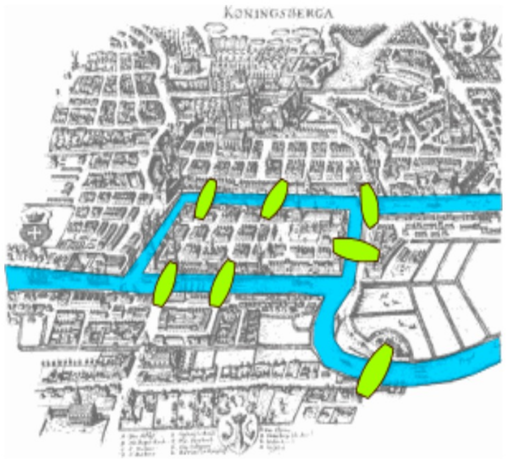
13. Einfache und schwere Graphen-Probleme

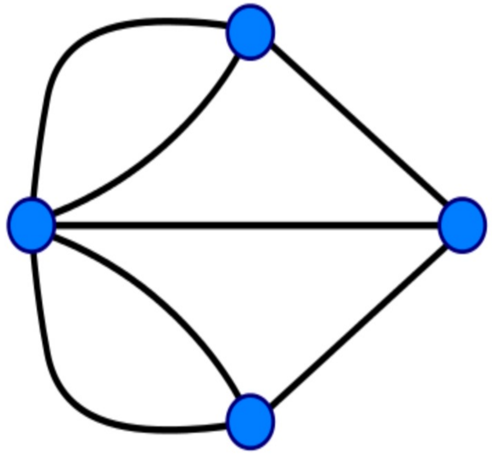
- Die Altstadt von Venedig: 118 Inseln, verbunden durch ca. 400 Brücken.
- **Hamilton-Tour:**
  Finde Rundtour, so dass <u>jede Insel genau einmal besucht</u> wird.
- **Euler-Tour:**
  Finde Rundtour, so dass <u>jede Brücke genau einmal überquert</u> wird.
- Was ist schwieriger?

13. Einfache und schwere Graphen-Probleme

- Euler- und Hamilton-Kreise
- Entscheidungs- und Suchprobleme
- Einfache und schwere Probleme:
  P, NP, NP-Vollständigkeit
- P-NP-Problem
- Beispiele für NP-vollständige
  Graphen-Probleme

Königsberger Brückenproblem

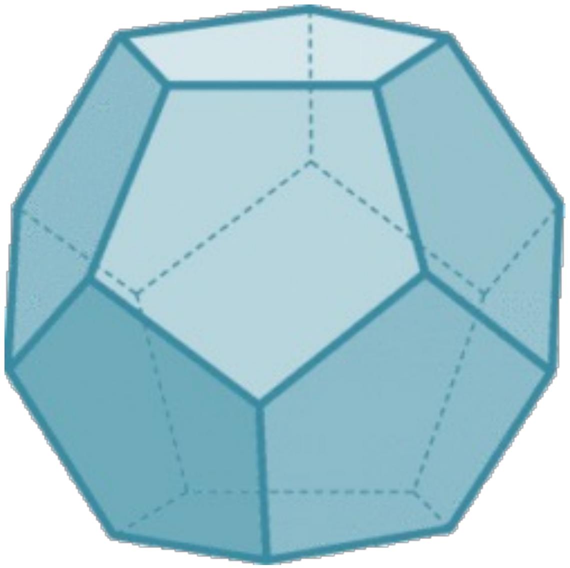


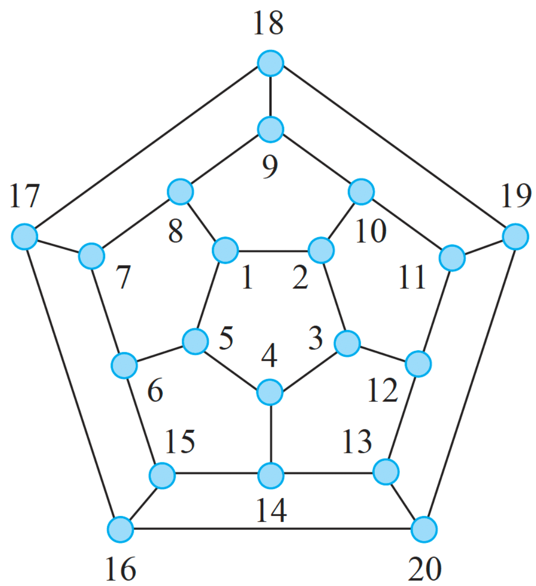


https://de.wikipedia.org/wiki/Königsberger_Brückenproblem

- Beliebte Frage:
  gibt es einen Rundweg durch Königsberg,
  so dass jede Brücke genau einmal überquert wird?
- Leonard Euler bewies 1736, dass es für Königsberg keine Lösung gibt.
- Begründete damit die Graphentheorie.

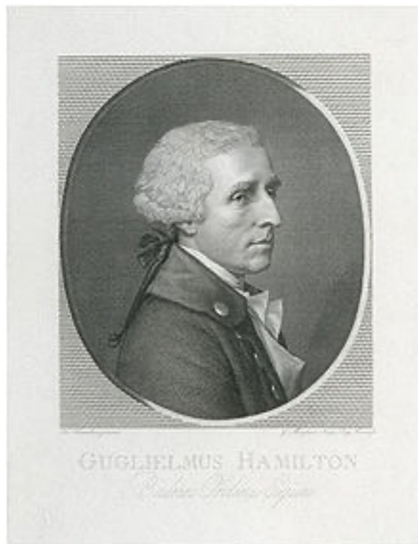
„Reise um die Welt“ von Hamilton





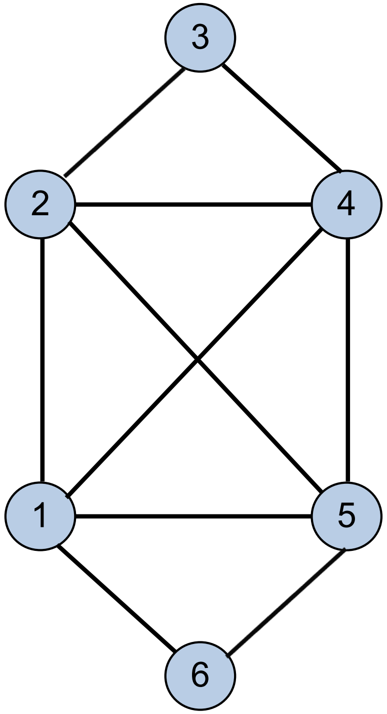
Hoffmann, Theoretische Informatik, Hanser-Verlag 2015

- Sir William Hamilton, berühmter irischer Mathematiker, erfand 1857 das Knobelspiel „Reise um die Welt“.
- Das Spiel besteht aus einem Dodekaeder, dessen 20 Ecken mit Städtenamen beschriftet sind.
- Ziel ist es, eine Rundtour zu finden, so dass jede Stadt genau einmal besucht wird.



Euler- und Hamilton-Kreis

- Sei G = (V, E) ein ungerichteter Graph.
- Ein Euler-Kreis in G ist ein Zyklus, bei dem jede Kante aus E genau einmal besucht wird. Knoten dürfen dabei mehrfach besucht werden.
- Ein Hamilton-Kreis in G ist ein Zyklus, bei dem jeder Knoten $v \in V$ genau einmal besucht wird.
  (Genauer: ein Knoten v gilt als besucht, wenn eine Kante (u,v) besucht wird.)



- Euler-Kreis („Haus des Nikolaus“ + 6 + 1):
  1 – 2 – 3 – 4 – 2 – 5 – 1 – 4 – 5 – 6 – 1
- Hamilton-Kreis:
  1 – 2 – 3 – 4 – 5 – 6 – 1

Welches Problem ist „schwerer“?

Entscheidungs- versus Suchproblem

Entscheidungsproblem
Hat eine Eingabe x eine bestimmte Eigenschaft A? Ja oder nein? A lässt sich auch als Menge auffassen: $x \in A$ gdw. x hat die Eigenschaft A.

Suchproblem
Gegeben ist eine Eingabe x. Suche ein y mit einer bestimmten Eigenschaft.

Entscheidungsproblem	Suchproblem
Hat Graph G einen Euler-Kreis?	Suche im Graph G einen Euler-Kreis.
Hat Graph G einen Hamilton-Kreis?	Suche im Graph G einen Hamilton-Kreis.
Hat Graph G einen Weg von Knoten S nach Knoten Z mit Länge $\le k$	Suche im Graph G einen kürzesten Weg von Knoten S nach Knoten Z.
Hat gewichteter Graph G einen Hamilton-Kreis mit Länge $\le k$	Suche im gewichteten Graph G einen Hamilton-Kreis mit kürzester Länge.

