Python - TP 9
Graphes : centralité spectrale

On commence par relire la fin du cours sur les graphes... Si M désigne la matrice d'adjacence d'un graphe $\mathcal{G}$, alors rechercher les centralités spectrales des sommets de $\mathcal{G}$ équivaut à :

- trouver une valeur propre λ strictement positive de M,
- trouver un vecteur propre associé...

Pour cela, on met en œuvre des méthodes numériques d'approximation de valeurs propres et vecteurs propres dont la plus simple (mais la moins efficace) est la méthode itérative qui suit :

- on choisit un vecteur V (à peu près comme on veut... on peut prendre le premier vecteur de la base canonique de $\mathcal{M}_{n,1}(\mathbb{R})$ par exemple),
- on calcule les termes successifs de la suite (X_n) définie par :

$$\begin{cases} X_0 = V \\ \forall n \in \mathbb{N},\ X_{n+1} = MX_n \end{cases}$$

★Subtile...★
En pratique, il faut en fait *normer* le vecteur X_n à chaque itération pour éviter l'*overflow*...

On (mais pas nous) peut démontrer que sous de bonnes conditions, dépendant du choix du vecteur initial et de la matrice M, cette méthode converge vers un vecteur propre de M associé à la plus grande valeur propre de M en valeur absolue.
En ce qui nous concerne, le fait que M soit la matrice d'adjacence d'un graphe nous assure que cette méthode converge vers un vecteur propre de la plus grande des valeurs propres de M (qui est, d'après le théorème de Perron-Frobenius, une valeur propre strictement positive, dont l'espace propre est de dimension 1 et engendré par un vecteur à coordonnées strictement positives). Bref, c'est parfait !

☞ Pour info...
L'algorithme PageRank de Google, qui trie les pages du Web selon leur pertinence/importance, est basé sur cette centralité spectrale... Voir ESSEC II 2008 voie E.

Les matrices en Python

On pourrait voir les matrices en Python comme une liste de listes, mais on se rend rapidement compte que cette notion est trop restrictive : comment faire alors du calcul matriciel ?
En Python (comme en mathématiques d'ailleurs), une matrice est un tableau. Nous avions déjà parlé de tableau dans le cours sur les représentations graphiques. Rappelons déjà qu'un tableau en Python est une liste de listes, avec un point de vue numérique ! Les calculs sont donc permis... Étoffons un peu par quelques commandes...

Petite remarque
Revoir le cours sur les représentations graphiques !

On commence par : `import numpy as np`

Tester les commandes suivantes et indiquer ce qu'elles permettent d'obtenir :

```
>>> X=np.zeros((5,1))
>>> M=np.zeros((2,3))
>>> np.shape(X)
>>> np.shape(M)
>>> np.transpose(X)
>>> J=np.ones((5,5))
>>> I=np.eye(10)
>>> N=np.array([[0,1,5],[0,0,0]])
>>> M==N
>>> (M==N).all()
>>> M[0,1]=1 ;M[0,2]=5
>>> M==N
>>> (M==N).all()
```

✘ Attention !
• On fait attention au double-parenthèsage !
• On se souvient de la numérotation des listes en Python...

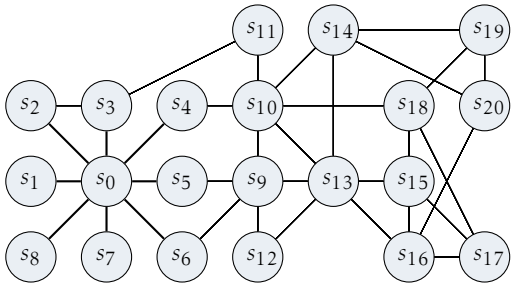
Également : le produit matriciel MN (quand il est possible) est donné par `np.dot(M,N)`.

Petite remarque
Pour la puissance M^k (quand M est carrée) :
on commence par : `import numpy.linalg as al`
puis : `al.matrix_power(M,k)`

Le module `numpy.linalg` fournit aussi les commandes `al.rank()` et `al.inv()`...

Résolution du problème !

Objectif : déterminer le sommet le plus influent (en terme d'indicateur spectral) du graphe suivant :



1. Récupérer le fichier `Python` : centralitespectrale.py, contenant une liste `L` qui ressemble à la liste des listes d'adjacence du graphe ci-dessus, mais dans laquelle n'ont été à chaque fois saisis que les sommets dont la numérotation est supérieure...
2. En utilisant la liste `L`, créer la matrice d'adjacence M du graphe ci-dessus.
3. Vérifier que M est symétrique.
4. Créer une fonction `norme` qui prend en argument d'entrée un vecteur et renvoie en sortie le vecteur normé correspondant.
5. Définir `X` comme le premier vecteur de la base canonique de $\mathcal{M}_{n,1}(\mathbb{R})$. Est-il normé ?
6. Calculer les 100 premiers termes de la suite $(X_n)_{n\in\mathbb{N}}$ définie par :

$$X_0 = X \ ; \ \forall n \in \mathbb{N},\ X_{n+1} = MX_n$$

7. En déduire le sommet le plus influent selon l'indicateur spectral.

☞ POUR INFO...

- la norme du vecteur $\vec{v} = (v_1, v_2, ..., v_n)$ est le nombre définie par : $\|\vec{v}\| = \sqrt{\sum_{k=1}^{n} v_k^2}$.
- un vecteur est *normé* s'il est de norme égale à 1.
- pour normer un vecteur (non nul), il suffit de le multiplier par l'inverse de sa norme.

```
import numpy as np

L=[[] for k in range(21)]
L[0]=[1,2,3,4,5,6,7,8]
L[2]=[3]
L[3]=[11]
L[4]=[10]
L[5]=[9]
L[6]=[9]
L[9]=[10,12,13]
L[10]=[11,13,14,18]
L[12]=[13]
L[13]=[14,15,16]
L[14]=[19,20]
L[15]=[16,17,18]
L[16]=[17,20]
L[17]=[18]
L[18]=[19]
L[19]=[20]
L[20]=[]

M=np.zeros((21,21))
for i in range(21):
    for j in L[i]:
        M[i,j]=1 #M est symétrique : remplissage simultané des coeff (i,j) et (j,i)
        M[j,i]=1

def norme(v):
    return np.sqrt(sum(v**2)) #v**2 renvoie la matrice des coeff élevés au carré

X=np.zeros((21,1))
X[0]=1
for k in range(100):
    X=np.dot(M,X)
    X=1/norme(X)*X #ne pas oublier de normaliser comme demandé

print(X)
m=max(X)
influenceur=list(X).index(m) #X n'est pas une liste... X.index() ne fonctionne pas. On convertit X en liste
print("l'influenceur le plus important est le sommet",influenceur)
```