Karlsruher Institut für Technologie

Algorithmen für Planare Graphen
Übung am 20.06.2017

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Organisatorisches

Terminänderungen:

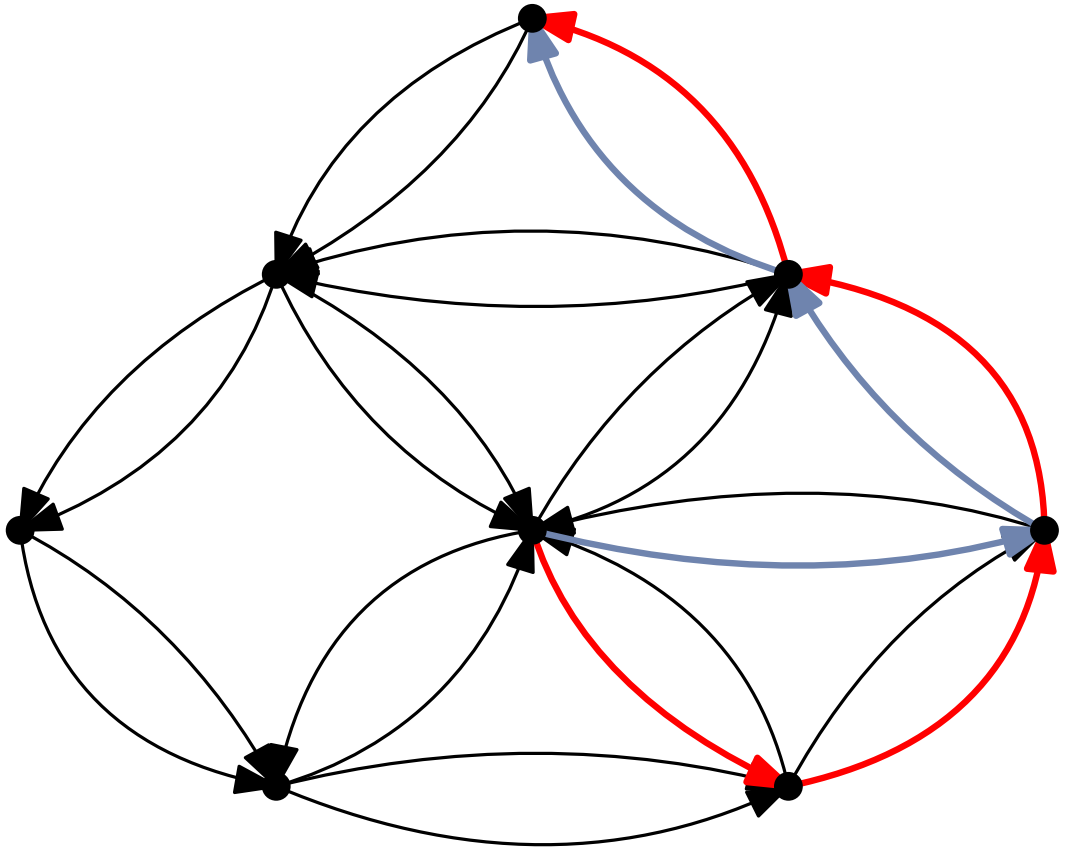
- letzte Übung am 3. Juli
- letzte Vorlesung am 4. Juli

Prüfungstermine:

- 24. Juli
- 9. August
- 10. August
- 22. August
- 13. September
- 21. September
- 25. September
- 9. Oktober

Effizient rechteste freie Kante finden

Hintergrund: finde *st*-Pfade möglichst weit rechts



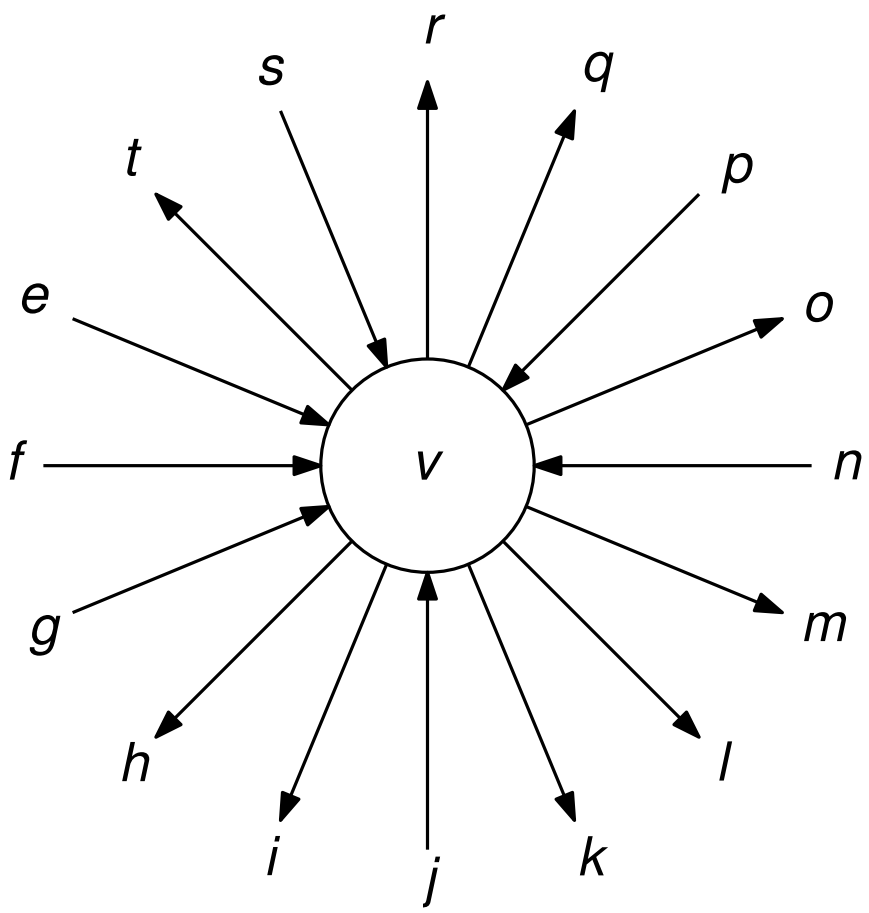
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Effizient rechteste freie Kante finden

Ziel: effizient rechteste 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



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Union-Find Datenstruktur

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 $x \in S$

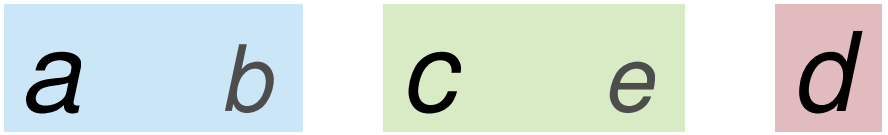
Union-Find Datenstruktur

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 $x \in S$

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

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



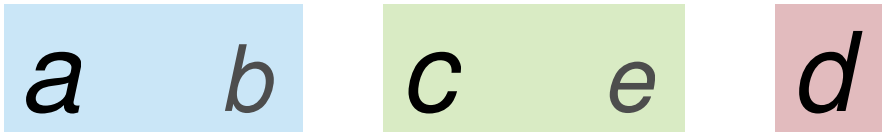
UNION(a, e) $\rightarrow$

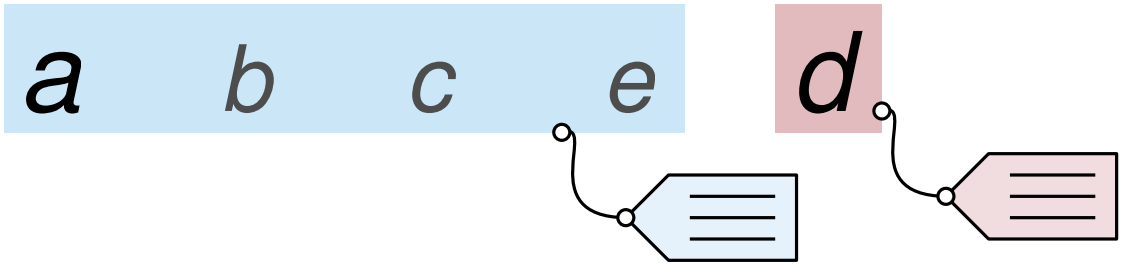
Union-Find Datenstruktur

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 $x \in S$

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

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

UNION(a, e) $\rightarrow$ 

Bemerkung: FIND(x) kann zusätzliche Attribute erhalten

Union-Find Datenstruktur

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]

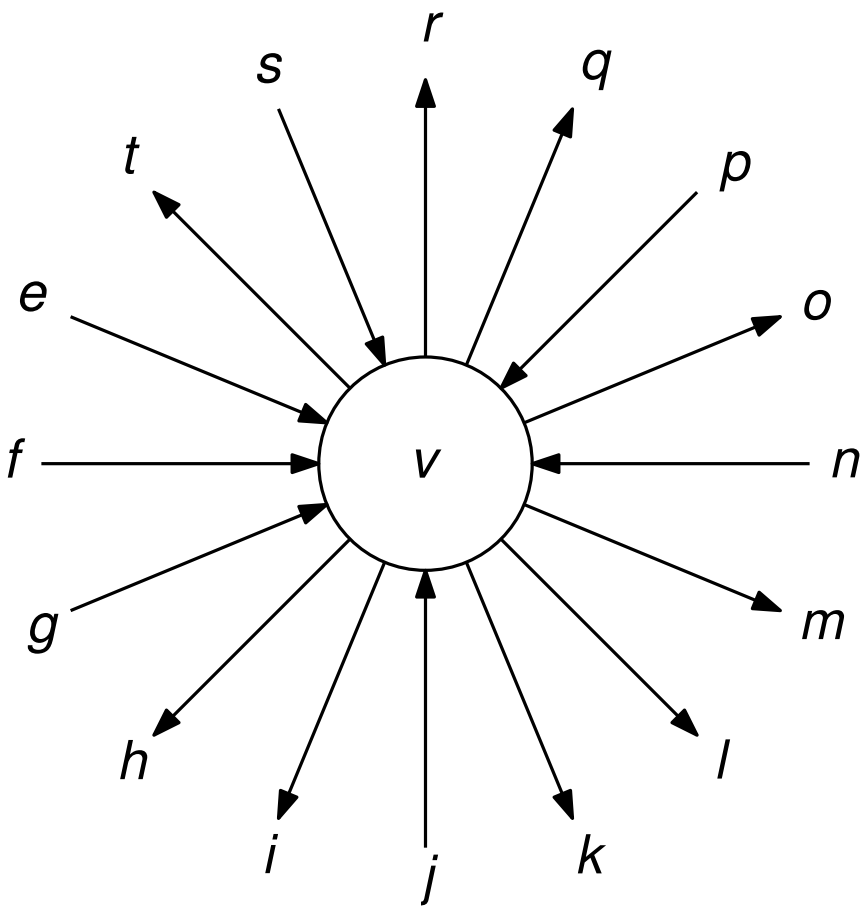
- UNION(x, y) $\in \mathcal{O}(1)$
- FIND(x, y) $\in \mathcal{O}(1)$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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

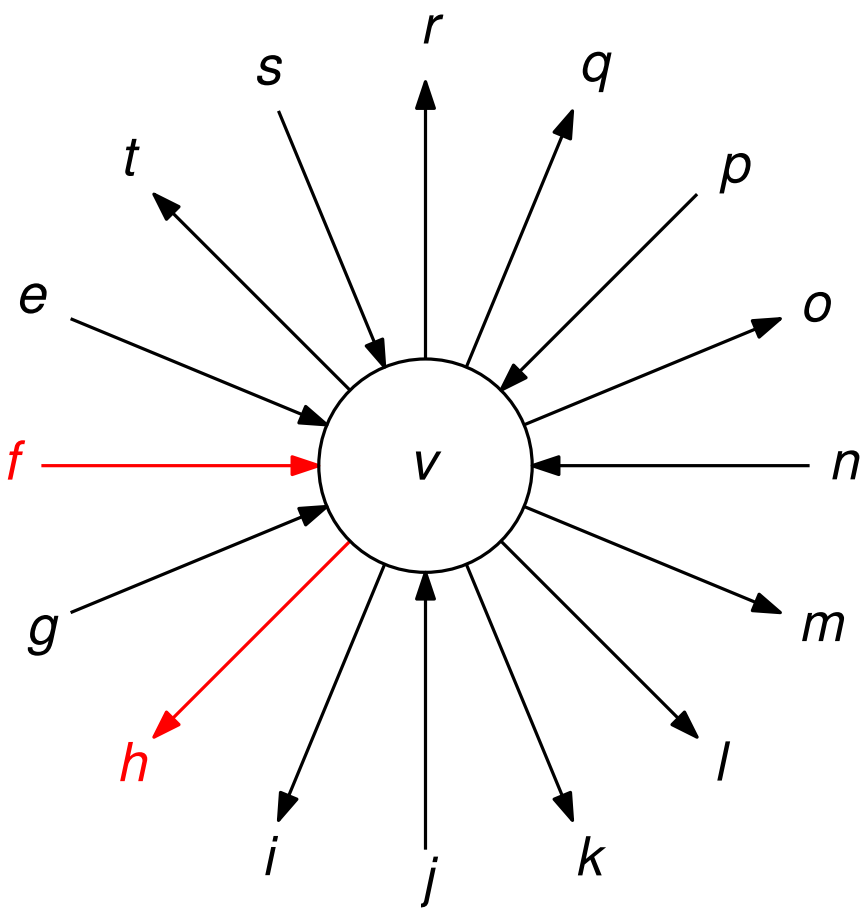


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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

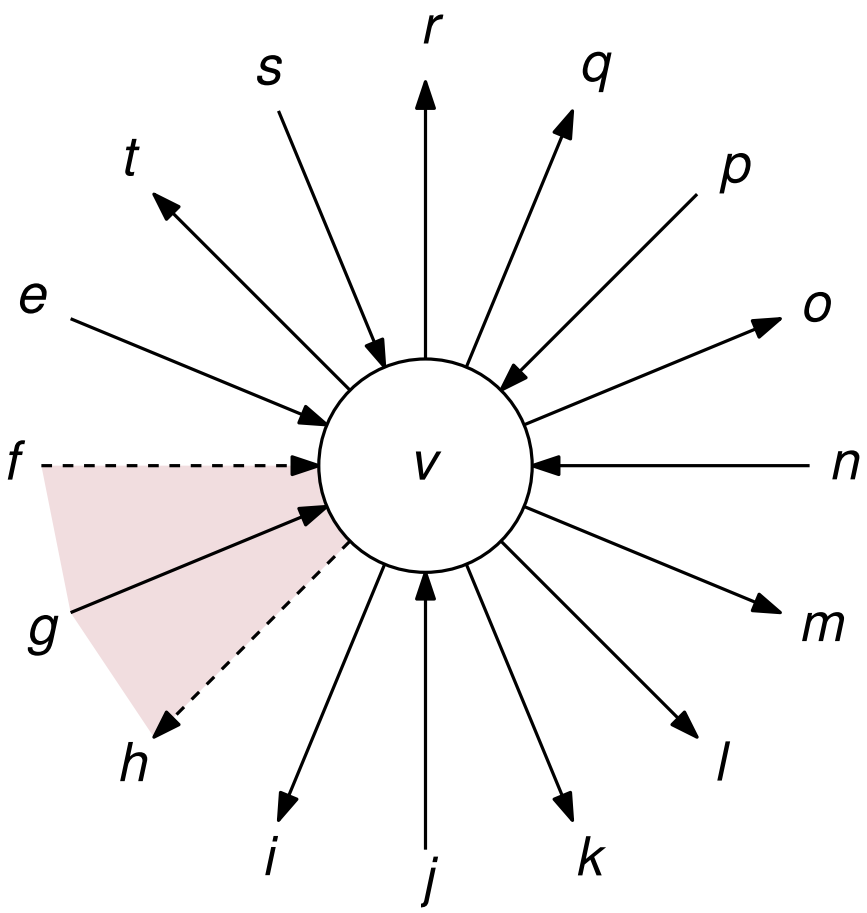


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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

