Corrigé

Une introduction aux chaînes de Markov

On importe les modules suivants :

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

Exercice 1

a) Le graphe est à l'évidence irréductible et apériodique.

b) On rédige la fonction suivante :

```
def simulation(n, x=1):
    t = [None, 0, 0, 0]
    t[x] = 1
    for _ in range(n):
        p = rd.rand()
        if x == 1:
            if p < 1 / 12:
                x = 2
            elif p < 1 / 6:
                x = 3
        elif x == 2:
            if p < 1 / 4:
                x = 3
            elif p < 1 / 2:
                x = 1
        else:
            if p < 1 / 4:
                x = 1
        t[x] += 1
    return t[1] / n, t[2] / n, t[3] / n
```

Quelques essais semblent montrer la convergence du triplet lorsque n tend vers $+\infty$:

```
In [1]: simulation(10000)
Out[1]: (0.588, 0.1041, 0.308)

In [2]: simulation(100000)
Out[2]: (0.6074, 0.0994, 0.29321)

In [3]: simulation(1000000)
Out[3]: (0.599514, 0.10025, 0.300237)
```

Il semble que Zébulon soit en moyenne en bonne santé 60% du temps, enrhumé 10% du temps, et grippé 30% du temps.

c) Ceci ne semble pas dépendre de l'état de santé initial de ce dernier :

```
In [4]: simulation(1000000, x=3)
Out[4]: (0.600371, 0.100519, 0.299111)
```

d) La matrice de transition vaut : $P = \begin{pmatrix} 5/6 & 1/12 & 1/12 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 0 & 3/4 \end{pmatrix}$. On utilise Python pour en trouver les valeurs propres :

```
In [5]: P = np.array([[5/6, 1/12, 1/12], [1/4, 1/2, 1/4], [1/4, 0, 3/4]], dtype=float)

In [6]: alg.eig(P.T)
Out[6]: (array([1.        , 0.5       , 0.58333333]),
         array([[ 8.84651737e-01,  1.22628141e-15, -4.08248290e-01],
                [ 1.47441956e-01, -7.07106781e-01, -4.08248290e-01],
                [ 4.42325868e-01,  7.07106781e-01,  8.16496581e-01]]))
```

La matrice P^T possède trois valeurs propres distinctes donc est diagonalisable ; la première colonne de la matrice de passage fournie par Python donne un vecteur propre pour la valeur propre 1 ; cependant il ne s'agit pas d'une loi de probabilité puisque la somme ne vaut pas 1 ; il faut donc normaliser ce vecteur :

```
In [7]: Q = alg.eig(P.T)[1]  # Q est la matrice de passage qui diagonalise P.T

In [8]: v = Q.T[0]           # v vérifie donc v P = v

In [9]: (v[0] / sum(v), v[1] / sum(v), v[2] / sum(v))
Out[9]: (0.6000000000000001, 0.09999999999999999, 0.29999999999999993)
```

On constate que $v = (0.6, 0.1, 0.3)$ est une loi invariante.

e) Calculons maintenant P^{100} :

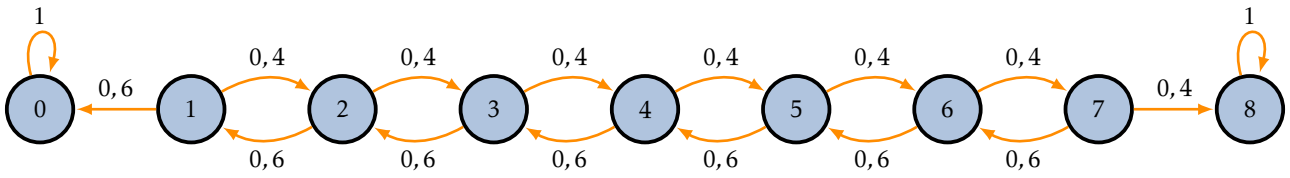
```
In [10]: alg.matrix_power(P, 100)
Out[10]: array([[0.6, 0.1, 0.3],
                [0.6, 0.1, 0.3],
                [0.6, 0.1, 0.3]])
```

