

Algorithmische Graphentheorie

Sommersemester 2021

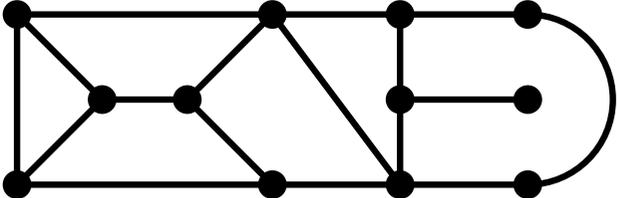
6. Vorlesung

Matchings / Paarungen II

- Kombinatorischer Algorithmus, Anwendung für Handlungsreisende, LP-Runden –

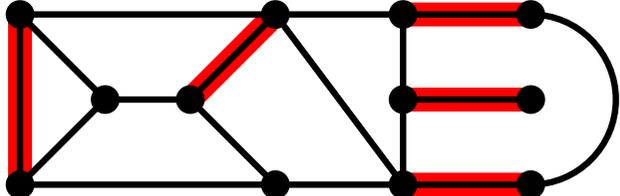
Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph

Alternierende und augmentierende Wege

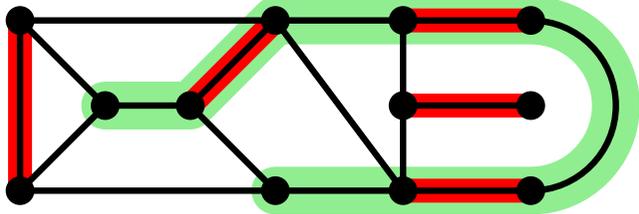
Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

Alternierende und augmentierende Wege

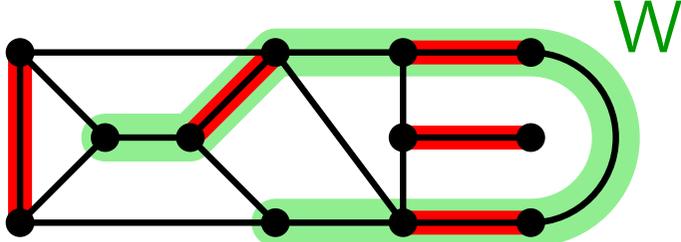
Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

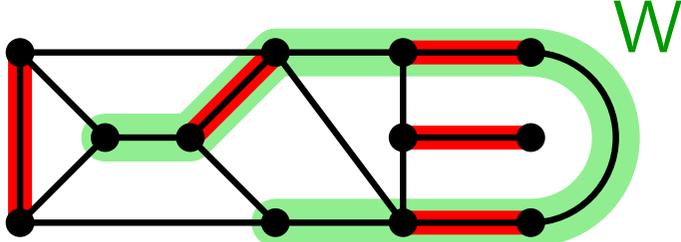
Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

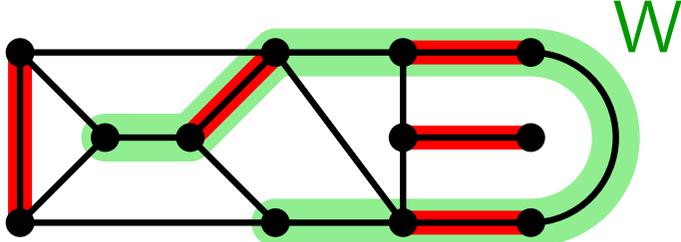
Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) alternierenden Weg W

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

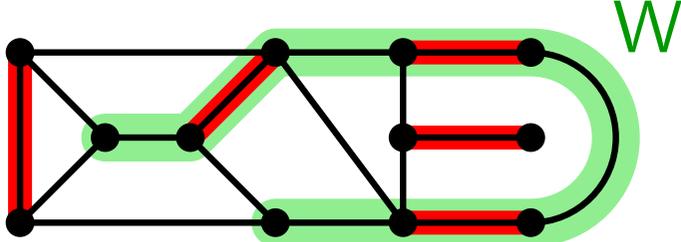
Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

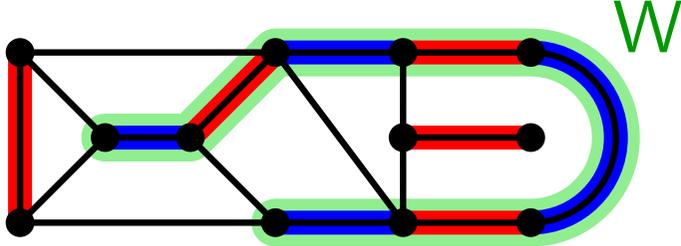
Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

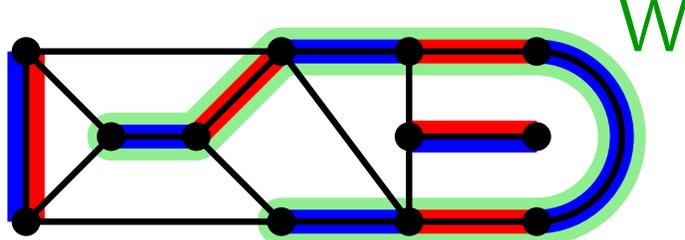
Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

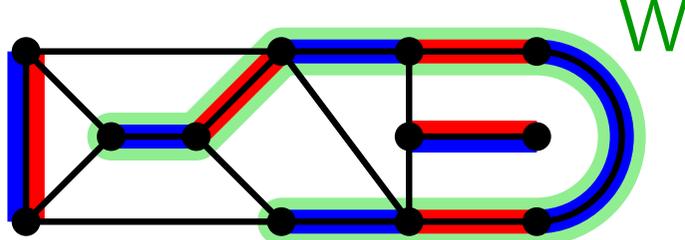
Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

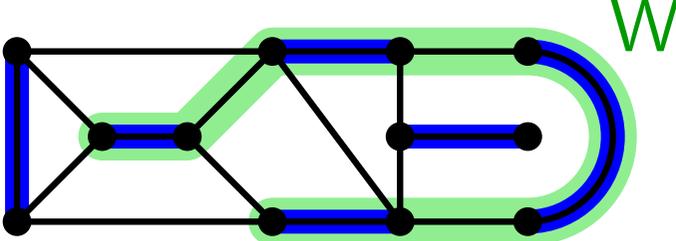
Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.
- Setze $M' = W \Delta M$, wobei $A \Delta B = (A \cup B) \setminus (A \cap B)$ die *symm. Differenz* von A und B ist.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

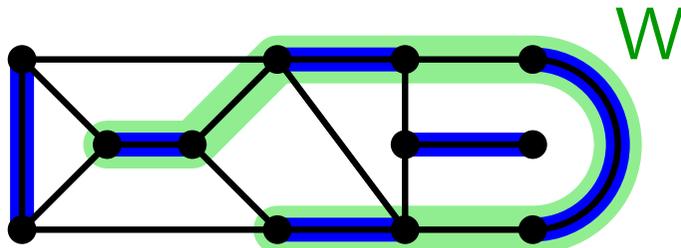
Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.
- Setze $M' = W \Delta M$, wobei $A \Delta B = (A \cup B) \setminus (A \cap B)$ die *symm. Differenz* von A und B ist.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

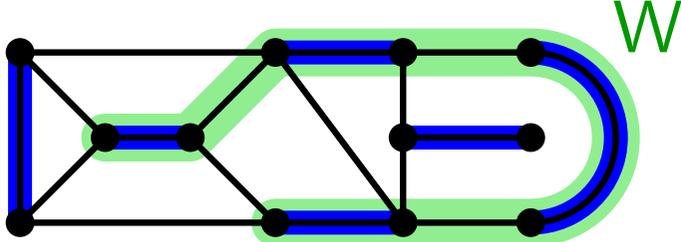
enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.
- Setze $M' = W \Delta M$, wobei $A \Delta B = (A \cup B) \setminus (A \cap B)$ die *symm. Differenz* von A und B ist.

Beob. – Augmentierende Wege haben ungerade Länge.

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

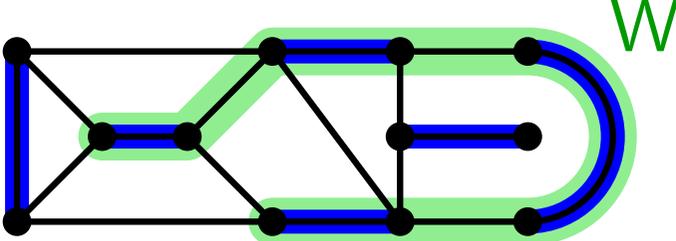
enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.
- Setze $M' = W \Delta M$, wobei $A \Delta B = (A \cup B) \setminus (A \cap B)$ die *symm. Differenz* von A und B ist.

Beob. – Augmentierende Wege haben ungerade Länge.
 – M' ist um 1 Kante größer als M

Alternierende und augmentierende Wege

Ziel: Besseres Problemverständnis \rightarrow kombinatorische (d.h. nicht flussbasierte) Algorithmen für größte Matchings.

Bsp.  $G = (V, E)$ unger. Graph
 $M \subseteq E$ Matching in G .

Wie können wir ein gegebenes (nicht-größtes) Matching vergrößern?

enthält abwechselnd M - und nicht- M -Kanten

- Finde einen (M -) *alternierenden Weg* W , dessen Endknoten M -frei sind. So ein Weg heißt (M -) *augmentierend*.
- Setze $M' = W \Delta M$, wobei $A \Delta B = (A \cup B) \setminus (A \cap B)$ die *symm. Differenz* von A und B ist.

Beob. – Augmentierende Wege haben ungerade Länge.
 – M' ist um 1 Kante größer als M
 (da M' zusätzlich die Endknoten von W paart).

Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

Paris, 1990 © Adrian Bondy



Claude Berge
(1926–2002)

Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .
 M ist ein größtes Matching in G
 \Leftrightarrow es gibt keinen M -augmentierenden Weg.

Paris, 1990 © Adrian Bondy



Claude Berge
(1926–2002)

Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .
 M ist ein größtes Matching in G
 \Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar:

Paris, 1990 © Adrian Bondy



Claude Berge
(1926–2002)

Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .
 M ist ein größtes Matching in G
 \Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡

Paris, 1990 © Adrian Bondy



Claude Berge
(1926–2002)

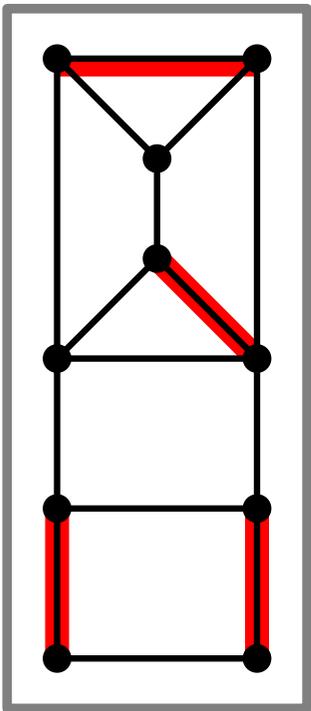
Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

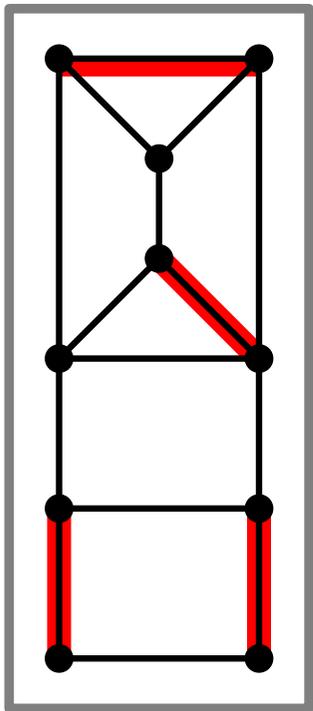
M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡

„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

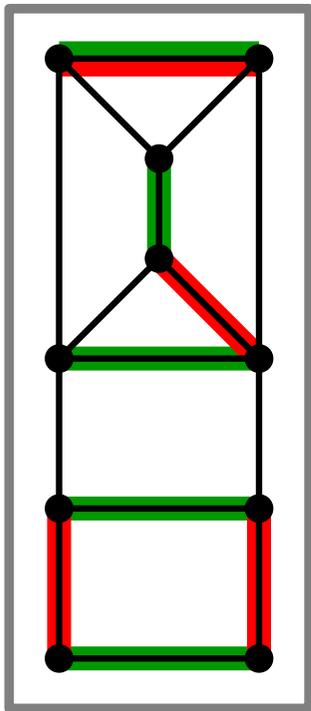
M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡

„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

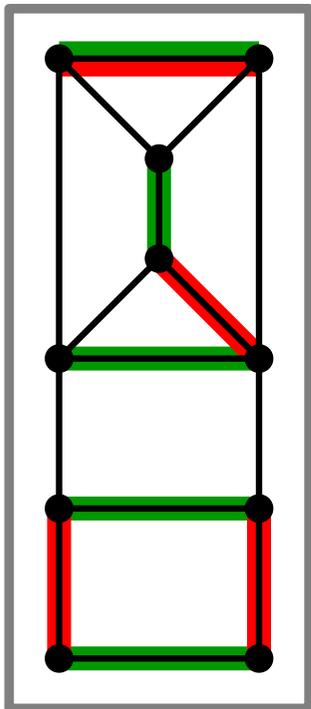
\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡

„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

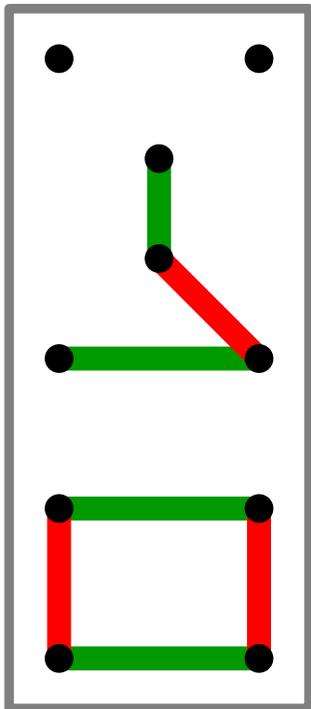
M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

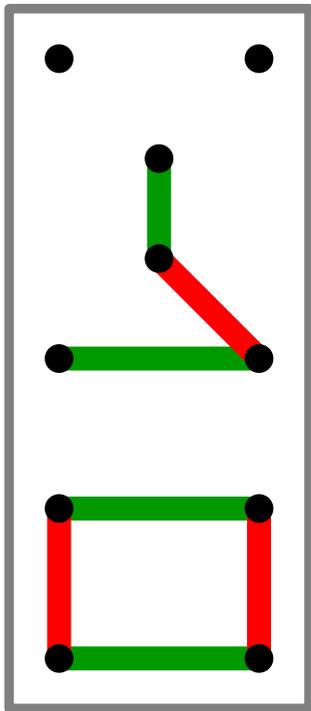
\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.

Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) =$



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

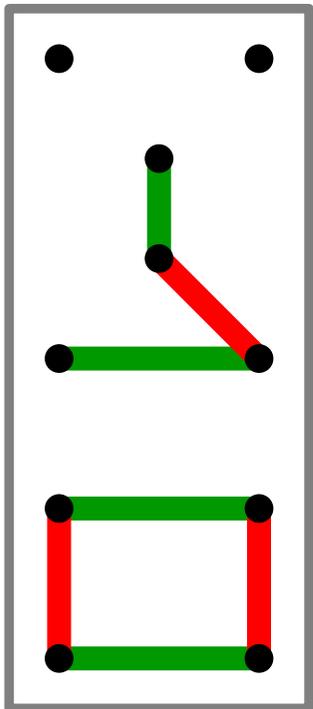
\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.

Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) = 2$.



Claude Berge
(1926–2002)

Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡

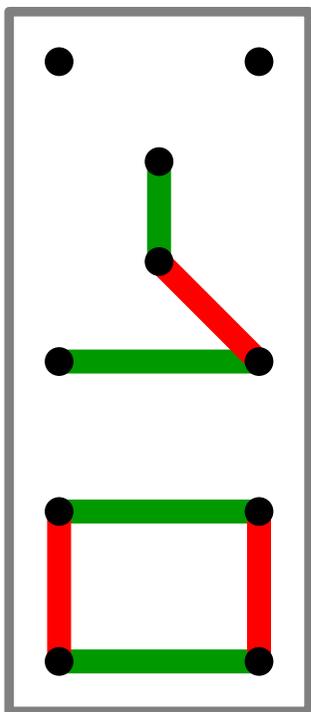
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.

Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) = 2$.

$\Rightarrow G_{\Delta}$ besteht aus einfachen alt. Wegen & Kreisen.



Claude Berge
(1926–2002)

Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

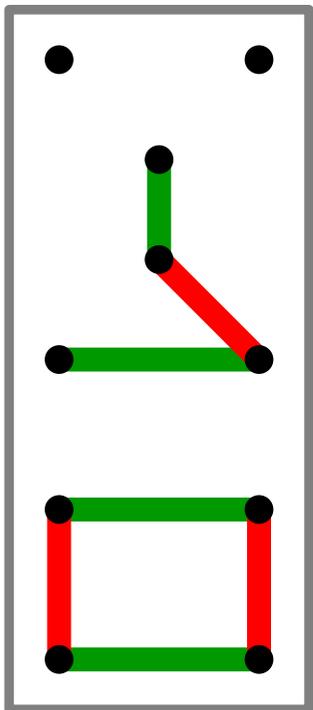
\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.

Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) = 2$.

$\Rightarrow G_{\Delta}$ besteht aus einfachen alt. Wegen & Kreisen.

Alle Kreise in G_{Δ} haben gerade Länge.



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

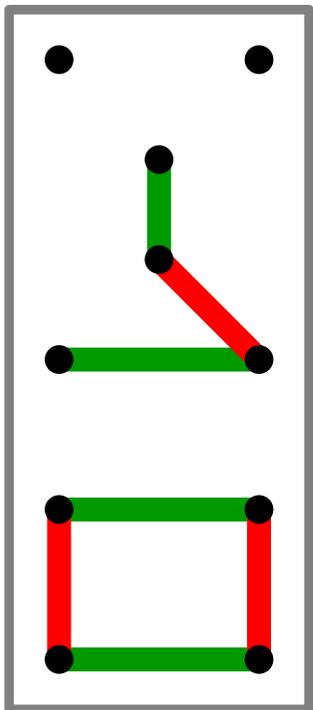
Betrachte $G_{\Delta} = (V, M \Delta M')$.

Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) = 2$.

$\Rightarrow G_{\Delta}$ besteht aus einfachen alt. Wegen & Kreisen.

Alle Kreise in G_{Δ} haben gerade Länge.

\Rightarrow Ex. alt. Weg mit mehr Kanten aus M' als aus M .



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

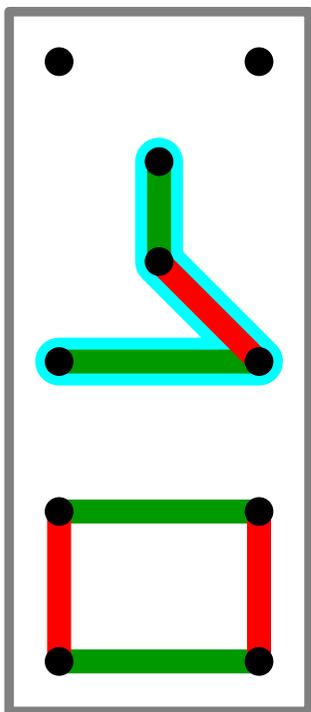
Betrachte $G_{\Delta} = (V, M \Delta M')$.

Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) = 2$.

$\Rightarrow G_{\Delta}$ besteht aus einfachen alt. Wegen & Kreisen.

Alle Kreise in G_{Δ} haben gerade Länge.

\Rightarrow Ex. alt. Weg mit mehr Kanten aus M' als aus M .



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.

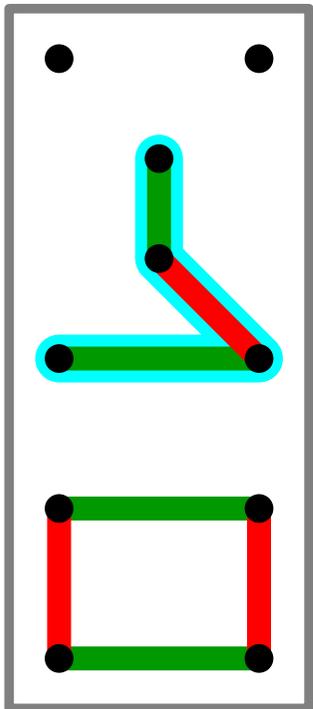
Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) = 2$.

$\Rightarrow G_{\Delta}$ besteht aus einfachen alt. Wegen & Kreisen.

Alle Kreise in G_{Δ} haben gerade Länge.

\Rightarrow Ex. alt. Weg mit mehr Kanten aus M' als aus M .

\Rightarrow Dieser Weg ist M -augment.



Satz von Berge

Satz. Sei $G = (V, E)$ Graph,
 $M \subseteq E$ Matching in G .

M ist ein größtes Matching in G

\Leftrightarrow es gibt keinen M -augmentierenden Weg.

Beweis. „ \Rightarrow “ Klar: jeder augm. Weg würde M vergrößern ⚡
„ \Leftarrow “ Annahme: M ist kein größtes Matching.

\Rightarrow Es gibt ein Matching M' mit $|M'| > |M|$.

Betrachte $G_{\Delta} = (V, M \Delta M')$.

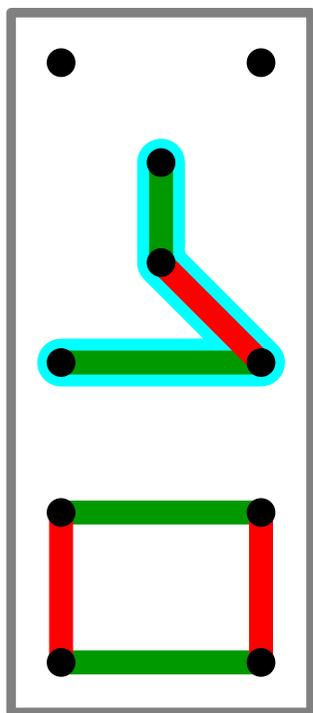
Beobachtung: $\max_{v \in V} \deg_{G_{\Delta}}(v) = 2$.

$\Rightarrow G_{\Delta}$ besteht aus einfachen alt. Wegen & Kreisen.

Alle Kreise in G_{Δ} haben gerade Länge.

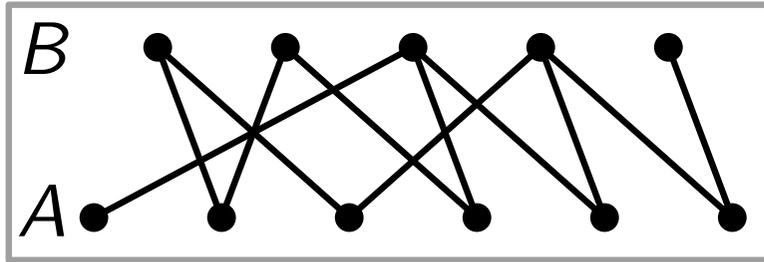
\Rightarrow Ex. alt. Weg mit mehr Kanten aus M' als aus M .

\Rightarrow Dieser Weg ist M -augment. ⚡ zur Annahme. \square



Zurück zu: größte Matchings in bip. Graphen

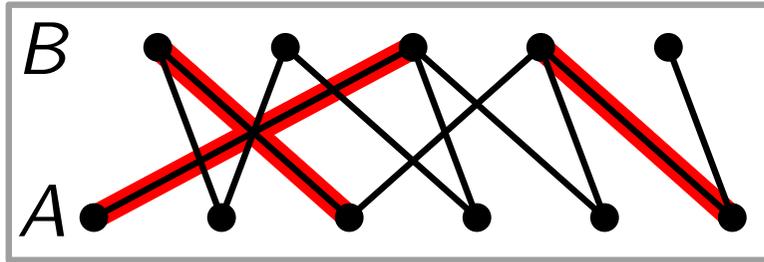
Frage: Wie finden wir augmentierende Wege?



$G = (A \cup B, E)$ bipartit,

Zurück zu: größte Matchings in bip. Graphen

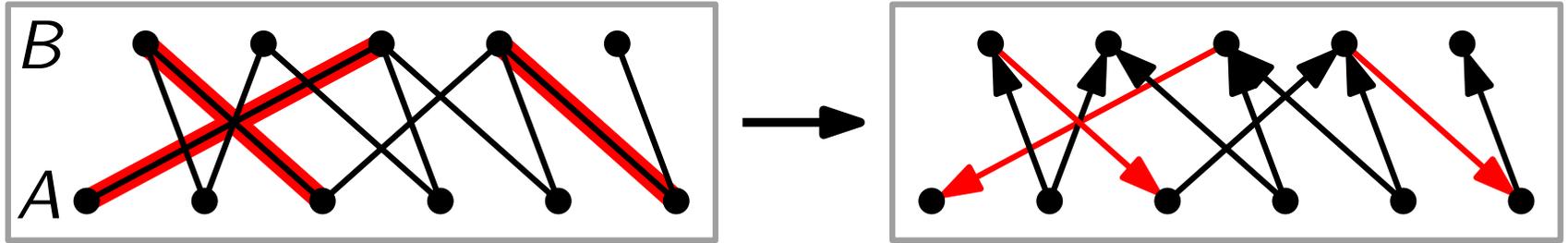
Frage: Wie finden wir augmentierende Wege?



$G = (A \cup B, E)$ bipartit, $M \subseteq E$

Zurück zu: größte Matchings in bip. Graphen

Frage: Wie finden wir augmentierende Wege?

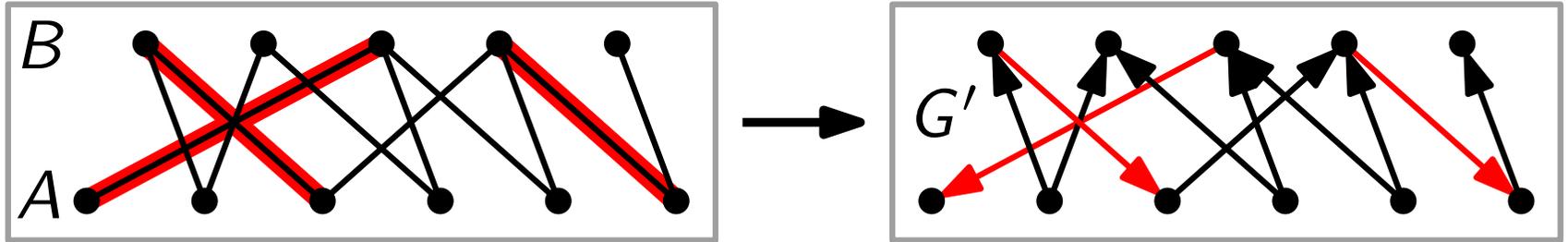


$G = (A \cup B, E)$ bipartit, $M \subseteq E$

Idee: Richte M -Kanten nach unten,
nicht- M -Kanten nach oben.

Zurück zu: größte Matchings in bip. Graphen

Frage: Wie finden wir augmentierende Wege?

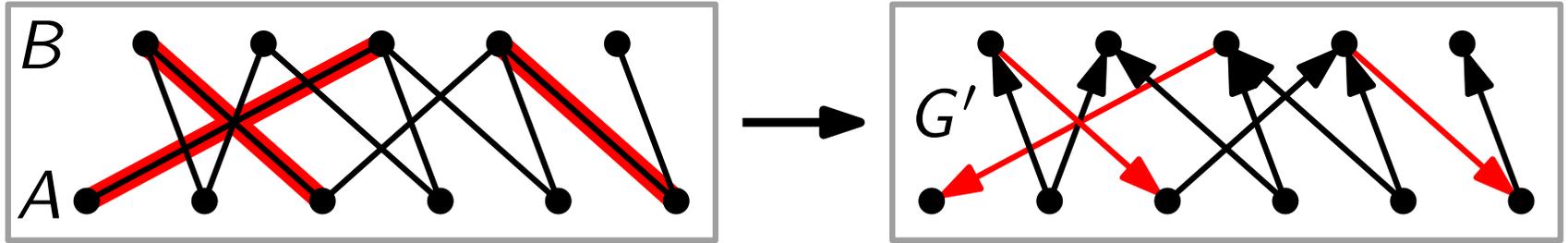


$G = (A \cup B, E)$ bipartit, $M \subseteq E$

Idee: Richte M -Kanten nach unten,
nicht- M -Kanten nach oben. } $\longrightarrow G' = (A \cup B, E')$

Zurück zu: größte Matchings in bip. Graphen

Frage: Wie finden wir augmentierende Wege?



$G = (A \cup B, E)$ bipartit, $M \subseteq E$

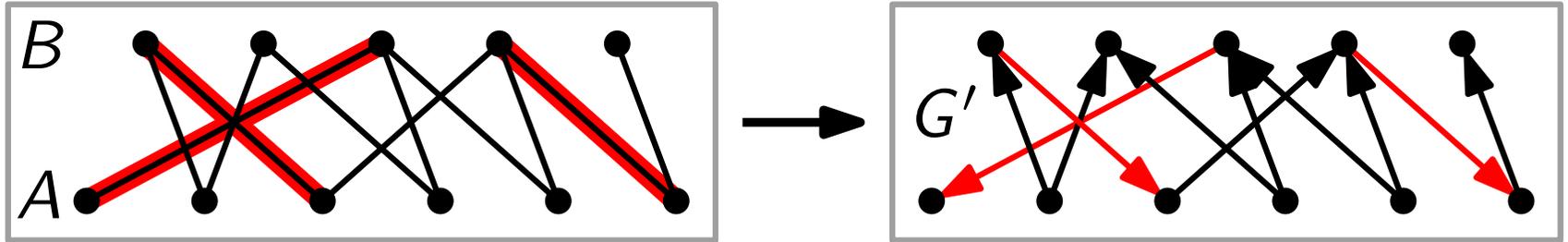
Idee: Richte M -Kanten nach unten,
nicht- M -Kanten nach oben. } $\longrightarrow G' = (A \cup B, E')$

Augmentierende
Wege in G



Zurück zu: größte Matchings in bip. Graphen

Frage: Wie finden wir augmentierende Wege?



