Kapitel 4, Teil 3

Flüsse: Goldberg-Tarjan

Effiziente Algorithmen, SS 2018

Professor Dr. Petra Mutzel

VO 8/9 am 8./15. Mai 2018

4.6 Der Goldberg-Tarjan Algorithmus

bisher betrachtete Flussalgorithmen

- Algorithmus von Ford und Fulkerson
- Algorithmus von Dinic
- Algorithmus von Malhotra, Pramodh Kumar und Maheshwari

Beobachtung Gemeinsamkeit rundenorientiert

1. Starte mit leerem Fluss Φ.
2. Berechne Fluss Ψ für den Restgraphen.
3. $\Phi := \Phi + \Psi$
4. Wiederholen, falls $\Psi \neq 0$.

konkret

- Ford-Fulkerson beliebiger flussvergrößernder Weg
- Dinic Sperrfluss in N_Φ („naiv“ berechnet)
- Malhotra-Pramodh Kumar-Maheshwari
  Sperrfluss in N_Φ mit Forward-Backward-Propagation

Muss das so sein? Geht es grundsätzlich anders?

Forward-Backward-Propagation Revisited

bei Forward-Backward-Propagation

1. erzeuge Überschuss bei Knoten mit minimalem Potenzial
2. treibe Überschuss zur Senke und zur Quelle

Beobachtung so lange Knoten mit Überschuss existiert haben wir keinen Fluss, d.h. Kirchhoff-Regel nicht erfüllt

Definition 4.20

Für Netzwerk $(G = (V, E), c)$ heißt $\Phi \colon E \to \mathbb{R}_0^+$ Präfluss, wenn

- $\forall e \in E \colon \Phi(e) \leq c(e)$
- $\forall v \in V \setminus \{Q\} \colon e(v) \geq 0$

mit $e(v) := \sum_{e=(\cdot,v)\in E} \Phi(e) - \sum_{e=(v,\cdot)\in E} \Phi(e)$ gilt.

$v \in V \setminus \{Q, S\}$ mit $e(v) > 0$ heißt aktiv.

Über Präflüsse

klar Präfluss ist weniger enger Begriff als Fluss

also jeder Fluss ist auch Präfluss

Beobachtung Restgraph auch für Präfluss sinnvoll

klar Ziel Umwandlung Präfluss $\rightsquigarrow$ Fluss

Geht das überhaupt?

Erinnerung bei Malhotra et al.: Überschuss zur Not $\rightsquigarrow$ Quelle

Geht das für einen Präfluss auch?

Überschuss zur Quelle bringen

Lemma 4.21

Sei $(G = (V, E), c)$ Netzwerk, Φ Präfluss für G, $v \in V$ aktiv.
In Rest_Φ gibt es einen Weg von v nach Q.

Beweis.

Schreibweise $x \rightsquigarrow_\Phi y \overset{\text{def}}{\Leftrightarrow} \exists$ Weg von x zu y in Rest_Φ

Definiere $V^* := \{w \in V \mid v \rightsquigarrow_\Phi w\}$
$\overline{V^*} := V \setminus V^*$

zu zeigen $Q \in V^*$

Annahme $Q \notin V^*$ (Ziel: Widerspruch)

Die Summe der Überschüsse

Annahme $Q \notin V^*$

Betrachte $\sum_{w \in V^*} e(w)$

klar $\sum_{w \in V^*} e(w) \geq 0$

weil $e(w) \geq 0$ für alle $w \neq Q$

wie für Min Cut=Max Flow Aufspaltung nach Kanten

klar $$\sum_{w \in V^*} e(w) = \sum_{w \in V^*} \left(\sum_{e=(\cdot,w) \in E} \Phi(e) - \sum_{e=(w,\cdot) \in E} \Phi(e) \right)$$

$e \in V^* \times V^*$ taucht positiv und negativ auf also Beitrag 0

$e \in \overline{V^*} \times \overline{V^*}$ taucht gar nicht auf also kein Beitrag

$e \in \overline{V^*} \times V^*$ taucht nur positiv auf also Beitrag $\Phi(e)$

$e \in V^* \times \overline{V^*}$ taucht nur negativ auf also Beitrag $-\Phi(e)$

Gesamtüberschuss in V^*

wir haben $\sum_{w \in V^*} e(w) = \sum_{e \in E \cap (\overline{V^*} \times V^*)} \Phi(e) - \sum_{e \in E \cap (V^* \times \overline{V^*})} \Phi(e)$

Betrachte $e = (v^*, v') \in E \cap (V^* \times \overline{V^*})$

klar es gibt Weg von v nach v^* (Def. V^*)

klar es gibt keinen Weg von v nach v'

also $(v^*, v') \notin \text{Rest}_\Phi$

also $\Phi(e) = c(e)$

Betrachte $e = (v', v^*) \in E \cap (\overline{V^*} \times V^*)$

klar es gibt Weg von v nach $v*$

klar es gibt keinen Weg von v nach v'

also $(v^*, v') \notin \text{Rest}_\Phi$

aber $(v', v^*) \in E$

also $\Phi(e) = 0$

(wäre $\Phi(e) > 0$, dann wäre $\text{rev}(e) = (v^*, v') \in \text{Rest}_\Phi$)

