Karlsruher Institut für Technologie

Algorithmen für Planare Graphen

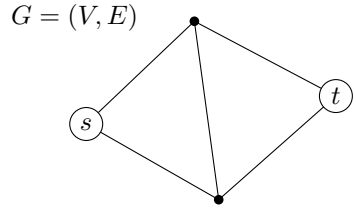
Vorlesung am 19.07.2022

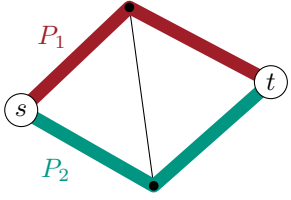
Torsten Ueckerdt | 19. Juli 2022

Das „Disjunkte-Wege-Problem“

Problem MENGER PROBLEM.

- **Gegeben:** Graph $G = (V, E)$ und $s, t \in V$.
- **Gesucht:** Max. Anzahl **kantendisjunkter** s-t-Pfade in G.

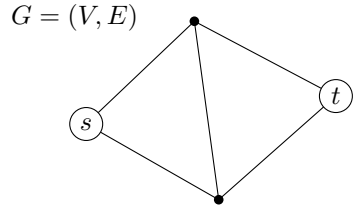




Das „Disjunkte-Wege-Problem“

Problem MENGER PROBLEM.

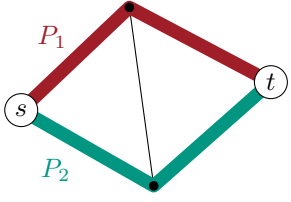
- **Gegeben:** Graph $G = (V, E)$ und $s, t \in V$.
- **Gesucht:** Max. Anzahl **kantendisjunkter** s-t-Pfade in G.



Kann im Allgemeinen mit MAX-FLOW gelöst werden:

- Gebe jeder Kante Kapazität 1.

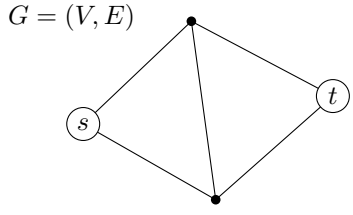
Damit erhalten wir für planare Graphen einen Algo in $O(n \log n)$.



Das „Disjunkte-Wege-Problem“

Problem MENGER PROBLEM.

- **Gegeben:** Graph $G = (V, E)$ und $s, t \in V$.
- **Gesucht:** Max. Anzahl **kantendisjunkter** s-t-Pfade in G.

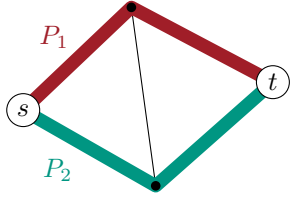


Kann im Allgemeinen mit MAX-FLOW gelöst werden:

- Gebe jeder Kante Kapazität 1.

Damit erhalten wir für planare Graphen einen Algo in $O(n \log n)$.

- Wir werden aber eine Lösung in Linearzeit kennenlernen.
- O.B.d.A. liegt t an der äußeren Facette.

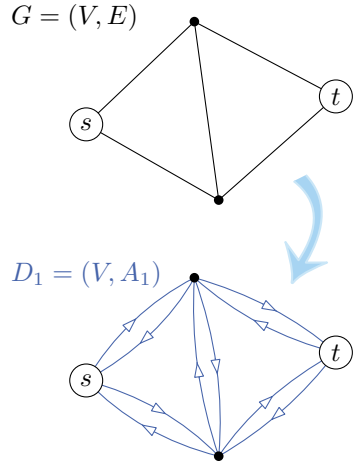


Grobes Vorgehen

Sei t an der äusseren Facette von $G = (V, E)$.

1. Schritt:

- Konstruiere $D_1 = (V, A_1)$ mit $uv \in A_1 \Leftrightarrow uv \in E$



Grobes Vorgehen

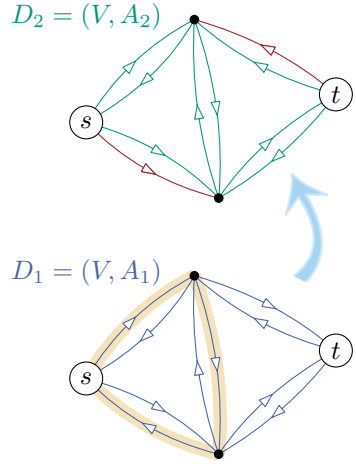
Sei t an der äusseren Facette von $G = (V, E)$.

1. Schritt:

- Konstruiere $D_1 = (V, A_1)$ mit $uv \in A_1 \Leftrightarrow uv \in E$

2. Schritt:

- Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine gerichteten Kreise im Uhrzeigersinn (Rechtskreise) enthält.



Grobes Vorgehen

Sei t an der äusseren Facette von $G = (V, E)$.

