Corrigé examen de fin de semestre

Exercice1 (6 points)

a- Montrer que : $t(n) = log[(n+1)!]\ est\ en\ O(n * logn)$

b- Résoudre l'équation :

$$\begin{cases} t(n) = 2 * t(n/2) + n^3 & si\ n > 1 \\ t(1) = 1 \end{cases}$$

c- Supposons que $f(n) = \Theta(g(n))$. Peut-on affirmer que $2^{f(n)} = \Theta(2^{g(n)})$?

Si oui, démontrer le, si non donnez un contre exemple.

---

a- $log_2[(n+1)!]$ est en $O(n\ log_2 n)$

$$\begin{aligned} log_2[(n+1)!] &= log_2[(n+1) * n!] \\ &= log_2(n+1) + log_2(n!) \\ &\leq log_2(n+1) + log_2(n^n) \qquad \textbf{2POINTS} \\ &= log_2(n+1) + n\ log_2 n \\ &\leq n + 1 + n\ log_2 n \\ &\leq 3\ n\ log_2 n \qquad \forall n \geq 3 \end{aligned}$$

b- $t(n) = 2 * t(n/2) + n^3$

$$t(n) = 2 * (2 * t(n/2^2) + (n/2)^3) + n^3$$

$$= 2^2 * t(n/2^2) + 2 * \frac{n^3}{2^3} + n^3$$

$$= 2^2 * t(\frac{n}{2^2}) + \frac{n^3}{2^2} + n^3$$

$$= 2^2 * (2 * t(\frac{n}{2^3}) + \left(\frac{n}{2^2}\right)^3) + \frac{n^3}{2^2} + n^3 \qquad \textbf{2POINTS}$$

$$= 2^3 * t(\frac{n}{2^3}) + 2^2 \frac{n^3}{(2^2)^3} + \frac{n^3}{2^2} + n^3$$

$$= 2^3 * t(\frac{n}{2^3}) + \frac{n^3}{(2^2)^2} + \frac{n^3}{(2^2)^1} + \frac{n^3}{(2^2)^0}$$

…

$= 2^k * t(\frac{n}{2^k}) + n^3 (\frac{1}{(2^2)^{k-1}} + \frac{1}{(2^2)^{k-2}} + \dots + \frac{n^3}{(2^2)^2} + \frac{n^3}{(2^2)^1} + \frac{n^3}{(2^2)^0})$

$= 2^k * t(\frac{n}{2^k}) + n^3 \sum_{i=0}^{k-1} \frac{1}{(2^2)^i}$

$= 2^k * t(\frac{n}{2^k}) + n^3 * \frac{1}{1-\frac{1}{2^2}}$

$= 2^k * t(\frac{n}{2^k}) + n^3 * \frac{4}{3}$

On pose $n=2^k$

$t(n)= n*t(1) + n^3 * \frac{4}{3}$ avec $t(1)=1$

$= n+ \frac{4}{3} * n^3 = \theta(n^3)$

c- $2^{f(n)} = \Theta(2^{g(n)})$?

On constate que si $f(n) =2n= \Theta(n)$

Donc es-ce-que $2^{2n} = \Theta(2^n)$? *2POINTS*

$2^{2n}=2^n * 2^n$ ne peut pas être égal à $\Theta(2^n)$

Donc $2^{f(n)} \neq \Theta(2^{g(n)})$

Exercice2 (12 points)

1. Soit E un ensemble d'entiers $E = \{e_1, \dots, e_n\}$, représenté par un tableau.
   a. Donnez la complexité de l'algorithme de recherche d'une valeur donnée x dans le tableau. Justifiez votre réponse.

