

INFO-F-203

Projet

Les bons comptes font les bons amis!

Année académique 2016–2017

Pour toutes les questions, nous vous invitons à vous adresser à Fabio Sciamannini
(fabio.sciamannini@ulb.ac.be)

1 Le problème

La jeune startup, *Radino*[®], s'apprête à lancer sa nouvelle application pour smartphone. Cette application a pour but de faciliter les bons comptes entre bons amis.

En effet, il n'y a rien de plus prise de tête que les calculs d'apothicaire quand, par exemple, on organise une fête surprise et on doit faire des courses, car après, il faut réfléchir qui doit combien et à qui, et ça peut devenir une source de discordes infinies.

Pour éviter cela et faire en sorte que l'on soit toujours bon amis, cette application permettra d'affecter à chacun les dépenses qu'il a effectuées et déterminer la part à payer afin que tous contribuent de manière équitable.

2 Objectifs du projet

Dans ce projet, les dettes entre amis seront modélisées par un graphe pondéré. Les noeuds du graphe représentent des personnes, et les arcs représentent une dette.

Par exemple, dans le graphe de la Figure 1 E doit 25 à F, A doit 10 à B et B lui même doit 20 à A.

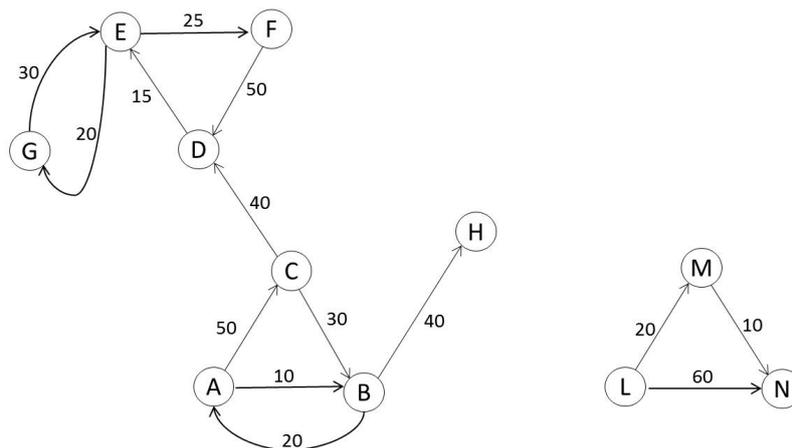


FIGURE 1 – Graphe des dettes entre amis.

2.1 Simplification des dettes

On se rend rapidement compte que ce graphe peut être simplifié. Par exemple, dans le graphe de la Figure 1, A doit 10 à B qui lui-même doit 20 à A. Ces deux dettes peuvent se simplifier en une seule dette : B doit 10 à A (et A ne doit plus rien à B). D'autres simplifications sont possibles dans le graphe en question.

Plus généralement, tous les cycles dirigés peuvent être simplifiés. Sur base de cette idée, la simplification du graphe de la Figure 1 permet d'obtenir le graphe de la Figure 2. Dans ce graphe chaque cycle a été simplifié. Les arcs qui devraient disparaître sont conservés avec un poids de zéro (cela indique une dette passée), afin de conserver l'information 'qui connaît qui'.

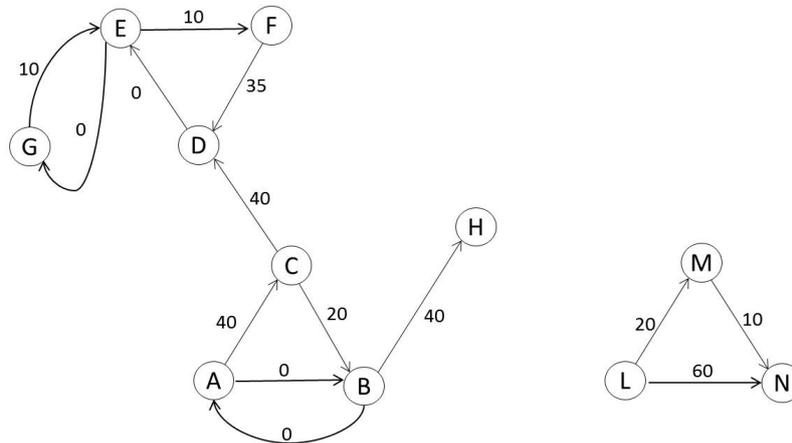


FIGURE 2 – Les dettes entre les amis après la suppression des cycles.

Nous vous demandons de concevoir et implémenter un algorithme qui reçoit en entrée le graphe décrivant les dettes existantes et modifie ce graphe afin d'obtenir le graphe simplifié.

