

Algorithmen für Planare Graphen

11. Juli 2019, Übung 6

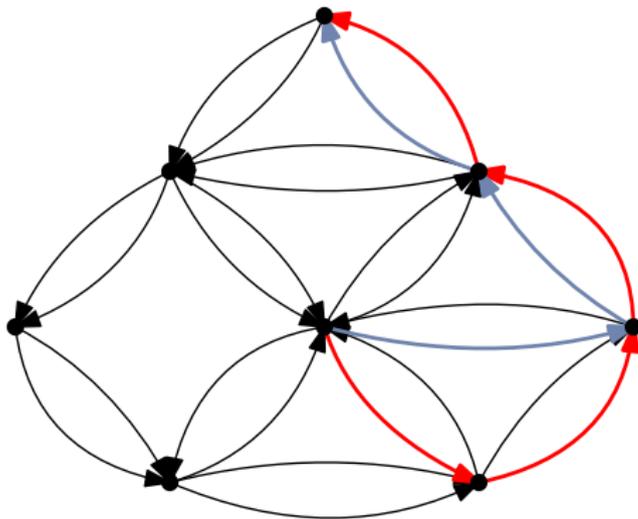
Guido Brückner

INSTITUT FÜR THEORETISCHE INFORMATIK



Effizient rechteste freie Kante finden

Hintergrund: finde st -Pfade möglichst weit rechts

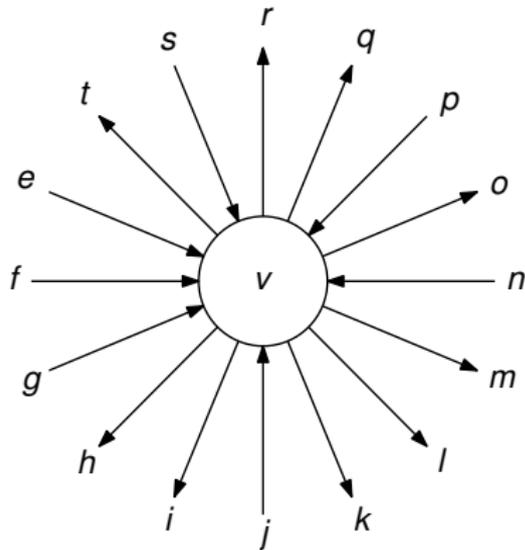


Effizient rechtteste freie Kante finden

Ziel: effizient rechtteste freie Kante finden

Naiv: fange bei einlaufender Kante e nach rechts an zu suchen, bis freie Kante f gefunden

$\Rightarrow \mathcal{O}(n)$ pro einlaufende Kante



Verwaltet dynamisch Partition von Menge \mathcal{U} mit folgenden Operationen:

- MAKE(x) für $x \in \mathcal{U}$: erstelle einelementige Menge $\{x\}$
- UNION(x, y) für $x, y \in \mathcal{U}$: vereinige S_x, S_y mit $x \in S_x, y \in S_y$
- FIND(x) für $x \in \mathcal{U}$: finde $S \subseteq \mathcal{U}$ mit $x \in S$

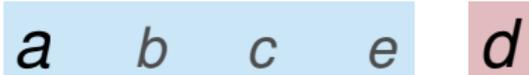
Effizient rechteste freie Kante finden

Verwaltet dynamisch Partition von Menge \mathcal{U} mit folgenden Operationen:

- MAKE(x) für $x \in \mathcal{U}$: erstelle einelementige Menge $\{x\}$
- UNION(x, y) für $x, y \in \mathcal{U}$: vereinige S_x, S_y mit $x \in S_x, y \in S_y$
- FIND(x) für $x \in \mathcal{U}$: finde $S \subseteq \mathcal{U}$ mit $x \in S$

$\forall x \in \{a, b, c, d, e\} : \text{MAKE}(x) \rightarrow$ 

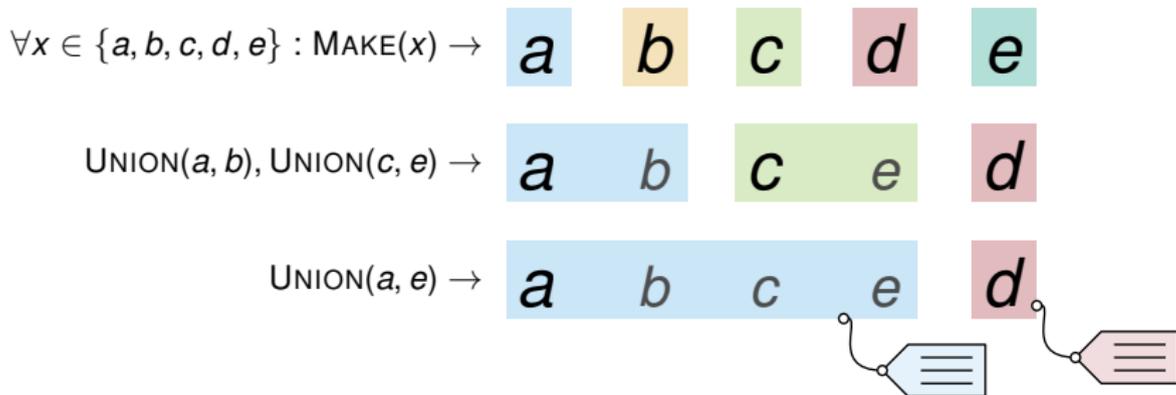
UNION(a, b), UNION(c, e) \rightarrow 

UNION(a, e) \rightarrow 

Effizient rechteste freie Kante finden

Verwaltet dynamisch Partition von Menge \mathcal{U} mit folgenden Operationen:

- MAKE(x) für $x \in \mathcal{U}$: erstelle einelementige Menge $\{x\}$
- UNION(x, y) für $x, y \in \mathcal{U}$: vereinige S_x, S_y mit $x \in S_x, y \in S_y$
- FIND(x) für $x \in \mathcal{U}$: finde $S \subseteq \mathcal{U}$ mit $x \in S$



Bemerkung: FIND(x) kann zusätzliche Attribute erhalten

Verwaltet dynamisch Partition von Menge \mathcal{U} mit folgenden Operationen:

- MAKE(x) für $x \in \mathcal{U}$: erstelle einelementige Menge $\{x\}$
- UNION(x, y) für $x, y \in \mathcal{U}$: vereinige S_x, S_y mit $x \in S_x, y \in S_y$
- FIND(x) für $x \in \mathcal{U}$: finde $S \subseteq \mathcal{U}$ mit $x \in S$

allgemeine Laufzeit:

- UNION(x, y) $\in \mathcal{O}(\alpha(n))$
- FIND(x, y) $\in \mathcal{O}(\alpha(n))$

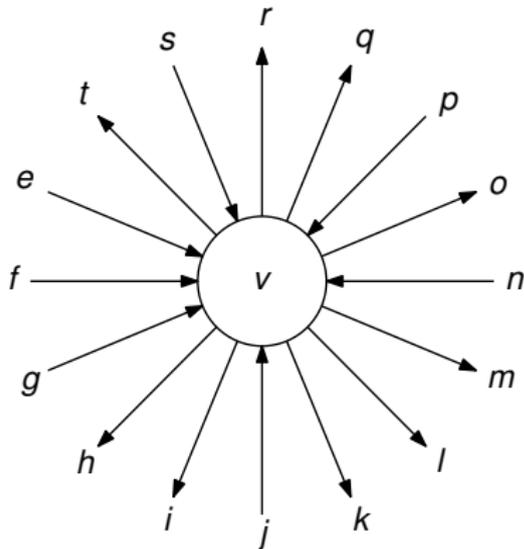
ziemlich gut: $\alpha \left(2^{2^{2^{2^{16}}}} \right) = 4$, aber nicht $\mathcal{O}(1)$

falls UNION-Struktur vorher bekannt: [Gabow '85]

- UNION(x, y) $\in \mathcal{O}(1)$
- FIND(x, y) $\in \mathcal{O}(1)$

Ziel: effizient rechtteste freie Kante finden

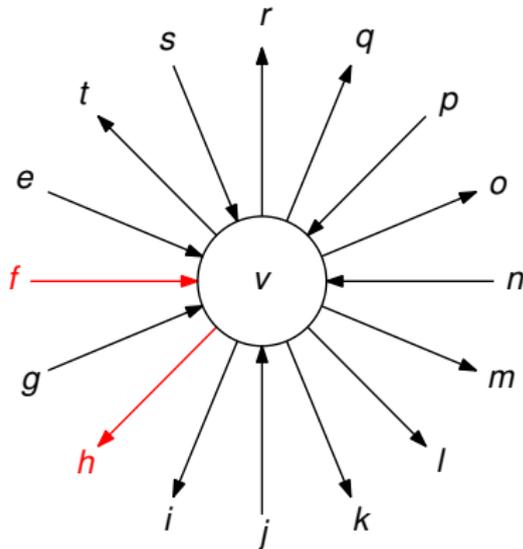
- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtteste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

Ziel: effizient rechtteste freie Kante finden

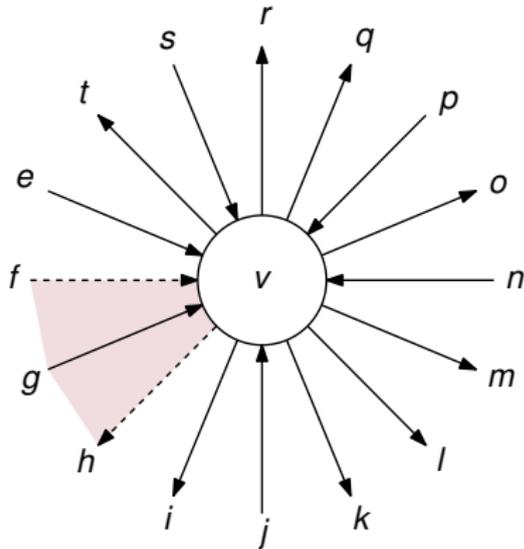
- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtteste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

Ziel: effizient rechtteste freie Kante finden

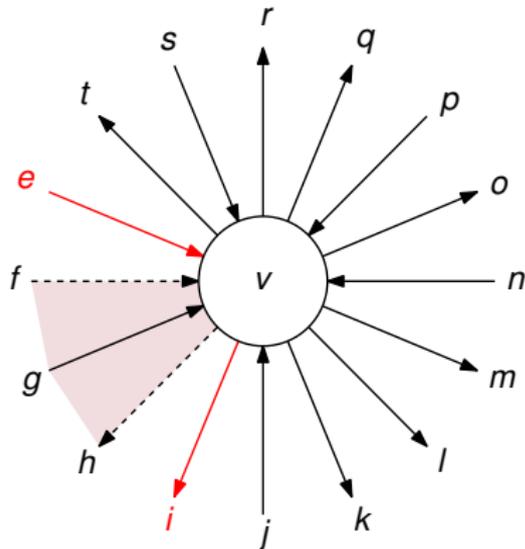
- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtteste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

Ziel: effizient rechtteste freie Kante finden

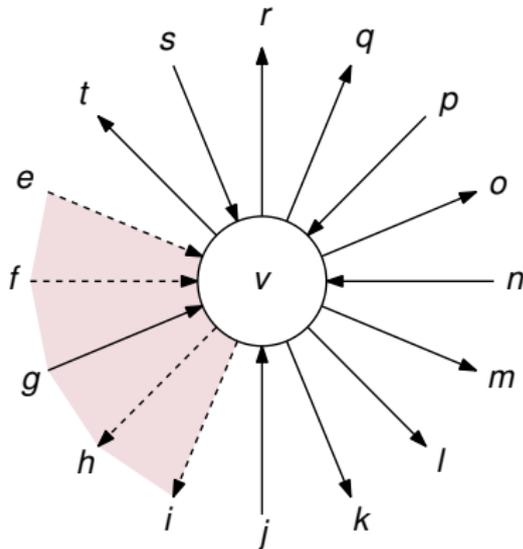
- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtteste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

Ziel: effizient rechtste freie Kante finden

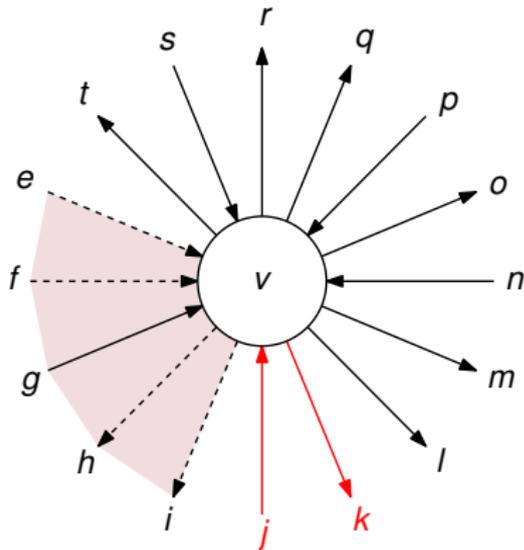
- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

Ziel: effizient rechtteste freie Kante finden

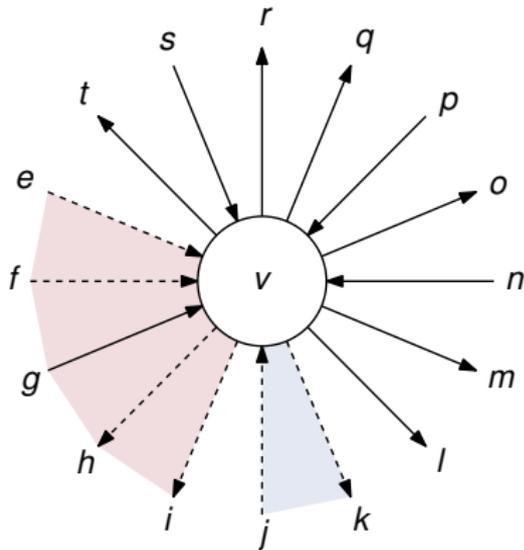
- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtteste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

Ziel: effizient rechtste freie Kante finden

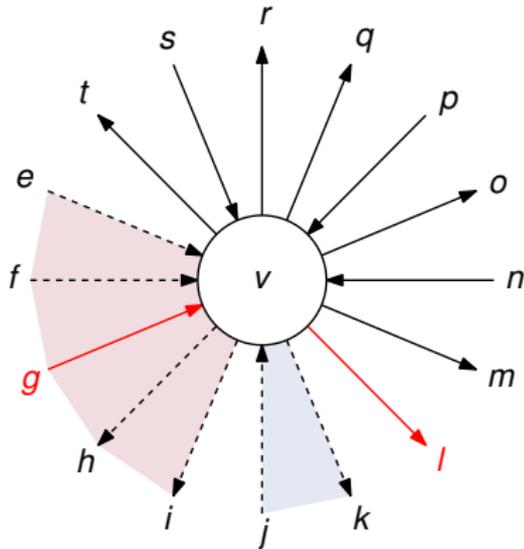
- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

Ziel: effizient rechtste freie Kante finden

- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtste Kante von Abschnitt
- UNION verbindet Abschnitt

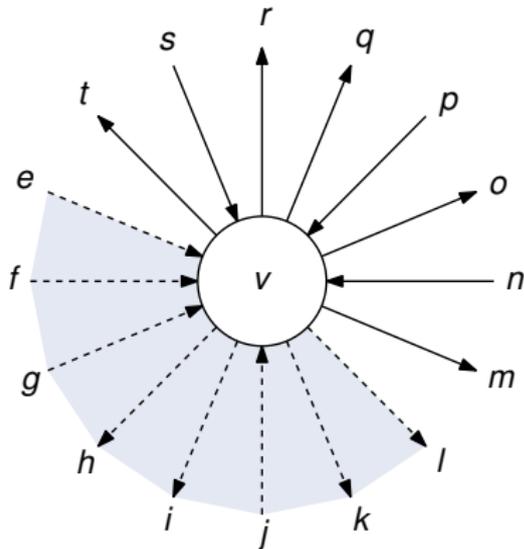


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechtteste freie Kante finden

Ziel: effizient rechtteste freie Kante finden

- Kantenabschnitte ohne freie ausgehende Kante gehören zu gleicher Menge
- am Anfang: jede Kante in eigenem Abschnitt
- FIND gibt rechtteste Kante von Abschnitt
- UNION verbindet Abschnitt



Intuition: überspringe Abschnitte ohne freie Kante effizient

1 – Kanten-Suche

Sei G ein ungerichteter, zusammenhängender, planar eingebetteter Graph, G^* der zugehörige Dualgraph und $e = (u, v)$ eine orientierte Kante.

Algorithm RIGHT-FIRST-KANTEN-DFS

Füge Kante (t, u) , $t \notin V$ im Gegenuhrzeigersinn vor e an u ein

Betrachte e als nicht orientiert

Lege (t, u) auf einen Stapel

while Stapel nicht leer **do**

 Betrachte oberste Kante (x, y)

if y inzident zu nichtorientierter Kante **then**

 Orientiere im Gegenuhrzeigersinn bzgl. y nächste nichtorientierte

 Kante $y \rightarrow w$ und lege diese auf den Stapel

else

 Entferne (x, y) vom Stapel

1 – Kanten-Suche

Sei G ein ungerichteter, zusammenhängender, planar eingebetteter Graph, G^* der zugehörige Dualgraph und $e = (u, v)$ eine orientierte Kante.

Algorithm LEFT-FIRST-KANTEN-BFS

Orientiere alle zu u inzidenten Kanten $u \rightarrow w$ und hänge diese, beginnend bei e , im Uhrzeigersinn bzgl. u an eine Warteschlange

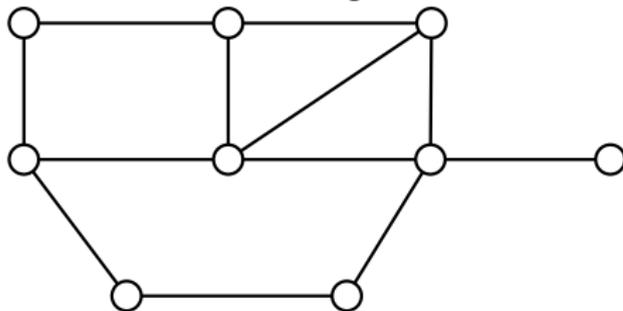
```
while Warteschlange nicht leer do
  Betrachte erste Kante  $(x, y)$ 
  if  $y$  inzident zu nichtorientierter Kante then
    Orientiere alle solche Kanten  $y \rightarrow w$  und hänge diese im Uhrzei-
    gersinn bzgl.  $y$  an die Warteschlange
  else
    Entferne  $(x, y)$  aus der Warteschlange
```

1.1 – Kanten-Suche

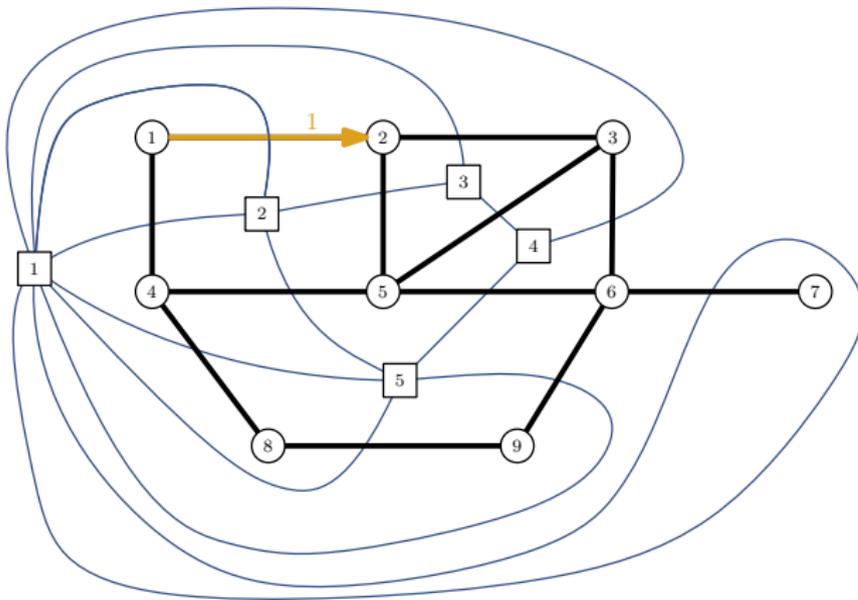
- Right-First-Tiefensuche in G ausgehend von Kante e :
 $R = (e = e_1, \dots, e_m)$
- Left-First-Breitensuche in G^* , beginnend bei Kante e^* :
 $R^* = (e^* = e_1^*, \dots, e_m^*)$.

Die Reihenfolgen R und R^* heißen dual, falls e_i Dualkante zu e_i^* für alle $1 \leq i \leq m$.

