DHBW Lörrach
Duale Hochschule Baden-Württemberg

Algorithmen und Komplexität
TIF 21 A/B
Dr. Bruno Becker

10. Komplexitätstheorie

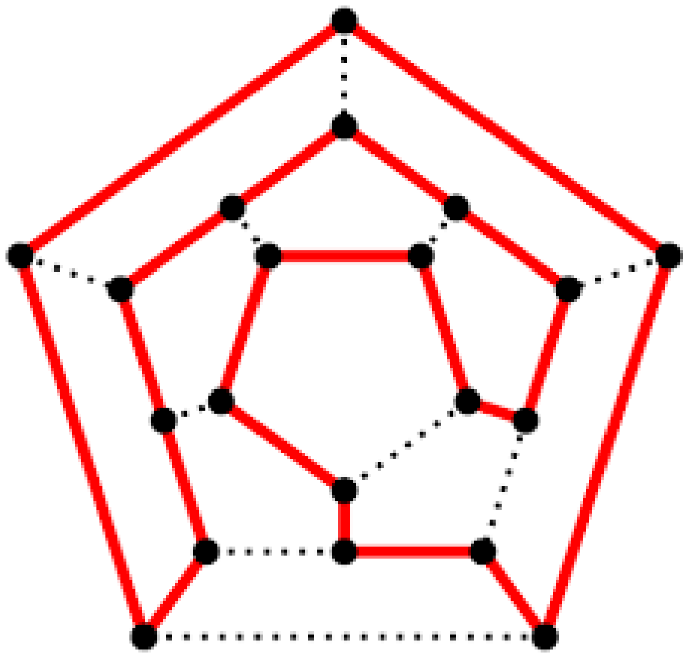
10.2. Travelling Salesman Problem

Travelling Salesman Problem

- **Problemdefinition**
- Exakte Lösungsverfahren
- Heuristiken
- Näherungslösungen

Hamiltonkreis

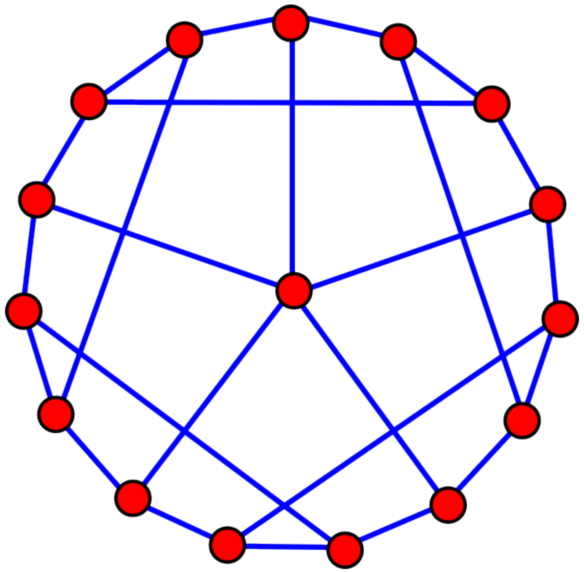
Ein **Hamiltonkreis** (*Rundreise, Tour)* in einem Graphen *G* ist ein Zyklus, der jeden Knoten von *G* einmal besucht.



Hamiltonkreis-Problem

Hat ein gegebener Graph einen Hamiltonkreis?

- Schwieriges (***NP***-vollständiges) Problem
- Alle bekannten Algorithmen haben im worst case exponentielle Laufzeit (in Anzahl der Knoten)

