

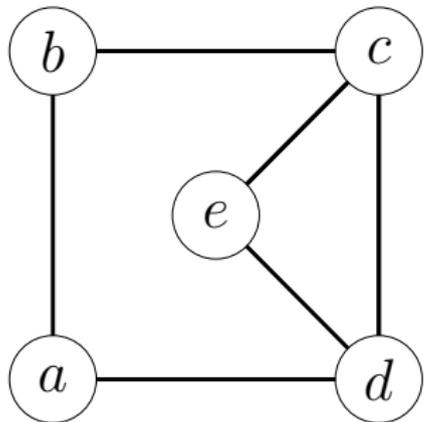
Algorithmen für Planare Graphen

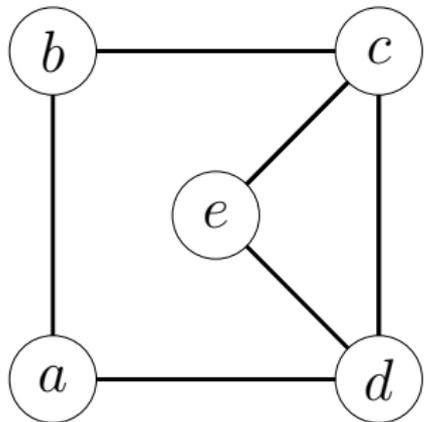
12. Mai 2022, Übung 2

Lars Gottesbüren

INSTITUT FÜR THEORETISCHE INFORMATIK

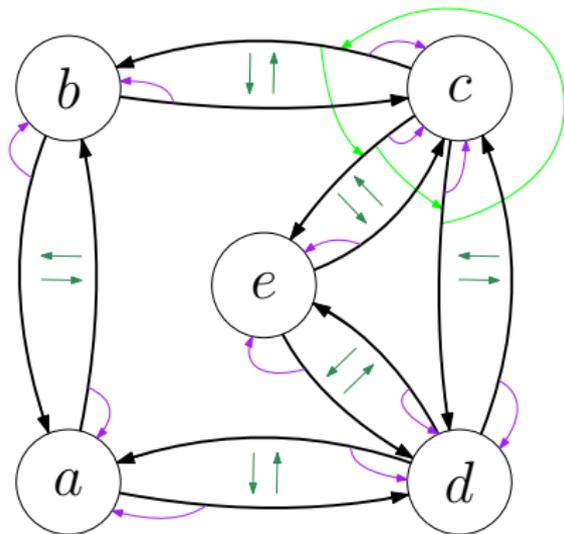




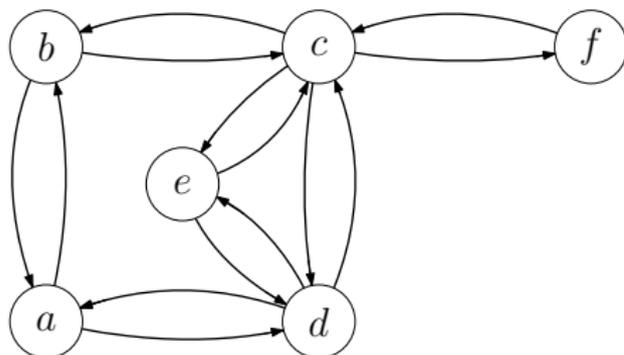


- Liste aller Knoten



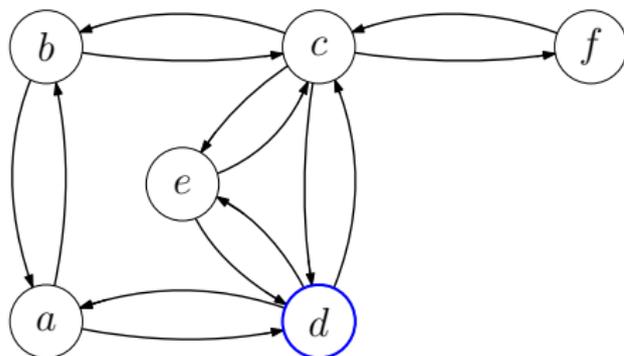


- Liste aller Knoten
- Knoten: Liste gerichteter Kanten entgegen dem Uhrzeigersinn
- Kante:
 - Zeiger auf entgegengerichtete Kante
 - Zeiger auf Ursprungsknoten



Duale Knoten

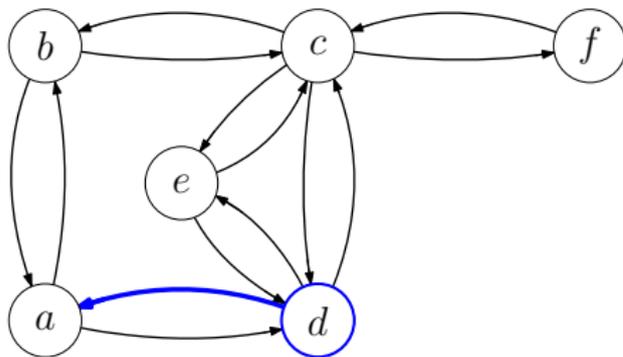
- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.



Duale Knoten

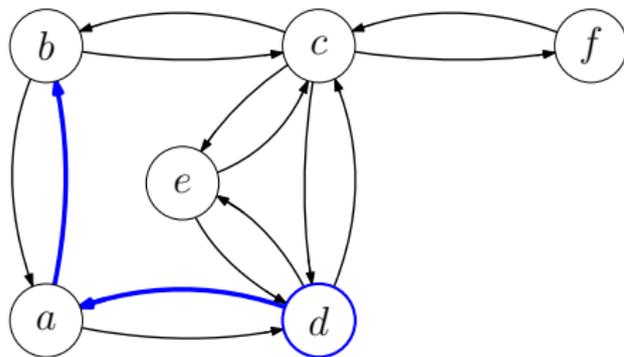
- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.