$G = (A \cup B, E)$ bipartit, $M \subseteq E$

Idee: Richte M -Kanten nach unten, nicht- M -Kanten nach oben. } $\longrightarrow G' = (A \cup B, E')$

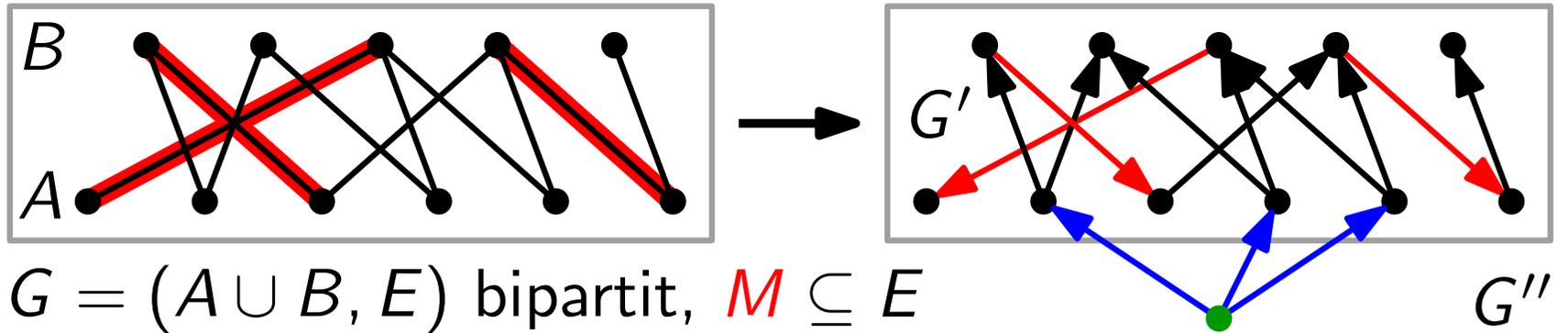
Augmentierende Wege in G



gerichtete Wege mit M -freien Endknoten in G'

Zurück zu: größte Matchings in bip. Graphen

Frage: Wie finden wir augmentierende Wege?



Idee: Richte M -Kanten nach unten, nicht- M -Kanten nach oben. $\} \longrightarrow G' = (A \cup B, E')$

Augmentierende Wege in G

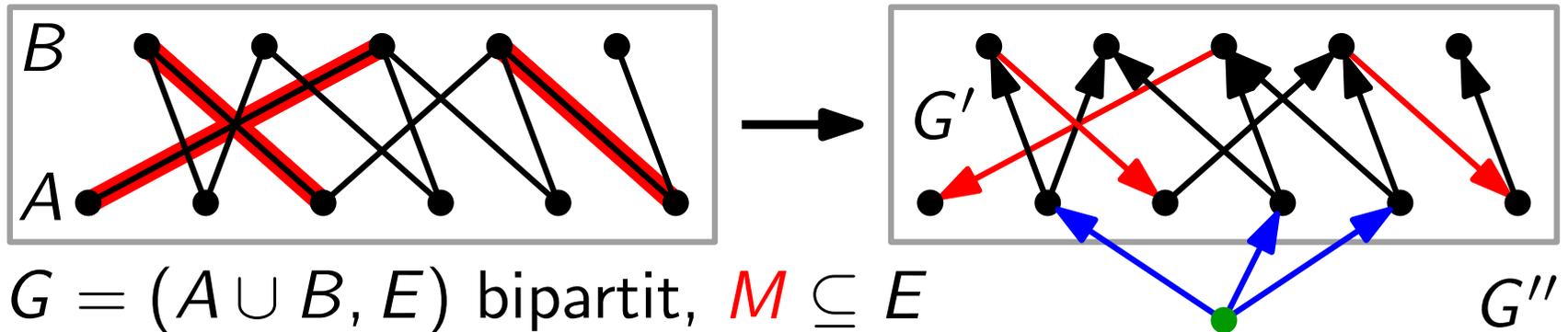


gerichtete Wege mit M -freien Endknoten in G'

Definiere $G'' = (A \cup B \cup \{s\}, E' \cup \{sa \mid a \in A, M\text{-frei}\})$

Zurück zu: größte Matchings in bip. Graphen

Frage: Wie finden wir augmentierende Wege?



Idee: Richte M -Kanten nach unten, nicht- M -Kanten nach oben. $\left. \vphantom{\text{Richte}} \right\} \longrightarrow G' = (A \cup B, E')$

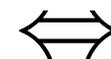
Augmentierende Wege in G



gerichtete Wege mit M -freien Endknoten in G'

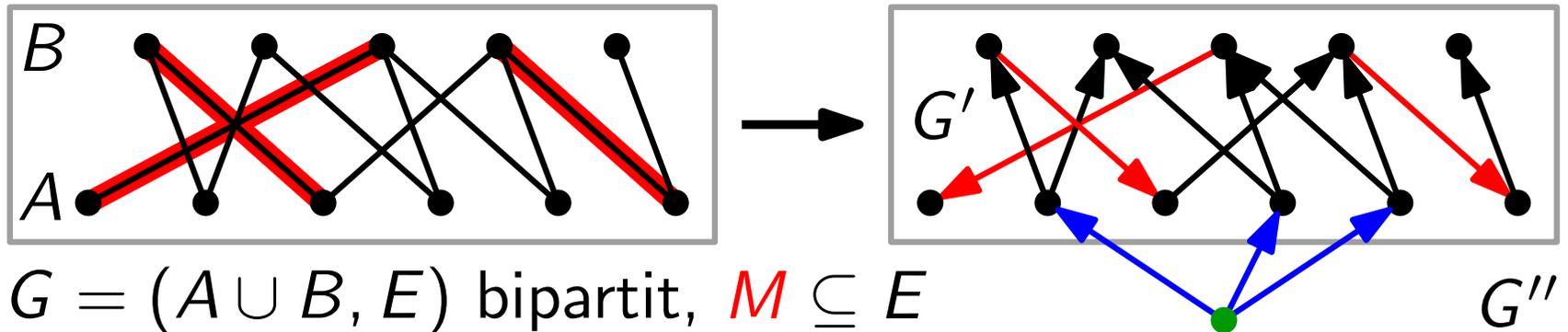
Definiere $G'' = (A \cup B \cup \{s\}, E' \cup \{sa \mid a \in A, M\text{-frei}\})$

Breitensuche $\text{BFS}(G'', s)$ erreicht einen M -freien Endknoten in B



Zurück zu: größte Matchings in bip. Graphen

Frage: Wie finden wir augmentierende Wege?



Idee: Richte M -Kanten nach unten, nicht- M -Kanten nach oben. $\left. \vphantom{\text{Richte}} \right\} \longrightarrow G' = (A \cup B, E')$

Augmentierende Wege in G



gerichtete Wege mit M -freien Endknoten in G'

Definiere $G'' = (A \cup B \cup \{s\}, E' \cup \{sa \mid a \in A, M\text{-frei}\})$

Breitensuche $\text{BFS}(G'', s)$ erreicht einen M -freien Endknoten in B



G hat M -augm. Weg.

Ergebnis

Algo: $M := \emptyset.$

Ergebnis

Algo: $M := \emptyset$.

Führe BFS $\leq |V|/2$ mal aus –

bis kein freier Knoten in B mehr gefunden wird.

Ergebnis

Algo: $M := \emptyset$.

Führe BFS $\leq |V|/2$ mal aus –

 bis kein freier Knoten in B mehr gefunden wird.

Gib größtes Matching zurück.

Ergebnis

Algo: $M := \emptyset$.

Führe BFS $\leq |V|/2$ mal aus –

 bis kein freier Knoten in B mehr gefunden wird.

Gib größtes Matching zurück.

Satz. In einem bipartiten Graphen $G = (V, E)$ lässt sich in $O(VE)$ Zeit ein größtes Matching bestimmen.

Ergebnis

Algo: $M := \emptyset$.

Führe BFS $\leq |V|/2$ mal aus –

– bis kein freier Knoten in B mehr gefunden wird.

Gib größtes Matching zurück.

Satz. In einem bipartiten Graphen $G = (V, E)$ lässt sich in $O(VE)$ Zeit ein größtes Matching bestimmen.

Bem. Dinic' Fluss-Algorithmus berechnet [KN, Kapitel 9.6]

- maximale Flüsse in allg. Graphen in $O(V^2E)$ Zeit
- Matchings in bipartiten Graphen in $O(\sqrt{VE})$ Zeit.

Ergebnis

Algo: $M := \emptyset$.

Führe BFS $\leq |V|/2$ mal aus –

bis kein freier Knoten in B mehr gefunden wird.

Gib größtes Matching zurück.

Satz. In einem bipartiten Graphen $G = (V, E)$ lässt sich in $O(VE)$ Zeit ein größtes Matching bestimmen.

Bem. Dinic' Fluss-Algorithmus berechnet [KN, Kapitel 9.6]
 – maximale Flüsse in allg. Graphen in $O(V^2E)$ Zeit
 – Matchings in bipartiten Graphen in $O(\sqrt{VE})$ Zeit.

Satz. Selbst in einem beliebigen Graphen $G = (V, E)$ lässt sich eine größte Paarung in $O(\sqrt{VE})$ Zeit berechnen.

[Micali & Vazirani, FOCS'80]

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Wiederholung – Tree-Doubling für Δ -TSP

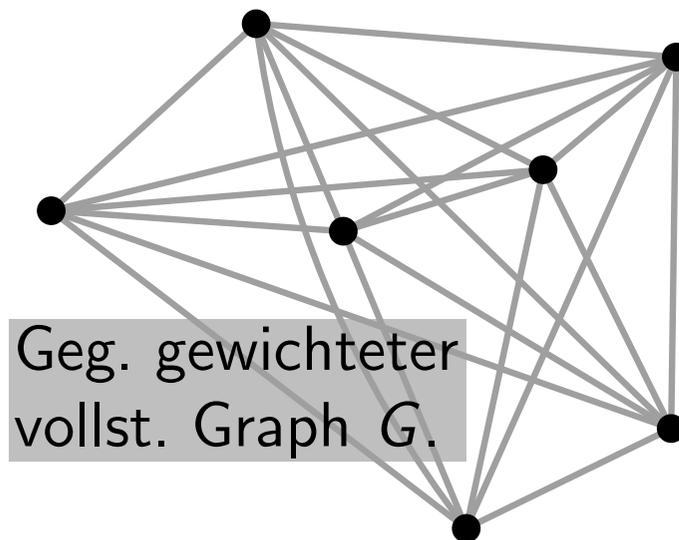
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



Wiederholung – Tree-Doubling für Δ -TSP

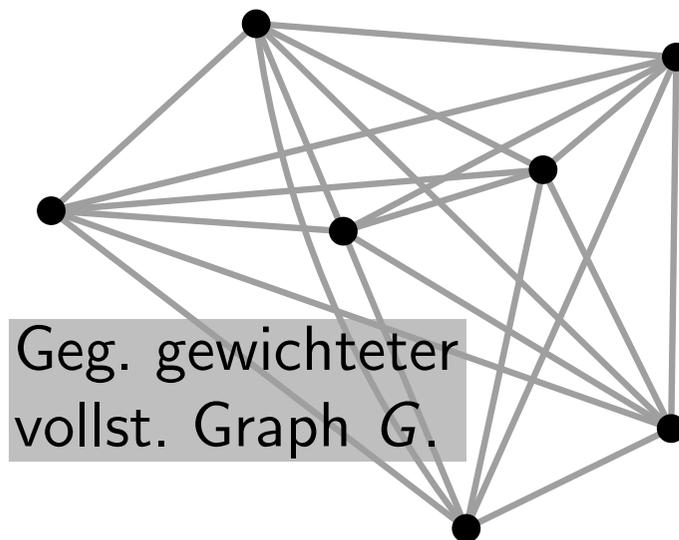
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



Geg. gewichteter
vollst. Graph G .

1. *Algorithmus*

Berechne **MSB** von G .

Wiederholung – Tree-Doubling für Δ -TSP

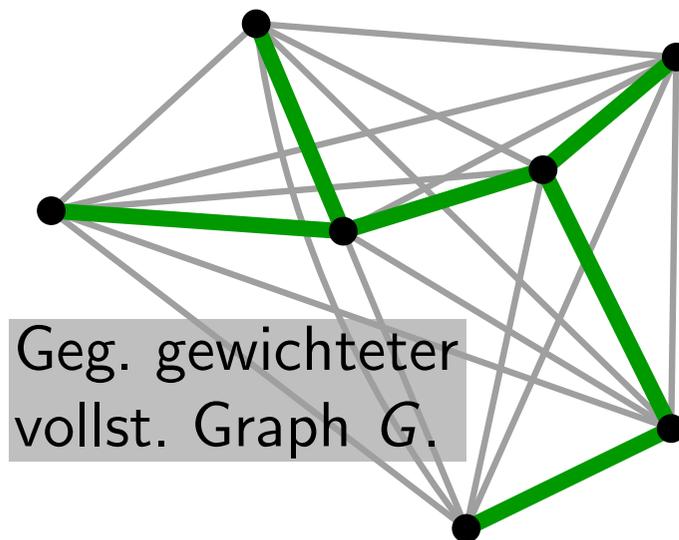
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Wiederholung – Tree-Doubling für Δ -TSP

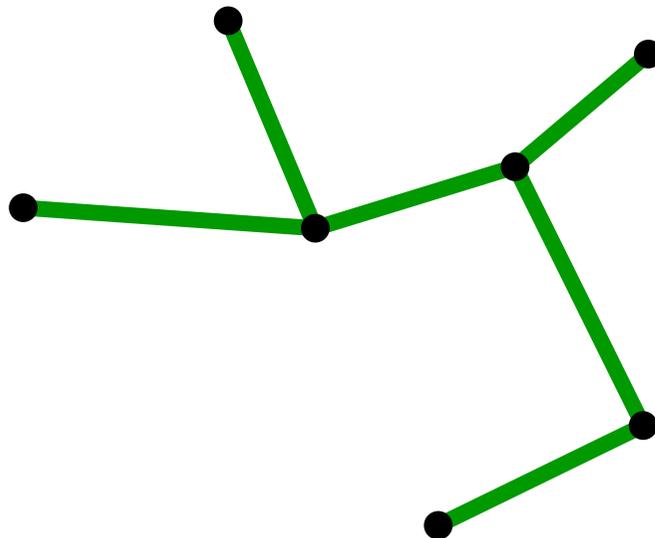
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Wiederholung – Tree-Doubling für Δ -TSP

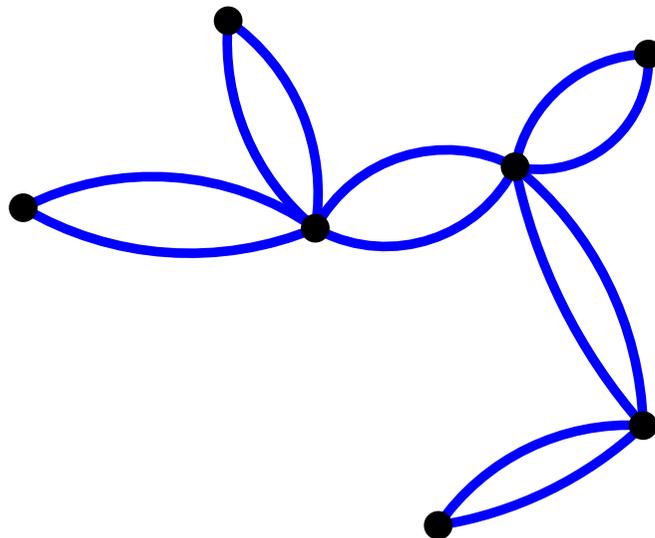
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Wiederholung – Tree-Doubling für Δ -TSP

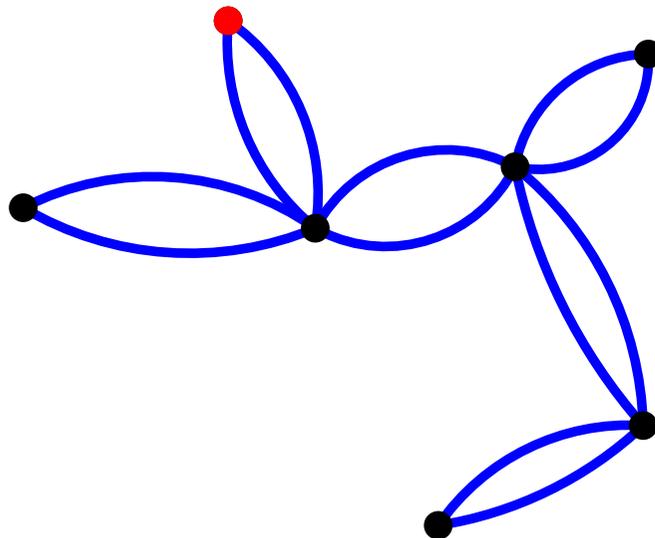
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Wiederholung – Tree-Doubling für Δ -TSP

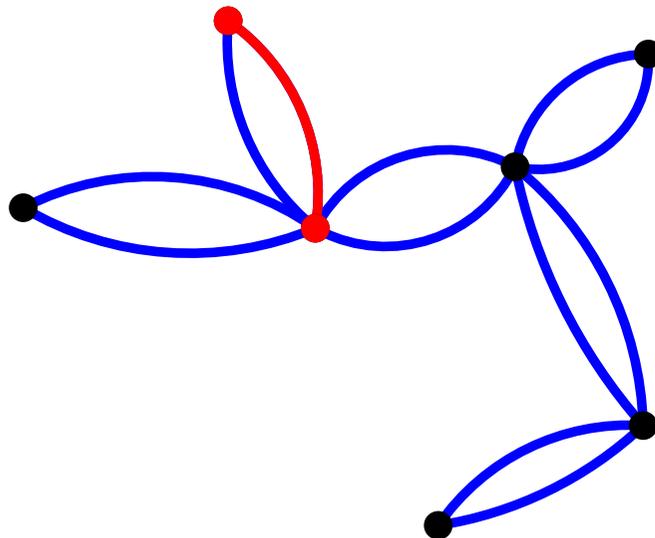
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Wiederholung – Tree-Doubling für Δ -TSP

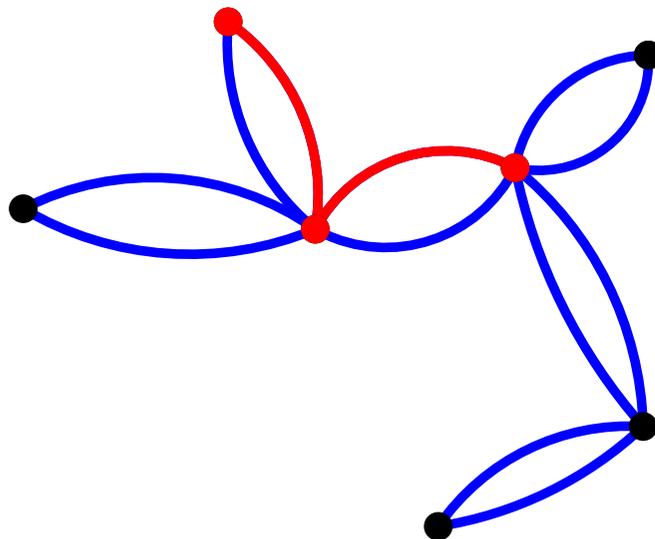
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Wiederholung – Tree-Doubling für Δ -TSP

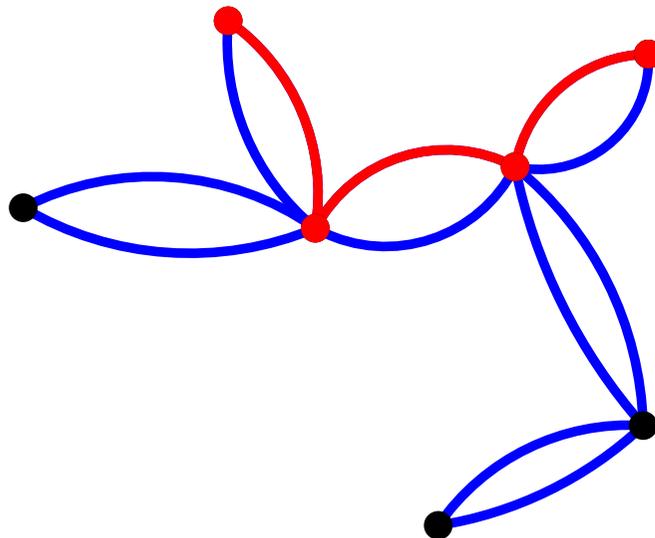
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Wiederholung – Tree-Doubling für Δ -TSP

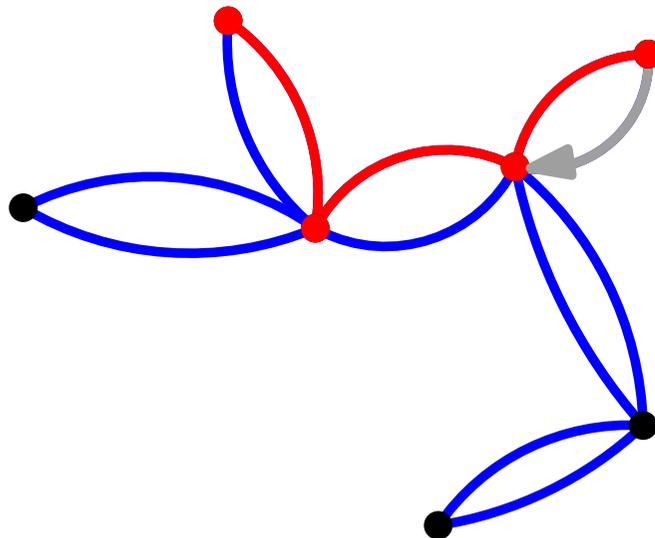
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Wiederholung – Tree-Doubling für Δ -TSP

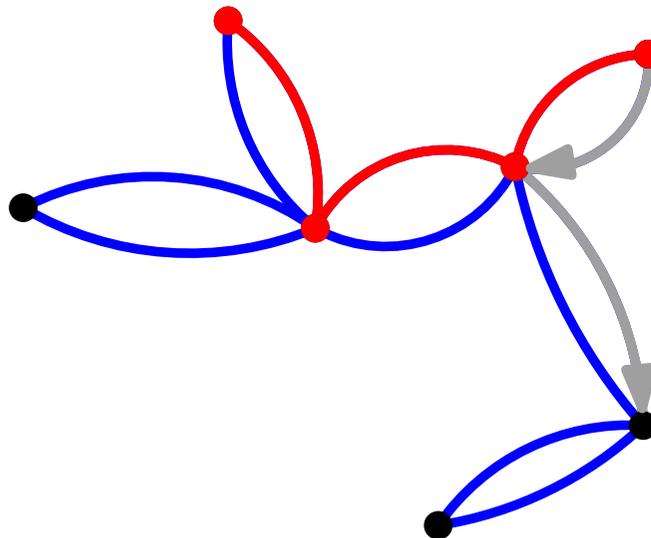
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Wiederholung – Tree-Doubling für Δ -TSP

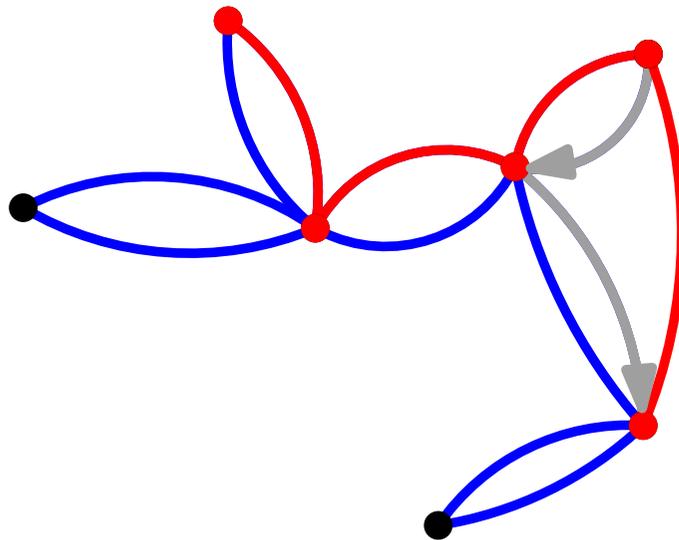
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

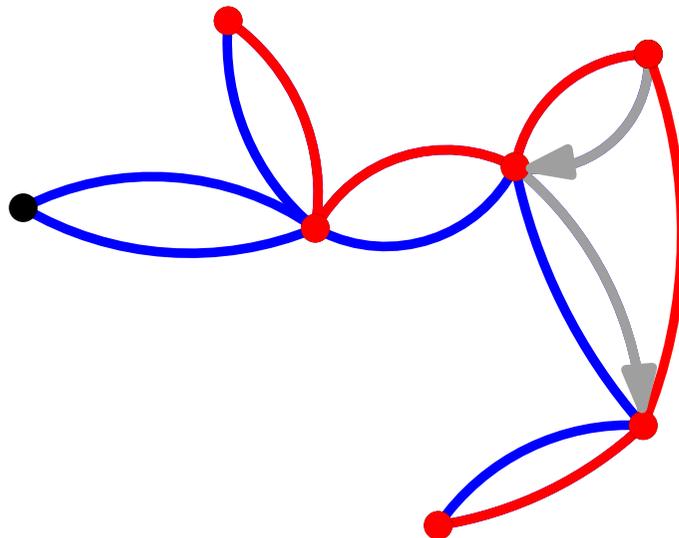
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

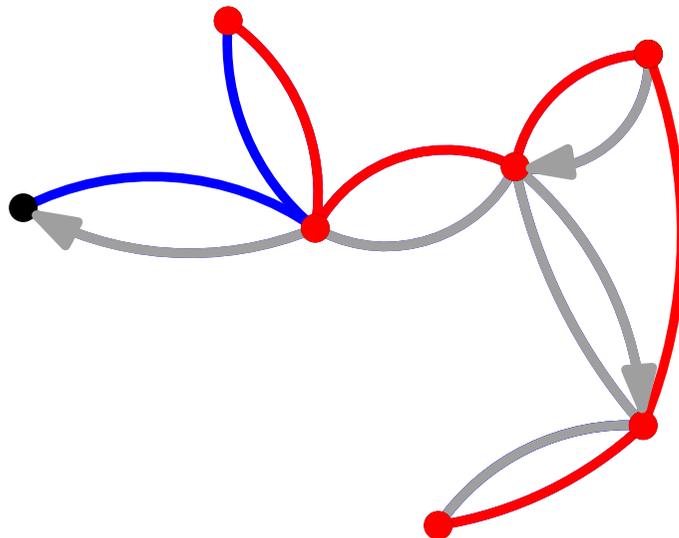
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

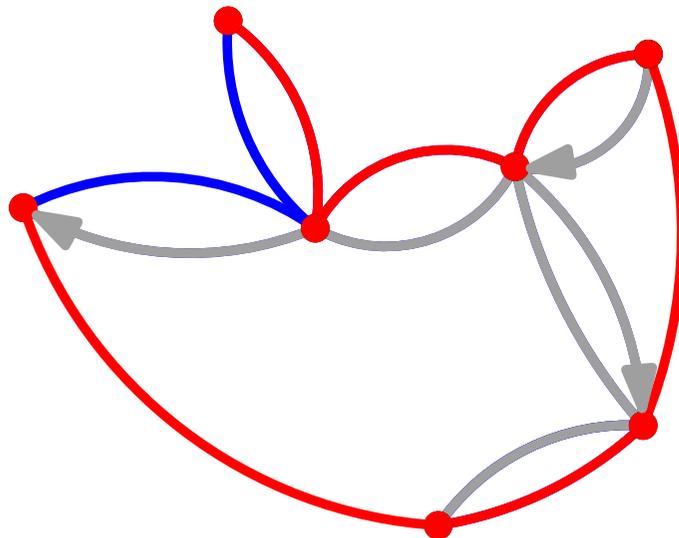
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

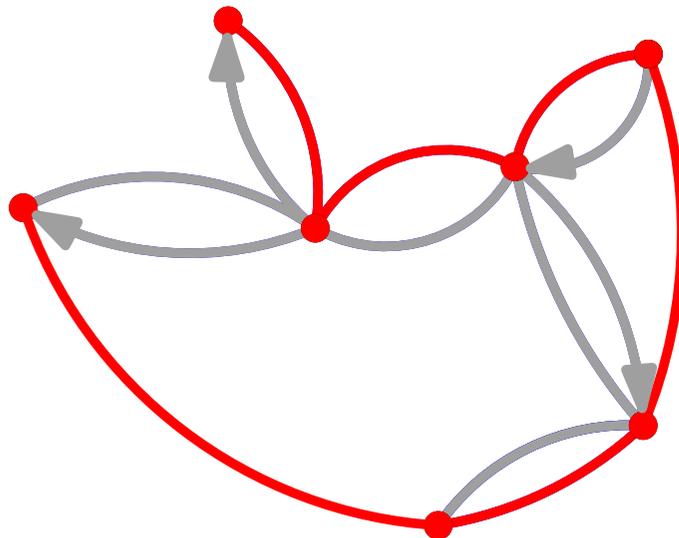
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

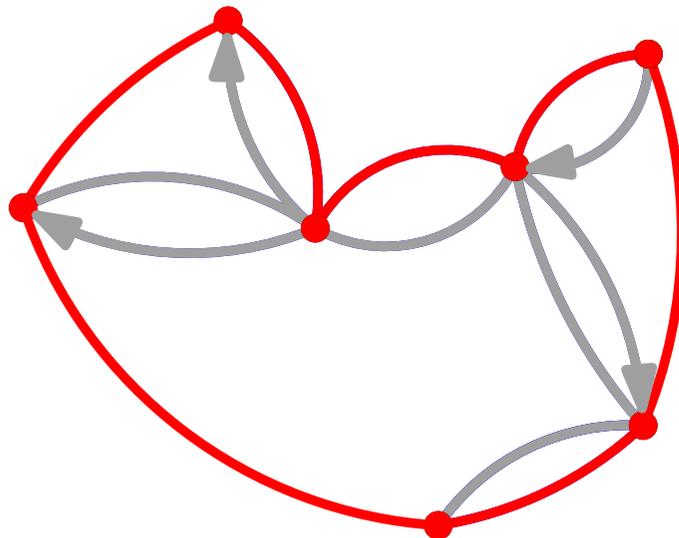
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

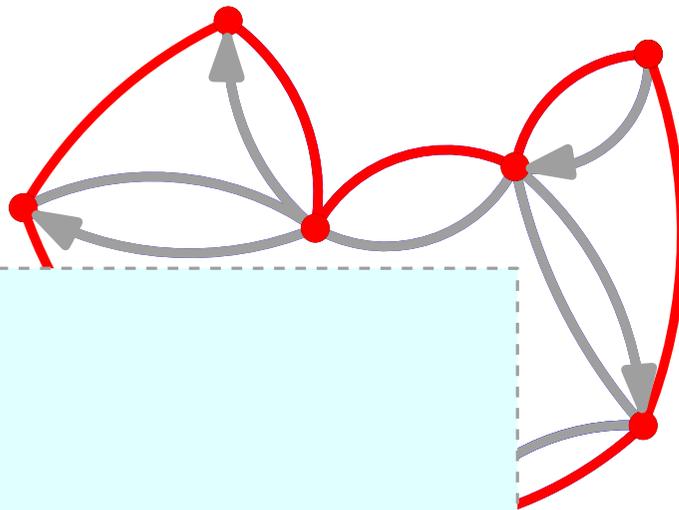
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



2. Analyse

1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

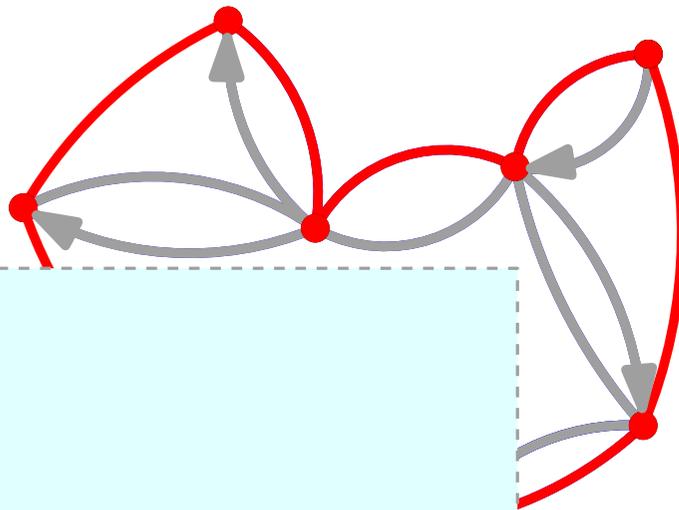
Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.



