CS 525: Theory of Computation
Problem Set 7

Dustin Ingram, Aaron Rosenfeld, Eric Simon

March 14, 2012

7.9 **Solution:** Assuming the graph has n vertices and e edges, TRIANGLE can be decided with the following algorithm:

(a) For each set of three vertices:

  i. If each pair of nodes in the set shares an edge, return *true*.

(b) Return *false*.

Step (a) is run $\binom{n}{3} = \frac{(n-2)(n-1)n}{6} = O(n^3)$ times, once for each set of three vertices. Assuming the worst, where testing for a shared edge takes $O(e)$ time, the total running time of the algorithm is $O(n^3 \cdot e)$. This is in P.

7.13 **Solution:** Let Q be a permutation on set $\mathcal{S}$. For all $e \in \mathcal{S}$, let $\tau(e)$ be the permuted value of e after Q is applied. For example, if $(2, 1, 3) \xrightarrow{Q} (3, 2, 1)$ implies $\tau(1) = 2$, $\tau(2) = 3$, and $\tau(3) = 1$. Further, let $\tau^n(e)$ be the permuted value of e after n applications of Q.

Assuming $q = (q_1, q_2, ..., q_k)$ and $p = (p_1, p_2, ..., p_k)$, PERM-POWER can be decided as follows:

(a) For all $e \in \mathcal{S}$, calculate $\tau^t(e)$.

(b) For i from 1 to k:

  i. If $q_i \neq \tau^t(p_i)$, return *false*.

(c) Return *true*.

The computation of $\tau(e)$ clearly takes constant time as Q is simply a mapping over the set $\mathcal{S}$. Thus, computing $\tau^n(e)$ takes $O(n)$ time.

Step 1 takes $O(k \cdot t)$ time to pre-compute all τ^t. Step (a) is repeated k times (by Step 2), comparing q_i to $\tau^t(p_i)$. Since all τ^ts are already computed, this entire loop takes $O(k)$ time. Clearly Step 3 runs in constant time, so the entire algorithm takes $O(k \cdot t)$ time, which is in P.

7.14 **Solution:** We will demonstrate by showing how to find substring A in polynomial time. Thus, if A can be decided in polynomial time, so can

A^*. Build the following machine D:

On input $w = w_1 \ldots w_n$ Summary: $table(i, j)$ will contain the smallest A such that $w_i \ldots w_j \in A^*$. Note that the degenerate case where $A = w_i \ldots w_j$ always holds.

(a) For each $i = 1 \ldots n$: (examine each substring of length 1)

   i. Place $A = w_i$ into table entry (i, i), this indicating that w_i is a member of A^* with multiplicity 1.

(b) For each $k = 2 \ldots n$: (examine each substring of length k)

   i. For $i = 1 \ldots n - k + 1$ (i is the start position of the substring)

      A. $j = i + k - 1$ (j is the end position of the substring)

      B. Let $table(i, j) = w_i \ldots w_j$.

      C. For $m = i \ldots j-1$ (m is the split position), if $table(i, m)$ and $table(m+1, j)$ contain the same entry B, then set $table(i, j) = B$.

D has three nested loops that are each proportional to the length of w, thus D runs in $O(n^3)$ time. To decide w, first run machine D to build the table in $O(n^3)$ time. The result $table(1, n)$ is the smallest A such that $w \in A^*$. Next, we must decide A. Since we are assuming that $A \in P$, we can decide A in polynomial time. Thus, the overall time is in P.