# Algorithms and Probability

## Week 2

Georg Hasebe — 29/02/2024

# Kreise

- Sei $G = (V, E)$ ein Graph.

- **Hamiltonkreis:**
  - Ein Kreis in G, der jeden Knoten genau einmal enthält.
- **Eulertour:**
  - Ein geschlossener Weg in G, der jede Kante genau einmal enthält.

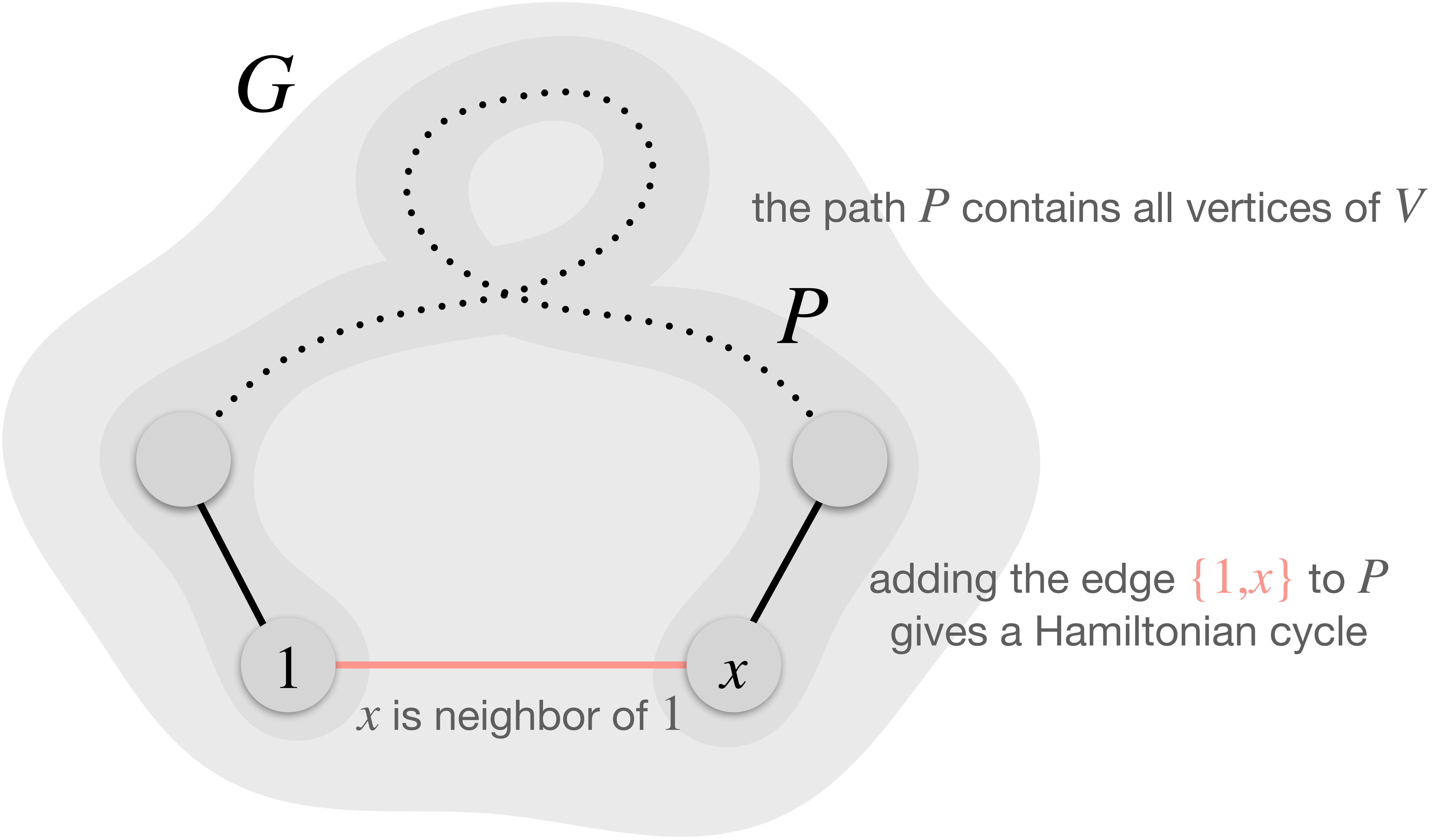
# On finding Hamiltonian cycles

- Finding Hamiltonian cycles is **hard** (NP-hard)
- The only known algorithms are **exponential**
- **Naive:** try out all possibilities for a Hamiltonian cycle.
  - **How many?** At most $(n-1)!/2$.

