1 NP

- **Berechnungsprobleme**

  - Sind alle Probleme in polynomieller Zeit lösbar? ($O(n^k)$)
  - Nein $\Rightarrow$ Manche nur in superpolynomieller Zeit lösbar
  - Polynomielle Probleme: "*einfach*"
  - Superpolynomielle Probleme: "*hart*"
- **Klasse P**
  - Klasse aller Polynomialzeitprobleme
  - Problem ist *effizient lösbar* gdw. es in polynomieller Zeit lösbar ist
  - Gilt für Polynome beliebigen Grades (auch n^k)
  - Zeitkomplexität n^k mit großem k bedenklich, jedoch fast nie notwendig
  - n beschreibt die Länge der Eingabe
  - Beispiele: Binäre Addition, Kürzeste Wege, Sortieren,...
- **Klasse NP**
  - Enthält "*einfach zu verifizierende*" Probleme (polynomieller Zeit)
  - Enthält Probleme mit "*kurzem Beweis*" (Länge polynomiell in Länge der Instanz)
  - Also: Klasse aller Probleme, deren Lösung in Polynomialzeit verifizierbar ist
  - Beispiele: Soduko, 3D-Matching,...
  - Beispiel: *Faktorisierungsproblem*
    * Jede nicht-Primzahl kann eindeutig als Primzahlprodukt geschrieben werden
    * $35 = 5 \cdot 7$, $117 = 3 \cdot 3 \cdot 13$,...
    * Faktorsieren auf klassischen Computern schwer
    * $n \longrightarrow^{schwer} p, q$
    * $n, p, q \longrightarrow^{leicht}$ ist $n = p \cdot q$?
  -  *Rucksackproblem* auch in polynomieller Laufzeit verifizierbar

- *Hamilton-Kreis-Problem*
  - Hamiltonischer Kreis: Zyklus, der alle Knoten, aber nicht unbedingt alle Kanten enthält
  - Entscheidungsalgorithmus listet alle möglichen Permutationen der Knoten aus G auf
  - Prüfung bei jeder Permutation, ob es ein Hamiltonischer Kreis ist
  - Laufzeit:
    - Kodierung via Adjazenzmatrix: m Knoten $\Rightarrow$ Matrix mit $n = m\ x\ m$ Einträgen
    - $m!$ mögliche Permutationen der Knoten
    - $\Omega(m!) = \Omega(\sqrt{n}!) = \Omega(2^{\sqrt{n}})$
    - $\Rightarrow$ superpolynomielle Laufzeit (liegt **nie** in $O(n^k)$)
  - **Allerdings:** Einfacher, wenn nur Beweis verifiziert werden muss

    $\Rightarrow$ Test, ob es sich um Permutation der Knoten handelt

    $\Rightarrow$ Test, ob alle angegebenen Kanten auf Kreis im Graphen existieren

    $\Rightarrow$ Verifikationsalgorithmus V mit quadratischer Laufzeit
  - Verifikationsalgorithmus: $V(x, y) = 1/0$ (1, falls Kreis bzw. 0, falls nicht)
  - Damit: Hamilton-Kreis $\in$ NP

- **Entscheidungsproblem vs Optimierungsproblem**
  - Optimierungsproblem: Lösung nimmt bestimmten Wert an
  - Entscheidungsproblem: Binäre Antwort (Ja/Nein)
  - Bei NP Betrachtung von Entscheidungsproblemen
  - Optimierungsproblem oft in verwandtes Entscheidungsproblem umwandelbar
  - Verwandtes Entscheidungsproblem: dem zu optimierenden Wert wird eine Schranke auferlegt

- **P versus NP**

$$L \in \mathrm{P} \longrightarrow L \in \mathrm{NP} \longrightarrow \mathrm{P} \subseteq \mathrm{NP}$$

  - Für viele wichtige Probleme ist jedoch unbekannt, ob sie in P (effizient) lösbar sind
  - Unbekannt ob $P \neq NP$
  - Intuitive Frage: Ist das Finden eines Beweises schwieriger als dessen Überprüfung?