Duale Knoten



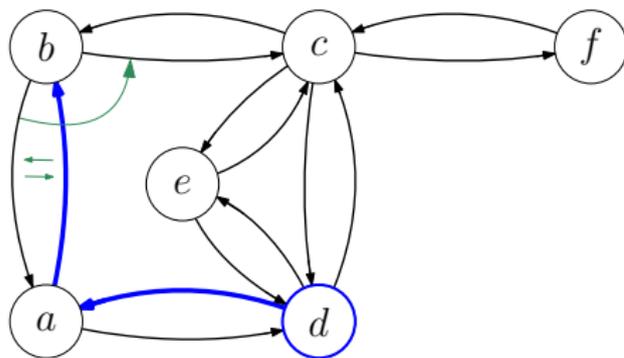
- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.

Duale Knoten



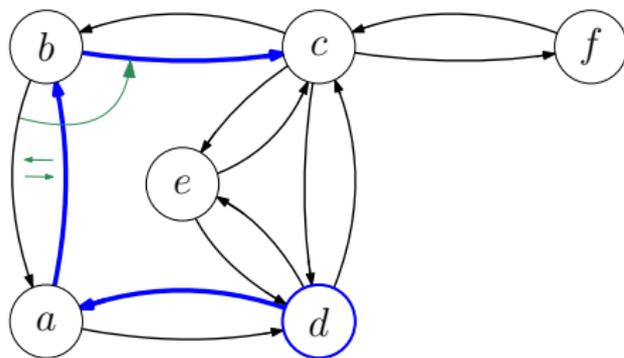
- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.

Duale Knoten



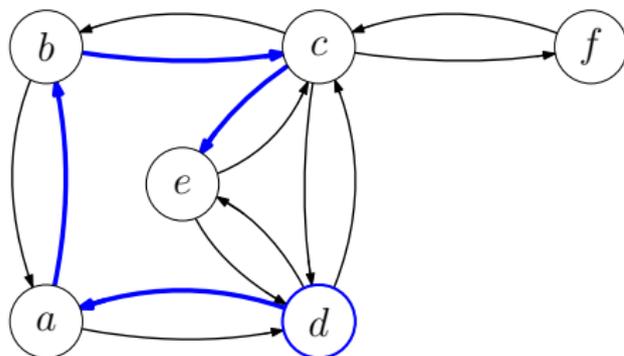
- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.

Duale Knoten

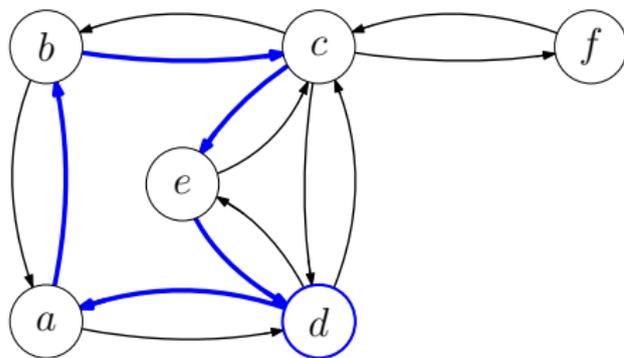


- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.

Duale Knoten

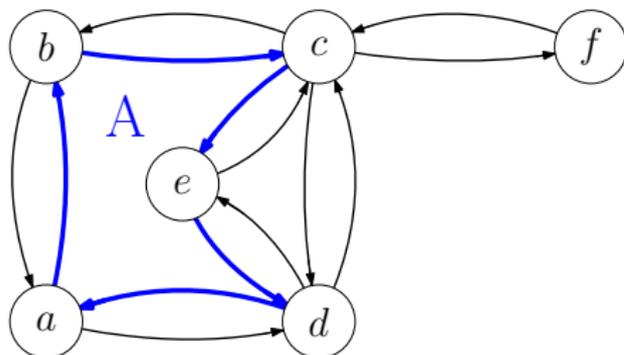


- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.



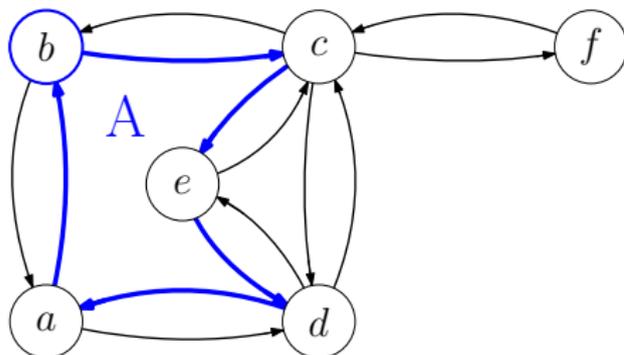
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



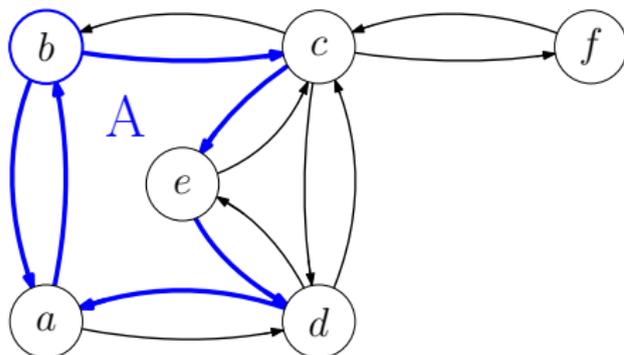
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



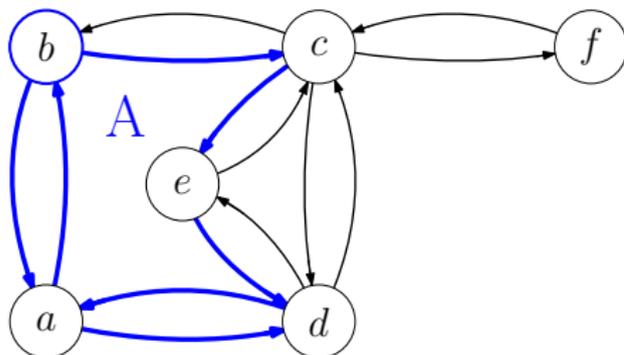
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



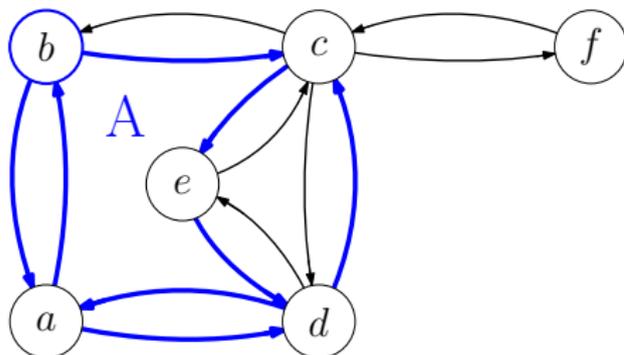
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