1. Schritt:

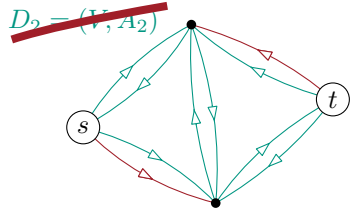
- Konstruiere $D_1 = (V, A_1)$ mit $uv \in A_1 \Leftrightarrow uv \in E$

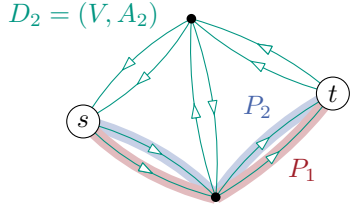
2. Schritt:

- Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine gerichteten Kreise im Uhrzeigersinn (Rechtskreise) enthält.

3. Schritt:

- Finde maximale Anzahl gerichteter kantendisjunkter s-t-Pfade $P_1, \ldots, P_k$ in D_2.





Grobes Vorgehen

Sei t an der äusseren Facette von $G = (V, E)$.

1. Schritt:

- Konstruiere $D_1 = (V, A_1)$ mit $uv \in A_1 \Leftrightarrow uv \in E$

2. Schritt:

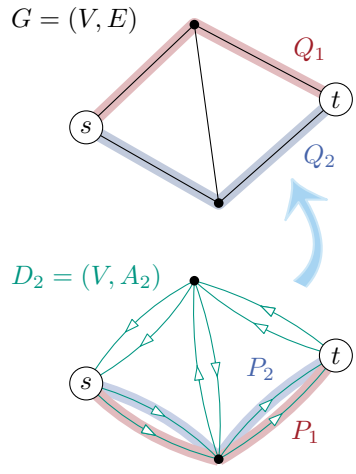
- Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine gerichteten Kreise im Uhrzeigersinn (Rechtskreise) enthält.

3. Schritt:

- Finde maximale Anzahl gerichteter kantendisjunkter s-t-Pfade $P_1, \ldots, P_k$ in D_2.

4. Schritt:

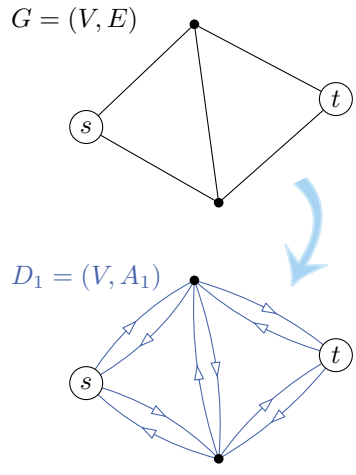
- Berechne aus $P_1, \ldots P_k$ ungerichtete s-t-Pfade $Q_1, \ldots, Q_k$ in G.



Korrektheit von Schritt 1

Konstruiere $D_1 = (V, A_1)$ mit $uv \in A_1 \Leftrightarrow uv \in E$

Laufzeit und Durchführung sind klar.



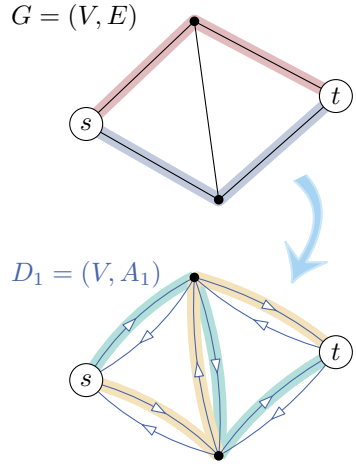
Korrektheit von Schritt 1

Konstruiere $D_1 = (V, A_1)$ mit $uv \in A_1 \Leftrightarrow uv \in E$

Lemma.

G hat *k* kantendisjunkte *s*-*t*-Pfade
$\Leftrightarrow$ D_1 hat *k* kantendisjunkte gerichtete *s*-*t*-Pfade

Beweisidee:



Korrektheit von Schritt 1

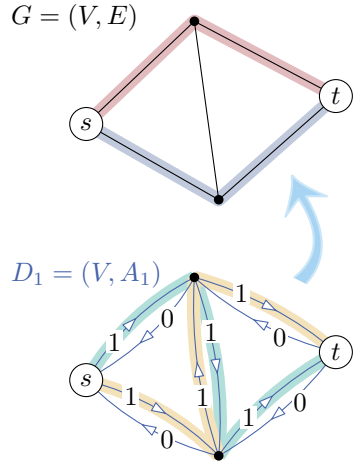
Konstruiere $D_1 = (V, A_1)$ mit $uv \in A_1 \Leftrightarrow uv \in E$

Lemma.

