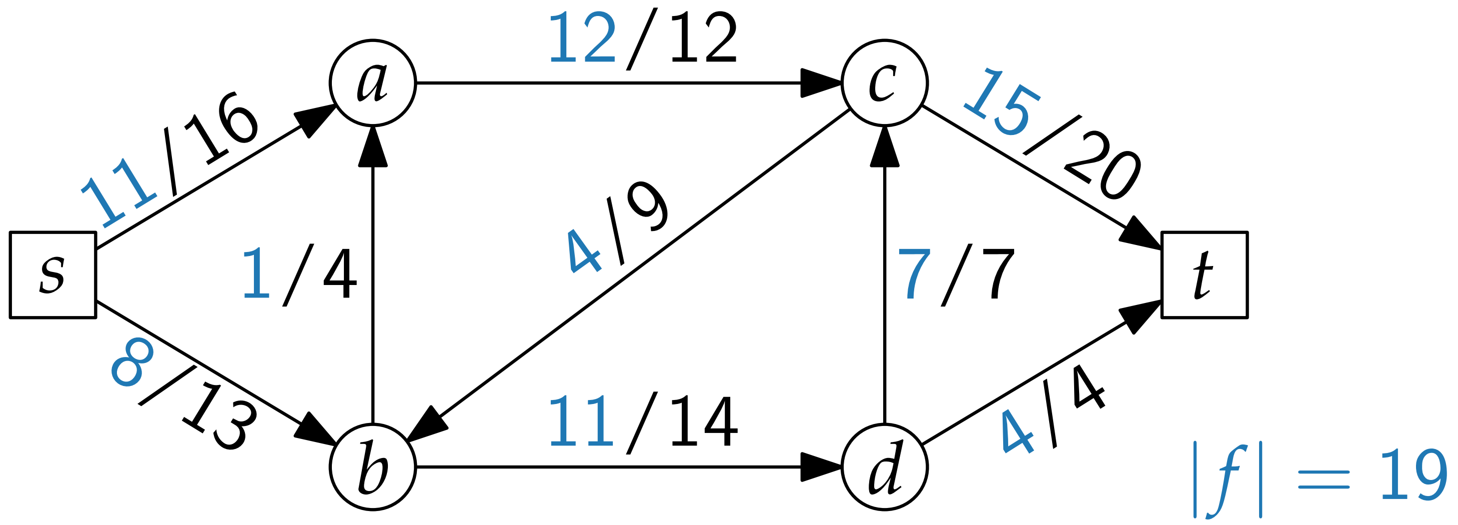
The **value** $|f|$ of an s–t flow f is defined as

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

Flow

An s–t **flow** in G is a real-valued function $f\colon V \times V \to \mathbb{R}$ that satisfies

- **flow conservation**,
  $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$ for all $u \in V \setminus \{s,t\}$, and
- **capacity constraint**, $0 \le f(u,v) \le c(u,v)$.



Maximum flow problem.
Given a flow network G with source s and sink t, find an s–t flow of maximum value.

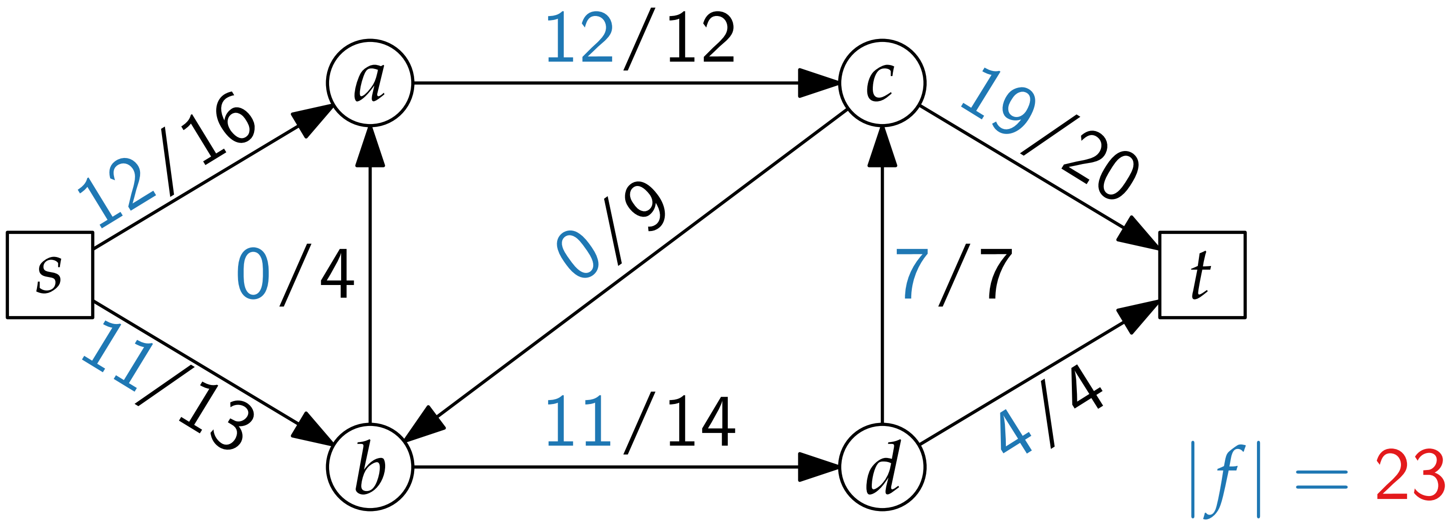
The **value** $|f|$ of an s–t flow f is defined as

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

Flow

An s–t **flow** in G is a real-valued function $f\colon V \times V \to \mathbb{R}$ that satisfies

- **flow conservation**,
  $\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$ for all $u \in V \setminus \{s,t\}$, and
- **capacity constraint**, $0 \le f(u,v) \le c(u,v)$.

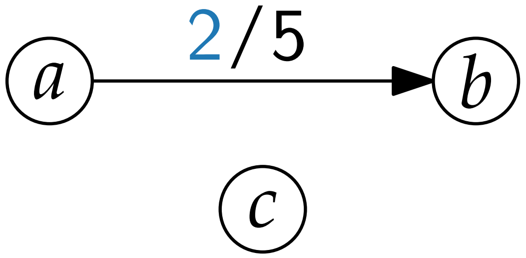


Maximum flow problem.
Given a flow network G with source s and sink t, find an s–t flow of maximum value.

The **value** $|f|$ of an s–t flow f is defined as

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

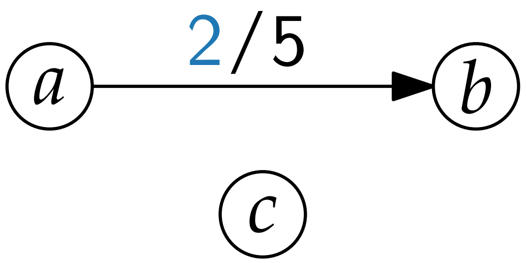
By How Much May Flow Change?



By How Much May Flow Change?

Given G and f, the **residual capacity** c_f for a pair $u, v \in V$ is

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

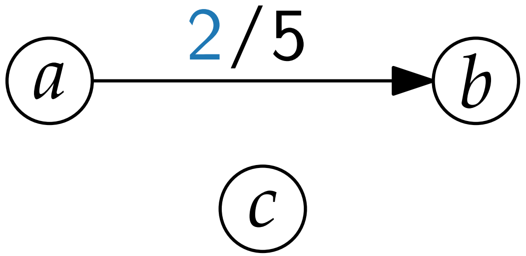


$$c_f(a,b) = 3$$
$$c_f(b,a) = 2$$
$$c_f(a,c) = 0$$

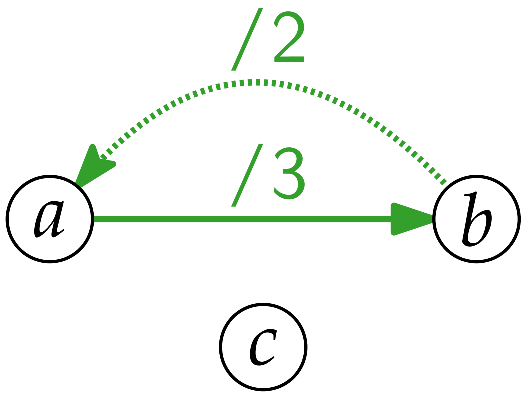
By How Much May Flow Change?

Given G and f, the **residual capacity** c_f for a pair $u, v \in V$ is

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



$$c_f(a,b) = 3$$
$$c_f(b,a) = 2$$
$$c_f(a,c) = 0$$



Residual Networks & Augmenting Paths

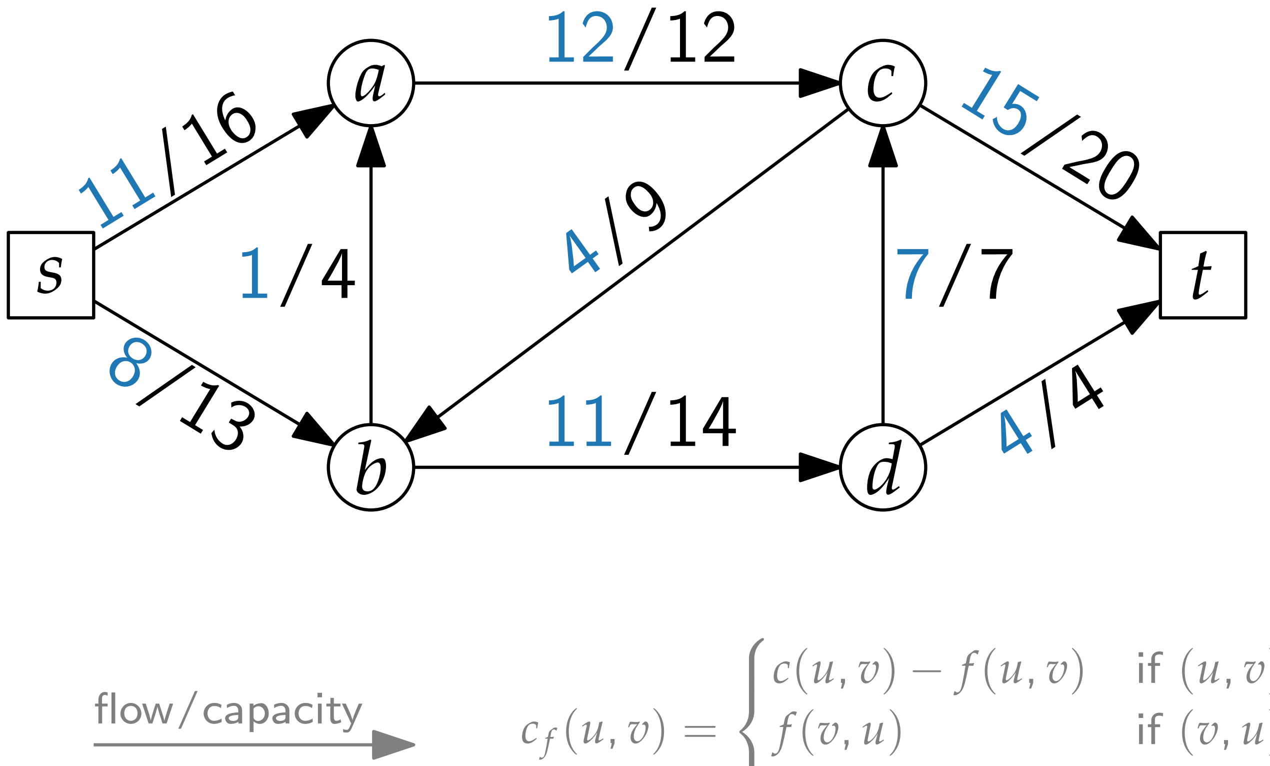
The **residual network** $G_f = (V, E_f)$ for a flow network G with s–t flow f has

- $E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$.

Residual Networks & Augmenting Paths

The **residual network** $G_f = (V, E_f)$ for a flow network G with s–t flow f has

- $E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$.

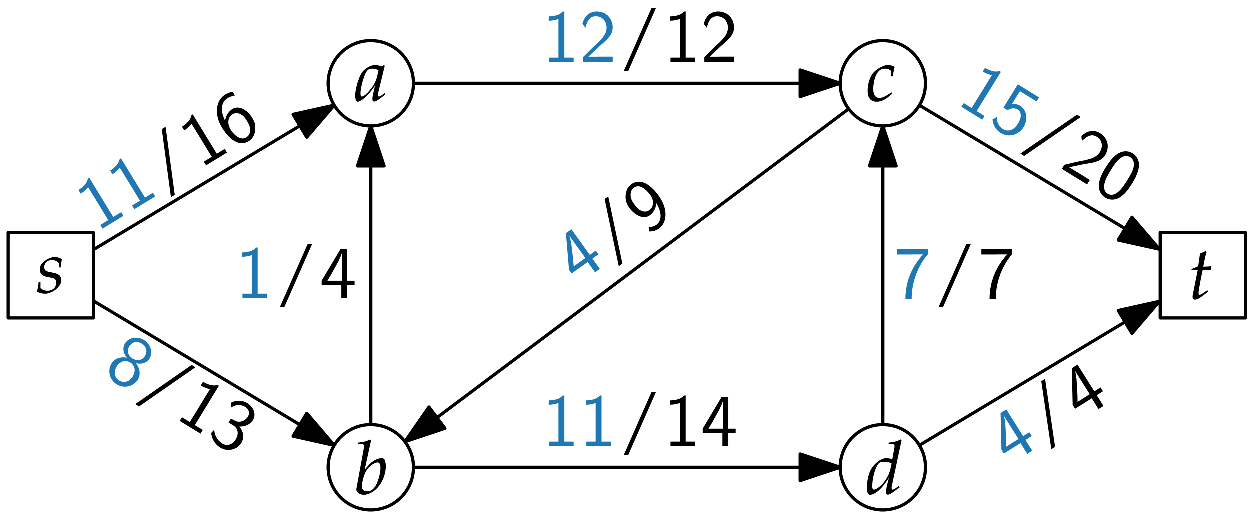


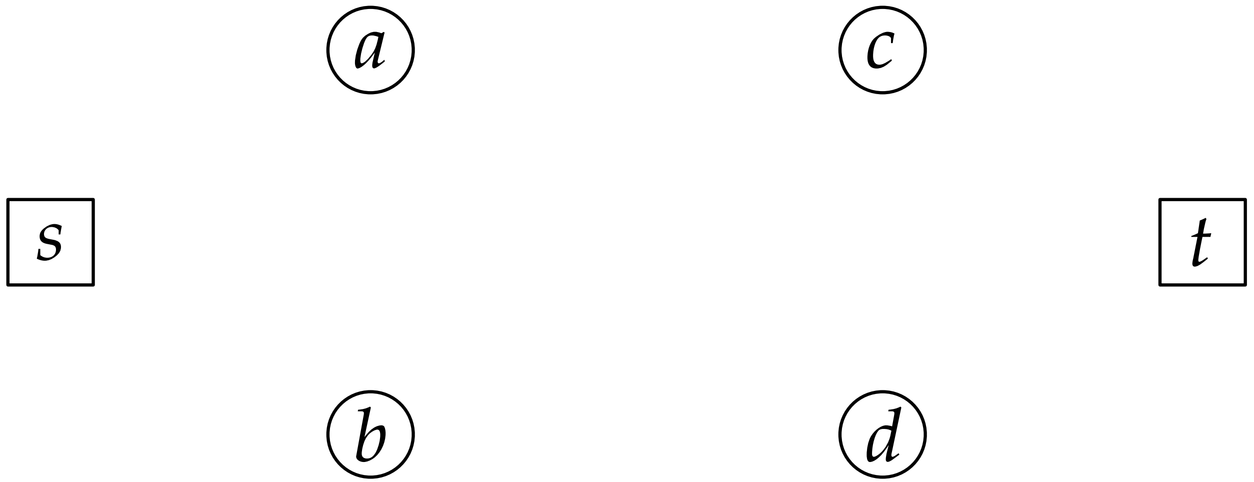
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Residual Networks & Augmenting Paths

The **residual network** $G_f = (V, E_f)$ for a flow network G with s–t flow f has

- $E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$.



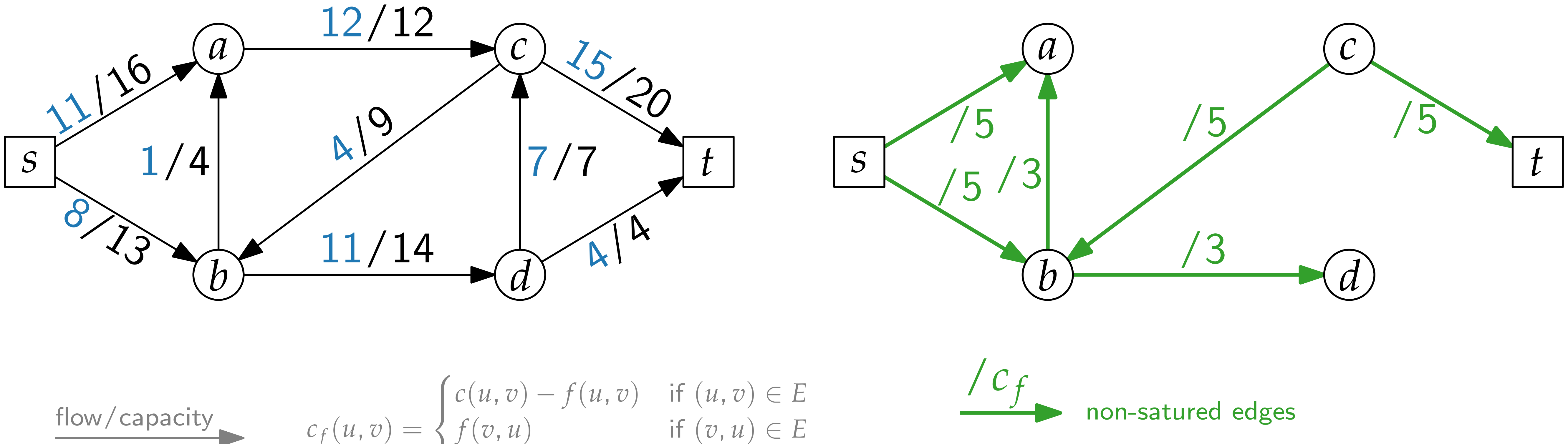


$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Residual Networks & Augmenting Paths

The **residual network** $G_f = (V, E_f)$ for a flow network G with s–t flow f has

- $E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$.

