UNIVERSITE GRENOBLE ALPES **M1-info**

M1-Complexité algorithmique de problèmes **Année 2016-17**

TD04 — "Ordres de grandeur' — énoncé

Rappels de cours et notations

On s'intéresse à des fonctions des entiers vers les réels positifs et pas seulement vers les entiers, pour pouvoir parler de coût moyen pour une certaine taille n de l'entrée d'un programme.
On rappelle les définitions suivantes :
$O(f) = \{\ g \mid (\exists k_1 > 0)\ (\exists n_1 \geq 0)\ (\forall n \geq n_1)\ [|g(n)| \leq k_1.f(n)\ ]\ \}$
$\Omega(f) = \{\ g \mid (\exists k_2 > 0)\ (\exists n_2 \geq 0)\ (\forall n \geq n_2)\ [\ g(n) \geq k_2.f(n)\ ]\ \}$
$\Theta(f) = O(f) \cap \Omega(f) = \{\ g \mid (\exists k_1 > 0)\ (\exists k_2 > 0)\ (\exists n_0 \geq 0)\ (\forall n \geq n_0)\ [k_2.f(n) \leq g(n) \leq k_1.f(n)\ ]\ \}$
On pourra utiliser les notations suivantes :
$O(f(n))$ pour $O(f)$, $\Omega(f(n))$ pour $\Omega(f)$, $\Theta(f(n))$ pour $\Theta(f)$.
Quand $g \in O(f)$ (respectivement $g \in \Omega(f)$, $g \in \Theta(f)$) on pourra noter :
$g(n) \in O(f(n))$ ou même $g(n) = O(f(n))$
(respectivement $g(n) \in \Omega(f(n))$ ou $g(n) = \Omega(f(n))$, $g(n) \in \Theta(f(n))$ ou $g(n) = \Theta(f(n))$)

On dira :
"g est grand O de f" (ou "$g(n)$ est grand O de $f(n)$"), ou encore "g (ou $g(n)$) a un ordre de grandeur inférieur à f (ou $f(n)$)."
"g est *Omega* de f" (ou "$g(n)$ est *Omega* de $f(n)$"), ou encore "g (ou $g(n)$) a un ordre de grandeur supérieur à f (ou $f(n)$)."
"g est *Theta* de f" (ou "$g(n)$ est *Theta* de $f(n)$"), ou encore "g (ou $g(n)$) a un ordre de grandeur égal à f (ou $f(n)$)."

On pourra ainsi écrire : $5.n^2 + 3 = O(n^2)$ (et aussi $5.n^2 + 3 = \Theta(n^2)$).

On utilise les fonctions de référence suivantes (supposer $n > 0$ pour les logarithmes) :

$$\lambda n[log_2 n], \lambda n[n], \lambda n[n\ log_2 n], \lambda n[n^2], \lambda n[n^3], \ldots, \lambda n[n^k], \lambda n[2^n], \lambda n[n!], \lambda n[n^n]$$

pour définir les ordres de grandeur (supérieurs, inférieurs ou exacts) des fonctions de complexité, qui déterminent le nombre maximum (ou le nombre *moyen* maximum, qui peut ne pas être entier) d'instructions élémentaires significatives effectuées par un algorithme en fonction de la taille de l'entrée, et qui sont donc des fonctions positives de $\mathbb{N}$ dans $\mathbb{R}_{>0}$ [1] (les réels strictement positifs).
Ces ordres de grandeur sont notés :

$$\Theta(log_2 n), \Theta(n), \Theta(nlog_2 n), \Theta(n^2), \Theta(n^3), \ldots, \Theta(n^k), \Theta(2^n), \Theta(n!), \Theta(n^n)$$

On utilise les notations analogues pour O et Ω.

---

1. On conseille d'utiliser cette notation de préférence à la classique $\mathbb{R}^{+^*}$ ou $\mathbb{R}^{+*}$ pour éviter les confusions avec les suites finies de suites finies non vides de réels.

Questions

1. Répondre aux questions suivantes en faisant référence aux définitions.
   (a) Montrer que $f1$ définie par $f1(n) = 3n^2 + 5n \dot{-} 1$ est dans $O(n^2)$ et que $f1$ est dans $\Omega(n^2)$. Conclusion ?
   (b) Montrer que $f2$ définie par $f2(n) = 2^K.n$ est dans $O(n)$ [K constante].
   (c) Soit f une fonction positive qui est dans $O(n^3)$ et g une fonction positive qui est dans $O(n)$. Que peut-on dire de :
   $f + g$
   $f \times g$
   f/g ?
   [évidemment le quotient est un "piège"...].
   (d) Montrer que log_a est dans $O(log_b(n))$ pour tout a et tout b (avec $(a, b) > 1$). Conclusion ?
   (e) Montrer que $n.2^n$ est $O(3^n)$, et que $n^K.2^n$ est $O(3^n)$ [K constante].
   [On admettra que $log\ n$ est d'ordre inférieur à n^a pour tout $a > 0$ et que e^n est d'ordre supérieur à n^a pour tout a.]
   (f) Montrer que si $f(n)$ est $O(g(n))$, alors $log(f(n))$ est $O(log(g(n)))$.
   Ex : $f(n) = n^K\ 2^n$ et $g(n) = 3^n$.
   (g) Montrer que si $f(n)$ est $O(g(n))$, il n'est pas toujours vrai que $e^{f(n)}$ est $O(e^{g(n)})$.
   Ex : $f(n) = n\ ln\ 3$ et $g(n) = n\ ln\ 2$.
2. (a) S'il existe
   $$\begin{array}{llll} & n_0 \text{ tel que} & f(n) \leq g(n) & \text{pour tout } n \geq n_0 \\ \text{et} & n_1 \text{ tel que} & g(n) \leq h(n) & \text{pour tout } n \geq n_1 \end{array},$$
   peut-on en déduire qu'il existe n_2 tel que $f(n) \leq h(n)$ pour tout $n \geq n_2$?
   (b) Soient f et g deux fonctions positives. Si f est dans $\Omega(g)$, peut-on en déduire que g est dand $O(f)$?
   (c) S'il existe n_0 tel que $f(n) \leq g(n)$ pour tout $n \geq n_0$
   et s'il n'existe pas n_1 tel que $g(n) \leq h(n)$ pour tout $n \geq n_1$,
   peut-on en déduire qu'il n'existe pas n_2 tel que $f(n) \leq h(n)$ pour tout $n \geq n_2$?
   (d) Soient f et g deux fonctions positives telles que $(\forall n < n_0)[f(n) \leq g(n)]$ et $(\forall n \geq n_0)[f(n) \geq g(n)]$, et soit $h(n) = f(n)$ si $n < n_0$, $g(n)$ sinon.
   Les ordres de grandeur de f, g et h peuvent-ils être comparés ?
3. (a) Soit $f(n)$ = le plus grand nombre premier s'écrivant avec au plus n chiffres décimaux.
   Sachant que $\Pi(x)$ [le nombre de nombre premiers inférieurs à x] est une fonction dont l'ordre de grandeur est $x/logx$ (théorème de Tchebicheff), que peut-on dire de l'ordre de grandeur de $f(n)$?
   (b) Soit $g(n)$ = le nombre de zéros dans l'écriture décimale de $3n+1$. Que peut-on dire de l'ordre de grandeur de $g(n)$?
   (c) Quel est l'ordre de grandeur du nombre de 0 dans la représentation binaire du plus grand nombre premier s'écrivant avec au plus n chiffres décimaux ?