# DP Algorithm

Let $G = (V, E)$ be a graph and let $V = [n] = \{1,2,\ldots,n\}$.

A path starting at $1$ and ending at $x$ (i.e. a $1$-$x$ path) containing all vertices of $V$, where $x$ is a neighbor of $1$ (i.e. $x \in N(1)$), can be turned into a Hamiltonian cycle.



# DP Algorithm

$G = (V, E)$, $V = [n] = \{1,2,\ldots,n\}$.

Let $S \subseteq V$ where $1 \in S$. Consider the following notation for all $x \in S, x \neq 1$

$$P_{S,x} = \begin{cases} 1, & \text{there exists a } 1\text{-}x \text{ path in } G \text{ that contains all vertices in } S \\ 0, & \text{otherwise.} \end{cases}$$

Now if there exists some $x \in N(1)$ where $P_{[n],x} = 1$, $G$ contains a Hamiltonian cycle.

We can calculate the values for $P_{S,x}$ using **dynamic programming**.

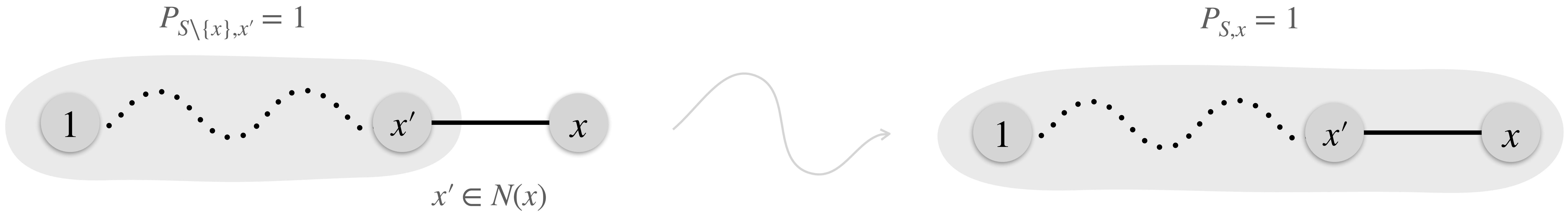
# DP Algorithm

$G = (V, E)$, $V = [n] = \{1,2,\ldots,n\}$.

**Base cases:** If $S = \{1,x\}$ for some $x \in V$, then $P_{S,x} = 1$ if $\{1,x\} \in E$.

**Recursion:**

$$P_{S,x} = \max\{P_{S\setminus\{x\},x'} \mid x' \in S \cap N(x), x' \neq 1\}$$



# Pseudocode

**Hamiltonkreis** $(G = ([n], E))$

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

The initialization part covers all subsets of size 2. We therefore start with subsets of size 3 and work our way up to $n$.

We go through all subsets of size $s$. If $s = n$, then the only subset will be $[n]$ itself. Remember, we try to determine $P_{[n],x}$.

We demand that $x \neq 1$, because we started with 1 already.

$S \setminus \{x\}$ ensures that any path that we extend by $x$ does not contain $x$.

Here we attempt to “close” a Hamiltonian $1$-$x$ path in order to get a Hamiltonian cycle.

