A&W 2026

G-13 Timo Stucki, LFW E 13

Week 2

Cycles

TSP

Matchings with Exercise

Exercise Info

Infos

ETH zürich

Feel free to contact me:

- tistucki@student.ethz.ch
- Discord: timostucki

Material:

- timostucki.com

Cycles

A&D Recap

	"non-distinct" vertices	distinct vertices
-	**walk**	**path**
endpoints must be the same	**closed walk** (German: Zyklus)	**cycle** (German: Kreis)

$$O(1) \subset O(\log\log n) \subset O(\log n) \subset O(\sqrt{n}) \subset O(n) \subset O(n\log n) \subset O(n\sqrt{n}) \subset O(n^2) \subset O(2^n) \subset O(n!) \subset O(n^n)$$

Cycles

Hamiltonian Cycle

- A cycle that contains every vertex exactly once

Eulerian Cycle

- A closed walk that contains every edge exactly once

G has a Eulerian cycle $\iff$ G is connected and every vertex has an even degree

!!! IMPORTANT !!!

Satz: (Dirac 1952)

Jeder Graph $G = (V, E)$ mit $|V| \geq 3$ und Minimalgrad

$\delta(G) \geq |V|/2$ enthält einen Hamiltonkreis.

Paul Dirac
(1902-1984)

- Proof by contradiction in Script on p. 56, Satz 1.40

(different from proof in lecture)

Cycles

$$P_{S,x} := \begin{cases} 1, & \text{es gibt in G einen 1-x-Pfad, der genau die Knoten aus S enthält} \\ 0, & \text{sonst.} \end{cases}$$

There must always be a guarantee that no vertex is ever repeated

Dann gilt:

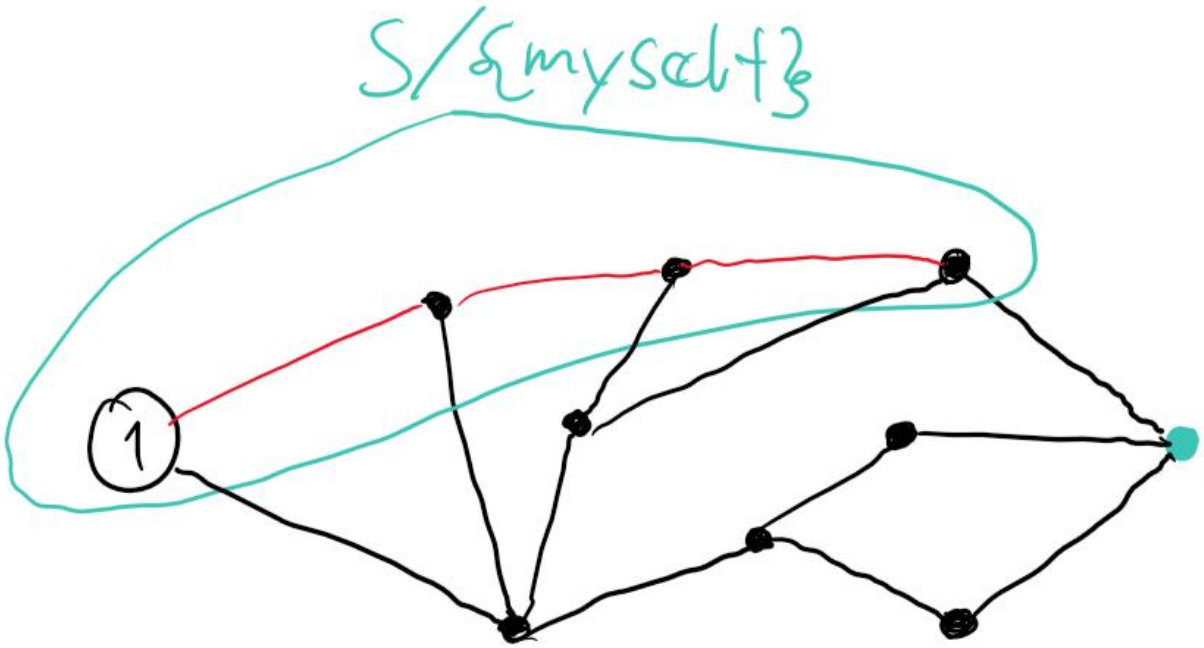
$$\text{G enthält einen Hamiltonkreis} \quad \Longleftrightarrow \quad \exists\, x \in N(1) \text{ mit } P_{[n],x} = 1$$

Cycles

HAMILTONKREIS $(G = ([n], E))$

1: *// Initialisierung*
2: **for all** $x \in [n]$, $x \neq 1$ **do**
3: $$P_{\{1,x\},x} := \begin{cases} 1, & \text{falls } \{1,x\} \in E \\ 0, & \text{sonst} \end{cases}$$
4: *// Rekursion*
5: **for all** $s = 3$ **to** n **do**
6: **for all** $S \subseteq [n]$ mit $1 \in S$ und $|S| = s$ **do**
7: **for all** $x \in S$, $x \neq 1$ **do**
8: $P_{S,x} = \max\{P_{S\setminus\{x\},x'} \mid x' \in S \cap N(x), x' \neq 1\}.$
9: *// Ausgabe*
10: **if** $\exists\, x \in N(1)$ mit $P_{[n],x} = 1$ **then**
11: **return** G enthält Hamiltonkreis
12: **else**
13: **return** G enthält keinen Hamiltonkreis

Cycles



What paths with S/{myself} are there already to my neighbours?

Cycles

HAMILTONKREIS $(G = ([n], E))$

1: *// Initialisierung*
2: **for all** $x \in [n], x \neq 1$ **do**
3: $\quad P_{\{1,x\},x} := \begin{cases} 1, & \text{falls } \{1,x\} \in E \\ 0, & \text{sonst} \end{cases}$
4: *// Rekursion*
5: **for all** $s = 3$ **to** n **do**
6: $\quad$ **for all** $S \subseteq [n]$ mit $1 \in S$ und $|S| = s$ **do**
7: $\quad\quad$ **for all** $x \in S, x \neq 1$ **do**
8: $\quad\quad\quad P_{S,x} = \max\{P_{S\setminus\{x\},x'} \mid x' \in S \cap N(x), x' \neq 1\}.$
9: *// Ausgabe*
10: **if** $\exists\, x \in N(1)$ mit $P_{[n],x} = 1$ **then**
11: $\quad$ **return** G enthält Hamiltonkreis
12: **else**
13: $\quad$ **return** G enthält keinen Hamiltonkreis

Intuition:

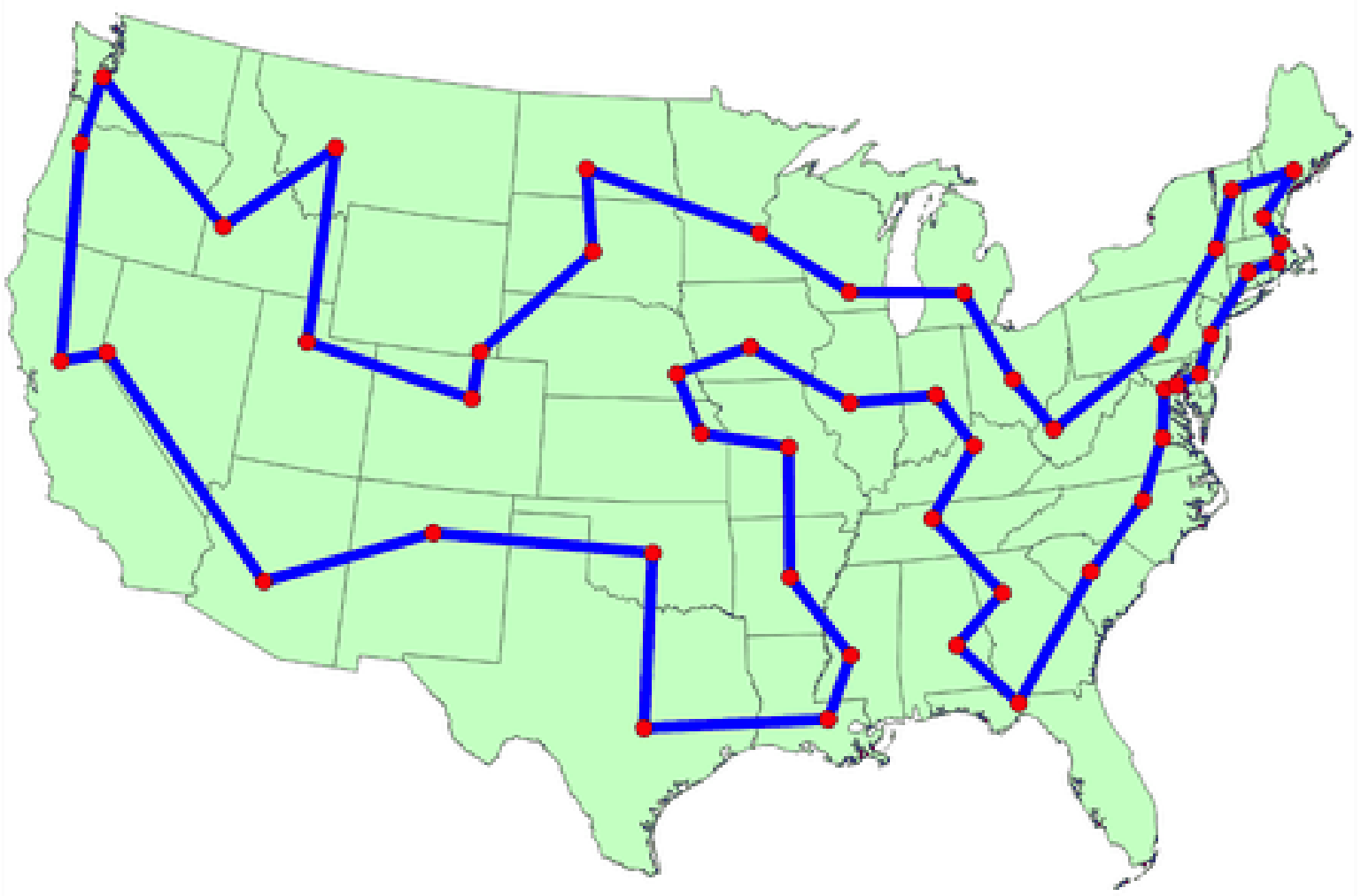
$2^n \cdot n \cdot n \} O(n^2 2^n)$

Satz 1.34. Algorithmus HAMILTONKREIS ist korrekt und benötigt Speicher $O(n \cdot 2^n)$ und Laufzeit $O(n^2 \cdot 2^n)$, wobei $n = |V|$.

