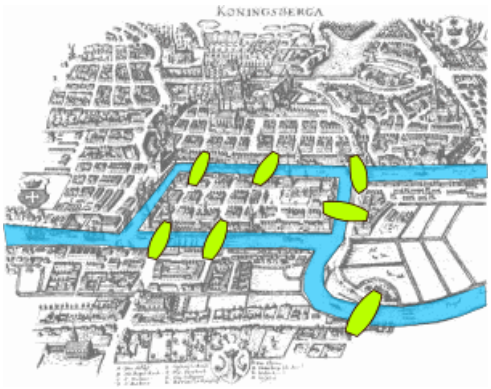
1 Introduction

Origine de la théorie des graphes.
En 1736, Euler s'est demandé si on ne pouvait aller partout dans la ville de Königsberg (Kalinigrad) en n'empruntant qu'une seule fois chaque pont.

De nombreuses applications dans le monde moderne

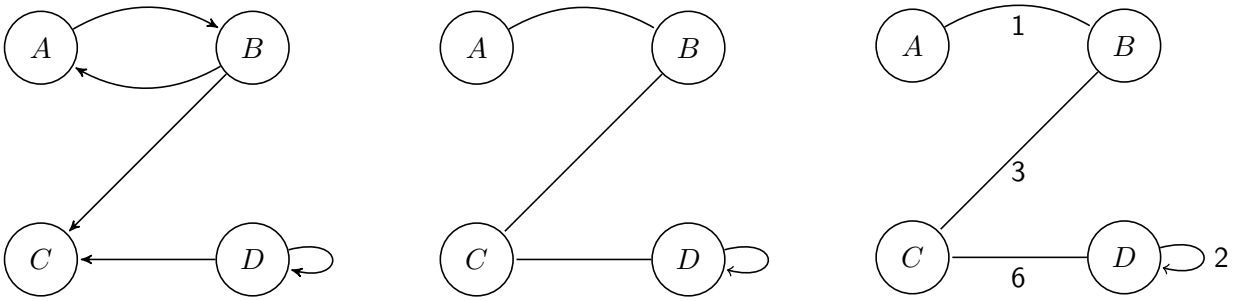
— circuits électriques
— trafic routier
— trafic aérien : sommets(aéroports), arêtes(vols existants)
— réseaux de communications
— structures de molécules en chimie
— génome en biologie
— relation entre individus en sciences sociales
— gestion des salles pour l'administration

Un promeneur peut-il visiter Königsberg en traversant chaque pont une fois uniquement ?



Définition : Un graphe orienté (resp. non orienté) (S, A) est la donnée :
— D'un ensemble S dont les éléments sont les sommets du graphe.
— D'un ensemble A dont les éléments , les arcs (resp. arête) du graphe sont des **couples** (resp. **paires** de S.

2 Différents type de graphes

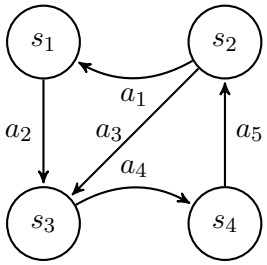


Vocabulaire (cas non orienté) :
Un **chemin** est dit **simple** s'il ne contient pas deux fois le même arc .
On appelle **circuit** un chemin dont le sommet de départ et celui d'arrivé sont identiques.
Un chemin (ou circuit) **eulérien** est un chemin qui passe une et une seule fois par toutes les arcs du graphe.

De préférence on utilise le vocabulaire suivant :

Orienté	**NON orienté**
arc	arête
chemin	chaîne
circuit	cycle

Exemple :



1. Trouver un circuit simple.
2. Trouver un chemin eulérien.

3 Degré

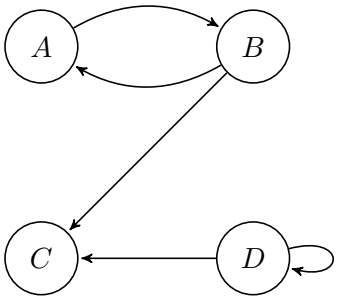
Cas orienté Soit $G = (A, S)$ un graphe orienté :

Soit (s_1, s_2) un arc de G
On dit que s_2 est un **successeur** de s_1.
On dit que s_1 est un **prédécesseur** de s_2.

Définition :
- le degré sortant du sommet s noté deg_+ est le nombre de successeurs de s.
- le degré entrant du sommet s noté deg_- est le nombre de prédécesseurs de s.

