

CS 430

Final Project:
Most profitable Purchase

Group Members:
Sagar Katiyar

UIN:
A20519413

Introduction

The goal of this project was to implement a program to find out the minimal payment of a purchase of a combination of items in a store given its ongoing promotions and deals. We provide 2 files as inputs based on which the program is made:

Input: two .txt files.

(1) input.txt. This file contains the purchase intention and organized as:

line1: the number of merchandises to buy;

line2: identifier/code of the first merchandise; amount; ticket price;

line3: identifier/code of the first merchandise; amount; ticket price;

... ..

(2) promotions.txt. This file addresses the promotion information and organized as:

line1: the number of the promotions that are in effect presently;

line2: the number of merchandise types; id1;amount1;id2;amount...
promotion

price

line3: the number of merchandise types;
id1;amount1;id2;amount2...promotion

price

Given the inputs to our program, the program generates an output that gives the most optimal purchase plan as well as the optimal amount to pay

Methodology

The program is implemented in python and primarily consists of a dynamic programming approach to calculate the optimal payment plan and the minimum associated cost after applying the appropriate promotions: Its high-level overview is as follows:

1. The input file as well as the promotion file are specified as paths upon which they are read and stored in separate lists.
2. A matrix is created of all combinations of the items needed from 0,0 to the total number of items required and we iterate over this matrix. At each iteration to go to the next step we check if there is a valid promotion that can be applied such that the minimum of an incremental price or a promotion price is chosen.
3. Furthermore, a back-pointer array is used to store the selected path so far with the items and/or promotions applied.
4. The cost matrix obtained in step 2 has the minimum total price as the most bottom right element and step 3 specifies the amounts of each item bought as well as the promotions applied which are then stored in output.txt

Test Case Run

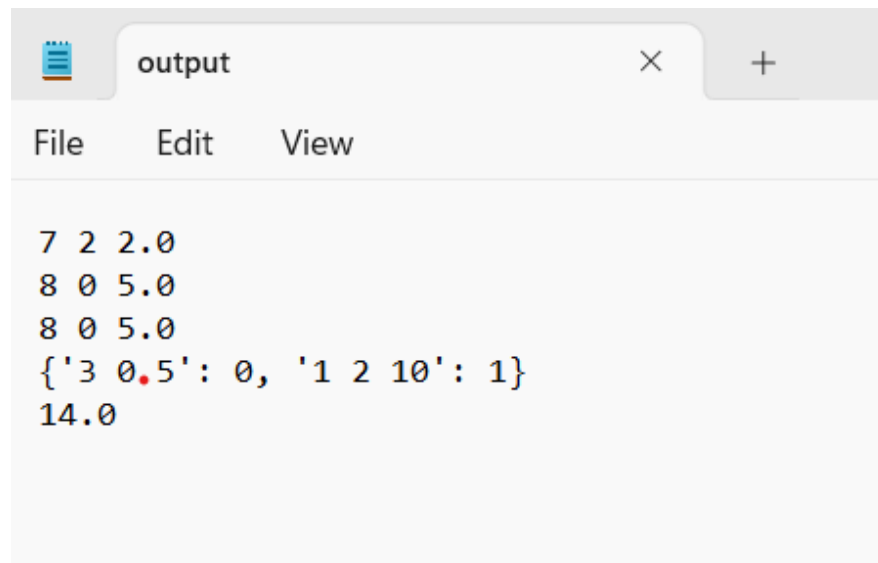
Given the test case:

input.txt	promotion.txt
2	2
7 3 2	1 7 3 5
8 2 5	2 7 1 8 2 10

The expected output is:

```
output.txt
7 2 2
7 1 8 2 10
14
```

After running my program, the output.txt is very similar in format, yielding the correct quantities of each item, the number of valid promotions applied as well as the correct minimum price.



```
7 2 2.0
8 0 5.0
8 0 5.0
{'3 0.5': 0, '1 2 10': 1}
14.0
```

The running time of the main algorithm as well as the backtracking algorithm is $O(n)$ where n refers to the number of items purchased.