4.1 Komplexitätsklassen

Erinnerung:

$time_M : \Sigma^* \to \mathbb{N}$: Fktion, die die Anzahl der Rechenschritte einer DTM M mit Eingabealphabet Σ angibt.

$TIME(f(n))$: Klasse aller Sprachen L, für die es eine
$f : \mathbb{N} \to \mathbb{N}$ DMTM M gibt mit $L = T(M)$ und
$time_M(x) \leq f(|x|)$ für jedes $x \in \Sigma^*$

Komplexitätsklasse P:

$$P = \bigcup_{p \text{ Polynom}} TIME(p(n)) = \bigcup_{k \in \mathbb{N}} TIME(n^k)$$

Bemerkung: (a) Nachweis, dass eine Sprache L in P liegt:
Algorithmus angeben, Laufzeit durch ein Polynom abschätzen.

(b) $O(n^k)$

(c) $time_M$ für DTM definiert.

Generelle Faktoren:

* Repräsentation des Problems
* Maß für die Problemgröße
* Kostenfunktion: abhängig vom Berechnungsmodell
* Kostenmaß: Abstraktion der Kostenfunktion
  - Uniformes Kostenmaß: fixer Kostenwert unabhängig von der Größe der beteiligten Operanden. Meist 1.
  - Logarithmisches Kostenmaß: z.B. Binärdarstellung natürlicher Zahlen. Definiere Kosten in Abhängigkeit von der Länge der Zahldarstellung.

Beispiel: Betrachte das folgende LOOP-Programm:

```
// Eingabe x_0 = n
x_2 := 2;
LOOP x_0 DO x_2 := x_2 * x_2;
// Ausgabe in x_1
x_1 := x_2
```

$x_0 = 0 \rightsquigarrow 2 = 2^{2^0}$

$x_0 = 1 \rightsquigarrow 4 \quad 2^{2^1}$

$x_0 = 2 \rightsquigarrow 16 \quad 2^{2^2}$

$x_0 = 3 \rightsquigarrow 256 \quad 2^{2^3}$

... berechnet $f(n) = 2^{2^n}$.

Unter uniformem Kostenmaß ist das Programm polynomiell. Implementiert auf einer TM sind aber 2^n viele Schritte nötig um allein die Zahl 2^{2^n} zu schreiben.

Plausibler hier: logarithmisches Kostenmaß.

Für Zuweisung $x_1 := x_2$ fallen dann Kosten $\underline{\log x_2}$ an.

Merke: Solange die durchlaufenen Zahlenwerte eine obere Schranke nicht übersteigen, sind uniformes und log. Kostenmaß äquivalent.

Analog zur Komplexitätsklasse P (für DTM definiert), können wir die Klasse jener Sprachen identifizieren, die in Polynomialzeit von nicht-deterministischen TM erkannt werden.

Definition Für eine NDTM M setze

$$ntime_M(x) := \begin{cases} \min\{\text{Länge einer akzeptierenden Rechnung von } M \text{ auf } x\} & \text{, falls } x \in T(M) \\ 0 & \text{, sonst} \end{cases}$$

Definition. Sei $f: \mathbb{N} \to \mathbb{N}$.

$$NTIME(f(n)) := \{L : \text{es gibt eine NDTM } M \text{ mit } L = T(M) \text{ und } ntime_M(x) \leq f(|x|)\}$$

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

Corollar: $P \subseteq NP$.

Bew: Jede DTM M lässt sich trivialerweise als NDTM M' darstellen.

Akzeptiert M die Eingabe x nach $time_M(x)$ Schritten, so vollzieht auch M' bei Eingabe von x eine akzeptierende Rechnung, d.h. $ntime_{M'}(x) \leq time_M(x)$. Weil M deterministisch ist, läuft auch M' deterministisch ab, d.h. es gibt genau eine akzeptierende Rechnung, d.h. $ntime_{M'}(x) = time_M(x)$. ... Daraus folgt dann $P \subseteq NP$.

Problem: $NP \subseteq P$? $P \neq NP$?

Beachte: NDTM lassen sich auf DTM simulieren.
Aber: die Simulation (so wie wir sie kennen) erfordert exponentiellen Zeitaufwand.

