Karlsruher Institut für Technologie

Algorithmen für Planare Graphen

9. Juni 2020, Übung 3

Lars Gottesbüren

INSTITUT FÜR THEORETISCHE INFORMATIK

1 – Triangulierung

Triangulierung in $\mathcal{O}(n)$

1. Füge Kanten hinzu so, dass es keine Grad-1 Knoten mehr gibt.
2. Trianguliere Graph ohne auf Schleifen/Multikanten zu achten.
3. Löse Schleifen und Multikanten durch Kantentausch auf.

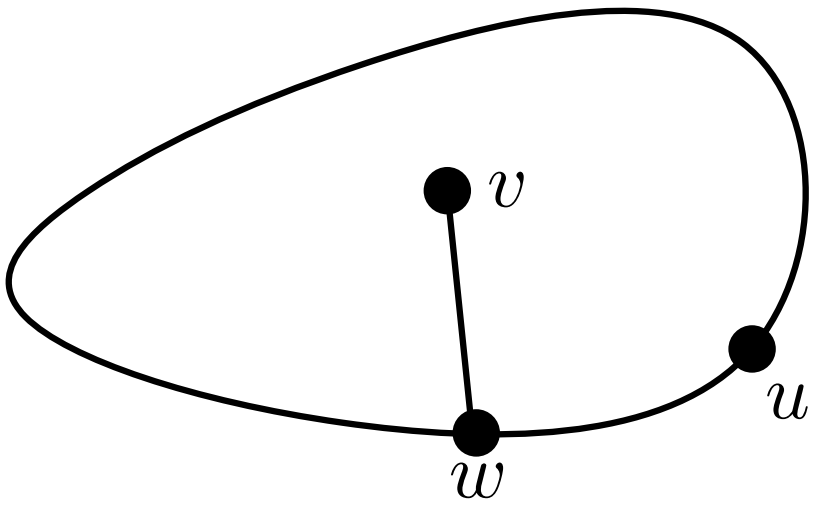
1 – Triangulierung

Triangulierung in $\mathcal{O}(n)$

1. Füge Kanten hinzu so, dass es keine Grad-1 Knoten mehr gibt.
2. Trianguliere Graph ohne auf Schleifen/Multikanten zu achten.
3. Löse Schleifen und Multikanten durch Kantentausch auf.

1.1 – Triangulierung

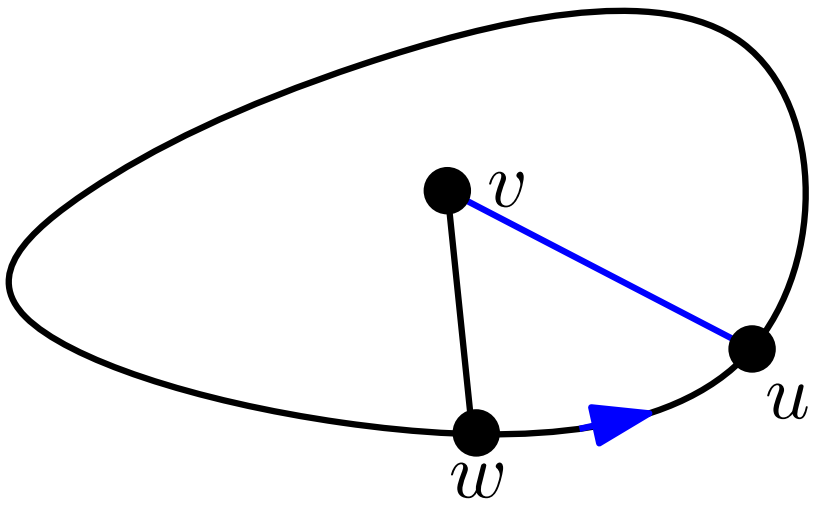
1. Füge Kanten hinzu so, dass es keine Grad-1-Knoten mehr gibt.



- Sei v Knoten mit Grad 1.
- Seine Kante sei $\{v, w\}$ und f die Facette in der er liegt.
- Laufe f von w aus im Gegenuhrzeigersinn ab.
- Verbinde v mit dem zweiten besuchten Knoten u.

1.1 – Triangulierung

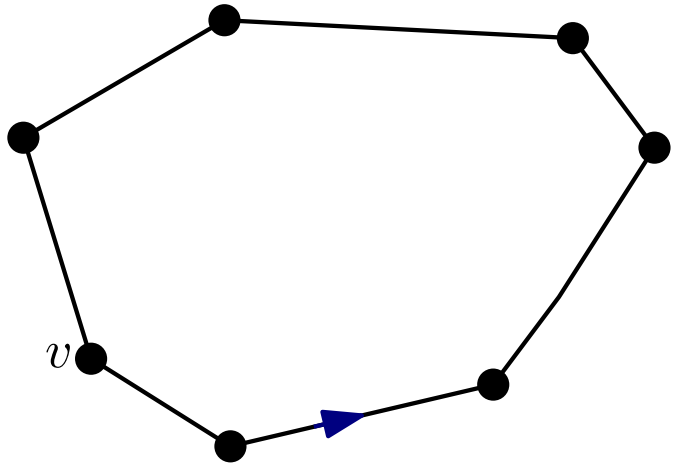
1. Füge Kanten hinzu so, dass es keine Grad-1-Knoten mehr gibt.



- Sei v Knoten mit Grad 1.
- Seine Kante sei $\{v, w\}$ und f die Facette in der er liegt.
- Laufe f von w aus im Gegenuhrzeigersinn ab.
- Verbinde v mit dem zweiten besuchten Knoten u.

1.2 – Triangulierung

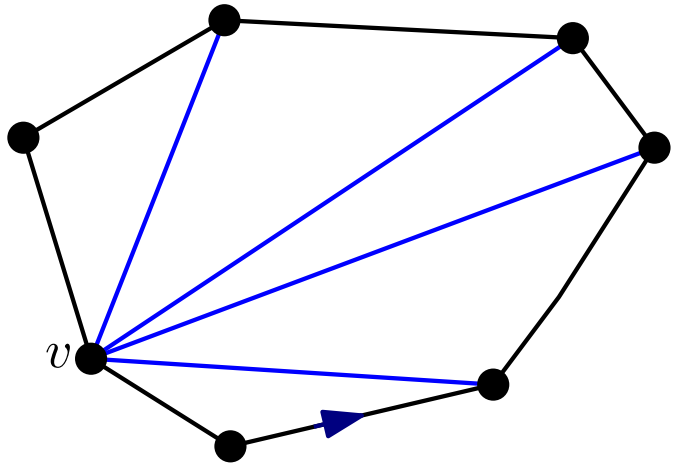
2. Trianguliere Graph ohne auf Schleifen/Multikanten zu achten.



- Für jede Facette f wähle beliebigen Knoten v.
- Laufe f ab und verbinde v mit allen besuchten Knoten, außer dem Vorgänger und Nachfolger von v auf f.

1.2 – Triangulierung

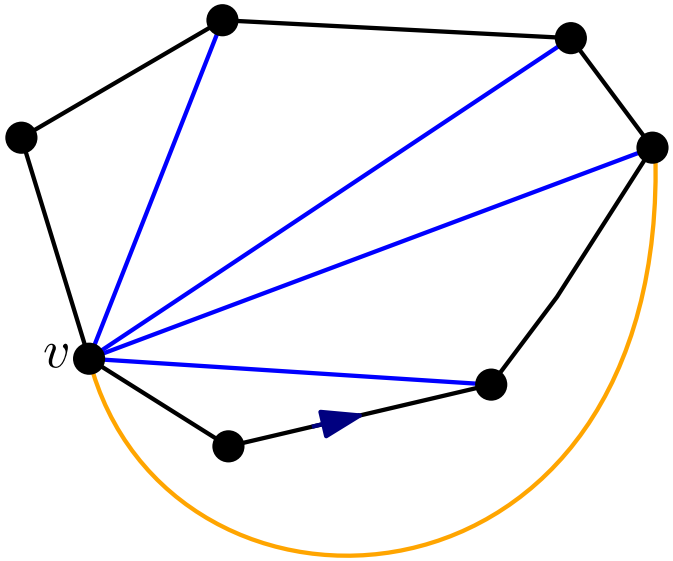
2. Trianguliere Graph ohne auf Schleifen/Multikanten zu achten.



- Für jede Facette f wähle beliebigen Knoten v.
- Laufe f ab und verbinde v mit allen besuchten Knoten, außer dem Vorgänger und Nachfolger von v auf f.

1.2 – Triangulierung

2. Trianguliere Graph ohne auf Schleifen/Multikanten zu achten.

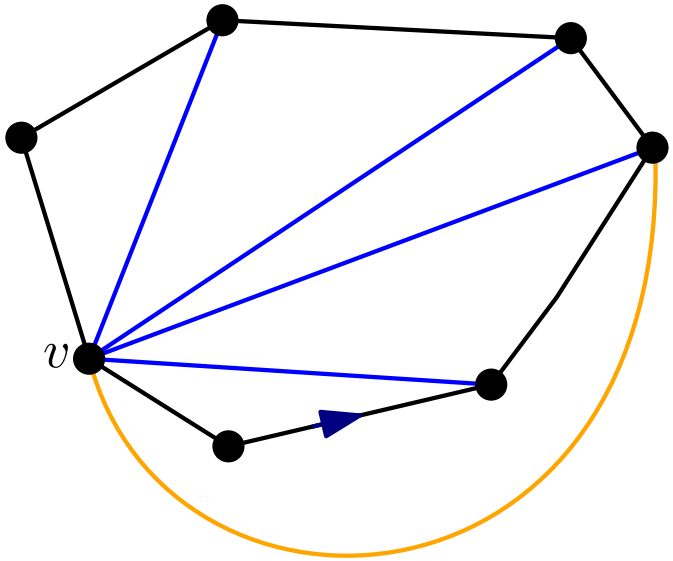


Doppelkanten

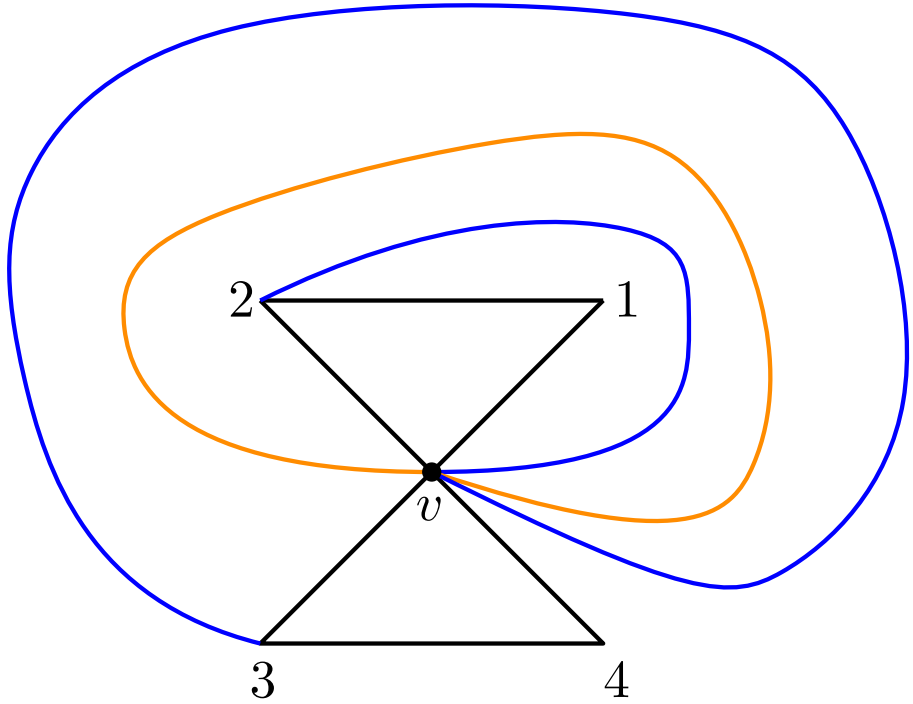
- Für jede Facette f wähle beliebigen Knoten v.
- Laufe f ab und verbinde v mit allen besuchten Knoten, außer dem Vorgänger und Nachfolger von v auf f.

1.2 – Triangulierung

2. Trianguliere Graph ohne auf Schleifen/Multikanten zu achten.



Doppelkanten



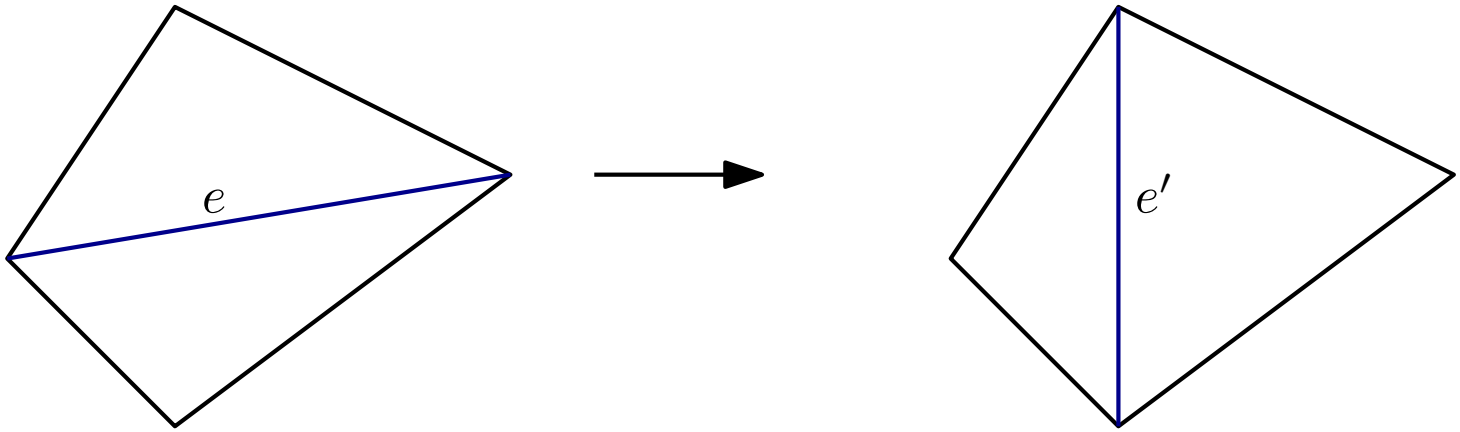
Schleifen

1.3 – Triangulierung

3. Löse Schleifen und Multikanten durch Kantentausch auf.

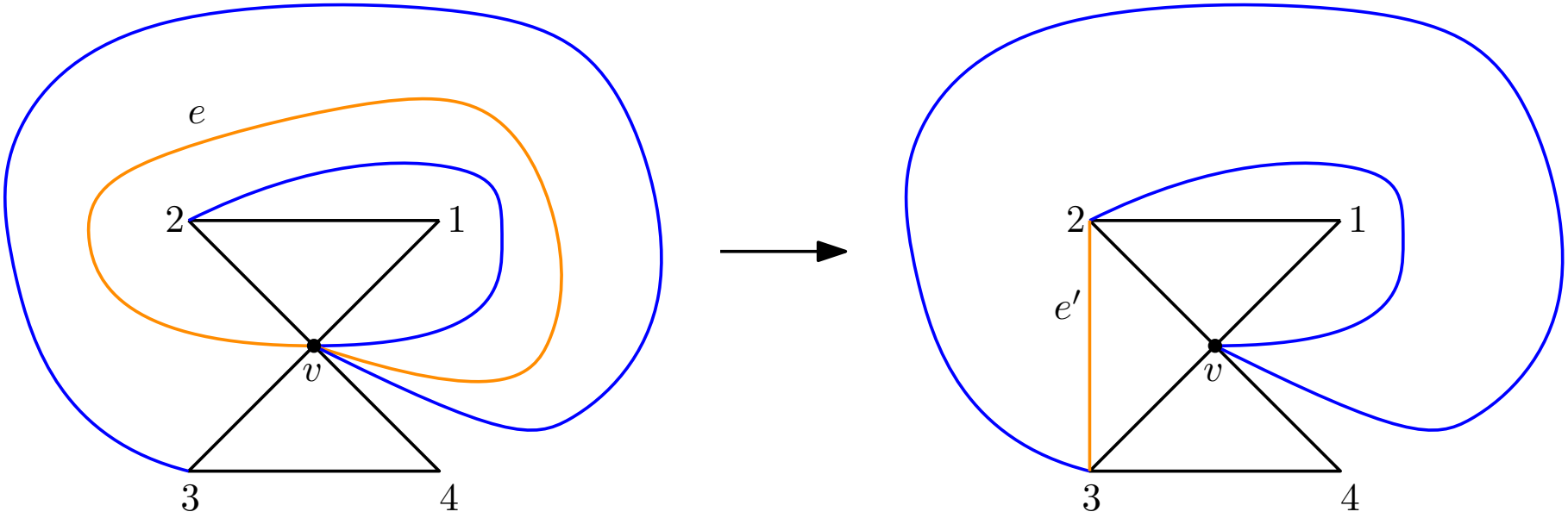
Kantentausch

- Betrachte Kante e eines triangulierten Graphen.
- Das Entfernen von e ergibt eine Facette f mit Grad 4.
- Füge Kante e' in f ein, die nicht die gleichen Knoten wie e verbindet.