Bemerkungen

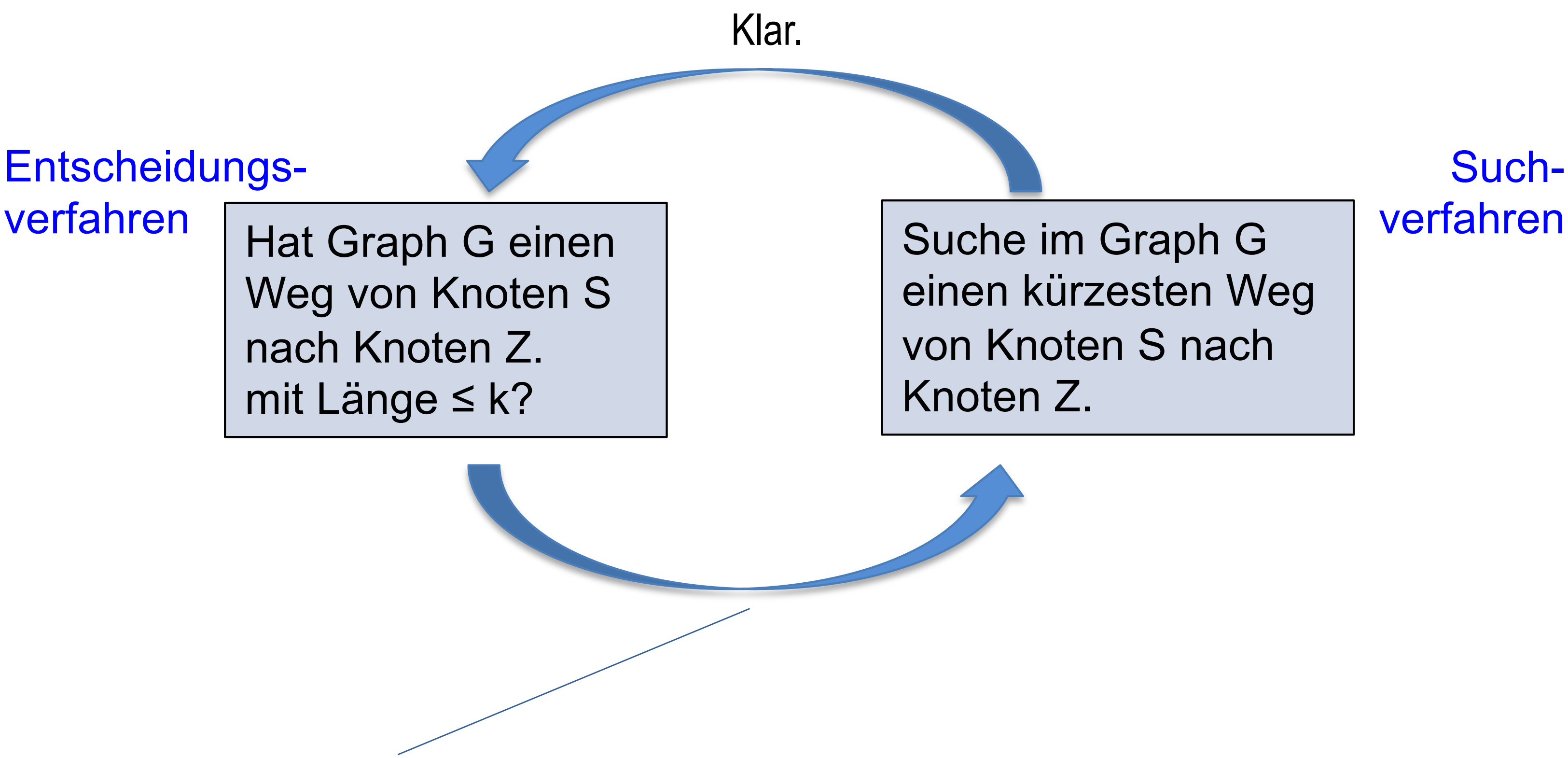
(1) Die Komplexitätsklassen P und NP (kommt gleich) werden aus technischen Gründen für Entscheidungsprobleme eingeführt.

(2) Unterschied zwischen Entscheidungs- und Suchprobleme ist bzgl. der Komplexität meistens unerheblich.

(3) Suchverfahren kann fast direkt als Entscheidungsverfahren benutzt werden.
(Bsp.: Nächste Folie)

(4) Umgekehrt kann aus Entscheidungsverfahren meistens ohne erheblichen Mehraufwand (d.h. polynomieller Mehraufwand) ein Suchverfahren gewonnen werden.
(Bsp.: nächste Folie)

Beispiel zu Bemerkung (3) und (4)



(1) Suche Länge eines kürzesten Wegs k_{Min} von Knoten S nach Knoten Z.
(Problem für verschiedene k mit binärer Suche lösen).

(2) Prüfe mit Entscheidungsverfahren für alle Kanten e im Graph G:

– Ist Weg ≤ k_{Min} möglich, falls e weggelassen wird.
– Falls ja, dann wird e nicht benötigt und kann endgültig entfernt werden.

Klasse P

Definition Klasse P
Die Klasse P ist die Menge aller Entscheidungsprobleme, die mit einem Algorithmus mit polynomieller Laufzeit gelöst werden können. Polynomielle Laufzeit: $O(n^k)$ für ein festes k. Also: $O(1)$, $O(n)$, $O(n^2)$, $O(n^3)$, ...,

Beispiele	Komplexität
Ist eine Zahlenfolge mit n Elementen Sortierung einer anderen Folge?	$O(n \log n)$
Kommt x in einem sortieren Feld mit n Elementen vor?	$O(\log n)$
Gibt es in einem gewichteten Graphen (positive Gewichte) mit n Knoten und $O(n)$ Kanten einen Weg mit Länge $\leq k$	$O(n \log n)$
Gibt es in einem gewichteten Graphen mit n Knoten und $O(n)$ Kanten einen aufspannenden Baum mit Gesamtkosten $\leq k$	$O(n \log n)$
...	

Einfache und schwere Probleme

- Alle Probleme aus P sind polynomiell lösbar und werden daher als effizient lösbar (d.h. in praktikabler Zeit lösbar) betrachtet.
- „Praktikabel“ wird dabei sehr großzügig ausgelegt: $O(n^{100})$ wird ebenso wie $O(n^2)$ als praktikabel eingestuft.
- Alle Probleme, die nur mit einem (über)exponentiellen Aufwand (z.B. $O(2^n)$, $O(n!)$, ...) gelöst werden können, gelten als nicht effizient lösbar (d.h. nicht in praktikabler Zeit lösbar).

Laufzeit	**10**	**100**	**1000**	**10^6**
n	0.01 µsec	0.1 µsec	1 µsec	1 msec
$n \log_{10} n$	0.01 µsec	0.2 µsec	3 µsec	6 msec
n^2	0.1 µsec	10 µsec	1 msec	16.7 min
n^3	1 µsec	1 msec	1 sec	31.7 Jahre
2^n	1 µsec	$4*10^{13}$ Jahre	...	...
$n!$	3.6 msec	$2.9*10^{141}$ Jahre	...	...

P = einfache Probleme

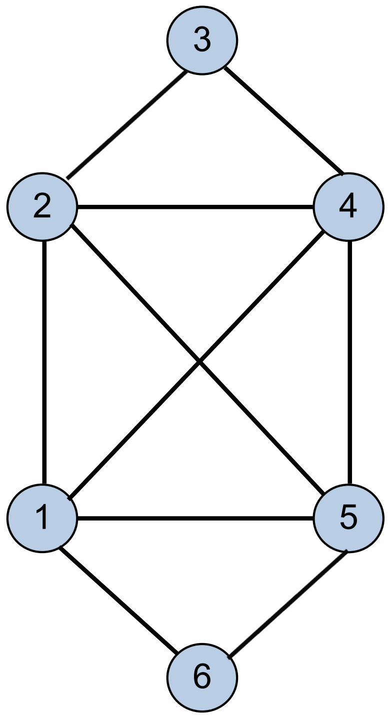
Schwere Probleme

Annahme: Rechner führt 10^9 elementare Operationen je Sekunde durch.
Geschätztes Alter des Weltalls: $18.8*10^9$ Jahre.

Euler-Kreis-Problem ist in P

Satz von Euler (1736)

- Ein zusammenhängender Graph (Knotenanzahl > 1) besitzt genau dann einen Euler-Kreis, wenn der Grad jedes Knotens (= Anzahl der Nachbarn) gerade ist.
- Es können auch Mehrfachkanten zugelassen werden. Dann muss jedoch der Grad eines Knotens v als die Anzahl der Kanten, die bei v enden, definiert werden.



- Jeder Knoten im Graph hat eine gerade Anzahl an Nachbarn.
- Daher muss es einen Euler-Kreis geben.

Finde Euler-Kreis ist effizient lösbar

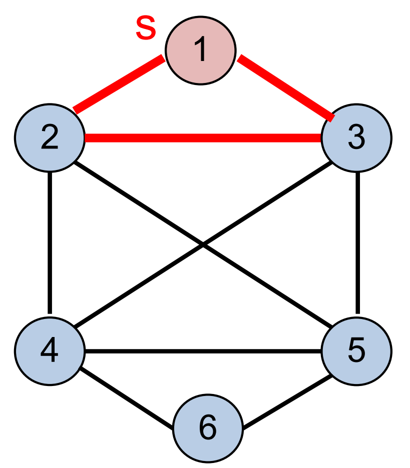
Algorithmus von Hierholzer
Graph G muss zusammenhängend und alle Knotengrade müssen gerade sein.