Zusammenfassung Summe der Überschüsse

Wir haben
$$\begin{aligned}\sum_{w\in V^*} e(w) &= \sum_{e\in E\cap(\overline{V^*}\times V^*)} \Phi(e) - \sum_{e\in E\cap(V^*\times\overline{V^*})} \Phi(e)\\ &= \sum_{e\in E\cap(\overline{V^*}\times V^*)} 0 - \sum_{e\in E\cap(V^*\times\overline{V^*})} c(e)\\ &= -\sum_{e\in E\cap(V^*\times\overline{V^*})} c(e) \quad \leq 0\end{aligned}$$

Erinnerung $\sum_{w\in V^*} e(w) \geq 0$

also $\sum_{w\in V^*} e(w) = 0$

Wir kommen zum Widerspruch

Wir haben $\sum_{w \in V^*} e(w) = 0$

Erinnerung $\forall w \in V^* : e(w) \geq 0$
weil $Q \notin V^*$ gemäß Annnahme
(und nur Q hat $e(w) \leq 0$)

also $\forall w \in V^* : e(w) = 0$

aber $v \in V^*$ mit $e(v) > 0$ Widerspruch
also $Q \in V^*$ (Q erreichbar von v in Rest_Φ) □

also immer möglich: Überschuss zur Quelle bringen

Sinnvolle Richtungen für den Überschuss

klar Überschuss zur Quelle bringen unproduktiv

Erinnerung bei Malhotra et al.: Überschuss erst zur Senke

Idee Markierungen zur Wegfindung hilfreich

hier Quelle „hoch“, Senke „tief“, Flussverschiebung „bergab“

Definition 4.22

Sei $(G = (V, E), c)$ Netzwerk, Φ Präfluss, $\text{Rest}_\Phi = (V, E_\Phi, r_\phi)$ Restgraph.

$d\colon V \to \mathbb{N}_0$ heißt gültige Knotenmarkierung, wenn

- $d(Q) = n$ $\qquad (n = |V|)$
- $d(S) = 0$
- $\forall e = (v, w) \in E_\Phi\colon d(v) \leq d(w) + 1$

gilt.

$e = (v, w) \in E_\Phi$ heißt wählbar, wenn $d(v) = d(w) + 1$ gilt.

Über gültige Knotenmarkierungen

Lemma 4.23

Sei $(G = (V, E), c)$ Netzwerk, Φ Präfluss, $\text{Rest}_\Phi = (V, E_\Phi, r_\phi)$ Restgraph, d gültige Knotenmarkierung.

1. $\forall v \neq w \in V$: jeder Weg in Rest_Φ von v nach w hat Länge $\geq d(v) - d(w)$.
2. Es gibt in Rest_Φ keinen Weg von Q nach S.

Beweis.

Beobachtung zweite Aussage folgt aus erster, weil:

klar Wege haben Länge $\leq n - 1$

gemäß Definition $d(Q) = n$, $d(S) = 0$

also jeder Q-S-Weg hat Länge $\geq d(Q) - d(S) = n - 0 = n$

also es gibt keinen Q-S-Weg

also nur noch erste Aussage zu zeigen

Gültige Knotenmarkierung und Weglängen

zu zeigen jeder v-w-Weg hat Länge $\geq d(v) - d(w)$

Betrachte Weg $(v, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{l-1}, w)$
der Länge l

Erinnerung $d(v_i) \leq d(v_{i+1}) + 1$ für alle i
(Def. gültige Knotenmarkierung)

also $d(v) \leq d(v_1) + 1 \leq d(v_2) + 2 \leq d(v_3) + 3 \leq \cdots \leq d(w) + l$

äquivalent $l \geq d(v) - d(w)$ □

Algorithmus von Goldberg und Tarjan

Algorithmus 4.24

1. Für alle $v \in V$
   $d(v) := 0$; $e(v) := 0$
2. $d(Q) := n$
3. $\Phi := 0$
4. Für alle $v \in V$ mit $e = (Q, v) \in E$
   $\Phi(e) := c(e)$; $e(v) := c(e)$
5. While $\exists v \in V \setminus S$ mit $e(v) > 0$
6. Führe anwendbare Basisoperation (Push oder Relabel) aus.
7. Ausgabe Φ

Basisoperationen

Erinnerung: $v \in V \setminus \{Q, S\}$ mit $e(v) > 0$ heißt aktiv.
Def.: $e = (v, w) \in E_\Phi$ heißt wählbar, wenn $d(v) = d(w) + 1$ gilt.

