Týden 13

Přednáška - první část

Poznámky k dokazatelně nezvládnutelným problémům

Jestliže prokážeme NP-obtížnost či PSPACE-obtížnost nějakého problému, říkáme, že je *prakticky nezvládnutelný* (intractable). Je totiž jasné, že navrhneme-li algoritmus, který řeší (přesně ten) daný problém, a neobjevíme-li přitom geniální „trik“, na který dosud nikdo nepřišel, bude mít algoritmus tzv. superpolynomiální (typicky exponenciální či ještě horší) složitost. Obecně používat algoritmus (resp. odpovídající program) budeme moci jen na velmi malá vstupní data. Teoreticky ovšem pořád ještě možnost rychlého algoritmu (založeného na „geniálním triku“) existuje.

Samozřejmě je možné, že exponenciální algoritmus ve skutečnosti chceme použít jen na malá data, anebo ho třeba používáme na data, při nichž se neprojeví ona (worst case) exponenciální složitost. V tomto smyslu praktická nezvládnutelnost (obecného) problému ještě neznamená, že ho v praxi nemůže počítačový program úspěšně řešit v pro nás zajímavých případech. (Existují i jiné možnosti, jak v praxi úspěšně řešit i „nezvládnutelný“ problém. Zmíníme se o tom v partiích o aproximačních a pravděpodobnostních algoritmech.)

Známe ovšem i tzv. *dokazatelně nezvládnutelné* problémy (provably intractable problems), tj. ty, u nichž máme *dokázáno*, že pro ně neexistují polynomiální algoritmy. Definujeme-li např. třídy

$$\mathsf{EXPTIME} = \bigcup_{k=0}^{\infty} \mathcal{T}(2^{n^k}) \,, \qquad \mathsf{EXPSPACE} = \bigcup_{k=0}^{\infty} \mathcal{S}(2^{n^k})$$

pak EXPTIME-těžký či EXPSPACE-těžký problém je takovým dokazatelně nezvládnutelným problémem.

Dá se totiž ukázat, že inkluze $\mathsf{PTIME} \subset \mathsf{EXPTIME}$, $\mathsf{PSPACE} \subset \mathsf{EXPSPACE}$ jsou skutečně vlastní (tzn., že neplatí rovnost). (My to zde ale dokazovat nebudeme.)

Např. následující problém je EXPSPACE-úplný:

NÁZEV: RE^2 (*ekvivalence regulárních výrazů s mocněním*)

VSTUP: dva regulární výrazy, v nichž je možné použít mocnění (tzn. je možno psát α^2 místo $\alpha \cdot \alpha$).

OTÁZKA: reprezentují zadané výrazy tentýž jazyk?

Připomeňme, že problém Eq-RegExp pro standardní regulární výrazy (bez mocnění) je PSPACE-úplný, stejně jako problém Eq-NFA jazykové ekvivalence *nedeterministických* konečných automatů. (Víme, že složitost je funkcí velikosti vstupu; mocnění v regulárních výrazech umožňuje exponenciálně zkrátit zápis některých dlouhých standardních výrazů.)

Existují samozřejmě i dokazatelně těžší (superexponenciální) problémy. Ilustrujme je příkladem problému Presburgerovy aritmetiky (rozhodování pravdivosti formulí teorie sčítání).

Rozhodnutelnost Presburgerovy aritmetiky (jen sčítání)

Nastínili jsme důkaz rozhodnutelnosti Presburgerovy aritmetiky (využívající jen predikát $PLUS(x, y, z)$), tedy důkaz věty 9.3. z kapitoly 9.

NÁZEV: ThAdd (*problém pravdivosti teorie sčítání*)

VSTUP: formule jazyka 1. řádu užívající jediný „nelogický“ symbol – ternární (tj. 3-ární) predikátový symbol $PLUS$ ($PLUS(x, y, z) \Leftrightarrow_{df} x + y = z$).