(1) Setze s auf irgendeinen Knoten aus G (z.B. mit kleinster Nummer).

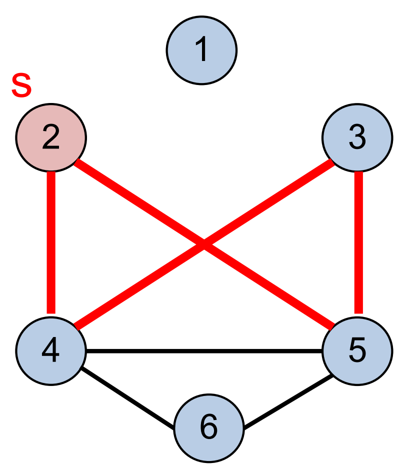
(2) Starte bei s und besuche irgendeinen Nachbarn (z.B. mit kleinster Nummer) über eine noch nicht besuchte Kante e; entferne besuchte Kante e aus Graph G und füge e zu Eulerkreis dazu; fahre fort, solange ein Nachbarknoten besucht werden kann.

(3) Falls alle Kanten besucht, dann breche ab. Eulerkreis wurde gefunden.

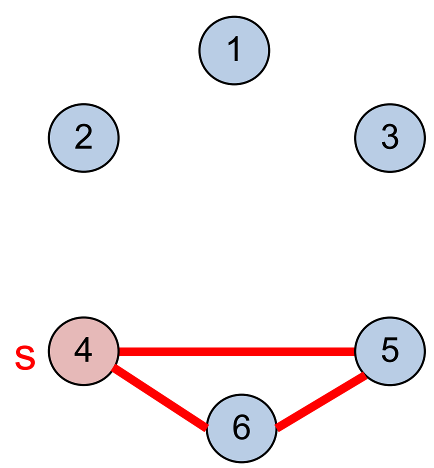
(4) Ansonsten suche auf bisherigen Eulerkreis einen Knoten s‘ (z.B. mit kleinster Nummer), von dem eine nicht besuchte Kante ausgeht;
schneide bisherige Eulerkreis bei s‘ durch und setze Suche bei (2) mit s = s‘ weiter.



Eulerkreis: 1-2-3-1

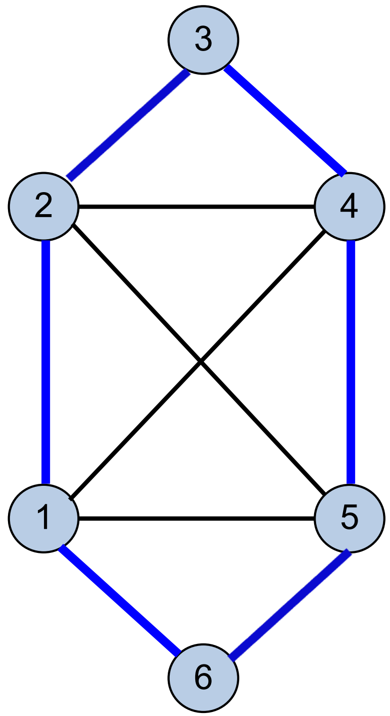


1-2-4-3-5-2-3-1



1-2-4-5-6-4-3-5-2-3-1

Ist Hamilton-Kreis-Problem effizient lösbar?



- Hamilton-Kreis =
  Zyklus, bei dem jeder Knoten genau einmal besucht wird.

Ist die Entscheidung, ob ein Graph ein Hamilton-Kreis hat, effizient lösbar?

Vermutlich nein!

Klasse NP

Definition Klasse NP

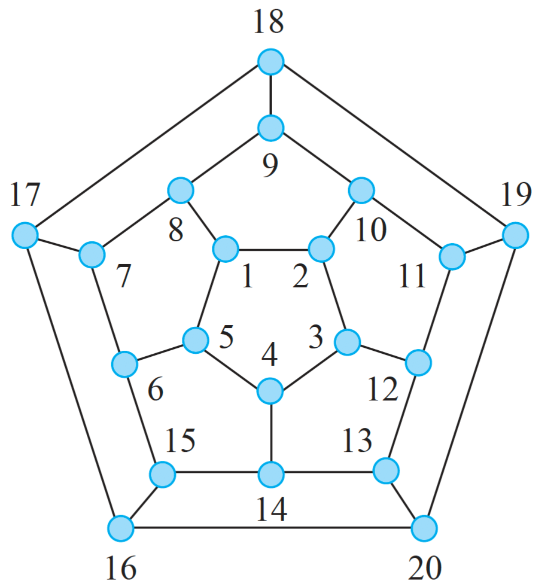
Die Klasse NP ist die Menge aller Entscheidungsprobleme, bei denen in polynomieller Zeit überprüfbar ist, ob eine vorgegebene (geratene) Lösung das Problem löst.

Bemerkungen

- Übliche Definition von NP:
  NP ist die Menge aller Entscheidungsprobleme, die mit einem nicht-deterministischen Algorithmus (Turingmaschine) in polynomieller Zeit lösbar sind.
- Ein nicht-deterministischer Algorithmus darf in einem Rechenschritt aus mehreren Möglichkeiten auswählen (raten), wobei der Algorithmus immer so wählt, dass es zu einer positiven Entscheidung führt, falls es eine gibt.
- Hier soll jedoch das schwierigere Konzept des Nicht-Determinismus vermieden werden.

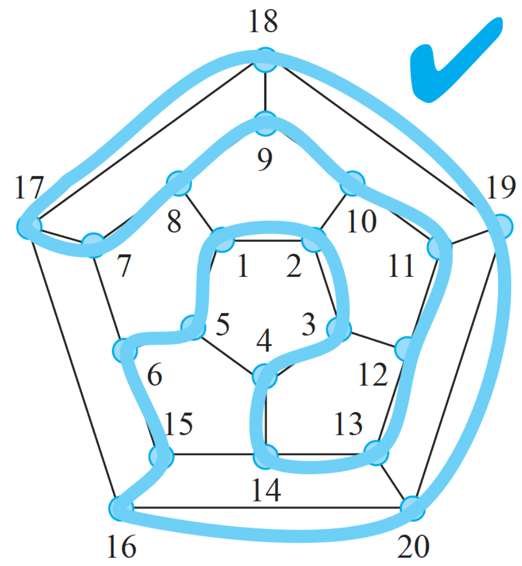
Hamilton-Kreis-Problem ist in NP

Eingabe: Dodekaeder-Graph



Rate Lösungsfolge

1 - 2 - 3 - 4 - 14 - 13 - 12 - 11 - 10 - 9 - 8 - 7 - 17 - 18 - 19 - 20 - 16 - 15 - 6 - 5 - 1

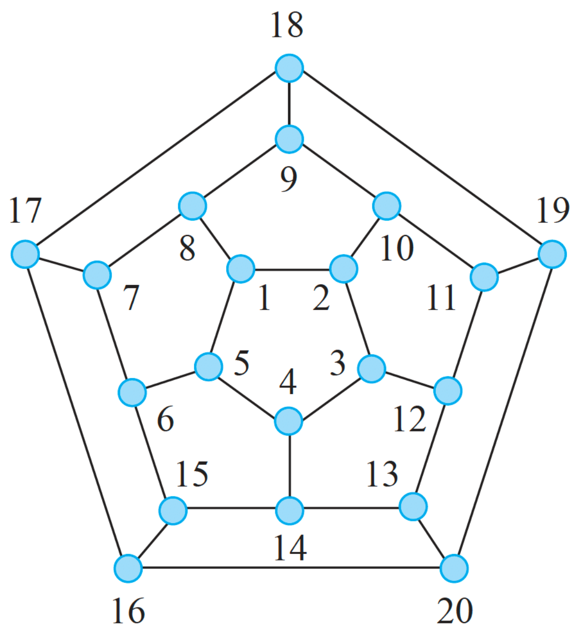


Prüfe in polynomieller Zeit, dass geratene Lösungsfolge ein Hamiltonkreis ist.

