

Algorithmen für Planare Graphen

12. Juni 2018, Übung 4

Lars Gottesbüren, Michael Hamann

INSTITUT FÜR THEORETISCHE INFORMATIK



- 11. Juli - AUSGEBUCHT
- 31. Juli - AUSGEBUCHT
- 28. August - AUSGEBUCHT
- 29. August - nur noch wenige Plätze
- 27. September
- 4. Oktober
- 15. Oktober

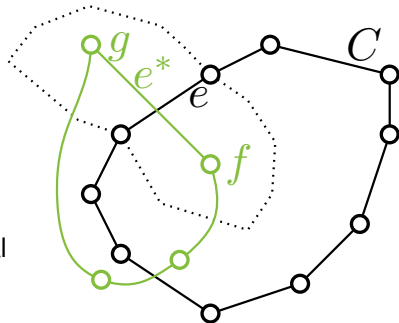
Aufgabe 1.1

Sei $G = (V, E)$ planar, zusammenhängend und $E' \subseteq E$.

Zeige (V, E') enthält Kreis $C \Leftrightarrow (V^*, (E \setminus E')^*)$ 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



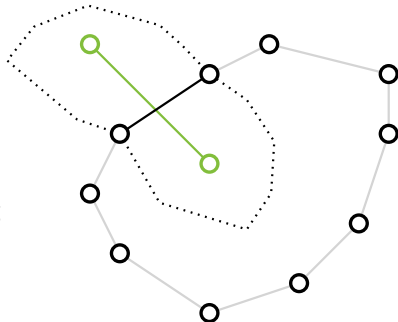
Aufgabe 1.1

Sei $G = (V, E)$ planar, zusammenhängend und $E' \subseteq E$.

Zeige (V, E') enthält Kreis $C \Leftrightarrow (V^*, (E \setminus E')^*)$ unzusammenhängend

“ \Leftarrow “ Annahme E' enthält keinen Kreis.

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



Aufgabe 1.2 – Bäume

Zeige: Für $E' \subseteq E$

$T = (V, E')$ ist ein Spannbaum von G

$\Leftrightarrow T^* = (V^*, (E \setminus E')^*)$ ist ein Spannbaum von G^*

Nutze Aufgabe 1.1

“ \Rightarrow “

T Spannbaum $\Rightarrow T$ ist zshg [kreisfrei] $\Rightarrow T^*$ ist kreisfrei [zshg]

“ \Leftarrow “

Analog.

2.1 – Perfektes Matching

Ein Matching M heißt *perfekt*, wenn 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 einfacher Weg mit n Knoten.
- n gerade



2.1 – Perfektes Matching

Ein Matching M heißt *perfekt*, wenn 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?

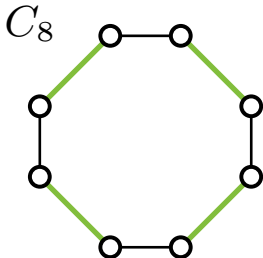
- P_n einfacher Weg mit n Knoten.
- n gerade



2.2 – Perfektes Matching

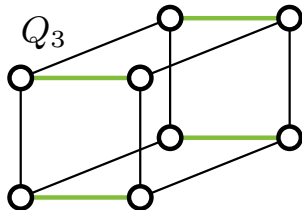
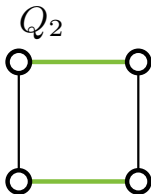
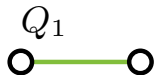
C_n einfacher Kreis mit n Knoten. **Ausnahmsweise $C_2 = K_2$.**

- n gerade



2.3 – Perfektes Matching

Q_n der n -te Hyperwürfel.



Kopiere das Matching mit dem Graphen.

2.4 – Perfektes Matching

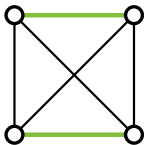
K_n

- n gerade

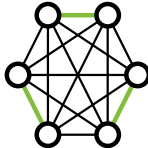
K_2



K_4



K_6

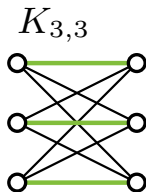
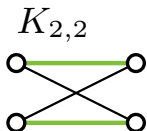
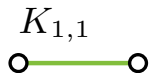


- Es muss gelten: $|M| = \frac{|V|}{2} \Rightarrow n$ gerade
- K_n enthält Kreis der Länge n .