Bestimme Reihenfolge R und R^* für:



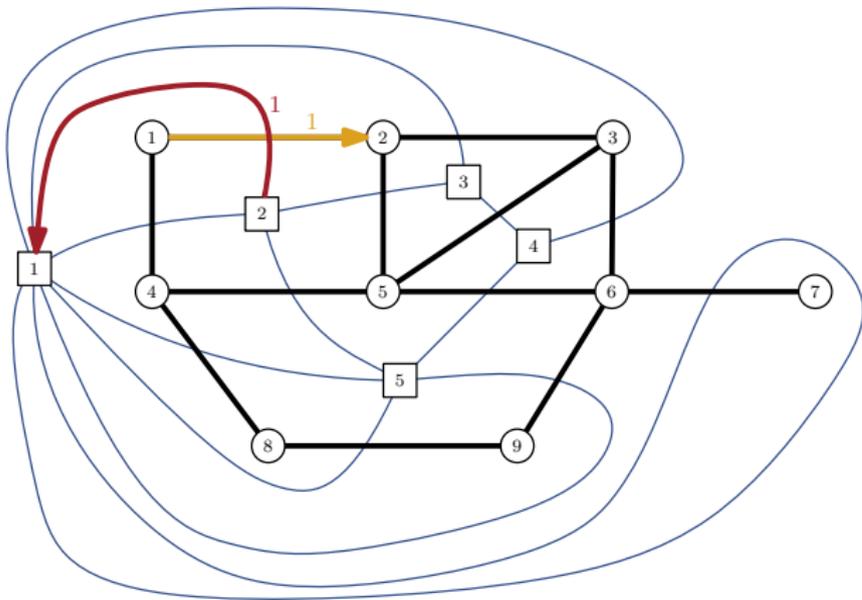
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

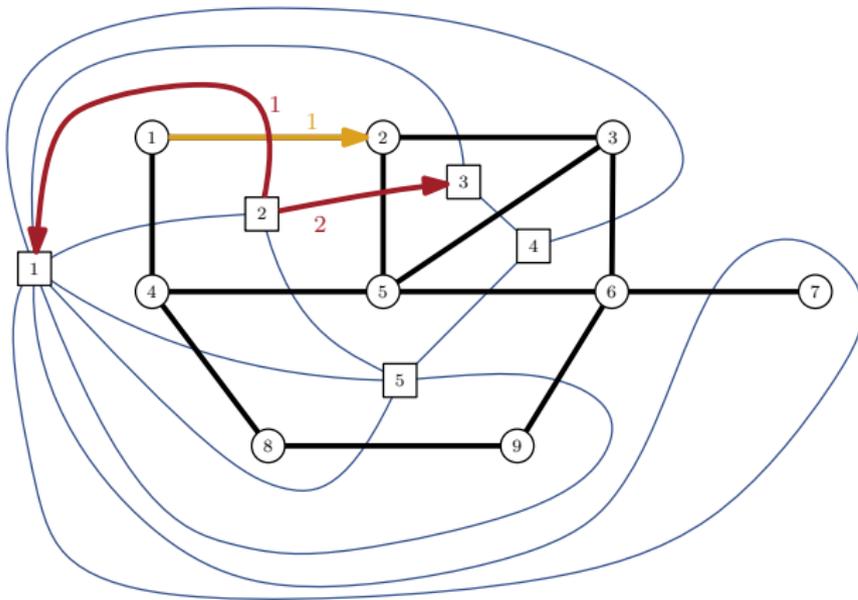
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

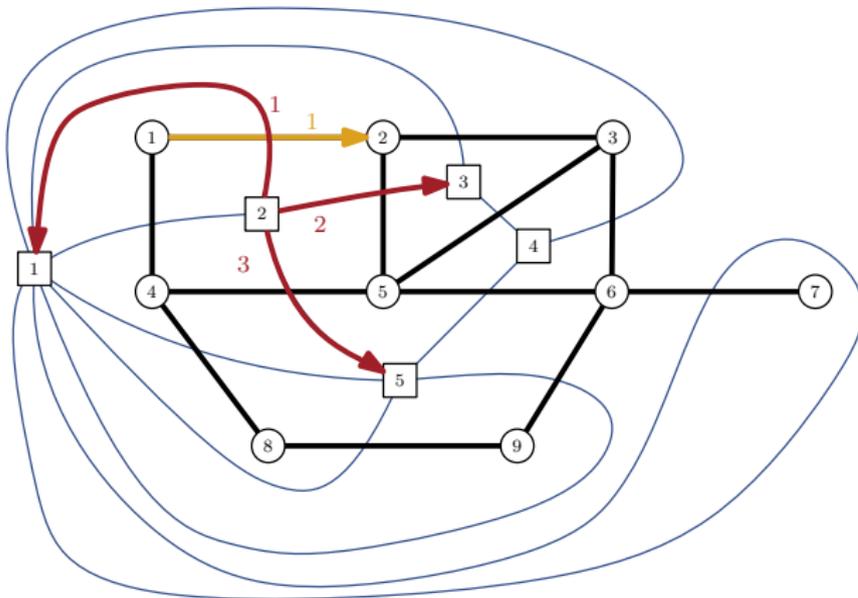
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

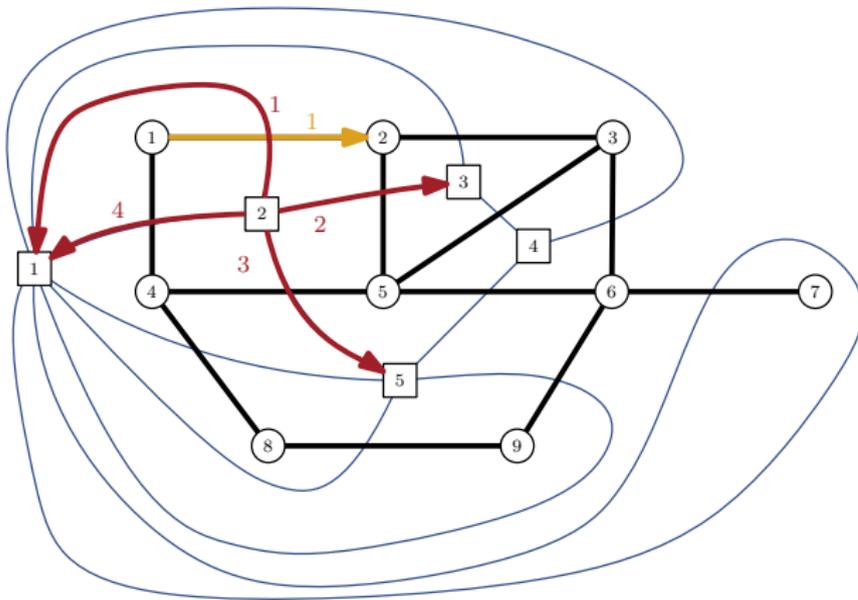
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

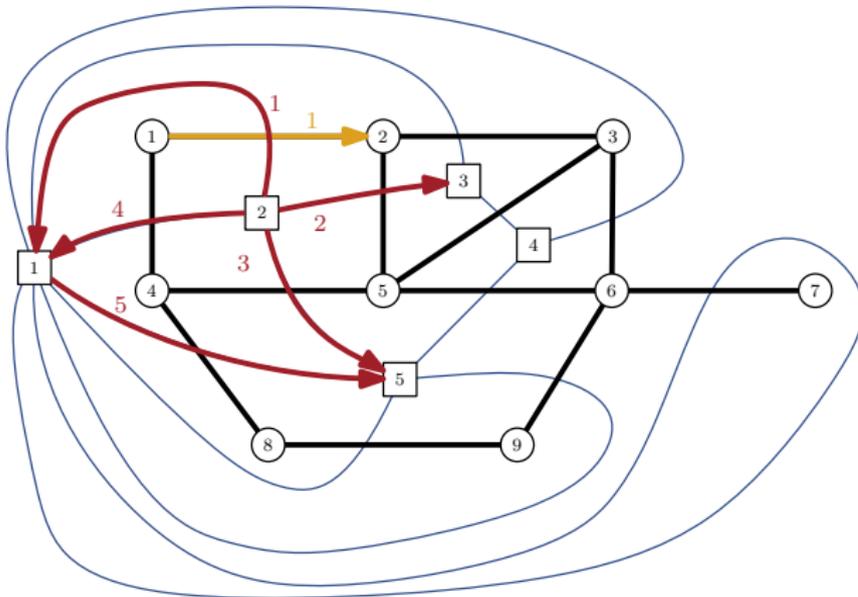
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

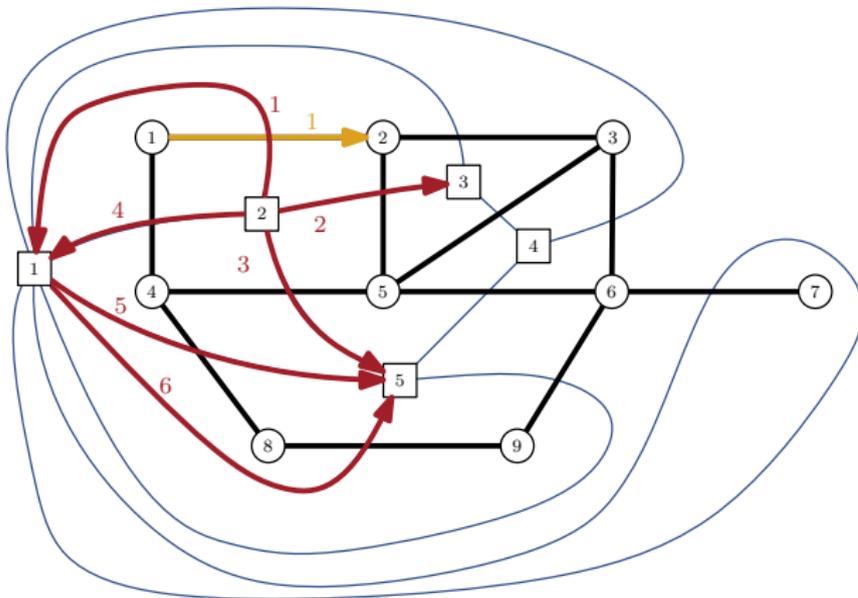
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

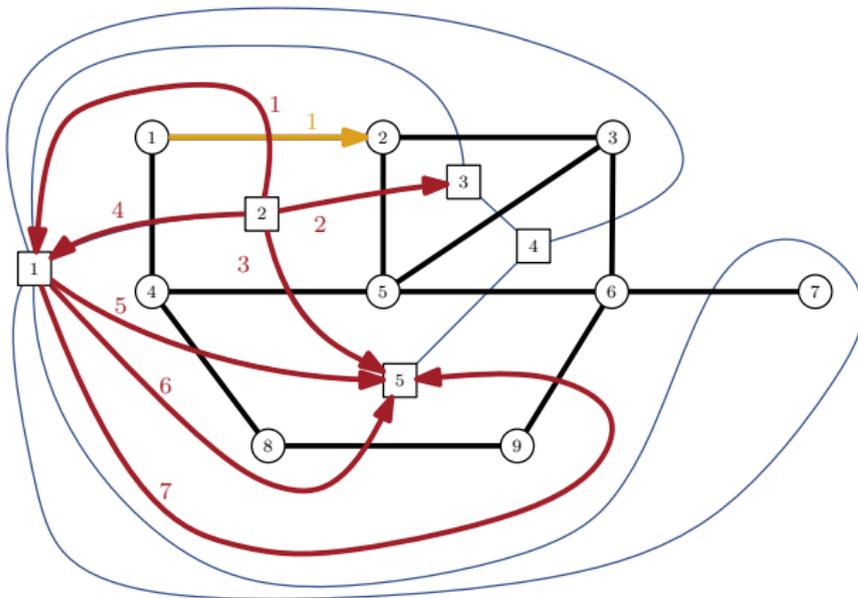
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

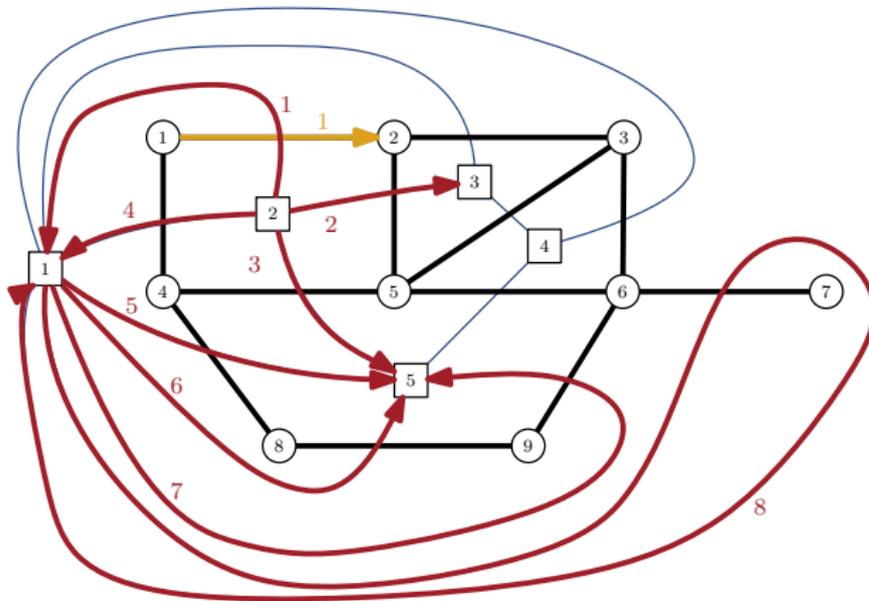
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

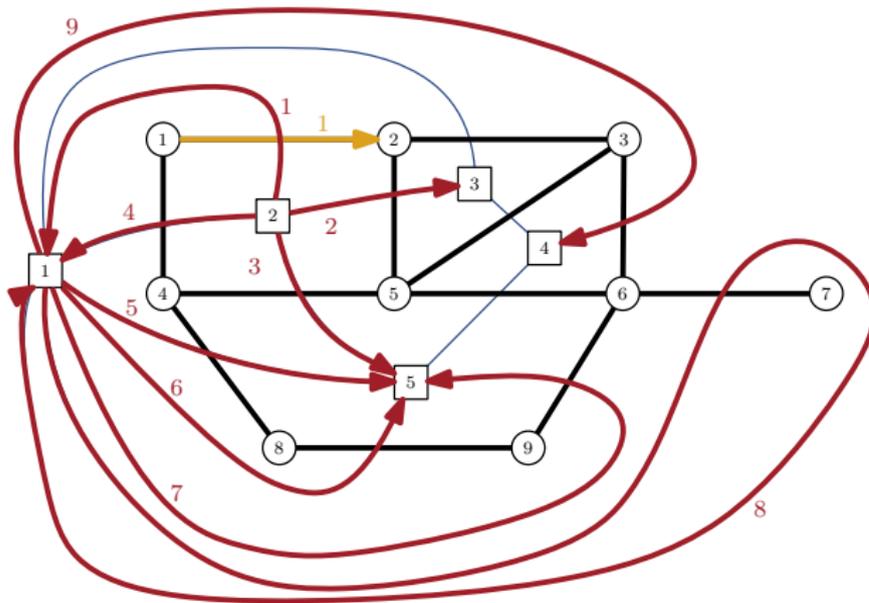
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

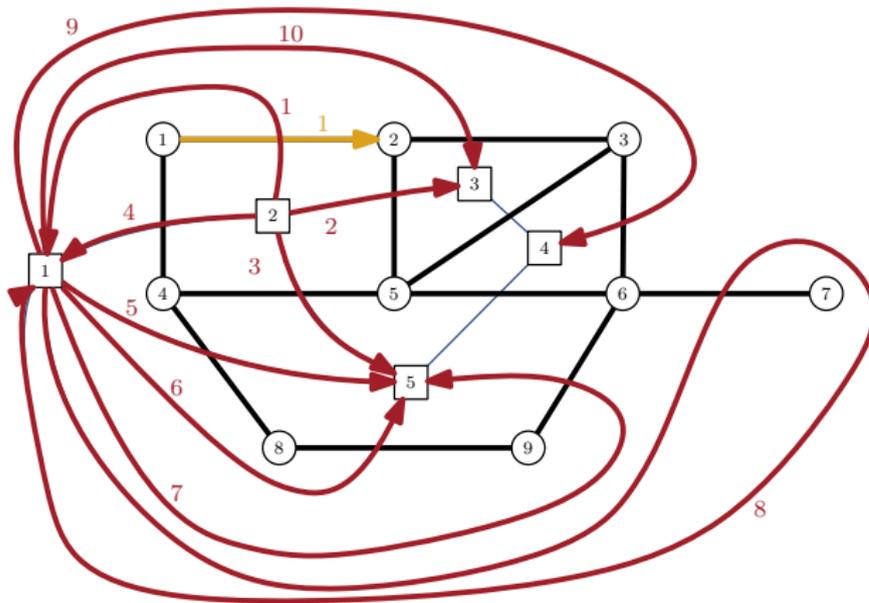
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

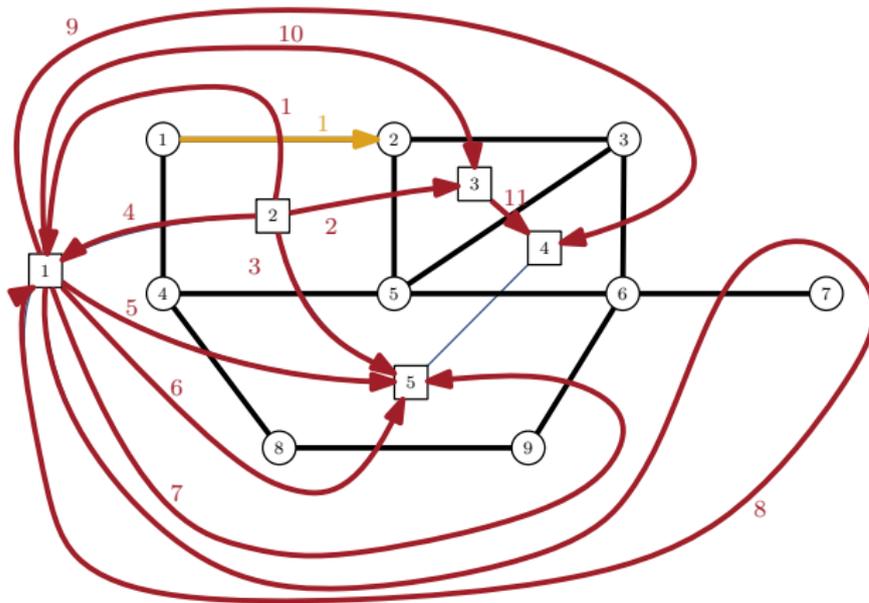
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

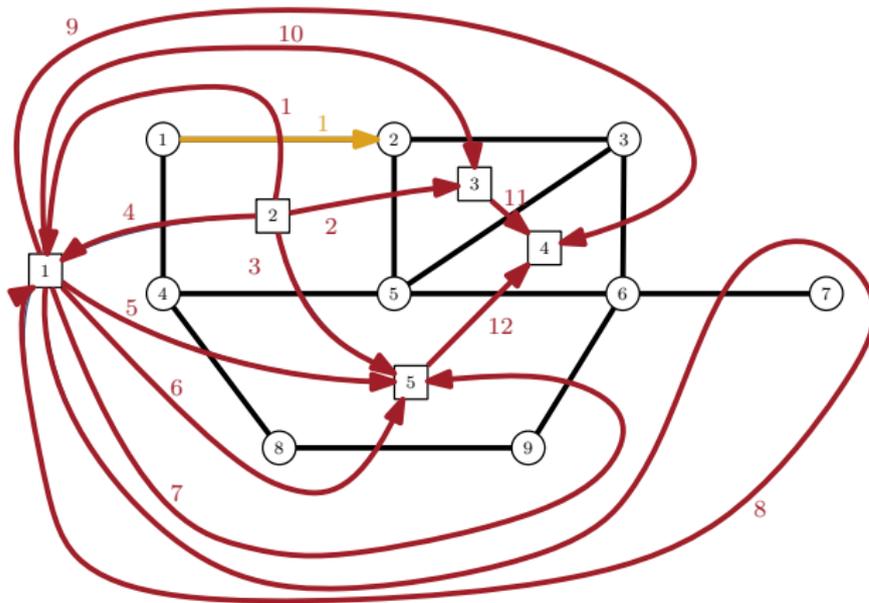
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

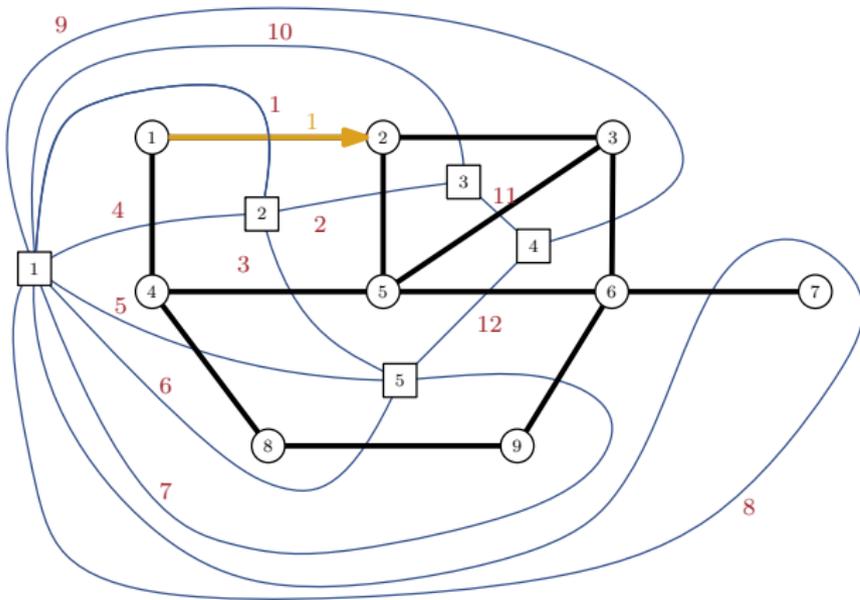
1.1 – Kanten-Suche



Left-First-Kanten-BFS

- Auf G^*
- Queue
- Im Uhrzeigersinn hinzufügen

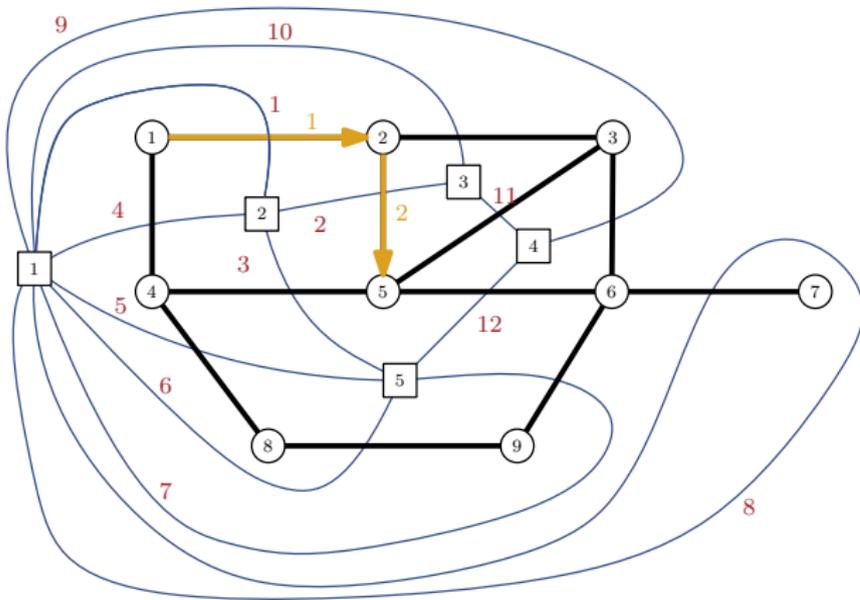
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

