University of Toronto at Scarborough
CSCC73 - Algorithm Design and Analysis

Assignment #4: Dynamic Programming

Due: October 24, 2018 at 11.59pm This exercise is worth 5% of your final grade.

Warning: Your electronic submission on MarkUs affirms that this exercise is your own work and no one else's, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters, the Code of Student Conduct, and the guidelines for avoiding plagiarism in CSCC73. Late assignments will not be accepted. If you are working with a partner your partners' name must be listed on your assignment and you must sign up as a "group" on MarkUs. Recall you must not consult **any outside sources except your partner, textbook, TAs and instructor**.

Note. Some of these questions may be easier once we have discussed the Bellman Ford algorithm.

1. (5 marks) You have decided to use your new algorithmic knowledge and try to make some money off your management friends who have brokerage jobs for the summer. You have discovered that a broker that trades shares in n different companies may take advantage of *opportunities* that arise. Suppose that for each pair of companies $i \neq j$, the broker maintains a trade ratio t_{ij} that represents the number of shares of j that one share of i trades for - this rate can be a fraction. For example, if $t_{ij} = \frac{3}{4}$ then you can trade 4 shares of i for 3 shares of j.

   You realize that that if there were a way to trade shares successively so that one ends up with more shares of the first company than they started with, there would be an opportunity to profit. You tell your friends in management that for a percentage of their profits, you will write an algorithm that enables them to recognize such opportunities.

   We will define such a *trading cycle*; let $i_1, i_2, \ldots, i_m$ be such a sequence where shares in company i_1 are traded for shares in company i_2 and so on until finally trading shares in company i_m to shares in company i_1. If trading around such a cycle results in increased shares in company i_1 then we say that this is a *profitable cycle*. Notice that this happens exactly when the product of the ratios along the cycle is greater than 1. Give a polynomial time algorithm that finds such a profitable cycle if it exists. You do not need to prove your algorithm correct but you must justify it.

2. (10 marks) To assess how "well-connected" two nodes in a directed graph are, one can not only look at the length of the shortest path between them, but can also count the *number* of shortest paths.

   This turns out to be a problem that can be solved efficiently, subject to some restrictions on the edge costs. Suppose we are given a directed graph $G = (V, E)$, with costs on the edges; the costs may be positive or negative, but every cycle in the graph has strictly positive cost. We are also given two nodes $v, w \in V$. Give an efficient algorithm that computes the number of shortest $v - w$ paths in G. (The algorithm should not list all the paths; just the number suffices.) You do not need to give a formal proof of correctness but should justify your recurrence relation and the complexity.

3. (10 marks) With all your school work you're running out of time to complete simple tasks like laundry. Since you are a poor student, you realize that if you're going to outsource your laundry there are only two ways you can afford to do this. Either you ask your mom - who immediately tells you to "grow up" or you get together with friends to organize a group laundry service. You poll all your friends to determine who wants in on which weeks and how many loads they each have. From this you have a weekly schedule for n weeks where each value is a number of loads ℓ_i for week i. You research the two closest laundromats that offer pick-up and delivery. Laundromat L_1 and laundromat L_2 have different methods for charging for their services.

- L_1 charges a fixed rate r per load (so it costs $r \cdot \ell_i$ to pick-up, wash and deliver a week's load ℓ_i.)
- L_2 makes contracts for a fixed amount w per week, independent of the number of loads. However, contracts with company L_2, must be made in blocks of three consecutive weeks at a time.

A *schedule*, for you and your friends laundry, is a choice of laundromat (L_1 or L_2) for each of the n weeks, with the restriction that laundromat L_2, whenever it is chosen, must be chosen for blocks of three contiguous weeks at a time. The *cost* of the schedule is the total amount paid to laundromat L_1 and L_2, according to the description above.

Give a polynomial-time algorithm that (makes you the laundry hero and) takes a sequence of weekly load values $\ell_1, \ell_2, \ldots \ell_n$ and returns a *schedule* of minimum cost. Explain your algorithm and justify your recurrence relation and complexity. A formal proof is not necessary.

Example Suppose $r = \$10$, $w = \$80$, and the sequence of values is

$$5, 8, 10, 11, 9, 6, 7.$$

Then the optimal schedule would be to choose laundromat L_1 for the first two weeks, then laundromat L_2 for a block of three consecutive weeks and then laundromat L_1 for the final two weeks.