Data Science

Programming Assignment 3

clustering on a given data set by using **DBSCAN**.

Design Document

2015004011

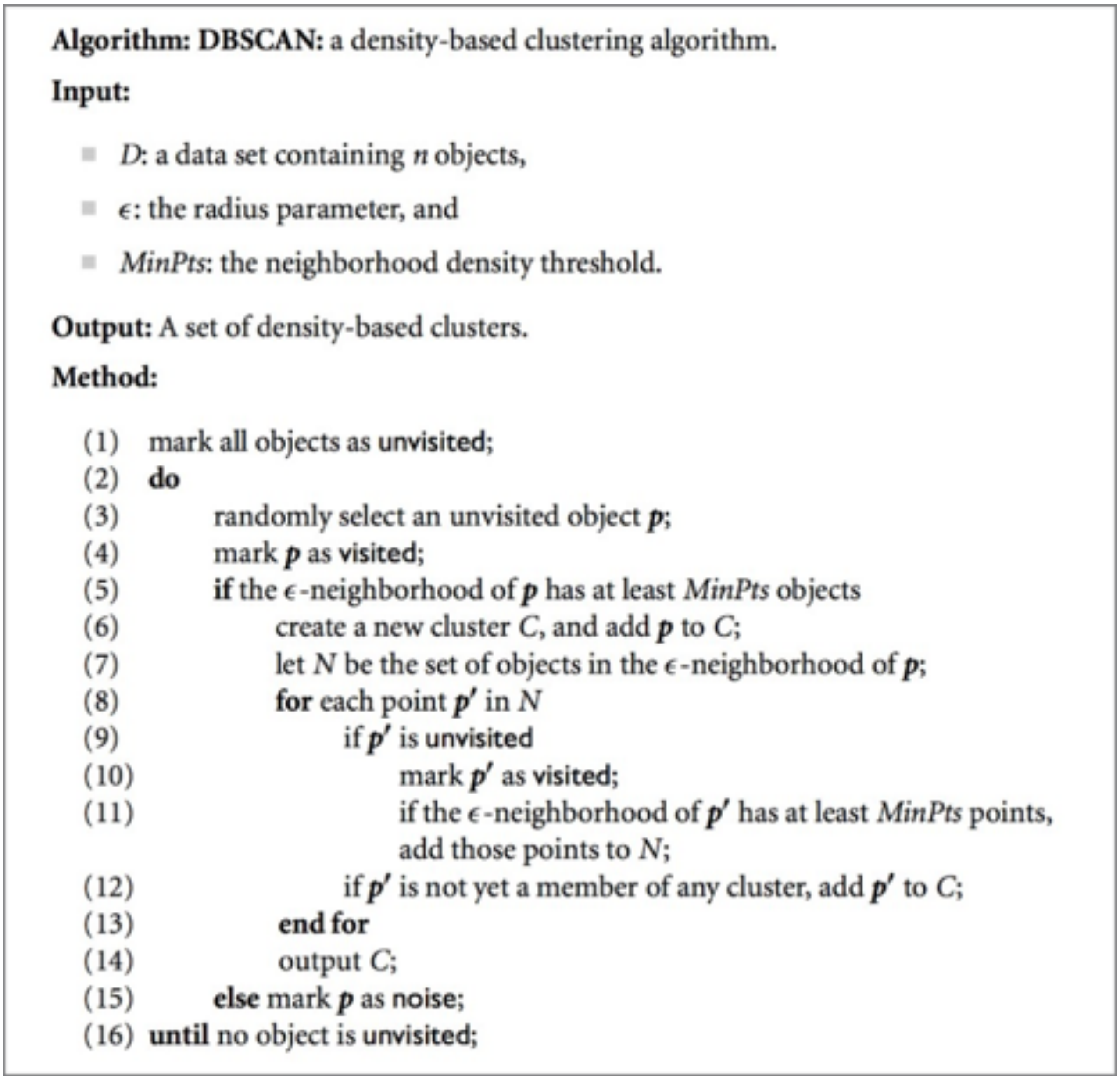
major in Software

Dohyun Kim

1. Summary of my algorithm

This assignment is to implement the '**clustering** on a given data set by using **DBSCAN**'. DBSCAN is a data clustering algorithm that is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.

