Examen d'algorithmique

L3 – durée 3 heures

Mardi 10 janvier 2012

Les notes de cours et de TD sont autorisées, à l'exclusion de tout autre document. Les appareils électroniques sont interdits. Le barème est donné seulement à titre indicatif. Le sujet comporte 3 pages.

Exercice 1 *(ABR, 6 points)*

1. Donner un algorithme renvoyant la valeur maximale contenue dans un ABR. Quelle est sa complexité ?
2. Donner un algorithme calculant le nombre de nœuds internes d'un ABR. Quelle est sa complexité ?
3. À hauteur fixée, quelle forme ont les ABR ayant le minimum de nœuds ? Donner un algorithme testant si un ABR a le minimum de nœuds pour sa hauteur.
4. Donner un algorithme renvoyant la valeur de l'un des nœuds internes les plus profonds d'un ABR. Quelle est sa complexité ?
5. On modifie maintenant nos ABR de sorte que chaque nœud contient une information supplémentaire (outre sa valeur) : la hauteur du sous-arbre dont il est la racine.
   (a) Modifier l'algorithme d'insertion dans un ABR pour mettre à jour cette information.
   (b) Grâce à cette nouvelle information, donner un algorithme renvoyant la valeur de l'un des nœuds internes les plus profonds d'un ABR. Votre algorithme doit être plus performant qu'à la question 4. Quelle est sa complexité ?

Exercice 2 *(Graphes, 3 points)*

On suppose que n villes $v_1, \ldots, v_n$ sont reliées entre elles par un réseau routier formant un arbre (non orienté). Ainsi pour aller d'une ville v_i à une ville v_j, il y a un unique chemin.

L'arbre est donné sous la forme d'un tableau Π de taille n spécifiant le père de chaque sommet : $\Pi(i) = j$ si v_j est le père de v_i dans l'arbre (la racine n'a pas de père, c'est-à-dire que $\Pi(a) = \texttt{nil}$ si a désigne la racine de l'arbre).

1. Donner un algorithme qui, étant donné l'arbre Π et deux indices i et j, renvoie le chemin permettant d'aller de v_i à v_j.
2. Déterminer la complexité de votre algorithme.

Exercice 3 *(Graphes, 6 points)*

Des villes $v_1, \ldots, v_n$ sont reliées entre elles par un réseau routier représenté par un graphe orienté. Un arc de u à v porte deux valeurs : le temps nécessaire pour aller de u à v et la quantité d'essence utilisée (un nombre entier). L'objectif est de concevoir un algorithme qui, sur l'entrée (v, v', k), trouve un trajet de v à v' consommant moins de k litres d'essence et prenant le moins de temps possible sous cette contrainte.

Par exemple dans la figure 1, pour aller de a à c en consommant au maximum $k = 6$ litres, on peut emprunter les chemins a, b, c (5 litres) ou a, d, c (4 litres) mais pas a, c (10 litres). Le chemin de a à c prenant le moins de temps possible tout en consommant moins de k litres est alors a, b, c puisque le temps de parcours est 3 (tandis que le temps de parcours de a, d, c est 4).

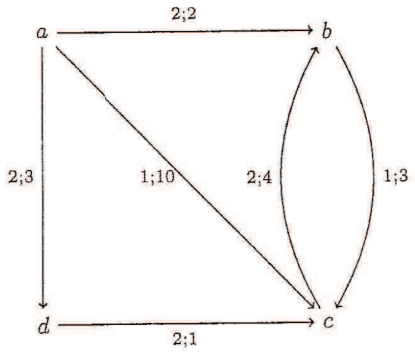

FIGURE 1 – Un réseau routier avec les temps de parcours t et l'essence consommée e indiqués sous la forme t;e.

