Janvier 2006

Contrôle de techniques algorithmiques

Le contrôle de techniques algorithmiques dure 2h, les documents (cours, TD, TP) sont autorisés.

Il est demandé d'apporter beaucoup de soin à la rédaction des réponses aux questions posées et de **justifier** toutes les réponses. Les algorithmes demandés peuvent être rédigés en pseudo-langage (cf. le cours) ou en C.

Soit G = (X, U) un graphe orienté ayant n sommets. On interprétera un arc (x, y) comme un ordre entre ces sommets : x précède y ou x est avant y. L'objectif de ce contrôle est de construire une liste de tous les sommets de G telle que aucun sommet n'apparaisse avant un de ses prédécesseurs. Ceci s'appelle réaliser un **tri topologique** du graphe.

Question n° 1 : On considère le graphe de la figure 1. Effectuer un tri topologique de ce graphe. Y a-t-il une solution unique ?

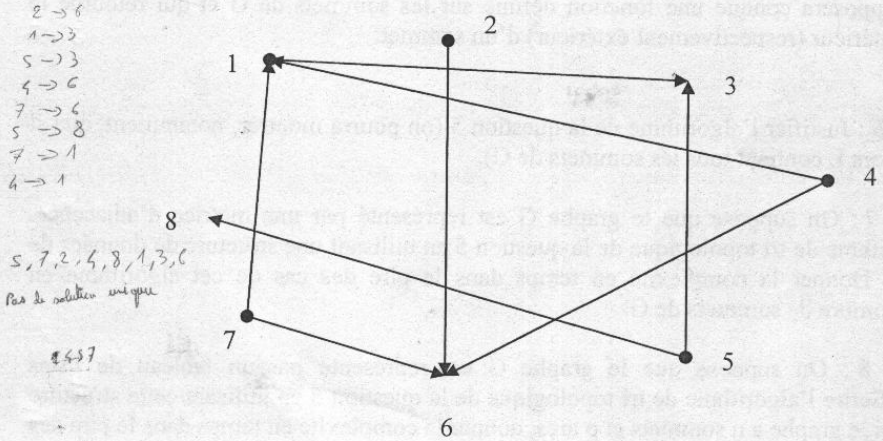


Figure 1

Question n° 2 : Ecrire une fonction degre_int(G : graphe ; x : sommet) qui rend le nombre d'arcs de G ayant pour extrémité le sommet x. Pour l'écriture de cette fonction, on distinguera :

1. le cas où le graphe G est représenté par une matrice d'adjacence ;
2. le cas où le graphe G est représenté par un tableau de listes d'adjacence.

Question n° 3 : Montrer que le problème du tri topologique a une solution si et seulement si le graphe G est sans circuit.

Question n° 4 : Proposer un algorithme **récursif** de tri topologique d'un graphe G sans circuit reposant sur l'idée suivante :

une liste est solution si son premier élément est un sommet sans antécédent (son demi-degré intérieur est nul) et si le reste de la liste est le résultat d'un tri topologique effectué sur le graphe G privé du premier élément de la liste.

Question n° 5 : Ecrire un algorithme **non récursif** réalisant le tri topologique du graphe G supposé sans circuit. On suggère de raisonner sur les degrés des sommets d'une suite de graphes construits à partir de G et d'utiliser les structures suivantes :

- Un tableau D ayant n éléments qui est construit de la manière suivante :
  D[s] = -1 si le sommet a déjà été supprimé
  D[s] = le demi-degré intérieur de s mis à jour à partir des sommets déjà supprimés

- Un ensemble M qui contient les sommets sans prédécesseurs du graphe concerné à l'étape courante de l'algorithme

- Une liste L qui contiendra le résultat du tri topologique.

On supposera connue une fonction définie sur les sommets de G et qui retourne le demi-degré intérieur (respectivement extérieur) d'un sommet.

Question n° 6 : Justifier l'algorithme de la question 5 (on pourra montrer, notamment, que si M est vide, alors L contient tous les sommets de G).

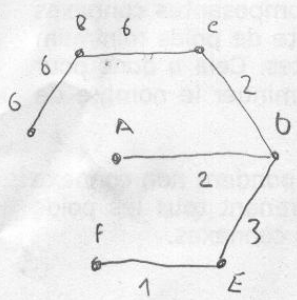
Question n° 7 : On suppose que le graphe G est représenté par une matrice d'adjacence. Ecrire l'algorithme de tri topologique de la question 5 en utilisant une structure de données de type matrice. Donner la complexité en temps dans le pire des cas de cet algorithme en fonction du nombre de sommets de G.

