République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

ESI ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
Ecole nationale Supérieure d' Informatique
ex. INI (Institut National de formation en Informatique)

2CS - SIL - 1

RAPPORT DU TP N°02 TPGO

Thème

Le problème de la recherche des points d'articulation dans un graphe non orienté

Réalisé par

- AHMED BACHA Ibrahim
- KHODJA Mehdi Nassim

Promotion : 2018/2019

Table des matières

Problématique

En théorie des graphes, un point d'articulation est un sommet d'un graphe non orienté qui, si on le retire du graphe, augmente le nombre de composantes connexes. Si le graphe était connexe avant de retirer ce sommet, il devient donc non connexe.

La recherche de ces points a beaucoup d'application dans différents domaines tels que : les réseaux de communications, les trafics aériens et les circuits électriques.

L'algorithme naïf consiste pour la détection de ces points consiste à comparer pour chaque sommet le nombre des composantes connexes du graphe sans le sommet en question avec le nombre de composantes connexes du graphe initial. Si le nombre n'est pas le même alors le sommet est un point d'articulation. La complexité de cet algorithme est O(n*(n+m)), tels que : n est le nombre de sommets et m est le nombre d'arrêtes.

Dans ce TP, nous allons étudier une autre approche pour la recherche des points d'articulation dans un graphe non orienté qui basé sur l'algorithme de « Hopcroft et Tarjan » qui utilise le parcours DFS du graphe.

Solution

Après avoir appliqué le parcours DFS au graphe, on obtient l'arbre de parcours DFS.

Pour tous nœud u de cet arbre on a :

- Si u est la racine de l'arbre alors u est un point d'articulation si et seulement si le nombre de ces fils est supérieur à 1.
- Si u n'est pas la racine de l'arbre alors u est un point d'articulation si et seulement s'il a au moins un fils v tels qu'il n'existe pas un nœud dans le sous-arbre de v qui a comme fils un des ancêtres de u.

Pour la première condition, il suffit d'utiliser un tableau « children » pour sauvegarder le nombre de fils d'un nœud. Cette condition résulte de l'algorithme DFS, c'est-à-dire que la racine n'aura pas plus qu'un fils si on avait pas au moins deux fils tels qu'on peut pas atteindre un fils à partir de l'autre (il n'existe pas un chemin entre eux).

Pour la deuxième condition, on utilise un tableau « order » qui contient l'ordre de visite (initialiser à 0 pour détecter en même temps si un nœud est déjà visité), un tableau « low » pour connaître l'ordre de visite inférieure quand peut atteindre à partir d'un nœud qui nous permettra de savoir pour un nœud u donné s'il y a un de ces fils qui a une valeur de low supérieure ou égale à l'ordre de visite de u, autrement dit, il n'y a pas une arrête de retour dans le sous-arbre d'un fils qui permettra de supprimer le nœud u sans rendre ce fils inaccessible.

La complexité de cet algorithme est O(n+m).

Jeu d’essai

Le fichier input.txt contient la représentation d’un graphe avec une liste d’adjacence.

```
ibrahim@ibrahim-Ubuntu:~/graphe_articulation_points$ ./main
Adjacency list :
0:      1       2
1:      0       2       3
2:      0       1
3:      1       4       5
4:      3       6
5:      3       6
6:      4       5
Orders :
0       1       2       3       4       5       6
1       2       3       4       5       7       6
Parents :
0       1       2       3       4       5       6
-1      0       1       1       3       6       4
Articulation points :
1       3
ibrahim@ibrahim-Ubuntu:~/graphe_articulation_points$
```