Algorithmen für Planare Graphen

13. Juli 2021, Übung 5

Lars Gottesbüren

INSTITUT FÜR THEORETISCHE INFORMATIK

Perfektes Matching

Karlsruher Institut für Technologie

Ein Matching M zu einem Graphen G heißt *perfekt*, falls jeder Knoten von G zu einer Kante aus M inzident ist. Für welche $n \geq 1$ und $m \geq 1$ besitzen die folgenden Graphen jeweils ein perfektes Matching?

- P_n (der Graph bestehend aus einem einfachen Weg mit n Knoten)
- C_n (der Graph bestehend aus einem einfachen Kreis mit n Knoten). Definiere ausnahmsweise C_2 als K_2.
- Q_n (der Hyperwürfel mit n-Bit Knoten IDs, siehe Übungsblatt 1)
- K_n
- $K_{n,m}$

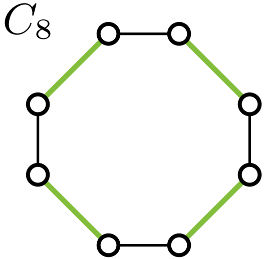
Perfektes Matching

- P_n einfacher Weg mit n Knoten.
- n gerade

Perfektes Matching

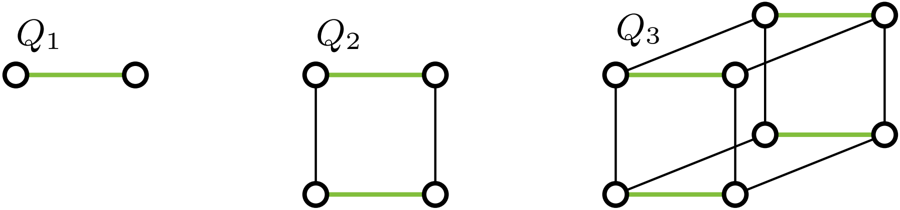
C_n einfacher Kreis mit n Knoten. Ausnahmsweise $C_2 = K_2$.

- n gerade



Perfektes Matching

Q_n der n-te Hyperwürfel.

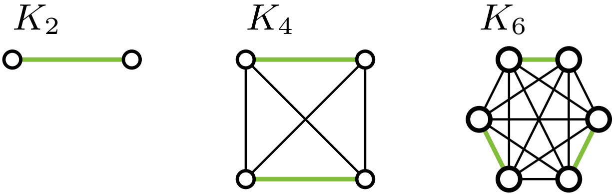


Kopiere das Matching mit dem Graphen.

Perfektes Matching

K_n

- n gerade

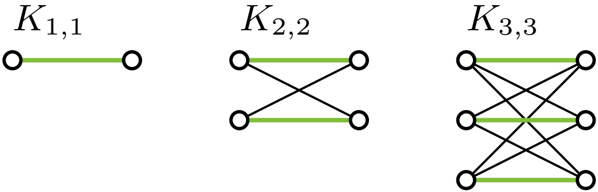


- Es muss gelten: $|M| = \frac{|V|}{2} \Rightarrow n$ gerade
- K_n enthält Kreis der Länge n.

Perfektes Matching

$K_{n,m}$

- $n = m$
- Für $n \neq m$ ist mindestens ein Knoten auf einer Seite nicht im Matching.

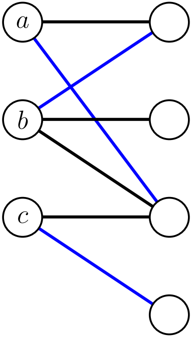


Füge die Kante zwischen den beiden neuen Knoten zu M hinzu.

Erhöhende Wege

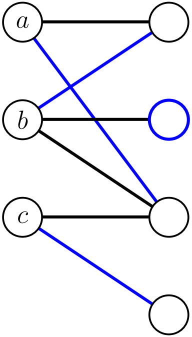
Sei $G = (V_1 \cup V_2, E)$ ein bipartiter Graph, $v \in V_1$ und M' ein max Matching für $G - v$. Erweitere M' zu einem max Matching für G in Linearzeit.

Erhöhende Wege



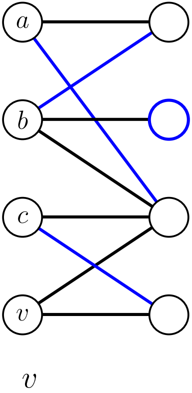
- Bestimme erhöhenden Weg bzgl. M' mit Endknoten v.
- Algorithmus soll in $\mathcal{O}(m)$ liegen.
- *Hinweis:* Modifizieren Sie eine Breitensuche mit Startknoten v.

