Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique
Département d'Informatique
LMD Master 1ère Année ''Réseaux et Systèmes Distribués'' 2010/2011
Module ''Algorithmique Avancée et Complexité''
Date : 13/01/2011

Corrigé de l'interrogation

Exercice 1 (NP-complétude) :

On considère le problème de décision CLIQUE suivant :

- **Description :** un graphe G et une constante entière k
- **Question :** G admet-il une clique de taille k ? une clique de taille k d'un graphe est un sous-graphe complet à k noeuds.

Le but de l'exercice est de trouver un algorithme polynômial de validation pour le problème CLIQUE ci-dessus. Pour ce faire, il vous est demandé de procéder comme suit :

1. Donnez un algorithme de validation pour le problème CLIQUE, que vous appellerez validation_c. L'algorithme, bien évidemment, doit être polynômial, la preuve de la polynômialité faisant l'objet des questions 2 et 3. Ecrivez l'algorithme sous forme d'une fonction booléenne dont il est important que vous expliquiez les paramètres.
2. Calculez le nombre d'opérations élémentaires de l'algorithme validation_c en fonction d'une taille n à préciser. Appelez ce nombre T(n).
3. Montrez que $T(n)=\Theta(n^k)$, pour une certaine constante k à préciser.
4. L'existence d'un algorithme polynômial de validation pour un problème de décision suffit-elle pour dire que le problème est NP-complet ? Expliquez.

Solution :

1. L'algorithme de validation est comme suit. Il est écrit sous forme d'une fonction booléenne validation_c à quatre entrées n, C, k et c. Le triplet (n,C,k) donne l'instance du problème (le nombre n de nœuds du graphe, une matrice carrée booéenne nxn représentant la matrice d'adjacence du graphe, et un entier k inférieur ou égal à n). c est un tableau de taille k d'entiers tous différents entre 1 et n (l'algorithme retourne VRAI si et seulement si le certificat valide l'instance, c'est-dire si et seulement si le sous-graphe constitué des nœuds $X_{c[1]}$, $X_{c[2]}$, …, $X_{c[k]}$ forme une clique de G).

   Si un entier est représenté sur p bits, le quadruplet (n,C,k,c) peut être vu comme un mot de $\{0,1\}^*$ de longueur $p*(1+n^2+1+k)$: les p premiers bits (0 ou 1) coderont le nombre n de nœuds du graphe, les $p*n^2$ suivants coderont la matrice C d'adjacence du graphe, les p suivants coderont l'entier k, les p*k derniers bits coderont le certificat c.

```
Booléen validation_c(n,C,k,c)
        début
                pour i=1 à k faire                              (1)
                        pour j=1 à k faire                      (2)
                                si C[c[i],c[j]]=0               (3)
                                        alors retourner FAUX    (4)
                                fsi
                        fait
                fait
        retourner VRAI                                          (5)
        fin
```

2. Le nombre T(k) d'opérations élémentaires dans le pire cas de l'algorithme de validation validation_c ci-dessus est calculée comme suit :

Instruction	Nombre d'opérations		Nombre de fois
(1)	3	1 addition, 1 affectation, 1 comparaison	K
(2)	3	1 addition, 1 affectation, 1 comparaison	K^2
(3)	1	1 comparaison	K^2
(4)	1	1 retour	1
(5)	1	1 retour	1

Donc :
$T(k)= 3k+3K^2+k^2+1=4k^2+3k+1$

3. On montre que $T(k)=\Theta(k^2)$, ce qui est équivalent à montrer que $T(k)=O(k^2)$ et $k^2=O(T(k))$.
   a. $T(k)=O(k^2)$?
      Il faut trouver $c\geq 0$ et k_0 tels que pour tout $k\geq k_0$ $T(k)\leq c*k^2$:
      $4k^2+3k+1\leq c*k^2$
      $4+\frac{3}{k}+\frac{1}{k^2}\leq c$
      $c=8$ et $k_0=1$
   b. $k^2=O(T(k))$?
      Il faut trouver $c\geq 0$ et k_0 tels que pour tout $k\geq k_0$ $k^2\leq c*T(k)$:
      $k^2\leq c*(4k^2+3k+1)$
      $\frac{1}{c}\leq 4+\frac{3}{k}+\frac{1}{k^2}$
      c et k_0 tels que $\frac{1}{c}=4$:
      $c=\frac{1}{4}$ et $k_0=1$
4. L'existence d'un algorithme polynomial de validation pour un problème de décision ne permet pas de dire que le problème est NP-complet. Elle permet de dire que le problème est dans la classe NP.

