Devoir 5 IFT2125-A-H19

Student: Qiang Ye (20139927)

Date: 8 février 2019

mailto: samuel.ducharme@umontreal.ca (mailto:samuel.ducharme@umontreal.ca)

Question

1.Vous êtes professeur du cours d'algorithmique et vous trouvez les réponses suivantes dans un examen. Corrigez-les. C'est-à-dire, trouvez toutes les erreurs et expliquez pourquoi ce sont des erreurs et quelles sont les bonnes réponses.

(1) $O(2^{n \lg n}) = O(2^{n \log_3 n})$ car

$$\lim_{n\to\infty} \frac{2^{n \lg n}}{2^{n \log_3 n}} = \lim_{n\to\infty} \frac{n \lg n}{n \log_3 n} = \lim_{n\to\infty} \frac{\lg n}{\log_3 n} = \frac{1}{\log_3 2}$$

et $\frac{1}{\lg 3}$ est une constante positive. On n'a qu'a utiliser le théoreme sur les limites: si $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}^{>0}$, alors $\Theta(f) = \Theta(g)$

Erreurs

- $O(2^{n \lg n}) = O(2^{n \log_3 n})$:

  Soit $f(n) = 2^{n \lg n}$, alors, $f(n) \in O(2^{n \lg n})$, mais $f(n) \notin O(2^{n \log_3 n})$. Une preuve détaillée est dans ce qui suit.
- $\lim_{n\to\infty} \frac{2^{n \lg n}}{2^{n \log_3 n}} = \lim_{n\to\infty} \frac{n \lg n}{n \log_3 n}$

  Il n'y a pas de règle comme celle-ci pour calculer la limite.

Réponse

$$\begin{aligned}
&\lim_{n\to\infty} \frac{2^{n \lg n}}{2^{n \log_3 n}} \\
&= \lim_{n\to\infty} \frac{2^{n \lg n}}{2^{n \lg n \log_3 2}} \\
&= \lim_{n\to\infty} \left(2^{n \lg n}\right)^{1-\log_3 2} \\
&= \lim_{n\to\infty} \left(2^{1-\log_3 2}\right)^{n \lg n} \\
&= +\infty \quad (2^{1-\log_3 2} > 1) \\
&\Rightarrow 2^{n \lg n} \in \Omega(2^{n \log_3 n}) \text{ et } 2^{n \lg n} \notin \Theta(2^{n \log_3 n}) \\
&\Rightarrow O(2^{n \lg n}) \supset O(2^{n \log_3 n})
\end{aligned}$$

---

(2) On sait que QuickSort fait, dans le pire des cas, $\sum_{i=1}^{n-1} = \frac{n(n-1)}{2} \in \Theta(n^2)$ comparaisons. Puisque $n^2 \in \Omega(n \lg n)$, on a que tout algorithme de tri par comparaison doit avoir sa complexité dans $\Omega(n \lg n)$.

Erreurs

- On ne peut pas conclure la limite inférieure de complexité (Ω) de tous les algorithmes de tri par comparaison, simplement en observant **un** algorithme de tri: QuickSort.

  C'est-à-dire, $\exists a \in A \wedge a \in \Omega(n \lg n)$ ne implique pas que $\forall a \in A, a \in \Omega(n \lg n)$

Réponse

On utilise le modèle d'arbre de décision binaire pour prouver que tous les algorithmes de tri, basés uniquement sur des comparaisons, ont la complexité de $\Omega(n \lg n)$ pour le pire des cas.

Suppose que les éléments à trier soient les nombres (distincts) 1 à n. Il doit y avoir $n!$ feuilles (une pour chacune des $n!$ permutations de n éléments). Arbre de hauteur h a au plus 2^h feuilles. h est les nombres de comparasion pour trier les n éléments

Alors,

$$2^h \geq n! \Rightarrow h \geq \log(n!) \Rightarrow h \in \Omega(log(n!))$$

puisque $\log(n!) \in \Theta(n \lg n)$, On a $\log(n!) \in \Omega(n \lg n)$

Finalement, on a

$$h \in \Omega(n \lg n)$$

C'est-à-dire, on a que tout algorithme de tri par comparaison doit avoir sa complexité dans $\Omega(n \lg n)$.

Question

2.Trouvez la solution exacte de la récurrence suivante pour $n = 2^{2^k}, k \in \mathbb{N}$.

$$T(n) = \begin{cases} 1 & n = 2 \\ 2T(\sqrt{n}) + \lg n & \text{sinon} \end{cases}$$

Réponse

n n'a aucune chance d'être égal à 2 si $n = 2^{2^k}$ et $k \in \mathbb{N}$. La valeur minimale de n devrait être 4. S'il est possible pour $n = 2$, il faut que $k \in \mathbb{Z}^{\geq 0}$.

On a déjà:

$$n = 2^{2^k} \tag{0}$$

Soit

$$S(k) = T(n) \tag{1}$$

On peut dire:

$$\begin{aligned} S(k) &= T(n) \\ &= T(2^{2^k}) \\ &= \begin{cases} 1 & k = 0 \\ 2T(\sqrt{2^{2^k}}) + \lg 2^{2^k} & \text{sinon} \end{cases} \\ &= \begin{cases} 1 & k = 0 \\ 2T(2^{2^{k-1}}) + 2^k & \text{sinon} \end{cases} \\ &= \begin{cases} 1 & k = 0 \\ 2S(k-1) + 2^k & \text{sinon} \end{cases} \end{aligned}$$

le polynôme caractéristique pour $S(k)$ est:

$$q(x) = (x-2)(x-2) = (x-2)^2 \tag{2}$$

Alors, pour $k > 1$,

$$S(k) = C_1 2^k + C_2 k 2^k \tag{3}$$

Puisque

- $S(0) = 1$, on a: $C_1 = 1$
- $S(1) = 2S(0) + 2 = 4$, on a: $C_2 = 1$

Alors,

$$S(k) = 2^k + k2^k \tag{4}$$

Combinant les formulaires (0), (1) et (4), pour $n > 2$ $(k > 1)$, on a:

$$T(n) = S(k) = \lg n + \lg(\lg n) \times \lg n$$

Lorsque $n = 2, \lg n + \lg(\lg n) \times \lg n = 1$, On peut dire, pour $n = 2^{2^k}, k \in \mathbb{Z}^{\geq 0}$,

$$T(n) = \lg n + \lg(\lg n) \times \lg n \tag{5}$$

Verification de $T(n)$ pour $n \geq 2$:

$$
\begin{aligned}
T(n) &= 2T(\sqrt{n}) + \lg n \\
&= 2(\lg \sqrt{n} + \lg(\lg \sqrt{n}) \times \lg \sqrt{n}) + \lg n \\
&= 2(\frac{1}{2}\lg n + \lg(\frac{1}{2}\lg n) \times \frac{1}{2}\lg n) + \lg n \\
&= \lg n + 2\Big(\lg \frac{1}{2} + \lg(\lg n)\Big) \times \frac{1}{2}\lg n + \lg n \\
&= \lg n + \Big(-1 + \lg(\lg n)\Big) \times \lg n + \lg n \\
&= \lg n + \lg(\lg n) \times \lg n
\end{aligned}
$$