Travaux Dirigés n°2

Cours d'Informatique de Deuxième Année

—Deug SSM M3-MI—

Récursivité, Complexité d'algorithme

La récursivité est certainement la manière la plus "naturelle" de coder la plupart des fonctions mathématiques récursives (telle factorielle), mais a pour principal inconvénient d'obliger le compilateur à utiliser une pile pour mémoriser les calculs intermédiaires.
Par ailleurs, il est possible que le fait d'écrire les algorithmes instantanément tels qu'on les perçoit conduise à des temps d'exécution rédhibitoires. Il est alors important d'estimer avant l'exécution physique, non seulement le temps, mais aussi parfois l'espace nécessaires au programme : on parle de complexité d'algorithme.

---

1 Récursivité

▸ **Exercice 1. (somme des premiers entiers)** *Soit S_n la somme définie par :*

$$S_n = 1 + 2 + \cdots + (n-1) + n$$

- *trouver un algorithme récursif pour calculer S_n ;*
- *en déduire une fonction* `Somme` *prenant en argument un entier* `n` *et qui implémente cet algorithme ;*
- *évaluer la complexité en temps de cet algorithme.*

▸ **Exercice 2. (somme des chiffres d'un entier)** *On veut calculer la somme des chiffres d'un entier* `n`*. Par exemple, la somme des chiffres formant 3221 est 8.*

- *Trouver un algorithme récursif pour calculer la somme des chiffres formant un entier quelconque.*
- *En déduire une fonction* `SommeChiffre` *prenant en argument un entier* `n` *et qui implémente l'algorithme précédant.*

On considère la décomposition sur la base 10 d'un entier `n` *:*

$$n = \sum_{i=0}^{k} 10^i c_i.$$

Les critères de divisibilité suivants peuvent se déduire facilement (essayez pour voir...) :

$$n \equiv 0\ [3] \Leftrightarrow \sum_{i=0}^{k} c_i \equiv 0\ [3]\,,$$
$$n \equiv 0\ [11] \Leftrightarrow \sum_{i=0}^{k} (-1)^i c_i \equiv 0[11].$$

Modifier la fonction `Sommechiffre` *pour obtenir les fonctions* `NonMultde3` *et* `NonMultde11` *qui testent si un entier* `n` *n'est pas multiple de* 3 *et de* 11.

▸ **Exercice 3. (Image miroir d'une chaîne de caractères)** *Par définition, l'image miroir d'une chaîne de caractères* `ch` *est la chaîne obtenue par la lecture de droite à gauche de* `ch`*. Par exemple, l'image miroir de* `"abcdefgh"` *est* `"hgfedcba"`.

- *Trouver un algorithme récursif qui affiche l'image miroir d'une chaîne de caractères.*
- *En déduire une procédure* `AfficheMiroir`.
- *Donner la complexité en temps de la procédure* `AfficheMiroir`.

▸ **Exercice 4. (Recherche dans un tableau trié)** *Pour qu'un élément appartienne à un tableau trié en ordre croissant, il suffit de tester s'il est plus grand que l'élément du milieu du tableau. Si c'est le cas, il ne reste plus qu'à relancer la recherche dans la partie droite du tableau, sinon dans la partie gauche.*

- *Ecrire une fonction récursive* `Dichotomie` *qui prend en arguments un tableau d'entiers, sa taille, ainsi que l'élément à trouver et qui renvoie vrai si l'élément est dans le tableau.*
- *Evaluer la complexité en temps de la fonction* `Dichotomie`.

▸ **Exercice 5. (Tri fusion)** *Dans cet exercice, on se propose de réaliser un tri en ordre croissant d'un tableau d'entiers* `T`*. Nous utilisons l'algorithme de tri fusion basé sur le principe de "diviser pour régner". Soient* `G` *le sous-tableau constitué avec les* $\lfloor n/2 \rfloor$ *premiers éléments de* `T`*, et* `D` *le sous-tableau constitué avec les* $\lceil n/2 \rceil$ *derniers éléments de* `T`*. L'algorithme peut se décomposer en deux étapes. On trie récursivement les sous-tableaux* `G` *et* `D`*. Finalement, un interclassement de* `G` *et* `D` *(qui sont maintenant triés) fournit le tableau* `T` *trié.*

- *L'algorithme de tri fusion étant récursif, quel est son cas de base ?*
- *Donner un algorithme pour effectuer l'interclassement de deux tableaux triés* `T1` *et* `T2`*. Par exemple,* `[1,3,4,5,6]` *est le résultat de l'interclassement de* `[1,5]` *et* `[3,4,6]`.
- *En déduire une procédure* `TriFusion` *qui réalise un tri fusion sur un tableau d'entiers* `T`.
- *Donner la complexité de la procédure* `TriFusion`.