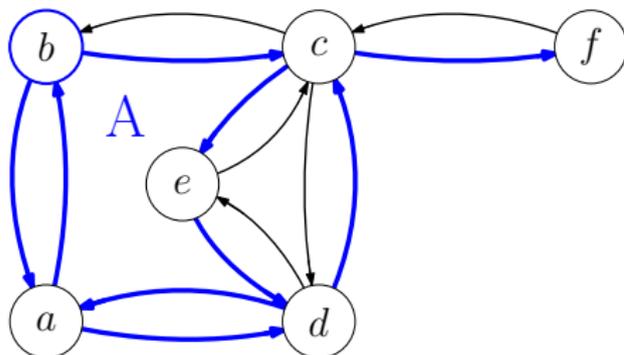
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



Duale Knoten

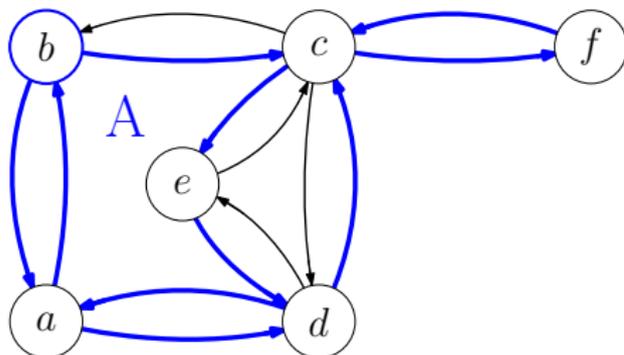
- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



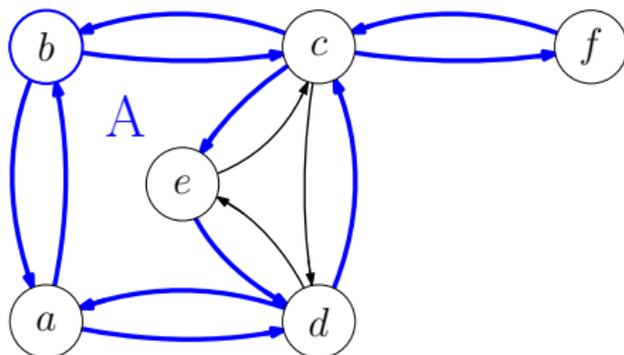
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.

Duale Knoten

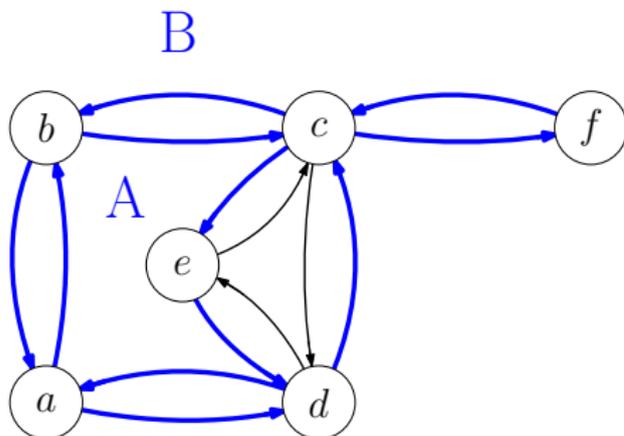


- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



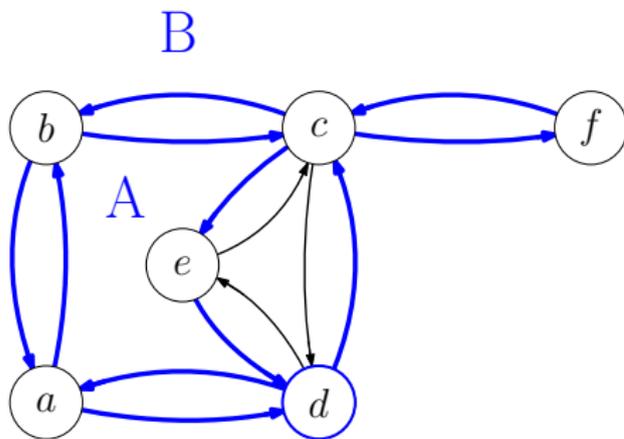
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



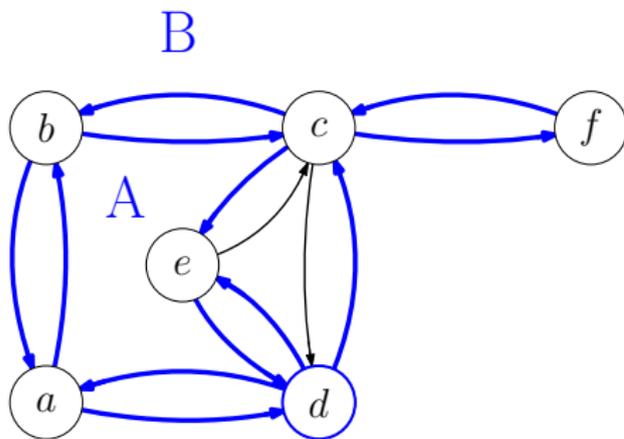
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



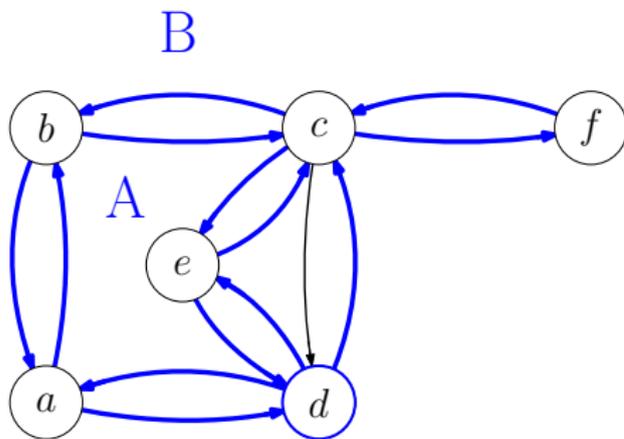
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



