

## - TP 5. Plus courts chemins entre tous les 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  $D = (V, A)$ . Un tableau **longueur** donne la longueur des arcs de  $D$ , avec la convention que **longueur** $[i][i] = 0$  pour tout sommet  $i$ , et **longueur** $[i][j] = \text{inf}$  lorsqu'il n'existe pas d'arc de  $i$  à  $j$ .

**Langage.** Programme en c++. Votre programme pourra commencer par :

```
#include <cstdlib>
#include <iostream>
#include <vector>
#include <fstream>
using namespace std;
const int n=5;
const int inf=99; //La valeur infinie.
int longueur[n][n]={{0,1,inf,2,inf}, //Les longueurs des arcs.
                  {inf,0,1,inf,inf},
                  {inf,inf,0,1,inf},
                  {inf,-1,inf,0,1},
                  {1,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.
```

Vous trouverez une entête à votre algorithme à l'adresse : <http://www.lirmm.fr/~thomasse/cours/TP/tp5.txt>

### - Exercice 1 - Le graphe.

Représenter le graphe orienté correspondant au TP.

### - Exercice 2 - Floyd-Warshall.

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

Puis, écrire une fonction void floyd\_warshall() qui construit le tableau **dist** dont chaque entrées **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 qu'il correspond bien au graphe.

### - Exercice 3 - Calcul des chemins.

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

- **chemin** $[i][j] = j$  lorsque  $ij$  est un arc.
- **chemin** $[i][i] = -1$  pour tout  $i$ . Ici  $-1$  remplace NULL.
- **chemin** $[i][j] = \text{inf}$  dans les autres cas.

Modifier ensuite la fonction `floyd_warshall()` de sorte que lors du calcul du tableau **dist**, le tableau **chemin** soit lui aussi réactualisé.

Afficher ensuite le tableau **chemin**, et vérifier qu'il correspond bien au graphe.

**- Exercice 4 - Calcul d'un itinéraire.**

Ecrire une fonction `void itineraire(int i, int j)` 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

**- Exercice 5 - Pour aller plus loin.**

- 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 le diamètre orienté du graphe  $D$ . Est-il fortement connexe?
- Effectuer des changements locaux, i.e. à chaque étape changer un voisin sortant d'un sommet  $v$ , afin de transformer  $D$  en un graphe fortement connexe. Puis afin de minimiser le diamètre.
- Quelle valeur minimale peut théoriquement atteindre le diamètre orienté du graphe  $D$ ?