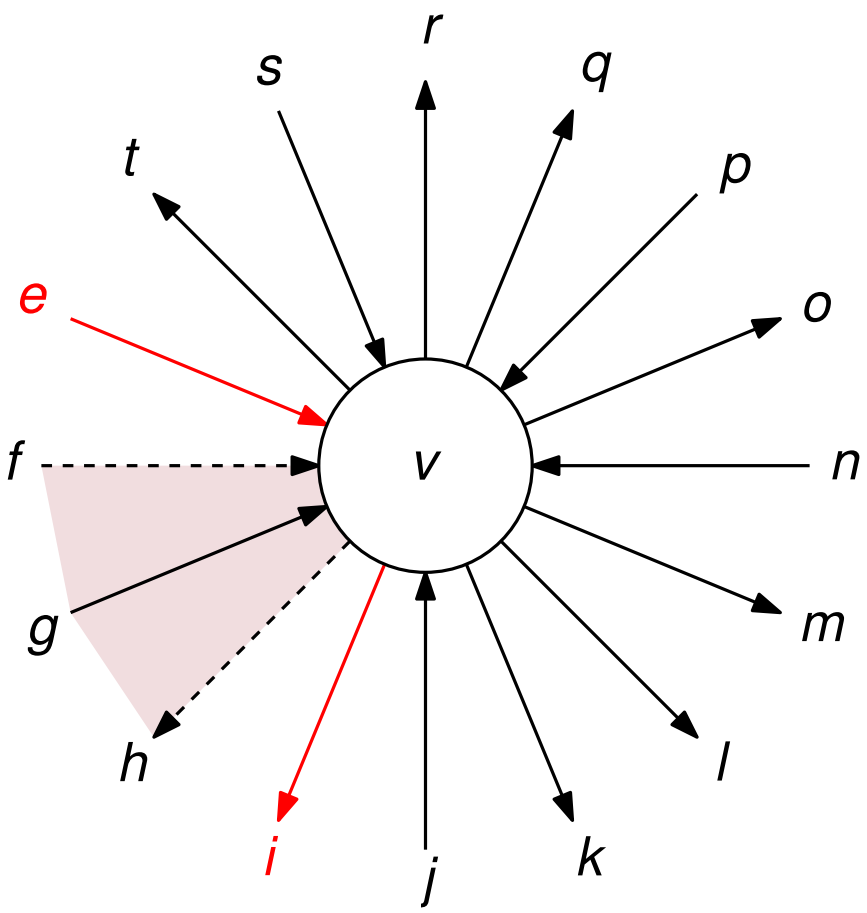


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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

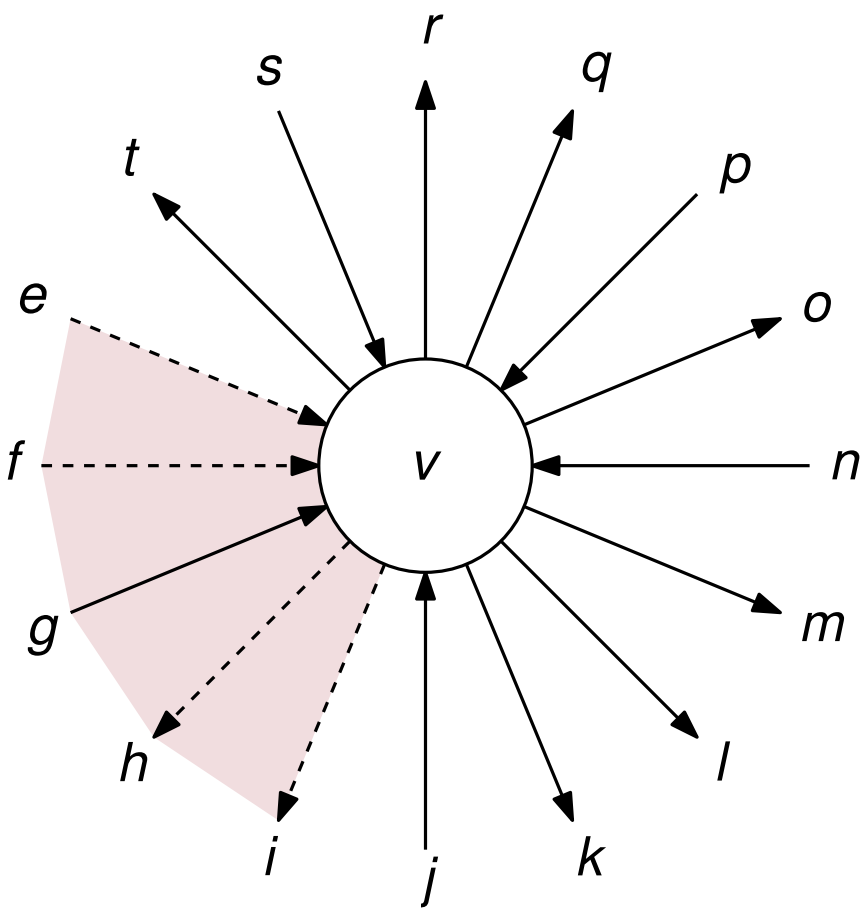


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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

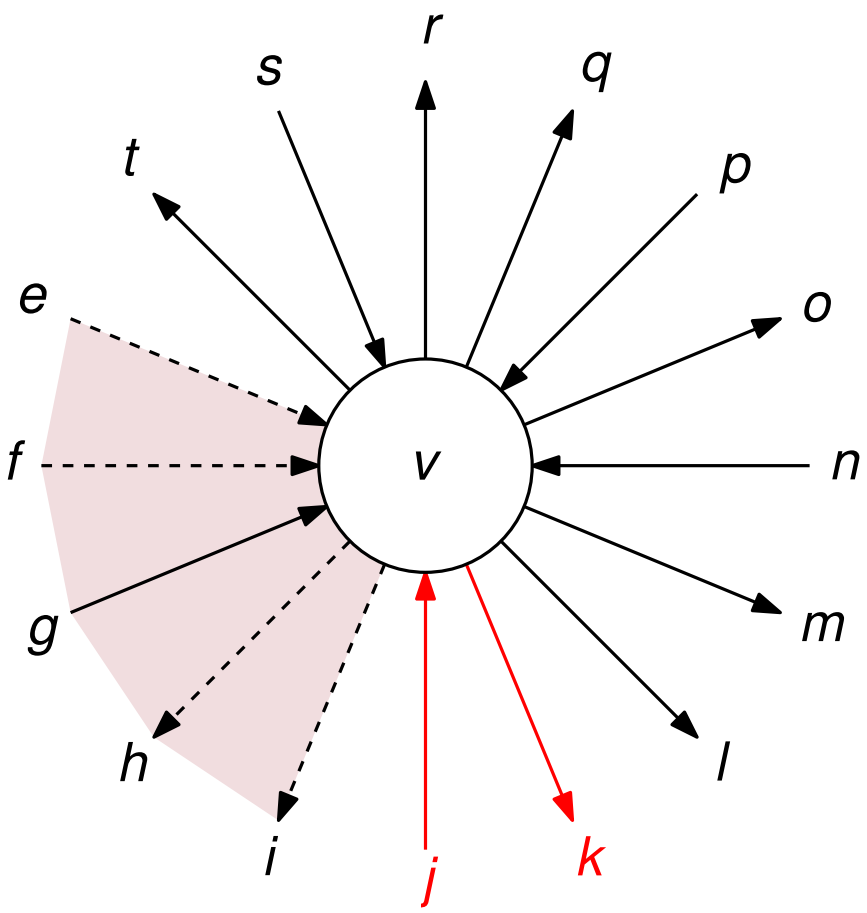


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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

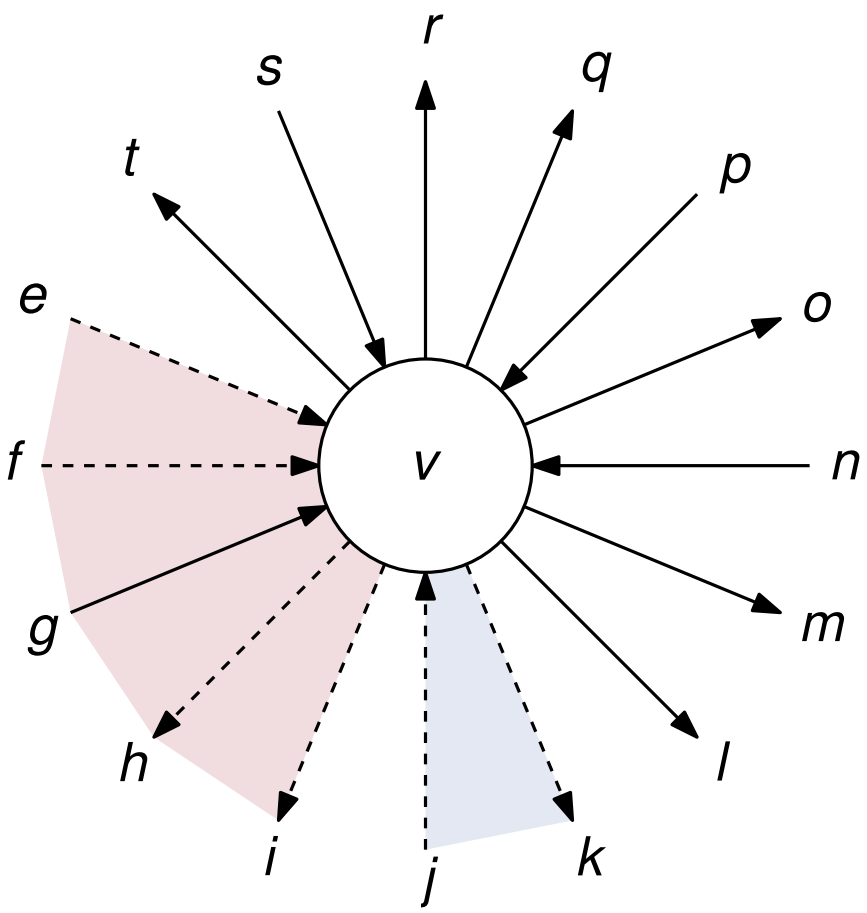


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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

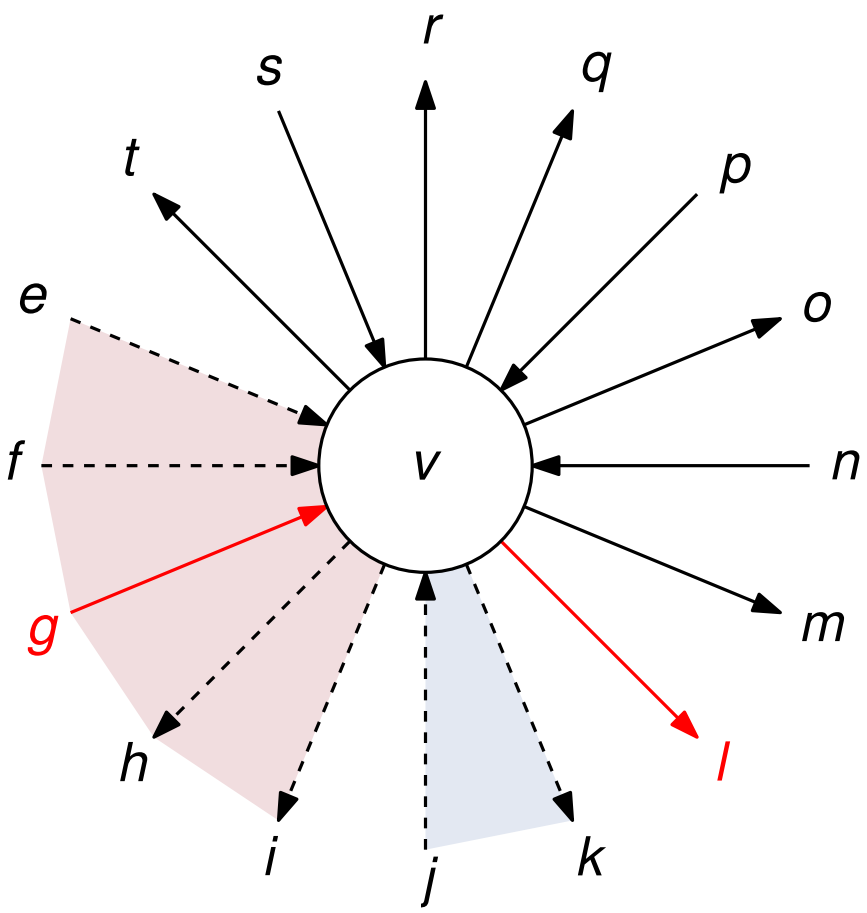


Intuition: überspringe Abschnitte ohne freie Kante effizient

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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



