Hochschule RheinMain

Automatentheorie und Formale Sprachen

für die Studiengänge
- Angewandte Informatik
- Informatik - Technische Systeme

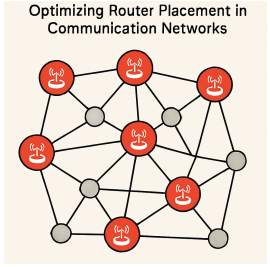
10 Komplexitätstheorie

Prof. Dr. David Sabel
Sommersemester 2025

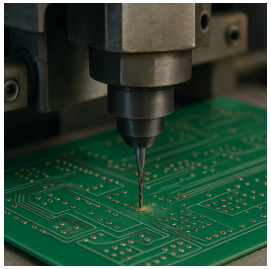
Stand der Folien: 9. Juli 2025

Motivation: Beispiele aus der Praxis

Hochschule RheinMain



Router-Platzieren:
Minimale Anzahl an Knoten für die Router finden



Löcherbohren in Leiterplatten:
Finde kürzeste „Rundreise“ für den Bohrer

Playlist erstellen:
X Minuten mit Songs füllen.

Fragen:
- Können wir solche Probleme (effizient) lösen?
- Was bedeutet effizient? Was machen wir, wenn nicht?

Komplexitätstheorie

Hochschule RheinMain

Was ist Komplexitätstheorie?

- Teilgebiet der Theoretischen Informatik
- Grobes Thema: Komplexität von **entscheidbaren Problemen**
- Maße zum Messen des Ressourcenbedarfs von Algorithmen:
  - Rechenzeit
  - Platzbedarf
  - ...
- Komplexität eines Problems
  = Komplexität des besten Algorithmus bezüglich des Maßes
- Ziel: Einordnung von Problemen in Komplexitätsklassen

Komplexitätstheorie: Teilbereiche

Hochschule RheinMain

1. Nachweis von oberen Schranken:
   - Finde möglichst guten Algorithmus für konkretes Problem
   - Analysieren der Laufzeit- und Platzkomplexität
   - z.B.: Der CYK-Algorithmus liefert obere Schranke $O(|P| \cdot n^3)$ für das Wortproblem für CFGs (mit n = Wortlänge, P Produktionen der CFG)
   - Möglichst genaue Schranken (bezüglich der O-Notation)

Komplexitätstheorie: Teilbereiche (2)

Hochschule RheinMain

2. Nachweis von unteren Schranken:
   - Zeige, dass es keinen besseren Algorithmus gibt.
   - Schwieriger, da man über alle Algorithmen argumentieren muss.
   - Möglichst genaue Schranken (bezüglich der Ω-Notation)
   - Oft ist $\Omega(n)$ ($n =$ Größe der Eingabe) eine untere Schranke für die Laufzeit, da man die Eingabe lesen muss.

Komplexitätstheorie: Teilbereiche (3)

Hochschule RheinMain

3. Auswirkung der **Maschinenmodelle auf den Ressourcenbedarf**
   - insbesondere **Determinismus vs. Nichtdeterminismus**
   - Wichtige ungelöste Frage: Das $\mathcal{P}$ **vs.** $\mathcal{NP}$**-Problem**
   - Komplexitätsklassen sind i.A. ungenauer (größer) als in den vorher genannten Teilbereichen
   - **Wir beschäftigen uns in diesem Abschnitt hiermit.**

Zeitkomplexität: Annahmen und Festlegungen

Hochschule RheinMain

Turingmaschinen:

- Wir betrachten nur **entscheidbare** Sprachen
- Deswegen nehmen wir an:

  Die Turingmaschinen, welche diese Sprachen entscheiden, **halten auf jeder Eingabe an**:
- DTMs: Wir nehmen an, dass es „Verwerfe“-Zustände gibt, für die die TM keine Nachfolgekonfiguration besitzen.
- NTMs: Haben solche Zustände von Haus aus

Zeitkomplexität: Annahmen und Festlegungen (2)

Hochschule RheinMain

Mehrband- vs. Einband-Turingmaschinen:

- Wir nehmen Mehrband-TMs, sie passen zu „normalen“ Rechnern.
- Kein „Vergeuden“ von Rechenzeit, nur um die Eingaben zu suchen.
- Unterschied:
  n-Schritte auf Mehrband-TM können in
  $O(n^2)$ Schritten auf Einband-TM ausgeführt werden
- Unterschied macht sich in Komplexitätsklassen $\mathcal{P}$ und $\mathcal{NP}$ **nicht** bemerkbar (siehe später)

Deterministische Laufzeit

Hochschule RheinMain

Definition ($time_M$)

Sei M eine stets anhaltende DTM mit Startzustand z_0.
Für Eingabe w definieren wir

$time_M(w) := i$, wenn $z_0 w \vdash_M^i uzw$ und
z ist Endzustand oder Verwerfe-Zustand.

Definition (Klasse $TIME(f(n))$)

Für eine Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ sei die Klasse $TIME(f(n))$ genau die Menge der Sprachen, für die es eine deterministische, stets anhaltende, Mehrband-TM M gibt, mit $L(M) = L$ und $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$

Sprache ist daher in $TIME(f(n))$, wenn sie von einer DTM für jede Eingabe der Länge n in $\leq f(n)$ Schritten entschieden wird.

DIE KLASSE $\mathcal{P}$

Hochschule RheinMain

Die Klasse $\mathcal{P}$

Hochschule RheinMain

Definition (Polynom)

Ein Polynom ist eine Funktion $p : \mathbb{N}_0 \to \mathbb{N}_0$ der Form:

$$p(n) = \sum_{i=0}^{k} a_i \cdot n^i = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

mit $a_i \in \mathbb{N}_0$ und $k \in \mathbb{N}_0$.

Definition (Komplexitätsklasse $\mathcal{P}$)

Die Klasse $\mathcal{P}$ ist definiert als

$$\mathcal{P} = \bigcup_{p \text{ Polynom}} TIME(p(n))$$

Nochmal: Zugehörigkeit zu $\mathcal{P}$

Hochschule RheinMain

Formale Sprache L ist in Klasse $\mathcal{P}$ enthalten, wenn:

Es gibt stets anhaltende DTM M und ein Polynom p mit:

- $L = L(M)$
- $\forall w \in \Sigma^* : time_M(w) \leq p(|w|)$

Polynomielle Komplexität

Hochschule RheinMain

Nachweis der polynomiellen Komplexität:

Zeige, dass die Komplexität $O(n^k)$ für Konstante k ist.

Beispiele: Algorithmus mit Laufzeit

- $n \log n$ hat polynomielle Komplexität, da $n \log n \in O(n^2)$
- 2^n hat keine polynomielle Komplexität
- $n^{\log n}$ hat keine polynomielle Komplexität

Sprechweisen:

- Algorithmus ist effizient = Algorithmus hat polynomielle Komplexität
- Analog: Problem ist effizient lösbar = Problem in $\mathcal{P}$

Erinnerung: Klausuranmeldung

Hochschule RheinMain

- bis zum 07.07.2025 im COMPASS
- PDF-Dokument als Nachweis erstellen und aufheben!

Quiz 1

Hochschule RheinMain

Für was steht das P der Komplexitätsklasse $\mathcal{P}$?

