Karlsruher Institut für Technologie

Maximale s–t-Flüsse in Planaren Graphen

Vorlesung „Algorithmen für planare Graphen“ · 6. Juni 2017

Guido Brückner

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Organisatorisches

mündliche **Prüfungstermine**:

- 24. Juli
- 10. August
- 21. August
- 13. September
- 21. September
- 25. September
- 5. Oktober
- 6. Oktober

Anmeldung:
per e-Mail an das Sekretariat `sekr-wagner@ira.uka.de` nach „first come, first served“-Prinzip

Organisatorisches

mündliche **Prüfungstermine**:

- 24. Juli
- 10. August
- 21. August
- 13. September
- 21. September
- 25. September
- 5. Oktober
- 6. Oktober

Anmeldung:
per e-Mail an das Sekretariat `sekr-wagner@ira.uka.de` nach „first come, first served“-Prinzip

Wichtig: An- und Abmeldung müssen bis spätestens 3 Wochen vor dem Prüfungstermin erfolgen!

Maximales Flussproblem

Gegeben:

- Gerichteter Graph $G = (V, E)$, rev$(u \to v) := (v \to u)$
- $c\colon E \to \mathbb{R}$, (OE rev$(e) \in E \quad \forall e \in E$)
- $s, t \in V$

Gesucht:

Zulässiger Fluss ϕ mit maximalem $\sum_{s \to v \in E} \phi(s \to v)$.

Maximales Flussproblem

Gegeben:

- Gerichteter Graph $G = (V, E)$, rev$(u \to v) := (v \to u)$
- $c\colon E \to \mathbb{R}$, (OE rev$(e) \in E \quad \forall e \in E$)
- $s, t \in V$

Gesucht:

Zulässiger Fluss ϕ mit maximalem $\sum_{s \to v \in E} \phi(s \to v)$.

$\phi\colon E \to \mathbb{R}$ **st-Fluß**:

- $\phi(e) = -\phi(\text{rev}(e))$ (**Symmetrie**)
- $\phi(e) \leq c(e)$ (**Zulässigkeit**)
- $\forall v \in V \setminus \{s, t\} : \sum_w \phi(v \to w) = 0$ (**Flusserhaltung**)

Maximales Flussproblem

Gegeben:

- Gerichteter Graph $G = (V, E)$, rev$(u \to v) := (v \to u)$
- $c\colon E \to \mathbb{R}$, (OE rev$(e) \in E \quad \forall e \in E$)
- $s, t \in V$

Gesucht:
Zulässiger Fluss ϕ mit maximalem $\sum_{s \to v \in E} \phi(s \to v)$.

$\exists$ viele verschiedene Lösungsalgorithmen, deren Laufzeit aber mindestens $\mathcal{O}(n^2)$ ist.

Heute: ein Algorithmus für planare Graphen mit $\mathcal{O}(n \log n)$ Laufzeit.

Inhalt

- **Teil I:** maximaler Fluss in st-planaren Graphen
- **Teil II:** maximaler Fluss in allgemeinen planaren Graphen

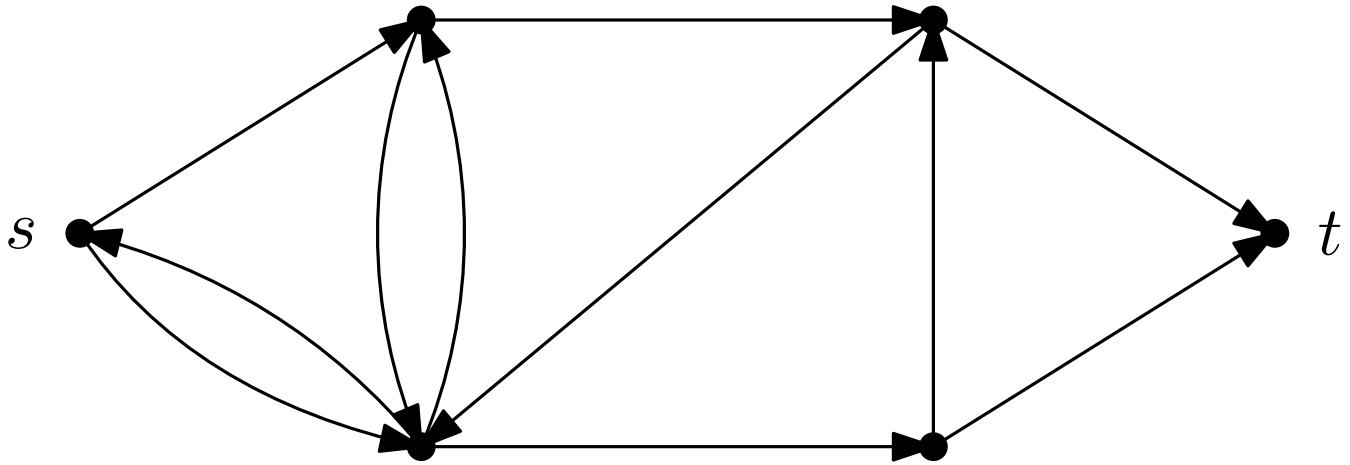
Maximaler Fluss in st-planaren Graphen

Graph $G = (V, E)$ mit $s, t \in V$ ist **st-planar** $\iff$ G lässt sich so einbetten, dass s, t inzident zur äußeren Facette

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Maximaler Fluss in st-planaren Graphen

Graph $G = (V, E)$ mit $s, t \in V$ ist **st-planar** $\iff$ G lässt sich so einbetten, dass s, t inzident zur äußeren Facette

