Vortrag über Max-Flow-Algorithmen

Edmonds-Karp, Push-Relabel

Moritz Rehbach, Florian Nelles

Institut für Informatik
Universität Bonn

12. Mai 2014

Inhaltsverzeichnis

Grundbegriffe

Flussnetzwerk

- gerichteter Graph $G = (V, E)$
- 2 ausgezeichnete Knoten $s, t \in V$ (Quelle s, Senke t)
- Kapazitätsfunktion $c : E \to \mathbb{N}$

Fluss in einem Flussnetzwerk

Funktion $f : E \to \mathbb{R}_+$ mit Kapazitätsbeschränkung:

$$0 \leq f(e) \leq c(e) \qquad \forall e \in E(V)$$

Grundbegriffe: Überschuss, s-t-Fluss, s-t-Präfluss

Überschussfunktion

Der **Überschuss** (engl. **excess**) eines Flusses f in $v \in V(G)$ ist

$$ex_f(v) := \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e)$$

s-t-Fluss

- $f(e) \leq u(e) \qquad \forall e \in E(G)$ (Kapazitätsbeschränkung)
- $ex_f(s) \leq 0$
- $ex_f(v) = 0 \qquad \forall v \in V \backslash \{s, t\}$ (Flusserhaltung)

s-t-Präfluss

- $f(e) \leq u(e) \qquad \forall e \in E$
- $ex_f(v) \geq 0 \qquad \forall v \in V \backslash \{s\}$

Grundbegriffe: Wert eines Flusses, maximaler Fluss

Wert eines Flusses

- $|f| := -ex_f(s)$

Grundbegriffe: Maximaler Fluss

f ist genau dann maximal, wenn die folgenden äquivalenten Bedingungen erfüllt sind:

- $ex_f(v) = 0 \qquad \forall v \in V \setminus \{s, t\}$
- Es gibt keinen f-augmentierenden Pfad mehr.

Warum dies so ist, werden wir bei der Betrachtung der Algorithmen genauer untersuchen.

Problembeschreibung

Algorithmen

Wir wollen zu einem gegebenen Flussnetzwerk einen maximalen Fluss f_{max} finden.
Die von uns betrachteten Algorithmen sind:

- Ford-Fulkerson-Algorithmus
- Edmonds-Karps-Algorithmus
- Push-Relabel-Algorithmus

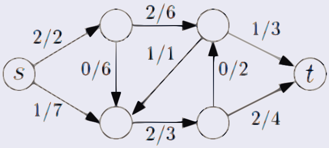
Edmonds-Karps ist eine Verbesserung von Fold-Fulkerson, beide sind Greedy-Algorithmen.
Die beste Laufzeit hat der Push-Relabel-Algorithmus.

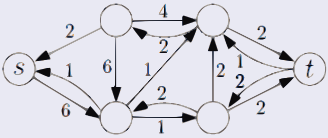
Restnetzwerk

Sei $G = (V, E)$ ein Flussnetzwerk mit Kapazitäten $c : E \rightarrow \mathbb{N}$ und sei f ein Fluss in G. Betrachte

$$rest_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{, falls } (u,v) \in E \\ f(v,u) & \text{, falls } (v,u) \in E \\ 0 & \text{, sonst} \end{cases}$$

Dann ist das dazugehörige Restnetzwerk $G_f = (V, E_f)$ mit
$E_f = \{(u,v) \in VxV | rest_f(u,v) > 0\}$
und Kantengewichten $(u,v) \in E_f \rightarrow rest_f(e) > 0$.
Das Restnetzwerk G_f hat also die **gleiche Knotenmenge** wie das Flussnetzwerk, aber andere Kanten und Gewichte.



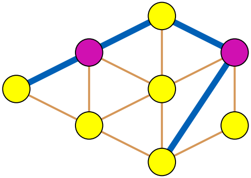


Das Beispiel stammt aus dem Skript zur Vorlesung *Algorithmen und Berechnungskomplexität I (Röglin)*

Ford-Fulkerson

Flussverbesserung

Wir wollen, ausgehend von einem (gültigen) Fluss $f : E \to \mathbb{R}$ zu einem verbesserten Fluss mit größerem Wert $|f|$ gelangen.
Wir suchen dazu eine geeignte Kantenmenge, deren Flusswerte wir dann erhöhen.



Ford-Fulkerson

Augmentierender Pfad

Ein geeigneter Pfad P ist jeder **einfache** Weg im Restnetzwerk, der s mit t verbindet.

Erhöhung entlang eines Pfades

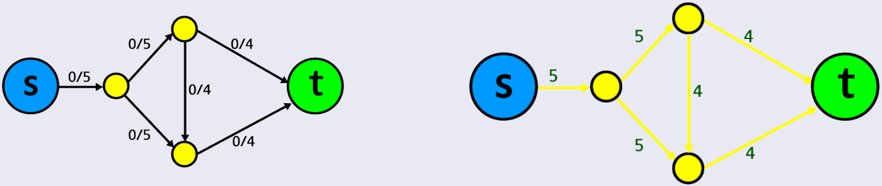
Für einen Fluss f und einen Pfad P in G_f definieren wir den entlang P erhöhten Pfad durch:

$$(f \uparrow P)(u,v) = \begin{cases} f(u,v) + \delta & \text{, falls } (u,v) \in E \text{ und } (u,v) \in P \\ f(u,v) - \delta & \text{, falls } (u,v) \in E \text{ und } (v,u) \in P \\ f(u,v) & \text{, sonst} \end{cases}$$

mit $\delta := min_{e \in P}(E_f)$.
E bezeichnet die Flusswerte in f und E_f die Kantengewichte im Restnetzwerk G_f.