- Entscheidbare Probleme
- Unentscheidbare Probleme
- Polynomielle Zeit
- Polynomieller Platz

arsnova.hs-rm.de
6750 1376

Wiederholung: Deterministische Laufzeit und $\mathcal{P}$

Hochschule RheinMain

Komplexitätsklasse $\mathcal{P}$

Die Klasse $\mathcal{P}$ ist definiert als

$$\mathcal{P} = \bigcup_{p \text{ Polynom}} TIME(p(n))$$

- $time_M(w) := i$, wenn $z_0 w \vdash_M^i uzw$ und z End- oder Verwerfe-Zustand
- für $f : \mathbb{N}_0 \to \mathbb{N}_0$ ist $TIME(f(n))$ die Menge der Sprachen, die durch DTM M entscheidbar und $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$

Hochschule RheinMain

DIE KLASSE $\mathcal{NP}$

Nichtdeterminismus

Auf für NTMs nehmen wir an, dass sie auf allen Berechnungspfaden anhalten.
Beachte: Ein Wort wird **nicht akzeptiert**, wenn auf allen Berechnungspfaden verworfen wird

Definition ($ntime_M$)

Sei M eine Mehrband-NTM, die für jede Eingabe anhält. Dann definieren wir die Laufzeit von M als

$$ntime_M(w) = \max\{i \mid z_0w \vdash_M^i uzw \text{ und } uzw \nvdash \ldots\}$$

Beachte: Schöning verwendet eine andere Definition (mit Minimum), es macht für die Definition der Klasse $\mathcal{NP}$ aber keinen Unterschied.

$NTIME$ und $\mathcal{NP}$

Definition

Für eine Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ bezeichne $NTIME(f(n))$ die Klasse aller Sprachen L, für die es eine nichtdeterministische Mehrband-TM M gibt mit $L(M) = L$ und für alle $w \in \Sigma^*$ gilt $ntime_M(w) \leq f(|w|)$.

Definition (Klasse $\mathcal{NP}$)

Die Klasse $\mathcal{NP}$ ist definiert als

$$\mathcal{NP} = \bigcup_{p \text{ Polynom}} NTIME(p(n))$$

$\mathcal{P} \subseteq \mathcal{NP}$

Lemma

Es gilt $TIME(f(n)) \subseteq NTIME(f(n))$ und damit auch $\mathcal{P} \subseteq \mathcal{NP}$.

Beweis:

- DTM als NTM auffassen
- ergibt NTM mit einem möglichen Berechnungspfad
- $time_M(w) = ntime_M(w)$
- $L \in TIME(f(n)) \implies L \in NTIME(f(n))$
- $\mathcal{P} \subseteq \mathcal{NP}$

$\mathcal{P}$-vs.-$\mathcal{NP}$

Hochschule RheinMain

Die Frage

Gilt $\mathcal{P} = \mathcal{NP}$ oder $\mathcal{P} \neq \mathcal{NP}$?

ist **ungelöst**!

- Das $\mathcal{P}$-vs-$\mathcal{NP}$-Problem ist eines der sieben sogenannten Millennium-Probleme, die vom Clay Mathematics Institute im Jahr 2000 als Liste ungelöster Probleme der Mathematik herausgegeben wurde
- Für dessen Lösung wurde ein Preisgeld von einer Million US Dollar ausgelobt

$\mathcal{P}$-vs.-$\mathcal{NP}$ (2)

Hochschule RheinMain

Wesentlicher Grund, der für $\mathcal{P} \neq \mathcal{NP}$ spricht:

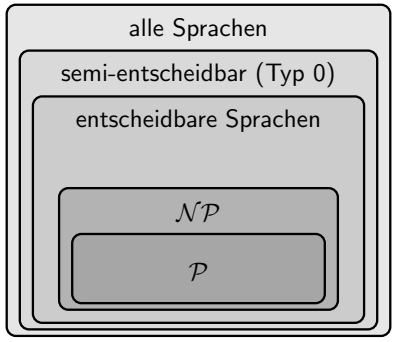
> Man müsste für $\mathcal{P} = \mathcal{NP}$ einen Polynomialzeitalgorithmus finden, für ein Problem, für das bisher nur (deterministische) Exponentialzeitalgorithmen bekannt sind. (*)

Viele haben gesucht, keiner hat einen solchen Algorithmus gefunden.

(*) Begründung: Folgt später

Lage der Komplexitätsklasse (Schema)

Hochschule RheinMain



Dabei unklar, ob $\mathcal{NP} = \mathcal{P}$

Quiz 2

Hochschule RheinMain

Für was steht das NP der Komplexitätsklasse $\mathcal{NP}$?

- Nicht in polynomieller Zeit lösbar
- Nur polynomielle Laufzeit
- Nicht polynomieller Platzbedarf
- Nichtdeterministisch polynomielle Laufzeit
- Nichtdeterministisch polynomieller Platzbedarf

arsnova.hs-rm.de
6750 1376

NP-VOLLSTÄNDIGKEIT

Hochschule RheinMain

Polynomialzeit-Reduktion

Hochschule RheinMain

Definition (Polynomialzeit-Reduktion (einer Sprache auf eine andere))

Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen.
Dann sagen wir L_1 ist auf L_2 polynomiell reduzierbar (geschrieben $L_1 \leq_p L_2$),
falls es eine totale und in deterministischer Polynomialzeit berechenbare Funktion
$f : \Sigma_1^* \to \Sigma_2^*$ gibt, sodass für alle $w \in \Sigma_1^*$ gilt: $w \in L_1 \iff f(w) \in L_2$.

Analogie zu Reduktionen in der Berechenbarkeitstheorie $L_1 \leq L_2$:

Zusatz hier: Polynomialzeit!
Nächste Analogie:

Berechenbarkeitstheorie:	Komplexitätstheorie:
$L_1 \leq L_2$ und L_2 (semi-) entscheidbar	$L_1 \leq_p L_2$ und $L_2 \in (\mathcal{N})\mathcal{P}$
$\implies$ L_1 (semi-) entscheidbar	$\implies$ $L_1 \in (\mathcal{N})\mathcal{P}$

Polynomialzeit-Reduktion (Eigenschaften)

Hochschule RheinMain

Lemma

Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{P}$, dann gilt $L_1 \in \mathcal{P}$.
Ebenso gilt: Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{NP}$, dann gilt $L_1 \in \mathcal{NP}$.

Beweis (für $\mathcal{P}$):

- Sei $L_1 \leq_p L_2$ und f in Polynomialzeit berechenbar
- Sei M_f die DTM, die f in Polynomialzeit durch p beschränkt berechnet.
- Sei $L_2 \in \mathcal{P}$, $L(M_2) = L_2$, wobei Schritte von M_2 auf $w \leq q(|w|)$
- Sei $M_f; M_2$ die Hintereinanderausführung von M_f und M_2. Dann gilt: $L(M_f; M_2) = L_1$
- $M_f; M_2$ hält stets in Polynomialzeit: $|f(w)| \leq |w| + p(|w|)$ (da M_f nicht mehr in $p(|w|)$-Schritten schreiben kann)
  $M_f; M_2$ macht höchstens $r(|w|) := p(|w|) + q(|w| + p(|w|))$ Schritte
- Es gilt $L_1 \in \mathcal{P}$.

Polynomialzeit-Reduktion (Eigenschaften) (2)

Hochschule RheinMain

Lemma

Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{P}$, dann gilt $L_1 \in \mathcal{P}$.
Ebenso gilt: Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{NP}$, dann gilt $L_1 \in \mathcal{NP}$.

Beweis (für $\mathcal{NP}$): Analog

Polynomialzeit-Reduktion (Eigenschaften) (3)

Hochschule RheinMain

Lemma

Die Relation $\leq_p$ ist transitiv, d.h. wenn $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$, dann gilt auch $L_1 \leq_p L_3$

Beweis: Komposition von zwei Polynomen bleibt Polynom.