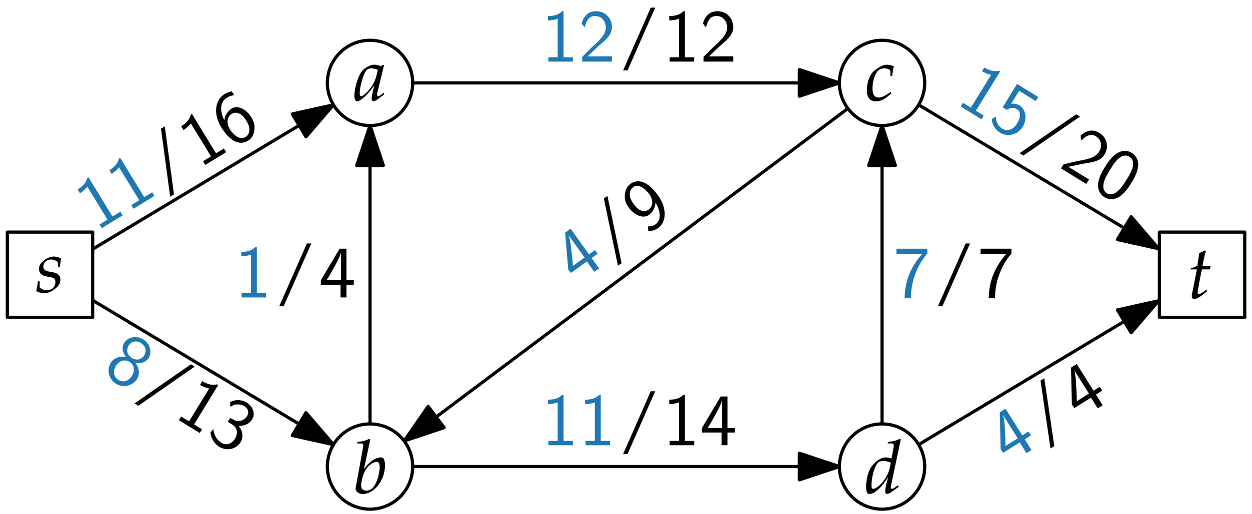


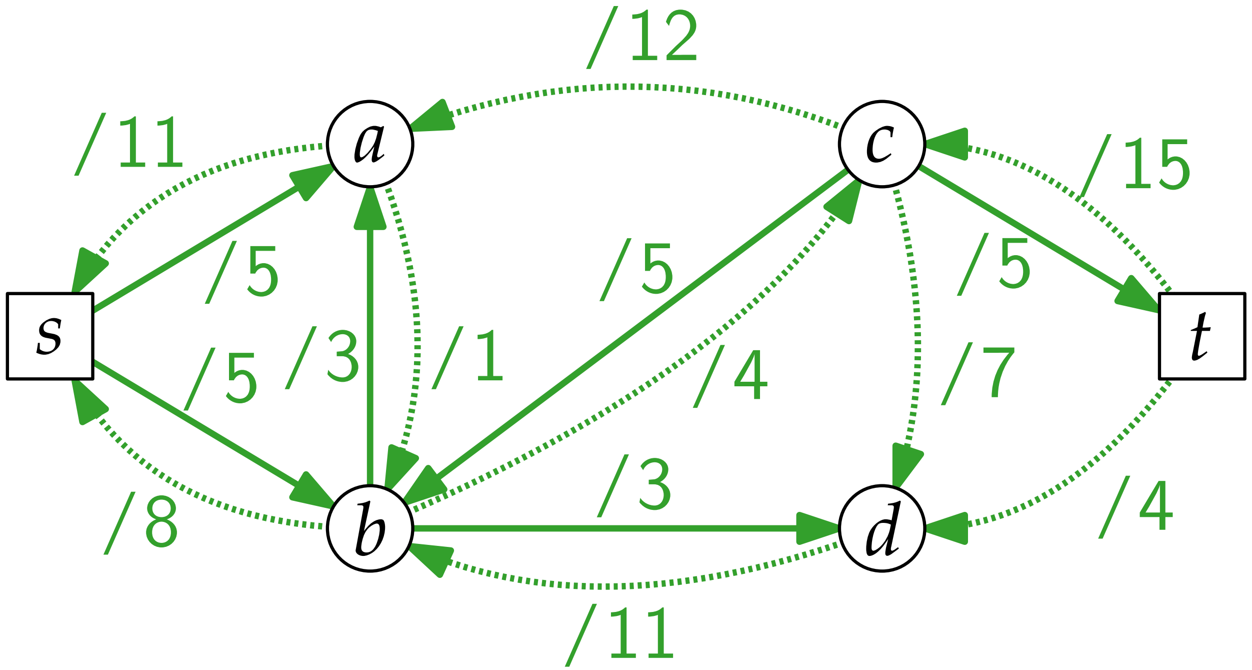
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Residual Networks & Augmenting Paths

The **residual network** $G_f = (V, E_f)$ for a flow network G with s–t flow f has

- $E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$.





flow/capacity

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

$/c_f$ non-satured edges

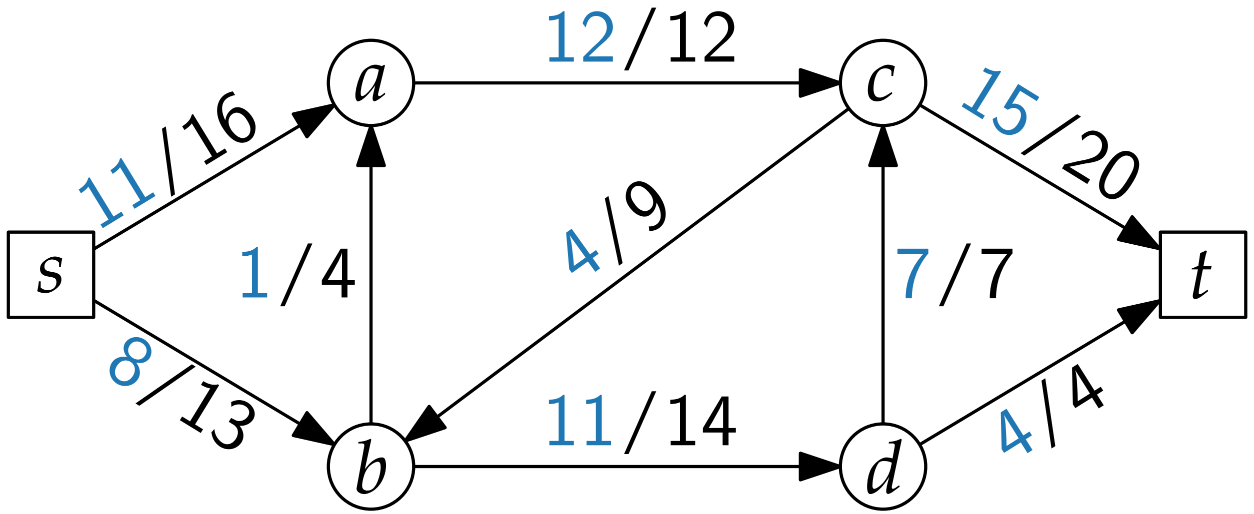
reverse edges

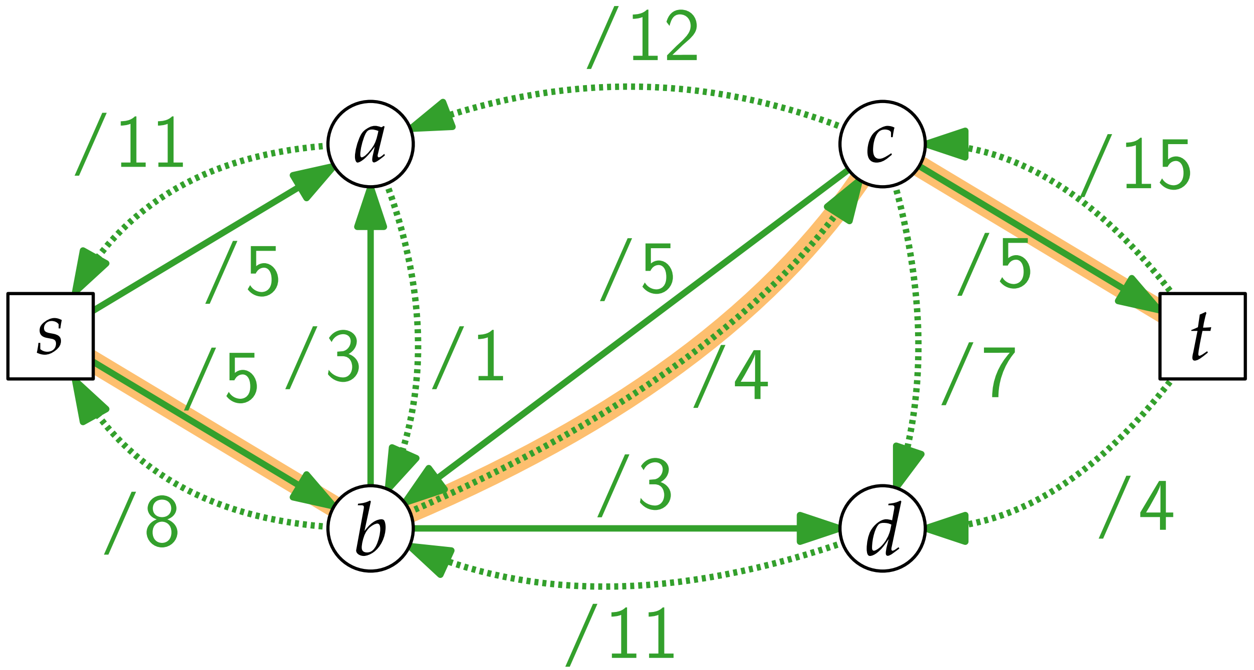
Residual Networks & Augmenting Paths

The **residual network** $G_f = (V, E_f)$ for a flow network G with s–t flow f has

- $E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$.

An **augmenting path** is an st-path in G_f.





flow/capacity

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

$/c_f$ non-satured edges

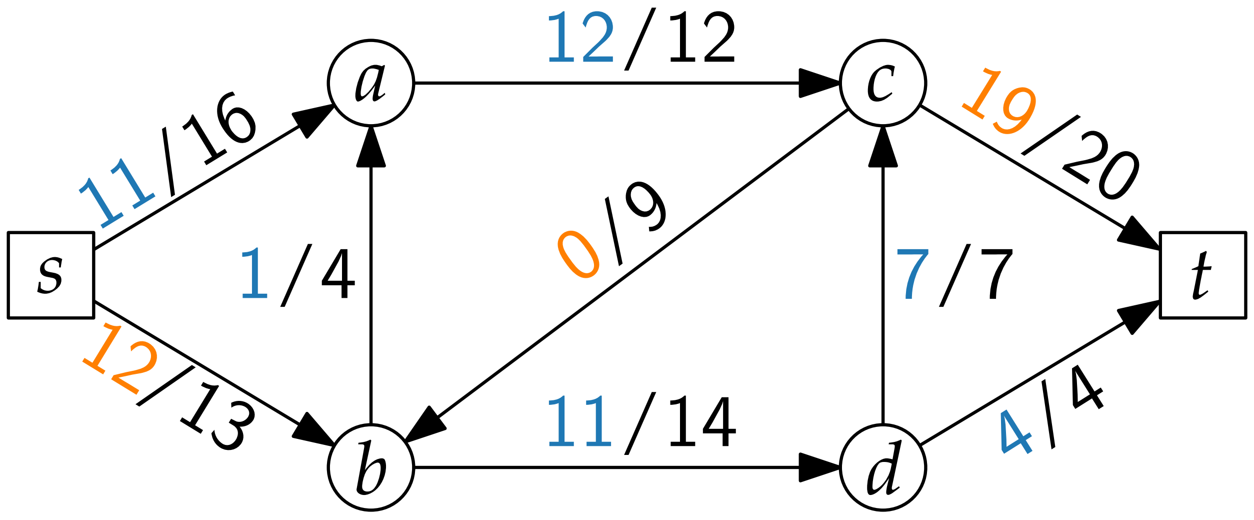
reverse edges

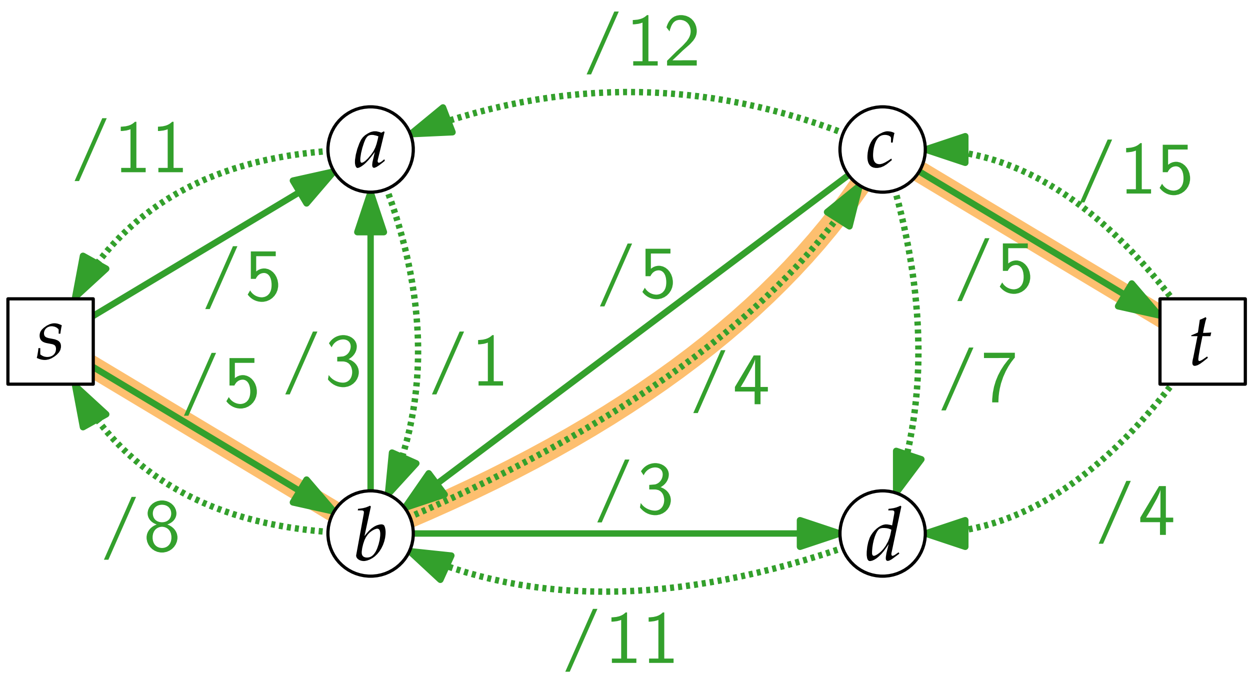
Residual Networks & Augmenting Paths

The **residual network** $G_f = (V, E_f)$ for a flow network G with s–t flow f has

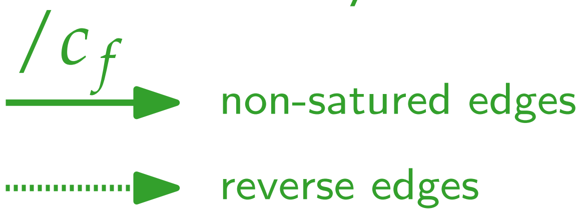
- $E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$.

An **augmenting path** is an st-path in G_f. $\Rightarrow$ use to increase f





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



The Algorithms of Ford–Fulkerson and Edmonds–Karp

FordFulkerson$(G = (V, E), c, s, t)$

```
foreach uv ∈ E do                              } initialising zero flow
    f_uv ← 0
while G_f contains augmenting path p do
    Δ ← min_{uv∈p} c_f(uv)                     } residual capacity of p
    foreach uv ∈ p do                          ⎫
        if uv ∈ E then                         ⎪
            f_uv ← f_uv + Δ                    ⎬ augmentation along p
        else                                   ⎪
            f_vu ← f_vu − Δ                    ⎭
return f                                       } return max flow
```

The Algorithms of Ford–Fulkerson and Edmonds–Karp

EdmondsKarp

~~FordFulkerson~~$(G = (V, E), c, s, t)$

foreach $uv \in E$ **do** } initialising zero flow

  $f_{uv} \leftarrow 0$

while G_f contains shortest augmenting path p **do**

  $\Delta \leftarrow \min_{uv \in p} c_f(uv)$ } residual capacity of p

  foreach $uv \in p$ **do** } augmentation along p

    if $uv \in E$ **then**

      $f_{uv} \leftarrow f_{uv} + \Delta$

    else

      $f_{vu} \leftarrow f_{vu} - \Delta$

return f } return max flow

The Algorithms of Ford–Fulkerson and Edmonds–Karp

EdmondsKarp
~~FordFulkerson~~$(G = (V, E), c, s, t)$

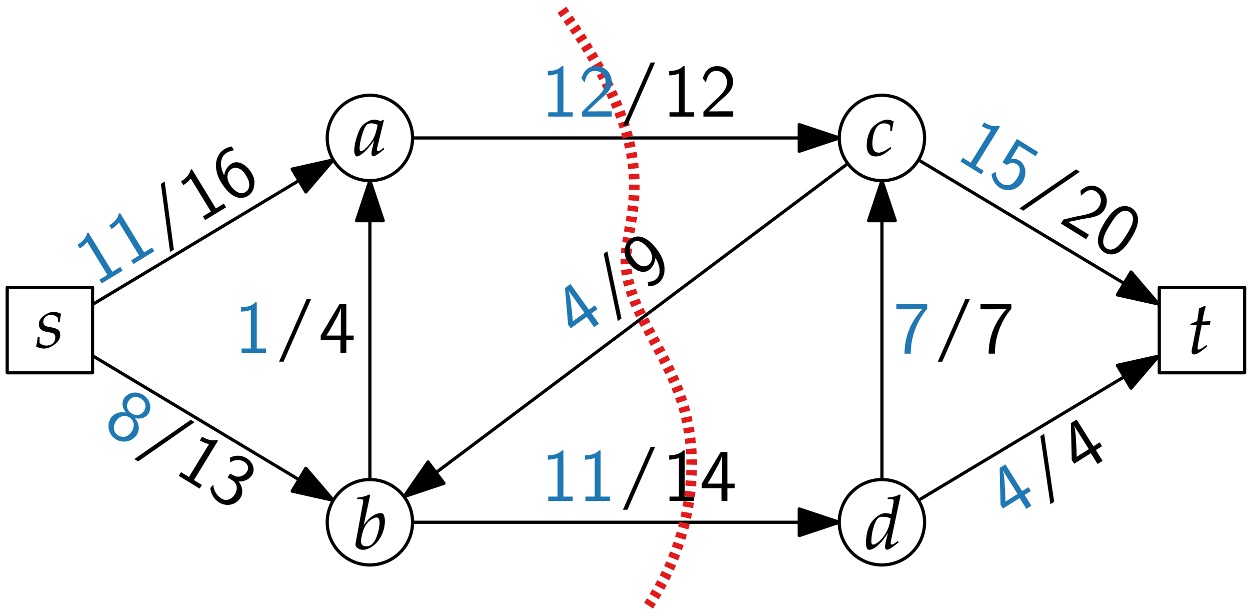
foreach $uv \in E$ **do** } initialising zero flow
  $f_{uv} \leftarrow 0$

                shortest
while G_f contains augmenting path p **do**
  $\Delta \leftarrow \min_{uv \in p} c_f(uv)$ } residual capacity of p
  foreach $uv \in p$ **do** } augmentation along p
    if $uv \in E$ **then**
      $f_{uv} \leftarrow f_{uv} + \Delta$
    else
      $f_{vu} \leftarrow f_{vu} - \Delta$

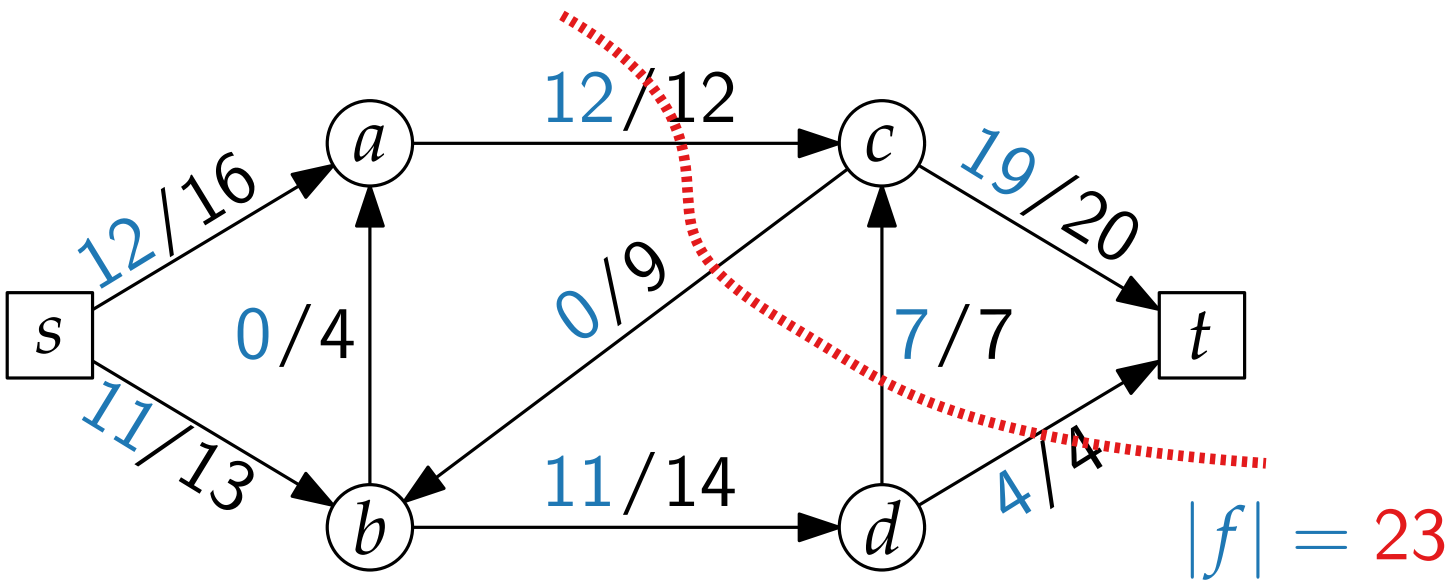
return f } return max flow

- Ford–Fulkerson runs in $\mathcal{O}(|E| \cdot |f^\star|)$ and Edmonds–Karp in $\mathcal{O}(|V| \cdot |E|^2)$ time.

