CS 344: Design and Analysis of Computer Algorithms **Rutgers: Spring 2021**

Midterm Exam #2

Due: Tuesday, April 13, 9:00am EST

Name: _____ *NetID:* _____

Instructions

1. Do not forget to write your name and NetID above, and to sign Rutgers honor pledge below.
2. The exam contains 4 problems worth 100 points in total *plus* one extra credit problem worth 10 points.
3. This is a take-home exam. You have exactly 24 hours to finish the exam.
4. The exam should be done **individually** and you are not allowed to discuss these questions with anyone else. This includes asking any questions or clarifications regarding the exam from other students or posting them publicly on Piazza (any inquiry should be posted privately on Piazza). You may however consult all the materials used in this course (video lectures, notes, textbook, etc.) while writing your solution, but **no other resources are allowed**.
5. Remember that you can leave a problem (or parts of it) entirely blank and receive 25% of the grade for that problem (or part). However, this should not discourage you from attempting a problem if you think you know how to approach it as you will receive partial credit more than 25% if you are on the right track. But keep in mind that if you simply do not know the answer, writing a very wrong answer may lead to 0% credit.

   The only **exception** to this rule is the extra credit problem: you do not get any credit for leaving the extra credit problem blank, and it is harder to get partial credit on that problem.
6. **You should always prove the correctness of your algorithm and analyze its runtime.** Also, as a general rule, avoid using complicated pseudo-code and instead explain your algorithm in English.
7. You may use any algorithm presented in the class or homeworks as a building block for your solutions.

Rutgers honor pledge:

On my honor, I have neither received nor given any unauthorized assistance on this examination.

Signature:_____

Problem. #	Points	Score
1	25	
2	25	
3	25	
4	25	
5	+10	
Total	100 + 10	

Problem 1.

(a) Suppose $G = (V, E)$ is any undirected graph and $(S, V - S)$ is a cut with zero cut edges in G. Prove that if we pick two arbitrary vertices $u \in S$ and $v \in V \setminus S$, and add a new edge (u, v), in the resulting graph, there is no cycle that contains the edge (u, v). **(12.5 points)**

(b) Suppose $G = (V, E)$ is an undirected graph with weight w_e on each edge e. Prove that if the weight of some edge f is *strictly larger* than weight of all other edges in *some cycle* in G, then *no* minimum spanning tree (MST) of G contains the edge f.

(Note that to prove this statement, it is *not* enough to say that some specific algorithm for MST never picks this edge f; you have to prove *no* MST of G can contain this edge). **(12.5 points)**

Problem 2. We are given a directed acyclic graph $G = (V, E)$ with a unique source s and a unique sink t. We say that an edge e in G is a *bottleneck edge* if *every* path from s to t passes through this edge e. Design and analyze an algorithm for finding *all* bottleneck edges in G in $O(n + m)$ time.

(a) *Algorithm (or graph reduction)*: **(10 points)**

(b) *Proof of Correctness*: **(10 points)**

(c) *Runtime Analysis*: **(5 points)**

Problem 3. Crazy City consists of n houses and m bidirectional streets connecting these houses together, and there is always at least one way to go from any house to another one following these streets. For every street e in this city, the cost of maintaining this street is some positive integer c_e. The mayor of Crazy City has come up with a brilliant cost saving plan: destroy(!) as many as the streets possible to maximize the cost of destroyed streets (so we no longer have to pay for their maintenance) while only ensuring that there is still a way for every house to reach mayor's house following the remaining streets.

Design an $O(m \log m)$ time algorithm that outputs the set of streets with *maximum total cost* that should be destroyed by the mayor.

(a) *Algorithm (or graph reduction)*: **(10 points)**

(b) *Proof of Correctness*: **(10 points)**

(c) *Runtime Analysis*: **(5 points)**

Problem 4. We are given a weighted undirected graph $G = (V, E)$ with positive weight w_e over each edge e, and two vertices $s, t \in V$. Additionally, we are given a list L of k *new* edges (not in G), where each edge $f \in L$ has some weight w_f. Our goal is to pick *exactly one* edge f from L to add to G such that we minimize the weight of the shortest path from s to t in the resulting graph $G + f$.

Design an $O(k + n + m \log m)$ time algorithm that determines which edge from L should be added to G.

(a) *Algorithm (or graph reduction)*: **(10 points)**

(b) *Proof of Correctness*: **(10 points)**

(c) *Runtime Analysis*: **(5 points)**

Problem 5. [**Extra credit**] You are given an undirected graph $G = (V, E)$ and two vertices s and t in G. Design and analyze an algorithm that in $O(n + m)$ time, decides if the number of *different shortest paths* from s to t is an *odd* number or an *even* one. **(+10 points)**

Hint: This problem is both related to Problem 3 of your homework 3 and also very different: here, you are looking for the number of *shortest* paths (not arbitrary paths) and in an undirected graph (not in a DAG). You should however still feel free to use a graph reduction to that problem if you see a proper way.

Extra Workspace