2.4 – 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.

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

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

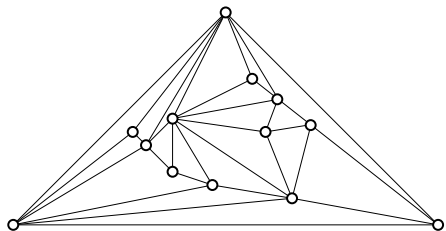
Hinweis: Richte Kanten so, dass jeder Knoten höchstens fünf ausgehende Kanten hat.

Adjazentztest 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.

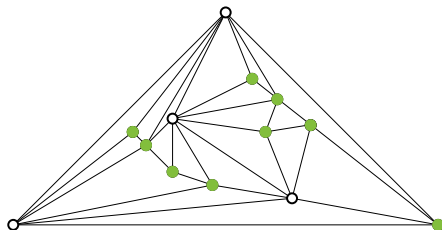


Algorithm VORBERECH-
NUNG

$d_v \leftarrow$ 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  
       $Q.push(u)$ 
```

3 – Adjazenztest

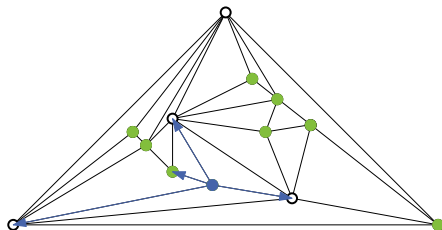


Algorithm VORBERECH-
NUNG

$d_v \leftarrow$ 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  
       $Q.push(u)$ 
```


3 – Adjazenztest

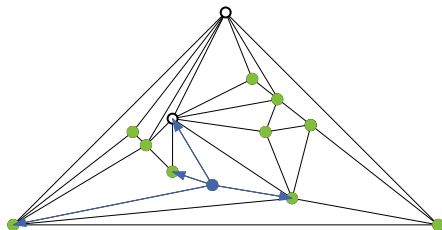


Algorithm VORBERECH-
NUNG

$d_v \leftarrow$ 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**
 $Q.push(u)$

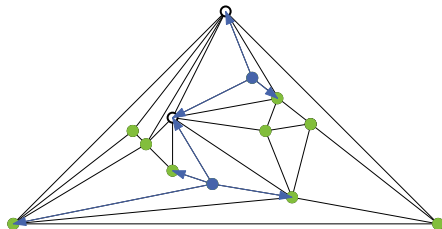


Algorithm VORBERECHNUNG

$d_v \leftarrow$ 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  
       $Q.push(u)$ 
```

3 – Adjazenztest

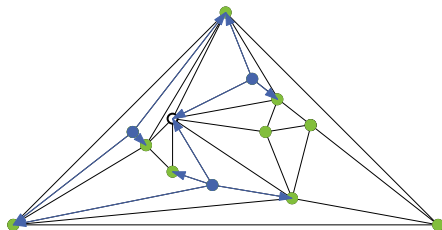


Algorithm VORBERECH-
NUNG

$d_v \leftarrow$ 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  
       $Q.push(u)$ 
```

3 – Adjazenztest

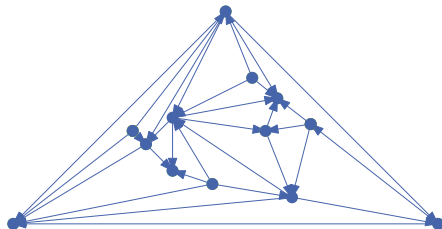


Algorithm VORBERECH-
NUNG

$d_v \leftarrow$ 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
       $Q.push(u)$ 
```

3 – Adjazenztest

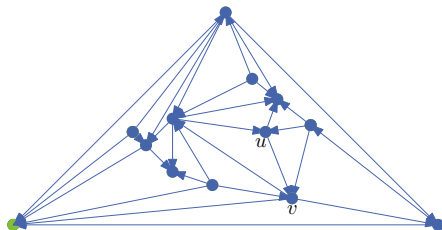


Algorithm VORBERECH-
NUNG

$d_v \leftarrow$ 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
       $Q.push(u)$ 
```

