Týden 10

Přednáška

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.

Připomenutí: 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.)

PSPACE, NPSPACE, PSPACE-úplnost

Uvědomili jsme si, 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 uvedli si Savitchovu větu (z učebního textu), která mj. implikuje PSPACE = NPSPACE.

Uvědomili jsme si 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). (Každý posluchač je jistě již schopen každý námi zkoumaný problém přesně specifikovat a uvést příklady pozitivních a negativních instancí ...)

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.

Partie textu k prostudování

Části 8.3., 8.4., 8.5. (třída PTIME, třída NPTIME, NP-úplné problémy). Kapitola 9 (speciálně Třída PSPACE).

Cvičení

Příklad 10.1

Definujte pojem *polynomiální převeditelnosti* jako speciální případ dříve uvedené (algoritmické) převeditelnosti mezi problémy. (Příslušný převádějící algoritmus musí mít polynomiální časovou složitost, tedy časovou složitost omezenou polynomem.)
Vysvětlete nejdříve přesně, co máme udělat, chceme-li prokázat polynomiální převeditelnost problému HC (hamiltonovský cyklus v orientovaném grafu) na problém HK (hamiltonovská kružnice v neorientovaném grafu).
Pak to udělejte.

Příklad 10.2

Vysvětlete pojem „NP-úplný problém“.
Definujte problémy SAT, 3-SAT, HC, HK, 3-CG a IS (s příklady pozitivních a negativních instancí). Tyto problémy jsou NP-úplné. U každého si alespoň uvědomte algoritmus, prokazující příslušnost k NP.

Příklad 10.3

Uvažujme následující problém (jeden z často uváděných NP-úplných problémů). (Už jsme jej v jiné souvislosti uvažovali minule.)

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.)

Příklad 10.4

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 $\triangleleft$ ILP.

(Zbytek příkladu je nepovinný.)

Zbude-li čas, zkuste tuto převeditelnost dokázat. Přinejmenším ale uveďte, co bychom mohli říci o složitosti problému ILP poté, co bychom prokázali 3-SAT $\triangleleft$ 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 10.5

Definujte třídu PSPACE a pojem „PSPACE-úplný problém“; příkladem je:

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

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

OTÁZKA: je daná formule pravdivá?

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

(Zbytek příkladu je nepovinný.)

Definujte pravidla hry pro hráče Eva („existenční hráč“) a Adam („univerzální hráč“), kteří postupně nasazují hodnoty proměnných. 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.