

Nerozhodnutelné problémy

Marta Vomlelová

MS 303

Časová složitost

Definition 12.24 (časová složitost)

Mějme Turingův stroj M , který zastaví na každém vstupu. **Časová složitost** M je funkce $f : \mathbb{N} \rightarrow \mathbb{N}$, kde $f(n)$ je maximální počet kroků výpočtu M nad vstupy délky n .

Definition 12.25 ((Asymptotická) horní hranice $O(g(n))$)

Mějme funkce $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. Říkáme, že $f(n) \in O(g(n))$, pokud existují $c, n_0 \in \mathbb{N}^+$ taková, že:

$$\forall n \geq n_0 \text{ platí } f(n) \leq c \cdot g(n).$$

V takovém případě říkáme, že $g(n)$ je (asymptotická) **horní hranice** pro $f(n)$. (Slovo asymptotická vyjadřující ignorování prvních n_0 i konstanty c se zpravidla vynechává.)

- Reálná čísla jsou tam kvůli logaritmu.
- Např. $f(5n^3 + 2n^2 + 22n + 6) \in O(n^3)$ s $n_0 = 10$, $c = 6$.

Definition 12.26 (třída časové složitosti)

Mějme funkci $t : \mathbb{N} \rightarrow \mathbb{R}^+$. Definujeme **třidu časové složitosti** $TIME(t(n))$ jakožto množinu všech jazyků, které jsou rozhodnutelné jednopáskový Turingovým strojem v čase $O(t(n))$ tj. vždy zastaví, pro vstup délky n nejpozději po $O(t(n))$ krocích, a vydá správnou odpověď.

Example 12.26 ($\{0^i 1^i \mid i \in \mathbb{N}_0\}$ je $O(n^2)$)

Jazyk $\{0^i 1^i \mid i \in \mathbb{N}_0\}$ je $O(n^2)$

- ➊ Zkontroluj vstup $0^i 1^j$, pokud za 1 je 0, nepřijmi (čas $O(n)$)
- ➋ návrat na začátek se schová v konstantě, $O(2n) = O(n)$
- ➌ procházej postupně 0 v čase $O(n^2)$
 - ➊ přepiš 0 na X
 - ➋ najdi 1 a přepiš na X
 - ➌ vrať se na začátek
- ➍ Když už není 0, ověř že není ani 1 a přijmi (s 1 nepřijmi). (čas $O(n)$)

Jde to rychleji?

Example 12.27 ($\{0^i 1^i \mid i \in \mathbb{N}_0\}$ je $O(n \log n)$)

Jazyk $\{0^i 1^i \mid i \in \mathbb{N}_0\}$ je $O(n \log n)$

- ➊ Zkontroluj vstup $0^i 1^j$, zkontroluj sudou délku (čas $O(n)$)
- ➋ procházej dokud najdeš 0 v čase ($O(n \log n)$)
 - ➊ přepiš každou druhou 0 na X
 - ➋ přepiš každou druhou 1 na X
 - ➌ zkontroluj sudost počtu nul a jedniček dohromady, pokud ne, nepřijmi.
 - ➍ a vrať se na začátek
- ➎ Když už není 0, ověř že není ani 1 a přijmi (s 1 nepřijmi). (čas $O(n)$)

Regulární jazyky - jen pro zajímavost

- O moc rychleji to nejde.

Definition ($o()$)

Mějme funkce $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. Říkáme, že $f(n) = o(g(n))$, pokud

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$$

Tj. pro každé $c > 0$ existuje n_0 takové, že $(\forall n > n_0) (f(n) < cg(n))$.

- Malé o má význam 'ostře menší', velké O menší nebo rovno.

Theorem (just for info)

Každý jazyk rozhodnutelný v čase $o(n \log n)$ na jednopáskovém Turingově stroji je regulární.

Nedeterministický Turingův stroj

Definition 12.27 (doba běhu nedeterministického TM)

Mějme **nedeterministický** Turingův stroj M , který zastaví na každém vstupu.

Doba běhu M je funkce $f : \mathbb{N} \rightarrow \mathbb{N}$, kde $f(n)$ je maximální počet kroků který M potřebuje v jakékoli větvi výpočtu nad jakýmkoli vstupem délky n .

O takovém nedeterministickém Turingovu stroji M říkáme, že **rozhoduje** jazyk $L(M)$ v čase $f(n)$.