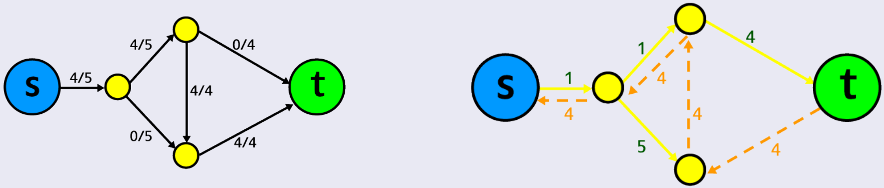
Ford-Fulkerson

Beispiel: Erhöhen entlang eines Pfades



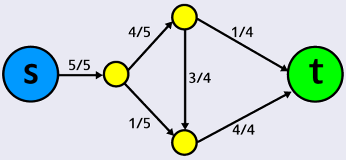
Links das Flussnetzwerk mit Kapazitäten, rechts das Restnetzwerk

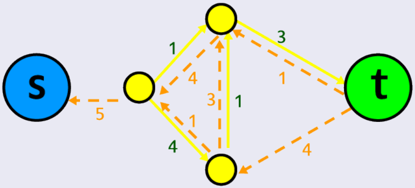
Beispiel: Erhöhen entlang eines Pfades



Es wurde entlang eines Pfades erhöht. Die orangen Kanten im Flussnetzwerk sind die „Rückkanten".

Beispiel: Erhöhen entlang eines Pfades





Hier musste beim Erhöhen der Fluss über die senkrecht gezeichnete Kante gesenkt werden.
Wie man sieht, ist kein flussverbessernder Pfad mehr im Restnetz enthalten, und der Fluss ist maximal.

Ford-Fulkerson

Der Algorithmus

Wir initialisieren mit einem Null-Fluss:
$f(v,u) := 0 \forall v,u \in V$ und erhöhen dann den Fluss entlang beliebiger augmentierender Pfade, bis kein solcher mehr vorhanden ist.

```
FordFulkerson(G, c, s, t) {
        f := nullFluss(G)
        while ∃ augmentierenderPfad(G,c,f) P do:
                f = erhöheEntlangP(f)
        return f
}
```

Ford-Fulkerson

Korrektheit

Wir wollen zeigen:

- Erhöhen entlang eines augmentierenden Pfades führt immer zu gültigem Fluss
- Wenn kein augmentierender Pfad mehr existiert, ist der Fluss maximal

Dass der Algorithmus immer terminiert, besprechen wir in der Laufzeitabschätzung.

Ford-Fulkerson

$(f \uparrow P)$ ist zulässiger Fluss

Um dies zu zeigen, werden wir die eben definierte Fallunterscheidung Schritt für Schritt durchgehen, und zeigen, dass die Flusseigenschaften jeweils erhalten bleiben.

Ansatz

Wir betrachten einen beliebigen Knoten $u \in V \backslash \{s, t\}$.
Wird der Knoten u von P **nicht** besucht, so wird keiner der Flusswerte $(f \uparrow P)(u, v)$ oder $(f \uparrow P)(v, u)$ modifiziert.
Wir brauchen also nur die Kanten zu betrachten, die min. einen Knoten mit einer Kante aus P gemeinsam haben.

Ford-Fulkerson

$(f \uparrow P)$ ist zulässiger Fluss - u liegt auf P

Wenn der Knoten u auf dem Pfad P liegt, so enthält P genau zwei Kanten (v, u) und (u, v') mit $v, v' \in V$.
Grund: der Pfad P ist ein **einfacher Weg** von s nach t, und u selbst ist nicht die Quelle oder Senke.

Erinnerung: Doppelte Kanten

Im Restnetzwerk kommen i.A. doppelte Kanten zwischen zwei Knoten vor. Für das Flussnetzwerk selbst macht das keinen Sinn - symmetrische Kanten können durch eine einfache Kante mit der Differenz der Kapazitäten ersetzt werden.
Damit gilt stets (exklusives *oder*):

- $(v, u) \in E$ oder $(u, v) \in E$ sowie
- $(v', u) \in E$ oder $(u, v') \in E$

Ford-Fulkerson

$(f \uparrow P)$ ist zulässiger Fluss

Man kann sich vereinfacht vorstellen, dass der Knoten v' auf P von u in Richtung Senke führt, der Knoten v von u Richtung Quelle.

u liegt auf P: Die vier Fälle

Wenn man die Flussbilanz in u nach Durchführung der Erhöhung betrachtet, sieht man, dass Flusserhaltung unter f auch Flusserhaltung in u für $(f \uparrow P)$ impliziert:

$$G_f \qquad v \xrightarrow[\in E_f]{} u \xrightarrow[\in E_f]{} v'$$

$$v \xrightarrow[\in E]{+\delta} u \xrightarrow[\in E]{+\delta} v' \qquad\qquad v \xrightarrow[\in E]{+\delta} u \xleftarrow[\in E]{-\delta} v'$$

$$v \xleftarrow[\in E]{-\delta} u \xrightarrow[\in E]{+\delta} v' \qquad\qquad v \xleftarrow[\in E]{-\delta} u \xleftarrow[\in E]{-\delta} v'$$

$(f \uparrow P)$ ist zulässiger Fluss: Kapazitätsbeschränkung

Wir wählen eine beliebige Kante aus (u, v) aus P.
Für $(u, v) \in P$ gilt:

1. Falls $e_{inc} = (u, v) \in E$, ist $(f \uparrow P)(u, v) \leq c(e_{inc})$, denn $0 \leq f(e_{inc}) + \delta \leq f(e_{inc}) + rest_f(e_{inc}) =$ $f(e_{inc}) + (c(e_{inc}) - f(e_{inc})) = c(e_{inc})$
2. Falls $e_{dec} = (v, u) \in E$, ist $(f \uparrow P)(v, u) \leq c(e_{dec})$, denn $c(e_{dec}) \geq f(e_{dec}) \rightarrow c(e_{dec}) \geq f(e_{dec}) - \delta$.
   f(e_{dec})$-\delta \geq 0$ folgt aus der Definition von δ.

Wert nach Erhöhung

Nach Definition ergibt sich damit $|(f \uparrow P)| = |f| + \delta$.

