CC : Multiplication de matrices

Durée de l'épreuve : 2h00+15 minutes

En plus des 2 heures que durent l'épreuve, le site où déposer votre copie vous donne 15 minutes supplémentaires pour prendre en compte le temps de télécharger l'énoncé, de scanner votre copie, et de la déposer sur le site web.

Vous ne devez utiliser l'ordinateur que pour télécharger puis afficher l'énoncé et pour déposer votre copie. En particulier il vous est demandé de ne pas utiliser python pendant l'épreuve. Lorsqu'un programme en python est demandé, l'important n'est pas de savoir s'il s'exécute correctement sans erreur de syntaxe, mais si les opérations effectuées répondent à la question.

Pour la rédaction de votre copie, gardez bien en tête que l'important est de convaincre l'examinateur que vous avez compris pourquoi le résultat est correct (il arrive qu'écrire une demi ligne suffise pour cela, mais le plus souvent écrire uniquement le résultat ne suffit pas).

Dans ce sujet, on manipulera des matrices. Par simplicité, on les entre en python comme des listes de listes : chaque sous liste correspond à une ligne de la matrice.

Par exemple, la liste `[[1,2],[3,4]]` correspond à la matrice $\begin{pmatrix}1&2\\3&4\end{pmatrix}$.

De plus, les coefficients des matrices seront toujours réels, et les espaces vectoriels considérés sont des $\mathbb{R}$-espaces vectoriels.

I. Rappels d'algèbre linéaire

Matrices et endomorphismes

Soient V un espace vectoriel de dimension $n = \dim V \in \mathbb{N}^*$ et soit $\mathcal{B}_V = (v_0, v_1, \ldots, v_{n-1})$ une base de V.

Une application $u : V \to V$ est appelée "Endomorphisme" si

$$\forall x,y \in V^2, \forall \lambda \in \mathbb{R}, u(x + \lambda y) = u(x) + \lambda u(y)$$

Soient V un espace vectoriel de dimension $n = \dim V \in \mathbb{N}^*$.

Une application $u : V \to V$ est appelée "Endomorphisme" si

$$\forall x,y \in V^2, \forall \lambda \in \mathbb{R}, u(x + \lambda y) = u(x) + \lambda u(y)$$

On désigne par $\mathcal{L}(V)$ l'ensemble des endomorphismes de V, et par $\mathcal{M}_n(\mathbb{R})$ l'ensemble des matrices (à coefficients réels) ayant n lignes et n colonnes.

1. On rappelle que toute base $(v_0, v_1, \ldots, v_{n-1}) \in V^n$ satisfait :

$$\forall u \in \mathcal{L}(V), \exists ! M \in \mathcal{M}_n(\mathbb{R}) : \forall (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{R}^n, u\left(\sum_{i=0}^{n-1} a_i v_i\right) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} m_{i,j} a_j v_i.$$

(dans ce cas cette matrice M s'appelle "la matrice associée à u dans la base $(v_0, v_1, \ldots, v_{n-1})$".)
Démontrer que réciproquement toute famille $(v_0, v_1, \ldots, v_{n-1}) \in V^n$ qui satisfait cette condition est une base de V.

2. Exemples de matrices :

   (a) Donner une base $\mathcal{B}_0$ de l'espace vectoriel $\mathbb{R}_n[X]$ formé des polynômes P de degré $\deg(P) \leqslant n$.

   (b) Les applications $D : P \mapsto P' + X^n P(0)$ et $T : P \mapsto XP'$ sont des endomorphismes de $\mathbb{R}_n[X]$. (ici XP' désigne le produit du polynôme X par le polynôme P' et $X^n P(0)$ désigne le produit du polynôme X^n par le nombre $P(0)$.)
   Déterminer leur matrice dans la base $\mathcal{B}_0$.

Implémentation en python

3. Écrivez deux fonctions en python qui produisent les matrices que vous avez obtenues en question 2.(b) pour n'importe quelle valeur de $n \in \mathbb{N}^*$ (vos fonctions doivent renvoyer un résultat sous forme de listes de listes).

```
def matrice_de_D(n):
    .....

def matrice_de_T(n):
    .....
```

Produit de matrices

Soient $(v_0, v_1, \ldots, v_{n-1})$ une base de V, u et v des endomorphismes de V, et A et B les matrices associées à u et v dans cette base.

4. Démontrer que la matrice associée à l'endomorphisme $u \circ v$ (dans cette base) est la matrice $C = (c_{i,j})_{\substack{0 \leqslant i < n \\ 0 \leqslant j < n}}$ dont les coefficients sont donnés par

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}$$

Cette matrice C est alors notée $C = A \cdot B$ et est le produit des matrices A et B.

