Etude guidée du type abstrait graphe orienté

Structures de données — FMIN 220

30 mars 2015

1 Déroulement du travail

Vous travaillerez par groupes de 4 ou 5 en vous répartissant les rôles (vous pouvez jouer plusieurs rôles) :

- de chef de projet (organise l'ordre des tâches et la répartition du travail, recherche sur internet des idées pour l'implémentation et les schémas d'algorithmes, suit l'avancement du travail).
- de concepteur (anime la réflexion sur la spécification, sur la forme des structures de données choisies et l'organisation du code, remet à jour les spécifications lorsque des difficultés sont rencontrées et les diffuse).
- d'intégrateur (reçoit les parties développées par les autres membres du groupe et teste leur intégration au fur et à mesure de l'avancement). Il prépare à l'avance les programmes de test (unitaires) pour chaque petite partie et les fournit aux développeurs. Il prépare aussi un programme de test montrant que toutes les parties fonctionnent ensemble.
- de développeur (reçoit une tâche, une spécification et un programme de test unitaire, développe une interface, une classe ou une méthode et la retourne à l'intégrateur après test).

La dernière partie du TD sera consacrée à une restitution, lors de laquelle vous présenterez votre organisation, vos choix et votre code aux autres groupes.

2 Notion de graphe orienté

- Un graphe orienté G = (V, E) est un couple composé d'un ensemble de noeuds / sommets V et d'un ensemble d'arcs $E \subseteq V \times V$. Cet ensemble d'arcs représente donc une relation entre les éléments de V. Un arc est un couple (s, t).
- Nous considérons ici que les sommets et les arcs sont étiquetés.

3 Type abstrait `graphe orienté`

Vous chercherez dans un premier temps à produire une description du type abstrait avec les premières indications suivantes sur les opérations attendues. Les sommets sont numérotés par des entiers. Dans un premier temps, les étiquettes peuvent être de simples chaînes de caractères (puis vous pourrez en faire des paramètres de généricité) et il est même recommandé de commencer l'exercice sans traiter les étiquettes.

- créer un graphe contenant un nombre de sommets donné
- associer une étiquette à un sommet
- ajouter un arc entre deux sommets
- associer une étiquette à un arc
- obtenir l'étiquette d'un sommet en donnant son numéro

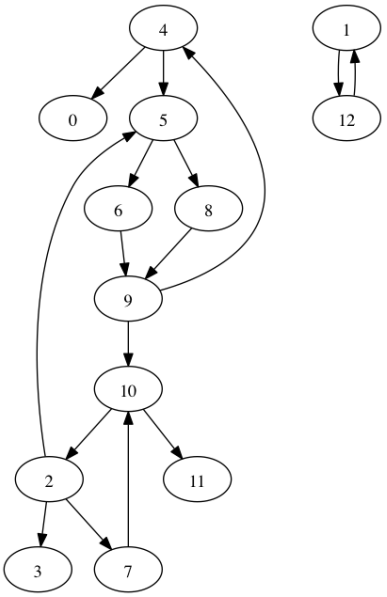

FIGURE 1 – Graphe dessiné par Graphviz

— obtenir l'étiquette d'un arc en donnant les numéros de ses deux extrémités
— lire le graphe dans un fichier
— sauvegarder le graphe dans un fichier
Déterminer les pré-conditions des opérations.

4 Implémentation du type graphe

Pour stocker les arcs, vous choisirez l'une des deux implémentations proposées, par matrice d'adjacence ou par liste de successeurs.

4.1 Implémentation par une matrice d'adjacence

La matrice d'adjacence est une matrice de booléens M, carrée et indexée par les indices des sommets. $M[i][j]$ contient la valeur vrai si et seulement s'il y a un arc entre i et j. C'est la représentation à préférer lorsque le nombre de sommets est connu à l'avance et lorsque le graphe est dense (il contient beaucoup d'arcs).

4.2 Implémentation par des listes de successeurs

Dans cette implémentation, on associe à chaque sommet la liste des sommets qui sont ses voisins par un arc sortant. On utilise cette représentation lorsque le graphe n'est pas très dense.

5 Lecture et sauvegarde dans un fichier texte

Nous proposons d'utiliser le format `dot` qui vous est présenté ci-dessous, pour la lecture et l'écriture dans un fichier. Il peut être complété en ajoutant ultérieurement des étiquettes aux sommets et aux arcs (en ajoutant `[ label = "le label choisi" ]` avant le ';').

