CS 430 Introduction to Algorithms	**Spring Semester, 2021**

Homework 5 Version 1.3

Assigned: April 7 — *Due: April 21*

Please respect the following guidelines for writing pseudocode:

1. C/Java/Python instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter.
2. One instruction per line
3. Match the brackets with a horizontal line
4. Number your lines
5. Write down if your array is indexed $0 \dots n-1$ or $1 \dots n$.

Problem 1 Given a directed graph $G = (V, E)$ we define the graph $G^2 = (V, E^2)$, such that $(u, w) \in E^2$ if and only if for some $v \in V$, both $(u, v) \in E$ and $(v, w) \in E$. That is, G^2 contains an edge between u and w whenever G contains a path with exactly two edges between u and w. Describe an efficient algorithm for computing the adjacency matrix of G^2 given the adjacency matrix of G. Present the pseudocode and analyze the running time in terms of $|V|$ and $|E|$.

(This was on a previous final exam)

Problem 2 Exercise 22.4-1 page 614 from the textbook. It has the same number (but different page) in the second edition.

Note: the DFS-based algorithm must be used, not the one from the next problem.

Problem 3 There is another $O(|V| + |E|)$ method for topological sort. Repeatedly find a vertex of in-degree 0, print it, and "remove it from the graph" by adjusting the in-degree of the other nodes as if this node was removed. Give pseudocode and analyze the running time. Argue correctness. Discuss what happens if the input graph is cyclic.

Problem 4 Consider the following divide-and-conquer algorithm for computing minimum spanning trees in graphs with non-negative weights. Given a complete graph $G = (V, E)$, partition the set V of vertices into two sets V_1 and V_2 such that $|V_1|$ and $|V_2|$ differ by at most 1. Let E_1 be the set of edges that are incident only on vertices in V_1, and let E_2 be the set of edges that are incident only on vertices in V_2. Recursevely solve a minimum spanning tree problem on each of the two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edge in e that crosses the cut (V_1, V_2), and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

Either argue the algorithm correctly computes a minimum spanning tree of G (regardless of how the partition is done), or provide an example for which the algorithm fails, showing the run of the algorithm (where you can choose the partition) and a better spanning tree.

(This was on a previous final exam)