

Vorlesung

Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

Ingo Wegener

Was bisher geschah...

- starke Zusammenhangskomponenten ✓
- Matchings ✓
- Flussproblem
 - Algorithmus von Ford und Fulkerson:
 - $O(\max\{w(\Phi) \mid \Phi \text{ Fluss}\} \cdot (|V| + |E|))$ ✓
 - Algorithmus von Dinic
 - Lemma 4.8 (Veränderung von Distanzen)
 - Definition und Beispiel Niveaunetzwerk

Distanzen in Graphen

Betrachte gerichteten Graphen $G = (V, E)$, Knoten $Q \in V$.
Für $v \in V$ bezeichnet $d(v)$ die Länge (=Anzahl Kanten) eines kürzesten Weges von Q nach v .

Lemma 4.8

Gegeben $v, w \in V$, Distanzen d , nach Einfügen von (v, w)
Distanzen d^+ , nach Entfernen von (v, w) Distanzen d^- .

- 1 $(d(v) \geq d(w) - 1) \Rightarrow (\forall u \in V: d^+(u) = d(u))$
- 2 $\forall u \in V$ mit $d(u) \leq d(w) - 1: d^-(u) = d(u)$
- 3 $\forall u \in V: d^-(u) \geq d(u)$

Niveaunetzwerke

für kürzeste Q - S -Wege Betrachte Distanz $d(v)$ in Rest_Φ
klar $d(v) \geq d(S) \Rightarrow v$ nicht auf kürzestem Q - S -Weg

Definition 4.9

Betrachte $(G = (V, E), c)$ mit Fluss Φ , Rest_Φ , Distanzen d .

Für $0 \leq i < d(s)$ definiere i -tes Niveau

$$V_i := \begin{cases} \{v \in V \mid d(v) = i\} & \text{für } i < d(S), \\ \{S\} & \text{für } i = d(S). \end{cases}$$

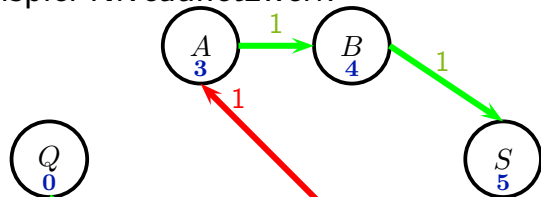
Definiere Niveaunetzwerk $N_\Phi = (V_\Phi, E_\Phi, r_\Phi)$ durch $V_\Phi := \bigcup_{i=0}^{d(S)} V_i$,

$E_i := \{(v, w) \in V_{i-1} \times V_i \mid (v, w) \in \text{Rest}_\Phi\}$, $E_\Phi := \bigcup_{i=1}^{d(S)} E_i$, und

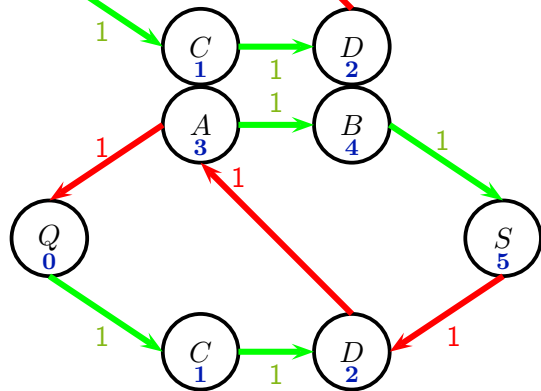
r_Φ wie in Rest_Φ .

klar Niveaunetzwerk berechenbar in Zeit $O(n + e)$

Beispiel Niveaunetzwerk



Niveaunetzwerk



Restgraph

Sperrflüsse

Wir wollen „möglichst viel Fluss“ auf einmal addieren . . .

Definition 4.10

Sei ψ ein Fluss **im Niveaunetzwerk** zum Fluss ϕ .

