Advanced Algorithms

Maximum Flow Problem

Push–relabel algorithm

Alexander Wolff · WS 2022
Flow Networks

\[ s \rightarrow a \rightarrow c \rightarrow t \]
\[ s \rightarrow b \rightarrow d \rightarrow t \]
Flow Networks

A flow network \( G = (V, E) \) is a digraph with

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

A flow network $G = (V, E)$ is a digraph with
- unique source $s$ and sink $t$,
Flow Networks

A flow network $G = (V, E)$ is a digraph with
- unique source $s$ and sink $t$,
- no antiparallel edges, and
A flow network $G = (V, E)$ is a digraph with
- unique source $s$ and sink $t$,
- no antiparallel edges, and
- a capacity $c(u, v) \geq 0$ for every $(u, v) \in E$. 

![Diagram of a flow network with labeled capacities on the edges.]

Flow Networks
Flow

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

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

![Graph showing an s–t flow with node capacities and flows labeled]
An $s$–$t$ flow in $G$ is a real-value function $f : V \times V \rightarrow \mathbb{R}$ that satisfies

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

- **capacity constraint**, 
  \[
  0 \leq f(u, v) \leq c(u, v).
  \]
An \(s-t\) flow in \(G\) is a real-value function \(f : V \times V \rightarrow \mathbb{R}\) that satisfies

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

An s–t flow in $G$ is a real-value function $f : V \times V \rightarrow \mathbb{R}$ 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

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s).$$
An \( s-t \) flow in \( G \) is a real-value function \( f : V \times V \rightarrow \mathbb{R} \) that satisfies

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

- **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 \textit{s–t flow} in \( G \) is a real-value function \( f : V \times V \rightarrow \mathbb{R} \) 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 \textit{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-value function $f : V \times V \to \mathbb{R}$ 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

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

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

\[|f| = 23\]
By How Much May Flow Change?

\[ a \xrightarrow{2/5} b \]

\[ c \]
By How Much May Flow Change?

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

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

\[ c_f(a, b) = 3 \quad \text{and} \quad c_f(b, a) = 2 \]
\[ c_f(a, c) = 0 \]
By How Much May Flow Change?

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

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

For the given graph:

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

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

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**

$c_f(u, v) = \begin{cases} 
  c(u, v) - f(u, v) & \text{if } (u, v) \in E \\
  f(v, u) & \text{if } (v, u) \in E \\
  0 & \text{otherwise.}
\end{cases}$
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
\[ E_f = \{ (u, v) \in V \times V \mid c_f(u, v) > 0 \}. \]

The flow/capacity of an edge $(u, v)$ is defined as:
\[ c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{if } (u, v) \in E \\ f(v, u) & \text{if } (v, u) \in E \\ 0 & \text{otherwise.} \end{cases} \]
The **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**

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

---

**Diagram:**

The diagram illustrates a flow network $G$ with $s–t$ flow $f$, and its residual network $G_f$. The flow capacity $c_f(u, v)$ is defined as:

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

The **residual network** \( G_f = (V, E_f) \) for a flow network \( G \) with \( s-t \) flow \( f \) has
\[
E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}.
\]

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

![Diagram of residual network and augmenting path](image)
The Algorithms of Ford–Fulkerson and Edmonds–Karp

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

\[
\begin{align*}
\text{foreach } uv \in E & \text{ do} \\
& f_{uv} \leftarrow 0 \quad \{\text{initialising zero flow}\}
\end{align*}
\]

\[
\begin{align*}
\text{while } G_f \text{ contains augmenting path } p & \text{ do} \\
& \Delta \leftarrow \min_{uv \in p} c_f(uv) \quad \{\text{residual capacity of } p\}
\end{align*}
\]

\[
\begin{align*}
\text{foreach } uv \in p & \text{ do} \\
& \quad \text{if } uv \in E \text{ then} \\
& & \quad f_{uv} \leftarrow f_{uv} + \Delta \quad \{\text{augmentation along } p\}
\end{align*}
\]

\[
\begin{align*}
& \quad \text{else} \\
& & \quad f_{vu} \leftarrow f_{vu} - \Delta
\end{align*}
\]

\[
\begin{align*}
\text{return } f \quad \{\text{return max flow}\}
\end{align*}
\]
The Algorithms of Ford–Fulkerson and Edmonds–Karp

