INFO–F–203 — Algorithmique 2
Projet 2: Manipulation de graphes

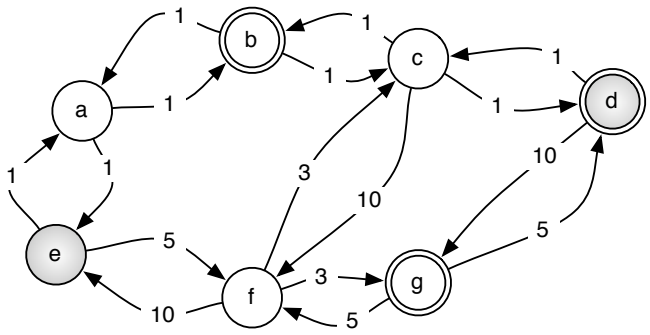
Gilles GEERAERTS Frédéric PLUQUET

Année académique 2006–2007

Le problème

Tout le monde a déjà rencontré le problème suivant au moins une fois dans sa vie : vous désirez retrouver un ami, se trouvant à un autre endroit que vous, le plus rapidement possible, où que ce soit. La solution la plus simple employée par le plus grand nombre est de se donner un point de rendez-vous, plus ou moins à égale distance entre votre ami et vous.

Considérons une ville, dont le plan est schématisé sous forme d'un graphe, comme celui qui se trouve ci-dessous. Chaque nœud représente un lieu particulier de la ville, et les arcs entre les nœuds représentent les routes qui les relient. Chaque arc est pondéré : le poids d'un arc est le temps nécessaire pour parcourir la rue à pied (comme certaines rues sont en pente, il est parfois plus rapide de les parcourir dans un sens que dans l'autre).



Certains lieux sont des points de rendez-vous possibles : ils ont été représentés par des nœuds doublement entourés. Les positions initiales des deux personnes sont grisées. Il est à noter qu'une position initiale peut être un point de rendez-vous (c'est le cas pour d).

Ayant à disposition la carte d'une ville, avec toutes les informations décrites plus haut et la position initiale des deux personnes, on vous demande de produire deux algorithmes, qui calculent un point de rendez-vous optimal, selon les critères suivants :

- Le premier algorithme calculera le point qui minimise la somme des temps de parcours des deux personnes. Dans l'exemple ci-dessus, le point de rendez-vous optimal pour ce critère est le point b, car la première personne peut l'atteindre par le chemin $e > a > b$, et la seconde par le chemin $d > c > b$. Le temps total est de *4*.
- Le second algorithme calculera le point qui minimse la somme du nombre de routes parcourues par les deux compères. S'il existe plusieurs points qui satisfont ce critère, il en choisira un qui minimise la somme des temps de parcours. Dans l'exemple ci-dessus, le point de rendez-vous minimisant le nombre de routes parcourues est d ($e > f > c > d$ et d) avec *3* routes parcourues et un temps total de *9*. On peut remarquer que le nœud g offre également un nombre de routes égal à *3* mais mais le temps total est de *18* ($e > f > g$ et $d > g$). La première solution sera donc choisie.

Indications pour la réalisation de l'algorithme

Les algorithmes de plus court chemin que vous avez étudiés au cours permettent de calculer un plus court chemin entre deux points donnés. Dans le cas présent, on vous demande de minimiser, non pas la longueur d'un seul chemin, mais la somme des longueurs de deux chemins. De plus, le point d'arrivé n'est pas donné (vous devez le déterminer). On ne peut donc pas appliquer tels quels les algorithmes classiques.

Il est par contre possible de construire, à partir des données décrites ci-dessus, un nouveau graphe grâce auquel ces algorithmes peuvent être exploités. Dans ce nouveau graphe, chaque sommet est une paire de sommets du graphe d'origine, et exprime la position de chacune des deux personnes. Par exemple, le sommet initial du nouveau graphe sera (e, d) puisque la première personne est initialement en e, et la seconde, initialement en d. Les arcs du nouveau graphe correspondent au déplacement d'un des deux compères. Par exemple, du sommet initial, il y a un arc pondéré par 5 qui va vers (f, d) (correspondant au déplacement de la première personne de e vers f, l'autre restant sur place) ; mais aussi un arc allant vers (e, c) ou (e, g), etc.

En construisant ce nouveau graphe, vous verrez qu'on peut extraire, de chacun de ses chemins, un chemin pour la première et un chemin pour la seconde personne, dans le graphe de départ. La longueur du chemin dans le nouveau graphe sera la somme des longueurs de ces deux chemins. Vous verrez également que certains nœuds correspondent à la rencontre des deux personnes...

Dans le rapport de votre projet, vous veillerez à expliquer soigneusement quel est l'algorithme que vous appliquerez sur le nouveau graphe, et comment le résultat qu'il fournit vous permet de trouver la solution aux problèmes posés ici.

Détails techniques

Ces algorithmes devront être implémentés en C++ 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 `g++`.

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 de :

- le nombre de nœuds du graphe (un entier) ;
- le nom de chaque nœud (un caractère alphanumérique uniquement), suivi d'un retour à la ligne ;
- le nombre de points de rendez-vous (un entier) ;
- le nom de chaque nœud qui est un point de rendez-vous (un caractère alphanumérique uniquement), suivi d'un retour à la ligne ;
- les deux noms des nœuds de départ (suivis tous deux par un retour à la ligne) ;
- une liste de triplets "NoeudDeDépart NoeudDArrivée Temps" représentant chaque arc du graphe. Chaque nœud est identifé par sont caractère alphanumérique et le temps est un entier.

Voici un exemple de fichier qui correspond au graphe de l'exemple :

```
7
a
b
c
d
e
f
g
3
b
d
g
d
e
a b 1
b a 1
b c 1
e f 5
...
```

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 (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 et le numéro du critère (1 ou 2) en argument et écrire sur l'output standard la solution de la manière suivante :

```
litpc14:~/projet2 fred$ ./monProjet graphe.txt 1
Le point optimal de rendez-vous est : b
Le premier chemin est : d > c > b
Le second chemin est : e > a > b
Le temps total est : 4


litpc14:~/projet2 fred$ ./monProjet graphe.txt 2
Le point optimal de rendez-vous est : d
Le premier chemin est : d
Le second chemin est : e > f > g > d
Le nombre de route est de : 3
Le temps total est : 13
```

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 une *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 (au moins 20%) 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 4 mai**.
   - Lieu : **au Secrétariat « étudiants » du Département d'Informatique, local 2N8.104**.
   - Heure : **Avant 16h**.

   Le secrétariat ferme à 16h. **Après 16h**, les projets sont considérés comme **en retard**, et vous perdez **1 point** 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.P2.NomEtudiant1-NomEtudiant2.zip`, si deux étudiants ont réalisé le projet ; `AG2.P2.NomEtudiant.zip` dans le cas contraire.
   - L'archive doit être envoyée à `fpluquet@ulb.ac.be`.
   - Le **sujet** du mail doit être **exactement** `AG2P2`.

   Si vous ne respectez pas ces critères, nous considérerons que vous n'avez pas rendu de code.

Bon travail !