Bramas Timothé

MPSI2

DS n°9 :

Question 1 :

```
SELECT idpatient FROM MEDICAL WHERE
etat = " hernie discale "
```

Question 2 :

```
SELECT nom, prenom FROM PATIENT,
MEDICAL WHERE PATIENT.id = idpatient
AND etat = " spondylolisthesis "
```

Question 3 :

```
SELECT etat, count(*) FROM MEDICAL
GROUP by etat
```

Question 4 :

cette bibliothèque est pratique puisqu'il y a déjà les opérations matricielles de base, cela évite de devoir les programmer.

Question 5 :

avec $N = 100\,000$, le tableau est de taille
$100\,000 \times 6 = 600\,000$ données

chacune est sur 32 bits ;

le tableau prend $600\,000 \times \frac{32}{8} = 2\,400\,000$ o

$= 2,4$ Mo.

le vecteur prend lui $100\,000 \times \frac{8}{8} = 0,1$ Mo.

ainsi les deux prennent une taille de 2,5 Mo.

Question 6 :

```
def separationParGroupe(data, etat):
    liste = [[], [], []]
    for i in range(numpy.size(etat)):
        liste[etat[i]] = liste[etat[i]] + data[i]
    return liste
```

Question 8 :

Les diagrammes de la diagonale servent à avoir le nombre de patients en fonction de l'indice de leur bassin, de l'orientation, etc.

Les diagrammes en dehors de la diagonale permettent d'avoir les données i et j en fonction d'une donnée k.

Question 9 :

$$x_{norm_j} = \frac{x_j - \min(X)}{\max(X) - \min(X)}$$

on a bien le min normalisé à 0, le max à 1 :
et $\forall x_j \in [\min(X), \max(X)]$, $x_{norm_j} \in [0; 1]$.

Question 10 :

```
def min_max(X):
    min = X[0]
    max = X[0]
    for i in range(numpy.size(X)):
        if X[i] < min:
            min = X[i]
        if X[i] > max:
            max = X[i]
    return min, max
```

avec n la taille du vecteur, la boucle fait n itérations, avec à chaque fois au plus 4 opérations. la complexité est donc en $\mathcal{O}(4n) = \mathcal{O}(n)$:

c'est linéaire.

Question 11:

avec $A, B \in \mathbb{R}^n$, $n \in \mathbb{N}$, la distance entre a et b vaut $\sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$ en notant $A = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}, B = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$

```
def distance (z, data):
    n = numpy.size(data)
    liste = [ ]
    for i in range (n):
        s = 0
        for j in range (numpy.size(data[0])):
            s = s + (z[j] - data[i][j])**2
        liste = liste + numpy.sqrt(s)
```

Question 12.

des lignes 3 à 7: création du tableau T

10 à 12: création et incrémentation de select.

15 à 21: récupération des k données les plus proches.

T est un tableau de couples (dist[i] ; i) représentant la position d'un élément de z et sa distance.

dist est la liste des distances entre les n-uplets du tableau et le n-uplet z.

select est un vecteur à 3 coordonnées donnant à la i-ème donnée ($i \in [0;2]$) le nombre de personnes dans l'état i.

Question 13 :

L'élément $K_{i,i}$ de la diagonale donne le nombre de patients ayant été estimés dans l'état i et l'étant réellement.

La ligne 0 contient les estimations des patients dans l'état 0 ; 23 ont été estimés dans cet état, 4 dans l'état 1 et 4 dans l'état 2.

La colonne 0 contient les état des patients estimés dans l'état 0 : 23 ont l'état 0, 7 l'état 1 et 5 le 2.

cette matrice permet d'avoir le taux de réussite de l'algorithme.

Question 14 :

L'efficacité est contenue entre 69 et 74%, c'est assez faible. Elle semble atteindre son

max pour K=9 ou 11 ; il faudrait privilégier ces valeurs.

Question 15 :

```
def moyenne(X):
    n = numpy.size(X)
    S = 0
    for i in range(n):
        S = S + X[i]
    return S/n,

def variance(X):
    moy = moyenne(X)
    n = numpy.size(X)
    S = 0
    for i in range(n):
        S = S + (X[i] - moy)**2
    return (S/n)
```