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

\[
\text{foreach } uv \in E \text{ do } \quad \text{\{initialising zero flow\}}
\]

\[
\Delta \leftarrow \min_{uv \in p} c_f(uv) \quad \text{\{residual capacity of } p\text{\}}
\]

\[
\text{foreach } uv \in p \text{ do } \quad \text{\{augmentation along } p\text{\}}
\]

\[
\text{if } uv \in E \text{ then } \quad f_{uv} \leftarrow f_{uv} + \Delta
\]

\[
\text{else } \quad f_{vu} \leftarrow f_{vu} - \Delta
\]

\[\text{return } f\quad \text{\{return max flow\}}\]

EdmondsKarp \(\text{shortest}\)
The Algorithms of Ford–Fulkerson and Edmonds–Karp

\[\text{FordFulkerson}(G = (V, E), c, s, t)\]

\[
\begin{align*}
\text{foreach } uv \in E & \text{ do} \\
   f_{uv} & \leftarrow 0 \quad \text{\{initialising zero flow\}}
\end{align*}
\]

\[
\text{while } G_f \text{ contains a shortest augmenting path } p \text{ do}
\]

\[
\begin{align*}
\Delta & \leftarrow \min_{uv \in p} c_f(uv) \quad \text{\{residual capacity of } p\}\text{}}
\end{align*}
\]

\[
\begin{align*}
\text{foreach } uv \in p & \text{ do} \\
   \text{if } uv \in E & \text{ then} \\
   f_{uv} & \leftarrow f_{uv} + \Delta \\
   \text{else} \\
   f_{vu} & \leftarrow f_{vu} - \Delta
\end{align*}
\]

\[\text{return } f\quad \text{\{return max flow\}}
\]

\[
\begin{align*}
\text{EdmondsKarp} & \quad (G = (V, E), c, s, t)
\end{align*}
\]

- Ford–Fulkerson runs in \(\mathcal{O}(|E| \cdot |f^*|)\) and Edmonds–Karp in \(\mathcal{O}(|V| \cdot |E|^2)\) time.
The Max-Flow Min-Cut Theorem
The Max-Flow Min-Cut Theorem

\[ |f| = 23 \]
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 $G$. 

\[
|f| = 23
\]
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 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.
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 Karp and Somani is introduced. A preflow is like a flow, except that the preflow 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.
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 residual 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.
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 Karp and Kung is introduced. A preflow is like a flow, except that the \( v \rightarrow s \) 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.
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 preflow amount can be negative 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.
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
A **preflow** in $G$ is a real-value function $f : V \times V \rightarrow \mathbb{R}$ 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 $f : V \times V \rightarrow \mathbb{R}$ 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 $f : V \times V \rightarrow \mathbb{R}$ 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 $f : V \times V \rightarrow \mathbb{R}$ 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 $e(u) > 0$.

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

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

---

**Diagram:**

- Edge capacities: $u$ with $11/16$, $10/12$, and $2/4$.
- Excess flow: $e(u) = 3$. 

---
The Push Operation

\textbf{Push}(u, v)

\textbf{Condition:} \( u \) is overflowing, \( cf(u, v) > 0 \), and \( h(u) = h(v) + 1 \)

\textbf{Effect:} Push \( \min(e(u), cf(u, v)) \) overflow from \( u \) to \( v \)
The **Push** Operation

**Push**($u, v$)

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

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

---

**Example.**

\[
\begin{align*}
\text{e}(u) &= 5 & \text{e}(v) &= 1 \\
h(u) &= 4 & h(v) &= 3
\end{align*}
\]
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.**

![Diagram](image)

