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 : 06/01/2013

Corrigé de l'interrogation

Exercice 1 <5 points=2,5+2,5>:
Soit $f(n)=20n^3+10n^4+3n^2.2^n$ le nombre d'opérations élémentaires du pire cas d'un algorithme A.

1. Montrez que $f(n)=O(n^2.2^n)$.
2. A-t-on $f(n)=\Theta(n^2.2^n)$? Expliquez.

Solution :

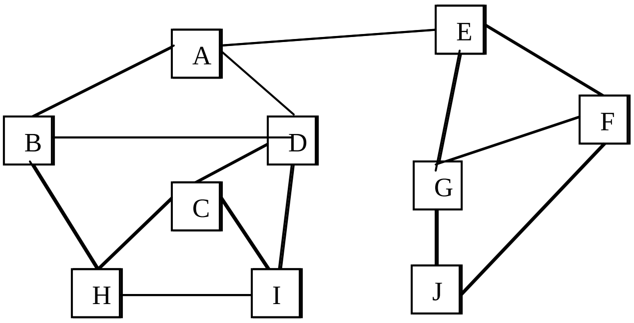
1) Pour montrer que $f(n)=O(n^2.2^n)$, il suffit de trouver n_0 et $c\geq 0$ (ou de montrer leur existence) tels que $\forall n\geq n_0$ $f(n) \leq c*n^2.2^n$:
$20n^3+10n^4+3n^2.2^n\leq c*n^2.2^n$ //on divise les deux membres par $n^2.2^n$
$\frac{20n}{2^n}+\frac{10n^2}{2^n}+3\leq c$ //$\lim_{n\to+\infty}\frac{20n}{2^n}=\lim_{n\to+\infty}\frac{10n^2}{2^n}=0$ donc le c et le n_0 demandés existent !
Conclusion : on a bien $f(n)=O(n^2.2^n)$

2) Pour avoir $f(n)=\Theta(n^2.2^n)$, vu qu'on a déjà $f(n)=O(n^2.2^n)$, il faut avoir en plus $n^2.2^n=O(f(n))$. Tentons donc de montrer $n^2.2^n=O(20n^3+10n^4+3n^2.2^n)$ en cherchant n_0 et $c\geq 0$ tels que $\forall n\geq n_0$ $n^2.2^n\leq c*(20n^3+10n^4+3n^2.2^n)$:
$n^2.2^n\leq c*(20n^3+10n^4+3n^2.2^n)$ //on divise les deux membres par $c*n^2.2^n$
$\frac{1}{c}\leq\frac{20n}{2^n}+\frac{10n^2}{2^n}+3$
Prendre $\frac{1}{c}=3$ et $n_0=1$; donc $c=\frac{1}{3}$ et $n_0=1$
Conclusion : $f(n)=\Theta(n^2.2^n)$

Exercice 2 <4 points>:
Soient A1, A2, A3 et A4 quatre algorithmes conçus pour la même tâche T, et dont les nombres d'opérations élémentaires du pire cas sont, respectivement, $f_1(n)=3n^5\log n$, $f_2(n)=n!$, $f_3(n)=2^n$ et $f_4(n)=10n^5$. Comparez les complexités des quatre algorithmes. Expliquez.

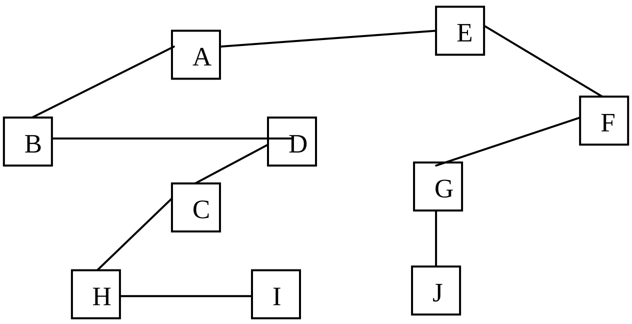
Solution :
$f_1(n)=$ $)=\Theta(n^5\log n)$; $f_2(n)=\Theta(n!)$; $f_3(n)=\Theta(2^n)$; $f_4(n)=\Theta(n^5)$
$\lim_{n\to+\infty}\frac{n^5}{n^5\log n}=\lim_{n\to+\infty}\frac{n^5\log n}{2^n}=\lim_{n\to+\infty}\frac{2^n}{n!}=0$ donc $\Theta(n^5)<\Theta(n^5\log n)<\Theta(2^n)<\Theta(n!)$ (l'algorithme A4 est meilleur que l'algorithme A1, qui est meilleur que l'algorithme A3, qui est meilleur que l'algorithme A2)

