Týden 13

Přednáška

První část (přednášel ing. Martin Kot)

- Zopakování pojmů (polynomiální) převeditelnost mezi problémy, nedeterministický algoritmus, třída NPTIME, NP-obtížnost a NP-úplnost
- Postupně probrány problémy SAT, 3-SAT, HC, HK, 3-CG a IS, u každého jsme si ukázali příklady instancí s odpovědí ANO i NE a u většiny si uvědomili, jak by vypadal polynomiální nedeterministický algoritmus řešící daný problém
- Převod SAT (formule v KNF) na 3-SAT
- Postupně jsme došli k algoritmu převodu HC na HK přes algoritmy, které pracovaly správně jen pro některé instance (s ukázkou toho, kdy selhávají)
- Jeden z možných převodů 3-SAT na 3-CG (jako příklad převodu mezi problémy z odlišných oblastí - logika a grafy)
- Převod 3-CG na IS

Druhá část (přednášel ing. Zdeněk Sawa)

- Důkaz NP-úplnosti problému SAT (Cookova věta) (s využitím vytvořené animace)
- Převod SAT na 3-SAT (v případě, kdy instancí SAT může být libovolná formule, tj. ne nutně v KNF) (s využitím vytvořené animace)

Sekce pro hlubší zájemce – důkazy (přednášel ing. Zdeněk Sawa)

- obtížnost problému ekvivalence (resp. neekvivalence) booleovských formulí
- speciální případy problému SAT řešitelné v polynomiálním čase:

    - 2-SAT - popsán polynomiální algoritmus
    - NHORNSAT - jen zmínka

Partie textu k prostudování

Celá kapitola 8 a příslušné animace.

Cvičení

Prezentace referátů

Referát č. 23 (Polynomiální převeditelnost)
Jedna z animací k předmětu ukazuje polynomiální převeditelnost problému nezávislé množiny na problém hamiltonovského cyklu.
Je to technicky netriviální konstrukce (která vyšla z Referátu 6 na `http://www.cs.vsb.cz/jancar/VYCSLOZ/vycsloz.htm`).
Prostudujte ji a prezentujte na co nejjednodušším konkrétním případu, který ještě umožňuje rozumnou demonstraci hlavní myšlenky.
Referát č. 24 (NP-úplnost)
Cookova věta říká, že SAT je NP-úplný problém. Zaměřme se na důkaz toho, že SAT je NP-těžký, tedy že pro každý problém $P \in \mathsf{NP}$ platí $P \lhd \mathrm{SAT}$ (P je polynomiálně převeditelný na SAT).
Uvažujme tedy libovolný, ale dále pevně daný, problém $P \in \mathsf{NP}$. Tou libovolností se rozumí to, že o P nemůžeme předpokládat nic jiného než $P \in \mathsf{NP}$ (nikoli to, že bychom se snad mohli zaměřit jen na jeden konkrétní problém, který bychom si zvolili podle vlastní libosti). O P tedy víme, že je rozhodován nějakým nedeterministickým Turingovým strojem M s časovou složitostí $T_M(n) \leq p(n)$ pro nějaký polynom p. (O M a p zase nic bližšího nevíme, ale jsou už teď pro nás pevně dané.) Máme ukázat, že existuje polynomiální algoritmus, který k libovolnému vstupu w stroje M zkonstruuje booleovskou formuli $\mathcal{F}_w$ (v konjunktivní normální formě), která je splnitelná právě tehdy, když pro slovo w existuje přijímající výpočet stroje M.
Formule $\mathcal{F}_w$ má být tedy splnitelná právě tehdy, když existuje posloupnost konfigurací $C_0, C_1, C_2, \ldots, C_m$ stroje M, kde $m \leq p(n)$, C_0 je počáteční konfigurace odpovídající vstupu w, C_m je přijímající konfigurace a pro každé $i = 0, 1, \ldots, m-1$ platí $C_i \vdash_M C_{i+1}$ (tedy z C_i může stroj M jedním krokem přejít do C_{i+1}).
Obecný návod k sestrojení formule $\mathcal{F}_w$ zachycující schéma takového (potenciálního) výpočtu lze nalézt např. v podkladu k referátu č. 7 na
`http://www.cs.vsb.cz/jancar/VYCSLOZ/vycsloz.htm`.)
Podívejte se také na příslušnou animaci v seznamu animací odkazovaném na web-stránce předmětu.
Uvažujme teď tento konkrétní případ. Stroj M je dán následujícím seznamem instrukcí (q_1 je počáteční stav, q_{acc} je přijímající stav, q_{rej} je zamítající stav, vstupní abeceda je $\{a, b\}$, pracovní abeceda je $\{a, b, \square\}$):

$$(q_1, 0) \to (q_1, 0, +1),\ \ (q_1, 0) \to (q_2, 0, +1) \text{ (nedeterminismus)}$$
$$(q_1, 1) \to (q_1, 1, +1),\ \ (q_1, 1) \to (q_2, 1, +1) \text{ (nedeterminismus)}$$
$$(q_1, \square) \to (q_{rej}, \square, 0)$$

$$(q_2, 0) \to (q_{rej}, 0, 0)$$
$$(q_2, 1) \to (q_{acc}, 1, 0)$$

$(q_2, \Box) \rightarrow (q_{rej}, \Box, 0)$

Je zřejmé, že časová složitost stroje M je $T_M(n) = n + 1$.
Předveďte a vysvětlete obecnou konstrukci formule $\mathcal{F}_w$ tak, že ji demonstrujete pro uvedený konkrétní M a vstupní slovo w délky 2, např. $w = 10$.
Referát č. 25 (Savitchova věta)
Popište konstrukci v důkazu Savitchovy věty. (K nastudování můžete např. využít podklad k referátu č. 8 na `http://www.cs.vsb.cz/jancar/VYCSLOZ/vycsloz.htm`.)
Můžete se ovšem omezit na tento speciální případ:

> Je-li problém P rozhodován nedeterministickým Turingovým strojem s prostorovou složitostí n, pak je také rozhodován deterministickým Turingovým strojem s polynomiální prostorovou složitostí.