$\mathcal{NP}$-Vollständigkeit

Hochschule RheinMain

Definition ($\mathcal{NP}$-Vollständigkeit)

Eine Sprache L heißt $\mathcal{NP}$-vollständig, wenn gilt

1. $L \in \mathcal{NP}$ und
2. L ist $\mathcal{NP}$-schwer (manchmal auch $\mathcal{NP}$-hart genannt):
   Für alle $L' \in \mathcal{NP}$ gilt $L' \leq_p L$

$\mathcal{NP}$-vollständige Probleme sind die schwierigsten Probleme in $\mathcal{NP}$:

$\mathcal{NP}$-Schwere besagt, dass man mit dem $\mathcal{NP}$-vollständigen Problem alle anderen Probleme aus $\mathcal{NP}$ (in zusätzlicher deterministischer Polynomialzeit) lösen kann

$\mathcal{NP}$-Vollständigkeit beweisen

Hochschule RheinMain

Nachweis der $\mathcal{NP}$-Vollständigkeit von L

- Zugehörigkeit zu $\mathcal{NP}$: Gebe polynomiell Laufzeit-beschränkte NTM an, die L entscheidet (alternativ: Polynomialzeitreduktion von $L \leq_p L_1$ mit $L_1 \in \mathcal{NP}$)
- **$\mathcal{NP}$-Schwere:** Statt jedes mal neu zu beweisen, dass alle Probleme aus $\mathcal{NP}$ auf L polynomiell reduzierbar, wähle ein $\mathcal{NP}$-schweres Problem L_0 und zeige $L_0 \leq_p L$.

  Dann folgt L ist $\mathcal{NP}$-schwer:

  Da L_0 $\mathcal{NP}$-schwer, gilt $L' \leq_p L_0$ für alle $L \in \mathcal{NP}$ und damit $L' \leq_p L_0 \leq_p L$ und mit Transitivität von $\leq_p$: $L' \leq_p L$ für alle $L' \in \mathcal{NP}$.

$\mathcal{NP}$-Schwere vs. Unentscheidbarkeit

Hochschule RheinMain

Beweis der $\mathcal{NP}$-Schwere ist komplett analog zum Vorgehen wie bei der Unentscheidbarkeit, wesentlicher Unterschied: Polynomialzeit-Reduktion:

Berechenbarkeitstheorie:	Komplexitätstheorie:
$L_0 \leq L$ und L_0 unentscheidbar	$L_0 \leq_p L$ und L_0 $\mathcal{NP}$-schwer
$\Longrightarrow$ L unentscheidbar	$\Longrightarrow$ L $\mathcal{NP}$-schwer

Wenn ein $\mathcal{NP}$-vollständiges Problem in $\mathcal{P}$ liegt, ...

Hochschule RheinMain

Satz

Sei L ein $\mathcal{NP}$-vollständiges Problem. Dann gilt $L \in \mathcal{P} \iff \mathcal{P} = \mathcal{NP}$.

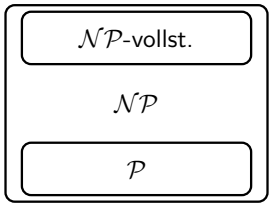
Beweis:

- Sei L $\mathcal{NP}$-vollständig und $L \in \mathcal{P}$
- Aus $\mathcal{NP}$-Schwere von L folgt:
  Für alle $L' \in \mathcal{NP}$: $L' \leq_p L$ und damit $L' \in \mathcal{P}$.
- Da dies für alle $L' \in \mathcal{NP}$ gilt, folgt $\mathcal{P} = \mathcal{NP}$.

Also: Es reicht aus für ein $\mathcal{NP}$-vollständiges Problem nachzuweisen, dass es in $\mathcal{P}$ bzw. nicht in $\mathcal{P}$ liegt, um die $\mathcal{P}$-vs-$\mathcal{NP}$-Frage ein für allemal beantworten.

Vermutete Lage der Probleme

Hochschule RheinMain



Es ist bekannt (Ladner 1975):
Unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ gibt es Probleme in $\mathcal{NP}$ gibt, die nicht in $\mathcal{P}$ liegen und nicht $\mathcal{NP}$-vollständig sind.

Möglicher Kandidat:
Graph-Isomorphismus-Problem. Weder ein polynomieller Algorithmus noch dessen $\mathcal{NP}$-Vollständigkeit bekannt.

Ausblick

Hochschule RheinMain

Was fehlt noch?

Ein erstes Problem L_0, dass man direkt als $\mathcal{NP}$-vollständig beweist.

Danach kann man $\mathcal{NP}$-Vollständigkeit von L zeigen durch:

- $L \in \mathcal{NP}$
- $L_0 \leq_p L$

Danach: Lerne einen Satz an $\mathcal{NP}$-vollständigen Problemen kennen.

Das SAT-Problem

Hochschule RheinMain

Definition (SAT-Problem)

Das Erfüllbarkeitsproblem der Aussagenlogik (kurz SAT) ist:

gegeben: Eine Aussagenlogische Formel F
gefragt: Ist F erfüllbar, d.h. gibt es eine erfüllende Belegung der Variablen mit den Wahrheitswerten 0 und 1, sodass F den Wert 1 erhält.

Als formale Sprache:
SAT $= \{code(F) \in \Sigma^* \mid F \text{ ist erfüllbare Formel der Aussagenlogik}\}$

SAT $\in \mathcal{NP}$

Hochschule RheinMain

Lemma

SAT $\in \mathcal{NP}$

Beweis:

- $code(F)$ Eingabe einer NTM M
- M berechnet, welche Variablen in F vorkommen. Sei dies $\{x_1, \ldots, x_n\}$.
- M verwendet Nichtdeterminismus, um Belegung zu „raten“:

$$I : \{x_1, \ldots, x_n\} \to \{0, 1\}$$

- Jede der 2^n nicht-deterministischen Berechnungen berechnet Wert von $I(F)$
- Akzeptanz bei 1, sonst verwerfen.
- Da jede Belegung überprüft wird, gilt:

M akzeptiert eine Formel F g.d.w. $F \in$ SAT

- Jeder Berechnungspfad von M läuft in Polynomialzeit in $|code(F)|$, da Anzahl der Variablen durch die Eingabegröße beschränkt ist.

$\mathcal{NP}$ = Polynomiell verifizierbar

Hochschule RheinMain

- Nachweis, dass Sprache in $\mathcal{NP}$ liegt, geht oft so wie bei SAT
- Verwende Nichtdeterminismus, um potentielle Lösung zu raten
- Zeige, dass eine Lösung in Polynomialzeit verifiziert werden kann

Satz von Cook

Hochschule RheinMain

Satz von Cook

Das Erfüllbarkeitsproblem der Aussagenlogik ist $\mathcal{NP}$-vollständig.

Beweis: siehe Literatur

Beweisidee: Reduziere jedes $\mathcal{NP}$-Problem L auf SAT mit Polynomialzeitreduktion:

Für NTM M_L, die L in polynomieller Zeit entscheidet und Eingabe w wird eine aussagenlogische Formel $F_{M_L,w}$ erzeugt mit

$F_{M_L,w}$ erfüllbar genau dann, wenn M_L akzeptiert w.

3-CNF-SAT

Hochschule RheinMain

Konjunktive Normalform

Aussagenlogische Formel F ist in konjunktiver Normalform (CNF), wenn:

$$F = \bigwedge_{i=1}^{m} (\bigvee_{j=1}^{n_i} L_{i,j})$$

- Literal $L_{i,j}$ ist aussagenlogische Variable oder deren Negation
- $(\bigvee_{j=1}^{n_i} L_{i,j})$ ist eine Klausel
- Erfüllende Belegung muss ≥ 1 Literal pro Klausel wahr machen.

Klauselmengenschreibweise: $\{\{L_{1,1}, \ldots, L_{1,n_1}\}, \ldots, \{L_{m,1}, \ldots, L_{m,n_m}\}\}$

Annahme: Keine Klauseln, die x und $\neg x$ enthalten
Diese Klauseln sind immer wahr und können gelöscht werden.

Konjunktive Normalform

Hochschule RheinMain

Jede aussagenlogische Formel kann in eine **äquivalente** konjunktive Normalform gebracht werden:

- $\iff$, $\implies$ auflösen
- Negationen nach innen schieben und anschließend Ausmultiplizieren (Distributivität, Kommutativität anwenden, um konjunktive Normalform herzustellen)

Aber:

- Algorithmus hat im worst case **exponentielle Laufzeit**
- Algorithmus kann daher **nicht** für eine **Polynomialzeitreduktion** verwendet werden.