G hat *k* kantendisjunkte *s*-*t*-Pfade
$\Leftrightarrow$ D_1 hat *k* kantendisjunkte gerichtete *s*-*t*-Pfade

Beweisidee:

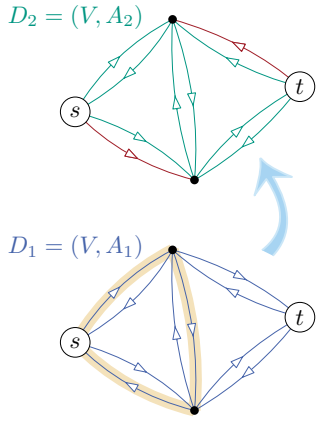
- Gegeben *k* Pfade in D_1
- Interpretiere diese als Fluss Φ von Wert *k* auf D_1 mit $\Phi(uv) \in \{0, 1\}$ für jede gerichtete Kante.
- Definiere Φ' als
  - $\Phi'(uv) = \Phi'(vu) = 0$ wenn $\Phi(uv) = \Phi(vu) = 1$.
  - Ansonsten $\Phi'(uv) = \Phi(uv)$.
- Φ' hat Wert *k* und gibt *k* kantendisjunkte Wege in *G*. ✓



Konstruktion in Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Ablauf:

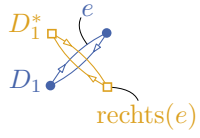


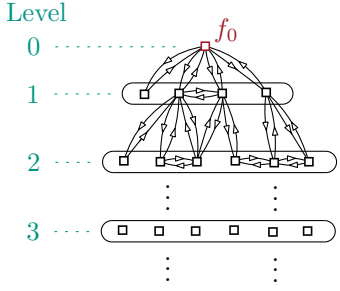
Konstruktion in Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Ablauf:

- Betrachte Breitensuche von der äußeren Facette f_0 im gerichteten Dualgraphen D_1^* von D_1.
- Auf Level i sind alle Facetten mit Abstand i zu f_0.
- Drehe für jedes i die Richtung aller gerichteten Kanten von Level i nach Level $i+1$ um.
- Ergebnis: D_2^* mit primalen Graph D_2.



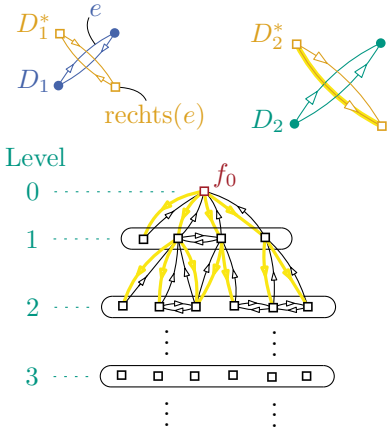


Konstruktion in Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Ablauf:

- Betrachte Breitensuche von der äußeren Facette f_0 im gerichteten Dualgraphen D_1^* von D_1.
- Auf Level i sind alle Facetten mit Abstand i zu f_0.
- Drehe für jedes i die Richtung aller gerichteten Kanten von Level i nach Level $i+1$ um.
- Ergebnis: D_2^* mit primalen Graph D_2.

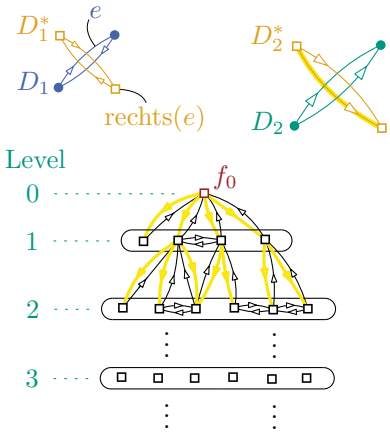


Konstruktion in Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Ablauf:

- Betrachte Breitensuche von der äußeren Facette f_0 im gerichteten Dualgraphen D_1^* von D_1.
- Auf Level i sind alle Facetten mit Abstand i zu f_0.
- Drehe für jedes i die Richtung aller gerichteten Kanten von Level i nach Level $i+1$ um.
- Ergebnis: D_2^* mit primalen Graph D_2.
- Linearzeit.



Konstruktion in Schritt 2

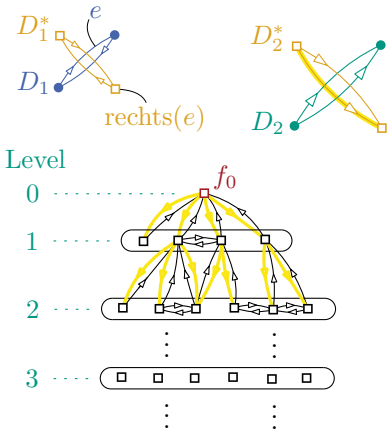
Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Ablauf:

- Betrachte Breitensuche von der äußeren Facette f_0 im gerichteten Dualgraphen D_1^* von D_1.
- Auf Level i sind alle Facetten mit Abstand i zu f_0.
- Drehe für jedes i die Richtung aller gerichteten Kanten von Level i nach Level $i+1$ um.
- Ergebnis: D_2^* mit primalen Graph D_2.