2.2 Identification des communautés

Une communauté est un ensemble de personnes connectées par leurs dettes actuelles ou passées. Dans l'exemple de la Figure 2 nous avons deux communautés. Les noeuds A, B, C, D, E, F, G et H forment une première communauté et L, M, N en forment une seconde.

Nous vous demandons de concevoir et implémenter un algorithme qui reçoit en entrée le graphe décrivant les dettes actuelles et passées et retourne l'ensemble des communautés.

2.3 Identification des hubs sociaux

L'équipe de Radino[®] s'intéresse à ce qu'ils appellent les hubs sociaux. Ce sont des personnes qui font le lien entre plusieurs communautés. Dans notre cas, on identifiera un hub social à un noeud du graphe tel que sa suppression scinde une communauté en deux. En d'autres mots, les hub sociaux sont les points d'articulation de notre graphe des dettes.

Dans l'exemple de la Figure 2 les noeuds E, D, C, B sont les hubs sociaux du graphe car leur suppression entraîne un morcellement des communautés.

Radino[®] ne s'intéresse qu'aux hubs sociaux qui 'connectent' 2 communautés d'au moins K individus et compte bien les inciter à faire la promo de sa nouvelle application.

Nous vous demandons de concevoir et implémenter un algorithme qui reçoit en entrée le graphe décrivant les dettes actuelles et passées, ainsi qu'un nombre K et retourne l'ensemble des hubs sociaux tels que leur suppression entraîne la création de 2 communautés d'au moins K individus.

2.4 Identification du plus grand groupe d'amis

Un groupe d'amis est un ensemble de personnes tel que chacune de ces personnes a une dette passée ou présente avec chacun des membres du groupe (une dette dans un sens ou dans l'autre). En d'autres mots, un groupe d'amis est l'ensemble des noeuds d'un sous graphe non-dirigé *complet* du graphe des dettes.

Dans l'exemple de la Figure 2, nous avons trois groupes d'amis. Les noeuds A, B, C forment un premier groupe, D, E, F en forment un deuxième et L, M, N en forment un troisième.

Radino[®] tient à encourager la formation de ces groupes d'utilisateurs avides de leur nouvelle application.

Nous vous demandons de concevoir et implémenter un algorithme qui reçoit en entrée le graphe décrivant les dettes actuelles et passées et retourne le plus grand groupe d'amis (n'importe lequel si plusieurs groupes ont la même taille).

2.5 Simplifications supplémentaires du graphe

D'autres simplifications du graphe sont possibles en plus des simplifications de cycles. Par exemple, la Figure 3 illustre une simplification concernant un cycle non-dirigés. Il est cependant fondamental que les simplifications ne modifient que les poids des arcs, elles n'ajouteront ni ne supprimeront aucun arc (même si cela permet d'en supprimer beaucoup d'autres). Le but des simplifications étant de supprimer des dettes, c'est à dire d'obtenir un graphe équivalent tout en maximisant le nombre d'arc avec un poids nul.

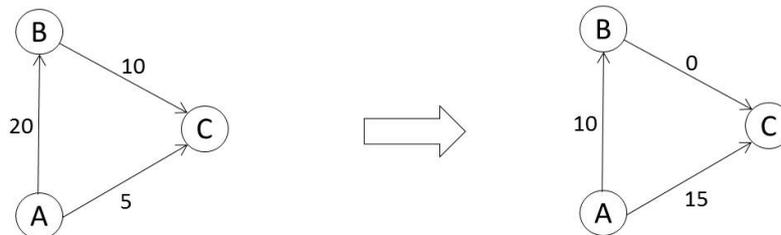


FIGURE 3 – Exemples de simplification de cycles des dettes.

Nous vous demandons de concevoir et implémenter un (ou éventuellement plusieurs) algorithme qui reçoit en entrée le graphe décrivant les dettes actuelles et passées et effectue une ou plusieurs simplifications supplémentaires. Ces simplifications seront expliquées et analysées dans votre rapport.

3 Implémentation

Nous vous demandons d'implémenter les deux algorithmes :

- Simplifications des dettes (suppression des cycles).
 - Identification des communautés.
- et un (ou deux ou les trois) algorithme au choix parmi :
- Identification des hubs sociaux.
 - Identification du plus grand groupe d'amis.
 - Simplifications supplémentaires du graphe de dette.

L'évaluation tiendra compte de la correction et de la complexité algorithmique de vos solutions.

Langages

Ces algorithmes devront être implémentés en *Java* ou en *Python* en suivant au maximum le paradigme Orienté Objet (OO). Il est conseillé de bien concevoir votre algorithme afin de minimiser le code redondant en pensant aux responsabilités de chaque classe et de chaque méthode et à l'utilité de l'héritage et de la composition. Vous pouvez utiliser les structures de données des bibliothèques standard fournies par *Java* ou par *Python*.