3 – Adjazenztest



Algorithm VORBERECH-
NUNG

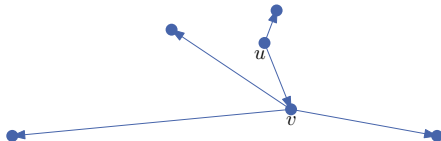
$d_v \leftarrow$ Grad von Knoten v
 $Q \leftarrow \{v \mid d_v \leq 5\}$

```
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
       $Q.push(u)$ 
```

Algorithm ADJAZENT(u, v)

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

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

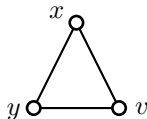


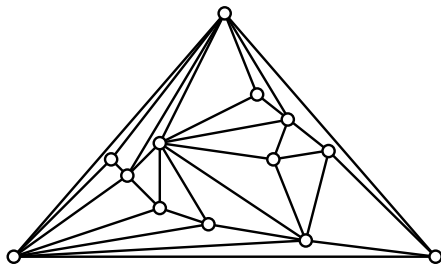
- Für jeden der beiden Knoten müssen höchstens 5 Kanten betrachtet werden.
- \Rightarrow ADJAZENT $\in \mathcal{O}(1)$

Für alle $v \in V$: bestimme Zahl der *Dreiecke* in denen v vorkommt.

Laufzeit: $\mathcal{O}(n)$

- Gegeben: $G = (V, E)$ ungerichteter, planarer Graph.
- Dreiecke in denen v vorkommt: $\{\{x, y\} \in E \mid \{v, x\}, \{v, y\} \in E\}$





Algorithm DREIECKE

VORBERECHNUNG()

```
for  $v \in V$  do
```

```
   $D_v = 0$ 
```

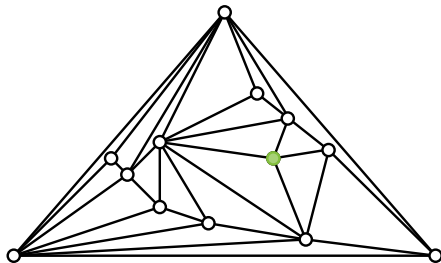
```
  for  $u \in \mathcal{N}(v)$  do
```

```
    for  $w \in \mathcal{N}^+(u)$  do
```

```
      if ADJAZENT( $v, w$ )
```

```
      then
```

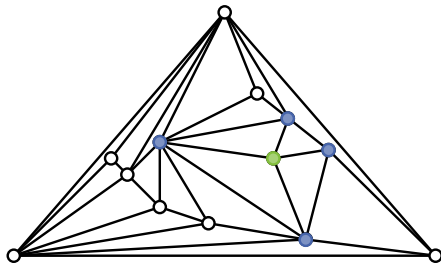
```
         $D_v = D_v + 1$ 
```



Algorithm DREIECKE

VORBERECHNUNG()

```
for  $v \in V$  do
   $D_v = 0$ 
  for  $u \in \mathcal{N}(v)$  do
    for  $w \in \mathcal{N}^+(u)$  do
      if ADJAZENT( $v, w$ )
      then
         $D_v = D_v + 1$ 
```



Algorithm DREIECKE

VORBERECHNUNG()

```
for  $v \in V$  do
```

```
   $D_v = 0$ 
```

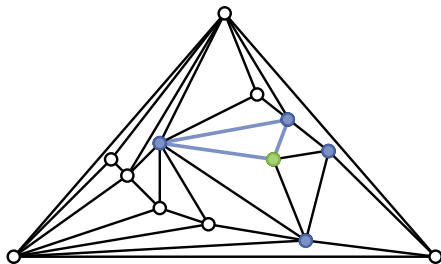
```
  for  $u \in \mathcal{N}(v)$  do
```

```
    for  $w \in \mathcal{N}^+(u)$  do
```

```
      if ADJAZENT( $v, w$ )
```

```
      then
```

```
         $D_v = D_v + 1$ 
```