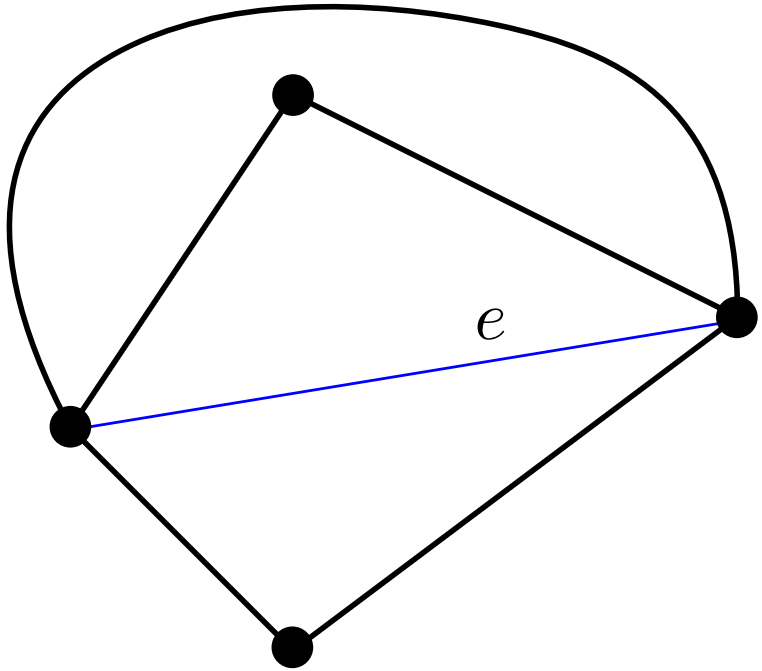
1.3 – Triangulierung

3. Löse **Schleifen** und Multikanten durch Kantentausch auf.



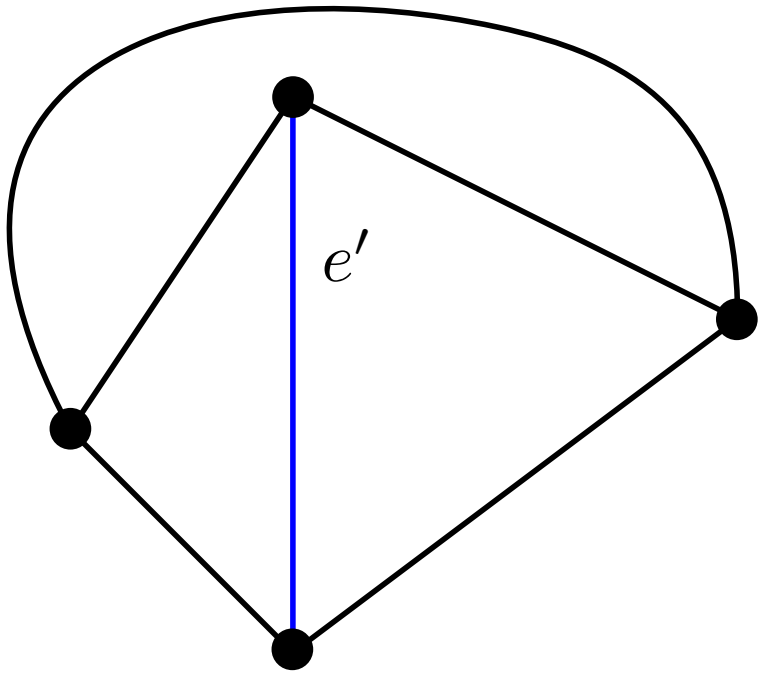
1.3 – Triangulierung

3. Löse Schleifen und **Multikanten** durch Kantentausch auf.
   - Sei e eine Multikanten.
   - Kann e' eine Multikante sein?



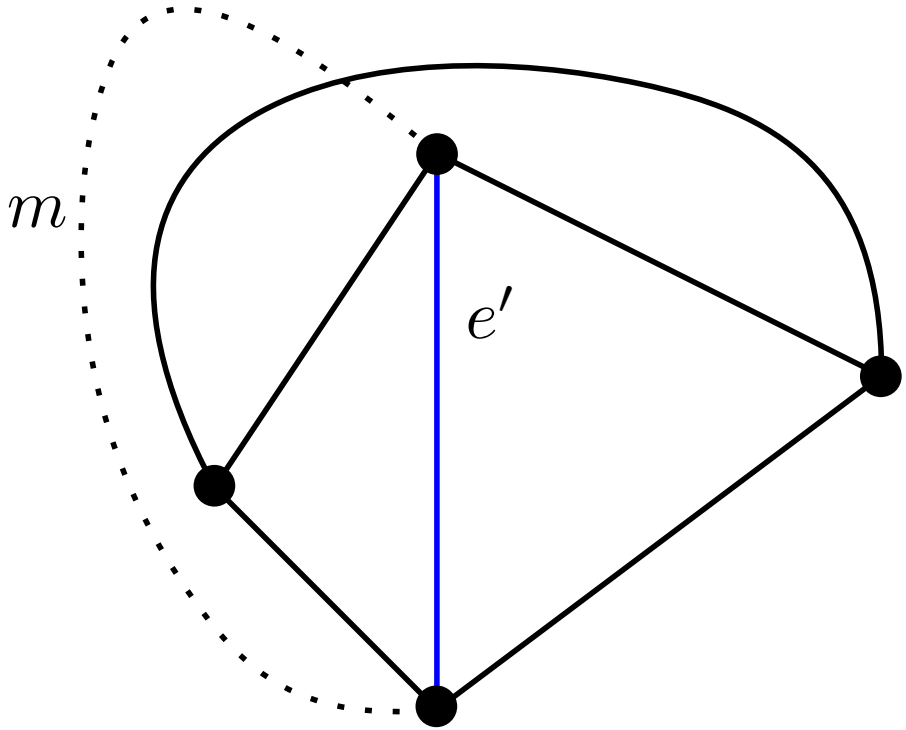
1.3 – Triangulierung

3. Löse Schleifen und **Multikanten** durch Kantentausch auf.
   - Sei e eine Multikanten.
   - Kann e' eine Multikante sein?



1.3 – Triangulierung

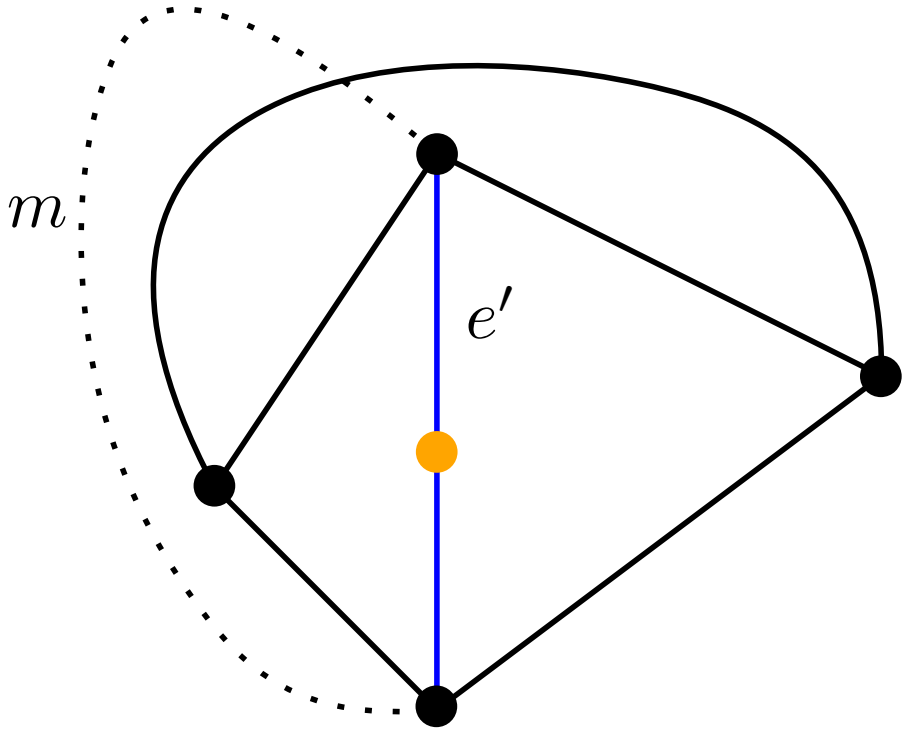
3. Löse Schleifen und **Multikanten** durch Kantentausch auf.



- Sei e eine Multikanten.
- Kann e' eine Multikante sein?
- Angenommen es gibt eine planare Einbettung mit Kante m.

1.3 – Triangulierung

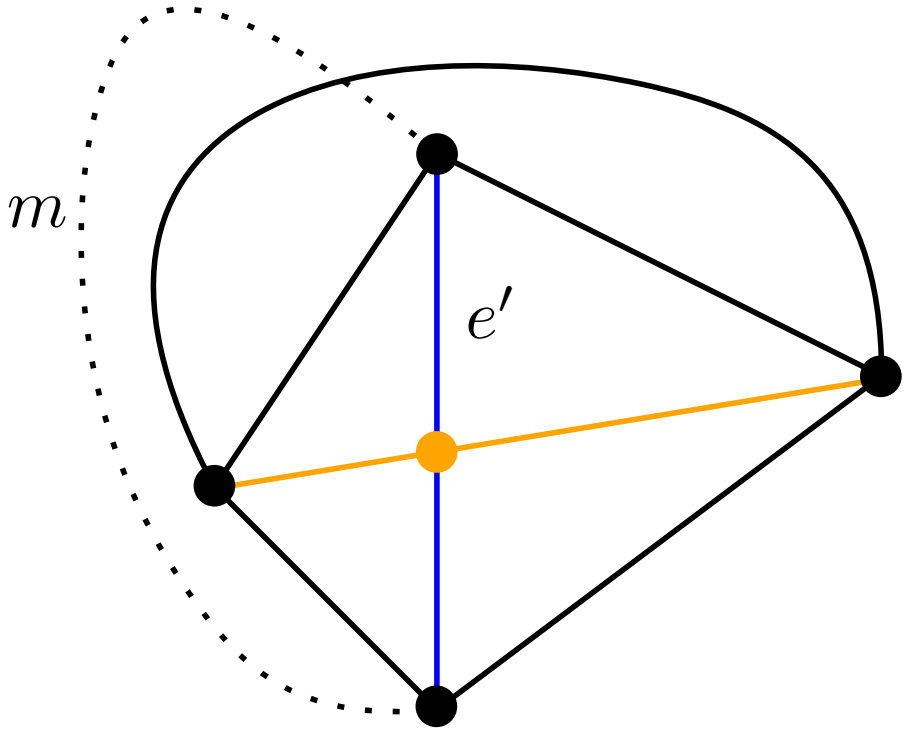
3. Löse Schleifen und **Multikanten** durch Kantentausch auf.



- Sei e eine Multikanten.
- Kann e' eine Multikante sein?
- Angenommen es gibt eine planare Einbettung mit Kante m.
- Unterteile e'.

1.3 – Triangulierung

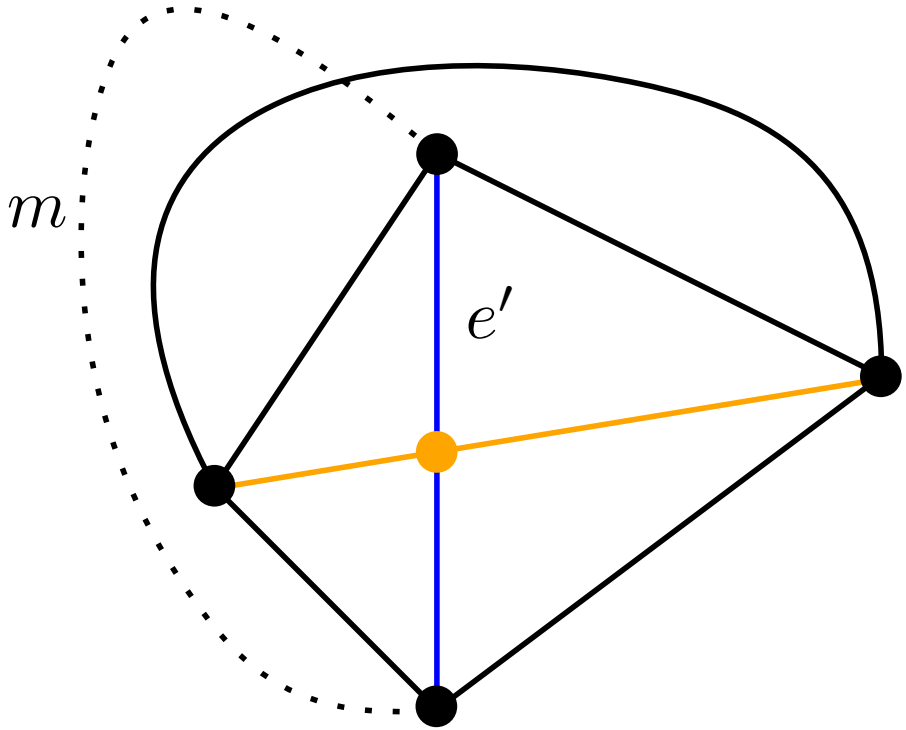
3. Löse Schleifen und **Multikanten** durch Kantentausch auf.



- Sei e eine Multikanten.
- Kann e' eine Multikante sein?
- Angenommen es gibt eine planare Einbettung mit Kante m.
- Unterteile e'.
- Füge weitere Kanten ein, die die Planarität nicht verletzen.

1.3 – Triangulierung

3. Löse Schleifen und **Multikanten** durch Kantentausch auf.



- Sei e eine Multikanten.
- Kann e' eine Multikante sein?
- Angenommen es gibt eine planare Einbettung mit Kante m.
- Unterteile e'.
- Füge weitere Kanten ein, die die Planarität nicht verletzen.
- Planare Einbettung für K_5 gefunden. ↯

1 – Triangulierung Laufzeit

Wenn Schleifen und Multikanten bekannt sind, dann braucht Schritt 3 lineare Zeit: $\mathcal{O}(1)$ pro Facette.

Lemma

Doppelkanten und Schleifen können in $\mathcal{O}(n)$ Zeit bestimmt werden.

