Effiziente Algorithmen (SS2022)
Chapter 6
Approximation I

Walter Unger

Lehrstuhl für Informatik 1

—— 03.06.2022 10:29:31 ——

Contents I

Einleitung

- NP-schwere Probleme doch lösen

Einleitung

- NP-schwere Probleme doch lösen
  - Exakt: "$\Longrightarrow$" exponentiale Laufzeit

Einleitung

- NP-schwere Probleme doch lösen
  - Exakt: "$\Longrightarrow$" exponentiale Laufzeit
  - Nicht Exakt: "$\Longrightarrow$" hoffentlich polynomiale Laufzeit

Einleitung

- NP-schwere Probleme doch lösen
  - Exakt: "$\Longrightarrow$" exponentiale Laufzeit
  - Nicht Exakt: "$\Longrightarrow$" hoffentlich polynomiale Laufzeit
- Hier: Approximationsalgorithmen in polynomialer Laufzeit.

Einleitung

- NP-schwere Probleme doch lösen
  - Exakt: "$\Longrightarrow$" exponentiale Laufzeit
  - Nicht Exakt: "$\Longrightarrow$" hoffentlich polynomiale Laufzeit
- Hier: Approximationsalgorithmen in polynomialer Laufzeit.
- D.h.: welche Approximationsfaktoren sind möglich?

Einleitung

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

Einleitung

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

Definition (Approximation)

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)

$\mathbb{N}_k = \{1, \ldots, k\}$

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:

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

Definition (Approximation)

$\mathbb{N}_k = \{1, \ldots, k\}$

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:

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

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 $\mathcal{NPC}$:

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 $|C| \leqslant k$.

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 $|C| \leqslant k$.
- Reduktionsidee: $V \setminus C$ ist eine stabile Menge.

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.

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 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.
- 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 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.
- 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 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.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
  - $M_{\max}(G)$ Maximales Matching von G.
  - Schätze Approximationsfaktor ab:

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

Vertex Cover

Theorem

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

1. Bestimme inklusions-maximales Matching $M \subset E$ auf 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.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
  - $M_{\max}(G)$ Maximales Matching von G.
  - Schätze Approximationsfaktor ab:

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

Vertex Cover

Theorem

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

1. Bestimme inklusions-maximales Matching $M \subset E$ auf 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.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
  - $M_{\max}(G)$ Maximales Matching von G.
  - Schätze Approximationsfaktor ab:

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

Vertex Cover

Theorem

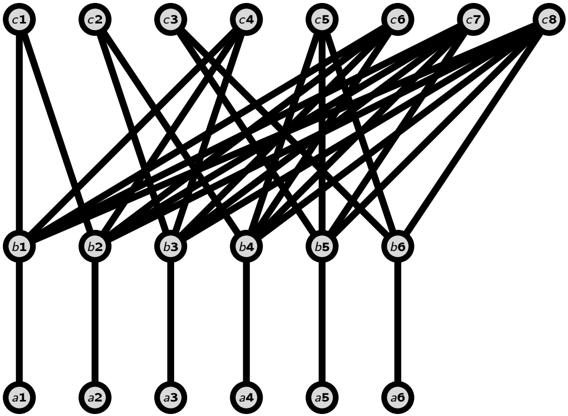
Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

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

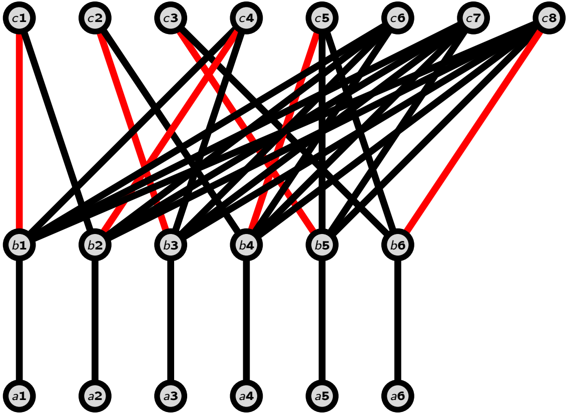
- Zeige Korrektheit:
  - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
  - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
  - $M_{\max}(G)$ Maximales Matching von G.
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

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

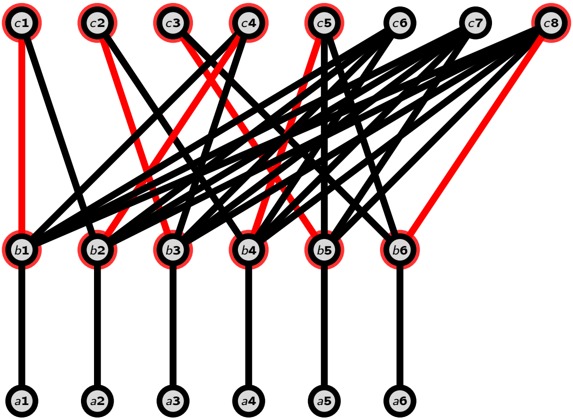
Beispiel



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.
- $\mathcal{NP}$-vollständige Probleme unterscheiden sich in ihrer Approximierbarkeit.

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.
- $\mathcal{NP}$-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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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 $c(C) \leqslant L$.
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.
  - Anderenfalls hat G' eine Tour mit Kosten $\geqslant n + 1$.

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.
Bestimme spannenden einfachen Kreis C mit $c(C) \leqslant L$.
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.
  - Anderenfalls hat G' eine Tour mit Kosten $\geqslant n + 1$.

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.
  - Anderenfalls hat G' eine Tour mit Kosten $\geqslant n - 1 + \alpha(n) \cdot n$.

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.
  - Anderenfalls hat G' eine Tour mit Kosten $\geqslant n - 1 + \alpha(n) \cdot n$.

Δ-TSP

Definition (Δ-TSP)

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

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

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

Δ-TSP

Definition (Δ-TSP)

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

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

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

Δ-TSP

Definition (Δ-TSP)

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

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

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

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

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

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

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

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

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

- 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 $\mathcal{NPC}$.

Beweis: Reduktion von Hamilton Kreis auf planare Graphen.

Δ-TSP

Definition (Δ-TSP)

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

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

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

- 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 $\mathcal{NPC}$.

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.

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

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

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.

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'.

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).

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

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.
6. Dadurch entsteht Kreis C, gebe C aus.

Approximation

Theorem

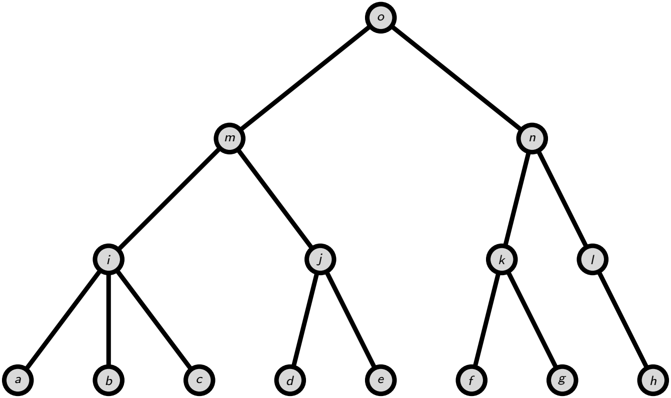
Δ-TSP ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

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.
6. Dadurch entsteht Kreis C, gebe C aus.

Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

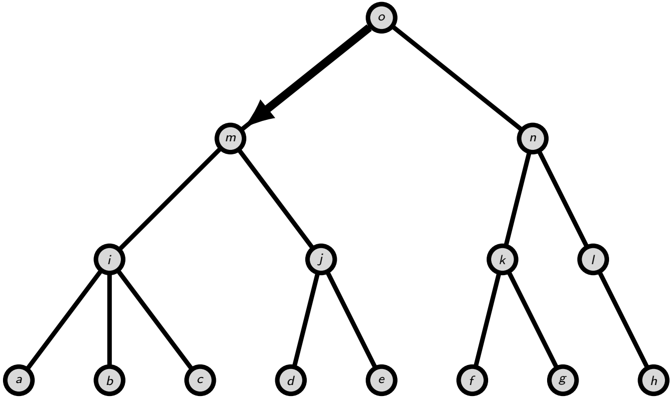
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

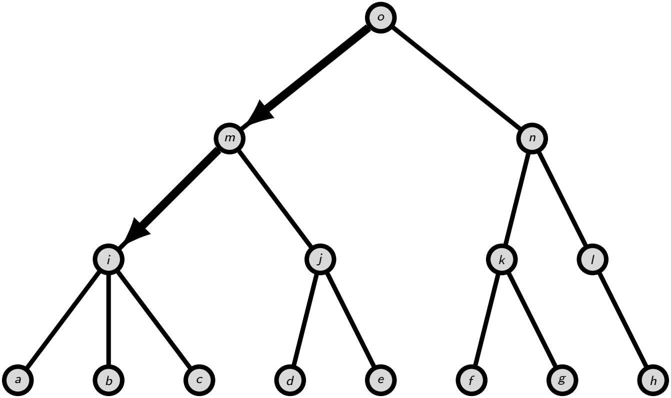
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

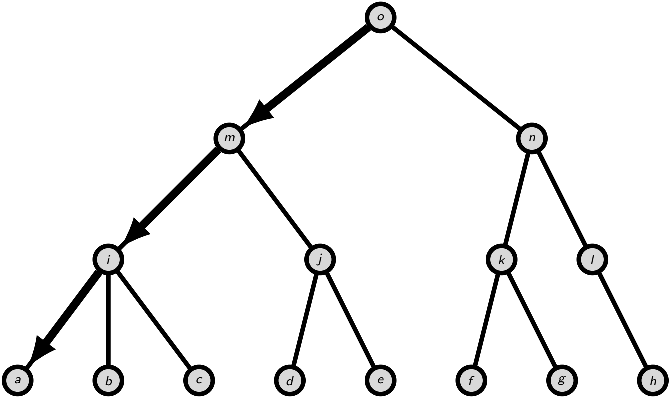
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

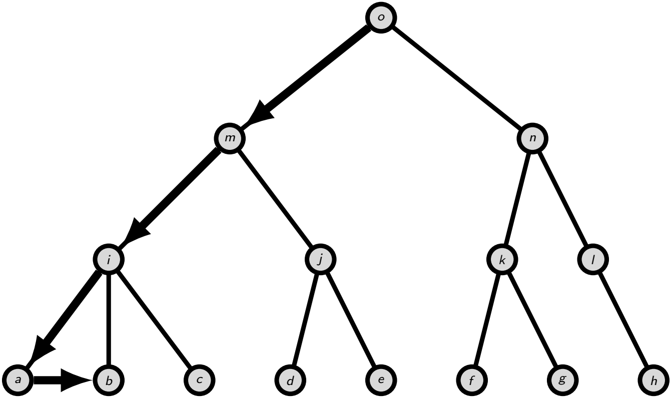
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

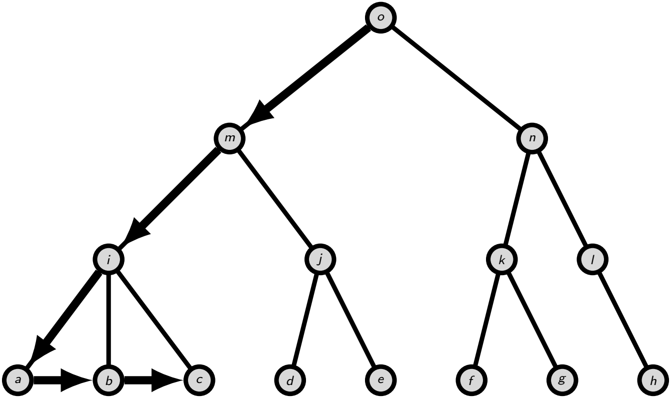
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

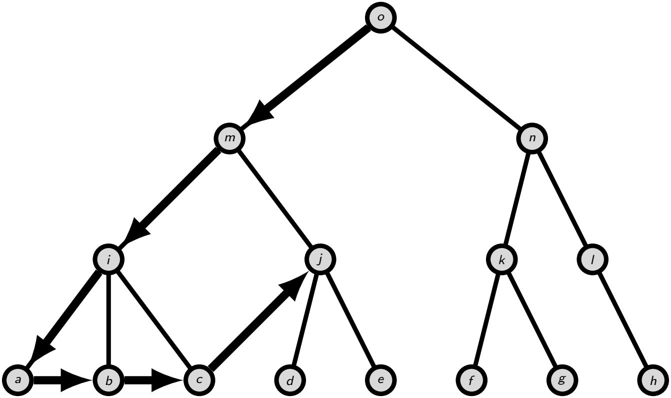
Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

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

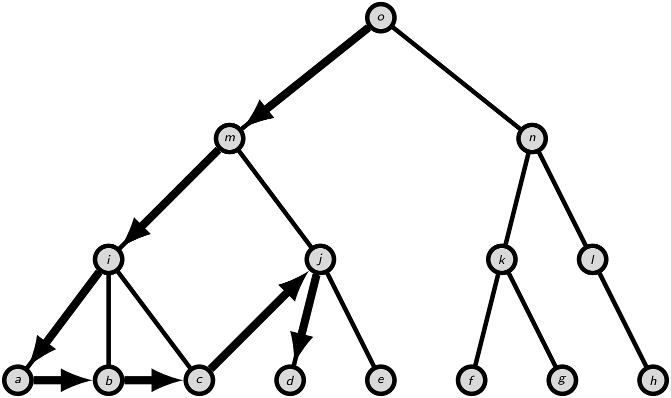
Bestimme Eulertour: Gehe einmal inorder um den Baum herum.
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

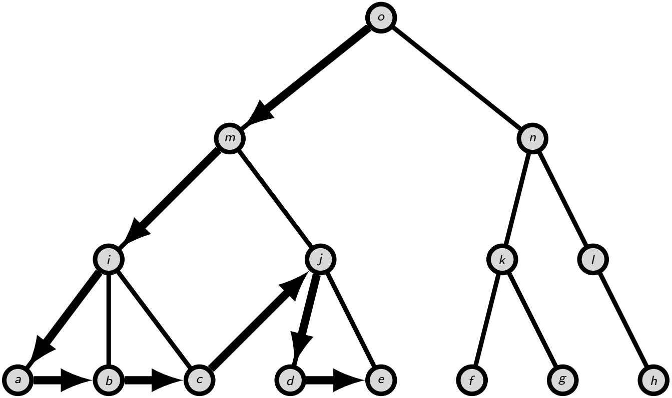
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

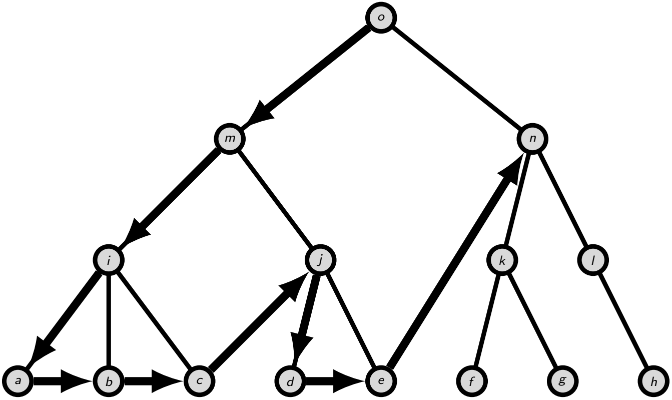
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

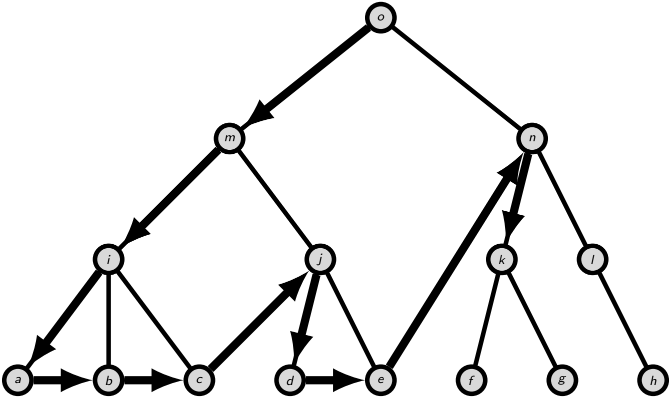
Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

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

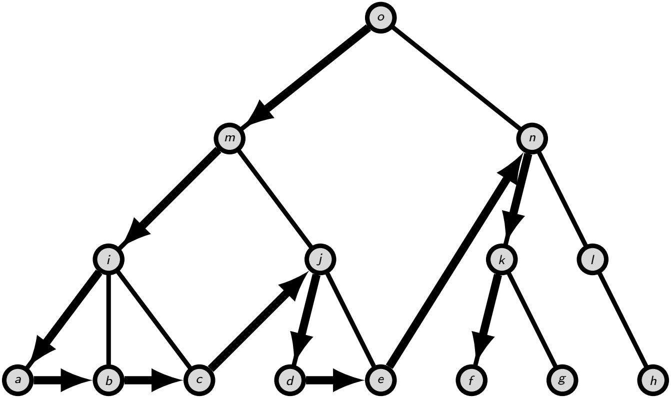
Bestimme Eulertour: Gehe einmal inorder um den Baum herum.
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

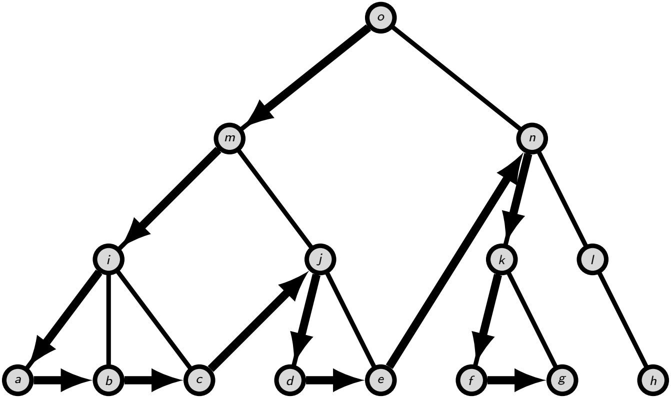
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

