Data Science Class

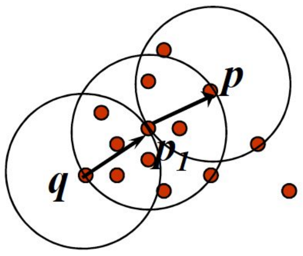
DBSCAN

2016025096 강수진

1. DBSCAN algorithm

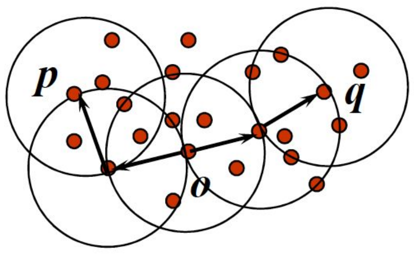
DBSCAN algorithm is a density based notion of cluster. In this clustering algorithm, cluster is defined as a maximal set of density-connected points.

cf) density-reachable



if p1(== a chain of points) is directly density-reachable from q
and
p is directly density-reachable from p1.
=> p is density-reachable from q w.r.t. eps and minpts

density-connected



if there is a point o such that

p is density reachable from o
and
q is density reachable from o

=> p and q are density reachable from o w.r.t. eps and minpts.

<DBSCAN algorithm process>

1. select a point p randomly.
2. find all points density-reachable from p w.r.t. eps and minpts.

3. if p is a core point, a cluster is formed.(core point : if the number of neighborhood of p is more than minimum points, we call this point ‘core point’)
4. if p is a border point(not core point), no points are density-reachable from p and dbscan visits the next point of the database.
5. continue the process until all of the points have been processed.

2. Code implementation

(1) Code structure

To simplify my code, I listed all classes with their member variables and method without specifying data types or return values.

```
struct objectInfo{
   int id;
  double xpos,ypos;
};

class Clustering
        private:
                int sz,n,eps,minpts;
                vector<objectInfo> obj;//objects(id, xpos, ypos)
                vector<vector<double>>dist;//distance between all pairs
                vector<bool> checker;//check objects included to cluster
                vector<vector<int>> result;//save cluster groups
        public:
                Clustering()//constructor
                run()//method for running dbscan algorithm
                calculateAllDistances()//calculate distances between all pairs
                calculateDistance()//calculate distance between two objects
                dbscan()//method for running dbscan algorithm
                discoverCluster()//form cluster recursively
                getResult()//return n clusters
```

class IO

private:

int n;
string inputFile;//input file name. ex)input1.txt
string outputFile;//part of output file name from input file. ex)input1
ifstream readFile;//input file stream
ofstream writeFile;//output file stream
vector<objectInfo>object;//objects(id, xpos, ypos)

public:

IO()//constructor
inputRead()//read input file
createOutputFile()//create result file
split()//split one input line by delimeter
getObject()//getter for objects

(2) Function description

I explained each member variables and functions(method) in details.

- Clustering class

Clustering class is for running dbscan algorithm.

member variables

```
private:
    int sz,n,eps,minpts;
    vector<objectInfo> obj;//object location
    vector<vector<double>> dist;
    vector<bool> checker;
    vector<vector<int>> result;
```

constructor

```
Clustering(int _n, int _eps, int _minpts,vector<objectInfo> object){
    sz=int(object.size());
    n=_n;
    eps=_eps;
    minpts=_minpts;
    obj=object;
    dist.resize(sz);
    for(int i=0;i<sz;i++)
        dist[i].resize(sz);
    checker.resize(sz);
}
```

This is a constructor in Clustering class.

run()

```
void run(){
    calculateAllDistances();
    dbscan();
}
```

This is the method for running dbscan algorithm. After calculating distances between all object pairs, I call dbscan method.

calculateAllDistances()

```
void calculateAllDistances(){
    for(int i=0;i<sz;i++)
        dist[i][i]=0;

    for(int i=0;i<sz;i++){
        for(int j=i+1;j<sz;j++){
            dist[i][j]=dist[j][i]=calculateDistance(obj[i],obj[j]);

        }
    }
}
```

For calculating distances between all pairs of objects, I use for loop and call ‘calculateDistance’ function for all pairs.

calculateDistance(objectInfo p1, objectInfo p2)

```
double calculateDistance(objectInfo p1, objectInfo p2){
    return sqrt(pow(p1.xpos-p2.xpos,2)+pow(p1.ypos-p2.ypos,2));
}
```

This is the method for calculate distance between one pair of objects. I used sqrt and pow function to calculate distance.

dbscan()

```
void dbscan(){
    for(int i=0;i<sz;i++){
        if(checker[i])continue;
        vector<int>cluster={};
        discoverCluster(i,cluster);
        if(cluster.size()>1)
            result.push_back(cluster);
    }
}
```

This is the method for dbscan algorithm. If objects are not included in cluster, call discoverCluster() to form new cluster. If new cluster size is bigger than 1, add that to result vector since it is not outlier.

getResult()

```
vector<vector<int>> getResult(){
    if(result.size()>n){
        sort(result.begin(),result.end(),[](vector<int>a, vector<int>b){
            return a.size()>b.size();
        });
        while(result.size()>n)
            result.pop_back();
    }
    return result;
}
```

Sort all the clusters based on their size and get n clusters.

