Travaux Dirigés d'algorithmique n°2

Cours d'informatique de deuxième année

—DEUG Sciences S3-MI—

Récursivité

L'objet du présent TD est l'introduction de la récursivité à travers des exemples simples. Nous parlerons aussi de complexité en temps et en espace, et de preuves d'algorithmes.

---

1 Appel de fonction

Le premier exercice a pour but de rappeler les mécanismes d'appels de fonction.

▸ **Exercice 1.** *Expliquer, en donnant le contenu de la pile d'exécution à chaque étape, ce qu'affiche le programme suivant :*

```
int h(int n) { printf("%d",n); return n; }

int g(int n)
{ if (n>0) printf("%d %d",h(n-1),n); else printf("%d",-n); return -n; }

void f(int n) { if (n>0) printf("%d %d",g(n-1),n); else printf("%d",n); }

int main(void) { f(3); return 0; }
```

Refaire la même chose avec cette fois-ci:

```
int h(int n) { f(n-1); printf("%d",n); return n; }
```

2 Récursivité

Nous abordons maintenant les fonctions *récursives*, i.e. qui sont utilisées (directement ou indirectement) dans leurs propres définitions.

▸ **Exercice 2.** *La fonction* **factorielle** *d'un entier positif ou nul n, notée $n!$, est définie par*

$$n! = \begin{cases} 1 & \text{si } n = 0, \\ (n-1)! * n & \text{si } n > 0. \end{cases}$$

Écrire une fonction C récursive (et une autre itérative) qui retourne la factorielle de son argument. Calculer le temps nécessaire à chacune des deux fonctions pour ce calcul. Calculer la place mémoire nécessaire à chacune des deux fonctions pour calculer $n!$. Des deux fonctions, quelle est la plus efficace ? Prouver que les fonctions écrites retournent bien $n!$.

▸ **Exercice 3. (une multiplication amusante)** *Dans cet exercice, nous cherchons à réaliser la multiplication de deux entiers* `n` *(≥ 0) et* `m`*. Malheureusement, nous ne dispose que de la division par deux et de l'addition ... Cependant, on peut remarquer ceci :*

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

- *Déduire des relations (1) un algorithme pour réaliser la multiplication de deux entiers* `n` *(≥ 0) et* `m`*;*
- *en déduire une fonction* `Multiplication` *prenant en arguments deux entiers* `n` *(≥ 0) et* `m` *et qui implémente l'algorithme précédent ;*
- *montrer que la fonction retourne le produit de ses arguments ;*
- *évaluer la complexité de la fonction* `Multiplication`*.*

▸ **Exercice 4.** *Le énième ($n \geq 0$) nombre de Fibonacci est défini récursivement par*

$$Fib_n = \begin{cases} Fib_{n-1} + Fib_{n-2} & \text{si } n > 1, \\ n & \text{si } n \in \{0, 1\}. \end{cases}$$

Écrire deux fonctions, une récursive et une itérative, qui prennent en argument un entier positif ou nul n et qui retournent le énième nombre de Fibonacci. Calculer les complexités (en temps) de chacune des deux fonctions. Laquelle est la plus rapide ?

▸ **Exercice 5.** *(on supposera que n est une puissance de 2.)*

1. *Que fait la fonction suivante ? (le démontrer),*

```
int f(int n)
{
  if (n == 0) return 1;
         else return(f(n/2)+n);
}
```

2. *Évaluer sa complexité.*