Ideen der Informatik

Ein Rundgang durch die Komplexität

[Was geht? Was geht schwer? Was geht gar nicht?]

Kurt Mehlhorn

Adrian Neumann

Folien von Kosta Panagiotou

Plan für Heute

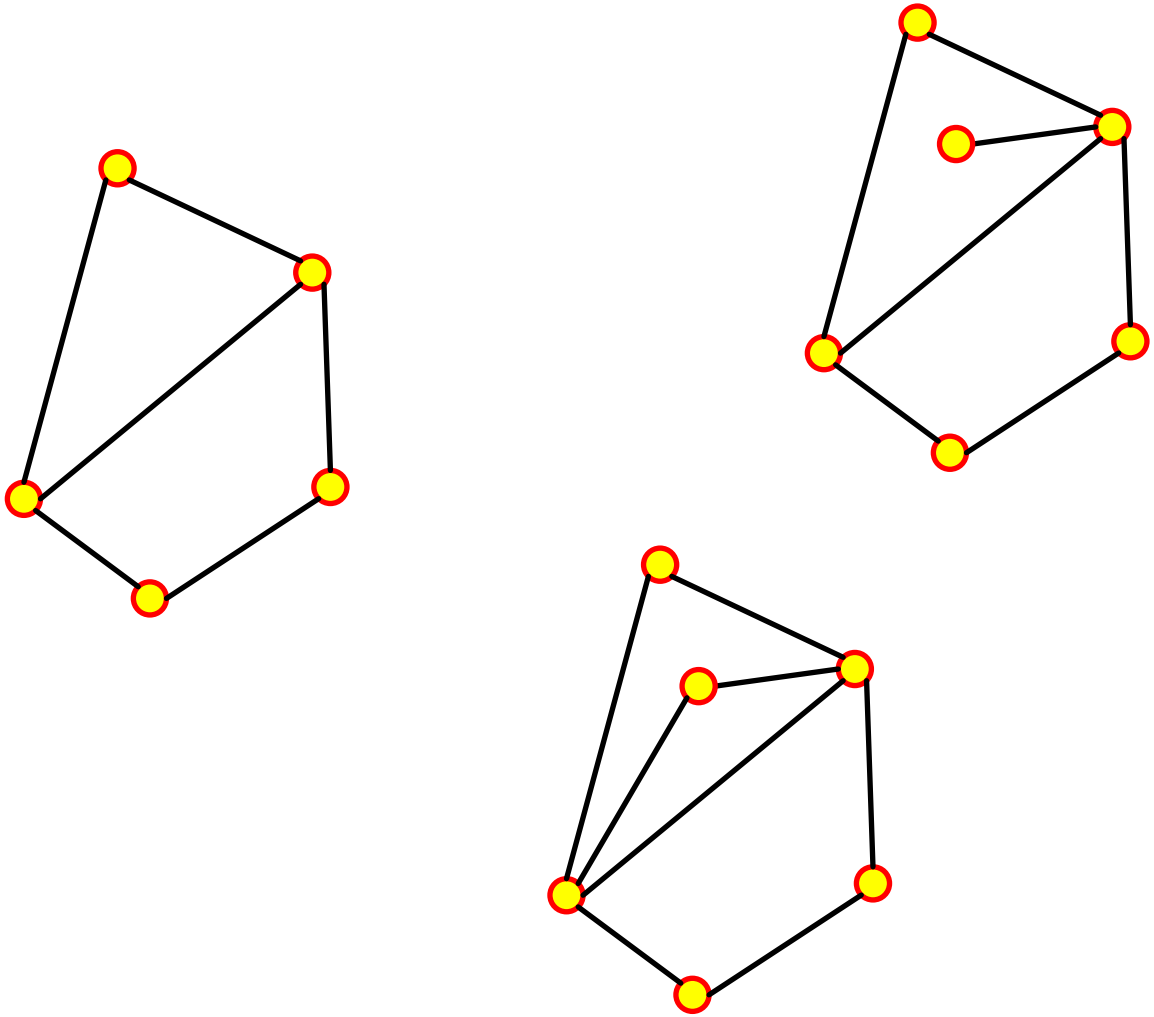
- Komplexitätsklassen
  - P = effizient lösbar
    - Kürzeste Wege, Sortieren, Gleichgewichtspreise
  - NP = wahrscheinlich nicht effizient lösbar
    - Hamiltonscher Kreis, Rucksackproblem, Dreifärbung, …
    - NP-vollständige Probleme
- Unentscheidbare Probleme

