

- Wir haben Turingmaschinen eingeführt.
- Bis auf einen polynomiellen Anstieg der Rechenzeit haben Turingmaschinen die Rechenkraft von parallelen Supercomputern!

Statt Turingmaschinen anzugeben, genügt die Angabe eines C++ Programms oder die Angabe eines Pseudocodes.

- Nichtdeterministische Turingmaschinen wählen einen aus möglicherweise vielen Befehlen aus.

Das Konzept „Nichtdeterminismus“ formalisiert die Fähigkeit „zu raten“.

Die Klasse \mathcal{P}

Die Klasse \mathcal{P} besteht aus allen Sprachen L für die es eine Turingmaschine M mit $L = L(M)$ und

$$\text{zeit}_M(n) = O((n+1)^k) \text{ für eine Konstante } k$$

gibt. Wir sagen, dass L **effizient berechenbar** ist, wenn $L \in \mathcal{P}$ gilt.

Die **Sprachenversionen** der folgenden Probleme liegen in \mathcal{P} .

- ▶ Sortieren.
- ▶ Graphprobleme wie:
 - ★ Das Zusammenhangsproblem,
 - ★ kürzeste Wege Probleme,
 - ★ die Berechnung minimaler Spannbäume.
- ▶ die Berechnung von Huffman Codes (durch Greedy Algorithmen).
- ▶ die Berechnung
 - ★ des gewichteten Intervall Scheduling
 - ★ und des paarweise Alignments(durch die dynamische Programmierung).

Die Definition von \mathcal{NP}

Eine Sprache L gehört genau dann zu \mathcal{NP} , wenn es eine nichtdeterministische Turingmaschine M mit $L = L(M)$ und $\text{Zeit}_M(n) = O((n+1)^k)$ für eine Konstante k gibt.

- Auch nichtdeterministische Supercomputer können durch nichtdeterministische Nähmaschinen mit nur polynomieller Verzögerung simuliert werden.
- **Um nachzuweisen, dass eine Sprache zu \mathcal{NP} gehört, genügt zum Beispiel die Angabe eines nichtdeterministischen C++ Programms, das in polynomieller Zeit läuft.**

\mathcal{NP} enthält alle Probleme mit kurzen, schnell verifizierbaren Lösungen: Übersetze „kurz und schnell“ mit „polynomiell“.

Welche Sprachen liegen in \mathcal{NP} ?

- Das Erfüllbarkeitsproblem **KNF-SAT**: Kann eine aussagenlogische Formel in konjunktiver Normalform erfüllt werden?
 - ▶ **3-SAT** (wie KNF-SAT mit höchstens drei Literalen pro Klausel).
 - ▶ Sind zwei **Schaltungen nicht äquivalent**?
- **Clique**: Besitzt ein Graph eine Menge von k Knoten, die paarweise miteinander verbunden sind?
 - ▶ **Independent Set**, ein eineiiger Zwilling von Clique: Diesmal dürfen die Knoten nicht miteinander verbunden sein.
 - ▶ **Vertex Cover**, ein zweieiiger Zwilling von Independent Set: Besitzt jede Kante mindestens einen Endpunkt in einer Menge von k Knoten?
- Wegeprobleme wie **TSP**, **längste Wege** und **Hamiltonsche Kreise**.
- **Set Cover**: Kann ein Universum von k Mengen überdeckt werden?

Welche Sprachen liegen wahrscheinlich nicht in \mathcal{NP} ?

- Sind zwei Schaltungen äquivalent?
 - ▶ Raten „bringt nichts“, wir müssen Äquivalenz auf allen Eingaben verifizieren.
 - ▶ Das Problem gehört zu $\text{co-}\mathcal{NP} = \{\bar{L} \mid L \in \mathcal{NP}\}$.
- Besitzt der ziehende Spieler in einem „interessanten“ Zweipersonenspiel einen Gewinnzug?
 - ▶ Wir können zwar den Gewinnzug raten,
 - ▶ aber wie sollen wir effizient verifizieren, dass jede gegnerische Strategie verliert?

Die polynomielle Reduktion

- Wir sagen, dass L_1 auf L_2 **polynomiell reduzierbar** ist und schreiben

$$L_1 \leq_p L_2,$$

wenn es eine **effiziente** Transformation M gibt, so dass für **alle** Eingaben w

$$w \in L_1 \Leftrightarrow M(w) \in L_2.$$

- Die wesentliche Eigenschaft der polynomiellen Reduktion:

Wenn $K \leq_p L$ und wenn $L \in \mathcal{P}$, dann auch $K \in \mathcal{P}$.