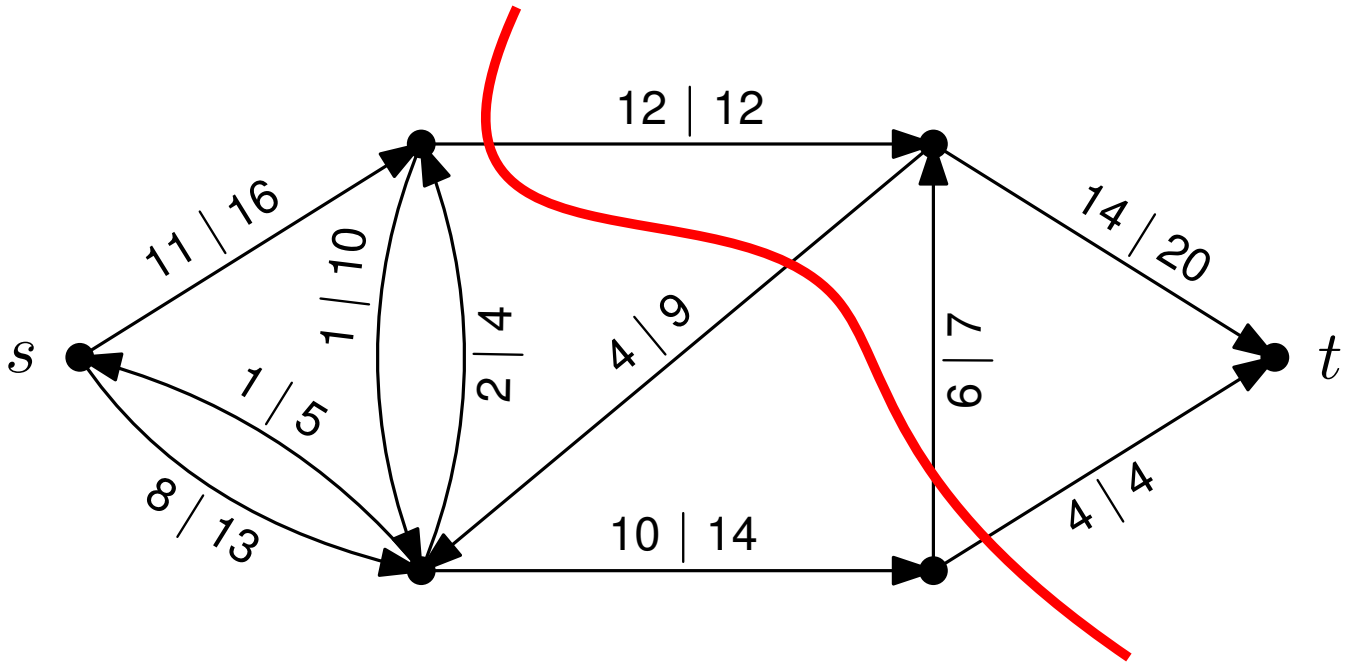


Maximaler Fluss in st-planaren Graphen

Lemma

Sei ϕ ein st-Fluss und $E(S, V \setminus S)$ ein st-Schnitt. Dann gilt:

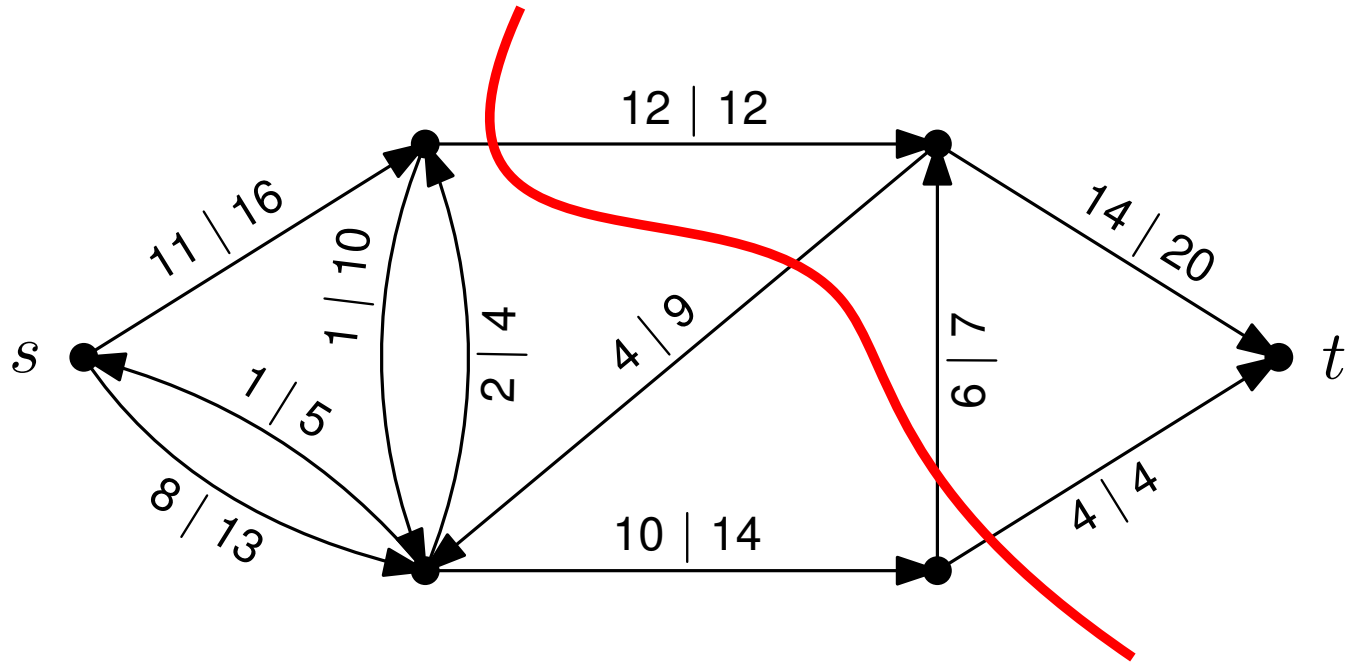
$$w(\phi) = \sum_{e \in E(S, V \setminus S)} \phi(e) - \sum_{e \in E(V \setminus S, S)} \phi(e) \leq c(S, V \setminus S)$$