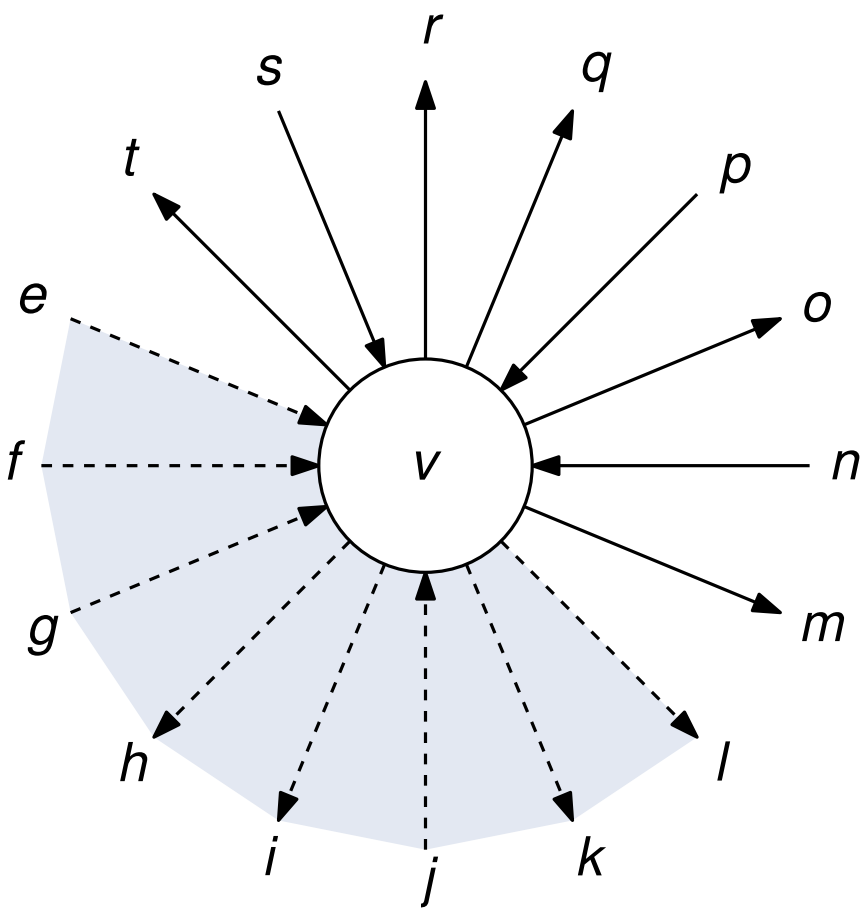
Intuition: überspringe Abschnitte ohne freie Kante effizient

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Effizient rechteste freie Kante finden

Ziel: effizient rechteste freie Kante finden

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



Intuition: überspringe Abschnitte ohne freie Kante effizient

Evaluation

Karlsruher Institut für Technologie

5. Übungsblatt

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Mixed-Max-Cut

Definition: Mixed-Max-Cut

Gegeben sei ein Graph $G = (V, E)$ mit einer Kantengewichtsfunktion $w\colon E \to K$, wobei $K = \mathbb{R}$. Finde einen Schnitt $S \subseteq E$ mit $w(S)$ maximal.

Definition: Mixed-Max-Cycle

Gegeben sei ein planarer Graph $G = (V, E)$ mit einer Kantengewichtsfunktion $w\colon E \to K$, wobei $K = \mathbb{R}$. Finde eine nichtleere gerade Menge $E' \subseteq E$ mit $w(E')$ maximal.

Mixed-Max-Cut auf G entspricht Mixed-Max-Cycle auf Dualgraph $G^\star$

Kantenmenge $E' \subseteq E$ heißt *gerade*, wenn in dem durch E' induzierten Subgraph von G jeder Knoten geraden Grad hat.

Mixed-Max-Cut

hritt 1: Trianguliere G in $O(n)$ und ordne den hinzugefügten Kanten Gewicht 0 zu.

hritt 2: Berechne in $O(n)$ den Dualgraph $G^* = (V^*, E^*)$ zu der Triangulierung von G, wobei $w(e^*) := w(e)$ mit e^* Dualkante zu e.

hritt 3: Konstruiere aus G^* in $O(n)$ einen Graph $G' = (V', E')$ derart, dass ein perfektes Matching minimalen Gewichts in G' eine gerade Menge maximalen Gewichts in G^* induziert.

hritt 4: Konstruiere in $O(n^{\frac{3}{2}} \log n)$ ein perfektes Matching M minimalen Gewichts in G'.

hritt 5: Falls M eine nichtleere gerade Menge in E^* induziert, gib den dazu dualen Schnitt in G aus. Ansonsten berechne in $O(n^{\frac{3}{2}} \log n)$ aus M eine nichttriviale gerade Menge in G^* maximalen Gewichts.

Mixed-Max-Cut

hritt 1: Trianguliere G in $O(n)$ und ordne den hinzugefügten Kanten Gewicht 0 zu.

hritt 2: Berechne in $O(n)$ den Dualgraph $G^* = (V^*, E^*)$ zu der Triangulierung von G, wobei $w(e^*) := w(e)$ mit e^* Dualkante zu e.

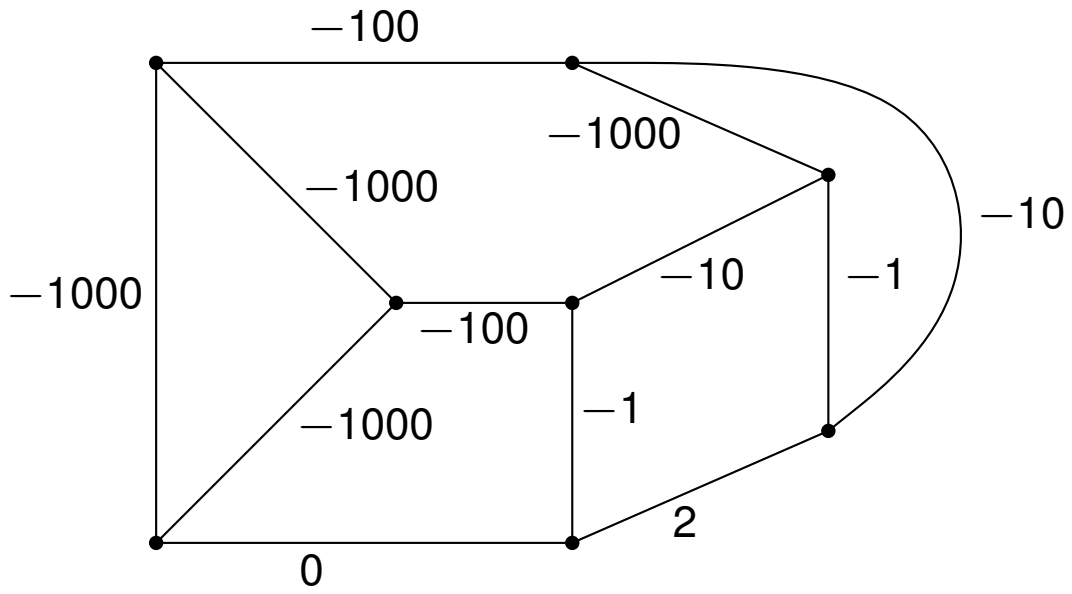
hritt 3: Konstruiere aus G^* in $O(n)$ einen Graph $G' = (V', E')$ derart, dass ein perfektes Matching minimalen Gewichts in G' eine gerade Menge maximalen Gewichts in G^* induziert.

hritt 4: Konstruiere in $O(n^{\frac{3}{2}} \log n)$ ein perfektes Matching M minimalen Gewichts in G'.

hritt 5: Falls M eine nichtleere gerade Menge in E^* induziert, gib den dazu dualen Schnitt in G aus. Ansonsten berechne in $O(n^{\frac{3}{2}} \log n)$ aus M eine nichttriviale gerade Menge in G^* maximalen Gewichts.

Aufgabe 3

Annahme: *G* ist 3-regulär und enthält keinen positiven Kreis.

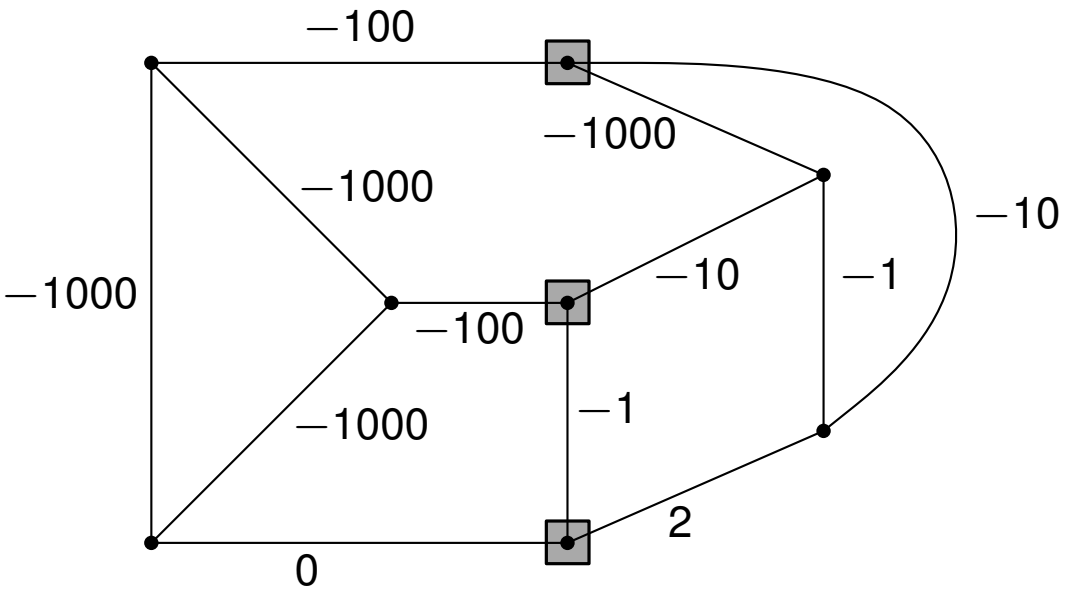


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Aufgabe 3

Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

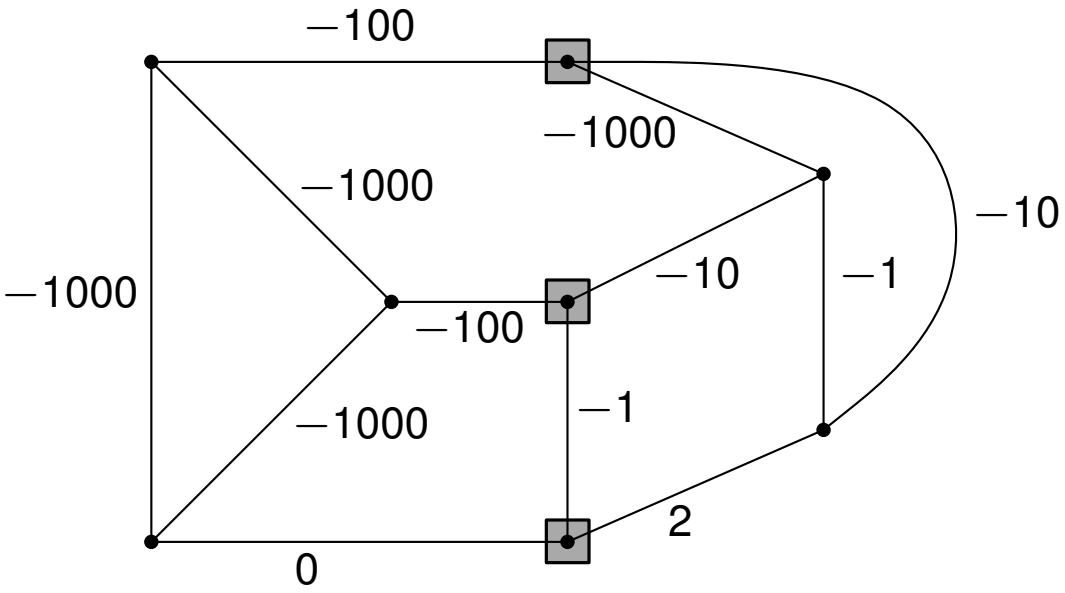
Teilschritt 1: Berechne eine Partition S, V_1, V_2 in G (Planar-Separator).



Aufgabe 3

Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

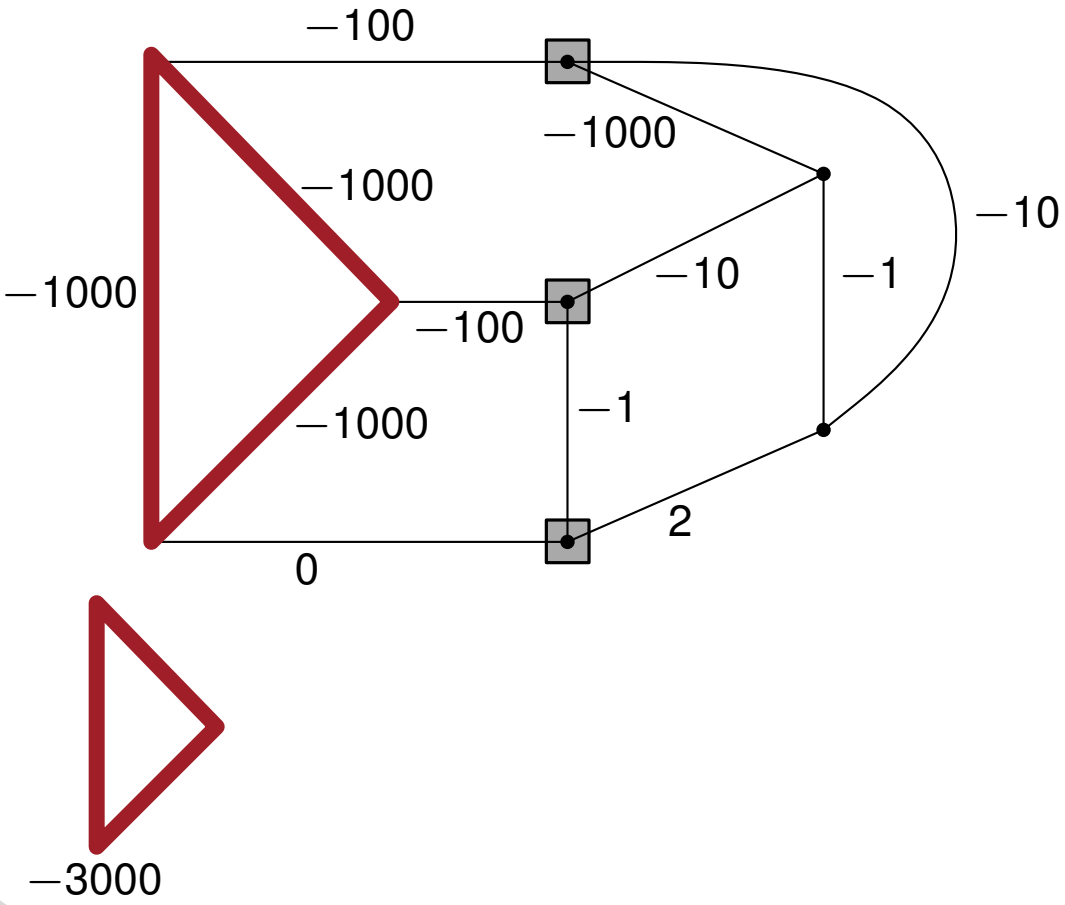
Teilschritt 2: Berechne rekursiv negative einfache Kreise maximalen Gewichts in den durch V_1 und V_2 induzierten Subgraphen von G.