Push($e = (v, w)$)

{∗ *anwendbar, wenn v aktiv und e wählbar ist* ∗}

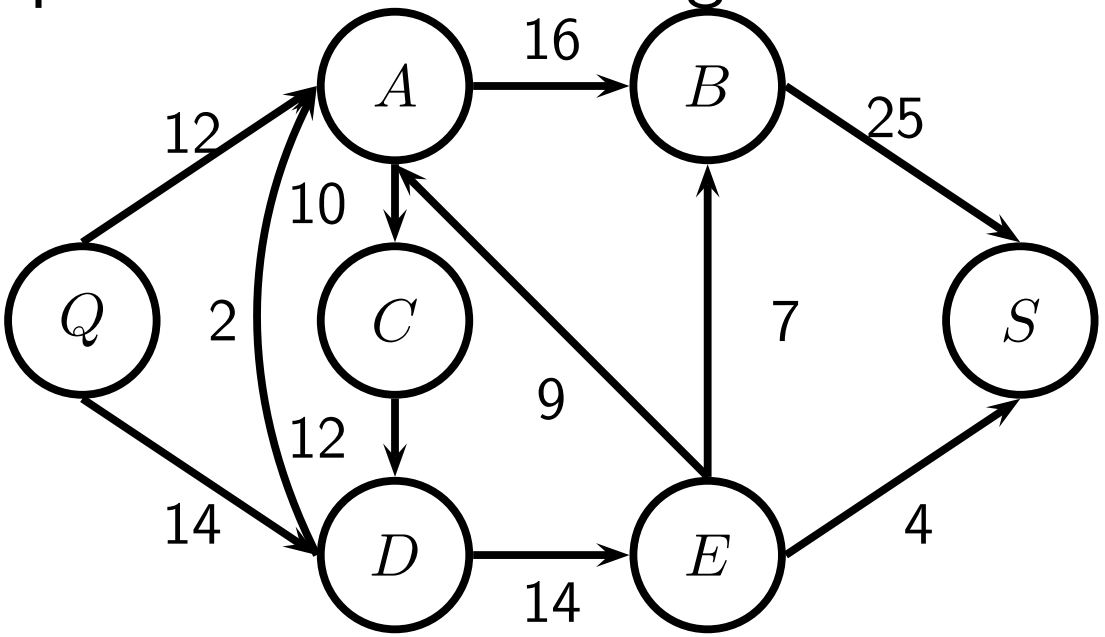
1. $\delta := \min\{e(v), r_\Phi(e)\}$
2. If $e \in E$
   Then $\Phi(e) := \Phi(e) + \delta$ {∗ *Vorwärtskante* ∗}
   Else $\Phi(e) := \Phi(e) - \delta$ {∗ *Rückwärtskante* ∗}
   $e(v) := e(v) - \delta$; $e(w) := e(w) + \delta$

Relabel(v)

{∗ *anwendbar, wenn v aktiv und keine Kante $(v, \cdot) \in E_\Phi$ wählbar ist* ∗}

1. $d(v) := \min\{d(w) + 1 \mid (v, w) \in E_\Phi\}$

Beispiel Preflow-Push-Algorithmus



Eingabe Netzwerk

1

Über das Beispiel

Anmerkungen

- sah nicht besonders schnell aus: viele Iterationen
- vielleicht viel Glück (oder „Lenkung“) im Spiel
- Ist der Fluss am Ende sicher maximal?
- Terminiert das Verfahren überhaupt immer?
- Falls ja, wie lange kann das dann dauern?

Einsicht Wir brauchen einen Korrektheitsbeweis.

Einsicht Wir brauchen eine Laufzeitanalyse.

Analyse Goldberg/Tarjan – Schritt für Schritt

klar zentral sind die Basisoperationen

Über Relabel

Lemma 4.25

Sei $(G = (V, E), c)$ Netzwerk, Φ Präfluss, $\text{Rest}_\Phi = (V, E_\Phi, r_\phi)$ Restgraph, d gültige Knotenmarkierung.
Wenn Relabel(v) anwendbar ist, erhöht Anwendung von Relabel(v) $d(v)$ um ≥ 1.

Beweis.

klar Relabel anwendbar $\Rightarrow v$ aktiv

also $\exists$ v-Q-Weg in Rest_Φ (Lemma 4.21)

also $\exists (v, w) \in E_\Phi$ für ein w $(v \neq Q)$ also

$\min\{d(w) + 1 \mid (v, w) \in E_\Phi\}$ wohldefiniert

Beweis von Lemma 4.25

Wir haben $\min\{d(w)+1\}$ über nichtleere Menge

klar Relabel anwendbar $\Rightarrow \forall (v,w) \in E_\Phi : d(v) \neq d(w)+1$
sonst (v,w) wählbar und Relabel(v) nicht anwendbar

klar d gültig $\Rightarrow \forall (v,w) \in E_\Phi : d(v) \leq d(w)+1$

also $\forall (v,w) \in E_\Phi : d(v) < d(w)+1$

nach Relabel(v) $\exists (v,w) \in E_\Phi : d(v) = d(w)+1$
also $d(v)$ um ≥ 1 gewachsen

