

# AI Assignment - I

---

Travelling Salesman Problem

# Problem Statement

- **Given a file containing the coordinates of cities and the distances between them, write a program that can find the optimal tour .**
-

# Overview

## **Assignment deadline**

October 7, 2016

## **Interim deadline(for trial run)**

October 1, 2016

## **Programming languages**

- C/C++
- Java
- Python

## **Format for file upload**

Group-No.tar.gz (ex: *1.tar.gz*)

---

# General Instructions : Submit a Makefile



Makefile should produce an executable file named **tsp**



Makefile making should produce an executable file named **tsp.class**



Makefile is not required. Please add the Linux hashbang **#!/usr/bin/python** to the file named tsp.

**Please do not place the makefile in nested folders!**

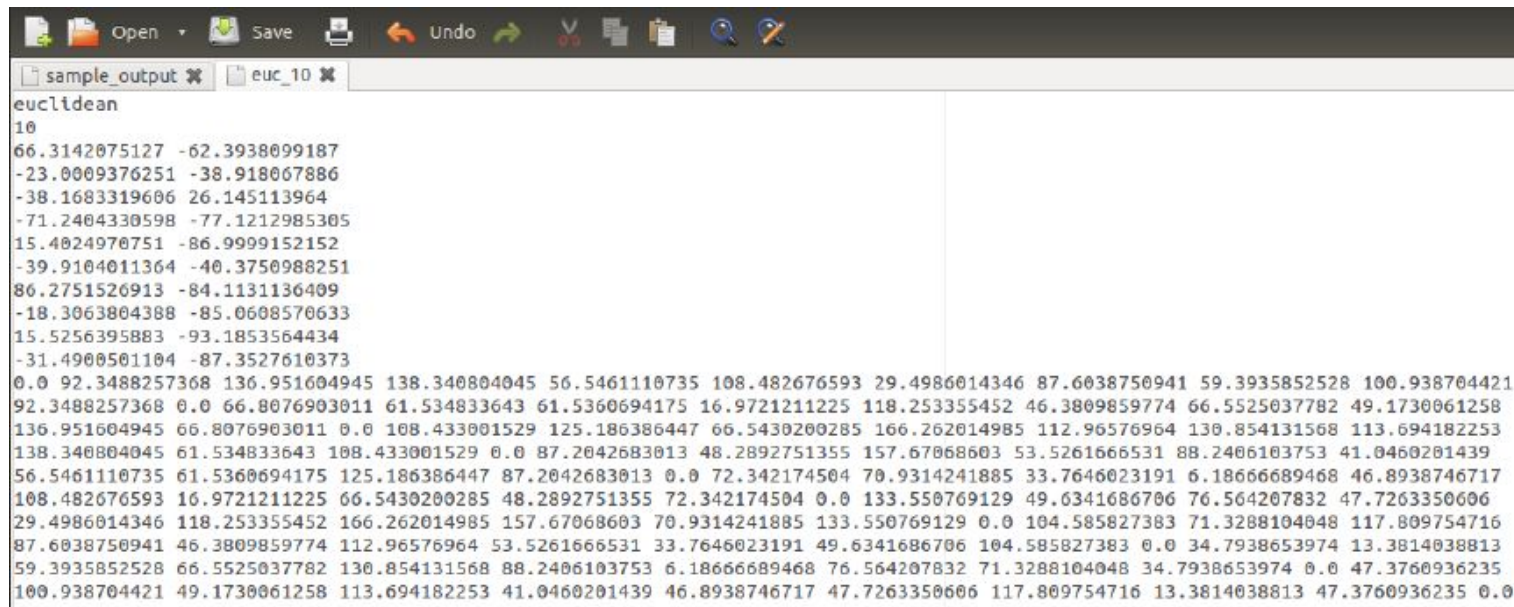
## General Instructions :

- You are allowed to read from stdin and write to stdout.
  - Example : `java tsp < input file > output file` (for Java users)
- You are not allowed to run your code for more than **5 minutes**. Make sure that you **flush stdout periodically**, so that your tours are written to stdout before it gets killed at time-out (5 min).
- If you get better tour than the last tour, you should write it to stdout. The **last line written to stdout will be considered as your final tour**.

