Manipuler des graphes

1 Représenter un graphe

1.1 Rappel de théorie des graphes

Definition 1 *(Intuitive) Un graphe orienté G est un schéma constitué par un ensemble de points et par un ensemble de flèches reliant chacune deux points. Les points sont appelés les sommets du graphe. L'ensemble des sommets est noté V. Les flèches sont appelées les arcs du graphe. L'ensemble des arcs est noté A. Soient $x, y \in V$, on a $(x, y) \in A$ si et seulement si il existe un arc avec pour origine x et pour extrémité y.*

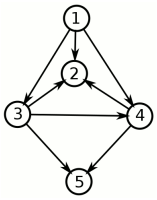

FIGURE 1 – Un graphe orienté

Dans la Figure 1, on observe l'ensemble des sommets $V = (1, 2, 3, 4, 5)$
et l'ensemble des arcs $A = ((1, 2), (1, 3), (1, 4), (3, 2), (3, 4), (3, 5), (4, 2), (4, 5))$

Definition 2 *On appelle degré entrant (resp. sortant) d'un sommet $v \in V$ le nombre d'arcs dont l'extrémité (resp. l'origine) est v. On le note $d^-(v)$ (resp. $d^+(v)$).*

Dans la Figure 1, on a $d^+(1) = 3$, $d^-(1) = 0$, $d^+(5) = 0$, $d^-(5) = 2$, $d^+(3) = 3$, $d^-(3) = 1$, etc

1.2 La liste d'adjacence

Definition 3 *On dit que $v \in V$ est un voisin sortant (resp. entrant) de $u \in V$ s'il existe un arc (u, v) (resp. $(v, u)) \in A$.*

Definition 4 *Pour $v \in V$, nous définissons $N^+(v)$ (resp. $N^-(v)$) comme l'ensemble des voisins* sortants *(resp.* entrants*) de v. L'ensemble des voisins de v est noté $N(v) = N^+(v) \cup N^-(v)$.*

Dans la Figure 1, $N^+(1) = (2, 3, 4)$ et $N^-(1) = \emptyset$, $N^+(5) = \emptyset$ et $N^-(5) = (3, 4)$, $N^+(3) = (2, 4, 5)$ et $N^-(3) = (1)$.

On peut choisir de représenter les graphes sous forme de liste d'adjacence. La liste d'adjacence est une liste qui donne pour chaque sommet $v1, \ldots, vn \in V$, la liste de ses voisins sortants $N^+(v1), \ldots, N^+(vn)$ sous la forme $((v1, N^+(v1)), \ldots, (vn, N^+(vn)))$. La liste d'adjacence du graphe représenté dans la Figure 1 est :

$$((1, (2, 3, 4)), (2, \emptyset), (3, (2, 5)), (4, (2, 5)), (5, \emptyset))$$

2 Travail

2.1 Stocker le graphe

Le stockage du graphe se fera sous forme de liste d'adjacence :
- La classe *Sommet* contient l'identifiant du noeud ainsi que ses voisins. D'autres informations peuvent y être stockées.
- La classe *Graphe* permet de stocker l'ensemble des noeuds du graphe.

2.2 Explorer le graphe

Pour commencer à étudier les propriétés d'un graphe, il s'agit tout d'abord d'être capable d'en explorer le contenu facilement et de disposer de quelques mesures simples mais très utilisées :

- Connaître le nombre de sommets, d'arcs
- Obtenir l'ensemble des sommets et leur degré
- Obtenir l'ensemble des sommets triés par degré sortant
- Obtenir l'ensemble des arcs

Une interface devra être proposée dans votre application pour qu'un utilisateur puisse appeler toutes ces fonctionnalités, en visualiser les résultats et ainsi commencer à fouiller les données de son graphe.

2.3 Parcourir un graphe

Les parcours en largeur (Breadth) et profondeur (Depth) font partie des algorithmes de base en théorie des graphes. Ils permettent par exemple de détecter les différentes composantes connexes d'un graphe, à savoir les ensembles de sommets connectés d'un graphe.

Algorithm 1: Parcours d'un graphe

```
1  Initialisation
2  S = ∅ ⊆ V
3  Q = ∅ ⊆ V
4  v_0 ∈ V la source de l'algorithme
5  Ajouter v_0 à Q
6  tant que Q ≠ ∅ faire
7      Sortir un élément v de Q
8      si v ∋ S alors
9          Ajouter v à S
10         pour tous les w ∈ N^(v)tqw ∋ S faire
11             Ajouter w à Q
12 S est une composante connexe du graphe G = (V, E)
```

Si Q est utilisé comme une file (FIFO), alors le parcours est un parcours en largeur. En revanche, si Q est utilisé comme une pile (LIFO), alors le parcours effectué est un parcours en profondeur.

Pour être plus souple, on implémentera ce programme de façon générique. On définira ainsi une interface *SDD* contenant les méthodes *push(Vertex v)* et *Vertex pop()*. Les classes Pile et File définies dans le TDm2 devront implémenter cette interface. L'algorithme de parcours aura pour paramètre un objet de type *SDD*. Ainsi, cela permettra de passer d'un parcours d'un type à un autre en toute transparence.