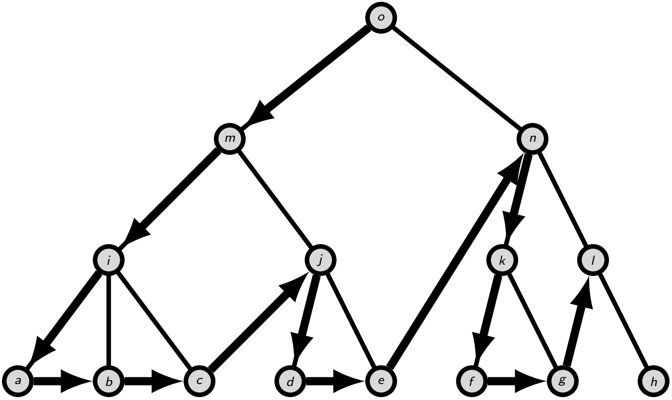
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

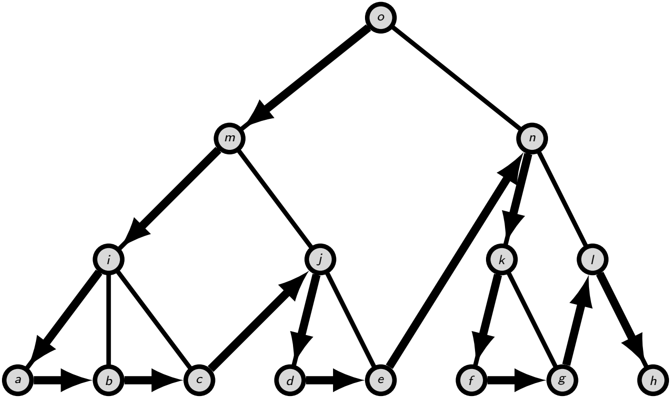
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

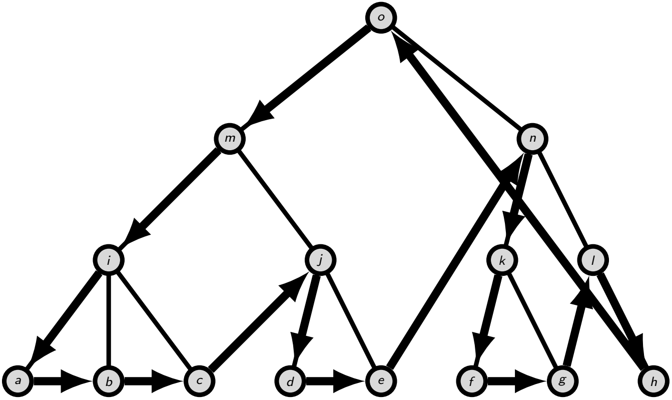
Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - 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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.
  - 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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.
  - 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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.
  - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{2 \cdot c(T)}{c(C^*)}$

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.
  - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{2 \cdot c(T)}{c(C^*)} \leqslant \frac{2 \cdot c(C^*)}{c(C^*)} = 2$

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.
  - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{2 \cdot c(T)}{c(C^*)} \leqslant \frac{2 \cdot c(C^*)}{c(C^*)} = 2$
- Laufzeit $O(n^2 \log n)$.

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.
  - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{2 \cdot c(T)}{c(C^*)} \leqslant \frac{2 \cdot c(C^*)}{c(C^*)} = 2$
- 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: “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.
- 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: “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.
- 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: "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.
- 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: "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.
- 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: "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.
- 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: “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.
- 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.
  - $2 \cdot |E| = \sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v)$.

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.
- 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

Δ-TSP ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

1.5-Approximation

Theorem

Δ-TSP ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G.

1.5-Approximation

Theorem

Δ-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

Δ-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

Δ-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

Δ-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

Δ-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

Δ-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

Δ-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.

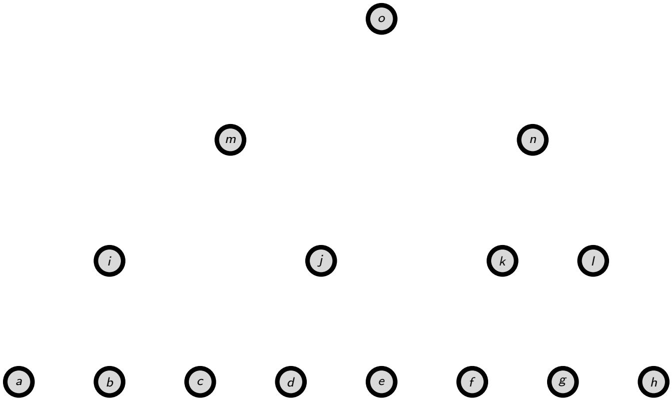
1.5-Approximation

Theorem

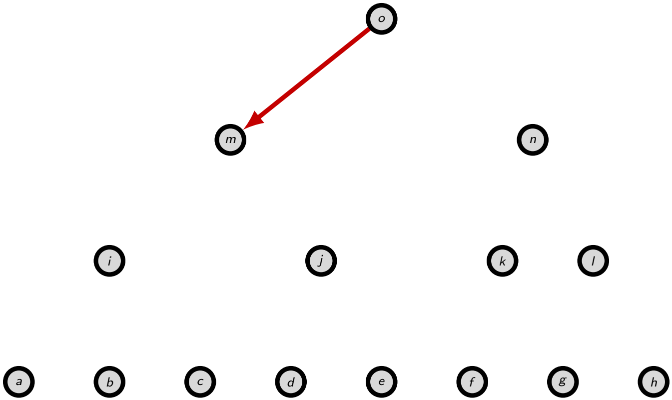
Δ-TSP ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- 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.

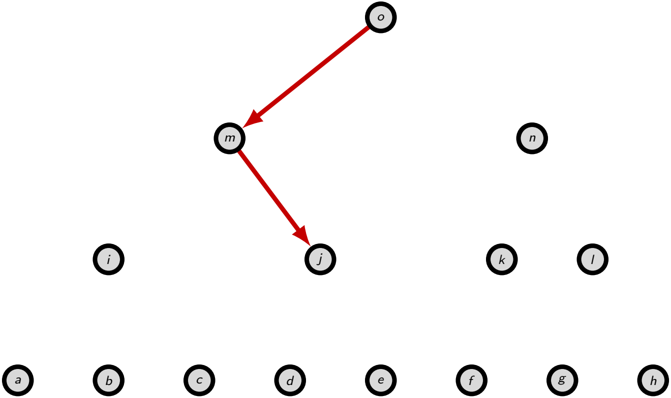
Beispiel (Bestimmen der Eulertour)



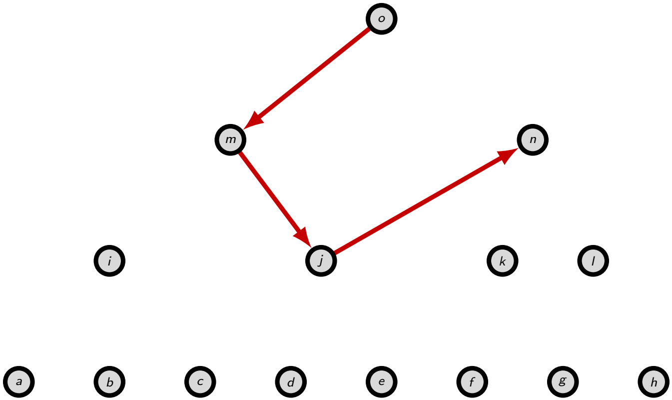
Beispiel (Bestimmen der Eulertour)



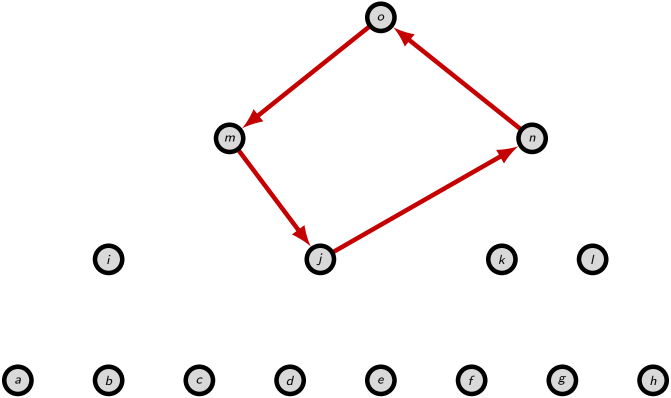
Beispiel (Bestimmen der Eulertour)



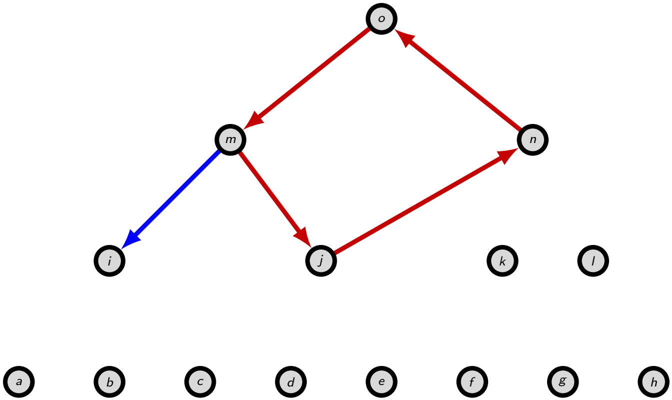
Beispiel (Bestimmen der Eulertour)



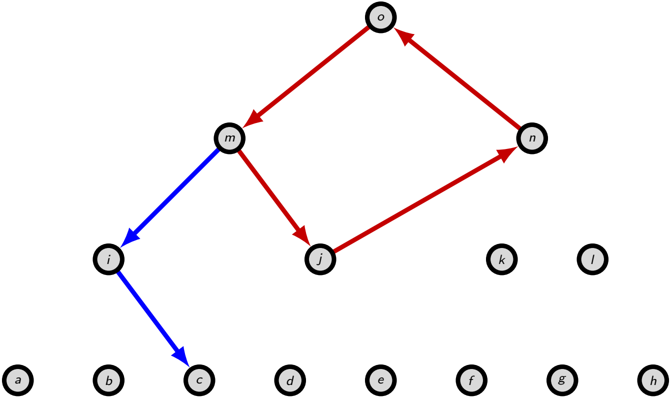
Beispiel (Bestimmen der Eulertour)



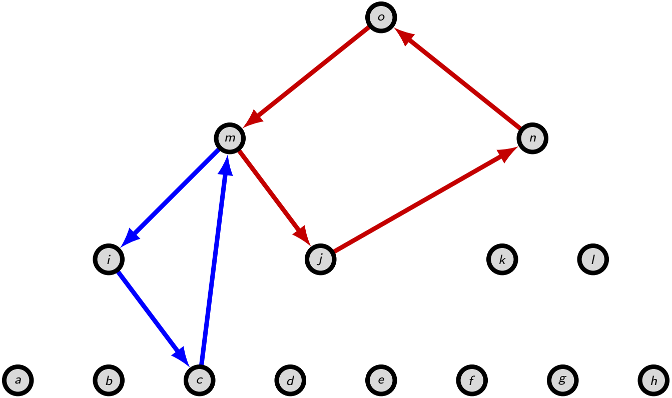
Beispiel (Bestimmen der Eulertour)



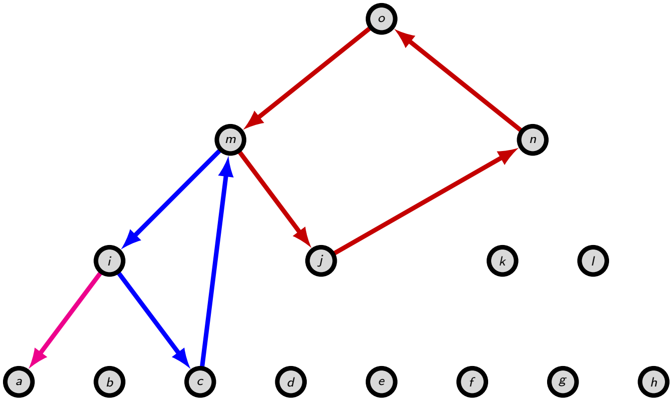
Beispiel (Bestimmen der Eulertour)



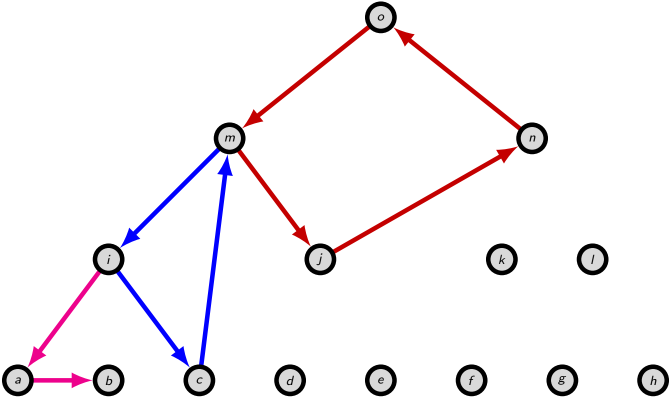
Beispiel (Bestimmen der Eulertour)



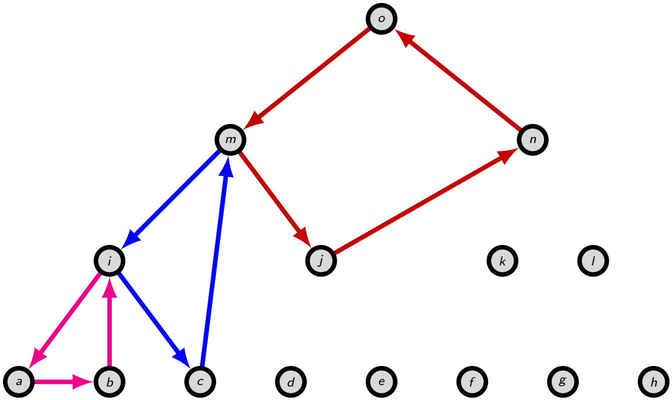
Beispiel (Bestimmen der Eulertour)



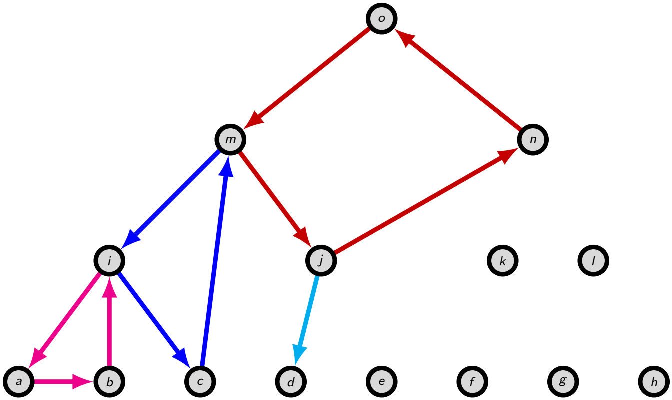
Beispiel (Bestimmen der Eulertour)



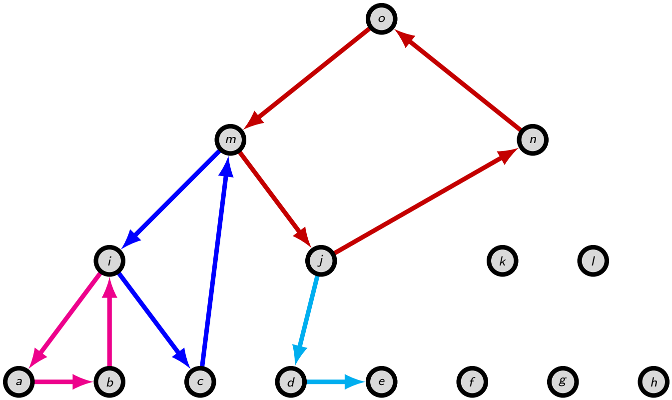
Beispiel (Bestimmen der Eulertour)



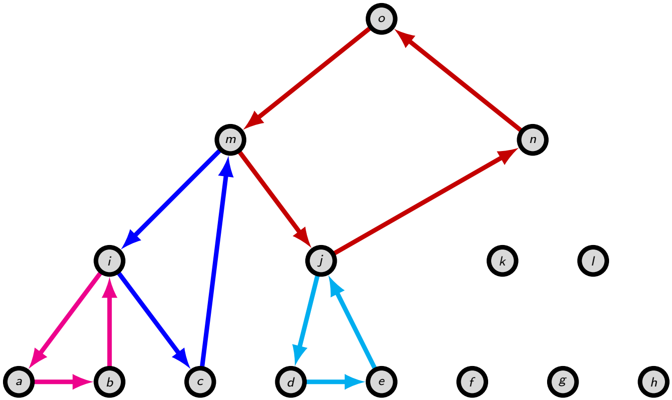
Beispiel (Bestimmen der Eulertour)



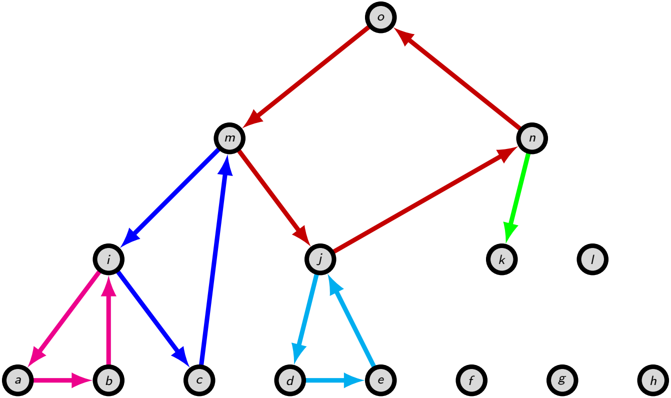
Beispiel (Bestimmen der Eulertour)