**Grading will be completely based on the cost of the tour!**

# General Instructions : Input from *stdin*

- **Sample input : 100 - 500 cities (Euclidean and Non-Euclidean)**

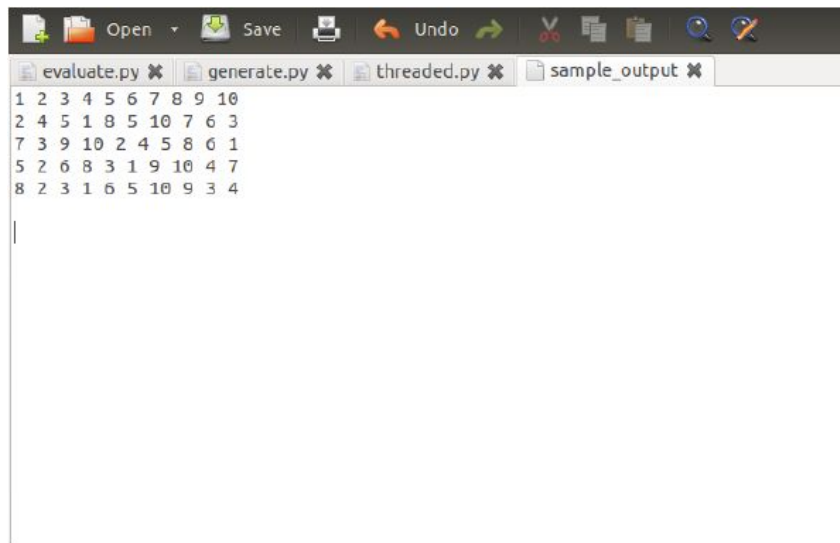


```
euclidean
10
66.3142075127 -62.3938099187
-23.0009376251 -38.918067886
-38.1683319606 26.145113964
-71.2404330598 -77.1212985305
15.4024970751 -86.9999152152
-39.9104011364 -40.3750988251
80.2751526913 -84.1131136409
-18.3063804388 -85.0608570633
15.5256395883 -93.1853564434
-31.4900501104 -87.3527610373
0.0 92.3488257368 136.951604945 138.340804045 56.5461110735 108.482676593 29.4986014346 87.6038750941 59.3935852528 100.938704421
92.3488257368 0.0 66.8076903011 61.534833643 61.5360694175 16.9721211225 118.253355452 46.3809859774 66.5525037782 49.1730061258
136.951604945 66.8076903011 0.0 108.433001529 125.186386447 66.5430200285 166.262014985 112.96576964 130.854131568 113.694182253
138.340804045 61.534833643 108.433001529 0.0 87.2042683013 48.2892751355 157.67068603 53.5261666531 88.2406103753 41.0460201439
56.5461110735 61.5360694175 125.186386447 87.2042683013 0.0 72.342174504 70.9314241885 33.7646023191 6.18666689468 46.8938746717
108.482676593 16.9721211225 66.5430200285 48.2892751355 72.342174504 0.0 133.550769129 49.6341686706 76.564207832 47.7263350606
29.4986014346 118.253355452 166.262014985 157.67068603 70.9314241885 133.550769129 0.0 104.585827383 71.3288104048 117.809754716
87.6038750941 46.3809859774 112.96576964 53.5261666531 33.7646023191 49.6341686706 104.585827383 0.0 34.7938653974 13.3814038813
59.3935852528 66.5525037782 130.854131568 88.2406103753 6.18666689468 76.564207832 71.3288104048 34.7938653974 0.0 47.3760936235
100.938704421 49.1730061258 113.694182253 41.0460201439 46.8938746717 47.7263350606 117.809754716 13.3814038813 47.3760936235 0.0
```

(Input format)

# General Instructions : Output to *stdout*

- Write the space separated indexes of the cities
- Each tour should be in single line.
- Avoid the cycle (Coming back to the origin city),
  - Example: 2 6 3 1 8 9 7 5 4 2



The screenshot shows a code editor with a toolbar at the top containing icons for Open, Save, Print, Undo, and other standard functions. Below the toolbar, there are four tabs: `evaluate.py`, `generate.py`, `threaded.py`, and `sample_output`. The `sample_output` tab is active and displays a 10x10 distance matrix. The matrix is as follows:

1	2	3	4	5	6	7	8	9	10
2	4	5	1	8	5	10	7	6	3
7	3	9	10	2	4	5	8	6	1
5	2	6	8	3	1	9	10	4	7
8	2	3	1	6	5	10	9	3	4

Below the matrix, there is a blank line with a cursor, indicating where the output should be written.

(Output format)