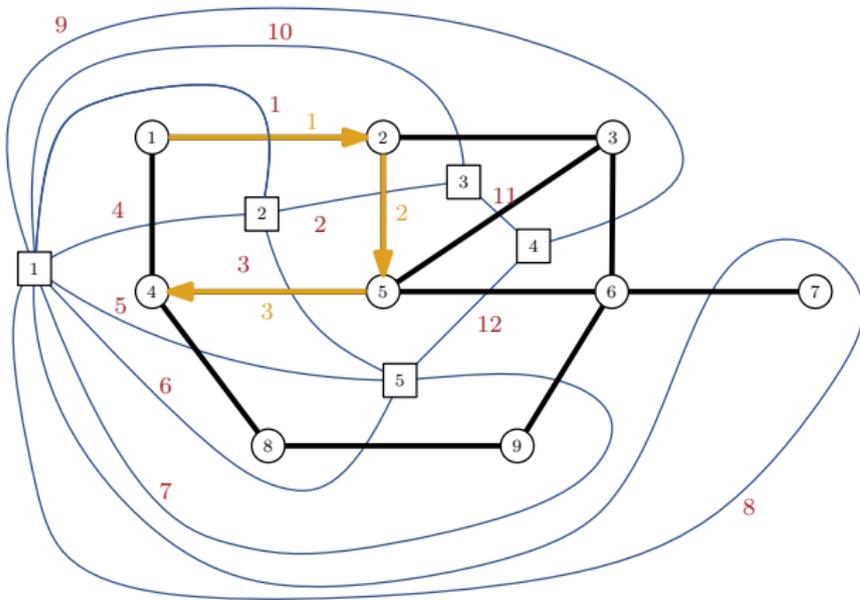
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

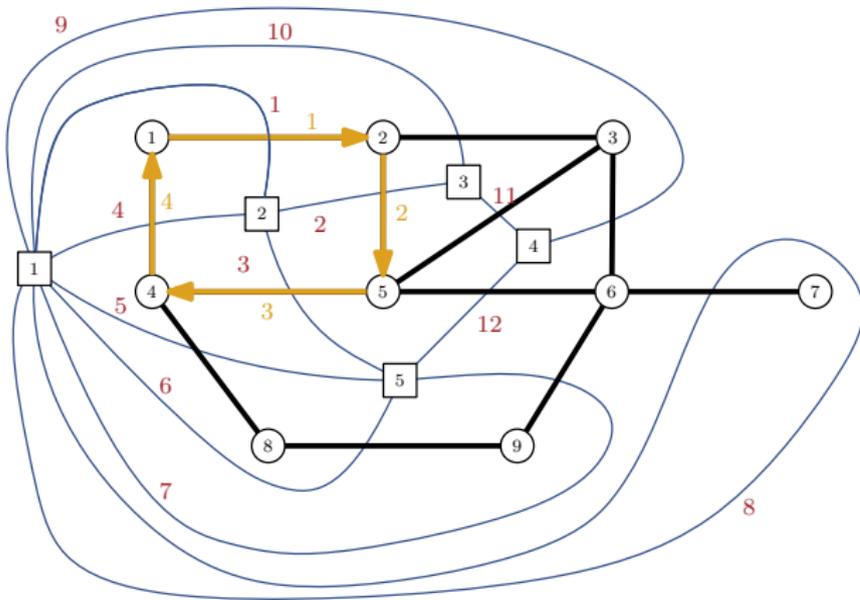
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

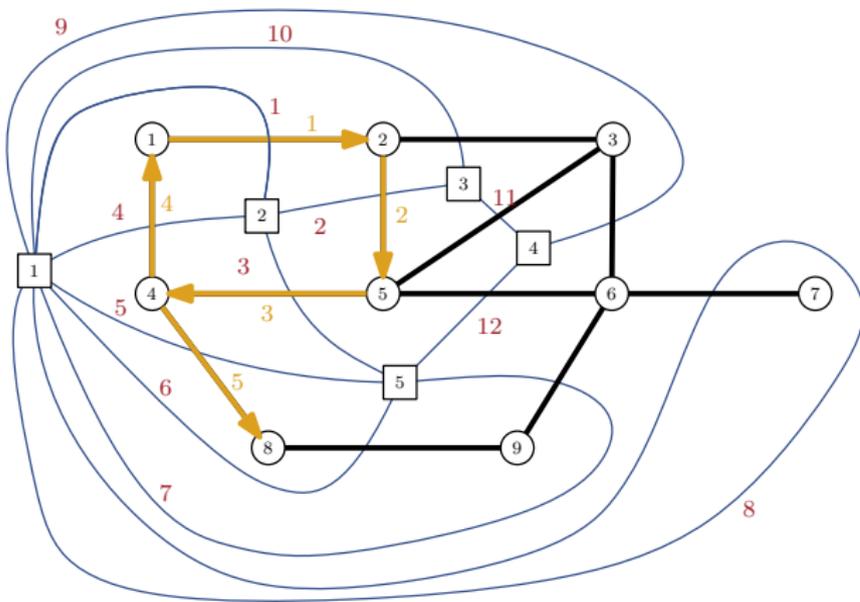
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

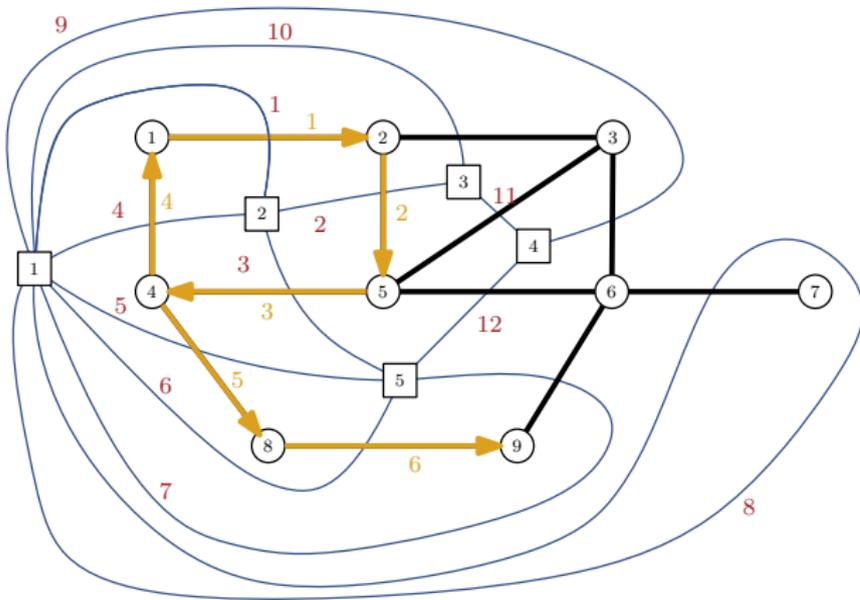
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

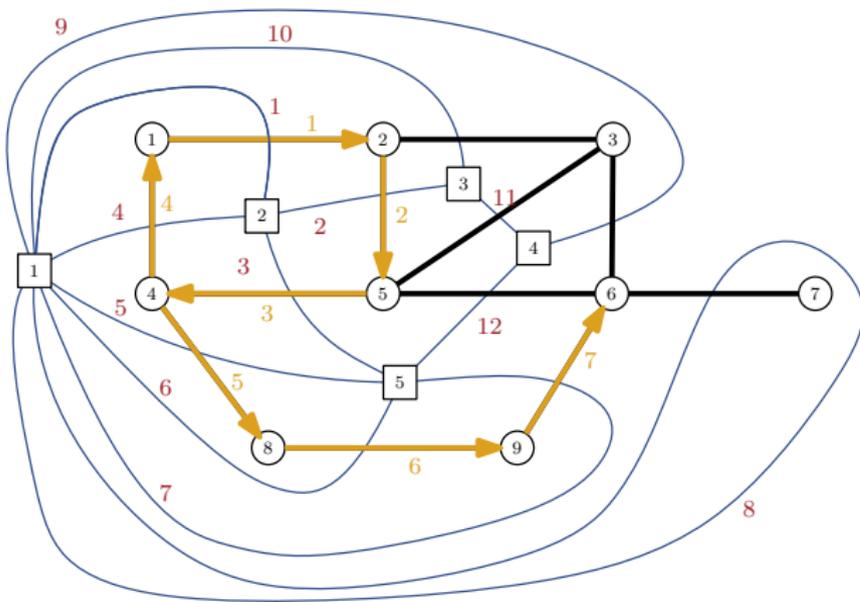
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

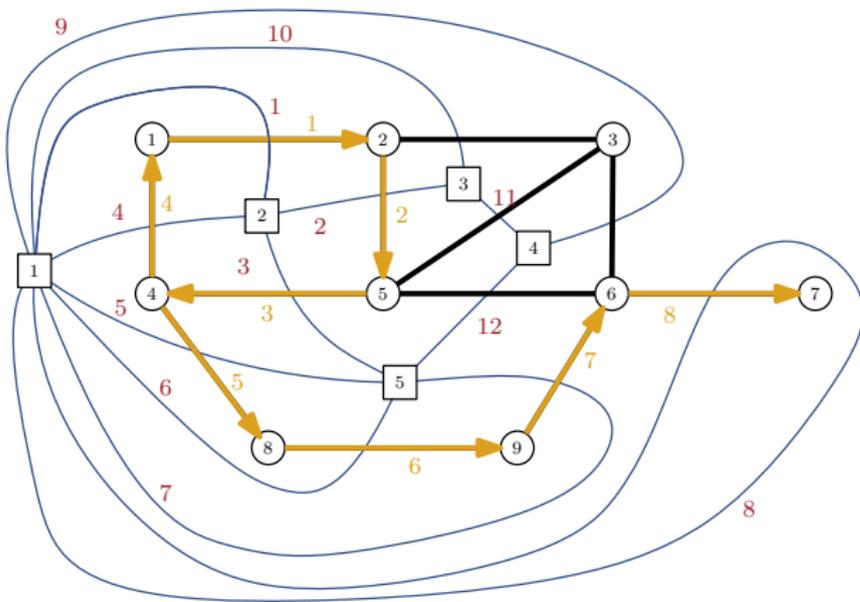
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

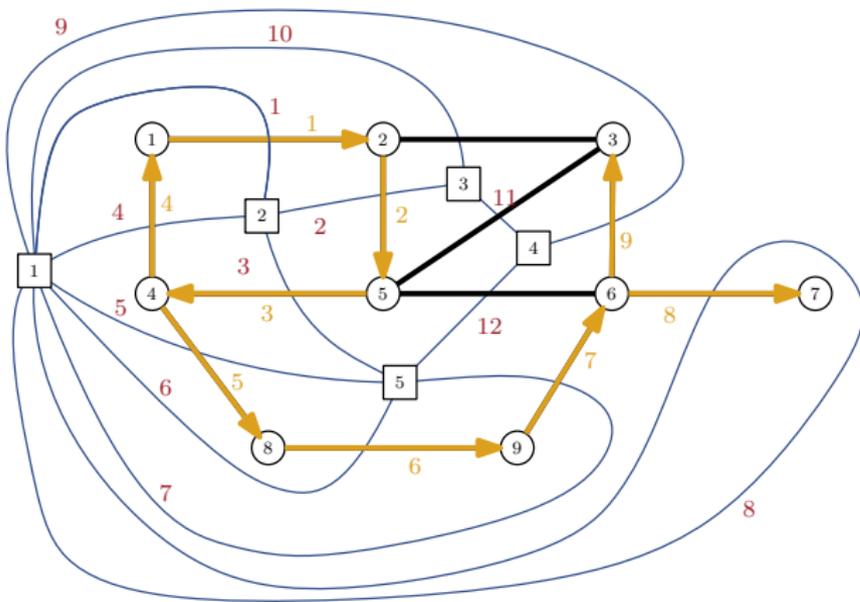
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

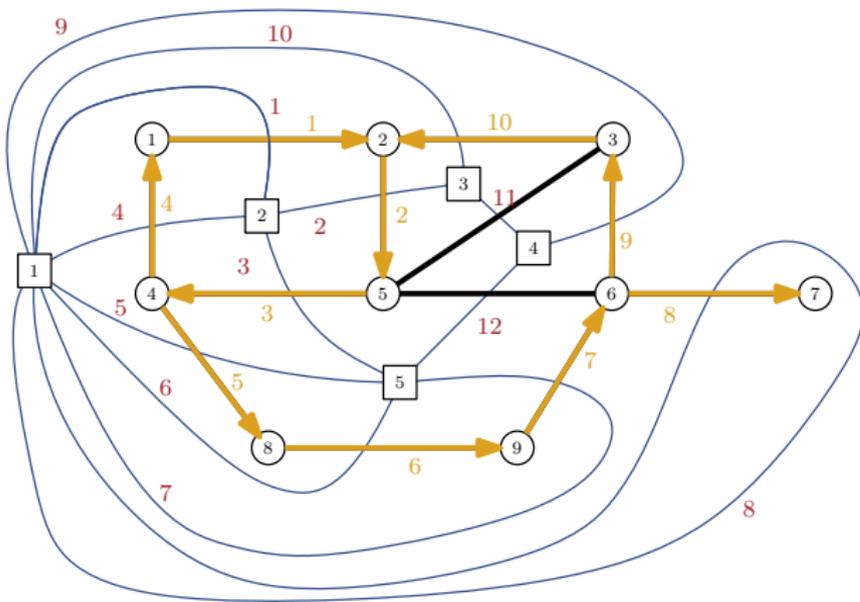
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

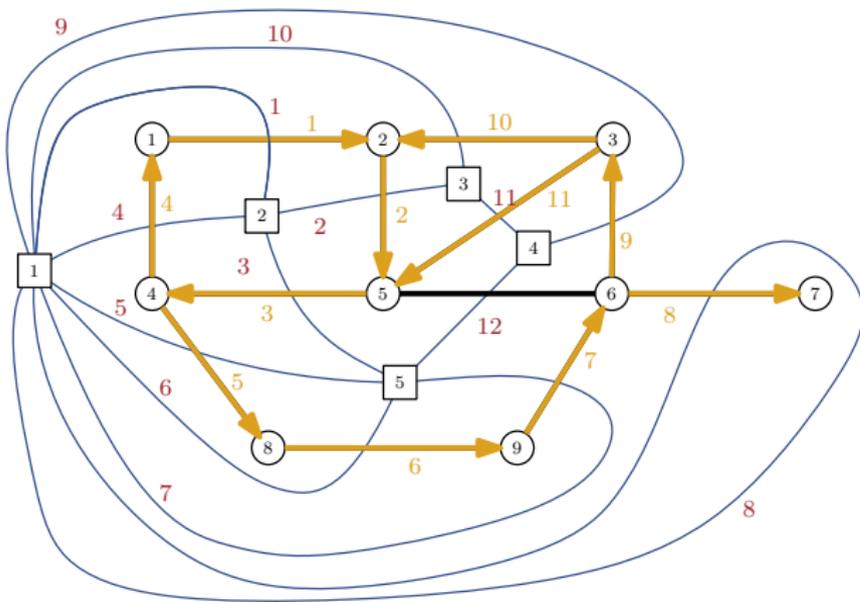
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

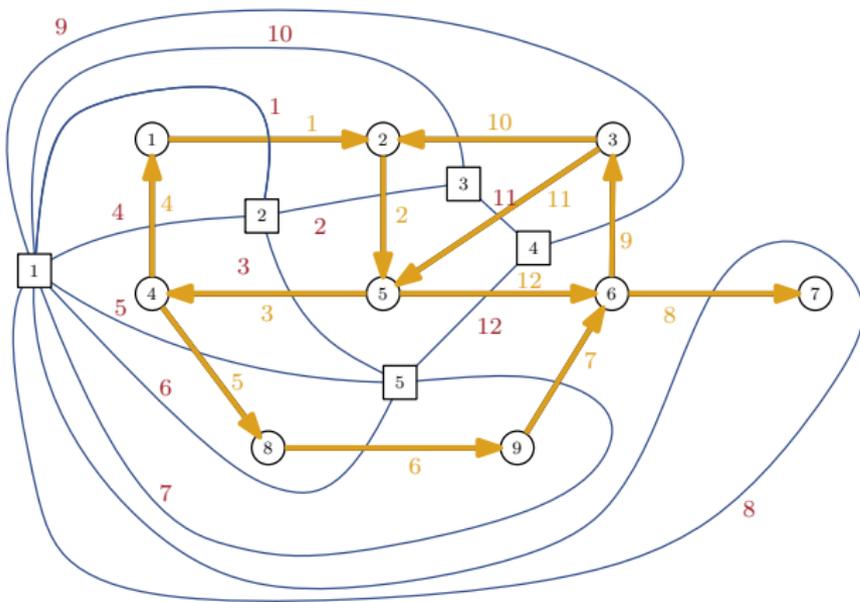
1.1 – Kanten-Suche



Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

1.1 – Kanten-Suche

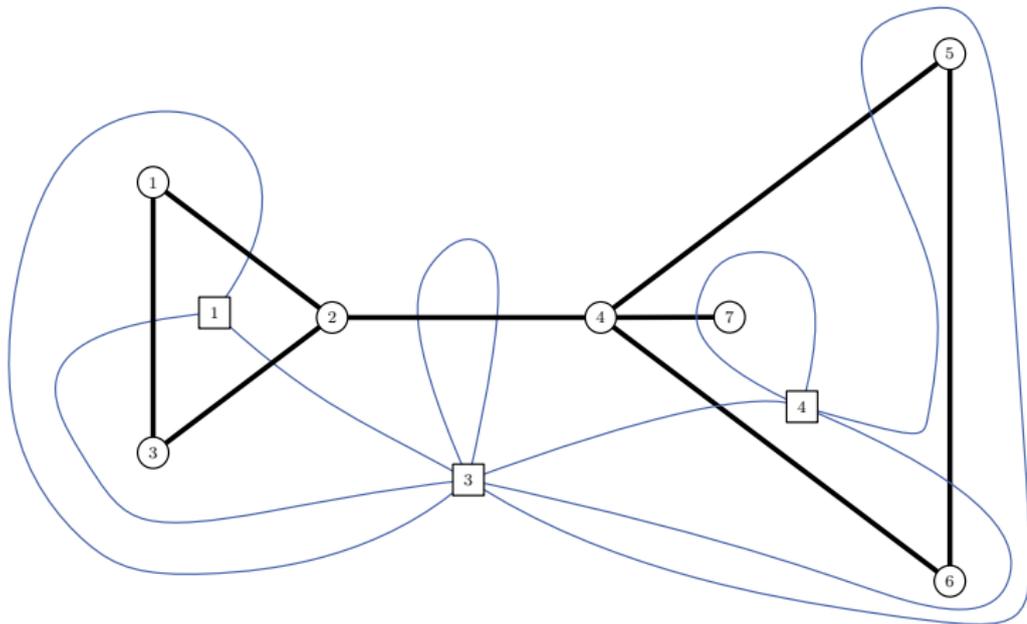


Right-First-Kanten-DFS

- Auf G
- Stack
- Gegen den Uhrzeigersinn hinzufügen

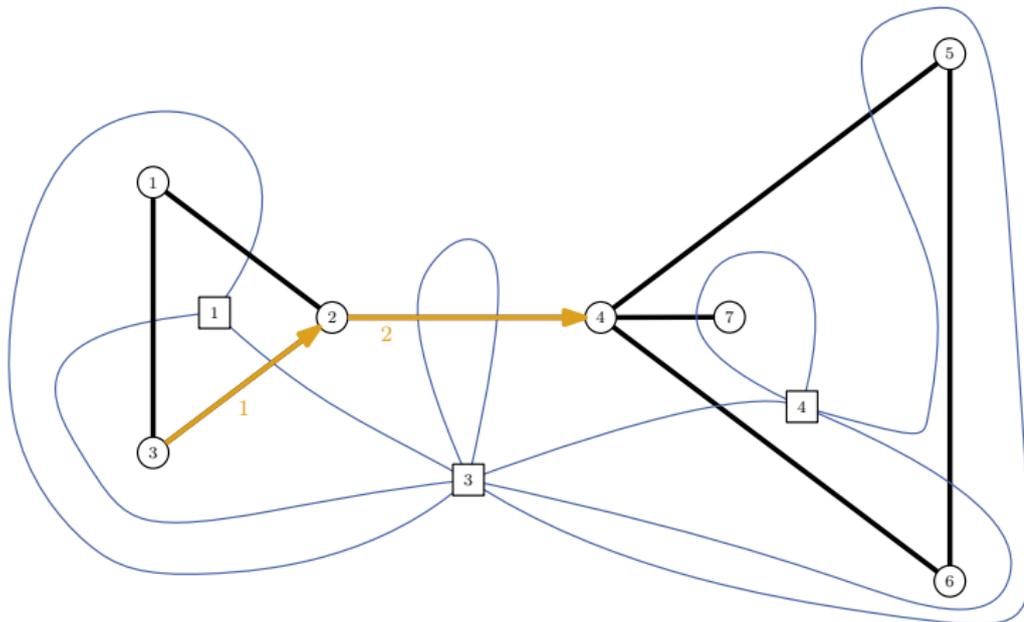
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



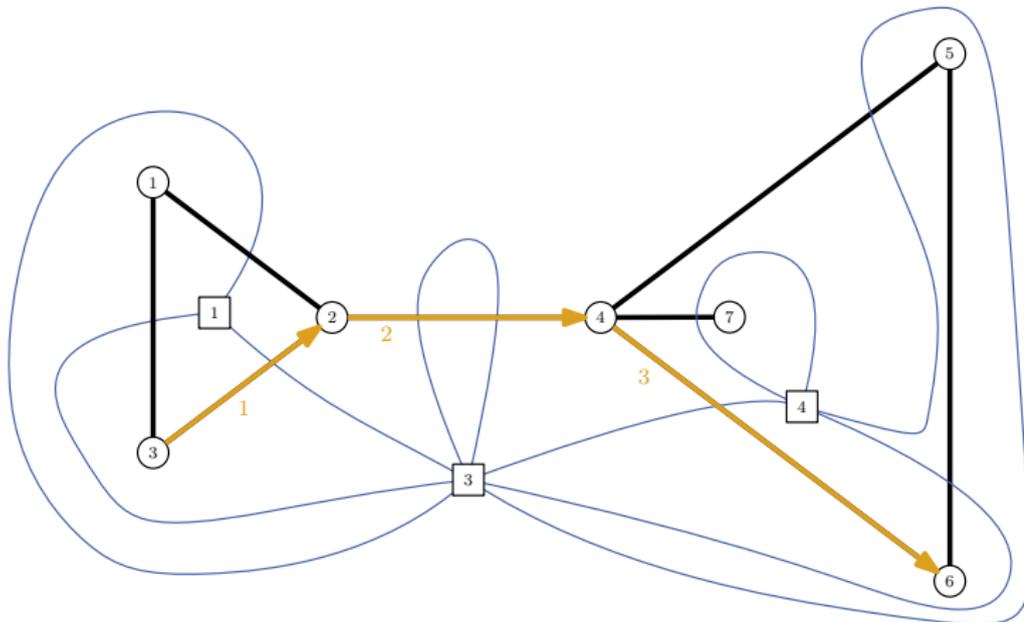
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



