Banque « Agro-Veto »
A TB - 0720

Algorithmique et Informatique
Durée : 45 mn

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, il doit alerter au plus tôt le surveillant qui vérifiera et, éventuellement, remplacera le sujet.

Ce sujet comporte 4 pages numérotées de 1 à 4.

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.

1 Questions préliminaires

a. On considère la liste de listes `matA` définie en `Python` par:

```
matA =[[5, 2, 3, 6, 1], [9, 7, 9, 7, 9], [7, 4, 1, 2, 5], [3, 2, 7, 1, 4]]
```

(i) Quelle est la valeur de `len(matA)` ?

(ii) Quel est le type de `matA[2]` ? Quelle est sa valeur ?

(iii) Quelle est la valeur de `len(matA[2])` ?

(iv) Quel est le type de `matA[2][1]` ? Quelle est sa valeur ?

b. Parmi les fonctions suivantes, déterminer l'unique fonction `zeros` qui, à partir de deux entiers n et p passés en arguments, renvoie une liste de n listes, contenant chacune p coefficients, tous nuls.

Aucune justification n'est attendue.

Par exemple, `zeros(3,4)` *devra renvoyer* `[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]`.

```
def zeros1(n, p):
    liste = []
    for i in range(n):
        for j in range(p):
            liste.append([0])
    return liste
```

```
def zeros2(n, p):
    liste = []
    for j in range(p):
        colonne = []
        for i in range(n):
            colonne.append(0)
        liste.append(colonne)
    return liste
```

```
def zeros3(n, p):
    liste = []
    for i in range(n):
        ligne = []
        for j in range(p):
            ligne.append(0)
        liste.append(ligne)
    return liste
```

```
def zeros4(n, p):
    liste = []
    for i in range(n):
        ligne = []
        for j in range(p):
            ligne.append(0)
        liste = liste + ligne
    return liste
```

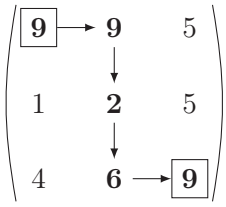
c. Écrire une fonction `miroir` qui renvoie le miroir d'une liste passée en argument, c'est-à-dire une liste dont les éléments sont énumérés dans l'ordre inverse de ceux de la liste passée en argument.

Par exemple, `miroir([4, 2, 5, 3, 3])` *et* `miroir([(2,3),(2,2),(1,2),(0,2),(0,1),(0,0)])` *devront respectivement renvoyer* `[3, 3, 5, 2, 4]` *et* `[(0,0),(0,1),(0,2),(1,2),(2,2),(2,3)]`.

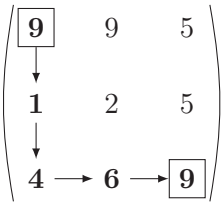
2 Chemins de somme minimale dans une matrice

On appelle **chemin** dans une matrice toute suite de coefficients adjacents reliant son coin supérieur gauche à son coin inférieur droit. **Les seuls déplacements licites sont les déplacements vers la droite ou vers le bas.** On s'intéresse ici à la somme des coefficients de chemin dans une matrice.

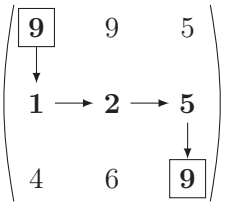
Par exemple, dans la matrice $A = \begin{pmatrix} 9 & 9 & 5 \\ 1 & 2 & 5 \\ 4 & 6 & 9 \end{pmatrix}$*, la somme des coefficients du chemin ci-dessous à gauche est égale à 35, tandis que la somme des coefficients du chemin du milieu et de droite sont respectivement égaux à 29 et 26.*



$9 + 9 + 2 + 6 + 9 = 35$



$9 + 1 + 4 + 6 + 9 = 29$



$9 + 1 + 2 + 5 + 9 = 26.$

On peut montrer que le chemin de droite est le chemin dont la somme des coefficients est minimale.
Dans cette partie, on cherche à déterminer la somme minimale des coefficients d'une matrice parmi tous les chemins reliant son coin supérieur gauche et son coin inférieur droit.

a. On modélise une matrice par une liste de listes notée `mat` de façon à ce que `mat[i][j]` désigne le coefficient de la ligne i et de la colonne j, **en commençant la numérotation des lignes et colonnes à 0**.