2. *Analyse*

$$c(\text{ALG}) \leq$$

1. *Algorithmus*

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

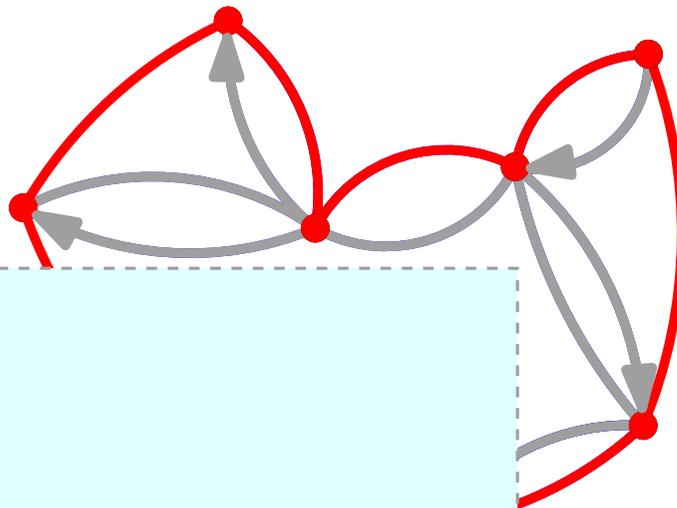
Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.

2. Analyse

$$c(\text{ALG}) \leq$$



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

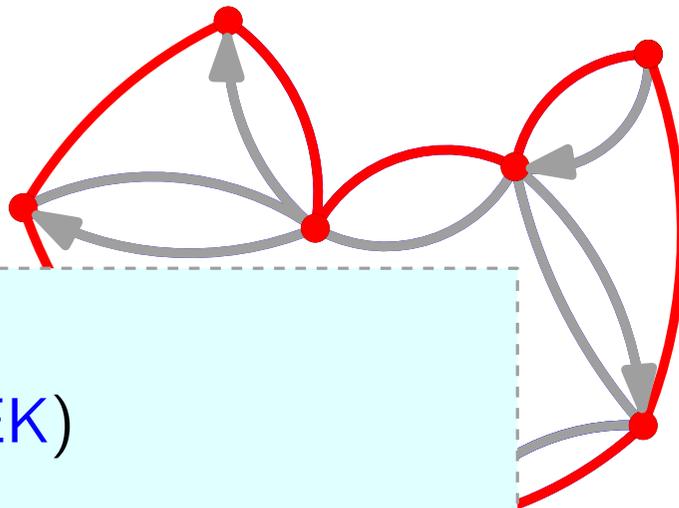
Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.

2. Analyse

$$c(\text{ALG}) \leq c(\text{EK})$$



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

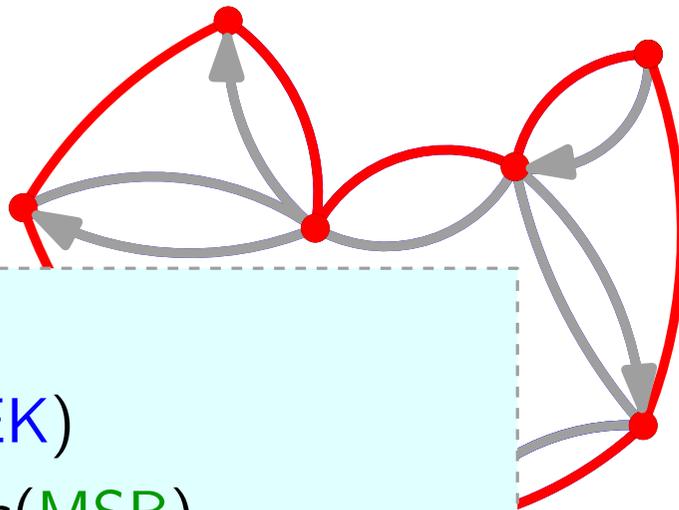
Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.

2. Analyse

$$\begin{aligned} c(\text{ALG}) &\leq c(\text{EK}) \\ &= 2 \cdot c(\text{MSB}) \end{aligned}$$



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

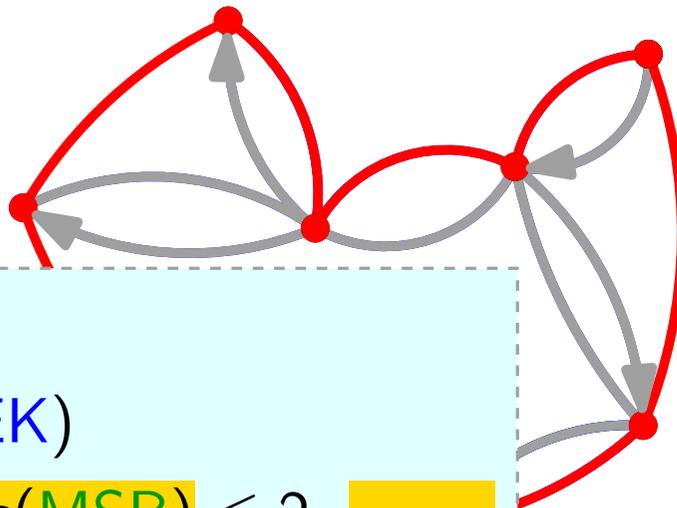
Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.

2. Analyse

$$c(\text{ALG}) \leq c(\text{EK})$$

$$= 2 \cdot c(\text{MSB}) \leq 2 \cdot \text{■}$$



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

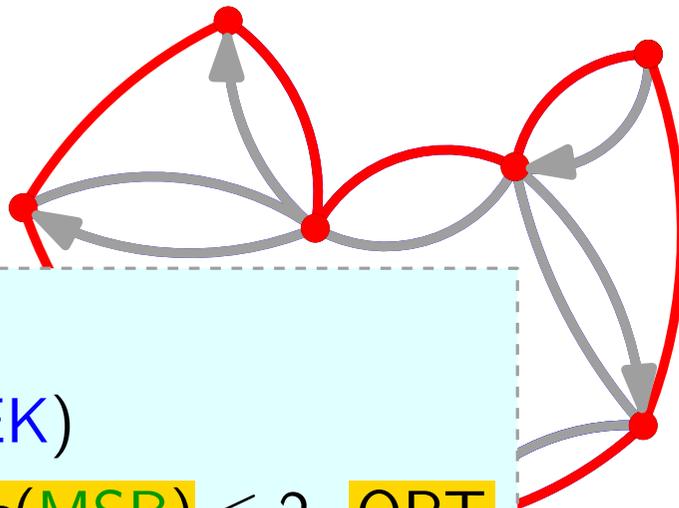
Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.

2. Analyse

$$\begin{aligned} c(\text{ALG}) &\leq c(\text{EK}) \\ &= 2 \cdot c(\text{MSB}) \leq 2 \cdot \text{OPT} \end{aligned}$$



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.

2. Analyse

$$\begin{aligned} c(\text{ALG}) &\leq c(\text{EK}) \\ &= 2 \cdot c(\text{MSB}) \leq 2 \cdot \text{OPT} \end{aligned}$$

1. Algorithmus

Berechne **MSB** von G .

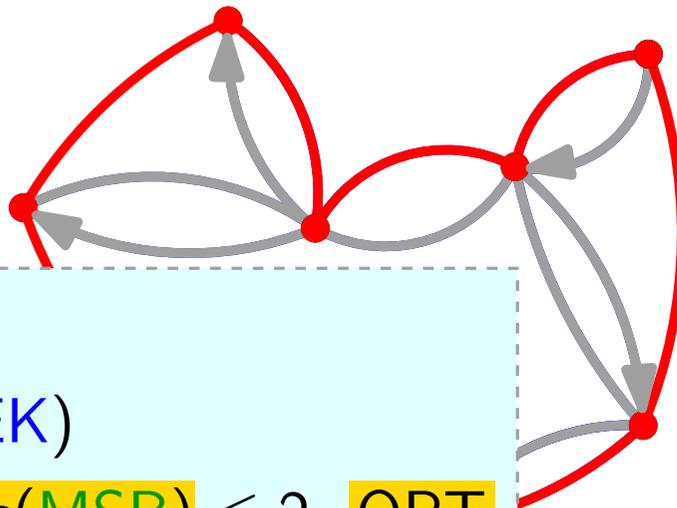
Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Opt. TSP-Tour minus eine Kante ist (i.A. nicht minimaler) Spannbaum!!



Wiederholung – Tree-Doubling für Δ -TSP

Problem: *Metrisches Traveling Salesperson Problem (Δ -TSP)*

Gegeben: unger. vollständiger Graph $G = (V, E)$
 mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$,
 die die Dreiecksungleichung erfüllen.

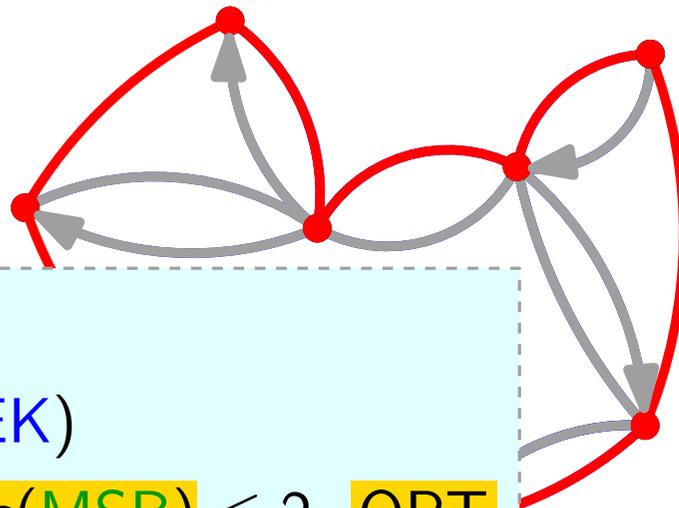
Gesucht: Hamiltonkreis in G mit minimalen Kosten.

Satz. Es gibt eine 2-Approximation für Δ -TSP.

Beweis.

2. Analyse

$$\begin{aligned} c(\text{ALG}) &\leq c(\text{EK}) \\ &= 2 \cdot c(\text{MSB}) \leq 2 \cdot \text{OPT} \end{aligned}$$



1. Algorithmus

Berechne **MSB** von G .

Verdopple MSB \Rightarrow eulersch!

Durchlaufe **Eulerkreis**.

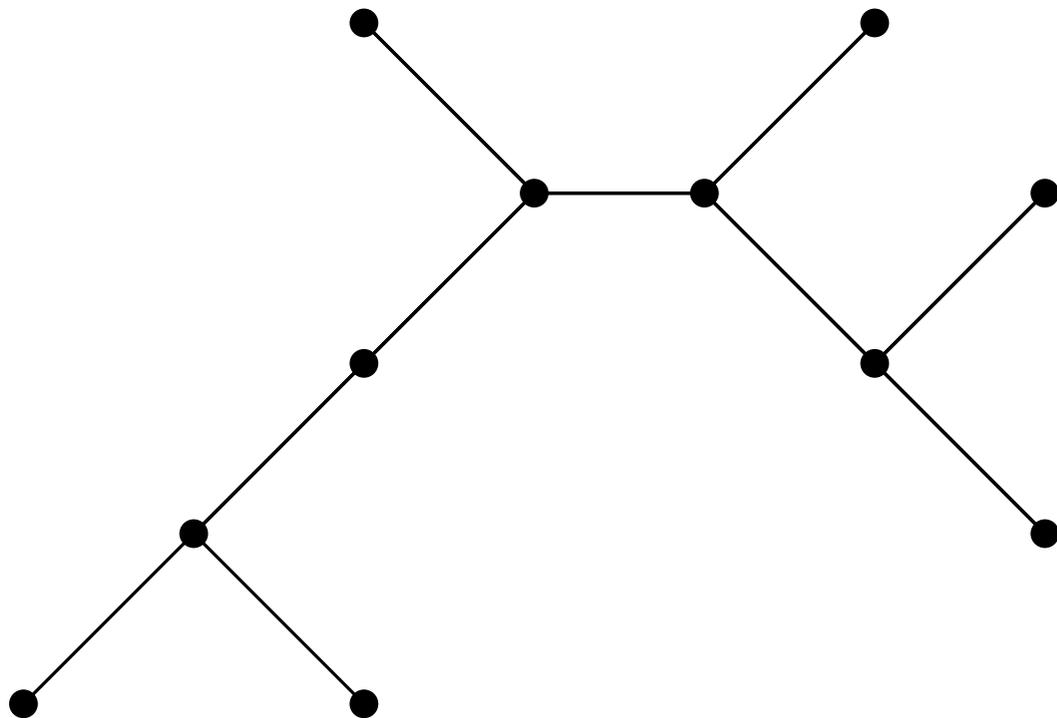
Überspringe besuchte Knoten.

Füge „Abkürzungen“ ein.

Opt. TSP-Tour minus eine Kante ist (i.A. nicht minimaler) Spannbaum!!
 „Die Kunst der unteren Schranke“

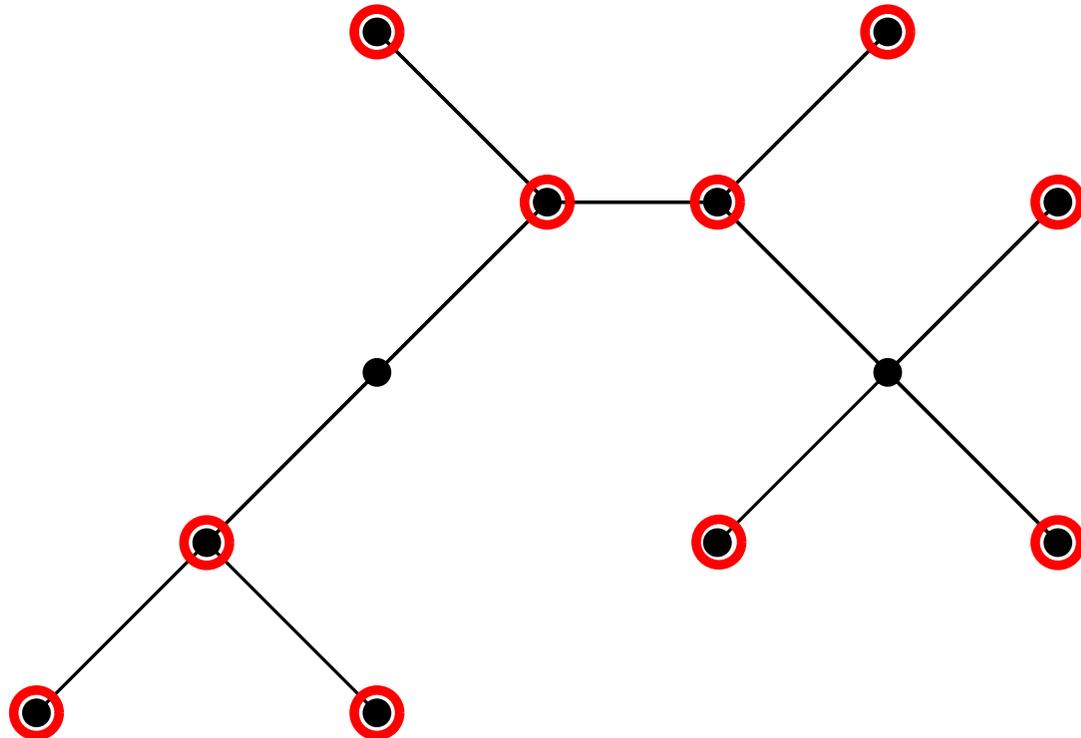
Christofides' Algorithmus

- Ermittle einen minimalen Spannbaum B für G .



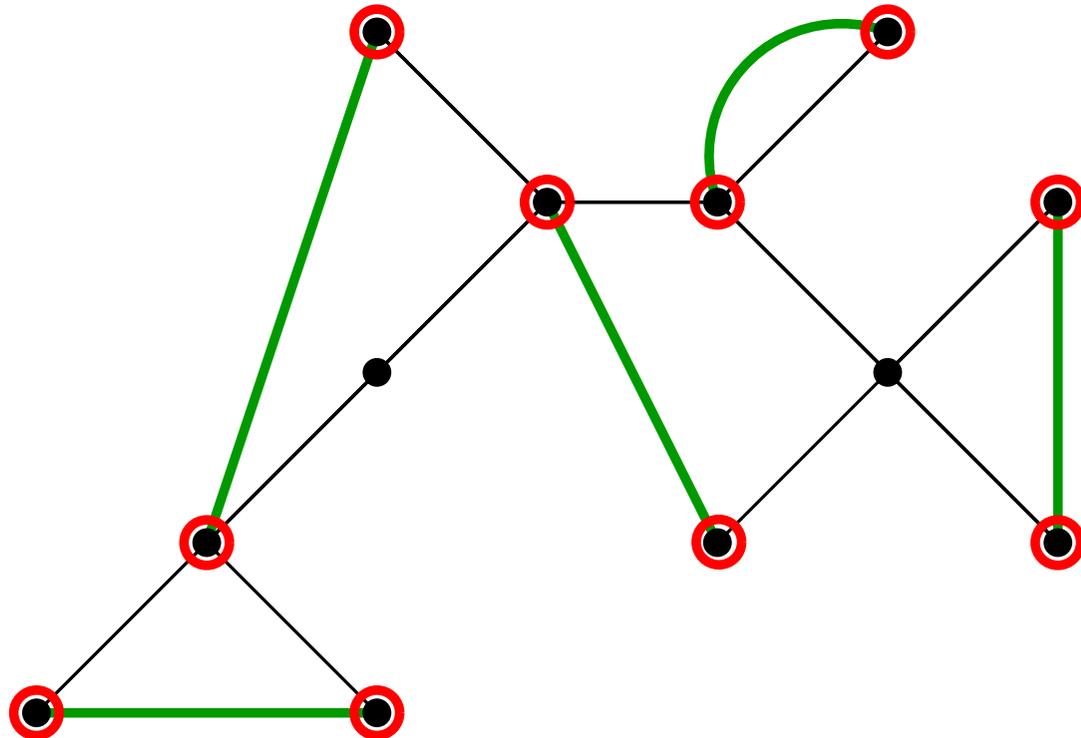
Christofides' Algorithmus

- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .



Christofides' Algorithmus

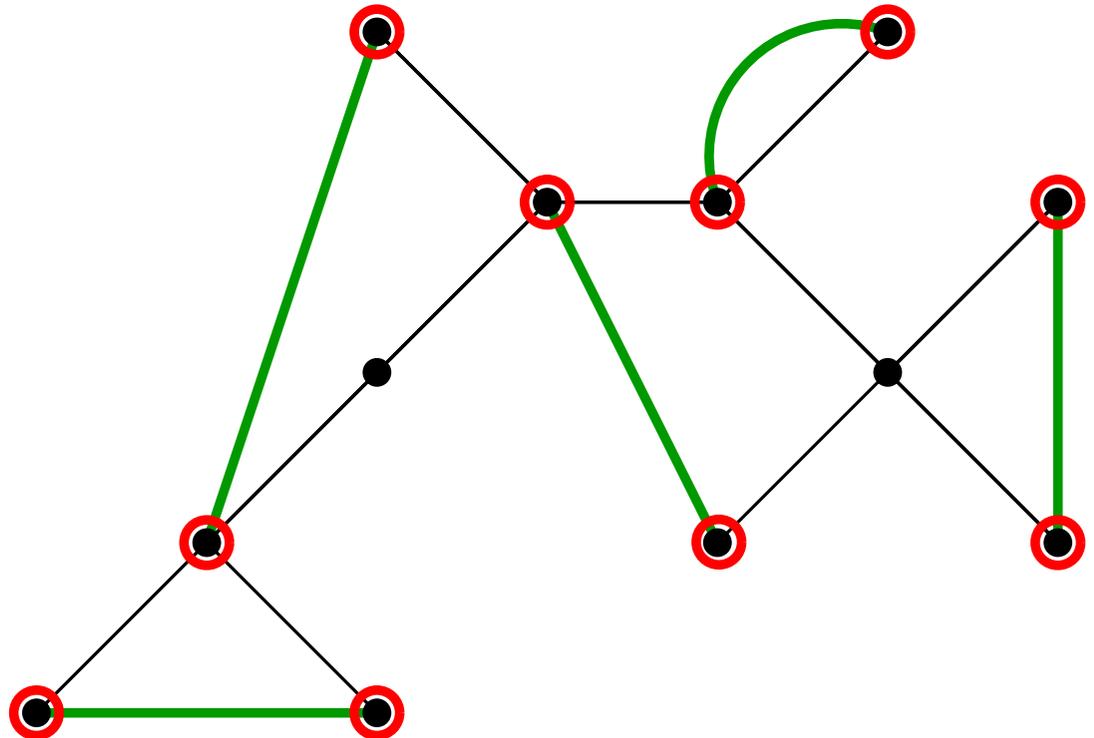
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$



Christofides' Algorithmus

- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

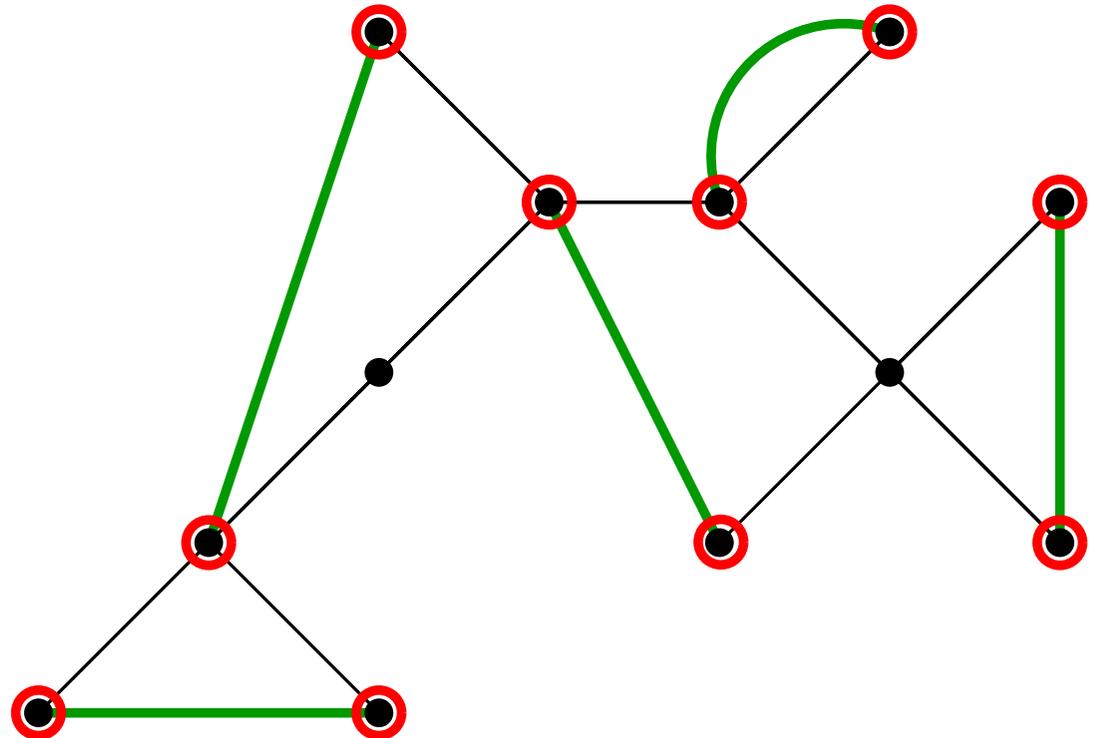


Christofides' Algorithmus

- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

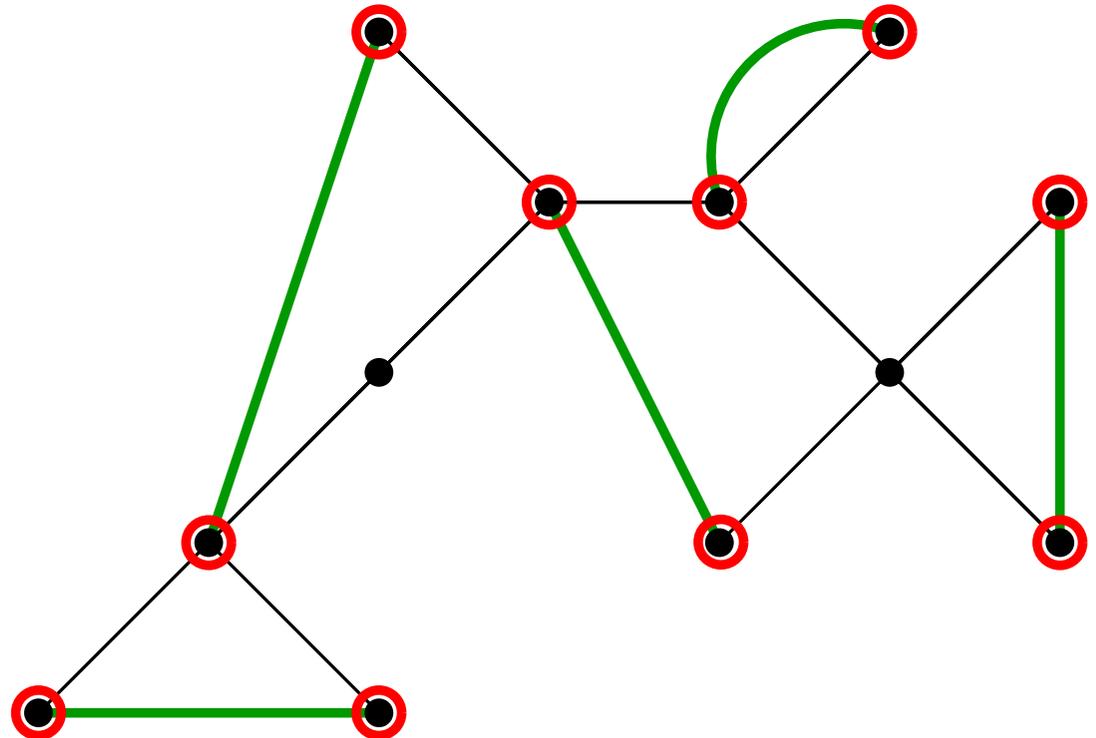


Christofides' Algorithmus

- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$



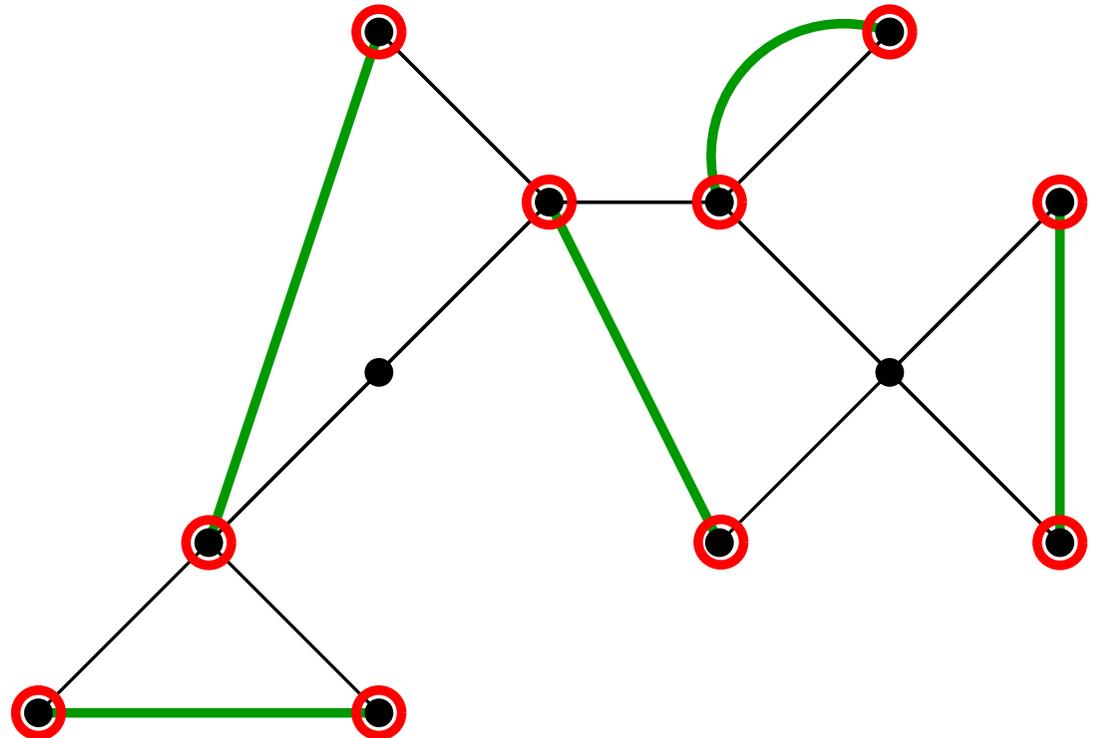
Christofides' Algorithmus

- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$



Christofides' Algorithmus

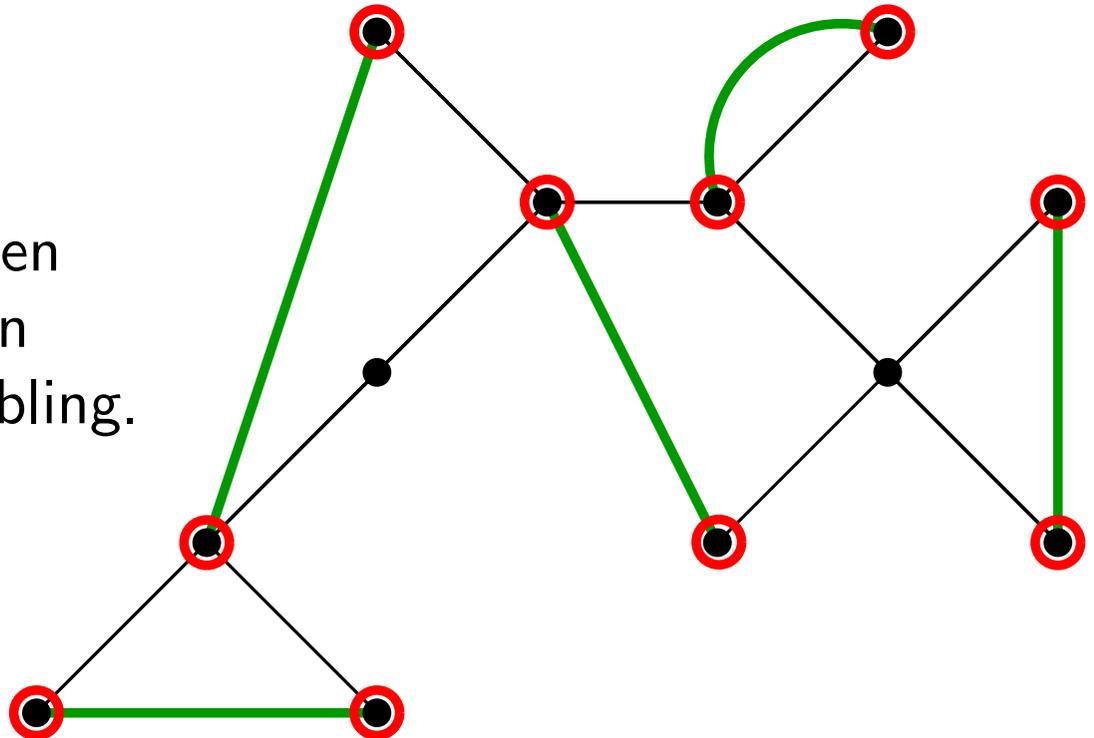
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$ (existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