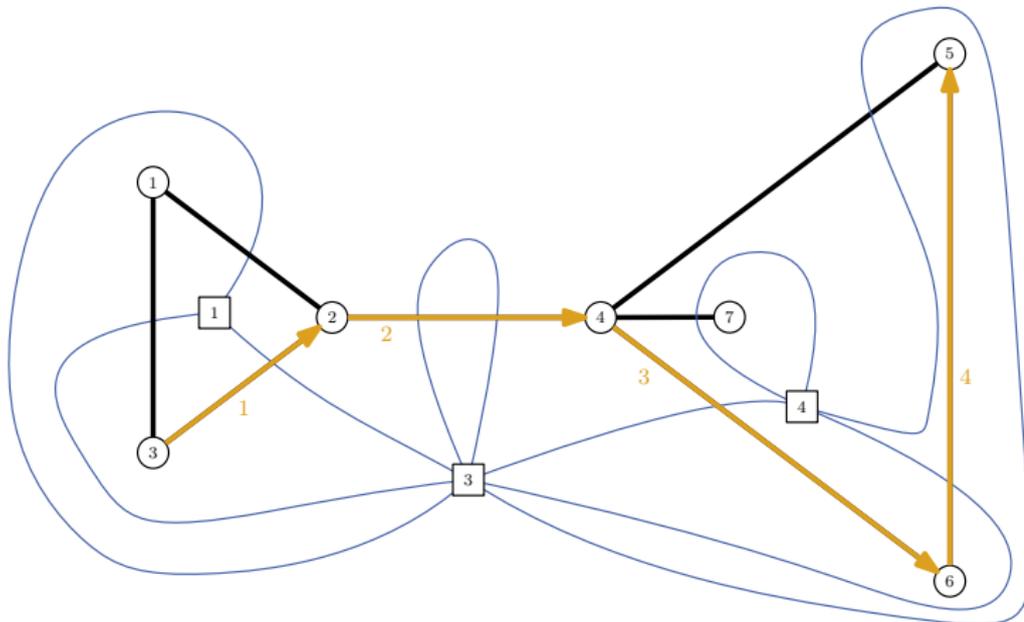
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



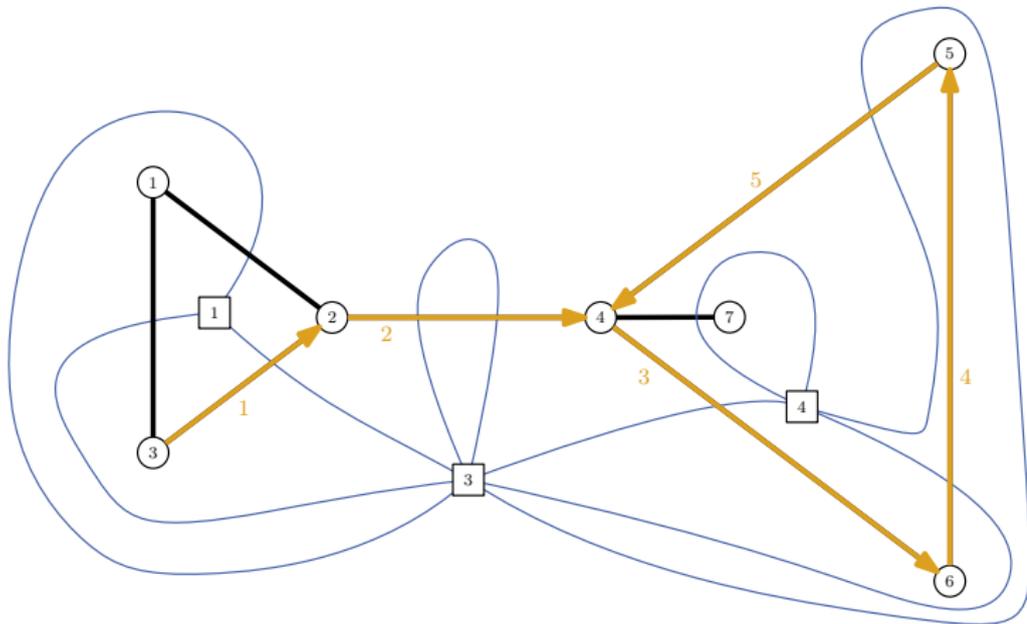
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



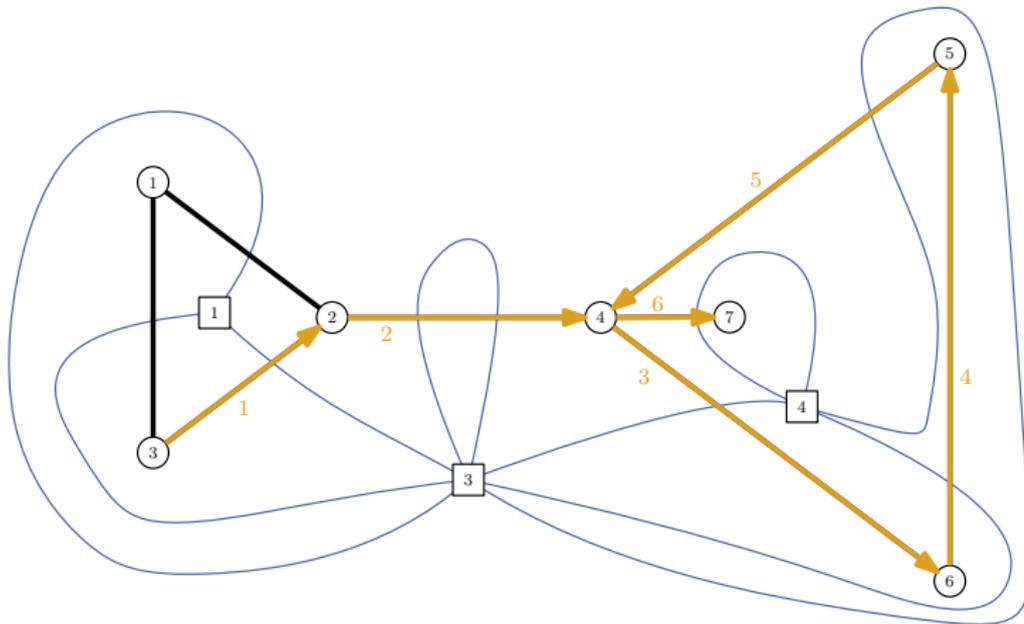
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



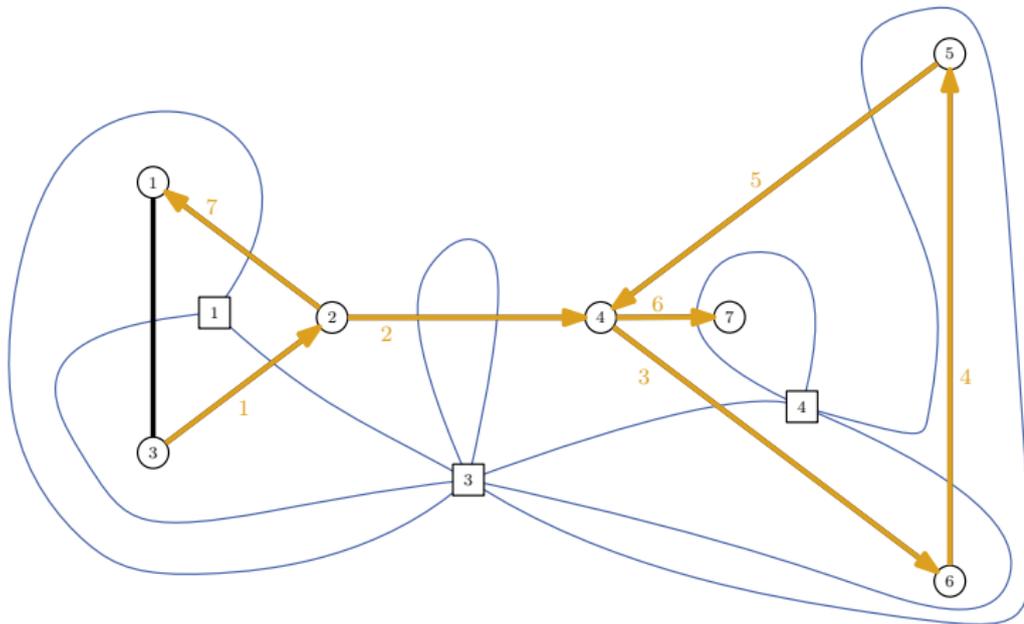
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



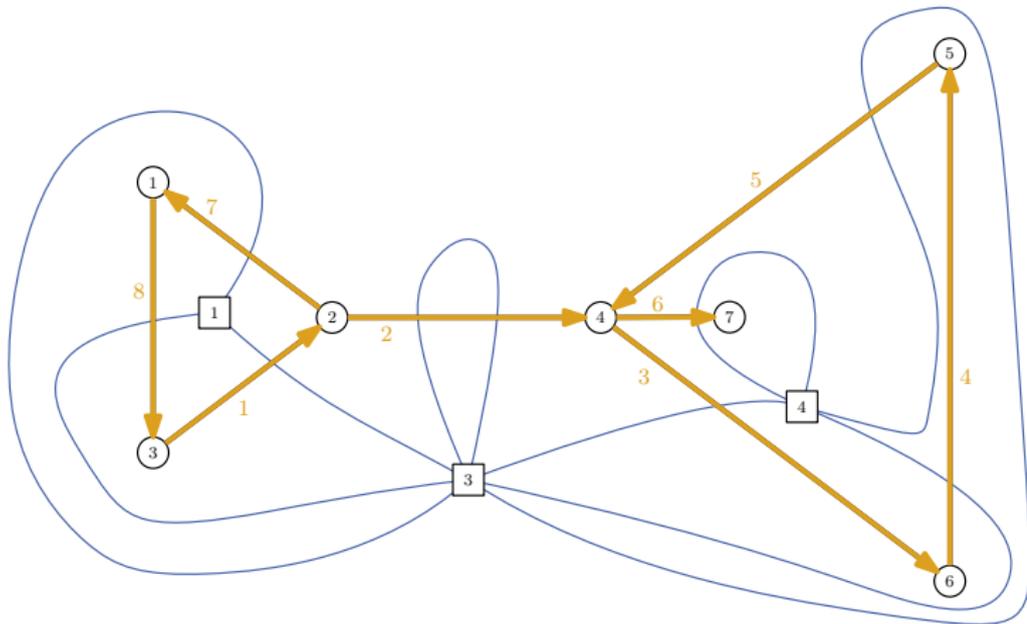
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



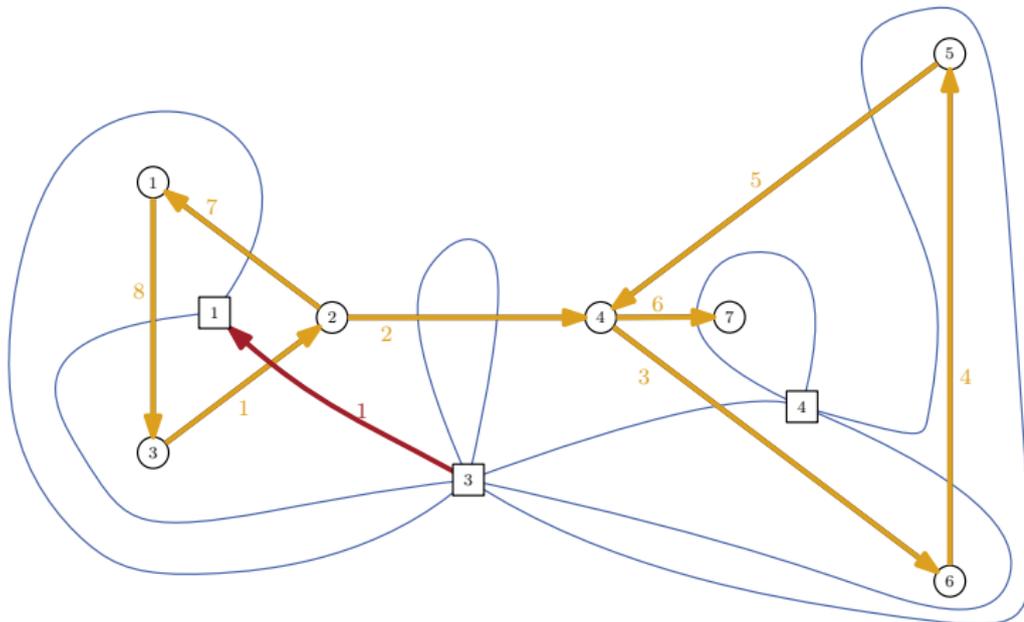
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



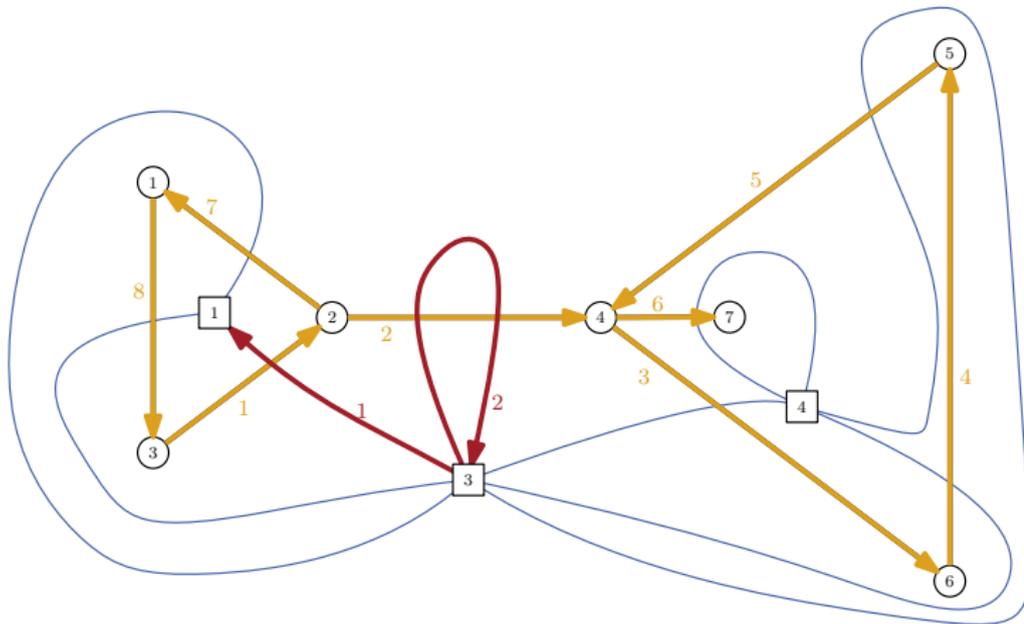
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



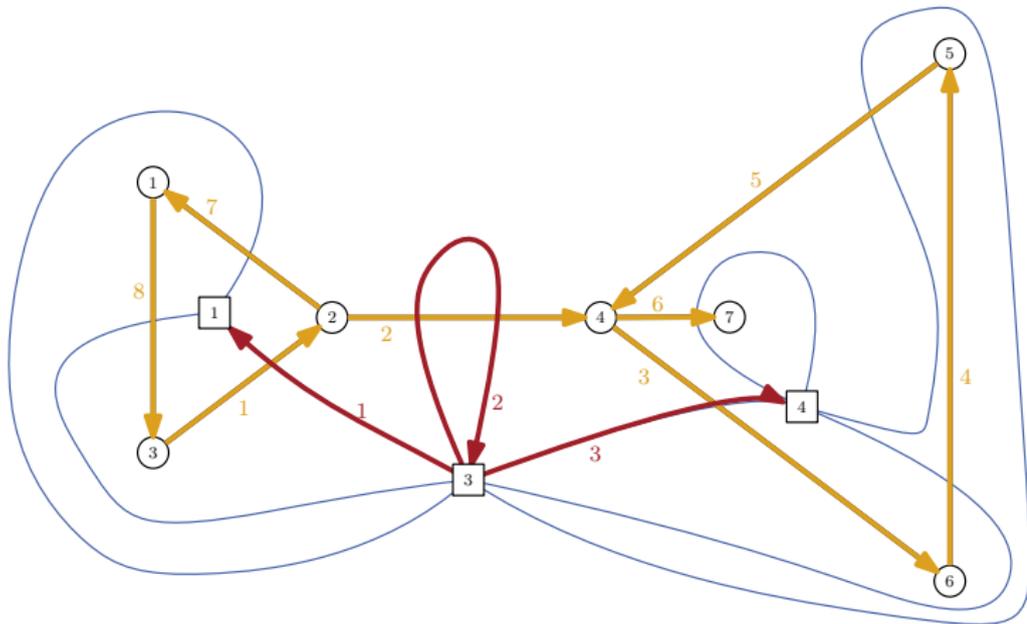
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



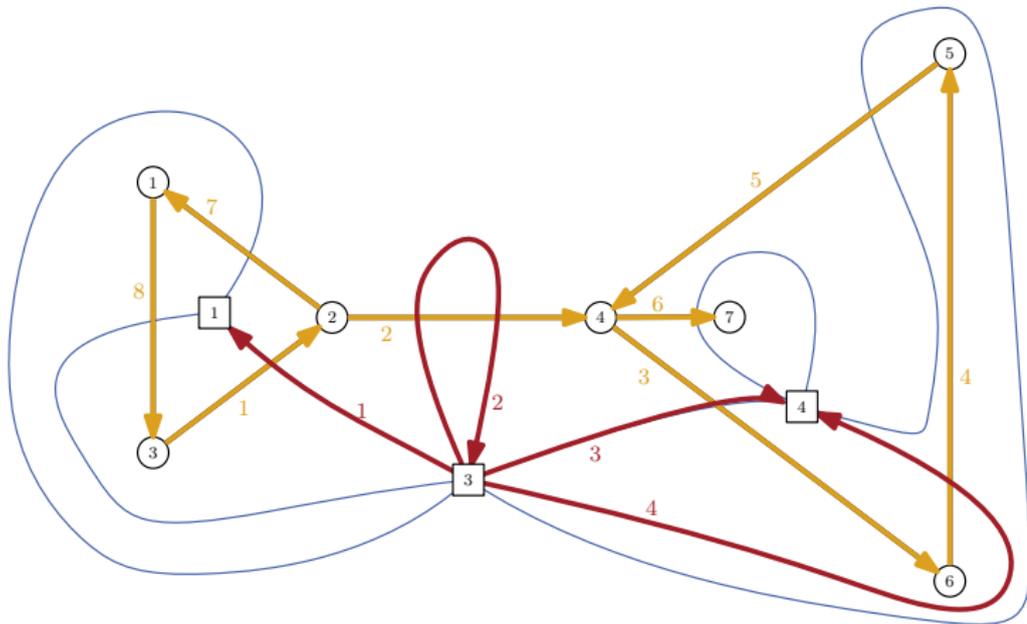
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



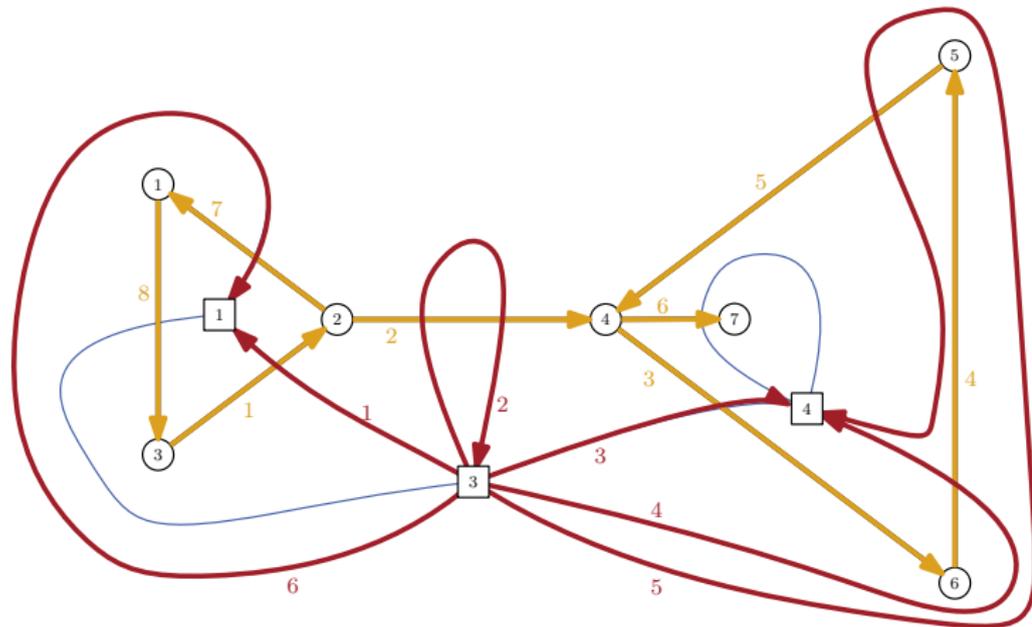
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



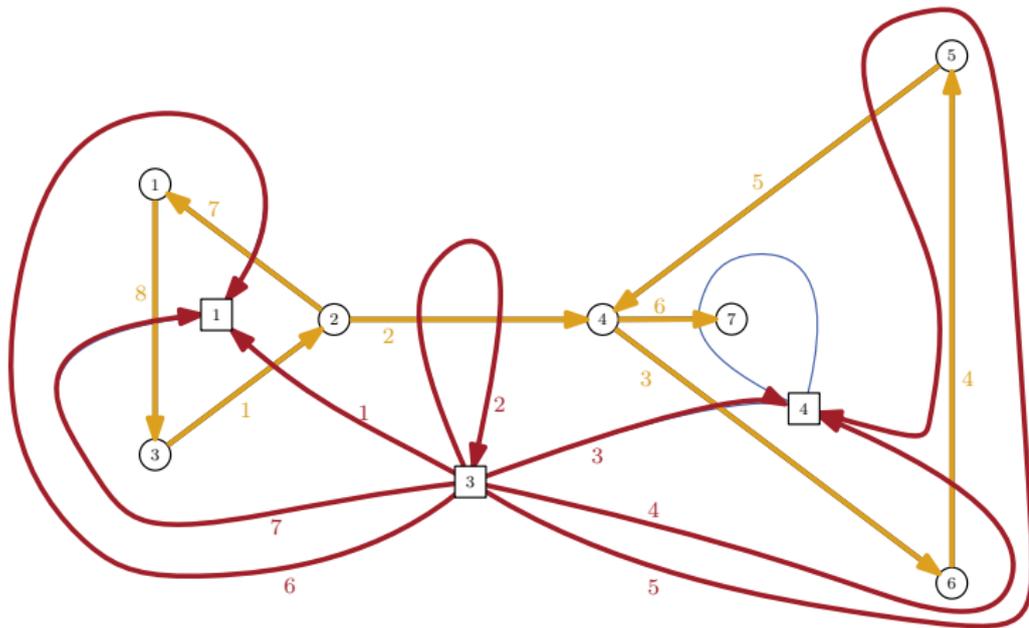
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



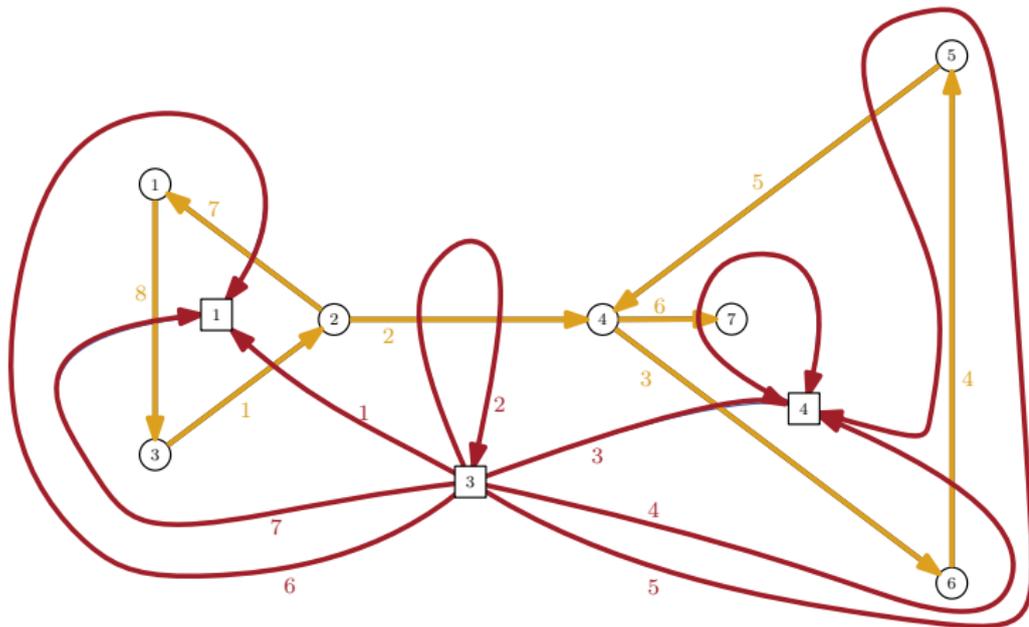
1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.

