6. Übung Algorithmentechnik

Lehrstuhl für Algorithmik I
Institut für Theoretische Informatik
Universität Karlsruhe (TH)

29.01.2008

Ankündigungen

- letztes Übungsblatt erst nächste Woche (6.KW)
- keine Vorlesung nächsten Donnerstag (05.02.09)
- Fragestunde
  - letzte Übung am Donnerstag, den 12.02.09
  - Fragen bis Montag, den 09.02.09, per Mail an Reinhard (rbauer@ira.uka.de)
- Prüfungsanmeldung
  - Anmeldung/Abmeldung bis 13.02. (online per Selbstverwaltungsfunktion)
  - danach Abmeldung bis 5 Minuten vor der Klausur nur noch persönlich

Veranstaltungen im kommenden Semester

Vorlesungen

- Algorithmen für Ad-hoc- und Sensornetze
- Algorithmen für planare Graphen
- Algorithmen für Routenplanung
- Algorithmen zur Visualisierung von Graphen

Seminar

- Parametrisierte Algorithmen für NP-schwere Probleme
  $\rightsquigarrow$ FPT

Praktikum

- Praktikum zum ICPC-Programmier-Wettbewerb

Simplex-Algorithmus für Flussproblem

Simplex- "Datenstruktur"

- gerichteter Graph D, der immer höchstens einen s-t-Weg enthält
- schwarze Kanten sind nur vorwärts benutzbar
- gelbe Kanten sind in beide Richtungen benutzbar
- rote Kanten sind nur rückwärts benutzbar
- Knoten sind gelb markiert (gelber Rand), wenn sie in D von s aus erreichbar sind

Simplex-Schritt

- füge Kante zu D hinzu und führe Markierungs-Update aus
- falls Kante (v, t) hinzugefügt wird, existiert eindeutiger erhöhender Weg in $D \rightarrow$ erhöhe Fluss

Aufgabe 1

Vertex Cover

- Gegeben: ungerichteter Graph $G = (V, E)$
- Gesucht: Teilmenge $C \subseteq E$, so dass für alle $\{u, v\} \in E$ mindestens einer der Knoten u, v in C ist

- $2 - \frac{\log \log |V|}{2 \log |V|}$-approximierbar
- nicht 1.1666-approximierbar

Aufgabe 1

Algorithmus 1 : VERTEX-COVER-APPROX-1(G)

```
1 C ← ∅
2 E' ← E[G]
3 solange E' ≠ ∅ tue
4 |   e ← {u, v} beliebige Kante aus E'
5 |   C ← C ∪ {u, v}
6 |   entferne alle Kanten inzident zu u, v aus E'
7 return C
```

(a) Zeigen Sie, dass Algorithmus 1 ein 2-Approximationsalgorithmus ist.

Aufgabe 1

- sei C^* optimales Vertex-Cover, C das Cover des Algorithmus
- Algorithmus wählt in jedem Schritt eine Kante, die noch nicht überdeckt ist, und nimmt beide Endknoten ins Cover
- Kanten, die durch die Endknoten überdeckt werden, werden entfernt
- in jedem Schritt gilt $u \in C^*$ oder $v \in C^*$
- damit gilt $|C| \leq 2|C^*|$

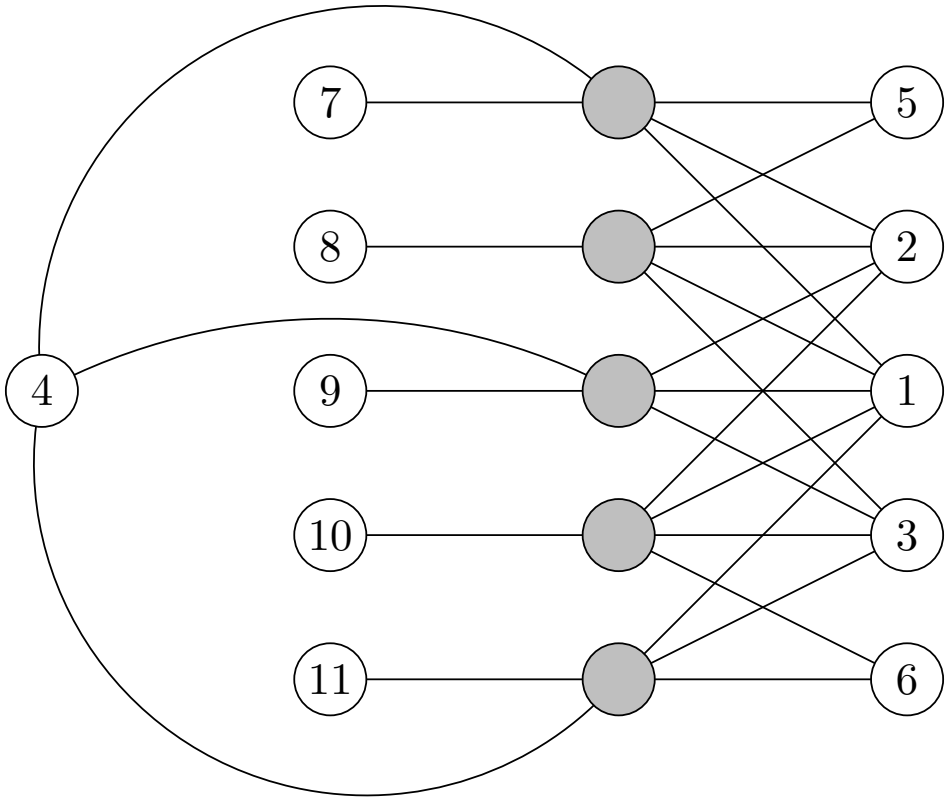
Aufgabe 1

Algorithmus 2 : VERTEX-COVER-APPROX-2(G)

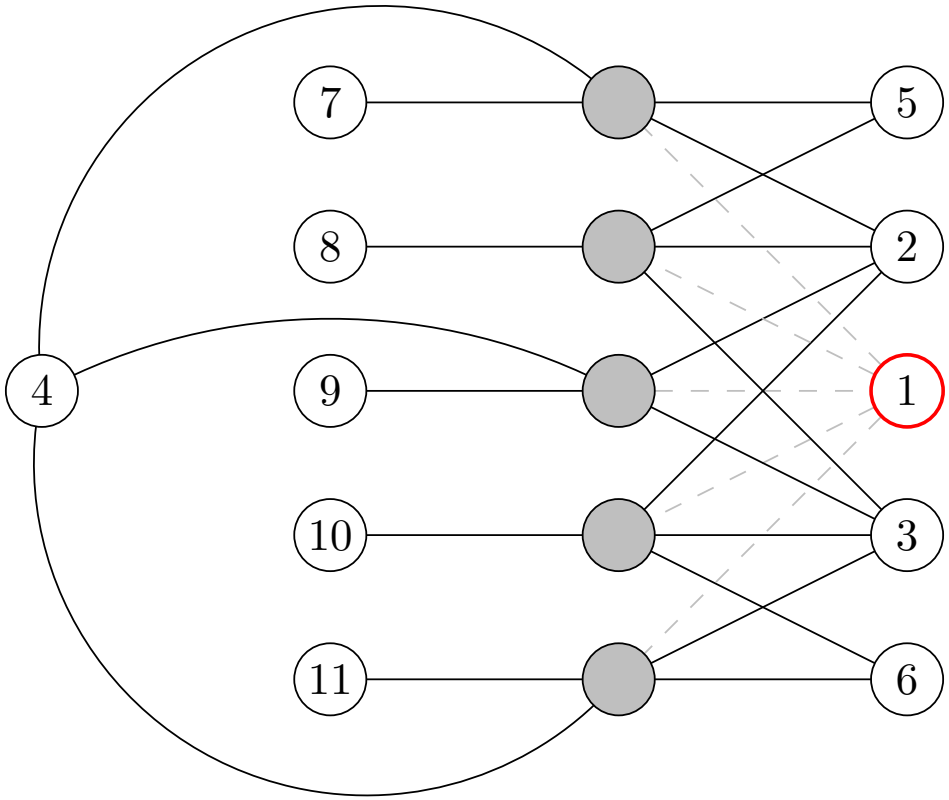
```
1 C ← ∅
2 E' ← E[G]
3 solange E' ≠ ∅ tue
4 |   v ← Knoten mit maximalem Grad in G' = (V \ C, E')
5 |   C ← C ∪ {v}
6 |   entferne alle Kanten inzident zu v aus E'
7 return C
```

(b) Zeigen Sie mit Hilfe eines Gegenbeispiels, dass Algorithmus 2 kein 2-Approximations-algorithmus ist.

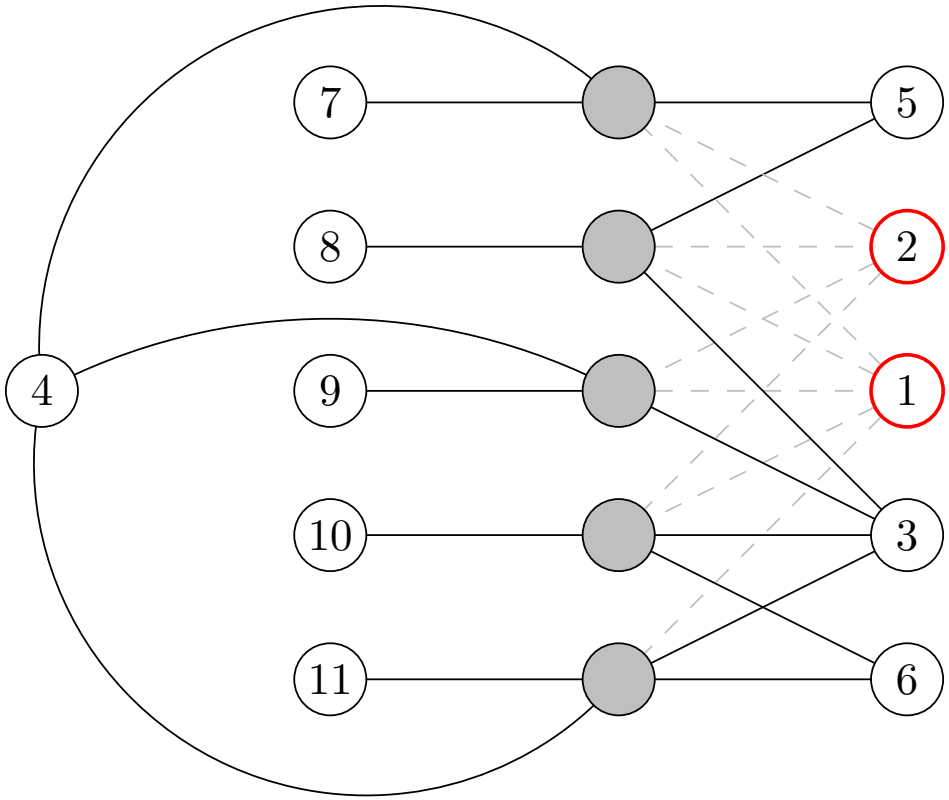
Aufgabe 1



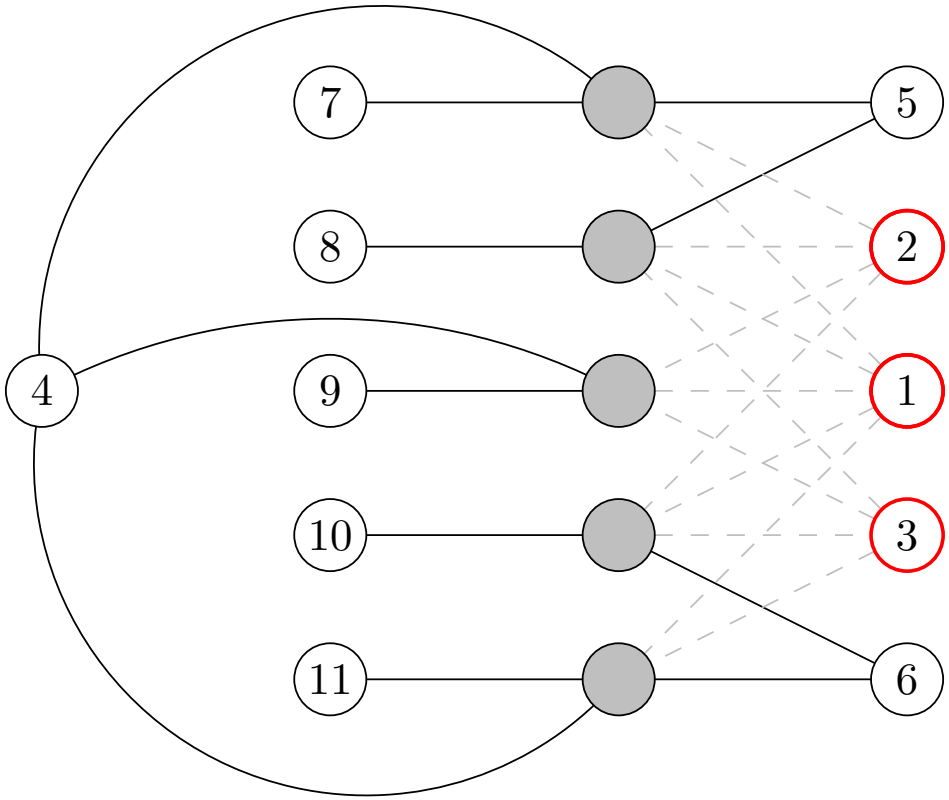
Aufgabe 1



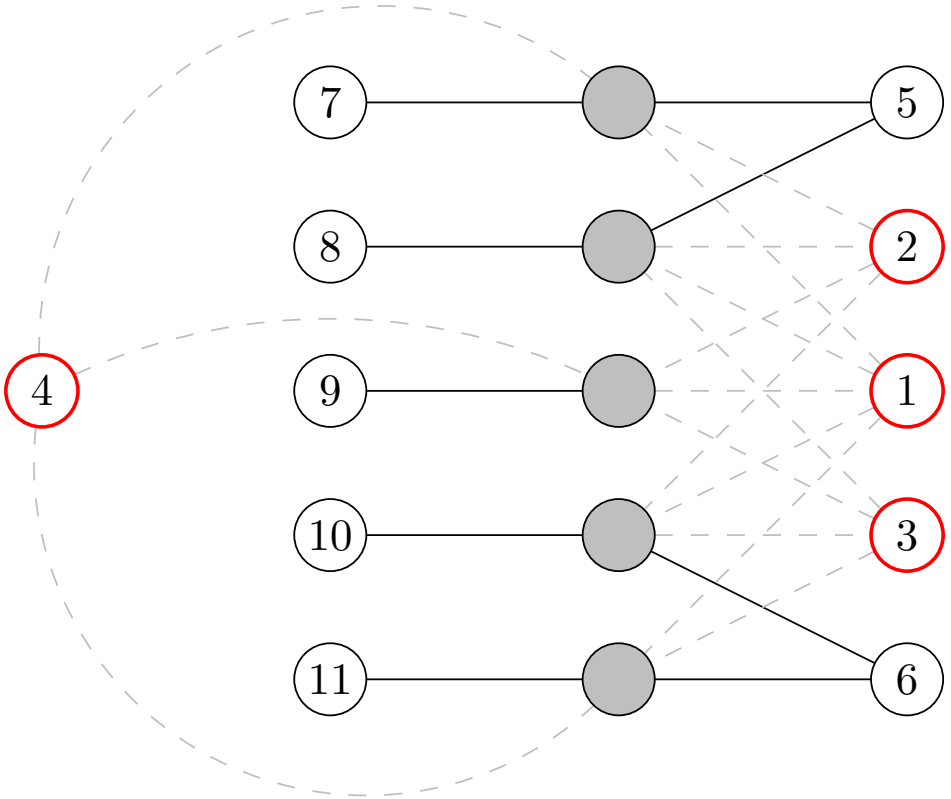
Aufgabe 1



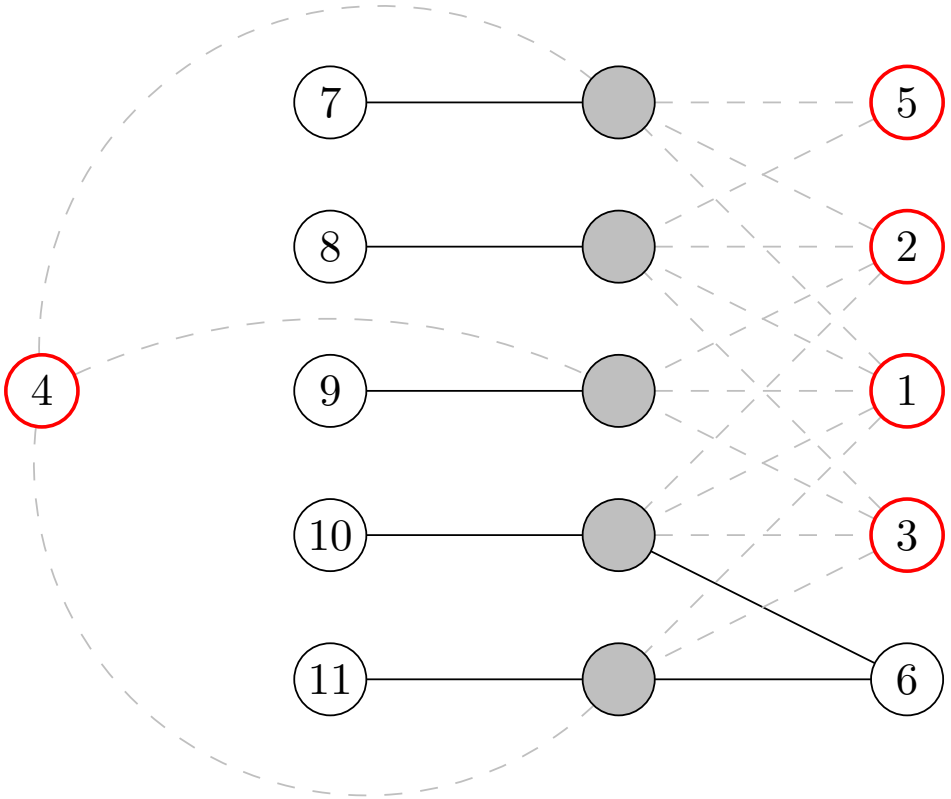
Aufgabe 1



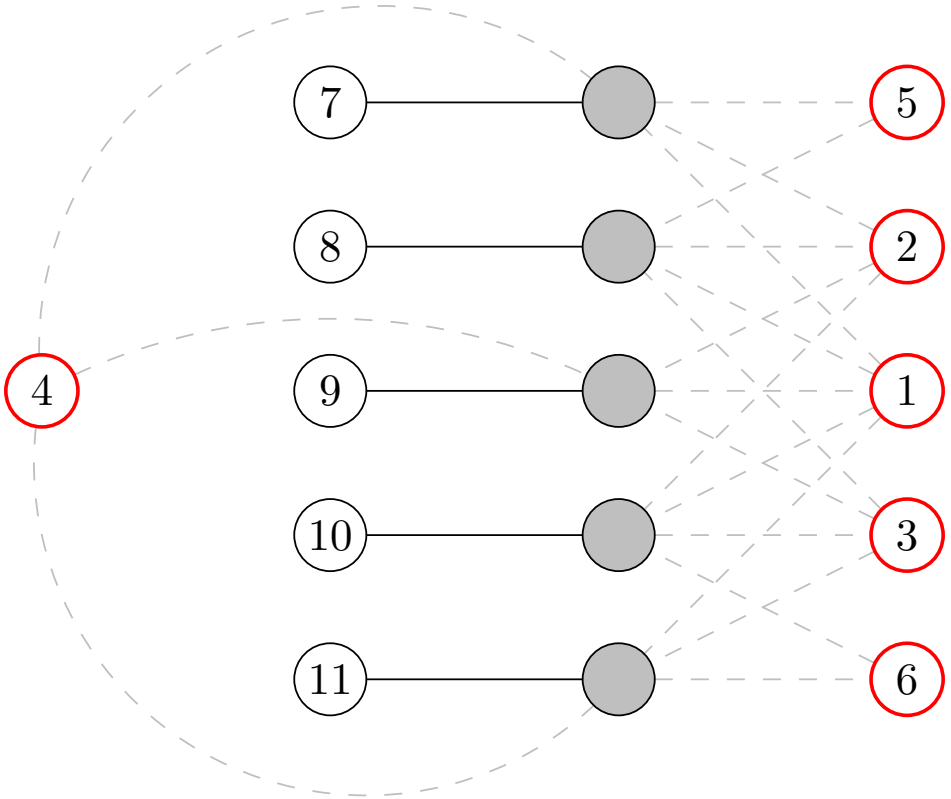
Aufgabe 1



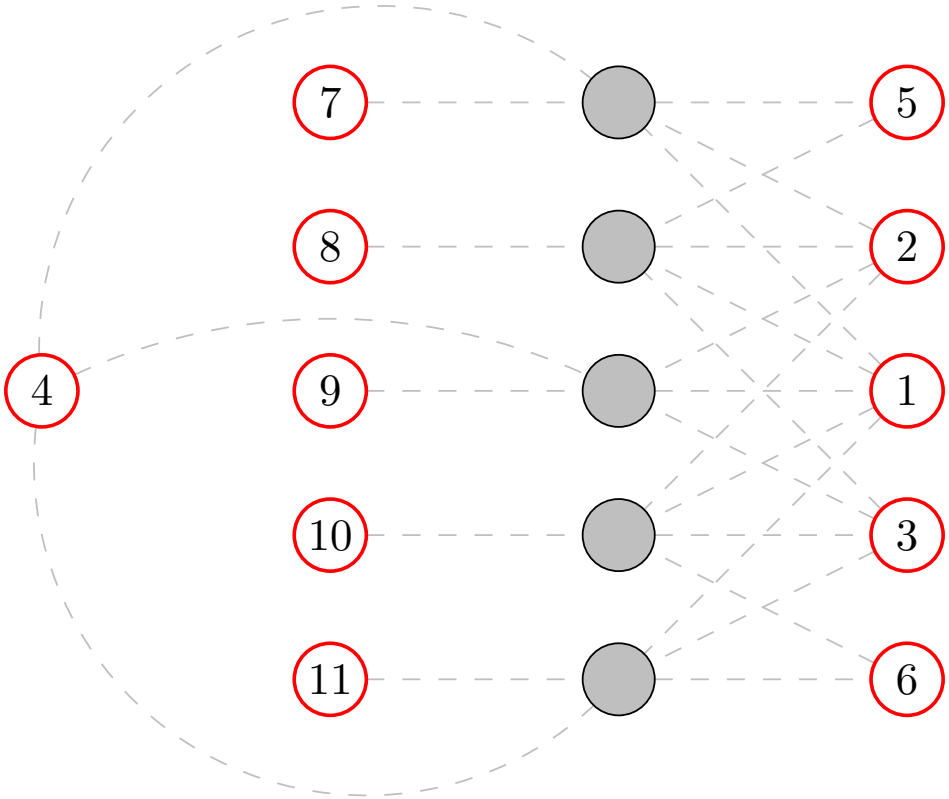
Aufgabe 1



Aufgabe 1



Aufgabe 1



Aufgabe 1

≫ Geben Sie einen effizienten Greedy-Algorithmus an, der in linearer Zeit (bezogen auf $|E|$) ein optimales Vertex Cover in einem Baum findet.

