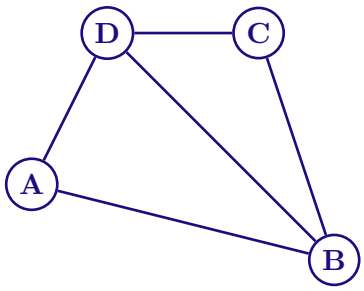
Exercice : Les graphes en Python

Exercice 1

On donne ci-dessous un graphe G :



1) Donner le dictionnaire d'adjacence du graphe G ci-dessus.

2) Donner la matrice d'adjacence du graphe G.

3) On oriente le graphe en respectant l'ordre alphabétique. Par exemple, les sommets A et B sont reliés par un arc orienté de A vers B.

a) Redessiner le graphe orienté.

b) Redonner le dictionnaire d'adjacence et la matrice d'adjacence du graphe orienté.

Exercice 2

On donne la matrice d'adjacence d'un graphe G non orienté :

```
G={
     'A':{'B','C'},\
     'B':{'A','D','E'},\
     'C':{'A','E','F'},\
     'D':{'B','F'},\
     'E':{'B','C','F'},\
     'F':{'C','D','E'}
}
```

1) Représenter le graphe G.

2) Donner la matrice d'adjacence du graphe G.

Exercice 3

On donne la matrice d'adjacence d'un graphe G non orienté :

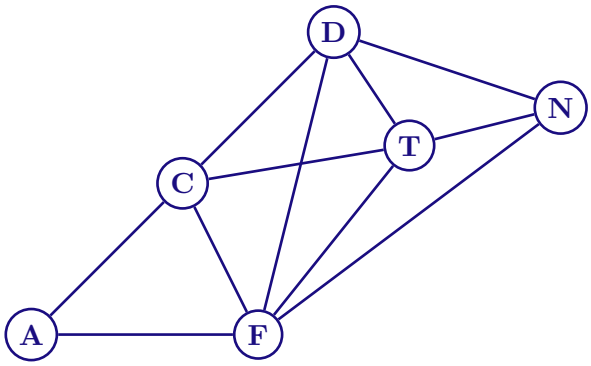
$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

1) Représenter le graphe G.

2) Donner le dictionnaire d'adjacence du graphe G.

Exercice 4

Un groupe d'amis organise une randonnée dans les Alpes. On a représenté par le graphe ci-dessous les sommets B, C, D, F, T et N par lesquels ils peuvent choisir de passer. Une arête (arc) entre deux sommets coïncide avec l'existence d'un chemin de randonnée entre les deux sommets.



1) On appelle degré d'un sommet le nombre de sommets qui lui sont adjacents. Dresser un tableau avec les degrés de chaque sommet.

2) On dit qu'un graphe est connexe lorsque 2 sommets quelconques sont reliés par un chemin. Le graphe G est-il connexe ?

3) **Euler** a montré qu'il existe un chemin reliant tous les sommets et passant une et une seule fois par toutes les arêtes du graphe lorsque le graphe contient exactement 2 sommets de degré impair. Un tel chemin est appelé **chaine eulérienne**.

Est-il envisageable de trouver une chaine eulérienne de randonnée passant par tous les chemins une et une seule fois ? Donner un tel chemin s'il existe.

Exercice 5

On donne la classe **Graphe** écrite en Python dont l'interface est la suivante :

Les attributs :

— **dict** qui est un objet **Graphe_dict**.
Un objet **Graphe_dict** a pour attribut **valeur** qui contient le dictionnaire d'adjacence du graphe ;

— **matrice** qui est un objet **Graphe_mat**.
Un objet **Graphe_mat** a pour attribut **valeur** qui contient la matrice d'adjacence du graphe ;

— **oriente** qui est booléen indiquant si le graphe est orienté ; par défaut le graphe est non orienté.

— **arcs** qui est une liste contenant les arcs du graphe.

Les méthodes :

— **ajouter_sommet** qui ajoute un sommet au graphe ;

— **ajouter_arc** qui ajoute un arc au graphe entre deux sommets (2 arcs si non orienté), le sommet est créé s'il n'existe pas ;

— **supprimer_sommet** qui supprime un sommet au graphe, les arcs associés sont supprimés ;

— **supprimer_arc** qui supprime un arc entre deux sommets ;

— **est_arc** qui renvoie un booléen d'existence d'arc entre deux sommets ;

— **sommets** qui renvoie la liste des sommets du graphe ;

— **voisins** qui renvoie la liste des sommets adjacents à un sommet ;

— **afficher** qui affiche une représentation du graphe.

Comment utiliser cette classe

```
1 from graphe import Graphe
```

Pour créer un objet G graphe non orienté, on doit appeler la classe Graphe. Si le graphe est orienté, on donne en argument la valeur **oriente=True**.

```
1 G=Graphe()
```

L'affichage des valeurs du dictionnaire d'adjacence et de la matrice d'adjacence se fait par les appels suivants :

```
1 print(G.dict.valeur) # dictionnaire d'adjacence
2 print(G.matrice.valeur) # matrice d'adjacence
```

L'ajout de sommets et d'arcs se fait par les méthodes de l'objet Graphe données ci-dessus :

```
1 G.ajouter_sommet('A')
2 G.ajouter_arc('A','B')
```

Pour représenter le graphe, on utilise la méthode afficher

```
1 d=G.afficher()
2 d.view()
```

Travail à chercher

1) Récupérer sur l'ENT, le fichier Python **graphe.py** contenant la classe Graphe et importer là dans un nouveau fichier python.

2) Créer l'objet graphe $G1$ représenté dans l'exercice 1 en ajoutant les sommets et les arcs qui le compose.

- **a)** Afficher le dictionnaire d'adjacence du graphe $G1$.
- **b)** Afficher la matrice d'adjacence du graphe $G1$.
- **c)** Afficher une vue du graphe $G1$.

3) La création d'un objet graphe se fait par l'ajout des arcs du graphe.

- **a)** Soit L=[('A','B'),('A','C'),('B','D'),('C','D')] la liste des arcs d'un graphe. En donner son dictionnaire et sa matrice d'adjacence.
- **b)** Écrire une fonction **arcs_en_graphe** qui a pour paramètre une liste d'arcs d'un graphe et renvoie un objet graphe associé à la liste des arcs passés en argument. Chaque arc sera un tuple composé des deux sommets reliés par l'arc.
- **c)** Vérifier votre fonction en créant le graphe $G2$ de l'exercice 2 à partir de la liste des arcs du graphe.

4) Le parcours d'un graphe consiste à prendre un chemin en suivant les arcs du graphe. Pour déterminer l'existence d'une chaine eulérienne, c'est à dire savoir si on peut parcourir un graphe en prenant chaque arc une et une seule fois, il est nécessaire de déterminer les degrés de chaque sommet.

- **a)** Écrire une fonction **degre_sommet** qui a pour paramètre un graphe et renvoie un dictionnaire dontles clefs sont chaque sommet du graphe et les valeurs le degré du sommet associé.
- **b)** Écrire une fonction **chaine_eulerienne** qui a pour paramètre un graphe et renvoie un booléen déterminant s'il existe une chaine eulérienne dans le graphe.
- **c)** Écrire un script python qui répond aux questions de l'exercice 4.

5) Écrire une fonction qui crée un graphe à partir de son dictionnaire d'adjacence. Vérifier avec les données de l'exercice 2 ou un autre graphe.

6) Écrire une fonction qui crée un graphe à partir de sa matrice d'adjacence. Vérifier avec les données de l'exercice 3 ou un autre graphe.