POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

LOG1810

STRUCTURES DISCRÈTES

TD 6 : ALGORITHMES ET ANALYSE DE COMPLEXITÉ

A2024

SOLUTIONNAIRE

Exercice 1 :

Dites si les affirmations suivantes sont vraies ou fausses. Justifiez vos réponses.

a) $3n^2 + 5n$ est $O(n^3)$
Solution : **Vrai**
Effectivement n^3 est une borne supérieure de $3n^2 + 5n$ $(Deg(n^3) > Deg(3n^2 + 5n))$.

b) $nlog(n)$ est $O(n^2)$
Solution : **Vrai**
Effectivement n croit plus vite que $log(n)$, il s'en suit que n^2 croit plus vite que $nlog(n)$.

c) $log^2(n)$ est $\Omega\left(\sqrt{n}log(n)\right)$
Solution : **Faux**
$\sqrt{n}$ croit plus vite que $log(n)$, il s'en suit que $\sqrt{n}log(n)$ croit plus vite que $log^{2(n)}$.

d) $n^6 + n^2 + 2$ est $o(n^6)$
Solution : **Faux**
n^6 est un borne supérieur et **inférieure** de $n^6 + n^2 + 2$, $n^6 + n^2 + 2$ n'est donc pas $o(n^6)$.

Exercice 2 :

Donnez une évaluation du comportement asymptotique de chacune des fonctions suivantes, en utilisant le grand-O. Justifiez votre réponse.

a) $[n + 5 + \sqrt{n}\,]$

Solution :
$[n + 5]$ est $O(n)$
$[\sqrt{n}]$ est $O(\sqrt{n})$
$[n + 5 + \sqrt{n}]$ est donc $O(n)$

b) $[n + 5][\sqrt{n} + 8]$

Solution :
$[n + 5]$ est $O(n)$
$[\sqrt{n} + 5]$ est $O(\sqrt{n})$
$[n + 5][[\sqrt{n} + 8]]$ est donc $O(n\sqrt{n})$

c) $\left[\sqrt{n}+\mathrm{n}^{\frac{1}{3}}\right][\mathrm{n}!+\mathrm{n}^{12}]$

Solution :

$\left[\sqrt{n}+\mathrm{n}^{\frac{1}{3}}\right]$ est $O(\sqrt{n})$

$[\mathrm{n}!+n\text{^}12]$ est $O(n!)$

$\left[\sqrt{n}+\mathrm{n}^{\frac{1}{3}}\right][\mathrm{n}!+\mathrm{n}^{12}]$ est donc $O(\sqrt{n}n!)$

d) $[log(n^n)+n][2^n+n!]$

Solution :

$log(n^n)=nlog(n)$

$[log(n^n)+n]$ est $O(nlog(n))$

$[2^n+n!]$ est $O(n!)$

$[log(n^n)+n][2^n+n!]$ est donc $O(n!\,\mathrm{nlog(n)})$

e) $\left[n+log\left(\frac{12n^7}{18n^2}\right)+n^2\right]\left[[17n^2+2n^n]+[n!+n^8]\right]$

Solution :

$log\left(\frac{12n^7}{18n^2}\right)=log\left(\frac{12}{18}\right)+log(n^5)=log\left(\frac{12}{18}\right)+5log(n)$

Alors, $\left[n+log\left(\frac{12n^7}{18n^2}\right)+n^2\right]$ est $O(n^2)$

$[17n^2+2n^n]$ est $O(n^n)$

$[n!+n^8]$ est $O(n!)$

$[17n^2+2n^n]+[n!+n^8]$ est donc $O(n^n)$

Finalement $\left[n+log\left(\frac{12n^7}{18n^2}\right)+n^2\right]\left[[17n^2+2n^n]+[n!+n^8]\right]$ est $O(n^{n+2})$

f) $[n+5]^{99}[log^2(n)+nlog(n)]$

Solution :

$[n+5]$ est $O(\mathrm{n})$

Or, comme l'exponentiation est une répétition de multiplication :

$[n+5]^{99}$ est $O((n)^{99})\Rightarrow O(n^{99})$

$[log^2(n)+nlog(n)]$ est $O(nlog(n))$

Ainsi, $[n+5]^{99}[log^2(n)+nlog(n)]$ est $O(n^{100}log(n))$

Exercice 3 :

Ordonnez les fonctions suivantes de sorte que chaque fonction soit grand-O de la suivante.

