



## Concours Mathématiques et Physique, Physique et Chimie et Technologie Epreuve d'Informatique

Date : Vendredi 1 Juin 2018 Heure : 11 H Durée : 2 H Nombre de pages : 6

Barème : Problème 1 : 14 points  
Problème 2 : 6 points

### DOCUMENTS NON AUTORISES

### L'USAGE DES CALCULATRICES EST INTERDIT

### II FAUT RESPECTER IMPERATIVEMENT LES NOTATIONS DE L'ENONCE

## PROBLEME 1

Une image en niveaux de gris est un ensemble de pixels organisés en lignes et colonnes. Chaque pixel est défini par une valeur représentant un niveau de gris (ou encore ton) parmi 256 valeurs possibles entre le noir et le blanc (0 : noir, 255 : blanc).

Binariser une image consiste à la partitionner en deux parties : une partie sombre et une partie claire selon un seuil de niveau de gris.

N.B : Les parties I et II sont indépendantes.

### Partie I

Une image peut être binarisée selon un seuil optimal déterminé par une méthode de seuillage. Dans cette partie, on propose d'implémenter quelques étapes d'une méthode de seuillage qui consiste à déterminer un couple de paramètres intervenant dans le calcul du seuil optimal.

### Description de la méthode :

Soit une image en niveaux de gris représentée par une matrice  $M$  à  $n$  lignes et  $m$  colonnes, où  $M_{i,j} \in [0, 255]$  est le niveau de gris associé au pixel d'indices  $(i, j)$  ( $0 \leq i \leq n-1$  et  $0 \leq j \leq m-1$ ).

La méthode consiste à :

- Déterminer la matrice, notée  $V$ , à  $n$  lignes et  $m$  colonnes nommée matrice des moyennes des voisinages, où chaque élément  $V_{i,j}$  ( $0 \leq i \leq n-1$  et  $0 \leq j \leq m-1$ ) a pour valeur :
  - $M_{i,j}$ , pour  $i=0$  ou  $j=0$  ou  $i=n-1$  ou  $j=m-1$  ;
  - la partie entière de la moyenne des niveaux de gris de l'élément  $M_{i,j}$  et de ses 8 voisins immédiats, pour  $0 < i < n-1$  et  $0 < j < m-1$ .

- Déterminer une matrice carrée d'ordre 256, notée  $H$  et nommée matrice histogramme, où chaque élément  $H_{i,j}$  ( $0 \leq i \leq 255$  et  $0 \leq j \leq 255$ ) représente le nombre des pixels ayant un niveau de gris égal à  $i$  et une moyenne des niveaux de gris du voisinage égale à  $j$ .
- Déterminer la matrice colonne, notée  $Moy$ , définie par :

$$Moy = \begin{pmatrix} \sum_{i=0}^{255} \sum_{j=0}^{255} i \cdot H_{i,j} \\ \sum_{i=0}^{255} \sum_{j=0}^{255} j \cdot H_{i,j} \end{pmatrix}$$

- A partir de la matrice  $H$ , déterminer parmi tous les couples  $(s, t)$  possibles, le couple optimal dont la valeur de la trace de la matrice variance correspondante, ci-dessous définie, est maximale.

Le couple optimal est déterminé comme suit :

- Pour chaque couple d'indices  $(s, t)$  de  $H$  tels que  $0 \leq s < 255$  et  $0 \leq t < 255$  :

- Calcul des probabilités  $P_0$  et  $P_1$  définies par :

$$P_0 = \frac{1}{m \cdot n} \sum_{i=0}^s \sum_{j=0}^t H_{i,j} \quad P_1 = \frac{1}{m \cdot n} \sum_{i=s+1}^{255} \sum_{j=t+1}^{255} H_{i,j}$$

- Construction des matrices colonnes  $Moy_0$  et  $Moy_1$  définies par :

$$Moy_0 = \begin{pmatrix} \frac{1}{P_0} \sum_{i=0}^s \sum_{j=0}^t i \cdot H_{i,j} \\ \frac{1}{P_0} \sum_{i=0}^s \sum_{j=0}^t j \cdot H_{i,j} \end{pmatrix} \quad Moy_1 = \begin{pmatrix} \frac{1}{P_1} \sum_{i=s+1}^{255} \sum_{j=t+1}^{255} i \cdot H_{i,j} \\ \frac{1}{P_1} \sum_{i=s+1}^{255} \sum_{j=t+1}^{255} j \cdot H_{i,j} \end{pmatrix}$$

- Construction de la matrice variance  $S$  définie par :

$$S = P_0 A A^T + P_1 B B^T \quad \text{où } X^T \text{ est la transposée d'une matrice } X ;$$

$$A = (Moy_0 - Moy) \text{ et } B = (Moy_1 - Moy)$$

- Calcul de la valeur de la trace de la matrice  $S$  et mise à jour de la valeur de la trace maximale. On rappelle que la trace d'une matrice correspond à la somme des valeurs des éléments de sa diagonale principale.