# Result

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

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

# Proof

```
HAMILTONKREIS (G = ([n], E))
// Initialisierung
for all x ∈ [n], x ≠ 1 do
    P_{1,x},x := { 1, falls {1, x} ∈ E
                 { 0, sonst
// Rekursion
for all s = 3 to n do
    for all S ⊆ [n] mit 1 ∈ S und |S| = s do
        for all x ∈ S, x ≠ 1 do
            P_S,x = max{P_S\{x},x' | x' ∈ S ∩ N(x), x' ≠ 1}.
// Ausgabe
if ∃ x ∈ N(1) mit P_[n],x = 1 then
    return G enthält Hamiltonkreis
else
    return G enthält keinen Hamiltonkreis
```

$$\sum_{s=3}^{n} \sum_{S\subseteq[n],1\in S,|S|=s} \sum_{x\in S,x\neq 1} O(???)$$

$$= \sum_{s=3}^{n} \sum_{S\subseteq[n],1\in S,|S|=s} \sum_{x\in S,x\neq 1} O(n)$$

$$= \sum_{s=3}^{n} \binom{n-1}{s-1} (s-1) O(n)$$

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

$|S| = s$, but we exclude $1$.

A subset of $S'$ of size $s-1$ from $n-1$ vertices; then $S = S' \cup \{1\}$.

(*) where we used $\sum_{s=0}^{n-1} \binom{n-1}{s} = 2^{n-1}$.

# Traveling salesman problem (TSP)

Given a weighted, complete graph $K_n$, we want to determine the Hamiltonian cycle of **smallest total weight** in $K_n$.

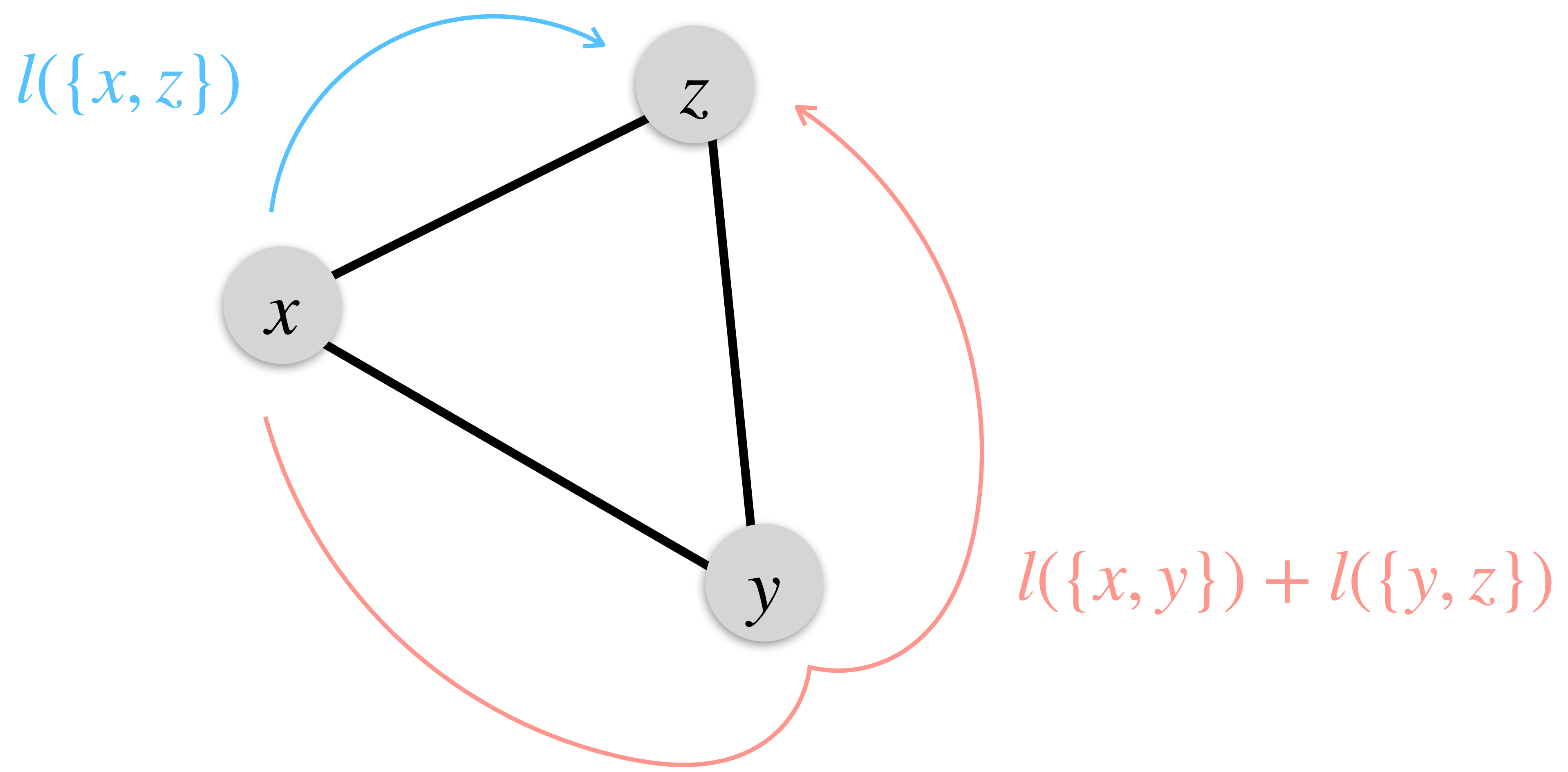
More formally, if $l$ is the edge weight function of $K_n$, we look for a Hamiltonian cycle $C$ such that

