

Algorithmique Avancée - TD 2 - 13 octobre 2011

Université Paris 7 — BioInformatique — M1

Parcours de graphes

1 Graphes bipartis

Un graphe est dit **biparti** si tous ses sommets peuvent être coloriés, chacun soit en vert soit en rouge, de sorte que chaque arête soit bicolore.

Exercice 1 Montrez qu'un graphe ayant un cycle de longueur impaire n'est pas biparti. La réciproque est-elle vraie ?

Exercice 2 On va maintenant faire un algorithme. On commence par colorier un sommet (disons, en vert). De quelle couleur doivent être tous ses voisins ? Et les voisins de ses voisins ? Déduisez-en un algorithme à base de parcours qui

- soit colorie le graphe
- soit dit “non, il n'est pas biparti”

Exercice 3 Et avec trois couleurs (disons, vert, bleu, rouge) ?

2 Parcours en largeur

Exercice 4 On note $deg(u)$ le degré, c'est-à-dire le nombre de voisins, d'un sommet u . Montrer que dans un graphe orienté de m arcs

$$\sum_u deg(u) = m$$

et que dans un graphe non-orienté de m arêtes

$$\sum_u deg(u) = 2m$$

On rappelle le code du parcours en largeur :

```
BFS( $G$  : graphe,  $x$  : sommet)
 $d[ ]$  : tableau initialisé à  $-\infty$ ;  $d[x] = 0$ 
 $Pere[ ]$  : tableau initialisé à null
 $Etat[ ]$  : tableau initialisé à Non-Atteint;  $Etat[x] = Atteint$ 
 $Atteints$  : file vide;  $MettreEnQueue(Atteints, x)$ 
tant que  $Atteints$  non vide faire
     $y = ExtraireTete(Atteints)$ 
     $Etat[y] = Marqué$ 
    pour tout voisin  $u$  de  $y$  faire
        si  $Etat[u] = Non-Atteint$  alors
             $d[u] = d[y] + 1$ 
             $Pere[u] = y$ 
             $Etat[u] = Atteint$ 
             $MettreEnQueue(Atteints, u)$ 
```

Exercice 5 Reprendre l'analyse de complexité pour se convaincre que cet algorithme est bien en $O(n + m)$. En particulier, comment traiter la ligne "pour tout voisin" ?

Exercice 6 Modifier l'algorithme pour qu'il affiche un plus court chemin entre deux sommets a et b passés en paramètre, ou bien que a est inaccessible depuis b

Exercice 7 Le graphe est maintenant orienté. Qu'est-ce qui change ? Calcule-t-on toujours des plus courts chemins, et en quel temps ?

Exercice 8 Le graphe est maintenant valué par des longueurs d'arêtes. Mêmes questions que l'exercice précédent.

3 Graphes Euleriens

Au 18ème siècle une question animait les dîners des notables de la petite ville de Königsberg en Prusse orientale (aujourd'hui Kaliningrad). À l'époque la ville comptait sept ponts qui franchissaient la rivière Pregolya. Ces ponts reliaient les deux îles et les rives de Königsberg. Un petit problème mathématique qui fit fureur à l'époque était de savoir si lors de la promenade dominicale il était possible de passer une fois et une seule par chaque pont de la ville et revenir à son point de départ. En 1736, Leonhard Euler apporta la réponse à ce problème. Saurez vous en faire autant ?

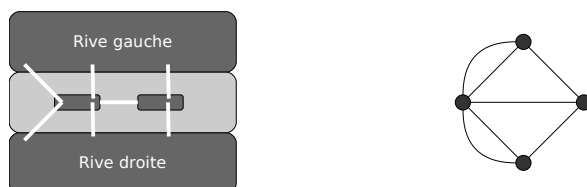


FIGURE 1 – Schéma des ponts de Königsberg, et le multi-graphe associé.

Soit G un graphe. G est dit *eulerien* s'il contient un cycle passant une et une seule fois par chaque arête.

En d'autres termes, on peut ordonner les arêtes en $e_1 \dots e_m$ tels que pour tout $i < m$ les arêtes e_i et e_{i+1} ont un sommet commun, ainsi que les arêtes e_m et e_1 .

Exercice 9 Montrez que si un sommet a degré impair alors le graphe n'est pas eulerien.

Exercice 10 Montrez qu'un graphe non connexe n'est pas eulerien

On imagine maintenant que l'on peut *marquer* les arêtes. Informellement, c'est ce que l'on fait quand on parcourt le graphe avec un feutre sans relever la main. Formellement, le marquage est une fonction booléenne $E(G) \mapsto \{0, 1\}$.

Une *marche* est l'algorithme suivant :

1. Prendre une arête non marquée e
2. La marquer
3. Choisir une arête non marquée f incidente à e (ayant un voisin en commun)
4. Goto 2

La marche s'arrête quand la condition 3 ne peut être remplie.

Exercice 11 Montrez que si tous les sommets ont degré pair, alors la marche stoppe nécessairement sur le sommet de départ.

Exercice 12 Sur un exemple, montrez que la marche ne construit pas toujours un cycle eulerien.

Exercice 13 En utilisant des marches, faire un algorithme qui trouve un cycle eulerien, s'il en existe un

Exercice 14 Retrouvez le fameux théorème d'Euler sur la condition nécessaire et suffisante pour qu'un graphe admette un cycle eulerien, et dites pourquoi votre algorithme en constitue une preuve.