E C G

DS n°4

Option économique

MATHEMATIQUES

20 Janvier 2024

La présentation, la lisibilité, l'orthographe, la qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'appréciation des copies.
Les candidats sont invités à ***encadrer*** *dans la mesure du possible les résultats de leurs calculs.*
Ils ne doivent faire usage d'aucun document ; l'utilisation de toute calculatrice et de tout matériel électronique est interdite. Seule l'utilisation d'une règle graduée est autorisée.
Si au cours de l'épreuve un candidat repère ce qui lui semble une erreur d'énoncé, il le signalera sur sa copie et poursuivra sa composition en expliquant les raisons des initiatives qu'il sera amené à prendre.

On suppose importés les modules suivants :

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
import numpy.linalg as al
```

Exercice n°1

On considère une liste `L`.

1. Créer une fonction `appartient(L,x)` qui teste si `x` appartient à `L`.
2. Écrire une fonction `comptage(L,x)` qui compte le nombre d'occurrences de x dans L.

Exercice n°2

Une urne contient initialement a boules blanches et b boules noires. On effectue des tirages successifs selon le protocole suivant :

- Si on obtient, à un rang quelconque, une boule blanche, celle-ci est remise dans l'urne avant le tirage suivant.
- Si on obtient, à un rang quelconque, une boule noire, celle-ci est jetée et remplacée par une boule blanche avant le tirage suivant.

1. Écrire une fonction Python `X(n,a,b)` simulant le variable aléatoire égale au nombre de boules noires obtenues au cours des n premiers tirages.
2. Écrire une fonction `Y(a,b)` simulant la variable égale au nombre de tirages effectués pour que l'urne ne contienne plus que des boules blanches.

Exercice n°3

1. Expliquer ce que font les commandes suivantes :

   (a) `np.eye(n)`
   (b) `np.zeros([n,p])`
   (c) `np.zeros(n)`
   (d) `np.ones([n,p])`
   (e) `np.arange(debut,fin,pas)`
   (f) `np.linspace(debut,fin,nbre)`

2. On note A et P les matrices

$$A = \begin{pmatrix} \frac{1}{3} & 0 & -\frac{2}{3} & 0 & -\frac{2}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & 1 & \frac{1}{3} \\ -\frac{2}{3} & 0 & \frac{1}{3} & 0 & -\frac{2}{3} \\ \frac{1}{3} & 1 & \frac{1}{3} & 0 & \frac{1}{3} \\ -\frac{2}{3} & 0 & -\frac{2}{3} & 0 & \frac{1}{3} \end{pmatrix} \qquad P = \begin{pmatrix} -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 \end{pmatrix}$$

   Ecrire un programme d'en tête `def mat(n):` dans lequel on définira les matrices A et P et qui renvoie l'inverse de A, le produit PAP^{-1} et A^n.

Exercice n°4

On donne la suite (u_n) définie par : $u_1 = 1 \quad$ et $\quad \forall n \in \mathbf{N}^* \quad u_{n+1} = \sqrt{4 + 3u_n}$

1. Écrire un programme en Python permettant de définir la fonction $f : x \mapsto \sqrt{4+3x}$ et de représenter sur un même graphique la fonction f et la droite $y = x$.
2. Écrire une fonction Python `def U(n):` qui renvoie u_n.
3. Montrer (mathématiquement) que :

$$\forall n \in \mathbf{N}^* \quad u_n < u_{n+1} < 4$$

   En déduire que (u_n) converge et déterminer sa limite ℓ

4. Écrire un programme Python déterminant le rang n_0 à partir duquel on a :

$$n \geqslant n_0 \Longrightarrow 3,9999 < u_n < 4$$

Exercice n°5

Soit $(F_n)_{n\geqslant 0}$ la suite définie par

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ \forall n \in \mathbb{N}, \quad F_{n+2} = F_{n+1} + F_n \end{cases}$$

Partie 1

1. (a) Montrer que pour tout entier naturel n non nul, $F_n > 0$.

   (b) Montrer que la suite $(F_n)_{n\geqslant 0}$ est strictement croissante à partir du rang 2 .
   (c) Pour tout entier n non nul, donner une expression de F_n uniquement en fonction de n.
   (d) En déduire que $(F_n)_{n\geqslant 0}$ diverge et que $\lim\limits_{n\to+\infty} F_n = +\infty$.

2. Écrire une fonction Python fibo(n) prenant en argument un entier naturel n et renvoyant la valeur de F_n.

Si on exécute le script Python suivant