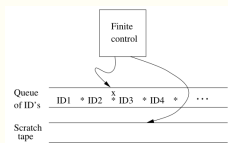
Theorem 12.24

Mějme funkci $t : \mathbb{N} \rightarrow \mathbb{R}^+$, $t(n) \geq n$. Každý nedeterministický Turingův stroj s časem $t(n)$ má deterministický ekvivalent $2^{O(t(n))}$.

- Pokud pro dvojici $Q \times \Gamma$ máme maximálně d variant, pak po k krocích se TM může dostat maximálně do d^k konfigurací.
- Jeden krok zvládneme 'schovat do konstanty' k $t(n)$, logaritmus d pro převod také, proto simulace je v čase $O(t(n)d^{t(n)}) = 2^{O(t(n))}$.
- Máme přidat převod simulace více pásek, ale $(2^{O(t(n))})^2 = 2^{O(2t(n))} = 2^{O(t(n))}$.

Proof: opakování - idea důkazu

- páska nekonečná – nelze použít podmnožinovou konstrukci
- prohledáváme do šířky všechny výpočty M_N
- modelujeme TM se dvěma páskami
 - ▶ první páska: posloupnost konfigurací
 - ★ aktuální označena (křížkem na obrázku)
 - ★ vlevo už prozkoumané, můžeme zapomenout
 - ★ vpravo aktuální a pak další čekající
 - ▶ druhá páska: pomocný výpočet
- zpracování jedné konfigurace obnáší
 - ▶ přečti stav a symbol aktuální konfigurace ID
 - ▶ je-li stav přijímající $\in F$, přijmi a skonči
 - ▶ napiš konfiguraci ID na pomocnou pásku
 - ▶ pro každý možný krok δ (uložený v hlavě M_D)
 - ★ proved' krok a napiš novou ID na konec první pásky
 - ▶ vrať se k označené ID, značku vymaž a posuň o 1 doprava
 - ▶ opakuj



Opakování - Jednosměrná páska

X_0	X_1	X_2	$\dots$
*	X_{-1}	X_{-2}	$\dots$

Lemma (Jednosměrná páska)

Pro každý Turingův stroj M_2 existuje Turing. stroj M_1 , který přijímá stejný jazyk a

- M_1 nikdy nejde vlevo od počáteční pozice
- M_1 nikdy nepíše blank B .

Proof.

- Místo B blank zavedeme nový páskový symbol, B_1 .
 - ▶ Místo každého psaní B píšeme B_1 a všechny instrukce pro čtení B zkopírujeme též pro čtení B_1 .
 - ▶ Takto modifikovaný TM nikdy nepíše B (píše B_1).
- Pro jednosměrnou pásku
 - ▶ Nejdřív přepíšeme vstup, aby byl v horní stopě dvoustopé pásky.
 - ▶ Pod nejlevější symbol dáme nový znak $*$, abychom věděli, že jsme na levém okraji, a máme přepnout z horní stopy do dolní. Ve stavu ('hlavě') si pamatujeme, jestli čteme horní stopu ('normálně') nebo spodní stopu (kde L znamená doprava a R doleva). Pokud vidíme $*$, odpovídající instrukce přepíšeme na změnu stopy (zhora vlevo přepnu na dolů, pokud jsem dole přešla na $**$, mám rovnou číst horní symbol a směřovat dle horní pásky).

Polynomiální problémy P

Definition 12.28 (třída P)

Definujeme P ($PTIME$) **třidu jazyků rozhodnutelných v polynomiálním čase** jednopáskovým deterministickým Turingovým strojem. Tedy:

$$P = \bigcup_k TIME(n^k).$$

Theorem 12.25 ($CFL \subseteq P$)

Každý bezkontextový jazyk patří to P .

Proof.

- Gramatiku převedeme do ChNF (velikost nezávisí na n),
- CYK algoritmus je polynomiální ($O(n^3)$).



Příklady problémů třídy P , NP

Example 12.28 (Nalezení cesty v grafu (s minimální cenou) $\in P$)

- Cesta v grafu $PATH$
 - ▶ Dijkstrův algoritmus $O(|V|^2 + |E|)$

Example 12.29 (Jsou x, y nesoudělná? $\in P$)

- Nesoudělná čísla $RELPRIME$
 - ▶ Repeat until $y = 0$
 - ▶ $x \leftarrow x \bmod y$
 - ▶ exchange x and y
 - ▶ Return x .

Example 12.30 ($HAMPATH \in NP$ (viz později))

Hamiltonovská cesta je taková (orientovaná) cesta, která obsahuje každý vrchol grafu právě jednou.

