CS29003 Algorithms Laboratory

Assignment No: 5

Date: 11–February–2020

Dynamic-Programming Algorithms

Foodolls Inc. manufactures n types of dolls. Each doll is manufactured in a machine of a particular type. In order to reduce cost, the CEO of Foodolls buys cheap machines of each type. The machines are rather unreliable, and produce defective dolls with some probabilities. Defective dolls have lower selling price than perfect dolls. In order to keep the doll production profitable, Foodolls CEO plans to buy multiple machines of each type. Because of limited space in Foodolls production factory, only one machine of each type can be operated. That is, if a machine fails to produce perfect dolls, another machine of the same type is substituted. For financial constraints, the CEO plans to make no more than a fixed amount of capital investment for purchasing the machines. He wants to adjust the quantities of the machines of different types such that the total cost of the machines is within the scope of his investment, and his total profit (the total selling price of the dolls) is maximized. The following table lists the parameters. The items (doll types, machine types, and so on) are numbered by $i = 0, 1, 2, \ldots, n-1$.

Item	Meaning	Type
C	Capital investment (Maximum total cost of the purchase)	`int`
c_i	Cost of a single machine of type i	`int`
m_i	Number of copies of machines of type i to be purchased	`int`
s_i	Selling price of perfect dolls of the i-th type	`int/double`
t_i	Selling price of defective dolls of the i-th type	`int/double`
r_i	Maintenance (repair) cost of each machine of the i-th type	`int/double`
p_i	Probability that a machine of the i-th type produces defective dolls	`double`
q_i	Probability that a machine of the i-th type produces defective dolls after maintenance	`double`

There must be one (and only one) running machine of each type even if it produces defective dolls. We also assume that C is large enough so that at least one copy of a machine of each type can be purchased. For the algorithms to work, the capital C and all machine costs c_i must be positive integers. Assume that $t_i < s_i$ and $q_i < p_i$ for all i.

Part 1: Maximizing profit without maintenance

All of the m_i machines of the i-th type produce defective dolls with probability $p_i^{m_i}$, and so the probability that at least one such machine produces perfect dolls is $1 - p_i^{m_i}$. The expected profit from the sale of dolls of the i-th type is therefore

$$e_i = \left(1 - p_i^{m_i}\right) s_i + p_i^{m_i} t_i,$$

and the total expected profit is

$$E = \sum_{i=0}^{n-1} e_i = \sum_{i=0}^{n-1} \left[\left(1 - p_i^{m_i}\right) s_i + p_i^{m_i} t_i\right].$$

Each m_i is a positive integer ($m_i = 0$ is, in particular, not permitted). Your task is to help the Foodolls CEO find values of $m_0, m_1, m_2, \ldots, m_{n-1}$ to maximize the total expected profit E subject to the constraint

$$m_0 c_0 + m_1 c_1 + m_2 c_2 + \cdots + m_{n-1} c_{n-1} \leqslant C.$$

In order to find an optimal solution, prepare an $n \times (C+1)$ table T. The entry $T(i, j)$ is supposed to store the maximum total expected profit from dolls only of types $0, 1, 2, \ldots, i$ if the capital investment is j. The final answer is therefore $T(n-1, C)$.

The top row of T is initialized as follows. For $0 \leqslant j < c_0$, no machine of type 0 can be bought, so $T(0, j) = -\infty$ for these values of j. For $j \geqslant c_0$, the optimal choice is to buy $m_0 = \lfloor j/c_0 \rfloor$ copies of the machines of type 0, and $T(0, j)$ should be set to e_0 (see the above formula for e_i).

For $i \geqslant 1$, the i-th row of T can be populated using the $(i-1)$-th row. For $0 \leqslant j < c_i$, we have $T(i, j) = -\infty$. For $j \geqslant c_i$, you have the choices $1, 2, 3, \ldots, \lfloor j/c_i \rfloor$ for m_i. For each such choice of m_i, you spend $m_i c_i$ from your investment j. The remaining capital $j - m_i c_i$ is used for purchasing machines of type $0, 1, 2, \ldots, i-1$. If that purchase is done optimally, the total expected profit is $T(i-1, j - m_i c_i) + e_i$. Maximize this quantity over all choices of m_i.

In order to find the values of m_i in an optimal solution, you need to remember, in another two-dimensional table, the choice of m_i that maximizes $T(i, j)$.

Write a function $optimalbuy1(n, C, c, s, t, p)$ to implement this dynamic-programming algorithm. This part does not use the arrays r (maintenance costs) and q (probabilities of failure after maintenance). The function should print the optimal values of all m_i, and the actual cost of purchasing these many machines. It should also print/return the maximum total expected profit.