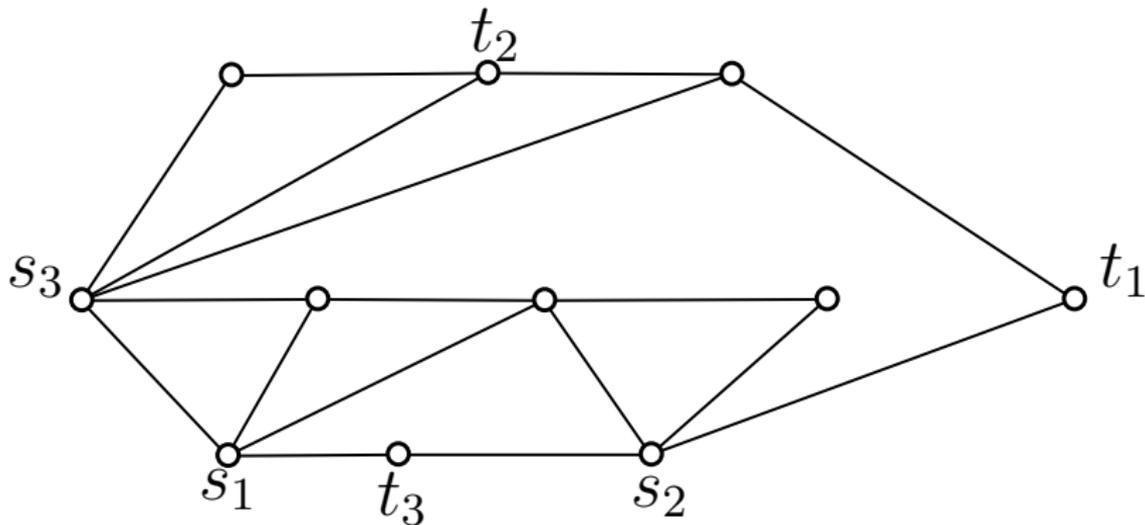


1.2 – Kanten-Suche

Geben Sie einen Graphen an, für den R und R^* nicht dual sind.



2 – Kantendisjunkte Wegpackung



- Ist die *Geradheitsbedingung* erfüllt?
- Ist das *kantendisjunkte Wegpackungsproblem* lösbar?

2 – Kantendisjunkte Wegpackung

Gegeben: Graph $G = (V, E)$ und Paare ausgezeichneter Knoten

$$D = \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$$

Finde: Paarweise kantendisjunkte s_i - t_i -Wege in G , $1 \leq i \leq k$.

Definition

Sei $G = (V, E)$, $X \subseteq V$, $D = \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$

Kapazität $c(X) := |\{\{u, v\} \in E : u \in X, v \in V \setminus X\}|$

Dichte $d(X) := |\{\{s_i, t_i\} \in D : |\{s_i, t_i\} \cap X| = 1\}|$

Freie Kapazität $fc(X) := c(X) - d(X)$

2 – Kantendisjunkte Wegpackung

Gegeben: Graph $G = (V, E)$ und Paare ausgezeichneter Knoten

$$D = \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$$

Finde: Paarweise kantendisjunkte s_i - t_i -Wege in G , $1 \leq i \leq k$.

Definition

Sei $G = (V, E)$, $X \subseteq V$, $D = \{\{s_i, t_i\} : s_i, t_i \in V, 1 \leq i \leq k\}$

Kapazität $c(X) := |\{\{u, v\} \in E : u \in X, v \in V \setminus X\}|$

Dichte $d(X) := |\{\{s_i, t_i\} \in D : |\{s_i, t_i\} \cap X| = 1\}|$

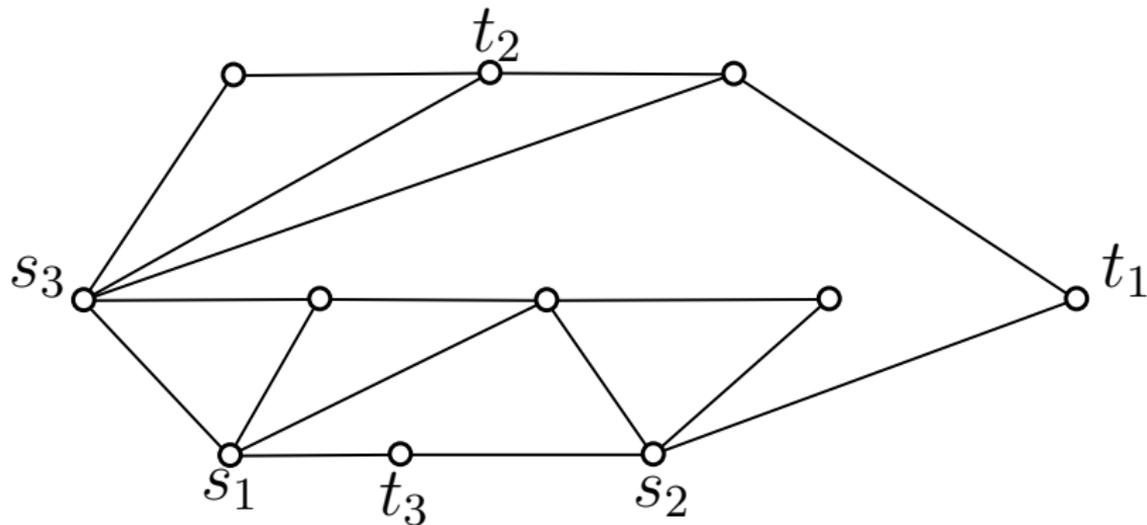
Freie Kapazität $fc(X) := c(X) - d(X)$

Geradheitsbedingung erfüllt

$\Leftrightarrow fc(X)$ gerade für alle $X \subseteq V$

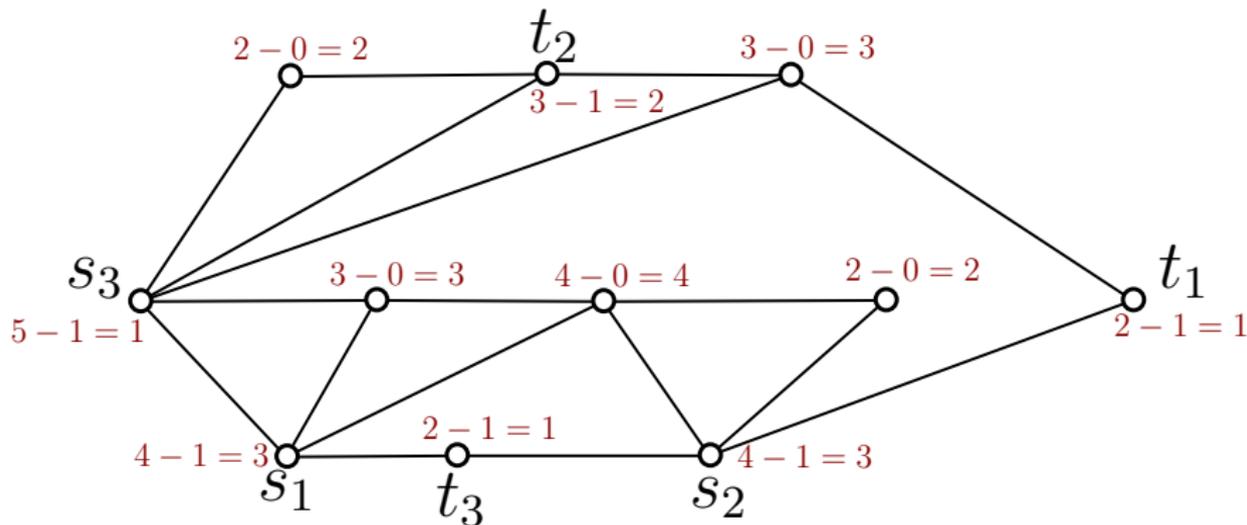
$\Leftrightarrow fc(\{v\})$ gerade für alle $v \in V$

2 – Kantendisjunkte Wegpackung



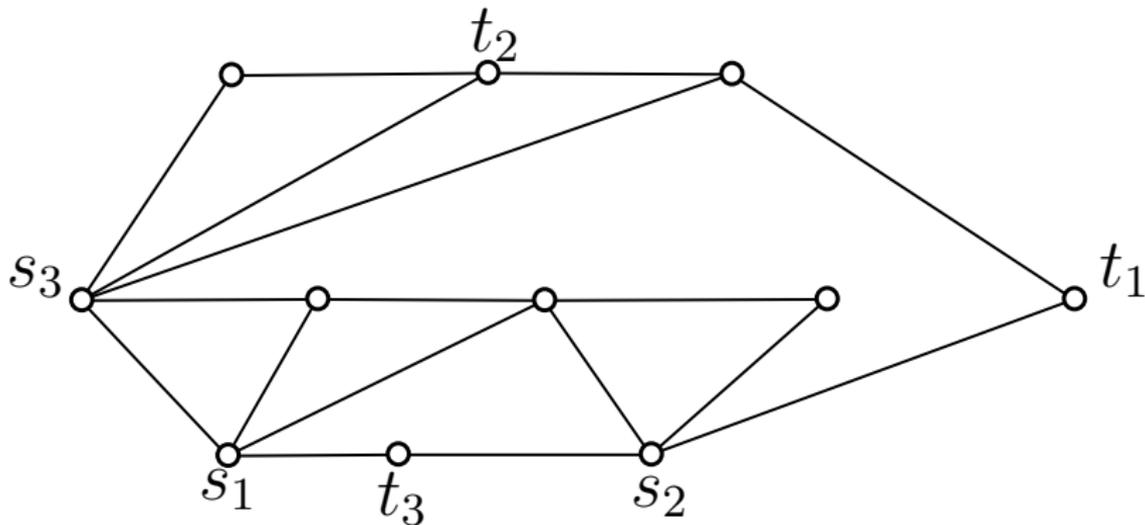
- Ist die *Geradheitsbedingung* erfüllt?
- Ist das *kantendisjunkte Wegpackungsproblem* lösbar?

2 – Kantendisjunkte Wegpackung



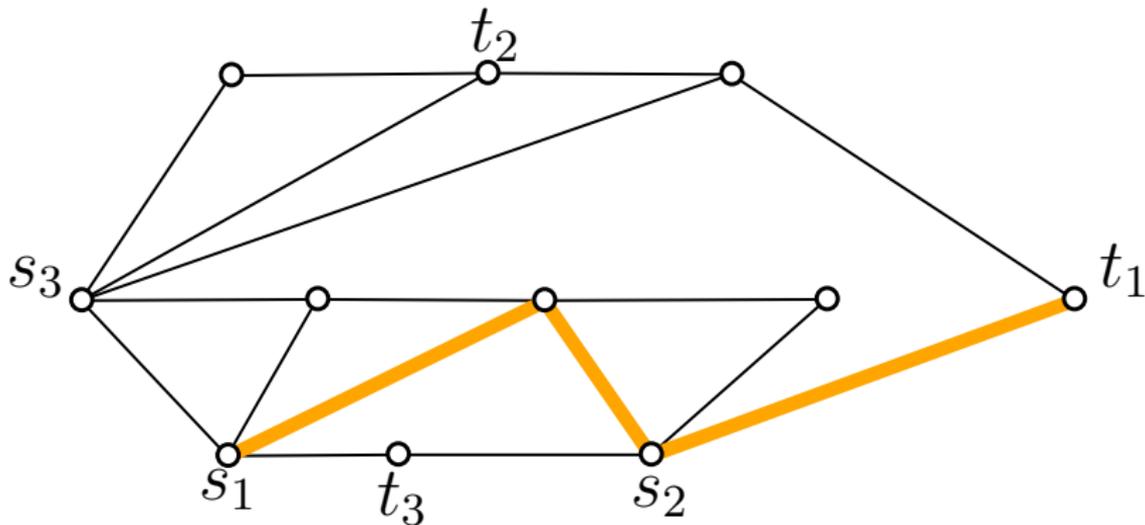
- Ist die *Geradheitsbedingung* erfüllt? Nein
- Ist das *kantendisjunkte Wegpackungsproblem* lösbar?

2 – Kantendisjunkte Wegpackung



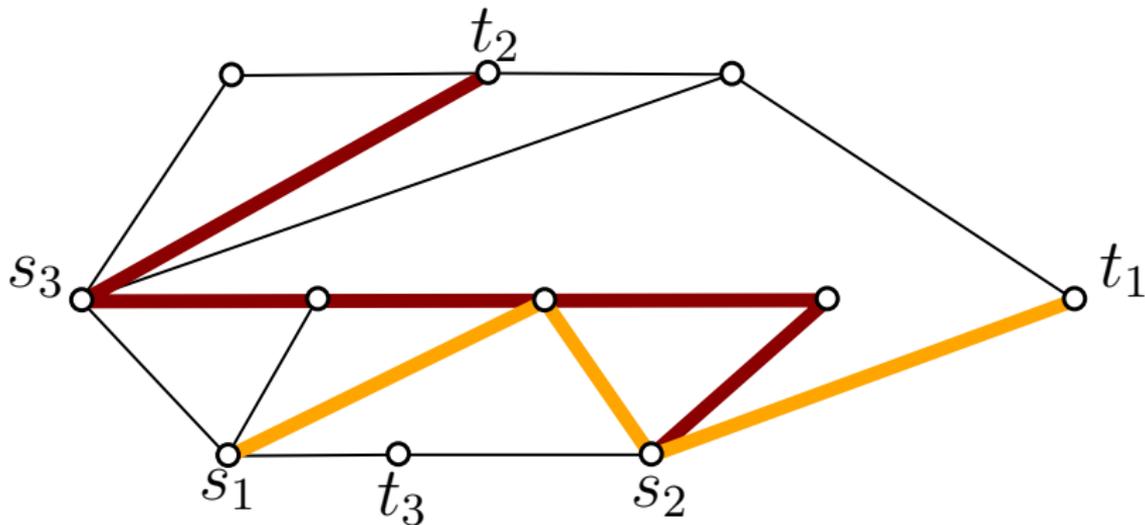
- Ist die *Geradheitsbedingung* erfüllt? Nein
- Ist das *kantendisjunkte Wegpackungsproblem* lösbar?

2 – Kantendisjunkte Wegpackung



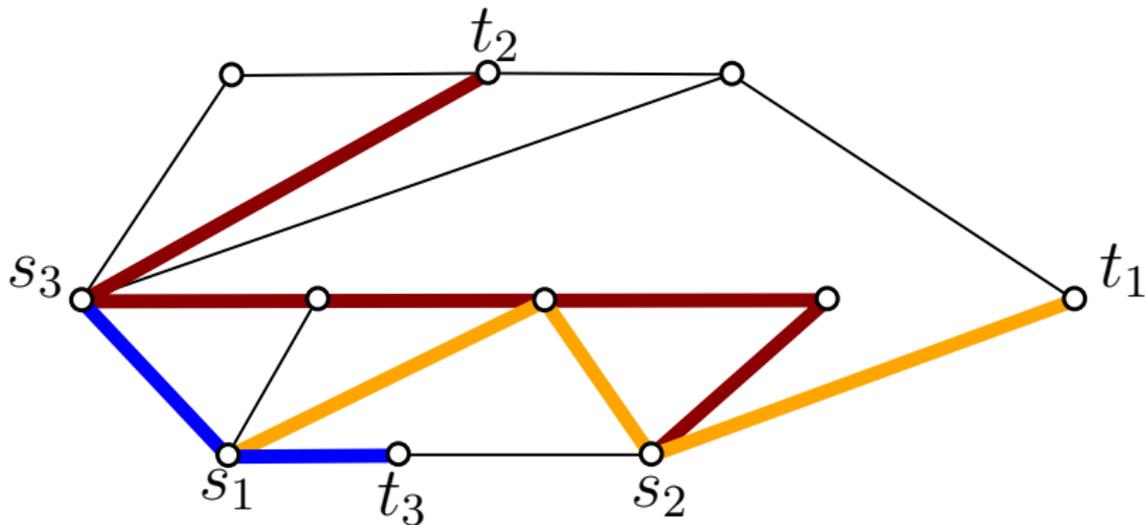
- Ist die *Geradheitsbedingung* erfüllt? Nein
- Ist das *kantendisjunkte Wegpackungsproblem* lösbar?

2 – Kantendisjunkte Wegpackung



- Ist die *Geradheitsbedingung* erfüllt? Nein
- Ist das *kantendisjunkte Wegpackungsproblem* lösbar?

2 – Kantendisjunkte Wegpackung



- Ist die *Geradheitsbedingung* erfüllt? Nein
- Ist das *kantendisjunkte Wegpackungsproblem* lösbar? Ja

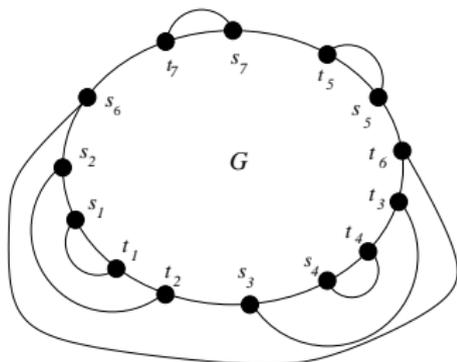
Definition (knotendisjunktes Wegpackungsproblem): Gegeben ein zusammenhängender, planarer Graph G mit fester Einbettung und paarweise verschiedenen Terminalen $s_1, t_1, s_2, t_2, \dots, s_k, t_k$ an der äußeren Facette. Gesucht sind paarweise knotendisjunkte s_i - t_i -Wege in G ($1 \leq i \leq k$).

Das Problem heißt *topologisch lösbar*, falls die Reihenfolge der Terminale „um die äußere Facette herum“ die Lösbarkeit nicht ausschließt.

3 – Knotendisjunkte Wegpackung

Definition

$G = (V, E)$ mit $D = \{\{s_1, t_1\}, \dots, \{s_k, t_k\}\}$ hat *Klammerstruktur*, falls $G + D$ so planar eingebettet werden kann, dass die Kanten aus D kreuzungsfrei in die äußere Facette der entsprechenden Einbettung von G eingebettet sind.



s_6	s_2	s_1	t_1	t_2	s_3	s_4	t_4	t_3	t_6	s_5	t_5	s_7	t_7
((())	(()))	()	()

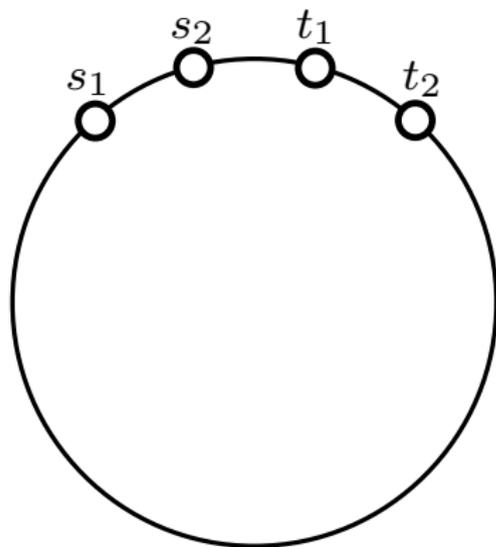
Aufgabe

Geben Sie für jede natürliche Zahl $k \geq 2$ einen zusammenhängenden, planaren Graphen G mit k Terminalpaaren (s_i, t_i) an der äußeren Facette an, sodass das knotendisjunkte Wegpackungsproblem

- nicht topologisch lösbar ist
- zwar topologisch lösbar ist, aber dennoch keine paarweise knotendisjunkten s_j - t_j -Wege in G existieren.

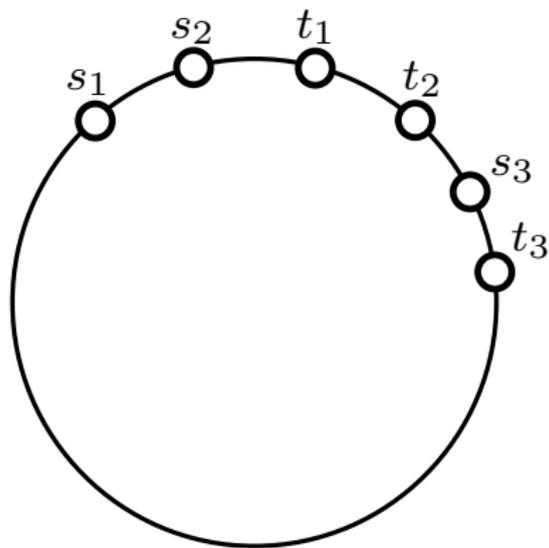
3.1 – Knotendisjunkte Wegpackung

Nicht topologisch lösbar.