OTÁZKA: je daná formule pravdivá pro množinu $\mathbb{N} = \{0, 1, 2, \dots\}$, kde $PLUS(a, b, c)$ je interpretováno jako $a + b = c$?

Zde vůbec není zřejmé, že existuje algoritmus, který daný problém řeší. Presburger ukázal takový algoritmus ve 20. létech 20. století (jedním z cílů tzv. Hilbertova programu bylo ukázat podobný algoritmus i s připuštěním predikátu pro násobení – nemožnost řešení tohoto úkolu ukázal později Gödel). Mnohem později bylo ukázáno, že každý algoritmus, řešící problém ThAdd má složitost minimálně 2^{2^n}.

Presburger ukázal algoritmus využitím tzv. metody eliminace kvantifikátorů. Elegantní důkaz umožňují také výsledky, které známe z teorie konečných automatů.

Existuje algoritmus rozhodující problém ThAdd.

Představme si třístopou pásku, kde v každé stopě je řetězec nul a jedniček, tj. binární zápis čísla. Na pásku samozřejmě můžeme hledět jako na jednostopou s tím, že povolené *symboly abecedy* jsou uspořádané *trojice* nul a jedniček. Snadno nahlédneme, že existuje konečný automat, který přijímá právě ta slova v „abecedě trojic“, která mají tu vlastnost, že součet čísla v první stopě s číslem v druhé stopě je roven číslu ve třetí stopě. Ihned je to jasné při čtení odzadu; pak si stačí připomenout, že regulární jazyky jsou uzavřeny na zrcadlový obraz.

Z dalších uzávěrových vlastností snadno vyvodíme, že pro libovolnou formuli $\mathcal{F}(x_1, x_2, \dots, x_n)$ jazyka ThAdd která *neobsahuje kvantifikátory*, lze zkonstruovat kon. automat $A_{\mathcal{F}}$ přijímající právě binární zápisy těch n-tic čísel (na n-stopé pásce), pro které je $\mathcal{F}(x_1, x_2, \dots, x_n)$ pravdivá.

Pro formuli $(\exists x_n)\mathcal{F}(x_1, x_2, \dots, x_n)$ je možné zkonstruovat automat, který přijímá právě binární zápisy těch $(n-1)$-tic čísel (dosazených za $x_1, x_2, \dots, x_{n-1}$), pro které je $(\exists x_n)\mathcal{F}(x_1, x_2, \dots, x_n)$ pravdivá: automat pracuje na $(n-1)$-stopé pásce, ale simuluje $A_{\mathcal{F}}$ tak, že obsah n-té stopy nedeterministicky hádá! (Pak ho samozřejmě lze převést na ekvivalentní deterministický automat.)

Jelikož $(\forall x_n)\mathcal{F}(x_1, x_2, \dots, x_n)$ je ekvivalentní $\neg(\exists x_n)\neg\mathcal{F}(x_1, x_2, \dots, x_n)$, načrtli jsme takto postup, který k formuli jazyka ThAdd v prenexní formě postupnou aplikací zmíněných kon-

strukcí sestrojí konečný automat, který přijme prázdné slovo (neboli nějakou „posloupnost 0-tic“) právě tehdy, když výchozí formule je pravdivá.

Nerozhodnutelnost aritmetiky (se sčítáním a násobením)

Uvedli jsme si jen hlavní myšlenku 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 $(\mathbb{N}, +, \cdot)$.
(Využili jsme nerozhodnutelnosti existence akceptujícího výpočtu zadaného Turingova stroje M na zadaném slově w.)

Aproximační algoritmy

Přiblížili jsme si elementární základy zachycené v sekci 10.3.

