Devoir surveillé numéro 7

*Devoir du **23 mai 2024**.*

Exercice 1 *Questions proche du cours.*

1. On pose $X_1 = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}, X_2 = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$ et $X_3 = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix}$.

   Démontrer que l'ensemble $F = \{(\lambda, \mu, \gamma) \in \mathbb{R}^3 | \lambda X_1 + \mu X_2 + \gamma X_3 = 0_{3,1}\}$ est un sous-espace vectoriel de $\mathbb{R}^3$, en donner une base et déterminer sa dimension.

2. Soit $n \in \mathbb{N}$. Soit $A \in \mathcal{M}_n(\mathbb{R})$. Démontrer que l'ensemble $E = \{M \in \mathcal{M}_n(\mathbb{R}) | AM = MA\}$ est un sous-espace vectoriel de $\mathcal{M}_n(\mathbb{R})$.

3. On lance indéfiniment une pièce à pile ou face ayant la probabilité $p \in]0,1[$ de faire Pile. Ces lancers sont considérés indépendants. Démontrer que l'événement "ne jamais faire Pile" est négligeable.

4. Soit $p \in]0,1[$, $n \in \mathbb{N}^*$ et X une variable aléatoire sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$. Définir : la variable aléatoire X suit la loi binomiale de paramètres n et p. Énoncer les résultats relatifs à l'espérance et à la variance d'une telle variable aléatoire, et démontrer le résultat portant sur l'espérance.

5. (a) Énoncer l'inégalité des accroissements finis.

   (b) Soient x et y deux réels tels que $1 < x < y$. Montrer que : $\ln(\frac{y}{x}) \leq y - x$.

6. Démontrer le résultat du cours suivant : Soient a et b deux réels tels que $a \leq b$, et $f : [a,b] \to \mathbb{R}$ une fonction continue et positive. Alors, $\int_a^b f(x)dx \geq 0$.

Exercice 2 *Analyse*

Pour tout couple (p, q) d'entiers naturels, on pose $I(p,q) = \displaystyle\int_0^1 x^p(1-x)^q \, \mathrm{d}x$.

1. (a) Expliquer rapidement pourquoi cette intégrale est bien définie.

   (b) Montrer, grâce à une intégration par parties, que l'on a :

$$\forall (p,q) \in \mathbb{N} \times \mathbb{N}^*, I(p,q) = \frac{q}{p+1} I(p+1, q-1)$$

On admet que l'on peut en déduire par récurrence l'égalité :

$$\forall (p,q) \in \mathbb{N} \times \mathbb{N}, I(p,q) = \frac{p!q!}{(p+q)!} I(p+q, 0)$$

2. (a) Pour tout couple (p, q) d'entiers naturels, déterminer $I(p+q, 0)$ puis exprimer $I(p, q)$ en fonction de p et q.

   (b) Montrer enfin que : $\forall p \in \mathbb{N}, \displaystyle\int_0^1 x^p(1-x)^p \, \mathrm{d}x = \frac{(p!)^2}{(2p+1)!}$.

Pour tout entier naturel n, on pose :

- $\alpha_n = \dfrac{(2n+1)!}{(n!)^2}$, et
- $f_n : x \mapsto \alpha_n \displaystyle\int_0^x t^n(1-t)^n \, \mathrm{d}t$.

On note aussi (C_n) la courbe représentative de f_n dans un repère orthonormé $(O, \vec{i}, \vec{j})$.

3. Justifier que, pour tout entier naturel n, la fonction f_n est définie sur $\mathbb{R}$.
4. Déterminer $f_0(x)$ pour tout réel x.
5. (a) Donner la valeur de $f_n(1)$.

   (b) Montrer, grâce au changement de variable $u = 1 - t$ dans l'intégrale définissant $f_n(x)$, que :

$$\forall x \in \mathbb{R}, f_n(x) + f_n(1-x) = 1$$

   (c) En déduire la valeur de $f_n\left(\frac{1}{2}\right)$.

6. (a) Montrer que f_n est dérivable sur $\mathbb{R}$ et déterminer, pour tout réel x, l'expression de $f_n'(x)$ en fonction de x et n.

   (b) Étudier, suivant la parité de n, le signe de $f_n'(x)$ pour tout réel x.

7. (a) En utilisant éventuellement la formule du binôme de Newton, montrer que f_n est une fonction polynomiale puis en déduire les valeurs de $\lim_{x\to+\infty} f_n(x)$ et de $\lim_{x\to-\infty} f_n(x)$ selon que n est pair ou impair.

