Graphen und diskrete Optimierung

im Bachelorstudiengang 'Informatik und Nachhaltigkeit'

Algorithmen für das TSP

Marie Schmidt

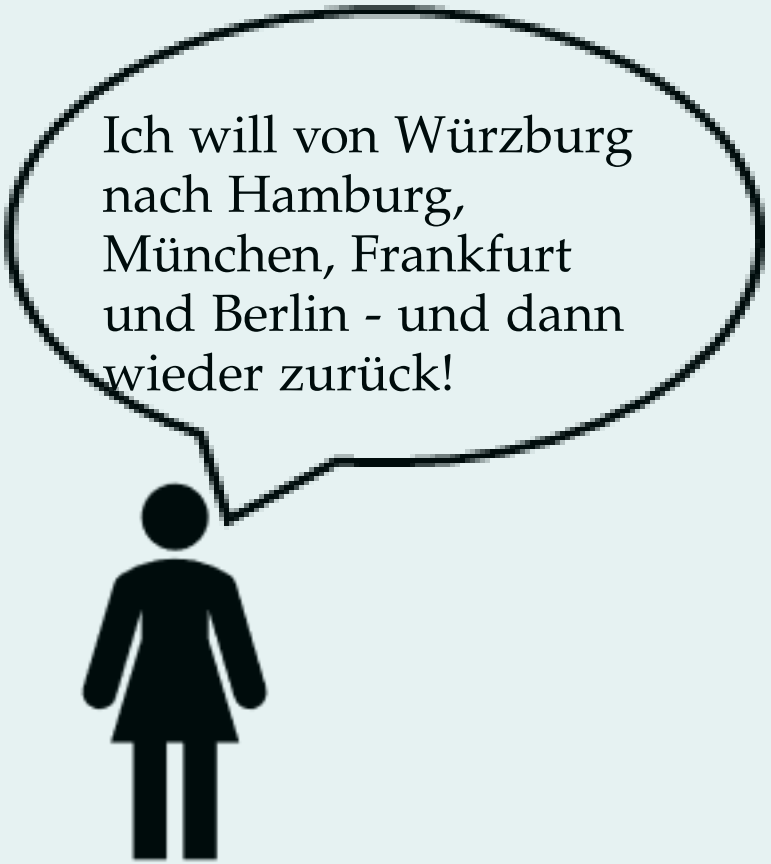
Ergänzung zur Vorlesung zu Wegen und Touren

Das Handlungsreisendenproblem

(*Eine* Variante des) **Handlungsreisendenproblem** (Englisch: **Traveling Salesperson Problem**, kurz **TSP**):
Gegeben: Eine Menge von Orten O und paarweise Entfernungen zwischen den Orten
Gesucht: Die kürzeste Rundtour, die jeden Ort genau einmal besucht.

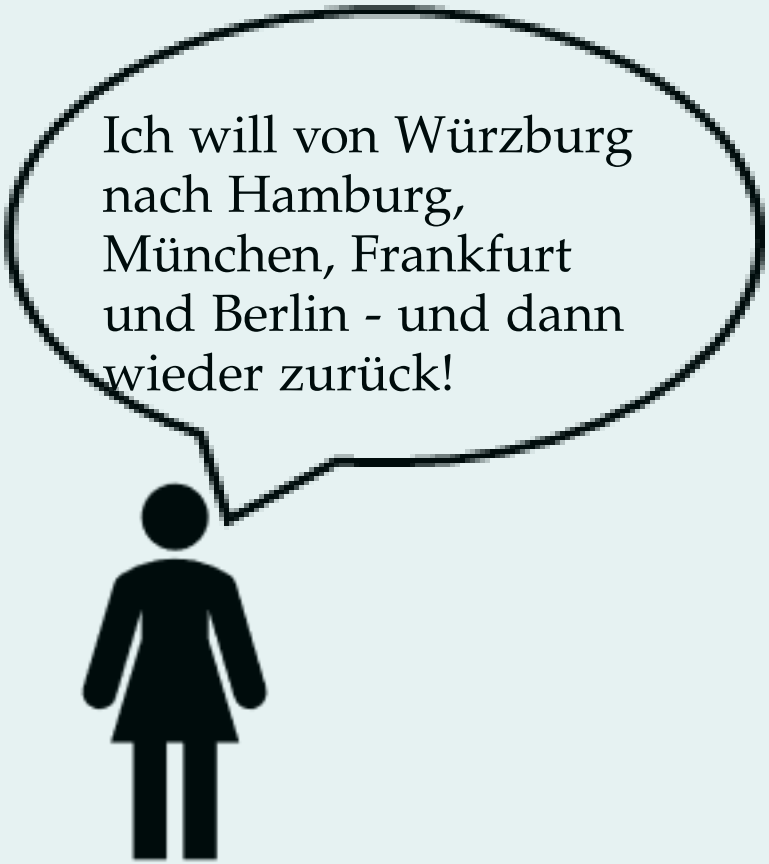
Das Handlungsreisendenproblem

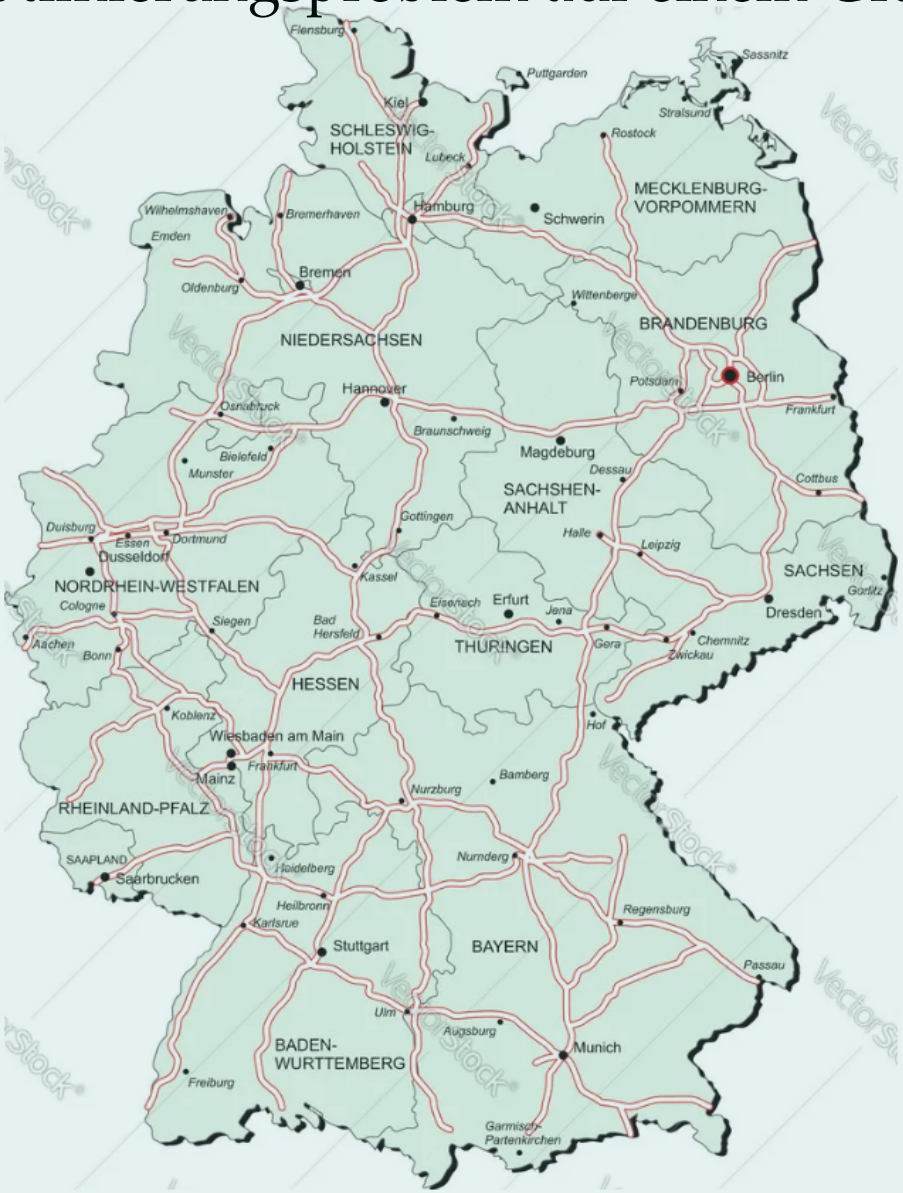
TSP als Optimierungsproblem auf einem Graph:



