Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique
Département d'Informatique
LMD Master 1ère Année IL 2012/2013
Module ''Algorithmique Avancée et Complexité''
Date : 08/01/2013

Corrigé de l'épreuve de moyenne durée

Exercice 1 <5 points=2+3>:

1. Rappelez les définitions des notations de Landau O et Θ.
2. Soit $f(n)=3n^2.\log(n)$ le nombre d'opérations élémentaires du pire cas d'un algorithme A.
   a. Montrez que $f(n)=O(n^4)$.
   b. A-t-on $f(n)=\Theta(n^4)$? Expliquez.

Solution :

1) Voir cours
2)
   a. Pour montrer que $f(n)=O(n^4)$, il suffit de trouver n_0 et $c\geq 0$ (ou de montrer leur existence) tels que $\forall n\geq n_0$ $f(n)\leq c*n^4$:
   $3n^2.\log(n)\leq c*n^4$ //on divise les deux membres par n^4
   $\frac{3\log(n)}{n^2}\leq c$
   $\lim_{n\to+\infty}\frac{3\log(n)}{n^2}=0$ donc le c et le n_0 demandés existent.
   Conclusion : on a bien $f(n)=O(n^4)$
   b. Pour avoir $f(n)=\Theta(n^4)$, vu qu'on a déjà $f(n)=O(n^4)$, il faut avoir en plus $n^4=O(f(n))$. Tentons donc de montrer $n^4=O(3n^2.\log(n))$ en cherchant n_0 et $c\geq 0$ tels que $\forall n\geq n_0$ $n^4\leq c*(3n^2.\log(n))$:
   $n^4\leq c*(3n^2.\log(n))$ //on divise les deux membres par $c*n^4$
   $\frac{1}{c}\leq\frac{3\log(n)}{n^2}$
   $\lim_{n\to+\infty}\frac{3\log(n)}{n^2}=0$ donc la constante c demandée n'existe pas.
   Conclusion : $f(n)\neq\Theta(n^4)$

Exercice 2 <5 points>:

Donnez un algorithme polynomial permettant la construction, à partir d'un arbre binaire de recherche, d'un autre arbre binaire de recherche en supprimant de ce dernier tous les nœuds dont les clés ne sont pas dans un intervalle [a,b] donné. Justifiez la polynômialité de l'algorithme.

Solution 1 :

```
infixe(A)
début
    si (A≠NIL) alors
        infixe(A.sa-gauche);
        si (A.clef in [a,b]) alors
            créer nœud z;z.clef=A.clef ;z.sa-gauche=NIL;z.sa-droit=NIL;
            si X=NIL alors X=z sinon père_de_z.sa-droit=z ;pere_de_z=z finsi
        finsi
        Infixe(A.sa-droit);
    rinsi
    retourner X
fin
```

Programme principal
```
début
    X=NIL ; B=infixe(A) ;
fin
```

Exercice 3 <4 points>:
Donnez un algorithme linéaire qui teste si un tableau de taille n est un « tableau de permutation » (i.e., tous les éléments sont distincts et compris entre 1 et n). Justifiez la linéarité de l'algorithme.

Indication : Utilisez un tableau auxiliaire.

Solution :
Soit T un tableau d'entiers de taille n dont on veut savoir s'il est un tableau de permutation ; on utilise un tableau auxiliaire S de taille n dont tous les éléments seront initialisés à 0 (pour i=1 à n, S[i]=0 signifiera que l'élément i n'a pas encore été rencontré dans le tableau T ; dès que i est rencontré dans T, S[i] sera mis à 1 pour que la deuxième fois qu'on rencontre i dans T, si toutefois deuxième fois il y a, on le sache).

```
booléen tab_perm(T,n)
début
pour i=1 à n faire S[i]=0 fait
pour i=1 à n faire
    si T[i]<1 ou T[i]> n alors retourner FAUX
    sinon   si S[T[i]]=1 alors retourner FAUX
            sinon S[T[i]]=1
            finsi
    finsi
fait
retourner VRAI
fin
```

Le nombre d'opérations élémentaires de l'algorithme est clairement un polynôme de degré 1 en n. L'algorithme est donc bien linéaire.

Exercice 4 <6 points=1+1+2+1+1>:
On considère le problème de décision P suivant :

- **Description :** un graphe non orienté G=(V,E) et un entier positif $k \leq |V|$.
- **Question :** Existe-t-il un sous-ensemble V' de V vérifiant $|V'| \leq k$ de telle sorte que pour toute arête (u,v) de G, $u \in V'$ ou $v \in V'$?

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

1. Donnez une structure de données permettant de représenter une instance du problème P. Expliquez
2. Expliquez la notion de certificat d'une instance du problème P. Donnez une structure de données permettant la représentation d'un tel certificat. Expliquez
3. Donnez un algorithme de validation pour le problème P, que vous appellerez validation_P. L'algorithme, bien évidemment, doit être polynômial, la preuve de la polynômialité faisant l'objet des questions 4 et 5. Ecrivez l'algorithme sous forme d'une fonction booléenne dont il est important que vous expliquiez les paramètres.
4. Calculez le nombre d'opérations élémentaires de l'algorithme validation_P en fonction d'une taille n à préciser. Appelez ce nombre T(n).
5. Montrez que $T(n)=\Theta(n^k)$, pour une certaine constante k à préciser.

Solution :

1. Une instance du problème de décision P est un graphe non orienté G=(V,E) à n sommets (nous supposons $V=\{v_1,\ldots,v_n\}$), et un entier positif $k\leq n$; instance que nous représentons par un triplet (M_G,n,k), M_G étant la matrice d'adjacence de G. M_G est une matrice carrée booléenne nxn vérifiant MG[i,j]=1 si et seulement si (v_i,v_j) est arête de G (en particulier, la diagonale de M_G est à 0, et la matrice M_G est symétrique).
2. Un certificat d'une instance du problème P est un sous-ensemble V' de V de taille inférieure ou égale à k, que nous représentons par un tableau c de taille n de booléens : c[i]=1 si et seulement si $v_i \in V'$.
3. nous donnons ci-après, sous forme d'une fonction booléenne, un algorithme de validation validation_P pour le problème de décision P : l'algorithme aura comme arguments un certificat c et une instance (M_G,n,k) de P :

```
booléen validation_P(c,M_G,n,k)
début
    pour i=1 à n faire
        pour j=1 à n faire
                si (M_G[i,j]=1 et c[i]=0 et c[j]=0) alors retourner FAUX finsi
        fait
    fait
    retourner VRAI
fin
```

4. Le nombre T(n) d'opérations élémentaires du pire cas de l'algorithme de validation est clairement un polynôme de degré 2 en n.
5. T(n) polynôme de degré 2 en n donc $T(n)=\Theta(n^2)$: l'algorithme de validation est bien polynômial, et le problème P appartient donc à la classe NP.