Correction TP 6 - Graphes

En première année, vous avez déjà étudié les graphes, voici quelques activités.

1 Recherche d'un chemin avec une liste d'adjacence

Les problèmes de passage de rivière sont des exercices relevant de jeux mathématiques ou de réflexions. Certains sont très anciens, tels ceux posés au $VIII^{e}$ siècle par l'abbé de Cantorbéry, Alcuin, dont le plus connu est le problème du loup, de la chèvre et des choux :

Exercice 1.

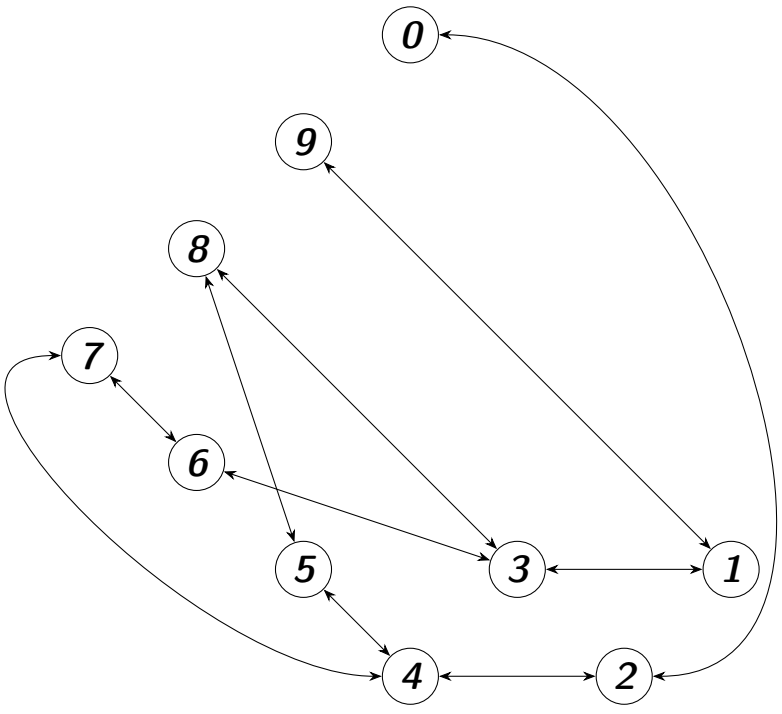
> *Un fermier doit passer la rivière dans une barque juste assez grande pour lui et son loup, ou lui et sa chèvre, ou lui et ses choux. Les choux seront mangés s'il les laisse seuls avec la chèvre, et la chèvre sera mangée s'il la laisse seule avec le loup. Comment faire passer tout ce monde sans dégâts ?*

Cette situation - qui peut être résolue "de tête" - peut aussi être modélisée à l'aide d'un graphe dont les sommets sont les situations.

1. *Ecrire tous les couples possibles représentant une situation en tenant compte des contraintes. Il y en a 10.*

0	Fermier / Chèvre / Choux / Loup $\approx$	
1	Fermier / Chèvre $\approx$	$\approx$ Choux / Loup
2	Choux / Loup $\approx$	$\approx$ Fermier / Chèvre
3	Chèvre $\approx$	$\approx$ Fermier / Choux / Loup
4	Fermier / Choux / Loup $\approx$	$\approx$ Chèvre
5	Loup $\approx$	$\approx$ Fermier / Choux / Chèvre
6	Fermier / Choux / Chèvre $\approx$	$\approx$ Loup
7	Choux $\approx$	$\approx$ Fermier / Chèvre / Loup
8	Fermier / Chèvre / Loup $\approx$	$\approx$ Choux
9		$\approx$ Fermier / Chèvre / Choux / Loup

2. *On place un arc dans ce graphe à chaque fois que l'on peut passer d'une situation à une autre.*
   (a) *Dessiner les 20 arcs de ce graphe.*



(b) *Ce graphe est-il orienté ?*
Toutes les flèches vont dans les deux sens donc le graphe n'est pas orienté.

