Týden 12

Přednáška-pondělí

Na začátku jsme se stručně vrátili k obecným metodám návrhu (rychlých) algoritmů.

Třída PTIME

Připomněli jsme si definici třídy PTIME. Přitom jsem zdůraznil, že všechny problémy ve studijním textu (nejen ty, které jsou v PTIME) je třeba důkladně promyslet – pak nemůže být pro nikoho problémem u zkoušky nějaký požadovaný problém přesně definovat a uvést příklady instancí s pozitivní odpovědí a instancí s negativní odpovědí.

Nedeterministické algoritmy a jejich složitost; třída NPTIME

Ujasnili jsme si pojem nedeterministického algoritmu (Turingova stroje) a definici toho, co to znamená, že nějaký nedeterministický Turingův stroj M rozhoduje daný problém P.

Také jsme přímočaře rozšířili pojem (časové) složitosti na nedeterministické stroje a definovali jsme třídu NPTIME.

Ukázali jsme si (nedeterministické) algoritmy prokazující, že problém SAT (splnitelnost booleovských formulí) a IS (Independent Set, rozhodovací verze problému nezávislé množiny v grafu) jsou v NPTIME.

Polynomiální převeditelnost; NP-úplné problémy

Již známý pojem převeditelnosti mezi problémy jsme využili v definici polynomiální převeditelnosti mezi problémy. (Příslušný převádějící algoritmus musí mít polynomiální časovou složitost, tedy časovou složitost omezenou polynomem.)

Všimli jsme si, jak může prokázaná polynomiální převeditelnost mezi problémy pomoci v určování (ne)příslušnosti k PTIME či NPTIME.

Definovali jsme pojem NP-úplného problému; problémy SAT, 3-SAT, HC, HK, 3-CG a IS jsou příklady NP-úplných problémů.

Postupně jsme došli k algoritmu, který prokazuje polynomiální převeditelnost HC na HK.

Přednáška-čtvrtek

Vrátili jsme se k nerozhodnutelnosti pravdivosti uzavřených formulí 1.řádu s predikáty $PLUS(x, y, z)$ (tj. $x + y = z$) a $MULT(x, y, z)$ (tj. $x \cdot y = z$) ve standardním modelu aritmetiky $(\mathbb{N}, +, \cdot)$.

Pak jsme diskutovali jeden pozoruhodný důkaz z oblasti teorie složitosti, který se dá vyjádřit sloganem

nedeterministický prostor je uzavřený na doplněk.

Ukázali jsme ovšem jen následující speciální Tvrzení:

Ke každému nedeterministickému Turingovu stroji M s prostorovou složitostí $O(n)$, který rozhoduje problém P, existuje (lze sestrojit) nedeterministický Turingův stroj M' rovněž s prostorovou složitostí $O(n)$, který rozhoduje problém $\overline{P}$ (tedy doplňkový problém problému P).

Jestliže tedy M má pro slovo w alespoň jeden přijímající výpočet, tak všechny výpočty stroje M' na w jsou nepřijímající; jestliže všechny výpočty stroje M na w jsou nepřijímající, pak existuje alespoň jeden výpočet M' na w, který je přijímající.

Toto tvrzení mj. znamená, že třída tzv. kontextových jazyků je uzvařena na doplněk. To byl od 60. let 20. století známý otevřený problém a mezi zainteresovanými převládal názor, že tato třída na doplněk uzavřena není. *Tvrzení dokázal jako první student informatiky* na MFF UK v Bratislavě R. Szelepcsenyi na jaře 1987. Než ovšem bylo řešení (dopracováno, zobecněno a) oznámeno odborné veřejnosti, přišel nezávisle s řešením známý vědec N. Immerman v USA. Proto se tomuto tvrzení (v obecnější podobě) dnes říká „The Immerman-Szelepcsenyi Theorem“.

Načrtli jsme si hlavní myšlenku:

Má-li M' uloženo v čítači c číslo c_i udávající počet konfigurací stroje M, do kterých se tento stroj může dostat v i krocích výpočtu na w, pak do čítače c' spočte c_{i+1} následovně:

Generuje systematicky všechny možné konfigurace $C_1, C_2, \ldots, C_m$ stroje M velikosti $S_M(|w|)$ (kde S_M je prostorová složitost stroje M); na začátku také vynuluje čítač c'.