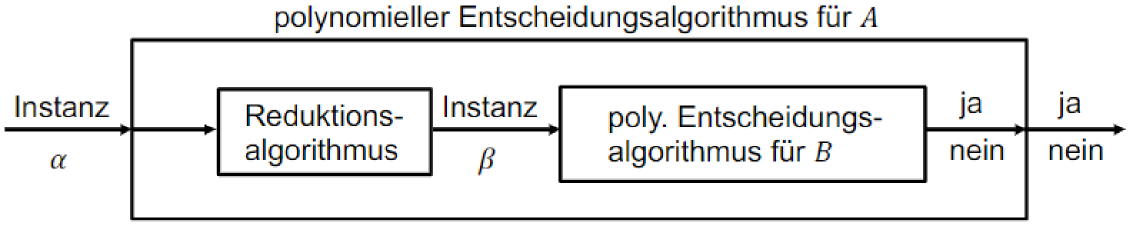
    $\Rightarrow$ Ja, also $P \neq NP$ gilt
  - In den letzten 50 Jahren kein Beweis für $P = NP$
  - Eines der wichtigsten offenen Probleme der theoretischen Informatik
  - Konsequenzen eines Beweises von $P = NP$:
    - $P = NP$: **dramatisch**, vieles bisher schwieriges einfacher lösbar (Rucksack, Kryptographie)
    - $P \neq NP$: **nicht dramatisch**, mgl. interessante Konsequenzen in Kryptographie

- **NP-Vollständigkeit**
  - Problem befindet sich in NP
  - Problem ist so "schwer" wie jedes Problem in NP
  - Beweis: Zeigen, dass kein effizienter Algorithmus existiert
  - Werkzeug: Reduktionen (zum Vergleich verschiedener Probleme)
  - *NP-Härte/NP-Schwere:*
    * Klassifikation von Problemen als schwierig, trotz fehlender genauer Zuordnung
    * *Starke Indikatoren*, dass Problem L nicht in P ist:
      · L ist mindestens so schwierig, wie alle anderen Probleme in NP
      · Daraus folgt, dass L nur in P, wenn $P = NP$ (unwahrscheinlich)
  - *Definitionen*
    * Problem L ist **NP-schwer**, wenn $L' \leq_p L$ für alle $L' \in NP$
    * Problem L ist **NP-vollständig**, wenn L sowohl NP-schwer als auch in NP ist
    * z.B.: Hamilton-Kreis ist NP-vollständig

- **Reduktionen**
  - *Reduktionsidee*
    - Betrachte Problem A, das wir in polynomieller Zeit lösen wollen
    - Bereits bekannt: Problem B (in polynomieller Zeit lösbar)
    - Benötigt wird Prozedur, die Instanzen der Probleme ineinander überführt

      $\Rightarrow$ Transformation benötigt polynomielle Zeit

      $\Rightarrow$ Antworten sind gleich



  - *Beispiel:*
    - Intuitiv: Reduktion von A auf B, wenn Umformulierung möglich

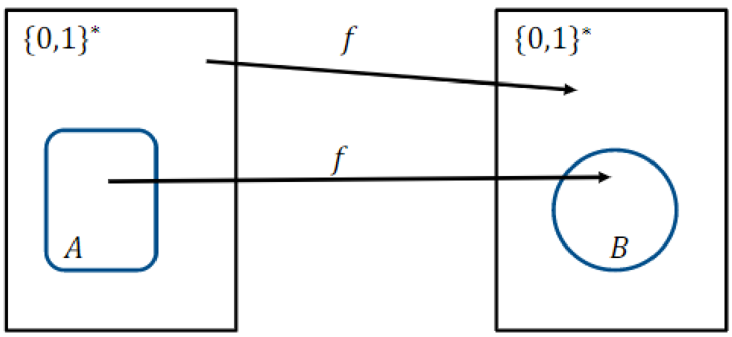
      $\Rightarrow$ Jede Instanz A kann leicht in Instanz von B umformuliert werden

      $\Rightarrow$ Lösung der Instanz B liefert Lösung von Instanz A
    - Reduktion: Lösen von linearen Gleichungen auf quadratische Gleichnungen
      - Lineare Gleichung $ax + b = 0 \Rightarrow x = \frac{-b}{a}$
      - Quadratische Gleichung $ax^2 + bx + 0 = 0 \Rightarrow x = \frac{-b}{a}, x = 0$
      - Quadratische Gleichung liefert also auch Lösung für lineare Gleichung
  - *Formale Definition:*

