CS 344: Design and Analysis of Computer Algorithms **Rutgers: Spring 2022**

Midterm Exam #2

April 07, 2022

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, starting from Thursday, April 07, 9am EST until Friday, April 08, 9am EST.
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 – also no hints or verifying a partial solution will be given during the exam**). 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 there is almost no 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 $v \in V - S$, 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. Additionally, suppose that G has an edge f with weight *strictly larger* than all other edges in G. Prove that if f belongs to a *cycle* in G, then *no* minimum spanning tree (MST) of G can contain the edge f. **(12.5 points)**

Problem 2. You have a bag of m cookies and a group of n friends. For each friend $1 \leq i \leq n$, you know the "greed factor" of your friend as a number $G[i]$: this is the minimum number of cookies you should give to this friend to make them stop complaining. Of course, you would like to find a way to distribute your cookies in a way to *minimize* the number of your friends that are still complaining.

Design an $O(n \log n)$ time greedy algorithm to find an assignment of the cookies to your friends that minimizes the number of the friends that are still complaining: recall that a friend i stops complaining if we assign them $c_i \geq G[i]$ cookies. **(25 points)**

Assume that the greed factors of the friends are all distinct. The output should be a list of n pairs consisting of a friend identified by their unique greed factor, and the number of cookies assigned to this friend.

Example: An example with $m = 11$ and $n = 6$ is as follows.

- **Input:** The array of greed factors $G = [6, 3, 5, 2, 8, 7]$.
- **Output:** A list of pairs with first elements chosen from G to identify the friend and the second elements denoting the number of cookies assigned to them:

$$[(5, 5), (3, 3), (2, 3), (6, 0), (8, 0), (7, 0)].$$

  This means that the friends with greed factors $5, 3,$ and 2, each received $5, 3,$ and 3 cookies respectively, while the remaining friends received zero cookies. This reduces the number of friends who are still complaining to three.

Problem 3. You are given a directed graph $G = (V, E)$ with blue and red edges, and two vertices s and t. Design and analyze an algorithm that outputs whether there exists a *walk* from s to t in G that contains an *even* number of red edges; the walk can contain an arbitrary number of blue edges. **(25 points)**

A complete solution consists of three part: the algorithm or reduction (**10 points**), the proof of correctness (**10 points**), and the runtime analysis (**5 points**).

Problem 4. A **feedback edge set** of an undirected connected graph $G = (V, E)$ is a set of edges $F \subseteq E$ such that any cycle in G has at least one edge in F. In other words, removing the edges in F from G leaves the graph $G - F$ without any cycle. Describe and analyze an $O(m \log m)$ time algorithm to compute the minimum-weight feedback edge set of a given graph $G = (V, E)$ with *positive* weight w_e on each edge $e \in E$.

(25 points)

A complete solution consists of three part: the algorithm or reduction (**10 points**), the proof of correctness (**10 points**), and the runtime analysis (**5 points**).

Hint: Observe that the graph $G - F$ should be a spanning tree. (You have already seen the rest of the solution in your homework!)

Problem 5 (**Extra Credit**)**.** Consider the following different (and less efficient) algorithm for computing an MST of a given undirected and connected graph $G = (V, E)$ with edge weight w_e on each $e \in E$:

1. Sort the edges in decreasing (non-increasing) order of their weights.
2. Let $H = G$ be a copy of the graph G.
3. For $i = 1$ to m (in the sorted ordering of edges):
   (a) If removing e_i from H does not make H disconnected, remove e_i from H.
4. Return H as a minimum spanning tree of G.

Prove the correctness of this algorithm, i.e., that it outputs an MST of any given graph G (we ignore the runtime of this algorithm in this problem). **(+10 points)**

Extra Workspace