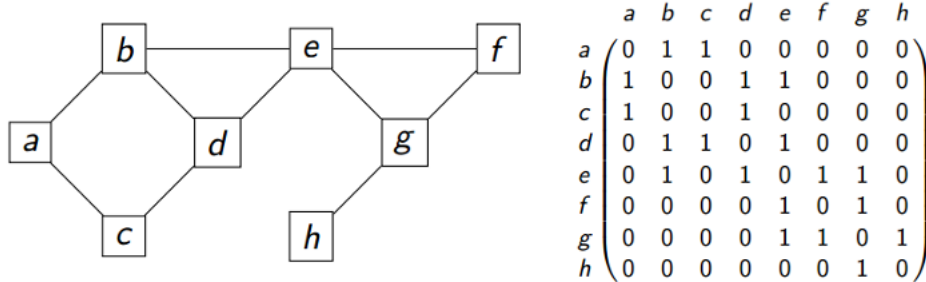


TD - Représentation d'un graphe - 2

Avec une bibliothèque

La bibliothèque networkX permet de manipuler des graphes.
Prenons par exemple ce graphe :



On importe le module

```
import networkx as nx
```

On crée un graphe vide

```
#création du graphe  
g1 = nx.Graph()
```

On ajoute les sommets (appelés node)

```
#création des sommets  
g1.add_node('a')  
g1.add_node('b')  
g1.add_node('c')  
g1.add_node('d')  
g1.add_node('e')  
g1.add_node('f')  
g1.add_node('g')  
g1.add_node('h')
```

On ajoute les arêtes (appelés edge)

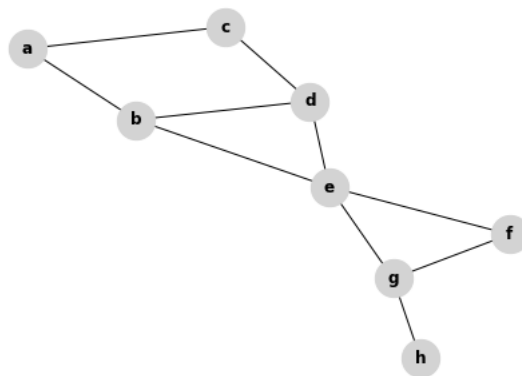
```
#Création des arêtes  
g1.add_edge('a', 'b')  
g1.add_edge('a', 'c')  
g1.add_edge('b', 'd')  
g1.add_edge('b', 'e')  
g1.add_edge('c', 'd')  
g1.add_edge('d', 'e')  
g1.add_edge('e', 'g')  
g1.add_edge('e', 'f')  
g1.add_edge('g', 'f')  
g1.add_edge('g', 'h')
```

On peut visualiser le graphe grâce à matplotlib et la fonction draw.

Ici on a de plus configuré l'affichage pour que les étiquettes des sommets soient affichées, la taille des sommets soit de 800, la couleur de fond des sommets gris clair.

```
import matplotlib.pyplot as plt

nx.draw(g1, with_labels=True, font_weight='bold',
node_size=800, node_color='lightgrey')
plt.show()
```



On peut également le faire avec des listes de sommets et d'arêtes :

```
import networkx as nx
import matplotlib.pyplot as plt
#création du graphe à partir de listes
liste1=['a','b','c','d','e','f','g','h']
g2 = nx.Graph()
g2.add_nodes_from(liste1)
liste2=[('a','b'),('a','c'),('b','d'),('b','e'),('c','d'),('d','e'),
('e','g'),('e','f'),('g','f'),('g','h')]
g2.add_edges_from(liste2)
nx.draw(g2, with_labels=True, font_weight='bold', node_size=800,
node_color='lightgrey')
plt.show()
```

Pour la matrice d'adjacence : networkx propose une méthode `nx.adjacency_matrix(g2)` qui stocke les coefficients a_{ij} de la matrice d'adjacence. Il suffit alors de remplir un tableau avec ces coefficients.

```
B = nx.adjacency_matrix(g2)
print(B[(0,0)])

n=len(liste1)
A=[[0]*n for i in range(n)]
for i in range(n):
    for j in range(n):
        A[i][j]= B[(i,j)]

print(A)
```

Exploiter la bibliothèque

La documentation de NetworkX est divisée en sections. Il existe notamment :

- une section pour obtenir les **méthodes** sur les sommets et les arêtes
- une section pour obtenir les algorithmes disponibles
- on pourra aussi consulter le tutoriel de NetworkX

En voici quelques unes

degrés d'un sommet du graphe g :	<code>g.predecessors(i)</code> : liste des prédecesseurs du sommet i
<code>g.degree('a')</code>	
nombre de sommets du graphe g :	<code>g.successors(i)</code> : liste des successeurs du sommet i
<code>g.number_of_nodes()</code>	
nombre d'arcs du graphe g :	<code>g.neighbors(i)</code> : liste des voisins du sommet i
<code>g.number_of_edges()</code>	

À FAIRE 1:

Avec NetworkX, cherchez les méthodes pour obtenir les informations suivantes sur le graphe G :

- le nombre de sommets du graphe

.....
.....

- le nombre d'arêtes du graphe

.....
.....

- le degré d'un sommet

.....
.....

- le sommet de plus haut degré

.....
.....
.....

- les voisins d'un sommet

.....
.....

À FAIRE 2:

Implémenter le graphe du réseau social du cours et faire afficher celui qui a le plus d'amis.