Eine Kante e heißt **saturiert** durch ψ , wenn $\psi(e) = r_\phi(e)$ gilt.

Der Fluss ψ heißt **Sperrfluss**, wenn auf jedem Q - S -Weg in N_Φ mindestens eine saturierte Kante liegt.

Analog zu **FVW** sieht man ein:

- Ein Sperrfluss ψ kann zum Fluss ϕ „addiert“ werden.
- Dabei wird zu Vorwärtskanten addiert und von Rückwärtskanten subtrahiert.
- Das Resultat ist ein Fluss ϕ' mit $w(\phi') > w(\phi)$.

Sperrflussberechnung

Algorithmus 4.11 (Sperrflussberechnung)

1. $\Psi := 0$
2. $i := 0; v_i := Q$
3. While $v_i \neq S$
4. If $\exists (v_i, w) \in N_\Phi$
5. Then $i := i + 1; v_i := w$
6. Else
7. Entferne v_i und Kanten (\cdot, v_i) aus N_Φ .
8. $i := i - 1$; If $i < 0$ Then Exit mit Ausgabe Ψ .
9. $r := \min\{r_\Phi((v_j, v_{j+1})) - \Psi((v_j, v_{j+1})) \mid 0 \leq j < i\}$
10. Für alle $j \in \{0, 1, \dots, i - 1\}$
11. $\Psi((v_j, v_{j+1})) := \Psi((v_j, v_{j+1})) + r$
12. If $\Psi((v_j, v_{j+1})) = r_\Phi((v_j, v_{j+1}))$
13. Then Entferne (v_j, v_{j+1}) aus N_Φ .
13. Weiter bei 2.

Zur Sperrflussberechnung

Lemma 4.12

Algorithmus 4.11 berechnet in Zeit $O(n \cdot e)$ einen Sperrfluss.

Beweis.

zur Korrektheit

- Algorithmus terminiert $\Leftrightarrow \nexists Q$ - S -Weg
- entfernte Knoten/Kanten saturiert oder in Sackgassen
- also am Ende **Sperrfluss**

zur Laufzeit

- bei **jeder** (erfolgreichen und erfolglosen) Pfadkonstruktion ≥ 1 Kante entfernt
- $\leq e$ Pfadkonstruktionen
- je Pfadkonstruktion Zeit $O(\text{Pfadlänge})$
- Länge jedes Q - S -Weges $\leq n$
- also insgesamt Zeit $O(n \cdot e)$



Der Algorithmus von Dinic

Algorithmus 4.13 (Algorithmus von Dinic)

1. $\Phi := 0$
2. Repeat
3. Berechne Niveaunetzwerk N_Φ .
4. Berechne Sperrfluss Ψ mit Algorithmus 4.11.
5. $\Phi := \Phi + \Psi$
6. Until $\Psi = 0$
7. Ausgabe Φ

Theorem 4.14

Der Algorithmus von Dinic berechnet in Zeit $O(n^2 \cdot e) = O(n^4)$ einen maximalen Fluss.

Zum Algorithmus von Dinic

Beobachtung Korrektheit offensichtlich ✓

Erinnerung Sperrflussberechnung in Zeit $O(n \cdot e)$

also genügt zu zeigen $O(n)$ Sperrflussberechnungen

Betrachte Fluss Φ und Nachfolger $\Phi' = \Phi + \Psi$ mit $\Psi \neq 0$

Betrachte Rest_{Φ} und $\text{Rest}_{\Phi'}$

Sei P kürzester Q - S -Weg in $\text{Rest}_{\Phi'}$

Behauptung P nicht kürzester Q - S -Weg in Rest_{Φ}

denn sonst in P eine Kante saturiert

$\rightsquigarrow P$ in $\text{Rest}_{\Phi'}$ nicht mehr vorhanden ✓

Welche Möglichkeiten gibt es?

