Theory Qual

Fall 2009

Please answer all four questions below.

1. Show that the following two definitions of hitting time of a directed graph G are equivalent up to a factor of 2:

   (a) the maximum over all pairs of vertices u and v of the minimum t such that the probability that a random walk of length t starting at u visits v is at least 1/2, and,

   (b) the maximum over all pairs of vertices u and v of the expected time t that a random walk starting at u first visits v.

2. Show that the following problem is complete for P^{NP} under polynomial-time mapping reductions (also known as polynomial-time many-one reductions): Given a Boolean formula ϕ in variables $x_1, \ldots, x_n$, is it the case that ϕ has at least one satisfying assignment and that the lexicographically least satisfying assignment $a_1 a_2 \ldots a_n$ sets $a_n = 1$?

3. Recall that for a complexity class $\mathcal{C}$ and a function $f : \mathbb{N} \to \mathbb{N}$, the class $\mathcal{C}/f$ consists of all languages L for which there exists a language $L' \in \mathcal{C}$ and a sequence of strings $a_0, a_1, a_2, \ldots$ with $|a_n| \leq f(n)$ such that $L = \{\, x : \langle x, a_{|x|} \rangle \in L' \}$. The class $\mathrm{P}^{\mathcal{C}[f]}$ contains all languages that can be decided in polynomial time with access to an oracle for a language in $\mathcal{C}$ such that no more than $f(n)$ oracle queries are made on an input of length n.

   Show the following for every positive constant c, where $\mathrm{NEXP} = \cup_{d>0}\mathrm{NTIME}(2^{n^d})$.

   (a) $\mathrm{NEXP} \not\subseteq \mathrm{DTIME}(2^{n^c})/n^c$.

   (b) $\mathrm{NEXP} \not\subseteq \mathrm{NP}/n^c$.

   (c) $\mathrm{NEXP} \not\subseteq \mathrm{P}^{\mathrm{NP}[n^c]}/n^c$.

   For parts (b) and (c), it suffices to prove (c) but (b) should point you in the right direction. *Hint:* Assume the opposite and obtain a contradiction with (a).

4. Two companies A and B make competing versions of n different software products. Company A charges p_i^A for item i and company B charges p_i^B for its version of i. A consumer wants to buy one version of each product. While the customer prefers cheaper versions, she also prefers to buy most software from the same company. In particular, items i and j, if bought from different companies, impose an incompatibility cost of $c(i, j)$ on the customer. The customer's total cost of buying software is the prices paid to the two companies, plus a sum over all pairs of the respective incompatibility costs.

   (a) Design a polynomial time algorithm to determine which company the customer could buy each item from to minimize her total cost.

   (b) Company A wants to increase the prices of its products to maximize its revenue, which is equal to the total price paid by the customer for all the items bought from A. Design a polynomial time algorithm to determine markups $k_i \geq 0$ such that if A increases its prices to $p_i^A + k_i$ for all i, its revenue is maximized (with B's prices staying the same as before). You may assume that the consumer breaks ties in favor of company A, that is, if there are two or more equally good buying choices for the consumer, she picks the one that gives the most revenue to company A.