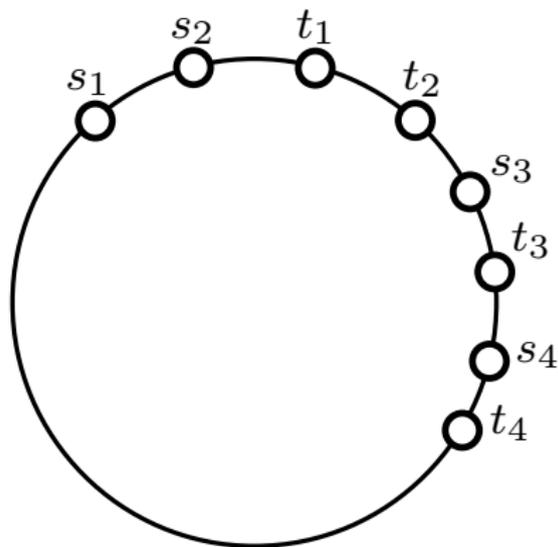
3.1 – Knotendisjunkte Wegpackung

Nicht topologisch lösbar.



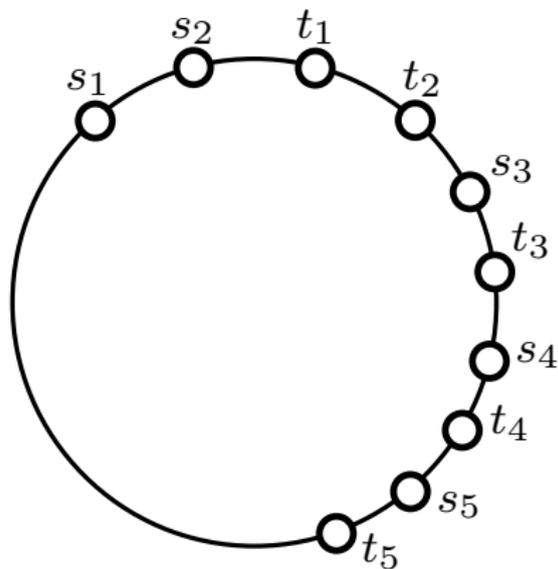
3.1 – Knotendisjunkte Wegpackung

Nicht topologisch lösbar.



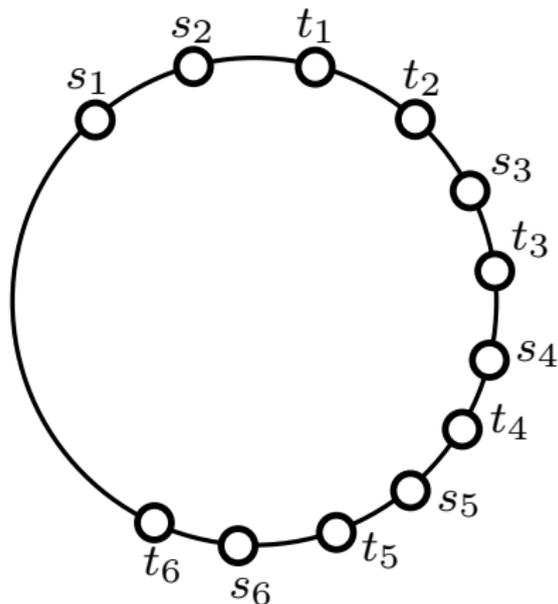
3.1 – Knotendisjunkte Wegpackung

Nicht topologisch lösbar.



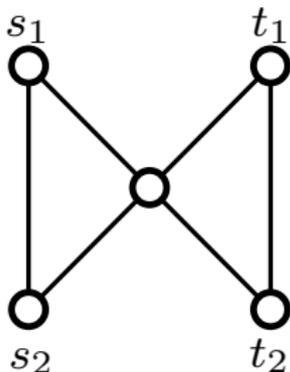
3.1 – Knotendisjunkte Wegpackung

Nicht topologisch lösbar.



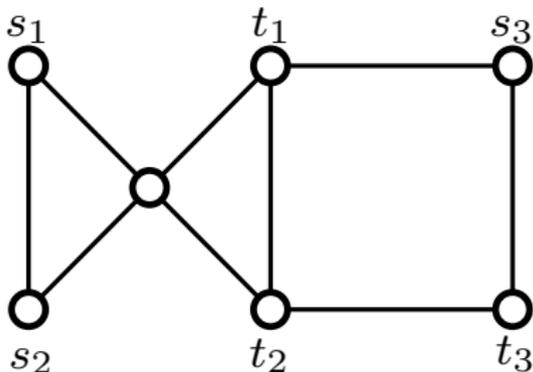
3.1 – Knotendisjunkte Wegpackung

Topologisch lösbar, aber Wege existieren nicht.



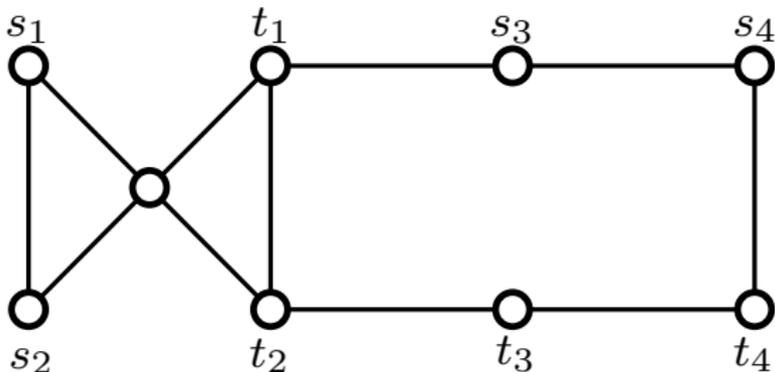
3.1 – Knotendisjunkte Wegpackung

Topologisch lösbar, aber Wege existieren nicht.



3.1 – Knotendisjunkte Wegpackung

Topologisch lösbar, aber Wege existieren nicht.

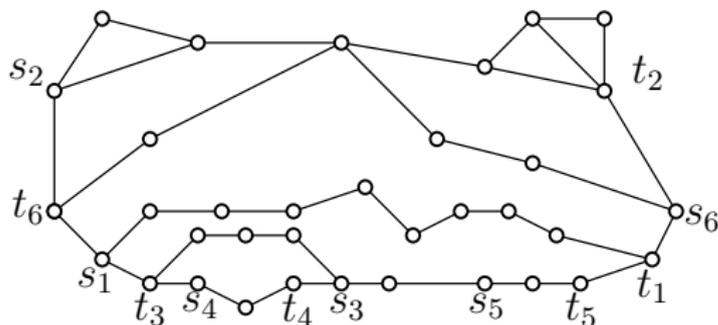


3.2 – Knotendisjunkte Wegpackung

Aufgabe

Sei o die Anzahl der Knoten von G , die zur äußeren Facette der Einbettung inzident sind (es gilt also $o \geq 2k$). Geben Sie einen Algorithmus an, der in Zeit $\mathcal{O}(o)$ feststellt, ob das knotendisjunkte Wegpackungsproblem topologisch lösbar ist.

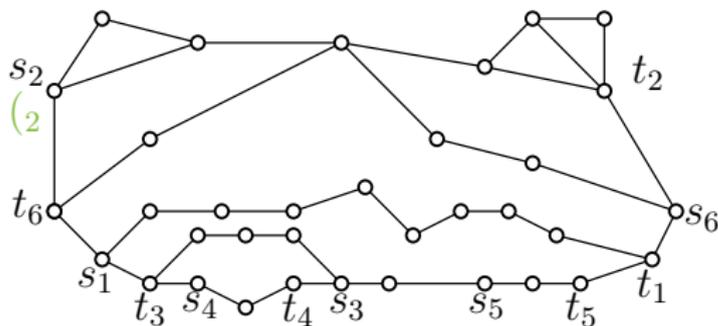
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
then
        return False
return  $STACK = \emptyset$ 
```

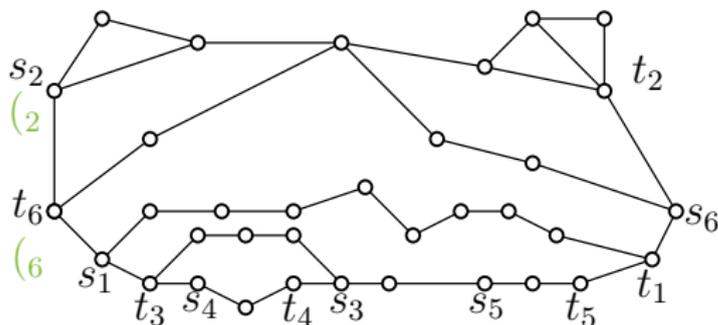
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
    ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
    then
        return False
return  $STACK = \emptyset$ 
```

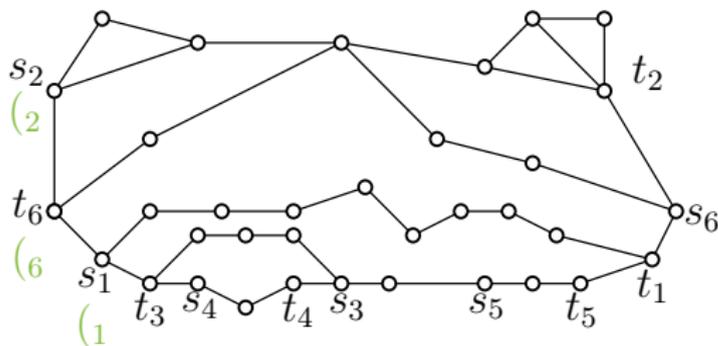
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
then
        return False
return  $STACK = \emptyset$ 
```

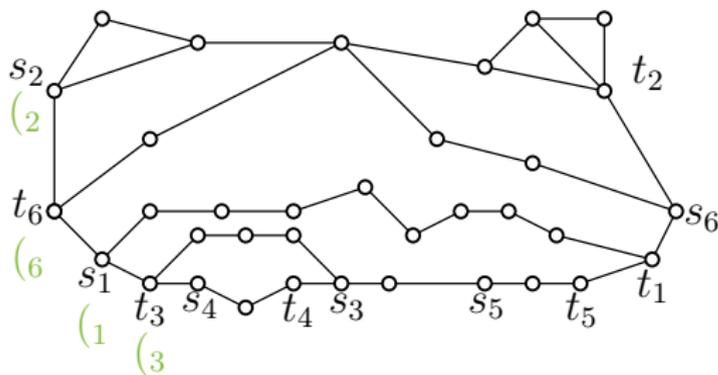
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
    ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
    then
        return False
return  $STACK = \emptyset$ 
```

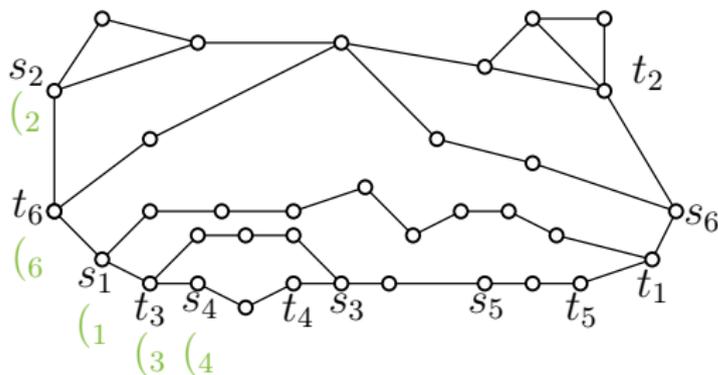
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
then
        return False
return  $STACK = \emptyset$ 
```

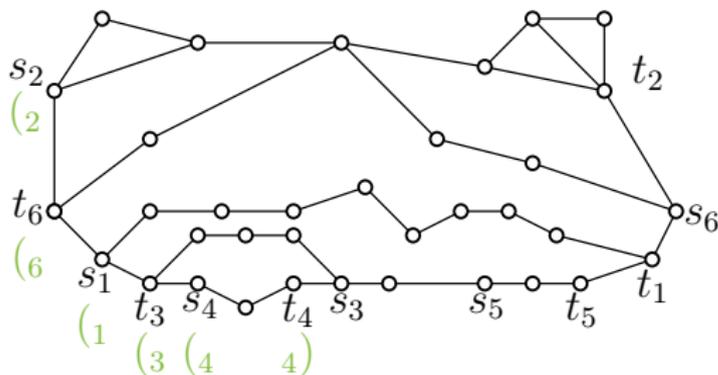
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
    ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
    then
        return False
return  $STACK = \emptyset$ 
```

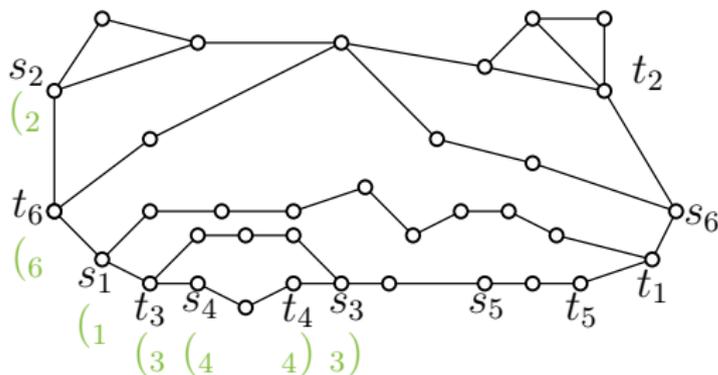
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
    ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
    then
        return False
return  $STACK = \emptyset$ 
```

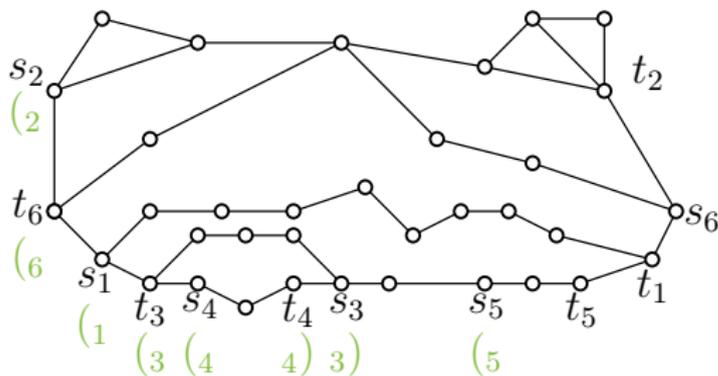
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
then
        return False
return  $STACK = \emptyset$ 
```

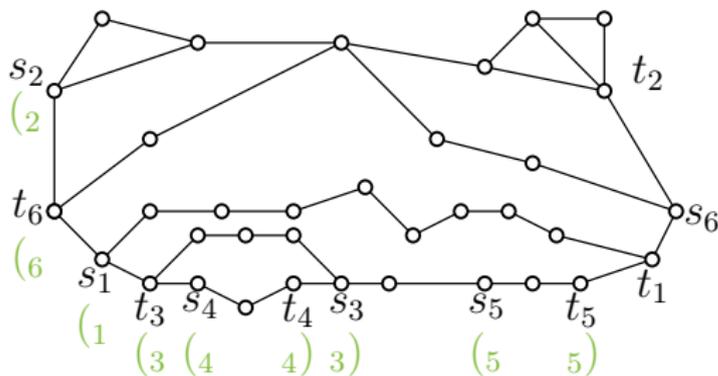
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
then
        return False
return  $STACK = \emptyset$ 
```

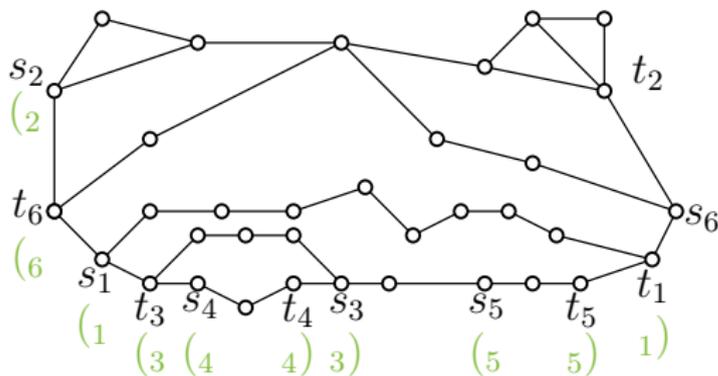
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
    ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
    then
        return False
return  $STACK = \emptyset$ 
```

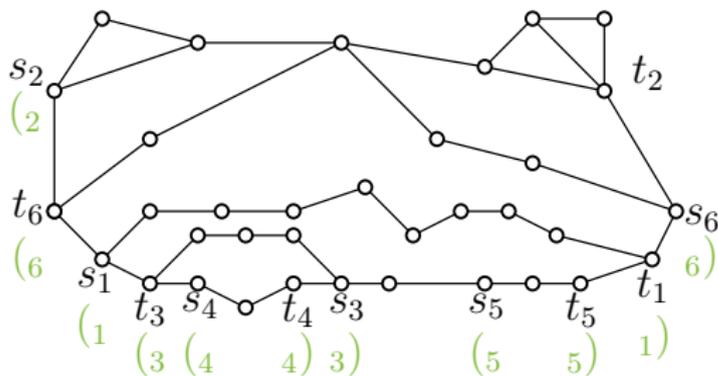
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
    ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
    then
        return False
return  $STACK = \emptyset$ 
```

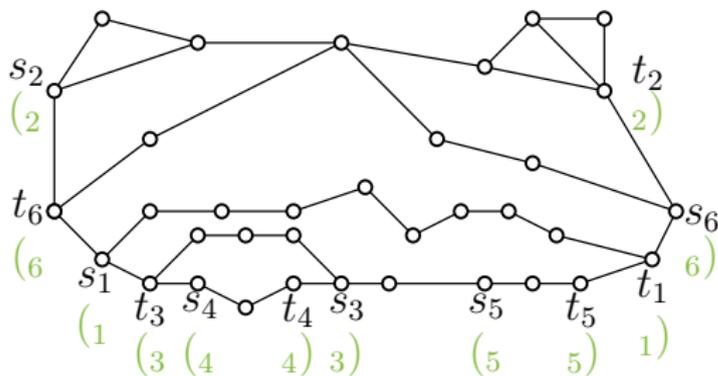
3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
then
        return False
return  $STACK = \emptyset$ 
```

3.2 – Knotendisjunkte Wegpackung



Algorithm KLAMMER

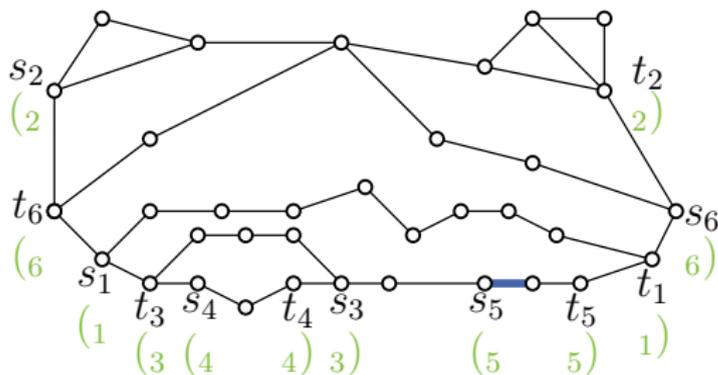
```
Laufe äußere Facette ab
if Knoten ist kein Terminal
then
    continue
if Terminal mit Index  $i$  zum
    ersten Mal besucht then
     $STACK.push(i)$ 
else
    if  $i \neq STACK.pop()$ 
    then
        return False
return  $STACK = \emptyset$ 
```

3.3 – Knotendisjunkte Wegpackung

Aufgabe

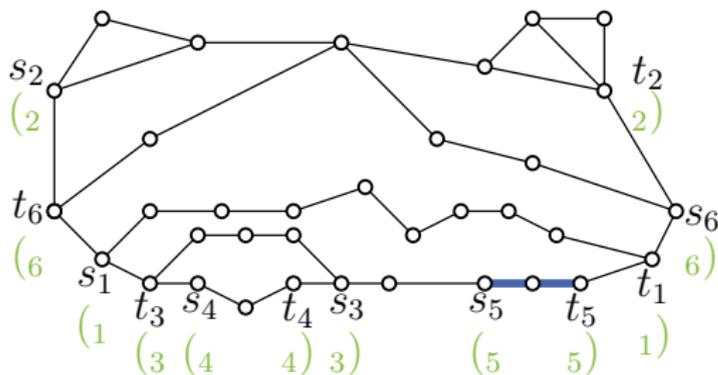
Angenommen, das Problem sei topologisch lösbar. Geben Sie einen Linearzeitalgorithmus an, der entweder k knotendisjunkte s_i - t_i -Wege findet oder feststellt, dass das Problem nicht lösbar ist.