Aufgabe 3

Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 2: Berechne rekursiv negative einfache Kreise maximalen Gewichts in den durch V_1 und V_2 induzierten Subgraphen von G.



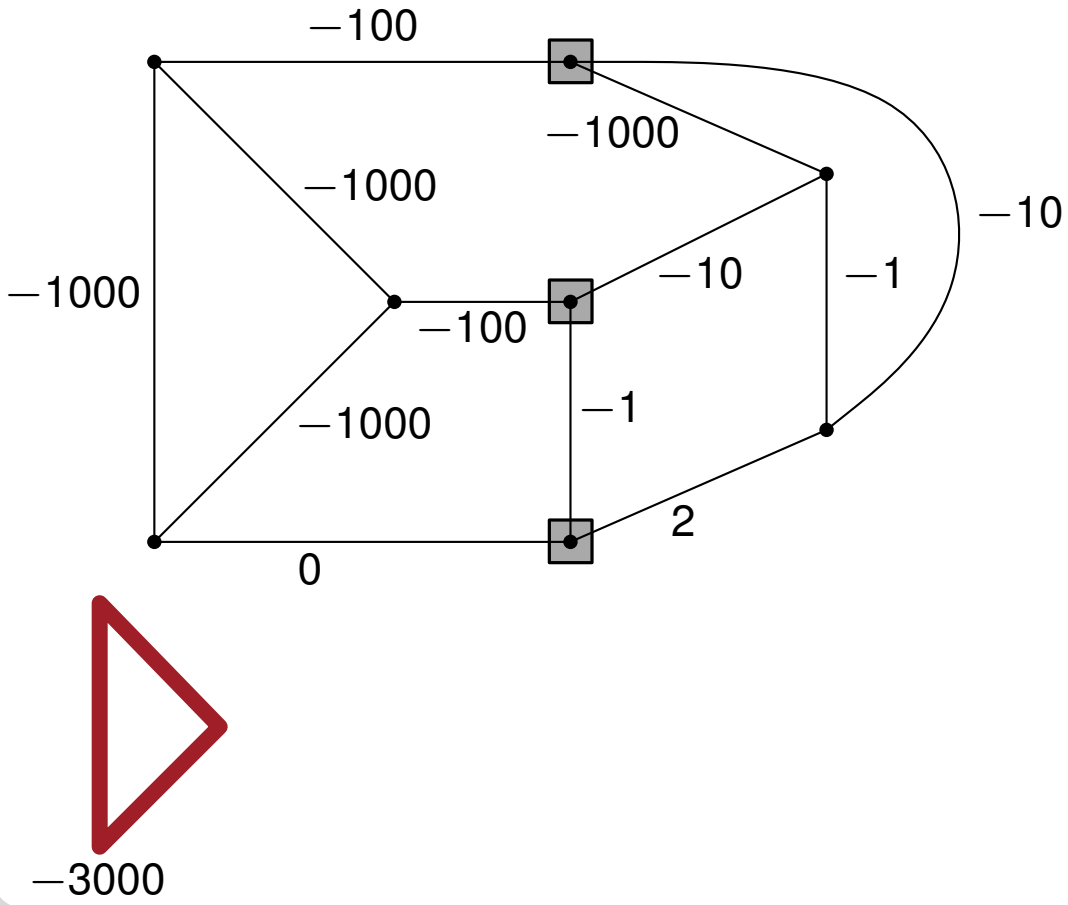
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

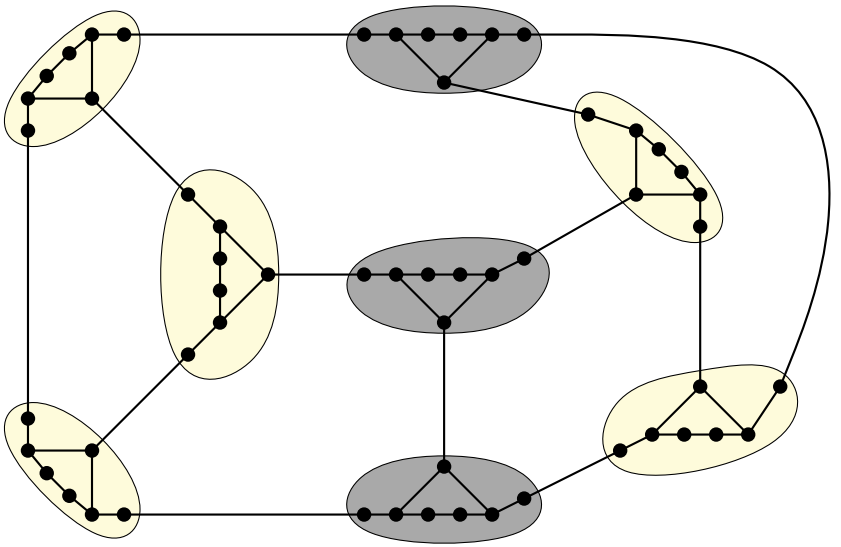
Aufgabe 3

Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Konstruiere Graphen G' gemäß Schritt 3 des Mixed-Max-Cut-Algorithmus.





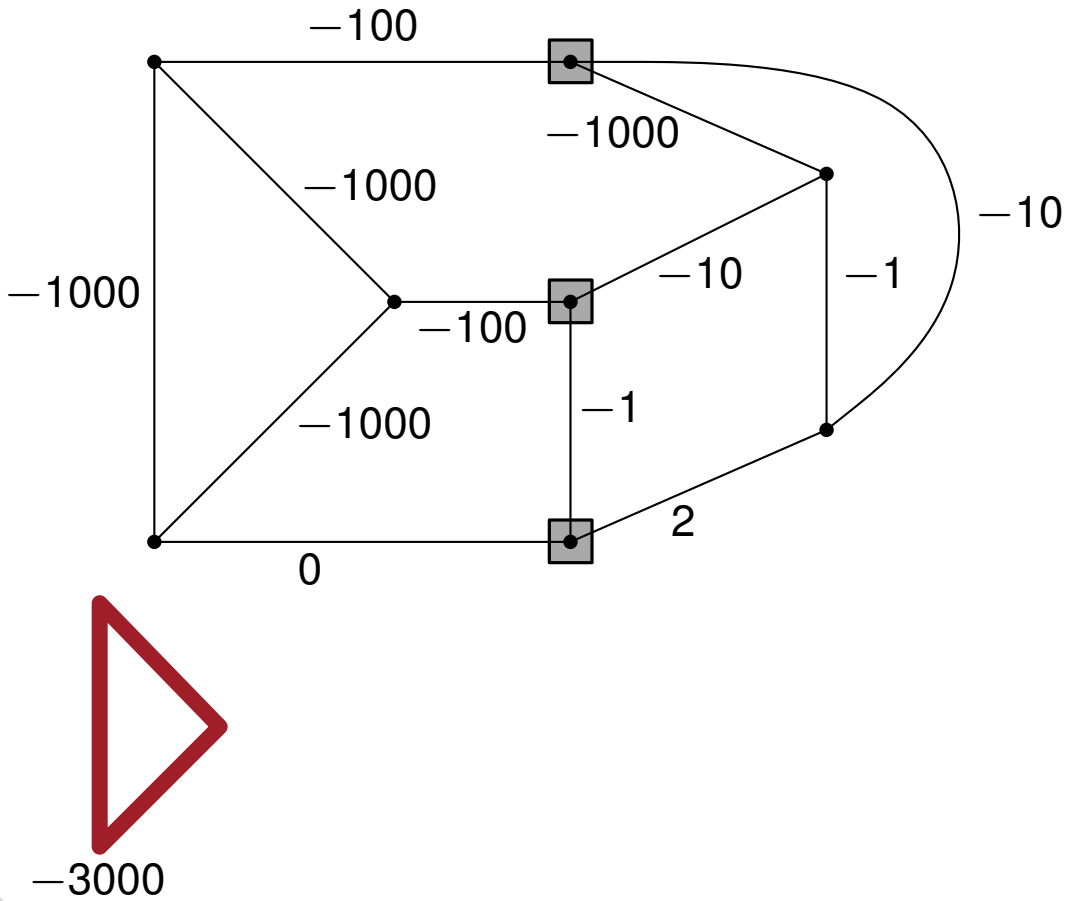
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Aufgabe 3

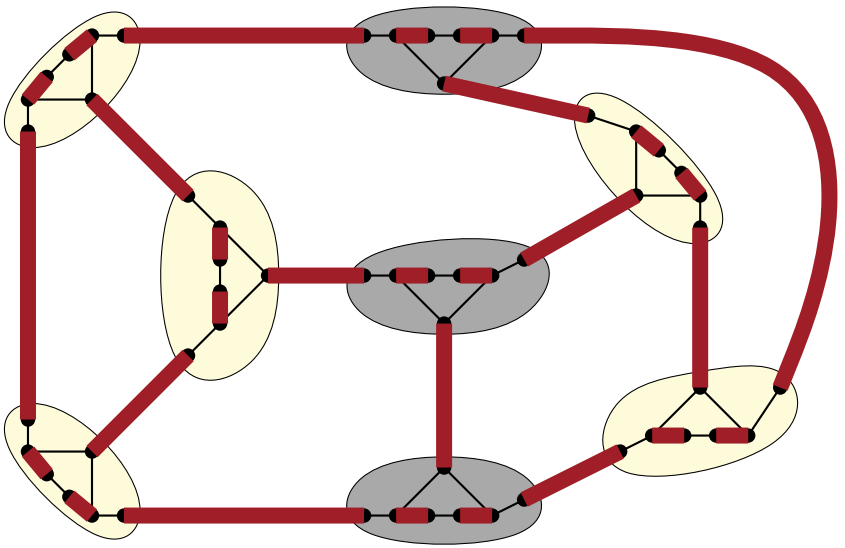
Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Konstruiere Graphen G' gemäß Schritt 3 des Mixed-Max-Cut-Algorithmus.



Perfektes Matching minimalen Gewichts aus Schritt 4 bekannt.

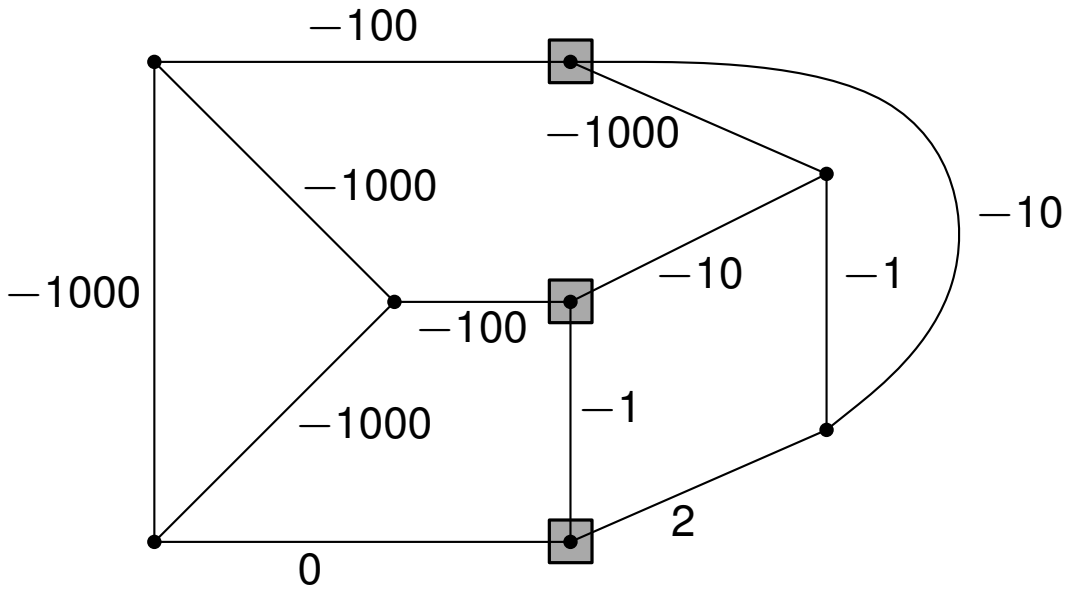


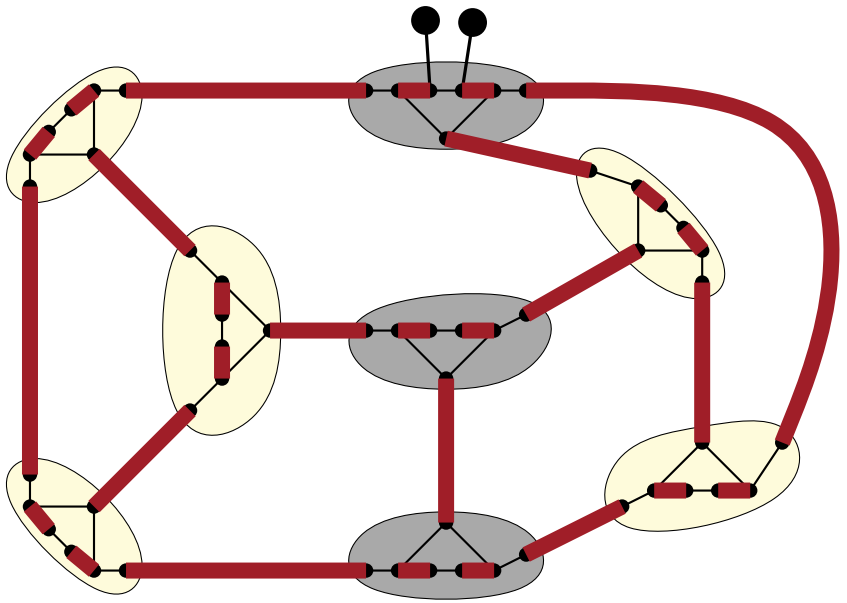
Aufgabe 3

Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Erweitere G' um die Knoten w und w'





Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

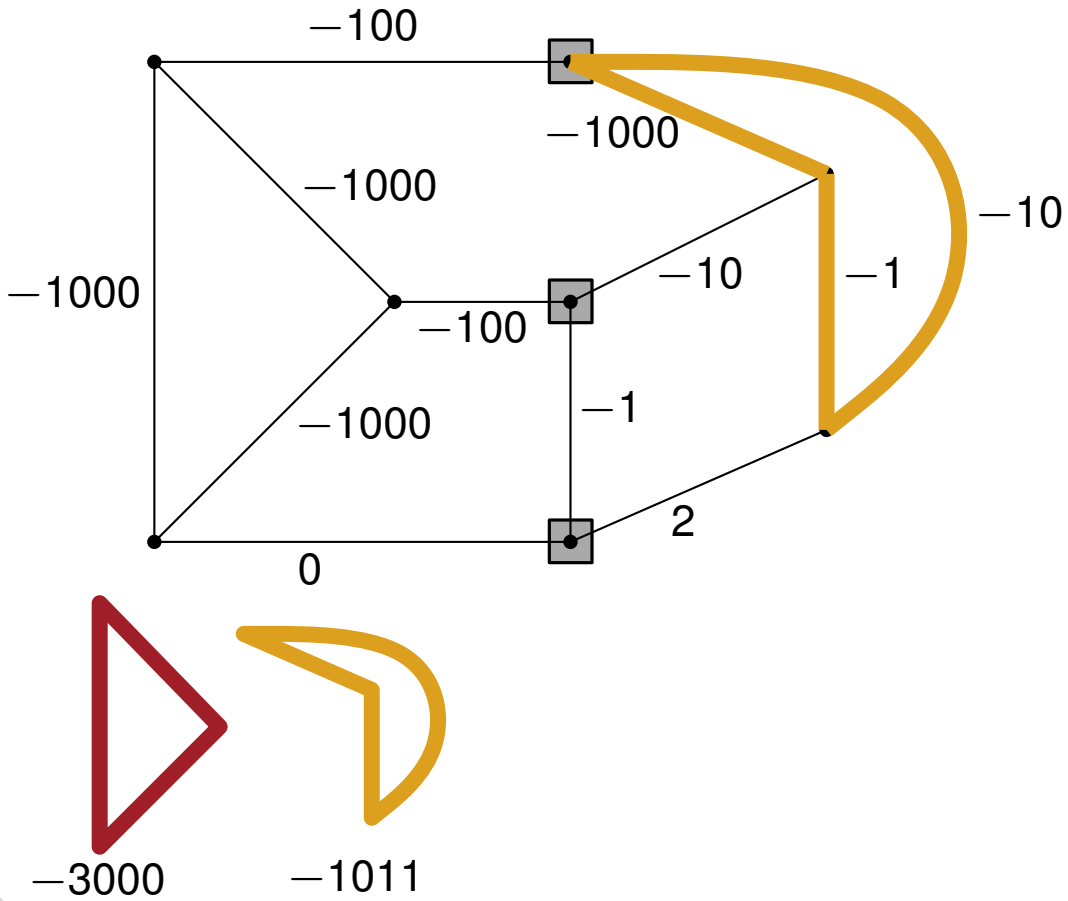
Aufgabe 3

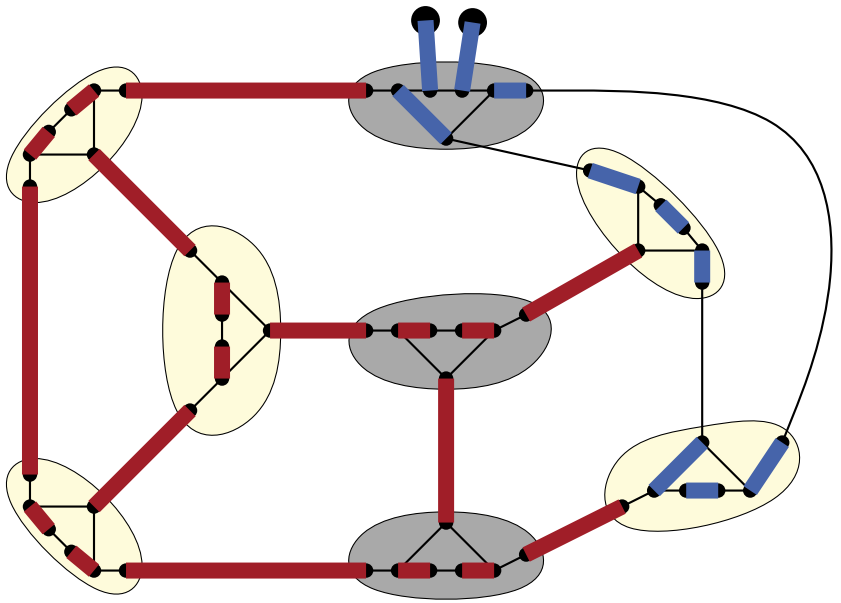
Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Berechne perfektes Matching: Finde erhöhende Wege von w und w'

$\rightarrow$ induziert Kreis.





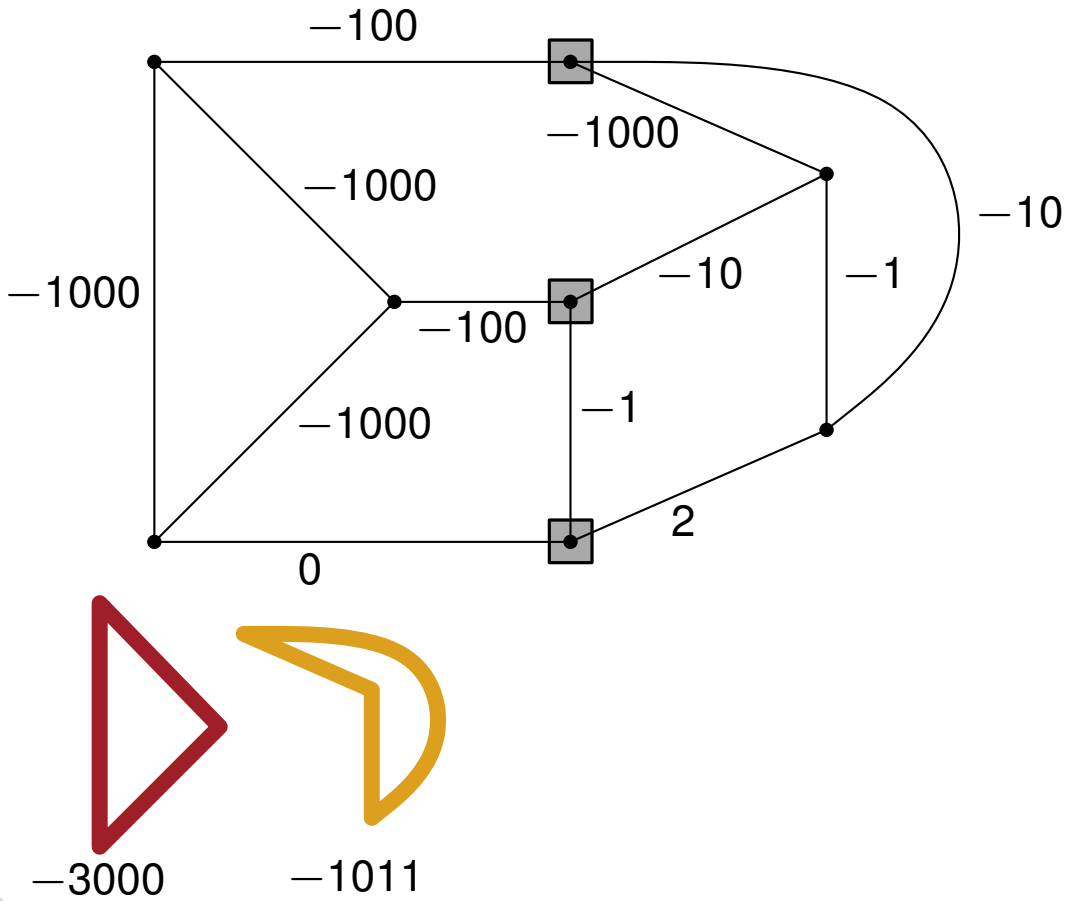
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

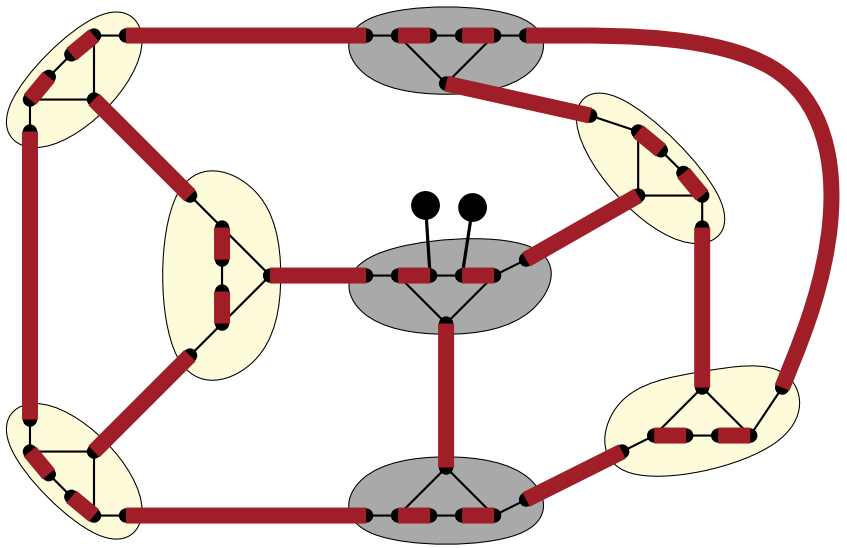
Aufgabe 3

Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Erweitere G' um die Knoten w und w'





Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

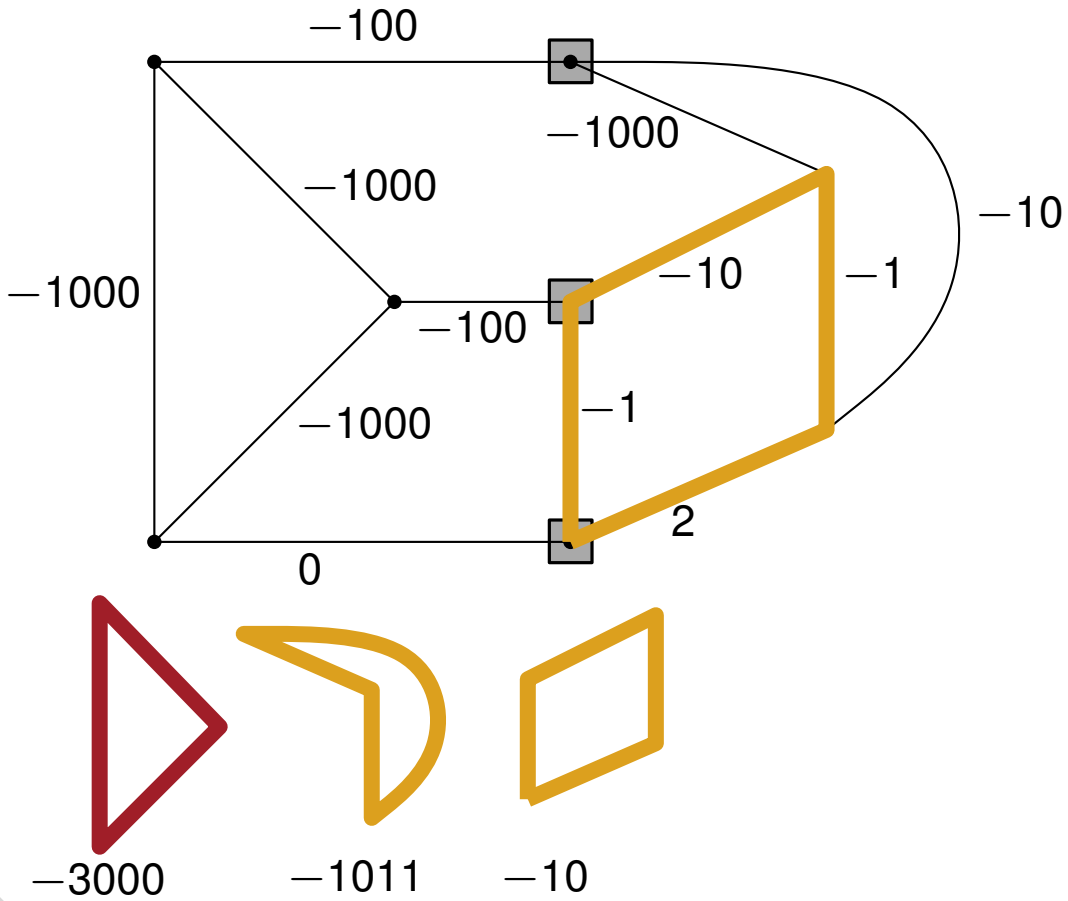
Aufgabe 3

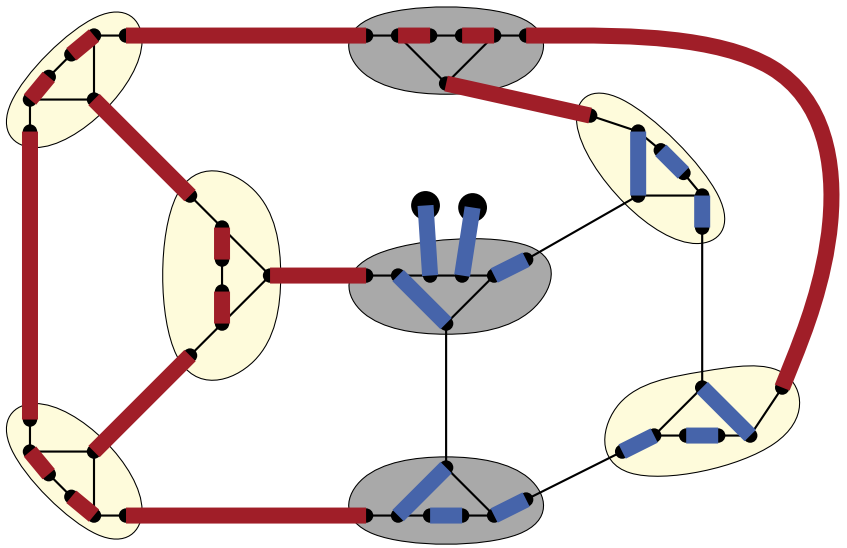
Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Berechne perfektes Matching: Finde erhöhende Wege von w und w'

$\rightarrow$ induziert Kreis.





