IB107 Vyčíslitelnost a složitost

polynomiální redukce, SAT, 3SAT, CLIQUE

Jan Strejček

Fakulta informatiky
Masarykova univerzita

polynomiální redukce

Definice (polynomiální redukce)

Nechť $A \subseteq \Sigma^*$ a $B \subseteq \Phi^*$ jsou jazyky. Řekneme, že A se polynomiálně redukuje na B, píšeme $A \leq_p B$, právě když existuje totální funkce $f : \Sigma^* \to \Phi^*$, která je vyčíslitelná deterministickým Turingovým strojem pracujícím v polynomiálním čase a taková, že

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

Funkci f nazveme polynomiální redukcí A na B.

polynomiální redukce

Definice (polynomiální redukce)

Nechť $A \subseteq \Sigma^*$ a $B \subseteq \Phi^*$ jsou jazyky. Řekneme, že A se polynomiálně redukuje na B, píšeme $A \leq_p B$, právě když existuje totální funkce $f : \Sigma^* \to \Phi^*$, která je vyčíslitelná deterministickým Turingovým strojem pracujícím v polynomiálním čase a taková, že

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

Funkci f nazveme polynomiální redukcí A na B.

Platí $A \leq_p B \implies \overline{A} \leq_p \overline{B}$.

polynomiální redukce

Definice (polynomiální redukce)

Nechť $A \subseteq \Sigma^*$ a $B \subseteq \Phi^*$ jsou jazyky. Řekneme, že A se polynomiálně redukuje na B, píšeme $A \leq_p B$, právě když existuje totální funkce $f : \Sigma^* \to \Phi^*$, která je vyčíslitelná deterministickým Turingovým strojem pracujícím v polynomiálním čase a taková, že

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

Funkci f nazveme polynomiální redukcí A na B.

Platí $A \leq_p B \implies \overline{A} \leq_p \overline{B}$.
Platí $A \leq_p B$ a $B \leq_p C \implies A \leq_p C$ (tj. $\leq_p$ je tranzitivní).

polynomiální redukce a složitostní třídy

Věta

Nechť $A \leq_p B$.

- $B \in P \implies A \in P$
- $B \in NP \implies A \in NP$

polynomiální redukce a složitostní třídy

Věta

Nechť $A \leq_p B$.

- $B \in P \implies A \in P$
- $B \in NP \implies A \in NP$

Důkaz: Nechť f je redukce A na B v polynomiálním čase a $\mathcal{M}_B$ je TM rozhodující B. Stroj $\mathcal{M}_A$ rozhodující A na vstupu w

1. spočítá $f(w)$
2. spustí $\mathcal{M}_B$ na vstupu $f(w)$ a vrátí stejný výsledek jako $\mathcal{M}_B$

Je-li $\mathcal{M}_B$ deterministický, pak je i $\mathcal{M}_A$ deterministický. Krok 1 lze provést v polynomiálním čase vzhledem k $|w|$, krok 2 v polynomiálním čase vzhledem k $|f(w)|$, což je polynomiální i vzhledem k $|w|$. $\mathcal{M}_A$ tedy pracuje v polynomiálním čase. ■

polynomiální redukce a složitostní třídy

Definice (těžká a úplná množina ve složitostní třídě)

Nechť $\mathbb{C}$ je složitostní třída splňující NP $\subseteq \mathbb{C}$. Jazyk L nazveme $\mathbb{C}$-těžký, právě když pro každý jazyk $L' \in \mathbb{C}$ platí $L' \leq_p L$. Je-li navíc $L \in \mathbb{C}$, pak L nazýváme $\mathbb{C}$-úplný nebo úplný ve třídě $\mathbb{C}$.

SAT je NP-úplný

Věta (Cookova-Levinova věta)

SAT $= \{\langle\varphi\rangle \mid \varphi$ *je splnitelná výroková formule*$\}$ *je NP-úplný.*

SAT je NP-úplný

Věta (Cookova-Levinova věta)

$SAT = \{\langle\varphi\rangle \mid \varphi$ je splnitelná výroková formule$\}$ je NP-úplný.

Důkaz: $SAT \in$ NP jsme již dokázali.
Ukážeme, že SAT je NP-těžký, tj. $A \in \text{NP} \implies A \leq_p SAT$.

SAT je NP-úplný

Věta (Cookova-Levinova věta)

$SAT = \{\langle\varphi\rangle \mid \varphi$ je splnitelná výroková formule$\}$ je NP-úplný.

Důkaz: $SAT \in$ NP jsme již dokázali.
Ukážeme, že SAT je NP-těžký, tj. $A \in \text{NP} \implies A \leq_p SAT$.

Nechť $\mathcal{M}$ je nedeterministický TM rozhodující A v čase $p(n)$, kde p je polynom. Pro každé slovo w sestrojíme výrokovou formuli Φ, která je splnitelná, právě když stroj $\mathcal{M}$ má akceptující výpočet na w.

