Polytech'Montpellier – IG3 **2009-2010**

PROJET C

Réseaux sociaux : recherche de composantes fortement connexes

Contexte du projet

Les réseaux sociaux tels que facebook sont de plus en plus présents dans notre quotidien et leur analyse est très importante, en particulier pour comprendre la structuration et les relations entre individus, et l'influence des réseaux. Les relations sont alors analysées par différentes méthodes issues notamment de la théorie des graphes. En particulier, la recherche de composantes fortement connexes permet non seulement de retrouver les sous-structures du réseau, mais aussi de pointer les « points d'articulation ».

Travail à réaliser

Dans le cadre de ce projet, il s'agira :

1) de trouver les composantes fortement connexes à l'aide des algorithmes connus dans ce cadre (voir annexe) et
2) de permettre la diffusion efficace de messages. On considèrera alors un réseau dans lequel chaque personne doit se connecter et vérifier son compte, en prenant pour hypothèse qu'une personne vérifie son compte régulièrement (toutes les X heures avec X = 1, 2, 24, 72, etc.). Le problème est d'envoyer un message d'une personne à une autre dans le temps le plus court possible. Les personnes n'étant pas forcement amies directement, il faut utiliser des intermédiaires pour transmettre le message. Pour cela, une personne peut renvoyer un message vers une autre personne lorsqu'elle vérifie son compte. On considère alors comme temps de transmission le cas le moins favorable.

Le projet ne sera considéré comme fini qu'après validation sur un très grand ensemble de tests comprenant des types et tailles de réseaux différents.
Les données en entrée devront être lues à partir d'un fichier puis stockées en mémoire à l'aide de structures adéquates. Les résultats devront être écrits dans un fichier de sortie.
Vous devrez utiliser l'API Facebook pour travailler sur des données réelles. Un code vous sera fourni pour ce faire, et vous devrez alors utiliser une fonction de signature

struct compte_facebook get_friends(long int uid, int* no_friends)*

Modalités

Le projet est à rendre par mail à Razvan.Dinu@lirmm.fr et laurent@lirmm.fr à une date qui vous sera indiquée ultérieurement.
Le sujet du mail devra être de la forme < [Projet C] noms > et l'archive jointe au mail devra être nommée <noms.zip> ou <noms.tar> où noms correspond à la liste des noms de famille des participants par ordre alphabétique séparés par un _
L'archive devra comporter un *readme*, un répertoire des fichiers sources avec un *makefile*, un sous-répertoire des fichiers .c et un sous-répertoire des fichiers .h, un répertoire des tests et leurs résultats, un répertoire de documentation comportant la charte de programmation adoptée, un rapport de synthèse (maximum 20 pages), un rapport de tests, un schéma (PDF) de description des modules et de leurs interactions.
Le projet peut être fait en C ou C++

Annexe

Composante fortement connexe d'un sommet

Soit $G = (X, U)$ un graphe orienté. On définit la *composante fortement connexe* d'un sommet x, notée $CFC_G(x)$ par :

$CFC_G(x) = \{y \in X$ tq il existe dans G un chemin de x à y et un chemin de y à $x\}$

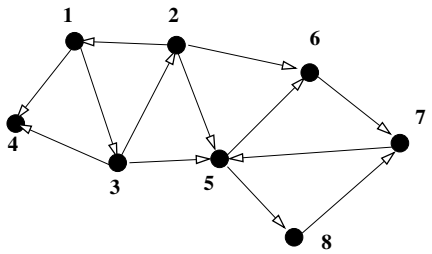

FIG. 1 – Sur ce graphe, $CFC_G(1) = \{1, 2, 3\}, CFC_G(4) = \{4\}, CFC_G(6) = \{5, 6, 7, 8\}$

L'ensemble des composantes fortement connexes de G forme une partition des sommets de G.

Le calcul des composantes fortement connexes de G repose sur deux parcours en profondeur (algorithme PP, voir annexe), l'un du graphe G, l'autre de son dual G^D (c'est-à-dire le graphe obtenu en inversant le sens des arcs de G)

Algorithme de calcul des composantes fortement connexes de G

Données : $G = (X, U)$ un graphe orienté

Résultat : Les composantes fortement connexes de G

1– Exécuter $PP(G)$ pour calculer $\{f(u)/u \in X\}$;

2– Caculer G^D le graphe dual de G (obtenu en inversant le sens des arcs de G);

3– Exécuter $PP(G^D)$ mais dans la boucle principale, on considérera les sommets u par ordre décroissant des $f(u)$;

4– Chaque arborescence de la forêt obtenue en 3 est une composante fortement connexe;

Sur l'exemple, le premier parcours en profondeur de G donne ce qui suit :

Le deuxième parcours en profondeur , parcours de G^D en utilisant l'ordre décroissant des $f(u)$ donne :

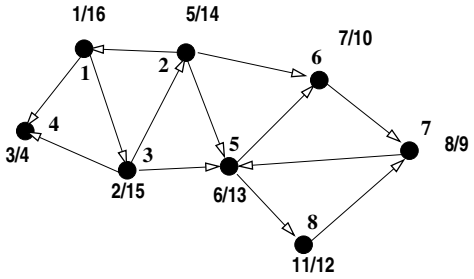

FIG. 2 – Les sommets ordonnés par ordre décroissant des $f(u)$ sont : $1, 3, 2, 5, 8, 6, 7, 4$

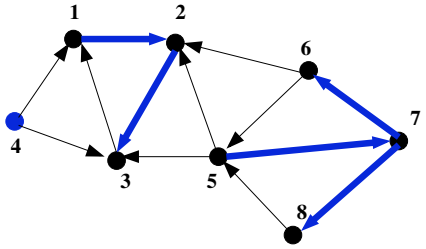

FIG. 3 – Les composantes connexes de la forêt de liaison sont bien $\{1, 2, 3\}, \{4\}, \{5, 6, 7, 8\}$

Annexe : algo de parcours en profondeur

PP(G)
Données : $G = (X, U)$ un graphe orienté
Résultat : les valeurs $d(u)$ et $f(u)$ pour tout u de X
pour chaque $u \in X$ **faire**
    etat(u) <- blanc;
fin
t <- 0;
pour chaque $u \in X$ **faire**
    si *etat(u)=blanc* **alors**
        PProf(u**)**;
    fin
fin

PProf(u)
Données : $G = (X, U)$ un graphe orienté, u un sommet de G
etat(u) = atteint;
t <- t + 1;
d(u) <- t ;
pour chaque $v \in Successeurs(u)$ **faire**
    si *etat(v)=blanc* **alors**
        PProf(v);
    fin
fin
etat(x)=explore ;
t <- t + 1;
f(u) <-t;