Problem Set 4

All parts are due on April 25, 2017 at 11:59PM Please download the .zip archive for this problem set. Remember, your goal is to communicate. Full credit will be given only to a correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Last, but not least, take a look at the collaboration policy outlined in the handout for this course.

Part A

Problem 4-1. [20 points] **Largest Strongly Connected Component**

Given a directed graph $G = (V, E)$, a *Strongly Connected Component (SCC)* is a subset of vertices $C \subseteq V$ such that there exists a directed path in G from any vertex $u \in C$ to any other vertex $v \in C$. The goal of this problem is to add a single directed edge to the graph G which maximizes the size of the largest SCC in the graph.

We consider this problem over a restricted class of graphs. More precisely, the initial graph is a DAG (directed acyclic graph), and moreover for any two vertices u and v, there is at most one *simple* directed path from u to v (note that a path is simple only if it does not visit a vertex more than once). Give an algorithm which in $O(|V| + |E|)$ finds the edge that should be added to G in order to maximize the size of the largest SCC in the graph.

Problem 4-2. [20 points] **Zootopia**

Judy has just arrived at the Zootopia train station. She rents a car and wants to get from the train station to the police department. There are n junctions in Zootopia connected by m bidirectional edges (which is either a street, or a bridge, etc). We know that it is possible to get from any junction to any other junction. For each edge $e_i = (u_i, v_i)$, she knows w_i, the number of minutes it takes to go from u_i to v_i. Moreover, at every vertex v, there is a traffic light that is red for $r_v > 0$ minutes and then green for $g_v > 0$ minutes starting at time 0. We know that the cycle length is the same for all traffic lights (i.e., $g_v + r_v = T$) and therefore all traffic lights become red again in time T.

Moreover, while driving across edge e_i, her car uses d_i gallons of fuel per minute, and while stopping at vertex v, her car uses s_v gallons of fuel per minute.

You are given all the information about Zootopia, that is all the vertices along with the *integer* values g_v, r_v and s_v, and the structure of all the edges along with the *integer* values w_i and d_i. Give an algorithm that, in time $O(mT + nT\log(nT))$, finds the minimum amount of fuel her car needs to get from the train station (vertex 1) to the police department (vertex n).

Problem 4-3. [15 points] **Computer Networks**

There is a network of n computers that are internally connected. More precisely there are m Fibre Channels and the ith channel connects computers u_i and v_i. We know that the ith channel might burn out with probability p_i.

(a) [5 points] Give an algorithm that in $O(n^3)$ for any pair of computers u and v, finds a path $P_{u,v}$ from u to v that minimizes the maximum probability of failure (p_e) of any edge e on the path $P_{u,v}$.

(b) [10 points] Give an algorithm that in $O(n^3)$ for any pair of computers u and v, finds a path $P_{u,v}$ from u to v that minimizes the failure probability of the path $P_{u,v}$, assuming the probability of failure of any two edges are independent. Note that a path fails if any of its edges fails.

Problem 4-4. [20 points] **Promotion**

There is a company with n employees (including the CEO) that are all equally being paid 2 thousand dollars a month. The CEO of the company decides to increase his own salary from 2 thousand to t thousand dollars a month. At the end of each month every employee of the company looks at the salary of all its neighbors and based on how he/she evaluates the neighbors' performance, the employee might ask for a promotion (which is always granted). More precisely, we are given a graph G with n vertices and m directed edges. Each edge from u to v has a weight w (not necessarily positive). If v sees that u earns s dollars per month, then v expects to receive a salary of $s + w$ dollars per month from the company. Finally, v will ask for a raise equal to the maximum expectation he/she from all the neighbors (if it is more that his current salary). The CEO would like to know what is the total salary the company needs to pay to its employees *after one year*. Give an algorithm that answers this question in time $O(m + n)$ and analyze its running time and correctness.

Part B

Problem 4-5. [25 points] **Labyrinth**

You are exploring an underground labyrinth consisting of rooms connected by corridors. The layout of the labyrinth is known completely to you. However, each corridor is filled with one of four types of poisonous gas. Fortunately, you have a gas mask that can provide immunity from each type of gas, provided it is fitted with a filter specific to that type of gas. The four filters (one of each type) are each in a given room in the labyrinth. Your goal is to get to a specific room in the labyrinth as fast as possible. During each timestep, you may pick up any number of filters located in the same room as you and then traverse at most one corridor (provided you possess the proper filter). Assume that attach or deattaching filters from the gas mask may be done instantaneously and as often as you like.

(a) [10 points] Implement `explore_single` in `labyrinth.py`.

(b) [15 points] Now, you are given a specific time limit, assumed to be a small constant (less than 20). Your goal is to determine the minimum number of explorers required to reach the goal room within the time limit. (Only one explorer needs to reach the goal room.) In addition to the above rules, filters can be dropped for other players to pick up from the same room. Dropping one or multiple filters may be done at the beginning of a timestep, so that an explorer may drop a filter and explore a corridor in the same timestep. However, another explorer may not pick up a filter dropped in the same timestep.

Implement `explore_multiple`. Your solution does not have to efficient; a solution exponential in the number of explorers is acceptable.

See the provided code for details and examples. Submit your finished `labyrinth.py` to `alg.csail.mit.edu`. Your code will be run as `python3`.