Für jeden Konten $v \in V$:

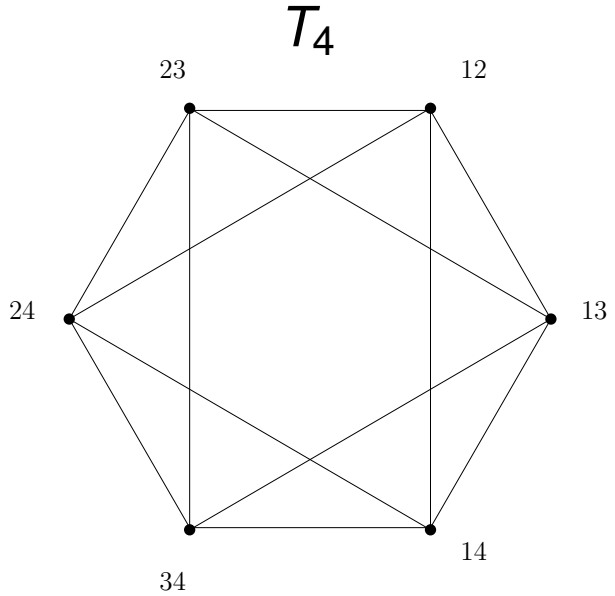
- Iteriere über alle ausgehende Kante von v und markiere benachbarte Knoten.
- Wird ein Knoten mehr als einmal markiert ist eine Multikante gefunden.
- Wird v markiert ist eine Schleife gefunden.

Jede *gerichtete* Kante wird einmal besucht $\Rightarrow \mathcal{O}(n)$

2.1 – Der Petersengraph

Definition: T_n

- Die Knoten sind alle zweielementigen Teilmengen von $\{1, \ldots, n\}$.
- Zwei Knoten sind genau dann verbunden, wenn der Schnitt der zugehörigen Mengen nicht leer ist.

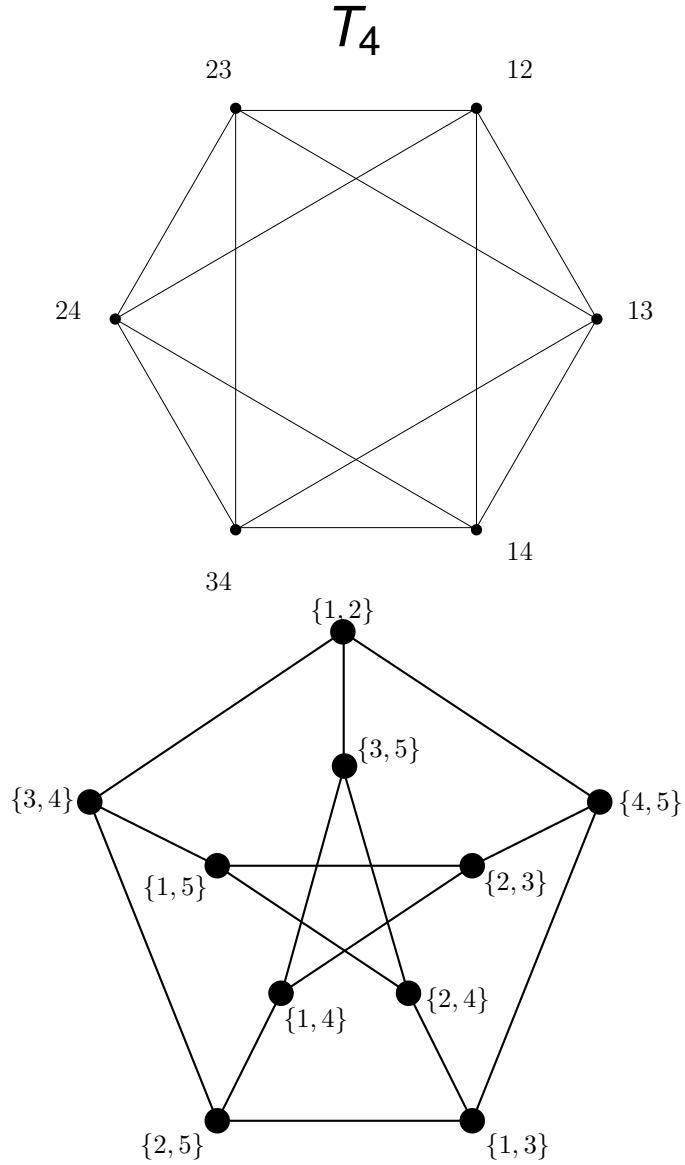


2.1 – Der Petersengraph

Definition: T_n

- Die Knoten sind alle zweielementigen Teilmengen von $\{1, \ldots, n\}$.
- Zwei Knoten sind genau dann verbunden, wenn der Schnitt der zugehörigen Mengen nicht leer ist.

Der Komplementgraph P von T_5 heißt Petersengraph.



Satz von Kuratowski

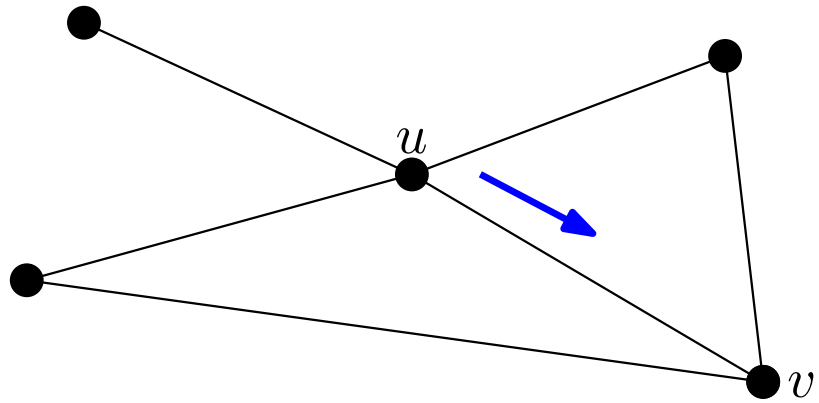
Es gibt zwei Varianten des Satzes von Kuratowski:

- Ein einfacher Graph G ist genau dann planar, wenn er weder eine Unterteilung von $K_{3,3}$ noch eine Unterteilung von K_5 als Teilgraph enthält. (Topologischer Minor)
- Ein einfacher Graph G ist genau dann planar, wenn er weder $K_{3,3}$ noch K_5 als Minor enthält. (Wagner)

Minor

Kontraktion der Kante $e = \{u, v\}$

- Lösche Kante e.
- Identifiziere u und v.
- Lösche entstehende Mehrfachkanten.



Minor

- G ist Minor von H, wenn H einen Teilgraphen enthält, aus dem durch Kantenkontraktion G hervorgeht.

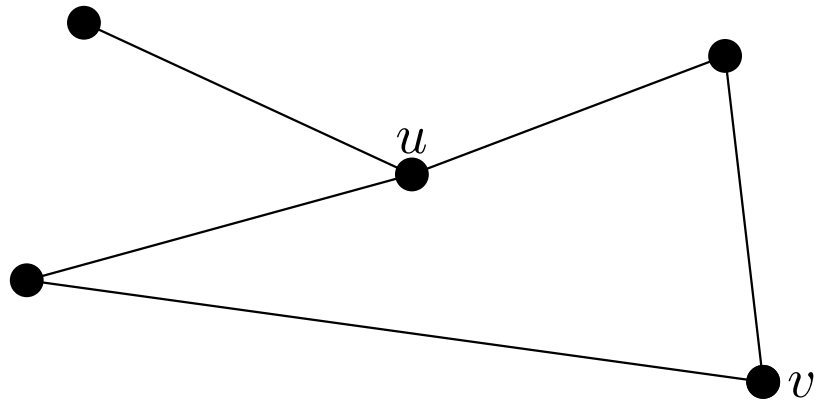
Topologischer Minor

- G ist topologischer Minor von H, wenn H einen Unterteilungsgraphen von G enthält.

Minor

Kontraktion der Kante $e = \{u, v\}$

- Lösche Kante e.
- Identifiziere u und v.
- Lösche entstehende Mehrfachkanten.



Minor

- G ist Minor von H, wenn H einen Teilgraphen enthält, aus dem durch Kantenkontraktion G hervorgeht.

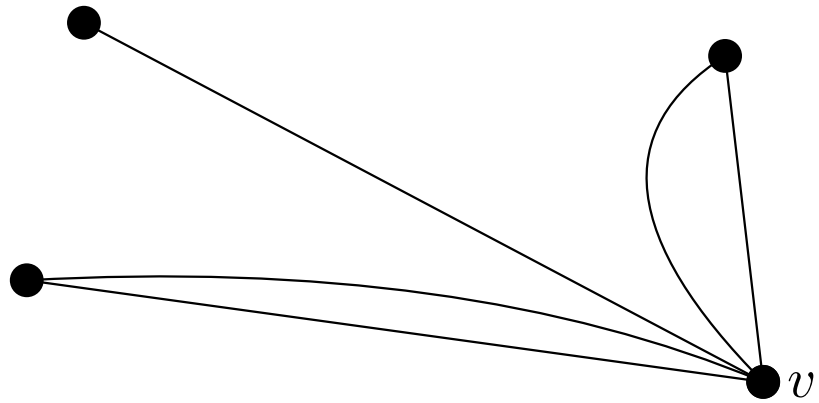
Topologischer Minor

- G ist topologischer Minor von H, wenn H einen Unterteilungsgraphen von G enthält.

Minor

Kontraktion der Kante $e = \{u, v\}$

- Lösche Kante e.
- Identifiziere u und v.
- Lösche entstehende Mehrfachkanten.



Minor

- G ist Minor von H, wenn H einen Teilgraphen enthält, aus dem durch Kantenkontraktion G hervorgeht.

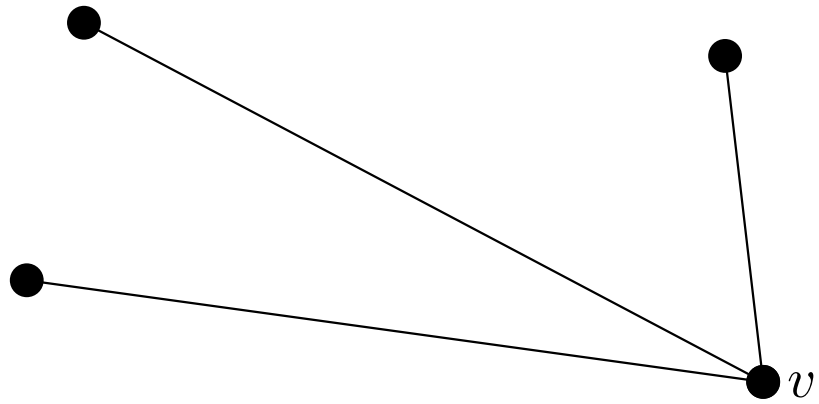
Topologischer Minor

- G ist topologischer Minor von H, wenn H einen Unterteilungsgraphen von G enthält.

Minor

Kontraktion der Kante $e = \{u, v\}$

- Lösche Kante e.
- Identifiziere u und v.
- Lösche entstehende Mehrfachkanten.



Minor

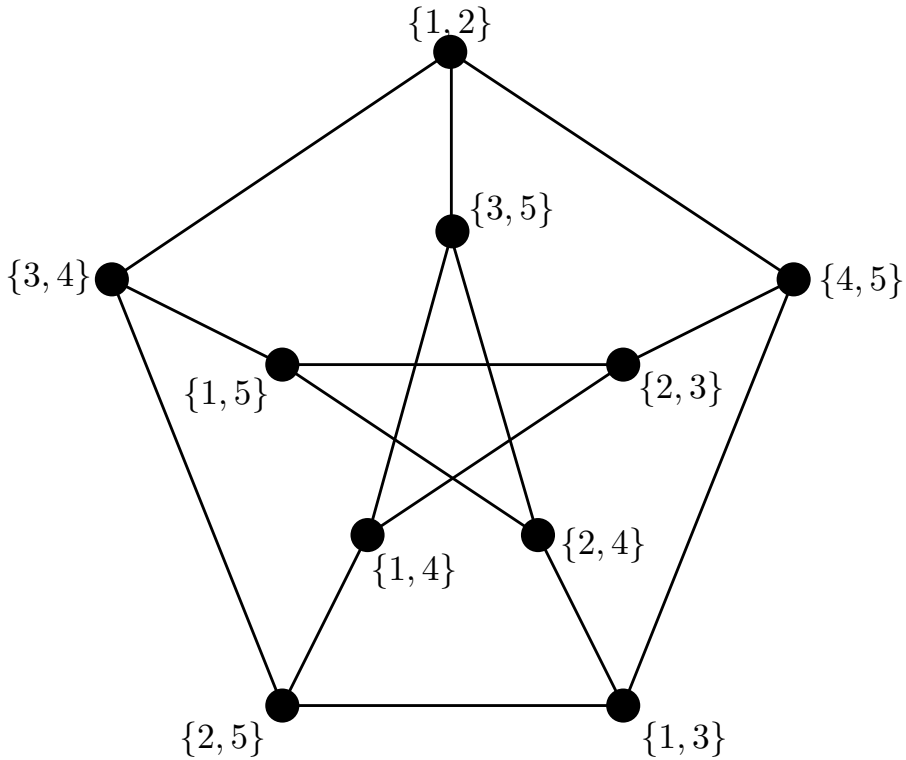
- G ist Minor von H, wenn H einen Teilgraphen enthält, aus dem durch Kantenkontraktion G hervorgeht.

Topologischer Minor

- G ist topologischer Minor von H, wenn H einen Unterteilungsgraphen von G enthält.

2.2 – Der Petersengraph

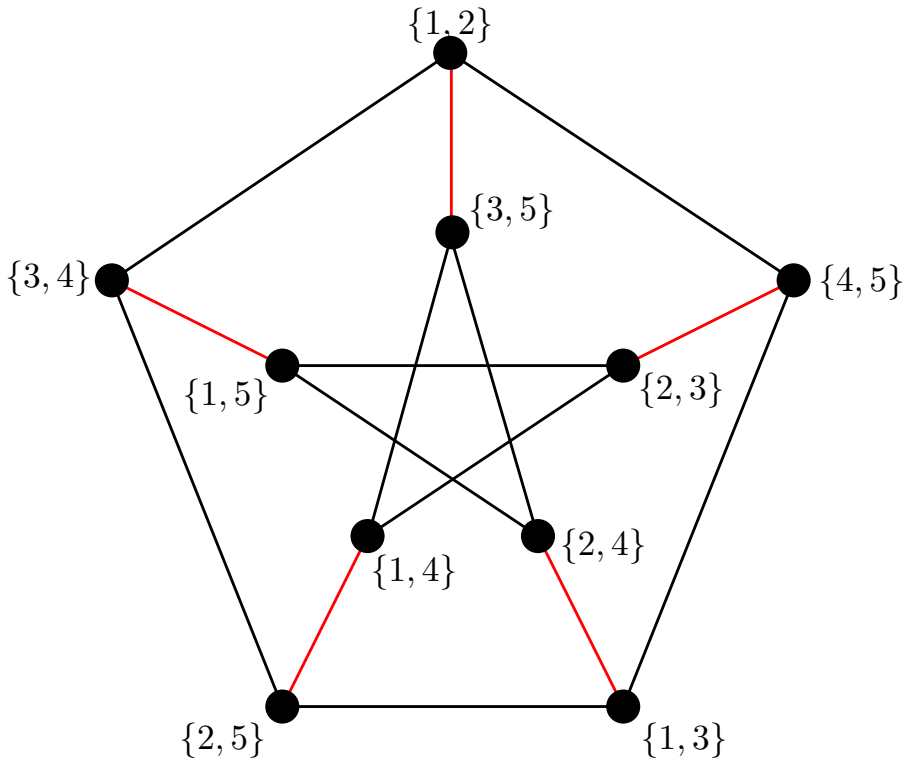
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält K_5 als Minor.

2.2 – Der Petersengraph

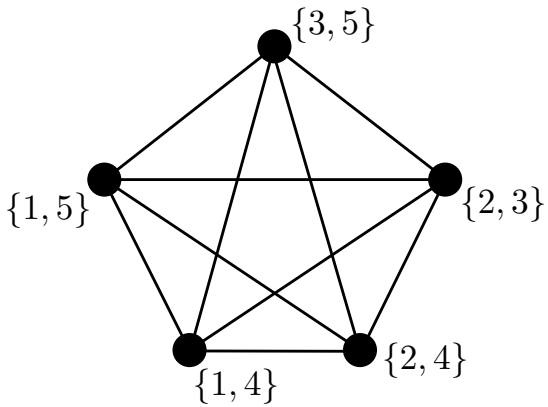
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält K_5 als Minor.