Notons λ_1 et λ_2 les deux autres valeurs propres de P^T, w_1 et w_2 deux vecteurs propres associés. Si Q est la matrice de passage de la base canonique vers la base (v^T, w_1, w_2) on a $P^T = Q\begin{pmatrix}1&0&0\\0&\lambda_1&0\\0&0&\lambda_2\end{pmatrix}Q^{-1}$ donc $(P^T)^n = Q^{-1}\begin{pmatrix}1&0&0\\0&\lambda_1^n&0\\0&0&\lambda_2^n\end{pmatrix}Q$ et puisque $|\lambda_1| < 1$ et $|\lambda_2| < 1$, $\lim\limits_{n\to+\infty}(P^T)^n = Q^{-1}\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}Q = \begin{pmatrix}v^T & v^T & v^T\end{pmatrix}$. On en déduit $\lim\limits_{n\to+\infty} P^n = \begin{pmatrix}v\\v\\v\end{pmatrix}$.

Ainsi, quelle que soit la loi initiale $v_0 = (a, b, c)$ de X_0, la loi de X_n est $v_n = v_0 P^n$ donc $\lim v_n = (a+b+c)v = v$: il y a convergence en loi vers la loi invariante.

Exercice 2

a) La somme des poids des arêtes qui partent d'un nœud du graphe doit être égale à 1, donc des nœuds 0 et 8 doivent partir une unique transition de poids 1 qui boucle sur elle-même.



Cette chaîne de Markov n'est pas irréductible (les autres états ne sont pas accessibles à partir d'un état absorbant) et n'est pas apériodique (mis à part les états absorbants, les autres états sont de période 2).

b) On définit la matrice P à l'aide du script suivant :

```
P = np.zeros((9, 9), dtype=float)

P[0, 0] = 1
P[8, 8] = 1

for k in range(1, 8):
    P[k, k-1] = .6
    P[k, k+1] = .4
```

Si on note X_n la variable aléatoire égale à la fortune de Zébulon après n parties, la loi de X_0 est $v_0 = (0,0,0,1,0,0,0,0,0)$ et celle de X_n égale à $v_n = v_0 P^n$, ce qui donne :

```
In [1]: v0 = np.array([0, 0, 0, 1, 0, 0, 0, 0, 0])

In [2]: v0 @ alg.matrix_power(P, 3)
Out[2]: array([0.216, 0., 0.432, 0., 0.288, 0., 0.064, 0., 0.])

In [3]: v0 @ alg.matrix_power(P, 10)
Out[3]: array([0.56710195, 0.10749542, 0., 0.16403005,
               0., 0.1008599, 0., 0.02618819, 0.03432448])

In [4]: v0 @ alg.matrix_power(P, 100)
Out[4]: array([9.03526902e-01, 1.25490981e-05, 0., 2.01974685e-05,
               0., 1.34649790e-05, 0., 3.71825129e-06, 9.64231678e-02])
```

c) Le calcul précédent laisse penser que la probabilité que Zébulon sorte de prison est d'approximativement 9,6%. Pour vérifier cette expérience, nous allons simuler un grand nombre de parties, en dénombrant le pourcentage de celles-ci qui aboutissent à la libération de Zébulon.

```
def partie():
    s = 3
    while s > 0 and s < 8:
        if rd.rand() < 0.6:
            s -= 1
        else:
            s += 1
    if s == 0:
        return 0
    else:
        return 1

def simulation(n):
    s = 0
    for _ in range(n):
        s += partie()
    return s / n * 100
```

Quelques essais montrent qu'en effet cette estimation est la bonne :

```
In [5]: simulation(100000)
Out[5]: 9.56

In [6]: simulation(100000)
Out[6]: 9.713

In [7]: simulation(100000)
Out[7]: 9.608
```

d) Observons le résultat de la diagonalisation de P^T :

```
In [1]: spectre, Q = alg.eig(P.T)

In [2]: spectre
Out[2]: array([ 1., 1., -0.90521338, -0.69282032, -0.37495166,
                0.37495166, 0.90521338, 0.69282032, 0.])
```

1 est valeur propre double, et les autres valeurs propres sont nulles ou de module strictement inférieur à 1.

```
In [3]: Q.shape
Out[3]: (9, 9)
```