1. Fall kürzeste Q - S -Wege in Rest_{Φ} kürzer als P

2. Fall Kante in P , die in Rest_{Φ} nicht vorhanden ist

Über Unterschiede von Rest_{Φ} und $\text{Rest}_{\Phi'}$

Betrachte kürzesten Q - S -Weg P in $\text{Rest}_{\Phi'}$
und Kante in P , die in Rest_{Φ} nicht vorhanden ist

Betrachte $e = (v, u)$ in $N_{\Phi'}$, die nicht in N_{Φ} ist

1. Fall in N_{Φ} Rückwärtskante (u, v) benutzt

2. Fall in N_{Φ} „normale Kante“ (u, v) benutzt

also in jedem Fall in N_{Φ} Kante (u, v) vorhanden

klar in N_{Φ} nur Kanten (u, v) mit $d(v) = d(u) + 1$

gedanklich füge (v, u) in N_{Φ} ein

klar Wege werden nicht kürzer (**Lemma 4.8**)

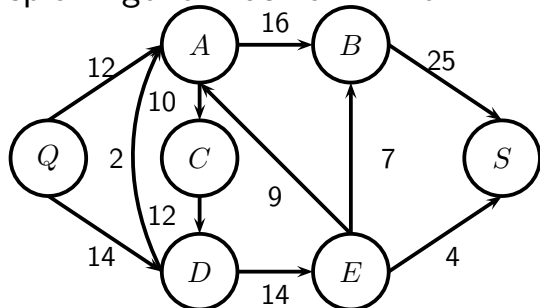
also kürzeste Q - S -Wege werden in jeder Runde länger

klar Länge kürzester Q - S -Wege $\in \{1, 2, \dots, n - 1\}$

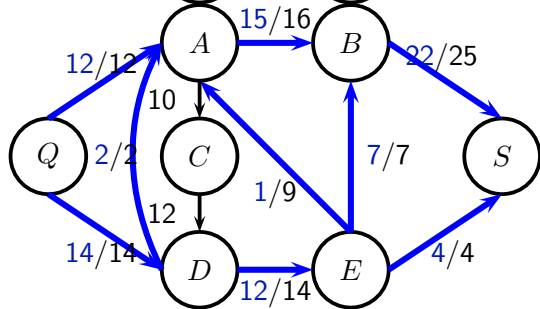
also $O(n)$ Runden



Beispiel Algorithmus von Dinic



Eingabe Netzwerk



Netzwerk mit max. Fluss

Schnellere Sperrflussberechnung

Zeit $O(n \cdot e)$ für nur eine Sperrflussberechnung ist **ziemlich lang**.

Kann man hier Zeit sparen?

Idee schrittweise Sperrflusskonstruktion

zentraler Begriff **Potenzial** eines Knotens

zum Restgraphen Rest_ϕ zum Fluss ϕ

bei aktuellem Fluss ψ im Niveaunetzwerk

$$\text{pot}(v) := \begin{cases} \min \left\{ \sum_{e=(v,\cdot)} r_\phi(e) - \psi(e), \sum_{e=(\cdot,v)} r_\phi(e) - \psi(e) \right\} & \text{für } v \notin \{Q, S\} \\ \sum_{e=(Q,\cdot)} r_\phi(e) - \psi(e) & \text{für } v = Q \\ \sum_{e=(\cdot,S)} r_\phi(e) - \psi(e) & \text{für } v = S \end{cases}$$

Idee zur Sperrflussberechnung

$\text{pot}(v)$ gibt an, wieviel Fluss höchstens durch v zusätzlich kann

Idee

- Wähle einen Knoten v .
- Treibe Fluss $\text{pot}(v)$ bis zur Senke S . (**Forward**)
- Treibe Fluss zurück bis zur Quelle Q . (**Backward**)