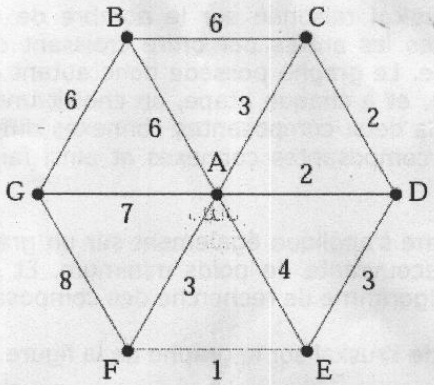
Question n° 8 : On suppose que le graphe G est représenté par un tableau de listes d'adjacence. Ecrire l'algorithme de tri topologique de la question 5 en utilisant cette structure de données. Si le graphe a n sommets et p arcs, donner la complexité en temps dans le pire des cas de cet algorithme.

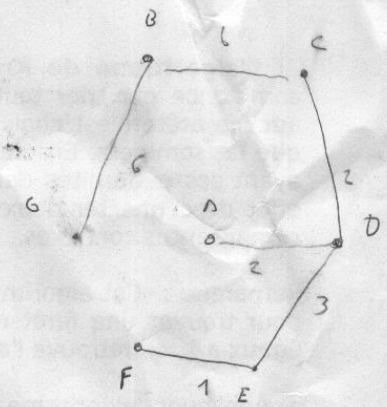
Question n° 9 (à ne faire que si toutes les questions 1 à 8 ont été traitées) : Comment modifier l'algorithme de la question 5 pour obtenir
- Soit le fait que le graphe a un circuit ;
- Un tri topologique sinon.

IF7 Algorithmique, complexité et programmation
L3 Informatique

Arbre de poids minimal







Soit G=(X,E) un graphe non orienté valué (ou pondéré) (v:E->R+, où v(e) est appelé le poids de l'arête e).

On rappelle que

- un arbre est un graphe non orienté, connexe et sans cycle;
- un graphe partiel G' de G est un graphe G'=(X,E') avec E' est un sous-ensemble de E

Le problème auquel on va s'intéresser est celui de la recherche, s'il existe, d'un graphe partiel de G qui soit un arbre recouvrant G de poids minimal. Un arbre recouvrant d'un graphe G connexe est un arbre joignant tous les sommets de G. Un arbre recouvrant est dit minimum si la somme des poids de ses arêtes est minimum sur l'ensemble des arbres recouvrants.

Nous allons étudier les deux algorithmes classiques de Kruskal et de Prim pour le calcul d'un arbre recouvrant de poids minimum (ARM). Ce sont des algorithmes gloutons : ils "engloutissent" les arêtes sans jamais revenir sur leurs choix. Ceci implique que lorsqu'ils choisissent une arête, ils sont sûrs qu'elle appartient à un ARM, et que cet ARM contient aussi les arêtes précédemment choisies. Les algorithmes gloutons ont souvent une complexité minimale.

rendez nous ses billes

1) Par definition, tous les sommets du graphe connexe doivent être reliés, donc le nombre d'arêtes d'un arbre recouvrant est exactement n - 1.

Que dire de l'algorithme qui choisit les n-1 arêtes de plus petits poids ? (testez-le sur le graphe pondéré de la figure ci-dessous)

2) **L'algorithme de Kruskal** raisonne sur le nombre de composantes connexes et commence par trier toutes les arêtes par ordre croissant de leurs poids. Au départ, aucune arête n'est choisie. Le graphe possède donc autant de composantes connexes que de sommets. Ensuite, et à chaque étape, on choisit une arête de poids minimum ayant ses extrémités dans deux composantes connexes différentes. Cela a donc pour effet de connecter deux composantes connexes et ainsi faire diminuer le nombre de composantes connexes.

Remarque : Cet algorithme s'applique également sur un graphe pondere non connexe pour trouver une forêt recouvrante de poids minimum. Et en prenant tous les poids égaux a 1, on retrouve l'algorithme de recherche des composantes connexes.

a) Appliquer l'algorithme de Kruskal sur le graphe de la figure.

b) Écrivez l'algorithme de Kruskal.

3) L'algorithme de Kruskal construit un ARM en intégrant une à une les arêtes du graphe. L'algorithme de Prim procède d'une toute autre manière : il intègre les sommets un par un. Que dire de l'algorithme qui choisit pour chaque sommet l'arête incidente de poids minimum? (testez-le sur le graphe de la figure)

4) Dans **l'algorithme de Prim**, on commence par choisir un sommet au hasard que l'on place dans un ensemble Y, puis à chaque étape on sélectionne une arête de poids minimum (a, b), avec a appartenant à Y et b n'appartenant pas à Y, et on place b dans Y.

a) Appliquer l'algorithme de Prim sur le graphe de la figure.

b) Écrivez l'algorithme de Prim.

5) Étudiez la complexité de ces deux algorithmes, en différenciant la représentation du graphe.

a) par liste d'adjacence

b) par matrice d'adjacence