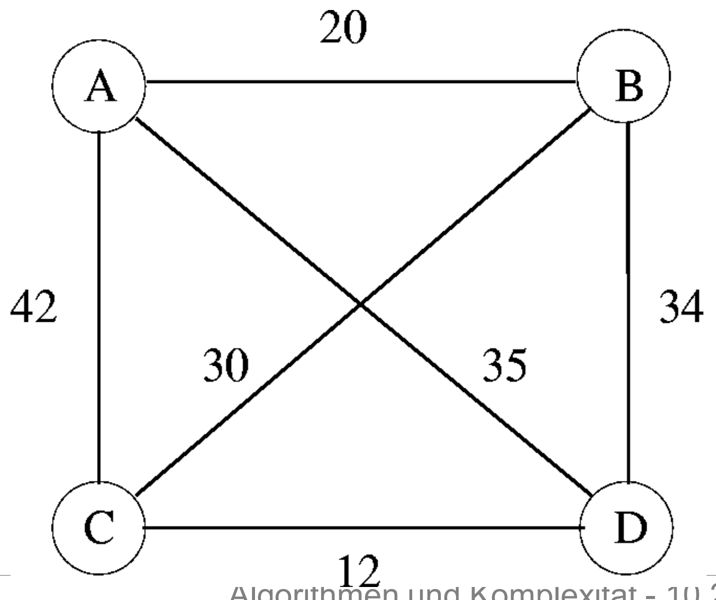


Travelling Salesman Problem (TSP)

Gegeben: Kantengewichteter, vollständiger Graph

- **Vollständig** → Hamiltonkreis existiert
- **Nichtnegative Kantengewichte**

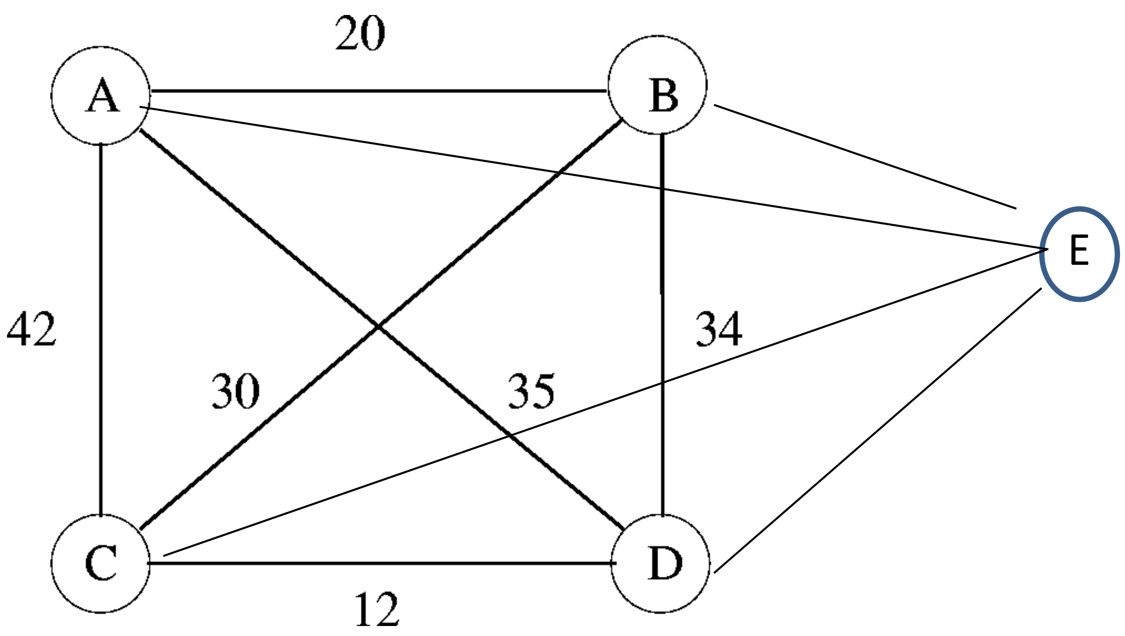
Gesucht: Hamiltonkreis mit minimalem Gesamtgewicht



TSP für unvollständige Graphen

Vollständigkeit ist keine echte Einschränkung:

→ Fehlende Kanten mit Gewicht ∞ ergänzen



Wichtige Spezialfälle des TSP

1. Metrisches TSP

Alle Kantengewichte *d* erfüllen die **Dreiecksungleichung**

$$d_{st} \leq d_{su} + d_{ut}$$

2. Euklidisches TSP

Knoten entsprechen Punkten in einem *d*-dimensionalen Raum

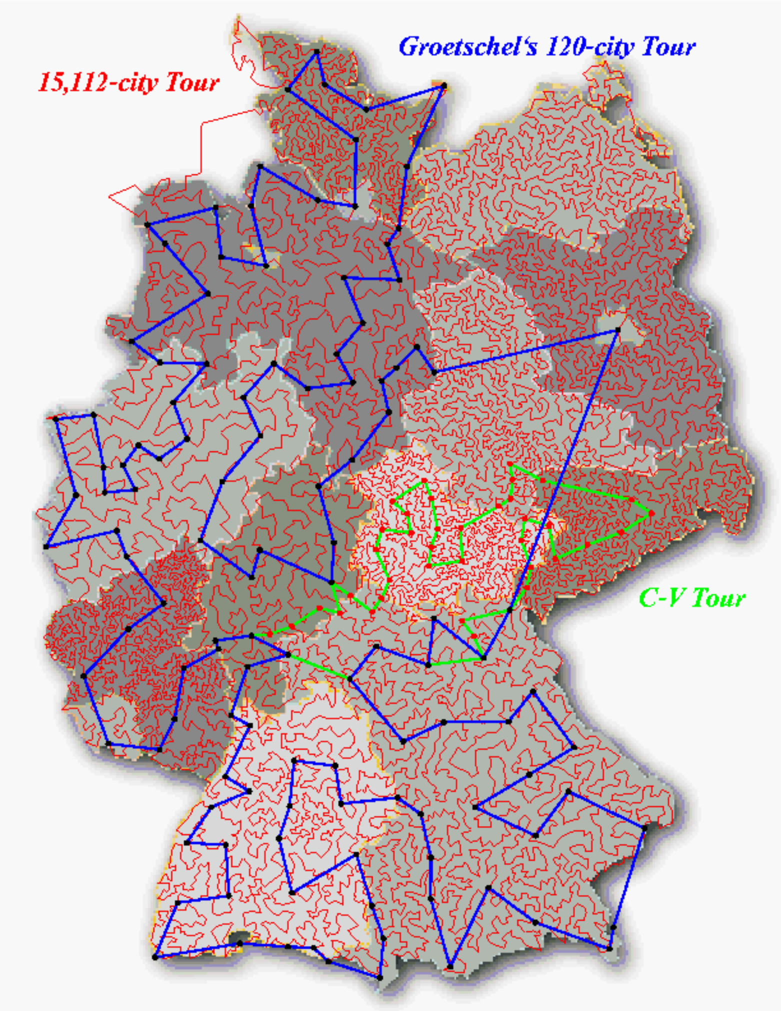
Kantengewichte = Euklidischer Abstand

Für diese „leichteren" Spezialfälle existieren spezielle Algorithmen

Anwendungsfälle für TSP

- Transport/Logistik: Tourenplanung (Spedition, Paketdienst, Crew-Planung Flughafen,…)
- Lager: Optimiertes „Picking“
- Produktionsplanung: Minimierung von Rüstzeiten
- Optimiertes VLSI-Design
- DNA-Analyse: Zusammensetzen von Gensequenzen
- ….

Optimale Touren durch Deutschland



courses.cs.vt.edu

Travelling Salesman Problem

- Problemdefinition
- **Exakte Lösungsverfahren**
- Heuristiken
- Näherungslösungen

Vollständiges Durchsuchen aller Möglichkeiten

- Bei *n* Knoten gibt es *(n-1)!* Möglichkeiten!

→Nur für kleine *n* *„erlebbar"*

n	n!
5	120
10	3.628.800
20	$2{,}4 * 10^{18}$
50	$2{,}4 * 10^{64}$
100	$9{,}3 * 10^{157}$

Dynamische Programmierung: Ansatz für TSP

Idee: Lösung bottom-up aus Teilproblemen aufbauen:

- Teilproblem: Finde kürzeste Rundreise von Knoten in einer Menge $S \subset V$ zulässiger Knoten
- Aber: Zyklen stören bei Erweiterung der Teillösungen

Ansatz: Sei $S \subset V$, mit $1, i \in S$.

