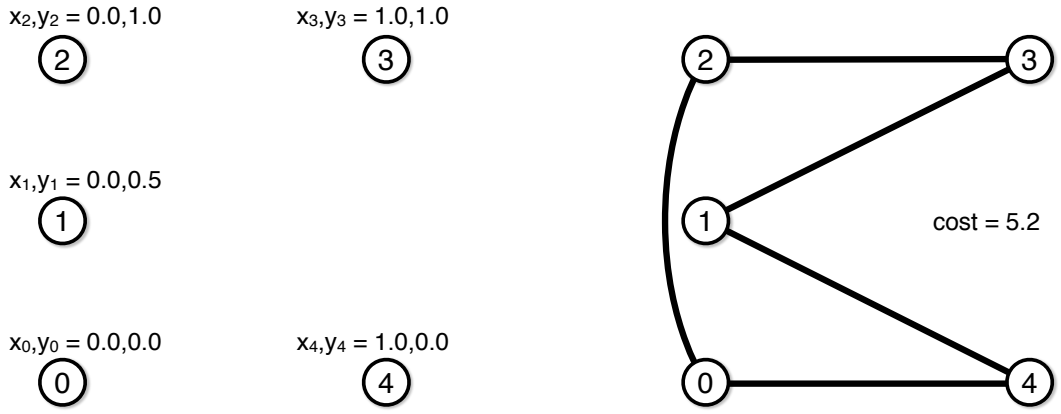
Discrete Optimization Assignment:
Traveling Salesman Problem

1 Problem Statement

In this assignment you will design an algorithm to solve a fundamental problem faced by every traveling salesperson, aptly named *The Traveling Salesman Problem (TSP)*. All traveling salespeople start from their home, travel to several cities to sell their goods, and complete the day by returning home. To minimize their costs, traveling salespeople strive to visit all of the cities using the shortest total travel distance. This amounts to finding a visitation order of all of the cities that minimizes the sum of distances traveled when moving from one city to another. Figure 1 illustrates a small TSP and a feasible solution to that problem. The cities are labeled from 0..4.



(a) The TSP input data.

(b) A solution to the TSP.

Figure 1: A Traveling Salesman Example

2 Assignment

Write an algorithm to solve the traveling salesman problem (a.k.a. minimize the length of a hamiltonian cycle[1] of a graph). The problem is mathematically formulated in the following way: Given a list of locations $N = 0 \ldots n-1$ and coordinates for each location $\langle x_i, y_i \rangle$ $i \in N$. Let v_i $i \in N$ be a variable denoting the visitation order (i.e. the value of v_i is the i-th location to be visited) and let $dist(l_1, l_2)$ be the Euclidean distance between two locations.[2] Then the traveling

[1] See `http://en.wikipedia.org/wiki/Hamiltonian_path`

[2] $dist(n, m) = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}$

salesman problem is formalized as the following optimization problem,

$$\begin{aligned} &\text{minimize:} && \sum_{i \in 0 \ldots n-1} dist(v_i, v_{i+1}) + dist(v_n, v_0) \\ &\text{subject to:} && \\ & && v_i \text{ are a permutation of } N \end{aligned}$$

In this variant of the traveling salesman problem, we assume the sales person travels by helicopter and can go directly in a straight line from one point to another.

3 Data Format Specification

The input consists of $|N| + 1$ lines. The first line contains one number $|N|$. It is followed by $|N|$ lines, each line represents a point $\langle x_i, y_i \rangle$ where $x_i, y_j \in \mathbb{R}$.

Input Format

```
|N|
x_0 y_0
x_1 y_1
...
x_|N|-1 y_|N|-1
```

The output has two lines. The first line contains two values *obj* and *opt*. *obj* is the length of the hamiltonian cycle (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 n values in N, one for each of the v_i variables. This line encodes the solution.

Output Format

```
obj opt
v_0 v_1 v_2 ... v_|N|-1
```

Examples (based on Figure 1)

Input Example

```
5
0 0
0 0.5
0 1
1 1
1 0
```

Output Example

```
5.2 0
0 4 1 3 2
```

This output represents the following hamiltonian cycle, $\{0 \to 4, 4 \to 1, 1 \to 3, 3 \to 2, 2 \to 0\}$

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 traveling salesman 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/`).