CS-473/573: Algorithms I Spring 2018

Assignment 5: Dynamic Programming

Instructor: Mehmet Koyutürk

Due Date and Instructions

Please return hard-copies at the beginning of the class meeting (8:40) on **Friday, April 20, 2018**.

- Hand-writing is accepted but the course personnel reserves the right to reject grading an assignment if the hand-writing is not legible.
- Assignments written in LaTeX will receive 5 bonus points.
- You can submit the assignment as a team of 2 students, however no pair of students are allowed to work together on more than 3 assignments.
- You are also allowed to talk to other students and the course personnel about the solutions, but you must write the answers yourself. Answers copied from other students or resources will be detected, and appropriate action will be taken.

Problem 1

We are given n currencies and an exchange rate r_{ij} for any pair of currencies i and j. Namely, if we exchange 1 unit of currency i with currency j, we receive r_{ij} units of currency j. If we are given a source currency s and a target currency t, then we can go through a path of different currencies to reach t from s so as to maximize our profit. The markets can also charge an exchange fee depending on the number of exchanges we make. For example if the exchange fee is $f(k)$ for making k exchanges and we start with 1 unit of currency s, then the path of exchanges $s \to t$ will yield $r_{st} - f(1)$ units of currency t, whereas the path of exchanges $s \to i \to j \to t$ will yield $r_{si} \times r_{ij} \times r_{jt} - f(3)$ units of currency t. The problem is to find the sequence of exchanges that will maximize the amount of target currency t we can obtain for a given source currency s.

(a) [10 pts.] Define and prove the optimal substructure property for this problem when there is no exchange fee ($f(k) = 0$ for all k).

(b) [10 pts.] Show that it is possible to find an exchange fee schedule $f(k)$ so that the optimal substructure you defined above will not hold anymore.

Problem 2

Provide a dynamic programming solution to each of the following problems by following the following steps:

(i) Identify the "last" decision you need to make in developing a solution.

(iii) Define subproblems, express the solution to the overall problem in terms of the subproblems.

(iv) Formulate a recursive solution to the subproblems. Do not forget to specify the base case(s).

(vi) Characterize the runtime of the resulting procedure (assuming that you would implement your solution using a bottom-up procedure).

In addition, for one of the problems (of your choice, you can choose a different problem for each item below):

(ii) [10 pts.] Define and prove optimal substructure.

(v) [10 pts.] Provide the pseudo code of the bottom-up procedure you use to compute the value of the optimal solution, as well as the procedure for reconstructing the optimal solution.

The problems are the following:

(a) [20 pts.] We are given an arithmetic expression $x_1 o_1 x_2 o_2 ... x_{n-1} o_{n-1} x_n$ such that x_i for $1 \leq i \leq n$ are positive numbers and $o_i \in \{+, \times\}$ for $1 \leq i \leq n-1$ are arithmetic operations (summation or multiplication). We would like to parenthesize the expression in a such a way that the value of the expression is maximized. For example, if the expression is $3+4\times2+6\times0.5$, then the optimal parenthesization is $(3+4) \times (2 + (6 \times 0.5))$, with a value of 35.

(b) [20 pts] We are given n types of coin denominations with integer values v_1, v_2, ..., v_n. Given an integer t, we would like to compute the minimum number of coins to make change for t (i.e., we would like to compute the minimum number of coins that add up to t, where repetitions are allowed). We know that one of the coins has value 1, so we can always make change for any amount of money t. For example, if we have coin denominations of 1, 2, and 5, then the optimal solution for $t = 9$ is 5, 2, 2.

(c) [20 pts] Given two strings $X =< x_1, x_2, ..., x_m >$ and $Y =< y_1, y_2, ..., y_n >$, the edit distance between X and Y is defined as the minimum number of edit operations (*replacement*, *insertion*, or *deletion* of a character) required to convert X to Y. For example, the edit distance between $X = esteban$ and $Y = stephen$ is 4, comprising of 1 deletion (e), 1 insertion (h), and 2 replacements ($b \rightarrow p$ and $a \rightarrow e$). We would like to compute the edit distance between two given strings.