Problem II: **Hamiltonkreise**

Ein Graphenproblem

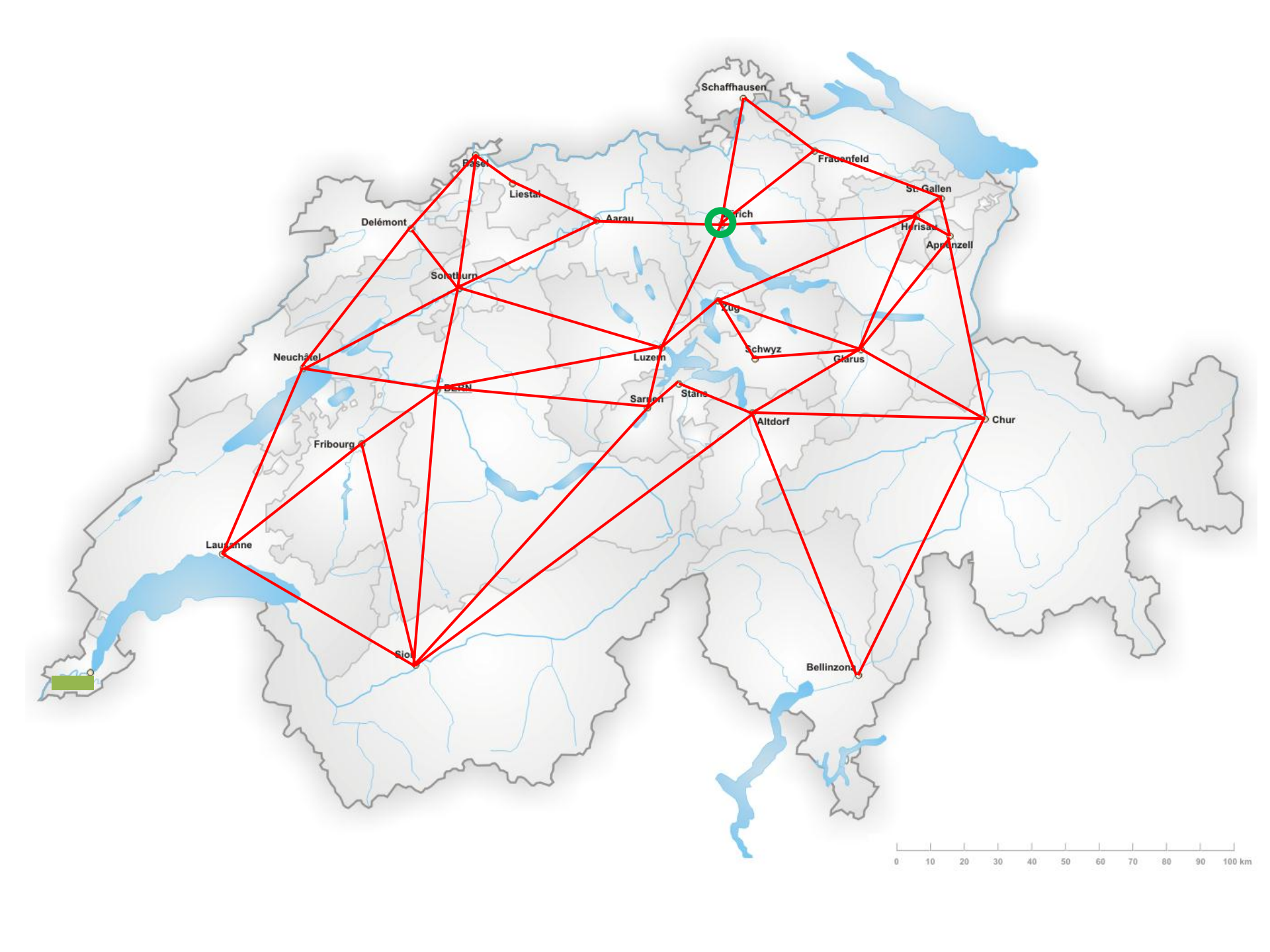
- Ein Reisender möchte bestimmte Städte besuchen
- Er kennt die Verbindungen zwischen den Städten
- Am Schluss möchte er wieder an seinem Ausgangsort ankommen
- Er will keine Stadt mehrmals besuchen