   (b) Dresser le tableau de variations de f_n (toujours en distinguant les cas n pair et n impair).

8. Dans cette question, n désigne un entier naturel supérieur ou égal à 1.

   (a) Pour tout réel x, déterminer $f_n''(x)$ en fonction de x et n.

   (b) En déduire que (C_n) possède un point d'inflexion si n est impair et trois si n est pair.

   (c) Tracer, selon la parité de n, l'allure de (C_n).

Exercice 3 *Probabilités*

On dispose de trois pièces indiscernables au toucher :

- une pièce numérotée 0 donnant Pile avec une probabilité $\frac{1}{2}$ et Face avec une probabilité $\frac{1}{2}$
- une pièce numérotée 1 donnant Pile à coup sûr
- une pièce numérotée 2 donnant Face à coup sûr

L'expérience consiste à choisir de façon équiprobable l'une de ces trois pièces puis à la lancer indéfiniment.

On note $(\Omega, \mathcal{A}, \mathbb{P})$ un espace probabilisé associé à cette expérience, et on suppose bien-sûr les différents lancers d'une même pièce indépendants.

Pour $i \in \{0;1;2\}$, on note A_i l'évènement "on a choisi la pièce numérotée i ".

Pour tout $k \in \mathbb{N}^*$, on note P_k l'évènement : "on obtient Pile au lancer numéro k " et $\mathrm{F}_k = \overline{\mathrm{P}_k}$.

On considère les deux variables aléatoires :

- X donnant le rang d'apparition du premier Pile
- Y donnant le rang d'apparition du premier Face

On convient de donner à X la valeur 0 si l'on n'obtient jamais Pile et de donner à Y la valeur 0 si l'on n'obtient jamais Face.

1. *Loi de X.*
   (a) Donner $\mathrm{X}(\Omega)$. *On se contentera d'une brève justification.*
   (b) Déterminer $\mathbb{P}([\mathrm{X}=1])$.
   (c) Montrer que : $\forall n \in [\![2;+\infty[\![, \mathbb{P}([\mathrm{X}=n]) = \frac{1}{3}\left(\frac{1}{2}\right)^n$.
   (d) En déduire la valeur de $\mathbb{P}([X=0])$.
2. Montrer que X admet une espérance et la calculer. Interpréter le résultat obtenu.
3. Montrer que X admet une variance et la calculer.
4. Justifier que Y suit la même loi que X.
5. (a) Montrer que pour tout $j \in [\![2;+\infty[\![, \mathbb{P}([\mathrm{X}=1] \cap [\mathrm{Y}=j]) = \mathbb{P}([\mathrm{Y}=j])$.
   (b) Montrer que pour tout $i \in [\![2;+\infty[\![, \mathbb{P}([\mathrm{X}=i] \cap [\mathrm{Y}=1]) = \mathbb{P}([\mathrm{X}=i])$.
   (c) Les variables aléatoires X et Y sont-elles indépendantes?
6. On considère la variable aléatoire $Z = X + Y$.
   (a) Expliquer pourquoi Z prend toutes les valeurs entières positives sauf 0 et 2.
   (b) Montrer que $\mathbb{P}([Z=1]) = \frac{2}{3}$.
   (c) Justifier que pour tout entier naturel n supérieur ou égal à 3 , on a :
   $$[\mathrm{Z}=n] = ([\mathrm{X}=1] \cap [\mathrm{Y}=n-1]) \cup ([\mathrm{Y}=1] \cap [\mathrm{X}=n-1])$$
   (d) En déduire que :
   $$\forall n \in [\![3;+\infty[\![, \mathbb{P}([Z=n]) = \frac{2}{3}\left(\frac{1}{2}\right)^{n-1}$$

Exercice 4 *Graphes et réseaux sociaux*

Pour analyser un réseau social, on peut former son graphe dans le but d'utiliser l'outil informatique pour avoir des résultats. Dans cet exercice, on considérera un réseau social "symétrique" : deux individus sont ou bien amis, ou bien non amis. On considérera de plus qu'un individu n'est pas ami avec lui même.

Le graphe d'un tel réseau social est construit de la manière suivante :

- Les sommets de ce graphes sont les individus du réseau social (les différents comptes).
- Deux sommets i et j sont reliés par une arête si et seulement si i et j sont amis.