Ford-Fulkerson: Terminierung

Maximaler Fluss nach Terminierung

Wir zeigen jetzt: Wenn G_f keinen augmentierenden Pfad enthält, ist f ein maximaler Fluss.
Dabei hilft folgende Definition:

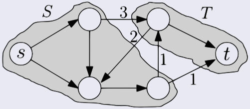
Schnitt in Flussnetzwerken

Sei $G = (V, E)$ mit Quelle s, Senke t und Kapazitäten $c : E \to \mathbb{N}$ ein Flussnetz.
Ein **Schnitt von G** ist eine Partitionierung der Knotenmenge in zwei Teile $S \subseteq V$ und $T = V \backslash S$, wobei S die Quelle und T die Senke enthalten muss.

Ford-Fulkerson: Schnitte in Flussnetzwerken

Kapazität eines Schnitts

Zu einem Schnitt (S, T) definieren wir
$c(S, T) := \sum_{u \in S} \sum_{v \in T} c(u, v)$
als **Kapazizät des Schnitts**.



Das Bild stammt aus dem Skript zur Vorlesung *Algorithmen und Berechnungskomplexität I (Röglin)*

Fluss über einen Schnitt

Für einen Fluss f und einen Schnitt (S, T) definieren wir den **Fluss über diesen Schnitt**:

$$f(S, T) := \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$

Ford-Fulkerson: Terminierung

Minimaler Schnitt

Ein Schnitt heißt **minimal** (S, T) wenn $\forall(S', T') : c(S', T') > c(S, T)$ gilt.

Schnitt und Flusswert

Intuitiv ist klar: der Fluss muss von S nach T passieren, um die Senke zu erreichen, somit kann $|f|$ nicht größer sein als $f(S, T)$.
Wegen Flusserhaltung und der Definition des Schnitts gilt sogar für beliebige Flüsse und Schnitte:
$|f| = f(S, T) \leq c(S, T)$
Der Beweis findet sich unter Lemma 4.5.7 im Skript zur Vorlesung *Algorithmen und Berechnungskomplexität I (Röglin)*

Ford-Fulkerson: Terminierung

Maximaler Fluss und Schnitte

Es gilt, wie wir jetzt (knapp) zeigen werden, folgende Äquivalenz:

1. f ist maximal
2. $\iff \nexists$ augmentierender Pfad $P \subseteq E_f$
3. $\iff \exists (S, T) : |f| = c(S, T) = f(S, T)$

Weil das Erhöhen entlang eines Pfades immer zu einer Verbesserung des Flusswertes führt (Greedy-Ansatz), terminiert die wiederholte Anwendung mit einem maximalen Fluss.

(1) $\to$ (2)

Angenommen, f ist ein maximaler Fluss (1). Würde (2) **nicht** gelten, könnten wir entlang P erhöhen und somit den Wert von f um δ erhöhen. Dann ist f aber nicht maximal (Widerspruch).

Ford-Fulkerson: Terminierung

(3) $\rightarrow$ (2)

Sei f ein Fluss und (S, T) der Schnitt mit der Kapazität $c(S, T) = f$ (soweit Aussage 3).
Sei f_{max} der maximale Fluss im betrachteten Flussnetzwerk. Dann gilt $|f| \leq |f_{max}|$.
Wenden wir die beiden letzten Beobachtungen an:

- für **jede** Kombination Schnitt+Fluss gilt: $f(S, T) = |f|$
- $f(S, T) \leq c(S, T)$

$\rightarrow |f_{max}| = f(S, T) \leq c(S, T) \leftrightarrow |f| = |f_{max}|$

Ford-Fulkerson: Terminierung

$(2) \rightarrow (3)$

(2) $\nexists$ augmentierender Pfad $P \subseteq E_f$
Wenn es keinen augmentierenden Pfad gibt, dann gibt es nach Definition keinen Weg in G_f von s nach t.
Definieren wir einen Schnitt (S, T) wie folgt:
$S := \{v \in V | \exists \text{ Weg von } v \text{ nach } t \text{ in } G_f\}$
$T := V \backslash S$.

Weil (2) gilt, liegt s in S und t in T

Ford-Fulkerson: Terminierung

(2) $\rightarrow$ (3)(*ff.*)

- Sei $(u, v) \in E$ mit $u \in S$ und $v \in T$. Dann gilt $(u, v) \notin E_f$, weil sonst ein Weg in G_f von s nach t entstanden wäre, in Widerspruch zur Annahme. Damit gilt $rest_f(u, v) = 0$ und f(u,v)=c(u,v)
- Sei $(v, u) \in E$ mit $u \in S$ und $v \in T$. Dann gilt $(u, v) \notin E_f$, weil sonst ein Weg in G_f von s nach t entstanden wäre, in Widerspruch zur Annahme. Damit gilt $rest_f(u, v) = 0$ und f(v,u)=0

Es ergibt sich $c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v) =$
$\sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) = f(S, T)$
$= c(S, T)$
Somit ist (S, T) der Schnitt, dessen Existenz in (3) behauptet wurde.

Fold-Fulkerson: Laufzeit

Terminierung

Wir haben bislang noch nicht gezeigt, dass der Algorithmus überhaupt immer terminiert.
Für einen ausführlichen Beweis sei wieder auf das Röglin-Skript verwiesen.
Wir beschränken uns auf eine Kurzzussammenfassung:

- Für ganzzahlige Kapazitäten ist der maximale Flusswert und der Wert von f in jeder Iteration des Algorithmus ebenfalls ganzzahlig
- Jede Iteration des Algorithmus führt zu einer Erhöhung des Flusswertes.

Daraus ergibt sich eine obere Schranke $\sum_{e \in E} c(e)$ für die Anzahl der Iterationen.