Beispiele

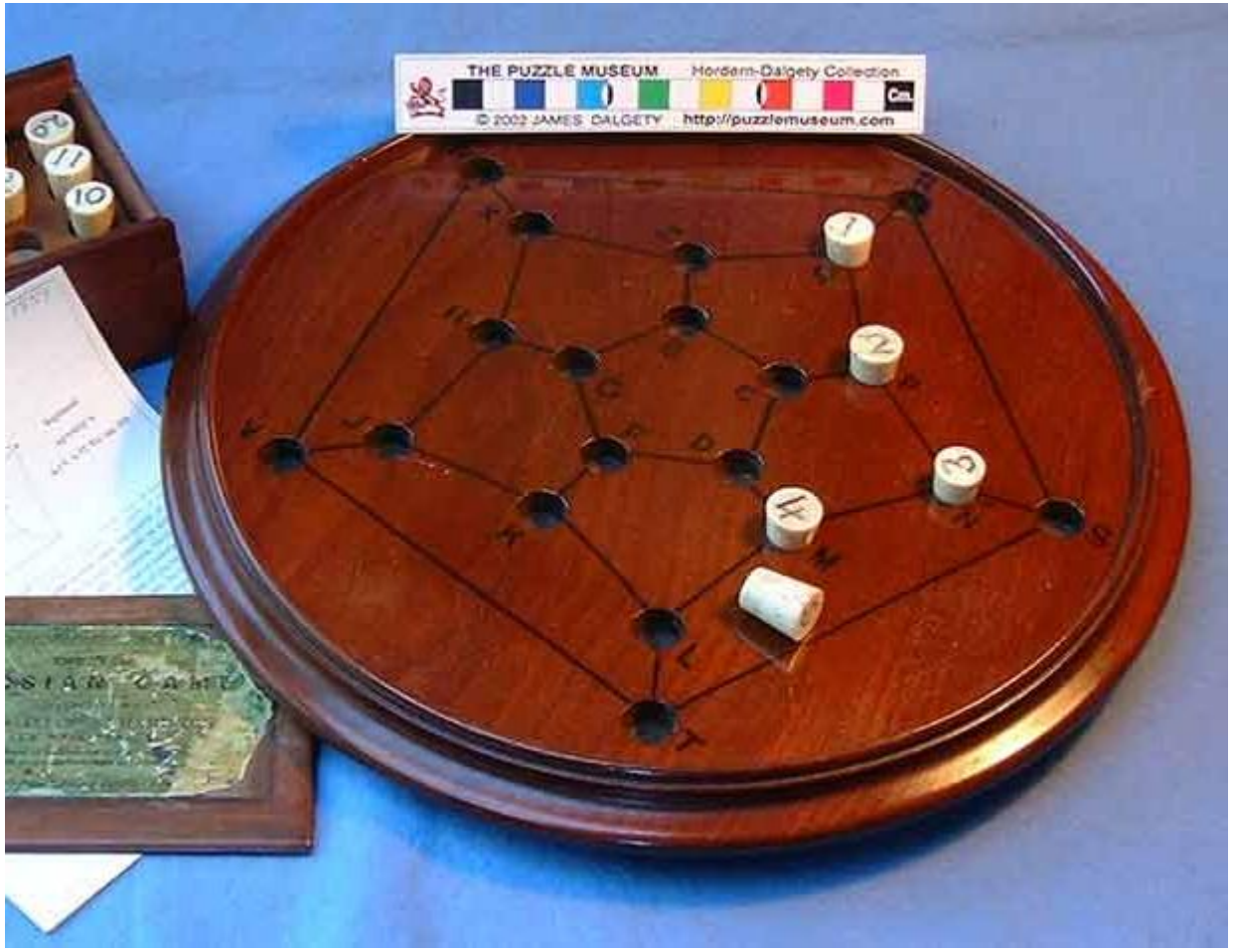


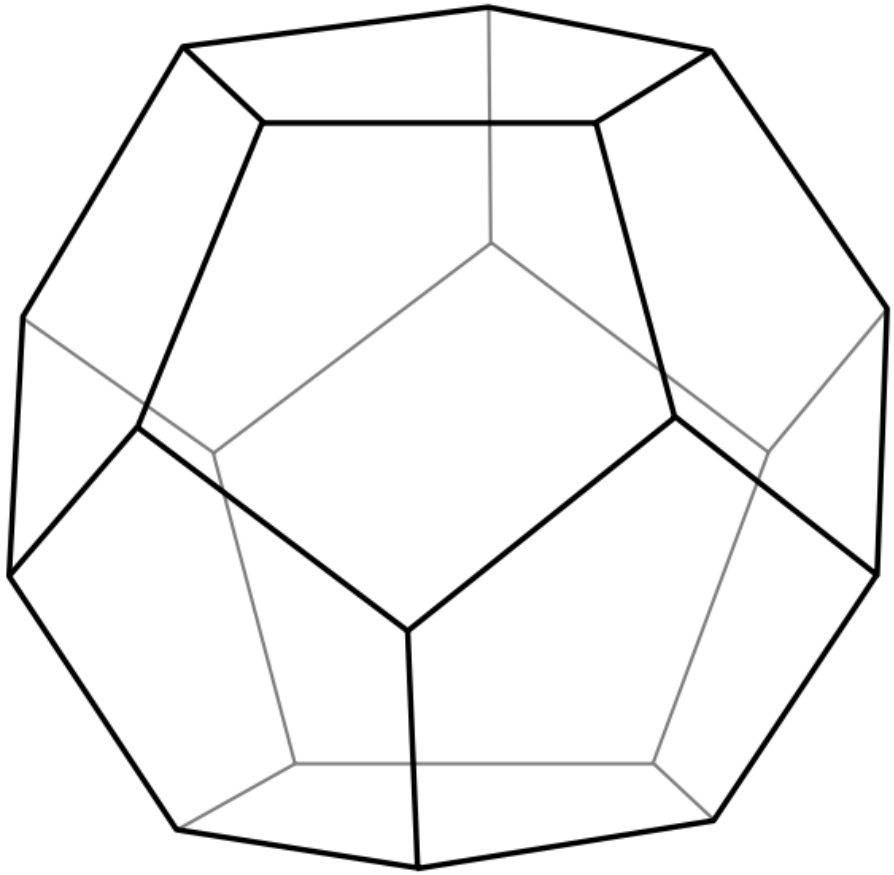
Das *Problem*

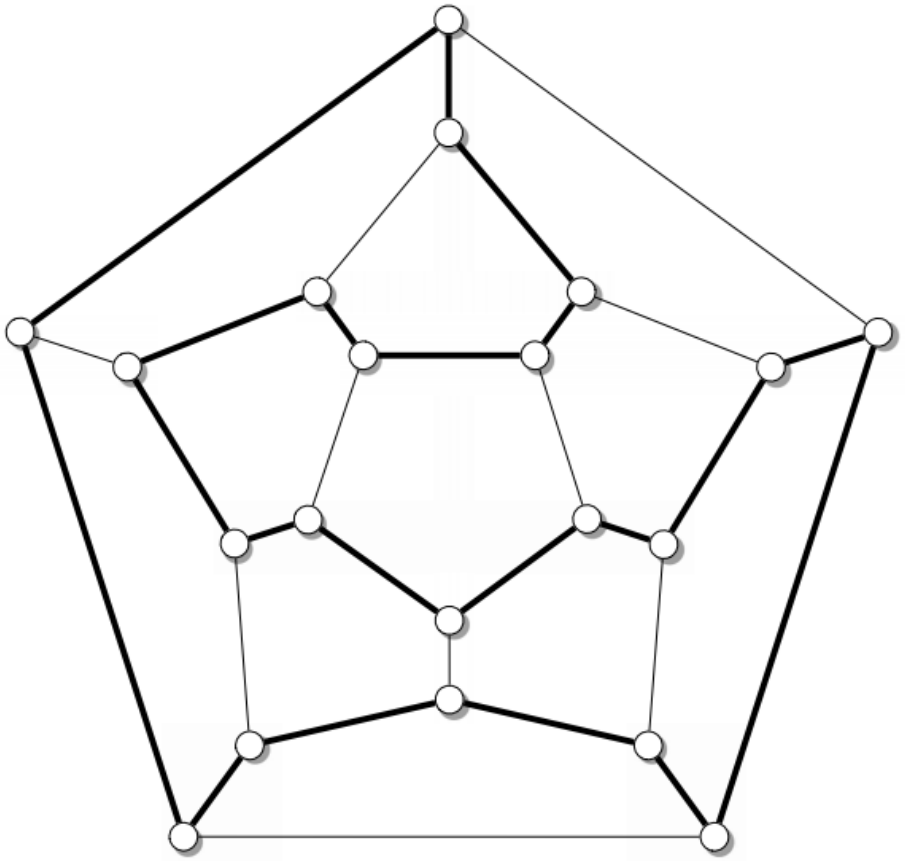
- Hamiltonkreis
  - Gegeben: ein Graph
  - Frage: gibt es einen Kreis, der jeden Knoten genau einmal besucht?


Schaffhausen
Frauenfeld
Basel
Liestal
St. Gallen
Aarau
Delémont
Herisau
Appenzell
Solothurn
Zug
Schwyz
Neuchâtel
Luzern
Glarus
Stans
Altdorf
Chur
Fribourg
Lausanne
Bellinzona
0 10 20 30 40 50 60 70 80 90 100 km

