Banque « Agro-Véto »
A TB - 0723

ALGORITHMIQUE ET INFORMATIQUE

Durée : 45 minutes

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

Ce sujet comporte 5 pages numérotées de 1 à 5.

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.

Ce sujet comporte trois indépendantes, mais qui peuvent faire appel au Préliminaire.

Préliminaire : Loi de Bernoulli

On rappelle qu'une variable aléatoire X qui suit une loi de Bernoulli de paramètre $p \in [0, 1]$ renvoie 1 avec probabilité p et 0 sinon.

On supposera que bibliothèque `random` a été importée.

```
from random import *
```

On rappelle que `random()` renvoie un nombre pseudo-aléatoire dans $[0, 1[$ avec une probabilité uniforme.

1. Écrire une fonction `ber(p)` qui simule une variable aléatoire X qui suit une loi de Bernoulli

```
def ber(p):
    ...
return(...)
```

On pourra utiliser librement dans les parties suivantes la fonction `ber(p)`, *même si on n'a pas répondu à cette question.*

Partie 1 : Trafic routier dans un tunnel

Soit n un entier naturel supérieur ou égal à 2.

Considérons n cases disposées en ligne représentant un tunnel à sens unique.

Chaque case représente un emplacement qui peut contenir un véhicule (représenté par un 1) ou pas de véhicule (représenté par 0).

Nous représenterons une telle configuration par une liste contenant des 0 ou des 1. Par exemple :

$$T = [0, 1, 1, 0, 1, 0, 1, 1]$$

2. Que vaut `len(T)` ?

3. Que renvoie `T[1]` ?

On supposera pour simplifier que les règles d'évolution dans ce tunnel sont les suivantes :

- Une voiture avance vers sa droite si la case à sa droite est libre, sinon la voiture reste immobile.
- Si une voiture se trouve complètement à droite (à la sortie du tunnel), elle sort du tunnel et disparaît de la liste.

Par exemple, la configuration après

$$T = [0, 1, 1, 0, 1, 0, 1, 1]$$

est

$$L = [0, 1, 0, 1, 0, 1, 1, 0]$$

4. Donner sous forme d'une liste l'état du trafic après la configuration $[0, 1, 0, 1, 0, 1, 1, 0]$.

5. Recopier en complétant la fonction `tunnel(T)` en langage Python qui, à partir d'une configuration T (une liste), renvoie la configuration suivante.

```
def tunnel(T):
    L=len(T)*[0]
    for i in range(0,len(T)-1):
        if T[i]==1 and T[i+1]==0:
            ...
        if T[i]==1 and T[i+1]==1:
            ...
    return(L)
```

On ajoute une nouvelle règle. Dans le cas où la case d'indice 0 est vide (contient un 0) à l'étape k, un véhicule s'introduit dans la case 0 avec probabilité $p \in [0, 1]$ à l'étape $k + 1$.

6. Recopier en complétant la fonction `tunnel2(T,p)` en langage Python qui, à partir d'une configuration T (une liste) et d'une probabilité $p \in [0, 1]$, renvoie la configuration suivante en tenant compte de la nouvelle règle précédente.

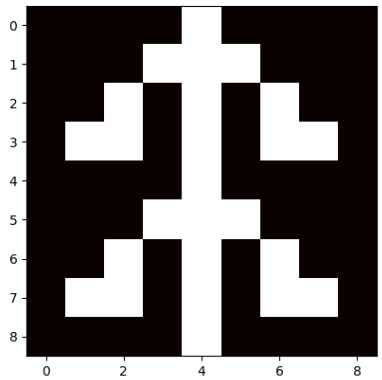
```
def tunnel2(T,p):
    L=len(T)*[0]
    for i in range(0,len(T)-1):
        if T[i]==1 and T[i+1]==0:
            ...
        if T[i]==1 and T[i+1]==1:
            ...
    if T[0]==0:
        ...
    return(L)
```

7. Soit $n \in \mathbb{N}^*$. Recopier en complétant la fonction `evol(T,n,p)` ci-dessous qui, à partir d'une configuration initiale T et d'une probabilité $p \in [0,1]$, affiche l'évolution du trafic dans le tunnel après n étapes :

```
def evol2(T,n,p):
    L=T
    print(L)
    for i in range(n):
        L= ...
        print(L)
```

Partie 2 : Motif sur un coquillage

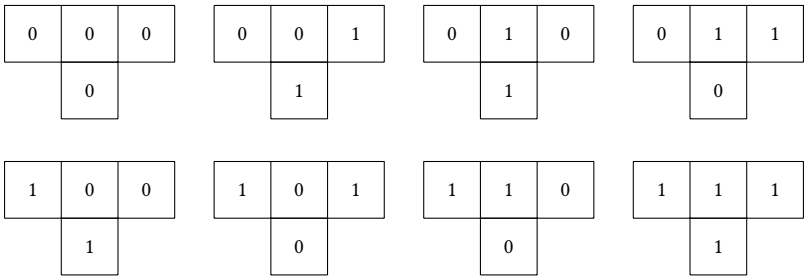
Nous nous intéressons à la formation de motifs sur un coquillage. On modélisera la surface du coquillage par une grille carrée à n lignes et n colonnes dont chaque case peut contenir du blanc (représenté par un 1) ou du noir (représenté par un 0). Par exemple la grille suivante :



sera représentée par la liste de listes :

```
[[0, 0, 0, 0, 1, 0, 0, 0, 0],
 [0, 0, 0, 1, 1, 1, 0, 0, 0],
 [0, 0, 1, 0, 1, 0, 1, 0, 0],
 [0, 1, 1, 0, 1, 0, 1, 1, 0],
 [0, 0, 0, 0, 1, 0, 0, 0, 0],
 [0, 0, 0, 1, 1, 1, 0, 0, 0],
 [0, 0, 1, 0, 1, 0, 1, 0, 0],
 [0, 1, 1, 0, 1, 0, 1, 1, 0],
 [0, 0, 0, 0, 1, 0, 0, 0, 0]]
```

Soit $k \in \mathbb{N}$. Les règles d'évolution s'effectuent de haut en bas : à partir de la ligne k, on obtient la ligne $k+1$. Le contenu d'une case à la ligne $k+1$ dépend du contenu des trois cases voisines à la ligne k.