Aufgabe 1

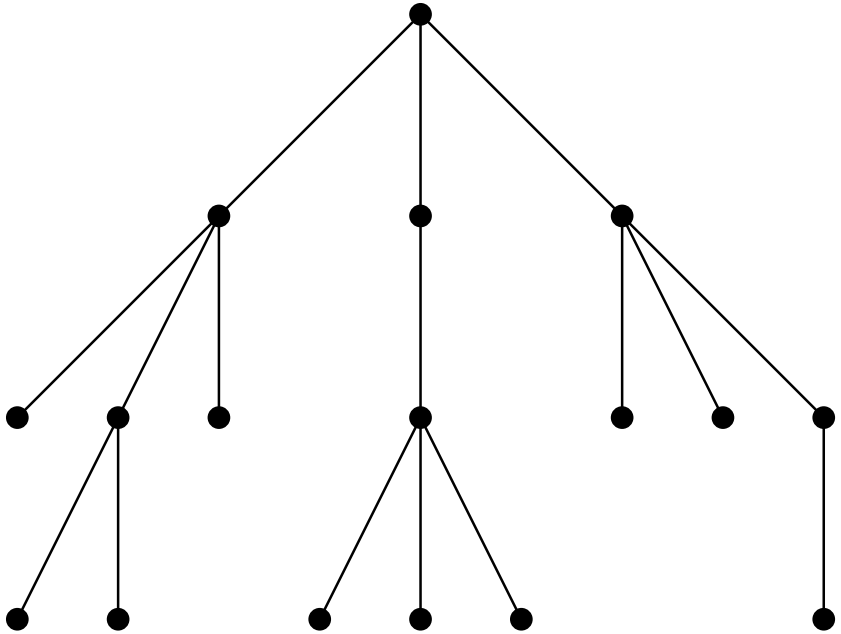
Algorithmus 3 : Optimales Vertex Cover für Bäume

Eingabe : Baum $G = (V, E)$
Ausgabe : Optimales Vertex Cover

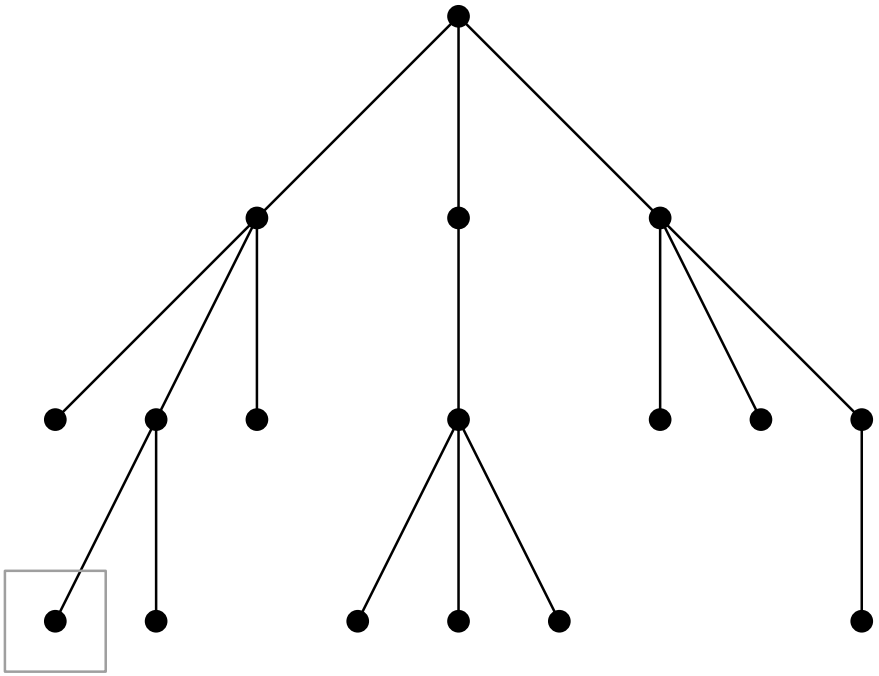
```
1 C ← ∅
2 solange |E| ≥ 1 tue
3     v ← Blattknoten in G
4     w ← Zu v adjazenter Knoten in G
5     C ← C ∪ {w}
6     Lösche w und alle zu w inzidenten Kanten
7 return C
```

≫ Wie kann man den Algorithmus mit linearer Laufzeit implementieren?