On rappelle qu'un graphe est dit *simple* s'il est sans boucle (et sans multi-arête). Le graphe associé à un tel réseau social est donc simple et non orienté. Deux sommet d'un graphe sont dits *voisins* s'il existe une arête entre ces sommets.

Dans tout cet exercice, les graphes considérés sont simples et non orientés.

Dans les questions d'informatique, on supposera les sommets des graphes considérés numérotés à partir de 0, et on pourra donc parler de *l*a matrice d'adjacence d'un graphe. On identifiera librement les sommets à leur numéro.

Dans les questions d'informatique, on supposera le module `numpy` importé à l'aide de la commande :

```
import numpy as np
```

Une annexe informatique est présente en fin d'exercice, contenant des rappels sur les commandes Python.

Partie I : Théorie des graphes et modélisation

Soit G un graphe non orienté. Notons n son ordre, et numérotons $s_1, \dots, s_n$ ses sommets.

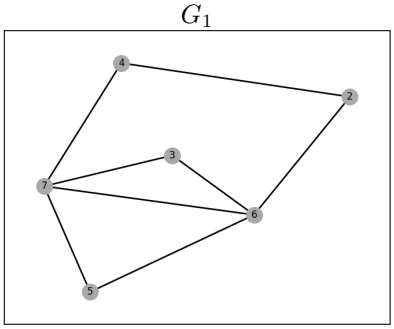
1. (*a*) Définir la matrice d'adjacence M de G associée à la numérotation $s_1, \dots, s_n$ des sommets de G.

   (*b*) Rappeler sans démonstration le lien entre M^k et les chaînes de longueur k de G (pour $k \in \mathbb{N}$).

   (*c*) Démontrer que si deux sommets s_i et s_j de G sont reliés par une chaîne, alors ils sont reliés par une chaîne de longueur au plus $n-1$ (où $1 \leq i, j \leq n$).

2. Déterminer les ensembles S et A tels que le graphe représenté ci-dessous, noté G_1 dans la suite, soit le graphe (S, A).

G_1



3. Soit $n \in \mathbb{N}$. On appelle graphe complet d'ordre n le graphe K_n dont les sommets sont les entiers de 0 à $n-1$, et tel :
   - K_n est un graphe simple non orienté, et
   - Deux sommets distincts de K_n sont toujours reliés par une arête.

   (*a*) Déterminer la matrice d'adjacence M de K_4.

   (*b*) Écrire le code d'une fonction `Python` d'entête `def K(n):` prenant en entrée un entier `n` et renvoyant en sortie la matrice d'adjacence de K_n.

(c) On pose $J = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$. Calculer J^k pour tout entier naturel k.

(d) Montrer que M est combinaison linéaire de J et de la matrice identité I_4.

(e) En déduire le calcul de M^k pour tout entier naturel k, à l'aide de la formule du binôme de Newton.

Une manière assez naïve d'analyser un réseau social afin d'en dégager des sources d'influence est de considérer que plus un individu a d'amis, plus il est influent.

4. Écrire le code d'une fonction `Python` d'entête `def degre(M,i):` prenant en entrée la matrice d'adjacence `M` d'un graphe et le numéro `i` de l'un de ses sommets et renvoyant en sortie le degré de ce sommet.

5. En déduire la code d'une fonction d'entête `def listeDegMax(M):` prenant en entrée la matrice d'adjacence `M` d'un graphe et renvoyant en sortie la liste des numéros des sommets de ce graphe dont le degré est maximal.

Partie II : Ensemble dominant de sommets

Dans le but de répandre une fake-news (ou, de manière équivalente, une publicité) sur les fils de publication, on peut chercher à identifier certains individus tel que : tout individu du réseau social est ami avec au moins un des individus identifiés.

En terme de théorie des graphes, cela donne la définition suivante :

Définition 1. Soit G un graphe non orienté, notons S l'ensemble de ses arêtes. On dit qu'un ensemble de sommets $A \subset S$ est *dominant* si tout sommet de G est ou bien élément de A, ou bien voisin d'un élément de A.

Dans un objectif d'efficacité, on cherche à déterminer un ensemble de sommets dominant ayant le moins d'éléments possibles. On dira dans ce cas que l'ensemble dominant trouvé est minimal.

6. Déterminer un ensemble de sommets dominant minimal pour le graphe G_1 de la partie I (on démontrera le caractère minimal de celui-ci).