Erhöhende Wege



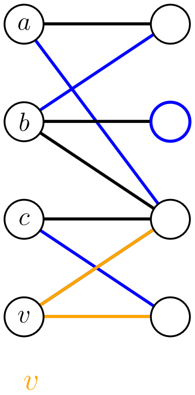
- Markiere ungematchte Knoten in $G - v$

Erhöhende Wege



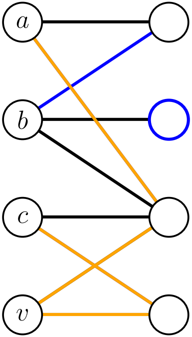
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.

Erhöhende Wege



- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.

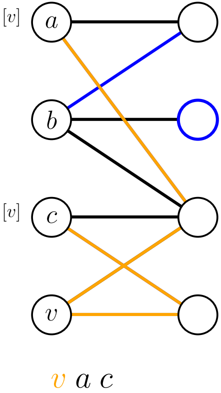
Erhöhende Wege



$v\ a\ c$

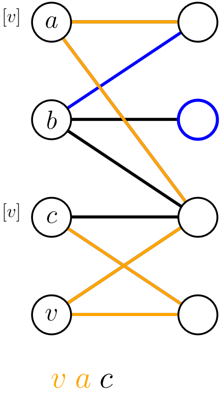
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten

Erhöhende Wege



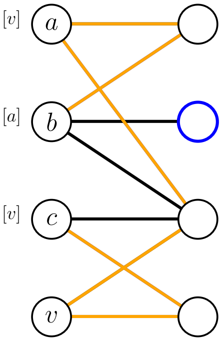
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten

Erhöhende Wege



- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten

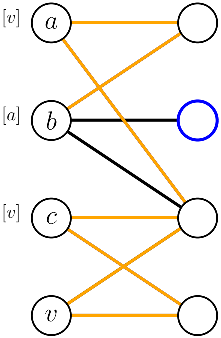
Erhöhende Wege



$v\ a\ c\ b$

- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten

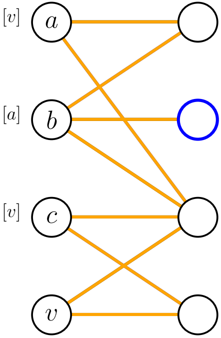
Erhöhende Wege



$v\ a\ c\ b$

- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten

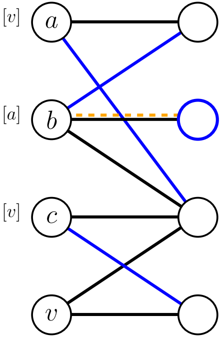
Erhöhende Wege



$v\ a\ c\ b$

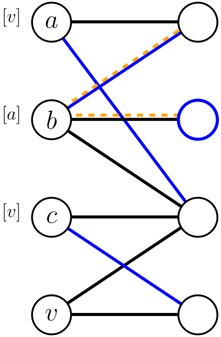
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten
- Brich ab, wenn markierter Knoten gefunden wird.

Erhöhende Wege



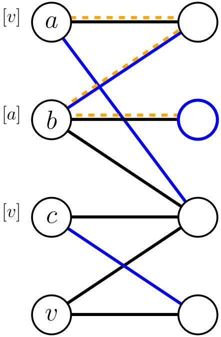
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten
- Brich ab, wenn markierter Knoten gefunden wird.
- Rekonstruiere erhöhenden Pfad.

Erhöhende Wege



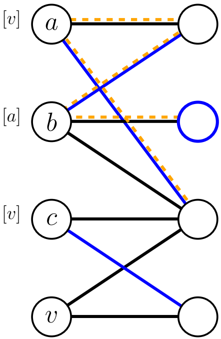
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten
- Brich ab, wenn markierter Knoten gefunden wird.
- Rekonstruiere erhöhenden Pfad.

Erhöhende Wege



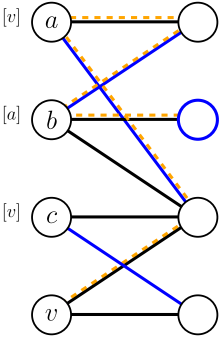
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten
- Brich ab, wenn markierter Knoten gefunden wird.
- Rekonstruiere erhöhenden Pfad.

