

# Partiel MPRI Algorithmes de graphes

Michel Habib

18 novembre 2008

Les trois parties sont indépendantes.

## 1 Largeur d'arborescence

1. Parmi les classes de graphes suivantes, lesquelles sont closes par l'opération  $H$  mineur de  $G$  ?  
Graphes d'intervalles, cographes, forêts, chordaux, distances héréditaires, planaires.
2. Montrer que si  $H$  mineur de  $G$  alors  $treewidth(H) \leq treewidth(G)$
3. Est-il vrai que si  $H$  mineur de  $G$  alors  $pathwidth(H) \leq pathwidth(G)$  ?
4. Modifions un peu la définition des brambles vue en cours :

**Definition 1**  $G = (V, E)$  ; une famille de parties  $\mathcal{B} = \{X_i | X_i \subseteq V\}$  est appelée un **pseudobramble** si elle vérifie :

$G(X_i)$  connexe et  $\forall i, j$   $G(X_i \cup X_j)$  connexe.

$\forall i \neq j$   $X_i \cap X_j = \emptyset$  (**restriction ajoutée aux brambles**)

Un transversal d'un pseudobramble est un ensemble de sommets  $\tau \subseteq V$  tel que  $\forall i$   $\tau \cap X_i \neq \emptyset$

On pose  $pseudobr(G) = \text{Max}_{\mathcal{B}}(\min_{\tau \text{ transversal de } \mathcal{B}} |\tau|)$

Montrer que pour tout graphe  $G$ ,  $treewidth(G) \geq pseudobr(G) - 1$

A-t-on l'égalité ?

## 2 Graphes triangulés

1. On considère  $\mathcal{C}(G)$  le graphe des cliques maximales d'un graphe triangulé (les sommets sont les cliques maximales et deux cliques maximales sont reliées par une arête si leur intersection est un séparateur minimal).  
Montrer qu'étant donné 3 cliques maximales  $C_1, C_2, C_3$  de  $G$ , si  $S = C_1 \cap C_2 = U = C_2 \cap C_3$  sont des arêtes de  $\mathcal{C}(G)$ , alors soit  $C_1 C_3 \in \mathcal{C}(G)$  soit les deux arêtes  $C_1 C_2, C_2 C_3$  ne peuvent appartenir à un même arbre de cliques maximales.
2. \* Montrer que si  $G$  est un graphe d'intervalle, alors la dernière clique visitée par un parcours LexBFS est une clique maximale qui peut être choisie comme la première dans une représentation intervallaire de  $G$ .

### 3 Algorithmes de graphes et structures de données

Nous avons vu en cours l'intérêt des procédures par affinage de partitions pour l'algorithme de graphes.

#### Définition d'un affinage standard pour les graphes :

On considère une partition  $P$  des sommets du graphe. On choisit un sommet  $z$  et l'on affine  $P$  suivant le voisinage  $N(z)$  de  $z$  (i.e. chaque classe  $C$  de la partition  $P$ , **exceptée la classe  $C_z$  contenant  $z$**  est réécrite en  $C - N(z)$  et  $C \cap N(z)$ ).

L'idée d'implémentation présentée en cours est la suivante : La partition des sommets du graphe est représentée à l'aide d'une liste doublement chaînée. On maintient d'autre part un pointeur qui relie chaque sommet (i.e. de la structure de données qui représente le graphe) vers sa position dans la partition. Par ailleurs les classes de la partition sont aussi implémentées à l'aide d'une liste doublement chaînée. En outre, chaque élément de la partition possède un pointeur vers sa classe. Il existe bien d'autres façons de faire, par exemple en utilisant un tableau pour représenter la partition.

Cette procédure peut se réaliser en  $O(|N(z)|)$ , on suppose dans ce cas que la donnée  $N(x)$  est une liste de sommets.

Dans les questions suivantes il s'agit de préciser la structure que vous allez choisir afin de garder la complexité en  $O(|N(z)|)$ , tout en garantissant les spécifications additionnelles définies ci-après.

1. Comment faire si l'on veut implémenter la règle de Hopcroft, i.e. ne garder que la partie la plus petite dans la liste des classes à traiter ?
2. Dans une autre application importante de cette technique, la recherche d'une orientation transitive d'un graphe de comparabilité, la partition manipulée est ordonnée (Gauche–Droite). Ainsi à la fin de l'algorithme, lorsque toutes les classes sont des singletons, cet ordre permet de construire une orientation du transitive du graphe.

Dans ce contexte, la règle d'affinage devient :

- Si la classe  $C$  est à gauche de  $C_z$  elle se réécrit en  $C \cap N(z)$  et  $C - N(z)$ .
- Si la classe  $C$  est à droite de  $C_z$  elle se réécrit en  $C - N(z)$  et  $C \cap N(z)$ .

Comment implémenter cette notion de gauche, droite ?

3. Il est naturel d'associer à toute procédure d'affinage de partitions un arbre dont la racine est la partition triviale  $P_0 = \{X\}$ . Préciser cet arbre et proposer un algorithme qui le calcule.
4. Retour arrière (backtrack)  
Proposer un algorithme qui permette de défaire un affinage qui a déjà été réalisé.  
Montrer que lorsque c'est le dernier réalisé, il est possible de le faire en  $O(|N(z)|)$ .  
Est-ce possible dans le cas général avec la même complexité ?