Maximaler Fluss in st-planaren Graphen

Korollar

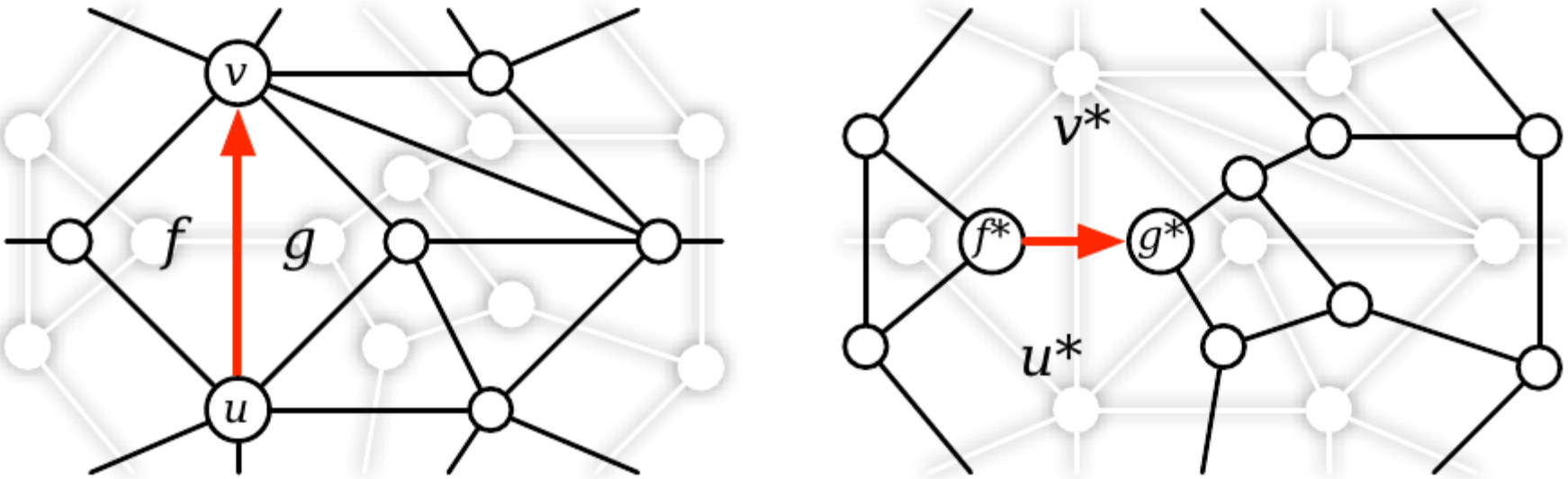
Ein st-Fluss ϕ mit $w(\phi) = c(S, V \setminus S)$ ist maximal.



Dual-Graphen, gerichtet(!)

$G = (V, E)$ gerichtet

Dualgraph: Wie gehabt, $(u \to v)^\star$ kreuzt $u \to v$ von links nach rechts!

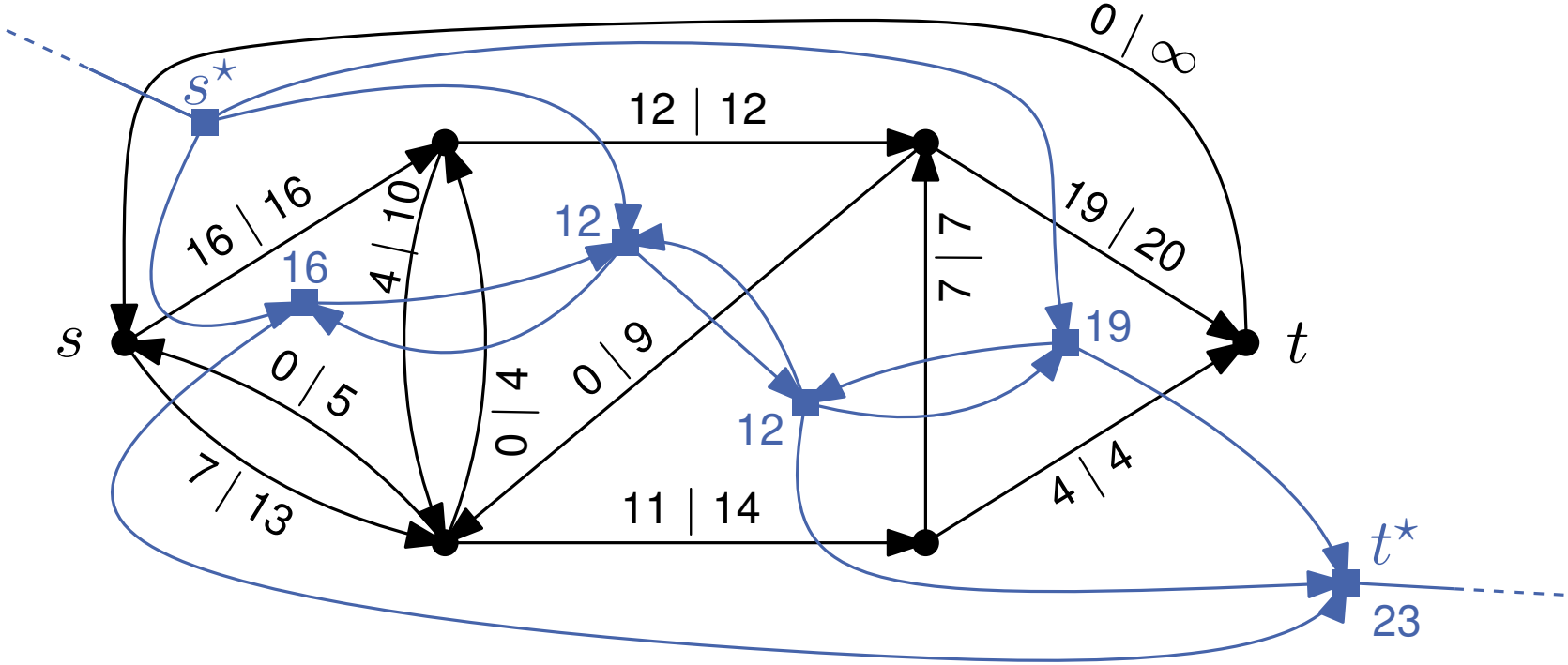


[Erickson'10]

Maximaler Fluss in st-planaren Graphen

Idee:

