TP2 – Programmation dynamique

Le but de ce TP est d'implémenter l'algorithme de Floyd-Warshall.

1 Prise en main du programme

Téléchargez l'archive https://perso.liris.cnrs.fr/tpierron/RO2023/TP2-etu.zip. Elle contient :

- Une bibliothèque de gestion des graphes **pondérés** `graph.cpp` `graph.hpp`.
- Un `Makefile`, permettant de compiler le programme en tapant la commande `make` dans un terminal.
- Un sous-dossier `data/` contenant des graphes (un par fichier) au format suivant :
  - la première ligne du fichier contient le nombre n de sommets dans le graphe,
  - toutes les autres lignes du fichier contiennent une arête du graphe sous la forme de deux entiers u v séparés par un espace, ou u et v sont les identifiants des deux sommets (distincts) qui sont les extrémités de l'arête,
  - l'identifiant d'un sommet est un entier compris entre 0 et $n-1$.
- Un fichier à compléter `main.cpp`.

Le seul fichier que vous avez à modifier est `main.cpp`. Il n'est pas nécessaire de changer les autres.

Question 1. Observer la fonction `main`. Comment les poids des arêtes sont-ils gérés ?

Question 2. Implémenter la fonction `random_clique` qui prend en entrée un entier n et renvoie une clique de taille n dont les arêtes sont pondérées par des entiers aléatoires entre 1 et n.

2 Algorithme de Floyd-Warshall

On cherche à implémenter l'algorithme de Floyd-Warshall vu en cours. Pour rappel, il s'agit de calculer un tableau tridimensionnel D tel que $D[i][j][k]$ contient la longueur d'un plus court chemin de i à j dont les sommets internes ont une étiquette inférieure à k.

Question 3. Rappeler la relation de récurrence vérifiée par D.

Question 4. Implémenter la fonction `floyd_warshall`, qui renvoie une matrice M telle que $M[i][j]$ contient la distance de i à j.

Question 5. Quelles sont les complexités temporelles et spatiales de votre algorithme ?

Question 6. Améliorer l'algorithme pour qu'il utilise une complexité spatiale quadratique.

3 Calcul des plus courts chemins

Pour calculer les plus courts chemins, on retient un tableau P tel que $P[i][j][k]$ contient le sommet qui précède j dans un plus court chemin de i à j dont les sommets internes ont une étiquette inférieure à k.

Question 7. Quelle relation de récurrence est satisfaite par P ? Implémenter la fonction `floyd_warshall2`, qui renvoie deux matrices M_d et M_p telles que $M_d[i][j]$ contient la distance de i à j, et $M_p[i][j]$ contient le sommet précédant j dans un plus court chemin de i à j.

Question 8. Peut-on à nouveau améliorer la complexité spatiale de cette fonction ?

Question 9. Compléter la fonction `get_path` qui, étant donné la matrice M_p et deux indices i, j, retourne le plus court chemin de i à j.