Bilder aus *Hoffmann, Theoretische Informatik, Hanser-Verlag 2015*

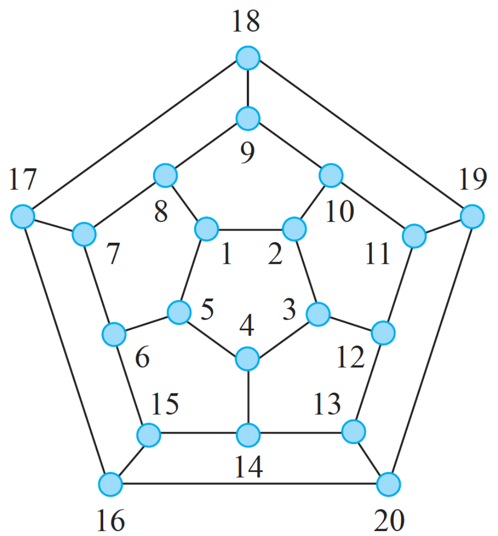
Brute-Force-Methode für Probleme aus NP

- Die Definition für NP sieht etwas merkwürdig aus.
- Die Definition hat jedoch den Vorteil, dass sehr bequem feststellbar ist, ob ein Problem aus NP ist.
- Möchte man einen NP-Algorithmus als deterministisches Verfahren implementieren, dann muss der Rateschritt einer Lösung nachgebaut werden, indem alle Lösungsmöglichkeiten systematisch ausprobiert werden. Brute-Force-Methode! Führt zu einem überexponentiellem Aufwand!



- (n-1)!/2 viele Knoten-Permutationen.
  (Startknoten fest. Zyklen, die sich durch Richtungsumkehr ergeben, werden ignoriert.)
- Also: Aufwand von O(n!)
- Bei $n = 20$:
  - $19!/2 = 6*10^{16}$ viele Permutationen
    (Startknoten fest. Invertierte Kreise werden weggelassen)
  - bei Prüfung von 10^9 Permutationen je sec:
    1.9 Jahre Laufzeit

Ist Hamilton-Kreis-Problem effizient lösbar?



- Brute-Force-Methode führt zu einem Aufwand von O(n!).
- Nicht praktikabel:
  1.9 Jahre Laufzeit bei n = 20 Knoten und Prüfung von 10^9 Permutationen je Sek.

Gibt es einen polynomiellen Algorithmus?

Vermutlich nein!

Hamilton-Kreis-Problem ist nämlich NP-vollständig!

Polynomielle Reduktion

Definition Polynomielle Reduzierbarkeit

Ein Entscheidungsproblem A ist auf ein Entscheidungsproblem B polynomiell reduzierbar, falls es ein polynomielles Verfahren (Reduktion) f gibt, so das

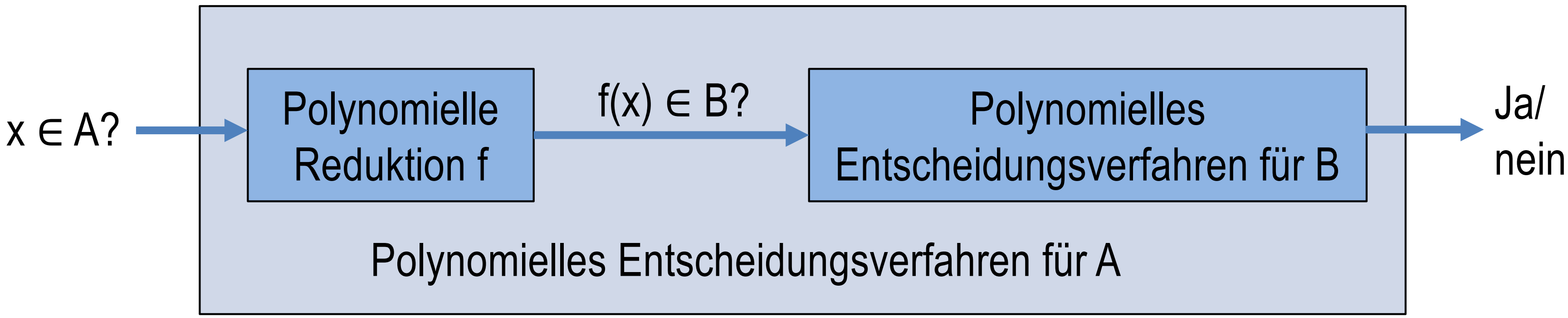
$x \in A$ gdw. $f(x) \in B$

Schreibweise:

$A \leq_p B$

Intuitiv: A ist nicht (wesentlich) schwerer als B.

Wenn nämlich B in polynomieller Zeit gelöst werden kann, dann kann auch A in polynomieller Zeit gelöst werden.



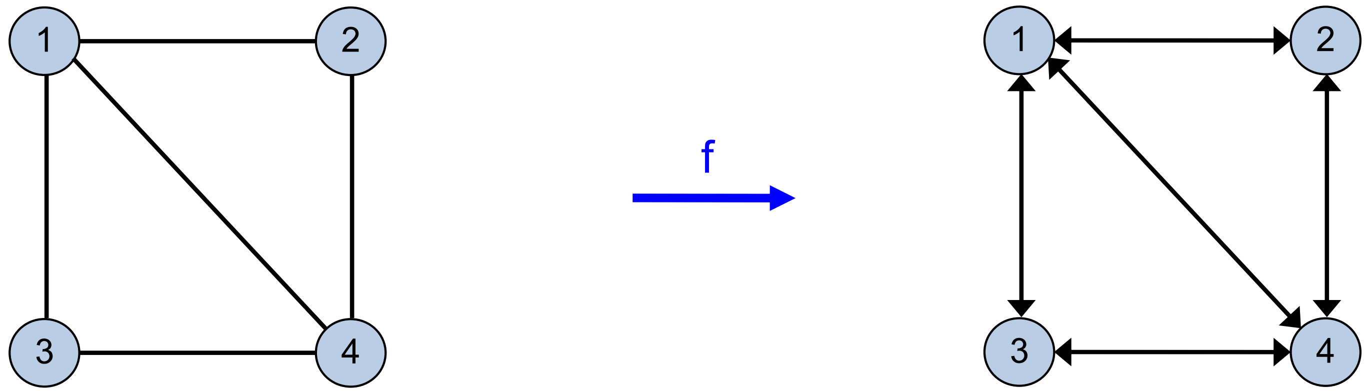
Beispiel: HC $\le_p$ DHC

- Ziel: Definiere eine Funktion (Reduktion) f, die einen ungerichteten Graphen G in einen gerichteten Graphen f(G) transformiert, so dass

  G hat einen Hamilton-Kreis (HC) gdw.

  f(G) hat einen Hamilton-Kreis mit gerichteten Kanten (DHC)

- Naheliegende Idee:
  f ersetzt jede ungerichtete Kante durch zwei gerichtete Kanten.



ungerichteter Graph G

gerichteter Graph f(G)
(Doppelpfeil steht für zwei gerichtete Kanten)

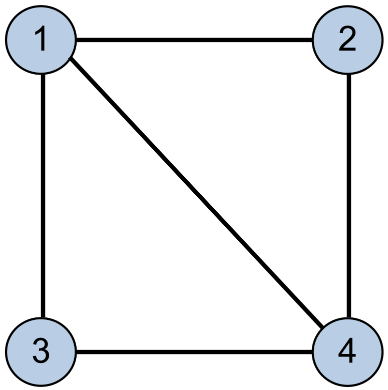
Beispiel: HC $\le_p$ DHC

Es gilt:

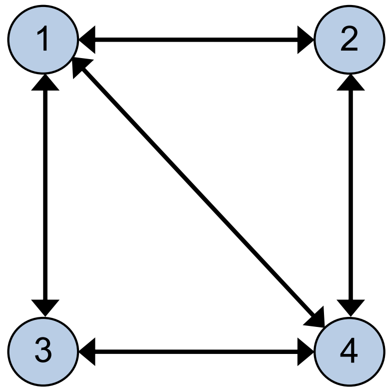
(1) f hat polynomiellen Aufwand

(2) G hat einen Hamilton-Kreis (HC) gdw.
f(G) hat einen Hamilton-Kreis mit gerichteten Kanten (DHC)

(3) Damit: HC $\le_p$ DHC



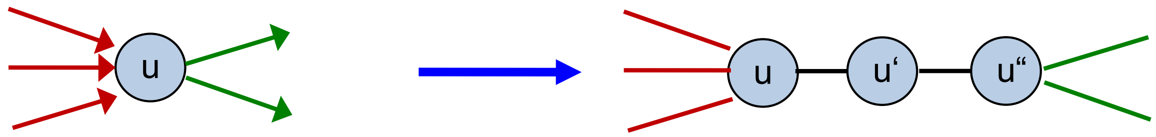
ungerichteter Graph G mit
Hamilton-Kreis 1, 2, 4, 3, 1.