$P(i,S)$ = kürzester, *einfacher* Weg von 1 über jeden Knoten in S nach Knoten i.

- Basisfälle $P(1, \{1\}) = 0$; $P(1, S) = \infty$ für S mit mehr als 1 Knoten
- Länge einer kürzesten Rundtour:

$$\min\left(P(i,\{1,..,n\}) + d_{i1}\text{ ; für alle } i \text{ von 1 bis } n\right) \quad (d_{ij} \text{ Kantengewicht } \langle i,j\rangle)$$

Dynamische Programmierung: TSP-Algorithmus

Für $|S| > 1$ gilt:

$$P(j,S) = \min(P(i, S\setminus \{j\}) + d_{ij}\ ;\ \text{für alle } i \in S\setminus \{j\}\ )$$

```
P(1, {1}) = 0;
for (k = 2,..,n)
    {
     foreach S ⊂ {1,2,..,n} with |S| = k and 1 ∈ S
          {
           P(1, S) = ∞;
           foreach j ∈ S, j !=1
                {
                P(j, S) = min(P(i, S\ {j})+ dij ; für alle i ∈ S\ {j} )
                }
          }
    }
return min(P(i,{1,..,n})+ d_{i,1} ; für alle i von 1 bis n )
```

Dynamische Programmierung: Aufwand TSP

- Die Anzahl der Teilmengen von $\{1,\ldots,n\}$ ist 2^n.
- Zu jeder solcher Teilmenge gibt es maximal *n* Teilprobleme
- Jedes Teilproblem kann in Zeit *O(n)* gelöst werden.

→**TSP kann mit dynamischer Programmierung in Zeit $O(n^2 2^n)$ gelöst werden.**

→Dies ist deutliche Verbesserung gegenüber *O(n!)*.

n	n!	$n^2 2^n$
5	120	800
10	3.628.800	102.400
20	$2{,}4 * 10^{18}$	$3{,}2 * 10^{8}$
50	$2{,}4 * 10^{64}$	$2{,}8 * 10^{18}$
100	$9{,}3 * 10^{157}$	$1{,}3 * 10^{34}$

TSP ist NP-vollständig

- **Beweis über Reduktion**
  - Hamiltonkreis-Problem ist NP-vollständig
  - Hamiltonkreis-Problem kann auf TSP reduziert werden:
    - Setze hierfür Kantengewichte geeignet. Wie?
    - Erweitere G in vollständigen G` : Kantengewichte von G = 1, die anderen z.B. 10
    - TSP(G`) >= n . Falls TSP(G‘)=n, dann sind alle Kanten von TSP sind aus G => es gibt HK(G)

→ Falls *P* <> *NP*, benötigt jeder Algorithmus zur **exakten** Lösung von TSP **exponentielle Laufzeit** in *n.*

Aber es gibt praktikable Algorithmen für **modifizierte Problemstellungen:**

- Spezielle TSP-Probleme (erfüllen zusätzliche Bedingungen)
- Berechne nicht die optimale, sondern *gute* Lösung

Travelling Salesman Problem

- Problemdefinition
- Exakte Lösungsverfahren
- **Heuristiken**
- Näherungslösungen

Heuristiken für Optimierungsprobleme

Heuristik: Verfahren zur Bestimmung *guter zulässiger Lösungen* für reale Entscheidungsprobleme, deren exakte Lösung zu aufwändig ist.

