Algorithmus GOLDBERG

Eingabe: Flussnetzwerk $G = (V, E, c, s, t)$
Ausgabe: maximaler Fluss f
Methode:

(1) Starte mit folgendem Preflow p

$$p(s,v) := c(s,v) \quad \forall (s,v) \in E$$
$$p(e) := 0 \quad \forall e \in E \setminus \{(s,v) \mid v \in V\}$$

und einer beliebigen gültigen Anfangsmarkierung

(2) Solange wie möglich führe in irgendeiner Anordnung folgende Basisoperationen aus:

- **Push(v,w):**

  Anwendbarkeit: v aktiv, $r_p(v,w) > 0$ und $d(v) = d(w) + 1$

  Durchführung:

  $\delta := \min\{\Delta p(v), r_p(v,w)\}$;
  $p(v,w) := p(v,w) + \delta$;
  $p(w,v) := p(w,v) - \delta$;
  $\Delta p(v) := \Delta p(v) - \delta$;
  $\Delta p(w) := \Delta p(w) + \delta$.

  Effekt: δ Preflow wird von v nach w gesendet

- **Relabel(v):**

Anwendbarkeit: v aktiv und bzgl. v ist keine Push-Operation anwendbar.

D.h., $\forall w \in V: r_p(v,w) > 0 \Rightarrow d(v) \leq d(w)$

Durchführung:

$$d(v) := \begin{cases} \min \{ d(w)+1 \mid r_p(v,w) > 0 \} & \text{falls } \{ w \mid r_p(v,w) > 0 \} \neq \emptyset \\ \infty & \text{sonst.} \end{cases}$$

Bezeichnung:

Eine Push-Operation Push (v,w) heißt saturierend falls $r_p(v,w) = 0$ nach Durchführung der Operation und nichtsaturierend sonst.

Beobachtung:

- Die Operation Relabel (v) erhöht $d(v)$ unter Beachtung der Gültigkeit der Markierung d weitmöglichst
- Falls nach Relabel (v) $d(v) \neq \infty$, dann existiert $w \in V$, so dass Push (v,w) durchführbar ist.

Folgendes Lemma kann leicht aus obigen Ausführungen gefolgert werden:

Lemma 2.7

Sei p ein Preflow, d eine beliebige gültige Markierung und v ein aktiver Knoten. Dann ist entweder

Push(v,w) für ein $w \in V$ oder Relabel(v) anwendbar.

Beweis: Übung.

Mögliche Anfangsmarkierungen:

a) $d(s) = n$ und $d(v) = 0 \quad \forall v \in V \setminus \{s\}$
(Beachte, dass zu Beginn jede ausgehende Kante von s saturiert ist.)

b) Sei p ein anfänglicher Preflow. $\forall v \in V$ definiere

$$d(v) := \min \{ d_{G_p}(v,t),\ d_{G_p}(v,s) + n \}.$$

↑ ok! nicht $d(s,v)$

Übung:
Zeigen Sie, dass obige Anfangsmarkierungen gültig sind und in $O(m+n)$ Zeit auch berechnet werden können.

Ziel:
Beweis der Korrektheit von Goldbergs Algorithmus und dass dieser auch terminiert.

Wir nehmen hierzu zunächst an, dass Goldbergs Algorithmus terminiert und beweisen unter dieser Voraussetzung dessen Korrektheit. Danach werden wir beweisen, dass Goldbergs Algorithmus terminiert. Folgendes Lemma sagt, dass die Markierung d niemals ungültig wird:

Lemma 2.8

Während des gesamten Algorithmus bleibt die Markierung d immer gültig.

Beweis:

Gemäß Konstruktion startet der Algorithmus mit einer gültigen Markierung d.

Annahme:

Die Behauptung des Lemmas gilt nicht.

Betrachte die erste Relabel- oder Push-Operation, nach der die Markierung d nicht mehr gültig ist:

1) **Relabel(v):**

Gemäß Konstruktion gilt nach Durchführung von Relabel(v):

$$d(v) \leq d(w) + 1 \quad \forall (v,w) \in E_p$$

$\Rightarrow$ d ist nach wie vor gültig $\lightning$.

2) **Push(v,w):**

Push(v,w) kann folgende Auswirkungen haben:

i) (w,v) wird nützlich (und war es vorher nicht).

ii) $r_p(v,w)$ wird 0, d.h., (v,w) ist nicht mehr nützlich.

Wegen $d(w) = d(v) - 1$ gilt $d(w) \leq d(v) + 1$.
Somit bleibt d auch nach Auswirkung i) gültig.

Die Aussage ii) hat keinen Effekt auf die Gültigkeit der Markierung d. ■

Lemma 2.9

Sei p ein Preflow und d eine gültige Markierung bzgl. p. Dann ist im Restgraph G_p die Senke t nicht von s aus erreichbar.