$$log\left(n^{n^8}\right), \sqrt{n}^{18}, log(17n^3), n^{n!}, n^3 + 17n^2 + 2, 10^6 n, (n!)^3, (n-1)^{17}$$

Solution :

$$log(17n^3)\ ;\ 10^6 n\ ;\ n^3 + 17n^2 + 2\ ;\ log\left(n^{n^8}\right);\ \sqrt{n}^{18}\ ;\ (n-1)^{17}\ ;\ (n!)^3\ ;\ n^{n!}$$

Exercice 4 :

a) Montrez que $2x^4 + 3x^3 + 5$ est $\Theta(x^4)$.

Solution :

Nous avons les deux fonctions suivantes :

- $f(x) = 2x^4 + 3x^3 + 5$
- $g(x) = x^4$

Montrons dans un premier temps que $2x^4 + 3x^3 + 5$ est $\Omega(x^4)$, i.e. trouvons deux constantes tel que :

$$C_1|g(x)| \leq |f(x)| \qquad x > k_1$$

Ainsi, nous avons :

$$\begin{array}{ll} x^4 \leq 2x^4 & \text{pour } x>0 \\ x^4 \leq 2x^4 + 3x^3 + 5 & \text{car } 3x^3 + 5 > 0 \text{ pour } x>0 \\ |x^4| \leq |2x^4 + 3x^3 + 5| & \text{pour } x > 0 \end{array}$$

Ainsi, nous trouvons $C_1 = 1$ et $k_1 = 0$. Nous avons donc montré que $2x^4 + 3x^3 + 5$ est $\Omega(x^4)$. Montrons maintenant que $2x^4 + 3x^3 + 5$ est $O(x^4)$, i.e. trouvons deux constantes tel que :

$$|g(x)| \leq C_2|f(x)| x > k_2$$

Ainsi, nous avons :

$$\begin{array}{ll} 2x^4 + 3x^3 + 5 \leq 2x^4 + 3x^4 + 5x^4 & \text{car } 1 \leq x^3 \leq x^4 \quad \text{pour } x>0 \\ 2x^4 + 3x^3 + 5 \leq 10x^4 & \text{pour } x>0 \\ |2x^4 + 3x^3 + 5| \leq 10|x^4| & \text{pour } x>0 \end{array}$$

Ainsi, nous trouvons $C_2 = 10$ et $k_2 = 0$. Nous avons donc montré que $2x^4 + 3x^3 + 5$ est $O(x^4)$. Ainsi, comme nous avons montré que $2x^4 + 3x^3 + 5$ est $O(x^4)$ et $\Omega(x^4)$, $2x^4 + 3x^3 + 5$ est $\Theta(x^4)$.

CQFD

b) Montrez que 4^n n'est pas $O(2^n)$.

Solution :

Raisonnons par l'absurde.
Supposons que 4^n est $O(2^n)$ et montrons que nous arrivons à une contradiction.

Par définition, pour que 4^n est $O(2^n)$, il existe deux constantes C et k telles que pour $n > k$, $4^n \leq \mathrm{C}2^n$. Nous avons donc :

$$
\begin{aligned}
&4^n \leq \mathrm{C}2^n \\
&\Rightarrow \frac{4^n}{2^n} \leq C \\
&\Rightarrow \left(\frac{4}{2}\right)^n \leq C \\
&2^n \leq C
\end{aligned}
$$

Or, peu importe la valeur de C, il n'est pas possible $2^n \leq C$ pour tout n > k, car n peut être arbitrairement grand. Ainsi, nous arrivons a une contradiction.

Par conséquent, notre supposition comme quoi 4^n est $O(2^n)$ est fausse. 4^n n'est donc pas $O(2^n)$.

CQFD

Exercice 5 :

À la suite des nombreuses plaintes de la lenteur du site de La Ruche, l'équipe du Service stages et emplois a décidé de changer l'algorithme de recherche de mandat utilisé précédemment. Pour n mandats présents dans la base de données de La Ruche, l'algorithme effectuait :

$$f(n) = \sum_{i=0}^{n} i^2 - 2i \text{ opérations}$$

Le nouvel algorithme divise le nombre d'opérations par une fonction proportionnelle au cube du nombre de mandats, plus précisément, le nouveau nombre d'opérations est :

$$g(n) = \frac{6f(n)}{n^3 + 1}$$

a) Déterminez la complexité Θ de l'ancien algorithme.

