Polynomielle Verifizierer und NP

Definition Polynomieller Verifizierer

Sei $L \subseteq \Sigma^*$ eine Sprache. Eine DTM V heißt *Verifizierer für* L, falls V für alle Eingaben $w \in \Sigma^*$ hält und folgendes gilt:

$$w \in L \Leftrightarrow \exists c \in \Sigma^* : V \text{ akzeptiert Eingabe } (w, c).$$

Das Wort c nennt man einen *Zeugen oder Zertifikat für* w.

V heißt *polynomieller Verifizierer für* L, falls für alle $w \in \Sigma^*$ in Laufzeit polynomiell in $|w|$ hält und folgendes gilt:

$$w \in L \Leftrightarrow \exists c \in \Sigma^*, |c| \leq |w|^k, k \in \mathbb{N} : V \text{ akzeptiert Eingabe } (w, c).$$

L ist *polynomiell verifizierbar* $\Leftrightarrow \exists$ polynomieller Verifizierer für L.

Definition Klasse $\mathcal{NP}$

$$\mathcal{NP} := \{L \mid L \text{ ist polynomiell verifizierbar.}\}$$

Polynomieller Verifizierer für RUCKSACK

Satz

RUCKSACK $\in \mathcal{NP}$.

Beweis:

Algorithmus Polynomieller Verifizierer für RUCKSACK

Eingabe: (W, P, B, k, c) mit Zeuge $c = I \subseteq [n]$

1. Falls $\sum_{i \in I} w_i \leq B$ und $\sum_{i \in I} p_i \geq k$, akzeptiere.
2. Lehne ab.

Laufzeit:

- Eingabegrößen: $\log w_i, \log p_i, \log B, \log k, n$
- Laufzeit: $\mathcal{O}(n \cdot \log(\max_i\{w_i, p_i, B, k\}))$ auf RAM.
- D.h. die Laufzeit ist polynomiell in den Eingabegrößen.

Optimaler Wert einer Lösung mittels Entscheidung

RUCKSACK$_{wert}$

- Gegeben: $W = \{w_1, \ldots, w_n\}$ $P = \{p_1, \ldots, p_n\}$ und B.
- Gesucht: $\max_{I \subseteq [n]} \{\sum_{i \in I} p_i \mid \sum_{i \in I} w_i \leq B\}$

Sei M eine DTM, die RUCKSACK in Laufzeit $T(M)$ entscheide.

Algorithmus OPTIMUM

Eingabe: W, P, B

1. $\ell \leftarrow 1, r \leftarrow \sum_{i=1}^{n} p_i$
2. WHILE $(\ell \neq r)$
   1. Falls M bei Eingabe $(W, P, B, \lceil \frac{\ell+r}{2} \rceil)$ akzeptiert, $\ell \leftarrow \lceil \frac{\ell+r}{2} \rceil$.
   2. Sonst $r \leftarrow \lceil \frac{\ell+r}{2} \rceil - 1$.

Ausgabe: ℓ

- Korrektheit: Binäre Suche nach Optimum auf Intervall $[1, \sum_{i=1}^{n} p_i]$.
- Laufzeit: $\mathcal{O}(\log(\sum_{i=1}^{n} p_i)) \cdot T(M)$.
- Insbesondere: Laufzeit ist polynomiell, falls $T(M)$ polynomiell ist.

Optimale Lösung mittels optimalem Wert

Algorithmus Optimale Lösung

Eingabe: W, P, B

1. $opt \leftarrow \text{OPTIMUM}(W, P, B)$, $I \leftarrow \emptyset$
2. For $i \leftarrow 1$ to n
   1. Falls ($\text{OPTIMUM}(W \setminus \{w_i\}, P \setminus \{p_i\}, B) = opt$, setze $W \leftarrow W \setminus \{w_i\}, P \leftarrow \{p_i\}$.
   2. Sonst $I \leftarrow I \cup \{i\}$.

Ausgabe: I

Korrektheit:

- Invariante vor i-tem Durchlauf: $\exists J \subseteq \{i, \ldots, n\}$: $I \cup J$ ist optimal.
- i wird nur dann in I aufgenommen, falls I zu optimaler Teilmenge erweitert werden kann.
- **Laufzeit:** $\mathcal{O}(n \cdot T(\text{OPTIMUM})) = \mathcal{O}(n \cdot \log(\sum_{i=1}^{n} p_i) \cdot T(M))$.
- D.h. Laufzeit ist polynomiell, falls $T(M)$ polynomiell ist.