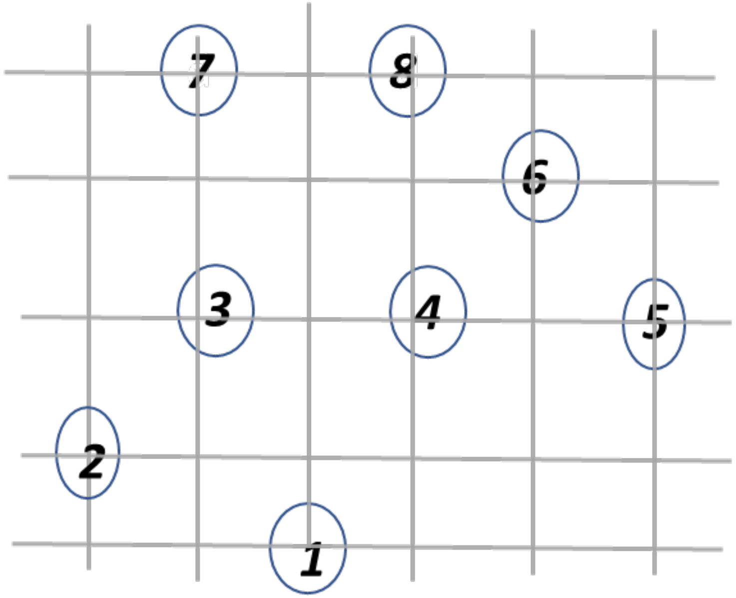
- Baut auf Hypothesen und Erfahrungswerten auf
- Garantiert keine optimale Lösung
- Liefert oft nicht einmal nachweisbar „gute" Lösung
- Kann evtl. fehlschlagen oder nicht terminieren

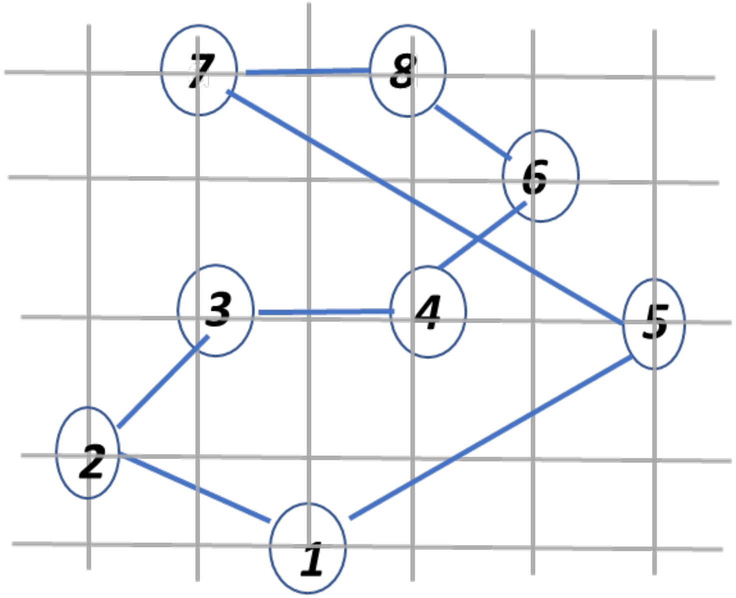
Heuristiken: Typische Ansätze

- *Greedy-Verfahren:* Gierige Lösungen treffen Entscheidungen auf Grund aktueller Kenntnis, werden nicht zurückgenommen
- *Eröffnungsverfahren:* Bestimmen erste zulässige Lösung, die dann weiter optimiert werden kann
- *Lokale Optimierungsverfahren:* Verbessern eine zulässige Lösung inkrementell
- *Unvollständig exakte Verfahren:* Branch-and-bound. Abbrechen, wenn aktuelle Lösung gut genug.
- *Randomisierte/genetische Verfahren:* Randomisierte Mutationen von zulässigen Lösungen untersuchen

Nearest-Neighbor Heuristiken für TSP

Greedy-Ansatz: Gehe immer zum ***nächsten noch nicht besuchten Nachbarknoten*** – abschließend zum Startknoten (Im Beispiel 1)





- Findet sehr schnell eine zulässige Lösung
- Keine garantierte Güte – kann *beliebig schlechter* als Optimum sein

Travelling Salesman Problem

- Problemdefinition
- Exakte Lösungsverfahren
- Heuristiken
- **Näherungslösungen**

Approximationsalgorithmen

Approximationsalgorithmus für ein Optimierungsproblem:

- Findet „schnell" Lösung mit **vorab garantierter Güte**
- Gütegarantie gibt worst case Abschätzung relativ zur optimalen Lösung
- *Beispiel: Gefundene TSP-Tour ist für jeden Input maximal 50% länger als optimale Lösung*

r*-Approximationsalgorithmus:** Zielfunktionswert der gelieferten Lösung ist garantiert höchstens den konstanten Faktor ***r vom Optimum entfernt ($r > 1$).

