Graphes : plus court chemin

Le but de ce TD est de calculer la longueur d'un plus court chemin entre les sommets d'un graphe (orienté, non valué). Celui-ci est représenté par sa matrice d'adjacence (puis dans l'exercice facultatif par le tableau de listes chaînées des successeurs). On utilise le matériel de la PC (aussi dans le poly). Appelez-nous pour des précisions.

1. Récupérer et exécuter le fichier *CodeTd9.java* ci-dessous. Que fait-il ?
2. Reprendre et terminer les Exercices 2 et 3 du TD 8.
3. On veut maintenant calculer la longueur d'un plus court chemin de i à j (pour tous les couples *i,j* de sommets du graphe). On se base sur le principe suivant (voir PC). On partitionne l'ensemble des chemins de i à j : ceux n'utilisant aucun sommet intermédiaire, puis ceux n'utilisant que le sommet 0, puis que 0 et 1, etc., puis 0 à k, etc., jusqu'à tous les sommets.

   En notant $lpcc(i,j,k)$ la longueur d'un plus court chemin de i à j n'utilisant que les sommets 0 à k, on a : $lpcc(i,j,k) = min(lpcc(i,j,k-1), lpcc(i,k,k-1) + lpcc(k,j,k-1))$.

   On stockera ces longueurs successives dans une matrice, initialisée par la matrice d'adjacence. Utilisez un objet *Graphe*, cette matrice pouvant être considérée comme un graphe valué par les longueurs. Une seule matrice suffit, on la modifie à chaque étape. Enfin, pourquoi se baser sur les numéros des sommets ? Rien de profond, c'est simplement une technique pour les énumérer.

   (a) Définissez la matrice initiale, variante de la matrice d'adjacence adaptée à l'algorithme. Appelez-nous pour des précisions.
   (b) Ecrivez l'instruction affectant à la case *i,j* la longueur de l'étape k en fonction des valeurs de l'étape *k-1* (sans compiler).
   (c) Pour un k fixé, écrivez le calcul de la nouvelle matrice à partir de l'ancienne (sans compiler).
   (d) En déduire le calcul de la matrice des longueurs des plus courts chemins. Testez.

Facultatif

4. **Représentation d'un graphe par "tableau de successeurs"** Pour un graphe à n sommets, on utilise un tableau à n cases. La case i contient une liste chaînée dont chaque cellule contient l'un des successeurs de i.

   (a) Déclarer la classe correspondante.
   (b) Créer un graphe constant "à la main". (Puis en faire le dessin, dessiner sa matrice d'adjacence et faire intuitivement la correspondance.)
   (c) L'afficher dans le "format" de la matrice d'adjacence : en ligne i, colonne j, on affiche 1 s'il y a un arc de i à j, et 0 sinon.

Code et corrigé : `http://www.lri.fr/~waller/COURS/POLYTECHNIQUE/TC/TD9` ou fichier à copier `~waller/CodeTd9.java`

```java
import java.io.*;

class Graphe {
    int nb_som;
    int mat[][];

    Graphe (int n){
        nb_som = n;
        mat = new int[n][n];
    }
}

public class CodeTd9 {

    public static String Lire () {
        ... comme d'habitude ...
    }

    public static Graphe saisit () {
        ...
    }

    public static void affiche (Graphe a) {
        ...
    }

    public static void f (Graphe a) {
        for (int i=a.nb_som-1; i>0; i=i-1) {
            int j = 0;
            while (j<a.nb_som && a.mat[i][j]==0)
                j = j + 1;
            if (j==a.nb_som)
                System.out.println(-1);
            else
                System.out.println(j);
        }
    }

    public static void main (String[] args) {
        Graphe g = saisit();
        affiche(g);
        f(g);
    }
}
```