**Demonstration of K-means**

***201133216***

***정유석***

**CODE**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#define MIN\_POINTS 30

//Centroid struct

typedef struct{

int pre\_N;

int Next\_N;

float pre\_X,pre\_Y;

float avg\_X,avg\_Y;

}Centroid;

//Point struct and it has centroid

typedef struct{

float x,y;

int centroid;

}Point;

//number is used to display

int Number = 0;

//Make centeroid and point

Centroid \*center;

Point \*p;

//iteration function -- it has recursive structure. So continue the function when it find number of centroid preview point and Next point

int iteration(int N, int C){

int i,j;

int\* distance; //It finds least distance

int check = 0;

float cal\_X = 0.0, cal\_Y = 0.0 ;// It used to set centroid position

distance = (int \*)malloc(sizeof(int)\*C); //It allocation array of each distance

for(i=0; i<C; i++) //Check the preveious Points and Next points

if(center[i].pre\_X == center[i].avg\_X && center[i].pre\_Y == center[i].avg\_Y)

check++;

if(check == C){ //If it same the check and C , Print and Exit

for(i=0; i<C; i++){

printf("Centroid %d-------------------------\n",i+1);

for(j=0; j<N; j++)

if(p[j].centroid == i)

printf("Centroid %d : ( %f , %f )\n",i+1,p[j].x,p[j].y);

}

return 1; //Return 1 means exit the program (exit recursive)

}

else{

printf("\n\n");

for(i=0; i<C; i++){ //initialize

center[i].Next\_N = 0;

center[i].pre\_N = 0;

center[i].pre\_X = center[i].avg\_X;

center[i].pre\_Y = center[i].avg\_Y;

}

for(i=0; i<N; i++){

int min = 10000; //initialize

int min\_Index;

for(j=0; j<C; j++){ //find least distance

distance[j] = sqrt((p[i].x-center[j].avg\_X)\*(p[i].x-center[j].avg\_X) + (p[i].y-center[j].avg\_Y)\*(p[i].y-center[j].avg\_Y));

if(min > distance[j]){ //re initialize

min = distance[j];

min\_Index = j;

p[i].centroid = j;//set the centroid of point

}

}

//Increase number of centroid

center[min\_Index].pre\_N++;

}

for(i=0; i<C; i++){

int cnt = 0;

//Using find next centroid

cal\_X = 0;

cal\_Y = 0;

//Set average points in each centroid and reset each centroid point

for(j=0; j<N; j++){

if(i == p[j].centroid){

cal\_X = cal\_X + p[j].x;

cal\_Y = cal\_Y + p[j].y;

cnt++;

}

}

center[i].avg\_X = (float)cal\_X / cnt;

center[i].avg\_Y = (float)cal\_Y / cnt;

//Display each centroid point

printf("%d centeroid %d : (%f,%f)\n",Number, i+1, center[i].avg\_X, center[i].avg\_Y);

}

Number++; //Using display

for(i=0; i<N; i++){

int min = 100;

int min\_Index;

for(j=0; j<C; j++){ //Find next points in centroid

distance[j] = sqrt((p[i].x-center[j].avg\_X)\*(p[i].x-center[j].avg\_X) + (p[i].y-center[j].avg\_Y)\*(p[i].y-center[j].avg\_Y));

if(min > distance[j]){

min = distance[j];

min\_Index = j;

p[i].centroid = j;

}

}

center[min\_Index].Next\_N++;

}

//recursive

iteration(N,C);

}

}

void main(){

int N=10,i,C=10;

int cnt;

//receive number of points

while(N < MIN\_POINTS){

scanf("%d",&N);

fflush(stdin);

}

//allocation point in N

p = (Point \*)malloc(sizeof(Point) \*N);

//receive number of Centroids

while(C>9){

scanf("%d",&C);

fflush(stdin);

}

//allocation centroid in center

center = (Centroid \*)malloc(sizeof(Centroid) \* C);