- Linearzeit. ✓
- Wir werden zeigen:

1.) D_2 enthält keine Rechtskreise
2.) D_2 enthält genauso viele kantendisj. s-t-Pfade wie D_1.
   - Damit ist **Schritt 2** korrekt.



Korrektheit von Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Lemma.

D_2 enthält keine Rechtskreise.

Beweis:

Korrektheit von Schritt 2

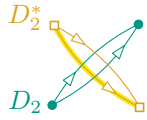
Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

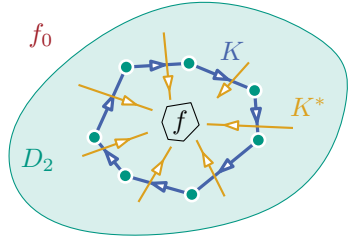
Lemma.

D_2 enthält keine Rechtskreise.

Beweis:

- Angenommen K ist ein Rechtskreis in D_2.
- Dann ist K^* ein gerichteter Schnitt in D_2^*.
- Sei f eine Facette innerhalb von K.
- Ein kürzester Weg von f zu f_0 muss den Schnitt passieren.
- Aber diese Kanten wurden in D_2^* umgedreht.
- Widerspruch dazu, dass K ein Rechtskreis ist. ✗





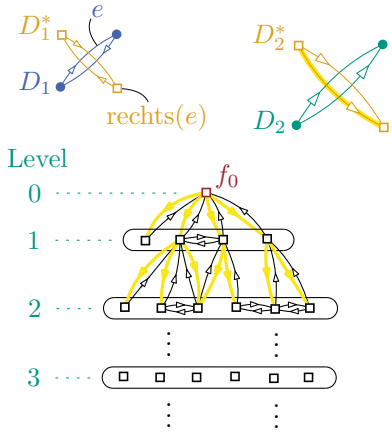
Korrektheit von Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Lemma. (Korrektheit)

D_2 enthält k kantendisjunkte s-t-Wege
$\Leftrightarrow$ D_1 enthält k kantendisjunkte s-t-Wege.

Beweis:



Korrektheit von Schritt 2

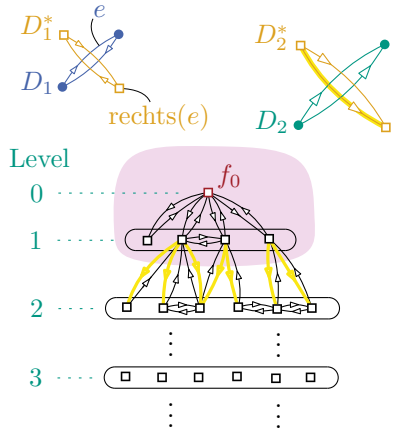
Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Lemma. (Korrektheit)

D_2 enthält k kantendisjunkte s-t-Wege
$\Leftrightarrow$ D_1 enthält k kantendisjunkte s-t-Wege.

Beweis:

- In **Schritt 2** wurden im Dualen immer Kantenmengen zwischen zwei Levels umgedreht.
- Das sind im Dualen D_1^* knoteninduzierte Schnitte.
- Im Primalen D_1 sind das für ein festes Level eine disjunkte Vereinigung von Kreisen.
- Es reicht also zu zeigen, dass k kantendisjunkte Wege erhalten bleiben wenn ein gerichteter Kreis umgedreht wird.



Korrektheit von Schritt 2

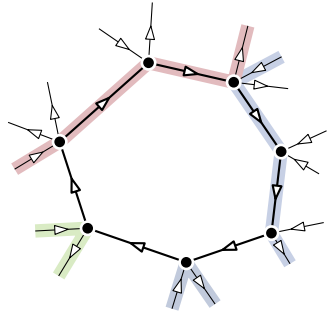
Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Lemma. (Korrektheit)

D_2 enthält k kantendisjunkte s-t-Wege
$\Leftrightarrow$ D_1 enthält k kantendisjunkte s-t-Wege.

Beweis:

- Sei also ein gerichteter Kreis umgedreht.
- Interpretiere wieder k Wege als Fluss ϕ mit Wert k.
- Drehe Orientierung der Kanten und gleichzeitig den Flusswert der Kante $0 \to 1$ oder $1 \to 0$.
- Dies erhält die Flusseigenschaften.
- Wir erhalten wieder ein Fluss ϕ' mit Wert k.
- Wir erhalten auch wieder k Wege. ✓

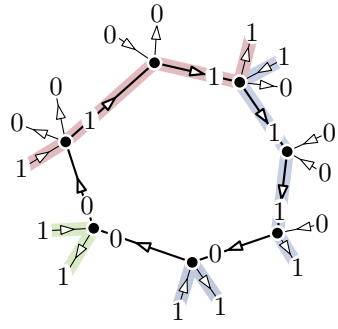


Korrektheit von Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Lemma. (Korrektheit)

D_2 enthält k kantendisjunkte s-t-Wege
$\Leftrightarrow$ D_1 enthält k kantendisjunkte s-t-Wege.



