Algorithmique et Programmation

TD n° 6 : Graphes Orientés

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

2015-2016

Exercice 1. Arbre des blocs

Les parcours en largeur et profondeur permettent de trouver les composantes connexes d'un graphe. Nous pouvons définir des notions de connexité plus élevées et dans certain cas, être en mesure de calculer les « composantes » pour ces notions. Dans cet exercice, nous verrons un exemple avec la 2-connexité.

Un *sommet séparant* d'un graphe est un sommet qui augment le nombre de composantes connexes du graphe lorsqu'on l'enlève. (Pour un graphe G connexe, $G - v$ n'est pas connexe.) Un (sous)graphe est *2-connexe* s'il n'a pas de sommet séparant. Un *bloc* d'un graphe G est un ensemble maximal de sommets de G qui induisent un sous graphe 2-connexe.

1. Montrer que la racine r d'un arbre de parcours en profondeur est un sommet séparant si et seulement si r a au moins deux fils.
2. Montrer qu'un sommet non-racine v d'un arbre de parcours en profondeur est séparant si et seulement si v a un fils u tel qu'il n'y a pas d'arête (hors-arbre) d'un descendant de u vers un ancêtre propre de v.
3. Donner un algorithme linéaire qui trouve pour tout sommet v dans un arbre de parcours en profondeur, le plus haut (plus près de la racine) sommet que l'on peux atteindre à partir du sous-arbre de v (arbre de v et tous ses descendants).

   Il est possible d'utiliser le temps de terminaison du parcours en profondeur.
4. Donner un algorithme pour trouver tous les sommets séparant d'un graphe connexe en temps linéaire.
5. Donner un algorithme pour trouver tous les blocs d'un graphe connexe en temps linéaire (à partir des sommets séparants et de l'arbre de parcours en profondeur du graphe).

Exercice 2. Parcours eulérien

Un *parcours eulérien* d'un graphe orienté fortement connexe G à n sommets et m arêtes est un cycle qui traverse chaque arête exactement une fois. Un tel parcours 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 parcours eulérien dans un tel graphe.

Exercice 3. Cycle orienté

1. Donner un algorithme polynomial qui trouve le plus court cycle orienté dans un graphe orienté (ou retourne que le graphe est acyclique).
2. Donner un algorithme linéaire pour déterminer s'il existe un cycle orienté de longueur impair dans un graphe orienté.