Laufzeit

In jedem Schleifendurchgang muss G_f auf einen augmentierenden Pfad untersucht werden.
Mit einfacher Graphensuche (Breitensuche, iterative Tiefensuche) können

Edmonds-Karps

Verbesserung des Ford-Fulkerson-Algorithmus

Dass die Laufzeit von den Kapazitäten abhängt, ist wenig effizient. Multipliziert man z.B. alle Kapazitäten mit 100, würde man nicht von einer größeren Problemkomplexität ausgehen.
Eine naheliegende Modifikation führt zu einer verbesserten Laufzeit, die nur noch von G abhängt.
Das Vorgehen bleibt fast gleich, wir haben weiterhin in jedem Durchlauf einen erlaubten Fluss, wählen aber nicht mehr einen *beliebigen* augmentierenden Pfad.

Edmonds-Karps: Erhöhen

Erhöhen entlang eines Pfades

Im Ford-Fulkerson-Algorithmus haben konnten wir in jeder Iteration einen beliebigen Pfad in G_f von s nach t auswählen.
Jetzt werden wir stattdessen **kurze** Pfade bevorzugen.

Finden des kürzesten Pfades

Einen möglichst kurzen Weg von s nach t in G_f findet man mit Breitensuche ($O(|E|)$). Dadurch entsteht kaum zusätzlicher Zeitaufwand verglichen mit dem Finden eines beliebigen Pfades.

Algorithmus

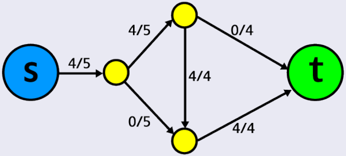
```
EdmondsKarps(G, c, s, t) {
        f := nullFluss(G)
        while true do:
                P = sucheKürzestenWeg(Restnetzwerk(G,f),s,t)
                if P = ∅ break
                f = erhöheEntlangP(f)
        return f
}
```

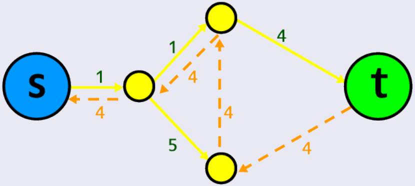
Laufzeit

Die Eigenschaft, dass der Flusswert in jeder Iteration verbessert wird, bleibt hier weiter gültig.
Durch die Wahl des kürzesten Weges von s nach t in G_f für die Erhöhung wird die Anzahl der Iterationen minimal gehalten.

Flaschenhalskanten

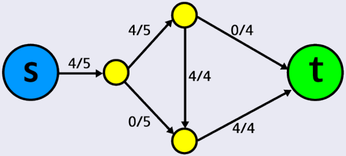
Wir sehen uns nochmal das Beispiel zu Ford-Fulkerson an:

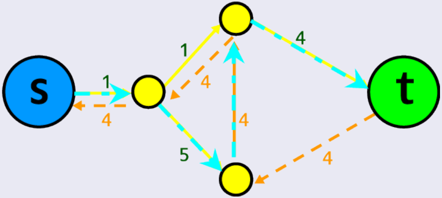




Edmonds-Karps: Laufzeit

Flaschenhalskanten



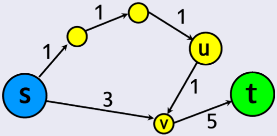


Hier wurde zuerst entlang eines Pfades mit 4 Kanten augmentiert, obwohl es auch einen Pfad mit 3 Kanten gegeben hätte. Die senkrecht gezeichnete Kante bildet einen Flaschenhals, deshalb muss im letzten Schritt der Fluss hier gesenkt werden.
Der Algorithmus hat eine Kante aus G_f entfernt und später wieder eingefügt.

Edmonds-Karps: Laufzeit

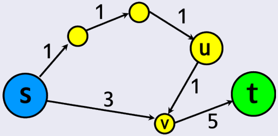
Distanzen

Dieser Fall tritt ein, wenn eine verbessender Pfad eine Kante in G_f enthält, die erst durch Flusserhöhung enstanden ist.
Wir nennen die minimale Anzahl Kanten zwischen zwei Knoten u, v in einem Graphen **Distanz**.
Beim Edmonds-Karps-Algorithmus kann die Distanz von s zu einem beliebigem Knoten u in G_f nicht sinken
Würde das passieren, müsste die Anzahl der Kanten zwischen s und u durch eine Abkürzung über eine neue Rückkante reduziert werden.
Betrachten wir dieses Flussnetzwerk:



Edmonds-Karps: Laufzeit

Distanz von s zu beliebigem Knoten u in G_f kann nicht sinken



Sei $u \in V \setminus \{s, t\}$ und m die Distanz von s zu u im Restnetzwerk des vorherigen Iterationsschritts. Angenommen, u ist durch eine neue Rückkante $(v, u) \in E_f$ über n Kanten von s zu erreichen, und $n < m$. Weil (v, u) neu in G_f ist, muss gerade entlang (u, v) erhöht worden sein. u lag also vor der Erhöhung aus einem flussverbessernden Pfad P_1, der mindestens die Länge $m + 1$ hatte.
Wäre $n \leq m$, hätte aber Pfad P_1 verkürzt werden können, indem der Weg von s über u nach v (m Kanten) durch den nur $n - 1$ Kanten langen Weg ersetzt würde. Dieser Pfad P_2 wäre also schon vorher der kürzere augementierende Pfad in G_f gewesen.

Laufzeit ist nur noch vom Graphen abhängig

Die Gesamtlaufzeit liegt durch diese Verbesserung in $O(|E|^2 * |V|)$.

Push-Relabel-Algorithmus

Wir wissen:

Ein s-t-Fluss f ist genau dann maximal, wenn gilt:

- $ex_f(v) = 0 \qquad \forall v \in V \setminus \{s, t\}$
- Es gibt keinen f-augmentierenden Pfad mehr.

Edmonds-Karps: 1. Bedingung immer erfüllt, streben nach 2. Bedingung.
Push-Relabel: 2. Bedingung immer erfüllt, streben nach 1. Bedingung.

