Die Klassen P und NP

P enthält die Probleme, die mit herkömmlichen Computern zeitgerecht lösbar sind

NP enthält eine Vielzahl praktisch relevanter Probleme (wie Tourenplanung, Maschinenbelegung, Stundenplanung, Lagerhaltung, . . .)

Definition von NP und P

$dp \in NP$, falls eine Lösung sol und ein $k \in \mathbb{N}$ existieren mit $T^{sol} \in O(n^k)$

$dp \in P$, falls zusätzlich für alle $w \in A^*$ gilt:

$$dp(w) = T \text{ und } sol(w) \overset{*}{\longleftrightarrow} t \text{ impl. } t \overset{*}{\longleftrightarrow} T$$

offensichtlich: $P \subseteq NP$

offen: $NP \subseteq P$

Beispiel: Little-Solitaire

solitaire

opns: $solitaire, won\colon \{0,1\}^* \to BOOL$

$move\colon \{0,1\}^* \to \{0,1\}^*$

vars: $u, v, w \in \{0,1\}^*$

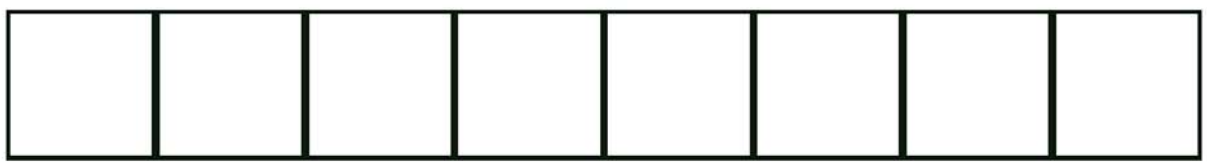
eqns: $solitaire(w) = won(move(w))$

$won(w) = count(1, w) = 1$

$move(u110v) = move(u001v)$

$move(u011v) = move(u100v)$

$move(w) = w$



Definition von NP und P

$dp \in NP$, falls eine Lösung sol und ein $k \in \mathbb{N}$ existieren mit $T^{sol} \in O(n^k)$

$dp \in P$, falls zusätzlich für alle $w \in A^*$ gilt:

$$dp(w) = T \text{ und } sol(w) \overset{*}{\longleftrightarrow} t \text{ impl. } t \overset{*}{\longleftrightarrow} T$$

