Algorithmique

TD2. Semaine du 15/10/2001

Ordres de grandeur

Rappels

La notation $f(x) = \Theta(g(x))$ signifie qu'il existe $c_1, c_2 > 0$ tels que $0 \leq c_1 \times g(x) \leq f(x) \leq c_2 \times g(x)$ dès que x est suffisamment grand. Noter qu'elle ne s'applique qu'à des fonctions positives à partir d'un certain rang. On a :

1. $f(x) = \Theta(a \times f(x))$ pour tout réel $a > 0$.
2. Θ est une relation d'équivalence, c'est-à-dire :
   - $f(x) = \Theta(f(x))$,
   - si $f(x) = \Theta(g(x))$ alors $g(x) = \Theta(f(x))$,
   - si $f(x) = \Theta(g(x))$ et $g(x) = \Theta(h(x))$, alors $f(x) = \Theta(h(x))$,
3. si $\lim f(x) = +\infty$ quand $x \to +\infty$, alors $f(x) + b = \Theta(f(x))$ pour toute constante réelle b.
4. si $f(x) = \Theta(g(x))$, alors $f(x) + h(x) = \Theta(g(x) + h(x))$.
5. si $f(x) = \Theta(g(x))$ et $h(x) = \Theta(k(x))$, alors $f(x) + h(x) = \Theta(g(x) + k(x))$.

Exercice 1

Démontrer les propriétés (1) à (5). Pour (3), distinguer les cas $b > 0$ et $b < 0$. Pour (4), on supposera que pour $d_1, d_2 > 0$ fixés, on a

$$0 \leq d_1 \times g(x) \leq f(x) \leq d_2 \times g(x)$$

dès que x est suffisamment grand, et on distinguera d'une part les cas $d_1 > 1$ et $d_1 < 1$, d'autre part les cas $d_2 < 1$ et $d_2 > 1$.

Exercice 2

Soient n, j entiers tels que $n > j \geq 0$.

1. Démontrer que pour tout $a \geq 0$, on a $x^n = \Theta(x^n + a\,x^j)$.
2. Démontrer que pour tout $a \leq 0$, on a $x^n = \Theta(x^n + a\,x^j)$.
3. En déduire que si $P(x) = a_n\,x^n + a_{n-1}\,x^{n-1} + \cdots + a_0$ avec $a_n > 0$, on a $P(x) = \Theta(x^n)$. On raisonnera par récurrence sur n.
4. En déduire pour tout réel a, on a $(x + a)^n = \Theta(x^n)$.

Exercice 3

Classer les fonctions suivantes suivant la relation d'équivalence Θ.

- $f_1(n) = n$;
- $f_2(n) = 2^n$;
- $f_3(n) = n \log_2(n)$;
- $f_4(n) = n - n^3 + 7n^5$;
- $f_5(n) = n^2 + \log_2(n)$;
- $f_6(n) = n^2$;
- $f_7(n) = \log_2(n)$;
- $f_8(n) = n^3$;
- $f_9(n) = \sqrt{n} + \log_2(n)$;
- $f_{10}(n) = (\log_2(n))^2$;
- $f_{11}(n) = \ln(n)$;
- $f_{12}(n) = 3^n + n^2$;

Exercice 4

1. Démontrer par récurrence sur n que $\Sigma_{i=0}^{n} 2^i = 2^{n+1} - 1$.
2. On considère la méthode `test` prenant en argument une liste finie de 0 et de 1, et renvoyant le premier élément de cette suite contenant un 0 s'il existe.

   Les positions dans une liste L de longueur n seront numérotées de 1 à n. On supposera que lorsque la première occurence de 0 dans L est à la position i, `test`(L) prend le temps i. D'autre part, si L est une suite de 1 de longueur n, `test`(L) prend le temps n. Donner la complexité en moyenne de `test`, en fonction de la longueur n de L.