ENSAE TD noté, mercredi 8 décembre 2010

Le programme construit au fur et à mesure des questions devra être imprimé à la fin du TD et rendu au chargé de TD. Il ne faut pas oublier de mentionner son nom en commentaires au début du programme et l'ajouter sur chaque page. Les réponses autres que des parties de programme seront insérées sous forme de commentaires. Les définitions de fonctions proposées ne sont que des suggestions.

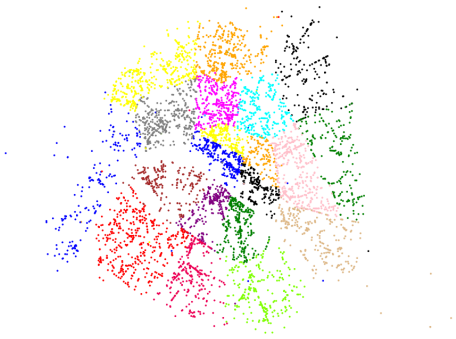
Lors de cette séance, on s'intéresse aux restaurants de Paris et à leur répartition selon les arrondissements. Ils sont décrits dans le fichier accessible à l'adresse : http://www.xavierdupre.fr/enseignement/examen_python/restaurant_paris.txt. Ce fichier contient trois colonnes : l'arrondissement, la longitude, la latitude. On supposera que Paris est suffisamment petit pour que la longitude et la latitude soient considérées comme des coordonnées cartésiennes.

1) La première question consiste à télécharger le fichier puis à récupérer ses informations via la fonction `def lit_fichier`. Cette fonction retourne une matrice (une liste de listes). (2 points)

2) On souhaite ensuite compter le nombre de restaurants par arrondissement pour déterminer celui qui en contient le plus et celui qui en contient le moins. On écrit une seule fonction `def compte_restaurant(mat)` qui retourne un dictionnaire contenant le décompte des restaurants par arrondissement. (3 points)

3) Certains arrondissements sont plus grands que d'autres et on aimerait mesurer la densité des restaurants par arrondissement plutôt que leur nombre. Cependant il n'est pas évident de deviner la superficie d'un arrondissement à partir des seuls restaurants. On construit donc l'approximation suivante :

1. Chaque arrondissement est un cercle.
2. Le centre de ce cercle est le barycentre de tous les restaurants qu'il contient.
3. Le rayon est la distance séparant le barycentre du premier restaurant en dehors de l'arrondissement.
4. La superficie de l'arrondissement est alors πR^2.



On cherche d'abord à calculer le barycentre de chaque arrondissement. On constuit la fonction `def barycentre(mat)` qui retourne un dictionnaire associant un couple de points à chaque arrondissement. (4 points) **Cette question vaut 5 points si le coût de l'algorithme est optimal.**

4) On cherche maintenant à connaître le plus proche restaurant en dehors de l'arrondissement. Ecrire une fonction qui calcule la distance euclidienne entre deux points : `def distance(x1, y1, x2, y2)`. (2 points)

5) Pour un arrondissement, écrire une fonction `def plus_proche_restaurant(x, y, arr, mat)` qui retourne le restaurant le plus proche du point `(x, y)` en dehors de l'arrondissement `arr`. (4 points)

6) Ecrire une dernière fonction qui retourne sous forme de dictionnaire la densité approchée pour chaque arrondissement : `def densite_approchee(mat)`. Quelles sont les arrondissements le plus et le moins dense ? Commentez les résultats et proposez des améliorations ? (4 points)

```
75002    2.407478    48.930141        75010    2.405963    48.921033
75002    2.400573    48.913487        75018    2.391144    48.934509
```