□

Push(v, w) genauer betrachtet

Definition 4.26

Sei $(G = (V, E), c)$ Netzwerk, Φ Präfluss, $\text{Rest}_\Phi = (V, E_\Phi, r_\phi)$ Restgraph, d gültige Knotenmarkierung, $v \in V$ aktiv, $e = (v, w) \in E_\Phi$ wählbar.
Push(v, w) heißt saturierend, wenn in der Operation $\delta = r_\phi(e)$ gilt.
Sonst heißt Push(v, w) nichtsaturierend.

klar saturierendes Push(v, w) saturiert (v, w)

Beobachtung saturierendes Push saturiert Kante
$\rightsquigarrow$ Kante wird entfernt
offensichtlich „produktiv“

Beobachtung nichtsaturierendes Push
weniger offensichtliche Folgen
Nutzen nicht so klar – unproduktiv?

Über die Knotenmarkierung d

Lemma 4.27

Im Ablauf des Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) ist d immer eine gültige Knotenmarkierung.

Beweis.

Strategie

1. Knotenmarkierung d initial gültig.
2. Basisoperationen lassen d auch bei Veränderung gültig.

initial $d(Q) = n,\ d(S) = 0$ ✓

initial $\forall v \in V \setminus \{Q\}\colon d(v) = 0$
$\Rightarrow d(v) \leq d(w) + 1$ nur für Kanten $(Q, w) \in E_\Phi$ kritisch

initial $\forall e = (Q, w) \in E\colon \Phi(e) = c(e)$

also $e \notin E_\Phi$ ✓

Auswirkung von Push(v, w) auf d

Voraussetzung Push(v, w) anwendbar

1. Fall nichtsaturierendes Push

möglich rev$(e) = (w, v)$ wird erzeugt in Rest_Φ

Erinnerung Push(v, w) anwendbar
also $d(v) = d(w) + 1$

also $d(w) = d(v) - 1 \leq d(v) + 1$ ✓

2. Fall saturierendes Push

Beobachtung wie nicht-saturierendes Push nur
$e = (v, w)$ wird aus Rest_Φ entfernt

klar das entfernt Bedingung, schafft keine neue ✓

Auswirkungen von Relabel(v) auf d

Voraussetzung Relabel(v) anwendbar

Beobachtung kein Problem für Kanten $(v, w) \in E_\Phi$
weil $d(v)$ explizit korrekt gesetzt
$(d(v) := \min\{d(w) + 1 \mid (v, w) \in E_\Phi\})$

Betrachte Kante $e = (w, v) \in E_\Phi$

klar für $e = (w, v) \in E_\Phi$ vorher $d(w) \leq d(v) + 1$

Erinnerung $d(v)$ wird nur größer (Lemma 4.25)

also d weiterhin gültig □

Wiederholung: Zentrale Begriffe und Einsichten

- Überschuss $e(v) = \sum_{e=(\cdot,v)\in E} \Phi(e) - \sum_{e=(v,\cdot)\in E} \Phi(e)$
- Präfluss $e(v) \geq 0$ (bei Fluss $e(v) = 0$)
- Rest_Φ unverändert für Präfluss Φ sinnvoll
- Einsicht Überschuss zur Quelle transportierbar (Lemma 4.21)
- gültige Knotenmarkierung $d(Q) = n$, $d(S) = 0$ und $d(v) \leq d(w) + 1$ für alle $(v, w) \in E_\Phi$
- Einsicht in Rest_Φ mit gültiger Knotenmarkierung d haben alle v-w-Wege Länge $\geq d(v) - d(w)$ und es gibt keinen Q-S-Weg (Lemma 4.23)

Algorithmus von Goldberg und Tarjan

Algorithmus 4.24

1. Für alle $v \in V$
   $d(v) := 0$; $e(v) := 0$
2. $d(Q) := n$
3. $\Phi := 0$
4. Für alle $v \in V$ mit $e = (Q, v) \in E$
   $\Phi(e) := c(e)$; $e(v) := c(e)$
5. While $\exists v \in V \setminus S$ mit $e(v) > 0$
6. Führe anwendbare Basisoperation (Push oder Relabel) aus.
7. Ausgabe Φ

Basisoperationen

Push$(e = (v, w))$

{∗ *anwendbar, wenn* v *aktiv und* e *wählbar ist* ∗}

1. $\delta := \min\{e(v), r_\Phi(e)\}$
2. If $e \in E$
   Then $\Phi(e) := \Phi(e) + \delta$
   Else $\Phi(e) := \Phi(e) - \delta$ {∗ *Rückwärtskante* ∗}
   $e(v) := e(v) - \delta$; $e(w) := e(w) + \delta$

Relabel(v)

{∗ *anwendbar, wenn* v *aktiv und keine Kante* $(v, \cdot) \in E_\Phi$ *wählbar ist* ∗}

1. $d(v) := \min\{d(w) + 1 \mid (v, w) \in E_\Phi\}$

Analyse Goldberg/Tarjan – Schritt für Schritt

schon gesehen

Lemma 4.25

Sei $(G = (V, E), c)$ Netzwerk, Φ Präfluss, $\text{Rest}_\Phi = (V, E_\Phi, r_\phi)$ Restgraph, d gültige Knotenmarkierung.
Wenn Relabel(v) anwendbar ist, erhöht Anwendung von Relabel(v) $d(v)$ um ≥ 1.