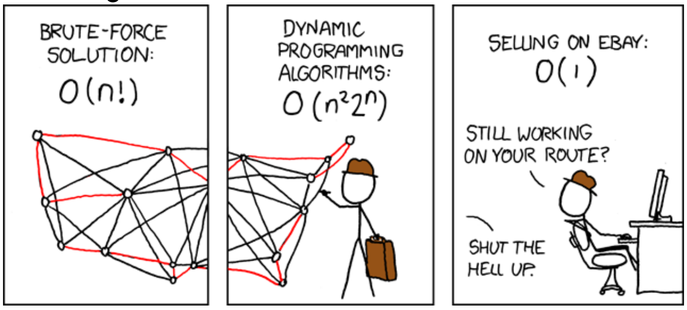
    A lässt sich auf B in **polynomieller Zeit reduzieren**, mit Schreibweise $A \leq_p B$, wenn eine in polynomieller Zeit berechenbare Funktion $f : \{0,1\}^* \to \{0,1\}^*$ existiert, sodass für alle $x \in \{0,1\}^*$ gilt:

    $x \in A$ genau dann, wenn $f(x) \in B$

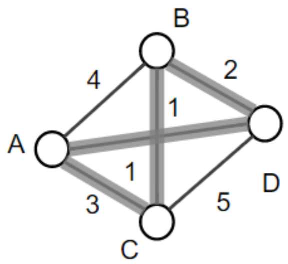
Illustration der Polynomialzeitreduktion:



- **Travelling-Salesman Problem**



  - *Beschreibung*
    * Reisender plant Rundreise durch mehrere Städte
    * Start und Ziel ist eine vorgegebene Stadt
    * Jede Stadt nur einmal besuchen
    * Ziel: Minimale Reisekosten



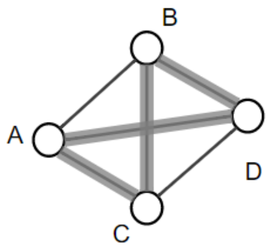
Eine optimale Route mit Kosten 7 verläuft von A → D → B → C → A

  - *Problem:*
    * Anzahl der verschiedenen Rundreisen $(n-1)!$
    * Stark nach oben explodierende Zahlen
    * Brute-Force für große n praktisch unmöglich
    * Es existiert kein effizienter Algorithmus, der das TSP effizient löst
    * TSP ist **NP-vollständig**
  - *Beweis NP-Vollständigkeit*
    * Zeigen: TSP gehört zu NP und TSP ist NP-schwer
  - *TSP gehört zu NP*
    * Gegeben: Instanz des Problems TSP, Folge der n Knoten der Tour (Zertifikat)
    * Verifikationsalgorithmus überprüft, ob Folge jeden Knoten genau einmal enthält
    * Außerdem Aufsummieren der Kantenkosten und überprüfen, ob diese maximal k ist
    * Verifikation läuft in polynomieller Laufzeit ⇒ gehört zu NP

– *TSP ist NP-schwer*

  * Wir zeigen $HAM - KREIS \leq_p TSP$
  * Start: Instanz von $HAM - KREIS$ mit $G = (V, E)$
  * Konstruiere Instanz von TSP

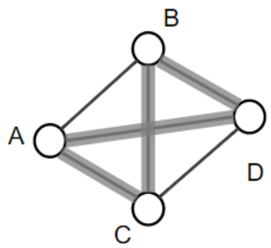
    $\Rightarrow G' = (V, E')$ mit $E' = \{(i, j) : i, j \in V$ und $i \neq j\}$
  * Definiere Kostenfunktion $c(i, j) = 0$, falls $(i, j) \in E$ / $c(i, j) = 1$, falls $(i, j) \notin E$
  * Instanz von TSP ist $< G', c, 0 >$ (Konstruktion in polynomieller Zeit) (0: Kosten von 0)
  * **Zeige jetzt:** G besitzt hamiltonischen Kreis $\Leftrightarrow$ G' enthält Tour mit Kosten ≤ 0
  * $\Rightarrow$ Graph G besitzt einen hamiltonischen Kreis h



Jede Kante von h gehört zu E und daher besitzt laut Kostenfunktion der Graph G' die Kosten 0.

Damit ist h eine Tour in G' mit den Kosten 0.

  * $\Leftarrow$ Graph G besitzt eine Tour h' mit Kosten kleiner gleich 0



Die Kosten der Kanten in E' haben die Werte 0 und 1. Die Kosten der Tour betragen exakt 0 und jede Kante muss die Kosten 0 haben.

Damit hat h' nur Kanten von E.

Damit folgt, dass h' ein Hamiltonischer Kreis des Graphen G ist.