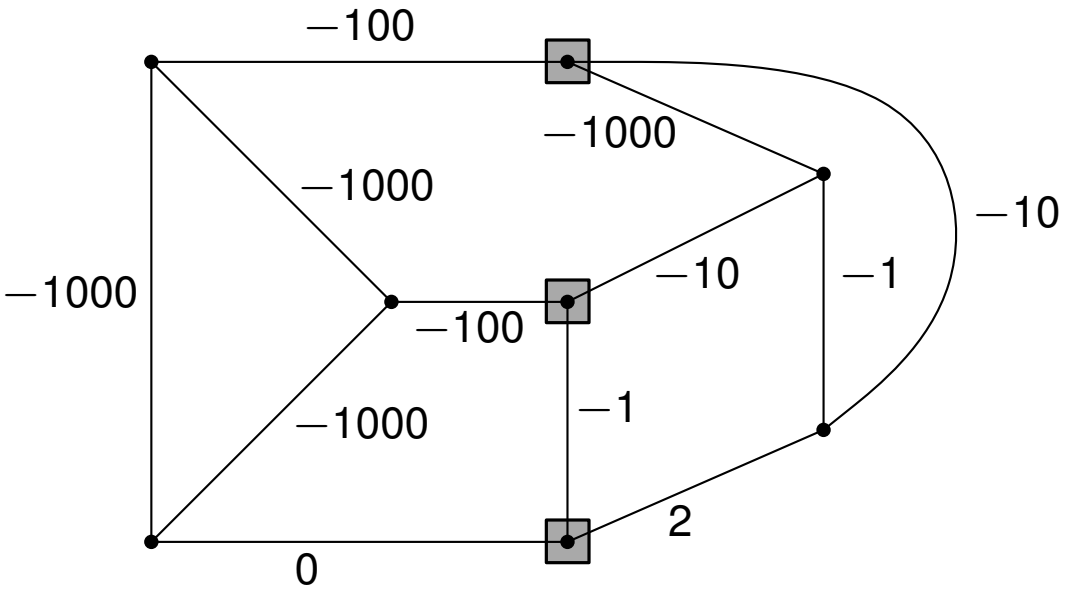
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

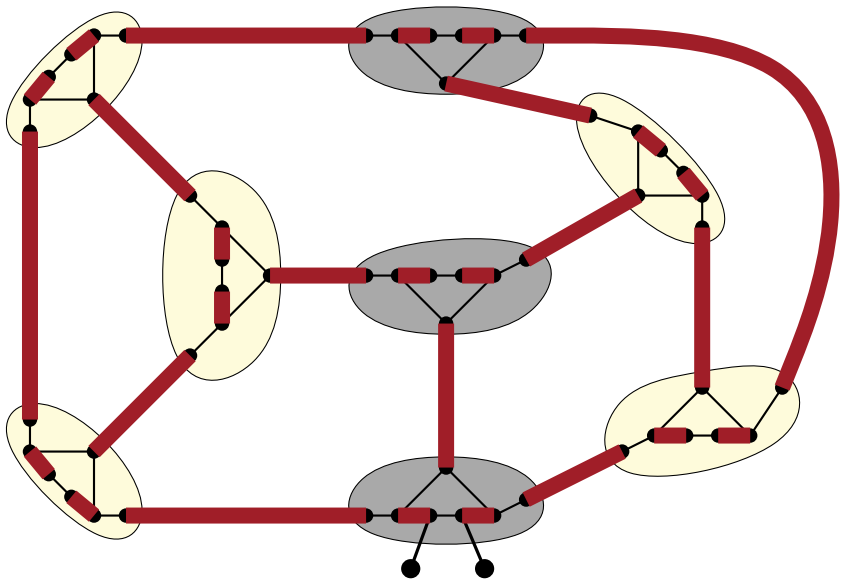
Aufgabe 3

Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Erweitere G' um die Knoten w und w'





Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

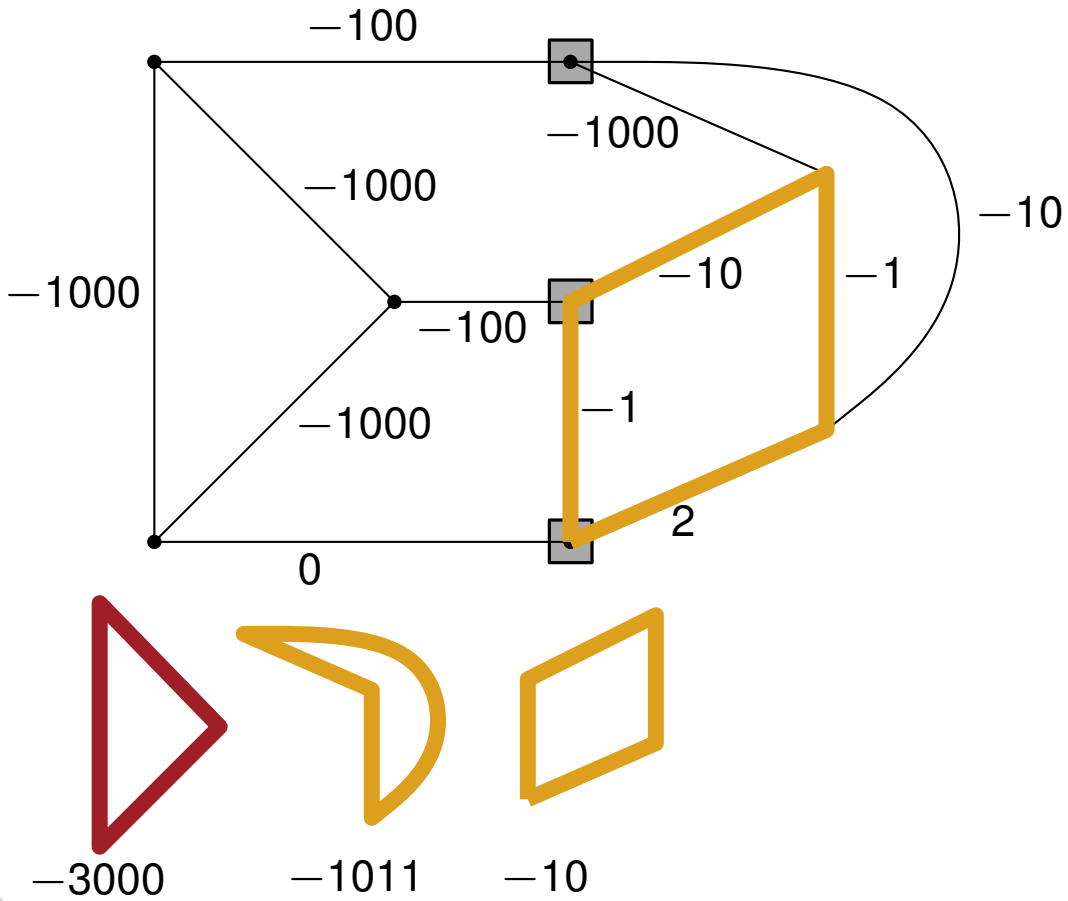
Aufgabe 3

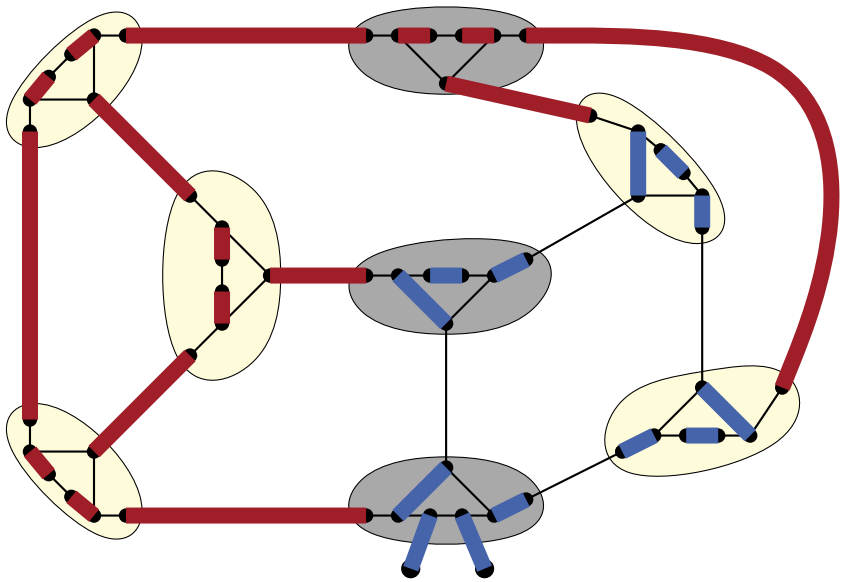
Annahme: G ist 3-regulär und enthält keinen positiven Kreis.

Teilschritt 3: Für jedes $v_i \in S$ berechne den negativen einfachen Kreis maximalen Gewichts in G, der v_i enthält:

Berechne perfektes Matching: Finde erhöhende Wege von w und w'

$\rightarrow$ induziert Kreis.





Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Aufgabe 1

$F =$	Menge der Facetten
$f_0 =$	äußere Facette von G.
$\text{dist}(f) =$	Länge eines kürzesten Weges vom der Facette f entsprechenden Dualknoten zum f_0 entsprechenden Dualknoten.
$l :=$	$\max_{f \in F} \text{dist}(f)$.

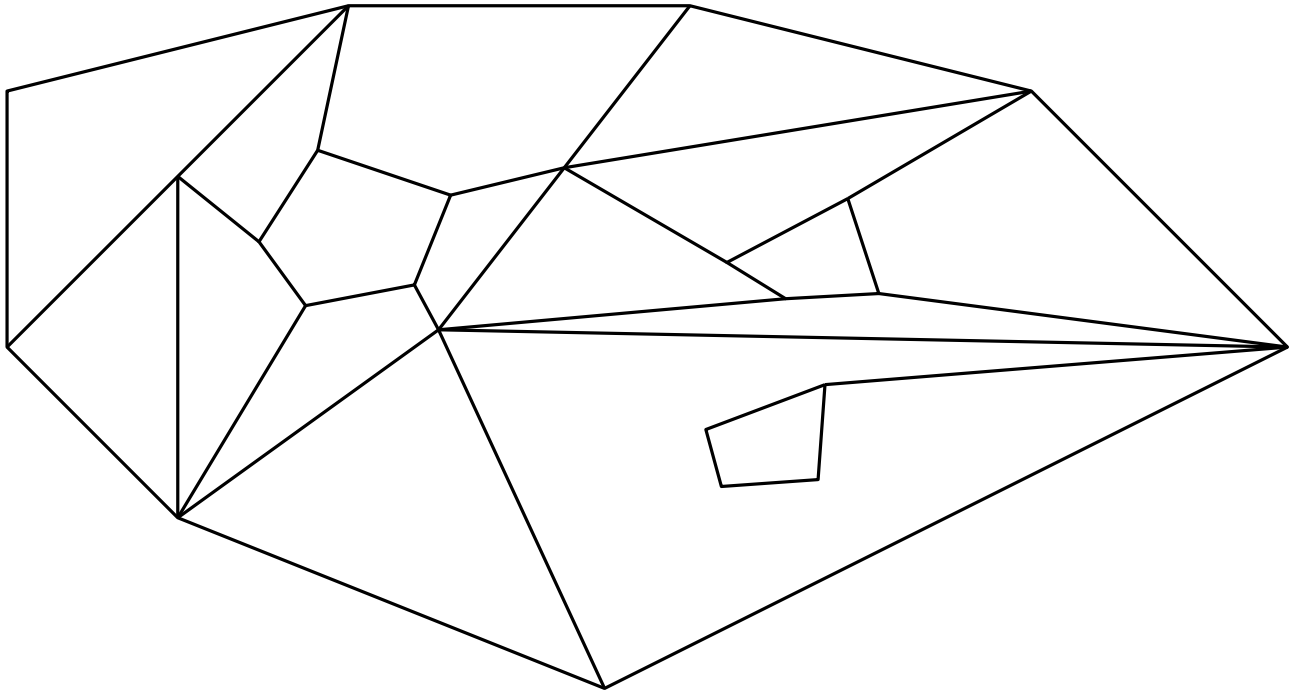
Aufgabe 1

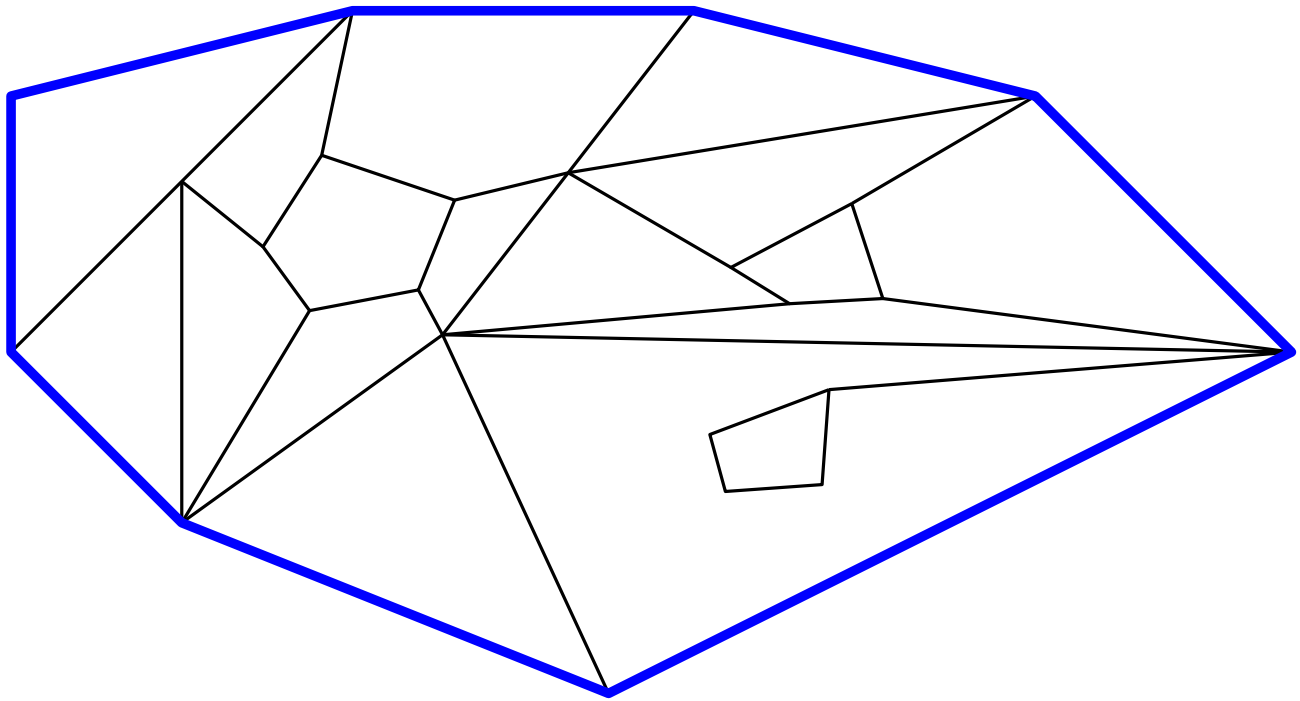
$F =$	Menge der Facetten
$f_0 =$	äußere Facette von G.
$\text{dist}(f) =$	Länge eines kürzesten Weges vom der Facette f entsprechenden Dualknoten zum f_0 entsprechenden Dualknoten.
$l :=$	$\max_{f \in F} \text{dist}(f)$.

