# Sujet n°6

## Algorithmique & Programmation

### Les graphes

## 1 Introduction

Un graphe non orienté est la donnée d'un ensemble de sommets $V$ et d'un ensemble d'arêtes $E$ liant deux sommets entre eux. Deux sommets liés par une arête sont dits voisins. Un exemple de graphe $G$ non orienté est donné dans la figure 1 avec $V = \{0, 1, 2, 3, 4, 5, 6, 7\}$ et $E = \{\{0, 1\}, \{0, 2\}, \{0, 3\}, \{1, 2\}, \{2, 3\}, \{1, 4\}, \{2, 4\}, \{3, 5\}, \{4, 5\}, \{4, 6\}, \{5, 7\}, \{6, 7\}\}$.

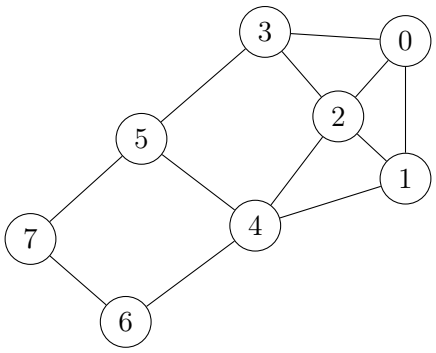
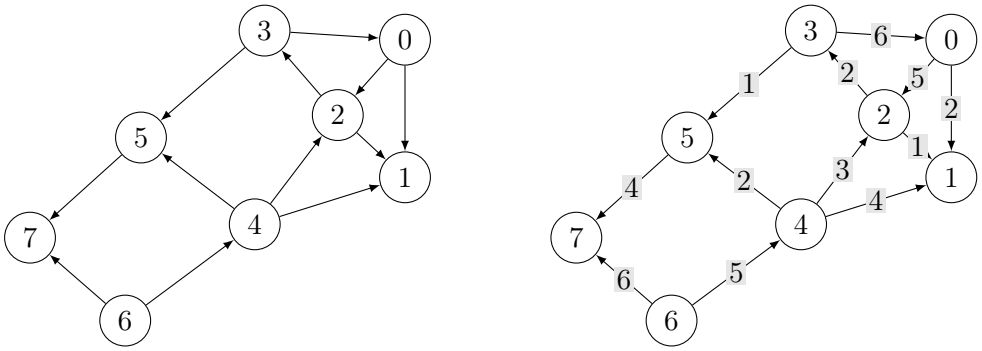



FIGURE 1 – Exemple de graphe non orienté.

Un graphe orienté est la donnée d'un ensemble de sommets $V$ et d'un ensemble d'arcs $E \subseteq V \times V$ liant un sommet à un autre. Si $(u, v)$ est un arc, $u$ est un *prédécesseur* de $v$, et $v$ est un *successeur* de $u$. La figure 2a montre un exemple de graphe orienté $\overrightarrow{G}$ de $G$. Un graphe pondéré (orienté ou non) est la donnée d'un ensemble de sommets $V$ et d'un ensemble d'arcs $E \subseteq V \times V$ liant un sommet à un autre et pondéré par une valeur. La figure 2b montre un exemple de graphe orienté pondéré $\overrightarrow{G}$ de $G$.




(a) Orienté. (b) Orienté pondéré.

FIGURE 2 – Exemples de graphes

## 2 Exercices

**Exercice 1** : *Structure de donnée de graphe*

En informatique, il existe plusieurs implémentations possibles des graphes orientés qui dépendent de la structure de donnée choisie pour coder la définition des graphes. On peut implémenter les graphes à l'aide d'une liste d'adjacence, à l'aide d'une matrice d'adjacence, ou à l'aide d'une matrice d'incidence.