### Travail demandé

**N.B :** Dans la suite, les fonctions demandées seront écrites en langage Python, en utilisant impérativement la nomenclature donnée par le tableau suivant.

Nom	Type	Description
$M$	<code>numpy.ndarray</code>	Matrice d'entiers (entre 0 et 255) représentant une image
$V$	<code>numpy.ndarray</code>	Matrice d'entiers contenant les moyennes des voisinages de $M$
$n, m$	<code>int</code>	Respectivement nombre de lignes et de colonnes de $M$ et $V$
$H$	<code>numpy.ndarray</code>	Matrice histogramme d'ordre 256 correspondant à $M$ et $V$



1. Ecrire la fonction, nommée **GenererMat**, qui prend en paramètres **n** et **m** et retourne une matrice **M** remplie par des valeurs entières aléatoires comprises entre 0 et 255.
2. Ecrire la fonction, nommée **MoyVoisinage**, qui prend en paramètre **M** et retourne **V**.
3. Ecrire la fonction, nommée **NbPixels**, qui prend en paramètres **M**, **V** et deux entiers **i** et **j**. Elle retourne le nombre d'éléments de **M** ayant  $M_{a,b} = i$  et  $V_{a,b} = j$  (avec  $0 \leq a \leq n-1$  et  $0 \leq b \leq m-1$ ).
4. Ecrire la fonction, nommée **Histogramme**, qui prend en paramètres **M** et **V** et retourne la matrice **H**.
5. Ecrire la fonction, nommée **Sigma**, qui prend en paramètres **H**, **binf1**, **binf2**, **bsup1**, **bsup2** et **comp** (paramètre ayant la valeur par défaut 0), calcule et retourne :

$$- \sum_{i=\text{binf}_1}^{\text{bsup}_1} \sum_{j=\text{binf}_2}^{\text{bsup}_2} i \cdot H_{i,j} \text{ si } \text{comp} \text{ égal à } 0 ;$$

$$- \sum_{i=\text{binf}_1}^{\text{bsup}_1} \sum_{j=\text{binf}_2}^{\text{bsup}_2} j \cdot H_{i,j} \text{ si } \text{comp} \text{ égal à } 1.$$

6. Ecrire la fonction, nommée **Seuillage**, qui à partir du paramètre **H**, retourne le couple optimal  $(s, t)$  correspondant au maximum de toutes les valeurs des traces des matrices **S**.

## Partie II

Une région est un ensemble de pixels représentant une partie ou la totalité d'une image.

L'exemple suivant illustre une image à 3 lignes et 5 colonnes où les cases hachurées forment une région.

	0	1	2	3	4
0	128	64	128	32	32
1	64	255	0	64	128
2	0	0	255	64	54

pixel de la région de coordonnées (1, 3)  
avec un ton de gris = 64

**Figure 1 :** Exemple de région d'une image et ses pixels

Dans le but de binariser une région d'une image, dans cette partie, on propose d'implémenter les classes : Pixel et Region.

### Description des classes

– Classe **Pixel** :

▪ **Attributs** :

- **posi** et **posj**, des entiers représentant les coordonnées du pixel ;
- **ton**, entier entre 0 et 255, représentant le niveau de gris du pixel.

▪ **Méthodes** :

- **\_\_init\_\_**(...) : permet d'initialiser un pixel ;
- **\_\_str\_\_**(...) : permet d'afficher un pixel selon le format `Pixel<posi,posj,ton>`.

Exemple : `px = Pixel(1, 3, 64)` est un objet, instance de la classe Pixel.

– Classe **Region** :

▪ Attributs :

- **label**, chaîne de caractères, désignant la région ;
- **dict\_pixel**, un dictionnaire où :
  - chaque clé représente un ton de gris ;
  - chaque valeur représente la liste des coordonnées des pixels ayant ce ton de gris.

*Exemple* : Le dictionnaire associé à l'instance de la classe Region donnée par l'exemple de la Figure 1 est :

```
dict_pixel = {255 : [(1,1), (2,2)], 64 : [(1,3), (2,3)], 0 : [(1,2)]}.
```

▪ Méthodes :

- **\_\_init\_\_(...)** : permet l'initialisation des attributs de la classe Region et dont le script Python est donné par :

```
def __init__(self, lab):  
    self.label = lab  
    self.dict_pixel = dict()
```

- **\_\_len\_\_(...)** : retourne la taille de la région, en nombre de pixels. Pour l'exemple de région de la Figure 1, cette méthode retourne 5.
- **\_\_call\_\_(...)** : retourne la liste des coordonnées des pixels de la région ayant un ton de gris *t*. Dans le cas où ce ton n'existe pas dans la région, cette méthode retourne None. Pour l'exemple de la Figure 1 et le ton 64, cette méthode retourne la liste [(1,3), (2,3)].
- **\_\_contains\_\_(...)** : retourne True si un pixel *px* appartient à la région et False sinon.
- **ajouter\_pixel(...)** : ajoute un pixel *px* à la région.
- **supprimer\_pixel(...)** : supprime un pixel *px* de la région.
- **binariser\_reg(...)** : retourne, à partir d'un ton *t*, une nouvelle région à deux tons 0 et 255 tels que le ton 0 est associé aux pixels ayant un ton strictement inférieur à *t* et le ton 255 est associé aux autres. Le label de la nouvelle région est celui de la région d'origine post-fixé par la chaîne 'bin'.

