Homework 1

Handed out Monday, July 12. Due at the start of class Friday, July 16. Late homeworks will not be accepted.

A general note about writing algorithms: Henceforth, whenever you are asked to present an algorithm, you should present three things: the algorithm, a justification of its correctness, and a derivation of its running time. Please do not give detailed C++ or Java code. Only provide sufficient detail to convince the grader and me that your method is correct and complete. Try to keep your answers concise.

Problem 1. Recall that a tree is an undirected graph which is connected and acyclic. We call a node \( v \) of a tree a leaf node if \( \deg(v) = 1 \), all other nodes are called internal. The degree of a graph \( G \) is defined as \( \deg(G) = \max_{v \in V} \deg(v) \).

Let \( T \) be a tree with degree \( k \) and \( n \) nodes, note that \( T \) is not a rooted tree. Give a lower bound on the number of internal nodes as a function of \( n \). Show an example matching your bound for \( k = 2 \) and \( k = 3 \).

Problem 2. In a directed graph, a get-stuck vertex is one that has in-degree \( |V| - 1 \) and out-degree 0. Assume that the adjacency matrix representation is used. Design an \( O(V) \) algorithm to determine if a given graph has a get-stuck vertex. (Yes, this problem can be solved without even looking at the entire input matrix.)

Problem 3. A bipartite graph is one whose vertex set can be partitioned into two sets \( A \) and \( B \), such that each edge in the graph goes between a vertex in \( A \) and a vertex in \( B \). (No edges between nodes in the same set are allowed.)

(a) Show that if \( G \) is bipartite then it does not have an odd length cycle.
(b) Give an \( O(E+V) \) algorithm that takes an input graph (represented in adjacency list form) and decides if the graph is bipartite. If the graph is bipartite, the algorithm should also produce the bipartition.
(c) Show that if a graph \( G \) does not have an odd length cycle then \( G \) is bipartite. This means that the above definition for the class of bipartite graphs and “does not have an odd length cycle” are equivalent!

Problem 4. A championship graph is a directed graph in which for any two distinct vertices \( u \) and \( v \) either there is an edge from \( u \) to \( v \) or there is an edge from \( v \) to \( u \), but not both. Prove that in every championship graph there is a path that visits every vertex \textit{exactly} once and provide an \( O(E) \) algorithm for finding such a path.