< Programming Assignment #3 >

Submission: **LMS** (learning.hanyang.ac.kr)

- Until **June 13** (Thursday), 23:59 PM. **And for this assignment #3, there is no late submission!**

1. Environment

- OS: Windows, Mac OS, or Linux
- Language: Python (any version is fine, but python3 is recommended)

2. Goal: Perform **clustering** on a given data set by using **DBSCAN**.

3. Requirements

The program must meet the following requirements:

- File name: studentID_name_hw3.**py** **(not ipynb!)**
  - You can put your name in either Korean or English. For english name students, do NOT put space or _ between your first and last names as well as inside your student ID. And only use plain alphabet, do not use like ñ.
  - Example 1: **2008011666_채동규_hw3.py**
  - Example 2: **2025016242_albertkarlo_hw3.py**
- I will execute your code with **four** arguments: **input data file name, n, Eps and MinPts**
  - ***n***: number of clusters for the corresponding input data
  - ***Eps***: maximum radius of the neighborhood
  - ***MinPts***: minimum number of points in an Eps-neighborhood of a given point
  - I will provide three input files. For each input file, I suggest you to use the following parameters (***n***, ***Eps***, ***MinPts***) for each input data.
    - For 'input1.txt', ***n***=8, ***Eps***=15, ***MinPts***=22
    - For 'input2.txt', ***n***=5, ***Eps***=2, ***MinPts***=7
    - For 'input3.txt', ***n***=4, ***Eps***=5, ***MinPts***=5
  - Example: python 2008011666_채동규_hw3.py input1.txt 8 15 22
    - This means that the chosen dataset filename is **input1.txt**, ***n***=8, ***Eps***=15, and ***MinPts***=22
    - I will test your assignment using several different datasets and the parameter values (***n***, ***Eps***, ***MinPts***) which are optimized to the datasets.
- File format for an input data

  [*object_id_1*]\t[*x_coordinate*]\t[*y_coordinate*]\n

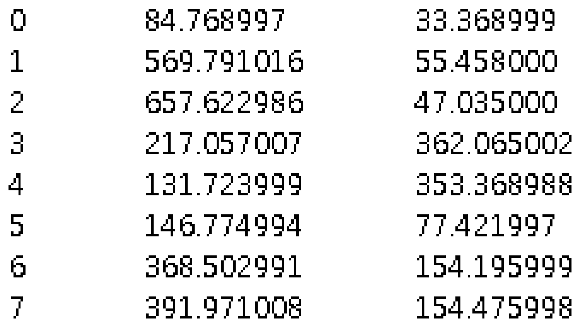
  [*object_id_2*]\t[*x_coordinate*]\t[*y_coordinate*]\n

  [*object_id_3*]\t[*x_coordinate*]\t[*y_coordinate*]\n

  [*object_id_4*]\t[*x_coordinate*]\t[*y_coordinate*]\n

  …

  - Row: information of an object
    - [*object_id_i*]: ID of the *i*th object
    - [*x_coordinate*], [*y_coordinate*]: the location of the corresponding object in the 2-dimensional space
  - Example:



```
0       84.768997       33.368999
1       569.791016      55.458000
2       657.622986      47.035000
3       217.057007      362.065002
4       131.723999      353.368988
5       146.774994      77.421997
6       368.502991      154.195999
7       391.971008      154.475998
```

- Output files
  - You must print ***n*** output files for each input data
    - (Optional) If your algorithm finds *m* clusters for an input data and *m* is greater than *n* (*n* = the number of clusters given), you can remove (*m-n*) clusters based on the number of objects within each cluster. In order to remove (*m-n*) clusters, for example, you can select (*m-n*) clusters with the small sizes in ascending order
    - You can remove outlier. In other words, you don't need to include outlier in a specific cluster
  - **File format for the output of 'input#.txt'**
    - 'input#_cluster_0.txt' Example for **input1.txt**: input1_cluster_0.txt

      [*object_id*]\n …

      [*object_id*]\n

      ...
    - 'input#_cluster_1.txt' Example for **input1.txt**: input1_cluster_1.txt

