Egzamin ze złożoności obliczeniowej – teoria. 6.6.2011

Imię i nazwisko:

Proszę wskazać i wyjaśnić błędy w poniższych rozumowaniach. Odpowiedź powinna się zmieścić na odwrocie tej kartki. Można też zaznaczyć bezpośrednio w tekście miejsca uznane za błędne.

1. Aby sprawdzić, czy liczba m jest dzielnikiem n, wystarczy dodawać m do siebie i badać, czy w końcu trafimy w n, czy też je przekroczymy. Do tego wystarczy $\mathcal{O}(\log n)$ pamięci. Zatem, testując kolejne $m < n$, możemy zbadać, czy liczba n jest pierwsza w pamieci jedynie logarytmicznej, a wiemy, że $L \subseteq P$. Dlaczego zatem tyle hałasu wokół algorytmu AKS wielomianowego rozpoznawania liczb pierwszych (z 2002 r.) ?

2. Niech $\varphi = \alpha_1 \wedge \ldots \wedge \alpha_k$, gdzie każde α_i jest klauzulą zawierającą *dwa* literały. Rozważmy

$$\begin{aligned} \varphi' &= (\alpha_1 \vee \neg x_1) \wedge \ldots \wedge (\alpha_k \vee \neg x_k) \wedge \\ &\quad (x_1 \vee x_1 \vee x_1) \wedge \ldots \wedge (x_k \vee x_k \vee x_k) \end{aligned}$$

Łatwo sprawdzić, że φ jest spełnialna wtedy i tylko wtedy, gdy φ' jest spełnialna, ale φ' ma już w kazdej klauzuli 3 literały, a wiemy, że problem 3-CNF SAT jest NP-zupełny. Ale właśnie opisaliśmy redukcję $\varphi \mapsto \varphi'$, zatem problem 2-CNF SAT jest także NP-zupełny. Jednak pamiętamy z ćwiczeń, że ten ostatni problem jest w P. Zatem wykazaliśmy, że $P = NP$.

3. Na wykładzie pokazano algorytm, która dla maszyny Turinga M i liczby n konstruuje obwód logiczny o n wejściach rozpoznający dokładnie słowa długości n akceptowane przez M. Ale każdą funkcję boolowską $\{0,1\}^n \to \{0,1\}$ można obliczyć obwodem o $\mathcal{O}(2^n)$ bramkach. Ewaluacji obwodu można dokonać w czasie wielomianowym od jego rozmiaru. A zatem każdą maszynę Turinga (przynajmniej, jeśli się zawsze zatrzymuje) można symulować przez maszynę pracującą w czasie wykładniczym. Co to więc za ściema z tym *Twierdzeniem o hierarchii czasowej* ?

4. Przypuśćmy, że algorytm probabilistyczny rozstrzyga język L z prawdopodobienstwem błedu $p \leq \frac{1}{4}$, dając odpowiedzi *tak* lub *nie*. Zatem, jeśli dla wejścia x otrzymamy na wyjściu odpowiedź *tak*, to z prawdopodobieństwem $1-p$ zachodzi $x \in L$, a jeśli otrzymamy odpowiedź *nie*, to z prawdopodobieństwem $1-p$ zachodzi $x \notin L$.

(Np. przypuśćmy, że algorytm rozstrzyga, czy $1 = 1$, rzucając 5 razy monetą. Jeśli wypadnie 5 reszek, mówi *nie*, poza tym *tak*. Przypuśćmy, że otrzymaliśmy wynik: *nie*. Czy zaczniemy wątpić, że $1 = 1$?)

5. Na wykładzie pokazano wielomianowy dowód interakcyjny, w wyniku którego Matematyk przekonuje Algorytm[1], że dwa grafy *nie* są izomorficzne. Zaadaptujmy ten protokół do sprawdzania, że dwie maszyny Turinga M_1 i M_2 są nierównoważne.

Dokładniej, Matematyk chce przekonać Algorytm, że $L(M_1) \neq L(M_2)$. W tym celu Algorytm losuje $i \in \{1,2\}$, a następnie, dokonując kilku dalszych losowań, modyfikuje M_i, tak że maszyna liczy wprawdzie to samo, ale „wygląda" zupełnie inaczej niż M_i. Tak otrzymaną maszynę N Algorytm przedstawia Matematykowi z pytaniem: *co to jest ?* (tzn. M_1 czy M_2).

Jeśli M_1 i M_2 są nierównoważne, Matematyk podaje jedyną dobrą odpowiedź. Jeśli jednak są równoważne, to Matematyk (który nie zna wyników losowań Algorytmu) może co najwyżej zgadnąć oczekiwane i z prawdopodobieństwem $\frac{1}{2}$.

Tym samym otrzymaliśmy dowód interakcyjny, rozstrzygajacy w czasie wielomianowym, czy dwie maszyny Turinga są nierównoważne, co jak wiadomo jest problemem *nierozstrzygalnym*. Tymczasem na wykładzie twierdzono (podając nawet odpowiednio zaciemniony „dowód"), że języki rozpoznawane przez dowody interakcyjne są nie tylko rozstrzygalne, ale nawet w *PSPACE* !

[1] W literaturze anglo-języcznej te postaci przedstawione są jako *Prover* i *Verifier*.