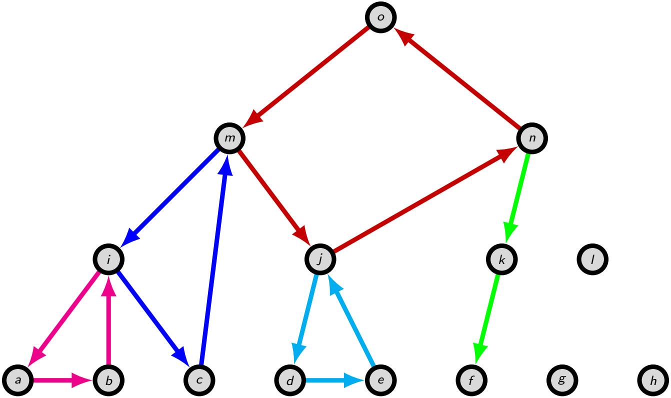
Beispiel (Bestimmen der Eulertour)



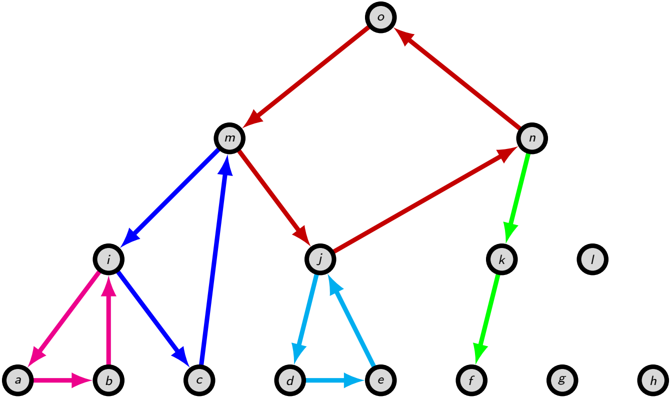
Beispiel (Bestimmen der Eulertour)



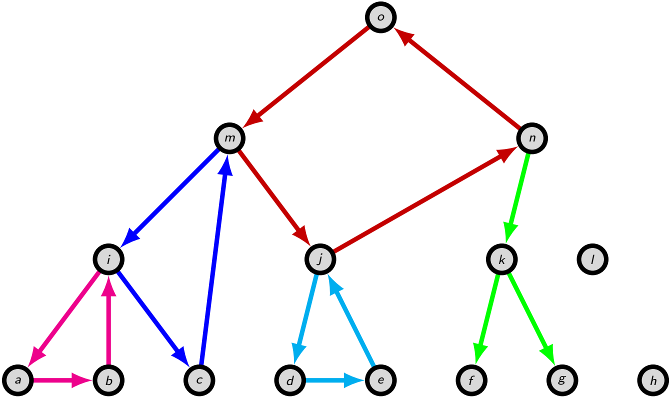
Beispiel (Bestimmen der Eulertour)



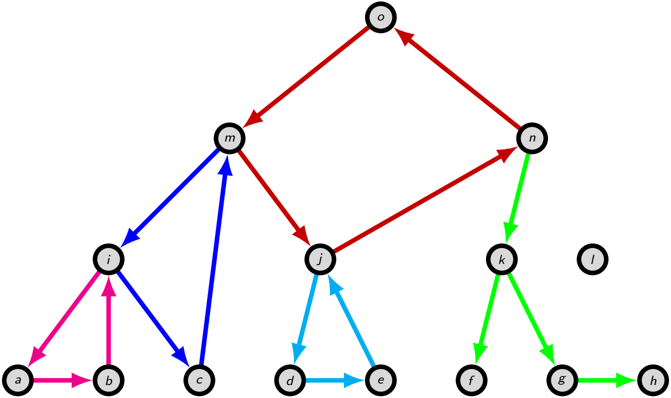
Beispiel (Bestimmen der Eulertour)



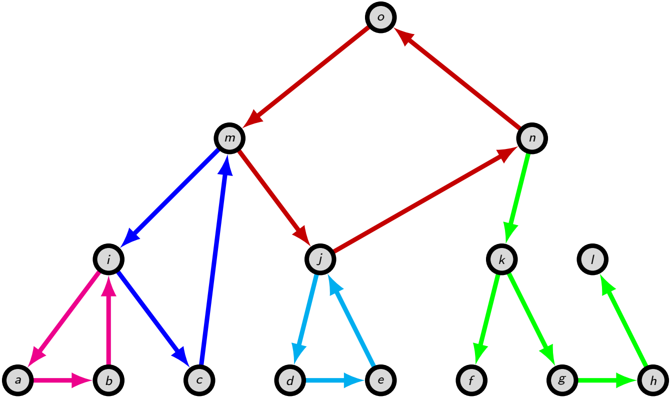
Beispiel (Bestimmen der Eulertour)



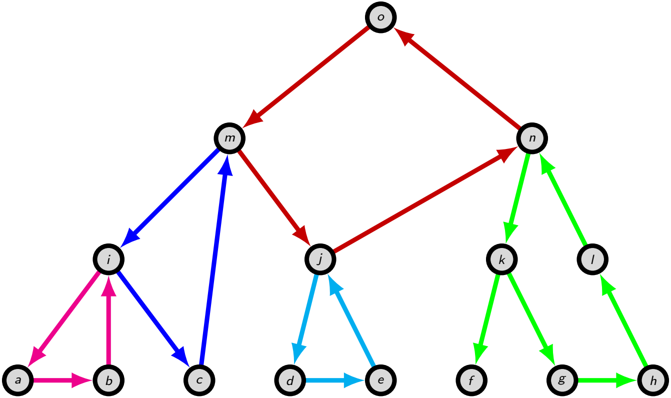
Beispiel (Bestimmen der Eulertour)



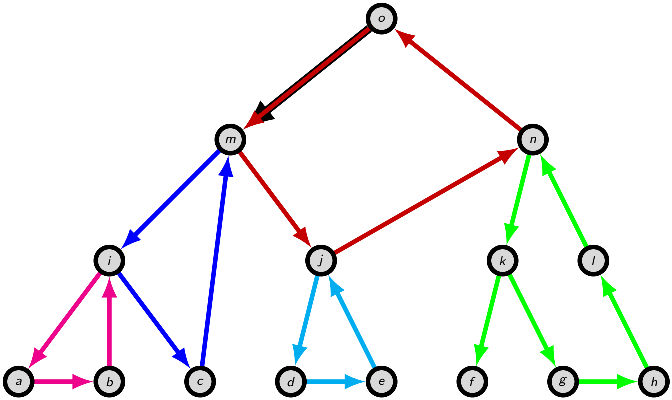
Beispiel (Bestimmen der Eulertour)



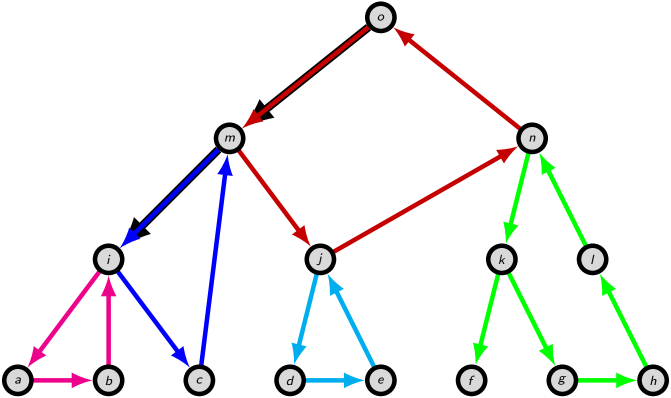
Beispiel (Bestimmen der Eulertour)



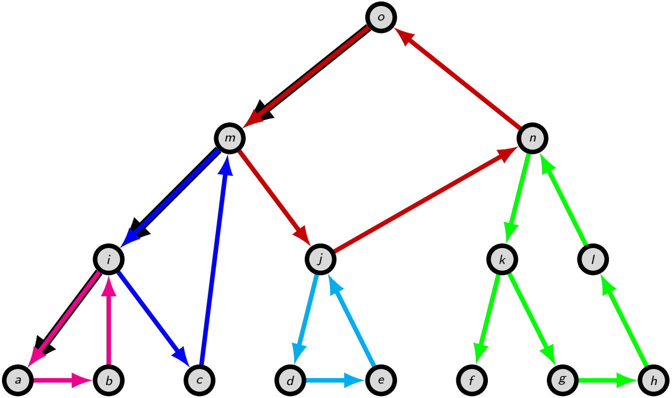
Beispiel (Bestimmen der Eulertour)



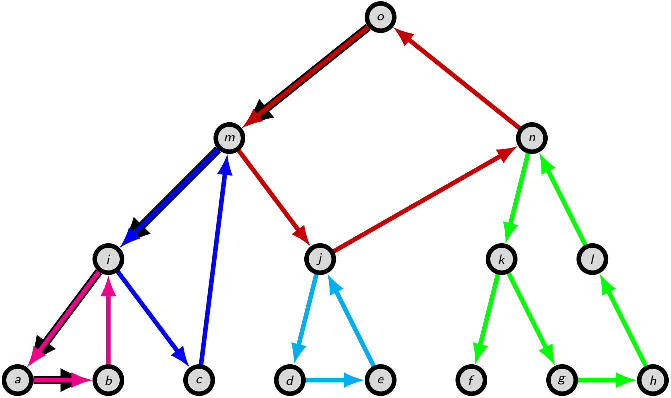
Beispiel (Bestimmen der Eulertour)



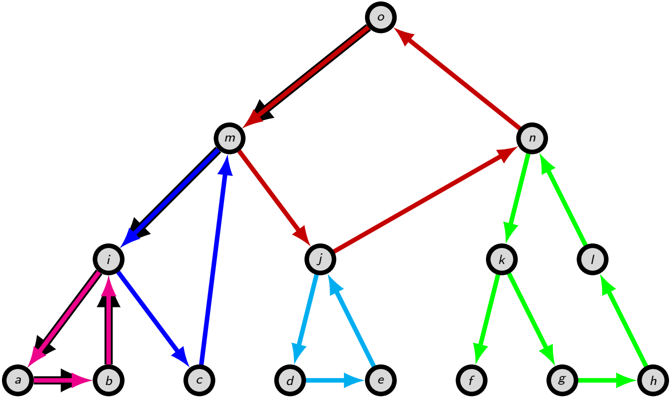
Beispiel (Bestimmen der Eulertour)



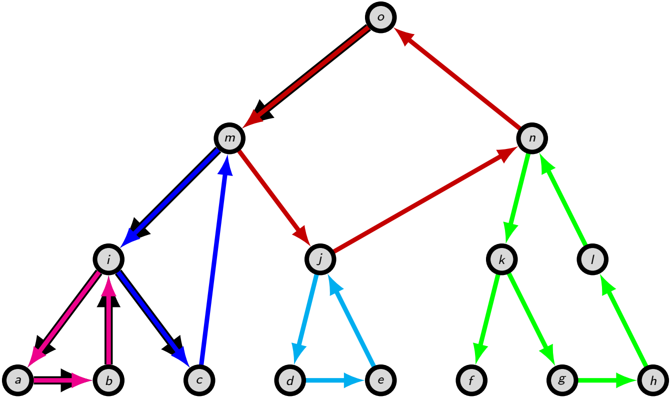
Beispiel (Bestimmen der Eulertour)



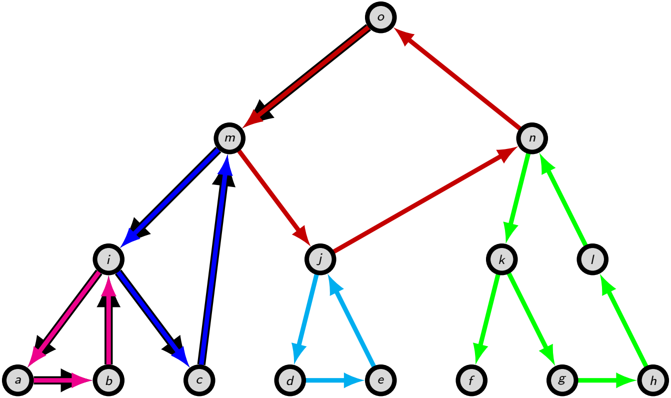
Beispiel (Bestimmen der Eulertour)



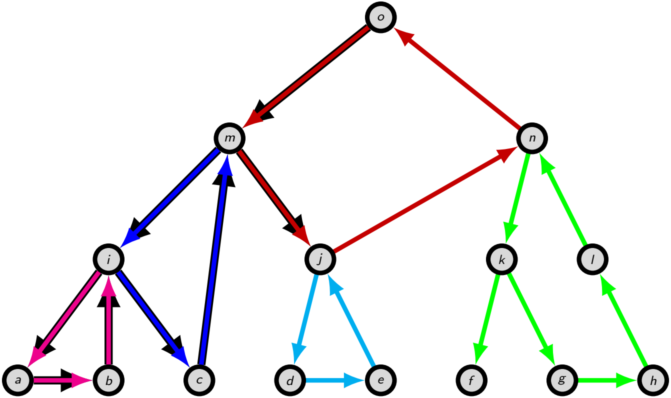
Beispiel (Bestimmen der Eulertour)



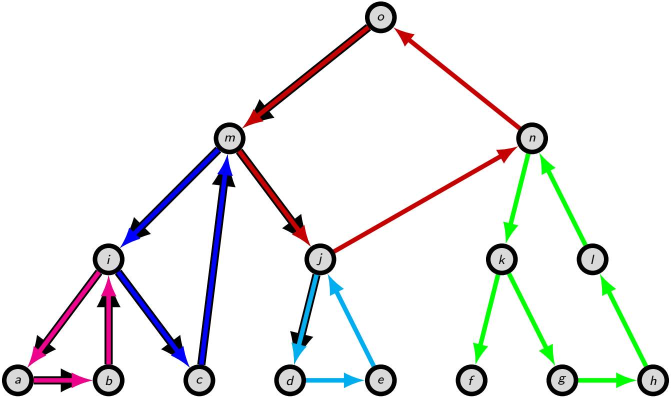
Beispiel (Bestimmen der Eulertour)



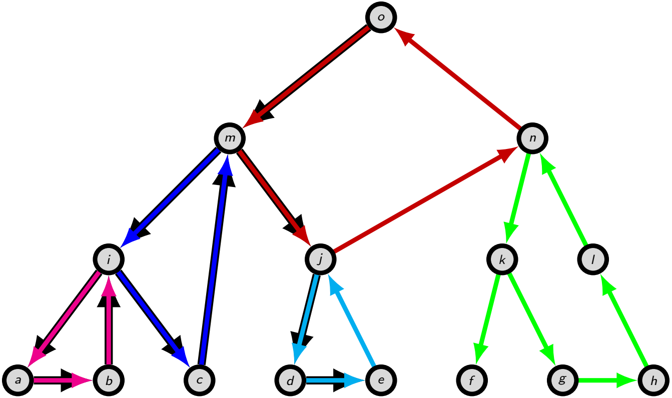
Beispiel (Bestimmen der Eulertour)



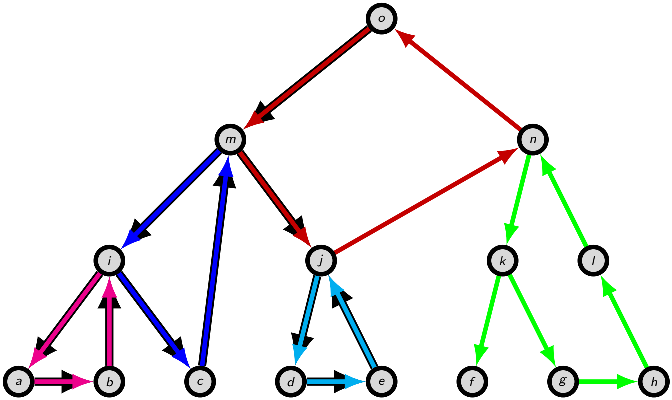
Beispiel (Bestimmen der Eulertour)



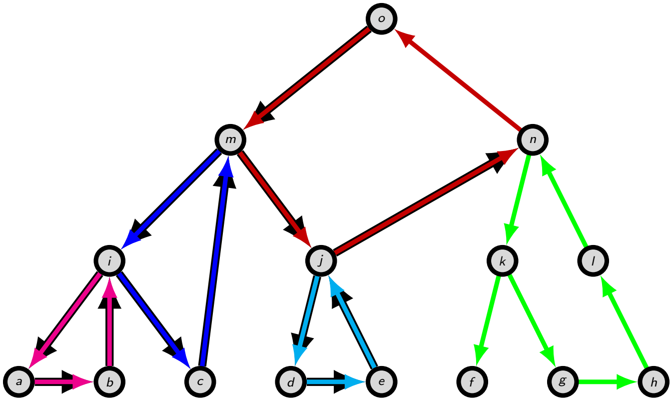
Beispiel (Bestimmen der Eulertour)



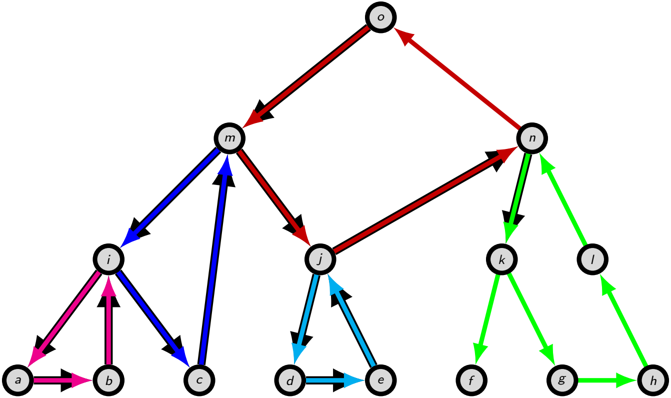
Beispiel (Bestimmen der Eulertour)