Pravděpodobnostní algoritmy

Přiblížili jsme si elementární základy zachycené v sekci 10.4.
Speciálně jsme se věnovali problému prvočíselnosti.
Uvědomili jsme si, že algoritmus

> Máš-li testovat prvočíselnost zadaného (např. několikasetmístného) k, projdi všechna $a, 1 < a < k$ a zjišťuj, zda $Divides(a, k)$ (tedy zda $(k \mod a) = 0$) ...

je exponenciální (ve velikosti zápisu k). (A to i při přímočarých vylepšeních, při nichž zkoumáme jen lichá $a \leq \sqrt{k}$ apod.)
Také jsme si všimli, že pravděpodobnostní algoritmus

> Vygeneruj náhodné a (řekněme liché $a, 1 < a \leq \sqrt{k}$); jestliže $Divides(a, k)$, return NE, jinak return ANO.

nám moc nepomůže. Vydá-li (nějaký) jeho běh NE, tak sice víme jistě, že k není prvočíslo, ale vydá-li ANO pro dané k třeba při miliónkrát opakovaném provedení, nemůžeme si vůbec být jisti, že k je prvočíslem. (Např. pro velké číslo $m = pq$, kde p, q jsou prvočísla, je náhodná trefa jednoho z dělitelů p, q téměř nemožná.)
Pak jsme naznačili, že (malá) Fermatova věta, je základem podstatně lepšího algoritmu (k čemuž se ještě vrátíme na cvičení).

Přednáška - druhá část

Diskutovali jsme mj. jeden pozoruhodný důkaz z oblasti teorie složitosti (Immerman-Szelepcsenyi Theorem), 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).

Partie textu k prostudování

Kapitola 9. Část 10.3. (aproximační algoritmy) a 10.4. (pravděpodobnostní algoritmy).

Cvičení

Prezentace referátů 23 a 24

Příklad 13.1

Zformulujte (přirozený) hltavý aproximační algoritmus pro problém TSP (jdi do nejbližšího dosud nenavštíveného města ...). Proveďte odhad jeho složitosti. Ukažte instanci, v níž algoritmus vydá cestu, která je více než dvakrát delší než optimální cesta. (Vaše instance nemusí splňovat trojúhelníkovou nerovnost.)

Příklad 13.2

Následující tvrzení je známo jako „Malá Fermatova věta“.
Tvrzení. Jestliže p je prvočíslo, tak pro každé $a, 0 < a < p$, platí

$$a^{p-1} \equiv 1 \pmod{p}$$

.
(Když p není prvočíslo, tak to neplatí, jak byste se měli být schopni sami snadno přesvědčit [např. zbude-li čas na cvičení]).
Přesvěčdte se, že tvrzení platí pro $p = 11$. Přitom si uvědomte, jak je užitečné tzv. opakované umocňování. Můžete postupovat vyplněním následující tabulky; přitom využijte, že $x^{10} = x^8 \cdot x^2$, tedy $x^{10} \mod 11 = (x^8 \mod 11) \cdot (x^2 \mod 11) \mod 11$.

x	$x^2 \mod 11$	$x^4 \mod 11$	$x^8 \mod 11$	$x^{10} \mod 11$
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Pak vyplňte podobnou tabulku pro neprvočíslo 15.

x	$x^2 \mod 15$	$x^4 \mod 15$	$x^8 \mod 15$	$x^{14} \mod 15$
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

Uvedená pozorování nabízejí zvážit jistý (polynomiální) pravděpodobnostní algoritmus k testování prvočíselnosti (velkých čísel). Jak vypadá tento algoritmus?

(Poznámka. Ten algoritmus „téměř“ funguje, „ošálí“ jej ale tzv. Carmichaelova čísla; naprosto korektní pravděpodobnostní algoritmus využívá o něco hlubší poznatky z teorie čísel.)

Příklad 13.3

Diskutujte ukázkovou zkouškovou písemku. Speciálně se zaměřte na druhou část, v níž rovněž musíte získat předepsané minimum bodů. (Turingovy stroje, RAMy, polynomiální převeditelnost mezi problémy, konkrétní pozitivní a negativní instance rozhodovacích problémů, aplikace Riceovy věty, základní třídy složitosti a jejich úplné problémy, …)