Part 2: Maximizing profit with maintenance

In order to enhance the reliability of the machines, Foodolls can explore the possibility of maintenance of the machines. For the machines of the i-th type, the probability of defective-doll production decreases from p_i to q_i after maintenance. However, for *each* machine of the i-th type, Foodolls has to pay r_i to the machine vendor. It is in the purchase contract that if a maintenance engineer of the i-th type of machine comes, he will work on all of the m_i copies. As a result, if the machines of the i-th type undergo maintenance, the amount $m_i r_i$ is to be deducted from the total profit. The choice is now whether or not to call the maintenance engineer of each type. The goal is again to maximize the total expected profit (after deduction of maintenance charges). The capital investment constraint continues to stay valid.

Write a function $optimalbuy2(n, C, c, s, t, r, p, q)$ to solve the maximization problem introduced in the last paragraph. The function should compute and report the maximum total expected profit, the choices of $m_0, m_1, m_2, \ldots, m_{n-1}$ leading to this profit, and the actual cost of purchase, alongside the decisions on which types of machines need maintenance. As in Part 1, make a dynamic-programming formulation, and store all intermediate results in two-dimensional tables.

The *main*() function

- The user specifies n, and all parameters $C, c_i, s_i, t_i, r_i, p_i, q_i$ at the beginning (see the sample output).
- Call *optimalbuy*1 to solve the problem of Part 1.
- Call *optimalbuy*2 to solve the problem of Part 2.

Submit a single C/C++ source file. Do not use global/static variables.

Sample output

```
n                    : 16
Capital          (C) : 2532
Costs            (c) : 41 55 51 38 83 91 43 96 98 46 95 23 89 29 52 92
Selling price    (s) : 98 70 50 61 94 84 91 81 81 70 82 51 97 76 95 55
Selling price    (t) : 10 35 12 13 36 11 25 30 39 27 17 25 19 26 14 28
Maintenance cost (r) :  5  4  5  4  3  1  3  2  2  2  5  2  4  4  1  5
Probabilities    (p) : 0.9588481828 0.6090085574 0.9949199038 0.5982189628
                       0.8812402696 0.5552817465 0.7384469380 0.6776996251
                       0.8359002484 0.8605754214 0.7293403466 0.7806705960
                       0.6731179239 0.7149133080 0.8381089516 0.8136845610
Probabilities    (q) : 0.7686569828 0.0610634239 0.0545880344 0.2801836246
                       0.8605711086 0.3952110880 0.2891662559 0.3816935232
                       0.7402798372 0.6947915922 0.2011213947 0.3774181376
                       0.5810420500 0.0518749737 0.8351910188 0.3619142445

+++ Part 1: Best buying option
    Machine   0:   1 copies, cost =    41
    Machine   1:   2 copies, cost =   110
    Machine   2:   1 copies, cost =    51
    Machine   3:   4 copies, cost =   152
    Machine   4:   1 copies, cost =    83
    Machine   5:   3 copies, cost =   273
    Machine   6:   5 copies, cost =   215
    Machine   7:   2 copies, cost =   192
    Machine   8:   1 copies, cost =    98
    Machine   9:   2 copies, cost =    92
    Machine  10:   3 copies, cost =   285
    Machine  11:   4 copies, cost =    92
    Machine  12:   3 copies, cost =   267
    Machine  13:   6 copies, cost =   174
    Machine  14:   6 copies, cost =   312
    Machine  15:   1 copies, cost =    92
--- Total buying cost           =  2529
--- Expected total profit       =   810.82540256

+++ Part 2: Best buying option
    Machine   0:   4 copies, cost =   164  [Maintenance needed]
    Machine   1:   1 copies, cost =    55  [Maintenance needed]
    Machine   2:   1 copies, cost =    51  [Maintenance needed]
    Machine   3:   5 copies, cost =   190  [Maintenance not needed]
    Machine   4:   3 copies, cost =   249  [Maintenance not needed]
    Machine   5:   3 copies, cost =   273  [Maintenance needed]
    Machine   6:   2 copies, cost =    86  [Maintenance needed]
    Machine   7:   2 copies, cost =   192  [Maintenance needed]
    Machine   8:   1 copies, cost =    98  [Maintenance needed]
    Machine   9:   3 copies, cost =   138  [Maintenance needed]
    Machine  10:   1 copies, cost =    95  [Maintenance needed]
    Machine  11:   2 copies, cost =    46  [Maintenance needed]
    Machine  12:   4 copies, cost =   356  [Maintenance not needed]
    Machine  13:   1 copies, cost =    29  [Maintenance needed]
    Machine  14:   8 copies, cost =   416  [Maintenance not needed]
    Machine  15:   1 copies, cost =    92  [Maintenance needed]
--- Total buying cost           =  2530
--- Expected total profit       =   961.85622325
```