Ursprung

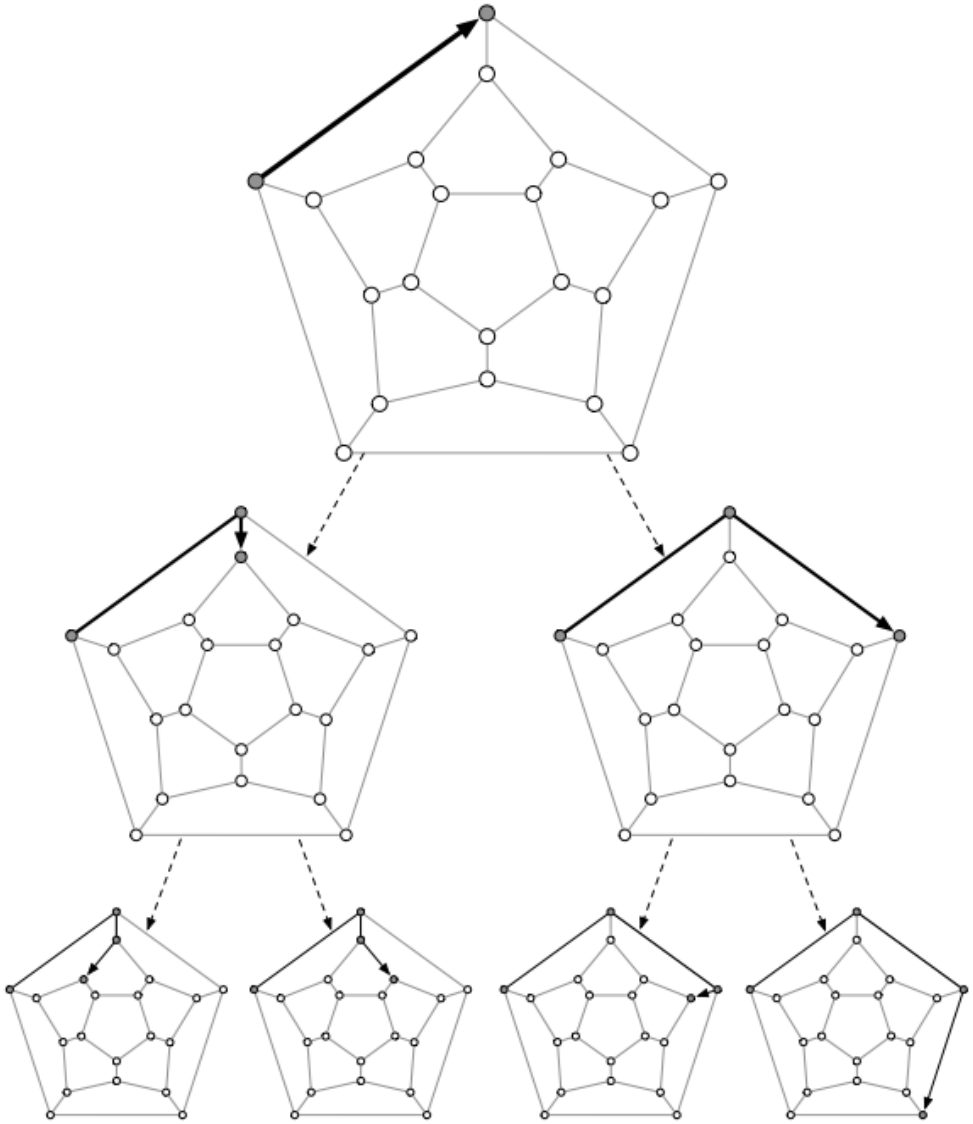

THE PUZZLE MUSEUM
Hordern-Dalgety Collection
© 2003 JAMES DALGETY
http://puzzlemuseum.com





Wie macht man das?

– „[…] so kann es gelöst werden, *indem alle möglichen Wege ausprobiert werden*, um herauszufinden […]“ (Euler)



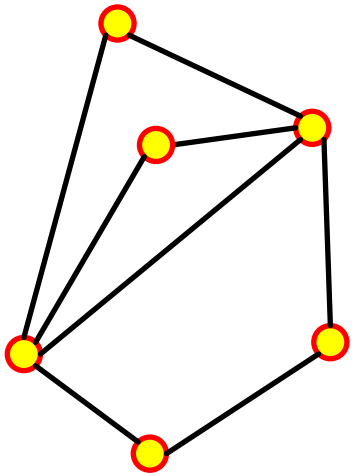
Stand der Kunst

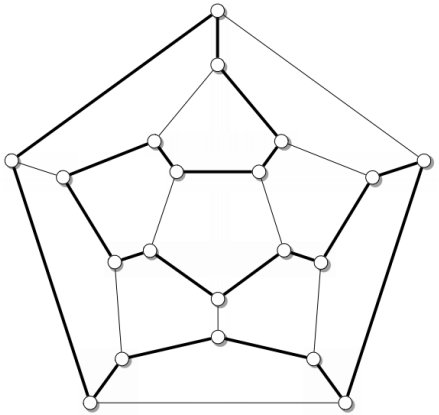
- Man kennt *keine* „*schöne*“ Eigenschaft, die uns ermöglicht, „*schnell*“ zu entscheiden, ob ein Graph einen Hamiltonkreis hat.
- Konsequenz: Der einzige Algorithmus, den man kennt, ist der triviale Algorithmus.
- Es gibt relativ kleine Instanzen (10000 Knoten), für die man die Lösung nicht kennt.

- Noch schlimmer: man glaubt, dass es keine solche Eigenschaft *gibt*!
- Kürzeste Wege und Hamiltonkreis sind in einer *fundamentalen* Weise unterschiedlich.

Andererseits…

- Es ist einfach zu überprüfen ob ein gegebener Graph eine Lösung ist
  - Sind alle Knoten unterschiedlich?
  - Ist das wirklich ein Kreis?
- Es ist aber gar nicht klar,
  wie man verifiziert,
  dass ein Graph *keinen*
  Hamiltonischen Kreis hat
- Das Problem ist in dieser Hinsicht *asymmetrisch*





Meilenstein II

- Eine cleverere Einsicht erlaubt uns
  - die vollständige Suche zu *vermeiden*
  - zu *verstehen*, warum es eine Lösung (nicht) gibt
- Kürzeste Wege : solche Einsicht ist möglich → *effizienter* Algorithmus
- Hamiltonkreisproblem: ??? → *exponentieller* Algorithmus
- Es ist trotzdem möglich, Lösungen zu verifizieren → *polynomieller* Algorithmus

Problem III: **Partition**

Problem

- Partition
  - Gegeben: *n* Zahlen $a_1, a_2, \dots, a_n$
  - Frage: Können die Zahlen in zwei Gruppen aufgeteilt werden, die dieselbe Summe haben?
  - Es ist nicht notwendig, dass die Gruppen gleich groß sind.

Beispiel

1, 7, 8, 12, 1, 3
13, 2, 6, 7, 5, 8, 11

1, 2, 4

Kommentare

- Es ist einfach, eine Lösung zu verifizieren
  - Überprüfe ob die Summen gleich sind
  - Überprüfe ob alle Elemente der Eingabe vorkommen
- Es ist nicht klar, wie man bestätigt, dass es keine Lösung gibt