Encodage du graphe de départ

Les données du graphe décrivant le cercle d'amis seront transmises à votre programme via la lecture d'un seul fichier (dont le nom sera passé en argument de votre programme). Le fichier contiendra la description du graphe, composée :

- du nombre de nœuds du graphe (un entier) ;
- du nom de chaque nœud (caractères alphanumériques uniquement) ;
- d'une liste de triplets "NoeudDeDépart NoeudD'Arrivée Dette" représentant chaque arc du graphe. Chaque nœud est identifié par son nom et la dette est un réel.

Voici un exemple de fichier :

```
11
A B 10
A C 50
B A 20
B H 40
C B 30
C D 40
D E 15
E F 25
E G 20
F D 50
G E 30
L M 20
L N 60
M N 10
```

La liste des arcs se finit lorsque le fichier se termine.

Effectuez le *parsing* dans une ou plusieurs méthodes (pas dans le main) et placez-les dans les classes qui vous paraissent les plus à même d'effectuer ce travail. Vous pouvez supposer que le fichier d'entrée ne comporte pas d'erreur.

Programme et sortie standard

Votre programme doit recevoir le nom du fichier en argument et écrire sur l'output standard la solution de la manière suivante :

- le graphe simplifié par votre premier algorithme sous la même forme que le fichier d'entrée ;
- la liste des communautés ;
- la liste des hub sociaux, et/ou le plus grand groupe d'amis, et/ou le graphe simplifié par votre dernier algorithme.

Tests et makefile

Chaque algorithme devra être testé à l'aide de plusieurs graphes permettant de couvrir les différentes cas possibles que ces algorithmes peuvent rencontrer. Ce sera donc à vous de bien penser et préparer des tests qui devront être remis avec votre code.

Si les algorithmes ont été réalisés en *Java*, votre code source sera accompagné d'un *makefile* permettant de compiler et exécuter votre projet ainsi que de lancer les tests.

Le programme doit être exécutable aux salles machines du bâtiment NO.

4 Rapport

On vous demande de remettre un *rapport* à la présentation soignée. Le rapport contiendra les informations suivantes :

- une brève introduction au projet ;
- une description de chaque algorithme utilisé et une présentation des tests le concernant ;
- pour l’algorithme choisi (identification des hubs, identification du groupe ou simplifications supplémentaires), une description du problème en terme de graphe, une analyse (est-ce un problème connu ?) et une présentation de vos solutions ;
- une liste des différentes sources (livres, sites internet, ...) utilisées ;
- un paragraphe conclusif contenant , parmi d’autres choses, une description de la répartition des tâches (si vous avez travaillé en groupe) et des difficultés rencontrées en résolvant le projet.

Attention : *le code de vos algorithmes ne devra pas être présent dans le rapport (ni dans des annexes)!* Mais le rapport doit présenter vos algorithmes (en pseudo code) afin de nous permettre de comprendre vos solutions sans lire votre code.

Le rapport (en ce compris sa présentation) fait partie *intégrale* de votre travail et compte pour 50% de la note finale du projet.

Consignes pour la remise du projet

À respecter scrupuleusement !

1. Date : **le 19 Décembre 2016 - 13h** (vous pouvez ne pas remettre votre projet dans le cas de fin du monde).
2. Le projet peut être réalisé individuellement ou par groupe de deux (ce que nous recommandons).
3. Les modalités pour la remise du rapport sont :
 - Lieu : **Université Virtuelle** (uv.ulb.a.c.be) et **au Secrétariat « étudiants » du Département d’Informatique, local 2N8.104.**
 - Heure : **Avant 13h.****Après 13h**, les projets seront considérés comme **en retard**, et vous perdrez **la moitié des points** sur votre note finale et vous perdrez en plus 3 points par jour de retard. Les projets en retard doivent être déposés **dans la caisse** prévue à cet effet près du secrétariat.
4. Les modalités pour la remise du code sont :
 - Lieu : **Université Virtuelle** (uv.ulb.a.c.be)
 - Tous les fichiers dans **une seule archive** au format **ZIP** (ni tar.gz, ni rar, ni autre chose !).
 - L’archive doit porter **le nom** AG2.NomEtudiant1-NomEtudiant2.zip, si deux étudiants ont réalisé le projet ; AG2.NomEtudiant.zip dans le cas contraire.Si vous ne respectez pas ces critères, nous considérerons que vous n’avez pas rendu de code. Si votre projet ne compile pas avec la commande `make`, on ne corrige pas votre projet.

Bon travail !