Úkol č. 4 do předmětu Složitost

Vojtěch Havlena (xhavle03)

1. Příklad
V případě, že uzel max má stupeň k, tak největší klika v grafu může být velikosti $k+1$, což nastane tehdy, když uzel max je součástí právě této největší kliky. Dále uvažujme následující třídu grafů $\{G_n | n \in \mathbb{N}, n \geq 2\}$, kde G_n je graf, který obsahuje uzel u stupně n (stupně ostatních uzlů jsou ostře menší než n) a zároveň největší klika, kterou je uzel u součástí má velikost 2. Dále největší klika, která v grafu existuje má velikost n (*příklad takového grafu).

Pokud na graf G_n použijeme algoritmus $klika$, dostaneme velikost 2 (i když velikost největší kliky je n). Nyní tedy předpokládáme, že algoritmus $klika$ je ϵ-aproximační pro nějaké $0 \leq \epsilon < 1$. Potom z definice ϵ-aproximačního algoritmu dostáváme

$$\frac{OPT(G_n) - |klika(G_n)|}{OPT(G_n)} = \frac{n-2}{n} = 1 - \frac{2}{n} \leq \epsilon,$$

což je ale spor, protože výraz $1 - \frac{2}{n}$ se pro rostoucí n blíží k 1 (limita výrazu je 1) a tudíž tento výraz není možné omezit nějakým pevným $\epsilon < 1$. Algoritmus $klika$ tedy není ϵ-aproximační pro žádné $\epsilon < 1$.

2. Příklad
Předpokládejme opak, tedy předpokládejme, že funkce $crypt$ je injektivní a $IMG_{crypt} \in NP \setminus UP$. Dále sestrojíme NTS M, který pro vstup $w \in IMG_{crypt}$ nedeterministiky uhodne $v \in \{0,1\}^*$ takové, že $w = crypt(v)$ (pokud takové v neexistuje, M vstup zamítne). M tedy pracuje tak, že nedeterministiky zvolí v a potom na něm simuluje výpočet funkce $crypt$. Platí tedy $L(M) = IMG_{crypt}$. Vzhledem k tomu, že $crypt \in FP$ – je spočitatelná v polynomiálním čase, i celý NTS M pracuje v polynomiálním čase. Navíc díky tomu, že $crypt$ je injektivní zobrazení, existuje pro NTS M a vstup w nejvýše jeden akceptační běh. A tím pádem $L(M) = IMG_{crypt} \in UP$. Což je ale spor s předpokladem. Platí tedy, že pokud $IMG_{crypt} \in NP \setminus UP$, potom $crypt$ není injektivní.

3. Příklad
NC redukci provedeme následovně. Na vstupu předpokládáme bezkontextovou gramatiku $G = (N, \Sigma, P, S)$ a řetězec w. Redukce pracuje ve třech krocích:

(a) K bezkontextové gramatice G vytvoříme zásobníkový automat P (přijímající s vyprázdněním zásobníku) takový, že $L(G) = L(P)$. Toto lze provést v konstantním čase pomocí $|P| + |\Sigma|$ procesorů v konstantním čase (každý procesor

z pravidla gramatiky udělá přechod ZA a pro každý symbol abecedy se přidá přechod typu $\delta(q, a, a)$, kde q je poč. stav a $a \in \Sigma$). Dále k řetězci w vytvoříme konečný automat A přijímající právě řetězec w. K tomu stačí $|w|$ procesorů s konstantním časem (každý přidá jeden přechod). Pro celý tento krok je tedy potřeba $|P| + |\Sigma| + |w| \leq n$ procesorů pracujících v konstantním čase.

(b) Provedeme průnik ZA P a konečného automatu A. Výsledný zás. automat $P \times A$ bude mít $S_1 \cdot S_2$ stavů (vzhledem k tomu, že ZA P má jeden stav a S_2 $|w|$ stavů, platí $S_1 \cdot S_2 \leq n$. Následně zjistíme přechodovou funkci. Každý procesor přidá nejvýše jeden přechod. Celkem je zapotřebí $(S_1 \cdot S_2) \cdot (|\Sigma| + 1) \cdot |\Gamma| \leq n^3$ procesorů, které ovšem pracují v konstantním čase.

(c) Nakonec ZA $P \times A$ převedeme zpět na bezkontextovou gramatiku. Pro to je nutné nejprve zjistit délky pravých stran pravidel a vygenerovat všechny posloupnosti stavů $P \times A$ o délce pravých stran pravidel P. Délka pravidel je konstanta nezávislá na vstupu, proto je toto možné provést v konstantním čase s využitím $O(n^c)$ procesorů (c je největší délka pravé strany). Nakonec už jen vygenerujeme pravidla gramatiky podle algoritmu (slidy k předmětu TIN). K tomu je potřeba polynomiální počet procesorů $O(n^{c+3}) \sim n^{O(1)}$ s konstantním časem.

Vzhledem k tomu, jak je redukce R navržena platí $(G, w) \in MEM \Leftrightarrow R(G, w) \in EMP$. Navíc R je ve třídě NC, protože je pro výpočet třeba polynomiální počet procesorů pracujících v konstantním čase.