Kdybychom tu cestu měli, umíme ověřit v polynomiálním čase. Cesta = certifikát.

Definition 12.29 (Verifikátor)

- **Verifikátor** jazyka L je algoritmus V , kde:

$$L = \{w \mid V \text{ pro nějaký řetězec } c \text{ přijímá } \langle w, c \rangle\}.$$

- Náповěda c pro snadné ověření se nazývá **certifikát**.
- Časová složitost verifikátoru se měří pouze vzhledem k délce w , **polynomiální verifikátor** rozhoduje v čase polynomiálním vzhledem k $|w|$.
- Jazyk L je **polynomiálně verifikovatelný**, pokud má polynomiální verifikátor. Pak i certifikát vždy existuje i polynomiální, delší by verifikátor nestihl ani přečíst.

Definition 12.30 (NP)

Třída jazyků rozhodnutelných v polynomiálním čase *NP* je tvořena jazyky s polynomiálním verifikátorem.

Hamiltonovská cesta

Example 12.31

$HAMPATH = \{\langle G, s, t \rangle \mid G \text{ je orientovaný graf s hamiltonovskou cestou z } s \text{ do } t\}$.

Hamiltonovská cesta je taková (orientovaná) cesta, která obsahuje každý vrchol grafu právě jednou.

- Složitost u grafů bereme vůči počtu uzlů, počet hran je max. kvadratický, tj. polynomiální.
- Náš certifikát je posloupnost vrcholů cesty.
- Algoritmus v polynomiálním čase ověří, že jde o hamiltonovskou cestu.
- Pro $\overline{HAMPATH}$, neznáme verifikátor (jen umíme ověřit v exponenciálním čase).

($\overline{HAMPATH}$ je množina (neplatných vstupů $HAMPATH$ a) orientovaných grafů s dvojicí vrcholů s, t , pro které neexistuje hamiltonovská cesta z s do t .)

Třídy $NTIME$

Definition 12.31 (NP)

Mějme funkci $t : \mathbb{N} \rightarrow \mathbb{R}^+$. Definujeme třídu

$$NTIME(t(n)) = \{L \mid L \text{ jazyk rozhodnutelný nedeterminist. TM v čase } O(t(n)).\}$$

Definition (definice)

Třída jazyků rozhodnutelných v polynomiálním čase (značíme NP) je tvořena jazyky s polynomiálním verifikátorem.

Theorem 12.26

$$NP = \bigcup_k NTIME(n^k).$$

- Idea důkazu: převedeme verifikátor na NTM a opačně.
- NTM uhodne certifikát a simuluje verifikátor.
- Verifikátor bere přijímající větev NTM jakožto certifikát.

verif. $\Rightarrow \cup$.

- Předpokládáme $L \in NP$.
- Hledáme nedeterministický TM M .
- Vezmeme verifikátor V z definice NP . Nechť rozhoduje L v čase n^k .
- M na vstupu w délky n :
 - ▶ Nedeterministicky uhodne řetězec c délky nanejvýš n^k .
 - ▶ Spustí V na vstupu $\langle w, c \rangle$.
 - ▶ Pokud V přijme, M také přijme.



$\cup \Rightarrow$ verifikátor.

- Předpokládáme $L = L(M)$ rozhodnutelný nějakým NTM M v polynomiálním čase.
- Hledáme verifikátor V .
- Certifikát c nám v každém kroku řekne, kterou z možných akcí/větvi volit.
- V na vstupu $\langle w, c \rangle$:
 - ▶ Simuluje M na vstupu w , v bodech větvení vybere větev podle c .
 - ▶ Pokud tato větev NTM přijme, V přijme.
- Pokud všechny větve selhaly, NTM nepřijímá.

Example 12.32 (*CLIQUE* je NP)

Problém k -kliky *CLIQUE* je v daném grafu G určit, jestli existuje klika velikosti k , tj.

$$CLIQUE = \{ \langle G, k \rangle \mid G \text{ je neorientovaný graf s klikou velikosti } k \}.$$

Verifikátor pro *CLIQUE*

- 1: **procedure** CLIQUE_VERIFIER($\langle G, k \rangle, c$)
- 2: Otestuj, jestli c je podgraf G o k uzlech.
- 3: Otestuj, jestli je c klika, tj. úplný podgraf, má všechny hrany.
- 4: **return** Oba předchozí testy TRUE?
- 5: **end procedure**