2.2 – Der Petersengraph

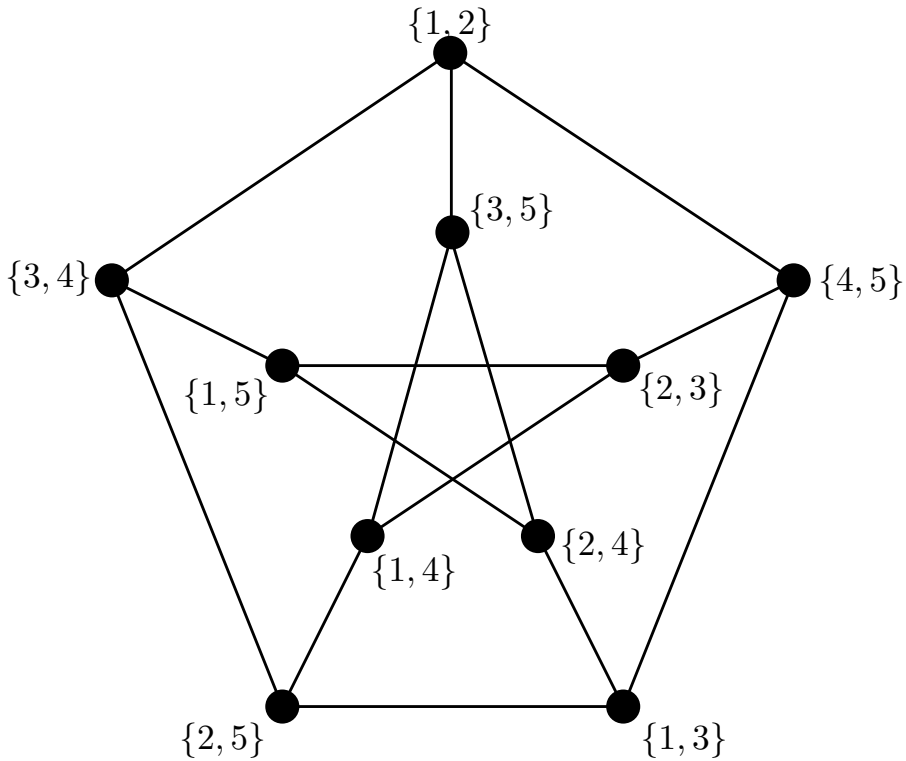
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält K_5 als Minor.

2.2 – Der Petersengraph

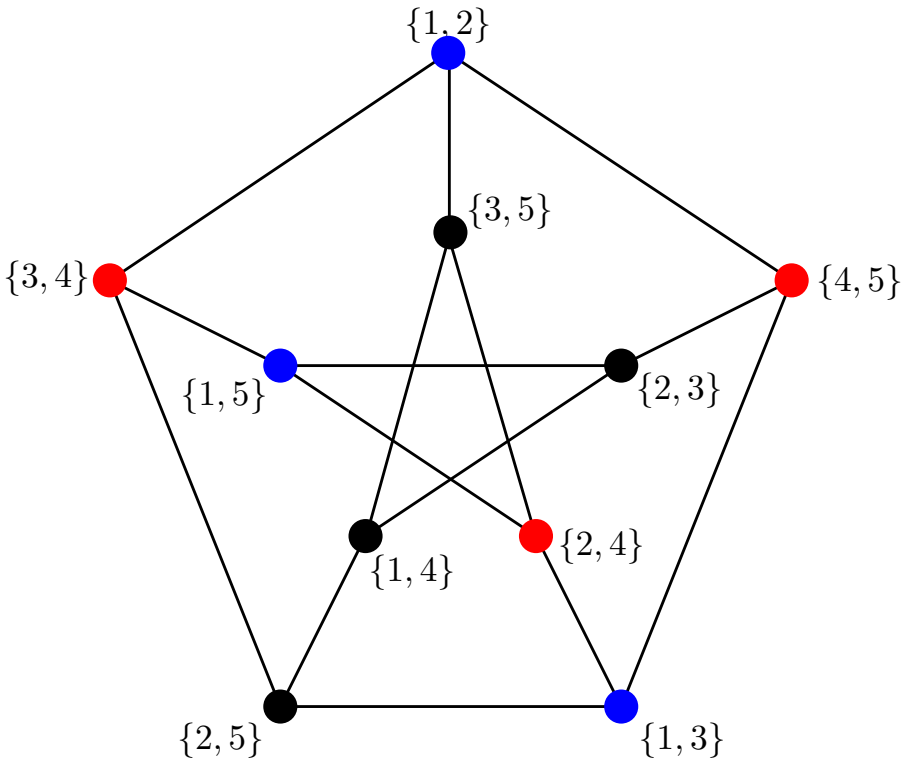
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält eine Unterteilung des $K_{3,3}$.

2.2 – Der Petersengraph

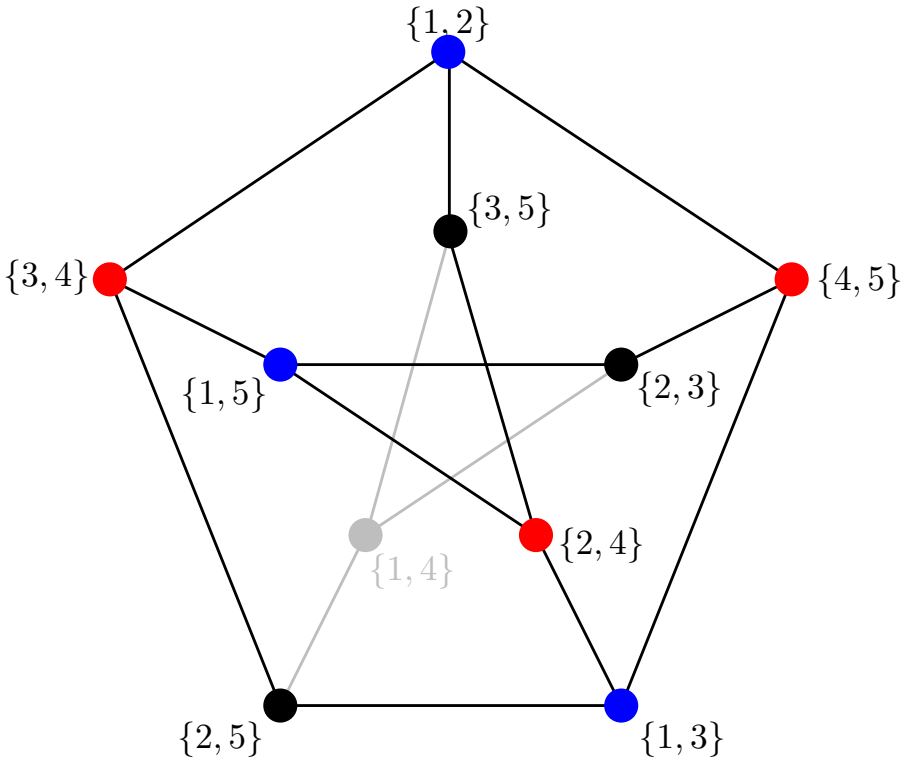
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält eine Unterteilung des $K_{3,3}$.

2.2 – Der Petersengraph

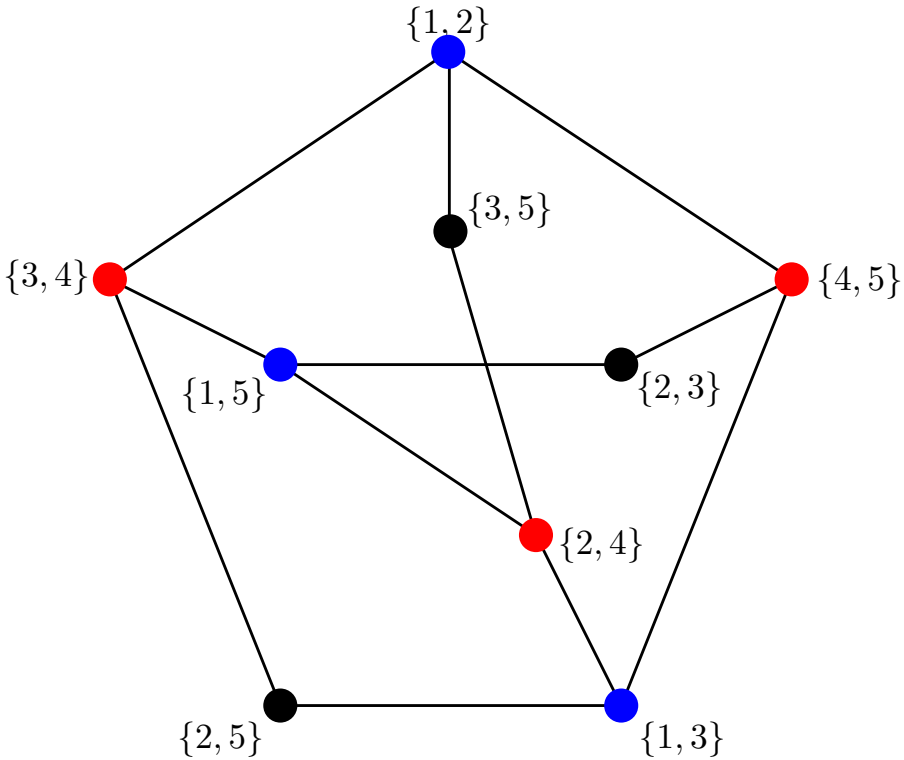
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält eine Unterteilung des $K_{3,3}$.

2.2 – Der Petersengraph

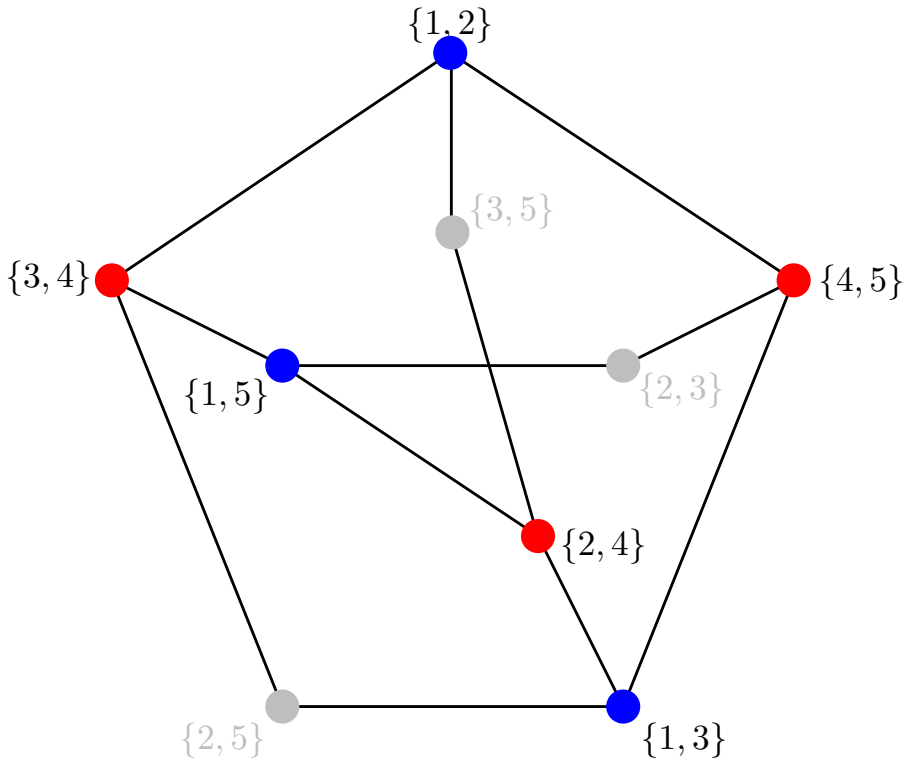
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält eine Unterteilung des $K_{3,3}$.

2.2 – Der Petersengraph

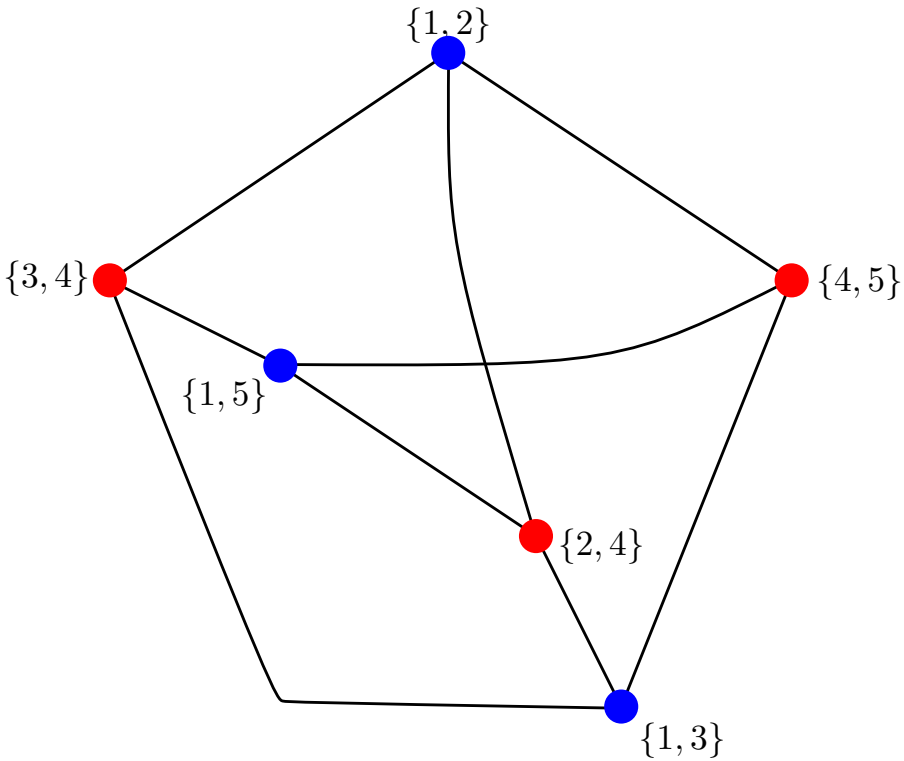
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält eine Unterteilung des $K_{3,3}$.

2.2 – Der Petersengraph

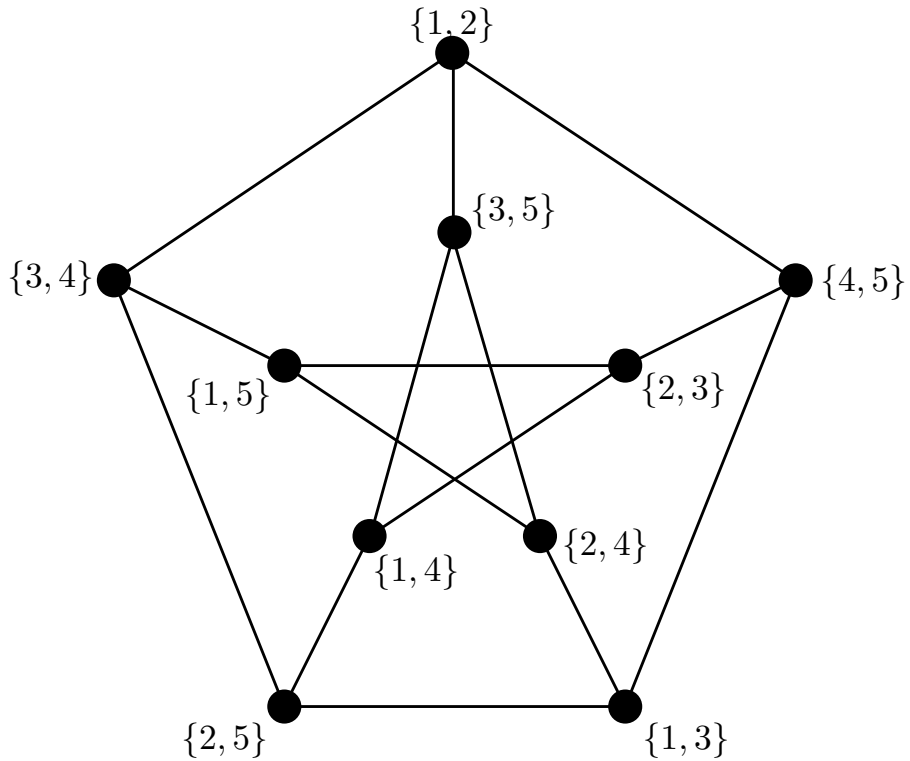
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Enthält eine Unterteilung des $K_{3,3}$.