Sprache Zusammengesetzt

$\text{ZUSAMMENGESETZT} := \{N \in \mathbb{N} \mid N = pq \text{ mit } p, q \geq 2\}$

Satz

ZUSAMMENGESETZT $\in \mathcal{NP}$.

Beweis:

Algorithmus Polynomieller Verifizierer für ZUSAMMENGESETZT

Eingabe: (N, c) mit $c = (p, q) \in \{2, \ldots, N-1\}^2$

1. Berechne $p \cdot q$. Falls $p \cdot q = N$, akzeptiere. Sonst lehne ab.

Laufzeit:

- Eingabelänge: $|N| = \Theta(\log N)$
- Laufzeit: $\mathcal{O}(\log^2 N)$, d.h. polynomiell in der Eingabelänge.

$\mathcal{P}$ versus $\mathcal{NP}$

Satz

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

$$
\begin{aligned}
L \in \mathcal{P} \quad &\Rightarrow \quad \exists \text{ DTM } M, \text{ die } L \text{ in polynomieller Laufzeit entscheidet.} \\
&\Rightarrow \quad \exists \text{ DTM } M, \text{ die stets hält und genau die Eingaben } w \in L \\
&\qquad \text{in Laufzeit polynomiell in } |w| \text{ akzeptiert.} \\
&\Rightarrow \quad \exists \text{ DTM } V, \text{ die stets hält und genau die Eingaben } (w, c) \text{ mit} \\
&\qquad w \in L,\ c = \epsilon \text{ in Laufzeit polynomiell in } |w| \text{ akzeptiert.} \\
&\qquad \text{Dabei ignoriert } V \text{ die Eingabe } c \text{ und verwendet } M \text{ auf } w. \\
&\Rightarrow \quad L \in \mathcal{NP}.
\end{aligned}
$$

- **Großes offenes Problem:** Gilt $\mathcal{P} = \mathcal{NP}$ oder $\mathcal{P} \subset \mathcal{NP}$?

Nichtdeterministische Turingmaschinen

Wir bezeichnen mit $\mathcal{P}(S)$ die Potenzmenge einer Menge S.

Definition Nichtderministische Turingmaschine

Eine *nicht-deterministische Turingmaschine (NTM)* ist ein Tupel $(Q, \Sigma, \Gamma, \delta)$, wobei

- Q, Σ, Γ sind wie bei DTM definiert.
- $\delta : Q \setminus \{q_a, q_r\} \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$

- Bsp: $\delta(q, a) = \{(q_1, a_1, L), (q_2, a_2, R)\}$.
- NTM besitzt Wahlmöglichkeiten für den Zustandsübergang.
- Beschränken uns oBdA auf NTMs mit ≤ 2 Wahlmöglichkeiten.

Berechnungsbaum

- Seien die Konfigurationen einer NTM Knoten in einem Berechnungsbaum.
  - Die Startkonfiguration bildet den Wurzelknoten.
  - Mögliche Nachfolgekonfigurationen bilden Kinderknoten.
- Pfade heißen Berechnungspfade der NTM.
- Betrachten nur NTMs mit Berechnungspfaden endlicher Länge.
- Ein Berechnungpfad heißt akzeptierend, falls er in q_a endet.

Definition Akzeptierte Sprache einer NTM

Sei N eine NTM.

- N akzeptiert Eingabe $w \Leftrightarrow \exists$ akzeptierenden Berechnungspfad im Berechnungsbaum von N bei Eingabe w.
- Die von N akzeptierte Sprache $L(N)$ ist definiert als $L(N) = \{w \in \Sigma^* \mid N \text{ akzeptiert die Eingabe } w.\}$.

Die Laufzeit einer NTM

Definition Laufzeit einer NTM

Sei N eine DTM mit Eingabe w.

- $T_N(w) :=$ **maximale** Anzahl Rechenschritte von N auf w, d.h. $T_N(w)$ ist die Länge eines längsten Berechnungspfades.
- $T_N : \mathbb{N} \to \mathbb{N}$, $T_N(n) := \max\{T_N(w) \mid w \in \Sigma^{\leq n}\}$ heißt *Laufzeit* oder Zeitkomplexität von N.

- Wir definieren die Klasse NTIME für NTMs analog zur Klasse DTIME für DTMs.

