Die Klasse P der in polynomieller Zeit lösbaren Probleme

Problem:

- EINGABE $\in \Sigma^*$, in einem passenden Alphabet Σ kodiert

  z.B. $\Sigma = \{0,1\}$

  z.B. Liste von Knoten eines Graphen (Nummern)
  Kanten mit zugehörigen Gewichten (Dezimalzahlen)

  1,7,2,6,13/(1,7):7.42, (6,2):12.6604, ...

Länge der Eingabe $x \in \Sigma^*$: $n = |x|$

- AUSGABE $y \in \Sigma^*$

  z.B. Liste von Knoten mit Abständen vom ersten Knoten (kürzest Wege)

  1:0; 7:3.14; 2: .....

LAUFZEIT eines Algorithmus:

$T(n) := \max \{$ Laufzeit (="Anzahl von Schritten") über alle Eingaben der Länge $\leq n \}$

DEFINITION

$$P = \{ \text{Probleme } A \mid \text{Es gibt einen Algorithmus, der } A \text{ löst und Konstanten } a, k \text{ mit Laufzeit } T(n) \leq a \cdot n^k \text{ für alle } n \geq 1 \}$$

→ Polynomielle Laufzeit

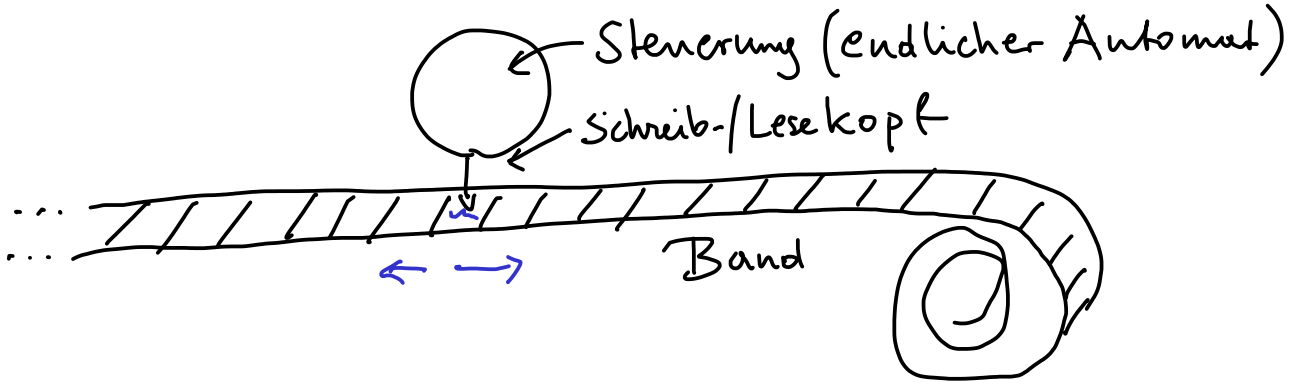
- kürzeste Wege in einem Graphen mit positiven Kantenlängen. $\in P$

- Anzahl der Lösungen für das k-Damenproblem auf einem k×k-Feld $\overset{?}{\in} P$

Laufzeit / Rechnermodell

Genaue Definition der Laufzeit erfordert eine (formale) Definition eines „Rechnermodells".

- „realitätsnahes" Modell: RAM (random-access machine) (dt. Registermaschine)
  - Zugriff auf eine Speicherzelle über die Adresse in konstanter Zeit.
- „primitives" möglichst einfaches Modell: Turing-Maschine



- Die Definition von P hängt nicht vom Rechnermodell ab!

SATZ: Ein RAM-Algorithmus mit Laufzeit $T(n)$ kann auf einer Turing-Maschine mit Laufzeit $O((T(n))^3)$ simuliert werden.

(Polynomielle) Reduktion eines Problemes auf ein anderes

> Problem A lässt sich mit Hilfe eines Algorithmus für Problem B lösen.

Algorithmus bekommt Eingabe x_A für Problem A und soll Ausgabe y_A berechnen.

Der Algorithmus darf beliebig oft einen hypothetischen Hilfsalgorithmus für Problem B aufrufen:
Er stellt eine Eingabe x_B bereit, und kann die Ausgabe y_B weiterverarbeiten.

Reduktion von A auf B: $A \leq B$

$$A \leq_p B \;\wedge\; B \leq_p C \;\Rightarrow\; A \leq_p C$$

Laufzeit der Reduktion:
Laufzeit für die Aufrufe von B werden nicht mitgezählt.

Polynomielle Reduktion von A auf B: $A \leq_p B$

SATZ: $A \leq_p B$ und $B \in P \Rightarrow A \in P$

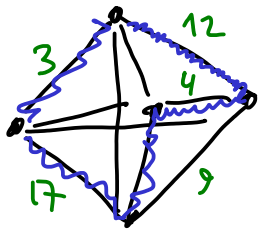
BEWEIS: T_R .. Laufzeit des Reduktionsalgorithmus R
T_B ... Laufzeit des Algorithmus für B
$|x_A| = n$. R kann den Algorithmus für B höchstens $T_R(n)$-mal aufrufen.
Die Eingabe x_B für B hat die Länge $|x_B| \leq T_R(n)$
$\Rightarrow$ Laufzeit von B: $T_B(T_R(n)) \times T_R(n)$

dazu: Laufzeit für R selbst $T_R(n) \Rightarrow T_A(n) \leq T_R(n) + T_R(n) \cdot T_B(T_R(n))$ □

Rundreiseproblem und Hamiltonkreis

Rundreiseproblem (Traveling Salesmen Problem, TSP)

EINGABE:
Ein vollständiger ungerichteter Graph G mit Kantengewichten



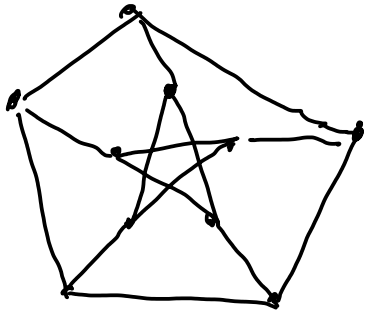
AUSGABE:
Ein Kreis, der jeden Knoten genau einmal besucht, mit kleinstem Gesamtgewicht.

Hamiltonkreis

Hamiltonkreisproblem (HAM)

EINGABE: Ein ungerichteter Graph G

FRAGE:
Enthält G einen Hamiltonkreis?



HAM $\leq_p$ TSP:

Reduktion: G_1 .. Eingabe für HAM mit n Knoten

G_2 .. vollständiger Graph mit Kantenkosten $c(e) = \begin{cases} 1 & e \in G_1 \\ 2 & e \notin G_1 \end{cases}$

Suche kürzeste Rundreise in G_2. Gesamtkosten $= n \Rightarrow$ Ausgabe JA.
$> n \Rightarrow$ Ausgabe NEIN.

polynomielle Laufzeit. ✓