Bemerkung: Ähnlich aussehende Probleme können in unterschiedliche Komplexitätsklassen liegen (unter der Annahme $P \neq NP$).

* **Shortest Path Problem (SPP):**

Einfacher Pfad: kein Knoten wird zweimal besucht.

Gegeben: Graph $G = \langle V, E \rangle$, $s, t \in V$, $k \in \mathbb{N}$

Gefragt: Gibt es einen (einfachen) Pfad der Länge höchstens k von s nach t

* **Longest Path Problem (LPP)**

Gegeben: Graph $G = \langle V, E \rangle$, $s, t \in V$, $k \in \mathbb{N}$

Gefragt: Gibt es einen einfachen Pfad der Länge mindestens k von s nach t.

SPP kann in $O(n^2)$ gelöst werden, d.h.

Was ist mit LPP?

Ein nicht-det. Alg. für LPP.

Current ← s
i ← 0 // Counter für Pfadlänge
WHILE Current ≠ t DO
  markiere Current als "besucht"
  wähle nicht-determ. Nachbarknoten v von Current,
    der nicht als "besucht" markiert ist.

  Current ← v
  i ← i+1

END
IF i ≥ k THEN return "Yes"
return "No"

In diesem Algorithmus „sucht" wir eine Lösung (sofern es eine solche gibt) durch wiederholte nicht-deterministische „Aktionen".

Einfache:

(1) Rate eine Sequenz von Knoten in G

(2) Verifiziere, dass diese Sequenz eine Lösung von LPP ist.

WICHTIG: Verifikation erfolgt in polynomieller Zeit.

"NP = guess and check in polynomial time"

Formal präziser:

Definition: Sei $L \subseteq \Sigma^*$ eine Sprache und $f: \mathbb{N} \to \mathbb{N}$ eine Funktion.

Eine DTM M' mit Eingabealphabet $\Sigma' \supseteq \Sigma \cup \{\#\}$ ist ein f-Verifizierer für L, falls M' auf allen Eingaben der Form $w\#z$ mit $w \in \Sigma^*$ arbeitet und zwar so, dass gilt:

(a) $time_{M}(w\#z) \leq f(|w|)$ für alle $w\#z$ Eingaben.

Zertifikat, Zeuge, Beweis

(b) $L = \{x \in \Sigma^* : M'$ akzeptiert $x\#z$, für ein $z \in \Sigma^*$ mit $|z| \leq f(|x|)\} =: V(M')$

Polynomielle Verifizierer: p-Verifizierer, wo p ein Polynom

Proposition: $L \subseteq \Sigma^*$ ist in NP

$\Longleftrightarrow$ es gibt einen polynomiellen Verifizierer für L

„entscheiden" $\neq$ „überprüfen"

Beweisskizze: „$\Longrightarrow$" Es sei L in NP, es gibt eine NDTM M mit $T(M) = L$ und ein Polynom p mit $time_M(x) \leq p(|x|)$. O.E. M hat in jedem Schritt höchstens 2 Möglichkeiten. Betrachte $\Sigma' = \Sigma \cup \{\#, 0, 1\}$. Definiere Verifizierer M', der auf jeder Eingabe $w\#z$ wie M auf w arbeitet, aber in jedem nichtdeterm. Schritt von M interpretiert M' das nächste Bit von z als Instruktion, welche Wahlmöglichkeit getroffen wird.

Zeige: M' Polynomialzeitverifizierer

$L = V(M')$

"$\Leftarrow$": M' polyn. Verifizierer für L.

M sei eine NDTM, die wie folgt arbeitet:

M erhält als Eingabe ein Wort $w \in \Sigma^*$, schreibt dahinter # und danach ein nicht-deterministisch gewähltes Wort z aus Σ'^*.

Danach arbeitet M wie M' angewendet auf die Eingabe $w\#z$.

<u>Zeige</u>: M ist polynomiell in $|w|$, $L = T(M)$

4.2 NP-Vollständigkeit

Definition: Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen.

L_1 ist auf L_2 polynomiell reduzierbar, falls es eine totale und in polynomieller Zeit berechenbare Funktion $f : \Sigma_1^* \longrightarrow \Sigma_2^*$, so dass für alle $w \in \Sigma_1^*$ gilt:

$$w \in L_1 \iff f(w) \in L_2 .$$