Solution :

Nous avons :

$$\sum_{i=0}^{n} i^2 - 2i = \sum_{i=0}^{n} i^2 - 2\sum_{i=0}^{n} i$$

En sachant que :

$$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \quad et \quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Nous avons:

$$\sum_{i=0}^{n} i^2 - 2\sum_{i=0}^{n} i = \frac{n(n+1)(2n+1)}{6} - n(n+1)$$

Nous arrivons donc à l'expression suivante :

$$\sum_{i=0}^{n} i^2 - 2\sum_{i=0}^{n} i = \frac{n(n+1)(2n+1)}{6} - n(n+1) = \frac{2n^3 - 3n^2 - 5n}{6}$$

Nous pouvons maintenons évaluer la complexité de l'algorithme. Montrons que l'algorithme est $\Theta(n^3)$:

Montrons d'abord que $\frac{2n^3-3n^2-5n}{6} \in O(n^3)$
Soit, $2n^3 \in O(n^3)$, $-3n^2 \in O(n^2)$ et $-5n \in O(n)$.
Alors, $2n^3 - 3n^2 - 5n \ \in O(max(n^3, n^2, n) \ \Rightarrow O(n^3)$
Finalement, comme $\frac{1}{6} \in O(1)$, $\frac{2n^3-3n^2-5n}{6} \in O(n^3)$ (multiplication)

D'autre part, montrons que $n^3 \in O\left(\frac{2n^3-3n^2-5n}{6}\right)$
Cherchons un C et k tel que $n^3 \leq C\,\frac{2n^3-3n^2-5n}{6}$ pour $n > k$

En prenant C=6, nous avons $n^3 \leq 2n^3 - 3n^2 - 5n$ pour $n > 4$*

* $n^3 = 2n^3 - 3n^2 - 5n \ \Rightarrow n(n^2 - 3n - 5) = 0 \ \Rightarrow \mathrm{n(n - 4{,}193)(n + 1{,}193)}$

Comme les deux fonctions sont croissantes pour $n > 1$ et qu'initialement n^3 est plus grand que $2n^3 - 3n^2 - 5n$. Il s'en suit qu'après leur point de rencontre à 4,193 : $n^3 \leq 2n^3 - 3n^2 - 5n$.

Nous venons ainsi de montrer que $n^3 \in O\left(\frac{2n^3-3n^2-5n}{6}\right)$

Ainsi comme $\frac{2n^3-3n^2-5n}{6} \in O(n^3)$ et $n^3 \in O\left(\frac{2n^3-3n^2-5n}{6}\right)$, nous avons montré que $\frac{2n^3-3n^2-5n}{6} \in \Theta(n^3)$. Ainsi il en découle que :

$$f(n) \in \Theta(n^3)$$

b) Déterminez la complexité Θ du nouvel algorithme.
Solution :

Montrons que le nouvel algorithme est $\Theta(1)$:
Montrons d'abord que $\frac{2n^3-3n^2-5n}{(n^3+1)} \in O(1)$:

$$2n^3 - 3n^2 - 5n \leq 2n^3 \qquad \text{pour n>0}$$
$$2n^3 - 3n^2 - 5n \leq 2n^3 + 2 \qquad \text{pour n>0}$$
$$\frac{2n^3-3n^2-5n}{(n^3+1)} \leq 2 \qquad \text{pour n>0}$$

Ainsi, nous trouvons $C_1 = 1$ et $k_1 = 0$. Nous avons donc montré que $\frac{2n^3-3n^2-5n}{(n^3+1)} \in O(1)$.

Montrons ensuite que $\frac{2n^3-3n^2-5n}{(n^3+1)}$ est $\Omega(1)$. Nous avons d'abord :

$$2n^3 - 3n^2 - 5n = n(2n^2 - 3n - 5) = n(n+1)(n-2{,}5)$$

Ainsi, nous avons :

$$0 \leq 2n^3 - 3n^2 - 5n \qquad \text{pour n>3}$$
$$0 \leq \frac{2n^3-3n^2-5n}{(n^3+1)} \qquad \text{pour n>3}$$

Ainsi, nous trouvons $C_2 = 0$ et $k_1 = 3$. Nous avons donc montré que $\frac{2n^3-3n^2-5n}{(n^3+1)} \in \Omega(1)$.

c) Sachant qu'il y a actuellement 120 mandats enregistrés sur la base de données de La Ruche et qu'une opération demande $1\mu s$ à s'exécuter sur les serveurs de La Ruche, combien de temps a été gagné pour cette recherche avec le nouvel algorithme. Refaites le calcul pour 600 mandats.
Solution :

Pour n=120, nous avons $f(120) = 568700\ opérations$ et $g(120) \approx 2\ opérations$.