Das Handlungsreisendenproblem

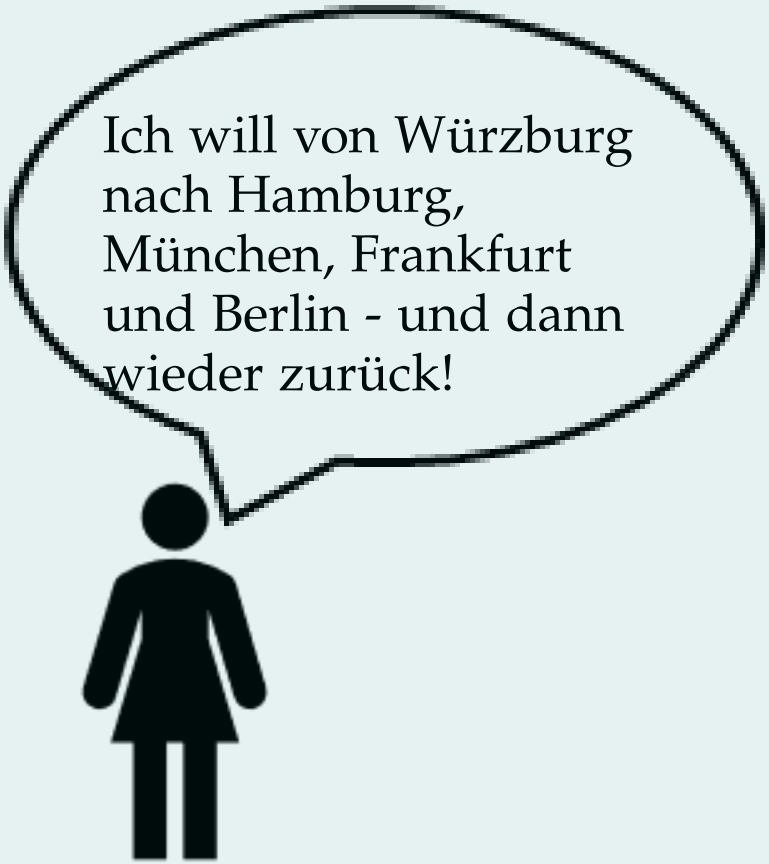
TSP als Optimierungsproblem auf einem Graph:

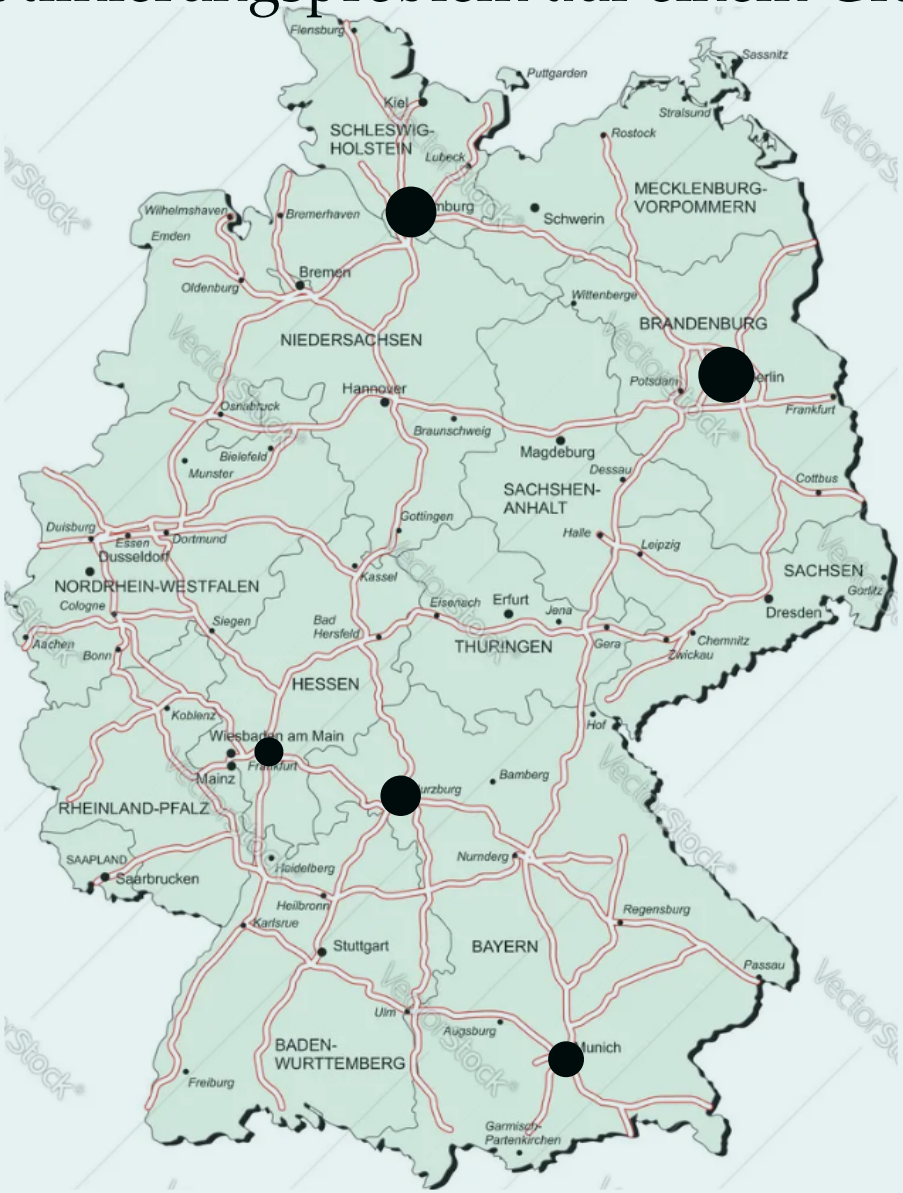




Das Handlungsreisendenproblem

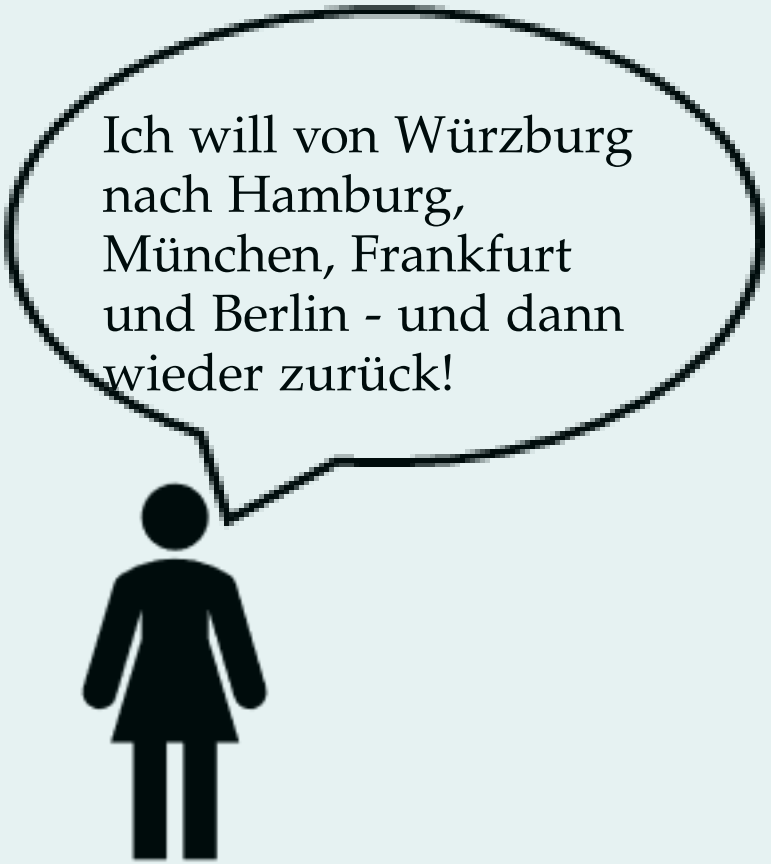
TSP als Optimierungsproblem auf einem Graph:

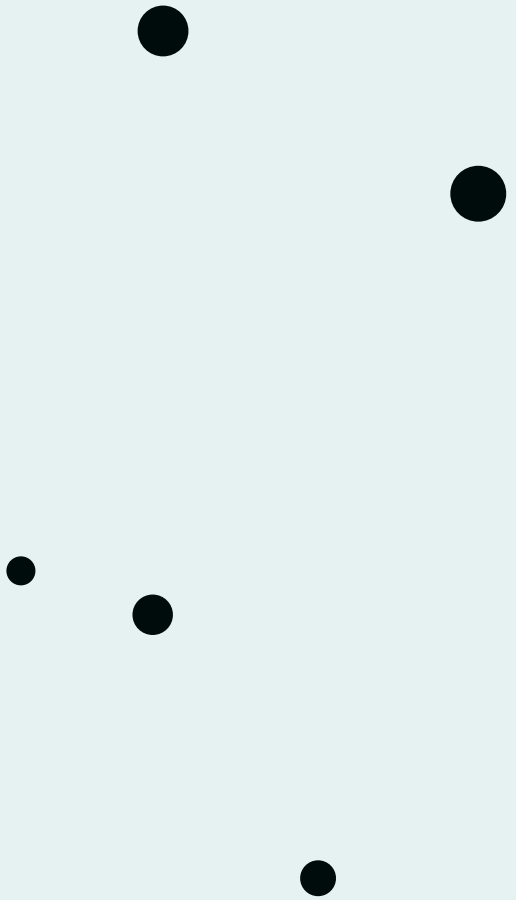




Das Handlungsreisendenproblem

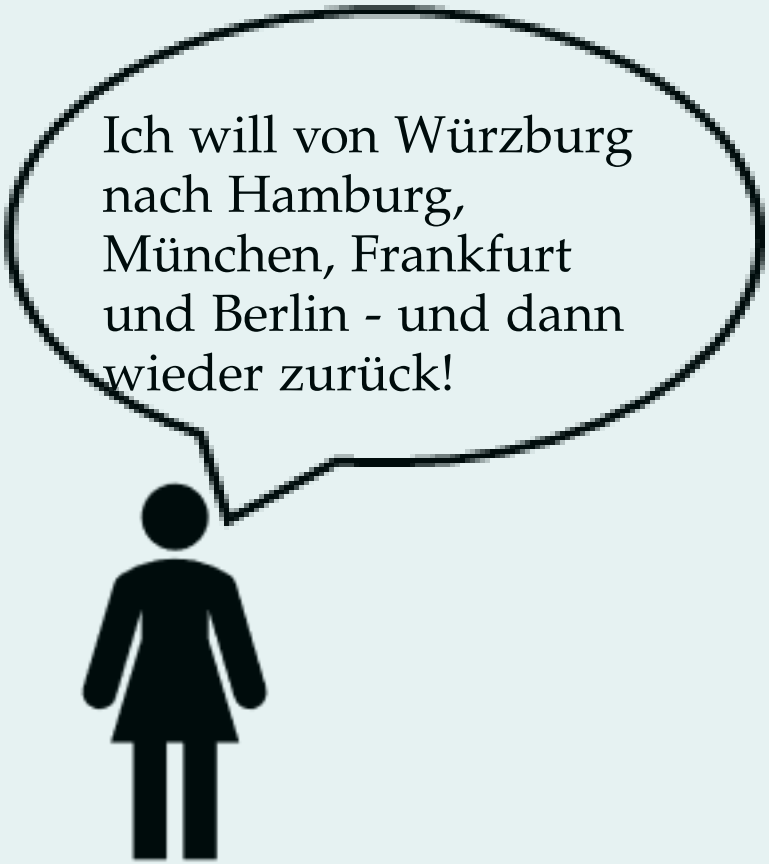
TSP als Optimierungsproblem auf einem Graph:

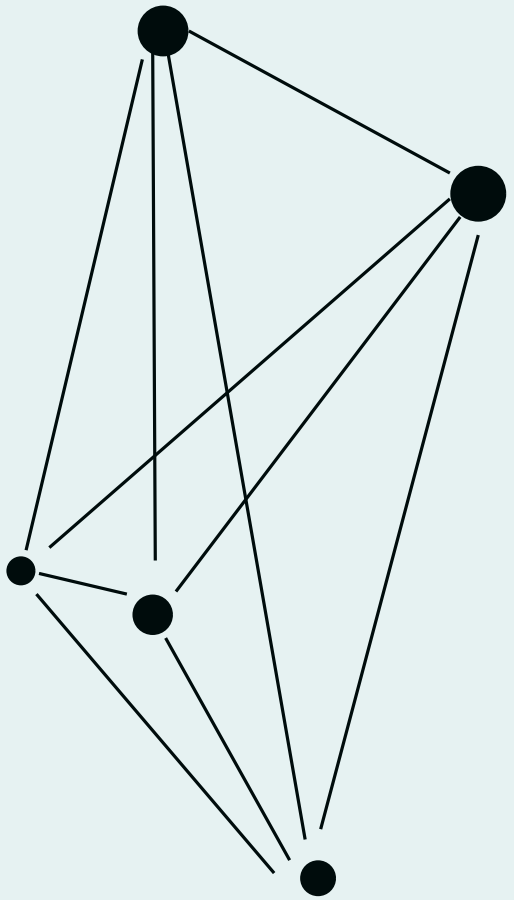




Das Handlungsreisendenproblem

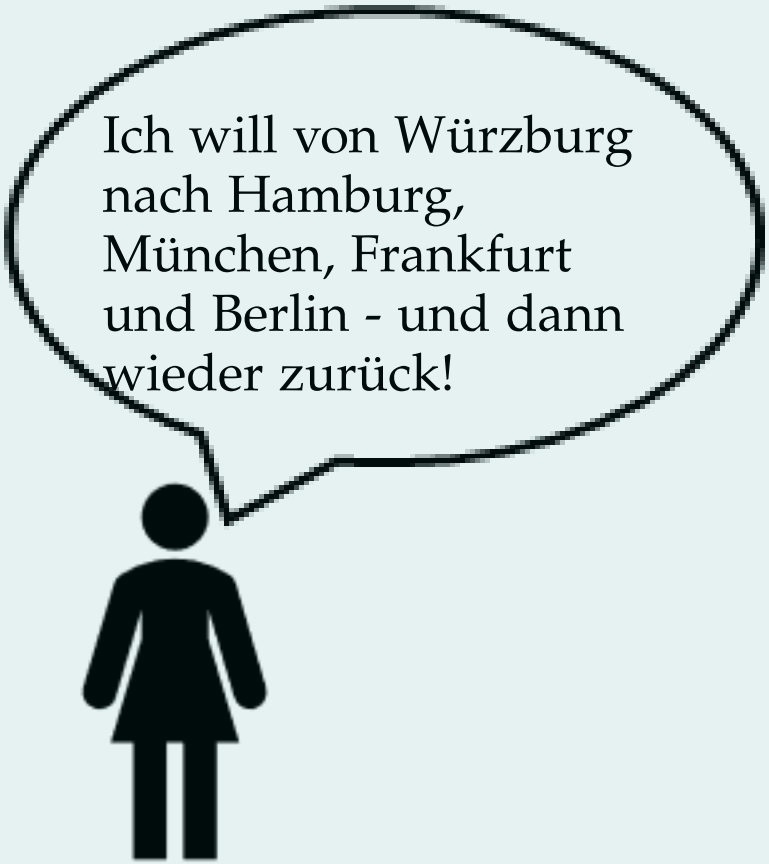
TSP als Optimierungsproblem auf einem Graph:

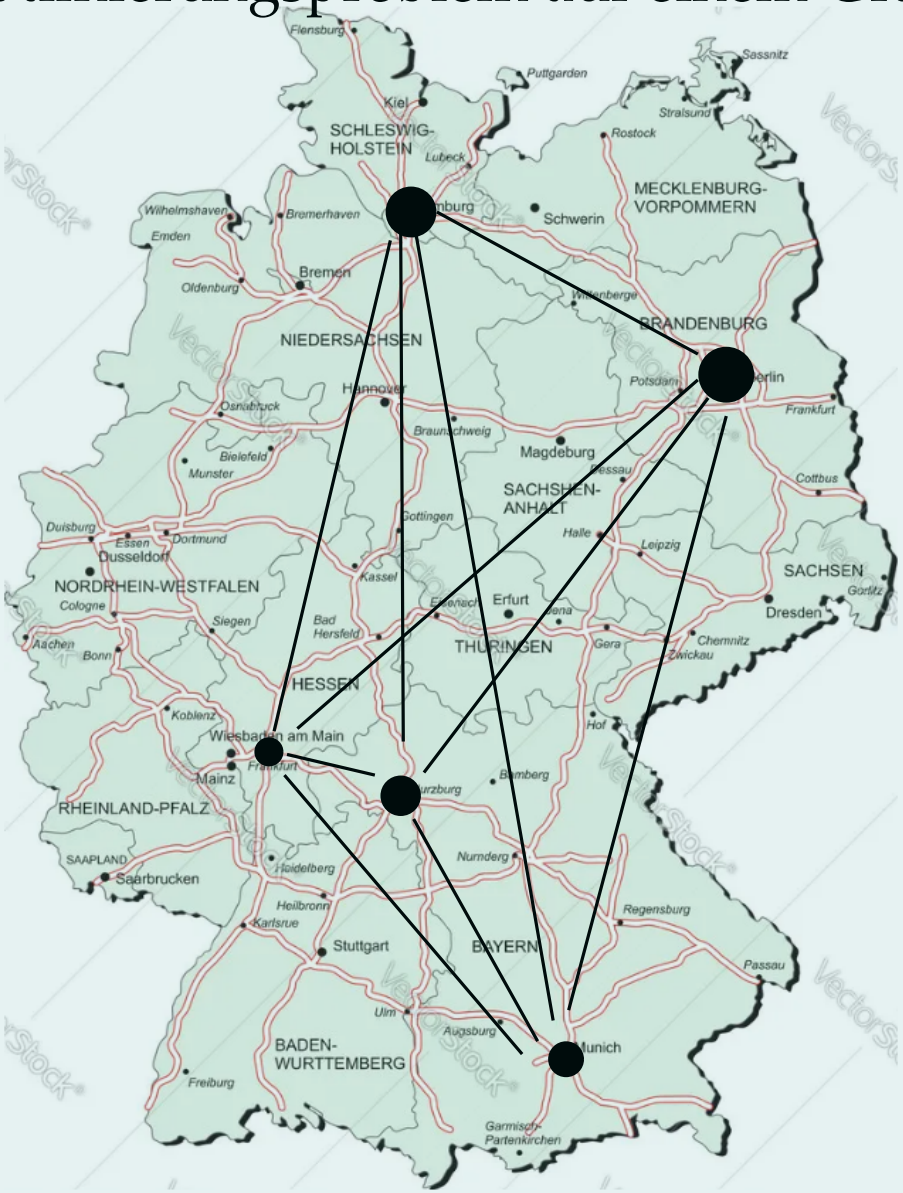




Das Handlungsreisendenproblem

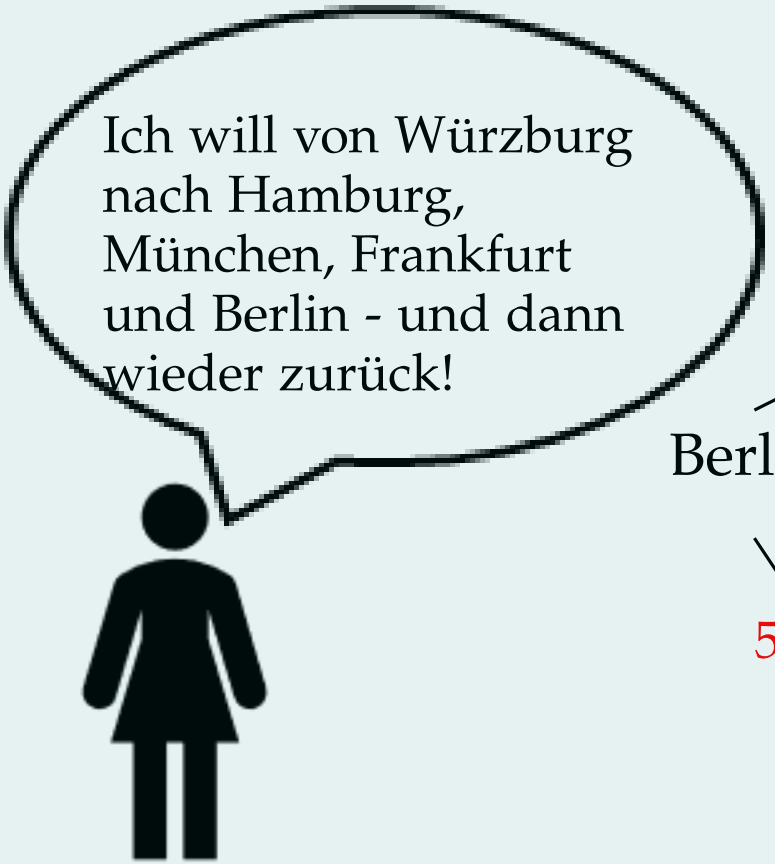
TSP als Optimierungsproblem auf einem Graph:

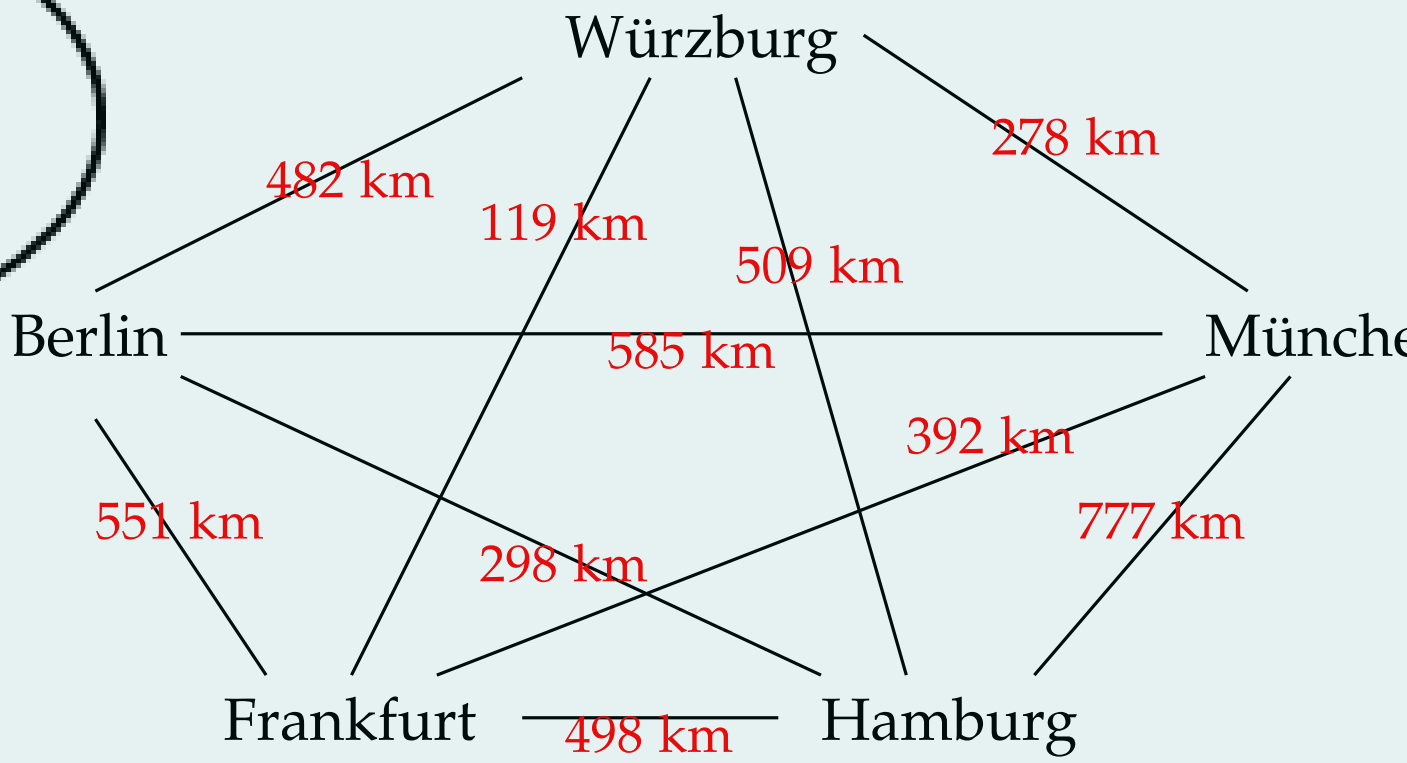




Das Handlungsreisendenproblem

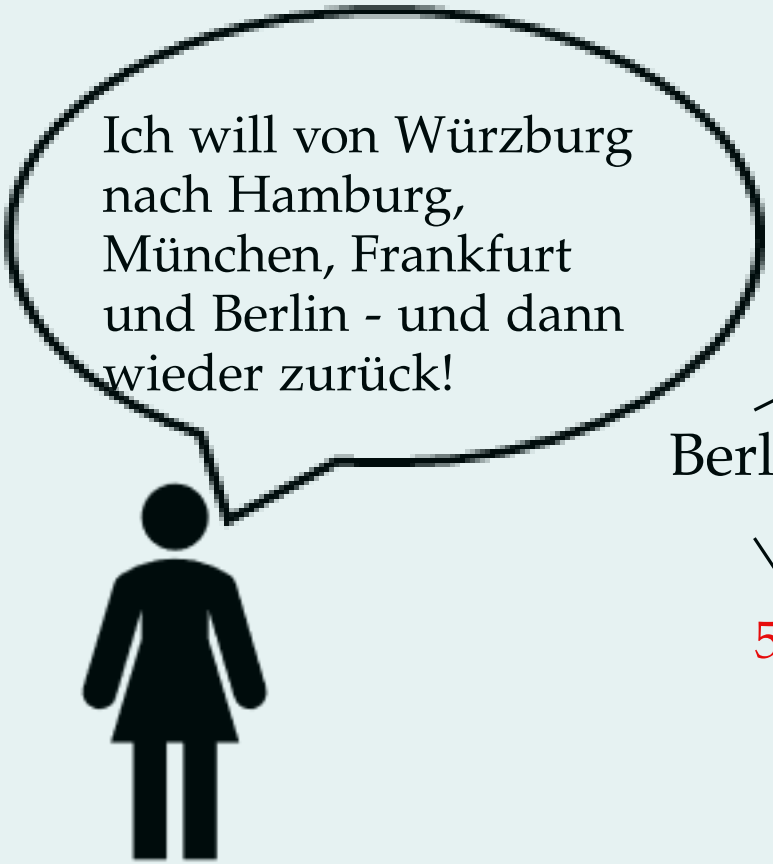
TSP als Optimierungsproblem auf einem Graph:

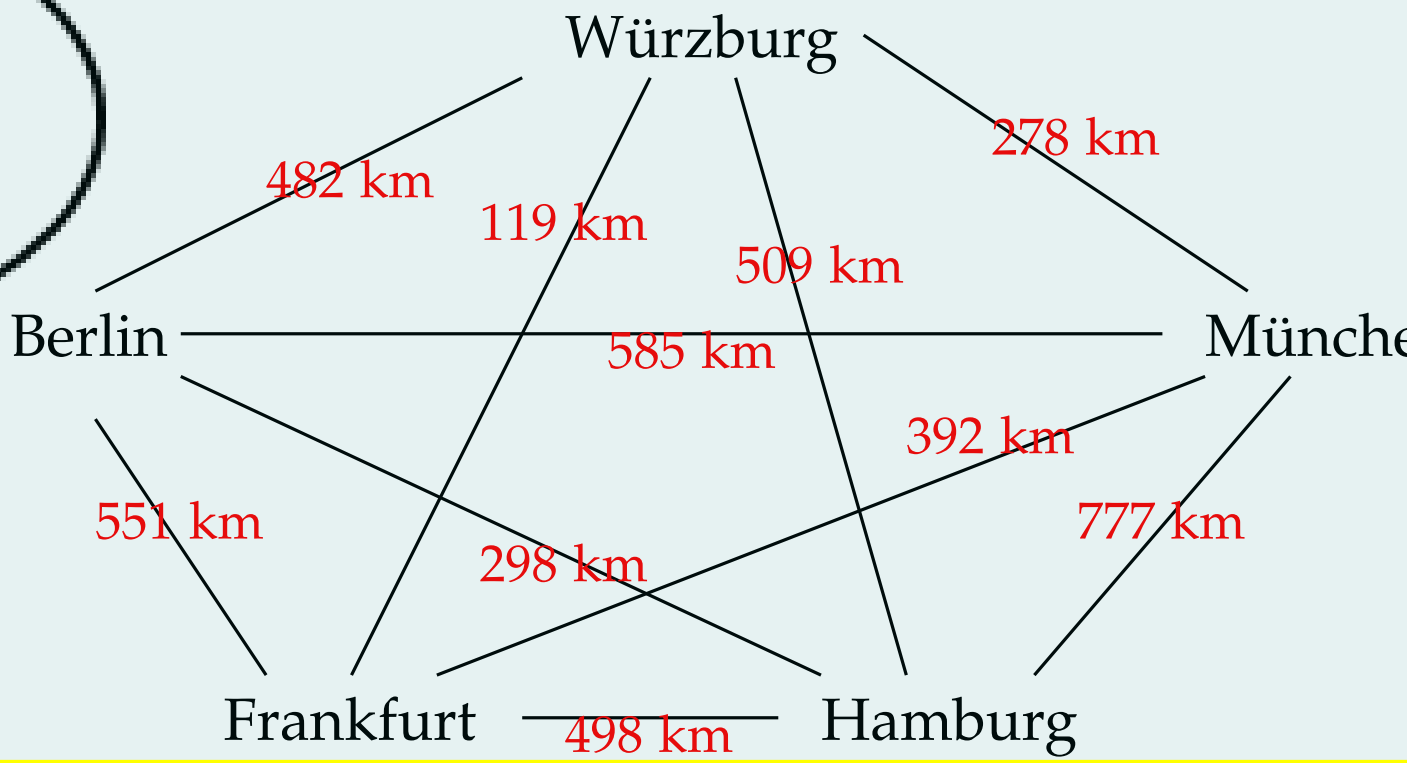




Das Handlungsreisendenproblem

TSP als Optimierungsproblem auf einem Graph:

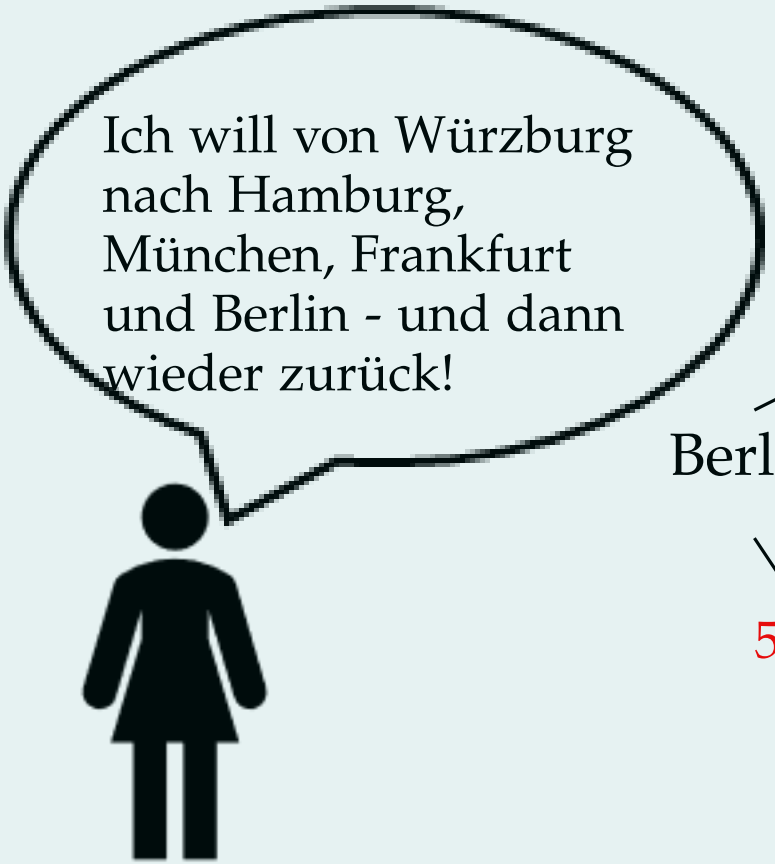


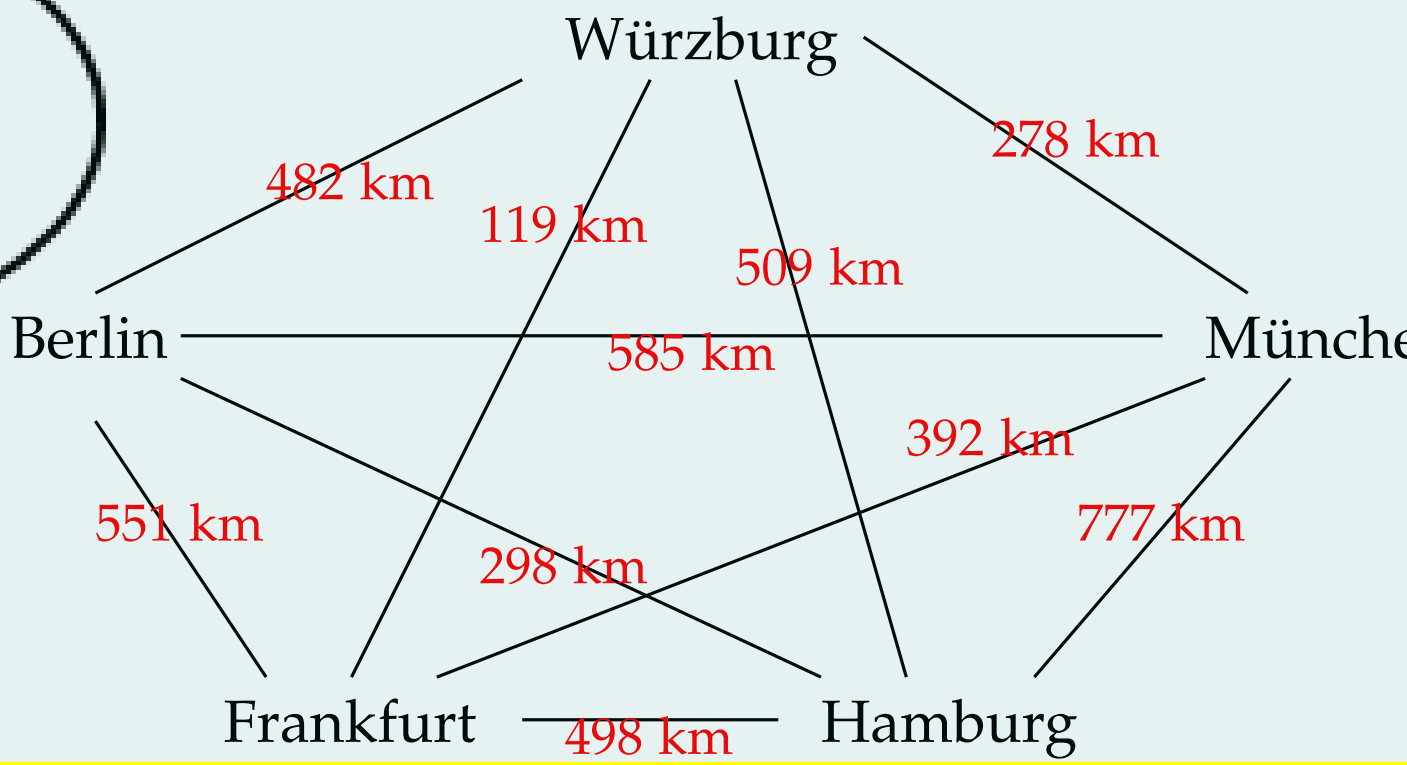


TSP auf Graph (eine Variante):
Gegeben: Vollständiger Graph $G = (V, E)$, Kantenlabels w_e für $e \in E$
Gesucht: Ein Kreis $K = (V_K, E_K)$, der alle Knoten besucht, so dass $\sum_{e \in E_K} w_e$ minimal

Das Handlungsreisendenproblem

TSP als Optimierungsproblem auf einem Graph:



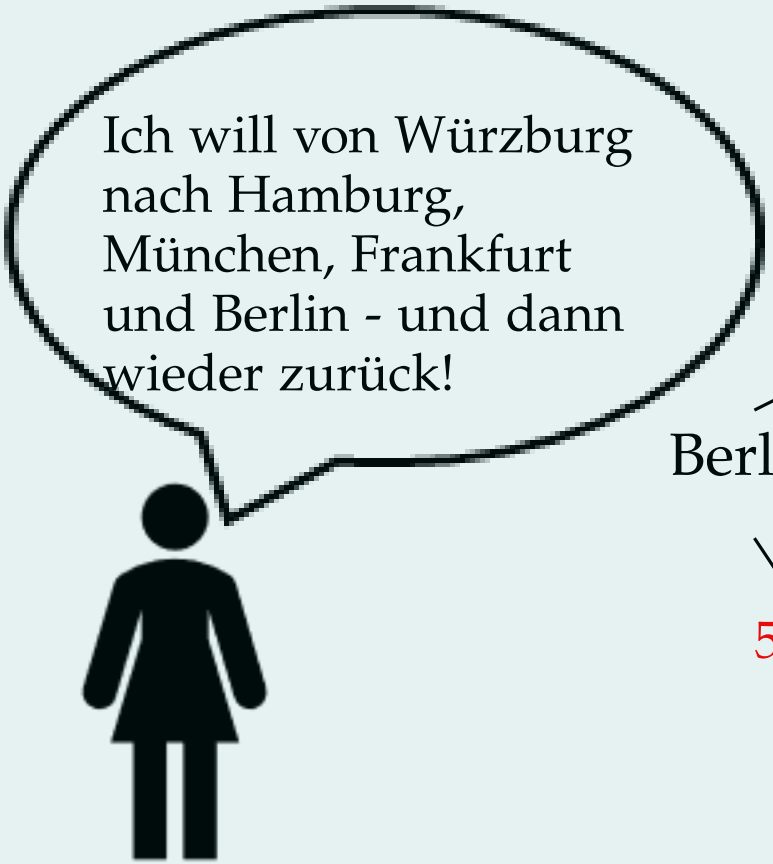


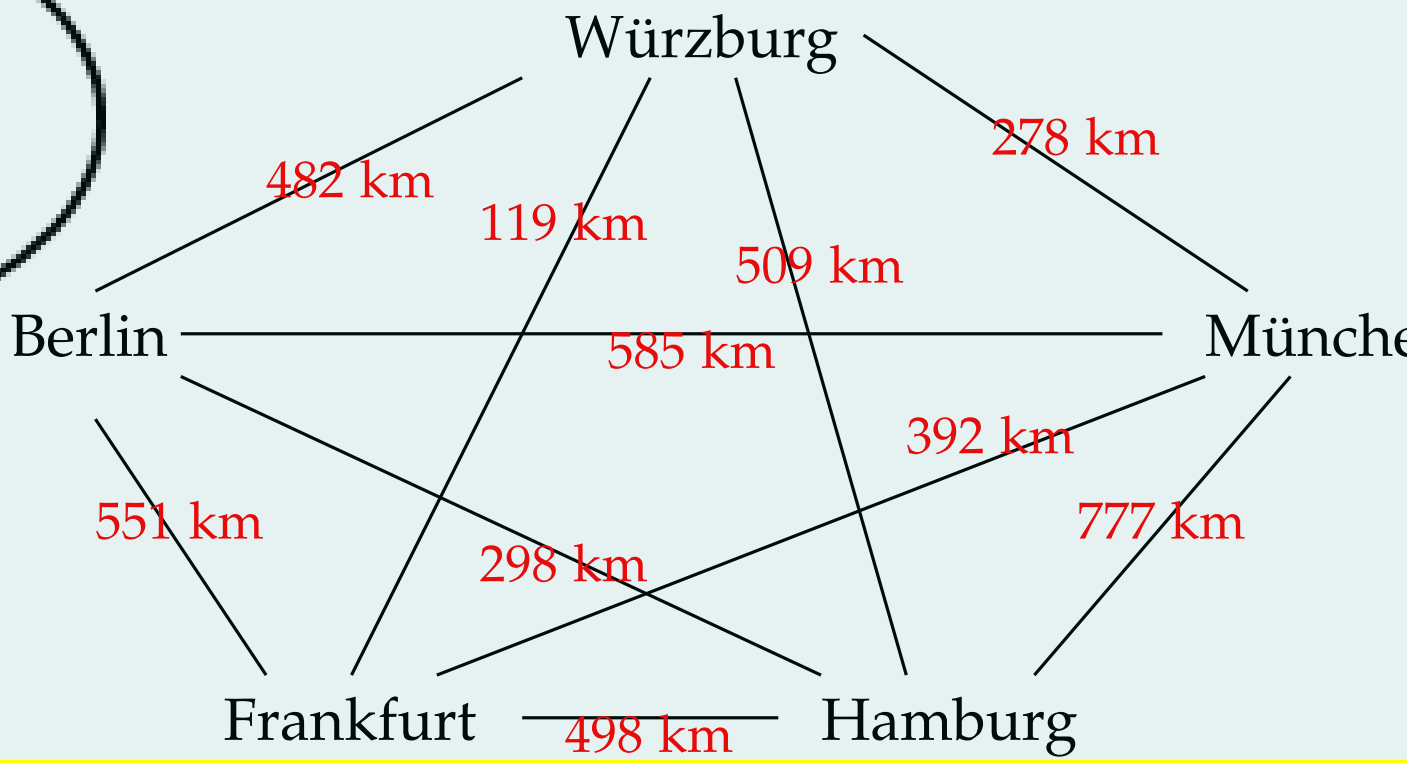
TSP auf Graph (eine Variante):
Gegeben: Vollständiger Graph $G = (V, E)$, Kantenlabels w_e für $e \in E$
Gesucht: Ein Kreis $K = (V_K, E_K)$, der alle Knoten besucht, so dass $\sum_{e \in E_K} w_e$ minimal

Ein ungerichteter Graph $G = (V, E)$ heißt **vollständig**, wenn für alle $v, w \in V$ mit $v \neq w$ gilt $\{v, w\} \in E$.

