Komplexität

Die Komplexitätstheorie unterteilt die entscheidbaren Sprachen/Probleme in Komplexitätsklassen bzgl. den Berechnungsressourcen (Zeit, Speicher) einer Turing-Maschine, die das Problem entscheidet.

Platz $\in \mathcal{O}(|x|^k)$

z.B. $\underbrace{P \subseteq NP}_{\text{Zeit} \in \mathcal{O}(|x|^k)} \subseteq PSPACE \subseteq \underbrace{EXPTIME \subseteq NEXPTIME}_{\text{Zeit} \in \mathcal{O}(2^{|x|^k})}$ ... & VIELE mehr

https://complexityzoo.net
(aktuell: 546 Klassen)

Wir beschränken uns auf P & NP!

O-Notation:

$$\mathcal{O}(f) := \{ g : \mathbb{N} \to \mathbb{N} \mid \exists C>0 \ \exists n_0 \in \mathbb{N} \ \forall n > n_0 : |g(n)| \leq C\,|f(n)| \}$$

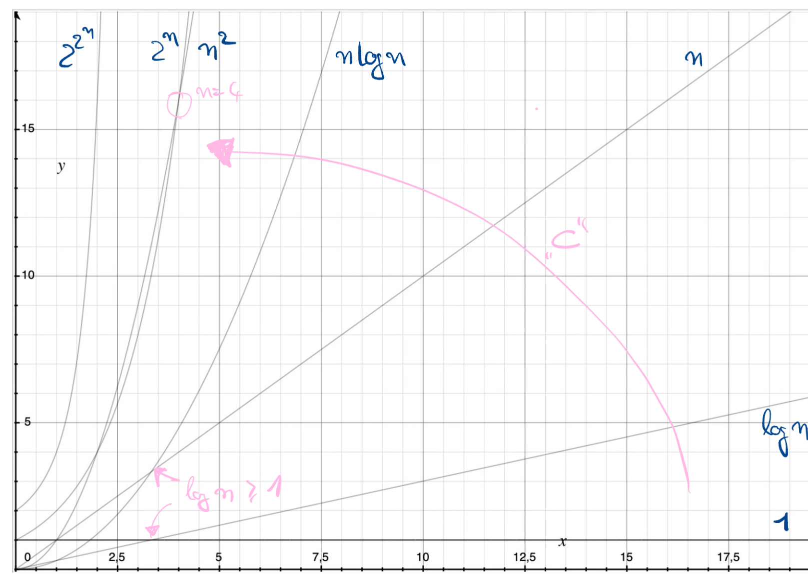
$\Downarrow$

$h \in \mathcal{O}(f)$ bedeutet h wächst mit n höchstens so schnell wie f

Die für uns wichtigen Klassen sind geordnet bzgl. „$\subset$":

man schreibt vereinfacht $\mathcal{O}(f(n)) := \mathcal{O}(f)$

$$\mathcal{O}(1) \subset \mathcal{O}(\log n) \subset \mathcal{O}(n) \subset \mathcal{O}(n \log n) \subset \cdots$$
$$\subset \mathcal{O}(n^2) \subset \cdots \subset \mathcal{O}(n^k) \subset \mathcal{O}(2^n) \subset \cdots$$
$$\subset \mathcal{O}(2^{2^n}) \subset \cdots$$



P

DEF

- $(d)time_M(x) :=$ Anzahl der δ-Übergänge der (Mehrband-) Turing-Maschine M bei Input x
  $\in \mathbb{N} \cup \{\infty\}$
  (d.h. „Länge" von $z_0 x \vdash \cdots \vdash \alpha z_e \beta$ oder ∞)
- $t_M(n) := \underset{(\sup)}{\max} \{ time_M(x) \mid x \in \Sigma^*, |x| \le n \}$
- $DTIME(f) :=$ alle Sprachen entscheidbar durch eine TM M mit $t_M \in O(f)$
  „deterministisch"
  $= \{ L \subset \Sigma^* \mid \exists M, C : L = T(M) \text{ s.d. } \forall x \; time_M(x) \le C f(|x|) \}$
  = alle ~~sem~~ entscheidbaren Sprachen mit Laufzeit $O(f)$

$\Downarrow$

$$P := \bigcup_{k=0}^{\infty} DTIME(n \mapsto n^k) = \bigcup_{p} DTIME(p)$$
p Polynom

Entscheidungsprobleme polynomialer Laufzeit
(gemessen an der Länge $|x|$)

BEISPIEL

„Selection sort"

5 1 3 4 (min)
1 5 3 4 (min)
1 3 5 4 (min)
1 3 4 5

$$(n-1) \times \underbrace{\min(\dots)}_{\in O(n)} \in O(n^2)$$

$time_M(x)$ wird in der Praxis abgeschätzt aus der Anzahl der Rechenschritte eines gegebenen Algorithmus:

1 Zugriff / Operation auf $y \in \mathbb{N}$ $\propto$ 1 Zeitschritt (uniformes Kostenmaß)

$\downarrow$

Funktioniert gut wenn die Zahlen eine konstante obere Schranke nicht überschreiten

d.h. konstant bzgl. Input-Größe $|x|$

[z.B. Sortieralgorithmen:
$x = \underbrace{(5,1,3,4,5,\dots)}_{\text{Länge } |x|}$, Größe von $5, 1, \dots$ unabh. von $|x|$]

BEISPIEL

INPUT (x)
$n := |x|$ (Länge der Binärdarstellung von x) $\mathcal{O}(n)$
$y := 2$
LOOP n DO $y = y * y$ END; $\mathcal{O}(n)$
OUTPUT (y)

$\Big\}\ \mathcal{O}(n)$

$n=1: \quad 2^2$

$n=2: \quad (2^2)^2 = 2^2 \cdot 2^2 = 2^{2+2} = 2^4 = 2^{(2^2)}$

$n=3: \quad ((2^2)^2)^2 = 2^{2^2} \cdot 2^{2^2} = 2^{2^2+2^2} = 2^{2 \cdot 2^2} = 2^{(2^3)}$

$\vdots$

$\Rightarrow$ Der Algorithmus berechnet $\underbrace{(((2^2)^2)^{\cdots})^2}_{|x|\text{-fach}} = 2^{(2^{|x|})}$

Bei 1-facher Zählung von „$y = y*y$" wäre der Algorithmus $\mathcal{O}(n)$

Aber: Alleine das Hinschreiben der Binärdarstellung von 2^{2^n} benötigt 2^n Schritte $\left(|2^{2^n}| = \log_2(2^{2^n}) = 2^n \right) \Rightarrow\ \geq \mathcal{O}(2^n)$

Problem: Die Zahlen $y \in \mathbb{N}$ in der LOOP-Schleife haben keine obere Schranke unabhängig von $|x|$!

Unter Berücksichtigung der Größe der Operanden:

1 Zugriff / Operation auf $y \in \mathbb{N}$ $\quad \propto \quad \lceil \log_2 y \rceil$ Zeitschritte (logarithmisches Kostenmaß)

[Kosten proportional zur Länge der Binärdarstellung]

BEM

(Obere Schranken an die) Komplexität häufiger Operationen:

Für $x, y \in \mathbb{N}$ mit $|x|, |y| \leq n$ $\quad (|x| = \lceil \log_2 x \rceil)$

$x+y$	$\mathcal{O}(n)$
$x \cdot y$	$\mathcal{O}(n^2)$ [Harvey-Hoeven-Multiplikation (2019): $\mathcal{O}(n \log n)$]
$p(x)$ (Polynom)	$\mathcal{O}(p(n))$
$\sqrt{x}$	$\mathcal{O}(n^2)$ (bzw. $\mathcal{O}(n \log n)$)
[Matrixprodukt ($n \times n$-Matrizen) Hier: Eingabegröße $\hat{=}$ „Matrix-Größe"]	$\mathcal{O}(n^3)$ $\quad \underbrace{\left((AB)_{ik} = \underbrace{\sum_{j=1}^{n} A_{ij} B_{jk}}_{\mathcal{O}(n) \text{ falls Zahlen nicht mit } n \text{ wachsen}} \right)}_{n^2\text{-mal benötigt}}$]

HINWEIS

Für uns in dieser Vorlesung sind die genauen Ordnungen nicht entscheidend, sondern nur $\mathcal{O}(\text{Polynom})$ oder nicht.

NP

DEF Eine nicht-deterministische Turing-Maschine (NTM) ist eine Turing-Maschine mit Übergangsfunktion der Form

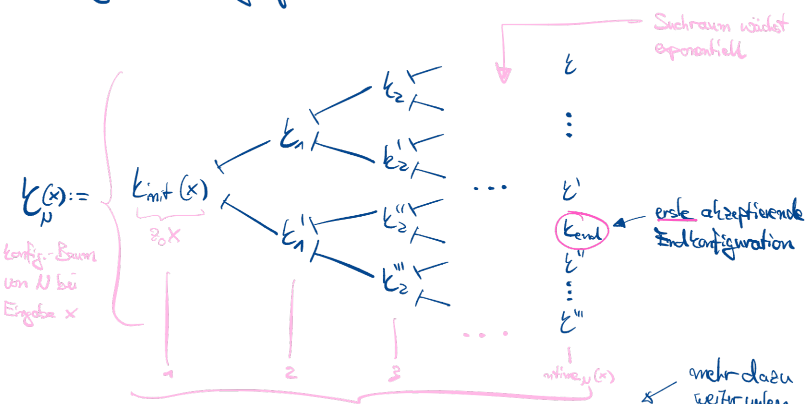
$$\delta : Z \times \Gamma \longrightarrow \mathcal{P}(Z \times \Gamma \times \{L, R, N\})$$

d.h. ein Aufruf von δ kann mehrere Aktionen auslösen, z.B.

