

Effiziente Algorithmen (SS2015)

Kapitel 6

Approximation I

Walter Unger

Lehrstuhl für Informatik 1

14:43 Uhr, den 29. November 2018

Inhalt I

- 1 Einleitung
 - Motivation
 - Cliquesproblem
- 2 Vertex Cover
 - Definition
 - Greedy
- 3 TSP und Delta-TSP
 - Einleitung
 - 2-Approximation
 - 1.5-Approximation
- 4 Steiner-Bäume

- 5 Zentrumproblem
 - Einleitung
 - Reduktion
 - Approximationsalgorithmus
- 6 Färbung
 - Greedy
 - Approximation
 - Aussagen

Einleitung

- NP-schwere Probleme doch lösen

Einleitung

- NP-schwere Probleme doch lösen
 - Exakt: “ \implies ” exponentielle Laufzeit

Einleitung

- NP-schwere Probleme doch lösen
 - Exakt: “ \Rightarrow ” exponentielle Laufzeit
 - Nicht Exakt: “ \Rightarrow ” hoffentlich polynomiale Laufzeit

Einleitung

- NP-schwere Probleme doch lösen
 - Exakt: “ \Rightarrow ” exponentielle Laufzeit
 - Nicht Exakt: “ \Rightarrow ” hoffentlich polynomiale Laufzeit
- Hier: Approximationsalgorithmen in polynomialer Laufzeit.

Einleitung

- NP-schwere Probleme doch lösen
 - Exakt: “ \Rightarrow ” exponentielle Laufzeit
 - Nicht Exakt: “ \Rightarrow ” hoffentlich polynomiale Laufzeit
- Hier: Approximationsalgorithmen in polynomialer Laufzeit.
- D.h.: welche Approximationsfaktoren sind möglich?

Einleitung

- NP-schwere Probleme doch lösen
 - Exakt: “ \Rightarrow ” exponentielle Laufzeit
 - Nicht Exakt: “ \Rightarrow ” hoffentlich polynomiale Laufzeit
- Hier: Approximationsalgorithmen in polynomialer Laufzeit.
- D.h.: welche Approximationsfaktoren sind möglich?
- Kommt man beliebig nah an das Optimum?

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Knotenfärbung)

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
 - Färbungszahl
- $$\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$$

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Knotenfärbung)

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Knotenfärbung)

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:
COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$
- k -Färbungsproblem:

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Knotenfärbung)

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:
COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$
- k -Färbungsproblem:
 k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Knotenfärbung)

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$

- k -Färbungsproblem:

k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

- praktisches Färbungsproblem:

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Knotenfärbung)

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$

- k -Färbungsproblem:

k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

- praktisches Färbungsproblem:
 - Gegeben: $G = (V, E)$

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Knotenfärbung)

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$

- k -Färbungsproblem:

k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

- praktisches Färbungsproblem:

- Gegeben: $G = (V, E)$
- Bestimme: $c : V \mapsto \mathbb{N}_{\chi(G)} : \forall \{v, w\} \in E : c(v) \neq c(w)$.

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

- $\chi(G) \leqslant 4$.

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

- $\chi(G) \leq 4$.
 - $3\text{-COL}(G) \in \mathcal{NPC}$.

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

- $\chi(G) \leq 4$.
 - $3\text{-COL}(G) \in \mathcal{NPC}$.

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

- $\chi(G) \leq 4$.
 - $3\text{-COL}(G) \in \mathcal{NPC}$.

Folgerung für das Färben von planaren Graphen:

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planar Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

- $\chi(G) \leq 4$.
 - $3\text{-COL}(G) \in \mathcal{NPC}$.

Folgerung für das Färben von planaren Graphen:

- kann mit additivem Fehler von 1 gelöst werden.

Einfaches Beispiel (Knotenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Planarer Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

- $\chi(G) \leq 4$.
- $3\text{-COL}(G) \in \mathcal{NPC}$.

Folgerung für das Färben von planaren Graphen:

- kann mit additivem Fehler von 1 gelöst werden.
- kann mit multipikativem Fehler von $4/3$ gelöst werden.

Einfaches Beispiel (Kantenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Einfaches Beispiel (Kantenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Theorem (Kantenfärbung)

Für Graphen G gilt:

Folgerung für die Kantenfärbung:

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Theorem (Kantenfärbung)

Für Graphen G gilt:

- $\delta(G) \leq \chi'(G) \leq \delta(G) + 1$.

Folgerung für die Kantenfärbung:

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Einfaches Beispiel (Kantenfärbung)

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Theorem (Kantenfärbung)

Für Graphen G gilt:

- $\delta(G) \leq \chi'(G) \leq \delta(G) + 1$.
- $(\delta(G)) - \text{EDGECOL}(G) \in \mathcal{NPC}$.

Folgerung für die Kantenfärbung:

Einfaches Beispiel (Kantenfärbung)

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Theorem (Kantenfärbung)

Für Graphen G gilt:

- $\delta(G) \leq \chi'(G) \leq \delta(G) + 1$.
- $(\delta(G)) - \text{EDGECOL}(G) \in \mathcal{NPC}$.

Folgerung für die Kantenfärbung:

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Theorem (Kantenfärbung)

Für Graphen G gilt:

- $\delta(G) \leq \chi'(G) \leq \delta(G) + 1$.
- $(\delta(G)) - \text{EDGECOL}(G) \in \mathcal{NPC}$.

Folgerung für die Kantenfärbung:

- kann mit additivem Fehler von 1 gelöst werden.

Einfaches Beispiel (Kantenfärbung)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Theorem (Kantenfärbung)

Für Graphen G gilt:

- $\delta(G) \leq \chi'(G) \leq \delta(G) + 1$.
- $(\delta(G)) - \text{EDGECOL}(G) \in \mathcal{NPC}$.

Folgerung für die Kantenfärbung:

- kann mit additivem Fehler von 1 gelöst werden.
- kann mit multipikativem Fehler von $(\delta(G) + 1)/\delta(G)$ gelöst werden.

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geq A(I) - k$ (Minimierungsproblem)

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geq A(I) - k$ (Minimierungsproblem)

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geq A(I) - k$ (Minimierungsproblem)

Definition

Ein Algorithmus A hat einen multiplikativen Approximationfehler, falls gilt:

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geq A(I) - k$ (Minimierungsproblem)

Definition

Ein Algorithmus A hat einen multiplikativen Approximationfehler, falls gilt:

- $\forall I : \frac{A(I)}{opt(I)} \leq \alpha$ und $\alpha \geq 1$ bei einem Minimierungsproblem und

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geq A(I) - k$ (Minimierungsproblem)

Definition

Ein Algorithmus A hat einen multiplikativen Approximationfehler, falls gilt:

- $\forall I : \frac{A(I)}{opt(I)} \leq \alpha$ und $\alpha \geq 1$ bei einem Minimierungsproblem und
- $\forall I : \frac{A(I)}{opt(I)} \geq \alpha$ und $\alpha \leq 1$ bei einem Maximierungsproblem.

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geq A(I) - k$ (Minimierungsproblem)

Definition

Ein Algorithmus A hat einen multiplikativen Approximationfehler, falls gilt:

- $\forall I : \frac{A(I)}{opt(I)} \leq \alpha$ und $\alpha \geq 1$ bei einem Minimierungsproblem und
- $\forall I : \frac{A(I)}{opt(I)} \geq \alpha$ und $\alpha \leq 1$ bei einem Maximierungsproblem.

Definition (Konstante Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leq A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geq A(I) - k$ (Minimierungsproblem)

Definition

Ein Algorithmus A hat einen multiplikativen Approximationfehler, falls gilt:

- $\forall I : \frac{A(I)}{opt(I)} \leq \alpha$ und $\alpha \geq 1$ bei einem Minimierungsproblem und
- $\forall I : \frac{A(I)}{opt(I)} \geq \alpha$ und $\alpha \leq 1$ bei einem Maximierungsproblem.

Oft wird vereinfacht: $\max\left\{\frac{A(I)}{opt(I)}, \frac{opt(I)}{A(I)}\right\} \leq \alpha$.

Definition (Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Sei $L : \mathbb{N} \mapsto \mathbb{R}$ eine Funktion und seien weiter I_n die Eingaben für Algorithmus A . Dann hat A einen Approximationsfaktor, falls gilt:

Definition (Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Sei $L : \mathbb{N} \mapsto \mathbb{R}$ eine Funktion und seien weiter I_n die Eingaben für Algorithmus A . Dann hat A einen Approximationsfaktor, falls gilt:

- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{A(I)}{opt(I)} \leq L(n)$ bei einem Minimierungsproblem und

Definition (Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Sei $L : \mathbb{N} \mapsto \mathbb{R}$ eine Funktion und seien weiter I_n die Eingaben für Algorithmus A . Dann hat A einen Approximationsfaktor, falls gilt:

- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{A(I)}{opt(I)} \leq L(n)$ bei einem Minimierungsproblem und
- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{opt(I)}{A(I)} \leq L(n)$ bei einem Maximierungsproblem.

Definition (Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Sei $L : \mathbb{N} \mapsto \mathbb{R}$ eine Funktion und seien weiter I_n die Eingaben für Algorithmus A . Dann hat A einen Approximationsfaktor, falls gilt:

- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{A(I)}{opt(I)} \leq L(n)$ bei einem Minimierungsproblem und
- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{opt(I)}{A(I)} \leq L(n)$ bei einem Maximierungsproblem.

Definition (Approximation)

$$\mathbb{N}_k = \{\mathbf{1}, \dots, k\}$$

Definition

Sei $L : \mathbb{N} \mapsto \mathbb{R}$ eine Funktion und seien weiter I_n die Eingaben für Algorithmus A . Dann hat A einen Approximationsfaktor, falls gilt:

- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{A(I)}{opt(I)} \leq L(n)$ bei einem Minimierungsproblem und
- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{opt(I)}{A(I)} \leq L(n)$ bei einem Maximierungsproblem.

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Kreuzprodukt von Graphen

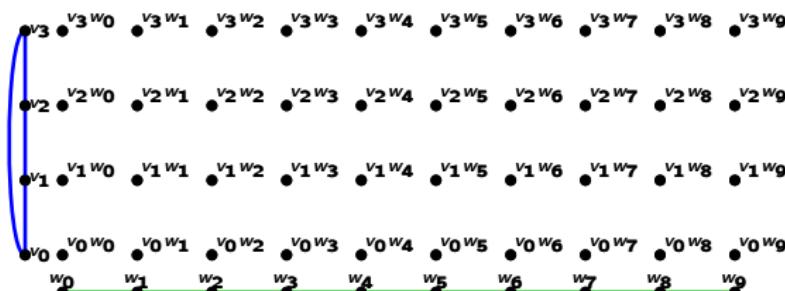
$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Beispiel $L(10) \times C(4)$:



Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Beispiel $L(10) \times C(4)$:



Kreuzprodukt von Graphen

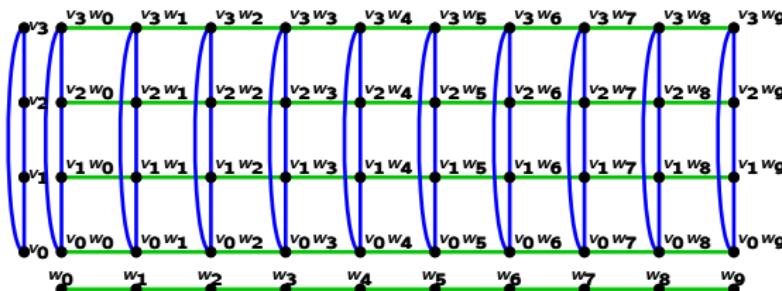
$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Beispiel $L(10) \times C(4)$:



Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.

Kreuzprodukt von Graphen

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquesproblem.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquesproblem.

- Zeigen hier nur additiven Fehler.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \text{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquesproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliqueproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.
- Damit: $\omega(G) \cdot k = \omega(G^k)$.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliqueproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.
- Damit: $\omega(G) \cdot k = \omega(G^k)$.
- Angenommen: es gibt einen Algorithmus A mit additiven Approximationsfehler k für das Cliqueproblem.

Cliquenproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquenproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.
- Damit: $\omega(G) \cdot k = \omega(G^k)$.
- Angenommen: es gibt einen Algorithmus A mit additiven Approximationsfehler k für das Cliquenproblem.
- Eingabe für A wird G^{k+1} .

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliqueproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.
- Damit: $\omega(G) \cdot k = \omega(G^k)$.
- Angenommen: es gibt einen Algorithmus A mit additiven Approximationsfehler k für das Cliqueproblem.
- Eingabe für A wird G^{k+1} .
- Damit gilt: $opt(I) \leq A(I) + k$ und weiter:

Cliquenproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquenproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.
- Damit: $\omega(G) \cdot k = \omega(G^k)$.
- Angenommen: es gibt einen Algorithmus A mit additiven Approximationsfehler k für das Cliquenproblem.
- Eingabe für A wird G^{k+1} .
- Damit gilt: $opt(I) \leq A(I) + k$ und weiter:
- $\omega(G^{k+1}) - A(G^{k+1}) \leq k$.

Cliquenproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquenproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.
- Damit: $\omega(G) \cdot k = \omega(G^k)$.
- Angenommen: es gibt einen Algorithmus A mit additiven Approximationsfehler k für das Cliquenproblem.
- Eingabe für A wird G^{k+1} .
- Damit gilt: $opt(I) \leq A(I) + k$ und weiter:
- $\omega(G^{k+1}) - A(G^{k+1}) \leq k$.
- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.

Cliquenproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \text{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquenproblem.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquesproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.

Cliquenproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \text{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquenproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k+1$) die Kopien von G in G^{k+1} .

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquesproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k+1$) die Kopien von G in G^{k+1} .
- Seien C_i die Lösungsanteile in G_i .

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquesproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k+1$) die Kopien von G in G^{k+1} .
- Seien C_i die Lösungsanteile in G_i .
- $\forall i : 1 \leq i \leq k+1 : C_i \leq \omega(G)$

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquesproblem.

- $(k + 1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k + 1$) die Kopien von G in G^{k+1} .
- Seien C_i die Lösungsanteile in G_i .
- $\forall i : 1 \leq i \leq k + 1 : C_i \leq \omega(G)$
- $(k + 1) \cdot \omega(G) - \sum_{i=1}^{k+1} |C_i| \leq k$.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquesproblem.

- $(k + 1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k + 1$) die Kopien von G in G^{k+1} .
- Seien C_i die Lösungsanteile in G_i .
- $\forall i : 1 \leq i \leq k + 1 : C_i \leq \omega(G)$
- $(k + 1) \cdot \omega(G) - \sum_{i=1}^{k+1} |C_i| \leq k$.
- $\sum_{i=1}^{k+1} \omega(G) - |C_i| \leq k$.

Cliqueproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquesproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k+1$) die Kopien von G in G^{k+1} .
- Seien C_i die Lösungsanteile in G_i .
- $\forall i : 1 \leq i \leq k+1 : C_i \leq \omega(G)$
- $(k+1) \cdot \omega(G) - \sum_{i=1}^{k+1} |C_i| \leq k$.
- $\sum_{i=1}^{k+1} \omega(G) - |C_i| \leq k$.
- $\exists i : C_i = \omega(G)$.

Cliquenproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationssfehler k für das Cliquenproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k+1$) die Kopien von G in G^{k+1} .
- Seien C_i die Lösungsanteile in G_i .
- $\forall i : 1 \leq i \leq k+1 : C_i \leq \omega(G)$
- $(k+1) \cdot \omega(G) - \sum_{i=1}^{k+1} |C_i| \leq k$.
- $\sum_{i=1}^{k+1} \omega(G) - |C_i| \leq k$.
- $\exists i : C_i = \omega(G)$.
- Damit ist das Cliquenproblem in \mathcal{P} .

Cliquenproblem

$$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquenproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leq k$.
- Sei G_i ($1 \leq i \leq k+1$) die Kopien von G in G^{k+1} .
- Seien C_i die Lösungsanteile in G_i .
- $\forall i : 1 \leq i \leq k+1 : C_i \leq \omega(G)$
- $(k+1) \cdot \omega(G) - \sum_{i=1}^{k+1} |C_i| \leq k$.
- $\sum_{i=1}^{k+1} \omega(G) - |C_i| \leq k$.
- $\exists i : C_i = \omega(G)$.
- Damit ist das Cliquenproblem in \mathcal{P} .
- Kann auf multiplikativen Fehler erweitert werden.

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in $\mathcal{NP}\mathcal{C}$:

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in \mathcal{NPC} :
 - Gegeben: G, k .

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in \mathcal{NPC} :
 - Gegeben: G, k .
 - Frage: Gibt es ein Vertex Cover C in G mit $|C| \leq k$.

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in \mathcal{NPC} :
 - Gegeben: G, k .
 - Frage: Gibt es ein Vertex Cover C in G mit $|C| \leq k$.
- Reduktionsidee: $V \setminus C$ ist eine stabile Menge.

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in \mathcal{NPC} :
 - Gegeben: G, k .
 - Frage: Gibt es ein Vertex Cover C in G mit $|C| \leq k$.
- Reduktionsidee: $V \setminus C$ ist eine stabile Menge.
- Versuche nun mittels Greedy das Vertex Cover Problem zu approximieren.

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in \mathcal{NPC} :
 - Gegeben: G, k .
 - Frage: Gibt es ein Vertex Cover C in G mit $|C| \leq k$.
- Reduktionsidee: $V \setminus C$ ist eine stabile Menge.
- Versuche nun mittels Greedy das Vertex Cover Problem zu approximieren.
- Strategie: Wähle Knoten vom höchsten Grad.

Einleitung

Definition

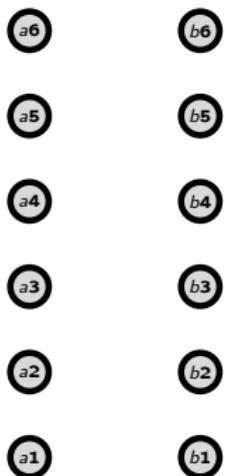
Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G , falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in \mathcal{NPC} :
 - Gegeben: G, k .
 - Frage: Gibt es ein Vertex Cover C in G mit $|C| \leq k$.
- Reduktionsidee: $V \setminus C$ ist eine stabile Menge.
- Versuche nun mittels Greedy das Vertex Cover Problem zu approximieren.
- Strategie: Wähle Knoten vom höchsten Grad.
- Betrachte dazu das folgende Beispiel:

Vertex Cover (Greedy)

Vertex Cover (Greedy)



Vertex Cover (Greedy)

- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$

a6

b6

a5

b5

a4

b4

a3

b3

a2

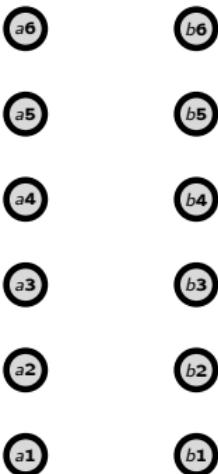
b2

a1

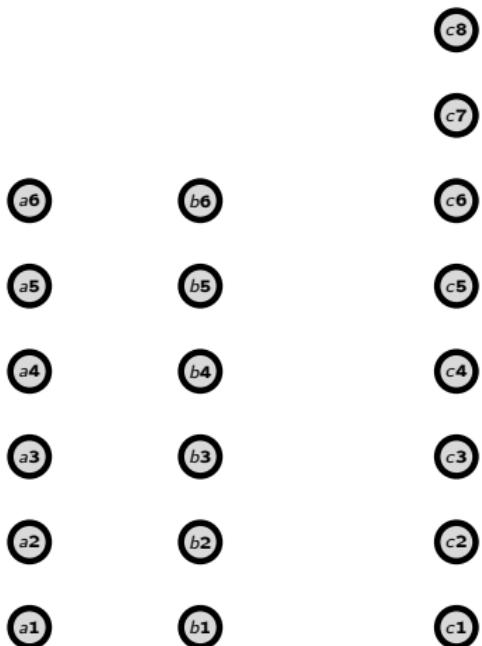
b1

Vertex Cover (Greedy)

- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$

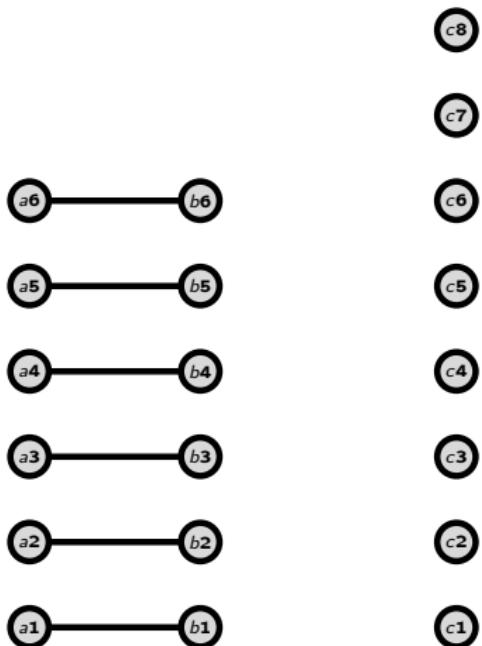


Vertex Cover (Greedy)



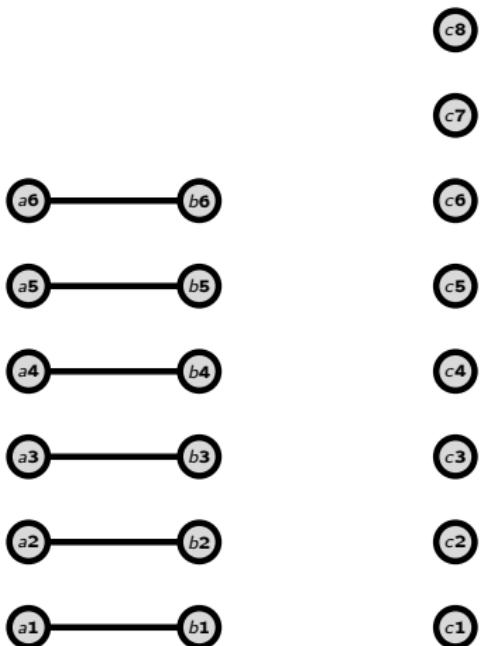
- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit
 $k' = \sum_{j=2}^k \lfloor k/j \rfloor$

Vertex Cover (Greedy)



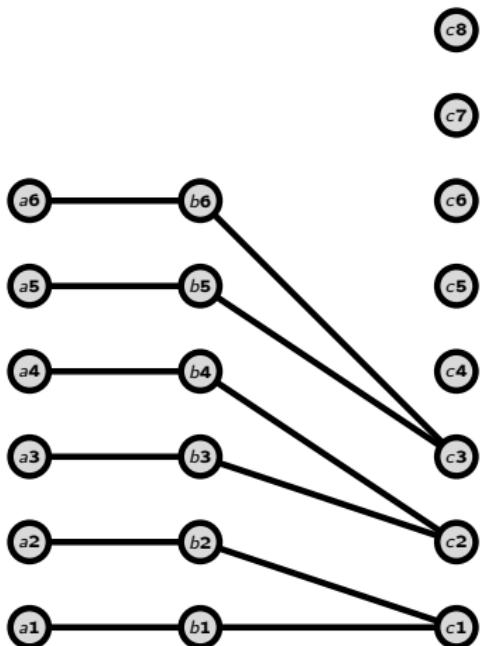
- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit
 $k' = \sum_{j=2}^k \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leq i \leq k\}$

Vertex Cover (Greedy)



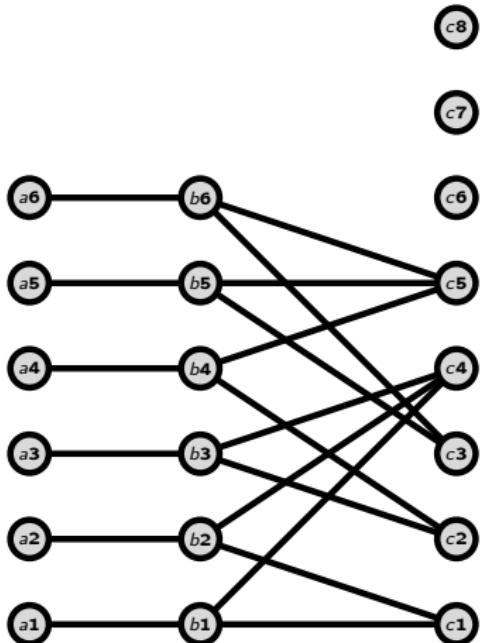
- $G_k = (A_k \cup B_k \cup C_{k'}, \bigcup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit
 $k' = \sum_{j=2}^k \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leq i \leq k\}$
- $k_i = \sum_{j=2}^i \lfloor k/j \rfloor$ für $i = 2, 3, \dots, k$

Vertex Cover (Greedy)



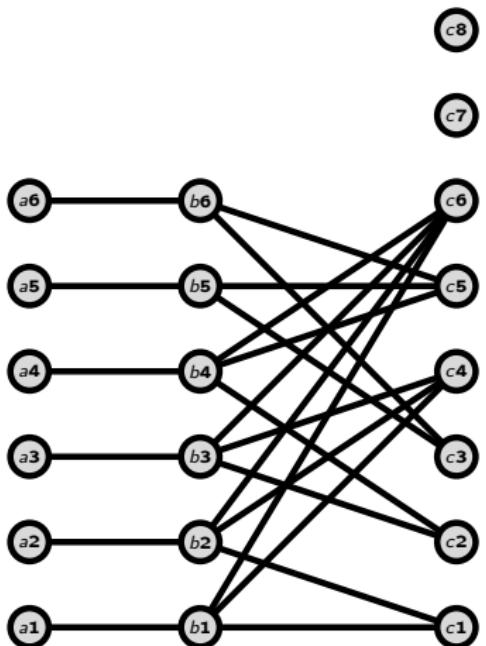
- $G_k = (A_k \cup B_k \cup C_{k'}, \bigcup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit
 $k' = \sum_{j=2}^k \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leq i \leq k\}$
- $k_i = \sum_{j=2}^i \lfloor k/j \rfloor$ für $i = 2, 3, \dots, k$
- $k_1 = 0$

Vertex Cover (Greedy)



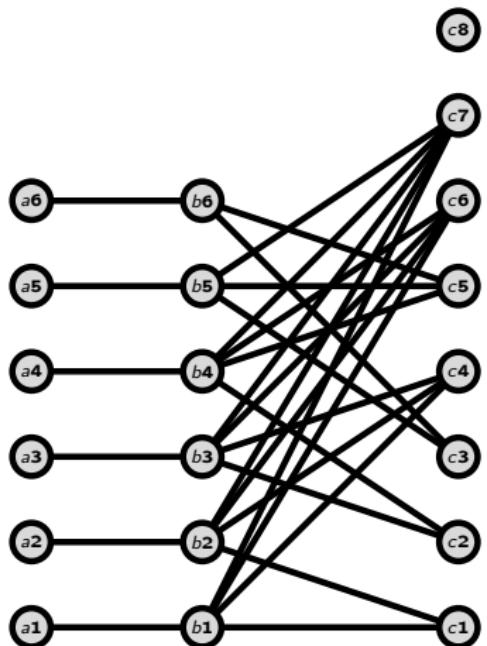
- $G_k = (A_k \cup B_k \cup C_{k'}, \bigcup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit $k' = \sum_{j=2}^k \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leq i \leq k\}$
- $k_i = \sum_{j=2}^i \lfloor k/j \rfloor$ für $i = 2, 3, \dots, k$
- $k_1 = 0$
- $E_i = \{\{c_{k_{i-1}+j}, b_{i*(j-1)+x}\} \mid 1 \leq j \leq \lfloor k/i \rfloor \wedge 1 \leq x \leq i\}$
für $i = 2, 3, \dots, k$

Vertex Cover (Greedy)



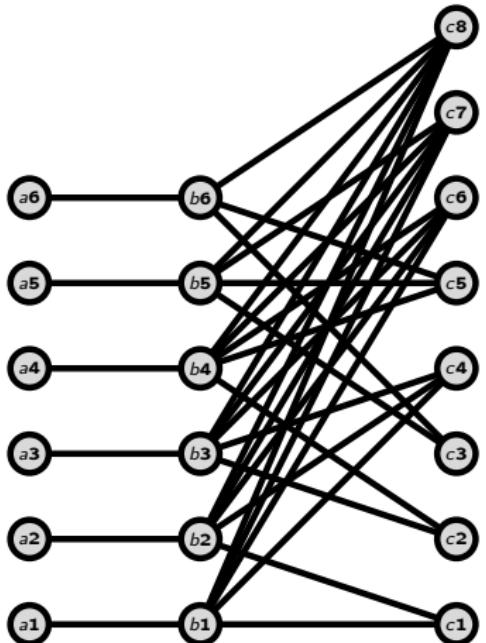
- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit
 $k' = \sum_{j=2}^k \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leq i \leq k\}$
- $k_i = \sum_{j=2}^i \lfloor k/j \rfloor$ für $i = 2, 3, \dots, k$
- $k_1 = 0$
- $E_i = \{\{c_{k_{i-1}+j}, b_{i*(j-1)+x}\} \mid 1 \leq j \leq \lfloor k/i \rfloor \wedge 1 \leq x \leq i\}$
für $i = 2, 3, \dots, k$
- In E_i sind $\lfloor k/i \rfloor$ Knoten aus C_k beteiligt.

Vertex Cover (Greedy)



- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit $k' = \sum_{j=2}^k \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leq i \leq k\}$
- $k_i = \sum_{j=2}^i \lfloor k/j \rfloor$ für $i = 2, 3, \dots, k$
- $k_1 = 0$
- $E_i = \{\{c_{k_{i-1}+j}, b_{i*(j-1)+x}\} \mid 1 \leq j \leq \lfloor k/i \rfloor \wedge 1 \leq x \leq i\}$
für $i = 2, 3, \dots, k$
- In E_i sind $\lfloor k/i \rfloor$ Knoten aus C_k beteiligt.

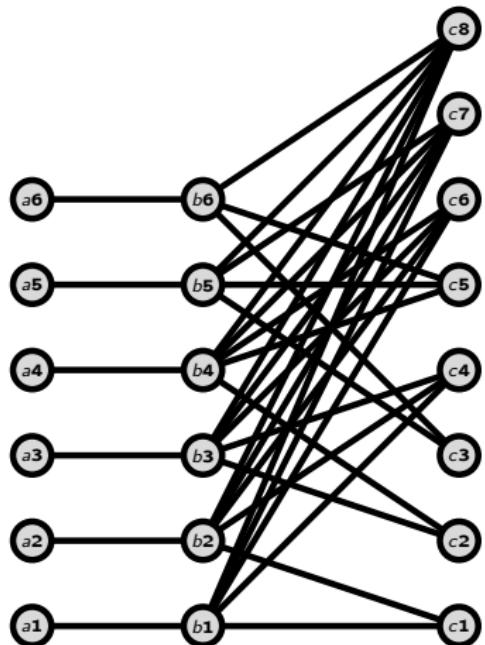
Vertex Cover (Greedy)



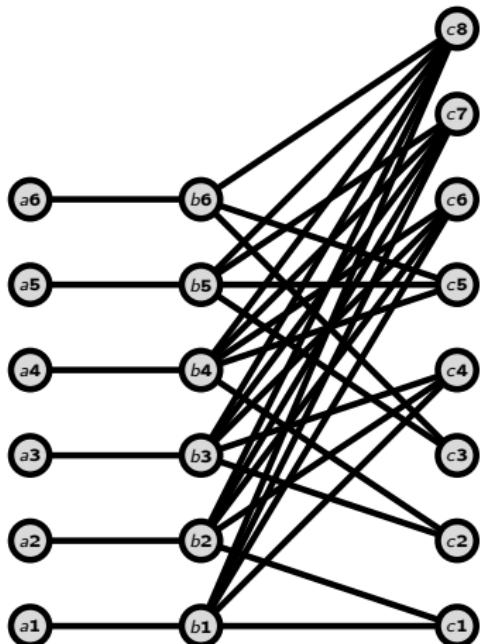
- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^k E_i)$
- $A_k = \{a_1, a_2, \dots, a_k\}$
- $B_k = \{b_1, b_2, \dots, b_k\}$
- $C_k = \{c_1, c_2, \dots, c_{k'}\}$ mit $k' = \sum_{j=2}^k \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leq i \leq k\}$
- $k_i = \sum_{j=2}^i \lfloor k/j \rfloor$ für $i = 2, 3, \dots, k$
- $k_1 = 0$
- $E_i = \{\{c_{k_{i-1}+j}, b_{i*(j-1)+x}\} \mid 1 \leq j \leq \lfloor k/i \rfloor \wedge 1 \leq x \leq i\}$ für $i = 2, 3, \dots, k$
- In E_i sind $\lfloor k/i \rfloor$ Knoten aus C_k beteiligt.

Vertex Cover (Greedy)

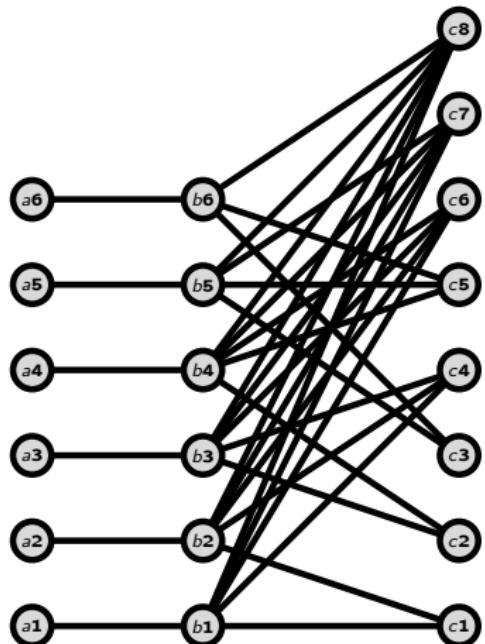
Vertex Cover (Greedy)



Vertex Cover (Greedy)

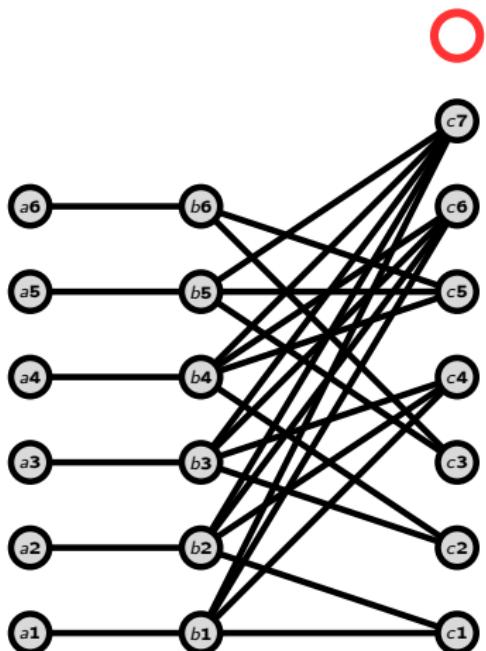


Vertex Cover (Greedy)



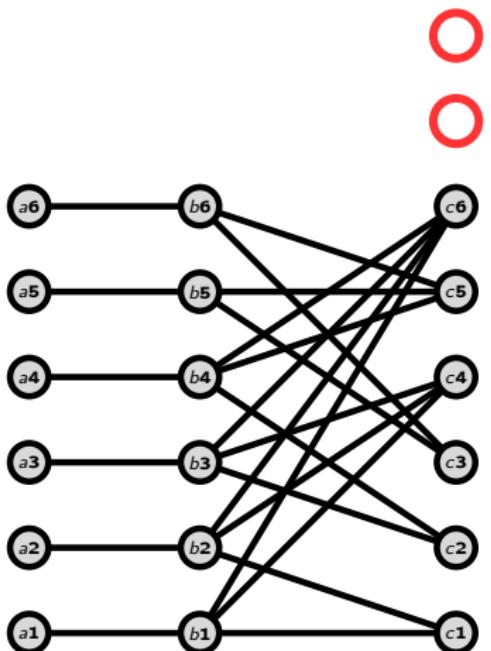
- Wähle Knoten in folgender Reihenfolge:
 - c_8 (Grad 6)

Vertex Cover (Greedy)



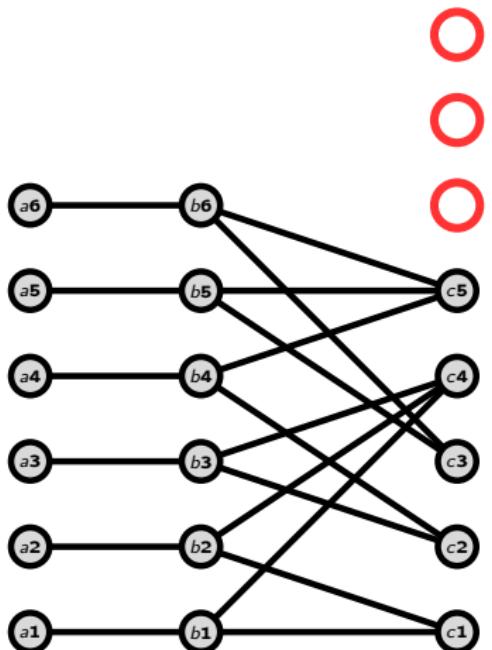
- Wähle Knoten in folgender Reihenfolge:
 - c_8 (Grad 6)
 - c_7 (Grad 5)

Vertex Cover (Greedy)



- Wähle Knoten in folgender Reihenfolge:
 - c_8 (Grad 6)
 - c_7 (Grad 5)
 - c_6 (Grad 4)

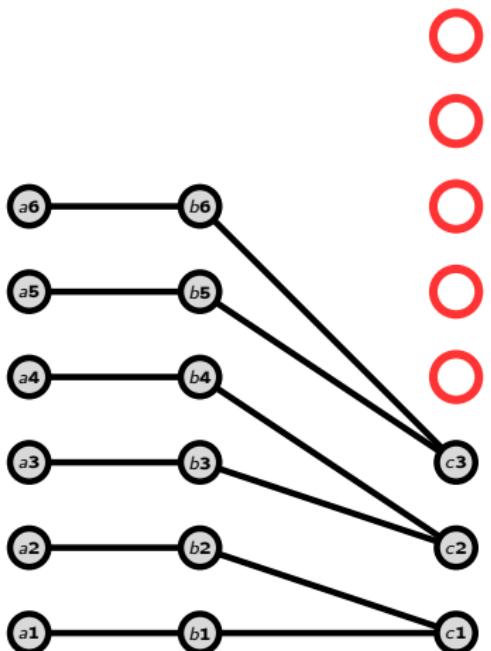
Vertex Cover (Greedy)



- Wähle Knoten in folgender Reihenfolge:

- c_8 (Grad 6)
- c_7 (Grad 5)
- c_6 (Grad 4)
- c_4, c_5 (Grad 3)

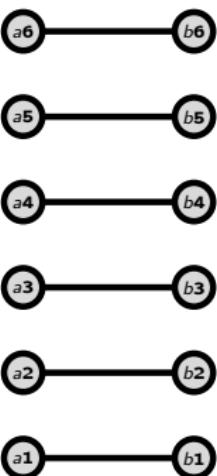
Vertex Cover (Greedy)



- Wähle Knoten in folgender Reihenfolge:

- c_8 (Grad 6)
 - c_7 (Grad 5)
 - c_6 (Grad 4)
 - c_4, c_5 (Grad 3)
 - c_1, c_2, c_3 (Grad 2)

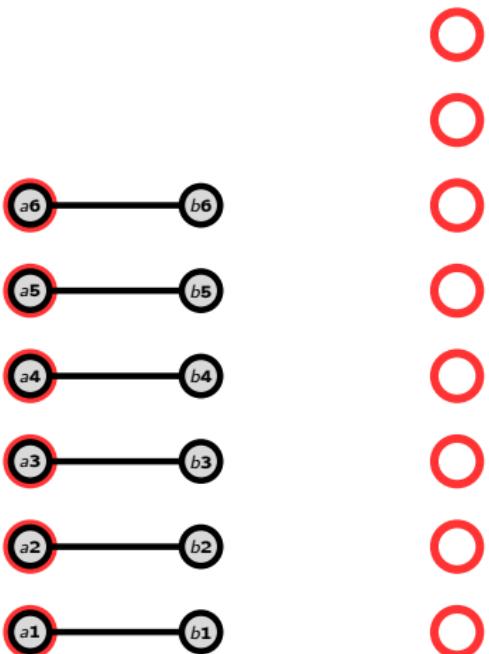
Vertex Cover (Greedy)



- Wähle Knoten in folgender Reihenfolge:

- c_8 (Grad 6)
- c_7 (Grad 5)
- c_6 (Grad 4)
- c_4, c_5 (Grad 3)
- c_1, c_2, c_3 (Grad 2)
- a_1, a_2, \dots, a_6 (Grad 1)

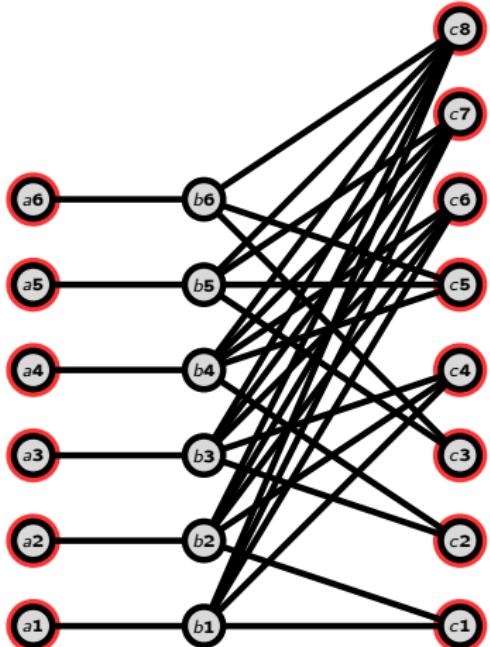
Vertex Cover (Greedy)



- Wähle Knoten in folgender Reihenfolge:

- c_8 (Grad 6)
- c_7 (Grad 5)
- c_6 (Grad 4)
- c_4, c_5 (Grad 3)
- c_1, c_2, c_3 (Grad 2)
- a_1, a_2, \dots, a_6 (Grad 1)

Vertex Cover (Greedy)



- Wähle Knoten in folgender Reihenfolge:

- c_8 (Grad 6)
- c_7 (Grad 5)
- c_6 (Grad 4)
- c_4, c_5 (Grad 3)
- c_1, c_2, c_3 (Grad 2)
- a_1, a_2, \dots, a_6 (Grad 1)

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

$$F(n) \geq \frac{|C_{greedy}(G_k)|}{|C_{opt}(G_k)|}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationssfaktor $F(n)$.

$$\begin{aligned} F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\ &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \end{aligned}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

$$\begin{aligned} F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\ &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \\ &= \frac{1}{k} \cdot \sum_{i=1}^k \left\lfloor \frac{k}{i} \right\rfloor \end{aligned}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

$$\begin{aligned}
 F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\
 &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \\
 &= \frac{1}{k} \cdot \sum_{i=1}^k \left\lfloor \frac{k}{i} \right\rfloor \\
 &\geq \frac{1}{k} \cdot \left(\sum_{i=1}^k \frac{k}{i} - (k - 2) \right)
 \end{aligned}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationssfaktor $F(n)$.

$$\begin{aligned}
 F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\
 &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \\
 &= \frac{1}{k} \cdot \sum_{i=1}^k \left\lfloor \frac{k}{i} \right\rfloor \\
 &\geq \frac{1}{k} \cdot \left(\sum_{i=1}^k \frac{k}{i} - (k - 2) \right) \\
 &\geq \sum_{i=1}^k \frac{1}{i} - 1
 \end{aligned}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

$$\begin{aligned} F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\ &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \\ &= \frac{1}{k} \cdot \sum_{i=1}^k \left\lfloor \frac{k}{i} \right\rfloor \\ &\geq \frac{1}{k} \cdot \left(\sum_{i=1}^k \frac{k}{i} - (k - 2) \right) \\ &\geq \sum_{i=1}^k \frac{1}{i} - 1 \\ &\geq \int_{i=1}^{k+1} \frac{dx}{x} - 1 \end{aligned}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationssfaktor $F(n)$.

$$\begin{aligned}
 F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\
 &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \\
 &= \frac{1}{k} \cdot \sum_{i=1}^k \left\lfloor \frac{k}{i} \right\rfloor \\
 &\geq \frac{1}{k} \cdot \left(\sum_{i=1}^k \frac{k}{i} - (k - 2) \right) \\
 &\geq \sum_{i=1}^k \frac{1}{i} - 1 \\
 &\geq \int_{i=1}^{k+1} \frac{dx}{x} - 1 \\
 &\geq \ln k - 1
 \end{aligned}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

$$\begin{aligned}
 F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\
 &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \\
 &= \frac{1}{k} \cdot \sum_{i=1}^k \left\lfloor \frac{k}{i} \right\rfloor \\
 &\geq \frac{1}{k} \cdot \left(\sum_{i=1}^k \frac{k}{i} - (k - 2) \right) \\
 &\geq \sum_{i=1}^k \frac{1}{i} - 1 \\
 &\geq \int_{i=1}^{k+1} \frac{dx}{x} - 1 \\
 &\geq \ln k - 1 \\
 &\geq \ln \Delta(G_k) - 1
 \end{aligned}$$

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationssfaktor $F(n)$.

$$\begin{aligned}
 F(n) &\geq \frac{|C_{\text{greedy}}(G_k)|}{|C_{\text{opt}}(G_k)|} \\
 &= \frac{1}{k} \cdot \left(k + \sum_{i=2}^k \left\lfloor \frac{k}{i} \right\rfloor \right) \\
 &= \frac{1}{k} \cdot \sum_{i=1}^k \left\lfloor \frac{k}{i} \right\rfloor \\
 &\geq \frac{1}{k} \cdot \left(\sum_{i=1}^k \frac{k}{i} - (k-2) \right) \\
 &\geq \sum_{i=1}^k \frac{1}{i} - 1 \\
 &\geq \int_{i=1}^{k+1} \frac{dx}{x} - 1 \\
 &\geq \ln k - 1 \\
 &\geq \ln \Delta(G_k) - 1
 \end{aligned}$$

- Damit kann dieser Greedyalgorithmus keinen konstanten Approximationssfaktor haben.

Aufbau der Idee

- Jede Kante muss überdeckt werden.

Aufbau der Idee

- Jede Kante muss überdeckt werden.
- Für die Abschätzung des Faktors brauchen wir eine untere Schranke.

Aufbau der Idee

- Jede Kante muss überdeckt werden.
- Für die Abschätzung des Faktors brauchen wir eine untere Schranke.
- Für zwei unabhängige Kanten braucht man zwei Knoten zur Überdeckung.

Aufbau der Idee

- Jede Kante muss überdeckt werden.
- Für die Abschätzung des Faktors brauchen wir eine untere Schranke.
- Für zwei unabhängige Kanten braucht man zwei Knoten zur Überdeckung.
- Für k paarweise unabhängige Kanten braucht man k Knoten zur Überdeckung.

Aufbau der Idee

- Jede Kante muss überdeckt werden.
- Für die Abschätzung des Faktors brauchen wir eine untere Schranke.
- Für zwei unabhängige Kanten braucht man zwei Knoten zur Überdeckung.
- Für k paarweise unabhängige Kanten braucht man k Knoten zur Überdeckung.
- Man weiß aber nicht, welcher Knoten von diesen k Kanten im Cover ist.

Aufbau der Idee

- Jede Kante muss überdeckt werden.
- Für die Abschätzung des Faktors brauchen wir eine untere Schranke.
- Für zwei unabhängige Kanten braucht man zwei Knoten zur Überdeckung.
- Für k paarweise unabhängige Kanten braucht man k Knoten zur Überdeckung.
- Man weiß aber nicht, welcher Knoten von diesen k Kanten im Cover ist.
- Idee: Wähle beide für einen Approximationsfaktor von zwei.

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabegraph $G = (V, E)$.

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabegraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabegraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabegraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabegraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabegraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
 - Damit ist M kein inklusions-maximales Matching, Widerspruch.

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabagraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
 - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabagraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
 - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
 - $M_{\max}(G)$ Maximales Matching von G .

