Licence d'informatique

TP graphes : tri topologique

Baptiste Jeudy

---

1 Représentation du graphe par TLA

Soit $G = (S, A)$ un 1-graphe orienté avec $S = \{1, ..., n\}$. Il sera représenté par un tableau de listes d'adjacence stocké dans une variable globale `TLA` de type tableau de liste (en réalité un tableau de pointeurs sur des listes). 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
Liste *TLA[NMAX+1];
int n;
```

Les sommets seront numérotés à partir de 1. Les cases d'indice 0 dans les tableaux ne seront donc pas utilisées (le type `Liste` est expliqué dans la section suivante).

2 Listes

La représentation par TLA et l'algorithme du tri topologique lui même nécessitent d'utiliser des listes. Comme le type liste n'existe pas par défaut en C, il faut le créer.

Le fichier `include/liste.h` contient la définition d'un type `Liste` pour représenter des listes d'entiers. Il contient aussi la définition de plusieurs fonctions pour manipuler les listes. Ces fonctions sont implantées dans le fichier `include/liste.o`.

Vous devrez utiliser ce type et ces fonctions pour le TLA et la liste de sommets que l'algorithme du tri topologique utilise. Voila les fonctions que vous pouvez utiliser sur les listes. Pour plus de détails, regardez le fichier `include/liste.h`.

```
Liste *creer_liste_vide();        /* Renvoie un pointeur sur une liste vide */
int est_vide(Liste *l);           /* Renvoie vrai si la liste l est vide */
void enfile(int sommet, Liste *l); /* Ajoute le sommet à la fin de la liste l */
void empile(int sommet, Liste *l); /* Ajoute le sommet au début de la liste l */
int depile(Liste *l);             /* Retire le premier sommet de la liste l et le renvoie */
int element(Liste *l, int index); /* Retourne l'élément de la liste à l'index indiqué */
int taille(Liste *l);             /* Retourne le nombre d'éléments de la liste */
void affiche_liste(Liste *l);     /* affiche la liste passée en paramètre */
```

3 Lecture des données

Le format des données est le même que dans le premier TP. Vous devrez écrire la fonction qui lit les données et les stocke dans le TLA. Inspirez vous de la fonction du TP 1.

4 Tri topologique

Soit $G = (S, A)$ un graphe orienté sans circuit avec $S = \{1, ..., n\}$. Un tri topologique des sommets de G est une application $rang$ de S dans $\{1, ..., n\}$ telle que si $(i, j) \in A$ alors $rang(i) < rang(j)$.

Par exemple, chaque sommet peut être une tâche à effectuer dans la construction d'une maison (creuser les fondations, couler les fondations, faire la dalle de sol, monter les mur, mettre l'isolation, poser les fenêtres...). Certaines tâches ne peuvent pas être effectuées avant que d'autres ne soient

terminées (on ne peut pas poser les fenêtres avant d'avoir monté les murs). Cela se traduit dans le graphe par un arc (i, j) si la tâche i doit précéder la tâche j. On recherche un ordre dans lequel on peut effectivement exécuter les tâches. Le tri topologique donne cet ordre : la première tâche à exécuter est la tâche i telle que $rang(i) = 1$, la deuxième est celle pour laquelle $rang$ vaut 2, etc.

Il peut y avoir plusieurs solutions au problème mais un tri topologique n'existe que si le graphe est sans circuit.

Il existe plusieurs algorithmes permettant de résoudre ce problème. Vous utiliserez celui vu en TD. En voici le principe général :

On calcule d'abord le degré entrant de tous les sommets (on les stocke dans un tableau `dmoins`). Les sommets sources (i.e., dont le degré entrant est nul) sont mis dans une liste (cela correspond aux tâches qui peuvent être commencées immédiatement, elles n'ont pas de pré-requis).

Ensuite on fait une boucle tant que la liste des sommets sources n'est pas vide. À chaque itération on enlève un sommet de la liste. La valeur de *rang* pour ce sommet sera le numéro de l'itération. Ensuite, on diminue le degré des successeurs du sommet dans le tableau `dmoins` (attention, il ne faut pas recalculer les degrés à partir de zéro). On rajoute dans la liste tous les sommets dont le degré entrant est devenu nul.

Finalement, si le nombre d'itérations correspond au nombre de sommets, on a gagné et on a bien calculé *rang*, sinon c'est que le graphe contient un circuit et le tri topologique est impossible (pourquoi ? réponse à indiquer dans le fichier .c).

5 À faire

Vous trouverez tous les fichiers dont vous avez besoin dans l'archive zip :

— un sous répertoire `include/` contenant les fichiers nécessaires pour utiliser les listes (`liste.h` et `liste.o`) ;
— `tri_topo.c` qui contient le programme proprement dit (c'est le fichier que vous allez devoir compléter et rendre) ;
— un `Makefile` (pour lancer une compilation, vous avez juste à taper `make`) ;
— un sous répertoire `graphes_exemples/` contenant des exemples de graphes ;
— un fichier `log` contenant les résultats sur les exemples de graphes.

Vous devez :

1. Mettre ces fichiers dans un répertoire `graphes/tp2/` sur votre compte.
2. Compilez et exécutez le programme `tri-topo` et lisez le `main` pour voir des exemples d'utilisation des fonctions sur les listes. Assurez vous de bien comprendre ce que le programme affiche en lien avec le code source.
3. Écrire la fonction `lire_graphe_tla` et la fonction `affiche_tla`. Testez votre programme avec les graphes exemples pour vérifier qu'il lit correctement les graphes et que les TLA sont corrects.
4. Écrire la fonction `tritopologique` qui remplit le tableau `rang` en effectuant un tri topologique. Votre fonction doit aussi, à chaque étape, afficher le sommet choisi, le contenu de la liste des sommets sources et les degrés des sommets (cf. le fichier `log`). Modifiez également la fonction `main` pour qu'elle appelle le tri topologique. La fonction `main` doit afficher les sommets dans l'ordre donné par le tri topologique si le tri est possible et afficher un message sinon. Vous vérifierez en dessinant les premiers graphes que l'ordre donné par votre programme est correct. Vous pourrez ensuite comparer avec les tris topologiques indiqués dans le fichier `log`.
5. Complétez dans le fichier `tri_topo.c` les lignes commençant par `X>>` (`X` est une lettre ou un chiffre) : la ligne avec vos nom et les lignes indiquant votre avancement. Répondez aussi à la question 6 sur le cas où le tri topologique est impossible.

   Vérifiez vos réponses en lançant le script :

   `./test_rendu.sh tri-topo.c`

   Cela doit afficher vos réponses.

   Déposez votre fichier `tri_topo.c` sur le cours en ligne à la fin de la séance.

   Vous pourrez déposer une version étendue jusqu'à la semaine prochaine si vous n'avez pas fini.