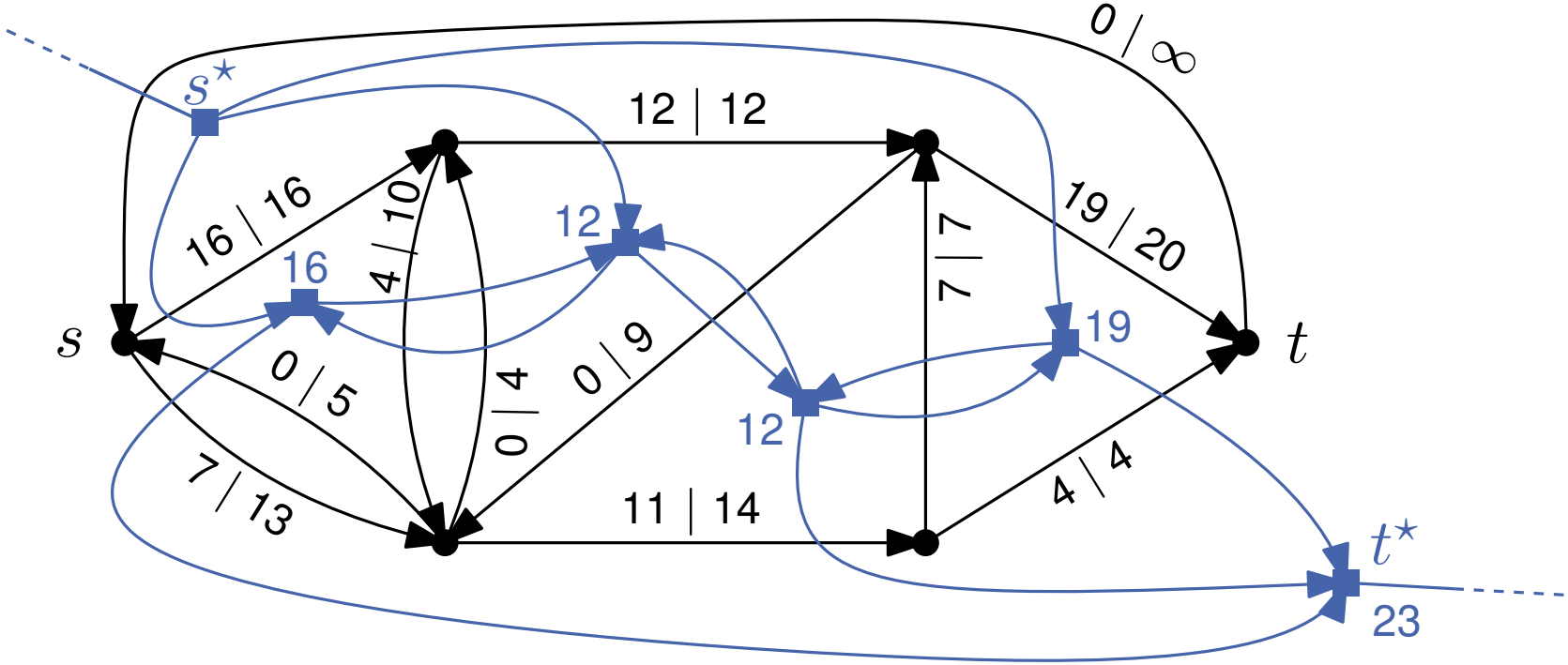
- füge Rückkante $t \to s$ ein
- konstruiere gerichteten Dualgraph $G^\star$, setze $\ell(e^\star) \coloneqq c(e)$
- suche kürzesten $s^\star$-$t^\star$-Pfad



Maximaler Fluss in st-planaren Graphen

Lemma

Die Funktion $\phi(e) \coloneqq \max(0, d(s^\star, \text{right}(e)) - d(s^\star, \text{left}(e)))$ beschreibt einen maximalen st-Fluss.



Maximaler Fluss in st-planaren Graphen

Algorithmus:

- konstruiere Dualgraph $\mathcal{O}(n)$
- finde kürzesten st-Pfad
  - z.B. mit Dijkstra $\mathcal{O}(n \log n)$
  - aber für planare Graphen schneller $\mathcal{O}(n)$

  (Henzinger et al., 1997)

Theorem

Für st-planare Graphen lässt sich ein maximaler Fluss in Linearzeit berechnen.

Inhalt

- **Teil I:** maximaler Fluss in st-planaren Graphen
- **Teil II:** maximaler Fluss in allgemeinen planaren Graphen

Parametrische Kürzeste Wege

Parametrisches Kürzeste-Wege-Problem:

Gegeben: Gerichteter Graph $G = (V, E)$, $E' \subseteq E$, $c\colon E \to \mathbb{R}$
$c(\lambda, e) = c(e) - \lambda$ für $e \in E'$, $c(\lambda, e) = c(e)$ für $e \notin E'$
Gesucht: Größtes λ mit G enthält bzgl. $c(\lambda, \cdot)$ keine negativen Kreise.

Lösbar in Zeit $O(nm \log n)$ [Karp, Orlin] bzw. $O(n^2 \log n)$ [Young et al.]

Parametrische Kürzeste Wege

Karlsruher Institut für Technologie

Parametrisches Kürzeste-Wege-Problem:

Gegeben: Gerichteter Graph $G = (V, E)$, $E' \subseteq E$, $c\colon E \to \mathbb{R}$
$c(\lambda, e) = c(e) - \lambda$ für $e \in E'$, $c(\lambda, e) = c(e)$ für $e \notin E'$
Gesucht: Größtes λ mit G enthält bzgl. $c(\lambda, \cdot)$ keine negativen Kreise.

Lösbar in Zeit $O(nm \log n)$ [Karp, Orlin] bzw. $O(n^2 \log n)$ [Young et al.]

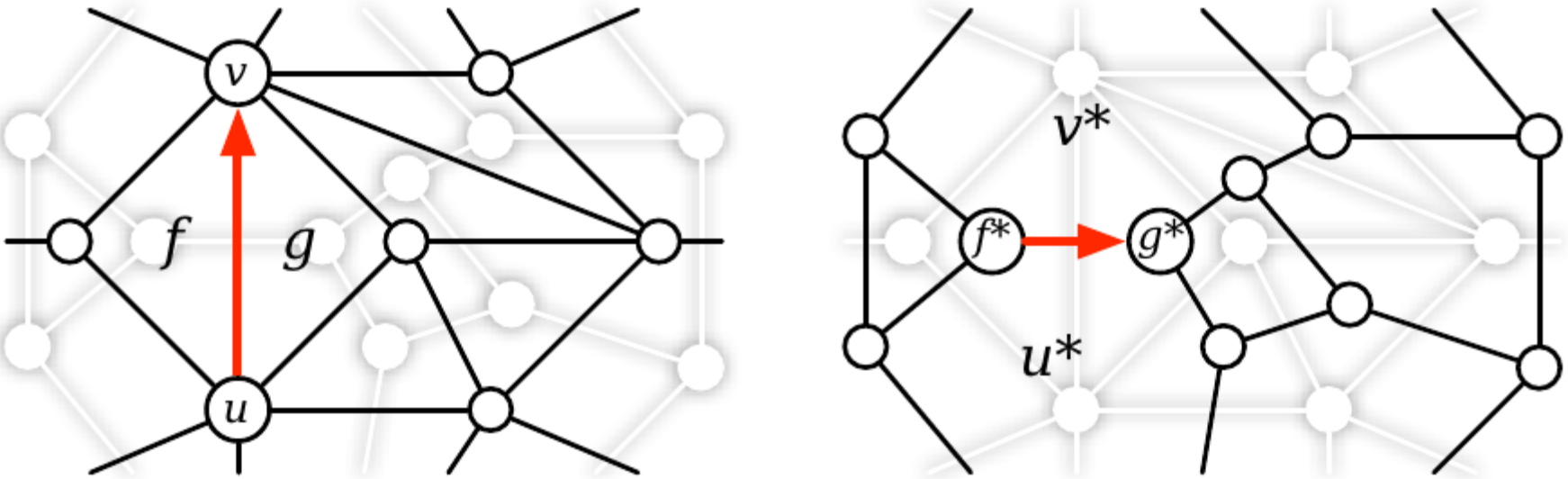
Idee:

- Betrachte kürzeste-Wege-Baum bzgl. $c(\lambda, \cdot)$ bei bel. Startknoten s.
- Erhöhe λ schrittweise.
- $\text{dist}(\lambda, v) \leq \text{dist}(\lambda, u) + c(\lambda, u \to v)$
- Kritische Stelle bei Gleichheit,
  $u \to v$ kommt in den Baum, ersetzt $u' \to v$.
- Jeder Pivot-Schritt erhöht Anzahl Kanten in E' auf einem $s-v$-Pfad
  $\rightsquigarrow O(n^2)$ Pivot-Schritte

Dual-Graphen, gerichtet(!)

$G = (V, E)$ gerichtet

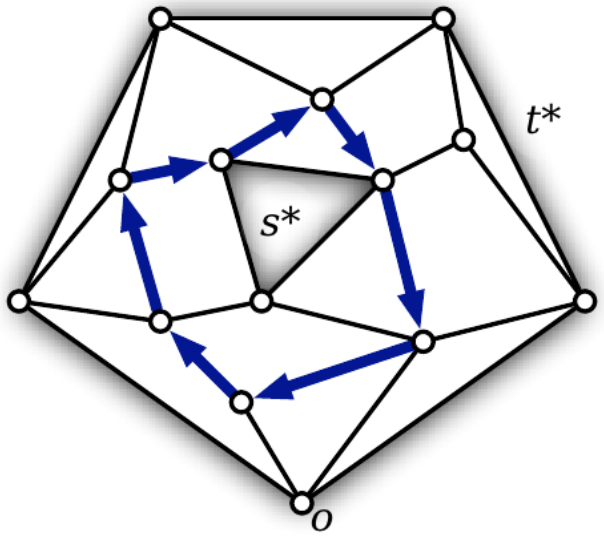
Dualgraph: Wie gehabt, $(u \to v)^\star$ kreuzt $u \to v$ von links nach rechts!

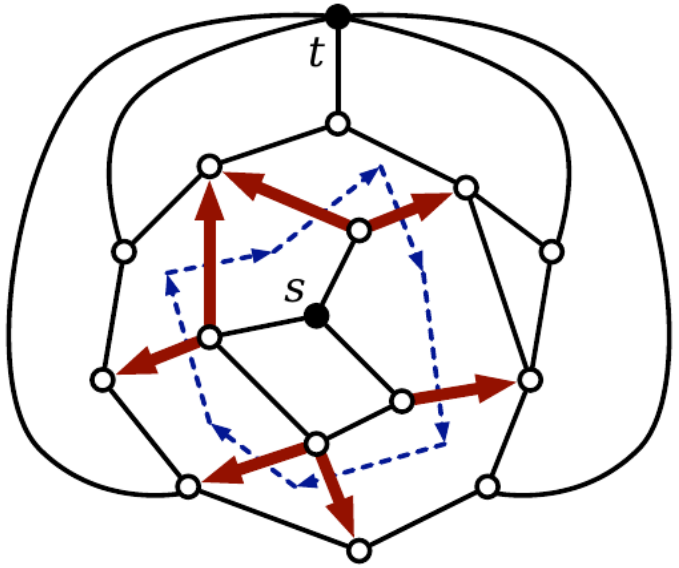


[Erickson'10]

Gerichteter Dual-Graph und Gerichtete Schnitte

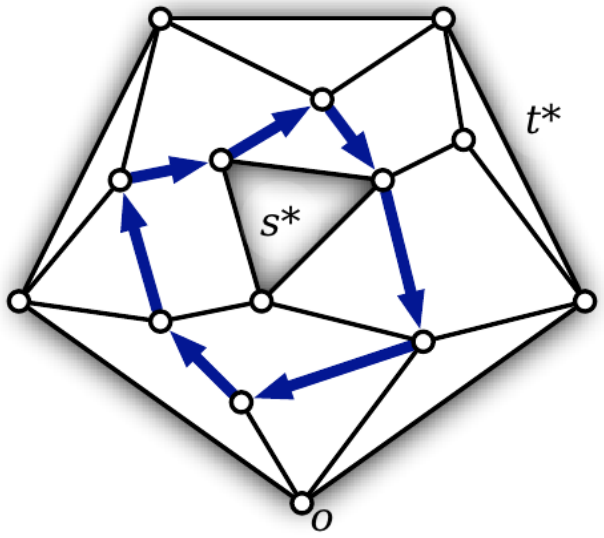
Gerichteter (s, t)-Schnitt:
Kantenmenge C, sodass jeder gerichtete st-Pfad Kante aus C enthält.





Gerichteter Dual-Graph und Gerichtete Schnitte

Gerichteter (s,t)-Schnitt:
Kantenmenge C, sodass jeder gerichtete st-Pfad Kante aus C enthält.

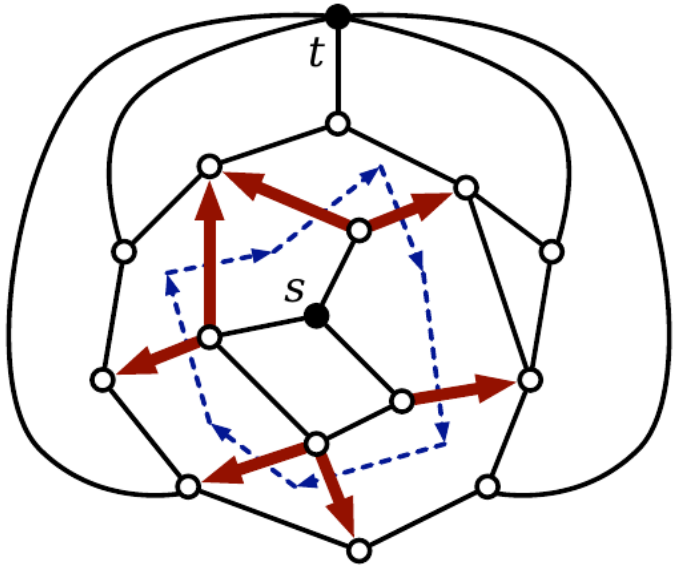


Dualität:

Gerichteter (s,t)-Schnitt in G

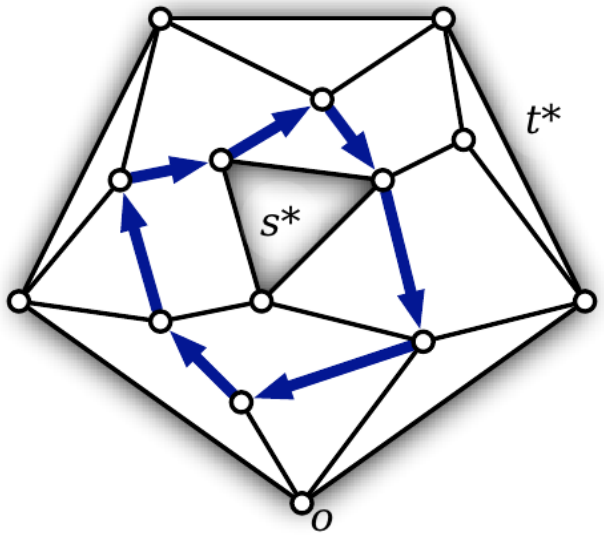
$\equiv$

gerichteter Kreis in $G^\star$, der $s^\star$ und $t^\star$ trennt.



Gerichteter Dual-Graph und Gerichtete Schnitte

Gerichteter (s,t)-Schnitt:
Kantenmenge C, sodass jeder gerichtete st-Pfad Kante aus C enthält.

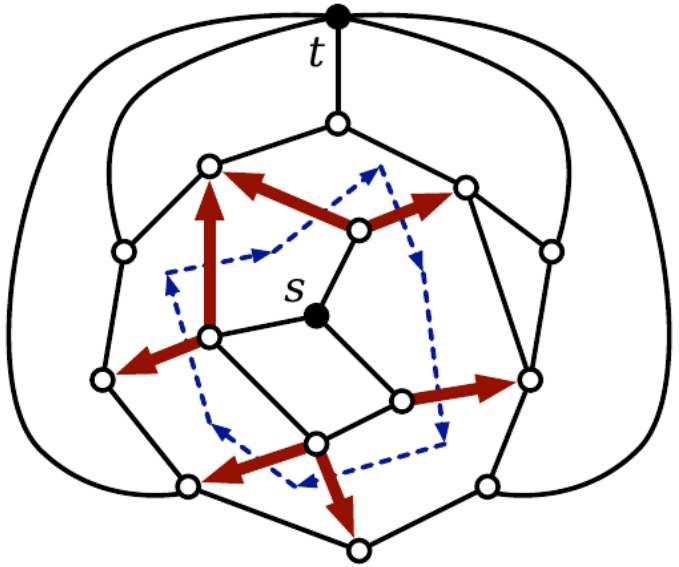


Dualität:

Gerichteter (s,t)-Schnitt in G

$\equiv$

gerichteter Kreis in $G^\star$, der $s^\star$ und $t^\star$ trennt.



Kantenmenge $C \subseteq E$ **Kozykel**

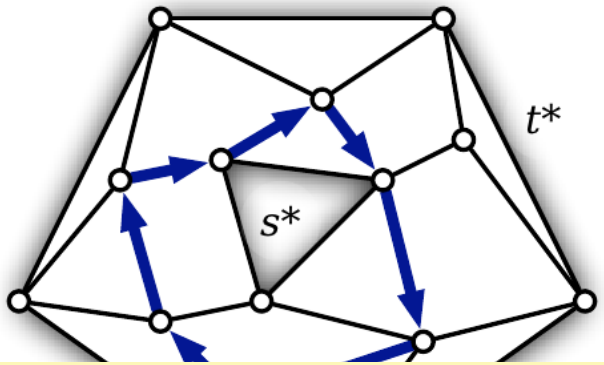
$:\Leftrightarrow$

$C^\star$ einfacher gerichter Kreis

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Gerichteter Dual-Graph und Gerichtete Schnitte

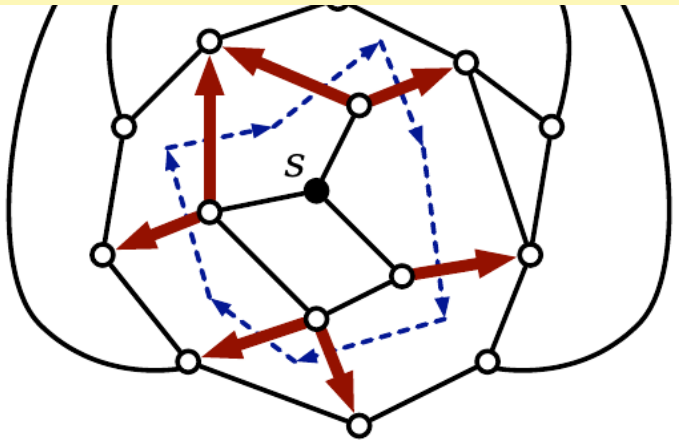
Gerichteter (s,t)-Schnitt:
Kantenmenge C, sodass jeder gerichtete st-Pfad Kante aus C enthält.



Idee: ähnlich wie bei st-planaren Graphen, aber *kürzeste Kreise* statt *kürzester Pfade*

Aber: wie kürzeste Kreise effizient finden?

Kantenmenge $C \subseteq E$ **Kozykel**
$:\Leftrightarrow$
$C^\star$ einfacher gerichter Kreis



Kozykel und st-Schnitte

P beliebiger gerichteter Pfad von s nach t.
Definiere $\pi : E \to \mathbb{R}$, **Einheitsfluß** auf P

$$\pi(e) := \begin{cases} 1 & e \in P \\ -1 & \text{rev}(e) \in P \\ 0 & \text{sonst} \end{cases}$$

Für $E' \subseteq E$ definiere: $\pi(E') := \sum_{e \in E'} \pi(e)$

Kozykel und st-Schnitte

P beliebiger gerichteter Pfad von s nach t.
Definiere $\pi : E \to \mathbb{R}$, **Einheitsfluß** auf P

$$\pi(e) := \begin{cases} 1 & e \in P \\ -1 & \text{rev}(e) \in P \\ 0 & \text{sonst} \end{cases}$$

Für $E' \subseteq E$ definiere: $\pi(E') := \sum_{e \in E'} \pi(e)$

Lemma

Für jeden Kozykel C gilt: $\pi(C) \in \{-1, 0, 1\}$

$\pi(C) = 1 \quad \Leftrightarrow \quad C$ ist (s,t)-Schnitt.