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabagraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
 - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
 - $M_{\max}(G)$ Maximales Matching von G .
 - Schätze Approximationsfaktor ab:

$$\frac{2 \cdot |M|}{\tau(G)} \leq \frac{2 \cdot |M_{\max}(G)|}{\tau(G)} \quad \text{beachte: } |M_{\max}(G)| \geq |M|$$



Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabagraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
 - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
 - $M_{\max}(G)$ Maximales Matching von G .
 - Schätze Approximationsfaktor ab:

$$\frac{2 \cdot |M|}{\tau(G)} \leq \frac{2 \cdot |M_{\max}(G)|}{\tau(G)} \quad \text{beachte: } |M_{\max}(G)| \geq |M|$$

Vertex Cover

Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabagraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
 - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
 - $M_{\max}(G)$ Maximales Matching von G .
 - Schätze Approximationsfaktor ab:

$$\begin{aligned}\frac{2 \cdot |M|}{\tau(G)} &\leq \frac{2 \cdot |M_{\max}(G)|}{\tau(G)} \quad \text{beachte: } |M_{\max}(G)| \geq |M| \\ &\leq \frac{2 \cdot |M_{\max}(G)|}{|M_{\max}(G)|} \quad \text{beachte: } |M_{\max}(G)| \leq \tau(G)\end{aligned}$$

Vertex Cover

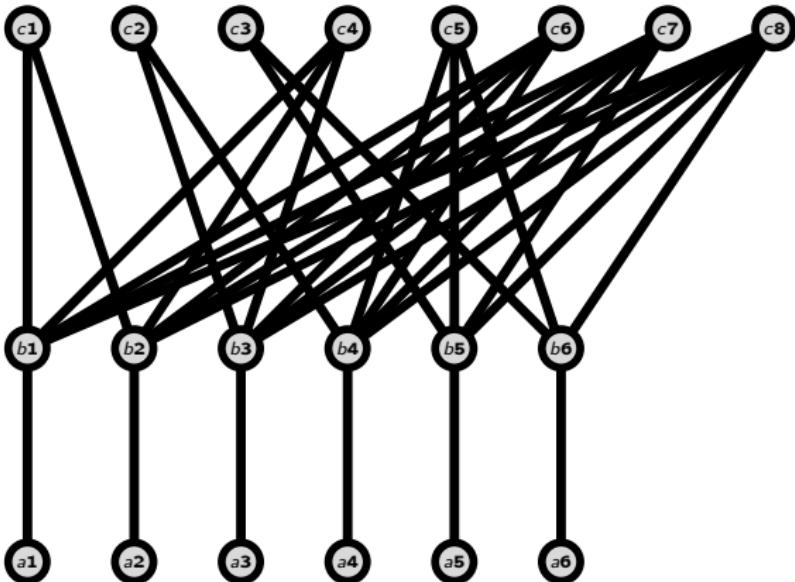
Theorem

Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

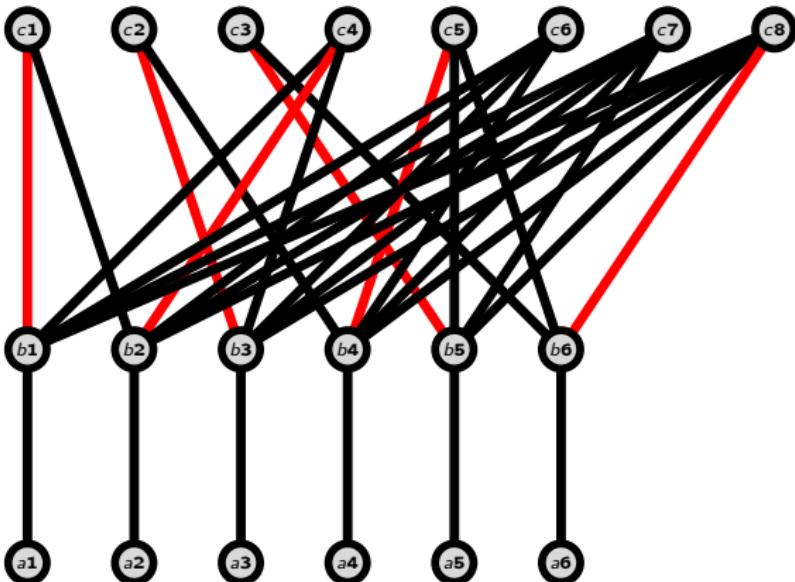
- ① Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabagraph $G = (V, E)$.
- ② Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.
- Zeige Korrektheit:
 - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
 - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
 - $M_{\max}(G)$ Maximales Matching von G .
 - Schätze Approximationsfaktor ab:

$$\begin{aligned}\frac{2 \cdot |M|}{\tau(G)} &\leq \frac{2 \cdot |M_{\max}(G)|}{\tau(G)} \quad \text{beachte: } |M_{\max}(G)| \geq |M| \\ &\leq \frac{2 \cdot |M_{\max}(G)|}{|M_{\max}(G)|} \quad \text{beachte: } |M_{\max}(G)| \leq \tau(G) \\ &= 2\end{aligned}$$

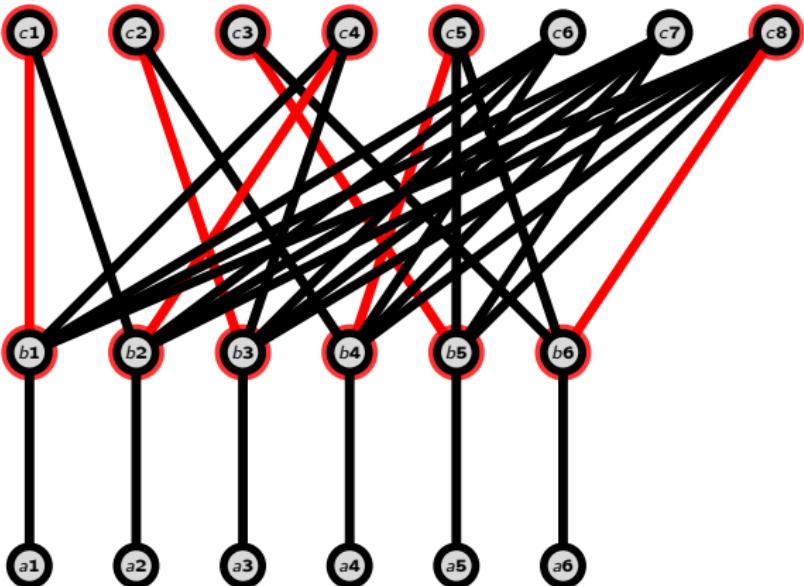
Beispiel



Beispiel



Beispiel



Vertex Cover

Theorem (Monien)

Das Vertex Cover Problem kann mit einem Faktor von $2 - \frac{\log \log n}{2 \cdot \log n}$ approximiert werden.

Vertex Cover

Theorem (Monien)

Das Vertex Cover Problem kann mit einem Faktor von $2 - \frac{\log \log n}{2 \cdot \log n}$ approximiert werden.

- Zusammenfassung:

Vertex Cover

Theorem (Monien)

Das Vertex Cover Problem kann mit einem Faktor von $2 - \frac{\log \log n}{2 \cdot \log n}$ approximiert werden.

- Zusammenfassung:
- Vertex Cover ist mit Faktor 2 approximierbar.

Vertex Cover

Theorem (Monien)

Das Vertex Cover Problem kann mit einem Faktor von $2 - \frac{\log \log n}{2 \cdot \log n}$ approximiert werden.

- Zusammenfassung:
- Vertex Cover ist mit Faktor 2 approximierbar.
- Clique ist mit keinem konstanten Faktor approximierbar.

Vertex Cover

Theorem (Monien)

Das Vertex Cover Problem kann mit einem Faktor von $2 - \frac{\log \log n}{2 \cdot \log n}$ approximiert werden.

- Zusammenfassung:
- Vertex Cover ist mit Faktor 2 approximierbar.
- Clique ist mit keinem konstanten Faktor approximierbar.
- Independent Set ist mit keinem konstanten Faktor approximierbar.

Vertex Cover

Theorem (Monien)

Das Vertex Cover Problem kann mit einem Faktor von $2 - \frac{\log \log n}{2 \cdot \log n}$ approximiert werden.

- Zusammenfassung:
- Vertex Cover ist mit Faktor 2 approximierbar.
- Clique ist mit keinem konstanten Faktor approximierbar.
- Independent Set ist mit keinem konstanten Faktor approximierbar.
- NP-vollständige Probleme unterscheiden sich in ihrer Approximierbarkeit.

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
 - $G = (V, E)$ Eingabe für Hamilton Kreis.

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E$ und

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E$ und
 - $c(e) = 2$ falls $e \notin E$.

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E'$ und
 - $c(e) = 2$ falls $e \notin E'$.
 - Falls G einen Hamiltonkreis hat, dann G' Tour mit Kosten n .

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E$ und
 - $c(e) = 2$ falls $e \notin E$.
 - Falls G einen Hamiltonkreis hat, dann G' Tour mit Kosten n .
 - Andernfalls hat G' eine Tour mit Kosten $\geq n + 1$.

TSP

Theorem

Sei $\alpha(n)$ eine polynomialzeit-berechenbare Funktion, $G = (V, E)$ und $n = |V|$.
Falls $\mathcal{P} \neq \mathcal{NP}$, gilt: TSP kann nicht in Polynomialzeit mit Faktor $\alpha(n)$ approximiert werden.

TSP

Theorem

Sei $\alpha(n)$ eine polynomialzeit-berechenbare Funktion, $G = (V, E)$ und $n = |V|$.
Falls $\mathcal{P} \neq \mathcal{NP}$, gilt: TSP kann nicht in Polynomialzeit mit Faktor $\alpha(n)$ approximiert werden.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).

TSP

Theorem

Sei $\alpha(n)$ eine polynomialzeit-berechenbare Funktion, $G = (V, E)$ und $n = |V|$.
Falls $\mathcal{P} \neq \mathcal{NP}$, gilt: TSP kann nicht in Polynomialzeit mit Faktor $\alpha(n)$ approximiert werden.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
- Gleiche Reduktion zeigt obigen Satz:
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E$ und

TSP

Theorem

Sei $\alpha(n)$ eine polynomialzeit-berechenbare Funktion, $G = (V, E)$ und $n = |V|$. Falls $\mathcal{P} \neq \mathcal{NP}$, gilt: TSP kann nicht in Polynomialzeit mit Faktor $\alpha(n)$ approximiert werden.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
- Gleiche Reduktion zeigt obigen Satz:
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E$ und
 - $c(e) = \alpha(n) \cdot n + 1$ falls $e \notin E$.

TSP

Theorem

Sei $\alpha(n)$ eine polynomialzeit-berechenbare Funktion, $G = (V, E)$ und $n = |V|$.
Falls $\mathcal{P} \neq \mathcal{NP}$, gilt: TSP kann nicht in Polynomialzeit mit Faktor $\alpha(n)$ approximiert werden.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
- Gleiche Reduktion zeigt obigen Satz:
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E$ und
 - $c(e) = \alpha(n) \cdot n + 1$ falls $e \notin E$.
 - Falls G einen Hamiltonkreis hat, dann G' Tour mit Kosten n .

TSP

Theorem

Sei $\alpha(n)$ eine polynomialzeit-berechenbare Funktion, $G = (V, E)$ und $n = |V|$. Falls $\mathcal{P} \neq \mathcal{NP}$, gilt: TSP kann nicht in Polynomialzeit mit Faktor $\alpha(n)$ approximiert werden.

- Problem ist in \mathcal{NPC} (Reduktion von Hamilton Kreis).
- Gleiche Reduktion zeigt obigen Satz:
 - $G = (V, E)$ Eingabe für Hamilton Kreis.
 - Eingabe für TSP ist Clique $C = (V, E')$ mit:
 - $c(e) = 1$ falls $e \in E$ und
 - $c(e) = \alpha(n) \cdot n + 1$ falls $e \notin E$.
 - Falls G einen Hamiltonkreis hat, dann G' Tour mit Kosten n .
 - Andernfalls hat G' eine Tour mit Kosten $\geq n - 1 + \alpha(n) \cdot n$.

Δ-TSP

Definition (Δ -TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$

- mit: $\forall v, w \in V : e = \{v, w\} \in E, c(e) \geq 0$ und
- mit: $\forall v, w, z \in V : c(v, z) \leq c(v, w) + c(w, z)$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

Δ-TSP

Definition (Δ -TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$

- mit: $\forall v, w \in V : e = \{v, w\} \in E, c(e) \geq 0$ und
- mit: $\forall v, w, z \in V : c(v, z) \leq c(v, w) + c(w, z)$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

Δ-TSP

Definition (Δ -TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$

- mit: $\forall v, w \in V : e = \{v, w\} \in E, c(e) \geq 0$ und
- mit: $\forall v, w, z \in V : c(v, z) \leq c(v, w) + c(w, z)$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

- Obige Eigenschaft auf c wird Metrik genannt.

Δ-TSP

Definition (Δ -TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$

- mit: $\forall v, w \in V : e = \{v, w\} \in E, c(e) \geq 0$ und
- mit: $\forall v, w, z \in V : c(v, z) \leq c(v, w) + c(w, z)$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

- Obige Eigenschaft auf c wird Metrik genannt.
- Motivation: Die Abstände von Punkten in der Ebene bilden eine Metrik.

Δ-TSP

Definition (Δ -TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$

- mit: $\forall v, w \in V : e = \{v, w\} \in E, c(e) \geq 0$ und
- mit: $\forall v, w, z \in V : c(v, z) \leq c(v, w) + c(w, z)$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

- Obige Eigenschaft auf c wird Metrik genannt.
- Motivation: Die Abstände von Punkten in der Ebene bilden eine Metrik.

Δ-TSP

Definition (Δ-TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$

- mit: $\forall v, w \in V : e = \{v, w\} \in E, c(e) \geq 0$ und
- mit: $\forall v, w, z \in V : c(v, z) \leq c(v, w) + c(w, z)$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leq L$.

- Obige Eigenschaft auf c wird Metrik genannt.
- Motivation: Die Abstände von Punkten in der Ebene bilden eine Metrik.

Theorem

Δ-TSP mit Gewichtsfunktion $c \mapsto \{1, 2\}$ ist in NP.

Beweis: Reduktion von Hamilton Kreis auf planare Graphen.

Aufbau der Idee

- Suche untere Schranke:

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)

Aufbau der Idee

- Suche untere Schranke:

- TSP-Tour ist ein spannender Graph (enthält alle Knoten)
- Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.

Aufbau der Idee

- Suche untere Schranke:

- TSP-Tour ist ein spannender Graph (enthält alle Knoten)
- Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
- Idee: Wähle minimalen Spannbaum!

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
 - D.h. jede Abkürzung verringert die Kosten.

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
 - D.h. jede Abkürzung verringert die Kosten.
 - Damit kann ein beliebiger Kreis, der jeden Knoten mindestens einmal besucht, zu einer kostengünstigeren TSP-Tour umgeformt werden.

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
 - D.h. jede Abkürzung verringert die Kosten.
 - Damit kann ein beliebiger Kreis, der jeden Knoten mindestens einmal besucht, zu einer kostengünstigeren TSP-Tour umgeformt werden.
 - Idee: Konstruiere daher einen Eulerkreis (besucht alle Kanten).

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
 - D.h. jede Abkürzung verringert die Kosten.
 - Damit kann ein beliebiger Kreis, der jeden Knoten mindestens einmal besucht, zu einer kostengünstigeren TSP-Tour umgeformt werden.
 - Idee: Konstruiere daher einen Eulerkreis (besucht alle Kanten).
- $G = (V, E)$ ist Eulergraph (enthält Eulerkreis):

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
 - D.h. jede Abkürzung verringert die Kosten.
 - Damit kann ein beliebiger Kreis, der jeden Knoten mindestens einmal besucht, zu einer kostengünstigeren TSP-Tour umgeformt werden.
 - Idee: Konstruiere daher einen Eulerkreis (besucht alle Kanten).
- $G = (V, E)$ ist Eulergraph (enthält Eulerkreis):
 - G ist zusammenhängend.

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
 - D.h. jede Abkürzung verringert die Kosten.
 - Damit kann ein beliebiger Kreis, der jeden Knoten mindestens einmal besucht, zu einer kostengünstigeren TSP-Tour umgeformt werden.
 - Idee: Konstruiere daher einen Eulerkreis (besucht alle Kanten).
- $G = (V, E)$ ist Eulergraph (enthält Eulerkreis):
 - G ist zusammenhängend.
 - Alle Knoten haben geraden Grad.

Aufbau der Idee

- Suche untere Schranke:
 - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
 - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
 - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
 - D.h. jede Abkürzung verringert die Kosten.
 - Damit kann ein beliebiger Kreis, der jeden Knoten mindestens einmal besucht, zu einer kostengünstigeren TSP-Tour umgeformt werden.
 - Idee: Konstruiere daher einen Eulerkreis (besucht alle Kanten).
- $G = (V, E)$ ist Eulergraph (enthält Eulerkreis):
 - G ist zusammenhängend.
 - Alle Knoten haben geraden Grad.
- Kombination: Mit Hilfe des Spannbaums wird Eulerkreis konstruiert.

Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

- ① Bestimme minimalen Spannbaum T von G .

Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

- ① Bestimme minimalen Spannbaum T von G .
- ② Verdoppele die Kanten von T und erzeuge Graphen T' .

Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

- ① Bestimme minimalen Spannbaum T von G .
- ② Verdoppele die Kanten von T und erzeuge Graphen T' .
- ③ Damit sind alle Knotengrade in T' gerade (da verdoppelt).

Approximation

Theorem

Δ-TSP ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

- ① Bestimme minimalen Spannbaum T von G .
- ② Verdoppele die Kanten von T und erzeuge Graphen T' .
- ③ Damit sind alle Knotengrade in T' gerade (da verdoppelt).
- ④ Bestimme in T' einen Euler-Kreis C' .

Approximation

Theorem

Δ-TSP ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

- ① Bestimme minimalen Spannbaum T von G .
- ② Verdoppele die Kanten von T und erzeuge Graphen T' .
- ③ Damit sind alle Knotengrade in T' gerade (da verdoppelt).
- ④ Bestimme in T' einen Euler-Kreis C' .
- ⑤ Verkürze C' durch Überspringen doppelter Knoten.

Approximation

Theorem

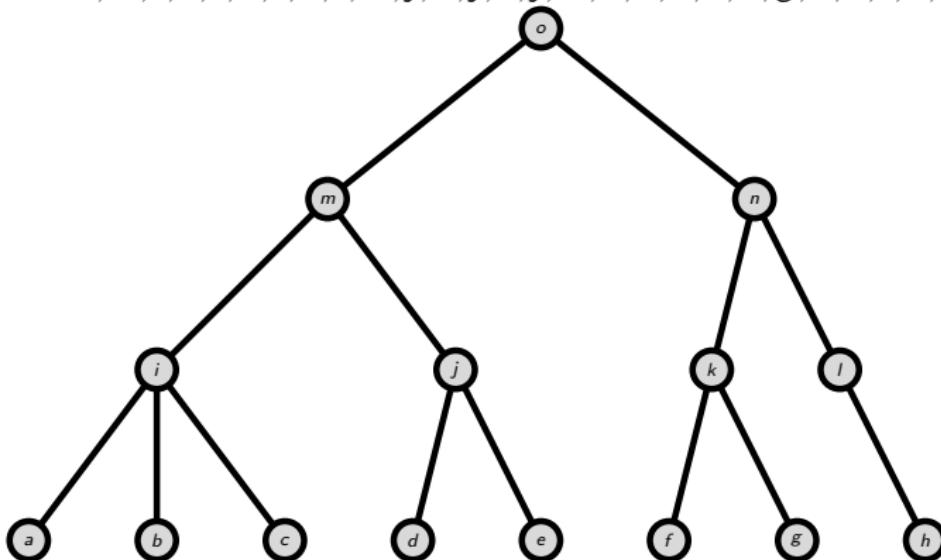
$\Delta\text{-TSP}$ ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

- ① Bestimme minimalen Spannbaum T von G .
- ② Verdoppele die Kanten von T und erzeuge Graphen T' .
- ③ Damit sind alle Knotengrade in T' gerade (da verdoppelt).
- ④ Bestimme in T' einen Euler-Kreis C' .
- ⑤ Verkürze C' durch Überspringen doppelter Knoten.
- ⑥ Dadurch entsteht Kreis C , gebe C aus.

Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

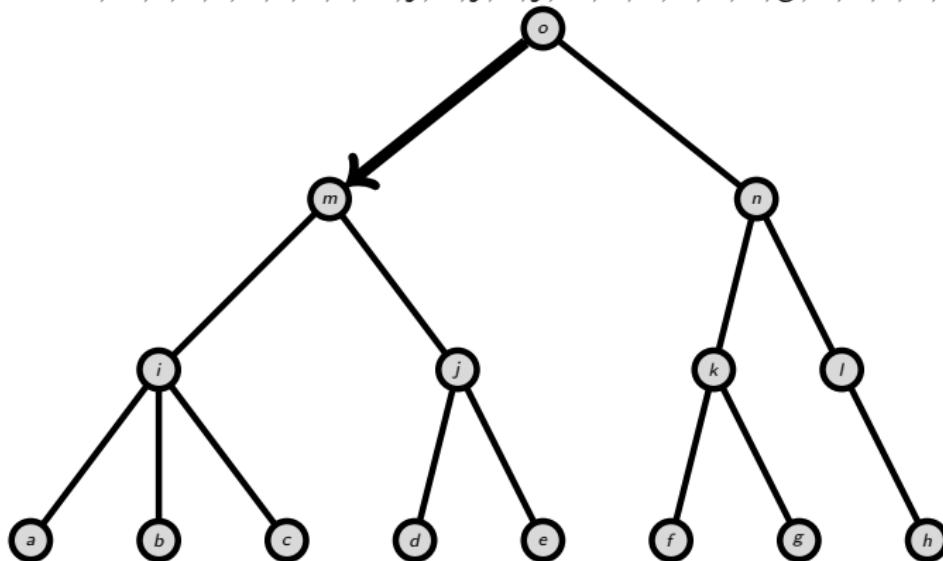
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

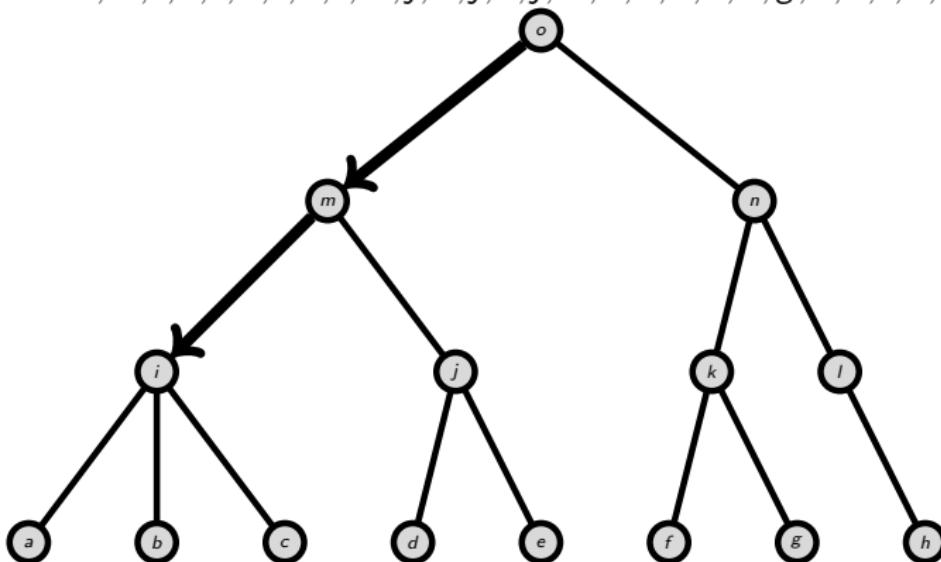
Eulertour: ***o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.***



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

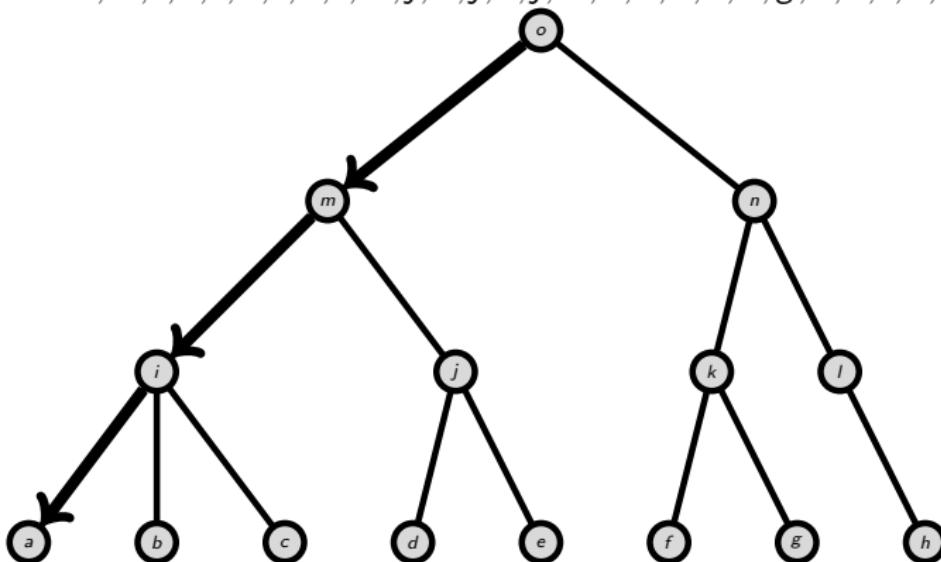
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o*.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

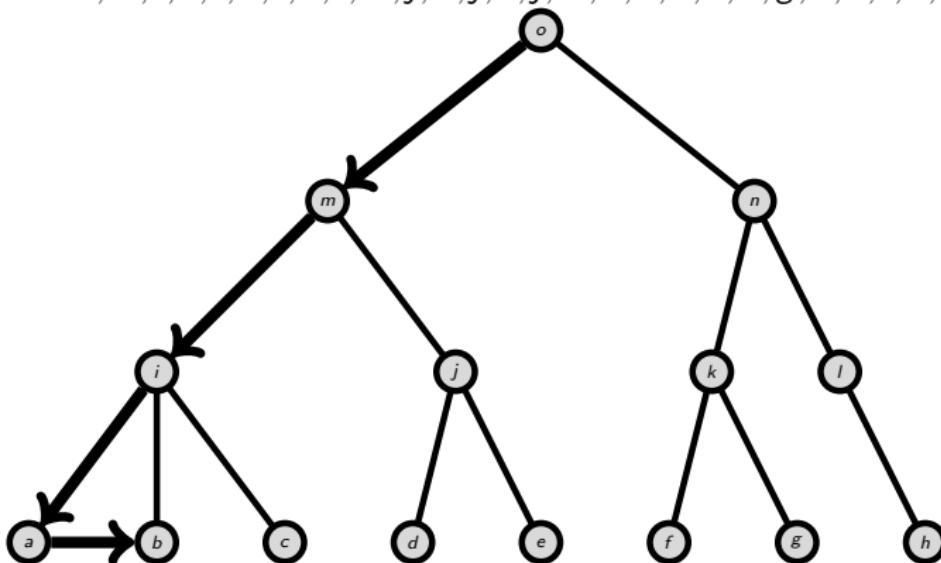
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o*.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

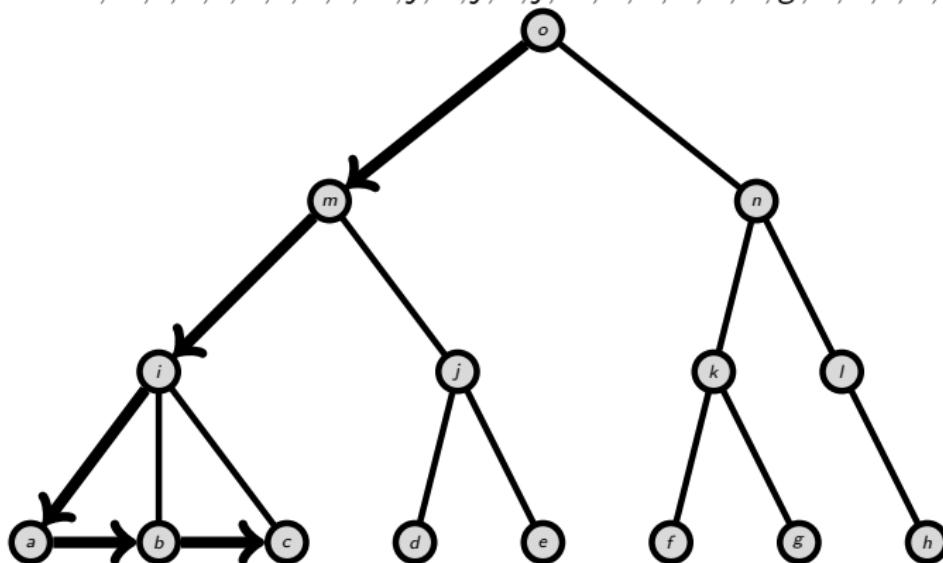
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o*.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

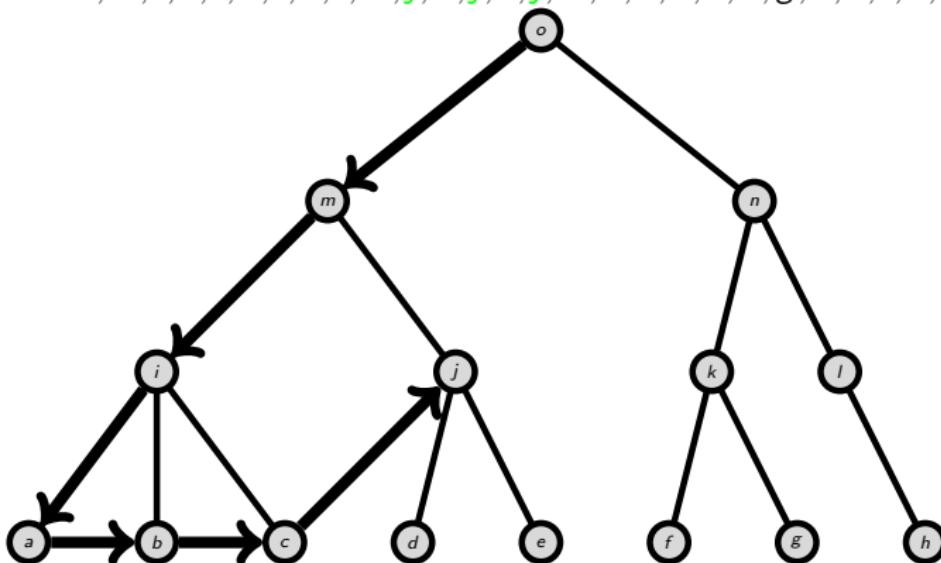
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o*.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

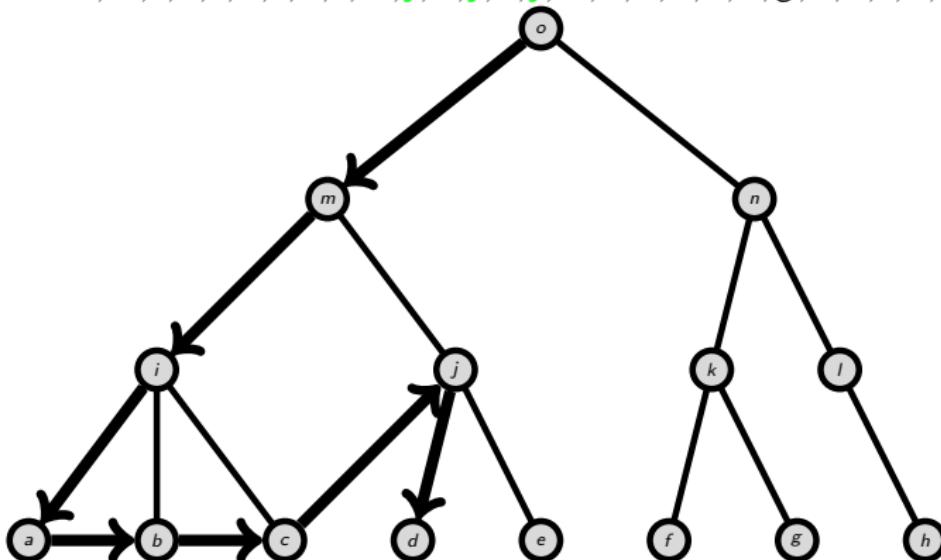
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

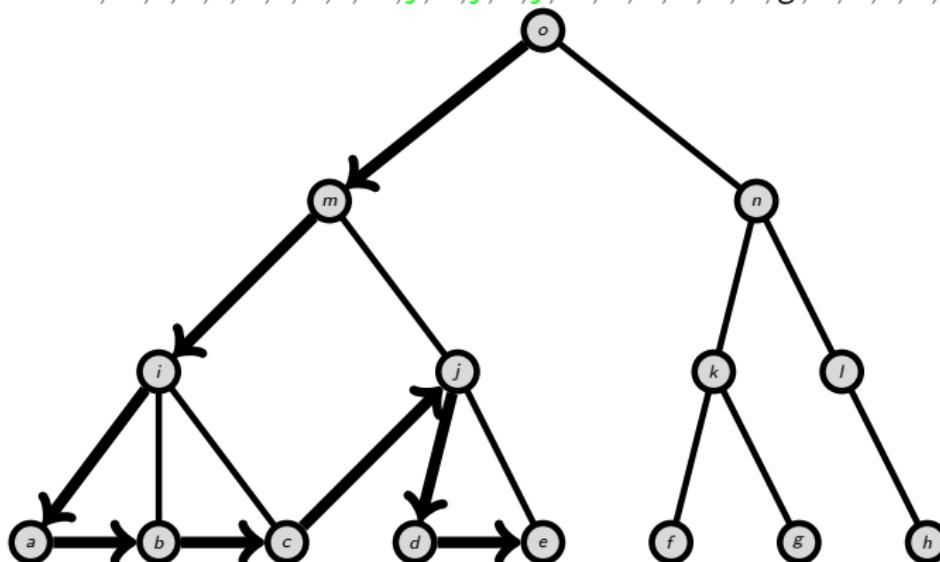
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

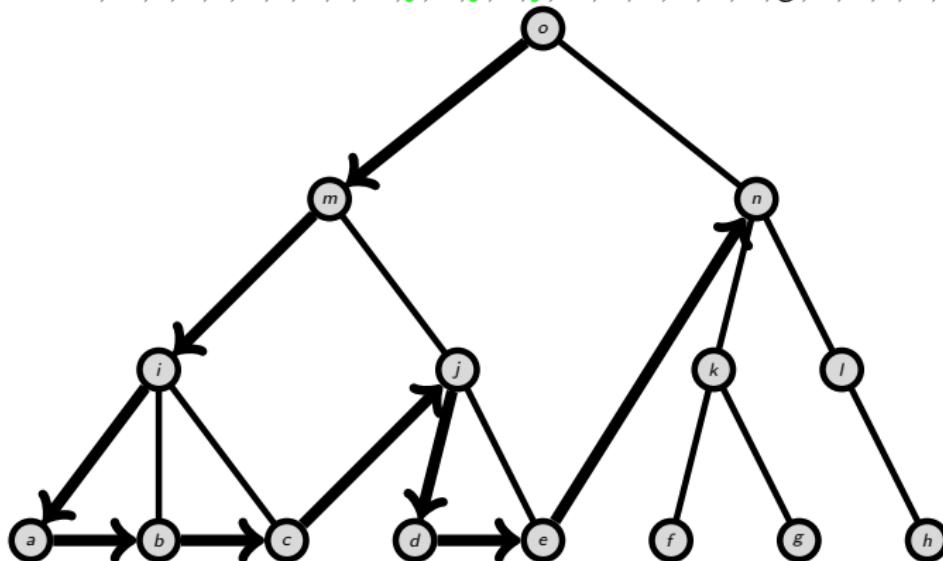
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

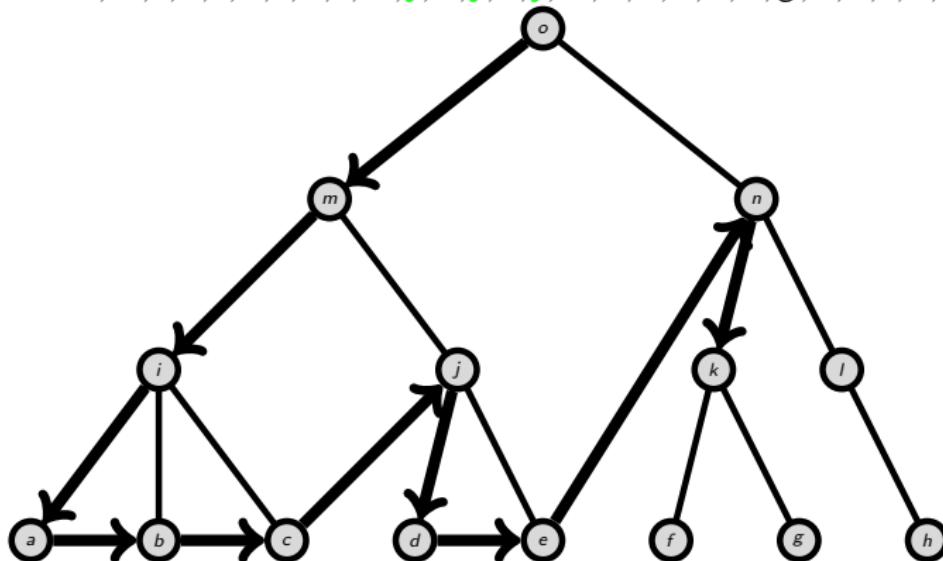
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

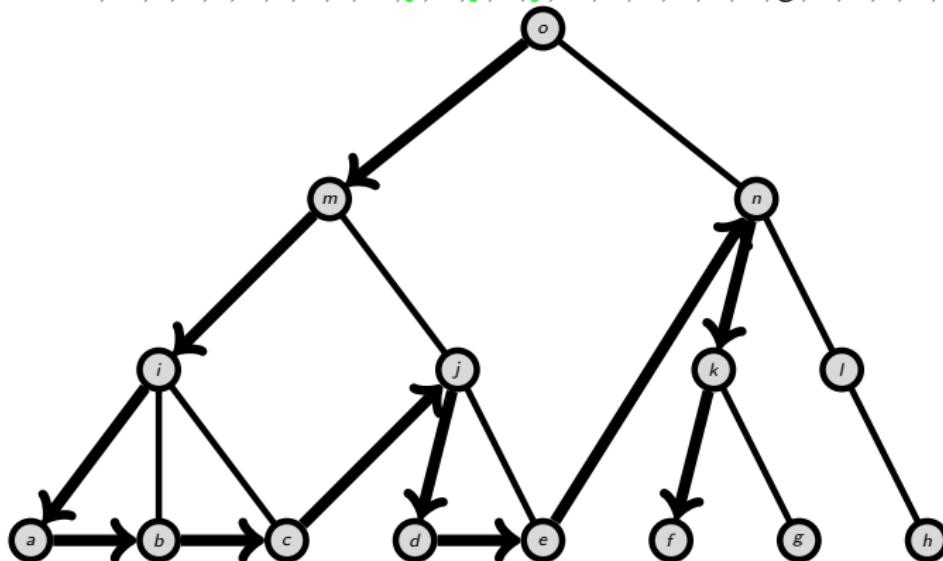
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

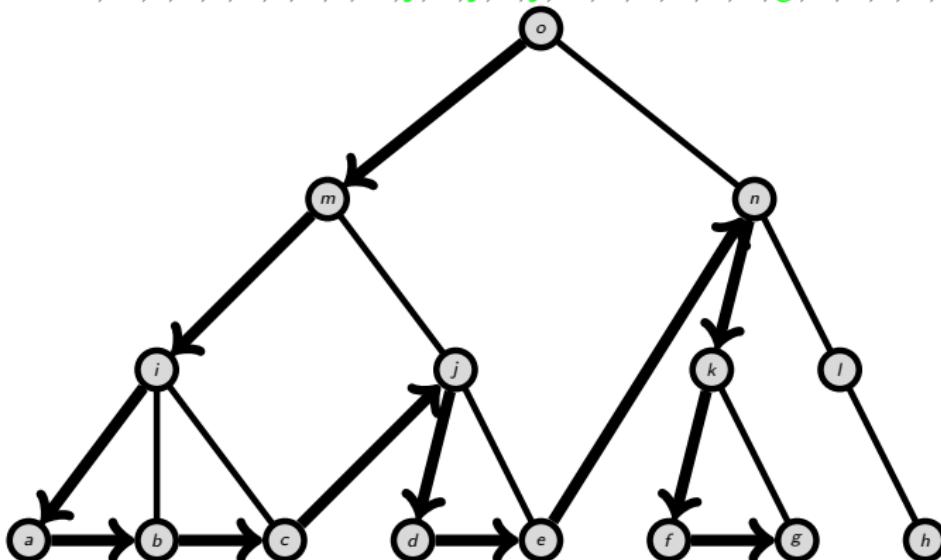
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

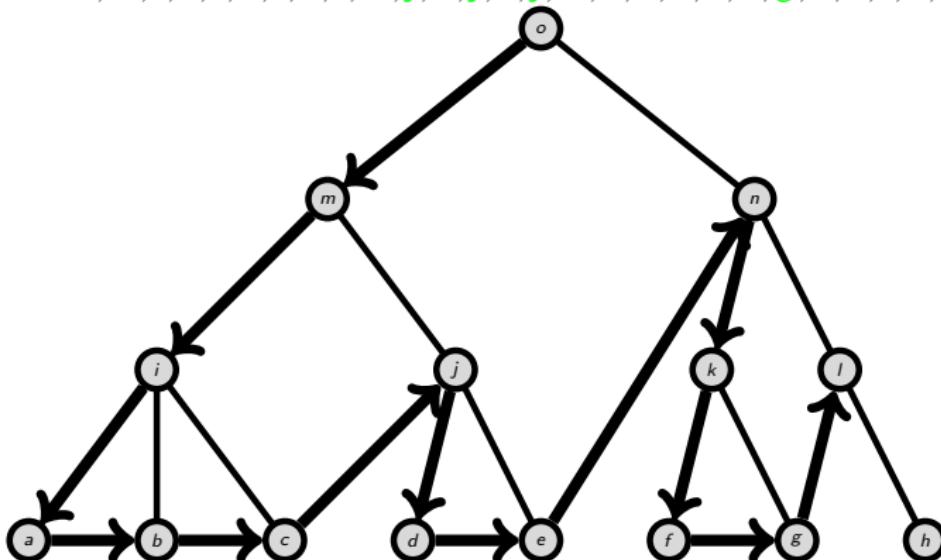
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

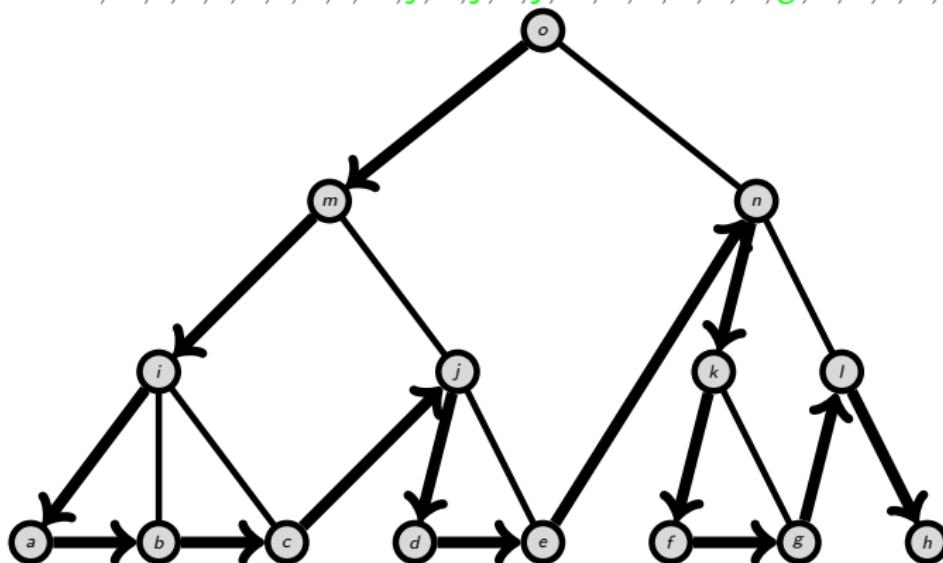
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

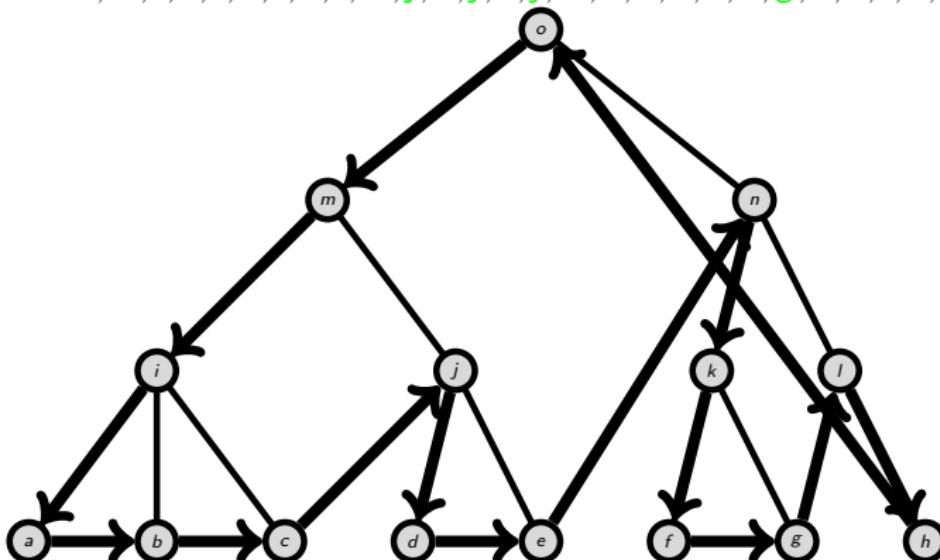
Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

Eulertour: *o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o.*



Approximation (Beweis)

 $G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.

Approximation (Beweis)

$G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.

Approximation (Beweis)

$G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.

Approximation (Beweis)

$G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum.

Approximation (Beweis)

 $G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leq c(C') = 2 \cdot c(T)$.

Approximation (Beweis)

 $G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leq c(C') = 2 \cdot c(T)$.
 - Da die Kanten verdoppelt worden sind.

Approximation (Beweis)

$G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leq c(C') = 2 \cdot c(T)$.
 - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)}$

Approximation (Beweis)

 $G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leq c(C') = 2 \cdot c(T)$.
 - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leq \frac{2 \cdot c(T)}{c(C^*)}$

Approximation (Beweis)

 $G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leq c(C') = 2 \cdot c(T)$.
 - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leq \frac{2 \cdot c(T)}{c(C^*)} \leq \frac{2 \cdot c(C^*)}{c(C^*)} = 2$

Approximation (Beweis)

$G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G ,
 T' mit verdoppelte Kanten,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leq c(C') = 2 \cdot c(T)$.
 - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leq \frac{2 \cdot c(T)}{c(C^*)} \leq \frac{2 \cdot c(C^*)}{c(C^*)} = 2$
- Laufzeit $O(n^2 \log n)$.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdoppele" den Knotengrad nur von den ungeraden Knoten.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdoppele" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdoppele" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdoppele" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdopple" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.
- Geht nur, wenn die Anzahl der ungeraden Knoten gerade ist:

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdopple" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.
- Geht nur, wenn die Anzahl der ungeraden Knoten gerade ist:
 - Zählen wir die Kombinationen X :
Knoten v inzident zu Kante $\{v, w\}$:

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdopple" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.
- Geht nur, wenn die Anzahl der ungeraden Knoten gerade ist:
 - Zählen wir die Kombinationen X :
Knoten v inzident zu Kante $\{v, w\}$:
 - Aus der Sicht der Kanten: $|X| = 2 \cdot |E|$.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdopple" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.
- Geht nur, wenn die Anzahl der ungeraden Knoten gerade ist:
 - Zählen wir die Kombinationen X :
Knoten v inzident zu Kante $\{v, w\}$:
 - Aus der Sicht der Kanten: $|X| = 2 \cdot |E|$.
 - Aus der Sicht der Knoten: $|X| = \sum_{v \in V} \delta(v)$.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdopple" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.
- Geht nur, wenn die Anzahl der ungeraden Knoten gerade ist:
 - Zählen wir die Kombinationen X :
Knoten v inzident zu Kante $\{v, w\}$:
 - Aus der Sicht der Kanten: $|X| = 2 \cdot |E|$.
 - Aus der Sicht der Knoten: $|X| = \sum_{v \in V} \delta(v)$.
 - Sei U die Menge der Knoten mit ungeradem Grad.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdopple" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.
- Geht nur, wenn die Anzahl der ungeraden Knoten gerade ist:
 - Zählen wir die Kombinationen X :
Knoten v inzident zu Kante $\{v, w\}$:
 - Aus der Sicht der Kanten: $|X| = 2 \cdot |E|$.
 - Aus der Sicht der Knoten: $|X| = \sum_{v \in V} \delta(v)$.
 - Sei U die Menge der Knoten mit ungeradem Grad.
 - $2 \cdot |E| = \sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v)$.

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .
- Seien U die Knoten von ungeraden Grad in T .

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .
- Seien U die Knoten von ungeraden Grad in T .
- Beachte $|U|$ ist gerade: $\sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v) = 2|E|$.

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .
- Seien U die Knoten von ungeraden Grad in T .
- Beachte $|U|$ ist gerade: $\sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v) = 2|E|$.
- Bestimme kostenminimales Matching M zwischen den Knoten aus U .

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .
- Seien U die Knoten von ungeraden Grad in T .
- Beachte $|U|$ ist gerade: $\sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v) = 2|E|$.
- Bestimme kostenminimales Matching M zwischen den Knoten aus U .
- Bestimme in $T \cup M$ einen Eulerkreis C' .

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .
- Seien U die Knoten von ungeraden Grad in T .
- Beachte $|U|$ ist gerade: $\sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v) = 2|E|$.
- Bestimme kostenminimales Matching M zwischen den Knoten aus U .
- Bestimme in $T \cup M$ einen Eulerkreis C' .
- Verkürze C' durch Überspringen doppelter Knoten.

1.5-Approximation

Theorem

$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .
- Seien U die Knoten von ungeraden Grad in T .
- Beachte $|U|$ ist gerade: $\sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v) = 2|E|$.
- Bestimme kostenminimales Matching M zwischen den Knoten aus U .
- Bestimme in $T \cup M$ einen Eulerkreis C' .
- Verkürze C' durch Überspringen doppelter Knoten.
- Dadurch entsteht Kreis C .

