CS316

TP 3 - Algorithmes élémentaires pour les graphes

Récuperer les squelettes fournis pour ce TP à partir de l'adresse suivante :
http ://debbabi.net/teaching/cs316/tp3.tar.gz

A rendre à la fin de la séance :
– Un fichier zip contenant l'ensemble des solutions, à envoyer aux adresses mail suivantes :
`bassem.debbabi@lcis.grenoble-inp.fr` et `dbassem@gmail.com`

Dans ce TP, on vous propose la structure de donnée suivante pour les graphes (voir `Graphe.h`) :

```
typedef struct _Adjacent {
  struct _Sommet *sommet;
  struct _Adjacent* suivant;
  struct _Adjacent* precedent;
} Adjacent;
typedef struct _Sommet {
  int id;
  struct _Adjacent *list_adjacents;
  struct _Sommet *suivant;
  struct _Sommet *precedent;
} Sommet;
typedef Sommet* Graphe;
```

Une fonction `charger_graphe` est déjà implémentée et elle permet de créer un graphe depuis un fichier texte.

```
Graphe *g = charger_graphe("mon_graph.txt");
// ... faire quelque chose!
detruire(g);
```

Exercice 1

Implémenter les méthodes `ajouter`, `afficher` et `détruire`.

Exercice 2 (parcours en largeur)

Étant donnés un graphe G et un sommet origine s, le parcours en largeur emprunte systématiquement les arcs de G pour « *découvrir* » tous les sommets accessibles depuis s. Il calcule la distance (plus petit nombre d'arcs) entre s et tous les sommets accessibles.

On ajoutant à la structure `Sommet` les trois attributs suivants : `distance` (de type `int`), `couleur` (de type `int`) et `parent` (de type `Sommet`), et à l'aide d'une file, implémenter la procédure `parcours_largeur` qui permet d'associé à chaque sommet une distance par rapport à un sommet origine s, et qui permetra de mettre en place l'arborescence de parcours en largeur entre les sommets du graphe accessibles.

Exercice 2 (plus court chemin)

Implémenter la méthode `imprimer_chemin` permettant de tracer les sommets d'un plus court chemin reliant deux sommets s à v.