The Max-Flow Min-Cut Theorem

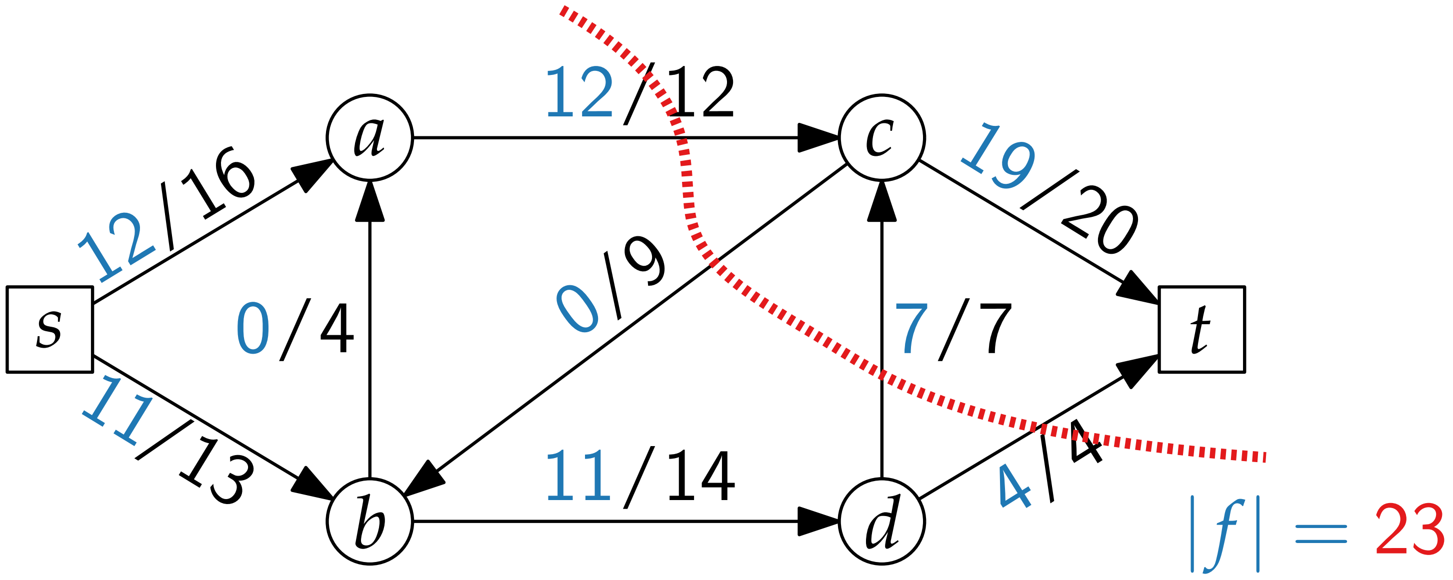


The Max-Flow Min-Cut Theorem



The Max-Flow Min-Cut Theorem

Theorem.
For an s–t flow f in a flow network G, the following conditions are equivalent:
- f is a maximum s–t flow in G.
- G_f contains no augmenting paths.
- $|f| = c(S, T)$, which is the capacity of some s–t cut (S, T) of G.



The Push–Relabel Idea

A New Approach to the Maximum-Flow Problem

ANDREW V. GOLDBERG

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

ROBERT E. TARJAN

Princeton University, Princeton, New Jersey, and AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. All previously known efficient maximum-flow algorithms work by finding augmenting paths, either one path at a time (as in the original Ford and Fulkerson algorithm) or all shortest-length augmenting paths at once (using the layered network approach of Dinic). An alternative method based on the *preflow* concept of Karzanov is introduced. A preflow is like a flow, except that the total amount

for the next phase. Our algorithm abandons the idea of finding a flow in each phase and also abandons the idea of global phases. Instead, our algorithm maintains a preflow in the original network and pushes local flow excess toward the sink along what it estimates to be shortest paths in the residual graph. This pushing of flow changes the residual graph, and paths to the sink may become saturated. Excess that cannot be moved to the sink is returned to the source, also along estimated shortest paths. Only when the algorithm terminates does the preflow become a flow, and then it is a maximum flow.

The Push–Relabel Idea

A New Approach to the Maximum-Flow Problem

ANDREW V. GOLDBERG

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

ROBERT E. TARJAN

Princeton University, Princeton, New Jersey, and AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. All previously known efficient maximum-flow algorithms work by finding augmenting paths, either one path at a time (as in the original Ford and Fulkerson algorithm) or all shortest-length augmenting paths at once (using the layered network approach of Dinic). An alternative method based on the *preflow* concept of Karzanov is introduced. A preflow is like a flow, except that the total amount

for the next phase. Our algorithm abandons the idea of finding a flow in each phase and also abandons the idea of global phases. Instead, our algorithm maintains a preflow in the original network and pushes local flow excess toward the sink along what it estimates to be shortest paths in the residual graph. This pushing of flow changes the residual graph, and paths to the sink may become saturated. Excess that cannot be moved to the sink is returned to the source, also along estimated shortest paths. Only when the algorithm terminates does the preflow become a flow, and then it is a maximum flow.

The Push–Relabel Idea

A New Approach to the Maximum-Flow Problem

ANDREW V. GOLDBERG

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

ROBERT E. TARJAN

Princeton University, Princeton, New Jersey, and AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. All previously known efficient maximum-flow algorithms work by finding augmenting paths, either one path at a time (as in the original Ford and Fulkerson algorithm) or all shortest-length augmenting paths at once (using the layered network approach of Dinic). An alternative method based on the *preflow* concept of Karzanov is introduced. A preflow is like a flow, except that the total amount

for the next phase. Our algorithm abandons the idea of finding a flow in each phase and also abandons the idea of global phases. Instead, our algorithm maintains a preflow in the original network and pushes local flow excess toward the sink along what it estimates to be shortest paths in the residual graph. This pushing of flow changes the residual graph, and paths to the sink may become saturated. Excess that cannot be moved to the sink is returned to the source, also along estimated shortest paths. Only when the algorithm terminates does the preflow become a flow, and then it is a maximum flow.

The Push–Relabel Idea

A New Approach to the Maximum-Flow Problem

ANDREW V. GOLDBERG

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

ROBERT E. TARJAN

Princeton University, Princeton, New Jersey, and AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. All previously known efficient maximum-flow algorithms work by finding augmenting paths, either one path at a time (as in the original Ford and Fulkerson algorithm) or all shortest-length augmenting paths at once (using the layered network approach of Dinic). An alternative method based on the *preflow* concept of Karzanov is introduced. A preflow is like a flow, except that the total amount

for the next phase. Our algorithm abandons the idea of finding a flow in each phase and also abandons the idea of global phases. Instead, our algorithm maintains a preflow in the original network and pushes local flow excess toward the sink along what it estimates to be shortest paths in the residual graph. This pushing of flow changes the residual graph, and paths to the sink may become saturated. Excess that cannot be moved to the sink is returned to the source, also along estimated shortest paths. Only when the algorithm terminates does the preflow become a flow, and then it is a maximum flow.

The Push–Relabel Idea

A New Approach to the Maximum-Flow Problem

ANDREW V. GOLDBERG

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

ROBERT E. TARJAN

Princeton University, Princeton, New Jersey, and AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. All previously known efficient maximum-flow algorithms work by finding augmenting paths, either one path at a time (as in the original Ford and Fulkerson algorithm) or all shortest-length augmenting paths at once (using the layered network approach of Dinic). An alternative method based on the *preflow* concept of Karzanov is introduced. A preflow is like a flow, except that the total amount

for the next phase. Our algorithm abandons the idea of finding a flow in each phase and also abandons the idea of global phases. Instead, our algorithm maintains a preflow in the original network and pushes local flow excess toward the sink along what it estimates to be shortest paths in the residual graph. This pushing of flow changes the residual graph, and paths to the sink may become saturated. Excess that cannot be moved to the sink is returned to the source, also along estimated shortest paths. Only when the algorithm terminates does the preflow become a flow, and then it is a maximum flow.

The Push–Relabel Idea

A New Approach to the Maximum-Flow Problem

ANDREW V. GOLDBERG

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

ROBERT E. TARJAN

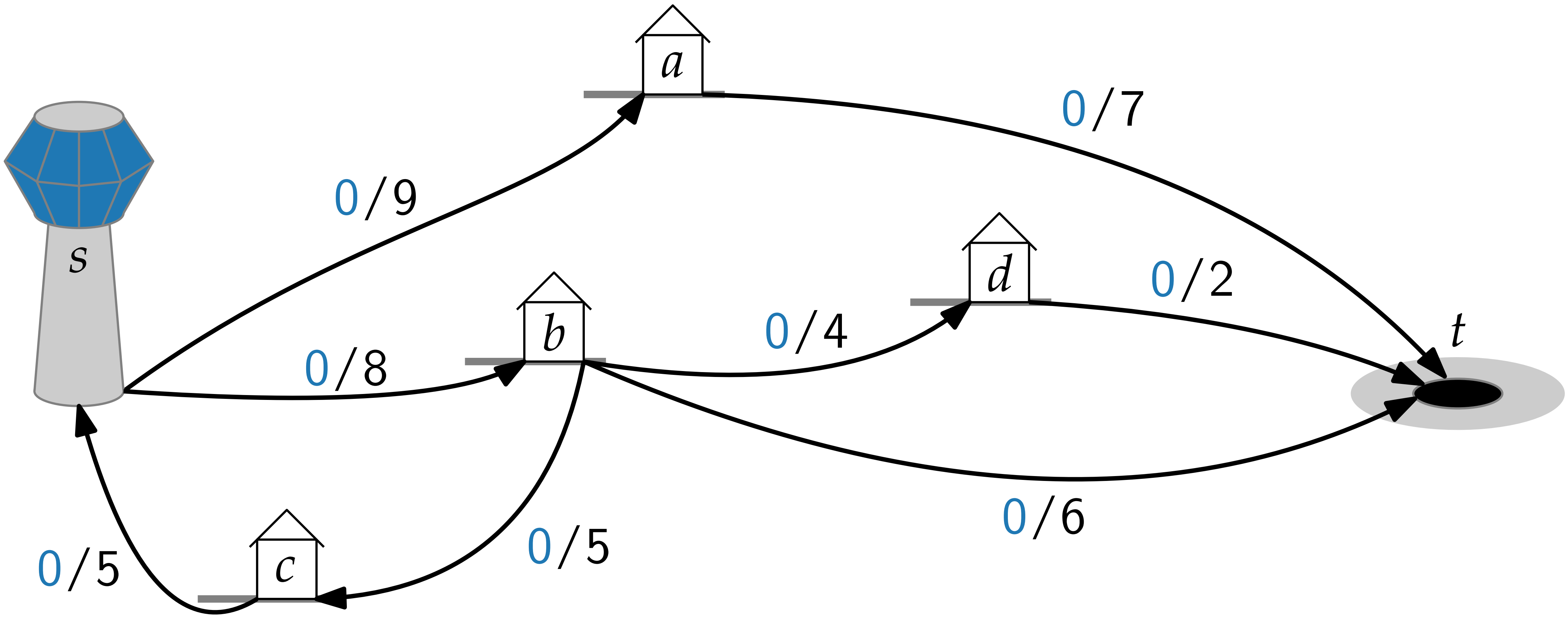
Princeton University, Princeton, New Jersey, and AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. All previously known efficient maximum-flow algorithms work by finding augmenting paths, either one path at a time (as in the original Ford and Fulkerson algorithm) or all shortest-length augmenting paths at once (using the layered network approach of Dinic). An alternative method based on the *preflow* concept of Karzanov is introduced. A preflow is like a flow, except that the total amount

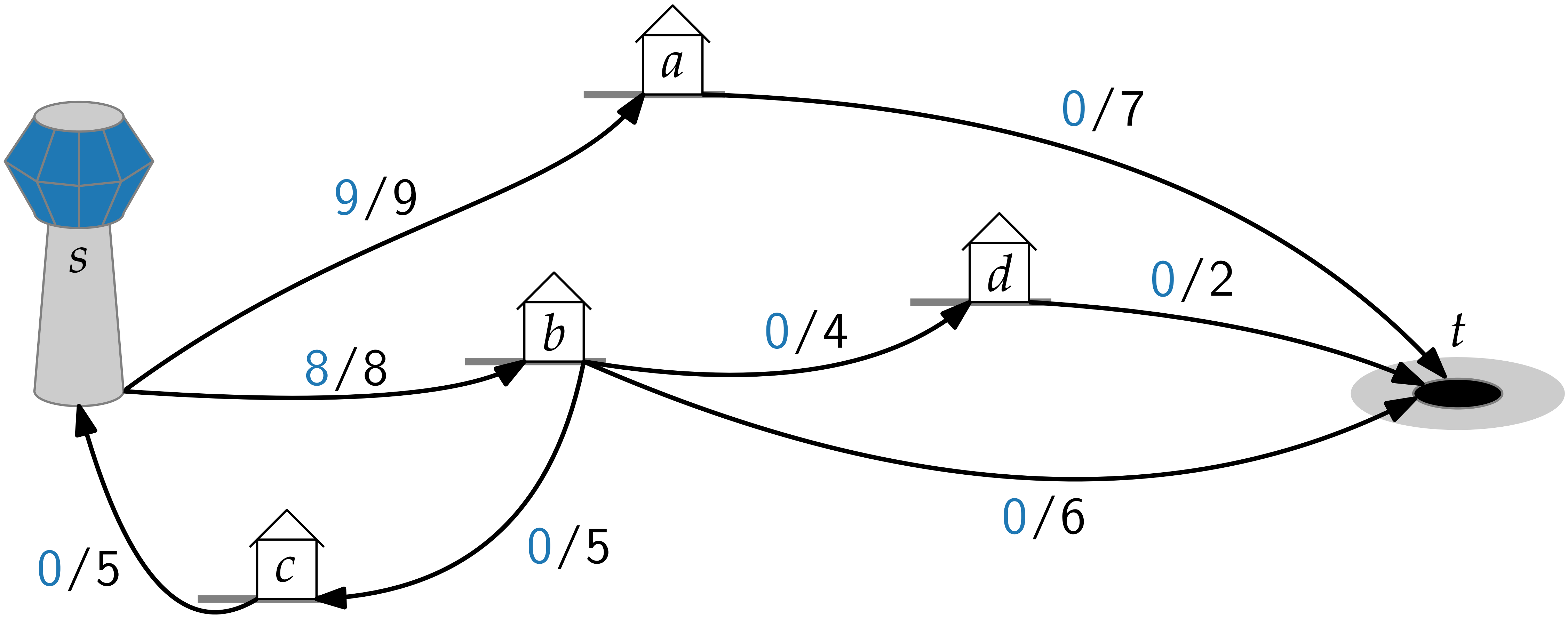
for the next phase. Our algorithm abandons the idea of finding a flow in each phase and also abandons the idea of global phases. Instead, our algorithm maintains a preflow in the original network and pushes local flow excess toward the sink along what it estimates to be shortest paths in the residual graph. This pushing of flow changes the residual graph, and paths to the sink may become saturated. Excess that cannot be moved to the sink is returned to the source, also along estimated shortest paths. Only when the algorithm terminates does the preflow become a flow, and then it is a maximum flow.

