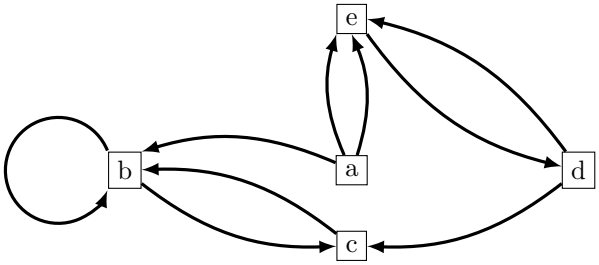
I Définition, premiers exemples

Définition

Un graphe G est un couple d'ensembles $G = (S, A)$, où :
- S est un ensemble de sommets,
- A est un multi-ensemble d'arcs (i, j) où i, j sont des sommets de S.

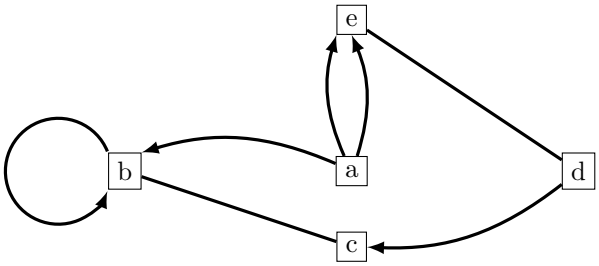
Exemple : $G = (S, A)$ avec $S = \{a, b, c, d, e\}$ et $A = \{\{(a, e), (a, e), (e, d), (d, e), (d, c), (c, b), (b, c), (b, b), (a, b)\}\}$
Ce graphe G se représente graphiquement par exemple comme ci-dessous :



Définition

Une **arête** est l'ensemble formé par deux arcs symétriques : $\{(i, j), (j, i)\} = \{i, j\}$.

Représentation :



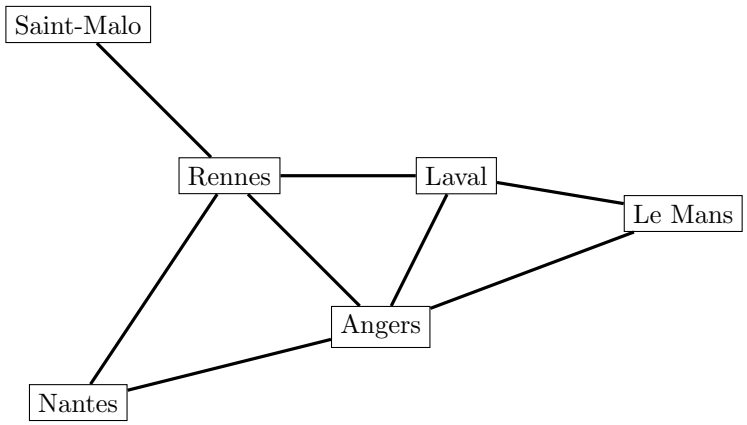
Définition

- Une arête ou un arc est **multiple** si elle apparaît plusieurs fois dans A.
- Une **boucle** est un arc reliant un sommet à lui-même.
- Un graphe $G = (S, A)$ est **non orienté** si A ne contient que des arêtes ou des boucles. Il est **orienté** sinon.
- Un graphe $G = (S, A)$ est **multiple** si A contient au moins un arc ou une arête multiple.
- Un graphe est **simple** s'il est non orienté, sans boucle, et non multiple.

Dans l'exemple ci-dessus, l'arc (a, e) est un arc multiple et l'arc (b, b) est une boucle.

Dans notre programme, on n'aura jamais de graphe avec des arêtes ou des arcs multiples.

Exemple de graphe simple :



II Représentation d'un graphe en langage *Python*

1 Si on peut numéroter les sommets du graphe 0 à $n-1$

Un graphe peut se définir comme une **liste d'adjacence**.
Dans le premier exemple, on peut renommer les sommets : $a \to 0$, $b \to 1, \dots, e \to 4$.
Le graphe G est alors défini par la liste : `G = [[1,4,4], [1,2], [1], [2,4], [3]]`.
`G[0] = [1,4,4]` signifie que le sommet 0 est relié aux sommets $1, 4$ et encore 4.
Cela représente les arcs : $(a,b), (a,e), (a,e)$ de l'ensemble A.
Cette liste est la *liste d'adjacence* du sommet 0.
Si un sommet n'est relié à aucun autre, sa liste d'adjacence est vide : `G[i] = []`.