3-CNF-SAT

Hochschule RheinMain

Definition (3-CNF-SAT)

Das 3-CNF-SAT-Problem in der gegeben/gefragt-Notation:

gegeben: Eine aussagenlogische Formel F in konjunktiver Normalform, sodass jede Klausel höchstens 3 Literale enthält.

gefragt: Ist F erfüllbar? Genauer: Gibt es eine erfüllende Belegung der Variablen mit den Wahrheitswerten 0 und 1, sodass F den Wert 1 erhält?

$\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT

Hochschule RheinMain

Satz

3-CNF-SAT ist $\mathcal{NP}$-vollständig.

Beweis, Teil 1: 3-CNF-SAT ist in $\mathcal{NP}$

- Rate nicht-deterministisch eine Belegung der Variablen
- Prüfe in jedem nicht-deterministischen Zweig deterministisch, ob die Belegung die 3-CNF wahr macht. Akzeptiere in diesem Fall, sonst verwerfe.
  Dies geht in Polynomialzeit in der Größe der 3-CNF.
- Daher kann 3-CNF-SAT auf einer NTM in Polynomialzeit entschieden werden.

$\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT (2)

Hochschule RheinMain

Beweis, Teil 2: 3-CNF-SAT ist $\mathcal{NP}$-schwer

- Wir zeigen SAT $\leq_p$ 3-CNF-SAT.
- Gesucht: Polynomiell berechenbare, totale Funktion f, sodass F erfüllbar g.d.w. 3-CNF $f(F)$ erfüllbar.
- f muss **die Erfüllbarkeit erhalten**, aber nicht die **Äquivalenz**
- Verfahren, um F in erfüllbarkeitsäquivalente 3-CNF $f(F)$ umzuformen, sodass $f(F)$ **polynomielle Größe** in $|F|$ hat:

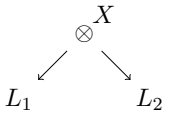
Tseitin-Transformation

$\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT (3)

Hochschule RheinMain

Tseitin-Transformation:

- Schiebe alle **Negationen nach innen** vor die Literale, dabei werden die Regeln $\neg\neg F \to F$, $\neg(F \wedge G) \to \neg F \vee \neg G$, $\neg(F \vee G) \to \neg F \wedge \neg G$, $\neg(F \iff G) \to (\neg F) \iff G$ und $\neg(F \implies G) \to F \wedge \neg G$ angewendet.
- Syntaxbaum der Formel (Blätter = Literale):
  Für jeden Nichtblatt-Knoten: neue aussagenlogische Variable
- pro Gabelung: erzeuge



$(X \iff (L_1 \otimes L_2))$

wobei $\otimes \in \{ \iff, \wedge, \vee, \implies \}$ und L_1, L_2 entweder die neue Variable oder Literal am Blatt.

- Konjugiere Formeln zu F' und erzeuge $W \wedge F'$, mit W Variable für die Wurzel.

$\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT (3)

Hochschule RheinMain

Tseitin-Transformation (Fortsetzung)

- Insgesamt: $W \wedge \bigwedge_{i,j,k}(X_i \iff (X_j \otimes_i X_k))$, wobei X_r Literale sind.
- Berechne für jede Subformel $(X_i \iff (X_j \otimes_i X_k))$ die CNF mit dem üblichen Algorithmus.
- Lösche doppelte Vorkommen von Literalen.
- Ergibt 3-CNF, da pro Klausel nur 3 verschiedene Variablen vorkommen können.

$\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT (4)

Hochschule RheinMain

Komplexität:

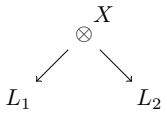
- Größe der kleinen CNFs ist konstant.
- Jede Klausel hat nur 3 Literale: Mehr Variablen gibt es in den kleinen Formeln nicht, doppelte Literale löschen
- Größe der 3-CNF ist polynomiell in der ursprünglichen Formel
- Berechnung in Polynomialzeit geht daher!

$\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT (5)

Hochschule RheinMain

F erfüllbar g.d.w. $f(F)$ erfüllbar:

- „$\Rightarrow$“: Sei I Belegung mit $I(F) = 1$. Sei I' Belegung mit
  - $I'(X) = I(X)$ für alle Variablen, die in F vorkommen
  - $I'(W) = 1$ für die Variable W an der Wurzel
  - $I'(X) = I'(L_1 \otimes L_2)$ für

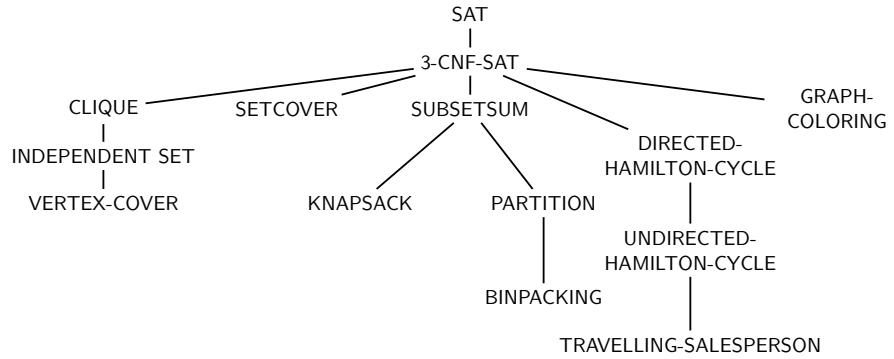


  Dann gilt: $I'(f(F)) = 1$
- „$\Leftarrow$“: Sei J Belegung von $f(F)$ mit $J(f(F)) = 1$
  Dann: $I = J_{|Var(F)}$ macht F wahr.

Damit: SAT $\leq_p$ 3-CNF-SAT.
Da SAT $\mathcal{NP}$-schwer, ist somit auch 3-CNF-SAT $\mathcal{NP}$-schwer.

Übersicht

Hochschule RheinMain



Wir lassen die meisten $\mathcal{NP}$-Vollständigkeitsbeweise weg, und lernen nur die Probleme kennen.

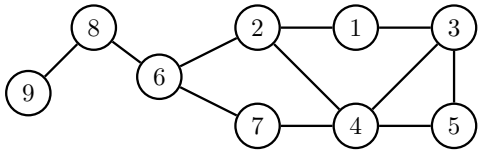
Wiederholung: Graphen

Hochschule RheinMain

Ein ungerichteter Graph $G = (V, E)$ besteht aus

- einer endlichen Menge V von Knoten (vertices)
- einer endlichen Menge E von Kanten (edges) wobei Kanten aus zwei Knoten bestehen, und für alle $\{u, v\} \in E$ gilt $u \neq v$

Z.B.: $G = (V, E)$ mit $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$E = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}, \{4, 5\}, \{2, 6\}, \{4, 7\}, \{6, 7\}, \{6, 8\}, \{8, 9\}\}$



Beachte: Wir verbieten

- Schlingen $\{u, u\} \in E$,
- Mehrfachkanten (= mehrere Kanten zwischen u und v)
- Hypergraphen (Kanten mit nicht genau 2 Knoten)

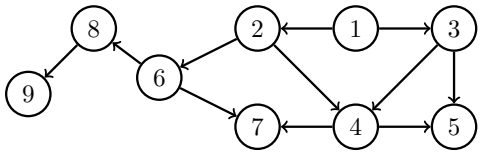
Wiederholung: Gerichtete Graphen

Hochschule RheinMain

Bei gerichteten Graphen sind die Kanten gerichtet:

$$(u, v) \in E \text{ statt } \{u, v\} \in E.$$

Daher sind Kanten (u, v) und (v, u) verschieden.
Z.B.: $G = (V, E)$ mit $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$E = \{(1, 2), (1, 3), (2, 4), (3, 4), (3, 5), (4, 5), (2, 6), (4, 7), (6, 7), (6, 8), (8, 9)\}$



