Grands Réseaux d'Interaction
TP 5 Détection de communautés: Algorithme de Louvain

Laurent Viennot
laurent.viennot@inria.fr

26 février 2021

à rendre avant le 12 mars 15h45

Participation : La notation tiendra compte du nombre de personnes dans votre groupe. Si tout le monde n'est pas présent dans votre groupe, envoyez un mail à `laurent.viennot@inria.fr`, avec copie aux autres membres du groupe, avec sujet `GRI TP5` pour excuser ceux qui ne participent pas (un seul mail par groupe).

Consignes : Si le nom de votre groupe est `nomgrp`, un seul membre du groupe doit déposer sur moodle une archive `nomgrp.zip` qui se décompresse en un répertoire `nomgrp` contenant le source de votre programme. Si vous utilisez java, la compilation par `javac *.java` doit produire un fichier `TP5.class` qui sera utilisé pour la correction par `java TP5 arg1 arg2 ...` avec certains arguments `arg1, arg2, ...` détaillés pour chaque question. Si vous utilisez python, votre programme doit s'appeler `TP5.py` et la correction utilisera `python3 TP5.py arg1 arg2 ....`

À vérifier :
- La sortie standard de votre programme correspond au caractère près à ce qui est indiqué dans l'énnoncé.
- Si vous voulez afficher des informations supplémentaires, c'est possible sur la sortie d'erreur (utilisez des `System.err.println()`).
- Nouveau : attention de ne pas afficher trop d'informations car cela peut provoquer un timeout dans les tests sur de grands graphes.
- Effacez tous les fichiers .class de votre répertoire et sous-répertoires avant de vérifier que `javac *.java` réussit bien à compiler votre programme. Au besoin, inclure un fichier `Makefile` de sorte que la commande `make` permette de compiler votre programme.
- Votre archive se décompresse bien en un répertoire du nom de votre groupe et ne contient aucun fichier `.class`.

Le but du TD est de programmer la première phase de l'algorithme de Louvain pour détecter des communautés.

1 Communautés et modularité

Rappelons que la modularité Q d'une partition en communautés d'un graphe G est donnée par :

$$Q = \frac{1}{2m} \sum_{u,v} \left(A_{uv} - \frac{d_u d_v}{2m} \right) \delta_{uv}$$

où $A_{uv} = 1$ si $uv \in E(G)$, $A_{uv} = 0$ sinon, où $\delta_{uv} = 1$ si u et v sont dans la même communauté et $\delta_{uv} = 0$ sinon, et où d_u désigne le degré d'un sommet u. Le graphe est supposé symétrique. Cependant la somme s'effectue sur tous les couples u, v, d'où la normalisation par $2m$ (et non m).

Une formulation équivalente, si d_u^C désigne le nombre de voisins de u dans la communauté C, est :

$$Q = \sum_C \left[\frac{\sum_{u,v \in C} A_{uv}}{2m} - \left(\frac{\sum_{u \in C} d_u}{2m} \right)^2 \right] = \sum_C \left[\frac{\sum_{u \in C} d_u^C}{2m} - \left(\frac{\sum_{u \in C} d_u}{2m} \right)^2 \right]$$

On en déduit que si un sommet u est déplacé de la communauté B vers la communauté C, alors la modularité va augmenter de ΔQ avec :

$$(2m)^2 \Delta Q = 4m(d_u^C - d_u^B) - 2d_u(S_C - S_B + d_u)$$

où $S_C = \sum_{u \in C} d_u$ désigne la somme des degrés des sommets d'une communauté C.

Écrire une classe pour représenter les communautés d'un graphe ainsi que la modularité de la partition correspondante. Stocker pour chaque sommet un numéro de communauté. Initialement, chaque sommet `u` est isolé dans sa propre communauté de numéro `u`. Pour un calcul exact, on maintiendra $(2m)^2 Q$ plutôt que Q. Attention d'utiliser des entiers de 64 bits pour tous les calculs (type `long`). (Si `d` est un `int`, on peut écrire `1L * d * d` pour que la multiplication se fasse avec des `long`.) Notez qu'il faut calculer la modularité initiale car elle n'est pas nulle !

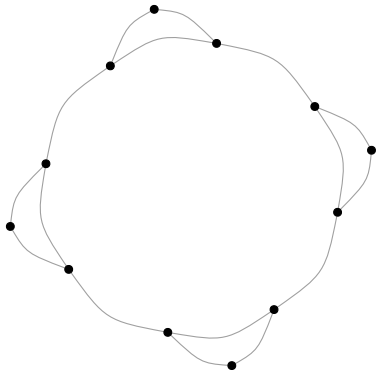
Pour pouvoir permettre des déplacements de sommets et maintenir la modularité de la partition courante selon la formule ci-dessus, on stockera pour chaque communauté C la somme $S_C = \sum_{u \in C} d_u$ des degrés des sommets de la communauté. Écrire une fonction permettant de déplacer un sommet u de sa communauté vers une autre en maintenant à jour la valeur de la modularité. Sa complexité devra être linéaire (en $O(d_u)$). Remarquons que le degré de u peut être calculé à la volée avec une telle complexité.

Commande `delta`. Si les arguments de votre programme sont : le mot `delta`, un nom de fichier contenant un graphe, son nombre d'arêtes, un sommet u et un sommet v, votre programme devra alors afficher $(2m)^2 Q$ après initialisation et après avoir déplacé u de sa communauté dans celle de v. Par exemple :

```
$ java TP5 delta clique3-ring4.txt 16 0 1
-88
-36
$ java TP5 delta clique3-ring4.txt 16 2 3
-88
```

```
-42
$ java TP5 delta clique3-ring4.txt 16 0 3
-88
-106
```

Les tests ci-avant sont faits sur un anneau de 4 cliques de 3 sommets :



```
$ cat clique3-ring4.txt
0 1
0 2
0 11
1 2
2 3
3 4
3 5
4 5
5 6
6 7
6 8
7 8
8 9
9 10
9 11
10 11
```

Le fichier `test/clique3-ring4.txt` ainsi que les autres exemples qui suivent sont disponibles sur `https://who.rocq.inria.fr/Laurent.Viennot/t/graphs/`.

