Concours Blanc - Epreuve d'informatique – Durée : 45 minutes

Banque – TB 2024

L'usage d'abaques, de tables, de calculatrice et de tout instrument électronique susceptible de permettre au candidat d'accéder à des données et de les traiter par les moyens autres que ceux fournis dans le sujet est interdit.

Chaque candidat est responsable de la vérification de son sujet d'épreuve : pagination et impression de chaque page. Ce contrôle doit être fait en début d'épreuve. En cas de doute, le candidat doit alerter au plus tôt le surveillant qui vérifiera et, éventuellement, remplacera le sujet.

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

L'imagerie médicale prend une place grandissante dans le diagnostic des pathologies en médecine. Nous proposons en première partie l'étude de manipulations simples d'images en niveau de gris. En deuxième partie, nous proposons de trier chronologiquement une liste d'images et de faire une recherche dans cette liste. Dans la troisième partie, nous étudions un ensemble de données d'images médicales issues de plusieurs laboratoires et organisées en base de données. Les trois parties sont indépendantes.

I Manipulation d'images

Une image numérique de type matriciel est constituée par un pavage de pixels contigus. Un pixel est un petit carré monochrome (une seule couleur). Dans les images à niveau de gris que nous étudierons dans ce sujet, chaque pixel est codé par un nombre compris entre 0 et 255.

— 0 correspond à un pixel noir ;

— 255 correspond à un pixel blanc.

Dans ce sujet, une image sera représentée par une liste de listes. Par exemple l'image représentée par

$$L = [[0, 255, 255], [122, 255, 0]]$$

possède 3 colonnes de pixels et 2 lignes de pixels.

On appelle *taille* de l'image, le produit du nombre de colonnes et de lignes.

Dans l'exemple, la taille de l'image vaut $3 \times 2 = 6$.

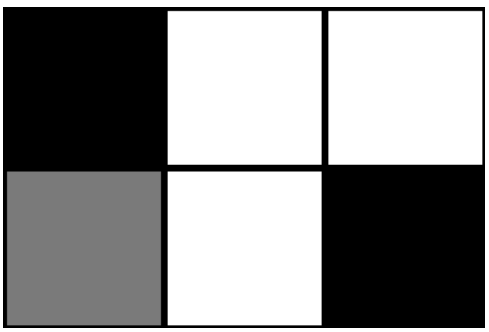

Fig.1 Image correspondante à la liste

A. Étude d'un exemple

On considère seulement dans ce paragraphe l'image représentée par

$$L = [[200, 100, 155], [0, 254, 255]].$$

1. Que renvoie l'instruction `L[1]` ?
2. Que renvoie l'instruction `L[1][1]` ?

3. Que renvoie l'instruction `len(L)` ?
4. Que renvoie l'instruction `len(L[1])` ?

On considère la fonction suivante :

```
def mz(n,m):
    return [m * [0] for k in range(n)]
```

5. Que renvoie `mz(2,3)` parmi les trois possibilités suivantes ?
   (a) `[[0,0],[0,0],[0,0]]`
   (b) `[[0,0,0],[0,0,0]]`
   (c) `[0,0,0],[0,0,0]`

B. Cas général

À partir de maintenant, on se propose d'effectuer des opérations sur une image quelconque représentée par sa liste de listes L.

6. Recopier en complétant la fonction `taille(L)` qui prend en argument une image L et qui renvoie la taille de l'image L :

```
def taille(L):
    ...
    ...
    return
```

7. Compléter la fonction `negatif(L)` qui prend en argument une image L et qui renvoie le négatif de l'image L dans laquelle chaque pixel a une couleur opposée (par exemple, le pixel 0 devient 255, le pixel 100 devient 155, le pixel 255 devient 0) :

```
def negatif(L):
    N = mz(len(L), len(L[0]))
    ...
    ...
    return(N)
```

8. Compléter la fonction `monochrome(L,s)` qui prend en arguments une image L, un seuil $s \in [\![1, 255]\!]$ et qui renvoie une image dans laquelle les pixels sont blancs dès que les pixels de L sont supérieurs ou égaux à s, les autres étant noirs.