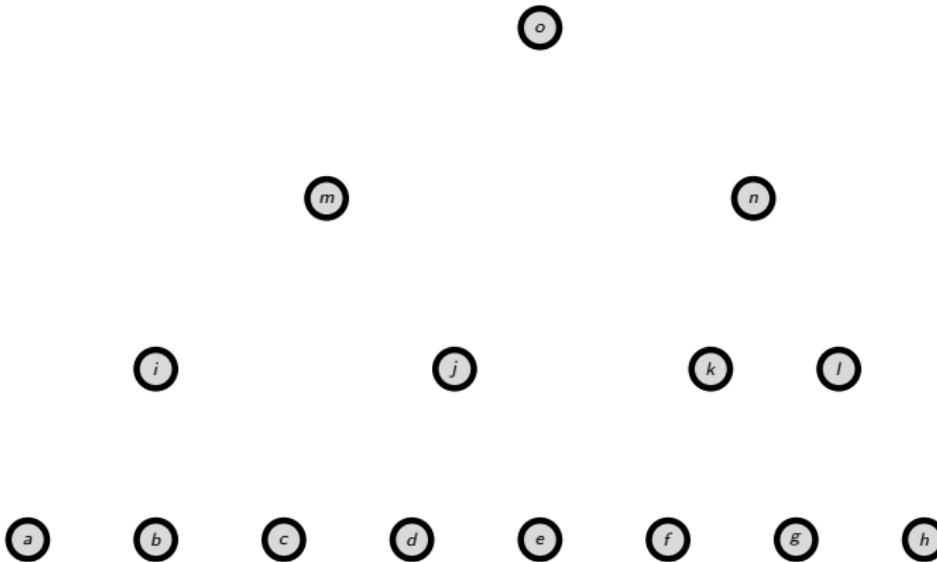
1.5-Approximation

Theorem

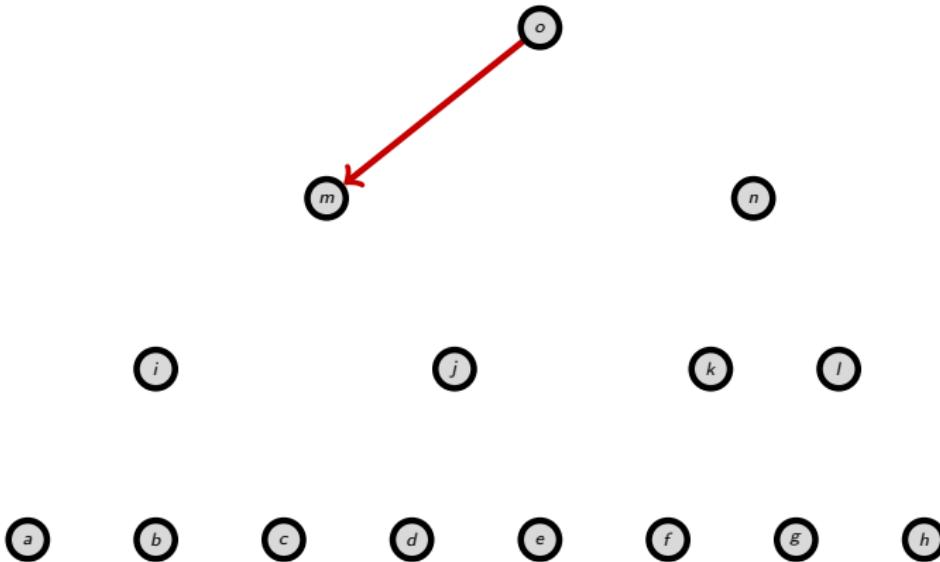
$\Delta\text{-TSP}$ ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G .
- Seien U die Knoten von ungeraden Grad in T .
- Beachte $|U|$ ist gerade: $\sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v) = 2|E|$.
- Bestimme kostenminimales Matching M zwischen den Knoten aus U .
- Bestimme in $T \cup M$ einen Eulerkreis C' .
- Verkürze C' durch Überspringen doppelter Knoten.
- Dadurch entsteht Kreis C .
- Gebe C aus.

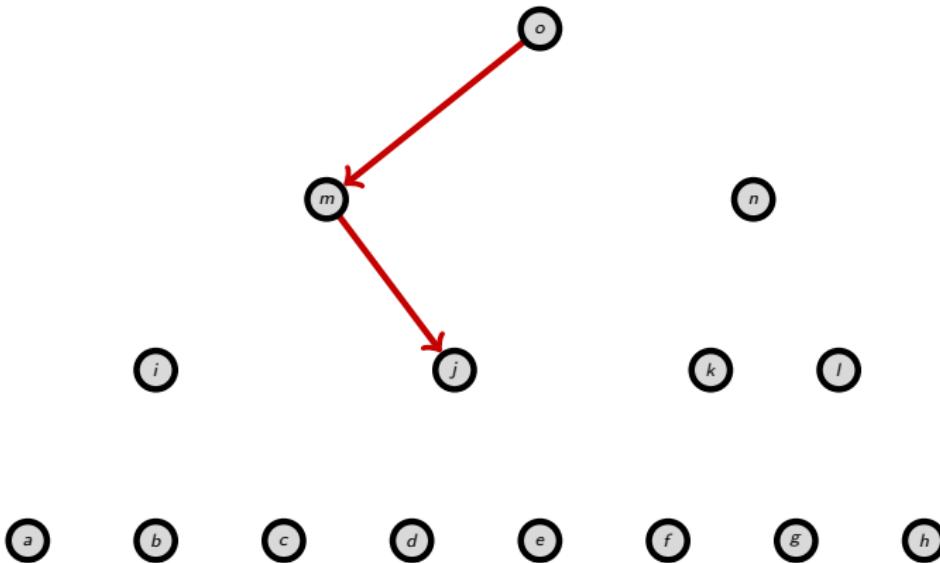
Beispiel (Bestimmen der Eulertour)



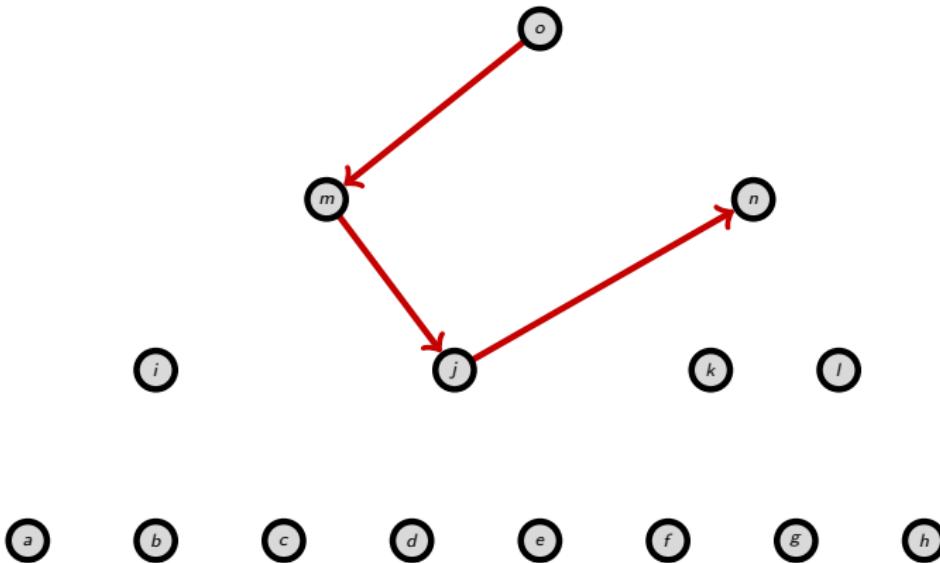
Beispiel (Bestimmen der Eulertour)



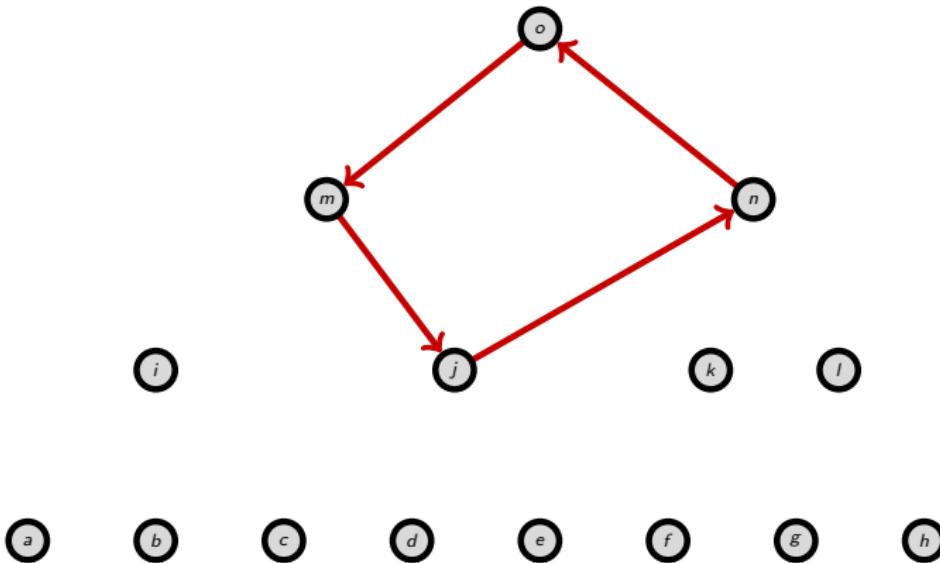
Beispiel (Bestimmen der Eulertour)



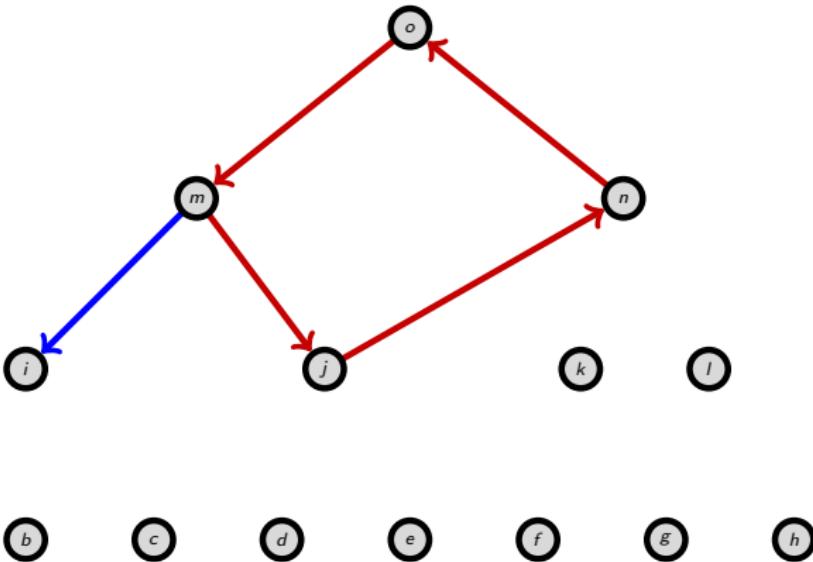
Beispiel (Bestimmen der Eulertour)



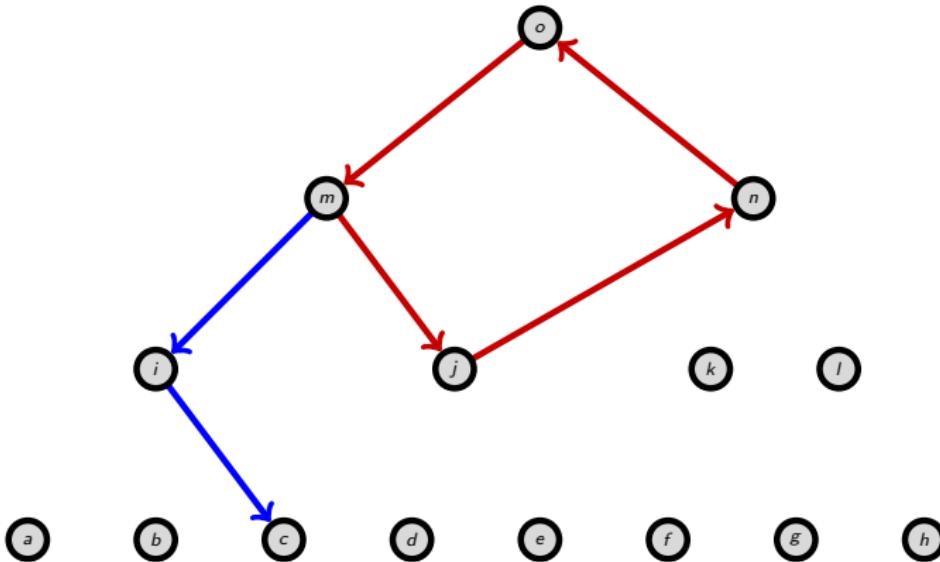
Beispiel (Bestimmen der Eulertour)



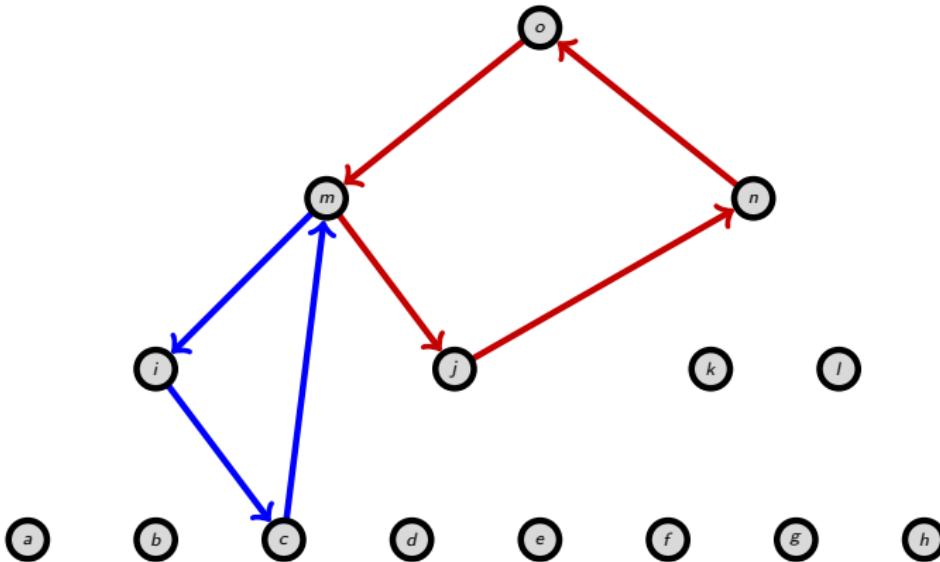
Beispiel (Bestimmen der Eulertour)



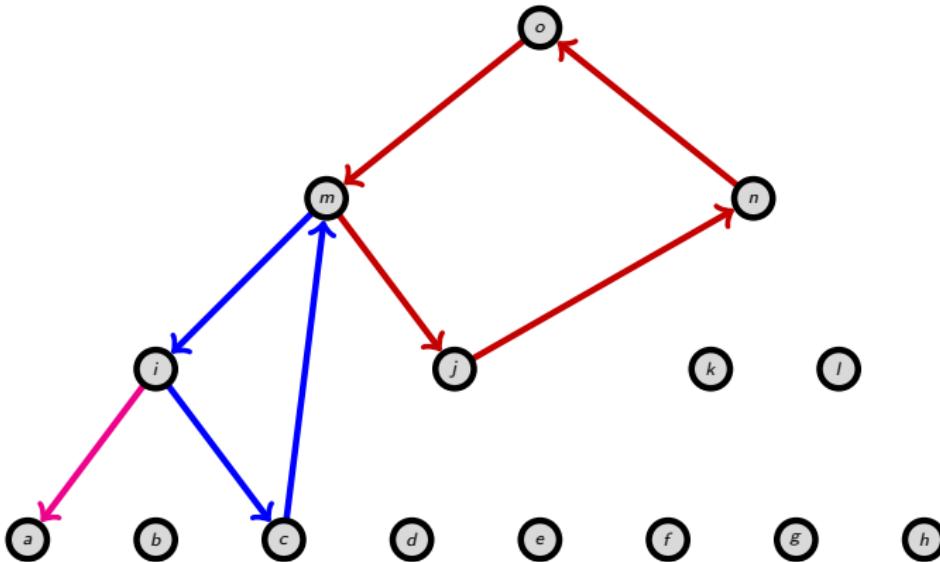
Beispiel (Bestimmen der Eulertour)



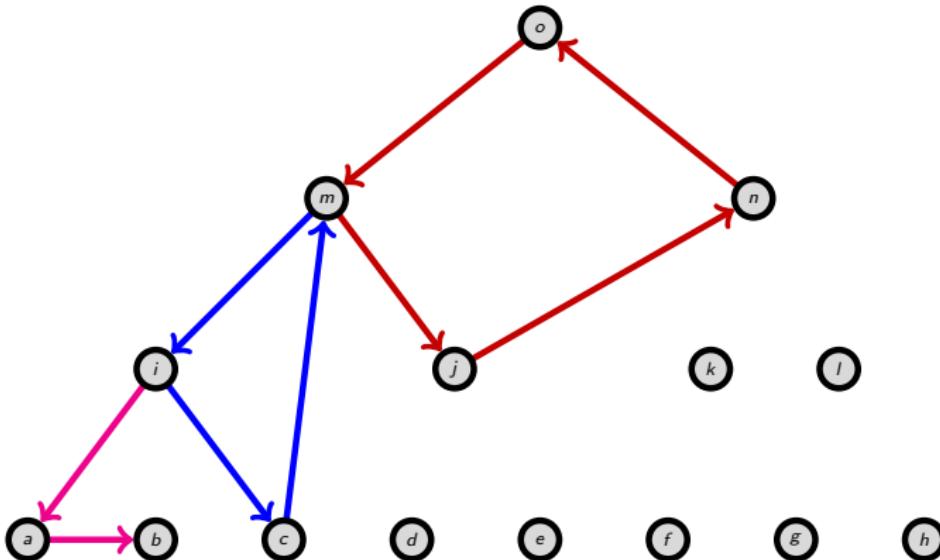
Beispiel (Bestimmen der Eulertour)



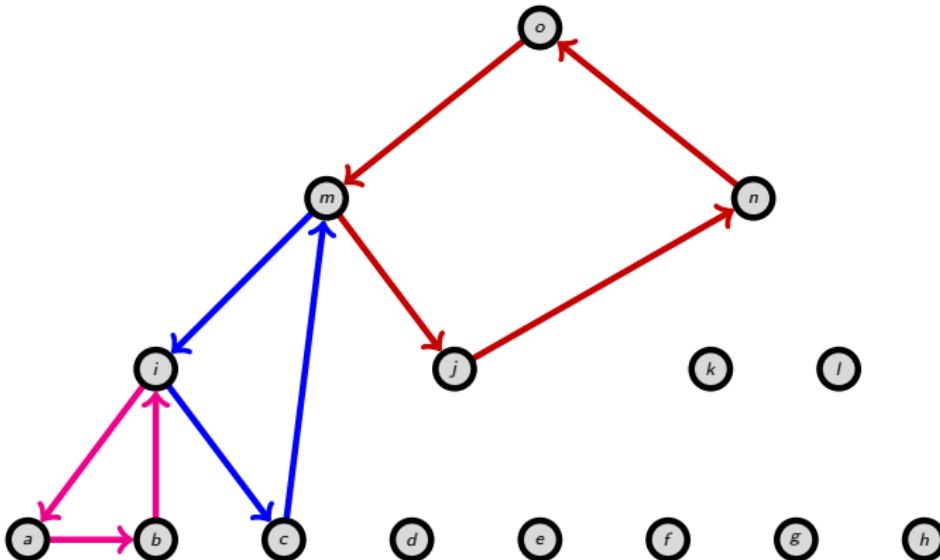
Beispiel (Bestimmen der Eulertour)



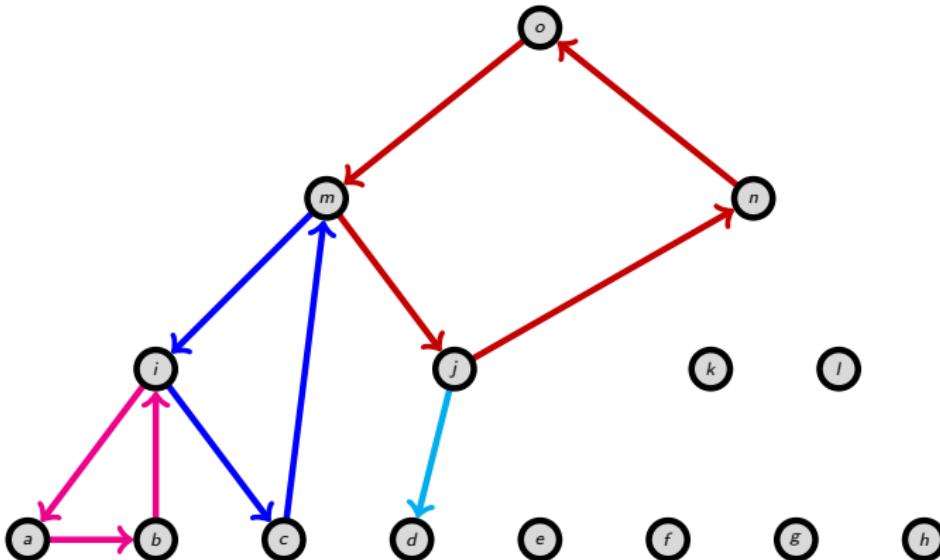
Beispiel (Bestimmen der Eulertour)



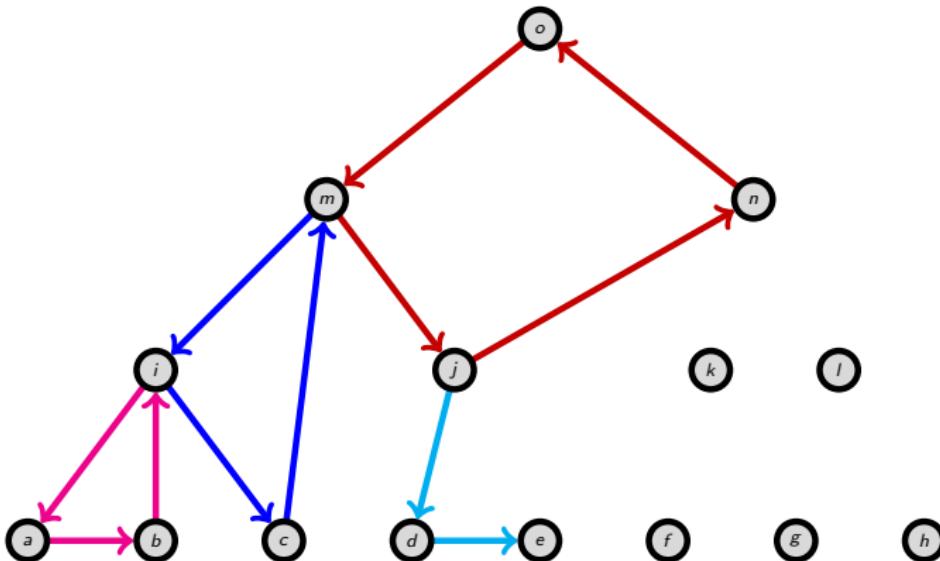
Beispiel (Bestimmen der Eulertour)



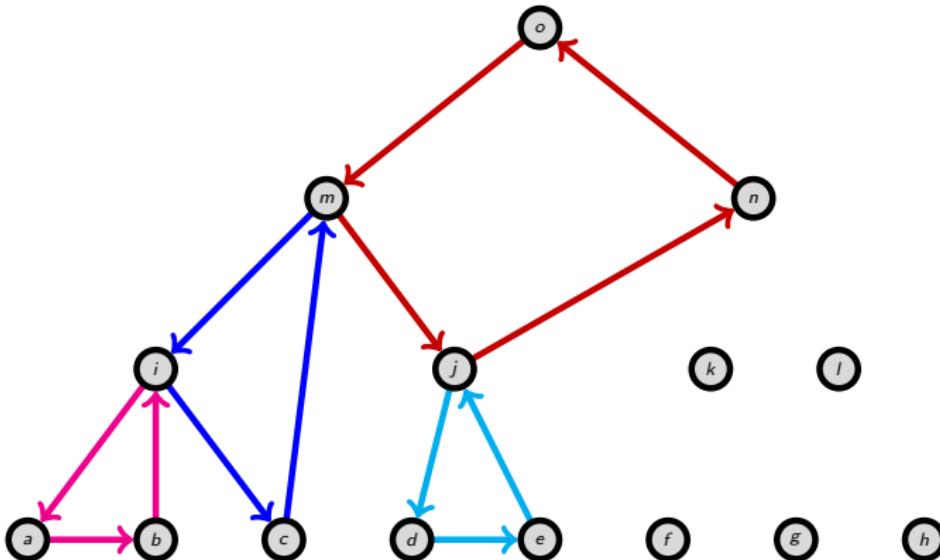
Beispiel (Bestimmen der Eulertour)



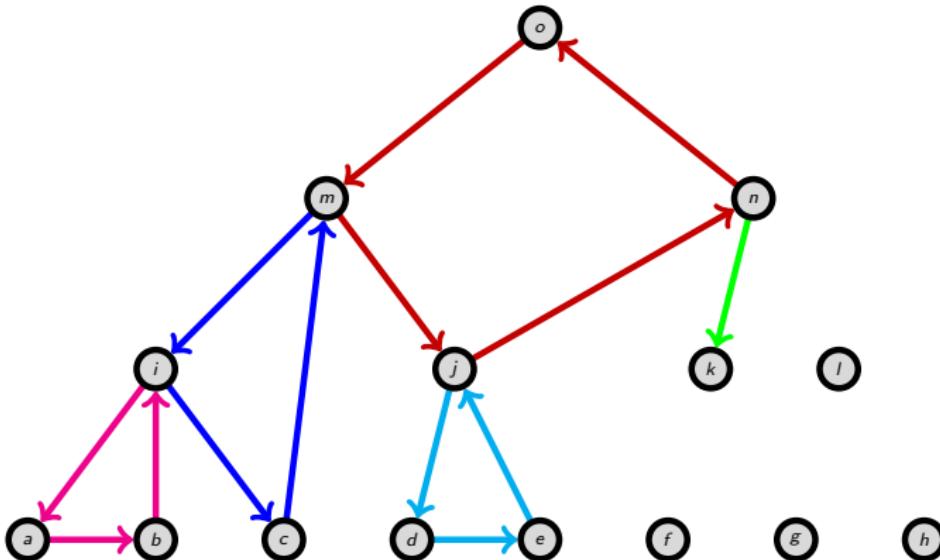
Beispiel (Bestimmen der Eulertour)



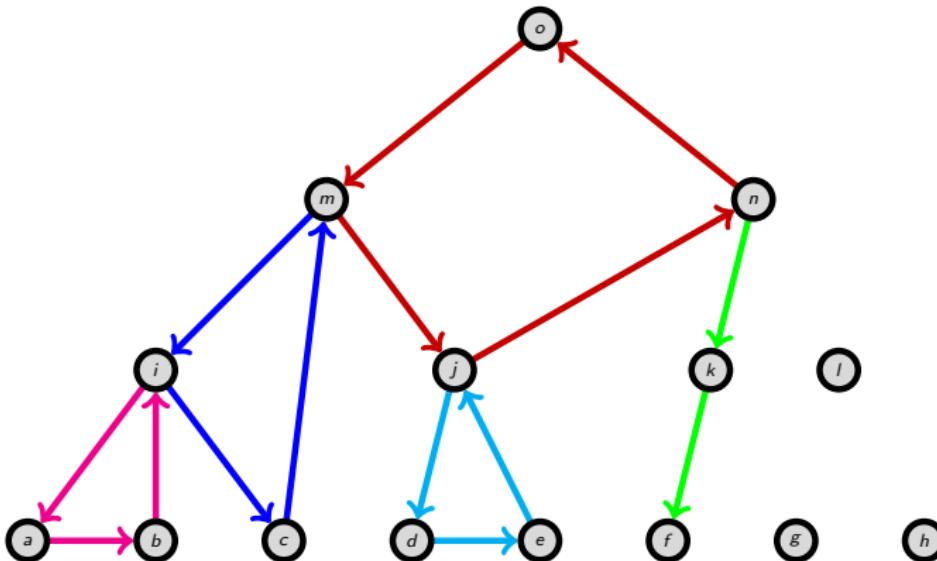
Beispiel (Bestimmen der Eulertour)



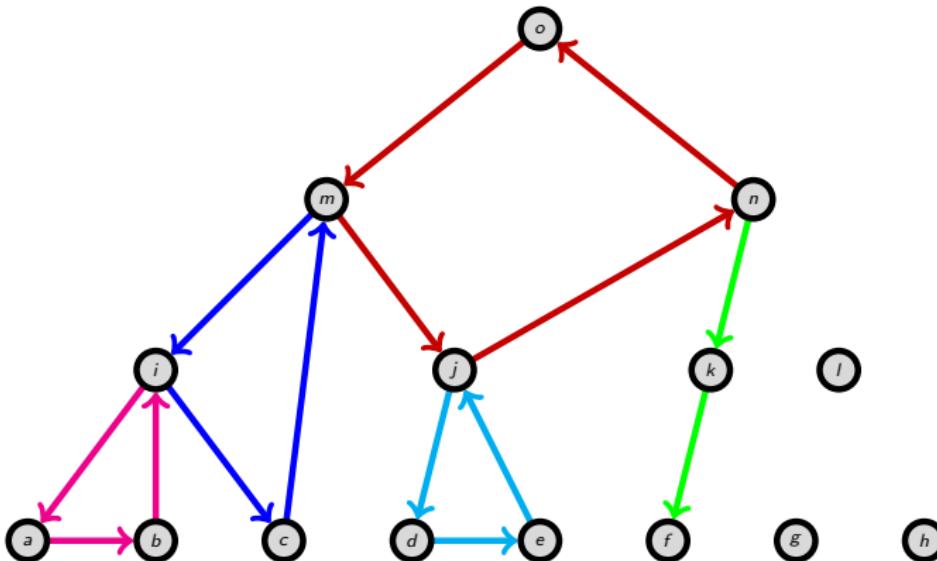
Beispiel (Bestimmen der Eulertour)



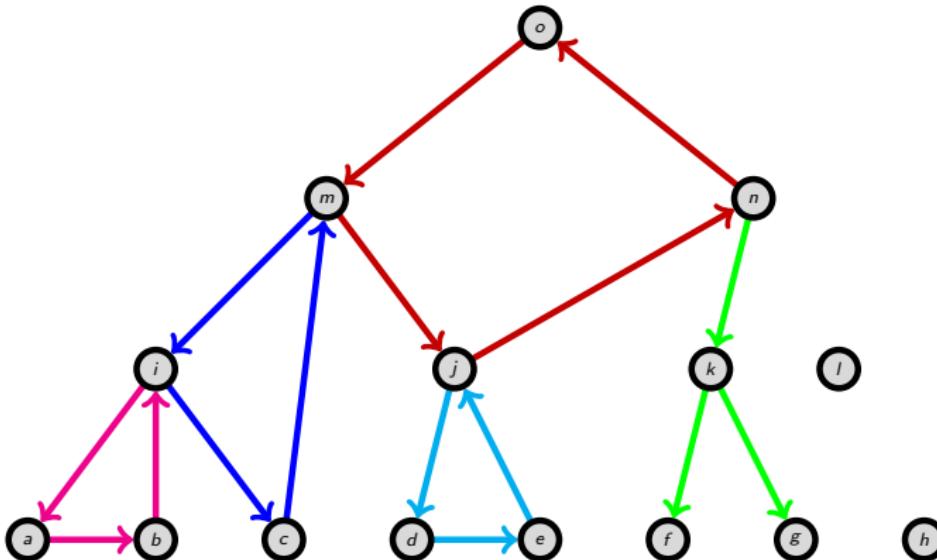
Beispiel (Bestimmen der Eulertour)



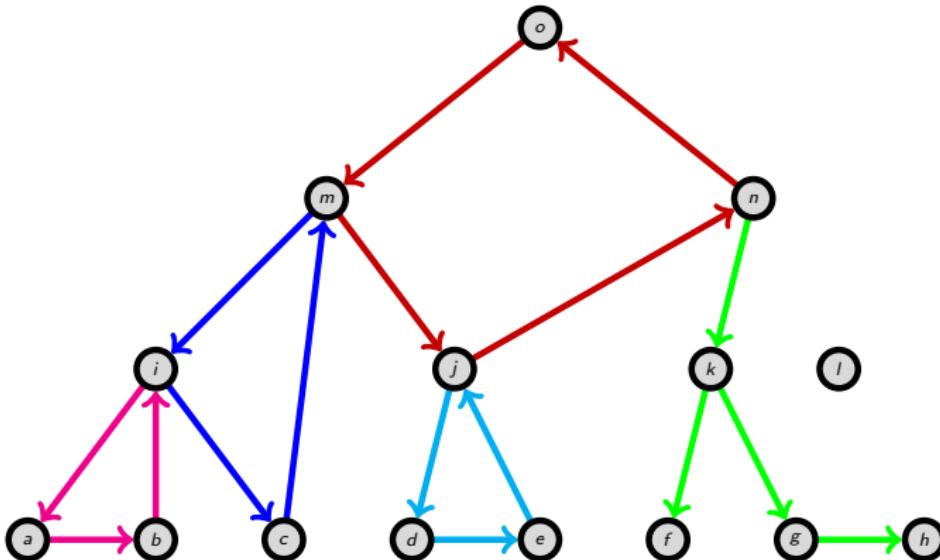
Beispiel (Bestimmen der Eulertour)



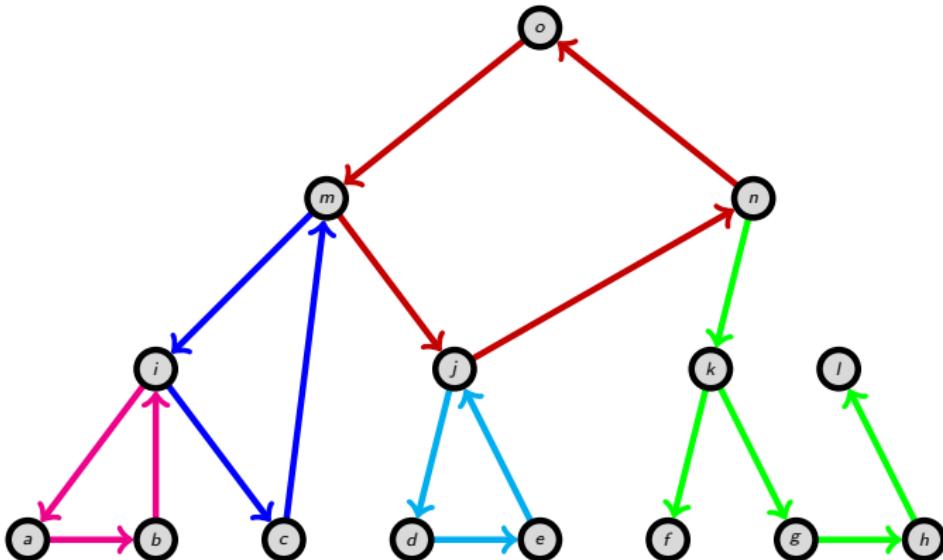
Beispiel (Bestimmen der Eulertour)



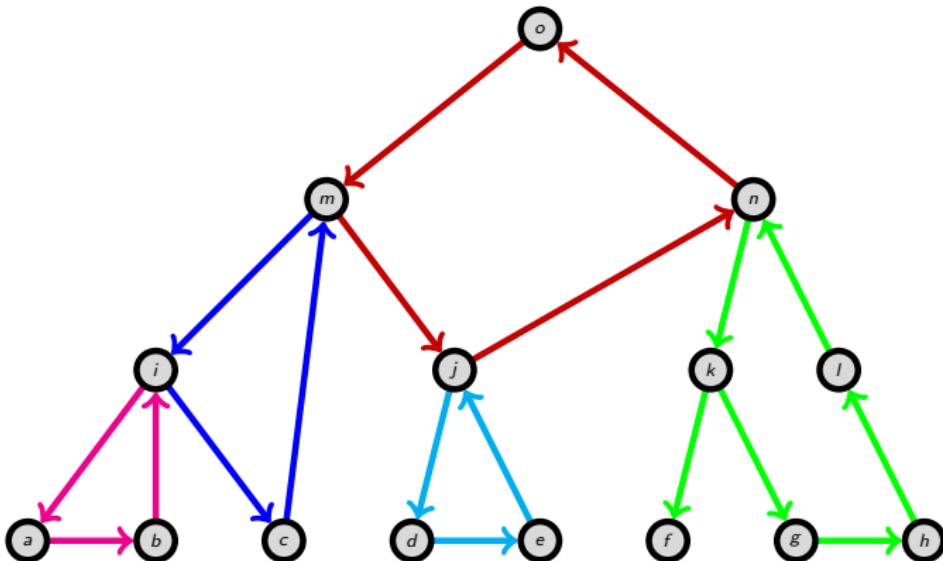
Beispiel (Bestimmen der Eulertour)



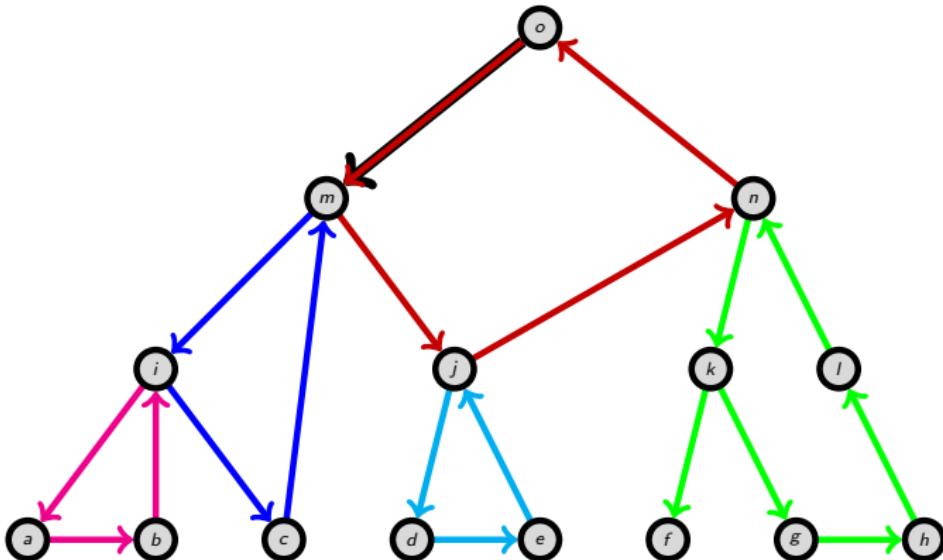
Beispiel (Bestimmen der Eulertour)



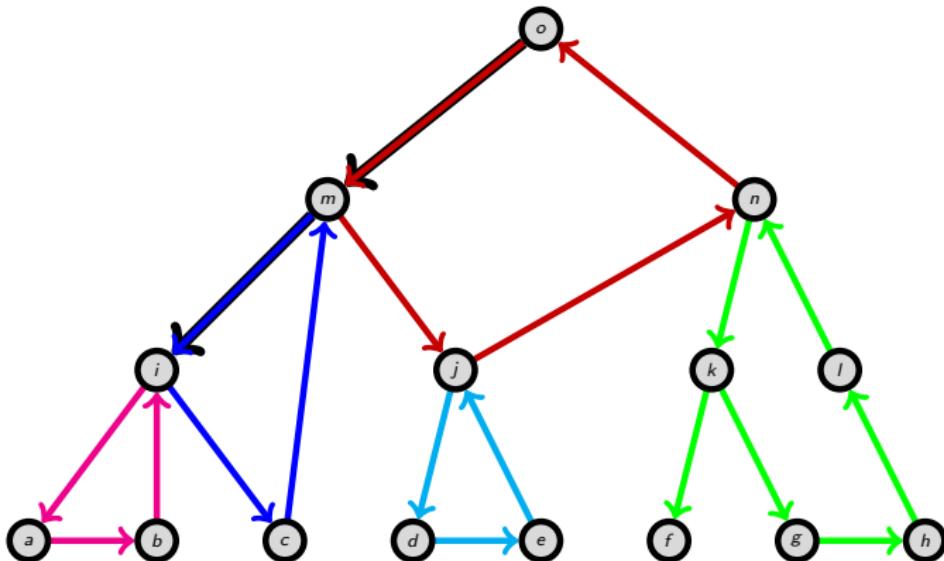
Beispiel (Bestimmen der Eulertour)



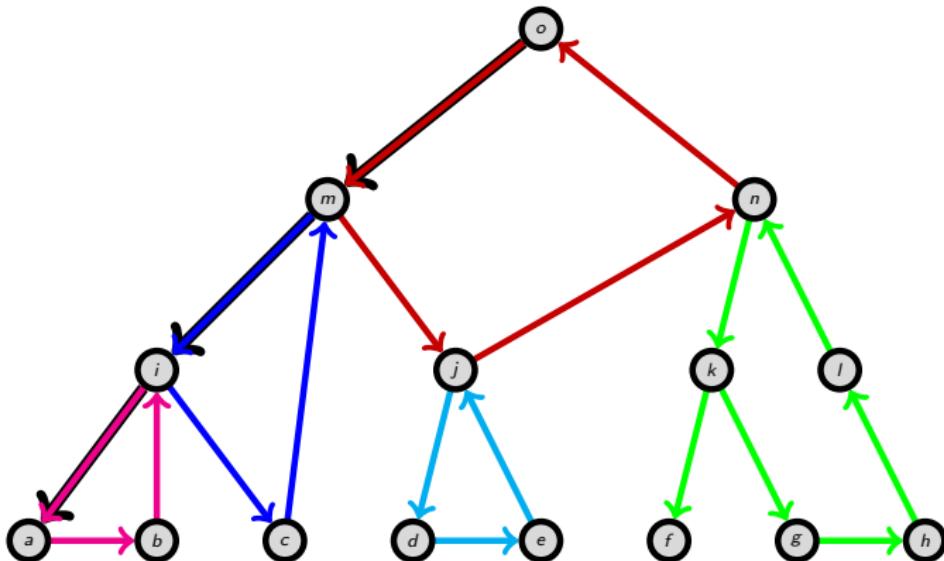
Beispiel (Bestimmen der Eulertour)



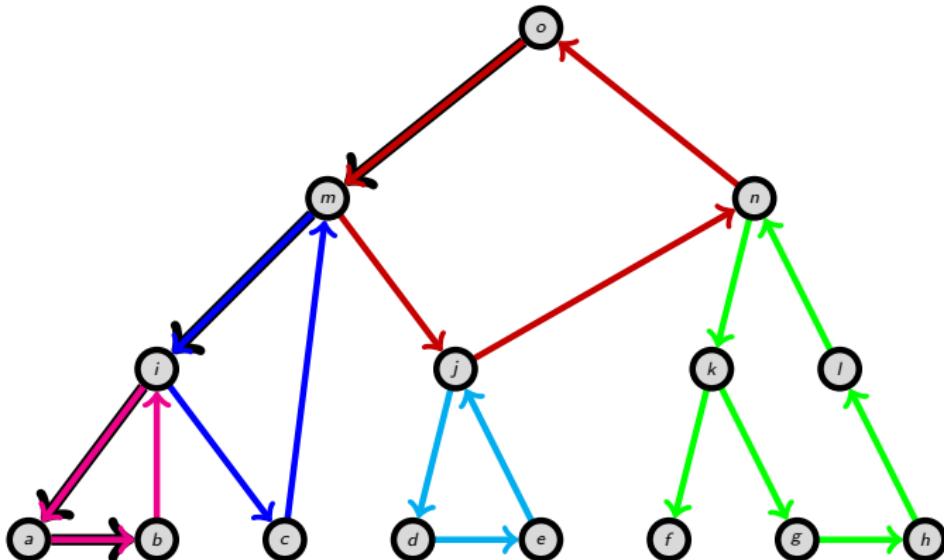
Beispiel (Bestimmen der Eulertour)



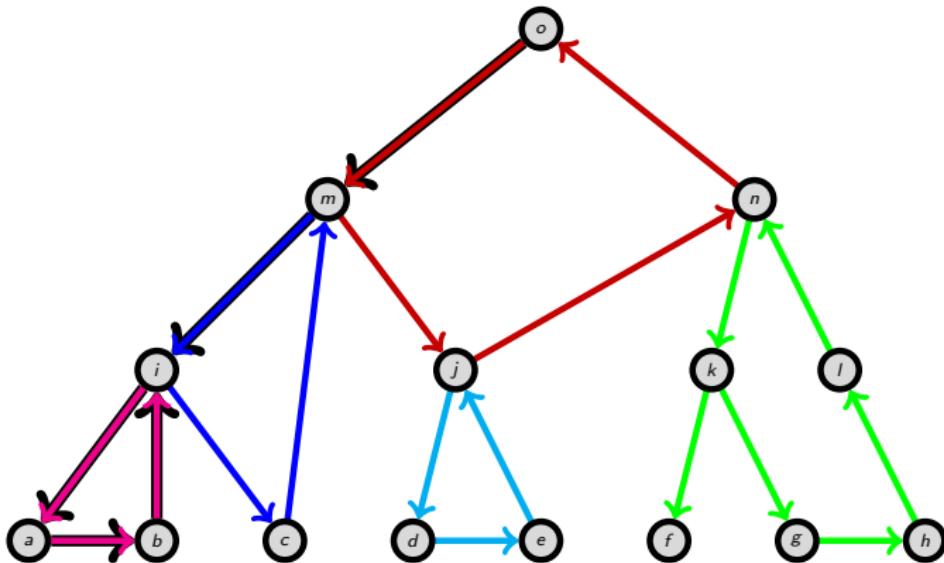
Beispiel (Bestimmen der Eulertour)



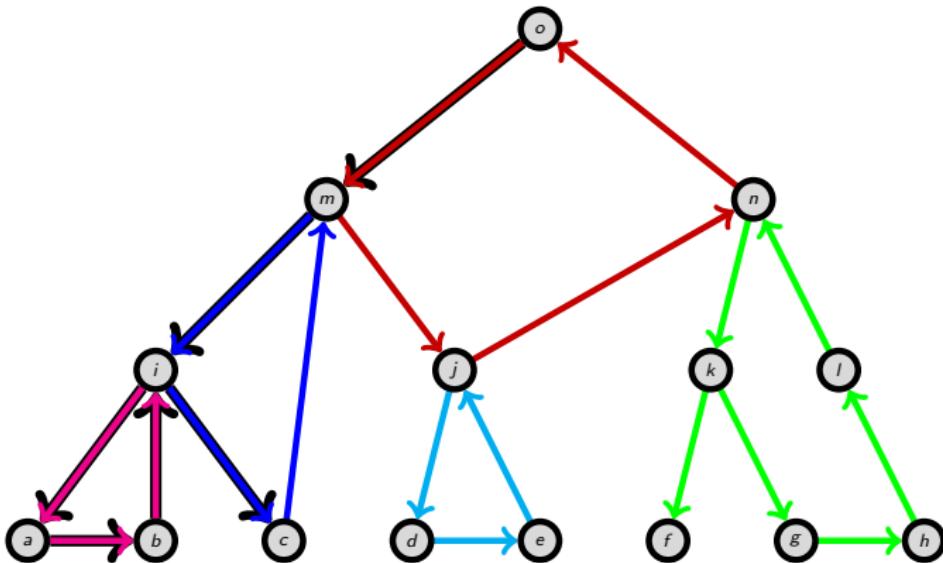
Beispiel (Bestimmen der Eulertour)



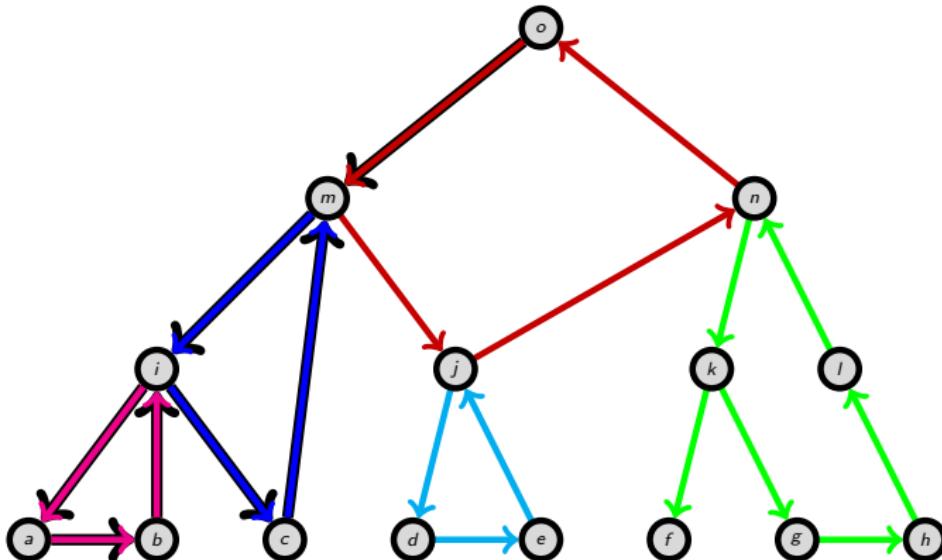
Beispiel (Bestimmen der Eulertour)



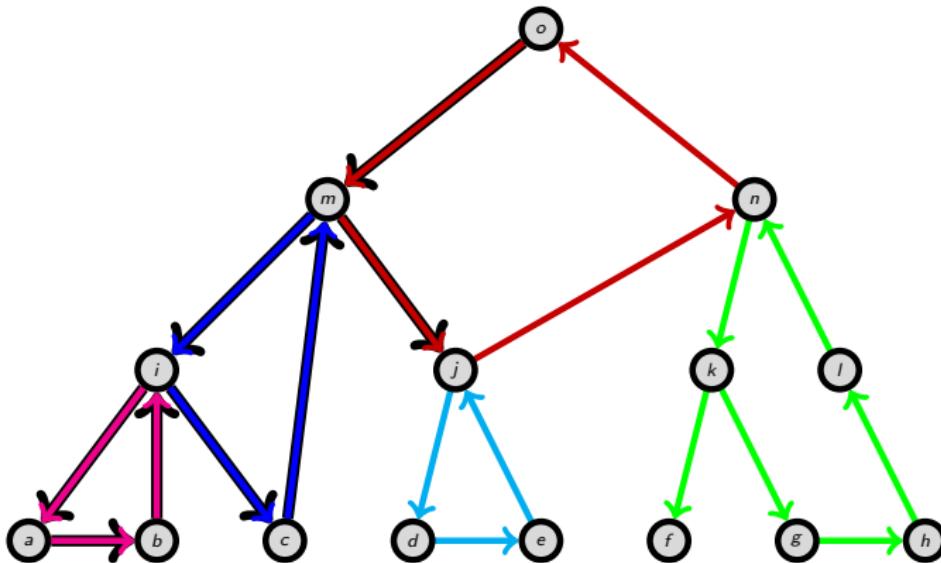
Beispiel (Bestimmen der Eulertour)