      [*object_id*]\n …

      [*object_id*]\n

      ...
    - 'input#_cluster_*n-1*.txt' …

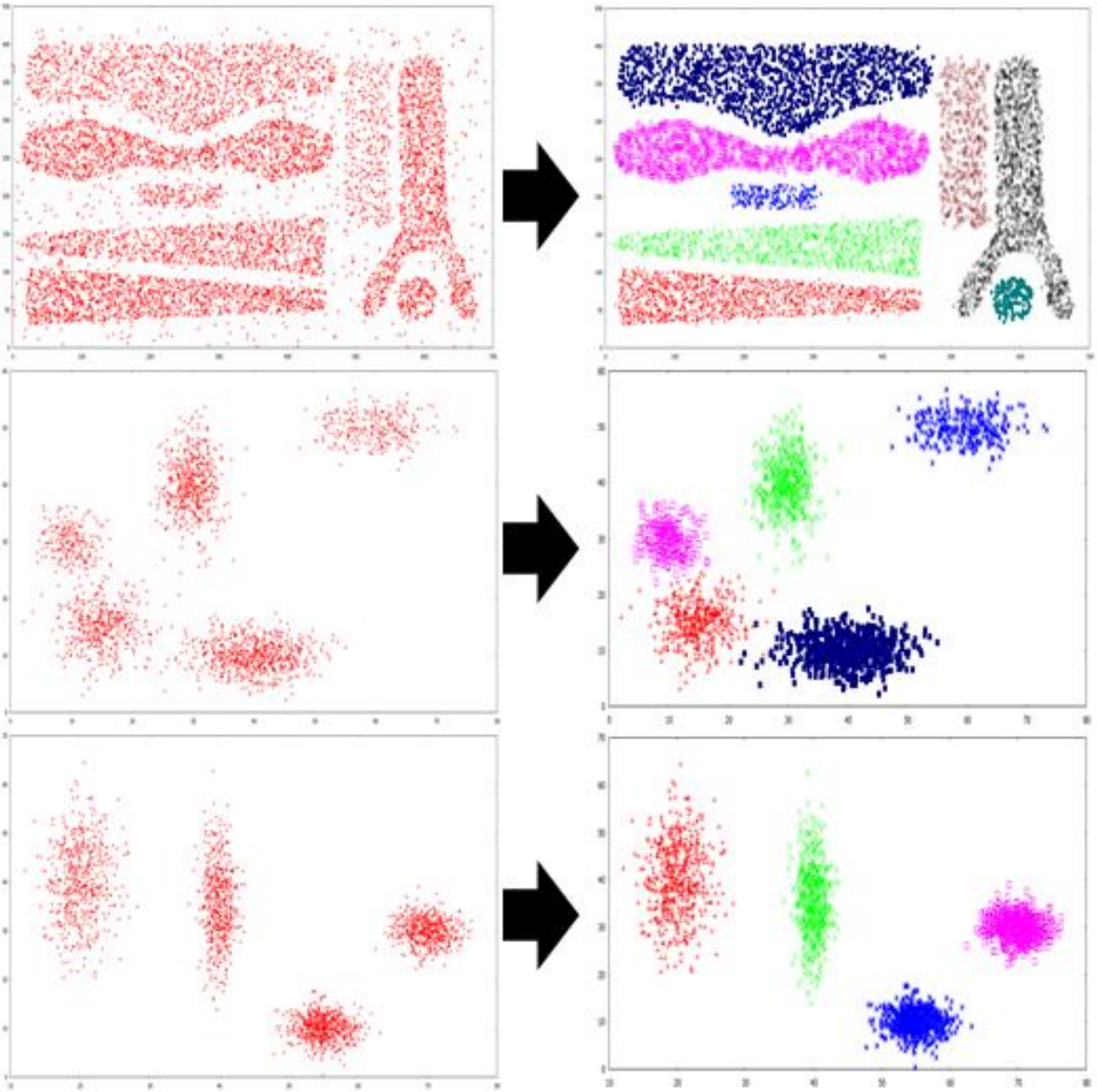
      [*object_id*]\n

      [*object_id*]\n

      ...
  - 'input#_cluster_*i*.txt' should contain all the IDs belonging to cluster *i* that were obtained by using your algorithm
- Important notes
  - Again, do NOT use the external libraries which are directly related to and have implementations of the DBSCAN algorithm. You need to implement the code from scratch, with using only basic libraries that help File I/O, data structure handling, distance computation, couting, etc.
  - **IMPORTANT:** your program must **NOT** print anything except the output files. If you want to see some visualization of your results, please do it only before submitting, and **in the final submission version, REMOVE all print-related functions** (ex. print("~~~~~~~"), tqdm(range())).