$$\delta(z,a) = \{ (z', b, R), (z'', c, L) \}$$

$\downarrow$

Konfigurationsübergänge bilden (z.B. binären) Baum



$=: ntime_N(x)$ (min. akz. Schrittzahl)

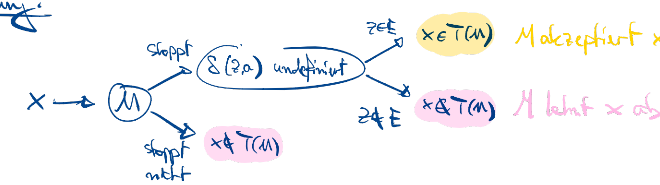
DEF Die von einer NTM N akzeptierte Sprache ist definiert durch

$$T(N) := \{ x \in \Sigma^* \mid \exists \text{ akzeptierende Endkonfiguration in } \mathcal{K}_N(x) \}$$

BEM Vergleiche mit deterministischem Fall

$$T(M) = \{ x \in \Sigma^* \mid \text{der (eindeutige) Berechnungspfad von } M \text{ startend in } z_0x \text{ endet mit einer akzeptierenden Endkonfiguration} \}$$

Erinnerung:



deterministisch

$x \in T(M)$

$k_{init}(x) \vdash \dots \vdash k_{end}$ akzeptierend

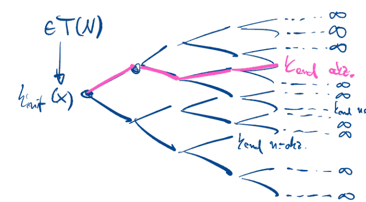
$x \notin T(M)$

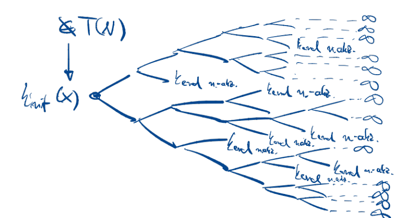
$k_{init}(x) \vdash \dots \dots \infty$

oder

$k_{init}(x) \vdash \dots \vdash k_{end}$ nicht-akz.

nicht-deterministisch





BEM

Unterschied im Sprachgebrauch (gilt für TMs & NTMs):

	vs	
„A akzeptierbar durch eine TM"		„A entscheidbar durch eine TM"
$\Leftrightarrow$		$\Leftrightarrow$
„A semi-entscheidbar"		„A decidable"
$\Leftrightarrow$		
„A recognizable"		
$\exists M$ mit $A = T(M)$		$\exists M$ mit $A = T(M)$ ($\rightsquigarrow$ Ausgabe $= 1$) und M stoppt immer
es kann sein, dass M nicht stoppt wenn $x \notin T(M)$		(k_{end} nicht-akz. $\rightsquigarrow$ Ausgabe $= 0$)

SATZ

A (semi-) entscheidbar durch eine TM $\Leftrightarrow$ A (semi-) entscheidbar durch eine NTM

BEWEIS

"$\Rightarrow$" trivial, da jede TM eine NTM ist (TMs $\subset$ NTMs)

"$\Leftarrow$" Jede NTM N kann durch eine TM M simuliert werden s.d. „N stoppt" $\Leftrightarrow$ M stoppt

($\exists$ stoppende Endkonfig.) ($\exists$ Band mit stoppender Endkonfig.)

indem jede Verzweigung auf einem neuen Band weitergeführt wird ∎

DEF

Für eine NTM N und alle $x \in T(N)$ sei

$$l_N(x) := \text{Länge des kürzesten akzeptierenden Berechnungspfads in } K_N(x)$$

Für $n \in \mathbb{N}$ sei

$$t_N(n) := \left.\begin{array}{l}\text{größtes } l_N(x) \text{ aller von } N \text{ akzeptierten} \\ \text{Eingaben der Länge kleiner gleich } n\end{array}\right\} = \max_{\substack{x \in T(N) \\ |x| \le n}} l_N(x)$$

Für eine Funktion $f: \mathbb{N} \to \mathbb{N}$ sei

$$NTIME(f) := \text{Alle durch NTMs akzeptierbaren Sprachen mit } t_N \in \mathcal{O}(f)$$
$$= \{ L \subset \Sigma^* \mid \exists \text{ NTM } N \text{ mit } t_N \in \mathcal{O}(f) \ \& \ L = T(N) \}$$