Nedeterministický Turingův stroj pro *CLIQUE*

- 1: **procedure** CLIQUE_NTM($\langle G, k \rangle$)
- 2: Nedeterministicky vyber k prvkovou podmnožinu c vrcholů G .
- 3: **return** Je c klika? tj. úplný podgraf, má všechny hrany.
- 4: **end procedure**

Převoditelnost v polynomiálním čase

Definition 12.32 (Polynomiální převoditelnost)

Funkci $f : \Sigma^* \rightarrow \Sigma^*$ nazveme **polynomiálně vyčíslitelnou**, pokud existuje Turingův stroj M , který pro každý vstup w v polynomiálním čase zastaví s $f(w)$ na pásce.

Jazyk A je **převoditelný v polynomiálním čase** na jazyk B , $A \leq_P B$, pokud existuje funkce $f : \Sigma^* \rightarrow \Sigma^*$ vyčíslitelná v polynomiálním čase a pro každé $w \in \Sigma^*$

$$w \in A \Leftrightarrow f(w) \in B.$$

Funkci f pak nazýváme **polynomiální redukcí A do B** .

Definition 12.33 (SAT, 3SAT)

Formuli ϕ nazveme **3-cnf formule**, pokud je formule výrokové logiky v CNF, kde v každé klauzuli jsou nejvýše tři literály.

Formule **je splnitelná**, pokud existuje takové ohodnocení výrokových proměnných, že je hodnota formule TRUE.

Problém **3SAT** je pro každou 3-cnf formuli rozhodnout, zda je splnitelná, tj.

$$3SAT = \{\phi \mid \phi \text{ je splnitelná 3-cnf formule}\}.$$

Problém **SAT** je pro každou booleovskou formuli rozhodnout, zda je splnitelná

$$SAT = \{\phi \mid \phi \text{ je splnitelná formule}\}.$$

Theorem 12.27

3SAT je polynomiálně převoditelný na CLIQUE.

Proof.

- Každý výskyt proměnné - uzel grafu.
- Hrany všude, jen ne:
 - ▶ mezi uzly z téže klauzule
 - ▶ mezi proměnnou a její negací x a $\neg x$.
 - ▶ k počet klauzulí, počet vrcholů v klíci.

NP úplnost

Definition 12.34 (NP úplnost)

Jazyk B je **NP úplný**, pokud je NP a každý jazyk $A \in NP$ je na B polynomiálně převoditelný.

Theorem 12.28

Pokud B je NP-úplný a $B \in P$, pak $P = NP$.

Proof.

Přímý důsledek definice polynomiální převoditelnosti a NP . □

Theorem 12.29

Pokud B je NP-úplný a $B \leq_P C$ pro nějaké $C \in NP$, pak C je NP-úplný.

Proof.

Převod problému na B dále převedeme na C , stačí polynomiální čas. □

Theorem 12.30 (Cook-Levin-ova věta)

SAT je NP-úplný.

- idea důkazu úplnosti: převedeme výpočet nedeterministického Turingova stroje na SAT.

- SAT is NP
 - ▶ Nedeterministický TM uhodne správné ohodnocení a v polynomiálním čase ověří, je pro něj formule pravdivá.
- SAT je NP-úplný
 - ▶ Vezmeme libovolný $L \in NP$
 - ▶ nechť M je nedeterministický TM který rozhoduje jazyk L v čase $n^k - 3$ pro nějaké k
 - ▶ (zde pro jednoduchost NTM s jednostrannou páskou)
 - ▶ Vytvoříme tabulku (tableau) $n^k \times n^k$, každý řádek odpovídá konfiguraci M na vstupu w
 - ▶ můžeme předpokládat (opatřit) každou konfiguraci ohraničenou zarážkami #.

#	q_0	w_1	w_2	...	w_n	_	_	...	#
#									#
#	⋮								#
#	⋮								#
#									#

- ▶ Výpočet budeme skládat po okýnkách 2×3 .