Matrices rectangulaires

Même si cet énoncé se focalise sur les matrices carrés, on admettra que la même formule permet de multiplier des matrices rectangulaires :

$\mathcal{M}_{n,p}(\mathbb{R})$ désigne les matrices ayant n lignes et p colonnes. Pour $(m,n,p) \in (\mathbb{N}^*)^3$, $A \in \mathcal{M}_{m,n}(\mathbb{R})$ et $B \in \mathcal{M}_{n,p}(\mathbb{R})$, le produit $A \cdot B$ est la matrice $C \in \mathcal{M}_{m,p}(\mathbb{R})$ dont les coefficients sont donnés par

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}$$

Complexité

Dans la suite, on supposera que dans le calcul du produit de matrices, le temps de calcul est principalement impacté par le nombre de multiplications de coefficients effectuées.

Pour estimer le temps de calcul des programmes, on calculera donc le nombre de multiplication de coefficients effectuées.

II. Premier algorithme

On essaie d'écrire une fonction qui calcule le produit de matrices carrés en utilisant la formule précédente.

On hésite entre les deux fonctions suivantes :

```
def mul1(A, B):
    n=len(A)
    return [ [ sum(A[i][k]*B[k][j] for k in range(n))
               for j in range(n) ]
             for i in range(n) ]

def mul2(A, B):
    n=len(A)
    return [ [ sum(A[i][k]*B[k][j] for k in range(n))
               for i in range(n) ]
             for j in range(n) ]
```

Remarque : ces deux fonctions diffèrent uniquement par une interversion des lettres `i` et `j` dans les `for i in` et `for j in`

5. Une seule des deux fonctions ci-dessus est correcte, laquelle ? On vous demande de justifier votre réponse, par exemple en expliquant ce que renvoie chacune des fonctions quand on calcule le produit de `matrice_de_D(1)` par `matrice_de_T(1)`.
6. Exprimer le nombre de multiplications effectuées pour calculer ainsi le produit de matrices, en fonctions de la dimension n.

III. Matrices par blocs

Définition

On appelle "matrice par blocs" une matrice obtenue par la juxtaposition de sous-matrices rectangulaires. Par exemple, si $A = \begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$, $B = \begin{pmatrix}5\\ 6\end{pmatrix}$, $C = \begin{pmatrix}7 & 8\end{pmatrix}$ et $D = \begin{pmatrix}9\end{pmatrix}$,
alors $\begin{pmatrix}A & B\\ C & D\end{pmatrix}$ désigne la matrice $\begin{pmatrix}1 & 2 & 5\\ 3 & 4 & 6\\ 7 & 8 & 9\end{pmatrix}$ (que l'on peut aussi noter $\left(\begin{array}{cc|c}1 & 2 & 5\\ 3 & 4 & 6\\ \hline 7 & 8 & 9\end{array}\right)$ pour rendre les blocs plus lisibles).

Une définition plus formelle est que si $A \in \mathcal{M}_{m,p}(\mathbb{R})$, $B \in \mathcal{M}_{m,q}(\mathbb{R})$, $C \in \mathcal{M}_{n,p}(\mathbb{R})$ et $D \in \mathcal{M}_{n,q}(\mathbb{R})$, alors

$$M = \begin{pmatrix}A & B\\ C & D\end{pmatrix} \in \mathcal{M}_{m+n,p+q}(\mathbb{R}) \text{ a pour coefficients } m_{i,j} = \begin{cases} a_{i,j} & \text{si } i < m \text{ et } j < p\\ b_{i,j-p} & \text{si } i < m \text{ et } j \geqslant p\\ c_{i-m,j} & \text{si } i \geqslant m \text{ et } j < p\\ d_{i-m,j-p} & \text{si } i \geqslant m \text{ et } j \geqslant p. \end{cases}$$

Produit de matrices par blocs

Soient $A \in \mathcal{M}_{n,n}(\mathbb{R})$, $B \in \mathcal{M}_{n,p}(\mathbb{R})$, $C \in \mathcal{M}_{p,n}(\mathbb{R})$, $D \in \mathcal{M}_{p,p}(\mathbb{R})$, $E \in \mathcal{M}_{n,n}(\mathbb{R})$, $F \in \mathcal{M}_{n,p}(\mathbb{R})$, $G \in \mathcal{M}_{p,n}(\mathbb{R})$, et $H \in \mathcal{M}_{p,p}(\mathbb{R})$.

7. Montrer que
$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

Matrices de taille $2^k \times 2^k$

En posant $B = 0$, $C = 0$, $D = 0$, $F = 0$, $G = 0$, et $H = 0$, l'égalité précédente devient $\begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} E & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} AE & 0 \\ 0 & 0 \end{pmatrix}$, qui permet de se ramener à des multiplications de matrices de taille $2^k \times 2^k$. (Notez que le symbole "0" a ici été utilisé pour désigner à chaque fois une matrice dont tous les coefficients sont nuls.)

