



Concours d'accès au Doctorat LMD Informatique, 2012/2013
Epreuve d'Analyse et complexité des algorithmes

USTHB le 26/11/2012

Exercice 1 : structure de données et algorithme de tri (10 pts)

La structure de données *tas* ou *heap* est une structure d'optimisation de l'espace mémoire et est utilisée par exemple pour trier un tableau d'entiers. Rappelons qu'un tas est un arbre binaire équilibré dans lequel chaque valeur associée à un nœud est supérieure aux valeurs associées respectivement à ses fils s'ils existent. De plus, un tas est représenté à l'aide d'un vecteur A en supposant les hypothèses suivantes :

- les fils de l'élément $A[i]$ se trouvent respectivement au niveau des positions $2*i$ et $2*i+1$, s'ils existent bien entendu.
- $A[i]$ est par conséquent une feuille si $2*i > n$
- 1. Quelle première conséquence peut-on tirer d'un tas ?
- 2. Pour trier un vecteur, ce dernier est d'abord transformé en un tas dans le but d'extraire le plus grand élément. Cette opération est renouvelée pour le reste du tableau. L'idée principale pour obtenir un tas est de permuter l'élément se trouvant à un nœud par le fils qui a la plus grande valeur.
 - a. Ecrire l'algorithme de construction d'un tas à partir d'un vecteur quelconque.
 - b. Calculer sa complexité.
 - c. Illustrer l'algorithme sur le tableau contenant les éléments suivants: 11,73,29,45,6,31,52,89,93,9
- 3. Ecrire un algorithme pour rechercher un élément dans un tas. Calculer sa complexité.
- 4. Ecrire un algorithme pour insérer un élément dans un tas. Calculer sa complexité. Illustrer votre algorithme en insérant l'entier 90 dans le tas construit en 2)

Exercice 2 : NP-complétude (10 pts)

Une proposition atomique est une variable booléenne, c'est-à-dire prenant ses valeurs dans l'ensemble $BOOL = \{VRAI, FAUX\}$. Un littéral est une proposition atomique ou la négation d'une proposition atomique. Une proposition atomique est aussi appelée littéral positif ; et la négation d'une proposition atomique littéral négatif. Une clause est une disjonction de littéraux.

Etant données m propositions atomiques p_1, \dots, p_m , une instantiation du m -uplet (p_1, \dots, p_m) est un élément de $\{VRAI, FAUX\}^m$. Une instantiation (e_1, \dots, e_m) de (p_1, \dots, p_m) satisfait une clause c (noté $(e_1, \dots, e_m) \models c$) si et seulement si l'une des conditions suivantes est satisfaite :

1. il existe $i \in \{1, \dots, m\}$ tel que $(e_i = VRAI)$ et $(p_i$ occure dans $c)$
2. il existe $i \in \{1, \dots, m\}$ tel que $(e_i = FAUX)$ et $(\neg p_i$ occure dans $c)$

Une instantiation satisfait une conjonction de clauses si et seulement si elle satisfait chacune de ses clauses. Une conjonction de clauses est satisfiable si et seulement si il existe une instantiation la satisfaisant. Une instantiation satisfaisant une conjonction est dite solution ou modèle de la conjonction.

Le problème SAT est maintenant défini comme suit :

Description : une conjonction C de n clauses construites à l'aide de m propositions atomiques p_1, \dots, p_m

Question : la conjonction C est-elle satisfiable ?

Le but de l'exercice est de montrer que le Problème SAT appartient à la classe de complexité NP, la classe des problèmes de décision non déterministes polynômiaux. Il faut pour ce faire trouver un algorithme polynômial de validation pour le problème, que vous appellerez *validation_s*. Il vous est demandé de procéder comme suit :

1. Donner une structure de données permettant de représenter une instance du problème SAT. Expliquer les paramètres.
2. Donner l'algorithme *validation_s* sous forme d'une fonction booléenne dont il est important que vous expliquiez les paramètres.
3. Calculer le nombre d'opérations élémentaires dans le pire des cas de l'algorithme.
4. Montrer que l'algorithme est polynômial.

BON COURAGE !

1/1