Université Lille1 Sciences et Technologies

ALGO

4 novembre 2013
Licence ST-A
François Lemaire

**TP Graphes / Présentation de la librairie**

---

# 1 Premiers essais

**Q 1 .** Allez sur le semainier dans la section documents, et trouvez le lien vers la librairie sur les graphes. Lisez intégralement la page principale ainsi que les 4 sections, et surtout testez le programme testAnalyseur.c sur le graphe d'exemple donné.

**Q 2 .** Créez le graphe orienté (en mettant 1 comme valeur à tous les arcs), composé des sommets $A$,$B$,$C$,$D$,$E$ et des arcs $u \to v$, tels que $u$ est un sommet inférieur ou égal à $v$ (au sens de l'ordre alphabétique), c'est-à-dire, $A \to B$, $A \to C$, $A \to D$, $A \to E$, $B \to C$, $B \to D$, $B \to E$, ...

**Q 3 .** Testez le programme `testAnalyseur` sur le graphe, et vérifiez toute la sortie à l'écran.

**Q 4 .** Créez une version non orientée du graphe précédent, et testez également `testAnalyseur` sur le graphe, et vérifiez toute la sortie à l'écran.

# 2 Accès aux données d'un graphe

En vous inspirant du fichier `testAnalyseur.c`, écrivez un programme qui prend un nom de fichier en ligne de commande, et qui :

- affiche la liste des sommets qui n'ont pas de voisins
- affiche la liste des sommets qui ont le plus de voisins (faire un premier parcours pour déterminer le nombre maximum de voisins, puis un second parcours pour afficher les sommets correspondants).

Précision, dans les deux cas, on affichera bien les noms des sommets, et pas leur numéros (voir la fonction `grapheRecupNomSommet`).

# 3 Visualisation

Nous allons visualiser les graphes en nous servant du paquetage `graphviz`, et plus particulièrement de la commande `dot`.

**Q 5 .** Créez un fichier `test.dot` contenant :

```
digraph {
A -> B;
B -> C;
E -> C;
A -> D;
A -> C;
D -> A;
D -> C;
E -> B;
F -> A;
F -> B;
F -> C;
F -> D;
F -> E;
}
```

Lancez la commande `dot -Tps test.dot -o test.ps`. Visualisez le fichier grâce à la commande `evince test.ps`. Vous pouvez choisir d'autres formats de sortie (pdf par exemple), à choisir parmi les formats affichés par `dot -Thelp`.

L'avantage du format ps, et de evince, est que evince met à jour automatiquement l'affichage si les fichier change, ce qui se produira lorsque l'on modifiera des couleurs dans le graphes.

**Q 6 .** Faites la même question avec le fichier

```
graph {
A -- B;
B -- C;
E -- C;
A -- D;
A -- C;
D -- C;
E -- B;
}
```

**Q 7 .** De quoi `digraph` est l'abréviation ?

**Q 8 .** Ecrire une fonction

`void graphe2visu(tGraphe graphe, char *outfile)`

qui prend un graphe en entrée et qui crée le fichier graphique au format ps dans `outfile`.

Cette fonction devra être utilisée de la manière suivante : `graphe2visu(unGraphe, "fichier.ps");` où `unGraphe` est une variable de type graphe. L'effet de la fonction est de créer le fichier nommé `fichier.ps` de type postscript, que vous pourrez visualiser avec evince.

Vous vous inspirez du squelette suivant, qui explique comment ouvrir un fichier en écriture.

```
#include "sys/wait.h"
/* Nécessaire pour la macro WEXITSTATUS */

void graphe2visu(tGraphe graphe, char *outfile) {
  FILE *fic;
  char commande[80];
  char dotfile[80]; /* le fichier dot pour créer le ps */
  int ret;

  /* on va créer un fichier pour graphviz, dans le fichier "outfile".dot */
  strcpy(dotfile, outfile);
  strcat(dotfile, ".dot");
  fic = fopen(dotfile, "w");
  if (fic==NULL)
    halt ("Ouverture du fichier %s en écriture impossible\n", dotfile);

  /*
     on parcourt le graphe pour en tirer les informations
     nécessaires pour graphviz.

     Pour écrire dans le fichier, on utilise fprintf (qui s'utilise
     comme printf mais on mettant en plus fic comme premier paramètre).
     Ex :
     fprintf(fic, "graph {\n");
     ou
     fprintf(fic, "  %s -> %s\n", origine, destination);

  */
  fclose(fic);

  sprintf(commande, "dot -Tps %s -o %s", dotfile, outfile);
  ret = system(commande);
  if (WEXITSTATUS(ret))
    halt("La commande suivante a échoué\n%s\n", commande);
}
```

**Q 9 .** Testez votre fonction en écrivant un programme qui prend un fichier de graphe en ligne de commande et qui crée un fichier ps nommé `visu.ps`. Une fois le premier fichier `visu.ps` créé, vous pouvez laisser le programme `evince visu.ps` tourner. Pour faire cela, vous appellerez la fonction `graphe2visu` avec comme second paramètre `"visu.ps"`.

# 4 Parcours

**Q 10 .** Si vous avez terminé, vous pouvez écrire le parcours en largeur vu en cours. 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.