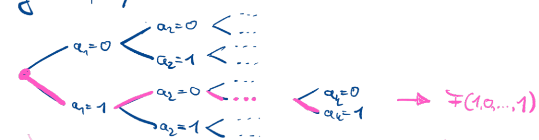
$$\boxed{NP := \bigcup_{k=0}^{\infty} NTIME(n \mapsto n^k) = \bigcup_{p \text{ Polynom}} NTIME(p)}$$

BEISPIEL

$$SAT = \left\{ \begin{array}{l} F \text{ Formel der} \\ \text{Aussagenlogik} \end{array} \middle| \begin{array}{l} \exists \text{ Belegung } (a_1, \dots, a_k) \in \{0,1\}^k : \\ F(a_1, \dots, a_k) = 1 \end{array} \right\} \in NP$$

2^k mögliche Belegungen

Entscheidung mithilfe systematischer Suche:



Wahl der Belegung $\rightarrow$ k Schritte

Auswertung $\mathcal{O}(|F|)$

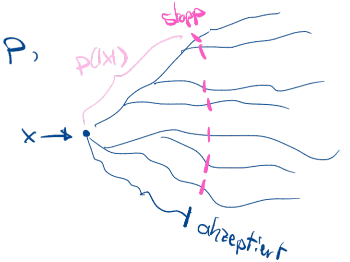
[später genauer]

REM ①

$$\boxed{NP \subset \{L \subset \Sigma^* \mid L \text{ berechenbar}\}}$$

Obwohl durch $t_N \in \mathcal{O}(\text{Polynom})$ nur die Laufzeit für $x \in T(N)$ beschränkt wird, können wir N so manipulieren, dass jeder Berechnungspfad polynomiell beschränkt wird:

Falls $t_N(n) \le p(n)$ für ein Polynom p, dann sei N' eine NTM die Pfade, in $t_N(x)$ stoppt, falls diese nach $p(|x|)$ Schritten nicht akzeptieren.



↓

Mehraufwand:

- Bestimmen von $p(|x|)$ $<$ $n = |x| \leftarrow \mathcal{O}(n)$; $p(n) \leftarrow \mathcal{O}(p(n))$
- Mitzählen der Schrittzahl ($\underbrace{\text{„+1"}}_{\mathcal{O}(\log p(n))}$ $p(|x|)$-mal) $\leftarrow \mathcal{O}(p(n) \log p(n)) \subset \mathcal{O}(p(n)^2)$

$\underbrace{\qquad\qquad}_{\mathcal{O}(\text{Polynom})}$

⇓

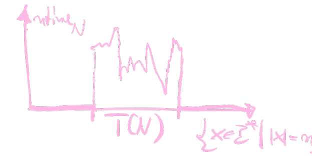
N' hat weiterhin polynomielle Komplexität.

REM ②

Man definiert manchmal auch (z.B. im Skript) „$\text{ntime}_N(x)$" $\forall x$ (analog zu dtime_M) anstelle von $\ell_N(x)$ nur für $x \in T(N)$:

$$\text{ntime}_N(x) = \begin{cases} \ell_N(x) & , x \in T(N) \\ \text{Konstante} & , x \notin T(N) \end{cases}$$

z.B. im Skript $= 0$



Die Wahl der Konstante ist nicht von Bedeutung, da ntime_N lediglich im folgenden äquivalenten Df. von NTIME auftaucht:

$$\text{NTIME}(f) = \left\{ L \subset \Sigma^* \;\middle|\; \begin{array}{l} \exists \text{ NTM } N, \exists C > 0 \\ L = T(N) \;\&\; \underbrace{\forall x \;\; \text{ntime}_N(x) \le C f(|x|)}_{} \end{array} \right\}$$

hier wird Schrittzahl nur für $x \in T(N)$ beschränkt, genauso wie eben durch $t_N \in \mathcal{O}(f)$

nur für $x \in T(N)$ def. $\rightarrow \ell_N(x) \le C f(|x|)$

REM ③

Ein einfaches Argument („guess & check") um $L \in NP$ zu zeigen:

Eine Sprache L für die es ein L' gibt sodass

$$L = \{x \mid \exists y \text{ s.d. } (x,y) \in L'\}$$

„Lösung" (z.B. Belegung bei SAT) — „Auswertung" ob y zu x passt (z.B. $F(a_1, \dots, a_k) = 1$?)

ist in NP, falls

- die Lösungen y der Länge $|y| \le p(|x|)$ (für ein Polynom p) sind
- die Auswertungen polynomiell sind, also falls $L' \in P$

Denn das folgende nicht-deterministische Verfahren entscheidet L:

1. Generiere eine mögliche Lösung y (guess) ← $|y| \leq p(|x|)$ nicht-determ. Schritte

2. Überprüfe ob (x,y) gewünschte Eigenschaft (L') erfüllt (check) ← deterministisch $\in P$