Lemma 4.27

Im Ablauf des Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) ist d immer eine gültige Knotenmarkierung.

Über die Ausgabe von Goldberg/Tarjan

Lemma 4.28

Wenn der Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) stoppt, ist Φ ein maximaler Fluss.

Beweis.

klar Algorithmus stoppt $\Leftrightarrow$ kein Knoten aktiv

Beobachtung Φ Präfluss und kein Knoten aktiv
$\Leftrightarrow$ Φ Fluss

Erinnerung Knotenmarkierung gültig (Lemma 4.27)

also kein Q-S-Weg in Rest_Φ (Lemma 4.23)

also Φ maximal $\square$

Auf dem Weg zum Korrektheitsbeweis

wir haben Algorithmus von Goldberg und Tarjan partiell korrekt
berechnet maximalen Fluss, wenn und falls er stoppt

Zu Korrektheit Einsichten in die Entwicklung
der gültigen Knotenmarkierung d

Lemma 4.29

Im Ablauf des Algorithmus von Goldberg und Tarjan (Algo. 4.24) gilt $\forall v \in V : d(v)$ wächst monoton und $d(v) \leq 2n - 1$.

Beweis von Lemma 4.29

Zu Beweisen $\forall v \in V : d(v)$ wächst monoton und $d(v) \leq 2n - 1$.

Erinnerung nur Relabel ändert d und Relabel vergrößert nur

also $d(v)$ wächst monoton für alle v ✓

klar $d(Q) = n$ und $d(S) = 0$ fest ✓

Betrachte $v \in V \setminus \{Q, S\}$, Relabel$(v)$ anwendbar

also $e(v) > 0$ und $\exists$ v-Q-Weg in Rest_Φ (Lemma 4.21)

Sei v' erster Knoten hinter v auf kreisfreiem v-Q-Weg

klar Länge des Weges $\leq n - 1$

also $\ell :=$ Länge des Weges $v' \rightsquigarrow Q$: $\ell \leq n - 2$

darum $d(v') - d(Q) \leq \ell \leq n - 2 \Rightarrow d(v') - n \leq n - 2$ (Lem. 4.23)

also $d(v') \leq 2n - 2$

Betrachte Relabel(v)

klar anschließend $d(v) = \min\{d(w) + 1 \mid (v, w) \in E_\Phi\}$

$\leq 2n - 2 + 1 = 2n - 1$ □

Endlichkeit des Algorithmus von Goldberg und Tarjan

Erinnerung Goldberg/Tarjan „lebt“ von Basisoperationen

Idee Anzahl Basisoperationen nach oben beschränken

Lemma 4.30

In einem Ablauf des Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) werden weniger als $2n^2$ Relabel-Operationen ausgeführt.

Beweis.

Erinnerung

- initial alle Knotenmarkierungen $= 0$, und $d(Q) = n$
- Relabel(v) vergrößert $d(v)$ um ≥ 1 (Lemma 4.25)
- $\forall v \in V : d(v) \leq 2n - 1$ (Lemma 4.29)

also $\leq n \cdot (2n - 1) < 2n^2$ Relabel-Operationen

Anzahl der Push-Operationen

Erinnerung Unterscheidung saturierende Push-Operationen
und nichtsaturierende Push-Operationen
saturierende Pushs offensichtlich produktiv
nichtsaturierende Pushs weniger offensichtlich hilfreich

Lemma 4.31

In einem Ablauf des Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) werden höchstens $2ne$ saturierende Push-Operationen ausgeführt.

Beweis. (etwas anders als im Skript)
Betrachte saturierendes Push(e) mit $e = (v, w)$
Betrachte nächstes saturierendes Push(e) mit $e = (v, w)$
Behauptung dazwischen liegt Push(rev(e))
Begründung nach saturierendem Push(e), e nicht mehr in E_Φ
klar Einfügung passiert nur nach Push(rev(e))

Konsekutive saturierende Push-Operationen

wir haben zwischen zwei saturierenden Push(e)
liegt ein Push(rev(e))

Beobachtung beim ersten saturierenden Push(e) $(e = (v, w))$
gilt $d(v) = d(w) + 1$

Beobachtung bei Push(rev(e))
gilt $d(w) = d(v) + 1$

klar $d(w)$ muss um ≥ 2 gewachsen sein

analog beim zweiten saturierenden Push(e)
gilt wieder $d(v) = d(w) + 1$
also auch $d(v)$ um ≥ 2 gewachsen

