CIS 1210 — Data Structures and Algorithms
Homework Assignment 7

Assigned: October 29, 2024 **Due:** November 4, 2024

Note: The homework is due **electronically on Gradescope** on November 4, 2024 by 11:59 pm ET. No late submissions will be accepted for this assignment.

A. **Gradescope**: You must select the appropriate pages on Gradescope. Gradescope makes this easy for you: before you submit, it asks you to associate pages with the homework questions. Forgetting to do so will incur a 5% penalty, which cannot be argued against after the fact.

B. **LaTeX**: You must use the LaTeX template provided on the course website, or a 5% penalty will be incurred. Handwritten solutions or solutions not typeset in LaTex will not be accepted.

C. **Solutions**: Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear. Please refer to the Written Homework Guidelines for all the requirements. Piazza will also contain a complete sample solution in a pinned post.

D. **Algorithms**: Whenever you present an algorithm, your answer must include 3 separate sections. Please see Piazza for an example complete solution.

1. A precise description of your algorithm in English. No pseudocode, no code.
2. Proof of correctness of your algorithm
3. Analysis of the running time complexity of your algorithm

E. **Collaboration**: You are allowed to discuss **ideas** for solving homework problems in groups of up to 3 people but *you must write your solutions independently.* Also, you must write on your homework the names of the people with whom you discussed. For more on the collaboration policy, please see the course webpage.

F. **Outside Resources**: Finally, you are not allowed to use *any* material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class. If you're unsure if something violates our policy, please ask.

1. [20 pts] Zimo's Ballot

Zimo is excited to vote in the 2024 election! However, he's busy on election day so he plans to mail in his ballot ahead of time. To help minimize costs, he want to plan the path his ballot takes through the n post offices to reach his destination post office. He won't know which post office is the destination until he's sending his ballot in!

For his ballot to travel between any two post offices, there is some cost, since it needs to be driven on a road and gas is expensive :(.

Deciding to make use of some CIS 1210 knowledge, Zimo devises a strategy to minimize costs. He decides to use Djikstra's algorithm to construct a graph $G = (V, E)$ where V represents the set of all post offices, and E represents the set of roads between locations. The weights of the edges indicate the cost of sending the ballot on that road. Thus, Djikstra's algorithm will return the route to the destination that minimizes cost.

However, just before he mails his ballot, Zimo realizes that there's a rebate program for mailing in his ballot. For only the first road originating from the Penn post office, the government will actually pay Zimo the cost of mailing the ballot (that is, the weight of the first edge is negative). Does Zimo's previous strategy of using Djikstra's algorithm still work to minimize the cost of mailing in his ballot?

Either prove that Zimo's plan still works, or disprove the claim by finding a counterexample.

Note: Only the roads leading out of the Penn post office will be paid for by the government. You may assume that there are no negative weight cycles.

2. [20 pts] Ha the Hero

This upcoming Election Day, Ha's proclaimed herself as the official Penn ballot-collector. Ha wants to set the record for being the fastest collector in Penn's history and needs your help to plan routes between the voting booths.

There are n voting booths, $p_1, \dots, p_n$ connected by m one-way routes on campus, each with a non-negative distance. Note that there may be a unique trail from p_i to p_j with distance w_1, but there may also be a unique trail from p_j to p_i with distance w_2. Ha wants to know the most efficient path between each pair of voting booths. Additionally, when going from one position to another, Ha wants to make a stop at her dorm room p^* in between to drop off the ballots.

This means that, if Ha wants to go from p_i to p_j, the path must go from p_i to p^*, then from p^* to p_j, with the total distance being minimized. You may assume such a path exists between all pairs of positions.

Ha is certain that she will set the record time for ballot collection. Help Ha secure her place in the Penn Hall of Fame by writing an algorithm that outputs a 2D array containing the length of the shortest path for *each pair of positions* while ensuring that each route passes through p^*. Your algorithm should run in $O(n^2 + m \log n)$ time.

3. [20 pts] The Ultimate Ballot Delivery Plan

It's election season, and the CIS 1210 TAs are taking part in a special initiative to collect and deliver all ballots from the stations around campus. Atharv is in charge of planning the delivery routes. There are $n \geq 10$ polling stations around campus, each connected by $n + 15$ undirected routes, where each route has a distinct length.

At the start of each round of deliveries, Atharv will be given two polling stations, x and y, and he must deliver ballots from station x to station y using the existing routes, as quickly as possible. The cost of traversing a route is directly proportional to it's length: that is, if the route between stations P and Q has a longer length than the route between stations P and R, the cost of taking route PQ will be higher than that of route PR.

Atharv wants to devise a plan to minimize the total cost of transferring ballots by eliminating certain routes. Specifically, he aims to reduce the total cost of traversing the routes while ensuring that any station x can still be reached from any other station y. This ensures that he can still take the fastest path between any two stations in each delivery round.

Help Atharv devise an $O(n)$ time algorithm that finds and returns a set of routes whose removal will reduce the total cost the most while maintaining connectivity. You may assume that you have access to a list containing information about the endpoints of each route and its corresponding length.