Erhöhende Wege



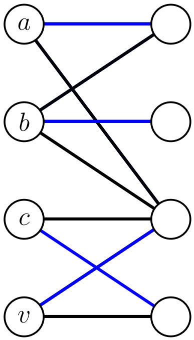
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten
- Brich ab, wenn markierter Knoten gefunden wird.
- Rekonstruiere erhöhenden Pfad.

Erhöhende Wege



- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten
- Brich ab, wenn markierter Knoten gefunden wird.
- Rekonstruiere erhöhenden Pfad.

Erhöhende Wege



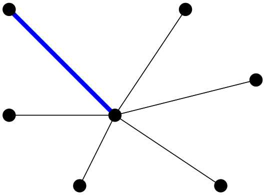
- Markiere ungematchte Knoten in $G - v$
- BFS beginnend bei v.
- $\rightarrow$ nur nicht-Matching-Kanten.
- $\leftarrow$ nur Matching-Kanten
- Brich ab, wenn markierter Knoten gefunden wird.
- Rekonstruiere erhöhenden Pfad.

Große und kleine Matchings

Geben Sie für jedes $n \geq 2$ einen zusammenhängenden Graphen an, mit kardinalitätsmaximalem Matching der Größe: 1 und $\lfloor \frac{n}{2} \rfloor$

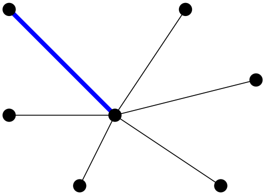
Große und kleine Matchings

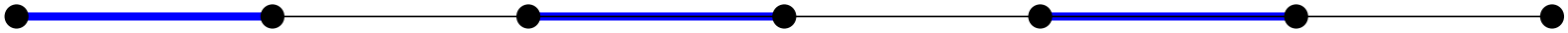
Geben Sie für jedes $n \geq 2$ einen zusammenhängenden Graphen an, mit kardinalitätsmaximalem Matching der Größe: 1 und $\lfloor \frac{n}{2} \rfloor$



Große und kleine Matchings

Geben Sie für jedes $n \geq 2$ einen zusammenhängenden Graphen an, mit kardinalitätsmaximalem Matching der Größe: 1 und $\lfloor \frac{n}{2} \rfloor$





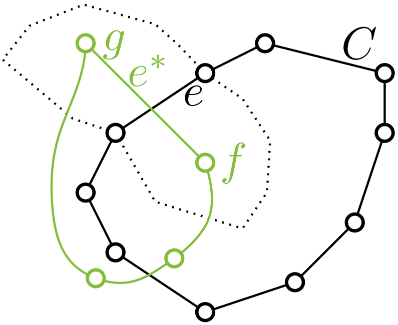
Schnitte und Kreise

Aufgabe a)

Sei $G = (V, E)$ planar, zusammenhängend und $E' \subseteq E$.
Zeige (V, E') enthält Kreis $C \Leftrightarrow (V^*, E^* - E'^*)$ unzusammenhängend

“$\Rightarrow$“ Sei C ein Kreis in E'

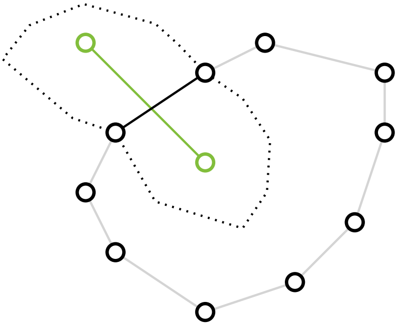
- Sei $f \in V^*$ im Inneren, $g \in V^*$ im Äußeren von C
- Alle Pfade in G^* die f und g verbinden, laufen über Kanten dual zu C



Schnitte und Kreise

Aufgabe a)

Sei $G = (V, E)$ planar, zusammenhängend und $E' \subseteq E$.
Zeige (V, E') enthält Kreis $C \Leftrightarrow (V^*, E^* - E'^*)$ unzusammenhängend



"$\Leftarrow$" Annahme E' enhält keinen Kreis.