Beweis:

Annahme:

t ist in $G_p = (V, E_p)$ von s aus erreichbar.

$\Rightarrow$

$\exists$ in G_p ein einfacher Pfad

$$P = (s = v_0), v_1, v_2, \ldots, (v_\ell = t).$$

Dann gilt

- $\ell < n$ und
- (v_i, v_{i+1}) ist möglich für $0 \leq i < \ell$.

$\Rightarrow$

$$d(s) \leq d(t) + \ell < n \quad (\text{da } d(t) = 0)$$

Dies widerspricht $d(s) = n$. ■

Satz 2.3

Falls der Algorithmus von Goldberg terminiert, dann ist der berechnete Preflow p ein maximaler Fluss.

Beweis:
Falls der Algorithmus terminiert, dann impliziert Lemma 2.7

$$\Delta_p(v) = 0 \quad \forall v \in V \setminus \{s,t\}$$

$\Rightarrow$ p ist ein Fluss.

Lemma 2.8 $\Rightarrow$

$\exists$ s,t-separierender Schnitt $(S, V \setminus S)$, so dass jede Kante von S nach $V \setminus S$ saturiert ist und jede Kante von $V \setminus S$ nach S Nullfluss hat.

Max-Flow Min-Cut Theorem $\Rightarrow$ p ist maximal. ∎

Als nächstes werden wir die Anzahl der durchgeführten Basisoperationen abschätzen.

Lemma 2.10
Sei p ein Preflow und $v \in V \setminus \{s,t\}$ mit $\Delta_p(v) > 0$
Dann ist in G_p der Startknoten s von v erreichbar

Beweis:
$\Delta_p(v) > 0 \Rightarrow$

$\exists$ Pfad $P = (s = v_0), v_1, \dots, (v_r = v)$ mit $p(v_i, v_{i+1}) > 0$, $0 \le i$

$\Rightarrow$ $r_p(v_{i+1}, v_i) > 0$, $0 \le i < r$
$\Rightarrow$ $(v_{i+1}, v_i) \in E_p$, $0 \le i < r$
$\Rightarrow$ s ist von v in G_p erreichbar. ∎

Lemma 2.11

Für jeden Knoten $v \in V \setminus \{s,t\}$ erhöht eine Anwendung von Relabel(v) seine Markierung $d(v)$ echt.

Beweis:

Konstruktion $\Rightarrow$

Vor Anwendung gilt $\forall w$ mit (v,w) nützlich

$$d(v) \leq d(w).$$

Nach Anwendung existiert ein w mit (v,w) nützlich und $d(v) = d(w) + 1$.

■

Lemma 2.12

$\forall v \in V$ gilt stets $d(v) \leq 2n-1$.

Beweis:

Das Lemma ist trivial für $v \in \{s,t\}$. Betrachte $v \in V \setminus \{s,t\}$.

Lemma 2.10 und die Tatsache, dass nur Relabel(v) die Markierung $d(v)$ ändert impliziert:

Zum Zeitpunkt der Änderung von $d(v)$ existiert stets ein einfacher Pfad von v nach s in G_p.

$\Rightarrow$

$$d(v) \leq d(s) + n - 2$$

nach der letzten Änderung von $d(v)$.

$\Rightarrow$

$$d(v) \leq d(s) + n - 1 = 2n - 1$$

■

Lemma 2.13

Die Anzahl der Relabeloperationen ist für jeden Knoten $\leq 2n-1$, also insgesamt $\leq (n-2)(2n-1) < 2n^2$.

Beweis:

Die Behauptung folgt direkt aus Lemmata 2.11 und 2.12.

■

Lemma 2.14

Die Anzahl der saturierenden Push-Operationen ist $\leq 2 \cdot n \cdot m$.

Beweis:

Für jedes Paar $(v,w) \in V \times V$ betrachte die saturierenden Push-Operationen

- von v nach w und von w nach v.

Falls solche Push-Operationen existieren, dann gilt:

$$(v,w) \in E \text{ oder } (w,v) \in E.$$

Nach einer saturierenden Push-Operation von v nach w muss zunächst Preflow von w nach v gebracht werden, bevor wieder Preflow von v nach w geschickt werden kann.

Anwendbarkeitsbedingung für Push(v,w)

$\Rightarrow$

- d(v) muss um mindestens 2 erhöht werden.

Analog muss zwischen saturierenden Push-Operationen von w nach v

- d(v) um mindestens 2 erhöht werden.