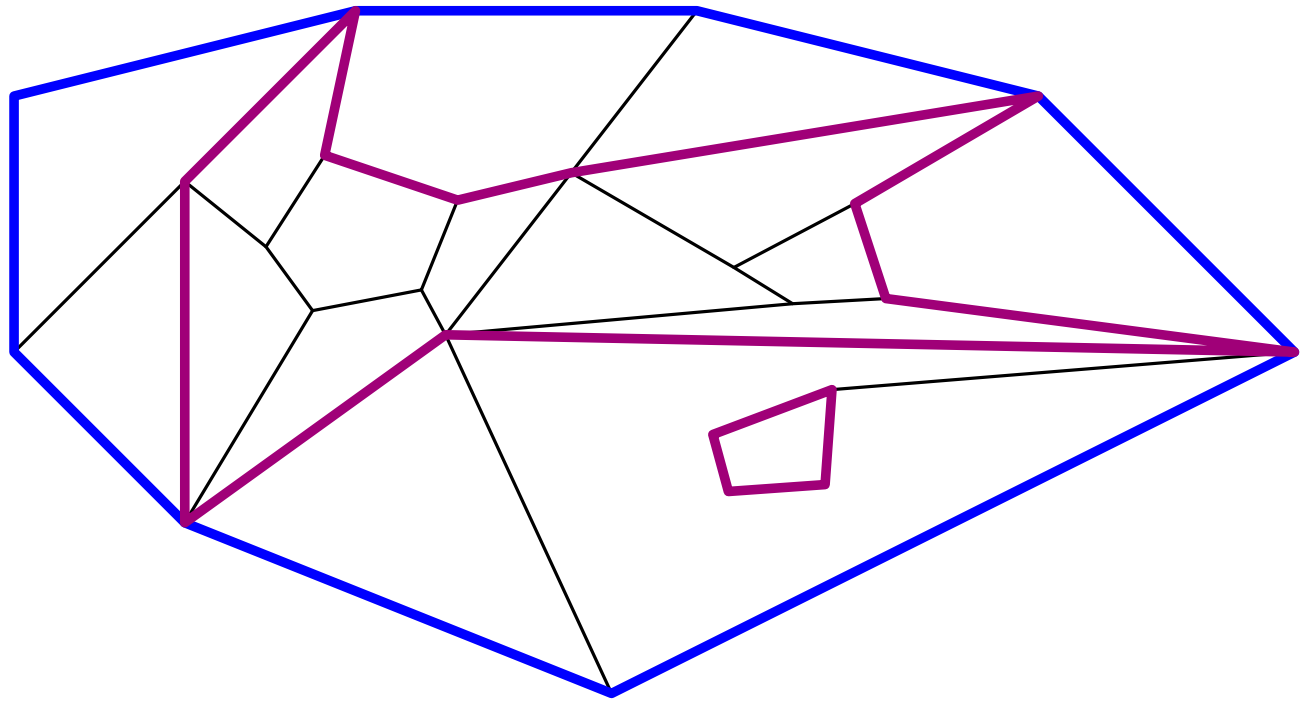
In Schritt 2 des Algorithmus für das kantendisjunkte Menger-Problem werden in einem planaren Graphen G mit fester Einbettung einfache Kreise $C_1, \ldots, C_l$ wie folgt konstruiert:

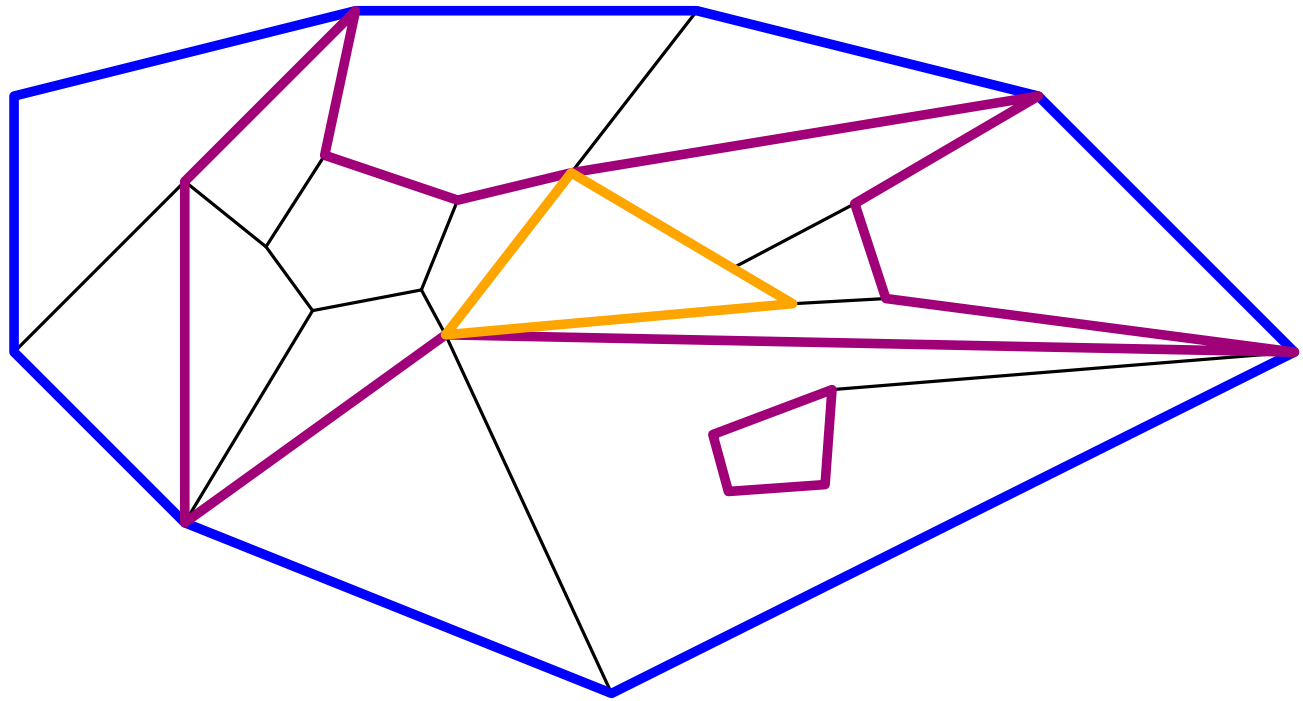
Für $1 \leq i \leq l$ sei C_i die Vereinigung der einfachen Kreise in G so, dass $\text{dist}(f) \geq i$ für alle Facetten f im Inneren und $\text{dist}(f) < i$ für alle Facetten f im Äußeren eines Kreises aus C_i gilt.

Aufgabe: Geben Sie einen Algorithmus mit linearer Laufzeit an, der zu einem gegebenen Graphen G mit fester Einbettung die Kantenmengen $C_1, \ldots, C_l$ bestimmt.