   Pour montrer que le problème est NP-complet, il faut, en plus de l'appartenance à la classe NP (existence d'un algorithme polynomial de validation), montrer que tout autre problème NP peut se ramener à ce problème via une réduction polynomiale ; ou, de façon équivalente, qu'il existe un problème NP-complet pouvant se ramener à ce problème via une réduction polynomiale.

Exercice 2 (ordre de complexité) :

Montrez que pour toute constante entière k :

1. $n^k=O(n^k\log n)$
2. $n^k\log n=O(n^{k+1})$

Solution 1 (notations de Landau) :

1. $n^k=O(n^k\log n)$?
   Il faut trouver $c\geq 0$ et n_0 tels que pour tout $n\geq n_0$ $n^k\leq c*(n^k\log n)$:
   $n^k\leq c*(n^k\log n)$
   $\frac{1}{c}\leq \log n$ (la fonction log est strictement croissante)

c et n_0 tels que $\frac{1}{c}$=1 (1=log10) :

c=1 et n_0=10

2. $n^k \log n = O(n^{k+1})$?

   Il faut trouver c≥0 et n_0 tels que pour tout $n \geq n_0$ $n^k \log n \leq c * n^{k+1}$:

   $n^k \log n \leq c * n^{k+1}$

   $\frac{\log n}{n} \leq c$ (la dérivée de $\frac{\log n}{n}$ est $\frac{\frac{1}{\ln 10} - \log n}{n^2}$, toujours négative à partir de n=10, donc $\frac{\log n}{n}$ strictement décroissante à partir de n=10)

   c et n_0 tels que $c=\frac{1}{10}$ ($\frac{1}{10}=\frac{\log 10}{10}$) :

   $c=\frac{1}{10}$ et n_0=10

Solution 2 (utiliser les limites quand la taille n tend vers l'infini) :

1. $n^k = O(n^k \log n)$?

   vrai car $\lim_{n\to\infty} \frac{n^k}{n^k \log n} = \lim_{n\to\infty} \frac{1}{\log n} = 0$ (à partir d'une certaine valeur de n, n^k est négligeable devant $n^k \log n$)

2. $n^k \log n = O(n^{k+1})$?

   vrai car $\lim_{n\to\infty} \frac{n^k \log n}{n^{k+1}} = \lim_{n\to\infty} \frac{\log n}{n} = 0$ (à partir d'une certaine valeur de n, $n^k \log n$ est négligeable devant n^{k+1})

Exercice 3 (Structures de données) :

Une file est une structure de données mettant en œuvre le principe « premier entré premier sorti » (FIFO : First In First Out). On considère ici le cas d'une file implémentée avec un tableau.

1. Une file doit être initialisée. Expliquez comment.
2. Ecrivez les différentes fonctions et procédures permettant la gestion d'une file.

Solution :

1. La file est implémentée avec un tableau F. On initialise la taille n de F à 100, par exemple ; la tête tête(F) de F à -1 (initialement, la file est vide) ; et la queue queue(F) de F à 1 (quand la file est vide, l'insertion d'un élément se fera à la position 1). De plus, on suppose que les éléments de la file sont indicés de 1 à 100 :

```
n=100 ;
tête(F)=-1 ;
queue(F)=1 ;
```

2. Les différentes fonctions et procédures permettant la gestion d'une file :

```
FILE-VIDE(F){
si tête(F)=-1
        alors retourner VRAI
        sinon retourner FAUX
}

INSERTION(F,x){
si [tête(F)≠-1 et queue(F)=tête(F)]
        alors erreur (débordement positif)
```

```
        sinon{
                F[queue(F)]=x
                queue(F)=[queue(F)+1](modulo n)
                si[tête(F)=-1] alors tête(F)=1
                }
}

SUPPRESSION(F){
si FILE-VIDE(F)
        alors erreur (débordement négatif)
        sinon{
                temp=F(tête(F));
                tête(F)=[tête(F)+1](modulo n);
                si[tête(F)=queue(F)]{
                        tête(F)=-1 ;
                        queue(F)=1;
                        }
                retourner temp;
                }
}
```