Ainsi, le nouvel algorithme fait gagner : $t_g = 1\mu s \cdot f(120) - 1\mu s \cdot g(120) \approx 0{,}56\ s$

Pour n=600, nous avons $f(120) = 71819500\ opérations$ et $g(120) \approx 2\ opérations$.

Ainsi, le nouvel algorithme fait gagner : $t_g = 1\mu s \cdot f(600) - 1\mu s \cdot g(600) \approx 72\ s$

Exercice 6 :

Soit l'algorithme suivant :

```
1.  for i := ⌊n⌋ to n
2.      a := n-i
```

Déterminez le nombre de soustractions qui seront exécutées pour cet algorithme (sachant que la boucle s'exécute jusqu'à n inclusivement) et en déduire sa complexité temporelle (considérez seulement les opérations de soustraction).

Solution :

La boucle s'exécute :

$$n - \left\lfloor \frac{n}{2} \right\rfloor + 1$$

De plus, nous savons que $\left\lfloor \frac{n}{2} \right\rfloor = \frac{n}{2}$ quand n est pair et $\left\lfloor \frac{n}{2} \right\rfloor = \frac{n-1}{2}$ quand n est impair.

Ainsi quand n est pair, nous avons :

$$n - \left\lfloor \frac{n}{2} \right\rfloor + 1 = n - \frac{n}{2} + 1 = \frac{n+2}{2}$$

Quand n est impair :

$$n - \left\lfloor \frac{n}{2} \right\rfloor + 1 = n - \frac{n-1}{2} + 1 = \frac{n+3}{2}$$

Dans les deux cas, nous avons $\frac{n+2}{2}$ est $\Theta(n)$ et $\frac{n+3}{2}$ est $\Theta(n)$. Ainsi peut importe la parité de n, l'algorithme est $\Theta(n)$.

Exercice 7 :

L'algorithme *tri insertion* permet de trier une liste d'éléments. Voici un pseudocode de l'algorithme *tri insertion* qui prend en entrée une liste a[0], a[1], …, a[n] :

```
for k := 1 to n
   x := a[k]
   j := k-1
   while(j ≠-1 &&* a[j]>x)
        a[j+1] := a[j]
        j := j-1
   a[j+1] := x
```

*L'opérateur && représente un **et** logique.

a) Donnez l'état de la liste a[]={6,3,5,7,2} après chaque passage de la boucle for :

	a[0]	a[1]	a[2]	a[3]	a[4]
Initialisation	**6**	3	5	7	2
k=1	**3**	6	5	7	2
k=2	3	**5**	6	7	2
k=3	3	5	6	**7**	2
k=4	**2**	3	5	6	7

b) Calculez le nombre de comparaisons effectuées dans la boucle while dans le pire cas. Déduisez la complexité temporelle de l'algorithme dans le pire cas.

Solution :

À chaque itération tentée de la boucle while, deux comparaisons sont effectuées : l'une pour vérifier $si\ j \neq$ -1, et l'autre pour vérifier si $x \leq a[j]$. Pendant que $a[k]$ est placé par rapport à $a[0], a[1], \dots, a[k-1]$, le nombre maximal d'itérations tentées de la boucle while est k. Cela se produit lorsque $a[k]$ est inférieur à tous les éléments; lors de la k-ème itération tentée, la condition de la boucle while n'est pas satisfaite car $j = -1$. Ainsi, le nombre maximum d'itération pour le while pour un k quelconque est 2k Comme la boucle for itère de $k = 1$ à $n - 1$ alors il s'ensuit que le nombre maximal de comparaisons est :

$$2 \cdot 1 + 1 + 2 \cdot 2 + 1 + \cdots + 2 \cdot (n-1) + 1 = 2\big(1 + 2 + .. (n-1)\big)$$

$$\Rightarrow 2\frac{n(n-1)}{2} = n^2 - n$$

Ainsi, comme $n^2 - n$ est un polynôme de degré 2 alors $\Theta(n^2)$.