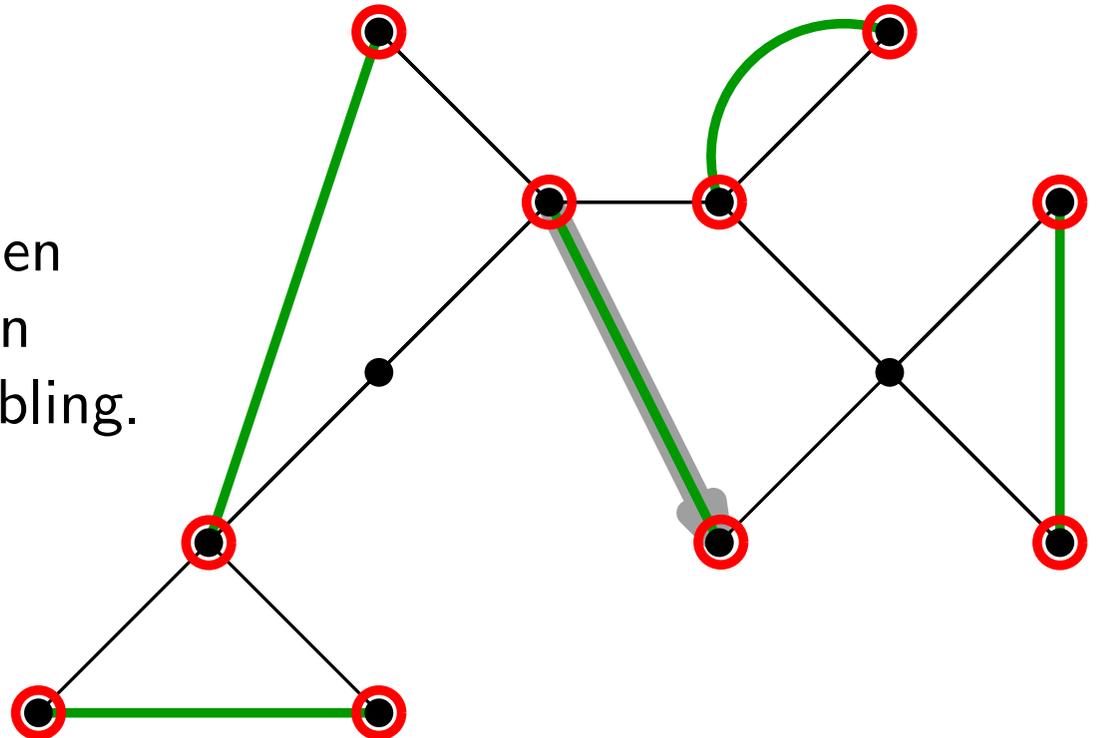
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$ (existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

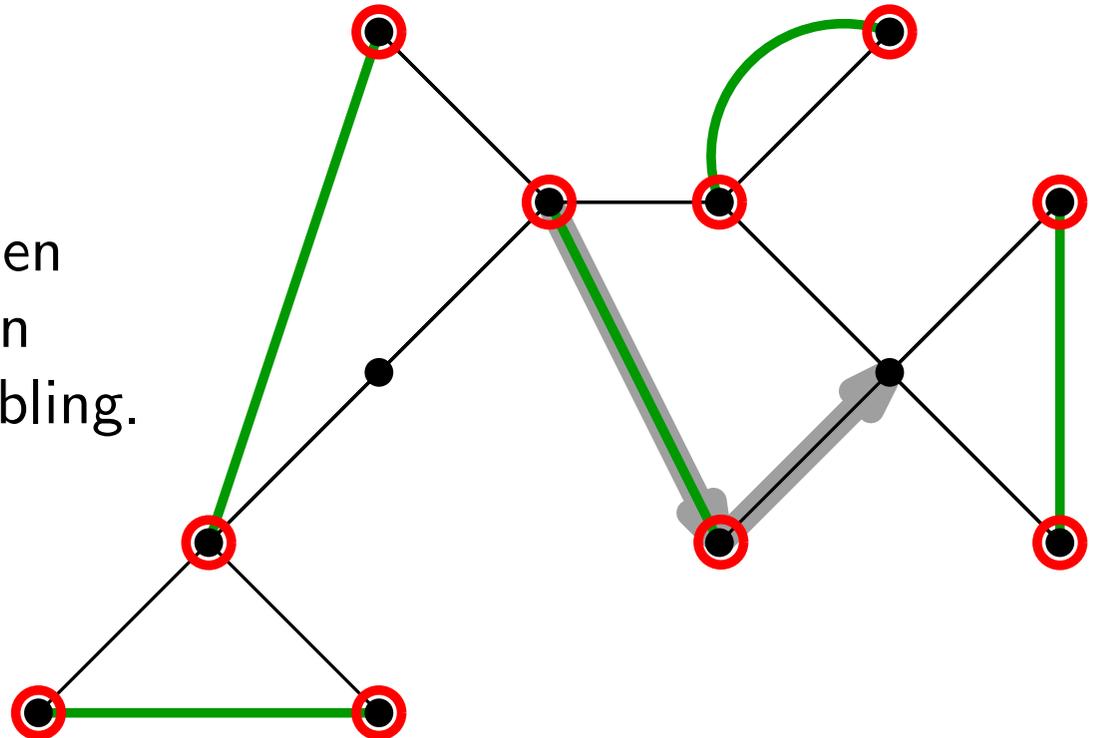
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$ (existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

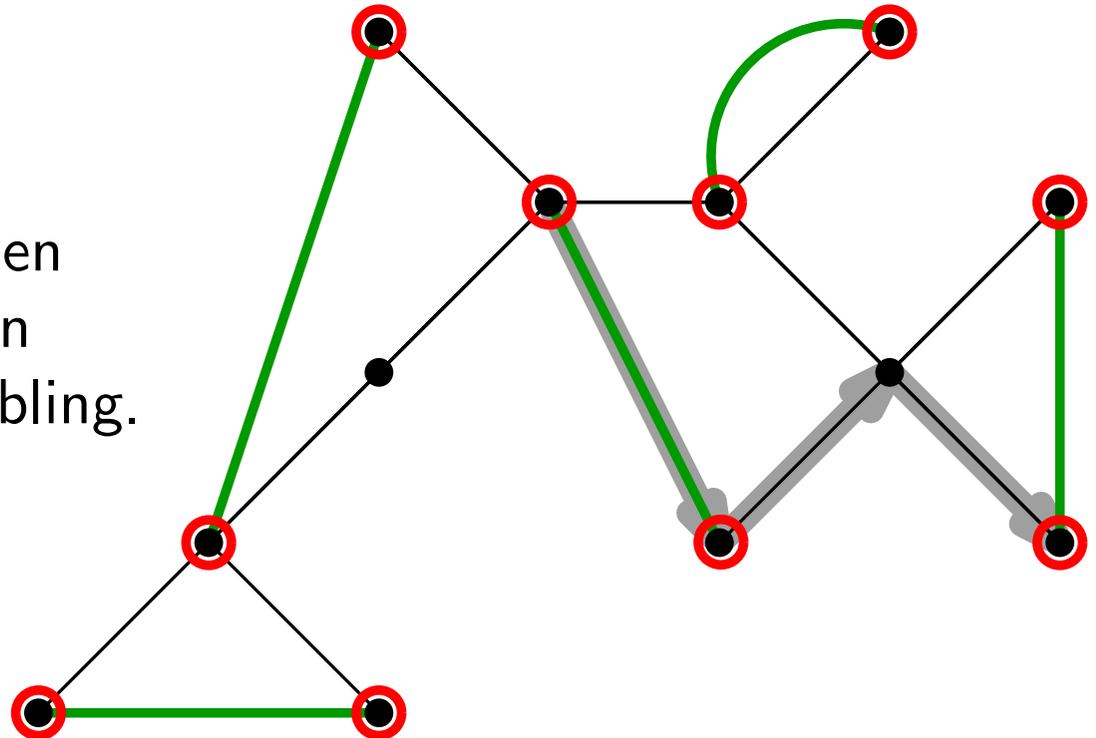
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$ (existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

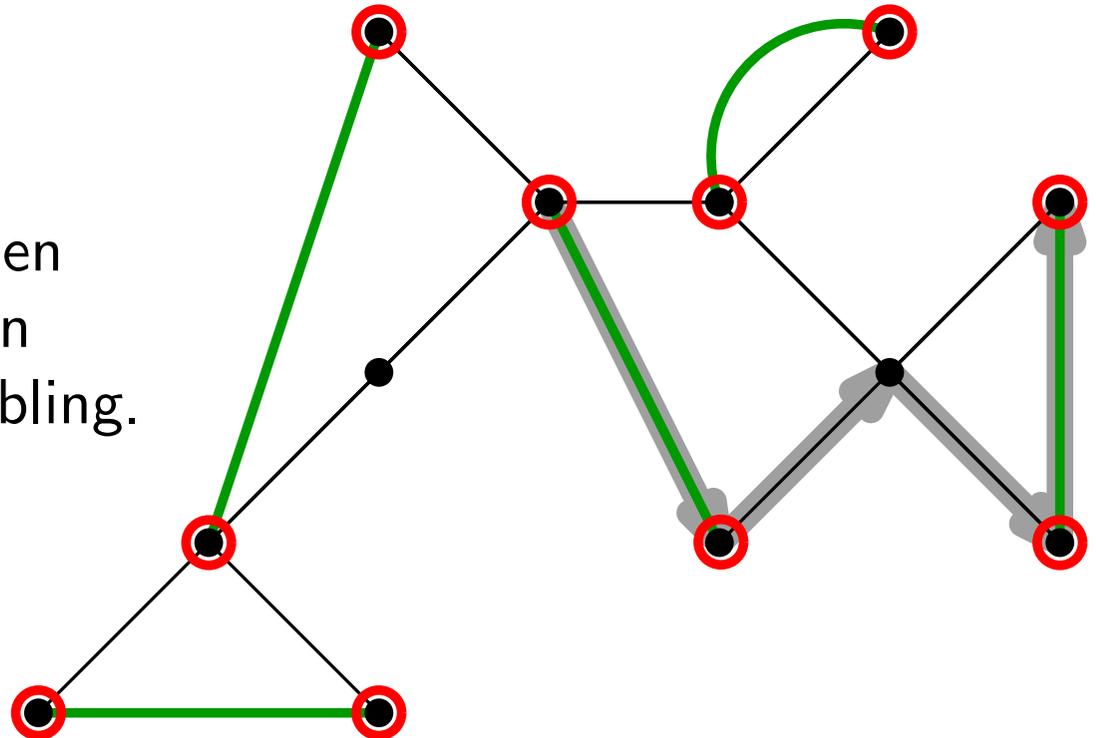
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

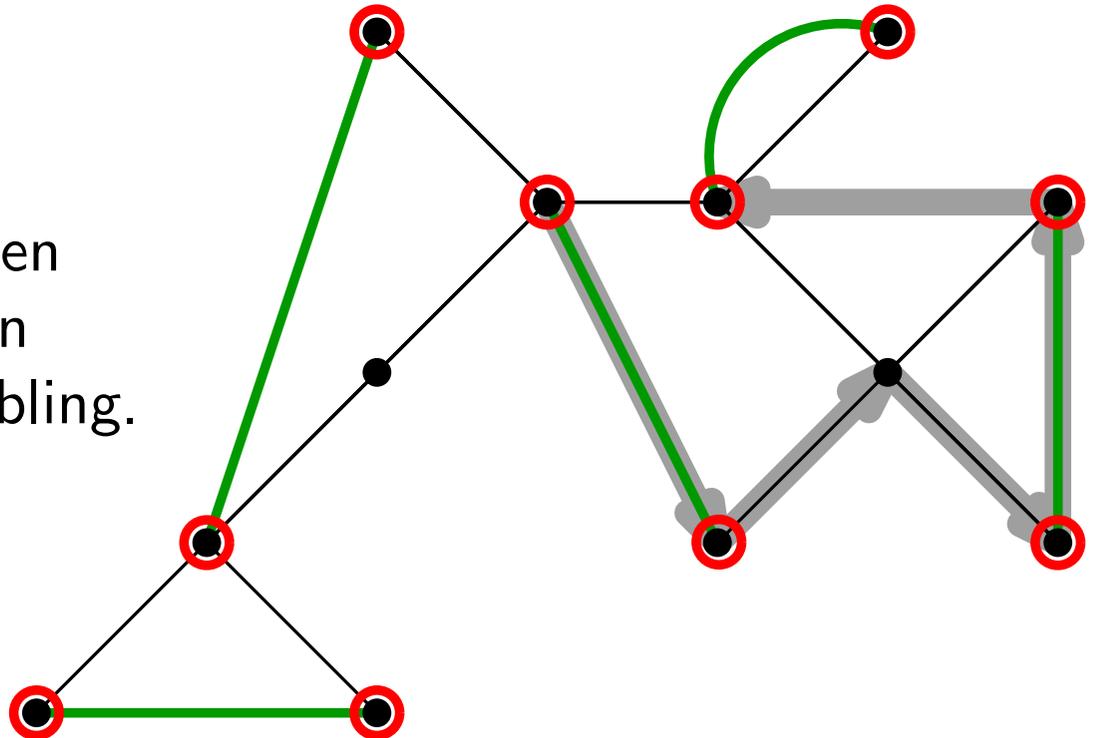
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

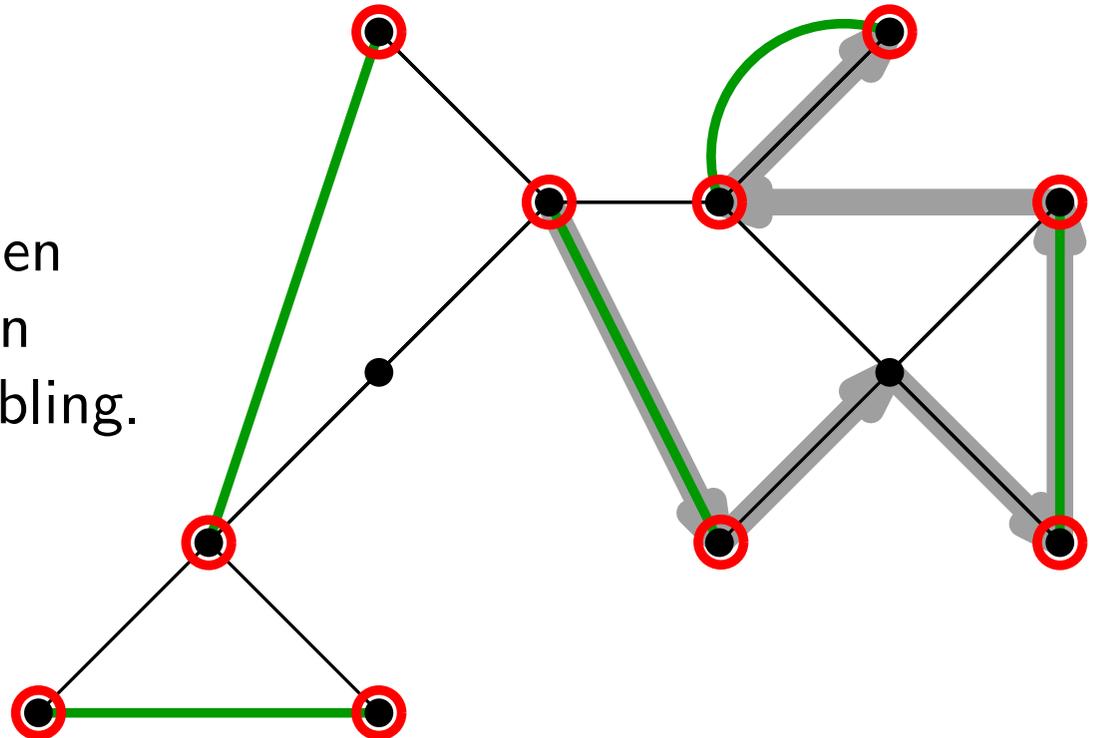
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

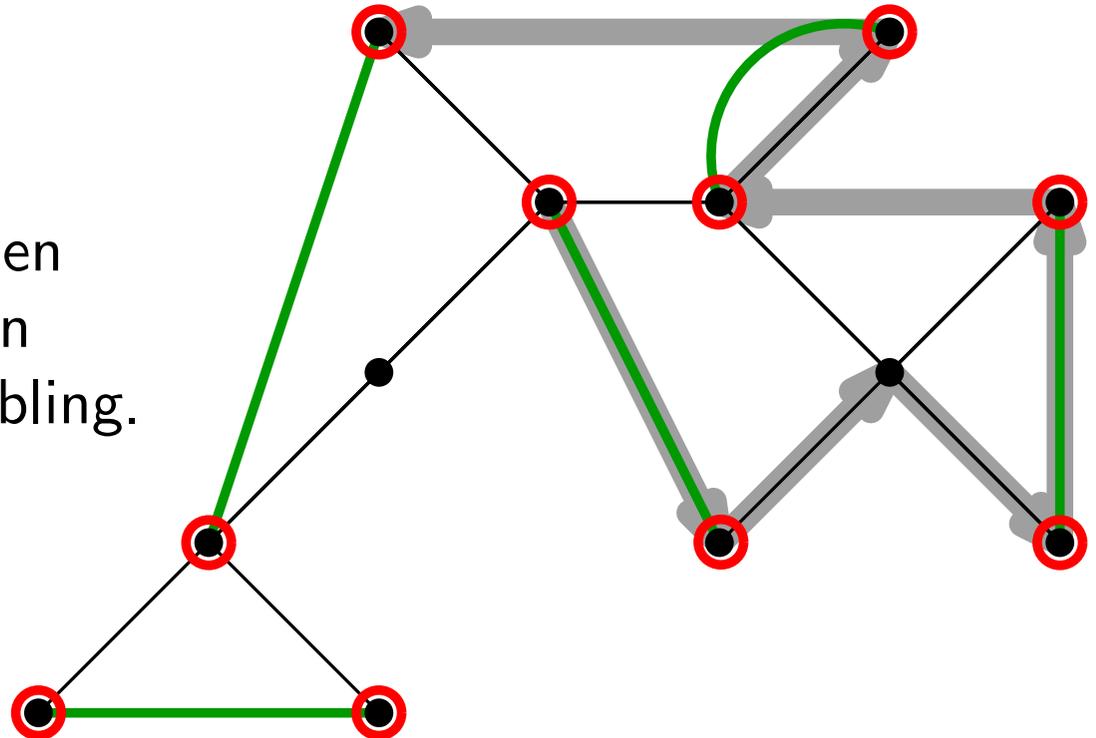
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$ (existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

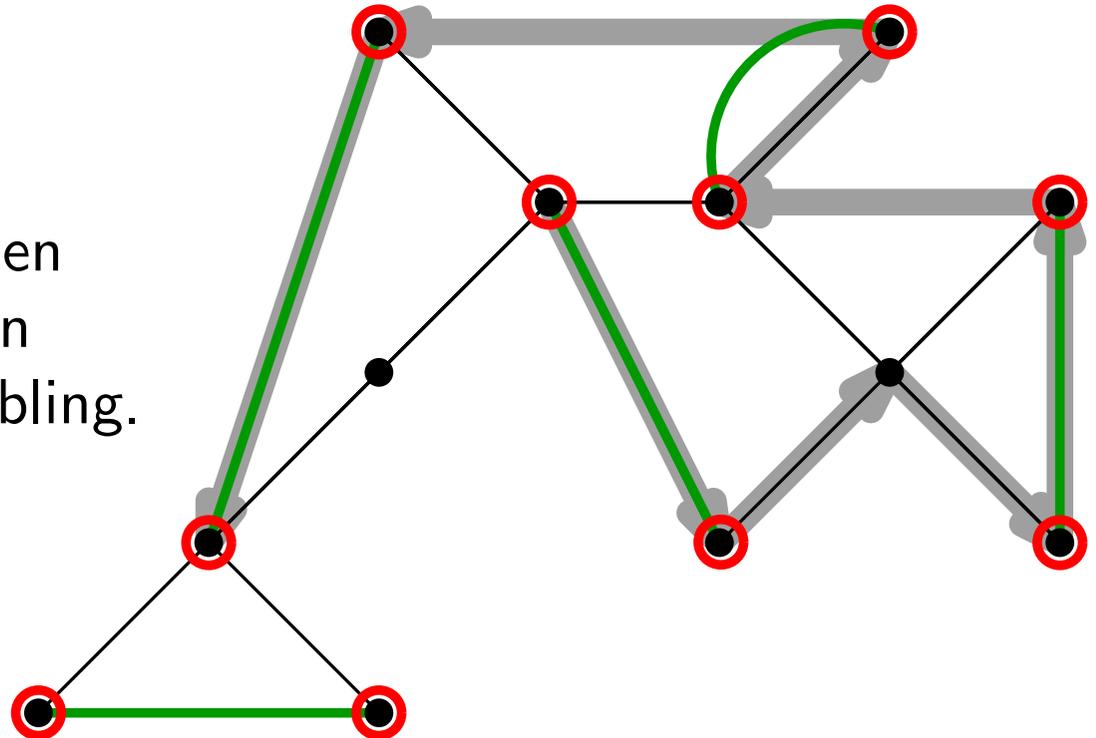
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

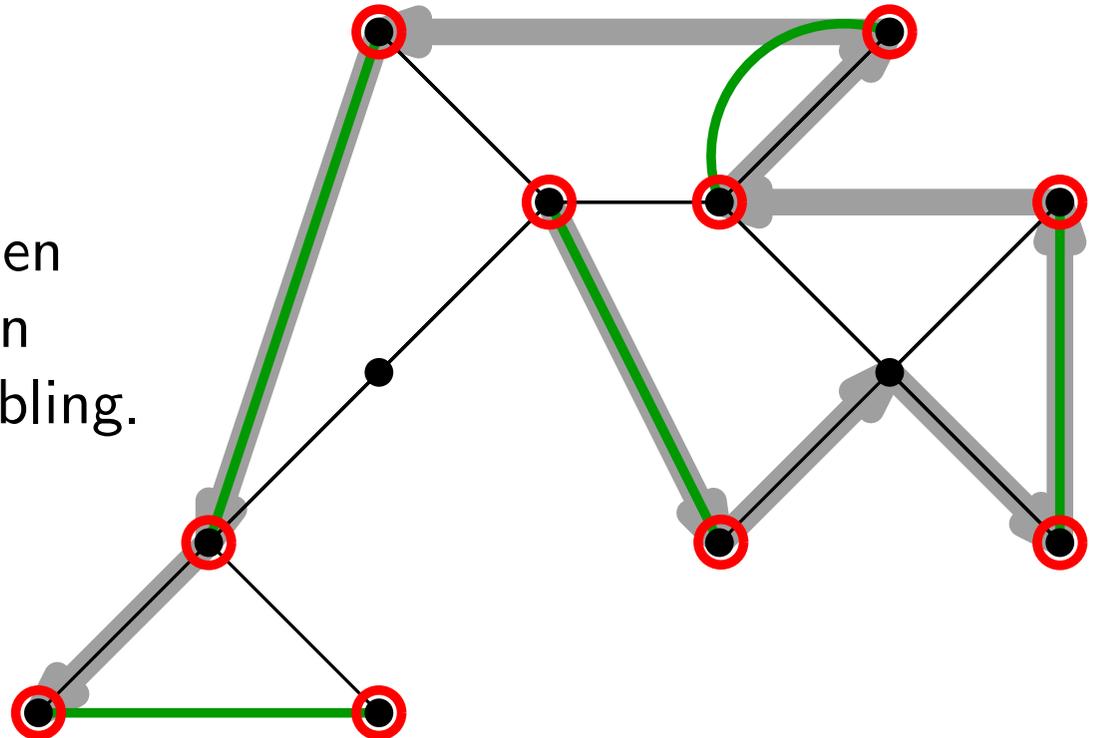
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

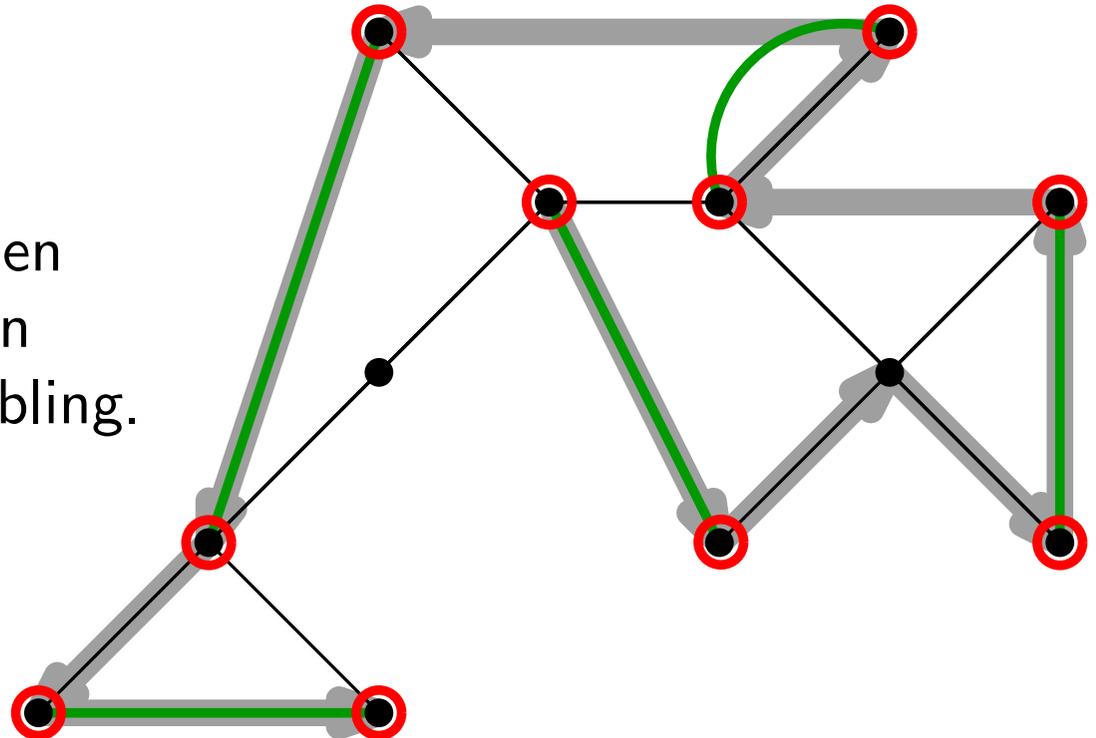
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

