

- TP 5. Plus courts chemins entre tous couples. Algorithme de Floyd-Warshall -

Le but de ce TP est de calculer l'ensemble des plus courts chemins entre tous les couples de sommets d'un graphe orienté $D = (V, A)$, puis d'appliquer ce calcul à la recherche de la fermeture transitive d'un graphe orienté.

Langage. Programme en C++. Votre programme pourra contenir :

```
#include <iostream>
#include <vector>
using namespace std;

const int n=5;
const int inf=9999;           //La valeur infinie.

int main(){

    int longueur[n][n]={0,2,inf,4,inf}, //Les longueurs des arcs.
                    {inf,0,2,inf,inf}, //longueur[i][j]=inf si l'arc ij n'existe pas
                    {inf,inf,0,2,inf},
                    {inf,-3,inf,0,2},
                    {2,inf,inf,inf,0}};

    int dist[n][n];           //Le tableau des distances.
    int chemin[n][n];        //Le tableau de la premiere etape du chemin de i a j.

return 0;
}
```

Ce début de code est récupérable là : <http://www.lirmm.fr/~bessy/GLIN501/TP/tp5.cc>

!!!! Pensez à tester chaque code produit sur de petits exemples!!!!

- Exercice 1 - À quoi ça sert, les cours de graphes ?

On trouve à cette adresse : <http://about-france.com/france-rail-map.htm> un plan des principales lignes de train en France. Essayer de trouver un itinéraire nécessitant plus (strictement) que deux changements et trouver le trajet correspondant sur le site <http://www.voyages-sncf.com/>. À titre d'exemple, on pourra essayer *Dinan-Mende* ou *Lons le Saunier-Guéret*.

Faites la même requête sur le site <http://www.bahn.de/i/view/FRA/fr/index.shtml> ...

- Exercice 2 - Floyd-Warshall.

Initialiser le tableau **dist** à **longueur**.

Écrire une fonction `void floyd_warshall(int longueur[][n], int dist[][n])` qui construit le tableau **dist** dont chaque entrée **dist**[*i*][*j*] est la longueur minimale d'un chemin de *i* à *j*, et vaut *inf* si un tel chemin n'existe pas.

Afficher ensuite le tableau **dist**, et vérifier la solution obtenue (à la main...).

- Exercice 3 - Calcul des chemins.

Initialiser le tableau **chemin** de telle sorte que :

- **chemin**[*i*][*j*] = *j* lorsque *ij* est un arc.

- $\text{chemin}[i][j] = -1$ dans les autres cas.

Modifier ensuite la fonction `floyd_warshall()` de sorte que lors du calcul du tableau `dist`, le tableau `chemin` soit aussi calculé, `chemin[i][j]` devant contenir le voisin sortant de i le long d'un plus court chemin de i à j , ou -1 si un tel chemin n'existe pas.

Afficher ensuite le tableau `chemin`, et vérifier qu'il soit correct (à la main...).

- Exercice 4 - Calcul d'un itinéraire.

Ecrire une fonction `void itineraire(int i, int j, int chemin[][n])` qui prenant en entrée deux sommets du graphe affiche un plus court chemin de i à j . L'appel à cette fonction affichera :

```
Entrer le depart : 2
Entrer la destination : 4
L'itineraire est : 2 3 4
```

Faire de même avec le réseau routier codé là : <http://www.lirmm.fr/~bessy/GLIN501/TP/villes.txt>

- Exercice 5 - Fermeture transitive.

La *fermeture transitive* d'un graphe orienté D est une matrice `fermeture` vérifiant `fermeture[i][j] = 1` s'il existe un chemin orienté de i à j dans D , et `fermeture[i][j] = 0` sinon.

En vous inspirant de la fonction `floyd_warshall()`, écrire une fonction `void fermeturetransitive(int arc[][n], int fermeture[][n])` qui calcule le tableau `fermeture`, le tableau `arc` étant la matrice d'adjacence d'un graphe orienté D .

Tester votre fonction sur l'exemple suivant :

```
const int n=6;
int arc[n][n]={0,0,0,1,0,1},//La matrice d'adjacence du graphe oriente D.
             {1,0,1,1,0,0},
             {0,0,0,1,0,0},
             {0,0,0,0,1,1},
             {0,0,1,0,0,1},
             {0,0,1,0,0,0}};
int fermeture[n][n];          // La matrice de la fermeture transitive de D.
```

- Exercice 6 - Composantes fortement connexe.

Ecrire une fonction `void compfortconnexe(int n, int fermeture[][n])` qui affiche les composantes fortement connexes du graphe D . Le résultat pourra être de la forme :

Les composantes fortement connexes sont : {1,4}, {2}, {3,5,6}, {7}

- Exercice 7 - Pour aller plus loin.

Concernant l'algorithme de Floyd-Warshall :

- Augmenter la taille de l'entrée, en tirant aléatoirement un graphe orienté D sur 100 sommets tel que tout sommet possède exactement deux arcs sortants, chacun de longueur 1. Calculer quelques itinéraires.
- Calculer $\text{diam}(D) = \max\{\text{dist}(u, v) : u, v \in V\}$, le diamètre orienté du graphe D . Faire plusieurs essais, et noter la proportion de graphes fortement connexes.
- Quelle valeur minimale peut théoriquement atteindre le diamètre orienté du graphe D ?

Concernant le calcul de la fermeture transitive :

- Engendrer un graphe orienté aléatoire sur $n = 500$ sommets et 500 arcs, puis calculer les tailles de ses composantes fortement connexe. Que peut-on dire de ce graphe?
- Même question avec 500 sommets et 1000 arcs.
- Afficher les composantes fortement connexes de telle sorte que si une composante C apparaît avant C' , il n'existe pas d'arc de C' vers C .