Distanzmarkierung

Definition

Sei (G, u, s, t) ein Netzwerk und f ein s-t-Präfluss. Eine **Distanzmarkierung** ist eine Funktion $\Psi : V(G) \rightarrow (Z_+)$ mit

- $\Psi(t) = 0$
- $\Psi(s) = n = |V(G)|$
- $\Psi(v) \leq \Psi(w) + 1$ für alle $(v, w) \in E(G_f)$

Eine Kante e heißt **erlaubte Kante**, falls $e \in E(G_f)$ und $\Psi(v) = \Psi(w) + 1$

aktiver Knoten

Ein $v \in V(G) \backslash \{s, t\}$ heißt **aktiver** Knoten, wenn

$$ex_f(v) > 0$$

Push-Relabel-Algorithmus

1. *Setze* $f(e) = u(e) \qquad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \qquad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \qquad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

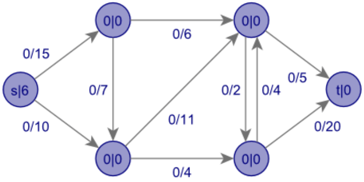
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \qquad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \qquad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \qquad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

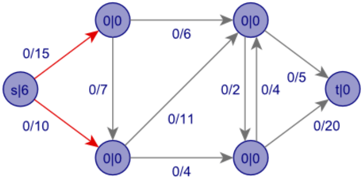
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

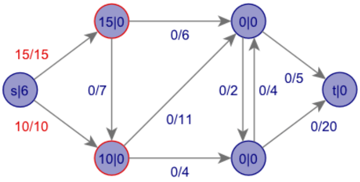
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. *Setze* $f(e) = u(e)$ $\quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0$ $\quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0$ $\quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

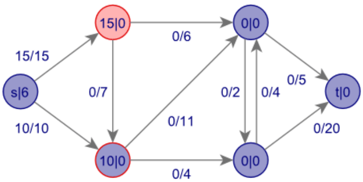
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. *Setze* $f(e) = u(e) \qquad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \qquad \forall e \in E \setminus \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \qquad \forall v \in V \setminus \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

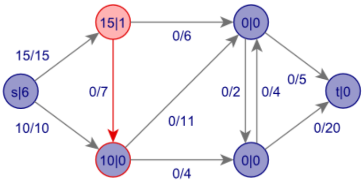
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

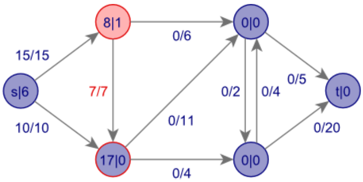
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

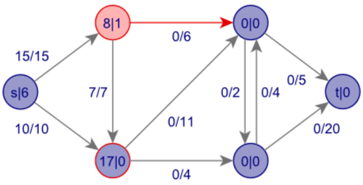
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \setminus \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \setminus \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

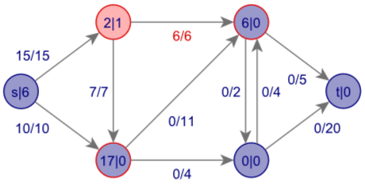
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

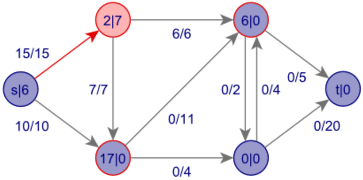
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then RELABEL*(v)
8. *else PUSH*(e) mit $e \in \delta^+_{G_f}(v)$ erlaubt

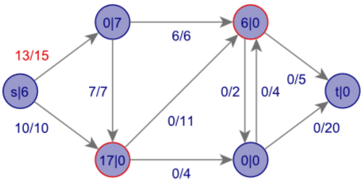
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

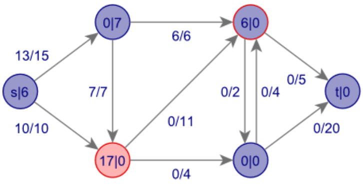
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

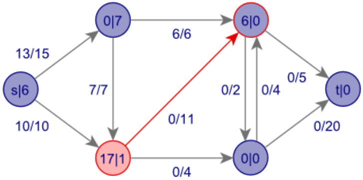
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

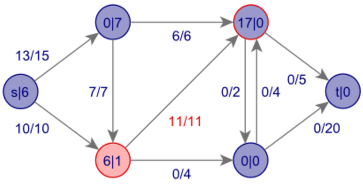
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

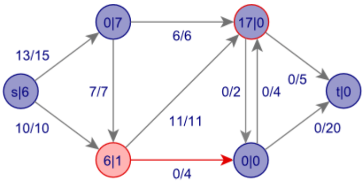
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

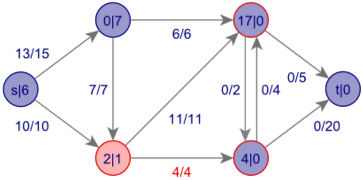
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

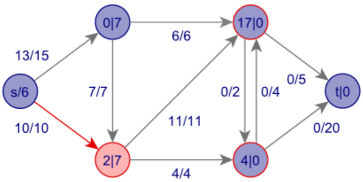
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

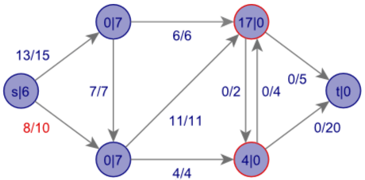
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

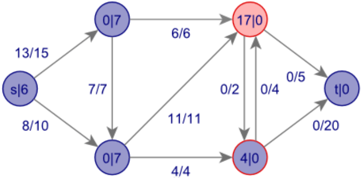
RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:

Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

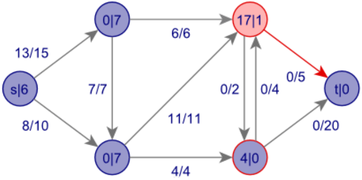
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

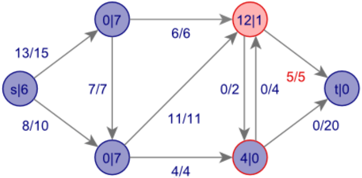
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

