

CMSC 451: Project, Spring 2006

The project, described in two pages now, is in two parts. The first part is due on April 18th, at the beginning of class. The second part is due on May 9th at the beginning of class. These are firm deadlines.

1 The problem

There are 5 different kind of items a, b, c, d, e selling in a store. To celebrate its anniversary, special offers are given to the customers. If certain items are bought together, they can be bought at a reduced price. For example, we could have the following: the price of a single item a is 5 dollars, the price of a single item b is 3 dollars, but buying 2 a and 2 b together will only cost you 12 dollars.

There is a certain quantity of each of the items that you want to buy, and you want to make use of the special offers to minimize your payment. Sometimes, adding items will reduce the amount you pay. If this happens, you should consider adding items. For instance in the above example, suppose you want to buy 2 a and 1 b . 2 a and 1 b will cost you \$13, but if you add an extra b , you will just need to pay \$12. In other words, you should make use of the special offers, as well as adding items, so that you can minimize your payment while getting everything you want.

The inputs to the problem are:

- (I1) non-negative integers n_1, n_2, n_3, n_4 and n_5 .
- (I2) positive integers c_1, c_2, c_3, c_4 and c_5 .
- (I3) a positive integer s .
- (I4) a matrix M with s rows and 6 columns. The i^{th} row of the matrix contains 6 non-negative integers, $m_{i1}, m_{i2}, m_{i3}, m_{i4}, m_{i5}$ and r_i .

n_1, \dots, n_5 are the number of item a, \dots, e that you want to buy respectively. c_1, \dots, c_5 are the price of a single item a, \dots, e respectively. s is the number of special offers; special offers are also sometimes simply called “offers” below. The i^{th} row of the matrix M contains the information about offer number i : m_{i1} of item a, \dots, m_{i5} of item e can be bought together at a total cost of r_i .

Consider the following example:

- $n_1 = 4, n_2 = 4, n_3 = 2, n_4 = n_5 = 0$.
- $c_1 = 5, c_2 = 3, c_3 = 4, c_4 = c_5 = 10$.
- $s = 3$.
- $m_{11} = 2, m_{12} = 1, m_{13} = m_{14} = m_{15} = 0$ and $r_1 = 12$.
- $m_{21} = 1, m_{22} = 2, m_{23} = m_{24} = m_{25} = 0$ and $r_2 = 8$.
- $m_{31} = 0, m_{32} = 2, m_{33} = 2, m_{34} = m_{35} = 0$ and $r_3 = 10$.

We now consider some possible ways of buying everything you want. In each of these, we only mention how much of each special offer is used: it is understood that the remaining amount (if any) of the required items are bought at their listed price (c_1, c_2 , etc.). For example, when we say below “make use of offer 2 twice and use no other offer”, we mean “use offer 2 twice, then buy 2 a , and 2 c ”. If you do not make use of any special offer, you will need to pay $4 * 5 + 4 * 3 + 2 * 4 = 40$. By making use of offer 1 once, you will need to pay $12 + 2 * 5 + 3 * 3 + 2 * 4 = 39$. But if you make use of offer 1 twice and offer 3 once, then you will just need to pay $12 * 2 + 10 = 34$. Or, you can make use of offer 2 twice and use no other offer, for a total cost of $8 * 2 + 2 * 5 + 2 * 4 = 34$. Another option is: make use of offer 2 once and offer 3 once, for a total cost of $8 + 10 + 3 * 5 = 33$.

- (i) For the moment, assume that you are not allowed to add items even though it may reduce your payment. (In other words, you must buy exactly n_1 of a in total, \dots , n_5 of e in total: no more, no less.) Develop as efficient an algorithm you can, using dynamic programming, for the given problem. Your algorithm *need not* output how you make use of the special offers; it just needs to output the minimum amount of money you need to pay for buying n_1 of item a , n_2 of item b , \dots , n_5 of item e . Analyze the running time of your algorithm. The running time should be at most a polynomial of the value $\max\{s, n_1, n_2, \dots, n_5\}$. **(7 points)**
- (ii) Describe how will you solve the problem efficiently if you are allowed to add items to reduce your payment. You can make use of the algorithm you developed in (i). Again, analyze the running time: it should be at most a polynomial of the value $\max\{s, n_1, n_2, \dots, n_5\}$. **(3 points)**

The solution to these problems will be given on April 18th.

Important Note 1: For problem (i) above, you have the option of “buying a hint” any time before the due date. If you choose to, you can get a hint for this problem from the instructor or the TA (come to their office hours or make an appointment by email). In such a case, you will be given the hint, and your final score for problem (i) will be half of the score you get. You can buy the hint any time before the due date; so, it is best for you to decide as quickly as possible whether you will need the hint.

Important Note 2: As you are doing the above, start coding up a simple “exhaustive search” based algorithm for the problem (i), which basically tries out all possible ways of buying the items, and chooses the optimal solution from these. You can use any programming language and operating system for this. Do not submit this code with Part I; it is due in Part II of the project. But it will help you greatly to start this implementation right away.

2 Part II of the Project

This part is due on May 9th, at the beginning of class. Here, you will code up the exhaustive search algorithm mentioned above, as well as the dynamic programming algorithm (which is the heart of Part I, and whose description will be given on April 18th); you will then compare their running times. The details of how to generate problem instances, report running times, etc., will be announced soon. This part is worth 10 points.