- IO class

IO class has two roles of reading input file and writing output file.

member variable

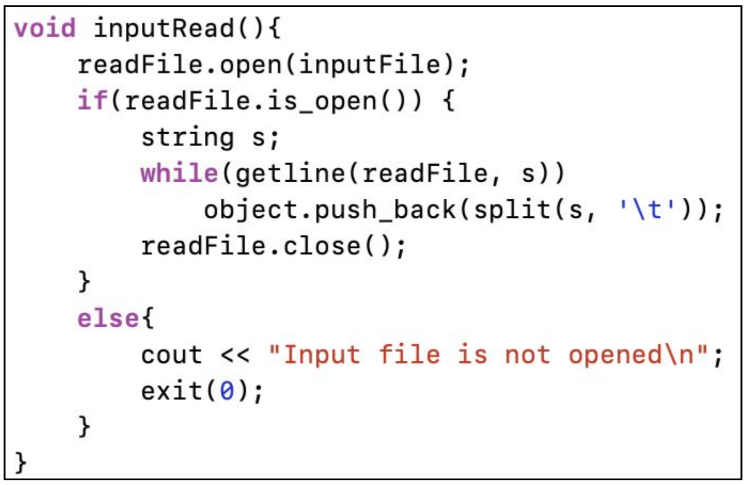
```
private:
    int n;
    string inputFile;
    string outputFile;
    ifstream readFile;
    ofstream writeFile;
    vector<objectInfo>object;
```

constructor

```
IO(int _n, string input){
    n=_n;
    inputFile=input;
    string str=inputFile;
    stringstream ss(str);
    string tmp;
    getline(ss,tmp,'.');
    outputFile=tmp;
}
```

This is a constructor for io class.
If inputFile is ‘input1.txt’, outputFile is ‘input1’. When I generate output file, I use outputFile variable to make file names.

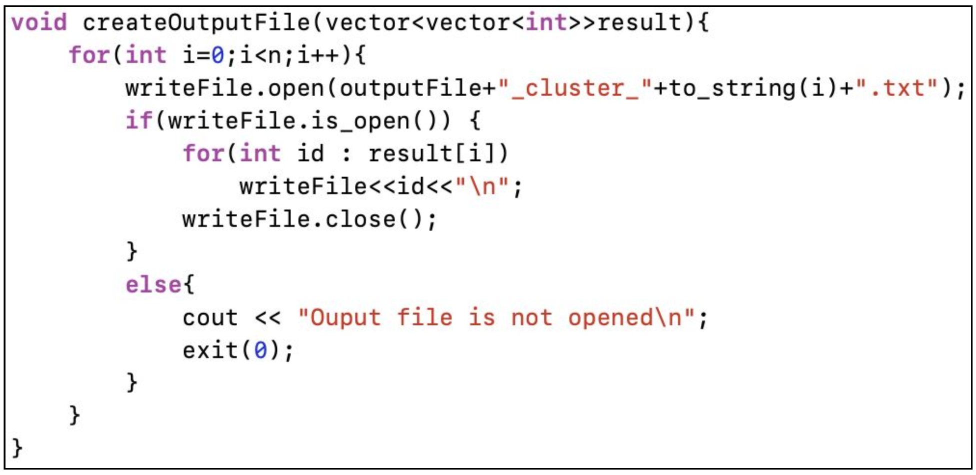
inputRead()



```
void inputRead(){
    readFile.open(inputFile);
    if(readFile.is_open()) {
        string s;
        while(getline(readFile, s))
            object.push_back(split(s, '\t'));
        readFile.close();
    }
    else{
        cout << "Input file is not opened\n";
        exit(0);
    }
}
```

This is the method for reading input file - object locations.