Hochschule RheinMain

WEITERE NP-VOLLSTÄNDIGE PROBLEME

Infos zur Klausur

Hochschule RheinMain

Klausur

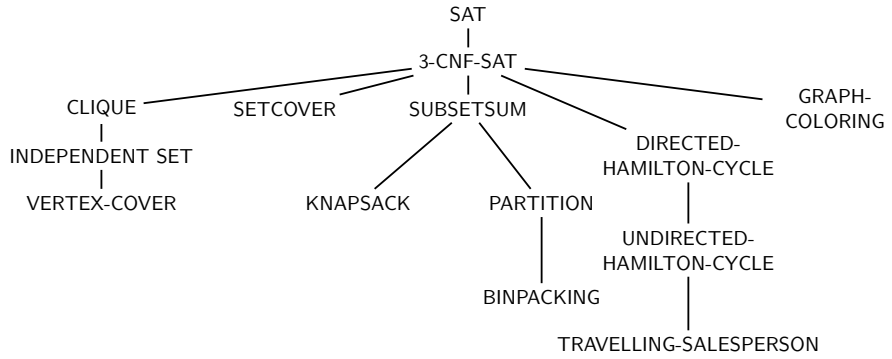
- 90 Minuten
- 12.08. 9-11 Uhr
- Keine Hilfsmittel erlaubt, auch kein Spickzettel o.ä.

Fragestunde

- 06.08. um 14 Uhr, **online**
- https://zapp.mi.hs-rm.de/meet/david.sabel
- Es gibt dazu aber auch noch eine Email ...

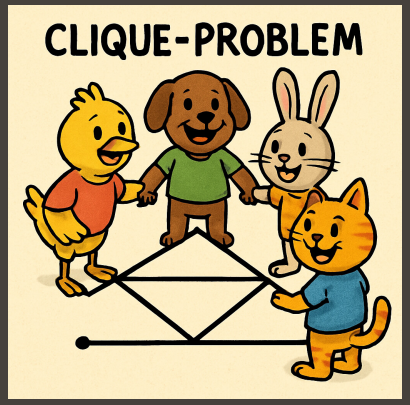
Überblick

Hochschule RheinMain



CLIQUE

Hochschule RheinMain



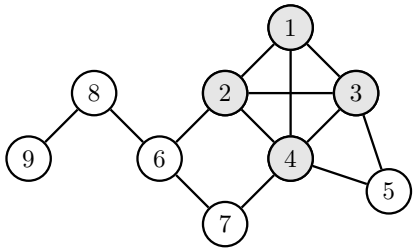
Cliquen in Graphen

Hochschule RheinMain

Definition

Für einen ungerichteten Graph $G = (V, E)$ ist eine **Clique der Größe k** eine Menge $V' \subseteq V$, sodass $|V'| = k$ und für alle $u, v \in V'$ mit $u \neq v$ gilt: $\{u, v\} \in E$.

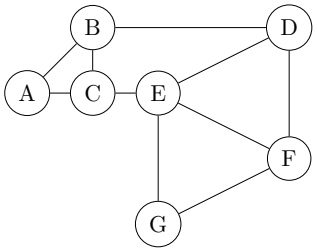
Beispiel:



Clique der Größe 4

Quiz 3

Hochschule RheinMain



Welches ist die größte Zahl k, so dass gilt: Der Graph hat eine k-Clique?

arsnova.hs-rm.de
6750 1376

CLIQUE-Problem

Hochschule RheinMain

Definition (CLIQUE-Problem)

Das CLIQUE-Problem in der gegeben/gefragt-Notation:

gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_0$.

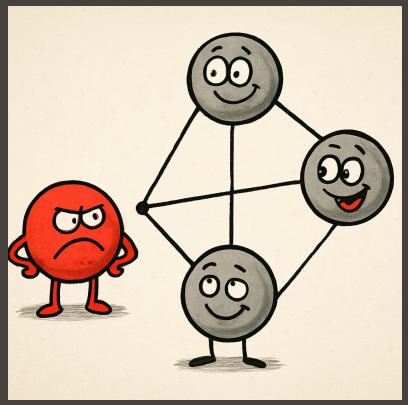
gefragt: Besitzt G eine Clique der Größe mindestens k?

Satz

CLIQUE ist $\mathcal{NP}$-vollständig.

INDEPENDENT SET

Hochschule RheinMain



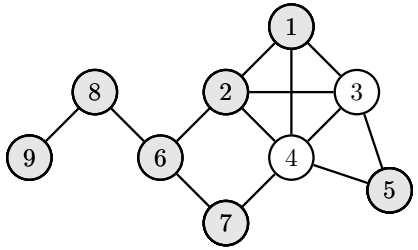
Unabhängige Knotenmenge

Hochschule RheinMain

Definition

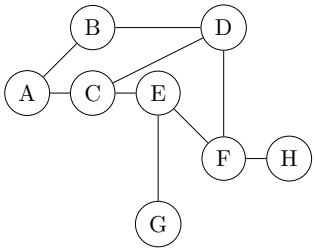
Für einen Graphen $G = (V, E)$ ist $V' \subseteq V$ eine unabhängige Knotenmenge (independent set), wenn keine zwei Knoten aus V' über eine Kante verbunden sind, d.h. $u, v \in V' \implies \{u, v\} \notin E$.

Beispiel:



Quiz 4

Hochschule RheinMain



Welches ist die größte Zahl k, so dass gilt: Der Graph hat eine unabhängige Knotenmenge der Mächtigkeit k?

arsnova.hs-rm.de
6750 1376

Komplementgraph

Hochschule RheinMain

Für Graph $G = (V, E)$ ist der Komplementgraph zu G der Graph

$$\overline{G} = (V, \overline{E}) \text{ mit } \overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$$

Lemma

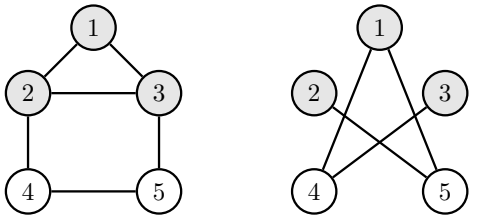
Für jeden ungerichteten Graph G gilt: G hat eine unabhängige Knotenmenge der Größe k genau dann, wenn $\overline{G}$ eine Clique der Größe k hat.

Beweis:

V' ist unabhängige Knotenmenge der Größe k
g.d.w. $V' \subseteq V$ mit $u, v \in V' \implies \{u, v\} \notin E$ und $|V'| = k$
g.d.w. $V' \subseteq V$ mit $u, v \in V' \implies \{u, v\} \in \overline{E}$ und $|V'| = k$
g.d.w. V' ist eine Clique der Größe k in $\overline{G}$

Beispiel

Hochschule RheinMain

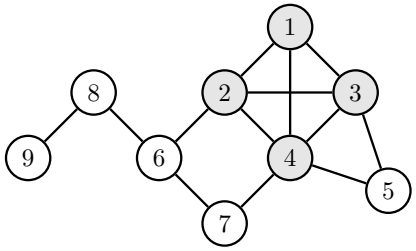


Graph

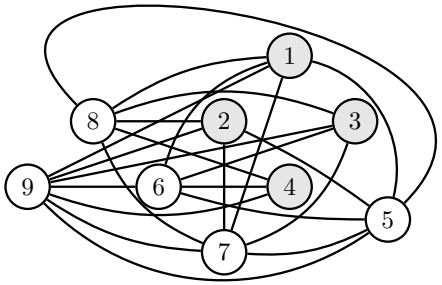
Komplementgraph

Beispiel (2)

Hochschule RheinMain



Graph



Komplementgraph

INDEPENDENT-SET Problem

Hochschule RheinMain

Definition (INDEPENDENT-SET-Problem)

Das INDEPENDENT-SET-Problem in der gegeben/gefragt-Notation:

gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_0$.

gefragt: Besitzt G eine unabhängige Knotenmenge der Größe mindestens k?

Satz

INDEPENDENT-SET ist $\mathcal{NP}$-vollständig.