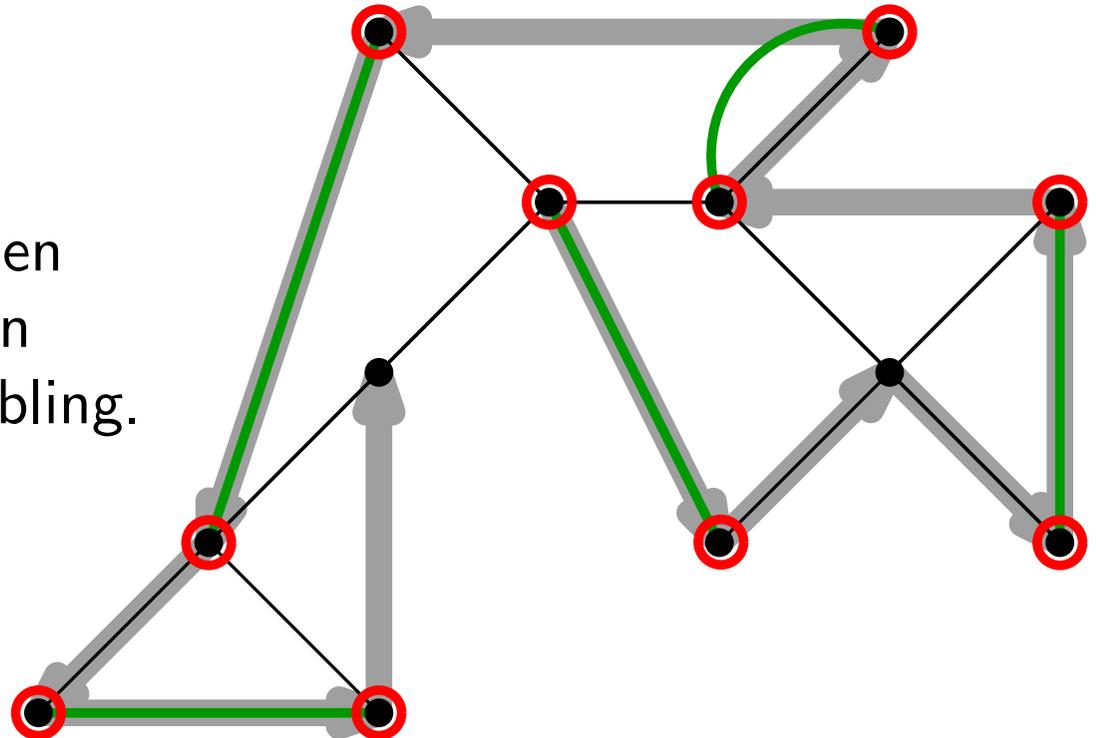
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

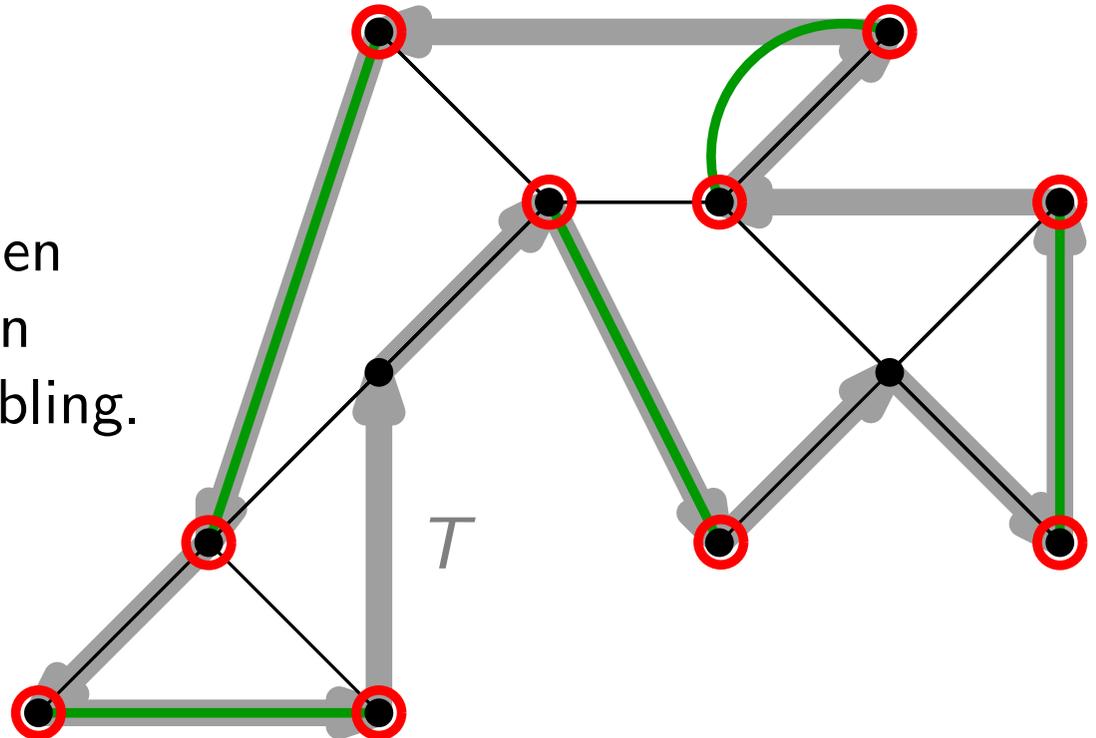
- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.



Christofides' Algorithmus

- Ermittle einen minimalen Spannbaum B für G .
- Sei U die Menge der Knoten ungeraden Grades in B .
- Ermittle *kostenminimales perfektes Matching* M für $G[U]$
(existiert, da $|U|$ gerade und $G[U]$ vollständig).

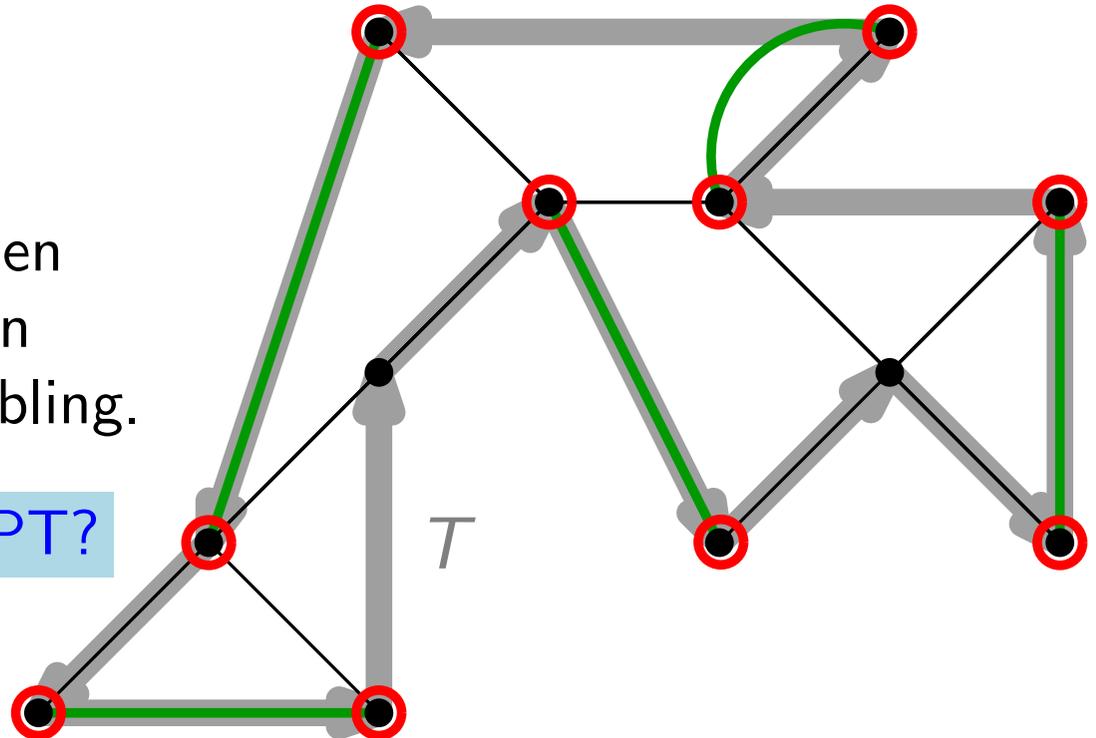
M ist unter allen perfekten Matchings in G eines mit minimalen Kosten
 $c(M) := \sum_{e \in M} c(e)$

Warum?

der von U induzierte Graph
 $(U, \{vw \in E(G) : v \in U, w \in U\})$

- Berechne im eulerschen Graphen $B \cup M$ erst Eulertour und dann Rundtour T wie bei Tree-Doubling.

Wie gut ist T im Vergleich zu OPT?



Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.

Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

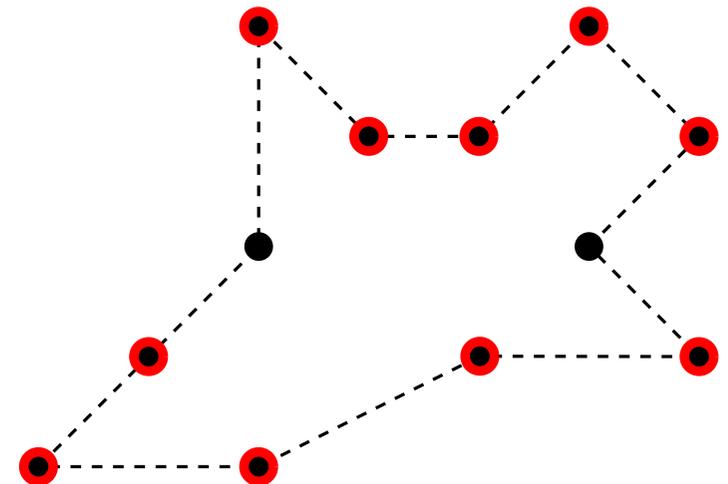
- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.

Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .

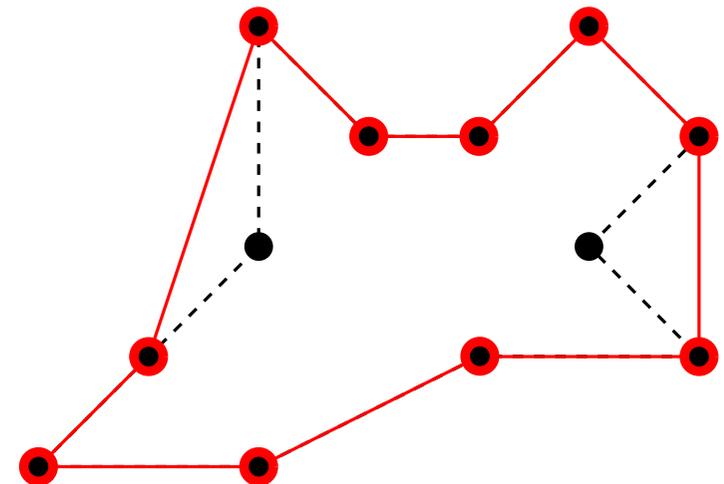


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.

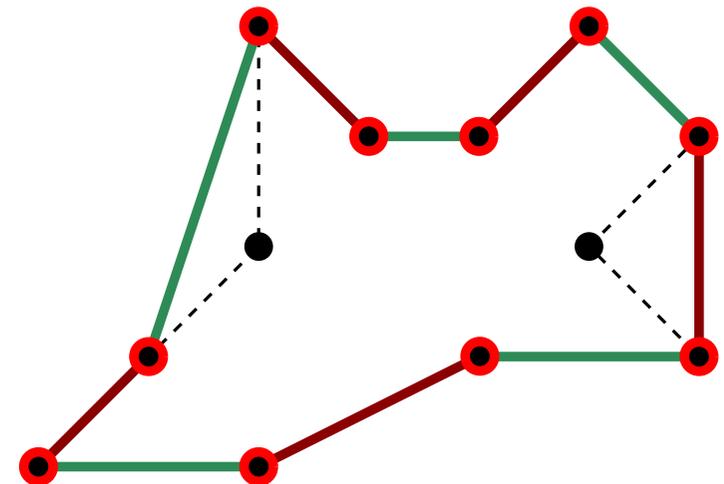


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.
- Zerlege in T^{**} in zwei disjunkte perfekte Matchings M' , M'' für $G[U]$.

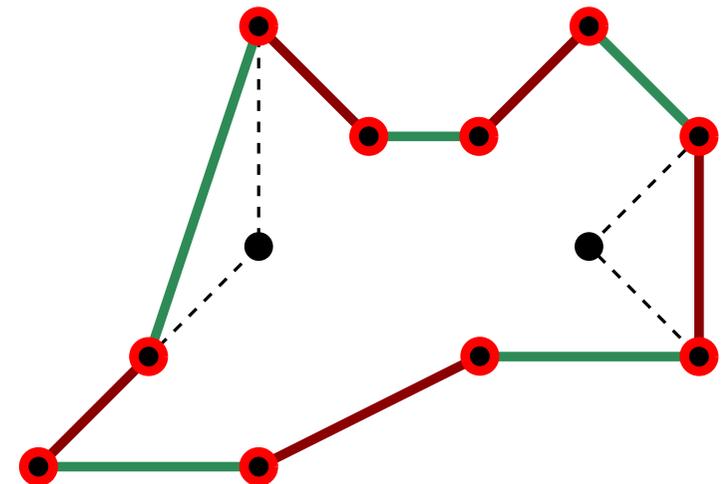


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.
- Zerlege in T^{**} in zwei disjunkte perfekte Matchings M' , M'' für $G[U]$.
- O.B.d.A. $c(M') \leq c(M'')$.

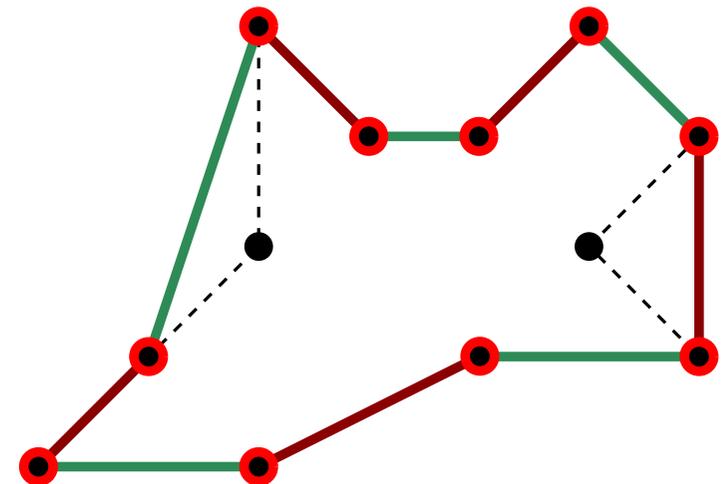


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.
- Zerlege in T^{**} in zwei disjunkte perfekte Matchings M' , M'' für $G[U]$.
- O.B.d.A. $c(M') \leq c(M'')$.
- Also gilt $c(M') \leq c(T^{**})/2 \leq \text{OPT}/2$.

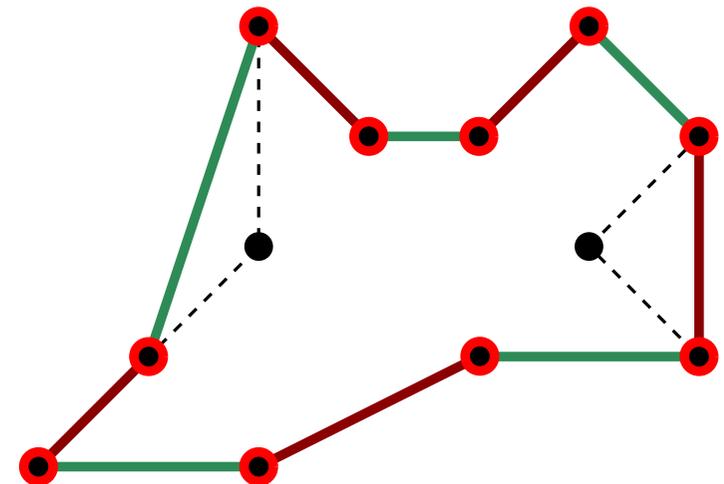


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.
- Zerlege in T^{**} in zwei disjunkte perfekte Matchings M' , M'' für $G[U]$.
- O.B.d.A. $c(M') \leq c(M'')$.
- Also gilt $c(M') \leq c(T^{**})/2 \leq \text{OPT}/2$.
- Außerdem gilt $c(M) \leq c(M')$, da

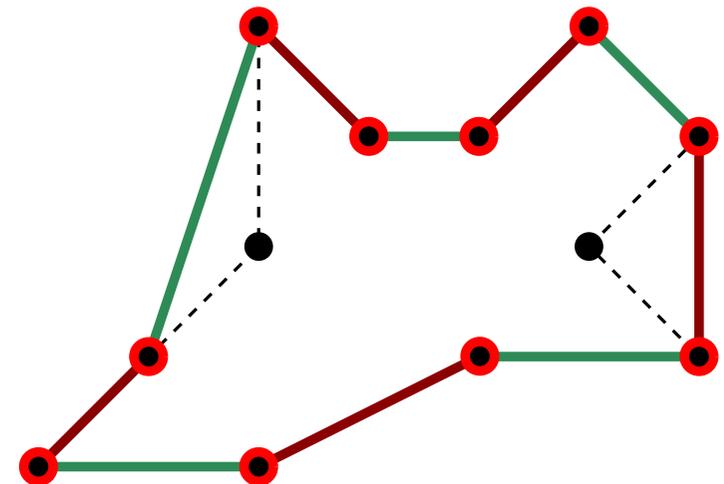


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.
- Zerlege in T^{**} in zwei disjunkte perfekte Matchings M' , M'' für $G[U]$.
- O.B.d.A. $c(M') \leq c(M'')$.
- Also gilt $c(M') \leq c(T^{**})/2 \leq \text{OPT}/2$.
- Außerdem gilt $c(M) \leq c(M')$, da
 - M' perfektes Matching in $G[U]$

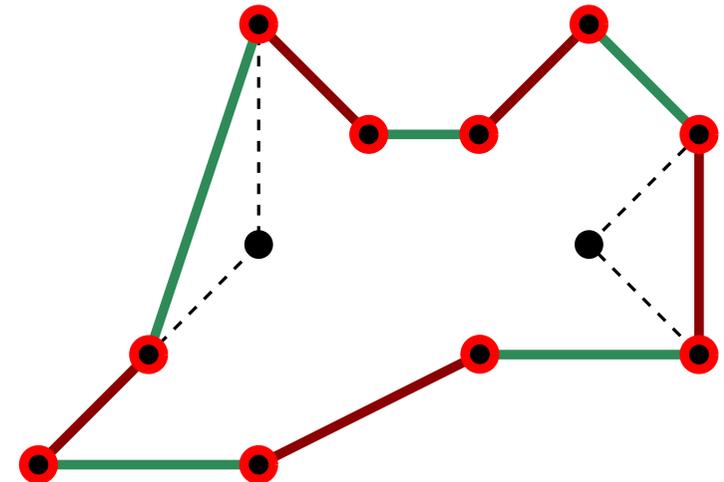


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.
- Zerlege in T^{**} in zwei disjunkte perfekte Matchings M' , M'' für $G[U]$.
- O.B.d.A. $c(M') \leq c(M'')$.
- Also gilt $c(M') \leq c(T^{**})/2 \leq \text{OPT}/2$.
- Außerdem gilt $c(M) \leq c(M')$, da
 - M' perfektes Matching in $G[U]$
 - M kostenminimales perfektes Matching in $G[U]$

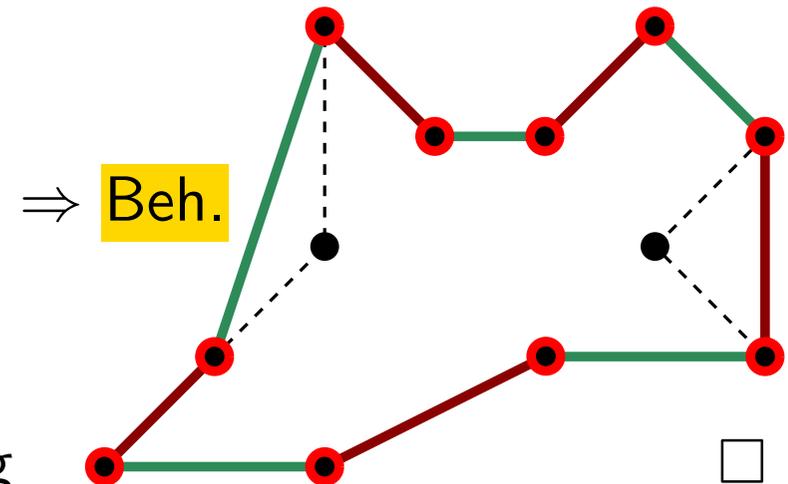


Analyse von Christofides

Satz. Christofides liefert eine $3/2$ -Approximation für Δ -TSP.

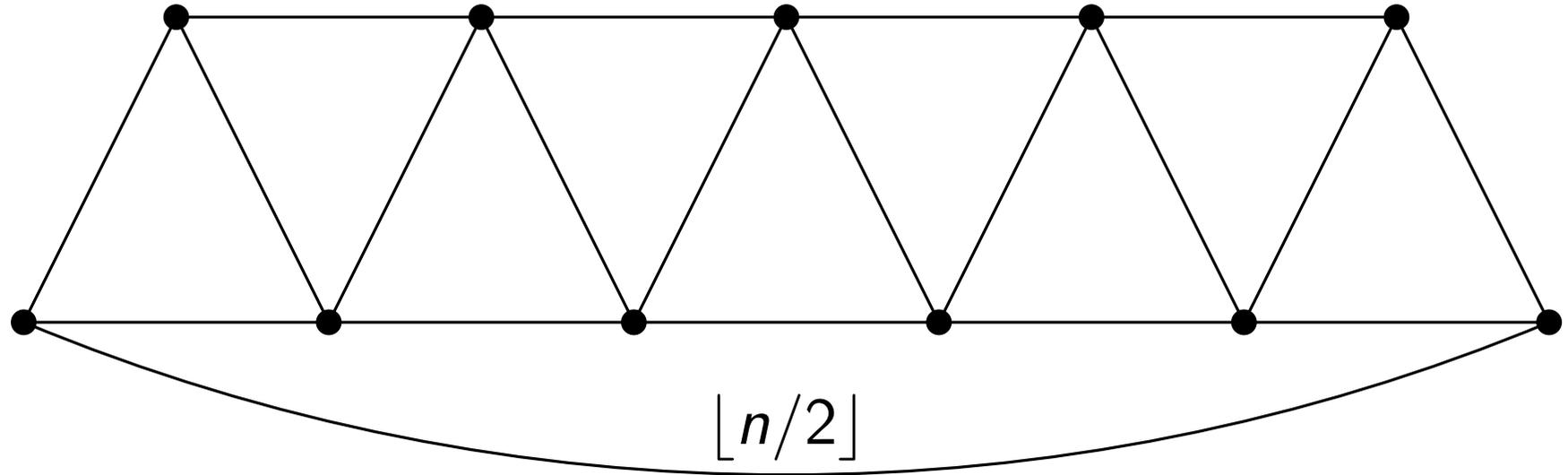
Beweis.

- Genügt zu zeigen, dass $c(B \cup M) \leq 3/2 \cdot \text{OPT}$, da $c(T) \leq c(B \cup M)$.
- Da $c(B) \leq \text{OPT}$, genügt es zu zeigen, dass $c(M) \leq \text{OPT}/2$.
- Betrachte optimale Tour T^* .
- Abkürzen von T^* liefert Tour T^{**} in $G[U]$ mit $c(T^{**}) \leq c(T^*) = \text{OPT}$.
- Zerlege in T^{**} in zwei disjunkte perfekte Matchings M' , M'' für $G[U]$.
- O.B.d.A. $c(M') \leq c(M'')$.
- Also gilt $c(M') \leq c(T^{**})/2 \leq \text{OPT}/2$.
- Außerdem gilt $c(M) \leq c(M')$, da
 - M' perfektes Matching in $G[U]$
 - M kostenminimales perfektes Matching in $G[U]$



Der Approximationsfaktor ist scharf

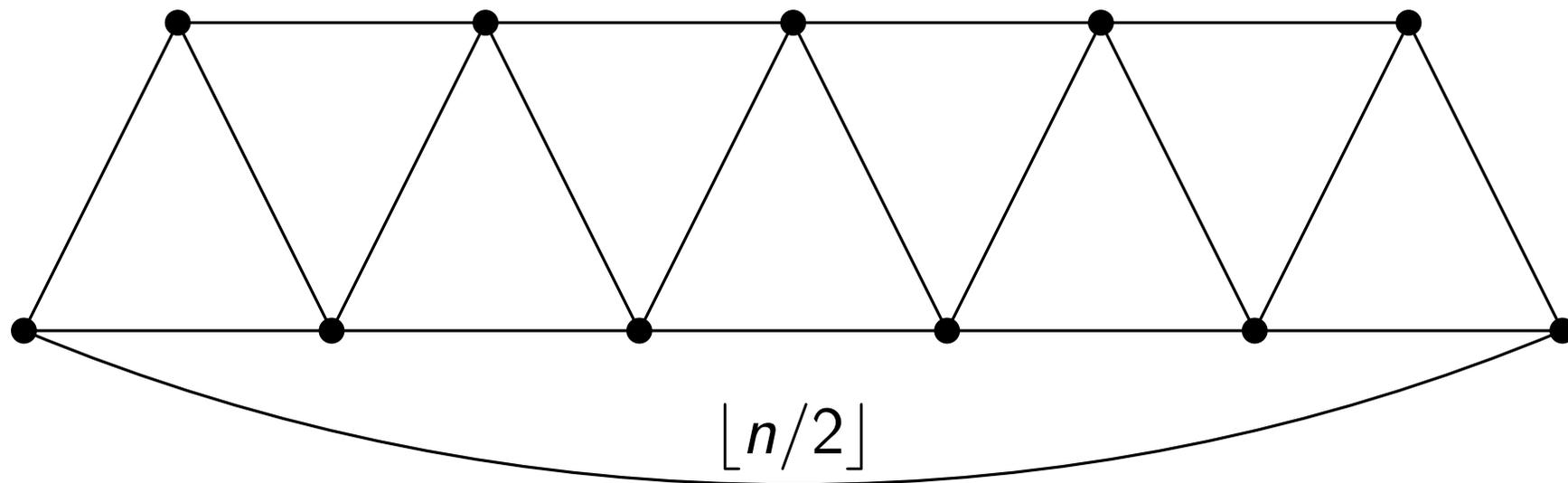
Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:



Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

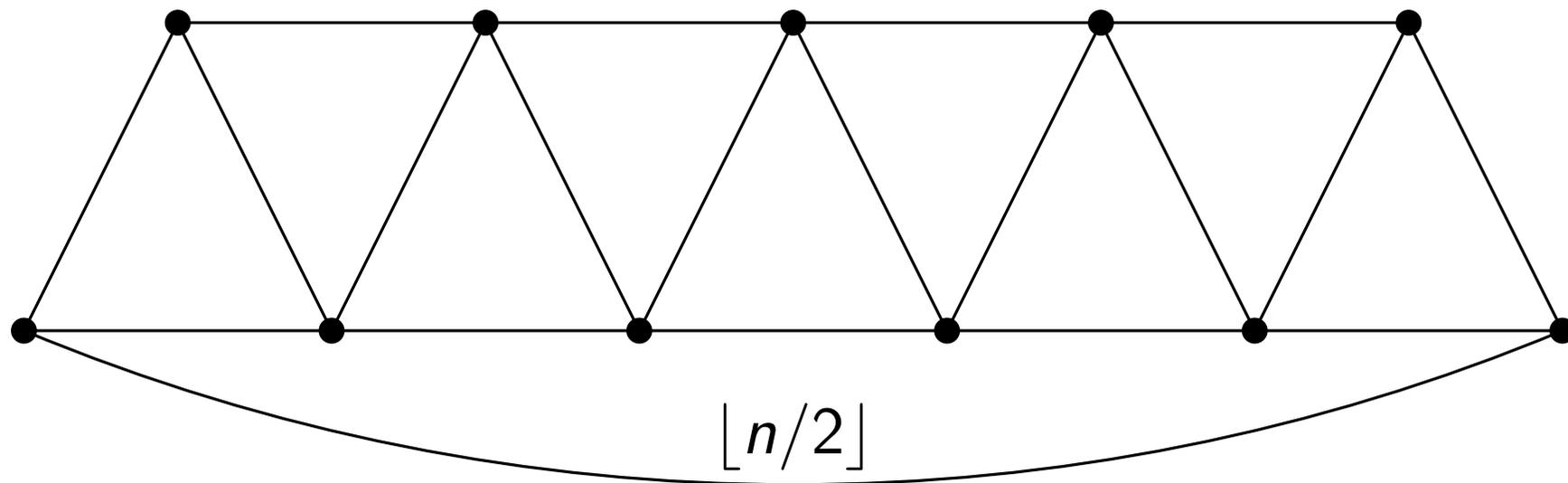
- Unbeschriftete Kanten haben Kosten 1.



Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

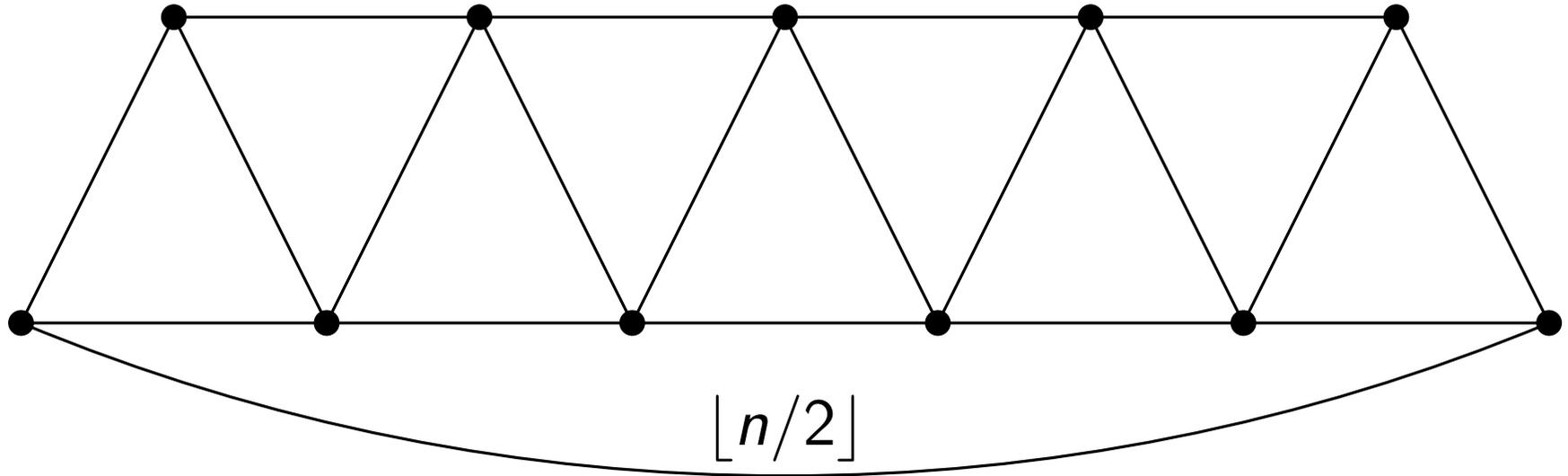
- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...



Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

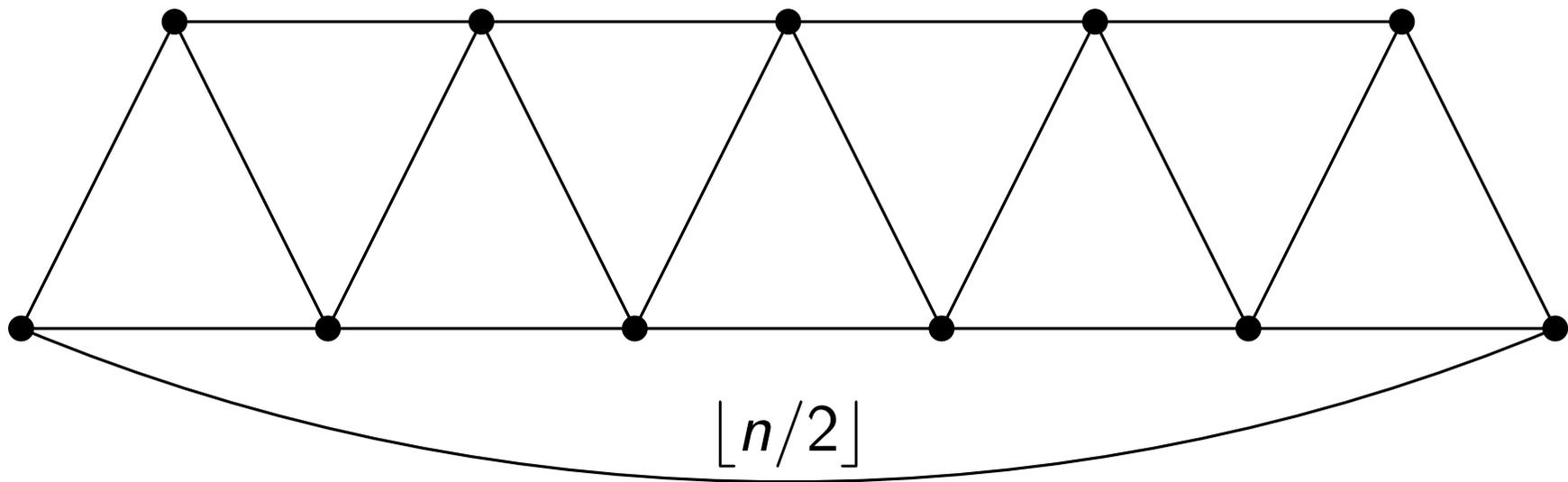
- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges



Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

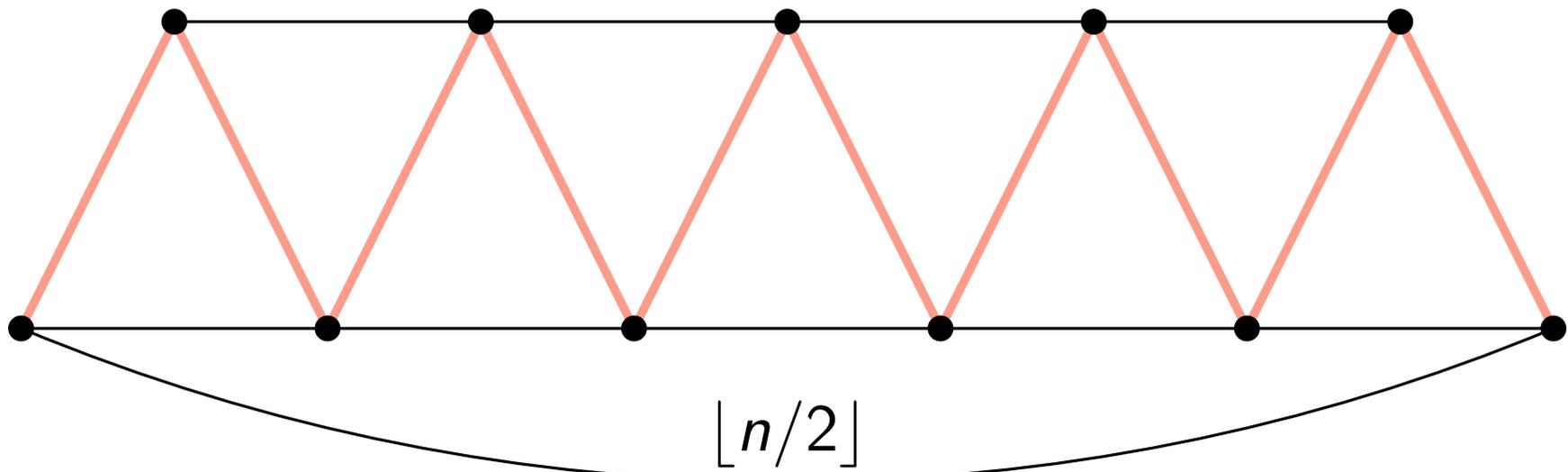
- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)

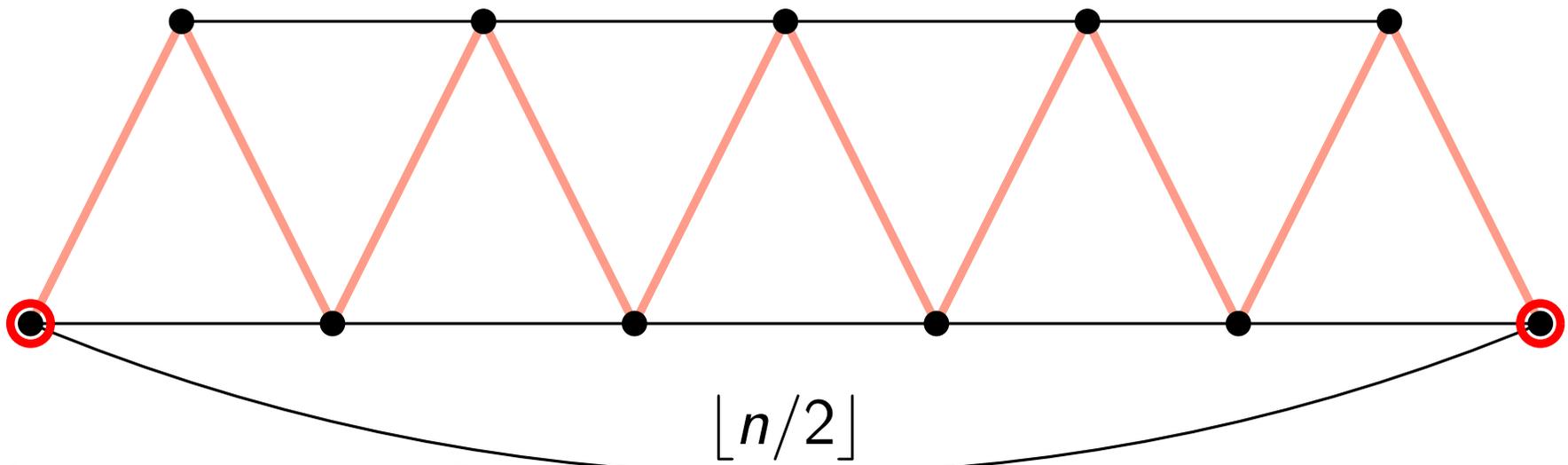


minimaler Spannbaum B

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)

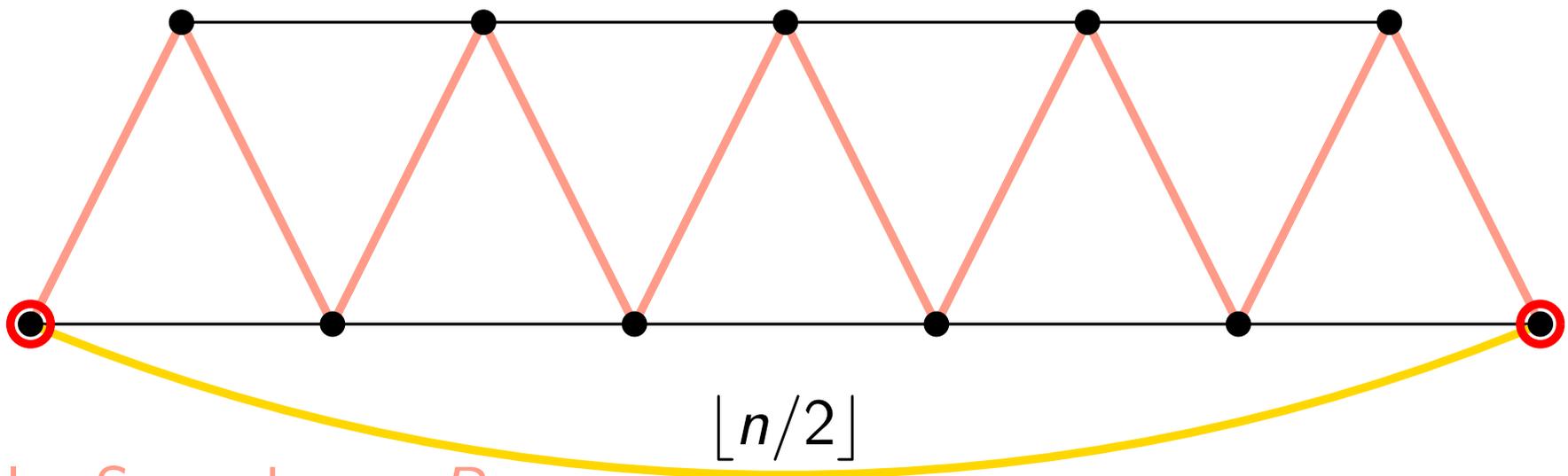


minimaler Spannbaum B

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



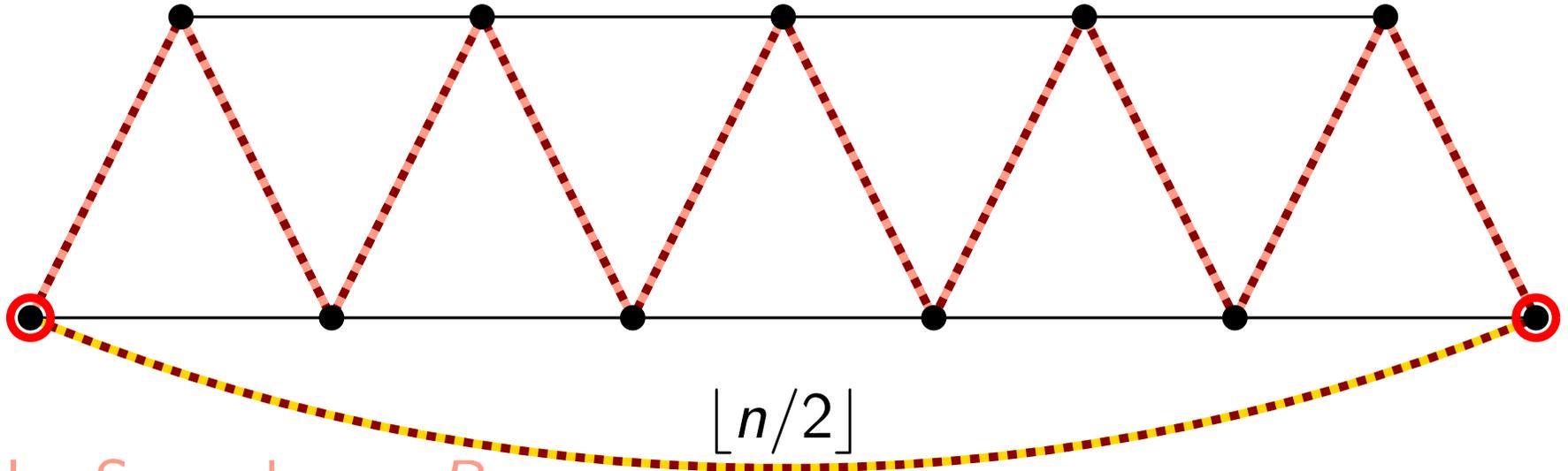
minimaler Spannbaum B

perfektes Matching M in $G[U]$

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



minimaler Spannbaum B

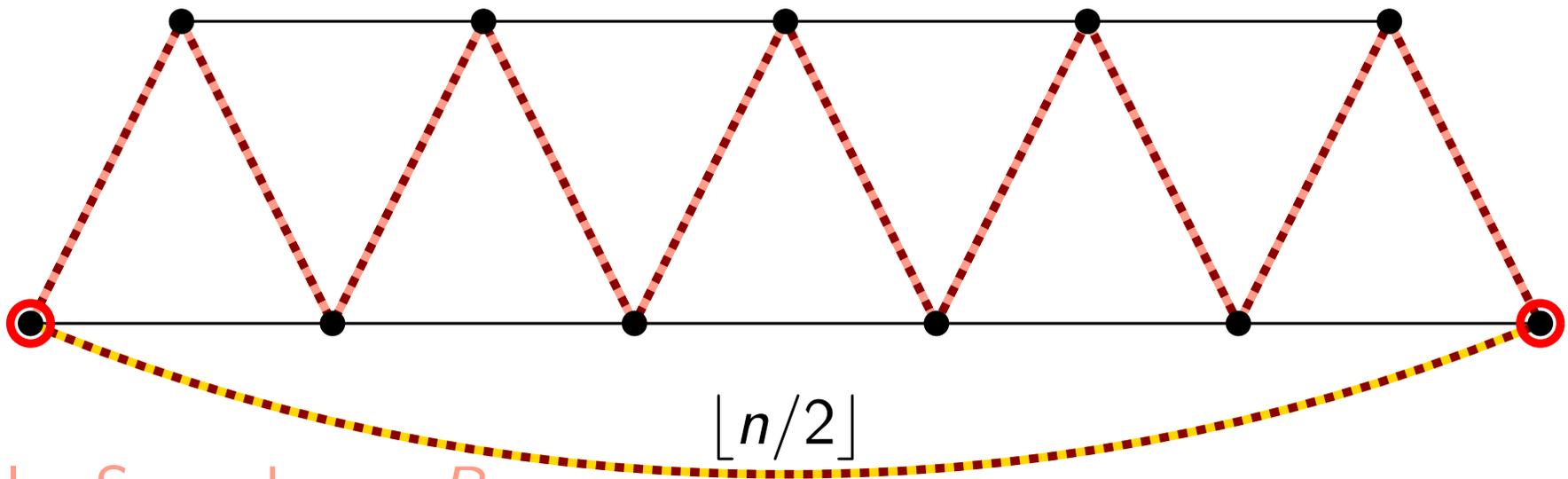
perfektes Matching M in $G[U]$

Christofides-Tour T mit Kosten $ALG =$

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



minimaler Spannbaum B

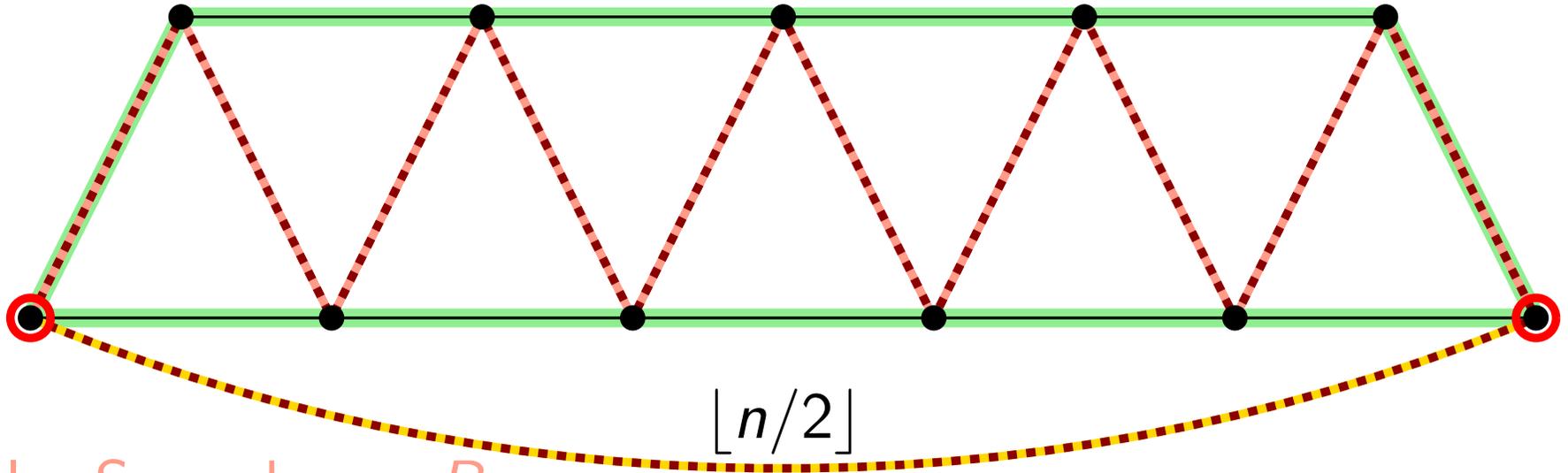
perfektes Matching M in $G[U]$

Christofides-Tour T mit Kosten $ALG = n + \lfloor n/2 \rfloor - 1$

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



minimaler Spannbaum B

perfektes Matching M in $G[U]$

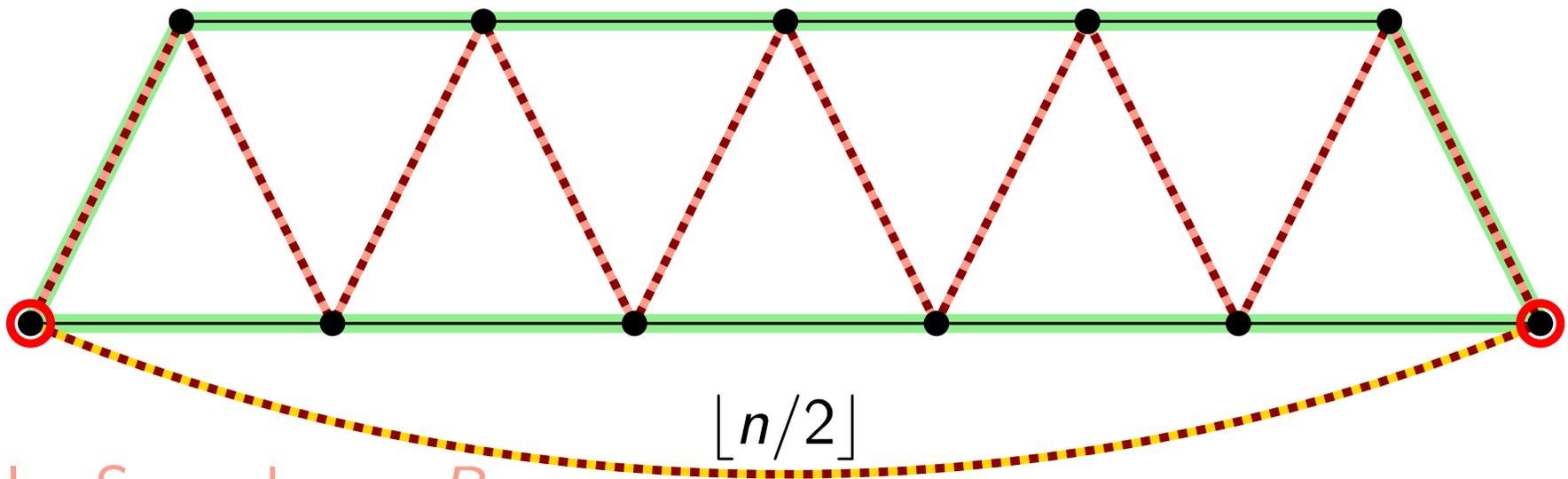
Christofides-Tour T mit Kosten $ALG = n + \lfloor n/2 \rfloor - 1$

$OPT =$

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



minimaler Spannbaum B

perfektes Matching M in $G[U]$

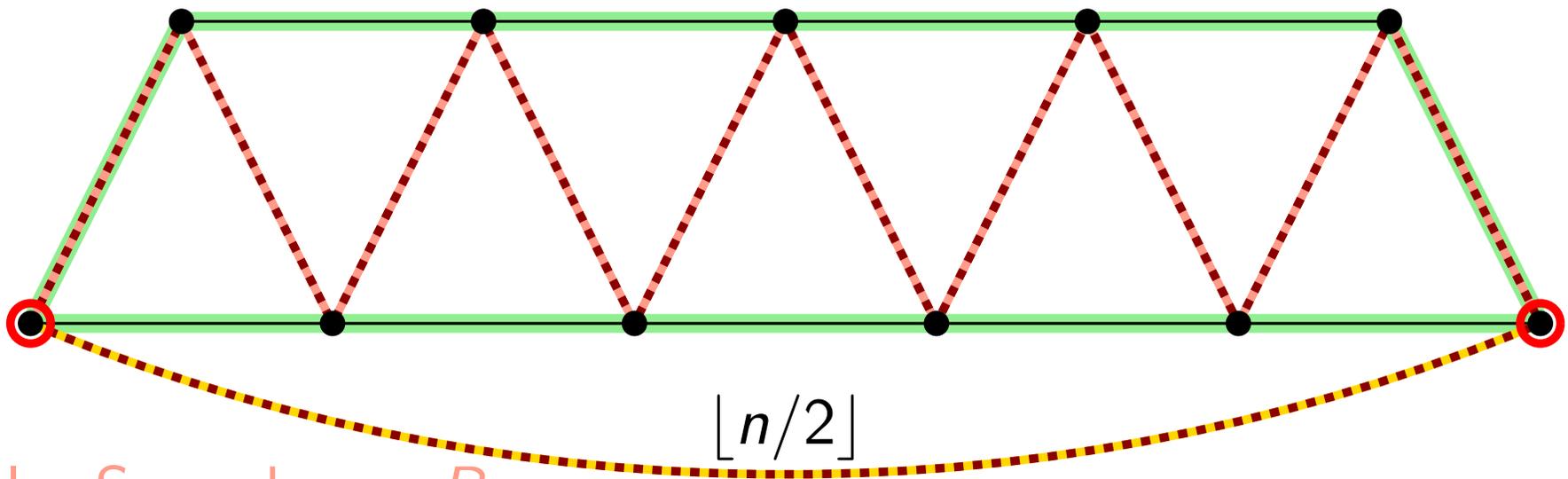
Christofides-Tour T mit Kosten $ALG = n + \lfloor n/2 \rfloor - 1$

$OPT = n$

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



minimaler Spannbaum B

perfektes Matching M in $G[U]$

Christofides-Tour T mit Kosten $ALG = n + \lfloor n/2 \rfloor - 1$

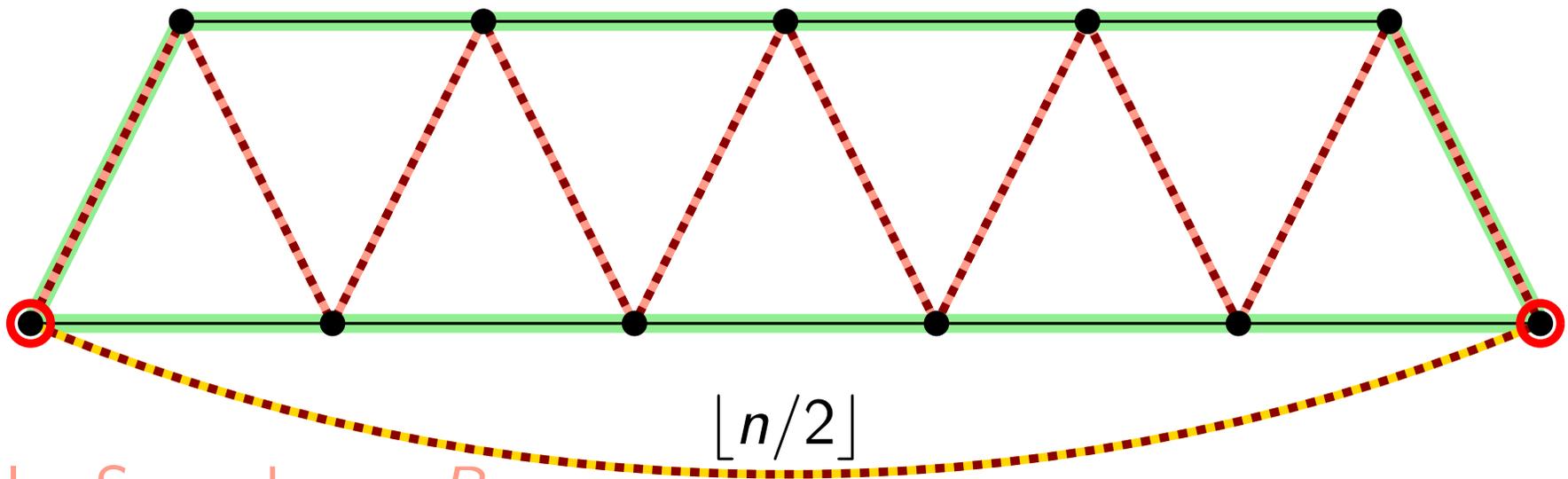
$OPT = n$

$ALG / OPT = (n + \lfloor n/2 \rfloor - 1) / n \rightarrow$

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



minimaler Spannbaum B

perfektes Matching M in $G[U]$

Christofides-Tour T mit Kosten $ALG = n + \lfloor n/2 \rfloor - 1$

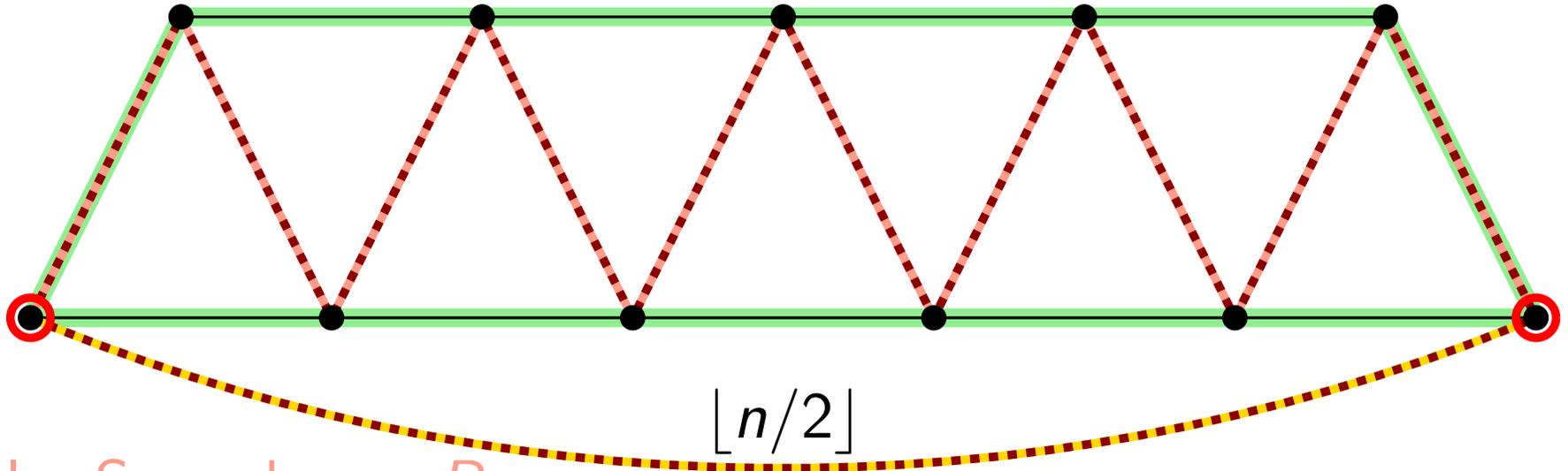
$OPT = n$

$ALG / OPT = (n + \lfloor n/2 \rfloor - 1) / n \rightarrow 3/2$

Der Approximationsfaktor ist scharf

Konstruiere metrische Instanz G , für die Christofides möglichst *schlecht* ist:

- Unbeschriftete Kanten haben Kosten 1.
- Jede nicht gezeichnete Kante uv hat Kosten...
Länge eines kürzesten u - v -Weges ($\Rightarrow G$ metrisch!)



minimaler Spannbaum B

perfektes Matching M in $G[U]$

Christofides-Tour T mit Kosten $ALG = n + \lfloor n/2 \rfloor - 1$

$OPT = n$

$ALG / OPT = (n + \lfloor n/2 \rfloor - 1) / n \rightarrow 3/2$

Also:

Nicht die Analyse ist schlecht –
sondern der Algorithmus!