$$\sum_{e \in C} l(e) = min\{ \sum_{e \in C'} \mid C' \text{ is a Hamiltonian cycle in } K_n \}.$$

# Metric TSP

Same as TSP, except the edge weight function $l$ has to have the following property (triangle inequality)

$$l(\{x, z\}) \leq l(\{x, y\}) + l(\{y, z\}) \quad \text{for all } x, y, z \in [n].$$




traveling the direct route $\{x, z\}$ is cheaper than $\{x, y\} \to \{y, z\}$.

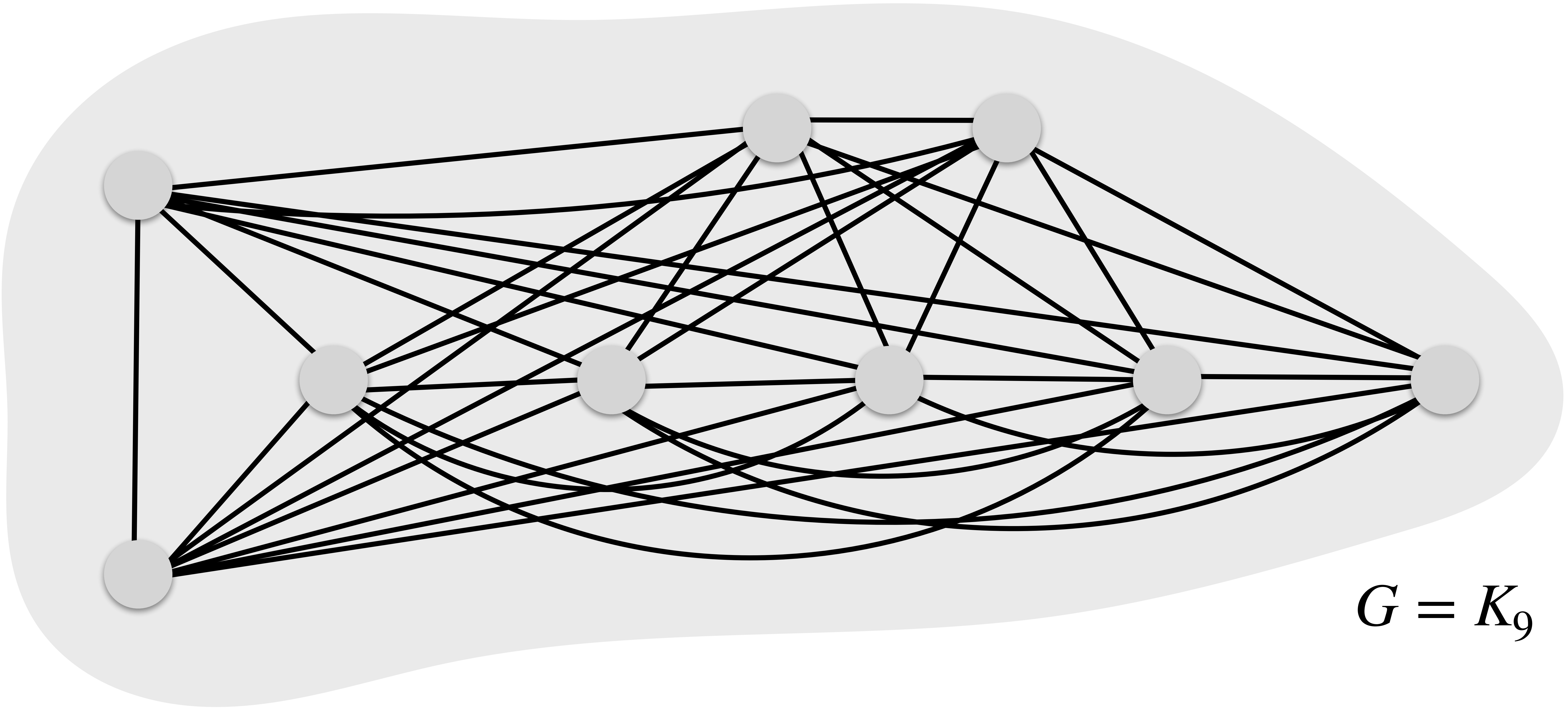
# 2-approximation-algorithm for metric TSP