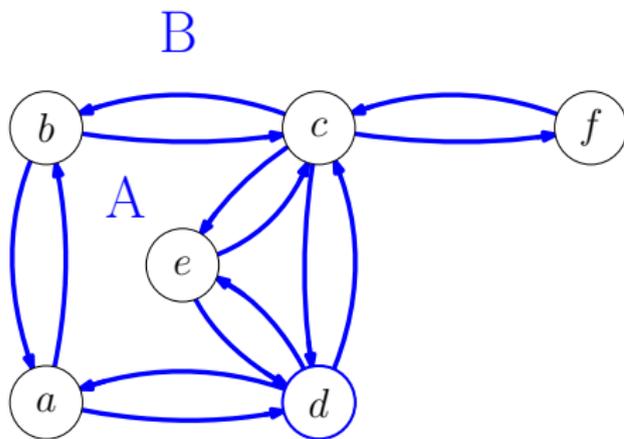
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



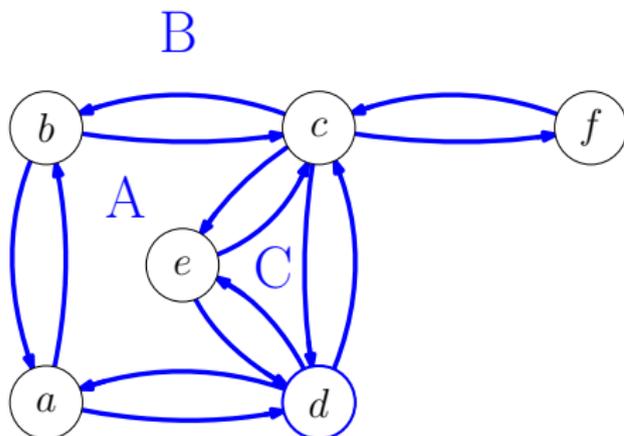
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



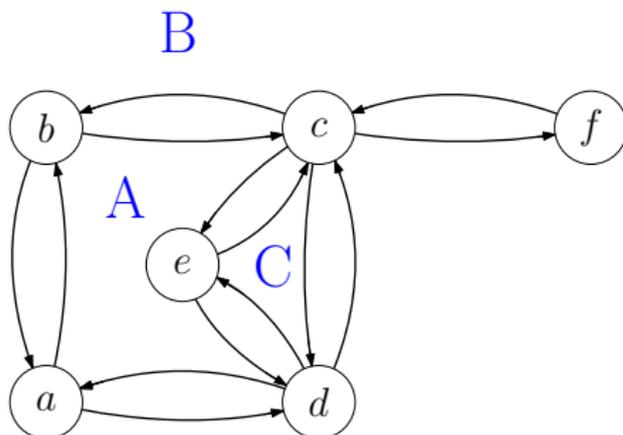
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



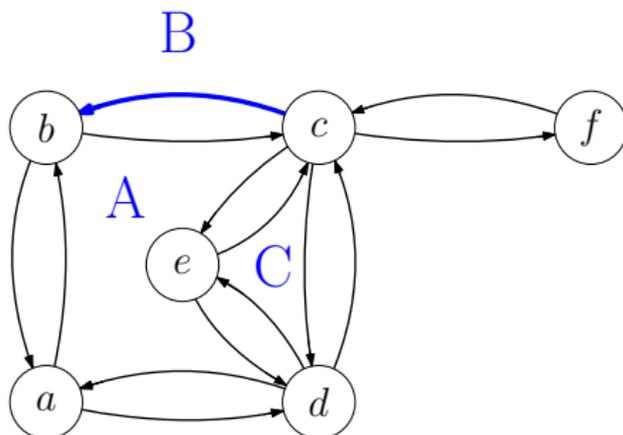
Duale Knoten

- Wähle einen Knoten und eine ausgehende Kante beliebig.
- Beim nächsten Knoten wähle erste Kante gegen den Uhrzeigersinn.
- Markiere jede traversierte Kante mit der aktuellen *Facetten-ID*.
- Erhöhe die *Facetten-ID* wenn wieder beim Ausgangsknoten angekommen.



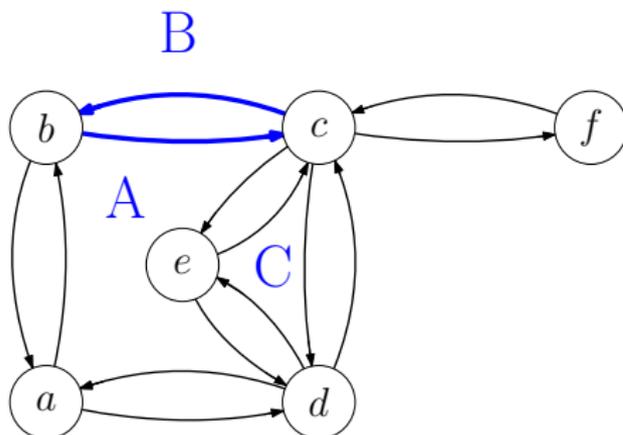
Duale Kanten

- Traversiere erneut jede Facette.
- Betrachte Hin- und Rückkante und füge für die dualen Knoten mit den entsprechenden *Facetten-IDs* eine duale Kante ein.
- Die dualen Kanten werden in der richtigen Reihenfolge eingefügt.



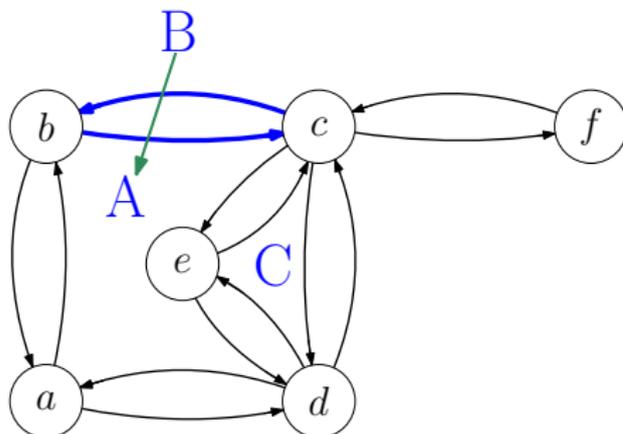
Duale Kanten

- Traversiere erneut jede Facette.
- Betrachte Hin- und Rückkante und füge für die dualen Knoten mit den entsprechenden *Facetten-IDs* eine duale Kante ein.
- Die dualen Kanten werden in der richtigen Reihenfolge eingefügt.



