Université des Sciences et de la Technologie Houari Boumediène
Date : 26/11/2012

Concours d'accès au doctorat
Filière Informatique
Epreuve 'Complexité'
Corrigé de l'exercice 2

Exercice 2 : NP-complétude
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 $p_1, \ldots, p_m$, une instanciation du m-uplet $(p_1, \ldots,p_m)$ est un élément de $\{VRAI,FAUX\}^m$. Une instanciation $(e_1, \ldots, e_m)$ de $(p_1, \ldots,p_m)$ satisfait une clause c (noté $(e_1, \ldots, e_m) \models c$) si et seulement si l'une des conditions suivantes est satisfaite :

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

Une instanciation 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 instanciation la satisfaisant. Une instanciation 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 $p_1,\ldots,p_m$
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. Donnez une structure de données permettant de représenter une instance du problème SAT. Expliquez
2. Donnez l'algorithme validation_s sous forme d'une fonction booléenne dont il est important que vous expliquiez les paramètres.
3. Calculez le nombre d'opérations élémentaires dans le pire des cas de l'algorithme.
4. Montrez que l'algorithme est polynômial.

Corrigé :

1. Soit C une instance du problème SAT, donnée par une conjonction de n clauses $c_1, \ldots, c_n$ construites à partir de m propositions atomiques $p_1, \ldots, p_m$. L'instance peut être représentée par une matrice nxm dont les éléments appartiennent à l'ensemble {0,1,2}, et vérifient les points suivants :
   - Pour tout i=1 à n, pour tout j=1 à m :
     - Si p_j occurre négativement dans la clause c_i alors M[i,j]=0
     - Si p_j occurre positivement dans la clause c_i alors M[i,j]=1
     - Si p_j n'occurre ni négativement ni positivement dans la clause c_i alors M[i,j]=2

   Ainsi donc, la ième ligne de la matrice représente la ième clause de la conjonction C ; et le jème élément de la ligne i indique s'il y a occurrence de la proposition p_j dans la clause c_i, et si, quand occurrence il y a, celle-ci est négative ou positive.
2. L'algorithme de validation validation-s est comme suit. Il est écrit sous forme d'une fonction booléenne à quatre paramètres n, m, C et inst. Les paramètres n et m

correspondent, respectivement, aux nombre de clauses et nombre de propositions atomiques à partir desquelles les clauses sont construites. Le paramètre C est une matrice nxm représentant la conjonction donnant l'instance du problème SAT. Le paramètre inst, enfin, est un certificat consistant en un tableau de taille m de booléens représentant une instanciation du m-uplet $(p_1,\ldots,p_m)$: pour tout i=1 à m, si inst[i]=VRAI alors l'instanciation représentée associe la valeur de vérité VRAI à la proposition atomique p_i ; et si inst[i]=FAUX, l'instanciation représentée associe la valeur de vérité FAUX à p_i.

```
Booléen validation_s(n,m,C,inst)
        début
        1.      solution=VRAI ;                 /* certificat supposé solution */
        2.      i=1                             /* i parcourt les lignes (clauses) */
        3.      tant que (solution et i≤n) faire
        4.              c_satisfaite=FAUX;      /* ième clause supposée non satisfaite */
        5.              j=1                     /* j parcourt les colonnes (propositions atomiques) */
        6.              tant que ((non c_satisfaite) et j≤m) faire
        7.                      si ( (inst[j]=FAUX et C[i,j]=0) ou (inst[j]=VRAI et C[i,j]=1) )
        8.                              alors c_satisfaite=VRAI
        9.                      sinon j=j+1
                                fsi
                        fait
        10.             si (non c_satisfaite) alors solution=FAUX sinon i=i+1 fsi
                fait
        11.     retourner solution
        Fin
```

3. Le tableau ci-dessous donne le nombre f(n,m) d'opérations élémentaires dans le pire des cas de l'algorithme :

Instruction	(Majorant du) nombre d'opérations élémentaires à chaque exécution	(Majorant du) nombre de fois que l'instruction est exécutée
1	1	1
2	1	1
3	2	n+1
4	1	n
5	1	n
6	2	n*(m+1)
7, 8 et 9	6	n*m
10	3	n
11	1	1

   f(n)=5+9n+8nm

4. L'algorithme de validation est quadratique en n et m. En effet, on peut montrer que $f(n,m)=\Theta(nm)$
   a. $f(n,m)=O(nm)$?
      Trouver n_0, m_0 et $c_0 \geq 0$ tels que $\forall n \geq n_0\ \forall m \geq m_0\ f(n,m) \leq c*nm$
      $8nm+9n+5 \leq c*nm$
      $8+9/m+5/(nm) \leq c$
      Prendre : **c=22 ; n_0=1 ; m_0=1**
   b. $nm=O(f(n,m))$?
      Trouver n_0, m_0 et $c_0 \geq 0$ tels que $\forall n \geq n_0\ \forall m \geq m_0\ nm \leq c*f(n,m)$
      $nm \leq c*(8nm+9n+5)$
      $1/c \leq 8+9/m+5/(nm)$
      Prendre : **1/c=8 (donc c=1/8) ; n_0=1 ; m_0=1**

Remarque : montrer seulement que $f(n,m)=O(nm)$ est suffisant