**Input: complete** graph $G$, metric edge weight function $l$.

1. Determine MST $T$ in $G$.
2. Double all edges in $T$, call it $T'$ (a multigraph).
3. Find Eulerian circuit $E$ in $T'$.
4. Shorten $E$ (we will see what this means).

# 2-approximation-algorithm for metric TSP

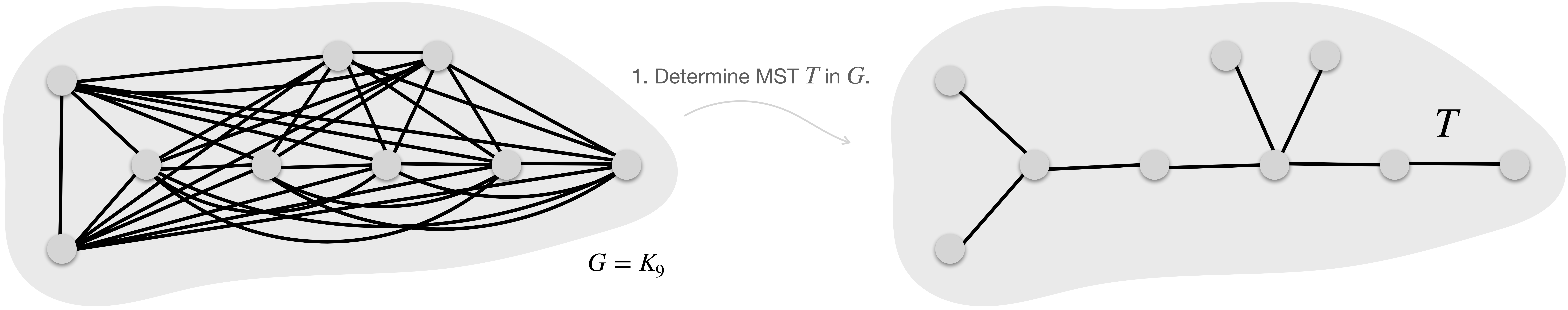
**Input:** complete graph $G$, metric edge weight function $l$.