- ▶ Wenn L einen effizienten Algorithmus besitzt, dann auch K .
- ▶ Warum?

(a) Eine Sprache L ist genau dann \mathcal{NP} -hart, wenn

$$K \leq_p L \text{ für alle Sprachen } K \in \mathcal{NP}.$$

(b) L ist genau dann \mathcal{NP} -vollständig, wenn $L \in \mathcal{NP}$ und L eine \mathcal{NP} -harte Sprache ist.

\mathcal{NP} -vollständige Sprachen sind schwer

Die Sprache L sei \mathcal{NP} -vollständig. Wenn $\mathcal{P} \neq \mathcal{NP}$, dann ist $L \notin \mathcal{P}$.
Warum?

Und die Konsequenzen:

Wahrscheinlich wird kein \mathcal{NP} -vollständiges Problem effiziente Algorithmen besitzen,
denn deterministische Programme können nicht effizient raten.

Wie zeigt man, dass eine Sprache \mathcal{NP} -vollständig ist?

- Wenn L_1 \mathcal{NP} -hart ist
- und wenn $L_1 \leq_p L_2$,
- dann ist L_2 eine \mathcal{NP} -harte Sprache.

Wenn zusätzlich $L_2 \in \mathcal{NP}$ gilt, dann ist L_2 \mathcal{NP} -vollständig.

- Warum ist die Aussage richtig?
 - ▶ Zum Beweis benötigt man die folgende Aussage: Wenn $L_0 \leq_p L_1$ und $L_1 \leq_p L_2$, dann ist $L_0 \leq_p L_2$.
 - ▶ Und warum ist diese Aussage richtig?

Der Satz von Cook: KNF-SAT ist \mathcal{NP} -vollständig.

- $\text{KNF-SAT} \leq_p \text{3-SAT}$.
- $\text{3-SAT} \leq_p \text{Clique}$.
- $\text{Clique} \leq_p \text{Independent Set}$.
- $\text{Independent Set} \leq_p \text{Vertex Cover}$.
- $\text{Vertex Cover} \leq_p \text{Set Cover}$.

KNF-SAT, 3-SAT, Clique, Independent Set, Vertex Cover und Set Cover sind \mathcal{NP} -vollständig.

- $K \leq_p L$ sei zu zeigen:
 - ▶ In der Konstruktion der Transformation M für Eingabe w wird angenommen, dass bekannt ist, ob $w \in L$ oder $w \notin L$.
 - ▶ Achtung: Die Transformation M muss **effizient** sein!
- Die Transformation M soll die Reduktion $K \leq_p L$ nachweisen, aber es wird nur $w \in K \Rightarrow M(w) \in L$ gezeigt.
 - ▶ Eine solche Aussage ist trivial und gilt z.B. für die Sprache $L = \Sigma^*$.
 - ▶ $L = \Sigma^*$ ist eine simple Sprache, die zur Klasse \mathcal{P} gehört.

Die polynomielle Reduktion: Was ist zu tun?

Die Reduktion $K \leq_p L$ sei zu zeigen.

- Konstruiere die Transformation M , aber wie?
 - ▶ Wieso kann L über K reden?
 - ★ $3\text{-SAT} \leq_p \text{Clique}$: Konstruiere G , so dass sich Cliques bestimmter Größe und erfüllende Belegungen entsprechen.
 - ★ $\text{Clique} \leq_p \text{Independent Set}$: Eine Clique für G ist eine unabhängige Menge im Komplementgraphen.
 - ★ $\text{Independent Set} \leq_p \text{Vertex Cover}$: Das Komplement einer unabhängigen Menge ist eine Knotenüberdeckung.
 - ★ $\text{Vertex Cover} \leq_p \text{Set Cover}$:
Wieso kann Set Cover über Vertex Cover reden?
 - ▶ Achtung: M muss **effizient** sein!
- Zeige für alle Eingaben w :
 - ▶ Wenn $w \in K$, dann ist $M(w) \in L$.
 - ▶ Wenn $M(w) \in L$, dann ist $w \in K$.

Entscheide für eine Turingmaschine M und ein Wort w , ob M auf Eingabe w hält.

- GL2: Das Halteproblem ist unentscheidbar.
 - ▶ Es gibt keinen Algorithmus, der das Halteproblem löst und stets hält.
 - ▶ Insbesondere liegt das Halteproblem nicht in NP.
- Zeige: $3\text{-SAT} \leq_p$ Halteproblem.

Das Halteproblem ist ein \mathcal{NP} -hartes Problem, das nicht in \mathcal{NP} liegt.