Beweis:

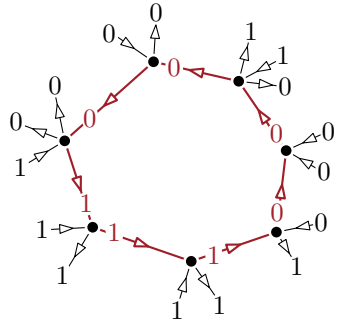
- Sei also ein gerichteter Kreis umgedreht.
- Interpretiere wieder k Wege als Fluss ϕ mit Wert k.
- Drehe Orientierung der Kanten und gleichzeitig den Flusswert der Kante $0 \to 1$ oder $1 \to 0$.
- Dies erhält die Flusseigenschaften.
- Wir erhalten wieder ein Fluss ϕ' mit Wert k.
- Wir erhalten auch wieder k Wege. ✓

Korrektheit von Schritt 2

Konstruiere $D_2 = (V, A_2)$ aus D_1, sodass D_2 keine Rechtskreise enthält.

Lemma. (Korrektheit)

D_2 enthält k kantendisjunkte s-t-Wege
$\Leftrightarrow$ D_1 enthält k kantendisjunkte s-t-Wege.



Beweis:

- Sei also ein gerichteter Kreis umgedreht.
- Interpretiere wieder k Wege als Fluss ϕ mit Wert k.
- Drehe Orientierung der Kanten und gleichzeitig den Flusswert der Kante $0 \to 1$ oder $1 \to 0$.
- Dies erhält die Flusseigenschaften.
- Wir erhalten wieder ein Fluss ϕ' mit Wert k.
- Wir erhalten auch wieder k Wege. ✓

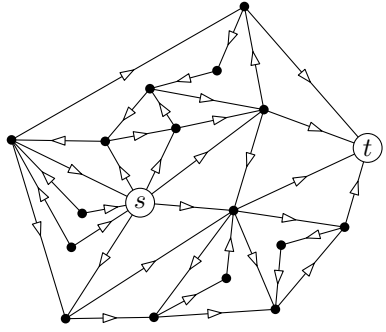
Zwischenstand

Gegeben:

- gerichteter planarer Graph D_2
  - mit Doppelkanten und antiparallelen Kanten
  - ohne Rechtskreise
  - mit $\mathrm{outdeg}(v) = \mathrm{indeg}(v)$ für jeden Knoten v
- zwei Knoten s, t
  - mit t an der äusseren Facette

Gesucht:

- kantendisjunkte s-t-Pfade $P_1, \ldots, P_k$ in D_2
  - mit k so groß wie möglich

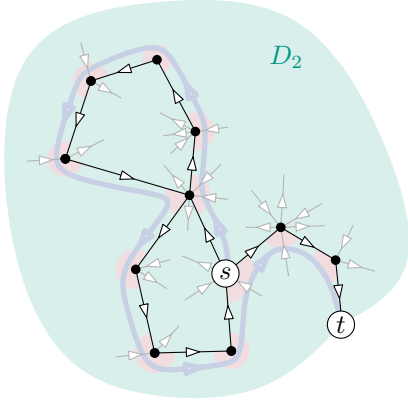


Wir wollen die Berechnung in Linearzeit.

Vorgehen in Schritt 3

Finde maximale Anzahl gerichteter kantendisjunkter s-t-Pfade $P_1, \ldots, P_k$ in D_2.

Ablauf:

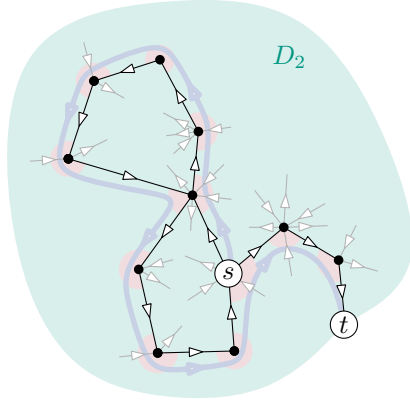


Vorgehen in Schritt 3

Finde maximale Anzahl gerichteter kantendisjunkter s-t-Pfade $P_1, \ldots, P_k$ in D_2.

Ablauf:

- Starte Tiefensuche bei s mit ausgehender Kante.
- Nimm immer die rechteste ausgehende Kante bzgl. der gerade genommenen eingehenden Kante.
- Gehe über jede Kante nur einmal.