$G = K_9$

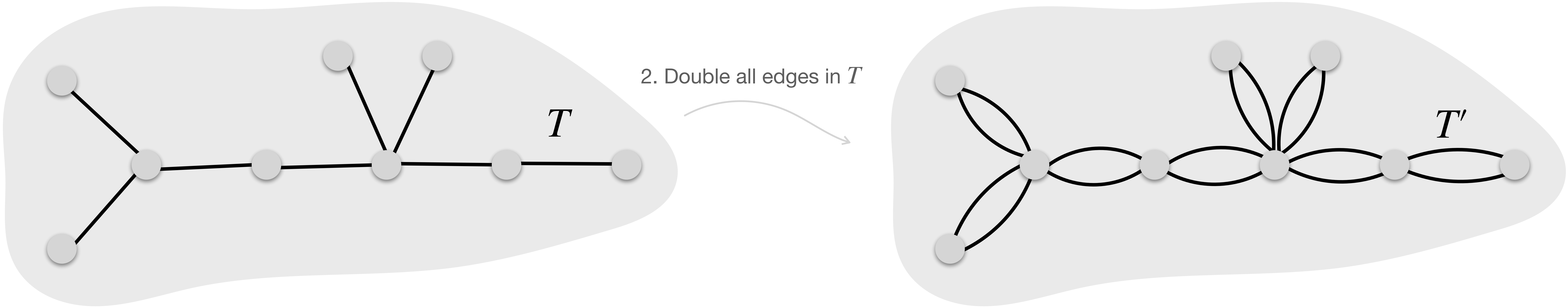
# 2-approximation-algorithm for metric TSP

**Input:** complete graph $G$, metric edge weight function $l$.



# 2-approximation-algorithm for metric TSP

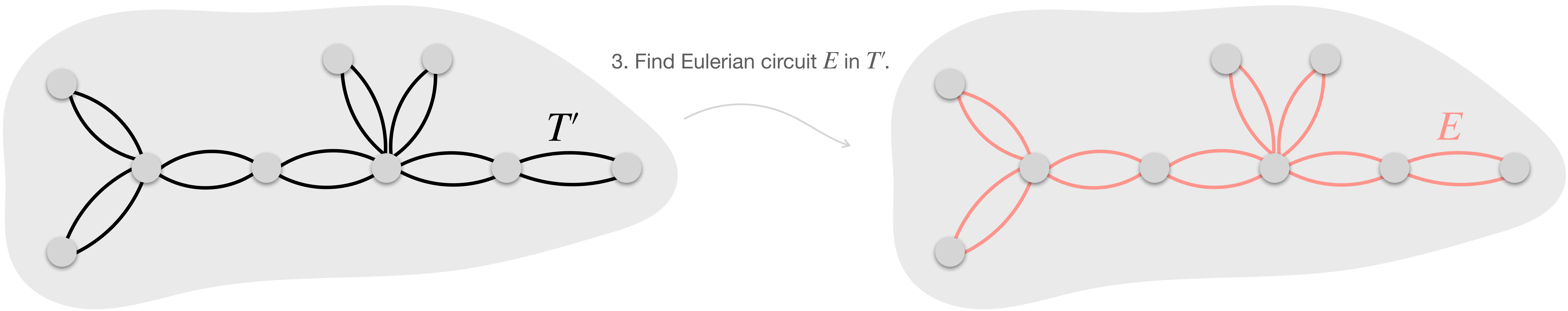
**Input:** complete graph $G$, metric edge weight function $l$.




(*) Note that $T'$ is a multigraph.