2 Si les sommets portent des noms qu'on doit conserver

On représente dans ce cas un graphe par un dictionnaire, où les clés sont les sommets, et les valeurs associées sont les listes d'adjacence du sommet :
`G = {'a':['b','e','e'], 'b':['b'], 'c':['b'], 'd':['c','e'], 'e':['d']}`.
`H = {'Saint-Malo':['Rennes'], 'Rennes':['Saint-Malo', 'Laval', 'Angers', 'Nantes'], ...}`

III Matrice d'adjacence

Définition

Lorsque les sommets sont numérotés de 0 à $n-1$, la **matrice d'adjacence** M d'un graphe $G = (S, A)$ est la matrice de $\mathcal{M}_n(\mathbf{R})$ définie par : $\forall (i,j) \in [\![1,n]\!]^2,\ M_{i,j}$ est le nombre d'arcs ou d'arêtes reliant le sommet $(i-1)$ au sommet $j-1$.

Proposition

- G est non multiple ssi les coefficients de sa matrice d'adjacence appartiennent à $\{0,1\}$.
- G est sans boucle ssi sa matrice d'adjacence a une diagonale nulle.
- G est non orienté ssi sa matrice d'adjacence est symétrique.

Exemple : La matrice d'adjacence du graphe G : $M = \begin{pmatrix} 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

IV Chemin (ou parcours)

Définition

Un **chemin** (ou **parcours**) reliant deux sommets a et b d'un graphe $G = (S, A)$ est une suite d'arcs ou d'arêtes : $(a_0, a_1), (a_1, a_2), \dots, (a_{\ell-1}, a_\ell)$ ou $\{a_0, a_1\}, \{a_1, a_2\}, \dots, \{a_{\ell-1}, a_\ell\}$
telle que : $\begin{cases} a_0 = a,\ a_\ell = b \\ \forall i \in [\![0, \ell-1]\!],\ (a_i, a_{i+1}) \in A \text{ ou } \{a_i, a_{i+1}\} \in A \end{cases}$
L'entier ℓ est alors la **longueur** du chemin.

Exemple : dans le graphe G, $(a,e), (e,d), (d,e), (e,d), (d,c)$ est un chemin de longueur 5 reliant a à c.
Il n'existe pas dans ce graphe de chemin reliant c à a.

Remarque : Pour un graphe non multiple, un chemin peut être donné uniquement pas la liste des sommets successivements visités.
Exemple : 'Saint-Malo', 'Rennes', 'Angers', 'Laval' est un chemin défini de façon non ambiguë.

Théorème **(Hors-programme)**

Soit M la matrice d'adjacence d'un graphe G d'ordre n (nombre de sommets).
Alors $\forall \ell \geqslant 1, \forall (i,j) \in [\![1,n]\!]^2,\ \left(M^\ell\right)_{i,j}$ est le nombre de chemin distincts de longueur ℓ reliant le sommet $(i-1)$ au sommet $(j-1)$.

Définition

Un sommet j est dit **connecté à** un sommet i s'il existe un chemin de i vers j.
Par convention, on dira de tout sommet qu'il est connecté à lui-même (par le chemin vide).

Remarque : si G n'est pas orienté, j connecté à $i \Leftrightarrow i$ connecté à j.
On dit dans ce cas que : i et j sont connectés.

Définition

Un graphe G est **connexe** lorsque tout sommet j est connecté à tout sommet i.

Définition

Soit i un sommet d'un graphe G non orienté.
On appelle **composante connexe** de i l'ensemble des sommets j connectés à i.

Définition

Soient i, j deux sommets d'un graphe, tels que j est connecté à i.
∗ On appelle **distance** de i vers j la longueur minimale d'un chemin de i vers j.
∗ On appelle **diamètre** d'un graphe connexe la plus grande distance entre 2 de ses sommets.

V Exercices

1 Matrice d'adjacence

Un graphe G étant donné par une liste d'adjacence, écrire une fonction `mat(G)` renvoyant la matrice d'adjacence du graphe G.

Rappel : on importe le module numpy : `import numpy as np`
et on peut définir la matrice carrée nulle de taille n par : `M = np.zeros((n,n))`.

2 Graphe orienté

Écrire une fonction testant si un graphe est orienté ou non (en renvoyant un booléen).

3 Nombre de chemins

En utilisant le théorème du chapitre **IV**, écrire une fonction renvoyant le nombre de chemins de longueur ℓ reliant deux sommets donnés d'un graphe.

