

Vorlesung

# Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

Ingo Wegener

## 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 $_{\Phi}$**  unverändert für Präfluss  $\Phi$  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  $\text{Rest}_{\Phi}$  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  $\exists v \in V$  mit  $e(v) > 0$
6.     Führe anwendbare Basisoperation (Push oder Relabel) aus.
7. Ausgabe  $\Phi$

# 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,  $\Phi$  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  $\geq 1$ .

## 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  $\Phi$  ein maximaler Fluss.

**Beweis.**

**klar** Algorithmus stoppt  $\Leftrightarrow$  kein Knoten aktiv

**Beobachtung**  $\Phi$  Präfluss und kein Knoten aktiv  
 $\Leftrightarrow \Phi$  Fluss

**Erinnerung** Knotenmarkierung gültig (Lemma 4.27)

**also** kein  $Q$ - $S$ -Weg in  $\text{Rest}_{\Phi}$  (Lemma 4.23)

**also**  $\Phi$  maximal



# Auf dem Weg zum Korrektheitsbeweis

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

dazu Einsichten in die Entwicklung der gültigen Knotenmarkierung  $d$

## Lemma 4.29

Im Ablauf des Algorithmus von Goldberg und Tarjan (Algorithmus 4.24) gilt

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

## Beweis von Lemma 4.29

**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  $\text{Rest}_\Phi$

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

klar Länge des Weges  $\leq n - 1$

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

darum  $d(v') - d(Q) = d(v') - n \leq n - 2$  (Lemma 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  $\geq 0$
- Relabel( $v$ ) vergrößert  $d(v)$  um  $\geq 1$  (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 nichtsaturierender Push-Operationen  
saturierende Pushes **offensichtlich produktiv**  
nichtsaturierende Pushes **weniger offensichtlich hilfreich**

## Lemma 4.31

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

**Beweis.**

**Betrachte** saturierendes  $\text{Push}(e)$  mit  $e = (v, w)$

**Betrachte** nächstes saturierendes  $\text{Push}(e)$  mit  $e = (v, w)$

**Behauptung** dazwischen liegt  $\text{Push}(\text{rev}(e))$

**Begründung** nach saturierendem  $\text{Push}(e)$   $e$  nicht mehr in  $E_\Phi$

**klar** Einfügung passiert nur nach  $\text{Push}(\text{rev}(e))$

# Konsequente saturierende Push-Operationen

**wir haben** zwischen zwei saturierenden Push( $e$ )  
liegt ein Push( $\text{rev}(e)$ )

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

**Beobachtung** bei Push( $\text{rev}(e)$ )  
gilt  $d(w) = d(v) + 1$

**klar**  $d(w)$  muss um  $\geq 2$  gewachsen sein

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

## Beweis von Lemma 4.31

**Wir haben** bei zwei konsekutive Push( $e$ ) mit  $e = (v, w)$   
wächst  $d(v) + d(w) \geq 4$

**Beobachtung**  $d(v) + d(w) \leq 4n - 4$  vor letztem saturierendem Push( $e$ )  
weil  $d(v), d(w) \leq 2n - 1$  (Lemma 4.29)

**also** insgesamt  $\leq (4n - 4)/4 = n - 1 < n$  saturierende Push( $e$ )

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

**also**  $< 2ne$  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.

**konkret** Betrachte **Potenzialfunktion**  $P = P_1 + P_2 - P_3$  mit

$$P_1 = (2n - 2) \cdot \# \text{saturierende Push-Operationen}$$

$$P_2 = \sum_{v \in V} d(v)$$

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

# Analyse der Potenzialfunktion $P$

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

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

Betrachte  $\text{Push}(e)$  mit  $e = (v, w)$

klar  $v$  aktiv,  $d(v) = d(w) + 1$ , weil  $\text{Push}(e)$  anwendbar

1. Fall  $\text{Push}(e)$  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  $\geq 1$

# Analyse der Potenzialfunktion $P$ (Fortsetzung)

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

## 2. Fall Push( $e$ ) saturierend

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  $d(w) \leq 2n - 2$

**insgesamt**  $P$  kann nicht kleiner werden

## Analyse der Potenzialfunktion $P$ (Relabel)

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

$$P_1 = (2n - 2) \cdot \#\text{saturierende Push-Operationen}$$

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

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  $\geq 1$  bei jedem nichtsaturierenden Push

## Anzahl nichtsaturierender Push-Operationen beschränken

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

$$P_1 = (2n - 2) \cdot \#\text{saturierende Push-Operationen}$$

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

$$P \geq n + \#\text{nichtsaturierender Push-Operationen}$$

## am Ende des Algorithmus

- $P_1 \leq (2n - 2) \cdot 2ne$ , 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)
- $P_3 = 0$ , weil kein Knoten mehr aktiv

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

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 Zeit $O(n^2e)$ erreichbar?