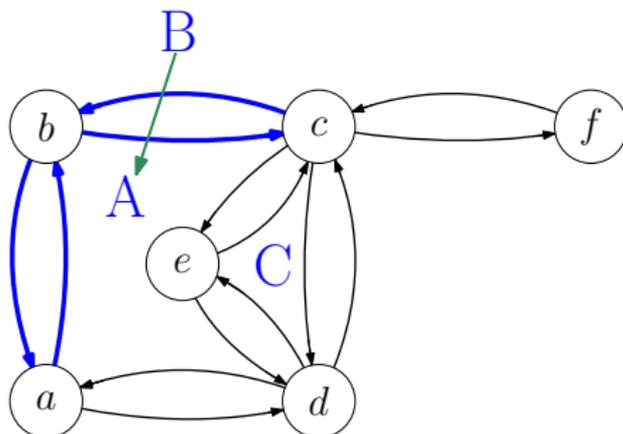
Duale Kanten

- Traversiere erneut jede Facette.
- Betrachte Hin- und Rückkante und füge für die dualen Knoten mit den entsprechenden *Facetten-IDs* eine duale Kante ein.
- Die dualen Kanten werden in der richtigen Reihenfolge eingefügt.



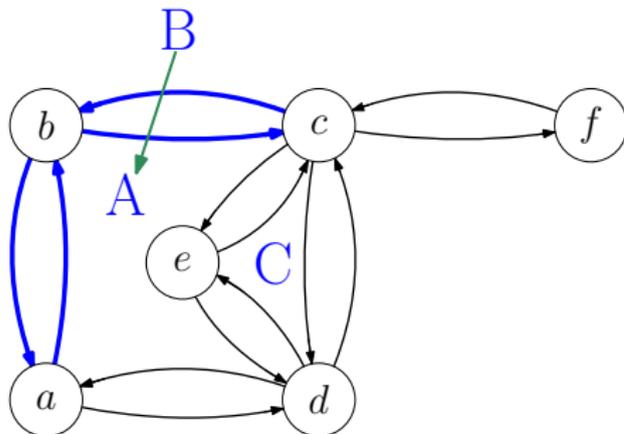
Duale Kanten

- Traversiere erneut jede Facette.
- Betrachte Hin- und Rückkante und füge für die dualen Knoten mit den entsprechenden *Facetten-IDs* eine duale Kante ein.
- Die dualen Kanten werden in der richtigen Reihenfolge eingefügt.



Duale Kanten

- Traversiere erneut jede Facette.
- Betrachte Hin- und Rückkante und füge für die dualen Knoten mit den entsprechenden *Facetten-IDs* eine duale Kante ein.
- Die dualen Kanten werden in der richtigen Reihenfolge eingefügt.



Rückkanten

- Speichere beim Einfügen der Kanten Zeiger von Kante im Ausgangsgraphen auf die Kante im Dualgraphen
- Wenn beim Einfügen einer Kante die Rückkante im Ausgangsgraphen schon einen Zeiger hat, ergänze beide Zeiger im Dualgraphen

2 – Färben außenplanarer Graphen

Zeigen Sei dass außenplanare Graphen 3-färbbar sind:

① mithilfe des 4-Farben-Satzes

② ohne den 4 -Farben-Satz

Zeigen Sei dass außenplanare Graphen 3-färbbar sind:

① mithilfe des 4-Farben-Satzes

- Füge Knoten v ein und verbinde zu allen Knoten auf der äußeren Facette
- Berechne 4-Färbung
- kein anderer Knoten hat die gleiche Farbe wie $v \Rightarrow$ 3-Färbung

② ohne den 4 -Farben-Satz

2.2 – Färben außenplanarer Graphen

- Klauere den Beweis für 5-Listenfärbbarkeit

2.2 – Färben außenplanarer Graphen

- Klauere den Beweis für 5-Listenfärbbarkeit
- Fall 1. Keine Sehne \Rightarrow Der Graph ist ein Kreis + Bäume.
 - gerade Länge \Rightarrow 2-färbbar
 - ungerade Länge \Rightarrow 3-färbbar

2.2 – Färben außenplanarer Graphen

- Klau den Beweis für 5-Listenfärbbarkeit
- Fall 1. Keine Sehne \Rightarrow Der Graph ist ein Kreis + Bäume.
 - gerade Länge \Rightarrow 2-färbbar
 - ungerade Länge \Rightarrow 3-färbbar
- Fall 2. Graph hat Sehne $e = (u, v)$

2.2 – Färben außenplanarer Graphen

- Klauere den Beweis für 5-Listenfärbbarkeit
- Fall 1. Keine Sehne \Rightarrow Der Graph ist ein Kreis + Bäume.
 - gerade Länge \Rightarrow 2-färbbar
 - ungerade Länge \Rightarrow 3-färbbar
- Fall 2. Graph hat Sehne $e = (u, v)$
 - Zerlege Graph an e in G_1, G_2 , färbe rekursiv

2.2 – Färben außenplanarer Graphen

- Klau den Beweis für 5-Listenfärbbarkeit
- Fall 1. Keine Sehne \Rightarrow Der Graph ist ein Kreis + Bäume.
 - gerade Länge \Rightarrow 2-färbbar
 - ungerade Länge \Rightarrow 3-färbbar
- Fall 2. Graph hat Sehne $e = (u, v)$
 - Zerlege Graph an e in G_1, G_2 , färbe rekursiv
 - Permutiere die Färbung von G_2 sodass u (v) in G_1 und G_2 die gleiche Farbe hat

2.2 – Färben außenplanarer Graphen

- Klau den Beweis für 5-Listenfärbbarkeit
- Fall 1. Keine Sehne \Rightarrow Der Graph ist ein Kreis + Bäume.
 - gerade Länge \Rightarrow 2-färbbar
 - ungerade Länge \Rightarrow 3-färbbar
- Fall 2. Graph hat Sehne $e = (u, v)$
 - Zerlege Graph an e in G_1, G_2 , färbe rekursiv
 - Permutiere die Färbung von G_2 sodass u (v) in G_1 und G_2 die gleiche Farbe hat
 - u und v haben unterschiedliche Farben. \Rightarrow Legt 2 Farben in G_2 fest.
 \Rightarrow actually 3

3.1 – Färbung von Graphen

Für einen Graphen G bezeichnet $\chi(G)$ die minimale Anzahl von Farben, die nötig ist um G so zu färben, dass benachbarte Knoten verschiedene Farben haben.