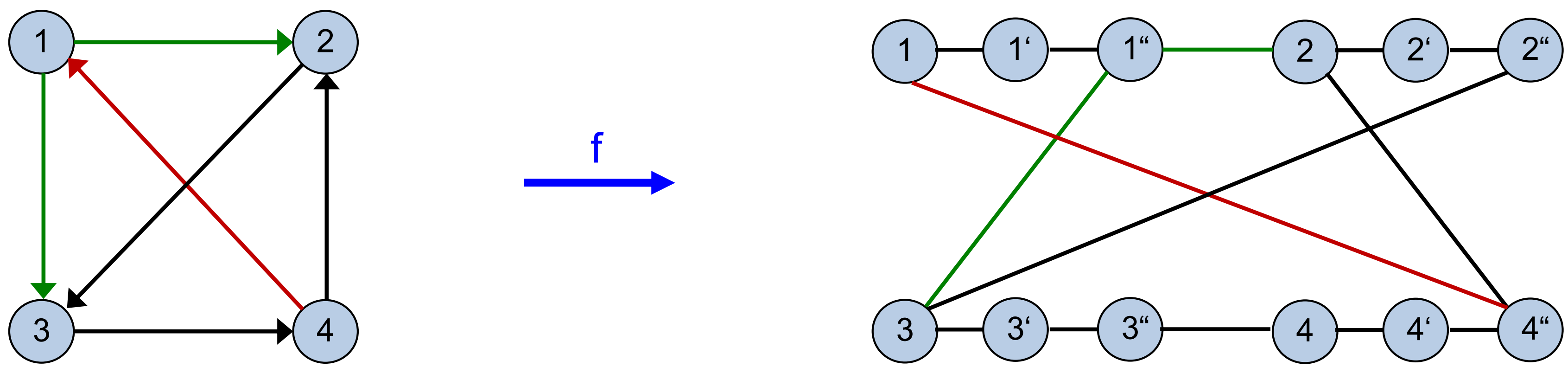


gerichteter Graph f(G) mit
(gerichteten) Hamilton-Kreis
1, 2, 4, 3, 1.

Beispiel: DHC $\le_p$ HC

- Reduktion f erzeugt aus einem gerichteten Graphen G einen ungerichteten Graphen f(G), indem jeder Knoten u mit seinen ein- und ausgehenden Kanten durch 3 Knoten u, u‘, u“ mit ungerichteten Kanten ersetzt wird.





Gerichteter Graph G

ungerichteter Graph f(G)

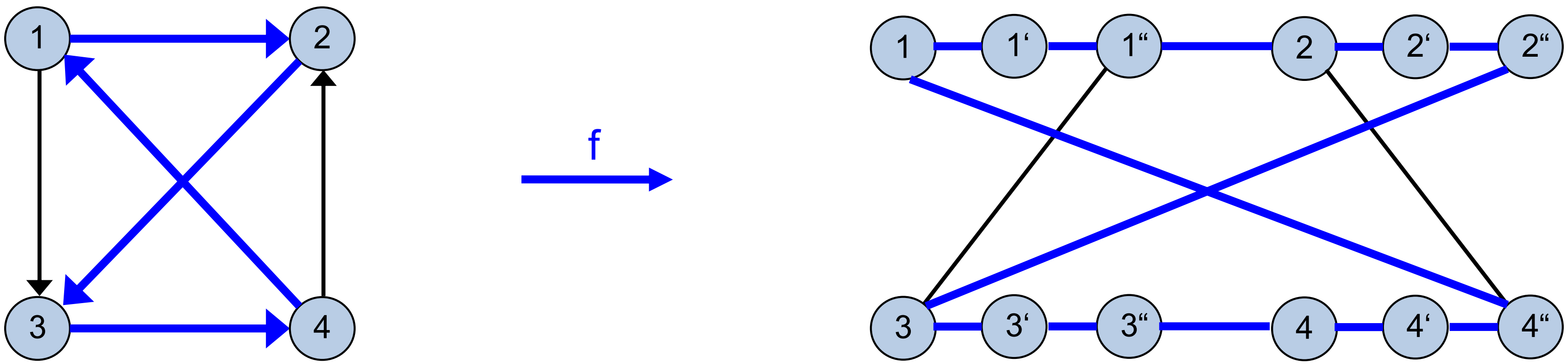
Beispiel: DHC $\leq_p$ HC

Es gilt:

(1) f hat polynomiellen Aufwand

(2) Gerichteter Graph G hat einen Hamilton-Kreis (DHC) gdw.
f(G) einen Hamilton-Kreis hat (HC).

(3) Damit: DHC $\leq_p$ HC



Gerichteter Graph G
mit Hamilton-Kreis 1, 2, 3, 4, 1

ungerichteter Graph f(G)
mit Hamilton-Kreis
1, 1‘, 1“, 2, 2‘, 2“, 3, ..., 1

NP-vollständige Probleme

Definition NP-Vollständigkeit (NP Completeness)

Ein Entscheidungsproblem E ist NP-vollständig, falls

(1) E ist NP-schwer, d.h. für alle $A \in NP$ ist $A \leq_p E$

(2) $E \in NP$

Intuitiv: NP-vollständige Probleme sind die schwersten Probleme in NP.

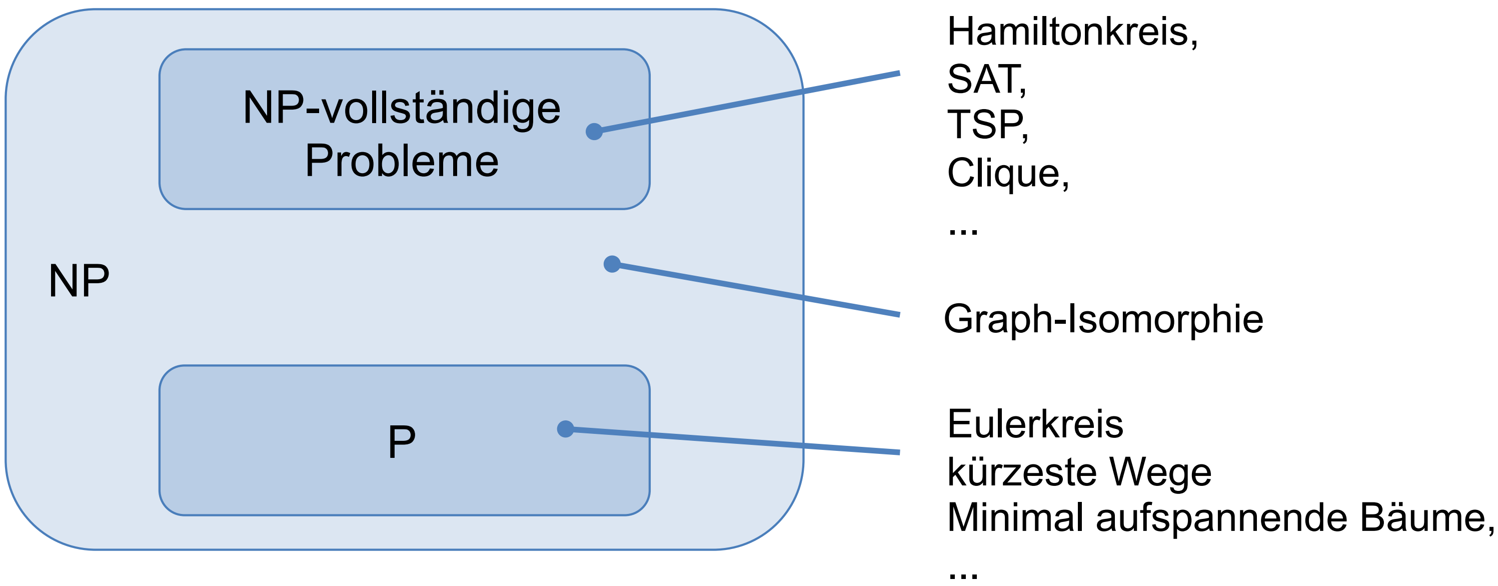
Lässt sich nämlich ein NP-vollständiges Problem effizient lösen, (d.h. in polynomieller Zeit), dann sind alle Probleme in NP effizient lösbar.

- Hamilton-Kreis-Problem ist NP-vollständig.
- Es gibt hunderte Probleme, die von praktischer Relevanz und NP-vollständig sind.
- Vermutlich ist kein NP-vollständiges Problem effizient lösbar.

P = NP?

- Offensichtlich gilt: $P \subseteq NP$
- Die Frage, ob $P \subset NP$ gilt, ist eines der wichtigsten ungelösten Probleme in der Informatik.
- Sehr viele Indizien sprechen für $P \subset NP$.
- Das Clay Mathematics Institute hat das P-NP-Problem in die Liste der Millennium-Probleme aufgenommen und mit einem Preisgeld von einer Million US-Dollar ausgelobt.

Vermutete Situation



10. Einfache und schwere Graphen-Probleme

- Euler- und Hamiltonkreise
- Entscheidungs- und Suchprobleme
- Einfache und schwere Probleme:
  P, NP, NP-Vollständigkeit
- P-NP-Problem
- Beispiele für NP-vollständige
  Graphen-Probleme

Problem des Handlungsreisenden (TSP, Travelling Salesman Problem)

- Gegeben ist ein vollständiger ungerichteter Graph mit positiven Gewichten
- Suchproblem: finde eine Tour (Hamilton-Kreis) mit minimalen Gesamtkosten (d.h. kürzeste Pfadlänge).
- Entscheidungsproblem: Gibt es eine Tour, die ein bestimmtes Kostenlimit K nicht überschreitet.