Bei der Durchführung der ersten Push-Operation zwischen v und w gilt:

$$d(v) + d(w) \geq 1.$$

Lemma 2.12 $\Rightarrow$

Bei der Durchführung der letzten Push-Operation zwischen v und w gilt:

$$d(v) + d(w) \leq 4n - 3$$

$\Rightarrow$

Gesamtanzahl der saturierenden Push-Operationen zwischen v und w

$$\leq 2n - 1.$$

Also gilt für die Gesamtanzahl GS aller saturierenden Push-Operationen

$$GS \leq m(2n-1) < 2nm$$

■

Lemma 2.15

Die Anzahl der nichtsaturierenden Push-Operationen ist $\leq 4n^2 \cdot m$.

Beweis:

Sei $$\Phi := \sum_{\{v \mid v \text{ ist aktiv}\}} d(v).$$

Beobachtung:

- Nach jeder nichtsaturierenden Push-Operation von v nach w wird v inaktiv. Ferner gilt $d(w) = d(v) - 1$

  $\Rightarrow$

  Φ wird mindestens um 1 verringert.

Eine saturierende Push-Operation erhöht Φ höchstens um $2n-1$, da Knotenmarkierungen stets $\leq 2n-1$ sind.

Lemma 2.14 $\Rightarrow$

Gesamterhöhung durch saturierende Push-Operationen
$$\leq 2 \cdot n \cdot m \,(2n-1)$$

Da eine Relabeling-Operation betreffende Knotenmarkierung stets strikt erhöht und diese niemals größer als $2n-1$ wird, ist die durch Relabeling-Operationen bedingte Gesamterhöhung von Φ
$$\leq (n-2)(2n-1).$$

operationen ermöglichen.

Datenstrukturen: (für jeden Knoten $v \in V$)

- Liste L_v, die jede ungerichtete Kante $\{v,w\}$ in einer beliebigen, jedoch festen Anordnung enthält.
  (D.h., jede ungerichtete Kante erscheint in genau zwei Listen).

- Die aktuelle Kante $\{v,w\}$ bzgl. v ist gekennzeichnet. (Diese repräsentiert den aktuellen Kandidaten für die nächste von v ausgehende Push-Operation).
  Zu Beginn ist die erste Kante in L_v gekennzeichnet.

Lemma 2.7 besagt, dass für einen aktiven Knoten stets entweder eine Push- oder die Relabel-Operation anwendbar ist. Wir fassen zunächst beide Operationen zu einer zusammen.

Push / Relabel (v):

Anwendbarkeit: v ist aktiv

Durchführung:

Sei $\{v,w\}$ die aktuelle Kante bzgl. v.

```
if Push(v,w) anwendbar
  then
      Push(v,w)
  else
    if {v,w} ist nicht letzte Kante in L_v
      then
        markiere nächste Kante in L_v als
        aktuelle Kante
      else
        markiere erste Kante in L_v als
        aktuelle Kante;
        Relabel(v)
      fi
  fi.
```

Lemma 2.16

Die Push/Relabel-Operation führt eine Relabel-Operation nur dann durch, wenn diese auch anwendbar ist.

Beweis:

Relabel(v) wird nur dann durchgeführt, wenn v aktiv ist.

2.2.: $\forall w \in V: r_p(v,w) > 0 \Rightarrow d(v) \leq d(w)$.

Es gilt:

- Zu Beginn und nach jeder Relabel(v)-Operation ist die erste Kante in L_v aktuell.
  Diese Eigenschaft wird von Kante $\{v,w\}$ genau dann weitergereicht, wenn der Knoten v bearbeitet wird und Push(v,w) nicht anwendbar (d.h., $r_p(v,w) \leq 0$ oder $d(v) \leq d(w)$) ist.
- $d(v)$ wird nur durch Anwendung von Relabel(v) erhöht und $r_p(v,w)$ kann nur anwachsen, falls $d(w) > d(v)$.

$\Rightarrow$

Relabel(v) ist anwendbar. ∎

Sei stets

$$Q := \{ v \in V \setminus \{s,t\} \mid v \text{ ist aktiv} \}.$$

Zu Beginn gilt

$$Q = \{ v \in V \setminus \{s,t\} \mid c(s,v) > 0 \}.$$

<u>Ziel:</u> effiziente Verwaltung von Q.

<u>Beobachtung:</u>

Pro Push / Relabel-Operation wird maximal ein Knoten in Q eingefügt und auch maximal ein Knoten aus Q entfernt. Genauer,

Die Gesamtlaufzeit ist somit begrenzt durch

$$O(n \cdot m) + \#\text{Push-Operationen} \cdot O(1)$$

$$\overset{=}{\text{Le. 2.14, 2.15}} \quad O(n^2 m).$$

Ziel: Bessere Laufzeiten.