Fluss $v \rightsquigarrow S$ **Forward(v)**

1. Für alle $w \in V$ Überschuss[w] := 0
2. Überschuss[v] := pot[v]
3. $Qu := \emptyset$; $Qu.Enqueue(v)$
4. While $Qu \neq \emptyset$
5. $v := Qu.Dequeue()$
6. While Überschuss[v] > 0
7. Für alle $e = (v, w) \in E$ { * nur für Überschuss[v] > 0 * }
8. $\delta := \min\{r_{\Phi}(e) - \Psi(e), \text{Überschuss}[v]\}$
9. $\Psi(e) := \Psi(e) + \delta$
10. pot[v] := pot[v] - δ ; pot[w] := pot[w] + δ
11. If Überschuss[w] = 0 und $\delta > 0$ und $w \neq Q$
 Then $Qu.Enqueue(w)$
12. Überschuss[w] := Überschuss[w] + δ ;
 Überschuss[v] := Überschuss[v] - δ
13. If $\Psi(e) = r_{\Phi}(e)$ Then Entferne e aus E .

Fluss $v \rightsquigarrow Q$ **Backward(v, p_v)**

1. Für alle $w \in V$ Überschuss[w] := 0
2. Überschuss[v] := p_v
3. $Qu := \emptyset$; $Qu.Enqueue(v)$
4. While $Qu \neq \emptyset$
5. $v := Qu.Dequeue()$
6. While Überschuss[v] > 0
7. Für alle $e = (w, v) \in E$ { * nur für Überschuss[v] > 0 * }
8. $\delta := \min\{r_\Phi(e) - \Psi(e), \text{Überschuss}[v]\}$
9. $\Psi(e) := \Psi(e) + \delta$
10. $\text{pot}[v] := \text{pot}[v] - \delta$; $\text{pot}[w] := \text{pot}[w] + \delta$
11. If Überschuss[w] = 0 und $\delta > 0$ und $w \neq S$
12. Then $Qu.Enqueue(w)$
13. Überschuss[w] := Überschuss[w] + δ ;
 Überschuss[v] := Überschuss[v] - δ
14. If $\Psi(e) = r_\Phi(e)$ Then Entferne e aus E .

Schnellere Sperrflussberechnung

Algorithmus 4.16 (Forward-Backward-Propagation)

1. $\Psi := 0$
2. Für alle $v \in V$ berechne $\text{pot}[v]$.
3. While $\{Q, S\} \subseteq V$
4. Wähle Knoten $v \in V$ mit minimalem Potenzial; $p_v := \text{pot}[v]$
5. If $\text{pot}[v] > 0$ Then Forward(v); Backward(v, p_v)
6. Entferne v aus V und Kanten $(\cdot, v), (v, \cdot)$ aus E .

Lemma 4.17

Algorithmus 4.16 berechnet in Zeit $O(n^2)$ einen Sperrfluss in N_Φ .

Korrektheit von Forward-Backward-Propagation

Beobachtung $p_v = \min \{ \text{pot}(v) \mid v \in V \}$
 also gesamter Überschuss
 zur Senke oder Quelle transportierbar

Beobachtung nachher alle eingehenden oder
 alle ausgehenden Kanten von v saturiert
 also v entfernbar

Beobachtung wenn Q oder S so saturiert, fertig ✓

schwieriger Laufzeitschranke $O(n^2)$

Laufzeit Sperrflussberechnung

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

Beobachtung Analyse Forward genügt
Backward analog

Beobachtung nur v mit $\text{Überschuss}(v) = 0$
kommen in die Queue

Beobachtung nur für v selbst sinkt Überschuss

also also kein Knoten in einem Durchlauf mehrfach in Queue

also je Forward-Durchlauf $O(n)$ Knoten

es fehlt noch Schleife über Kanten in Forward

Forward: Die innere Schleife

Beobachtung für jede betrachtete Kante $e = (v, w)$ gilt
entweder wird e saturiert oder Schleife endet

klar saturierte Kanten werden entfernt

Erinnerung v nachher aus Netzwerk entfernt

also jede Kante ≤ 2 Mal betrachtet

außerdem $\leq n$ Forward-Aufrufe

zusammen Laufzeit $O(n \cdot n + e) = O(n^2)$



Algo. von Malhotra, Pramodh Kumar & Maheshwari

Algorithmus 4.18 (Algorithmus von Malhotra, Pramodh Kumar & Maheshwari)

