INFO–F–203 — Algorithmique 2
Manipulation de graphes : transfert d'argent

Année académique 2013–2014

Le problème

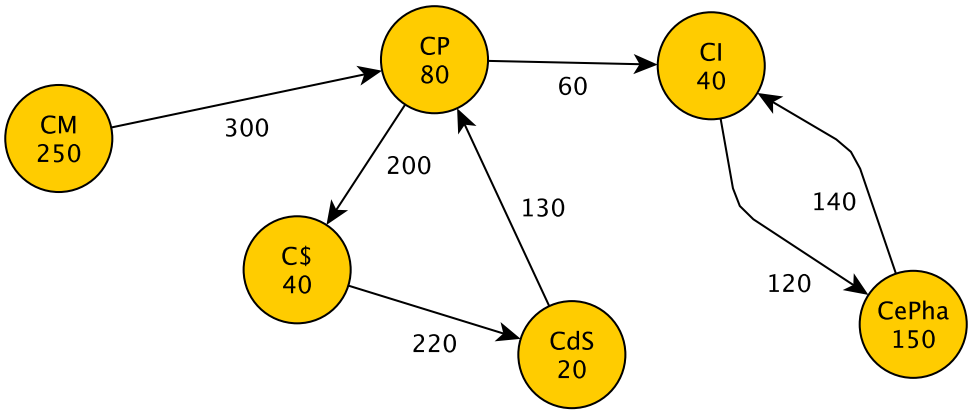

FIGURE 1 – Les dettes entre les cercles (en euros).

Les trésoriers des cercles de l'ULB vous appellent à l'aide ! Les cercles de l'ULB ont des dettes envers d'autres cercles. Malheureusement beaucoup d'entre eux se trouvent dans la situation suivante : il n'y a pas assez d'argent sur le compte pour rembourser leurs dettes, mais il existe des "cycles de dettes", voir Figure 1. Malheureusement, même après la suppression des cycles, certains cercles ne savent pas rembourser toutes leurs dettes (voir Figure 2).

Objectif du projet

On vous demande de produire un algorithme efficace pour résoudre ce problème. L'algorithme en question reçoit en entrée le graphe décrivant les dettes et l'état des comptes des cercles de l'ULB. L'algorithme doit produire :

1. un fichier (on vous invite à utiliser *graphviz* [1]) avec une représentation des dettes après la suppression des cycles ;
2. la séquence dans laquelle les cercles doivent rendre l'argent les uns aux autres (après la suppression des cycles) ;
3. un fichier avec une représentation de la situation une fois qu'un maximum de dettes ont été remboursées.

1. `http://www.graphviz.org/`

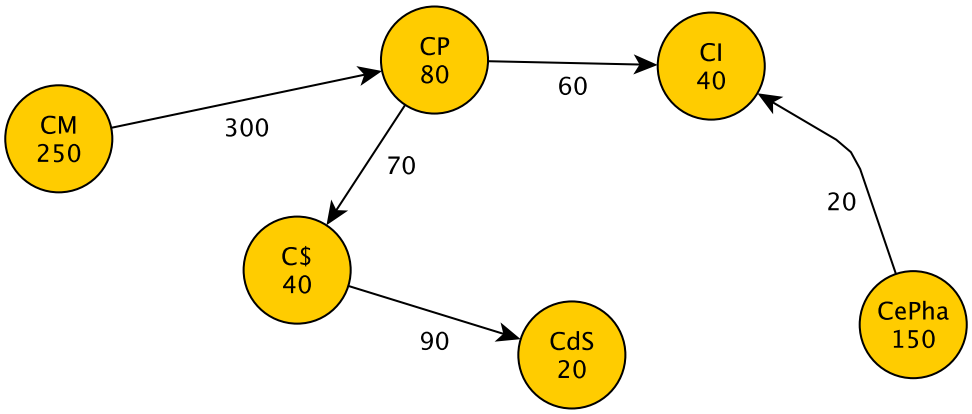

FIGURE 2 – Les dettes entre les cercles après la suppression des cycles.

Détails techniques

Ces algorithmes devront être implémentés en *Java* en suivant au maximum le paradigme Orienté Objet (OO). De plus, les deux algorithmes demandés n'étant pas très différents, il est possible de bien concevoir votre code 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. *Un conseil : ne faites jamais de copier-coller.*

Vous pouvez utiliser les structures de données des librairies standard fournies par Java.

Les données du graphe décrivant la ville 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 (une chaîne de caractères alphanumériques uniquement), suivi d'un nombre (la quantité d'argent sur le compte) suivi d'un retour à la ligne ;
- d'une liste de triplets "NoeudDeDépart NoeudDArrivée Dette" représentant chaque arc du graphe. Chaque nœud est identifié par son nom et la dette est un réel.

Voici un exemple de fichier :

```
20
CI 40
CP 80
Agro 120
CdS 20
CePHA 150
CD 340
CM 250
CPSY 140
C$ 40
CGEO 70
...
CM CP 300
CP CI 60
...
C$ CdS 220
CI CePha 150
```

La liste des arcs se finit lorsque le fichier se termine. *Le graphe n'est pas forcément un graphe complet.*

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 (pensez aussi aux méthodes statiques). Vous pouvez supposer que le fichier d'entrée ne comporte pas d'erreur.

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

```
~>java dettes cercles-ULB.txt

Cycles supprimés:

1) reduction de 130
CP (200)-> C$ (220)-> CdS (130)-> CP (200)-> ...
nouvelle situation :
CP (70)-> C$ (90)-> CdS

2) reduction de 120
CI (120)-> CePha (140)-> CI (120)-> ...
nouvelle situation :
CePha (20)-> CI

Fichier dettesNoCycles.gv contient la situation sans cycles.
Utilisez la commande : <Commande graphviz pour créer le fichier png>
pour créer l'image.

Ordre des remboursements :

CM (250)-> CP (Il reste 50 à rembourser)
CePha (20)-> CI
CP (70)-> C$
C$ (90)-> CdS
CP (60)-> CI

Fichier dettesRemb.gv contient la situation actuelle.
Utilisez la commande : <Commande graphviz pour créer le fichier png>
pour créer l'image.
```

Utilisez les exceptions dans vos algorithmes en cas de problèmes (et évitez les retours de booléens). Votre code source sera évidemment accompagné d'un `makefile` permettant de compiler facilement votre projet.

Le projet peut être réalisé individuellement ou par groupe de deux (ce que nous recommandons).

Tout cela sera naturellement présenté dans un *rapport* à la présentation soignée. Pour rappel, un rapport est un *texte suivi en français correct* qui *décrit* votre travail. Cela va sans dire, le rapport fait partie *intégrale* de votre travail et une partie importante de la note finale portera sur le rapport (y compris sa présentation).

Consignes pour la remise du projet

À respecter scrupuleusement !

1. Les modalités pour la remise du rapport sont :
   - Date : **le 20 Décembre 2013** (vous pouvez ne pas remettre votre projet dans le cas de fin du monde).
   - Lieu : **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 (plus un point par jour de retard). Les projets en retard doivent être déposés **dans la caisse** prévue à cet effet près du secrétariat.
2. Les modalités pour la remise du code sont :
   - 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.
   - L'archive doit être envoyée à `lporrett@ulb.ac.be` **et** `nveshchi@ulb.ac.be`.
   - Le **sujet** du mail doit être **exactement** `Algo2-Projet2013`.

   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 !