

Course name: Data Science (ITE4005)

Professor: Dong-Kyu Chae (email: dongkyu@hanyang.ac.kr)

Teaching Assistant: Byeongtae Park (email: byeongtae@hanyang.ac.kr)

< Programming Assignment #3 >

09 May 2022

Due Date: 23 May 2022, 11:59 pm

1. Environment

- OS: Windows, Mac OS, or Linux
- Languages: Java or Python (any version is ok)

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

3. Requirements

The program must meet the following requirements:

- Execution file name: **clustering.py** (or clustering.exe, clustering.etc ...)
 - Execute the program with four arguments: input data file name,
 - ***n***, ***Eps*** and ***MinPts***
 - Three input data will be provided: 'input1.txt', 'input2.txt', 'input3.txt
 - ***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
 - We suggest that you 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:

```
clustering.exe input1.txt 8 15 22
```

- Input data file name = 'input1.txt', ***n*** = 8, ***Eps*** = 15, ***MinPts*** = 22

- 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]`: identifier 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

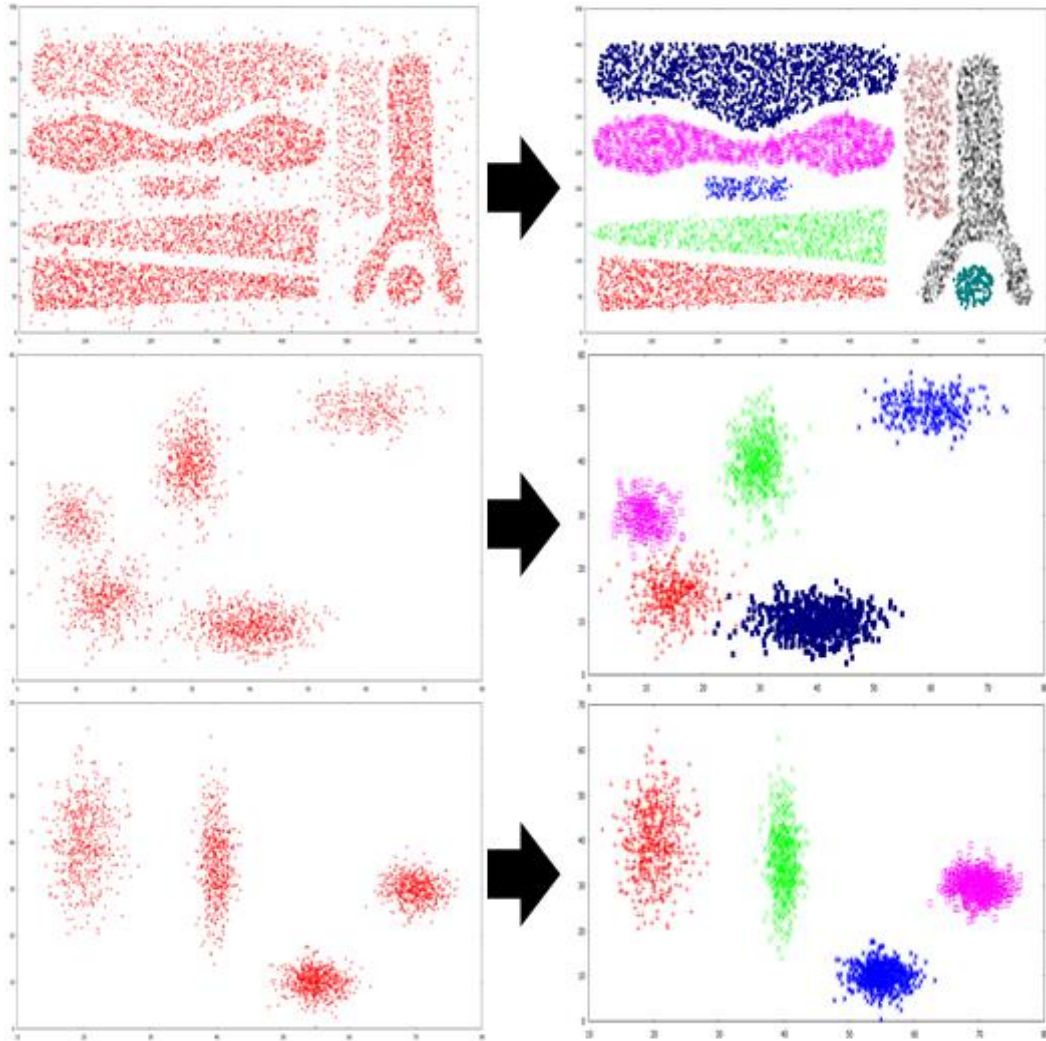
Figure 1. An example of an input data.

- 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'
 - `[object_id]`\n
 - `[object_id]`\n
 - ...
 - 'input#_cluster_1.txt'
 - `[object_id]`\n
 - `[object_id]`\n
 - ...
 - 'input#_cluster_n-1.txt'
 - `[object_id]`\n
 - `[object_id]`\n
 - ...
- 'output#_cluster_i.txt' should contain all the ids belonging to cluster i that were obtained by using your algorithm
- Supposed to follow the naming scheme for the output file as above

4. Rubric

- The following figure shows the clustering result for each input data



- Test method

- For testing, we will use a measure similar to the Kendall's tau measure. 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:

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

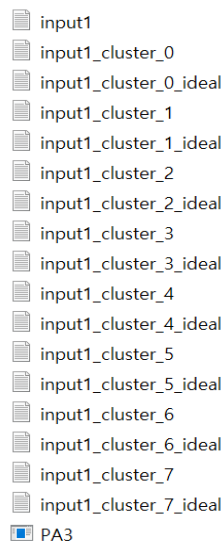
5. Submission

- Please submit the program files and the report to *GitLab*
- Report
 - File format must be *.pdf.
 - Guideline

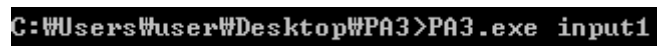
- ✓ Summary of your algorithm
 - ✓ Detailed description of your codes (for each function)
 - ✓ **Instructions for compiling/executing your source codes at TA's computer (e.g. screenshot) (Important!!)**
 - ✓ Any other specification of your implementation and testing
- Program and code
- An executable file (.exe or .py)
 - All source files
 - ✓ MakeFile if you use Linux

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)



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



- 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
 - 1 week delay: 20%
 - 2 weeks delay: 50%
 - Delay more than 2 weeks: 100%
- Requirements unsatisfied
 - Penalty up to 100% will be given depending on how the requirements are well-satisfied