Ainsi, la matrice A ci-dessus est représentée par la liste de listes :

```
matA =[[9, 9, 5], [1, 2, 5], [4, 6, 9]]
```

(i) Quelle liste de listes représente la matrice $\begin{pmatrix} 9 & 2 \\ 3 & 6 \\ 4 & 7 \end{pmatrix}$?

(ii) Quelle matrice la liste de listes `[[9, 3, 4], [2, 6, 7]]` représente-t-elle ?

b. Pour résoudre le problème, il serait très inefficace de calculer la longueur de tous les chemins reliant le coin supérieur gauche de A à son coin inférieur droit.

Pour une matrice $A = (a_{i,j})_{\substack{0 \leqslant i \leqslant n-1 \\ 0 \leqslant j \leqslant p-1}}$ (de taille $n \times p$), on note $s_{i,j}$ la somme minimale des coefficients de la matrice parmi tous les chemins entre le coefficient $a_{0,0}$ (coin supérieur gauche) et le coefficient $a_{i,j}$ (ligne i colonne j). On admet les résultats suivants :

(i) $s_{0,0} = a_{0,0}$

(ii) Pour tout $i \in [\![0, n-2]\!]$, $s_{i+1,0} = s_{i,0} + a_{i+1,0}$.

(iii) Pour tout $j \in [\![0, p-2]\!]$, $s_{0,j+1} = s_{0,j} + a_{0,j+1}$.

(iv) Pour tous $i \in [\![0, n-2]\!]$ et $j \in [\![0, p-2]\!]$, $s_{i+1,j+1} = \begin{cases} s_{i+1,j} + a_{i+1,j+1} & \text{si } s_{i+1,j} < s_{i,j+1} \\ s_{i,j+1} + a_{i+1,j+1} & \text{sinon} \end{cases}$.

La valeur recherchée est alors $s_{n-1,p-1}$.

Pour éviter de recalculer plusieurs fois les coefficients $s_{i,j}$, on les stockera dans une matrice $S = (s_{i,j})$ de même taille que A. On appelle cette matrice la **matrice des sommes minimales** de A. En effet, chacun des ses coefficients $s_{i,j}$ est la somme minimale des coefficients de A reliant le coefficient $a_{0,0}$ (coin supérieur gauche) au coefficient $a_{i,j}$.

À l'aide des formules ci-dessous, on trouve que la matrice S *des sommes minimales de* $A = \begin{pmatrix} 9 & 9 & 5 \\ 1 & 2 & 5 \\ 4 & 6 & 9 \end{pmatrix}$ *est :*

$$S = \begin{pmatrix} 9 & 18 & 23 \\ 10 & 12 & 17 \\ 14 & 18 & 26 \end{pmatrix}.$$

(i) Déterminer la matrice S des sommes minimales de la matrice $B = \begin{pmatrix} 5 & 9 & 2 \\ 8 & 7 & 1 \\ 2 & 5 & 3 \end{pmatrix}$.

(ii) En déduire que la somme minimale des coefficients de B, parmi tous les chemins reliant le coin supérieur gauche de B à son coin inférieur droit, est égale à 20.

c. Recopier sur votre copie et compléter le code de la fonction ci-dessous de manière à ce qu'elle renvoie une liste de listes représentant la matrice des sommes minimales associée à une matrice `matA` passée en argument (comme une liste de listes).

```
def mat_sommes_minimales(matA):
    n = len(matA)
    p = len(...)
    matS = zeros(..., ...)
    matS[0][0] = ...
    for i in range(...):
        matS[i+1][0] = ...
    for j in range(...):
        matS[0][j+1] = ...
    for i in range(...):
        for j in range(...):
            if matS[i+1][j] < ...:
                matS[i+1][j+1] = ...
            else:
                matS[i+1][j+1] = ...
    return matS
```

d. Écrire une fonction `somme_minimale(matA)` qui renvoie la somme minimale des coefficients d'une matrice `matA`, parmi tous les chemins reliant son coin supérieur gauche à son coin inférieur droit.

