Discrete Optimization Assignment:
Warehouse Location

1 Problem Statement

In this assignment you will design an algorithm to solve a problem faced by distribution companies, *The Warehouse Location Problem*. A distribution company uses bulk storage facilities (i.e. warehouses) to provide goods to many different customers. The goal of this problem is to determine which warehouses will be the most cost effective for serving the customers. The complexity of the problem comes from the fact that each warehouse has different costs and storage capabilities.

2 Assignment

Write an algorithm to solve the warehouse location problem. The problem is mathematically formulated in the following way: there are $N = 0 \dots n-1$ warehouses to choose from and $M = 0 \dots m-1$ customers that need to be served. Each warehouse, $w \in N$ has a capacity cap_w and a setup cost s_w. Each customer, $c \in M$, has a demand d_c and travel cost t_{cw} based on which warehouse, $w \in N$ serves it. Lastly, all customers must be served by exactly 1 warehouse. Let a_w be a set variable denoting the customers assigned to warehouse w. Then the warehouse location problem is formalized as the following optimization problem:

$$\begin{aligned}
\text{minimize:} \quad & \sum_{w \in N} \left((|a_w| > 0) s_w + \sum_{c \in a_w} t_{cw} \right) \\
\text{subject to:} \quad & \\
& \sum_{c \in a_w} d_c \leq cap_w \quad (w \in N) \\
& \sum_{w \in N} (c \in a_w) = 1 \quad (c \in M)
\end{aligned}$$

3 Data Format Specification

1 The input consists of $|N| + 2|M| + 1$ lines. The first line contains two numbers, $|N|$ followed by $|M|$. The first line is followed by $|N|$ lines, where each line represents a warehouse capacity cap_w and setup cost s_w. The last $2|M|$ capture the customer information. Each customer block begins with a line with one number, the customer's demand, d_c. The following line has $|N|$ values, one for each warehouse. These values capture the cost to service that customer from each warehouse, t_{cw}.

Input Format

```
|N| |M|
cap_0 s_0
cap_1 s_1
...
cap_|N|-1 s_|N|-1
d_0
t_0_0 t_0_1 t_0_2 ... t_0_|N|-1
d_1
t_1_0 t_1_1 t_1_2 ... t_1_|N|-1
...
d_|M|-1
t_|M|-1_0 t_|M|-1_1 t_|M|-1_2 ... t_|M|-1_|N|-1
```

The output has two lines. The first line contains two values: *obj* and *opt*. *obj* is the cost of the customer warehouse assignment (i.e. the objective value) as a real number. *opt* should be 1 if your algorithm proved optimality and 0 otherwise. The next line is a list of $|M|$ values in N – this is the mapping of customers to warehouses.

Output Format

```
obj opt
c_0 c_2 c_3 ... c_|M|-1
```

Examples Input Example

```
3 4
100 100.123
100 100.456
500 100.789
50
100.1 200.2 2000.3
50
100.4 200.5 2000.6
75
200.7 100.8 2000.9
75
200.10 200.11 100.12
```

Output Example

```
1002.888 0
1 1 0 2
```

This output represents the assignment of customers to warehouses, $a_0 = \{2\}, a_1 = \{0, 1\}, a_2 = \{3\}$. That is, customers 0 and 1 are assigned to warehouse 1, customer 2 is assigned to warehouse 0, and customer 3 is assigned to warehouse 2.

4 Instructions

Edit `solver.py` and modify the `solveIt(inputData)` function to solve the optimization problem described above. The function argument, `inputData`, contains the problem data in the format described above. The return value of `solveIt` is a solution to the problem in the output format described above. Your `solveIt` implementation can be tested with the command,

```
python ./solver.py ./data/<inputFileName>
```

You should limit the `solveIt` method to terminate within 5 hours, otherwise the submission will not be eligible for full credit. You may choose to implement your solver directly in python or modify the `solveIt` function to call an external application.

Resources You will find several warehouse location problem instances in the `data` directory provided with the handout.

Handin Run `submit.pyc` with the command, `python ./submit.pyc`. Follow the instructions to apply your `solveIt` method on the various assignment parts. You can submit multiple times and your grade will be the best of all submissions. However, it may take several minutes before your assignment is graded; please be patient. You can track the status of your submission on the *feedback* section of the assignment website.

Grading Infeasible solutions (i.e. those that do not conform to the output format or violate problem constraints) will receive 0 points. Feasible solutions will receive at least 3 points. Feasible solutions passing a low quality bar will receive at least 7 points and solutions meeting a high quality bar will receive all 10 points. The grading feedback indicates how much your solution must improve to receive a higher grade.

Collaboration Rules In all assignments we encourage collaboration and the exchange of ideas on the discussion forums. However, please refrain from the following:

1. Posting code or pseudo-code related to the assignments.
2. Using code which is not your own.
3. Posting problem solutions.

Discussion of solution quality (i.e. objective value) and algorithm performance (i.e. run time) is allowed and the assignment leader board is designed to encourage such discussions.

Warnings

1. It is recommended you do not modify the `data` directory. Modifying the files in the data directory risks making your assignment submissions incorrect.
2. You cannot rename the `solver.py` file or the `solveIt()` method.
3. Be careful when using global variables in your implementation. The `solveIt()` method will be run repeatedly and it is your job to clear the global data between runs.
4. `solver.py` must remain in the same directory as `submit.pyc`.

5 Technical Requirements

You will need to have python 2.7.x installed on your system (installation instructions, `http://www.python.org/getit/`).