On rappelle que la matrice d'adjacence $Adj$ d'un graphe orienté $G = (V, E)$ est une matrice carrée $|V| \times |V|$ telle que :

$$Adj_{i,j} = \begin{cases} 1 & \text{si } (i,j) \in E \\ 0 & \text{sinon} \end{cases}$$

On définit de la même manière la matrice d'adjacence d'un graphe non orienté. Notez que celle-ci est alors symétrique.

On rappelle également que la matrice d'incidence $Inc$ d'un graphe orienté $G = (V, E)$ est une matrice $|V| \times |E|$, telle que :

$$Inc_{i,j} = \begin{cases} -1 & \text{si l'arc } e_j \text{ sort du sommet } v_i \\ 1 & \text{si l'arc } e_j \text{ entre dans le sommet } v_i \\ 0 & \text{sinon} \end{cases}$$

Si le graphe est non orienté le calcul des coefficients de la matrices sont :

$$Inc_{i,j} = \begin{cases} 1 & \text{si le sommet } v_i \text{ est une extrémité de l'arête } e_j \\ 2 & \text{si l'arête } e_j \text{ est une boucle sur } v_i \\ 0 & \text{sinon} \end{cases}$$

Pour distinguer les deux définitions, on parle de matrice d'incidence orientée et de matrice d'incidence non orientée.

Les listes d'adjacence d'un graphe non orienté (resp. orienté) $G = (V, E)$ correspondent à $|V|$ listes des voisins (resp. successeurs) de chaque sommet.

Question 1. Écrire une fonction qui prend la matrice d'adjacence d'un graphe en paramètre et dit s'il peut ou non s'agir d'un graphe non orienté.

Question 2. Établir une structure de donnée permettant de stocker les listes d'adjacence d'un graphe.

Question 3. Écrire une fonction qui prend les listes d'adjacence d'un graphe et dit s'il peut ou non s'agir d'un graphe non orienté.

Question 4. Modifier la structure de donnée conçue dans la question 2 afin de stocker le caractère orienté ou non du graphe.

Question 5. Écrire une fonction qui prend la matrice d'adjacence d'un graphe et retourne le même graphe sous forme de listes d'adjacence. Écrire la transformation inverse : listes d'adjacence vers matrice d'adjacence.

Question 6. Écrire une fonction qui prend les listes d'adjacence d'un graphe et retourne le même graphe sous forme matrice d'incidence.

Question 7. Enfin, intégrez les fonctions développées précédemment afin de créer un module permettant de manipuler des graphes orientés et des graphes non orientés. Ce module doit permettre à l'utilisateur

— d'ajouter et supprimer une arête,

— de donner la liste d'adjacence d'un sommet,

— de récupérer le nombre de sommets d'un graphe,
— d'obtenir le nombre d'arêtes d'un graphe,
— de transformer un graphe orienté en un graphe non orienté.

**Exercice 2** : *Génération d'un graphe*

Dans les questions suivantes, vous utiliserez une représentation de graphe sous la forme de liste d'adjacences.

QUESTION 1. Écrire une fonction qui prend un graphe non orienté et qui renvoie une orientation aléatoire du graphe.

QUESTION 2. Écrire une fonction qui prend un entier $n$ et un réel $0 \leq p \leq 1$ et qui renvoie un graphe aléatoire à $n$ sommets et ayant pour tout couple de sommets $u$, $v$, l'arc $(u, v)$ avec probabilité $p$.

**Exercice 3** : *Parcours d'un graphe*

Considérons un graphe $G = (V, E)$, la bordure $\Gamma(T)$ d'une partie $T$ de $V$ est le sous-ensemble des sommets de $V \setminus T$ qui sont les extrémités d'un arc dont l'origine est dans $T$.

Un parcours de $G$ depuis un sommet $v$ est une liste des sommets $L$ de $G$ telle que :
— chaque sommet de $G$ apparaît une fois et une seule dans $L$,
— chaque sommet de $L$ (sauf le premier) appartient à la bordure du sous-ensemble des sommets placés avant lui dans $L$, si cette bordure est non vide.