3. *Avec Python :*

(a) *Ecrire la* ***liste d'adjacence*** *de ce graphe sous forme d'un dictionnaire : les clés sont les sommets et les valeurs les listes des sommets voisins.*

```
G={0:[2],1:[3,9],2:[0,4],3:[1,6,8],4:[2,5,7],5:[4,8],6:[3,7],7:[4,6],8:[3,5],9:[1]}
```

On va utiliser un ***parcours en largeur*** *pour trouver tous les chemins reliant le sommet 0 au sommet 9.*

(b) *Compléter la fonction ci-dessous* `suivant` *qui reçoit un chemin donné sous forme d'une liste de sommets pour qu'elle renvoie la liste des chemins qu'il est possible de réaliser avec un pas de plus.*

```
G={0:[2],1:[3,9],2:[0,4],3:[1,6,8],4:[2,5,7],5:[4,8],6:[3,7],7:[4,6],8:[3,5],9:[1]}

def suivant(chemin) :
    # On initialise la liste des chemins
    reponse = []
    # Dernier sommet du chemin
    s = chemin[-1]
    # Liste des voisins de s
    voisins = G[s]
    for v in voisins :
        reponse.append(chemin + [v])
    return reponse
```

(c) *Modifier la fonction précédente pour qu'on ne passe pas 2 fois par le même sommet.*

```
def suivantSansRetour(chemin) :
    # On initialise la liste des chemins
    reponse = []
    # Dernier sommet du chemin
    s = chemin[-1]
    # Liste des voisins de s
    voisins = G[s]
    for v in voisins:
        if v not in chemin:
            reponse.append(chemin + [v])
    return reponse
```

(d) *En déduire un script pour trouver tous les chemins possibles reliant le sommet 0 au sommet 9.*
On commence par lister tous les chemins possibles partant de 0*, puis on garde ceux qui partent de* 0 *et arrivent à* 9*.*

```
def CheminsPossibles():
    chemin=[0]
    possible=[chemin]
    for k in range(10):
        for v in possible:
            a=suivantSansRetour(v)
            if a!=[]:
                possible=possible+a
                possible.remove(v)
    for v in possible:
        if v[-1]!=9:
            possible.remove(v)
    return possible
```

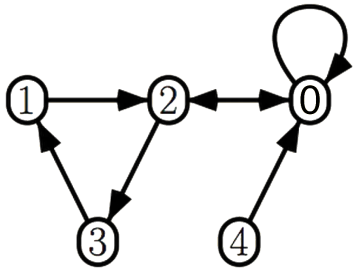
2 Nombre de chemins et matrice d'adjacence

On rappelle que la matrice d'adjacence d'un graphe possédant les sommets numérotés de 1 à n est la matrice $M = (m_{i,j})$ de taille $n \times n$ telle que $m_{i,j} = 1$ si il existe un arc du sommet i vers le sommet j.

On admet le théorème suivant :

> Soit G un graphe et M sa matrice d'adjacence, alors le coefficient de la matrice M^n situé à la ligne i et à la colonne j donne le nombre de chemins de longueur n reliant le sommet i au sommet j.

Exercice 2. *On considère le graphe ci-dessous :*



1. *Ecrire la matrice d'adjacence M du graphe et la définir dans Python grâce à la commande `array` du module `numpy`.*
   La matrice d'adjacence est la suivante :

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

   En Python, on la définit grâce aux lignes suivantes :

```
import numpy as np
M=np.array([[1,0,1,0,0],[0,0,1,0,0],[1,0,0,1,0],[0,1,0,0,0],[1,0,0,0,0]])
```

2. *Calculer M^2 et M^3 à l'aide de Python.*
   On calcule M^2 et M^3 grâce aux lignes suivantes :

```
M2=np.dot(M,M)
M3=np.dot(M2,M)
```

   On obtient les matrices suivantes :

```
>>> M2
array([[2, 0, 1, 1, 0],
       [1, 0, 0, 1, 0],
       [1, 1, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [1, 0, 1, 0, 0]])

>>> M3
array([[3, 1, 2, 1, 0],
       [1, 1, 1, 0, 0],
       [2, 0, 2, 1, 0],
       [1, 0, 0, 1, 0],
       [2, 0, 1, 1, 0]])
```

3. *Peut-on aller du sommet 1 à 2 en :*
   (a) *1 étape ? Oui car il y a un 1 en ligne 2 colonne 3 de la matrice d'ajacence M.*
   (b) *2 étapes ? Non car il y a un 0 en ligne 2 colonne 3 de la matrice M^2.*
   (c) *3 étapes ? Oui car il y a un 1 en ligne 2 colonne 3 de la matrice M^3.*
4. *Combien de cycles de longueur 3 y a-t-il ? (Un cycle est un chemin qui commence et se termine sur le même sommet).*
   On cherche les 3-cycles, donc on regarde les coefficients diagonaux de M^3 : il y a 3 3−cycles de 0 à 0, 1 3−cycle de 1 à 1, etc. On obtient donc, en additionnant les coefficients diagonaux de M^3, 7 3−cycles.