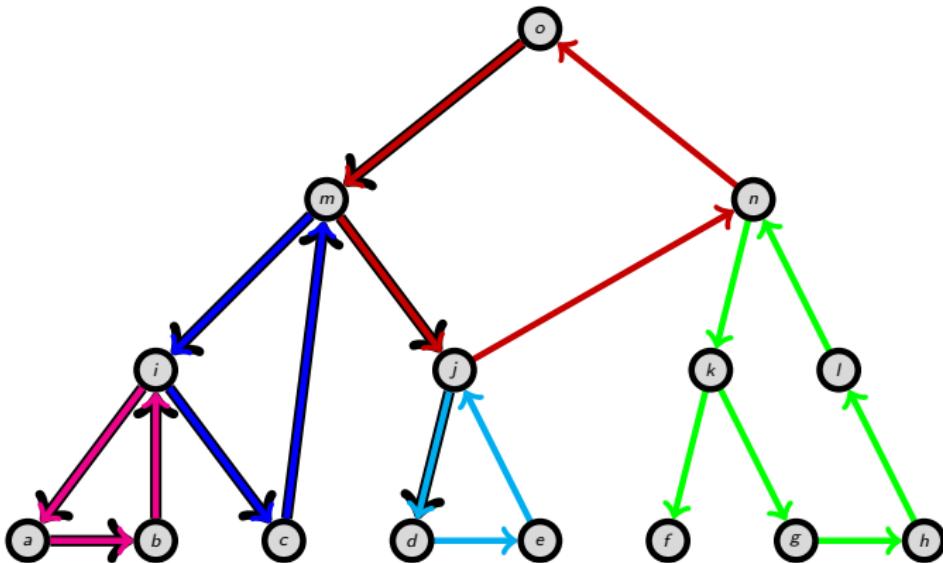
Beispiel (Bestimmen der Eulertour)



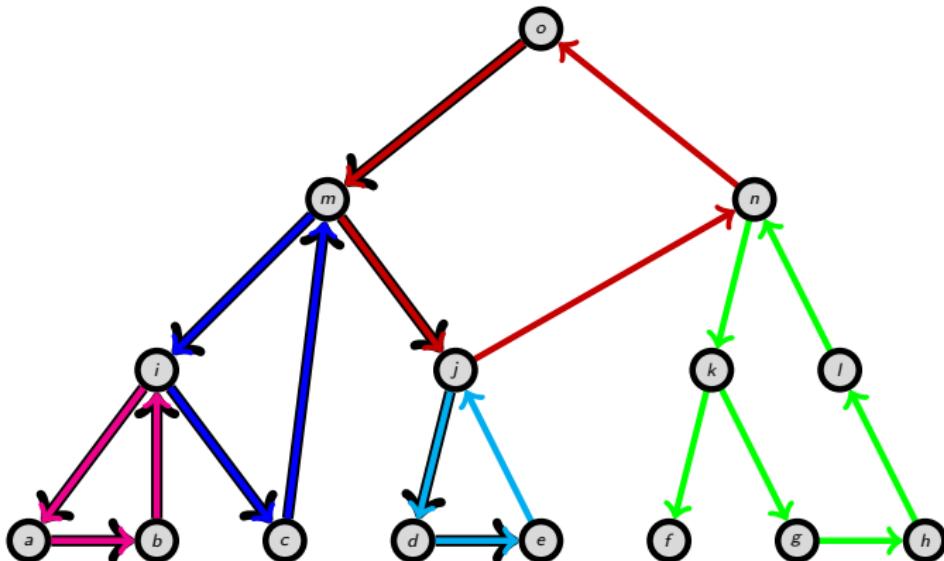
Beispiel (Bestimmen der Eulertour)



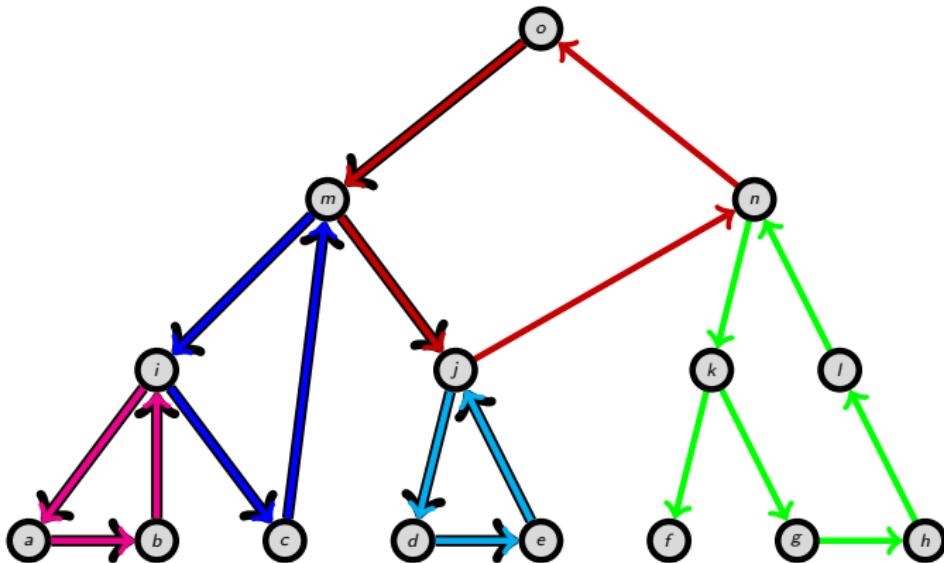
Beispiel (Bestimmen der Eulertour)



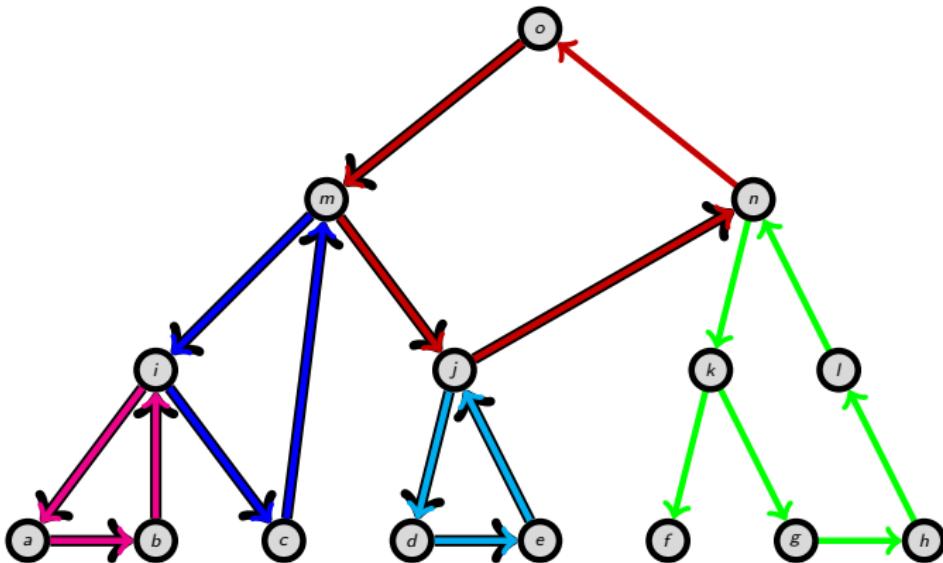
Beispiel (Bestimmen der Eulertour)



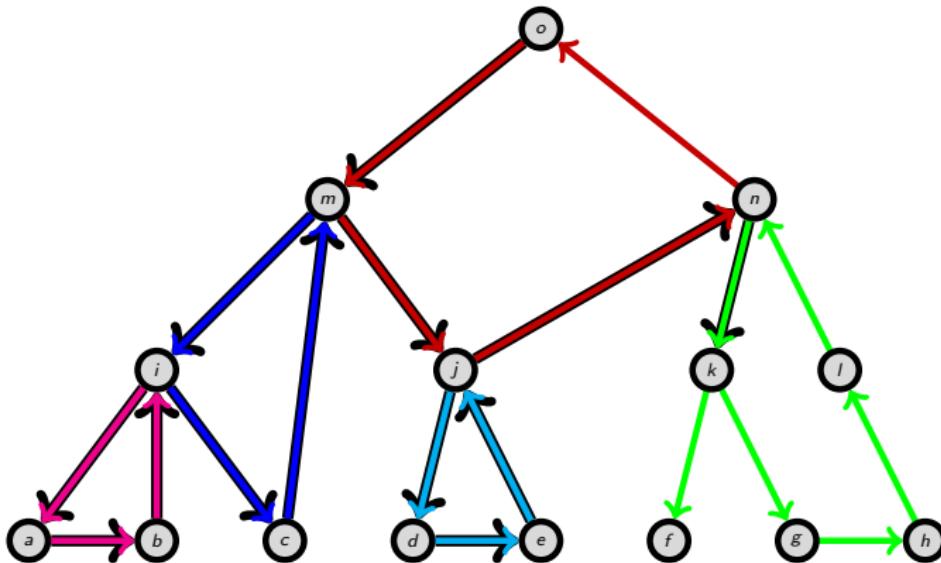
Beispiel (Bestimmen der Eulertour)



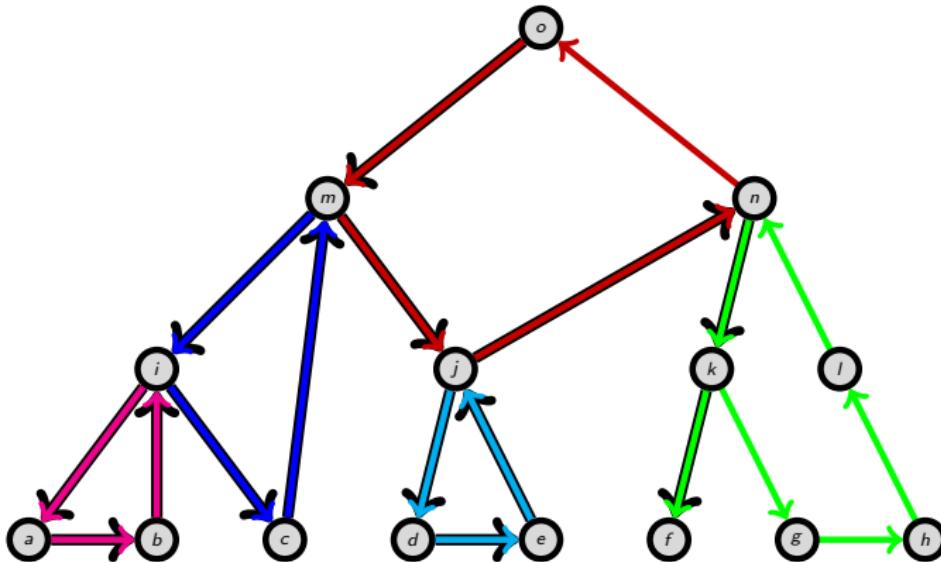
Beispiel (Bestimmen der Eulertour)



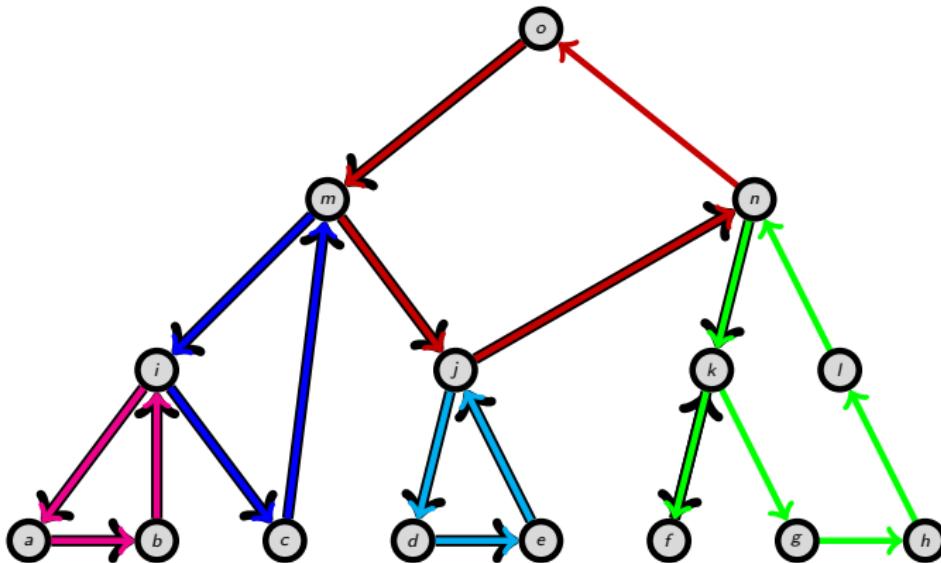
Beispiel (Bestimmen der Eulertour)



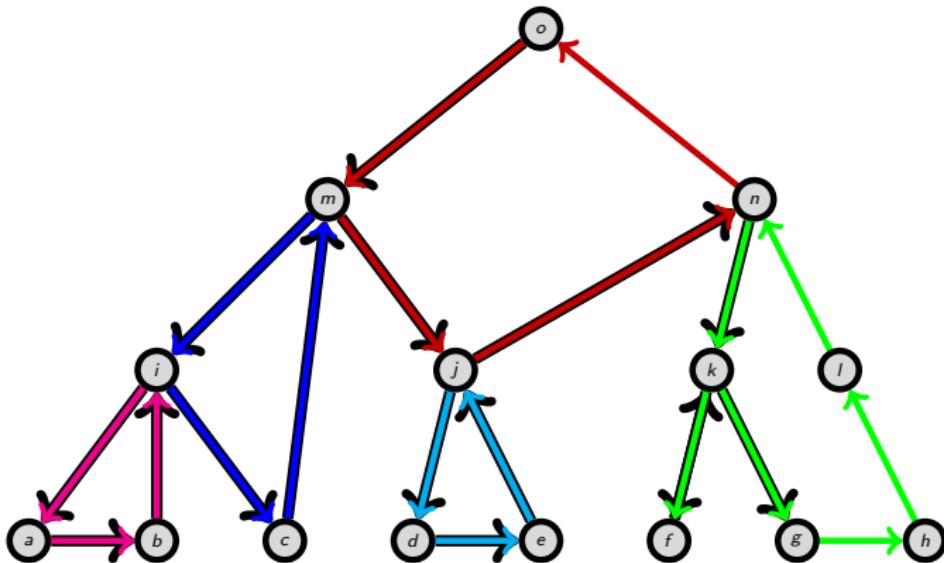
Beispiel (Bestimmen der Eulertour)



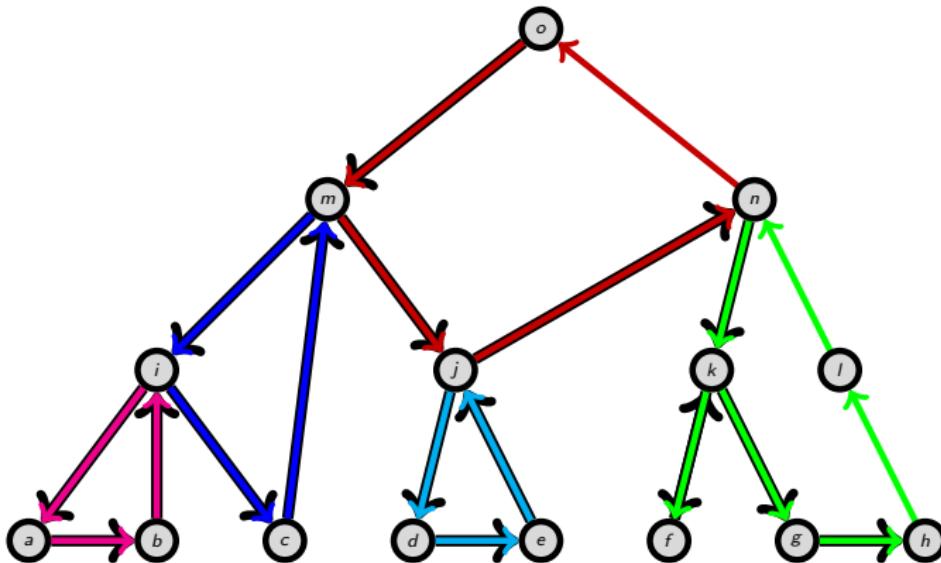
Beispiel (Bestimmen der Eulertour)



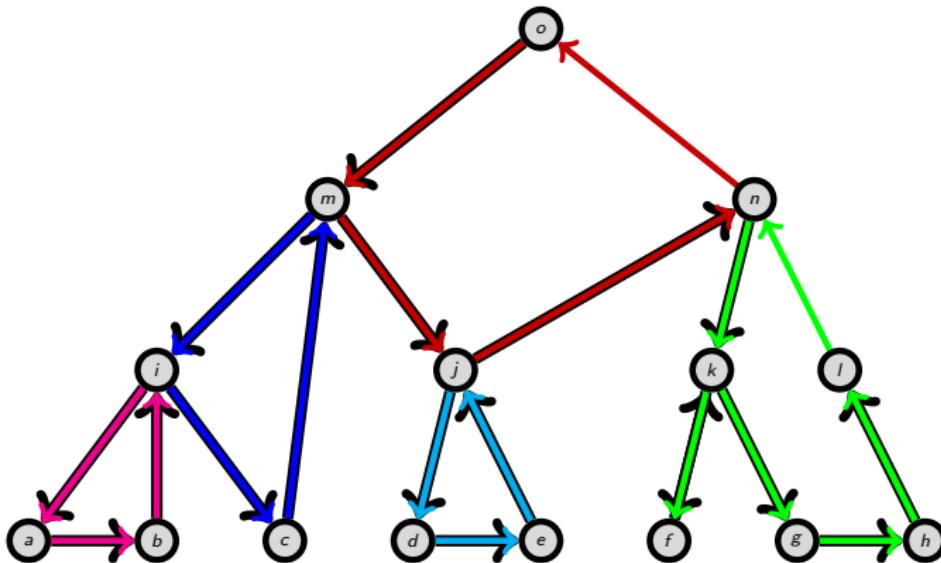
Beispiel (Bestimmen der Eulertour)



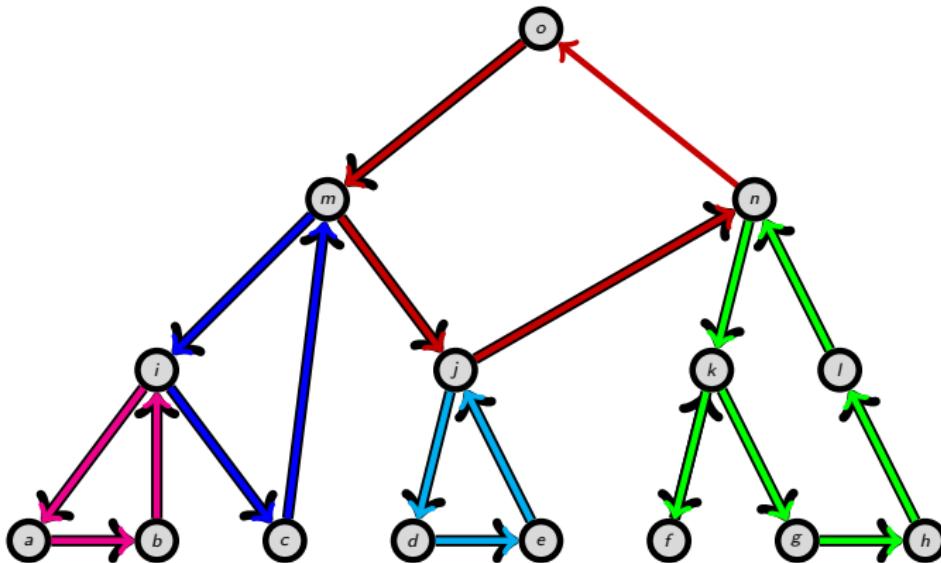
Beispiel (Bestimmen der Eulertour)



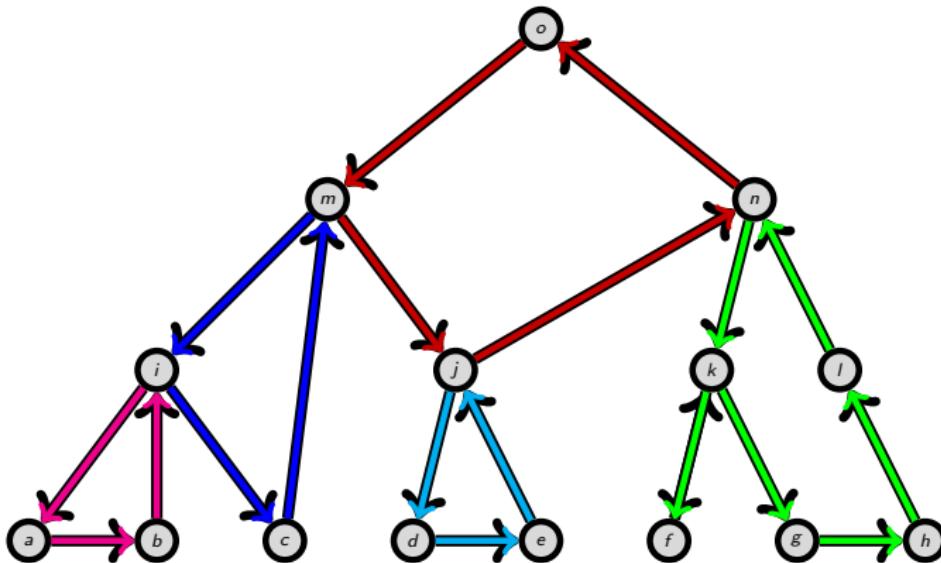
Beispiel (Bestimmen der Eulertour)



Beispiel (Bestimmen der Eulertour)



Beispiel (Bestimmen der Eulertour)



Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
 - ② Setze $i = i + 1$.

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
 - ② Setze $i = i + 1$.
 - ③ Bestimme Kreis C_i , der bei v_i startet.

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
 - ② Setze $i = i + 1$.
 - ③ Bestimme Kreis C_i , der bei v_i startet.
- ⑥ Nun können die Kreise einfach kombiniert werden:

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
 - ② Setze $i = i + 1$.
 - ③ Bestimme Kreis C_i , der bei v_i startet.
- ⑥ Nun können die Kreise einfach kombiniert werden:
 - ① Gehe auf C_0 von v_0 bis v_1 .

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
 - ② Setze $i = i + 1$.
 - ③ Bestimme Kreis C_i , der bei v_i startet.
- ⑥ Nun können die Kreise einfach kombiniert werden:
 - ① Gehe auf C_0 von v_0 bis v_1 .
 - ② Durchlaufe rekursiv alle Kreise C_1, C_2, \dots

Bestimmung des Eulerkreises

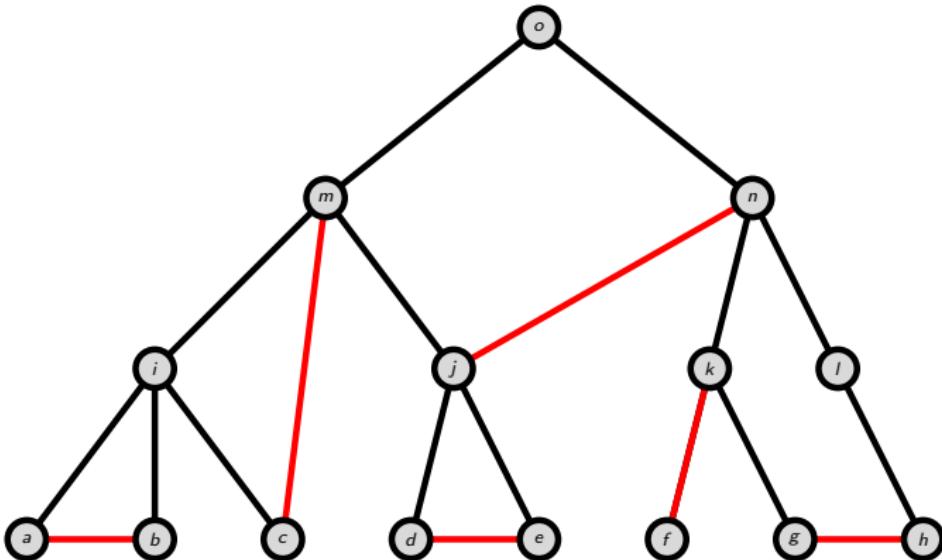
- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
 - ② Setze $i = i + 1$.
 - ③ Bestimme Kreis C_i , der bei v_i startet.
- ⑥ Nun können die Kreise einfach kombiniert werden:
 - ① Gehe auf C_0 von v_0 bis v_1 .
 - ② Durchlaufe rekursiv alle Kreise C_1, C_2, \dots
 - ③ Gehe auf C_0 von v_1 bis v_0 .

Bestimmung des Eulerkreises

- ① Gegeben $G = (V, E)$ Eulergraph.
- ② Setze $i = 0$
- ③ Wähle Startknoten v_0 .
- ④ Bestimme Kreis C_0 , der bei v_0 startet.
- ⑤ Solange es noch ungenutzte Kanten gibt:
 - ① Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
 - ② Setze $i = i + 1$.
 - ③ Bestimme Kreis C_i , der bei v_i startet.
- ⑥ Nun können die Kreise einfach kombiniert werden:
 - ① Gehe auf C_0 von v_0 bis v_1 .
 - ② Durchlaufe rekursiv alle Kreise C_1, C_2, \dots
 - ③ Gehe auf C_0 von v_1 bis v_0 .

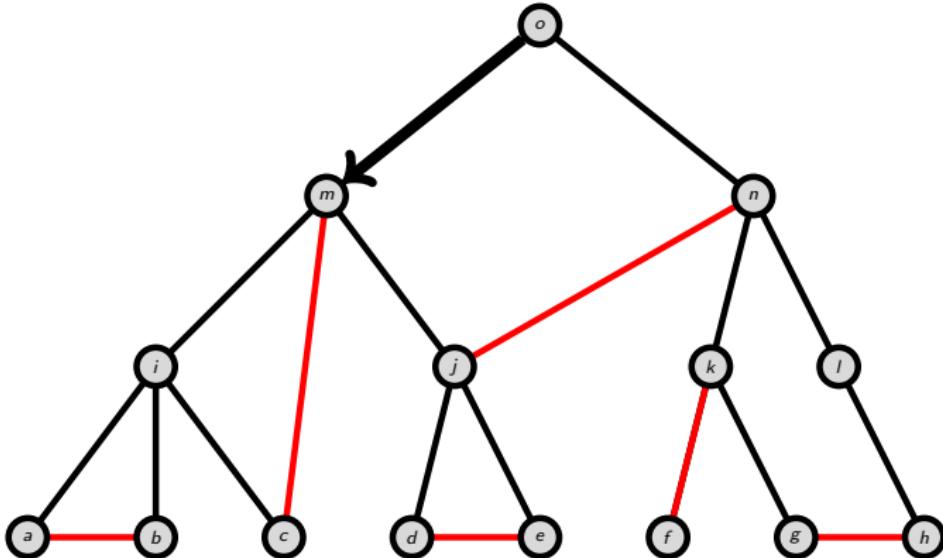
Laufzeit: $O(|E|)$.

Beispiel (1.5 Approximation)



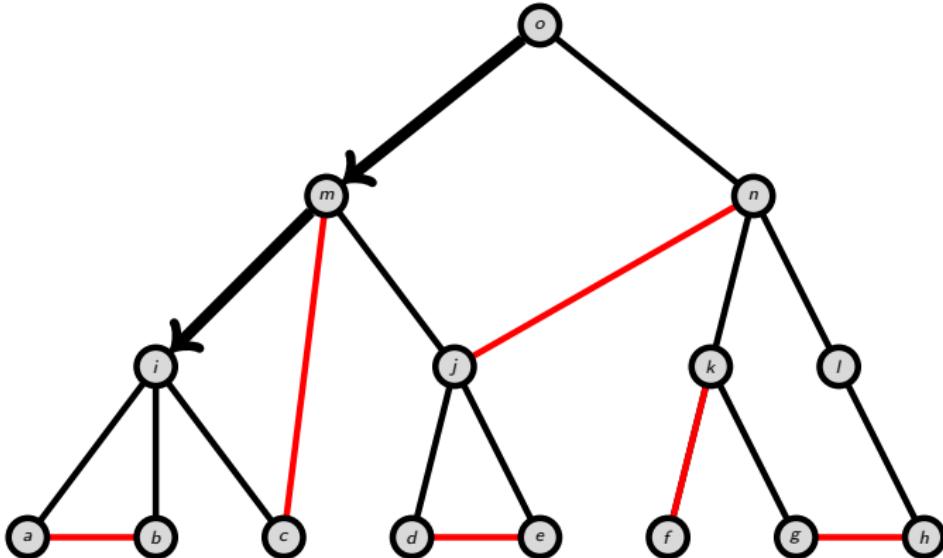
Beispiel (1.5 Approximation)

Eulertour: $o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o$



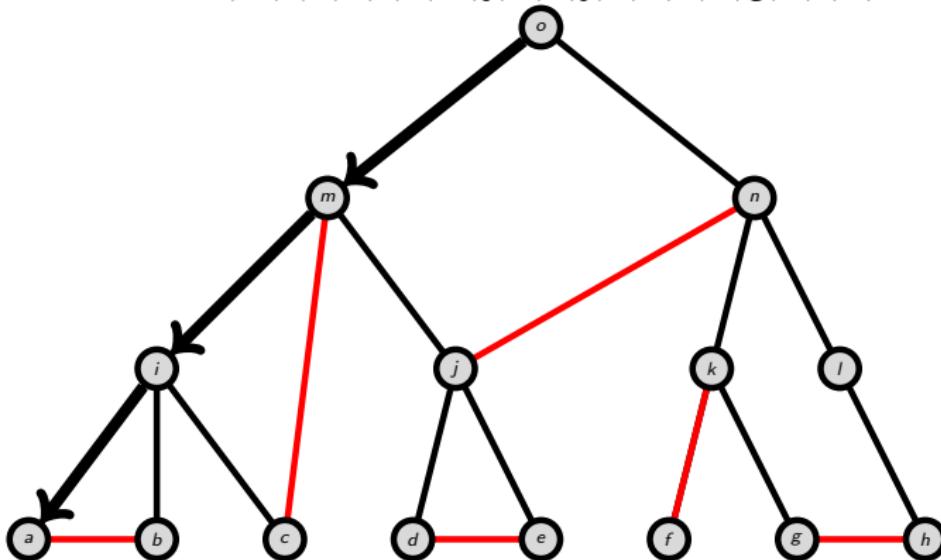
Beispiel (1.5 Approximation)

Eulertour: $o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o$



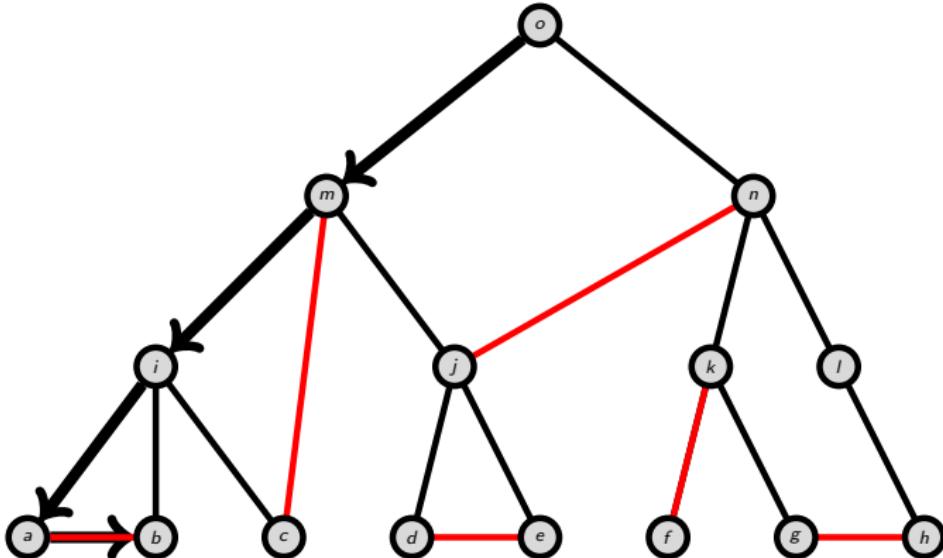
Beispiel (1.5 Approximation)

Eulertour: $o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o$



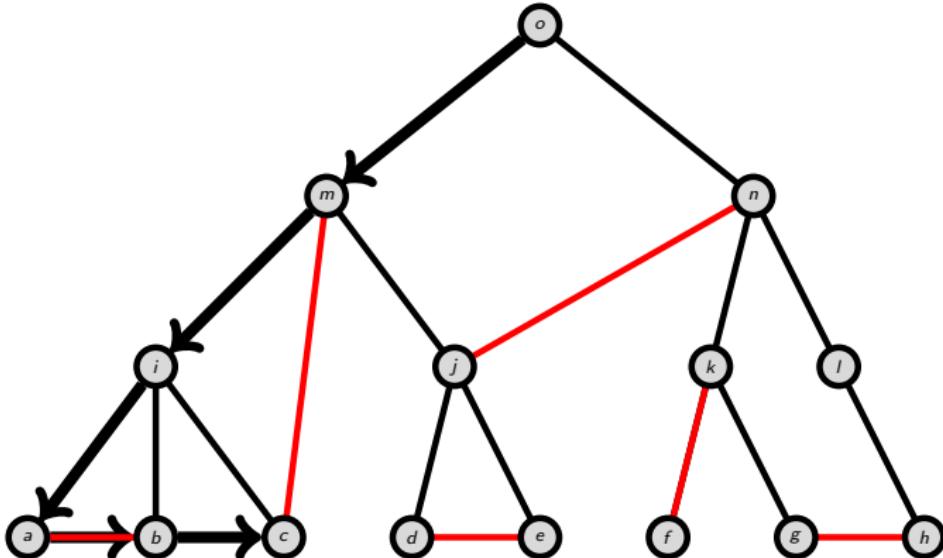
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



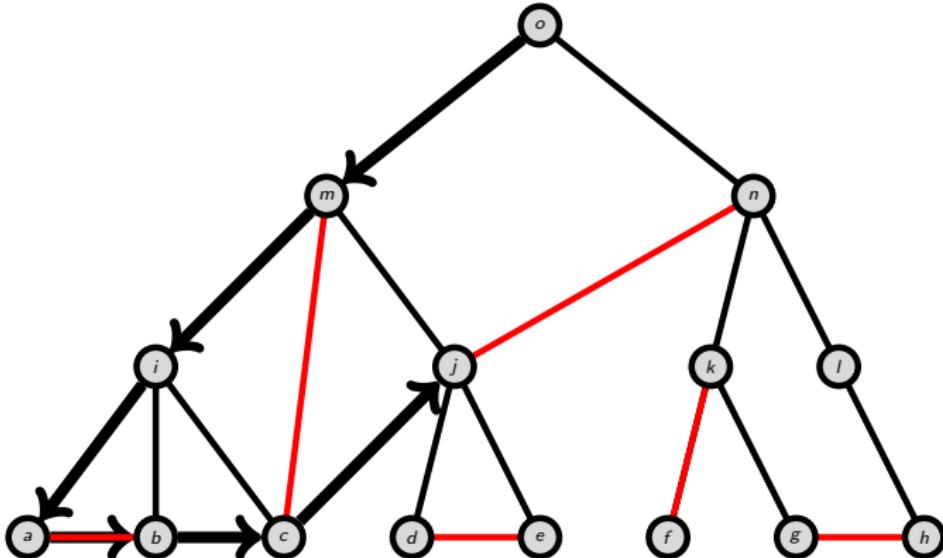
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



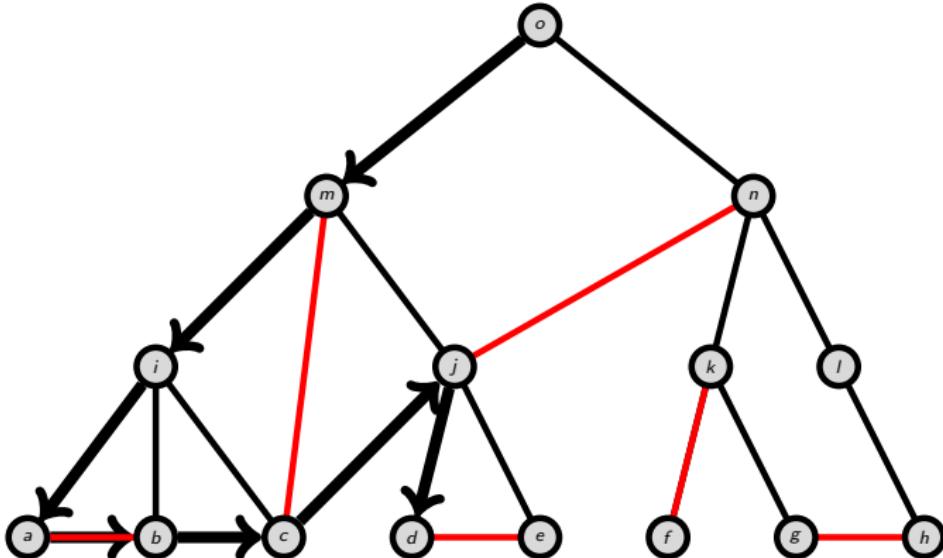
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



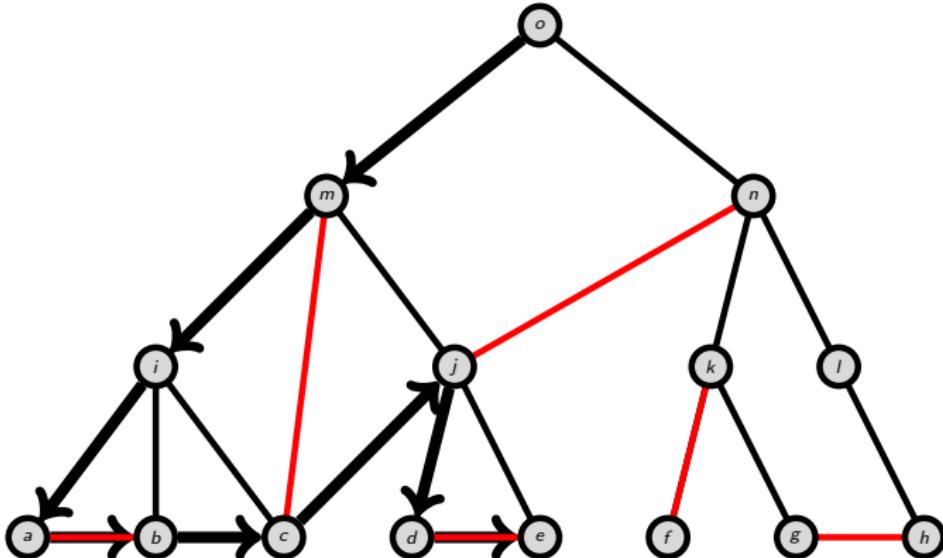
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



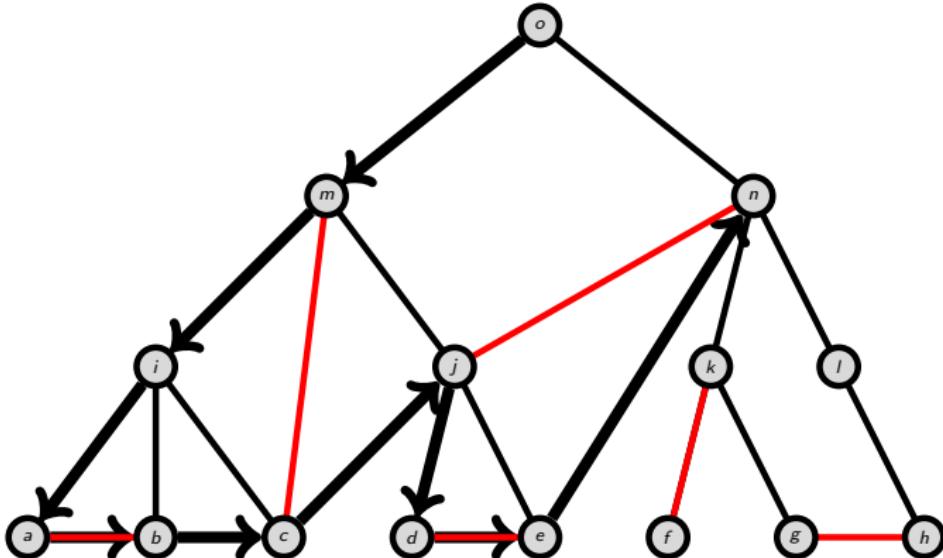
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



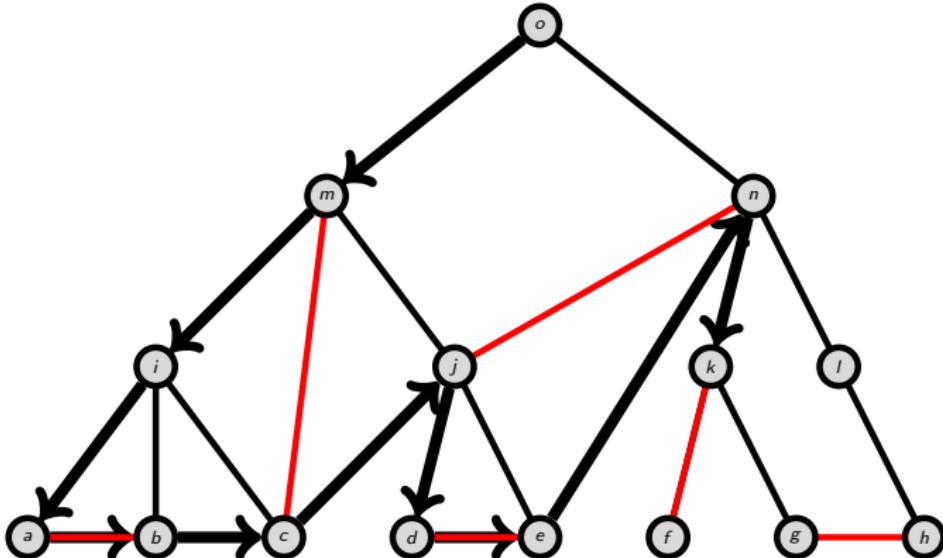
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



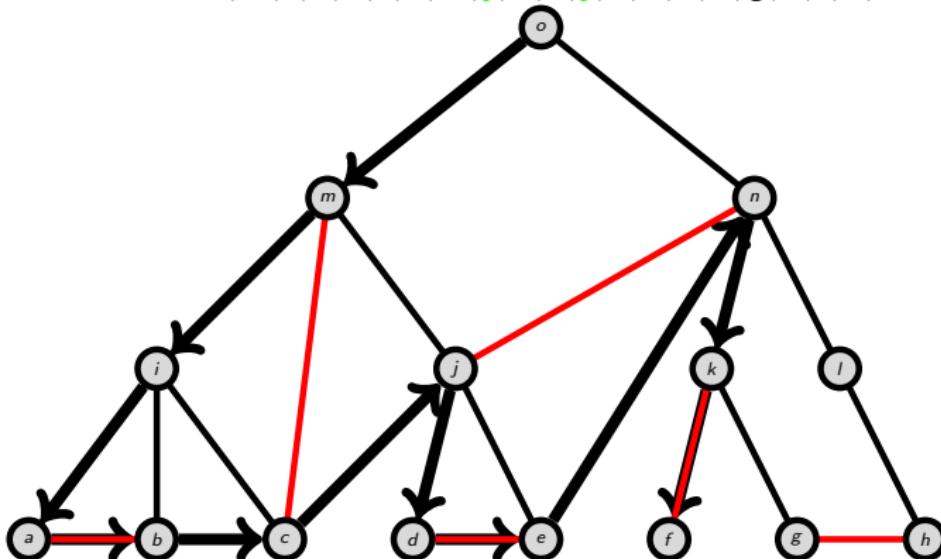
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



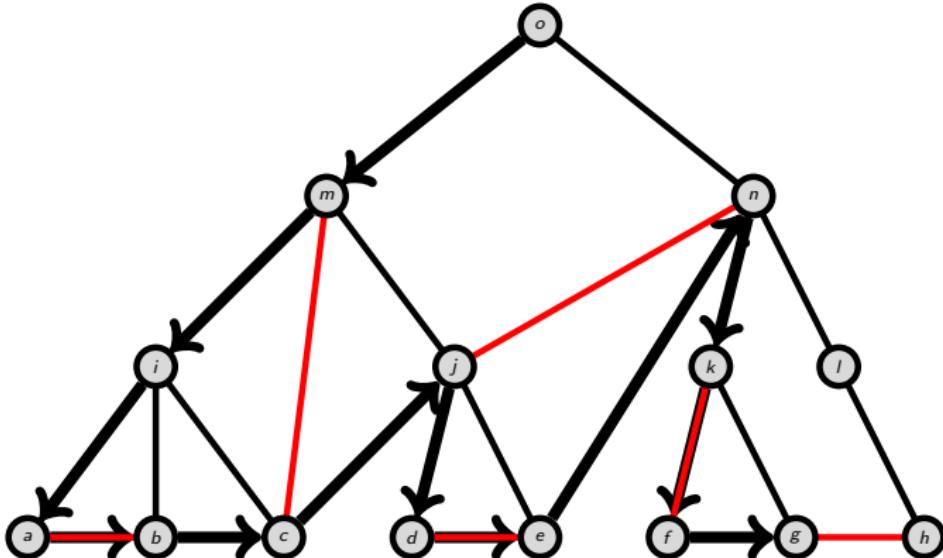
Beispiel (1.5 Approximation)

Eulertour: $o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o$



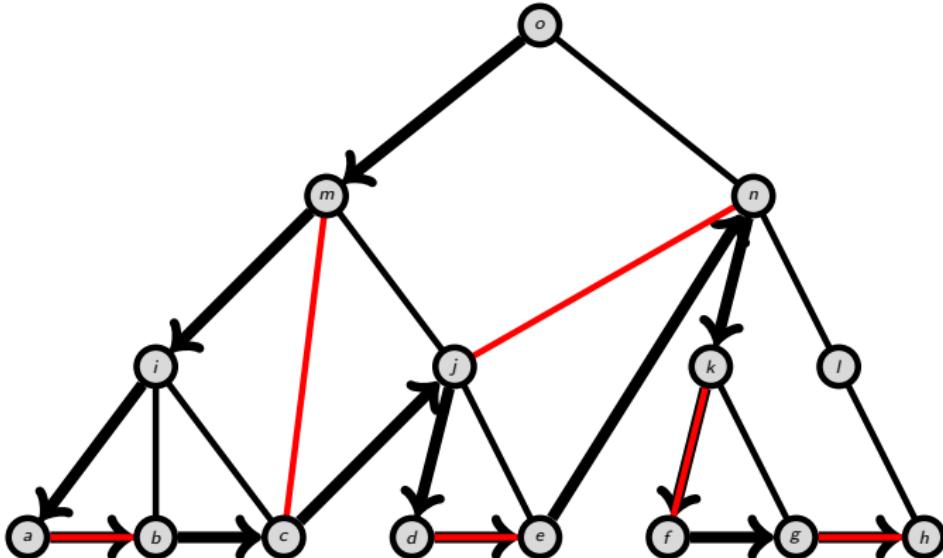
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



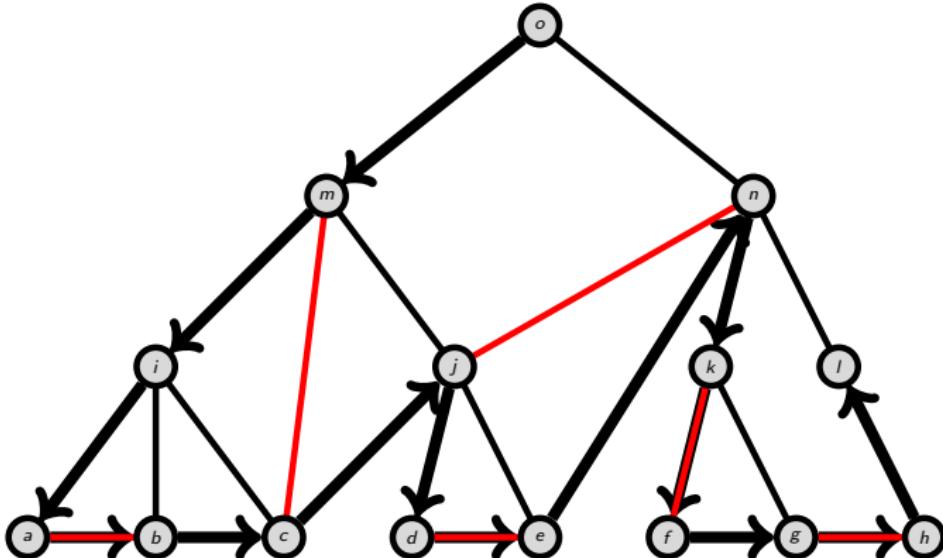
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



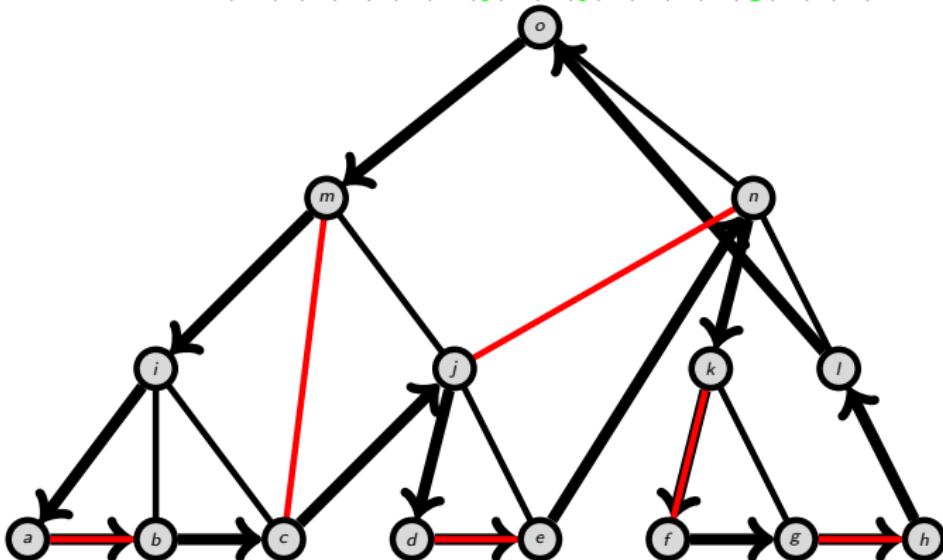
Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



Beispiel (1.5 Approximation)

Eulertour: *o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o*



1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.