Par défaut les cases à la ligne $k+1$ qui ne peuvent subir d'évolution (faute de voisin) contiennent des 0. Donnons un exemple d'évolution :

0	0	0	0	1	1	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0

8. À partir des règles précédentes, compléter à partir de la ligne k ci-dessous la ligne $k+1$:

0	0	1	0	1	0	0	1	1	1	0

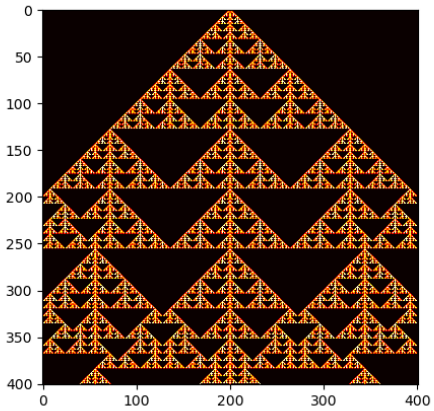
9. Recopier en complétant la fonction `coq(T)` qui, à une liste T représentant une ligne de la grille, renvoie une liste représentant la ligne suivante.

```
def coq(T):
    L=len(T)*[0]
    for i in range(1,len(T)-1):
        ...
    return(L)
```

10. Recopier en complétant la fonction `evol4(T,n)` qui, à partir d'une liste initiale T, renvoie la liste de listes A contenant l'évolution successive des lignes sur n étapes.

```
def evol4(T,n):
    A=[T]
    L=T
    for i in range(n):
        L= ...
        ...
    return(A)
```

En partant de la liste T contenant 401 zéros sauf un 1 au milieu, on obtient la figure suivante :



Les motifs qui apparaissent sont très similaires au coquillage *Conus textile* :

11. On souhaite connaître la proportion de 1 contenus dans la liste de listes M représentant la grille. Parmi les fonctions suivantes, laquelle permet d'effectuer ce calcul ?

```
def prop1(M):
    S=0
    for i in range(len(M)):
    for j in range(len(M[i])):
        S+=M[i][j]
    return(S/len(M)**2)
```

```
def prop2(M):
    S=0
    for i in range(len(M)):
        for j in range(len(M[i])):
            S+=M[i][j]
        return(S/len(M)**2)
```

```
def prop3(M):
    S=0
    for i in range(len(M)):
        for j in range(len(M[i])):
            S+=M[i][j]
    return(S/len(M)**2)
```

```
def prop4(M):
    S=0
    for i in range(len(M)):
    for j in range(len(M[i])):
        S+=M[i][j]
        return(S/len(M)**2)
```

```
def prop5(M):
    S=0
    for i in range(len(M)):
        for j in range(len(M[i])):
            S+=M[i][j]
            return(S/len(M)**2)
```

```
def prop6(M):
    S=0
    for i in range(len(M)):
    for j in range(len(M[i])):
            S+=M[i][j]
    return(S/len(M)**2)
```

Partie 3 : Épidémie dans une étable

Soit n un entier naturel supérieur ou égal à 3. On s'intéresse à la transmission de la tuberculose dans une étable au sein d'une population de bovins.

L'étable est modélisée par une grille carrée comportant n^2 cases. Dans chaque case se trouve un bovin qui peut être infecté (état 1) par la tuberculose ou sain (état 0).

On représentera informatiquement cette grille par une liste de listes comportant des 0 ou des 1 comme dans l'exemple ci-dessous :

0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	1	0	1

```
[[0,0,1,1,1],
 [0,0,1,1,1],
 [0,0,1,1,1],
 [0,0,1,1,1],
 [0,0,1,0,1]]
```

Le voisinage d'un bovin au sein de l'étable dépend de sa position dans la grille :

V	V	V
V	B	V
V	V	V

Dans cet exemple, le bovin B possède 8 voisins V.

V	V
B	V

Ici, le bovin B est dans un coin de la grille, il possède 3 voisins V.

V	V
B	V
V	V

Le bovin B se trouve sur un coté de la grille sans être dans un coin et possède 5 voisins V.

Soit $p \in \left[0, \frac{1}{10}\right]$.

Les règles de transmission de la maladie sont les suivantes :

- Un bovin à l'état 1 reste à l'état 1.
- Un bovin à l'état 0 passe à l'état 1 avec la probabilité kp où k est le nombre de bovins infectés dans son voisinage.

12. Proposer un algorithme permettant de coder une fonction `epi(T,p)` qui, à partir d'une grille T et d'une probabilité $p \in [0, \frac{1}{10}]$, renvoie une grille A représentant l'évolution de l'étable à l'étape suivante.

On pourra utiliser la fonction `ber(p)`.

Le codage en Python de `epi(T,p)` n'est pas demandé, mais peut éventuellement constituer une manière de répondre à la question.

FIN DU SUJET