15-451/651 Algorithms, Spring 2019

Homework #5 **Due: March 26, 2019**

This HW has three regular problems and a programming assignment. All problems on the HWs are to be done *individually*, no collaboration is allowed. Solutions to the written problems should be submitted as a single PDF file using `gradescope`, with the answer to each problem starting on a new page.

(25 pts) 1. **(The Gates Elevators.)** Ever wonder whether you should wait for the elevators, or just take the stairs? If wait, how long before you give up? This is the ideal problem for you, the 451 student!!

We model this process as a zero-sum game. Taking the steps takes S units of time. The elevator takes E units of time. The elevator arrives at time 1 or 2 or 3, etc. If the elevator arrives at time $t \geq 1$ and you take it, you reach your destination at time $t + E$. However, if you have waited $w \geq 0$ units of time, and if the elevator has not yet arrived, you may decide to walk up, which takes S units of time, so you reach at time $w + S$. (Once you decide to start walking you cannot catch the elevator any more.)

Your actions are *"wait w time; if no elevator yet, walk up"*: one action for every integer $w \geq 0$. (So $w = 0$ means you don't wait at all.) The adversary's actions are: *"elevator arrives at time t"*, one for every positive integer $t \geq 1$.

> For instance if $w = 2$ and $t \leq 2$ then you catch the elevator and arrive at time $t + E$, but if $w = 2, t = 3$ then you wait 2 time steps, (leave just before the elevator comes) and reaching at time $2 + S$.

Given the pair of actions (t, w), your pain (aka. the payoff to the adversary) is the ratio: (the time it took you) divided by (the optimum time it would take you if you knew when the elevator were to arrive).

(a) Let R be the matrix of payoffs to the adversary whose rows t are the adversary's actions and columns w are your actions, then for any $t \geq 1$ and $w \geq 0$, write down the mathematical expression for the payoff to the adversary.

For the rest of the parts, assume that $E = 1$ and $S = 3$.

(b) This game has an infinite number of columns and an infinite number of rows. Let's add one more row, called "row ∞" for the scenario that the elevator never arrives. Argue that without loss of generality, we can assume the adversary chooses only rows $t = 1$ or $t = \infty$. Formally, argue that for any $t \geq 2$ we have $R_{tw} \leq R_{\infty w}$ for all t. This means that your strategy achieving expected payoff V in a world with only those two scenarios possible (elevator arrives at time $t = 1$, or it never arrives) will achieve also payoff V over the whole range of scenarios.

(c) Now that the game has just two rows, argue that the minimax-optimal strategy can safely put probability 0 on all columns except for $w \in \{0, 1\}$. Formally, argue that for any $w > 1$ we have $R_{tw} \geq R_{t1}$ for $t \in \{1, \infty\}$.

(d) Now write down the 2-by-2 matrix that results. Write down the LP for the minimax-optimal strategy for you, the column player, and solve it to find the value of the game.

(e) Finally, suppose there is an elevator operator (row player) who controls the timing of the elevator, and is trying to cause you the most pain (get the highest payoff for herself). Solve for the row player's strategy.

(25 pts) 2. **What are Widgets, Anyways?**

You are in charge of the supply chain for Widgets-r-Us, where you have to find ways to transport materials from their storage warehouses to the factories, subject to capacity constraints on the transportation network. You model it as a directed graph $G = (V, E)$, where each directed edge $e \in E$ represents a one-way road. (Roads capable of carrying traffic in both directions can be thought of as two one-way roads.)

There are k "warehouse" vertices, and k kinds of materials, where the i^{th} material is originally located at the warehouse vertex $s_i \in V$, and the total amount of material i is D_i. Each road (directed edge) e has a capacity u_e, such that the total amount of material (of all kinds) sent over this road must be at most u_e. You may assume that all quantities given as input in this problem are non-negative integers.

You also have ℓ "factory" vertices. There are ℓ factories producing widgets (the j^{th} factory is located at the j^{th} factory vertex $f_j \in V$). The factory f_j has a request vector $\mathbf{r}_j = (r_{1j}, r_{2j}, \ldots, r_{kj})$, such that to produce one unit of widget, it requires r_{ij} amounts of material i for every $i \in \{1, \ldots, k\}$. [1]

(a) Write an LP to figure out how to transport the material from their warehouses to the factories (respecting the road capacity constraints), to maximize the total amount of widgets produced, subject to these constraints. (It is OK if you produce fractional amounts of widgets.)