1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.

1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.

1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum

1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leq c(C^*)/2$.

1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leq c(C^*)/2$.
 - Siehe dazu nächste Folie.

1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leq c(C^*)/2$.
 - Siehe dazu nächste Folie.
- $\frac{c(C)}{c(C^*)}$

1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leq c(C^*)/2$.
 - Siehe dazu nächste Folie.
- $\frac{c(C)}{c(C^*)} \leq \frac{c(T) + c(M)}{c(C^*)}$

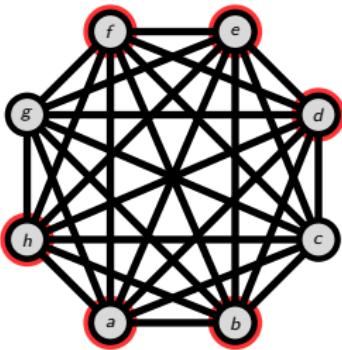
1.5-Approximation (Beweis)

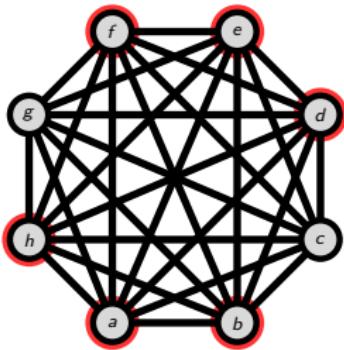
- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leq c(C^*)/2$.
 - Siehe dazu nächste Folie.
- $\frac{c(C)}{c(C^*)} \leq \frac{c(T) + c(M)}{c(C^*)} \leq \frac{c(C^*) + c(C^*)/2}{c(C^*)}$

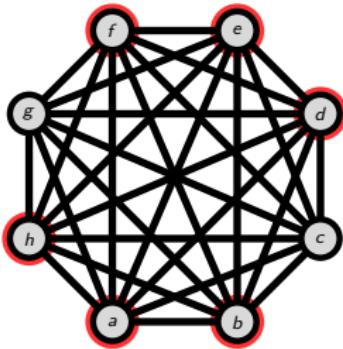
1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G ,
 M das Matching,
 C' Eulerkreis in T' und
 C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leq c(C^* - e) \leq c(C^*)$.
 - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leq c(C^*)/2$.
 - Siehe dazu nächste Folie.
- $\frac{c(C)}{c(C^*)} \leq \frac{c(T)+c(M)}{c(C^*)} \leq \frac{c(C^*)+c(C^*)/2}{c(C^*)} \leq \frac{1.5 \cdot c(C^*)}{c(C^*)} = 1.5$

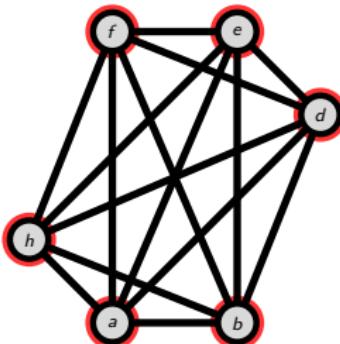
1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)

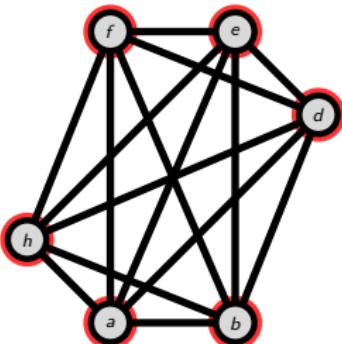
1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)

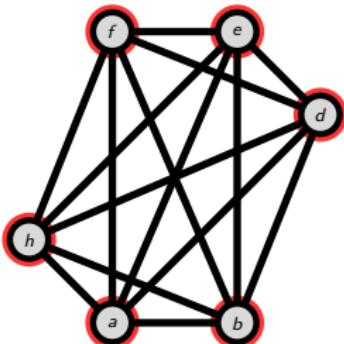
- Sei C^* eine optimale Lösung.

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)

- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T .

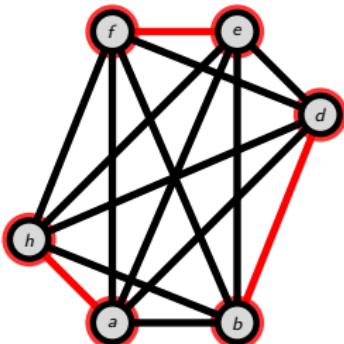
1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)

- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T .
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)

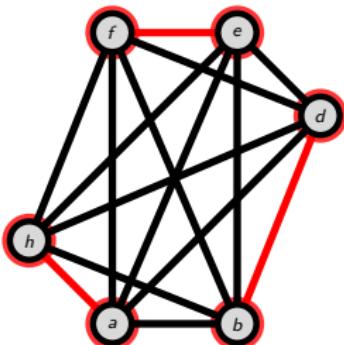
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T .
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)



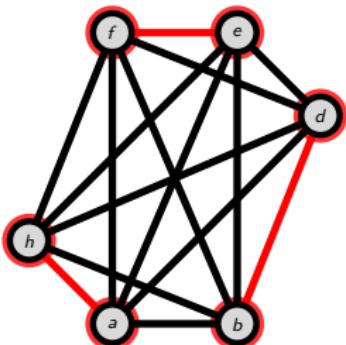
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T .
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)



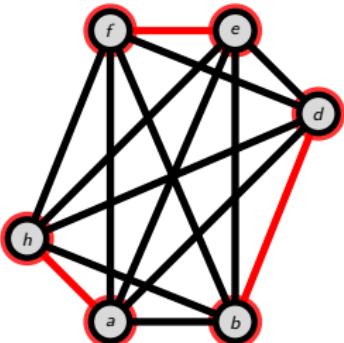
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T .
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2
- $c(M) \leq c(M_1)$ und $c(M) \leq c(M_2)$.

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)



- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T .
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2 .
- $c(M) \leq c(M_1)$ und $c(M) \leq c(M_2)$.
- $2 \cdot c(M) \leq c(M_1) + c(M_2) = c(C'') \leq c(C^*)$

1.5-Approximation (Zeige $c(M) \leq c(C^*)/2$)



- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T .
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2 .
- $c(M) \leq c(M_1)$ und $c(M) \leq c(M_2)$.
- $2 \cdot c(M) \leq c(M_1) + c(M_2) = c(C'') \leq c(C^*)$
- $c(M) \leq c(C^*)/2$

Steinerbaum

Definition (Steinerbaum Problem)

Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.

Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

Steinerbaum

Definition (Steinerbaum Problem)

Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.

Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

- Die Blätter von S sind alle aus T .

Steinerbaum

Definition (Steinerbaum Problem)

Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.

Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

- Die Blätter von S sind alle aus T .
- Die Menge P heißt Steinerpunkte.

Steinerbaum

Definition (Steinerbaum Problem)

Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.

Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

- Die Blätter von S sind alle aus T .
- Die Menge P heißt Steinerpunkte.
- Falls $T = V$ gilt, ist ein minimaler Spannbaum zu finden.

Steinerbaum

Definition (Steinerbaum Problem)

Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.

Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

- Die Blätter von S sind alle aus T .
- Die Menge P heißt Steinerpunkte.
- Falls $T = V$ gilt, ist ein minimaler Spannbaum zu finden.
- Falls $|T| = 2$ gilt, ist ein minimaler Weg zu finden.

Steinerbaum

Definition (Steinerbaum Problem)

Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.

Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

- Die Blätter von S sind alle aus T .
- Die Menge P heißt Steinerpunkte.
- Falls $T = V$ gilt, ist ein minimaler Spannbaum zu finden.
- Falls $|T| = 2$ gilt, ist ein minimaler Weg zu finden.
- Anwendung: VLSI-Chip-Design

Steinerbaum

Definition (Steinerbaum Problem)

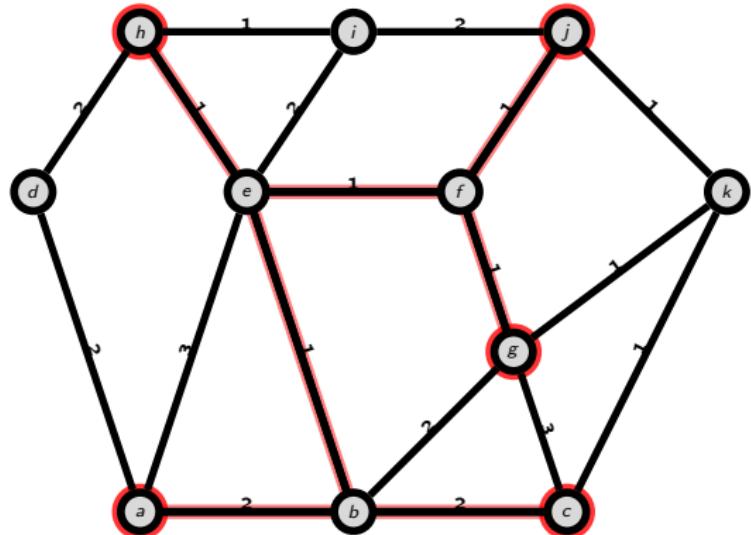
Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.

Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

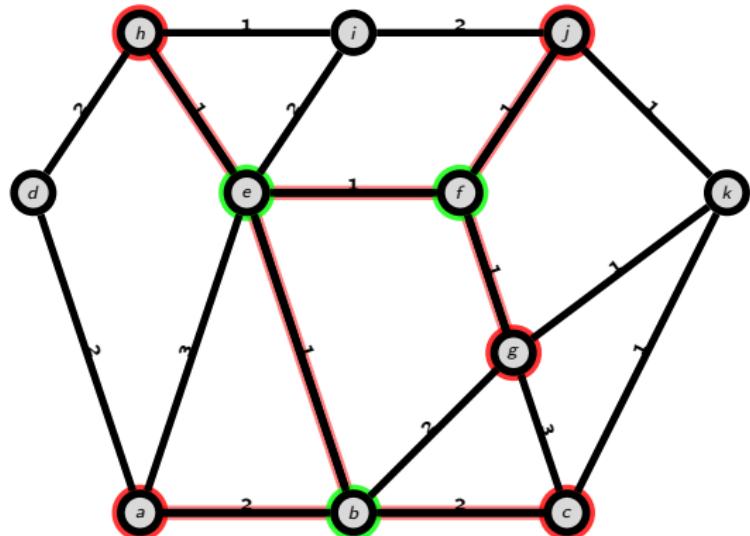
- Die Blätter von S sind alle aus T .
- Die Menge P heißt Steinerpunkte.
- Falls $T = V$ gilt, ist ein minimaler Spannbaum zu finden.
- Falls $|T| = 2$ gilt, ist ein minimaler Weg zu finden.
- Anwendung: VLSI-Chip-Design
- Entscheidungsvariante: Gibt es Menge $P \subset V \setminus T$ der Größe k , so daß $G[S \cup P]$ zusammenhängend ist.

Beispiel

Beispiel

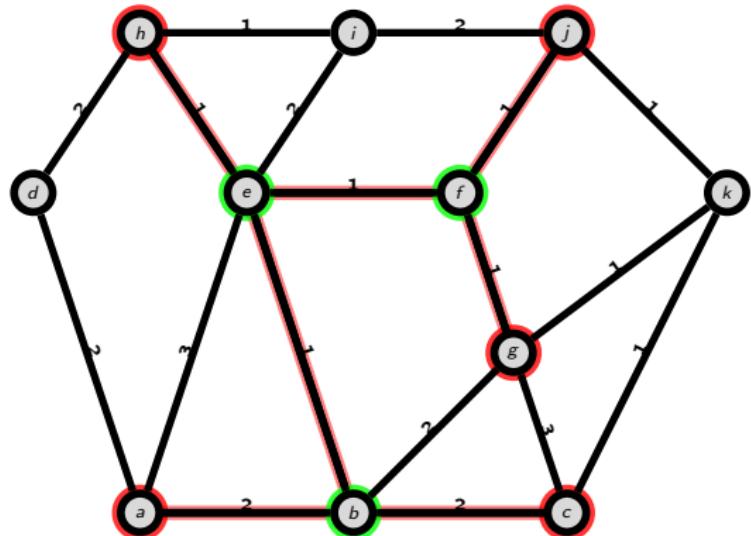


Beispiel



- Die Terminals T sind rot.
- Die Steinerpunkte sind grün.

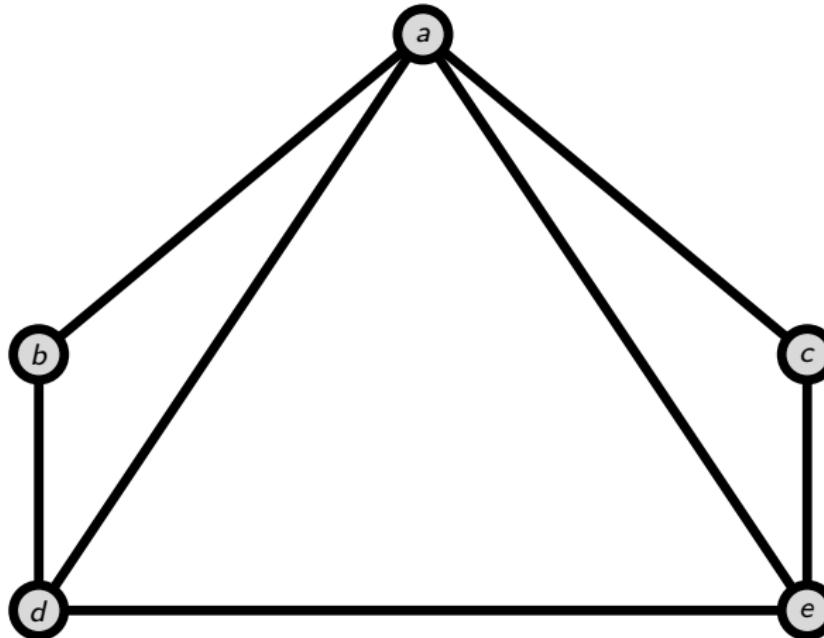
Beispiel



- Die Terminals T sind rot.
- Die Steinerpunkte sind grün.
- Die Kosten sind: 9.

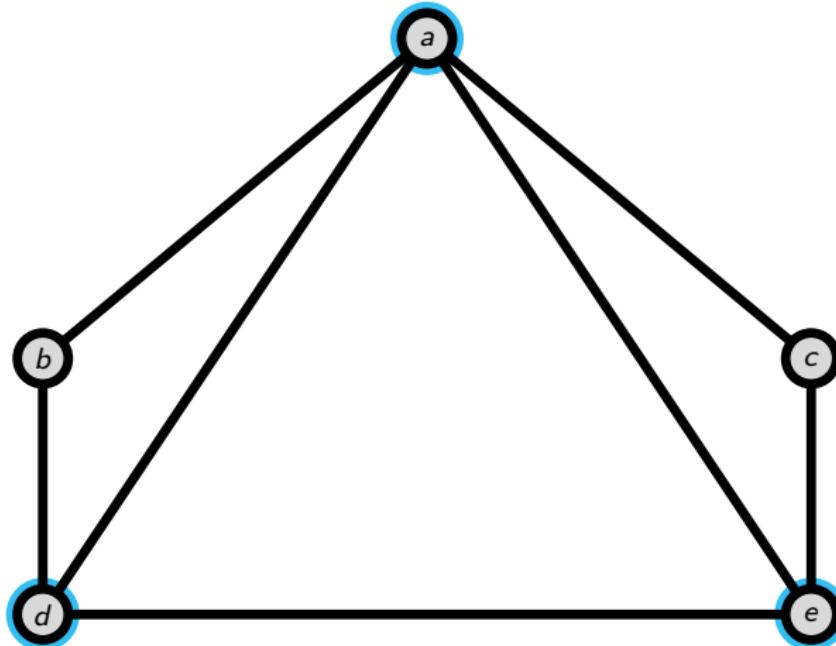
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



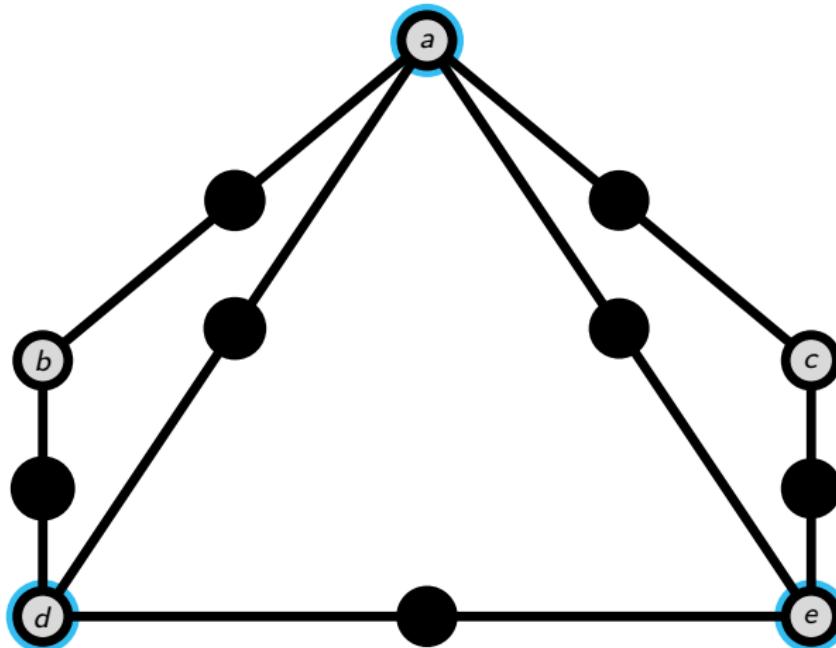
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



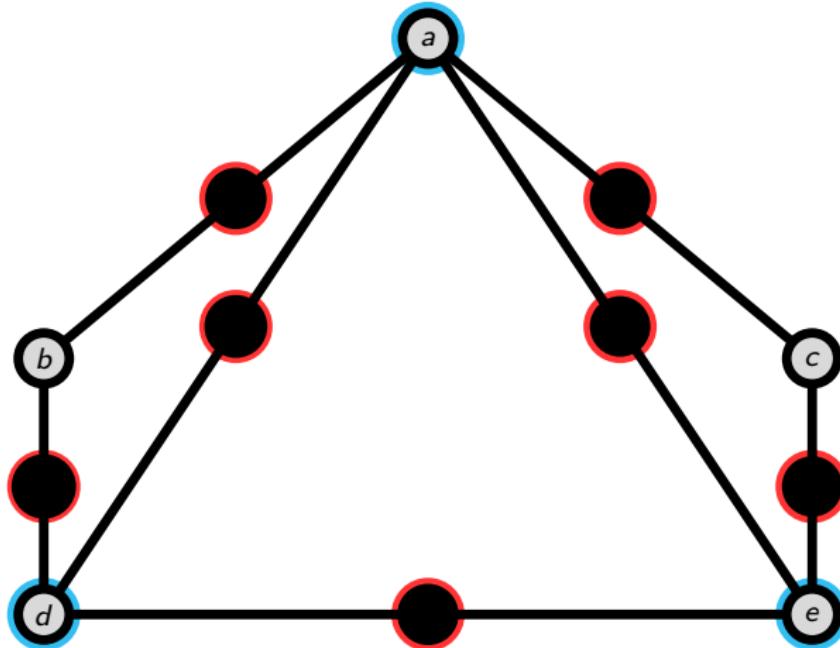
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



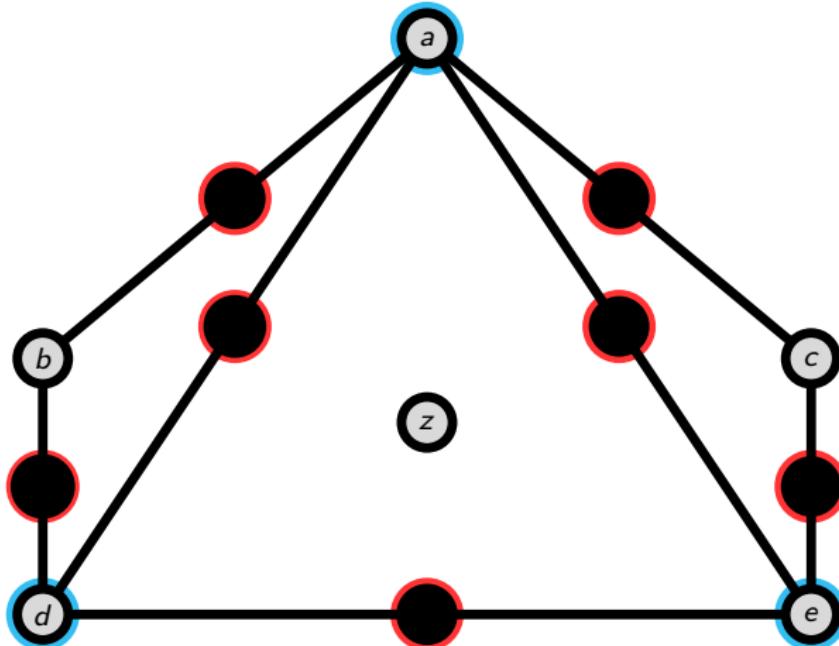
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



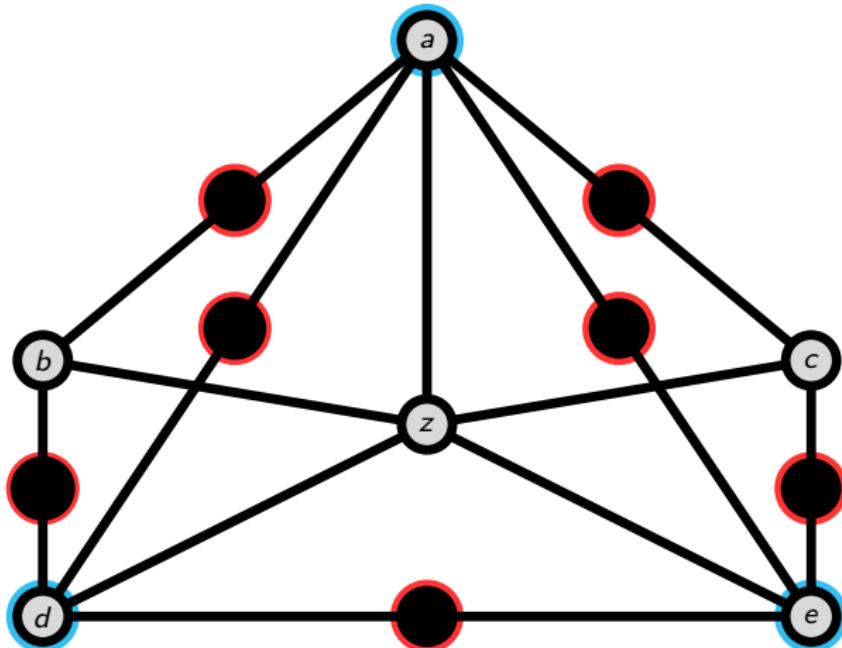
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in NPC (Reduktion von Vertex Cover).



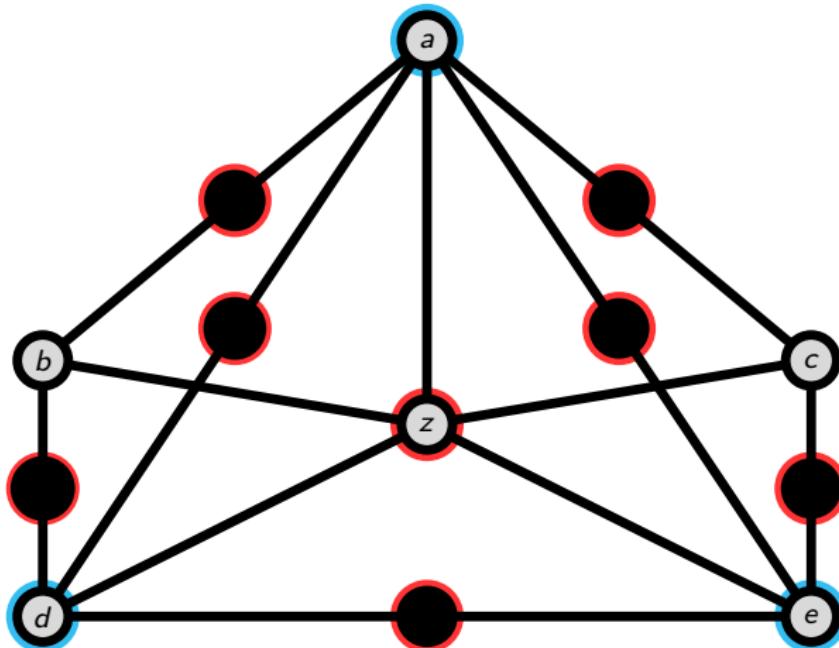
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



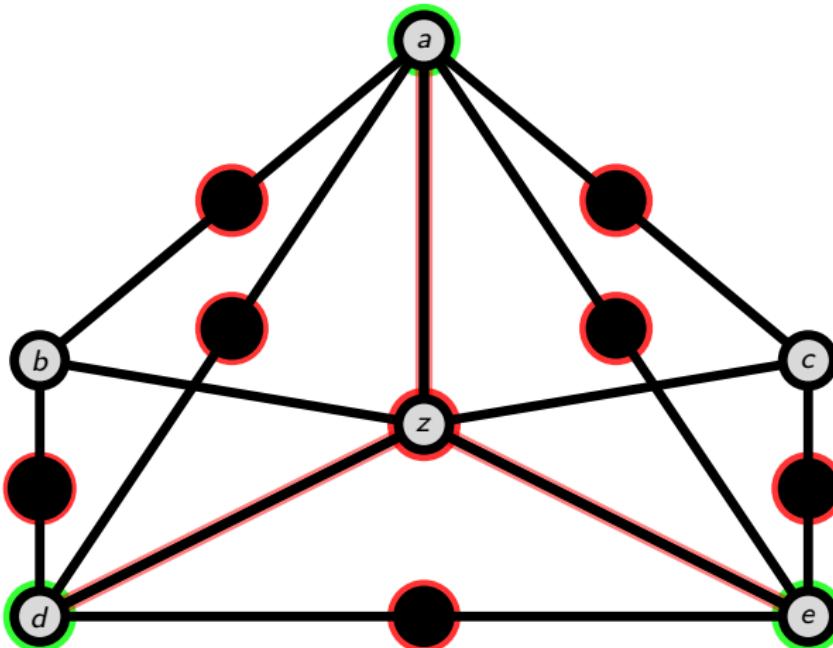
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



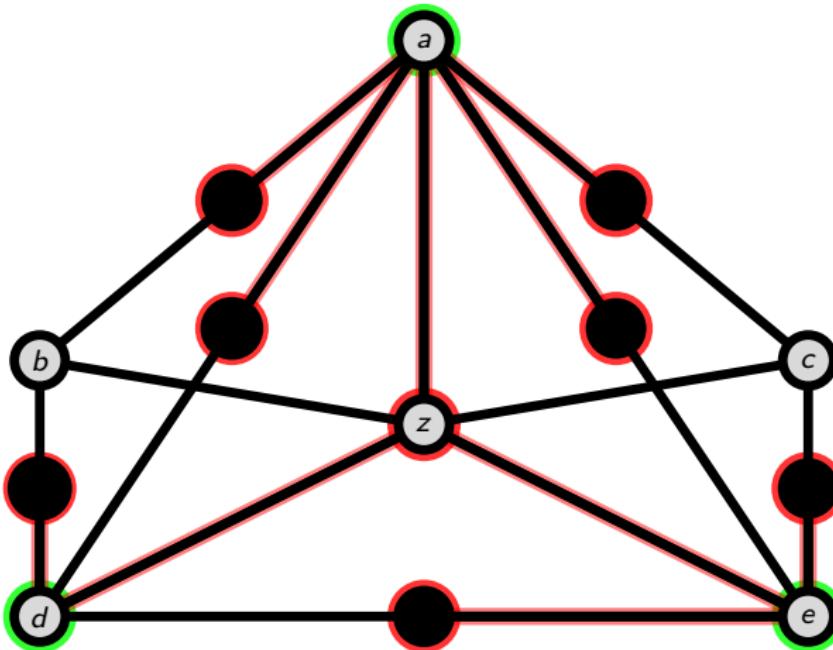
Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



Steinerbaum (Idee der Reduktion)

Zeige: Das Steinerbaumproblem ist in \mathcal{NPC} (Reduktion von Vertex Cover).



Steinerbaum

Theorem

Das Steiner Baum Problem ist NP-vollständig.

Steinerbaum

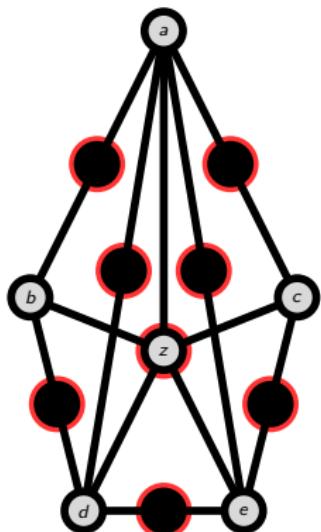
Theorem

Das Steiner Baum Problem ist NP-vollständig.

Steinerbaum

Theorem

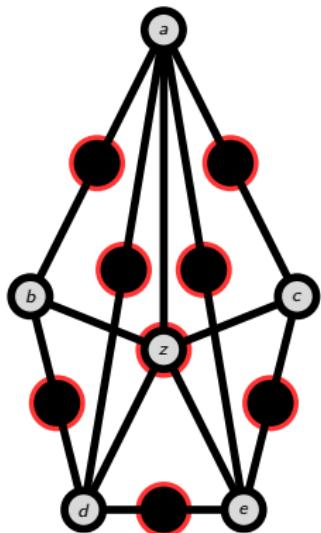
Das Steiner Baum Problem ist \mathcal{NP} -vollständig.



Steinerbaum

Theorem

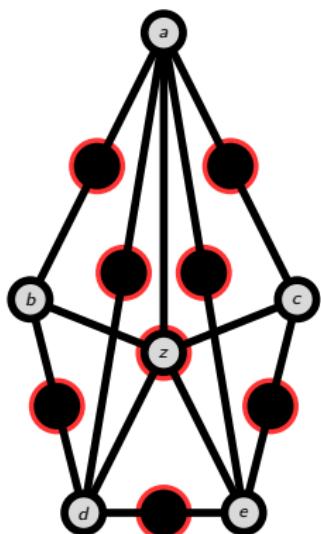
Das Steiner Baum Problem ist \mathcal{NP} -vollständig.



Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

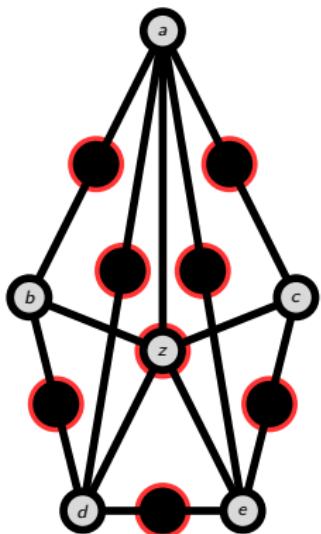


- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

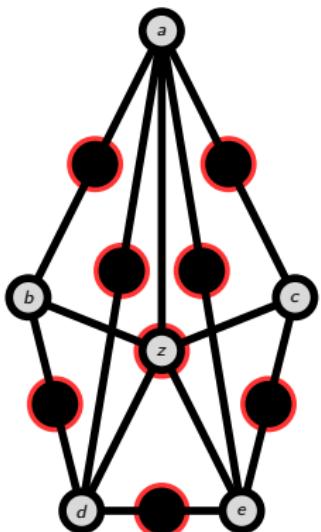


- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems
- Konstruiere Graphen G' aus G wie folgt:

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

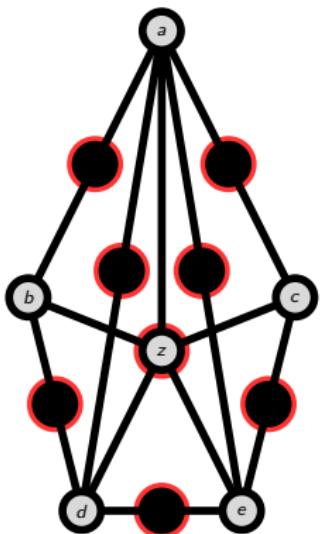


- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems
- Konstruiere Graphen G' aus G wie folgt:
 - $G' = (V \cup V_e \cup \{z\}, F \cup E_z)$

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

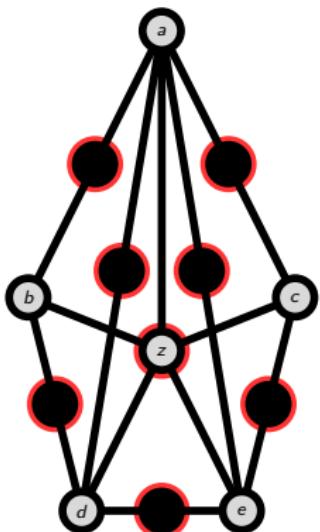


- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems
- Konstruiere Graphen G' aus G wie folgt:
 - $G' = (V \cup V_e \cup \{z\}, F \cup E_z)$
 - $V_e = \{v_e \mid e \in E\}$

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

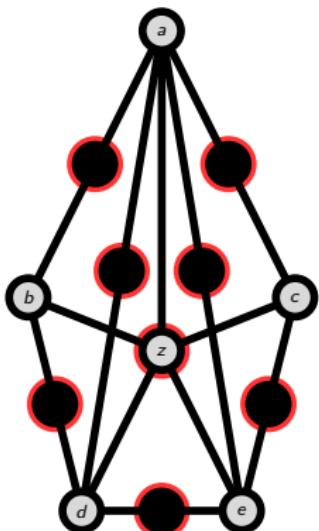


- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems
- Konstruiere Graphen G' aus G wie folgt:
 - $G' = (V \cup V_e \cup \{z\}, F \cup E_z)$
 - $V_e = \{v_e \mid e \in E\}$
 - $F = \{\{a, v_e\}, \{v_e, b\} \mid e = \{a, b\} \in E\}$

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

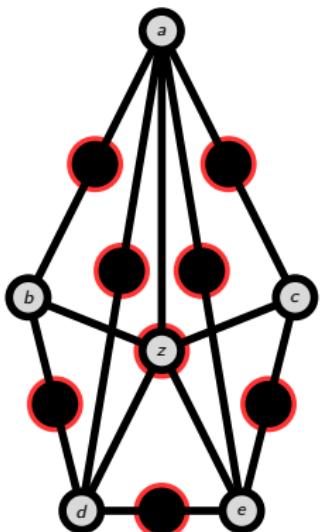


- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems
- Konstruiere Graphen G' aus G wie folgt:
 - $G' = (V \cup V_e \cup \{z\}, F \cup E_z)$
 - $V_e = \{v_e \mid e \in E\}$
 - $F = \{\{a, v_e\}, \{v_e, b\} \mid e = \{a, b\} \in E\}$
 - $E_z = \{\{v, z\} \mid v \in V\}$

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.



- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems
- Konstruiere Graphen G' aus G wie folgt:
 - $G' = (V \dot{\cup} V_e \dot{\cup} \{z\}, F \cup E_z)$
 - $V_e = \{v_e \mid e \in E\}$
 - $F = \{\{a, v_e\}, \{v_e, b\} \mid e = \{a, b\} \in E\}$
 - $E_z = \{\{v, z\} \mid v \in V\}$
 - Setze $T = V_e \cup \{z\}$.

Steinerbaum

Theorem

Das Steiner Baum Problem ist NP-vollständig.

Steinerbaum

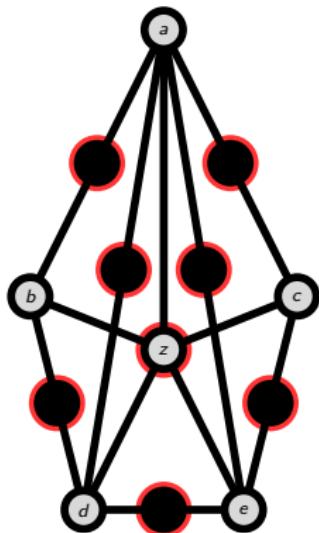
Theorem

Das Steiner Baum Problem ist NP-vollständig.

Steinerbaum

Theorem

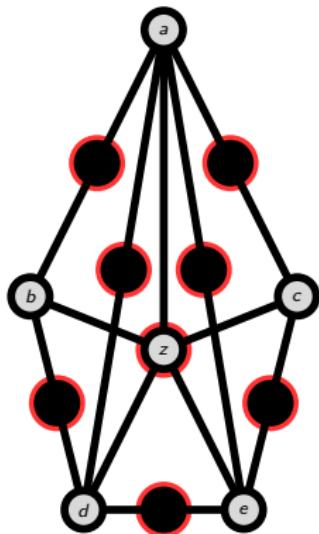
Das Steiner Baum Problem ist \mathcal{NP} -vollständig.



Steinerbaum

Theorem

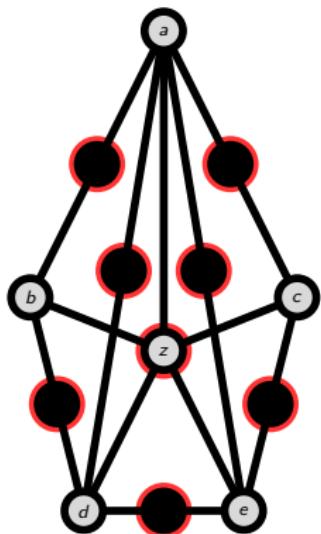
Das Steiner Baum Problem ist \mathcal{NP} -vollständig.



Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

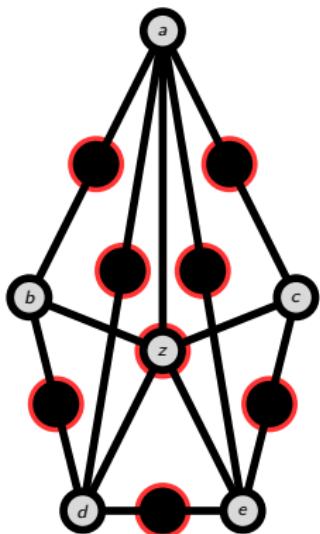


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

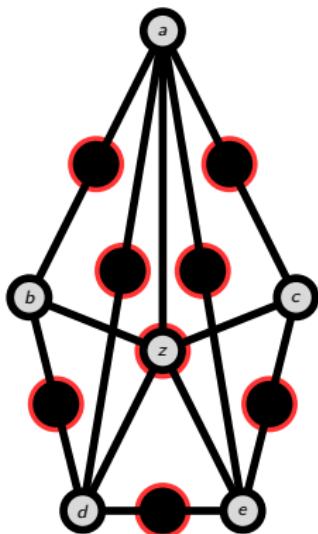


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

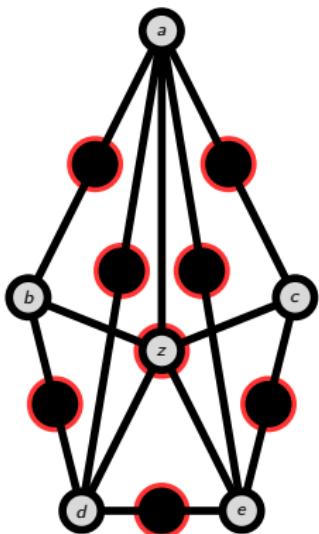


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

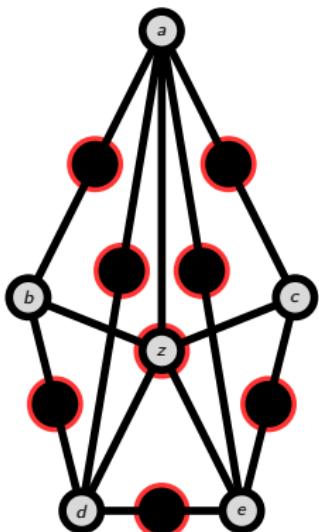


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.
- Beachte: $T = V_e \cup \{z\}$.

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

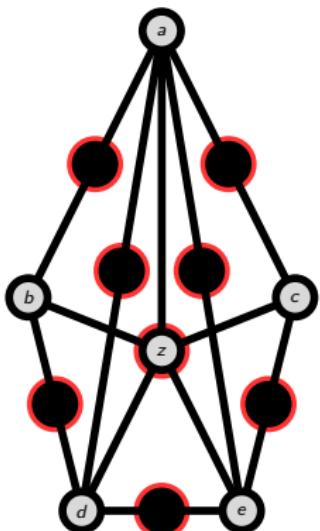


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.
- Beachte: $T = V_e \cup \{z\}$.
- G' wird $\tau(G)$ Steinerpunkte haben.

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

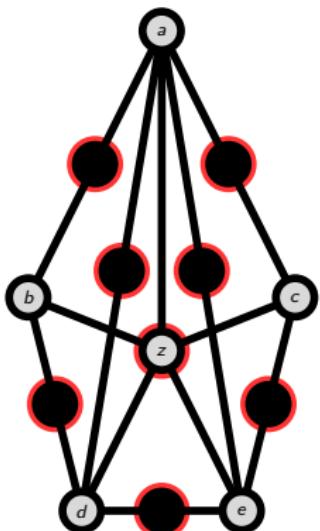


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.
- Beachte: $T = V_e \cup \{z\}$.
- G' wird $\tau(G)$ Steinerpunkte haben.
- Der Steinerbaum wird $\gamma(G') = \tau(G) + m$ Kanten haben.

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

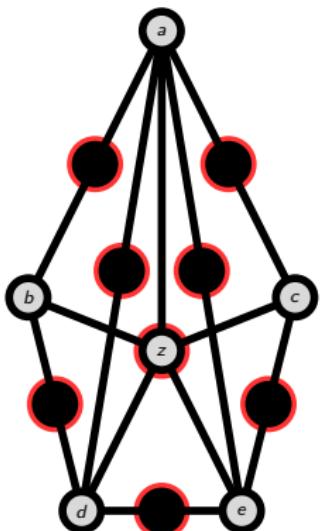


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.
- Beachte: $T = V_e \cup \{z\}$.
- G' wird $\tau(G)$ Steinerpunkte haben.
- Der Steinerbaum wird $\gamma(G') = \tau(G) + m$ Kanten haben.
- Zeige im Weiteren:

Steinerbaum

Theorem

Das Steiner Baum Problem ist \mathcal{NP} -vollständig.

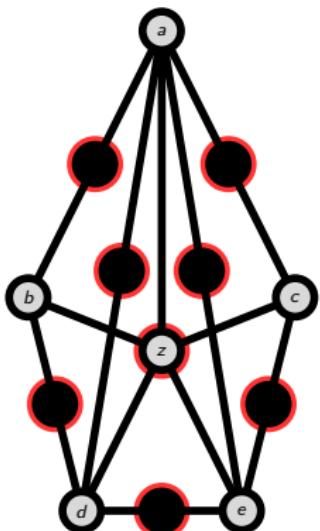


- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.
- Beachte: $T = V_e \cup \{z\}$.
- G' wird $\tau(G)$ Steinerpunkte haben.
- Der Steinerbaum wird $\gamma(G') = \tau(G) + m$ Kanten haben.
- Zeige im Weiteren:
 - $\gamma(G') \leq \tau(G) + m$

Steinerbaum

Theorem

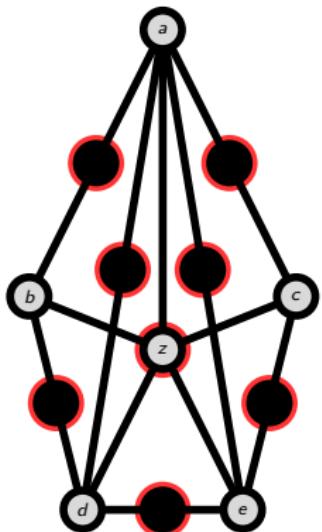
Das Steiner Baum Problem ist \mathcal{NP} -vollständig.



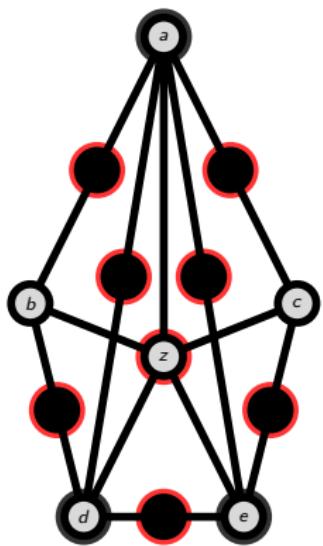
- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.
- Beachte: $T = V_e \cup \{z\}$.
- G' wird $\tau(G)$ Steinerpunkte haben.
- Der Steinerbaum wird $\gamma(G') = \tau(G) + m$ Kanten haben.
- Zeige im Weiteren:
 - $\gamma(G') \leq \tau(G) + m$
 - $\gamma(G') \geq \tau(G) + m$

Steinerbaum (Beweis)

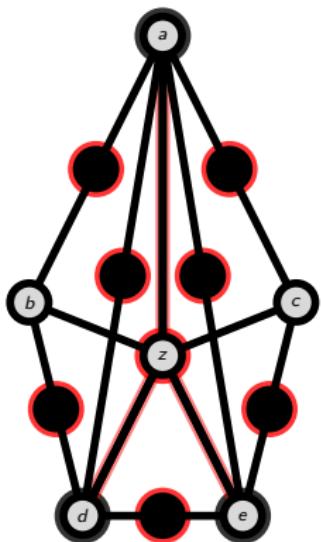
Steinerbaum (Beweis)



Steinerbaum (Beweis)

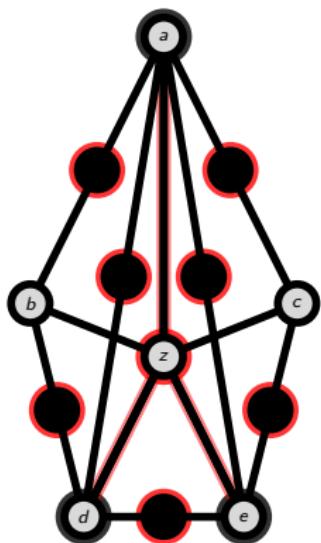


Steinerbaum (Beweis)



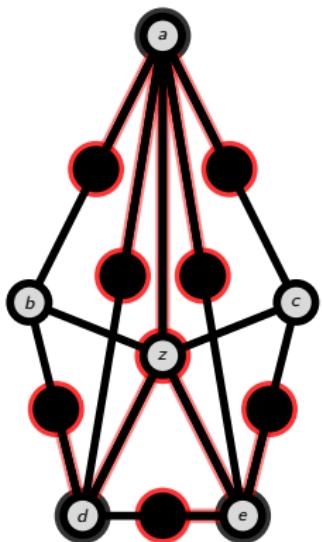
- Zeige: $\gamma(G') \leq \tau(G) + m$

Steinerbaum (Beweis)



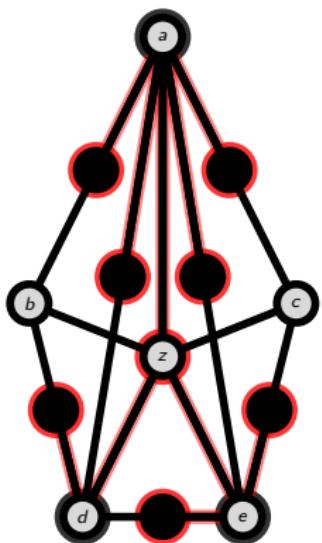
- Zeige: $\gamma(G') \leq \tau(G) + m$
- Sei C Vertex Cover der Größe $\tau(G)$.

Steinerbaum (Beweis)



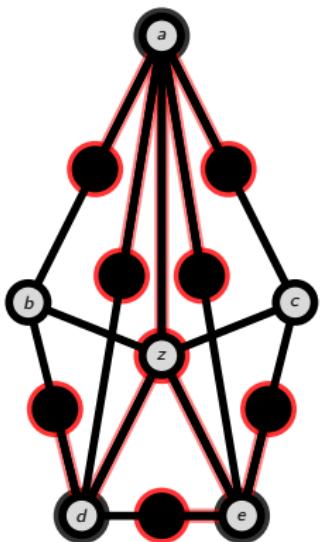
- Zeige: $\gamma(G') \leq \tau(G) + m$
- Sei C Vertex Cover der Größe $\tau(G)$.
- Wähle C als Steinerpunkte von G' .

Steinerbaum (Beweis)



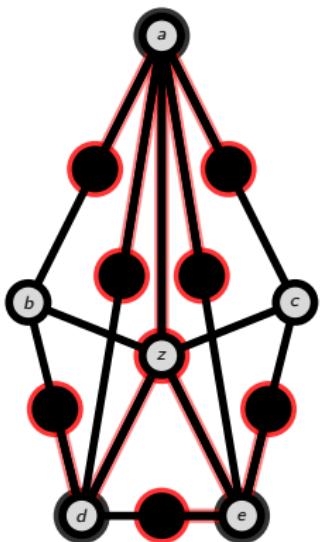
- Zeige: $\gamma(G') \leq \tau(G) + m$
- Sei C Vertex Cover der Größe $\tau(G)$.
- Wähle C als Steinerpunkte von G' .
- Mit $\tau(G)$ Kanten werden die Knoten aus C mit z verbunden.

Steinerbaum (Beweis)



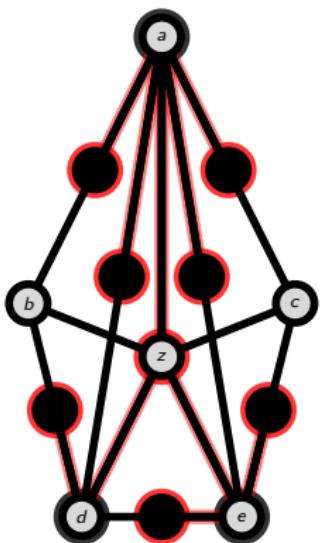
- Zeige: $\gamma(G') \leq \tau(G) + m$
 - Sei C Vertex Cover der Größe $\tau(G)$.
 - Wähle C als Steinerpunkte von G' .
 - Mit $\tau(G)$ Kanten werden die Knoten aus C mit z verbunden.
 - Da C ein Vertex Cover ist,
ist jeder Knoten aus V_e mit einem Knoten aus C verbunden.