```
L=[]
for k in range(20):
        L.append(fibo(k))
print(L)
```

on doit obtenir :

$$[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181]$$

3. Écrire une fonction `recherche (x, L)` prenant en entrée :

- un entier naturel x
- une liste L déjà triée dans l'ordre croissant, dont le premier élément est inférieur ou égal à x et le dernier est strictement supérieur à x

et qui renvoie le plus grand élément de la liste L qui soit inférieur ou égal à x.

Partie 2

On s'intéresse dans cette partie au théorème suivant, appelé Théorème de Zeckendorf

Pour tout entier naturel non nul n, il existe un unique entier k et un unique k-uplet d'entiers $(c_1, \dots, c_k)$, verifiant :

$\star c_1 \geqslant 2$

$\star$ pour tout i appartenant à $[\![1, k-1]\!], c_i + 1 < c_{i+1}$,

tels que :

$$n = \sum_{i=1}^{k} F_{c_i}$$

Cette décomposition s'appelle la décomposition de Zeckendorf du nombre n.

Par exemple, $n = 4$ se décompose en : $4 = 1 + 3 = F_2 + F_4$. Donc $k = 2$ et $(c_1, c_2) = (2, 4)$.

Par ailleurs, $n = 17$ se décompose en : $17 = 1 + 3 + 13 = F_2 + F_4 + F_7$. Donc $k = 3$ et $(c_1, c_2, c_3) = (2, 4, 7)$.

4. On rappelle que la liste des premiers termes de la suite $(F_n)_{n\geqslant 0}$ a été donnée dans la question 2 .

(a) En remarquant que $6 = 1+2+3 = F_2+F_3+F_4$ et aussi que $6 = 1+5 = F_2+F_5$, donner la décomposition de Zeckendorf de 6 et justifier votre choix.

(b) Donner la décomposition de Zeckendorf du nombre 35.

(c) Donner la décomposition de Zeckendorf du nombre 130.

5. Soit $n \in \mathbb{N}^*$.

(a) Justifier l'existence d'un entier J supérieur ou égal à 2 tel que $\forall i \geqslant J, F_i \geqslant n + 1$.

(b) Notons $A_n = \{i \in \mathbb{N}^*, F_i \leqslant n\}$.

Montrer que $2 \in A_n$ et que A_n contient au plus J éléments.

(c) Soit alors $j = \max(A_n)$, c'est-à-dire j est le plus grand entier appartenant à A_n. Montrer que $j \geqslant 2$ et que

$$F_j \leqslant n < F_{j+1}$$

(d) Démontrer que $n - F_j < F_{j-1}$.

(e) Supposons qu'il existe un entier k' et un k'-uplet $\left(c'_1, \dots, c'_{k'}\right)$ d'entiers naturels tels que

$$c'_1 \geqslant 2, \quad \forall i \in [\![1, k'-1]\!], c'_i + 1 < c'_{i+1} \quad \text{et} \quad n - F_j = \sum_{i=1}^{k'} F_{c'_i}$$

Montrer qu'il existe un entier k que l'on exprimera à l'aide de k' et qu'il existe un k-uplet $(c_1, \dots, c_k)$ d'entiers naturels tels que

$$c_1 \geqslant 2, \quad \forall i \in [\![1, k-1]\!], c_i + 1 < c_{i+1} \quad \text{et} \quad n = \sum_{i=1}^{k} F_{c_i}$$

6. Que renvoie la fonction suivante ?

```
def Zeckendorf(n):
        i=0
        L=[fibo(i)]
        while L[-1]<=n:
                i=i+1
                L.append(fibo(i))
        k=n
        T=[]
        while k>0:
                f=recherche(k,L)
                T.append(f)
                k=k-f
        return(T)
```

7. En quoi l'algorithme précédent est-il un algorithme glouton ?

Problème

Introduction

Le but est d'étudier une méthode pour créer des graphes aléatoires. Les sommets d'un graphe à n sommets seront toujours numérotés par l'ensemble $[\![0, n-1]\!] = \{0, 1, \ldots, n-1\}$
Tous les programmes et fonctions demandées doivent être écrits en python. Veuillez marquer de façon claire les indentations. On considère que les modules `numpy, numpy.random` ont été importés avec les commandes.