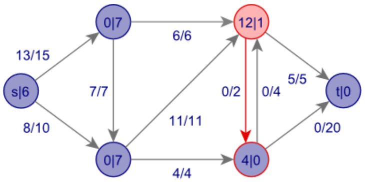
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \setminus \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \setminus \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

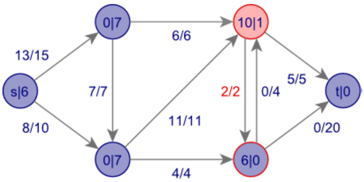
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \setminus \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \setminus \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then RELABEL*(v)
8. *else PUSH*(e) mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

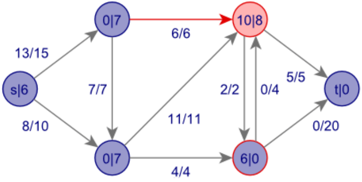
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \qquad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \qquad \forall e \in E \setminus \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \qquad \forall v \in V \setminus \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

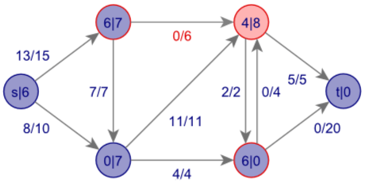
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. Setze $f(e) = u(e)$ $\quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0$ $\quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0$ $\quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

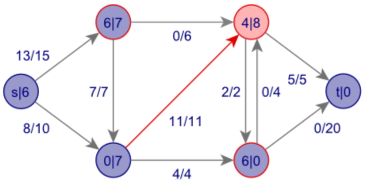
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

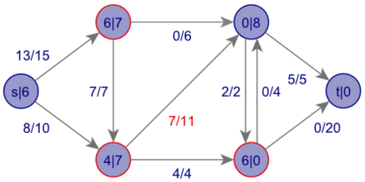
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

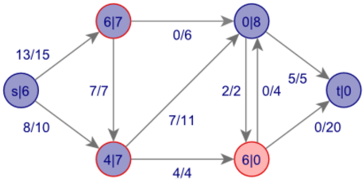
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

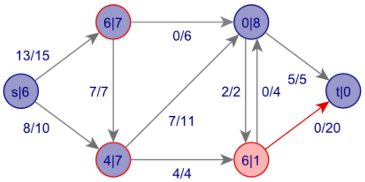
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

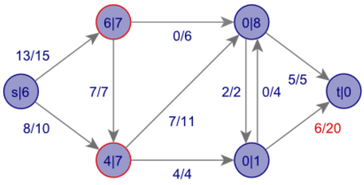
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

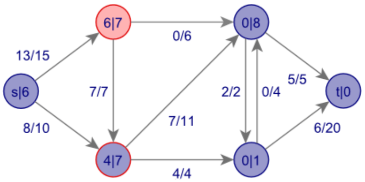
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

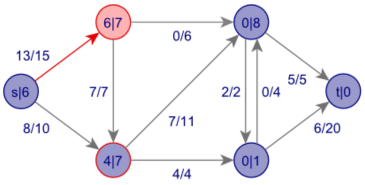
PUSH(e)

1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

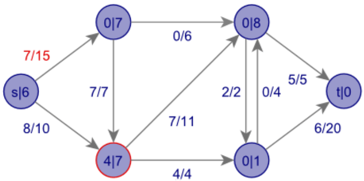
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

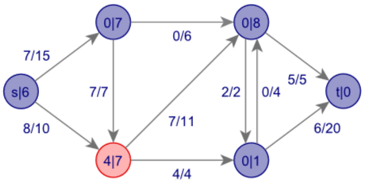
PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

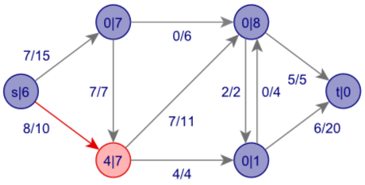
1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if* (kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

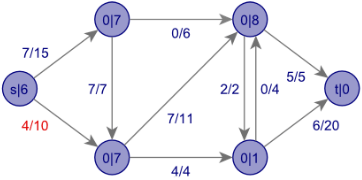
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. Setze $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. Setze $f(e) = 0 \quad \forall e \in E \backslash \delta^+(s)$
3. Setze $\Psi(s) = n = |V|$
4. Setze $\Psi(v) = 0 \quad \forall v \in V \backslash \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

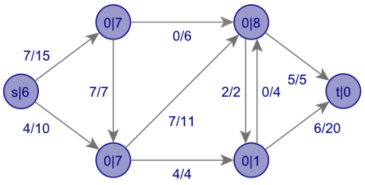
1. Setze $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. Setze $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Beispiel Push-Relabel



1. *Setze* $f(e) = u(e) \quad \forall e \in \delta^+(s)$
2. *Setze* $f(e) = 0 \quad \forall e \in E \setminus \delta^+(s)$
3. *Setze* $\Psi(s) = n = |V|$
4. *Setze* $\Psi(v) = 0 \quad \forall v \in V \setminus \{s\}$
5. *while* (es gibt einen aktiven Knoten v) *do*
6. *if*(kein $e \in \delta^+_{G_f}(v)$ ist erlaubte Kante)
7. *then* $RELABEL(v)$
8. *else* $PUSH(e)$ mit $e \in \delta^+_{G_f}(v)$ erlaubt

PUSH(e)

1. *Setze* $\gamma := min\{ex_f(v), u_f(e)\}$, wobei e in v beginnt
2. Augmentiere f entlang e um γ

RELABEL(v)

1. *Setze* $\Psi(v) := min\{\Psi(w) + 1$,wobei $(v, w) \in \delta^+_{G_f}(v)\}$

Beschriftung der Knoten:
Überschuss $ex_f(v)$ | Distanzmarkierung $\Psi(v)$

Korrektheit

Beobachtung

Während des gesamten Ablaufs des Algorithmus ist f stets ein s-t-Präfluss und Ψ stets eine Distanzmarkierung bezüglich f.
Für jedes $v \in V(G)$ gilt: $\Psi(v)$ wird durch jedes $RELABEL(v)$ streng erhöht.