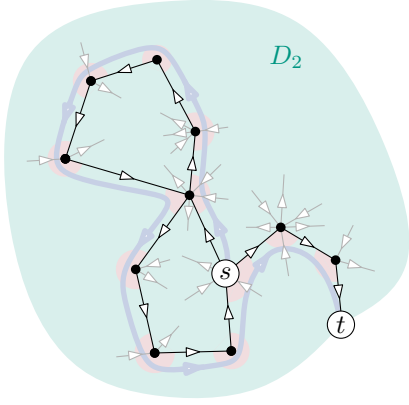


Vorgehen in Schritt 3

Finde maximale Anzahl gerichteter kantendisjunkter s-t-Pfade $P_1, \ldots, P_k$ in D_2.

Ablauf:

- Starte Tiefensuche bei s mit ausgehender Kante.
- Nimm immer die rechteste ausgehende Kante bzgl. der gerade genommenen eingehenden Kante.
- Gehe über jede Kante nur einmal.
- Wenn t erreicht wird
  - ist ein s-t-Pfad gefunden
  - und man fängt wieder bei s mit neuer Kante an.



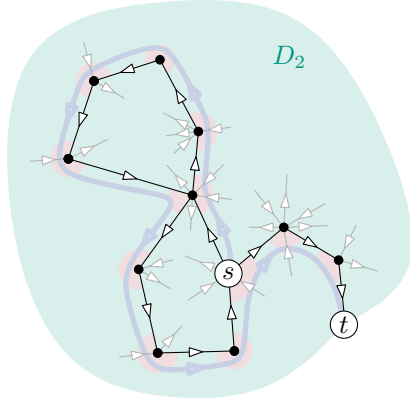
Vorgehen in Schritt 3

Finde maximale Anzahl gerichteter kantendisjunkter s-t-Pfade $P_1, \ldots, P_k$ in D_2.

Ablauf:

- Starte Tiefensuche bei s mit ausgehender Kante.
- Nimm immer die rechteste ausgehende Kante bzgl. der gerade genommenen eingehenden Kante.
- Gehe über jede Kante nur einmal.
- Wenn t erreicht wird
  - ist ein s-t-Pfad gefunden
  - und man fängt wieder bei s mit neuer Kante an.
- Aufhören wenn keine ausgehende Kante verfügbar ist (das passiert an Knoten s)

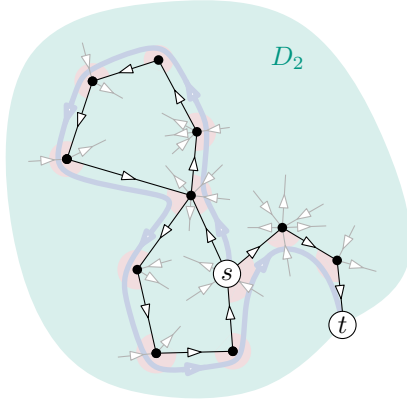
Wenn Knoten t insgesamt k-mal erreicht wurde, haben wir k kantendisjunkte s-t-Pfade gefunden.



Laufzeit und Korrektheit

Laufzeit:

- Nicht ganz so einfach.
  - Rechteste Kante kann mit Union-Find-Datenstruktur gefunden werden.
  - Insgesamt kann somit Linearzeit erreicht werden.



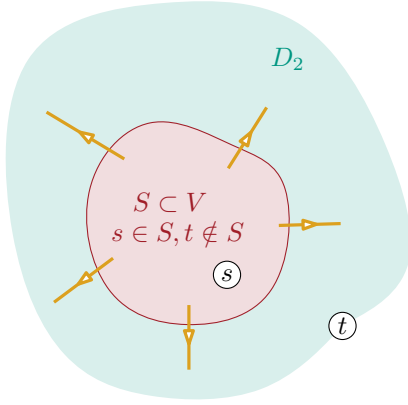
Laufzeit und Korrektheit

Laufzeit:

- Nicht ganz so einfach.
  - Rechteste Kante kann mit Union-Find-Datenstruktur gefunden werden.
  - Insgesamt kann somit Linearzeit erreicht werden.

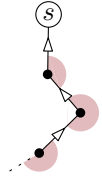
Korrektheit:

- Korrektheit auch noch nicht klar.
  - Warum kann der Algorithmus nicht zu wenige Wege ausgeben?
  - Wir finden einen *s*-*t*-Schnitt dessen Kapazität der Anzahl *k* der gefundenen *s*-*t*-Pfade entspricht.



Finden eines s-t-Schnitts mit Kapazität k

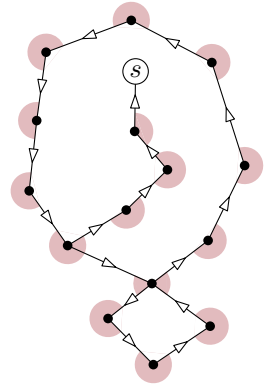
- Betrachte den Graphen $D = (V, A)$ auf allen Kanten, die im Algorithmus genommen wurden.
- Starte Tiefensuche in D bei s aber gehe Kanten rückwärts und nehme die linkeste mögliche Fortsetzung.



