CSCI 338: Assignment 6 (7 points)

River Kelly

Monday, April 19

Problem 1

A ***triangle*** in an undirected graph is a 3-clique. Show that $TRIANGLE \in P$, where $TRIANGLE = \{\langle G \rangle \mid G \text{ contains a triangle}\}$.

Proof. To prove that $TRIANGLE \in P$, we must construct a Turning Machine M that decides $TRIANGLE$ in polynomial time. Let G be a graph with a set of V vertices and a set of E edges such as $G = (V, E)$. The TM will then enumerate all triples (x, y, z) with vertices $x, y, z \in V$ and $x < y < z$. Then the TM M will check whether or not all three edges, (x, y), (y, z) and (z, x) exist in E. TM M is defines as:

```
M = "On input $\langle G \rangle$, a graph:
    1. For each enumeration $(x, y, z) \in V \times V \times V$
    2.      Test if each edge $(x, y)$, $(y, z)$ and $(z, x) \in E$
        If test pass for each enumeration, accept
    3. If a triple does not pass the test, reject."
```

Enumeration of all triples requires $O(|V|^3)$ time, and checking whether or not each edge exists in E take $O(|E|)$ time. This would conclude that the overall time complexity is $O(|V|^3|E|)$, which is polynomial for input $\langle G \rangle$.

Notice that for $TRIANGLE$ a clique has a fixed size of 3, so even though it is an exponent, it is constant. Thus it is bound in time complexity.

$\therefore$ Since TM M decides in polynomial time, $TRIANGLE \in P$. $\square$

Problem 2

Let G represent an undirected graph. Also let

$$SPATH = \{\langle G, a, b, k\rangle \mid G \text{ contains a simple path of length at most } k \text{ from } a \text{ to } b\}$$

and

$$LPATH = \{\langle G, a, b, k\rangle \mid G \text{ contains a simple path of length at least } k \text{ from } a \text{ to } b\}$$

a.) Show that $SPATH \in P$

Proof. We will construct a TM which recognizes $SPATH$. It will use a modified algorithm of the marking algorithm. The TM is defined as:

M = "On input $\langle G, x, y, k\rangle$ where n-node graph G has nodes x and y:
1. Mark x
2. Iterate each from i to our mark"
3. If an edge (a, b) is found connecting a to an unmarked node c, mark node c with $i + 1$.
4. If node b's marked values is at most k, *accept*. Otherwise, *reject*."

This algorithm runs in polynomial time, therefore $SPATH$ must be in P. □

b.) Show that $LPATH$ is NP-complete

Proof. First, we will consider the reduction $UHAMPATH \leq_P LPATH$. Lets construct a TM M which computes such a reduction.

M = "On input $\langle G, x, y\rangle$ where graph G has nodes x and y:
1. Let k represent the number of node in G
2. Output $\langle G, x, y, k\rangle$."

G contains a Hamiltonian path of length k from a to b if $\langle G, x, y\rangle \in UHAMPATH$. Thus $\langle G, x, y, k\rangle \in LPATH$.

If $\langle G, x, y, k\rangle \in LPATH$, then G contains a simple path of length k from a to b. G only has k nodes, so the path is Hamiltonian. Thus, $\langle G, x, y\rangle \in UHAMPATH$.

Note, $LPATH$ must exist in NP because we can guess a simple path of length at least k from a to b and verify it in polynomial time.

$\therefore LPATH$ is NP-complete. $\square$

Problem 3

Let $DOUBLE - SAT = \{\langle\phi\rangle \mid \phi$ has at least two satisfying assignments$\}$.

Show that $DOUBLE - SAT$ is NP-complete.

Proof. We will create a non-deterministic polynomial time TM M which can guess two assignments. The TM M will accept if both assignments satisfy, showing that $DOUBLE - SAT$ is NP.

The polynomial time reduction of m:

M = "On input $\langle\phi\rangle$, a boolean formula with variables $x_1, x_2, \ldots, x_n$:
1. Let ϕ' be $\phi \wedge (x \vee \bar{x})$ where x is a new variable.
2. Output $\langle\phi'\rangle$."

If $\phi \in SAT$, then ϕ' has at least two satisfying assignments from the original assignment of ϕ by changing the value of x. If $\phi' \in DOUBLE - SAT$, then ϕ is also satisfied because x does not appear in ϕ.

$\therefore \phi \in SAT$ if, and only if, $m(\phi) \in DOUBLE - SAT$.

$\square$

Problem 4

A subset of the nodes of a graph G is a ***dominating set*** if every other node of G is adjacent to some node in the subset. Let

$$DOMINATING-SET = \{\langle G, k\rangle \mid G \text{ has a dominating set with } k \text{ nodes}\}$$

Show that it is NP-complete by giving a reduction from $VERTEX-COVER$.

Proof. We first show that $DOMINATING-SET \in NP$ by giving it a nondeterministic decider.

```
$A =$ "On input $\langle G, k\rangle$:
    1. Nondeterministically select a subset $V'$ with $|V'| = k$
    2. For each node $v$ in $G \setminus V'$
    3.      If there is no edge $(v, v')$ for any $v' \in V$, reject
    4. All nodes tested, accept"
```

The above machine is $O(|E|)$, so $DOMINATING-SET \in NP$. We now show, through a reduction of $VERTEX-COVER$, that $DOMINATING-SET$ is a member of NP-complete. Consider the following description of a Turing Machine.

For $\langle G, k\rangle$, an input to VC, we then construct the input to DS. For each edge $(u, v) \in G$, add the node w_{uv} and edges (u, w_{uv}) and (v, w_{uv}). Then match the output of DS.

Note that the reduction takes only polynomial time with the number of edges in G. As far as correctness goes, VC accepts graphs which have subsets of nodes such that each edge contains one of the nodes, while DS accepts graphs where nodes are adjacent. The above TM takes any edge and adds a node in it's place (while keeping the original graph), which would force a DS to contain the new node, and so contain the original edge. The other direction can be shown with the inverse mapping (deleting w_{uv}).

$\therefore$ $DOMINATING-SET$ is a member of NP and a NP-complete problem can reduce to $DOMINATING-SET$. So $DOMINATING-SET$ must be NP-complete. □