TSP

Travelling-Sales-Person Problem

TSP



TSP

- Finding approximations for TSP is also NP-complete:

Das Traveling Salesman Problem (TSP) ist NP-vollständig:

Graph $G=([n], E)$ $\Longleftrightarrow$ vollst Graph $K_n=([n],\binom{[n]}{2})$

mit Längenfunktion $\ell : \binom{[n]}{2} \to \{0,1\}$

so dass $\ell(\{x,y\}) = \begin{cases} 0, & \text{falls } \{x,y\} \in E \\ 1, & \text{sonst.} \end{cases}$

Dann gilt:

G enthält Hamiltonkreis $\Longleftrightarrow$ $opt(K_n,\ell) = 0$

Hieraus folgt auch: für kein C >0 kann es einen polynominellen C-Approximationsalgorithmus geben (ausser es gilt P=NP).

TSP

Graph $G=([n], E)$ $\Longleftrightarrow$ vollst Graph $K_n=([n],\binom{[n]}{2})$

mit Längenfunktion $\ell : \binom{[n]}{2} \to \{0,1\}$

so dass $\ell(\{x,y\}) = \begin{cases} 0, & \text{falls } \{x,y\} \in E \\ 1, & \text{sonst.} \end{cases}$

Dann gilt:

G enthält Hamiltonkreis $\Longleftrightarrow$ $\text{opt}(K_n,\ell) = 0$ (1)

For C > 0 it holds that (C*0 = 0) and (C*L ≠ 0 for L ≠ 0)

→ (1) also holds for C-approx. algorithm

→ *Hieraus folgt auch:* für kein C >0 kann es einen polynominellen C-Approximationsalgorithmus geben (ausser es gilt P=NP).

Metric TSP

<u>Metrisches Travelling Salesman Problem</u>

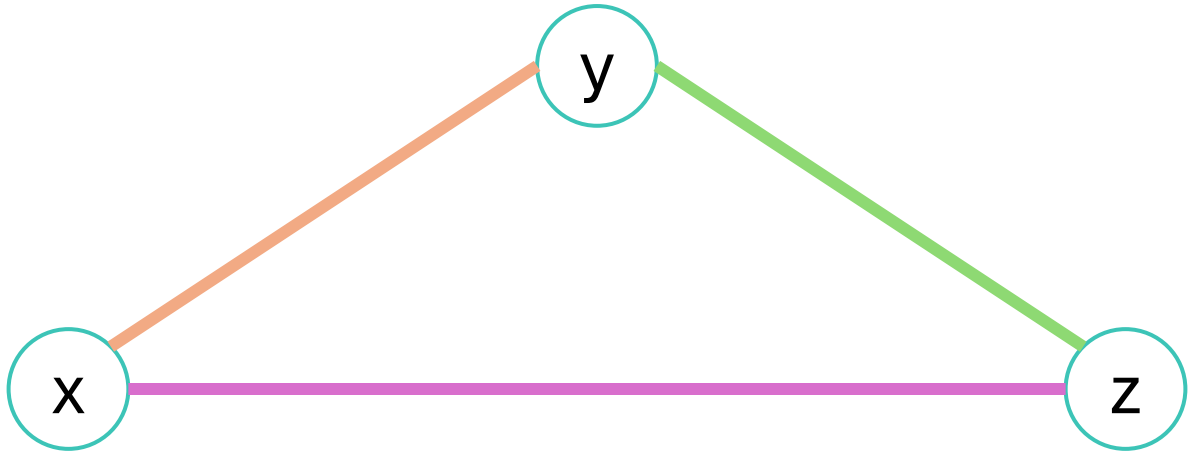
Gegeben: ein vollständiger Graph K_n und eine Funktion $\ell : \binom{[n]}{2} \to \mathbb{N}_0$ mit $\ell(\{x,z\}) \leq \ell(\{x,y\}) + \ell(\{y,z\})$ für alle $x,y,z \in [n]$

Gesucht: ein Hamiltonkreis C in K_n mit

$$\sum_{e \in C} \ell(e) = \min\{\sum_{e \in C'} \ell(e) \mid C' \text{ ist ein Hamiltonkreis in } K_n\}.$$

Metric TSP

Intuition:



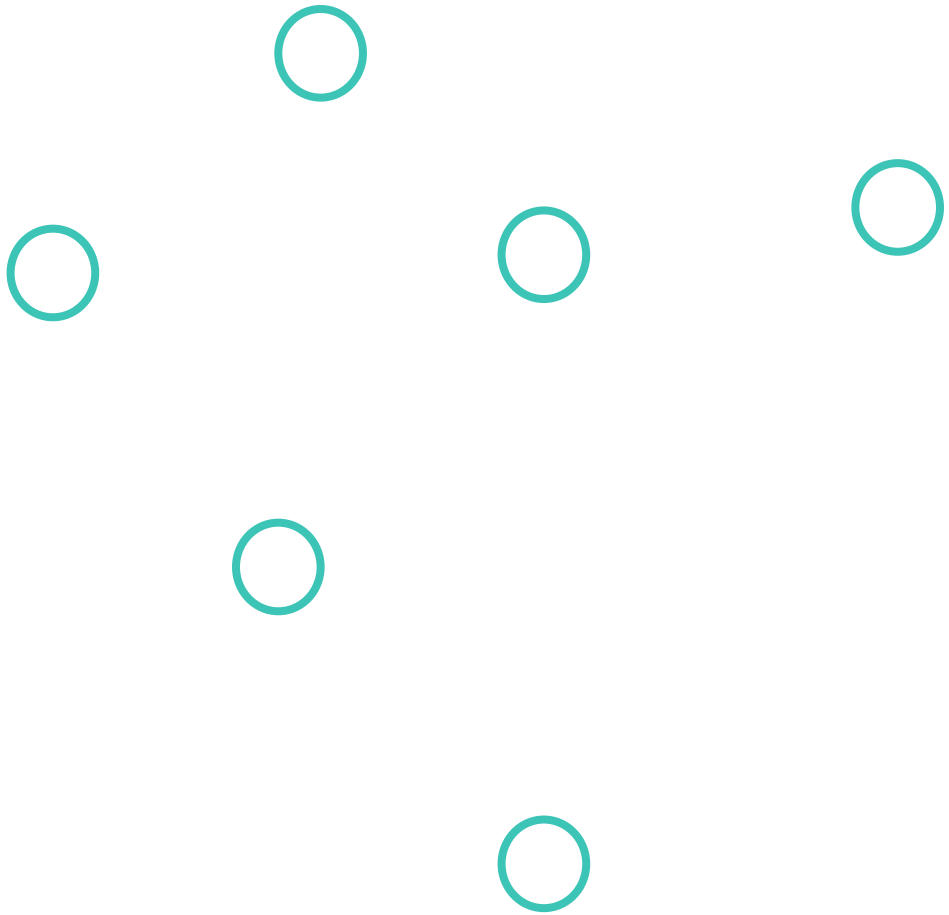
For all x, y, z in [n] with e = {x, z}, f = {x, y}, g = {y, z} the inequality holds:

$$l(\mathbf{e}) \leq l(\mathbf{f}) + l(\mathbf{g})$$

In other words: A “shortcut” can never be longer than an indirect path

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



Goal:

- Find a 2-approx. algorithm for the metric TSP

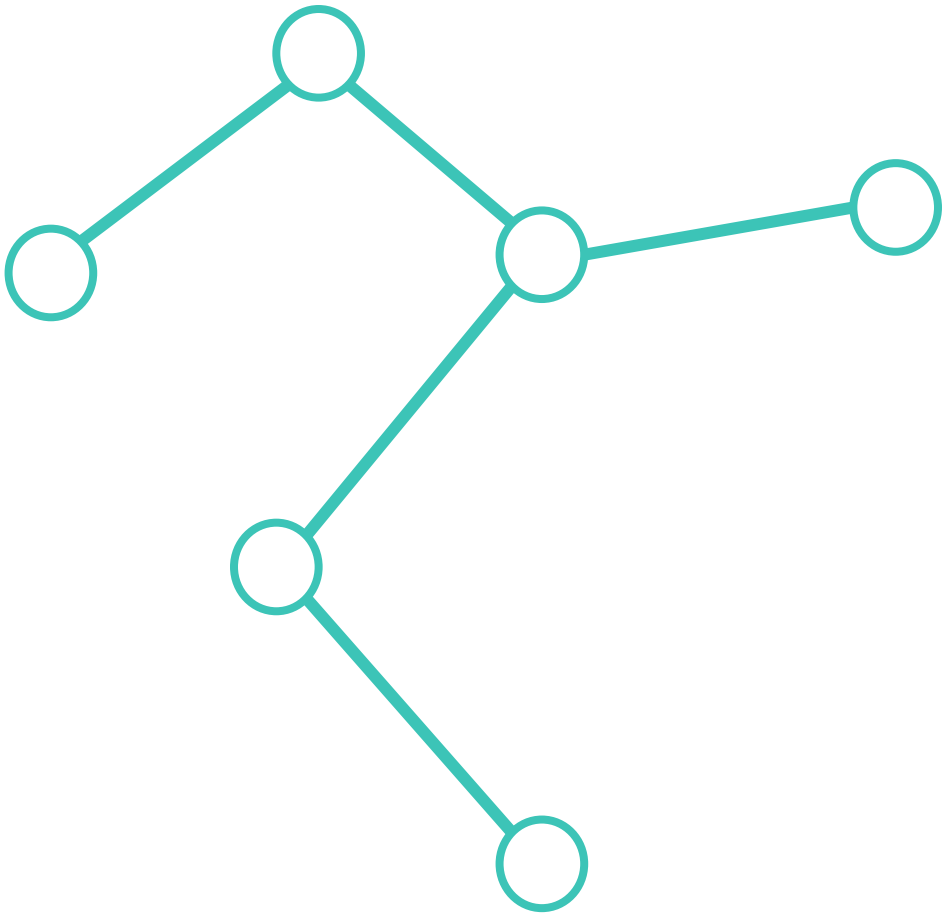
This means:

Algorithm has to find Hamiltonian cycle C s.t.