## Travail demandé

Dans la suite, les réponses seront écrites en langage Python, en se basant sur les descriptions des classes déjà données.

1. Construire la classe **Pixel**.

Pour la classe **Region** :

2. Ecrire la méthode **\_\_len\_\_**.
3. Ecrire la méthode **\_\_call\_\_**.
4. Ecrire la méthode **\_\_contains\_\_**.
5. Ecrire la méthode **ajouter\_pixel**.
6. Ecrire la méthode **supprimer\_pixel**.
7. Ecrire la méthode **binariser\_reg**.



8. En supposant disposer d'une matrice **Im** de type `numpy.ndarray` représentant une image en niveaux de gris, écrire les instructions Python permettant de :
- créer, à partir de **Im**, une instance de la classe **Region**, notée **Reg** et ayant le label 'Paysage' ;
  - binariser la région **Reg** à partir d'un ton de gris, noté **seuil**, supposé déjà calculé.

## PROBLEME 2

Soit le schéma relationnel de la base de données des images 'BDDImage.db', défini par les trois relations suivantes :

▪ **Image** (IdImage, nomF, extF, largeur, hauteur)

La table Image contient toutes les images actuellement disponibles dans la base.

- IdImage : identifiant de l'image (chaîne de caractères), clé primaire.
- nomF : nom du fichier correspondant à l'image (chaîne de caractères).
- extF : extension du fichier correspondant à l'image (chaîne de caractères).
- hauteur : hauteur de l'image en nombre de pixels (entier).
- largeur : largeur de l'image en nombre de pixels (entier).

▪ **Region** (IdRegion, label, IdImage)

La table Region contient toutes les régions associées aux images sachant qu'une région est une partie d'une image.

- IdRegion : identifiant de la région (chaîne de caractères), clé primaire.
- label : désignation de la région (chaîne de caractères).
- IdImage : identifiant de l'image (chaîne de caractères) à laquelle appartient la région identifiée par IdRegion, clé étrangère.

▪ **Pixel** (X, Y, IdRegion, rouge, vert, bleu).

La table Pixel contient tous les pixels couleur associés aux régions.

- X : position au niveau des lignes dans l'image (entier).
- Y : position au niveau des colonnes dans l'image (entier).
- IdRegion : identifiant de la région à laquelle appartient le pixel de positions X et Y (chaîne de caractères), clé étrangère.
- rouge : valeur de la couleur rouge, (entier entre 0 et 255).
- vert : valeur de la couleur vert, (entier entre 0 et 255).
- bleu : valeur de la couleur bleu, (entier entre 0 et 255).

Le triplet (X, Y, IdRegion) constitue la clé primaire de la table Pixel.

Dans la suite, on suppose que les trois tables de 'BDDImage.db' sont déjà créées et remplies.

## Partie I

Soit le script python suivant :

```
import sqlite3
connexion = sqlite3.connect('BDDImage.db')
curseur = connexion.cursor()

...

curseur.close()
connexion.close()
```

Dans la suite, connexion et curseur seront utilisées par les fonctions comme des variables globales.

1. Ecrire la fonction Python, nommée **GetData**, qui, à partir d'une table **T**, détermine et retourne la liste de toutes les informations des lignes de cette table.
2. Ecrire la fonction Python, nommée **RechercheImages**, qui, à partir de deux entiers **minK** et **maxK**:
  - détermine la liste **L** de toutes les informations des images de la table Image ;
  - détermine, à partir de **L**, la liste des identifiants des images dont la taille (en Kilo octets) est comprise entre **minK** et **maxK**;
  - retourne la liste résultante.

**N.B :** La valeur de chaque couleur (entier entre 0 et 255) est codée sur 1 octet.

## Partie II

Donner les requêtes SQL permettant de :

3. Déterminer les noms et les dimensions en nombre de pixels des images ayant l'extension 'png'. Le résultat doit être classé par ordre décroissant selon la dimension.
4. Déterminer le nombre de pixels par région.
5. Déterminer les identifiants des images comportant exactement deux régions.
6. Déterminer les positions des pixels de dominance rouge de la région ayant le label 'Cercle'. Un pixel de dominance rouge est un pixel dont la valeur de la couleur rouge est supérieure à celles des deux autres couleurs.
7. Sachant qu'un pixel bleu a la valeur du triplet (rouge, vert, bleu) = (0, 0, 255), modifier en bleu tous les pixels de la région d'identifiant 'MER102'.

## Partie III

Ecrire en algèbre relationnelle les expressions suivantes :

8. Déterminer les positions des pixels de la région ayant le label 'Cercle'.
9. Déterminer les noms des fichiers des images n'ayant aucune région associée.