Zeigen Sie: Für jeden Graphen mit Maximalgrad Δ gilt $\chi(G) \leq \Delta + 1$.

3.1 – Färbung von Graphen

Für einen Graphen G bezeichnet $\chi(G)$ die minimale Anzahl von Farben, die nötig ist um G so zu färben, dass benachbarte Knoten verschiedene Farben haben.

Zeigen Sie: Für jeden Graphen mit Maximalgrad Δ gilt $\chi(G) \leq \Delta + 1$.

- Färbe die Knoten iterativ.
- Gib jedem Knoten eine Farbe, die noch keiner seiner Nachbarn hat.
- Es gibt maximal Δ Nachbarn.
- \Rightarrow Es gibt immer eine Farbe, die noch nicht verwendet wird.

3.2 – Färbung von Graphen

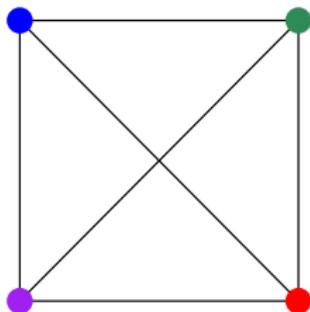
Für einen Graphen G bezeichnet $\chi(G)$ die minimale Anzahl von Farben, die nötig ist um G so zu färben, dass benachbarte Knoten verschiedene Farben haben.

Versuchen Sie Familien von Graphen anzugeben, für die $\chi(G) = \Delta + 1$ gilt.

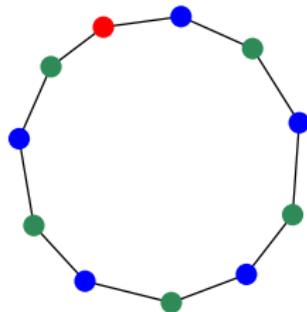
3.2 – Färbung von Graphen

Für einen Graphen G bezeichnet $\chi(G)$ die minimale Anzahl von Farben, die nötig ist um G so zu färben, dass benachbarte Knoten verschiedene Farben haben.

Versuchen Sie Familien von Graphen anzugeben, für die $\chi(G) = \Delta + 1$ gilt.



K_n



Kreise ungerader Länge

Zeigen Sie

Ein Graph G ist genau dann 2-färbbar, wenn G keine Kreise ungerader Länge enthält.

Zeigen Sie

Ein Graph G ist genau dann 2-färbbar, wenn G keine Kreise ungerader Länge enthält.

“ \Rightarrow ”

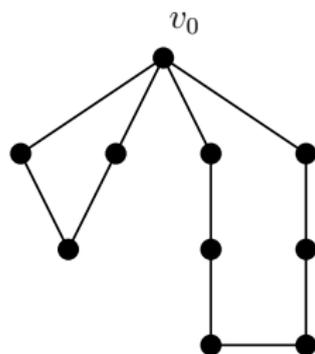
- Angenommen wir haben eine 2-Färbung von G , mit Knotenmenge R ist rot und Knotenmenge B ist blau.
- Jede Kante führt von R nach B oder von B nach R .
- Um mit einem Pfad durch G am Startknoten zu enden, muss man eine gerade Anzahl an Schritten gemacht haben.

Zeigen Sie

Ein Graph G ist genau dann 2-färbbar, wenn G keine Kreise ungerader Länge enthält.

“ \Leftarrow ”

- Angenommen jeder Kreis in G hat gerade Länge.
- Für jede Zusammenhangskomponente:
 - Wähle beliebigen Knoten v_0 .
 - $d(v)$: Abstand von v zu v_0 .
 - Färbe v mit $d(v)$ gerade rot, sonst blau.
 - Wäre die Färbung ungültig, wird ein Kreis ungerader Länge induziert.

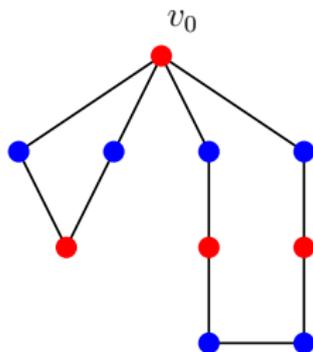


Zeigen Sie

Ein Graph G ist genau dann 2-färbbar, wenn G keine Kreise ungerader Länge enthält.

“ \Leftarrow ”

- Angenommen jeder Kreis in G hat gerade Länge.
- Für jede Zusammenhangskomponente:
 - Wähle beliebigen Knoten v_0 .
 - $d(v)$: Abstand von v zu v_0 .
 - Färbe v mit $d(v)$ gerade rot, sonst blau.
 - Wäre die Färbung ungültig, wird ein Kreis ungerader Länge induziert.

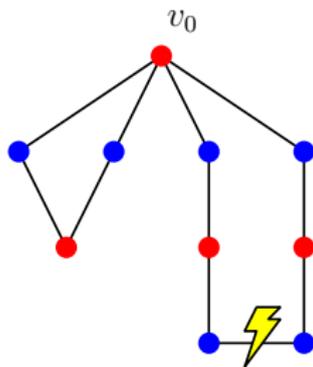


Zeigen Sie

Ein Graph G ist genau dann 2-färbbar, wenn G keine Kreise ungerader Länge enthält.

“ \Leftarrow ”

- Angenommen jeder Kreis in G hat gerade Länge.
- Für jede Zusammenhangskomponente:
 - Wähle beliebigen Knoten v_0 .
 - $d(v)$: Abstand von v zu v_0 .
 - Färbe v mit $d(v)$ gerade rot, sonst blau.
 - Wäre die Färbung ungültig, wird ein Kreis ungerader Länge induziert.



