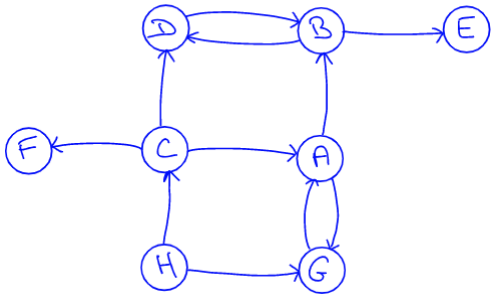
Graphes

Exercice 1 - Représentations d'un graphe

1. On définit le graphe G_1 par :



   Représenter G_1 de toutes les autres façons possibles.

2. Pour chacun des graphes suivants, donner la représentation schématique.

   (a) G_2 défini par :

$$\begin{cases} G_2 \text{ n'est pas orienté} \\ G_2 \text{ a pour liste de sommets } S_2 = \{0,1,2,3,4\} \\ G_2 \text{ a pour liste d'arêtes } A_2 = \{01, 02, 12, 23\} \end{cases}$$

   (b) G_3 défini par :

$$L3 = [\,[1]\,,\,[2,4]\,,\,[0,3]\,,\,[0,1]\,,\,[0,2]\,]$$

   (c) G_4 défini par :

$$M_4 = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

   (d) G_5 défini par :

$$D5 = \{0 : [1,2,3,4], 1 : [0,2,3,4], 2 : [0,1,3,4], 3 : [0,1,2,4], 4 : [0,1,2,3]\}$$

Exercice 2 - Fonctions de conversion

Dans cet exercice, on manipule uniquement des graphes dont la liste des sommets est une liste d'entiers naturels consécutifs commençant par 0.

1. Ecrire une fonction Python qui prend en argument un graphe donné par sa liste d'adjacence et qui renvoie :
   (a) La matrice d'adjacence associée.
   (b) Le dictionnaire associé.
2. Ecrire une fonction Python qui prend en argument un graphe donné par sa matrice d'adjacence sous forme de tableau numpy et qui renvoie :
   (a) La liste d'adjacence associée.
   (b) Le dictionnaire associé.
3. Ecrire une fonction Python qui prend en argument un graphe donné par un dictionnaire et qui renvoie :
   (a) La liste d'adjacence associée.
   (b) La matrice d'adjacence associée.

Exercice 3 - Fonctions de base sur les graphes

Dans cet exercice, on manipule uniquement des graphes dont la liste des sommets est une liste d'entiers naturels consécutifs en commençant par 0.

Dans cet exercice, les fonctions Pythons prendront en argument un graphe, représenté par la forme qui vous semblera la plus adaptée.

1. Ecrire une fonction **sommets_atteignables** qui prend en argument un graphe et qui renvoie la liste des sommets atteignable à partir du sommet `i` en parcourant une seule arête.
2. (a) Ecrire une fonction **voisins** qui prend en argument un graphe et deux sommets `i` et `j` et qui renvoie **True** si `i` et `j` sont voisins, c'est-à-dire s'il existe une arête qui relie `i` et `j`.
   (b) Ecrire une fonction **liste_voisins** qui prend en argument un graphe et un sommet `i` et qui renvoie la liste des voisins de `i`, c'est-à-dire la liste des sommets reliés à `i` par une arête.
   (c) Ecrire une fonction **nb_voisins** qui prend en argument un graphe et un sommet `i` et qui renvoie le nombre de voisins de `i`.
3. Ecrire une fonction **nb_boucles** qui prend en argument un graphe et qui renvoie le nombre de boucles.
4. (a) Ecrire une fonction **symetrique** qui prend en argument un matrice carrée et qui renvoie **True** si cette matrice est symétrique et **False** sinon.
   (b) Ecrire une fonction **oriente** qui prend en argument un graphe et qui renvoie **True** si ce graphe est orienté et **False** sinon.

Exercice 4

Un groupe de chercheurs s'intéresse au comportement d'une souris lorsqu'elle est placée dans un labyrinthe.

Ce labyrinthe comporte trois croisements numérotés 1, 2 et 3 et de nombreux chemins. Il est schématisé sur la figure ci-dessous. Les flèches indiquent les déplacements possibles et les nombres portés le long des flèches indiquent les probabilités respectives que la souris choisisse ces déplacements.

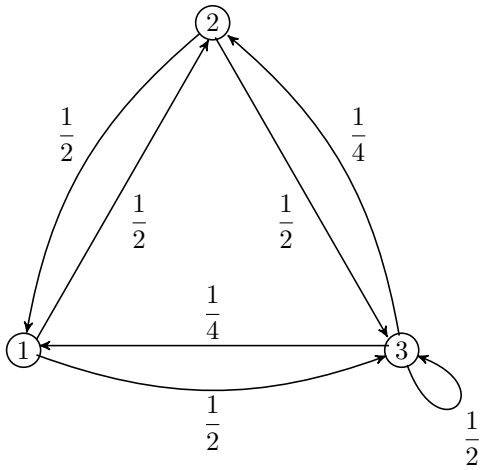


Schéma du labyrinthe.

A l'instant initial $n = 0$, on place la souris sur le croisement numéro 1.
Pour tout $n \in \mathbb{N}$, on définit les événements :

- A_n : la souris se trouve au croisement 1 après le le n$^{\text{ème}}$ déplacement
- B_n : la souris se trouve au croisement 2 après le le n$^{\text{ème}}$ déplacement
- C_n : la souris se trouve au croisement 3 après le le n$^{\text{ème}}$ déplacement

Pour tout $n \in \mathbb{N}$, on note aussi Y_n le vecteur colonne défini par : $Y_n = \begin{pmatrix} P(A_n) \\ P(B_n) \\ P(C_n) \end{pmatrix}$.

Ainsi, $Y_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ et $Y_1 = \begin{pmatrix} 0 \\ 1/2 \\ 1/2 \end{pmatrix}$

1. (a) On note A la matrice de $\mathcal{M}_3(\mathbb{R})$ définie par : $A = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$.

   Montrer que, pour tout entier naturel n, $Y_{n+1} = AY_n$.

   (b) En déduire que, pour tout entier naturel n, $Y_n = A^n Y_0$.

   (c) Ecrire une fonction Python prenant en argument un entier naturel n et qui renvoie la matrice Y_n
2. Ecrire une ou des fonctions Python permettant de simuler le déplacement de la souris et de renvoyer sa position après n déplacements.

Exercice 5 - Parcours

Dans cet exercice, on manipule uniquement des graphes dont la liste des sommets est une liste d'entiers naturels consécutifs en commençant par 0.

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.