TSP und Minimal Spannende Bäume

Wenn man aus einem Hamiltonkreis genau ein Kante wegnimmt:

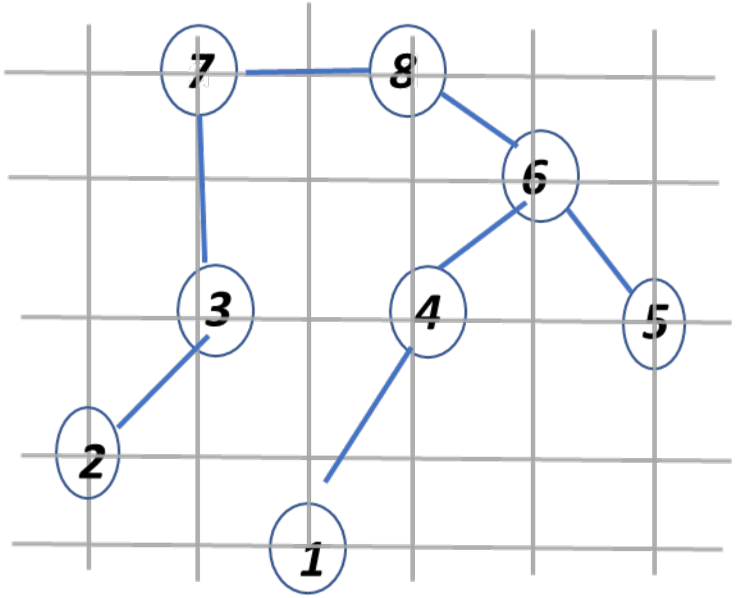
- Wird der Zyklus aufgebrochen
- Es entsteht ein *Spannender Baum* des Graphen! (i.a. kein MST)

→**MST liefert untere Schranke für TSP!**

Länge der optimalen TSP-Tour $\geq$ Gewicht eines MST

Vom MST zu einer TSP-Approximationslösung

Idee: Aus *MST* einen *Hamiltonkreis* konstruieren



MST

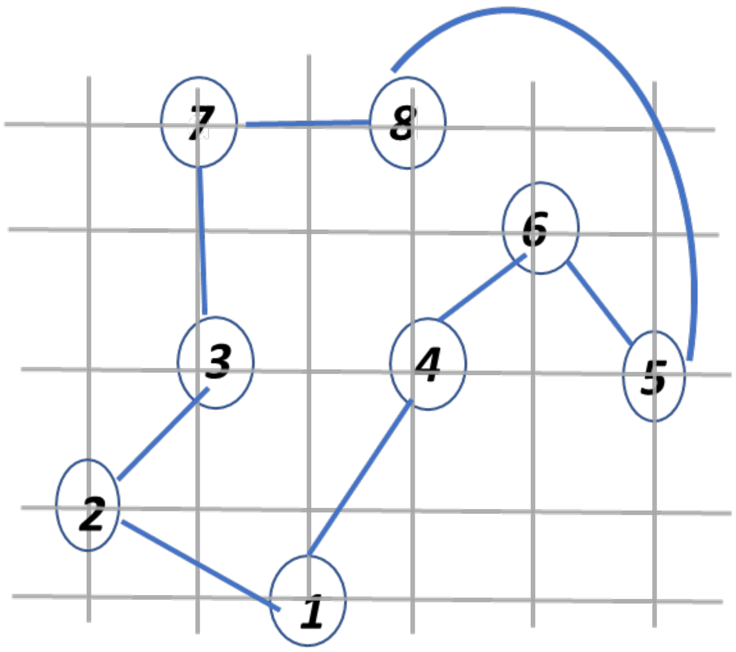
Preorder-Traversierung des MST:

1-4-6-5-6-8-7-3-2-3-7-8-6-4-1

Kein *Hamiltonkreis,* alle Kanten doppelt durchlaufen

Vom MST zu einer TSP-Approximationslösung (2)