2.2 – Der Petersengraph

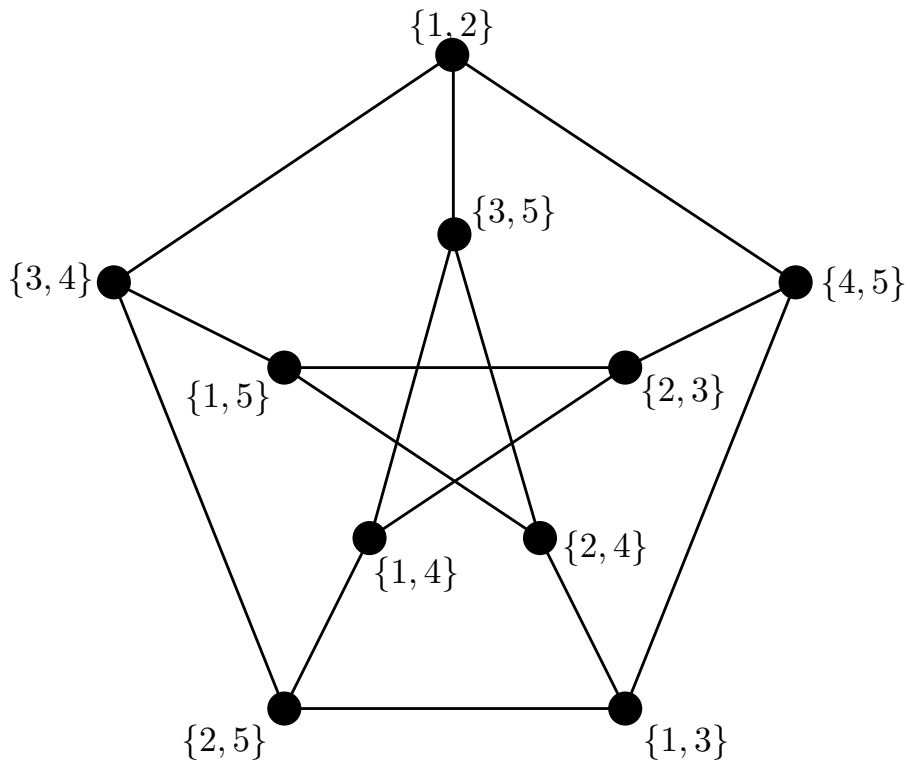
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Der kürzeste Kreis im Petersengraph hat Länge 5.
- Jede Kante gehört zu einem Kreis.
- Angenommen P wäre planar.
- P hat aber nur 15 Kanten. ↯

2.2 – Der Petersengraph

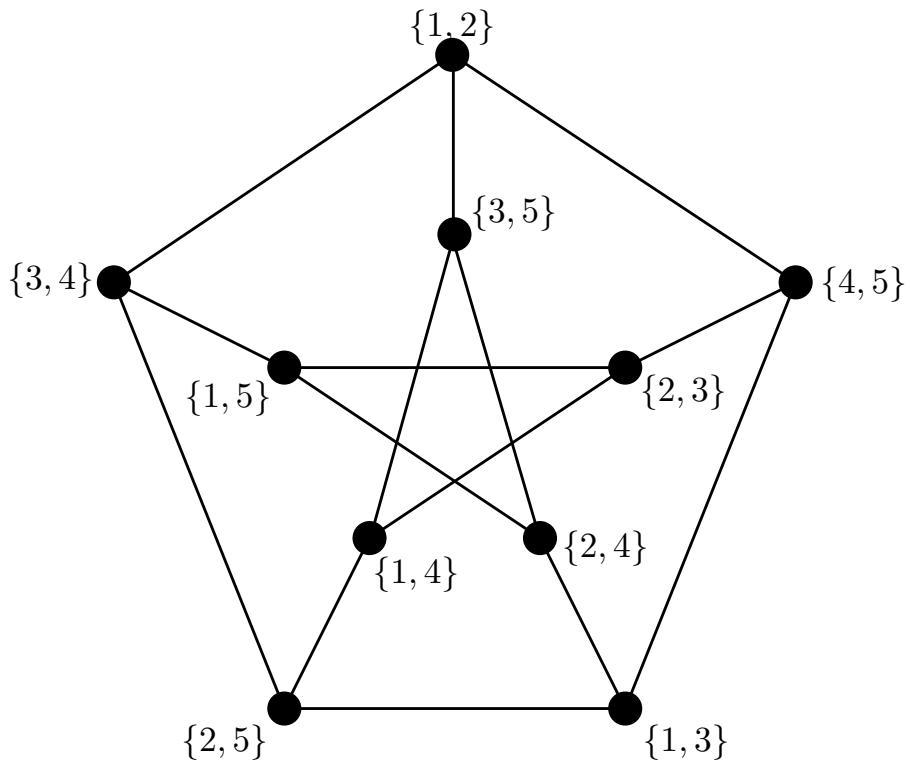
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Der kürzeste Kreis im Petersengraph hat Länge 5.
- Jede Kante gehört zu einem Kreis.
- Angenommen P wäre planar.
  - $f = m - n + 2 = 15 - 10 + 2 = 7$
  - Jede Facette wird durch mindestens 5 Kanten begrenzt.
  - P hat mindestens $(5 \cdot 7)/2 = 17$ Kanten.
- P hat aber nur 15 Kanten. ↯

2.2 – Der Petersengraph

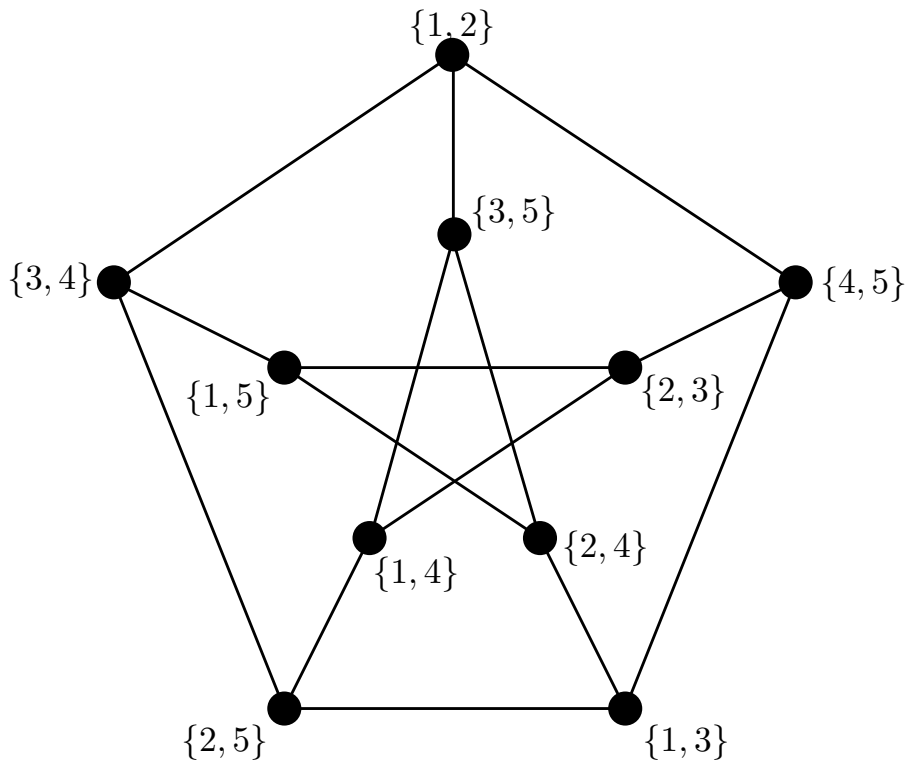
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Der kürzeste Kreis im Petersengraph hat Länge 5.
- Jede Kante gehört zu einem Kreis.
- Angenommen P wäre planar.
  - $f = m - n + 2 = 15 - 10 + 2 = 7$
  - Jede Facette wird durch mindestens 5 Kanten begrenzt.
  - P hat mindestens $(5 \cdot 7)/2 = 17$ Kanten.
- P hat aber nur 15 Kanten. ↯

2.2 – Der Petersengraph

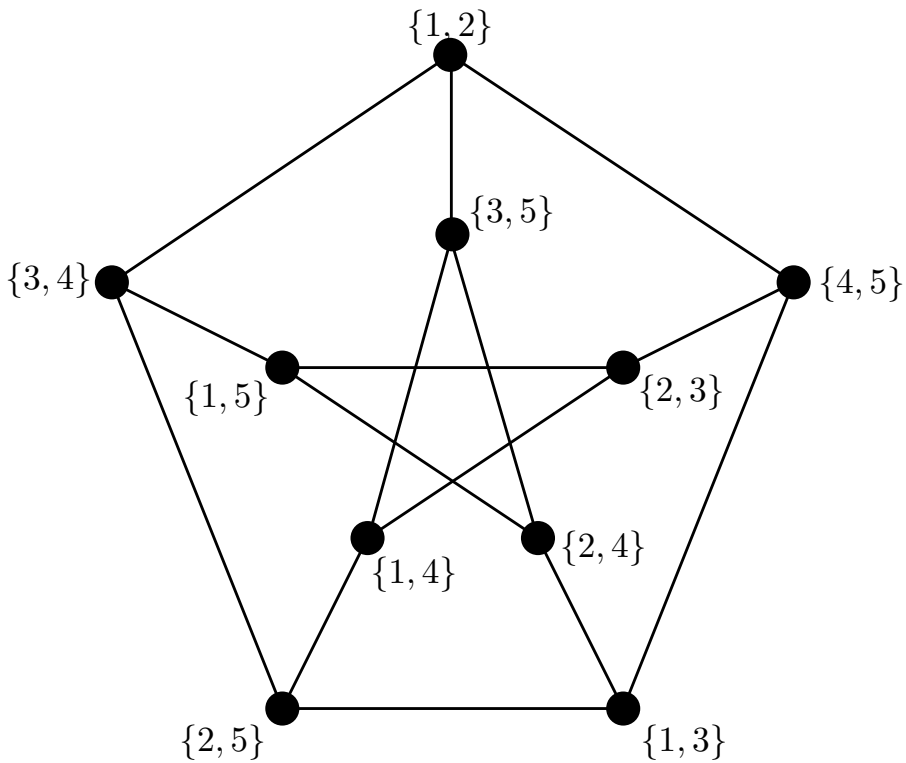
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Der kürzeste Kreis im Petersengraph hat Länge 5.
- Jede Kante gehört zu einem Kreis.
- Angenommen P wäre planar.
  - $f = m - n + 2 = 15 - 10 + 2 = 7$
  - Jede Facette wird durch mindestens 5 Kanten begrenzt.
  - P hat mindestens $(5 \cdot 7)/2 = 17$ Kanten.
- P hat aber nur 15 Kanten. ↯

2.2 – Der Petersengraph

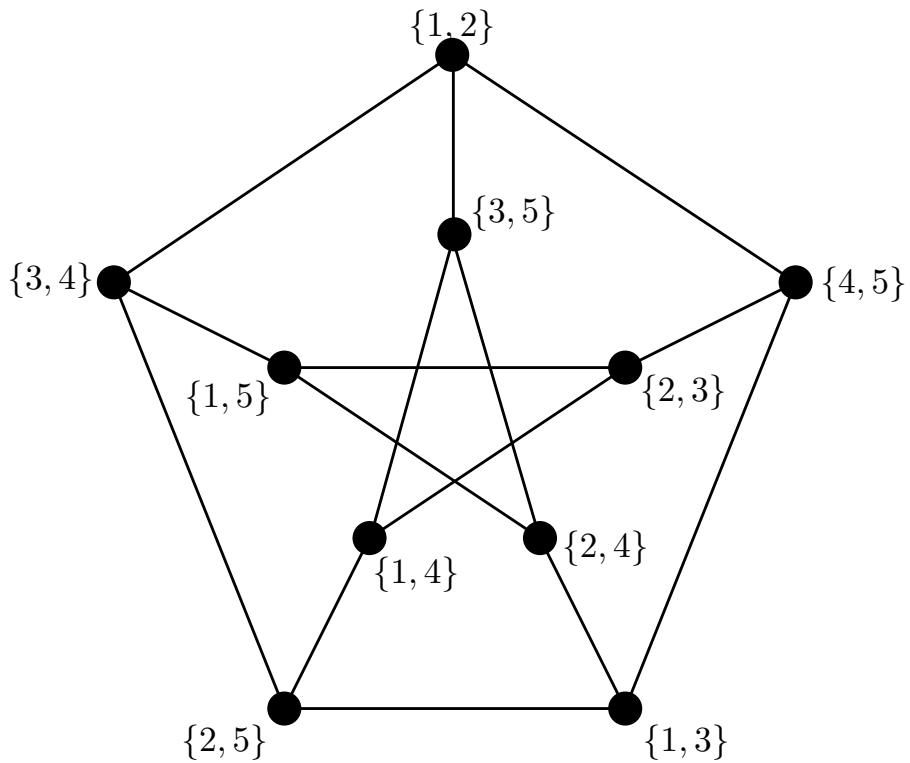
Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Der kürzeste Kreis im Petersengraph hat Länge 5.
- Jede Kante gehört zu einem Kreis.
- Angenommen P wäre planar.
  - $f = m - n + 2 = 15 - 10 + 2 = 7$
  - Jede Facette wird durch mindestens 5 Kanten begrenzt.
  - P hat mindestens $(5 \cdot 7)/2 = 17$ Kanten.
- P hat aber nur 15 Kanten. ↯

2.2 – Der Petersengraph

Zeigen Sie auf drei verschiedene Arten, dass der Petersengraph nicht planar ist.



- Der kürzeste Kreis im Petersengraph hat Länge 5.
- Jede Kante gehört zu einem Kreis.
- Angenommen P wäre planar.
  - $f = m - n + 2 = 15 - 10 + 2 = 7$
  - Jede Facette wird durch mindestens 5 Kanten begrenzt.
  - P hat mindestens $(5 \cdot 7)/2 = 17$ Kanten.
- P hat aber nur 15 Kanten. ↯

2.3 – Topologischer Minor $\Rightarrow$ Minor

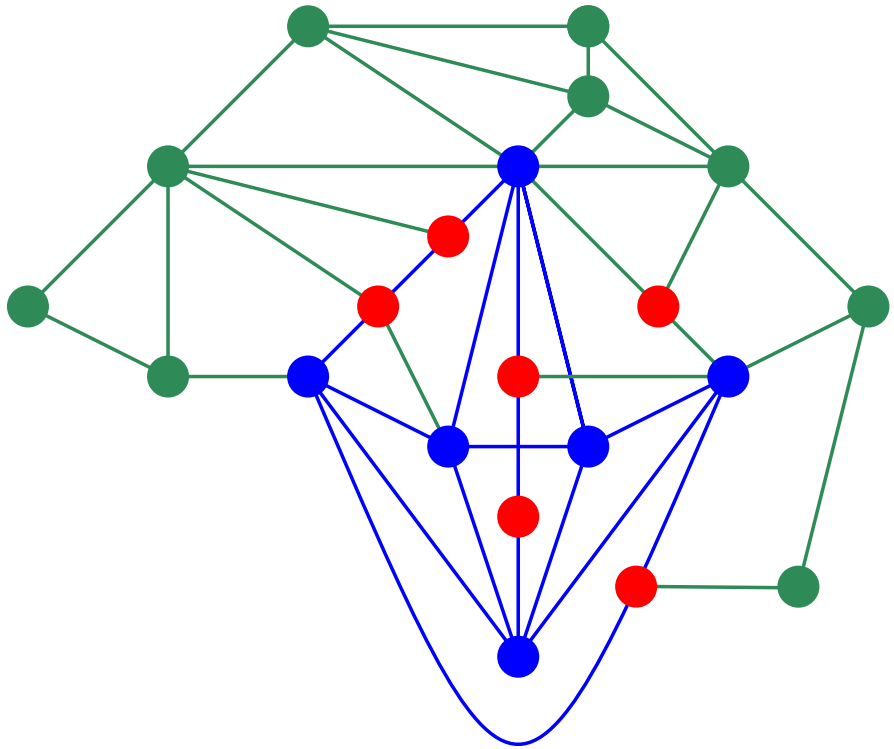
Wenn ein einfacher Graph H eine Unterteilung eines einfachen Graphen G als Teilgraph enthält, dann enthält H den Graphen G auch als Minor.