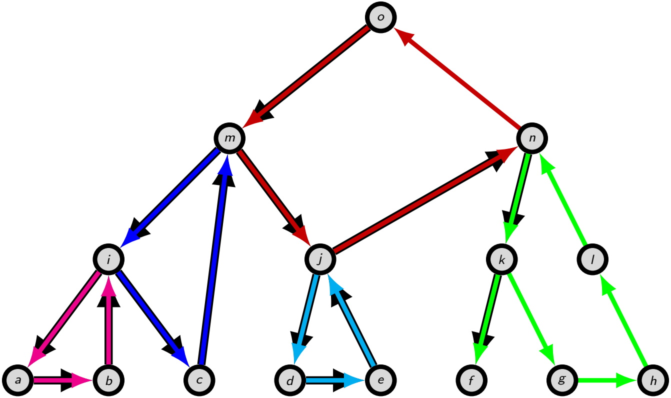
Beispiel (Bestimmen der Eulertour)



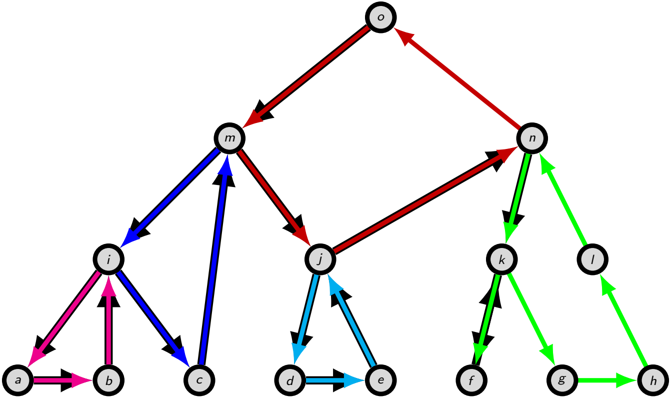
Beispiel (Bestimmen der Eulertour)



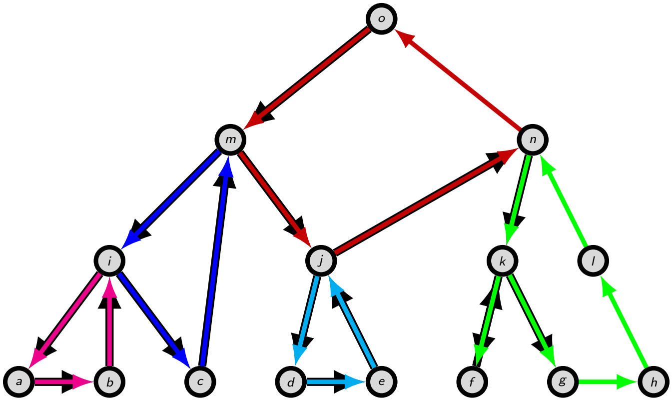
Beispiel (Bestimmen der Eulertour)



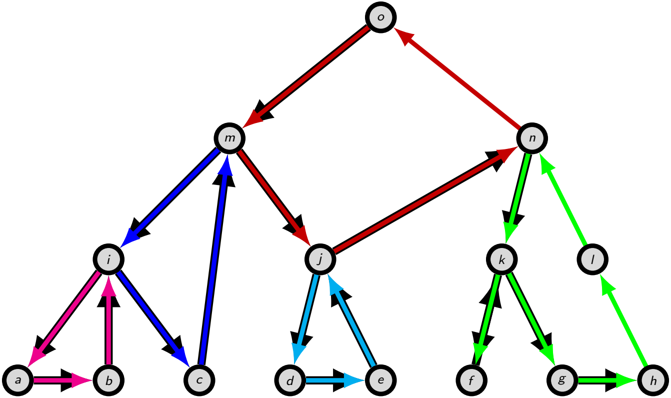
Beispiel (Bestimmen der Eulertour)



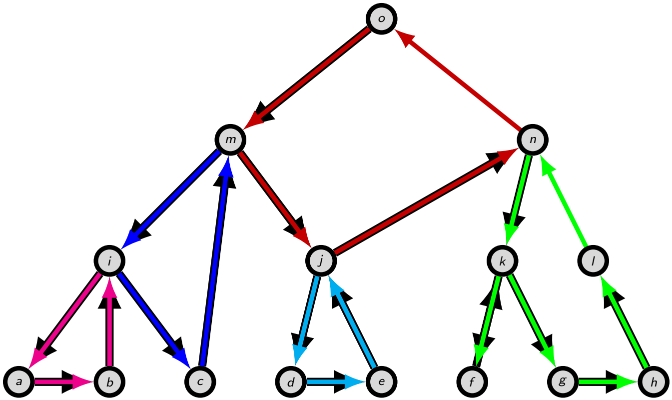
Beispiel (Bestimmen der Eulertour)



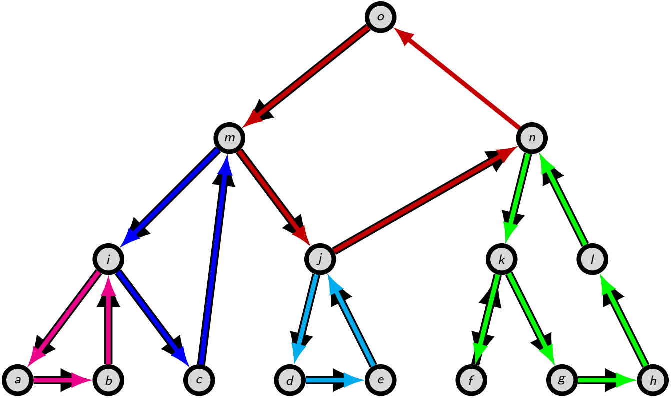
Beispiel (Bestimmen der Eulertour)



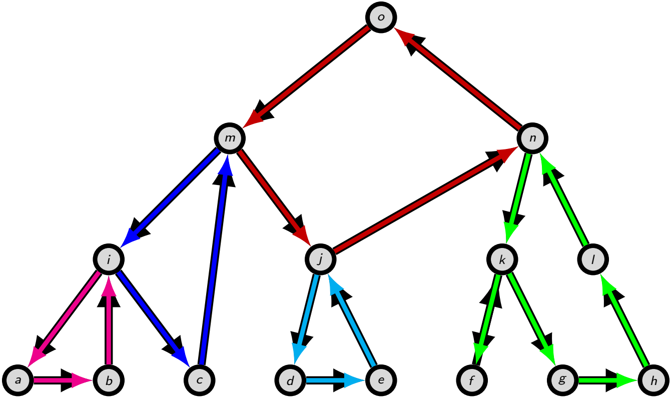
Beispiel (Bestimmen der Eulertour)



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 $C_1, C_2, \ldots$.

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 $C_1, C_2, \ldots$.
   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 $C_1, C_2, \ldots$.
   3. Gehe auf C_0 von v_1 bis v_0.

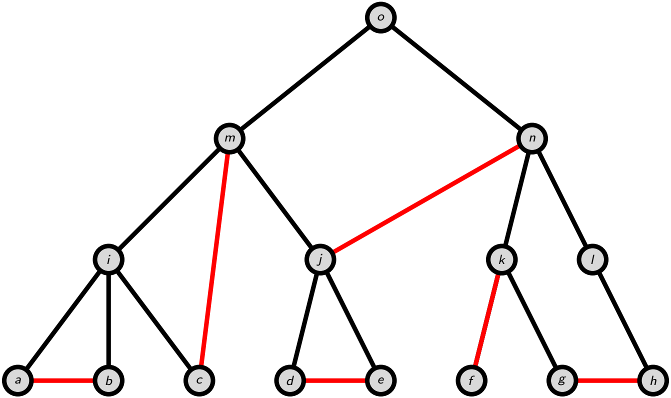
Laufzeit: $O(|E|)$.

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 $C_1, C_2, \ldots$.
   3. Gehe auf C_0 von v_1 bis v_0.

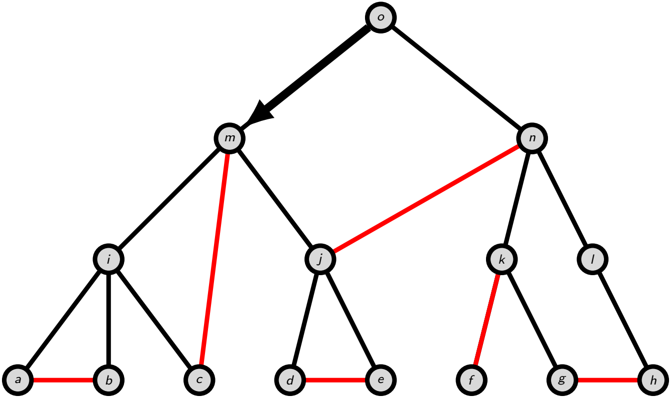
Laufzeit: $O(|E|)$.

Beispiel (1.5 Approximation)



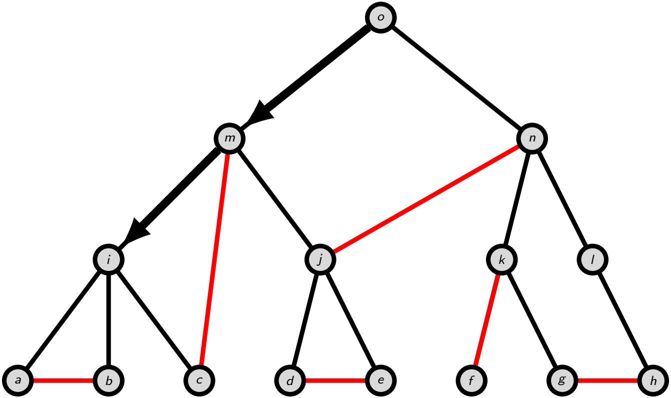
Beispiel (1.5 Approximation)

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



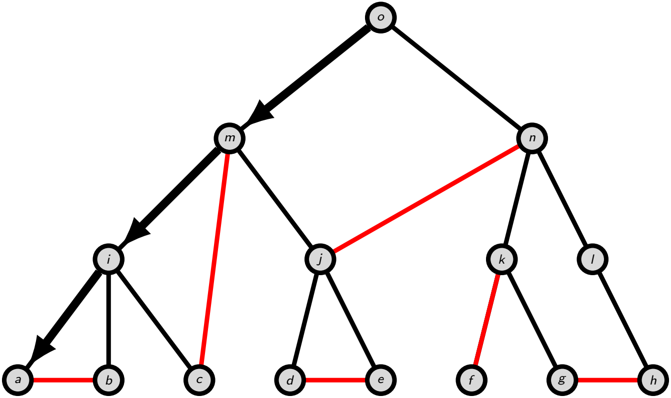
Beispiel (1.5 Approximation)

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



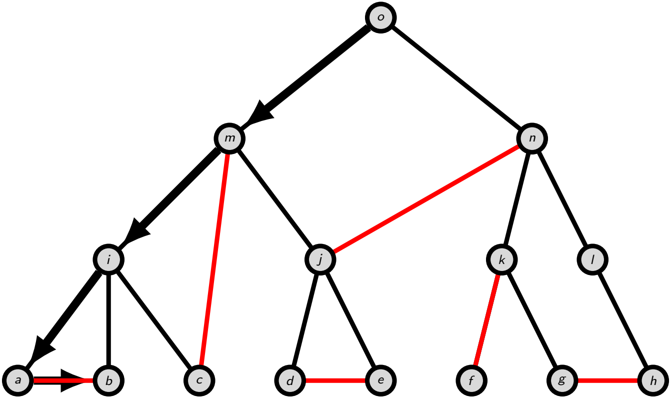
Beispiel (1.5 Approximation)

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



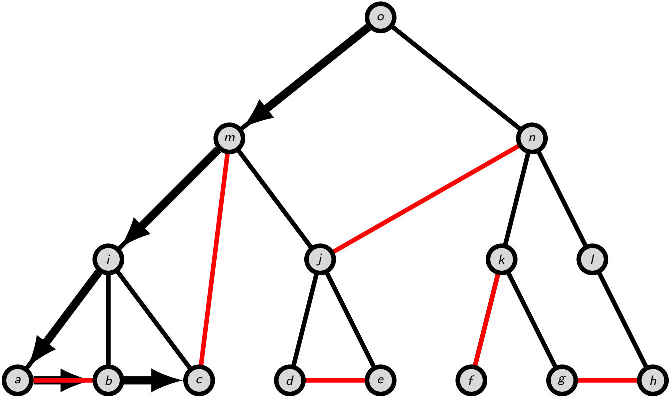
Beispiel (1.5 Approximation)

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



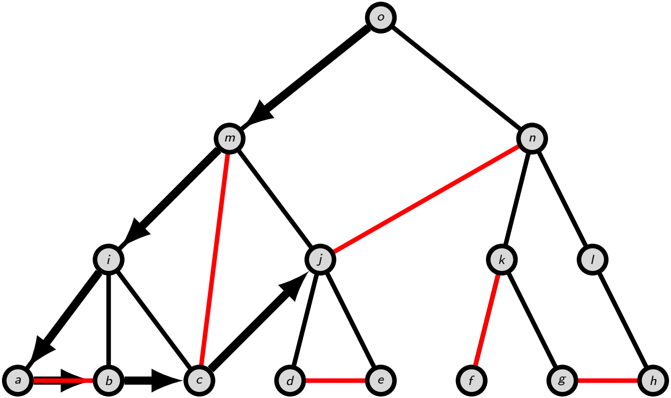
Beispiel (1.5 Approximation)

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



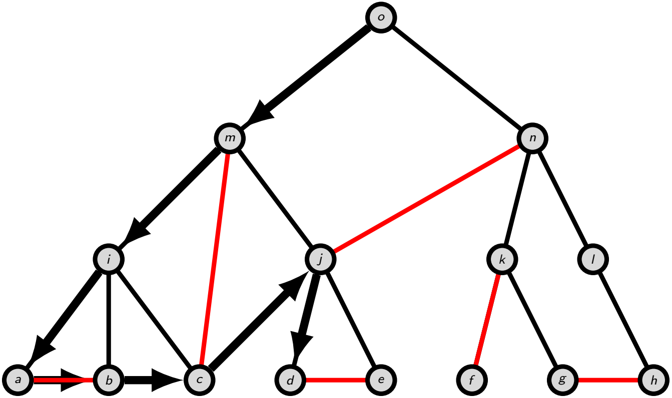
Beispiel (1.5 Approximation)

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



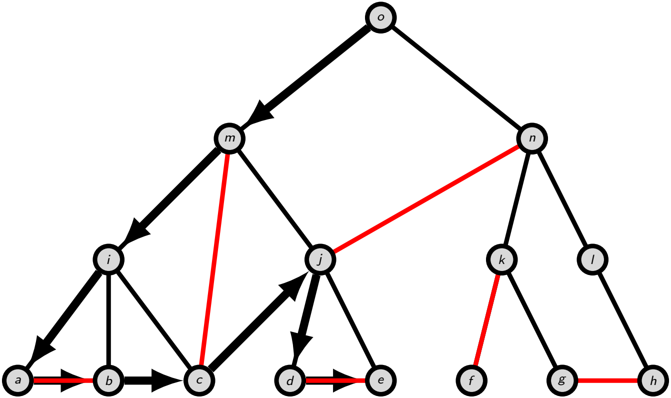
Beispiel (1.5 Approximation)

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



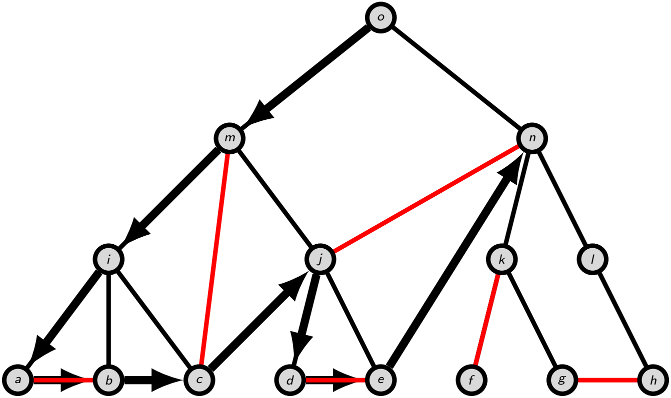
Beispiel (1.5 Approximation)

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



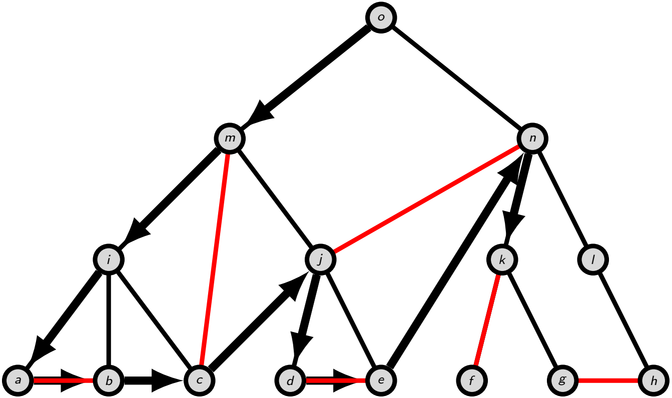
Beispiel (1.5 Approximation)

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



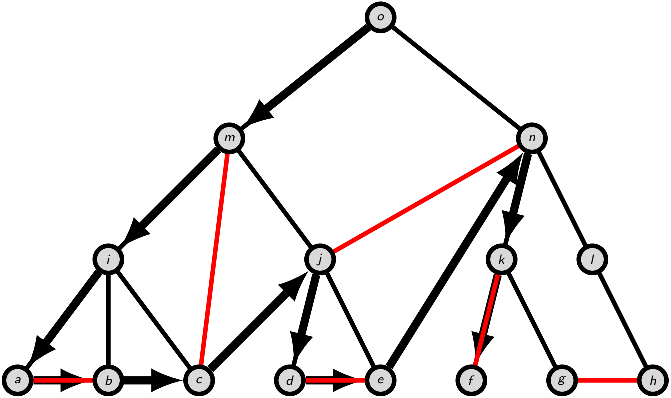
Beispiel (1.5 Approximation)