   Exemple :

```
L = [[200, 100, 155], [0, 254, 255]].
print(monochrome(L,155))
>>> [[255,0,255],[0,255,255]]
```

```
def monochrome(L, s):
    M = mz(len(L), len(L[0]))
    ...
    ...
    return(M)
```

9. Compléter la fonction `reflexion(L)` qui prend en argument une image L et qui renvoie l'image symétrique par rapport à l'axe vertical médiateur :

```
def reflexion(L):
    R = mz(len(L), len(L[0]))
    ...
    ...
    return(R)
```

II Tri des images par ordre chronologique

On dispose d'une liste d'images nommée `photos`. Chaque image est représentée par une liste contenant son numéro (type entier) et la date de la prise de la photographie qui constitue l'image (type entier, affiché au format `aaaammjj`, par exemple `20240503` pour le 03 mai 2024).

On souhaite trier par ordre chronologique les photos en fonction de la date de prise du cliché en utilisant le tri par sélection. Rappelons le principe de ce tri. Soit L une liste d'entiers :

- on parcourt la liste L à la recherche du plus petit élément ;
- on échange sa position avec le premier élément ;
- on reprend les étapes précédentes avec la sous-liste `L[1:]` ;
- on continue jusqu'à ce que la liste soit triée.

10. Compléter la fonction `selection(L)` qui applique à une liste L constituée d'entiers l'algorithme du tri par sélection présenté ci-dessus.

```
def selection(L):
    n = len(L)
    for i in range(n - 1):
        # on suppose que le minimum est à la position i
        indice_min = i
        for j in range(i + 1, n):
            if .... < .....:
                indice_min = j
        # échange
        L[i], L[indice_min] =........,  .......
    return L
```

11. Modifier l'algorithme précédent pour qu'il s'applique à la liste `photos`.
    On rappelle que la liste `photos` est de la forme `photos= [ [0, 20240501] ,[1,20230312],...]`. un entier le numéro de la photo, puis la date.

12. On suppose dans cette question que la liste `photos` a été triée dans l'ordre chronologique. On souhaite savoir si une photo a été prise à une date d. Ecrire une fonction Python qui permet prend en argument la liste des photos triées et renvoie la liste de toutes les photos prises à la date `d`. Elle renverra une liste vide si aucune photo n'a été prise le jour en question.

III Gestion des images en SQL

Les données d'imagerie d'un centre médical sont désormais stockées dans une table relationnelle `images` comportant les colonnes suivantes :

— `id_image` : numéro de l'image (entier) ;

— `taille_mo` : taille de l'image en Mo (float) ;

— `date_prise` : date à laquelle l'image a été prise (entier au format `aaaammjj`) ;

— `patient` : nom du patient ;

— `laboratoire` : nom du laboratoire qui a fourni le cliché ;

— `pathologie` : nom de la pathologie détectée, chaîne vide avant diagnostic ;

— `medecin` : nom du médecin ayant posé le diagnostic, chaîne vide avant diagnostic.

On considère par exemple les cinq premières lignes suivantes :

```
(109, 2.9, 20221223, 'DUKIC', 'OCEANA', '', ''),
(203, 1.2, 20221005, 'MOING', 'RADIO1', 'FRACTURE', 'TROUSSEAU'),
(405, 6.9, 20230178, 'KARL', 'RADIO2', 'ARTHROSE', 'JENNER'),
(108, 4.1, 20230206, 'DUKIC', 'OCEANA', 'FRACTURE', 'PARE'),
(406, 2.0, 20230612, 'KARL', 'RADIO2', 'TENDINITE', 'JENNER');
```

L'image 109 a une taille de 2,9 Mo, a été prise le 23 décembre 2022 sur le patient DUKIC par le laboratoire OCEANA, mais aucun diagnostic n'a été établi en l'utilisant.

13. Écrire une requête SQL qui fournit, sans répétition, le nom des patients de la table `images`.
14. Écrire une requête SQL qui calcule le nombre de clichés qui ont été pris à la date du `20240301`.
15. Écrire une requête SQL qui donne, sans répétition, le nom des laboratoires qui ont fourni un cliché permettant de diagnostiquer une `FRACTURE`.

