Travaux Dirigés d'Algorithmique n°3

Cours d'Informatique de Deuxième Année

—Deug SSM M3-MI—

Récursivité, Complexité d'algorithme

Au TD précédent, nous avons introduit la notion de récursivité sur de nombreux exemples. Nous allons maintenant évaluer et comparer les complexités de différents algorithmes faisant intervenir -ou non- le concept de "récursivité".

▸ **Exercice 1. (Minimum et maximum d'un tableau)** *On désire calculer simultanément la valeur minimale et la valeur maximale d'un tableau d'entiers. On se propose de résoudre le problème de deux manières différentes :*

1. *en utilisant une méthode naïve ;*
2. *par dichotomie (on a donc un algorithme récursif).*

- *Ecrire une fonction* `MaxMinIterative` *qui implémente la méthode naïve et une fonction* `MaxMinRec` *qui implémente la méthode récursive.*
- *Montrer que le nombre de comparaisons effectuées par* `MaxMinIterative` *est au plus égal à* $2 \times (n-1)$.
- *De même, montrer que* `MaxMinRec` *effectue au plus* $\lceil \frac{3 \times n}{2} \rceil - 2$ *comparaisons.*

▸ **Exercice 2. (Suite de Fibonacci)** *Dans cet exercice, nous allons écrire deux fonctions (l'une récursive, l'autre itérative) pour calculer les termes de la suite de Fibonacci* $(\mathtt{F}_n)$. *Cette suite est définie par les relations :*

$$\mathtt{F}_n = \begin{cases} 0 & \textit{si } n = 0, \\ 1 & \textit{si } n = 1, \\ \mathtt{F}_{n-1} + \mathtt{F}_{n-2} & \textit{sinon.} \end{cases} \tag{1}$$

- *Ecrire une fonction* `FiboRec` *prenant en argument un entier* `n` *et qui calcule* $(\mathtt{F}_n)$ *en utilisant les relations (1). Evaluer la complexité en temps de cette fonction. Quelle est la hauteur de la pile de récursion ?*
- *Ecrire une fonction* `FiboIte` *qu prend en argument un entier* `n` *et qui calcule de manière itérative* $(\mathtt{F}_n)$. *Evaluer la complexité en temps et en espace de la fonction* `FiboIte`.

▸ **Exercice 3. (Une multiplication amusante)** *Dans cet exercice, nous cherchons à réaliser la multiplication de deux entiers* `n` *et* `m`*. Malheureusement, on ne dispose que de la division par deux et de l'addition... Cependant, on peut remarquer ceci :*

$$\begin{array}{rcll} 0 * m & = & 0 & \textit{pour tout } m, \\ n * m & = & (n/2) * (m+m) & \textit{si } n \textit{ est pair}, \\ n * m & = & (n-1) * m + m & \textit{si } n \textit{ est impair.} \end{array} \tag{2}$$

- *Déduire des relations (2) un algorithme pour réaliser la multiplication de deux entiers.*
- *En déduire une fonction* `Multiplication` *prenant en arguments deux entiers* `n` *et* `m` *et qui implémente l'algorithme précédant.*
- *Evaluer la complexité de la fonction* `Multiplication`*.*

▸ **Exercice 4. (Résolution d'équation)** *Soit l'équation :*

$$T(n) = \begin{cases} 1, & \textit{si } n = 1, \\ 2 \times T(\lfloor \frac{n}{2} \rfloor) + n, & \textit{sinon.} \end{cases}$$

Donnez une approximation de la complexité de la fonction T.