The Push–Relabel Idea

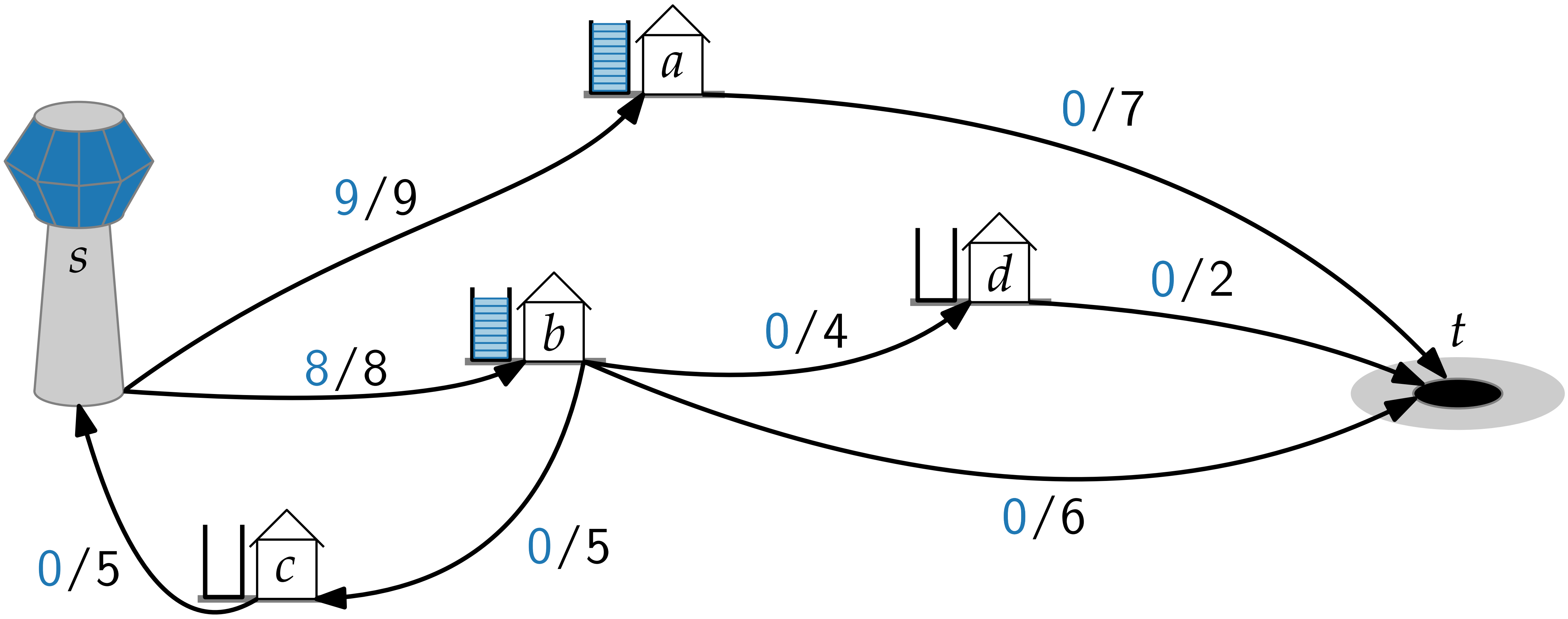
The Push–Relabel Idea



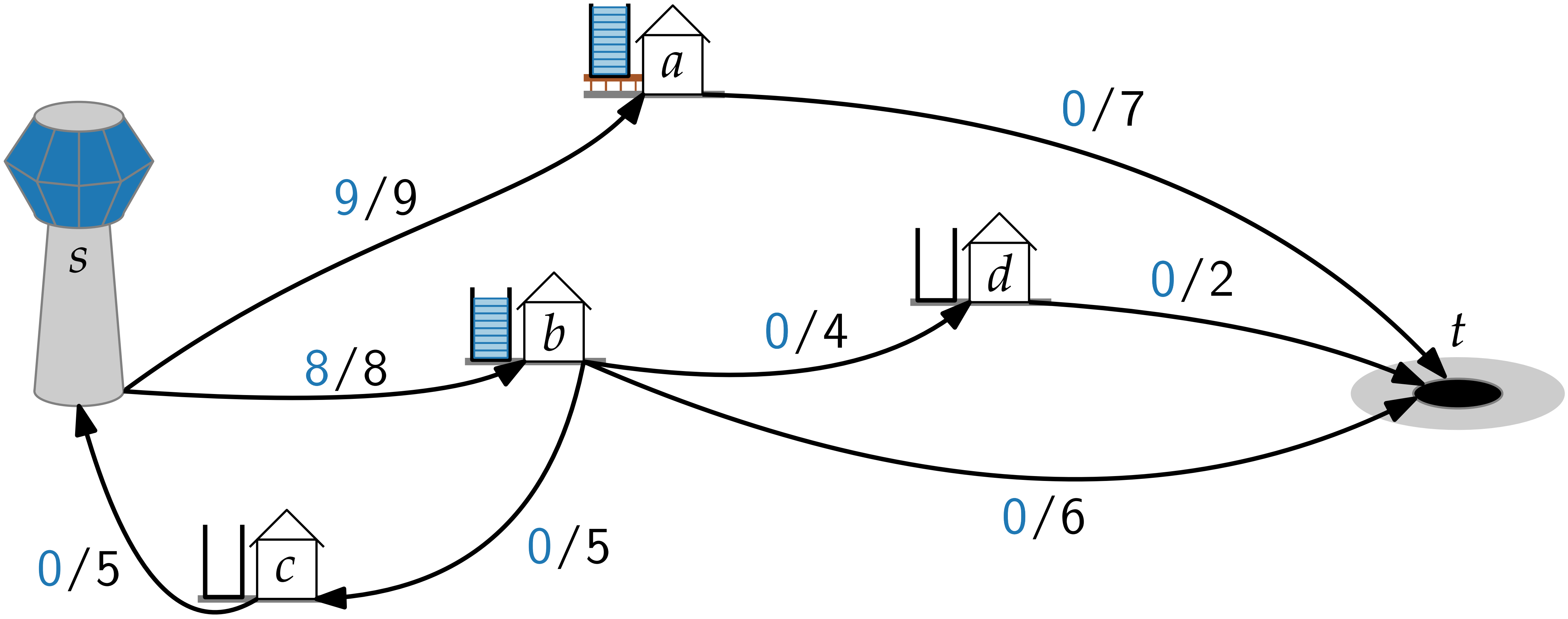
The Push–Relabel Idea



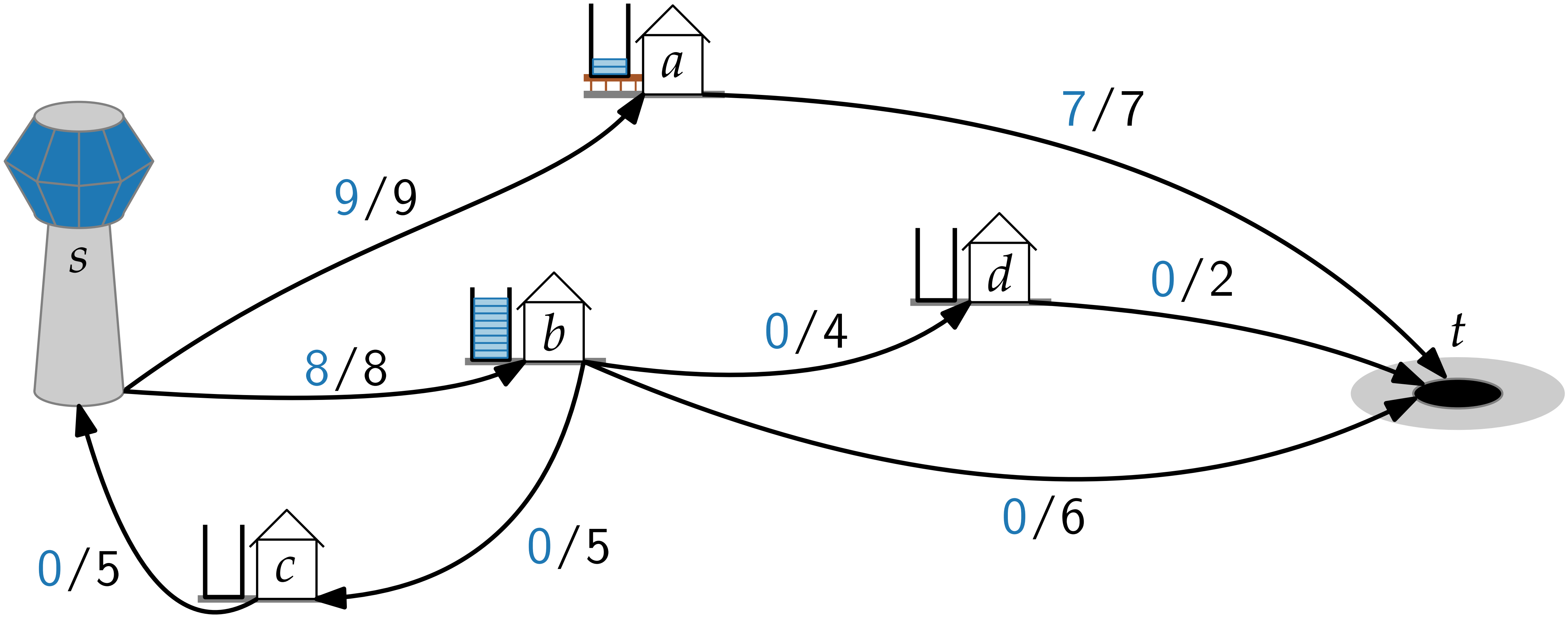
The Push–Relabel Idea



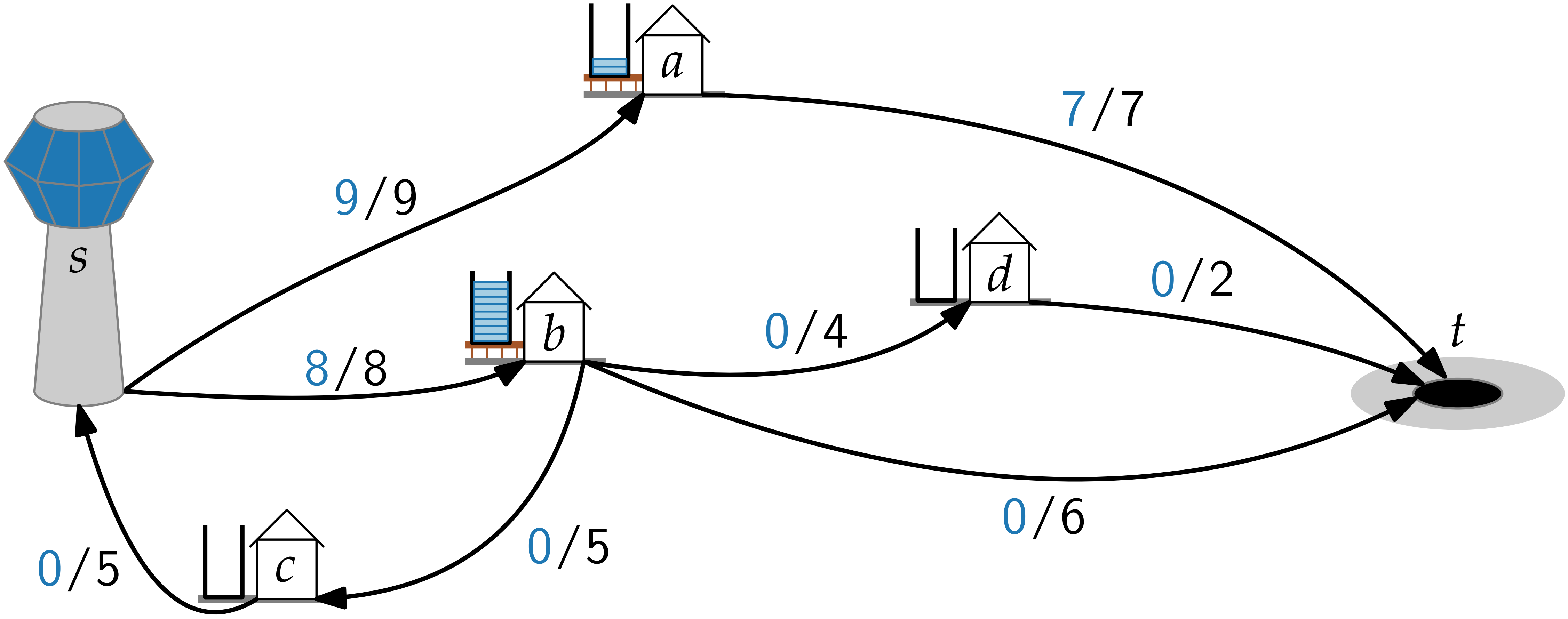
The Push–Relabel Idea



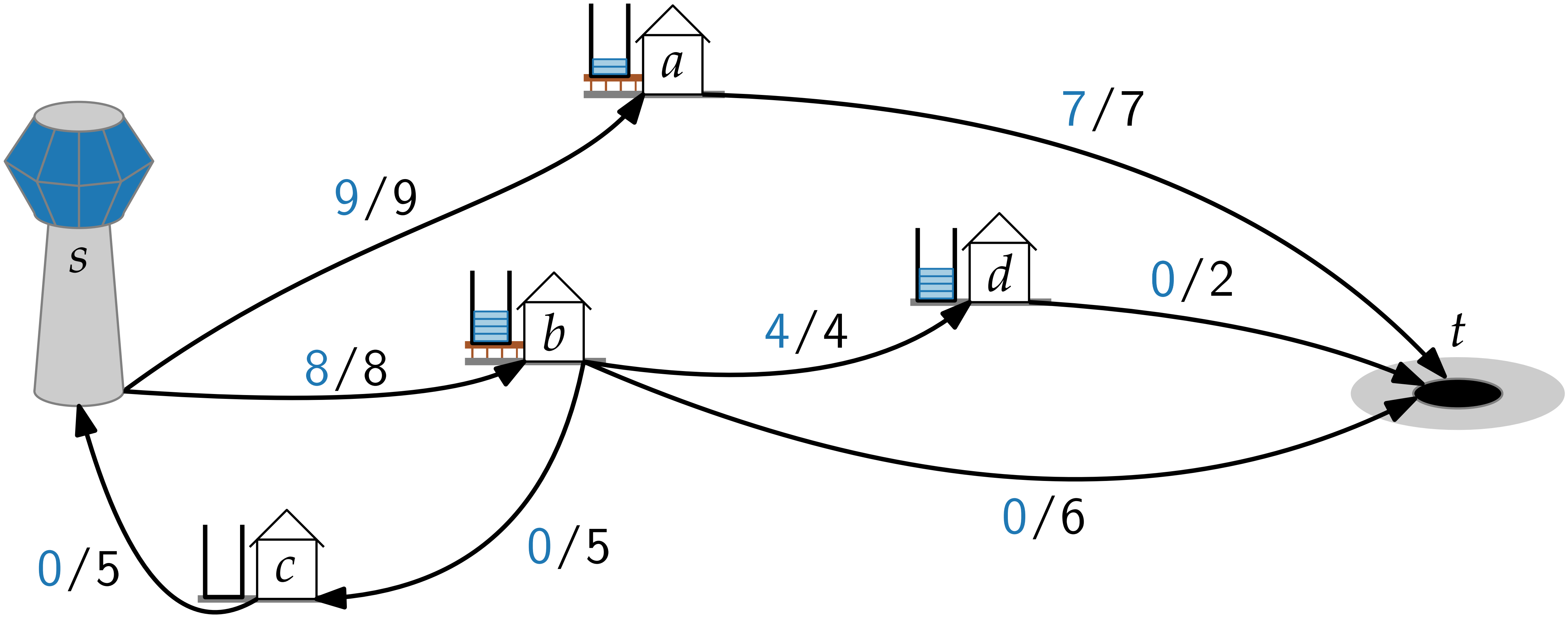
The Push–Relabel Idea



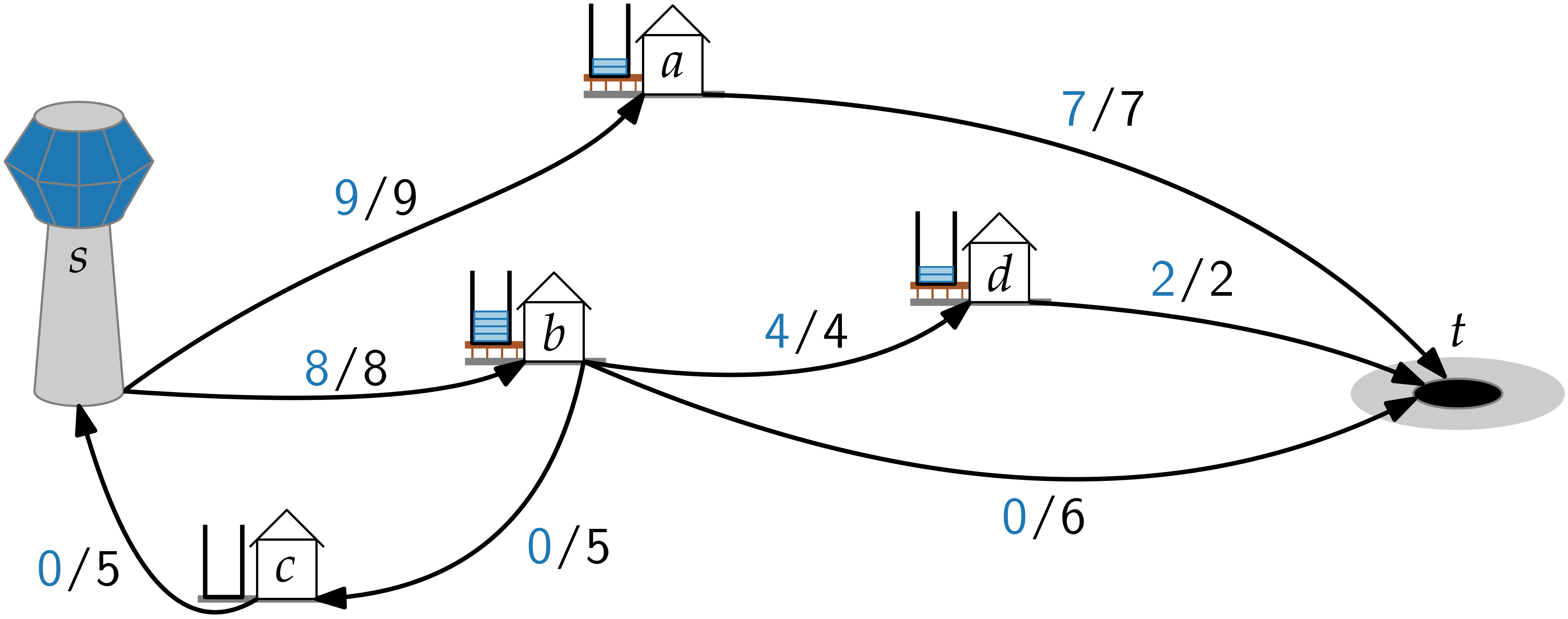
The Push–Relabel Idea



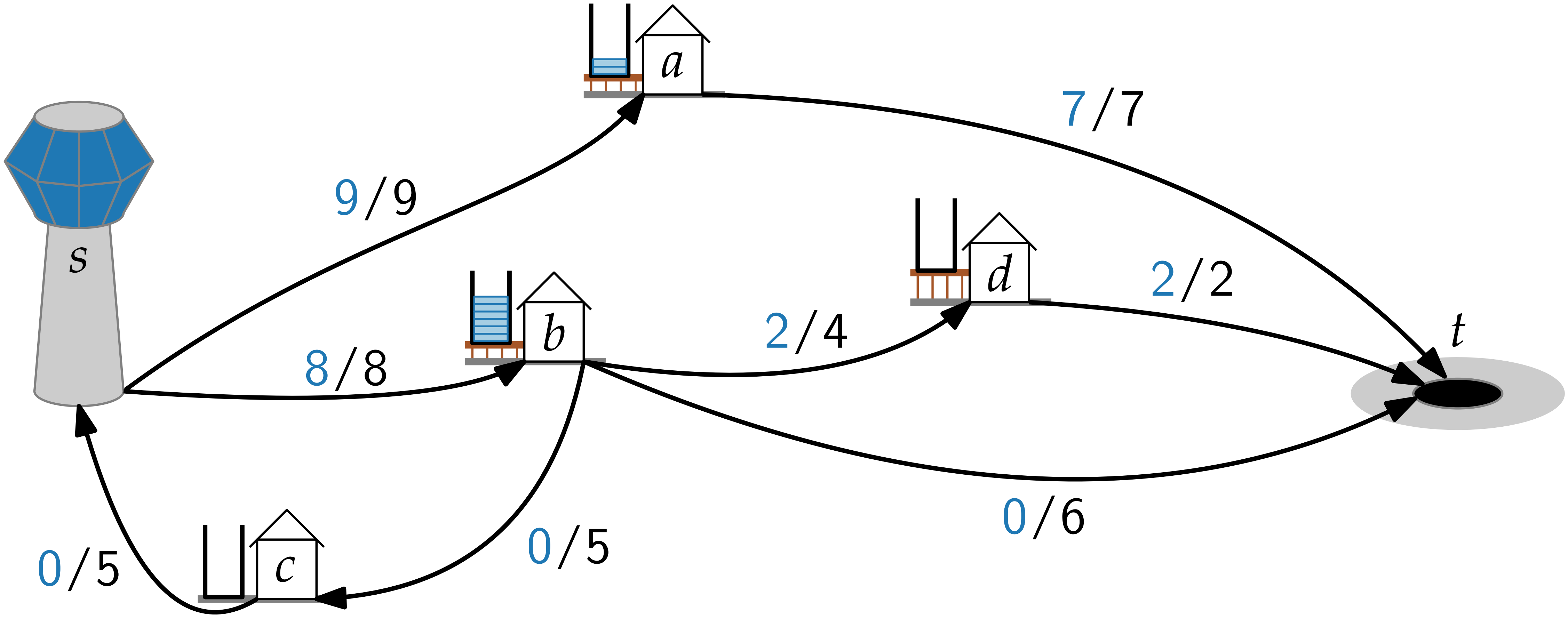
The Push–Relabel Idea



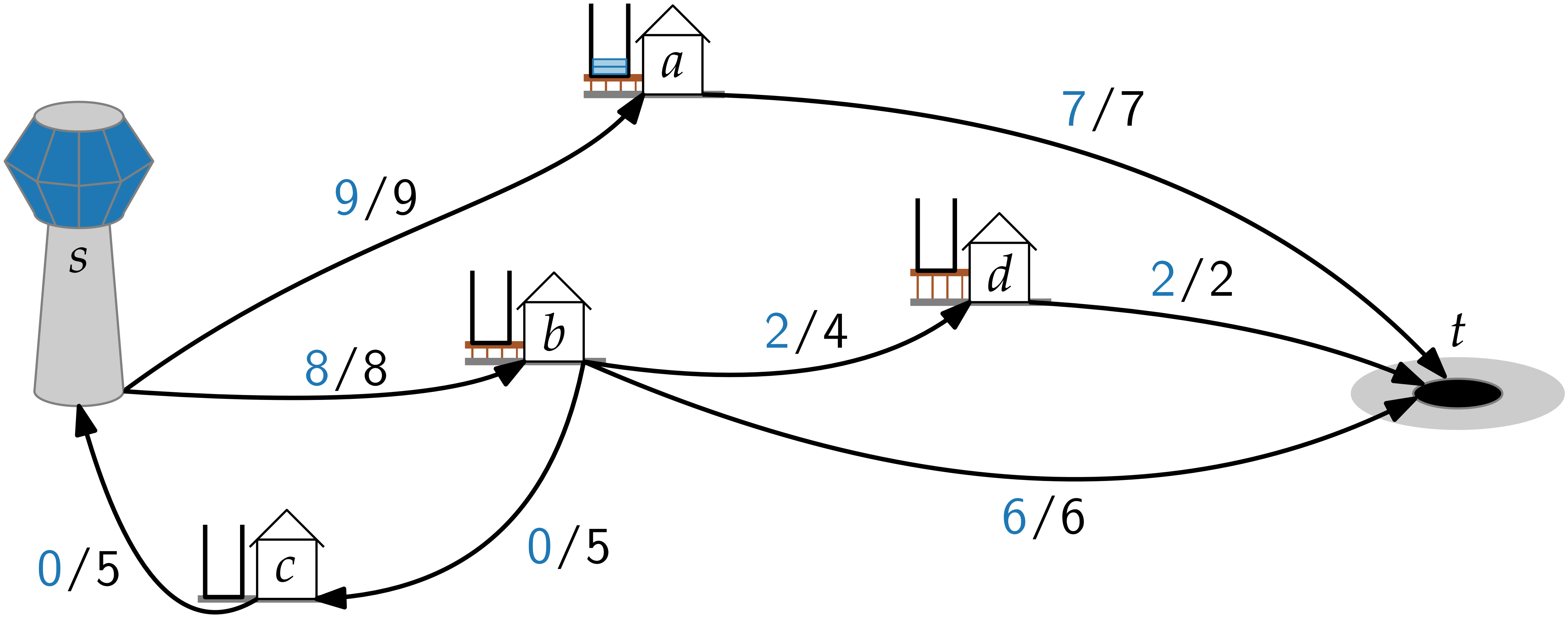
The Push–Relabel Idea



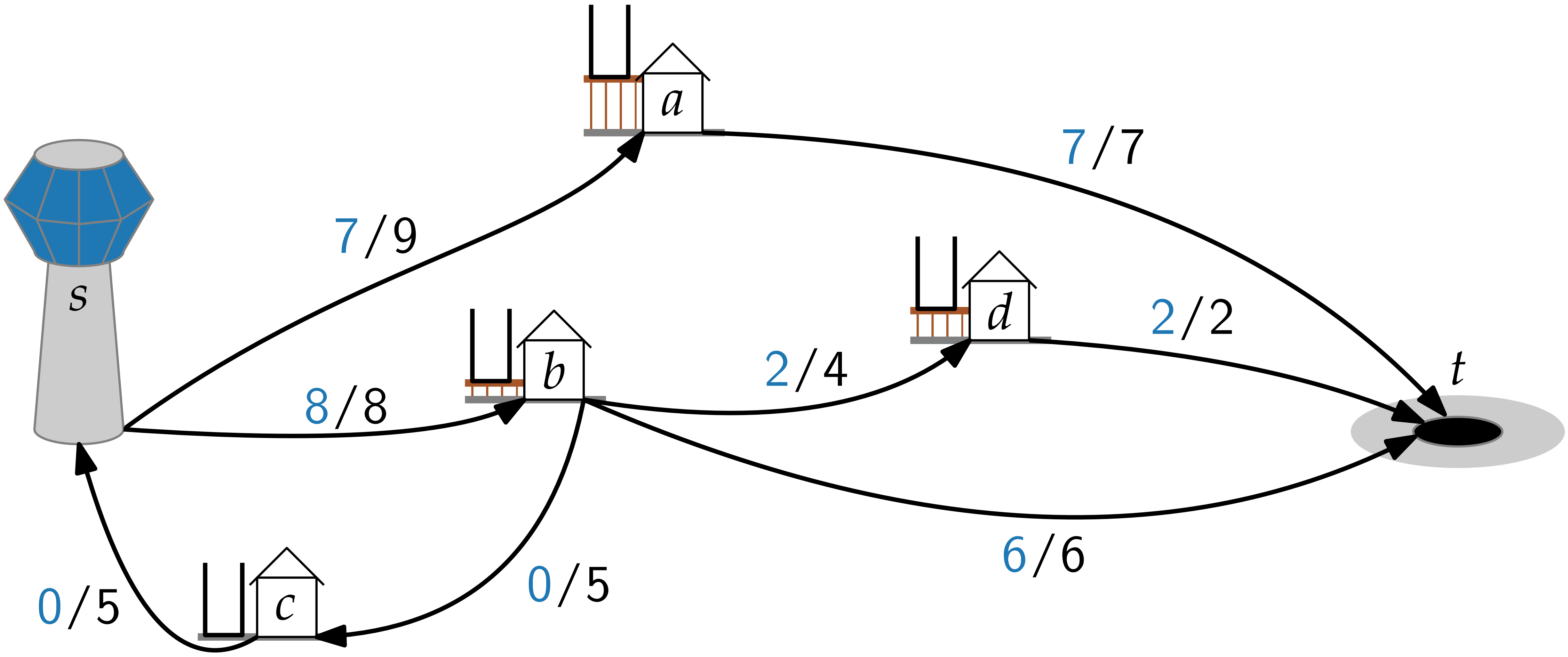
The Push–Relabel Idea



The Push–Relabel Idea



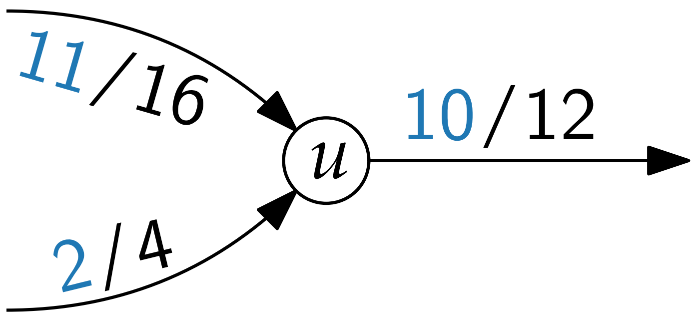
The Push–Relabel Idea



Preflow, Excess Flow, and Height

A **preflow** in G is a real-value function $f\colon V \times V \to \mathbb{R}$ that satisfies the capacity constraint and, for each $u \in V \setminus \{s\}$,

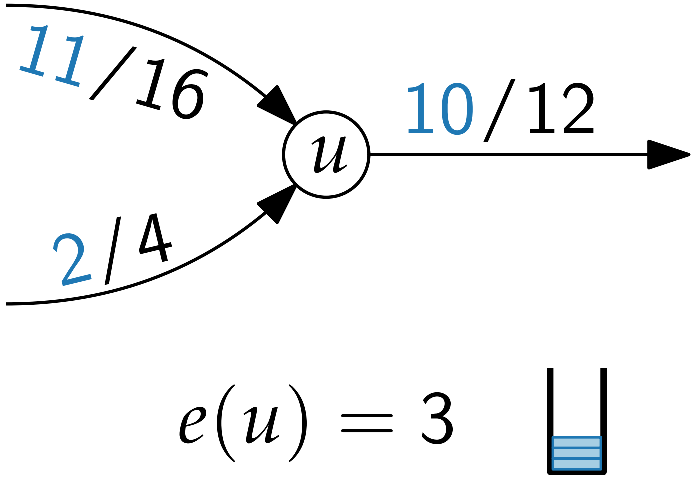
- $\sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v) \geq 0$.



