Algorithmique et Programmation

TD n° 5 : Graphes

École normale supérieure – Département d'informatique
algoL3@di.ens.fr

2012-2013

Exercice 1. Étant donné un graphe acyclique $G = (V, E)$, une k-coloration de G est une fonction $c : V \to \{1, \ldots, k\}$ associant à chaque sommet u de G une couleur $c(u)$, telle que si $u, v \in E$ alors $c(u) \neq c(v)$. La fonction c est dit *fonction de coloration*. Un graphe pour lequel il existe une coloration qui utilise k couleurs est dit k-coloriable.

1. Montrer que chaque graphe connexe et acyclique est 2-coloriable.
2. Montrer qu'un graphe qui admet un cycle de longueur 3 n'est pas 2-coloriable.
3. Donner un algorithme qui détermine si un graphe est 2-coloriable et, si c'est le cas, donne une 2-coloration. Quelle est la complexité de l'algorithme (on représentera un graphe par liste d'adjacences) ?
4. On suppose que, pour tout $u \in V$, la paire u, u n'est pas une arête et qu'il y a au plus Δ sommets v tel que u, v est une arête. Montrer en construisant un algorithme, que le graphe est alors $(\Delta + 1)$-coloriable. Quelle est la complexité de l'algorithme ?
5. Montrer que tout graphe possédant n sommets et 3-coloriable peut être colorié avec $O(\sqrt{n})$ couleurs par un algorithme effectuant un nombre non exponentiel d'opérations. On utilisera les algorithmes des questions précédentes en remarquant que dans un graphe 3-coloriable, pour tout sommet u donné, le sous-graphe correspondant aux sommets v tel que u, g est une arête est lui 2-coloriable.

Exercice 2. Un tour eulérien d'un graphe orienté G à n sommets et m arêtes est un cycle qui traverse chaque arête exactement une fois. Un tel tour existe toujours si le degré entrant de chaque sommet de V est égal à son degré sortant. Donner un algorithme de complexité $O(n+m)$ pour trouver un tour eulérien dans un tel graphe.

Exercice 3.

1. Un graphe orienté $G = (V, E)$ contient un *puit universel* $v \in V$ si v est un sommet de degré entrant $|V| - 1$ et de degré sortant 0. Donner un algorithme qui fait seulement $O(n)$ tests à la matrice d'adjacence d'un graphe orienté G et détermine si G contient un *puit universel*.
2. Un scorpion est un graphe non-orienté G de la forme suivante : il y a 3 sommets spéciaux, appelés le dard (*sting*), la queue (*tail*), et le corps (*body*), de degré 1, 2 et $n-2$, respectivement. Le dard est connecté seulement à la queue ; la queue est connectée seulement au dard et au corps ; et le corps est connecté à tous les sommets sauf le dard. Les autres sommets de G peuvent être connectés aux autres de façon arbitraire.

   Donner un algorithme qui fait seulement $O(n)$ tests à la matrice d'adjacence d'un graphe G et détermine si G est un scorpion.