offensichtlich: $P \subseteq NP$

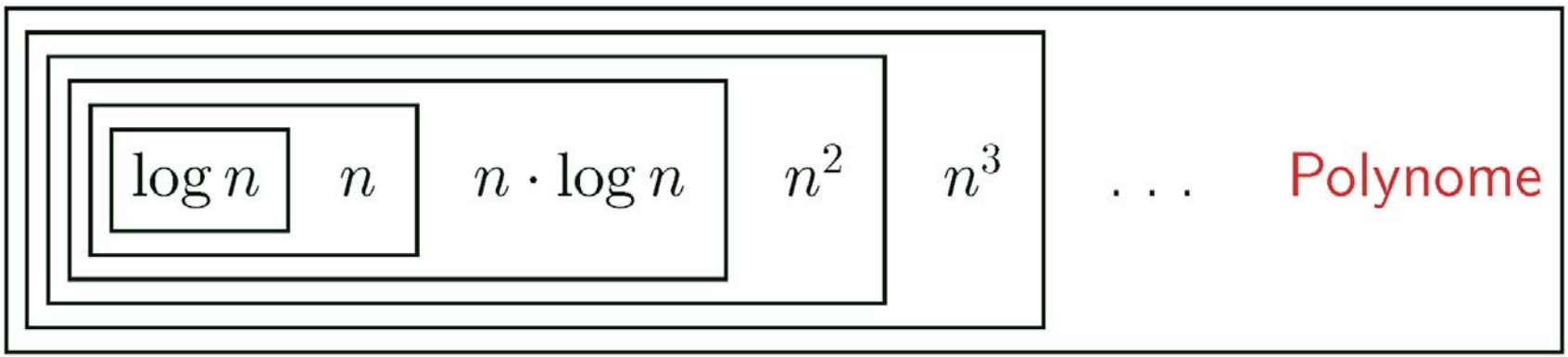
offen: $NP \subseteq P$

Beispiele

Problem aus P	Aufwand
Suchen in balancierten Bäumen	$\log n$
Suchen in Wörtern, Transponieren, Zählen, Filtern	n
Sortieren durch Mischen	$n \cdot \log n$
Sortieren durch Einsortieren, *quicksort*	n^2
Matrizenmultiplikation, Wortproblem kontextfreier Sprachen	n^3

Polynomieller Aufwand

- Probleme mit polynomiellem Aufwand können mit herkömmlichen Rechnern in verfügbarer Zeit gelöst werden (wenn der Exponent nicht zu groß ist).



Anmerkungen zu P

- Die meisten praktisch interessanten Probleme in P lassen sich in $O(n^3)$ Schritten oder schneller lösen.
- Das Laufzeitverhalten polynomieller Algorithmen lässt sich häufig weiter verbessern (Beispiel: Matrizenmultiplikation).

- Wenn jede Gleichungsanwendung in CE-S auf dem Computer polynomiellen Aufwand hat, können wir Gleichungsanwendung als konstant zählen, ohne dabei die Klasse P zu verlassen.

  Aber:

  Es kann sich bei einer CE-S-Spezifikation ein anderer Aufwand ergeben (als bei einem entsprechenden Computer-Programm), wenn der Aufwand einer Gleichungsanwendung nicht konstant ist.

- Gleichungsanwendungen haben nicht immer konstanten Aufwand.

Ein Problem in NP

- SAT (Erfüllbarkeitsproblem der Aussagenlogik)
  - Eingabe: Aussagenlogische Formel f.
  - Ausgabe: T gdw eine Belegung der Variablen in f mit 1 oder 0 existiert, so dass f gilt.
  - Lösung 1: Probiere alle 2^k Belegungen $\in O(2^k)$
  - Lösung 2: Rate richtige Belegung $\in O(k)$

Anwendungsgebiete: Entwurf und Analyse von Schaltkreisen, Model Checkers, Robotik, . . .

Beispiel

Ein Gerät mit vier $0/1$-Schaltern befindet sich bei folgenden Schalterstellungen in einem betriebssicheren Zustand:

1. Wenn A und B gleich 0, dann C gleich 1
2. A oder C gleich 1
3. A, B, C gleich 1 oder B, C, D gleich 1 oder A, D gleich 1

Anmerkungen zu NP

- Zur Lösung von Problemen aus NP muss meist eine Anzahl von $Lösungskandidaten$ durchsucht werden.
- Diese Anzahl steigt mit wachsender Eingabe exponentiell an.
- Ein nichtdeterministischer Algorithmus löst diese Probleme in polynomieller Zeit durch Raten einer richtigen Lösung.
- Die Überprüfung, ob ein gegebener $Lösungskandidat$ eine korrekte Lösung ist, kann mit einem deterministischen Algorithmus in $O(n^k)$ Schritten gelöst werden.
- Es wird allgemein vermutet, dass NP größer als P ist.