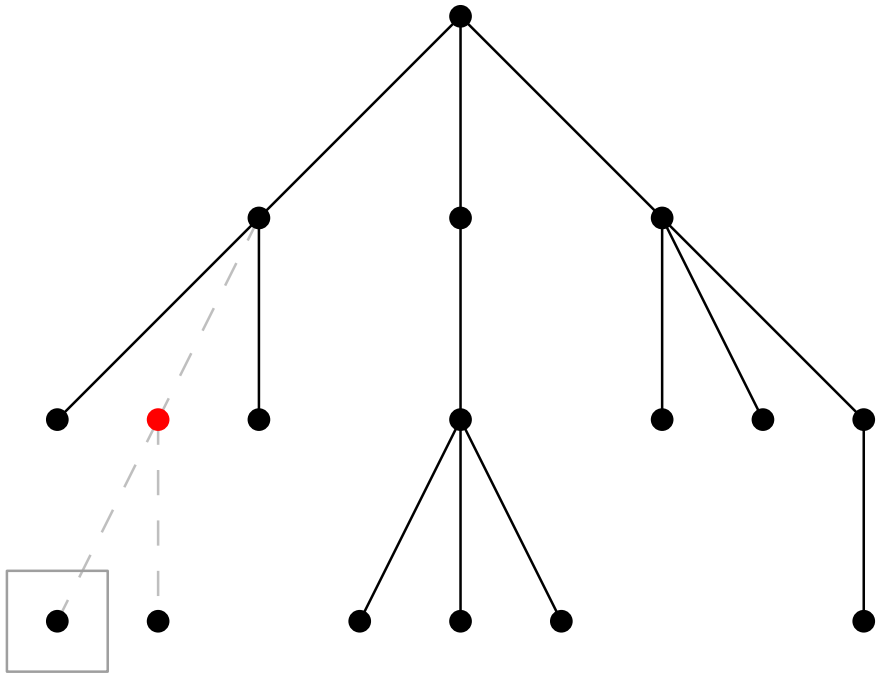
Aufgabe 1



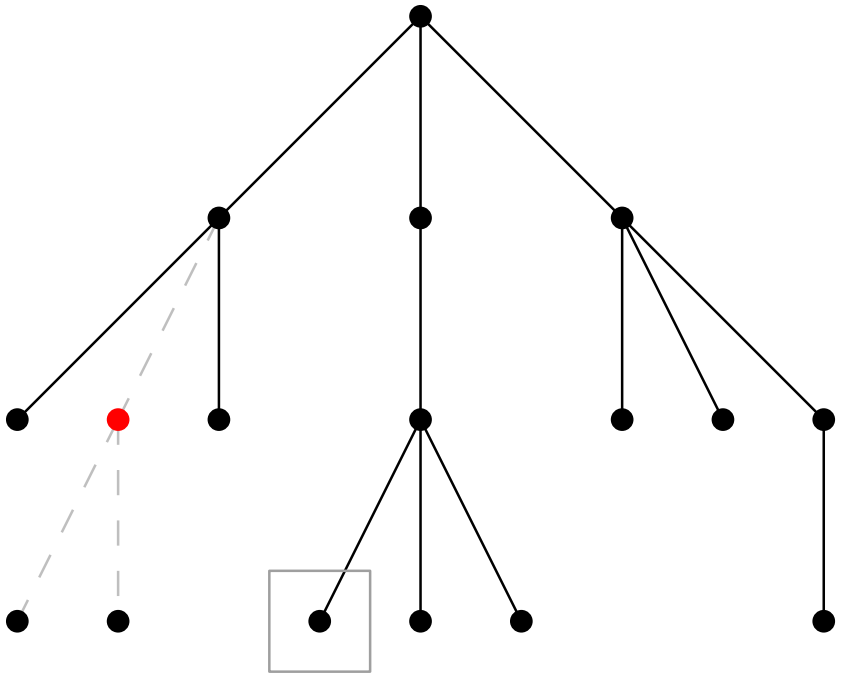
Aufgabe 1



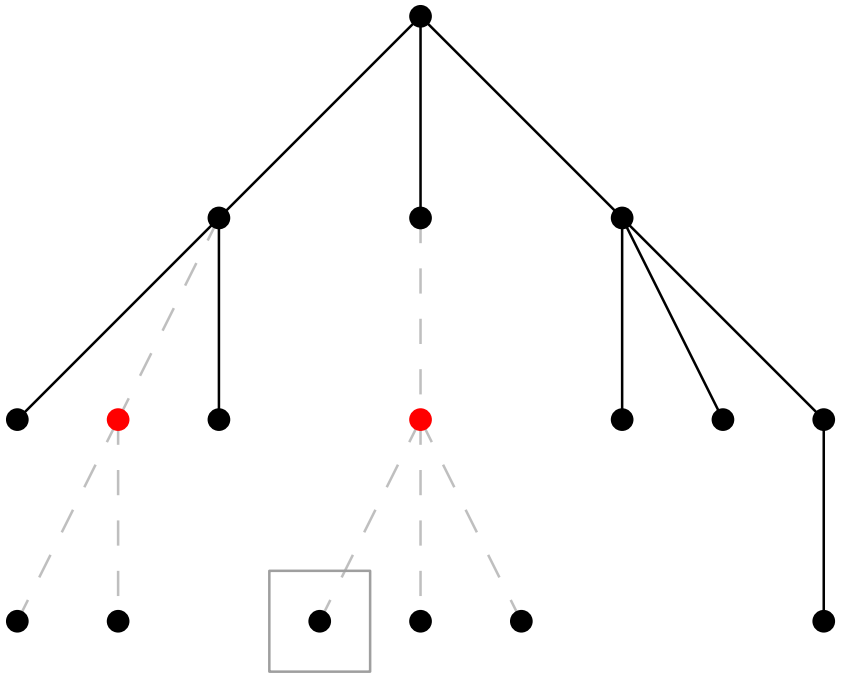
Aufgabe 1



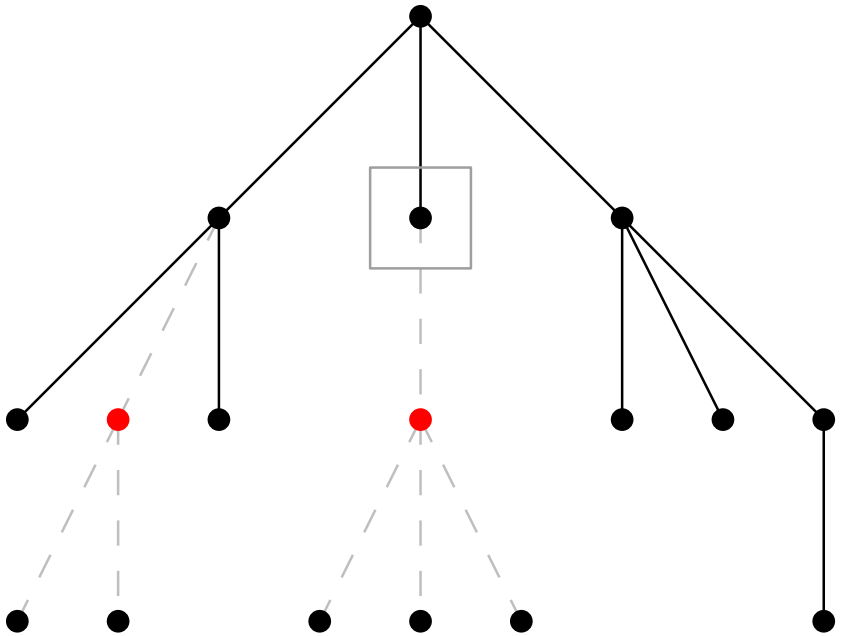
Aufgabe 1



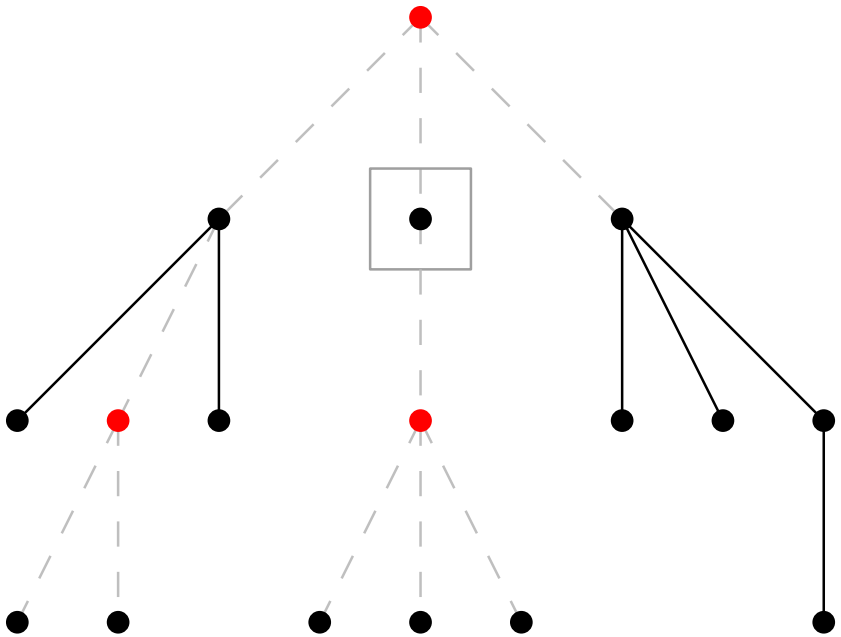
Aufgabe 1



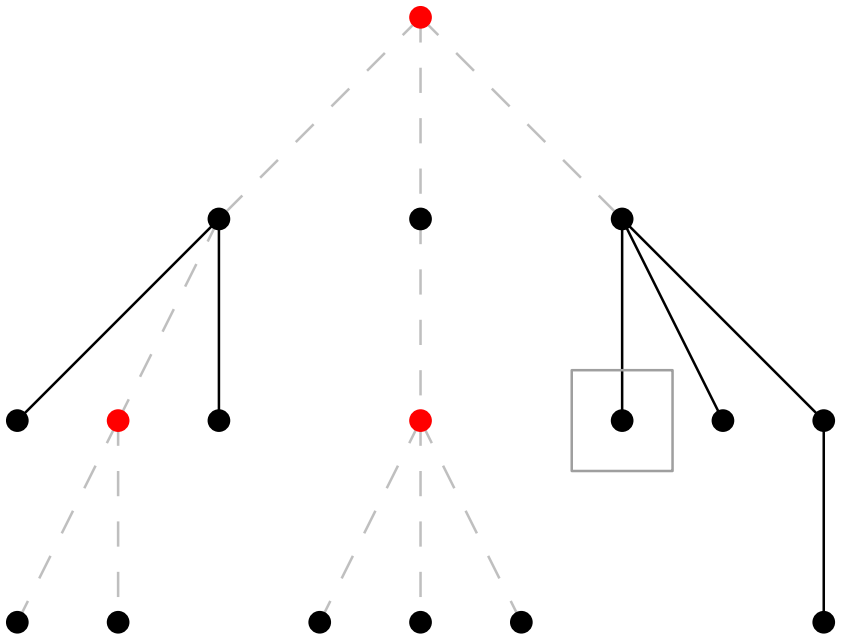
Aufgabe 1



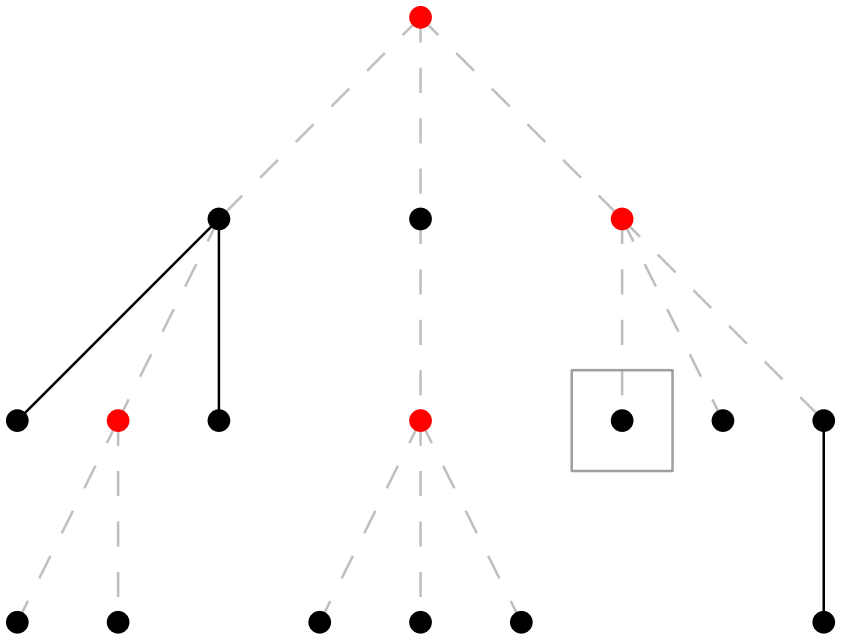
Aufgabe 1



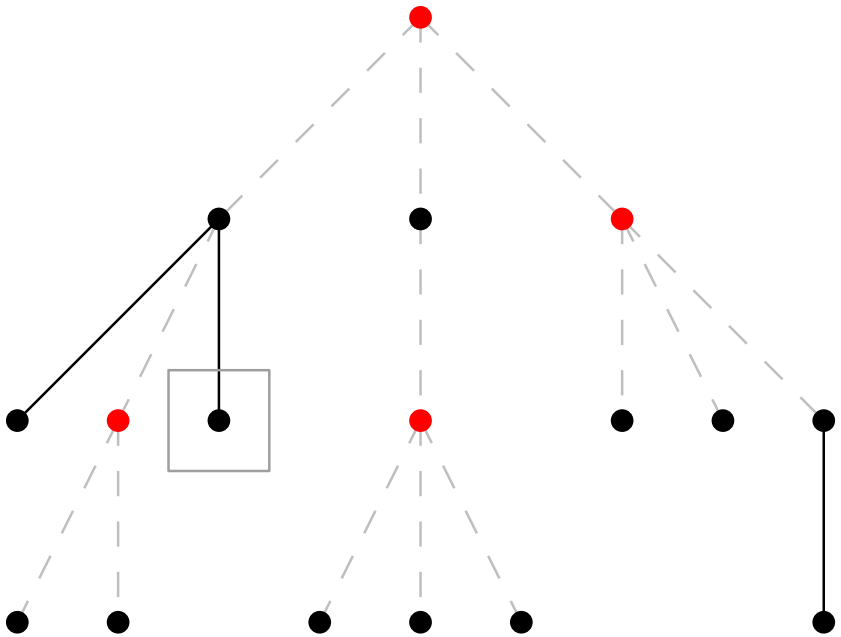
Aufgabe 1



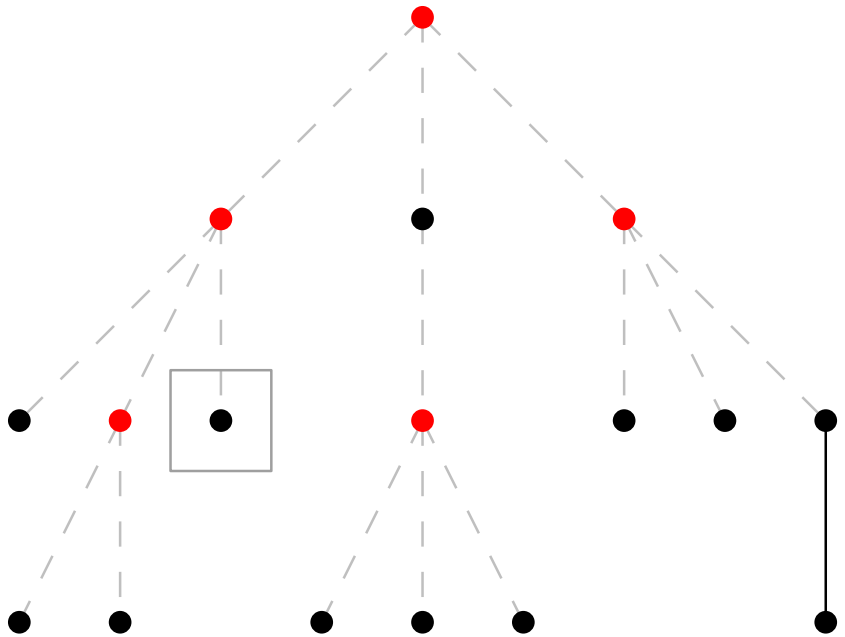
Aufgabe 1



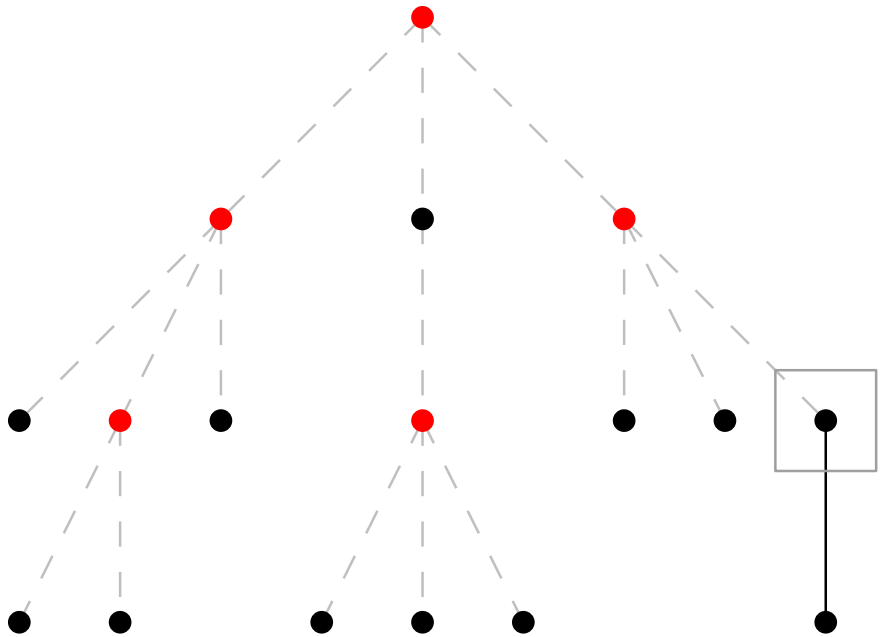
Aufgabe 1



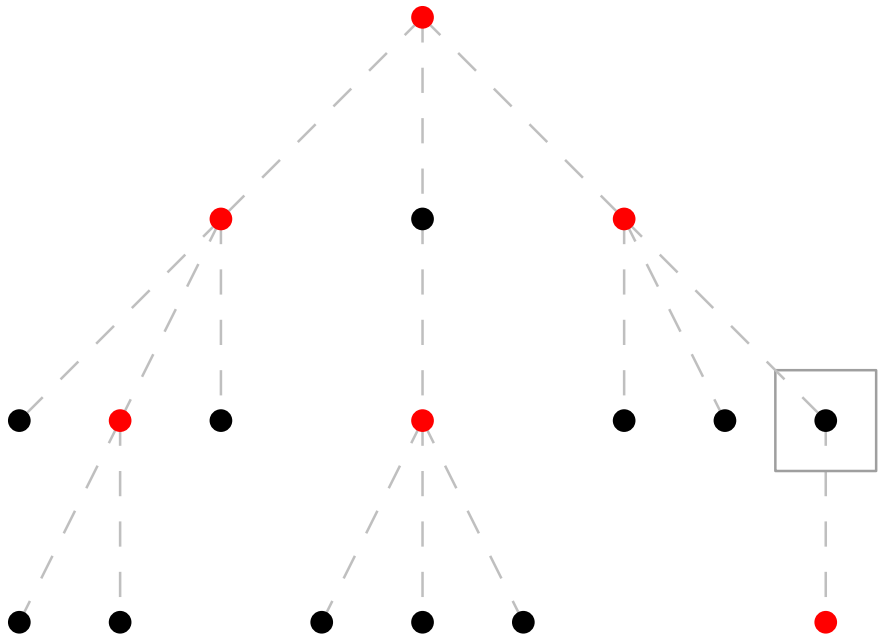
Aufgabe 1



Aufgabe 1



Aufgabe 1



Aufgabe 1

Optimalität

- Induktion über Anzahl m der Kanten
- Induktionsanfang: optimal für $m \leq 1$
- Induktionsschritt: $m > 1$
- wähle blatt v und inzidente Kante $\{v, w\}$ (die nicht überdeckt ist)
- Induktionsvoraussetzung $\Rightarrow$ Algorithmus findet optimale Lösung für $G - w$ mit c Knoten
- Annahme: G besitzt ebenfalls Vertex-Cover mit c Knoten
- sei C VC von G mit c Knoten $\Rightarrow$ mindestens einer der Knoten v, w muss in C enthalten sein

$\Rightarrow$ $G - w$ besitzt ein VC der Größe $c - 1$ im Widerspruch zur IV

Aufgabe 2

Euklidischer Handlungsreisender

- Gegeben: n Punkte $P = \{p_1, \ldots, p_n\}$ mit Koordinaten (x_i, y_i) in der Ebene
- Gesucht: kürzeste Rundtour, die jeden Punkt mindestens einmal besucht
- Abstandsmaß d ist euklidische Distanz

- 3/2-approximierbar
- nicht $\frac{3813}{3812} - \varepsilon$-approximierbar $(\varepsilon > 0)$

Aufgabe 2

Bemerkung:

- $\gg$ $d \geq 0$
- $\gg$ d erfüllt die Dreiecksungleichung, d.h. für $u, v, w \in P$ gilt

$$d(u, w) \leq d(u, v) + d(v, w)$$

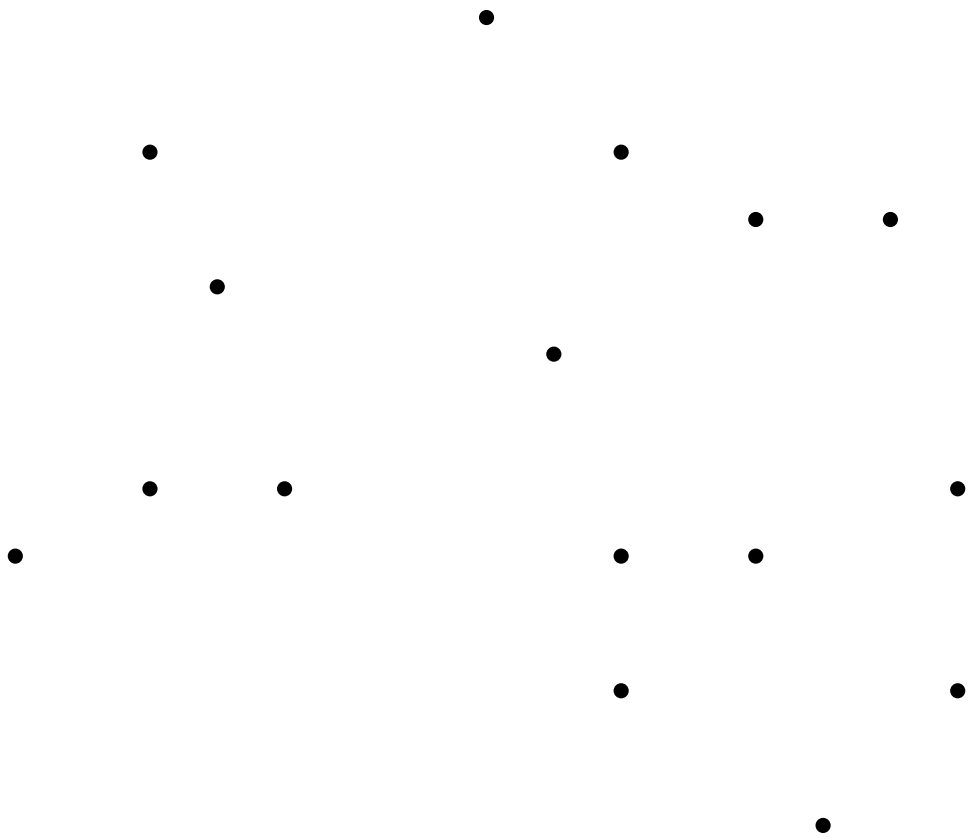
- $\Rightarrow$ für Folge von Knoten $u_1, \ldots, u_k$ gilt

$$d(u_1, u_k) \leq d(u_1, u_2) + d(u_2, u_3) + \cdots + d(u_{k-1}, u_k)$$

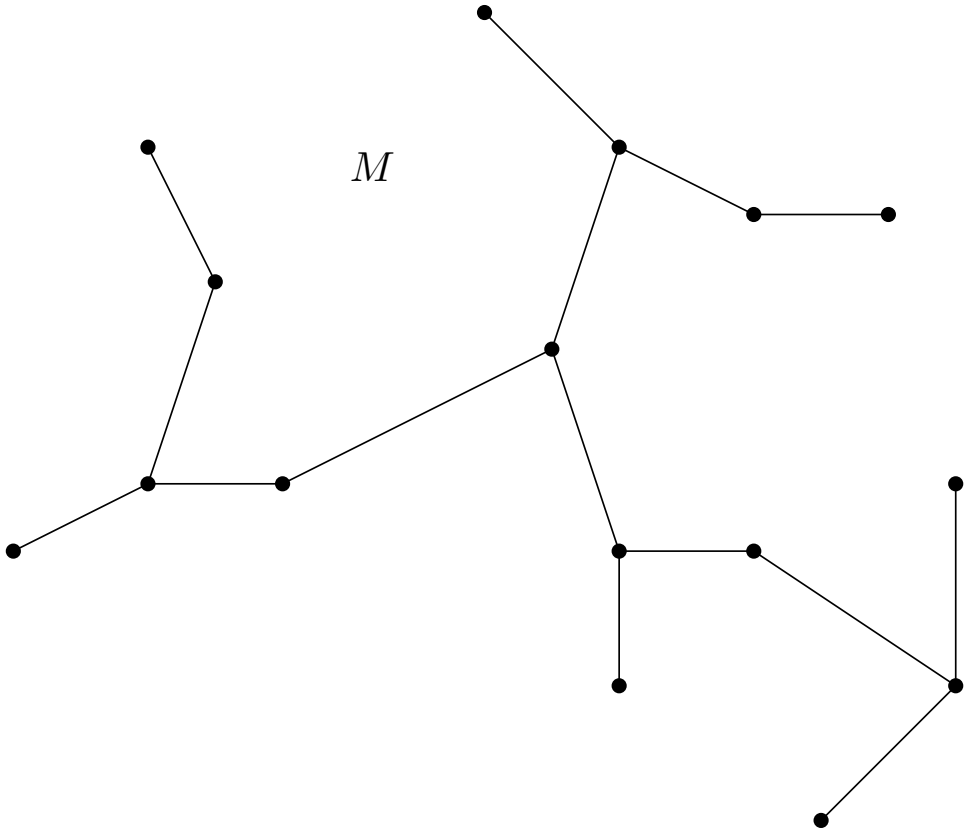
Einfache Approximation für das euklidische Handlungsreisendenproblem

- $\gg$ berechne minimalen Spannbaum M bzgl. d
- $\gg$ Prä- und Postorder Durchlauf definiert Rundtour T', die einige der Knoten mehrfach besucht
- $\gg$ $d(T') = 2d(M)$
- $\gg$ entferne mehrfach vorkommende Knoten $\rightsquigarrow$ Tour T, die jeden Knoten genau einmal enthält
- $\gg$ $d(T) \leq d(T') = 2d(M)$ (wegen Dreiecksungleichung)
- $\gg$ sei T^* optimale Tour, e beliebige Kante auf $T^* \Rightarrow T^* - e$ ist Spannbaum
- $\Rightarrow$ $d(T^*) \geq d(T^* - e) \geq d(M)$
- $\Rightarrow$ $d(T) \leq 2d(M) \leq 2d(T^* - e) \leq 2d(T^*)$