- Kürzeste Rundreise für die 15 größten Städte in Deutschland.
- Die angegebene Route ist die kürzeste von $14!/2 = 43{,}589{,}145{,}600$ möglichen Routen.
- https://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden

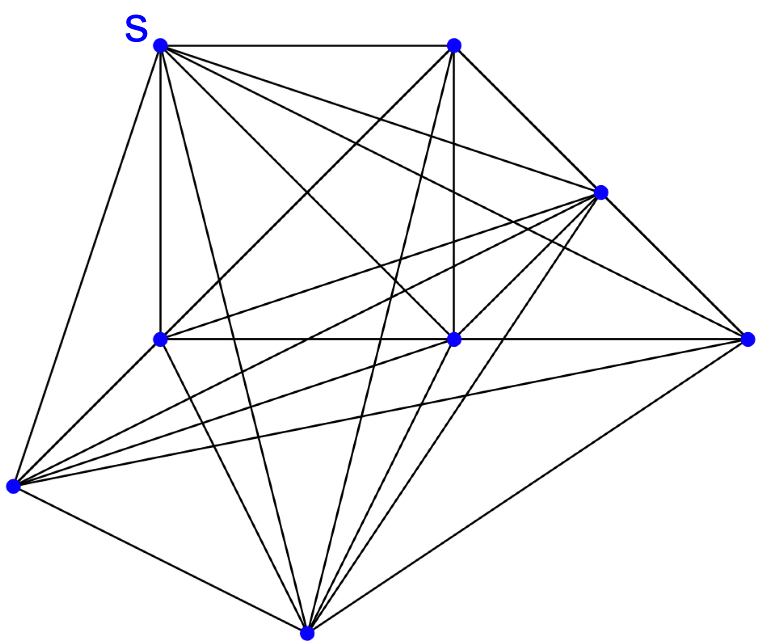
Brute-Force-Methode (nur für kleine Graphen praktikabel!)

```
V s;                          // Startknoten
List<V> kürzesteTour;    // bisher kürzeste Tour

void solveTSP(V v) {
    kürzesteTour = undef;
    s = v;
    solveTSP(v, empty);
    print(kürzesteTour);
}

void solveTSP(V v, List<V> weg) {
    weg.add(v);
    if (weg enthält alle Knoten)
        aktualisiere kürzesteTour, falls aktueller Weg weg
        mit Kante zurück nach Startknoten s kürzer ist;
    else
        for (jeden Knoten w)
            if (! weg.contains(w))
                solveTSP(w, weg);
    weg.remove(v);
}
```

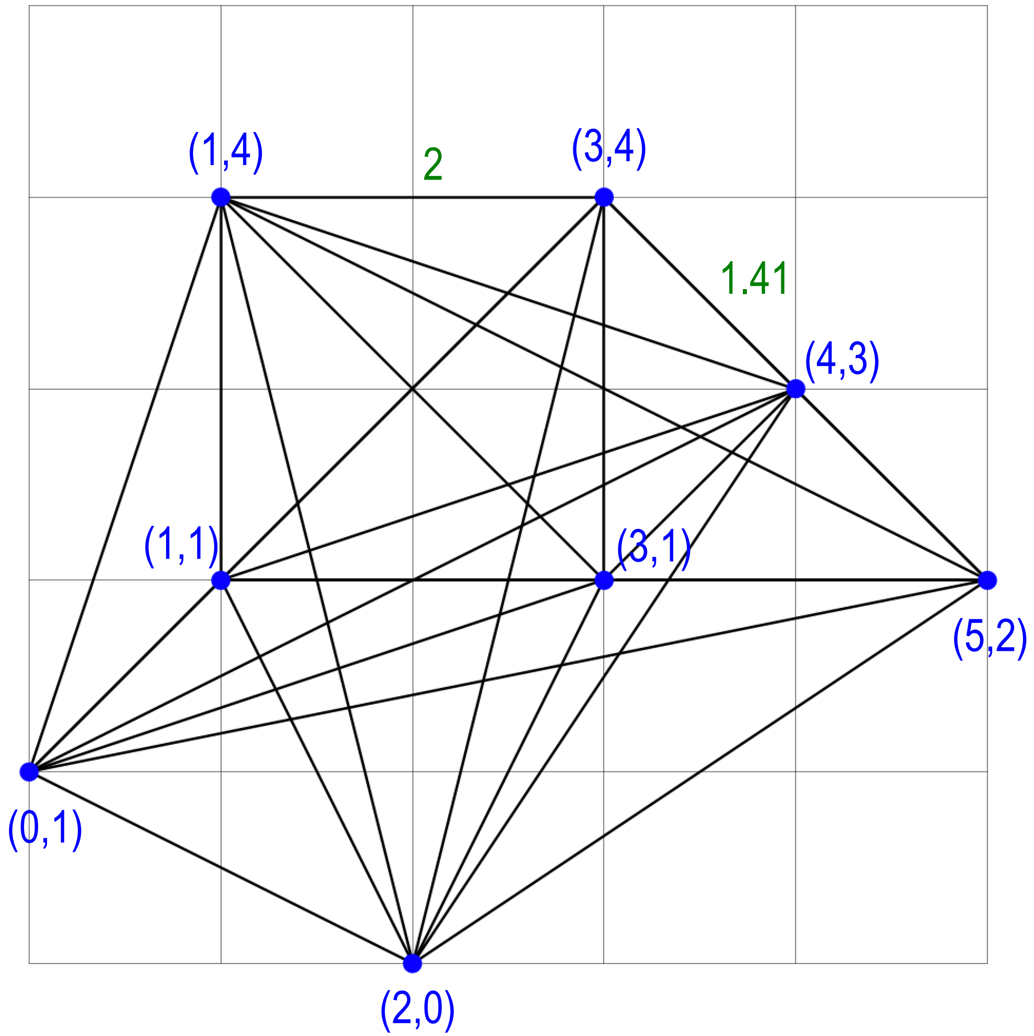
erweitert aktuellen Weg weg um Knoten v



Vollständiger Graph mit 8 Knoten.

Führt bei festem Startknoten s auf 7! = 5040 verschiedene Touren.

Euklidische Graphen und Dreiecksungleichung



- Ein Graph heißt Euklidisch, falls die Knoten durch Punkte in der Ebene repräsentiert werden.
- Euklidische Graphen sind vollständig und Kanten haben als Gewicht (Kosten) den Euklidischen Abstand.
- Auch Manhattan-Distanz ist möglich.
- Euklidische Graphen erfüllen die sogenannte Dreiecksungleichung:

  $$c(u,w) \leq c(u,v) + c(v,w)$$

- Für Graphen, die die Dreiecksungleichung erfüllen, gibt es Algorithmen, die TSP suboptimal lösen
  → approximative Algorithmen.

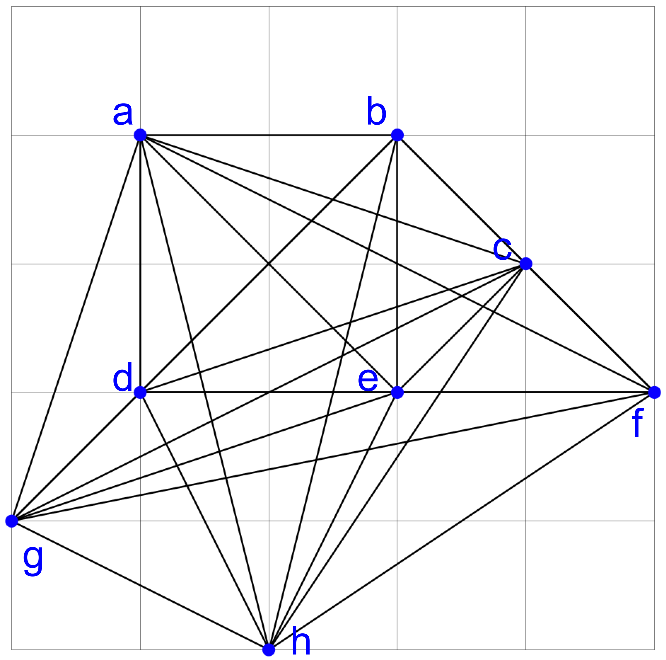
Einfacher approximativer TSP-Algorithmus für Euklidische Graphen

(1) Ermittle für den Euklidischen Graph einen minimal aufspannenden Baum (z.B. mit Kruskal-Algorithmus).

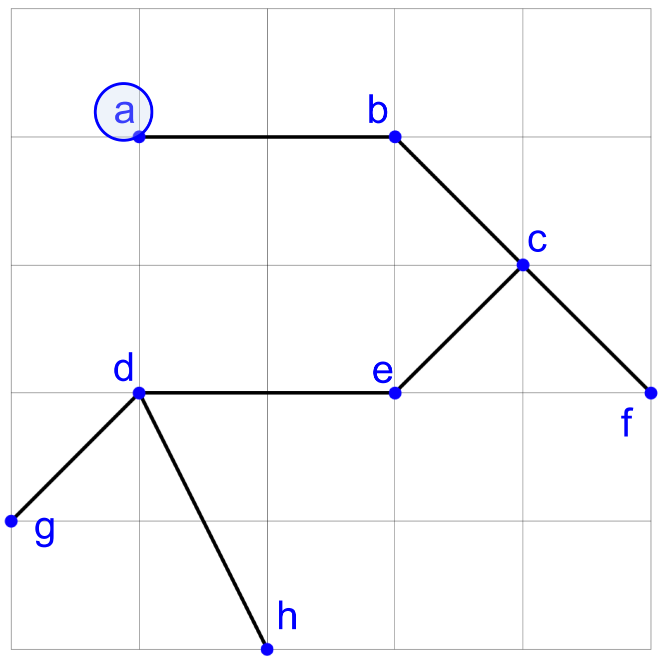
(2) Führe im Baum eine Tiefensuche durch mit Start bei einem beliebigen Knoten s. Ermittle dabei die Pre-Order-Reihenfolge p.