3 Recherche du plus court chemin : algorithme de Dijkstra

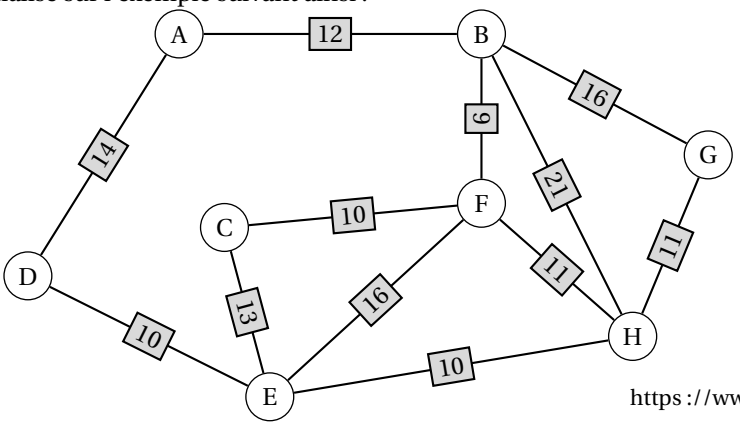
On se donne un graphe pondéré $\mathcal{G} = (S, \Sigma, p)$. On suppose de plus que les poids sont tous des réels positifs.

- Un **chemin** dans le graphe est une suite de sommets $(x_0, x_1, \ldots, x_N)$ dont les sommets consécutifs sont adjacents : pour tout $i \in [\![0, N-1]\!]$ on a $(x_i, x_{i+1}) \in \Sigma$.
- Le **poids d'un chemin** est la somme des poids des arêtes qui le compose : $\pi(x_0, \ldots, x_N) = \sum_{i=0}^{N-1} p(x_i, x_{i+1})$.

Étant donnés deux sommets i et j, on cherche à déterminer le chemin de poids minimal reliant i à j. L'algorithme de **Dijkstra** est un algorithme de détermination d'un chemin de poids minimal.

Le principe qui sous-tend l'algorithme est que si $x_0, x_1, \ldots, x_N$ est un plus court chemin de x_0 à x_N alors tout sous chemin $x_p, \ldots, x_q$ est le plus court chemin de x_p à x_q. On cherche ainsi à construire de proche en proche un chemin en choisissant parmi tous les sommets possibles pour poursuivre la marche celui qui n'a pas encore été visité et dont l'évaluation de la distance à x_0 est la plus petite possible.

Plus précisément, l'algorithme de Dijkstra calcule la distance minimale entre un sommet de départ (ici A) et un sommet d'arrivée (ici H) et consiste à mettre à jour un tableau de distances où chaque colonne correspond à un sommet que l'on initialise sur l'exemple suivant ainsi :



https ://www.youtube.com/watch?v=rI-Rc7eF4iw

A	B	C	D	E	F	G	H	Choix
0	∞	∞	∞	∞	∞	∞	∞	$A(0)$

A partir de A, on peut aller en $D(+14)$ ou en $B(+12)$. On met à jour le tableau en calculant les distances et en mémorisant pour la suite d'où on vient :

A	B	C	D	E	F	G	H	Choix
0	∞	∞	∞	∞	∞	∞	∞	$A(0)$
	$0+12=12_A$	∞	$0+14=14_A$	∞	∞	∞	∞	$B(12)$

Le plus court chemin pour aller de B à A est à présent connu, il a un coup de 12. On fixe alors le sommet B et on repars de celui-ci, on peut aller à A (déja traité), à $F(+9)$ à $G(+16)$ ou à $H(+21)$:

A	B	C	D	E	F	G	H	Choix
0	∞	∞	∞	∞	∞	∞	∞	$A(0)$
	12_A	∞	14_A	∞	∞	∞	∞	$B(12)$
		∞	14_A	∞	$12+9=21_B$	$12+16=28_B$	$12+21=33_B$	$D(14)$