Beweis von Lemma 4.31

Wir haben bei zwei konsekutiven Push(e) mit $e = (v, w)$ wächst $d(v) + d(w)$ um ≥ 4

Beobachtung $d(v) + d(w) \leq 4n - 3$ nach letztem sat. Push(e)
weil $d(v) \leq 2n - 1$ (Lemma 4.29)
und $d(v) = d(w) + 1$ (e wählbar)

also insgesamt $\leq n$ saturierende Push(e)

klar in $\text{Rest}_\Phi \leq 2e$ Kanten

also $\leq 2ne$ saturierende Push-Operationen □

Anzahl nichtsaturierender Push-Operationen

Lemma 4.32

In einem Ablauf des Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) werden weniger als $4n^2e$ nichtsaturierende Push-Operationen ausgeführt.

Beweis.

Idee (grob) Beobachte Algorithmus.
Beschreibe „Zustand“ numerisch $\rightsquigarrow$ Potenzialfunktion
Analysiere Verlauf Potenzialfunktion.

etwas konkreter zeige für Potenzialfunktion:
- wächst um ≥ 1 für nichtsaturierendes Push
- wird nicht kleiner für saturierendes Push und Relabel
- initial ≥ 0 und am Ende $\leq 4n^2e$

(Details zu Potenzialfunktionen und amortisierter Analyse in Kapitel 5)

Analyse der Potenzialfunktion

konkret Potenzialfunktion $P := P_1 + P_2 - P_3$ mit
$P_1 = (2n-2) \cdot$ #saturierende Push-Operationen bisher
$P_2 = \sum_{v \in V} d(v)$, $P_3 = \sum_{v \in V,\ v \text{ aktiv}} d(v)$

initial $P_1 = 0$, $P_2 = n$, $P_3 = 0$
also $P = 0 + n - 0 = n \geq 0$

Betrachte Push(e) mit $e = (v, w)$
klar v aktiv, $d(v) = d(w) + 1$, weil Push(e) anwendbar
1. Fall Push(e) nichtsaturierend
Beobachtung P_1 unverändert, P_2 unverändert
Beobachtung P_3 fällt um $d(v)$, kann wachsen um $d(w)$
zusammen P_3 fällt um $\geq d(v) - d(w) = 1$
insgesamt P wächst um ≥ 1

Analyse der Potenzialfunktion P (Fortsetzung)

haben Potenzialfunktion $P = P_1 + P_2 - P_3$ mit

$P_1 = (2n-2) \cdot$ #saturierende Push-Operationen bisher

$P_2 = \sum_{v \in V} d(v)$, $P_3 = \sum_{v \in V,\ v \text{ aktiv}} d(v)$

2. Fall Push(e) saturierend $(e = (v, w))$

Beobachtung P_1 wächst um $2n - 2$

Beobachtung P_2 unverändert

Beobachtung P_3 bezüglich $d(v)$ kann um $d(v)$ fallen

Beobachtung P_3 bezüglich $d(w)$ kann wachsen um $d(w) \leq 2n - 2$ (da $d(v) = d(w) + 1 \leq 2n - 1$)

insgesamt P kann nicht kleiner werden

Analyse der Potenzialfunktion P (Relabel)

haben Potenzialfunktion $P = P_1 + P_2 - P_3$ mit
$P_1 = (2n-2) \cdot \#$saturierende Push-Operationen bisher
$P_2 = \sum_{v \in V} d(v)$, $P_3 = \sum_{v \in V,\ v \text{ aktiv}} d(v)$

Betrachte Relabel(v)

Beobachtung P_1 unverändert

Beobachtung P_2 wächst um $h \geq 1$

Beobachtung P_3 wächst um $h \geq 1$

zusammen P unverändert

Fazit für alle Basisoperationen:
P – sinkt nie
P – wächst um ≥ 1 bei jedem nichtsaturierenden Push

Anzahl nichtsaturierender Push-Operationen beschränken

haben Potenzialfunktion $P = P_1 + P_2 - P_3$ mit

$P_1 = (2n-2) \cdot \#$saturierende Push-Operationen bisher

$P_2 = \sum_{v \in V} d(v)$, $P_3 = \sum_{v \in V,\ v \text{ aktiv}} d(v)$

initial $P = n$

Maximaler Wert der Potenzialfunktion

- $P_1 \leq (2n-2) \cdot 2ne$, weil $\leq 2ne$ saturierende Push-Operationen (Lemma 4.31)
- $P_2 \leq n \cdot (2n-1)$, weil alle Knotenmarkierungen $\leq 2n-1$ (Lemma 4.29)
- $P_3 \geq 0$

also $P \leq (2n-2) \cdot 2ne + n \cdot (2n-1) = 4n^2e - 4ne + 2n^2 - n$

$< 4n^2e + 2n(n-2e) \leq 4n^2e$

also $< 4n^2e$ nichtsaturierende Push-Operationen □

Zusammenfassung der kleinen Schritte

Theorem 4.33

Der Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) berechnet mit Anwendung von $O(n^2e)$ anwendbarer Basisoperationen in beliebiger Reihenfolge einen maximalen Fluss.