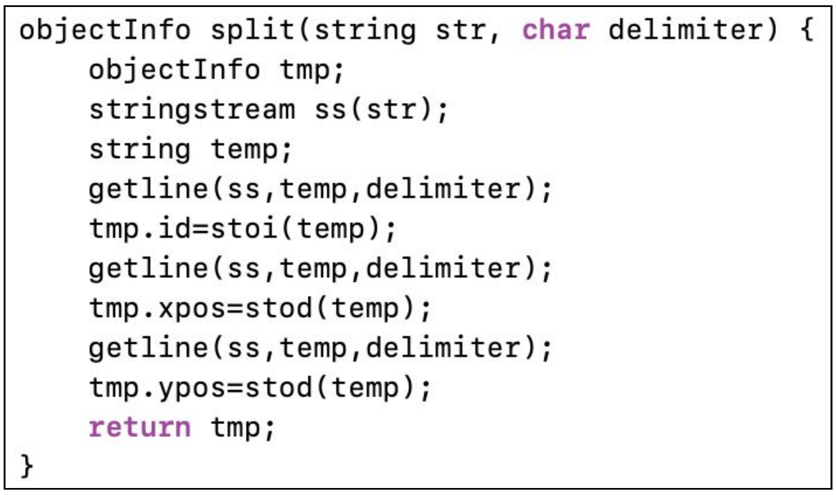
createOutputFile()



```
void createOutputFile(vector<vector<int>>result){
    for(int i=0;i<n;i++){
        writeFile.open(outputFile+"_cluster_"+to_string(i)+".txt");
        if(writeFile.is_open()) {
            for(int id : result[i])
                writeFile<<id<<"\n";
            writeFile.close();
        }
        else{
            cout << "Ouput file is not opened\n";
            exit(0);
        }
    }
}
```

This is the method for writing output file.

split()



```
objectInfo split(string str, char delimiter) {
    objectInfo tmp;
    stringstream ss(str);
    string temp;
    getline(ss,temp,delimiter);
    tmp.id=stoi(temp);
    getline(ss,temp,delimiter);
    tmp.xpos=stod(temp);
    getline(ss,temp,delimiter);
    tmp.ypos=stod(temp);
    return tmp;
}
```

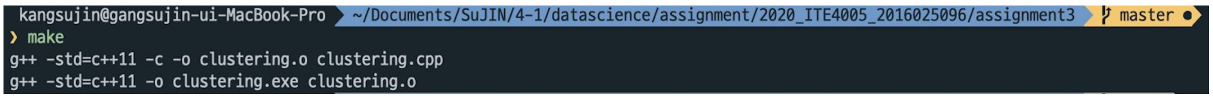
To get object id and location, I added split method to split each line of input file.

3. Instructions for compilation

Environment
OS : Mac OS
Language : C++

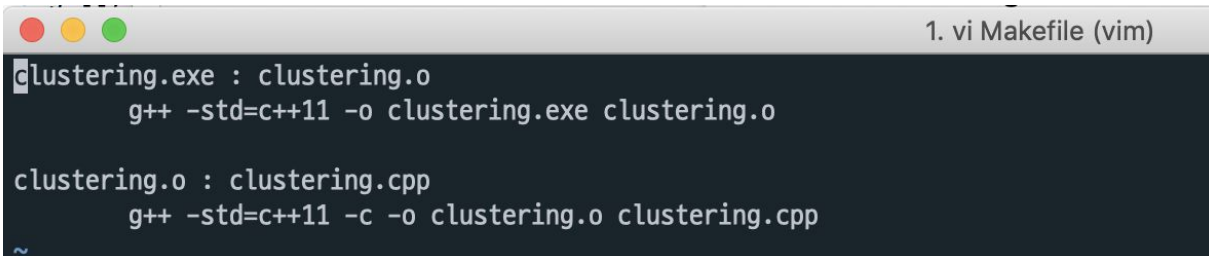
I created Makefile and followed two steps for compilation below.

1. command : make



```
kangsujin@gangsujin-ui-MacBook-Pro  ~/Documents/SuJIN/4-1/datascience/assignment/2020_ITE4005_2016025096/assignment3  master ●
❯ make
g++ -std=c++11 -c -o clustering.o clustering.cpp
g++ -std=c++11 -o clustering.exe clustering.o
```

in Makefile, two commands are written.



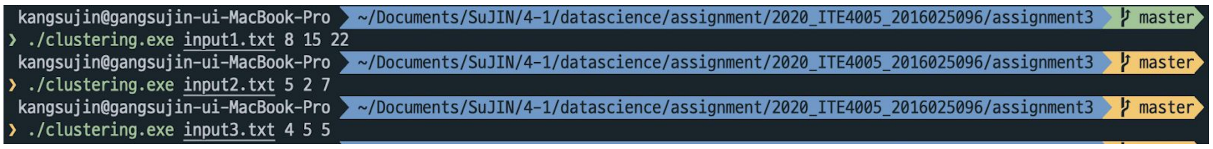
```
1. vi Makefile (vim)
clustering.exe : clustering.o
        g++ -std=c++11 -o clustering.exe clustering.o

clustering.o : clustering.cpp
        g++ -std=c++11 -c -o clustering.o clustering.cpp
~
```

g++ -std=c++11 -c -o clustering.o clustering.cpp
g++ -std=c++11 -o clustering.exe clustering.o