$$l(C) \leq 2 * opt(K_n, l)$$

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$

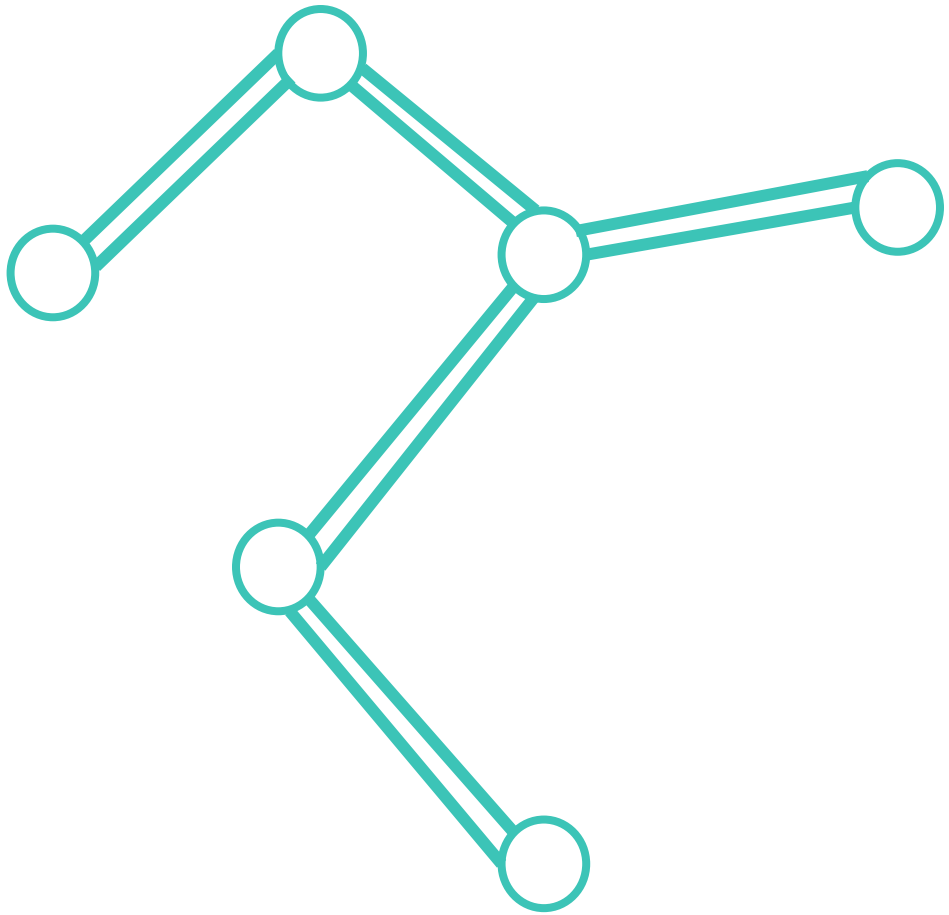


1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

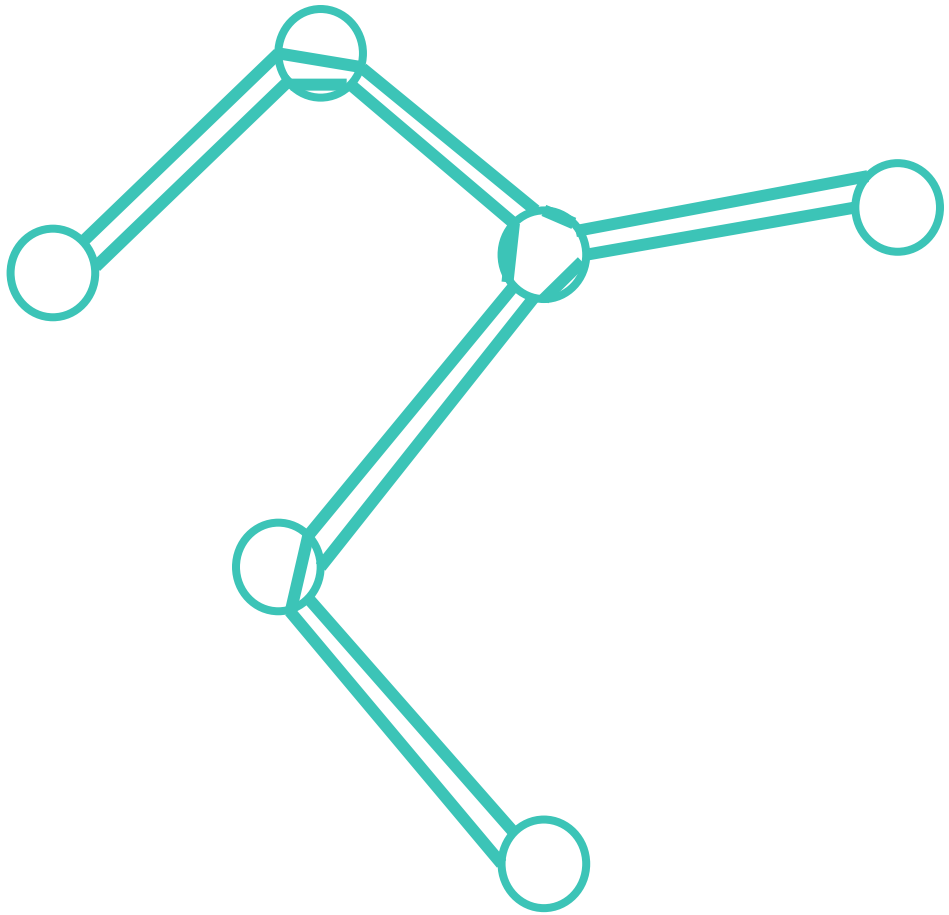
$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

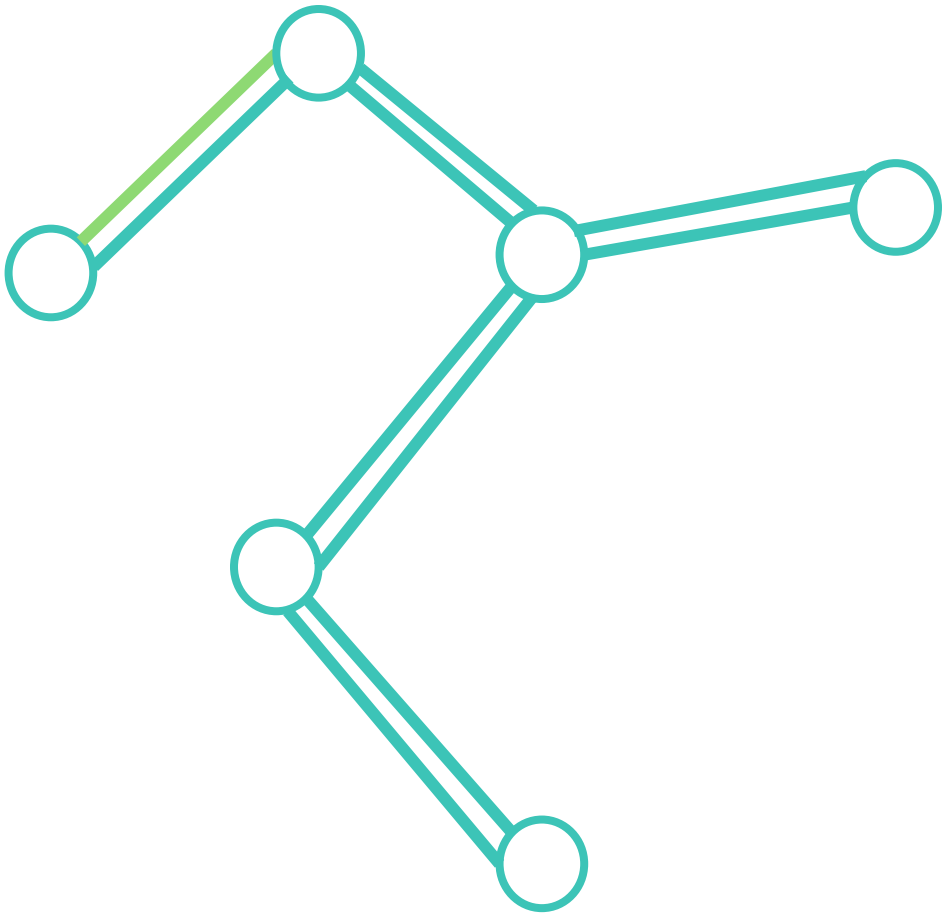
$$2 * l(T) \leq 2 * opt(K_n, l)$$

3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

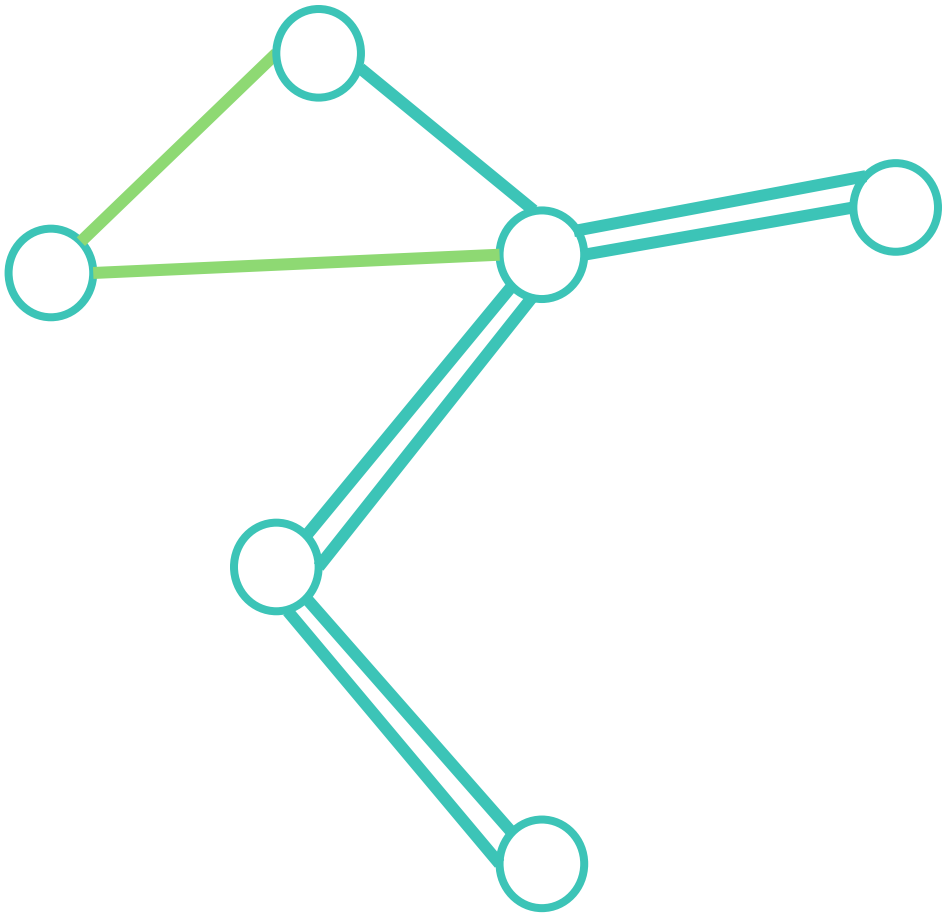
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