3.2 – Knotendisjunkte Wegpackung



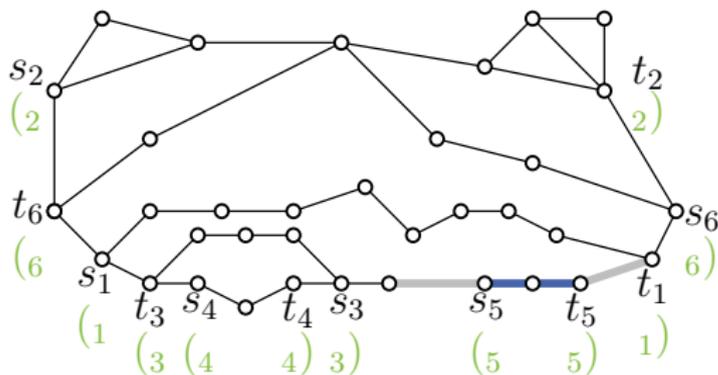
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



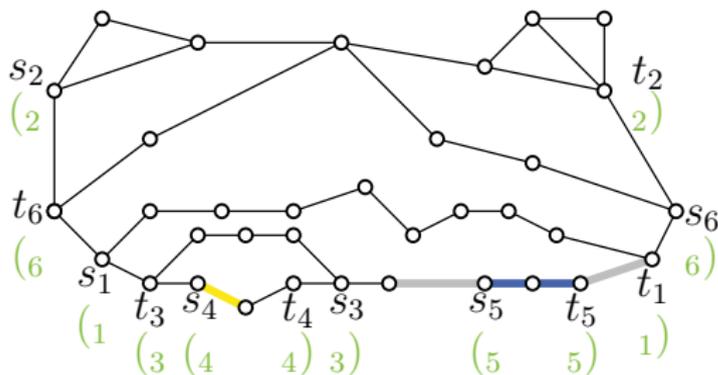
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



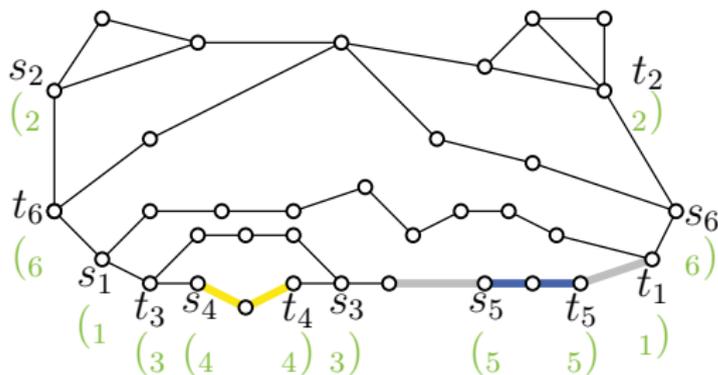
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



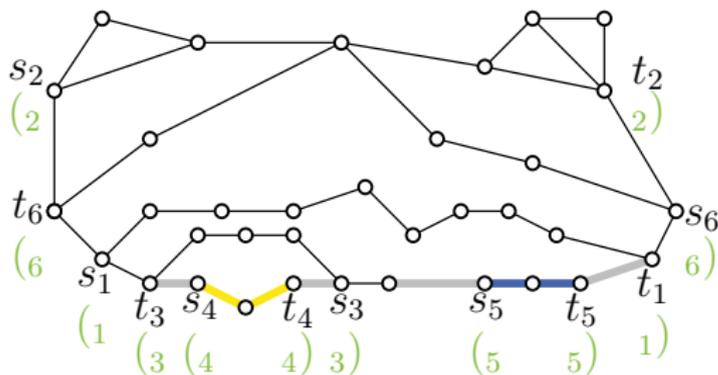
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



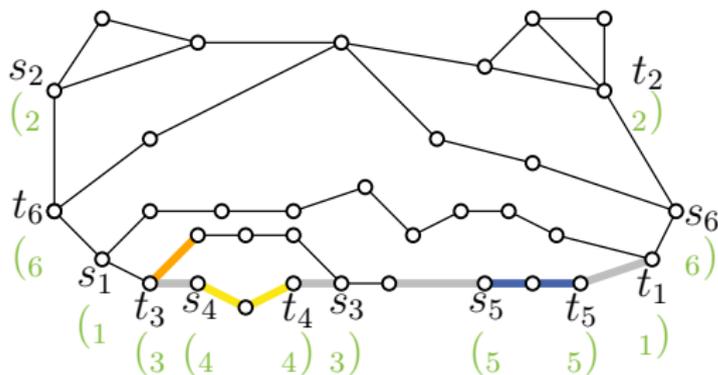
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



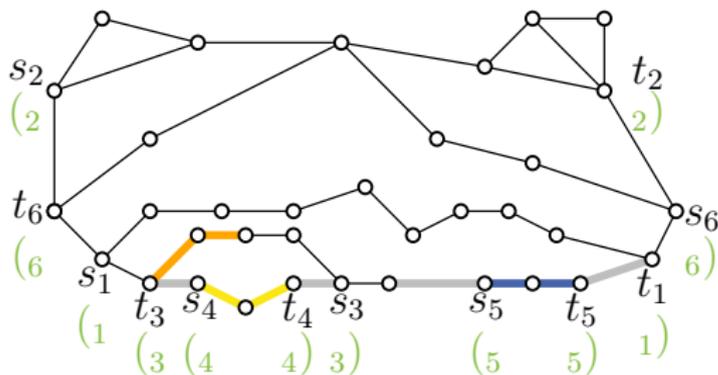
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



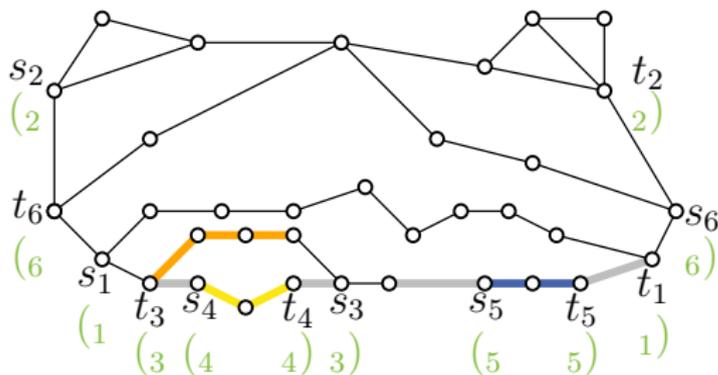
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



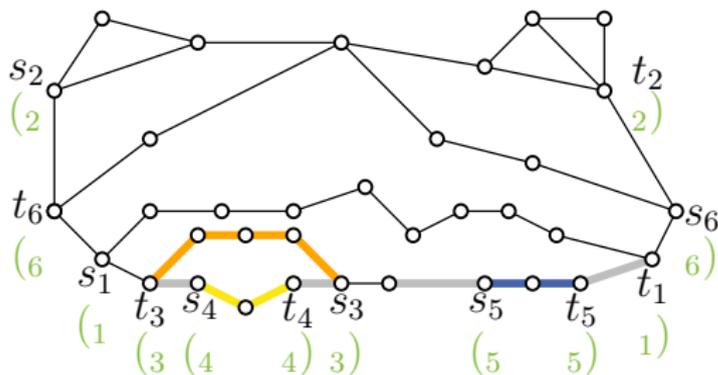
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



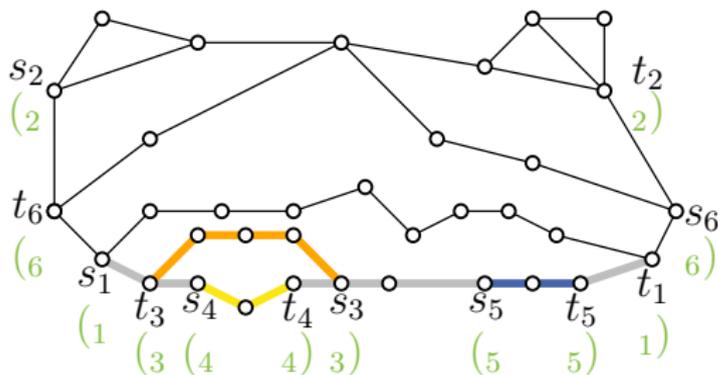
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



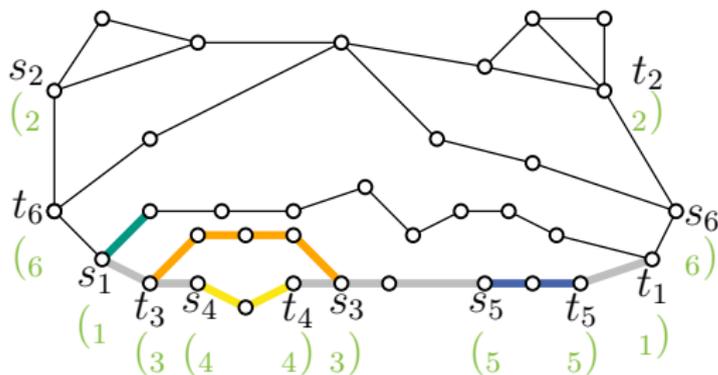
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



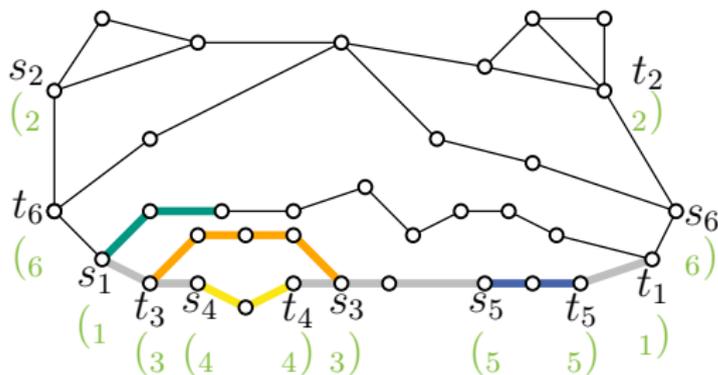
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



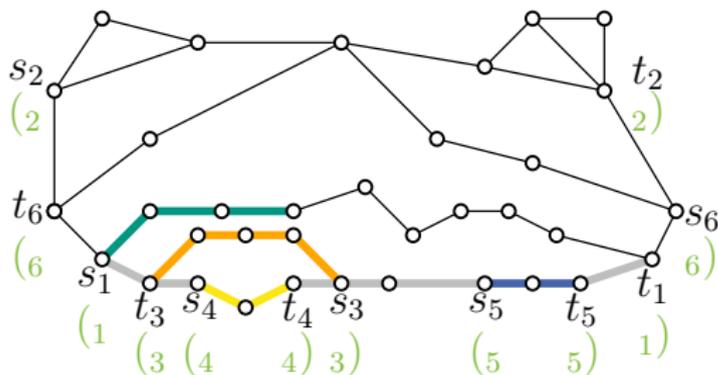
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



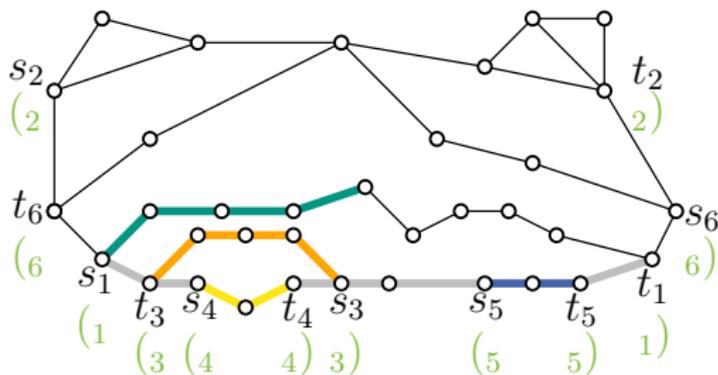
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



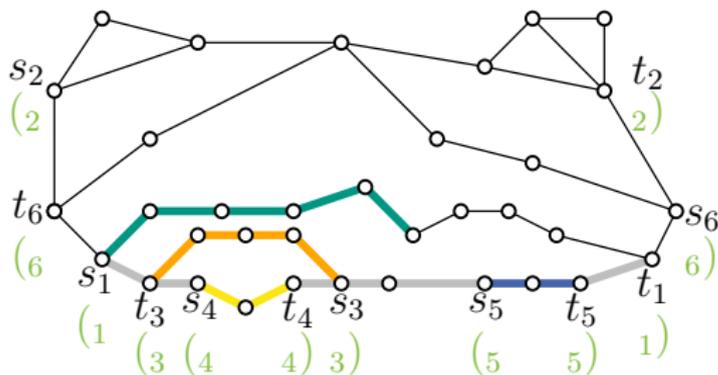
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



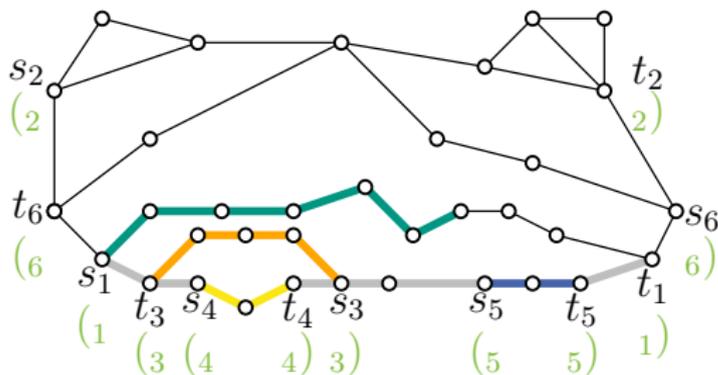
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



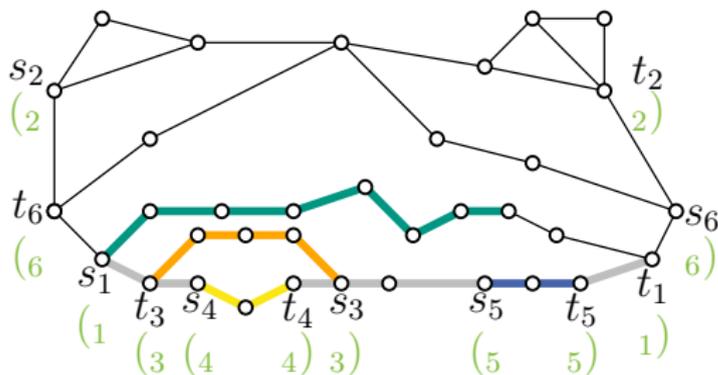
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



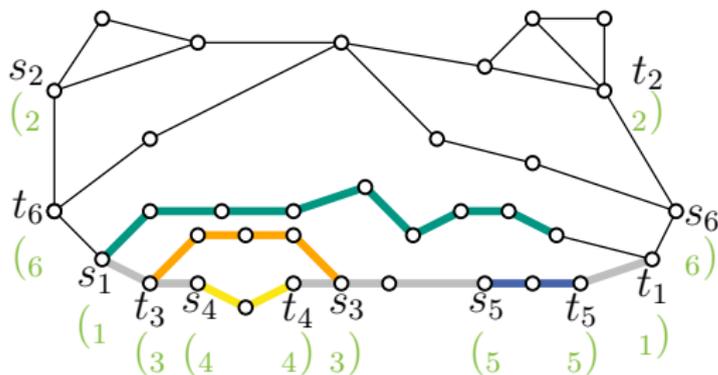
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



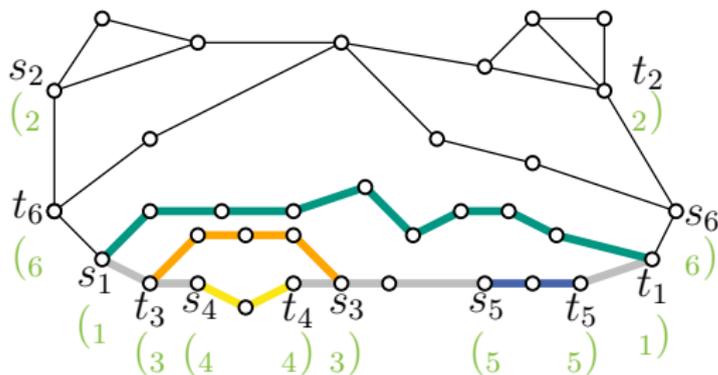
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



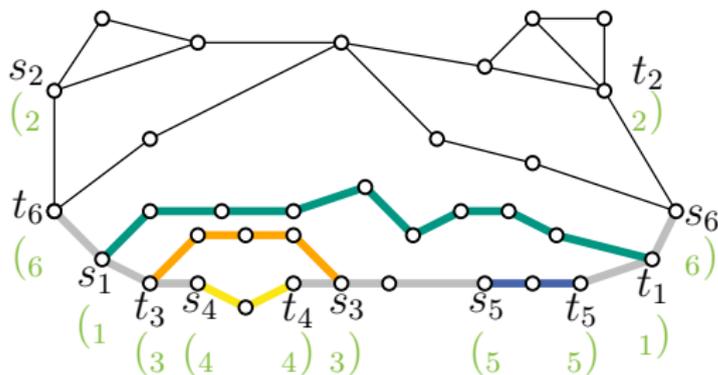
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



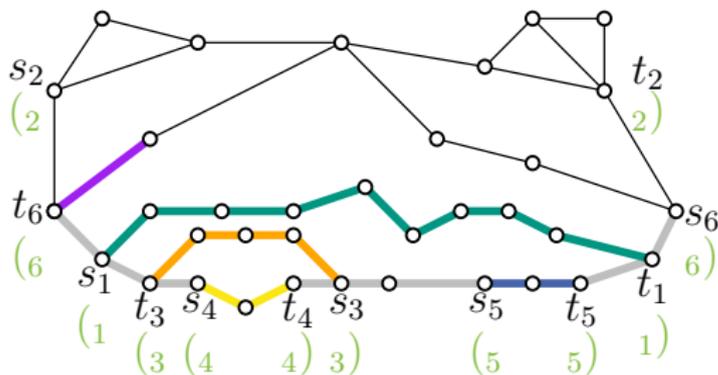
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



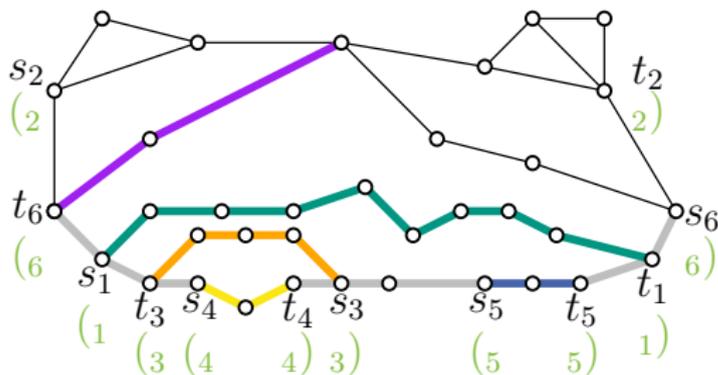
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



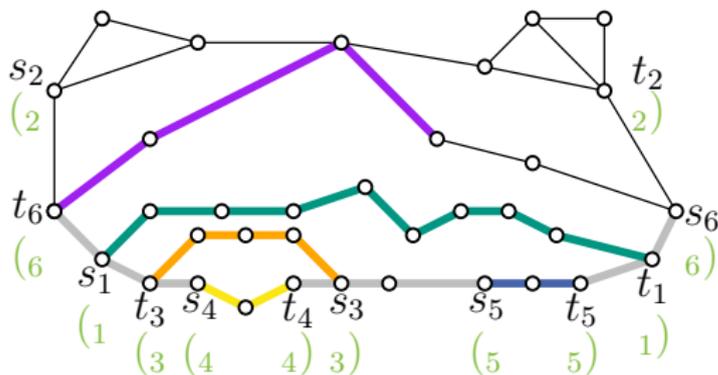
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



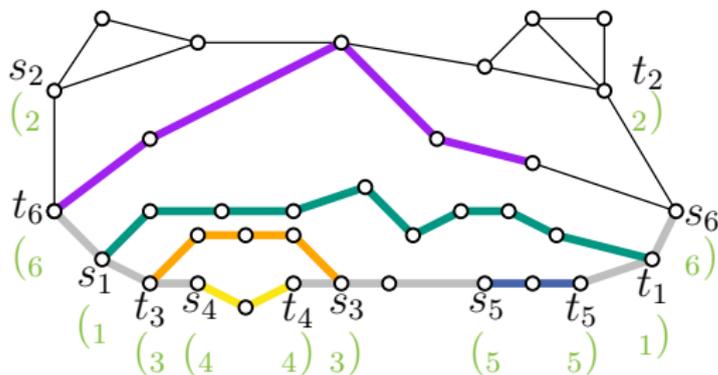
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



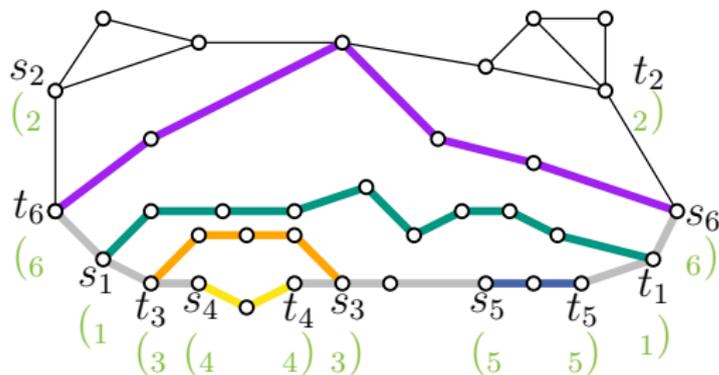
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



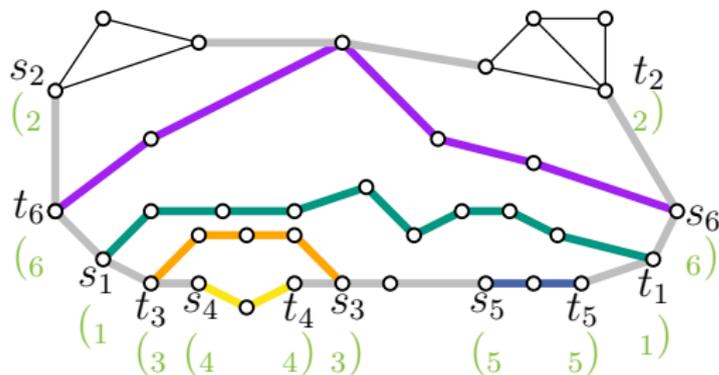
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



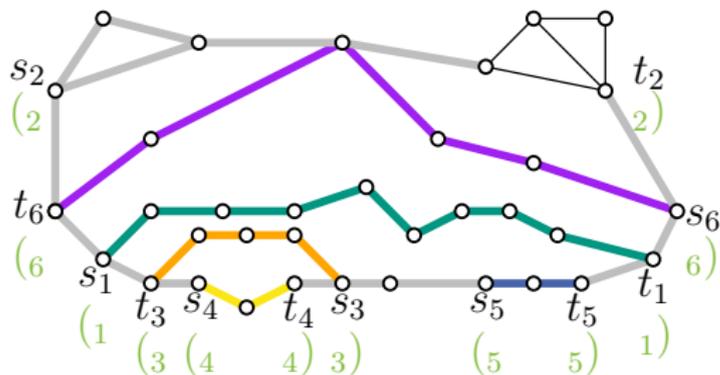
- Arbeite die Klammern von innen nach außen ab.
- Laufe entlang der äußeren Facette.
- Lösche Knoten (und inzidente Kanten) auf einem gefundenen Weg.

3.2 – Knotendisjunkte Wegpackung



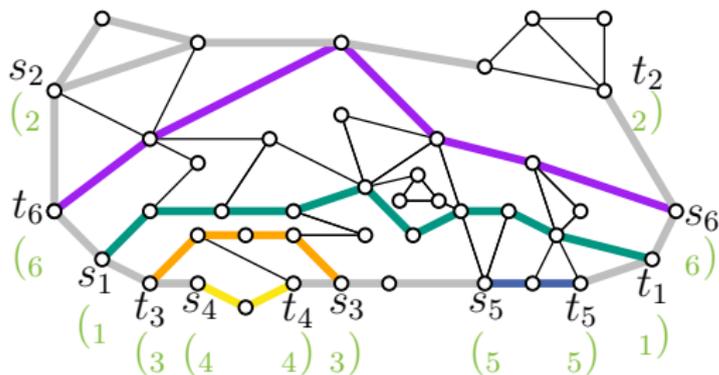
- Ist die Instanz nicht lösbar zerfällt der Graph in zwei Zusammenhangskomponenten und es wird ein Terminalpaar getrennt.