Überdeckende Knotenmenge

Hochschule RheinMain

Definition

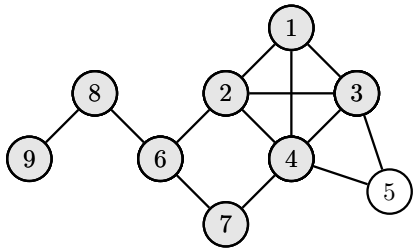
Für einen Graph $G = (V, E)$ ist $V' \subseteq V$ eine überdeckende Knotenmenge (vertex cover), wenn jede Kante aus E mindestens 1 Knoten in V' hat, d.h. für alle Knoten $u, v \in V : \{u, v\} \in E \implies u \in V' \vee v \in V'$.

Beachte:

- V ist immer eine überdeckende Knotenmenge
- Man möchte ein möglichst kleine Menge V' finden.

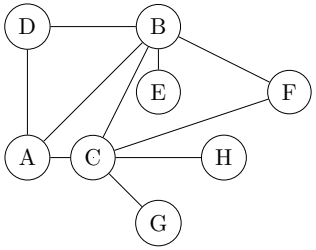
Beispiel

Hochschule RheinMain



Quiz 5

Hochschule RheinMain



Welches ist die kleinste Zahl k, so dass gilt: Der Graph hat eine überdeckende Knotenmenge der Mächtigkeit k?

arsnova.hs-rm.de
6750 1376

Überdeckende Knotenmengen vs. unabhängige Knotenmengen

Hochschule RheinMain

Lemma

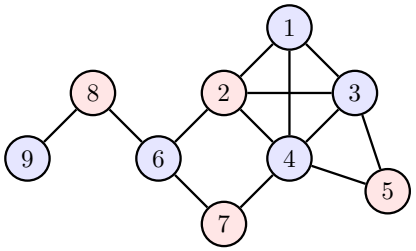
$G = (V, E)$ hat eine unabhängige Knotenmenge der Größe k
g.d.w.
G hat eine überdeckende Knotenmenge der Größe $|V| - k$.

Beweis:

	$V' \subseteq V$ ist unabhängige Knotenmenge
g.d.w.	$u, v \in V' \implies \{u, v\} \notin E$
g.d.w.	$\{u, v\} \in E \implies u \notin V' \vee v \notin V'$
g.d.w.	$\{u, v\} \in E \implies (u \in V \setminus V') \vee (v \in V \setminus V')$
g.d.w.	$V \setminus V'$ ist überdeckende Knotenmenge

Beispiel

Hochschule RheinMain

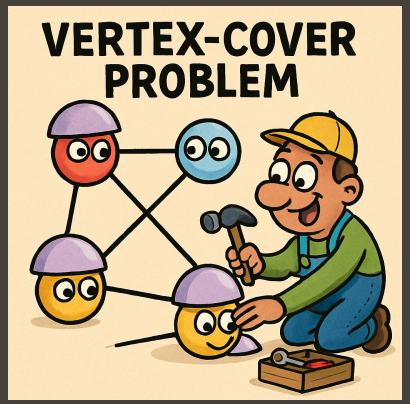


unabhängige Knotenmenge

überdeckende Knotenmenge

VERTEX COVER

Hochschule RheinMain



Das VERTEX-COVER-Problem

Hochschule RheinMain

Definition (VERTEX-COVER-Problem)

Das VERTEX-COVER-Problem in der gegeben/gefragt-Notation:

gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_0$.

gefragt: Besitzt G eine überdeckende Knotenmenge der Größe höchstens k?

Satz

VERTEX-COVER- ist $\mathcal{NP}$-vollständig.

SET COVER

Hochschule RheinMain

Das SETCOVER-Problem

Hochschule RheinMain

Definition (SETCOVER-Problem)

Das SETCOVER-Problem in der gegeben/gefragt-Notation:

gegeben: Mengen $T_1, \ldots, T_k$ mit $T_1, \ldots, T_k \subseteq M$, wobei M eine endliche Grundmenge ist und eine Zahl $n \leq k$.

gefragt: Gibt es eine Auswahl von n Mengen $T_{i_1}, \ldots, T_{i_n}$ $(i_j \in \{1, \ldots, k\})$ mit $T_{i_1} \cup \ldots \cup T_{i_n} = M$?

Beispiel:

$T_1 = \{1, 2, 3, 5\}, T_2 = \{1, 2\}, T_3 = \{3, 4\}, T_4 = \{3\}$ mit $M = \{1, 2, 3, 4, 5\}$ und $n = 2$

Lösung: T_1, T_3, da $T_1 \cup T_3 = M$.

Satz

SETCOVER ist $\mathcal{NP}$-vollständig.

Quiz 6

Hochschule RheinMain

Ein Lehrer möchte sicherstellen, dass alle Themen abgedeckt sind, indem er eine minimale Anzahl von Unterrichtseinheiten auswählt.
Unterrichtseinheiten und Themen:

Einheit 1: A,C Einheit 2: A,B
Einheit 3: B,C Einheit 4: B,D

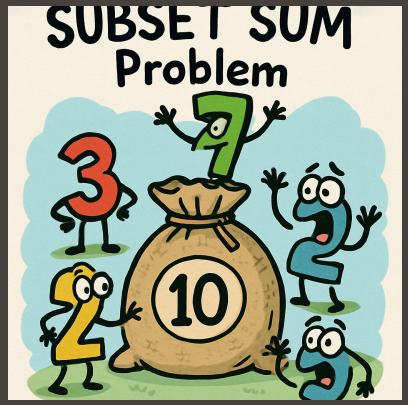
Welche Unterrichtseinheiten sollten ausgewählt werden, um alle Themen A,B,C,D abzudecken?

1. 1,2
2. 1,2,3
3. 1,2,4
4. 1,4

arsnova.hs-rm.de
6750 1376

SUBSETSUM

Hochschule RheinMain



Das SUBSETSUM-Problem

Hochschule RheinMain

Definition (SUBSETSUM-Problem)

Das SUBSETSUM-Problem in der gegeben/gefragt-Notation:

gegeben: Natürliche Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$ und $s \in \mathbb{N}_0$
gefragt: Gibt es eine Teilmenge $I \subseteq \{1, \ldots, k\}$, sodass $\sum_{i \in I} a_i = s$?

Beispiel:$a_1, \ldots, a_6 = 1, 21, 5, 16, 12, 19$ und $s = 49$

Lösung: $2, 4, 5$, da $21 + 16 + 12 = 49$

Satz

Das SUBSETSUM-Problem ist $\mathcal{NP}$-vollständig.

Quiz 7

Hochschule RheinMain

Sie haben ein Budget von 1000 Euro für die Dekoration einer Veranstaltung. Sie haben verschiedene Dekorationsartikel zur Auswahl, die unterschiedliche Kosten haben:
Dekorationsartikel:
Artikel A: 200 Euro Artikel D: 400 Euro
Artikel B: 300 Euro Artikel E: 250 Euro
Artikel C: 150 Euro

Gibt es eine Kombination von Dekorationsartikeln, die genau Ihr Budget von 1000 Euro erreicht?

- Ja, A,C,D
- Ja B,C,E
- Ja, A,C,D,E
- Nein

arsnova.hs-rm.de
6750 1376

KNAPSACK

Hochschule RheinMain

Das Rucksack-Problem

Hochschule RheinMain

Definition (KNAPSACK-Problem)

Das KNAPSACK-Problem in der gegeben/gefragt-Notation:

gegeben: k Gegenstände mit Gewichten $w_1, \ldots, w_k \in \mathbb{N}_0$ und Nutzenwerten $n_1, \ldots, n_k \in \mathbb{N}_0$,
sowie zwei Zahlen $s_w, s_n \in \mathbb{N}_0$.

gefragt: Gibt es Teilmenge $I \subseteq \{1, \ldots, k\}$, sodass $\sum_{i \in I} w_i \leq s_w$ und $\sum_{i \in I} n_i \geq s_n$?