La matrice des vecteurs propres est bien de dimension 9 ; P^T est diagonalisable. On en déduit que

$$P^T = Q\begin{pmatrix}1 & & & & \\ & 1 & & & \\ & & \lambda_3 & & \\ & & & \ddots & \\ & & & & \lambda_9\end{pmatrix}Q^{-1}, \quad (P^n)^T = Q\begin{pmatrix}1 & & & & \\ & 1 & & & \\ & & \lambda_3^n & & \\ & & & \ddots & \\ & & & & \lambda_9^n\end{pmatrix}Q^{-1}, \quad \lim(P^n)^T = Q\begin{pmatrix}1 & & & & \\ & 1 & & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & 0\end{pmatrix}Q^{-1}$$

On en déduit que $\lim X_n = \lim v_0 P^n = v_0 (Q^{-1})^T D Q^T$ avec $D = \mathrm{diag}(1,1,0,\dots,0)$.

```
In [4]: v0 @ alg.inv(Q).T @ np.diag([1, 1, 0, 0, 0, 0, 0, 0, 0]) @ Q.T
Out[4]: array([0.9035686, 0., 0., 0., 0., 0., 0., 0., 0.0964314])
```

On retrouve bien le faut que Zébulon a environ 9,6% de chance de sortir de prison.

e) On modifie légèrement le code pour dénombrer cette fois le nombre de parties réalisées :

```
def nbPartie():
    s = 3
    n = 0
    while s > 0 and s < 8:
        if rd.rand() < 0.6:
            s -= 1
        else:
            s += 1
        n += 1
    return n

def simulation3(n):
    s = 0
    for _ in range(n):
        s += nbPartie()
    return s / n
```

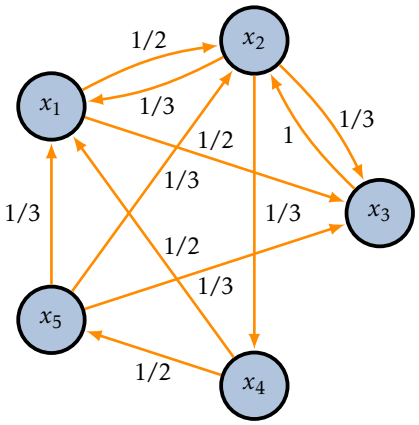
```
In [8]: simulation3(100000)
Out[8]: 11.17204

In [9]: simulation3(100000)
Out[9]: 11.13394
```

Zébulon va en moyenne réaliser un peu plus de 11 parties.

Exercice 3

a)



Le graphe est irréductible car il existe un chemin fermé reliant tous les sommets : $x_1 \to x_3 \to x_2 \to x_4 \to x_5 \to x_1$. Le sommet x_1 est apériodique car un existe deux chemins fermés $x_1 \to x_2 \to x_1$ et $x_1 \to x_3 \to x_2 \to x_1$ de longueurs respectives 2 et 3, et $\mathrm{pgcd}(2,3) = 1$. On en déduit que le graphe tout entier est apériodique.

b) La matrice de transition est égale à $P = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1/3 & 1/3 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \end{pmatrix}$.

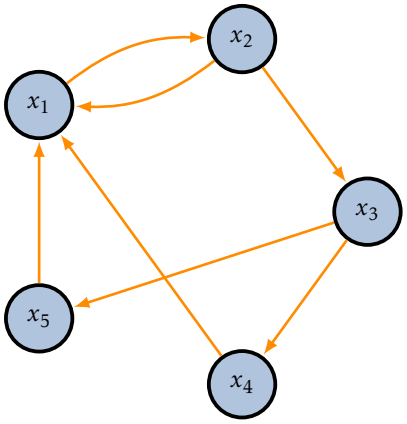
c) D'après le théorème 1.3 la chaîne de Markov possède une unique loi invariante, obtenue en cherchant les vecteurs propres associés à la valeur propre 1 pour la matrice P^T :