I followed the pseudo-code of the lecture material.



Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

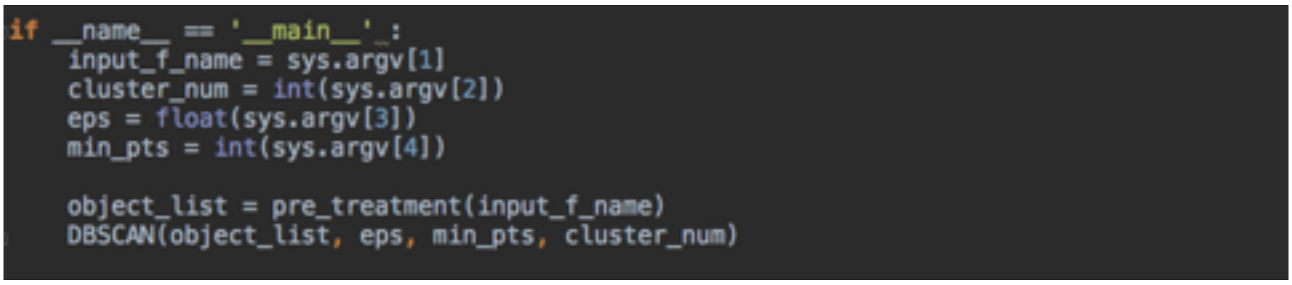
Output: A set of density-based clusters.

Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the $\epsilon$-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the $\epsilon$-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the $\epsilon$-neighborhood of p' has at least MinPts points,
                     add those points to N;
(12)             if p' is not yet a member of any cluster, add p' to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

2. Detailed description of my codes

1.main

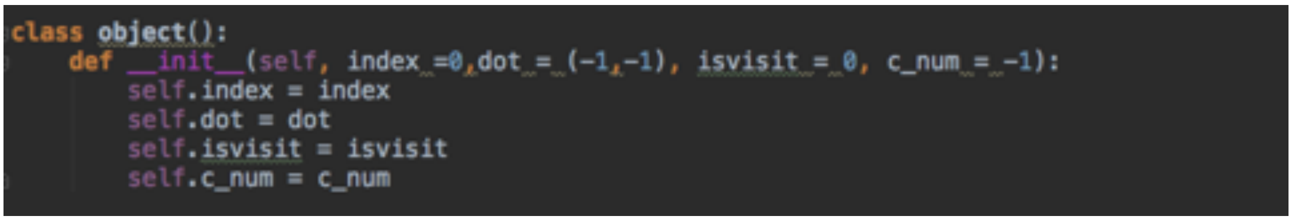


```python
if __name__ == '__main__':
    input_f_name = sys.argv[1]
    cluster_num = int(sys.argv[2])
    eps = float(sys.argv[3])
    min_pts = int(sys.argv[4])

    object_list = pre_treatment(input_f_name)
    DBSCAN(object_list, eps, min_pts, cluster_num)
```

first, It is used as an argument when executing, and it specifies the input file , number of clusters for the corresponding input data, maximum radius of the neighborhood, minimum number of points in an Eps-neighborhood of a given point. Reads the file, preprocesses it, and returns an object_list consisting of an array of class objects. Then DBSCAN algorithm is executed by using DBSCAN function.

