CS 477/677 Analysis of Algorithms

Homework 6

Due April 19, 2024

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

Suppose that you have a party organization business hired to set up the flower arrangement for a big event to be held in an outdoor garden. The garden has a fixed set tables where the flowers can be arranged, with only one flower bouquet allowed on each table. The tables are arranged along one straight side of the garden, at specific locations $x_1, x_2, \ldots, x_n$, given in meters. The leftmost point of the side of the garden has location *0* and rightmost point has location L (also in meters). The flower shop company from where you get the flowers will give you a bonus b_i ($b_i > 0$), in dollars, if you place their bouquet on table at a location x_i (bonus values are different, as some locations are more desirable than others). However, to allow space for other displays, the event organizers do not allow you to place flowers on tables that are within *less than* or *equal* to 5 meters of each other. Your goal is to place flower bouquets at a subset of possible tables in such a way that you maximize your total bonus from the flower shop. More specifically, given a sequence of n locations, a *flower arrangement plan* for the event is a subset of locations where you place the flowers. The *total bonus* of the plan is the sum of all bonus values obtained by placing flowers on the respective tables. The goal of the problem is that: given a value for the length of the garden L, a sequence of n table locations $x_1, x_2, \ldots, x_n$ and n corresponding bonus values $b_1, b_2, \ldots, b_n$ to find a *flower arrangement plan* that maximizes your *total bonus*. Develop a dynamic programming algorithm that finds the value and solution for an optimal plan using the steps outlined below.

(a) [20 points] Write a recursive formula for computing the optimal value for the *total bonus* (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 (in a PDF file).

(b) [30 points] Write an algorithm that computes an optimal solution to this problem, based on the recurrence above. The algorithm should save in an output file the optimal values for all the subproblems as well as the optimal value for the entire problem. Implement your algorithm in C/C++ and run it on the following values:

$L = 30$

$n = 5$

	Table 1	Table 2	Table 3	Table 4	Table 5
Locations x_i	8	10	15	22	26
Bonus b_i	\$15	\$5	\$25	\$15	\$5

Submit:

- The source file containing your algorithm (name the file **`bonus_pb.c`** or **`bonus_pb.cpp`**)
- The output file created by your algorithm (name the file **`bonus_pb_out.txt`**), which contains:
  - The table with the optimal values to all subproblems (save the entire table)
  - The optimal value for the entire problem (indicate this on a separate line after the table, even if the value is found in the table above)

(c) [20 points] Update the algorithm you developed at point (b) to enable the reconstruction of the optimal solution, i.e., to **store the choices** you made when computing the optimal values for each subproblem in part (b). Your updated C/C++ program should store those choices in an auxiliary table and then save that table in an output file. Include these updates in your implementation from point (b).

Submit:

- The source file containing your algorithm (name the file **`bonus_pc.c`** or **`bonus_pc.cpp`**)
- The output file created by your algorithm (name the file **`bonus_pc_out.txt`**), which contains the values of the table containing the additional information (choices) needed to reconstruct the optimal solution (print the entire table).

(d) [30 points] Using the additional information computed at point (c), write an algorithm that prints the optimal solution, i.e., it prints the IDs of the tables (IDs are from 1 to 5 for the 5 tables) that have been used in the flower arrangement. **Important notes**: 1) this should be a stand-alone algorithm, separate from the ones in parts b and c; 2) the algorithm should read in the choices from the **`bonus_pc_out.txt`** file saved in part c). Implement this algorithm in C/C+.

Submit:

- The source file containing your algorithm (name the file **`bonus_pd.c`** or **`bonus_pd.cpp`**)
- The output file created by your algorithm (name the file **`bonus_pd_out.txt`**) that contains the optimal solution to the problem given by the numerical values in part (b).

2. **(G-required)** [20 points] Show how the algorithm for finding the longest common subsequence (discussed in class) operates on the following two strings:

X = < A, R, B, C, D, A, N, C>

Y = < A, B, C, A, C, N, O, C>

Give both the length and the actual solution for the longest common subsequence of X and Y.

Extra Credit

3. [20 points] List all of the different orders in which we can multiply five matrices A, B, C, D, and E.