IR2 - Algorithmique des graphes

Fiche 2 - Algorithme d'Euler

L'objectif de ce TP est d'implanter un algorithme pour décider si un graphe admet une chaîne ou un cycle eulérien, et, si tel est le cas, d'en exhiber un. Dans ce TP, la structure de donnée à choisir est celle de **liste d'adjacence**. Le langage de programmation est le Java.

Étant donné un graphe connexe G non orienté, non pondéré, le problème d'Euler consiste à trouver une chaîne (resp. un cycle) contenant toutes les arêtes de G, une seule et une unique fois. Une telle chaîne (resp. cycle) est appelée *chaîne eulérienne* (resp. *cycle eulérien*).

Exercice 1. Existence d'une chaîne eulérienne
Voici le Théorème d'Euler :

(i) *Un graphe connexe admet un cycle eulérien si et seulement si il ne possède aucun sommet de degré impair.*

(ii) *Un graphe connexe admet une chaîne eulérienne si et seulement si il possède* 0 *ou* 2 *sommets de degré impair.*

1. Écrire une méthode qui retourne le nombre de sommets de degrés impairs dans un graphe.
2. Utiliser le Théorème d'Euler pour écrire une méthode qui détermine si un graphe connexe admet un cycle eulérien. En écrire une autre pour déterminer si un graphe connexe admet une chaîne eulérienne.
3. Que se passe-t-il si un graphe possède exactement un sommet de degré impair ?

Exercice 2. Calcul d'un cycle eulérien ou d'une chaîne eulérienne
L'algorithme suivant permet de calculer un cycle eulérien dans un graphe G initialement connexe et sans sommet de degré impair :

Algorithm 1 CycleEulérien(G, x)

Require: un graphe G sans sommet de degré impair et x un sommet de G.
Ensure: une liste $(x, x_2, \ldots, x_k, x)$ de sommets de G formant un cycle eulérien.

```
1:  A ← ensemble des sommets adjacents à x dans G.
2:  if A = ∅ then
3:     return (x)
4:  else
5:     C = (x = y_1, ..., y_ℓ = x) ← un cycle quelconque d'origine x dans G.
6:     supprimer les arêtes de C dans G.
7:     R ← ()
8:     for i = 1 à ℓ do
9:        R ← R · CycleEulérien(G, y_i)
10:    end for
11:    return R
12: end if
```

L'algorithme suivant, quand à lui, permet de calculer une chaîne eulérienne dans un graphe G initialement connexe et avec exactement deux sommets x et y de degré impair :

Algorithm 2 ChaîneEulérienne(G, x, y)

Require: un graphe G ayant deux sommets de degré impair x et y.
Ensure: une liste $(x, x_2, \ldots, x_k)$ de sommets de G formant une chaîne eulérienne.
1: $C = (x = y_1, \ldots, y_\ell) \leftarrow$ une chaîne quelconque d'origine x et d'arrivée y dans G.
2: supprimer les arêtes de C dans G.
3: $R \leftarrow ()$
4: **for** $i = 1$ à ℓ **do**
5: $\quad R \leftarrow R \cdot$ CycleEulérien(G, y_i)
6: **end for**
7: **return** R

Les chaînes et les cycles sont encodés par des listes de sommets.

1. Écrire une méthode qui, étant donnés deux sommets de degrés impairs x et y dans un graphe G connexe qui possède exactement deux sommets de degrés impairs, retourne une chaîne d'origine x et d'arrivée y et supprime les arêtes qui la constituent dans G.
2. Écrire une méthode qui, étant donné un sommet x dans un graphe G qui ne possède aucun sommet de degré impair, retourne un cycle d'origine x et supprime les arêtes qui le constituent dans G.
3. En s'inspirant des algorithmes donnés, écrire une méthode qui, à l'entrée d'un graphe G connexe, retourne :
   – la liste vide si G n'admet ni cycle ni chaîne eulérienne ;
   – un cycle eulérien si G en admet un ;
   – une chaîne eulérienne si G en admet une.