=> clustering.o, clustering.exe files are created by using ‘make’ command

2. command : ./clustering.exe input1.txt 8 15 22
   ./clustering.exe input2.txt 5 2 7
   ./clustering.exe input3.txt 4 5 5

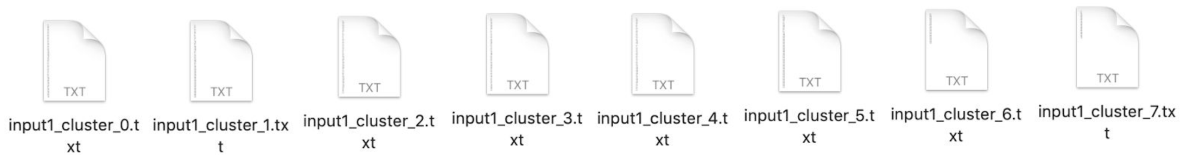


```
kangsujin@gangsujin-ui-MacBook-Pro  ~/Documents/SuJIN/4-1/datascience/assignment/2020_ITE4005_2016025096/assignment3  master
❯ ./clustering.exe input1.txt 8 15 22
kangsujin@gangsujin-ui-MacBook-Pro  ~/Documents/SuJIN/4-1/datascience/assignment/2020_ITE4005_2016025096/assignment3  master
❯ ./clustering.exe input2.txt 5 2 7
kangsujin@gangsujin-ui-MacBook-Pro  ~/Documents/SuJIN/4-1/datascience/assignment/2020_ITE4005_2016025096/assignment3  master
❯ ./clustering.exe input3.txt 4 5 5
```

When you use these commands, you have to put 1) clustering.exe, 2)input files(input1.txt, input2.txt, input3.txt) in same folder.
=> result files are created.

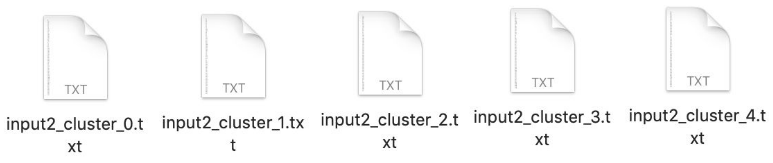
4. Result

For input1.txt,



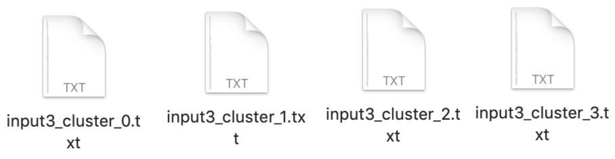
input1_cluster_0.txt input1_cluster_1.txt input1_cluster_2.txt input1_cluster_3.txt input1_cluster_4.txt input1_cluster_5.txt input1_cluster_6.txt input1_cluster_7.txt

For input2.txt,



input2_cluster_0.txt input2_cluster_1.txt input2_cluster_2.txt input2_cluster_3.txt input2_cluster_4.txt

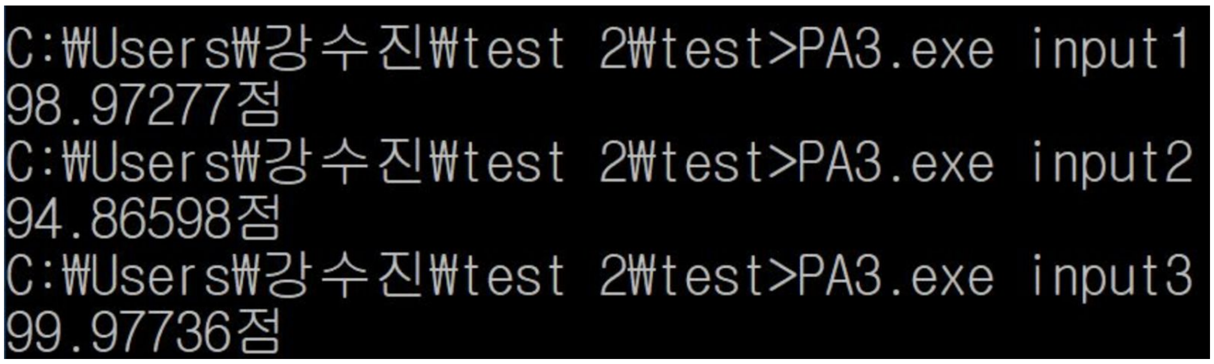
For input3.txt,



input3_cluster_0.txt input3_cluster_1.txt input3_cluster_2.txt input3_cluster_3.txt

Result files are created like above.

By using windows testing program, I got the result below.
input1 : 98.97277점
input2 : 94.86598점
input3 : 99.97736점



```
C:₩Users₩강수진₩test 2₩test>PA3.exe input1
98.97277점
C:₩Users₩강수진₩test 2₩test>PA3.exe input2
94.86598점
C:₩Users₩강수진₩test 2₩test>PA3.exe input3
99.97736점
```