//Receive points

for(i=0; i<N; i++){

scanf("%f %f",&p[i].x,&p[i].y);

fflush(stdin);

}

printf("\n\n");

for(i=0; i<C; i++){

int rnd = rand()%N;

//Set first centroid

center[i].avg\_X = p[rnd].x;

center[i].avg\_Y = p[rnd].y;

center[i].pre\_N = 0;

center[i].Next\_N = 1;

center[i].pre\_X = 0;

center[i].pre\_Y = 0;

printf("%d centeroid %d : (%f,%f)\n",Number,i+1, center[i].avg\_X, center[i].avg\_Y);

}

Number++;

//Start iteration

cnt = iteration(N, C);

//Display End

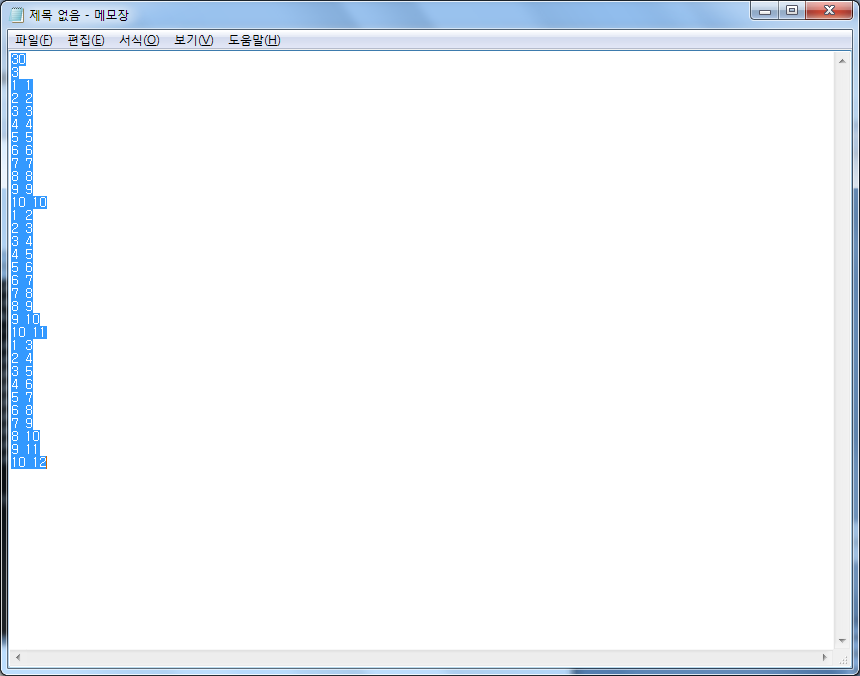
if(cnt == 1)

printf("\n--End--\n");

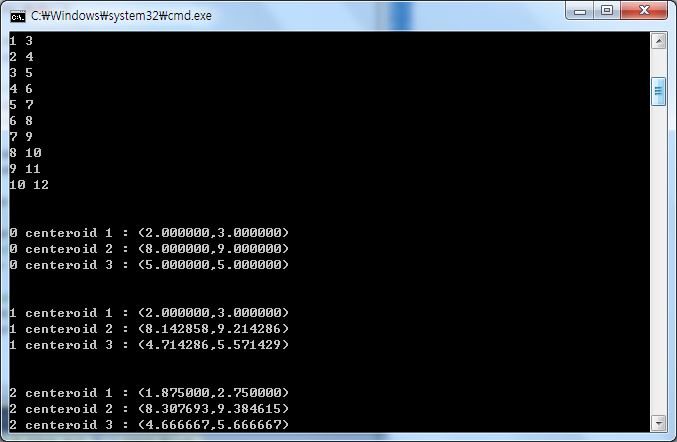
}

**DISPLAY**

* Points : 30
* Centroid : 3



* Display Each postion of centroids



* Each points in centroid