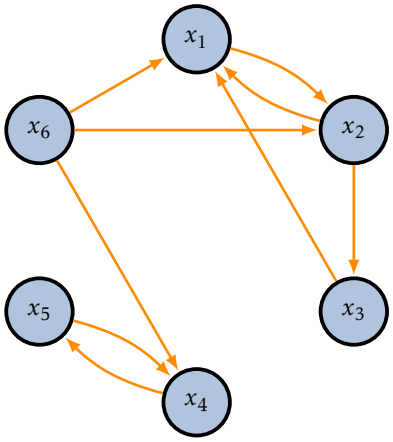
```
In [1]: v = alg.eig(P.T)[1][:, 0]

In [2]: [v[k] / sum(v) for k in range(5)]
Out[2]: [0.204081632653, 0.367346938775, 0.244897959183, 0.1224489795918, 0.0612244897959]
```

Ces cinq pages sont donc classées par ordre décroissant de score *PageRank* : x_2, x_3, x_1, x_4, x_5.

d) On obtient les représentations suivantes pour A_2 et A_3 :





Le graphe A_2 est irréductible mais pas apériodique (toutes les périodes sont paires). Il y a bien unicité de la loi invariante, mais il n'y a pas nécessairement convergence en loi de la suite (X_n).
Le graphe A_3 n'est pas irréductible car il est impossible de relier x_4 à x_1 ; il n'y a donc pas nécessairement unicité de la loi invariante.

Exercice 4

a) Soit k le nombre de liens vers une autre page dans un graphe de taille n ; on a $k \in [\![0, n-1]\!]$.
Si $k = 0$ (cas d'un dangling node) il faut attribuer la probabilité $\frac{1}{n}$ à chacune des pages ; si $k > 0$ il faut attribuer la probabilité $\frac{\alpha}{n} + \frac{1-\alpha}{k}$ aux pages vers lesquelles se dirige un des liens de la page courante, et $\frac{\alpha}{n}$ aux autres pages.

```
def markov(A, alpha):
    n = A.shape[0]
    P = np.zeros((n, n), dtype=float)
    for i in range(n):
        k = sum(A[i])
        if k == 0:
            for j in range(n):
                P[i, j] = 1 / n
        else:
            for j in range(n):
                P[i, j] = alpha / n + A[i, j] * (1 - alpha) / k
    return P
```

b) On en déduit :

```
def pagerank(A, alpha):
    P = markov(A, alpha)
    v = alg.eig(P.T)[1][:, 0]
    return [v[k] / sum(v) for k in range(A.shape[0])]
```

c) Le score de la matrice A_1 est maintenant le suivant :

```
In [3]: A1 = np.array([[0, 1, 1, 0, 0],
                       [1, 0, 1, 1, 0],
                       [0, 1, 0, 0, 0],
                       [1, 0, 0, 0, 1],
                       [1, 1, 1, 0, 0]], dtype=int)

In [4]: pagerank(A1, .15)
Out[4]: [0.2055306055807, 0.3440947647792, 0.2386963683603, 0.1274935166874, 0.0841847445921]
```

Il y a peu de modification par rapport au résultat déjà obtenu, sans téléportation.

```
In [5]: A2 = np.array([[0, 1, 0, 0, 0],
                       [1, 0, 1, 0, 0],
                       [0, 0, 0, 1, 1],
                       [1, 0, 0, 0, 0],
                       [1, 0, 0, 0, 0]], dtype=int)

In [6]: pagerank(A2, .15)
Out[6]: [0.3299481299481, 0.3104559104559, 0.1619437619437, 0.0988260988260, 0.0988260988260]
```

Pour la matrice A_2, les pages 4 et 5 obtiennent le même score, ce qui est normal compte tenu de leur symétrie

```
In [7]: A3 = np.array([[0, 1, 0, 0, 0, 0],
                       [1, 0, 1, 0, 0, 0],
                       [1, 0, 0, 0, 0, 0],
                       [0, 0, 0, 0, 1, 0],
                       [0, 0, 0, 1, 0, 0],
                       [1, 1, 0, 1, 0, 0]], dtype=int)

In [8]: pagerank(A3, .15)
Out[8]: [0.2368459895, 0.2334024244, 0.1241960304, 0.1921921921, 0.1883633633, 0.0250000000]
```

La page 6 possède un très faible score, qui s'explique par le fait qu'aucune page ne pointe vers celle-ci.