MA421 Advanced Algorithms Lent 2019

Exercises 1

1. Define the optimization problem LONGEST-PATH-LENGTH as the relation that associates each instance of an undirected graph and two vertices with the number of edges in a longest simple path between the two vertices. Define the decision problem LONGEST-PATH as the set of strings $\langle G, u, v, k\rangle$ such that $G = (V, E)$ is an undirected graph, $u, v \in V$, k is an integer, and there exists a simple path from u to v in G consisting of at least k edges. Show that the optimization problem LONGEST-PATH-LENGTH can be solved in polynomial time if and only if LONGEST-PATH $\in \mathcal{P}$.

2. Show that if an algorithm makes at most a constant number of calls to polynomial-time subroutines and performs an additional amount of work that also takes polynomial time, then it runs in polynomial time. Also show that a polynomial number of calls to polynomial-time subroutines may result in an exponential-time algorithm.

3. Define the concatenation of languages L_1 and L_2 as $L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$ and the complement of a language L as $L^c = \{x \mid x \notin L\}$.

   Show that the class $\mathcal{P}$, viewed as a set of languages, is closed under union, intesection, concatenation, and complement. That is, if $L_1, L_2 \in \mathcal{P}$, then $L_1 \cup L_2 \in \mathcal{P}$, $L_1 \cap L_2 \in \mathcal{P}$, $L_1L_2 \in \mathcal{P}$, and $L_1^c \in \mathcal{P}$.

4. Two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a bijection $f : V_1 \to V_2$ such that, for any two vertices $u, v \in V_1$, $uv \in E_1$ if and only if $f(u)f(v) \in E_2$.

   Consider the language

   $$\text{GRAPH-ISOMORPHISM} = \{\langle G_1, G_2\rangle \mid G_1 \text{ and } G_2 \text{ are isomorphic graphs}\}.$$

   Prove that GRAPH-ISOMORPHISM $\in \mathcal{NP}$ by describing a polynomial-time algorithm to verify the language.

5. Prove that the class $\mathcal{NP}$ of languages is closed under union, intersection, and concatenation.

6. Show that any language in $\mathcal{NP}$ can be decided by an algorithm running in time $2^{O(n^k)}$ for some constant k.

Solutions to these exercises are to be handed in at the lecture on **January 22, 2019**.

