DS 6
Sujet d'informatique

Exercice : dichotomie

Soit $n \in \mathbb{N}^*$. Soit la fonction $f_n : x \mapsto x^n + \dfrac{x}{n} - 1$ sur $[0, +\infty[$.

1. Tracer le tableau de variations de f_n.
2. Justifier que l'équation $f_n(x) = 0$ admet une unique solution dans $]0, 1[$, qu'on appelle x_n.
3. Écrire une fonction Python `f(n, x)` qui prend en argument l'entier n et un nombre réel $x \geqslant 0$ et renvoie la valeur de $f_n(x)$.
4. Le but est de rechercher par dichotomie une approximation de la solution x_n. Recopier et compléter la fonction suivante pour qu'à tout moment on ait $a \leqslant x_n \leqslant b$ et qu'à la fin de la boucle l'écart $b - a$ soit strictement inférieur à la valeur ε passée en argument :

```
def solution(n, epsilon):
    a = ...
    b = ...
    while ... :
        m = (a+b) / 2
        ...
    return (a, b)
```

Problème : chemins dans les matrices

Soit $(n, p) \in (\mathbb{N}^*)^2$. Soit $A = (a_{i,j})_{(i,j)\in[\![0,n-1]\!]\times[\![0,p-1]\!]}$ une matrice à n lignes et p colonnes, où les lignes et les colonnes sont numérotées à partir de 0. On appelle **chemin** dans A une suite de coefficients partant du coin en haut à gauche $a_{0,0}$, allant jusqu'au coin en bas à droite $a_{n-1,p-1}$, et tel que chaque coefficient est soit immédiatement à droite soit immédiatement en dessous du précédent (on dit que les coefficients sont **adjacents**). Ainsi nos chemins se déplacent toujours vers le bas droit. On s'intéresse à la somme des coefficients sur un chemin.
Par exemple pour la matrice $A = \begin{pmatrix} 9 & 9 & 5 \\ 1 & 2 & 5 \\ 4 & 6 & 9 \end{pmatrix}$ on peut tracer les chemins suivants, avec indiqué en dessous la somme des coefficients sur le chemin :

$$\underset{9+9+2+6+9=35}{\begin{pmatrix} \mathbf{9} & \to & \mathbf{9} & & 5 \\ & & \downarrow & & \\ 1 & & \mathbf{2} & & 5 \\ & & \downarrow & & \\ 4 & & \mathbf{6} & \to & \mathbf{9} \end{pmatrix}} \qquad \underset{9+1+4+6+9=29}{\begin{pmatrix} \mathbf{9} & & 9 & & 5 \\ \downarrow & & & & \\ \mathbf{1} & & 2 & & 5 \\ \downarrow & & & & \\ \mathbf{4} & \to & \mathbf{6} & \to & \mathbf{9} \end{pmatrix}} \qquad \underset{9+1+2+5+9=26}{\begin{pmatrix} \mathbf{9} & & 9 & & 5 \\ \downarrow & & & & \\ \mathbf{1} & \to & \mathbf{2} & \to & \mathbf{5} \\ & & & & \downarrow \\ 4 & & 6 & & \mathbf{9} \end{pmatrix}} \qquad (*)$$

Le but du problème est de déterminer la somme minimale que l'on peut former, parmi tous les chemins dans la matrice A. Dans cet exemple, c'est le chemin de droite qui a une somme égale à 26, et on peut montrer que c'est bien la somme minimale parmi tous les chemins possibles pour cette matrice-là.

1. Préliminaires

On représente les matrices de taille $(n, p) \in \mathbb{N}^* \times \mathbb{N}^*$ par des listes de listes ; plus précisément une matrice A de taille (n, p) est donnée par la **liste** de ses **lignes**. Ainsi le coefficient $a_{i,j}$ est bien donné par `A[i][j]`.

1. (a) Quelle matrice la liste `[[9, 3, 4], [2, 6, 7]]` représente-t-elle ?
   (b) Quelle liste de listes représente la matrice $\begin{pmatrix} 9 & 2 \\ 3 & 6 \\ 4 & 7 \end{pmatrix}$?
2. Si `A` représente une matrice à n lignes et p colonnes, qu'est-ce que `len(A)` ? Et `len(A[0])` ? **Justifier**.
3. Laquelle de ces deux syntaxes permet de créer une matrice nulle à n lignes et p colonnes ? **Justifier**.

(i) `[[0 for i in range(n)] for j in range(p)]`

(ii) `[[0 for j in range(p)] for i in range(n)]`

4. Écrire une fonction `miroir(L)` qui prend en argument une liste `L` (de taille quelconque) et qui renvoie une nouvelle liste qui est le miroir de `L`, c'est à dire la liste rangée en ordre inverse.
Par exemple, `miroir([4, 2, 5, 3, 3])` doit renvoyer `[3, 3, 5, 2, 4]`.