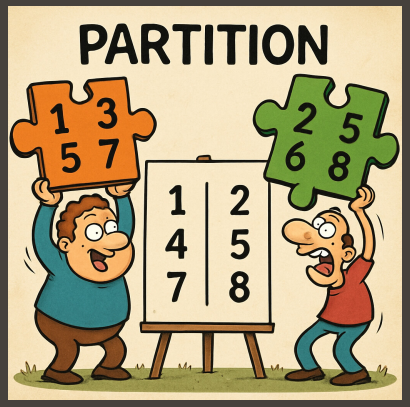
Beachte für $w_i = n_i$ und $s_n = s_w$ ergibt sich genau das SUBSETSUM-Problem!

Satz

KNAPSACK ist $\mathcal{NP}$-vollständig.

PARTITION

Hochschule RheinMain



Das PARTITION-Problem

Hochschule RheinMain

Definition (PARTITION-Problem)

Das PARTITION-Problem in der gegeben/gefragt-Notation:

gegeben: Natürliche Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$

gefragt: Gibt es eine Teilmenge $I \subseteq \{1, \ldots, k\}$, sodass $\sum_{i \in I} a_i = \sum_{i \in \{1,\ldots,k\} \setminus I} a_i$?

Satz

PARTITION ist $\mathcal{NP}$-vollständig.

BINPACKING

Hochschule RheinMain

Das BINPACKING-Problem

Hochschule RheinMain

Definition (BINPACKING-Problem)

Das BINPACKING-Problem in der gegeben/gefragt-Notation:

gegeben: Natürliche Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$, die Behältergröße $b \in \mathbb{N}_0$ und die Anzahl der Behälter m

gefragt: Kann man alle gegeben Zahlen so auf die Behälter aufteilen, sodass keiner der Behälter überläuft?
Formal: Gibt es eine totale Funktion $assign : \{1, \ldots, k\} \to \{1, \ldots, m\}$, sodass für alle $j \in \{1, \ldots, m\}$ gilt: $\sum_{assign(i)=j} a_i \leq b$?

Satz

BINPACKING ist $\mathcal{NP}$-vollständig.

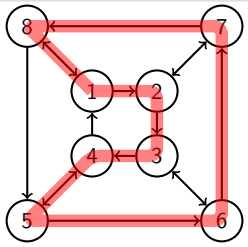
Hamiltonische Kreise

Hochschule RheinMain

Definition

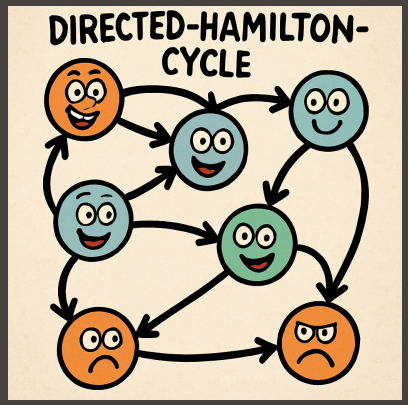
In einem gerichteten Graphen ist ein Hamilton-Kreis, ein Kreis, der genau alle Knoten einmal besucht.
Formal: Für $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ ist ein Hamilton-Kreis eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ und $(v_{\pi(n)}, v_{\pi(1)}) \in E$.

Beispiel:



DIRECTED HAMILTON CYCLE

Hochschule RheinMain



Das DIRECTED-HAMILTON-CYCLE-Problem

Hochschule RheinMain

Definition (DIRECTED-HAMILTON-CYCLE-Problem)

Das DIRECTED-HAMILTON-CYCLE-Problem in der gegeben/gefragt-Notation:

gegeben: Ein gerichteter Graph $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$

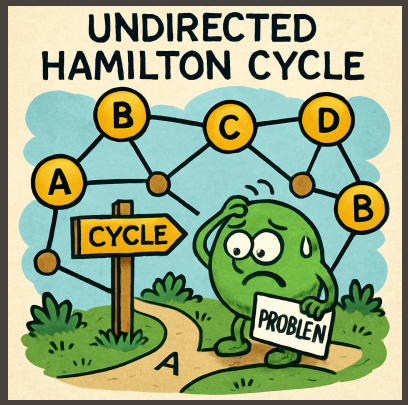
gefragt: Gibt es einen Hamilton-Kreis in G?

Satz

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-vollständig.

UNDIRECTED HAMILTON CYCLE

Hochschule RheinMain



Das UNDIRECTED-HAMILTON-CYCLE-Problem

Hochschule RheinMain

Hamilton-Kreis im ungerichteten Graph

Sei $G = (V, E)$ ein ungerichteter Graph. Ein Hamilton-Kreis ist eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $\{v_{\pi(i)}, v_{\pi(i+1)}\} \in E$ und $\{v_{\pi(n)}, v_{\pi(1)}\} \in E$?

Definition (UNDIRECTED-HAMILTON-CYCLE-Problem)

Das UNDIRECTED-HAMILTON-CYCLE-Problem in der gegeben/gefragt-Notation:

gegeben: Ein ungerichteter Graph $G = (V, E)$.

gefragt: Gibt es einen Hamilton-Kreis in G?

Satz

UNDIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-vollständig.

TRAVELLING SALES PERSON

Hochschule RheinMain

Das TRAVELLING-SALESPERSON-Problem

Hochschule RheinMain

Definition (TRAVELLING-SALESPERSON-Problem)

Das TRAVELLING-SALESPERSON-Problem in der gegeben/gefragt-Notation:

gegeben: Ein Menge von Knoten (Städten) $V = \{v_1, \ldots, v_n\}$, eine $(n \times n)$-Matrix $(M_{i,j})$ mit Entfernungen $M_{i,j} \in \mathbb{N}_0$ zwischen den Städten v_i und v_j, sowie eine Zahl $k \in \mathbb{N}_0$.

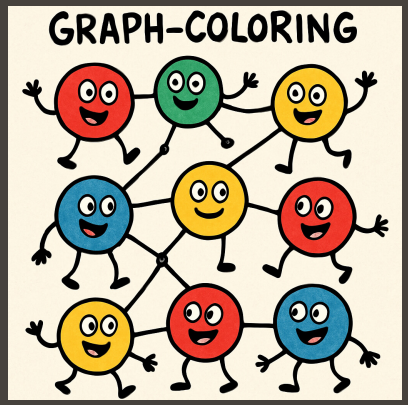
gefragt: Gibt es eine Rundreise, die alle Städte besucht, der Startort gleich dem Zielort ist, und nicht länger als k ist.Formal: Gibt es eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $(\sum_{i=1}^{n-1} M_{\pi(i),\pi(i+1)}) + M_{\pi(n)+\pi(1)} \leq k$.

Satz

Das TRAVELLING-SALESPERSON-Problem ist $\mathcal{NP}$-vollständig.

GRAPH COLORING

Hochschule RheinMain



Das GRAPH-COLORING-Problem

Hochschule RheinMain

Definition (GRAPH-COLORING-Problem)

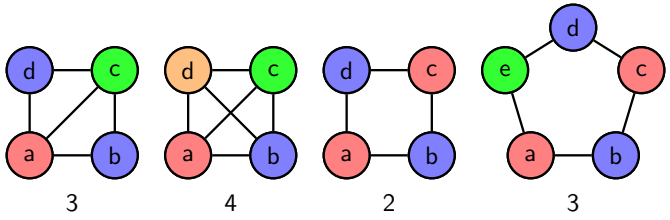
Das GRAPH-COLORING-Problem in der gegeben/gefragt-Notation:

gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_0$.

gefragt: Gibt es Färbung der Knoten in V mit $\leq k$ Farben, sodass keine zwei benachbarten Knoten in G, die gleiche Farbe erhalten.

Beispiel

Hochschule RheinMain



Wie viele Farben sind jeweils minimal notwendig?