- $e(u) = 5$
- $e(v) = 1$
- $h(u) = 4$
- $h(v) = 3$
The **Push Operation**

**Push**($u, v$)

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

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

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

**Example.**

- $e(u) = 5$, $e(v) = 1$
- $h(u) = 4$, $h(v) = 3$
- $\Delta = 4$
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.**

$$e(u) = 5 \quad e(v) = 1$$

$$h(u) = 4 \quad h(v) = 3$$

$$\Delta = 4$$
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 \(v\)

\[
\Delta \leftarrow \min(e(u), c_f(u, v))
\]

\[
\text{if } (u, v) \in E \text{ then}
\[
\Delta \leftarrow \min(e(u), c_f(u, v))
\]
\[
\text{else}
\]
\[
f(v, u) \leftarrow f(v, u) - \Delta
\]

**Example.**

\[
\begin{align*}
e(u) &= 5 \\
h(u) &= 4
\end{align*}
\]

\[
\begin{align*}
e(v) &= 1 \\
h(v) &= 3
\end{align*}
\]

\[
\Delta = 4
\]

\[
\text{Push}(u, v)
\]

\[
\begin{align*}
2/6 & \quad /2 \\
\end{align*}
\]
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 \(v\)

\[
\Delta \leftarrow \min(e(u), c_f(u, v))
\]

\[
\text{if } (u, v) \in E \text{ then}
\]

\[
f(u, v) \leftarrow f(u, v) + \Delta
\]

\[
\text{else}
\]

\[
f(v, u) \leftarrow f(v, u) - \Delta
\]

**Example.**

\[
\text{Push}(u, v)
\]

\[
e(u) = 5 \quad e(v) = 1
\]

\[
h(u) = 4 \quad h(v) = 3
\]
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.**

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

- $e(u) = 5$
- $e(v) = 1$
- $h(u) = 4$
- $h(v) = 3$

$\Delta = 4$
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.**

\[
e(u) = 5 \quad e(v) = 1
\]

\[
h(u) = 4 \quad h(v) = 3
\]

\[
\Delta = 4
\]

\[
e(u) = 1 \quad e(v) = 5
\]

\[
h(u) = 4 \quad h(v) = 3
\]
The \textbf{RELABEL} Operation

\textbf{RELABEL}(u)

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

\textbf{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.**

![Diagram](image-url)
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.**

![Diagram of a graph before and after Relabel operation](image)
The Push-Relabel Algorithm

**Push-Relabel(G)**

**InitPreflow(G, s)**

while ∃ applicable Push or Relabel operation x do

  apply x
The **Push-Relabel Algorithm**

**Push-Relabel**\( (G) \)

**InitPreflow**\( (G, s) \)

\[
\text{while } \exists \text{ applicable Push or Relabel operation } x \text{ do }
\]

\[
\text{apply } x
\]

**InitPreflow**\( (G, s) \)

\[
\begin{align*}
\text{foreach } v \in V & \text{ do } h(v) \leftarrow 0; e(v) \leftarrow 0 \\
 h(s) & \leftarrow |V| \\
\text{foreach } (u, v) \in E & \text{ do } f(u, v) \leftarrow 0 \\
\text{foreach } v \text{ such that } (s, v) \in E & \text{ do } \\
& f(s, v) \leftarrow c(s, v) \\
& e(v) \leftarrow c(s, v)
\end{align*}
\]
The **Push-Relabel** Algorithm

**Push-Relabel**($G$)

**InitPreflow**($G, s$)

while ∃ 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

  $f(s, v) \leftarrow c(s, v)$

  $e(v) \leftarrow c(s, v)$

- initialises 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 Relabel.
- Find upper bound for the number of calls to Push.
Lemma 1. If a vertex $u$ is overflowing, either a push or a relabel operation applies to $u$.

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

**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 $(u, v) \in E$
  - $f(u, v) \leftarrow f(u, v) + \Delta$
- 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 $(u, v) \in E_f$

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

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

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 $(u, v) \in E_f$

\[
h(u) \leftarrow 1 + \min\{h(v): (u, v) \in E_f\}\]
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:
- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \ \forall (u, v) \in E_f$

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
\[ h(u) \leq h(v) \ \forall v \in V \text{ with } (u, v) \in E_f \]
\[ h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\} \]
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, $\text{RELABEL}(u)$ is applicable.

---

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

**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
  - $h(u) \leq h(v) \quad \forall v \in V$ with $(u, v) \in E_f$
- $h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$
Lemma 2.
The push-relabel algorithm maintains a preflow $f$.

