University of Liège

Théorie des graphes

MATH-0499

Algorithme de Kosaraju-Sharir

Julien Gustin, Alexandre Bruno

December 12, 2019

1 Explication algorithme

Kosaraju-Sharir est un algorithme permettant de calculer les composantes fortement connexes d'un graphique orienté.

Pour cela prenons G un graphique orienté, l'algorithme s'opère en 2 étapes principales.

1. Parcourir G à l'aide de l'agorithme de parcours en profondeurs (DFS), et noter chacun de ces sommets par leur "ordres de visites" et celui qui nous intéresse dans notre cas sera le post-ordre.

2. Faire de même avec la transposée de G, en suivant l'ordre des sommets donné par la première étape.

Les arbres produits par le deuxième parcours nous donne les composantes fortement connexes de G.

2 Stratégie utilisée

Pour faciliter l'implémentation, nous avons modifié la structure sommet de base tel que ;
le champ *couleur* nous permet de savoir si un sommet a déjà été parcouru par DFS, si c'est le cas, nous lui donnons la couleur rouge, afin de ne pas parcourir plusieurs fois un meme circuit de sommet déjà parcourus.
info nous sert juste dans la fonction DecroissantDateFin qui nous renvoit le label du sommet avec la date de fin la plus élévée. Afin d'éviter de renvoyer plusieurs fois le meme sommet, nous lui donnons l'information '1' qui veut dire que ce sommet à déjà été renvoyé précédement. Si $info \leftarrow 2$ cela veut dire que le sommet utilisé provient de la transposée du graphique principale.

Listing 1: structure

```
struct sommet {
        int label;
        int couleur; /* 0 = blanc, 1 = rouge */
        int info; /* 0 = init, 1 = d j  v rifier, 2 = transpos e*/
        struct date date;
        struct sommet *suivant;
        struct eltadj *adj;
};

int decroissantDateFin(SOMMET *psommet)
{
  SOMMET *sommetR = psommet;
  int sommetN;
  sommetN = -1;

  while(psommet != NULL){

    if((sommetN < psommet->date.fin) && psommet->info == 0){
      sommetN = psommet->date.fin;
      sommetR = psommet;
    }
```

```
    psommet = psommet->suivant;
  }
  sommetR->info = 1;
  return sommetR->label;
}
```

Pour réaliser la transposée du graphique principal, nous avons fait au plus simple : nous lisons le fichier du graphique et stockons son contenus dans une matrice temporaire qui la transpose directement. On réecrit cette matrice dans un autre fichier et finalement on utilise la fonction LIREFICHIER de graph.h/c avec le fichier nouvellement crée et nous avons notre transposée au bon format (celui de graph.c).

3 Utilisation du programme

1. Ouvrir la console dans le répertoire du fichier et écriver make ↩ ;
2. Compiler avec ./MAIN -I <"GRAPHE"> où "graphe" est un fichier contenant un fichier représenté sous la forme:

   x, 1, x, x, 1, x, x, x
   x, x, 1, x, 1, x, x, x
   x, x, x, x, x, 1, 1, x
   x, x, x, x, x, x, 1, 1
   x, x, x, x, x, 1, x, x
   x, 1, x, x, x, x, x, x
   x, x, x, x, x, 1, x, x
   x, x, x, x, x, x, 1, x

   Qui correspond à ce graphique ci

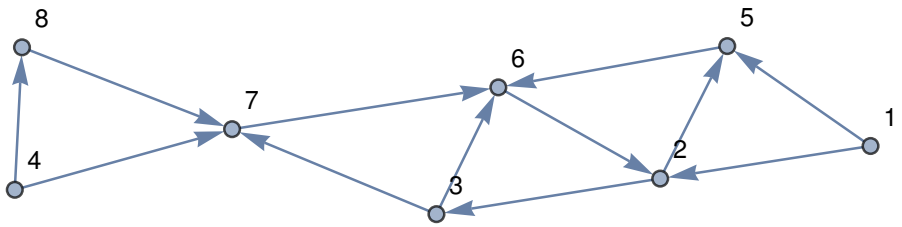

Figure 1: Graphique

3. Regarder l'output du fichier dans la console qui montre les composantes f-connexes du graphique.

4 Exemple commenté

Prenons l'exemple du points **3.2**, après le premier passage de DFS dans G, les dates d'ordre de parcours des sommets allant du premier jusqu'au huitième sont noté par **(début, fin)**

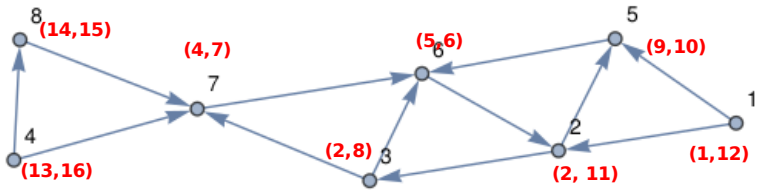

Figure 2: G

Ainsi nous savons que pour la transposée de G nous allons devoir parcourir les sommets par l'ordre décroissant par raport à leurs date fin. (4, 8, 1, 2, 5, 3, 7, 6)