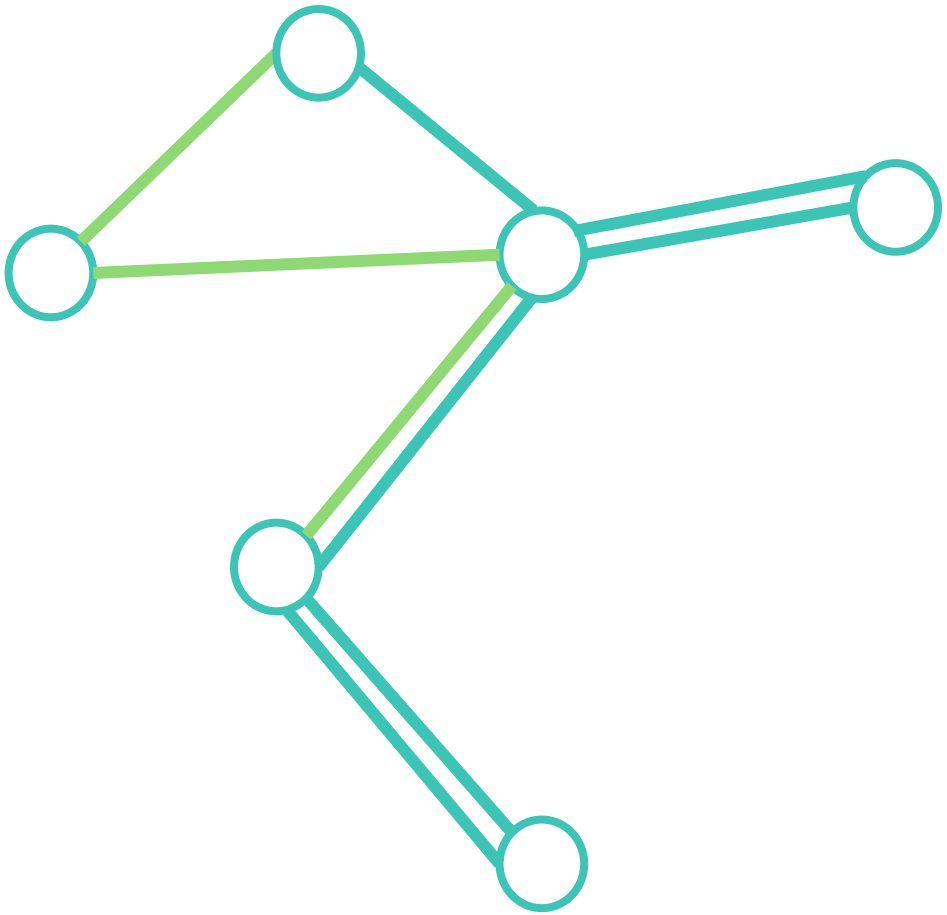
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

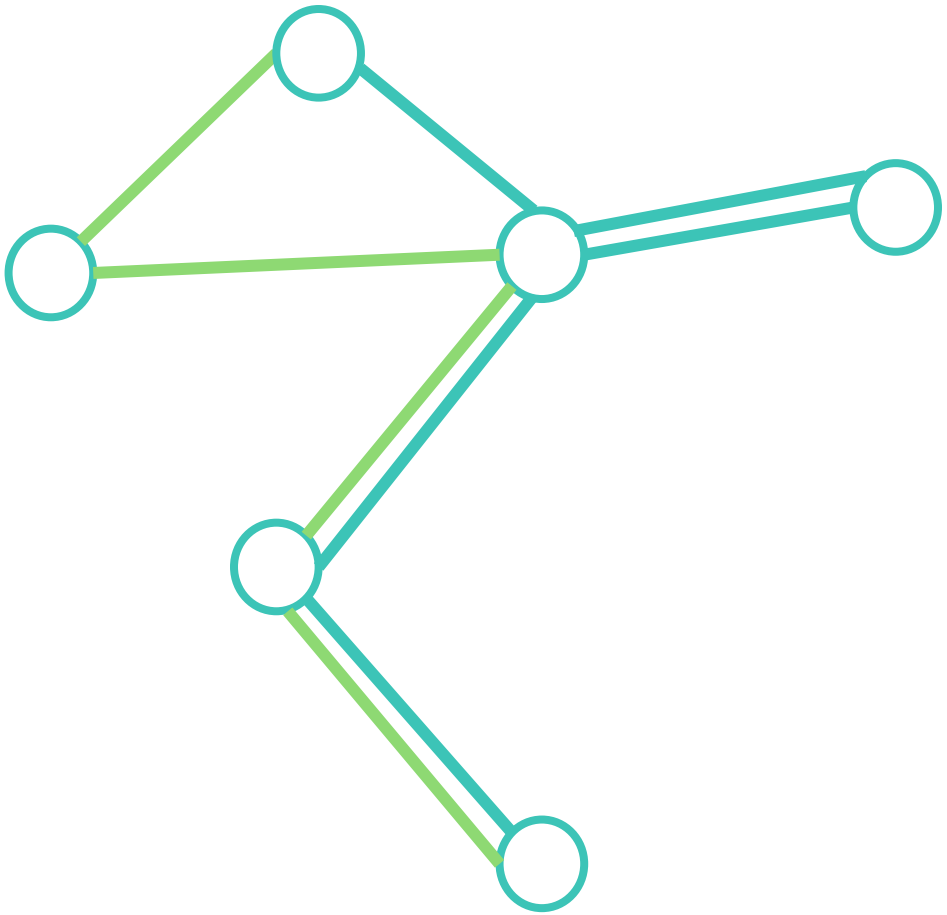
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

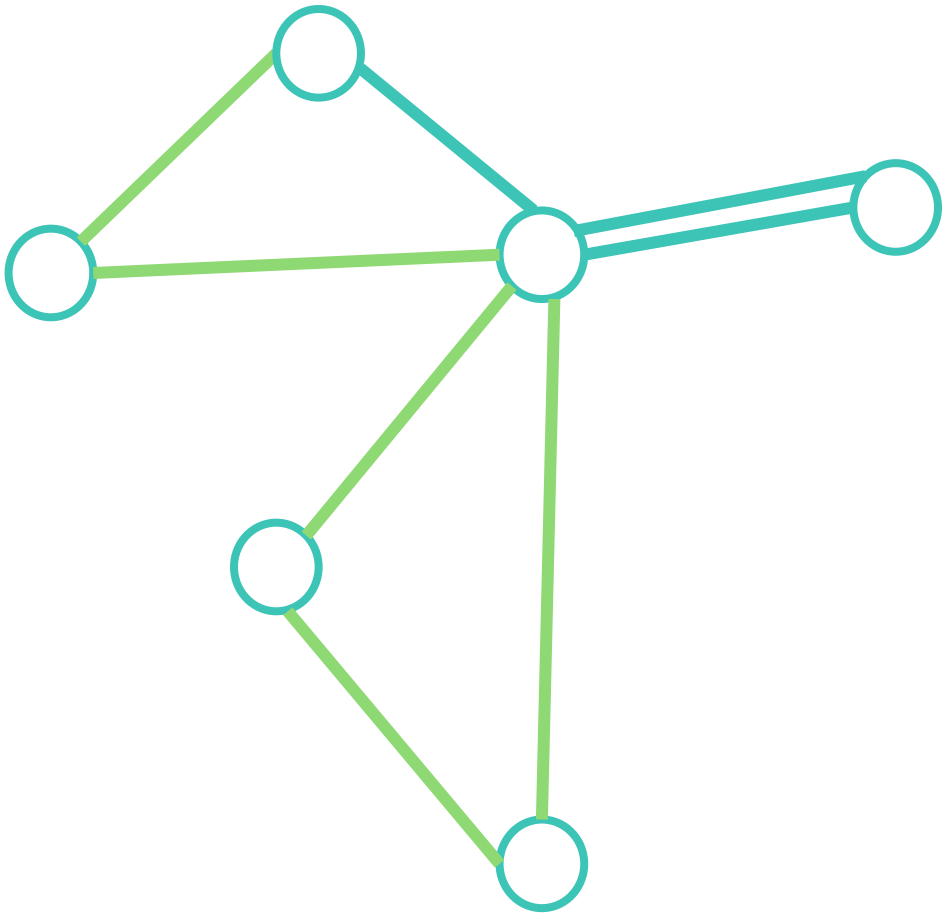
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

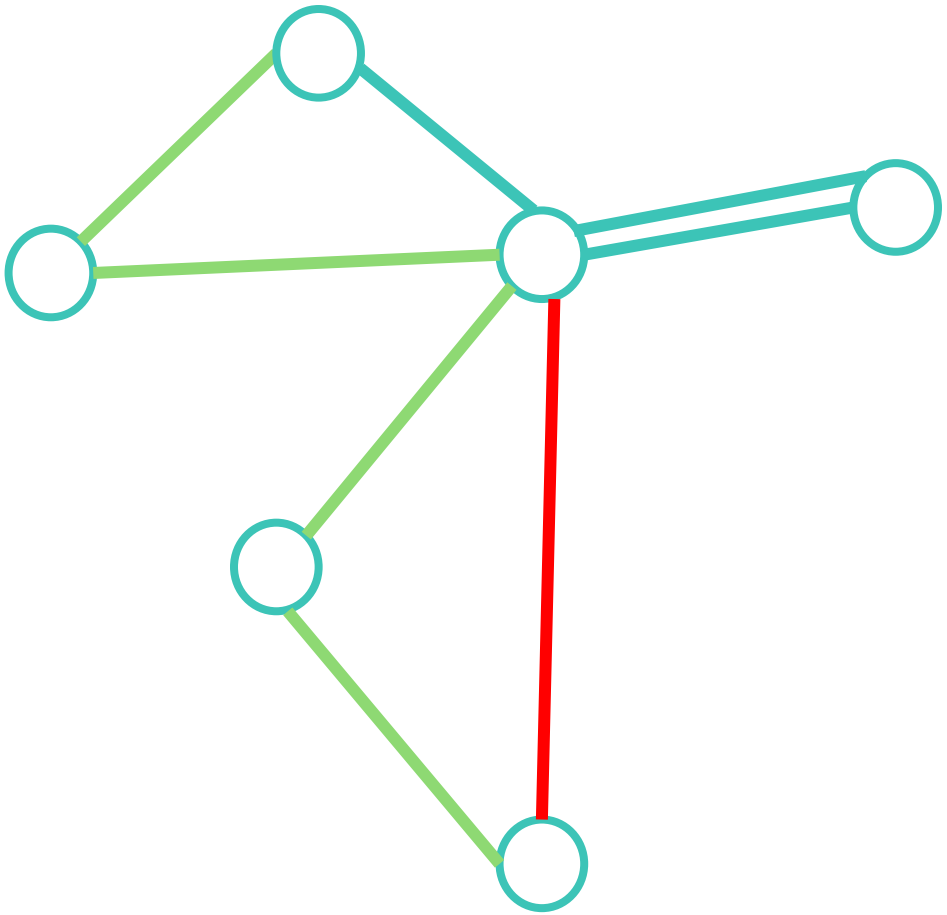
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

