6331 - Algorithms, Spring 2014, CSE, OSU
Homework 6
Instructor: Anastasios Sidiropoulos

Problem 1.

(a) Recall that a graph is called a *tree* if it is connected, and acyclic, i.e. it does not contain any cycles. Describe an algorithm which given an undirected graph $G = (V, E)$, decides whether G is a tree, with running time $O(|V|)$. Note that the required running time is *not* $O(|V|+|E|)$, which can be much larger when the input graph has many edges.

(b) Recall that a graph is a *simple cycle* if we can number its vertices as $\{1, 2, \ldots, n\}$, so that the edge set is $\{(1, 2), (2, 3), \ldots, (n - 1, n), (n, 1)\}$. Describe an algorithm which given an undirected graph $G = (V, E)$, decides whether G is a simple cycle, with running time $O(|V|)$. Note that your algorithm is *not* given the above numbering of the vertices.

Problem 2. An undirected graph $G = (V, E)$ is called *bipartite* if there exists a bipartition of V, such that no edge has both endpoints in the same part of the bipartition. Formally, there exist $U, U' \subset V$, with $U \cap U' = \emptyset$, and $V = U \cup U'$, and such that for every edge $(u, v) \in E$, we have either $u \in U$ and $v \in U'$, or $u \in U'$ and $v \in U$.

Describe an algorithm which given a graph $G = (V, E)$, decides whether G is bipartite, with running time $O(|V| + |E|)$.

Problem 3. An *Euler tour* of a strongly connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once.

(a) Show that G has an Euler tour if and only if for all $v \in V$, we have in-degree(v) = out-degree(v). Note that this is an "if and only if" statement. That is, you have to prove that both directions of the assertion hold.

(b) Describe an algorithm with running time $O(|E|)$, which finds an Euler tour of a given strongly connected directed graph G if one exists. (Hint: Merge edge-disjoint cycles.)