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

Homework #4

Deadline: Monday, April 05, 11:59 PM

Homework Policy

- If you leave a question completely blank, you will receive 25% of the grade for that question. This however does not apply to the extra credit questions.
- You are allowed to discuss the homework problems with other students in the class. **But you must write your solutions independently.** You may also consult all the materials used in this course (video recordings, notes, textbook, etc.) while writing your solution, but no other resources are allowed.
- Do not forget to write down your name and whether or not you are using one of your two extensions. Submit your homework on Canvas.
- Unless specified otherwise, you may use any algorithm covered in class as a "black box" – for example you can simply write "use Prim's or Kruskal's algorithm to find an MST of the input graph in $O(m \log m)$ time". You are **strongly encouraged to use graph reductions** instead of designing an algorithm from scratch whenever possible (even when the question does not ask you to do so explicitly).
- Remember to always **prove the correctness** of your algorithms and **analyze their running time** (or any other efficiency measure asked in the question).
- The "Challenge yourself" and "Fun with algorithms" are both extra credit. These problems are significantly more challenging than the standard problems you see in this course (including lectures, homeworks, and exams). As a general rule, only attempt to solve these problems if you enjoy them.

Problem 1. You are given the map of n cities with m bidirectional roads between different cities. You are asked to construct airports in some of the cities such that each city either has an airport itself or there is a way to go from this city to a city with an airport using the given roads—moreover, you can also construct a road between any two cities if there is no road between them already. Finally, you are told that the cost of constructing an airport is a and the cost of connecting any two cities by a new road is r.

Design and analyze an $O(n+m)$ time algorithm that given the number n of the cities, the m roads between them, and the costs a and r, outputs the locations of airports and roads to be constructed to satisfy the conditions above, while having the minimum possible cost. **(25 points)**

Examples:

1. *Input:* $n = 4$ cities with $m = 2$ roads $(1,2),(2,3)$, cost of constructing an airport $a = 7$, and constructing a road is $r = 5$.

   Output: The minimum cost needed is 12 – we construct an airport in city 4 and connect it this city via a road to any of the cities 1, 2, or 3 (chosen arbitrarily).

2. *Input:* $n = 4$ cities with $m = 2$ roads $(1,2),(2,3)$, cost of constructing an airport $a = 7$, and constructing a road is $r = 8$.

   Output: The minimum cost needed is 14 – we construct an airport in city 4 and another one in any one of the cities 1, 2, or 3 (chosen arbitrarily).

Problem 2. We say that an undirected graph $G = (V, E)$ is **2-edge-connected** if we need to remove *at least two* edges from G to make it disconnected. Prove that a graph $G = (V, E)$ is 2-edge-connected if and only if for every cut $(S, V - S)$ in G, there are *at least two cut edges*, i.e., $|\delta(S)| \geq 2$. **(25 points)**

Problem 3. The Muddy City consists of n houses but no proper streets; instead, the different houses are connected to each other via m bidirectional muddy roads. The newly elected mayor of the city aims to pave some of these roads to ease the travel inside the city but also does not want to spend too much money on this project, as paving each road e between houses u and v has a certain cost c_e (different across the muddy roads). The mayor thus specifies two conditions:

- Enough streets must be paved so that everyone can travel from any house to another one using only the paved roads (you may assume that this is always possible);
- The paving should cost as little as possible.

You are chosen to help the mayor in this endeavor.

(a) Design and analyze an $O(m \log m)$ time algorithm for this problem. **(10 points)**

(b) The mayor of a neighboring city is feeling particularly generous and has made the following offer to Muddy City: they have identified a list of $O(\log m)$ different muddy roads in the city and are willing to entirely pay the cost of paving *exactly one* of them (in exchange for calling the new street after the neighboring city).

Design and analyze an $O(m \log m)$ time algorithm that identifies the paving of which of these roads, if any, the mayor should delegate to the neighboring city to further minimize the total cost—note that if you decide to pave one of the roads payed by the neighboring city, you only need to pay a cost of 1 (for making a plaque of the name of the street). **(15 points)**

Hint: Design an algorithm that given a MST T of a graph G, and a single edge e, in only $O(m)$ time finds an MST T' for the graph G' obtained by changing the weight of the edge e to 1.

Problem 4. You are given a weighted undirected graph $G = (V, E)$ with integer weights $w_e \in \{1, 2, \ldots, W\}$ on each edge e, where $W = O(1)$. Given two vertices $s, t \in V$, the goal is to find the minimum weight path (or shortest path) from s to t. Recall that Dijkstra's algorithm solves this problem in $O(n + m \log m)$ time even if we do not have the condition that $W = O(1)$. However, we now want to use this extra condition to design an even faster algorithm.

Design and analyze an algorithm to find the minimum weight (shortest) s-t path in $O(n + m)$ time.

Challenge Yourself. A **bottleneck spanning tree (BST)** of a undirected connected graph $G = (V, E)$ with positive weights w_e over each edge e, is a spanning tree T of G such that the weight of maximum-weight edge in T is minimized across all spanning trees of G. In other words, if we define the cost of T as $\max_{e \in T} w_e$, a BST has minimum cost across all spanning trees of G. Design and analyze an $O(n + m)$ time algorithm for finding a BST of a given graph. **(+10 points)**

Fun with Algorithms. Let us go back to the bottleneck spanning tree (BST) problem defined above.

(a) Prove that any MST of any graph G is also a BST. Use this to obtain an $O(m \log m)$ time algorithm for the BST problem (notice that this is slower than the algorithm from the previous question). **(+8 points)**

(b) Give an example of a BST of some graph G which is *not* an MST in G. **(+2 points)**