(b) You find out that widgets produced at different factories sell for different amounts of money: the price per unit of widget produced at factory j is p_j. You want to maximize your revenue. Change the LP from the previous part to handle this.

(c) You are informed that two of the roads $e_1 = (u, v)$ and $e_2 = (v, u)$ are special: they represent the two directions of traffic over a bridge. For structural reasons, you have the balance requirement that the absolute value of the difference between the amount of material send in the two directions can be at most $\delta \in \mathbb{Z}_{\geq 0}$. How can you add such a constraint to your LP.

(25 pts) 3. **(Circuit Board Wiring)**

There are n terminals on a circuit board. Each terminal is either positive or negative. A wire will be attached to each terminal. It must be the case that for every pair of terminals of opposite polarity the two wires must be long enough so that they can be made to touch.

[1] Nodes like s_i and f_j can have both incoming and outgoing arcs, and the route taking material i from the warehouse s_i to some factory f_j is allowed to pass through some other warehouse vertex $s_{i'}$ or some other factory vertex $f_{j'}$. You may assume that all the $k + \ell$ warehouse and factory vertices are distinct.

> E.g., if there's a positive terminal at point $a = (0, 0)$, and a negative terminal at $b = (0, 10)$, and another negative terminal at $c = (10, 0)$ then one solution is to put a wires of length 5 at all three terminals. Or to put a wire of length 10 at a and length zero at the others. Or, for any $x \in [0, 10]$, wires of length $\geq 10 - x$ at a, and length $\geq x$ at both b and c.

So given the locations of the terminals (and their polarities), the problem is to compute the shortest total length of wires needed to satisfy the stated requirement.

(a) Consider the following specific instance of the problem. All the terminals are along the x-axis. There are positive terminals at 0, 110, 111, 112. And there are negative terminals at 99 and 100. What's the optimal solution to the problem? (You don't need to prove your solution is optimal.)

(b) Formulate this problem as an LP. (Illustrate your LP by writing this for an example with two positive and two negative terminals.)

(c) Write the dual of the LP in part (b). (Again, write down this dual for the two-terminal example in part (b).)

(d) Show how to compute an optimal solution to the dual problem of part (c), without using a general LP solver.

(25 pts) 4. **Travelling Salesperson Problem**

In this problem you will implement an algorithm that can solve the Travelling Salesperson Problem (TSP). We discussed this problem in Lecture #9 and an algorithm based on subset dynamic programming. The input to your program will be a connected graph G with n vertices and m edges. Each edge $\{u, v\}$ has a non-negative length in each direction, and $\text{len}(u, v)$ may be different from $\text{len}(v, u)$. Your program will compute the length of the shortest TSP tour for the given graph, and also output the tour. The time limit is 10 seconds. The filename is `tsp.c`, or analogues thereof. Note you do not need to provide a written explanation of your algorithm.

Input: The first line consists of two positive integers n and m. $n \leq 20$ and $n - 1 \leq m \leq n(n-1)/2$. The following m lines each contain four integers that represent an edge. These numbers are $i\ j\ d_{i,j}\ d_{j,i}$. This means that there is an edge $\{i, j\}$ in the graph and the directed length from i to j is $d_{i,j}$ and the directed length from j to i is $d_{j,i}$. Each edge will occur at most once in the list in either direction. There are no self loops, and the graph is connected. The vertex numbers are in $[0, n-1]$, and the lengths of the edges are in $[0, 10^6]$.

Output: The first line of output contains the cost of the shortest travelling salesperson tour for the given graph. The second line contains list of the cities in the order visited, starting at city 0 and ending at city 0. Each pair of neighboring cities on this list must be an edge in G. Any tour which is of minimum cost is acceptable.

Samples: For example, if the input is:

```
5 5
0 1 1 1
1 2 1 1
2 3 1 2
3 4 1 1
4 0 1 1
```

Then the output (which is unique in this case) will be:

```
optimal tour cost = 5
tour: 0 1 2 3 4 0
```

For example, if the input is:

```
6 6
0 1 1 1
1 2 1 1
2 3 1 1
3 1 1 2
2 4 1 1
3 5 1 1
```

Then the output might be:

```
optimal tour cost = 9
tour: 0 1 2 4 2 3 5 3 1 0
```