Zeigen Sie

Ein zweifach zusammenhängender planarer Graph (mit fester Einbettung) ist bipartit genau dann wenn sein Dualgraph Eulersch ist.

Hinweise

- Eulersch $\Leftrightarrow \forall v \in V \deg(v) \pmod 2 \equiv 0$
- zweifach zusammenhängend \Rightarrow jede Facette ist von einem Kreis begrenzt. Also keine Brücken oder angehängte Bäume.

Zeigen Sie

2-färbbar \Leftrightarrow Dual eulersch

Zeigen Sie

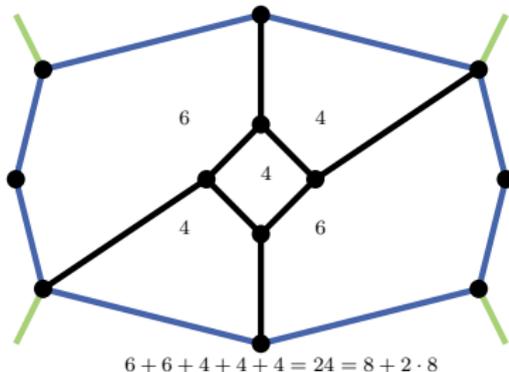
2-färbbar \Leftrightarrow Dual eulersch

" \Rightarrow " Jeder Kreis hat gerade Länge \Rightarrow Jede Facette hat geraden Grad \Rightarrow
Dual eulersch

4.2 – Euler

”←”

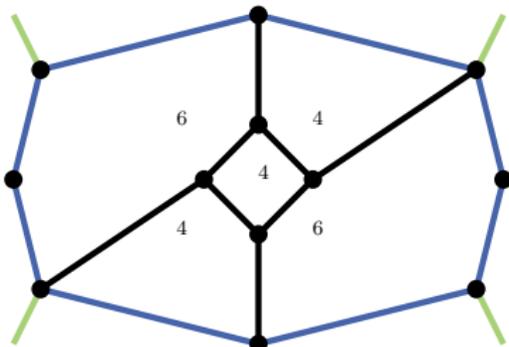
- Sei C ein Kreis. Zähle Facettengrade im Inneren von C .



4.2 – Euler

”←”

- Sei C ein Kreis. Zähle Facettengrade im Inneren von C .
- Kanten auf C 1x, Kanten im Inneren 2x



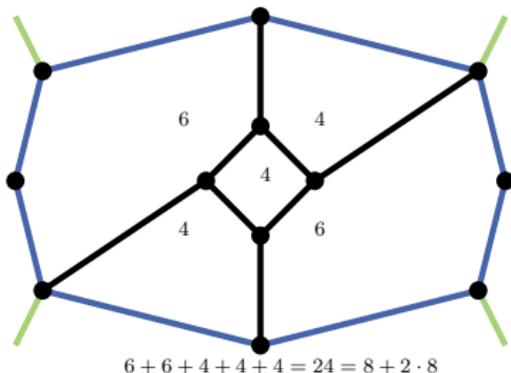
$$6 + 6 + 4 + 4 + 4 + 4 = 24 = 8 + 2 \cdot 8$$

$$\sum_{\text{face } f \in \text{inside}(C)} \underbrace{|f|}_{\text{gerade}} = |C| + 2|E \cap \text{inside}(C)|$$

4.2 – Euler

”←”

- Sei C ein Kreis. Zähle Facettengrade im Inneren von C .
- Kanten auf C 1x, Kanten im Inneren 2x



$$\sum_{\text{face } f \in \text{inside}(C)} \underbrace{|f|}_{\text{gerade}} = |C| + 2|E \cap \text{inside}(C)|$$

$$\Rightarrow |C| \text{ gerade} \Rightarrow \text{Graph bipartit}$$

Adjazentztest in $\mathcal{O}(1)$

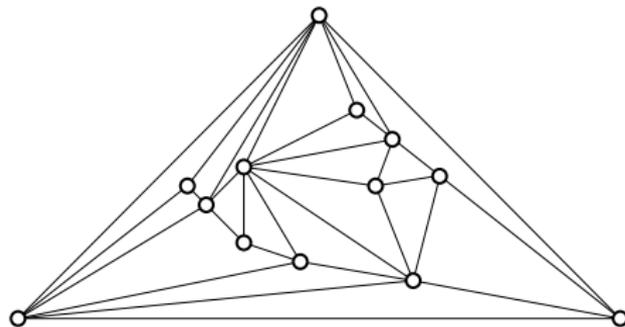
- Gegeben: G ungerichteter, planarer Graph.
- Wir haben lineare Vorberechnungszeit und können linear viel zusätzlichen Speicher benutzen.

Adjazenztest in $\mathcal{O}(1)$

- Gegeben: G ungerichteter, planarer Graph.
- Wir haben lineare Vorberechnungszeit und können linear viel zusätzlichen Speicher benutzen.

Hinweis: Richte Kanten so, dass jeder Knoten höchstens fünf ausgehende Kanten hat.

- In den folgenden Algorithmen werden wir Kanten *richten*.
- Kanten $\{u, v\}$ sind zunächst ungerichtet und können zu (u, v) oder (v, u) gerichtet werden.
- $\mathcal{N}(v)$: Nachbarschaft von v im Eingabegraphen.
- $\tilde{\mathcal{N}}(v)$: Nachbarschaft von v über Kanten die noch nicht gerichtet wurden.
- $\mathcal{N}^+(v)$: Über gerichtete, von v ausgehende Kanten benachbarte Knoten.