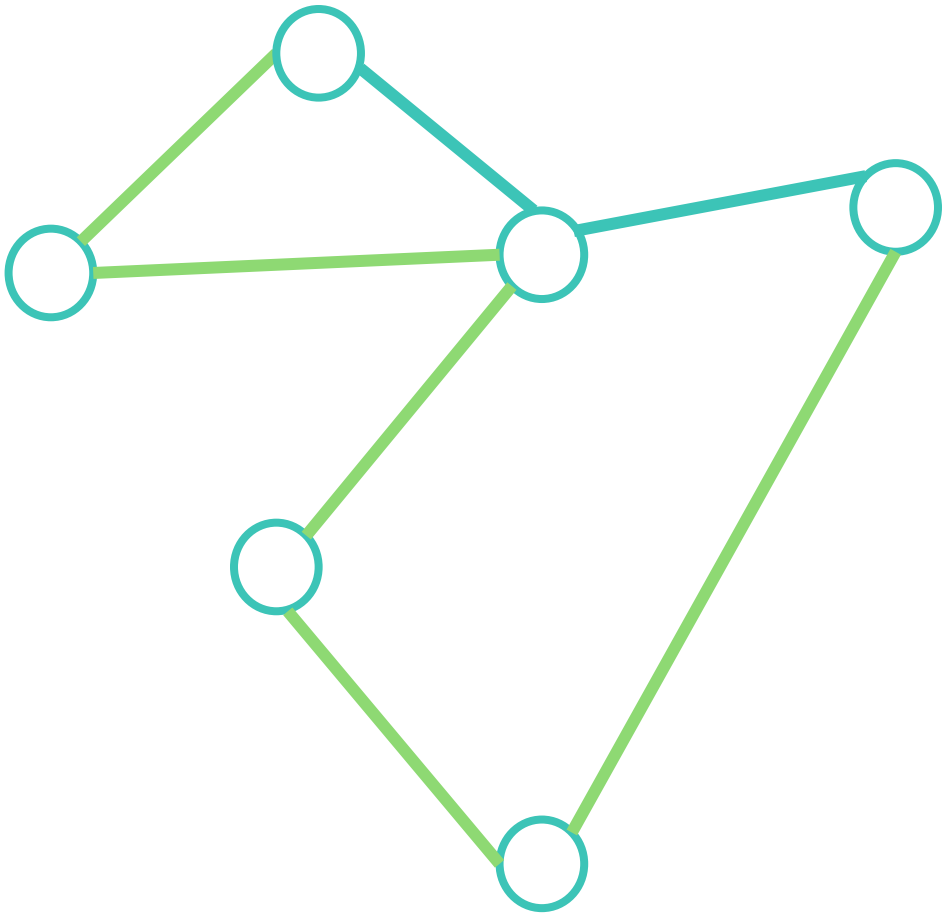
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

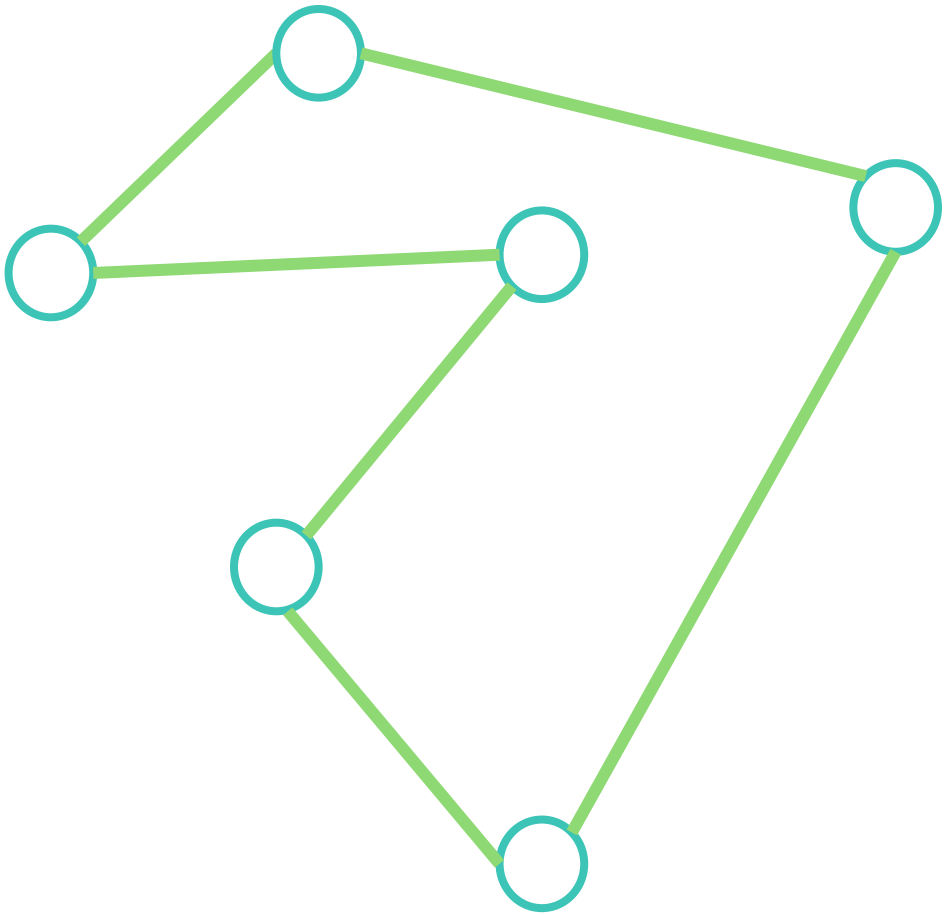
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

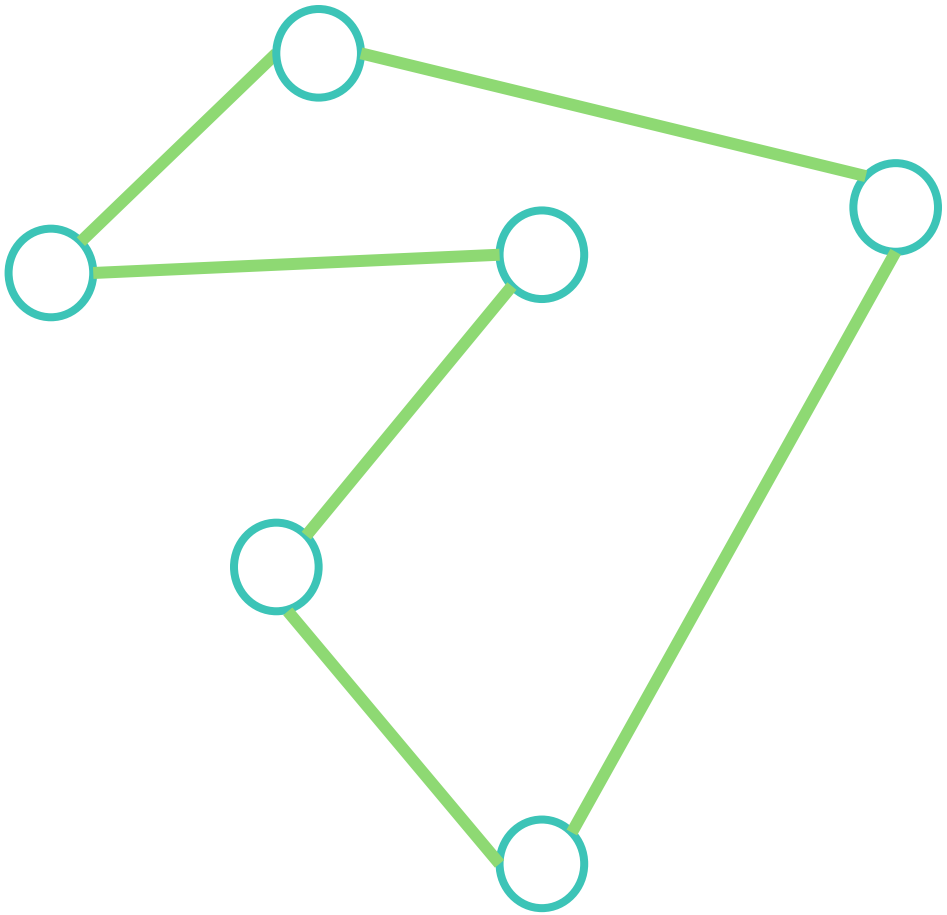
3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

3. Find Eulerian cycle W

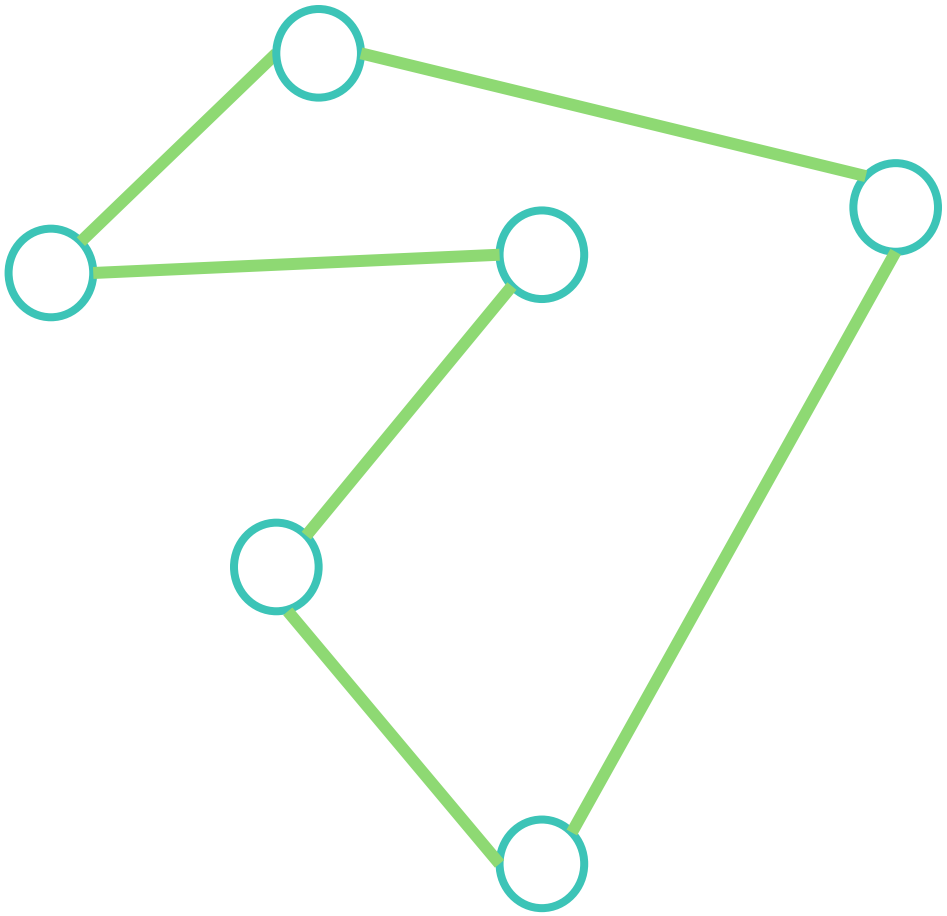
$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

$$l(C) \leq l(W) \leq 2 * opt(K_n, l)$$

Metric TSP

For all x, y, z in [n] inequality holds: $l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\})$