Correction

```
# coding: latin-1
import urllib2, math

# question 1
def lit_fichier () :
    # le principe est le même que pour le chargement d'un fichier
    # le programme lit directement les informations depuis Internet
    f = urllib2.urlopen ("http://www.xavierdupre.fr/enseignement"\
                         "/examen_python/restaurant_paris.txt")
    s = f.read ()
    f.close ()
    lines = s.split ("\n")  # on découpe en lignes
    # on découpe en colonnes
    lines = [ _.strip ("\n\r ").split ("\t") for _ in lines if len (_) > 0 ]
    lines = [ _ for _ in lines if len (_) == 3 ]  # on supprime les lignes vides
    # on convertit les coordonnées en réel
    lines = [ (a [3:], float (b), float (c)) for a,b,c in lines ]
    return lines

# question 2
def compte_restaurant (mat) :
    # simple comptage, voir le chapitre 3...
    compte = { }
    for cp,x,y in mat :
        if cp not in compte : compte [cp] = 0
        compte [cp] += 1
    return compte

# question 3
def barycentre (mat) :
    # un barycentre est un point (X,Y)
    # où X et Y sont respectivement la moyenne des X et des Y
    barycentre = { }
    # boucle sur la matrice
    for cp,x,y in mat :
        if cp not in barycentre : barycentre [cp] = [ 0, 0.0, 0.0 ]
        a,b,c           = barycentre [cp]
        barycentre [cp] = [a+1, b+x, c+y]
    # boucle sur les barycentres
    for cp in barycentre :
        a,b,c = barycentre [cp]
        barycentre [cp] = [b/a, c/a]

    # le coût de cette fonction est en O (n log k)
    # où k est le nombre de barycentre
    # de nombreux élèves ont deux boucles imbriquées,
    # d'abord sur la matrice, ensuite sur les barycentres
    # ce qui donne un coût en O (nk), beaucoup plus grand
    return barycentre

# question 4
def distance (x1, y1, x2, y2) :
    return ((x1-x2)**2 + (y1-y2)**2)**0.5

# question 5
def plus_proche_restaurant (x,y, arr, mat) :
    m,mx,my = None, None, None
```

```
    for cp,a,b in mat :
        if cp != arr and (m == None or distance (a,b,x,y) < m) :
            mx,my = a,b
            m     = distance (a,b,x,y)
    return mx,my

# question 6
def densite_approchee (mat) :
    g      = barycentre (mat)
    compte = compte_restaurant (mat)
    res    = { }

    for cp in g :
        out  = plus_proche_restaurant (g [cp][0], g [cp][1], cp, mat)
        r    = distance (g [cp][0], g [cp][1], out [0], out [1])
        aire = math.pi * r ** 2
        res [cp] = compte [cp] / aire

    return res

if __name__ == "__main__" :

    if False :  #mettre à vrai pour remplacer la fonction plus_proche_restaurant
        # ajout par rapport à l'énoncé
        # en réponse à la dernière question
        # plutôt que de prendre le premier point à hors de l'arrondissement
        # on considère celui correspondant à un quantile (5%)
        # ce qui évite les quelques restaurants dont les données
        #sont erronées
        def plus_proche_restaurant_avec_amelioration (x,y, arr, mat) :
            all     = []
            for cp,a,b in mat :
                if cp != arr :
                    m = distance (a,b,x,y)
                    all.append ( (m,a,b))
            all.sort ()
            a,b = all [len(all)/20][1:]
            return a,b

        # ajout par rapport à l'énoncé
        plus_proche_restaurant = plus_proche_restaurant_avec_amelioration

    mat = lit_fichier ()
    com = densite_approchee (mat)
    ret = [ (v,k) for k,v in com.iteritems () ]
    ret.sort ()
    for a,b in ret : print "%d\t%s" % (a,b)

    # ajout par rapport à l'énoncé
    # permet de dessiner les restaurants, une couleur par arrondissement
    # on observe que certains points sont aberrants, ce qui réduit d'autant
    # l'estimation du rayon d'un arrondissement (il suffit qu'un restaurant
    # étiquetés dans le 15ème soit situé près du barycentre du 14ème.)
    import matplotlib
    import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.mlab as mlab
    import matplotlib.cbook as cboo
```

```
fig = plt.figure()
ax = fig.add_subplot(111)
colors = [  'red', 'blue', 'yellow', 'orange', 'black', 'green',
            'purple', 'brown', 'gray', 'magenta', 'cyan', 'pink', 'burlywood',
            'chartreuse', '#ee0055']
for cp in barycentre (mat) :
    lx = [ m[1] for m in mat if m [0] == cp ]
    ly = [ m[2] for m in mat if m [0] == cp ]
    c  = colors [ int(cp) % len (colors) ]
    #if cp not in ["02", "20"] : continue
    ax.scatter(lx,ly, s = 5., c=c,edgecolors = 'none'  )
plt.show ()
```