Korrektheit

Beweis.

- Aussage gilt am Anfang des Algorithmus

Korrektheit

Beweis.

- Aussage gilt am Anfang des Algorithmus
- s-t-Flusseigenschaft:
  - nach PUSH-Operation immer noch Präfluss, da nicht mehr als $ex_f(v)$ gepusht wird

Korrektheit

Beweis.

- Aussage gilt am Anfang des Algorithmus
- s-t-Flusseigenschaft:
  - nach PUSH-Operation immer noch Präfluss, da nicht mehr als $ex_f(v)$ gepusht wird
  - bei RELABEL-Operation wird Fluss nicht verändert

Korrektheit

Beweis.

- Aussage gilt am Anfang des Algorithmus
- s-t-Flusseigenschaft:
  - nach PUSH-Operation immer noch Präfluss, da nicht mehr als $ex_f(v)$ gepusht wird
  - bei RELABEL-Operation wird Fluss nicht verändert
- Distanzmarkierung:
  - RELABEL(v): $\Psi(v)$ war vorher schon Distanzmarkierung und kein $e \in \delta +_{G_f}$ ist erlaubt
    $\Rightarrow \Psi(v) < \Psi(w) + 1$ und $\Psi(v)$ wird streng erhöht auf
    $min\{\Psi(w) + 1 | (v, w) \in \delta +_{G_f} (v)\}$

Korrektheit

Beweis.

- Aussage gilt am Anfang des Algorithmus
- s-t-Flusseigenschaft:
  - nach PUSH-Operation immer noch Präfluss, da nicht mehr als $ex_f(v)$ gepusht wird
  - bei RELABEL-Operation wird Fluss nicht verändert
- Distanzmarkierung:
  - RELABEL(v): $\Psi(v)$ war vorher schon Distanzmarkierung und kein $e \in \delta +_{G_f}$ ist erlaubt
    $\Rightarrow \Psi(v) < \Psi(w) + 1$ und $\Psi(v)$ wird streng erhöht auf $min\{\Psi(w) + 1 | (v, w) \in \delta +_{G_f} (v)\}$
  - PUSH auf (v,w) : Einzige neue Kante ist Rückkante (w,v). Da aber (v,w) erlaubte Kante ist, gilt $\Psi(w) = \Psi(v) - 1$

$\square$

Korrektheit

Lemma

Sei f ein s-t-Präfluss und Ψ eine Distanzmarkierung bezüglich f.
Dann gelten folgende Aussagen:

(a) *Es ist s von jedem aktiven Knoten v aus in G_f erreichbar.*

(b) *Es ist t nicht von s aus in G_f erreichbar.*

Korrektheit

Lemma

Sei f ein s-t-Präfluss und Ψ eine Distanzmarkierung bezüglich f.
Dann gelten folgende Aussagen:

(a) *Es ist s von jedem aktiven Knoten v aus in G_f erreichbar.*
(b) *Es ist t nicht von s aus in G_f erreichbar.*

Beweis.

(a) Sei v beliebiger aktive Knoten und R die Menge der von v aus in G_f erreichbaren Knoten. Dann ist $f(e) = 0$ für alle $e \in \delta_G^-(R)$ und es gilt:

$$\sum_{w \in R} ex_f(w) = \sum_{e \in \delta_G^-(R)} f(e) - \sum_{e \in \delta_G^+(R)} f(e) \leq 0$$

v ist aktiv $\Rightarrow ex_f(v) > 0$ und somit gibt es Knoten $w \in R$ mit $ex_f(w) < 0$.
Da f ein s-t-Präfluss ist, muss dieser Knoten s sein.

$\square$

Korrektheit

Lemma

Sei f ein s-t-Präfluss und Ψ eine Distanzmarkierung bezüglich f.
Dann gelten folgende Aussagen:

(a) *Es ist s von jedem aktiven Knoten v aus in G_f erreichbar.*

(b) *Es ist t nicht von s aus in G_f erreichbar.*

Beweis.

(b) Annahme: Es gibt einen s-t-Weg in G_f mit Knoten
$s = v_0, v_1, \ldots, v_k = t$
Da Ψ Distanzmarkierung bezüglich f ist, gilt $\Psi(v_i) \leq \Psi(v_{i+1}) + 1$ für $i = 0, \ldots, k-1$.
Somit gilt $n = \Psi(s) \leq \Psi(t) + k = 0 + k$. Hierbei gilt $k \leq n - 1$.

□

Korrektheit

Satz

Bei Terminierung des Algorithmus ist f ein s-t-Fluss mit maximalem Wert.

Korrektheit

Satz

Bei Terminierung des Algorithmus ist f ein s-t-Fluss mit maximalem Wert.

Beweis.

Bei Terminierung gibt es keine aktiven Knoten mehr $\Rightarrow$ f ist ein s-t-Fluss.
Nach vorherigem Lemma (b) gibt es keinen augmentierenden Weg mehr in G_f
$\Rightarrow$ f ist maximal.

$\square$

Laufzeit

Wie oft wird PUSH und RELABEL aufgerufen?

Laufzeit

Wie oft wird PUSH und RELABEL aufgerufen?

Lemma

(a) *Für jedes* $v \in V(G)$ *gilt :* $\Psi(v) \leq 2n - 1$ *zu jedem Zeitpunkt des Algorithmus.*

(b) *Kein Knoten wird mehr als* $(2n - 1)$ *mal geRELABELt. Die Gesamterhöhung von* $\sum\limits_{v \in V(G)} \Psi(v)$ *während des Ablaufs des Algorithmus ist höchstens gleich* $2n^2 - n$.

Laufzeit

Wie oft wird PUSH und RELABEL aufgerufen?

