IR2 - Algorithmique des graphes

Fiche 3 - Connexité

L'objectif de ce TP est d'implanter des algorithmes pour calculer les composantes connexes (resp. fortement connexes) d'un graphe (resp. graphe orienté). Le langage de programmation est le Java.

Dans un graphe G non orienté, une *composante connexe* est un sous-graphe induit maximal H de G, tel que pour tout couple $\{x, y\}$ de sommets de H, il existe une chaîne contenant x et y dans H.
Dans un graphe orienté G, une *composante fortement connexe* est un sous-graphe induit maximal H de G tel que pour toute paire $\{x, y\}$ de sommets de H, il existe un chemin de x à y et un chemin de y à x dans H.
Il est ainsi possible de partitionner l'ensemble des sommets d'un graphe (resp. graphe orienté) en fonction de leur appartenance à la même composante connexe (resp. fortement connexe).

Exercice 1. Composantes connexes d'un graphe non orienté
Dans cet exercice, on utilisera la représentation des graphes par liste d'adjacence. Pour distinguer les composantes connexes les unes des autres, chaque sommet du graphe se verra attribuer une couleur ; deux sommets seront dans la même composante connexe si et seulement si ils sont de même couleur.

1. Modifier la structure de graphe représentés par liste d'adjacence pour intégrer le coloriage des sommets.
2. Écrire une méthode qui colorie la composante connexe d'un sommet x avec la couleur c dans un graphe G initialement non coloré, en s'inspirant de l'algorithme récursif suivant :

Algorithm 1 ColorierSommet(G, x, c)

Require: un graphe G, x un sommet de G et c une couleur.
Ensure: colorie la composante connexe de x de la couleur c.

```
1: colorier le sommet x avec la couleur c.
2: for tout sommet y adjacent à x do
3:    if y n'est pas colorié then
4:       ColorierSommet(G, y, c)
5:    end if
6: end for
```

3. Écrire une méthode qui colorie toutes les composantes connexes d'un graphe avec une couleur par composante connexe en s'inspirant de l'algorithme suivant :

Algorithm 2 ColorierComposantesConnexes(G)

Require: un graphe G non colorié.
Ensure: colorie les sommets de G, une couleur par composante connexe.

```
1: while il reste un sommet x non colorié dans G do
2:    c ← une couleur non utilisée.
3:    ColorierSommet(G, x, c)
4: end while
```

Exercice 2. Composantes fortement connexes d'un graphe orienté, première approche
Dans cet exercice, on utilisera la représentation des graphes orientés par liste d'adjacence. Comme dans l'exercice précédent, les composantes fortement connexes sont distinguées par coloriage des sommets du graphe.

1. Modifier la structure de graphes orientés représentés par liste d'adjacence pour intégrer le coloriage des sommets.
2. Écrire une méthode qui, étant donnés un graphe orienté G et un sommet x, retourne la liste des sommets accessibles depuis x dans G.
3. Écrire une méthode qui colorie toutes les composantes fortement connexes d'un graphe orienté avec une couleur par composante connexe en s'inspirant de l'algorithme suivant :

Algorithm 3 ColorierComposantesFortementConnexes(G)

Require: un graphe orienté G non colorié.
Ensure: colorie les sommets de G, une couleur par composante fortement connexe.

```
while il reste un sommet x non colorié dans G do
    c ← une couleur non utilisée.
    colorier x avec la couleur c.
    A ← liste des sommets accessibles depuis x dans G.
    for pour tout sommet y dans A do
        B ← liste des sommets accessibles depuis y dans G.
        if x est dans B then
            colorier y avec la couleur c.
        end if
    end for
end while
```

4. Évaluer la complexité temporelle et spatiale de cet algorithme.

Exercice 3. Composantes fortement connexes d'un graphe orienté, algorithme de Kosaraju
Dans cet exercice, on utilisera la représentation des graphes orientés par matrice d'adjacence.

1. Sachant qu'il va être nécessaire dans par la suite de calculer le transposé d'un graphe, expliquer pourquoi le choix de représentation par matrice d'adjacence est plus judicieux que la représentation par liste d'adjacence. S'appuyer sur la complexité temporelle de cette opération en fonction de la représentation de graphe considérée.
2. Implanter une méthode qui, étant donné un graphe orienté G retourne son transposé.
3. Il est maintenant nécessaire de pouvoir parcourir un graphe orienté en profondeur. La structure doit être enrichie de trois nouveaux attributs :
   (a) une couleur `c(x)` parmi `blanc`, `gris` ou `noir`, selon l'état du sommet x, signifiant respectivement non visité, en cours de traitement et traité ;
   (b) un sommet `p(x)`, qui égal au sommet qui a permi d'atteindre x dans le parcours en profondeur ; cet attribut permet de construire un arbre couvrant du graphe ;
   (c) un tableau de fin de traitement `f(i)` qui permet d'ordonner les sommets du graphe orienté en fonction de leur ordre de fin de traitement. Le sommet `f(0)` est le premier sommet à avoir été traité et `f(n - 1)`, où n est le nombre de sommets du graphe orienté, le dernier.

   Implanter l'algorithme de parcours en profondeur d'un graphe orienté en s'inspirant des algorithmes suivants :

Algorithm 4 ParcoursProfondeur(G)

Require: un graphe orienté G.
Ensure: visite les sommets de G selon un parcours en profondeur.

```
1: for tout sommet x de G do
2:     c(x) ← blanc
3:     p(x) ← null
4: end for
5: t ← 0
6: for tout sommet x de G do
7:     if c(x) = blanc then
8:         Visiter(G, x)
9:     end if
10: end for
```

Algorithm 5 Visiter(G, x)

Require: un graphe orienté G, x un sommet de G.
Ensure: visite les sommets de G depuis le sommet x.

```
1: c(x) ← gris
2: for tout sommet y successeur de x dans G do
3:     if c(y) = blanc then
4:         p(y) ← x
5:         Visiter(G, y)
6:     end if
7: end for
8: c(x) ← noir
9: f(t) ← x
10: t ← t + 1
```

4. Pour calculer les composantes fortement connexes d'un graphe orienté G, on va utiliser l'algorithme suivant. Le résultat est contenu dans le tableau des sommets parents p : deux sommets sont dans une même composante fortement connexe si et seulement si ils sont dans un même arbre.

Algorithm 6 ComposantesConnexes(G)

Require: un graphe orienté G.
Ensure: calcule les composantes fortement connexes de G.

```
1: ParcoursProfondeur(G)
2: T ← graphe transposé de G.
3: ParcoursProfondeur(T) (en traitant les sommets de T par ordre de traitement décroissant indiqué dans f)
```

Implanter cet algorithme.

5. Évaluer sa complexité temporelle et spatiale.
6. Montrer qu'un graphe orienté et son transposé possèdent le même nombre de composantes fortement connexes.
7. Soit un graphe orienté $G = (V, E)$ tel que $x, y \in V$. Exprimer le nombre de composantes fortement connexes du graphe orienté $G' = (V, E \cup \{(x, y), (y, x)\})$ en fonction de celui de G.