Fiche TD 6

Exercice 1 - Composantes fortement connexes d'un graphe orienté

On travaille sur des graphes *orientés*, que l'on suppose représentés par des *listes d'adjacence* : soit S l'ensemble des sommets et A l'ensemble des arcs, on associe à chaque sommet u dans S la liste $Adj[u]$ des éléments v tels qu'il existe un arc de u à v.

1 - **Parcours en profondeur**

Le parcours en profondeur d'un graphe G fait usage des constructions suivantes :

- A chaque sommet du graphe est associée une *couleur* : au début de l'exécution de la procédure, tous les sommets sont blancs. Lorsqu'un sommet est rencontré pour la première fois, il devient gris. On noircit enfin un sommet lorsque l'on est en fin de traitement, et que sa liste d'adjacence a été complètement examinée.
- Chaque sommet est également *daté* par l'algorithme, et ceci deux fois : pour un sommet u, $d[u]$ représente le moment où le sommet a été rencontré pour la première fois, et $f[u]$ indique le moment où l'on a fini d'explorer la liste d'adjacence de u. On se servira par conséquent d'une variable entière "temps" comme compteur évênementiel pour la datation.
- La manière dont le graphe est parcouru déterminera aussi une relation de paternité entre les sommets, et l'on notera $\pi[v] = u$ pour dire que u est le père de v (selon l'exploration qui est faite, c'est à dire pendant le parcours v a été découvert à partir de u).

On définit alors le parcours en profondeur (**PP**) de la manière suivante :

```
PP(S,Adj[],temps) :=
 pour chaque sommet u de S                 Initialisation
   faire couleur[u] ← BLANC
         π[u] ← NIL
 temps← 0
 pour chaque sommet u de S                 Exploration
   faire si couleur[u]=BLANC
            alors Visiter_PP(u)

Visiter_PP(u) :=
 couleur[u] ← GRIS
 d[u] ←temps ←temps+1
 pour chaque v ∈ Adj[u]
   faire si couleur[v]=BLANC
            alors π[v] ← u
                  Visiter_PP(v)
 couleur[u] ← NOIR
f[u] ← temps← temps+1
```

→ Quel est la complexité de PP ? Que peut-on dire de $d[u]$, $f[u]$, $d[v]$ et $f[v]$, pour deux sommets u et v du graphe ?

→ Montrer que v est un descendant de u si et seulement si à l'instant $d[u]$, il existe un *chemin blanc* de u à v.

2 - **Composantes fortement connexes**

On définit la relation binaire R des sommets d'un graphe orienté par : xRy si et seulement s'il existe un chemin orienté de x à y et un chemin orienté de y à x.
→ Montrer que c'est une relation d'équivalence.

Les classes d'équivalence sont appelées *composantes fortement connexes* du graphe orienté. Ce sont les sous-ensembles maximaux du graphe tels qu'il existe un chemin allant d'un sommet u à un sommet v pour tous sommets u et v de l'ensemble.
→ Montrer que lors d'un parcours en profondeur, tous les sommets appartenant à la même composante fortement connexe sont dans le même arbre en profondeur (issu de l'application de PP sur G).

Pour tout sommet u, et pour une exécution d'un parcours en profondeur, on appelle $\phi(u)$ le sommet w tel que $u \leadsto w$ (i.e. il existe un chemin éventuellement nul de u à w) et $f[w]$ est maximal. On appelle $\phi(u)$ l'*aïeul de u*.
→ Montrer que $\phi(u)$ est un ancêtre de u.

Remarque : par conséquent, u et $\phi(u)$ sont dans la même composante fortement connexe, et plus globalement, deux sommets sont dans la même composante fortement connexe si et seulement si ils ont le même aïeul.

3 - On considère l'algorithme suivant :

- Exécuter le parcours en profondeur de G pour calculer f sur l'ensemble des sommets.
- Calculer le graphe ${}^tG$ qui a les mêmes sommets que G mais où le sens des arcs est inversé.
- Exécuter le parcours en profondeur de ${}^tG$ en appelant dans la boucle principale les sommets par f décroissant. Les composantes fortement connexes de G (et aussi de ${}^tG$) sont les arbres ainsi obtenus.

→ Montrer la correction de l'algorithme précédent (par exemple en montrant par récurrence sur le nombre d'arbres déjà calculés qu'une composante fortement connexe, et donc l'ensemble des sommets de même aïeul selon G, est un arbre). Quelle est sa complexité ?

Exercice 2 - 2-SAT

Le problème 2-SAT est défini de la manière suivante : on se pose la question de la satisfiabilité d'une formule booléenne en *Forme Normale Conjonctive* d'ordre 2 (2-FNC). Une telle formule est une *conjonction* de *clauses*, chaque clause étant elle-même constituée d'une *disjonction* de deux *littéraux*. Un littéral dans une formule booléenne est une variable ou sa négation.

Exemple : $(x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$
est une 2-FNC, satisfiable en mettant par exemple x_1 à vrai et x_2 à faux.

1 - **Une condition nécessaire**

En remarquant que chaque clause peut s'écrire de deux manières sous forme d'une implication entre deux littéraux, représenter la formule comme un graphe orienté où les sommets sont des littéraux et les arcs des implications.
→ Donner une condition nécessaire sur le graphe pour que la formule soit satisfiable.

On va montrer qu'il s'agit d'une condition suffisante.

2 - Montrer qu'une valuation v qui satisfait la formule F est constante sur une composante fortement connexe du graphe associé G.

3 - **Graphe des composantes fortement connexes**

On construit un graphe orienté H associé au graphe G de la manière suivante :

- Les sommets de H sont les composantes fortement connexes de G.
- Il y a un arc de X à Y dans H s'il existe un arc d'un sommet de X vers un sommet de Y dans G.

→ Montrer que H est sans circuit.

4 - **Tri topologique**

Les arcs induisent alors un ordre partiel sur les sommets de H et cet ordre peut-être complété en un ordre total. L'algorithme du tri topologique est défini de la manière suivante : le graphe est parcouru en profondeur et les sommets classés par $f[u]$ décroissant en les insérant au fur et à mesure au début d'une liste chaînée.
→ Montrer que l'ordre total ainsi obtenu étend bien l'ordre partiel.
→ Donner la complexité de la construction de H à partir de G et du tri topologique.

5 - On définit la valuation v de la manière suivante : si la variable x est dans la composante X et $\neg x$ dans la composante Y alors on pose $v(x) = faux$ si $X < Y$ et $v(x) = vrai$ sinon.
→ Montrer que v est définie de manière cohérente et conclure. Quelle est la complexité de 2-SAT ?