On choisit alors le sommet D qui a la distance totale la plus petite. Noter, que l'on a déjà trouvé un chemin allant de H à A, mais on peut espérer trouver plus court. De D, on peut aller à A (déjà traité) et à $E(+10)$:

A	B	C	D	E	F	G	H	Choix
0	∞	∞	∞	∞	∞	∞	∞	$A(0)$
	12_A	∞	14_A	∞	∞	∞	∞	$B(12)$
		∞	14_A	∞	21_B	28_B	33_B	$D(14)$
		∞		$14+10=24_D$	21_B	28_B	33_B	$F(21)$

De F, on peut aller à $C(+10)$, $E(+16)$ mais dans ce cas, on a un coup de 37 qui est moins bien que les 24 déjà trouvés. Par contre, on peut aller à $H(+11)$ avec un coup de 32 qui est mieux, on ne met donc pas la colonne E à jour, mais juste la colonne H :

A	B	C	D	E	F	G	H	Choix
$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	$A(0)$
	$\boxed{12_A}$	∞	14_A	∞	∞	∞	∞	$B(12)$
		∞	$\boxed{14_A}$	∞	21_B	28_B	33_B	$D(14)$
		∞		24_D	$\boxed{21_B}$	28_B	33_B	$F(21)$
		$21+10=31_F$		$\boxed{21+16=37_F>24_D}$		28_B	$21+11=32_F<33_B$	$E(24)$

On continue ainsi en partant de E, on ne peut aller qu'à $C(+13)$ et $H(+10)$:

A	B	C	D	E	F	G	H	Choix
$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	$A(0)$
	$\boxed{12_A}$	∞	14_A	∞	∞	∞	∞	$B(12)$
		∞	$\boxed{14_A}$	∞	21_B	28_B	33_B	$D(14)$
		∞		24_D	$\boxed{21_B}$	28_B	33_B	$F(21)$
		31_F		$\boxed{24_D}$		28_B	32_F	$E(24)$
		$24+13=37_E>31_F$				$\boxed{28_B}$	$24+10=34_E>32_F$	$G(28)$

On fixe G, on ne peut aller qu'en $H(+11)$:

A	B	C	D	E	F	G	H	Choix
$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	$A(0)$
	$\boxed{12_A}$	∞	14_A	∞	∞	∞	∞	$B(12)$
		∞	$\boxed{14_A}$	∞	21_B	28_B	33_B	$D(14)$
		∞		24_D	$\boxed{21_B}$	28_B	33_B	$F(21)$
		31_F		$\boxed{24_D}$		28_B	32_F	$E(24)$
		31_F				$\boxed{28_B}$	32_F	$G(28)$
		$\boxed{31_F}$					$28+11=39_C>32_F$	$C(31)$

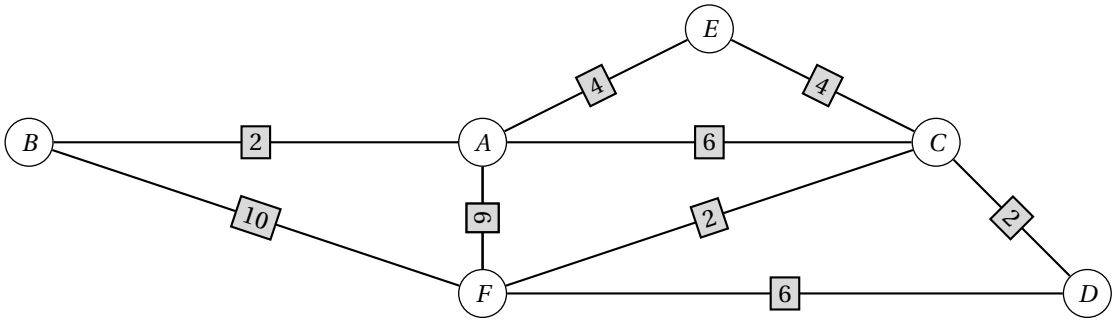
On fixe, C, on ne peut aller nulle part

A	B	C	D	E	F	G	H	Choix
$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞	$A(0)$
	$\boxed{12_A}$	∞	14_A	∞	∞	∞	∞	$B(12)$
		∞	$\boxed{14_A}$	∞	21_B	28_B	33_B	$D(14)$
		∞		24_D	$\boxed{21_B}$	28_B	33_B	$F(21)$
		31_F		$\boxed{24_D}$		28_B	32_F	$E(24)$
		31_F				$\boxed{28_B}$	32_F	$G(28)$
		$\boxed{31_F}$					32_F	$C(31)$
							$\boxed{32_F}$	$H(32)$