□

Konkrete Laufzeit?

klar hängt von Implementierung ab
und konkreter Reihenfolge der Operationen

Ist Gesamtlaufzeit $O(n^2e)$ erreichbar?

ja Beweis konstruktiv

Die Push/Relabel-Variante (Algorithmus 4.34)

1. Für alle $v \in V$ $d(v) := 0$; $e(v) := 0$;
2. Markiere erste Kante der Adj.-Liste von v als aktuell$_v$
3. $d(Q) := n$; $\Phi := 0$
4. Für alle $v \in V$ mit $e = (Q, v) \in E$
   $\Phi(e) := c(e)$; $e(v) := c(e)$
5. While $\exists v \in V \setminus S$ mit $e(v) > 0$
6. Führe Push/Relabel(v) aus.
7. Ausgabe Φ

Push/Relabel(v) {∗ *anwendbar, wenn v aktiv ist* ∗}

1. If $e = (v, w) =$ aktuell$_v$ wählbar Then Push(e); aktualisiere aktuell$_v$
2. Else
4. If e letztes Element in Liste Then
5. Relabel(v)
6. Mache erste Kante der Adj.-Liste von v zu aktuell$_v$.
3. Else Mache die nächste Kante aktuell$_v$.

Korrektheit der Push/Relabel-Variante?

Was müssen wir für die Korrektheit nachweisen?

klar Relabel-Operationen müssen anwendbar sein

Lemma 4.35

Wird im Ablauf von Algorithmus 4.34 Relabel(v) aufgerufen, dann ist Relabel(v) anwendbar.

Beweis.

Erinnerung Relabel(v) anwendbar
$\Leftrightarrow (e(v) > 0) \wedge (\forall (v,w) \in E_\phi \colon d(v) < d(w) + 1)$

klar beim Aufruf von Push/Relabel $e(v) > 0$

klar Relabel(v) nur ausgeführt, wenn Push nicht anwendbar

klar $e(v) > 0$ beim Aufruf von Relabel(v) ✓

Beweis von Lemma 4.35

noch zu zeigen beim Aufruf von Relabel(v)
keine Kante (v, w) wählbar

klar wenn $e = (v, w)$ inaktuell wird, ist e nicht wählbar
also zu zeigen $e = (v, w)$ wird nicht wieder wählbar

Beobachtung Relabel(v') mit $v' \neq v$
kann nur Kanten $(v', \cdot)$ wählbar machen ✓

Beobachtung Push ändert d nicht $\rightsquigarrow$ keine Kante neu wählbar ✓
Fertig? Nein! Push kann neue Kanten in E_Φ einfügen!
zu zeigen keine wählbare Kante (v, v') wird eingefügt
Betrachte Push(e') mit $e' = (x, y)$
klar e' wählbar, also $d(x) = d(y) + 1$
klar neue Kante kann nur rev$(e') = (y, x)$ sein
klar (y, x) mit $d(y) = d(x) - 1$ nicht wählbar ✓ □

Laufzeit der Push/Relabel-Variante

Theorem 4.36

Die Push/Relabel-Variante (Algorithmus 4.34) berechnet einen maximalen Fluss in Zeit $O(n^2e) = O(n^4)$.

Beweis.

klar Initialisierung in Zeit $O(n + e)$

Anzahl Basisoperationen

Relabel	$O(n^2)$	(Lemma 4.30)
saturierende Pushs	$O(ne) = O(n^3)$	(Lemma 4.31)
nichtsaturierende Pushs	$O(n^2e) = O(n^4)$	(Lemma 4.32)

klar aktive Knoten in Stack $\rightsquigarrow$ Zugriff in Zeit $O(1)$

Beobachtung jedes Push in Zeit $O(1)$
$\rightsquigarrow$ Zeit $O(n^2e)$ für alle Pushs

Gesamtaufwand durch Relabel-Operationen

wir haben $O(n^2)$ Relabel-Operationen

klar je Relabel von Knoten v: Zeit $O(deg(v)) = O(n)$

also Gesamtzeit Relabel $O(n^3)$ ✓

Anmerkung/Einschub bessere Abschätzung möglich:

Erinnerung Relabel(v) vergrößert $d(v)$ (Lemma 4.25)
$d(v) \leq 2n - 1$ (Lemma 4.29)

also jede Kante nur an $O(n)$ Relabel-Operationen beteiligt
(egal ob sie das Minimum durchpropagiert oder nicht)

also Gesamtzeit Relabel $O(ne)$ ✓

offen Durchlaufen der Adjazenzliste in Push/Relabel

Beweis von Theorem 4.36

Betrachte Durchlaufen der Adjazenzliste in Push/Relabel

klar Aufwand bei Aufruf von Basisoperation schon gezählt ✓

offen Schritte ohne Basisoperation

Beobachtung für festes v: $\deg(v)$ Aufrufe, dann 1 Relabel

also nur Zeit $O(n \cdot n)$, weil $\leq 2n - 1$ Relabel pro Knoten

insgesamt: $O(n^3)$ für alle Knoten

alles zusammen Gesamtzeit $O(n^2 e) = O(n^4)$ □

Erinnerung maximalen Fluss berechnen in Zeit $O(n^3)$ möglich

also also Goldberg/Tarjan bis hier enttäuschend

gute Nachricht Es geht leicht besser.