- Angenommen H enthält Unterteilung U von G.
- Lösche alle Kanten aus H die nicht zur Unterteilung von G gehören.
- Es gibt Knoten mit Grad 2, die eingefügt wurden, um U aus G zu erhalten.
- Kontrahiere jeden dieser Knoten mit einem seiner Nachbarn.

2.3 – Topologischer Minor $\Rightarrow$ Minor

Wenn ein einfacher Graph H eine Unterteilung eines einfachen Graphen G als Teilgraph enthält, dann enthält H den Graphen G auch als Minor.

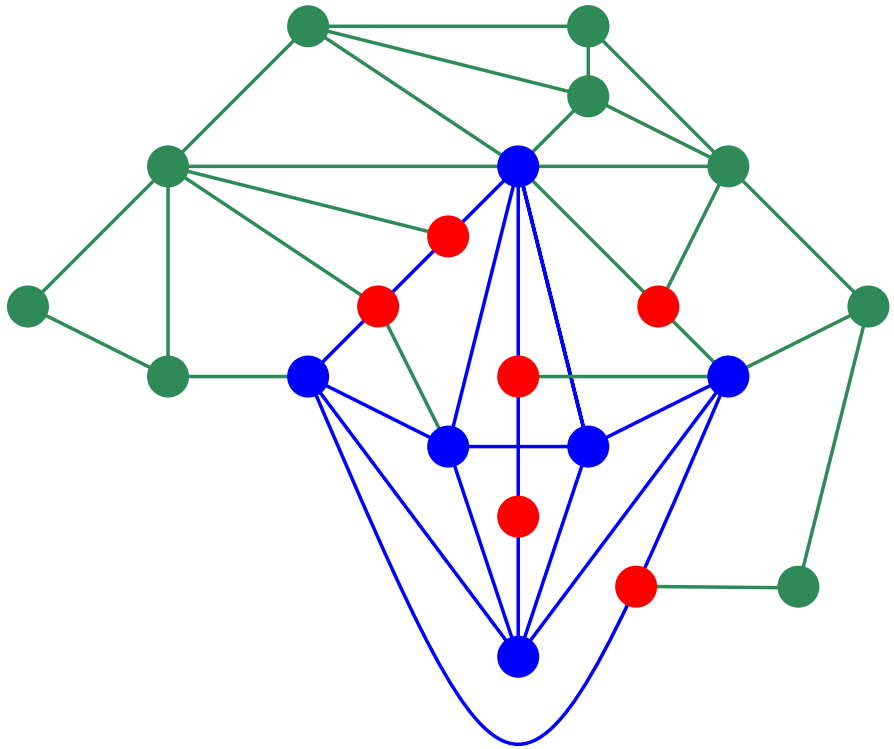
- Angenommen H enthält Unterteilung U von G.
- Lösche alle Kanten aus H die nicht zur Unterteilung von G gehören.
- Es gibt Knoten mit Grad 2, die eingefügt wurden, um U aus G zu erhalten.
- Kontrahiere jeden dieser Knoten mit einem seiner Nachbarn.



2.3 – Topologischer Minor $\Rightarrow$ Minor

Wenn ein einfacher Graph H eine Unterteilung eines einfachen Graphen G als Teilgraph enthält, dann enthält H den Graphen G auch als Minor.

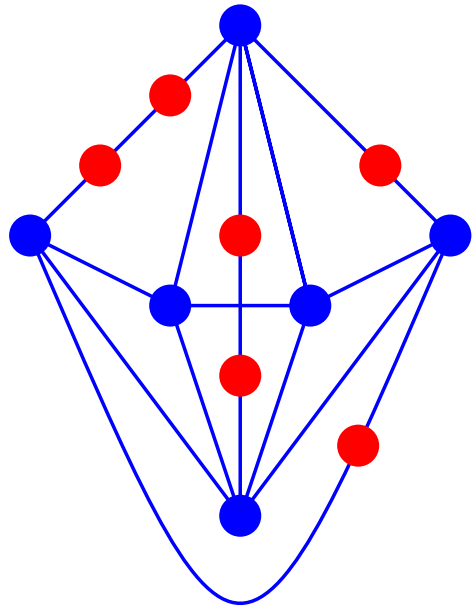
- Angenommen H enthält Unterteilung U von G.
- Lösche alle Kanten aus H die nicht zur Unterteilung von G gehören.
- Es gibt Knoten mit Grad 2, die eingefügt wurden, um U aus G zu erhalten.
- Kontrahiere jeden dieser Knoten mit einem seiner Nachbarn.



2.3 – Topologischer Minor $\Rightarrow$ Minor

Wenn ein einfacher Graph H eine Unterteilung eines einfachen Graphen G als Teilgraph enthält, dann enthält H den Graphen G auch als Minor.

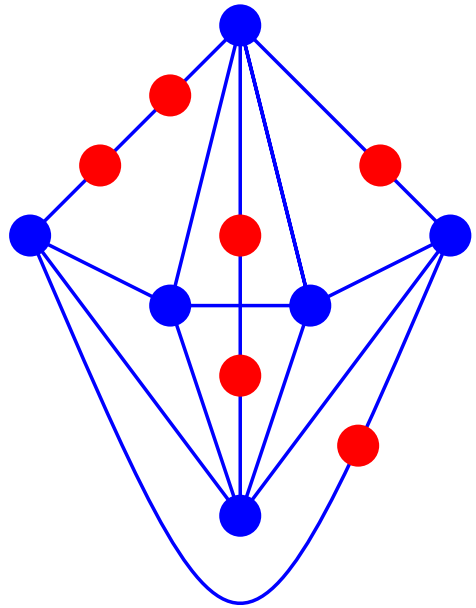
- Angenommen H enthält Unterteilung U von G.
- Lösche alle Kanten aus H die nicht zur Unterteilung von G gehören.
- Es gibt Knoten mit Grad 2, die eingefügt wurden, um U aus G zu erhalten.
- Kontrahiere jeden dieser Knoten mit einem seiner Nachbarn.



2.3 – Topologischer Minor $\Rightarrow$ Minor

Wenn ein einfacher Graph H eine Unterteilung eines einfachen Graphen G als Teilgraph enthält, dann enthält H den Graphen G auch als Minor.

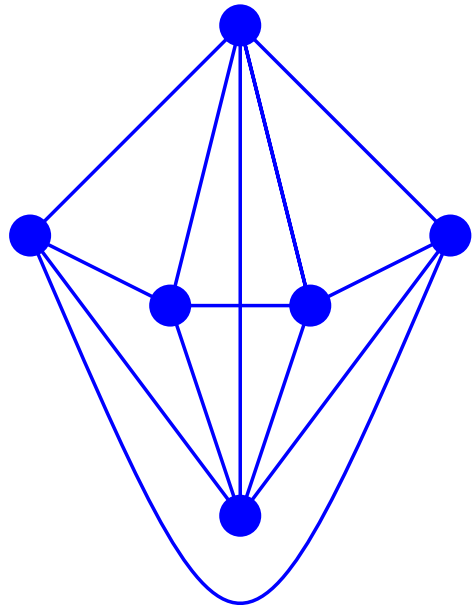
- Angenommen H enthält Unterteilung U von G.
- Lösche alle Kanten aus H die nicht zur Unterteilung von G gehören.
- Es gibt Knoten mit Grad 2, die eingefügt wurden, um U aus G zu erhalten.
- Kontrahiere jeden dieser Knoten mit einem seiner Nachbarn.



2.3 – Topologischer Minor $\Rightarrow$ Minor

Wenn ein einfacher Graph H eine Unterteilung eines einfachen Graphen G als Teilgraph enthält, dann enthält H den Graphen G auch als Minor.

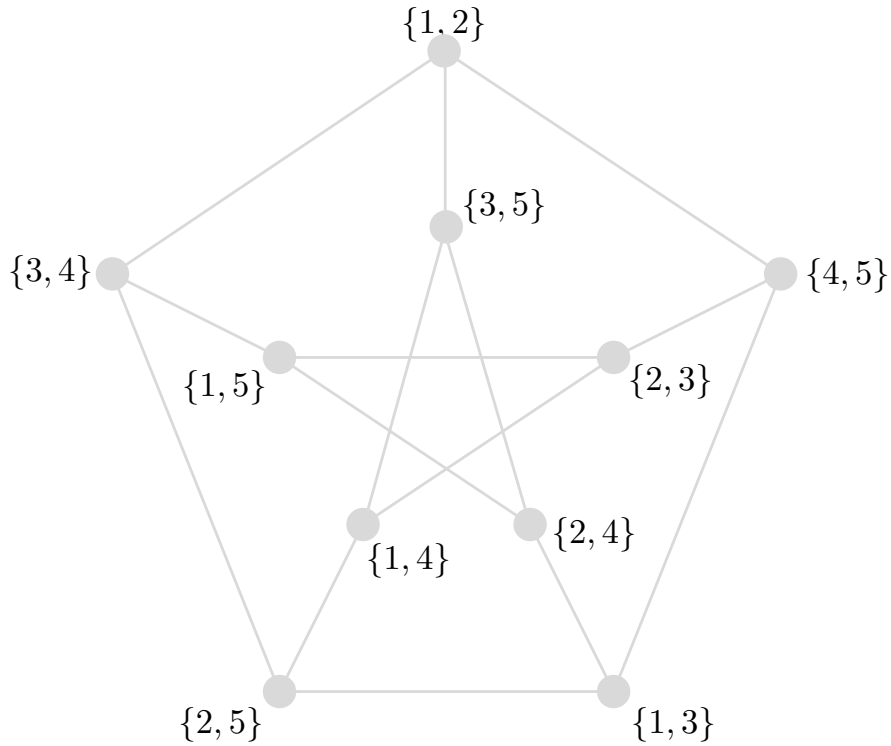
- Angenommen H enthält Unterteilung U von G.
- Lösche alle Kanten aus H die nicht zur Unterteilung von G gehören.
- Es gibt Knoten mit Grad 2, die eingefügt wurden, um U aus G zu erhalten.
- Kontrahiere jeden dieser Knoten mit einem seiner Nachbarn.



2.4 – Minor $\not\Rightarrow$ Topologischer Minor

Wenn ein einfacher Graph H einen Graphen G als Minor enthält, dann enthält H auch eine Unterteilung von G als Teilgraph. Stimmt nicht!

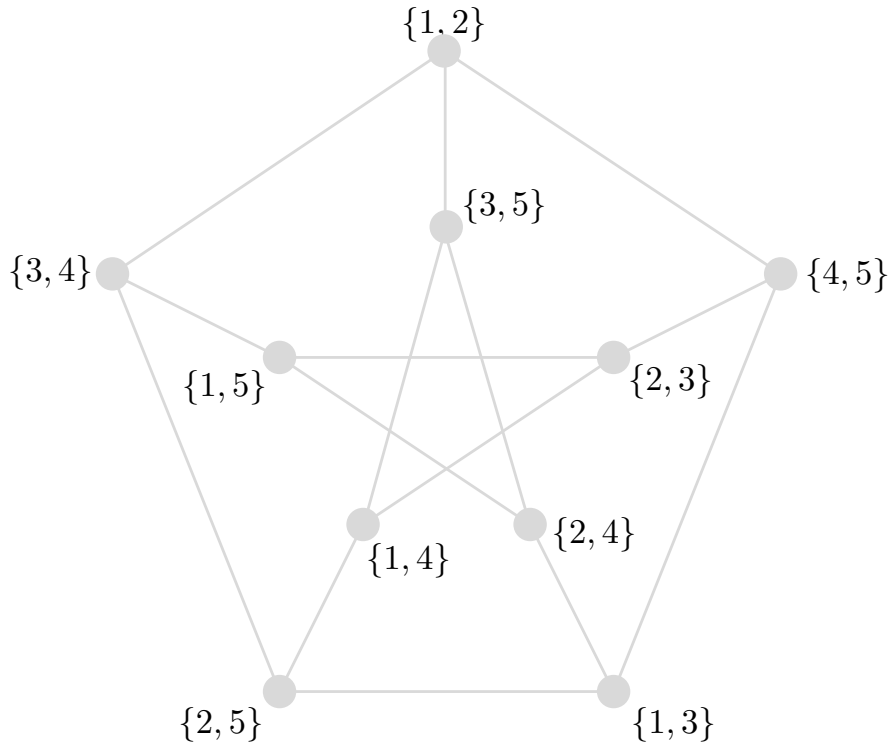
- Der Petersen-Graph P enthält K_5 als Minor.
- Unterteilung erhält den Knotengrad der ursprünglichen Knoten.
- Würde P den Graph K_5 als Unterteilung enthalten, müsste P mindestens 5 Knoten mit Grad 4 haben.



2.4 – Minor $\not\Rightarrow$ Topologischer Minor

Wenn ein einfacher Graph H einen Graphen G als Minor enthält, dann enthält H auch eine Unterteilung von G als Teilgraph. Stimmt nicht!

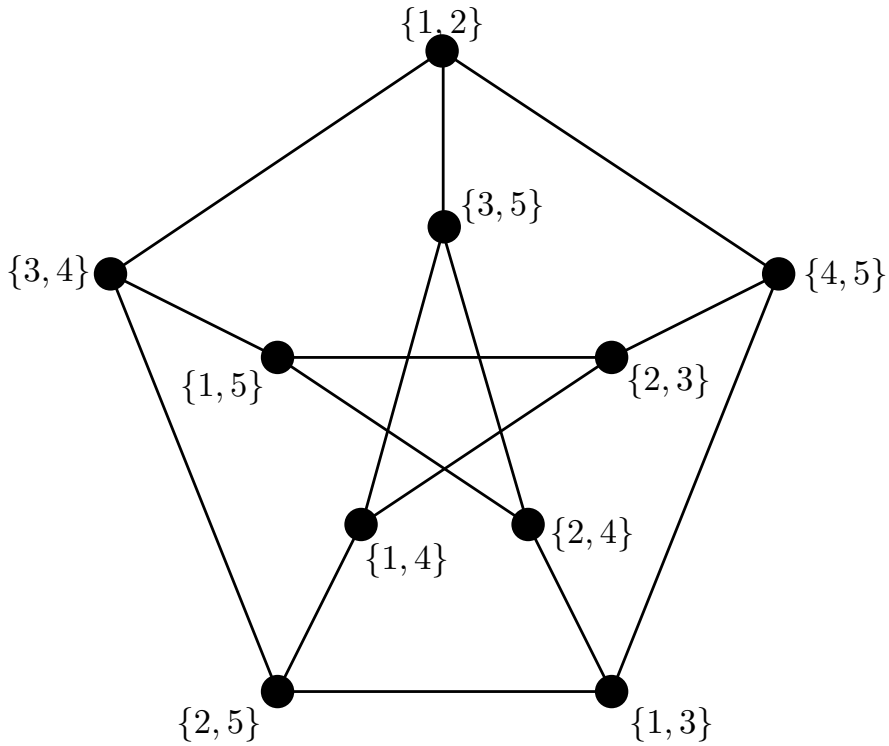
- Der Petersen-Graph P enthält K_5 als Minor.
- Unterteilung erhält den Knotengrad der ursprünglichen Knoten.
- Würde P den Graph K_5 als Unterteilung enthalten, müsste P mindestens 5 Knoten mit Grad 4 haben.



