TD Informatique TSI2

GRAPHS : definition and representation

SOLUTION

EXERCICE 01

Parmi les assertions suivantes, lesquelles sont vraies ?

A. Un graphe ne peut pas avoit plus de sommets que d'arêtes ?

Un graphe peut avoir des sommets isolés et donc avoir plus de sommets que d'arêtes. Ainsi, **l'assertion A est False**

B. Une matrice d'adjacence d'un graphe à n sommets contient $2n$ coefficients ?

En fait c'est une matrice carrée d'ordre n et contient n^2 coefficients. Ainsi, **l'assertion B est False**

C. Une matrice d'adjacence d'un graphe orienté est symétrique ?

Ce n'est le cas que si chaque arête a les deux sens d'orientation ce qui n'est pas obligatoire. Par contre la matrice est symétrique pour un graphe non orienté. Ainsi, **l'assertion C est False**

D. L'ordre d'un graphe est le nombre d'arcs ou d'arêtes ?

L'ordre est le nombre de sommets. Ainsi, **l'assertion D est False**

E. Un graphe est connexe si chaque sommet est adjacent à tous les autres ?

Un graphe est connexe si pour tout couple de sommets, il existe un chemin (c'est-à-dire une séquence $(s_0, s_1, ..., s_n)$ où deux sommets consécutifs sont adjacents) reliant ces deux sommets. Cela n'implique pas que chaque sommet doit être adjacent à tous les autres. Ainsi, **l'assertion E est False**

F. Un chemin qui passe deux fois par le même sommet contient un cycle ?

Si s est ce sommet, alors le chemin $(s, s_1, ..., s_n, s)$ est un cycle. Ainsi, **l'assertion F est True**

G. Un graphe non orienté d'ordre 5 a au maximum 10 arêtes reliant des sommets distincts ?

On peut par exemple remarquer que le nombre d'éléments de la matrice d'adjacence contient alors au plus 20 coefficients non nuls. Ainsi, **l'assertion G est True**

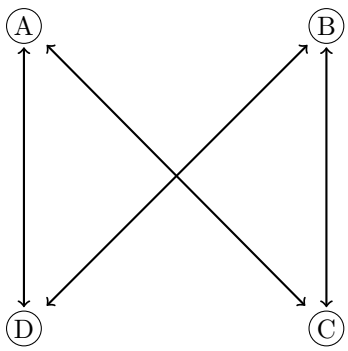
EXERCICE 02

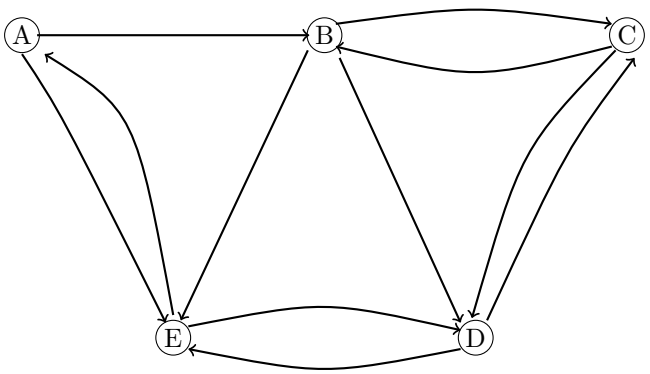
1. On a $M_1 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$ et $M_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

2.
```
Graphe1 = { 'A': ['B', 'D'], 'B': ['A','C','D'], 'C': [ 'B', 'D'], 'D': ['A','B','C'] }
```
et
```
Graphe2 = { 'A': ['D'], 'B': ['A','C','D'], 'C': [ 'B', 'D'], 'D': ['B'] }
```

EXERCICE 03

• À gauche, le graphe (qu'on peut prendre non orienté car la matrice est symétrique) associé à la matrice d'adjacence M_1 et à droite le graphe (orienté nécessairement) associé à la matrice d'adjacence M_2 :





EXERCICE 04

1. On tape la fonction.

```
In [1]: def UNKNOWN(m):
   ...:     nb = 0
   ...:     n = len(m)
   ...:     for i in range(n):
   ...:         for j in range(i+1,n):
   ...:             if m[i][j] == 1 :
   ...:                 nb = nb + 1
   ...:     return nb
```

Faisons à la main le cas de `m = [[0,1,0,1], [1,0,1,1],[0,1,0,1],[1,1,1,0]]` :