Exercice 3 <5 points=2+1,5+1,5>:
On considère le graphe suivant :

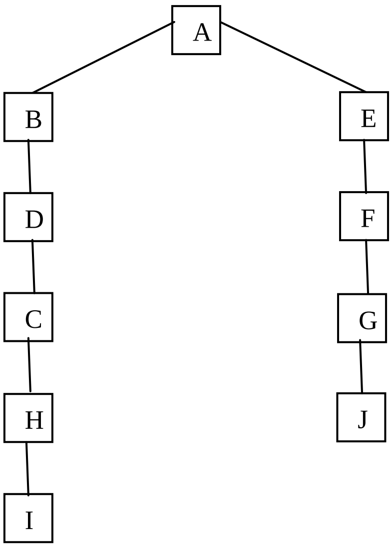


1. Donnez un arbre de recouvrement issu d'un parcours en profondeur d'abord du graphe
2. Donnez l'ordre dans lequel le parcours considéré a visité les sommets du graphe
3. Déduire des questions précédentes l'arbre de recouvrement ordonné issu du parcours considéré : le parcours en profondeur d'abord de l'arbre ordonné doit visiter les sommets dans l'ordre dans lequel ils l'ont été par le parcours considéré du graphe

Solution :
1.



2. ABDCHIEFGJ
3.



Exercice 4 <6 points=1+1+2+1+1>:
On considère le problème de décision PARTITION3 suivant :

- **Description :** un ensemble d'entiers
- **Question :** Peut-on partitionner l'ensemble en trois sous-ensembles de même somme ?

Le but de l'exercice est de trouver un algorithme polynômial de validation pour le problème PARTITION3 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 PARTITION3. Expliquez
2. Expliquez la notion de certificat d'une instance du problème PARTITION3. Donnez une structure de données permettant la représentation d'un tel certificat. Expliquez
3. Donnez un algorithme de validation pour le problème PARTITION3, que vous appellerez validation_p3. 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_p3 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 PARTITION3 est un ensemble de n entiers que nous représentons par une structure de données (I,n), I consistant en un tableau d'entiers et n son nombre d'éléments.
2. Un certificat d'une instance (I,n) du problème PARTITION3 est un tableau c de n entiers appartenant à {1,2,3} : pour tout i, si c[i]=1, l'élément I[i] fait partie du premier sous-ensemble de la partition ; si c[i]=2, il appartient au deuxième sous-ensemble de la partition ; si enfin c[i]=3, il appartient au troisième sous-ensemble de la partition.
3. nous donnons ci-après, sous forme d'une fonction booléenne, un algorithme de validation validation_p3 pour le problème PARTITION3 : l'algorithme aura comme arguments un certificat c et une instance (I,n) de PARTITION3 :

```
Booléen validation_p3(c,I,n)
début
    Somme1=0 ;somme2=0 ;somme3=0
    pour i=1 à n faire
        si c[i]=1 alors somme1=somme1+I[i]
        sinon  si c[i]=2 alors somme2=somme2+I[i]

               sinon somme3=somme3+I[i]

               finsi

        finsi

    fait

    si somme1=somme2 et somme1=somme3 alors retourner VRAI sinon retourner FAUX finsi
fin
```

4. Le nombre T(n) d'opérations élémentaires du pire cas de l'algorithme de validation est clairement un polynôme de degré 1 en n.

5. T(n) polynôme de degré 1 donc $T(n)=\Theta(n)$: l'algorithme de validation est bien polynômial, et le problème PARTITION3 appartient donc à la classe NP.