Si votre fichier s'appelle `graphesimple.dot`, vous pouvez produire une représentation graphique dans un fichier `.png` grâce à la commande :

`dot -Tpng graphesimple.dot -o graphesimple.png`.
Ce peut être fait aussi pour produire d'autres formats, tels que `pdf`.

La figure 1 vous montre le résultat pour le fichier montré ci-dessous. Dans la dernière section vous trouverez des embryons de programmes pour lire/écrire dans un fichier texte inspiré de ce fichier.

```
digraph graphe_simple {
0 ;
1 ;
2 ;
3 ;
4 ;
5 ;
6 ;
7 ;
8 ;
9  ;
10 ;
11 ;
12 ;
2 -> 3 ;
2 -> 5 ;
2 -> 7 ;
4 -> 5 ;
5 -> 8 ;
8 -> 9 ;
5 -> 6 ;
6 -> 9 ;
9 -> 10 ;
10 -> 2 ;
7 -> 10 ;
10 -> 11 ;
9 -> 4 ;
4 -> 0 ;
1 -> 12 ;
12 -> 1 ;
}
```

6 Algorithmes

Vous pouvez à présent écrire d'autres opérations sur vos graphes :
— calculer la densité du graphe (nombre d'arcs sur nombre d'arcs possibles)
— calculer le nombre moyen de voisins
— parcours en profondeur ou en largeur à partir d'un sommet
— parcours en profondeur ou en largeur de tout le graphe
— savoir si le graphe est connexe (alternativement trouver ses composantes connexes)
— savoir si deux sommets sont reliés

7 Exemple de code utilisant des fichiers (lecture/écriture)

```
public class Graphe {

public void lireFichier(String nomFichier) throws IOException{
     java.io.File fichier = new java.io.File(nomFichier);
     Scanner lecteur = new Scanner(fichier);
     System.out.println("nom du graphe "+lecteur.next());// passer l'entête
     System.out.println("nom du graphe "+lecteur.next());// passer le nom du graphe
     System.out.println(" { "+lecteur.next()); // passer {
     // lire les sommets, lire deux zones
     String numSommet = lecteur.next();
     System.out.println("num sommet "+numSommet);
     String cars = lecteur.next(); // ; ou ->
     System.out.println("cars "+cars);
     int nbSommets = 0;
     while (!cars.equals("->")) {
          numSommet = lecteur.next();
          System.out.println("num sommet "+numSommet);
          cars = lecteur.next(); // ; ou ->
          System.out.println("cars "+cars);
     nbSommets++;
     }
     System.out.println("nombre de sommets = "+nbSommets);
     // le dernier numSommet est l'origine d'un arc -   lire les arcs
     String numSommetDest="";
     while (!numSommet.equals("}")) {
          numSommetDest = lecteur.next();
          System.out.println("num sommet dest "+numSommetDest);
          System.out.println("arc "+numSommet+ " "+ numSommetDest);
          cars = lecteur.next(); // ; ou
          System.out.println("cars "+cars);
          numSommet = lecteur.next();
          System.out.println("num sommet orig "+numSommet);
          if (!numSommet.equals("}")) cars = lecteur.next(); // ->
     }
     lecteur.close();
}

     public void ecrireFichier(String nomFichier) throws IOException{
          BufferedReader fc = new BufferedReader(new InputStreamReader (System.in));
          BufferedWriter ff = new BufferedWriter(new FileWriter (nomFichier));
          System.out.println("On ecrit dans le fichier seulement la partie sommets");
          ff.write("digraph mon_nouveau_graphe {");
          ff.newLine();
          Integer i = 0; // on suppose 12 sommets
          while (i <= 12){
           ff.write(i.toString()+" ; "); // TQ pas chaîne vide
          ff.newLine(); i=i+1;
          }
          // a compléter par l'écriture des arcs
          ff.write("} ");  ff.close();
}

     public static void main(String[] arg) throws IOException{
          Graphe g = new Graphe();
          g.lireFichier("graphesimple.dot"); // fichier dans le dossier du projet
          g.ecrireFichier("graphesimplesommets.dot"); // fichier dans le dossier du projet
     }
}
```