4. Rubric

- The following figure shows the clustering result for each input data. BUT again, if you want to see some visualization of your results, please do it only before submitting, and **in the final submission version, REMOVE all print-related functions!**



- Test method
  - (**just for your information**) For testing, I will use a measure similar to the Kendall’s tau distance. Please refer to the following wikipedia page.

    (http://en.wikipedia.org/wiki/Kendall_tau_rank_correlation_coefficient)

    - Example
      - Correct answer: [*object_id_1*] and [*object_id_2*] are contained in different clusters
      - Your answer
        - [*object_id_1*] and [*object_id_2*] are contained in the same cluster → *INCORRECT*
        - [*object_id_1*] and [*object_id_2*] are contained in different clusters → *CORRECT*
    - The final score will be computed as follows:

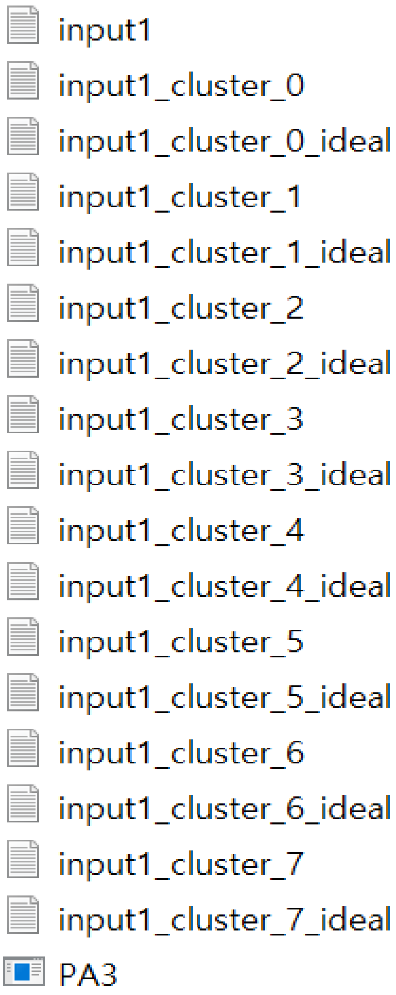
$$\frac{\textit{The number of correct pairs}}{\textit{The number of all possible pairs}}$$

5. Submission

- Please make **a single, runnable .py file** and upload it on LMS.

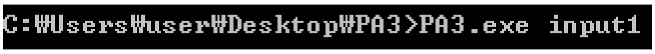
6. Testing program

- Please put the following files in a same directory: Testing program, your output files, given input files, attached answer files(~ideal.txt)



input1
input1_cluster_0
input1_cluster_0_ideal
input1_cluster_1
input1_cluster_1_ideal
input1_cluster_2
input1_cluster_2_ideal
input1_cluster_3
input1_cluster_3_ideal
input1_cluster_4
input1_cluster_4_ideal
input1_cluster_5
input1_cluster_5_ideal
input1_cluster_6
input1_cluster_6_ideal
input1_cluster_7
input1_cluster_7_ideal
PA3

- Execute the testing program with one argument (input file name)



```
C:₩Users₩user₩Desktop₩PA3>PA3.exe input1
```

- Check your score for the input file
  - If you implement your DBSCAN algorithm successfully and use the given parameters mentioned above, you will be able to get the similar scores with the following score for each input data
    - For ‘input1.txt’,   Score=99
    - For ‘input2.txt’,   Score=95
    - For ‘input3.txt’,   Score=99
  - The test program was build with program 'mono'. So, even if you are using mac or linux instead of window, you can run dt_test.exe using C# mono.

7. Penalty

- **Late submission**
  - **I will not accept submissions after the deadline.**
- Requirements unsatisfied
  - Penalty up to 100% will be given depending on how the requirements are well-satisfied
- Plagiarim (from Internet, assisted by ChatGPT or any other AI-based generation, borrowed from someone else, etc)
  - Such plagiarim can be easily detected, and if detected, F grade will be given.
  - Please do it your own. This is not a difficult assignment.