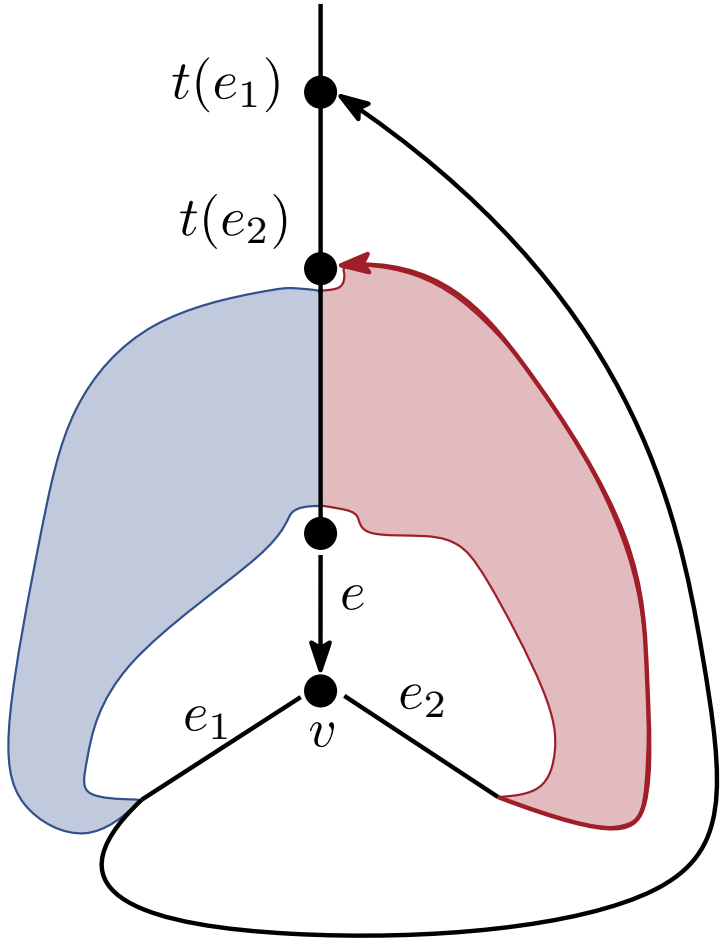
2.4 – Minor $\not\Rightarrow$ Topologischer Minor

Wenn ein einfacher Graph H einen Graphen G als Minor enthält, dann enthält H auch eine Unterteilung von G als Teilgraph. Stimmt nicht!

- Der Petersen-Graph P enthält K_5 als Minor.
- Unterteilung erhält den Knotengrad der ursprünglichen Knoten.
- Würde P den Graph K_5 als Unterteilung enthalten, müsste P mindestens 5 Knoten mit Grad 4 haben.



3 – LR-Planarität Wiederholung

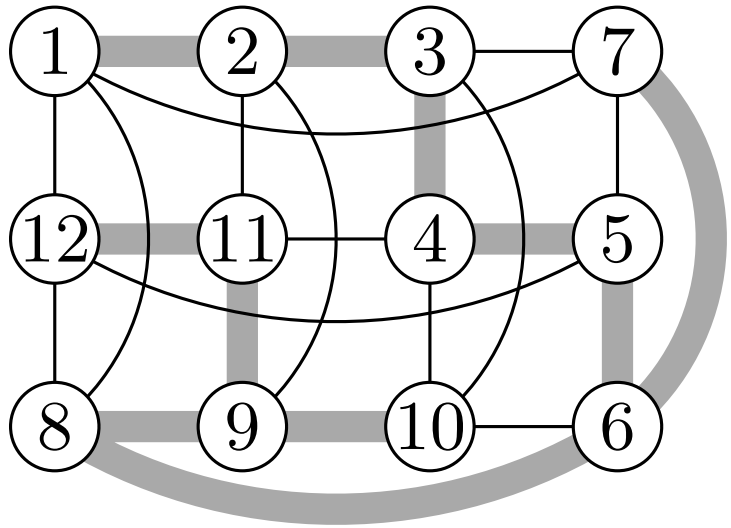


1) alle return-Kanten von e_1 oberhalb von $t(e_2)$ auf eine Seite
2) alle return-Kanten von e_2 oberhalb von $t(e_1)$ auf die andere

3.1 – LR-Planarität ausprobieren

return-Kanten von

- $(8, 9) : (11, 4), (10, 3), (12, 1), \ldots$
- $(9, 10) : (10, 3), \ldots$
- $(9, 11) : (11, 4), (12, 1), \ldots$



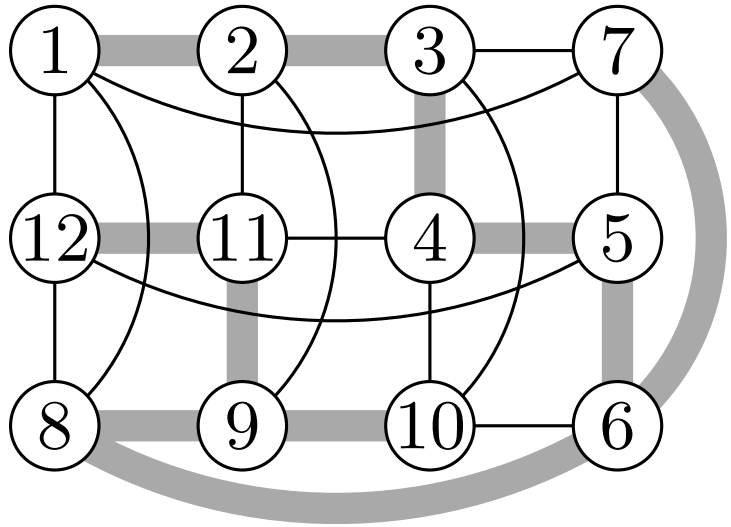
3.1 – LR-Planarität ausprobieren

return-Kanten von

- $(8, 9) : (11, 4), (10, 3), (12, 1), \ldots$
- $(9, 10) : (10, 3), \ldots$
- $(9, 11) : (11, 4), (12, 1), \ldots$

forks

- $(9, 10) \times (9, 11) \rightsquigarrow (10, 3), (11, 4)$ auf verschiedene Seiten
- $(8, 1) \times (8, 9) \rightsquigarrow (10, 3), (11, 4)$ auf gleiche Seiten



3.2 – ineffizienter Planaritätstest Sketch

- $m > 3n - 6 \Rightarrow$ nicht planar
- DFS $O(n)$
- Baue Constraint-Graph $O(?)$
  - back-edges = Knoten
  - gleiche Seite constraint $\rightsquigarrow$ Kante mit label $\varphi(a_0, a_1) = 1$
  - ungleiche Seite constraint $\rightsquigarrow$ Kante mit label $\varphi(a_0, a_1) = -1$
- LR-Partition existiert $\Leftrightarrow$ Constraint-Graph hat keinen Kreis mit ungerader Anzahl -1 labels $O(n^2)$
  - wähle beliebigen Startknoten v_0 und setze $\pi(v_0) = 1$
  - BFS von v_0
  - $\pi(v) = \pi(u) \cdot \varphi(u, v)$ wenn u parent von v ist
  - intra-layer Kanten = potentielle Konflikte

3.2 – ineffizienter Planaritätstest Sketch

- $m > 3n - 6 \Rightarrow$ nicht planar
- DFS $O(n)$
- Baue Constraint-Graph $O(?)$
  - back-edges = Knoten
  - gleiche Seite constraint $\rightsquigarrow$ Kante mit label $\varphi(a_0, a_1) = 1$
  - ungleiche Seite constraint $\rightsquigarrow$ Kante mit label $\varphi(a_0, a_1) = -1$
- LR-Partition existiert $\Leftrightarrow$ Constraint-Graph hat keinen Kreis mit ungerader Anzahl -1 labels $O(n^2)$
  - wähle beliebigen Startknoten v_0 und setze $\pi(v_0) = 1$
  - BFS von v_0
  - $\pi(v) = \pi(u) \cdot \varphi(u, v)$ wenn u parent von v ist
  - intra-layer Kanten = potentielle Konflikte

3.2 – ineffizienter Planaritätstest Sketch

- $m > 3n - 6 \Rightarrow$ nicht planar
- DFS $O(n)$
- Baue Constraint-Graph $O(?)$
  - back-edges = Knoten
  - gleiche Seite constraint $\rightsquigarrow$ Kante mit label $\varphi(a_0, a_1) = 1$
  - ungleiche Seite constraint $\rightsquigarrow$ Kante mit label $\varphi(a_0, a_1) = -1$
- LR-Partition existiert $\Leftrightarrow$ Constraint-Graph hat keinen Kreis mit ungerader Anzahl -1 labels $O(n^2)$
  - wähle beliebigen Startknoten v_0 und setze $\pi(v_0) = 1$
  - BFS von v_0
  - $\pi(v) = \pi(u) \cdot \varphi(u, v)$ wenn u parent von v ist
  - intra-layer Kanten = potentielle Konflikte

3.2 – Laufzeit Schritt 2

- post-order DFS um return edges einzusammeln. merge und filter
- Ein ungleich-constraint kann nur an einem einzigen fork entstehen
- Danach sind die return-Kanten im gleichen Subbaum
- $\rightsquigarrow O(n^2)$ um ungleich-constraints zu generieren
- das Argument funktioniert nicht für gleich-constraints

3.2 – Laufzeit Schritt 2

- post-order DFS um return edges einzusammeln. merge und filter
- Ein ungleich-constraint kann nur an einem einzigen fork entstehen
- Danach sind die return-Kanten im gleichen Subbaum
- $\rightsquigarrow O(n^2)$ um ungleich-constraints zu generieren
- das Argument funktioniert nicht für gleich-constraints

3.2 – Laufzeit Schritt 2

- post-order DFS um return edges einzusammeln. merge und filter
- Ein ungleich-constraint kann nur an einem einzigen fork entstehen
- Danach sind die return-Kanten im gleichen Subbaum
- $\rightsquigarrow O(n^2)$ um ungleich-constraints zu generieren
- das Argument funktioniert nicht für gleich-constraints

4.1 – Umfang

Der *Umfang* (engl. girth) eines Graphen G ist die Länge eines kürzesten Kreises in G. Enthält G keinen Kreis, so ist der Umfang ∞.

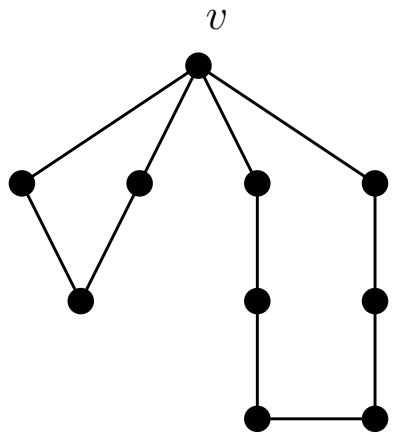
Geben Sie einen Algorithmus an, der für einen Knoten v von G entweder

- die Länge des kürzesten Kreises berechnet auf dem v liegt, oder
- entscheidet, dass v nicht auf einem Kreis liegt.

4.1 – Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.



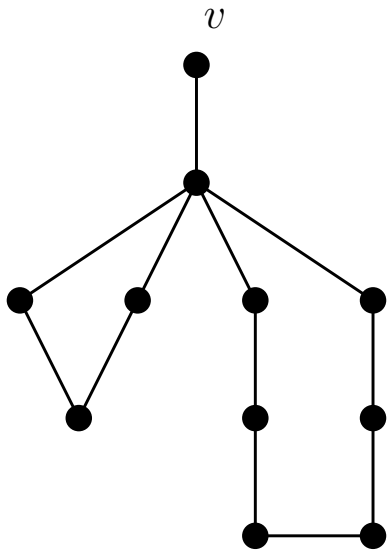
4.1 – Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.

Aber: v liegt nicht notwendigerweise auf diesem Kreis. Deshalb:

- Nummeriere (“labele”) Knoten auf dem ersten Level (Nachbarn von v)
- Vererbe Label auf neu gefundene Knoten
- Wenn sich unterschiedliche Label treffen ist das ein Kreis, der v enthält.



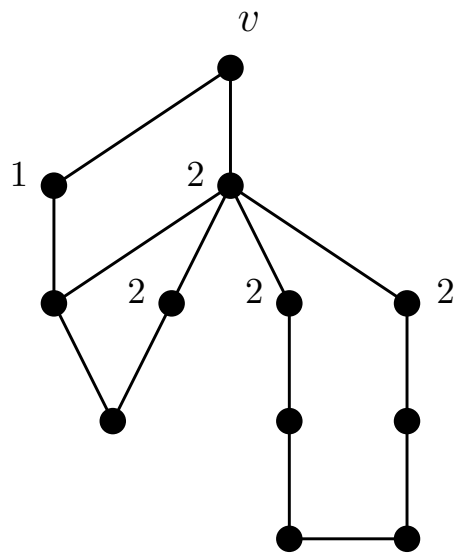
4.1 – Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.

Aber: v liegt nicht notwendigerweise auf diesem Kreis. Deshalb:

- Nummeriere (“labele”) Knoten auf dem ersten Level (Nachbarn von v)
- Vererbe Label auf neu gefundene Knoten
- Wenn sich unterschiedliche Label treffen ist das ein Kreis, der v enthält.



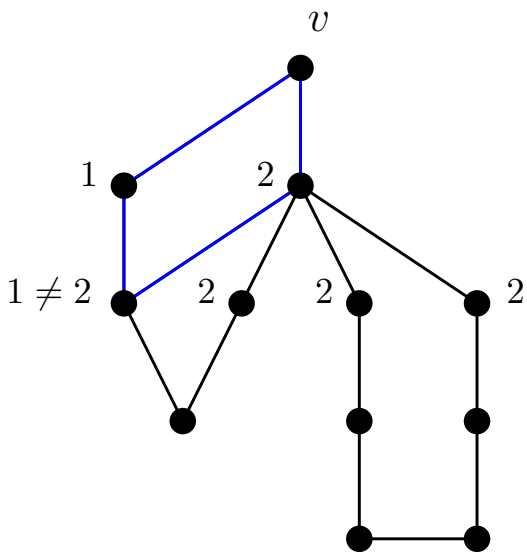
4.1 – Umfang

SHORTCIRCLE(v)

- Breitensuche beginnend bei v
- Wird ein Knoten zum zweiten Mal besucht, ist ein Kreis gefunden.
- Ein kürzester Kreis wird zuerst gefunden.

Aber: v liegt nicht notwendigerweise auf diesem Kreis. Deshalb:

- Nummeriere (“labele”) Knoten auf dem ersten Level (Nachbarn von v)
- Vererbe Label auf neu gefundene Knoten
- Wenn sich unterschiedliche Label treffen ist das ein Kreis, der v enthält.



4.2 – Umfang

Verwenden Sie das Verfahren aus Aufgabenteil 6.1, um für einen beliebigen Graphen den Umfang zu berechnen. Welche Laufzeit erhalten Sie?

- Breitensuche für eine Zusammenhangskomponente liegt in $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$.
- Wiederhole für jeden Knoten und gib den kleinsten Kreis aus: $\mathcal{O}(|V||E|)$.

4.2 – Umfang

Verwenden Sie das Verfahren aus Aufgabenteil 6.1, um für einen beliebigen Graphen den Umfang zu berechnen. Welche Laufzeit erhalten Sie?

- Breitensuche für eine Zusammenhangskomponente liegt in $\mathcal{O}(|V| + |E|) = \mathcal{O}(|E|)$.
- Wiederhole für jeden Knoten und gib den kleinsten Kreis aus: $\mathcal{O}(|V||E|)$.