On fixe H. On sait alors que la distance minimale est de 32. Comme on a marqué d'où l'on vient, on peut retrouver le chemin en partant de l'arrivée : $H \to F \to B \to A$. Le chemin le plus court est donc (A, B, F, H).

Remarque : on voit aussi avec ce tableau comment aller de A à G par exemple : (A, B, F, G)

Exercice 3. *https ://www.youtube.com/watch ?v=vJOHB2yXI_s*
Soit le graphe suivant :



1. *Ecrire la matrice d'adjacence associée.*
   La matrice d'ajacence associée est :

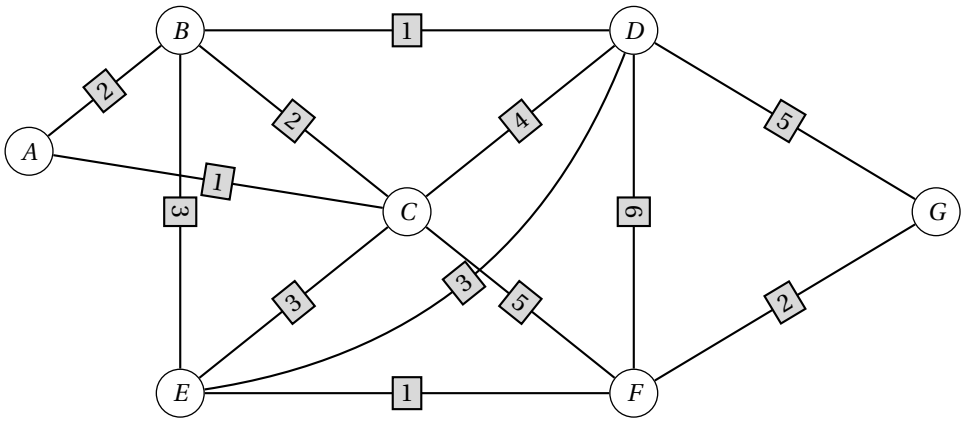
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

2. *Appliquer l'algorithme de Dijkstra pour déterminer le plus court chemin entre D et B.*
   On crée le tableau suivant en appliquant l'algorithme de Dijkstra :

A	*B*	*C*	*D*	*E*	*F*	*Choix*
∞	∞	∞	*0*	∞	∞	*D(0)*
∞	∞	$\boxed{2_D}$		∞	6_D	*C(2)*
8_C	∞			6_C	$\boxed{4_C}$	*F(4)*
8_C	14_F			$\boxed{6_C}$		*E(6)*
$\boxed{8_C}$	14_F					*A(8)*
	$\boxed{10_A}$					*B(10)*

On en déduit que le plus court chemin de D à A est D − C − A − B, de longueur 10.

Exercice 4. *https ://www.youtube.com/watch ?v=k7MsXexTIgE*
Soit le graphe suivant :



1. *Ecrire la matrice d'adjacence associée.*
   La matrice d'ajacence associée est :

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

2. *Appliquer l'algorithme de Dijkstra pour déterminer le plus court chemin entre A et G.*
   On crée le tableau suivant en appliquant l'algorithme de Dijkstra :

A	*B*	*C*	*D*	*E*	*F*	*G*	*Choix*
0	∞	∞	∞	∞	∞	∞	*A(0)*
	2_A	$\boxed{1_A}$	∞	∞	∞	∞	*C(1)*
	$\boxed{2_A}$		5_C	4_C	6_C	∞	*B(2)*
			$\boxed{3_B}$	4_C	6_C	∞	*D(3)*
				$\boxed{4_C}$	6_C	8_D	*E(4)*
					$\boxed{5_E}$	8_D	*F(5)*
						$\boxed{7_F}$	*G(7)*

   On en déduit que le plus court chemin de A à G est $A-C-E-F-G$, de longueur 7.