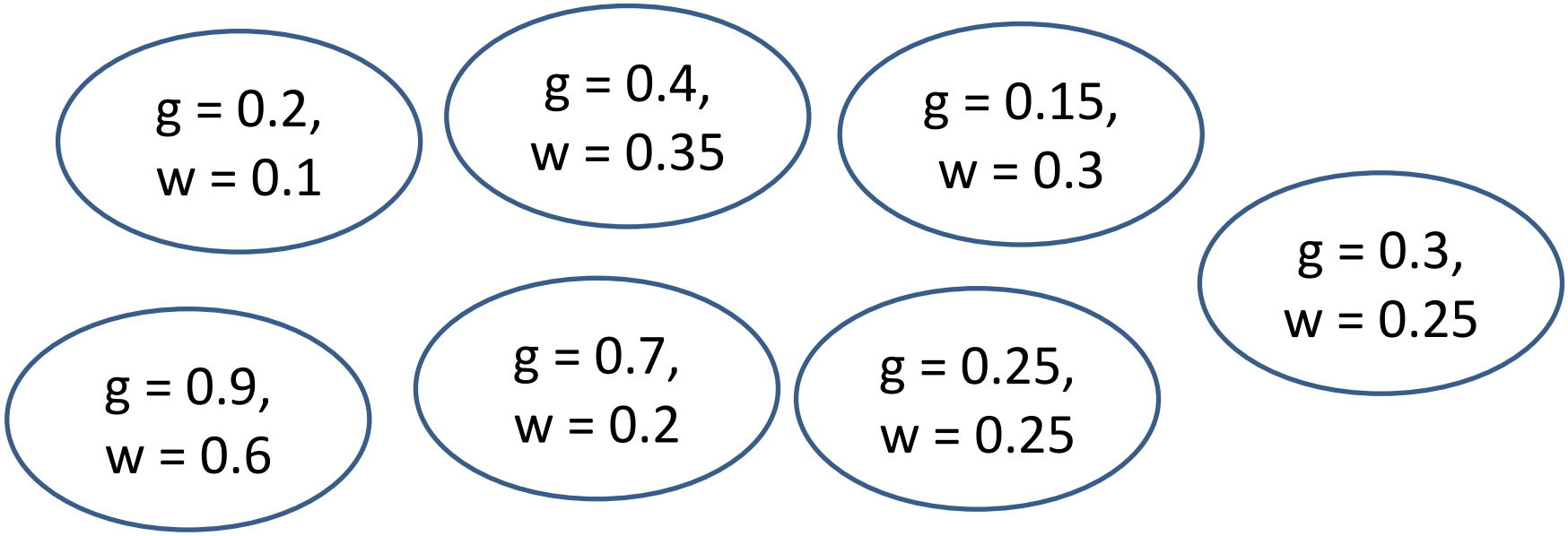
- Man weiß es ebenfalls nicht…

Problem IV: **Rucksack**

Sachen packen

- Knapsack
- Gegeben
  - *n* Gegenstände
  - Jeder Gegenstand hat ein Gewicht: $g_1, g_2, \ldots, g_n$
  - Jeder Gegenstand hat einen Wert: $w_1, w_2, \ldots, w_n$
  - Ein Rucksack, der Kapazität 1 hat
  - Ein Wert *W*
- Frage: Gibt es eine Teilmenge der Gegenstände, die in Summe Wert $\geq W$ haben und in den Rucksack *passen*?

Beispiel



Kommentare

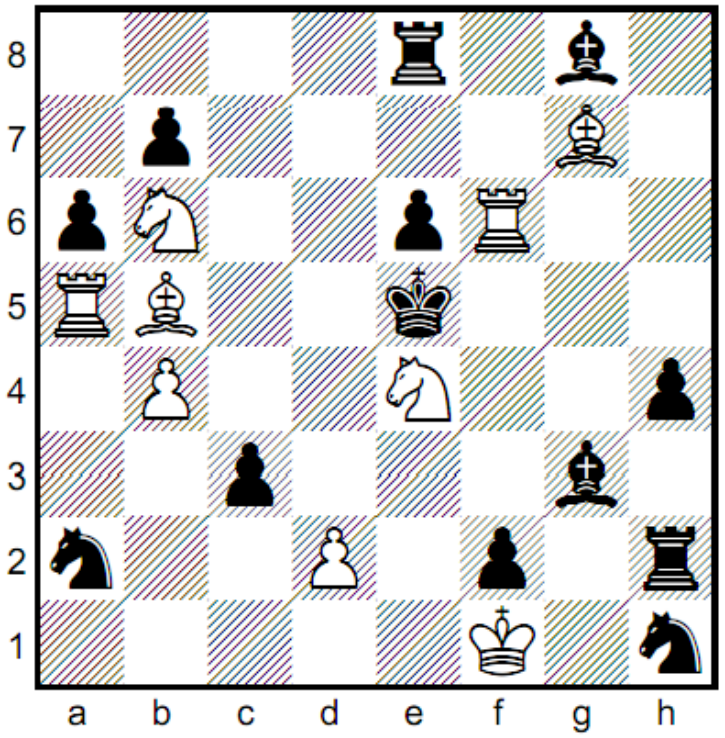
- Es ist einfach, eine Lösung zu verifizieren
  - Überprüfe ob die Summe der Gewichte ≤ 1 ist
  - Überprüfe ob die Summe der Werte $\geq W$ ist
  - Überprüfe ob alle Elemente in der Eingabe vorkommen
- Es ist nicht klar, wie man bestätigt, dass es keine Lösung gibt

- Man weiß es ebenfalls nicht…

Problem V: **Schach**

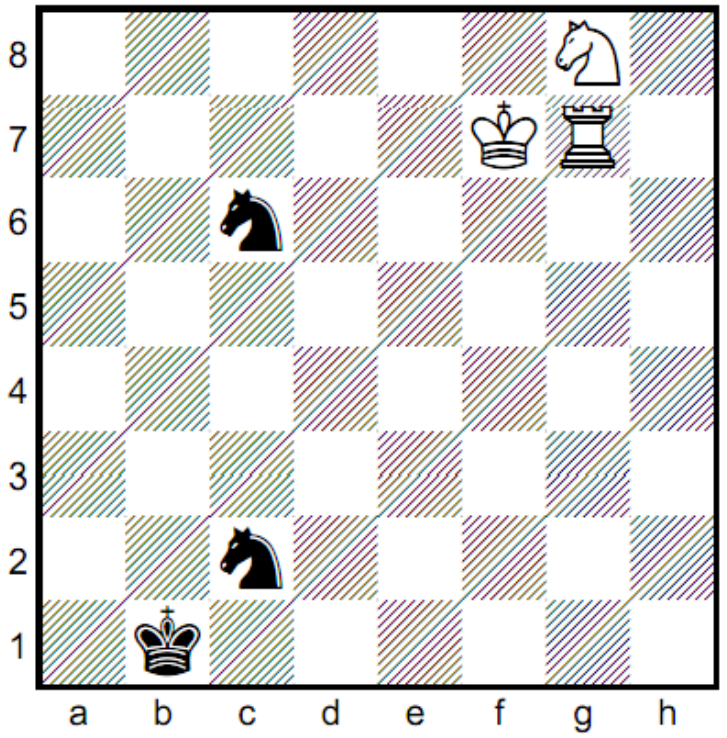
Ein Spiel

Sam Loyd (1903)



Matt in 3 Zügen?

