Université de Reims Champagne-Ardenne
Département d'Informatique

Info0501 – 2022-2023
Pierre Delisle

Travaux pratiques 1

Graphes – Représentation

L'objectif de ce TP est de développer un programme permettant de représenter des graphes. La première étape est d'implémenter une liste chaînée qui sera utilisée pour représenter les listes d'adjacences.

Partie 1 – Implémentation d'une liste chaînée

1. Créez les fichiers **cellule.h** et **cellule.c** implémentant une structure de cellule permettant de stocker un sommet du graphe représenté par un entier, ainsi que de la relier à un prédécesseur et à un successeur du même type. Prévoyez une fonction permettant d'initialiser une cellule à partir d'un sommet.
2. Créez les fichiers **liste.h** et **liste.c** et implémentez-y une structure de liste doublement chaînée de cellules en implémentant les fonctions de liste suivantes :
    - **initialiser_liste** : initialise une liste vide ;
    - **inserer** : prend en paramètre un pointeur sur une cellule et insère cette cellule au début d'une liste ;
    - **afficher_liste** : affiche à l'écran les informations relatives à une liste ;
    - **rechercher** : recherche un sommet dans une liste et retourne soit un pointeur sur la cellule qui contient le sommet, soit NULL ;
    - **supprimer** : prend en paramètre un pointeur sur une cellule d'une liste et supprime cette cellule de la liste ;
    - **detruire_liste** : libère (si nécessaire) les ressources mémoire d'une liste.
3. Créez un programme de test **prog_liste.c** permettant de vérifier le fonctionnement de votre implémentation en demandant à l'utilisateur de saisir des sommets (entiers), en les insérant successivement dans une liste et en affichant la liste ainsi créée. Essayez aussi de supprimer un sommet, de rechercher un sommet et de détruire la liste.

Partie 2 – Création et utilisation d'un Makefile

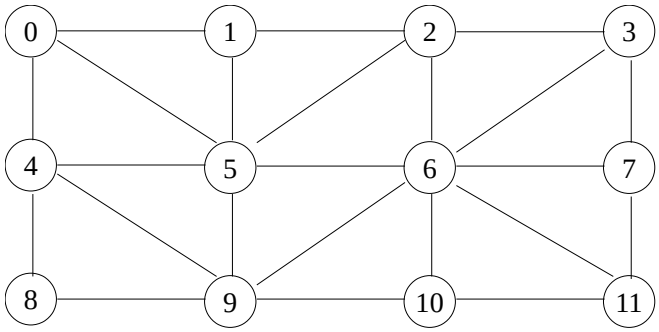
Comme vos programmes des prochains TP nécessiteront un nombre grandissant de modules (**.h et .c)**, il sera utile d'utiliser un Makefile pour compiler vos programmes.

1. Créez et testez un Makefile permettant de compiler l'ensemble de votre programme précédent. Si vous n'êtes pas familier avec les Makefiles et/ou n'en avez pas déjà un modèle privilégié, vous pouvez copier et utiliser directement le Makefile basique suivant :

```
CC = gcc -Wall -O3
OBJETS = prog_liste.o liste.o cellule.o

prog_liste : $(OBJETS)
	$(CC) -o prog_liste $(OBJETS)

prog_liste.o : prog_liste.c liste.h
	$(CC) -c prog_liste.c

liste.o : liste.c liste.h cellule.h
	$(CC) -c liste.c

cellule.o : cellule.c cellule.h
	$(CC) -c cellule.c

clean:
	rm prog_liste $(OBJETS)
```

Partie 3 – Représentation d'un graphe

Maintenant que vous avez une structure de liste chaînée fonctionnelle, il est maintenant possible d'implémenter la représentation d'un graphe. Pour ce faire, nous utiliserons le graphe suivant comme exemple :



Vous pouvez télécharger le fichier de données correspondant à ce graphe (**graphe2.txt**) sur la page Moodle du cours.

Sur la première ligne du fichier, on retrouve le nombre de sommets du graphe (dans la suite, les *n* sommets sont identifiés par un entier entre 0 et *n* – 1). Sur les 2 lignes suivantes, on retrouve les caractéristiques du graphe sous forme de booléens. Les valeurs indiquent donc que ce graphe est non orienté et non valué. Dans la section délimitée par « **DEBUT_DEF_ARETES** » et « **FIN_DEF_ARETES** », chaque arête est représentée par 2 nombres : le sommet d'origine et le sommet d'extrémité. Il faut donc représenter ce graphe en mémoire en utilisant des listes d'adjacences ou une matrice d'adjacences.

1. Créez les fichiers **graphe.h** et **graphe.c** implémentant une structure de graphe incluant des attributs pour représenter des listes d'adjacences et une matrice d'adjacences.
2. Ajoutez les fonctions de graphe suivantes :
    - **initialiser_graphe** : prend en paramètre un fichier texte associé à un graphe et construit le graphe à partir des données du fichier, incluant les listes d'adjacences et la matrice d'adjacences ;

- **afficher_graphe** : affiche à l'écran les informations relatives à un graphe, incluant les listes d'adjacences et la matrice d'adjacences ;
- **detruire_graphe** : détruit un graphe en libérant (si nécessaire) ses ressources mémoire.

3. Écrivez un programme de test qui récupère le nom d'un fichier texte, qui construit le graphe associé et qui affiche le graphe à l'écran. Testez votre programme avec les fichiers **graphe1.txt** (le graphe étudié dans le TD 1) et **graphe2.txt**. Vérifiez que vous obtenez bien un résultat semblable au suivant pour **graphe2.txt** :

```
Nombre de sommets : 12
Non oriente
Non value
Listes d'adjacences:
0 --> 5 4 1
1 --> 5 2 0
2 --> 6 5 3 1
3 --> 7 6 2
4 --> 9 8 5 0
5 --> 9 6 4 2 1 0
6 --> 11 10 9 7 5 3 2
7 --> 11 6 3
8 --> 9 4
9 --> 10 8 6 5 4
10 --> 11 9 6
11 --> 10 7 6

Matrice d'adjacences:

 0  1  0  0  1  1  0  0  0  0  0  0
 1  0  1  0  0  1  0  0  0  0  0  0
 0  1  0  1  0  1  1  0  0  0  0  0
 0  0  1  0  0  0  1  1  0  0  0  0
 1  0  0  0  0  1  0  0  1  1  0  0
 1  1  1  0  1  0  1  0  0  1  0  0
 0  0  1  1  0  1  0  1  0  1  1  1
 0  0  0  1  0  0  1  0  0  0  0  1
 0  0  0  0  1  0  0  0  0  1  0  0
 0  0  0  0  1  1  1  0  1  0  1  0
 0  0  0  0  0  0  1  0  0  1  0  1
 0  0  0  0  0  0  1  1  0  0  1  0
```