Kostenminimale perfekte Matchings

Def. Gegeben: vollständiger Graph $G = (V, E)$ mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$.

Gesucht: perfektes Matching M mit minimalen Kosten $c(M) = \sum_{e \in M} c(e)$.

Kostenminimale perfekte Matchings

Def. Gegeben: vollständiger Graph $G = (V, E)$ mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$.
Gesucht: perfektes Matching M mit minimalen Kosten $c(M) = \sum_{e \in M} c(e)$.

Satz. Ein kostenminimales perfektes Matching kann in $O(V^3)$ Zeit berechnet werden.

Kostenminimale perfekte Matchings

Def. Gegeben: vollständiger Graph $G = (V, E)$ mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$.
Gesucht: perfektes Matching M mit minimalen Kosten $c(M) = \sum_{e \in M} c(e)$.

Satz. Ein kostenminimales perfektes Matching kann in $O(V^3)$ Zeit berechnet werden.

Beweis. Siehe [Edmonds '65]; ziemlich kompliziert :- (□

Kostenminimale perfekte Matchings

Def. Gegeben: vollständiger Graph $G = (V, E)$ mit Kantenkosten $c: E \rightarrow \mathbb{R}_{\geq 0}$.
Gesucht: perfektes Matching M mit minimalen Kosten $c(M) = \sum_{e \in M} c(e)$.

Satz. Ein kostenminimales perfektes Matching kann in $O(V^3)$ Zeit berechnet werden.

Beweis. Siehe [Edmonds '65]; ziemlich kompliziert :- (□

Im Folgenden betrachten wir das Problem nur in *bipartiten* Graphen $G = (A \cup B, E)$.

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!

Minimiere

unter den Nebenbed.

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!

Minimiere

unter den Nebenbed.

$$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$$

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!

Minimiere

$$\sum_{uv \in E} c_{uv} x_{uv}$$

unter den Nebenbed.

$$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$$

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \cup B, E)$ bipartit!

Minimiere

$$\sum_{uv \in E} c_{uv} x_{uv}$$

unter den Nebenbed.

$$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$$

$$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$$

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!

Minimiere

$$\sum_{uv \in E} c_{uv} x_{uv}$$

unter den Nebenbed.

$$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$$

$$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$$

- Effizient lösbar?

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$

- Effizient lösbar? I.A. nicht – VC is Spezialfall von ILP.

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!

Minimiere

$$\sum_{uv \in E} c_{uv} x_{uv}$$

unter den Nebenbed.

$$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$$

~~$$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$$~~

$$x_{uv} \geq 0$$

- Effizient lösbar? I.A. nicht – VC is Spezialfall von ILP.
- Betrachte sogenannte *LP-Relaxierung* \Rightarrow effizient lösbar!

Kostenminimale perfekte Matchings in bipartiten Graphen

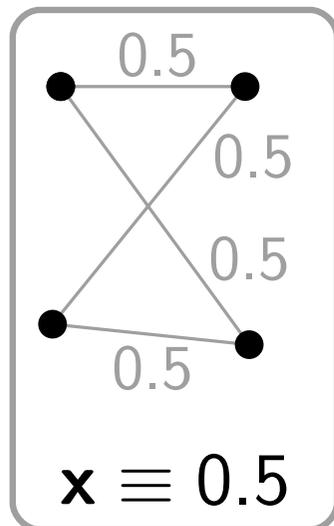
- Aufgabe: Formulieren Sie ein ILP für $G = (A \cup B, E)$ bipartit!

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$
	$x_{uv} \geq 0$

- Effizient lösbar? I.A. nicht – VC is Spezialfall von ILP.
- Betrachte sogenannte *LP-Relaxierung* \Rightarrow effizient lösbar!
Bei Minimierungsproblemen gilt $\text{OPT}_{\text{LP}} \leq \text{OPT}_{\text{ILP}}$.

Kostenminimale perfekte Matchings in bipartiten Graphen

- Aufgabe: Formulieren Sie ein ILP für $G = (A \dot{\cup} B, E)$ bipartit!



Minimiere

$$\sum_{uv \in E} c_{uv} x_{uv}$$

unter den Nebenbed.

$$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$$

~~$$x_{uv} \in \{0, 1\} \quad \text{für } uv \in E$$~~

$$x_{uv} \geq 0$$

- Effizient lösbar? I.A. nicht – VC is Spezialfall von ILP.
- Betrachte sogenannte *LP-Relaxierung* \Rightarrow effizient lösbar!
Bei Minimierungsproblemen gilt $\text{OPT}_{\text{LP}} \leq \text{OPT}_{\text{ILP}}$.
- Problem: *fraktionale* Lösungen!

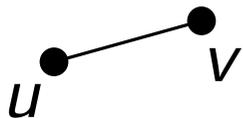
LP-Runden

$$\begin{array}{ll} \text{Minimiere} & \sum_{uv \in E} c_{uv} x_{uv} \\ \text{unter den Nebenbed.} & \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\ & x_{uv} \geq 0 \quad \text{für } uv \in E \end{array}$$

LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

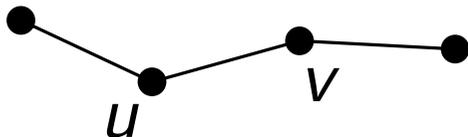


LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.



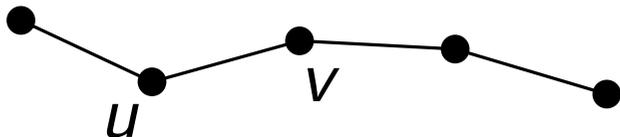
LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.

Erweitere Pfad iterativ, bis fraktionaler Kreis $K = (u_1, \dots, u_{2k})$ gefunden.



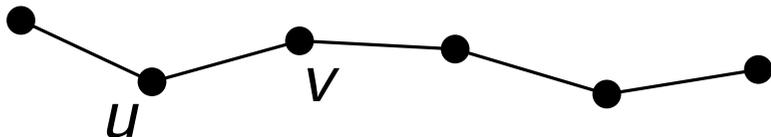
LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.

Erweitere Pfad iterativ, bis fraktionaler Kreis $K = (u_1, \dots, u_{2k})$ gefunden.



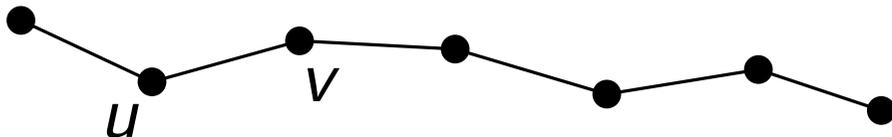
LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.

Erweitere Pfad iterativ, bis fraktionaler Kreis $K = (u_1, \dots, u_{2k})$ gefunden.



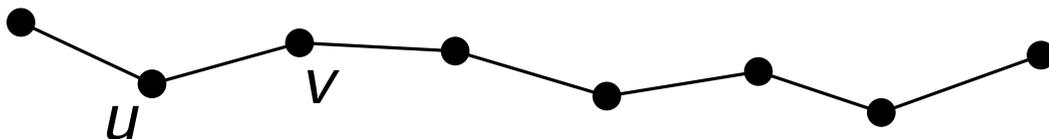
LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.

Erweitere Pfad iterativ, bis fraktionaler Kreis $K = (u_1, \dots, u_{2k})$ gefunden.



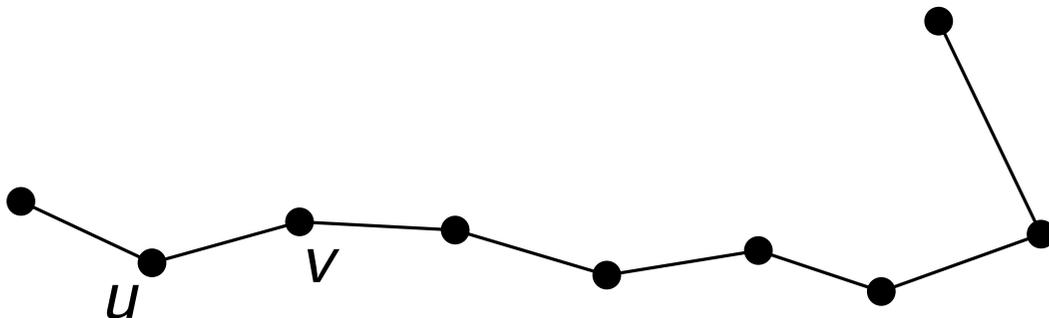
LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.

Erweitere Pfad iterativ, bis fraktionaler Kreis $K = (u_1, \dots, u_{2k})$ gefunden.



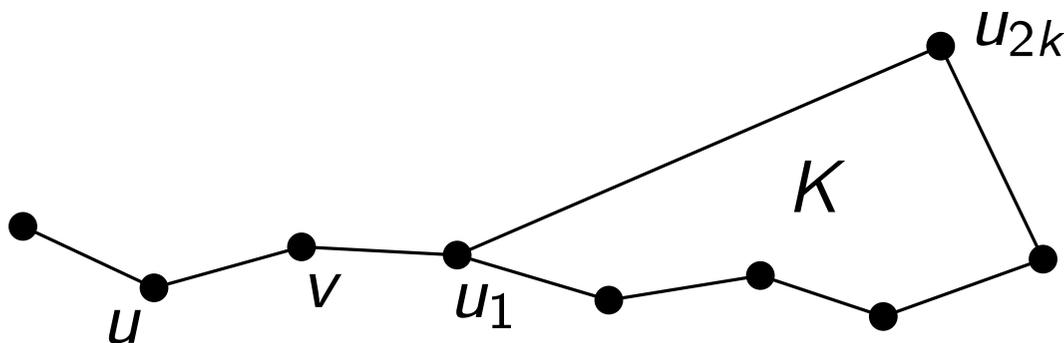
LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.

Erweitere Pfad iterativ, bis fraktionaler Kreis $K = (u_1, \dots, u_{2k})$ gefunden.



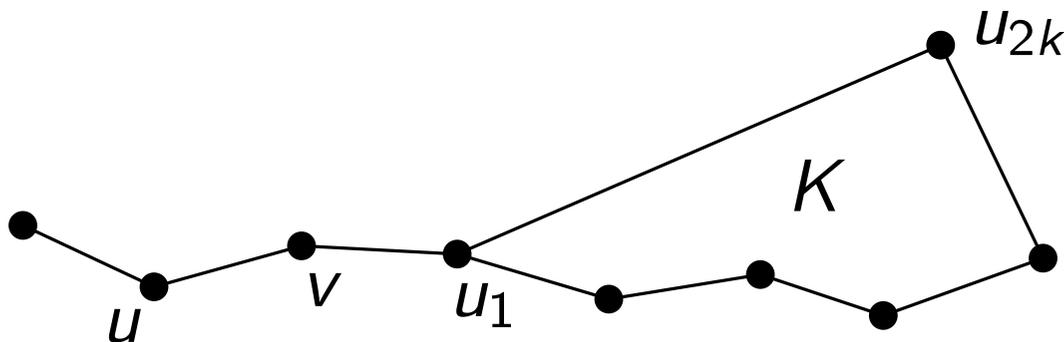
LP-Runden

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Betrachte *fraktionale* Kante uv (d.h. $0 < x_{uv} < 1$):

u und v sind jeweils zu weiterer fraktionaler Kante adjazent.

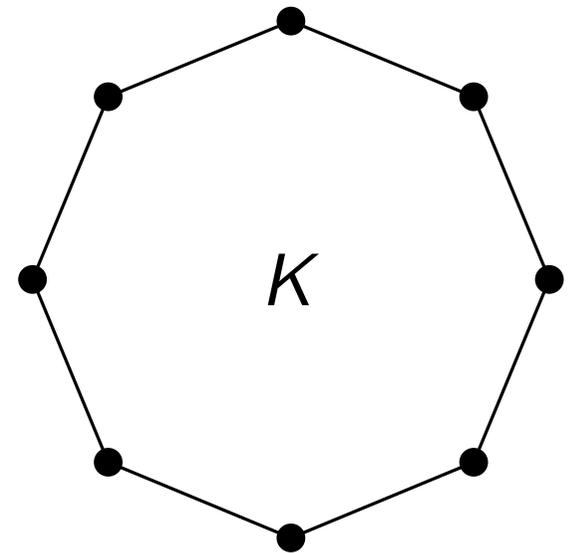
Erweitere Pfad iterativ, bis fraktionaler Kreis $K = (u_1, \dots, u_{2k})$ gefunden.



Kreis gerade, da Graph bipartit (siehe ÜA).

LP-Runden – Fortsetzung (I)

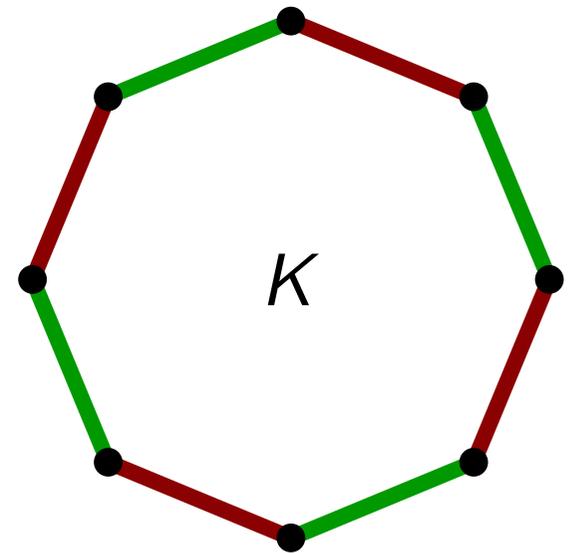
$$\begin{array}{ll} \text{Minimiere} & \sum_{uv \in E} c_{uv} x_{uv} \\ \text{unter den Nebenbed.} & \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\ & x_{uv} \geq 0 \quad \text{für } uv \in E \end{array}$$



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

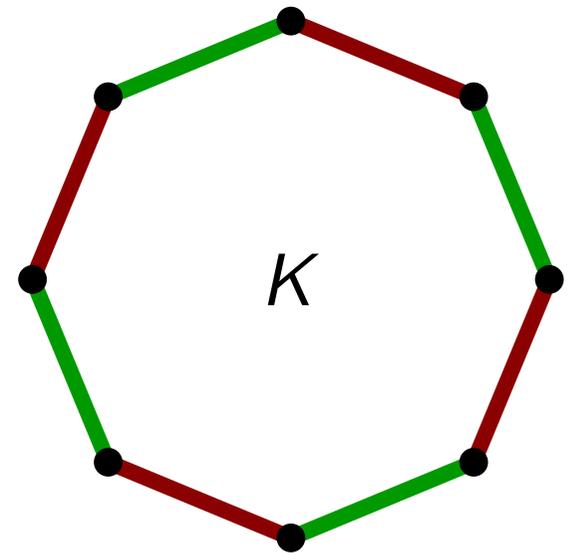
Zerlege K in disjunkte Matchings M_1, M_2 .



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

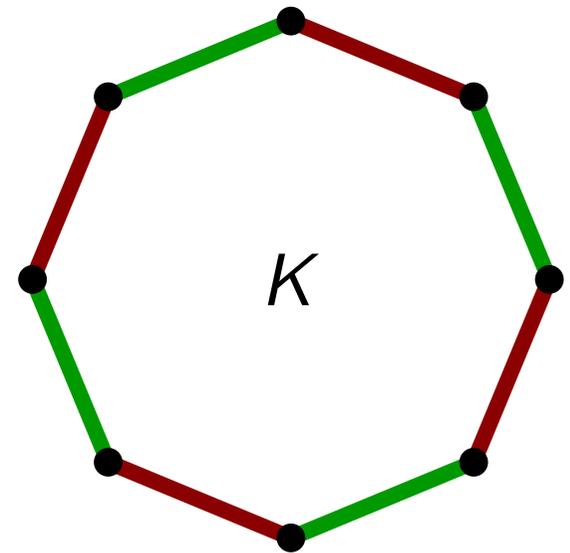
Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.
 Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.



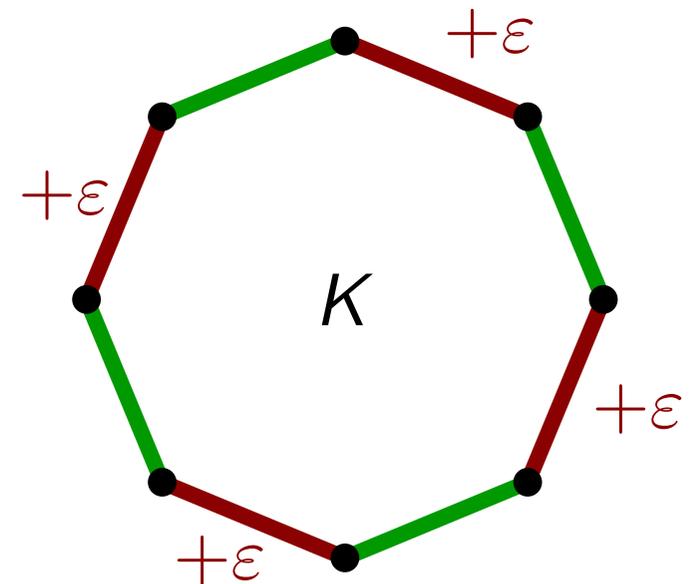
LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze $x'_e := x_e + \varepsilon$ für $e \in M_1$,



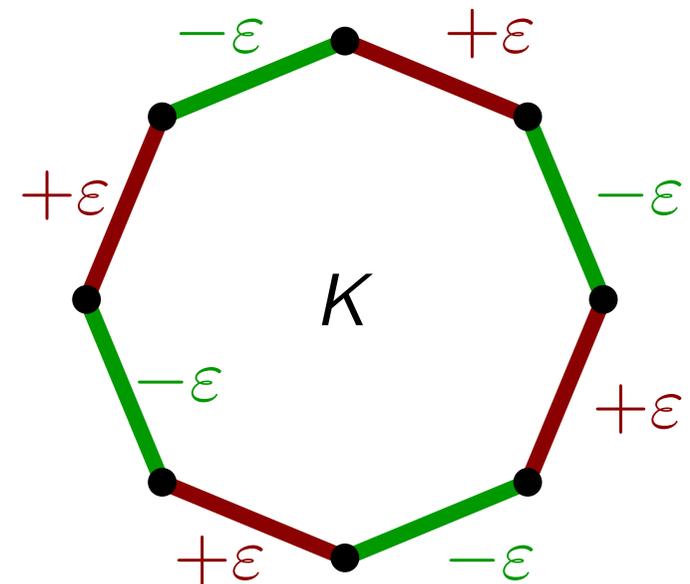
LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze $x'_e := x_e + \varepsilon$ für $e \in M_1$,
 $x'_e := x_e - \varepsilon$ für $e \in M_2$,



LP-Runden – Fortsetzung (I)

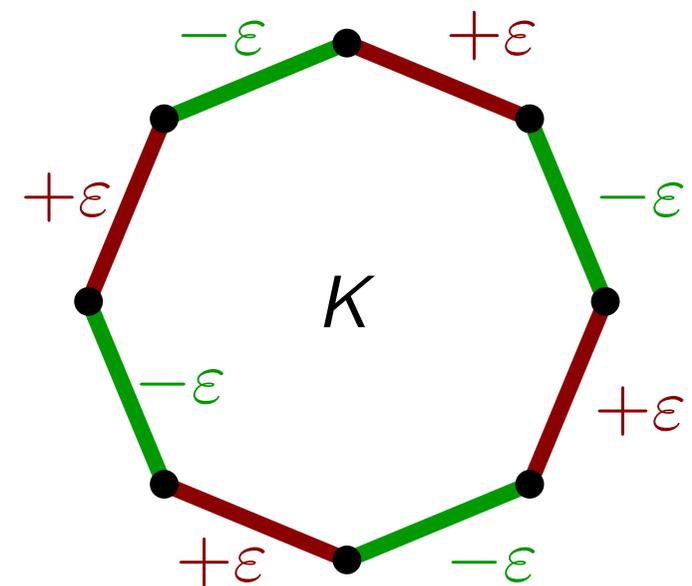
Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

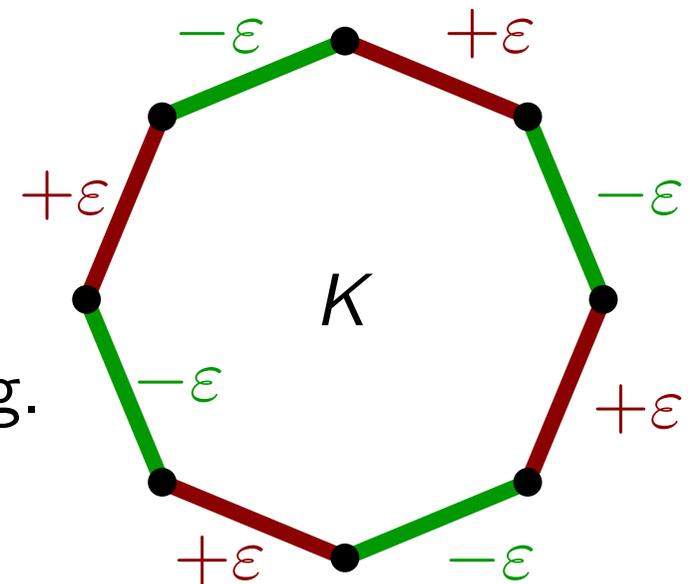
Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

– Resultierender Lösungsvektor \mathbf{x}' zulässig.



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

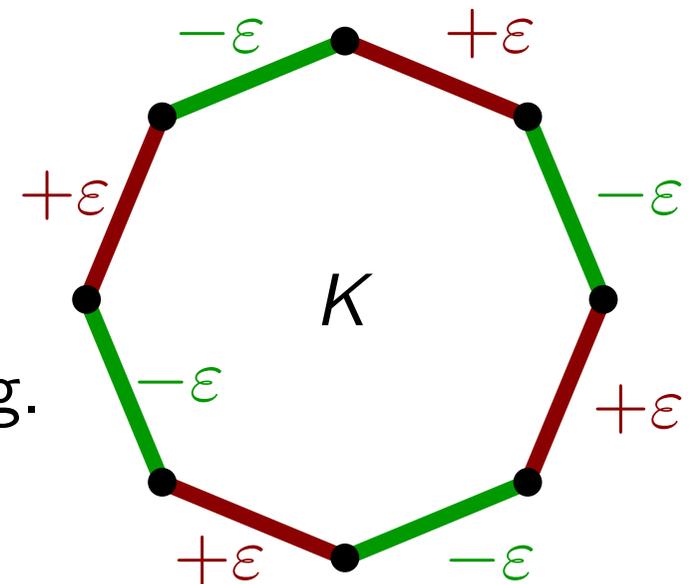
Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

- Resultierender Lösungsvektor \mathbf{x}' zulässig.
- Kostenänderung:



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

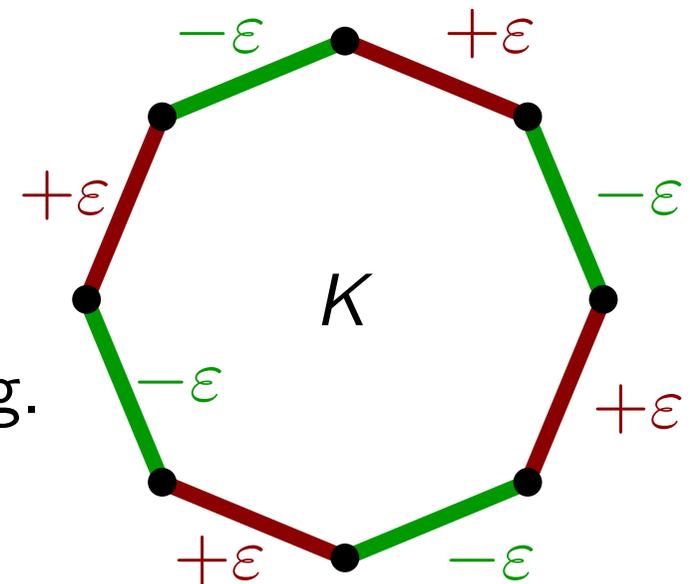
Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

- Resultierender Lösungsvektor \mathbf{x}' zulässig.
- Kostenänderung: $\varepsilon \cdot c(M_1)$



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

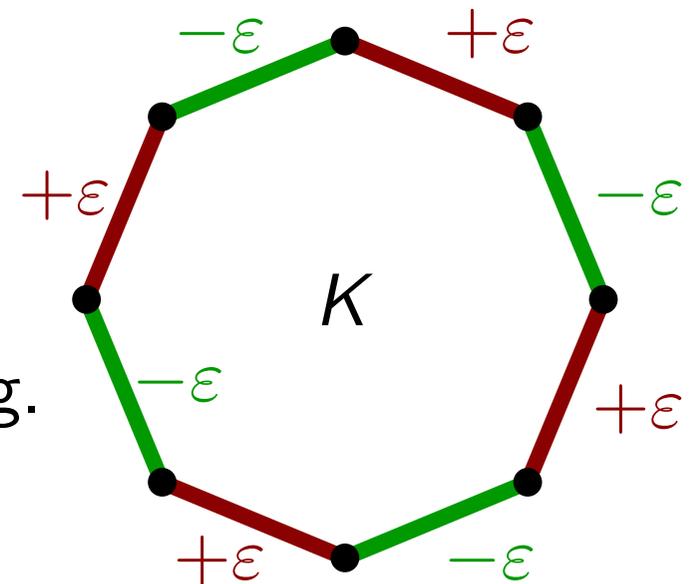
Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

- Resultierender Lösungsvektor \mathbf{x}' zulässig.
- Kostenänderung: $\varepsilon \cdot c(M_1) - \varepsilon \cdot c(M_2)$



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

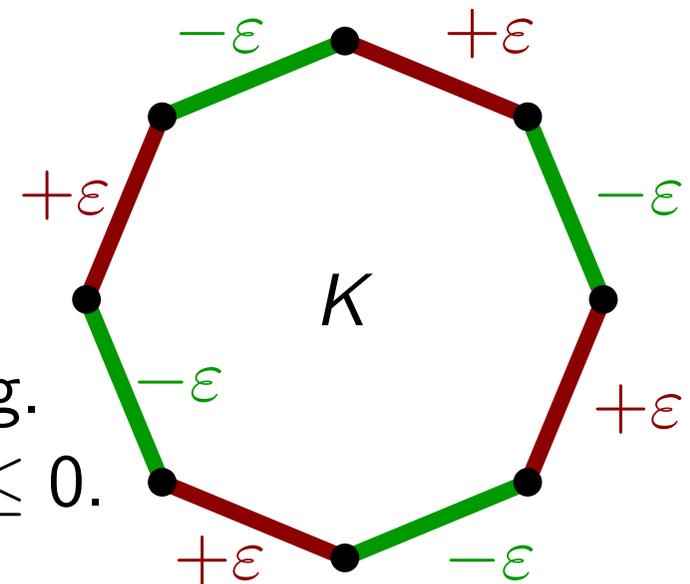
Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

- Resultierender Lösungsvektor \mathbf{x}' zulässig.
- Kostenänderung: $\varepsilon \cdot c(M_1) - \varepsilon \cdot c(M_2) \leq 0$.



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

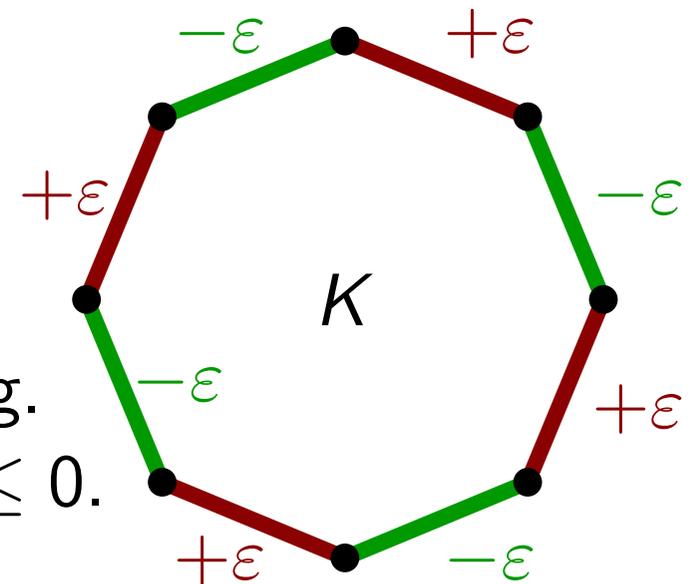
Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

- Resultierender Lösungsvektor \mathbf{x}' zulässig.
- Kostenänderung: $\varepsilon \cdot c(M_1) - \varepsilon \cdot c(M_2) \leq 0$.

$\Rightarrow \mathbf{x}'$ ist auch optimal



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

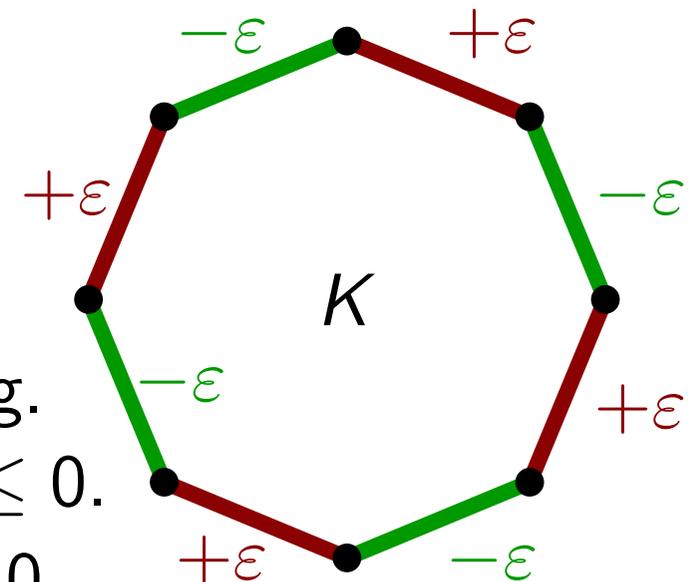
Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

- Resultierender Lösungsvektor \mathbf{x}' zulässig.
 - Kostenänderung: $\varepsilon \cdot c(M_1) - \varepsilon \cdot c(M_2) \leq 0$.
 - Für mind. eine Kante $e \in M_2$ gilt $x'_e = 0$.
- $\Rightarrow \mathbf{x}'$ ist auch optimal



LP-Runden – Fortsetzung (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 . Sei $c(M_1) \leq c(M_2)$.