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



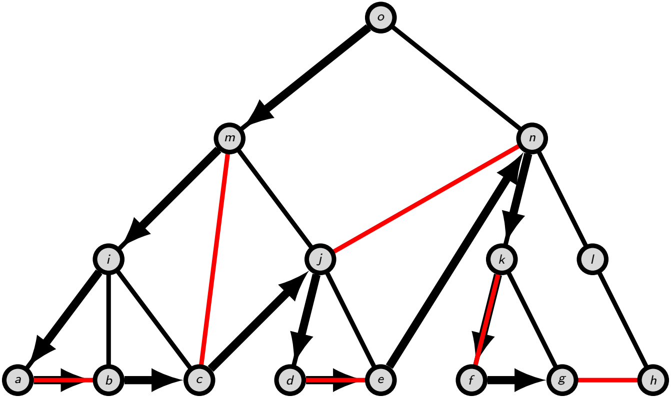
Beispiel (1.5 Approximation)

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



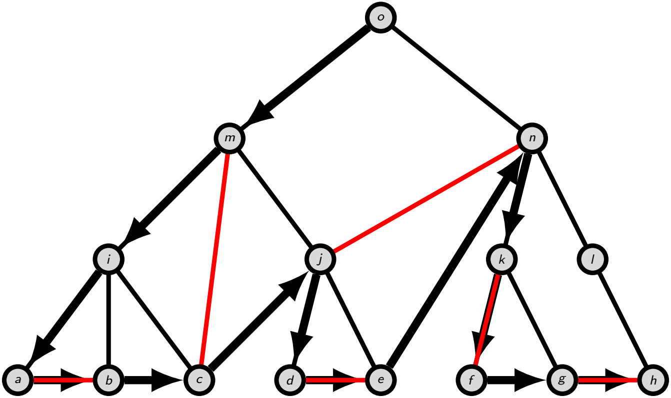
Beispiel (1.5 Approximation)

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



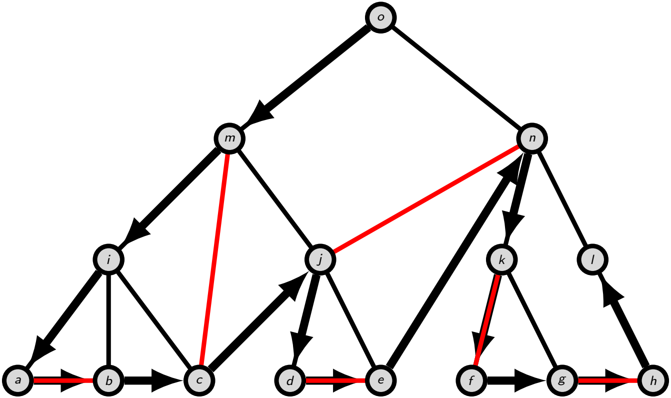
Beispiel (1.5 Approximation)

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



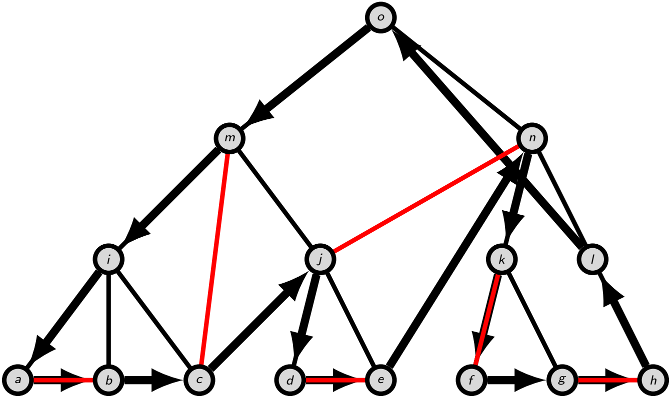
Beispiel (1.5 Approximation)

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - 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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leqslant c(C^*)/2$.

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leqslant c(C^*)/2$.
  - 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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leqslant c(C^*)/2$.
  - 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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leqslant c(C^*)/2$.
  - Siehe dazu nächste Folie.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{c(T)+c(M)}{c(C^*)}$

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.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leqslant c(C^*)/2$.
  - Siehe dazu nächste Folie.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{c(T)+c(M)}{c(C^*)} \leqslant \frac{c(C^*)+c(C^*)/2}{c(C^*)}$

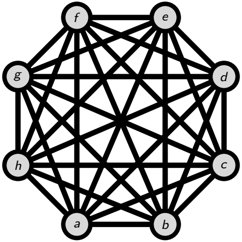
1.5-Approximation (Beweis)

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

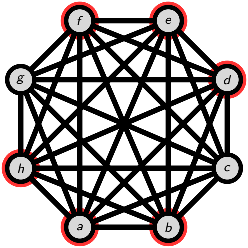
1.5-Approximation (Beweis)

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

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

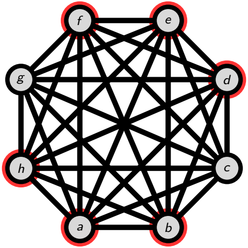


1.5-Approximation (Zeige $c(M) \leqslant c(C^*)/2$)



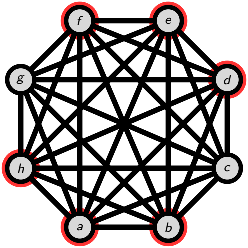
- Sei C^* eine optimale Lösung.

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



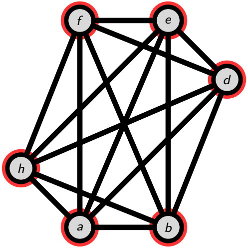
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T.

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



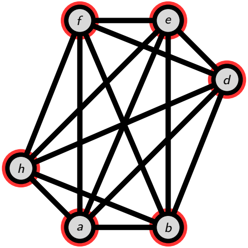
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T.
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.

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



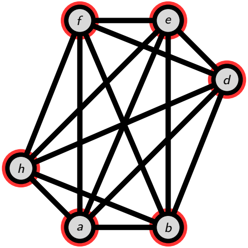
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T.
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.

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



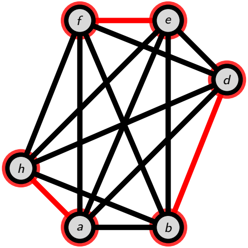
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T.
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2

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



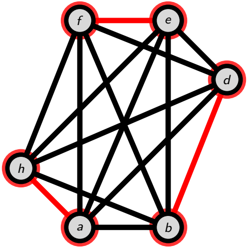
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T.
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2
- $c(M) \leqslant c(M_1)$ und $c(M) \leqslant c(M_2)$.

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



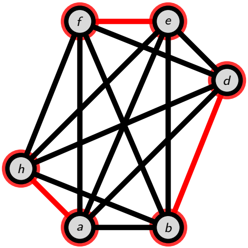
- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T.
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2
- $c(M) \leqslant c(M_1)$ und $c(M) \leqslant c(M_2)$.
- $2 \cdot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

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



- 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
- $c(M) \leqslant c(M_1)$ und $c(M) \leqslant c(M_2)$.
- $2 \cdot c(M) \leqslant c(M_1) + c(M_2) = c(C'') \leqslant c(C^*)$
- $c(M) \leqslant c(C^*)/2$

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



- 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
- $c(M) \leqslant c(M_1)$ und $c(M) \leqslant c(M_2)$.
- $2 \cdot c(M) \leqslant c(M_1) + c(M_2) = c(C'') \leqslant c(C^*)$
- $c(M) \leqslant c(C^*)/2$

Zentrumsproblem

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

Zentrumsproblem

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

Zentrumsproblem

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

Zentrumsproblem

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

Zentrumsproblem

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

Zentrumsproblem

- 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 (Zentrumsproblem (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 $\text{rad}(Z) \leqslant L$.

Zentrumsproblem

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

Definition (Zentrumsproblem (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 $\mathrm{rad}(Z) \leqslant L$.

Zentrumsproblem

- 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 (Zentrumsproblem (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 $\text{rad}(Z) \leqslant L$.

Definition (Zentrumsproblem)

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

Zentrumsproblem

- 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 (Zentrumsproblem (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 $\text{rad}(Z) \leqslant L$.

Definition (Zentrumsproblem)

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

Komplexität

Theorem

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

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationsfaktor 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 Approximationsfaktor 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 Approximationsfaktor 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 $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationsfaktor 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 Approximationsfaktor 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 Approximationsfaktor 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
  - $\mathrm{rad}(D) = 1$.

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationsfaktor 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 Approximationsfaktor 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
  - $\mathrm{rad}(D) = 1$.

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 Approximationsfaktor $r < 2$ 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.

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 $c : E \mapsto \mathbb{Q}^+$.

Zentrumsproblem (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 $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$

Zentrumsproblem (mit Knotenkosten)

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

Zentrumsproblem (mit Knotenkosten)

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

Definition (Zentrumsproblem)

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.

Zentrumsproblem (mit Knotenkosten)

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

Definition (Zentrumsproblem)

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 $R = \mathrm{rad}(Z)$ 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 $R = \mathrm{rad}(Z)$ 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 $R = \mathrm{rad}(Z)$ 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 $\underline{2 \cdot R}$ werden nun als überdeckt betrachtet.
- Achtung: der Algorithmus beinhaltet den Approximationsfaktor!

Idee

- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \mathrm{rad}(Z)$ 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.
- Achtung: der Algorithmus beinhaltet den Approximationsfaktor!
- Teste nun verschiedene Werte R mit Hilfe dieses Greedyverfahrens.

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.
- Achtung: der Algorithmus beinhaltet den Approximationsfaktor!
- Teste nun verschiedene Werte R mit Hilfe dieses Greedyverfahrens.

Approximation

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

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

Approximation

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

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

Approximation

$\text{argmax}\{X\} = x$ **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

Approximation

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

- Algorithmus $\mathit{GreedyZentrum}(G, c, w, R)$:
  1. $Z := \emptyset$ und $U := V$
  2. Solange $U \neq \emptyset$ mache
     1. $z := \text{argmax}\{w(u) \mid u \in U\}$

Approximation

argmax$\{X\} = x$ **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. $z :=$ argmax$\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$

Approximation

$\text{argmax}\{X\} = x$ **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. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot dist(v, z) \leqslant 2 \cdot R\}$

Approximation

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

- Algorithmus $\mathit{GreedyZentrum}(G, c, w, R)$:
  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. $U := U \setminus \{v \in U \mid w(v) \cdot \mathit{dist}(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z

Approximation

$\text{argmax}\{X\} = x$ **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. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot dist(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.

Approximation

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

- Algorithmus $\mathit{GreedyZentrum}(G, c, w, R)$:
  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. $U := U \setminus \{v \in U \mid w(v) \cdot \mathit{dist}(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen.

Approximation

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

- Algorithmus $\mathit{GreedyZentrum}(G, c, w, R)$:
  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. $U := U \setminus \{v \in U \mid w(v) \cdot \mathit{dist}(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen.
- Definiere: $f(R) := |\mathit{GreedyZentrum}(G, c, w, R)|$

Approximation

argmax$\{X\} = x$ **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. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot \mathit{dist}(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen.
- Definiere: $f(R) := |\mathit{GreedyZentrum}(G, c, w, R)|$
  - Beachte $f(R)$ ist aber nicht monoton.

Approximation

$\text{argmax}\{X\} = x$ **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. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot dist(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen.
- Definiere: $f(R) := |GreedyZentrum(G, c, w, R)|$
  - Beachte $f(R)$ ist aber nicht monoton.
- Definiere: $R' = \max_{v \in V} w(v) \cdot \sum_{e \in E} c(e)$

Approximation

$\text{argmax}\{X\} = x$ **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. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot dist(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen.
- Definiere: $f(R) := |\mathit{GreedyZentrum}(G, c, w, R)|$
  - Beachte $f(R)$ ist aber nicht monoton.
- 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

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

- Algorithmus $\mathit{GreedyZentrum}(G, c, w, R)$:
  1. $Z := \emptyset$ und $U := V$
  2. Solange $U \neq \emptyset$ mache
     1. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot \mathit{dist}(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen.
- Definiere: $f(R) := |\mathit{GreedyZentrum}(G, c, w, R)|$
  - Beachte $f(R)$ ist aber nicht monoton.
- 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 $f(0) = n$

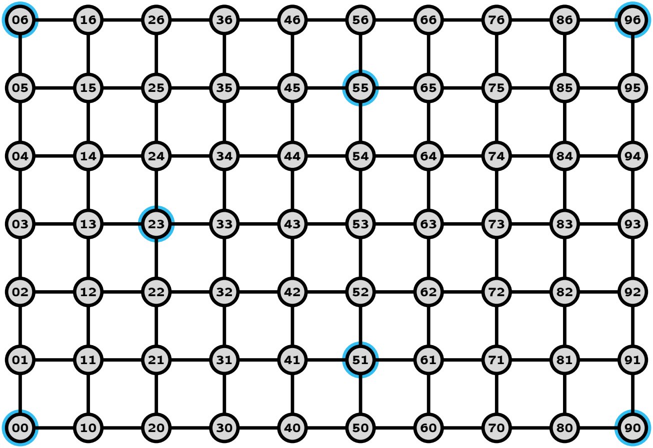
Approximation

$\text{argmax}\{X\} = x$ **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. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot dist(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$.
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen.
- Definiere: $f(R) := |GreedyZentrum(G, c, w, R)|$
  - Beachte $f(R)$ ist aber nicht monoton.
- 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 $f(0) = n$

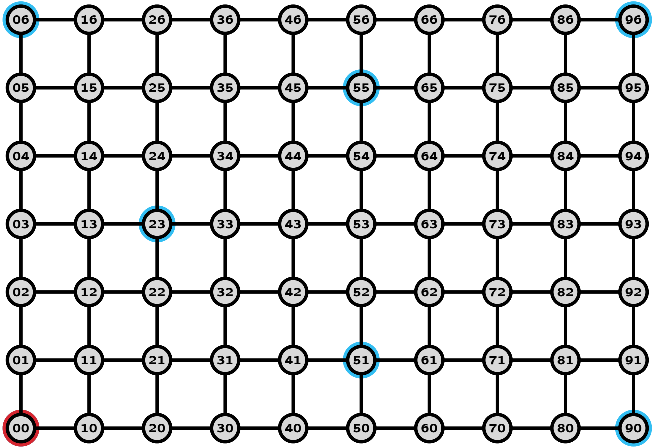
Bildhaftes Beispiel ($R=2$)

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



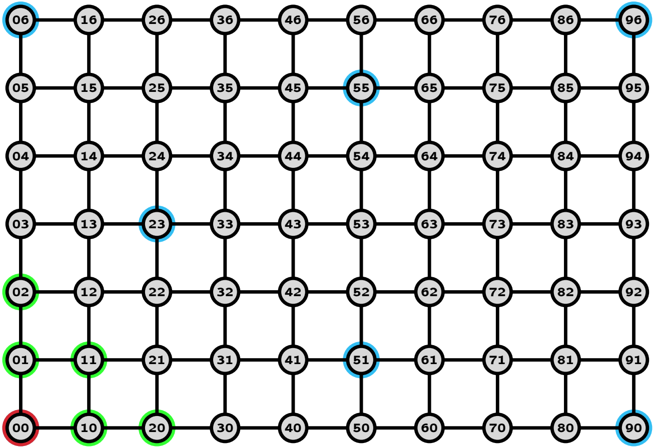
Bildhaftes Beispiel ($R = 2$)

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



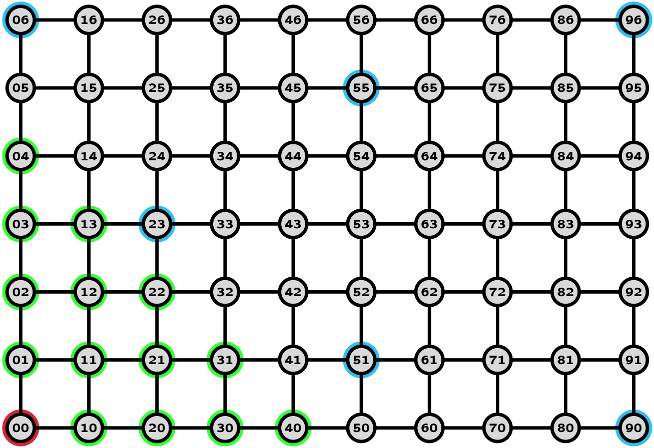
Bildhaftes Beispiel ($R = 2$)

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



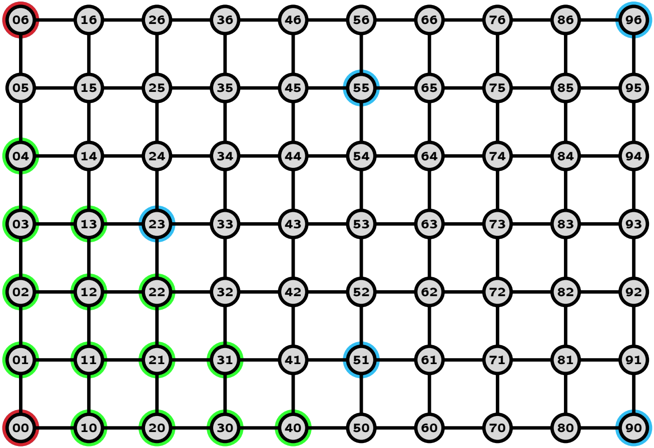
Bildhaftes Beispiel ($R = 2$)