On commence `nb = 0` et `n = 4`
D'abord `i=0` et `for j in range(1,4):`
`j=1` et `m[0][1] == 1` est `True` et donc `nb=1`
`j=2` et `m[0][2] == 1` est `False` et donc `nb=1`
`j=3` et `m[0][3] == 1` est `True` et donc `nb=2`
Puis `i=1` et `for j in range(2,4):`
`j=2` et `m[1][2] == 1` est `True` et donc `nb=3`
`j=3` et `m[1][3] == 1` est `False` et donc `nb=4`
Enfin `i=2` et `for j in range(3,4):`
`j=3` et `m[2][3] == 1` est `True` et donc `nb=5`
On arrête et donc on renvoie `nb = 5` et la fonction `UNKNOWN` qui prend en paramètre une matrice `m` représentant un graphe et renvoie le nombre d'arêtes du graphe. En effet, comme la matrice est symétrique, on ne s'intéresse qu'à `m[i][j]` avec `i < j` et chaque fois que `m[i][j] == 1` est `True` alors cela signifie que les deux points sont rejoints et on a une arête.

```
In [2]: m = [[0,1,0,1], [1,0,1,1],[0,1,0,1],[1,1,1,0]]
In [3]: UNKNOWN(m)
Out[3]: 5
```

2. `Graphe1 = { 'A': ['B', 'D'], 'B': ['A','C','D'], 'C': [ 'B', 'D'], 'D': ['A','B','C'] }`
C'est le même graphe qu'à l'exercice 02.

3.

```
In [1]: def sommets(s,g):
   ...:     # on teste si s est une cle de g
   ...:     if s in g:
   ...:         return g[s]
In [2]: Graphe1 = {'A':['B','D' ],'B':['A','C','D'],'C':['B','D'],'D':['A'
   ,'B','C']}
In [3]: sommets('A',Graphe1)
Out[3]: ['B', 'D']
```

On peut améliorer la fonction.

```
In [4]:  def new_sommets(s,g):
   ...:     if s in g:
   ...:         return g[s]
   ...:     else :
   ...:         print("cette valeur n'est pas un sommet du graphe")
In [5]: new_sommets('E',Graphe1)
cette valeur n'est pas un sommet du graphe
```

EXERCICE 05

1.

```
In [7]: def ajoute_sommet1(g,s):
   ...:     g[s]=[]
   ...:
In [8]: ajoute_sommet1(Graphe1, 'E')
In [9]: Graphe1
Out[9]:
{'A': ['B', 'D'],
 'B': ['A', 'C', 'D'],
 'C': ['B', 'D'],
 'D': ['A', 'B', 'C'],
 'E': []}
```

2.

```
In [10]: def ajoute_sommet2(g,s):
    ...:     g.append([s,[]])
    ...:
In [11]:  N_Graphe1 = [['A',['B','D' ]],['B',['A','C','D']],['C',['B','D'
    ]],['D',['A','B','C']]]
    ...:
In [12]: ajoute_sommet2(N_Graphe1,'E')
In [13]: N_Graphe1
Out[13]:
[['A', ['B', 'D']],
 ['B', ['A', 'C', 'D']],
 ['C', ['B', 'D']],
 ['D', ['A', 'B', 'C']],
 ['E', []]]
```

3.

```
In [1]: def ajoute_sommet3(g):
   ...:     # On commence par completer les lignes avec 0
   ...:     for ligne in g:
   ...:         ligne.append(0)
   ...:     # Puis on ajoute une ligne de 0
   ...:     new_ligne = [0 for i in range(len(g)+1)]
   ...:     g.append(new_ligne)

In [2]: M1 = [[0,1,0,1],[1,0,1,1],[0,1,0,1],[1,1,1,0]]

In [3]: ajoute_sommet3(M1)

In [4]: M1
Out[4]:
[[0, 1, 0, 1, 0],
 [1, 0, 1, 1, 0],
 [0, 1, 0, 1, 0],
 [1, 1, 1, 0, 0],
 [0, 0, 0, 0, 0]]
```

EXERCICE 06

1.

```
In [37]: for i in range(65,65+26):
    ...:     print(chr(i))
    ...:
Out[37]: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

On remarque que l'on obtient les lettres de l'alphabet.
On peut renvoyer d'autres symboles.

```
In [28]: chr(120), chr(256), chr(400)
Out[28]: 'x'  'Ä'  'Æ'
```

2. Dans la boucle `for j in range(len(m[i])):` donc on parcourt la ligne `i` et si `m[i][j] == 1` est `True` alors `[i]` et `[j]` sont reliés et dans la clé `sommets[i]`, on ajoute dans la valeur de cette clé qui est une liste l'élément `sommets[j]`

```
In [38]: def conversion(m):
    ...:     sommets = {}
    ...:     n = len(m)
    ...:     for i in range(n):
    ...:         sommets[i] = chr(65+i)
    ...:     g = {}
    ...:     for i in range(n):
    ...:         g[sommets[i]] = []
    ...:         for j in range(len(m[i])):
    ...:             if m[i][j] == 1 :
    ...:                 g[sommets[i]].append(sommets[j])
    ...:     return g

In [39]: conversion([[0,1,0,1],[1,0,1,1],[0,1,0,0],[1,1,0,0]])
Out[39]: {'A':['B','D'],'B':['A','C','D'],'C':['B'],'D': ['A','B']}
```