      Complexité de l'ordre de O(n) car le pire cas c'est quand la valeur n'existe pas et il faut comparer x à toute les valeurs du tableau. ***1 Pt***

   b. Soient deux ensembles d'entiers, de même taille, E1 et E2,
      - Ecrire l'algorithme qui retourne l'ensemble $E = E1 \cap E2$.

```
Action intersection(E/ E1, E2 :tableau ;E/ n :entier ;S/ E3 :tableau ; S/k :entier)
   Variables i, k :entiers ;
   Debut   k=1 ;// les valeurs sont uniques                    3Pts
      Pour i=1 à n faire
        Si (recherche(E1[i], E2)) alors E3[k]=E1[i] ; k=k+1 ; finsi ;
      Finpour ;
    Fin ;
```

Déterminer sa complexité : complexité de l'ordre de $O(nxn)=O(n^2)$ **1Pt**

2. On considère un ensemble d'entiers strictement positifs $E' = \{e'_1, \ldots, e'_n\}$ tel que $1 \leq e'_i \leq k$. Exemple k=20 et E = {1, 7, 11, 14, 17}

   E' est représenté par un tableau T de booléens comme suit :

   $$\begin{cases} T(e'_i) = 1 & si\ e'_i \in E' \\ T(e'_i) = 0 & sinon \end{cases}$$

a. Donnez la complexité de l'algorithme de recherche d'une valeur donnée *x* dans le tableau T, et justifiez votre réponse.

   Fonction recherche (E/ T :tableau ; E/ k, x :entier) :booléen

   Début retourner T[x] ; fin ; complexité O(1) ; l'accès est direct puisque x est l'indice dont le contenu peut être 1 ou 0 selon si x existe ou non. **1pt**

   Soient deux ensembles d'entiers, de même taille, $E'1$ et $E'2$,

   - Ecrire l'algorithme qui retourne l'ensemble $E' = E'1 \cap E'2$.

```
Action intersection(E/ E'1, E'2 :tableau ;E/ n :entier ;S/ E'3 :tableau)      2pts
    Variables i, k : entiers ;
    Debut   // les valeurs sont uniques
        Pour i=1 à n faire
            Si (E'1[i] et E'2[i])) alors E'3[i]=1
            Sinon E'3[i]=0 ; finsi ;
        Finpour ;
    Fin ;
```

   - Déterminer sa complexité : complexité O(n) **1pt**
   - Donner l'invariant de boucle de l'algorithme
     « A la fin de l'itération i, E'3[1..i]=E'1[1..i]∩ E'2[1..i] si E'1[i]=1 et E'2[i]=1 alors E'3[i]=1» **1pt**
   - et montrer sa terminaison la vérification si E'1[i] et E'2[i] est une opération finie aussi l'instruction d'affectation est une opération finie, la boucle pour se termine puisqu'elle varie de 1 à n donc l'algorithme se termine. **1pt**

3. Quelle représentation allez-vous préférer ? On va préférer la deuxième représentation si le nombre d'éléments de l'ensemble est assez petit car la recherche est constante O(1) par contre si le nombre d'élément est important donc l'espace mémoire sera trop grand dans la deuxième représentation donc on va préférer la première représentation. **1pt**

Exercice3 (2 points) **(0,5 ; 0,5 ; 0,5 ; 0,5)**

Donnez les définitions des classes suivantes : *P, NP et NP-Complet*

- La classe P, est formée des problèmes qui peuvent être résolus en temps polynomial. C'est-à-dire en O (n^k), où k est un entier donné.

- Classe NP : signifie « non déterministe polynomial »
Il est facile, pour une solution donnée, de vérifier par un algorithme déterministe en un temps polynomial si celle-ci répond au problème pour une instance donnée, mais que le nombre de solutions à tester pour résoudre le problème peut très bien être exponentiel par rapport à la taille de l'instance.

- Classe NP-complet : est composée des problèmes les plus difficiles de NP. Les problèmes NP-complets sont les problèmes de NP que personne ne sait résoudre plus vite qu'en temps exponentiel et réductible en un problème connu NP-complet.

b- Peut-on affirmer qu'un problème qui appartient à NP-Complet n'appartient pas à P ? Non : Personne n'a trouvé d'algorithme polynomial pour un problème NP-complet. En outre personne n'a pu prouver qu'il n'en existait pas.

$P \neq NP$ on sait que $P \sqsubseteq NP$ mais on ne sait pas si $P = NP$ ou $P \neq NP$. Ce problème est si important qu'il fait partie des 7 problèmes du millénaire, dont la résolution est primée 1 million de dollars par Clay Mathematic Institute.

Bon courage