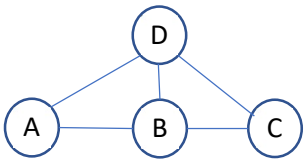
1 Exercices

1.1 Matrice d'adjacence par liste de listes

Le graphe ci-dessous est considéré.



Les arêtes entre les sommets peuvent être représentées dans un tableau (voir ci-contre).

Q1. Compléter le tableau des liens.

	A	B	C	D	
A	Non	Oui	Non	Oui	0
B	Oui	Non	Oui	Oui	1
C	Non	Oui	Non	Oui	2
D	Oui	Oui	Oui	Non	3
	0	1	2	3	

Une matrice d'adjacence est une modélisation mathématique du tableau des liens (voir ci-contre).
Une matrice d'adjacence peut s'implémenter sous python sous forme de sous-listes imbriquées dans une liste principale (chaque sous-liste représente une ligne du tableau précédent). La présence d'une arête peut être représentée par un booléen True ou par un entier 1.

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

Q2. Écrire la définition de la matrice d'adjacence `m` sous forme de liste de listes python.

```
m=[[0,1,0,1],
   [1,0,1,1],
   [0,1,0,1],
   [1,1,1,0]]
```

Q3. Que vaut `len(m)` ?

Il y a 4 listes dans `m` donc `len(m) = 4`.

Q4. Écrire une fonction `Nb_aretes` qui prend en argument une matrice `m` représentant un graphe et qui renvoie le nombre d'arêtes d'un graphe non orienté.

```
def Nb_aretes(m):
    #initialisation
    nb=0
    n=len(m)
    #parcours de toutes les lignes
    for i in range(n):
        #parcours du côté haut à gauche
        for j in range(i+1,n):
            if m[i][j]==1:
                nb=nb+1
    return nb
```

Q5. Écrire une fonction `ajout_sommet_isole` qui prend en argument un graphe `g` représenté par une matrice d'adjacence de listes et qui complète la matrice par un sommet isolé.

Il faut rajouter un 0 à la fin de chaque ligne de la liste et rajouter une liste de 0.

```
def ajout_sommet_isole(g):
    #ajout d'un 0 à la fin de chaque ligne
    for ligne in g:
        ligne.append(0)
    #ajout d'une ligne de 0
    ligne=[0 for i in range(len(g)+1)]
    g.append(ligne)
    return g
```

Q6. Écrire une fonction `ajout_arete` qui prend en argument un graphe g sous forme définie précédemment et une arête a et qui ajoute l'arête au graphe. Une arête est représentée par une liste de deux indices. Ces indices correspondent à l'emplacement du nœud dans la matrice (càd, nœud A à l'indice 0, nœud B à l'indice 1…)

```
def ajout_arete(g,L):
    g[L[0]][L[1]]=1
    g[L[1]][L[0]]=1
    return g
```

Q7. Créer un dictionnaire d dont les clés sont les indices et les valeurs associées sont les noms de sommets.	`d={0:"A",1:"B",2:"C",3:"D",4:"E"}`

Q8. Écrire une fonction `liste_arete` qui prend en argument un graphe g sous forme définie précédemment et un dictionnaire d qui recense la position des sommets dans la matrice et qui renvoie la liste des arêtes du graphe. Pour cette question, chaque arête sera représentée par une liste qui contient les 2 sommets extrémité.	`def liste_aretes(g,d):`
	`  **#initialisation**`
	`  liste=[]`
	`  n=len(g)`
	`  **#parcours de toutes les lignes**`
	`  for i in range(n):`
	`  **#parcours du côté haut à gauche**`
	`  for j in range(i+1,n):`
	`  if g[i][j]==1:`
	`  liste.append([d[i],d[j]])`
	`  return liste`