Algorithms Final Exam

June 15, 2020, 09:10 - 12:10

Please answer the following questions and submit a photo/scan of your answer as a single pdf file on COOL. This is a closed book exam. All exam rules apply. You are not allowed to discuss with anyone or read any related materials, with the exception of one double-sided, hand-written A4 note previously prepared.

If you want to use any result or theorem that has been taught in class (including homeworks), you may do so but you must state the result or theorem clearly before using it. You may assume that all arithmetic operations take $O(1)$ time.

Problem 1 (15%)

Given a directed acyclic (= no cycle) graph $G = (V, E)$ and two nodes $u, v \in V$, find the number of paths from u to v. (all paths, not necessarily the shortest.) Notice that you do not need to specify all paths, just counting the number is sufficient. Your algorithm should run in time $O(V + E)$ in order to receive full credit. Briefly justify the correctness and analyze the running time.

Problem 2 (15%)

Given the graph (map) of a region consisting of n nodes (cities) and m edges (roads). The n cities are labeled with $v_1, v_2, \ldots, v_n$. Each road e has a (possibly different) driving time w_e. A person wants to drive from v_1 to v_n. He may take a one hour rest every time he passes through a city. He does not have to rest at every city he passes through but he can only drive at most k hours between any two rests. Given that all w_e and k are integers. Design an algorithm which finds the shortest trip (driving time and resting time combined). Your algorithm must run in time $O(k^2(m + n))$ in order to receive full credits. Briefly justify the correctness and analyze the running time.

Problem 3 (10%)

Given an undirected graph G in which every vertex has a positive weight. (Edges have weight 0.) Given two vertices s, t. Design an algorithm which finds the shortest path (= minimum total vertex weights) from s to t. Your algorithm must run in time $O(E + V \log V)$ in order to receive full credits. Briefly justify the correctness and analyze the running time.

Problem 4 (15%)

1. (8%) In Prim's algorithm, at each round, if there are multiple edges with equal minimum costs, the algorithm picks one of them arbitrarily. Consider the following modification: if adding all minimum-cost edges do not create cycles, the modified algorithm adds all of them to the output instead of just an arbitrary one. Is the modified algorithm still guaranteed to find the MST? Prove the correctness of the modified algorithm or find a counter example. (You do not have to worry about the running time and/or implementation and only need to focus on the correctness.)

2. (7%) Repeat the above problem for Kruskal's algorithm.

For problems 5 and 6, you must either

1. design an algorithm with polynomial running time, briefly justify the correctness, (hint: One possible solution uses max flow.)

   or

2. prove that the problem is NP-complete. You must describe the reduction in detail and briefly explain the correctness of your reduction.

You may use the fact that SAT, 3-SAT, Vertex Cover, Set Cover, Independent Set, Hamiltonian Path and Hamiltonian Cycle problems are all NP-complete. Also, you do not need to prove that problems 5, 6 are in NP.

Problem 5 (15%)

Given a set $U = \{x_1, x_2, \ldots, x_n\}$, m subsets $S_1, S_2, \ldots, S_m \subseteq U$, and an integer k. Determine whether there exists a subset $X \subseteq U$ such that X has exactly k elements and $|X \cap S_i| \geq 2$ for all i (i.e. the intersection of X and any S_i has at least two elements). If you cannot manage this problem, change "$|X \cap S_i| \geq 2$" to "$|X \cap S_i| \geq 1$" gives partial credits.

Problem 6 (15%)

Given an n by n integer matrix, determine whether it is possible to permute (exchange) rows such that all diagonal elements $a_{11}, a_{22}, \ldots, a_{nn}$ are even numbers.

Problem 7 (15%)

The goal of this problem is to find a minimum vertex cover for undirected graphs in which every vertex has degree 3 (i.e. has exactly 3 neighbors). Design a $\frac{3}{2}$-approximation algorithm which runs in time $O(V)$. (Hint and partial credit: Compute the approximation ratio if an algorithm always outputs all vertices as a vertex cover.) Briefly justify the correctness and analyze the running time.