Lewis Stiller (1995)



Matt in 262 Zügen?

Allgemeines Schach

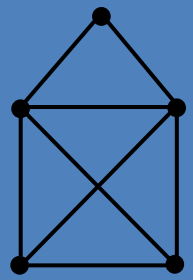
- Gegeben
  - Eine Stellung = *n* x *n* Brett + Figuren auf dem Brett
  - Zahl *z*
- Frage: kann Weiß in *z* Zügen gewinnen?

Kommentare

- Was ist eine Lösung?
- Wir haben einen Gegner, der versuchen wird, unsere Strategie zu widerlegen
- Wir müssen garantieren, dass *unabhängig davon*, wie der Gegner zieht, *wir immer gewinnen.*
- *Lösung = vollständiger Spielbaum der Tiefe 2z*
- *Größe der Lösung ist exponentiell in z*

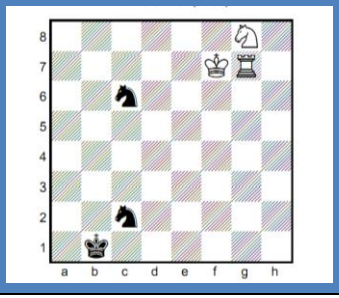
Zusammenfassung

Kürzeste Wege

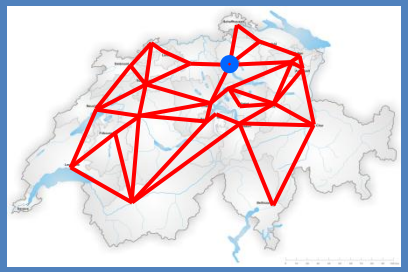


Partition

Schach



Hamiltonkreis



Rucksackproblem

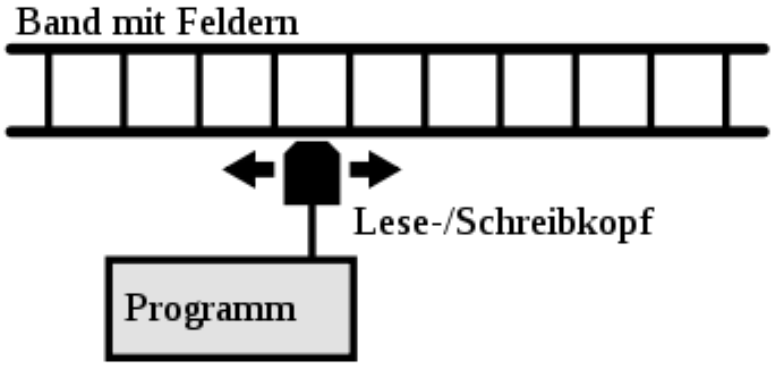
Meilenstein III

- Ein cleverer Einblick erlaubt uns
  - die vollständige Suche zu *vermeiden*
  - zu *verstehen*, warum es eine Lösung (nicht) gibt
- Kürzeste Wege : solcher Einblick ist möglich → *effizienter* Algorithmus
- Hamiltonkreisproblem, Partition, Rucksack: ??? → *exponentieller* Algorithmus
- Es ist trotzdem *effizient* möglich, Lösungen zu verifizieren,
- Schach: es ist *nicht* einmal *klar*, wie man Lösungen effizient *verifiziert, da Lösungen riesig groß sind*

Komplexität
und
Komplexitätsklassen

Die Turing Maschine

- Turing Maschine
  - ***Speicher***, in Zellen unterteilt. Jede Zelle enthält ein Symbol aus einem endlichen Alphabet *A*.
  - ***Kopf***, der in einem aus endlich vielen *Zuständen* ist. Zeigt auf eine Position im Speicher.
- Berechnung erfolgt in Schritten



In jedem Schritt:

- Speicher wird an der Stelle des Kopfes aktualisiert
- Zustand wird aktualisiert
- Kopf bewegt sich rechts/links

$$F: A \times S \rightarrow A \times S \times \{-1, 1\}$$

Die Maschine stoppt, falls keine Regel mehr anwendbar ist.

Ein Maß für Komplexität

- Gegeben:
  - Turing Maschine M
  - Eingabe/Instanz *e*, geschrieben auf dem Band

- ***Komplexität*** von *e* = ***Anzahl Schritte*** die benötigt werden, bis die Maschine mit Eingabe *e* stoppt.

- ***Komplexität*** eines ***Problems***: ***maximale Anzahl Schritte*** für Instanzen einer bestimmten *Größe*

Die Klasse **P**

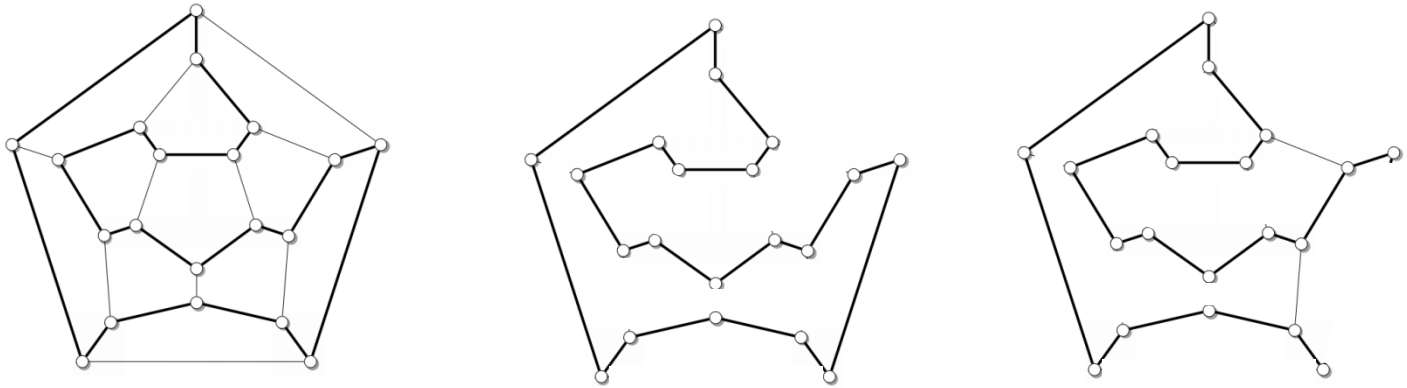
- Die Klasse **P** enthält alle Probleme für die es eine Turing Maschine gibt, die für jede Eingabe ***polynomiell*** viele Schritte in der Größe der Eingabe macht, und mit der ***richtigen Antwort*** stoppt.
- Beispiele: $\mathrm{Kürzeste\ Wege}$ braucht n^2 Schritte bei einem Graphen mit n Knoten
- Sortieren von n Elementen braucht Zeit $n \log n$

Weitere Probleme in **P**

- Arithmetische Operationen
- Eulertour
- Kürzeste Pfade
- Berechnung optimaler Codes
- Matrixmultiplikation
- Gegeben n Zahlen: sind sie alle verschieden?