Finden eines s-t-Schnitts mit Kapazität k

- Betrachte den Graphen $D = (V, A)$ auf allen Kanten, die im Algorithmus genommen wurden.
- Starte Tiefensuche in D bei s aber gehe Kanten rückwärts und nehme die linkeste mögliche Fortsetzung.

- Baue damit rückwärts einen gerichteten Weg W zu s auf.
- Stoppe,
  - wenn wir eine Kante aus W als Fortsetzung nehmen würden
  - oder wenn wir wieder auf s treffen.

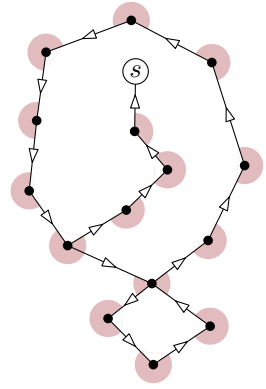


Finden eines s-t-Schnitts mit Kapazität k

- Betrachte den Graphen $D = (V, A)$ auf allen Kanten, die im Algorithmus genommen wurden.
- Starte Tiefensuche in D bei s aber gehe Kanten rückwärts und nehme die linkeste mögliche Fortsetzung.

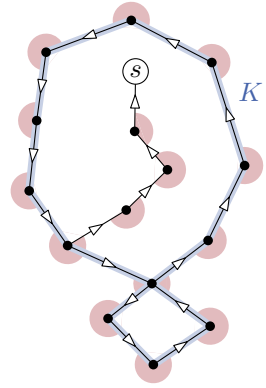
- Baue damit rückwärts einen gerichteten Weg W zu s auf.
- Stoppe,
  - wenn wir eine Kante aus W als Fortsetzung nehmen würden
  - oder wenn wir wieder auf s treffen.

- Wir treffen nicht auf t, da t in D keine ausgehende Kanten hat.
- Wir bleiben nicht hängen, da alle anderen Knoten so viele eingehende wie ausgehende Kanten haben.



Finden eines s-t-Schnitts mit Kapazität k

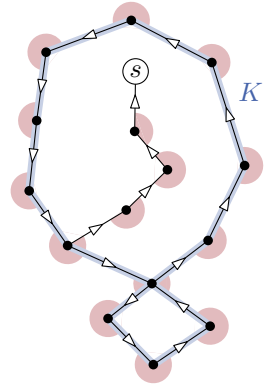
- W umschließt ein Gebiet (Menge von Facetten), dessen Rand K eine Menge gerichteter Kreise ist.
- Entlang von K (Richtung der Orientierung) sind auf der rechten Seite Winkel, die keine eingehenden Kanten haben.



Finden eines s-t-Schnitts mit Kapazität k

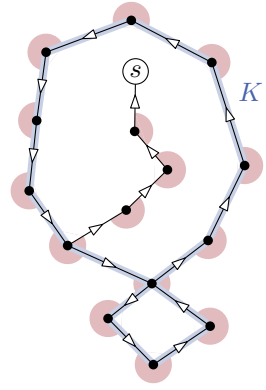
- W umschließt ein Gebiet (Menge von Facetten), dessen Rand K eine Menge gerichteter Kreise ist.
- Entlang von K (Richtung der Orientierung) sind auf der rechten Seite Winkel, die keine eingehenden Kanten haben.

- K enthält keinen Rechtskreis (**Schritt 2**).
- Die Winkel sind also außen an den Kreisen (also in ext(K)).
- Also muss s entweder auf K oder int(K) liegen, da Wege von S zu den Knoten in K existieren.
  - Diese Wege können nicht in die Winkel laufen.



Finden eines s-t-Schnitts mit Kapazität k

- W umschließt ein Gebiet (Menge von Facetten), dessen Rand K eine Menge gerichteter Kreise ist.
- Entlang von K (Richtung der Orientierung) sind auf der rechten Seite Winkel, die keine eingehenden Kanten haben.

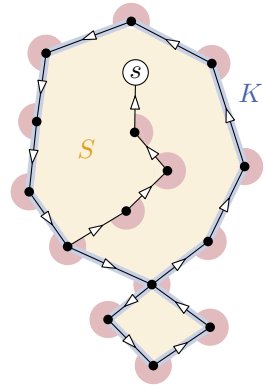


- K enthält keinen Rechtskreis (**Schritt 2**).
- Die Winkel sind also außen an den Kreisen (also in ext(K)).
- Also muss s entweder auf K oder int(K) liegen, da Wege von S zu den Knoten in K existieren.
  - Diese Wege können nicht in die Winkel laufen.