Le degré du sommet s est : $deg(s) = deg_+ + deg_-$

Exemple :



$deg(A) =$
$deg(B) =$
$deg(C) =$
$deg(D) =$

Cas non orienté
Le degré du sommet s est : $deg(s) =$ nombre d'extrémités d'arc (d'arêtes) reliée(s) au sommet s.

Proposition :

La somme des degrés de tous les sommets d'un graphe est égale ...

4 Représentation d'un graphe

4.1 Dictionnaire des successeurs (cas d'un graphe orienté)

4.2 Dictionnaire des prédécesseurs (cas d'un graphe orienté)

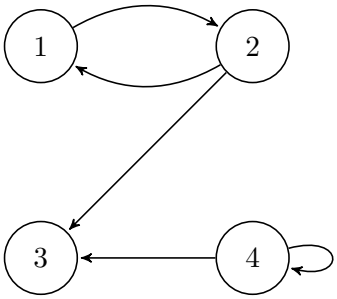
4.3 Matrice d'adjacence (quelque soit le graphe)

Définition :

Soit $G = (S, A)$ un graphe dont les sommets sont numérotés de 1 à n. La **matrice d'adjacence** de G est la matrice M dont les coefficients sont :

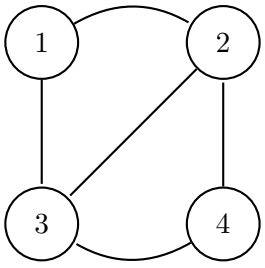
$$m_{ij} = \begin{cases} \text{pondération ou nb arête (ou nb d'arc) si } (i,j) \in A \\ 0 \text{ sinon} \end{cases}$$

Exemple 0 :



i \ j	1	2	3	4
1	0	1	0	0
2	1	0	1	0
3	0	0	0	0
4	0	0	1	1

Exemple 1 : : Construire la matrice d'adjacence de



i \ j	1	2	3	4
1				
2				
3				
4				

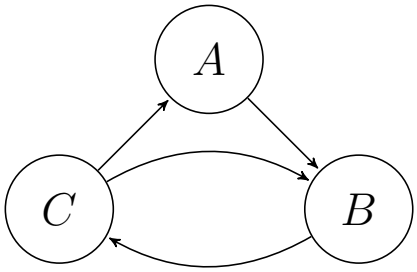
Exemple 2 : : Construire le graphe orienté donc la matrice d'adjacence est :

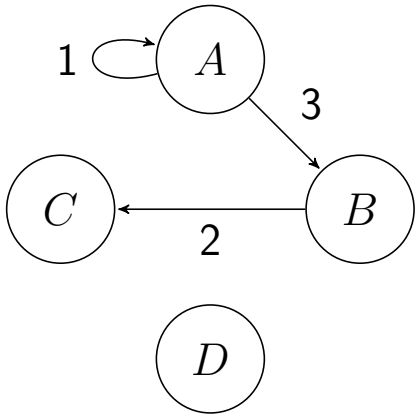
i \ j	1	2	3
1	1	0	1
2	0	0	1
3	1	0	0

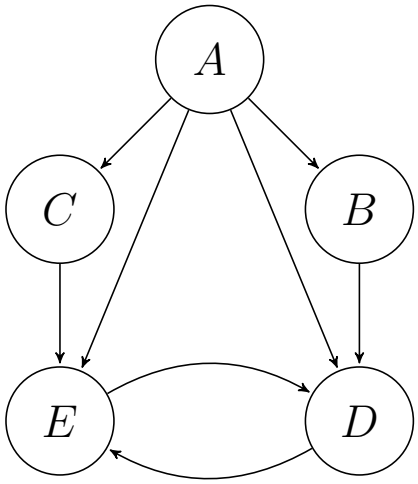
Exercice n° 1

Pour chacun des trois graphes ci-dessous, déterminer :

1. Le degré de chaque sommet.
2. Le dictionnaire des successeurs.
3. Le dictionnaire des prédécesseurs.
4. La matrice d'adjacence.