7. Écrire une fonction `Python` d'entête `def estVoisin(M,i,j):` prenant en entrée la matrice d'adjacence `M` d'un graphe et les numéros `i` et `j` de deux de ses sommets, et renvoyant en sortie `True` si ces sommets sont voisins dans ce graphe, et `False` sinon.

8. Que fait la fonction `Mystere` suivante, prenant en entrée la matrice d'adjacence `M` d'un graphe, le numéro `j` de l'un de ses sommets et une liste `A` constituée de (numéros de) sommets de ce graphes ?

```
def Mystere(M,j,A):
        if j in A:
                return(True)
        for a in A:
                if estVoisin(M,j,a)==True:
                        return(True)
        return(False)
```

9. En déduire une fonction d'entête `def estDominant(M,A):` prenant en entrée la matrice d'adjacence `M` d'un graphe et une liste `A` constituée de sommets de ce graphes, et renvoyant en sortie `True` si `A` est la liste des sommets d'un ensemble dominant, et `False` sinon.

10. En déduire une fonction d'entête `def minimiseDominant(M,A):` prenant en entrée la matrice d'adjacence `M` d'un graphe et la liste `A` des sommets d'un ensemble dominant, et renvoyant en sortie :

    - Une liste `L` obtenue à partir de `A`, et sans modifier `A`, en retirant un de ses éléments de sorte que `L` soit toujours la liste des sommets d'un ensemble dominant, si c'est possible,
    - `False` si aucun sommet ne peut être enlevé à la liste `A` sans obtenir une liste de sommets d'un ensemble dominant.

11. Expliquer le code suivant, dans lequel `M` est la matrice d'adjacence d'un graphe.

```
n,p=np.shape(M)
L=[i for i in range(n)]
while minimiseDominant(M,L)!= False:
        L=minimiseDominant(M,L)
print(L)
```

Partie III : Centralité de proximité

On peut détecter des sources d'influence avec un indicateur appelé le degré de proximité. L'idée est de dire que plus un individu est proche d'un grand nombre d'individus, plus il est influent.

Dans cette partie, on fixe un graphe G d'ordre n. On suppose que l'ensemble des sommets de G est $[\![0, n-1]\!]$ et on note `M` sa matrice d'adjacence. Si i et j sont deux sommets de G, on note $d(i,j)$ la longueur de la plus petite chaîne reliant les sommets i et j si une telle chaîne existe, et on pose $d(i,j) = +\infty$ si i et j ne sont pas reliés par une chaine. En particulier, $d(i,i) = 0$.

On dit que $d(i,j)$ est la distance du sommet i au sommet j.

Enfin, pour tout sommet i, on pose :

$$D(i) = \sum_{\substack{j=0 \\ j\neq i}}^{n-1} \frac{1}{d(i,j)}$$

avec la convention $\dfrac{1}{+\infty} = 0$.

On dit que $D(i)$ est le degré de proximité du sommet i.

12. (a) On se place dans le cas du graphe G_1 de la partie I. Déterminer $D(2)$ sans justification.

    On revient au cas général.

    (b) Montrer que pour tout sommet i, $0 \leq D(i) \leq n-1$.

    (c) Que dire du sommet i si $D(i) = 0$? Que dire si $D(i) = n-1$?

    (d) Soient i, j et k trois sommets de G.

    Démontrer que $d(i,j) \leq d(i,k) + d(k,j)$ avec la convention $+\infty \leq +\infty$ et $x \leq +\infty$ pour tout réel x.

13. A l'aide de résultats de la partie I, écrire une fonction d'entête `def dist(M,i,j):` prenant en entrée la matrice `M` et les numéros `i` et `j` de deux sommets de G, et renvoyant en sortie $d($`i`$,$`j`$)$ si ces sommets sont reliés par une chaîne, et -1 sinon.

    On justifiera que le code proposé est correcte.

14. En déduire le code d'une fonction d'entête `def D(M,i):` prenant en entrée la matrice `M` et le numéro `i` d'un des sommets de G, et renvoyant en sortie le degré de proximité $D($`i`$)$ du sommet `i`.

15. Écrire le code d'une fonction d'entête `def ClassementCentralite(M):` prenant en entrée la matrice `M` et renvoyant en sortie la liste des sommets de G triée par degré de proximité décroissant.

    On pourra s'inspirer du code suivant, triant une liste de nombres `L` préalablement définie dans l'ordre décroissant :

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

Partie IV : Détection de Cliques

Définition 2. Soit $G = (S, A)$ un graphe non orienté. On dit qu'un ensemble de sommets $A \subset S$ est *une clique* si pour tous sommets i et j distincts et appartenant à A, ces sommets sont voisins.

