

Code de partage avec Cappytale : 3286-7027587

Rappels pour manipuler des matrices

`np.array([[...],[...],[...]])` définit une matrice

Exemple : `A=np.array([[1, 0, 0, 1], [1, 0, 0, 1], [1, 0, 0, 1], [1, 1, 1, 1]])` définit

la matrice

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

`A[i, :]` : renvoie la ligne i de la matrice A

`A[:, j]` : renvoie la colonne j de la matrice A

`np.zeros((n, p))` : renvoie la matrice nulle de $\mathcal{M}_{n,p}(\mathbf{R})$

`np.ones((n, p))` : renvoie la matrice de $\mathcal{M}_{n,p}(\mathbf{R})$ ne contenant que des 1

`np.eye(n)` : renvoie la matrice identité de taille n

`np.dot(A, B)` effectue le produit des deux matrices A et B (quand il est défini)

`np.transpose(A)` : renvoie la transposée de A

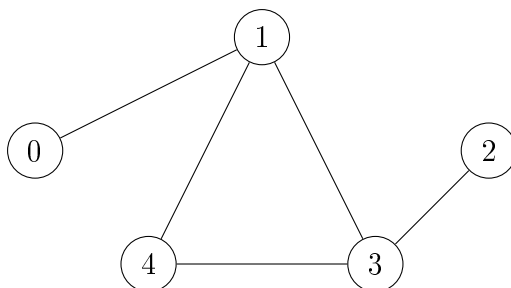
`al.inv(A)` : renvoie l'inverse de A (si elle existe).

`al.matrix power(A, k)` : renvoie A^k

`np.shape(A)` : donne le nombre de lignes et de colonnes de A sous la forme d'un couple de nombres.

Exercice 1

On considère le graphe G suivant :



1. écrire la matrice d'adjacence A du graphe G

Définition : on appelle **liste d'adjacence d'un graphe** la liste contenant pour chaque sommet (dans l'ordre) la liste des sommets qui lui sont reliés. C'est donc une liste de listes.

2. écrire la liste d'adjacence L du graphe G
3. écrire une fonction `ListeAdj(A)` qui renvoie la liste d'adjacence d'un graphe simple à partir de sa matrice d'adjacence A donnée en entrée.
 - on définira une variable n égale au nombre de sommets du graphe à l'aide de la commande `np.shape(A)`
 - on initialisera une liste L à la liste vide
 - pour un indice i allant de 0 à $n - 1$:
 - on initialisera la liste V des voisins de i à la liste vide

- pour un indice j allant de 0 à $n - 1$: on augmentera la liste V de la valeur j si j est voisin de i
- on augmente la liste L de la liste V
- renvoyer la liste L ainsi construite

Vérifier le résultat avec le graphe G de la question 1

```
def ListeAdj(A):
    ...

    return
```

4. écrire une fonction **MatAdj(L)** qui renvoie la matrice d'adjacence d'un graphe simple à partir de sa liste d'adjacence **L** donnée en entrée.

- on pensera à définir dans la fonction une variable **n** égale au nombre de sommets de G
- on initialisera la matrice A à la matrice carrée nulle d'ordre n
- pour tout indice i allant de 0 à $n - 1$: pour tout voisin v de i : modifier la valeur de **A[i, v]**

Vérifier le résultat avec le graphe G précédent.

```
def MatAdj(L):
    ...

    return
```

5. écrire une fonction **EstConnexe(A)** qui renvoie **True** si le graphe (quelconque) de matrice d'adjacence **A** est connexe, et **False** sinon.

```
def EstConnexe(A):
    ...

    return
```

Vérifier le résultat avec le graphe G précédent.