Esiee-Paris - unité d'algorithmique - Second projet

11 avril 2021 – R. Natowicz, I. Alame, D. Garrigue

On se donne un graphe G à n sommets. Tout sommet i de ce graphe envoie au moins un arc vers un sommet de numéro supérieur et n'envoie aucun arc vers un sommet de numéro inférieur. L'arc $i \to j$ est de coût c_{ij}.

Le coût d'un chemin est la somme des coûts de ses arcs. On s'intéresse à deux sortes de chemins : les chemins de coût minimum et les chemins obtenus par minimisation locale.

1) Chemin de coût minimum c^* allant du sommet 0 au sommet $n-1$. Tout autre chemin de 0 à $n-1$ est de coût supérieur ou égal à c^*.

2) Chemin $0 \to .... \, i \to j \, ... \, n-1$ localement minimum. Tout arc $i \to j$ de ce chemin est le premier arc sortant de i dont le coût est minimum : $i \to j \; = \; \arg\min_{j' \in S(i)} \; c_{ij'}$ où $S(i)$ est l'ensemble des sommets vers lesquels le sommet i envoie un arc.

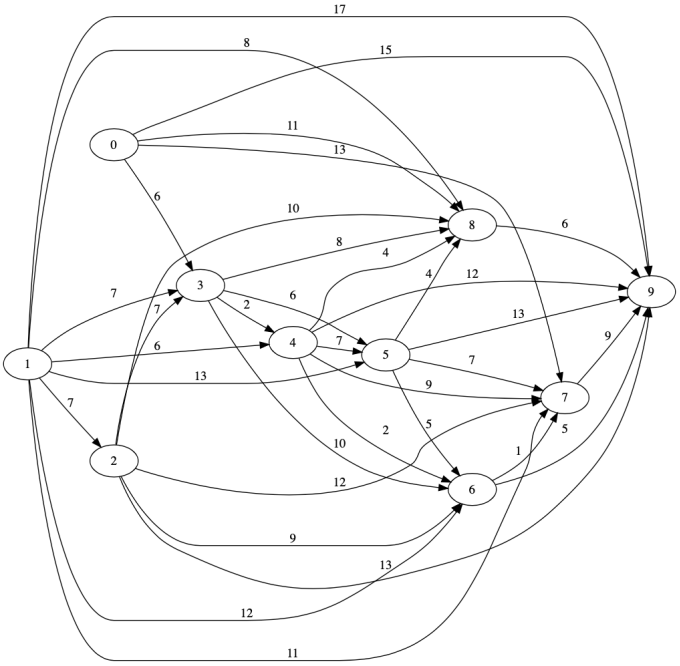
On demande :

1. de calculer un chemin de coût minimum allant du sommet 0 au sommet $n-1$. La totalité du calcul doit être en $\Theta(m+n)$ où m est le nombre d'arcs et n le nombre de sommets du graphe. Exemples de temps de calcul sans affichage (processeur : 2,5 GHz Intel Core i5 double cœur) :

```
% time java -Xms2g CCM 1000 0
java -Xms2g CCM 1000 0  0,36s user 0,06s system 156% cpu 0,265 total
% time java -Xms2g CCM 2500 0
java -Xms2g CCM 2500 0  1,09s user 0,14s system 166% cpu 0,732 total
% time java -Xms2g CCM 5000 0
java -Xms2g CCM 5000 0  3,65s user 0,30s system 151% cpu 2,611 total
% time java -Xms2g CCM 7500 0
java -Xms2g CCM 7500 0  8,76s user 0,74s system 181% cpu 5,252 total
% time java -Xms2g CCM 10000 0
java -Xms2g CCM 10000 0  20,33s user 2,18s system 159% cpu 14,084 total
```

2. de comparer, par validation statistique, les coûts minimum aux coûts des chemins obtenus par minimsation locale.

Exemple d'exécution du programme sur le graphe ci-dessous :




```
graphe G :
0 : (3,6) (7,13) (8,11) (9,15)     <<< arc 0  --> 3 de coût 6, arc 0 --> 7 de coût 13, etc.
1 : (2,7) (4,6) (9,17) (8,8) (3,7) (7,11) (6,12) (5,13)
2 : (6,9) (9,13) (3,7) (7,12) (8,10)
3 : (6,10) (4,2) (8,8) (5,6)
4 : (6,2) (7,9) (5,7) (8,4) (9,12)
5 : (6,5) (7,7) (9,13) (8,4)
6 : (7,1) (9,5)
7 : (9,9)
8 : (9,6)
9 :
M = [0, 1073741823, 1073741823, 6, 8, 12, 10, 11, 11, 15]   <<< M[0:n] de t.g. M[j] = m(j) = coût min. d'un chemin de 0 à j
A = [0, 1, 2, 0, 3, 3, 4, 6, 0, 6].  <<<  A = arg M
Coût d'un chemin de coût minimum jusqu'en 9 : 15
0--(6)-->3--(2)-->4--(2)-->6--(5)-->9
affichage des chemins de coût minimum de 0 à tous les sommets :
Il n'y a pas de chemin de 0 à 1
Il n'y a pas de chemin de 0 à 2
0--(6)-->3 coût = 6
0--(6)-->3--(2)-->4 coût = 8
0--(6)-->3--(6)-->5 coût = 12
0--(6)-->3--(2)-->4--(2)-->6 coût = 10
0--(6)-->3--(2)-->4--(2)-->6--(1)-->7 coût = 11
0--(11)-->8 coût = 11
0--(6)-->3--(2)-->4--(2)-->6--(5)-->9 coût = 15
Description du graphe dans le fichier g.graphviz   <<< l'image du graphe ci-dessus a été générée par le logiciel graphViz
Coût par minimisation locale = 20   <<< coût du chemin de 0 à 9 obtenu par minimisation locale
Validation statistique à 10000 runs ..........
Médiane des distances relatives : 1,428571   <<< en moyenne le coût  par minimisation locale est plus de 42% supérieur au coût minimum.
Max des distances relatives : 4,444445. <<< dans le pire cas il est plus de 4 fois supérieur.
```

Compléter l'ébauche du programme CCM.java et donner les résultats de son exécution pour des graphes à 10 et 100 sommets, avec validation statistique sur 10 000 *runs*.

Les noms du binôme ou du monôme seront en commentaire en première ligne du programme CCM.java ; les résultats de l'exécution seront en commentaire `/* ... */` à la fin du fichier CCM.java ; le ficher CCM.java sera mis dans un dossier dont le nom sera le nom du monôme ou les noms du binôme ; ce dossier sera compressé en un fichier `.zip` ; ce dossier *zippé* sera envoyé avant vendredi 7 mai par courrier électronique à `rene.natowicz@esiee.fr`, l'objet du email étant "Algorithmique : projet2".

Pour l'évaluation du projet, votre programme CCM.java sera compilé et exécuté dans un terminal Unix.