CS 170 Homework 4

Due **2/20/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

Solution: I worked on this homework with the following collaborators:

- Lakshya Nagal, SID: 3037935253

2 Arbitrage

Shortest-path algorithms can also be applied to currency trading. Suppose we have n currencies $C = \{c_1, c_2, \ldots, c_n\}$: e.g., dollars, Euros, bitcoins, dogecoins, etc. For any pair of currencies c_i, c_j, there is an exchange rate $r_{i,j}$: you can buy $r_{i,j}$ units of currency c_j at the price of one unit of currency c_i. Assume that $r_{i,i} = 1$ and $r_{i,j} \geq 0$ for all i, j.

The Foreign Exchange Market Organization (FEMO) has hired Oski, a CS170 alumnus, to make sure that it is not possible to generate a profit through a cycle of exchanges; that is, for any currency $i \in C$, it is not possible to start with one unit of currency i, perform a series of exchanges, and end with more than one unit of currency i. (That is called *arbitrage*.)

More precisely, arbitrage is possible when there is a sequence of currencies $c_{i_1}, \ldots, c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdot \cdots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$. This means that by starting with one unit of currency c_{i_1} and then successively converting it to currencies $c_{i_2}, c_{i_3}, \ldots, c_{i_k}$ and finally back to c_{i_1}, you would end up with more than one unit of currency c_{i_1}. Such anomalies last only a fraction of a minute on the currency exchange, but they provide an opportunity for profit.

We say that a set of exchange rates is arbitrage-free when there is no such sequence, i.e. it is not possible to profit by a series of exchanges.

(a) Give an efficient algorithm for the following problem: given a set of exchange rates $(r_{i,j})_{i,j \in n}$ which is *arbitrage-free*, and two specific currencies a, b, find the most profitable sequence of currency exchanges for converting currency a into currency b. That is, if you have a fixed amount of currency a, output a sequence of exchanges that gets you the maximum amount of currency b.

Hint 1: represent the currencies and rates by a graph whose edge weights are real numbers.

Hint 2: $\log(xy) = \log(x) + \log(y)$

Solution:

Algorithm: The idea is to construct a directed graph were there is a node for every currency $c \in C$. For any two nodes we can represent the edge as the negated logarithm of the exchange rate. The reason to this is that it allows us to turn this problem into a shortest path problem, since we avoid having to multiply the edge weights (we can use log property and add) and we can use Bellman-Fords algorithm to find the shortest path with the negative edges.

Proof of Correctness: The problem is reduced from finding the path with the max path or longest path to finding the min path or shortest path. In other words we go from trying to find the max product of $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdot \cdots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1}$ to finding the min sum of $-\log(r_{i_1,i_2}) - \log(r_{i_2,i_3}) - \cdots - \log(r_{i_{k-1},i_k}) - \log(r_{i_k,i_1})$. The rest of the algorithm relies on Bellman-Ford for which I omit the proof.

Runtime Analysis: Since the algorithm is reliant on Bellman-Ford, the runtime is $O(|V| \cdot |E|)$ where $|V| = |C|$ and $|E|$ is the number exchange rates.

(b) Oski is fed up of manually checking exchange rates, and has asked you for help to write a computer program to do his job for him. Give an efficient algorithm for detecting the possibility of arbitrage.
Solution: Algorithm: In order to detect arbitrage we can use the algorithm from the part (a). The only difference being that we run it one extra time for a total of $|V|$ times. If there is arbitrage, we expect to see the minimum distance from a to b continue to decrease, otherwise there is no arbitrage.
Proof of Correctness: Proof of correctness follows from part (a).
Runtime Analysis: $O(|V| \cdot |E|)$ to run Bellman-Ford algorithm.

3 Not So Crazy Delivery

PNPenguins are running a new food delivery business where they wish to deliver ice-cakes to all their fellow penguins in the shortest amount of time. The penguins community is a network consisting of m households total and r two-way/bidirectional roads. The time it takes to travel between two households h_1,h_2 is given by $c(h_1, h_2)$. Due to limited budget, PNPenguins can only afford to have a maximum of k delivery centers within the community. PNPenguins have found a subset of n households ($m \gg n > k$) within the community who are willing to sell their home to PNPenguins and become a delivery center. For this problem, assume that households always get ice-cakes delivered by the closest delivery center to them.

a) Design an algorithm that helps PNPenguins to find the most optimal locations of delivery centers; that is, a set of delivery centers that minimizes the maximum delivery time needed to all households. This algorithm should run in time polynomial in m and r.

