TP 10 - Graphes - Partie 1

On importera `numpy` *en début de script Python. Si nécessaire, on utilisera dans la console l'aide intégrée à Python à l'aide du mot clé* `help`.

I) Description d'un graphe via sa matrice d'adjacence

Exercice 1. Écrire une fonction Python `degre` qui prend en argument la matrice d'adjacence d'un graphe et qui renvoie un array de type `numpy` contenant le degré de chaque sommet. On pourra utiliser la fonction `np.sum`.

Exercice 2. Un classique.
Écrire une fonction Python `puiss_mat` qui prend en argument une matrice carrée A et un nombre entier positif k et qui renvoie A^k.

Exercice 3. Écrire une fonction `somme_puiss_mat` qui prend en argument une matrice carrée A et qui renvoie $I_n + A + ... + A^{n-1}$, où n est la taille de A.

Exercice 4. Écrire une fonction `test_connexe` qui prend en argument une matrice d'adjacence d'un graphe et qui renvoie `True` si ce graphe est connexe, `False` sinon.

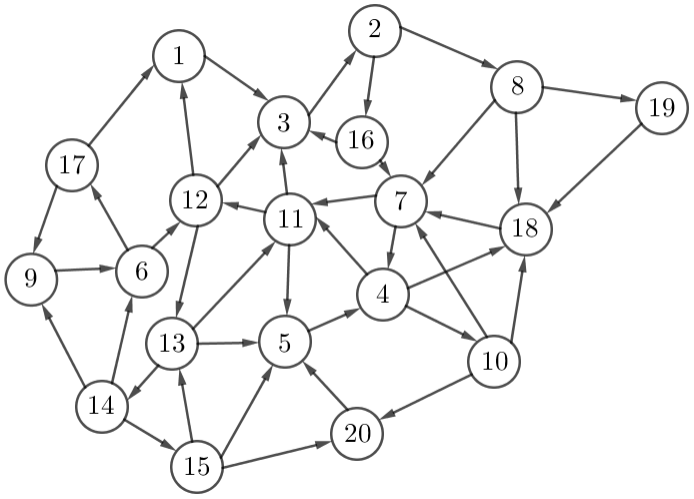
Exercice 5. On considère un graphe non orienté dont la matrice d'adjacence M est :

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

1. Construire sur une feuille une représentation d'un graphe ayant cette matrice d'adjacence, en numérotant les sommets avec les nombres de 1 à 6.
2. Ce graphe est-il connexe ?
3. A l'aide de Python, calculer $I_6 + M + M^2 + M^3 + M^4 + M^5$ et vérifier le résultat précédent.
4. Comment "voir" plus simplement le résultat obtenu dans la question 2 en observant la matrice d'adjacence ?
5. Que dire de la connexité d'un graphe non orienté dont la matrice d'adjacence est la suivante ? On commencera par utiliser Python pour trouver le résultat, et on vérifiera en traçant le graphe.

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

Exercice 6. A l'aide de Python, dire si le graphe orienté suivant est connexe ou non :



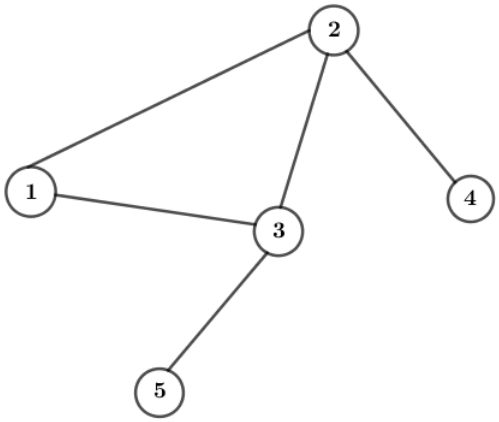
Exercice 7. Ecrire une fonction `test_eulerien` qui prend en argument la matrice d'adjacence d'un graphe G et qui renvoie `True` si G est eulérien, `False` sinon. *On pensera à commencer par vérifier si G est connexe.*

Exercice 8. Ecrire une fonction `test_semi_eulerien` qui prend en argument la matrice d'adjacence d'un graphe G et qui renvoie `True` si G est semi-eulérien, `False` sinon. *On pensera à commencer par vérifier si G est connexe.*

II) Description d'un graphe via les arêtes

Pour décrire un graphe, une autre possibilité est de donner la liste des arêtes, en supposant que les sommets sont numérotés (disons à partir de 1). Par exemple, on peut lister les arêtes dans une matrice ayant autant de lignes que d'arêtes, et ayant deux colonnes : chaque ligne de cette matrice sera du type `[x,y]`, où x et y sont deux entiers positifs non nuls, la présence d'une telle ligne indiquant qu'il y a une arête entre x et y (ou de x vers y si le graphe est orienté). Par convention, une arête `[x,y]` sera placée avant une arête `[z,t]` si $x < z$ ou si : $x = z$ et $y < t$.

Ci-après un exemple dans le cas d'un graphe (non orienté) :




```
array([[1, 2],
       [1, 3],
       [2, 1]
       [2, 3],
       [2, 4],
       [3, 1],
       [3, 2],
       [3, 5],
       [4, 2],
       [5, 3]])
```

Exercice 9. Écrire une fonction `complet` qui prend en argument un entier n et qui renvoie la matrice des arêtes du graphe complet.

Exercice 10. Écrire une fonction `enleve_arete` qui prend en argument une matrices des arêtes `A` et un array de la forme `[u,v]`, et qui renvoie :

- la matrice A dans laquelle l'arête `[u,v]` a été enlevée si elle y figure ;
- A sinon.

Exercice 11. Écrire une fonction `enleve_arete_tout` qui prend en argument une matrices avec les arêtes `A` et un nombre entier u, et qui renvoie :

- la matrice A dans laquelle toutes les arêtes partant de u ont été enlevées, si u est effectivement un sommet du graphe et si u n'est pas un sommet sans arêtes.
- A sinon.

Exercice 12. Écrire une fonction `ajout_arete` qui prend en argument une matrices avec les arêtes `A` (de la forme présentée précédemment) et un array de la forme `[u,v]`, et qui renvoie :

- la matrice A dans laquelle l'arête `[u,v]` a été rajoutée si elle n'y figure pas déjà ;
- A sinon.

Exercice 13. Écrire une fonction `ajout_arete_tout` qui prend en argument une matrices avec les arêtes `A` et un nombre entier u, et qui renvoie :

- la matrice A dans laquelle toutes les arêtes manquantes partant de u ont été ajoutées, si u est effectivement un sommet du graphe qui n'est pas déjà relié à tous les autres sommets ;
- A sinon.

Exercice 14. Écrire une fonction `aretes_vers_matrice` qui prend en argument une matrice des arêtes et qui renvoie la matrice d'adjacence associée.

Exercice 15. Écrire une fonction `matrice_vers_aretes` qui prend en argument une matrice d'adjacence et qui renvoie la matrice des arêtes associée.

Exercice 16. Propose une amélioration de la structure de données proposée dans cette partie (par exemple, lorsque le graphe est non orienté, une arête apparaît deux fois, ce qui est coûteux en terme de place). On pourra envisager de lister, pour chaque sommet, les sommets auxquels il est relié.