# 2-approximation-algorithm for metric TSP

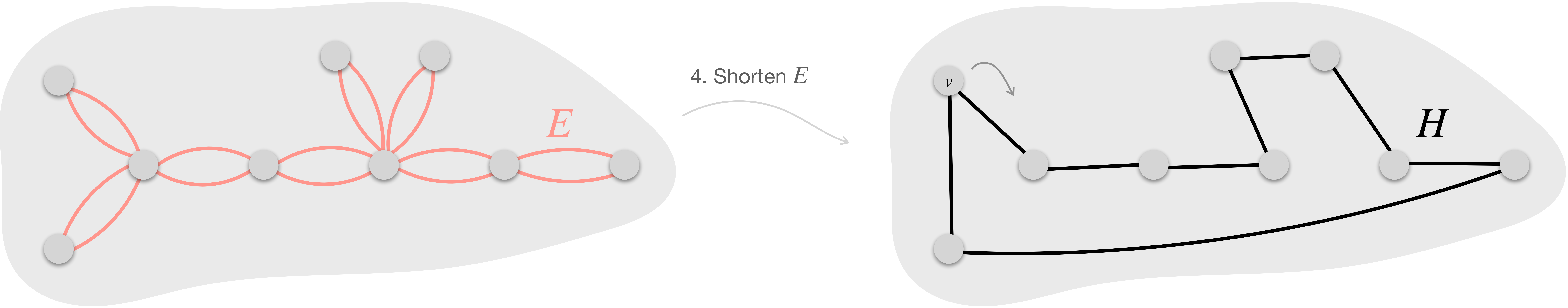
**Input:** complete graph $G$, metric edge weight function $l$.




(*) possible since every degree is even by construction of $T'$.

# 2-approximation-algorithm for metric TSP

**Input:** complete graph $G$, metric edge weight function $l$.

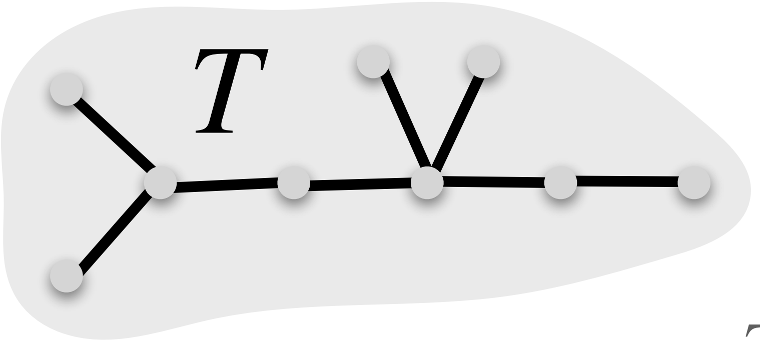



We shorten $E$ by walking along it and skipping vertices we already walked over. In this example we start at $v$ and walk to the right.

(*) here we use the fact that the edge weight function fulfills the triangle inequality.
(**) remember that $G$ is a complete graph, thus making it possible to walk over any edge we get by shortening.

# Analysis

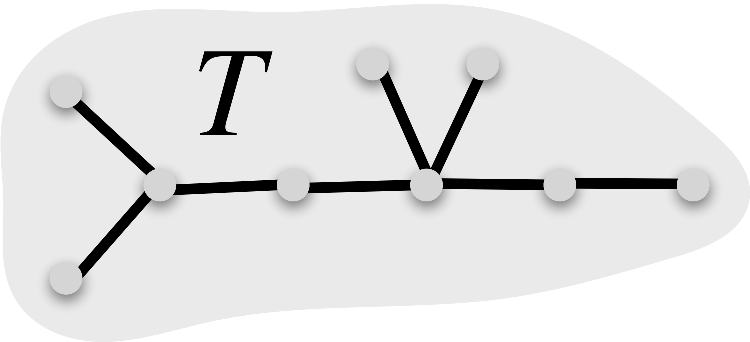
Let OPT be the cheapest Hamiltonian cycle. Then OPT$\setminus\{e\}$ is a spanning tree.

cost( 

) $\leq$ cost(OPT$\setminus\{e\}$) $\leq$ cost(OPT)

