Týden 13

Přednáška-pondělí

Na začátku jsme se ještě vrátili k NP-úplnosti ...

PSPACE, NPSPACE, PSPACE-úplnost

Uvědomili jsme si nejprve, že např. pro zjištění toho, zda Bílý má nějakou strategii ve hře ŠACHY, která mu zaručuje vítězství v 200 tazích (rozumí se, že Bílý táhne maximálně 200-krát), bychom uměli celkem přímočaře sestavit algoritmus; např. zavoláme MaBilyVS(VychoziPozice, Bílý, 200), kde

> MaBilyVS(Pozice, NaTahu, Limit):
>
> if ((Pozice, NaTahu) představuje mat Černému) return ANO;
> if ((Pozice, NaTahu) představuje pat nebo mat Bílému nebo Limit=0) return NE;
> if (NaTahu=Bílý) {Postupně pro každý tah Bílého v Pozice zavolej MaBilyVS(Pozice', Černý, Limit), kde Pozice' vznikne z Pozice provedením příslušného tahu; když je v nějakém případě vráceno ANO, tak return ANO, jinak return NE};
> if (NaTahu=Černý) {Postupně pro každý tah Černého zavolej MaBilyVS(Pozice', Bílý, Limit-1); když je ve všech případech vráceno ANO, tak return ANO, jinak return NE}.

Snadno si ovšem spočteme, že odpovědi bychom se od tohoto algoritmu nedočkali, ale není to tím, že by přetekla paměť. Je snadno vidět, že pro přirozenou implementaci v zásadě stačí paměť velikosti 400 pozic (400 „šachovnic“). (Ano, jedná se o jistý průchod stromem hloubky ≤ 400; přitom není třeba konstruovat v paměti celý strom, ale stačí vždy udržovat aktuální větev.)
Tím jsme si připomněli, že i v malém prostoru (malé paměti) se pochopitelně dají provádět časově náročné výpočty.

Nadefinovali jsme třídy PSPACE, NPSPACE a připomněli jsme si Savitchovu větu z referátu na cvičení (a z učebního textu), která mj. implikuje PSPACE = NPSPACE.
Znázornili jsme si obrázkem inkluze

$$\mathsf{PTIME} \subseteq \mathsf{NPTIME} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE}.$$

Má se obecně zato, že obě inkluze jsou vlastní, byť nikdo nevyvrátil možnost PTIME = PSPACE.
Připomněli jsme si, co jsou NP-úplné problémy a nadefinovali jsme PSPACE-*úplné problémy*. Jako příklady PSPACE-úplných problému jsme uvedli QBF (problém pravdivosti

kvantifikovaných booleovských formulí), Eq-NFA (ekvivalence nedeterministických konečných automatů) a Eq-RegExp (ekvivalence regulárních výrazů). (Žádnému posluchači samozřejmě nedělá nejmenší problém uvést přesné definice problémů a příklady pozitivních a negativních instancí, že ano.)
Uvedli jsme také, že nejrůznější deskové a grafové hry se dají zformalizovat jako PSPACE-těžké (případně PSPACE-úplné) problémy. Např. u šachů by to ovšem chtělo definovat např. $(n \times n)$-šachy (pro všechna n, nejen $n = 8$). (Připomeňme, že podle našich definic patří každý problém s konečně mnoha instancemi do třídy $\mathcal{T}(1)$, tedy má konstantní složitost!)
Všimli jsme si, že problém QBF lze definovat jako zjišťování existence vítězné strategie ve hře dvou hráčů, kde Eva („existenční hráč“) nasazuje existenčně vázané proměnné a Adam („univerzální hráč“) nasazuje univerzálně vázané proměnné.

Dokazatelně nezvládnutelné problémy

Uvedli jsme alespoň krátce problémy z kapitoly 9.

Aproximační algoritmy

Přiblížili jsme si elementární základy zachycené v sekci 10.3.

Přednáška-čtvrtek

Podívali jsme se na některé aproximační algoritmy.

Partie textu k prostudování

Kapitola 9 (speciálně Třída PSPACE). Sekce 10.3. (Aproximační algoritmy).

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 $P \triangleleft \mathrm{SAT}$ (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 $\{a, b, \Box\}$):

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

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

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. $w = 10$.

Příklady

Příklad 13.1

Připomeňme si (PSPACE-úplný) problém

NÁZEV: QBF (*problém pravdivosti kvantifikovaných booleovských formulí*)

VSTUP: formule $(\exists x_1)(\forall x_2)(\exists x_3)(\forall x_4)\dots(\exists x_{2n-1})(\forall x_{2n})\mathcal{F}(x_1, x_2, \dots, x_{2n})$, kde $\mathcal{F}(x_1, x_2, \dots, x_{2n})$ je booleovská formule v konjunktivní normální formě.

OTÁZKA: je daná formule pravdivá?

Uveďte nějaké malé, ale netriviální, příklady pozitivních a negativních instancí problému.

Pak definujte přesně pravidla hry pro hráče Eva („existenční hráč“) a Adam („univerzální hráč“) načrtnuté na přednášce. Jde o to, definovat hru tak, aby Eva měla vítěznou strategii (mimochodem, co to je vítězná strategie?) právě tehdy, když je zadaná formule pravdivá (a Adam měl vítěznou strategii právě tehdy, když je zadaná formule nepravdivá).

Zbude-li čas, nakonec ilustrujte na malém příkladu, jak lze obecnou plně kvantifikovanou booleovskou formuli ϕ převést (v polynomiálním čase) na ekvivalentní ϕ', která je ve tvaru požadovaném pro vstup problému QBF.

Příklad 13.2

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

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, ...)