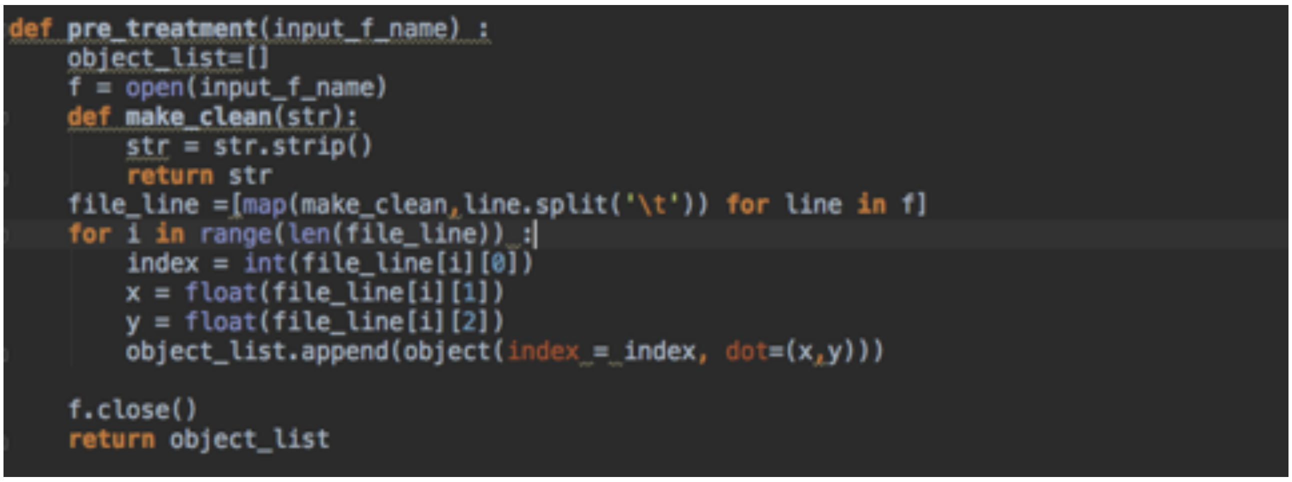
2. class object



```python
class object():
    def __init__(self, index =0,dot = (-1,-1), isvisit = 0, c_num = -1):
        self.index = index
        self.dot = dot
        self.isvisit = isvisit
        self.c_num = c_num
```

This class contains information about each point. This class was written for use as a struct in C language. This class consists of an index, a tuple named 'dot' with each coordinate, 'isvisit' to check visitation, and 'c_num' to store cluster number.

3.pre_treatment(input_f_name)

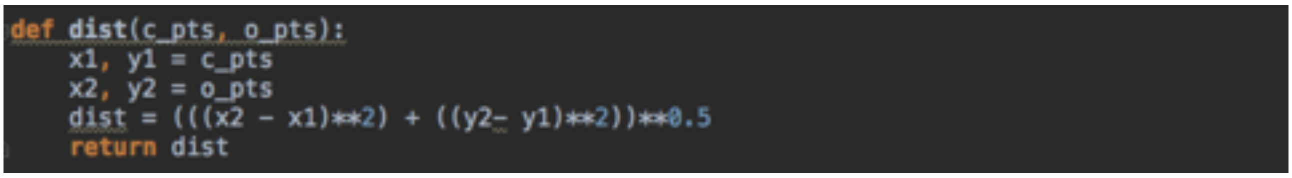


```python
def pre_treatment(input_f_name) :
    object_list=[]
    f = open(input_f_name)
    def make_clean(str):
        str = str.strip()
        return str
    file_line =[map(make_clean,line.split('\t')) for line in f]
    for i in range(len(file_line)) :
        index = int(file_line[i][0])
        x = float(file_line[i][1])
        y = float(file_line[i][2])
        object_list.append(object(index = index, dot=(x,y)))

    f.close()
    return object_list
```

This function reads the file in the argument and divides it by "\ t". It reads the coordinates of each point, creates it as a class called object, creates the list as a list, and returns it.

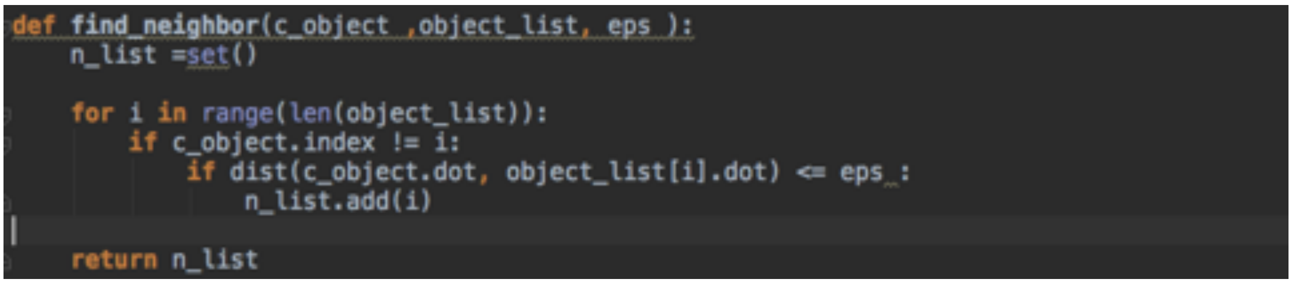
4. dist(c_pts,o_pts)



```python
def dist(c_pts, o_pts):
    x1, y1 = c_pts
    x2, y2 = o_pts
    dist = (((x2 - x1)**2) + ((y2- y1)**2))**0.5
    return dist
```