1. $\Phi := 0$
2. Repeat
3. Berechne Niveaunetzwerk N_Φ .
4. Berechne Sperrfluss Ψ mit Algorithmus 4.16.
5. $\Phi := \Phi + \Psi$
6. Until $\Psi = 0$
7. Ausgabe Φ

Beobachtung wie Dinic mit anderer Sperrflussberechnung

Über den Algorithmus von Malhotra et. al

Theorem 4.19

Der Algorithmus von Malhotra, Pramodh Kumar und Maheshwari berechnet einen maximalen Fluss in Zeit $O(n^3)$.

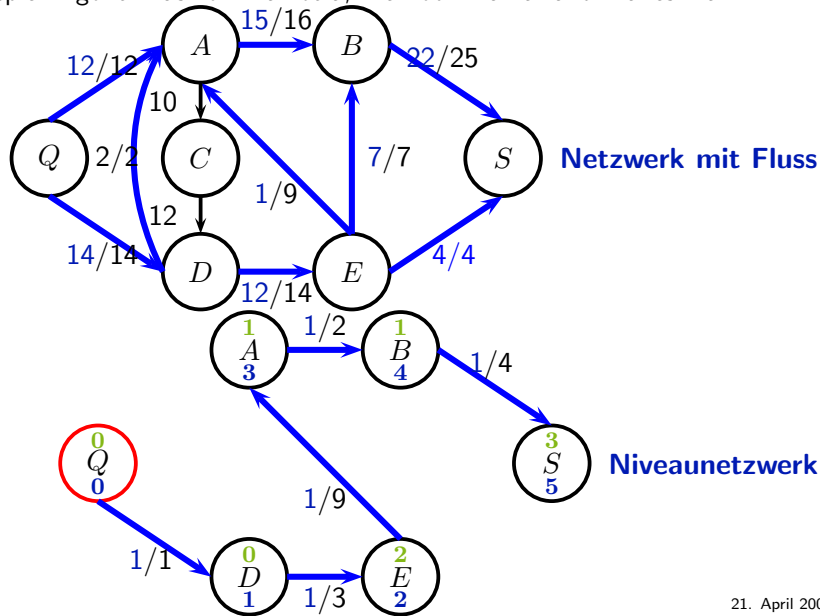
Korrektheit wie bei Dinic

Anzahl Sperrflussberechnungen wie bei Dinic

also Gesamtlaufzeit $O(n \cdot n^2) = O(n^3)$



Beispiel Algorithmus von Malhotra, Pramodh Kumar und Maheshwari



Rückblick

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 oder Quelle

Beobachtung so lange Knoten mit Überschuss existiert
haben wir **keinen Fluss**

Definition 4.20

Für Netzwerk $(G = (V, E), C)$ heißt $\Phi: E \rightarrow \mathbb{R}_0^+$ **Präfluss**, wenn

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

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

also jeder Fluss ist auch Präfluss

Was ist mit unseren Begriffen?

Beobachtung Restgraph auch für Präfluss sinnvoll

klar Ziel Umwandlung Präfluss \rightsquigarrow Fluss

Geht das überhaupt?

Erinnerung bei M./P. K./M. Überschuss zur Not \rightsquigarrow Quelle

Geht das denn hier immer?