3 Recherche d'un chemin solution

Dans la partie précédente, on implémente une méthode déterminant la somme minimale des coefficients d'une matrice parmi tous les chemins reliant son coin supérieur gauche et son coin inférieur droit. Cette méthode permet de déterminer la longueur d'un chemin solution, mais pas un chemin solution.
Pour trouver un chemin solution, on procède de la manière suivante :

- on construit la matrice $S = (s_{i,j})$ des sommes minimales associée à la matrice A,
- On commence par se placer au coin inférieur droit de la matrice A (soit à la ligne $n-1$, colonne $p-1$ si elle a n lignes et p colonnes),
- On note i et j les numéros respectifs de ligne et de colonne actuels. Tant qu'on ne se trouve pas au point de départ (i.e. $(i,j) \neq (0,0)$), on se déplace sur la matrice A de la manière décrite ci-dessous.
  - Si on se trouve sur la première colonne, on remonte d'une ligne.
  - Si $s_{i-1,j} < s_{i,j-1}$, c'est que la somme minimale $s_{i-1,j}$ pour relier le coin supérieur gauche au coefficient $a_{i-1,j}$ est inférieure à celle pour relier le coin supérieur gauche au coefficient $a_{i,j-1}$. On remonte alors d'une ligne.
  - Dans tous les autres cas, on se décale vers la gauche, i.e. on recule d'une colonne.

Parmi les fonctions ci-après, déterminer l'unique fonction `chemin` prenant en argument une liste de listes représentant une matrice A et renvoyant la liste ordonnée (du coin supérieur gauche au coin inférieur droit) des coordonnées des coefficients d'un chemin solution.
On indiquera les raisons qui ont amené à éliminer les trois autres fonctions.
L'exécution des instructions ci-dessous affichera par exemple la liste `[(0,0),(1,0),(1,1),(1,2),(2,2)]`.

```
matA =[[9, 9, 5], [1, 2, 5], [4, 6, 9]]
print(chemin(matA))
```

```
def chemin1(matA):
    n, p = len(matA), len(matA[0])
    i, j = n-1, p-1
    matS = sommes_minimales(A)
    print(A[i][j])
    while (i,j) != (0,0):
        if j == 0 or matS[i-1][j] < matS[i][j-1]:
            i = i - 1
        else:
            j = j - 1
        print(matA[i][j])
```

```
def chemin2(matA):
    n, p = len(matA), len(matA[0])
    matS = sommes_minimales(A)
    i, j = n-1, p-1
    liste = [(n-1,p-1)]
    while i != 0 or j != 0:
        if j == 0 or matS[i-1][j] < matS[i][j-1]:
            i = i - 1
        else:
            j = j - 1
        liste.append((i,j))
     return miroir(liste)
```

```
def chemin3(matA):
    n, p = len(matA), len(matA[0])
    matS = sommes_minimales(A)
    i, j = n-1, p-1
    liste = [(n-1,p-1)]
    while i != 0 and j != 0:
        if j == 0 or matS[i-1][j] < matS[i][j-1]:
            i = i - 1
        else:
            j = j - 1
        liste.append((i,j))
     return miroir(liste)
```

```
def chemin4(matA):
    n, p = len(matA), len(matA[0])
    matS = sommes_minimales(A)
    i, j = n-1, p-1
    liste = [(n-1,p-1)]
    while i != 0 or j != 0:
        if j == 0 or matS[i-1][j] < matS[i][j-1]:
            i = i - 1
        else:
            j = j - 1
        liste.append((i,j))
     return liste
```

FIN DU SUJET