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

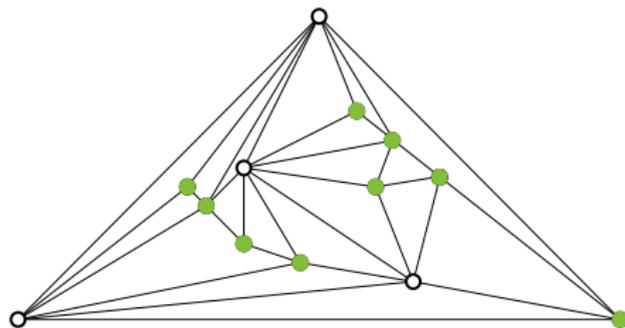
if $d_u = 5$ **then**

$Q.push(u)$

end if

end for

end while



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

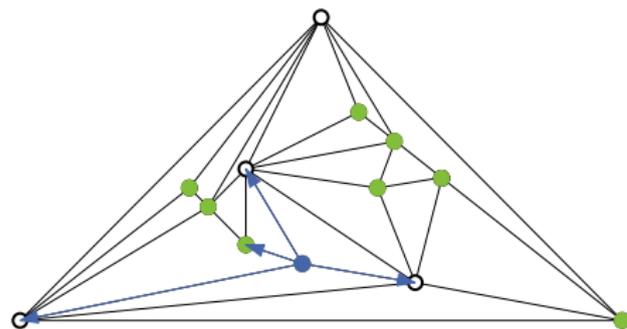
if $d_u = 5$ **then**

$Q.push(u)$

end if

end for

end while



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

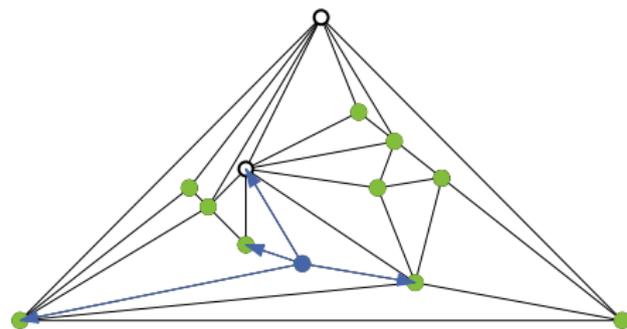
if $d_u = 5$ **then**

$Q.push(u)$

end if

end for

end while



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

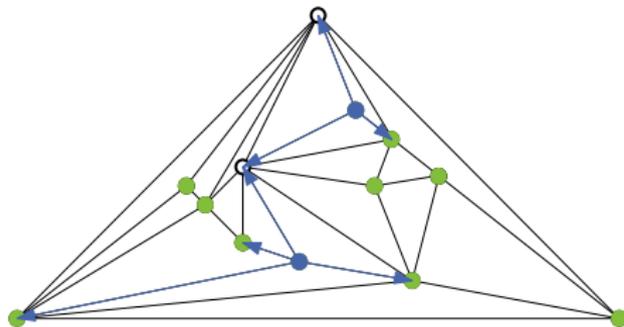
if $d_u = 5$ **then**

$Q.push(u)$

end if

end for

end while



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

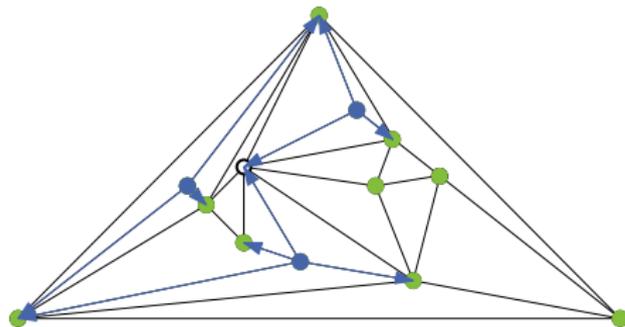
if $d_u = 5$ **then**

$Q.push(u)$

end if

end for

end while



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

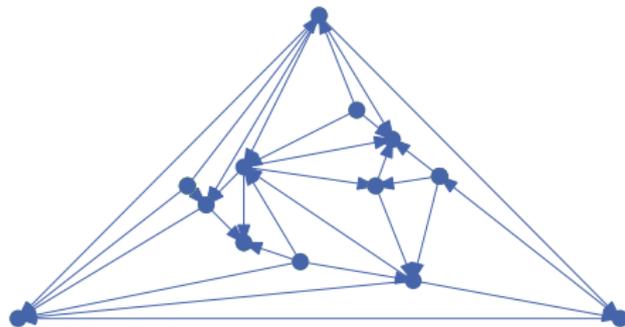
if $d_u = 5$ **then**

$Q.push(u)$

end if

end for

end while



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

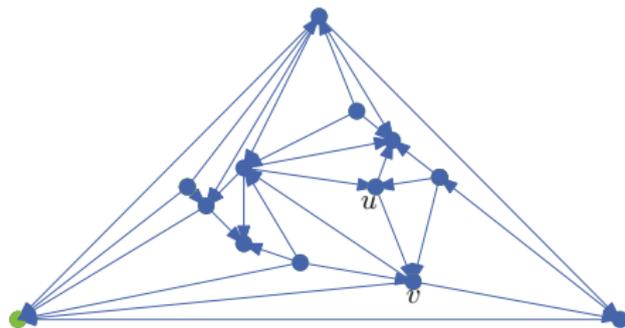
if $d_u = 5$ **then**

$Q.push(u)$

end if

end for

end while



Algorithm VORBERECHNUNG

$d_v \leftarrow$ Grad von Knoten v

$Q \leftarrow \{v \mid d_v \leq 5\}$

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.pop()$

for $u \in \tilde{\mathcal{N}}(v)$ **do**

$\{v, u\} \rightsquigarrow (v, u)$

$d_u = d_u - 1$

if $d_u = 5$ **then**

$Q.push(u)$

end if

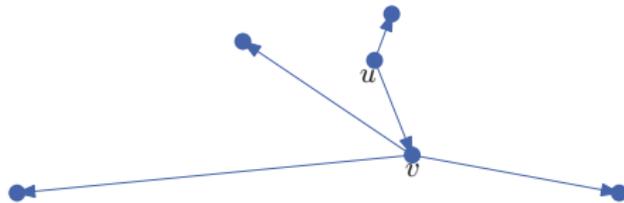
end for

end while

Algorithm ADJAZENT(u, v)

$v \stackrel{?}{\in} \mathcal{N}^+(u)$ oder

$u \stackrel{?}{\in} \mathcal{N}^+(v)$



- Für jeden der beiden Knoten müssen höchstens 5 Kanten betrachtet werden.
- \Rightarrow ADJAZENT $\in \mathcal{O}(1)$