En effet si on suppose qu'on a déjà écrit des fonctions `plus_petit_k`, `augmente_taille`, `sous_matrice` et `mul0` telles que

- Pour $n \in \mathbb{N}^*$, `plus_petit_k(n)` renvoie le plus petit entier k tel que $n \leqslant 2^k$.
- Si $A \in \mathcal{M}_{p,q}(\mathbb{R})$ et $n \geqslant \max(p,q)$ alors `augmente_taille(A,n)` renvoie la matrice $\begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix}$ de taille $n \times n$.
- Si $A \in \mathcal{M}_{m,n}(\mathbb{R})$ avec $m \geqslant p$ et $n \geqslant q$, alors `sous_matrice(A,p,q)` renvoie la sous-matrice de A constituée des p premières lignes des q premières colonnes.
- S'il existe $k \in \mathbb{N}$ tel que A et B sont toutes deux de taille $2^k \times 2^k$, alors `mul0(A,B)` renvoie la matrice $C = A \cdot B$.

Alors on peut définir ci-dessous la multiplication de matrices de tailles arbitraires :

```
def mul(A,B):
    p=len(B)
    m=len(A)
    n=len(B[0])
    K=max(plus_petit_k(d) for d in [p,m,n])
    A0=augmente_taille(A,2**K)
    B0=augmente_taille(B,2**K)
    C0=mul0(A0,B0)
    return sous_matrice(C0,m,n)
```

8. Écrire en python la fonction `sous_matrice` décrite précédemment.

Premier algorithme de type "Diviser pour mieux régner"

L'algorithme ci-dessous calcule le produit de deux matrices carrées de même taille $2^k \times 2^k$:

```
def mul0(M1,M2):
    if len(M1)==1 : #c'est à dire que k=0
        return [[M1[0][0]*M2[0][0]]]
    (A,B,C,D)=blocs(M1)
    (E,F,G,H)=blocs(M2)
    return mat_blocs( mul0(A,E)+mul0(B,G) , mul0(A,F)+mul0(B,H),
                      mul0(C,E)+mul0(D,G) , mul0(C,F)+mul0(D,H)  )
```

Il fonctionne à condition d'avoir déjà écrit des fonctions `blocs` et `mat_blocs` telles que

- Si M est de taille $2^k \times 2^k$, alors `blocs(M)` renvoie (A,B,C,D) tel que $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ (avec A, B, C, et D de taille $2^{k-1} \times 2^{k-1}$).
- Si A, B, C, et D sont des matrices de taille $2^k \times 2^k$, alors `mat_blocs(A,B,C,D)` renvoie la matrice $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$, de taille $2^{k+1} \times 2^{k+1}$.

9. Écrire en python la fonction `blocs` décrite précédemment.
10. On suppose que les fonctions `blocs` et `mat_blocs` sont suffisamment rapides pour ne pas impacter le temps de calcul (en particulier elles n'effectuent pas de multiplications de coefficients).

    On désigne par N_p le nombre de multiplications de coefficients qu'effectue le programme `mul0` ci-dessus pour multiplier deux matrices de taille $2^p \times 2^p$.

    (a) Déterminer une relation de récurrence satisfaite par la suite $(N_p)_{p \in \mathbb{N}^*}$.

    (b) En déduire l'expression de N_p.
11. Pour des matrices de grande taille, ce programme est-il plus rapide (ou plus lent) que celui de la partie II ?

Deuxième algorithme de type "Diviser pour mieux régner"

Afin de réduire le nombre de multiplication, on cherche à améliorer cet algorithme en évitant de calculer indépendemment les produits AE, BG, AF, BH, CE, DG, CF et DH (qui nécessitent chacun de multiplier de nombreux coefficients).

Pour cela on pose

$$M_1 = (A+D)\cdot(E+H) \quad M_2 = (C+D)\cdot E \quad M_3 = A\cdot(F-H) \quad M_4 = D\cdot(G-E)$$
$$M_5 = (A+B)\cdot H \quad M_6 = (C-A)\cdot(E+F) \quad M_7 = (B-D)\cdot(G+H)$$

Et on s'appuie sur la remarque que la matrice

$$M = \begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{pmatrix}$$

est précisément égale à la matrice $\begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot \begin{pmatrix} E & F \\ G & H \end{pmatrix}$

12. Démontrer que pour le bloc supérieur droit, l'affirmation précédente est correcte, c'est à dire que M_3+M_5 est bien égale au bloc supérieure droit du produit de matrice $\begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot \begin{pmatrix} E & F \\ G & H \end{pmatrix}$.
13. Montrer la même propriété pour le bloc inférieur gauche.

On pourrait le démontrer pour les autres blocs de la matrice, mais pour gagner du temps, on admet simplement que cela fonctionne aussi pour les autres blocs de la matrice M.

14. Réécrire la fonction `mul0` définie précédemment, de telle sorte qu'au lieu de calculer les produits AE, BG, AF, BH, CE, DG, CF et DH, elle utilise les matrices M_1, M_2, …, M_7.
15. On désigne par N'_p le nombre de multiplications de coefficients qu'effectue cette nouvelle version de `mul0` pour multiplier deux matrices de taille $2^p \times 2^p$.

(a) Déterminer une relation de récurrence satisfaite par la suite $\left(N'_p\right)_{p\in\mathbb{N}^*}$.

(b) En déduire l'expression de N'_p.

(c) Pour des matrices de grande taille, ce programme est-il plus rapide (ou plus lent) que le précédent et/ou que celui de la partie II ?