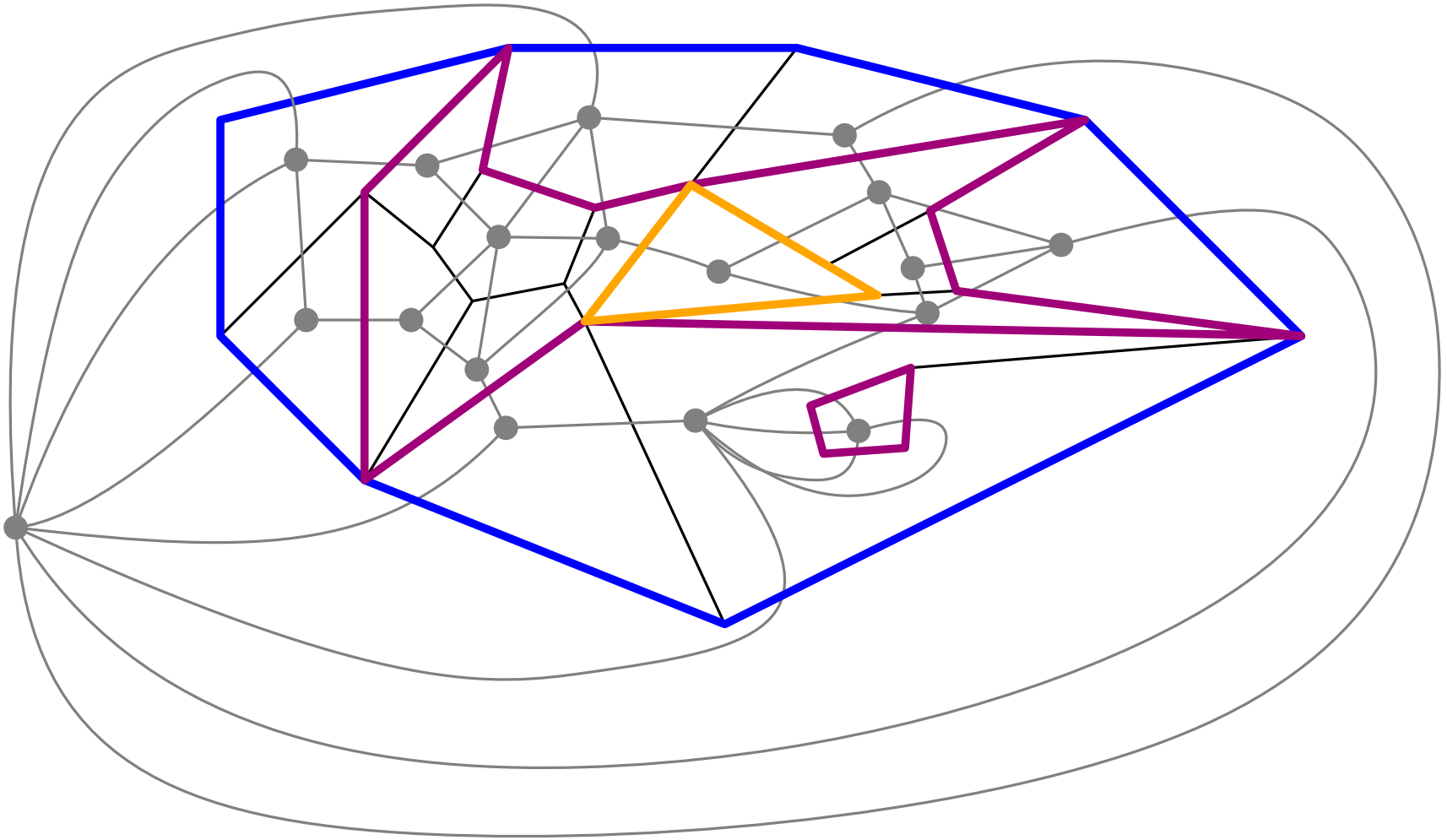


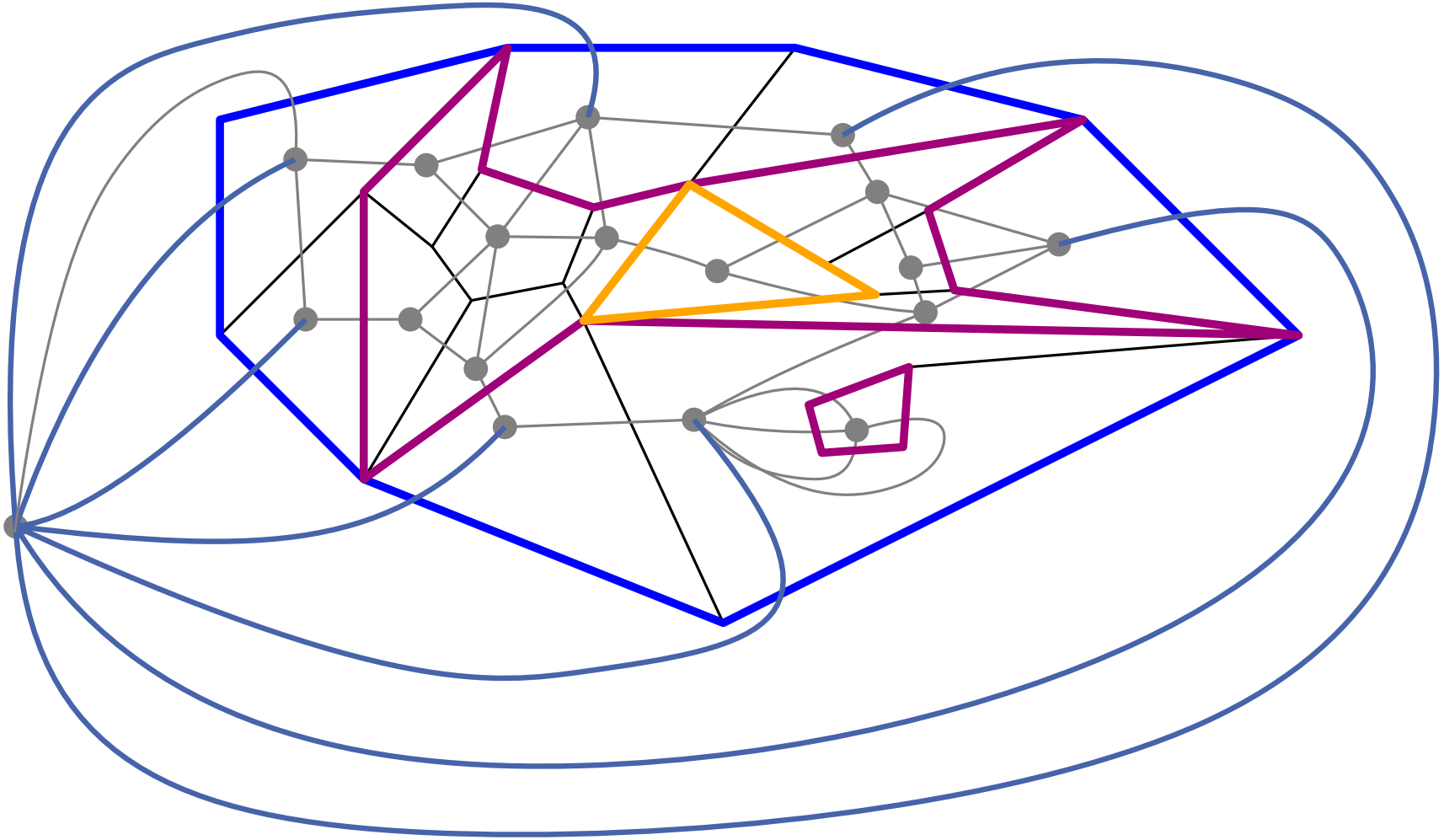


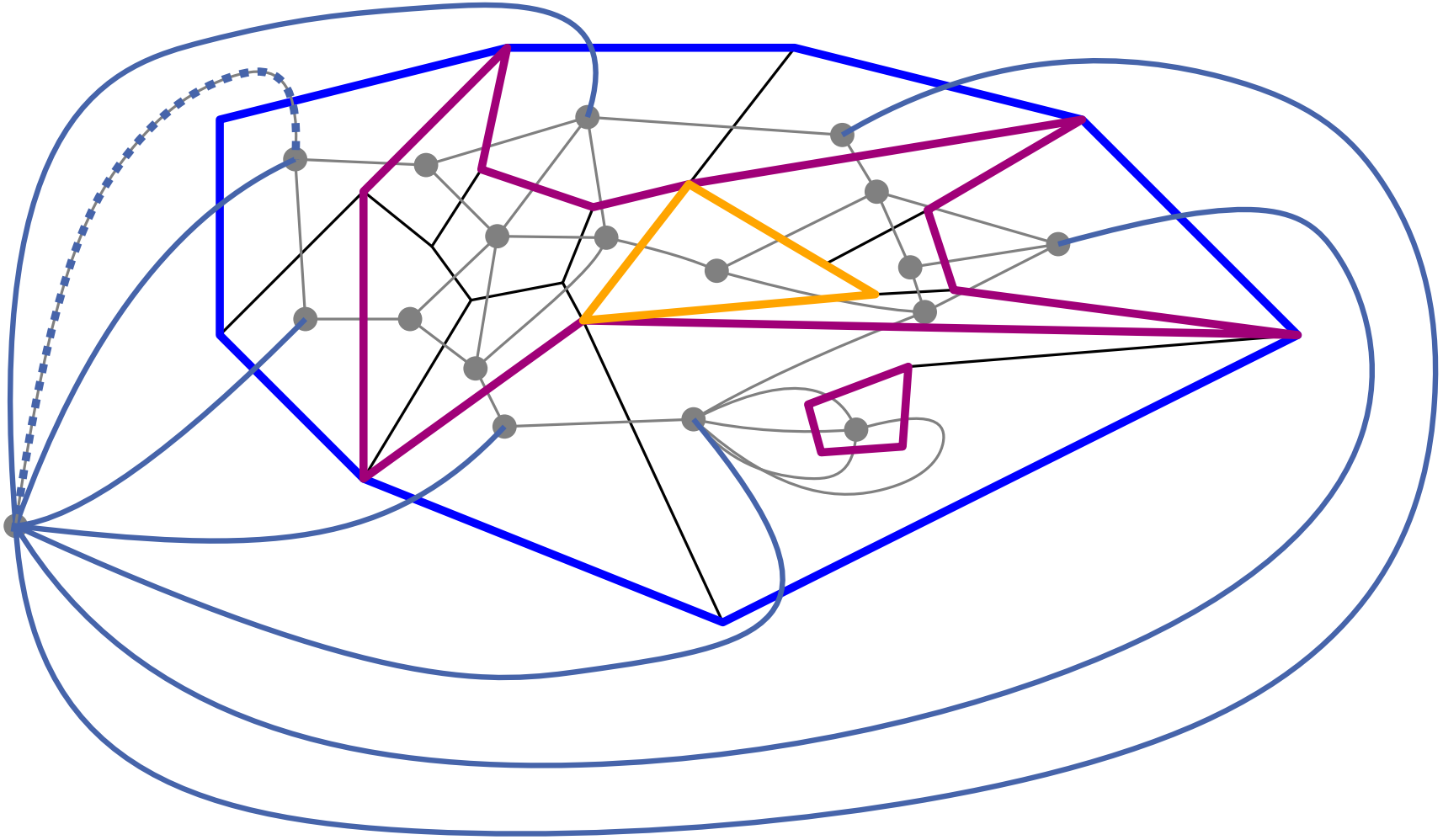


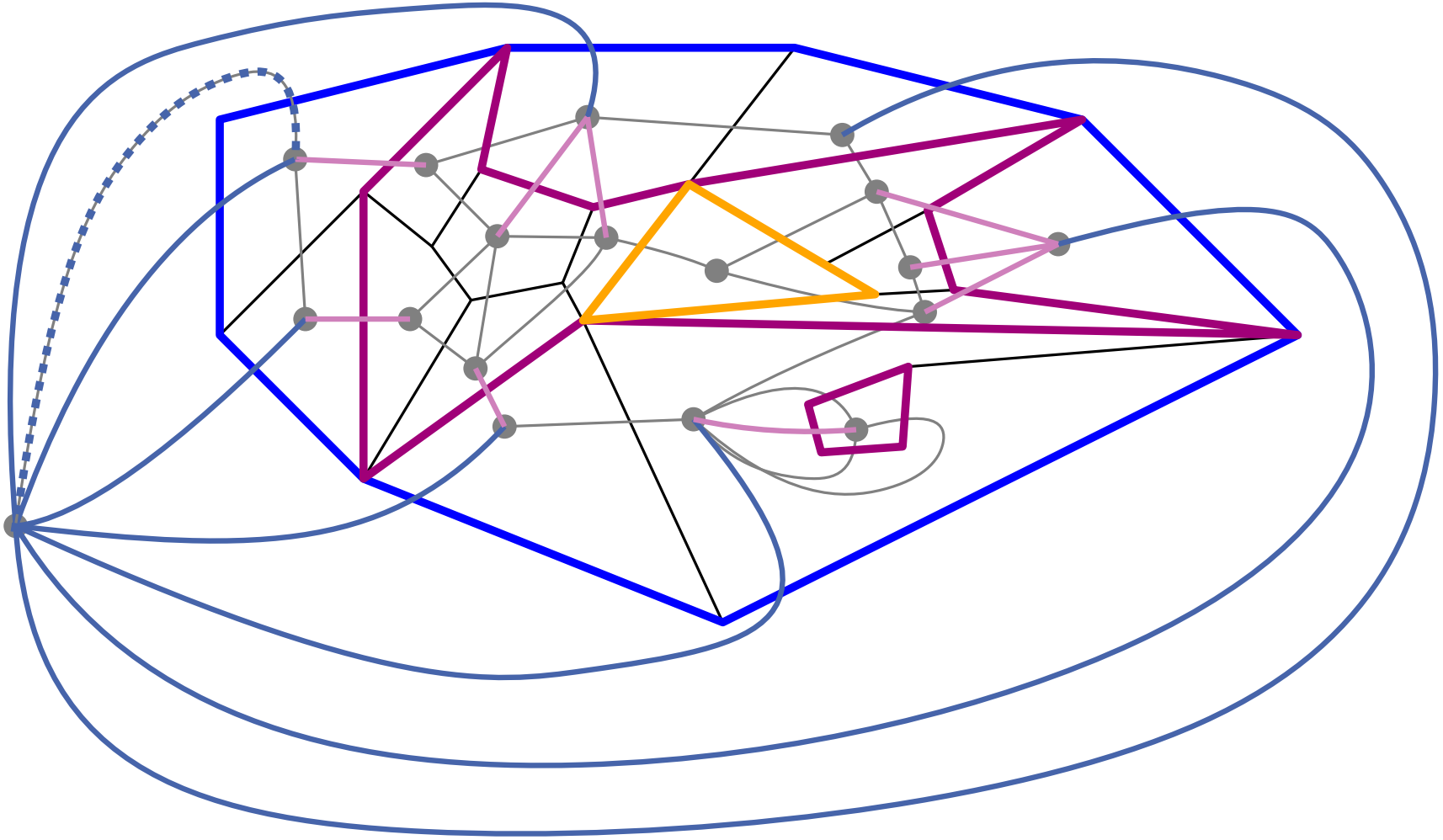


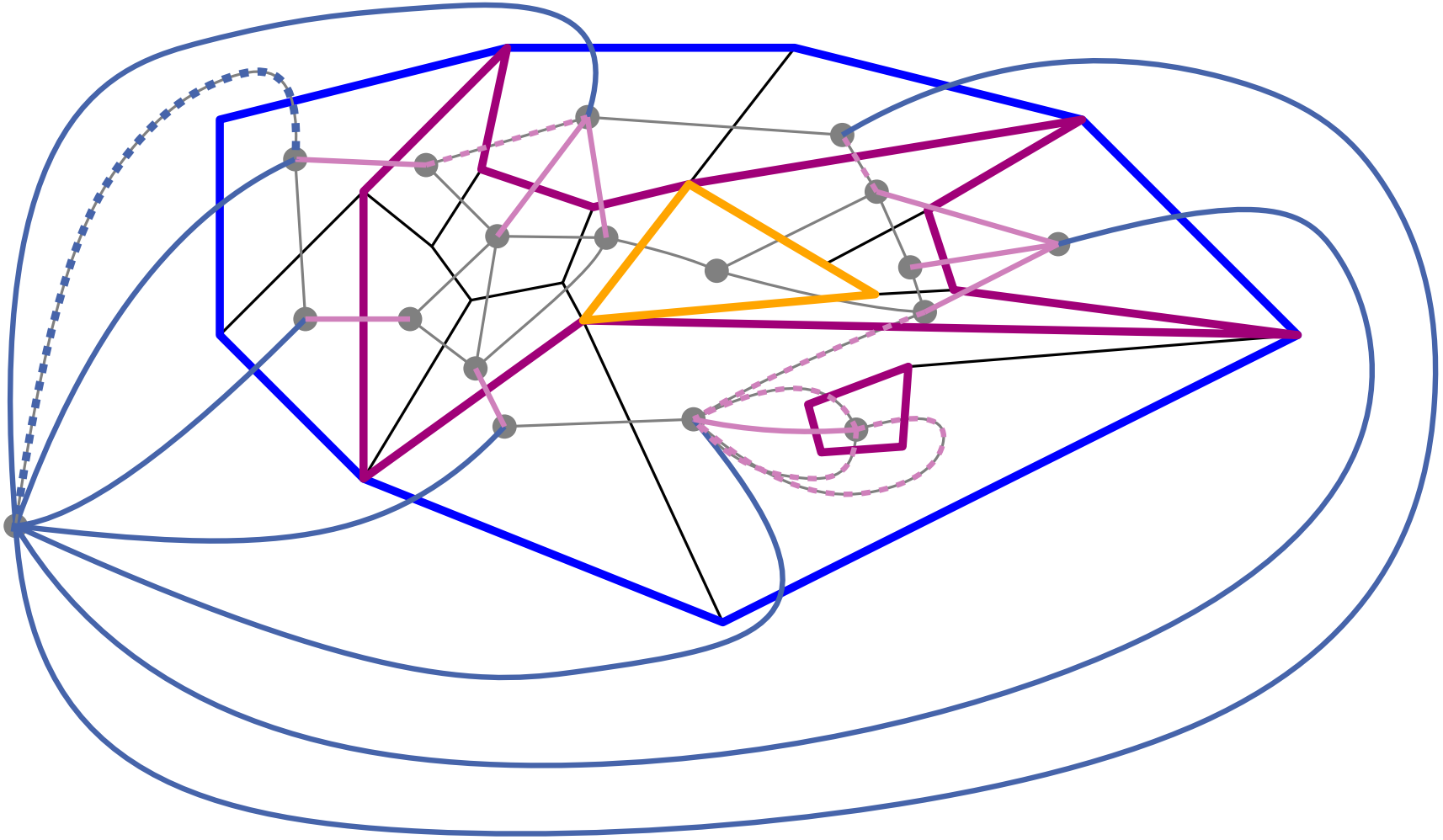
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

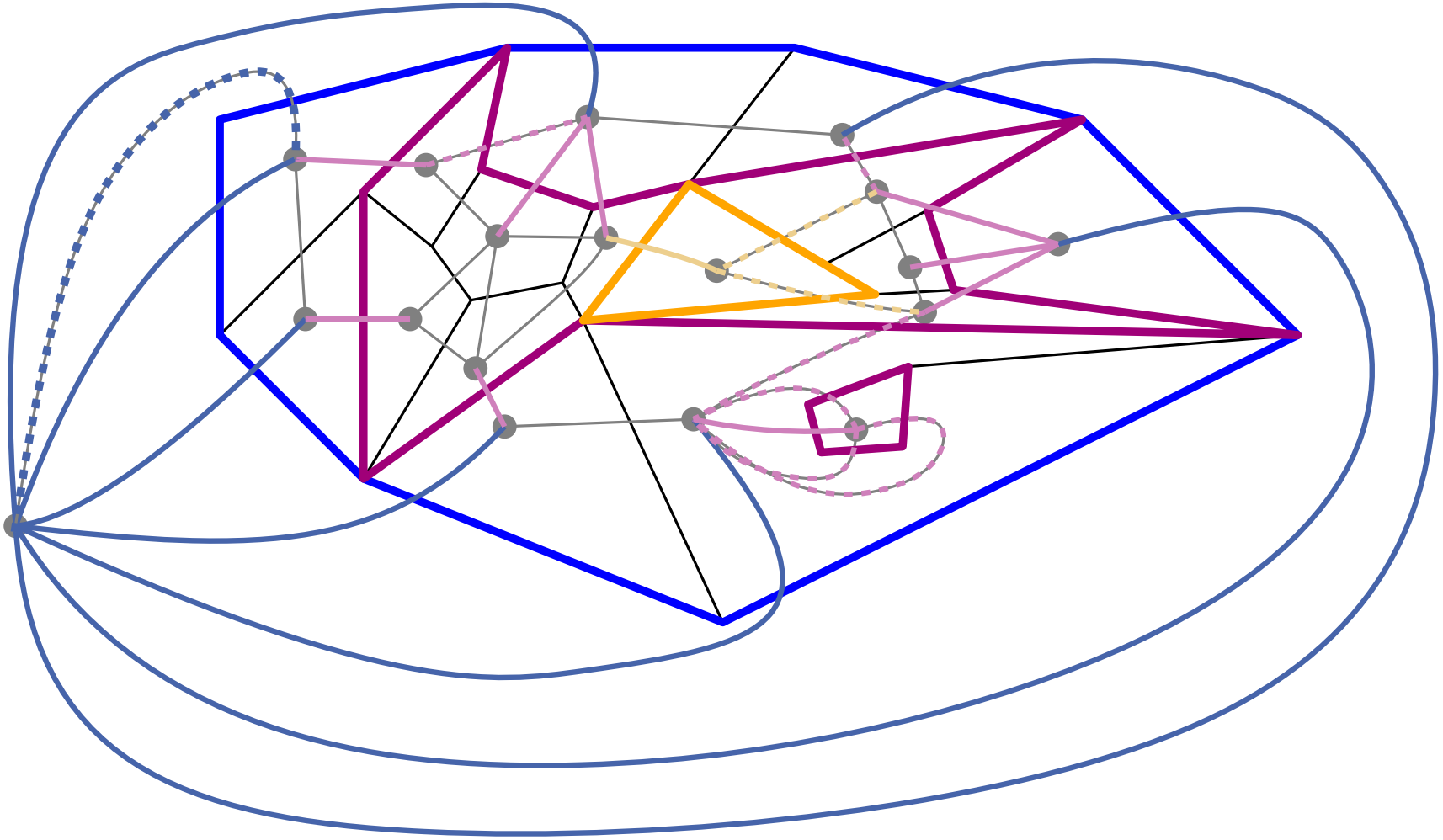












Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Aufgabe 2

Geben Sie einen Algorithmus an, der folgendes Problem in Linearzeit löst:

Gegeben ein planarer Graph G mit fester Einbettung und ausgezeichneten Knoten s und t, die an der äußeren Facette liegen, bestimme eine maximale Anzahl von paarweise kantendisjunkten s-t-Wegen in G.