Lemma

(a) *Für jedes $v \in V(G)$ gilt : $\Psi(v) \leq 2n - 1$ zu jedem Zeitpunkt des Algorithmus.*

(b) *Kein Knoten wird mehr als $(2n-1)$ mal geRELABELt. Die Gesamterhöhung von $\sum_{v \in V(G)} \Psi(v)$ während des Ablaufs des Algorithmus ist höchstens gleich $2n^2 - n$.*

Beweis.

$\Psi(v)$ wird durch jedes *RELABEL*(v) streng erhöht.
RELABEL(v) wird nur aufgerufen, wenn v aktiver Knoten ist.
v ist nach der RELABEL-Operation immer noch aktiv.
Nach vorherigem Lemma ist s von v zu erreichen und es gilt:
$\Psi(v) \leq \Psi(s) + n - 1 = 2n - 1$.

□

Laufzeit

Bei der Anzahl der PUSH-Operationen unterscheiden wir zwei Arten:

- **saturierender Push** : mit $u_f(e) = 0$ nach dem Push (Kante wurde voll ausgelastet durch den Push)
- **nichtsaturierender Push**

Lemma

Die Anzahl der saturierenden Pushes ist höchstens $2mn$.

Lemma

Die Anzahl der saturierenden Pushes ist höchstens $2mn$.

Beweis.

Nach einem saturierendem Push von v nach w ist $(v, w) \notin E(G_f)$. Damit ein weiterer saturierender Push auf (v, w) stattfinden kann, muss

1. $\Psi(w)$ um mind. 2 wachsen
2. Ein Push von w nach v stattfinden
3. $\Psi(v)$ um mind. 2 wachsen

Da stats $\Psi(v) \leq 2n$ gilt, gibt es höchstens n saturierende Pushes pro Kante.

$\Rightarrow$ höchstens $2mn$ saturierende Pushes im Algorithmus. $\square$

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v \in V(G) : v \ \ aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v \in V(G): v \ \ aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v \in V(G):v \quad aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$
- **RELABEL-Operation** auf v:
  Da $|\{w \in V(G) : \Psi(w) \leq \Psi(v)\}| \leq n$ wird Φ um höchstens n erhöht
  $\Rightarrow \Phi$ wird sich durch *RELABEL*-Operation nicht mehr als um
  $n * (2n^2 - n)$ erhöhen.

□

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v(G):v \quad aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$
- *RELABEL* : max Erhöhung um $n * (2n^2 - n)$

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v(G):v \quad aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$
- *RELABEL* : max Erhöhung um $n * (2n^2 - n)$
- **saturierender Push** auf Kante (v,w) :
  Event. wird w ein neuer aktiver Knoten und v bleibt ein aktiver Knoten.
  Dann erhöht sich Φ um maximal n.
  $\Rightarrow \Phi$ wird sich durch *saturierende Pushes* nicht mehr als um $n * (2mn)$ erhöhen.

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v(G):v \quad aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$
- *RELABEL* : max Erhöhung um $n * (2n^2 - n)$
- *saturierende Pushes*: max Erhöhung um $n * 2mn$.

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v(G):v \quad aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$
- *RELABEL* : max Erhöhung um $n * (2n^2 - n)$
- *saturierende Pushes*: max Erhöhung um $n * 2mn$.
- **nichtsaturierende Push** auf Kante (v,w):
  Event. wird w ein neuer aktiver Knoten, aber v wird auf jeden Fall inaktiv.
  Dann erniedrigt sich Φ um mind. eins.
  $\Rightarrow \Phi$ wird sich durch *nichtsaturiende Pushes* um mind. 1 verringern.

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v(G):v \;\; aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$
- *RELABEL* : max Erhöhung um $n * (2n^2 - n)$
- *saturierende Pushes*: max Erhöhung um $n * 2mn$.
- *nichtsaturierende Pushes*: min Erniedrigung um 1.

Lemma

Die Anzahl der nichtsaturierenden Pushes ist $O(n^2 m)$.

Beweis.

Betrachte

$$\Phi = \sum_{v(G):v \quad aktiv} |\{w \in V(G) : \Psi(w) \leq \Psi(v)\}|.$$

- Zu Beginn des Algorithmus gilt $\Phi \leq n^2$
- *RELABEL* : max Erhöhung um $n * (2n^2 - n)$
- *saturierende Pushes*: max Erhöhung um $n * 2mn$.
- *nichtsaturierende Pushes*: min Erniedrigung um 1.

$\Rightarrow$ Da am Ende $\Phi = 0$ gilt, ist die Gesamtabnahme von Φ höchstens:

$$n^2 + n(2n^2 - n) + n(2mn) \leq 2n^3 + 2n^2 m \leq 4n^2 m$$

$\square$

Lemma

Die Laufzeit des Push-Relabel-Algorithmus ist $O(n^2 m)$.

Beweis.

- max. $2n^2 - n$ mal wird *RELABEL* aufgerufen
- max. $2mn$ saturierende Pushes
- max. $4n^2 m$ nichtsaturierende Pushes

$\Rightarrow$ Laufzeit: $O(n^2 m)$ □

Quellen

- Korte, Vygen: Kombinatorische Optimierung, Springer
- Röglin: Skript zur Vorlesung Älgorithmen und Berechnungskomplexität I", WiSe10/11
- FAU-Erlangen-Nürnberg: https://www2.cs.fau.de/EN/teaching/SS2008/HalloWelt/fsb2008.pdf (Beispiel Push-Relabel)

Quellen

- Korte, Vygen: Kombinatorische Optimierung, Springer
- Röglin: Skript zur Vorlesung Älgorithmen und Berechnungskomplexität I", WiSe10/11
- FAU-Erlangen-Nürnberg: https://www2.cs.fau.de/EN/teaching/SS2008/HalloWelt/fsb2008.pdf (Beispiel Push-Relabel)

Vielen Dank für die Aufmerksamkeit!