Solutions to Problem 1 of Homework 11 (8 points)

Name: **** INSERT YOUR NAME HERE ****  Due: Wednesday, April 29

(a) (5 points) Design $O(n)$ algorithm to test if a given undirected graph $G$ is acyclic. Notice, the running time of your algorithm should not depend on the number of edges $m$!

(Hint: Could you argue faster termination of a regular DFS tester on undirected graph?)

Solution: **************** INSERT YOUR SOLUTION HERE ****************

(b) (3 points) Extend the above algorithm to actually print the cycle, in case $G$ is cyclic.

Solution: **************** INSERT YOUR SOLUTION HERE ****************
Your job is to arrange \( n \) rambunctious children in a straight line, facing front, i.e., in the direction of the line. You are given a list of \( m \) statements of the form \( i \) hates \( j \). If \( i \) hates \( j \), then you do not want put \( i \) somewhere behind \( j \), because then \( i \) is capable of throwing a pebble at \( j \).

(a) (4 points) Give an algorithm that orders the line, (or says that it is not possible) in \( O(m + n) \) time.

**Solution:** **************** INSERT YOUR SOLUTION HERE ************

(b) (6 points) Suppose instead you want to arrange the children in rows, such that if \( i \) hates \( j \) then \( i \) must be in a (strictly) lower numbered row than \( j \). Give an efficient algorithm to find the minimum number of rows needed, if it is possible.

**Solution:** **************** INSERT YOUR SOLUTION HERE ************
Assume $G$ is an undirected graph with weight function $w$, and $e_1 \ldots e_m$ are the $m$ edges of $G$ sorted according to their weight: $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$. Imagine you just ran the Kruskal’s algorithm of $G$ and it output an MST $T$ of $G$. Now assume that somebody changes the weight of a single edge $e_i$ from $w(e_i)$ to some other value $w'$. For each of the following 4 scenarios, describe the fastest algorithm you can think of to transform the original MST $T$ of $G$ to a new (and correct) MST $T'$ of $G$ after the edge weight change. Make sure you justify your answer, and express your running time as a function of $m$ and $n$.

(a) (4 points) Assume $e_i \in T$ and $w' < w(e_i)$ (so we decreased an MST edge).

Solution: **************** INSERT YOUR SOLUTION HERE ****************

(b) (4 points) Assume $e_i \not\in T$ and $w' < w(e_i)$ (so we decreased a non-MST edge).

(Hint: Compute the unique shortest path in $T$ between the two end-points of $e_i$.)

Solution: **************** INSERT YOUR SOLUTION HERE ****************

(c) (4 points) Assume $e_i \not\in T$ and $w' > w(e_i)$ (so we increased a non-MST edge).

Solution: **************** INSERT YOUR SOLUTION HERE ****************

(d) Extra Credit: (6 points) Assume $e_i \in T$ and $w' > w(e_i)$ (so we increased an MST edge).

(Hint: Try to find the smallest weight edge $e_j$ which should replace $e_i$ under the new weight.)

Solution: **************** INSERT YOUR SOLUTION HERE ****************
(a) (5 points) Let \( e \) be the maximum weight edge on some cycle of a connected graph \( G = (V, E) \). Prove that there exists an MST \( T \) of \( G' = (V, E \setminus \{e\}) \) which is also an MST of \( G \). Namely, some MST of \( G \) does not include \( e \).

Solution: **************** INSERT YOUR SOLUTION HERE ****************

(b) (2 points) Consider the following idea for a greedy algorithm for finding some minimal spanning tree in an undirected weighted graph \( G \):

(I) find a cycle in \( G \).
(II) if there is no cycle, output \( G \) and terminate;
(II) else, if there is a cycle,
    * find an edge \( e \) with maximum weight on this cycle;
    * remove \( e \) from the graph \( G \);
    * return to step (I).

Prove correctness of this algorithm using induction and part (a).

Solution: **************** INSERT YOUR SOLUTION HERE ****************

(c) (4 points) Describe details of the fastest implementation you can find for the algorithm in part (b) (or write a pseudocode), and analyze its complexity as a function of \( n \) (number of vertices) and \( m \) (number of edges).

Solution: **************** INSERT YOUR SOLUTION HERE ****************