EFREI – 2013/2014 – L3
Théorie des Graphes – TP #3

Calcul d'arbre couvrant de poids minimum

Travail à effectuer en binôme.

Travail à effectuer

Vous devez mettre en œuvre un programme qui :
- lit les arêtes d'un graphe dans un fichier texte et en sauvegarde le contenu dans une structure de données de votre choix ;
- exécute un algorithme pour calculer un arbre couvrant de poids minimum.
  Le choix de l'algorithme dépend de ce que vous avez réussi à faire en TP2. Si vous avez réussi à mener à bien la détection d'un cycle, il vaut mieux choisir le Kruskal constructif ; si c'est la détection de la connexité, vous pouvez choisir le Kruskal destructif ; si ni l'un ni l'autre ne marchent, il ne vous reste que Prim.

Votre code doit afficher les décisions prises par l'algorithme lors de son exécution :
- arête ajoutée ou supprimée à chaque étape ;
- arbre couvrant final (sous forme de matrice d'adjacence ou de liste des arêtes) et poids calculé.

Graphes

Il s'agit de graphes non orientés, valués et connexes.
Les sommets sont représentés par les nombres entiers de 0 à N-1 pour un graphe comportant N sommets.
Les valeurs associées aux arcs sont des nombres entiers quelconques.

Fichier de données

Basez-vous sur ce que vous avez fait pour les TP précédents.

Rappel sur les méthodes de calcul

Soit G=<S,A> pour lequel on recherche un arbre couvrant de poids minimal G'=<S',A'>.

Kruskal constructif
- Initialisation :
  - A' ← ∅
  - S' ← S
- Itérations :
  - tant que G' n'est pas un arbre couvrant,
    - on sélectionne dans A une arête (x,y) de plus faible poids telle que le graphe <S',A'∪{(x,y)}> ne contienne pas de cycle
    - on ajoute (x,y) à A'

Kruskal destructif
- Initialisation :
  - A' ← A
  - S' ← S
- Itérations :
  - tant que G' n'est pas un arbre,
    - on sélectionne dans G' une arête (x,y) de plus fort poids telle que le graphe <S',A'-{(x,y)}> reste connexe

- on supprime (x,y) de A'

Prim

Initialisation :

S' ← { sommet quelconque de S }

A' ← ∅

Itérations :

tant que G' n'est pas un arbre couvrant,

- on identifie l'ensemble des arêtes (xi,yi) de A telles que xi∈S', yi∉S'
- on sélectionne parmi ces arêtes celle, notée (x,y), qui a le plus faible poids
- on ajoute y à S' et (x,y) à A'

Remarques utiles pour la mise en œuvre :
Soit N le nombre de sommets de G et P le nombre d'arêtes dans G.

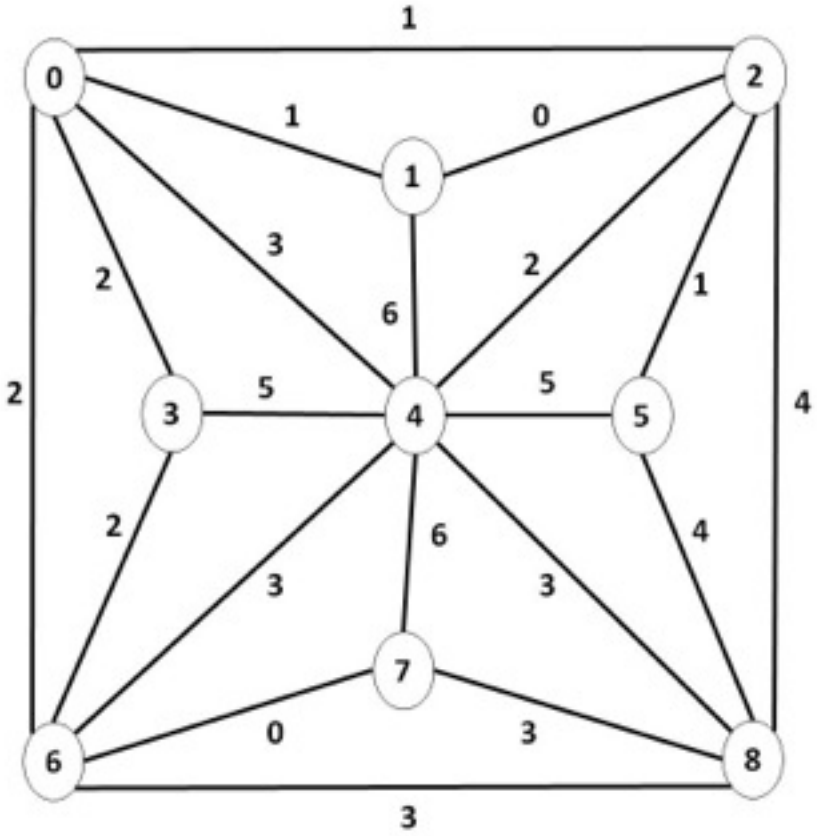
- dans G', il y aura N-1 arêtes.
- N-1 = P – (P – N + 1)

Langage de programmation

Comme pour les TP précédents.

Rendu du travail

Vous devrez exécuter votre programme sur le graphe ci-dessous :



Rendu du travail : 24 heures après la fin de séance.
Envoi par email à l'adresse qui vous sera donnée en séance.
Sujet du mail : EFREI – L3 – TG – TP3 – *nom1 nom2*
Corps du message : vide
Pièces jointes :

Un fichier au format .txt contenant les traces d'exécution de votre programme.

Autant de fichier de code source (.c, .cpp, .h) que nécessaire.
Les fichiers de données utilisés par votre programme pour lire les graphes de test.
Pas d'archive. Pas de fichier binaire. Pas de « fichier projet » géré par votre outil de développement.
Tous les fichiers doivent avoir un nom préfixé par les noms des deux binômes, par exemple « barbot-velikson-tp3.cpp », « barbot-velikson-tp3-graphe1.txt », ….

Tout manquement à ces consignes sera pénalisé.

Eléments de notation

Seront pris en compte :

- Votre code lui-même : choix des structures de données, algorithmes choisis et mise en œuvre de ces algorithmes.
- Son adaptabilité à d'autres graphes que ceux demandés en traces d'exécution.
- Les commentaires, et plus généralement la clarté de votre code.
- Les traces fournies.

Toute récupération évidente, même partielle, de code fait par un autre groupe se verra sanctionnée par un 0/20 à tous les étudiants concernés.