(3) Gesuchte (suboptimale) Tour ist p als Weg erweitert um Rückkehr zu Knoten s.

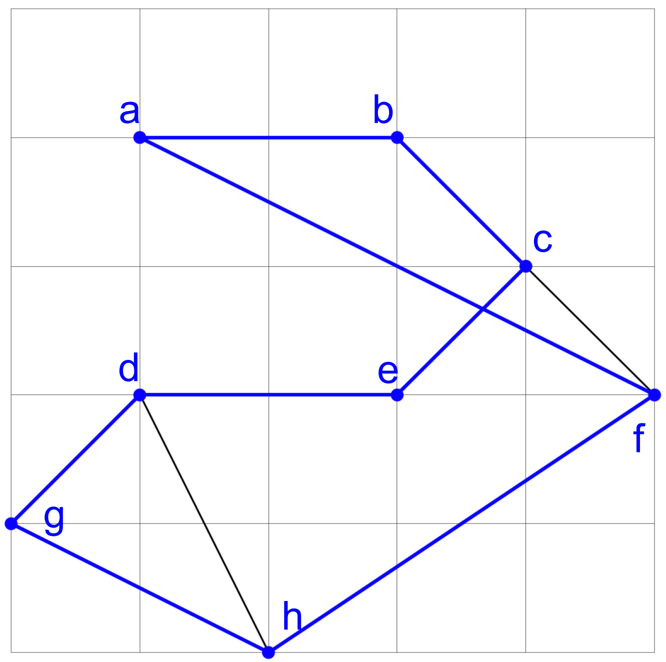
Euklidischer Graph



Minimal aufspannender Baum.
Pre-Order-Reihenfolge (Start mit a):
a, b, c, e, d, g, h, f

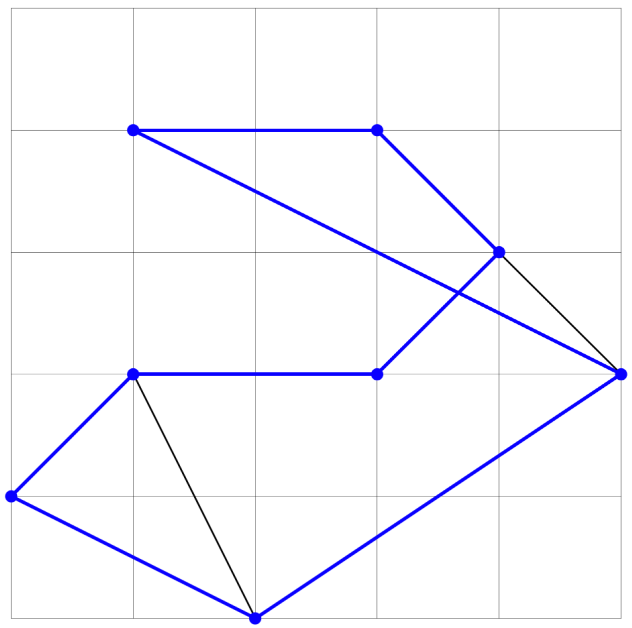


Gesuchte Tour:
a, b, c, e, d, g, h, f, a

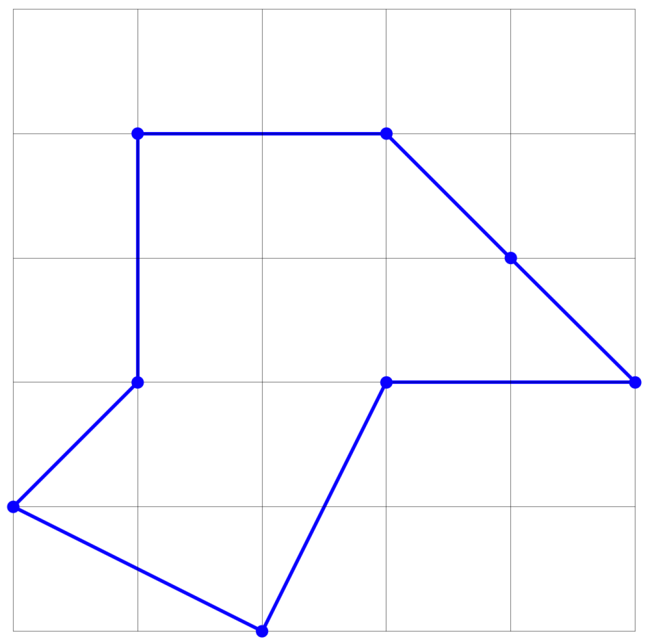


Einfacher approximativer TSP-Algorithmus für Euklidische Graphen

- Die durch den einfachen approximativen Algorithmus gefundene Touren sind maximal 2-mal so lang wie die Gesamtlänge von optimalen Touren.



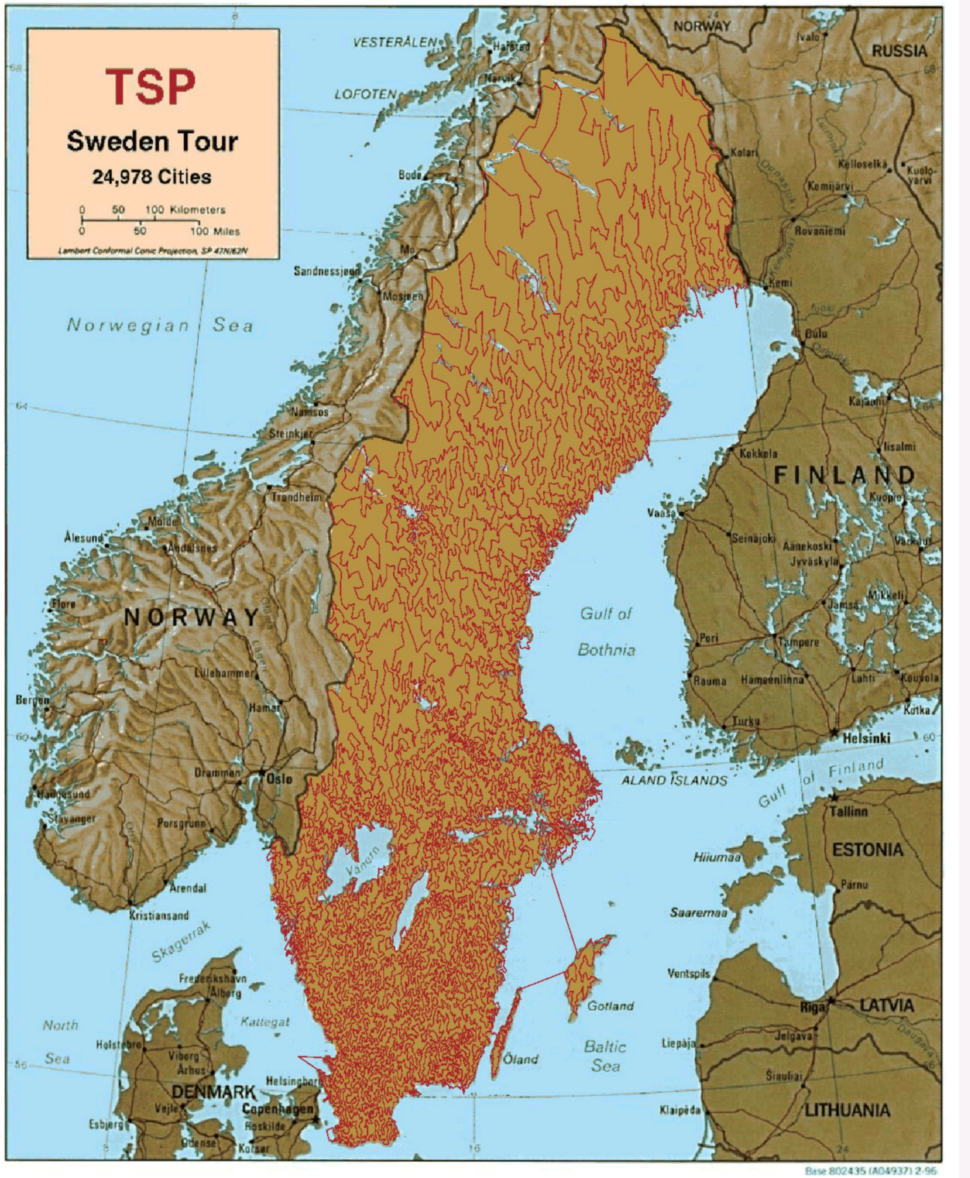
Suboptimale Tour mit Länge 18.56.