2. Calcul du minimum

Pour résoudre notre problème, il serait très inefficace de lister *tous* les chemins puis de chercher lesquels ont une longueur minimale. Partant d'une matrice A, on forme alors une matrice $S = (s_{i,j})$ de même taille que A appelée **matrice des sommes minimales** où $s_{i,j}$ est la somme minimale obtenus sur les chemins reliant le coefficient $a_{0,0}$ (coin haut gauche de A) au coefficient $a_{i,j}$. Dans l'exemple (*) avec $A = \begin{pmatrix} 9 & 9 & 5 \\ 1 & 2 & 5 \\ 4 & 6 & 9 \end{pmatrix}$, on trouve $S = \begin{pmatrix} 9 & 18 & 23 \\ 10 & 12 & 17 \\ 14 & 18 & 26 \end{pmatrix}$, ce qui démontre que 26 est bien la somme minimale qu'on peut réaliser par des chemins allant du coin haut gauche au coin bas droit.
La matrice S se calcule pas à pas en partant du coin haut gauche.

5. Montrer que *(on pourra admettre ces formules pour passer à la suite)* les coefficients de S se calculent avec la relation de récurrence suivante :

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

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

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

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

6. (a) En appliquant ce procédé, déterminer la matrice des sommes minimales pour $B = \begin{pmatrix} 5 & 9 & 2 \\ 8 & 7 & 1 \\ 2 & 5 & 3 \end{pmatrix}$

(b) En déduire que la somme minimale de B, parmi tous les chemins reliant le coin haut gauche au coin bas droit, est égale à 20.

7. (a) Compléter le programme suivant pour écrire une fonction `matrice_sommes_minimales(A)` qui renvoie la matrice S des sommes minimales d'une matrice A passée en argument en utilisant les relations de récurrence de la question 5.

```
def matrice_sommes_minimales(A):
    n = ...
    p = ...
    S = ...
    for i in range(...):
        S[i+1][0] = ...
    for j in range(...):
        S[0][j+1] = ...
    for i in range(...):
        for j in range(...):
            ...
    return S
```

(b) En déduire une fonction `somme_minimale(A)` qui renvoie la somme minimale des chemins de A.

3. Calcul sur les chemins

Un chemin dans une matrice sera représenté tout simplement par une **liste** de **tuples** `(i, j)` représentant la suite de coefficients par lesquels passe le chemin. Dans notre exemple initial (*), les chemins sont représentés respectivement par

— `[(0, 0), (0, 1), (1, 1), (2, 1), (2, 2)]`

— `[(0, 0), (1, 0), (2, 0), (2, 1), (2, 2)]`

— `[(0, 0), (1, 0), (1, 1), (1, 2), (2, 2)]`

8. La fonction suivante contient une ou plusieurs erreurs. Le but est qu'elle renvoie `True` si la liste de tuples (supposée non-vide) passée en argument représente bien un chemin partant du coin haut gauche et constitué de coefficients adjacents, et `False` sinon. Recopier et corriger la fonction :

```
def est_chemin(L):
    # si L[0] n'est pas le coin haut gauche, c'est faux
    (i, j) = L[0]
    if i != 0 and j != 0:
        return False
    # puis il faut tester si les coefficients sont adjacents
    for k in range(len(L)):
        (i, j) = L[k]
        (i2, j2) = L[k+1]
        # teste si le coeffcient suivant est à droite ou en dessous
        if (i2 == i and j2 == j+1) or (i2 == i+1 and j2 == j):
            return True
        else:
            return False
```

9. Écrire une fonction `somme_chemin(A, L)` qui prend en argument une matrice `A` et un chemin `L` et qui renvoie la somme des coefficients de A le long du chemin donné.

4. Recherche du chemin

La méthode présentée permet de calculer la somme minimale des chemins joignant le coin haut gauche au coin bas droit, mais elle ne dit pas *quel* chemin donne ce minimum. Il pourrait d'ailleurs y en avoir plusieurs.
Pour trouver un chemin donnant une somme minimale, dans une matrice A donnée, on s'y prend de la façon suivante :

— On calcule la matrice S des sommes minimales,

— On démarre au coin bas droit de A. On note i et j les numéros de ligne et de colonnes actuels. Tant qu'on n'est pas remonté jusqu'au coin haut gauche alors :

  — 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 haut gauche au coefficient $a_{i-1,j}$ est inférieure à celle pour relier le coin haut gauche au coefficient $a_{i,j-1}$; autrement dit, pour arriver à $a_{i,j}$ le chemin de somme minimal provient du coefficient d'au dessus. On remonte alors d'une ligne.

  — Dans tous les autres cas, on remonte d'une colonne.

10. (a) Implémenter cet algorithme en écrivant une fonction `remonter_chemin(A)` qui prend en argument une matrice A et qui renvoie la liste des tuples qui donnent les coefficients parcourus par cet algorithme, partant du coin bas droit et remontant jusqu'au coin haut gauche.

    (b) En déduire une fonction `chemin(A)` qui renvoie un chemin de somme minimale dans A, rangé dans l'ordre depuis le coin haut gauche jusqu'au coin bas droit.

11. En appliquant ce procédé sur l'exemple 6 avec $B = \begin{pmatrix} 5 & 9 & 2 \\ 8 & 7 & 1 \\ 2 & 5 & 3 \end{pmatrix}$, donner un chemin de somme minimal pour B.