Devoir surveillé n°4

Durée : 45 minutes

L'utilisation d'une calculatrice ou de tout document est interdite.
Toute sortie anticipée est interdite.

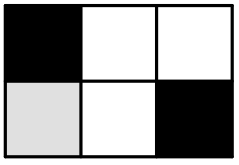
L'imagerie médicale prend une place grandissante dans le diagnostic des pathologies en médecine. Nous proposons en première partie l'étude des manipulations simples d'images en niveau de gris. En deuxième partie, nous proposons de trier chronologiquement une liste d'images et de faire une recherche dans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



sera représentée par : `L = [[0,255,255],[122,255,0]]`.

Cette image possède 3 colonnes de pixels et 2 lignes de pixels. Le produit $3 \times 2 = 6$ donne la taille de l'image.

1) 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]`

2) 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` 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` qui prend en argument une image `L` et qui renvoie le négatif de l'image `L` dans lequel 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` qui prend en argument une image `L`, un seuil `s` appartenant à $[\![1, 255]\!]$ 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 :

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

9. Compléter la fonction `reflexion` 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
```

10. Compléter la fonction `transposeL` qui prend en argument une image `L` et qui renvoie une image dont les lignes sont les colonnes de `L` :

```
def transpose(L):
    T=mz(len(L[0]),len(L))
    ...
    ...
    return T
```

II. — Tri des images par ordre chronologique

On dispose d'une liste d'images nommées `photos`. Chaque image est représentée par une liste contenant son numéro (type entier) et la date de 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. Le premier élément est maintenant le plus petit ;
— on reprend les deux étapes précédentes avec la sous liste `L[1:]`. Les deux premiers éléments sont triés ;
— on continue jusqu'à ce que la liste soit triée.

11. Écrire une fonction `selection` qui applique à une liste `L` constituée d'entiers l'algorithme du tri par sélection présenté ci-dessus :

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

12. Modifier l'algorithme précédent pour qu'il s'applique à la liste `photos`.

13. On considère la liste `L` d'entiers triés dans l'ordre croissant, on souhaite à l'aide d'une recherche dichotomique savoir si un entier `x` est dans la liste. Compléter la fonction ci-dessous qui prend en argument un entier `x` et une liste `T` d'entiers triés et qui renvoie `True` si `x` est dans la liste et renvoie `False` sinon.

```
def dicho(x,T):
    a=0
    b=len(T-1)
    while ...
        m=...
        if x==T[m]:
            return ...
        elif x<T[m]:
            b= ..
        else:
            a=...
    return
```

14. 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`. Modifier le programme précédent afin qu'il réponde à la question lorsque l'on exécute `dicho(d,photos)`.

III. — Gestion des images

Les données d'imagerie d'un centre médical ont été organisées dans un dictionnaire `images` qui comprend comme clé le numéro de l'image (type entier) et comme valeur une liste de plusieurs données :

- taille de l'image en Mo (type flottant) ;
- date à laquelle l'image a été prise (type entier dont la valeur vaut aaaammjj, par exemple 20240503 pour le 03 mai 2024) ;
- nom du patient : chaîne de caractères ;
- nom du laboratoire qui a fourni le cliché : chaîne de caractères ;
- nom de la pathologie détectée : chaîne de caractères, réduite à une chaîne de longueur 0 avant le diagnostic ;
- nom du médecin qui a fait le diagnostic : chaîne de caractères, réduite à une chaîne de longueur 0 avant le diagnostic ;

Le code suivant initialise par exemple les cinq premiers éléments du dictionnaire `images` :

```
images={
109:[2.9, 20221223, "DUKIC", "OCEANA","",""],
203:[1.2, 20221005, "MOING", "RADIO1","FRACTURE","TROUSSEAU"],
405:[6.9, 20230108, "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.

On rappelle qu'on peut parcourir un dictionnaire avec une boucle, comme par exemple :

```
for unecle, unevaleur in images.items():
    print(unecle, unevaleur)
```

qui va conduire à un début d'affichage :

```
109 [2.9, 20221223, 'DUKIC', 'OCEANA','','',
203 [1.2, 20221005, 'MOING', 'RADIO1','FRACTURE','TROUSSEAU'],
405 [6.9, 20230178, 'KARL', 'RADIO2','ARTHROSE','JENNER'],
...
...
```

15. Écrire un code qui fournit, sans répétition, le nom des patients du dictionnaire `images`.

16. Écrire un code qui calcule dans `images` le nombre de clichés qui ont été pris à la date du 20240301.

17. Écrire un code qui donne, sans répétition, les noms des laboratoires qui ont fourni un cliché permettant de diagnostiquer une `FRACTURE`.

Corrigé

1. L'instruction `L[1]` renvoie `[0,254,255]`.
2. L'instruction `L[1][1]` renvoie `254`.
3. L'instruction `len(L)` renvoie 2.
4. L'instruction `len(L[1])` renvoie 3.
5. Parmi les trois possibilités de l'énoncé, `mz(2,3)` renvoie `[[0,0,0],[0,0,0]]`.
6.

```
def taille(L):
  n = len(L)
  m = len(L[0])
  return n*m
```

   Remarque : on a suivi l'énoncé en écrivant 4 lignes de code mais on aurait pu simplement écrire

```
def taille(L):
  return len(L)*len(L[0])
```

7.

```
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.

```
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
  return M
```

9.

```
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.

```
def transpose(L):
  T=mz(len(L[0]),len(L))
  for i in range(len(L)):
    for j in range(len(L[i])):
      T[j][i] = L[i][j]
  return T
```

11.

```
def selection(L):
  for i in range(len(L)-1):
    ind_min = i
    for j in range(i+1,len(L)):
      if L[j] < L[ind_min]:
        ind_min = j
        L[i],L[ind_min] = L[ind_min],L[i]
  return L
```

12.

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

13.

```
def dicho(x,T):
  a = 0
  b = len(T)-1
  while a <= b:
    m = (a+b)//2
    if x == T[m]:
      return True
    elif x < T[m]:
      b = m-1
    else:
      a = m+1
  return False
```

14.

```
def dicho(d,photos):
  a = 0
  b = len(T)-1
  while a <= b:
    m = (a+b)//2
    if d == photos[m][1]:
      return True
    elif d < photos[m][1]:
      b = m-1
    else:
      a = m+1
  return False
```

15.

```
P = []
for unecle, unevaleur in images.items():
  if not(unevaleur[2] in P):
    P.append(unevaleur[2])
print(P)
```

16.

```
nb = 0
for unecle, unevaleur in images.items():
  if unevaleur[1] == 20240301:
    nb += 1
print(nb)
```

17.

```
L = []
for unecle, unevaleur in images.items():
  if not(unevaleur[3] in L) and (unevaleur[4] == "FRACTURE"):
    L.append(unevaleur[3])
print(L)
```