Probleme in NP

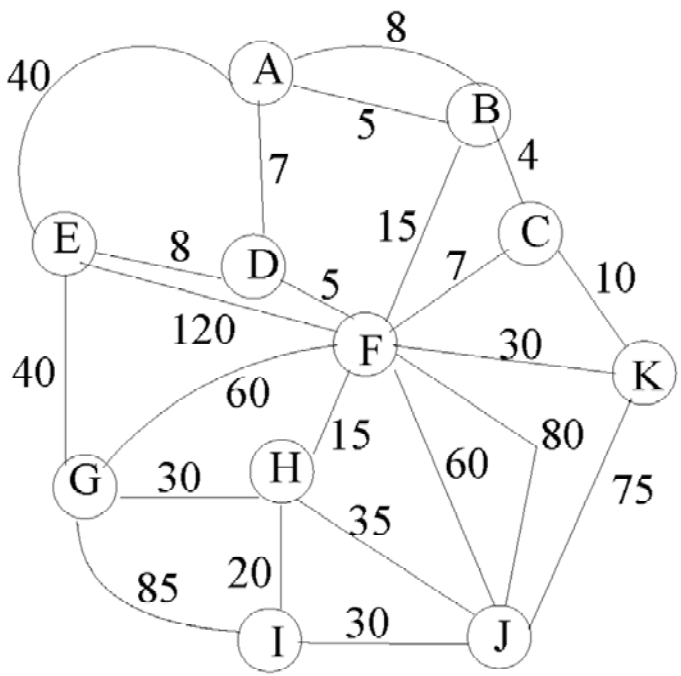
- 3SAT (Spezialfall von SAT)
  - Eingabe: Aussagenlogische Formel f der Form

$$c_1 \wedge c_2 \wedge \cdots \wedge c_n,$$

    so dass
    - c_i: Klausel der Form $L_1 \vee L_2 \vee L_3$,
    - L_j: Literal der Form x oder $\neg x$
    - x: Variable.
  - Ausgabe: T gdw eine Belegung der Variablen in f mit 1 oder 0 existiert, so dass f gilt.

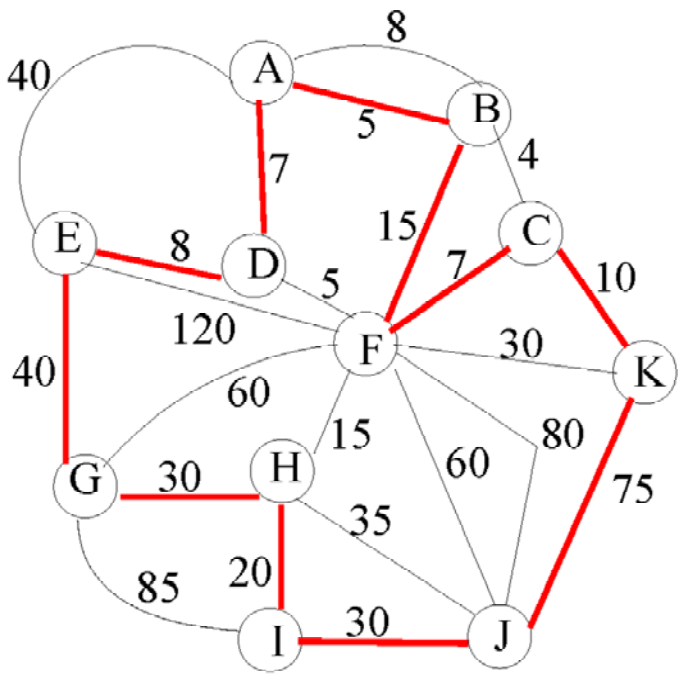
- TSP (Traveling Salesperson Problemm, Entscheidungsproblemvariante)
  - Eingabe: Ungerichteter Graph G mit natürlichen Zahlen als Kantenmarkierungen und eine Zahl k.

    Beispiel für G:



- Ausgabe: T gdw G einen Rundweg besitzt, der jeden Knoten genau einmal besucht und höchstens k lang ist.

  Beispiel: T für k = 250



Rundweg: 247

Beweis der NP-Vollständigkeit eines Problems dp mittels Reduktion

Theorem

dp ist NP-vollständig.

Beweis:

1. Zeige, dass dp in NP liegt.
2. Wähle ein NP-vollständiges Problem und reduziere es auf dp.

Anmerkung: Ist nur möglich, wenn man bereits ein NP-vollständiges Problem hat.