Preflow, Excess Flow, and Height

A **preflow** in G is a real-value function $f\colon V \times V \to \mathbb{R}$ that satisfies the capacity constraint and, for each $u \in V \setminus \{s\}$,

- $\sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v) \geq 0.$



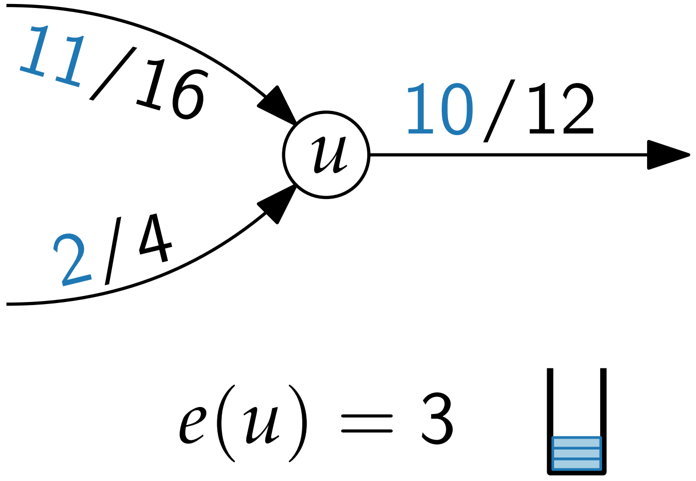
The **excess flow** of a vertex u is

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

Preflow, Excess Flow, and Height

A **preflow** in G is a real-value function $f\colon V \times V \to \mathbb{R}$ that satisfies the capacity constraint and, for each $u \in V \setminus \{s\}$,

- $\sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v) \geq 0.$



The **excess flow** of a vertex u is

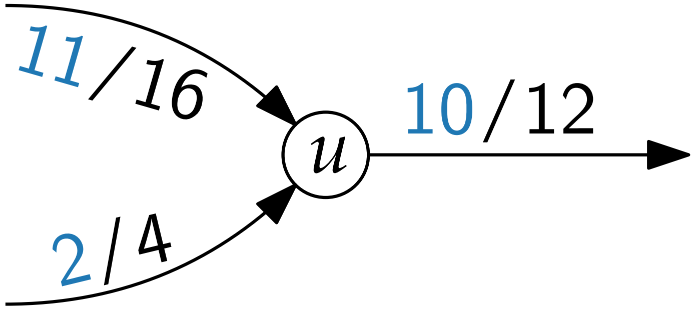
- $e(u) = \sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v).$

A vertex u is called **overflowing**, when $e(u) > 0$.

Preflow, Excess Flow, and Height

A **preflow** in G is a real-value function $f\colon V \times V \to \mathbb{R}$ that satisfies the capacity constraint and, for each $u \in V \setminus \{s\}$,

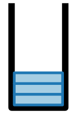
- $\sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v) \geq 0.$



The **excess flow** of a vertex u is

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

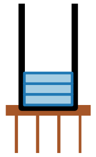
$e(u) = 3$



A vertex u is called **overflowing**, when $e(u) > 0$.

For a flow network G with preflow f, a **height function** is a function $h\colon V \to \mathbb{N}$ such that

- $h(s) = |V|$,
- $h(t) = 0$, and
- $h(u) \leq h(v) + 1$ for every residual edge $(u,v) \in E_f$.



The PUSH Operation

PUSH(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

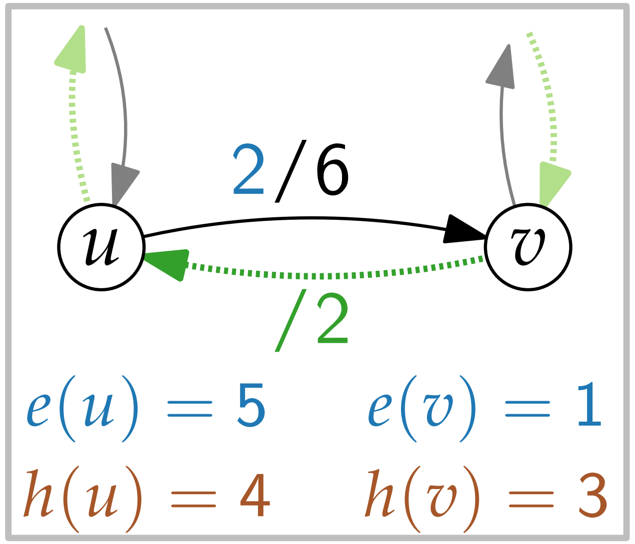
The Push Operation

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

Example.



The PUSH Operation

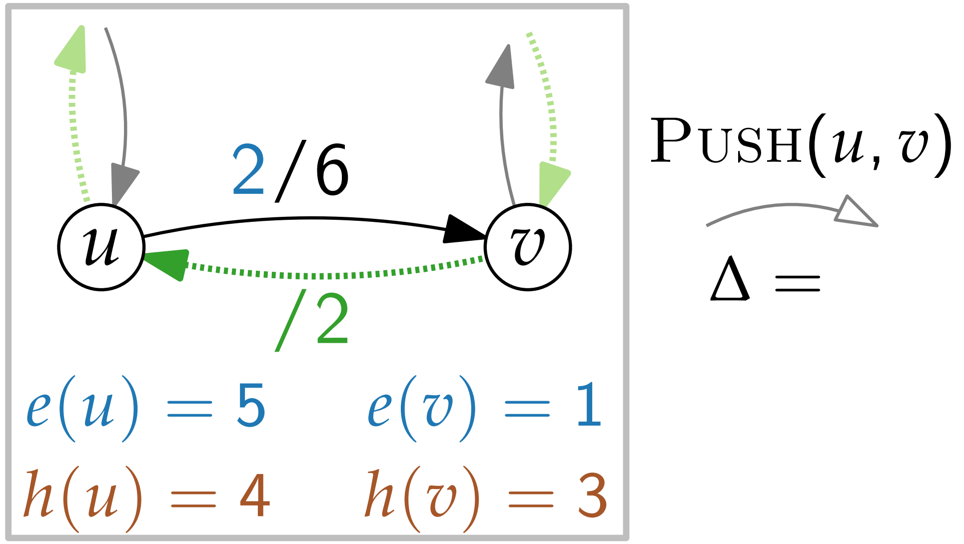
PUSH(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

$\Delta \leftarrow \min(e(u), c_f(u, v))$

Example.



The PUSH Operation

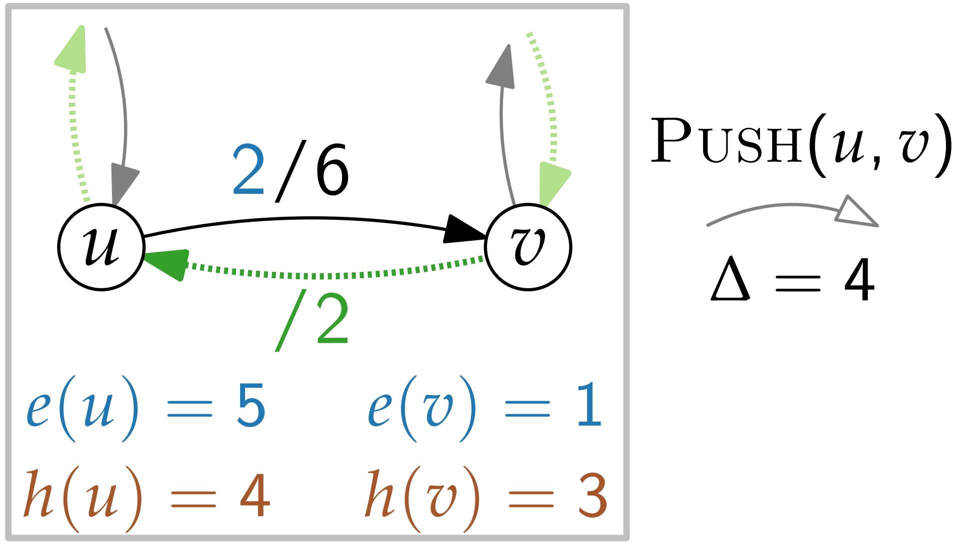
PUSH(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

$\Delta \leftarrow \min(e(u), c_f(u, v))$

Example.



The Push Operation

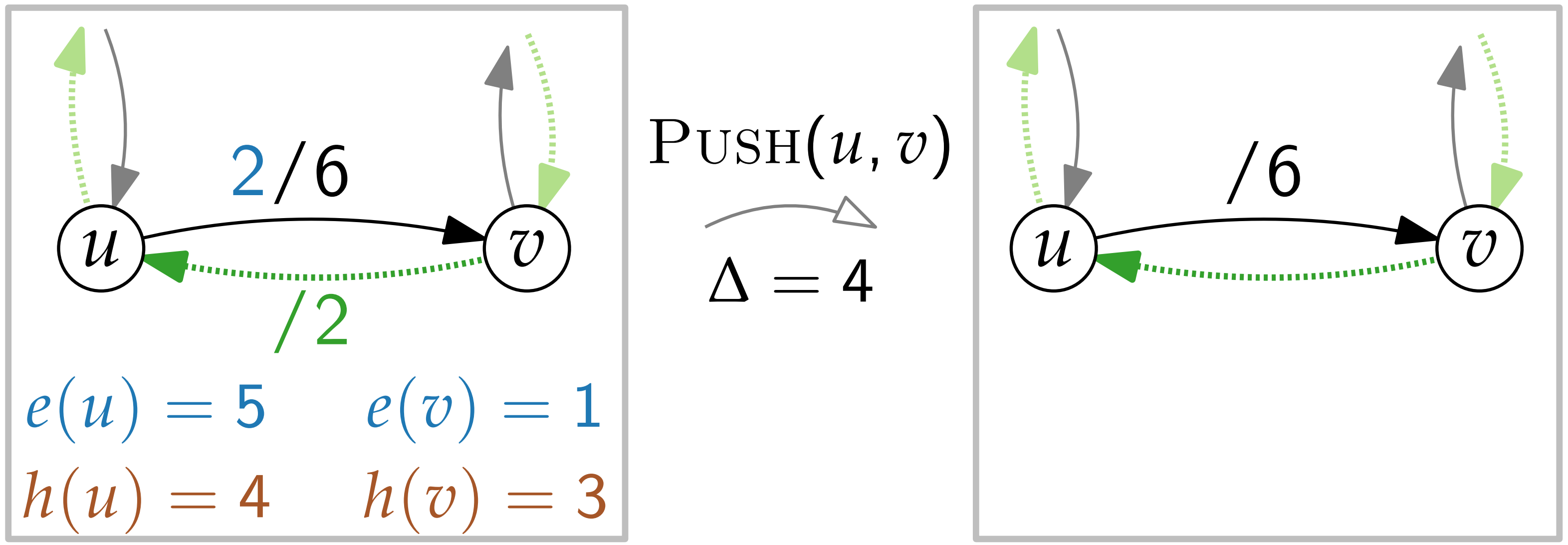
$\textsc{Push}(u, v)$

Condition: u is overflowing, $c_f(u,v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u,v))$ overflow from u to v

$\Delta \leftarrow \min(e(u), c_f(u,v))$

Example.



The Push Operation

PUSH(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

$\Delta \leftarrow \min(e(u), c_f(u, v))$

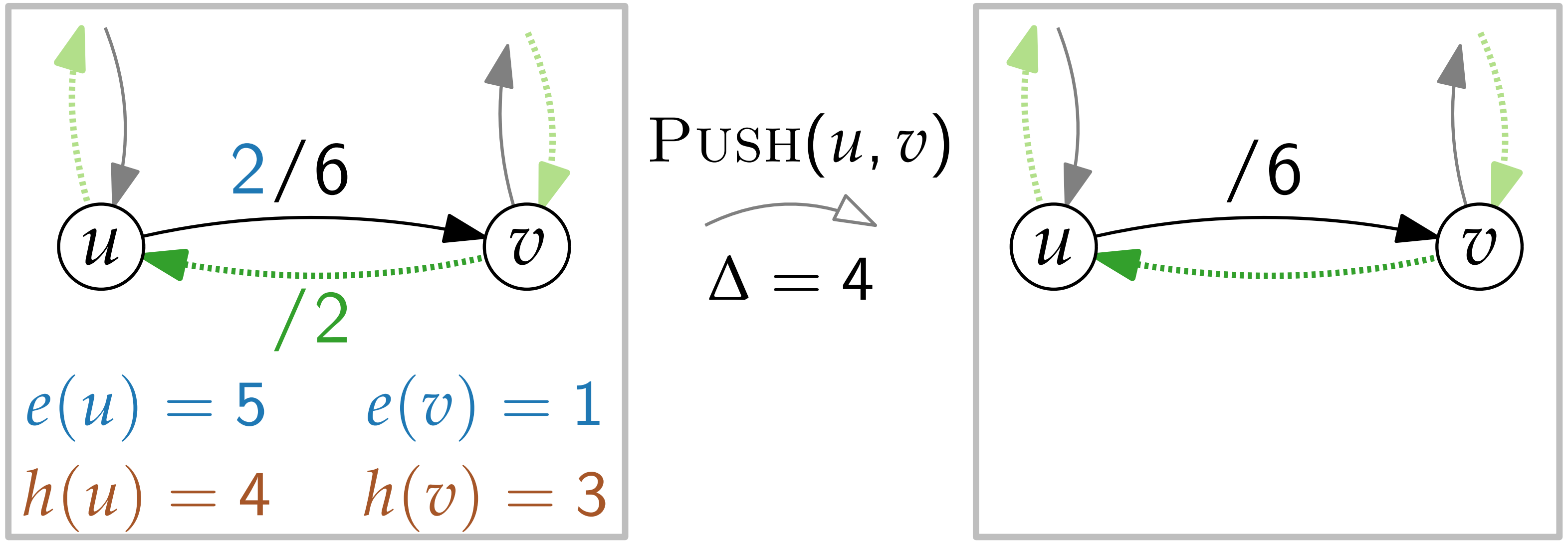
if $(u, v) \in E$ **then**

  $f(u, v) \leftarrow f(u, v) + \Delta$

else

  $f(v, u) \leftarrow f(v, u) - \Delta$

Example.



The Push Operation

$\textsc{Push}(u, v)$

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

$\Delta \leftarrow \min(e(u), c_f(u, v))$

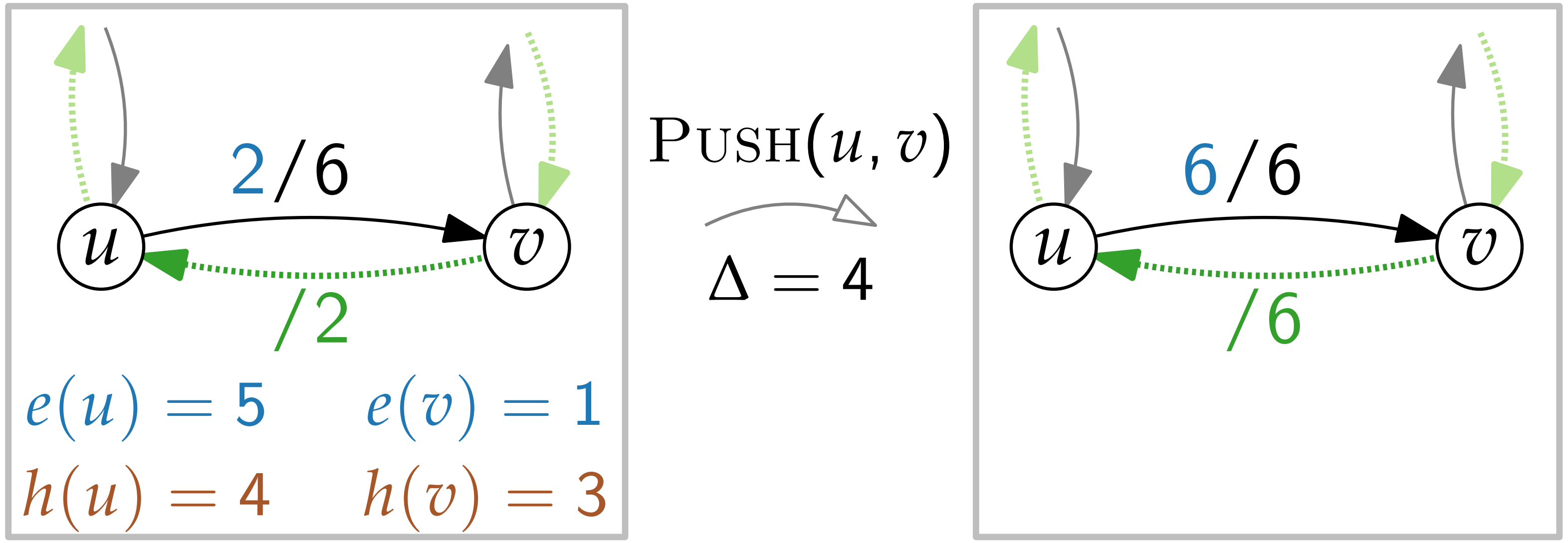
if $(u, v) \in E$ **then**

  $f(u, v) \leftarrow f(u, v) + \Delta$

else

  $f(v, u) \leftarrow f(v, u) - \Delta$

Example.