This function is a function to determine the distance between two points, each of which consists of a tuple containing coordinates.

5. find_neighbor(c_object, object_list, eps)



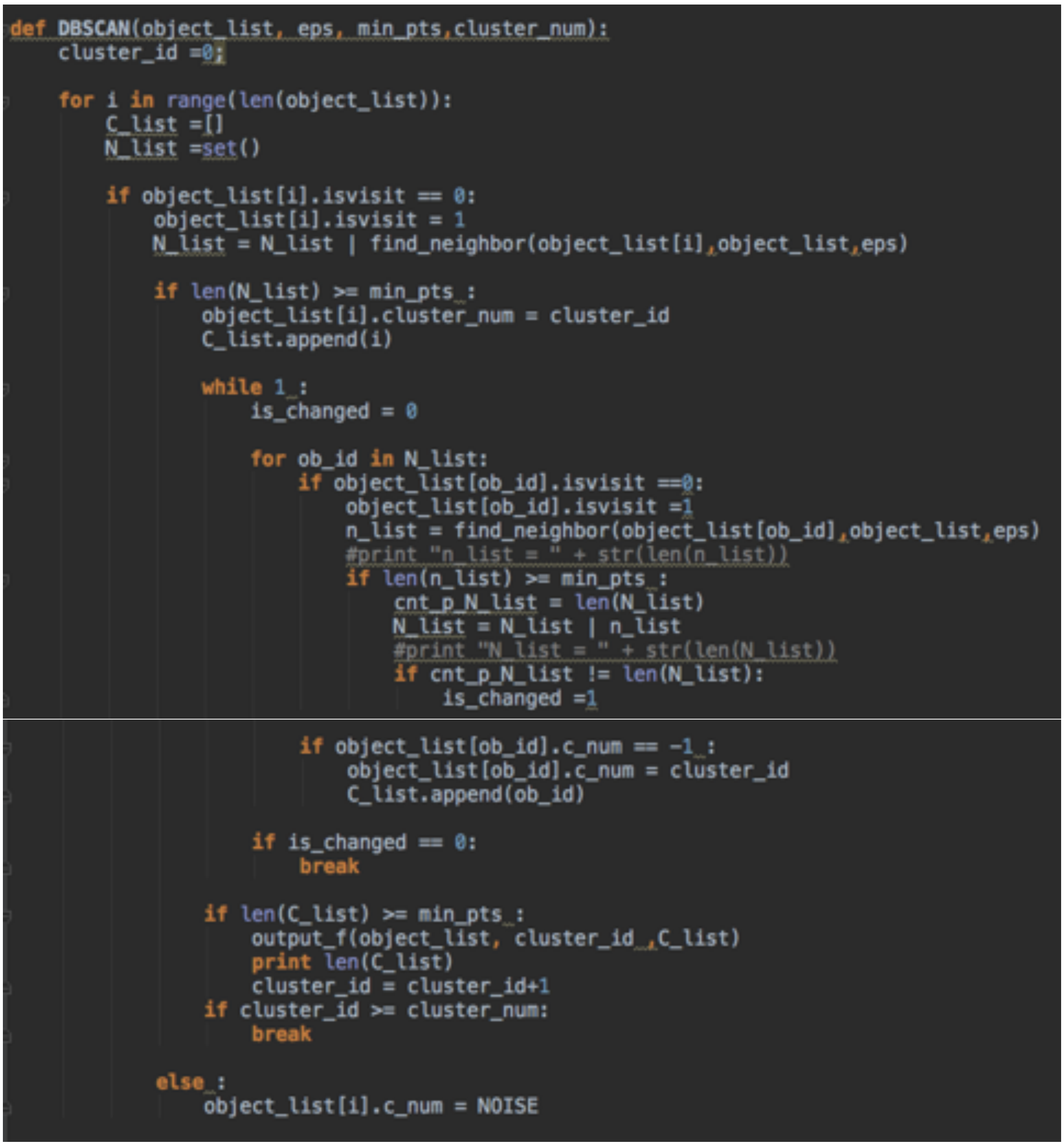
```python
def find_neighbor(c_object ,object_list, eps ):
    n_list =set()

    for i in range(len(object_list)):
        if c_object.index != i:
            if dist(c_object.dot, object_list[i].dot) <= eps :
                n_list.add(i)

    return n_list
```

