Graphes (Cours)

S4 - Arbres et graphes

1. Notion de graphe

1.1. Définition

Définition

Un **graphe non orienté** est constitué :

- d'un ensemble fini de **sommets** (représentés par des points) ;
- d'un ensemble d'**arêtes** (représentées par des traits) qui relient des sommets entre eux.

La figure ci-dessous représente un graphe dont les sommets sont numérotés de 0 à 6.

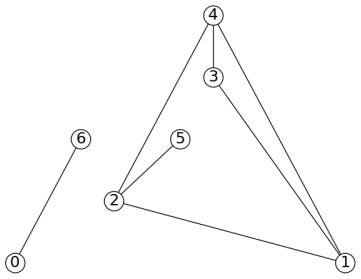


Figure 1: Graphe non orienté

Définition

Un **graphe orienté** est constitué :

- d'un ensemble fini de **sommets** (représentés par des points) ;
- d'un ensemble d'**arcs** (représentés par des flèches) qui relient des sommets entre eux.

La figure ci-dessous représente un graphe orienté dont les sommets sont numérotés de 0 à 6.

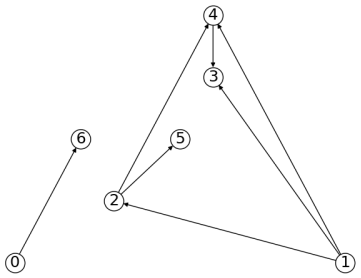


Figure 2: Graphe orienté

Un peu de vocabulaire :

- deux sommets reliés par une arête (ou un arc) sont dits **adjacents**, ou **voisins** ;
- un sommet est dit **isolé** s'il n'est relié à aucun autre sommet ;
- l'**ordre** d'un graphe est le nombre de ses sommets ;
- le **degré** d'un sommet S, noté $deg(S)$, est le nombre d'arêtes (ou d'arcs) qui le relient à d'autres sommets ;

1.2. Lien avec les arbres

La structure d'arbre déjà rencontrée dans le cours est un cas particulier de graphe non orienté.

Définition

Un **arbre** est un graphe non orienté qui satisfait les conditions suivantes :

- il est **connexe** ;
- il n'a pas de cycle.

Explications : *connexe* signifie que tous les sommets sont reliés entre eux par un chemin (il n'y a pas de points isolés). *pas de cycle* signifie que l'on ne peut pas revenir au même sommet en passant par le même chemin.

En particulier le graphe ci-dessus (Figure 1) n'est pas un arbre, car il n'est pas connexe (le sommet 6 n'est pas relié au sommet 1) et il contient un cycle (le chemin 1-2-4-1).

1.3. Exemples

Voici quelques situations pouvant être modélisées par un graphe :

- un réseau routier ;
- un réseau de télécommunications ;
- un réseau social ;
- un réseau électrique ;

2. Implémentations d'un graphe

L'objectif de cette section est de définir une structure de données en Python permettant de représenter un graphe.

Les opérations suivantes doivent être possibles (cf. FORTIER (2022)):

- créer un graphe vide ;
- ajouter un sommet ;
- ajouter une arête (ou un arc) ;
- supprimer un sommet ;
- supprimer une arête (ou un arc) ;
- vérifier si deux sommets sont adjacents ;
- connaître la liste des sommets adjacents à un sommet donné.

Nous étudions deux implémentations : par matrice d'adjacence et par liste d'adjacence.

2.1. Matrice d'adjacence

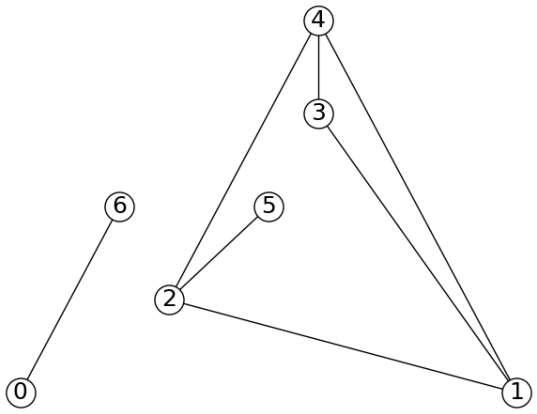
Rappel : une **matrice** est un tableau à deux dimensions, où chaque élément est identifié par un couple de coordonnées (ligne, colonne). En Python, on peut représenter une matrice par une liste de listes.

Définition

La **matrice d'adjacence** d'un graphe G (orienté ou non) est une matrice carrée A de taille n telle que :

- $A_{i,j} = 1$ si les sommets i et j sont adjacents ;
- $A_{i,j} = 0$ si les sommets i et j ne sont pas adjacents.

Exemple : la matrice d'adjacence du graphe non orienté de la figure ci-dessus (Figure 1) est la suivante :



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

Figure 3: Graphe Fig.1

Remarques : dans le cas d'un graphe non orienté, la matrice d'adjacence est symétrique par rapport à la diagonale principale. Dans le cas d'un graphe orienté, la matrice d'adjacence n'est pas forcément symétrique.

TP Python

TP Python : implémentation des graphes par matrice d'adjacence

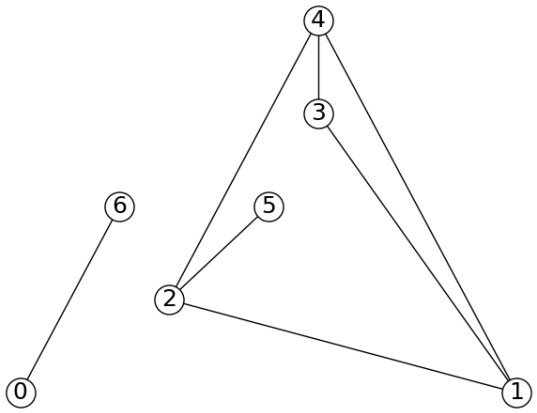
2.2. Liste d'adjacence

Définition : liste d'adjacence

La **liste d'adjacence** d'un graphe G (orienté ou non) dont les sommets sont les entiers compris entre 0 et $n-1$ est une liste L de taille n telle que :

- $L[i]$ est la liste des sommets adjacents au sommet i.

Exemple : la liste d'adjacence du graphe non orienté de la figure ci-dessus (Figure 1) est la suivante :



0	[6]
1	[2, 3, 4]
2	[1, 4, 5]
3	[1, 4]
4	[1, 2, 3]
5	[2]
6	[0]

Figure 4: Graphe Fig.1

Dans le cas d'un graphe comportant peu d'arêtes, la liste d'adjacence occupe moins de mémoire que la matrice d'adjacence. Mais certaines opérations, comme la vérification de l'adjacence de deux sommets ou la suppression d'une arête, sont plus coûteuses en temps d'exécution.

TP Python

Dans le TP ci-dessous, vous devez implémenter la structure de graphe par liste d'adjacence. Vous devez également implémenter les fonctions permettant de passer d'une représentation à l'autre.
TP Python : implémentation des graphes par liste d'adjacence

Sources utilisées pour la rédaction de ce chapitre

FORTIER, Quentin. 2022. « Informatique commune en 1ère année en CPGE ». *CPGE-ITC*. https://cpge-itc.github.io/itc1/ .