Optimale Tour mit Länge 14.71.

- Der einfache approximative Algorithmus hat eine Komplexität von $O(\log|V| \cdot |V|^2)$.
- Christofides verbesserte (1976) den Algorithmus so, dass gefundene Touren maximal 1.5-mal so lang wie die optimalen sind.

Zahlreiche ausgefeilte Lösungsverfahren für exakte optimale Touren



- Benchmarks für TSP: www.math.uwaterloo.ca/tsp/index.html
- Beispiel: kürzeste Tour für 24,978 Städte in Schweden.
- Gesamtlänge der Tour: 72,500 Kilometer
- Rechenzeit: 8 Jahre auf einem Linux-Workstation-Cluster (96 dual processor Intel Xeon 2.8 GHz)
- Verfahren: ganzzahlige lineare Optimierung mit Branch-and-Cut
- Ausgangspunkt war bereits eine sehr gute suboptimale Lösung
- Stand: 2003

Erfüllbarkeitsproblem SAT

Erfüllbarkeitsproblem SAT

- Gegeben sei eine aussagenlogische Formel mit n Aussagenvariablen.
- Entscheide, ob die Formel erfüllbar ist.

Beispiel mit 4 Aussagevariablen:

$$((x_1 \rightarrow x_2) \wedge ((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge x_2$$

ist erfüllbar für $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$.

Satz von Cook (1971)

SAT ist NP-vollständig.

Beweis: siehe z.B. [Cormen u.a.]

- SAT war das erste Problem aus NP, dessen NP-Vollständigkeit bewiesen wurde.
- Die NP-Vollständigkeit eines anderen Problems C wird bewiesen, in dem ein bereits bekanntes NP-vollständiges Problem auf C reduziert wird.

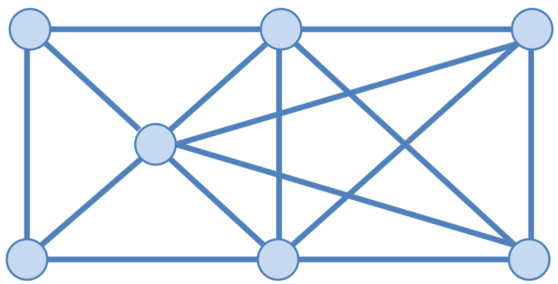
Clique

Entscheidungsproblem

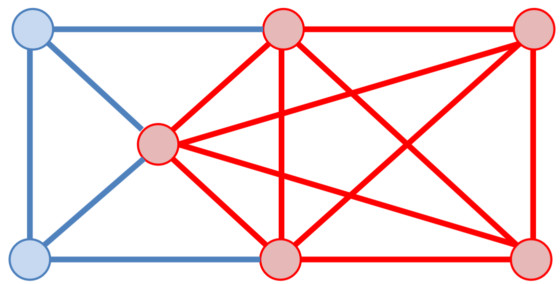
Sei G ein ungerichteter Graph.
Gibt es einen vollständigen Teilgraph (Clique) mit k Knoten?

Suchproblem

Suche eine Clique mit größtmöglicher Knotenzahl.



Graph enthält Cliquen mit 2, 3, 4 und 5 Knoten.



Größte Clique mit 5 Knoten

Knotenüberdeckungsproblem (Vertex Coverage Problem)

Entscheidungsproblem

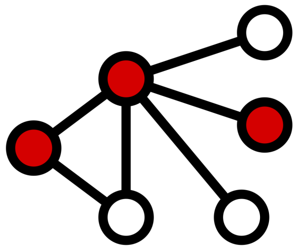
Sei G =(V, E) ein ungerichteter Graph. Eine Teilmenge V′ ⊆ V heisst Knotenüberdeckung, falls für jede Kante (u,v) ∈ E gilt: u ∈ V′ oder v ∈ V′.

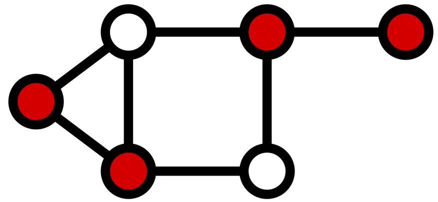
Gibt es in G eine Knotenüberdeckung mit maximal K Knoten?

Suchproblem

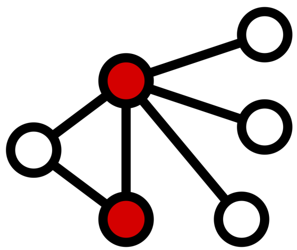
Suche in einem Graph eine Knotenüberdeckung mit kleinster Knotenzahl.

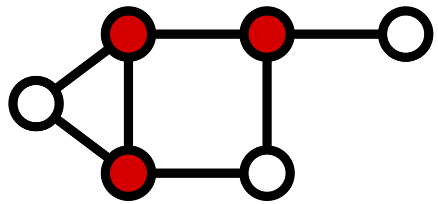
Knotenüberdeckung
mit 3 bzw. 4 Knoten.





Minimale Knotenüberdeckung
mit 2 bzw. 3 Knoten.





https://en.wikipedia.org/wiki/Vertex_cover

Longest Path Problem

Entscheidungsproblem

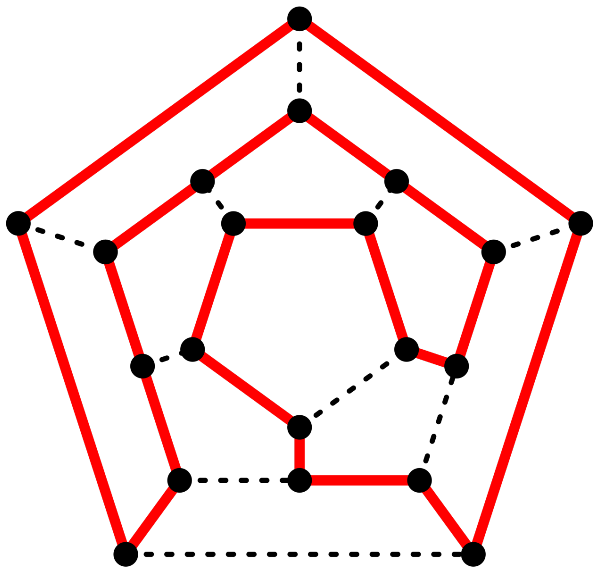
Sei G ein gewichteter oder ungewichteter Graph.

Gibt es in G einen einfachen Weg (keine Knoten werden mehrfach besucht), deren Weglänge ≥ K ist?

(In einem ungewichteten Graphen ist die Weglänge gleich der Anzahl der Kanten.)

Suchproblem

Suche in einem Graph einen einfachen Weg mit maximaler Weglänge.



Durch Entfernen einer beliebigen Kante aus dem roten Hamilton-Kreis erhält man einen einfachen Weg maximaler Länge.

https://de.wikipedia.org/wiki/Längster_Pfad

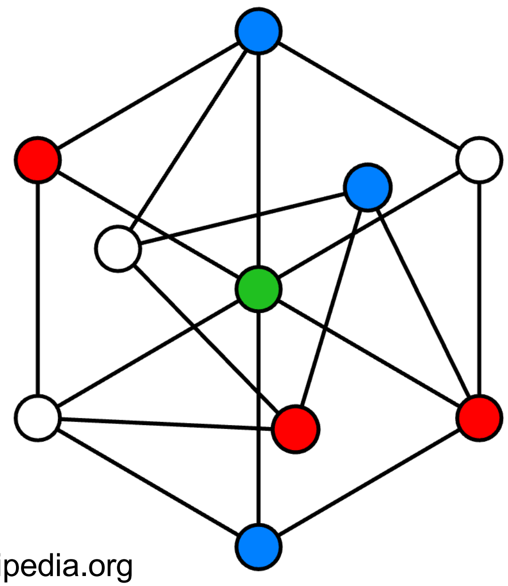
k-Färbungsproblem

k-Färbungsproblem

- Set G ein ungerichteter Graph und k eine Anzahl von Farben. Kann jedem Knoten aus G einer der k Farben so zugeordnet werden, dass benachbarte Knoten nicht dieselbe Farbe haben?
- Das k-Färbungsproblem ist für $k \geq 3$ NP-vollständig.

Bemerkungen

- Für $k = 2$ lässt sich das Problem durch einfache Tiefensuche lösen (siehe bipartite Graphen).
- Falls der Graph planar ist, dann lässt sich der Graph mit 4 Farben einfärben (4-Farbenproblem, Satz von Appel und Haken, 1976).



de.wikipedia.org

- Graph mit $n = 10$ Knoten.
- Graph ist mit 4 Farben korrekt eingefärbt.
- Graph ist damit 4-färbbar.
- Graph ist aber nicht 3-färbbar!
- Aufgabe: Warum ist Graph nicht 3-färbbar? Überlegung ohne Ausprobieren der $3^{n-1} = 3^9 = 19683$ Möglichkeiten!