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



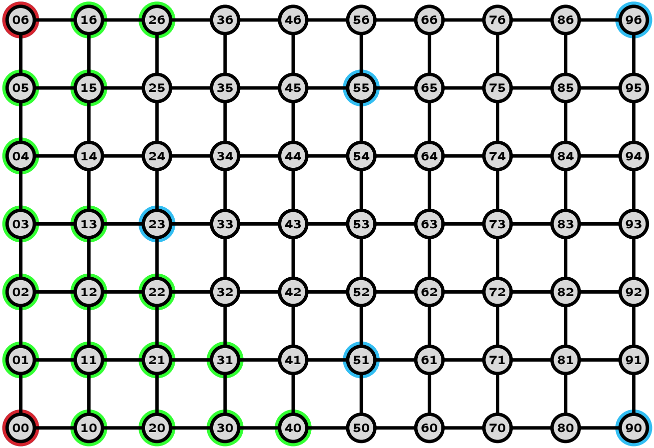
Bildhaftes Beispiel ($R = 2$)

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



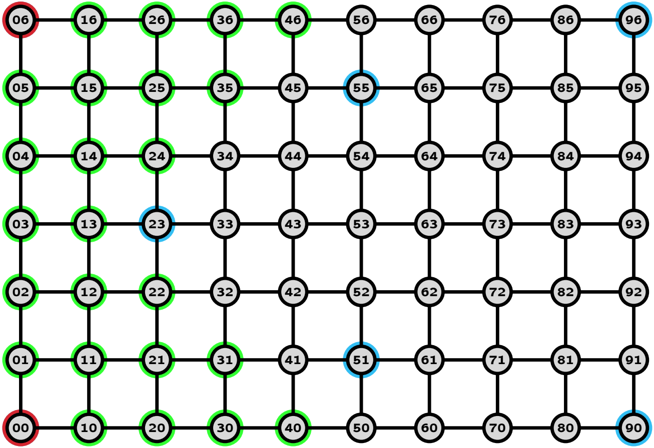
Bildhaftes Beispiel ($R = 2$)

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



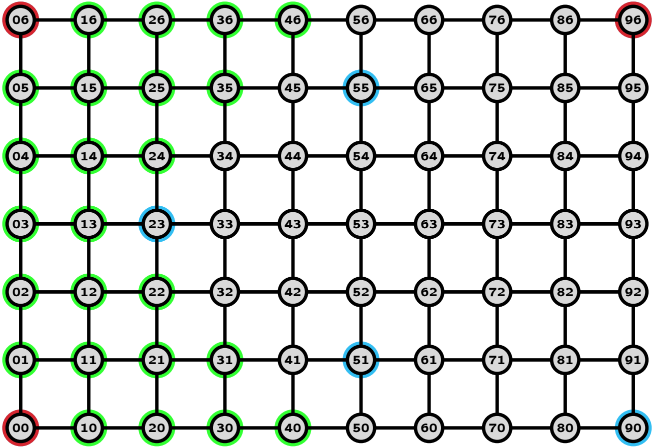
Bildhaftes Beispiel ($R = 2$)

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



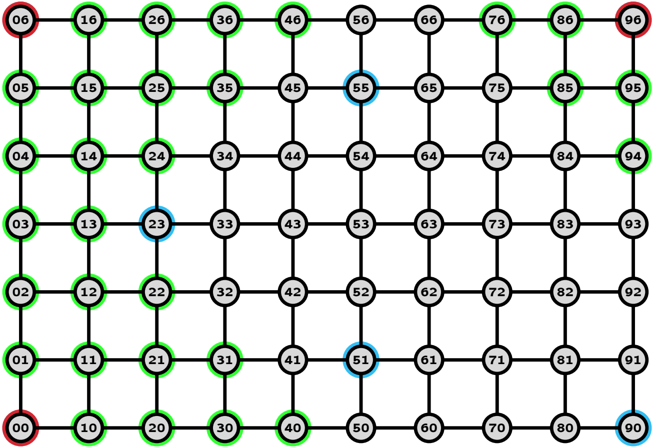
Bildhaftes Beispiel ($R = 2$)

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



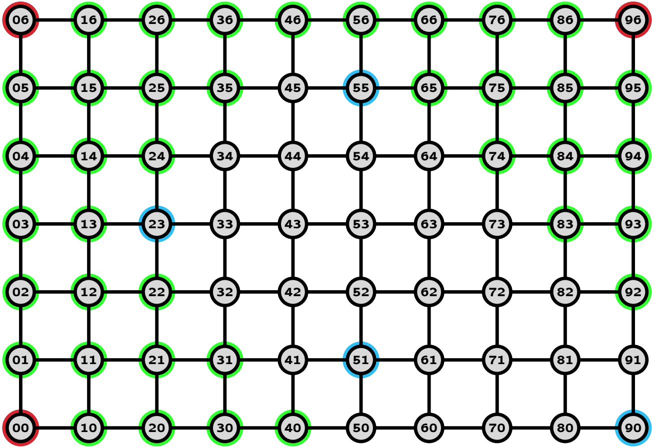
Bildhaftes Beispiel ($R = 2$)

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



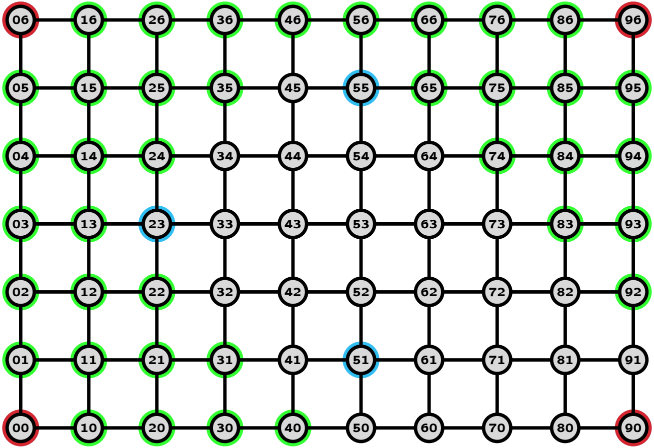
Bildhaftes Beispiel ($R = 2$)

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



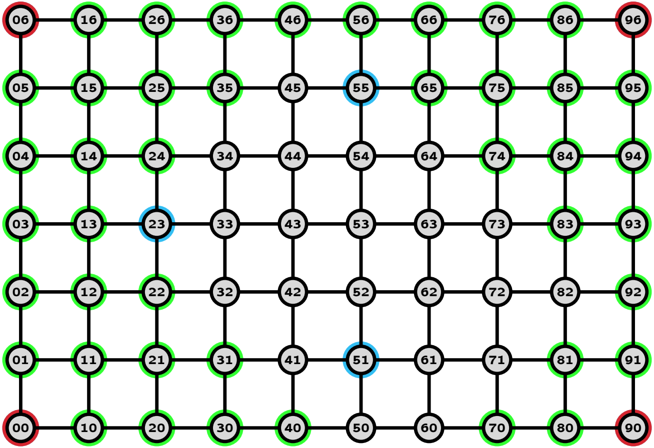
Bildhaftes Beispiel ($R = 2$)

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



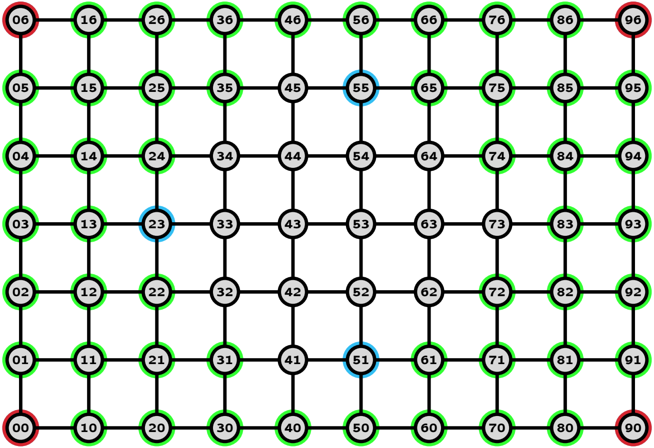
Bildhaftes Beispiel ($R = 2$)

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



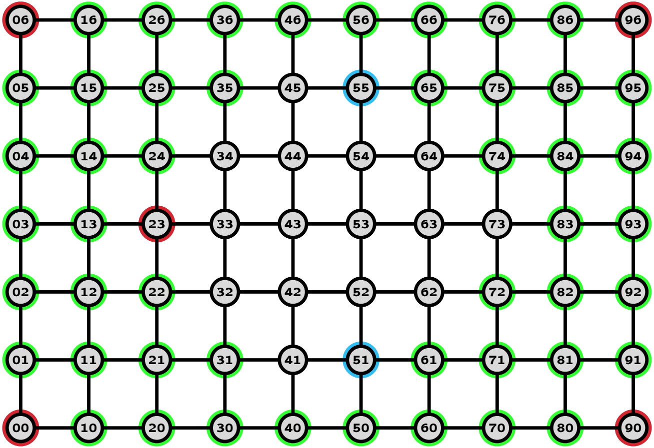
Bildhaftes Beispiel ($R = 2$)

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



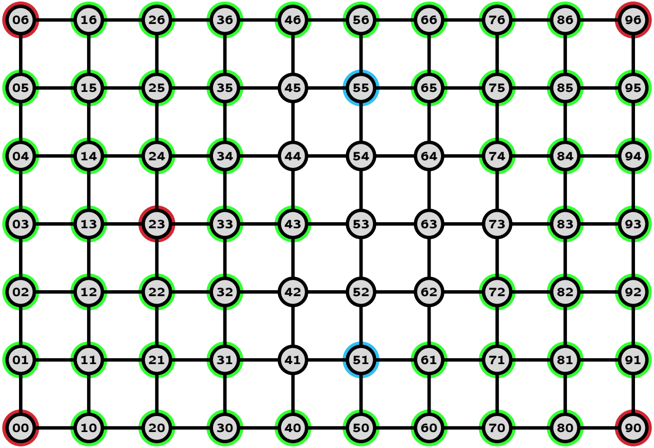
Bildhaftes Beispiel ($R = 2$)

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



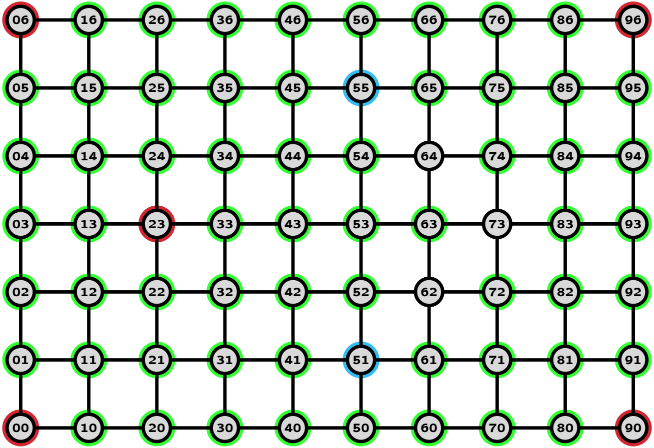
Bildhaftes Beispiel ($R = 2$)

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



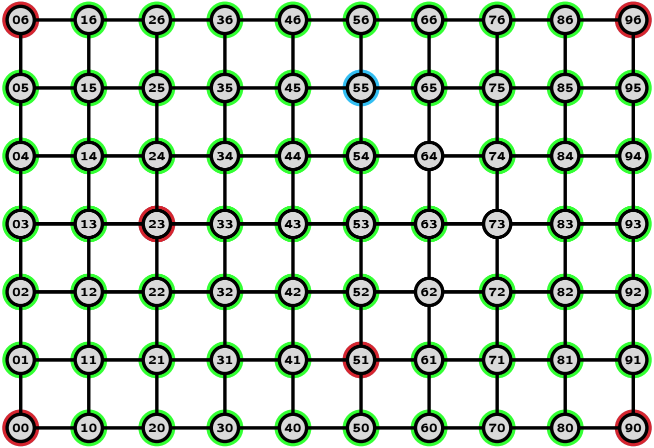
Bildhaftes Beispiel ($R = 2$)

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



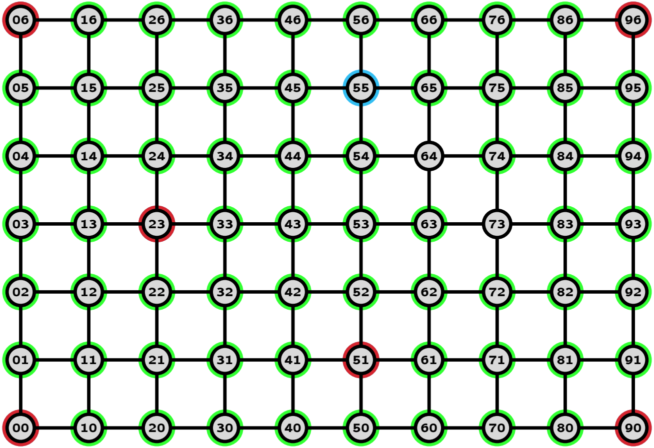
Bildhaftes Beispiel ($R = 2$)

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



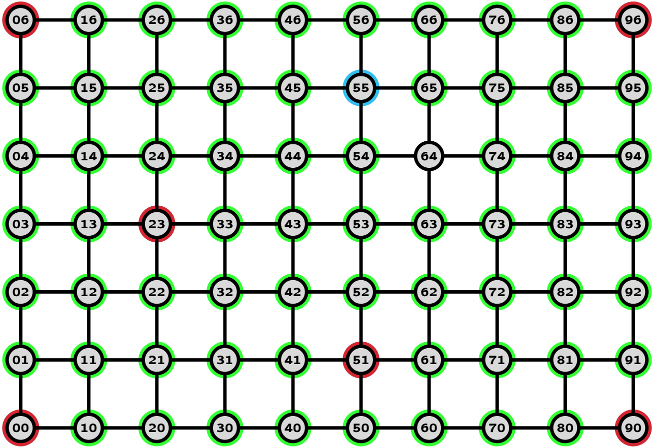
Bildhaftes Beispiel ($R = 2$)

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



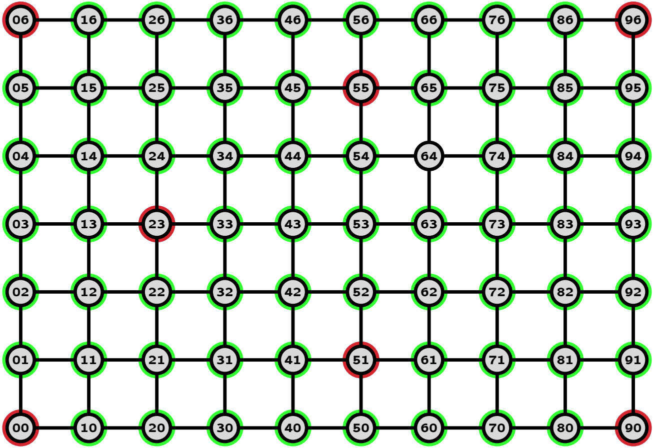
Bildhaftes Beispiel ($R = 2$)

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



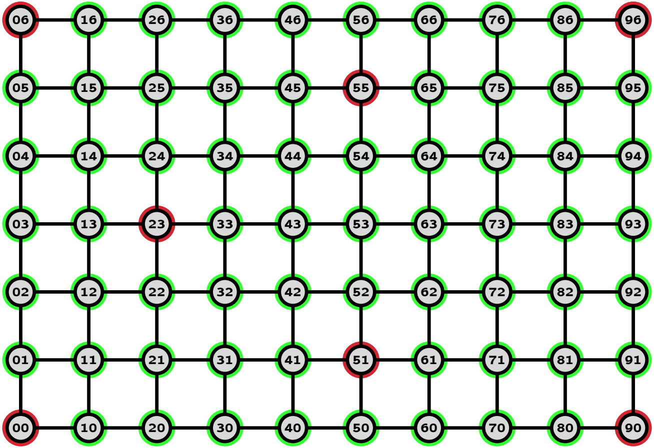
Bildhaftes Beispiel ($R = 2$)

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



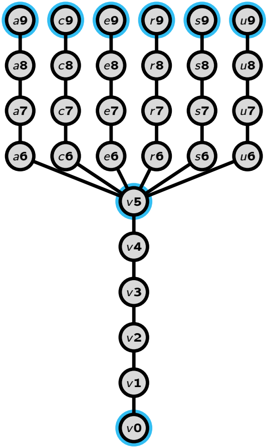
Bildhaftes Beispiel ($R = 2$)

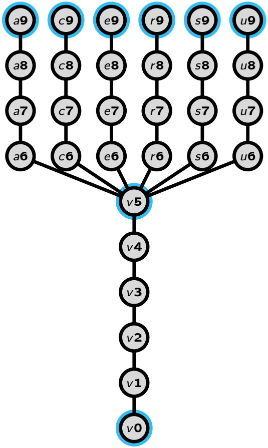
$$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$$