- Facetten sind durch Kreise voneinander getrennt
- Kein Kreis gelöscht $\Rightarrow$ kein Schnitt zwischen Facetten

Bäume

Aufgabe b)

Zeige: Für $E' \subseteq E$

$$
\begin{array}{rrll}
 & T = & (V, E') & \text{ist ein Spannbaum von } G \\
\Leftrightarrow & T^* = & (V^*, E^* - E'^*) & \text{ist ein Spannbaum von } G^*
\end{array}
$$

Nutze Aufgabe a)

"$\Rightarrow$"

- T Spannbaum $\Rightarrow$ T ist kreisfrei $\Rightarrow$ T^* ist zshg
- $n - m + f = 2 \Rightarrow n - m = -f + 2$
- $|E^* - E'^*| = m - (n-1) = -(n-m) + 1 = (f-2) + 1 = f - 1$

Bäume

Aufgabe b)

Zeige: Für $E' \subseteq E$

$$\begin{array}{cll} & T = (V, E') & \text{ist ein Spannbaum von } G \\ \Leftrightarrow & T^* = (V^*, E^* - E'^*) & \text{ist ein Spannbaum von } G^* \end{array}$$

Nutze Aufgabe a)

"$\Rightarrow$"

- T Spannbaum $\Rightarrow$ T ist kreisfrei $\Rightarrow$ T^* ist zshg
- $n - m + f = 2 \Rightarrow n - m = -f + 2$
- $|E^* - E'^*| = m - (n-1) = -(n-m) + 1 = (f-2) + 1 = f - 1$

Bäume

Aufgabe b)

Zeige: Für $E' \subseteq E$

$$\begin{array}{llll} & T = & (V, E') & \text{ist ein Spannbaum von } G \\ \Leftrightarrow & T^* = & (V^*, E^* - E'^*) & \text{ist ein Spannbaum von } G^* \end{array}$$

Nutze Aufgabe a)

"$\Rightarrow$"

- T Spannbaum $\Rightarrow$ T ist kreisfrei $\Rightarrow$ T^* ist zshg
- $n - m + f = 2 \Rightarrow n - m = -f + 2$
- $|E^* - E'^*| = m - (n-1) = -(n-m) + 1 = (f-2) + 1 = f - 1$

Bäume

Aufgabe b)

Zeige: Für $E' \subseteq E$

$$\begin{array}{rlll} & T = & (V, E') & \text{ist ein Spannbaum von } G \\ \Leftrightarrow & T^* = & (V^*, E^* - E'^*) & \text{ist ein Spannbaum von } G^* \end{array}$$

Nutze Aufgabe a)

"$\Rightarrow$"

- T Spannbaum $\Rightarrow$ T ist kreisfrei $\Rightarrow$ T^* ist zshg
- $n - m + f = 2 \Rightarrow n - m = -f + 2$
- $|E^* - E'^*| = m - (n-1) = -(n-m) + 1 = (f-2) + 1 = f - 1$

"$\Leftarrow$" Analog.

Dreiecke zählen

- Sei G ein Graph mit n Knoten und m Kanten.
- Der k-Core von G ist der maximale Subgraph in dem jeder Knoten mindestens Grad k hat.
- Die Core-Zerlegung weist jedem Knoten das maximale k zu für welches er im k-Core liegt.
- Die Entartetheit $D(G)$ ist das größte k für welches G einen nicht-leeren k-Core hat.
- $\mathcal{N}(v)$: Nachbarschaft von v in G.

k-Core

- Geben Sie einen Algorithmus an, der die Core-Zerlegung in $\mathcal{O}(n + m)$ berechnet.

k-Core

Algorithm CORE DECOMPOSITION

$\delta(v) \leftarrow$ Grad von Knoten v

$\Delta \leftarrow max\{\delta(v) \mid v \in V\}$

Sortiere die Knoten nach $\delta(v)$ in *Buckets* $b_0, \ldots, b_\Delta$

for $b_k \in (b_0, \ldots, b_\Delta)$ **do**

  while $b_k \neq \emptyset$ **do**

    $v \leftarrow b_k.pop()$

    $\textsc{core}[v] \leftarrow k$

    for $u \in N(v)$ **do**

      $b_j \leftarrow$ Bucket in dem u liegt

      if $j > k$ **then**

        Verschiebe v von b_j nach b_{j-1}

$D(G) \leftarrow max\{\textsc{core}[v] \mid v \in V\}$

Dreiecke zählen

Aufgabe

Geben Sie einen Algorithmus an der die Anzahl Dreiecke in G in $\mathcal{O}((n+m)\cdot D(G))$ berechnet.
Hinweis: Modifizieren Sie den Algorithmus von Übungsblatt 4.

