SMAI: Assignment 2

Submission: Trial run: September 27, 2300 hrs
Final run: October 14, 2300 hrs

Problem Statement

Given a set of N cities and the distance between every pair of cities, find the shortest possible route that starts from a city and visits every city exactly once and returns to the starting city.

Write a program to find as short a tour as possible of the N cities.

Input Specification

Your program must read from the standard input:

First line: either EUCLIDEAN or NON-EUCLIDEAN.
Second line: an integer N, number of cities.
Next N lines: each line contains 2D coordinates.
Next N lines: each line contains N distance values, i.e., distance to k^{th} city from each city.

The coordinates and distances are space separated list of floating-point numbers.

Use the coordinates for display during development.
Use the distance matrix for computing tour cost.

For example, a Euclidean TSP for N=5:

```
EUCLIDEAN
5
10.391379 8.405525
14.780237 7.036543
1.511838 6.366090
9.276912 5.418818
11.376465 4.216809
0.0 4.597411 9.110738 3.187861 4.302992
4.597411 0.0 13.285327 5.736168 4.420019
9.110738 13.285327 0.0 7.822640 10.096052
3.187861 5.736168 7.822640 0.0 2.419287
4.302992 4.420019 10.096052 2.419287 0.0
```

Output Specification

Your program must write the current best tour (in path representation) to standard output, one tour per line.

output line: PATH REPRESENTATION OF TOUR

The city indices in path-representation are zero based indices, it goes from 0 to N-1.

For example, for N=5:

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

Programming Language

Use Python 3.

Your program will be tested in a Linux environment.

Use Linux compatible tools, libraries and scripts.

Code Submission

You must upload a zip file “your-roll-number.zip” that must contain a single folder named “your-roll-number”, henceforth referred to as HOME folder. The HOME folder must contain the following:

1. source code (source files, libraries, folders),
2. a shell script (named “run”) that calls your program,
3. “your-roll-number.pdf” report that describes your approach, implementation and improvements.

All paths in run-script must be relative to HOME folder.

Sample run script:
python3 relativePath/yourProgramName.py

Evaluation

Each program (process) will be allotted 300 seconds to complete its task and thereafter it will be terminated.

The timeout period (300s) may be revised based on how the submissions perform in the trial run.

The last valid tour in the standard output will be selected for evaluation.

Your program must write at-least one valid tour to the standard output before termination, otherwise evaluation will fail and you will get zero points for that test case.

> *Grading is relative, the best tour will get the highest points.*

Trial Run

You must submit a working (maybe suboptimal) program for a trial run, mainly, to test the compilation, execution and evaluation workflow in the Linux environment.

The PDF report is optional for the trial run.

If the participants submit good quality code (near-final code) then the ranking from the trial run will give you an idea of your relative-performance.

Participation in the trial run carries some points, but the rankings in the trial run do not carry any points.

Final Run

You must submit your final code along with a report for evaluation and grading.

No changes will be permitted after final submission.

During the final run, the submissions that fail to compile or execute will get zero points.

There will be a total of 4 (may be up to 6) test cases with an increasing number of cities, equally split between Euclidean and non-Euclidean TSPs.

Grades will be based on relative performance plus some points for participating in the trial run.