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

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 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 $h(u) \leq h(v) \quad \forall v \in V$ 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** $(u, v) \in E$
  - $f(u, v) \leftarrow f(u, v) + \Delta$
- **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 $h(u) \leq h(v) \quad \forall v \in V$ with $(u, v) \in E_f$
- $h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$
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
$h(u) \leq h(v) \quad \forall v \in V$ 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** initialised $h$ as a height function. ✓
- Under **Push**($u, v$), $h$ remains a height function:

  **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 $(u, v) \in E_f$

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

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

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

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

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

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

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) ∈ $E_f$, then $h(u) \leq h(v) + 1$
  - (w, u) ∈ $E_f$, then $h(w) < h(u) + 1$

Conditions:

- **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) ∈ $E$ 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 $h(u) \leq h(v)$ ∀v ∈ V with (u, v) ∈ $E_f$
  - $h(u) \leftarrow 1 + \min\{h(v) : (u, v) \in E_f\}$

Height function:

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

**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 $(u, v) \in E$
  - $f(u, v) \leftarrow f(u, v) + \Delta$
- 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:**
- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \ \forall (u, v) \in E_f$
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 $G_f$.

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

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

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

Height function:
- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \quad \forall (u, v) \in E_f$
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 $0 \leq i \leq k - 1$, and
- $h(v_i) \leq h(v_{i+1}) + 1$ for $0 \leq i \leq k - 1$.

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

Height function:
- $h(s) = |V|$
- $h(t) = 0$
- $h(u) \leq h(v) + 1 \ \forall (u, v) \in E_f$
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 $0 \leq i \leq k - 1$, and
- $h(v_i) \leq h(v_{i+1}) + 1$ for $0 \leq i \leq k - 1$.

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

But since $k < |V|$, it follows that $h(s) < |V|$. $\times$
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.
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.
  \[ \Rightarrow f \text{ is a flow.} \]
- By Lemma 3, \( h \) is a height function.
- So by Lemma 4, there is no \( s \rightarrow t \) 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.
  \[ \Rightarrow f \text{ is a flow.} \]
- By Lemma 3, \( h \) is a height function.
- So by Lemma 4, there is no \( s \rightarrow t \) path in \( G_f \).
  \[ \Rightarrow \text{By the Max-Flow Min-Cut Theorem, the flow } f \text{ is a maximum flow.} \]
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 Relabel.
- Find upper bound for the number of calls to 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 \( G_f \).
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 $s \notin S_v$. 

\[
ed(v) > 0
\]
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 $\sum_{w \in S_v} e(w) \geq 0$. 

\[ e(v) > 0 \]
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 $\sum_{w \in S_v} e(w) \geq 0$.
- Since $v \in S_v$, we even have $\sum_{w \in S_v} e(w) > 0$. 
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 $v$. $\times$

![Diagram](image-url)
Upper Bound on the Height

Lemma 7.
During the push–relable 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
- $h(u) \leq h(v) \quad \forall v \in V$ 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–relable 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.
Lemma 7. 
During the push–relable 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 $(u, v) \in E_f$

$h(u) \leftarrow 1 + \min \{ h(v) : (u, v) \in E_f \}$
Lemma 7.
During the push–relable 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 $h(u) \leq h(v) \quad \forall v \in V$ 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–relable 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**

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

![Diagram](image)
Upper Bounds on the Height and \# \textsc{Relabel} Operations

**Lemma 7.**
During the push–relable 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-relable algorithm executes at most $2|V|^2$ \textsc{Relabel} operations.
Saturating and Unsaturating \textbf{Push} Operations

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

\begin{itemize}
  \item \textbf{saturating} if afterwards \( c_f(u, v) = 0 \),
\end{itemize}
Saturating and Unsaturating \textbf{PUSH} Operations