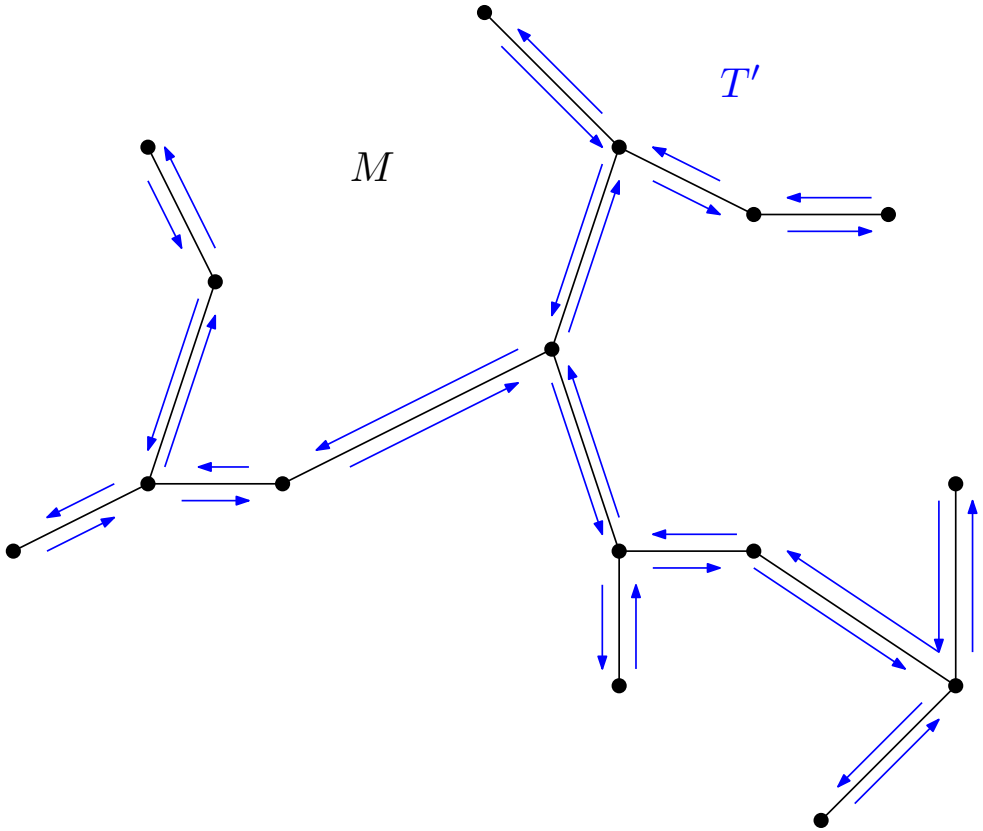
Aufgabe 1



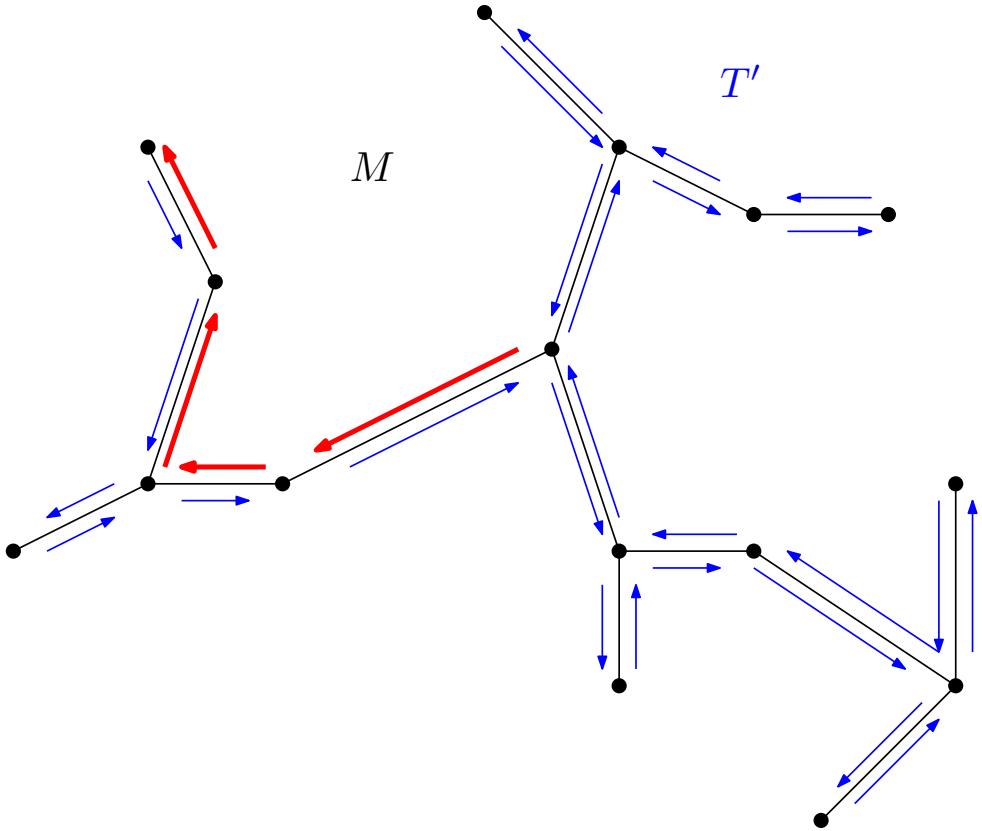
Aufgabe 1



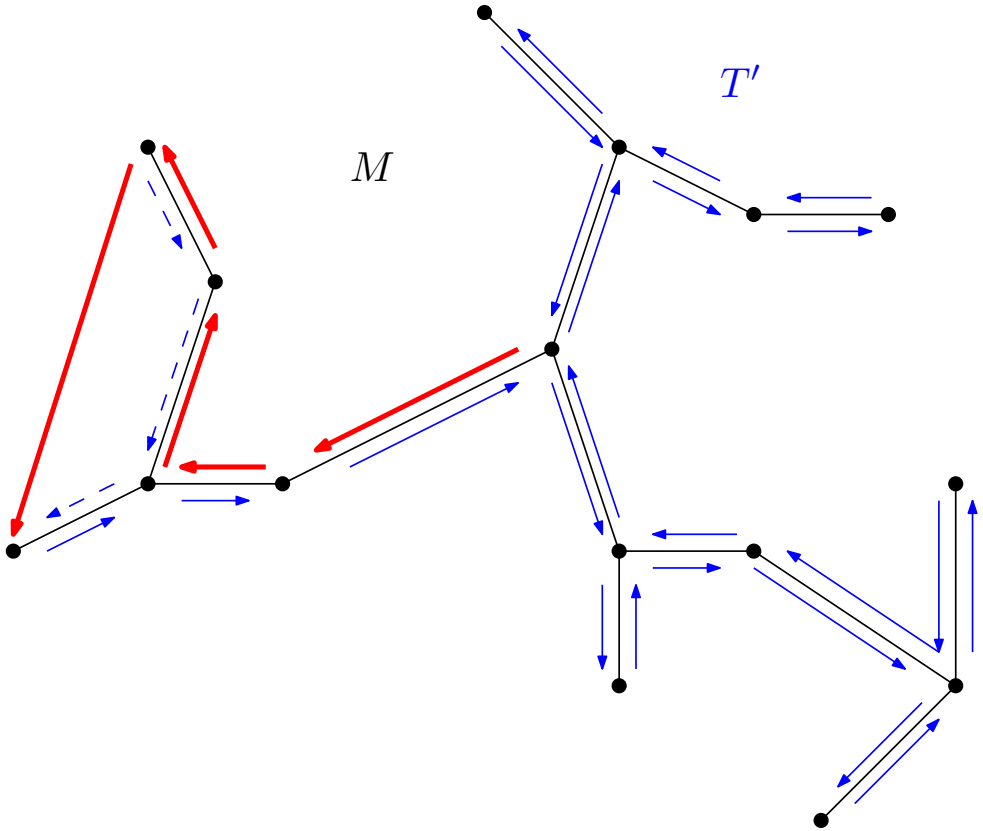
Aufgabe 1



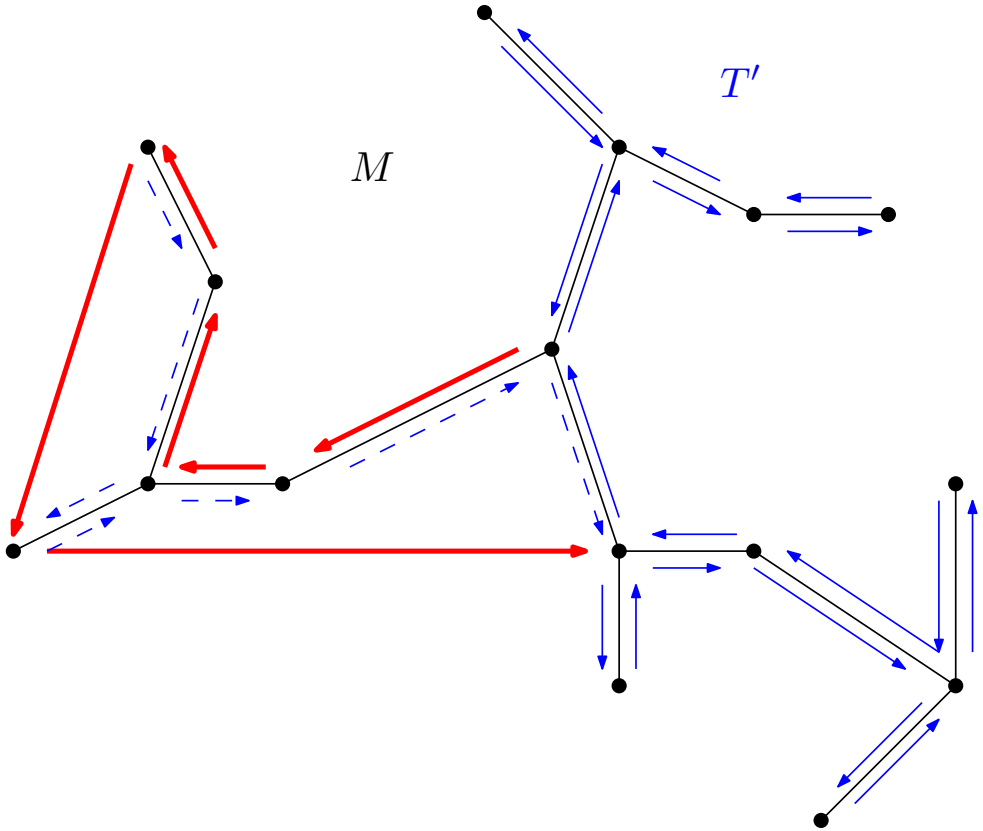
Aufgabe 1



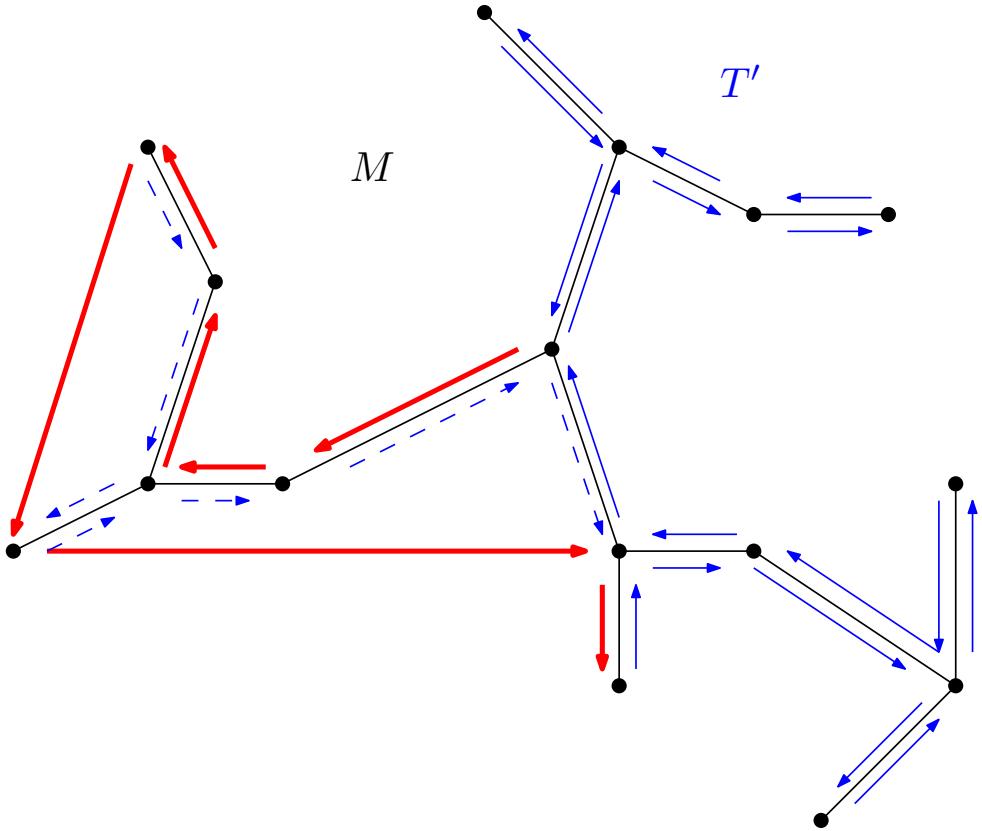
Aufgabe 1



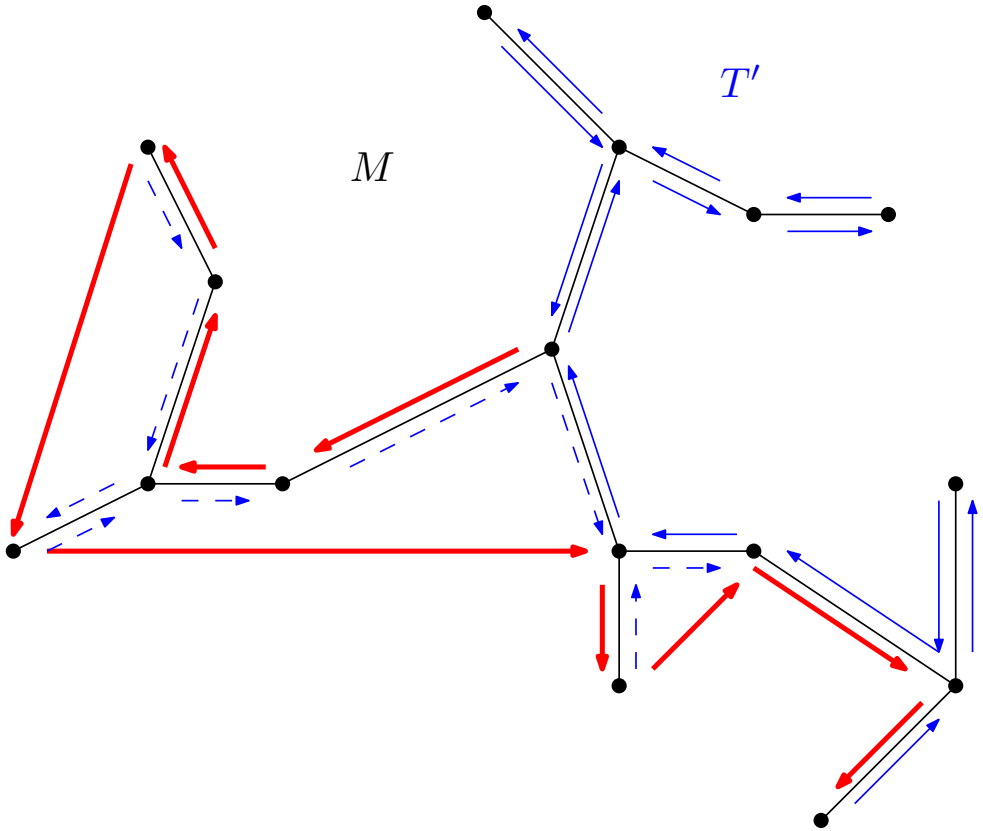
Aufgabe 1



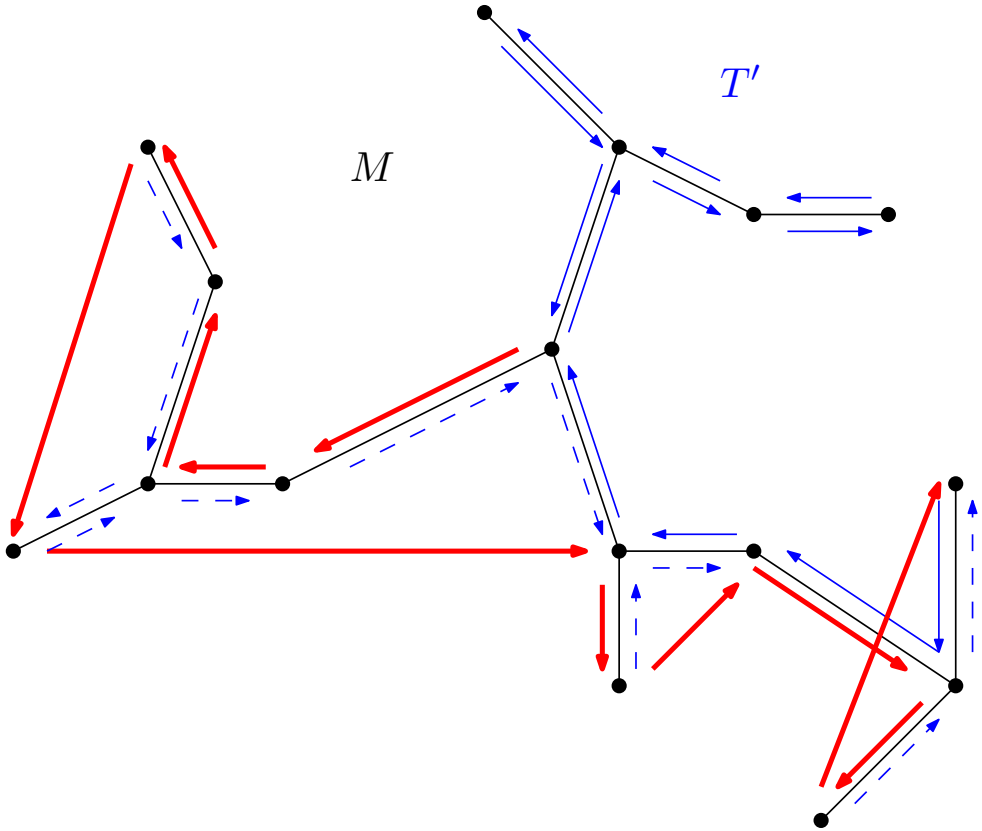
Aufgabe 1



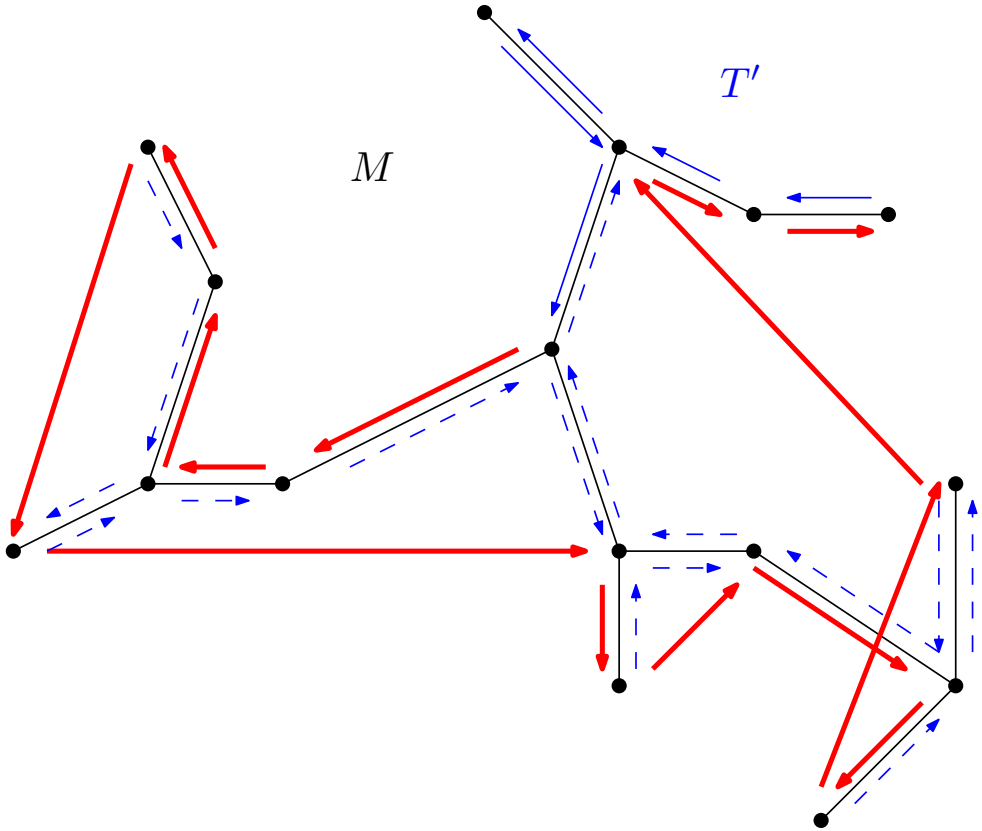
Aufgabe 1



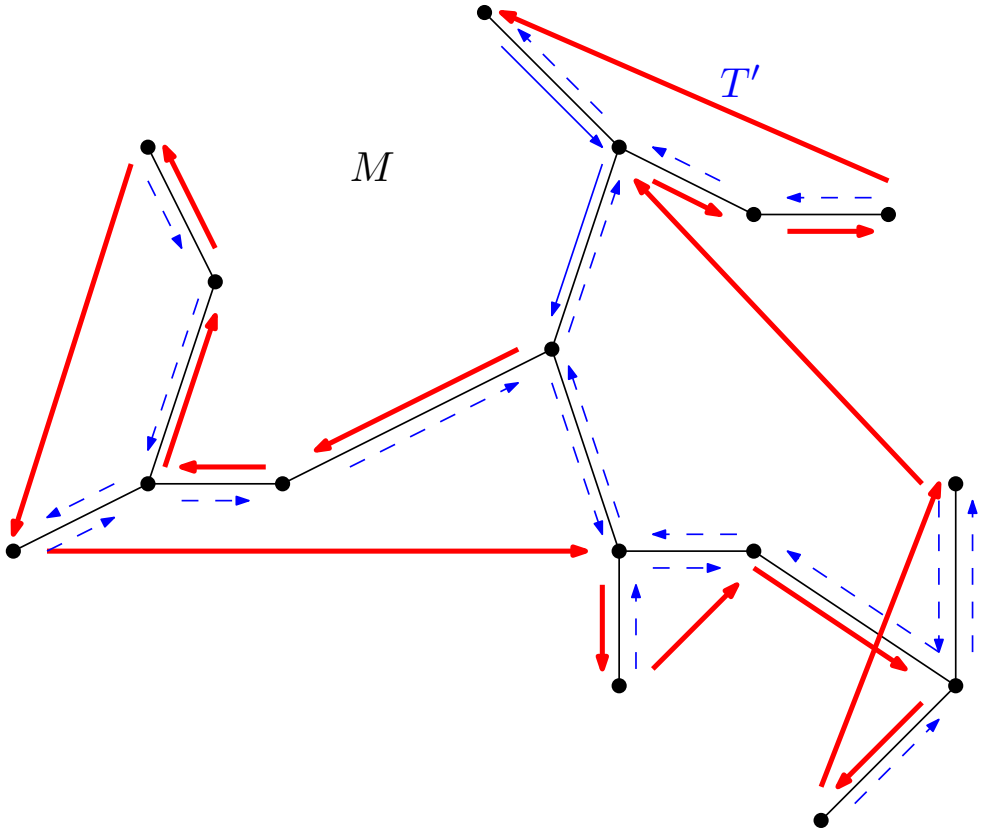
Aufgabe 1



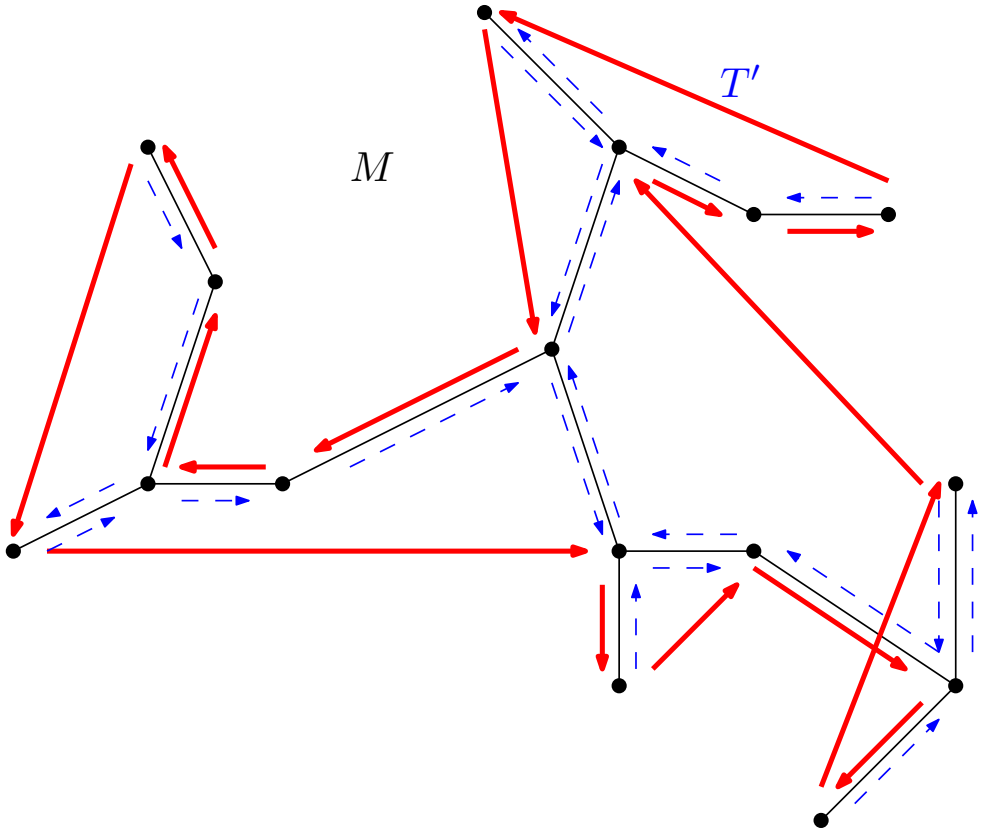