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING

Hochschule RheinMain

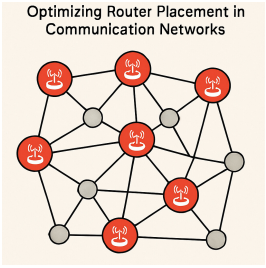
Satz

GRAPH-COLORING ist $\mathcal{NP}$-vollständig.

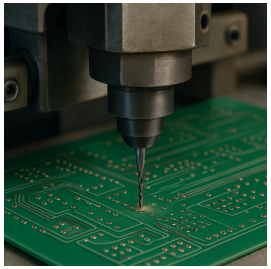
Treten $\mathcal{NP}$-vollständige Probleme in der Praxis auf?

Hochschule RheinMain

Ja!



Router-Platzieren:
minimale Anzahl an Knoten für die Router finden
→ VERTEX-COVER



Löcherbohren in Leiterplatten:
Finde kürzeste „Rundreise“ für den Bohrer
→ TRAVELLING-SALESPERSON

Playlist erstellen:
X Minuten mit Songs füllen.
→ SUBSETSUM

Wie geht man in der Praxis damit um?

Hochschule RheinMain

- Bewusst machen, dass es **schwierig** wird.
- Aber: Aufgeben ist keine Wahl!
- Verschiedene Vorgehensweisen, je nach Ziel und Gegebenheiten:
  - Exponentielle Laufzeit hinnehmen
    (funktioniert nur bei sehr kleinen Instanzen)
  - Exponentielle Laufzeit angreifen
  - Auf optimale Ergebnisse verzichten

Exponentielle Laufzeit hinnehmen

Hochschule RheinMain

Brute-Force-Methode:

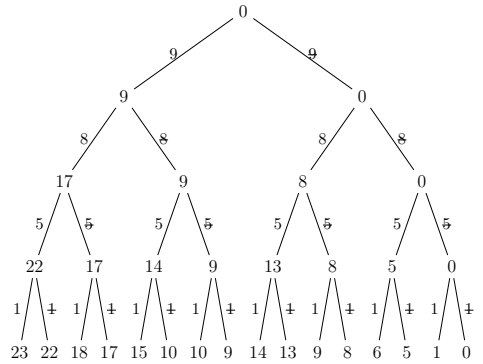
Alle Möglichkeiten durchprobieren
Z.B. mit Entscheidungsbaum und Tiefensuche

Evtl. sinnvoll, wenn man

- **sehr kleine** Instanzen hat, oder
- kleine Instanzen und Zeit hat

Beispiele:

- SUBSETSUM mit wenigen Elementen
- Plan, der nur sehr selten berechnet wird



Beispiel (SUBSETSUM): $\{9, 8, 5, 1\}$ und 13

Exponentielle Laufzeit angreifen: Verbessern des Brute-Force-Ansatzes

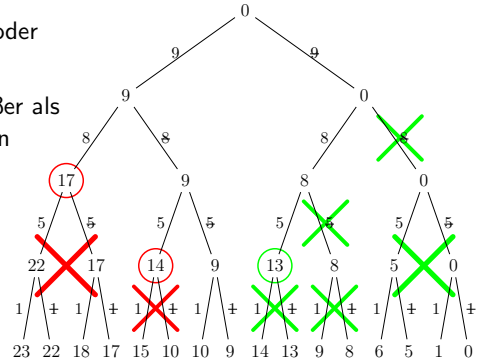
Hochschule RheinMain

Suchraum verkleinern, z.B. durch:

- **Vorverarbeitung**: Eingabe vereinfachen oder Teillösungen vorberechnen.

  Z.B. SUBSETSUM: Zahlen in S, die größer als Zielsumme z sind, können entfernt werden

- **Abschneiden des Entscheidungsbaums**
  Ziel: Basis verkleinern ($2^n \to 1, \ldots^n$)
  Beachte (bei 1 Operation $= 10^{-9}$ s) gilt:
  - $2^{42} \approx 1,2$ h
  - $1,5^{72} \approx 1,3$ h
  - $1,2^{160} \approx 1,3$ h
  - $1,1^{305} \approx 1,2$ h



Entscheidungsbaum für SUBSETSUM mit $\{9, 8, 5, 1\}$ und 13

Exponentielle Laufzeit angreifen: Weitere Möglichkeiten

Hochschule RheinMain

- Kodieren des Problems in **bekannte Probleme** (LP,SAT,SMT).
  Anschließend einen in der Praxis **guten Solver** verwenden.
- **Polynomielle Verfahren**, wenn **Parameter fest** sind

  Beispiel: Summe z im SUBSETSUM-Problem ist nicht zu groß.

  Algorithmus mit Zeitkomplexität $O(n \cdot z)$ verwendet dynamisches Programmieren.

  Idee: Für $S = \{x_1, \ldots, x_n\}$ und z fülle für $i = 1, \ldots, n$ und $k = 0, \ldots, z$:

$$t[i][k] = \begin{cases} 1, & \text{wenn } \{x_1, \ldots, x_i\} \text{ hat Teilmenge mit Summe } k \\ 0, & \text{sonst} \end{cases}$$

  Danach prüfe $t[n][z]$.

Auf optimale Ergebnisse verzichten

Hochschule RheinMain

- Oft muss das Ergebnis nur **gut**, aber **nicht optimal** sein
- **Näherungslösungen** statt genauer Lösung berechnen
- Einige Methoden:
  - Lokale Suche
  - Randomisierte Suche
  - Heuristische Suche
  - Evolutionäre Algorithmen
  - Approximationsalgorithmen
- Kombination der Methoden möglich

Zusammenfassung

Hochschule RheinMain

Inhaltsübersicht

Hochschule RheinMain

Teil I: Formale Sprachen und Automatentheorie

- Chomsky-Grammatiken und die Chomsky-Hierarchie
- Reguläre Sprachen: DFAs, NFAs, reguläre Ausdrücke, Äquivalenz der Formalismen, Pumping-Lemma, Minimierung von DFAs, Abschlusseigenschaften, Entscheidbarkeitsresultate
- Kontextfreie Sprachen: Normalformen, Pumping-Lemma, CYK-Algorithmus, Kellerautomaten (PDA und DPDA), Abschlusseigenschaften und Entscheidbarkeitsresultate
- Kontextsensitive und Typ 0-Sprachen: Turingmaschinen (DTM und NTM), LBAs, Abschlusseigenschaften

Inhaltsübersicht

Hochschule RheinMain

Teil II: Berechenbarkeitstheorie

- Berechenbarkeit
- Turingmaschinen und Turingberechenbarkeit
- Unentscheidbarkeit: Halteproblem
- Reduktionen, PCP

Teil III: Komplexitätstheorie

- $\mathcal{P}$ und $\mathcal{NP}$
- NP-Vollständigkeit
- Polynomialzeitreduktionen
- Satz von Cook
- NP-vollständige Probleme

Klassiker: „Rechenaufgaben“

Hochschule RheinMain

- Automat angeben
- Transformation NFA in DFA mit Potenzmengenkonstruktion
- Minimierung von DFAs
- CYK-Algorithmus ausführen

Klassiker: „Beweisaufgaben“

Hochschule RheinMain

- Nichtregularität einer Sprache zeigen mit Pumping-Lemma
- Nicht-Kontextfreiheit einer Sprache zeigen mit Pumping-Lemma für CFLs
- Unentscheidbarkeit zeigen mit Reduktion
- ~~NP-Vollständigkeit zeigen, u.a. mit Polynomialzeitreduktionen~~

Weitere typische Aufgaben

Hochschule RheinMain

- Sprachen angeben in einem der vielen Formalismen: DFA, NFA, regulärer Ausdruck, reguläre Grammatik Kontextfreie Grammatik, Kellerautomat, deterministischer Kellerautomat Turingmaschine angeben
- Formalismen ineinander überführen, z.B. regulärer Ausdruck in DFA usw.
- Sprachen „typisieren“ in der Chomsky-Hierarchie