Exercice n° 2

1. Tracer le graph **NON orienté** dont la matrice d'adjacence est :

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

2. Tracer le graph **orienté** dont la matrice d'adjacence est :

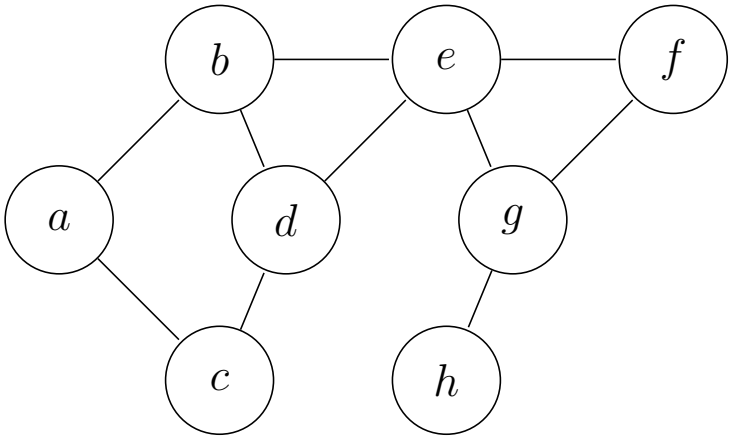
$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

3. Tracer le graphe oriente dont le dictionnaire des prédécesseurs est :

$$\{A : [C], B : [A, C, E], C : [A], D : [A], E : [D]\}$$

Exercice n° 3

On considère le graphe ci-dessous :



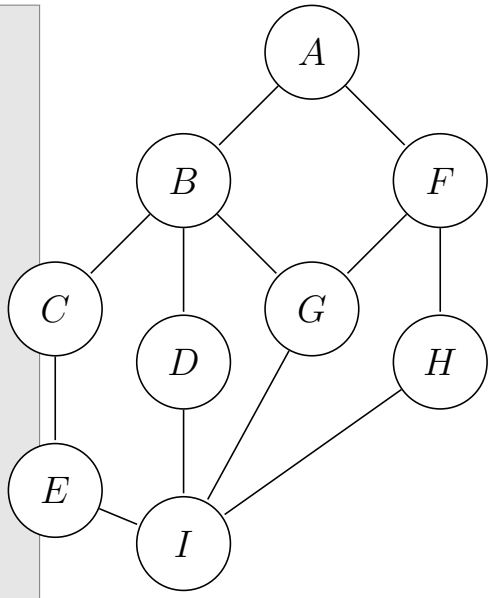
1. Partant du sommet **b**, proposer deux parcours en largueur de ce graphe.
2. Partant du sommet **b**, proposer deux parcours en profondeur de ce graphe.

Exercice n° 4 Parcours ...

Pour cet exercice, compléter le fichier parcours.py à récupérer sur bfourlegnie.com

1. Appliquer l'algorithme au graphe ci-dessous (on prendra A pour départ).

```
VARIABLE
s, u, v : noeud (s est le départ)
p : pile (vide)
visited : liste des noeuds visités (vide)
DEBUT
Ajouter s à visited
empiler s
tant que p non vide :
    u est le sommet de la pile
    depiler p
    Si u n'est pas dans visited
      ajouter u à visited
    Fin du si
    Pour chaque sommet v voisin du sommet u :
      Si v n'est ni dans visited ni dans la pile :
        ajouter v a visited
      Fin du si
    Fin du pour
Fin du tant que
Afficher visited
FIN
```



2. A l'aide du code ci-dessus, écrire en Python une fonction `profondeur` qui prend en paramètre un graphe (dict) et un sommet (str) et qui renvoie un parcours en profondeur des sommets sous la forme d'un liste.

Conditions d'utilisation :
- le graphe sera donné sous la forme du dictionnaire des successeurs.
- le sommet doit être une clef de ce dictionnaire.

3. Tester votre implémentation avec le graphe de l'exercice précédent dont voici le dictionnaire des successeurs :

```
G={'A':['B','F'],'B':['C','D','G'],'C': ['B','E'],
'D':['B','I'],'E':['C','I'],'F':['A','G','H'],
'G':['B','F','I'],'H':['F','I'],'I':['D','E','G','H']}
```

4. Réaliser une fonction `largeur` qui permet de réaliser un parcours en largeur d'un graphe donné sous la forme du dictionnaire des successeurs.

5. Tester vos implémentations de parcours avec le graphe de l'exercice précédent.

Aide :
```
G={'A':['B','C'],'B':['A','D','E'],'C': ['A','D'],
'D':['B','C','E'],'E':['B','D','G','F'],
'F':['E','G'],'G':['H','F','E'],'H':['G']}
```

Exercice n° 5

Écrire une fonction `convert` qui prend un graphe sous la forme d'un dictionnaire des successeurs et qui renvoie le dictionnaire des prédécesseurs.

Exercice n° 6

1. Les graphes ci-dessous contiennent-ils un cycle ?

2. A l'aide du parcours en largeur, écrire une fonction `contientCycle` qui permet de de détecter la présence d'un cycle dans un graphe non orienté.