The operation $\text{PUSH}(u, v)$ is

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

\[ 2/6 \]

\[ u \rightarrow v \]
Saturating and Unsaturating Push Operations

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

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

\[
\begin{array}{c}
u \\
\xrightarrow{2/6} \\
v
\end{array}
\quad \text{Push}(u, v)
\]

\[
\Delta = 4
\]
Saturating and Unsaturating \textbf{PUSH} Operations

The operation \textbf{PUSH}(u, v) is

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

\[
\begin{align*}
&\overset{2/6}{\quad u \quad} \xrightarrow{\text{PUSH}(u,v)} \overset{6/6}{\quad v \quad} \\
&\Delta = 4
\end{align*}
\]
Saturating and Unsaturating Push Operations

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

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

\[
\begin{align*}
& u \quad \text{2/6} \quad \text{Push}(u, v) \quad \Delta = 4 \\
& \Rightarrow \quad \rightarrow \quad v \quad \text{6/6}
\end{align*}
\]

- **unsaturating** otherwise.
Saturating and Unsaturating \textbf{Push} Operations

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

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

\[ \Delta = 4 \]

- and **unsaturating** otherwise.
Saturating and Unsaturating Push Operations

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

- **saturating** if afterwards $c_f(u, v) = 0,$
  
  \[ \Delta = 4 \]

- and **unsaturating** otherwise.
  
  \[ \Delta = 2 \]
Saturating and Unsaturating Push Operations

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

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

  \[
  \begin{array}{c}
  \text{Push}(u, v) \\
  \Delta = 4
  \end{array}
  \]

- and **unsaturating** otherwise.

  \[
  \begin{array}{c}
  \text{Push}(u, v) \\
  \Delta = 2
  \end{array}
  \]
Lemma 9.
The push-relable algorithm executes at most $2|V| \cdot |E|$ saturating \texttt{Push} operations.

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

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

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

\textbf{if} $(u,v) \in E$ \textbf{then}

\[ f(u,v) \leftarrow f(u,v) + \Delta \]

\textbf{else}

\[ f(v,u) \leftarrow f(v,u) + \Delta \]

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

\(e(v) \leftarrow e(v) + \Delta\)
Lemma 9.
The push-relable algorithm executes at most $2|V| \cdot |E|$ saturating Push operations.

Proof.
- Consider saturating $\text{Push}(u, v)$
  - $h(u) = h(v) + 1$

\[\text{Push}(u, v)\]
\begin{algorithmic}
  \STATE \textbf{Condition:} $u$ is overflowing, $c_f(u, v) > 0$, and $h(u) = h(v) + 1$
  \STATE $\Delta \leftarrow \min(e(u), c_f(u, v))$
  \IF{$(u, v) \in E$}
    \STATE $f(u, v) \leftarrow f(u, v) + \Delta$
  \ELSE
    \STATE $f(v, u) \leftarrow f(v, u) + \Delta$
  \ENDIF
  \STATE $e(u) \leftarrow e(u) - \Delta$
  \STATE $e(v) \leftarrow e(v) + \Delta$
\end{algorithmic}
Lemma 9.
The push-relable algorithm executes at most $2|V| \cdot |E|$ saturating Push operations.

Proof.
- Consider saturating $\text{Push}(u, v)$
  - $h(u) = h(v) + 1$

- For another saturating $\text{Push}(u, v)$, first $\text{Push}(v, u)$ necessary
  - $h(v) = h(u) + 1$ necessary

Push($u, v$)

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

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

if $(u, v) \in E$

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

else

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

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

$e(v) \leftarrow e(v) + \Delta$
Lemma 9. The push-relable algorithm executes at most $2|V| \cdot |E|$ saturating \texttt{Push} operations.

Proof.

- Consider saturating \texttt{Push}(u, v)
  - $h(u) = h(v) + 1$

- For another saturating \texttt{Push}(u, v), first \texttt{Push}(v, u) necessary
  - $h(v) = h(u) + 1$ necessary

- After another saturating \texttt{Push}(u, v), both $h(u)$ and $h(v)$ have increased by at least two.
Lemma 9.
The push-relable algorithm executes at most $2|V| \cdot |E|$ saturating Push operations.

Proof.

- Consider saturating $\text{Push}(u, v)$
  - $h(u) = h(v) + 1$

- For another saturating $\text{Push}(u, v)$, first $\text{Push}(v, u)$ necessary
  - $h(v) = h(u) + 1$ necessary

- After another saturating $\text{Push}(u, v)$, both $h(u)$ and $h(v)$ have increased by at least two.

