Université Clermont Auvergne - IUT Clermont Ferrand
Département Informatique
deuxième année
Développement efficace

TP3-4 : Graphes

Objectif : Représentation d'un réseau informatique par un graphe, manipulation et gestion d'un graphe, parcours en profondeur.

Dans un réseau, les ordinateurs sont des nœuds et les connexions entre ordinateurs des arcs entre des nœuds. On dit qu'un ordinateur A est connecté à un ordinateur B si l'arc $A \rightarrow B$ existe dans le réseau. Attention, dire qu'un ordinateur A est connecté à un ordinateur B n'implique pas nécessairement que B est connecté à A.

Etape 1 :
Écrire les structures nécessaires à la représentation d'un réseau.

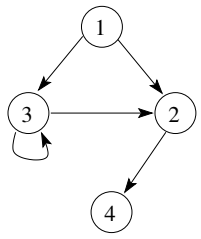
Etape 2 :
Écrire les fonctions et primitives permettant la gestion d'un réseau.

Les différents codes retournés par les fonctions seront définis par des constantes dans un type enuméré CodeRetour comme vu en cours (télécharger les fichiers codeRetour.h et codeRetour.c).

Vous écrirez en parallèle un programme principal permettant à un utilisateur de « gérer »un réseau grâce aux fonctionnalités programmées.

Fonctions et primitives demandées :

1. une fonction d'initialisation d'un réseau à vide, et une vérifiant si le réseau est vide ;
2. une fonction d'affichage en mode texte d'un réseau.



exemple de graphe

```
1 : -> 3
    -> 2
2 : -> 4
3 : -> 2
    -> 3
4 :
```
affichage obtenu

3. une fonction de recherche d'un nœud dans un réseau. Pour un numéro de nœud donné, cette fonction doit retourner un pointeur sur le nœud correspondant (ou NULL si non trouvé) ;

4. une fonction d'ajout dans un réseau d'un nœud connu par son numéro. Si ce noeud existe déjà le noeud n'est pas ajouté. Cette fonction retourne un code indiquant si l'ajout a été effectué, si le noeud était déjà présent, ou s'il y a eu un problème d'allocation mémoire ;
5. une fonction vérifiant l'existence d'une connexion dans un réseau. Cette fonction indique si la connexion connue par les numéros de ces deux noeuds extrémité (« départ et arrivée ») existe ou pas en retournant un code. (Attention : les numéros donnés ne désignent pas nécessairement des nœuds existants). **Cette fonction utilisera une fonction locale** vérifiant l'existence d'une connexion donnée par deux pointeurs sur ses extrémités ;
6. une fonction d'ajout d'une connexion dans un réseau. Pour deux numéros de nœuds donnés (« départ et arrivée »de la connexion), cette fonction réalise l'ajout de la connexion si possible et indique par un code retourné si l'ajout a bien eu lieu ou non et si non quelle erreur l'a empêché ;
7. une fonction de destruction d'une connexion du réseau. Pour deux numéros de nœuds donnés (« départ et arrivée »de la connexion), cette fonction réalise la destruction de la connexion si possible et retourne un code indiquant si la destruction a bien eu lieu. (Attention : les numéros donnés ne désignent pas nécessairement des nœuds existants). **Cette fonction utilisera une fonction locale** réalisant la destruction d'une connexion donnée par deux pointeurs sur ses extrémités ;
8. une fonction de destruction d'un nœud du réseau. Pour un numéro de nœud donné, cette fonction réalise la destruction du nœud si possible (un code retourné indique si la destruction a bien été faite). *Remarque* : dans un graphe, de manière générale la destruction d'un nœud implique celle de tous les arcs partant de lui et pointant sur lui ;
9. une fonction comptant le nombre de nœuds d'un réseau ;
10. une fonction comptant le nombre d'arcs d'un réseau ;
11. Mise en oeuvre du parcours en profondeur :
    (a) une fonction qui détermine si un noeud peut communiquer dans le réseau avec un autre noeud, ou bien si la configuration des cables rend cette communication impossible. On utilisera un parcours en profondeur.

    (b) une fonction qui détermine et affiche l'ensemble des noeuds avec lesquels un noeud donné par son numéro peut communiquer.

Etape 3 : On pourra, si il reste du temps, ajouter les fonctions suivantes :

1. des fonctions de sauvegarde et de chargement d'un réseau dans ou à partir d'un fichier texte. Ces fonctions prennent en paramètre le nom du fichier concerné et non un pointeur de fichier.
   Le format de tels fichiers textes est :
   — le nombre n de nœuds du réseau ;
   — le nombre m de connexions du réseau (une seule entre deux nœuds) ;
   — la liste des n nœuds nommés chacun par un identifiant qui ici sera un nombre ;
   — la liste des m connexions, chaque connexion étant représentée par les numéros des nœuds extrémités.

2. une fonction qui exporte un réseau dans un fichier texte de format `dot` (voir parenthèse) ;

Petite Parenthèse :
Un réseau exporté dans un fichier texte au format `dot` peut être visualisé en exécutant la commande bash `dotty mon_fichier.dot`. Un fichier `.dot` est de la forme :

— la première ligne contient les deux mots `digraph family` ;
— la deuxième ligne contient une accolade ouvrante ;
— puis vient la liste des nœuds avec un nœud par ligne. Chaque nœud est déclaré par un identifiant (un numéro, un nom), et entre crochets un label qui correspond au nom du nœud qui sera affiché.
— et enfin vient la liste des arcs, un par ligne, et chaque arc est donné de la manière suivante : Identifiant1→Identifiant2.
— la dernière ligne contient une accolade fermante.

```
digraph family
{
1 [label="Sommet 1"]
2 [label="Sommet 2"]
3 [label="Sommet 3"]
4 [label="Sommet 4"]

1->2
2->3
1->3
2->4
3->4
}
```