This function recognizes all points with a distance less than eps as a neighbor for a specific point, and returns a set of neighbors.

6.DBSCAN(object_list, eps, min_pts, clluster_num)



```
def DBSCAN(object_list, eps, min_pts,cluster_num):
    cluster_id =0;

    for i in range(len(object_list)):
        C_list =[]
        N_list =set()

        if object_list[i].isvisit == 0:
            object_list[i].isvisit = 1
            N_list = N_list | find_neighbor(object_list[i],object_list,eps)

            if len(N_list) >= min_pts :
                object_list[i].cluster_num = cluster_id
                C_list.append(i)

                while 1 :
                    is_changed = 0

                    for ob_id in N_list:
                        if object_list[ob_id].isvisit ==0:
                            object_list[ob_id].isvisit =1
                            n_list = find_neighbor(object_list[ob_id],object_list,eps)
                            #print "n_list = " + str(len(n_list))
                            if len(n_list) >= min_pts :
                                cnt_p_N_list = len(N_list)
                                N_list = N_list | n_list
                                #print "N_list = " + str(len(N_list))
                                if cnt_p_N_list != len(N_list):
                                    is_changed =1
                        if object_list[ob_id].c_num == -1 :
                            object_list[ob_id].c_num = cluster_id
                            C_list.append(ob_id)

                    if is_changed == 0:
                        break

                if len(C_list) >= min_pts :
                    output_f(object_list, cluster_id ,C_list)
                    print len(C_list)
                    cluster_id = cluster_id+1
                if cluster_id >= cluster_num:
                    break

            else :
                object_list[i].c_num = NOISE
```

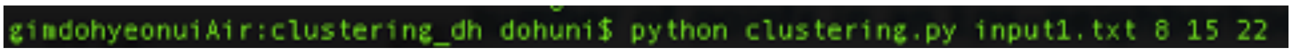
This function is the core function of the DBSCAN program. The difference from the pseudo-code is that we do not randomly visit the points, but rather visit them sequentially. The reason is that if you visit at random, it will be difficult to visit without missing all points. Pseudo-code, and we assumed that a neigbor closer to eps is larger than min_pts to form a cluster. The reason for creating the is_changed variable is that the problem occurs because the N_list in the for loop is not updated when the N_list changes.

3. Instructions for compiling my source code

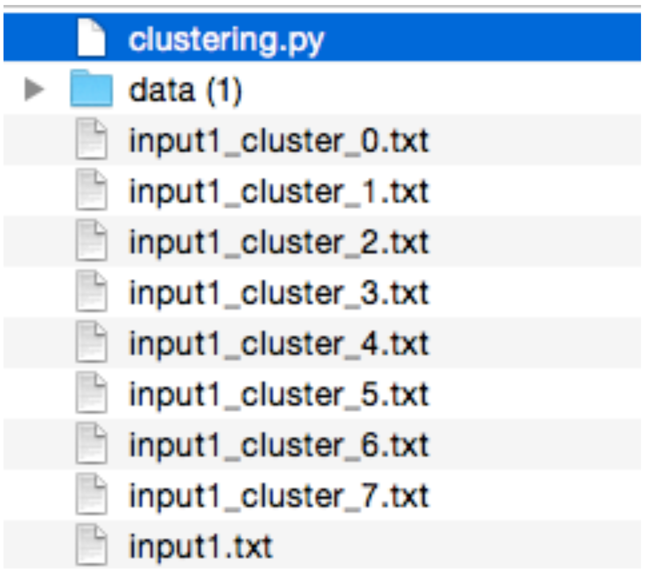
My code was written in Python so when you run it you need to input
"python clustering.py input1.txt 8 15 22" in this way.
Enter python first, followed by the input file name, number of clusters for the corresponding input data , maximum radius of the neighborhood, and minimum number of points in an Eps-neighborhood of a given point