- Vybraná dovolená okénka, $(a, b, c, d \in \Gamma)$

$$\delta(q_1, b) \ni (q_2, c, L)$$

a	q_1	b
q_2	a	c

přenos beze změny

#	a	b
#	a	b

$$\delta(q_1, b) \ni (q_2, c, R)$$

a	q_1	b
a	c	q_2

$$\delta(_, _) \ni (q_2, _, L)$$

a	b	c
a	b	q_2

$$\delta(q_1, b) \ni (q_2, c, R)$$

d	a	q_1
d	a	c

$$\delta(_, _) \ni (_, c, L)$$

a	b	d
c	b	d

- ▶ Přenos beze změny všude, kde není v okolí stav (hlava)
- ▶ přenos beze změny pokud stav je přijímající
- ▶ na každém řádku nejvýše jeden stav
- ▶ okénko musí být částí povoleného přechodu
- ▶ rozbor technický, udělali za nás jiní.
- ▶ **Tvrzení:** Pokud je první řádek tabulky počáteční konfigurace a každé okénko je legální, pak každý řádek odpovídá legální konfiguraci dosažitelné jedním krokem z předchozího řádku.
 - ★ V horním řádku není stav, pak se prostřední symbol musí opsat beze změny.
 - ★ V horním řádku stav vprostřed: okénko garantuje korektnost přepisu obou stran.



Proof.

- Z tabulky vytvoříme formuli $\phi = \phi_{cell} \ \& \ \phi_{start} \ \& \ \phi_{move} \ \& \ \phi_{accept}$.
- pro každé políčko tabulky (i, j) a písmeno $a \in \Gamma \cup Q \cup \{\#\}$ vytvoříme výrokovou proměnnou $x_{i,j,a}$

$$\phi_{cell} = \bigwedge_{1 \leq i, j \leq n^k} \left[\left(\bigvee_{a \in \Gamma \cup Q \cup \{\#\}} x_{i,j,a} \right) \ \& \ \bigwedge_{s \neq t \in \Gamma \cup Q \cup \{\#\}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right] \quad \# \text{právě jed}$$

$$\phi_{start} = x_{1,1,\#} \ \& \ x_{1,2,q_0} \ \& \ x_{1,3,w_1} \ \& \ x_{1,4,w_2} \ \& \ \dots \ \& \ x_{1,n+2,w_n} \ \& \ x_{1,n+3,-} \ \& \ \dots \ \& \ x_{1,n^k,\#}$$

$$\phi_{accept} = \bigvee_{1 \leq i, j \leq n^k} x_{i,j,q_{accept}}$$

- Celková formule ϕ_{move} bude konjunkce, že každé okénko s horním středem na pozici i, j je legální

$$\phi_{move} = \bigwedge_{1 \leq i < n^k, 1 < j < n^k} \phi_{i,j} \quad \# \text{ okénko } (i, j) \text{ je legální}$$

- legálnost okénka (i, j) zajistíme disjunkcí legálních okének

$$\phi_{i,j} = \bigvee_{(a_1, \dots, a_6) \in LEGAL} (x_{i,j-1,a_1} \ \& \ x_{i,j,a_2} \ \& \ x_{i,j+1,a_3} \ \& \ x_{i+1,j-1,a_4} \ \& \ x_{i+1,j,a_5} \ \& \ x_{i+1,j+1,a_6})$$

Proof.

- **Tvrzení:** Převod má polynomiální složitost, konkrétně $O(n^{2k} \log n)$.
 - ▶ $\phi_{cell} \in O(n^{2k} \log n)$, procházíme dvojice indexů do n krát konstanta $|\Gamma|$ indexující buňku
 - ▶ $\phi_{start} \in O(n^k \log n)$, procházíme první řádek
 - ▶ $\phi_{move}, \phi_{accept} \in O(n^{2k} \log n)$,
 - ★ počet buněk n^{2k} , pro každou konstantní velikost formule.
- pro štouraly $\log n$ pro zápis indexu proměnných, jehož délka závisí na n .



3SAT je NP-úplný.

Theorem 12.31

3SAT je NP-úplný.

Proof.

- Upravíme předchozí převod na formuli 3SAT.
- V CNF je skoro vše, kromě disjunkce okének pohybu, která jsou konjunkcí,
 - ▶ převedeme do CNF
 - ▶ stačí polynomiální (konstantní) čas - velikost podformule závisí jen na stroji N , nikoli na délce vstupu
- v krátkých klauzulích zkopírujeme proměnnou
- dlouhé rozdělíme zavedením nových proměnných.



- Definice časové složitosti
- asymptotické složitosti $O()$
- třída P polynomiálně složitých problémů
- třída NP zavedená přes verifikátory
- třída NP odpovídá jazykům rozhodovaným turingovými stroji v polynomiálním čase
 - ▶ všechny větve výpočtu zastaví v polynomiálním čase.
- polynomiální převoditelnost
- NP-úplnost
- SAT je NP-úplný.