Steinerbaum (Beweis)



- Zeige: $\gamma(G') \leq \tau(G) + m$
- Sei C Vertex Cover der Größe $\tau(G)$.
- Wähle C als Steinerpunkte von G' .
- Mit $\tau(G)$ Kanten werden die Knoten aus C mit z verbunden.
- Da C ein Vertex Cover ist, ist jeder Knoten aus V_e mit einem Knoten aus C verbunden.
- Damit sind alle Knoten aus V_e mit jeweils einem Knoten aus C verbunden.

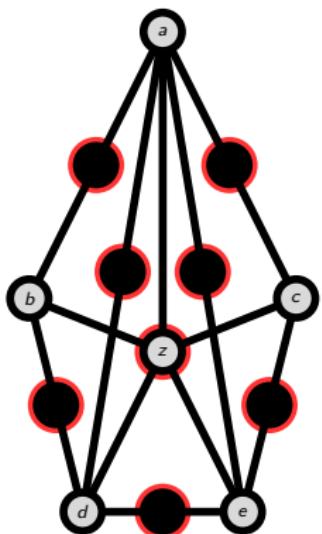
Steinerbaum (Beweis)



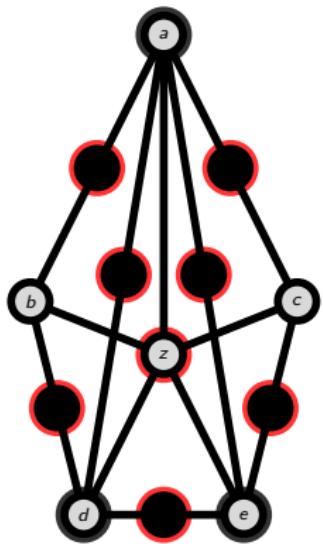
- Zeige: $\gamma(G') \leq \tau(G) + m$
- Sei C Vertex Cover der Größe $\tau(G)$.
- Wähle C als Steinerpunkte von G' .
- Mit $\tau(G)$ Kanten werden die Knoten aus C mit z verbunden.
- Da C ein Vertex Cover ist, ist jeder Knoten aus V_e mit einem Knoten aus C verbunden.
- Damit sind alle Knoten aus V_e mit jeweils einem Knoten aus C verbunden.
- Das sind dann nochmals m Kanten

Steinerbaum (Beweis)

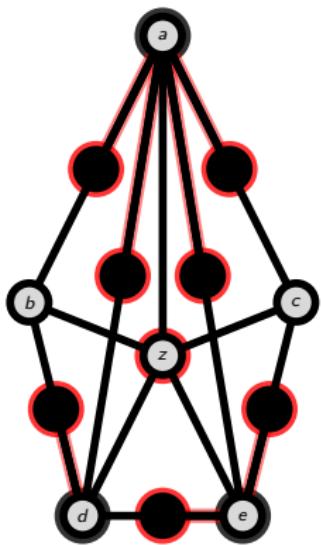
Steinerbaum (Beweis)



Steinerbaum (Beweis)

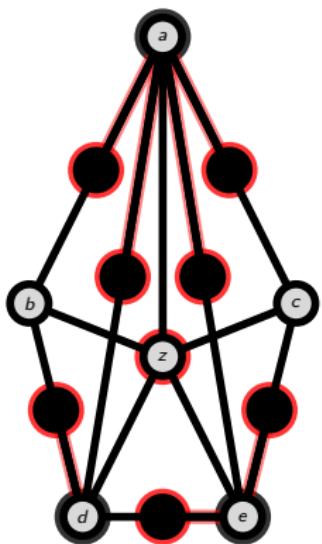


Steinerbaum (Beweis)



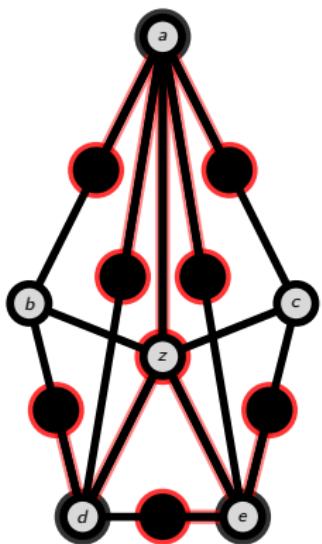
- Zeige: $\gamma(G') \geq \tau(G) + m$

Steinerbaum (Beweis)



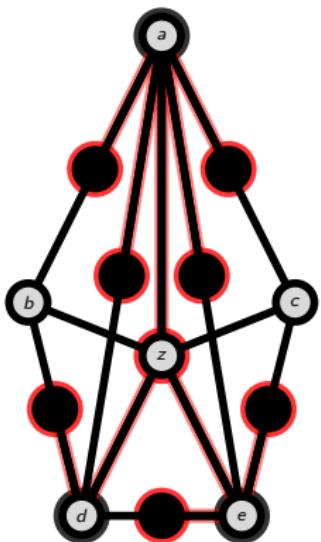
- Zeige: $\gamma(G') \geq \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.

Steinerbaum (Beweis)



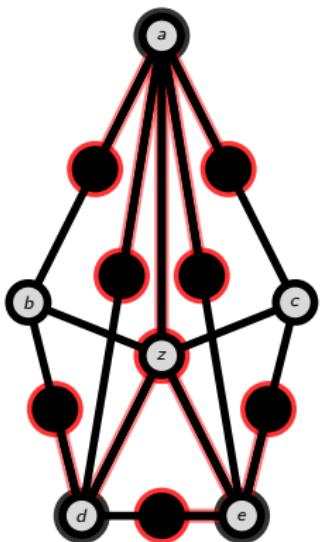
- Zeige: $\gamma(G') \geq \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.
- Die Knoten $C = V(S) \setminus T$ sind dann die Steinerknoten.

Steinerbaum (Beweis)



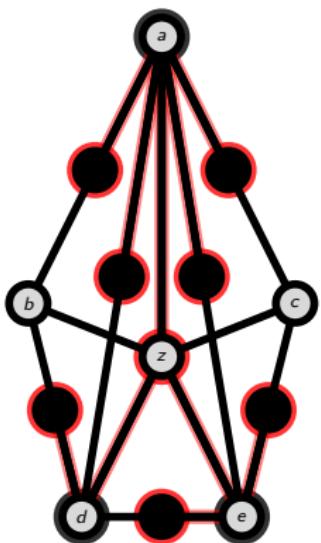
- Zeige: $\gamma(G') \geq \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.
- Die Knoten $C = V(S) \setminus T$ sind dann die Steinerknoten.
- Jeder Knoten aus V_e ist mit einem Knoten aus C über S verbunden.

Steinerbaum (Beweis)



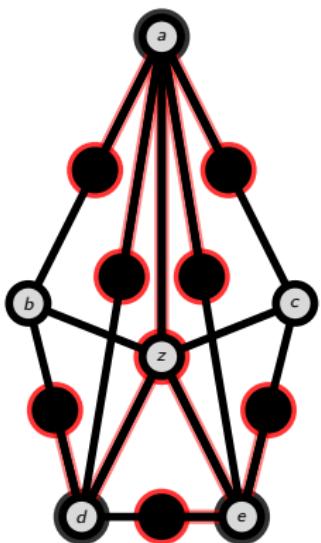
- Zeige: $\gamma(G') \geq \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.
- Die Knoten $C = V(S) \setminus T$ sind dann die Steinerknoten.
- Jeder Knoten aus V_e ist mit einem Knoten aus C über S verbunden.
- Damit ist C ein Vertex Cover für G .

Steinerbaum (Beweis)



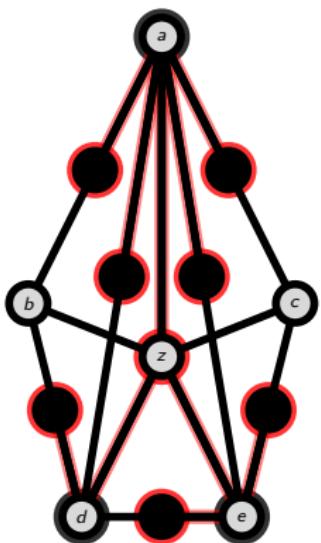
- Zeige: $\gamma(G') \geq \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.
- Die Knoten $C = V(S) \setminus T$ sind dann die Steinerknoten.
- Jeder Knoten aus V_e ist mit einem Knoten aus C über S verbunden.
- Damit ist C ein Vertex Cover für G .
- Weiter hat S $\gamma(G') + 1$ Knoten.

Steinerbaum (Beweis)



- Zeige: $\gamma(G') \geq \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.
- Die Knoten $C = V(S) \setminus T$ sind dann die Steinerknoten.
- Jeder Knoten aus V_e ist mit einem Knoten aus C über S verbunden.
- Damit ist C ein Vertex Cover für G .
- Weiter hat S $\gamma(G') + 1$ Knoten.
- $\tau(G) \leq |C| = \gamma(G') + 1 - (m + 1) = \gamma(G') - m$.

Steinerbaum (Beweis)



- Zeige: $\gamma(G') \geq \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.
- Die Knoten $C = V(S) \setminus T$ sind dann die Steinerknoten.
- Jeder Knoten aus V_e ist mit einem Knoten aus C über S verbunden.
- Damit ist C ein Vertex Cover für G .
- Weiter hat S $\gamma(G') + 1$ Knoten.
- $\tau(G) \leq |C| = \gamma(G') + 1 - (m + 1) = \gamma(G') - m$.
- $\tau(G) + m \leq \gamma(G')$.

Steinerbaum (Approximation)

- Aufbau der Idee

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.
- Eine kostenminimale Menge von diesen kürzesten Wegen, die alle Knoten aus T verbinden, sind kein untere Schranke.

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.
- Eine kostenminimale Menge von diesen kürzesten Wegen, die alle Knoten aus T verbinden, sind kein untere Schranke.
- Aber wir werden sehen, das liefert eine 2-Approximation.

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.
- Eine kostenminimale Menge von diesen kürzesten Wegen, die alle Knoten aus T verbinden, sind kein untere Schranke.
- Aber wir werden sehen, das liefert eine 2-Approximation.
- Damit ergibt sich das folgende Vorgehen:

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.
- Eine kostenminimale Menge von diesen kürzesten Wegen, die alle Knoten aus T verbinden, sind kein untere Schranke.
- Aber wir werden sehen, das liefert eine 2-Approximation.
- Damit ergibt sich das folgende Vorgehen:
 - Bestimme paarweise kürzeste Wege.

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.
- Eine kostenminimale Menge von diesen kürzesten Wegen, die alle Knoten aus T verbinden, sind kein untere Schranke.
- Aber wir werden sehen, das liefert eine 2-Approximation.
- Damit ergibt sich das folgende Vorgehen:
 - Bestimme paarweise kürzeste Wege.
 - Bestimme kostenminimale Menge kürzester Wege, die T verbinden.

Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.
- Eine kostenminimale Menge von diesen kürzesten Wegen, die alle Knoten aus T verbinden, sind kein untere Schranke.
- Aber wir werden sehen, das liefert eine 2-Approximation.
- Damit ergibt sich das folgende Vorgehen:
 - Bestimme paarweise kürzeste Wege.
 - Bestimme kostenminimale Menge kürzester Wege, die T verbinden.
 - Bestimme daraus den Steinerbaum.

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:
 - Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:
 - Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
 - Bestimme Clique $G_D = (T, F)$ und

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:
 - Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
 - Bestimme Clique $G_D = (T, F)$ und
 - $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:

- Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
- Bestimme Clique $G_D = (T, F)$ und
 - $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.
- Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D .

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:
 - ➊ Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
 - ➋ Bestimme Clique $G_D = (T, F)$ und
 - ➌ $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.
 - ➍ Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D .
 - ➎ Für jede Kante $e \in F'$ sei $W_e = (V_e, F_e)$ ein Weg der Länge $c_D(e)$ in G .

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:

- Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
- Bestimme Clique $G_D = (T, F)$ und
 - $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.
- Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D .
- Für jede Kante $e \in F'$ sei $W_e = (V_e, F_e)$ ein Weg der Länge $c_D(e)$ in G .
- Bestimme $H = (\cup_{e \in F'} V_e, \cup_{e \in F'} F_e)$ als Teilgraph von G .

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:

- Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
- Bestimme Clique $G_D = (T, F)$ und
 - $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.
- Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D .
- Für jede Kante $e \in F'$ sei $W_e = (V_e, F_e)$ ein Weg der Länge $c_D(e)$ in G .
- Bestimme $H = (\cup_{e \in F'} V_e, \cup_{e \in F'} F_e)$ als Teilgraph von G .
- Bestimme minimalen Spannbaum S_H auf H .

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:

- Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
- Bestimme Clique $G_D = (T, F)$ und
 - $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.
- Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D .
- Für jede Kante $e \in F'$ sei $W_e = (V_e, F_e)$ ein Weg der Länge $c_D(e)$ in G .
- Bestimme $H = (\cup_{e \in F'} V_e, \cup_{e \in F'} F_e)$ als Teilgraph von G .
- Bestimme minimalen Spannbaum S_H auf H .
- Lösche sukzessive Blätter aus S_H , die nicht in T sind.

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:

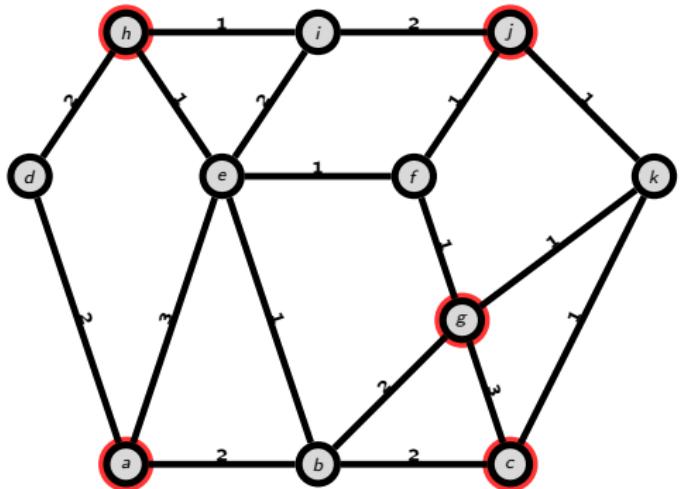
- Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
- Bestimme Clique $G_D = (T, F)$ und
 - $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.
- Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D .
- Für jede Kante $e \in F'$ sei $W_e = (V_e, F_e)$ ein Weg der Länge $c_D(e)$ in G .
- Bestimme $H = (\cup_{e \in F'} V_e, \cup_{e \in F'} F_e)$ als Teilgraph von G .
- Bestimme minimalen Spannbaum S_H auf H .
- Lösche sukzessive Blätter aus S_H , die nicht in T sind.
- Ergebnis davon ist dann der Baum S_{KMB} .

Steinerbaum (Approximation)

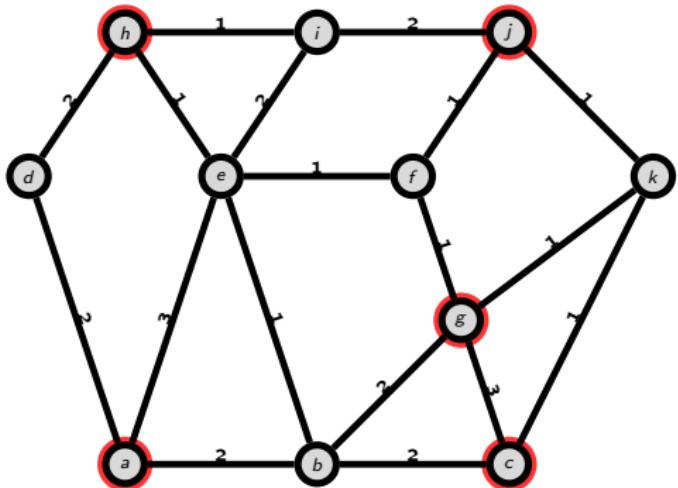
- Verfahren von Kou, Markowsky und Berman:
 - ➊ Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
 - ➋ Bestimme Clique $G_D = (T, F)$ und
 - ➌ $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = \text{dist}_G(a, b)$.
 - ➍ Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D .
 - ➎ Für jede Kante $e \in F'$ sei $W_e = (V_e, F_e)$ ein Weg der Länge $c_D(e)$ in G .
 - ➏ Bestimme $H = (\cup_{e \in F'} V_e, \cup_{e \in F'} F_e)$ als Teilgraph von G .
 - ➐ Bestimme minimalen Spannbaum S_H auf H .
 - ➑ Lösche sukzessive Blätter aus S_H , die nicht in T sind.
 - ➒ Ergebnis davon ist dann der Baum S_{KMB} .
- Damit gilt: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.

Beispiel

Beispiel



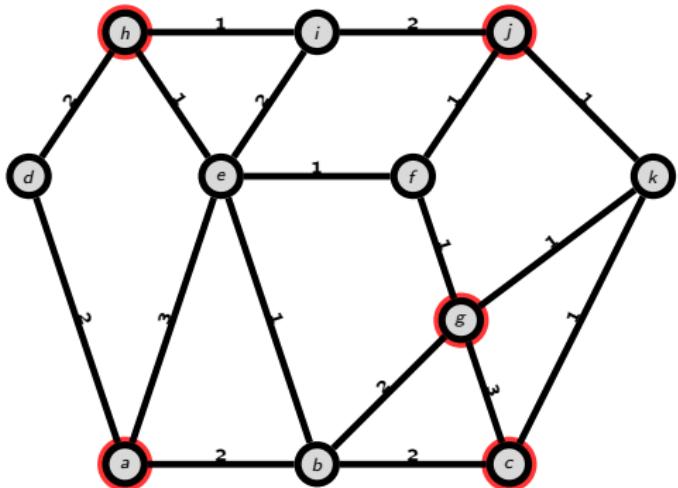
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D

x	y	$c(x,y)$	$\in S_D$
c	g	2	
g	j	2	
c	j	2	
h	j	3	
g	h	3	
a	c	4	
a	g	4	
a	h	4	
c	h	4	
a	j	5	

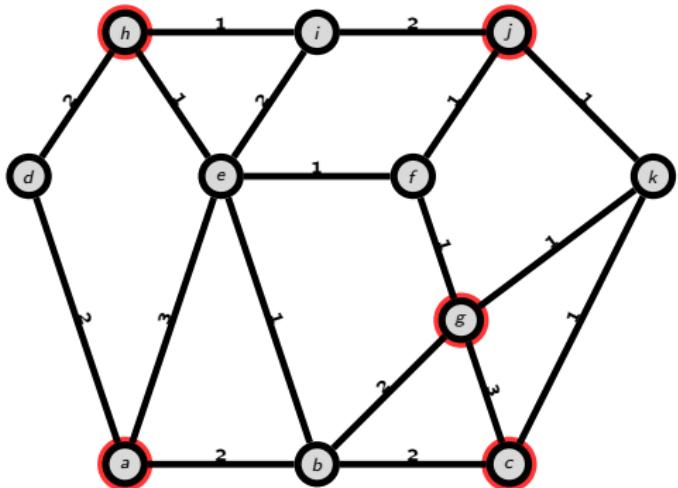
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	
c	j	2	
h	j	3	
g	h	3	
a	c	4	
a	g	4	
a	h	4	
c	h	4	
a	j	5	

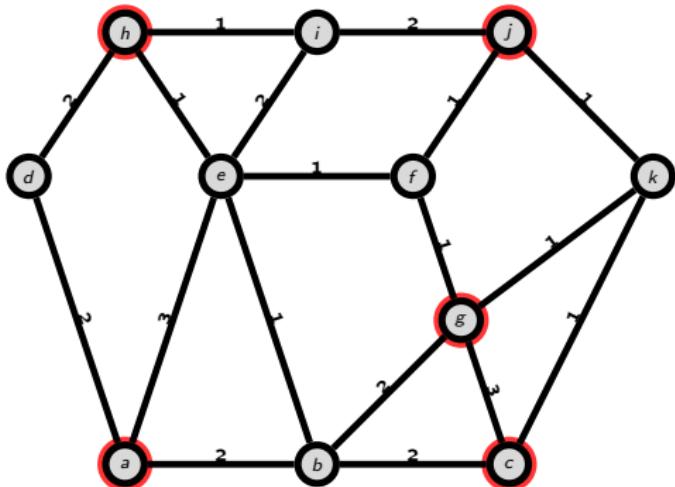
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	
g	h	3	
a	c	4	
a	g	4	
a	h	4	
c	h	4	
a	j	5	

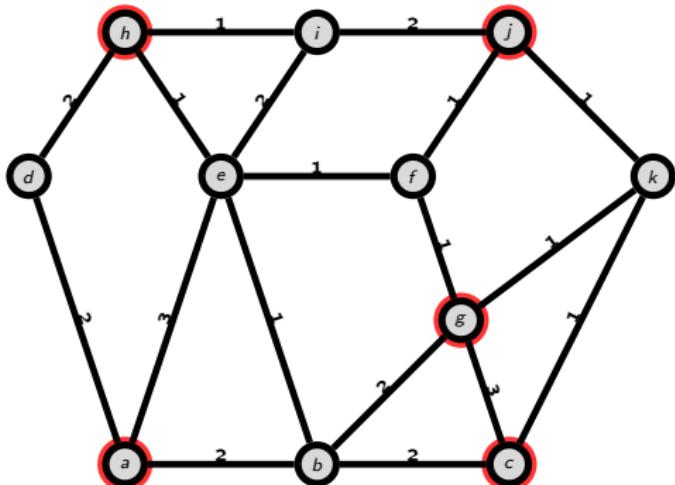
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	3
g	h	3	
a	c	4	
a	g	4	
a	h	4	
c	h	4	
a	j	5	

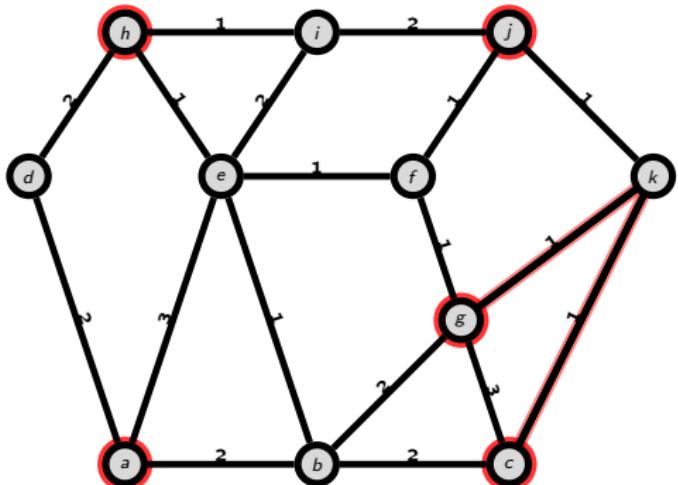
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D
- Bestimme passende Wege in G .

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	3
g	h	3	
a	c	4	4
a	g	4	
a	h	4	
c	h	4	
a	j	5	

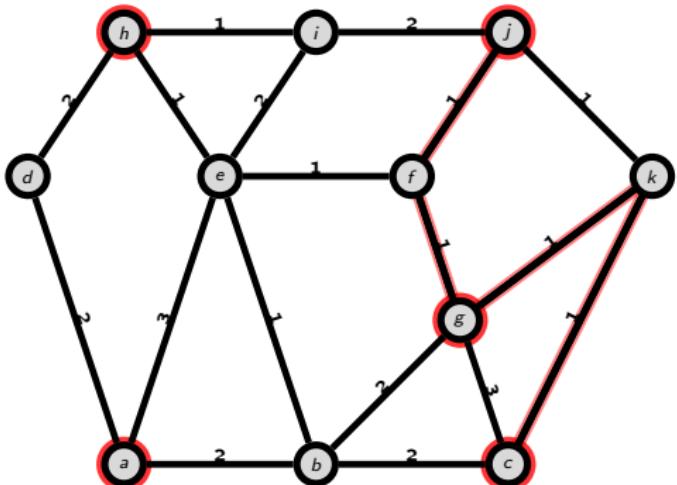
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D
- Bestimme passende Wege in G .

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	3
g	h	3	
a	c	4	4
a	g	4	
a	h	4	
c	h	4	
a	j	5	

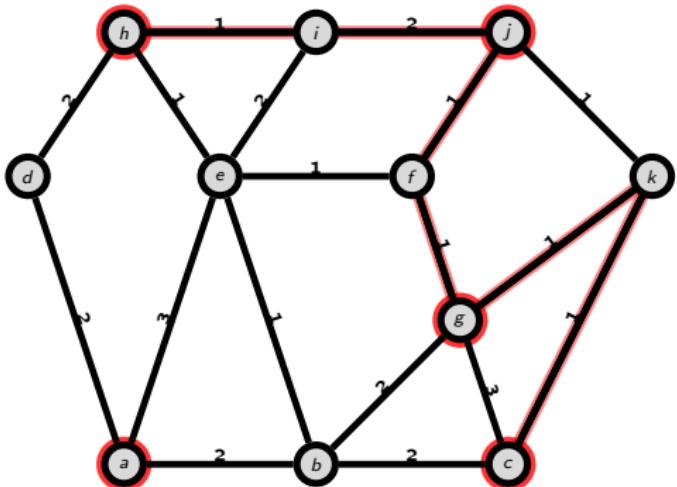
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D
- Bestimme passende Wege in G .

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	3
g	h	3	
a	c	4	4
a	g	4	
a	h	4	
c	h	4	
a	j	5	

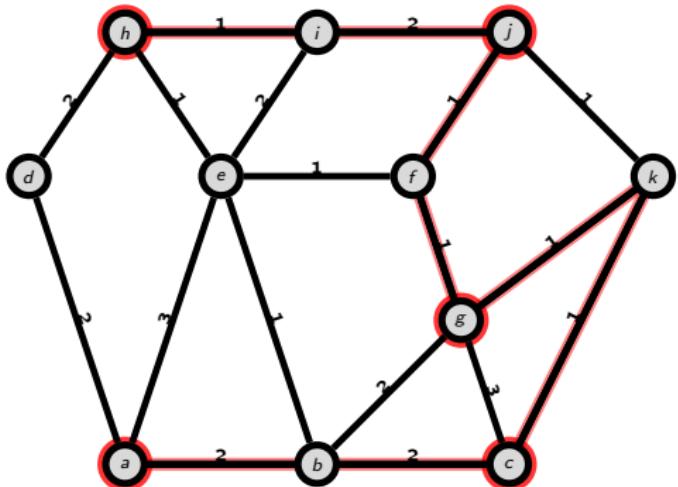
Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D
- Bestimme passende Wege in G .

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	3
g	h	3	
a	c	4	4
a	g	4	
a	h	4	
c	h	4	
a	j	5	

Beispiel



- Erzeuge G_D und c_D :
- Bestimme S_D auf G_D
- Bestimme passende Wege in G .
- Kosten: 11.

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	3
g	h	3	
a	c	4	4
a	g	4	
a	h	4	
c	h	4	
a	j	5	

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätzt nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .
- Damit ergibt sich Zyklus Z , der alle Knoten aus T trifft.

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .
- Damit ergibt sich Zyklus Z , der alle Knoten aus T trifft.
- Z definiert spannenden Baum auf G_D :

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .
- Damit ergibt sich Zyklus Z , der alle Knoten aus T trifft.
- Z definiert spannenden Baum auf G_D :
 - Start bei $s_1 \in T$ auf Z .

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .
- Damit ergibt sich Zyklus Z , der alle Knoten aus T trifft.
- Z definiert spannenden Baum auf G_D :
 - Start bei $s_1 \in T$ auf Z .
 - Sei $s_2, s_3, \dots, s_{|T|}$ die Folge der Erstbesuche der Knoten aus T .

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .
- Damit ergibt sich Zyklus Z , der alle Knoten aus T trifft.
- Z definiert spannenden Baum auf G_D :
 - Start bei $s_1 \in T$ auf Z .
 - Sei $s_2, s_3, \dots, s_{|T|}$ die Folge der Erstbesuche der Knoten aus T .
 - Die Kanten $\{s_i, s_{i+1}\}$ bilden einen Spannbaum T_Z auf G_D .

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .
- Damit ergibt sich Zyklus Z , der alle Knoten aus T trifft.
- Z definiert spannenden Baum auf G_D :
 - Start bei $s_1 \in T$ auf Z .
 - Sei $s_2, s_3, \dots, s_{|T|}$ die Folge der Erstbesuche der Knoten aus T .
 - Die Kanten $\{s_i, s_{i+1}\}$ bilden einen Spannbaum T_Z auf G_D .
 - Also: $c(T_Z) \leq c(Z)$ und damit $c(T_Z) \leq c(Z) = 2 \cdot c(S^*)$.

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leq c(H) \leq c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leq 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^* .
- Damit ergibt sich Zyklus Z , der alle Knoten aus T trifft.
- Z definiert spannenden Baum auf G_D :
 - Start bei $s_1 \in T$ auf Z .
 - Sei $s_2, s_3, \dots, s_{|T|}$ die Folge der Erstbesuche der Knoten aus T .
 - Die Kanten $\{s_i, s_{i+1}\}$ bilden einen Spannbaum T_Z auf G_D .
 - Also: $c(T_Z) \leq c(Z)$ und damit $c(T_Z) \leq c(Z) = 2 \cdot c(S^*)$.
 - Insgesamt: $c(S_{KMB}) \leq c(H) \leq c(S_D) \leq c(T_Z) \leq c(Z) = 2 \cdot c(S^*)$.

Steinerbaum (Zusammenfassung)

Theorem

Das Steinerbaum Problem kann mit einem Faktor von 2 approximiert werden.

Steinerbaum (Zusammenfassung)

Theorem

Das Steinerbaum Problem kann mit einem Faktor von 2 approximiert werden.

Steinerbaum (Zusammenfassung)

Theorem

Das Steinerbaum Problem kann mit einem Faktor von 2 approximiert werden.

Theorem

Das Steinerbaum Problem kann mit einem Faktor von $\frac{11}{6}$ approximiert werden.

Wird hier nicht bewiesen.

Zentrumproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.

Zentrumproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$

Zentrumproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$

Zentrumproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} dist(v, Z)$

Zentrumproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} dist(v, Z)$

Zentrumproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} dist(v, Z)$

Definition (Zentrumproblem (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$, $L \in \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme $\exists Z : |Z| = k$ und $\text{rad}(Z) \leq L$.

Zentrumproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} dist(v, Z)$

Definition (Zentrumproblem (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$, $L \in \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme $\exists Z : |Z| = k$ und $\text{rad}(Z) \leq L$.

Zentrumssproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} dist(v, Z)$

Definition (Zentrumssproblem (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$, $L \in \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme $\exists Z : |Z| = k$ und $\text{rad}(Z) \leq L$.

Definition (Zentrumssproblem)

Gegeben $G = (V, E)$, $k \in \mathbb{N}$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme Z mit $|Z| = k$ und $\text{rad}(Z)$ minimal.

Komplexität

Theorem

Falls $\mathcal{P} \neq \text{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Komplexität

Theorem

Falls $\mathcal{P} \neq \text{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Komplexität

Theorem

Falls $\mathcal{P} \neq \text{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Komplexität

Theorem

Falls $\mathcal{P} \neq \text{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Definition (Dominating Set (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$.

Bestimme $\exists D : |D| = k$ und $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$.

Komplexität

Theorem

Falls $\mathcal{P} \neq \text{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Definition (Dominating Set (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$.

Bestimme $\exists D : |D| = k$ und $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$.

- Alternativ:

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Definition (Dominating Set (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$.

Bestimme $\exists D : |D| = k$ und $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$.

- Alternativ:

- $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$ genau dann, wenn

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Definition (Dominating Set (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$.

Bestimme $\exists D : |D| = k$ und $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$.

- Alternativ:

- $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$ genau dann, wenn
- $\text{rad}(D) = 1$.

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Definition (Dominating Set (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$.

Bestimme $\exists D : |D| = k$ und $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$.

- Alternativ:

- $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$ genau dann, wenn
- $\text{rad}(D) = 1$.

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationssfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Definition (Dominating Set (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$.

Bestimme $\exists D : |D| = k$ und $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$.

- Alternativ:

- $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$ genau dann, wenn
- $\text{rad}(D) = 1$.

Komplexität (Beweis)

- Reduktion vom Dominating Set Problem.

Komplexität (Beweis)

- Reduktion vom Dominating Set Problem.
- Sei (G, k) Eingabe für das Dominating Set Problem.

Komplexität (Beweis)

- Reduktion vom Dominating Set Problem.
- Sei (G, k) Eingabe für das Dominating Set Problem.
- Setze $c(e) = 1$ für $e \in E(G)$.

Komplexität (Beweis)

- Reduktion vom Dominating Set Problem.
- Sei (G, k) Eingabe für das Dominating Set Problem.
- Setze $c(e) = 1$ für $e \in E(G)$.
- Dann hat G Dominating Set der Größe k , falls G Zentrum mit k Knoten und Radius 1 hat.

Komplexität (Beweis)

- Reduktion vom Dominating Set Problem.
- Sei (G, k) Eingabe für das Dominating Set Problem.
- Setze $c(e) = 1$ für $e \in E(G)$.
- Dann hat G Dominating Set der Größe k , falls G Zentrum mit k Knoten und Radius 1 hat.
- Falls es einen Algorithmus A mit Approximationssfaktor $r < 2$ gibt,

Komplexität (Beweis)

- Reduktion vom Dominating Set Problem.
- Sei (G, k) Eingabe für das Dominating Set Problem.
- Setze $c(e) = 1$ für $e \in E(G)$.
- Dann hat G Dominating Set der Größe k , falls G Zentrum mit k Knoten und Radius 1 hat.
- Falls es einen Algorithmus A mit Approximationsfaktor $r < 2$ gibt,
- So liefert A bei Eingabe $(G, 1, k, c)$ ein Dominating Set Problem der Größe k .

Zentrumsproblem (mit Knotenkosten)

- Gegeben $G = (V, E)$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.

Zentrumssproblem (mit Knotenkosten)

- Gegeben $G = (V, E)$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$

Zentrumsproblem (mit Knotenkosten)

- Gegeben $G = (V, E)$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$

Zentrumssproblem (mit Knotenkosten)

- Gegeben $G = (V, E)$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} w(v) \cdot dist(v, Z)$ von nun an.

Zentrumssproblem (mit Knotenkosten)

- Gegeben $G = (V, E)$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} w(v) \cdot dist(v, Z)$ von nun an.

Definition (Zentrumssproblem)

Gegeben $G = (V, E)$, $k \in \mathbb{N}$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme Z mit $|Z| = k$ und $\text{rad}(Z)$ minimal.

Idee

- Wir kennen die Knoten aus Z nicht.

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.
- Idee: suche Z mit Greedy Verfahren für ein festes R .

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.
- Idee: suche Z mit Greedy Verfahren für ein festes R .
 - Solange noch Knoten nicht durch Z überdeckt sind.

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.
- Idee: suche Z mit Greedy Verfahren für ein festes R .
 - Solange noch Knoten nicht durch Z überdeckt sind.
 - Wähle mit Greedy einen weiteren Knoten z zu Z hinzu.

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.
- Idee: suche Z mit Greedy Verfahren für ein festes R .
 - Solange noch Knoten nicht durch Z überdeckt sind.
 - Wähle mit Greedy einen weiteren Knoten z zu Z hinzu.
 - Alle Knoten im Abstand $\underline{2 \cdot R}$ werden nun als überdeckt betrachtet.

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.
- Idee: suche Z mit Greedy Verfahren für ein festes R .
 - Solange noch Knoten nicht durch Z überdeckt sind.
 - Wähle mit Greedy einen weiteren Knoten z zu Z hinzu.
 - Alle Knoten im Abstand $2 \cdot R$ werden nun als überdeckt betrachtet.
- Achtung: der Algorithmus beinhaltet den Approximationsfaktor!

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.
- Idee: suche Z mit Greedy Verfahren für ein festes R .
 - Solange noch Knoten nicht durch Z überdeckt sind.
 - Wähle mit Greedy einen weiteren Knoten z zu Z hinzu.
 - Alle Knoten im Abstand $2 \cdot R$ werden nun als überdeckt betrachtet.
- Achtung: der Algorithmus beinhaltet den Approximationsfaktor!
- Teste nun verschiedene Werte R mit Hilfe dieses Greedyverfahrens.

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus $\text{GreedyZentrum}(G, c, w, R)$:
 - $Z := \emptyset$ und $U := V$

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):
 - $Z := \emptyset$ und $U := V$
 - Solange $U \neq \emptyset$ mache

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
- ④ Ausgabe: Z

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
- ④ Ausgabe: Z

- Sei R^* optimale Lösung mit Z^* . Falls $R \geq R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leq |Z^*|$ (Siehe Folie: 54).

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
- ④ Ausgabe: Z

- Sei R^* optimale Lösung mit Z^* . Falls $R \geq R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leq |Z^*|$ (Siehe Folie: 54).
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen (Siehe Folie: 55).

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
- ④ Ausgabe: Z

- Sei R^* optimale Lösung mit Z^* . Falls $R \geq R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leq |Z^*|$ (Siehe Folie: 54).
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen (Siehe Folie: 55).
- Definiere: $f(R) := |\operatorname{GreedyZentrum}(G, c, w, R)|$

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):

- ① $Z := \emptyset$ und $U := V$
- ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
- ④ Ausgabe: Z

- Sei R^* optimale Lösung mit Z^* . Falls $R \geq R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leq |Z^*|$ (Siehe Folie: 54).
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen (Siehe Folie: 55).
- Definiere: $f(R) := |\operatorname{GreedyZentrum}(G, c, w, R)|$
 - Beachte $f(R)$ ist aber nicht monoton (Siehe Folie: 52).

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):
 - ① $Z := \emptyset$ und $U := V$
 - ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
 - ④ Ausgabe: Z
- Sei R^* optimale Lösung mit Z^* . Falls $R \geq R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leq |Z^*|$ (Siehe Folie: 54).
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen (Siehe Folie: 55).
- Definiere: $f(R) := |\operatorname{GreedyZentrum}(G, c, w, R)|$
 - Beachte $f(R)$ ist aber nicht monoton (Siehe Folie: 52).
- Definiere: $R' = \max_{v \in V} w(v) \cdot \sum_{e \in E} c(e)$

Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):
 - ① $Z := \emptyset$ und $U := V$
 - ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
 - ④ Ausgabe: Z
- Sei R^* optimale Lösung mit Z^* . Falls $R \geq R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leq |Z^*|$ (Siehe Folie: 54).
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen (Siehe Folie: 55).
- Definiere: $f(R) := |\operatorname{GreedyZentrum}(G, c, w, R)|$
 - Beachte $f(R)$ ist aber nicht monoton (Siehe Folie: 52).
- Definiere: $R' = \max_{v \in V} w(v) \cdot \sum_{e \in E} c(e)$
 - Mit so einem großen Radius kann ein Knoten alles überdecken.

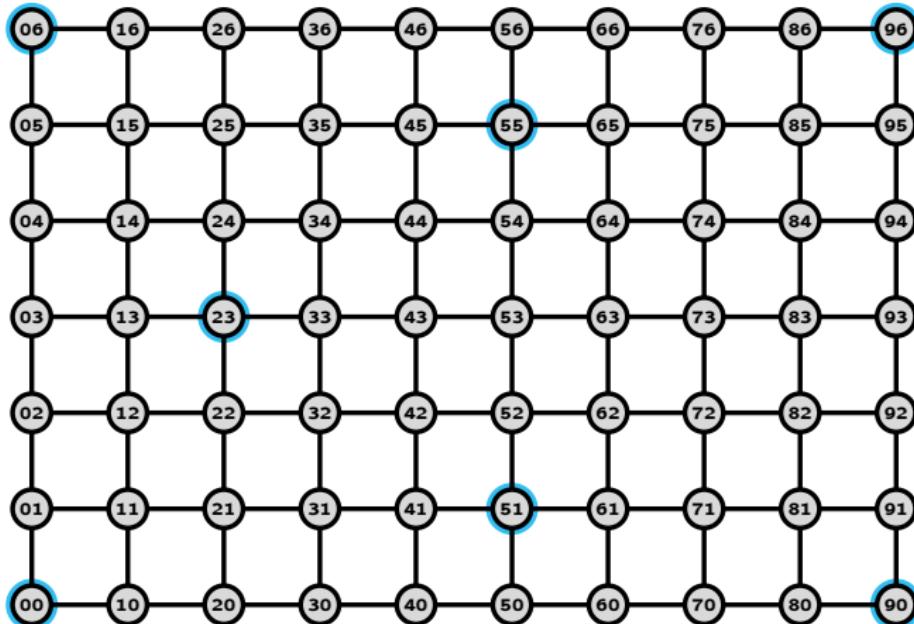
Approximation

$$\operatorname{argmax}\{X\} = x \text{ mit } c(x) = \max\{c(x') \mid x' \in X\}$$

- Algorithmus *GreedyZentrum*(G, c, w, R):
 - ① $Z := \emptyset$ und $U := V$
 - ② Solange $U \neq \emptyset$ mache
 - ① $z := \operatorname{argmax}\{w(u) \mid u \in U\}$
 - ② $Z := Z \cup \{z\}$
 - ③ $U := U \setminus \{v \in U \mid w(v) \cdot \operatorname{dist}(v, z) \leq 2 \cdot R\}$
 - ④ Ausgabe: Z
- Sei R^* optimale Lösung mit Z^* . Falls $R \geq R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leq |Z^*|$ (Siehe Folie: 54).
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen (Siehe Folie: 55).
- Definiere: $f(R) := |\operatorname{GreedyZentrum}(G, c, w, R)|$
 - Beachte $f(R)$ ist aber nicht monoton (Siehe Folie: 52).
- Definiere: $R' = \max_{v \in V} w(v) \cdot \sum_{e \in E} c(e)$
 - Mit so einem großen Radius kann ein Knoten alles überdecken.
 - $f(R') = 1$ und $f(0) = n$

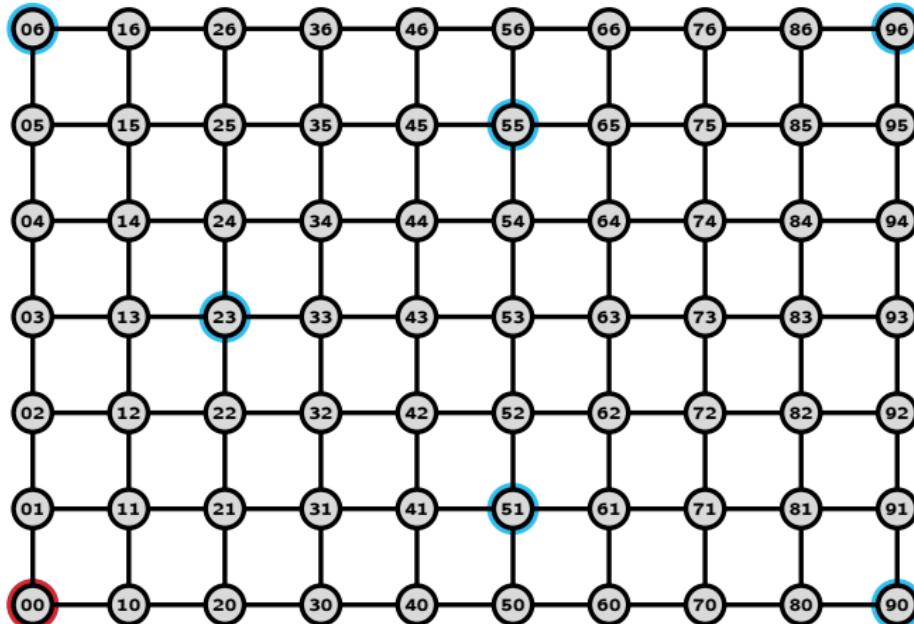
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



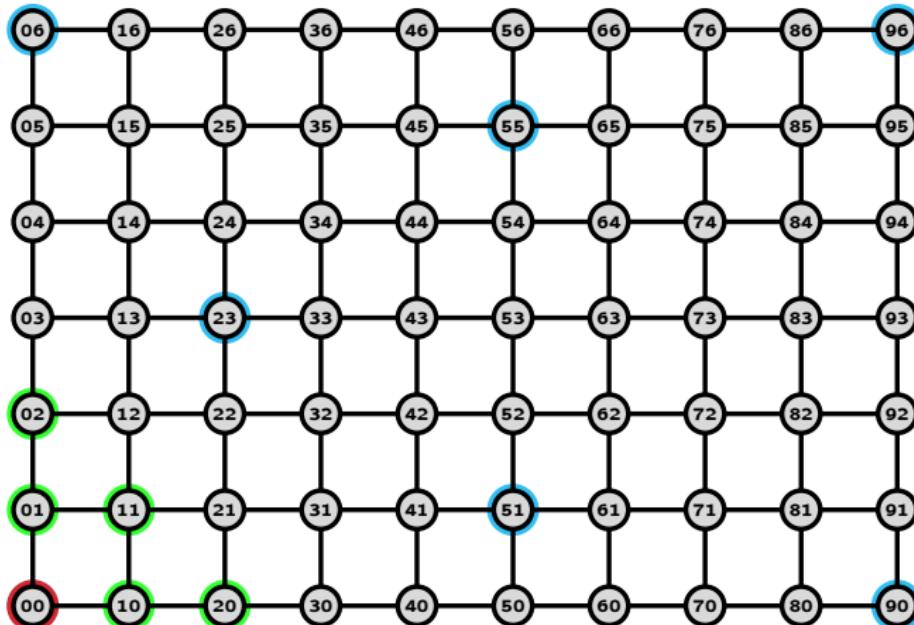
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



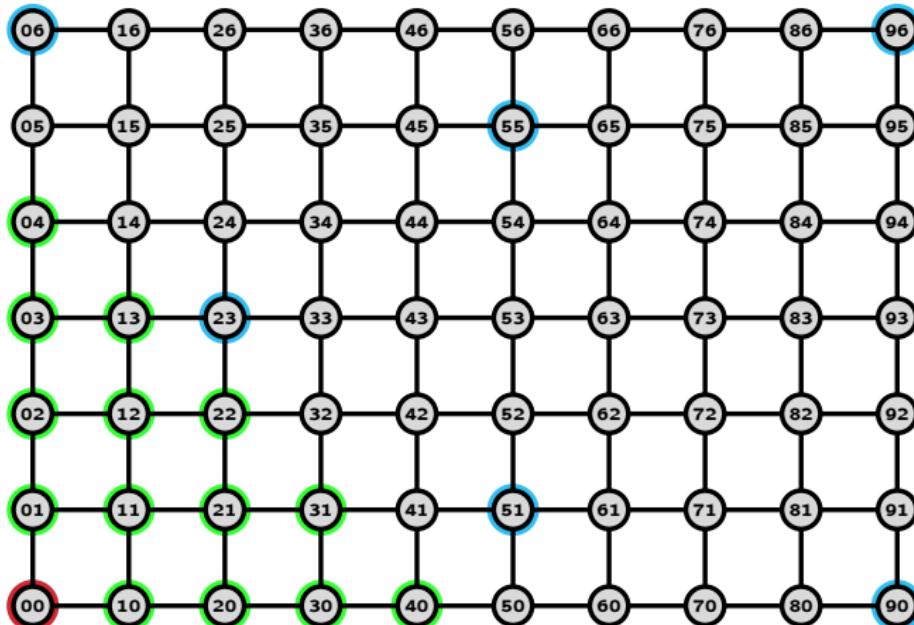
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



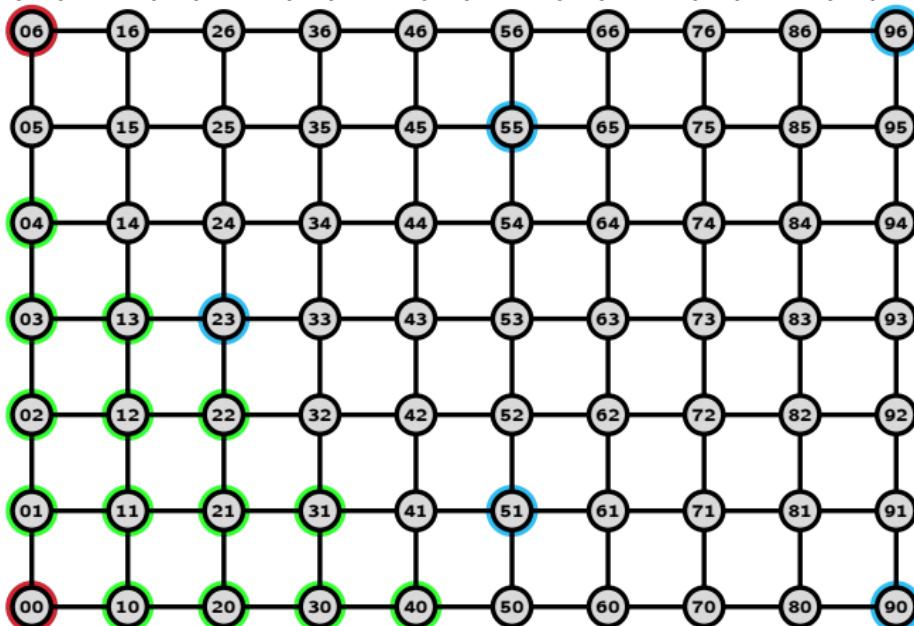
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