Máte tedy vysvětlit, jak lze k tzv. lineárně omezenému automatu (linear bounded automaton) M,

> tj. k nedeterministickému Turingovu stroji M, který při výpočtu na vstupním w, $|w| = n$, nenavštíví jiná políčka než ta, na nichž je zapsán vstup (a má tedy prostorovou složitost n)

navrhnout (deterministický) algoritmus A, který pro zadané w zjistí, zda M má přijímající výpočet pro w (tedy zda $w \in L(M)$). Algoritmu A přitom musí stačit polynomiálně omezená paměť.
(Připomenutí. Počet konfigurací délky n stroje M je omezen hodnotou c^n, kde konstantu c lze snadno spočítat z velikosti (stavové množiny a abecedy) stroje M. Délka nejkratšího přijímajícího výpočtu M nad w, $|w| = n$, (pokud takový existuje) je tedy také omezena oním c^n.)
(Bylo by dobré ukázat, že ta prostorová složitost A se dá omezit kvadraticky, je v $O(n^2)$, a naznačit, proč A lze přímočaře implementovat deterministickým Turingovým strojem s prostorovou složitostí $O(n^2)$.)

Příklady

Příklad 13.1

Jako rozcvičku vyřešte nejprve tato cvičení z učebního textu:
Seřaďte následující tři funkce podle rychlosti jejich růstu.
a/ $n/2005$, $\sqrt{n} \cdot 3n$, $n + n \cdot \log n$
b/ $(\log n)^n$, n^n, $2^{\sqrt{n}}$

Příklad 13.2

Uvažujme následující problém (jeden z často uváděných NP-úplných problémů).

NÁZEV: TSP (*problém obchodního cestujícího (*ANO/NE *verze)*)

VSTUP: množina „měst“ $\{1, 2, \ldots, n\}$, přir. čísla („vzdálenosti“) d_{ij} ($i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$); dále číslo ℓ („limit“).

OTÁZKA: existuje „okružní jízda“ dlouhá nejvýše ℓ, tj. existuje permutace $\{i_1, i_2, \ldots, i_n\}$ množiny $\{1, 2, \ldots, n\}$ tž. $d(i_1, i_2) + d(i_2, i_3) + \cdots + d(i_{n-1}, i_n) + d(i_n, i_1) \leq \ell$?

Je to rozhodovací (neboli ANO/NE) verze optimalizačního problému. Odvoďte nejdříve, jak vypadá onen optimalizační problém (tedy co je jeho vstupem a co odpovídajícím výstupem).

Dále ukažte nějakou malou (ale ne úplně triviální) instanci (tedy vstup) uvedeného problému TSP, pro niž je odpověď ANO, a instanci, pro niž je odpověď NE.

Pak prokažte (návrhem konkrétního nedeterministického algoritmu), že TSP je v NP.

Nakonec zkuste vymyslet důkaz NP-obtížnosti problému TSP.
(Nápověda. Můžete využít faktu, že problém hamiltonovské kružnice (HK) je NP-úplný.)

Připomenutí bokem: animace ukazují důkaz Cookovy věty, tedy důkaz toho, že SAT je NP-úplný, a dále demonstrují SAT $\lhd$ 3-SAT, 3-SAT $\lhd$ IS, IS $\lhd$ HC, HC $\lhd$ HK (a také nyní požadovaný převod HK $\lhd$ TSP.)

Příklad 13.3

Uvažujme problém

NÁZEV: ILP (*problém celočíselného lineárního programování*)

VSTUP: Matice A typu $m \times n$ a sloupcový vektor b velikosti m, jejichž prvky jsou celá čísla.

OTÁZKA: Existuje celočíselný sloupcový vektor x (velikosti n) tž. $Ax \geq b$?

Ukažte nejprve nějakou malou (ale ne úplně triviální) instanci (tedy vstup) uvedeného problému ILP, pro niž je odpověď ANO, a instanci, pro niž je odpověď NE.

Vysvětlete přesně, co bychom museli udělat, kdybychom chtěli ukázat, že 3-SAT $\lhd$ ILP.

Zbude-li čas, zkuste tuto převeditelnost dokázat. Přinejmenším ale uveďte, co bychom mohli říci o složitosti problému ILP poté, co bychom prokázali 3-SAT $\lhd$ ILP.

Dále pouvažujte o tom, zda ILP patří do NP.

Je to tak, ale je to příklad problému, jehož příslušnost k NP není ihned zřejmá – na rozdíl od dřívějších příkladů problémů v NP.

(Spokojíme se zde jen s odkazem na fakt, že se dá ukázat, že existuje-li řešení nerovnosti $Ax \geq b$, pak existuje i řešení „dostatečně malé“ – jeho zápis je polynomiální vzhledem k zápisu A a b; řešení se tedy dá v polynomiálním čase „uhodnout“ a ověřit.)