2. Idee: *Überspringe schon durchlaufene Knoten!*



MST

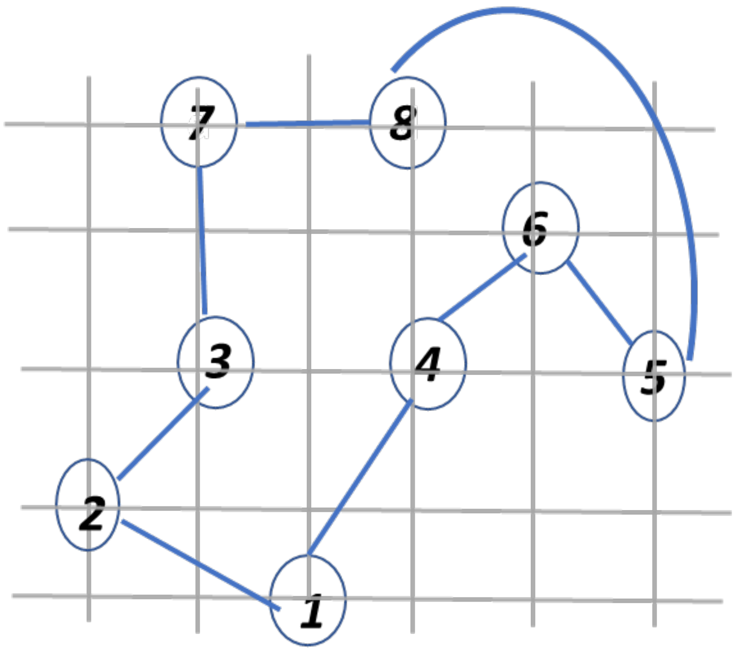
Preorder-Traversierung des MST:

1-4-6-5-8-7-3-2-1 → Hat *n* Kanten

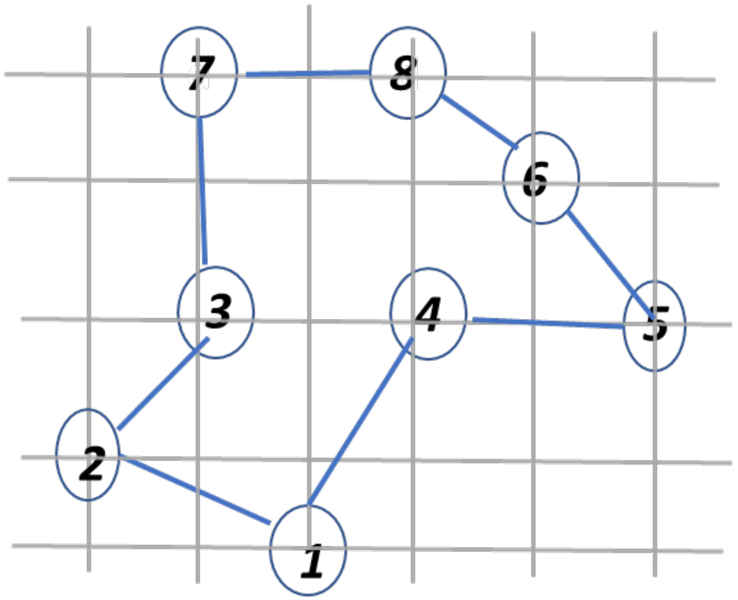
- Zulässige Lösung
- Kann Tour durch Überspringen länger werden?
- **Nein, wenn die Dreiecksungleichung gilt**
  (Von a nach b nicht länger sein als a-> c->b)

Wie gut ist die Lösung?

