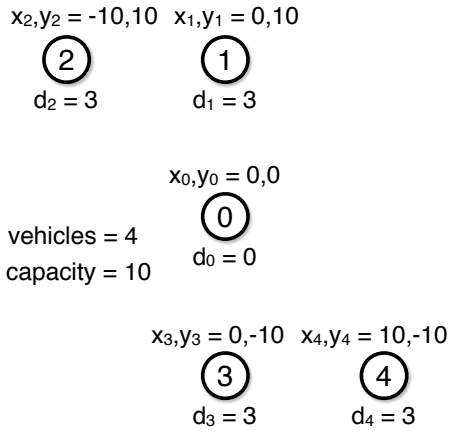
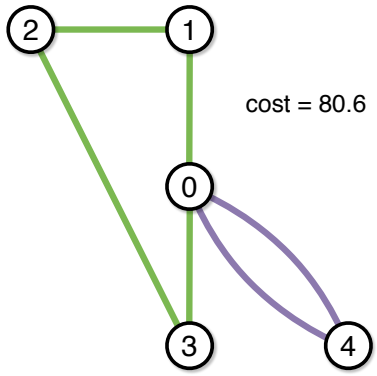
Discrete Optimization Assignment:
Vehicle Routing

1 Problem Statement

In this assignment you will design an algorithm to solve a problem faced by package delivery companies, *The Vehicle Routing Problem (VRP)*. Every day, a delivery company needs to deliver goods to many different customers. The deliveries are achieved by dispatching a fleet of vehicles from a centralized storage warehouse. The goal of this problem is to design a route for each vehicle (similar to traveling salesman tours) so that all of the customers are served by exactly one vehicle and the travel distance of the vehicles is minimized. Additional problem complexity comes from the fact that the vehicles have a fixed storage capacity and the customers have different demands. Figure 1 illustrates a small VRP and a feasible solution to that problem. The customers are labeled from 0 to 4, with 0 being the warehouse. The solution uses two vehicles which are indicated by different colored routes.



(a) The VRP input data.



(b) A solution to the VRP using 2 vehicles.

Figure 1: A Vehicle Routing Example

2 Assignment

Write an algorithm to solve the vehicle routing problem. The problem is mathematically formulated in the following way: We are given a list of locations $N = 0 \ldots n-1$. By convention, location 0 is the warehouse location, where all of the vehicles start and end their routes. The remaining locations are customers. Each location is characterized by three values $\langle d_i, x_i, y_i \rangle$ $i \in N$ a demand d_i and a point x_i, y_i. The fleet of vehicles $V = 0 \ldots v-1$ is fixed and each vehicle has a limited capacity c. All of the demands assigned to a vehicle cannot exceed its capacity c. For each vehicle $i \in V$, let T_i be the sequence of customer deliveries made by that vehicle and let $dist(c_1, c_2)$ be the

Euclidean distance between two customers.[1] Then the vehicle routing problem is formalized as the following optimization problem,

$$\begin{aligned}
\text{minimize:} \quad & \sum_{i \in V} \left(dist(0, T_{i,0}) + \sum_{\langle j,k \rangle \in T_i} dist(j,k) + dist(T_{i,|T_i|-1}, 0) \right) \\
\text{subject to:} \quad & \\
& \sum_{j \in T_i} d_j \leq c \quad (i \in V) \\
& \sum_{i \in V} (j \in T_i) = 1 \quad (j \in N \setminus 0)
\end{aligned}$$

In this variant of the vehicle routing problem, we assume the vehicles can travel in straight lines between each pair of locations.

3 Data Format Specification

The input consists of $|N| + 1$ lines. The first line contains 3 numbers: The number of customers $|N|$, the number of vehicles $|V|$, and the vehicle capacity c. It is followed by $|N|$ lines, each line represents a location triple $\langle d_i, x_i, y_i \rangle$, with a demand $d_i \in \mathbb{N}$ and a point $x_i, y_j \in \mathbb{R}$.

Input Format

```
|N| |V| c
d_0 x_0 y_0
d_1 x_1 y_1
...
d_|N|-1 x_|N|-1 y_|N|-1
```

The output has $|V| + 1$ lines. The first line contains two values *obj* and *opt*. *obj* is the length of all of the vehicle routes (i.e. the objective value) as a real number. *opt* should be 1 if your algorithm proved optimality and 0 otherwise. The following $|V|$ lines represent the vehicle routes T encoding the solution. Each vehicle line starts with warehouse identifier 0 followed by the identifiers of the customers serviced by that vehicle and ends with the warehouse identifier 0. Each vehicle line can contain between 2 and $|N|+2$ values depending on how many customers that vehicle services. Each customer identifier must appear in one of these vehicle lines.

Output Format

```
obj opt
0 t_0_1 t_0_2 ... 0
0 t_1_1 t_1_2 ... 0
...
0 t_|V|-1_1 t_|V|-1_2 ... 0
```

[1] $dist(j,k) = \sqrt{(x_j - x_k)^2 + (y_j - y_k)^2}$

Examples (based on Figure 1)

Input Example

```
5 4 10
0 0 0
3 0 10
3 -10 10
3 0 -10
3 10 -10
```

Output Example 1

```
80.6 0
0 1 2 3 0
0 4 0
0 0
0 0
```

This output represents the following routes for each vehicle. Vehicle 0 - $\{0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 0\}$; Vehicle 1 - $\{0 \rightarrow 4, 4 \rightarrow 0\}$; Vehicle 2 - $\{0 \rightarrow 0\}$; Vehicle 3 - $\{0 \rightarrow 0\}$. Note the following equivalent solution using the same routes with different vehicles.

Output Example 2

```
80.6 0
0 4 0
0 0
0 1 2 3 0
0 0
```

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 vehicle routing 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/`).