1. Find MST T (Minimal Spanning Tree)

$$l(T) \leq opt(K_n, l)$$

2. Double all edges in T

$$2 * l(T) \leq 2 * opt(K_n, l)$$

3. Find Eulerian cycle W

$$l(W) = 2 * l(T) \leq 2 * opt(K_n, l)$$

4. Take shortcuts in W s.t. every node is visited only once → Hamiltonian cycle C

$$l(C) \leq l(W) \leq 2 * opt(K_n, l)$$

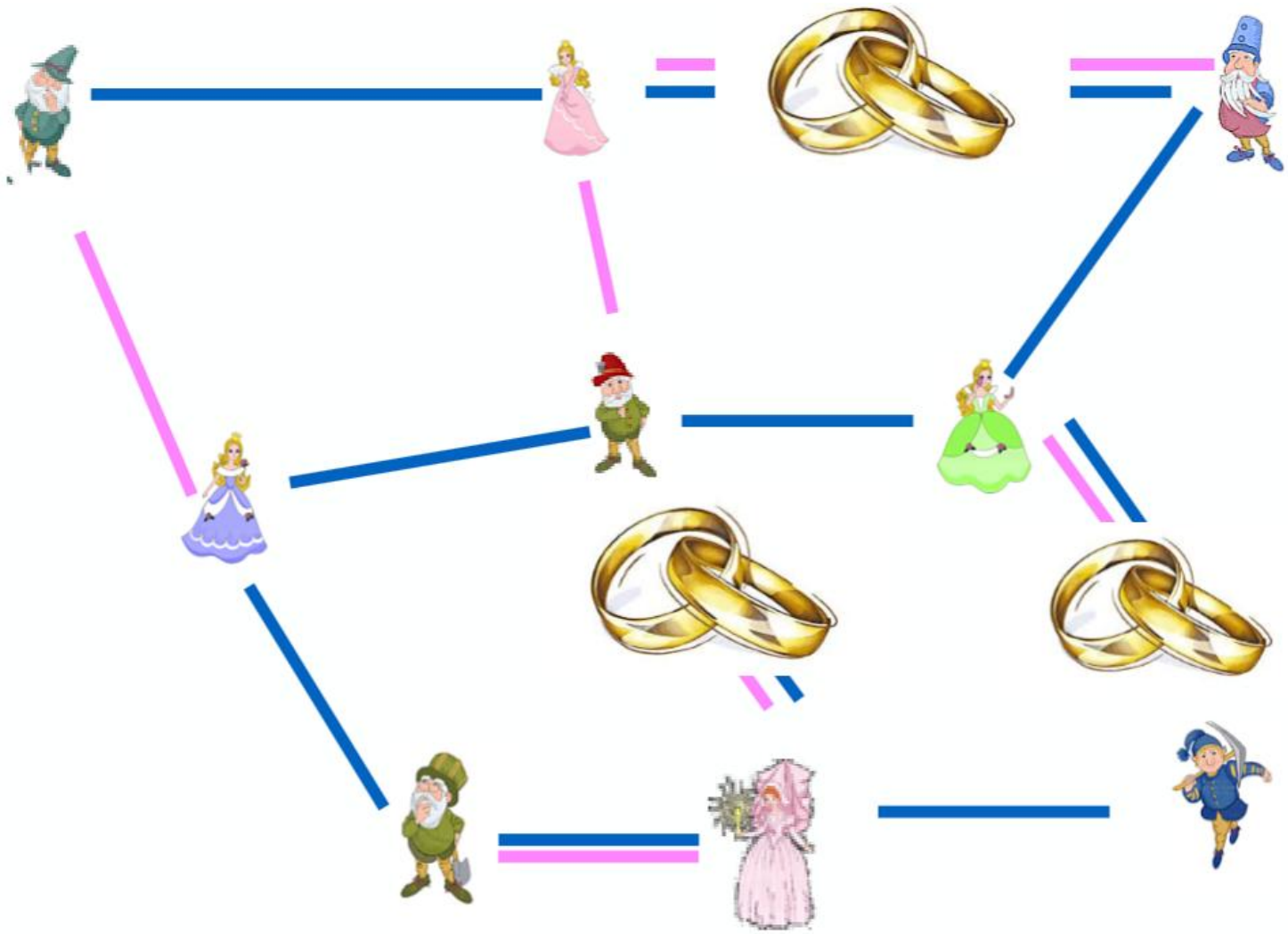
Metric TSP

Satz 1.43. Für das METRISCHE TRAVELLING SALESMAN PROBLEM gibt es einen 2-Approximationsalgorithmus mit Laufzeit $O(n^2)$.

- Finding MST dominates the runtime

Matchings

Matchings

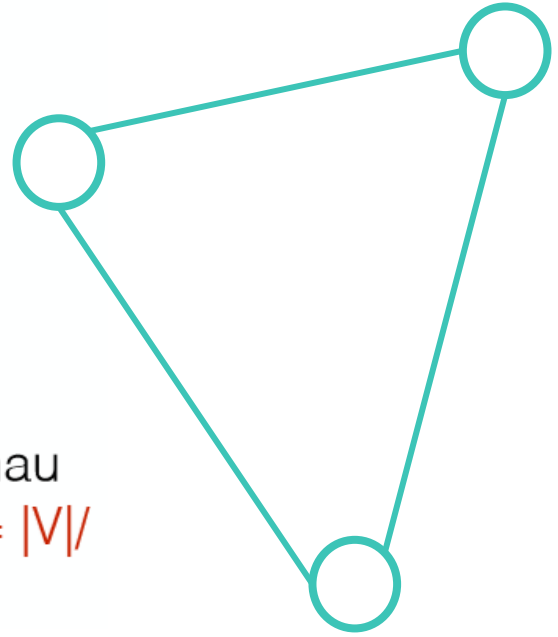


Matchings

Eine Kantenmenge $M \subseteq E$ heisst **Matching** in einem Graphen $G = (V, E)$, falls kein Knoten des Graphen zu mehr als einer Kante aus M inzident ist.

Ein Knoten v wird von M **überdeckt**, falls es eine Kante $e \in M$ gibt, die v enthält.

Ein Matching M heisst **perfektes Matching**, wenn jeder Knoten durch genau eine Kante aus M überdeckt wird, oder, anders ausgedrückt, wenn $|M| = |V|/2$.

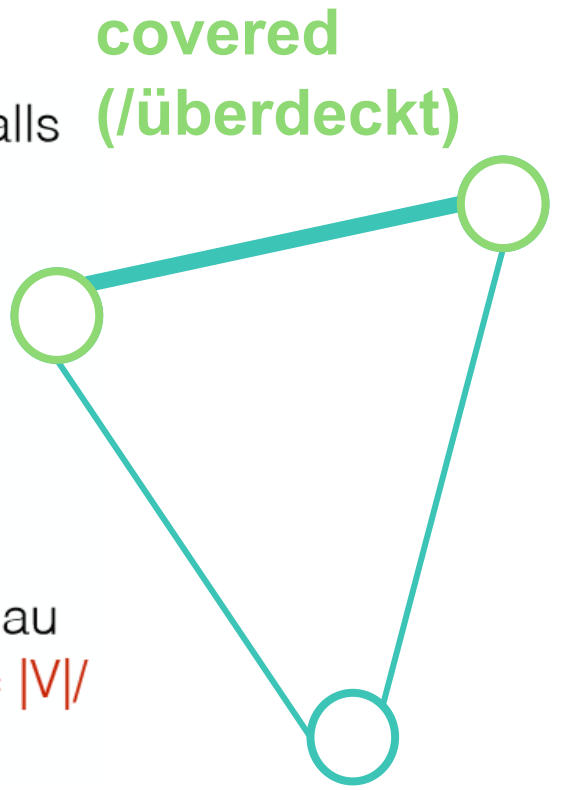


Matchings

Eine Kantenmenge $M \subseteq E$ heisst **Matching** in einem Graphen $G = (V, E)$, falls kein Knoten des Graphen zu mehr als einer Kante aus M inzident ist.

Ein Knoten v wird von M **überdeckt**, falls es eine Kante $e \in M$ gibt, die v enthält.

Ein Matching M heisst **perfektes Matching**, wenn jeder Knoten durch genau eine Kante aus M überdeckt wird, oder, anders ausgedrückt, wenn $|M| = |V|/2$.



Matchings

Ein Matching $M \subseteq E$ ist ein **inklusionsmaximales Matching**, wenn es kein Matching M' gibt mit $M \subseteq M'$ und $|M'| > |M|$.

Ein Matching $M \subseteq E$ ist ein **(kardinalitäts-)maximales Matching**, wenn es kein Matching M' gibt mit $|M'| > |M|$.

(In der englischsprachigen Literatur hat sich die folgende abkürzende Schreibweise eingebürgert: Ein *maximal matching* bezeichnet ein inklusionsmaximales Matching, während ein *maximum matching* ein kardinalitätsmaximales Matching ist.)

inklusionsmaximales Matching
(aber nicht kardinalitätsmaximal)

kardinalitätsmaximales Matching
(auch inklusionsmaximal!)