3.2 – Knotendisjunkte Wegpackung



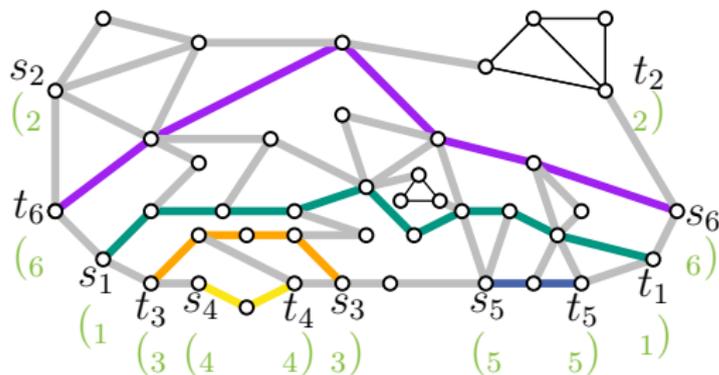
- Ist die Instanz nicht lösbar zerfällt der Graph in zwei Zusammenhangskomponenten und es wird ein Terminalpaar getrennt.

3.2 – Knotendisjunkte Wegpackung



- Ist die Instanz nicht lösbar zerfällt der Graph in zwei Zusammenhangskomponenten und es wird ein Terminalpaar getrennt.

3.2 – Knotendisjunkte Wegpackung



- Ist die Instanz nicht lösbar zerfällt der Graph in zwei Zusammenhangskomponenten und es wird ein Terminalpaar getrennt.

3.4 – Knotendisjunkte Wegpackung

Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

- nicht topologisch lösbar \Rightarrow nicht lösbar ✓

Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

- nicht topologisch lösbar \Rightarrow nicht lösbar \checkmark
- \exists Kurve $K \Rightarrow$ nicht lösbar \checkmark [K ist Separator]

Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

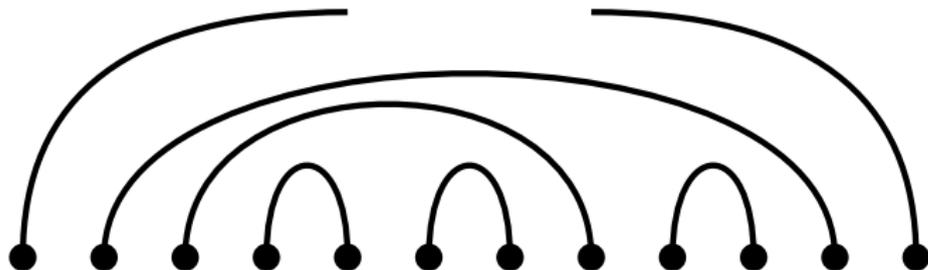
- nicht topologisch lösbar \Rightarrow nicht lösbar \checkmark
- \exists Kurve $K \Rightarrow$ nicht lösbar \checkmark [K ist Separator]
- Verbleibt: Topologisch lösbar und nicht lösbar $\Rightarrow \exists$ Kurve K

3.4 – Knotendisjunkte Wegpackung

Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

- nicht topologisch lösbar \Rightarrow nicht lösbar \checkmark
- \exists Kurve $K \Rightarrow$ nicht lösbar \checkmark [K ist Separator]
- Verbleibt: Topologisch lösbar und nicht lösbar $\Rightarrow \exists$ Kurve K

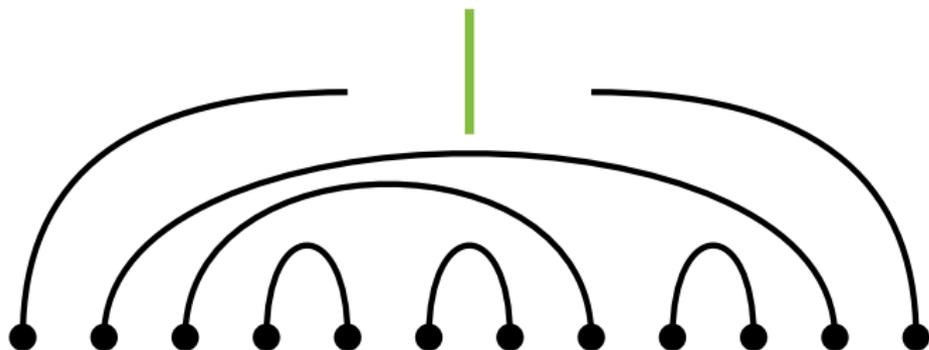


3.4 – Knotendisjunkte Wegpackung

Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

- nicht topologisch lösbar \Rightarrow nicht lösbar \checkmark
- \exists Kurve $K \Rightarrow$ nicht lösbar \checkmark [K ist Separator]
- Verbleibt: Topologisch lösbar und nicht lösbar $\Rightarrow \exists$ Kurve K

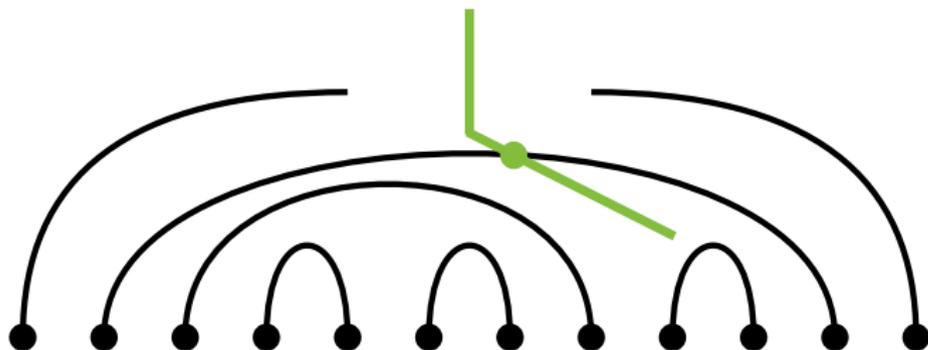


3.4 – Knotendisjunkte Wegpackung

Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

- nicht topologisch lösbar \Rightarrow nicht lösbar \checkmark
- \exists Kurve $K \Rightarrow$ nicht lösbar \checkmark [K ist Separator]
- Verbleibt: Topologisch lösbar und nicht lösbar $\Rightarrow \exists$ Kurve K

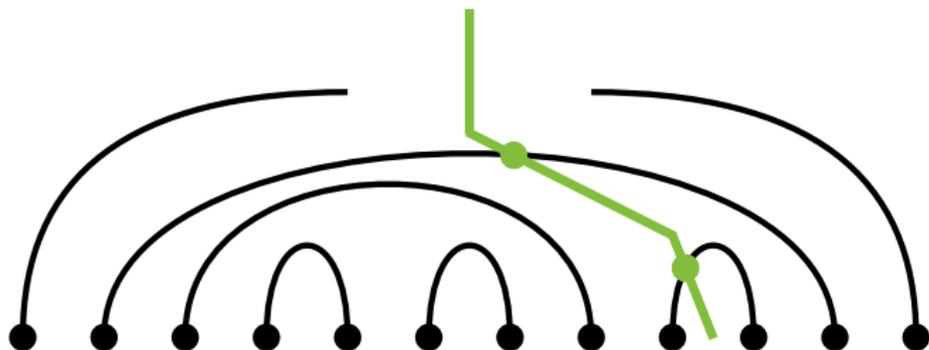


3.4 – Knotendisjunkte Wegpackung

Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

- nicht topologisch lösbar \Rightarrow nicht lösbar \checkmark
- \exists Kurve $K \Rightarrow$ nicht lösbar \checkmark [K ist Separator]
- Verbleibt: Topologisch lösbar und nicht lösbar $\Rightarrow \exists$ Kurve K

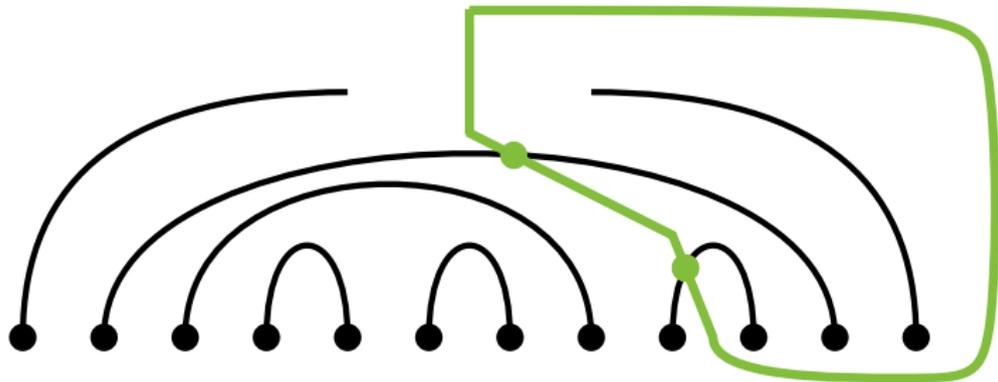


3.4 – Knotendisjunkte Wegpackung

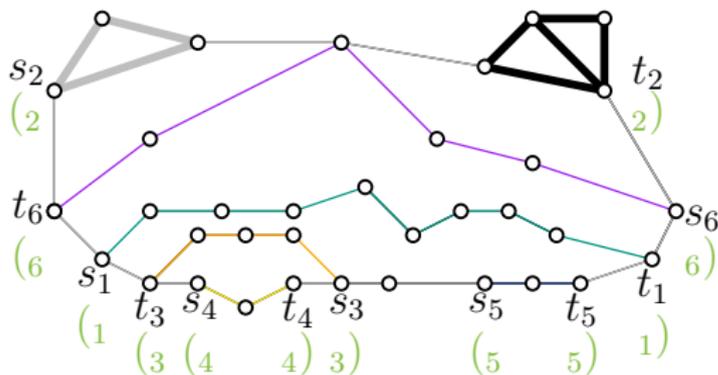
Zeigen Sie:

Problem nicht lösbar \Leftrightarrow nicht topologisch lösbar oder \exists Kurve K die G nur in Knoten schneidet und mehr Terminalpaare trennt als Knoten schneidet.

- nicht topologisch lösbar \Rightarrow nicht lösbar \checkmark
- \exists Kurve $K \Rightarrow$ nicht lösbar \checkmark [K ist Separator]
- Verbleibt: Topologisch lösbar und nicht lösbar $\Rightarrow \exists$ Kurve K

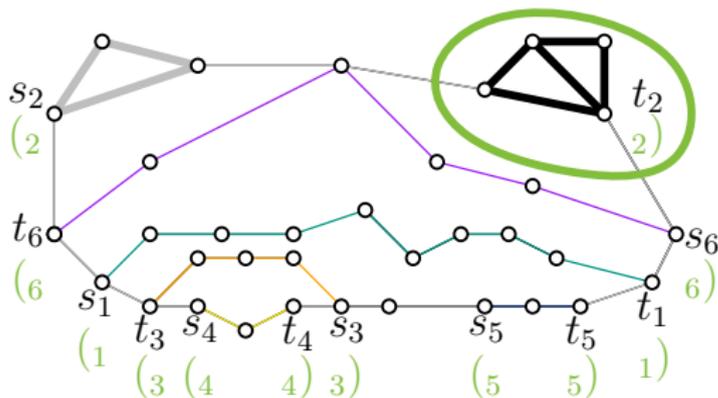


3.2 – Knotendisjunkte Wegpackung



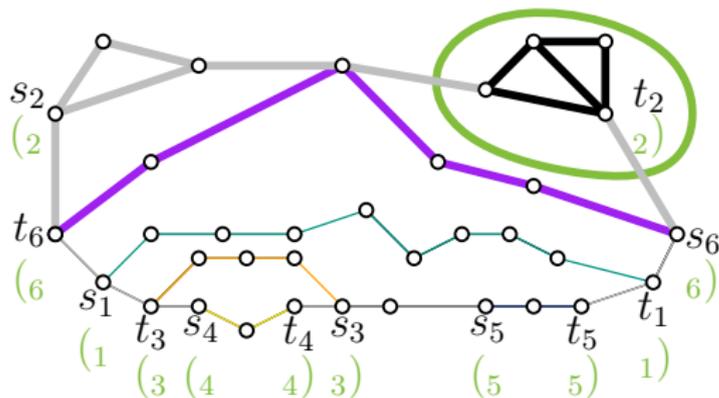
- Betrachte den zerfallenen Graphen.
- K lässt sich zeichnen ohne einen Knoten zu berühren und trennt mindestens ein Terminalpaar.

3.2 – Knotendisjunkte Wegpackung



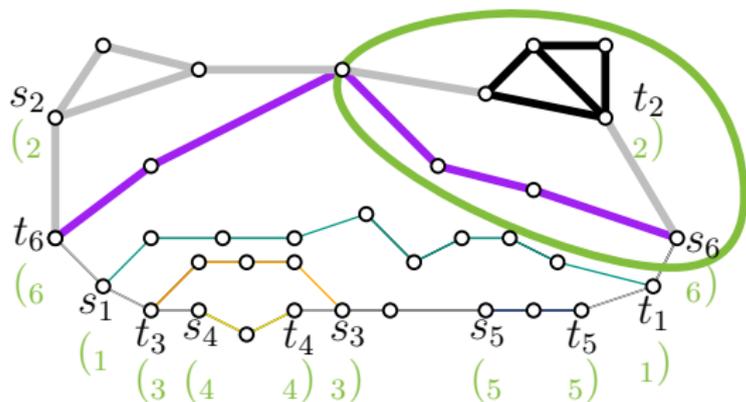
- Betrachte den zerfallenen Graphen.
- K lässt sich zeichnen ohne einen Knoten zu berühren und trennt mindestens ein Terminalpaar.

3.2 – Knotendisjunkte Wegpackung



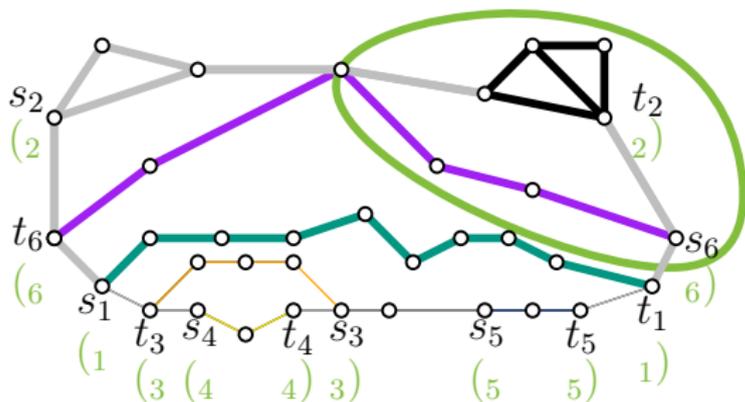
- Füge den zuletzt gefundenen Pfad wieder ein. Schneide in einem Knoten.
- Im aktuellen Zustand liegt Kurve wieder in äußerer Facette.

3.2 – Knotendisjunkte Wegpackung



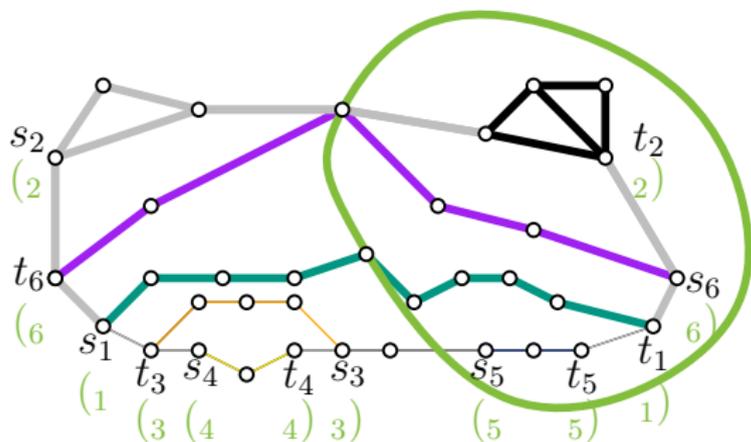
- Füge den zuletzt gefundenen Pfad wieder ein. Schneide in einem Knoten.
- Im aktuellen Zustand liegt Kurve wieder in äußerer Facette.

3.2 – Knotendisjunkte Wegpackung



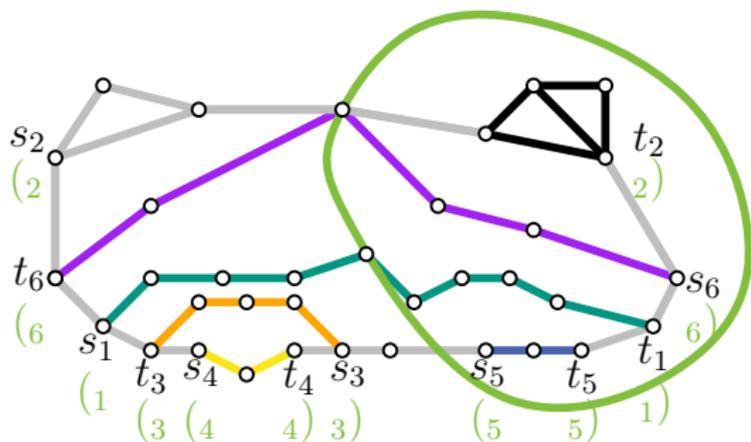
- Füge den zuletzt gefundenen Pfad wieder ein. Schneide in einem Knoten.
- Im aktuellen Zustand liegt Kurve wieder in äußerer Facette.

3.2 – Knotendisjunkte Wegpackung



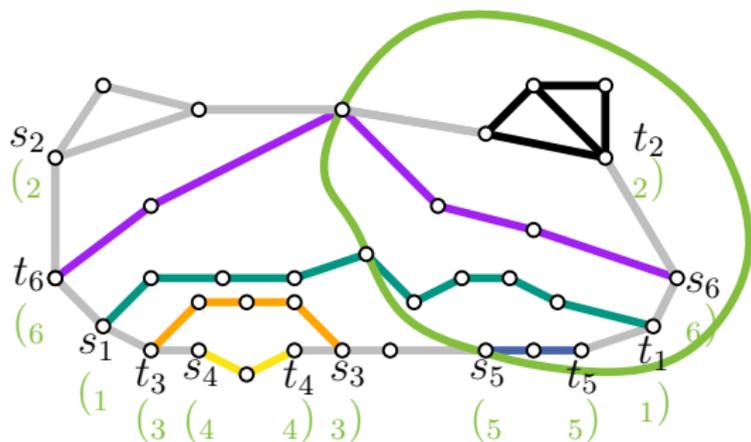
- Füge den zuletzt gefundenen Pfad wieder ein. Schneide in einem Knoten.
- Im aktuellen Zustand liegt Kurve wieder in äußerer Facette.

3.2 – Knotendisjunkte Wegpackung



- Füge den zuletzt gefundenen Pfad wieder ein. Schneide in einem Knoten.
- Im aktuellen Zustand liegt Kurve wieder in äußerer Facette.

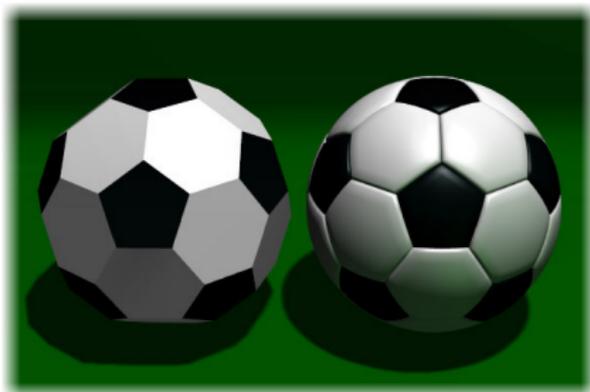
3.2 – Knotendisjunkte Wegpackung



- Füge den zuletzt gefundenen Pfad wieder ein. Schneide in einem Knoten.
- Im aktuellen Zustand liegt Kurve wieder in äußerer Facette.

Ein Fußball ist ein Polyeder dessen Oberfläche aus regelmäßigen Fünf- und Sechsecken besteht.

- Zeigen Sie: Die Anzahl der Fünfecke ist 12.



- Fasse Polyeder als Graph $G = (V, E)$ auf.

⇒ Der Satz von Euler gilt.

- Sei

n = Anzahl Knoten

m = Anzahl Kanten

f_5 = Anzahl Fünfecke

f_6 = Anzahl Sechsecke

f = Anzahl Facetten ($= f_5 + f_6$)

- Für k -Ecke gilt: Innenwinkelsumme $= (k - 2) \cdot 180^\circ$

- Jeder Knoten hat Grad 3.
 - Offensichtlich: Jeder Knoten ist inzident zu mindestens 3 Facetten.
 - Zu zeigen: Jeder Knoten ist inzident zu maximal 3 Facetten.
 - Facette mit 5 Knoten: Summe der Innenwinkel = 540°
 - Facette mit 6 Knoten: Summe der Innenwinkel = 720°
- ⇒ Da die Fünfecke regelmäßig sind, ist ein Innenwinkel mindestens $\frac{540^\circ}{5} = 108^\circ (< \frac{720^\circ}{6} = 120^\circ)$
 - Die Summe der Innenwinkel um einen Knoten ist < 360 .
 - ($= 360^\circ$ für Facetten in einer Ebene)

$$\begin{aligned} \blacksquare 2m &= \sum_{v \in V} \delta(v) = \sum_{v \in V} 3 = 3n \\ \Rightarrow n &= \frac{2}{3}m \end{aligned}$$

- Jeder Knoten hat Grad 3.

- $2m = \sum_{v \in V} \delta(v) = \sum_{v \in V} 3 = 3n$

$$\Rightarrow n = \frac{2}{3}m$$

- $2m = \sum_{f \in F} \delta(f) = 5f_5 + 6f_6$

- $\delta(f) = \text{Anzahl Kanten inzident zu } f$

$$\Rightarrow m = \frac{5}{2} \cdot f_5 + \frac{6}{2} \cdot f_6$$

- Euler

- $n - m + f = 2$

- $\frac{2}{3}m - m + f = 2$

- $-\frac{1}{3}m + f = 2$

- $-\frac{5}{6} \cdot f_5 - \frac{6}{6} \cdot f_6 + f_5 + f_6 = 2$

- $\frac{1}{6}f_5 = 2$

- $f_5 = 12$