LANGAGE C AVANCÉ

Projet

Outil de visualisation de graphes

1 Objectifs

Le but de ce projet à faire en binômes est de programmer une application qui trouve les composantes connexes dans un graphe et de visualiser le graphe en colorant d'une même couleur les composantes connexes.

Un graphe est un ensemble de noeuds, dont certaines paires sont directement reliées par un lien (voisins). Un chemin dans un graphe est une séquence de noeuds telle qu'il existe un lien entre chaque paire successive de noeuds dans la séquence. Une composante connexe dans un graphe est un ensemble de noeuds connectés de telle sorte qu'il existe un chemin entre tous les noeuds de l'ensemble.

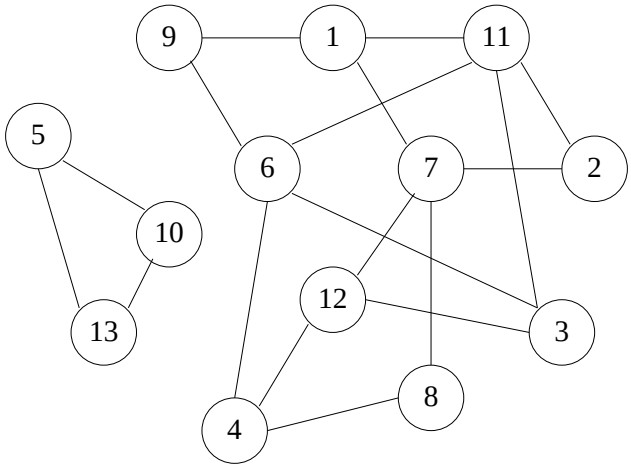

FIGURE 1 – Exemple de graphe composé de 13 noeuds. Les noeuds 5,10 et 13 forment une composante connexe, le reste des noeuds forme l'autre composante connexe. Un exemple de chemin : 4,6,3,12,7,2.

2 Notions à mettre en oeuvre

Pour réaliser ce projet vous allez utiliser toutes les notions vues en cours :
- Pour la représentation des graphes : manipulation avancée des pointeurs ainsi que la gestion de la mémoire ;
- pour la gestion des fichiers d'entrées : manipulation avancée des fichiers textes ;
- pour la modulation de code : utilisation d'une librairie externe ;
- pour la structure du code : découpage en modules et utilisation de l'outil Make ;
- pour les aspects visuels : utilisation d'une librairie graphique.

3 Cahier des charges

Votre code devra contenir au moins les éléments suivants :
- Un module d'entrées/sorties : qui se chargera de toutes les lectures depuis le fichier d'entrées (représentation du graphe) ainsi que les affichages sur la console (messages d'erreurs, d'information etc.).
- Un module pour la manipulation de graphe : qui se chargera de la création/destruction, de la construction et du parcours du graphe.
- Un module pour l'exploitation du graphe : qui se chargera de trouver les composantes connexes dans le graphe.
- Un module graphique : qui se chargera de tous les aspects visuels, comme la création d'une fenêtre et le dessin du graphe.

Vous êtes libres d'ajouter plus de modules si vous le jugez utile.

3.1 Contraintes

Vous devez respecter les contraintes suivantes :
- le fichier d'entrée doit respecter le format suivant (voir exemple à la fin de ce document) :
  - chaque ligne décrit le noeud et ses voisins séparés par un espace ;
  - il y a autant de lignes que de noeuds (un noeud sans voisins doit apparaître seul dans une ligne) ;
  - rien d'autre.
- Ne faire aucune hypothèse sur la taille d'un graphe (ne pas définir de constante de taille maximum par exemple) toutes les allocations doivent être dynamiques ;
- utiliser la bibliothèque SDL (Simple DirectMedia Layer) [1] pour la partie graphique ;
- votre programme doit prendre un seul argument : le chemin vers le fichier d'entrée ;
- vous devez rendre une archive contenant tous vos modules, le makefile pour la compilation ainsi qu'un rapport au format PDF. Pour le format de l'archive utilisez soit tar.gz, zip, ou rar ;
- votre projet doit compiler sans erreurs et sans warnings et devra suivre le format ANSI, pour cela utilisez les options suivantes lors de la compilation : `-Wall -ansi`.

4 Évaluation

Les points suivants seront évalués :
- (10%) la bonne structuration du code (découpage en modules, utilisation de make) et la bonne compilation (sans erreurs et sans warnings avec l'option -ansi)
- (60%) le bon fonctionnement
  - la lecture du fichier texte d'un graphe (10%)
  - la construction et la gestion de la représentation en mémoire (30%)
  - le calcul des composantes connexes (40%)
  - le dessin du graphe (20%)
- (30%) une présentation de 10 min avec transparents et 5 min questions.

⚠ Attention le format de fichier devra être suivi à la lettre. Un exemple est disponible sur le site `http://www-lisic.univ-littoral.fr/~boumaza/cours.php` pour que vous puissiez tester. L'évaluation sera faite sur un autre fichier.

Essayez de bien gérer votre projet en rendant un programme qui marche même si ce dernier ne répond pas à tous les points ci-dessus. Par exemple entre un projet qui gère bien le format de fichier et la construction du graphe en mémoire mais qui ne l'affiche pas correctement, sera mieux considéré

1. Disponible sur `http://www.libsdl.org/`

qu'un autre qui dessine bien mais qui plante lors de la manipulation du graphe.

5 Rendu

Chaque binôme doit envoyer l'archive, décrite plus haut, de son projet à l'adresse suivante `boumaza@lisic.univ-littoral.fr` avec comme sujet `[l3-miage-projet] nom1 nom2` le tout en minuscules sans mettre les prénoms.

Le rapport (au format PDF) devra contenir les éléments suivants :
- la description de la structure de l'application ;
- la description des structures de données utilisées ;
- la liste des modules avec pour chacun la liste des fonctions écrites ;
- une capture écran du rendu du graphe fourni pour le test ;
- l'algorithme commenté pour le calcul des composantes connexes.

⚠ Attention à bien respecter le format du sujet pour éviter que votre projet ne parte dans le répertoire de spam. Si tout va bien, vous recevrez un mail automatique de confirmation quelques minutes à la suite du vôtre. Vous pouvez envoyer plusieurs versions de votre projet seule la dernière sera prise en compte.

6 Annexes

6.1 Format de fichier d'entrée

Exemple pour le graphe de la figure 1.

```
11 1 6 3 2
2 11 7
7 2 1 12 8
3 11 6 12
1 9 7 11
12 7 3 4
4 8 12 6
8 7 4
13 10 5
5 10 13
6 9 11 3 4
9 1 6
10 13 5
```

Le premier élément d'une ligne désigne un noeud, les suivants ses voisins. L'ordre des voisins dans une ligne et celui des lignes n'est pas important. Entre deux voisins il n'y a qu'un seul espace et il n'y a rien après le dernier élément sauf un retour chariot. Il y a autant de lignes que de noeuds, plus la dernière (qui est vide) induite par le retour chariot à la fin du dernier noeud de la dernière ligne.

Bon courage !