4 Test de connexion

Écrire une fonction testant si deux sommets i, j d'un graphe vérifient : j est connecté à i.

5 Graphe connexe

Écrire une fonction testant si un graphe est connexe.

6 Chemin minimal

Soit G un graphe non multiple et i, j deux sommets de G tels que j est connecté à i.
Écrire une fonction renvoyant un chemin de longueur minimale de i vers j, sous forme d'une liste des sommets visités.
En déduire une fonction donnant la distance de i vers j.

7 Diamètre

Écrire une fonction renvoyant le diamètre d'un graphe connexe.

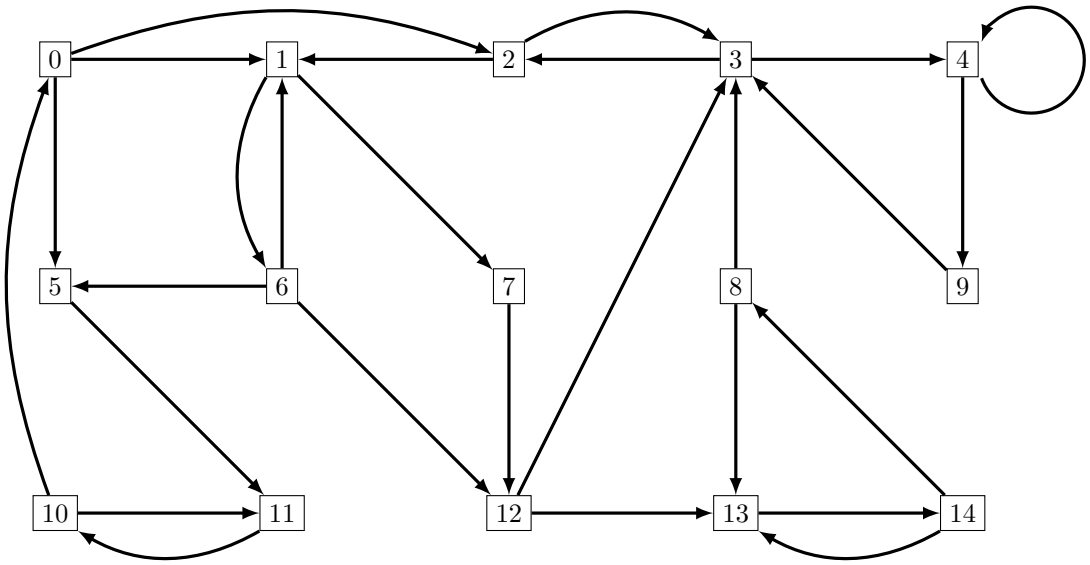
8 Composante connexe

Écrire une fonction renvoyant la composante connexe d'un sommet donné.

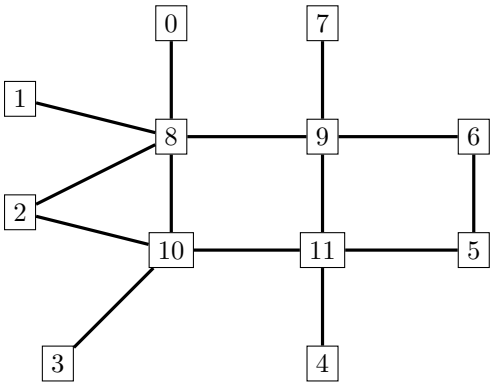
VI Des graphes pour tester vos fonctions

Les graphes représentés ci-dessous sont définis dans le fichier *Graphes.py* par des listes d'adjacence.

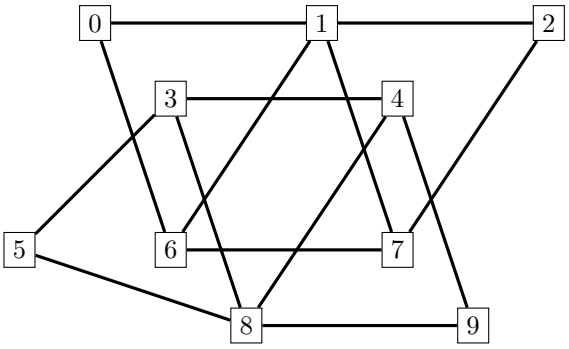
Graphe G_1



Graphe G_2



Graphe G_3



Graphe G_4 (un arbre)