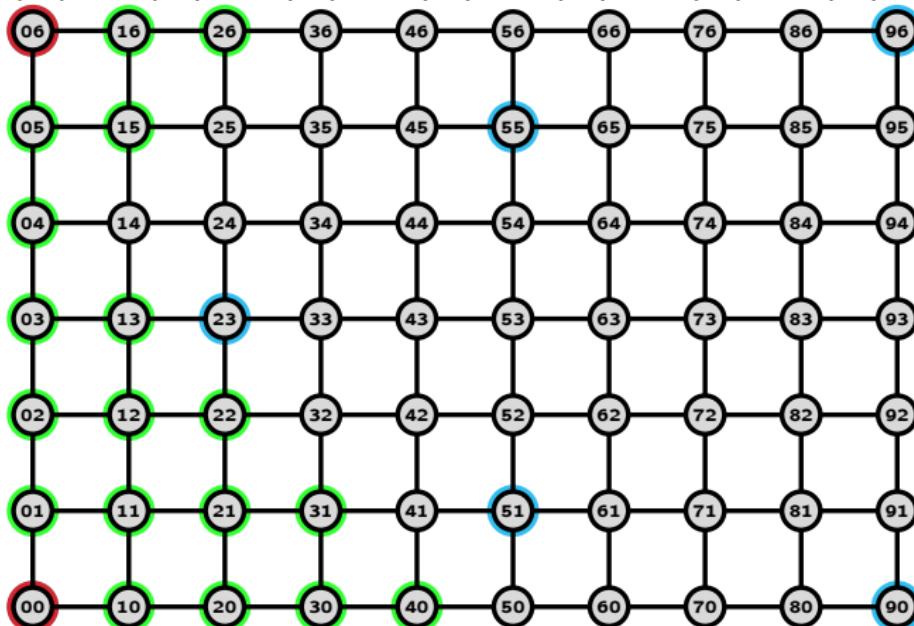
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



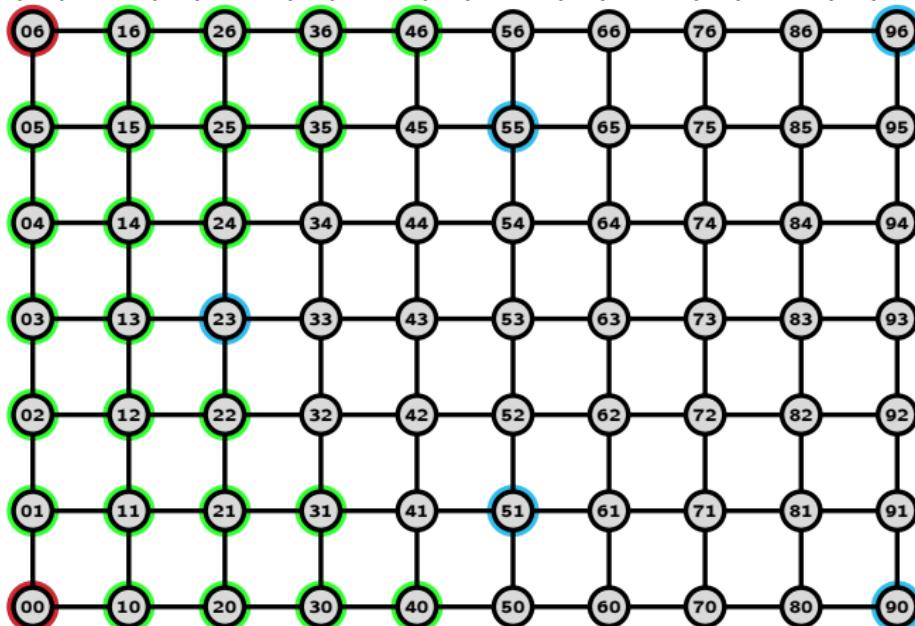
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



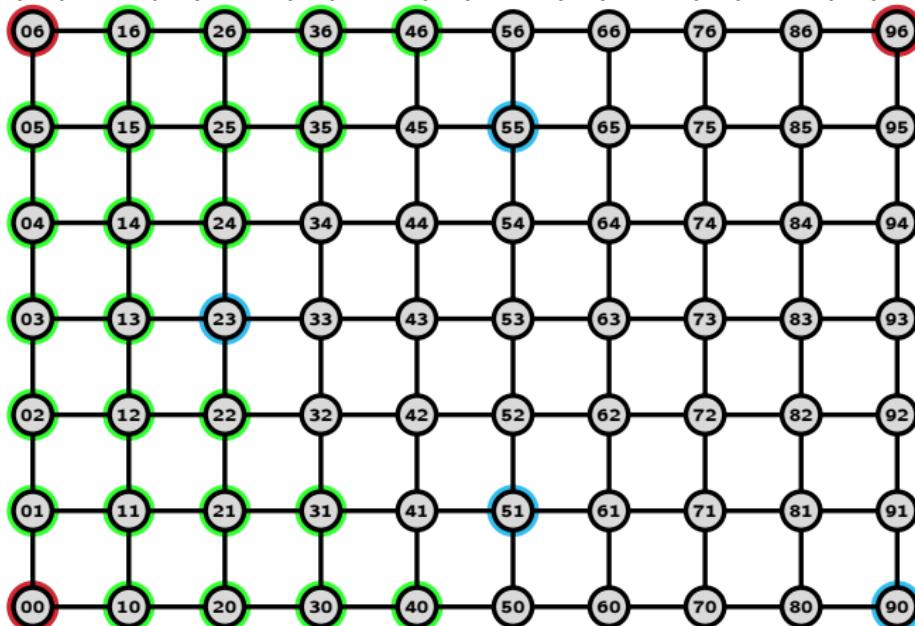
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



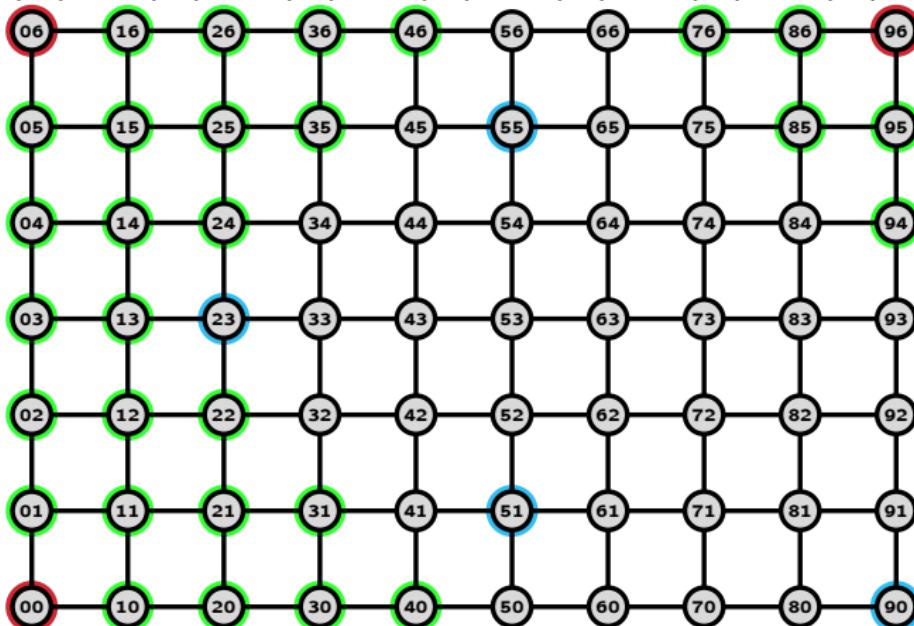
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



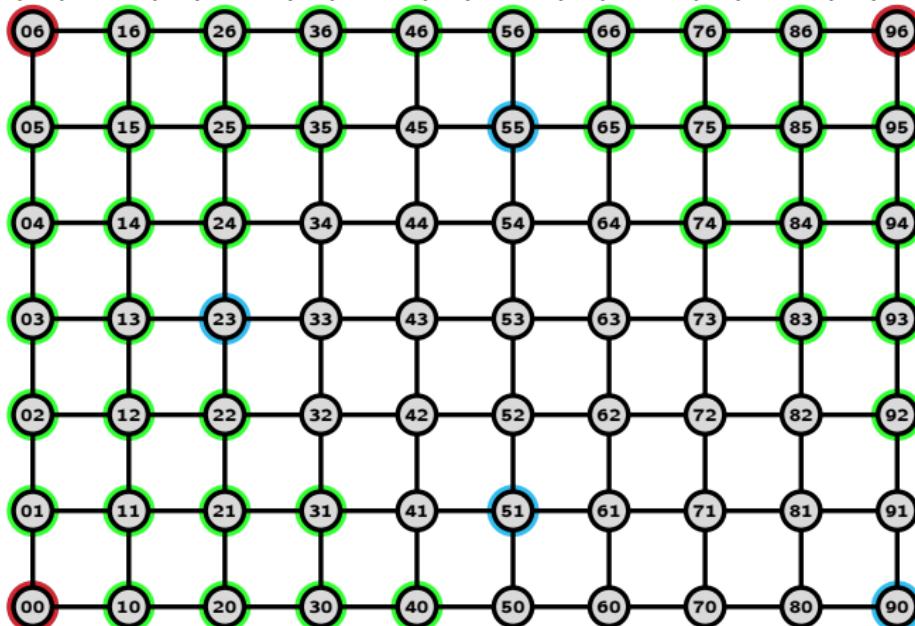
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



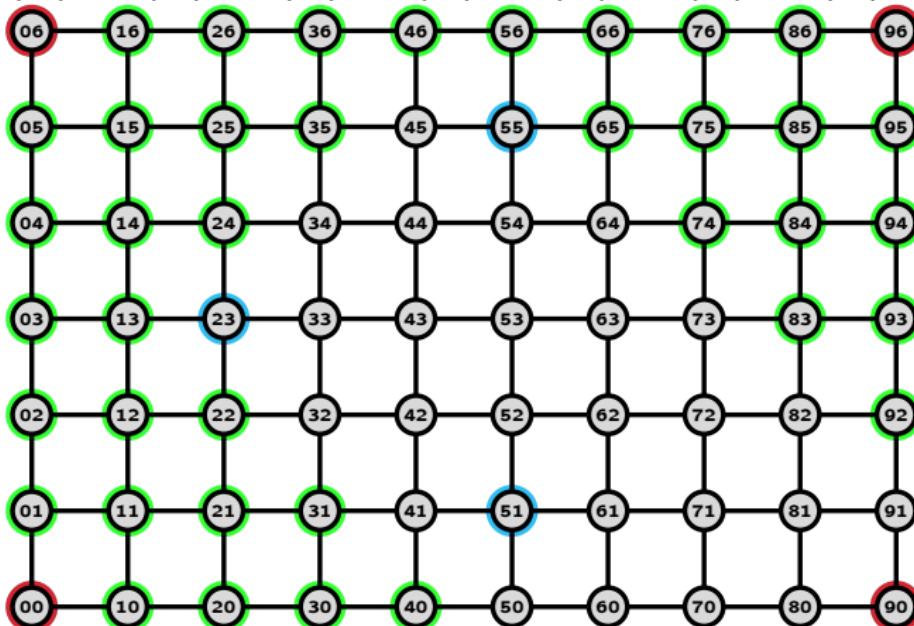
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



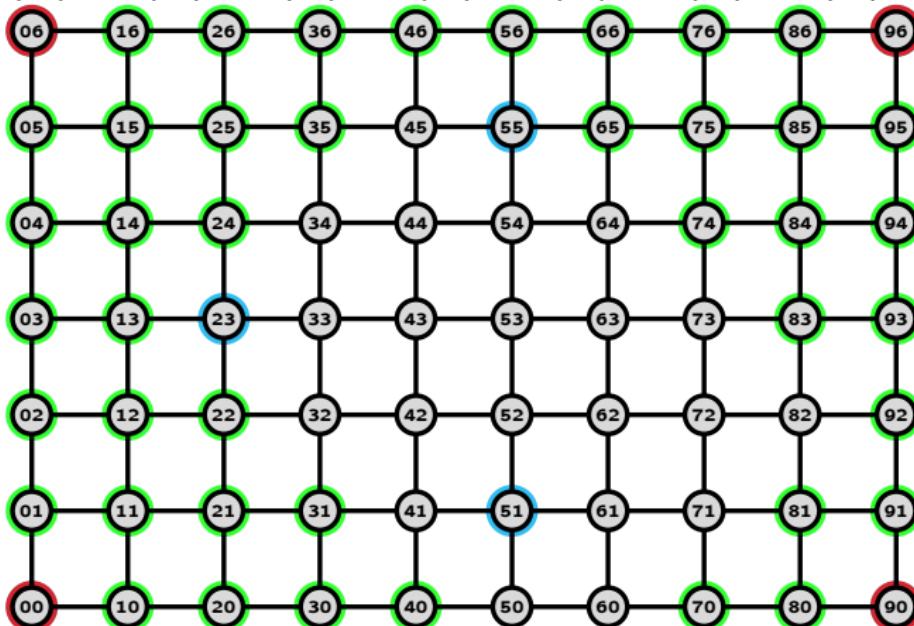
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



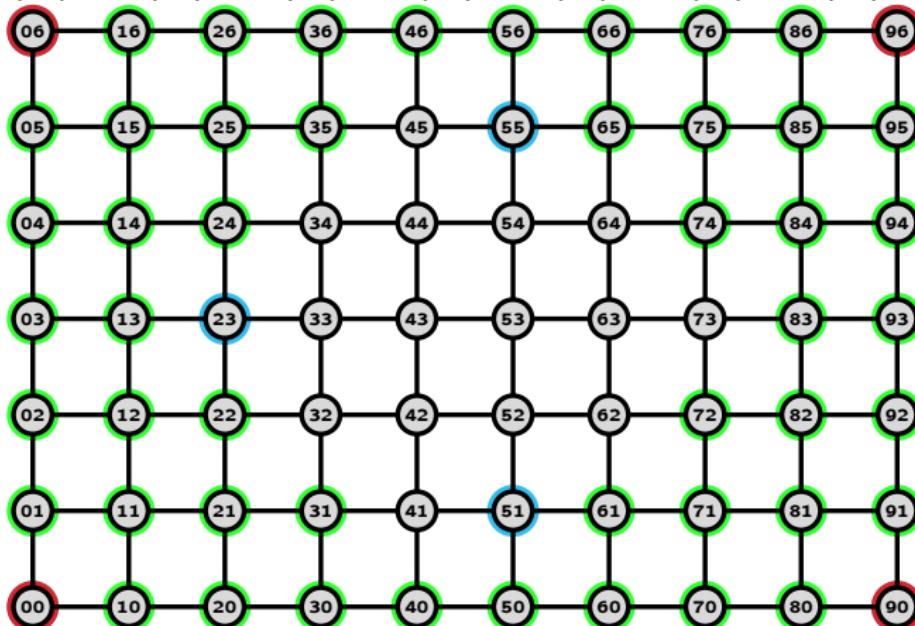
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



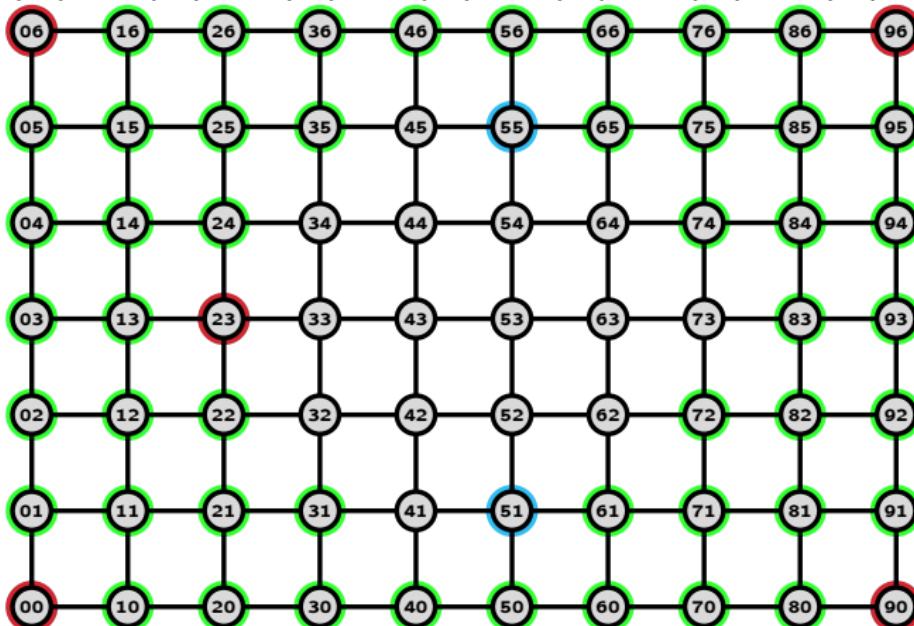
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



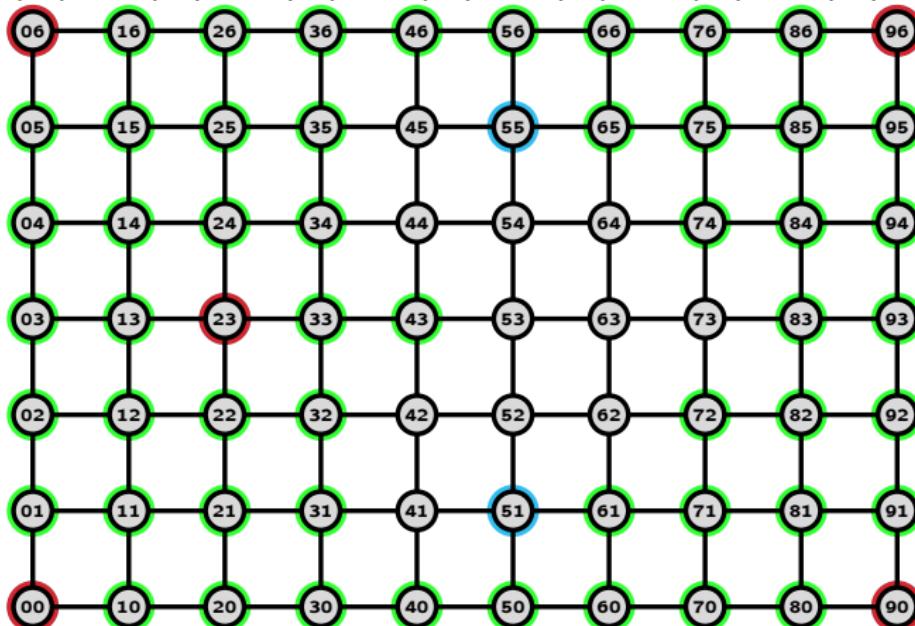
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



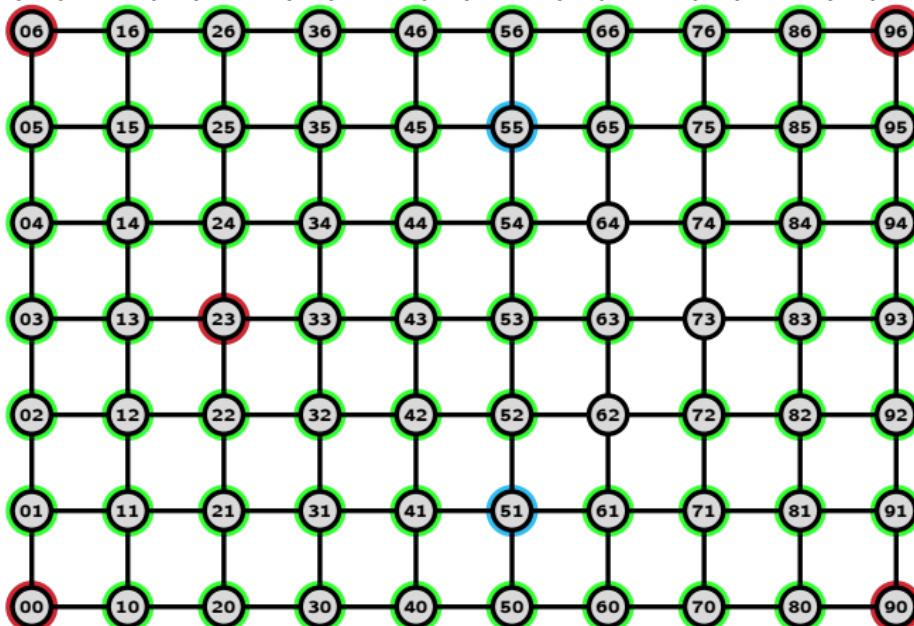
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



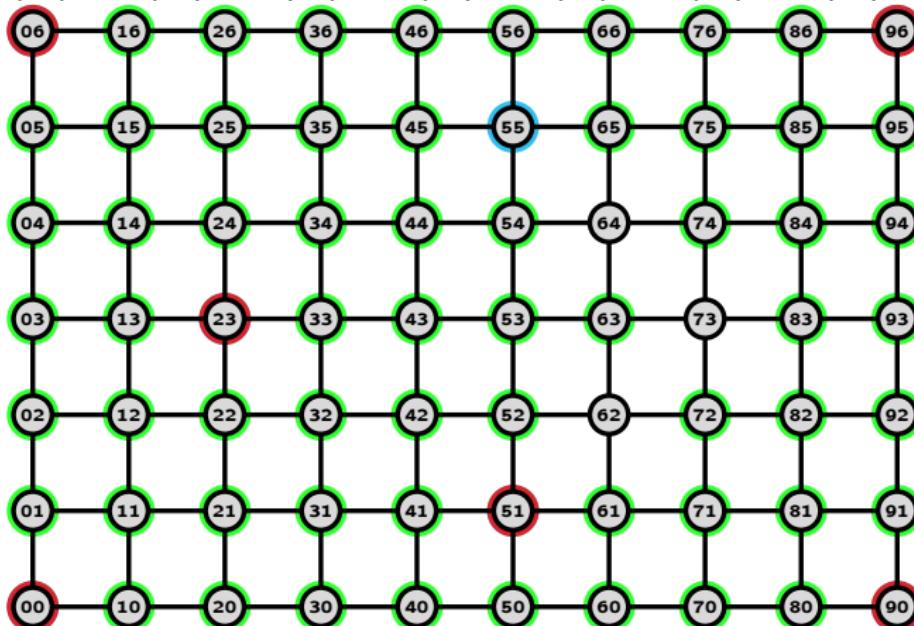
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



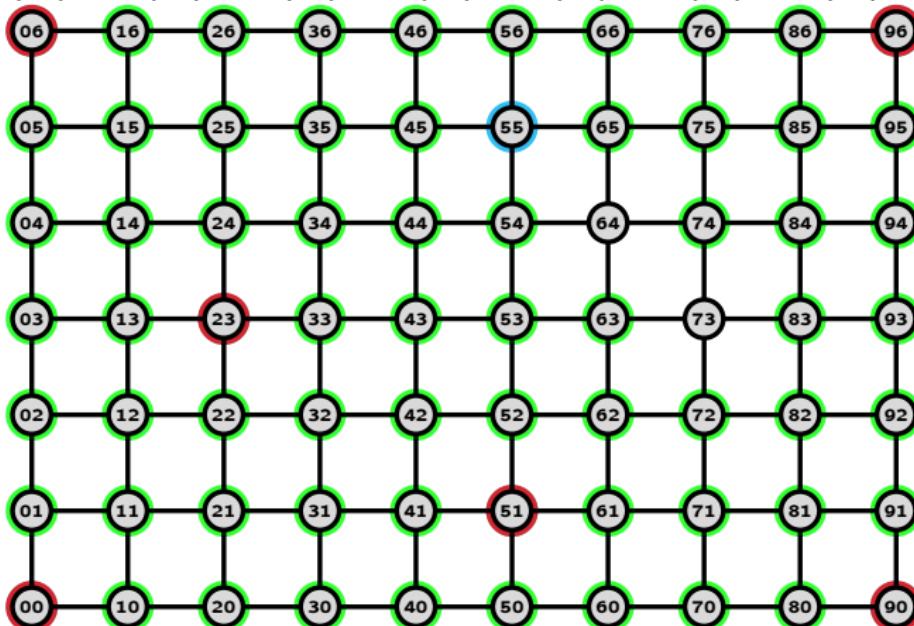
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



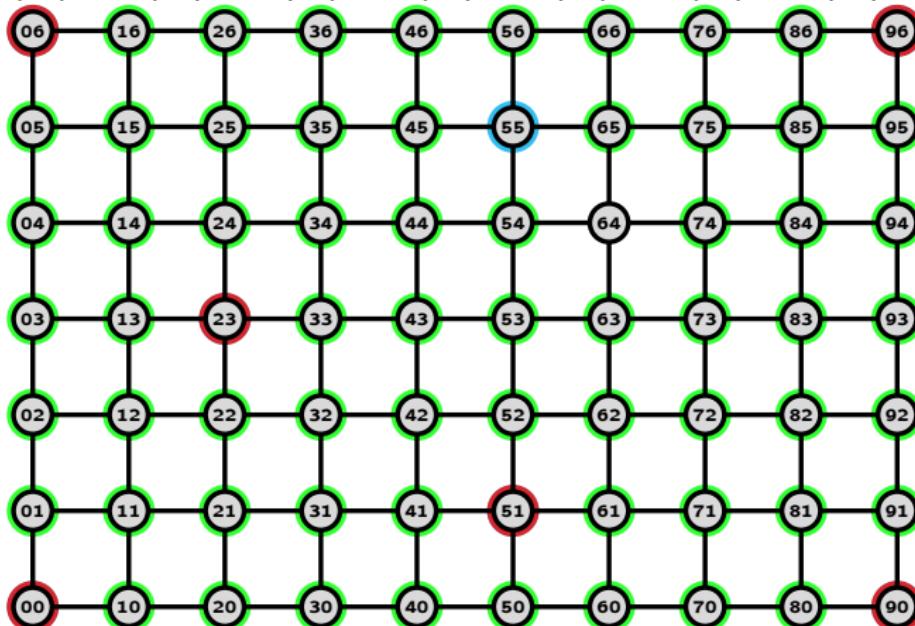
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



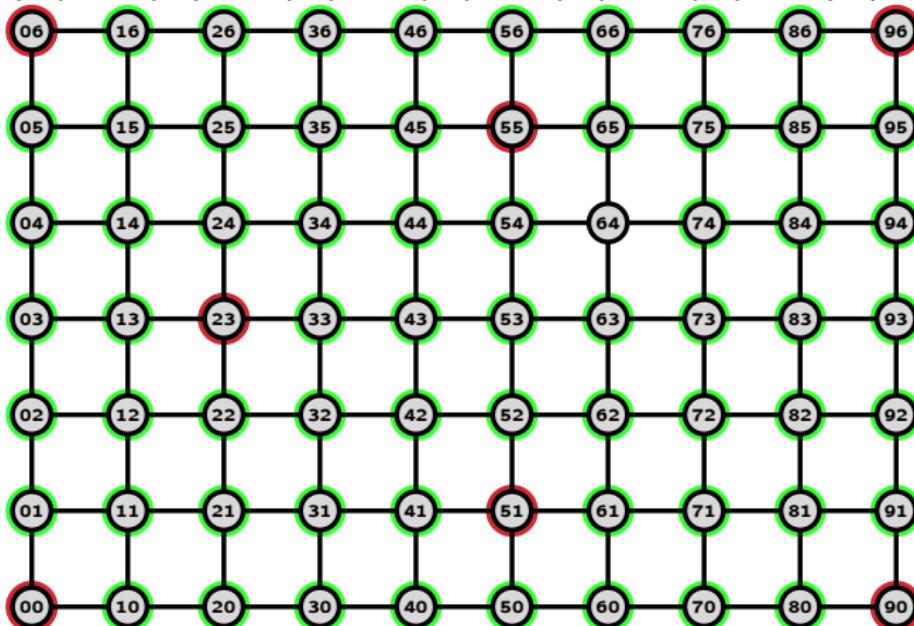
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



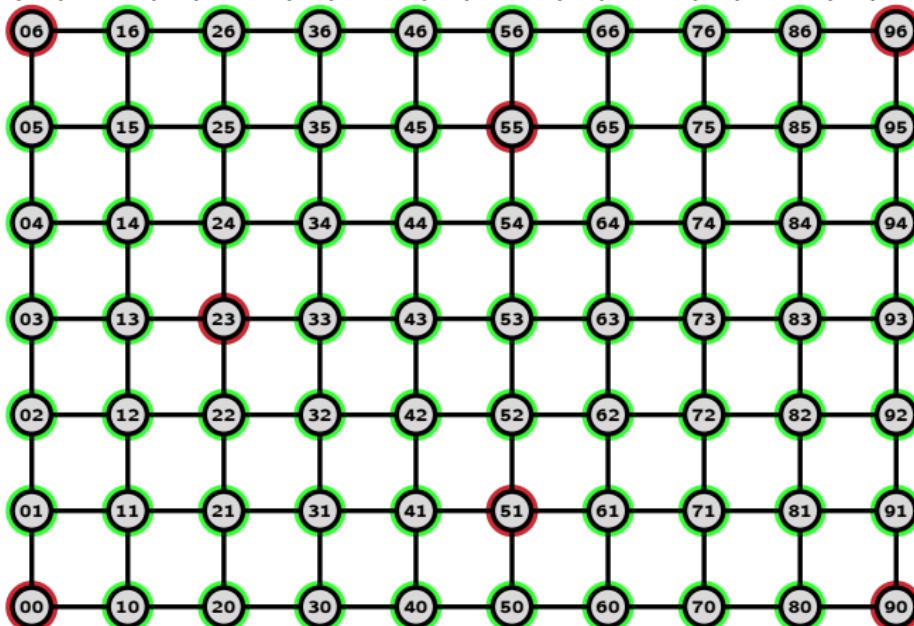
Bildhaftes Beispiel ($R = 2$)

$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



Bildhaftes Beispiel ($R = 2$)

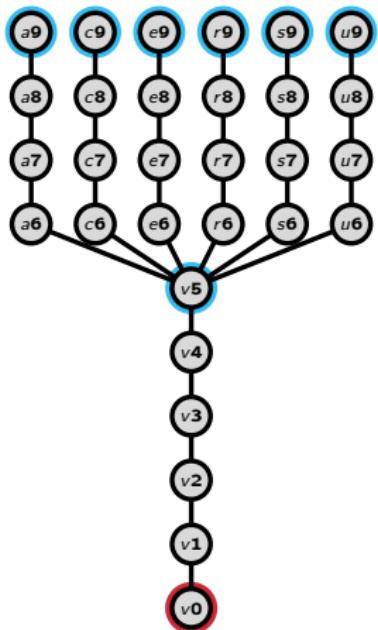
$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



Beispiel für $f(R)$ ist nicht monoton

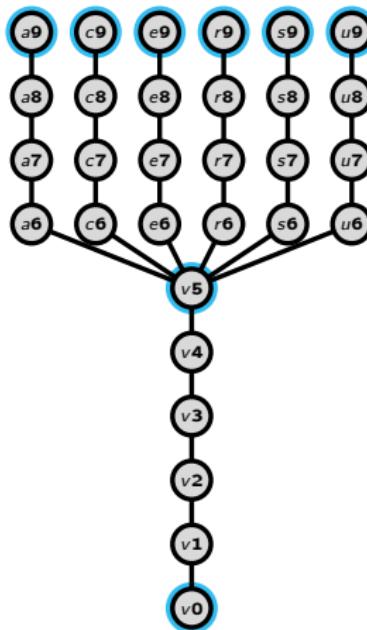
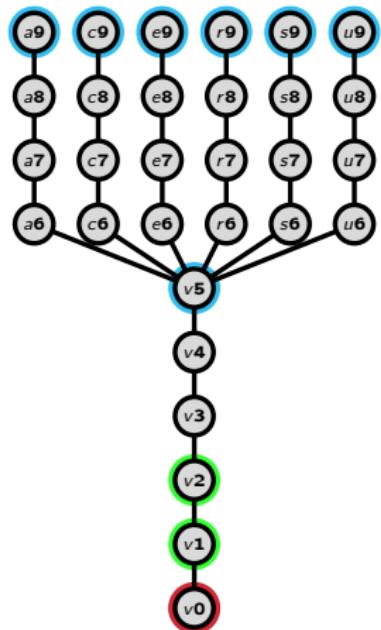
Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) =$



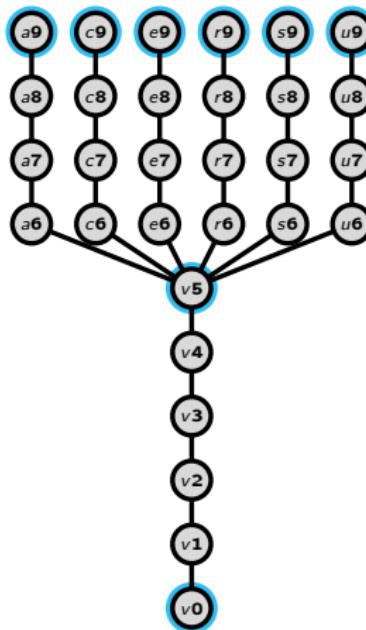
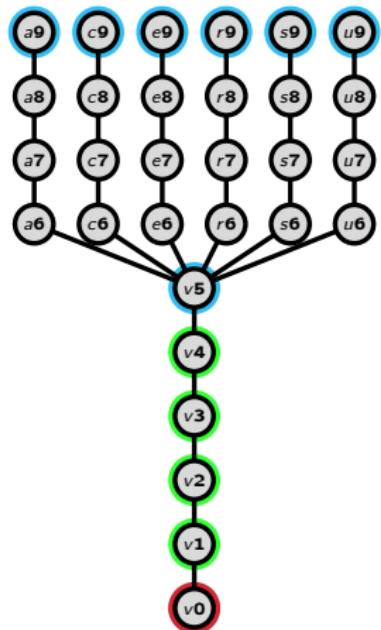
Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) =$



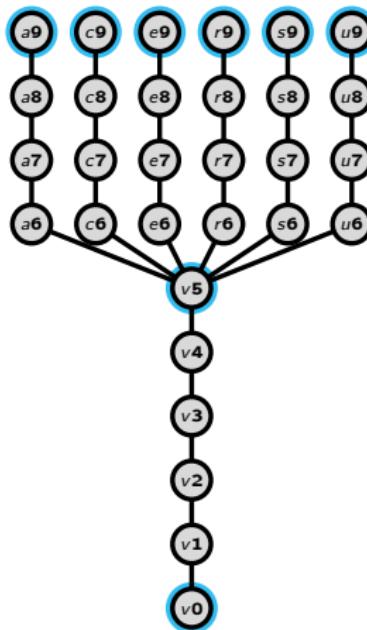
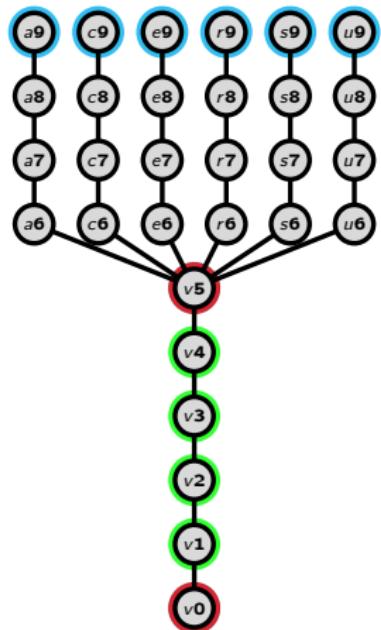
Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) =$



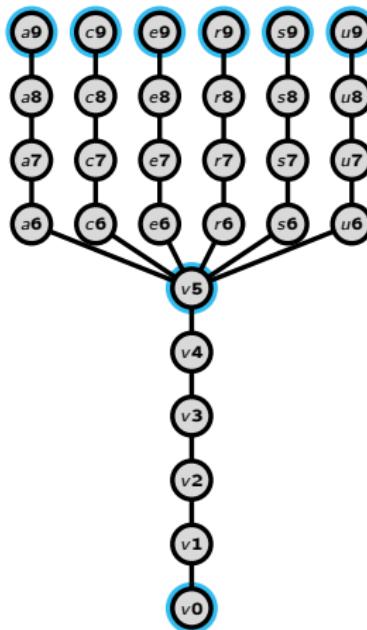
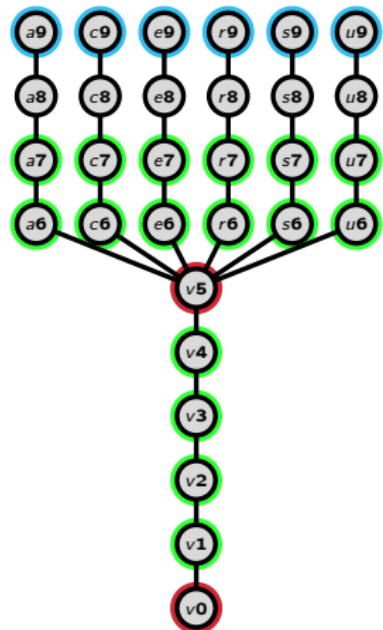
Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) =$



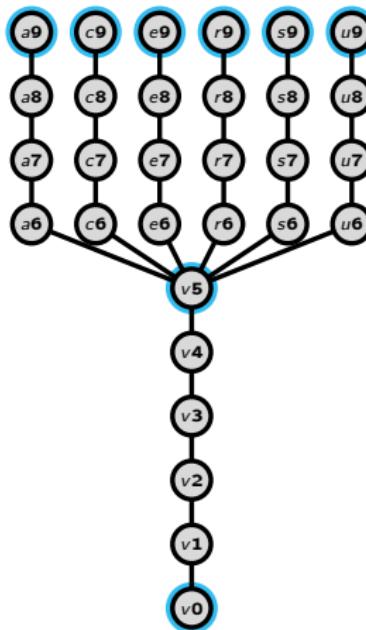
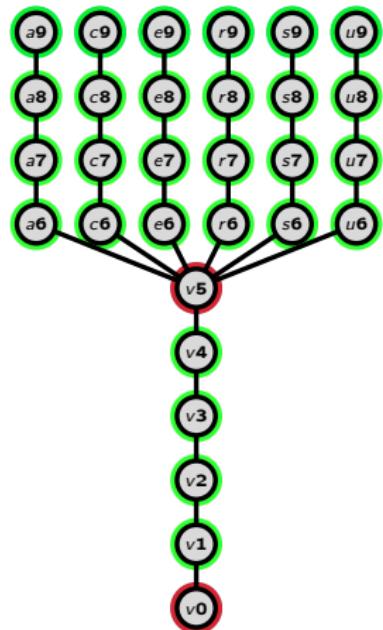
Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) =$



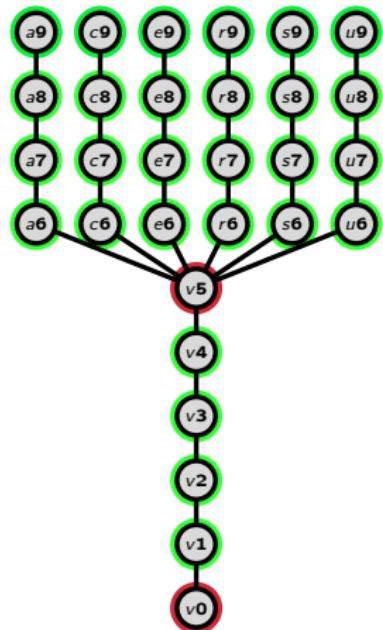
Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) =$