Die FIFO-Variante (Algorithmus 4.37)

1. Für alle $v \in V$
   $d(v) := 0$; $e(v) := 0$;
2. Markiere erste Kante der Adj.-Liste von v als aktuell$_v$
3. $d(Q) := n$
4. $\Phi := 0$; $Qu := \emptyset$
5. Für alle $v \in V$ mit $e = (Q, v) \in E$
   $\Phi(e) := c(e)$; $e(v) := c(e)$; Qu.Enqueue(v)
6. While $Qu \neq \emptyset$
7. $v := Qu$.Dequeue$()$
8. Repeat
9. Push/Relabel(v), füge dabei aktiv werdende Knoten in Qu ein.
10. Until $e(v) = 0$ oder Relabel(v) aufgerufen wurde.
11. If $e(v) > 0$ Then Qu.Enqueue(v)
12. Ausgabe Φ

Über die FIFO-Variante

Theorem 4.38

Die FIFO-Variante (Algorithmus 4.37) berechnet in Zeit $O(n^3)$ einen maximalen Fluss.

Beweis.

Beobachtung bis auf Auswahl des aktiven Knotens genau wie Push/Relabel-Variante

Schlussfolgerungen

- korrekt
- bis auf nichtsaturierende Push-Operationen Laufzeit $O(ne) = O(n^3)$

also nur nichtsaturierende Push-Operationen betrachten

Durchläufe in der FIFO-Variante

Definiere Durchlauf

1. Durchlauf alle Schritte für die Knoten, die initial in Qu kommen

i-ter Durchlauf alle Schritte für die Knoten, die im $(i-1)$-ten Durchlauf in Qu kommen

Beobachtung für jeden Knoten ≤ 1 nichtsaturierendes Push je Durchlauf
weil danach $e(v) = 0$
Wiedereinfügung danach wieder möglich
aber Behandlung frühestens im nächsten Durchlauf

genügt zu zeigen $O(n^2)$ Durchläufe

Anzahl der Durchläufe

Definiere Potenzialfunktion $P = P_1 - P_2$ mit

$$P_1 = 2 \sum_{v \in V} d(v)$$

$$P_2 = \max \{d(v) \mid v \text{ aktiv}\}$$

Beobachtung initial $P = 2n - 0 = 2n$

Beobachtung am Ende $P \leq 2 \cdot n(2n-1) < 4n^2$

genügt zu zeigen P wächst je Durchlauf um ≥ 1

Beobachtung Relabel(v) vergrößert $d(v)$ um $h \geq 1$
also P_1 wächst um $2h$, P_2 wächst um $\leq h$
zusammen P wächst um $h \geq 1$

Beobachung Push kann P_1 nicht ändern

aber Push kann P_2 ändern
durch Aktivieren/Deaktivieren von Knoten

Auswirkungen von Push auf P

haben Potenzialfunktion $P = P_1 - P_2$ mit

$$P_1 = 2 \sum_{v \in V} d(v),\ P_2 = \max \{d(v) \mid v \text{ aktiv}\}$$

Deaktivierung Falls Push v deaktiviert

wird P_2 kleiner und P wächst unkritisch ✓

Aktivierung Push(e) mit $e = (v, w)$

klar v ist schon aktiv

also nur w kann neu aktiv werden

klar $d(v) = d(w) + 1$, sonst Push(e) nicht anwendbar

also P_2 auch beim Aktivieren von w nicht größer ✓

insgesamt Durchlauf mit Relabel vergrößert P ✓

offen Durchlauf ohne Relabel-Aufruf

Durchläufe ohne Relabel-Aufruf

Beobachtung Durchlauf ohne Relabel
$\Leftrightarrow$ alle Repeat-Aufrufe mit $e(v) = 0$ beendet

also Qu enthält nur „neue“ Knoten (vgl. Z.11 in FIFO)
also Qu enthält nur Knoten w
mit Push$((v, w))$ in diesem Durchlauf
also Qu enthält nur Knoten w mit $d(w) = d(v) - 1$

also v wurde inaktiv und in Qu sind nur Knoten w mit $d(w) < d(v)$
also P_2 sinkt ≥ 1
also P wächst um ≥ 1

Damit ist die Laufzeit der FIFO Variante von $O(ne) = O(n^3)$ gezeigt. □

Dies. . .

beendet Flussalgorithmen