ja Beweis konstruktiv

## Die Push/Relabel-Variante (Algorithmus 4.34)

1. Für alle  $v \in V$   $d(v) := 0$ ;  $e(v) := 0$ ; 1. Kante  $(v, w)$  ist aktiv
2.  $d(Q) := n$ ;  $\Phi := 0$
3. Für alle  $v \in V$  mit  $e = (Q, v) \in E$   
 $\Phi(e) := c(e)$ ;  $e(v) := c(e)$
4. While  $\exists v \in V$  mit  $e(v) > 0$
5.     Führe Push/Relabel( $v$ ) aus.
6. Ausgabe  $\Phi$

### Push/Relabel( $v$ )

{\* *anwendbar, wenn  $v$  aktiv ist* \*}

1. If aktive Kante  $e = (v, w)$  wählbar Then Push( $e$ )
2. Else
3.     Mache die nächste Kante aktiv.
4.     If Listenende erreicht Then
5.         Relabel( $v$ )
6.     Mache erste Kante der Adjazenzliste von  $v$  zur aktiven Kante.

# Korrektheit der Push/Relabel-Variante?

Was müssen für die Korrektheit nachweisen?

klar Relabel-Operationen müssen anwendbar sein

## Lemma 4.35

Wird im Ablauf von Algorithmus 4.34  $\text{Relabel}(v)$  aufgerufen, dann ist  $\text{Relabel}(v)$  anwendbar.

Beweis.

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

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

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

klar  $\text{Relabel}(v)$  nur ausgeführt, wenn Push nicht anwendbar

also  $e(v) > 0$  beim Aufruf von  $\text{Relabel}(v)$  ✓

## Beweis von Lemma 4.35

noch zu zeigen    beim Aufruf von  $\text{Relabel}(v)$   
keine Kante  $(v, w)$  wählbar

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

Beobachtung     $\text{Relabel}(v')$  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  $E_\Phi$  einfügen!

zu zeigen    keine wählbare Kante  $(v, v')$  wird eingefügt

Betrachte     $\text{Push}(e')$  mit  $e' = (x, y)$

klar     $e'$  wählbar, also  $d(x) = d(y) + 1$

klar    neue Kante kann nur  $\text{rev}(e') = (y, x)$  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 Zeit  $O(e)$

also Gesamtzeit Relabel  $O(n^2e)$  ✓

Anmerkung geht besser

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

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 Basisoperation schon dabei gezählt ✓

**offen** Schritte ohne Basisoperation

**Beobachtung** jede Kante nur einmal, dann Relabel

**also** nur Zeit  $O(n \cdot e)$ , weil  $\leq 2n - 1$  Relabel pro Kante/Knoten  
(Vorsicht: im Skript zu schwache Abschätzung)

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

**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  $v \in V$   
 $d(v) := 0; e(v) := 0$ ; 1. Kante  $(v, w)$  ist aktiv
2.  $d(Q) := n$
3.  $\Phi := 0; Qu := \emptyset$
4. Für alle  $v \in V$  mit  $e = (Q, v) \in E$   
 $\Phi(e) := c(e); e(v) := c(e); Qu.Enqueue(v)$
5. While  $Q \neq \emptyset$
6.      $v := Qu.Dequeue()$
7.     Repeat
8.         Push/Relabel( $v$ ), füge dabei aktiv werdende Knoten in  $Qu$  ein.
9.     Until  $e(v) = 0$  oder Relabel( $v$ ) aufgerufen wurde.
10.    If  $e(v) > 0$  Then  $Qu.Enqueue(v)$
11. Ausgabe  $\Phi$

# 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  $\leq 1$  nichtsaturierendes Push  
je Durchlauf

**weil** danach  $e(v) = 0$ , Wiedereinfügung  
 $\rightsquigarrow$  frühestens nächster 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  $\geq 1$

**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  $h \geq 1$

**Beobachtung** Push kann  $P_1$  nicht ändern

**aber** Push kann  $P_2$  ändern  
 durch Aktivieren/Deaktivieren von Knoten

# Auswirkungen von Push auf $P$

**Beobachtung**    Push deaktiviert Knoten  
 $\rightsquigarrow P_2$  wird kleiner  $\rightsquigarrow P$  wächst  
 unkritisch ✓

**Betrachte**    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 Puhs( $e$ ) 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-Durchläufe mit  $e(v) = 0$  beendet

also  $Qu$  enthält nur „neue“ Knoten

also  $Qu$  enthält nur Knoten  $w$   
 mit  $\text{Push}((v, w))$  in diesem Durchlauf

also  $Qu$  enthält nur Knoten  $w$   
 mit  $d(v) = d(w) + 1, d(w) < P_2$

also  $P_2$  sinkt  $\geq 1$   
 $\rightsquigarrow P$  wächst um  $\geq 1$

