SDA2 - TD2
complexité et récursivité

Jean-Michel Dischler, Pascal Schreck

Complexité

Pour les fonctions f et g suivantes, vérifier les affirmations : $f = O(g)$, $f = \Omega(g)$ et $f = \Theta(g)$.

1. $f = n^2$, $g = n$
2. $f = \sqrt{n}$, $g = n$
3. $f = 3n^2 + \sqrt{n}$, $g = n^2$

Nombres parfaits

Un nombre est dit **parfait** s'il est égal à la somme de ses diviseurs (lui-même exclus). Exemple : $6 = 3 + 2 + 1$, c'est un nombre parfait.

1. Écrivez un algorithme itératif qui permet d'afficher tous les nombres parfaits entre 1 et n.
2. Calculer le coût T(n) en nombre de divisions. En déduire que la complexité de cet algorithme en nombre de divisions est en $\Theta(n^2)$.

Suite de Fibonacci

La suite de Fibonacci est définie comme suit :

$$\mathrm{Fib}(n) = \begin{cases} 1 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ \mathrm{Fib}(n-1) + \mathrm{Fib}(n-2) & \text{sinon.} \end{cases}$$

1. Écrivez un algorithme récursif calculant $\mathrm{Fib}(n)$.
2. Compter le nombre d'appels à la fonction. Montrez que la complexité (en nombre d'additions) de cet algorithme est en $\Omega(2^{\frac{n}{2}})$.
3. Écrire un algorithme récursif qui calcule, pour $n > 0$, le couple ($\textsc{Fibonacci}(n)$, $\textsc{Fibonacci}(n-1)$).
4. Utilisez l'algorithme précédent pour écrire un nouvel algorithme calculant $\textsc{Fibonacci}(n)$.
5. Qu'elle est la complexité (en nombre d'additions) de cet algorithme ?

Nombre de combinaisons

Le nombre de combinaison C_n^p est défini comme suit :

$$\mathrm{C}(n,p) = \begin{cases} 1 & \text{si } n = 0 \ \vee \ n = p \\ \mathrm{C}(n-1,p) + \mathrm{C}(n-1,p-1) & \text{sinon.} \end{cases}$$

1. Écrivez un algorithme récursif calculant $\mathrm{C}(n,p)$.
2. Compter le nombre d'appels à la fonction. Montrez que la complexité de cet algorithme est en $\theta(2^n)$.