CS 477/677 Analysis of Algorithms

Homework 5

Due April 14, 2014

For the programming problem below, include in your hardcopy submission a printout of your algorithm and of the output. Also send your source code as an attachment to cs477677@cse.unr.edu. The subject line of the email should be HW3 - <Your section (CS477 or CS677> - <Your Name>.

1. (U & G-required) [100 points]

Consider that you are the manager of a consulting team of expert computer programmers, and each week you have to select a job for them to work on. The jobs are of two categories: either *low-stress* (e.g. setting up a Web site for a small fundraising event), or *high-stress* (e.g., protecting a company's valuable patents). Each week, the main question with what type of job to take on: low-stress or high-stress.

For a particular week *i*, choosing a low-stress job will earn you a revenue of $l_i > 0$ dollars, while for a high-stress job, you get a revenue of $h_i > 0$ dollars (high-stress jobs typically pay more). However, if the team works on a high-stress job in week *i*, they cannot do any job (of either type) in the previous week *i-1* (they need that previous week to prepare for the high stress level). If they work on a low-stress job in week i, they can work on any job (of either type) in the previous week *i-1*.

A *plan* for the team, is specified as a choice of *low-stress*, *high-stress* or *none*, for a sequence of *n* given weeks (with the constraint that if *high-stress* is selected for week *i* > 1, then *none* must be chosen for week *i-1*. (It is permitted to choose a high-stress job in week 1.) The revenue of the plan is computed as follows: for each week *i*, add l_i to the total if choosing *low-stress* in week i, and add h_i to the total if choosing *high-stress* in week *i* (add 0 if choosing *none* in week *i*.)

The goal of the problem is that given a set of values $l_1, l_2, \dots, l_n$ and $h_1, h_2, \dots, h_n$ to find a plan of maximum value. Develop a dynamic programming algorithm that finds the value of an optimal plan using the steps outlined above.

(a) [20 points] Determine and **prove** the optimal substructure of the problem and write a recursive formula of an optimal solution (i.e., define the variable that you wish to optimize and explain how a solution to computing it can be obtained from solutions to subproblems). **Submit**: the recursive formula, along with definitions and explanations on what is computed.

(b) [30 points] Write an algorithm that computes an optimal solution to this problem, based on the recurrence above. Implement your algorithm in C/C++ and run it on the following values:

	Week 1	Week 2	Week 3	Week 4
l	10	1	10	10
h	5	50	5	1

Submit:

- A printed version of the algorithm
- A printout of the table that contains the solutions to the subproblems, run on the values given above (print the entire table!)

(c) [20 points] Update the algorithm you developed at point (b) to enable the reconstruction of the optimal solution, i.e., which jobs were selected in an optimal solution for the sequence of 4 weeks. (Hint: use an auxiliary table like we did in the examples in class.) Include these updates in your algorithm implementation from point (b).

Submit:

- A printed version of the algorithm
- A printout of the values that you obtain in the table containing the additional information needed to reconstruct the optimal solution, run on the values given above (print the entire table!)

(d) [30 points] Using the additional information computed at point (c), write an algorithm that outputs which jobs have been selected in every week. Implement this algorithm in C/C+.

Submit:

- A printed version of the algorithm
- A printout of the **solution** to the problem given by the numerical values in point (b).

2. **(G-required)** [20 points] Show how the algorithm MATRIX-CHAIN-ORDER discussed in class computes the number of scalar multiplications for the product of the following three matrices (i.e., give the values in table "**m**" as computed by the algorithm):

A: size 4x3

B: size 3x5

C: size 5x2

Extra Credit

3. [20 points] Indicate whether the following statements are true or false and justify your answers.

(a) If X and Y are sequences that both begin with the character A, every longest common subsequence of X and Y begins with A.

(b) If X and Y are sequences that both end with the character A, some longest common subsequence of X and Y ends with A.