Definition NTIME

Sei $t : \mathbb{N} \to \mathbb{N}$ eine monoton wachsende Funktion.

$$\text{NTIME}(t(n)) := \{L \mid L \text{ wird von NTM in Laufzeit } \mathcal{O}(t(n)) \text{ entschieden.}\}$$

NTM, die RUCKSACK entscheidet

Algorithmus NTM für RUCKSACK

Eingabe: W, P, B, k

1. Erzeuge nichtdeterministisch einen Zeugen $I \subseteq [n]$.
2. Falls $\sum_{i \in I} w_i \leq B$ und $\sum_{i \in I} p_i \geq k$, akzeptiere.
3. Sonst lehne ab.

- D.h. NTM erzeugt sich im Gegensatz zum Verifizierer ihren Zeugen I selbst.
- Laufzeit: Schritt 1: $\mathcal{O}(n)$, Schritt 2: $\mathcal{O}(n \cdot \log(\max_i\{w_i, p_i\}))$.
- D.h. die Laufzeit ist polynomiell in der Eingabelänge.

$\mathcal{NP}$ mittels NTMs

Satz

$\mathcal{NP}$ ist die Klasse aller Sprachen, die von einer NTM in polynomieller Laufzeit entschieden wird, d.h.

$$\mathcal{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k).$$

Zeigen:

$\exists$ polynomieller Verifizierer für L

$\Leftrightarrow$ $\exists$ NTM N, die L in polynomieller Laufzeit entscheidet.

Verifizierer $\Rightarrow$ NTM

"$\Rightarrow$": Sei V ein Verifizierer für L mit Laufzeit $\mathcal{O}(n^k)$ für ein festes k.

Algorithmus NTM N für L

Eingabe: w mit $|w| = n$.

1. Erzeuge nicht-deterministisch einen Zeugen c mit $|c| = \mathcal{O}(n^k)$.
2. Simuliere V mit Eingabe (w, c).
3. Falls V akzeptiert, akzeptiere. Sonst lehne ab.

- Korrektheit:
  $$w \in L \quad \Leftrightarrow \quad \exists c \text{ mit } |c| = \mathcal{O}(n^k) : V \text{ akzeptiert } (w, c).$$
  $$\Leftrightarrow \quad N \text{ akzeptiert die Eingabe } w \text{ in Laufzeit } \mathcal{O}(n^k).$$
- Damit entscheidet N die Sprache L in polynomieller Laufzeit.

NTM $\Rightarrow$ Verifizierer

"$\Leftarrow$: Sei N eine NTM, die L in Laufzeit $\mathcal{O}(n^k)$ entscheidet.

Algorithmus Verifizierer

Eingabe: w, c

1. c ist Kodierung eines Berechnungspfades von N bei Eingabe w.
2. Simuliere N auf Eingabe w auf dem Berechnungspfad c.
3. Falls N akzeptiert, akzeptiere. Sonst lehne ab.

Korrektheit:

$$\begin{aligned} w \in L &\Leftrightarrow \exists \text{ akzeptierender Berechnungspfad } c \text{ von } N \text{ für } w \\ &\Leftrightarrow V \text{ akzeptiert } (w, c). \end{aligned}$$

Laufzeit:

- Längster Berechnungspfad von N besitzt Länge $\mathcal{O}(n^k)$.
- D.h. die Gesamtlaufzeit von V ist ebenfalls $\mathcal{O}(n^k)$.

Boolesche Formeln

Definition Boolesche Formel

- Eine Boolesche Variable x_i kann Werte aus $\{0, 1\}$ annehmen, wobei $0 \cong$ falsch und $1 \cong$ wahr.
- Jede Boolesche Variable x_i ist eine Boolesche Formel.
- Sind ϕ, ϕ' Boolesche Formeln, so auch $\neg\phi, \phi \wedge \phi', \phi \vee \phi'$.
- Operatoren geordnet nach absteigender Priorität: $\neg, \wedge, \vee$.
- ϕ ist erfüllbar $\Leftrightarrow \exists$ Belegung der Variablen in ϕ mit $\phi = 1$.

Bsp:

- $\phi = \neg(x_1 \vee x_2) \wedge x_3$ ist erfüllbar mit $(x_1, x_2, x_3) = (0, 0, 1)$.
- $\phi' = x_1 \wedge \neg x_1$ ist eine nicht-erfüllbare Boolesche Formel.

Satisfiability SAT

Definition SAT

SAT $:= \{\phi \mid \phi$ ist eine erfüllbare Boolesche Formel.$\}$

Kodierung von ϕ:

- Kodieren Variable x_i durch $\mathrm{bin}(i)$.
- Kodieren ϕ über dem Alphabet $\{0, 1, (,), \neg, \wedge, \vee\}$.