Il y a essentiellement deux paradigmes pour parcourir un graphe : en largeur et en profondeur. Le parcours en largeur consiste, à partir d'un sommet, à explorer tous les voisins, puis à continuer le parcours à partir de chaque voisin. À l'inverse, le parcours en profondeur explore chaque branche les unes après les autres.

Par exemple, à partir du graphe représenté dans la figure 3 et de son sommet 0, le parcours en largeur donne successivement les sommets 0, 1, 2, 3, 4, 5, 6 et 7 ; tandis que le parcours en profondeur donne successivement les sommets 0, 1, 5, 7, 2, 3, 4 et 6.

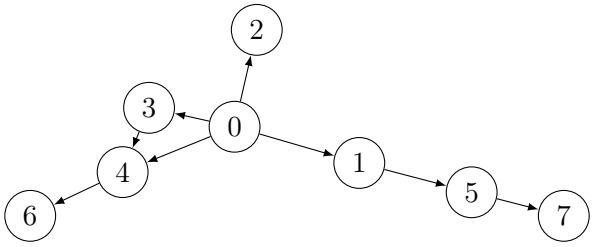



FIGURE 3 – Parcours d'un graphe.

Dans les questions suivantes, vous utiliserez une représentation de graphe sous la forme de liste d'adjacences.

QUESTION 1. Appliquer l'algorithme de parcours en largeur sur un graphe.

QUESTION 2. Appliquer l'algorithme de parcours en profondeur sur un graphe.

**Application au calcul des composantes connexes dans un graphe non-orienté**

On se place dans le cas des graphes non-orientés. Un sous-graphe $G'$ de $G$ est un graphe $(V', E')$ tel que $V'$ est un sous-ensemble de $V$ et $E'$ l'ensemble des arêtes de $G$ ayant ses deux extrémités dans $V'$.

Une composante connexe de $G$ est un sous-graphe $G'$ de $G$ tels que pour tous sommets $u$ et $v$ de $G'$ il existe un chemin de $u$ a $v$ dans $G'$, $G'$ étant maximal pour cette propriété. Une composante connexe de $G$ est donc entièrement décrite par la liste de ses sommets.

Par exemple, le graphe présenté dans la figure 4 se compose de 2 composantes connexes.

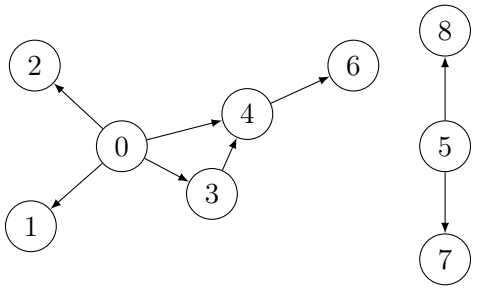



FIGURE 4 – Graphe composé de deux composantes connexes.

QUESTION 3. Écrire une fonction, qui étant donné un graphe non-orienté, retourne la liste de ses composantes connexes.

**Exercice 4** : *Algorithme de Kruskal*

L'algorithme de Kruskal est un algorithme de recherche d'arbre recouvrant de poids minimum dans un graphe connexe valué et non-orienté.

QUESTION 1. Modifier le module de gestion des graphes pour qu'il gère les graphes non orientés avec arêtes pondérées.

QUESTION 2. Déterminer un arbre recouvrant d'un graphe donné, de poids minimum, depuis un sommet donné en utilisant l'algorithme de Kruskal.

**Exercice 5** : *Calculs de plus courts chemins dans un graphe*

QUESTION 1. En quoi consiste l'algorithme de Dijkstra ?

QUESTION 2. Rapellez les condition d'utilisation de l'algorithme.

QUESTION 3. Appliquez l'algorithme de Dijkstra pour determiner la distance (longueur du plus court chemin) du sommet $v$ donné à chacun des sommets d'un graphe $G$ donné.