Aufgabe 1



Aufgabe 1



Aufgabe 1



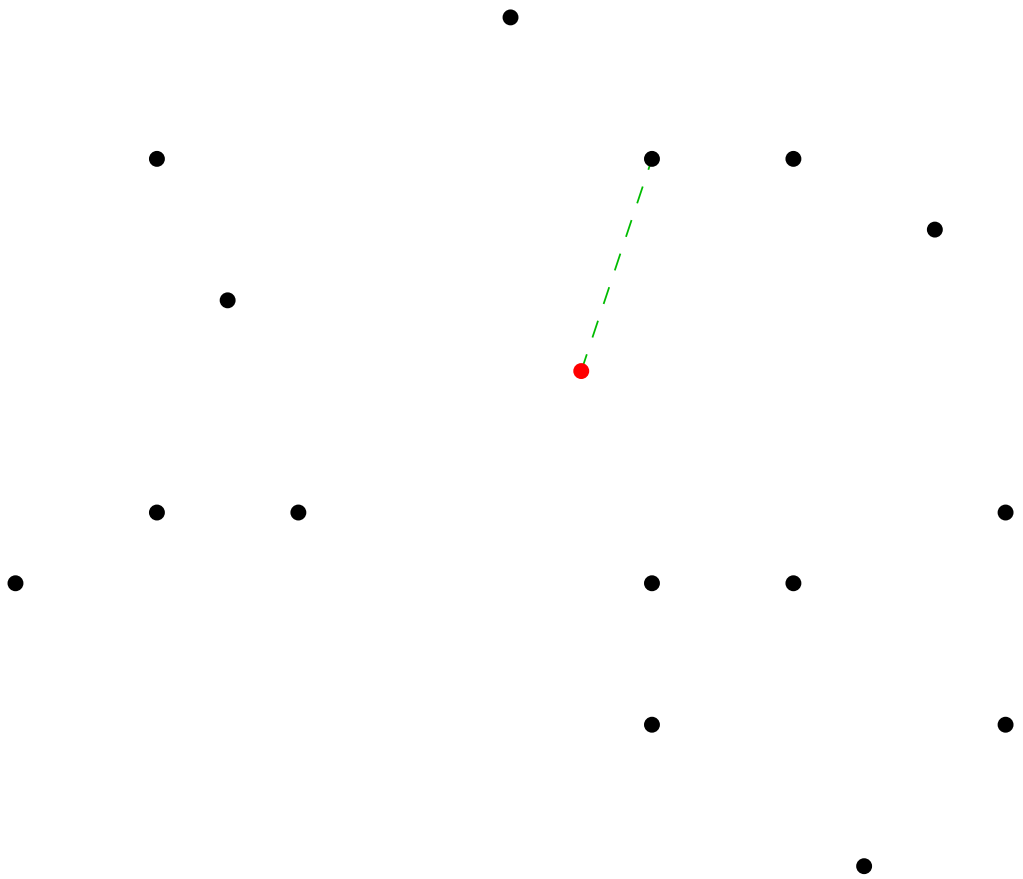
Aufgabe 2

Algorithmus 4 : E-TSP-APPROX-1(G)

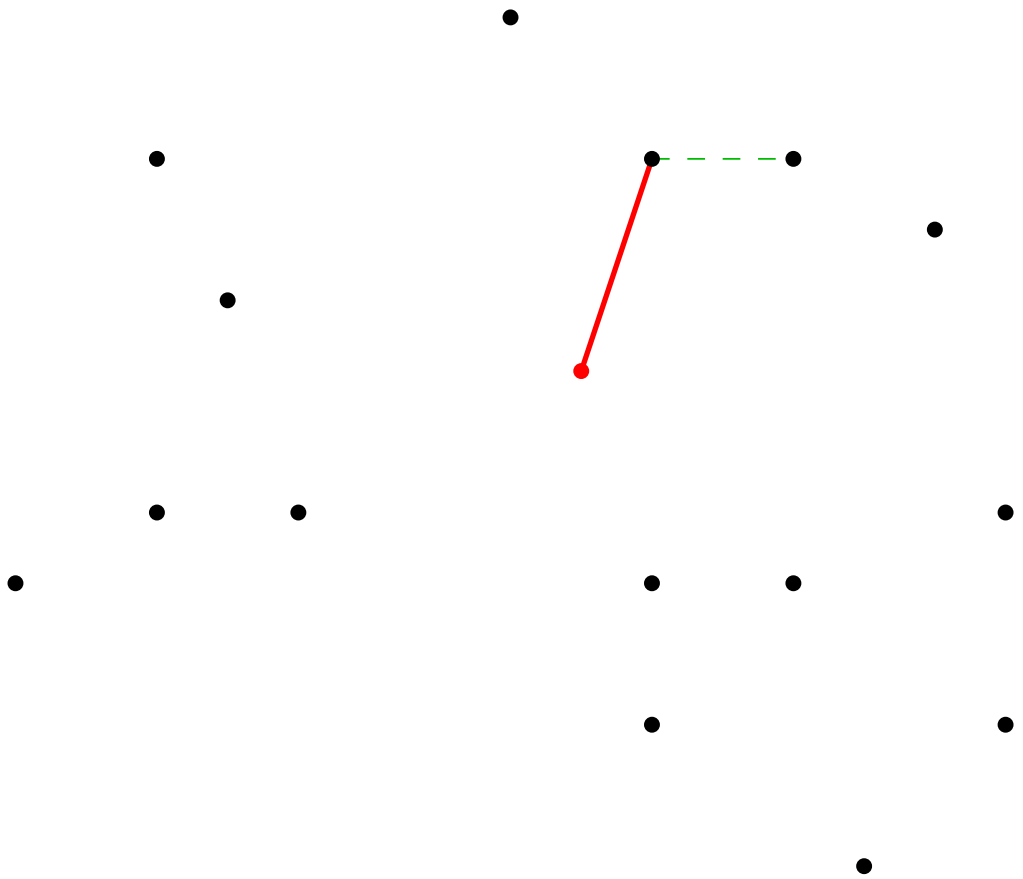
```
1 T ← p_1
2 solange |T| < |V| tue
3     v, p_{i_j} ← Knoten mit d(p_{i_j}, v) minimal, mit p_{i_j} ∈ T, v ∈ V \ T
4     k ← |T|
5     T ← (p_{i_1}, ..., p_{i_j}, v, p_{i_{j+1}}, ... p_{i_k}) //füge v zwischen p_{i_j} und
      p_{i_{j+1}} ein
6     (p_{i_1}, ..., p_{i_{k+1}}) ← T
7 return T
```

≫ Zeigen Sie, dass Algorithmus 4 ein 2-Approximations-algorithmus ist.