Mot-clé	Rôle	Syntaxe générale
SELECT	Sélectionner des colonnes	`SELECT col1, col2`
FROM	Indiquer la table source	`FROM table`
WHERE	Filtrer selon une condition	`WHERE condition`
DISTINCT	Supprimer les doublons	`SELECT DISTINCT col`
ORDER BY	Trier les résultats	`ORDER BY col ASC/DESC`
GROUP BY	Regrouper les lignes	`GROUP BY col`
HAVING	Filtrer après regroupement	`HAVING condition`
COUNT	Compter le nombre de lignes	`COUNT(*)`
SUM	Calculer une somme	`SUM(col)`
AVG	Calculer une moyenne	`AVG(col)`
MIN / MAX	Minimum / Maximum	`MIN(col), MAX(col)`
AND	Conjonction logique	`cond1 AND cond2`
OR	Disjonction logique	`cond1 OR cond2`

FIN DU SUJET

Correction 1.

1. L'instruction `L[1]` renvoie la deuxième ligne de l'image, c'est-à-dire

$$[0, 254, 255].$$

2. L'instruction `L[1][1]` renvoie le deuxième élément de la deuxième ligne, c'est-à-dire

$$254.$$

3. L'instruction `len(L)` renvoie le nombre de lignes de l'image, donc

   2.

4. L'instruction `len(L[1])` renvoie le nombre d'éléments de la deuxième ligne, donc le nombre de colonnes :

   3.

5. La fonction `mz(2,3)` renvoie une liste de 2 lignes contenant chacune 3 zéros. La bonne réponse est donc :

   `[[0,0,0],[0,0,0]]`.

6. Une image représentée par une liste de listes possède `len(L)` lignes et `len(L[0])` colonnes. Sa taille est donc le produit de ces deux nombres.

```
def taille(L):
    n = len(L)        # nombre de lignes
    m = len(L[0])     # nombre de colonnes
    return n * m
```

7. Pour obtenir le négatif d'une image, chaque pixel de valeur p est remplacé par $255 - p$.

```
def negatif(L):
    N = mz(len(L), len(L[0]))
    for i in range(len(L)):
        for j in range(len(L[0])):
            N[i][j] = 255 - L[i][j]
    return N
```

8. On parcourt tous les pixels de l'image. Si le pixel est supérieur ou égal au seuil s, on le remplace par 255, sinon par 0.

```
def monochrome(L, s):
    M = mz(len(L), len(L[0]))
    for i in range(len(L)):
        for j in range(len(L[0])):
            if L[i][j] >= s:
                M[i][j] = 255
            else:
                M[i][j] = 0
    return M
```

9. La réflexion par rapport à l'axe vertical inverse l'ordre des colonnes dans chaque ligne.

```
def reflexion(L):
    R = mz(len(L), len(L[0]))
    for i in range(len(L)):
        for j in range(len(L[0])):
            R[i][j] = L[i][len(L[0]) - 1 - j]
    return R
```

10. Il faut trier les photos selon leur date, c'est-à-dire selon le deuxième élément de chaque sous-liste.

```
def selection_photos(photos):
    n = len(photos)
    for i in range(n - 1):
        indice_min = i
        for j in range(i + 1, n):
            if photos[j][1] < photos[indice_min][1]:
                indice_min = j
        photos[i], photos[indice_min] = photos[indice_min], photos[i]
    return photos
```

11. On parcourt la liste des photos et on ajoute à la réponse toutes les photos dont la date est égale à d.

```
def photos_date(photos, d):
    R = []
    for i in range(len(photos)):
        if photos[i][1] == d:
            R.append(photos[i])
    return R
```

12. La requête SQL demandée est :

```
SELECT DISTINCT patient
FROM images;
```

13. La requête SQL demandée est :

```
SELECT COUNT(*)
FROM images
WHERE date_prise = 20240301;
```

14. La requête SQL demandée est :

```
SELECT DISTINCT laboratoire
FROM images
WHERE pathologie = 'FRACTURE';
```