- Dreiecke im gesamten Graphen, nicht pro Knoten
- $\mathcal{N}(v)$ Nachbarschaft von v
- $\mathcal{N}^+(v)$ Nachbarn von v über orientierte Kanten (v, u)
- $\widetilde{\mathcal{N}}(v)$ Nachbarschaft von v über noch nicht orientierte Kanten

Dreiecke zählen

Karlsruher Institut für Technologie

Aufgabe

Geben Sie einen Algorithmus an der die Anzahl Dreiecke in G in $\mathcal{O}((n+m) \cdot D(G))$ berechnet.
Hinweis: Modifizieren Sie den Algorithmus von Übungsblatt 4.

- Dreiecke im gesamten Graphen, nicht pro Knoten
- $\mathcal{N}(v)$ Nachbarschaft von v
- $\mathcal{N}^+(v)$ Nachbarn von v über orientierte Kanten (v, u)
- $\widetilde{\mathcal{N}}(v)$ Nachbarschaft von v über noch nicht orientierte Kanten

Dreiecke zählen

Algorithm KANTEN-ORIENTIERUNG

Sei $v_1, \ldots, v_n$ die Knotenreihenfolge in der sie bei der Core-Zerlegung gelöscht wurden

for $\{v_i, v_j\} \in E$ **do**

  if $j > i$ **then**

    Orientiere $\{v_i, v_j\} \to (v_i, v_j)$

  else

    Orientiere $\{v_i, v_j\} \to (v_j, v_i)$

$\Rightarrow \forall v \in V : |\mathcal{N}^+(v)| \leq D(G)$

Dreiecke zählen

Algorithm KANTEN-ORIENTIERUNG

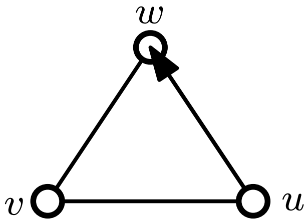
Sei $v_1, \ldots, v_n$ die Knotenreihenfolge in der sie bei der Core-Zerlegung gelöscht wurden
for $\{v_i, v_j\} \in E$ **do**
  if $j > i$ **then**
    Orientiere $\{v_i, v_j\} \to (v_i, v_j)$
  else
    Orientiere $\{v_i, v_j\} \to (v_j, v_i)$

$\Rightarrow \forall v \in V : |\mathcal{N}^+(v)| \leq D(G)$

Dreiecke zählen

Algorithm DREIECKE

KANTEN-ORIENTIERUNG()
$D = 0$
for $v \in V$ **do**
  for $u \in \mathcal{N}(v)$ **do**
    for $w \in \mathcal{N}^+(u)$ **do**
      if ADJAZENT(v, w) **then**
        $D = D + 1$
return $\frac{D}{3}$



Dreiecke zählen

INIT_ADJAZENZ()

- Initialisiere Array

SET_ADJAZENZ(v)

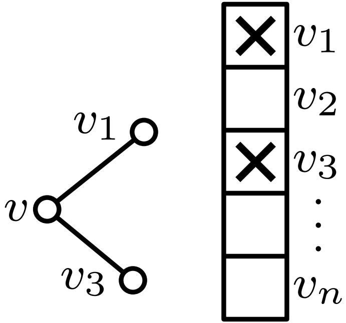
- Setze Bits in $\mathcal{N}(v)$

UNSET_ADJAZENZ(v)

- Entferne Bits in $\mathcal{N}(v)$

ADJAZENT(v, w)

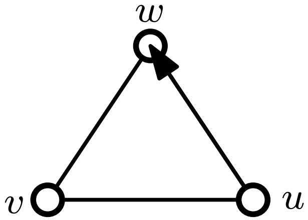
- Ist Bit an Stelle w gesetzt?



Dreiecke zählen

Algorithm DREIECKE

```
KANTEN-ORIENTIERUNG()
INIT_ADJAZENZ()
D = 0
for v ∈ V do
    SET_ADJAZENZ(v)
    for u ∈ N(v) do
        for w ∈ N⁺(u) do
            if ADJAZENT(v, w) then
                D = D + 1
    UNSET_ADJAZENZ(v)
return D/3
```



Entartetheit von planaren Graphen

Zeigen Sie, dass für planare Graphen G $D(G) \leq 5$ gilt.

- In planaren Graphen gibt es mindestens einen Knoten v mit $\delta(v) \leq 5$.
- $G - v$ ist wieder planar.

Entartetheit von planaren Graphen

Zeigen Sie, dass für planare Graphen G $D(G) \leq 5$ gilt.

- In planaren Graphen gibt es mindestens einen Knoten v mit $\delta(v) \leq 5$.
- $G - v$ ist wieder planar.