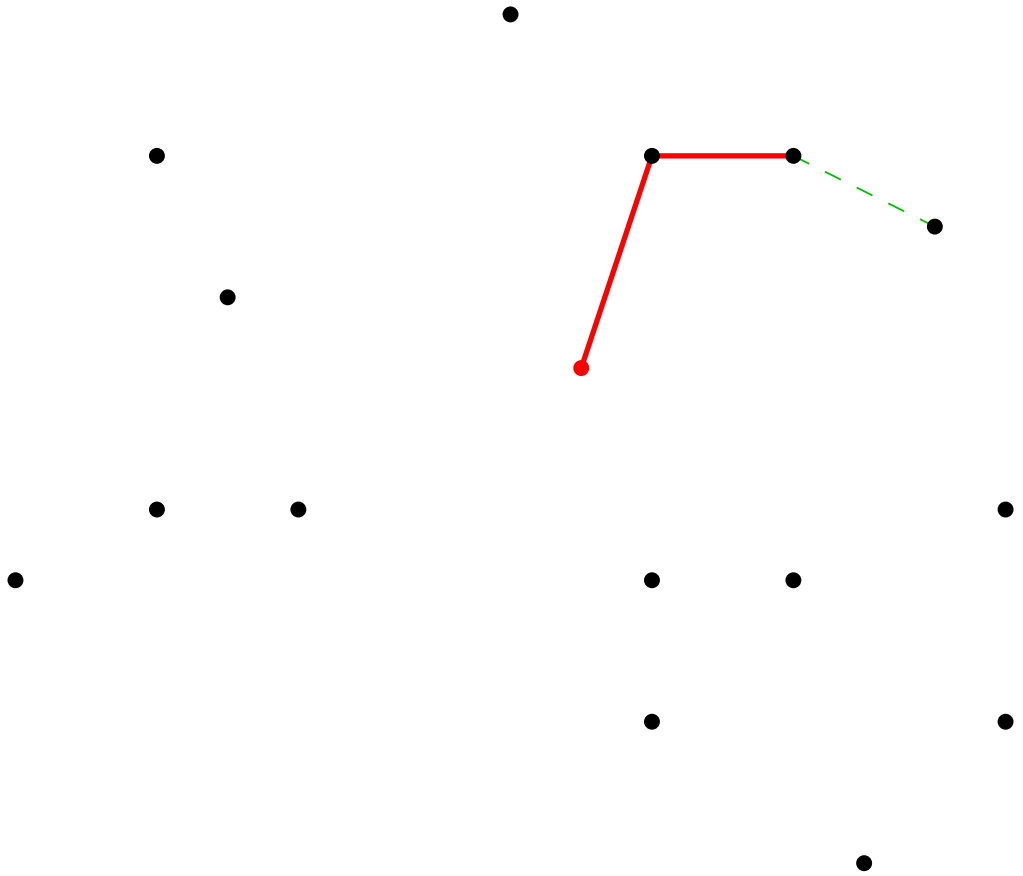
Aufgabe 1



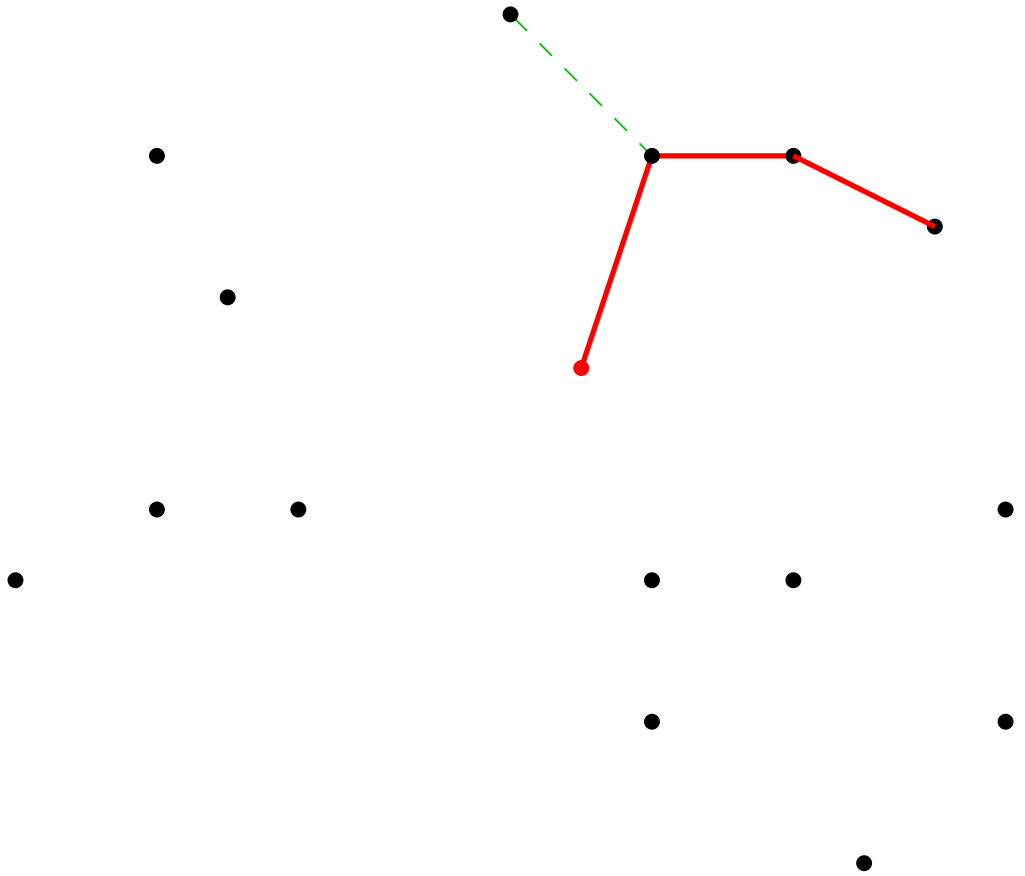
Aufgabe 1



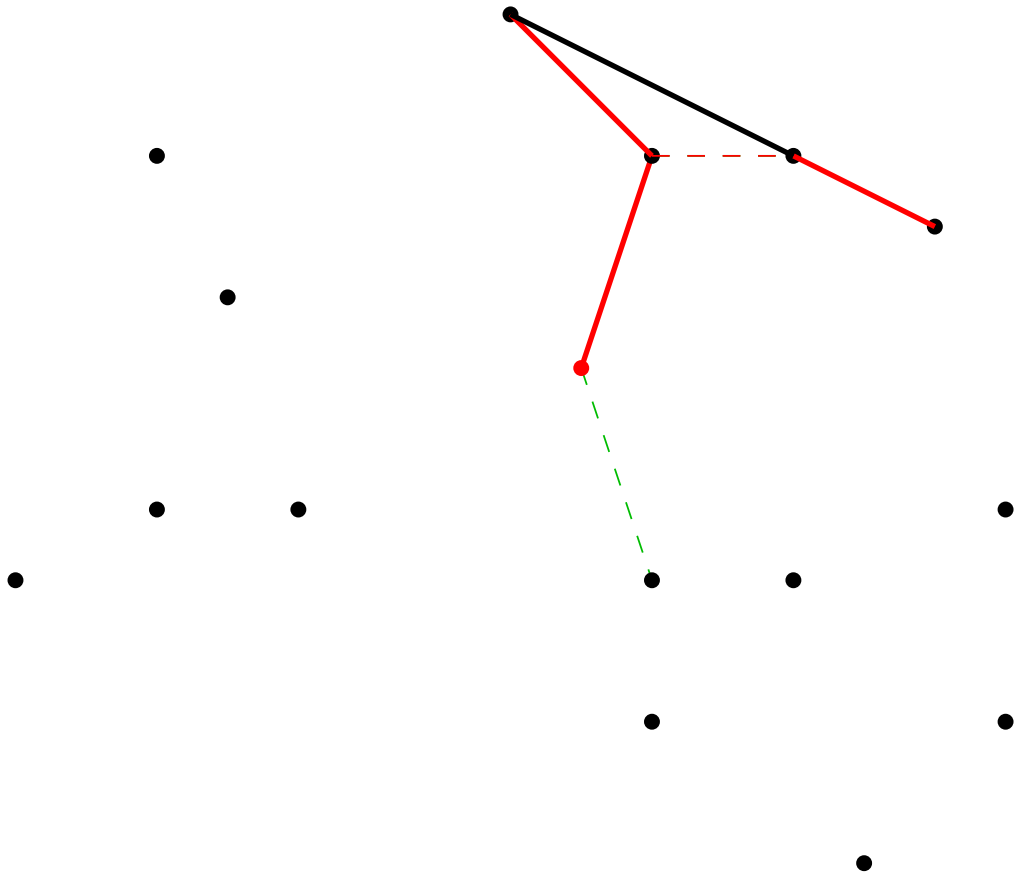
Aufgabe 1



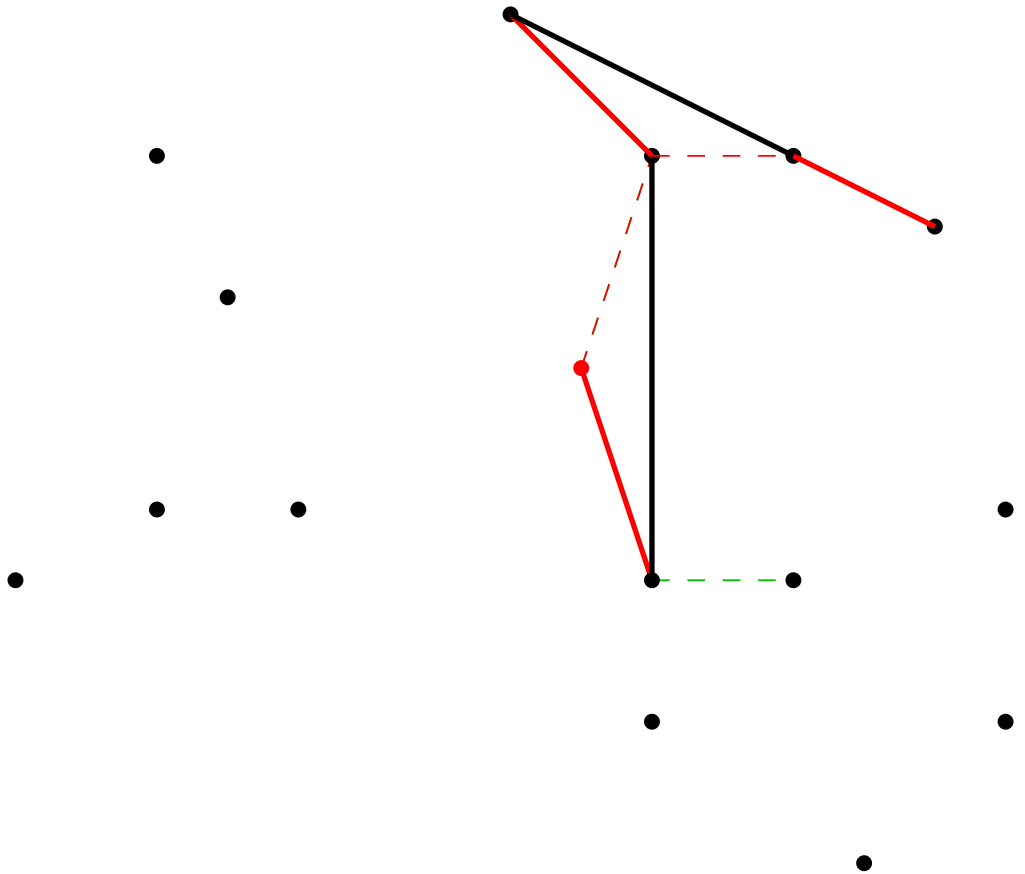
Aufgabe 1



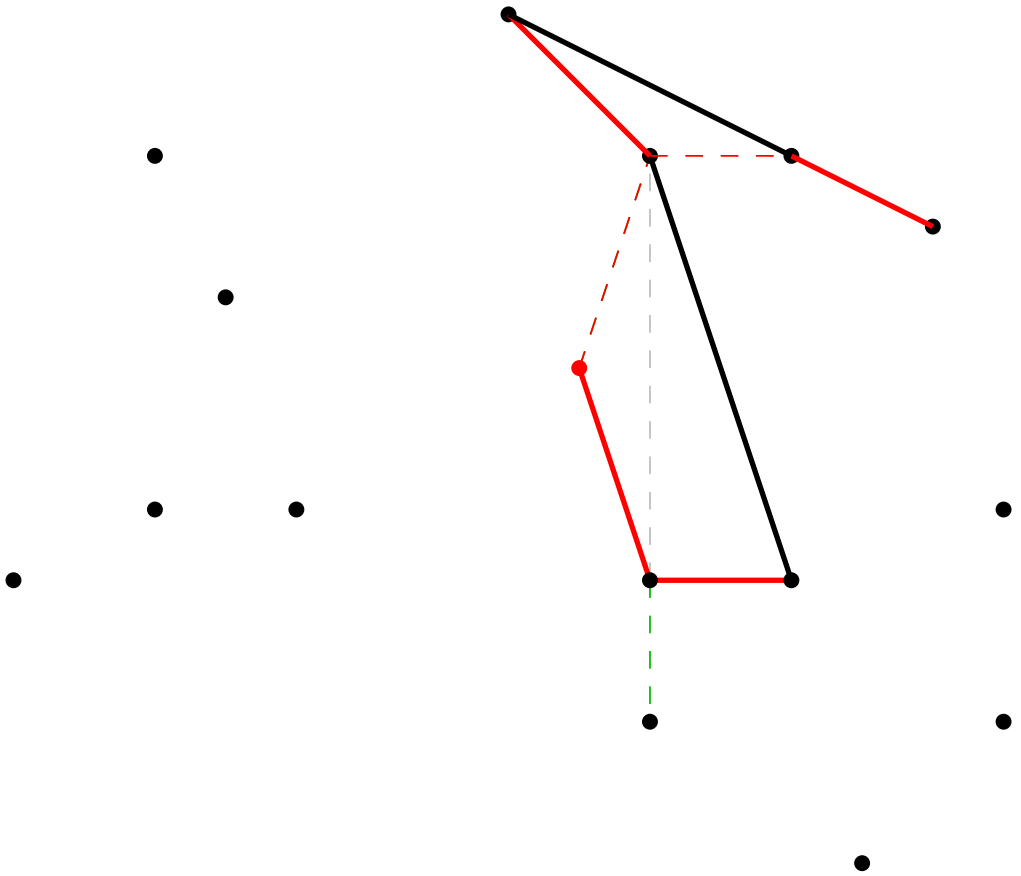
Aufgabe 1



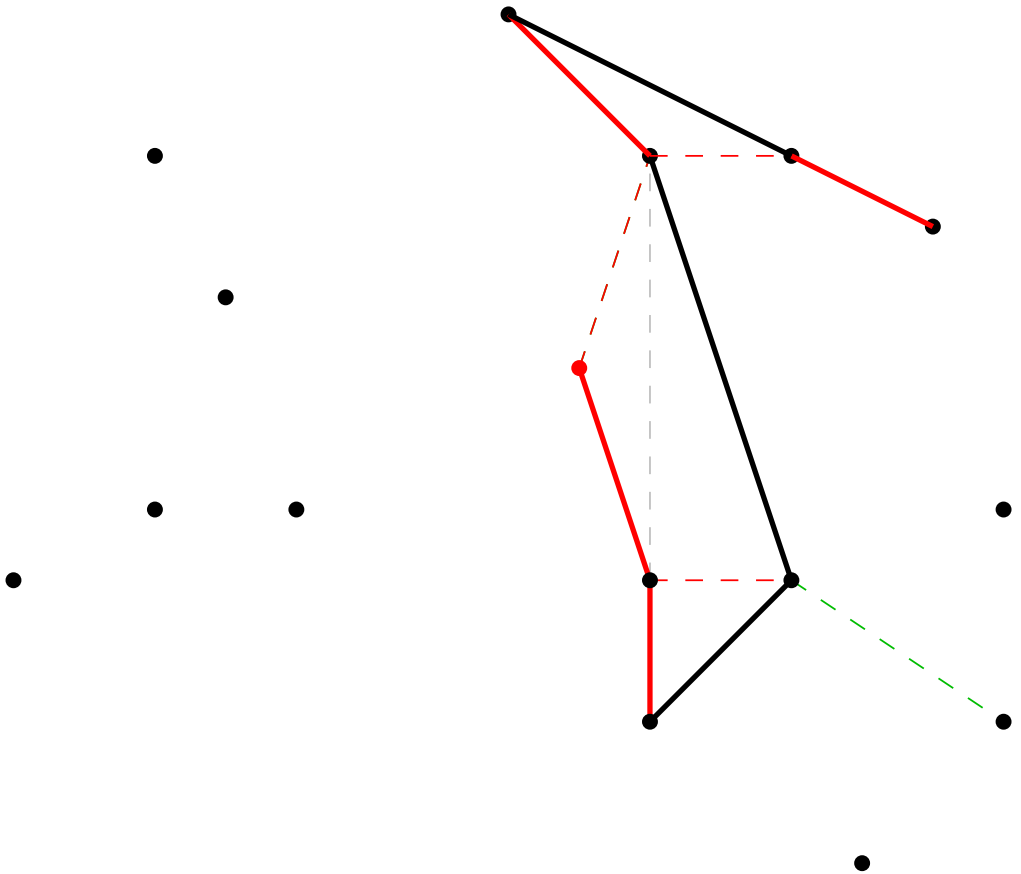
Aufgabe 1



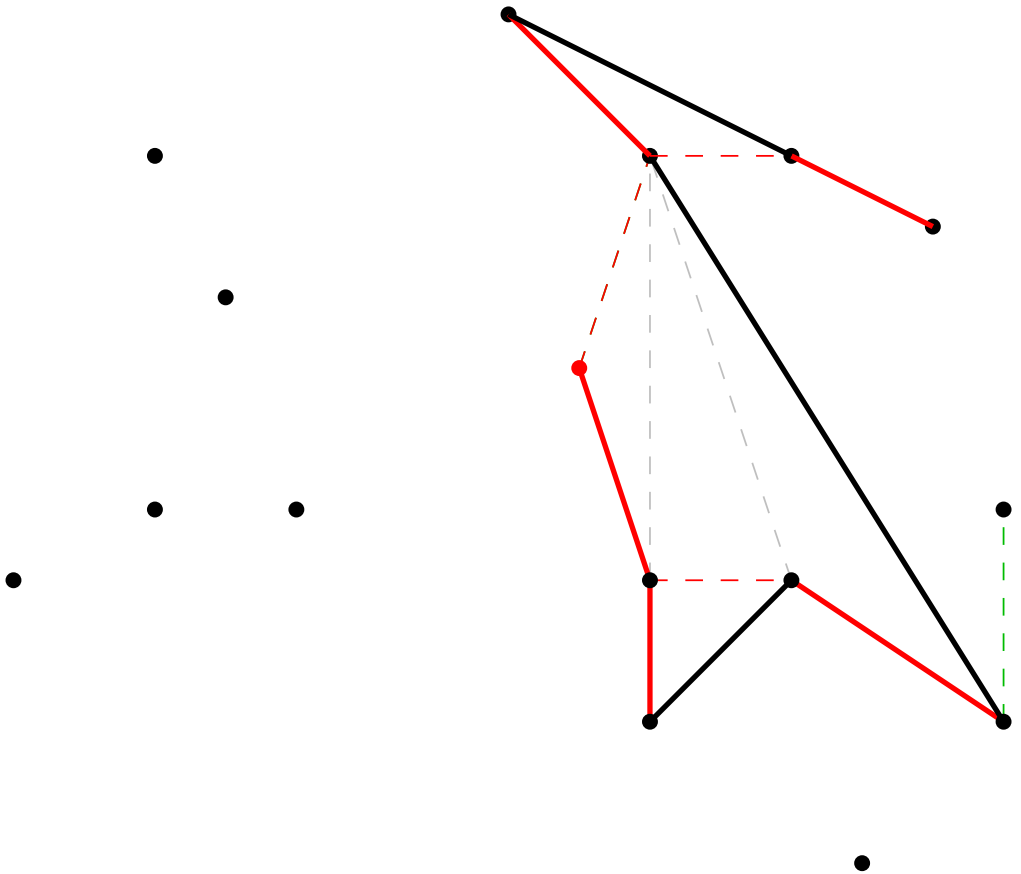
Aufgabe 1



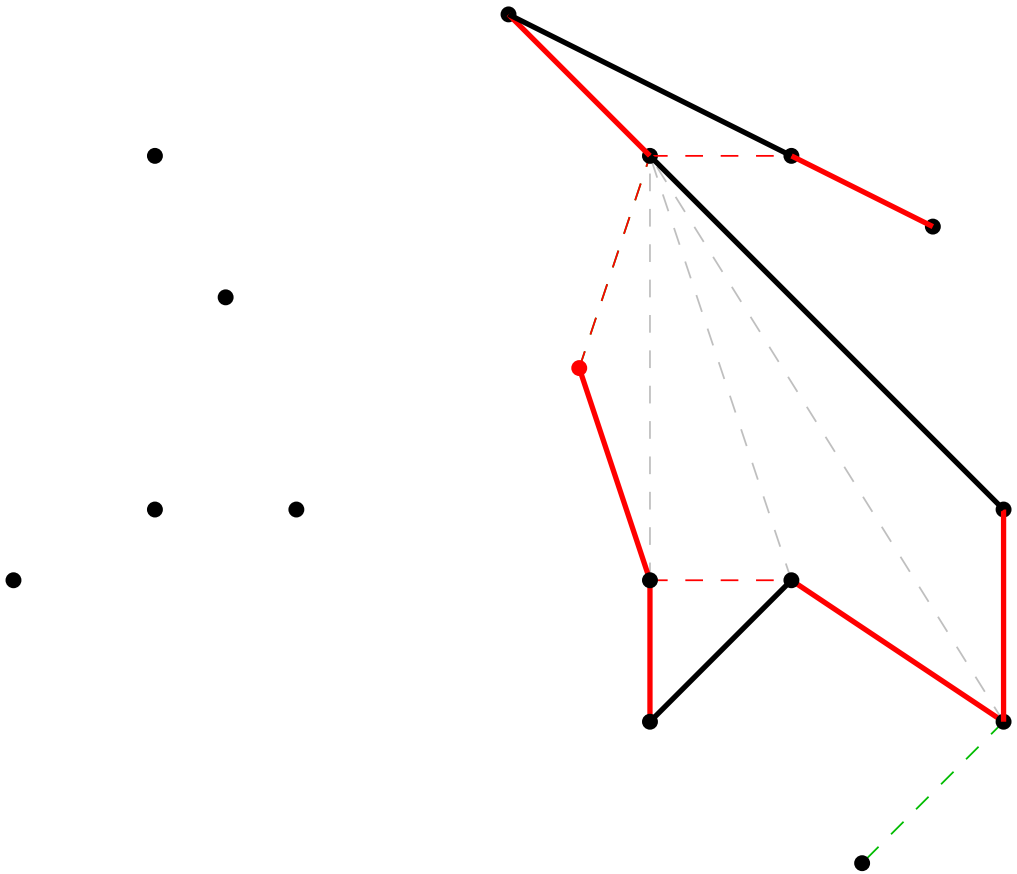
Aufgabe 1



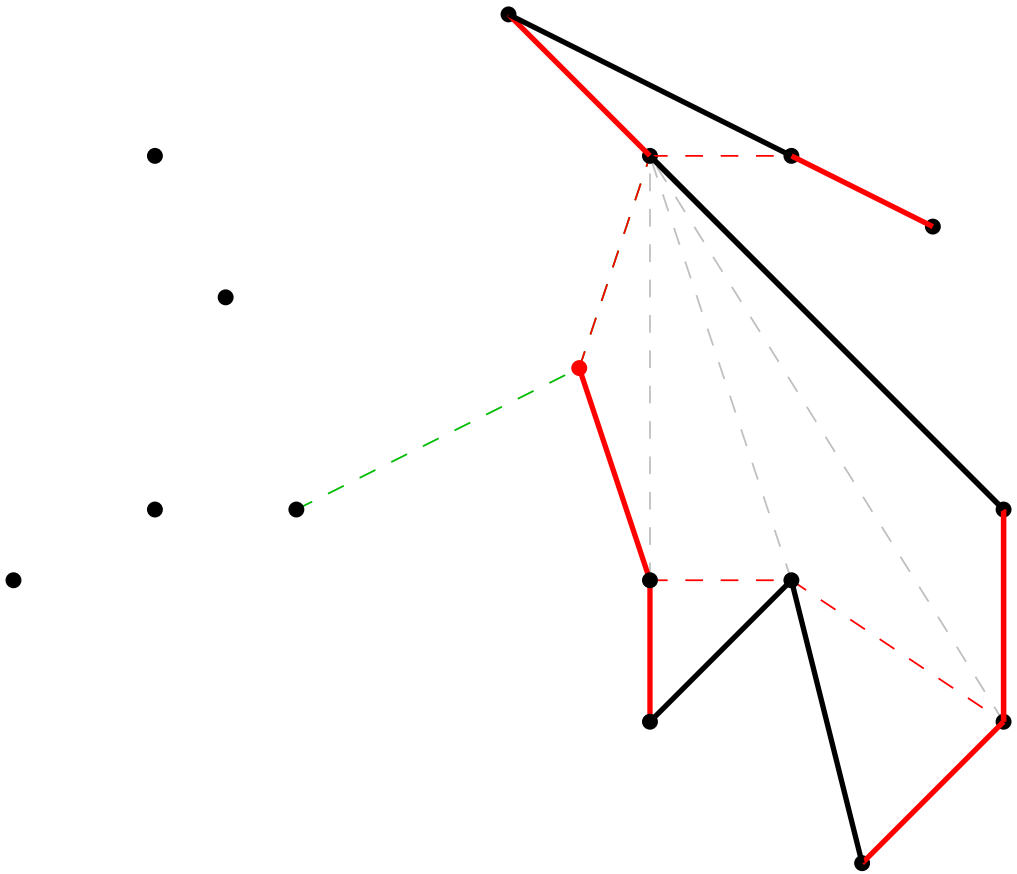
Aufgabe 1



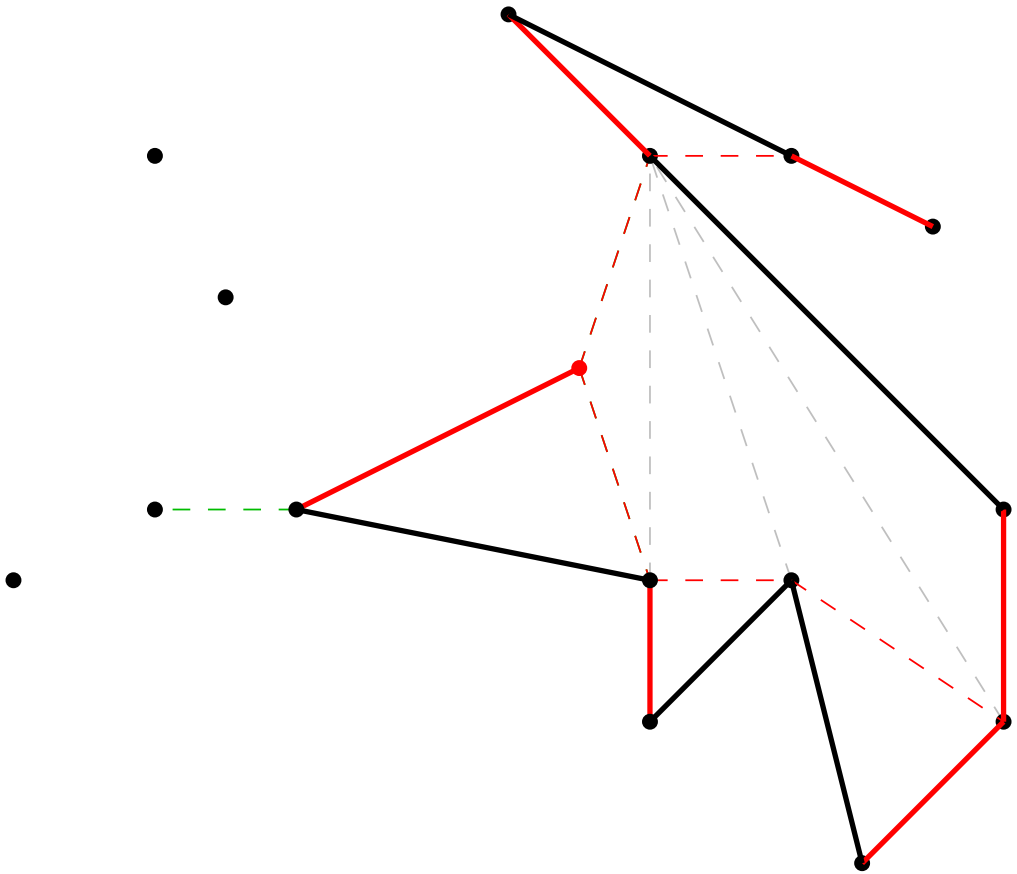
Aufgabe 1



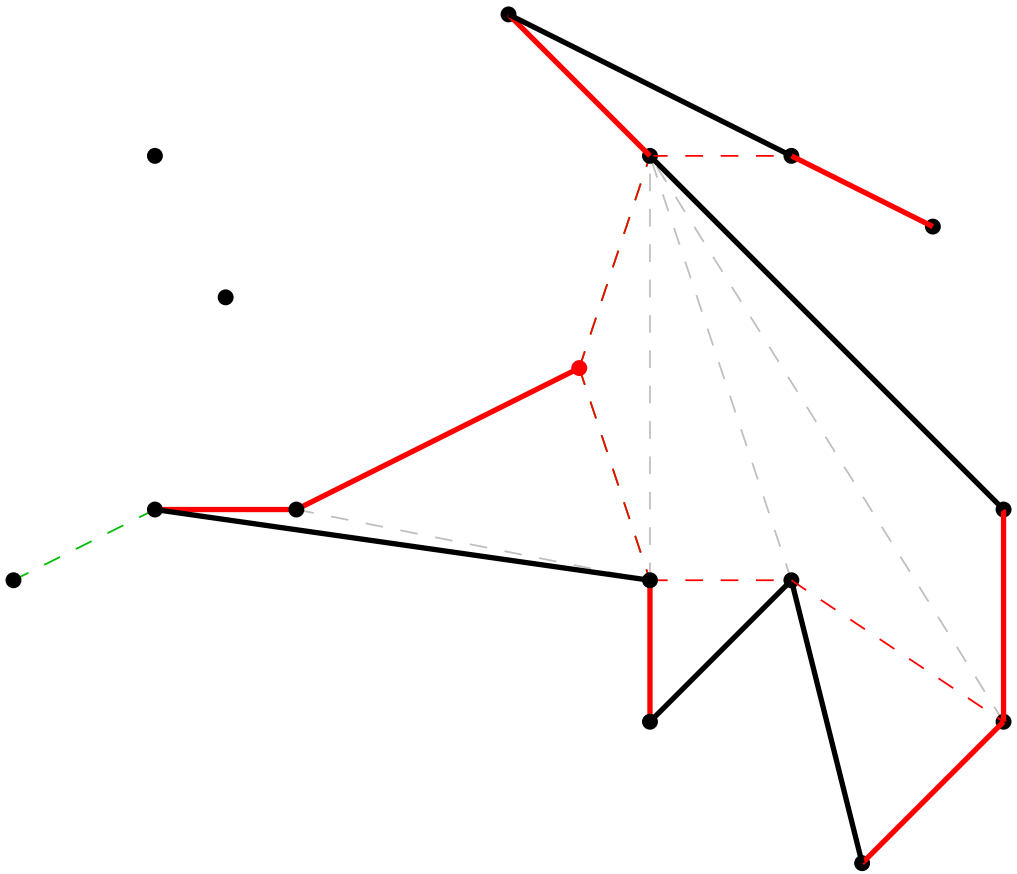
Aufgabe 1



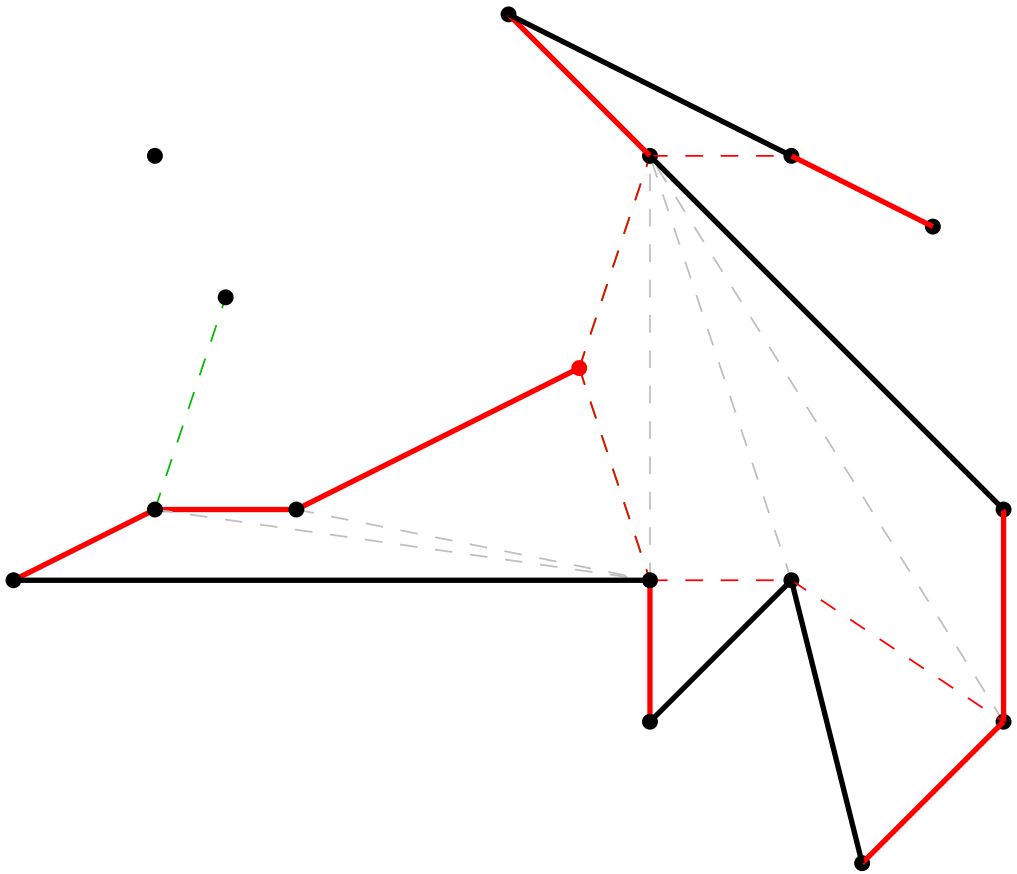
Aufgabe 1



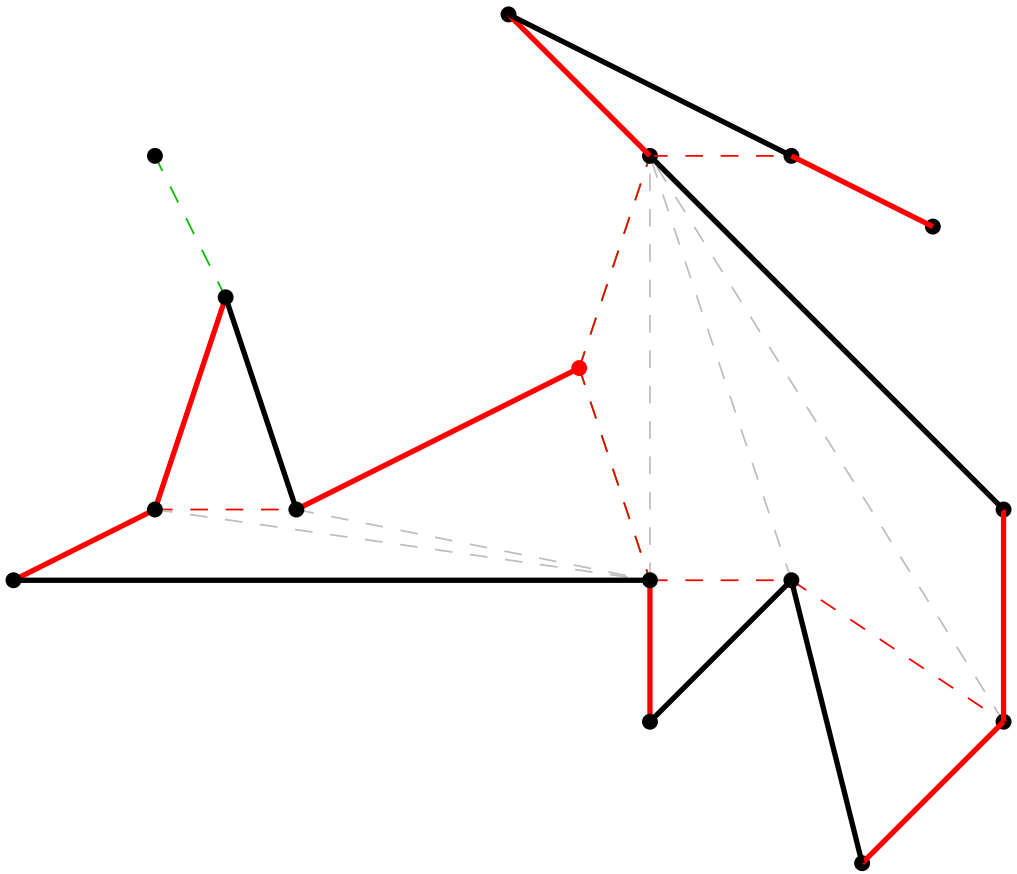
Aufgabe 1



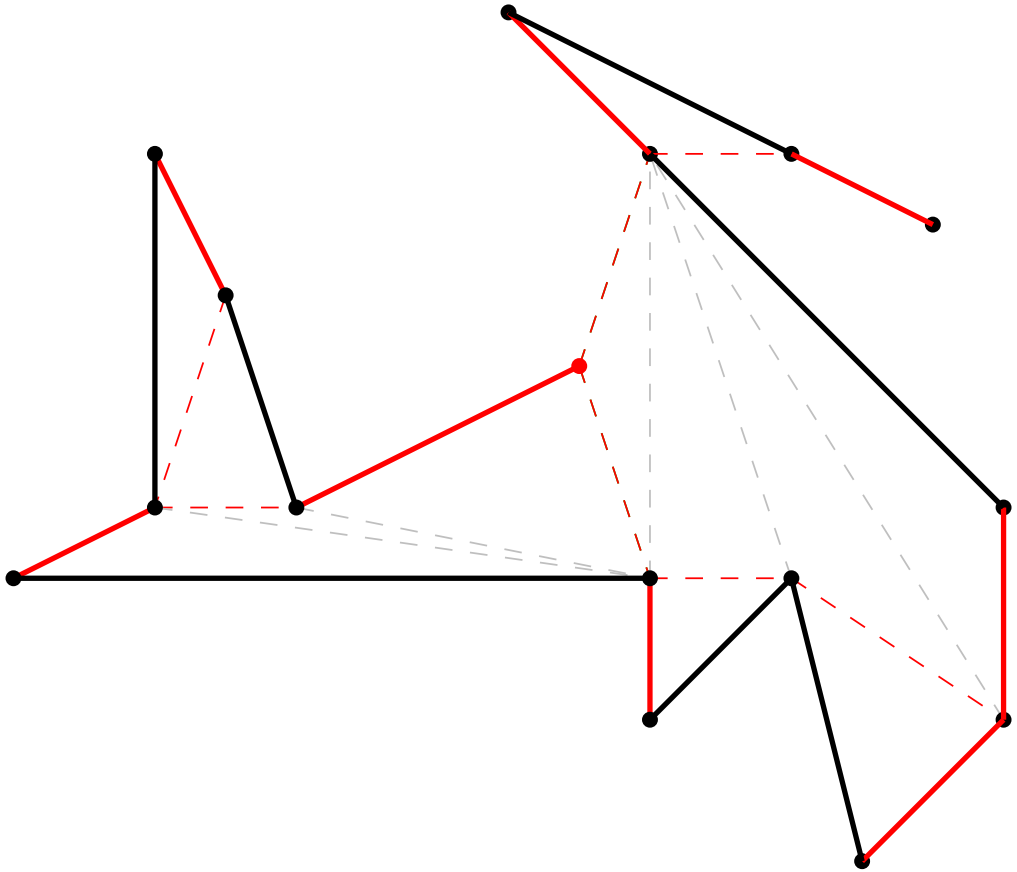
Aufgabe 1



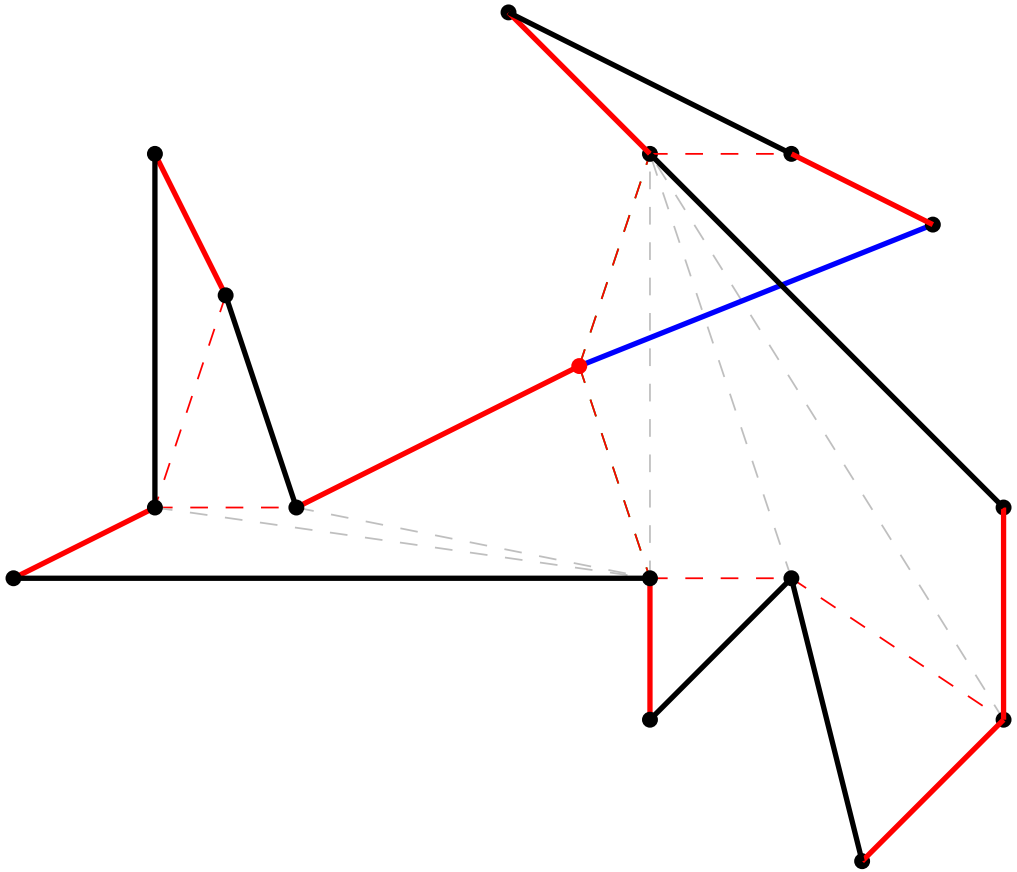
Aufgabe 1



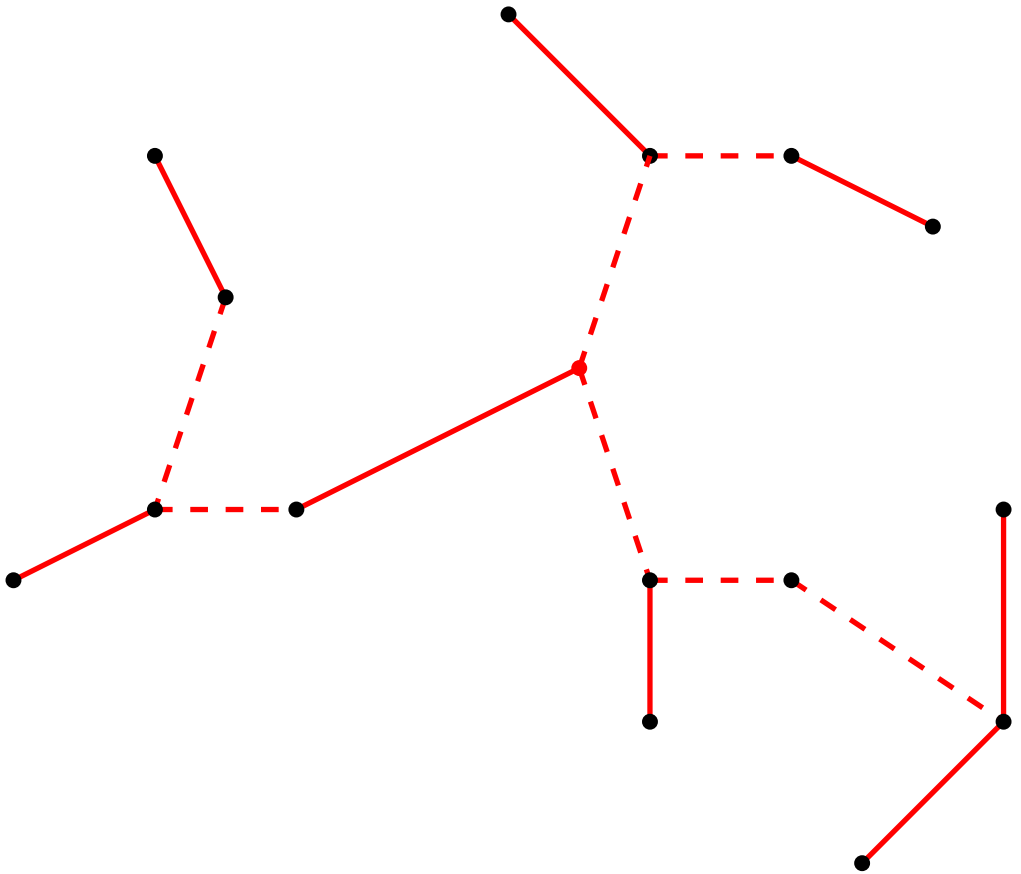
Aufgabe 1



Aufgabe 1



Aufgabe 1



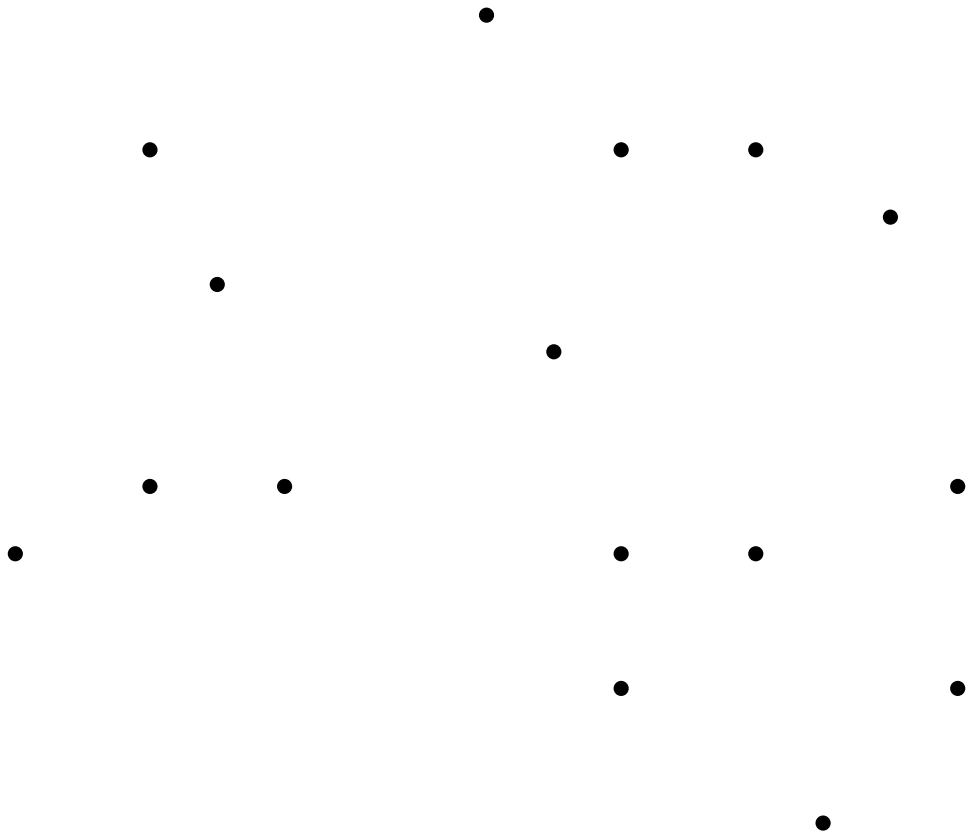
Aufgabe 1

- Schritt:
  $v, p_{i_j} \leftarrow$ Knoten mit $d(p_{i_j}, v)$ minimal, mit $p_{i_j} \in T, v \in V \setminus T$