Beispiel für $f(R)$ ist nicht monoton

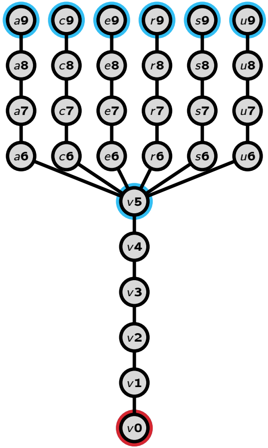
Bestimme $f(2) =$

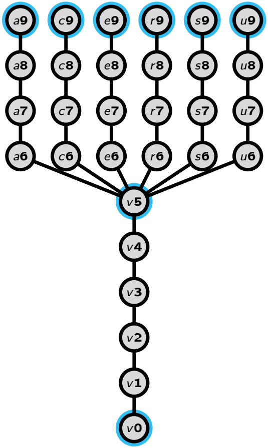




Beispiel für $f(R)$ ist nicht monoton

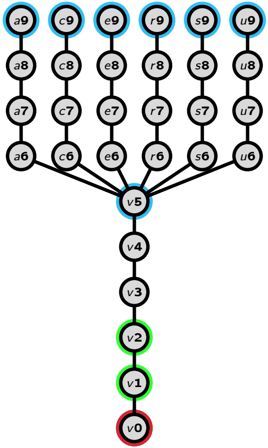
Bestimme $f(2) =$

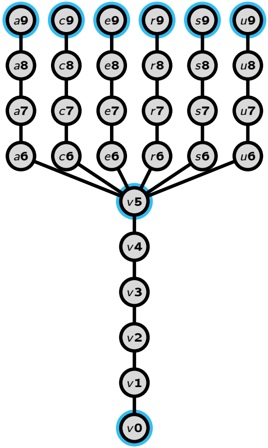




Beispiel für $f(R)$ ist nicht monoton

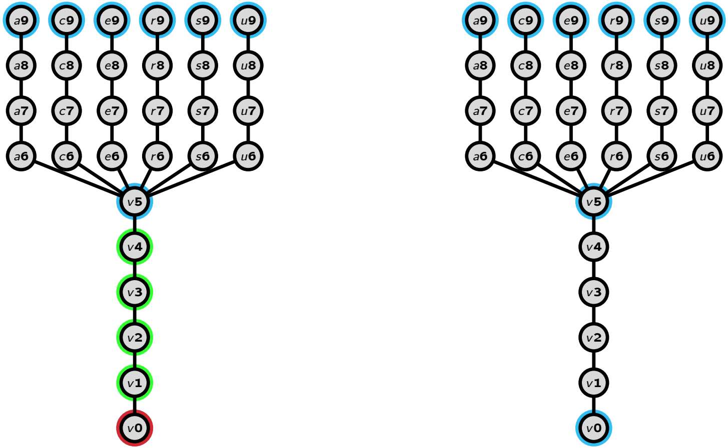
Bestimme $f(2) =$





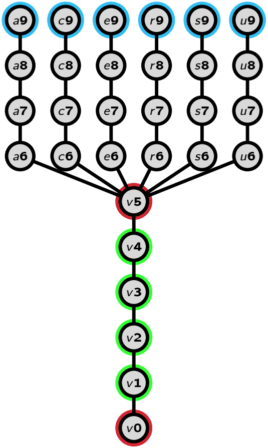
Beispiel für $f(R)$ ist nicht monoton

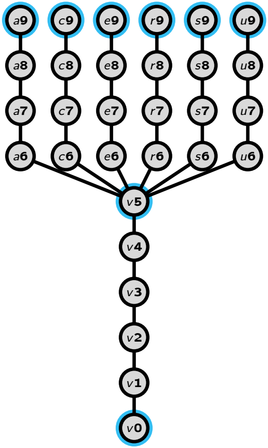
Bestimme $f(2) =$



Beispiel für $f(R)$ ist nicht monoton

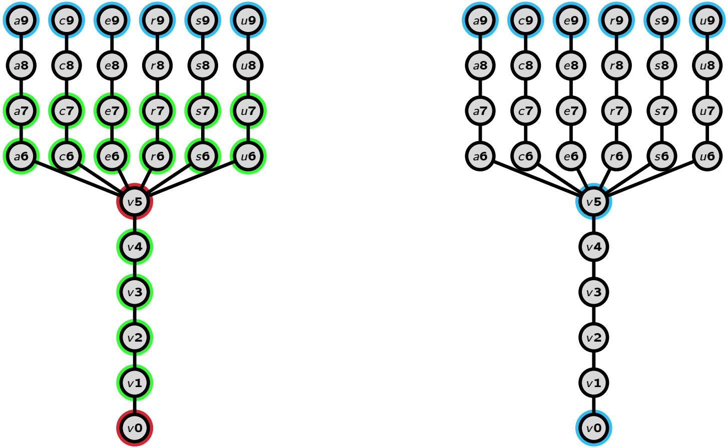
Bestimme $f(2) =$





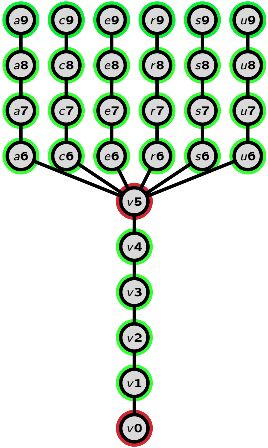
Beispiel für $f(R)$ ist nicht monoton

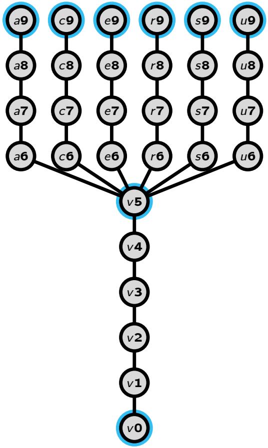
Bestimme $f(2) =$



Beispiel für $f(R)$ ist nicht monoton

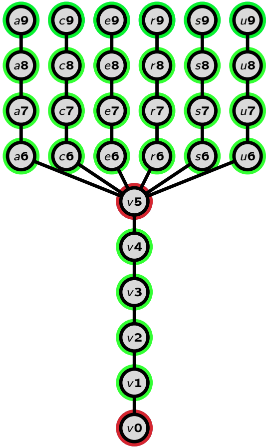
Bestimme $f(2) =$



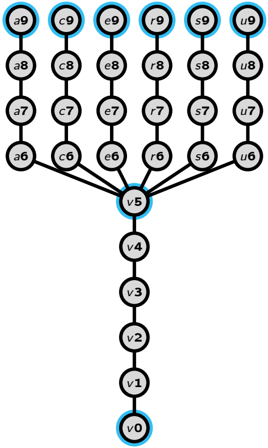


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

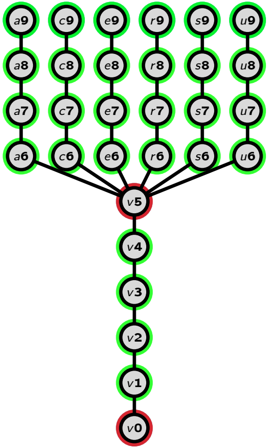


Bestimme $f(3) =$

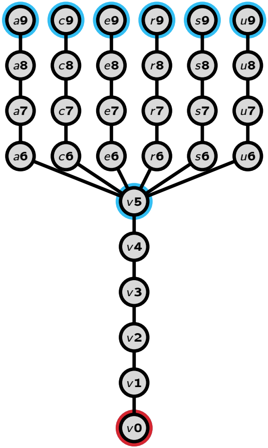


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

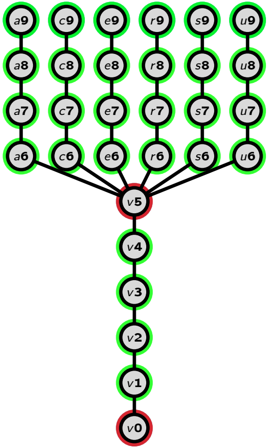


Bestimme $f(3) =$

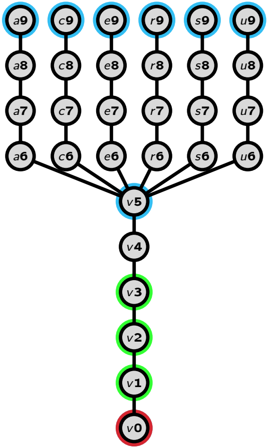


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

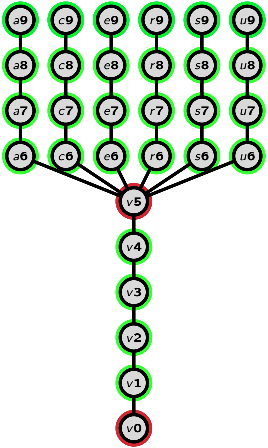


Bestimme $f(3) =$

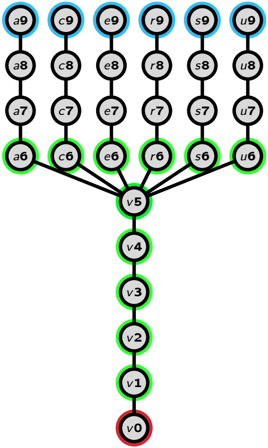


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

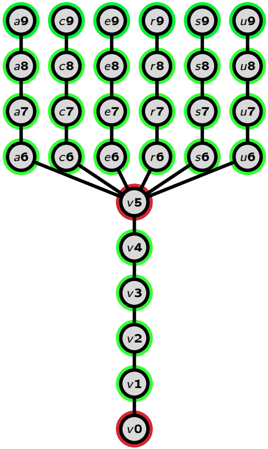


Bestimme $f(3) =$

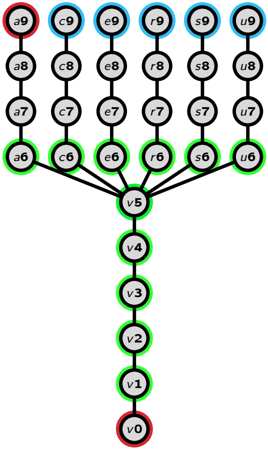


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

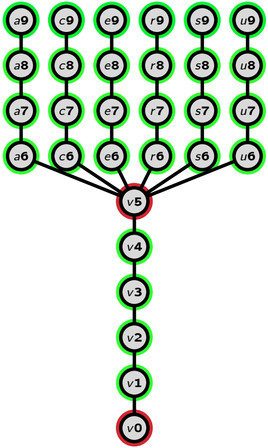


Bestimme $f(3) =$

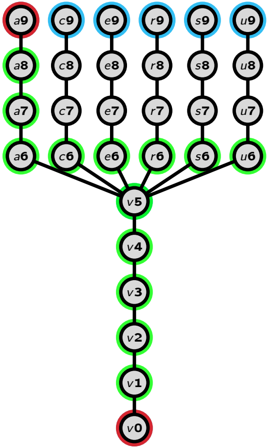


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

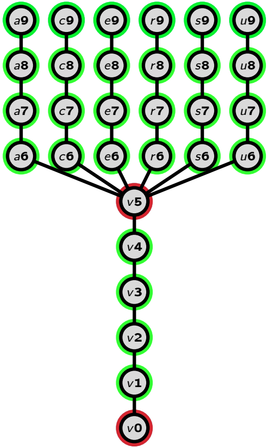


Bestimme $f(3) =$

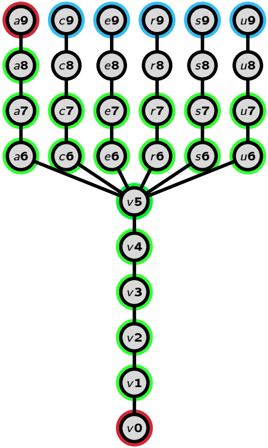


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

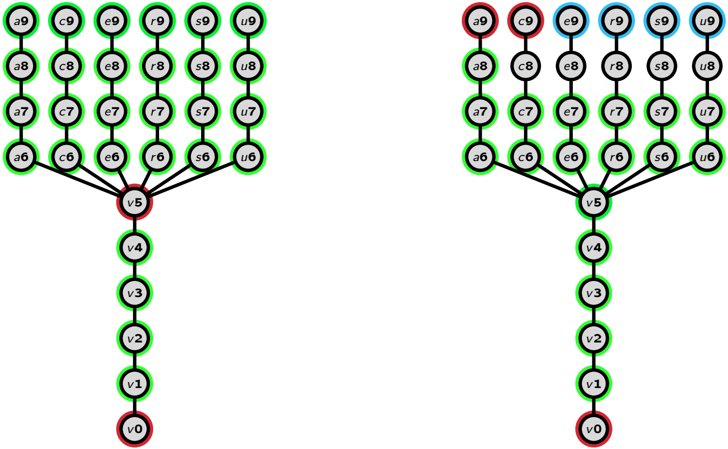


Bestimme $f(3) =$



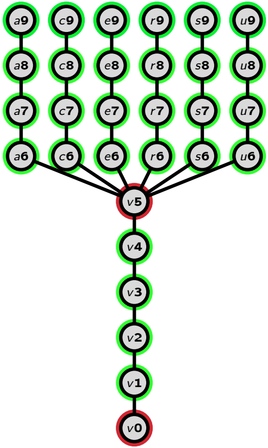
Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$ Bestimme $f(3) =$

