Discrete Optimization Assignment:
Facility Location

1 Problem Statement

In this assignment you will design an algorithm to solve a problem faced by distribution companies, *The Facility Location Problem.* A distribution company uses bulk storage facilities to provide goods to many different customers. The goal of this problem is to determine which facilities will be the most cost effective for serving the customers. The complexity of the problem comes from the fact that each facility has different costs and storage capabilities.[1]

2 Assignment

Write an algorithm to solve the facility location problem. The problem is mathematically formulated in the following way: there are $N = 0 \dots n-1$ facilities to choose from and $M = n \dots n+m-1$ customers that need to be served. Each facility, $f \in N$ has a setup cost s_f and a capacity cap_f. Each customer, $c \in M$, has a demand d_c. Both the facilities and customers are located in a Euclidian space, $\langle x_i, y_i \rangle$ $i \in N \cup M$. The cost to deliver goods to a particular customer c from a facility f is the Euclidean distance between two locations, $dist(f, c)$.[2] Lastly, all customers must be served by exactly 1 facility. Let a_f be a set variable denoting the customers assigned to facility f. Then the facility location problem is formalized as the following optimization problem:

$$\begin{aligned}
\text{minimize:} \quad & \sum_{f \in N} \left((|a_f| > 0) s_f + \sum_{c \in a_f} dist(f, c) \right) \\
\text{subject to:} \quad & \\
& \sum_{c \in a_f} d_c \leq cap_f \quad (f \in N) \\
& \sum_{f \in N} (c \in a_f) = 1 \quad (c \in M)
\end{aligned}$$

[1] Hint: The facility location problem is closely related to the warehouse location problem discussed in the lectures.

[2] $dist(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

3 Data Format Specification

The input consists of $|N|+|M|+1$ lines. The first line contains two numbers, $|N|$ followed by $|M|$. The first line is followed by $|N|$ lines, where each line encodes the facility's setup cost s_f, capacity cap_f, and the location x_f, y_f. The remaining $|M|$ lines capture the customer information, where each line encodes the customer's demand, d_c, and location x_c, y_c.

Input Format

```
|N| |M|
s_0 cap_0 x_0 y_0
s_1 cap_1 x_1 y_1
...
s_|N|-1 cap_|N|-1 x_|N|-1 y_|N|-1
d_|N| x_|N| y_|N|
d_|N|+1 x_|N|+1 y_|N|+1
...
d_|N|+|M|-1  x_|N|+|M|-1 y_|N|+|M|-1
```

The output has two lines. The first line contains two values: *obj* and *opt*. *obj* is the cost of the customer to facility 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 facilities.

Output Format

```
obj opt
c_0 c_1 c_2 ... c_|M|-1
```

Examples Input Example

```
3 4
100 100 1065.0 1065.0
100 100 1062.0 1062.0
100 500 0.0 0.0
50 1397.0 1397.0
50 1398.0 1398.0
75 1399.0 1399.0
75 586.0 586.0
```

Output Example

```
2550.013 0
1 1 0 2
```

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

4 Instructions

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

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

You should limit the `solve_it` 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 `solve_it` function to call an external application.

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

Handin Run `submit.py` with the command, `python ./submit.py`. Follow the instructions to apply your `solve_it` 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 or sharing 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 `solve_it()` method.
3. Be careful when using global variables in your implementation. The `solve_it()` 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.py`.

5 Technical Requirements

You will need to have python 2.7.9 or 3.5 (at least) installed on your system (installation instructions,
`http://www.python.org/downloads/`).