NP-vollständige Probleme

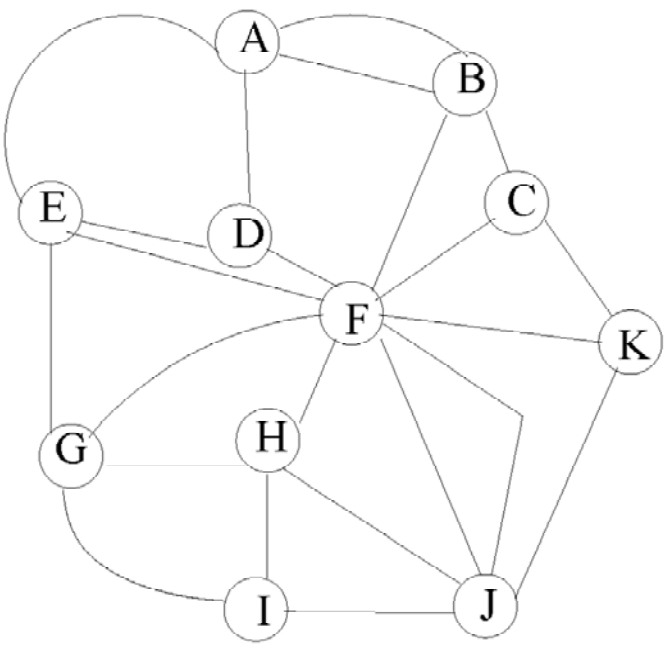
Theorem (Cook und Levin)
Das Erfüllbarkeitsproblem der Aussagenlogik ist NP-vollständig.
Beweisidee: Reduziere jedes Problem aus NP auf SAT.

Theorem
3SAT ist NP-vollständig.
Beweisidee: Reduziere SAT auf 3SAT.

HC

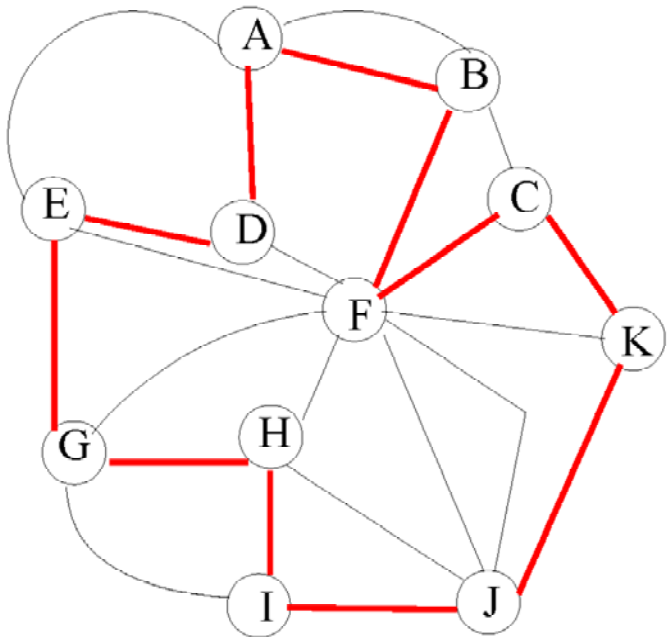
- HC (Hamiltonian Circuit Problem)
  - Eingabe: Ein ungerichteter Graph G.

    Beispiel:



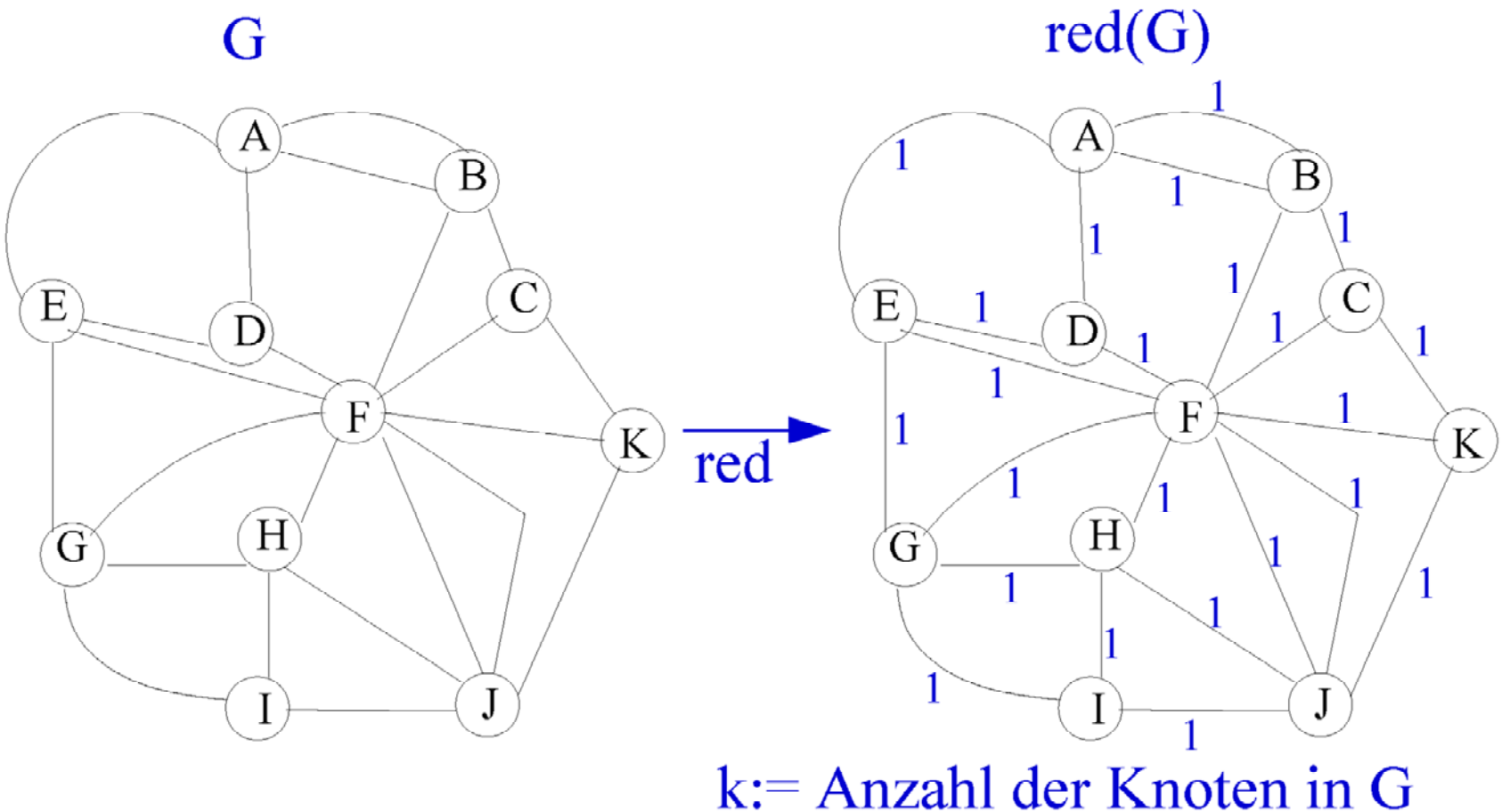
- Ausgabe: T gdw G einen Rundweg besitzt, der jeden Knoten genau einmal besucht.

  Beispiel:



Beobachtung: HC ist in NP.