4.3 – Umfang für planare Graphen

Algorithm PLANARGIRTH

$C \leftarrow \infty$

for jede Zusammenhangskomponente H von G **do**

$(V_1, V_2, S) \leftarrow$ PLANARSEPARATOR(H)
$C \leftarrow \min\{C, \min\{\text{SHORTCIRCLE}(v) \mid v \in S\}\}$

$C \leftarrow \min\{C,$ PLANARGIRTH$(V_1)\}$
$C \leftarrow \min\{C,$ PLANARGIRTH$(V_2)\}$

end for

return C

4.3 – Umfang für planare Graphen

Algorithm PLANARGIRTH

$C \leftarrow \infty$

for jede Zusammenhangskomponente H von G **do** $\mathcal{O}(n_G^{1,5} \log(n_G))$

$(V_1, V_2, S) \leftarrow$ PLANARSEPARATOR(H) $\mathcal{O}(n_H)$ $\mathcal{O}(n_H^{1,5})$

$C \leftarrow \min\{C, \min\{\text{SHORTCIRCLE}(v) \mid v \in S\}\}$ $\mathcal{O}(\sqrt{n_H} \cdot n_H)$

$C \leftarrow \min\{C,$ PLANARGIRTH$(V_1)\}$ $\mathcal{O}(T(|V_1|))$ $\mathcal{O}(T(n_H))$

$C \leftarrow \min\{C,$ PLANARGIRTH$(V_2)\}$ $\mathcal{O}(T(|V_2|))$

end for

return C

- $\mathcal{O}(\log(n))$ Rekursionslevel
- Jeder Knoten nur einmal pro Level $\Rightarrow$ pro Level $\mathcal{O}(n^{1,5})$ Arbeit
- Insgesamt: $\mathcal{O}(\log(n) \cdot n^{1,5})$

5 – Adjazenztest

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

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

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

5 – Adjazenztest

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

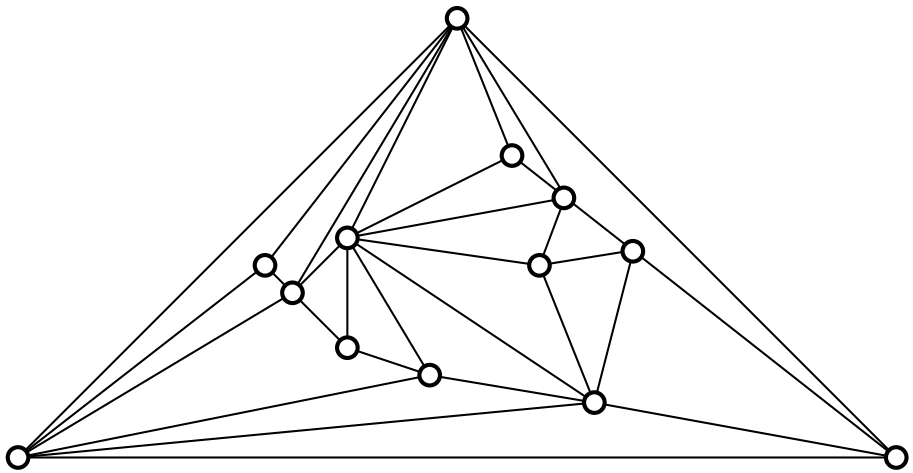
- Gegeben: G ungerichteter, planarer Graph.
- Wir haben lineare Vorberechnungszeit und können linear viel zusätzlichen Speicher benutzen.

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

5 – Adjazenztest

- 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.
- $\widetilde{\mathcal{N}}(v)$: Nachbarschaft von v über Kanten die noch nicht gerichtet wurden.
- $\mathcal{N}^+(v)$: Über gerichtete, von v ausgehende Kanten benachbarte Knoten.

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

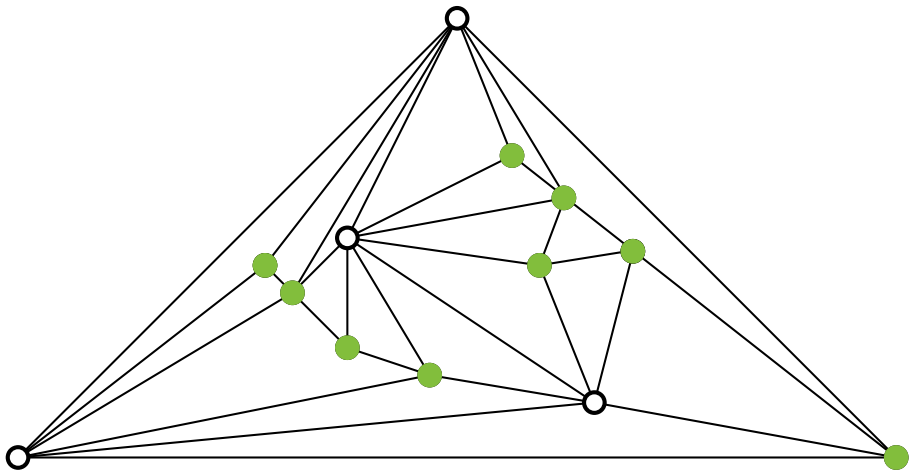
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

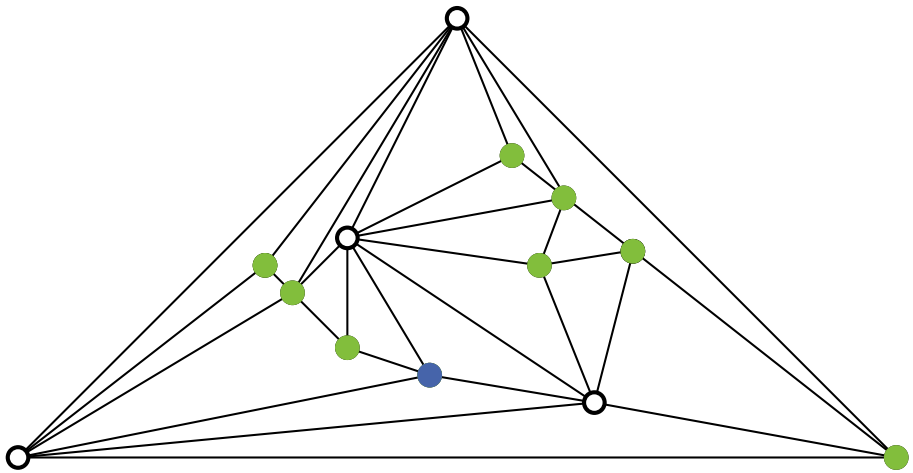
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

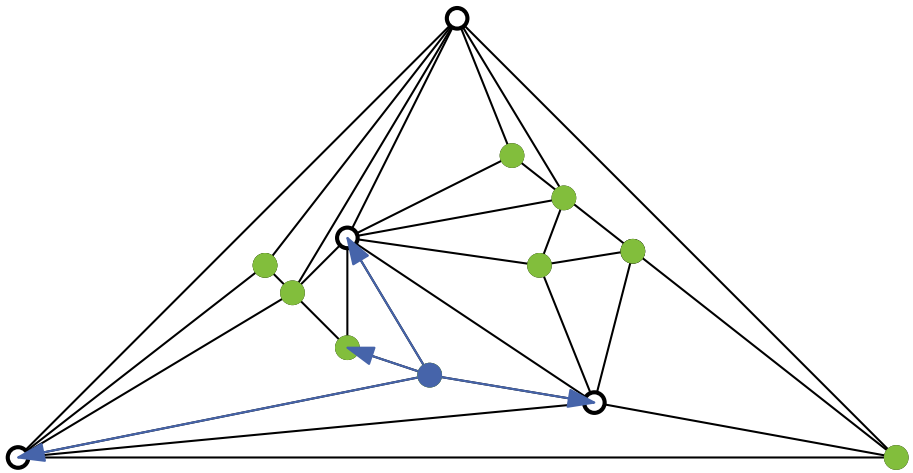
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

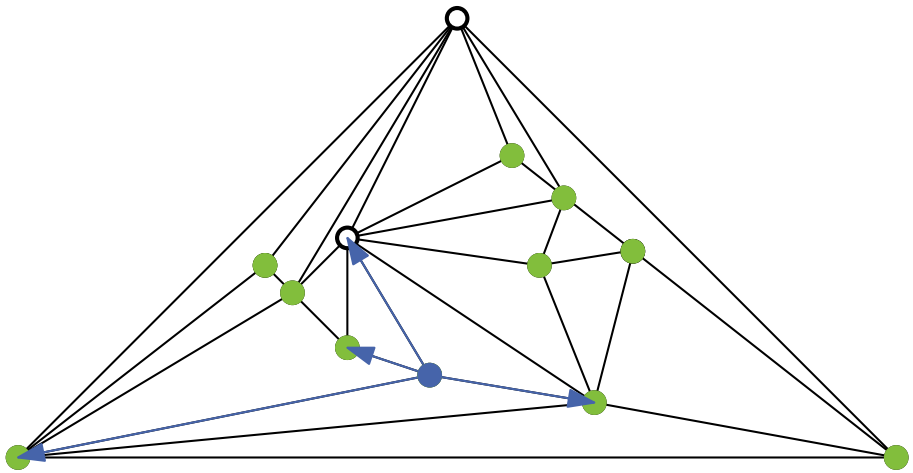
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

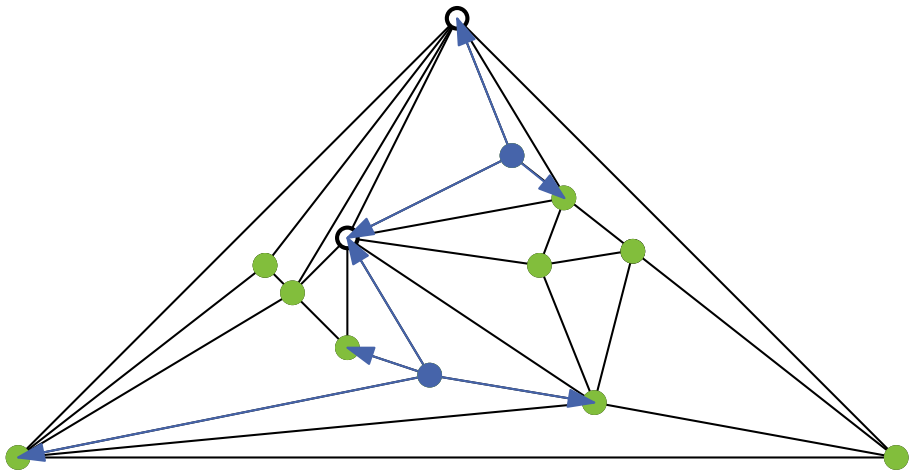
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

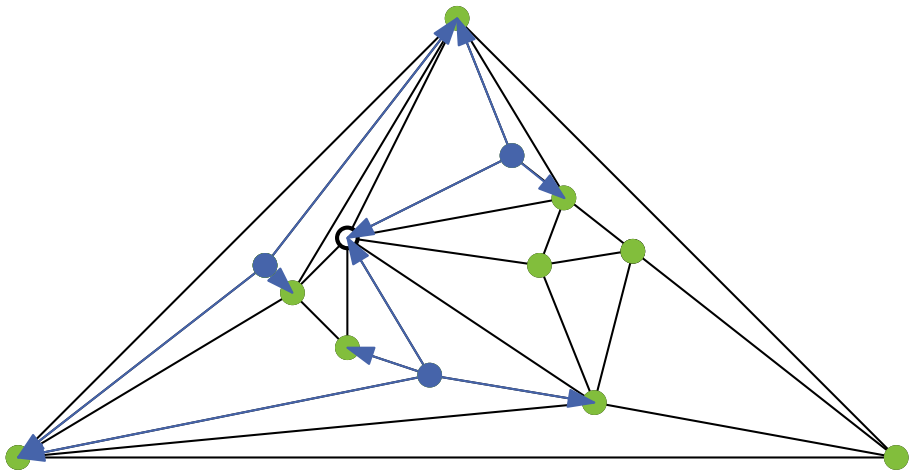
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

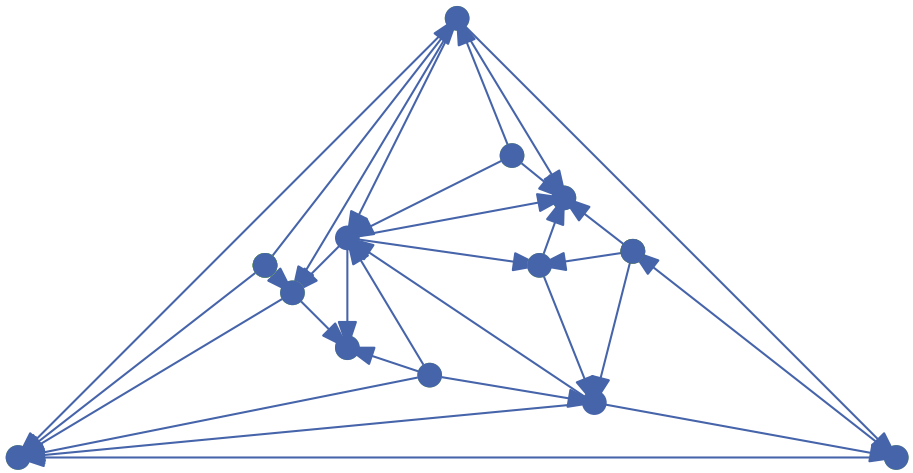
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

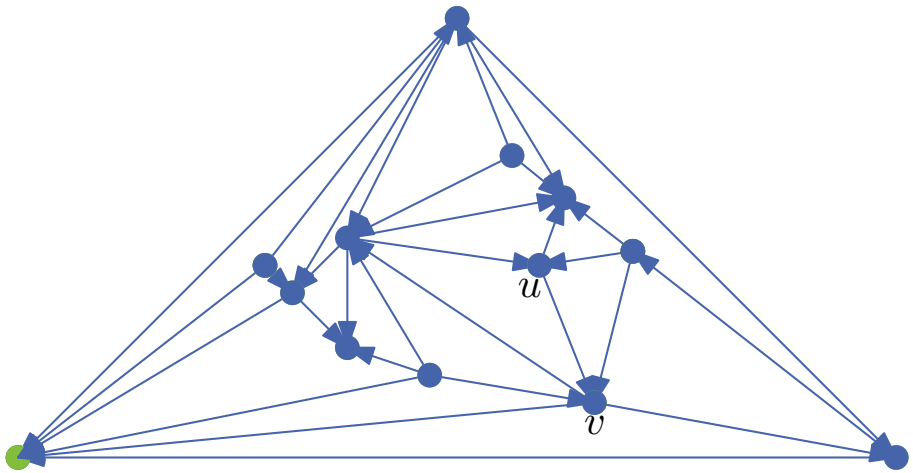
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



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

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

    $d_u = d_u - 1$

    if $d_u = 5$ **then**

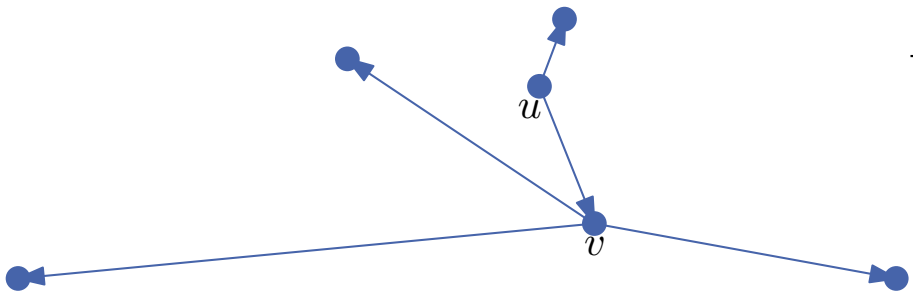
      $Q.push(u)$

    end if

  end for

end while

5 – Adjazenztest



Algorithm ADJAZENT(u, v)

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

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

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