Kozykel und st-Schnitte

P beliebiger gerichteter Pfad von s nach t.
Definiere $\lambda \cdot \pi : E \to \mathbb{R}$, **$\lambda$-Fluss** auf P

$$\lambda \cdot \pi(e) \coloneqq \begin{cases} \lambda & e \in P \\ -\lambda & \text{rev}(e) \in P \\ 0 & \text{sonst} \end{cases}$$

Vorgehen: Erhöhe λ schrittweise. Was passiert dann mit Kreisen?

- Kreise, die s enthalten, werden um λ kürzer
- Kreise, die t enthalten, werden um λ länger
- Kreise, die s und t nicht trennen, bleiben gleich lang

Kozykel und st-Schnitte

Vorgehen: Erhöhe λ schrittweise. Was passiert dann mit Kreisen?

- Kreise, die s enthalten, werden um λ kürzer
- Kreise, die t enthalten, werden um λ länger
- Kreise, die s und t nicht trennen, bleiben gleich lang

Also: Erhöhe λ solange, bis ein Kreis der Länge 0 entsteht. Das kann man effizient feststellen.

Betrachte Fluß von λ auf P

Setze $\phi := \lambda \cdot \pi$

Betrachte **Residual-Netzwerk** $G_\lambda := G_{\lambda \cdot \phi}$

Das ist G mit residualer Kapazitätsfunktion $c(\lambda, e) := c(e) - \lambda \cdot \pi(e)$

$G^\star_\lambda$: Dual-Graph mit $e^\star$ hat Kosten $c(\lambda, e^\star) := c(\lambda, e)$.

Betrachte Fluß von λ auf P

Setze $\phi := \lambda \cdot \pi$

Betrachte **Residual-Netzwerk** $G_\lambda := G_{\lambda \cdot \phi}$

Das ist G mit residualer Kapazitätsfunktion $c(\lambda, e) := c(e) - \lambda \cdot \pi(e)$

$G^\star_\lambda$: Dual-Graph mit $e^\star$ hat Kosten $c(\lambda, e^\star) := c(\lambda, e)$.

Lemma

G besitzt gültigen $s-t$-Fluß mit Wert λ

$\Leftrightarrow$

$G^\star_\lambda$ enthält keinen negativen Kreis

Betrachte Fluß von λ auf P

Setze $\phi := \lambda \cdot \pi$

Betrachte **Residual-Netzwerk** $G_\lambda := G_{\lambda \cdot \phi}$

Das ist G mit residualer Kapazitätsfunktion $c(\lambda, e) := c(e) - \lambda \cdot \pi(e)$

$G^\star_\lambda$: Dual-Graph mit $e^\star$ hat Kosten $c(\lambda, e^\star) := c(\lambda, e)$.

Lemma

G besitzt gültigen $s-t$-Fluß mit Wert λ

$\Leftrightarrow$

$G^\star_\lambda$ enthält keinen negativen Kreis

$\rightsquigarrow$ parametrisches Kürzeste-Wege-Problem mit Koeffizienten $-1, 0, 1$

Der Allgemeine Fall

Lemma

G besitzt gültigen $s-t$-Fluß mit Wert λ

$\Leftrightarrow$

$G^\star_\lambda$ enthält keinen negativen Kreis

Löse also spezielles parametrisches Kürzeste-Wege-Problem

Koeffizienten $-1, 0, 1$

Der Allgemeine Fall

Lemma

G besitzt gültigen $s-t$-Fluß mit Wert λ

$\Leftrightarrow$

$G^\star_\lambda$ enthält keinen negativen Kreis

Löse also spezielles parametrisches Kürzeste-Wege-Problem
Koeffizienten $-1, 0, 1$

Verwalte Kürzeste-Wege-Baum T_λ in $G^\star_\lambda$ mit Wurzel o.
PlanarMaxFlow(G,c,s,t)

- Berechnet T_0
- Verwalte T_λ, während λ kontinuierlich von 0 bis λ_{max} läuft.
- Berechne $\phi(\lambda_{\text{max}}, \cdot)$ aus $T_{\lambda_{\text{max}}}$.

Abbruchkriterium

Speichere Baum mittels Vorgängerfunktion $\text{pred}(\mathbf{p}, \lambda)$.
Duale Kante $e^\star$ **straff** bei λ $\quad :\Leftrightarrow \quad$ $\text{slack}(\lambda, e^\star) = 0$.

Abbruchkriterium

Speichere Baum mittels Vorgängerfunktion $\text{pred}(\mathbf{p}, \lambda)$.
Duale Kante $e^\star$ **straff** bei λ $:\Leftrightarrow$ slack$(\lambda, e^\star) = 0$.
Außer bei kritischen λ-Werten: $e^\star$ straff $\Leftrightarrow$ $e^\star$ in T_λ

Bei kritischen Werten:

Duale Nichtbaumkante $p \to q$ wird straff.

$\rightsquigarrow$ $p \to q$ ersetzt pred$(\lambda, q) \to q$

Abbruchkriterium

Speichere Baum mittels Vorgängerfunktion $\text{pred}(\mathbf{p}, \lambda)$.
Duale Kante $e^\star$ **straff** bei λ $:\Leftrightarrow$ slack$(\lambda, e^\star) = 0$.
Außer bei kritischen λ-Werten: $e^\star$ straff $\Leftrightarrow$ $e^\star$ in T_λ

Bei kritischen Werten:
Duale Nichtbaumkante $p \to q$ wird straff.
$\rightsquigarrow$ $p \to q$ ersetzt pred$(\lambda, q) \to q$

Pivotschritt

Abbruchkriterium

Speichere Baum mittels Vorgängerfunktion $\text{pred}(\mathbf{p}, \lambda)$.
Duale Kante $e^\star$ **straff** bei λ $:\Leftrightarrow$ slack$(\lambda, e^\star) = 0$.
Außer bei kritischen λ-Werten: $e^\star$ straff $\Leftrightarrow$ $e^\star$ in T_λ

Bei kritischen Werten:
Duale Nichtbaumkante $p \to q$ wird straff.
$\rightsquigarrow$ $p \to q$ ersetzt pred$(\lambda, q) \to q$

Pivotschritt

Lemma

λ_{max} ist erster kritischer Wert von λ, dessen Pivot einen gerichteten Kreis in T_λ erzeugt.