Die Beobachtung

- Hamiltonkreis, Partition und Knapsack
  - einzige (bekannte) Möglichkeit: Alles durchsuchen
  - Aber: falls es eine Lösung gibt, so kann man sie in polynomieller Zeit verifizieren



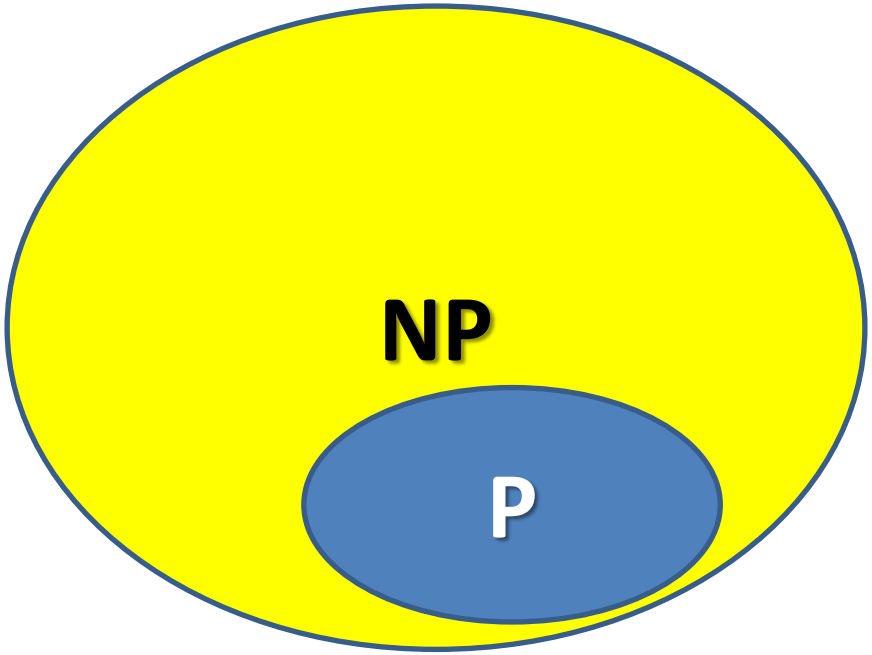
Sind alle Knoten unterschiedlich?
Ist das wirklich ein Kreis?

Zwei Probleme

- Hamiltonkreis
  - Gegeben: ein Graph G
  - Frage: gibt es einen Kreis in G, der jeden Knoten in G genau einmal besucht?
- Hamiltonkreis – Check
  - Gegeben: ein Graph G, und ein weiterer Graph K
  - Ist K ein Hamiltonkreis von G?

Die Klasse **NP**

- Ein Problem ist **NP** wenn folgendes gilt:
  falls die Antwort für die gegebene Instanz „Ja“ ist, so gibt es einen Grund dafür, und das Problem ihn zu verifizieren ist in **P**.



Jedes Problem in **P** ist auch in **NP** enthalten!

$\mathbf{P} \subseteq \mathbf{NP}$

Etwas formaler

- Ein Problem **A** ist in **NP** falls
  - Sei *x* eine „Ja“ Instanz von **A**
  - Dann gibt es ein *w*, so dass
    - Die Länge von *w* (in bits) ist ***polynomiell*** in der Länge von *x*.
    - Es gibt ein Problem **B**, dessen „Ja“ Instanzen ***genau alle solchen Paare*** (*x,w*) sind, und **B** ist in **P**.

Weitere Beispiele

- Partition
  - Gegeben: *n* Zahlen $a_1, a_2, \ldots, a_n$
  - Frage: Können die Zahlen in zwei Gruppen aufgeteilt werden, die dieselbe Summe haben?
- Partition – Check
  - Gegeben: *n* Zahlen $a_1, a_2, \ldots, a_n$, und eine Aufteilung in zwei Mengen *X* und *Y*
  - Ist die Summe aller Elemente in *X* gleich derer in *Y*?

- Knapsack
  - Gegeben: *n* Gegenstände
  - Jeder Gegenstand hat ein Gewicht: $g_1, g_2, \ldots, g_n$ und einen Wert: $w_1, w_2, \ldots, w_n$
  - Rucksack mit Kapazität 1
  - Ein Wert *W*
  - Frage: Gibt es eine Teilmenge der Gegenstände, die in Summe Wert ≥ *W* haben und in den Rucksack *passen*?
- Knapsack – Check
  - Eingabe wie vorher, und eine Auswahl an Elementen
  - Ist die Summe der Gewichte ≤ 1 und der Gesamtwert ≥ *W*?

Ein Nicht-Beispiel (?)

- Allgemeines Schach
  - Gegeben: *n* x *n* Brett, Figuren
  - Zahl *z*
  - Frage: kann Weiß nach höchstens *z* Zügen gewinnen?
- Allgemeines Schach – Check
  - Gegeben: wie vorher, und eine Strategie
  - Frage: führt die Strategie nach höchstens *z* Zügen zum Sieg für Weiß?

NP-Vollständige Probleme

Diskussion

- Viele Probleme sind in **NP**
  - mit vielen praktischen Anwendungen
    - Packprobleme
    - Scheduling
- Für viele von denen weiß man nicht ob sie in **P** sind
- Sind wir nicht schlau genug??
- Ist **P** = **NP**??

Reduktion

- Eine *Reduktion* ermöglicht uns zu sagen, dass ein Problem „nicht wesentlich schwerer“ ist als ein anderes
- Seien **A** und **B** zwei Probleme. Dann heißt

$$\mathbf{A} \leq \mathbf{B}$$

dass **A** höchstens so schwer ist wie **B** im folgenden Sinne:

  - Falls **B** in polynomieller Zeit lösbar, so auch **A**

Reduktion (II)

- Wie zeigt man dass **A** ≤ **B**?
- Idee: Man löse jede **A**-Instanz *e*, indem man sie in eine **B**-Instanz *e‘* übersetzt, aus deren Lösung dann die Antwort für *e* bestimmt werden kann.
- Zwei Bedingungen:
  - Die (Rück-)Übersetzung muss in polynomieller Zeit stattfinden.
  - Die Instanz *e‘* darf nicht zu groß sein, d.h., höchstens polynomiell groß in der Größe von *e*.