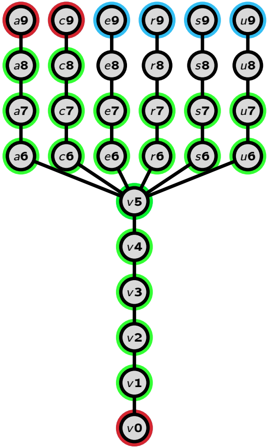


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

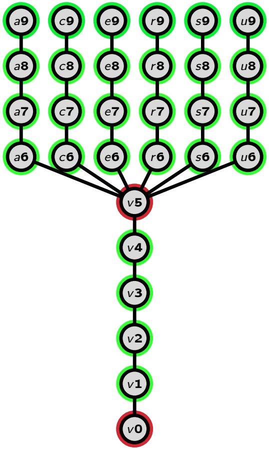


Bestimme $f(3) =$

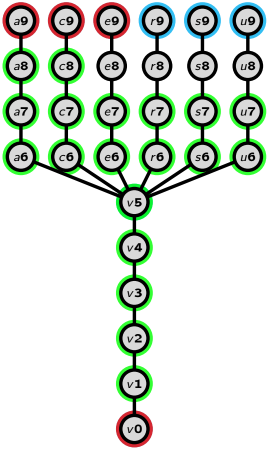


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

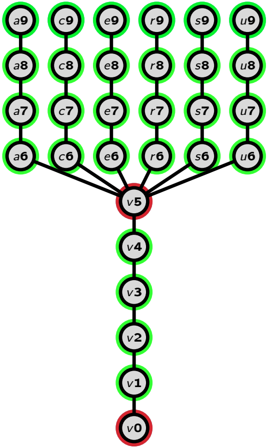


Bestimme $f(3) =$

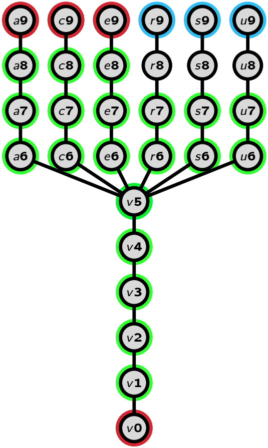


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

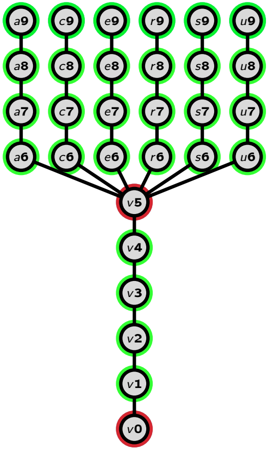


Bestimme $f(3) =$

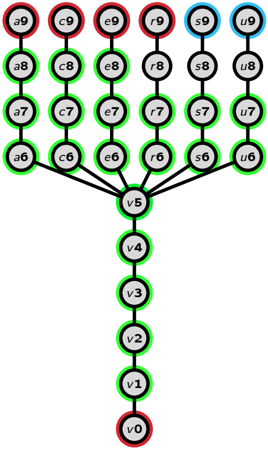


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

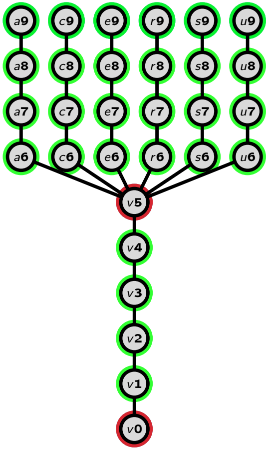


Bestimme $f(3) =$

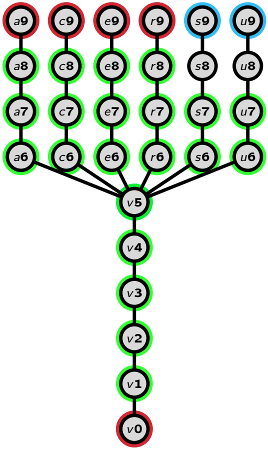


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

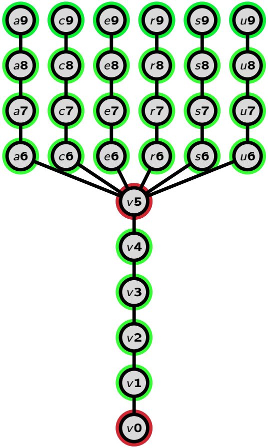


Bestimme $f(3) =$

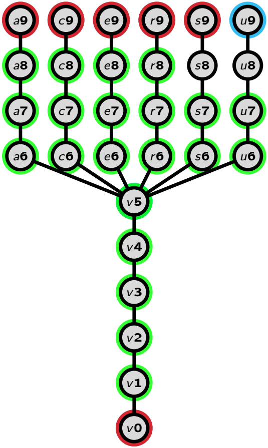


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

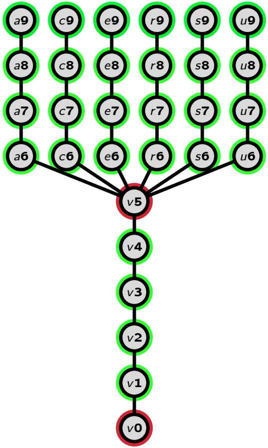


Bestimme $f(3) =$

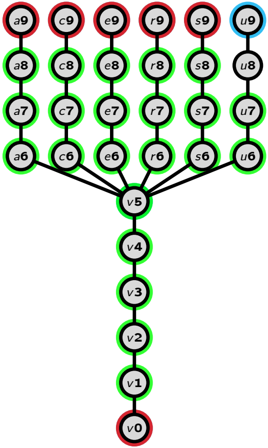


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

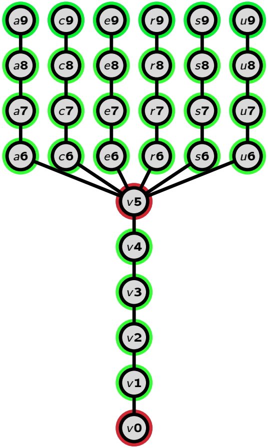


Bestimme $f(3) =$

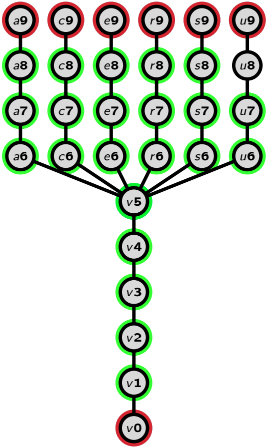


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

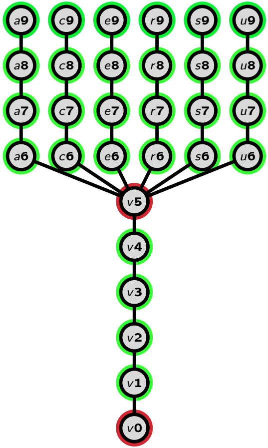


Bestimme $f(3) =$

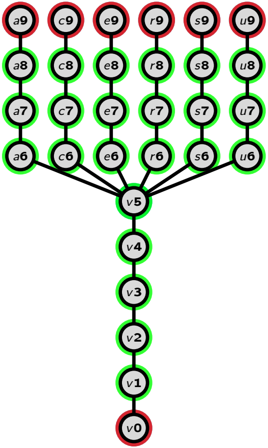


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$

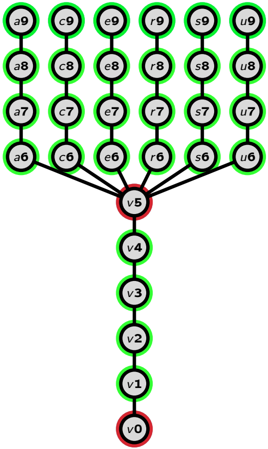


Bestimme $f(3) =$

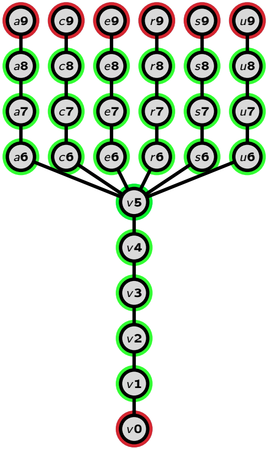


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$



Bestimme $f(3) = 7$



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \mathrm{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Approximation

Lemma

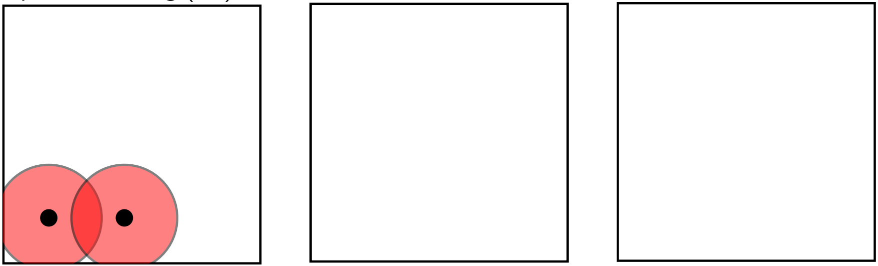
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):

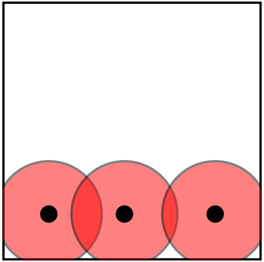


Approximation

Lemma

Sei $G=(V,E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):

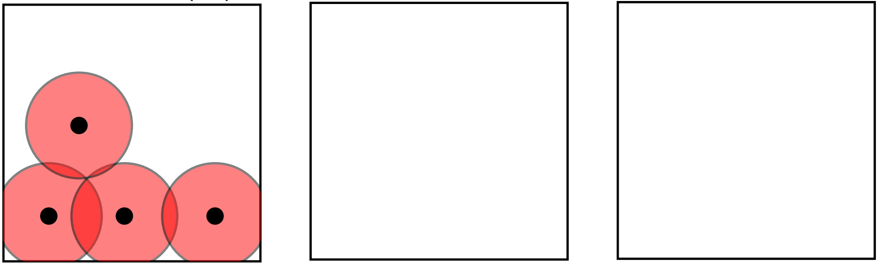


Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):

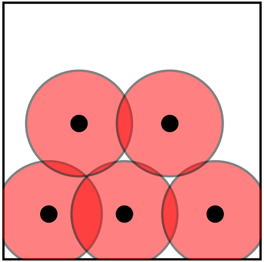


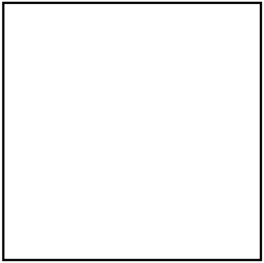
Approximation

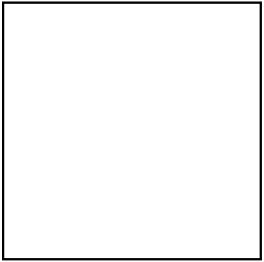
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):





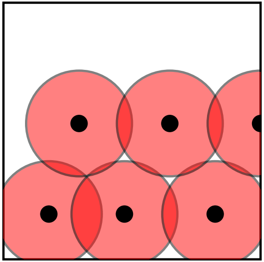


Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):

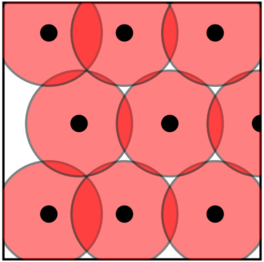


Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):



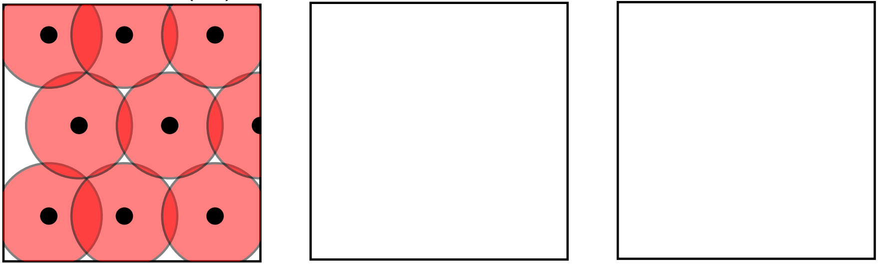
Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):

Approximation:

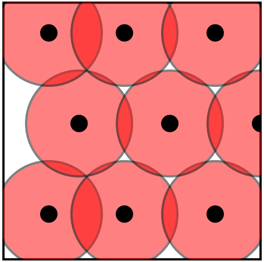


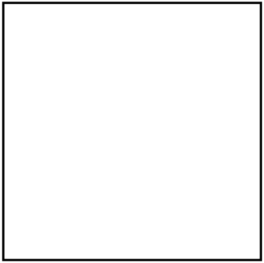
Approximation

Lemma

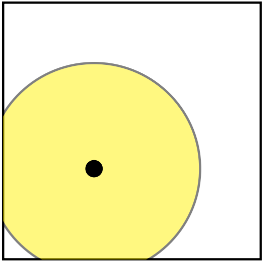
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):





Approximation:

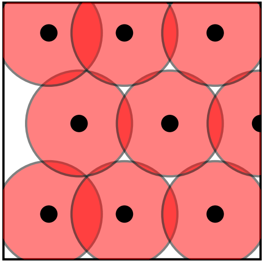


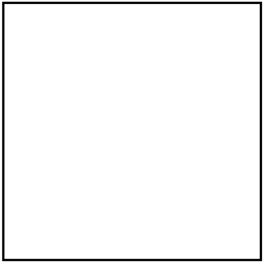
Approximation

Lemma

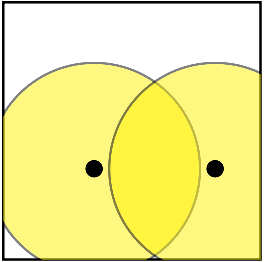
Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \mathrm{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):





Approximation:

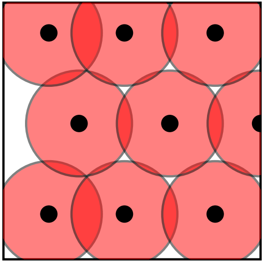


Approximation

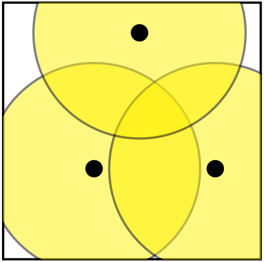
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

Optimale Lösung (R^*):



Approximation:

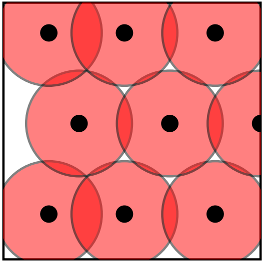


Approximation

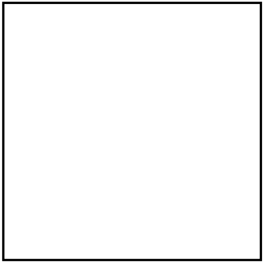
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \mathrm{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

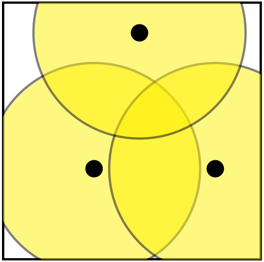
Optimale Lösung (R^*):



Vergleich:



Approximation:

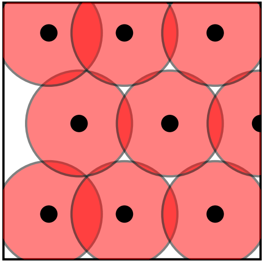


Approximation

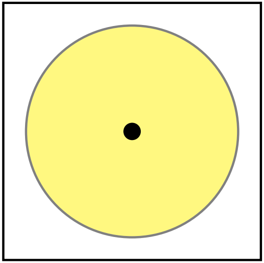
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \mathrm{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

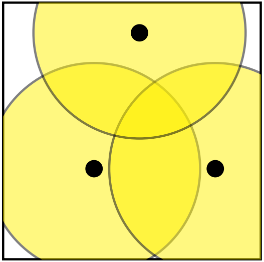
Optimale Lösung (R^*):



Vergleich:



Approximation:

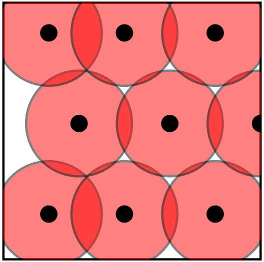


Approximation

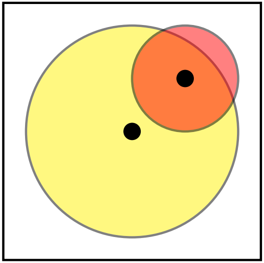
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \mathrm{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

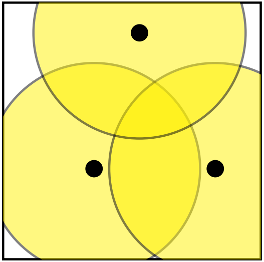
Optimale Lösung (R^*):



Vergleich:



Approximation:

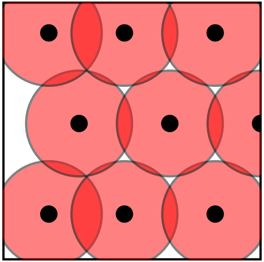


Approximation

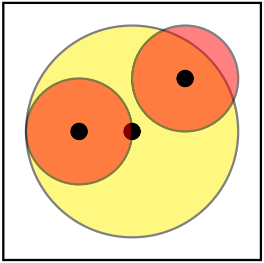
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

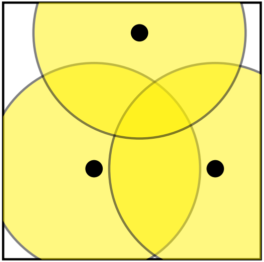
Optimale Lösung (R^*):



Vergleich:



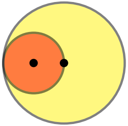
Approximation:



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

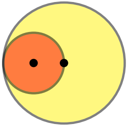


Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.

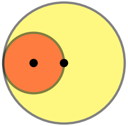


Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.

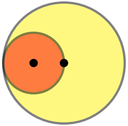


Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

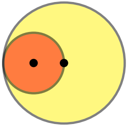
- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.



- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

$$w(v) \cdot dist(v, z) \quad \leqslant \quad w(v) \cdot (dist(v, v_i) + dist(v_i, z))$$

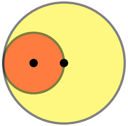
Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

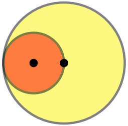
$$w(v) \cdot dist(v, z) \quad \leqslant \quad w(v) \cdot (dist(v, v_i) + dist(v_i, z))$$



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.



- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

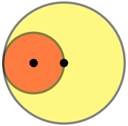
$$\begin{aligned} w(v) \cdot dist(v, z) &\leqslant w(v) \cdot (dist(v, v_i) + dist(v_i, z)) \\ &\leqslant w(v) \cdot dist(v, v_i) + w(z) \cdot dist(v_i, z) \end{aligned}$$

Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \mathrm{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

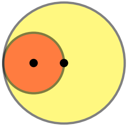


$$\begin{array}{rcl} w(v) \cdot dist(v, z) & \leqslant & w(v) \cdot (dist(v, v_i) + dist(v_i, z)) \\ & \leqslant & w(v) \cdot dist(v, v_i) + w(z) \cdot dist(v_i, z) \\ & \leqslant & 2 \cdot R^* \end{array}$$

Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.



- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

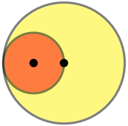
$$\begin{aligned} w(v) \cdot dist(v, z) &\leqslant w(v) \cdot (dist(v, v_i) + dist(v_i, z)) \\ &\leqslant w(v) \cdot dist(v, v_i) + w(z) \cdot dist(v_i, z) \\ &\leqslant 2 \cdot R^* \\ &\leqslant 2 \cdot R \end{aligned}$$

- 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 $R^* = \mathrm{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.



- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

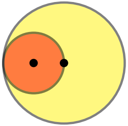
$$\begin{array}{rcl} w(v) \cdot dist(v, z) & \leqslant & w(v) \cdot (dist(v, v_i) + dist(v_i, z)) \\ & \leqslant & w(v) \cdot dist(v, v_i) + w(z) \cdot dist(v_i, z) \\ & \leqslant & 2 \cdot R^* \\ & \leqslant & 2 \cdot R \end{array}$$

- Alle Knoten aus V_i sind am Ende des Schleifendurchlaufes aus U gelöscht.
- Schleife terminiert nach höchstens $k = |Z^*|$ Runden. Und es gilt: $|Z| \leqslant k$.

Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.



- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

$$
\begin{array}{rcl}
w(v) \cdot dist(v, z) & \leqslant & w(v) \cdot (dist(v, v_i) + dist(v_i, z)) \\
& \leqslant & w(v) \cdot dist(v, v_i) + w(z) \cdot dist(v_i, z) \\
& \leqslant & 2 \cdot R^* \\
& \leqslant & 2 \cdot R
\end{array}
$$

- Alle Knoten aus V_i sind am Ende des Schleifendurchlaufes aus U gelöscht.
- Schleife terminiert nach höchstens $k = |Z^*|$ Runden. Und es gilt: $|Z| \leqslant k$.

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:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- Algorithmus $ApproxZentrum(G,c,w,k)$

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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 $R_1 < R_2 < \ldots < R_{anz}$ diese Werte.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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 $R_1 < R_2 < \ldots < R_{anz}$ diese Werte.
  4. $i := 0$, $Z = V$.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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 $R_1 < R_2 < \ldots < R_{anz}$ diese Werte.
  4. $i := 0$, $Z = V$.
  5. Solange $|Z| > k$ führe aus:

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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 $R_1 < R_2 < \ldots < R_{anz}$ 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:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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 $R_1 < R_2 < \ldots < R_{anz}$ 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:
  - $\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u, v) \mid u, v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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 $R_1 < R_2 < \ldots < R_{anz}$ 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

- Es werden nun die alle möglichen Distanzwerte untersucht:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- 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 $R_1 < R_2 < \ldots < R_{anz}$ 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 Zentrumsproblem 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 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.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$

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.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$

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.
- D.h. der Durchlauf mit Ausgabe Z mit $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.

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.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:

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.
- D.h. der Durchlauf mit Ausgabe Z mit $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
    - Distanzmatrix: $O(n^3)$.

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.
- D.h. der Durchlauf mit Ausgabe Z mit $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
  - Distanzmatrix: $O(n^3)$.
  - Sortierung: $O(n^2 \log n)$.

Approximation

Theorem

Das 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.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- 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 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.
- D.h. der Durchlauf mit Ausgabe Z mit $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\mathrm{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
  - Distanzmatrix: $O(n^3)$.
  - Sortierung: $O(n^2 \log n)$.
  - *GreedyZentrum*: $O(n^2)$.
- Gesamtlaufzeit $O(n^4)$.

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.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
  - Distanzmatrix: $O(n^3)$.
  - Sortierung: $O(n^2 \log n)$.
  - *GreedyZentrum*: $O(n^2)$.
- Gesamtlaufzeit $O(n^4)$.

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: Approximationsalgorithmen – 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 Intractibility, Freeman and Company, 1979.

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 Intractibility, Freeman and Company, 1979.