  ist grüne Regel für Spannbaum
- wenn Knoten v wie folgt eingefügt würde:
  $T \leftarrow (p_{i_1}, \ldots, p_{i_j}, v, p_{i_j}, p_{i_{j+1}}, \ldots p_{i_k})$
  $\rightsquigarrow$ doppeltes Traversieren des Spannbaums
- entstehende Tour hätte Länge $\leq 2(M)$ wie im ersten Ansatz
- kürze Tour ab, um Knoten nicht mehrfach besuchen zu müssen:
  $T \leftarrow (p_{i_1}, \ldots, p_{i_j}, v, p_{i_{j+1}}, \ldots p_{i_k})$
- also $c(T) \leq 2c(M) \leq 2c(T^*)$ (Dreiecksungleichung)

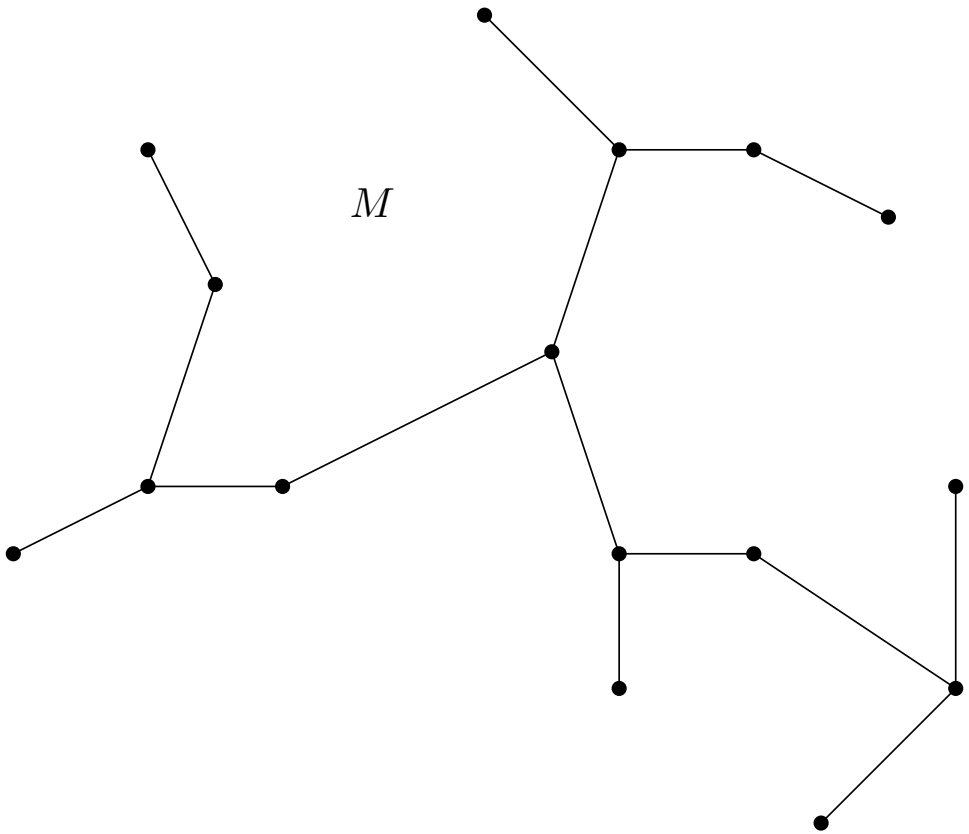
3/2-Approximation

- berechne minimalen Spannbaum M
- sei U die Menge der Knoten mit ungeradem Grad in M
- berechne perfektes Matching P in vollständem Graph auf U mit Gewicht d (in $\mathcal{O}(|V|^2 \log |V| + |V| \cdot |E|)$)
- es gilt $d(P) \leq d(T^*)/2$
- sei $G := M + P$
- alle Knoten in G haben geraden Grad
- berechne Euler-Tour T' in G
- füge ggf. Abkürzungen ein: $\rightsquigarrow$ Tour T
- $d(T) \leq d(T') \leq d(M) + d(P) \leq d(T^*) + d(T^*)/2 = 3/2 d(T^*)$

