Travaux Pratiques d'algorithmique n°4

Cours d'Informatique de Deuxième Année

—DEUG Sciences M3-MI—

Récursivité

Au TP précédent, nous avons introduit la notion de récursivité sur des exemples simples. Nous allons maintenant appliquer cette méthode pour résoudre des problèmes plus complexes. Vous devrez présenter clairement les algorithmes, ainsi que **le calcul de leur complexité**, que nous généraliserons la séance suivante.

▸ **Exercice 1.** **(Nombre de Stirling)** *On sait que les nombres de Stirling $S(n,k)$ de deuxième espèce, vérifient la récurrence suivante :*

$$S(n,k) = S(n-1,k-1) + k\,S(n-1,k) \qquad \text{pour } k > 0\ .$$

De plus :

$$\begin{cases} S(n,k) = 0 & \text{si } k > n\ , \\ S(n,k) = 0 & \text{si } k = 0\ , \\ S(n,k) = 0 & \text{si } n = 0\ , \\ S(n,k) = 1 & \text{si } n = 0 \text{ et } k = 0\ . \end{cases}$$

Ecrire une fonction récursive qui calcule $S(n,k)$.

▸ **Exercice 2.** *Ecrire une fonction* `Puissance` *en appliquant la définition suivante :*

$$b^0 = 1\ , \qquad b^n = b\,b^{n-1}\ .$$

Ecrire une fonction `PuissanceBis` *qui tient compte des relations suivantes :*

$$\begin{cases} b^n = (b^{n/2})^2 & \text{si } n \text{ est pair}\ , \\ b^n = b\,(b^{(n-1)/2})^2 & \text{si } n \text{ est impair}\ , \\ b^n = 1 & \text{si } n = 0\ . \end{cases}$$

Evaluer la complexité dans les deux cas.

▸ **Exercice 3.** *On définit les nombres de Fibonacci par :*

$$\begin{cases} F(0) = 0 \ , \qquad F(1) = 1 \ , \\ F(n) = F(n-1) + F(n-2) \quad si \ n \geq 2 \ . \end{cases}$$

1. *Calculer F(6) ;*
2. *programmer de façon récursive, la fonction* `int FiboR(int n)` *qui retourne le nième nombre de Fibonacci ;*
3. *quel est le défaut de cette méthode ?*
4. *programmer une nouvelle fonction* `int FiboIt(int n)` *qui calcule le nième nombre de Fibonnaci de façon itérative. Celle-ci utilisera un tableau auxiliaire qui stockera les calculs intermédiaires ;*
5. *évaluer la complexité en temps et en espace, de ce dernier algorithme.*
6. *quel est le défaut de cette méthode ?*
7. *finalement, programmer une nouvelle fonction* `int FiboItBis(int n)` *qui sera optimale en temps et en espace.*