EFREI 2015/2016 -- L3/L'3 – Théorie des Graphes -- TP #1

A réaliser en trinôme. Une équipe de 4 sera autorisée pour un groupe TD de 3n+1 étudiants, et un binôme sera autorisé pour un groupe TD de 3n+2 étudiants.
Langages autorisés : C / C++

Préambule

Le travail de réalisation que vous aurez à effectuer s'étalera sur les séances de TP et de Projet. Les mêmes équipes doivent être constituées pour l'ensemble du travail. Aucun changement ne sera toléré sans avoir été préalablement autorisé par les enseignants.

Il n'y aura qu'une seule note associée au livrable final (projet).

Bien entendu, tout plagiat de code évident entre binômes différents sera sanctionné par un '0/20' à tous les binômes concernés (nous ne chercherons pas à savoir qui a travaillé et qui a récupéré le code). Cette décision sera sans appel.

Ecrivez vos algorithmes d'abord en pseudo-langage (que vous laisserez en commentaire dans votre code) afin que votre enseignant puisse vous aider plus rapidement et efficacement en cas de besoin.

Graphes à prendre en compte

Votre programme doit être capable de prendre en compte n'importe quel graphe orienté valué, sachant que :

- Les sommets sont des numéros entiers de '0' à 'n-1' pour un graphe contenant n sommets ;
- Les valeurs associées aux graphes sont des nombres entiers quelconques ;
- Il y a au plus un arc reliant deux sommets ;
- Il n'y a pas de boucle ;
- Il peut y avoir des sommets isolés (sans prédécesseur ni successeur).

Mise en œuvre d'un graphe en mémoire / Lecture du graphe sur fichier

Vous commencerez par choisir une mise en œuvre de graphes en mémoire.
Lorsque ce choix a été fait, et que les définitions appropriées sont disponibles dans votre programme, vous développerez le code nécessaire à la lecture d'un graphe sur fichier.

Aide disponible :

- Sur campus, un document donne des indications sur quelques représentations mémoire possibles pour un graphe.
- En annexe à ce sujet de TP, du code C++ à compléter peut vous servir de base pour la lecture de graphe sur fichier et sa sauvegarde en mémoire.

Affichage du graphe

Après avoir sauvegardé un graphe dans votre structure de données, votre programme doit afficher le graphe sous forme textuelle, par exemple :

- une matrice d'adjacence dans laquelle les valeurs 'vrai' sont remplacées par la valeur associée à l'arc ; les valeurs 'faux' sont remplacées par un symbole de votre choix ;
- une liste des arcs sous la forme '(extrémité initiale , extrémité terminale , valeur)' ;
- …

Cette étape doit être mise en œuvre en parcourant votre structure de données, et non pas en parcourant le fichier d'entrée !

Cette étape permettra de vérifier que votre structure de données est correctement initialisée.

Détection de circuit

Sur la base de la structure de données que vous avez choisie, vous devez écrire une fonction permettant de déterminer si le graphe contient ou non au moins un circuit.

Vous pouvez utiliser l'une ou l'autre des deux méthodes vues en cours :

- par l'élimination successive des points d'entrée (et/ou des points de sortie),
- par le calcul de la fermeture transitive du graphe.

Attention : vous devrez utiliser à nouveau votre graphe après cette détection, donc n'hésitez pas à travailler sur une copie de votre structure de données si vous devez la modifier durant la vérification…

Traces d'exécution :
Votre programme doit indiquer clairement les résultats intermédiaires de votre algorithme, par exemple :

- pour l'élimination des points d'entrée et/ou des points de sortie : indication du ou des sommets supprimés à chaque étape ;
- pour le calcul de la fermeture transitive : valeur (matrice) disponible à chaque itération.

Calcul de rang

Si le graphe ne contient pas de circuit, vous devez calculer un rang pour chaque sommet.

En plus des résultats intermédiaires (valeurs de rang données à chaque sommet et ordre d'affectation), votre code doit se terminer par l'affichage synthétique des valeurs pour l'ensemble des sommets.

Remarque :
Il n'est pas nécessaire de coder la détection de circuit avant de commencer le codage du calcul de rang. Vous pouvez en effet coder le calcul de rang et tester votre programme sur des graphes que vous savez ne pas contenir de circuit !

ANNEXE

Supposons un graphe représenté dans un fichier texte sous la forme suivante :

Fichier :	*Commentaire :*
7	La première ligne contient le nombre de sommets.
8	La seconde ligne contient le nombre d'arcs.
0 1 14	Chacune des lignes suivantes a la forme :
0 4 5	extrémité_initiale extrémité_terminale valeur_de_l_arc
1 3 6	
2 3 8	
5 6 2	
2 1 1	
5 1 9	
6 4 7	
-1	Le fichier est terminé par « -1 » en début de ligne.

Voici un code C++ permettant de lire ce fichier qui peut servir de base au remplissage de votre structure de données représentant le graphe. Les indications en rouge doivent bien évidemment être remplacées par du code effectif qui créée et initialise vos structures de données représentant le graphe.
Ce code nécessite l'insertion de « `#include <fstream>` » dans votre code.

```
// Variables locales
ifstream sourceGraphe ( "monFichier.txt" )
;
      // Déclaration d'une variable de type "flux de données"
      // associée au ficher « monFichier.txt »
      // Le pointeur de lecture est positionné en début de fichier.

int nbSommets ;               // Nombre de sommets du graphe
int nbArcs ;                  // Nombre d'arcs du graphe
int extremiteInitiale ;       // Arcs du graphe
int extremiteTerminale ;
int valeurArc ;

// Lecture nombre de sommets et nombre d'arcs
sourceGraphe >> nbSommets ;
      // La première valeur de type 'int' contenue dans le fichier
      // est stockée dans la variable nbSommets
sourceGraphe >> nbArcs ;
      // La seconde valeur de type 'int' contenue dans le fichier
      // est stockée dans la variable nbArcs
… ajoutez ici votre code pour stocker cette valeur, si nécessaire,
…et éventuellement pour créer et initialiser vos SdD représentant le graphe…

// Lecture des arcs du graphe
sourceGraphe >> extremiteInitiale ;
      // 1ère valeur contenue sur la 3ième ligne
```

```
while ( extremiteInitiale != -1 ) {
      sourceGraphe >> extremiteTerminale ;
      sourceGraphe >> valeurArc ;
      … ajoutez ici votre code pour stocker cet arc dans vos SdD…
      sourceGraphe >> extremiteInitiale ;
            // 1ère valeur contenue sur la ligne suivante
} ;
```

Ce code peut tout simplement être inséré dans la fonction « main » de votre programme.