- Knoten t liegt nicht auf K da er ohne ausgehende Kanten nicht von der Rückwärts-Tiefensuche gefunden werden kann.
- Knoten t liegt auf der äußeren Facette von D.
- $\Rightarrow$ t liegt in ext(K).

Finden eines s-t-Schnitts mit Kapazität k

- Definiere $S \subseteq V$ als Menge aller Knoten auf K und in int(K).
- S induziert unseren s-t-Schnitt $C \subseteq A_2$.
- **Zu zeigen:** Kapazität c(C) $\leq k$.

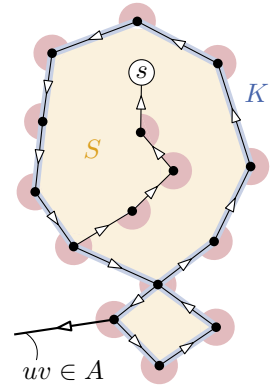


Finden eines s-t-Schnitts mit Kapazität k

- Definiere $S \subseteq V$ als Menge aller Knoten auf K und in int(K).
- S induziert unseren s-t-Schnitt $C \subseteq A_2$.
- **Zu zeigen:** Kapazität c(C) $\leq k$.

1. Sei $uv \in A$ Kante mit $u \in V(K)$ und $v \in V - S$:

- uv wurde vom Algo benutzt.
- Dann ist die Tiefensuche nach uv nicht wieder nach S gekommen, endete also in t.
  - Denn: Winkel haben keine eingehenden Kanten aus A!
- Es gibt also höchstens k solcher Kanten!

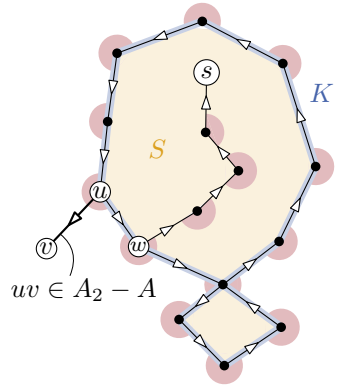


Finden eines s-t-Schnitts mit Kapazität k

- Definiere $S \subseteq V$ als Menge aller Knoten auf K und in int(K).
- S induziert unseren s-t-Schnitt $C \subseteq A_2$.
- **Zu zeigen:** Kapazität $c(C) \leq k$.

2. Sei $uv \in A_2 - A$ Kante mit $u \in V(K)$ und $v \in V - S$:

- uv wurde vom Algorithmus nicht benutzt.
- Betrachte $uw \in E(K)$, die benutzt wurde und deren benutzte Vorgängerkante xu.
- Dann liegt $x \in S$, da keine benutzten eingehenden Kanten in den Winkeln liegen.
- Dann ist aber uv eine bessere (weiter rechts) Fortsetzung für xu als uw Widerspruch! ✗
- $\Rightarrow$ Es gibt keine solche Kanten!

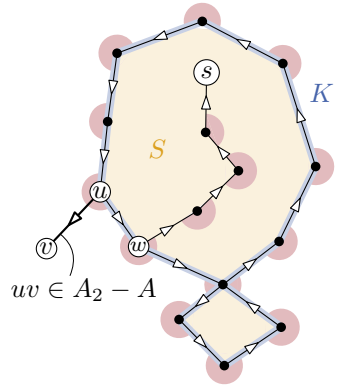


Finden eines s-t-Schnitts mit Kapazität k

- Definiere $S \subseteq V$ als Menge aller Knoten auf K und in int(K).
- S induziert unseren s-t-Schnitt $C \subseteq A_2$.
- **Zu zeigen:** Kapazität $c(C) \leq k$.

2. Sei $uv \in A_2 - A$ Kante mit $u \in V(K)$ und $v \in V - S$:

- uv wurde vom Algorithmus nicht benutzt.
- Betrachte $uw \in E(K)$, die benutzt wurde und deren benutzte Vorgängerkante xu.
- Dann liegt $x \in S$, da keine benutzten eingehenden Kanten in den Winkeln liegen.
- Dann ist aber uv eine bessere (weiter rechts) Fortsetzung für xu als uw Widerspruch! ✗
- $\Rightarrow$ Es gibt keine solche Kanten!



Der Schnitt hat also Kapazität höchstens k

$\Rightarrow$ Der Algorithmus ist optimal!

Abschluss

Satz.

Das disjunkte Wege-Problem in planaren Graphen kann in Linearzeit gelöst werden.

- Wir sind fast fertig, nur müssen wir aus den disjunkten Wegen auf D_2 disjunkte Wege auf G machen (**Schritt 4**).
- Wie das geht haben wir aber schon in **Schritt 1** und **Schritt 2** beschrieben. ✓