TP graphes

Baptiste Jeudy

Les TP se font en groupes de 2 étudiants au maximum. Les redoublants travaillent seul. Le rendu du TP est obligatoire sous peine de défaillance.

Lisez tout le sujet avant de commencer. Ce que vous avez à faire est indiqué dans la dernière section.

1 Représentation du graphe

Soit $G = (S, A)$ un 1-graphe orienté avec $S = \{1, ..., n\}$. Il sera représenté par une matrice d'adjacence stockée dans une variable globale `MAD` de type tableau d'entier à deux dimensions. Le nombre de sommets sera lui aussi stocké dans une variable globale `n`. Ce nombre devra toujours être strictement inférieur à la pseudo-constante `NMAX` :

```
#define NMAX 100          // le nombre max de sommets
int MAD[NMAX+1][NMAX+1];    // matrice d'adjacence
int n;                    // ordre du graphe
```

Comme ces deux variables sont globales, elles sont accessibles dans toutes les fonctions sans avoir besoin de les passer en paramètre.

Les sommets seront numérotés à partir de 1. Les cases d'indice 0 dans les tableaux ne seront donc pas utilisées (en C les tableaux commencent à l'indice 0). Si `MAD[i][j]` vaut 1, cela signifie qu'il y a un arc du sommet `i` au sommet `j`, sinon `MAD[i][j]` vaut 0.

2 Lecture des données

Les données définissant un graphe sont lues par le programme sur son entrée standard. La fonction `lire_graphe` qui remplit le tableau `MAD` et la variable `n` est fournie.

Les données du graphes sont, dans l'ordre de lecture :

— le nombre n de sommets,

— le nombre m d'arcs,

— pour chacun des m arcs, x le sommet origine et y le sommet extrémité.

Les fichiers fournis dans le répertoire `graphes_exemples` contiennent des exemples de graphes, vous pouver en regarder un.

Remarque :

Votre programme lira ces données sur son entrée standard. Vous pourrez donc rentrer le graphe directement au clavier. Cependant, cela peut s'avérer fastidieux... Vous aurez donc tout intérêt à envoyer directement un fichier exemple sur l'entrée standard de votre programme (redirection de l'entrée standard). (quelque chose du type : `./mon_prog < exemple_graphe1.txt`).

3 Trou noir

Un sommet s d'un graphe orienté à n sommets est un trou noir si tous les autres sommets ont un arc allant vers s et si aucun arc ne part de s (qu'est-ce que cela signifie dans la matrice d'adjacence ?). On veut pouvoir tester cette propriété sans lire toute la matrice d'adjacence (donc sans calculer les degrés de tous les sommets). Voici un algorithme qui fait cela :

1. Au départ, le sommet courant est le sommet 1.
2. On emprunte ensuite le premier arc partant du sommet courant et menant à un sommet de numéro plus élevé que le sommet courant. Pour cela, on considère les arcs dans l'ordre croissant du sommet d'arrivée. Le sommet d'arrivée devient le nouveau sommet courant. Cette étape est répétée tant que cela est possible.

3. Quand l'étape 2 n'est plus possible, le sommet courant est le seul sommet du graphe qui peut éventuellement être un trou noir (pourquoi ?). Il reste juste à le vérifier (comment ?).

Si cet algorithme est correctement implémenté, il ne nécessite de lire (dans le pire des cas) que $3n$ valeurs dans la matrice d'adjacence (n valeurs pour l'étape 2 et $2n$ valeurs pour l'étape 3), c'est-à-dire que sa complexité est de $O(n)$.

4 À faire

N'oubliez pas de déposer votre TP sur le cours en ligne (uniquement le fichier `graphe_oriente.c`) à la fin de la séance (cf. point 11 ci-dessous). Si vous n'avez pas terminé, vous pouvez le continuer chez vous et déposer une 2e version avant la séance de TP suivante (dans ce cas, ne supprimez pas la première version que vous avez déposé).

Vous devez :

1. Créer dans votre répertoire personnel un sous-répertoire `graphes/` (votre répertoire `graphes` contiendra tous les tps de graphe).
2. Copier dans ce répertoire le fichier `tp1.zip` disponible sur claroline et le décompresser (cela va créer un sous répertoire `tp1_...`)
3. Le répertoire `graphes_exemples/` contient des exemples de graphes. Dessinez le graphe contenu dans le fichier `graphe5.txt` puis celui dans `graphe1.txt`.
4. Indiquer le nom des membres du groupe au début du fichier `graphe_oriente.c` (groupes de 2 au maximum, les redoublants doivent être seuls). Lisez les commentaires de ce fichier. En particulier, vous aurez à répondre aux questions au début de ce fichier pour indiquer votre avancement dans le tp.
5. Compléter le programme `graphe_oriente.c` fourni en écrivant une fonction `void afficheMAD()` qui affiche la matrice d'adjacence du graphe. Pour compiler votre programme, vous avez juste à taper `make` (le `Makefile` est fourni). Vous testerez votre programme avec les exemples de graphes fournis pour vérifier que les matrices d'adjacences sont correctes. Le fichier texte `log` contient le résultat de ce que devrait afficher votre programme pour chaque exemple. Par exemple, pour le premier graphe, la fonction doit afficher :

```
  Matrice d'adjacence :
   1 2 3 4 5 6 7 8
1: X X . . . . . .
2: . . . . X . . .
3: . . . X . . X .
4: . X . . X . . .
5: . . . . . X . .
6: . . . . . . . .
7: . . . . . . . X
8: . . . . X X . .
```

6. Rajoutez dans votre programme le calcul des degrés entrants et sortants que vous stockerez dans deux tableaux d'entiers `dplus` et `dmoins`. Vous afficherez ensuite ces degrés sous la matrice d'adjacence. Ex :

```
1 2 0 1 3 2 1 1 : degrés entrants
2 1 2 2 1 0 1 2 : degrés sortants
```

7. Votre programme affichera ensuite la liste des sommets isolés, puits et sources. Vous testerez votre programme sur les graphes exemples fournis. Le fichier `log` contient les résultats sur chacun de ces graphes.
8. Finalement, vous écrirez une fonction `int trou_noir()` qui retourne 0 si le graphe ne contient pas de trou noir et qui retourne le numéro du sommet qui est trou noir s'il y en a un. Cette fonction devra utiliser l'algorithme présenté plus haut. En particulier, vous ne devez pas utiliser les tableaux `dplus` et `dmoins`. Comparez encore vos résultats avec ceux fournis.

9. Une fois que vous avez testé votre programme sur un ou deux graphes exemples, vous pouvez le lancer sur tous les graphes avec le script `test.sh`. La sortie de ce script doit être la même que le contenu du fichier `log`.
10. Vous pouvez aussi rediriger la sortie du script dans un fichier et comparer les résultats de votre algorithme avec le fichier `log` fourni en utilisant la commande diff :

    ```
    ./test.sh > mon_log
    diff -b mon_log log
    ```

    La commande diff affiche les lignes qui diffèrent entre les deux fichiers. Si votre programme est correct, elle ne devrait presque rien afficher (seulement la première ligne du fichier qui indique la date où le test a été fait).
11. À la fin de la séance, un des membres du groupe doit déposer votre fichier sur claroline (pas de zip, uniquement le fichier `graphe_oriente.c`). Vérifiez que vous avez mis vos noms et répondu aux questions dans les commentaires avant. (utilisez le script `test_rendu.sh` pour vérifier que vos réponses sont dans le bon format).