TP n° 11 : Chaînes de Markov

1 Une première chaîne de Markov

On va traiter des chaînes de Markov tout d'abord sur l'exemple suivant.
On considère un robot que l'on lâche sur le coin (tiré au sort) d'un carré dont les sommets (en tournant dans le sens des aiguilles d'une montre) sont notés 0, 1, 2 et 3. Quand il est en 0, 1 ou 3, le robot choisit au hasard un des deux sommets reliés au sommet sur lequel il est. Quand il est en 2 il "saute" en 0.
On note par la suite S_n variable aléatoire donnant la position du robot après n mouvements. On note enfin X_n la matrice :

$$X_n = \begin{pmatrix} P(S_n = 0) & P(S_n = 1) & P(S_n = 2) & P(S_n = 3) \end{pmatrix} .$$

1.1 Modélisation du problème

1. Dessiner le graphe probabiliste associé à ce processus.
2. Ecrire le vecteur X_0.
3. Ecrire la matrice carrée M telle que $X_{n+1} = X_n M$.
4. Déterminer l'état stable de la chaîne (éventuellement à l'aide de Python).

1.2 Étude par la simulation du processus aléatoire

Une façon d'étudier cette chaîne de Markov est de simuler la réalisation d'une trajectoire de ce processus jusqu'à un temps n fixé. On répète ensuite cette expérience un grand nombre de fois afin d'estimer la loi de S_n.
Compléter la fonction suivante afin qu'elle renvoie la position simulée au temps n d'une trajectoire de cette chaîne de Markov.

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

def simulMarkov(n):
    S = rd.randint(0,4)  #position initiale
    for i in range(1,n+1):
        if S==0 :
            a = rd.random()
            if a < 0.5 :
                S = ...
            else :
                S = ...
        elif S == 1 :
            ...
        ... (plusieurs lignes)
    return S
```

Tester votre fonction.
On veut ensuite estimer la loi de S_n. Pour cela, on répète un grand nombre (N) de fois cette expérience. On détermine la fréquence (empirique) avec laquelle S_n prend chacune des valeurs 0, 1, 2, 3.
Compléter le programme suivant :

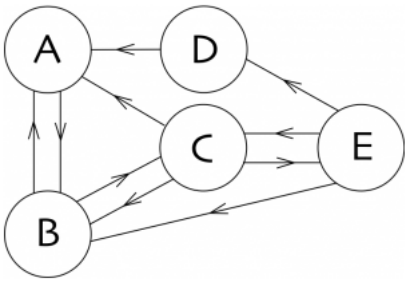
```
n=100
N=1000
occurences = np.zeros(4)
for k in range(N):
    S = simulMarkov(n)
    occurences[S] = ...

freq = ...
print(freq)
```

Ces résultats sont-ils cohérents avec ceux de la première partie ?

2 Une application : le PageRank de Google

On se propose ici d'étudier l'algorithme utilisé dans les moteurs de recherche sur internet pour ordonner les pages entre elles. Prenons un exemple simple. On considère cinq pages internet A, B, C, D, E avec les liens présentés sur le schéma suivant :



On associe alors à ce schéma une matrice dans laquelle on affecte à chaque colonne et à chaque ligne l'une des pages, et on indique en position (i, j) la probabilité d'aller en i si on est en j et que l'on choisit au hasard (avec la loi uniforme) la page suivante parmi celles accessibles à partir de la page j.

1. Déterminer la matrice de transition M associée au graphe précédent.
2. L'idée de l'algorithme de PageRank est d'ordonner les pages suivant la probabilité de cette page dans l'état limite de la chaîne (c'est à dire l'état stable).
   Déterminer cet état et le classement des pages sur l'exemple ci-dessus.
   Plus précisément, on déterminera l'état stable non pas par un calcul matriciel mais en simulant un grand nombre de trajectoires.