Université Lille1 Sciences et Technologies

ALGO

4 novembre 2013
Licence ST-A
François Lemaire

TP Graphes / parcours de graphe

---

1 Transformation de graphe en fichier postscript

Terminer la fonction `graphe2visu(tGraphe graphe, char *outfile)` du TP précédent.

2 Parcours en largeur

Ecrire le parcours non orienté en largeur vu en cours. Ainsi, vous utiliserez les voisins, et votre parcours devra donc fonctionner pour des graphes orientés ou pas.

Pour cela, écrire un programme qui prend en ligne de commande :
- un nom de fichier contenant un graphe
- un nom de sommet de départ

et qui affiche la liste des sommets dans l'ordre de leur parcours en partant du sommet de départ.

Pour faire les coloriages de sommet, une solution simple consiste à utiliser les déclarations :

```
/* Couleurs */
typedef enum {ROUGE=0, BLEU=1, VERT=2} tCouleur;
typedef tCouleur tTabCouleurs[MAX_SOMMETS];
```

et à utiliser utiliser une variable de type `tTabCouleurs`.

3 Parcours avec affichage des couleurs

Si vous avez terminé, vous pouvez modifier votre fonction de parcours pour qu'elle affiche le graphe à chaque fois qu'une couleur est changée. Pour réaliser cela, vous pouvez écrire une variante de `graphe2visu` :

`graphe2visuCouleurs(tGraphe graphe, char *outfile,tTabCouleurs tabCouleurs);`

qui crée un graphe en colorant les sommets d'après les couleurs du paramètre `tabCouleurs`.

Pour information, les indications de couleur pour le programme `dot` doivent se faire de la façon suivante :

```
digraph {
A [color=red];
B [color=red];
C [color=green];
D [color=red];
E [color=green];
F [color=green];
A -> B;
B -> C;
E -> C;
A -> D;
A -> C;
}
```