TP3 AA
Polytech Grenoble, RICM3

Jean-François Méhaut
UGA-CEA/LIG

19 Mars 2019

Ce TP comprend deux parties : Une première partie sur des fonctions de base sur les graphes. Une seconde partie sur l'examen d'Algo qui a été proposé en 2017.

1 Graphes

Nous supposerons que les fonctions présentées en cours ou développées pendant les séances de TP sont opérationnelles. Vous pouvez les réutiliser (appeler), si vous en avez besoin, pour répondre à certaines des questions. Un graphe est composé d'un ensemble de nœuds/sommets reliés par des arcs/arêtes. Le sujet de cet examen porte sur des graphes **pondérés** (poids sur les arcs/arêtes) et **orientés** (flèches sur les arcs/arêtes).

Quelques définitions caractérisant les chemins et les graphes :

1. Un chemin est une suite consécutive d'arcs dans un graphe orienté.
2. Un chemin **élémentaire** est un chemin ne passant pas deux fois par un même nœud, c'est à dire un chemin dont tous les nœuds sont distincts.
3. Un chemin **simple** est un chemin ne passant pas deux fois par une même arête, c'est à dire un chemin dont toutes les arêtes sont distinctes.
4. Un chemin est dit **Eulérien** si toutes les arêtes du graphe sont utilisées dans le chemin.
5. Un graphe est dit **Eulérien** si il existe au moins un chemin qui soit **Eulérien**.
6. Un chemin est dit **Hamiltonien** si tous les nœuds du graphe sont utilisés dans le chemin.
7. Un graphe est dit **Hamiltonien** si il existe au moins un chemin qui soit **Hamiltonien**.
8. La **longueur** d'un chemin est la somme des poids des arêtes.
9. La **distance** entre deux nœuds x et y est la longueur du plus court chemin entre x et y.
10. L'**excentricité** d'un nœud est sa distance maximale avec les autres nœuds du graphe.
11. Le **diamètre** d'un graphe est l'excentricité maximale de ses nœuds.

1.1 Questions

1. (2 points) Définissez le type `chemin_t` mémorisant les informations nécessaires pour un chemin. Cette déclaration de type s'appuiera, soit sur la représentation par matrice d'adjacence des graphes, soit sur la représensation par liste chaînee des graphes. Le choix est important pour les questions suivantes. Ce type `chemin_t` va être utilisé dans les questions suivantes.

2. (1 point) Décrivez en C l'implémentation de la fonction `elementaire` qui vérifie si un chemin est **élémentaire** ou pas. La fonction `elementaire` renvoie 1 si le chemin `c` est **élémentaire**, 0 sinon.

```
int elementaire (pgraphe_t g, chemin_t c)
{
  ...
}
```

3. (1 point) Décrivez en C l'implémentation de la fonction `simple` qui vérifie si un chemin est **simple** ou pas. La fonction `simple` renvoie 1 si le chemin `c` est **simple**, 0 sinon.

```
int simple (pgraphe_t g, chemin_t c)
{
  ...
}
```

4. (2 points) Décrivez en C l'implémentation de la fonction `eulerien` qui vérifie si un chemin est **Eulérien** ou pas. La fonction `eulerien` renvoie 1 si le chemin `c` est **Eulérien**, 0 sinon.

```
int eulerien (pgraphe_t g, chemin_t c)
{
  ...
}
```

5. (2 points) Décrivez en C l'implémentation de la fonction `hamiltonien` qui vérifie si un chemin est **Hamiltonien** ou pas. La fonction `hamiltonien` renvoie 1 si le chemin `c` est **Hamiltonien**, 0 sinon.

```
int hamiltonien (pgraphe_t g, chemin_t c)
{
  ...
}
```

6. (3 points) Décrivez en C l'implémentation de la fonction `graphe_eulerien` qui vérifie si un graphe est **Eulérien** ou pas. La fonction `graphe_eulerien` renvoie 1 si le graphe `g` est **Eulérien**, 0 sinon.

```
int graphe_eulerien (pgraphe_t g)
{
  ...
}
```

7. (3 points) Décrivez en C l'implémentation de la fonction `graphe_hamiltonien` qui vérifie si un graphe est **Hamiltonien** ou pas. La fonction `graphe_hamiltonien` renvoie 1 si le graphe `g` est **Hamiltonien**, 0 sinon.

```
int graphe_hamiltonien (pgraphe_t g)
{
  ...
}
```

8. (2 points) Décrivez en C l'implémentation de la fonction `distance` qui calcule la **distance** entre deux nœuds x et y du graphe `g`.

```
int excentricite (pgraphe_t g, pnoeud_t x, pnoeud_t y)
{
  ...
}
```

9. (2 points) Décrivez en C l'implémentation de la fonction `excentricite` qui calcule pour un nœud `n` son **excentricité** dans le graphe `g`.

```
int excentricite (pgraphe_t g, pnoeud_t n)
{
  ...
}
```

10. (3 points) Décrivez en C l'implémentation de la fonction `diametre` qui calcule le **diamètre** du graphe `g`.

```
int diametre (pgraphe_t g)
{
  ...
}
```