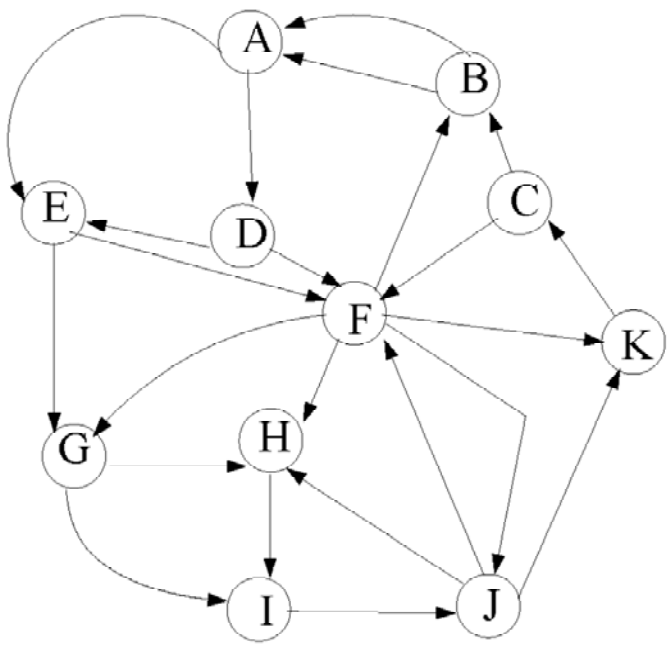
Reduktion von HC auf TSP



DHC

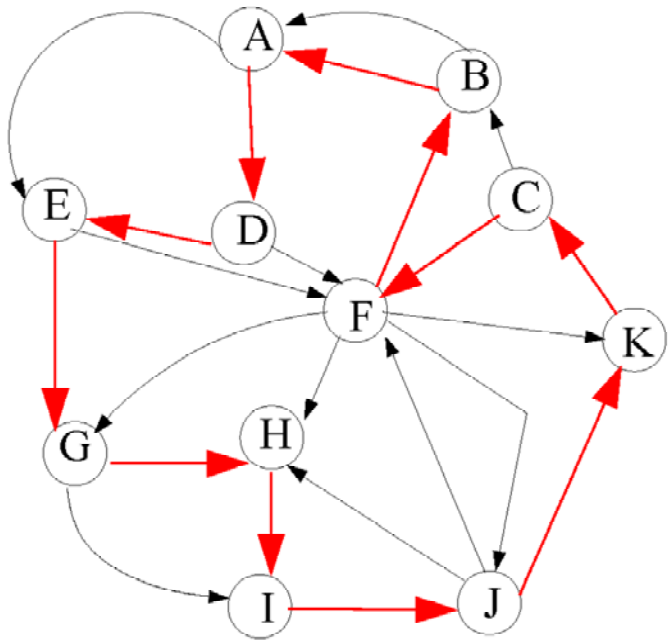
- DHC (Directed Hamiltonian Circuit Problem)
  - Eingabe: Ein gerichteter Graph G.

    Beispiel:



- Ausgabe: T gdw G einen Rundweg (in Pfeilrichtung) besitzt, der jeden Knoten genau einmal besucht.

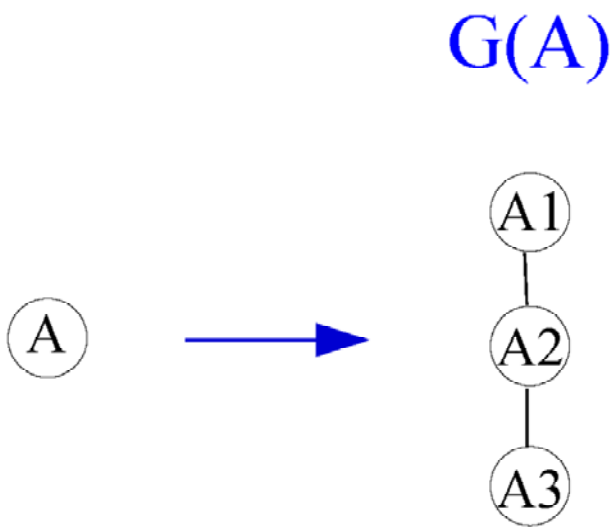
  Beispiel:



Beobachtung: DHC ist in NP.

Reduktion von DHC auf HC

1. Für jeden Knoten in G konstruiere drei miteinander verbundene Knoten wie folgt:



2. Für jede Kante von A nach B ziehe eine Kante in $red(G)$ von $A3$ nach $B1$.

Skizze