Dans un graphe, une clique est donc la donnée de sommets qui sont "tous voisins entre eux". Identifier les cliques du graphe d'un réseau social permet de détecter des communautés (des groupes d'individus partageant un même intérêt).

16. Déterminer une clique de cardinal 3 du graphe G_1 de la partie I. Donner sans justification la matrice d'adjacence du sous-graphe induit par cette clique. Que remarque-t-on?

    On rappelle que le sous graphe induit par un ensemble S' de sommets d'un graphe G est le sous-graphe obtenu à partir de G en ne gardant que les sommets de S', et en gardant toutes les arêtes de G entre les sommets de S'.

17. Écrire le code d'une fonction d'entête `def gardeLignes(M,L):` prenant en entrée une matrice `M` et une liste `L` formée d'entiers distincts entre 0 et $n-1$, où n est le nombre de lignes de `M`, et renvoyant en sortie la matrice obtenue à partir de `M` en ne gardant que les lignes dont les indices sont présents dans `L`.

    Par exemple, si `M` est la matrice $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 5 & 6 \\ 5 & 6 & 7 & 8 \end{pmatrix}$ et si `L=[0,2]`, alors cette fonction devra renvoyer la matrice

    $$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

18. Compléter le code suivant permettant de définir une fonction d'entête `def gardeColonnes(M,L):` effectuant la même opération que dans la question précédente, mais sur les colonnes de `M`.

```
def gardeColonnes(M,L):
        A=gardeLignes(np.transpose(M),L)
        return(...)
```

19. Écrire une fonction d'entête `def sousMatrice(M,L):` prenant en entrée une matrice carrée `M` et une liste `L` formée d'entiers distincts entre 0 et $n-1$, où n est la taille de `M`, et renvoyant en sortie la matrice obtenue à partir de `M` en ne gardant que les lignes et les colonnes dont les indices sont présents dans `L`.

    Par exemple, si `M` est la matrice $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 5 & 6 & 7 & 8 \\ 6 & 7 & 8 & 9 \end{pmatrix}$ et si `L= [0,3]`, alors cette fonction devra renvoyer la matrice

    $$\begin{pmatrix} 1 & 4 \\ 6 & 9 \end{pmatrix}$$

20. En déduire le code d'une fonction d'entête `def verifieClique(M,L):` prenant en entrée la matrice d'adjacence `M` d'un graphe et une liste `L` de sommets (distincts) de ce graphe, et renvoyant en sortie `True` si `L` est la liste des sommets d'une clique, et `False` sinon.

Annexe informatique

Si `L` est une liste de longueur n :

- `del L[i]` modifie `L` en enlevant l'élément en `i`-ième position (pour $0 \leq \texttt{i} \leq n-1$)
- `L.copy()` renvoie une copie indépendante de la liste `L`

La fonction `np.array` transforme une liste en matrice (de type `np.ndarray`). Par exemple, la matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ est donnée en Python par :

```
np.array([[1,2],[3,4]])
```

Dans la commande ci-dessus, les lignes `[1,2]` et `[3,4]` peuvent aussi être données avec le type `np.ndarray`.

Si `M` est une matrice de taille `(n,p)` :

- `np.shape(M)` renvoie le couple `(n,p)`
- `M[i,:]` renvoie la matrice qui est la `i`-ième ligne de `M` (sous réserve de bonne définition)
- `M[:,i]` renvoie la matrice qui est la `i`-ième colonne de `M`
- `np.transpose(M)` renvoie la transposée de M.
- Si `N` est une autre matrice, la commande `np.dot(M,N)` renvoie le résultat du produit matriciel `MN` si celui-ci est bien défini.
- Si `N` est une autre matrice, la commande `(M==N).all()` renvoie `True` si `M` et `N` sont égales, `False` sinon.
- `np.eye(n)` renvoie la matrice identité de taille `n`.
- `np.ones((n,p))` renvoie la matrice de taille $(\texttt{n}, \texttt{p})$ dont tous les coefficients valent 1.

Si le module numpy.linalg est importé à l'aide de la commande `import numpy.linalg as al` :

- `al.matrix_power(M,k)` renvoie la puissance $\texttt{M}^{\texttt{k}}$ de `M`.

— Fin de l'énoncé —