CSE 431 – Theory of Computation. Spring, 2014. Instructor: James R. Lee

ASSIGNMENT 4. Due Thursday, May 8th, in class (or via email to cse431-staff@cs before class starts)

1. **1.** Two undirected graphs H and G are called **isomorphic** if the nodes of G can be reordered so that it is identical to H. Define the language
$$ISO = \{\langle H, G\rangle : G \text{ and } H \text{ are isomorphic graphs}\}.$$
Show that $ISO \in \boldsymbol{NP}$.

2. **2. Dynamic programming:** Recall that if L is a language, then L^* is the language defined by
$$L^* = \{\, w_1 w_2 \cdots w_k : k \geq 0, w_i \in L \text{ for each } i \,\}$$
In other words, L^* contains strings that are concatenations of zero or more strings in L. The goal of this problem is to prove that if $L \in \boldsymbol{P}$ then $L^* \in \boldsymbol{P}$, where $\boldsymbol{P} = \bigcup_{k\geq 1} TIME(n^k)$ is the set of languages decidable in polynomial time.

This requires an idea known as "dynamic programming." Suppose we are given a string $y_1 y_2 \cdots y_n$ where each $y_i \in \Sigma$. We want to know if the string is in L^*. To do this, we build an $n \times n$ table A, where the entry $A[i,j]$ (for $i \leq j$) is supposed to represent whether the substring $y_i y_{i+1} \cdots y_j$ is in L^*. If we can build this table, then we can just look at the entry $A[1,n]$ to figure out if $y_1 y_2 \cdots y_n \in L^*$.

What's left is to see that we can fill in the table A in polynomial time. Remember that we have assumed $L \in \boldsymbol{P}$, so we have an algorithm that tests membership in L. We can easily fill in some entries of the table: For each $i = 1,2,\ldots,n$, we have $A[i,i] = 1$ if the string y_i is in L and $A[i,i] = 0$ otherwise. Now consider the following pseudocode to fill in the rest of A:

For $i = 1, 2, \ldots, n-1$, do:
    For $j = 1, 2, \ldots, n-i$
        $A[j, j+i] = \cdots$

Your goal is to fill in $A[i,j]$ using entries of the table A that have already been filled in. The whole algorithm should run in polynomial time, and at the end, we should have $A[i,j] = 1$ if $y_i y_{i+1} \ldots y_j \in L^*$ and $A[i,j] = 0$ otherwise. As we said before, $A[1,n]$ then contains the answer to whether $y_1 y_2 \ldots y_n \in L$.

3. Show that if $\boldsymbol{P} = \boldsymbol{NP}$ then every language $A \in \boldsymbol{P}$ except $A = \emptyset$ and $A = \Sigma^*$ is $\boldsymbol{NP}$-complete. (You can solve this problem just by thinking carefully about the notions of reduction and $\boldsymbol{NP}$-completeness.)

OPTIONAL PROBLEM (You may do this problem for extra credit, OR you can do it instead of the first three problems!)

Suppose we have n variables $x_1, x_2, \ldots, x_n$ and each variable can take only value 0 or 1. We also have an expression of the form

$$C_1 \cdot C_2 \cdot C_3 \cdots C_m$$

Where each C_i is of the form $C_i = \max(a_i, b_i)$ and each a_i or b_i is a variable x or $1 - x$. For instance, consider the following expression over the variables x_1, x_2, x_3:

$$E = \max(x_1, 1 - x_2) \cdot \max(1 - x_1, x_3) \cdot \max(x_1, x_3) \cdot \max(x_2, 1 - x_3)$$

You are given such a formula as input and the goal is to decide if there exists a setting of the variables to 0/1 values such that the expression equals 1. For example, for E there is a solution:

$$x_1 = 1, x_2 = 1, x_3 = 1$$

Plugging these in we get $E = \max(1,0) \cdot \max(0,1) \cdot \max(1,0) = 1 \cdot 1 \cdot 1 = 1$.

On the other hand, the expression

$$\max(x_1, 1 - x_2) \cdot \max(1 - x_1, 1 - x_2) \cdot \max(x_1, x_2) \cdot \max(1 - x_1, x_2) = 1$$

has no solution (you can try all four possible values for x_1, x_2).

Consider the language of formulas of this form that have a solution. Show that this language is in $\boldsymbol{P}$.