Beispiel

- AlleUnterschiedlich
  - Eingabe:
  - Frage:

Partition ≤ Knapsack

- Partition
  - Gegeben: *n* Zahlen $a_1, a_2, \dots, a_n$
  - Frage: Können die Zahlen in zwei Gruppen aufgeteilt werden, die dieselbe Summe haben?
- Knapsack
  - Gegeben: *n* Gegenstände
  - Jeder Gegenstand hat ein Gewicht: $g_1, g_2, \dots, g_n$ und einen Wert: $w_1, w_2, \dots, w_n$
  - Rucksack mit Kapazität 1
  - Ein Wert *W*
  - Frage: Gibt es eine Teilmenge der Gegenstände, die in Summe Wert ≥ *W* haben und in den Rucksack *passen*?

NP-Vollständigkeit

- Ein Problem **A** heißt **NP**-*Vollständig*, falls *für alle* Probleme **B** in **NP** gilt

$$\mathbf{B} \leq \mathbf{A}$$

- **A** ist also unter den schwierigsten Problemen
  - Könnte man **A** in poly Zeit lösen, so könnte man alle Probleme in **NP** in poly Zeit lösen!

- Kennt man so ein Problem?
- Ja, Partition! [Beweis ist sehr kompliziert.]

Moment mal…

- Wir haben doch gezeigt, dass

$$\text{Partition} \leq \text{Knapsack}$$

- Also, Knapsack *ist auch* NP-Vollständig!
- … Hamiltonkreis auch!
- … und viele andere tausende Probleme!

- Fazit: Hat man eins davon effizient gelöst, so hat man ***alle*** gelöst!
- … und man hat auch ausgesorgt…! ☺

Pessimismus

- Für ein einzelnes Problem könnte man denken, dass wir nicht schlau genug sind…
- Aber für so viele gleichzeitig…?!

- Die meisten Forscher glauben dass **P ≠ NP**
- Eines der größten offenen Probleme nicht nur in der Informatik

Was wäre, wenn **P** = **NP**?

- Das heißt: Lösungen zu finden ist (fast) so einfach wie sie zu verifizieren.
- Fermat‘s Last Theorem:

  $x^n + y^n = z^n$ hat keine positive

  ganzzahlige Lösungen für $n \geq 3$
- Mehr als 350 (!) Jahre hat es gedauert, um einen Beweis zu finden. [Wiles ‘95]
- Falls **P** = **NP**, dann ginge das (vermutlich) sehr schnell.

Was wäre, wenn P = NP?

- Gibt es für Fermat‘s Last Theorem einen (formalen) Beweis der Länge L?
- formaler Beweis = Beweis mit allen Details, so dass ein Algorithmus ihn prüfen kann
- Das obige Problem ist in NP.
- Menschen können nur „kurze“ Beweise finden.
- Falls P = NP, werden Mathematiker überflüssig

Unentscheidbare Probleme

Church-Turing

- Alle Probleme, die durch einen Menschen gelöst werden können, können auch von einer Turing Maschine gelöst werden.

- Gibt es ein Problem, dass wir/Turing Maschinen nicht lösen können?
  - Das wäre ein Problem, dass kein Algorithmus lösen kann.

Turing Maschinen aufs Band

- Jede Turing Maschine kann codiert werden
  - $F: A \times S \rightarrow A \times S \times \{-1, 1\}$
- Benötige Codes für
  - Das Alphabet *A*
  - Den Zustandsraum *S*
  - Die Übergangsfunktion *F*
- Genügt: Binärcode

Das Halteproblem

- Gegeben: eine Turing Maschine *M,* eine Eingabe *e*
- Frage: *hält M* mit Eingabe *e*?

Es gibt keine Turingmaschine, die das Haltproblem löst. Das Halteproblem ist *unentscheidbar.*

Fermat ≤ Halteproblem

- Betrachte folgenden Algorithmus

  for t = 3, 4, 5, …

  for *n*, x, y, z = 1, 2, …, t

  if $x^n + y^n = z^n$ **return** „Fermat hatte unrecht“

- Wir wissen seit Wiles: dieser Algorithmus terminiert nicht
- Falls wir einen Algorithmus für das Haltproblem hätten, hätte es Wiles nicht gebraucht

Eine merkwürdige Konstruktion

- Angenommen, es gäbe eine Turing Maschine Halt(.,.), die Halt löst
- Catch
  - Eingabe: eine Turing Maschine M

    if Halt(*M*,*M*) = „hält an“ **then** Endlosschleife

    else halte an
- Verhalten von Catch
  - *stoppt nicht*, falls *M* mit Eingabe *M* *stoppt*
  - *stoppt*, falls *M* mit Eingabe *M* *nicht stoppt*

Geht nicht ☹

- Catch
  - *stoppt nicht*, falls *M* mit Eingabe *M stoppt*
  - *stoppt*, falls *M* mit Eingabe *M nicht stoppt*
- Sei M = Catch
- Dann
  - *stoppt nicht*, falls Catch mit Eingabe Catch *stoppt*
  - *stoppt*, falls Catch mit Eingabe Catch *nicht stoppt*
  - -> Halt(.,.) kann es nicht geben.

Zusammenfassung

- Komplexität: universell definiert durch Turing Maschinen
- Klassen: **P** vs. **NP**
- Unentscheidbare Probleme

Kleine BUGs, große GAUs

- Am 4.Juni 1996 startete die ESA eine Rakete von Französisch Guyana aus.
- Vierzig Sekunden nach dem Start explodierte die Rakete.
- Verlust ca. 500 Millionen Euro

- Ursache: Absturz des Bordcomputers 36.7 Sek. nach dem Start
- Problem: Additionsfehler (!)

Kleine BUGs, große GAUs (II)

- Pentium-Prozessor Fehler (1994)
- Beispiel des auftretenden Fehlers bei Division:
- $x = 4195835.0$
- $y = 3145727.0$
- Berechne: $z = x - (x / y) \cdot y$
- Bei exakter Rechnung: $z = 0$
- Intel Pentium lieferte: $z = 256$

Weitere GAUs…

- Wall Street Börsencrash 19.10.1987
  - US Federal Reserve System (Zentralbank)
  - Falsche Überweisung von 28 Milliarden $
  - durch neues Computersystem; zurück kamen 24 Milliarden $!

- … und viele mehr.