L3 ISFA (rattrapage) – Programmation
Projet Individuel – Graphe

À rendre fin octobre (date exacte à préciser)

Résumé

L'objectif de ce projet est de rattraper le cours de Programmation de L3. Il est divisé en deux parties : une partie obligatoire sur 8 points, et une partie facultative sur 12.

La partie facultative du projet est très peu dirigée. Elle a pour objectif d'évaluer votre capacité d'initiative, en plus de votre compréhension des notions de base du C++.

Vous pouvez nous adresser vos questions (de préférence par mail) si vous avez des problèmes.

Consignes de rendu

Vous devrez rendre le projet à Yahia Salhi (yahia.salhi@gmail.com) et Antoine Dailly (antoine.dailly@univ-lyon1.fr). La date vous sera communiquée ultérieurement. Ne vous y prenez pas au dernier moment, car aucun projet rendu en retard ne sera corrigé. Telle est la dure loi de la vie.

Tous les fichiers déjà présents dans le projet devront être rendus. Si vous avez utilisé d'autres fichiers (de code ou de test), ils devront également être joints au rendu.

Le code devra être soigneusement commenté.

L'objet de votre mail de rendu devra être : [Rendu Projet Rattrapage] NOM Prénom.

□

Modalités d'évaluation Si votre projet génère une erreur à la compilation ou a des fuites de mémoire importantes, vous aurez zéro.

Le sujet est divisé en deux parties, notées sur 8 et 12 points. □

Exercice 1 : Définitions

Un *graphe* $G = (V, E)$ est une structure de données consistant en un ensemble de *sommets* V reliés par des *arêtes* réunies dans l'ensemble E. La Figure 1 montre un exemple de graphe.

Les graphes peuvent être utilisés pour représenter divers types de données plus ou moins concrètes, ou être vus comme des objets mathématiques. La théorie des graphes est un domaine très actif de la recherche en informatique.

Dans ce projet, vous allez modéliser des graphes et travailler dessus.

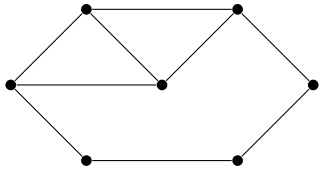

Figure 1 – Un graphe.

Exercice 2 : Complétion du code

Prenez connaissance des fichiers `Sommet.h` et `Graphe.h`. La classe `Sommet` sert à modéliser un sommet et contient deux attributs : son *degré* (c'est à dire le nombre des sommets du graphe qui sont reliés à lui par une arête) et la liste de ses *voisins* (les sommets du graphe qui sont reliés à lui par une arête). La classe `Graphe` sert à modéliser le graphe et contient deux attributs : le nombre et la liste des sommets qu'il contient.

Complétez les fichiers `Sommet.cpp` et `Graphe.cpp`. Vous devrez utiliser de l'allocation de mémoire via `malloc` afin de gérer les listes de sommets et de voisins. Soyez attentifs au fait que les listes stockent des pointeurs sur des `Sommet`, et pas directement des `Sommet`.

Une fois les deux fichiers complétés, exécutez le fichier `main.cpp`. Le texte "*Le graphe a 250 sommets.*" devrait s'afficher.

Prenez bien le temps de vérifier que vous comprenez le modèle, et que vous ne faites pas d'erreur de programmation.

Créez ensuite, au sein de la classe `Graphe`, une méthode `int delta()` qui renverra le degré le plus élevé parmi les sommets du graphe. Testez cette méthode en affichant son résultat dans le `main.cpp`.

Exercice 3 : Pour aller plus loin (facultatif)

Complétez les classes `Sommet` et `Graphe` ainsi que le `main.cpp` afin de manipuler le graphe que vous chargez. Vous êtes libres d'implémenter les algorithmes que vous le souhaitez (ou aucun). Internet regorge de sources vous permettant de comprendre les possibles manipulations et paramètres d'un graphe. Nous vous donnons des pistes de problèmes potentiellement intéressants :

— Trouver le *diamètre* du graphe ;
— Détecter si le graphe est *connexe* ou non ;
— Trouver le *plus court chemin* entre deux sommets choisis par l'utilisateur ;
— Trouver une *coloration* du graphe.

Vous pouvez tester ces algorithmes sur le graphe fourni avec le projet, ou sur d'autres graphes du benchmark DIMACS [1].

1. `http://www.info.univ-angers.fr/pub/porumbel/graphs/`