The Push Operation

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

$\Delta \leftarrow \min(e(u), c_f(u, v))$

if $(u, v) \in E$ **then**

  $f(u, v) \leftarrow f(u, v) + \Delta$

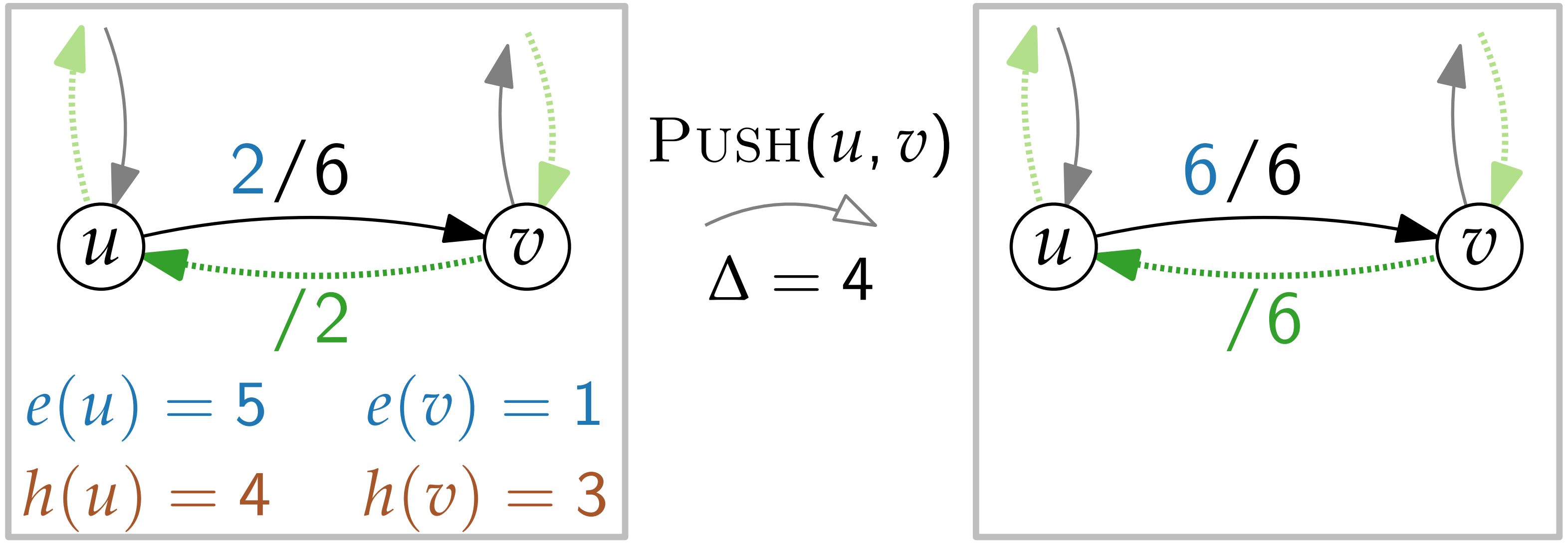
else

  $f(v, u) \leftarrow f(v, u) - \Delta$

$e(u) \leftarrow e(u) - \Delta$

$e(v) \leftarrow e(v) + \Delta$

Example.



The Push Operation

$\text{PUSH}(u, v)$

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

Effect: Push $\min(e(u), c_f(u, v))$ overflow from u to v

$\Delta \leftarrow \min(e(u), c_f(u, v))$

if $(u, v) \in E$ **then**

  $f(u, v) \leftarrow f(u, v) + \Delta$

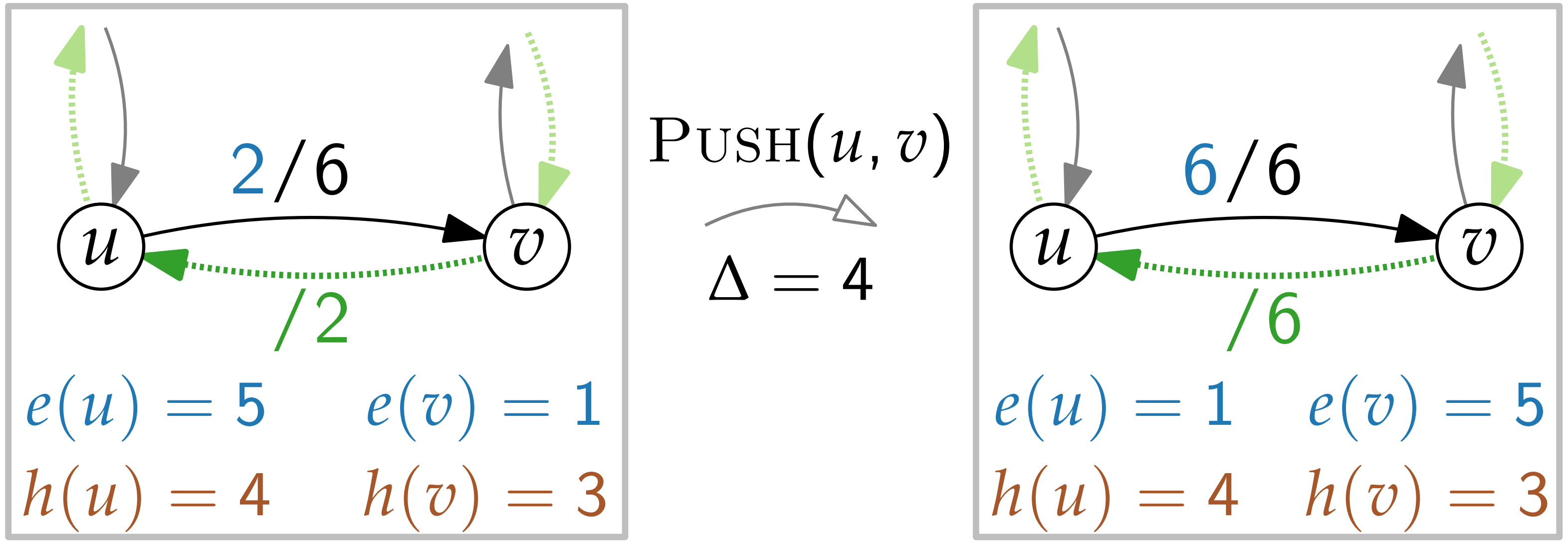
else

  $f(v, u) \leftarrow f(v, u) - \Delta$

$e(u) \leftarrow e(u) - \Delta$

$e(v) \leftarrow e(v) + \Delta$

Example.



The Relabel Operation

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ for every $v \in V$ with $(u, v) \in E_f$

Effect: Increase the height of u

$h(u) \leftarrow 1 + \min\{h(v) : v \in V \text{ with } (u, v) \in E_f\}$

The Relabel Operation

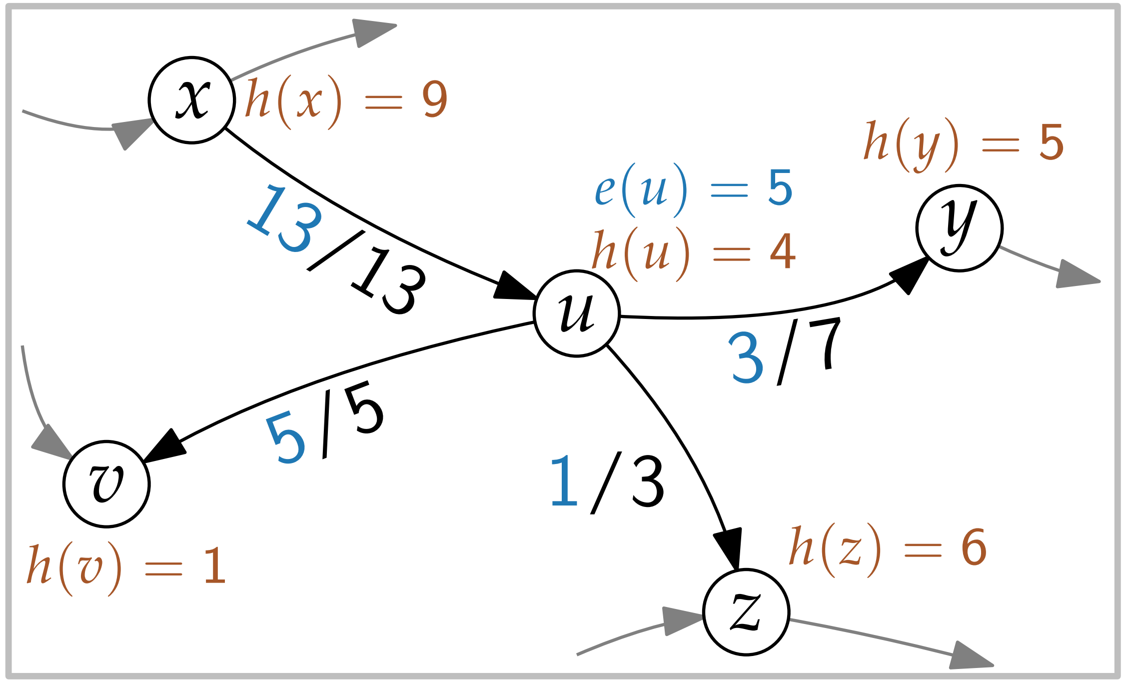
Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ for every $v \in V$ with $(u, v) \in E_f$

Effect: Increase the height of u

$h(u) \leftarrow 1 + \min\{h(v) : v \in V \text{ with } (u, v) \in E_f\}$

Example.



The Relabel Operation

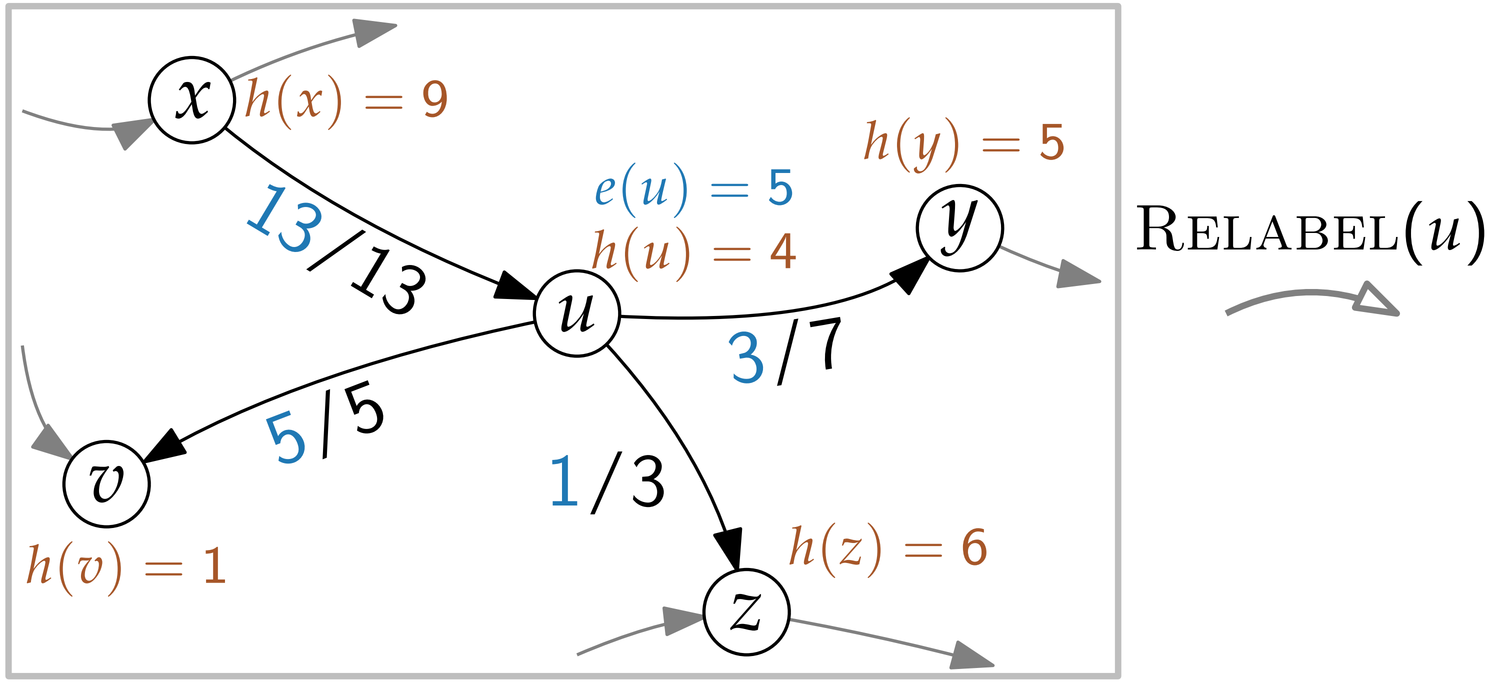
Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ for every $v \in V$ with $(u, v) \in E_f$

Effect: Increase the height of u

$h(u) \leftarrow 1 + \min\{h(v) : v \in V \text{ with } (u, v) \in E_f\}$

Example.



The Relabel Operation

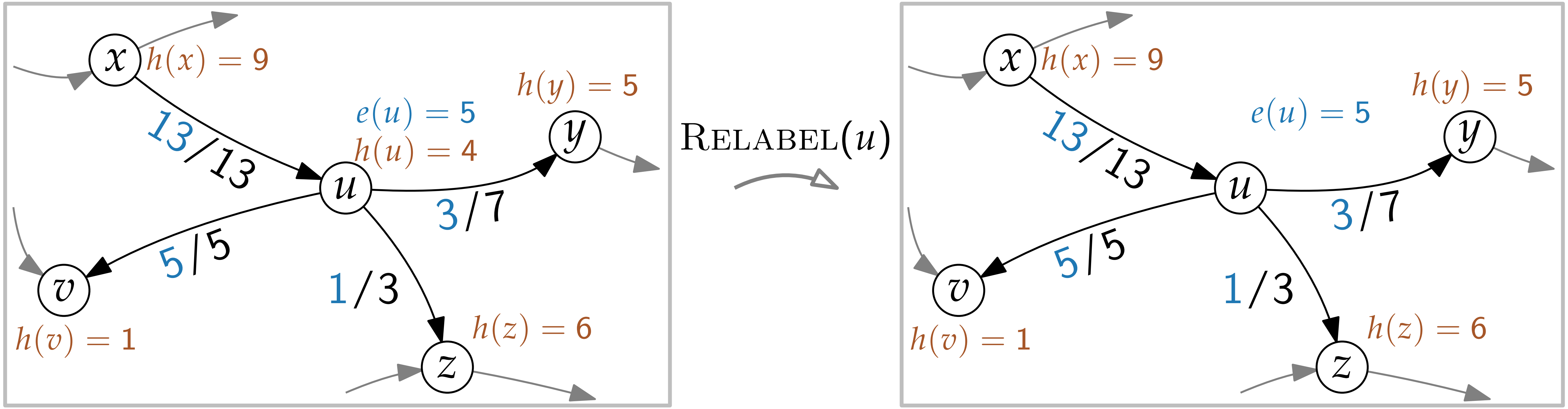
Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ for every $v \in V$ with $(u, v) \in E_f$

Effect: Increase the height of u

$h(u) \leftarrow 1 + \min\{h(v) : v \in V \text{ with } (u, v) \in E_f\}$

Example.



The Relabel Operation

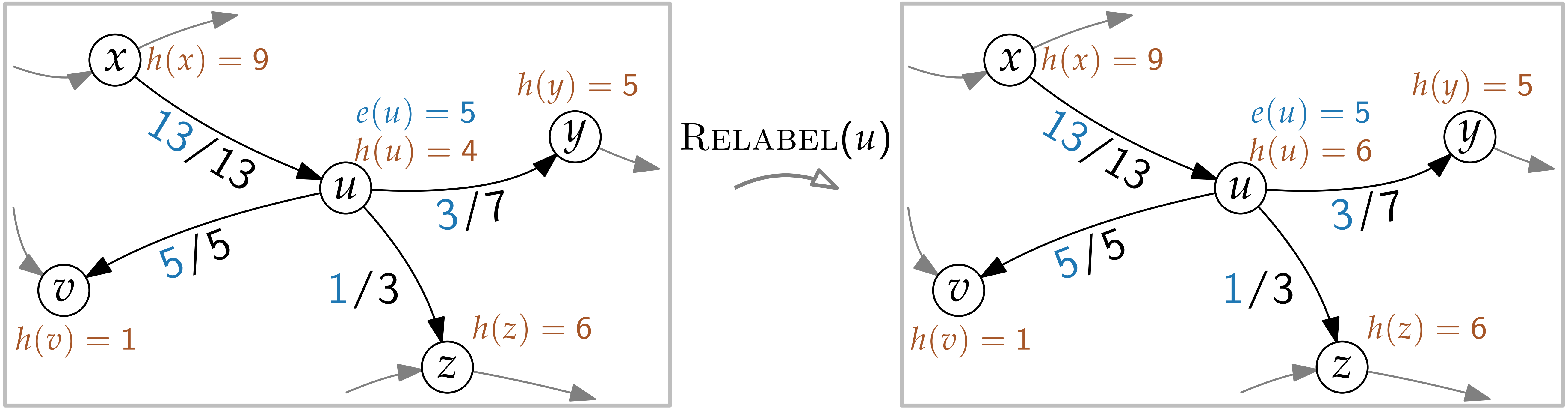
Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ for every $v \in V$ with $(u, v) \in E_f$

Effect: Increase the height of u

$h(u) \leftarrow 1 + \min\{h(v) : v \in V \text{ with } (u, v) \in E_f\}$

Example.



The Push-Relabel Algorithm

Push-Relabel(G)

InitPreflow(G, s)

while $\exists$ applicable Push or Relabel operation x **do**

⌊ apply x

The Push-Relabel Algorithm

Push-Relabel(G)

InitPreflow(G, s)
while $\exists$ applicable Push or Relabel operation x **do**
  apply x

InitPreflow(G, s)

foreach $v \in V$ **do** $h(v) \leftarrow 0$; $e(v) \leftarrow 0$
$h(s) \leftarrow |V|$
foreach $(u, v) \in E$ **do** $f(u, v) \leftarrow 0$
foreach v such that $(s, v) \in E$ **do**
  $f(s, v) \leftarrow c(s, v)$
  $e(v) \leftarrow c(s, v)$

The Push-Relabel Algorithm

Push-Relabel(G)

  InitPreflow(G, s)
  while $\exists$ applicable Push or Relabel operation x **do**
    apply x

InitPreflow(G, s)

  foreach $v \in V$ **do** $h(v) \leftarrow 0$; $e(v) \leftarrow 0$
  $h(s) \leftarrow |V|$
  foreach $(u, v) \in E$ **do** $f(u, v) \leftarrow 0$
  foreach v such that $(s, v) \in E$ **do**
    $f(s, v) \leftarrow c(s, v)$
    $e(v) \leftarrow c(s, v)$

- initializes heights
- pushes max flow over every edge that leaves s

Correctness

Part 1.

If the algorithm terminates, the preflow is a maximum flow.

- If an overflowing vertex exists, the algorithm can continue.
- The algorithm maintains f as a preflow and h as a height function.
- The sink t is not reachable from source s in G_f.

Correctness

Part 1.

If the algorithm terminates, the preflow is a maximum flow.

- If an overflowing vertex exists, the algorithm can continue.
- The algorithm maintains f as a preflow and h as a height function.
- The sink t is not reachable from source s in G_f.

Part 2.

The algorithm terminates and the heights stay finite.

- Find upper bound on heights.
- Find upper bound for the number of calls to $\textsc{Relabel}$.
- Find upper bound for the number of calls to $\textsc{Push}$.

Continuation

Lemma 1.
If a vertex u is overflowing, either a push or a relabel operation applies to u.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

$\textsc{Push}(u, v)$

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

$\textsc{Relabel}(u)$

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Continuation

Lemma 1.
If a vertex u is overflowing, either a push or a relabel operation applies to u.

Proof.
Assuming $h(u)$ is valid, we have

- $h(u) \leq h(v) + 1$ for all v with $(u, v) \in E_f$.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Continuation

Lemma 1.
If a vertex u is overflowing, either a push or a relabel operation applies to u.

Proof.
Assuming $h(u)$ is valid, we have
- $h(u) \leq h(v) + 1$ for all v with $(u, v) \in E_f$.

If no push operation is valid for $(u, v) \in E_f$, then
- $h(u) \leq h(v)$ for all v with $(u, v) \in E_f$.

Height function:
- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

PUSH(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

RELABEL(u)

Condition: u is overflowing and
  $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Continuation

Lemma 1.
If a vertex u is overflowing, either a push or a relabel operation applies to u.

Proof.
Assuming $h(u)$ is valid, we have

- $h(u) \leq h(v) + 1$ for all v with $(u, v) \in E_f$.

If no push operation is valid for $(u, v) \in E_f$, then

- $h(u) \leq h(v)$ for all v with $(u, v) \in E_f$.

Therefore, $\textsc{Relabel}(u)$ is applicable.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

$\textsc{Push}(u, v)$

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

$\textsc{Relabel}(u)$

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Maintaining the Preflow

Lemma 2.
The push-relabel algorithm maintains a preflow f.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

$\textsc{Push}(u, v)$

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

$\Delta \leftarrow \min(e(u), c_f(u, v))$

if $(u, v) \in E$ **then**

$\quad f(u, v) \leftarrow f(u, v) + \Delta$

else

$\quad f(v, u) \leftarrow f(v, u) + \Delta$

$e(u) \leftarrow e(u) - \Delta$

$e(v) \leftarrow e(v) + \Delta$

$\textsc{Relabel}(u)$

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) \colon (u, v) \in E_f\}$

Maintaining the Preflow

Lemma 2.
The push-relabel algorithm maintains a preflow f.

Proof.

- InitPreflow initialises a preflow f. ✓
- Relabel(u) doesn't affect f. ✓
- Push(u, v) maintains f as a preflow. ✓

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

$\Delta \leftarrow \min(e(u), c_f(u, v))$

if $(u, v) \in E$ **then**

  $f(u, v) \leftarrow f(u, v) + \Delta$

else

  $f(v, u) \leftarrow f(v, u) + \Delta$

$e(u) \leftarrow e(u) - \Delta$

$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Maintaining the Height Function

Lemma 3.
The push–relabel algorithm maintains h as a height function.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) \colon (u, v) \in E_f\}$

Maintaining the Height Function

Lemma 3.
The push–relabel algorithm maintains h as a height function.