Matchings

Maxim**al** matching:

We cannot add another edge to the matching

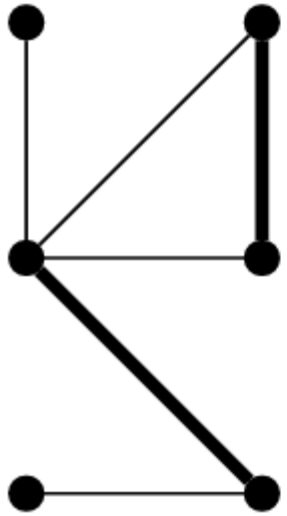
Maxim**um** matching:

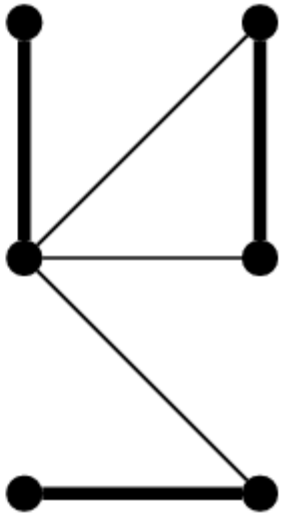
We couldn’t find a better matching for a given graph

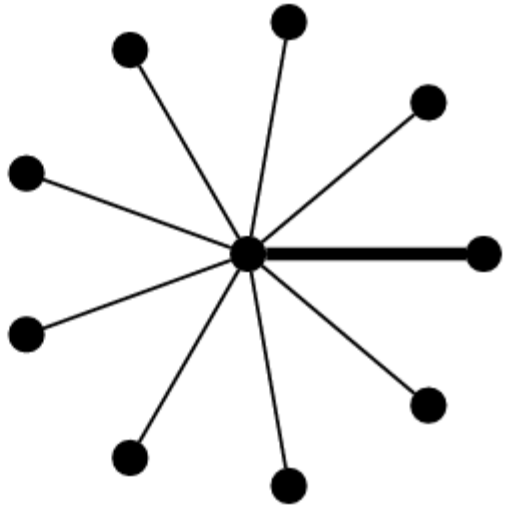
inklusionsmaximales Matching
(aber nicht kardinalitätsmaximal)

kardinalitätsmaximales Matching
(auch inklusionsmaximal!)

Matchings







Matchings

$$|M_{inc}| \leq |M_{max}| \quad (1)$$

$$|M_{inc}| \geq \frac{|M_{max}|}{2} \quad (2)$$

Every edge in M_{max} must be adjacent to at least one edge in M_{inc} otherwise, that edge could be added to M_{inc}, and M_{inc} wouldn't really be maximal

Additionally, every vertex covered by M_{inc} (of which there are $2|M_{inc}|$) must only be incident to one edge in M_{max} since it is a matching

→ $|M_{max}| \leq 2|M_{inc}|$ → (2)

Matchings

GREEDY-MATCHING (G)

1: $M \leftarrow \emptyset$
2: **while** $E \neq \emptyset$ **do**
3: wähle eine beliebige Kante $e \in E$
4: $M \leftarrow M \cup \{e\}$
5: lösche e und alle inzidenten Kanten in G

Satz 1.47. Der Algorithmus GREEDY-MATCHING bestimmt in Zeit $O(|E|)$ ein inklusionsmaximales Matching M_{Greedy} für das gilt:

$$|M_{Greedy}| \geq \frac{1}{2}|M_{max}|,$$

wobei M_{max} ein kardinalitätsmaximales Matching sei.

M_{Greedy} is maximal!

Umfrage

A&W G-13

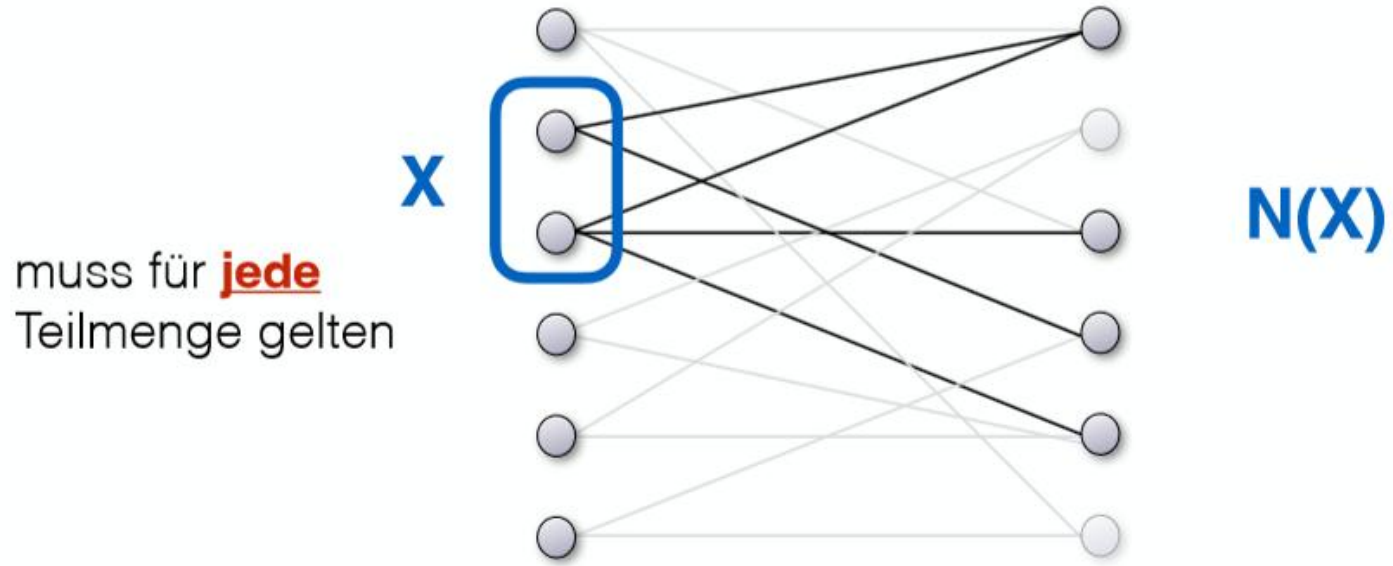
Matchings

Satz: (Hall, Heiratssatz)

Ein bipartiter graph $G=(A \dot\cup B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw**

$\forall X \subseteq A : |X| \leq |N(X)|$

Philip Hall
(1904-1982)



Matchings

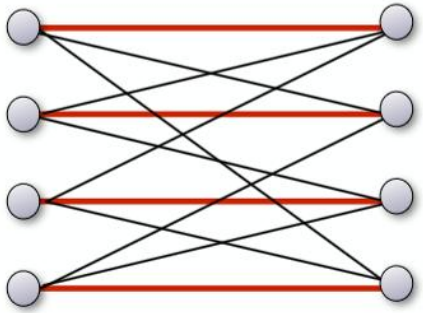
Theorem: (Hall, 1935)
Ein bipartiter graph $G=(A \dot\cup B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw** $\forall A' \subseteq A : |A'| \leq |N(A')|$


Philip Hall
(1904-1982)

Korollar: (Frobenius, 1917)
Für alle k gilt: jeder k-reguläre bipartite graph enthält ein perfektes Matching.

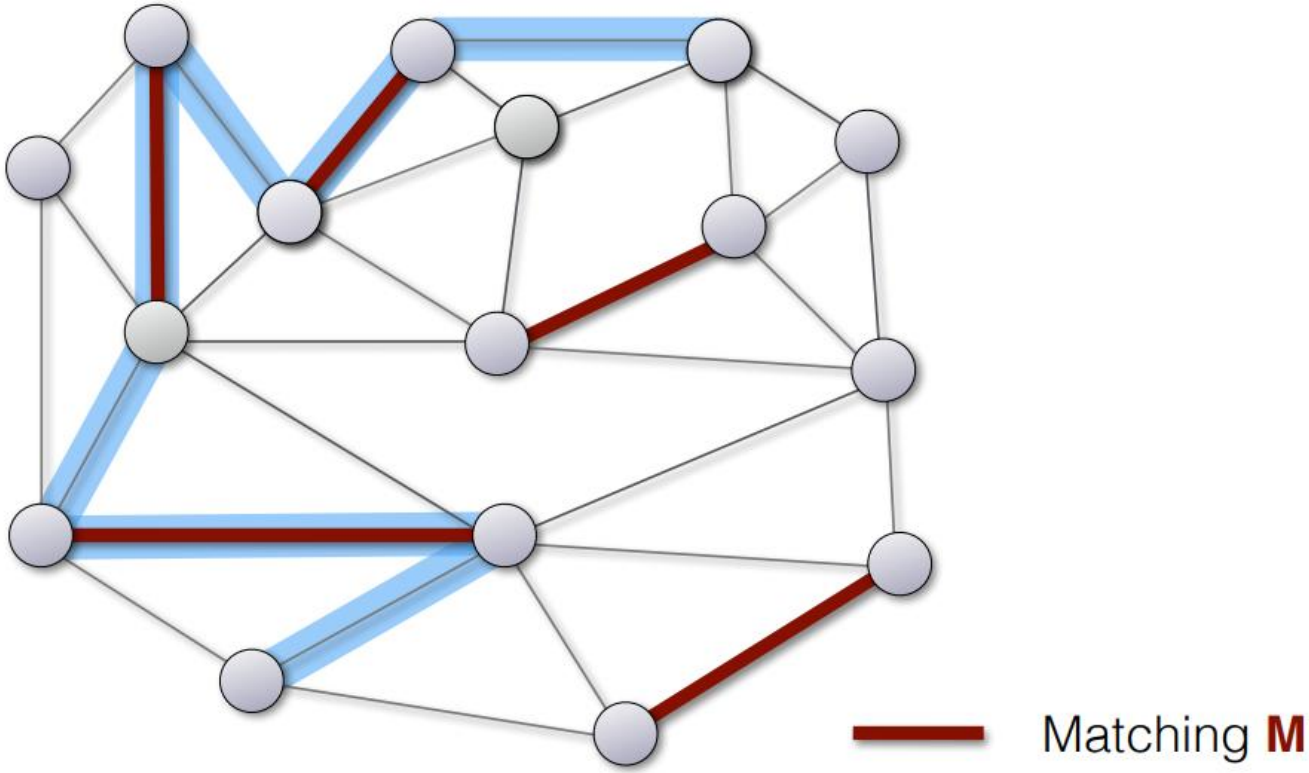
Es gilt sogar: Graph ist Vereinigung von perfekten Matchings.