Please provide a 3-part solution.
Hint: First write an algorithm which computes the maximum delivery time for some fixed set of delivery centers. Then, use this algorithm to check the maximum delivery time for each possible set of delivery centers.
Solution:

Algorithm: We can begin by running Dijkstra's algorithm on n candidate households. We keep track of the longest path/max time for every candidate household and store this information in an array. After calculating the maximum delivery time for each candidate household, we sort the candidates based on their maximum delivery time, from shortest time to longest time using the information we stored from running Dijkstra's algorithm. Then take the first k candidates from this sorted list
Proof of Correctness: This problem is solved using a greedy algorithm where we select the next best candidate with the shortest-max path. Let G be the set of k households returned from the algorithm above after sorting by maximum delivery time, and let O be a solution that yields a shorter maximum delivery time. Since $O \neq G$, there must be at least one candidate household h_o in O that is not in G, and let h_g be the corresponding candidate household in G. Now, exchange h_o with h_g. Since $h_o < h_g$, the maximum delivery time will either decrease or stay the same in G, contradicting that O is the best solution.
Runtime Analysis: The initial run of Dijkstra's would yield the runtime of $O(n(m+r)\log m)$ and sorting the candidates would be $O(n \log n)$. The final iteration to retrieve the first k candidates would be $O(k)$. Making this algorithm $O(nm \log n + nr \log m + n \log n + k) = O(n(m+r)\log m)$

PNPenguins have now decided to use the locations you chose from part a). But right before they start the delivery service, Pesla (a highly innovative iceship company headquartered in PNP-Borderland) releases a new magical iceship model that allows passengers to skip over the t most time-consuming roads along any of their journeys. PNPenguins all agree to integrate this new technology into their delivery business.

b) Design an algorithm that finds the updated/new maximum delivery time for each household.
Please provide a 3-part solution.
Hint: create a graph such that edges and nodes encode information about whether an edge is skipped when traversed by a graph algorithm.
Solution:

Algorithm: We can use a priority queue to find the t max weighted edges. In a boolean array, for every edge in the graph we mark the t edges as true if we are going to skip that edge. We can then run the algorithm from part (a) and if the edge is false, we can update the distance array from part (a) by adding 0 for that edge, so that our max paths are updated correctly. We then proceed with the algorithm from above.
Proof of Correctness: The proof of correctness follows from part (a) since changing the edge weight to 0 for the t edges only changes the total distance of a path; however, the algorithm stays the same.
Runtime Analysis: Runtime would be same as the part (a), $O(n(m+r)\log m)$.

4 Three-Legged Race

You are a teacher for two classes, each of n students. You are organizing a three-legged race, where students are paired with students in the opposing class. Each student must be paired up, and each student will only be paired with one other student. In an ideal three-legged race, you and your partner are evenly matched in terms of stride. Luckily, you have data on the stride lengths of all your students. Design an algorithm to minimize the total difference in stride length between pairs.

Please provide a 3-part solution.
Solution:

Algorithm: We sort the data on the length of all the students strides in ascending order (from shortest stride to longest stride) then we pair up student A_i from one class, with student B_i from the opposite class such that the difference between the stride of student A and student B is minimal.
Proof of Correctness: Let G be the greedy solution and let O be some optimal solution such that $O \neq G$. Since $O \neq G$, there must be at least one pair (A_{o_i}, B_{o_i}) in O that differs from the corresponding pair in G. Without loss of generality, assume $(A_{o_1}, B_{o_1}) \neq (A_{g_1}, B_{g_1})$ and $|A_{o_1} - B_{o_1}| < |A_{g_1} - B_{g_1}|$. Now, in G, since we sorted the students' strides in ascending order, $|A_{g_1} - B_{g_1}| \leq |A_{g_i} - B_{g_i}|$ for all i. Therefore, we can perform a swap such that G becomes $(A_{o_1}, B_{o_1}), (A_{g_2}, B_{g_2}), \ldots, (A_{g_n}, B_{g_n})$. But this implies that the total difference in stride length in G is no greater than in O, contradicting the assumption that O is optimal. Hence, G must be the optimal solution.
Runtime Analysis: $O(n \log n)$ from sorting the data.

5 Cars vs. Bikes

PNPenguin is an avid cyclist, and has traveled to Penguinland to go on an epic bike ride. Unfortunately for PNPenguin, however, car traffic in Penguinland makes it dangerous to ride on certain roads.

More specifically, Penguinland consists of N buildings and M unidirectional roads each connecting two buildings, with each road having a nonnegative length l_i. One of the buildings is Penguincorp, a large local company, and all the other $N-1$ buildings are houses of Penguinland residents, all of whom work at Penguincorp. Every morning, each resident of Penguinland will drive from their house to Penguincorp on the shortest possible route. Each resident has a unique shortest route.

To avoid being hit by any cars on his bike ride, PNPenguin wants to find which roads definitely won't be traveled by any Penguinland drivers on their morning commutes. Write an efficient algorithm to find this list of roads in $O((N+M)\log(N))$.

Please provide a 3-part solution.
Solution:

Algorithm: The idea for this problem is to construct a graph with N nodes and M directed edges. We then reverse all the edges and run Dijkstra's algorithm with the starting vertex being Penguincorp. We keep track of the edges we visit to get to get every other building. We can then check which edges were not used.
Proof of Correctness: The reason for reversing the graph is so that we can find the shortest path from Penguincorp to every other building avoiding having to run Dijkstra's from every building to find the shortest path to Penguincorp. At the end of the algorithm above we are left with the edges that are not traversed by the commuters, giving us our answer. The proof Dijkstra's is given in the textbook.
Runtime Analysis: The runtime is that of Dijkstra's which is $O((N+M)\log(N))$.