- Garantie: 2-Approximation, in diesem Beispiel nur 6% über Optimum



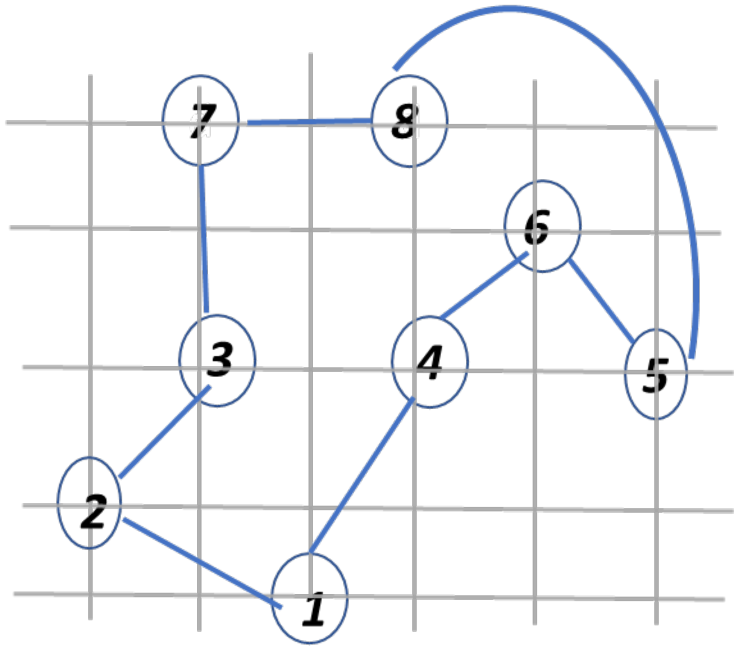
MST: 15.54



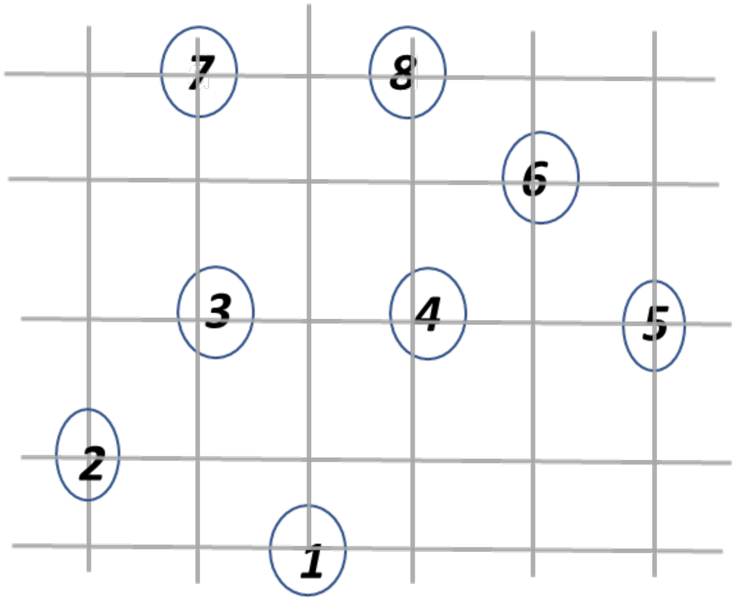
TSP-Optimum: 14.71

Übung: Gleiche Konstruktion, anderer MST

- Anderen MST und anderen Startknoten verwenden



MST (alt) mit Start 1



MST (neu) mit Start x

Approximationsalgorithmen für metrisches TSP

- MST-Konstruktion: *2*-Approximation in $O(n^2)$
- Algorithmus von Christofides (1976): *1,5*-Approximation in $O(n^3)$
- Es geht noch genauer, wenn man mehr Rechenzeit spendiert
- Aber nicht beliebig genau:

„Falls P ungleich NP, kann das metrische TSP in Polynonomialzeit ***nicht genauer als 1,00456 approximiert werden“***
(Papadimitrou/Vempala 2003)

Zusammenfassung Komplexitätstheorie

- Berechenbarkeitsmodelle auf Basis von Turing- und Registermaschinen ➔ Churchsche These
- Komplexitätsklassen $\mathcal{P}$ und $\mathcal{NP}$
- Reduktion von Problemen
- $\mathcal{NP}$-harte und $\mathcal{NP}$-vollständige Probleme
- Travelling Salesman Problem
- Exakte Lösung mit Dynamischer Programmierung
- Heuristiken und Approximationslösungen