Abbruchkriterium

Speichere Baum mittels Vorgängerfunktion $\text{pred}(\mathbf{p}, \lambda)$.
Duale Kante $e^\star$ **straff** bei λ $:\Leftrightarrow$ slack$(\lambda, e^\star) = 0$.
Außer bei kritischen λ-Werten: $e^\star$ straff $\Leftrightarrow$ $e^\star$ in T_λ

Bei kritischen Werten:
Duale Nichtbaumkante $p \to q$ wird straff.
$\rightsquigarrow$ $p \to q$ ersetzt pred$(\lambda, q) \to q$

Pivotschritt

Lemma

λ_{max} ist erster kritischer Wert von λ, dessen Pivot einen gerichteten Kreis in T_λ erzeugt.

Primale Kante e ist **locker** wenn weder $e^\star$ noch rev$(e^\star)$ straff.
L_λ: lockere Kanten
Außer bei kritischen Werten: L_λ Spannbaum von G.

Abbruchkriterium

Speichere Baum mittels Vorgängerfunktion $\text{pred}(\mathbf{p}, \lambda)$.
Duale Kante $e^\star$ **straff** bei λ $:\Leftrightarrow$ slack$(\lambda, e^\star) = 0$.
Außer bei kritischen λ-Werten: $e^\star$ straff $\Leftrightarrow$ $e^\star$ in T_λ

Bei kritischen Werten:
Duale Nichtbaumkante $p \to q$ wird straff.
$\rightsquigarrow$ $p \to q$ ersetzt pred$(\lambda, q) \to q$

Pivotschritt

Lemma

λ_{max} ist erster kritischer Wert von λ, dessen Pivot einen gerichteten Kreis in T_λ erzeugt.

Primale Kante e ist **locker** wenn weder $e^\star$ noch rev$(e^\star)$ straff.
L_λ: lockere Kanten
Außer bei kritischen Werten: L_λ Spannbaum von G.

Lemma

λ_{max} ist erster kritischer Wert von λ, dessen Pivot L_λ unzusammenhängend macht.

Duale Kante **aktiv** wenn slack mit λ abnimmt.

L_λ enthält eindeutigen st-Pfad LP_λ.

Lemma

Duale Kante $e^\star$ aktiv $\Leftrightarrow$ e Kante von LP_λ.

PlanarMaxFlow(G,c,s,t)

Initialisiere Spannbaum L, Vorgänger und Schlupfvariablen.

while s, t *in selber Komponente von* L **do**

  $LP \leftarrow$ st-Pfad in L

  $p \to q \leftarrow$ Kante in $LP^\star$ mit minimalem Schlupf

  $\delta \leftarrow$ slack$(p \to q)$

  for *jede Kante* e *in* LP **do**

    slack$(e^\star) \leftarrow$ slack$(e^\star) - \delta$

    slack(rev$(e^\star)) \leftarrow$ slack(rev$(e^\star))$ + δ

  end

  Entferne $(p \to q)^\star$ aus L

  if $q \neq o$ **then**

    Füge (pred$(q) \to q)^\star$ in L ein

  end

  pred$(q) \leftarrow p$

end

for *jede Kante* e **do**

  ■$(e) \leftarrow c(e) -$ slack$(e^\star)$

end

PlanarMaxFlow(G,c,s,t)

Initialisiere Spannbaum L, Vorgänger und Schlupfvariablen.

while s, t *in selber Komponente von* L **do**

  $LP \leftarrow$ st-Pfad in L

  $p \dashrightarrow q \leftarrow$ Kante in $LP^\star$ mit minimalem Schlupf

  Datenstruktur Top-Tree verwaltet dynamischen Wald L_λ.
  Erlaubt folgende Operationen mit $O(\log n)$ Zeit:

  - Löschen und Einfügen von Kanten
  - Anfragen ob zwei Knoten in selber Zusammenhangskomponente
  - Finden des kleinsten Gewichts auf einem Pfad
  - Modifizieren aller Gewichte auf einem Pfad um denselben Wert

  end

  $\text{pred}(q) \leftarrow p$

end

for *jede Kante* e **do**

  $\blacksquare(e) \leftarrow c(e) - \text{slack}(e^\star)$

end

PlanarMaxFlow(G,c,s,t)

Initialisiere Spannbaum L, Vorgänger und Schlupfvariablen.

while s, t *in selber Komponente von* L **do** $O(\log n)$

- $LP \leftarrow$ st-Pfad in L
- $p \to q \leftarrow$ Kante in $LP^\star$ mit minimalem Schlupf $O(\log n)$
- $\delta \leftarrow$ slack$(p \to q)$
- **for** *jede Kante* e *in* LP **do**
  - slack$(e^\star) \leftarrow$ slack$(e^\star) - \delta$ $O(\log n)$
  - slack(rev$(e^\star)) \leftarrow$ slack(rev$(e^\star))$ + δ
- **end**
- Entferne $(p \to q)^\star$ aus L $O(\log n)$
- **if** $q \neq o$ **then**
  - Füge (pred$(q) \to q)^\star$ in L ein $O(\log n)$
- **end**
- pred$(q) \leftarrow p$

end

for *jede Kante* e **do**

- $\blacksquare(e) \leftarrow c(e) -$ slack$(e^\star)$ $O(n)$

end

Laufzeit

- Iteration lässt sich in $O(\log n)$ Zeit implementieren mittels top-trees
- Spezielle Struktur $\Rightarrow O(n)$ Pivot-Schritte, (ohne Beweis)
  d.h. $O(n)$ Durchläufe der while-Schleife

$\rightsquigarrow$ Gesamtlaufzeit $O(n \log n)$

Laufzeit

- Iteration lässt sich in $O(\log n)$ Zeit implementieren mittels top-trees
- Spezielle Struktur $\Rightarrow O(n)$ Pivot-Schritte, (ohne Beweis)
  d.h. $O(n)$ Durchläufe der while-Schleife

$\rightsquigarrow$ Gesamtlaufzeit $O(n \log n)$

siehe
”Maximum Flows and Parametric Shortest Paths in Planar Graphs”,
Jeff Erickson, SODA’10

Algorithmus kann als iterative Augmentierung entlang links-liegendster Pfade aufgefasst werden.