Pro každou vygenerovanou C_j zjišťuje, zda C_j může být dosažena v $i+1$ krocích takto:

Generuje (v jiném kousku paměti) systematicky všechny možné konfigurace $D_1, D_2, \ldots, D_m$ stroje M velikosti $S_M(|w|)$; na začátku také vynuluje čítač d.

Pro každou D_ℓ nedeterministicky „hádá“, zda D_ℓ je dosažitelná v i krocích. Pokud si tipne, že ne, pokračuje vygenerováním $D_{\ell+1}$ … Pokud si tipne, že ano, odsimuluje nedeterministicky zvolených i kroků stroje M na w: když takto dosažená konfigurace není totožná s D_ℓ, stroj M' neúspěšně skončí; když takto dosažená konfigurace je totožná s D_ℓ (M' tedy ověřil, že D_ℓ je dosažitelná v i krocích), M' zvýší čítač d o 1 a ověří, zda z D_ℓ lze jedním krokem dosáhnout C_j: pokud ano, zvýší čítač c' a začne zkoumat C_{j+1}, pokud ne, pokračuje vygenerováním $D_{\ell+1}$ …

Pokud takto prošel všechny $D_1, D_2, \ldots, D_m$, aniž zjistil, že C_j je dosažitelná v $i+1$ krocích, tak ověří, zda hodnoty v čítačích c a d jsou stejné: když nejsou, M' neúspěšně končí, když jsou (tedy M' skutečně správně uhodl a ověřil všechny konfigurace D_ℓ, které jsou dosažitelné v i krocích, a takto ověřil, že C_j skutečně není dosažitelná v $i+1$ krocích), pokračuje M' zkoumáním C_{j+1} (aniž zvýšil c') …

Po (úspěšném) spočtení c_{i+1} (v čítači c'), zkopíruje M' hodnotu c' do čítače c, vynuluje c' a pustí se do výpočtu c_{i+2} … Toto provádí pro vš. $i \leq m$, kde m je počet všech možných konfigurací stroje M velikosti $S_M(|w|)$; kontroluje si tento hlavní cyklus speciálním čítačem, pro nějž mu zajisté stačí prostor $O(n)$. Pokud během práce M' někdy zjistí, že je dosažitelná nějaká přijímající konfigurace (stroje M při výpočtu na w), M' okamžitě skončí neúspěšně

(tedy nepřijme). Pokud se to nestalo a stroj M' prošel (bez neúspěšného ukončení z důvodů popsaných výše) výpočet $c_1, c_2, \ldots, c_m$, tak přijme.
K ověření korektnosti dodejme: Když M má přijímající výpočet pro w, tak má také přijímající výpočet, v němž se neopakují dosažené konfigurace, a tedy má výpočet délky $\leq m$.
Poznámka. Detailnější popis důkazu (obecnějšího tvrzení) lze najít např. i v Internetových zdrojích. (Google: Immerman-Szelepcsenyi Theorem.)

Partie textu k prostudování

Části 8.3., 8.4., 8.5. (třída PTIME, třída NPTIME, NP-úplné problémy).

Cvičení

Prezentace referátů

Referát č. 21 (Převeditelnost mezi problémy)
Demonstrujte myšlenku převeditelnosti IPKP (iniciálního Postova korespondenčního problému) na PKP (Postův korespondenční problém). (Můžete vyjít např. z příslušné animace, zvolte si ale jiný příklad, na němž myšlenku srozumitelně předvedete a vysvětlíte.)

Referát č. 22 (Časová složitost algoritmů, asymptotická notace)
Podejte matematický důkaz (využívající např. l'Hospitalova pravidla) toho, že je-li $f(n) \leq p(n)$ pro nějaký polynom p a $g(n) \geq c^n$ pro nějakou konstantu $c > 1$, tak platí $f \in o(g)$.
Podobně to ukažte pro případ, kde $f(n) \leq (\log n)^k$ pro nějakou konstantu k a $g(n) \geq n^c$ pro nějakou konstantu $c > 0$.

Příklady

Příklad 12.1

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 12.2

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.)

Příklad 12.3

Diskutujte problémy SAT, 3-SAT, HC, HK, 3-CG a IS z přednášky. (Příklady instancí s odpovědí ANO, s odpovědí NE, nedeterministické polynomiální algoritmy pro tyto problémy, ...)