```
import numpy as np
import numpy.random as rd
```

Dans le module `numpy.random` on utilisera la fonction `random()` qui renvoie un réel entre 0 et 1 choisi de façon uniforme. Les graphes seront implémentés par leur matrice d'adjacence. Lorsque l'on écrira des fonctions, on admettra, sans le verifier, que les matrices sont bien des matrices d'adjacence.

Même si vous n'avez pas réussi à écrire une fonction demandée dans une question, vous pouvez l'utiliser dans les questions suivantes

Préliminaires

1. Quelles sont les caractéristiques d'une matrice d'adjacence d'un graphe non orienté sans boucle.
2. Rappeler la commande qui renvoie une matrice à n et lignes n colonnes dont tous les coefficients sont nuls.
3. Écrire une fonction `verification(M):` qui vérifie si la matrice M est bien symétrique. **On ne demande pas de vérifier que la matrice M est une matrice d'adjacence**.
   Par exemple
   - `verification(np.array([[0,1,0],[1,0,0],[0,0,0]]))` renvoie la valeur `True`.
   - `verification(np.array([[0,1,0],[1,0,0],[0,1,0]]))` renvoie la valeur `False`.
4. Démontrez par récurrence que pour $n \in \mathbb{N} \setminus \{0, 1\}$, $\displaystyle\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$.
5. Quelle est le nombre maximal et minimal d'arêtes d'un graphe à n sommets ?
6. On fixe n le nombre de sommets du graphe. Combien de graphes différents peut on construire ? On pourra raisonner sur les coefficients de la matrice d'adjacence.

7. Écrire une fonction `aretes(M)` qui, étant donné une matrice d'adjacence d'un graphe non orienté, renvoie le nombre d'arêtes du graphe.
   Par exemple
   - `aretes(np.array([[0,1,0],[1,0,0],[0,0,0]]))` renvoie 1.
   - `aretes(np.array([[0,1,0],[1,0,1],[0,1,0]]))` renvoie 2.

Dans toute la suite n est un entier naturel plus grand que 2 fixé, qui représente le nombre de sommet des graphes étudiés. Les graphes considérés seront des graphes non orientés sans boucle.

Graphe binomiale

Dans ce modèle on fixe un réel $p \in]0;\ 1[$. On regarde successivement tous les couples (i, j) de sommets tels que $i < j$ et on décide aléatoirement, avec une probabilité p, de relier les sommets i, j par une arête.
On note $G(n, p)$ le graphe alétoire obtenu avec le procédé précédent.

8. Recopier et compléter le programme suivant.

```
def graphe(n,p):
    '''retourne la matrice d'adjacence d'un graphe aléatoire à n sommets'''
    M=nulle(n)
    for i in ........:
        for j in ........:
            if .......... < p:
                M[i][j]=1
                ..........
    return M
```

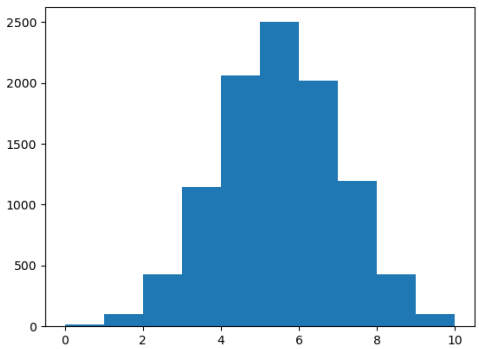
9. On note X la variable aléatoire du nombre d'arêtes d'un tel graphe. Quelle loi suit X ? Donner son espérance et sa variance.

10. On propose le programme suivant

```
n=5
p=0.5
N=10000
L=[]
for i in range(N):
    G=graphe(n,p)
    L.append(aretes(G))

plt.hist(L)
```

On obtient le graphique suivant



Expliquer ce que fait le programme et pourquoi le graphe obtenu est cohérent avec les questions précédentes.

11. Dans le programme précédent on remplace les deux premières lignes par

```
n=10
p=0.25
```

Donner l'allure du graphique qui sera alors obtenu.

12. On note N_0 la variable aléatoire donnant le nombre de sommets adjacents au sommet 0 dans le graphe $G(n, p)$. Donner la loi de N_0.

Sommets isolés

On dit qu'un sommet est isolé si il n'est relié à aucun autre sommet. Le but de cette partie est de calculer le nombre moyen de sommets isolé dans un graphe aléatoire.

Soit c un réel strictement positif. On note $p(n) = \dfrac{\ln n}{n} + \dfrac{c}{n}$, on s'intéresse au graphe aléatoire $G(n, p(n))$ (on utilise la définition de la partie précédente).

Pour $i \in [\![0;\ n-1]\!]$ on note Y_i la variable aléatoire qui vaut 1 si le sommet i est isolé 0 sinon. On note X_n l nombre de sommets isolés du graphe.

13. Écrire une fonction `isole(M)` qui pour un graphe donné par sa matrice M renvoie le nombre sommets isolés. Par exemple
    - `np.array(isoles([[0,0],[0,0]]))` renvoie 2.
    - `np.array(isoles([[0,0,1],[0,0,0],[1,0,0]]))` renvoie 1.
    - `np.array(isoles([[0,1,1],[1,0,0],[1,0,0]]))` renvoie 0.
14. Donner une relation entre X_n et $Y_0, \dots Y_{n-1}$.
15. Donner une expression de $P(Y_0 = 1)$, en déduire $E(Y_1)$.
16. En déduire que $E(X_n) = n\left(1 - \dfrac{c + \ln n}{n}\right)^{n-1}$
17. Montrer que $\lim\limits_{n\to+\infty} E(X_n) = \mathrm{e}^{-c}$. *Indications :* $\lim\limits_{x\to 0} \dfrac{\ln(1+x)}{x} = 1$. On a aussi pour a strictement positif et b réel $a^b = \mathrm{e}^{b\ln a}$.
18. Proposer un programme illustrant le résultat précédent.