Algorithm DREIECKE

VORBERECHNUNG()

```
for  $v \in V$  do
```

```
   $D_v = 0$ 
```

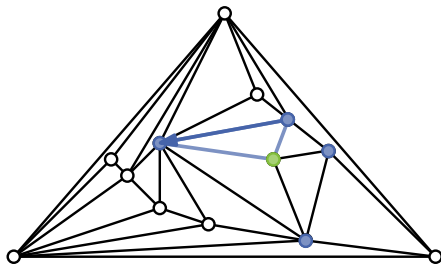
```
  for  $u \in \mathcal{N}(v)$  do
```

```
    for  $w \in \mathcal{N}^+(u)$  do
```

```
      if ADJAZENT( $v, w$ )
```

```
      then
```

```
         $D_v = D_v + 1$ 
```



Algorithm DREIECKE

VORBERECHNUNG()

```
for  $v \in V$  do
```

```
   $D_v = 0$ 
```

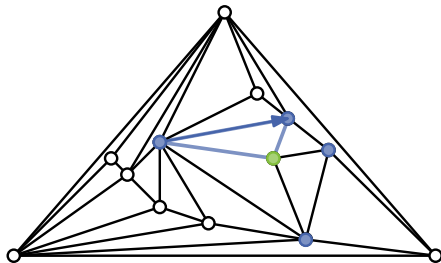
```
  for  $u \in \mathcal{N}(v)$  do
```

```
    for  $w \in \mathcal{N}^+(u)$  do
```

```
      if ADJAZENT( $v, w$ )
```

```
      then
```

```
         $D_v = D_v + 1$ 
```



Algorithm DREIECKE

VORBERECHNUNG()

```
for  $v \in V$  do
```

```
   $D_v = 0$ 
```

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

```
    for  $w \in \mathcal{N}^+(u)$  do
```

```
      if ADJAZENT( $v, w$ )
```

```
      then
```

```
         $D_v = D_v + 1$ 
```

MST in erwartet $\mathcal{O}(n)$

- Gegeben: G ungerichteter, planarer, zusammenhängend Graph.

Ohne Beweis zu verwenden:

- Sei v ein Knoten und e eine Kante minimalen Gewichts inzident zu v . Dann gibt es einen Spannbaum minimalen Gewichts von G , der e enthält.
- Sei e eine Kante, die einem Spannbaum minimalen Gewichts von G vorkommt. Sei T ein Spannbaum minimalen Gewichts auf dem Graphen, den man durch die Kontraktion von e erhält. Dann ist $T \cup e$ ein Spannbaum minimalen Gewichts auf G .

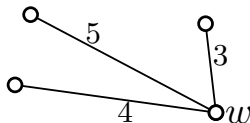
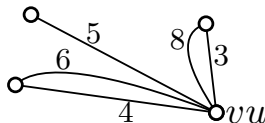
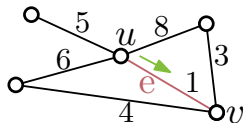
Algorithm MST

 $T \leftarrow \emptyset$
while G nicht leer **do**

 Wähle $v \in V$ mit $\deg(v) \leq 5$
 $e \leftarrow \operatorname{argmin}_{e' \in \mathcal{N}(v)} w(e')$
 $T \leftarrow T \cup e$

 Kontrahiere e

Entferne Multikanten und behalte nur die mit geringstem Gewicht



Kontraktion von $\{u, v\}$ mit $\deg(v) \leq 5$

- Pro Knoten eine Hashmap für seine Nachbarschaft
- Für $x \in \mathcal{N}(v)$ erkenne ob $x \in \mathcal{N}(u)$; in erwartet $\mathcal{O}(1)$
- Falls ja, behalte die Kante mit kleinerem Gewicht in Hashmap für u