Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

Taxi!
Projet intégrateur
(32%)
CSI2520 Paradigmes de Programmation

Hiver 2022

Ce projet est à réaliser de façon individuelle

Partie 1 due le 7 Février avant 23:00

Partie 2 due le 28 Février avant 23:00

Partie 3 et 4 dues le 8 Avril avant 23:00

Pénalité de retard : perte de 1% de la note obtenue par heure de retard

Description du problème

In this comprehensive assignment, you will implement a data clustering algorithm named DBSCAN - *Density-Based Spatial Clustering of Applications with Noise. Given a large set of data points in a space of arbitrary dimension and given a distance metric, this algorithm can discover clusters of different shapes and sizes, marking as outliers isolated points in low-density regions (i.e. points whose nearest neighbors are too far away).*

The DBSCAN algorithm uses two parameters:

- ***minPts:*** *The minimum number of points (a threshold) in the neighborhood of a point for this one to be considered to belong to a dense region.*
- ***eps (ε):*** *A distance measure that is used to identify the points in the neighborhood of any point.*

Ce projet intégrateur vous demande de programmer un algorithme de groupement de données appelé DBSCAN. Soit un ensemble de points dans un espace de dimension quelconque et soit une métrique de distance, cet algorithme permet de trouver les groupements de points de dimension et forme diverses tout en identifiant les points isolés n'appartement à aucun groupement (i.e. les points pour lesquels leurs plus proches voisins sont éloignés).

L'algorithme DBSCAN utilise deux paramètres :

- **minPts:** le nombre minimum de voisins qu'un point doit avoir pour être considéré être dans une région dense.
- **eps (ε):** la distance permettant d'identifier les points appartenant au voisinage d'un point donné.

DBSCAN Algo:

```
DBSCAN(DB, distFunc, eps, minPts) {
    C := 0                                                  /* Cluster counter */
    for each point P in database DB {
        if label(P) ≠ undefined then continue               /* Previously processed in inner loop */
        Neighbors N := RangeQuery(DB, distFunc, P, eps)     /* Find neighbors */
        if |N| < minPts then {                              /* Density check */
            label(P) := Noise                               /* Label as Noise */
            continue
        }
        C := C + 1                                          /* next cluster label */
        label(P) := C                                       /* Label initial point */
        SeedSet S := N \ {P}                                /* Neighbors to expand */
        for each point Q in S {                             /* Process every seed point Q */
            if label(Q) = Noise then label(Q) := C          /* Change Noise to border point */
            if label(Q) ≠ undefined then continue           /* Previously processed */
            label(Q) := C                                   /* Label neighbor */
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) /* Find neighbors */
            if |N| ≥ minPts then {                          /* Density check (if Q is a core point) */
                S := S ∪ N                                  /* Add new neighbors to seed set */
            }
        }
    }
}

RangeQuery(DB, distFunc, Q, eps) {
    Neighbors N := empty list
    for each point P in database DB {                       /* Scan all points in the database */
        if distFunc(Q, P) ≤ eps then {                      /* Compute distance and check epsilon */
            N := N ∪ {P}                                    /* Add to result */
        }
    }
    return N
}

/* Reference: https://en.wikipedia.org/wiki/DBSCAN */
```

Problème à résoudre

You are managing a taxi fleet in NYC and you would like to identify the best waiting areas for your vehicles. To solve this problem, you have at your disposal a large dataset of taxi trip records; more specifically you will use the 2009 records available here.