.

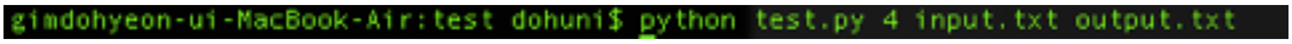
4. test result



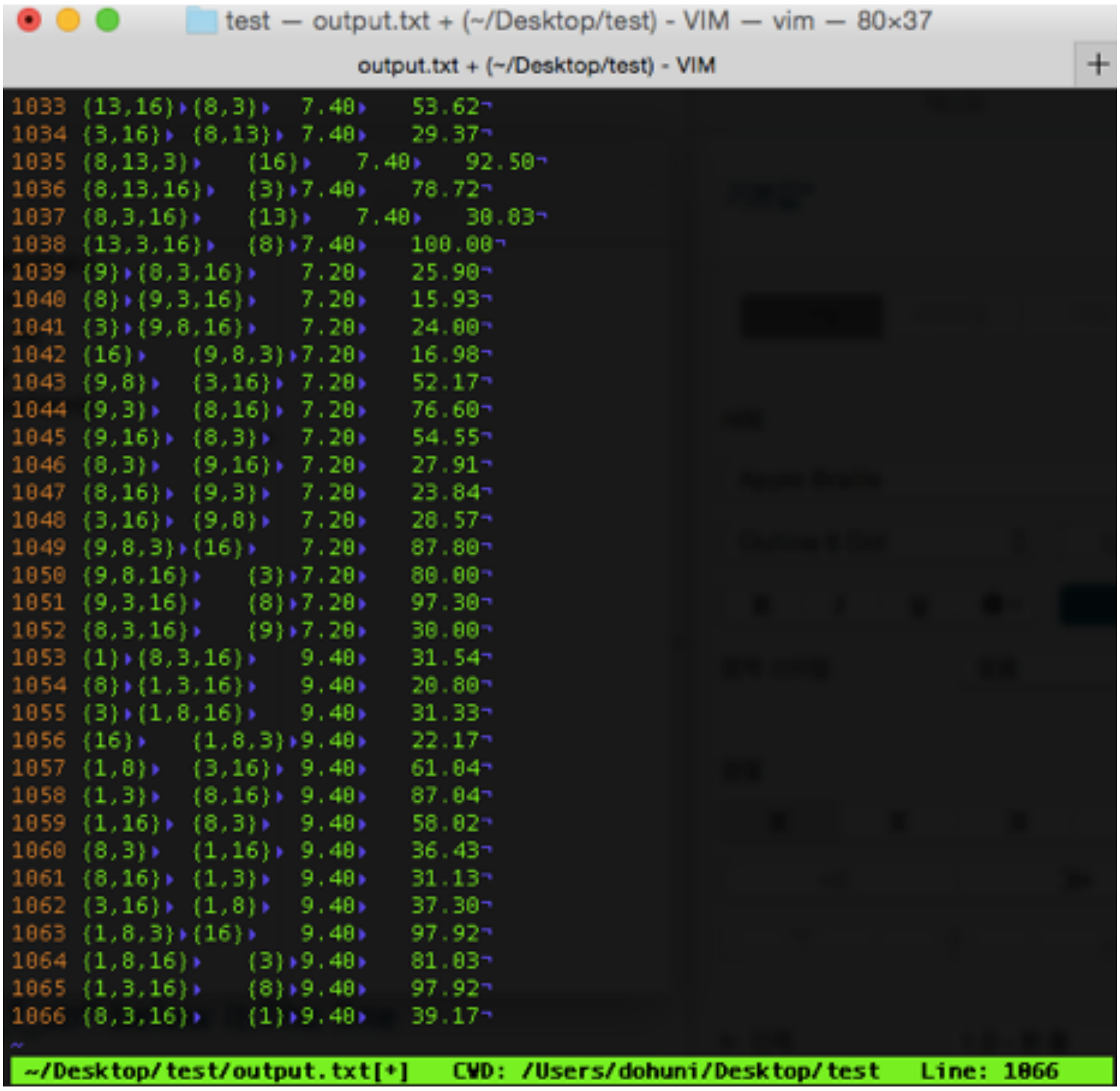
```
gimdohyeonuiAir:clustering_dh dohuni$ python clustering.py input1.txt 8 15 22
```



```
clustering.py
data (1)
input1_cluster_0.txt
input1_cluster_1.txt
input1_cluster_2.txt
input1_cluster_3.txt
input1_cluster_4.txt
input1_cluster_5.txt
input1_cluster_6.txt
input1_cluster_7.txt
input1.txt
```



```
gimdohyeon-ui-MacBook-Air:test dohuni$ python test.py 4 input.txt output.txt
```