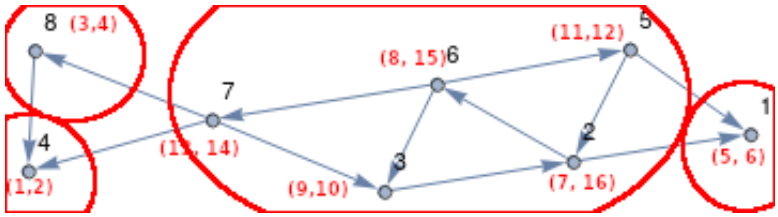

Figure 3: G

Le sommet 4 n'ayant aucun degré sortant est lui même un élément fortement connexe, de même pour le 8 et et le 1. Cependant quand nous lancons DFS à partir du sommet 2 chacun des éléments qu'il traverse sont f-connexes.

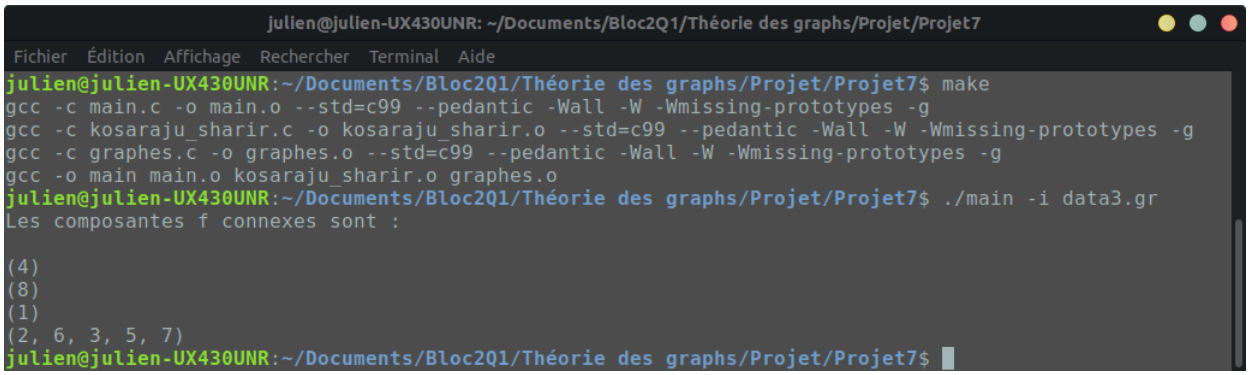

Figure 4: Exemple

Nous pouvons ainsi voir sur l'exemple si dessus que les éléments fortement connexes sont les bons.

5 Différence avec l'algorithme de Tarjan

La différence principale entre l'algorithme de Tarjan et l'algorithme de Kosaraju, c'est le fait que Tarjan ne parcours qu'une seule fois le graphe, tandis que Kosaraju le parcourt 2 fois : Un parcours du graphe et un autre parcours de la transposée du graphe (et avec cela viens le calcul de la transposé). Toutefois, la complexité des deux algorithmes est linéaire, O(v+e) avec v le nombre d'arrêtes et e le nombre de sommets (v et e de l'anglais, vertices et edges).

L'implémentation de Kosaraju nous semble plus évidente par rapport à celle de Tarjan, en effet Tarjan ne fait qu'un seul parcours mais pour cela il utilise un algorithme plus "lourd" que celui de Kosaraju. Sans rentrer trop dans les détails, l'algorithme de Tarjan utilise des dates d'ordres de passage et le numéro du sommet du début d'une SCC , noté : (numero de passage, numéro du sommet "racine").

Notes :

- SCC est l'abrégé anglais pour composante fortement connexe
- "racine" le premier sommmet d'une SCC dans le parcours du graphe

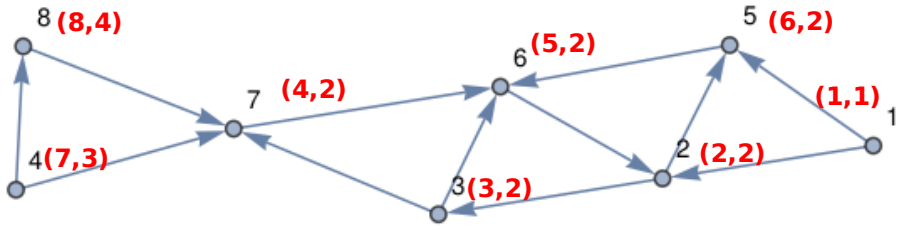

Figure 5: Notation à la Tarjan

6 Bibliographie

6.1 Source

- ☐ https://www.lrde.epita.fr/ max/theg/cours3.pdf
- ☐ https://fr.wikipedia.org/wiki/Algorithme_de_Kosaraju
- ☐ https://www.quora.com