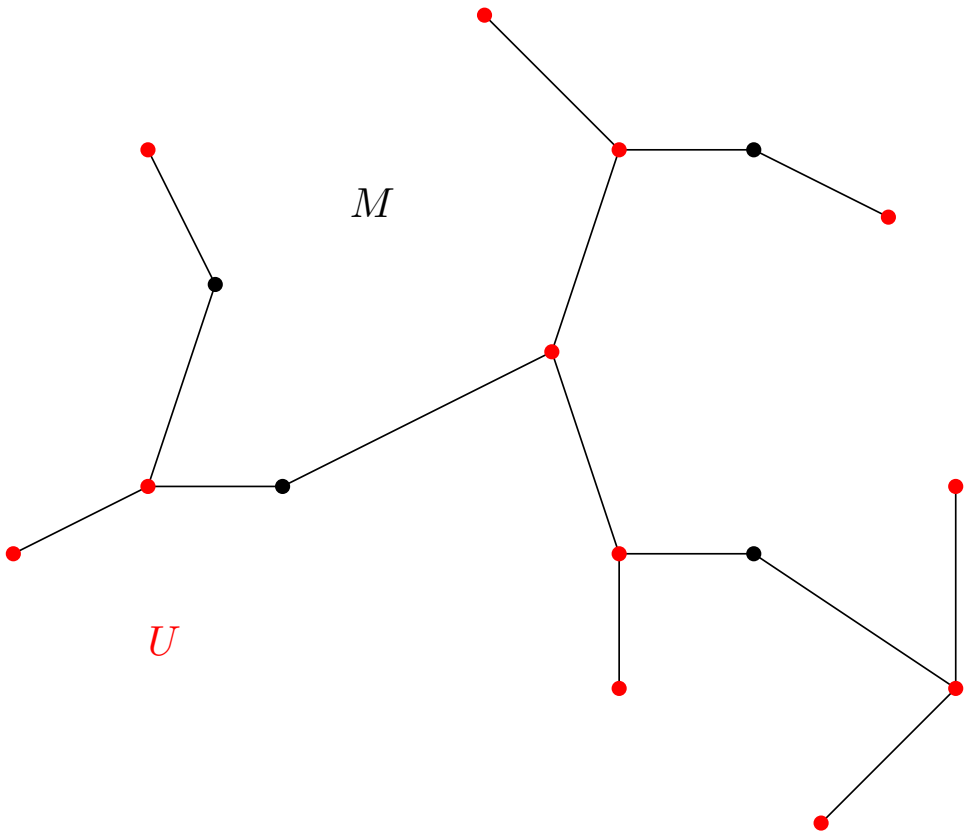
3/2-Approximation



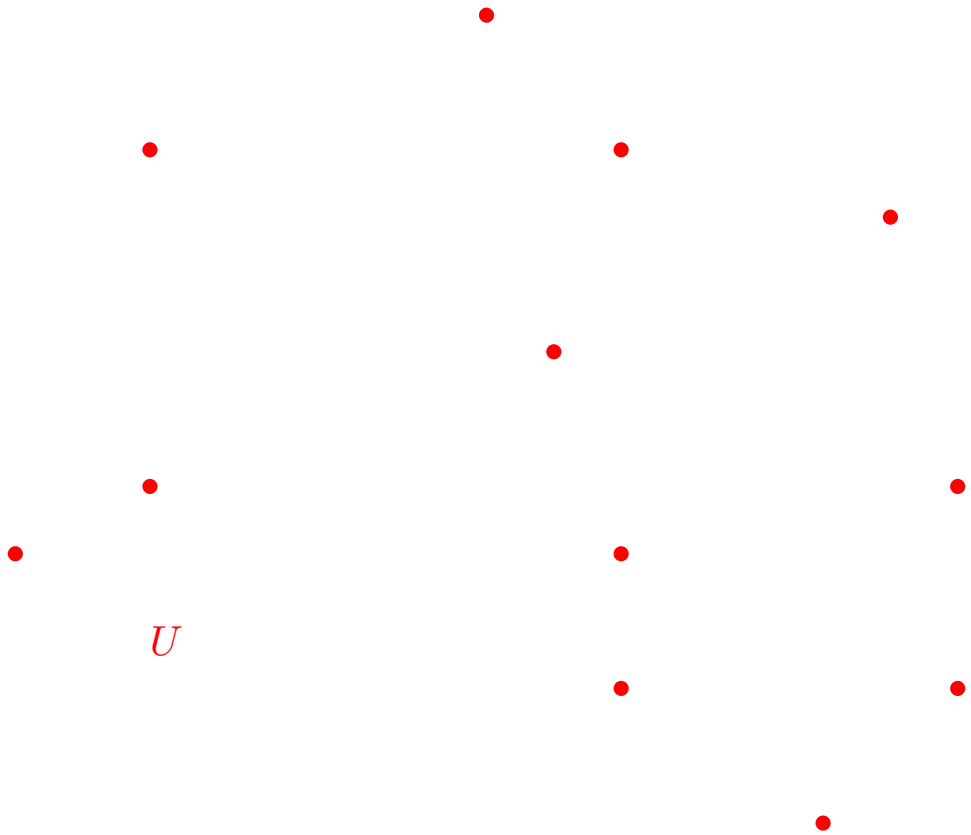
3/2-Approximation



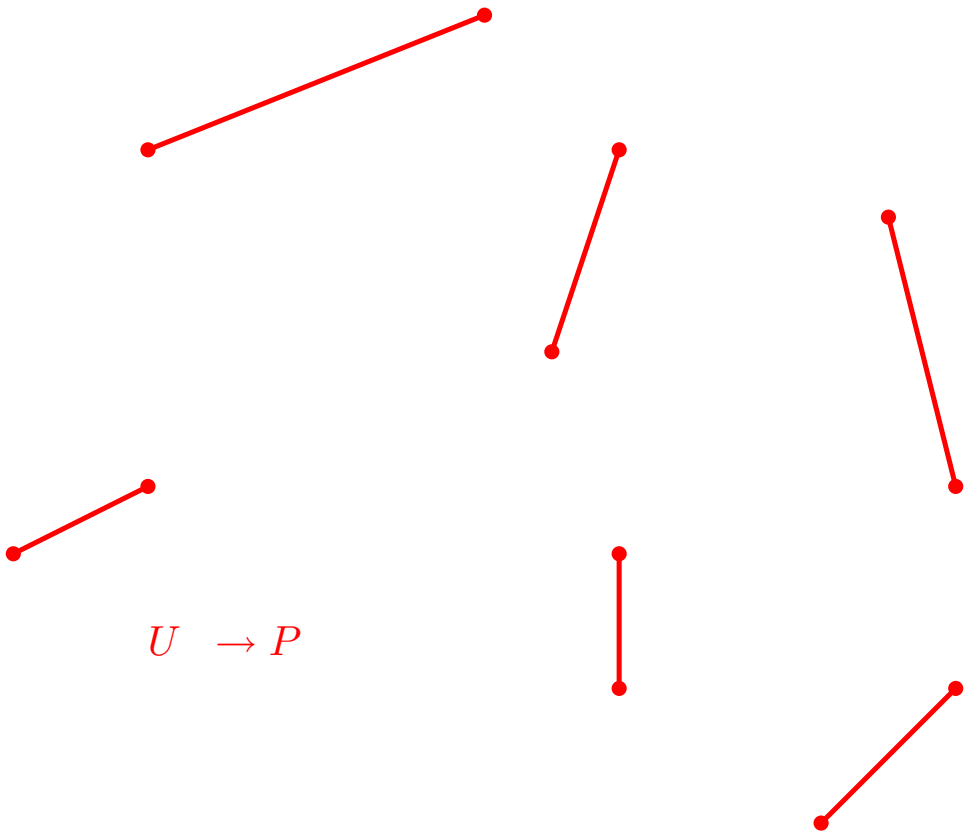
3/2-Approximation



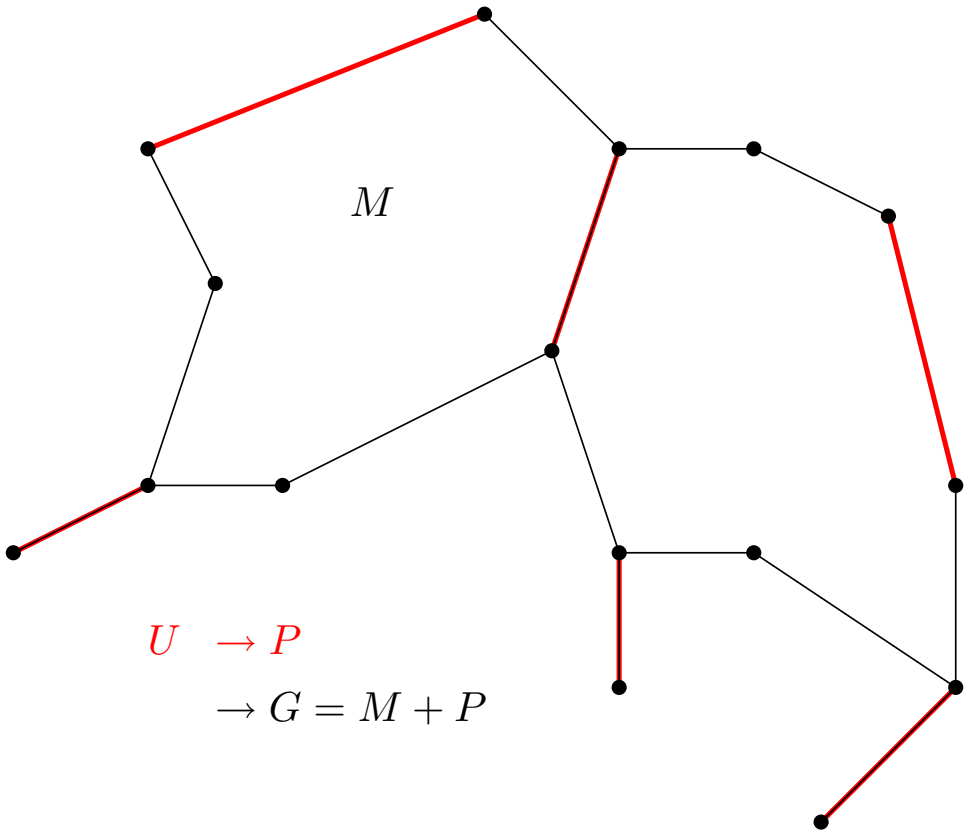
3/2-Approximation



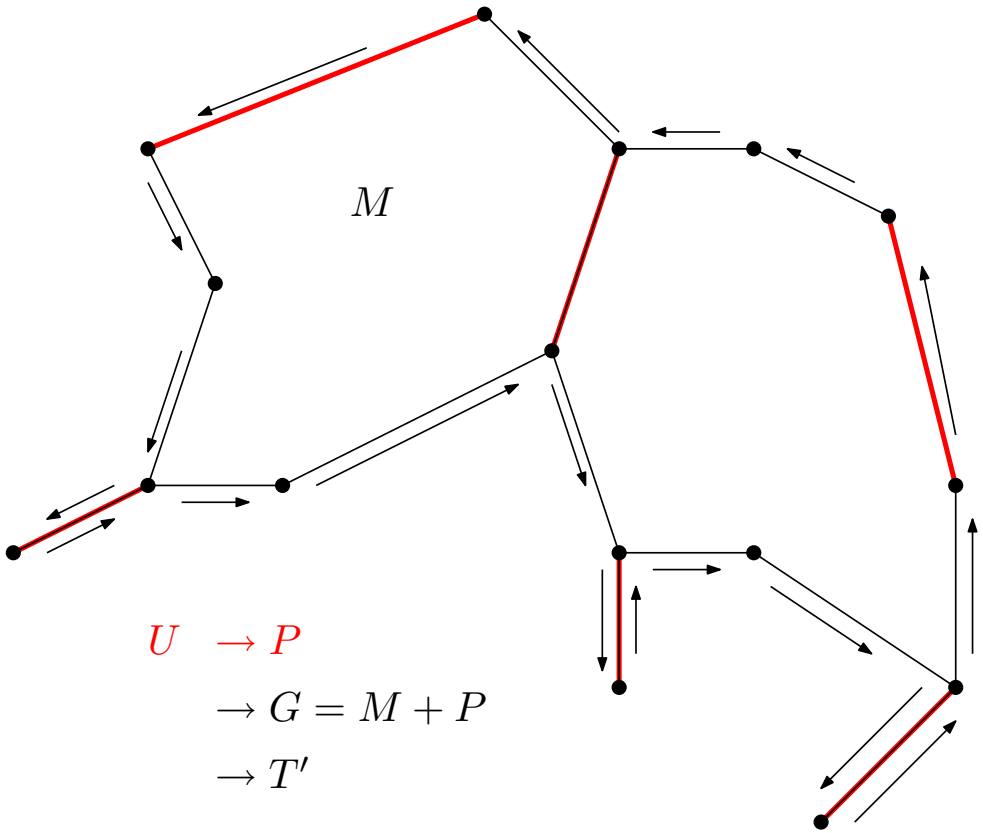
3/2-Approximation



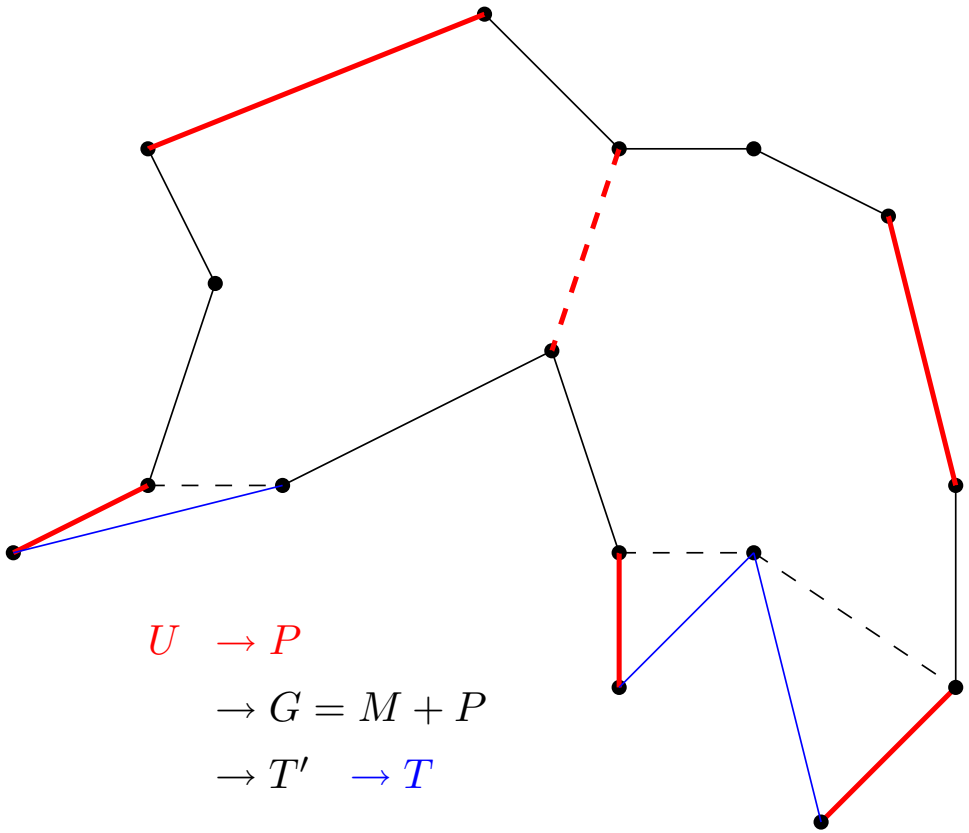
3/2-Approximation



3/2-Approximation



3/2-Approximation



Aufgabe 3

Clique

- Gegeben: Graph $G = (V, E)$
- Gesucht: Clique C maximaler Größe in G

- $\frac{|V|}{\log^2 |V|}$-approximierbar
- nicht $|V|^{1/2-\varepsilon}$-approximierbar mit $\varepsilon > 0$

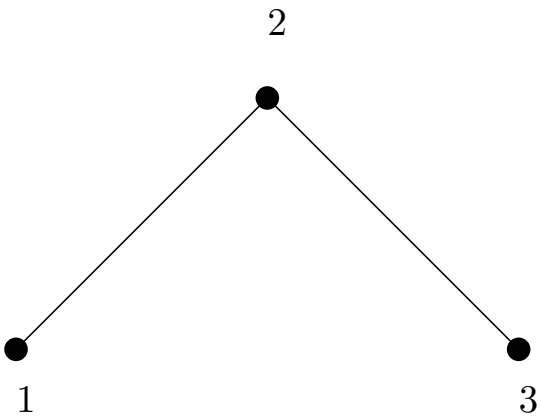
Aufgabe 3

Definition $G^{(k)}$

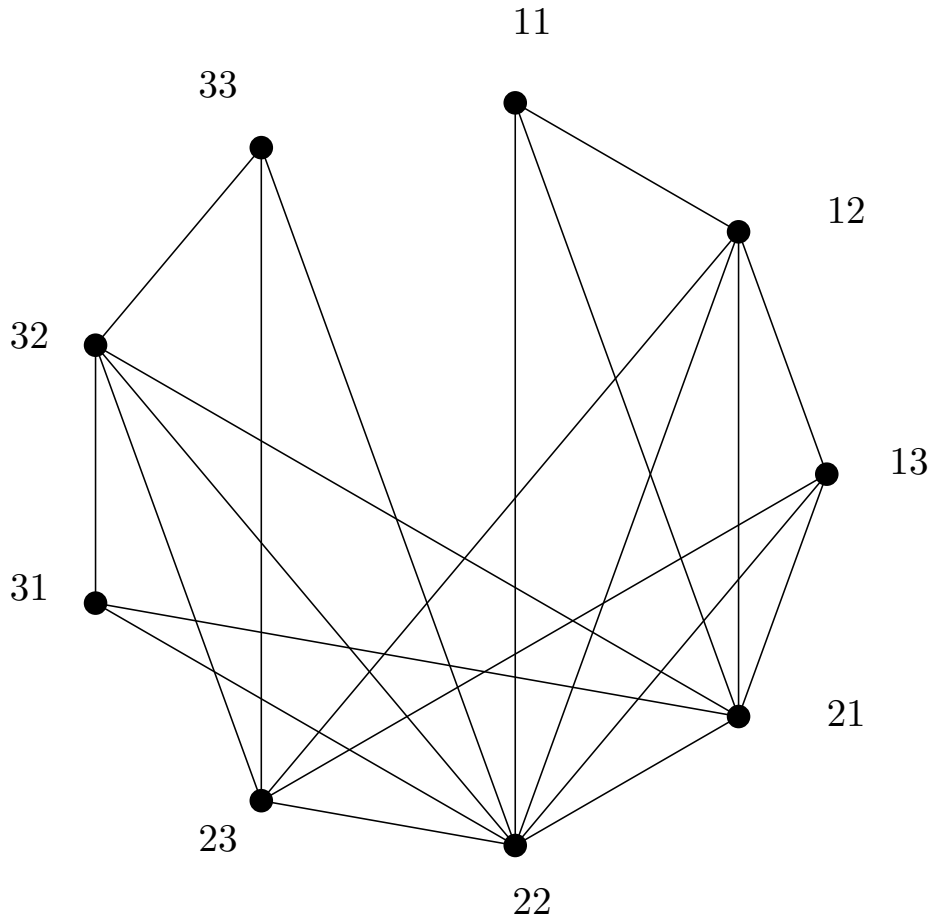
- $G^{(k)} = (V^{(k)}, E^{(k)})$ mit
- $V^{(k)} = \{(v_1, \ldots, v_k) \mid v_i \in V\} = \underbrace{V \times \cdots \times V}_{k}$
- $\{(v_1, \ldots, v_k), (w_1, \ldots, w_k)\} \in E^{(k)}$ g.d.w. für all i entweder $\{v_i, w_i\} \in E$ oder $v_i = w_i$ gilt.

Aufgabe 3

G



$G^{(2)}$



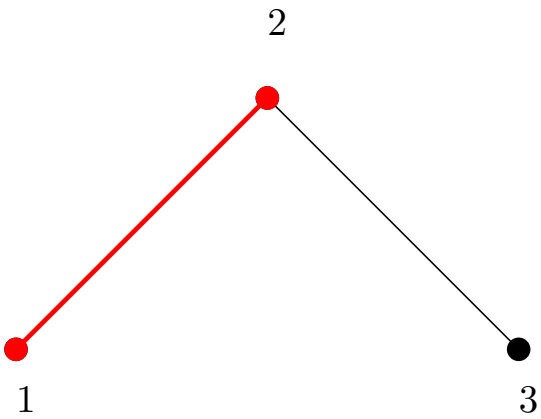
Aufgabe 3

Aufgabe (CLIQUE-Approximation)

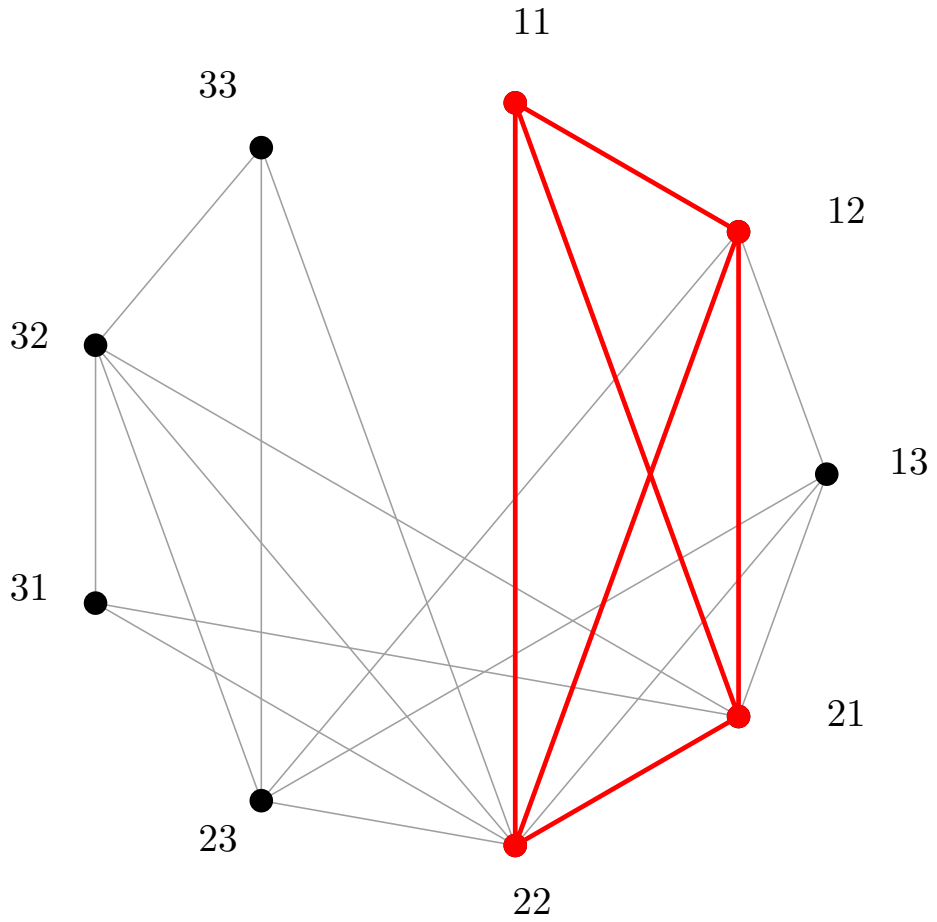
(a) Sei c die Größe einer maximalen Clique in G. Dann ist c^k die Größe einer maximalen Clique in $G^{(k)}$.

Aufgabe 3

G



$G^{(2)}$



Aufgabe 3

Sei c die Größe einer maximalen Clique in G. Dann ist c^k die Größe einer maximalen Clique in $G^{(k)}$.

- $\gg$ C maximale Clique in G mit Größe c
- $\gg$ $C^{(k)} := \{(c_1, \ldots, c_k) \mid c_i \in C\}$ ist Clique in $G^{(k)}$ der Größe c^k

$$C^{(k)} = \underbrace{C \times C \times \cdots \times C}_{k}$$

- $\gg$ $C^{(k)}$ ist Clique in $G^{(k)}$ der Größe c^k
- $\Rightarrow$ Größe der maximalen Clique in $G^{(k)}$ ist mindestens c^k

Aufgabe 3

Sei c die Größe einer maximalen Clique in G. Dann ist c^k die Größe einer maximalen Clique in $G^{(k)}$.

- $\gg$ D Clique in $G^{(k)}$ $D = \{d^1, \ldots, d^\ell\}$ mit $d^i = (d^i_1, \ldots, d^i_k)$
- $\gg$ $S_j := \{d^i_j \mid i = 1, \ldots, \ell\}$ $(j = 1, \ldots, k)$
- $\Rightarrow$ S_j ist Clique in G
- $\Rightarrow$ $|S_j| \leq c$ und

$$D \subseteq S_1 \times S_2 \times \cdots \times S_k$$

- $\Rightarrow$ $|D| \leq \prod_{j=1}^{k} |S_j| \leq c^k$
- $\gg$ Größe der maximalen Clique in $G^{(k)}$ ist höchstens c^k
- $\Rightarrow$ Größe der maximalen Clique in $G^{(k)}$ ist genau c^k

Aufgabe 3

Aufgabe (Clique-Approximation)

(b) Es existiert ein Approximationschema mit polynomieller Laufzeit (PAS) für Clique, falls es einen Approximationsalgorithmus $\mathcal{A}'$ mit konstantem Approximationsverhältnis für dieses Problem gibt.

Sei $\mathcal{A}'$ Algorithmus mit konstantem Approximationsfaktor $a > 1$.
PAS $\mathcal{A}_\varepsilon$ Approximationsfaktor $(1+\varepsilon)$:

- wähle $k := \lceil \frac{1}{\log_a(1+\varepsilon)} \rceil$
- wende $\mathcal{A}'$ auf $G^{(k)}$ an:
  Ergebnis sei Clique $D' = \{d^1, \ldots, d^\ell\} \Rightarrow \frac{OPT(G^{(k)})}{|D'|} \leq a$
- sei i^* so, dass $|S'_{i^*}| = |\{d_j^{i^*} \mid j = 1, \ldots, |D'|\}|$ maximal
- dann gilt $|D'| \leq \prod_{j=1}^{k} |S'_j| \leq |S'_{i^*}|^k$
- sei D Clique in $G^{(k)}$ mit $S_i = S'_{i^*}$ für $i = 1, \ldots, k$

$$D = \underbrace{S_{i^*} \times S_{i^*} \times \cdots \times S_{i^*}}_{k}$$

- dann gilt: $|D| = |S'_{i^*}|^k \geq |D'|$, da $D' \subseteq D$
- gib Clique zu S'_{i^*} in G aus $(|S'_{i^*}| = \sqrt[k]{|D|})$

Aufgabe 3

Sei C maximale Clique in G. Mit $|D| = |S'_{i*}|^k \geq |D'|$ gilt dann:

$$\begin{aligned}
\frac{OPT(G)}{\mathcal{A}_\varepsilon(G)} = \frac{OPT(G)}{|S'_{i*}|} &= \frac{\sqrt[k]{OPT(G)^k}}{\sqrt[k]{|D|}} = \sqrt[k]{\frac{OPT(G^{(k)})}{|D|}} \\
&\leq \sqrt[k]{\frac{OPT(G^{(k)})}{|\overline{D}|}} \\
&\leq \sqrt[k]{a} = a^{1/k} \\
&\leq a^{\log_a(1+\varepsilon)} \\
&= (1+\varepsilon)
\end{aligned}$$

Aufgabe 3

Aufwand

- Graph hat n^k Knoten und $\mathcal{O}(n^{2k})$ Kanten
- sei $p(n)$ der Aufwand für den Approximationsalgorithmus $\mathcal{A}'$
- dann ist der Gesamtaufwand in $\mathcal{O}(n^{2k} + p(n^{2k}))$

- $k := \lceil \frac{1}{\log_a(1+\varepsilon)} \rceil$
- $\Rightarrow n^k = n^{\lceil \frac{1}{\log_a(1+\varepsilon)} \rceil} \geq n^{\frac{1}{\varepsilon}}$
- $\Rightarrow$ also ist $\mathcal{A}$ nicht polynomiell in $\frac{1}{\varepsilon}$ und somit kein *FPAS*
- Aufwand ist in $\Omega(n^{\frac{1}{\varepsilon}})$

Aufgabe 3

Aufgabe (Clique-Approximation)

(c) Falls $\mathcal{P} \neq \mathcal{NP}$ ist, gibt es kein vollpolynomiales Approximationsschema (FPAS) für Clique.

Aufgabe 3

Annahme: Es gibt eine Familie von Algorithmen $\mathcal{A}_\varepsilon$ für jedes $\varepsilon > 0$ mit polynomieller Laufzeit in n und $\frac{1}{\varepsilon}$.

- $\gg$ offensichtlich gilt $\mathcal{A}_\varepsilon \leq \mathcal{O}PT(I) \leq n$
- $\gg$ wähle festes $\varepsilon < \frac{1}{n}$

$$OPT(I) \leq (1+\varepsilon)\mathcal{A}_\varepsilon(I) < \mathcal{A}_\varepsilon(I) + \frac{\mathcal{A}_\varepsilon(I)}{n} \leq \mathcal{A}_\varepsilon(I) + 1$$

- $\gg$ also gilt

$$\mathcal{A}_\varepsilon(I) \leq OPT(I) < \mathcal{A}_\varepsilon(I) + 1$$

- $\gg$ $\mathcal{A}_\varepsilon$ ist polynomiell in n und $\frac{1}{\varepsilon}$ und berechnet OPT
- $\Rightarrow$ $\mathcal{P} = \mathcal{NP}$ im Widerspruch zur Annahme