Proof.

- InitPreflow initialises h as a height function. ✓

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Maintaining the Height Function

Lemma 3.
The push–relabel algorithm maintains h as a height function.

Proof.

- InitPreflow initialises h as a height function. ✓
- Under $\textsc{Push}(u,v)$, h remains a height function:

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u,v) \in E_f$

$\textsc{Push}(u,v)$

Condition: u is overflowing, $c_f(u,v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u,v))$
if $(u,v) \in E$ **then**
  $f(u,v) \leftarrow f(u,v) + \Delta$
else
  $f(v,u) \leftarrow f(v,u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

$\textsc{Relabel}(u)$

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u,v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) : (u,v) \in E_f\}$

Maintaining the Height Function

Lemma 3.
The push–relabel algorithm maintains h as a height function.

Proof.

- InitPreflow initialises h as a height function. ✓
- Under Push(u, v), h remains a height function:
  - If (v, u) is added to E_f, then $h(v) = h(u) - 1 < h(u) + 1$.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \le h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \le h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) \colon (u, v) \in E_f\}$

Maintaining the Height Function

Lemma 3.
The push–relabel algorithm maintains h as a height function.

Proof.

- InitPreflow initialises h as a height function. ✓
- Under Push(u, v), h remains a height function:
  - If (v, u) is added to E_f, then $h(v) = h(u) - 1 < h(u) + 1$. ✓
  - If (u, v) is removed from E_f, then ✓ .

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Maintaining the Height Function

Lemma 3.
The push–relabel algorithm maintains h as a height function.

Proof.

- InitPreflow initialises h as a height function. ✓
- Under Push(u, v), h remains a height function:
  - If (v, u) is added to E_f, then $h(v) = h(u) - 1 < h(u) + 1$. ✓
  - If (u, v) is removed from E_f, then ✓.
- Under Relabel(u), h remains a height function:
  - $(u, v) \in E_f$, then $h(u) \leq h(v) + 1$
  - $(w, u) \in E_f$, then $h(w) < h(u) + 1$

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)\ \forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Maintaining the Height Function

Lemma 3.
The push–relabel algorithm maintains h as a height function.

Proof.

- InitPreflow initialises h as a height function. ✓
- Under Push(u, v), h remains a height function:
  - If (v, u) is added to E_f, then $h(v) = h(u) - 1 < h(u) + 1$. ✓
  - If (u, v) is removed from E_f, then ✓.
- Under Relabel(u), h remains a height function: ✓
  - $(u, v) \in E_f$, then $h(u) \leq h(v) + 1$
  - $(w, u) \in E_f$, then $h(w) < h(u) + 1$

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Reachability of the Sink

Lemma 4.
During the execution of the push–relabel algorithm, there is no path from s to t in G_f.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Reachability of the Sink

Lemma 4.
During the execution of the push–relabel algorithm, there is no path from s to t in G_f.

Proof.
Suppose there is a path $s = v_0, v_1, \ldots, v_k = t$ in G_f.
Then

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Reachability of the Sink

Lemma 4.
During the execution of the push–relabel algorithm, there is no path from s to t in G_f.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Proof.
Suppose there is a path $s = v_0, v_1, \ldots, v_k = t$ in G_f.
Then

- $(v_i, v_{i+1}) \in E_f$ for $0 \leq i \leq k - 1$, and

Reachability of the Sink

Lemma 4.
During the execution of the push–relabel algorithm, there is no path from s to t in G_f.

Proof.
Suppose there is a path $s = v_0, v_1, \ldots, v_k = t$ in G_f.
Then

- $(v_i, v_{i+1}) \in E_f$ for $0 \le i \le k-1$, and
- $h(v_i) \le h(v_{i+1}) + 1$ for $0 \le i \le k-1$.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \le h(v) + 1 \quad \forall (u,v) \in E_f$

Reachability of the Sink

Lemma 4.
During the execution of the push–relabel algorithm, there is no path from s to t in G_f.

Proof.
Suppose there is a path $s = v_0, v_1, \ldots, v_k = t$ in G_f.
Then

- $(v_i, v_{i+1}) \in E_f$ for $0 \le i \le k-1$, and
- $h(v_i) \le h(v_{i+1}) + 1$ for $0 \le i \le k-1$.

$$\Rightarrow h(s) \le h(t) + k = k$$

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \le h(v) + 1 \quad \forall (u,v) \in E_f$

Reachability of the Sink

Lemma 4.
During the execution of the push–relabel algorithm, there is no path from s to t in G_f.

Proof.
Suppose there is a path $s = v_0, v_1, \ldots, v_k = t$ in G_f.
Then

- $(v_i, v_{i+1}) \in E_f$ for $0 \le i \le k-1$, and
- $h(v_i) \le h(v_{i+1}) + 1$ for $0 \le i \le k-1$.

$$\Rightarrow h(s) \le h(t) + k = k$$

But since $k < |V|$, it follows that $h(s) < |V|$. ✗

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \le h(v) + 1 \quad \forall (u, v) \in E_f$

Correctness of the Algorithm (Part I)

Theorem 5.
When the push–relabel algorithm terminates, the computed preflow f is a maximum flow.

Correctness of the Algorithm (Part I)

Theorem 5.
When the push–relabel algorithm terminates, the computed preflow f is a maximum flow.

Proof.

- By Lemma 1, the algorithm stops when there is no overflowing vertex.
- By Lemma 2, f is a preflow.

Correctness of the Algorithm (Part I)

Theorem 5.
When the push–relabel algorithm terminates, the computed preflow f is a maximum flow.

Proof.

- By Lemma 1, the algorithm stops when there is no overflowing vertex.
- By Lemma 2, f is a preflow.
  $\Rightarrow$ f is a flow.

Correctness of the Algorithm (Part I)

Theorem 5.
When the push–relabel algorithm terminates, the computed preflow f is a maximum flow.

Proof.

- By Lemma 1, the algorithm stops when there is no overflowing vertex.
- By Lemma 2, f is a preflow.
  $\Rightarrow$ f is a flow.
- By Lemma 3, h is a height function.
- So by Lemma 4, there is no s–t path in G_f.

Correctness of the Algorithm (Part I)

Theorem 5.
When the push–relabel algorithm terminates, the computed preflow f is a maximum flow.

Proof.

- By Lemma 1, the algorithm stops when there is no overflowing vertex.
- By Lemma 2, f is a preflow.
  $\Rightarrow$ f is a flow.
- By Lemma 3, h is a height function.
- So by Lemma 4, there is no s–t path in G_f.

  $\Rightarrow$ By the Max-Flow Min-Cut Theorem, the flow f is a maximum flow.

Correctness

Part 1. ✓

If the algorithm terminates, the preflow is maximum flow.

- If an overflowing vertex exists, the algorithm can continue.
- The algorithm maintains f as a preflow and h as a height function.
- Sink t is not reachable from source s in G_f.

Correctness

Part 1. ✓

If the algorithm terminates, the preflow is maximum flow.

- If an overflowing vertex exists, the algorithm can continue.
- The algorithm maintains f as a preflow and h as a height function.
- Sink t is not reachable from source s in G_f.

Part 2.

The algorithm terminates and the heights stay finite.

- Find upper bound on heights.
- Find upper bound for the number of calls to $\textsc{Relabel}$.
- Find upper bound for the number of calls to $\textsc{Push}$.

Reachability of the Source in the Residual Graph

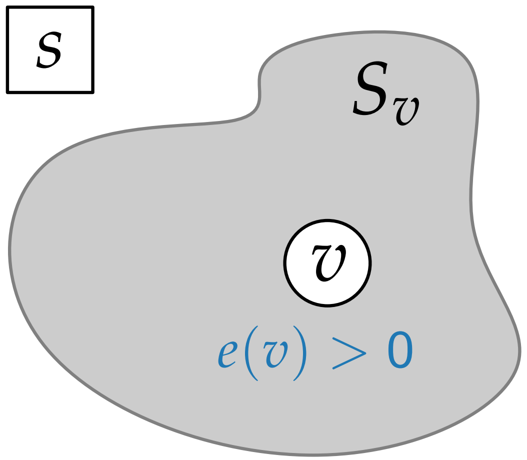
Lemma 6.
For every overflowing vertex v,
there is a path from v to s in G_f.

Reachability of the Source in the Residual Graph

Lemma 6.
For every overflowing vertex v, there is a path from v to s in G_f.

Proof.

- $S_v \leftarrow$ set of vertices reachable from v in G_f.
- Suppose that $s \notin S_v$.



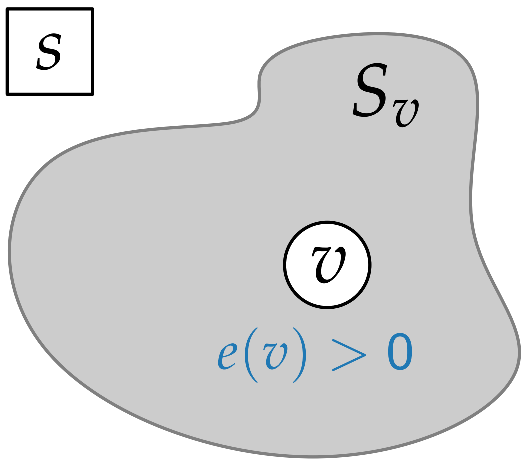
0.

Reachability of the Source in the Residual Graph

Lemma 6.
For every overflowing vertex v, there is a path from v to s in G_f.

Proof.

- $S_v \leftarrow$ set of vertices reachable from v in G_f.
- Suppose that $s \notin S_v$.
- Since f is a preflow and $s \notin S_v$, we have $\sum_{w \in S_v} e(w) \geq 0$.

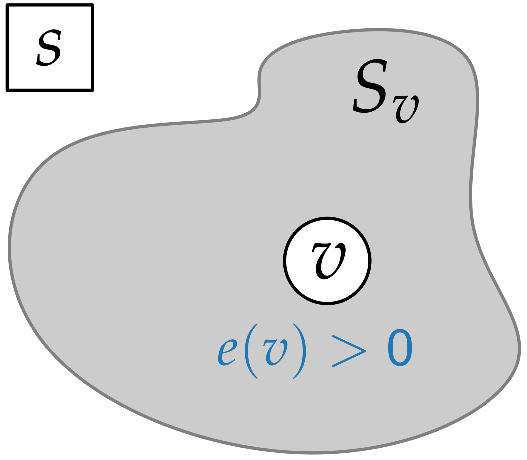


Reachability of the Source in the Residual Graph

Lemma 6.
For every overflowing vertex v, there is a path from v to s in G_f.

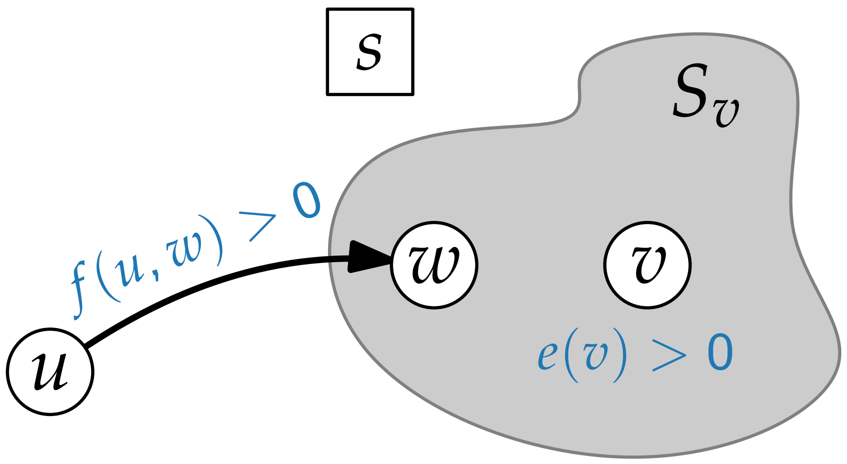
Proof.

- $S_v \leftarrow$ set of vertices reachable from v in G_f.
- Suppose that $s \notin S_v$.
- Since f is a preflow and $s \notin S_v$, we have $\sum_{w \in S_v} e(w) \geq 0$.
- Since $v \in S_v$, we even have $\sum_{w \in S_v} e(w) > 0$.



Reachability of the Source in the Residual Graph

Lemma 6.
For every overflowing vertex v,
there is a path from v to s in G_f.

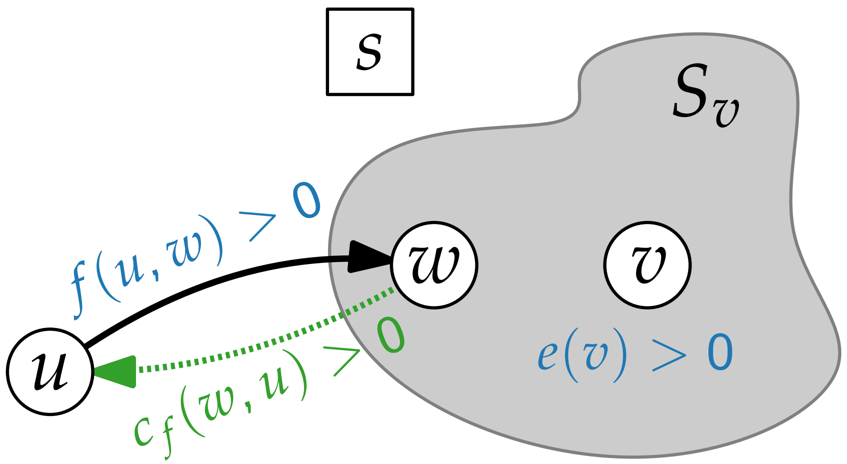


Proof.

- $S_v \leftarrow$ set of vertices reachable from v in G_f.
- Suppose that $s \notin S_v$.
- Since f is a preflow and $s \notin S_v$, we have $\sum_{w \in S_v} e(w) \geq 0$.
- Since $v \in S_v$, we even have $\sum_{w \in S_v} e(w) > 0$.
- There is an edge (u, w) with $u \notin S_v, w \in S_v$ and $f(u, w) > 0$.

Reachability of the Source in the Residual Graph

Lemma 6.
For every overflowing vertex v, there is a path from v to s in G_f.



Proof.

- $S_v \leftarrow$ set of vertices reachable from v in G_f.
- Suppose that $s \notin S_v$.
- Since f is a preflow and $s \notin S_v$, we have $\sum_{w \in S_v} e(w) \geq 0$.
- Since $v \in S_v$, we even have $\sum_{w \in S_v} e(w) > 0$.
- There is an edge (u, w) with $u \notin S_v, w \in S_v$ and $f(u, w) > 0$.
- But then $c_f(w, u) > 0$, meaning u is reachable from v. ✗

Upper Bound on the Height

Lemma 7.
During the push–relabel algorithm,
we have $h(v) \leq 2|V| - 1$ for all $v \in V$.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Relabel(u)

Condition: u is overflowing and
$h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) \colon (u, v) \in E_f\}$

Upper Bound on the Height

Lemma 7.
During the push–relabel algorithm, we have $h(v) \leq 2|V| - 1$ for all $v \in V$.

Proof.

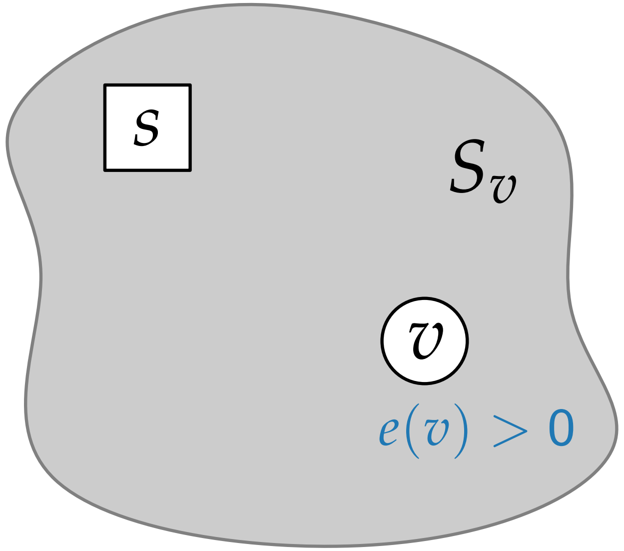
- Statement holds after initialisation.
- Let v be an overflowing vertex that is relabeled.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

RELABEL(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) \colon (u, v) \in E_f\}$



Upper Bound on the Height

Lemma 7.
During the push–relabel algorithm, we have $h(v) \leq 2|V| - 1$ for all $v \in V$.

Proof.

- Statement holds after initialisation.
- Let v be an overflowing vertex that is relabeled.
- By Lemma 6, there is a path $v = v_0, v_1, \ldots, v_k = s$ in G_f.

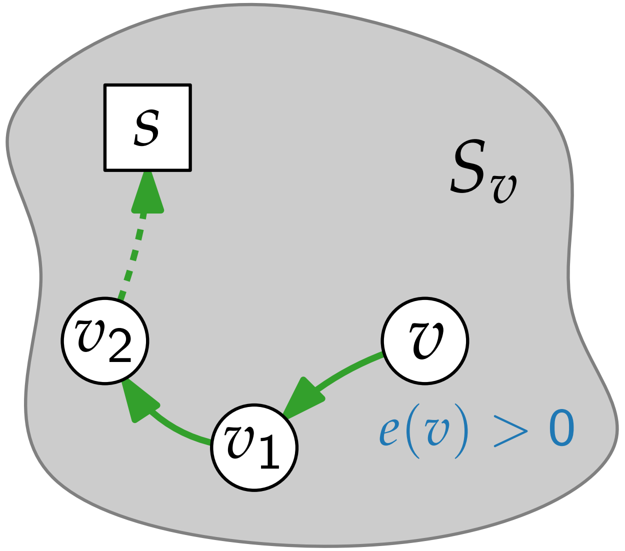
Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

RELABEL(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$



Upper Bound on the Height

Lemma 7.
During the push–relabel algorithm, we have $h(v) \leq 2|V| - 1$ for all $v \in V$.

Proof.

- Statement holds after initialisation.
- Let v be an overflowing vertex that is relabeled.
- By Lemma 6, there is a path $v = v_0, v_1, \ldots, v_k = s$ in G_f.
- Then $h(v_i) \leq h(v_{i+1}) + 1$ for $0 \leq i \leq k - 1$.

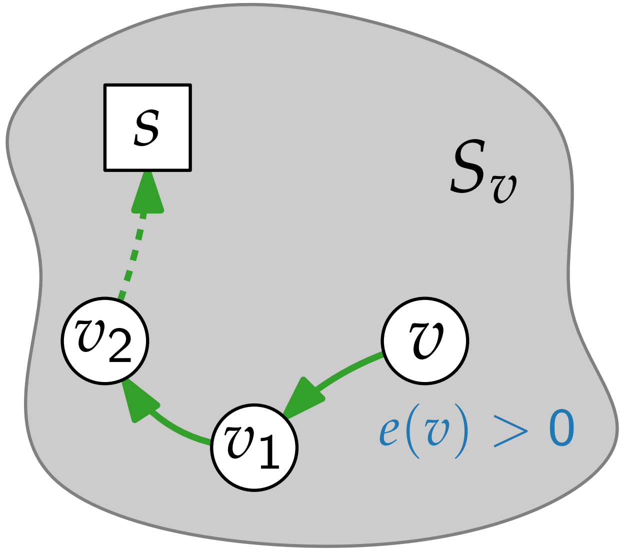
Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)\ \forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$



Upper Bound on the Height

Lemma 7.
During the push–relabel algorithm, we have $h(v) \leq 2|V| - 1$ for all $v \in V$.

Proof.

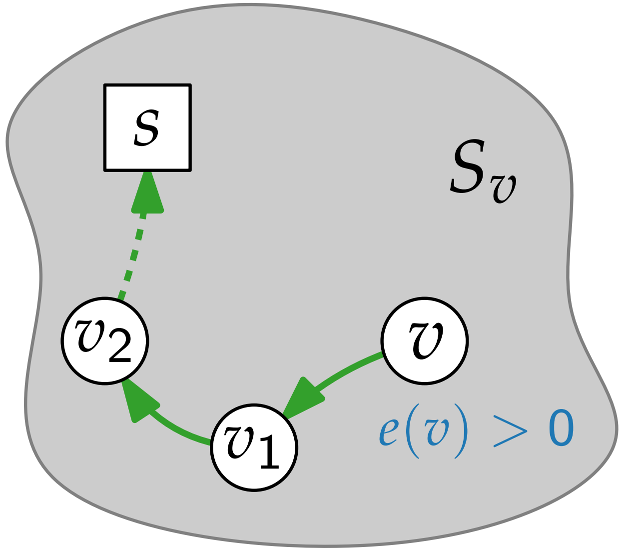
- Statement holds after initialisation.
- Let v be an overflowing vertex that is relabeled.
- By Lemma 6, there is a path $v = v_0, v_1, \ldots, v_k = s$ in G_f.
- Then $h(v_i) \leq h(v_{i+1}) + 1$ for $0 \leq i \leq k - 1$.
- Since $k \leq |V| - 1$, we have $h(v) \leq h(s) + k \leq 2|V| - 1$.

Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

RELABEL(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$
$h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$



Upper Bounds on the Height and # Relabel Operations

Lemma 7.
During the push–relabel algorithm, we have $h(v) \leq 2|V| - 1$ for all $v \in V$.

Proof.

- Statement holds after initialisation.
- Let v be an overflowing vertex that is relabeled.
- By Lemma 6, there is a path $v = v_0, v_1, \ldots, v_k = s$ in G_f.
- Then $h(v_i) \leq h(v_{i+1}) + 1$ for $0 \leq i \leq k - 1$.
- Since $k \leq |V| - 1$, we have $h(v) \leq h(s) + k \leq 2|V| - 1$.

Corollary 8.
The push–relabel algorithm executes at most $2|V|^2$ Relabel operations.

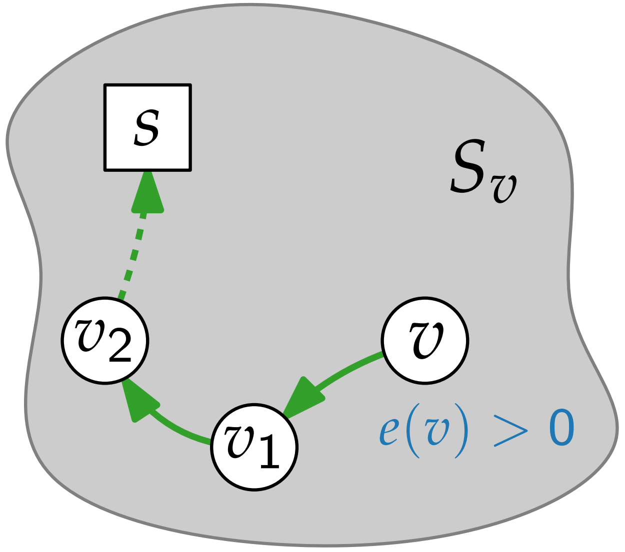
Height function:

- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$

Relabel(u)

Condition: u is overflowing and $h(u) \leq h(v)$ $\forall v \in V$ with $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min\{h(v) \colon (u, v) \in E_f\}$



Saturating and Unsaturating $\textsc{Push}$ Operations

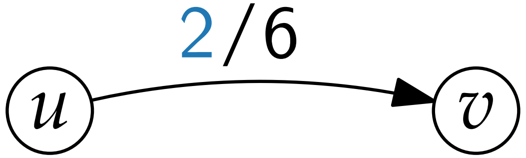
The operation $\textsc{Push}(u, v)$ is

- **saturating** if afterwards $c_f(u, v) = 0$,

Saturating and Unsaturating PUSH Operations

The operation PUSH(u, v) is

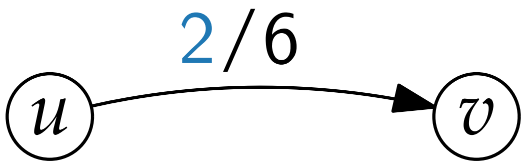
- **saturating** if afterwards $c_f(u, v) = 0$,



Saturating and Unsaturating Push Operations

The operation $\textsc{Push}(u, v)$ is

- **saturating** if afterwards $c_f(u, v) = 0$,



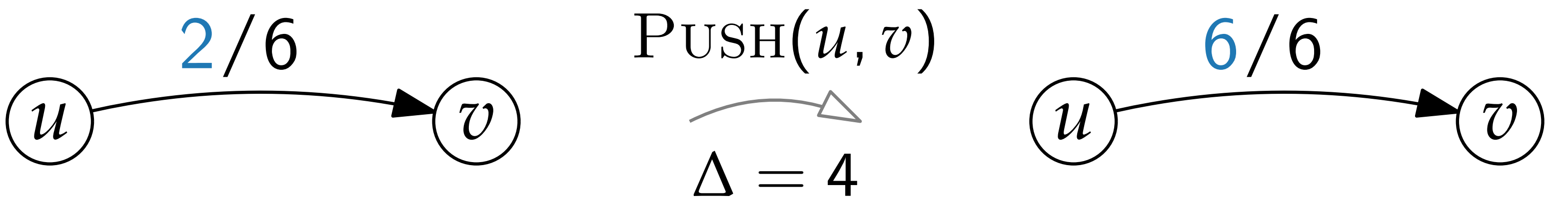
$\textsc{Push}(u, v)$

$\Delta = 4$

Saturating and Unsaturating PUSH Operations

The operation $\textsc{Push}(u, v)$ is

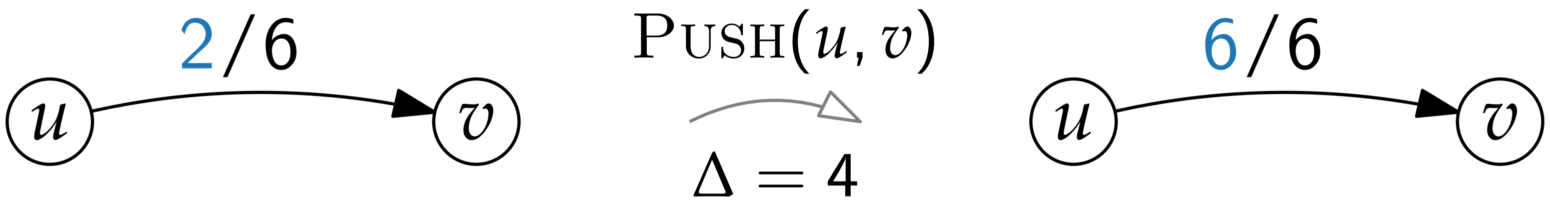
- **saturating** if afterwards $c_f(u, v) = 0$,



Saturating and Unsaturating PUSH Operations

The operation PUSH(u, v) is

- **saturating** if afterwards $c_f(u, v) = 0$,

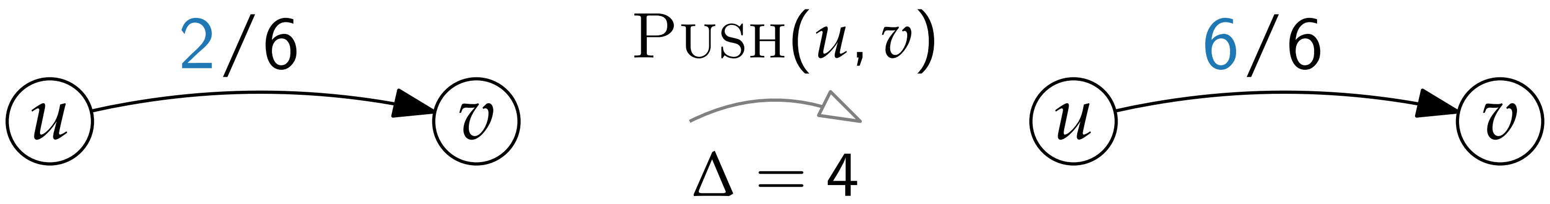


- and **unsaturating** otherwise.

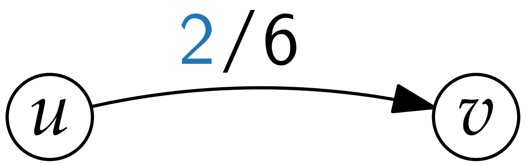
Saturating and Unsaturating Push Operations

The operation $\textsc{Push}(u, v)$ is

- **saturating** if afterwards $c_f(u, v) = 0$,



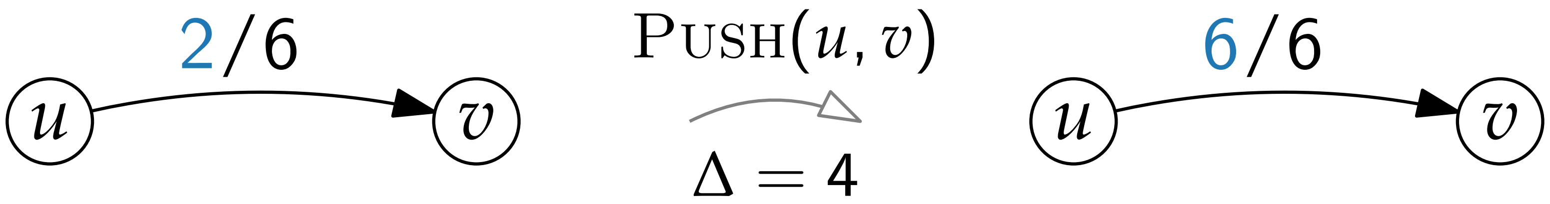
- and **unsaturating** otherwise.



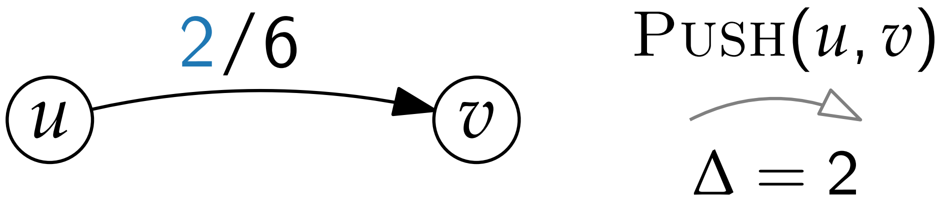
Saturating and Unsaturating PUSH Operations

The operation PUSH(u, v) is

- **saturating** if afterwards $c_f(u, v) = 0$,



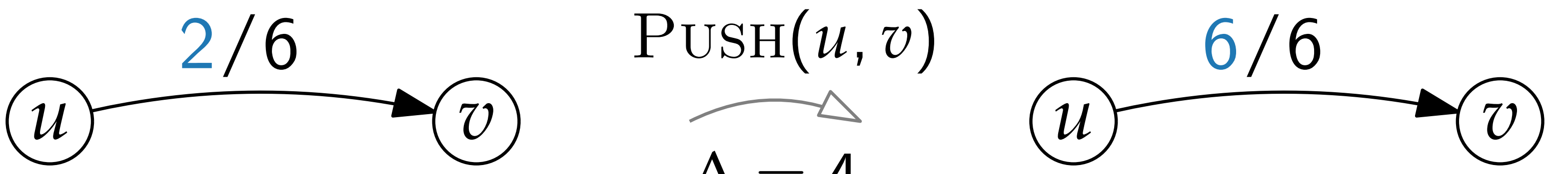
- and **unsaturating** otherwise.



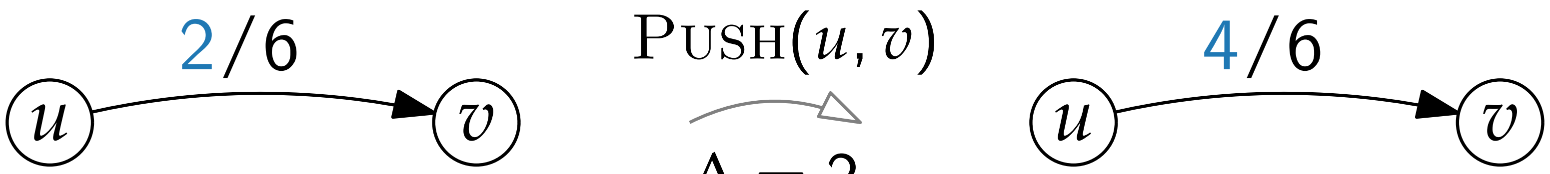
Saturating and Unsaturating PUSH Operations

The operation $\textsc{Push}(u, v)$ is

- **saturating** if afterwards $c_f(u, v) = 0$,



- and **unsaturating** otherwise.



Upper Bound on the Number of Saturating Push Operations

Lemma 9.
The push–relabel algorithm executes at most $2|V| \cdot |E|$ saturating Push operations.

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Upper Bound on the Number of Saturating Push Operations

Lemma 9.
The push–relabel algorithm executes at most $2|V| \cdot |E|$ saturating Push operations.

Proof.

- Consider saturating Push(u, v)
  - $h(u) = h(v) + 1$

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
    $f(u, v) \leftarrow f(u, v) + \Delta$
else
    $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Push(u, v)
...

Upper Bound on the Number of Saturating PUSH Operations

Lemma 9.
The push–relabel algorithm executes at most $2|V| \cdot |E|$ saturating PUSH operations.

Proof.

- Consider saturating PUSH(u, v)
  - $h(u) = h(v) + 1$
- For another saturating PUSH(u, v), first PUSH(v, u) necessary
  - $h(v) = h(u) + 1$ necessary

PUSH(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$

```
Δ ← min(e(u), c_f(u, v))
if (u, v) ∈ E then
    f(u, v) ← f(u, v) + Δ
else
    f(v, u) ← f(v, u) + Δ
e(u) ← e(u) − Δ
e(v) ← e(v) + Δ
```

PUSH(u, v)
...
PUSH(v, u)

Upper Bound on the Number of Saturating $\textsc{Push}$ Operations

Lemma 9.
The push–relabel algorithm executes at most $2|V| \cdot |E|$ saturating $\textsc{Push}$ operations.

$\textsc{Push}(u, v)$
Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Proof.

- Consider saturating $\textsc{Push}(u, v)$
  - $h(u) = h(v) + 1$
- For another saturating $\textsc{Push}(u, v)$, first $\textsc{Push}(v, u)$ necessary
  - $h(v) = h(u) + 1$ necessary
- After another saturating $\textsc{Push}(u, v)$, both $h(u)$ and $h(v)$ have increased by at least two.

$\textsc{Push}(u, v)$
...
$\textsc{Push}(v, u)$
...
$\textsc{Push}(u, v)$
...

Upper Bound on the Number of Saturating Push Operations

Lemma 9.
The push–relabel algorithm executes at most $2|V| \cdot |E|$ saturating Push operations.

Proof.

- Consider saturating $\textsc{Push}(u, v)$
  - $h(u) = h(v) + 1$
- For another saturating $\textsc{Push}(u, v)$, first $\textsc{Push}(v, u)$ necessary
  - $h(v) = h(u) + 1$ necessary
- After another saturating $\textsc{Push}(u, v)$, both $h(u)$ and $h(v)$ have increased by at least two.
- But by Lemma 6, $h(u) \leq 2|V| - 1$ and $h(v) \leq 2|V| - 1$.

$\textsc{Push}(u, v)$

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

$\textsc{Push}(u, v)$
...
$\textsc{Push}(v, u)$
...
$\textsc{Push}(u, v)$
...

Upper Bound on the Number of Saturating PUSH Operations

Lemma 9.
The push–relabel algorithm executes at most $2|V| \cdot |E|$ saturating PUSH operations.

PUSH(u, v)
Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Proof.

- Consider saturating PUSH(u, v)
  - $h(u) = h(v) + 1$
- For another saturating PUSH(u, v), first PUSH(v, u) necessary
  - $h(v) = h(u) + 1$ necessary
- After another saturating PUSH(u, v), both $h(u)$ and $h(v)$ have increased by at least two.
- But by Lemma 6, $h(u) \leq 2|V| - 1$ and $h(v) \leq 2|V| - 1$.
- There are at most $2|V| - 1$ saturated PUSH operations for edge (u, v).

PUSH(u, v)
...
PUSH(v, u)
...
PUSH(u, v)
...

Upper Bound on the Number of Unsaturating Push Ops

Lemma 10.
The push–relabel algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating Push ops.

Push(u, v)

Condition: u is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u, v))$
if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + \Delta$
else
  $f(v, u) \leftarrow f(v, u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Upper Bound on the Number of Unsaturating Push Ops

Lemma 10.
The push–relabel algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating Push ops.

Proof.

- Consider $\mathcal{H} = \sum_{v \in V \setminus \{s,t\},\ v \text{ overflowing}} h(v)$.
- After initialization and at the end $\mathcal{H} = 0$.

Push(u, v)

Condition: u is overflowing, $c_f(u,v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u,v))$
if $(u,v) \in E$ **then**
  $f(u,v) \leftarrow f(u,v) + \Delta$
else
  $f(v,u) \leftarrow f(v,u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Upper Bound on the Number of Unsaturating Push Ops

Lemma 10.
The push–relabel algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating Push ops.

Proof.

- Consider $\mathcal{H} = \sum_{v \in V \setminus \{s,t\},\ v \text{ overflowing}} h(v)$.
- After initialization and at the end $\mathcal{H} = 0$.
- A saturating Push increases $\mathcal{H}$ by at most $2|V| - 1$.
- By Lemma 9, all saturating Push operations increase $\mathcal{H}$ by at most $(2|V| - 1) \cdot 2|V| \cdot |E|$.

Push(u, v)

Condition: u is overflowing, $c_f(u,v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u,v))$
if $(u,v) \in E$ **then**
  $f(u,v) \leftarrow f(u,v) + \Delta$
else
  $f(v,u) \leftarrow f(v,u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Upper Bound on the Number of Unsaturating PUSH Ops

Lemma 10.
The push–relabel algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating PUSH ops.

PUSH(u, v)
- **Condition:** u is overflowing, $c_f(u,v) > 0$, and $h(u) = h(v) + 1$
- $\Delta \leftarrow \min(e(u), c_f(u,v))$
- **if** $(u,v) \in E$ **then**
  - $f(u,v) \leftarrow f(u,v) + \Delta$
- **else**
  - $f(v,u) \leftarrow f(v,u) + \Delta$
- $e(u) \leftarrow e(u) - \Delta$
- $e(v) \leftarrow e(v) + \Delta$

Proof.

- Consider $\mathcal{H} = \sum_{v \in V \setminus \{s,t\},\ v \text{ overflowing}} h(v)$.
- After initialization and at the end $\mathcal{H} = 0$.
- A saturating PUSH increases $\mathcal{H}$ by at most $2|V| - 1$.
- By Lemma 9, all saturating PUSH operations increase $\mathcal{H}$ by at most $(2|V| - 1) \cdot 2|V| \cdot |E|$.
- By Lemma 7, all RELABEL operations increase $\mathcal{H}$ by at most $(2|V| - 1) \cdot |V|$.

Upper Bound on the Number of Unsaturating Push Ops

Lemma 10.
The push–relabel algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating Push ops.

Push(u, v)

Condition: u is overflowing, $c_f(u,v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), c_f(u,v))$
if $(u,v) \in E$ **then**
  $f(u,v) \leftarrow f(u,v) + \Delta$
else
  $f(v,u) \leftarrow f(v,u) + \Delta$
$e(u) \leftarrow e(u) - \Delta$
$e(v) \leftarrow e(v) + \Delta$

Proof.

- Consider $\mathcal{H} = \sum_{v \in V \setminus \{s,t\},\ v \text{ overflowing}} h(v)$.
- After initialization and at the end $\mathcal{H} = 0$.
- A saturating Push increases $\mathcal{H}$ by at most $2|V| - 1$.
- By Lemma 9, all saturating Push operations increase $\mathcal{H}$ by at most $(2|V| - 1) \cdot 2|V| \cdot |E|$.
- By Lemma 7, all Relabel operations increase $\mathcal{H}$ by at most $(2|V| - 1) \cdot |V|$.
- An unsaturating Push(u, v) decreases $\mathcal{H}$ by at least 1 since $h(u) - h(v) \geq 1$.

Termination of the Algorithm

Theorem 5.
When the push–relabel algorithm terminates, the computed preflow f is a maximum flow.

Theorem 11.
The push–relabel algorithm terminates after $\mathcal{O}(|V|^2|E|)$ valid $\textsc{Push}$ or $\textsc{Relabel}$ ops.

Proof.

- Follows by Corollary 8 and Lemmas 9+10.

Implementation

The actual running time depends on the selection order of the overflowing vertices:

- **FIFO implementation:**
  Pick overflowing vertex by *first-in-first-out* principle: $\mathcal{O}(|V|^3)$ running time.
  with dynamic trees: $\mathcal{O}(|V||E| \log \frac{|V|^2}{|E|})$
- **Highest label:**
  For $\textsc{Push}$ select highest overflowing vertex: $\mathcal{O}(|V|^2|E|^{\frac{1}{2}})$
- **Excess scaling:**
  For $\textsc{Push}(u, v)$ choose edge (u, v) such that u is overflowing, $e(u)$ is *sufficiently high* and $e(v)$ *sufficiently small*: $\mathcal{O}(|E| + |V|^2 \log C)$, where $C = \max_{(u,v)\in E} c(u, v)$

Discussion

- The push–relabel method offers an alternative framework to the Ford–Fulkerson method to develop algorithms that solve the maximum flow problem.
- Push–relabel algorithms are regarded as benchmarks for maximum flow algorithms.
- In practice, heuristics are used to improve the performance of push–relabel algorithms. Any ideas?
- The algorithm can be extended to solve the minimum-cost flow problem.

Literature

Main source:

- [CLRS Ch26] $\leftarrow$ Cormen et al. “Introduction to Algorithms”

Original paper:

- [Goldberg, Tarjan ’88] A new approach to the maximum-flow problem

Links:

- Animations of the max-flow algorithms by Ford–Fulkerson and Edmonds–Karp: https://visualgo.net/en/maxflow