Graphes, combinatoire et probabilités discrètes
Projet - Algorithme de Floyd-Warshall

Lespagnol Killian, Loya Dylan

15 Janvier 2023

Table des matières

1 Algorithme de Floyd-Warshall

L'algorithme de Floyd-Warshall est un algorithme pour déterminer les distances des plus courts chemins entre toutes les paires de sommets dans un graphe orienté et pondéré, en temps cubique au nombre de sommets. Cet algorithme construit une matrice de taille égale au nombre de sommets du graphe. La diagonale de cette matrice est initialisé à 0 car pour aller d'un sommet à lui même, on considère que la distance est nulle. Puis pour chaque arc du graphe on écrit sa taille dans la matrice entre les bons sommets et enfin on définit une valeur pour les cases pour lesquelles les sommets ne sont pas reliés entre eux par un arc. Au fur et à mesure de l'algorithme on va autoriser le fait de passer par différents sommets, si en passant par ce sommet on trouve un chemin plus court que le chemin existant déjà dans la matrice, alors on le remplace et on spécifie, dans une autre matrice, le sommet intermédiaire par lequel on est passé. On a ainsi, à la fin de l'exécution de l'algorithme, deux matrices :

— Une matrice contenant les poids minimaux pour aller de n'importe quel sommet à un autre.
— Une matrice permettant de retrouver le chemin de poids minimal de n'importe quel sommet à un autre.

Dans le code java cela se traduit par une classe contenant un tableau de tableau d'entier (on considère que les poids des arcs sont des nombres entiers). Les deux dimensions de ce tableau sont de la taille du nombre de sommets de la matrice sur laquelle ont souhaite agir. À l'initialisation on remplit ce tableau avec une valeur spéciale indiquant l'absence de chemin (ici la valeur maximale possible pour les entiers en java). Puis grâce à une liste fournie indiquant la distance entre 2 sommets, on remplace les cases du tableau correspondantes. On a ainsi notre matrice de poids initialisée.

2 Comment compiler et exécuter le code

Le projet a été écrit, compilé et testé avec le SDK de Java version 16. Un fichier **.jar** est fourni, permettant de lancer simplement le programme. Pour l'utiliser il faut se placer dans le dossier racine et taper la commande :

```
java -jar exe/tp5.jar
```

Si toutefois vous souhaitez compiler le projet vous même, vous pouvez en vous plaçant dans le dossier racine et en tapant la commande :

```
javac ./**/*.java -encoding ISO-8859-1 -d out/production
```

et enfin vous pouvez exécuter le programme en tapant la commande suivante :

```
java -Dfile.encoding=ISO-8859-1 -classpath out/production src.Main
```

3 Mode d'emploi

Le programme utilise les fichiers dans le dossier **res** (pour ressources) pour fonctionner. Au lancement, le programme utilise le fichier **graph-001.alists** pour construire les différentes matrices dont il a besoin, puis il utilise le fichier **fr-dept-distances.data** pour mettres les bonnes valeurs aux bons endroits dans les matrices. Le fichier **paths.command** est le fichier contenant les numéros des départements dont on veut trouver le chemin. C'est ce fichier que vous devez modifier en fonction de ce que vous voulez obtenir comme résultat, en plaçant des couples de nombres entier représentant les numéros des départements.

Toutes les sorties du programme se font dans le dossier **fw_out** *sauf* les chemins demandés par l'utilisateur dans le fichier **res/paths.command** qui eux, s'affichent en console. Après une exécution normale du programme, il y aura 4 fichiers dans le dossier **fw_out** :

— **all_paths.txt** est un fichier contenant tout les chemins possibles entre tout les sommets.
— **graph-001.costs** est la matrice des coûts calculée.
— **graph-011.paths** est la matrice permettant de retrouver les chemins
— **res.amatrix** est la matrice construite au début du programme grace au fichier **.alist** fourni.