- But by Lemma 6, $h(u) \leq 2|V| - 1$ and $h(v) \leq 2|V| - 1$. 

\begin{align*}
\text{Push}(u, v) \\
\text{Condition: } u \text{ is overflowing, } c_f(u, v) > 0, \text{ and } h(u) = h(v) + 1 \\
\Delta \leftarrow \min(e(u), c_f(u, v)) \\
\text{if } (u, v) \in E \text{ then} \\
& f(u, v) \leftarrow f(u, v) + \Delta \\
\text{else} \\
& f(v, u) \leftarrow f(v, u) + \Delta \\
e(u) \leftarrow e(u) - \Delta \\
e(v) \leftarrow e(v) + \Delta
\end{align*}
Lemma 9.
The push-relable algorithm executes at most $2|V| \cdot |E|$ saturating \texttt{Push} operations.

Proof.

- Consider saturating \texttt{Push}(u, v)
  - $h(u) = h(v) + 1$
  - For another saturating \texttt{Push}(u, v), first \texttt{Push}(v, u) necessary
    - $h(v) = h(u) + 1$ necessary
  - After another saturating \texttt{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 \texttt{Push} operations for edge $(u, v)$. 

\texttt{Push}(u, v)

\textbf{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 $(u, v) \in E$

\[
\begin{align*}
f(u, v) &\leftarrow f(u, v) + \Delta \\
f(v, u) &\leftarrow f(v, u) + \Delta
\end{align*}
\]

else

\[
\begin{align*}
e(u) &\leftarrow e(u) - \Delta \\
e(v) &\leftarrow e(v) + \Delta
\end{align*}
\]
Lemma 10.
The push–relable 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

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

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

$e(v) \leftarrow e(v) + \Delta$
Lemma 10.
The push–relable algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating Push ops.

Proof.
- Consider $\mathcal{H} = \sum_{v \in V \setminus \{s,t\}, \text{v overflowing}} h(v)$.
- After initialisation 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 \textbf{Push} Ops

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

\textbf{Proof.}

- Consider $\mathcal{H} = \sum_{v \in V \setminus \{s,t\}, v \text{ overflowing}} h(v)$.

- After initialisation and at the end $\mathcal{H} = 0$.

- A saturating \textbf{Push} increases $\mathcal{H}$ by at most $2|V| - 1$.

- By Lemma 9, all saturating \textbf{Push} operations increase $\mathcal{H}$ by at most $(2|V| - 1) \cdot 2|V| \cdot |E|$.

\textbf{Push}(u, v)

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

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

\textbf{if} $(u, v) \in E$ \textbf{then}

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

\textbf{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–relable algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating Push ops.

Proof.

- Consider $H = \sum_{v \in V \setminus \{s,t\}, v \text{ overflowing}} h(v)$.

- After initialisation and at the end $H = 0$.

- A saturating Push increases $H$ by at most $2|V| - 1$.

- By Lemma 9, all saturating Push operations increase $H$ by at most $(2|V| - 1) \cdot 2|V| \cdot |E|$.

- By Lemma 7, all Relabel operations increase $H$ by at most $(2|V| - 1) \cdot |V|$. 

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

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

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

$e(v) \leftarrow e(v) + \Delta$
Lemma 10.
The push–relable algorithm executes at most $4|V|^2 \cdot |E|$ unsaturating PUSH ops.

Proof.

- Consider $\mathcal{H} = \sum_{v \in V \setminus \{s,t\}, \text{v overflowing}} h(v)$.
- After initialisation 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 $h(u) - h(v) \geq 1$.

Push($u, v$)
Condition: $u$ is overflowing, $cf(u, v) > 0$, and $h(u) = h(v) + 1$
$\Delta \leftarrow \min(e(u), cf(u, v))$
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
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 $O(|V|^2|E|)$ valid Push or 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 **Push** select *highest* overflowing vertex: \( \mathcal{O}(|V|^2|E|^\frac{1}{2}) \)

- **Excess scaling:**
  For **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] ← 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