Each record of this dataset includes the GPS coordinates of the starting point and end point for the corresponding trip (note that for the most recent years, locations are specified by zone number which is not useful for us). Since we want to identify the best waiting area, we are interested by the starting points. The dataset is contained in a simple comma-separated value file (csv), each line corresponding to a trip record. The different columns correspond to the attributes of each trip and are named as follows (if needed, more details can be found in the NYC taxi trip records website):

```
vendor_name,Trip_Pickup_DateTime,Trip_Dropoff_DateTime,Passenger_Count,Trip_Distance,
Start_Lon,Start_Lat,Rate_Code,store_and_forward,End_Lon,End_Lat,Payment_Type,
Fare_Amt,surcharge,mta_tax,Tip_Amt,Tolls_Amt,Total_Amt
```

We therefore ask you to use the DB-SCAN algorithm in order to cluster the starting point locations of the trip records in the NYC 2009 taxi dataset. The centers of the largest clusters will become the waiting area for your taxi fleet. Note that in order to accurately apply the DBSCAN algorithm, the distance between two GPS locations should be measured in meters. However, as a simplification and because the GPS locations are all located in a relatively small area, you can simply use the Euclidean distance between the GPS coordinates.

Vous êtes responsable d'une flotte de taxis dans la ville de New York et vous désirez donc identifier les meilleurs points d'attente pour vos véhicules. Afin de résoudre ce problème, vous avez à votre disposition une large base de données contenant l'information à propos de toutes les courses de taxi effectuées en 2009. Cette base de données est disponible ici.

Chacun des enregistrements de cette base de données inclut les coordonnées GPS des points de départ et d'arrivée de la course correspondante (à noter que pour les années plus récentes, les coordonnées GPS ont été remplacées par des numéros de zone, ce qui n'est pas utile pour nous). Dans notre cas, puisque nous désirons identifier les aires d'attente, nous sommes intéressés par les points de départ. La base de données est sauvegardée dans un simple fichier csv (*comma separated values*), chaque ligne correspondant à une course. Les différentes colonnes correspondent à différents attributs ainsi nommés (au besoin, consulter le site web pour plus d'explications sur chacun de ces attributs):

```
vendor_name,Trip_Pickup_DateTime,Trip_Dropoff_DateTime,Passenger_Count,Trip_Distance,
Start_Lon,Start_Lat,Rate_Code,store_and_forward,End_Lon,End_Lat,Payment_Type,
Fare_Amt,surcharge,mta_tax,Tip_Amt,Tolls_Amt,Total_Amt
```

Nous vous demandons donc d'utiliser l'algorithme DBSCAN afin de grouper les points de départ des enregistrements contenus dans la base de données NYC taxi 2009. Les centres des groupes les plus grands deviendront les aires d'attente pour votre flotte de taxi. A noter que pour appliquer de façon précise l'algorithme de groupement, la distance entre deux points de départs devrait être mesurée en mètres. Toutefois, afin de simplifier le problème et parce que les points GPS se trouvent tous à l'intérieur d'une région géographique relativement petite, nous utiliserons simplement la distance Euclidienne entre les coordonnées GPS.

Programmation

You have to write programs under different paradigms that solve different versions of this problem. You will receive additional instructions for each language.

Each program will be marked as follows:

Program produces the correct value	*[1.5 points]*
Program produces the correct set of items	*[1.5 points]*
Adherence to programming paradigm	*[3 points]*
Quality of programming (structures, organisation, etc)	*[1 point]*
Quality of documentation (comments and documents)	*[1 point]*

All your files must include a header showing student name and number. These files must be submitted in a zip file.

Vous devez écrire une série de programmes sous les différents paradigmes de programmation afin de résoudre différentes versions du problème proposé. Vous recevrez des instructions additionnelles spécifiques à chacun des langages.

Chaque programme sera corrigé en suivant la grille suivante :

Le programme produit la valeur maximale correcte	[1.5 points]
La programme produit correctement l'ensemble des items	[1.5 points]
Adhérence au paradigme de programmation	[3 points]
Qualité de la programmation (structure, organisation, etc)	[1 point]
Qualité de la documentation (commentaires, documents)	[1 point]

Tous vos fichiers doivent inclure un entête incluant votre nom et numéro d'étudiant. Tous ces fichiers doivent être soumis dans un fichier zip.