Das Handlungsreisendenproblem

TSP als Optimierungsproblem auf einem Graph:

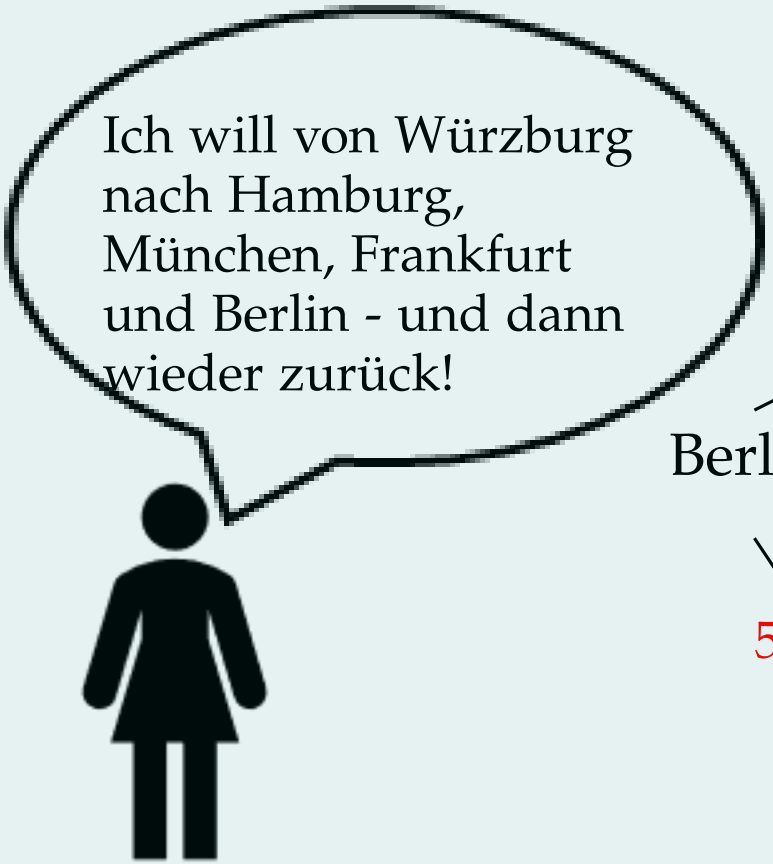


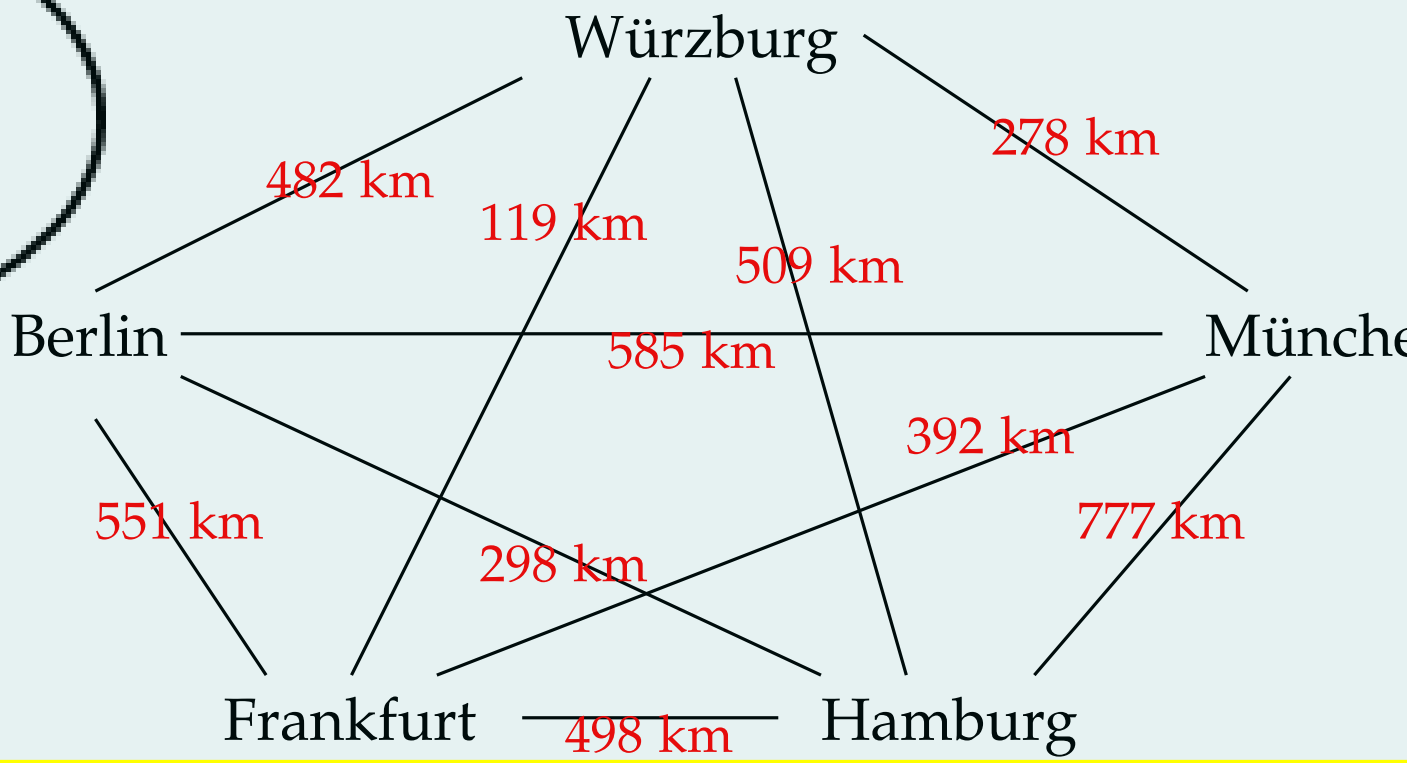


TSP auf Graph (eine Variante):
Gegeben: Vollständiger Graph $G = (V, E)$, Kantenlabels w_e für $e \in E$
Gesucht: Ein Kreis $K = (V_K, E_K)$, der alle Knoten besucht, so dass $\sum_{e \in E_K} w_e$ minimal

Das Handlungsreisendenproblem

TSP als Optimierungsproblem auf einem Graph:



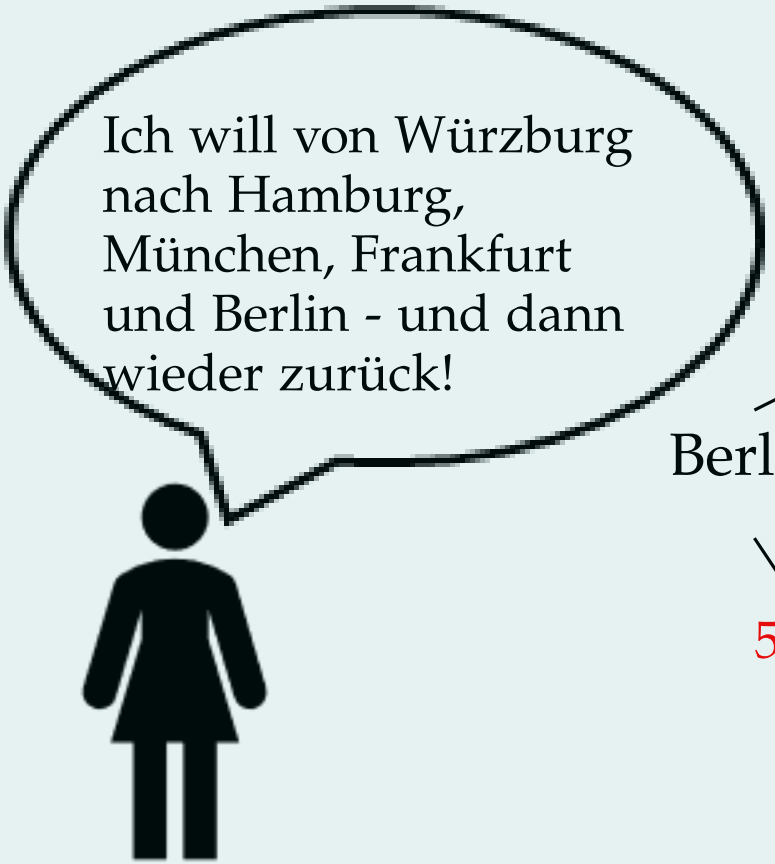


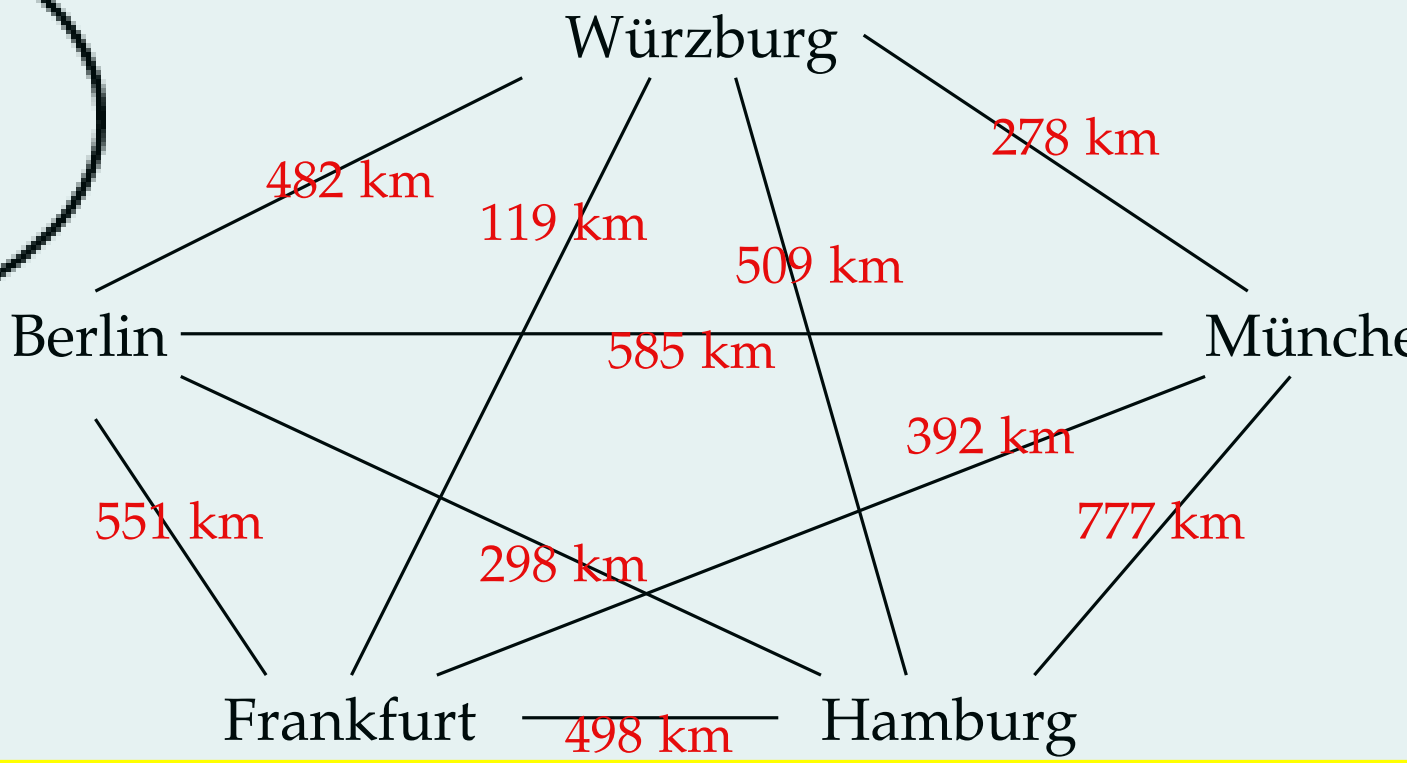
TSP auf Graph (eine Variante):
Gegeben: Vollständiger Graph $G = (V, E)$, Kantenlabels w_e für $e \in E$
Gesucht: Ein Kreis $K = (V_K, E_K)$, der alle Knoten besucht, so dass $\sum_{e \in E_K} w_e$ minimal

Algorithmus um optimale TSP-Tour zu finden?

Das Handlungsreisendenproblem

TSP als Optimierungsproblem auf einem Graph:





TSP auf Graph (eine Variante):
Gegeben: Vollständiger Graph $G = (V, E)$, Kantenlabels w_e für $e \in E$
Gesucht: Ein Kreis $K = (V_K, E_K)$, der alle Knoten besucht, so dass $\sum_{e \in E_K} w_e$ minimal

Zusammenhang Hamiltonscher Kreis?

Algorithmen für das Handlungsreisendenproblem

Algorithmen für das Handlungsreisendenproblem

Nearest-Neighbor-Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
    Wähle unter unbesuchten Knoten einen Knoten v^* mit kürzesten Abstand zu v.
    Setze $K = K \circ (\{v, v^*\}, v^*)$.
    Setze $v = v^*$.
end while
return K

Algorithmen für das Handlungsreisendenproblem

Nearest-Neighbor-Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
  Wähle unter unbesuchten Knoten einen Knoten v^* mit kürzesten Abstand zu v.
  Setze $K = K \circ (\{v, v^*\}, v^*)$.
  Setze $v = v^*$.
end while
return K

Laufzeit?
Findet dieser Algorithmus (immer) eine Optimallösung?

Algorithmen für das Handlungsreisendenproblem

Nearest-Neighbor-Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
  Wähle unter unbesuchten Knoten einen Knoten v^* mit kürzesten Abstand zu v.
  Setze $K = K \circ (\{v, v^*\}, v^*)$.
  Setze $v = v^*$.
end while
return K

Exakter Algorithmus für Problem P: Algorithmus, der für jede Instanz des Problems P eine Optimallösung findet

Heuristischer Algorithmus (kurz: **Heuristik**) für Problem P: Algorithmus, der eine *möglichst gute zulässige* Lösung für P findet.

Algorithmen für das Handlungsreisendenproblem

Nearest-Neighbor-Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
  Wähle unter unbesuchten Knoten einen Knoten v^* mit kürzesten Abstand zu v.
  Setze $K = K \circ (\{v, v^*\}, v^*)$.
  Setze $v = v^*$.
end while
return K

Exakter Algorithmus für Problem P: Algorithmus, der für jede Instanz des Problems P eine Optimallösung findet

Heuristischer Algorithmus (kurz: **Heuristik**) für Problem P: Algorithmus, der eine *möglichst gute zulässige* Lösung für P findet.

$\rightarrow$ Nearest-Neighbor ist eine Heuristik für das TSP

Algorithmen für das Handlungsreisendenproblem

Nearest-Neighbor-Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
  Wähle unter unbesuchten Knoten einen Knoten v^* mit kürzesten Abstand zu v.
  Setze $K = K \circ (\{v, v^*\}, v^*)$.
  Setze $v = v^*$.
end while
return K

Nearest Neighbor ist eine sogenannter Greedy-Algorithmus

Greedy-Algorithmus: Klasse von Algorithmen, die schrittweise den Folgezustand auswählen, der zum Zeitpunkt der Wahl den größten Gewinn bzw. das beste Ergebnis (berechnet durch eine Bewertungsfunktion) verspricht

Algorithmen für das Handlungsreisendenproblem

Noch ein Greedy-Ansatz: Cheapest Insertion Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
    Wähle unter unbesuchten Knoten einen Knoten v^*, der sich kostengünstigst in den Kreis K einsetzen lässt.
    Setze diesen Knoten in K ein.
    Setze $v = v^*$.
end while
return K

Algorithmen für das Handlungsreisendenproblem

Noch ein Greedy-Ansatz: Cheapest Insertion Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
    Wähle unter unbesuchten Knoten einen Knoten v^*, der sich kostengünstigst in den Kreis K einsetzen lässt.
    Setze diesen Knoten in K ein.
    Setze $v = v^*$.
end while
return K

Heuristik oder exakt?

Algorithmen für das Handlungsreisendenproblem

Noch ein Greedy-Ansatz: Cheapest Insertion Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
    Wähle unter unbesuchten Knoten einen Knoten v^*, der sich kostengünstigst in den Kreis K einsetzen lässt.
    Setze diesen Knoten in K ein.
    Setze $v = v^*$.
end while
return K

Heuristik oder exakt? $\rightarrow$ Heuristik

Nearest Neighbor und Cheapest Insertion sind sogenannte *Konstruktionsheuristiken* für das TSP.

Algorithmen für das Handlungsreisendenproblem

Noch ein Greedy-Ansatz: Cheapest Insertion Algorithmus

Input: vollständiger Graph $G = (V, E)$ mit Kantenlabels w_e
Output: eine TSP-Tour (Kreis) K in G

Wähle Knoten v_0. Setzte $K = (v_0)$.
Setze $v = v_0$.
while Noch nicht alle Knoten besucht **do**
  Wähle unter unbesuchten Knoten einen Knoten v^*, der sich kostengünstigst in den Kreis K einsetzen lässt.
  Setze diesen Knoten in K ein.
  Setze $v = v^*$.
end while
return K

Heuristik oder exakt? $\rightarrow$ Heuristik

Nearest Neighbor und Cheapest Insertion sind sogenannte *Konstruktionsheuristiken* für das TSP.

Weitere TSP-Konstruktionsheuristiken: Nearest insertion, Farthest insertion, ...