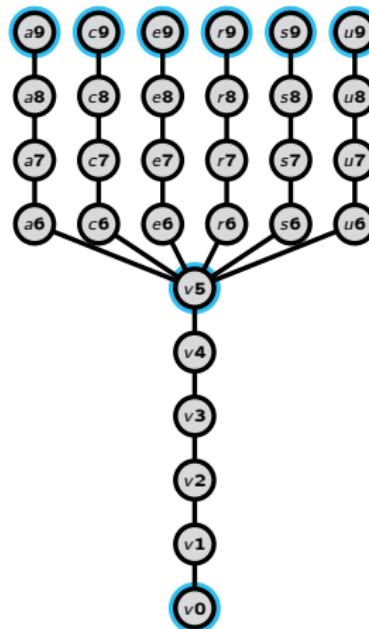


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

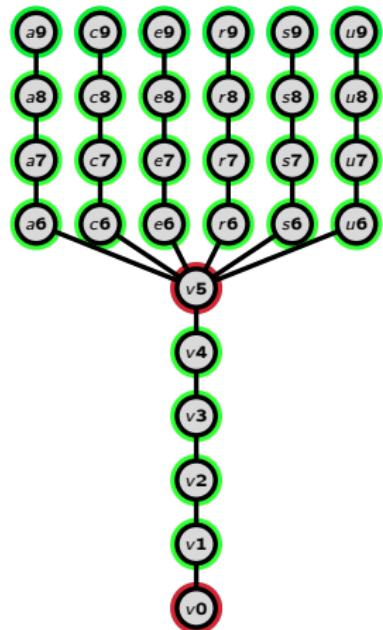


Bestimme $f(3) =$

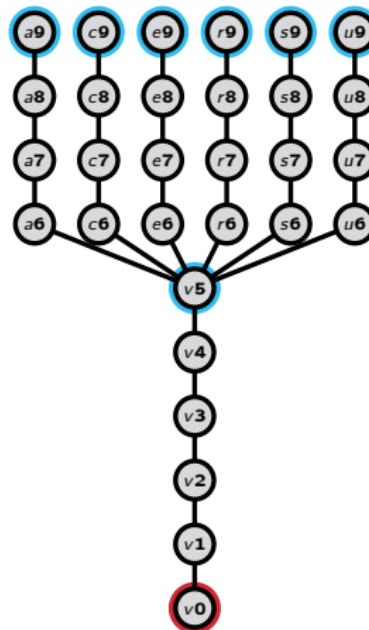


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

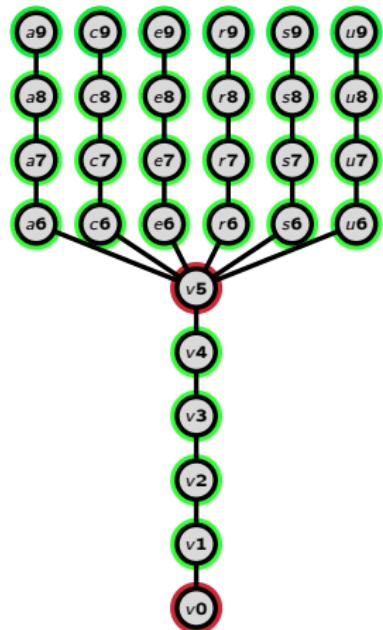


Bestimme $f(3) =$

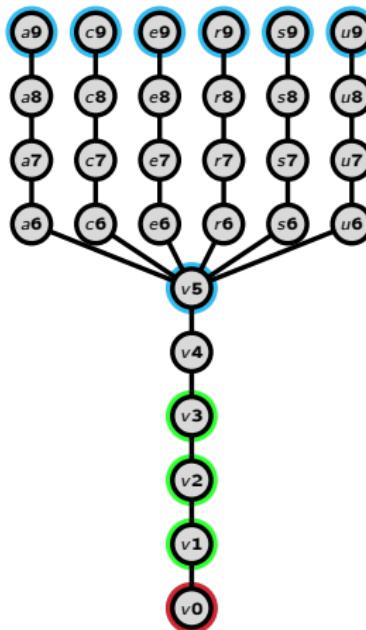


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

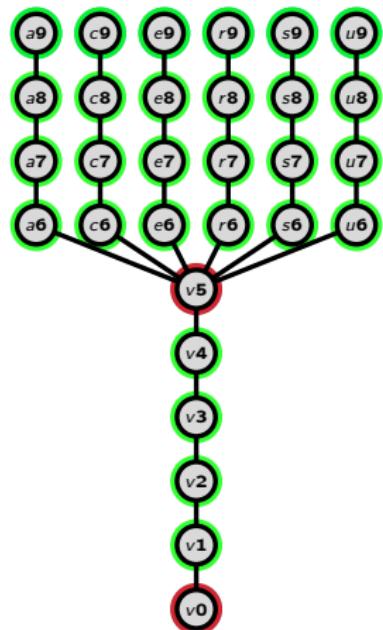


Bestimme $f(3) =$

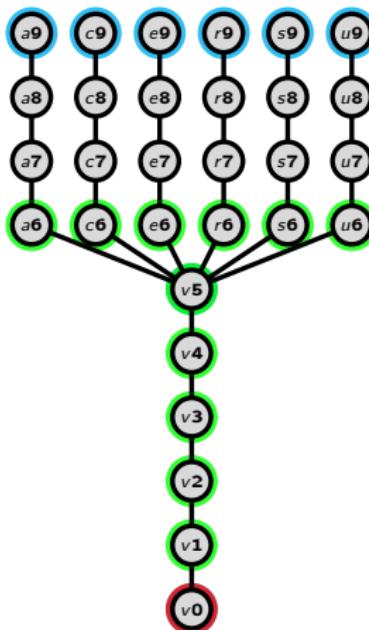


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

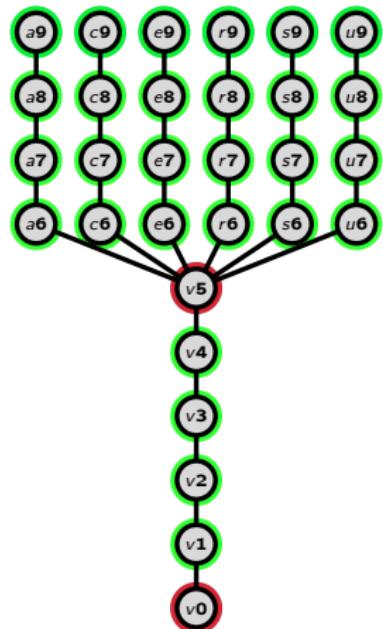


Bestimme $f(3) =$

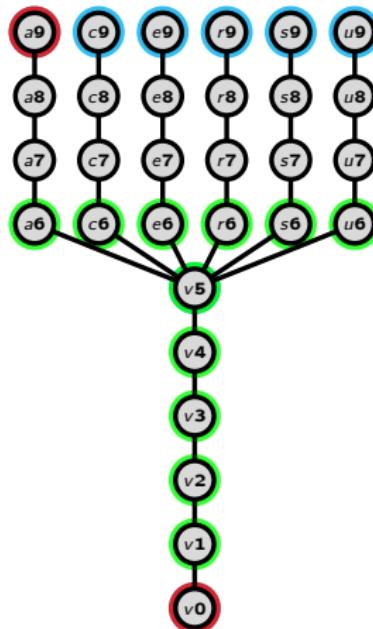


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

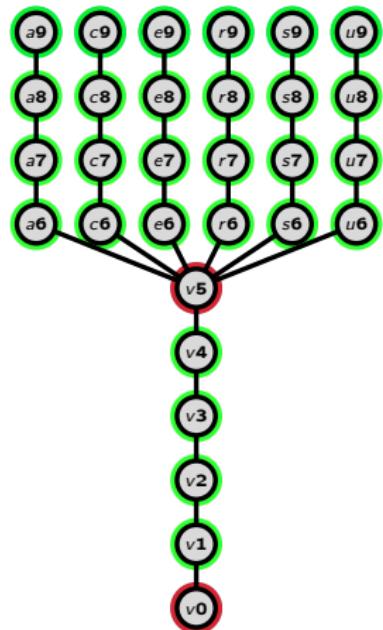


Bestimme $f(3) =$

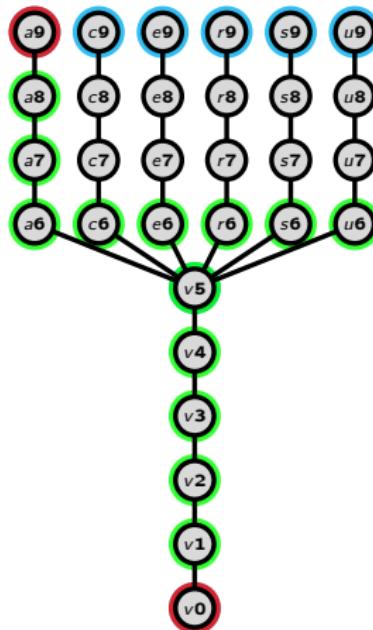


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

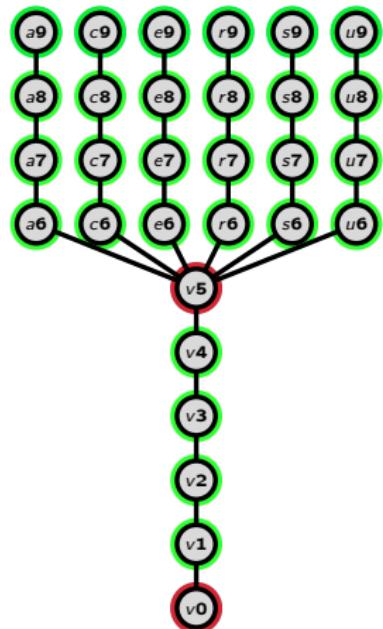


Bestimme $f(3) =$

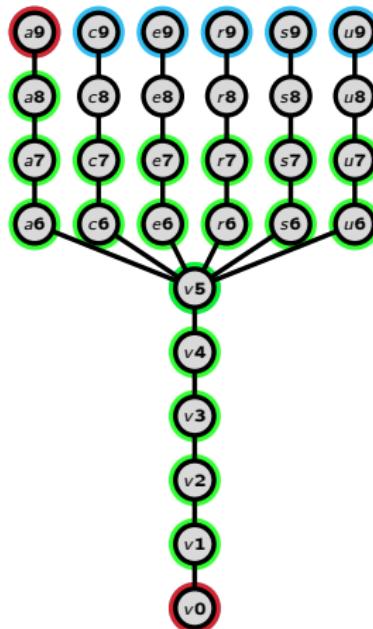


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

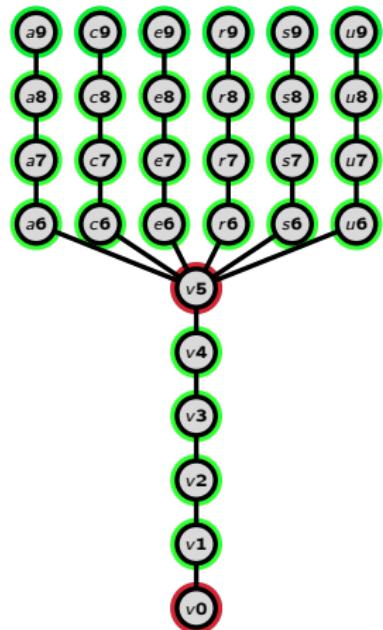


Bestimme $f(3) =$

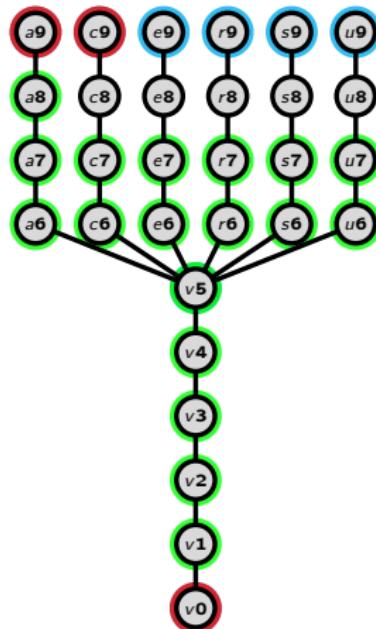


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

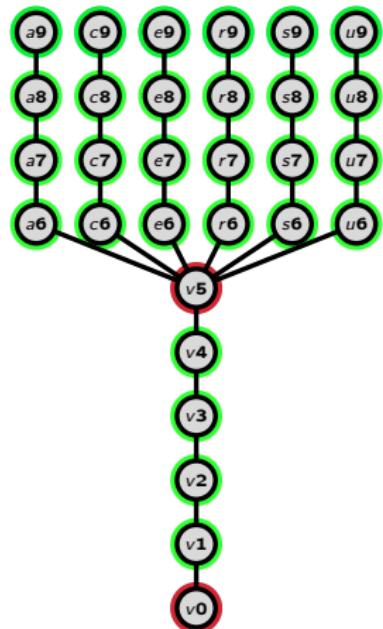


Bestimme $f(3) =$

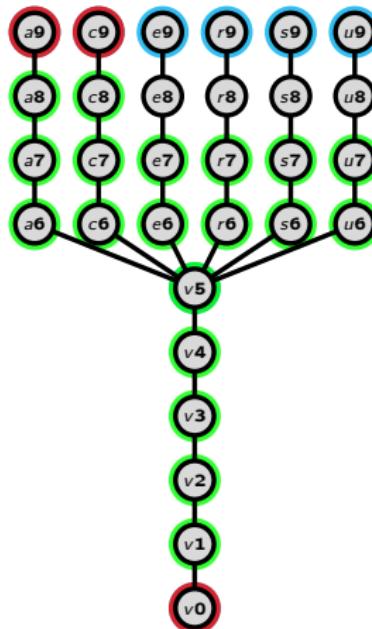


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

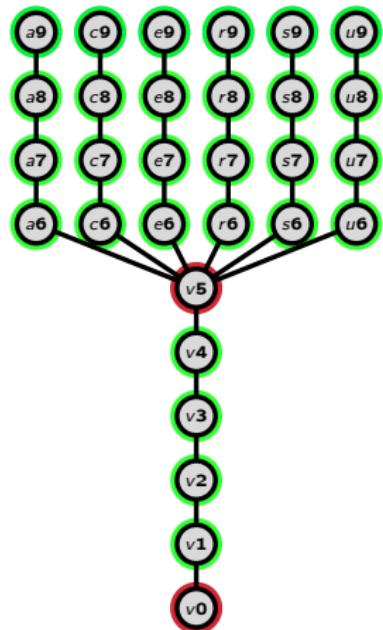


Bestimme $f(3) =$

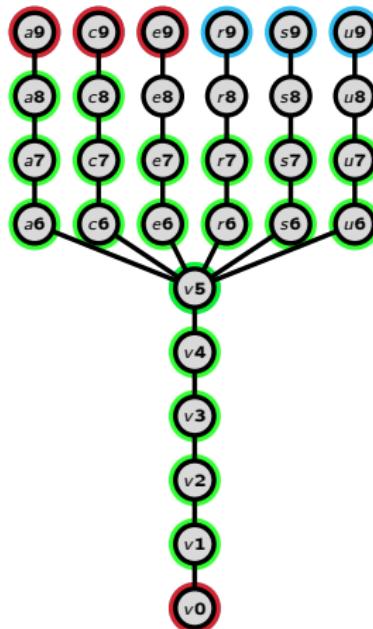


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

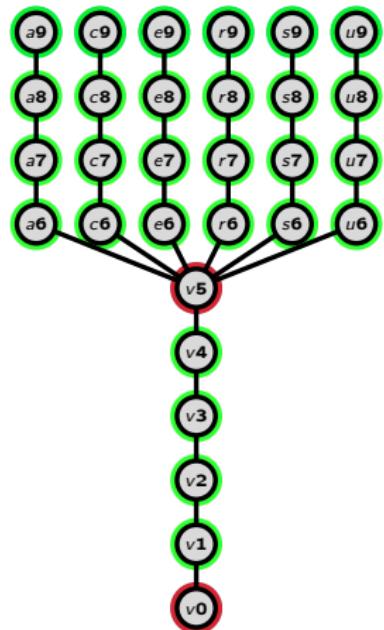


Bestimme $f(3) =$

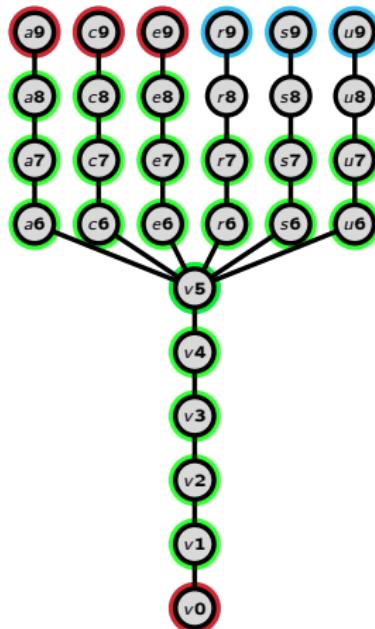


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

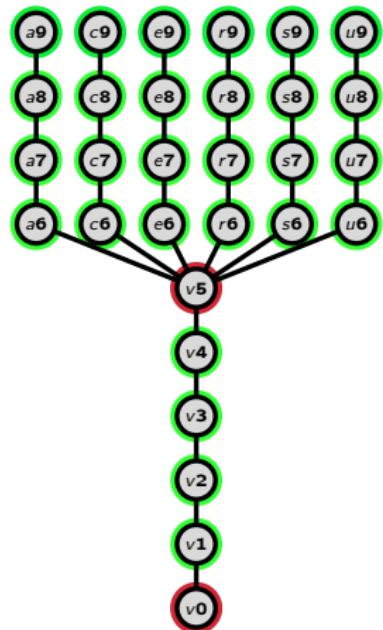


Bestimme $f(3) =$

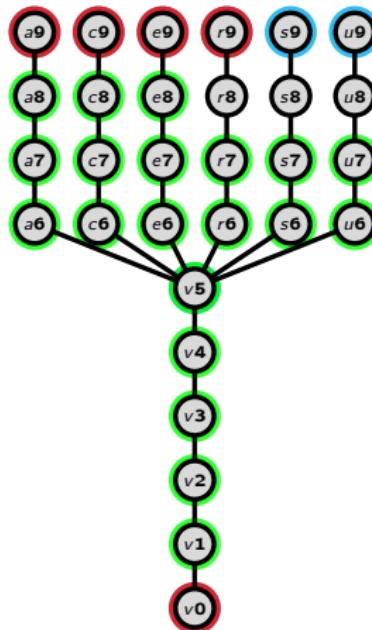


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

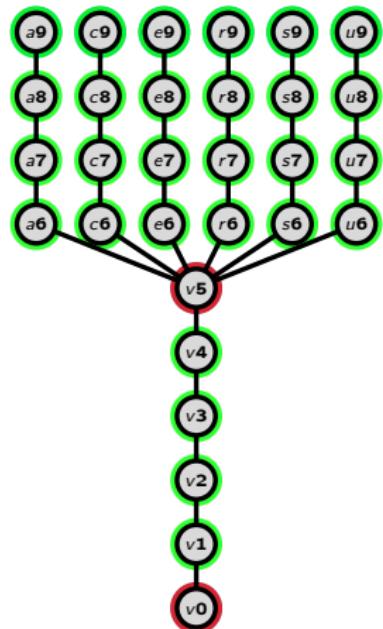


Bestimme $f(3) =$

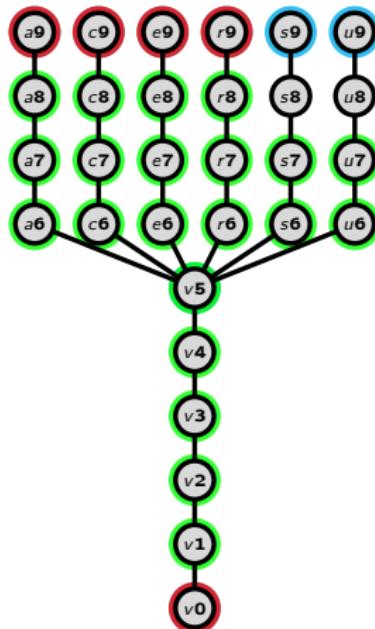


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

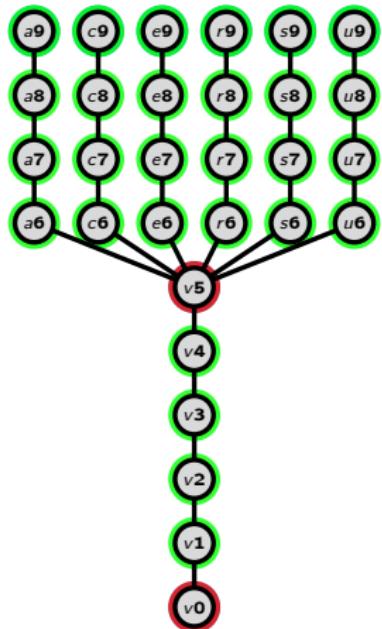


Bestimme $f(3) =$

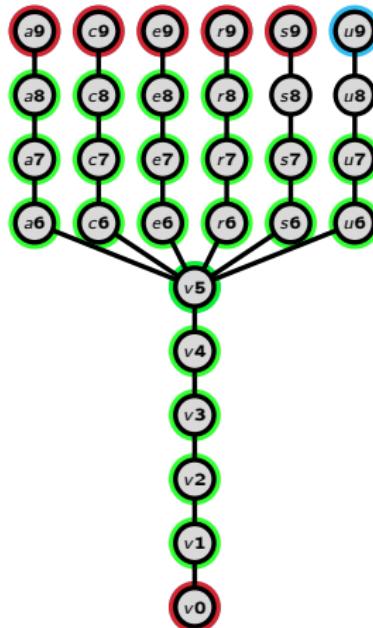


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

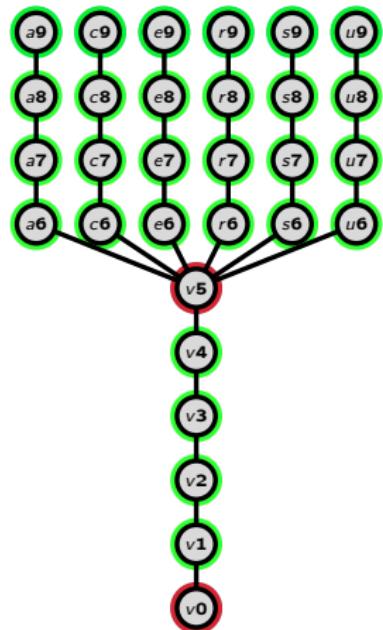


Bestimme $f(3) =$

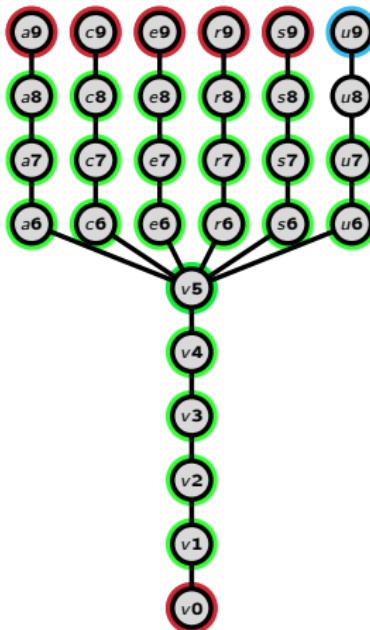


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

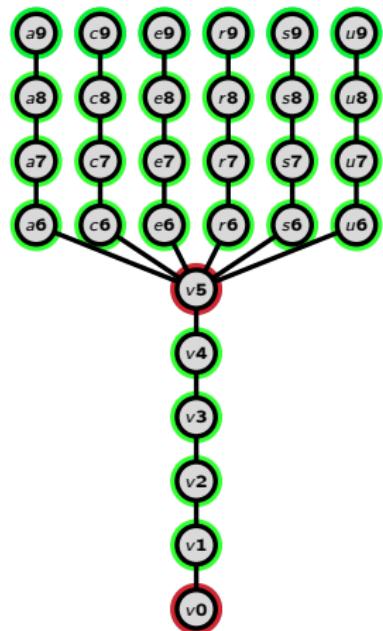


Bestimme $f(3) =$

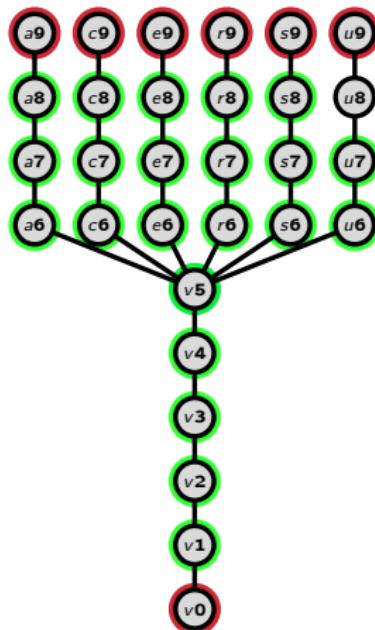


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

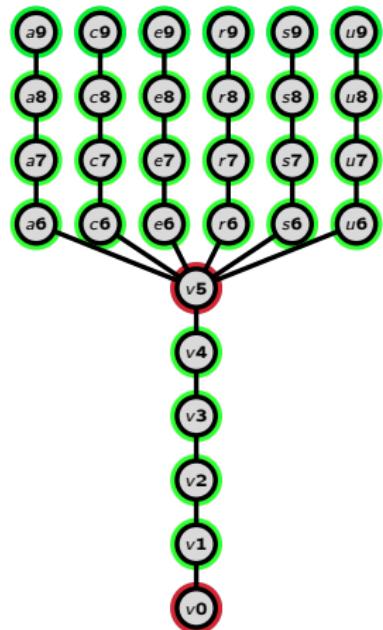


Bestimme $f(3) =$

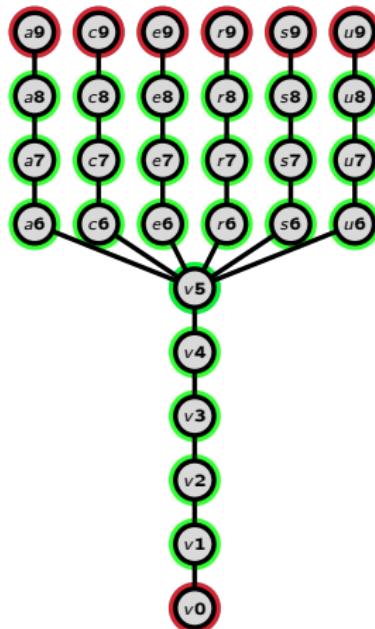


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

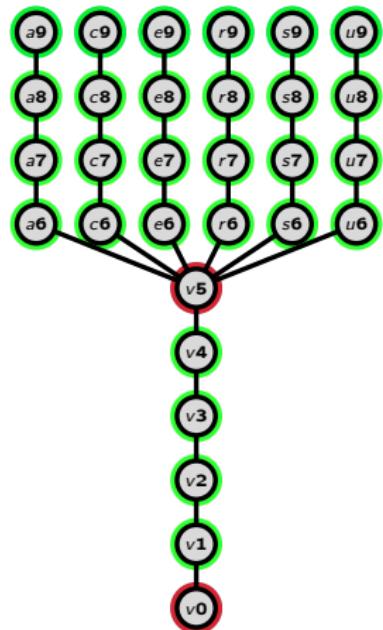


Bestimme $f(3) =$

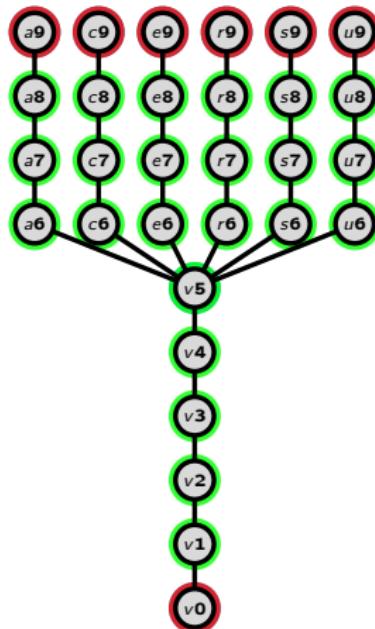


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$



Bestimme $f(3) = 7$



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

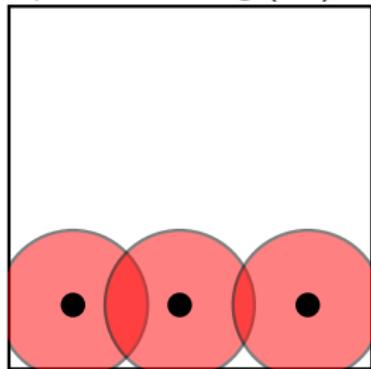
Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



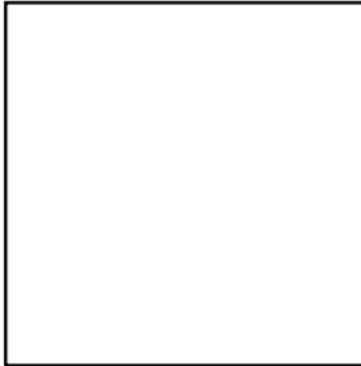
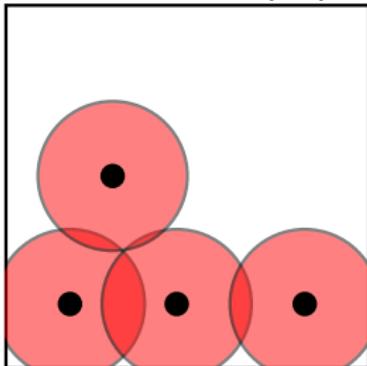
Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



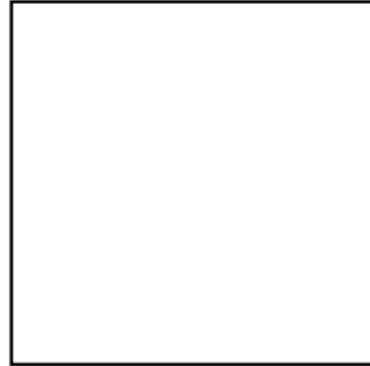
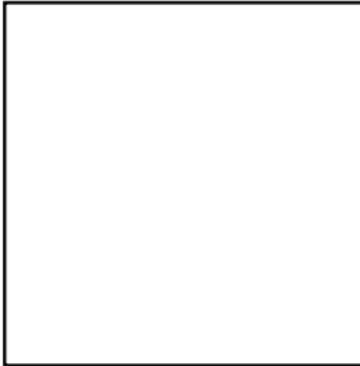
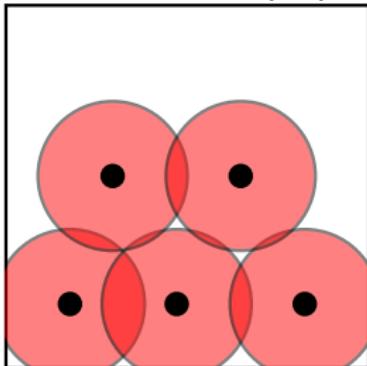
Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



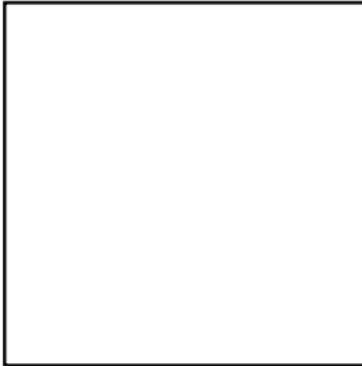
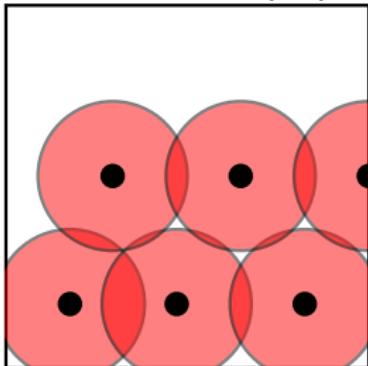
Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



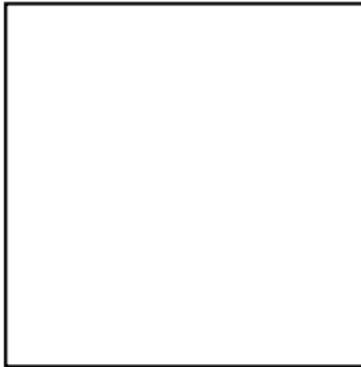
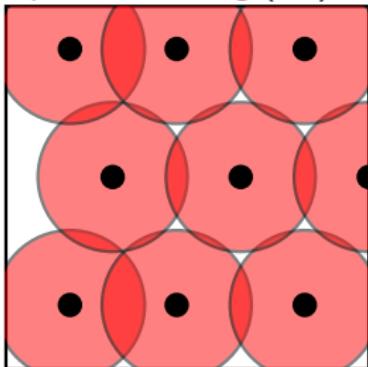
Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



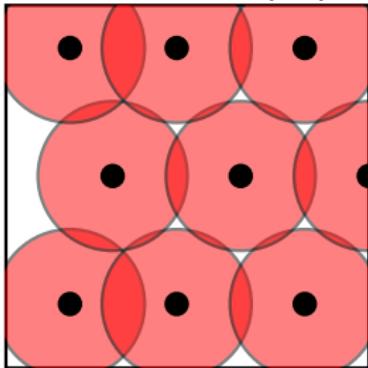
Approximation

Lemma

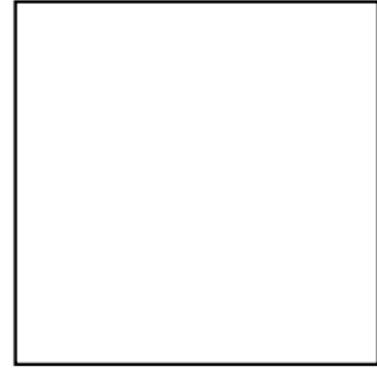
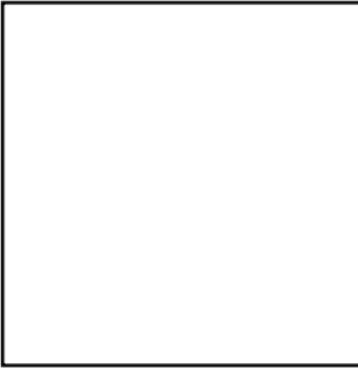
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



Approximation:



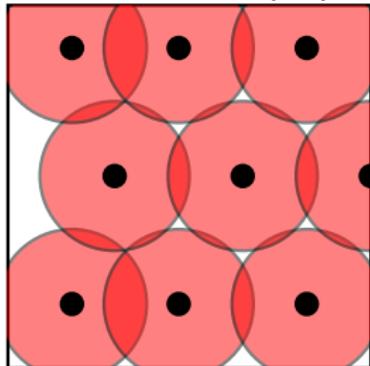
Approximation

Lemma

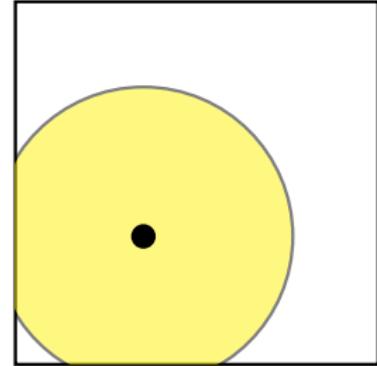
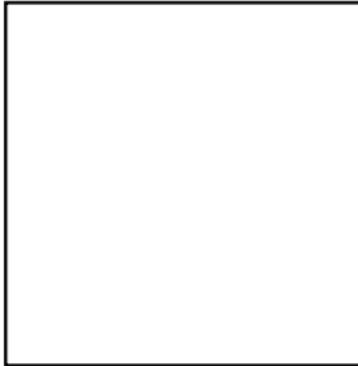
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



Approximation:



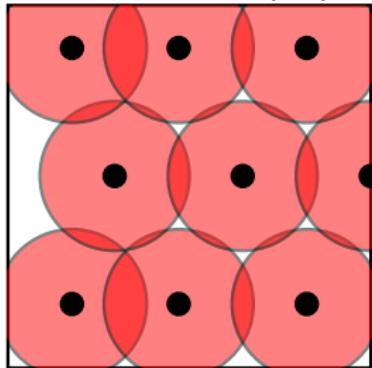
Approximation

Lemma

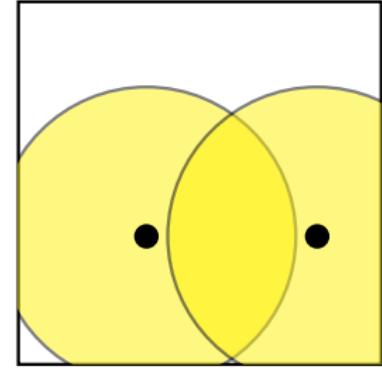
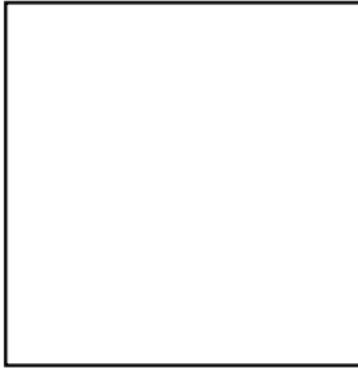
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



Approximation:



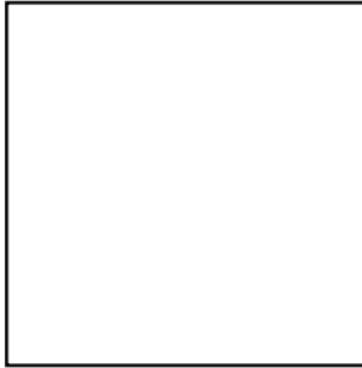
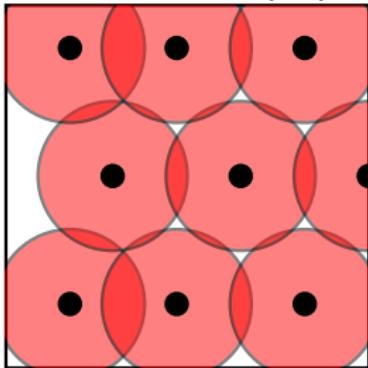
Approximation

Lemma

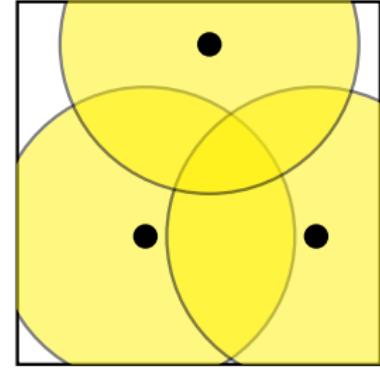
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

Optimale Lösung (R^*):



Approximation:



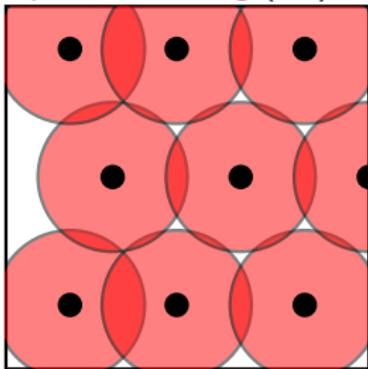
Approximation

Lemma

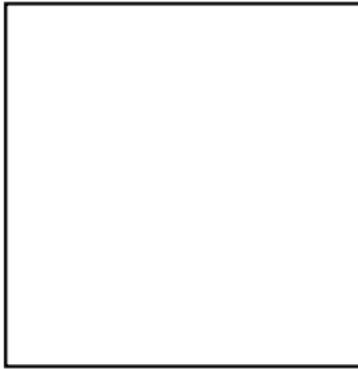
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

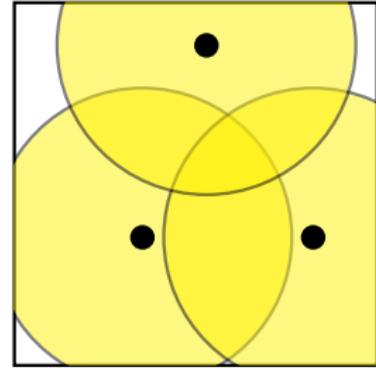
Optimale Lösung (R^*):



Vergleich:



Approximation:



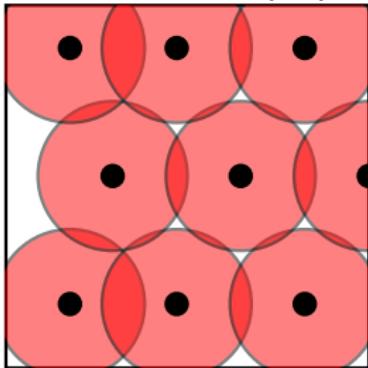
Approximation

Lemma

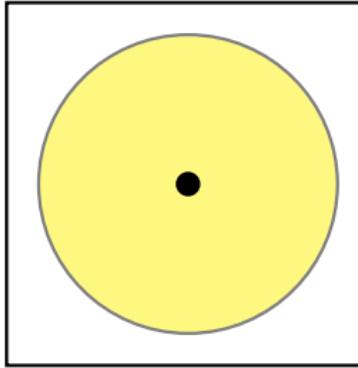
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

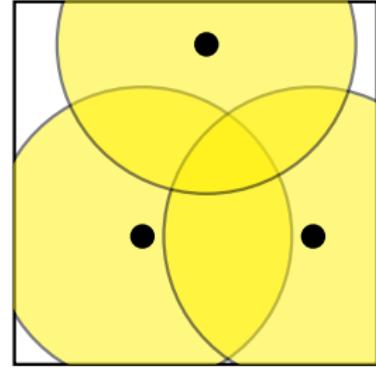
Optimale Lösung (R^*):



Vergleich:



Approximation:



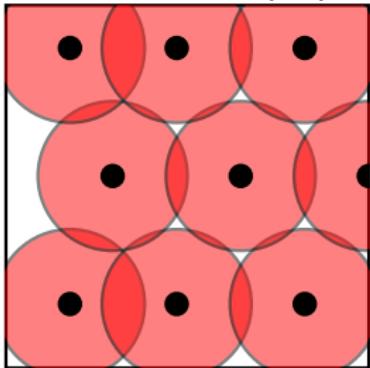
Approximation

Lemma

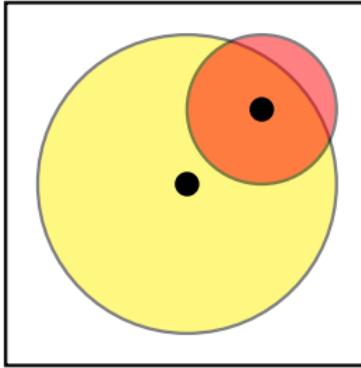
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

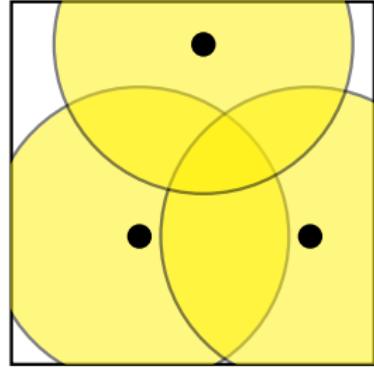
Optimale Lösung (R^*):



Vergleich:



Approximation:

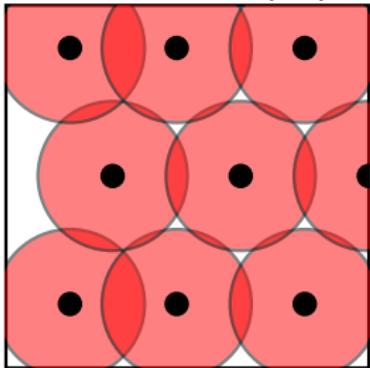


Approximation

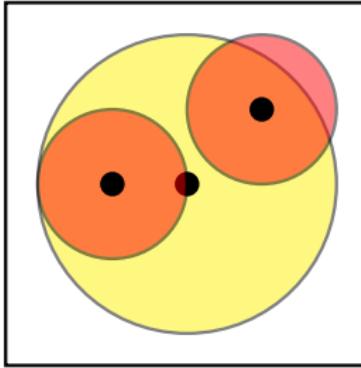
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

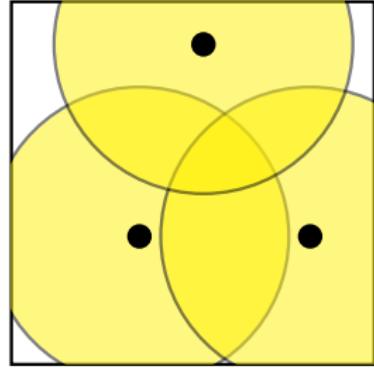
Optimale Lösung (R^*):



Vergleich:



Approximation:



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.
- Damit $\cup_{i=1}^k V_i = V$.



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.
- Damit $\cup_{i=1}^k V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i . Damit gilt:

$$w(v) \cdot \text{dist}(v, z) \leq w(v) \cdot (\text{dist}(v, v_i) + \text{dist}(v_i, z))$$



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.
- Damit $\cup_{i=1}^k V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i . Damit gilt:

$$w(v) \cdot \text{dist}(v, z) \leq w(v) \cdot (\text{dist}(v, v_i) + \text{dist}(v_i, z))$$



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.
- Damit $\cup_{i=1}^k V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i . Damit gilt:

$$\begin{aligned} w(v) \cdot \text{dist}(v, z) &\leq w(v) \cdot (\text{dist}(v, v_i) + \text{dist}(v_i, z)) \\ &\leq w(v) \cdot \text{dist}(v, v_i) + w(z) \cdot \text{dist}(v_i, z) \end{aligned}$$



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.
- Damit $\cup_{i=1}^k V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i . Damit gilt:

$$\begin{aligned} w(v) \cdot \text{dist}(v, z) &\leq w(v) \cdot (\text{dist}(v, v_i) + \text{dist}(v_i, z)) \\ &\leq w(v) \cdot \text{dist}(v, v_i) + w(z) \cdot \text{dist}(v_i, z) \\ &\leq 2 \cdot R^* \end{aligned}$$



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.
- Damit $\cup_{i=1}^k V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i . Damit gilt:

$$\begin{aligned} w(v) \cdot \text{dist}(v, z) &\leq w(v) \cdot (\text{dist}(v, v_i) + \text{dist}(v_i, z)) \\ &\leq w(v) \cdot \text{dist}(v, v_i) + w(z) \cdot \text{dist}(v_i, z) \\ &\leq 2 \cdot R^* \\ &\leq 2 \cdot R \end{aligned}$$



- Alle Knoten aus V_i sind am Ende des Schleifendurchlaufes aus U gelöscht.

Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.

Dann gilt: $f(R) \leq k$ für alle $R \geq R^*$.

- Sei $Z^* = \{z_1, z_2, \dots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot \text{dist}(v, z_i) \leq R^*\}$.
- Damit $\cup_{i=1}^k V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i . Damit gilt:

$$\begin{aligned} w(v) \cdot \text{dist}(v, z) &\leq w(v) \cdot (\text{dist}(v, v_i) + \text{dist}(v_i, z)) \\ &\leq w(v) \cdot \text{dist}(v, v_i) + w(z) \cdot \text{dist}(v_i, z) \\ &\leq 2 \cdot R^* \\ &\leq 2 \cdot R \end{aligned}$$



- Alle Knoten aus V_i sind am Ende des Schleifendurchlaufes aus U gelöscht.
- Schleife terminiert nach höchstens $k = |Z^*|$ Runden. Und es gilt: $|Z| \leq k$.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .
 - ② Sortiere Werte $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .
 - ② Sortiere Werte $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$.
 - ③ Seien $R_1 < R_2 < \dots < R_{anz}$ diese Werte.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .
 - ② Sortiere Werte $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$.
 - ③ Seien $R_1 < R_2 < \dots < R_{anz}$ diese Werte.
 - ④ $i := 0, Z = V$.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .
 - ② Sortiere Werte $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$.
 - ③ Seien $R_1 < R_2 < \dots < R_{anz}$ diese Werte.
 - ④ $i := 0, Z = V$.
 - ⑤ Solange $|Z| > k$ führe aus:

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .
 - ② Sortiere Werte $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$.
 - ③ Seien $R_1 < R_2 < \dots < R_{anz}$ diese Werte.
 - ④ $i := 0, Z = V$.
 - ⑤ Solange $|Z| > k$ führe aus:
 - ① $i = i + 1$.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .
 - ② Sortiere Werte $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$.
 - ③ Seien $R_1 < R_2 < \dots < R_{anz}$ diese Werte.
 - ④ $i := 0, Z = V$.
 - ⑤ Solange $|Z| > k$ führe aus:
 - ① $i = i + 1$.
 - ② $Z := GreedyZentrum(G, c, w, R_i)$.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
 - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
 - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leq n \cdot (n - 1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
 - ① Bestimme Distanzmatrix $(dist(u, v))_{u, v \in V}$ für (G, c) .
 - ② Sortiere Werte $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$.
 - ③ Seien $R_1 < R_2 < \dots < R_{anz}$ diese Werte.
 - ④ $i := 0, Z = V$.
 - ⑤ Solange $|Z| > k$ führe aus:
 - ① $i := i + 1$.
 - ② $Z := GreedyZentrum(G, c, w, R_i)$.
 - ⑥ Ausgabe: Z .

Approximation

Theorem

Das Zentrumsproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

Approximation

Theorem

Das Zentrumssproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.

Approximation

Theorem

Das Zentrumsproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.

Approximation

Theorem

Das Zentrumssproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$

Approximation

Theorem

Das Zentrumproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Durch Konstrukton von GreedyZentrum gilt: $\text{rad}(Z) \leq 2 \cdot R_{i_0}$

Approximation

Theorem

Das Zentrumproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Durch Konstrukton von GreedyZentrum gilt: $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.

Approximation

Theorem

Das Zentrumproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Durch Konstruktion von GreedyZentrum gilt: $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:

Approximation

Theorem

Das Zentrumproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Durch Konstrukton von GreedyZentrum gilt: $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
 - Distanzmatrix: $O(n^3)$.

Approximation

Theorem

Das Zentrumproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Durch Konstrukton von GreedyZentrum gilt: $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
 - Distanzmatrix: $O(n^3)$.
 - Sortierung: $O(n^2 \log n)$.

Approximation

Theorem

Das Zentrumproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Durch Konstrukton von GreedyZentrum gilt: $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
 - Distanzmatrix: $O(n^3)$.
 - Sortierung: $O(n^2 \log n)$.
 - GreedyZentrum: $O(n^2)$.

Approximation

Theorem

Das Zentrumproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Durch Konstrukton von GreedyZentrum gilt: $\text{rad}(Z) \leq 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
 - Distanzmatrix: $O(n^3)$.
 - Sortierung: $O(n^2 \log n)$.
 - GreedyZentrum: $O(n^2)$.
- Gesamlaufzeit $O(n^4)$.

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl

$$\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$$

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:
COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$
- k -Färbungsproblem:

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:
COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$
- k -Färbungsproblem:
 k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leq k$

- k -Färbungsproblem:

k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

- praktisches Färbungsproblem:

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E)$, k , Frage: Gilt $\chi(G) \leq k$

- k -Färbungsproblem:

k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

- praktisches Färbungsproblem:
 - Gegeben: $G = (V, E)$

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k -Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
 $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

COL: Gegeben: $G = (V, E)$, k , Frage: Gilt $\chi(G) \leq k$

- k -Färbungsproblem:

k -COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leq k$

- praktisches Färbungsproblem:

- Gegeben: $G = (V, E)$
- Bestimme: $c : V \mapsto \mathbb{N}_{\chi(G)} : \forall \{v, w\} \in E : c(v) \neq c(w)$.

Approximationsfehler von Greedy

- ➊ Eingabe: $G = (V, E)$.

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
- ② $\forall v \in V : c(v) = 0$.

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
- ② $\forall v \in V : c(v) = 0$.
- ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
- ② $\forall v \in V : c(v) = 0$.
- ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
- ② $\forall v \in V : c(v) = 0$.
- ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.
 - ② Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
- ② $\forall v \in V : c(v) = 0$.
- ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.
 - ② Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
- ② $\forall v \in V : c(v) = 0$.
- ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.
 - ② Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$
- Greedyverfahren liefert Färbung mit $A_{Greedy} \leq \Delta(G) + 1$ Farben.

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
- ② $\forall v \in V : c(v) = 0$.
- ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.
 - ② Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$
- Greedyverfahren liefert Färbung mit $A_{Greedy} \leq \Delta(G) + 1$ Farben.
- Zweifärbung ist in \mathcal{P} .

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
 - ② $\forall v \in V : c(v) = 0$.
 - ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.
 - ② Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$
- Greedyverfahren liefert Färbung mit $A_{Greedy} \leq \Delta(G) + 1$ Farben.
 - Zweifärbung ist in \mathcal{P} .
 - Approximationsfehler: $\frac{A_{Greedy}}{\chi(G)} \leq \frac{\Delta(G)+1}{3}$.

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
 - ② $\forall v \in V : c(v) = 0$.
 - ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.
 - ② Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$
- Greedyverfahren liefert Färbung mit $A_{Greedy} \leq \Delta(G) + 1$ Farben.
 - Zweifärbung ist in \mathcal{P} .
 - Approximationsfehler: $\frac{A_{Greedy}}{\chi(G)} \leq \frac{\Delta(G)+1}{3}$.

Approximationsfehler von Greedy

- ① Eingabe: $G = (V, E)$.
 - ② $\forall v \in V : c(v) = 0$.
 - ③ Solange es $v \in V$ gibt, mit $c(v) = 0$:
 - ① Wähle $v \in V$ gibt, mit $c(v) = 0$.
 - ② Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$
- Greedyverfahren liefert Färbung mit $A_{Greedy} \leq \Delta(G) + 1$ Farben.
 - Zweifärbung ist in \mathcal{P} .
 - Approximationsfehler: $\frac{A_{Greedy}}{\chi(G)} \leq \frac{\Delta(G)+1}{3}$.

Lemma

Es gibt eine Folge der Knoten (v_1, v_2, \dots, v_n) , so dass "Greedy" den Graphen mit $\chi(G)$ färbt.

Beweis: wähle Folge mit: $\chi(v_i) \leq \chi(v_{i+1})$.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
 - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
 - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.
 - Laufzeit: $O(1)$ und Fehlerfaktor 1.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
 - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.
 - Laufzeit: $O(1)$ und Fehlerfaktor 1.
- Falls $|V| > 2/c$, dann färbe G :

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
 - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.
 - Laufzeit: $O(1)$ und Fehlerfaktor 1.
- Falls $|V| > 2/c$, dann färbe G :
 - Partitioniere $V(G)$ in $\lfloor c \cdot n \rfloor$ Teile der Größe $\lfloor n/\lfloor c \cdot n \rfloor \rfloor$ oder $\lceil n/\lfloor c \cdot n \rfloor \rceil$.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
 - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.
 - Laufzeit: $O(1)$ und Fehlerfaktor 1.
- Falls $|V| > 2/c$, dann färbe G :
 - Partitioniere $V(G)$ in $\lfloor c \cdot n \rfloor$ Teile der Größe $\lfloor n/\lfloor c \cdot n \rfloor \rfloor$ oder $\lceil n/\lfloor c \cdot n \rfloor \rceil$.
 - Jeder Teil hat Größe $\leq \frac{n}{cn-1} + 1 \leq \frac{2}{c} = O(1)$.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
 - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.
 - Laufzeit: $O(1)$ und Fehlerfaktor 1.
- Falls $|V| > 2/c$, dann färbe G :
 - Partitioniere $V(G)$ in $\lfloor c \cdot n \rfloor$ Teile der Größe $\lfloor n/\lfloor c \cdot n \rfloor \rfloor$ oder $\lceil n/\lfloor c \cdot n \rfloor \rceil$.
 - Jeder Teil hat Größe $\leq \frac{n}{cn-1} + 1 \leq \frac{2}{c} = O(1)$.
 - Jeder Teil kann in konstanter Zeit optimal gefärbt werden.

Aussagen

Lemma

Sei $0 < c \leq 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leq 2/c$, dann färbe G :
 - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
 - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.
 - Laufzeit: $O(1)$ und Fehlerfaktor 1.
- Falls $|V| > 2/c$, dann färbe G :
 - Partitioniere $V(G)$ in $\lfloor c \cdot n \rfloor$ Teile der Größe $\lfloor n/\lfloor c \cdot n \rfloor \rfloor$ oder $\lceil n/\lfloor c \cdot n \rfloor \rceil$.
 - Jeder Teil hat Größe $\leq \frac{n}{cn-1} + 1 \leq \frac{2}{c} = O(1)$.
 - Jeder Teil kann in konstanter Zeit optimal gefärbt werden.
 - Gesamtfarbenanzahl: $\frac{\lfloor c \cdot n \rfloor \cdot \chi(G)}{\chi(G)} \leq cn$.

Aussagen

Lemma

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler $4/3$ für das Färbungsproblem.

Aussagen

Lemma

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler $4/3$ für das Färbungsproblem.

Aussagen

Lemma

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler $4/3$ für das Färbungsproblem.

Theorem (Garry, Johnson 1976)

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler 2 für das Färbungsproblem.

Aussagen

Lemma

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler $4/3$ für das Färbungsproblem.

Theorem (Garry, Johnson 1976)

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler 2 für das Färbungsproblem.

Aussagen

Lemma

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler $4/3$ für das Färbungsproblem.

Theorem (Garry, Johnson 1976)

Falls $\mathcal{P} \neq \text{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler 2 für das Färbungsproblem.

Theorem (Johnson 1974)

Das Färbungsproblem kann mit einem Faktor von $O(n/\log n)$ in Zeit $O(nm)$ approximiert werden.

Aussagen

ZPP = Zero-error Probabilistic Polynomial Time

Theorem (Lund, Yannakakis 1993)

Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es für beliebiges $\varepsilon > 0$ keinen Polynomzeitalgorithmus mit Approximationsfehler n^ε für das Färbungsproblem.

Aussagen

ZPP = Zero-error Probabilistic Polynomial Time

Theorem (Lund, Yannakakis 1993)

Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es für beliebiges $\varepsilon > 0$ keinen Polynomzeitalgorithmus mit Approximationsfehler n^ε für das Färbungsproblem.

Aussagen

ZPP = Zero-error Probabilistic Polynomial Time

Theorem (Lund, Yannakakis 1993)

Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es für beliebiges $\varepsilon > 0$ keinen Polynomzeitalgorithmus mit Approximationsfehler n^ε für das Färbungsproblem.

Theorem (Feige, Kilian 1996)

Falls $\mathcal{P} \neq \mathcal{ZPP}$, dann gibt es für beliebiges $\varepsilon > 0$ keinen Polynomzeitalgorithmus mit Approximationsfehler $n^{1-\varepsilon}$ für das Färbungsproblem.

Literatur

- Vazirani: Approximation Algorithms, Springer Verlag, 2001.

Literatur

- Vazirani: Approximation Algorithms, Springer Verlag, 2001.
- Jansen, Margraf: Approximative Algorithmen und Nichtapproximierbarkeit, de Gruyter, 2008.

Literatur

- Vazirani: Approximation Algorithms, Springer Verlag, 2001.
- Jansen, Margraf: Approximative Algorithmen und Nichtapproximierbarkeit, de Gruyter, 2008.
- Wanka: Approximation algorithms – Eine Einführung, Teubner Verlag 2006.

Literatur

- Vazirani: Approximation Algorithms, Springer Verlag, 2001.
- Jansen, Margraf: Approximative Algorithmen und Nichtapproximierbarkeit, de Gruyter, 2008.
- Wanka: Approximationsalgorithmen – Eine Einführung, Teubner Verlag 2006.
- Hochbaum: Approximation Algorithms for NP-Hard Problems, Thomson Publishing, 1996.

Literatur

- Vazirani: Approximation Algorithms, Springer Verlag, 2001.
- Jansen, Margraf: Approximative Algorithmen und Nichtapproximierbarkeit, de Gruyter, 2008.
- Wanka: Approximationsalgorithmen – Eine Einführung, Teubner Verlag 2006.
- Hochbaum: Approximation Algorithms for NP-Hard Problems, Thomson Publishing, 1996.
- Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela, Protasi: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer Verlag, 1999.

Literatur

- Vazirani: Approximation Algorithms, Springer Verlag, 2001.
- Jansen, Margraf: Approximative Algorithmen und Nichtapproximierbarkeit, de Gruyter, 2008.
- Wanka: Approximationsalgorithmen – Eine Einführung, Teubner Verlag 2006.
- Hochbaum: Approximation Algorithms for NP-Hard Problems, Thomson Publishing, 1996.
- Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela, Protasi: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer Verlag, 1999.
- Garey, Johnson: Computers and Intractability, Freeman and Company, 1979.

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Vertex Cover Problem approximiert werden? Welche untere Schranken sind dazu bekannt?

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Vertex Cover Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Independent Set Problem approximiert werden? Welche untere Schranken sind dazu bekannt?

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Vertex Cover Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Independent Set Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Steinerbaum Problem approximiert werden? Welche untere Schranken sind dazu bekannt?

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Vertex Cover Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Independent Set Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Steinerbaum Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem TSP approximiert werden? Welche untere Schranken sind dazu bekannt?

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Vertex Cover Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Independent Set Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Steinerbaum Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem TSP approximiert werden? Welche untere Schranken sind dazu bekannt?
- **Wie kann das Problem Δ -TSP approximiert werden? Welche untere Schranken sind dazu bekannt?**

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Vertex Cover Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Independent Set Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Steinerbaum Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem TSP approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Δ -TSP approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Zentrumsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?