1. Pourquoi l'algorithme de Dijkstra ne peut-il pas être utilisé pour ce problème ?

Le principe de l'algorithme va être celui de la programmation dynamique. Supposons qu'on veuille aller de la ville v à la ville v' avec moins de k litres. Pour toute ville w, tout entier $i \leq k$ et $j \leq n$, on va calculer $t_{w,i,j}$ correspondant au temps minimal pour aller de v à w en empruntant $\leq j$ arcs et en consommant $\leq i$ litres d'essence. S'il n'est pas possible d'aller de v à w en empruntant $\leq j$ arcs et en consommant $\leq i$ litres d'essence, alors $t_{w,i,j} = +\infty$.

2. Expliquer comment le calcul de toutes ces valeurs $t_{w,i,j}$ permet de connaître le temps minimal de parcours de v à v' avec au plus k litres d'essence.
3. Que valent $t_{w,i,0}$ et $t_{w,0,j}$ (pour tout i, j) selon la ville w ?
4. Expliquer comment calculer $t_{w,i,j}$ en fonction des valeurs $t_{w',i',j-1}$ où $i' \leq i$ et w' est arbitraire.
5. Donner un algorithme calculant tous les $t_{w,i,j}$ dans l'ordre de i et j croissant.
6. En déduire un algorithme qui, sur l'entrée (v, v', k), calcule le temps minimal de parcours de v à v' avec au plus k litres d'essence. Quelle est la complexité de votre algorithme ?
7. Comment modifier cet algorithme afin de connaître le chemin lui-même (et pas seulement le temps de parcours) ?

Exercice 4 *(Graphes, 5 points)*

Dans cet exercice, $G = (S, A)$ désigne un graphe non orienté. Une *couverture par sommets* (CPS) de G est un ensemble de sommets $S' \subseteq S$ tel que toute arête de G est incidente à au moins l'un des sommets de S'. Par exemple, le graphe de la figure 2 a une couverture par sommets de taille 2.

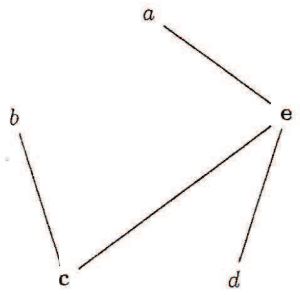

FIGURE 2 – Les deux sommets c et e couvrent toutes les arêtes.

Le but est de trouver une CPS de taille minimale. On considère l'algorithme suivant sur un graphe $G = (S, A)$, dans lequel S' désigne un ensemble de sommets et A' un ensemble d'arêtes :

```
Fonction algoCPS(G) :
1 S' ← ∅ ;
2 A' ← A ;
3 tant que A' ≠ ∅ faire
4       soit (u, v) une arête quelconque de A',
5       S' ← S' ∪ {u, v},
6       enlever de A' toutes les arêtes incidentes à u ou v ;
7 renvoyer S'.
```

Les questions suivantes portent sur cet algorithme.

1. Donner la complexité de cet algorithme.
2. Montrer que l'ensemble S' renvoyé par l'algorithme est une CPS.
3. Soit B l'ensemble des arêtes choisies à la ligne 4. Montrer que deux arêtes différentes de B ne sont pas adjacentes (c'est-à-dire qu'elle n'ont pas de sommet commun). Si B contient i arêtes, combien faut-il de sommets au minimum pour les couvrir toutes ?
4. En déduire que s'il existe une CPS de G de taille k, alors l'algorithme renvoie une CPS de taille $\leq 2k$.

L'algorithme donne donc une approximation de la solution optimale à un facteur 2.
En bonus, nous allons montrer que le problème de trouver une CPS de taille minimale est difficile.

5. *(Bonus optionnel à traiter seulement si vous avez tout fini)*
   On considère le problème de décision *Couverture* suivant :
   – Entrée : un graphe G et un entier k ;
   – Question : existe-t-il une CPS de taille $\leq k$?
   Montrer que le problème *Clique* vu en cours se réduit au problème *Couverture*. En déduire que *Couverture* est NP-complet.
   Indication : on pourra montrer que G a une clique de taille k ssi ${}^cG = (S, {}^cA)$ a une CPS de taille $(n - k)$.