

GRAS  
Romyin

DS Infa:  
Cycle 4:

Q.1:

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

Q.2:

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

Q.3:

```
SELECT etat, COUNT(idpatient) FROM MEDICAL GROUP BY  
etat
```



Q.4:

Un intérêt: On peut faire des opérations comme  $+ \text{ ou } *$  entre tableaux.

Q.5:

D'après l'énoncé: Chaque entier est codé sur 32 bits, soit 4 octets.

Et on a:  $N \times n$  informations.

Ainsi:  $\text{mémoire}_{\text{data}} = 4 \times N \times n = 4 \times 6 \times 100\,000 = 2,4 \times 10^6$  octets.

Donc  $\text{mémoire}_{\text{data}} = 2,4 \text{ Mo}$

$\text{mémoire}_{\text{etat}} = 10000 \times 1 = 0,1 \text{ Mo}$

Q.6:

import sys  
def separationParGroupe(data, etat):  
 conn = sys.argv[3].connect

def separationParGroupe(data, etat):

for i in range(len(data)):

Norm = [ ]

Horn = [ ]

Spon = [ ]



```

for i in range(len(data)):
    if etat[i] == 0:
        Norm.append(etat[i] data[i])
    elif etat[i] == 1:
        Herm.append(etat[i] data[i])
    else:
        Spon.append(data[i])

return Norm, Herm, Spon.

```

Q.7:

TEST est  $i=j$   
 ARGS1 est  $i, j, 6^{*}i+j$   
 ARGS2 est  
 ARGS3 est

Q.8:

Les diagrammes hors-diagonale permettent de voir si un état ~~est~~ est relié à des caractéristiques physiques.  
 Les diagrammes de la diagonale permettent de voir une norme moyenne d'un paramètre physique et leurs répartition.

Q.9:

$$x_{\text{norm}j} = \frac{x_j - \min(X)}{\max(X) - \min(X)}$$



Q.10:

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

Q.11:

```
def distance(z, data):  
    Distances_ligne_i = []  
    Distances = []  
    for i in range(len(data)):  
        for j in range(len(data[i])):  
            Distance_ligne_i = Distance_ligne_i.append(abs(z - data[i][j]))  
        Distances.append(Distance_ligne_i)  
    return Distances
```

Q.12:

partie 1: trie les listes de distances avec leur ligne i (correspond à l'œil du patient)  
partie 2: ~~forme un tableau composé de 12 de 0, avec dans la 4<sup>ème</sup> colonne~~  
~~les patients un 1 au patient le plus proche.~~  
~~partie 3:~~



partie 2: forme un tableau d'entiers positifs

Partie 3: Trouver ~~le~~ le numéro de la ligne tel que l'état du patient  $i$  soit le plus proche de l'état 3 (avec ~~respect~~  $\text{res} = \text{mal}(\text{état})$ )

$T$  concatène les distances du patient  $i$  à  $\pi_3$  et le numéro  $i$  du patient

Q.13:

On a donc: en colonne ~~la~~ l'état d'un patient et en ligne

Pour chaque patient, on en case en colonne son état et en ligne l'état prédit par la machine. On incrémente de 1 la case alors sélectionnée.

En observant la diagonale, on voit que dans la plupart des cas la machine a mis en coefs de la diagonale plus grands que les autres.

En observant la première colonne, on voit que 23 personnes d'état normal ont bien été mises comme normales, et 12 avec une maladie.

En observant la première ligne, on voit que 12 personnes ont été notées normales mais avaient une maladie.

Cette matrice sert à prouver la validité du programme.



Q. 14: l'algorithme a une efficacité qui tourne autour de 72%, ce qui est bien, mais pas assez révélateur. On a un taux d'échec de plus de 25%, l'algorithme ne ~~peut pas être~~ n'est pas sûr.

Q. 15:

```
def moyenne(x):  
    somme = 0  
    for i in range x:  
        somme = somme + i  
    return somme / len(x)
```

def ~~int~~ variance(x):

```
    somme = 0  
    for i in x:  
        somme = somme + i * - moyenne(x) ** 2  
    return somme / len(x)
```

Q. 16:

```
def synthese(data, etat):  
    m = []  
    groupes = separation Par Groupe(data, etat)  
    for i in range(len(groupe))  
        L = []  
        for j in groupe range(len(groupe[i]))  
            x = [moyenne(groupe[i][j]), variance(groupe[i][j])]   
            L.append(x)  
        m.append(L)  
    return m[0], m[1], m[2]
```