test — output.txt + (~/Desktop/test) - VIM — vim — 80×37
output.txt + (~/Desktop/test) - VIM

```
1033 {13,16}▸{8,3}▸  7.40▸   53.62¬
1034 {3,16}▸ {8,13}▸ 7.40▸   29.37¬
1035 {8,13,3}▸    {16}▸   7.40▸   92.50¬
1036 {8,13,16}▸   {3}▸7.40▸   78.72¬
1037 {8,3,16}▸    {13}▸   7.40▸   30.83¬
1038 {13,3,16}▸   {8}▸7.40▸   100.00¬
1039 {9}▸{8,3,16}▸    7.20▸   25.90¬
1040 {8}▸{9,3,16}▸    7.20▸   15.93¬
1041 {3}▸{9,8,16}▸    7.20▸   24.00¬
1042 {16}▸    {9,8,3}▸7.20▸   16.98¬
1043 {9,8}▸   {3,16}▸ 7.20▸   52.17¬
1044 {9,3}▸   {8,16}▸ 7.20▸   76.60¬
1045 {9,16}▸  {8,3}▸  7.20▸   54.55¬
1046 {8,3}▸   {9,16}▸ 7.20▸   27.91¬
1047 {8,16}▸  {9,3}▸  7.20▸   23.84¬
1048 {3,16}▸  {9,8}▸  7.20▸   20.57¬
1049 {9,8,3}▸{16}▸    7.20▸   87.80¬
1050 {9,8,16}▸    {3}▸7.20▸   80.00¬
1051 {9,3,16}▸    {8}▸7.20▸   97.30¬
1052 {8,3,16}▸    {9}▸7.20▸   30.00¬
1053 {1}▸{8,3,16}▸    9.40▸   31.54¬
1054 {8}▸{1,3,16}▸    9.40▸   20.80¬
1055 {3}▸{1,8,16}▸    9.40▸   31.33¬
1056 {16}▸    {1,8,3}▸9.40▸   22.17¬
1057 {1,8}▸   {3,16}▸ 9.40▸   61.04¬
1058 {1,3}▸   {8,16}▸ 9.40▸   87.04¬
1059 {1,16}▸  {8,3}▸  9.40▸   58.02¬
1060 {8,3}▸   {1,16}▸ 9.40▸   36.43¬
1061 {8,16}▸  {1,3}▸  9.40▸   31.13¬
1062 {3,16}▸  {1,8}▸  9.40▸   37.30¬
1063 {1,8,3}▸{16}▸    9.40▸   97.92¬
1064 {1,8,16}▸    {3}▸9.40▸   81.03¬
1065 {1,3,16}▸    {8}▸9.40▸   97.92¬
1066 {8,3,16}▸    {1}▸9.40▸   39.17¬
~
 ~/Desktop/test/output.txt[+]   CWD: /Users/dohuni/Desktop/test   Line: 1066
```