Commande `delta12321`. Si les arguments de votre programme sont : le mot `delta12321`, un nom de fichier contenant un graphe, son nombre d'arêtes, un sommet u, un sommet v et un numéro de communauté c, votre programme devra alors afficher $(2m)^2Q$ après initialisation, après déplacement de u dans la communauté c, de v dans la communauté c, de u dans sa communauté d'origine u et de v dans sa communauté d'origine v. Par exemple :

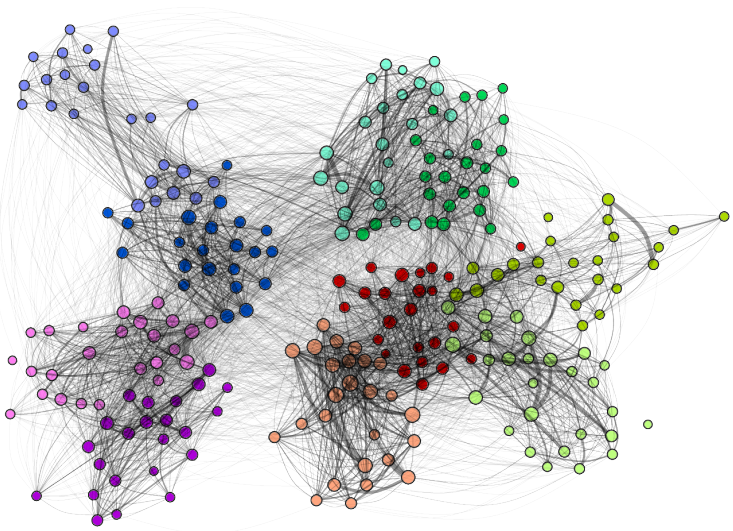
```
$ java TP5 delta12321 clique3-ring4.txt 16 0 1 2
-88
-42
62
-36
-88
$ java TP5 delta12321 clique3-ring4.txt 16 1 2 3
-88
-100
-2
-42
-88
$ java TP5 delta12321 clique3-ring4.txt 16 1 3 4
-88
-96
-56
-36
-88
```

2 Modularité d'une partition donnée

Commande `deplacements`. Si les arguments de votre programme sont : le mot `deplacements`, un nom de fichier contenant un graphe, son nombre d'arêtes, un nom de fichier de déplacements, le nombre de déplacements dans ce fichier, votre programme devra alors afficher la modularité Q obtenue une fois tous les déplacements effectués. Afficher Q (et non $(2m)^2Q$) avec 5 décimales (utiliser `System.out.format("%.5f\n", val)`). Le fichier de déplacements est fourni au même format qu'un graphe, chaque ligne uc étant interprétée comme déplacer u dans la communauté numéro c. Par exemple :

```
$ java TP5 deplacements clique3-ring4.txt 16 clique3-ring4-part2.txt 12
0,33984
$ java TP5 deplacements primaryschool.txt 1957 primaryschool-classes.txt 236
0,66845
```

Le fichier `clique3-ring4-part2.txt` contient les déplacements correspondants à la partition naturelle en cliques. Le fichier `primaryschool.txt` est un graphe de contacts face à face sur une journée dans une école primaire (voir `http://www.sociopatterns.org/`). Le fichier `primaryschool-classes.txt` permet d'obtenir la partition en classes illustrée ci-après. Ils sont accessible sur `https://who.rocq.inria.fr/Laurent.Viennot/t/graphs/`.



3 Algorithme de Louvain

Une phase de l'algorithme de Louvain procède par passes. Lors d'une passe, chaque sommet u est inspecté : pour chaque communauté c qui contient un voisin de u, on calcule l'incrément de modularité qu'on obtiendrait en déplaçant u dans c. Si l'incrément maximum est strictement positif, alors on déplace u dans la communauté correspondante. La complexité de l'inspection d'un sommet doit être linéaire en son degré. Ainsi une passe sur tous les sommets doit prendre un temps linéaire en la taille du graphe. L'algorithme consiste à répéter des passes tant que la modularité augmente (on finit forcément par être bloqué sur un maximum local à un moment). Le résultat obtenu dépendant de l'ordre dans lequel on considère les sommets, on les inspectera dans l'ordre $0, 1, \ldots, n-1$. De même, si plusieurs communautés voisines procurent un incrément maximal, on choisira celle de plus petit numéro.

Commande `phase`. Si les arguments de votre programme sont : le mot `phase`, un nom de fichier contenant un graphe, et son nombre d'arêtes, votre programme devra alors afficher la modularité Q obtenue après chaque passe d'une phase de l'algorithme de Louvain. Afficher Q avec 5 décimales. Par exemple :

```
$ java TP5 phase clique3-ring4.txt 16
0,50000
0,50000
$ java TP5 phase clique5-ring30.txt 330
0,52383
0,87576
0,87576
$ java TP5 phase com-dblp.ungraph.txt 1049866
0,42444
0,57870
0,60383
```

```
0,61151
0,61429
0,61547
0,61588
0,61603
0,61612
0,61613
0,61615
0,61615
0,61615
```

Fichier `remarques.txt`**.** Testez cet algorithme sur le graphe `primaryschool.txt` dont on connaît les communautés naturelles. Qu'observez vous ? Joindre un fichier `remarques.txt` avec vos remarques.