Théorie des Programmes & Résolution de Problèmes - TP2

Par:
Kerim Yahia
Khelil Mohamed Wael
SIL2 TPGO 2018/2019

Les points d'articulation

l'énoncé

En mathématiques, et en particulier 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. Le but de TP est d'écrire un algorithme qui permet de trouver tout les points d'articulation d'un graphe donné.

Une solution simple

Soit G un graphe à *n* sommets et *m* arêtes :

```
a = nombre de composantes connexes de G (déterminé à l'aide de DFS)
pour chaque sommet v de V ayant des arêtes incidentes
        retirer v de V
        b = nombre de composantes connexes de G-{ v }
        si b > a
                v est un point d'articulation de G
        remettre en place v
```

l'algorithme a une complexité d'ordre O(nm),il en existe de bien meilleurs. John Hopcroft et Robert Tarjan en ont décrit un en 1973 de complexité d'ordre O(n+m) et qui utilise l'algorithme de parcours en profondeur

L'algorithme de John Hopcroft et Robert Tarjan

L'idée c'est d'utiliser l'algorithme de parcours en profondeur pour construire un arbre qu'on va le nommé l'arbre DFS.Dans l'arbre DFS, un sommet "x" est un parent de "y" si "y" est découvris à partir de "x". On dit qu'un sommet "u" est un point d'articulation si l'un des deux condition est vrai :

- Le sommet "u" est la racine de l'arbre DFS, et il a au minimum 2 fils.
- Le sommet "u" n'est pas la racine de l'arbre DFS, et il existe un sommet de la sous-arbre enraciné par "u" (nommé "v"), tel qu'il a un arret de retour vers l'un des ancêtres de "u" (dans l'arbre DFS).

Le graphe est représenté par une matrice. Pour l’arbre DFS, voici les structures utilisé :

```
struct _Node {
    int name;
    int parent;
    struct _ListNode *children;
} typedef Node;

struct _ListNode {
    Node *node;
    struct _ListNode *nextNode;
}typedef ListNode;
```

On va ajouter quelque traitement dans l’algorithme de DFS afin de trouver les PAs. Parmi les 2 conditions mentionné au dessus, la première est simple à détecter : il suffit de calculer pour chaque noeud, le nombre de ses fils, s'il est supérieur ou égale à 2, et le champs “parent” égale à “-1” (équivalent de NULL), il est un point d’articulation.

la deuxième condition est un peu plus compliqué. Pour la vérifier : on va utiliser deux tableaux : un premier tableau (nommé: order[]) qui va contenir le temps de découverte pour chaque sommet lors de la création de l’arbre DFS, est un deuxième tableau (nommé low[]) initialisé comme suite :

```
Pour chaque noeud v :
        low[v] = order[v]
```

Dès qu’on crée l’arbre DFS, on commence la mise à jour du tableau low[] a partir des feuilles comme suite :

```
Pour chaque sommet u:
        Pour chaque v adjacent de u :
                SI (u est le parent de v)
                        low[u] = min ( low[u] , low[v] )            // (1)
                SINON SI (v n’est le parent de u)
                        low[u] = min ( low[u] , order[v] )          // (2)
```

A la fin de l’algorithme, on aura des ensembles de noeuds qui ont la même valeur low[] (et qui représente les composants connexe quand on supprime les points d’articulation), la valeur du low[] est égale à la valeur order[] (temps de découverte) du noeud le plus haut dans l’arbre et qui appartient au même composant. Donc, les points d’articulation sont les noeuds (i.e : u) qui ont un fils (i.e : v) tel que la valeur : low[v] > order[u].

Pour le cas ou deux points d'articulation possibles sont adjacents, on peut détecter le deuxième point on modifiant dans la condition précédente tel que :

low[v] >= order[u].

(car on a dit que la valeur low[] est égale à la valeur order[] (temps de découverte) du noeud le plus haut dans l'arbre et qui appartient au même composant). Alors, la valeur de low[] est égale à son temps de découverte.

Donc, on vérifie les APs comme suite :

```
Pour chaque sommet u:
        SI (u.parent == -1) et (u.nb fils > 2)
                u est un point d'articulation
        SINON SI (u.parent != -1) et u a un fils v tq : low[v] >= order[u]
                u est un point d'articulation
```