BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°04

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Écrire une fonction `recherche` qui prend en paramètre un tableau de nombres entiers `tab`, et qui renvoie la liste (éventuellement vide) des couples d'entiers consécutifs successifs qu'il peut y avoir dans `tab`.

Exemples :

```
>>> recherche([1, 4, 3, 5])
[]
>>> recherche([1, 4, 5, 3])
[(4, 5)]
>>> recherche([7, 1, 2, 5, 3, 4])
[(1, 2), (3, 4)]
>>> recherche([5, 1, 2, 3, 8, -5, -4, 7])
[(1, 2), (2, 3), (-5, -4)]
```

EXERCICE 2 (4 points)

Soit une image binaire représentée dans un tableau à 2 dimensions. Les éléments `M[i][j]`, appelés pixels, sont égaux soit à `0` soit à `1`.

Une composante d'une image est un sous-ensemble de l'image constitué uniquement de `1` et de `0` qui sont côte à côte, soit horizontalement soit verticalement.

Par exemple, les composantes de

M =

0	0	1	0
0	1	0	1
1	1	1	0
0	1	1	0

sont

M =

0	0	1	0
0	1	0	1
1	1	1	0
0	1	1	0

On souhaite, à partir d'un pixel égal à 1 dans une image M, donner la valeur `val` à tous les pixels de la composante à laquelle appartient ce pixel.

La fonction `propager` prend pour paramètre une image `M`, deux entiers `i` et `j` et une valeur entière `val`. Elle met à la valeur `val` tous les pixels de la composante du pixel `M[i][j]` s'il vaut `1` et ne fait rien s'il vaut `0`.

Par exemple, `propager(M,2,1,3)` donne

M =

0	0	1	0
0	3	0	1
3	**3**	3	0
0	3	3	0

Compléter le code récursif de la fonction `propager` donné ci-dessous

```
def propager(M, i, j, val):
    if M[i][j]== ...:
        return

    M[i][j]=val

    # l'élément en haut fait partie de la composante
    if ((i-1) >= 0 and M[i-1][j] == ...):
        propager(M, i-1, j, val)

    # l'élément en bas fait partie de la composante
    if ((...) < len(M) and M[i+1][j] == 1):
        propager(M, ..., j, val)

    # l'élément à gauche fait partie de la composante
    if ((...) >= 0 and M[i][j-1] == 1):
        propager(M, i, ..., val)

    # l'élément à droite fait partie de la composante
    if ((...) < len(M) and M[i][j+1] == 1):
        propager(M, i, ..., val)
```

Exemple :

```
>>> M = [[0,0,1,0],[0,1,0,1],[1,1,1,0],[0,1,1,0]]
>>> propager(M,2,1,3)
>>> M
[[0, 0, 1, 0], [0, 3, 0, 1], [3, 3, 3, 0], [0, 3, 3, 0]]
```

```python
from doctest import testmod


def recherche(tab):
    """ Recherche les couples d'entiers consécutifs d'un tableau.

    Args:
        tab (list): tableau d'entiers

    Returns:
        list: tableau de couples d'entiers consécutifs

    Tests et Exemples:
    >>> recherche([1, 4, 3, 5])
    []
    >>> recherche([1, 4, 5, 3])
    [(4, 5)]
    >>> recherche([7, 1, 2, 5, 3, 4])
    [(1, 2), (3, 4)]
    >>> recherche([5, 1, 2, 3, 8, -5, -4, 7])
    [(1, 2), (2, 3), (-5, -4)]
    """
    couple_consecutifs = []

    n = len(tab)
    for i in range(1, n):
        pred = tab[i-1]
        suiv = tab[i]

        if suiv == pred + 1:
            couple_consecutifs.append( (pred, suiv) )

    return couple_consecutifs



assert recherche([1, 4, 3, 5]) == []
assert recherche([1, 4, 5, 3]) == [(4, 5)]
assert recherche([7, 1, 2, 5, 3, 4]) == [(1, 2), (3, 4)]
assert recherche([5, 1, 2, 3, 8, -5, -4, 7]) == [(1, 2), (2, 3), (-5, -4)]

testmod()
```

```python
from doctest import testmod


def propager(M, i, j, val):
    """ Met tous les pixels de la composante de M[i][j]
    * à la valeur val si M[i][j] vaut 1 et
    * à 0 sinon.

    Args:
        M (list): matrice de l'image
        i (int): abscisse du pixel de référence
        j (int): ordonnée du pixel de référence
        val (int): valeur de seuil différente de 1

    Tests et Exemples:
    >>> M = [[0, 0, 1, 0], [0, 1, 0, 1], [1, 1, 1, 0], [0, 1, 1, 0]]
    >>> propager(M, 2, 1, 3)
    >>> M
    [[0, 0, 1, 0], [0, 3, 0, 1], [3, 3, 3, 0], [0, 3, 3, 0]]
    """
    # cas où il n'y a pas de composante
    # et donc rien à propager
    # et donc l'image M est inchangée
    if M[i][j] == 0:
        return

    M[i][j]=val

    # l'élément en haut fait partie de la composante
    # car il fait parti de l'image (le pixel ne déborde pas)
    # car sa valeur vaut 1
    if ( (i-1) >= 0 and M[i-1][j] == 1 ):
        propager(M, i-1, j, val)

    # l'élément en bas fait partie de la composante
    # car il ne déborde par de l'image
    # et car sa valeur vaut 1
    if ((i+1) < len(M) and M[i+1][j] == 1):
        # appel récursif à propager sur le pixel du bas
        propager(M, i+1, j, val)

    # l'élément à gauche fait partie de la composante
    # car il ne déborde par de l'image
    # et car sa valeur vaut 1
    if ((j-1) >= 0 and M[i][j-1] == 1):
        # appel récursif à propager sur le pixel de gauche
        propager(M, i, j-1, val)

    # l'élément à droite fait partie de la composante
    # car il ne déborde par de l'image
    # et car sa valeur vaut 1
    if ((j+1) < len(M) and M[i][j+1] == 1):
        # appel récursif à propager sur le pixel de droite
        propager(M, i, j+1, val)


M = [[0,0,1,0],
     [0,1,0,1],
     [1,1,1,0],
     [0,1,1,0]]

propager(M,2,1,3)
assert M == [[0, 0, 1, 0],
             [0, 3, 0, 1],
             [3, 3, 3, 0],
             [0, 3, 3, 0]]


testmod()
```