SAT je NP-úplný

Každý výpočet stroje $\mathcal{M}$ pracujícího v čase $p(n)$ na slově w délky n lze reprezentovat tabulkou:

Vytvoříme Φ, aby platilo:

Φ je splnitelné $\iff$ existuje tabulka reprezentující akceptující výpočet na w

SAT je NP-úplný

$$\Phi \;=\; \Phi_{\text{cell}} \;\wedge\; \Phi_{\text{start}} \;\wedge\; \Phi_{\text{move}} \;\wedge\; \Phi_{\text{accept}}$$

$\Phi_{\text{cell}} =$ “každé $x_{i,j,s}$ platí $\iff$ v tabulce na pozici i, j je symbol s, kde $s \in C = Q \cup \Gamma \cup \{\#\}$”

$$\Phi_{\text{cell}} = \bigwedge_{\substack{1 \le i \le p(n)+1 \\ 1 \le j \le p(n)+3}} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \;\wedge\; \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} \neg (x_{i,j,s} \wedge x_{i,j,t}) \right) \right]$$

$|\Phi_{\text{cell}}| \in \mathcal{O}(p^2(n))$

SAT je NP-úplný

$$\Phi \;=\; \Phi_{\text{cell}} \;\wedge\; \Phi_{\text{start}} \;\wedge\; \Phi_{\text{move}} \;\wedge\; \Phi_{\text{accept}}$$

$\Phi_{\text{start}} =$ “na prvním řádku je iniciální konfigurace pro $w = w_1 \ldots w_n$”

$$\begin{aligned}\Phi_{\text{start}} \;=\;& x_{1,1,\#} \;\wedge\; x_{1,2,q_0} \;\wedge\; x_{1,3,\triangleright} \;\wedge \\ & x_{1,4,w_1} \;\wedge\; x_{1,5,w_2} \;\wedge\; \ldots \;\wedge\; x_{1,n+3,w_n} \;\wedge \\ & x_{1,n+4,\sqcup} \;\wedge\; x_{1,n+5,\sqcup} \;\wedge\; \ldots \;\wedge\; x_{1,p(n)+2,\sqcup} \;\wedge\; x_{1,p(n)+3,\#}\end{aligned}$$

$|\Phi_{\text{start}}| \in \mathcal{O}(p(n))$

SAT je NP-úplný

$$\Phi \;=\; \Phi_{\text{cell}} \;\wedge\; \Phi_{\text{start}} \;\wedge\; \Phi_{\text{move}} \;\wedge\; \Phi_{\text{accept}}$$

$\Phi_{\text{move}} =$ “každé dva po sobě jdoucí řádky odpovídají kroku výpočtu”

popíšeme pomocí “legálních oken” 2×3

příklady legálních oken pro $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$:

SAT je NP-úplný

$$\Phi \;=\; \Phi_{\text{cell}} \;\wedge\; \Phi_{\text{start}} \;\wedge\; \Phi_{\text{move}} \;\wedge\; \Phi_{\text{accept}}$$

$\Phi_{\text{move}} =$ “každé okno 2×3 v tabulce je legální” (okna se překrývají)

$$\Phi_{\text{move}} = \bigwedge_{\substack{1 \le i < p(n)+1 \\ 1 < j < p(n)+3}} \Bigg(\bigvee_{\text{legální okno}} \begin{array}{l} x_{i,j-1,a_1} \;\wedge\; x_{i,j,a_2} \;\wedge\; x_{i,j+1,a_3} \;\wedge \\ x_{i+1,j-1,a_4} \;\wedge\; x_{i+1,j,a_5} \;\wedge\; x_{i+1,j+1,a_6} \end{array} \Bigg)$$

a_1	a_2	a_3
a_4	a_5	a_6

$|\Phi_{\text{move}}| \in \mathcal{O}(p^2(n))$

SAT je NP-úplný

$$\Phi = \Phi_{\text{cell}} \wedge \Phi_{\text{start}} \wedge \Phi_{\text{move}} \wedge \Phi_{\text{accept}}$$

$\Phi_{\text{accept}} =$ “v tabulce je stav q_{acc}”

$$\Phi_{\text{accept}} = \bigvee_{\substack{1 < i \leq p(n)+1 \\ 1 < j < p(n)+3}} x_{i,j,q_{acc}}$$

$|\Phi_{\text{accept}}| \in \mathcal{O}(p^2(n))$

SAT je NP-úplný

$$\Phi \;=\; \Phi_{\text{cell}} \;\wedge\; \Phi_{\text{start}} \;\wedge\; \Phi_{\text{move}} \;\wedge\; \Phi_{\text{accept}}$$

$$|\Phi| = \mathcal{O}(p^2(n)) + \mathcal{O}(p(n)) + \mathcal{O}(p^2(n)) + \mathcal{O}(p^2(n)) = \mathcal{O}(p^2(n))$$

Počet proměnných $x_{i,j,s}$ závisí je $\mathcal{O}(p^2(n))$, tedy závisí na $n = |w|$.
Proměnnou lze zakódovat do binární abecedy $\mathcal{O}(\log n)$ znaky.
Tedy $|\langle\Phi\rangle| = \mathcal{O}(p^2(n) \cdot \log n)$, což znamená $|\langle\Phi\rangle| = \mathcal{O}(p^2(n) \cdot n)$.

$\langle\Phi\rangle$ má polynomiální délku vzhledem k $n = |w|$ a lze spočítat v polynomiálním čase. Tedy $A \leq_p SAT$. ■