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 $x \rightsquigarrow_\Phi y$
 $:\Leftrightarrow \exists \text{Weg von } x \text{ zu } y \text{ in } \text{Rest}_\Phi$

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

zu zeigen $Q \in V^*$

Annahme $Q \notin V^*$

Die Summe der Überschüsse

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^*

klar es gibt nicht 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 nicht Weg von v nach v'

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

aber $(v', v^*) \in \text{Rest}_\Phi$

also $\Phi(e) = 0$

Zusammenfassung Summe der Überschüsse

Wir haben

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

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 **Annahme**

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

aber $v \in V^*$ mit $e(v) > 0$ **Widerspruch** □

also Überschuss jedenfalls zur Quelle bringbar

Sinnvolle Richtungen für den Überschuss

klar Überschuss zur Quelle bringen **unproduktiv**

Erinnerung bei M./P. K./M. Überschuss **erst** zur Senke

Einsicht **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.

$d: V \rightarrow \mathbb{N}_0$ heißt **gültige Knotenmarkierung**, wenn

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

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

Über gültige Knotenmarkierungen

Findet man 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

klar kürzeste Wege haben Länge $< n$

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

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

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 $(v, v_1), (v_1, v_2), (v_2, v_3), \dots, (v_{l-1}, w)$
mit $v = v_0, v_l = w$ hat Länge l

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

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

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 $\exists v \in V$ 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 *}

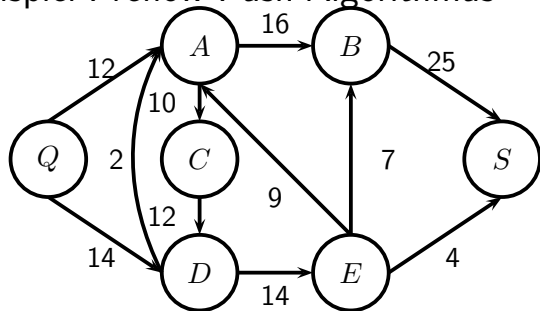
1. $\delta := \min\{e(v), r_{\Phi}(e)\}$
2. If $e \in E$
 Then $\Phi(e) := \Phi(e) + \delta$
 Else $\Phi(e) := \Phi(e) - \delta$ { * Rückwärtskante * }
 $e(v) := e(v) - \delta$; $e(w) := e(w) + \delta$

Relabel(v)

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

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

Beispiel Preflow-Push-Algorithmus



Eingabe Netzwerk

1

Über das Beispiel

Anmerkungen

- sah nicht besonders schnell aus
- 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 Korrektheits**beweis**.

Einsicht Wir brauchen eine Laufzeit**analyse**.

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 $\text{Relabel}(v)$ anwendbar ist, erhöht Anwendung von $\text{Relabel}(v)$
 $d(v)$ um ≥ 1 .

Beweis.

klar anwendbar $\Rightarrow v$ aktiv

also $\exists v$ - Q -Weg in Rest_Φ (Lemma 4.23)

also $\exists(v, w) \in E_\Phi$

also $\min\{\cdot\}$ wohldefiniert

Beweis von Lemma 4.25

Wir haben $\min\{\cdot\}$ über nichtleere Menge

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

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

also $e \notin E_\Phi$ ✓

Auswirkung von $\text{Push}(v, w)$ auf d

Voraussetzung $\text{Push}(v, w)$ anwendbar

1. Fall saturierendes Push

dann e wird aus Rest_Φ entfernt

klar das entfernt Bedingung, schafft keine neue ✓

möglich $\text{rev}(e) = (w, v)$ wird erzeugt in Rest_Φ

Erinnerung $\text{Push}(v, w)$ anwendbar

also $d(v) = d(w) + 1$

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

2. Fall nichtsaturierendes Push

Beobachtung wie saturierendes Push **ohne** entfernte Kante

also d auch weiterhin gültig ✓

Auswirkungen von $\text{Relabel}(v)$ auf d

Voraussetzung $\text{Relabel}(v)$ anwendbar

Beobachtung **kein Problem** für $(v, w) \in E_{\Phi}$
weil $d(v)$ explizit korrekt gesetzt

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

also d weiterhin gültig