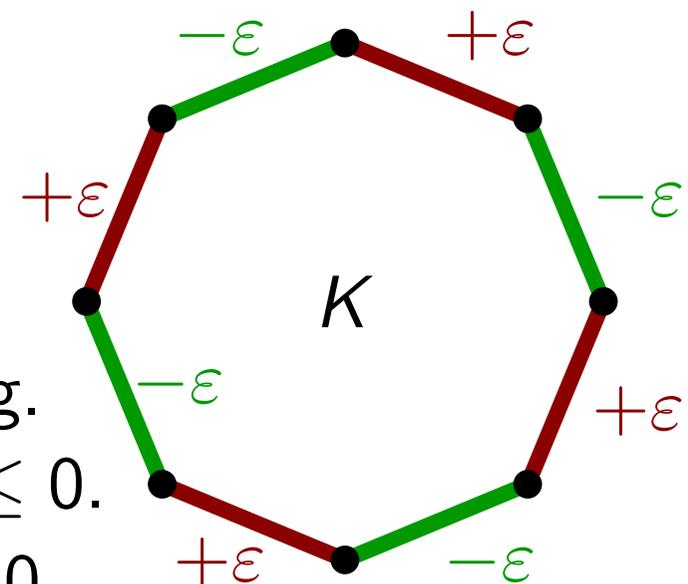
Wähle $\varepsilon := \min\{x_e \mid e \in M_2\}$.

Setze

$x'_e := x_e + \varepsilon$	für $e \in M_1$,
$x'_e := x_e - \varepsilon$	für $e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

- Resultierender Lösungsvektor \mathbf{x}' zulässig.
- Kostenänderung: $\varepsilon \cdot c(M_1) - \varepsilon \cdot c(M_2) \leq 0$.
- Für mind. eine Kante $e \in M_2$ gilt $x'_e = 0$.

$\Rightarrow \mathbf{x}'$ ist auch optimal und hat mehr integrale Variable als \mathbf{x} !!



LP-Runden – Fortsetzung (II)

$$\begin{array}{ll} \text{Minimiere} & \sum_{uv \in E} c_{uv} x_{uv} \\ \text{unter den Nebenbed.} & \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\ & x_{uv} \geq 0 \quad \text{für } uv \in E \end{array}$$

Wiederhole vorige Schritte, solange fraktionale Kanten existieren.

LP-Runden – Fortsetzung (II)

$$\begin{array}{ll} \text{Minimiere} & \sum_{uv \in E} c_{uv} x_{uv} \\ \text{unter den Nebenbed.} & \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\ & x_{uv} \geq 0 \quad \text{für } uv \in E \end{array}$$

Wiederhole vorige Schritte, solange fraktionale Kanten existieren.

Beob. – Integrale Kanten werden nicht weiter verändert.

LP-Runden – Fortsetzung (II)

$$\begin{array}{ll} \text{Minimiere} & \sum_{uv \in E} c_{uv} x_{uv} \\ \text{unter den Nebenbed.} & \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\ & x_{uv} \geq 0 \quad \text{für } uv \in E \end{array}$$

Wiederhole vorige Schritte, solange fraktionale Kanten existieren.

- Beob.**
- Integrale Kanten werden nicht weiter verändert.
 - Zielfunktion erhöht sich nicht.

LP-Runden – Fortsetzung (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Wiederhole vorige Schritte, solange fraktionale Kanten existieren.

- Beob.**
- Integrale Kanten werden nicht weiter verändert.
 - Zielfunktion erhöht sich nicht.

⇒ Algorithmus terminiert nach Iterationen mit ganzzahliger und optimaler (!) Lösung.

LP-Runden – Fortsetzung (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Wiederhole vorige Schritte, solange fraktionale Kanten existieren.

- Beob.** – Integrale Kanten werden nicht weiter verändert.
 – Zielfunktion erhöht sich nicht.

\Rightarrow Algorithmus terminiert nach $\leq |E|$ Iterationen mit ganzzahliger und optimaler (!) Lösung.

LP-Runden – Fortsetzung (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Wiederhole vorige Schritte, solange fraktionale Kanten existieren.

Beob. – Integrale Kanten werden nicht weiter verändert.
– Zielfunktion erhöht sich nicht.

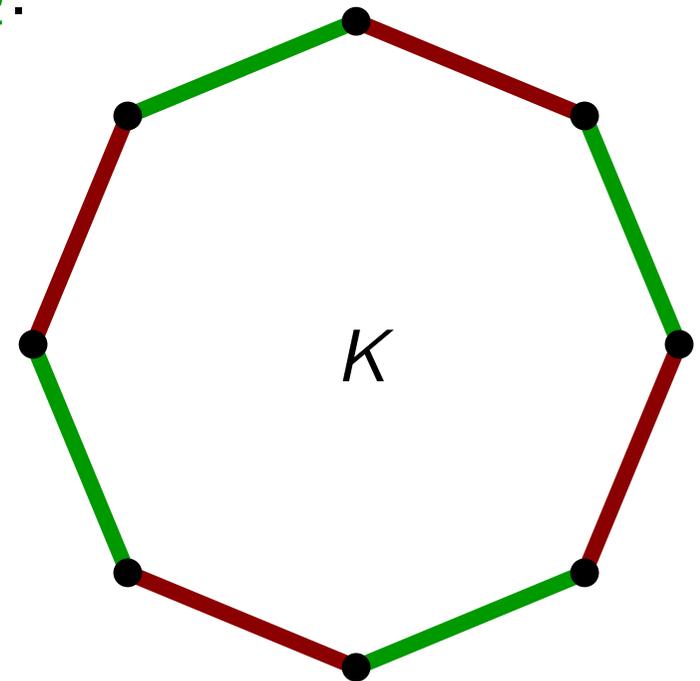
⇒ Algorithmus terminiert nach $\leq |E|$ Iterationen mit ganzzahliger und optimaler (!) Lösung.

Satz. In bipartiten Graphen kann man kostenminimale perfekte Matchings in Polynomialzeit ermitteln.

Extrempunkt-Lösungen (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 .

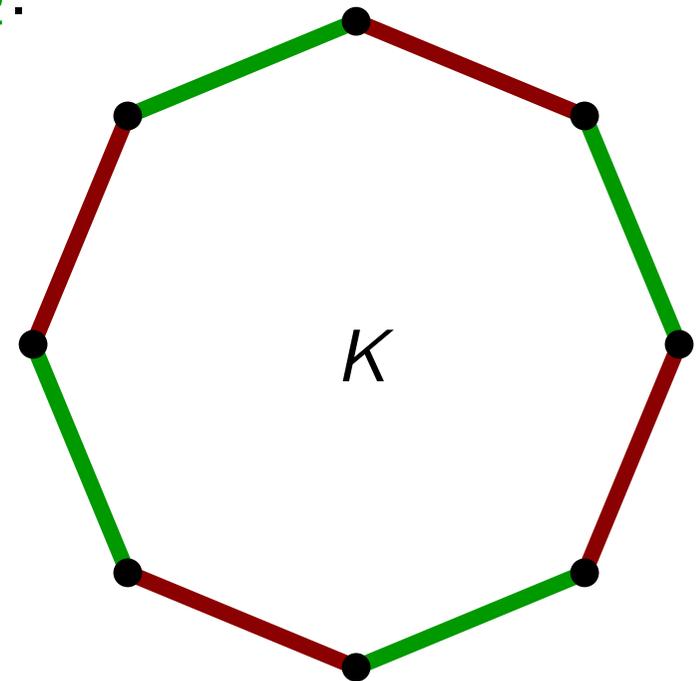


Extrempunkt-Lösungen (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 .

Wähle $\varepsilon := \min\{x_e \mid e \in K\}$.



Extrempunkt-Lösungen (I)

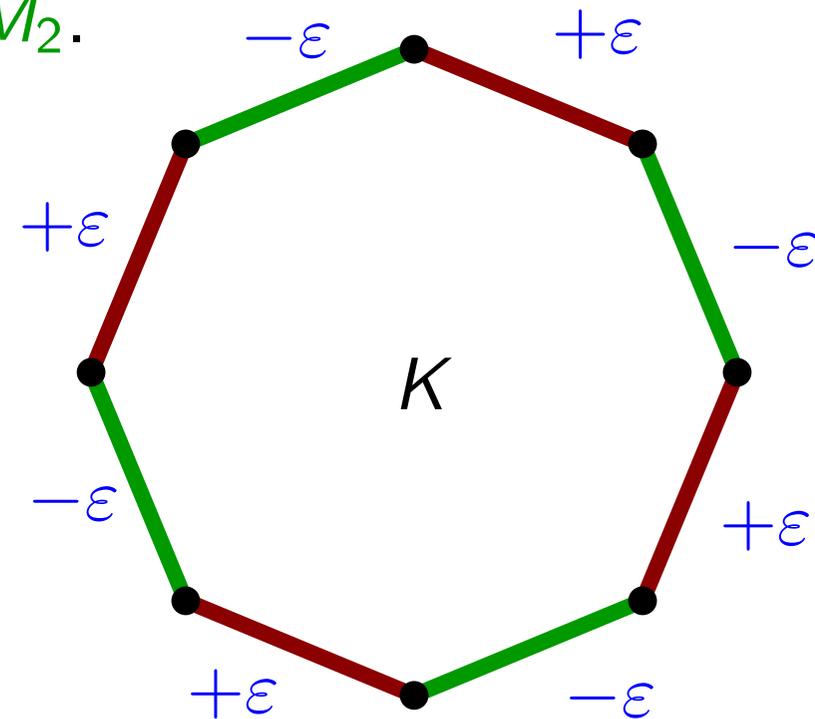
Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 .

Wähle $\varepsilon := \min\{x_e \mid e \in K\}$.

Setze

$x'_e := x_e + \varepsilon$	für $x_e \in M_1$,
$x'_e := x_e - \varepsilon$	für $x_e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.



Extrempunkt-Lösungen (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 .

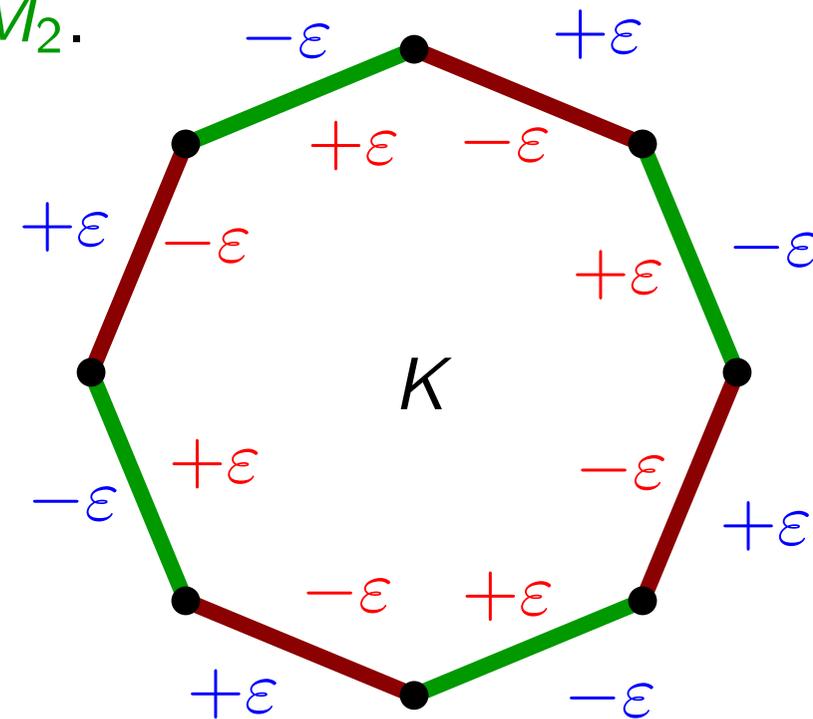
Wähle $\varepsilon := \min\{x_e \mid e \in K\}$.

Setze

$x'_e := x_e + \varepsilon$	für $x_e \in M_1$,
$x'_e := x_e - \varepsilon$	für $x_e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

Setze

$x''_e := x_e - \varepsilon$	für $x_e \in M_1$,
$x''_e := x_e + \varepsilon$	für $x_e \in M_2$.
$x''_e := x_e$	für $e \in E \setminus K$.



Extrempunkt-Lösungen (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 .

Wähle $\varepsilon := \min\{x_e \mid e \in K\}$.

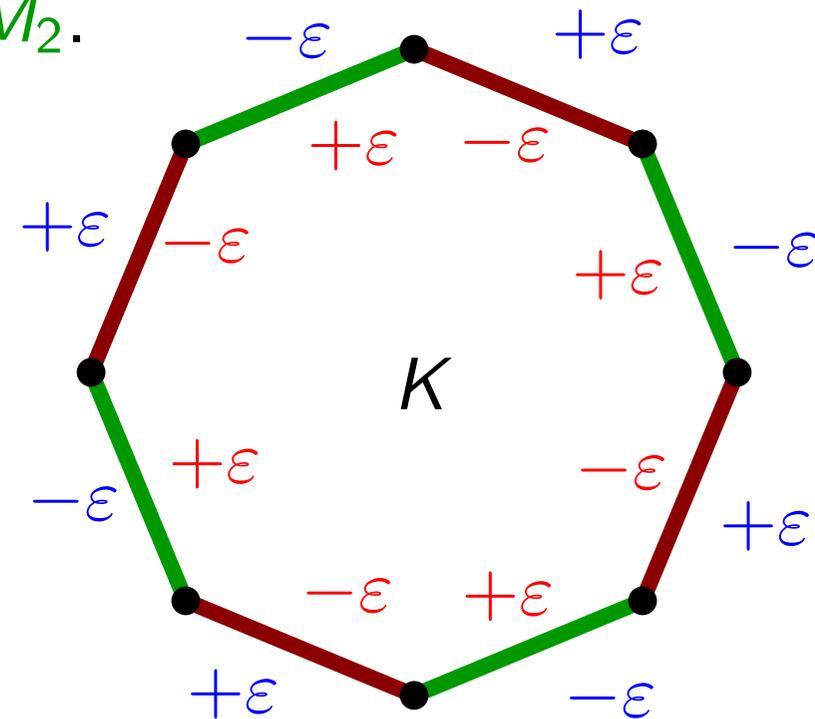
Setze

$x'_e := x_e + \varepsilon$	für $x_e \in M_1$,
$x'_e := x_e - \varepsilon$	für $x_e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

Setze

$x''_e := x_e - \varepsilon$	für $x_e \in M_1$,
$x''_e := x_e + \varepsilon$	für $x_e \in M_2$.
$x''_e := x_e$	für $e \in E \setminus K$.

Beob.: Lösungen \mathbf{x}' , \mathbf{x}'' sind beide zulässig



Extrempunkt-Lösungen (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 .

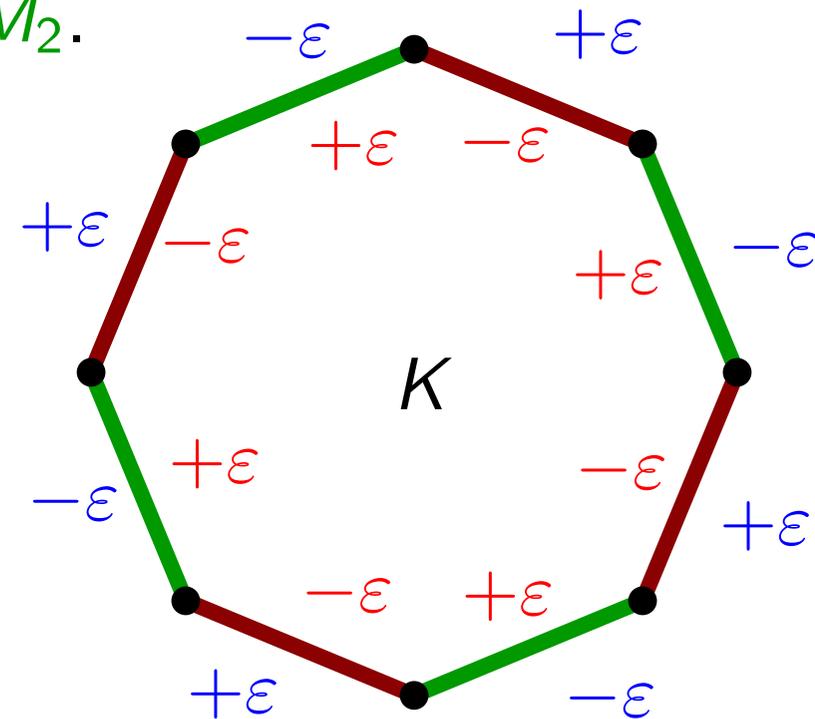
Wähle $\varepsilon := \min\{x_e \mid e \in K\}$.

Setze

$x'_e := x_e + \varepsilon$	für $x_e \in M_1$,
$x'_e := x_e - \varepsilon$	für $x_e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

Setze

$x''_e := x_e - \varepsilon$	für $x_e \in M_1$,
$x''_e := x_e + \varepsilon$	für $x_e \in M_2$.
$x''_e := x_e$	für $e \in E \setminus K$.



Beob.: Lösungen \mathbf{x}' , \mathbf{x}'' sind beide zulässig und $\mathbf{x} =$

Extrempunkt-Lösungen (I)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

Zerlege K in disjunkte Matchings M_1, M_2 .

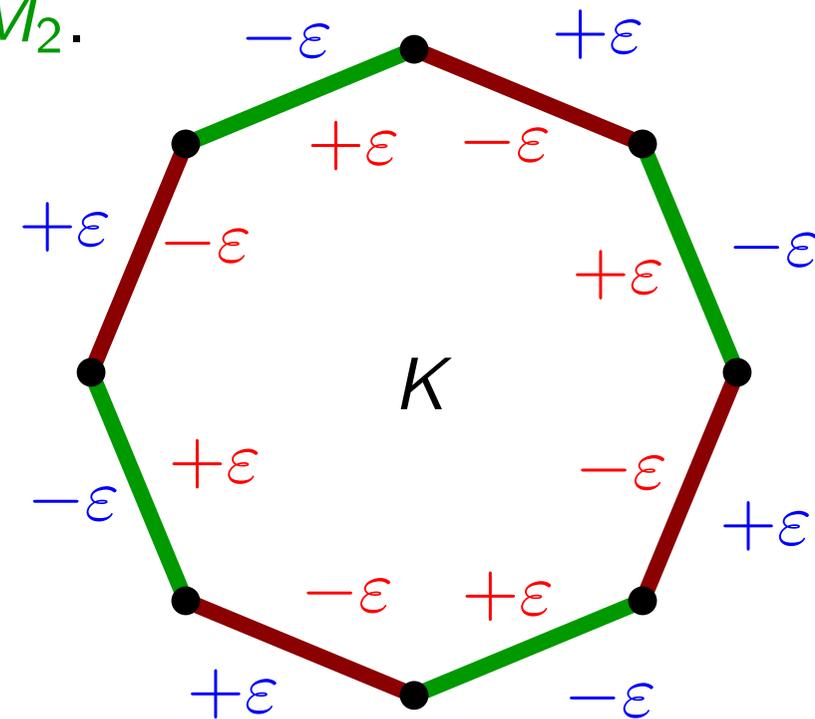
Wähle $\varepsilon := \min\{x_e \mid e \in K\}$.

Setze

$x'_e := x_e + \varepsilon$	für $x_e \in M_1$,
$x'_e := x_e - \varepsilon$	für $x_e \in M_2$,
$x'_e := x_e$	für $e \in E \setminus K$.

Setze

$x''_e := x_e - \varepsilon$	für $x_e \in M_1$,
$x''_e := x_e + \varepsilon$	für $x_e \in M_2$.
$x''_e := x_e$	für $e \in E \setminus K$.



Beob.: Lösungen \mathbf{x}' , \mathbf{x}'' sind beide zulässig und $\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'')$!

Extrempunkt-Lösungen (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

$$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x} \text{ liegt}$$

Extrempunkt-Lösungen (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

$$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x} \text{ liegt auf der Strecke } \overline{\mathbf{x}'\mathbf{x}''}.$$

Extrempunkt-Lösungen (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x}$ liegt auf der Strecke $\overline{\mathbf{x}'\mathbf{x}''}$.

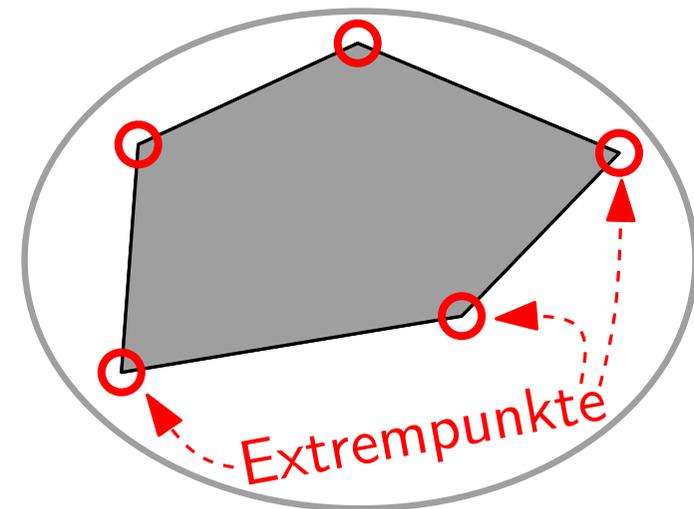
Außerdem sind x , x' und x'' paarweise verschieden.

Extrempunkt-Lösungen (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x}$ liegt auf der Strecke $\overline{x'x''}$.

Außerdem sind x , x' und x'' paarweise verschieden.

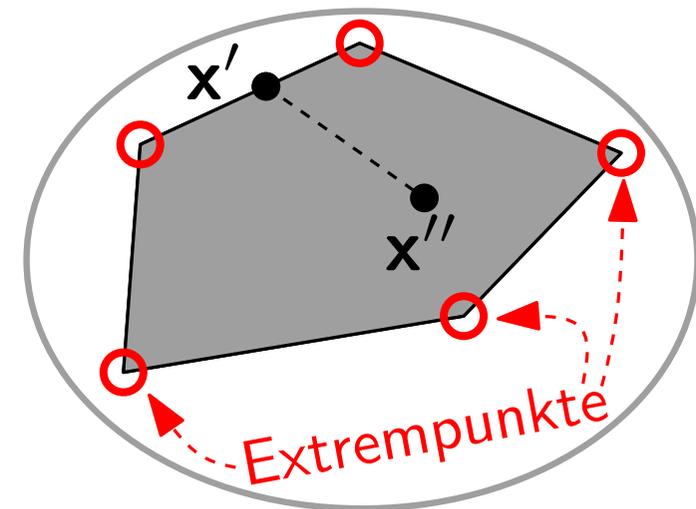


Extrempunkt-Lösungen (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x}$ liegt auf der Strecke $\overline{\mathbf{x}'\mathbf{x}''}$.

Außerdem sind \mathbf{x} , \mathbf{x}' und \mathbf{x}'' paarweise verschieden.

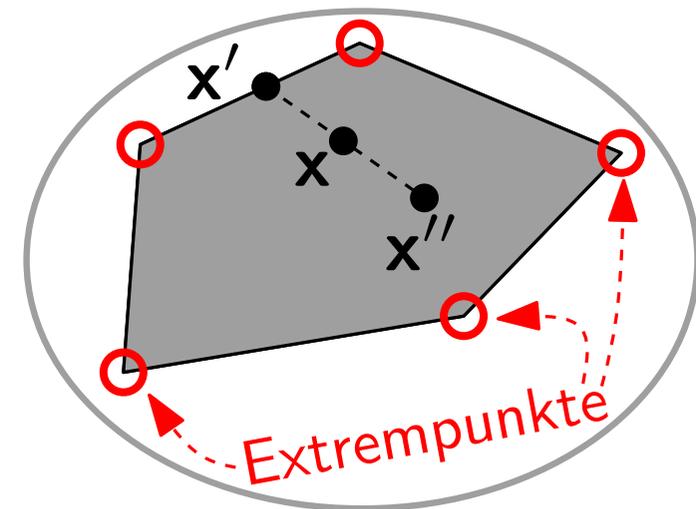


Extrempunkt-Lösungen (II)

Minimiere	$\sum_{uv \in E} c_{uv} x_{uv}$
unter den Nebenbed.	$\sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B$
	$x_{uv} \geq 0 \quad \text{für } uv \in E$

$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x}$ liegt auf der Strecke $\overline{\mathbf{x}'\mathbf{x}''}$.

Außerdem sind \mathbf{x} , \mathbf{x}' und \mathbf{x}'' paarweise verschieden.



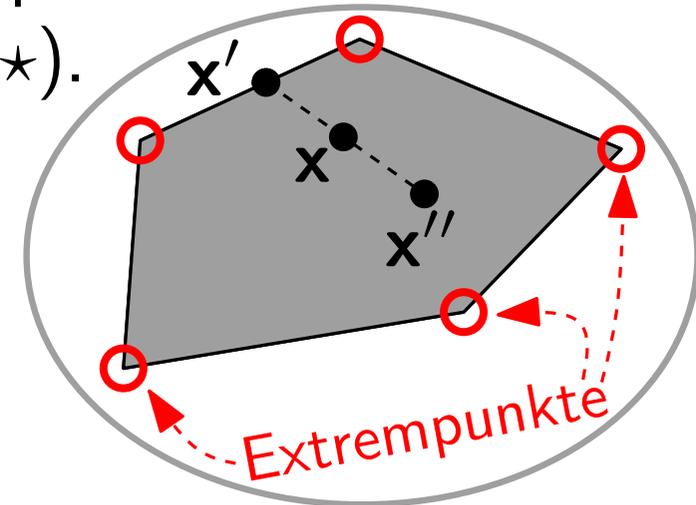
Extrempunkt-Lösungen (II)

$$\begin{array}{l}
 \text{Minimiere} \\
 \text{unter den Nebenbed.}
 \end{array}
 \left.
 \begin{array}{l}
 \sum_{uv \in E} c_{uv} x_{uv} \\
 \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\
 x_{uv} \geq 0 \quad \text{für } uv \in E
 \end{array}
 \right\} (*)$$

$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x}$ liegt auf der Strecke $\overline{\mathbf{x}'\mathbf{x}''}$.

Außerdem sind \mathbf{x} , \mathbf{x}' und \mathbf{x}'' paarweise verschieden.

\Rightarrow Die fraktionale Lösung \mathbf{x} ist *kein* Extrempunkt des Lösungsraums (konvexes Polytop) von $(*)$.



Extrempunkt-Lösungen (II)

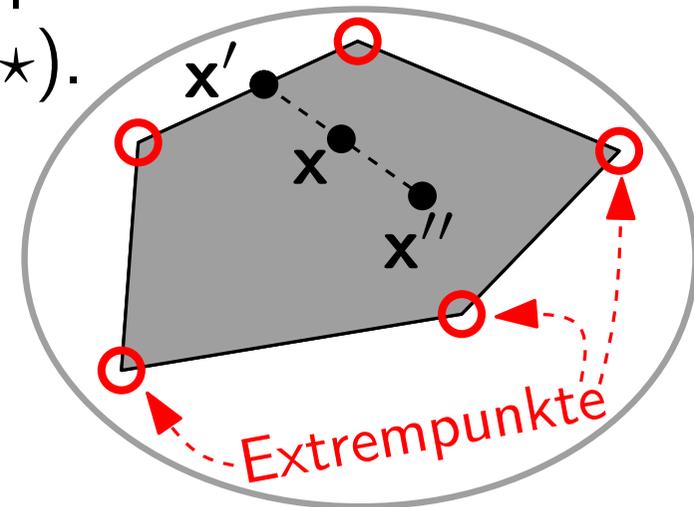
$$\begin{array}{l}
 \text{Minimiere} \\
 \text{unter den Nebenbed.}
 \end{array}
 \left.
 \begin{array}{l}
 \sum_{uv \in E} c_{uv} x_{uv} \\
 \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\
 x_{uv} \geq 0 \quad \text{für } uv \in E
 \end{array}
 \right\} (*)$$

$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x}$ liegt auf der Strecke $\overline{\mathbf{x}'\mathbf{x}''}$.

Außerdem sind \mathbf{x} , \mathbf{x}' und \mathbf{x}'' paarweise verschieden.

\Rightarrow Die fraktionale Lösung \mathbf{x} ist *kein* Extrempunkt des Lösungsraums (konvexes Polytop) von $(*)$.

Satz. Polytop $(*)$ für bipartite Matchings hat nur *ganzzahlige* Extrempunkte.



Extrempunkt-Lösungen (II)

$$\begin{array}{l}
 \text{Minimiere} \\
 \text{unter den Nebenbed.}
 \end{array}
 \left.
 \begin{array}{l}
 \sum_{uv \in E} c_{uv} x_{uv} \\
 \sum_{v \in \text{Adj}[u]} x_{uv} = 1 \quad \text{für } u \in A \cup B \\
 x_{uv} \geq 0 \quad \text{für } uv \in E
 \end{array}
 \right\} (*)$$

$\mathbf{x} = \frac{1}{2}(\mathbf{x}' + \mathbf{x}'') \Rightarrow \mathbf{x}$ liegt auf der Strecke $\overline{x'x''}$.

Außerdem sind x , x' und x'' paarweise verschieden.

\Rightarrow Die fraktionale Lösung \mathbf{x} ist *kein* Extrempunkt des Lösungsraums (konvexes Polytop) von $(*)$.

Satz. Polytop $(*)$ für bipartite Matchings hat nur *ganzzahlige* Extrempunkte.

Bem. Gängige LP-Algorithmen terminieren mit Extrempunkt-Lösungen (sofern existent) und erfordern für obiges LP *kein* anschließendes LP-Runden!!