Idee:
Reduktion der Anzahl der nichtsaturierenden Push-Operationen mittels Anwendung von Regeln bei der Abarbeitung von Q.

FIFO-Strategie: Verwaltung von Q als Schlange.

Algorithmus MAXFLOW/FIFO
Eingabe: Flussnetzwerk $G = (V, E, c, s, t)$
Ausgabe: maximaler Fluss f
Methode:

```
Q := { v ∈ V\{s,t} | c(s,v) > 0 };
while Q ≠ ∅
    do
        Entferne v von der Front von Q;
        repeat
            Push/Relabel(v);
            if w wird aktiv während der
                Push/Relabel-Operation
```

then
füge w an das Ende von Q hinten
fi
until ($\Delta_p(v) = 0$ oder $d(v)$ wird erhöht.)
if v ist noch immer aktiv
then
füge v an das Ende von Q
fi
od.

Für die Analyse von MAXFLOW/FIFO teilen wir das Abarbeiten von Q in Phasen ein.

Phase 1:
Push/Relabel(v), wobei v durch die Initialisierung noch Q kam.

Phase 2:
Push/Relabel(v), wobei v in Phase 1 an das Ende von Q gehängt wurde.

Phase 3:
Push/Relabel(v), wobei v in Phase 2 an das Ende von Q gehängt wurde.

⋮

Phase i:
Push/Relabel(v), wobei v in Phase(i-1) an das Ende von Q gehängt wurde.

⋮

Lemma 2.17

Die Anzahl der Phasen ist $< 4 \cdot n^2$.

Beweis:

Sei $\phi := \max \{ d(v) \mid v \text{ ist aktiv} \}$.

Lemma 2.13 $\Rightarrow$

Die Anzahl der Phasen mit mindestens einer Relabel-Operation ist

$$< 2n^2.$$

- In einer Phase, in der keine Relabel-Operation erfolgt, hat jeder Knoten seinen überschießenden Preflow zu Knoten mit kleinerer Markierung geschickt.

  $\Rightarrow$

  ϕ vermindert sich mindestens um 1.

- Wenn ϕ erhöht wird, dann erhöht sich die Markierung eines Knotens mindestens um den Wert, um den ϕ erhöht wird.

- Zu Beginn gilt $\phi = 0$.

$\Rightarrow$

Die Anzahl der Phasen ohne Relabel-Operation ist $\leq$ Summe der Markierungserhöhungen

$$\underset{\text{Le 2.12}}{<} 2n^2.$$

Satz 2.6

Der Algorithmus MaxFlow/FIFO benötigt $O(n^3)$ Zeit.

Beweis:

Nach einer nichtsaturierenden Push-Operation bzgl. v gilt $\Delta p(v) = 0$.

$\Rightarrow$

$\forall\, v \in V \setminus \{s,t\}$ existiert pro Phase höchstens eine nichtsaturierende Push-Operation.

Lemma 2.17 $\Rightarrow$

Die Gesamtanzahl der nichtsaturierenden Push-Operationen ist

$$< (n-2) \cdot 4n^2 < 4n^3.$$

∎

Maximalwert-Strategie:

Bearbeite stets einen aktiven Knoten v mit

$$d(v) = \max \{ d(x) \mid x \text{ ist aktiv} \}.$$

Dann kann gezeigt werden, dass die Anzahl der nichtsaturierenden Push-Operationen $O(n^2 \cdot \sqrt{m})$ ist und dass auch eine $O(n^2 \sqrt{m})$-Implementierung existiert.

Siehe: Ahuja, Magnanti, Orlin. Network flows, S. 233 ff.

3. Das minimale Kosten Netzwerkflussproblem

Ein gerichteter Graph $G = (V, E, u, c, b)$ mit

$c: E \to \mathbb{R}$, c_{ij} Kosten bzgl. Kante $(i,j) \in E$

$u: E \to \mathbb{R}^+$, u_{ij} Kapazität der Kante $(i,j) \in E$

$b: V \to \mathbb{R}$ mit

$$b(i) = \begin{cases} \text{Angebot des Knotens } i & \text{falls } b(i) \geq 0 \\ \text{Nachfrage des Knotens } i & \text{falls } b(i) \leq 0 \end{cases}$$

heißt (allgemeines) Flußnetzwerk

Ein Knoten $i \in V$ mit $b(i) = 0$ heißt Durchgangs-knoten.

Folgendes lineare Programm definiert das minimale Kosten Netzwerkflussproblem:

$$\text{minimiere} \sum_{(i,j) \in E} c_{ij} x_{ij} =: z(x)$$

wobei

$$(1) \sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} = b(i) \quad \forall i \in V$$

(Flussbalancierungsbedingung)

$$(2) \quad 0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in E$$

(Kapazitätsbedingung)