Graphes

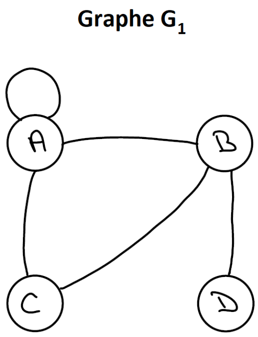
I/ Description

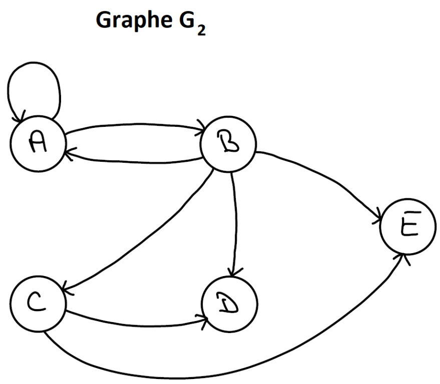
1. Représentation schématique et vocabulaire

Un ***graphe*** est un couple $G = (S, \mathcal{A})$ où :

- S est un ensemble de points appelés ***sommets***, ou ***noeuds***
- $\mathcal{A}$ est un ensemble de liens entre les sommets, appelés ***arêtes***

Un graphe est schématisé de la façon suivante :





Le graphe G_2 est appelé ***graphe orienté***, parce qu'on a imposé un sens de parcours sur ses arêtes. Dans ce cas, les arêtes sont des ***arcs***, représentés par des flèches.

En revanche, dans le graphe G_1, aucun sens de parcours n'est imposé. Les arêtes peuvent être parcourues dans les deux sens et on dit que le graphe est un ***graphe non orienté***.

Une arête qui va d'un sommet vers lui-même est appelée ***boucle***.

- Dans G_1 : ....
- Dans G_2 : ....

Le graphe G_1 peut également être défini par :

- G_1 est non orienté
- L'ensemble des sommets est $S = \{A, B, C, D\}$
- L'ensemble des arêtes est $\mathcal{A} = \{AA, AB, AC, BC, BD\}$

De même, pour le graphe 2 :

- G_2 est orienté
- L'ensemble des sommets est $S =$ ....
- L'ensemble des arêtes est $\mathcal{A} =$ ....

Deux sommets sont dits ***voisins*** s'ils sont reliés par une arête.

Dans un graphe orienté, on appelle ***chemin*** entre les sommets A et B une suite alternant sommets et arêtes reliant les sommet de départ A au sommet d'arrivée B.

Un graphe non orienté est ***connexe*** s'il est possible, à partir de n'importe quel sommet de rejoindre tous les autres sommets en suivant les arêtes. Un graphe non connexe se décompose en plusieurs sous graphes connexes.

2. Représentation par une matrice d'adjacence

La ***matrice d'adjacence*** M d'un graphe G est définie par :

- Si G a n sommets, M est carrée de taille $n \times n$
- Le coefficient (i, j) de M vaut 1 s'il existe une arête qui va du sommet i au sommet j et 0 sinon.

Ainsi, la matrice d'adjacence de G_1 et :

Et la matrice d'adjacence de G_2 est :

Donner la représentation schématique d'un graphe qui a pour matrice d'adjacence $M = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$

Pour un graphe non orienté, la matrice d'adjacence est symétrique ; elle ne l'est pas forcément pour un graphe orienté.

3. Représentation en Python par une liste d'adjacence

Un graphe est représenté en Python par sa ***liste d'adjacence***, c'est-à-dire par la liste des voisins de chaque sommet. Pour un graphe orienté, la liste d'adjacence contient la liste des sommets atteignables depuis chaque sommet.

Pour le graphe G_1, la liste d'adjacence est :

Pour le graphe G_2, la liste d'adjacence est :

Donner la représentation schématique d'un graphe non orienté ayant pour liste d'adjacence
$\mathsf{L} = [\,[1,2,4]\,,\ [0,2]\,,\ [0,1,3,4]\,,\ [2,3,4]\,,\ [0,2,3]\,]$:

4. Représentation en Python par un dictionnaire

Un graphe peut aussi être représenté en Python par un dictionnaire. Chaque clé représente un sommet. La valeur de la clé est la liste des sommets atteignables depuis cette clé.

Pour le graphe G_1, le dictionnaire est :

Pour le graphe G_2, le dictionnaire est :

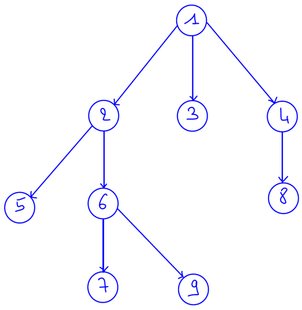
Ecrire une fonction Python **dictionnaire_vers_liste** qui prend en argument un dictionnaire dont on sait qu'il représente un graphe et qui renvoie la liste d'adjacence associée :

Ecrire une fonction Python **liste_vers_dictionnaire** qui prend en argument une liste d'adjacence et qui renvoie le dictionnaire associé :

II/ Parcours d'un graphe

1. Description du principe sur un exemple

On considère le graphe orienté suivant :



Le dictionnaire associé à ce graphe est :

dico_exemple =

Pour parcourir ce graphe, on propose le script suivant :

```
def enfiler(F,x):
    F.append(x)
def defiler(F):
    x = F.pop(0)
    return x
def parcourslargeur(dico,debut):
    L=[debut]
    while L!=[]:
        x = defiler(L)
        for elt in dico[x]:
            enfiler(L,elt)
```

Si on tape dans le shell la commande parcourslargeur(dico_exemple,1), les variables prennent les valeurs suivantes :

L	x	dico[x]

On retient le vocabulaire suivant :

- L est une **file**
- Quand on ajoute un élément x à la fin de la file, on dit qu'on l'**enfile**
- Quand on retire un élément x au début de la file, on dit qu'on le **défile**

2. Applications

1. Ecrire une fonction Python qui prend en argument une matrice d'adjacence et qui renvoie **True** si le graphe est connexe et **False** sinon.
2. Ecrire une fonction Python qui prend en argument un matrice d'adjacence et un sommet de départ et qui renvoie **True** s'il existe un circuit au départ de ce sommet et **False** sinon.