Ferdinand Georg Frobenius
(1849 – 1917)



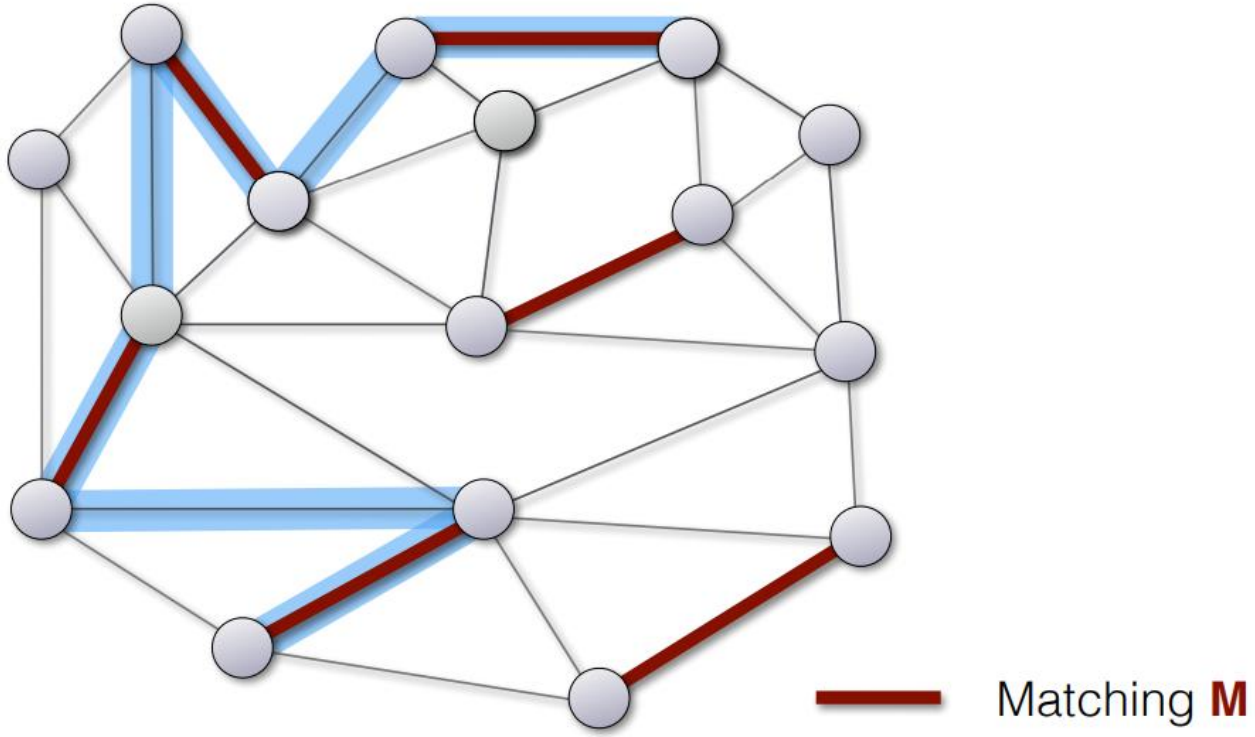
Ein Graph G heisst k-*regulär* (engl. k-*regular*), falls für alle Knoten $v \in V$ gilt, dass $\deg(v) = k$.

Matchings



Ein **M-augmentierender Pfad P** ist ein Pfad, der abwechselnd Kanten aus **M** und nicht aus **M** enthält und der in von **M** nicht überdeckten Knoten beginnt und endet.

Matchings



Ein **M-augmentierender Pfad P** ist ein Pfad, der abwechselnd Kanten aus **M** und nicht aus **M** enthält und der in von **M** nicht überdeckten Knoten beginnt und endet.

⇒ durch *Tauschen* entlang M können wir das Matching vergrössern:

M' := **M** ⊕ **P**

Matchings

Satz 1.48 (Satz von Berge). Ist M ein Matching in einem Graphen $G = (V, E)$, das nicht kardinalitätsmaximal ist, so existiert ein augmentierender Pfad zu M.

→ We can make an algorithm out of this!

Matchings

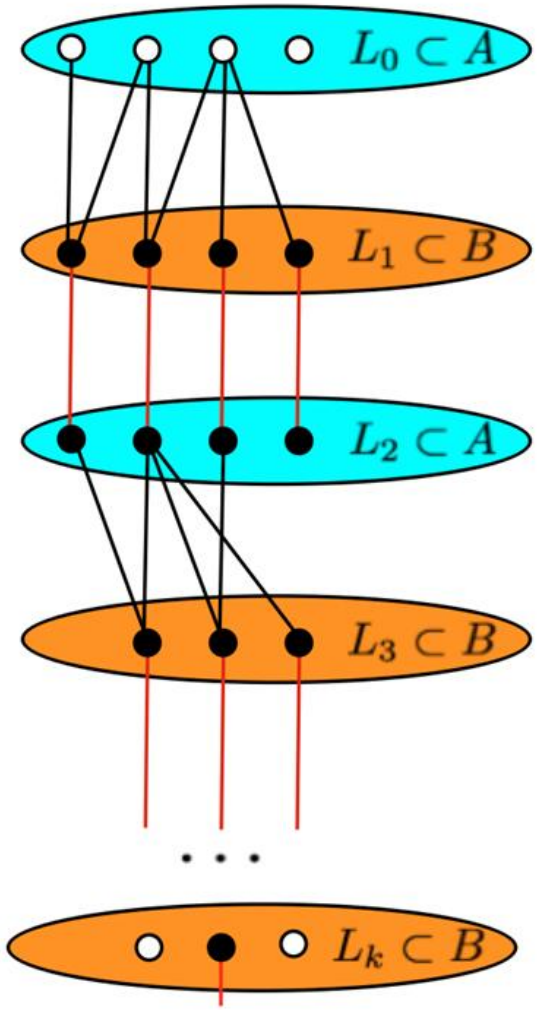
AUGMENTING_PATH $(G = (A \uplus B, E), M)$

1: $L_0 := \{\text{unüberdeckte Knoten in } A\}$
2: Markiere alle Knoten aus L_0 als besucht.
3: **if** $L_0 = \emptyset$ **then**
4: **return** M ist maximal
5: **for all** $i = 1$ **to** n **do**
6: **if** i ungerade **then**
7: $L_i := \{\text{unbesuchte Nachbarn von } L_{i-1} \text{ via Kanten in } E \setminus M\}$
8: **else**
9: $L_i := \{\text{unbesuchte Nachbarn von } L_{i-1} \text{ via Kanten in } M\}$
10: Markiere alle Knoten aus L_i als besucht.
11: **if** L_i enthält unüberdeckten Knoten v **then**
12: Finde Pfad P von L_0 nach v durch backtracking
13: **return** P // *terminiert Algorithmus*
14: **return** M ist schon maximal

Matchings

AUGMENTING_PATH $(G = (A \uplus B, E), M)$

1: $L_0 := \{\text{unüberdeckte Knoten in } A\}$
2: Markiere alle Knoten aus L_0 als besucht.
3: **if** $L_0 = \emptyset$ **then**
4: **return** M ist maximal
5: **for all** $i = 1$ **to** n **do**
6: **if** i ungerade **then**
7: $L_i := \{\text{unbesuchte Nachbarn von } L_{i-1} \text{ via Kanten in } E \setminus M\}$
8: **else**
9: $L_i := \{\text{unbesuchte Nachbarn von } L_{i-1} \text{ via Kanten in } M\}$
10: Markiere alle Knoten aus L_i als besucht.
11: **if** L_i enthält unüberdeckten Knoten v **then**
12: Finde Pfad P von L_0 nach v durch backtracking
13: **return** P // *terminiert Algorithmus*
14: **return** M ist schon maximal



Matchings

- For 1 iteration: $O(|E|)$
- In total after at most n/2 iterations: $O(|V||E|)$

(since a maximum matching has at most n/2 edges)

Matchings

- Preview:

Satz 1.49. Der Algorithmus von Hopcroft und Karp durchläuft die while-Schleife nur $O(\sqrt{|V|})$ Mal. Er berechnet daher ein maximales Matching in einem bipartiten Graphen in Zeit $O(\sqrt{|V|} \cdot (|V| + |E|))$.

Exercise S3.1

Exercise S3.1 – *Bipartite Matching*

Let $G = (A \dot{\cup} B, E)$ be a bipartite graph.

(a) Prove or disprove: If G has a Hamiltonian cycle, then G contains two disjoint perfect matchings.

(b) Prove or disprove: If G contains two disjoint perfect matchings, then G has a Hamiltonian cycle.

(c) Let $A' \subseteq A$ and $B' \subseteq B$. Assume that there is a matching M_A covering A' and a matching M_B covering B' (these two matchings do not have to be disjoint!). Show that there is a matching M that covers both A' and B' (i.e. it covers $A' \cup B'$). (*Hint:* Which properties has the graph $(V, M_A \cup M_B)$? Try to build a matching using only edges in $M_A \cup M_B$.)

Exercise Preview

Exercise P1.1 – *Matchings and magic tricks*

(a) We say that a bipartite graph $G = (A \cup B, E)$ is (k, ℓ)-regular if every vertex in A has degree exactly k, and every vertex in B has degree exactly ℓ. Show that when $k \geq \ell$, any (k, ℓ)-regular graph G has a matching of size exactly $|A|$.

(b) A magician together with their assistant performs the following trick. An audience member picks 5 arbitrary cards from a standard 52-card playing deck, and gives those cards to the magician's assistant. The assistant then picks four out of the five cards, and shows those cards to the magician, in order of the assistant's choice. The magician now, with certainty, announces what is the remaining fifth card in the assistant's hand, not seen by the magician at any point.

How could the magician and the assistant (who both have unlimited brain-memory) have chosen a strategy ahead of time so that they would succeed in this trick, regardless of which five cards are selected by the audience member? Once the show has started, magician and the assistant cannot communicate in any way outside of the scope of the trick (i.e. the only communication from the assistant to the magician is the sequence of the 4 cards the assistant presents to them in the order of their choice). *Hint:* Use subproblem (a).

Exercise Review

The following exercise is worth 2 bonus points.

Exercise T1.1 – *Disjoint Paths*

Let $G = (V, E)$ be an undirected graph. For two vertices $s, t \in V$ we denote by $\lambda_{s,t}$ the maximal number of edge-disjoint path between s and t in G.

(a) Show that for three pairwise different vertices $a, b, c \in V$ we have $\lambda_{a,c} \geq \min\{\lambda_{a,b}, \lambda_{b,c}\}$.

(b) Let $x, y, z \in V$ be three pairwise different vertices and $\alpha, \beta \in \mathbb{N}$ such that $\alpha \leq \lambda_{x,y}$, $\beta \leq \lambda_{x,z}$, and $\alpha + \beta \leq \max\{\lambda_{x,y}, \lambda_{x,z}\}$. Show that there are α many x-y paths and β many x-z-path such that all $\alpha + \beta$ paths are pairwise edge-disjoint.
(Hint: try to construct a graph that contains a vertex v such that the task reduces to finding $\alpha + \beta$ edge-disjoint x-v paths.)

A general approach

- Understand the property P you need to prove
- Try to map the problem onto a different one (on a different graph G') s.t.

  P' holds for G' → P holds for G (1)

- Construct the new graph for which a modified property P' is easier to prove
- Prove P'
- Make use of (1)

The following exercise does not give bonus points.

Exercise T1.2 – *Connectivity*

(a) Prove or give a counterexample to the following claim: If G is a connected graph such that every vertex of G has even degree, then G is 2-edge-connected.

(b) Prove or give a counterexample to the following claim: If G is a connected graph such that every vertex has degree at least 4, then G is 2-edge-connected.

(c) Prove or give a counterexample to the following claim: For a graph $G = (V, E)$, we define a relation on *vertices* of G: vertices $a \in V$ and $b \in V$ satisfy $a \sim b$ if there is a cycle in G containing both a and b. Then for every graph G relation $\sim$ is an equivalence relation.