INFORMATIQUE PSI P2B

Année 2023/24

TP n° 5

Graphes

Table des matières

Vincent Rohart

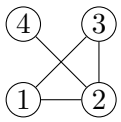
http://vrohart.e-monsite.com/

1 Matrice d'adjacence et connexité

Dans cette section les graphes n'ont qu'un nombre fini de sommets et ces derniers sont les entiers naturels : $\mathrm{V} = \{1, 2, \ldots, n\}$.

1. Écrire une fonction de signature `mat(V,E)` qui prend en argument un graphe `[V,E]` et qui renvoie sa matrice d'adjacence.

2. Écrire une fonction `dico` renvoyant le dictionnaire des adjacences, c'est-à-dire un dictionnaire dont les clés sont les sommets et les valeurs sont des listes contenant les sommets adjacents. Par exemple, le dictionnaire des adjacences du graphe suivant



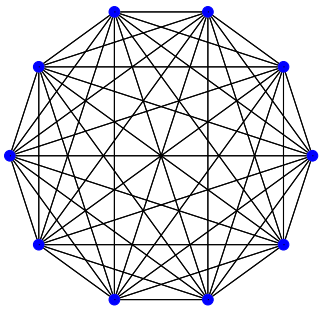
est `{"1":["2", "3"] , "2": ["1", "3", "4"] , "3":["1", "2"] , "4": ["2"]}`

3. Inversement, construire une fonction `graphe` qui prend un dictionnaire des adjacences et qui renvoie l'ensemble `E` des arêtes.

4. Construire une fonction de signature `deg(V,E,x)` qui prend en argument un graphe `[V,E]`, un sommet `x` et qui renvoie le degré $d(x)$.

5. Créer une fonction de signature `complet(V,E)` permettant de savoir si un graphe est complet ou non.

6. Proposer une fonction prenant en argument un entier $n \geqslant 2$ et renvoyant une représentation du graphe complet K_n. Pour $n = 10$, vous devrez obtenir la figure suivante.



7. Soit M la matrice d'adjacence d'un graphe $\mathscr{G}$: si $(i, j) \in [\![0, n-1]\!]$, $\mathrm{M}[i, j]$ vaut donc 1 si $\{i, j\}$ est une arête, et 0 sinon.

   Expliquer pourquoi $\mathrm{M}^2[i, j]$ vaut le nombre de chemins de longueur 2 reliant i et j. En déduire que $\mathscr{G}$ est un graphe connexe si et seulement si $\mathrm{I}_n + \mathrm{M} + \ldots + \mathrm{M}^{n-1}$ a des coefficients strictement positifs.

   En déduire enfin une fonction de signature `is_connexe(V,E)` renvoyant `True` si le graphe `[V,E]` est connexe, et `False` sinon.

   Bibliothèques utiles : `import numpy.linalg as al` et `numpy as np`.

   — La matrice $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ peut s'écrire `np.mat('[1,2,3 ; 4,5 ,6]')`.

   — La commande `al.matrix_power(M,n)` calcule M^n, si M est une matrice carrée et $n \in \mathbb{N}$.

2 Parcours en largeur d'un graphe

On rappelle que `from collections import deque` permet d'utiliser le type `deque`, simulant des files (entre autres).

Si $\mathcal{G} = (\mathrm{V}, \mathrm{E})$ est un graphe non orienté, parcourir $\mathcal{G}$ c'est lister tous ses sommets. Pour ce faire, il y a plusieurs méthodes, dont le *parcours en largeur*. Cette méthode consiste en l'algorithme suivant.

— `F = deque([]) # file initialement vide`
— On enfile un sommet (si bien sûr V n'est pas vide !)
— Tant que F n'est pas vide, faire :
  — `S = F[0] # c'est la tête de la file F`
  — On enfile les voisins de S qui ne sont pas déjà dans F et qui n'ont pas déjà été défilés.
  — On défile F (on enlève sa tête).

Écrire une fonction de signature `par_larg(V,E,s)` renvoyant la liste des sommets d'un graphe non orienté (V, E) par un parcours en largeur en partant d'un sommet donné s. Expliquer pourquoi les files sont bien adaptées au parcours en largeur, contrairement aux piles. Tester votre fonction sur le graphe de la page précédente.

3 Parcours en profondeur d'un graphe

On rappelle que le type `list` de Python se comporte comme une pile.

Si $\mathcal{G} = (\mathrm{V}, \mathrm{E})$ est un graphe non orienté, parcourir $\mathcal{G}$ en profondeur c'est lister tous ses sommets grâce à l'algorithme suivant.

— `P = [] # pile initialement vide`
— On empile un sommet (si bien sûr V n'est pas vide !)
— Tant que P n'est pas vide, faire :
  — `S = P.pop() # on enlève le dernier élément de la pile P.`
  — On empile les voisins de S qui ne sont pas déjà dans P et qui n'ont pas déjà été dépilés.

On voit que contrairement au parcours en largeur, on part d'un sommet en « allant jusqu'au bout » des chemins issus de ce sommet.

Écrire une fonction de signature `par_prof(V,E,s)` renvoyant la liste des sommets d'un graphe non orienté (V, E) par un parcours en profondeur en partant d'un sommet donné s. Tester votre fonction sur le graphe de la page précédente.