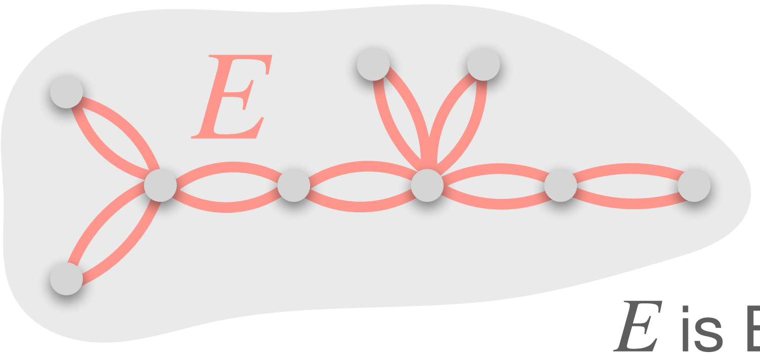
$T$ is a MST

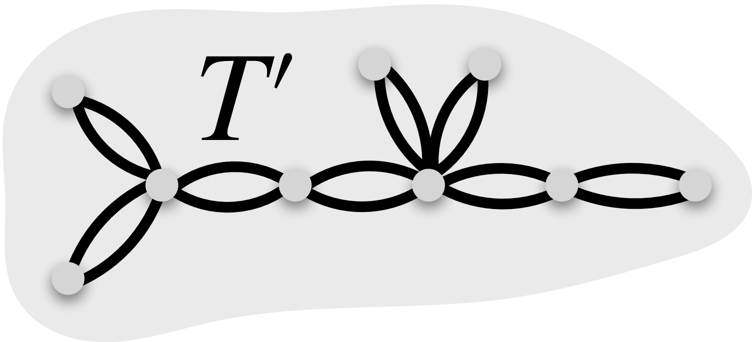
cost( 

) $= 2 \cdot$ cost( 

) $\leq 2 \cdot$ cost(OPT)

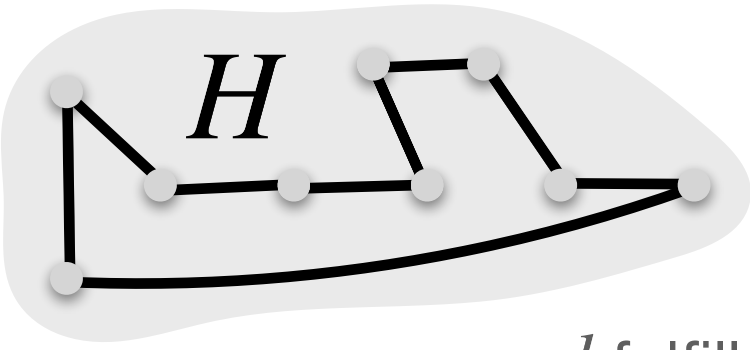
construction of $T'$

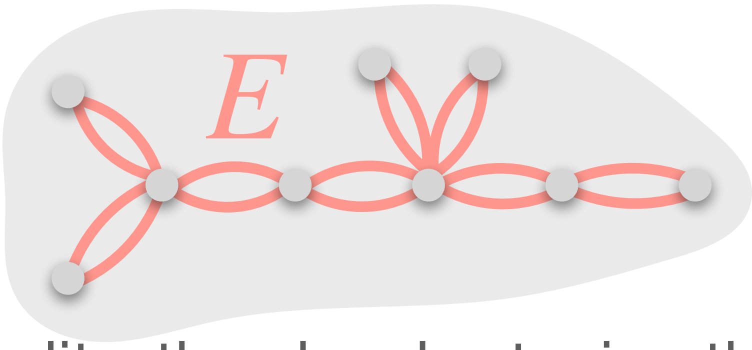
cost( 

) $=$ cost( 

) $\leq 2 \cdot$ cost(OPT)

$E$ is Eulerian circuit.

cost( 

) $\leq$ cost( 

) $\leq 2 \cdot$ cost(OPT)

$l$ fulfills the triangle inequality, thus by shortening the cost can only get shorter.

# Runtime

1. Determine MST $T$ in $G \implies O(n^2)$

   See theorem 1.19. Remember that $G$ is complete, thus $m = O(n^2)$.

2. Double all edges in $T$, call it $T'$ (a multigraph) $\implies O(m)$

3. Find Eulerian circuit $E$ in $T' \implies O(n)$

   Theorem 1.31. (b) we can find a Eulerian circuit in $O(m)$. Since $T'$ has $2(n-1)$ edges we have $O(n)$.

4. Shorten $E$ (we will see what this means) $\implies O(n)$

# Result

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